Cloud Benchmarking in Bare-metal, Virtualized, and Containerized Execution
Environments

by

Soheil Mazaheri, B.S.C.S.

A Thesis

In

COMPUTER SCIENCE

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCES

Approved

Dr. Yong Chen
Chair of Committee

Dr. Alan Sill

Dr. Noe Lopez-Benitez

Mark Sheridan
Dean of the Graduate School

May, 2015

Copyright 2015, Soheil Mazaheri

ACKNOWLEDGMENTS

I consider myself extremely fortunate to have met and worked with some remarkable people during my stay at Texas Tech University. While a brief note of thanks does not do justice to their impact on my life, I deeply appreciate their contributions.

I begin by thanking my thesis committee members, Dr. Yong Chen, Dr. Alan Sill, and Dr. Noe Lopez-Benitez for all of their guidance through this process.

First and foremost I wish to thank my advisor, Dr. Yong Chen. His guidance and advice during the course of my master studies have shaped my career. I have enjoyed working with him over the years and am grateful for his friendship and counsel. My experience under his advice has been nothing short of amazing. I have been given unique opportunities and taken advantage of them. This includes working at Cloud and Autonomic Computing (CAC) as the first researcher, working as system administrator for a research Hadoop cluster, and being in touch with companies. His strong support and confidence in my abilities were benefits not all students have a chance to experience.

I also extend my gratitude to Dr. Alan Sill, who gave me an opportunity to work closely with the High Performance Computing group at Texas Tech University and have been very generous by providing resources for my research and guiding me with his invaluable research advice and ideas. Graduate school can be a difficult, draining experience. I am proud to say my experience with him was intellectually exciting and fun, and has energized me to continue in academic research. Of course, this work would not have been possible without the support of him. I sincerely hope I continue to have opportunities to interact with Dr. Alan Sill for the rest of my research career.

I am grateful to Dr. James Abbott at the High Performance Computing Center, Texas Tech University. Dr. Abbott was incredibly helpful by providing me opportunities to work with HPC systems. He went out of his way to ensure that my work

with the HPC center was rewarding and productive. I have learnt lots of invaluable experiences from him and I am thankful for them.

The time I spent as a graduate assistant at the Cloud and Autonomic Computing Center (CAC) and High Performance Computing Center (HPCC) was exciting and memorable. I am indebted to the HPCC team, especially Hui (Amy) Wang, for her advice and support. Amy was instrumental in fostering my interest in data center subjects and I would like to thank her for her help and guidance.

I would like to thank the StackVelocity for their well-integrated hardware and software systems that made this research possible. I have enjoyed working with the StackVelocity team. I wish to thank Jose Bustamante and Kirk Lewis for their friendship and counsel. Moreover, I am grateful to the Aerospace Corporation for their software, Cloud Tester Benchmark Suite (CTBS). This research is supported by the Cloud and Autonomic Computing site at Texas Tech University via the StackVelocity membership contribution and the Aerospace Corporation technical partnerships.

Finally, I finish with Iran, where the most basic source of my life energy resides: my family. I have an amazing family, unique in many ways, and the stereotype of a perfect family in many others. Their support has been unconditional all these years; they have given up many things for me to be at Texas Tech University; they have cherished with me every great moment and supported me whenever I needed it. I thank my family, especially my sister, Sara Mazaheri, and my spouse, Sara Darvishan. Their love, affection, and faith have been a constant source of strength for me. None of this would have been possible without them.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
ABSTRACT	vii
LIST OF TABLES	viii
LIST OF FIGURES	xi
1 INTRODUCTION.....	1
1.1 Contributions.....	3
1.2 Organization.....	3
2 CLOUD COMPUTING	5
2.1 Definition	5
2.2 Essential Characteristics.....	6
2.3 Service Models.....	7
2.3.1 Infrastructure as a Service (IaaS)	7
2.3.2 Platform as a Service (PaaS).....	8
2.3.3 Software as a Service (SaaS).....	9
2.4 Deployment Models.....	10
2.5 Virtualization vs. Cloud Computing	13
3 VIRTUALIZATION.....	15
3.1 Virtual Machines.....	17
3.1.1 Hypervisor Type 1 (Bare-metal Hypervisor)	17
3.1.2 Hypervisor Type 2 (Hosted Hypervisor)	17
3.1.3 Kernel-Based Virtual Machine (KVM): Type 1 vs. Type 2	18
3.2 Linux Containers.....	19
3.2.1 Docker.....	21
3.2.2 Docker vs. LXC	22
3.2.3 Related Projects Based on Linux Containers	24
4 BENCHMARKING	27
4.1 Cloud Benchmarking	27
4.2 IOR (Interleave Or Random)	31
4.3 SHOC (Scalable Heterogeneous Computing).....	32
4.4 LINPACK (Linear Algebra Package).....	33
4.5 HPL	33
4.6 HPCC	33
4.7 Cloud Tester Benchmark Suite (CTBS).....	34
4.7.1 System Requirements.....	37
5 BENCHMARKING BARE-METAL EXECUTION ENVIRONMENT	38
5.1 System Configuration	38
5.2 HPCC Result	38

5.2.1	HPL	40
5.2.2	DGEMM and FFT	42
5.2.3	Random Access	43
5.2.4	STREAM and Communication Bandwidth	44
5.2.5	Communication Latency	46
5.3	IOR Result	47
5.3.1	Random IOPS	48
5.3.2	Throughput	52
6	BENCHMARKING CONTAINERIZED EXECUTION ENVIRONMENT	56
6.1	System Configuration	56
6.2	HPCC Result	56
6.2.1	HPL	57
6.2.2	DGEMM and FFT	59
6.2.3	Random Access	60
6.2.4	STREAM and Communication Bandwidth	61
6.2.5	Communication Latency	63
6.3	IOR Result	64
6.3.1	Random IOPS	64
6.3.2	Throughput	68
7	BENCHMARKING VIRTUALIZED EXECUTION ENVIRONMENT	72
7.1	System Configuration	72
7.2	HPCC Result	72
7.2.1	HPL	73
7.2.2	DGEMM and FFT	75
7.2.3	Random Access	76
7.2.4	STREAM and Communication Bandwidth	77
7.2.5	Communication Latency	79
7.3	IOR Result	80
7.3.1	Random IOPS	80
7.3.2	Throughput	84
8	ANALYSIS OF BENCHMARKING RESULTS ON DIFFERENT EXECUTION ENVIRONMENTS	88
8.1	HPCC Comparison	90
8.1.1	HPL	90
8.1.2	DGEMM and FFT	92
8.1.3	Random Access	94
8.1.4	STREAM and Communication Bandwidth	95
8.1.5	Communication Latency	97
8.2	IOR Result	98
8.2.1	Random IOPS	98
8.2.2	Throughput	102

8.3	Benchmark Analysis and Findings.....	105
8.3.1	HPCC Performance Comparison	105
8.3.2	IOR Performance Comparison	108
9	CONCLUSION AND FUTURE WORK	112
9.1	Conclusion	112
9.2	Future Work	113
	REFERENCES	115
A	HPCC OUTPUT.....	119
B	IOR OUTPUT	147
C	ENVIRONMENT INFORMATION	166

ABSTRACT

Datacenters usually use virtualization methods to increase their productivity and to reduce complexity for end users while allowing resources to be utilized more effectively. Cloud computing takes the use of those resources to the next level by delivering access to those components on-demand as a service, further reducing complexity, cost, and burden. Because virtualization involves increased abstraction, there is an inevitable trade-off between these advantages and a disadvantage of potentially losing performance. As a result, there has been a considerable amount of work to develop and to improve different virtualized environments that can isolate a workload from another. Virtual machines and Linux containers are the most well-known techniques that provide an isolated environment to allow applications to run independently. Each of these methods has its own advantages and disadvantages. This thesis research aims to compare the performance of systems in bare-metal, containerized (Linux containers), and virtualized (virtual machines) execution environments. This research also analyzes and understands the implications of the benchmarking results in these different execution environments. These benchmarking results, analyses, and insights discussed can provide a guidance to cloud computing researchers and developers in the process of designing cloud computing solutions, deploying cloud systems, and developing algorithms, programs, and applications on different cloud platforms.

LIST OF TABLES

4.1 Technical metrics or Key Performance Indicator (KPI) [26].....	28
4.2 Economic metrics or Key Performance Indicator (KPI) [26]	29
4.3 CTBS Technical Metrics [37]	36
5.1 HPCC Key Performance Metrics (KPMs) [38, p. 32]	39
5.2 HPL result on Bare-metal.....	41
5.3 DGEMM result on Bare-metal.....	42
5.4 StarFFT result on Bare-metal.....	42
5.5 StarRandomAccess_LCG result on Bare-metal.....	43
5.6 StarSTREAM result on Bare-metal	44
5.7 PingPongBandwidth result on Bare-metal	44
5.8 PingPongLatency result on Bare-metal.....	46
5.9 IOR Key Performance Metrics For Throughput & IOPS Operations [38, p. 34] ..	47
5.10 IOPS MPI-IO result on Bare-metal.....	48
5.11 IOPS POSIX Buffered IO result on Bare-metal	50
5.12 IOPS POSIX Direct IO result on Bare-metal.....	51
5.13 Throughput MPI-IO result on Bare-metal.....	52
5.14 Throughput POSIX Buffered IO result on Bare-metal	54
5.15 Throughput POSIX Direct IO result on Bare-metal	55
6.1 HPL result on Docker.....	58
6.2 DGEMM result on Docker.....	59
6.3 StarFFT result on Docker.....	59
6.4 StarRandomAccess_LCG result on Docker.....	60
6.5 StarSTREAM result on Docker	61

6.6 PingPongBandwidth result on Docker	61
6.7 PingPongLatency result on Docker.....	63
6.8 IOPS MPI-IO result on Docker.....	64
6.9 IOPS POSIX Buffered IO result on Docker	66
6.10 IOPS POSIX Direct IO result on Docker.....	67
6.11 Throughput MPI-IO result on Docker	68
6.12 Throughput POSIX Buffered IO result on Docker	70
6.13 Throughput POSIX Direct IO result on Docker	71
7.1 HPL result on KVM.....	74
7.2 DGEMM result on KVM	75
7.3 StarFFT result on KVM	75
7.4 StarRandomAccess_LCG result on KVM	76
7.5 StarSTREAM result on KVM.....	77
7.6 PingPongBandwidth result on KVM	77
7.7 PingPongLatency result on KVM	79
7.8 IOPS MPI-IO result on KVM	80
7.9 IOPS POSIX Buffered IO result on KVM.....	82
7.10 IOPS POSIX Direct IO result on KVM	83
7.11 Throughput MPI-IO result on KVM.....	84
7.12 Throughput POSIX Buffered IO result on KVM.....	86
7.13 Throughput POSIX Direct IO result on KVM.....	87
8.1 HPL result, Bare-metal vs. Docker vs. KVM	90
8.2 DGEMM result, Bare-metal vs. Docker vs. KVM.....	92
8.3 StarFFT result, Bare-metal vs. Docker vs. KVM	92

8.4 StarRandomAccess_LCG result, Bare-metal vs. Docker vs. KVM.....	94
8.5 StarSTREAM result, Bare-metal vs. Docker vs. KVM	95
8.6 PingPongBandwidth result, Bare-metal vs. Docker vs. KVM.....	95
8.7 PingPongLatency result, Bare-metal vs. Docker vs. KVM	97
8.8 IOPS MPI-IO result, Bare-metal vs. Docker vs. KVM.....	98
8.9 IOPS POSIX Buffered IO result, Bare-metal vs. Docker vs. KVM	100
8.10 IOPS POSIX Direct IO result, Bare-metal vs. Docker vs. KVM	101
8.11 Throughput MPI-IO result, Bare-metal vs. Docker vs. KVM	102
8.12 Throughput POSIX Buffered IO result, Bare-metal vs. Docker vs. KVM.....	103
8.13 Throughput POSIX Direct IO result, Bare-metal vs. Docker vs. KVM	104
8.14 Random IOPS Percentage of Slowdown and Speedup Based on Bare-Metal: Docker vs KVM	109
8.15 Throughput Percentage of Slowdown and Speedup Based on Bare-Metal: Docker vs KVM.....	111

LIST OF FIGURES

2.1 IaaS layer of cloud [45].....	8
2.2 PaaS layer of cloud [45].....	9
2.3 SaaS layer of cloud [45].....	10
2.4 NIST Cloud Service Models [37]	12
3.1 Separating operating system from underlying hardware [3].....	15
3.2 Hypervisor type 2 [46]	18
3.3 Hypervisor type 1 [46]	18
3.4 Privilege rings for the x86 available in protected mode [43].....	19
3.5 Docker Execution Drivers [44]	21
3.6 A sample CoreOS host [21]	26
3.7 Communication between containers with etcd [24].....	26
5.1 HPL chart on BareMetal	41
5.2 StarDGEMM and StarFFT chart on BareMetal.....	42
5.3 StarRandomAccess_LCG chart on BareMetal.....	43
5.4 StarSTREAM_Triad and AvgPingPongBandwidth chart on BareMetal.....	45
5.5 AvgPingPongLatency chart on BareMetal.....	46
5.6 IOPS MPI-IO chart on BareMetal.....	49
5.7 IOPS POSIX Buffered IO chart on BareMetal	50
5.8 IOPS POSIX Direct IO chart on BareMetal	51
5.9 Throughput MPI-IO chart on BareMetal	53
5.10 Throughput POSIX Buffered IO chart on BareMetal	54
5.11 Throughput POSIX Direct IO chart on BareMetal	55
6.1 HPL chart on Docker	58

6.2 StarDGEMM and StarFFT chart on Docker	59
6.3 StarRandomAccess_LCG chart on Docker.....	60
6.4 StarSTREAM_Triad and AvgPingPongBandwidth chart on Docker	62
6.5 AvgPingPongLatency chart on Docker.....	63
6.6 IOPS MPI-IO chart on Docker.....	65
6.7 IOPS POSIX Buffered IO chart on Docker	66
6.8 IOPS POSIX Direct IO chart on Docker.....	67
6.9 Throughput MPI-IO chart on Docker	69
6.10 Throughput POSIX Buffered IO chart on Docker	70
6.11 Throughput POSIX Direct IO chart on Docker	71
7.1 HPL chart on KVM.....	74
7.2 StarDGEMM and StarFFT chart on KVM	75
7.3 StarRandomAccess_LCG chart on KVM	76
7.4 StarSTREAM_Triad and AvgPingPongBandwidth chart on KVM	78
7.5 AvgPingPongLatency chart on KVM	79
7.6 IOPS MPI-IO chart on KVM.....	81
7.7 IOPS POSIX Buffered IO chart on KVM.....	82
7.8 IOPS POSIX Direct IO chart on KVM.....	83
7.9 Throughput MPI-IO chart on KVM.....	85
7.10 Throughput POSIX Buffered IO chart on KVM.....	86
7.11 Throughput POSIX Direct IO chart on KVM.....	87
8.1 NIST Cloud Service Models, CTBS Activity Focus [37].....	89
8.2 HPL chart, BareMetal vs. Docker vs. KVM	91
8.3 StarDGEMM and StarFFT chart, BareMetal vs. Docker vs. KVM.....	93

8.4 StarRandomAccess_LCG chart, BareMetal vs. Docker vs. KVM	94
8.5 StarSTREAM_Triad and AvgPingPongBandwidth chart, BareMetal vs. Docker vs. KVM.....	96
8.6 AvgPingPongLatency chart, BareMetal vs. Docker vs. KVM	97
8.7 IOPS MPI-IO chart, BareMetal vs. Docker vs. KVM	99
8.8 IOPS POSIX Buffered IO chart, BareMetal vs. Docker vs. KVM.....	100
8.9 IOPS POSIX Direct IO chart, BareMetal vs. Docker vs. KVM	101
8.10 Throughput MPI-IO chart, BareMetal vs. Docker vs. KVM	102
8.11 Throughput POSIX Buffered IO chart, BareMetal vs. Docker vs. KVM.....	103
8.12 Throughput POSIX Direct IO chart, BareMetal vs. Docker vs. KVM.....	104
8.13 HPCC Performance Comparison, BareMetal vs. Docker vs. KVM	105
8.14 IOR - Random IOPS Performance Comparison, BareMetal vs. Docker vs. KVM	108
8.15 IOR - Throughput Performance Comparison, BareMetal vs. Docker vs. KVM	110

CHAPTER I

INTRODUCTION

Cloud computing is a solution based on consumption of computing resources as services. Cloud computing involves deploying remote servers and software networks that allow on-demand online access to computing services or resources. At the foundation of cloud computing is the broader concept of shared services [1]. Cloud computing uses virtualization to encapsulate and distribute instances of services and resources. Although cloud and virtualization are two close concepts, they are not the same. While these technologies might share a common bond of maximizing computing resources, there are differences between them. Virtualization can reduce complexity for end users while allowing resources to be utilized more effectively. Cloud computing takes the use of those resources to the next level by delivering on-demand access to those components as a service, further reducing complexity, cost, and burden [2].

The principal virtualization methods can be categorized as full virtualization, virtual machines, para-virtualization, and OS level virtualization [3]. Datacenters use these virtualization methods to achieve technical and business goals such as increasing uptime, using more efficient of hardware, saving energy and cost, portability, and improving security. On the other hand, using these virtualization techniques increases complexity and carries the potential disadvantages of losing performance. As a result, there are extensive efforts to improve the performance of these virtualization methods.

The physical host or the actual machine is usually called bare-metal. In this research, by bare-metal, we mean the real host on which an operating system (OS) and other software layers are installed. Virtual machines (VMs) imitate certain factual or conceptual hardware. A virtual machine is a software implementation or an emulation of a machine that executes programs in the same manner as a physical machine [4]. This approach can be implemented using a type 1 hypervisor, which runs directly on the hardware, or a type 2 hypervisor, which runs on another operating system. The details of these implementation are described in sections 3.1.1 and 3.1.2.

Operating system level virtualization allows resources of a system to be partitioned via kernel's support for multiple isolated user space instances, which are usually called containers. A feature in the kernel of Linux known as control groups provides the ability to organize processes into virtualized containerized environments called Linux Containers (LXC). Based on this feature, open-source projects such as Docker [5] and Rocket [6] have emerged. These projects add to the basic functionality of LXC by adding Application Programmer Interface (API) controls and features to manage containers in groups. Containers are sometimes described as extremely lightweight virtual machines. More accurately, however, in the reality, they are just a way to group processes and manage them together in isolated environments that share underlying features of the OS, such as the kernel, but are otherwise separated from other containers. This approach allows the containers to share the host machine's resources, without the overhead of a hypervisor. In addition to this, projects like CoreOS [7] have emerged to provide operating system environments that are streamlined for this purpose and that include tools to manage and to orchestrate these containers. As an example, CoreOS is a distribution of Linux that provides a minimal operating system and includes "etcd" as a System Discovery method, and in which everything runs in a container. As a result of this minimal operating system environment, we have more hardware resources, including memory, to run applications inside of containers.

In this research, we study and evaluate the performance of these environments to have a clear point of view regarding the comparison between the characteristics of each environment with one another. In detail, using a variety of benchmark tools, we investigate, measure, and analyze the performance of a system in bare-metal versus Linux containers versus virtual machines. These results, analyses, and findings can assist cloud computing researchers and developers in designing and developing more effective cloud computing solutions, systems, algorithms, and applications.

1.1 Contributions

Our work makes the following contributions:

- We provide an up-to-date comparison of native, container, and virtual machine environments using 2015-era hardware and software across a cross-section of interesting benchmarks and workloads that are relevant to the cloud.
- We provide analysis and understanding on Cloud Tester Benchmark Suite (CTBS).
- We evaluate the performance of the research hardware which had been provided to Cloud and Autonomic Computing Center (CAC) at Texas Tech University.
- We identify the primary performance impact of current virtualization options for cloud, HPC, and server workloads.
- We show that containers are viable even at the scale of an entire server with minimal performance impact.

1.2 Organization

This thesis is divided into nine chapters. Following this introductory chapter is Chapter II, which presents background information on cloud computing. Section 2.1 describes the definition of cloud computing, section 2.2 describes the five essential characteristics for the cloud model, section 2.3 explains about the three layers of cloud, section 2.4 describes different types of cloud, and section 2.5 describes the difference between virtualization and cloud. After this, Chapter III discusses the current virtualization techniques. This chapter is divided into section 3.1, which discusses virtual machines, hypervisor type 1, hypervisor type 2, and an argument about KVM; and section 3.2 wherein we discussed Linux containers, LXC, Docker, and some related projects based on Linux containers technology.

Chapter IV contains seven short sections. These begin with section 4.1, which covers cloud benchmarking related to this thesis. Sections 4.2 to 4.5 are brief

explanation of the benchmarks suites which is used in this thesis. Section 4.7 is about the CTBS which is the main benchmark suite in this thesis.

Following this benchmarking chapter is Chapter V, which presents the result of CTBS on target systems. Section 5.1 describes the configuration of the tested system, section 5.2 describes the result of seven benchmarks of HPCC, and section 5.3 shows the result of IOR benchmark. After this, Chapter VI similarly discusses the results of the same benchmarks in Linux containers (Docker). Then, Chapter VII shows the results of the CTBS on virtual machines (KVM).

Chapter VII compares the results of the three previous chapters. Chapter IX is the conclusions we have drawn from our research in addition to several ideas for related future work. Following these concluding sections are several appendices. We shows the exact HPCC output of CTBS benchmark in Appendix A. Similarly, Appendix B shows the exact IOR output of CTBS in this thesis. Appendix C contains the environment information of the tested systems including the hardware information and software configuration.

CHAPTER II

CLOUD COMPUTING

2.1 Definition

Based on the definition of NIST (National Institute of Standards and Technology, “cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” [8].

In very simple word, by cloud computing, it is tried to separate hardware layer from operating system layer from applications layer. This way of architecture has a lot of advantages. To understand the benefits of such an architecture, let's say an example. As an illustration, we can think about a new business company which needs vary regular services. Probably, the company needs an email service, a web service, a file service, and etc. As a result, it would need racks and servers and some equipment to maintain them. Then, it would need an IT department to take care of the hardware. Moreover, the company would have to spend time to conhardware. If the company want to extend their business, they will have to buy extra equipment end upgrade their systems. If some of these equipment failed, the business of the company would be interrupted. In addition to these problems, the company would have to worry about security, backup, maintaining cost of equipment, and similar related problems. Instead of all of these, the company could use cloud computing to satisfy its needs. It means that the company can rent servers, storages, and network to use them depends of the company's needs. The services can be upgraded during time. Moreover, the services are on-demand which means that the company can setup all of them in very short period of time and focus on its business. In addition to all of them, the company would pay what they uses.

In general, cloud computing and services based on cloud have lots of benefits including but not limited on-demand access, pay-as-you-use, lower maintaining cost,

secure storage and management, location-independent, scalability and sustainability, agile deployment, virtualized and dynamic, utility-based, and time-sharing models.

2.2 Essential Characteristics

Based on the definition of NIST, the cloud model is composed of five essential characteristics [8].

1. **On-demand self-service:** A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
2. **Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
3. **Resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.
4. **Rapid elasticity:** Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.
5. **Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capabilities at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

2.3 Service Models

Based on the definition of NIST, the cloud model is composed of three service models. These services models are usually called cloud layers as well.

2.3.1 Infrastructure as a Service (IaaS)

Based on the definition of NIST, the capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls) [8].

In fact, Infrastructure as a Service is a provision model in which an organization outsources the equipment used to support operations, including storage, hardware, servers, and networking components. The service provider owns the equipment and is responsible for housing, running and maintaining them. The client typically pays on a per-use basis.

Basically, cloud-service providers of IaaS offer computers physical, virtual machines and other resources. IaaS clouds often offer additional resources such as a virtual-machine disk image library, raw block storage, and file or object storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles. IaaS cloud providers supply these resources on-demand from their large pools installed in data centers. To deploy applications, cloud users install operating system images and their application software on the cloud infrastructure. In this model, the cloud user patches and maintains the operating systems and the application software.

Cloud providers typically bill IaaS services on a utility computing basis which means that cost reflects the amount of resources allocated and consumed [1].

Figure 2.1 IaaS layer of cloud [45]

2.3.2 Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment [8].

In fact, Platform as a Service is a way to rent hardware, operating systems, storage and network capacity over the Internet. The service delivery model allows the customer to rent virtualized servers and associated services for running existing applications or developing and testing new ones.

In this layer, cloud providers deliver a computing platform, including operating system, programming language execution environment, database, and web server. As a result, developers can focus on developing applications and run their software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. Microsoft Azure and Google App Engine are the well-known example of it. One of the important point in this layer is the underlying

computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually [1].

As it was mentioned, this layer is usually for programmers who are developing a new web base application. In PaaS layer, developers get a base level of platform that they can dump their code into and the code will run. For example, a developer does not have to worry about installing PHP, PHP.ini file, Perl, Database, and etc. He/she would have one folder and the only thing to be done is uploading and running the new programs.

Figure 2.2 PaaS layer of cloud [45]

2.3.3 Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings [8].

In fact, Software as a Service is a software distribution model in which a vendor or a service provider host applications and make them available to customers over a network, typically the Internet. In this layer, it is tried to outsource software. It means that applications are running in other place and use the resource of other systems. The

only thing which a user needs is an access over a network to the software. Users use a web browser or a thin client program to work with the software. Users can use their mobile device, tablet, computers or any other systems to use the software as a service. In this architecture, the software is running other place and uses other place's resources.

In the SaaS model, users are provided access to application software. Cloud providers manage the infrastructure and platforms that run those applications. Providers install and operate application software in the cloud to let users access the software from cloud clients. As a result, Cloud users do not manage the cloud infrastructure and platform where the application runs. This way of architecture eliminates the unnecessary need to install and run the application on the cloud user's own computers. The end result is something which simplifies maintenance and support. SaaS is also referred as "on-demand software" [1].

Figure 2.3 SaaS layer of cloud [45]

2.4 Deployment Models

Based on the definition of NIST, the cloud model is composed of four deployment models [8].

1. **Private cloud:** The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

2. **Community cloud:** The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.
3. **Public cloud:** The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
4. **Hybrid cloud:** The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

Figure 2.4 NIST Cloud Service Models [37]

2.5 Virtualization vs. Cloud Computing

One common misconception is the idea that virtualization and cloud computing are the same thing. Cloud and virtualization are two close concepts which should not be considered the same. It is important to understand the difference between these two. While these technologies might share a common bond of maximizing computing resources, there is a difference between them and it's important to understand what that difference is.

Basically, virtualization is the process of simulating “virtual” versions of infrastructure resources, such as computing environments, operating systems, storage devices or network components, as opposed to creating actual or physical versions of those same resources. As an illustration, there is a physical server which is usually called the “host” that controls all of its physical resources of the system such as operating system, memory, storage, etc. Those resources can be allocated to virtual machines that run in isolated environment provided by that host. In general, virtualization enables multiple instances of infrastructure resources to run on the same hardware, with access to those resources being controlled by a hypervisor. The details of the hypervisors are described in the next chapter. On the other hand, cloud computing is the delivery of shared computing resources, software or data as an on-demand service via the Internet, as opposed to virtualization, which is part of a physical infrastructure. For users of either technology, it may appear that the two are one in the same, as the applications or data they access are pulled from a virtual machine somewhere unconnected to a physical host, which is where some of the confusion happens [2].

Virtualization is considered as one of the component of cloud computing. In fact, cloud computing is built on top of a virtualized infrastructure, consisting of compute, storage and network components. In conclusion, virtualization can reduce complexity and add some great features for fail recovery while allowing resources to be utilized more effectively. But, cloud computing takes the use of those resources to the next level

by delivering access to those components on-demand as a service, further reducing complexity, cost and burden.

CHAPTER III

VIRTUALIZATION

Virtualization is the creation of a virtual (rather than actual) version of something, such as an operating system, a server, a storage device or network resources. In this research, we focus on operating system virtualization. Operating system virtualization is the use of software to allow a piece of hardware to run multiple operating system images at the same time. The technology got its start on mainframes decades ago, allowing administrators to avoid wasting expensive processing power. In a simple word, by virtualization it is tired to separate hardware from operating systems.

Figure 3.1 Separating operating system from underlying hardware [3]

This way of architecture has lots of advantages including, but not limited to the following [9]:

- **Increase uptime:** There are some advanced capabilities which aren't found in physical servers such as live migration, storage migration, fault tolerance, high availability, and distributed resource scheduling. These technologies give system administrators the ability to quickly recover from unplanned outages, the ability to quickly and easily move a virtual machine from one server to another one.
- **Use more efficient of hardware:** By virtualization, it is possible to use some fascinating features like Over Allocating to have more efficient performance of hardware. It means a system can be assigned more work to a resource than the resource is able to perform. As an example, there is a physical server with total

amount of memory of 8 GB. A system administrator may assign 2GB memory to VM1, 4GB memory to VM2, and 4GB memory to VM3. Although the total amount of the memory of the system is 8GB, the total allocated memory to virtual machines is 10GB which is more than the system's resource. However, since the system administrator knows that none of the virtual machines work at their pick workload concurrently, the system can be allocated a total memory more than its physical resource.

- **Save energy and cost:** Migrating physical servers over to virtual machines and consolidating them onto far fewer physical servers means lowering monthly power and cooling and maintaining costs in the data center
- **More security:** By virtualization, it is possible to have separate instance for each service to have more security (e.g. Web Service, Email Service, File Service)
- **Faster server provisioning:** As a data center administrator, imagine being able to provide your business units with near instant-on capacity when a request comes down the chain. Server virtualization enables elastic capacity to provide system provisioning and deployment at a moment's notice
- **Extends the life of older applications:** Think that there is old legacy applications still running in a company's environment. These applications probably fit into of these categories: whether it doesn't run on a modern OS, or it may not run on newer hardware. By virtualizing and encapsulating the application and its environment, it is possible to extend life of the applications, maintain uptime, and finally get rid of the old machines hidden in the corner of the data center
- **Help to move things to the cloud:** By virtualizing servers and abstracting away the underlying hardware, companies prepare themselves for a move into cloud

3.1 Virtual Machines

Virtual machines are one of the techniques which datacenters use to take the advantages of virtualization. To understand the concept, it is necessary to know the concept of Hypervisor. Basically, hypervisor or virtual machine monitor (VMM) is a program that allows systems to run multiple OSs on a piece of hardware. In fact, by hypervisor, it is tried to separate the OSs layer from the underlying hardware layer.

A hypervisor runs on a computer. On top of this hypervisor multiple virtual machines is defined as a host machine. Usually, each of these virtual machine is called a guest machine. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. As the result, multiple instances of a variety of operating systems may share the virtualized hardware resources.

Generally, there are two types of hypervisor, hypervisor type 1 (bare-metal hypervisor) and hypervisor type 2 (Hosted Hypervisor) [10].

3.1.1 Hypervisor Type 1 (Bare-metal Hypervisor)

The hypervisor is directly installed on the hardware. So, they are sometimes called bare metal hypervisors. The hypervisor only prepares a platform for managing the hardware. A guest operating system runs as a process on the host. In addition to it, there is a management software to manage and to control the hypervisor. Kernel-Based Virtual Machine (KVM) and VMWare ESXi are the two well known examples of this type of hypervisor.

3.1.2 Hypervisor Type 2 (Hosted Hypervisor)

The hypervisor is installed and run on an operating system (e.g. Windows, Linux). The hypervisor reserves all of its required resource. As the result, there is no Over Allocation concept and other great features of hypervisor type 2. The examples of this type of hypervisor are Parallel Desktop, Oracle VirtualBox, VMware, and etc.

Figure 3.3 Hypervisor type 1 [46]

Figure 3.2 Hypervisor type 2 [46]

By comparing Figure 3.2 and Figure 3.3, it is easy to understand the difference between hypervisor type 1 and type 2. However, the distinction between the two types of hypervisor is not necessarily clear. As an example, Linux's Kernel-based Virtual Machine (KVM) effectively converts the host operating system to a type 1 hypervisor. On the other hand, since Linux distributions are considered as general-purpose operating systems, with other applications competing for VM resources, KVM can also be categorized as type 2 hypervisors.

3.1.3 Kernel-Based Virtual Machine (KVM): Type 1 vs. Type 2

There are some misconceptions about whether KVM is a hypervisor type 1 or hypervisor type 2. The distinction between whether KVM is a Type 1 vs. Type 2 server virtualization hypervisors is an age-old debate, since it doesn't fit neatly into either category.

KVM is the virtualization layer in the Linux kernel. Like all server virtualization implementations, it includes an element called a virtual machine monitor (VMM) that provides isolation between workloads and translation between physical hardware resources and the virtual hardware presented to applications for consumption [11].

The distinction between type 1 and type 2 hypervisors hinges on the number of times that translation occurs between the virtual machine monitor and the guest operating system. With type 1, or bare-metal hypervisors, only one translation occurs.

On the other hand, type 2 hypervisors require a two-layered process that traverses the host OS as well as the VMM.

There are confusion and debate over KVM's classification because it is a part of the Linux OS. Usually, hypervisors type 1 is run and managed at a lower level (ring 0), Figure 3.4. Even for new virtual machine creation, and guests do not share memory blocks, CPU instructions or any of the underlying. But, Linux operating system like KVM are not the same. This means KVM suffers performance, latency, security, scalability, isolation, and other issues that do not affect a true bare-metal hypervisor. However, KVM runs in kernel mode on bare metal and uses a hardware virtualizer. Moreover, KVM guests spend almost all their time running in direct execution mode, which is another criterion for a type 1 hypervisor. As a result, it seems that KVM fits more the definition of a type 1 hypervisor [11].

Figure 3.4 Privilege rings for the x86 available in protected mode [43]

3.2 Linux Containers

Linux Container (LXC) is an operating system-level virtualization method. It provides operating system-level virtualization through a virtual environment that has its own process and network space, instead of creating a full virtual machine. LXC relies on cgroups functionality that was released in Linux kernel version 2.6.24. It also relies on other kinds of namespace-isolation functionality, which were developed and integrated into the mainline Linux kernel [12]. To contain process, LXC uses some of

the kernel features. The following is some of those common features that is provided by the kernel [13]:

- **Kernel Namespaces:** ipc, uts, mount, pid, network and user
- **AppArmor and SELinux profiles:** AppArmor allows the system administrator to associate with each program a security profile that restricts the capabilities of that program
- **Secure Computing Mode (Secomp):** It is a simple sandboxing mechanism for the Linux kernel
- **Chroots (using pivot_root):** A chroot on UNIX operating systems is an operation that changes the apparent root directory for the current running process and its children.
- **Kernel Capabilities**
- **Control Groups (cgroups)**

In summary, the goal of LXC is to create an environment as close as possible as a standard Linux installation but without the need for a separate kernel.

As it is mentioned, cgroups is one of the key feature of the kernel that LXC use to contain processes. Linux kernel uses cgroups for resource isolation (CPU, memory, block I/O, network, etc.). Cgroups provides namespace isolation to completely isolate applications' view of the operating environment, including process trees, network, user ids and mounted file systems. Working on the project of cgroups was started by engineers at Google in 2006 under the name "process containers" [14]. In late 2007, the project was renamed to "Control Groups" and finally merged into kernel version 2.6.24 [15]. LXC combines cgroups and namespace to provide an isolated environment for applications. In general, cgroups feature of the kernel provides:

- **Resource Limitation:** It means that a group of processes can be set not to exceed a set memory limit [16] [17]

- **Prioritization:** Some groups may get a larger share of CPU [18] or disk I/O throughput [19]
- **Accounting:** To measure how much resources certain systems use for e.g. billing purposes [20].
- **Control:** Freezing groups or Checkpointing and restarting [20].

3.2.1 Docker

As it was mentioned before, LXC is a feature of kernel that has been in the kernel for a while. But, it is only recently being used in innovative ways. Docker is an open-source project that uses LXC to isolate processes. In fact, it is a user-friendly interface to Linux containers that makes creating and managing Linux containers really easy. We can think of a Docker container as extremely lightweight virtual machines. However, in reality, they are just a process. They allow code to run in isolation from other containers; but safely share the machine's resources, all without the overhead of a hypervisor.

Docker containers can boot extremely fast, in scale of milliseconds, which gives us unprecedented flexibility in managing load across a cluster. Perhaps, the most powerful feature of containers is the ability to run any Linux userland that is compatible with the latest kernel. This means that on a container in a host, it is possible to run

Figure 3.5 Docker Execution Drivers [44]

applications with very specific requirements beside each other. For example, we can run containers that contain an Ubuntu userland on the same machine as containers that contain a Fedora or CentOS userland [21].

Docker uses cgroups feature of the Linux kernel for process isolation and a network namespace for isolated networking. In general, Docker is built on top of the following (Figure 3.5):

- **libvirt:** It is an open source API, daemon and management tool for managing platform virtualization
- **LXC**
- **systemd-nspawn:** systemd is a system management daemon designed for Linux to orchestrate layer of hypervisors

3.2.2 Docker vs. LXC

It is important to understand the difference between LXC and Docker. Basically, LXC refers to capabilities of the Linux kernel (specifically namespaces and control groups) which allow sandboxing processes from one another. But, on top of this low-level foundation of kernel features, Docker offers a high-level tool. In fact, Docker is not a replacement for LXC. We may categorize the difference as the following [22]:

- **Portable Deployment across Machines:** Docker defines a format for bundling an application and all its dependencies into a single object which can be transferred to any Docker-enabled machine, and executed there with the guarantee that the execution environment will be the same. LXC, alone is not enough for portable deployment. If you sent me a copy of your application installed in a custom LXC configuration, it would almost certainly not run on my machine the way it does on yours, because it is tied to your machine's specific configuration.

- **Application-Centric:** Docker is optimized for the deployment of applications, as opposed to machines. This is reflected in its API, user interface. By contrast, the LXC helper scripts focus on containers as lightweight machines
- **Automatic Build:** Docker includes a tool for developers to automatically assemble a container from their source code, with full control over application dependencies, build tools, packaging etc. They are free to use make, maven, chef, puppet, salt, debian packages, rpms, source tarballs, or any combination of the above, regardless of the configuration of the machines.
- **Versioning:** Docker includes git-like capabilities for tracking successive versions of a container, inspecting the diff between versions, committing new versions, etc. Docker also implements incremental uploads and downloads, similar to "git pull", so new versions of a container can be transferred by only sending diffs.
- **Component re-Use:** Any container can be used as a "base image" to create more specialized components. This can be done manually or as part of an automated build. For example you can prepare the ideal python environment, and use it as a base for 10 different applications.
- **Sharing:** Docker has access to a public registry (<http://index.docker.io>)
- **Tool Ecosystem:** Docker defines an API for automating and customizing the creation and deployment of containers. There are a huge number of tools integrating with docker to extend its capabilities. For example, Salt, Mesos, OpenStack nova.

In summary, LXC refers to capabilities of the Linux kernel which allow sandboxing processes from one another. While, Docker is a high-level tool on top of the low-level foundation of kernel features.

3.2.3 Related Projects Based on Linux Containers

As it was mentioned, LXC has been a feature in the kernel of Linux that provides a virtualized containerized environment for a process to run. Based on this feature, there are some open-source projects that Docker is the center of them. In addition to Docker which provides a user-friendly interface to Linux containers, there are some other open-source projects based on LXC, including, but not limited to, Kubernetes and CentOS.

To have a better understanding of these projects, it is necessary to know System Discovery.

3.2.3.1 System Discovery

System Discovery is a concept in modern service-oriented architectures. Basically, service discovery protocols are network protocols which allow automatic detection of devices and services offered by these devices on a computer network. In datacenters and cluster environment, we refer system discovery to services and processes. So, service discovery tools manage how processes and services in a cluster can find and talk to one another. At its core, service discovery is about knowing when any process in the cluster is listening on a TCP or UDP port, and being able to look up and connect to that port by name. Service discovery is a general idea. But, in this research we use it for Linux containers and specifically for Docker. The ultimate goal is to get Docker containers to easily communicate across hosts.

3.2.3.2 etcd

“etcd” is one of the well-known implementation of service discovery. It is an open-source distributed key-value store that serves as the backbone of distributed systems by providing a hub for cluster coordination and state management. Basically, etcd is written in Go and uses Raft consensus algorithm to take care of storing and replicating data across the entire cluster. Raft is a protocol for multiple nodes to maintain identical logs of state changing commands, and any node in a raft will coordinate with the others to agree on the last state. As a result of this algorithm, etcd can recover from hardware failure and network partitions.

Clusters are usually built from a large collection of machines with the ability to run any workload at any given time. In order to have high levels of efficiency, we need to distribute workloads appropriately across all machines in the cluster. So, clusters need a way of coordinating with each other. Moreover, clusters often suffer from network partitions and nasty race conditions that need to be resolved somewhere. All the above cases are where etcd comes to play [23].

3.2.3.3 CoreOS

As Docker and etcd were discussed, there is another open-source project which is built on top of them. CoreOS is an open-source project which provides an ecosystem of tools to manage and to orchestrate these containers.

In fact, CoreOS is a distribution of Linux. It is a minimal operating system consists of etcd and Docker. So, everything, which is running in CoreOS, is running in Docker containers. Figure 3.6 can abstract CoreOS's structure. CoreOS uses etcd as Service Discover to let containers communicate with one another. It means that each container has an IP address to talk with another one. These containers can read, write and listen to etcd. As a result, with these three actions we can construct extremely sophisticated orchestration to happen whenever etcd values change [24]. Figure 3.7 can represent the communication between containers in CoreOS system base in a cluster.

Since CoreOS is a minimal operating system that only provides the basic functionality for running Docker containers and communicating between them, it uses less hardware resources. As a result, there are more hardware resources for containers to run applications. Based on the official claim of the CoreOS project, it uses 40% less RAM than any other average Linux distribution. It uses 114 MiB RAM in total that causes applications have more memory to run faster [21].

CoreOS has some other advantages namely, an active/passive dual-partition scheme to update the OS as a single unit instead of package by package. This makes each update quick, reliable and able to be easily rolled back.

Figure 3.7 Communication between containers with etcd [24]

Figure 3.6 A sample CoreOS host [21]

3.2.3.4 Kubernetes

Kubernetes is the open source container cluster manager from Google. Kubernetes leverages "etcd" to take care of storing and replicating data across the entire cluster in the way that can recover from failures [23]. It is a strong and open container management framework for orchestrating groups of Docker containers (Pods).

CHAPTER IV

BENCHMARKING

Benchmark definitions often refer to the concept of a System Under Test (SUT). The SUT is a collection of components necessary to run a benchmark scenario [25].

In fact, benchmarking is an activity that tells us our position or status by comparing ourselves to others. The reason for such a comparison is:

- To understand where we are today
- What might be some of the areas that we need to improve
- Are there others out who have similar problems
- Have they solved them already

It is important to understand that only things that are comparable can be compared. When these comparison are done correctly, it is called Benchmarking. To benchmark, it is necessary to define what we want to benchmark. It is important to define appropriate comparators which is called Key Performance Indicator (KPI), i.e. MTBF (Mean Time Between Failure) or MTTR (Mean Time to Repair). Moreover, it is important to ensure that our comparison are comparable. This point implies that for example we cannot compare the result of a benchmark suite on a system with the result of another benchmark suit on the same system.

4.1 Cloud Benchmarking

As it was explained in the previous section, benchmark definitions often refer to the concept of a System Under Test (SUT). The SUT is a collection of components necessary to run the benchmark scenario [25]. Under benchmarking in the cloud, we understand the process of benchmarking services provided by the cloud. So, a cloud benchmark is a benchmark in which the SUT contains a cloud service as component of interest.

Table 4.1 Technical metrics or Key Performance Indicator (KPI) [26]

Metrics	Measures
Reliability	Mean Time Between Failures (MTBF)
Recoverability	Time to recover (sec. / min. / hrs.) VM fail-over time VM migration time
Availability	% uptime
Rapid elastically (Scalability)	Resource provisioning time: Servers, storage, network connection
Security and Information Assurance	# of known security vulnerabilities # of security or risk incidents for each platform or service
Improved accessibility of datasets, services, and resources by external organizations.	# of internal and external data sets accessible # of networks connected # of organizations using them
Analyst time efficiency	# of portals available # of SaaS widgets available # of end-user services, e.g. workflow engines, Map-Reduce, etc.
User satisfaction	Quality of Service (QoS) Perceived throughput Perceived response latency Mean time to respond to data center incident Mean time to repair infrastructure problems # of deficiency report submitted
Workflow Throughput	Product Generation Time Number of Products Produces VM startup and termination time Bandwidth (GB/s) and Throughput Rat (GFLOPs/s)

Table 4.2 Economic metrics or Key Performance Indicator (KPI) [26]

Metrics	Measures
Power	Watt \$/week, month, year
Space	Square feet # of racks \$ of operation
Cooling costs	Watts, therms, BTUs \$/week, month, year
Software licensing costs	\$/seat/week, month, year
Number of data centers required to support the enterprise	# of centers \$ of operation
Excess capacity necessary to accommodate processing surge demands	# of servers MB of storage Network bandwidth \$ of operation
Development time and cost through rapid provisioning of IT infrastructure	Seconds, minutes vs. month, years \$ savings through reduced time
Development time and cost through on-demand deployment and reuse of well-known established software tools	Seconds, minutes vs. months, years \$ saving through reduce time
Personnel costs	\$
Recapitalization cost and schedule	Time and \$

Cloud benchmarking can be done in each layers of cloud (IaaS, PaaS, and SaaS). In this research, we have focused on benchmarking in IaaS layer of cloud. So, based on the characteristic of the Infrastructure as a Service layer, the appropriate Key Performance Indicator (KPI) are defined. Table 4.1 and Table 4.2 show some of these metrics.

As it was mentioned, in this research we have focused on the IaaS layer of the cloud. Since, the focus is on IaaS layer, the interest is more low level performance testing. There are some well-known benchmarking suite which is useful for benchmarking in this layer. These benchmarks includes, but not limited to:

- **IOR (Interleave Or Random):** It is used for benchmarking parallel file systems. The benchmarks is discussed in the IOR section in more details [27].
- **Bonnie++:** It is a benchmark which test the performance of hard drive and file system. This benchmark try to simulate a program which is going to have access to a database. It means that Bonnie++ tests creation, reading, and deleting of small files which can simulate the usage of programs such as “Maildir” format email [28].
- **IOZone:** It is a file system benchmark tool. The benchmark tests file I/O performance for operation such as: Read, Write, Re-Read, Re-Write, Read Backwards, Read Strided, FRead, FWrite, Random Read, PRead, MMap, AIO Read, and AIO Write [29].
- **SHOC (Scalable Heterogeneous Computing):** It is benchmark suite to test the performance and stability of heterogeneous system. The benchmarks is discussed in the SHOC section in more details [30].
- **VMmark:** It is a free tool that can be used to measure the performance and scalability of applications running in virtual machines. The suite runs on several virtual machines simultaneously to mimic typical software applications such as email servers, database servers, and web servers. Then, it collects performance

statistics that are relevant to each type of application, such as commits per second for database servers, or page accesses per second for web servers [31].

- **HPCC (High Performance Computing Challenge):** It is a benchmarking suite which assesses compute, memory, and network capabilities of systems. The benchmarks is discussed in the HPCC section in more details [32].
- **LINPACK:** It is benchmark to measure floating point computing power of systems. The benchmarks is discussed in the LINPACK section in more details [33].
- **HPL (High Performance LINPACK):** It is a portable implementation of LINPACK. The benchmarks is discussed in the HPL section in more details [34].
- **CTBS (Cloud Tester Benchmark Suite):** It is benchmark suite which is provided by the Aerospace Corporation as the technical partnership of Cloud and Autonomic Computing Center (CAC) at Texas Tech University. The benchmark suite is discussed in the CTBS chapter in more details.

4.2 IOR (Interleave Or Random)

The IOR (Interleaved Or Random) is a benchmark program developed at LLNL (Lawrence Livermore National Laboratory). The IOR software is used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces. In fact, it is usually used for performance testing of parallel file systems for high performance clusters. Basically, IOR test aggregates I/O rates via several typical middleware libraries including MPI collective I/O calls and HDF5 library calls, in addition to POSIX I/O calls [27].

There are varieties of input arguments which allow variance of the overall I/O size, individual transfer size, file access mode (single shared file, one file per client), and whether the data is sequentially or randomly accessed.

In general, the IOR benchmark program can be used to mimic the I/O patterns of real HPC applications. It is an Open Source software which is developed by C and it can be downloaded and compiled from “Sourceforge” and “GitHub”

4.3 SHOC (Scalable Heterogeneous Computing)

The Scalable Heterogeneous Computing (SHOC) benchmark suite is a collection of benchmark programs which test the performance and stability of systems using computing devices with non-traditional architectures for general purpose computing. Its initial focus is on systems with Graphics Processing Units (GPUs) and multi-core processors based on the OpenCL programming standard which is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs) and other processors. In addition to OpenCL-based benchmark programs, SHOC also includes a Compute Unified Device Architecture (CUDA) version which is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce [35].

The benchmark suite can be used on clusters as well as individual hosts. In general, SHOC benchmark suite is divided into two primary categories [30]:

1. **Stress tests:** it uses computationally demanding kernels to identify OpenCL devices with bad memory, insufficient cooling, or other component problems
2. **Performance tests:** The performance tests are further subdivided according to their complexity and the nature of the device capability. This categorization is similar in spirit to that used in the BLAS API.

As it is mentioned before, SHOC is an Open Source software and it can be downloaded “GitHub”.

4.4 LINPACK (Linear Algebra Package)

Linear Algebra Package is a measure of a system's floating point computing power. It measures how fast a computer solves a dense “n by n” system of linear equations “ $Ax = b$ ”, which is a common task in engineering. It is used to build the Top500 list, ranking the world's most powerful supercomputers. It is a collection of algebra FORTRAN subroutines which runs $\frac{2}{3}n^3 + 2n^2$ floating point operations. There are three versions of LINPACK: LINPACK 100, LINPACK 1000 and HPLinpack. In HPLinpack the size n of the problem can be made as large as it is needed to optimize the performance results of the machine. HPLinpack is suitable for testing parallel computers [33].

4.5 HPL

HPL stands for High Performance LINPACK Benchmark. It is a portable implementation of the High-Performance LINPACK Benchmark for Distributed-Memory computers. It is a benchmarking for double precision for distributed memory. HPL is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers. The HPL package provides a testing and timing program to quantify the accuracy of the obtained solution as well as the time it took to compute it. The HPL software package requires the availability on your system of an implementation of the MPI and BLAS or the VSIPL [34].

4.6 HPCC

HPCC, High Performance Computing Challenge, is a benchmarking suite which consists of 7 tests [36]:

1. **HPL:** The LINPACK benchmark which measures the floating point rate of execution for solving a linear system of equations.
2. **DGEMM:** Measures the floating point rate of execution of double precision real matrix-matrix multiplication.

3. **STREAM:** A simple synthetic benchmark program that measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernel.
4. **PTRANS (Parallel Matrix Transpose):** Exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.
5. **FFT (Fast Fourier Transform):** Measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform.
6. **Communication Bandwidth and Latency:** A set of tests to measure latency and bandwidth of a number of simultaneous communication patterns.
7. **Random Access:** Measures the rate of integer random updates of memory, Giga-updates per second (GUPS).

4.7 Cloud Tester Benchmark Suite (CTBS)

CTBS stands for Cloud Tester Benchmark Suite. It is a benchmark suite with Bottom-to-Top approach. It means that it evaluates the performance of physical hardware then impact of virtualization of compute, network, storage performance and finally the performance of the management software [37].

CTBS is a benchmark suite for standardized computing performance benchmarking of cloud Infrastructure-as-a-Service (IaaS) environments. It is a benchmark suite which is provided by the Aerospace Corporation as the technical partnership of Cloud and Autonomic Computing Center (CAC) at Texas Tech University.

CTBS determines basic computer system (i.e. compute, interconnect, and storage) performance for individual servers deployed within a cloud IaaS system. Part of the suite can also be used to determine native (non-virtualized) server performance as well [38, p. 3]. It is a benchmark suit which consists of some well-known benchmarks including HPCC, IOR, SHOC and LAPACK benchmarks. Compute and interconnect

performance is determined using the HPCC software suite; storage performance is determined with the IOR software suite; and heterogeneous processing performance is determined with the SHOC software suite. In addition to them, there is a VM Acquisition and Release timing tool (VM A&R) which is provided to determine virtual server provisioning performance a cloud controller [38, p. 3]. However, since in this research the target is a local server, we have not used the VM A&R and SHOC parts of the benchmark suite.

The entire CTBS system is self-contained within a Cloud Tester VM Image (CTI) which can be easily deployed to a cloud IaaS system which uses either a KVM or VMware ESX/ESXi hypervisor [38, p. 3]. However, in this research, we have not used the CTIs. Instead, the benchmark suite is compiled from the scratch to have the best performance.

Basically, the benchmark suite has tried to see whether the system is built right or not. The benchmark suite target questions such as [38, pp. 3, 4]:

- **Evaluation:** What are the system's strengths and weaknesses?
- **Acceptance Testing:** Does the system meet performance requirements?
- **Diagnostic Value:** Where does the system need reconfiguration and/or tuning?
- **Guide Future System Design:** What technologies can provide performance improvement to targeted applications of interest?

As it was mentioned in the beginning of the chapter, it is important to define appropriate comparators which is called Key Performance Indicator (KPI). In general, as it is shown in the Table 4.3 by red color, CTBS addresses standard technical computer system metrics such as Workflow Throughput, Reliability, Recoverability, Scalability, and Availability for Cloud Infrastructure-as-a-Service Systems.

Table 4.3 CTBS Technical Metrics [37]

Metrics	Measures
Reliability	Mean Time Between Failures (MTBF)
Recoverability	Time to recover (sec. / min. / hrs.) VM fail-over time VM migration time
Availability	% uptime
Rapid elastically (Scalability)	Resource provisioning time: Servers, storage, network connection
Security and Information Assurance	# of known security vulnerabilities # of security or risk incidents for each platform or service
Improved accessibility of datasets, services, and resources by external organizations.	# of internal and external data sets accessible # of networks connected # of organizations using them
Analyst time efficiency	# of portals available # of SaaS widgets available # of end-user services, e.g. workflow engines, Map-Reduce, etc.
User satisfaction	Quality of Service (QoS) Perceived throughput Perceived response latency Mean time to respond to data center incident Mean time to repair infrastructure problems # of deficiency report submitted
Workflow Throughput	Product Generation Time Number of Products Produces VM startup and termination time Bandwidth (GB/s) and Throughput Rat (GFLOPs/s)

4.7.1 System Requirements

A basic scientific and engineering workstation software stack on the server is required for CTBS operation. In particular perl, python, gcc, gfortran, make and autotools are required in order to successfully compile the CTBS [38, p. 6]. In addition to these software requirement, the system has to meet some hardware requirement to run the benchmarks. HPCC testing with the defined CTBS workloads requires a server with at least 8 CPUs, 16 GB of memory, and 20 GB of storage. IOR testing with the defined CTBS workloads requires a server with 8 CPUs, 16 GB of memory, and 20 GB of storage. SHOC testing with the defined CTBS workload requires a server with 8 CPUs, 16 GB of memory, 20 GB of storage, and 1 GPUs (compute capability 2.x or greater, Tesla C2xxx or M2xxx- Family). Although the research has not done any benchmarking with SHOC, one of the future work is testing the Xeon Phi co-processor by CTBS [38, pp. 8, 9].

CHAPTER V

BENCHMARKING BARE-METAL EXECUTION ENVIRONMENT

In this phase of research, we used one of the nodes of a research cluster which had been provided by one of the CAC's member to do all the performance testing. The node is hosted in High Performance Computing Center in Texas Tech University.

Basically, the node was installed from the scratch with CentOS 6.6. To minimize the variance of the result of the benchmarks, the minimum base distribution of CentOS was used. Then, all the required packages which was necessary to compile CTBS was installed on the node. The list of the required software is mentioned in the section 4.7.1. Since the required installed packaged are not a service to run in background, they do not have any overhead to the system. So, we can consider the system as the pure minimal installation of CentOS 6.6.

5.1 System Configuration

As it was mentioned the system was installed from the scratch by minimal base CentOS 6.6. The following are the general specification of the hardware.

- CPU: 2 * Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz (24 CPUs)
- RAM: 64GB (8 * 8GB) @ 1600MHz
- Storage: 250GB Hard Disk Drive, 8 * 2TB Hard Disk Drive (16 TB)
- Network Adapters: 1GbE BMC, 2 * 1GbE, 2 * 10GbE

However, the complete configuration of the system can be find in Appendix C, section C1.

5.2 HPCC Result

As it was mentioned, HPCC (High Performance Computing Challenge), is a benchmarking suite which consists of 7 tests, HPL which measures the floating point rate of execution for solving a linear system of equations, DGEMM which measures the floating point rate of execution of double precision real matrix-matrix multiplication,

STREAM which is a simple synthetic benchmark program that measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernel, PTRANS (Parallel Matrix Transpose) which exercises the communications where pairs of processors communicate with each other simultaneously, FFT (Fast Fourier Transform) which measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform, “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns, and “Random Access” which measures the rate of integer random updates of memory, Giga-updates per second (GUPS) [36].

As the result, we can show the HPCC Key Performance Metrics (KPMs) as Table 5.1.

Table 5.1 HPCC Key Performance Metrics (KPMs) [38, p. 32]

Metric	Unit	Description
HPL	TFlops	Dense Linear Solve
StarDGEMM	GFlops	Multiple Concurrent Double-Precision Matrix Multiplies
StarFFT	GFlops	Multiple Concurrent Double-Precision FFTs
StarRandomAccess LCG	GUPs	Multiple Concurrent Random Memory Location Accesses
StarSTREAM Triad	GB/s	Multiple Concurrent STREAM Triad Operations
AvgPingPongBandwidth	GB/s	Average “Ping-Pong” Bandwidth between two processes
AvgPingPongLatency	usec	Average “Ping-Pong” Latency between two processes

5.2.1 HPL

As it discussed before, HPL is the High Performance version of LINPACK Benchmark which measures the floating point rate of execution for solving a linear system of equations. The following scaled residual check is computed:

$$\|Ax-b\| / (\text{eps} * (\|x\| * \|A\| + \|b\|) * N)$$

The matrix A is randomly generated for each test. The relative machine precision (eps) is taken to be 1.110223e-16. Computational tests pass if scaled residuals are less than 16.0.

The following is a short explanation of the input/output parameters follows:

- **T/V:** Wall time / encoded variant.
- **N:** The order of the coefficient matrix A.
- **NB:** The partitioning blocking factor.
- **P:** The number of process rows.
- **Q:** The number of process columns.
- **Time:** Time in seconds to solve the linear system.
- **Gflops:** Rate of execution for solving the linear system.

The following parameter values is used:

N: 4096	NBMIN: 4
NB: 128	NDIV: 2
PMAP: Row major process mapping	RFACT: Crout
P: 2	BCAST: 1ringM
Q: 4	DEPTH: 1
PFACT: Right	SWAP: Mix (threshold = 64)

L1: transposed form

EQUIL: yes

U: transposed form

ALIGN: 8 double precision word

The result of the linear system of equations:

Table 5.2 HPL result on Bare-metal

T/V	N	NB	P	Q	Time	Gflop
WR11C2R4	4096	128	2	4	4.42	1.038e+01

$$\|Ax-b\| / (\text{eps} * (\|x\| * \|A\| + \|b\|) * N) = 0.0042988$$

The following bar plots show result of HPL runs by CTBS:

Figure 5.1 HPL chart on BareMetal

5.2.2 DGEMM and FFT

As it is mentioned before, DGEMM measures the floating point rate of execution of double precision real matrix-matrix multiplication. The following is the result of StarDGEMM runs by HPCC part of CTBS:

Table 5.3 DGEMM result on Bare-metal

Scaled residual	Minimum Gflop/s	Average Gflop/s	Maximum Gflop/s
0.0181865	1.488568	1.513195	1.524991

As it is mentioned before, FFT measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform. The following is the result of StarFFT runs by HPCC part of CTBS:

Table 5.4 StarFFT result on Bare-metal

Vector size	Generation time	Computing	Inverse FFT	Minimum Gflop/s	Average Gflop/s	Maximum Gflop/s
524288	0.027	0.030	0.032	1.620491	1.657962	1.681833

The following bar plots show result of StarDGEMM and StarFFT:

Figure 5.2 StarDGEMM and StarFFT chart on BareMetal

5.2.3 Random Access

As it is mentioned before, Measures the rate of integer random updates of memory, Giga-updates per second (GUPS). The following is the result of StarRandomAccess_LCG runs by HPCC part of CTBS:

Table 5.5 StarRandomAccess_LCG result on Bare-metal

Main table size	Number of updates	CPU time used	Real time used	Minimum GUP/s	Average GUP/s	Maximum GUP/s
2097152 Words	8388608	0.122981	0.123470	0.067555	0.068617	0.069307

The following bar plots show result of StarRandomAccess_LCG:

Figure 5.3 StarRandomAccess_LCG chart on BareMetal

5.2.4 STREAM and Communication Bandwidth

As it is mentioned before, STREAM is a simple synthetic benchmark program that measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernel. The following is the result of StarSTREAM runs by HPCC part of CTBS:

The system, which used to test, uses 8 bytes per DOUBLE PRECISION word. The benchmark run each test 10 times, but only the best time for each is used.

Table 5.6 StarSTREAM result on Bare-metal

Function	Rate (GB/s)	Avg time	Min time	Max time
Copy	6.4097	0.0018	0.0017	0.0019
Scale	6.3335	0.0018	0.0018	0.0019
Add	6.8339	0.0025	0.0025	0.0025
Triad	7.2255	0.0024	0.0023	0.0024

As it is mentioned before, HPCC includes “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. The following is the result of PingPongBandwidth runs by HPCC part of CTBS:

Table 5.7 PingPongBandwidth result on Bare-metal

Minimum Ping Pong Bandwidth (GBytes)	Average Ping Pong Bandwidth (GBytes)	Maximum Ping Pong Bandwidth (GBytes)
3.90622	5.25721	6.97018

The following bar plots show result of StarSTREAM_Triad and AvgPingPongBandwidth

Figure 5.4 StarSTREAM_Triad and AvgPingPongBandwidth chart on BareMetal

5.2.5 Communication Latency

As it is mentioned before, HPCC includes “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. The following is the result of PingPongLatency runs by HPCC part of CTBS:

Table 5.8 PingPongLatency result on Bare-metal

Minimum Ping Pong Latency (usec)	Average Ping Pong Latency (usec)	Maximum Ping Pong Latency (usec)
0.357628	0.711649	1.05964

The following bar plots show result of AvgPingPongLatency:

Figure 5.5 AvgPingPongLatency chart on BareMetal

5.3 IOR Result

As it is mentioned before, IOR (Interleaved Or Random) is a benchmark program used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces [27].

As the result, we can show the IOR Key Performance Metrics (KPMs) as Table 5.9.

Table 5.9 IOR Key Performance Metrics For Throughput & IOPS Operations [38, p. 34]

Metric	Unit	Description
Mean Read	MiB/s	Mean Read File System Throughput
Mean Write	MiB/s	Mean Write File System Throughput
Min Read	MiB/s	Minimum Read File System Throughput
Max Read	MiB/s	Maximum Read File System Throughput
Min Write	MiB/s	Minimum Write File System Throughput
Max Write	MiB/s	Maximum Write File System Throughput

CTBS benchmark run gathers the following performance information for the specified file system using three different file I/O APIs: POSIX Buffered-IO, POSIX DirectIO, and MPI-IO. Moreover, for each file I/O API, a set of five 5 throughput (thpt) and random IOPS (iops) operations are performed [38, p. 34].

5.3.1 Random IOPS

5.3.1.1 MPI-IO

As it is mentioned before, there are varieties of input arguments which allow variance of the overall I/O size, individual transfer size, file access mode (single shared file, one file per client), and whether the data is sequentially or randomly accessed. The following is the command passed for the test:

```
ior -b 100M -t 4K -z -a MPIIO -i 5
```

As a result, we have the following:

API: MPIIO v2	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in MPI-IO

Table 5.10 IOPS MPI-IO result on Bare-metal

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1004.03	804.17	963.51
Read	2806.70	2793.54	2800.30

The following bar plots show result of IOR

Figure 5.6 IOPS MPI-IO chart on BareMetal

5.3.1.2 Posix Buffered IO

Basically, this is the same as the previous test. But, here the POSIX Buffered API is used. The following is the command passed for the test.

```
ior -b 100M -t 4K -z -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in POSIX Buffered IO

Table 5.11 IOPS POSIX Buffered IO result on Bare-metal

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1081.40	959.84	1055.16
Read	2999.49	2991.15	2994.07

The following bar plots show result of IOR

Figure 5.7 IOPS POSIX Buffered IO chart on BareMetal

5.3.1.3 Posix Direct IO

Basically, this is the same as the previous test. But, here the POSIX Direct API is used. The following is the command passed for the test.

```
ior -b 100M -t 4K -B -z -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in POSIX Direct IO

Table 5.12 IOPS POSIX Direct IO result on Bare-metal

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	0.56	0.49	0.51
Read	0.95	0.72	0.89

The following bar plots show result of IOR

Figure 5.8 IOPS POSIX Direct IO chart on BareMetal

5.3.2 Throughput

5.3.2.1 MPI-IO

Basically, this is the same as the Random IOPS test. But, here the test is for Throughput. The following is the command passed for the test:

```
ior -b 1G -t 2M -a MPIIO -i 5
```

As a result, we have the following:

API: MPIIO v2	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in MPI-IO

Table 5.13 Throughput MPI-IO result on Bare-metal

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1320.67	1301.00	1316.42
Read	5043.19	5026.46	5038.50

The following bar plots show result of IOR

Figure 5.9 Throughput MPI-IO chart on BareMetal

5.3.2.2 Posix Buffered IO

Basically, this is the same as the previous test. But, here the POSIX Buffered API is used. The following is the command passed for the test.

```
ior -b 1G -t 2M -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in POSIX Buffered IO

Table 5.14 Throughput POSIX Buffered IO result on Bare-metal

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1318.04	1314.92	1316.78
Read	5045.42	5041.46	5042.92

The following bar plots show result of IOR

Figure 5.10 Throughput POSIX Buffered IO chart on BareMetal

5.3.2.3 Posix Direct IO

Basically, this is the same as the previous test. But, here the POSIX Direct API is used. The following is the command passed for the test.

```
ior -b 1G -t 2M -B -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in POSIX Direct IO

Table 5.15 Throughput POSIX Direct IO result on Bare-metal

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	74.57	72.30	73.67
Read	136.75	108.91	118.95

The following bar plots show result of IOR

Figure 5.11 Throughput POSIX Direct IO chart on BareMetal

CHAPTER VI

BENCHMARKING CONTAINERIZED EXECUTION ENVIRONMENT

In this phase of research, we used the same node as the previous phase. So, basically, the node which, was installed from the scratch with minimum base CentOS 6.6, is used in this phase. Then, Docker v1.4.1 is installed on the node. After that, a Docker container is run with no limitation in terms of the resource of the system. As the result, the container had access to the all of the system's resources (CPUs, Memory). Then, the required packages which was necessary to compile CTBS was installed inside of the container. As it is mentioned before, since the required installed packaged are not a service to run in background, they do not have any overhead to the container.

6.1 System Configuration

As it is mentioned, the container did not have any limitation in terms of the access to the system resources. It means that the container can use all the CPUs and Memory of the system.

6.2 HPCC Result

As it was mentioned, HPCC (High Performance Computing Challenge), is a benchmarking suite which consists of 7 tests, HP, DGEMM, STREAM, PTRANS, FFT, "Communication Bandwidth and Latency", and "Random Access" [36].

6.2.1 HPL

As it discussed before, HPL is the High Performance version of LINPACK Benchmark which measures the floating point rate of execution for solving a linear system of equations. The following scaled residual check is computed:

$$\|Ax-b\| / (\text{eps} * (\|x\| * \|A\| + \|b\|) * N)$$

The matrix A is randomly generated for each test. The relative machine precision (eps) is taken to be 1.110223e-16. Computational tests pass if scaled residuals are less than 16.0.

The following is a short explanation of the input/output parameters follows:

- **T/V:** Wall time / encoded variant.
- **N:** The order of the coefficient matrix A.
- **NB:** The partitioning blocking factor.
- **P:** The number of process rows.
- **Q:** The number of process columns.
- **Time:** Time in seconds to solve the linear system.
- **Gflops:** Rate of execution for solving the linear system.

The following parameter values is used:

N: 4096	NBMIN: 4
NB: 128	NDIV: 2
PMAP: Row major process mapping	RFACT: Crout
P: 2	BCAST: 1ringM
Q: 4	DEPTH: 1
PFACT: Right	SWAP: Mix (threshold = 64)

L1: transposed form

EQUIL: yes

U: transposed form

ALIGN: 8 double precision word

The result of the linear system of equations:

Table 6.1 HPL result on Docker

T/V	N	NB	P	Q	Time	Gflop
WR11C2R4	4096	128	2	4	4.41	1.038e+01

$$\|Ax-b\| / (\text{eps} * (\|x\| * \|A\| + \|b\|) * N) = 0.0042988$$

The following bar plots show result of HPL runs by CTBS:

Figure 6.1 HPL chart on Docker

6.2.2 DGEMM and FFT

As it is mentioned before, DGEMM measures the floating point rate of execution of double precision real matrix-matrix multiplication. The following is the result of StarDGEMM runs by HPCC part of CTBS:

Table 6.2 DGEMM result on Docker

Scaled residual	Minimum Gflop/s	Average Gflop/s	Maximum Gflop/s
0.0231783	1.498816	1.513326	1.520135

As it is mentioned before, FFT measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform. The following is the result of StarFFT runs by HPCC part of CTBS:

Table 6.3 StarFFT result on Docker

Vector size	Generation time	Computing	Inverse FFT	Minimum Gflop/s	Average Gflop/s	Maximum Gflop/s
524288	0.027	0.030	0.032	1.644084	1.669933	1.691132

The following bar plots show result of StarDGEMM and StarFFT:

Figure 6.2 StarDGEMM and StarFFT chart on Docker

6.2.3 Random Access

As it is mentioned before, Measures the rate of integer random updates of memory, Giga-updates per second (GUPS). The following is the result of StarRandomAccess_LCG runs by HPCC part of CTBS:

Table 6.4 StarRandomAccess_LCG result on Docker

Main table size	Number of updates	CPU time used	Real time used	Minimum GUP/s	Average GUP/s	Maximum GUP/s
2097152 Words	8388608	0.122981	0.123171	0.067817	0.068603	0.069324

The following bar plots show result of StarRandomAccess_LCG:

Figure 6.3 StarRandomAccess_LCG chart on Docker

6.2.4 STREAM and Communication Bandwidth

As it is mentioned before, STREAM is a simple synthetic benchmark program that measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernel. The following is the result of StarSTREAM runs by HPCC part of CTBS:

The system, which used to test, uses 8 bytes per DOUBLE PRECISION word. The benchmark run each test 10 times, but only the best time for each is used.

Table 6.5 StarSTREAM result on Docker

Function	Rate (GB/s)	Avg time	Min time	Max time
Copy	6.3992	0.0018	0.0017	0.0018
Scale	6.1996	0.0018	0.0018	0.0019
Add	6.8847	0.0025	0.0024	0.0025
Triad	7.1303	0.0024	0.0024	0.0024

As it is mentioned before, HPCC includes “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. The following is the result of PingPongBandwidth runs by HPCC part of CTBS.

Table 6.6 PingPongBandwidth result on Docker

Minimum Ping Pong Bandwidth (GBytes)	Average Ping Pong Bandwidth (GBytes)	Maximum Ping Pong Bandwidth (GBytes)
3.93739	5.25829	6.94709

The following bar plots show result of StarSTREAM_Triad and AvgPingPongBandwidth

Figure 6.4 StarSTREAM_Triad and AvgPingPongBandwidth chart on Docker

6.2.5 Communication Latency

As it is mentioned before, HPCC includes “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. The following is the result of PingPongLatency runs by HPCC part of CTBS:

Table 6.7 PingPongLatency result on Docker

Minimum Ping Pong Latency (usec)	Average Ping Pong Latency (usec)	Maximum Ping Pong Latency (usec)
0.357628	0.715847	1.00003

The following bar plots show result of AvgPingPongLatency:

Figure 6.5 AvgPingPongLatency chart on Docker

6.3 IOR Result

As it is mentioned before, IOR (Interleaved Or Random) is a benchmark program used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces [27].

CTBS benchmark run gathers the following performance information for the specified file system using three different file I/O APIs: POSIX Buffered-IO, POSIX DirectIO, and MPI-IO. Moreover, for each file I/O API, a set of five 5 throughput (thpt) and random IOPS (iops) operations are performed [38, p. 34].

6.3.1 Random IOPS

6.3.1.1 MPI-IO

As it is mentioned before, there are varieties of input arguments which allow variance of the overall I/O size, individual transfer size, file access mode (single shared file, one file per client), and whether the data is sequentially or randomly accessed. The following is the command passed for the test:

```
ior -b 100M -t 4K -z -a MPIIO -i 5
```

As a result, we have the following:

API: MPIIO v2	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in MPI-IO

Table 6.8 IOPS MPI-IO result on Docker

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	980.91	972.08	977.23
Read	2780.70	2724.42	2742.45

The following bar plots show result of IOR

Figure 6.6 IOPS MPI-IO chart on Docker

6.3.1.2 Posix Buffered IO

Basically, this is the same as the previous test. But, here the POSIX Buffered API is used. The following is the command passed for the test.

```
ior -b 100M -t 4K -z -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in POSIX Buffered IO

Table 6.9 IOPS POSIX Buffered IO result on Docker

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1049.34	1042.91	1046.19
Read	2948.98	2945.60	2947.46

The following bar plots show result of IOR

Figure 6.7 IOPS POSIX Buffered IO chart on Docker

6.3.1.3 Posix Direct IO

Basically, this is the same as the previous test. But, here the POSIX Direct API is used. The following is the command passed for the test.

```
ior -b 100M -t 4K -B -z -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in POSIX Direct IO

Table 6.10 IOPS POSIX Direct IO result on Docker

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	0.52	0.51	0.51
Read	0.95	0.87	0.92

The following bar plots show result of IOR

Figure 6.8 IOPS POSIX Direct IO chart on Docker

6.3.2 Throughput

6.3.2.1 MPI-IO

Basically, this is the same as the Random IOPS test. But, here the test is for Throughput. The following is the command passed for the test:

```
ior -b 1G -t 2M -a MPIIO -i 5
```

As a result, we have the following:

API: MPIIO v2	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in MPI-IO

Table 6.11 Throughput MPI-IO result on Docker

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1203.08	1186.97	1192.44
Read	5033.90	5024.63	5029.22

The following bar plots show result of IOR

Figure 6.9 Throughput MPI-IO chart on Docker

6.3.2.2 Posix Buffered IO

Basically, this is the same as the previous test. But, here the POSIX Buffered API is used. The following is the command passed for the test.

```
ior -b 1G -t 2M -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in POSIX Buffered IO

Table 6.12 Throughput POSIX Buffered IO result on Docker

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1269.08	1256.28	1263.83
Read	5028.93	5018.82	5024.96

The following bar plots show result of IOR

Figure 6.10 Throughput POSIX Buffered IO chart on Docker

6.3.2.3 Posix Direct IO

Basically, this is the same as the previous test. But, here the POSIX Direct API is used. The following is the command passed for the test.

```
ior -b 1G -t 2M -B -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in POSIX Direct IO

Table 6.13 Throughput POSIX Direct IO result on Docker

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	74.77	71.92	73.27
Read	129.27	106.62	114.41

The following bar plots show result of IOR

Figure 6.11 Throughput POSIX Direct IO chart on Docker

CHAPTER VII

BENCHMARKING VIRTUALIZED EXECUTION ENVIRONMENT

In this phase of research, we used the same node as the previous phase. So, it means that the node which was installed from the scratch with minimum base CentOS 6.6, is re-used in this phase. Then, the KVM is installed on the nodes. After that, a KVM virtual machine with minimum based CentOS 6.6 installed on the physical host. All the physical resources of the system was assigned to the virtual machine. It means that the KVM virtual machine had access to the all of the system's resources (CPUs, Memory). Then, the required packages which was necessary to compile CTBS was installed inside of the virtual machine. As it is mentioned before, since the required installed packaged are not a service to run in background, they do not have any overhead to the VM. So, we can consider the VM as the pure installation of base minimal CentOS 6.6.

7.1 System Configuration

As it is mentioned, the KVM virtual machine did not have any limitation in terms of the access to the system resources. It means that the VM can use all the CPUs and Memory of the system.

7.2 HPCC Result

As it was mentioned, HPCC (High Performance Computing Challenge), is a benchmarking suite which consists of 7 tests, HP, DGEMM, STREAM, PTRANS, FFT, "Communication Bandwidth and Latency", and "Random Access" [36].

7.2.1 HPL

As it discussed before, HPL is the High Performance version of LINPACK Benchmark which measures the floating point rate of execution for solving a linear system of equations. The following scaled residual check is computed:

$$\|Ax-b\| / (\text{eps} * (\|x\| * \|A\| + \|b\|) * N)$$

The matrix A is randomly generated for each test. The relative machine precision (eps) is taken to be 1.110223e-16. Computational tests pass if scaled residuals are less than 16.0.

The following is a short explanation of the input/output parameters follows:

- **T/V:** Wall time / encoded variant.
- **N:** The order of the coefficient matrix A.
- **NB:** The partitioning blocking factor.
- **P:** The number of process rows.
- **Q:** The number of process columns.
- **Time:** Time in seconds to solve the linear system.
- **Gflops:** Rate of execution for solving the linear system.

The following parameter values is used:

N: 4096	NBMIN: 4
NB: 128	NDIV: 2
PMAP: Row major process mapping	RFACT: Crout
P: 2	BCAST: 1ringM
Q: 4	DEPTH: 1
PFACT: Right	SWAP: Mix (threshold = 64)

L1: transposed form

EQUIL: yes

U: transposed form

ALIGN: 8 double precision word

The result of the linear system of equations:

Table 7.1 HPL result on KVM

T/V	N	NB	P	Q	Time	Gflop
WR11C2R4	4096	128	2	4	4.52	1.014e+01

$$\|Ax-b\| / (\text{eps} * (\|x\| * \|A\| + \|b\|) * N) = 0.0042988$$

The following bar plots show result of HPL runs by CTBS:

Figure 7.1 HPL chart on KVM

7.2.2 DGEMM and FFT

As it is mentioned before, DGEMM measures the floating point rate of execution of double precision real matrix-matrix multiplication. The following is the result of StarDGEMM runs by HPCC part of CTBS:

Table 7.2 DGEMM result on KVM

Scaled residual	Minimum Gflop/s	Average Gflop/s	Maximum Gflop/s
0.0237793	1.309540	1.393274	1.485902

As it is mentioned before, FFT measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform. The following is the result of StarFFT runs by HPCC part of CTBS:

Table 7.3 StarFFT result on KVM

Vector size	Generation time	Computing	Inverse FFT	Minimum Gflop/s	Average Gflop/s	Maximum Gflop/s
524288	0.028	0.032	0.034	1.484929	1.537678	1.598433

The following bar plots show result of StarDGEMM and StarFFT:

Figure 7.2 StarDGEMM and StarFFT chart on KVM

7.2.3 Random Access

As it is mentioned before, Measures the rate of integer random updates of memory, Giga-updates per second (GUPS). The following is the result of StarRandomAccess_LCG runs by HPCC part of CTBS:

Table 7.4 StarRandomAccess_LCG result on KVM

Main table size	Number of updates	CPU time used	Real time used	Minimum GUP/s	Average GUP/s	Maximum GUP/s
2097152 Words	8388608	0.142978	0.143332	0.051127	0.058636	0.064864

The following bar plots show result of StarRandomAccess_LCG:

Figure 7.3 StarRandomAccess_LCG chart on KVM

7.2.4 STREAM and Communication Bandwidth

As it is mentioned before, STREAM is a simple synthetic benchmark program that measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernel. The following is the result of StarSTREAM runs by HPCC part of CTBS:

The system, which used to test, uses 8 bytes per DOUBLE PRECISION word. The benchmark run each test 10 times, but only the best time for each is used.

Table 7.5 StarSTREAM result on KVM

Function	Rate (GB/s)	Avg time	Min time	Max time
Copy	5.3879	0.0021	0.0021	0.0022
Scale	5.3468	0.0022	0.0021	0.0023
Add	5.8538	0.0029	0.0029	0.0029
Triad	7.3263	0.0028	0.0023	0.0029

As it is mentioned before, HPCC includes “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. The following is the result of PingPongBandwidth runs by HPCC part of CTBS:

Table 7.6 PingPongBandwidth result on KVM

Minimum Ping Pong Bandwidth (GBytes)	Average Ping Pong Bandwidth (GBytes)	Maximum Ping Pong Bandwidth (GBytes)
3.70685	5.13036	6.94421

The following bar plots show result of StarSTREAM_Triad and AvgPingPongBandwidth

Figure 7.4 StarSTREAM_Triad and AvgPingPongBandwidth chart on KVM

7.2.5 Communication Latency

As it is mentioned before, HPCC includes “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. The following is the result of PingPongLatency runs by HPCC part of CTBS:

Table 7.7 PingPongLatency result on KVM

Minimum Ping Pong Latency (usec)	Average Ping Pong Latency (usec)	Maximum Ping Pong Latency (usec)
0.384119	0.722263	1.00003

The following bar plots show result of AvgPingPongLatency:

Figure 7.5 AvgPingPongLatency chart on KVM

7.3 IOR Result

As it is mentioned before, IOR (Interleaved Or Random) is a benchmark program used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces [27].

CTBS benchmark run gathers the following performance information for the specified file system using three different file I/O APIs: POSIX Buffered-IO, POSIX DirectIO, and MPI-IO. Moreover, for each file I/O API, a set of five 5 throughput (thpt) and random IOPS (iops) operations are performed [38, p. 34].

7.3.1 Random IOPS

7.3.1.1 MPI-IO

As it is mentioned before, there are varieties of input arguments which allow variance of the overall I/O size, individual transfer size, file access mode (single shared file, one file per client), and whether the data is sequentially or randomly accessed. The following is the command passed for the test:

```
ior -b 100M -t 4K -z -a MPIIO -i 5
```

As a result, we have the following:

API: MPIIO v2	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in MPI-IO

Table 7.8 IOPS MPI-IO result on KVM

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	904.16	897.36	900.67
Read	2147.72	2054.52	2084.78

The following bar plots show result of IOR

Figure 7.6 IOPS MPI-IO chart on KVM

7.3.1.2 Posix Buffered IO

Basically, this is the same as the previous test. But, here the POSIX Buffered API is used. The following is the command passed for the test.

```
ior -b 100M -t 4K -z -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in POSIX Buffered IO

Table 7.9 IOPS POSIX Buffered IO result on KVM

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1007.78	937.79	988.13
Read	2219.31	2166.42	2195.53

The following bar plots show result of IOR

Figure 7.7 IOPS POSIX Buffered IO chart on KVM

7.3.1.3 Posix Direct IO

Basically, this is the same as the previous test. But, here the POSIX Direct API is used. The following is the command passed for the test.

```
ior -b 100M -t 4K -B -z -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 4096 bytes
Ordering in a file: random offsets	Block size: 100 MiB
Ordering inter file: no tasks offsets	Aggregate file size: 100 MiB

The following is the result of IOR for Random IOPS in POSIX Direct IO

Table 7.10 IOPS POSIX Direct IO result on KVM

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	0.52	0.50	0.51
Read	0.92	0.89	0.91

The following bar plots show result of IOR

Figure 7.8 IOPS POSIX Direct IO chart on KVM

7.3.2 Throughput

7.3.2.1 MPI-IO

Basically, this is the same as the Random IOPS test. But, here the test is for Throughput. The following is the command passed for the test:

```
ior -b 1G -t 2M -a MPIIO -i 5
```

As a result, we have the following:

API: MPIIO v2	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in MPI-IO

Table 7.11 Throughput MPI-IO result on KVM

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1236.72	1214.47	1223.93
Read	3500.94	3480.33	3490.07

The following bar plots show result of IOR

Figure 7.9 Throughput MPI-IO chart on KVM

7.3.2.2 Posix Buffered IO

Basically, this is the same as the previous test. But, here the POSIX Buffered API is used. The following is the command passed for the test.

```
ior -b 1G -t 2M -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in POSIX Buffered IO

Table 7.12 Throughput POSIX Buffered IO result on KVM

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	1333.30	1047.72	1264.58
Read	4082.74	3987.51	4022.34

The following bar plots show result of IOR

Figure 7.10 Throughput POSIX Buffered IO chart on KVM

7.3.2.3 Posix Direct IO

Basically, this is the same as the previous test. But, here the POSIX Direct API is used. The following is the command passed for the test.

```
ior -b 1G -t 2M -B -i 5
```

As a result, we have the following:

API: POSIX	Repetitions: 5
Access: single-shared-file	Xfersize: 2 MiB
Ordering in a file: sequential offsets	Block size: 1 GiB
Ordering inter file: no tasks offsets	Aggregate file size: 1 GiB

The following is the result of IOR for Throughput in POSIX Direct IO

Table 7.13 Throughput POSIX Direct IO result on KVM

Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Write	52.66	51.88	52.27
Read	128.07	119.88	123.64

The following bar plots show result of IO

Figure 7.11 Throughput POSIX Direct IO chart on KVM

CHAPTER VIII

ANALYSIS OF BENCHMARKING RESULTS ON DIFFERENT EXECUTION ENVIRONMENTS

In this chapter, we analyze and compare benchmarking results on three different execution environments, including bare-metal, containerized (Docker), and virtualized (KVM) execution environments. As the ultimate goal of this research, we try to answer the question about the difference of the performance of execution environments, especially in IaaS layer of cloud. In another word, we try to compare the performance of bare-metal versus Linux containers versus virtual machines.

In general, in this research we focus on the benchmarking of IaaS layer of cloud which Figure 8.1 represents it. Three different execution environments ultimately have practical usage to build the IaaS layer of cloud. We test a system in bare-metal mode versus Docker (as the representative of Linux containers) versus KVM (as the representative of virtual machines). In this research, we installed the minimum base CentOS 6.6 on a node as the bare-metal. Then, we installed Docker on the node as Linux containers. After that, we installed KVM as virtual machines. We have used Cloud Tester Benchmark Suite (CTBS), which is provided to CAC by AERPSPACE CORPORATION through the technical partnership, to measure the performance of the node in the three execution environments.

As it was mentioned before, all the execution environment were installed from the scratch. Moreover, to minimize the variance of the result of the benchmarks, the minimum base distribution of CentOS 6.6 was used. Then, all the required packages which was necessary to compile CTBS were installed on the each environment. Since the required installed packages are not a service to run in background, they do not have any impact on the system. So, we can consider each environment as the pure minimal installation of base minimal CentOS 6.6.

Figure 8.1 NIST Cloud Service Models, CTBS Activity Focus [37]

8.1 HPCC Comparison

As discussed before, HPCC (High Performance Computing Challenge), is a benchmarking suite which consists of 7 tests, HP, DGEMM, STREAM, PTRANS, FFT, “Communication Bandwidth and Latency”, and “Random Access” [36].

8.1.1 HPL

HPL is the High Performance version of LINPACK Benchmark which measures the floating point rate of execution for solving a linear system of equations. The following scaled residual check is computed:

$$\|Ax-b\| / (\text{eps} * (\|x\| * \|A\| + \|b\|) * N)$$

The matrix A is randomly generated for each test. The relative machine precision (eps) is taken to be 1.110223e-16. Computational tests pass if scaled residuals are less than 16.0.

The result of the linear system of equations:

Table 8.1 HPL result, Bare-metal vs. Docker vs. KVM

	T/V	N	NB	P	Q	Time	Gflop
Bare-metal	WR11C2R4	4096	128	2	4	4.42	1.038e+01
Docker	WR11C2R4	4096	128	2	4	4.41	1.038e+01
KVM	WR11C2R4	4096	128	2	4	4.52	1.014e+01

$$\|Ax-b\| / (\text{eps} * (\|x\| * \|A\| + \|b\|) * N) = 0.0042988$$

The following bar plots show result of HPL runs by CTBS:

Figure 8.2 HPL chart, BareMetal vs. Docker vs. KVM

8.1.2 DGEMM and FFT

As it is mentioned before, DGEMM measures the floating point rate of execution of double precision real matrix-matrix multiplication. The following is the result of StarDGEMM runs by HPCC part of CTBS:

Table 8.2 DGEMM result, Bare-metal vs. Docker vs. KVM

	Scaled residual	Minimum Gflop/s	Average Gflop/s	Maximum Gflop/s
Bare-metal	0.0181865	1.488568	1.513195	1.524991
Docker	0.0231783	1.498816	1.513326	1.520135
KVM	0.0237793	1.309540	1.393274	1.485902

As it is mentioned before, FFT measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform. The following is the result of StarFFT runs by HPCC part of CTBS:

Table 8.3 StarFFT result, Bare-metal vs. Docker vs. KVM

	Generation time	Computing	Inverse FFT	Min Gflop/s	Ave Gflop/s	Max Gflop/s
Bare Metal	0.027	0.030	0.032	1.620491	1.657962	1.681833
Docker	0.027	0.030	0.032	1.644084	1.669933	1.691132
KVM	0.028	0.032	0.034	1.484929	1.537678	1.598433

The following bar plots show result of StarDGEMM and StarFFT:

Figure 8.3 StarDGEMM and StarFFT chart, BareMetal vs. Docker vs. KVM

8.1.3 Random Access

As it is mentioned before, Measures the rate of integer random updates of memory, Giga-updates per second (GUPS). The following is the result of StarRandomAccess_LCG runs by HPCC part of CTBS:

Table 8.4 StarRandomAccess_LCG result, Bare-metal vs. Docker vs. KVM

	Main table size	Number of updates	CPU time used	Real time used	Min GUP/s	Ave GUP/s	Max GUP/s
Bare Metal	2097152 Words	8388608	0.122981	0.123470	0.067555	0.068617	0.069307
Docker	2097152 Words	8388608	0.122981	0.123171	0.067817	0.068603	0.069324
KVM	2097152 Words	8388608	0.142978	0.143332	0.051127	0.058636	0.064864

The following bar plots show result of StarRandomAccess_LCG:

Figure 8.4 StarRandomAccess_LCG chart, BareMetal vs. Docker vs. KVM

8.1.4 STREAM and Communication Bandwidth

As it is mentioned before, STREAM is a simple synthetic benchmark program that measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernel. The following is the result of StarSTREAM runs by HPCC part of CTBS:

The system, which used to test, uses 8 bytes per DOUBLE PRECISION word. The benchmark run each test 10 times, but only the best time for each is used.

Table 8.5 StarSTREAM result, Bare-metal vs. Docker vs. KVM

	Function	Rate (GB/s)	Avg time	Min time	Max time
Bare Metal	Copy	6.4097	0.0018	0.0017	0.0019
	Scale	6.3335	0.0018	0.0018	0.0019
	Add	6.8339	0.0025	0.0025	0.0025
	Triad	7.2255	0.0024	0.0023	0.0024
Docker	Copy	6.3992	0.0018	0.0017	0.0018
	Scale	6.1996	0.0018	0.0018	0.0019
	Add	6.8847	0.0025	0.0024	0.0025
	Triad	7.1303	0.0024	0.0024	0.0024
KVM	Copy	5.3879	0.0021	0.0021	0.0022
	Scale	5.3468	0.0022	0.0021	0.0023
	Add	5.8538	0.0029	0.0029	0.0029
	Triad	7.3263	0.0028	0.0023	0.0029

As it is mentioned before, HPCC includes “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. The following is the result of PingPongBandwidth runs by HPCC part of CTBS:

Table 8.6 PingPongBandwidth result, Bare-metal vs. Docker vs. KVM

	Minimum Ping Pong Bandwidth (GBytes)	Average Ping Pong Bandwidth (GBytes)	Maximum Ping Pong Bandwidth (GBytes)
Bare Metal	3.90622	5.25721	6.97018
Docker	3.93739	5.25829	6.94709
KVM	3.70685	5.13036	6.94421

The following bar plots show result of StarSTREAM_Triad and AvgPingPongBandwidth

Figure 8.5 StarSTREAM_Triad and AvgPingPongBandwidth chart, BareMetal vs. Docker vs. KVM

8.1.5 Communication Latency

As it is mentioned before, HPCC includes “Communication Bandwidth and Latency” which is a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. The following is the result of PingPongLatency runs by HPCC part of CTBS:

Table 8.7 PingPongLatency result, Bare-metal vs. Docker vs. KVM

	Minimum Ping Pong Latency (usec)	Average Ping Pong Latency (usec)	Maximum Ping Pong Latency (usec)
Bare Metal	0.357628	0.711649	1.05964
Docker	0.357628	0.715847	1.00003
KVM	0.384119	0.722263	1.00003

The following bar plots show result of AvgPingPongLatency:

Figure 8.6 AvgPingPongLatency chart, BareMetal vs. Docker vs. KVM

8.2 IOR Result

As it is mentioned before, IOR (Interleaved Or Random) is a benchmark program used for benchmarking parallel file systems using POSIX, MPIIO, or HDF5 interfaces [27].

CTBS benchmark run gathers the following performance information for the specified file system using three different file I/O APIs: POSIX Buffered-IO, POSIX DirectIO, and MPI-IO. Moreover, for each file I/O API, a set of five 5 throughput (thpt) and random IOPS (iops) operations are performed [38, p. 34].

8.2.1 Random IOPS

8.2.1.1 MPI-IO

As it is mentioned before, there are varieties of input arguments which allow variance of the overall I/O size, individual transfer size, file access mode (single shared file, one file per client), and whether the data is sequentially or randomly accessed. The following is the command passed for the test:

```
ior -b 100M -t 4K -z -a MPIIO -i 5
```

The following is the result of IOR for Random IOPS in MPI-IO

Table 8.8 IOPS MPI-IO result, Bare-metal vs. Docker vs. KVM

	Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Bare-metal	Write	1004.03	804.17	963.51
	Read	2806.70	2793.54	2800.30
Docker	Write	980.91	972.08	977.23
	Read	2780.70	2724.42	2742.45
KVM	Write	904.16	897.36	900.67
	Read	2147.72	2054.52	2084.78

The following bar plots show result of IOR

Figure 8.7 IOPS MPI-IO chart, BareMetal vs. Docker vs. KVM

8.2.1.2 Posix Buffered IO

Basically, this is the same as the previous test. But, here the POSIX Buffered API is used. The following is the command passed for the test.

```
ior -b 100M -t 4K -z -i 5
```

The following is the result of IOR for Random IOPS in POSIX Buffered IO

Table 8.9 IOPS POSIX Buffered IO result, Bare-metal vs. Docker vs. KVM

	Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Bare-metal	Write	1081.40	959.84	1055.16
	Read	2999.49	2991.15	2994.07
Docker	Write	1049.34	1042.91	1046.19
	Read	2948.98	2945.60	2947.46
KVM	Write	1007.78	937.79	988.13
	Read	2219.31	2166.42	2195.53

The following bar plots show result of IOR

Figure 8.8 IOPS POSIX Buffered IO chart, BareMetal vs. Docker vs. KVM

8.2.1.3 Posix Direct IO

Basically, this is the same as the previous test. But, here the POSIX Direct API is used. The following is the command passed for the test.

```
ior -b 100M -t 4K -B -z -i 5
```

The following is the result of IOR for Random IOPS in POSIX Direct IO

Table 8.10 IOPS POSIX Direct IO result, Bare-metal vs. Docker vs. KVM

	Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Bare-metal	Write	0.56	0.49	0.51
	Read	0.95	0.72	0.89
Docker	Write	0.52	0.51	0.51
	Read	0.95	0.87	0.92
KVM	Write	0.52	0.50	0.51
	Read	0.92	0.89	0.91

The following bar plots show result of IOR

Figure 8.9 IOPS POSIX Direct IO chart, BareMetal vs. Docker vs. KVM

8.2.2 Throughput

8.2.2.1 MPI-IO

Basically, this is the same as the Random IOPS test. But, here the test is for Throughput. The following is the command passed for the test:

```
ior -b 1G -t 2M -a MPIIO -i 5
```

As a result, we have the following:

The following is the result of IOR for Throughput in MPI-IO

Table 8.11 Throughput MPI-IO result, Bare-metal vs. Docker vs. KVM

	Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Bare-metal	Write	1320.67	1301.00	1316.42
	Read	5043.19	5026.46	5038.50
Docker	Write	1203.08	1186.97	1192.44
	Read	5033.90	5024.63	5029.22
KVM	Write	1236.72	1214.47	1223.93
	Read	3500.94	3480.33	3490.07

The following bar plots show result of IOR

Figure 8.10 Throughput MPI-IO chart, BareMetal vs. Docker vs. KVM

8.2.2.2 Posix Buffered IO

Basically, this is the same as the previous test. But, here the POSIX Buffered API is used. The following is the command passed for the test.

```
ior -b 1G -t 2M -i 5
```

The following is the result of IOR for Throughput in POSIX Buffered IO

Table 8.12 Throughput POSIX Buffered IO result, Bare-metal vs. Docker vs. KVM

	Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Bare-metal	Write	1318.04	1314.92	1316.78
	Read	5045.42	5041.46	5042.92
Docker	Write	1269.08	1256.28	1263.83
	Read	5028.93	5018.82	5024.96
KVM	Write	1333.30	1047.72	1264.58
	Read	4082.74	3987.51	4022.34

The following bar plots show result of IOR

Figure 8.11 Throughput POSIX Buffered IO chart, BareMetal vs. Docker vs. KVM

8.2.2.3 Posix Direct IO

Basically, this is the same as the previous test. But, here the POSIX Direct API is used. The following is the command passed for the test.

```
ior -b 1G -t 2M -B -i 5
```

The following is the result of IOR for Random IOPS in POSIX Direct IO

Table 8.13 Throughput POSIX Direct IO result, Bare-metal vs. Docker vs. KVM

	Operation	Max(MiB)	Min(MiB)	Mean(MiB)
Bare-metal	Write	74.57	72.30	73.67
	Read	136.75	108.91	118.95
Docker	Write	74.77	71.92	73.27
	Read	129.27	106.62	114.41
KVM	Write	52.66	51.88	52.27
	Read	128.07	119.88	123.64

The following bar plots show result of IO

Figure 8.12 Throughput POSIX Direct IO chart, BareMetal vs. Docker vs. KVM

8.3 Benchmark Analysis and Findings

In this section the results of the benchmarking analysis are presented. The benchmarks were run and the results were collected and then processed in response to the discussion posed in Chapter 1 of this research. Two fundamental goals drove the collection of the result of benchmarks and the subsequent benchmark analysis. Those goals were to develop a base of knowledge about the technology of Docker and KVM, as it is perceived and utilized relative to other virtualization methods, and to determine if the current perception of the performance of these execution environments are consistent and correct. These objectives were accomplished.

8.3.1 HPCC Performance Comparison

As it discussed in the previous chapters, the results of HPCC is the representation of compute and interconnect performance of systems. The following is the radar chart which compares the HPCC KPMs (Table 5.1) for all the CTBS benchmark runs in bare-metal, Docker, and KVM. The radar plot is a normalized chart with the best (highest) result from all the data files assigned a 1.0 value and the other values normalized to this result.

Figure 8.13 HPCC Performance Comparison, BareMetal vs. Docker vs. KVM

As the Figure 8.13 shows, In general, the performance of the Bare-metal closely overlaps with the performance of Docker. On the other hand, KVM has less performance in comparison with the other two execution environments.

The top vertex of the pentagon shape (Figure 8.13) is the representative of HPL benchmarks. As HPL is the High Performance version of LINPACK, the vast majority of compute operations are spent in double-precision floating point multiplication of a scalar with a vector and adding the results to another vector [39]. As the result, HPL execution spends the most of its time on performing mathematical floating point operations. It means that the execution has fairly regular memory accesses and mainly stresses the floating point capability of the core. As the Figure 8.13 shows, performance is almost identical on both bare-metal and Docker. However, the KVM performance is 2.33% worse than the bare-metal. This shows the costs of abstracting and hiding the nature of the system from the execution environment. Such behavior is expected to be normal for other similarly executions, unless the system topology is truly carried forth into the virtualized environment. In general, CPU-bound programs that don't attempt such tuning will likely have equal but equally poor performance across bare-metal, Docker, and KVM [39].

Similar to HPL, DGEMM and FFT spend the most of their time on performing mathematical floating point operations, respectively, on double precision real matrix-matrix multiplication and double precision complex one-dimensional Discrete Fourier Transform (DFT). It means that their execution has limited regular memory access like HPL. However, since their execution require more access to memory than those of HPL, they have more memory access than HPL, but it is still limited. As the result, they have relatively worse performance than HPL. If we consider the bare-metal performance as the base reference, StarDGEMM and StarFFT have respectively 7.93% and 7.25% worse Gflops which again represents the costs of hiding system information from the execution environments.

The bottom right vertex of the pentagon shape (Figure 8.13) is the representative of STREAM benchmarks which measures sustainable memory bandwidth when

performing simple operations on vectors [40]. The benchmark execution is dominated by the memory bandwidth and the benchmark's workload is significantly larger than caches. The benchmarks has regular memory access pattern. On the other hand, the hardware prefetchers typically latch on to the access pattern and prefetch data before it is needed. As the result, the performance of the benchmark is gated by memory bandwidth and not latency [39]. As it was mention in Table 8.5, the benchmark consists of four components COPY, SCALE, ADD, and TRIAD. These components compute the value of $a[i] = b[i]$, $a[i] = q * b[i]$, $a[i] = b[i] + c[i]$, and $a[i] = b[i] + q * c[i]$ respectively. As it can be extracted from Table 8.5, performance is almost identical on both bare-metal and Docker. However, the KVM performance is markedly worse than the bare-metal especially in COPY, SCALE, and ADD parts. If we consider the result of bare-metal as the base reference, the GB/s rate of each components for KVM are respectively 15.94%, 15.58%, and 14.34% worse. Although there are significant performance overhead in COPY, SCALE, and ADD components, there is only 1.40% slowdown in TRIAD component which needs more research to find the reason. However, the overall performance overhead in KVM shows the costs of adding another layer of abstraction to the execution environment.

The last benchmark of HPCC is Random Access which is in the bottom left vertex of the pentagon shape (Figure 8.13). In contrast to STREAM benchmark which has regular memory access pattern, Random Access stresses random memory pattern. Similar to the STREAM benchmark, the workload is significantly larger than caches. The benchmark strongly exercises the hardware page table walker that handles TLB misses [39] (TLB is Translation Look aside Buffer which is a cache that memory management hardware uses to improve virtual address translation speed [41]). As the result, the result of the benchmark is determined by the latency of the hardware page table walks and main memory load latency. Since there is one hardware page table walks for bare-metal and Docker, the results are somehow identical for these two execution environments. On the other hand, for KVM, there are two layer, one hardware page table walk for the guest and another hardware page table walk for host. So, this explains

the reason why KVM suffer 14.55% less Giga updates per second (GUPs) in comparison with bare-metal.

8.3.2 IOR Performance Comparison

8.3.2.1 Random IOPS

The following is the radar chart which compares the IOR KPMs for the three different file I/O APIs, POSIX Buffered-IO, POSIX DirectIO, and MPI-IO of running the Random IOPS operations in bare-metal, Docker, and KVM. The radar plot is a normalized chart with the best (highest) result from all the data files assigned a 1.0 value and the other values normalized to this result.

Figure 8.14 IOR - Random IOPS Performance Comparison, BareMetal vs. Docker vs. KVM

As the Figure 8.14 shows, in general, the performance of the bare-metal closely overlaps with the performance of Docker. Even in some cases like Posix Direct IO API for Read operation, Docker shows better performance than bare-metal which seems to be a noise. On the other hand, KVM has less performance in comparison with the other two execution environments, especially in Read operations.

IOR is used for testing performance of parallel file systems using various interfaces and access patterns. The top right vertexes of the pentagon shape (Figure 8.14) are the representative of IOR benchmarks for MPI-IO, Posix Buffered IO, and Posix Direct IO APIs in Read operation and the bottom left vertexes of the are for Write operation with the same APIs. As the shows, KVM is relatively weak in Read IO performance. For Random Read in KVM execution environment, we observed that there were 25.55% and 26.67%, reduction in mean achievable IOPS respectively for MPI-IO and Posix Buffered IO APIs when compared to the bare-metal case. However, there were 2.25% increase in performance for the Posix Direct IO API. Although the benchmark of each APIs are run 5 times, it seems that this increase in performance is a noise for the benchmarks.

While the result shows that KVM does not perform well for Read IO, we observed almost the same performance for the Random Write operating in all the three APIs compared to the bare-metal configuration. Moreover, we observed almost identical performance on both bare-metal and Docker within the noise margin of 2%. If we consider the performance of bare-metal as the base reference, Table 8.1 shows the percentage of slowdown and speedup of the performance of the each APIs for Read and Write operations.

Table 8.14 Random IOPS Percentage of Slowdown and Speedup Based on Bare-Metal: Docker vs KVM

	Read			Write		
	MPI-IO	Posix Buffered IO	Posix Direct IO	MPI-IO	Posix Buffered IO	Posix Direct IO
Docker	-2.07%	-1.51%	3.37%	1.42%	-0.55%	0.00%
KVM	-25.55%	-26.67%	2.25%	-6.52%	-6.35%	0.00%

8.3.2.2 Throughput

The following is the radar chart which compares the IOR KPMs for the three different file I/O APIs, POSIX Buffered-IO, POSIX DirectIO, and MPI-IO of running the Throughput operations in bare-metal, Docker, and KVM. The radar plot is a normalized chart with the best (highest) result from all the data files assigned a 1.0 value and the other values normalized to this result.

Figure 8.15 IOR - Throughput Performance Comparison, BareMetal vs. Docker vs. KVM

The differences between Random IOPS (the previous result) and Throughput are the block size and the ordering in benchmark file. In Random IOPS the ordering in the file is random offsets. On the other hand, this value for the Throughput is sequential offsets. Moreover, block size in Random IOPS is 100 MiB. But, Throughput has the block size of 1 GiB.

As the Figure 8.15 shows, in general, the performance of the Bare-metal closely overlaps with the performance of Docker. On the other hand, KVM has less performance in comparison with the other two execution environments, especially in Read operations of MPI-IO and Posix Buffered IO APIs.

The top right vertexes of the pentagon shape (Figure 8.15) are the representative of IOR benchmarks for MPI-IO, Posix Buffered IO, and Posix Direct IO APIs in Read operation and the bottom left vertexes of the are for Write operation with the same APIs. As the shows, KVM is markedly weak in Read IO performance. But, for Write operating, although KVM showed identical performance in Random IOPS test in comparison with both bare-metal and Docker, this time, KVM is significantly weak in Posix Direct IO API. It means that the changes in “order of the benchmark file” and “block size” effect the performance of Write operating for Posix Direct IO API. This time, KVM shows 29.05% slowdown in comparison with bare-metal for Write operating. This is while the percentage was 0% in Random IOPS test.

For Throughput Read in KVM execution environment, we observed that there were 30.73% and 20.24%, reduction in mean achievable IOPS respectively for MPI-IO and Posix Buffered IO APIs when compared to the bare-metal case. However, there were 3.94% increase in performance for the Posix Direct IO API. Although the benchmark of each APIs are run 5 times, it seems this increase in performance is a noise.

We again observed almost the same performance for the Throughput Write operating of MPI-IO and Posix Buffered IO in KVM compared to the bare-metal. We have observed almost identical performance on both bare-metal and Docker. If we consider the performance of bare-metal as the base reference, Table 8.1 shows the percentage of slowdown and speedup of the performance of the each APIs for Read and Write operations.

Table 8.15 Throughput Percentage of Slowdown and Speedup Based on Bare-Metal: Docker vs KVM

	Read			Write		
	MPI-IO	Posix Buffered IO	Posix Direct IO	MPI-IO	Posix Buffered IO	Posix Direct IO
Docker	-0.18%	-0.36%	-3.82%	-9.42%	-4.02%	-0.54%
KVM	-30.73%	-20.24%	3.94%	-7.03%	-3.96%	-29.05%

CHAPTER IX

CONCLUSION AND FUTURE WORK

9.1 Conclusion

For a long time, cloud administrators have had multiple virtualization options to choose from. Recently, Linux containers, especially Docker container has emerged as another virtualization methods. According to the IBM report [39], this method could be one of the practical choice regarding performance. Although the basic premise behind containers is an easier delivery approach, containers present a performance boost as well.

In this research, we used Cloud Tester Benchmark Suite which is benchmark suite for IaaS layer of cloud including HPCC and IOR to evaluate the performance of systems in three different execution environments, bare-metal versus Docker versus KVM. As the analysis of benchmarking results showed in chapter 8, there are differences between the performances of these environments. In most of the benchmarks, we observed that Docker equals or exceed the performance of KVM. In fact, in most of the tests, we observed almost identical performance on both bare-metal and Docker. Using the HPCC benchmark, we measured the impact of virtualization and found Docker containers to be the clear winner. We found that Docker delivered near bare-metal performance while KVM performance was relevantly less, especially in Random Access benchmark that KVM suffers 14.55% less Giga updates per second (GUPs) in comparison with bare-metal. Moreover, we found that since KVM adds some overhead to I/O operation, KVM is less suitable for workloads that are latency-sensitive or have high I/O rates. For example, we observed 25.55% and 26.67%, reduction in mean achievable IOPS respectively for MPI-IO and Posix Buffered IO APIs when compared to the bare-metal.

Since containers offer the control and isolation of virtual machines with the performance of bare-metal, cloud administrators can build the IaaS layer of their cloud based on containers. Some features such as faster deployment and near bare-metal performance

can be reasonable enough to use containers as the building block of Infrastructure-as-a-Service of a cloud. However, virtual machines has been working for a long time and features like Live Migration, Over Allocation, and Snapshots can be challenging for container to be substituted for virtual machines. As a results, there is a tradeoff between ease of management and performance.

9.2 Future Work

As future work, the performance information derived from the Cloud Tester Benchmark Suite can also be used by application developers to assist in performance tuning efforts and to help determine when and how best to use co-processors (e.g. GPUs, Intel Xeon Phi, etc.) and other specialized computer hardware. Researchers can use CTBS to evaluate the performance of Xeon Phi architecture versus GPUs (like NVIDIA Tesla)

Moreover, the CTBS benchmark can be used with different workload to have a better perception of the pattern of the performance of each execution environments based on different workload. It means that researcher can run CTBS, especially IOR benchmark with less and more workload than what has been done in this research. Then, we may be able to define a pattern of performance depends of the amount of workload for each of the environment.

In addition to this, the Cloud Tester package is intended for use in IaaS layer of cloud, especially for IaaS systems which provide HPC-like resources, e.g. Amazon EC2 and DODCS Computer Cluster instances. So, more performance testing can be done in Cloud point of view by Cloud Tester Benchmark Suite. It means that the similar performance testing can be done in OpenStack or Amazon EC2 as well.

Moreover, the new version of CTBS, v2.0.0, are available by the Aerospace Corporation to test the IaaS layer of a cloud. The new version includes Cloud Barometer which is a collection of cloud benchmarking software, including the Cloud Tester Benchmark Suite (CTBS) and the Application System Resource Profiling and Usage Prediction Tools (TraceApp). Researchers can use the new version to evaluate the

performance IaaS layer of cloud to have a better perception of systems to tune them based on needs.

REFERENCES

- [1] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Cloud_computing. [Accessed 26 February 2015].
- [2] H. Boothe, "Virtual Command," Virtual Command, LLC, [Online]. Available: <http://www.virtualcommand.com/virtualization-cloud-computing-difference/>. [Accessed 26 February 2015].
- [3] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Hardware_virtualization. [Accessed 26 February 2015].
- [4] J. Smith and R. Nair, "The architecture of virtual machines," *Computer (IEEE Computer Society)*, vol. 38, no. 5, pp. 32 - 38, May 2005.
- [5] "Docker," [Online]. Available: <https://www.docker.com>. [Accessed 26 February 2015].
- [6] "GitHub," [Online]. Available: <https://github.com/coreos/rocket>. [Accessed 26 February 2015].
- [7] "CoreOS," [Online]. Available: <https://coreos.com>. [Accessed 26 February 2015].
- [8] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology, 2011.
- [9] D. Marshall, "InfoWorld," 2 November 2011. [Online]. Available: <http://www.infoworld.com/article/2621446/server-virtualization/top-10-benefits-of-server-virtualization.html>. [Accessed 10 October 2013].
- [10] G. Popek and R. Goldberg, "Formal requirements for virtualizable third generation architectures," *Communications of the ACM*, vol. 17, no. 7, pp. 412-421, 01 07 1974.
- [11] B. Pariseau, "Search Server Virtualization," [Online]. Available: <http://searchservervirtualization.techtarget.com/news/2240034817/KVM-reignites-Type-1-vs-Type-2-hypervisor-debate>. [Accessed 15 April 2011].
- [12] "Wikipedia," [Online]. Available: <http://en.wikipedia.org/wiki/LXC>. [Accessed 26 February 2015].

- [13] "Linux Containers," [Online]. Available:
<https://linuxcontainers.org/lxc/introduction/>. [Accessed 26 February 2015].
- [14] J. Corbet, "LWN," May 29 2007. [Online]. Available:
<http://lwn.net/Articles/236038/>. [Accessed 26 February 2015].
- [15] J. Corbet, "LWN," 29 October 2007. [Online]. Available:
<http://lwn.net/Articles/256389/>. [Accessed 26 February 2015].
- [16] J. Corbet, "LWN," 31 July 2007. [Online]. Available:
<http://lwn.net/Articles/243795/>. [Accessed 26 February 2015].
- [17] B. Singh and V. Srinivasan, "Containers: Challenges with the memory resource controller and its performance," in *Ottawa Linux Symposium*, Ottawa, 2007.
- [18] J. Corbet, "Network World," 23 October 2007. [Online]. Available:
<http://www.networkworld.com/article/2287247/software/kernel-space--fair-user-scheduling-for-linux.html>. [Accessed 26 February 2015].
- [19] K. Hiroyu, "Cgroup and Memory Resource Controller," in *Japan Linux Symposium*, 2008.
- [20] D. Hansen, "Resource Management," [Online]. Available:
http://events.linuxfoundation.org/slides/lfcs09_hansen2.pdf. [Accessed 26 February 2015].
- [21] "CoreOS," CoreOS, Inc, [Online]. Available: <https://coreos.com/using-coreos/containers>. [Accessed 26 February 2015].
- [22] "Docker," Docker, Inc, [Online]. Available: <https://docs.docker.com/faq/>. [Accessed 26 February 2015].
- [23] B. Mizerany, "The New Stack," June 2013. [Online]. Available:
<http://thenewstack.io/about-etcd-the-distributed-key-value-store-used-for-kubernetes-googles-cluster-container-manager/>. [Accessed 26 February 2015].
- [24] "CoreOS," CoreOS, Inc, [Online]. Available: <https://coreos.com/using-coreos/etcd/>. [Accessed 26 February 2015].
- [25] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl and C. Tosun, "Benchmarking in the Cloud: What it Should, Can, and Cannot Be," in *Selected Topics in Performance Evaluation and Benchmarking*, Springer.

- [26] D. Enright, A. Brethorst, J. Everist, R. Scrofano and L. Wang, "Cloud Benchmarking: The Ongoing Evolution of Computer System Evaluation," in *Mini-Symposium, 2013 SIAM Annual Meeting*, 2013.
- [27] "The National Energy Research Scientific Computing Center (NERSC)," 12 July 2013. [Online]. Available: <https://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/ior/>. [Accessed 26 February 2015].
- [28] R. Coker, "Coker," [Online]. Available: <http://www.coker.com.au/bonnie++/>. [Accessed 26 February 2015].
- [29] "IOZone," [Online]. Available: <http://www.iozone.org>. [Accessed 26 February 2015].
- [30] A. Danalis, G. Marin, C. McCurdy, J. Meredith, P. Roth, K. Spafford, V. Tipparaju and J. Vetter, "The Scalable Heterogeneous Computing (SHOC) Benchmark Suite," in *GPGPU General-Purpose Computation on Graphics Processing Units*, 2010.
- [31] "VMware," [Online]. Available: <http://www.vmware.com/products/vmmark/overview.html>. [Accessed 1 November 2013].
- [32] P. Luszczek, D. Bailey, J. Dongarra, J. Kepner, R. Lucas, R. Rabenseifner and D. Takahashi, "The HPC Challenge (HPCC) Benchmark Suite," in *Supercomputing Conference*, Tampa, 2006.
- [33] J. Dongarra, P. Luszczek and A. Petit, "The LINPACK Benchmark: Past, Present, and Future," July 2002.
- [34] A. Petit, C. Whaley, J. Dongarra, A. Cleary and P. Luszczek, "HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers," 10 September 2008. [Online]. Available: <http://www.netlib.org/benchmark/hpl/>. [Accessed 26 February 2015].
- [35] "GitHub," 9 July 2012. [Online]. Available: <https://github.com/vetter/shoc/wiki>. [Accessed 26 February 2015].
- [36] "HPC Challenge," [Online]. Available: <http://icl.cs.utk.edu/hpcc/>. [Accessed 26 February 2015].

- [37] D. Enright, A. Brethorst, J. Everist, J. Stoermer, B. Winkenbach and L. Wang, "Cloud Benchmarking & The Cloud Tester Project," The Aerospace Corporation, 2014.
- [38] A. Brethorst, D. Enright, J. Everist and B. Winkenbach, "Cloud Tester Benchmark Suite v1.0.2 Users Guide," The Aerospace Corporation, 2013.
- [39] W. Felter, A. Ferreira, R. Rajamony and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," Computer Science, Austin, 2014.
- [40] P. Luszczek, J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. McCalpin, D. Bailey and D. Takahashi, "Introduction to the HPC Challenge Benchmark Suite," 25 04 2005.
- [41] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Translation_lookaside_buffer. [Accessed 26 February 2015].
- [42] A. Brethorst, D. Enright, J. Everist and B. Winkenbach, "Cloud Tester Benchmark Suite (v1.0.2) Installation Guide," The Aerospace Corporation, 2013.
- [43] "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Protection_ring. [Accessed 26 February 2015].
- [44] S. Hykes, "Docker," Docker, Inc, 10 March 2014. [Online]. Available: <http://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/>. [Accessed 26 February 2015].
- [45] "Yong Land Computing," [Online]. Available: <http://www.iyoungland.com/en-us/ChallengesandSolutions/ByTechnologies/Cloud.aspx>. [Accessed 26 February 2015].
- [46] "Virt Zone," [Online]. Available: <http://www.virtzone.net/the-difference-between-a-type-2-hypervisor-and-a-type-1-hypervisor/>. [Accessed 26 February 2015].

APPENDIX A

HPCC OUTPUT

The following is CTBS generated HPCC output in each of the execution environment.

A.1 Bare-metal

#####

This is the DARPA/DOE HPC Challenge Benchmark version 1.4.2 October 2012

Produced by Jack Dongarra and Piotr Luszczek

Innovative Computing Laboratory

University of Tennessee Knoxville and Oak Ridge National Laboratory

See the source files for authors of specific codes.

Compiled on Feb 4 2015 at 16:41:21

Current time (1423090110) is Wed Feb 4 16:48:30 2015

Hostname: 'benchmarks.local'

#####

=====

HPLinpack 2.0 -- High-Performance Linpack benchmark -- September 10, 2008

Written by A. Petitet and R. Clint Whaley, Innovative Computing Laboratory, UTK

Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK

Modified by Julien Langou, University of Colorado Denver

=====

An explanation of the input/output parameters follows:

T/V : Wall time / encoded variant.

N : The order of the coefficient matrix A.

NB : The partitioning blocking factor.

P : The number of process rows.

Q : The number of process columns.

Time : Time in seconds to solve the linear system.

Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N : 4096

NB : 128

PMAP : Row-major process mapping

P : 2

Q : 4

PFACT : Right

NBMIN : 4

NDIV : 2

RFACT : Crout

BCAST : 1ringM

DEPTH : 1

SWAP : Mix (threshold = 64)

L1 : transposed form

U : transposed form

EQUIL : yes

ALIGN : 8 double precision words

- The matrix A is randomly generated for each test.

- The following scaled residual check will be computed:

$\|Ax-b\|_{\infty} / (\epsilon * (\|x\|_{\infty} * \|A\|_{\infty} + \|b\|_{\infty}) * N)$

- The relative machine precision (eps) is taken to be 1.110223e-16

- Computational tests pass if scaled residuals are less than 16.0

Begin of MPIRandomAccess section.
Running on 8 processors (PowerofTwo)
Total Main table size = $2^{24} = 16777216$ words
PE Main table size = $2^{21} = 2097152$ words/PE
Default number of updates (RECOMMENDED) = 67108864
Number of updates EXECUTED = 67108864 (for a TIME BOUND of 60.00 secs)
CPU time used = 0.923860 seconds
Real time used = 0.923018 seconds
0.072705912 Billion(10^9) Updates per second [GUP/s]
0.009088239 Billion(10^9) Updates/PE per second [GUP/s]
Verification: CPU time used = 0.291955 seconds
Verification: Real time used = 0.291986 seconds
Found 0 errors in 16777216 locations (passed).
Current time (1423090111) is Wed Feb 4 16:48:31 2015

End of MPIRandomAccess section.
Begin of StarRandomAccess section.
Main table size = $2^{21} = 2097152$ words
Number of updates = 8388608
CPU time used = 0.123981 seconds
Real time used = 0.124032 seconds
0.067632600 Billion(10^9) Updates per second [GUP/s]
Found 0 errors in 2097152 locations (passed).
Node(s) with error 0
Minimum GUP/s 0.067177
Average GUP/s 0.068186
Maximum GUP/s 0.068769
Current time (1423090111) is Wed Feb 4 16:48:31 2015

End of StarRandomAccess section.
Begin of SingleRandomAccess section.
Node(s) with error 0
Node selected 7
Single GUP/s 0.153069
Current time (1423090111) is Wed Feb 4 16:48:31 2015

End of SingleRandomAccess section.
Begin of MPIRandomAccess_LCG section.
Running on 8 processors (PowerofTwo)
Total Main table size = $2^{24} = 16777216$ words
PE Main table size = $2^{21} = 2097152$ words/PE
Default number of updates (RECOMMENDED) = 67108864
Number of updates EXECUTED = 67108864 (for a TIME BOUND of 60.00 secs)
CPU time used = 0.846871 seconds
Real time used = 0.846990 seconds
0.079232170 Billion(10^9) Updates per second [GUP/s]
0.009904021 Billion(10^9) Updates/PE per second [GUP/s]
Verification: CPU time used = 0.215967 seconds
Verification: Real time used = 0.216433 seconds
Found 0 errors in 16777216 locations (passed).
Current time (1423090112) is Wed Feb 4 16:48:32 2015

End of MPIRandomAccess_LCG section.
Begin of StarRandomAccess_LCG section.
Main table size = $2^{21} = 2097152$ words
Number of updates = 8388608
CPU time used = 0.122981 seconds
Real time used = 0.123470 seconds
0.067940418 Billion(10^9) Updates per second [GUP/s]
Found 0 errors in 2097152 locations (passed).
Node(s) with error 0
Minimum GUP/s 0.067555
Average GUP/s 0.068617
Maximum GUP/s 0.069307

Current time (1423090113) is Wed Feb 4 16:48:33 2015

End of StarRandomAccess_LCG section.

Begin of SingleRandomAccess_LCG section.

Node(s) with error 0

Node selected 1

Single GUP/s 0.156782

Current time (1423090113) is Wed Feb 4 16:48:33 2015

End of SingleRandomAccess_LCG section.

Begin of PTRANS section.

M: 2048

N: 2048

MB: 128

NB: 128

P: 2

Q: 4

TIME	M	N	MB	NB	P	Q	TIME	CHECK	GB/s	RESID
------	---	---	----	----	---	---	------	-------	------	-------

WALL	2048	2048	128	128	2	4	0.01	PASSED	3.811	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.811	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.811	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	4.195	0.00
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.811	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.811	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00

Finished 5 tests, with the following results:

5 tests completed and passed residual checks.

0 tests completed and failed residual checks.

0 tests skipped because of illegal input values.

END OF TESTS.

Current time (1423090113) is Wed Feb 4 16:48:33 2015

End of PTRANS section.

Begin of StarDGEMM section.

Scaled residual: 0.0181865

Node(s) with error 0

Minimum Gflop/s 1.488568

Average Gflop/s 1.513195

Maximum Gflop/s 1.524991

Current time (1423090114) is Wed Feb 4 16:48:34 2015

End of StarDGEMM section.

Begin of SingleDGEMM section.

Node(s) with error 0

Node selected 1

Single DGEMM Gflop/s 1.743331

Current time (1423090115) is Wed Feb 4 16:48:35 2015

End of SingleDGEMM section.

Begin of StarSTREAM section.

This system uses 8 bytes per DOUBLE PRECISION word.

Array size = 699050, Offset = 0

Total memory required = 0.0156 GiB.

Each test is run 10 times, but only

the *best* time for each is used.

Your clock granularity/precision appears to be 1 microseconds.

Each test below will take on the order of 1317 microseconds.

(= 1317 clock ticks)

Increase the size of the arrays if this shows that you are not getting at least 20 clock ticks per test.

 WARNING -- The above is only a rough guideline.
 For best results, please be sure you know the precision of your system timer.

Function	Rate (GB/s)	Avg time	Min time	Max time
Copy:	6.4097	0.0018	0.0017	0.0019
Scale:	6.3335	0.0018	0.0018	0.0019
Add:	6.8339	0.0025	0.0025	0.0025
Triad:	7.2255	0.0024	0.0023	0.0024

 Results Comparison:

Expected : 806214911132812544.000000 161242982226562496.000000 214990642968750016.000000

Observed : 806214911144195456.000000 161242982228413824.000000 214990642972468000.000000

Solution Validates

Node(s) with error 0

Minimum Copy GB/s 6.409680

Average Copy GB/s 6.548682

Maximum Copy GB/s 6.661808

Minimum Scale GB/s 6.255828

Average Scale GB/s 6.360442

Maximum Scale GB/s 6.443133

Minimum Add GB/s 6.819992

Average Add GB/s 6.893697

Maximum Add GB/s 6.946562

Minimum Triad GB/s 7.100058

Average Triad GB/s 7.174252

Maximum Triad GB/s 7.234366

Current time (1423090115) is Wed Feb 4 16:48:35 2015

End of StarSTREAM section.

Begin of SingleSTREAM section.

Node(s) with error 0

Node selected 3

Single STREAM Copy GB/s 13.331188

Single STREAM Scale GB/s 12.768767

Single STREAM Add GB/s 13.865749

Single STREAM Triad GB/s 13.292157

Current time (1423090115) is Wed Feb 4 16:48:35 2015

End of SingleSTREAM section.

Begin of MPIFFT section.

Number of nodes: 8

Vector size: 2097152

Generation time: 0.014

Tuning: 0.030

Computing: 0.031

Inverse FFT: 0.033

max(|x-x0|): 1.578e-15

Gflop/s: 7.002

Current time (1423090115) is Wed Feb 4 16:48:35 2015

End of MPIFFT section.

Begin of StarFFT section.

Vector size: 524288

Generation time: 0.027

Tuning: 0.000

Computing: 0.030

Inverse FFT: 0.032

max(|x-x0|): 1.712e-15

Node(s) with error 0

Minimum Gflop/s 1.620491

Average Gflop/s 1.657962
 Maximum Gflop/s 1.681833
 Current time (1423090115) is Wed Feb 4 16:48:35 2015

End of StarFFT section.
 Begin of SingleFFT section.
 Node(s) with error 0
 Node selected 3
 Single FFT Gflop/s 1.622870
 Current time (1423090115) is Wed Feb 4 16:48:35 2015

End of SingleFFT section.
 Begin of LatencyBandwidth section.

 Latency-Bandwidth-Benchmark R1.5.1 (c) HLRS, University of Stuttgart
 Written by Rolf Rabenseifner, Gerrit Schulz, and Michael Speck, Germany

Details - level 2

MPI_Wtime granularity.
 Max. MPI_Wtick is 0.000001 sec
 wtick is set to 0.000001 sec

Message Length: 8
 Latency min / avg / max: 0.001118 / 0.001118 / 0.001118 msecs
 Bandwidth min / avg / max: 7.158 / 7.158 / 7.158 MByte/s

MPI_Wtime granularity is ok.
 message size: 8
 max time : 10.000000 secs
 latency for msg: 0.001118 msecs
 estimation for ping pong: 0.100583 msecs
 max number of ping pong pairs = 99420
 max client pings = max server pongs = 315
 stride for latency = 1
 Message Length: 8
 Latency min / avg / max: 0.000358 / 0.000712 / 0.001060 msecs
 Bandwidth min / avg / max: 7.550 / 13.655 / 22.370 MByte/s

Message Length: 2000000
 Latency min / avg / max: 0.491023 / 0.491023 / 0.491023 msecs
 Bandwidth min / avg / max: 4073.128 / 4073.128 / 4073.128 MByte/s

MPI_Wtime granularity is ok.
 message size: 2000000
 max time : 30.000000 secs
 latency for msg: 0.491023 msecs
 estimation for ping pong: 3.928185 msecs
 max number of ping pong pairs = 7637
 max client pings = max server pongs = 87
 stride for latency = 1
 Message Length: 2000000
 Latency min / avg / max: 0.286937 / 0.408609 / 0.512004 msecs
 Bandwidth min / avg / max: 3906.220 / 5257.208 / 6970.177 MByte/s

Message Size: 8 Byte
 Natural Order Latency: 0.001216 msec
 Natural Order Bandwidth: 6.579300 MB/s
 Avg Random Order Latency: 0.001272 msec
 Avg Random Order Bandwidth: 6.290771 MB/s

Message Size: 2000000 Byte
 Natural Order Latency: 1.897514 msec
 Natural Order Bandwidth: 1054.010743 MB/s

Avg Random Order Latency: 1.469890 msec
Avg Random Order Bandwidth: 1360.646169 MB/s

Execution time (wall clock) = 0.811 sec on 8 processes
- for cross ping_pong latency = 0.008 sec
- for cross ping_pong bandwidth = 0.189 sec
- for ring latency = 0.011 sec
- for ring bandwidth = 0.603 sec

Latency-Bandwidth-Benchmark R1.5.1 (c) HLRS, University of Stuttgart
Written by Rolf Rabenseifner, Gerrit Schulz, and Michael Speck, Germany

Major Benchmark results:

Max Ping Pong Latency: 0.001060 msecs
Randomly Ordered Ring Latency: 0.001272 msecs
Min Ping Pong Bandwidth: 3906.220256 MB/s
Naturally Ordered Ring Bandwidth: 1054.010743 MB/s
Randomly Ordered Ring Bandwidth: 1360.646169 MB/s

Detailed benchmark results:

Ping Pong:
Latency min / avg / max: 0.000358 / 0.000712 / 0.001060 msecs
Bandwidth min / avg / max: 3906.220 / 5257.208 / 6970.177 MByte/s
Ring:
On naturally ordered ring: latency= 0.001216 msec, bandwidth= 1054.010743 MB/s
On randomly ordered ring: latency= 0.001272 msec, bandwidth= 1360.646169 MB/s

Benchmark conditions:

The latency measurements were done with 8 bytes
The bandwidth measurements were done with 2000000 bytes
The ring communication was done in both directions on 8 processes
The Ping Pong measurements were done on
- 56 pairs of processes for latency benchmarking, and
- 56 pairs of processes for bandwidth benchmarking,
out of $8*(8-1) = 56$ possible combinations on 8 processes.
(1 MB/s = 10^6 byte/sec)

Current time (1423090116) is Wed Feb 4 16:48:36 2015

End of LatencyBandwidth section.

Begin of HPL section.

=====

HLIPack 2.0 -- High-Performance Linpack benchmark -- September 10, 2008
Written by A. Petit and R. Clint Whaley, Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
=====

An explanation of the input/output parameters follows:

T/V : Wall time / encoded variant.
N : The order of the coefficient matrix A.
NB : The partitioning blocking factor.
P : The number of process rows.
Q : The number of process columns.
Time : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N : 4096
 NB : 128
 PMAP : Row-major process mapping
 P : 2
 Q : 4
 PFACT : Right
 NBMIN : 4
 NDIV : 2
 RFACT : Crout
 BCAST : 1ringM
 DEPTH : 1
 SWAP : Mix (threshold = 64)
 L1 : transposed form
 U : transposed form
 EQUIL : yes
 ALIGN : 8 double precision words

-
- The matrix A is randomly generated for each test.
 - The following scaled residual check will be computed:

$$\frac{\|Ax-b\|_{\infty}}{(\text{eps} * (\|x\|_{\infty} * \|A\|_{\infty} + \|b\|_{\infty}) * N)}$$
 - The relative machine precision (eps) is taken to be 1.110223e-16
 - Computational tests pass if scaled residuals are less than 16.0

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	4096	128	2	4	4.42	1.038e+01

$\|Ax-b\|_{\infty}/(\text{eps}*(\|A\|_{\infty}*\|x\|_{\infty}+\|b\|_{\infty})*N)=$ 0.0042988 PASSED

Finished 1 tests with the following results:
 1 tests completed and passed residual checks,
 0 tests completed and failed residual checks,
 0 tests skipped because of illegal input values.

End of Tests.

Current time (1423090121) is Wed Feb 4 16:48:41 2015

End of HPL section.
 Begin of Summary section.
 VersionMajor=1
 VersionMinor=4
 VersionMicro=2
 VersionRelease=f
 LANG=C
 Success=1
 sizeof_char=1
 sizeof_short=2
 sizeof_int=4
 sizeof_long=8
 sizeof_void_ptr=8
 sizeof_size_t=8
 sizeof_float=4
 sizeof_double=8
 sizeof_s64Int=8
 sizeof_u64Int=8
 sizeof_struct_double_double=16
 CommWorldProcs=8
 MPI_Wtick=1.000000e-06
 HPL_Tflops=0.0103777

```

HPL_time=4.417
HPL_eps=1.11022e-16
HPL_Rnorml=8.56681e-12
HPL_Anorml=1055
HPL_Anorml=1062.42
HPL_Xnorml=3501.5
HPL_Xnorml=4.12431
HPL_Bnorml=0.499778
HPL_N=4096
HPL_NB=128
HPL_nprow=2
HPL_npcol=4
HPL_depth=1
HPL_nbdiv=2
HPL_nbmin=4
HPL_cpfact=R
HPL_crfact=C
HPL_ctop=1
HPL_order=R
HPL_dMACH_EPS=1.110223e-16
HPL_dMACH_SFMIN=2.225074e-308
HPL_dMACH_BASE=2.000000e+00
HPL_dMACH_PREC=2.220446e-16
HPL_dMACH_MLEN=5.300000e+01
HPL_dMACH_RND=1.000000e+00
HPL_dMACH_EMIN=-1.021000e+03
HPL_dMACH_RMIN=2.225074e-308
HPL_dMACH_EMAX=1.024000e+03
HPL_dMACH_RMAX=1.797693e+308
HPL_sMACH_EPS=5.960464e-08
HPL_sMACH_SFMIN=1.175494e-38
HPL_sMACH_BASE=2.000000e+00
HPL_sMACH_PREC=1.192093e-07
HPL_sMACH_MLEN=2.400000e+01
HPL_sMACH_RND=1.000000e+00
HPL_sMACH_EMIN=-1.250000e+02
HPL_sMACH_RMIN=1.175494e-38
HPL_sMACH_EMAX=1.280000e+02
HPL_sMACH_RMAX=3.402823e+38
dweeps=1.110223e-16
sweeps=5.960464e-08
HPLMaxProcs=8
HPLMinProcs=8
DGEMM_N=835
StarDGEMM_Gflops=1.5132
SingleDGEMM_Gflops=1.74333
PTRANS_GB=3.81082
PTRANS_time=0.00851989
PTRANS_residual=0
PTRANS_n=2048
PTRANS_nb=128
PTRANS_nprow=2
PTRANS_npcol=4
MPIRandomAccess_LCG_N=16777216
MPIRandomAccess_LCG_time=0.84699
MPIRandomAccess_LCG_CheckTime=0.216433
MPIRandomAccess_LCG_Errors=0
MPIRandomAccess_LCG_ErrorsFraction=0
MPIRandomAccess_LCG_ExeUpdates=67108864
MPIRandomAccess_LCG_GUPs=0.0792322
MPIRandomAccess_LCG_TimeBound=60
MPIRandomAccess_LCG_Algorithm=0
MPIRandomAccess_N=16777216
MPIRandomAccess_time=0.923018
MPIRandomAccess_CheckTime=0.291986
MPIRandomAccess_Errors=0

```

```

MPIRandomAccess_ErrorsFraction=0
MPIRandomAccess_ExeUpdates=67108864
MPIRandomAccess_GUPs=0.0727059
MPIRandomAccess_TimeBound=60
MPIRandomAccess_Algorithm=0
RandomAccess_LCG_N=2097152
StarRandomAccess_LCG_GUPs=0.0686171
SingleRandomAccess_LCG_GUPs=0.156782
RandomAccess_N=2097152
StarRandomAccess_GUPs=0.0681863
SingleRandomAccess_GUPs=0.153069
STREAM_VectorSize=699050
STREAM_Threads=1
StarSTREAM_Copy=6.54868
StarSTREAM_Scale=6.36044
StarSTREAM_Add=6.8937
StarSTREAM_Triad=7.17425
SingleSTREAM_Copy=13.3312
SingleSTREAM_Scale=12.7688
SingleSTREAM_Add=13.8657
SingleSTREAM_Triad=13.2922
FFT_N=524288
StarFFT_Gflops=1.65796
SingleFFT_Gflops=1.62287
MPIFFT_N=2097152
MPIFFT_Gflops=7.00161
MPIFFT_maxErr=1.57792e-15
MPIFFT_Procs=8
MaxPingPongLatency_usec=1.05964
RandomlyOrderedRingLatency_usec=1.2717
MinPingPongBandwidth_GBytes=3.90622
NaturallyOrderedRingBandwidth_GBytes=1.05401
RandomlyOrderedRingBandwidth_GBytes=1.36065
MinPingPongLatency_usec=0.357628
AvgPingPongLatency_usec=0.711649
MaxPingPongBandwidth_GBytes=6.97018
AvgPingPongBandwidth_GBytes=5.25721
NaturallyOrderedRingLatency_usec=1.21593
FFTEblk=16
FFTEnp=8
FFTEl2size=1048576
M_OPENMP=-1
omp_get_num_threads=0
omp_get_max_threads=0
omp_get_num_procs=0
MemProc=-1
MemSpec=-1
MemVal=-1
MPIFFT_time0=9.53674e-07
MPIFFT_time1=0.00489402
MPIFFT_time2=0.00440311
MPIFFT_time3=0.00318193
MPIFFT_time4=0.012639
MPIFFT_time5=0.00498295
MPIFFT_time6=9.53674e-07
CPS_HPCC_FFT_235=0
CPS_HPCC_FFTW_ESTIMATE=0
CPS_HPCC_MEMALLCTR=0
CPS_HPL_USE_GETPROCESSTIMES=0
CPS_RA_SANDIA_NOPT=0
CPS_RA_SANDIA_OPT2=0
CPS_USING_FFTW=0
End of Summary section.
#####
End of HPC Challenge tests.
Current time (1423090121) is Wed Feb 4 16:48:41 2015

```

#####

A.2 Docker

```
#####
This is the DARPA/DOE HPC Challenge Benchmark version 1.4.2 October 2012
Produced by Jack Dongarra and Piotr Luszczek
Innovative Computing Laboratory
University of Tennessee Knoxville and Oak Ridge National Laboratory
```

See the source files for authors of specific codes.
 Compiled on Feb 8 2015 at 16:06:04
 Current time (1423413289) is Sun Feb 8 16:34:49 2015

```
Hostname: 'CTBS'
#####
```

```
=====
HPLinpack 2.0 -- High-Performance Linpack benchmark -- September 10, 2008
Written by A. Petitet and R. Clint Whaley, Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
=====
```

An explanation of the input/output parameters follows:
 T/V : Wall time / encoded variant.
 N : The order of the coefficient matrix A.
 NB : The partitioning blocking factor.
 P : The number of process rows.
 Q : The number of process columns.
 Time : Time in seconds to solve the linear system.
 Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

```
N : 4096
NB : 128
PMAP : Row-major process mapping
P : 2
Q : 4
PFACT : Right
NBMIN : 4
NDIV : 2
RFACT : Crout
BCAST : 1ringM
DEPTH : 1
SWAP : Mix (threshold = 64)
L1 : transposed form
U : transposed form
EQUIL : yes
ALIGN : 8 double precision words
```

-
- The matrix A is randomly generated for each test.
 - The following scaled residual check will be computed:

$$\|Ax-b\|_{\infty} / (\epsilon * (\|x\|_{\infty} * \|A\|_{\infty} + \|b\|_{\infty}) * N)$$
 - The relative machine precision (eps) is taken to be 1.110223e-16
 - Computational tests pass if scaled residuals are less than 16.0

Begin of MPIRandomAccess section.
 Running on 8 processors (PowerofTwo)
 Total Main table size = $2^{24} = 16777216$ words
 PE Main table size = $2^{21} = 2097152$ words/PE
 Default number of updates (RECOMMENDED) = 67108864
 Number of updates EXECUTED = 67108864 (for a TIME BOUND of 60.00 secs)
 CPU time used = 0.844872 seconds
 Real time used = 0.844939 seconds
 0.079424508 Billion(10^9) Updates per second [GUP/s]

0.009928064 Billion(10^9) Updates/PE per second [GUP/s]
Verification: CPU time used = 0.220966 seconds
Verification: Real time used = 0.221228 seconds
Found 0 errors in 16777216 locations (passed).
Current time (1423413290) is Sun Feb 8 16:34:50 2015

End of MPIRandomAccess section.
Begin of StarRandomAccess section.
Main table size = $2^{21} = 2097152$ words
Number of updates = 8388608
CPU time used = 0.123981 seconds
Real time used = 0.123616 seconds
0.067860223 Billion(10^9) Updates per second [GUP/s]
Found 0 errors in 2097152 locations (passed).
Node(s) with error 0
Minimum GUP/s 0.067491
Average GUP/s 0.068296
Maximum GUP/s 0.069087
Current time (1423413291) is Sun Feb 8 16:34:51 2015

End of StarRandomAccess section.
Begin of SingleRandomAccess section.
Node(s) with error 0
Node selected 1
Single GUP/s 0.149586
Current time (1423413291) is Sun Feb 8 16:34:51 2015

End of SingleRandomAccess section.
Begin of MPIRandomAccess_LCG section.
Running on 8 processors (PowerofTwo)
Total Main table size = $2^{24} = 16777216$ words
PE Main table size = $2^{21} = 2097152$ words/PE
Default number of updates (RECOMMENDED) = 67108864
Number of updates EXECUTED = 67108864 (for a TIME BOUND of 60.00 secs)
CPU time used = 0.850870 seconds
Real time used = 0.850176 seconds
0.078935252 Billion(10^9) Updates per second [GUP/s]
0.009866906 Billion(10^9) Updates/PE per second [GUP/s]
Verification: CPU time used = 0.216967 seconds
Verification: Real time used = 0.217598 seconds
Found 0 errors in 16777216 locations (passed).
Current time (1423413292) is Sun Feb 8 16:34:52 2015

End of MPIRandomAccess_LCG section.
Begin of StarRandomAccess_LCG section.
Main table size = $2^{21} = 2097152$ words
Number of updates = 8388608
CPU time used = 0.122981 seconds
Real time used = 0.123171 seconds
0.068105332 Billion(10^9) Updates per second [GUP/s]
Found 0 errors in 2097152 locations (passed).
Node(s) with error 0
Minimum GUP/s 0.067817
Average GUP/s 0.068603
Maximum GUP/s 0.069324
Current time (1423413292) is Sun Feb 8 16:34:52 2015

End of StarRandomAccess_LCG section.
Begin of SingleRandomAccess_LCG section.
Node(s) with error 0
Node selected 4
Single GUP/s 0.154923
Current time (1423413292) is Sun Feb 8 16:34:52 2015

End of SingleRandomAccess_LCG section.
Begin of PTRANS section.

M: 2048

N: 2048

MB: 128

NB: 128

P: 2

Q: 4

TIME	M	N	MB	NB	P	Q	TIME	CHECK	GB/s	RESID
------	---	---	----	----	---	---	------	-------	------	-------

WALL	2048	2048	128	128	2	4	0.01	PASSED	3.778	0.00
------	------	------	-----	-----	---	---	------	--------	-------	------

CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00
-----	------	------	-----	-----	---	---	------	--------	-------	------

WALL	2048	2048	128	128	2	4	0.01	PASSED	3.778	0.00
------	------	------	-----	-----	---	---	------	--------	-------	------

CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00
-----	------	------	-----	-----	---	---	------	--------	-------	------

WALL	2048	2048	128	128	2	4	0.01	PASSED	3.778	0.00
------	------	------	-----	-----	---	---	------	--------	-------	------

CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00
-----	------	------	-----	-----	---	---	------	--------	-------	------

WALL	2048	2048	128	128	2	4	0.01	PASSED	3.778	0.00
------	------	------	-----	-----	---	---	------	--------	-------	------

CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00
-----	------	------	-----	-----	---	---	------	--------	-------	------

WALL	2048	2048	128	128	2	4	0.01	PASSED	3.778	0.00
------	------	------	-----	-----	---	---	------	--------	-------	------

CPU	2048	2048	128	128	2	4	0.01	PASSED	3.729	0.00
-----	------	------	-----	-----	---	---	------	--------	-------	------

Finished 5 tests, with the following results:

5 tests completed and passed residual checks.

0 tests completed and failed residual checks.

0 tests skipped because of illegal input values.

END OF TESTS.

Current time (1423413292) is Sun Feb 8 16:34:52 2015

End of PTRANS section.

Begin of StarDGEMM section.

Scaled residual: 0.0231783

Node(s) with error 0

Minimum Gflop/s 1.498816

Average Gflop/s 1.513326

Maximum Gflop/s 1.520135

Current time (1423413293) is Sun Feb 8 16:34:53 2015

End of StarDGEMM section.

Begin of SingleDGEMM section.

Node(s) with error 0

Node selected 2

Single DGEMM Gflop/s 1.742304

Current time (1423413294) is Sun Feb 8 16:34:54 2015

End of SingleDGEMM section.

Begin of StarSTREAM section.

This system uses 8 bytes per DOUBLE PRECISION word.

Array size = 699050, Offset = 0

Total memory required = 0.0156 GiB.

Each test is run 10 times, but only

the *best* time for each is used.

Your clock granularity/precision appears to be 1 microseconds.

Each test below will take on the order of 1236 microseconds.

(= 1236 clock ticks)

Increase the size of the arrays if this shows that

you are not getting at least 20 clock ticks per test.

WARNING -- The above is only a rough guideline.

For best results, please be sure you know the

precision of your system timer.

Function	Rate (GB/s)	Avg time	Min time	Max time
----------	-------------	----------	----------	----------

Copy:	6.3992	0.0018	0.0017	0.0018
-------	--------	--------	--------	--------

Scale:	6.1996	0.0018	0.0018	0.0019
--------	--------	--------	--------	--------

Add:	6.8847	0.0025	0.0024	0.0025
Triad:	7.1303	0.0024	0.0024	0.0024

Results Comparison:

Expected : 806214911132812544.000000 161242982226562496.000000 214990642968750016.000000

Observed : 806214911144195456.000000 161242982228413824.000000 214990642972468000.000000

Solution Validates

Node(s) with error 0

Minimum Copy GB/s 6.399189

Average Copy GB/s 6.548047

Maximum Copy GB/s 6.693173

Minimum Scale GB/s 6.199610

Average Scale GB/s 6.324691

Maximum Scale GB/s 6.428124

Minimum Add GB/s 6.809433

Average Add GB/s 6.889603

Maximum Add GB/s 6.930143

Minimum Triad GB/s 7.105794

Average Triad GB/s 7.147473

Maximum Triad GB/s 7.225452

Current time (1423413294) is Sun Feb 8 16:34:54 2015

End of StarSTREAM section.

Begin of SingleSTREAM section.

Node(s) with error 0

Node selected 1

Single STREAM Copy GB/s 13.316052

Single STREAM Scale GB/s 13.020386

Single STREAM Add GB/s 14.051253

Single STREAM Triad GB/s 13.401005

Current time (1423413294) is Sun Feb 8 16:34:54 2015

End of SingleSTREAM section.

Begin of MPIFFT section.

Number of nodes: 8

Vector size: 2097152

Generation time: 0.014

Tuning: 0.020

Computing: 0.030

Inverse FFT: 0.032

max(|x-x0|): 1.578e-15

Gflop/s: 7.350

Current time (1423413294) is Sun Feb 8 16:34:54 2015

End of MPIFFT section.

Begin of StarFFT section.

Vector size: 524288

Generation time: 0.027

Tuning: 0.000

Computing: 0.030

Inverse FFT: 0.032

max(|x-x0|): 1.712e-15

Node(s) with error 0

Minimum Gflop/s 1.644084

Average Gflop/s 1.669933

Maximum Gflop/s 1.691132

Current time (1423413294) is Sun Feb 8 16:34:54 2015

End of StarFFT section.

Begin of SingleFFT section.

Node(s) with error 0

Node selected 1

Single FFT Gflop/s 1.665634

Current time (1423413295) is Sun Feb 8 16:34:55 2015

End of SingleFFT section.
 Begin of LatencyBandwidth section.

 Latency-Bandwidth-Benchmark R1.5.1 (c) HLRS, University of Stuttgart
 Written by Rolf Rabenseifner, Gerrit Schulz, and Michael Speck, Germany

Details - level 2

MPI_Wtime granularity.
 Max. MPI_Wtick is 0.000001 sec
 wtick is set to 0.000001 sec

Message Length: 8
 Latency min / avg / max: 0.000939 / 0.000939 / 0.000939 msec
 Bandwidth min / avg / max: 8.522 / 8.522 / 8.522 MByte/s

MPI_Wtime granularity is ok.
 message size: 8
 max time : 10.000000 secs
 latency for msg: 0.000939 msec
 estimation for ping pong: 0.084490 msec
 max number of ping pong pairs = 118357
 max client pings = max server pongs = 344
 stride for latency = 1
 Message Length: 8
 Latency min / avg / max: 0.000358 / 0.000716 / 0.001000 msec
 Bandwidth min / avg / max: 8.000 / 13.636 / 22.370 MByte/s

Message Length: 2000000
 Latency min / avg / max: 0.499487 / 0.499487 / 0.499487 msec
 Bandwidth min / avg / max: 4004.109 / 4004.109 / 4004.109 MByte/s

MPI_Wtime granularity is ok.
 message size: 2000000
 max time : 30.000000 secs
 latency for msg: 0.499487 msec
 estimation for ping pong: 3.995895 msec
 max number of ping pong pairs = 7507
 max client pings = max server pongs = 86
 stride for latency = 1
 Message Length: 2000000
 Latency min / avg / max: 0.287890 / 0.407438 / 0.507951 msec
 Bandwidth min / avg / max: 3937.389 / 5258.285 / 6947.087 MByte/s

Message Size: 8 Byte
 Natural Order Latency: 0.001216 msec
 Natural Order Bandwidth: 6.579300 MB/s
 Avg Random Order Latency: 0.001184 msec
 Avg Random Order Bandwidth: 6.759495 MB/s

Message Size: 2000000 Byte
 Natural Order Latency: 1.120985 msec
 Natural Order Bandwidth: 1784.145903 MB/s
 Avg Random Order Latency: 1.207647 msec
 Avg Random Order Bandwidth: 1656.113554 MB/s

Execution time (wall clock) = 0.692 sec on 8 processes
 - for cross ping_pong latency = 0.008 sec
 - for cross ping_pong bandwidth = 0.189 sec
 - for ring latency = 0.011 sec
 - for ring bandwidth = 0.484 sec

 Latency-Bandwidth-Benchmark R1.5.1 (c) HLRS, University of Stuttgart

Written by Rolf Rabenseifner, Gerrit Schulz, and Michael Speck, Germany

Major Benchmark results:

Max Ping Pong Latency: 0.001000 msecs
Randomly Ordered Ring Latency: 0.001184 msecs
Min Ping Pong Bandwidth: 3937.389345 MB/s
Naturally Ordered Ring Bandwidth: 1784.145903 MB/s
Randomly Ordered Ring Bandwidth: 1656.113554 MB/s

Detailed benchmark results:

Ping Pong:
Latency min / avg / max: 0.000358 / 0.000716 / 0.001000 msecs
Bandwidth min / avg / max: 3937.389 / 5258.285 / 6947.087 MByte/s
Ring:
On naturally ordered ring: latency= 0.001216 msec, bandwidth= 1784.145903 MB/s
On randomly ordered ring: latency= 0.001184 msec, bandwidth= 1656.113554 MB/s

Benchmark conditions:

The latency measurements were done with 8 bytes
The bandwidth measurements were done with 2000000 bytes
The ring communication was done in both directions on 8 processes
The Ping Pong measurements were done on
- 56 pairs of processes for latency benchmarking, and
- 56 pairs of processes for bandwidth benchmarking,
out of $8 \cdot (8-1) = 56$ possible combinations on 8 processes.
(1 MB/s = 10^6 byte/sec)

Current time (1423413295) is Sun Feb 8 16:34:55 2015

End of LatencyBandwidth section.

Begin of HPL section.

=====

HPLinpack 2.0 -- High-Performance Linpack benchmark -- September 10, 2008
Written by A. Petitet and R. Clint Whaley, Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
=====

An explanation of the input/output parameters follows:

T/V : Wall time / encoded variant.
N : The order of the coefficient matrix A.
NB : The partitioning blocking factor.
P : The number of process rows.
Q : The number of process columns.
Time : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N : 4096
NB : 128
PMAP : Row-major process mapping
P : 2
Q : 4
PFACT : Right
NBMIN : 4
NDIV : 2
RFACT : Crout
BCAST : lringM

DEPTH : 1
 SWAP : Mix (threshold = 64)
 L1 : transposed form
 U : transposed form
 EQUIL : yes
 ALIGN : 8 double precision words

-
- The matrix A is randomly generated for each test.
 - The following scaled residual check will be computed:

$$\|Ax-b\|_{\infty} / (\text{eps} * (\|x\|_{\infty} * \|A\|_{\infty} + \|b\|_{\infty}) * N)$$
 - The relative machine precision (eps) is taken to be 1.110223e-16
 - Computational tests pass if scaled residuals are less than 16.0

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	4096	128	2	4	4.41	1.038e+01

$\|Ax-b\|_{\infty}/(\text{eps}*(\|A\|_{\infty}\|x\|_{\infty}+\|b\|_{\infty})N)=$ 0.0042988 PASSED

Finished 1 tests with the following results:
 1 tests completed and passed residual checks,
 0 tests completed and failed residual checks,
 0 tests skipped because of illegal input values.

End of Tests.

Current time (1423413300) is Sun Feb 8 16:35:00 2015

End of HPL section.
 Begin of Summary section.
 VersionMajor=1
 VersionMinor=4
 VersionMicro=2
 VersionRelease=f
 LANG=C
 Success=1
 sizeof_char=1
 sizeof_short=2
 sizeof_int=4
 sizeof_long=8
 sizeof_void_ptr=8
 sizeof_size_t=8
 sizeof_float=4
 sizeof_double=8
 sizeof_s64Int=8
 sizeof_u64Int=8
 sizeof_struct_double_double=16
 CommWorldProcs=8
 MPI_Wtick=1.000000e-06
 HPL_Tflops=0.0103845
 HPL_time=4.41409
 HPL_eps=1.11022e-16
 HPL_RnormI=8.56681e-12
 HPL_AnormI=1055
 HPL_AnormI=1062.42
 HPL_XnormI=3501.5
 HPL_XnormI=4.12431
 HPL_BnormI=0.499778
 HPL_N=4096
 HPL_NB=128
 HPL_nprow=2

```

HPL_npcol=4
HPL_depth=1
HPL_nbdiv=2
HPL_nbmin=4
HPL_cpfact=R
HPL_crfact=C
HPL_ctop=1
HPL_order=R
HPL_dMACH_EPS=1.110223e-16
HPL_dMACH_SFMIN=2.225074e-308
HPL_dMACH_BASE=2.000000e+00
HPL_dMACH_PREC=2.220446e-16
HPL_dMACH_MLEN=5.300000e+01
HPL_dMACH_RND=1.000000e+00
HPL_dMACH_EMIN=-1.021000e+03
HPL_dMACH_RMIN=2.225074e-308
HPL_dMACH_EMAX=1.024000e+03
HPL_dMACH_RMAX=1.797693e+308
HPL_sMACH_EPS=5.960464e-08
HPL_sMACH_SFMIN=1.175494e-38
HPL_sMACH_BASE=2.000000e+00
HPL_sMACH_PREC=1.192093e-07
HPL_sMACH_MLEN=2.400000e+01
HPL_sMACH_RND=1.000000e+00
HPL_sMACH_EMIN=-1.250000e+02
HPL_sMACH_RMIN=1.175494e-38
HPL_sMACH_EMAX=1.280000e+02
HPL_sMACH_RMAX=3.402823e+38
dweps=1.110223e-16
sweps=5.960464e-08
HPLMaxProcs=8
HPLMinProcs=8
DGEMM_N=835
StarDGEMM_Gflops=1.51333
SingleDGEMM_Gflops=1.7423
PTRANS_GB=3.77819
PTRANS_time=0.00862098
PTRANS_residual=0
PTRANS_n=2048
PTRANS_nb=128
PTRANS_nprow=2
PTRANS_npcol=4
MPIRandomAccess_LCG_N=16777216
MPIRandomAccess_LCG_time=0.850176
MPIRandomAccess_LCG_CheckTime=0.217598
MPIRandomAccess_LCG_Errors=0
MPIRandomAccess_LCG_ErrorsFraction=0
MPIRandomAccess_LCG_ExeUpdates=67108864
MPIRandomAccess_LCG_GUPs=0.0789353
MPIRandomAccess_LCG_TimeBound=60
MPIRandomAccess_LCG_Algorithm=0
MPIRandomAccess_N=16777216
MPIRandomAccess_time=0.844939
MPIRandomAccess_CheckTime=0.221228
MPIRandomAccess_Errors=0
MPIRandomAccess_ErrorsFraction=0
MPIRandomAccess_ExeUpdates=67108864
MPIRandomAccess_GUPs=0.0794245
MPIRandomAccess_TimeBound=60
MPIRandomAccess_Algorithm=0
RandomAccess_LCG_N=2097152
StarRandomAccess_LCG_GUPs=0.0686029
SingleRandomAccess_LCG_GUPs=0.154923
RandomAccess_N=2097152
StarRandomAccess_GUPs=0.0682962
SingleRandomAccess_GUPs=0.149586

```

```
STREAM_VectorSize=699050
STREAM_Threads=1
StarSTREAM_Copy=6.54805
StarSTREAM_Scale=6.32469
StarSTREAM_Add=6.8896
StarSTREAM_Triad=7.14747
SingleSTREAM_Copy=13.3161
SingleSTREAM_Scale=13.0204
SingleSTREAM_Add=14.0513
SingleSTREAM_Triad=13.401
FFT_N=524288
StarFFT_Gflops=1.66993
SingleFFT_Gflops=1.66563
MPIFFT_N=2097152
MPIFFT_Gflops=7.34985
MPIFFT_maxErr=1.57792e-15
MPIFFT_Procs=8
MaxPingPongLatency_usec=1.00003
RandomlyOrderedRingLatency_usec=1.18352
MinPingPongBandwidth_GBytes=3.93739
NaturallyOrderedRingBandwidth_GBytes=1.78415
RandomlyOrderedRingBandwidth_GBytes=1.65611
MinPingPongLatency_usec=0.357628
AvgPingPongLatency_usec=0.715847
MaxPingPongBandwidth_GBytes=6.94709
AvgPingPongBandwidth_GBytes=5.25829
NaturallyOrderedRingLatency_usec=1.21593
FFTEblk=16
FFTEnp=8
FFTEl2size=1048576
M_OPENMP=-1
omp_get_num_threads=0
omp_get_max_threads=0
omp_get_num_procs=0
MemProc=-1
MemSpec=-1
MemVal=-1
MPIFFT_time0=9.53674e-07
MPIFFT_time1=0.00443792
MPIFFT_time2=0.0044651
MPIFFT_time3=0.00267196
MPIFFT_time4=0.0131061
MPIFFT_time5=0.00392485
MPIFFT_time6=0
CPS_HPCC_FFT_235=0
CPS_HPCC_FFTW_ESTIMATE=0
CPS_HPCC_MEMALLCTR=0
CPS_HPL_USE_GETPROCESSTIMES=0
CPS_RA_SANDIA_NOPT=0
CPS_RA_SANDIA_OPT2=0
CPS_USING_FFTW=0
End of Summary section.
#####
End of HPC Challenge tests.
Current time (1423413300) is Sun Feb  8 16:35:00 2015
#####
```

A.3 KVM

```
#####
This is the DARPA/DOE HPC Challenge Benchmark version 1.4.2 October 2012
Produced by Jack Dongarra and Piotr Luszczek
Innovative Computing Laboratory
University of Tennessee Knoxville and Oak Ridge National Laboratory
```

```
See the source files for authors of specific codes.
Compiled on Feb 12 2015 at 06:32:12
Current time (1423746344) is Thu Feb 12 07:05:44 2015
```

```
Hostname: 'KVM.localdomain'
#####
```

```
=====
HPLinpack 2.0 -- High-Performance Linpack benchmark -- September 10, 2008
Written by A. Petitet and R. Clint Whaley, Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
=====
```

An explanation of the input/output parameters follows:

```
T/V : Wall time / encoded variant.
N : The order of the coefficient matrix A.
NB : The partitioning blocking factor.
P : The number of process rows.
Q : The number of process columns.
Time : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.
```

The following parameter values will be used:

```
N : 4096
NB : 128
PMAP : Row-major process mapping
P : 2
Q : 4
PFACT : Right
NBMIN : 4
NDIV : 2
RFACT : Crout
BCAST : 1ringM
DEPTH : 1
SWAP : Mix (threshold = 64)
L1 : transposed form
U : transposed form
EQUIL : yes
ALIGN : 8 double precision words
```

-
- The matrix A is randomly generated for each test.
 - The following scaled residual check will be computed:

$$\|Ax-b\|_{\infty} / (\epsilon * (\|x\|_{\infty} * \|A\|_{\infty} + \|b\|_{\infty}) * N)$$
 - The relative machine precision (eps) is taken to be 1.110223e-16
 - Computational tests pass if scaled residuals are less than 16.0

```
Begin of MPIRandomAccess section.
Running on 8 processors (PowerofTwo)
Total Main table size = 2^24 = 16777216 words
PE Main table size = 2^21 = 2097152 words/PE
Default number of updates (RECOMMENDED) = 67108864
Number of updates EXECUTED = 67108864 (for a TIME BOUND of 60.00 secs)
CPU time used = 0.930858 seconds
Real time used = 0.933970 seconds
0.071853342 Billion(10^9) Updates per second [GUP/s]
```


0.008981668 Billion(10^9) Updates/PE per second [GUP/s]
Verification: CPU time used = 0.267959 seconds
Verification: Real time used = 0.267738 seconds
Found 0 errors in 16777216 locations (passed).
Current time (1423746345) is Thu Feb 12 07:05:45 2015

End of MPIRandomAccess section.
Begin of StarRandomAccess section.
Main table size = $2^{21} = 2097152$ words
Number of updates = 8388608
CPU time used = 0.147978 seconds
Real time used = 0.147988 seconds
0.056684440 Billion(10^9) Updates per second [GUP/s]
Found 0 errors in 2097152 locations (passed).
Node(s) with error 0
Minimum GUP/s 0.050334
Average GUP/s 0.056960
Maximum GUP/s 0.062300
Current time (1423746346) is Thu Feb 12 07:05:46 2015

End of StarRandomAccess section.
Begin of SingleRandomAccess section.
Node(s) with error 0
Node selected 2
Single GUP/s 0.122743
Current time (1423746346) is Thu Feb 12 07:05:46 2015

End of SingleRandomAccess section.
Begin of MPIRandomAccess_LCG section.
Running on 8 processors (PowerofTwo)
Total Main table size = $2^{24} = 16777216$ words
PE Main table size = $2^{21} = 2097152$ words/PE
Default number of updates (RECOMMENDED) = 67108864
Number of updates EXECUTED = 67108864 (for a TIME BOUND of 60.00 secs)
CPU time used = 0.942857 seconds
Real time used = 0.943140 seconds
0.071154719 Billion(10^9) Updates per second [GUP/s]
0.008894340 Billion(10^9) Updates/PE per second [GUP/s]
Verification: CPU time used = 0.266959 seconds
Verification: Real time used = 0.267311 seconds
Found 0 errors in 16777216 locations (passed).
Current time (1423746347) is Thu Feb 12 07:05:47 2015

End of MPIRandomAccess_LCG section.
Begin of StarRandomAccess_LCG section.
Main table size = $2^{21} = 2097152$ words
Number of updates = 8388608
CPU time used = 0.142978 seconds
Real time used = 0.143332 seconds
0.058525715 Billion(10^9) Updates per second [GUP/s]
Found 0 errors in 2097152 locations (passed).
Node(s) with error 0
Minimum GUP/s 0.051127
Average GUP/s 0.058636
Maximum GUP/s 0.064864
Current time (1423746347) is Thu Feb 12 07:05:47 2015

End of StarRandomAccess_LCG section.
Begin of SingleRandomAccess_LCG section.
Node(s) with error 0
Node selected 5
Single GUP/s 0.109031
Current time (1423746348) is Thu Feb 12 07:05:48 2015

End of SingleRandomAccess_LCG section.
Begin of PTRANS section.

M: 2048
 N: 2048
 MB: 128
 NB: 128
 P: 2
 Q: 4

TIME	M	N	MB	NB	P	Q	TIME	CHECK	GB/s	RESID
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.317	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.356	0.00
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.317	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.356	0.00
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.317	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.356	0.00
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.317	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.356	0.00
WALL	2048	2048	128	128	2	4	0.01	PASSED	3.317	0.00
CPU	2048	2048	128	128	2	4	0.01	PASSED	3.356	0.00

Finished 5 tests, with the following results:
 5 tests completed and passed residual checks.
 0 tests completed and failed residual checks.
 0 tests skipped because of illegal input values.

END OF TESTS.

Current time (1423746348) is Thu Feb 12 07:05:48 2015

End of PTRANS section.

Begin of StarDGEMM section.

Scaled residual: 0.0237793

Node(s) with error 0

Minimum Gflop/s 1.309540

Average Gflop/s 1.393274

Maximum Gflop/s 1.485902

Current time (1423746349) is Thu Feb 12 07:05:49 2015

End of StarDGEMM section.

Begin of SingleDGEMM section.

Node(s) with error 0

Node selected 2

Single DGEMM Gflop/s 1.711240

Current time (1423746350) is Thu Feb 12 07:05:50 2015

End of SingleDGEMM section.

Begin of StarSTREAM section.

 This system uses 8 bytes per DOUBLE PRECISION word.

 Array size = 699050, Offset = 0

Total memory required = 0.0156 GiB.

Each test is run 10 times, but only
 the *best* time for each is used.

 Your clock granularity/precision appears to be 1 microseconds.

Each test below will take on the order of 1570 microseconds.

(= 1570 clock ticks)

Increase the size of the arrays if this shows that
 you are not getting at least 20 clock ticks per test.

 WARNING -- The above is only a rough guideline.

For best results, please be sure you know the
 precision of your system timer.

Function	Rate (GB/s)	Avg time	Min time	Max time
Copy:	5.3879	0.0021	0.0021	0.0022
Scale:	5.3468	0.0022	0.0021	0.0023

Add:	5.8538	0.0029	0.0029	0.0029
Triad:	7.3263	0.0028	0.0023	0.0029

Results Comparison:

Expected : 806214911132812544.000000 161242982226562496.000000 214990642968750016.000000
Observed : 806214911144195456.000000 161242982228413824.000000 214990642972468000.000000

Solution Validates

Node(s) with error 0

Minimum Copy GB/s 5.387901

Average Copy GB/s 6.006561

Maximum Copy GB/s 6.819661

Minimum Scale GB/s 5.346758

Average Scale GB/s 5.902708

Maximum Scale GB/s 7.211753

Minimum Add GB/s 5.775025

Average Add GB/s 6.463922

Maximum Add GB/s 7.814401

Minimum Triad GB/s 6.273955

Average Triad GB/s 7.239377

Maximum Triad GB/s 9.933467

Current time (1423746350) is Thu Feb 12 07:05:50 2015

End of StarSTREAM section.

Begin of SingleSTREAM section.

Node(s) with error 0

Node selected 5

Single STREAM Copy GB/s 11.577604

Single STREAM Scale GB/s 11.873564

Single STREAM Add GB/s 12.606356

Single STREAM Triad GB/s 11.949173

Current time (1423746350) is Thu Feb 12 07:05:50 2015

End of SingleSTREAM section.

Begin of MPIFFT section.

Number of nodes: 8

Vector size: 2097152

Generation time: 0.014

Tuning: 0.020

Computing: 0.033

Inverse FFT: 0.035

max(|x-x0|): 1.578e-15

Gflop/s: 6.652

Current time (1423746350) is Thu Feb 12 07:05:50 2015

End of MPIFFT section.

Begin of StarFFT section.

Vector size: 524288

Generation time: 0.028

Tuning: 0.000

Computing: 0.032

Inverse FFT: 0.034

max(|x-x0|): 1.712e-15

Node(s) with error 0

Minimum Gflop/s 1.484929

Average Gflop/s 1.537678

Maximum Gflop/s 1.598433

Current time (1423746350) is Thu Feb 12 07:05:50 2015

End of StarFFT section.

Begin of SingleFFT section.

Node(s) with error 0

Node selected 5

Single FFT Gflop/s 1.502216

Current time (1423746350) is Thu Feb 12 07:05:50 2015

End of SingleFFT section.
Begin of LatencyBandwidth section.

Latency-Bandwidth-Benchmark R1.5.1 (c) HLRS, University of Stuttgart
Written by Rolf Rabenseifner, Gerrit Schulz, and Michael Speck, Germany

Details - level 2

MPI_Wtime granularity.
Max. MPI_Wtick is 0.000001 sec
wtick is set to 0.000001 sec

Message Length: 8
Latency min / avg / max: 0.000417 / 0.000417 / 0.000417 msecs
Bandwidth min / avg / max: 19.174 / 19.174 / 19.174 MByte/s

Use MPI_Wtick for estimation of max pairs
message size: 8
max time : 10.000000 secs
latency for msg: 0.000417 msecs
estimation for ping pong: 0.037551 msecs
max number of ping pong pairs = 200000
max client pings = max server pongs = 447
stride for latency = 1
Message Length: 8
Latency min / avg / max: 0.000384 / 0.000722 / 0.001000 msecs
Bandwidth min / avg / max: 8.000 / 13.304 / 20.827 MByte/s

Message Length: 2000000
Latency min / avg / max: 0.304461 / 0.304461 / 0.304461 msecs
Bandwidth min / avg / max: 6568.996 / 6568.996 / 6568.996 MByte/s

MPI_Wtime granularity is ok.
message size: 2000000
max time : 30.000000 secs
latency for msg: 0.304461 msecs
estimation for ping pong: 2.435684 msecs
max number of ping pong pairs = 12316
max client pings = max server pongs = 110
stride for latency = 1
Message Length: 2000000
Latency min / avg / max: 0.288010 / 0.419974 / 0.539541 msecs
Bandwidth min / avg / max: 3706.853 / 5130.356 / 6944.212 MByte/s

Message Size: 8 Byte
Natural Order Latency: 0.001311 msec
Natural Order Bandwidth: 6.100806 MB/s
Avg Random Order Latency: 0.001300 msec
Avg Random Order Bandwidth: 6.155193 MB/s

Message Size: 2000000 Byte
Natural Order Latency: 1.383245 msec
Natural Order Bandwidth: 1445.875469 MB/s
Avg Random Order Latency: 1.442972 msec
Avg Random Order Bandwidth: 1386.027970 MB/s

Execution time (wall clock) = 0.792 sec on 8 processes
- for cross ping_pong latency = 0.008 sec
- for cross ping_pong bandwidth = 0.195 sec
- for ring latency = 0.013 sec
- for ring bandwidth = 0.576 sec

Latency-Bandwidth-Benchmark R1.5.1 (c) HLRS, University of Stuttgart

Written by Rolf Rabenseifner, Gerrit Schulz, and Michael Speck, Germany

Major Benchmark results:

Max Ping Pong Latency: 0.001000 msecs
Randomly Ordered Ring Latency: 0.001300 msecs
Min Ping Pong Bandwidth: 3706.852850 MB/s
Naturally Ordered Ring Bandwidth: 1445.875469 MB/s
Randomly Ordered Ring Bandwidth: 1386.027970 MB/s

Detailed benchmark results:

Ping Pong:
Latency min / avg / max: 0.000384 / 0.000722 / 0.001000 msecs
Bandwidth min / avg / max: 3706.853 / 5130.356 / 6944.212 MByte/s
Ring:
On naturally ordered ring: latency= 0.001311 msec, bandwidth= 1445.875469 MB/s
On randomly ordered ring: latency= 0.001300 msec, bandwidth= 1386.027970 MB/s

Benchmark conditions:

The latency measurements were done with 8 bytes
The bandwidth measurements were done with 2000000 bytes
The ring communication was done in both directions on 8 processes
The Ping Pong measurements were done on
- 56 pairs of processes for latency benchmarking, and
- 56 pairs of processes for bandwidth benchmarking,
out of $8 \cdot (8-1) = 56$ possible combinations on 8 processes.
(1 MB/s = 10^6 byte/sec)

Current time (1423746351) is Thu Feb 12 07:05:51 2015

End of LatencyBandwidth section.

Begin of HPL section.

=====

HPLinpack 2.0 -- High-Performance Linpack benchmark -- September 10, 2008
Written by A. Petitet and R. Clint Whaley, Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
=====

An explanation of the input/output parameters follows:

T/V : Wall time / encoded variant.
N : The order of the coefficient matrix A.
NB : The partitioning blocking factor.
P : The number of process rows.
Q : The number of process columns.
Time : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N : 4096
NB : 128
PMAP : Row-major process mapping
P : 2
Q : 4
PFACT : Right
NBMIN : 4
NDIV : 2
RFACT : Crout
BCAST : lringM

DEPTH : 1
 SWAP : Mix (threshold = 64)
 L1 : transposed form
 U : transposed form
 EQUIL : yes
 ALIGN : 8 double precision words

-
- The matrix A is randomly generated for each test.
 - The following scaled residual check will be computed:

$$\|Ax-b\|_{\infty} / (\text{eps} * (\|x\|_{\infty} * \|A\|_{\infty} + \|b\|_{\infty}) * N)$$
 - The relative machine precision (eps) is taken to be 1.110223e-16
 - Computational tests pass if scaled residuals are less than 16.0

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	4096	128	2	4	4.52	1.014e+01

$\|Ax-b\|_{\infty}/(\text{eps}*(\|A\|_{\infty}\|x\|_{\infty}+\|b\|_{\infty})N)=$ 0.0042988 PASSED

Finished 1 tests with the following results:
 1 tests completed and passed residual checks,
 0 tests completed and failed residual checks,
 0 tests skipped because of illegal input values.

End of Tests.

Current time (1423746356) is Thu Feb 12 07:05:56 2015

End of HPL section.
 Begin of Summary section.
 VersionMajor=1
 VersionMinor=4
 VersionMicro=2
 VersionRelease=f
 LANG=C
 Success=1
 sizeof_char=1
 sizeof_short=2
 sizeof_int=4
 sizeof_long=8
 sizeof_void_ptr=8
 sizeof_size_t=8
 sizeof_float=4
 sizeof_double=8
 sizeof_s64Int=8
 sizeof_u64Int=8
 sizeof_struct_double_double=16
 CommWorldProcs=8
 MPI_Wtick=1.000000e-06
 HPL_Tflops=0.0101361
 HPL_time=4.52225
 HPL_eps=1.11022e-16
 HPL_RnormI=8.56681e-12
 HPL_AnormI=1055
 HPL_AnormI=1062.42
 HPL_XnormI=3501.5
 HPL_XnormI=4.12431
 HPL_BnormI=0.499778
 HPL_N=4096
 HPL_NB=128
 HPL_nprow=2

```

HPL_npcol=4
HPL_depth=1
HPL_nbdiv=2
HPL_nbmin=4
HPL_cpfact=R
HPL_crfact=C
HPL_ctop=1
HPL_order=R
HPL_dMACH_EPS=1.110223e-16
HPL_dMACH_SFMIN=2.225074e-308
HPL_dMACH_BASE=2.000000e+00
HPL_dMACH_PREC=2.220446e-16
HPL_dMACH_MLEN=5.300000e+01
HPL_dMACH_RND=1.000000e+00
HPL_dMACH_EMIN=-1.021000e+03
HPL_dMACH_RMIN=2.225074e-308
HPL_dMACH_EMAX=1.024000e+03
HPL_dMACH_RMAX=1.797693e+308
HPL_sMACH_EPS=5.960464e-08
HPL_sMACH_SFMIN=1.175494e-38
HPL_sMACH_BASE=2.000000e+00
HPL_sMACH_PREC=1.192093e-07
HPL_sMACH_MLEN=2.400000e+01
HPL_sMACH_RND=1.000000e+00
HPL_sMACH_EMIN=-1.250000e+02
HPL_sMACH_RMIN=1.175494e-38
HPL_sMACH_EMAX=1.280000e+02
HPL_sMACH_RMAX=3.402823e+38
dweps=1.110223e-16
sweps=5.960464e-08
HPLMaxProcs=8
HPLMinProcs=8
DGEMM_N=835
StarDGEMM_Gflops=1.39327
SingleDGEMM_Gflops=1.71124
PTRANS_GB=3.31701
PTRANS_time=0.00981402
PTRANS_residual=0
PTRANS_n=2048
PTRANS_nb=128
PTRANS_nprow=2
PTRANS_npcol=4
MPIRandomAccess_LCG_N=16777216
MPIRandomAccess_LCG_time=0.94314
MPIRandomAccess_LCG_CheckTime=0.267311
MPIRandomAccess_LCG_Errors=0
MPIRandomAccess_LCG_ErrorsFraction=0
MPIRandomAccess_LCG_ExeUpdates=67108864
MPIRandomAccess_LCG_GUPs=0.0711547
MPIRandomAccess_LCG_TimeBound=60
MPIRandomAccess_LCG_Algorithm=0
MPIRandomAccess_N=16777216
MPIRandomAccess_time=0.93397
MPIRandomAccess_CheckTime=0.267738
MPIRandomAccess_Errors=0
MPIRandomAccess_ErrorsFraction=0
MPIRandomAccess_ExeUpdates=67108864
MPIRandomAccess_GUPs=0.0718533
MPIRandomAccess_TimeBound=60
MPIRandomAccess_Algorithm=0
RandomAccess_LCG_N=2097152
StarRandomAccess_LCG_GUPs=0.0586357
SingleRandomAccess_LCG_GUPs=0.109031
RandomAccess_N=2097152
StarRandomAccess_GUPs=0.0569605
SingleRandomAccess_GUPs=0.122743

```

```

STREAM_VectorSize=699050
STREAM_Threads=1
StarSTREAM_Copy=6.00656
StarSTREAM_Scale=5.90271
StarSTREAM_Add=6.46392
StarSTREAM_Triad=7.23938
SingleSTREAM_Copy=11.5776
SingleSTREAM_Scale=11.8736
SingleSTREAM_Add=12.6064
SingleSTREAM_Triad=11.9492
FFT_N=524288
StarFFT_Gflops=1.53768
SingleFFT_Gflops=1.50222
MPIFFT_N=2097152
MPIFFT_Gflops=6.6518
MPIFFT_maxErr=1.57792e-15
MPIFFT_Procs=8
MaxPingPongLatency_usec=1.00003
RandomlyOrderedRingLatency_usec=1.29972
MinPingPongBandwidth_GBytes=3.70685
NaturallyOrderedRingBandwidth_GBytes=1.44588
RandomlyOrderedRingBandwidth_GBytes=1.38603
MinPingPongLatency_usec=0.384119
AvgPingPongLatency_usec=0.722263
MaxPingPongBandwidth_GBytes=6.94421
AvgPingPongBandwidth_GBytes=5.13036
NaturallyOrderedRingLatency_usec=1.3113
FFTEblk=16
FFTEnp=8
FFTEl2size=1048576
M_OPENMP=-1
omp_get_num_threads=0
omp_get_max_threads=0
omp_get_num_procs=0
MemProc=-1
MemSpec=-1
MemVal=-1
MPIFFT_time0=0
MPIFFT_time1=0.00531816
MPIFFT_time2=0.00451398
MPIFFT_time3=0.00327897
MPIFFT_time4=0.0137889
MPIFFT_time5=0.00451994
MPIFFT_time6=9.53674e-07
CPS_HPCC_FFT_235=0
CPS_HPCC_FFTW_ESTIMATE=0
CPS_HPCC_MEMALLCTR=0
CPS_HPL_USE_GETPROCESSTIMES=0
CPS_RA_SANDIA_NOPT=0
CPS_RA_SANDIA_OPT2=0
CPS_USING_FFTW=0
End of Summary section.
#####
End of HPC Challenge tests.
Current time (1423746356) is Thu Feb 12 07:05:56 2015

#####

```


APPENDIX B

IOR OUTPUT

The following is CTBS generated IOR output for the Random IOPS and Throughput IOR file system workload using the Posix Buffered IO API, The MPI-IO, and Posix Direct IO APIs in each of the execution environments.

B.1 Bare-metal

B.1.1 Random IOPS

B.1.1.1 MPI-IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Wed Feb 4 17:14:31 2015

Command line used: /root/CTBS/CTBS_BareMetal/ctbs_1.0.2/ior -b 100M -t 4K -z -a MPIIO -i 5

Machine: Linux benchmarks.local

Test 0 started: Wed Feb 4 17:14:31 2015

Summary:

```
api           = MPIIO (version=2, subversion=1)
test filename  = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB
```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	804.17	102400	4.00	0.021139	0.099726	0.003487	0.124352	0
read	2806.70	102400	4.00	0.000045	0.035553	0.000031	0.035629	0
remove	-	-	-	-	-	0.026045	0	
write	1003.10	102400	4.00	0.000155	0.099504	0.000032	0.099691	1
read	2796.02	102400	4.00	0.000041	0.035693	0.000031	0.035765	1
remove	-	-	-	-	-	0.026031	1	
write	1004.03	102400	4.00	0.000159	0.099411	0.000029	0.099599	2
read	2793.54	102400	4.00	0.000067	0.035700	0.000030	0.035797	2
remove	-	-	-	-	-	0.026084	2	
write	1002.49	102400	4.00	0.000173	0.099549	0.000030	0.099752	3
read	2805.84	102400	4.00	0.000040	0.035571	0.000029	0.035640	3
remove	-	-	-	-	-	0.026018	3	
write	1003.76	102400	4.00	0.000154	0.099440	0.000031	0.099625	4
read	2799.39	102400	4.00	0.000039	0.035653	0.000030	0.035722	4
remove	-	-	-	-	-	0.026037	4	

Max Write: 1004.03 MiB/sec (1052.80 MB/sec)

Max Read: 2806.70 MiB/sec (2943.04 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt	blksiz	xsize	aggsize	API	RefNum									
write	1004.03	804.17	963.51	79.67	0.10460	0	1	1	5	0	0	1	0	1
read	2806.70	2793.54	2800.30	5.22	0.03571	0	1	1	5	0	0	1	0	1

Finished: Wed Feb 4 17:14:31 2015

B.1.1.2 Posix Buffered IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Wed Feb 4 16:48:41 2015

Command line used: /root/CTBS/CTBS_BareMetal/ctbs_1.0.2/ior -b 100M -t 4K -z -i 5

Machine: Linux benchmarks.local

Test 0 started: Wed Feb 4 16:48:41 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	1077.52	102400	4.00	0.000047	0.092750	0.000008	0.092806	0
read	2999.49	102400	4.00	0.000008	0.033324	0.000006	0.033339	0
remove	-	-	-	-	-	0.025918	0	
write	959.84	102400	4.00	0.000102	0.104072	0.000008	0.104184	1
read	2991.15	102400	4.00	0.000009	0.033417	0.000006	0.033432	1
remove	-	-	-	-	-	0.025947	1	
write	1077.75	102400	4.00	0.000111	0.092668	0.000007	0.092786	2
read	2996.42	102400	4.00	0.000008	0.033358	0.000006	0.033373	2
remove	-	-	-	-	-	0.026009	2	
write	1079.31	102400	4.00	0.000107	0.092537	0.000006	0.092652	3
read	2992.04	102400	4.00	0.000006	0.033408	0.000006	0.033422	3
remove	-	-	-	-	-	0.025932	3	
write	1081.40	102400	4.00	0.000110	0.092355	0.000007	0.092473	4
read	2991.23	102400	4.00	0.000007	0.033418	0.000006	0.033431	4
remove	-	-	-	-	-	0.025896	4	

Max Write: 1081.40 MiB/sec (1133.93 MB/sec)

Max Read: 2999.49 MiB/sec (3145.19 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed						
segcnt	blksize	xsize	aggsz	API	RefNum															
write	1081.40	959.84	1055.16	47.68	0.09498	0	1	1	5	0	0	1	0	0	1	104857600	4096	104857600	POSIX	0
read	2999.49	2991.15	2994.07	3.34	0.03340	0	1	1	5	0	0	1	0	0	1	104857600	4096	104857600	POSIX	0

Finished: Wed Feb 4 16:48:41 2015

B.1.1.3 Posix Direct IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Wed Feb 4 16:48:41 2015

Command line used: /root/CTBS/CTBS_BareMetal/ctbs_1.0.2/ior -b 100M -t 4K -B -z -i 5

Machine: Linux benchmarks.local

Test 0 started: Wed Feb 4 16:48:41 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	0.487836	102400	4.00	0.000057	204.99	0.000061	204.99	0
read	0.724084	102400	4.00	0.000006	138.11	0.000008	138.11	0
remove	-	-	-	-	0.001674	0		
write	0.564256	102400	4.00	0.000061	177.22	0.000058	177.22	1
read	0.923963	102400	4.00	0.000005	108.23	0.000007	108.23	1
remove	-	-	-	-	0.001698	1		
write	0.518641	102400	4.00	0.000063	192.81	0.000058	192.81	2
read	0.950986	102400	4.00	0.000004	105.15	0.000008	105.15	2
remove	-	-	-	-	0.001654	2		
write	0.487821	102400	4.00	0.000058	204.99	0.000061	204.99	3
read	0.892541	102400	4.00	0.000004	112.04	0.000011	112.04	3
remove	-	-	-	-	0.001667	3		
write	0.499593	102400	4.00	0.000055	200.16	0.000056	200.16	4
read	0.949323	102400	4.00	0.000005	105.34	0.000008	105.34	4
remove	-	-	-	-	0.001683	4		

Max Write: 0.56 MiB/sec (0.59 MB/sec)

Max Read: 0.95 MiB/sec (1.00 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt	blksize	xsize	aggsz	API	RefNum									
write	0.56	0.49	0.51	0.03	196.03590	0	1	1	5	0	1	0	0	1
read	0.95	0.72	0.89	0.08	113.77339	0	1	1	5	0	1	0	0	1

Finished: Wed Feb 4 17:14:31.

B.1.2 Throughput

B.1.2.1 MPI-IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Wed Feb 4 17:16:30 2015

Command line used: /root/CTBS/CTBS_BareMetal/ctbs_1.0.2/ior -b 1G -t 2M -a MPIIO -i 5

Machine: Linux benchmarks.local

Test 0 started: Wed Feb 4 17:16:30 2015

Summary:

```

api           = MPIIO (version=2, subversion=1)
test filename = testFile
access        = single-shared-file
ordering in a file = sequential offsets
ordering inter file = no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 2 MiB
blocksize     = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	1301.00	1048576	2048.00	0.000757	0.786247	0.000081	0.787087	0
read	5026	1048576	2048.00	0.000044	0.203643	0.000035	0.203722	0
remove	-	-	-	-	0.218083	0		
write	1319.99	1048576	2048.00	0.000262	0.775469	0.000032	0.775764	1
read	5043	1048576	2048.00	0.000040	0.202977	0.000029	0.203046	1
remove	-	-	-	-	0.218132	1		
write	1320.18	1048576	2048.00	0.000226	0.775388	0.000035	0.775649	2
read	5039	1048576	2048.00	0.000042	0.203129	0.000029	0.203200	2
remove	-	-	-	-	0.218069	2		
write	1320.24	1048576	2048.00	0.000216	0.775368	0.000031	0.775615	3
read	5040	1048576	2048.00	0.000041	0.203083	0.000031	0.203155	3
remove	-	-	-	-	0.218214	3		
write	1320.67	1048576	2048.00	0.000224	0.775108	0.000034	0.775366	4
read	5043	1048576	2048.00	0.000040	0.202983	0.000031	0.203054	4
remove	-	-	-	-	0.218052	4		

Max Write: 1320.67 MiB/sec (1384.82 MB/sec)

Max Read: 5043.19 MiB/sec (5288.17 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt	blksiz	xsize	aggsize	API	RefNum									
write	1320.67	1301.00	1316.42	7.71	0.77790	0	1	1	5	0	1	0	0	1
read	5043.19	5026.46	5038.50	6.20	0.20324	0	1	1	5	0	1	0	0	1

1073741824 2097152 1073741824 MPIIO 0

Finished: Wed Feb 4 17:16:36 2015

B.1.2.2 Posix Buffered IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Wed Feb 4 17:14:32 2015

Command line used: /root/CTBS/CTBS_BareMetal/ctbs_1.0.2/ior -b 1G -t 2M -i 5

Machine: Linux benchmarks.local

Test 0 started: Wed Feb 4 17:14:32 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = sequential offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 2 MiB
blocksize     = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	1314.92	1048576	2048.00	0.000047	0.778694	0.000009	0.778752	0
read	5041	1048576	2048.00	0.000009	0.203101	0.000006	0.203116	0
remove	-	-	-	-	0.218558	0		
write	1318.04	1048576	2048.00	0.000111	0.776791	0.000007	0.776911	1
read	5042	1048576	2048.00	0.000007	0.203092	0.000007	0.203106	1
remove	-	-	-	-	0.218452	1		
write	1317.19	1048576	2048.00	0.000115	0.777287	0.000007	0.777411	2
read	5044	1048576	2048.00	0.000006	0.202995	0.000006	0.203009	2
remove	-	-	-	-	0.218523	2		
write	1316.38	1048576	2048.00	0.000110	0.777771	0.000008	0.777891	3
read	5042	1048576	2048.00	0.000007	0.203084	0.000006	0.203098	3
remove	-	-	-	-	0.218543	3		
write	1317.35	1048576	2048.00	0.000113	0.777194	0.000007	0.777316	4
read	5045	1048576	2048.00	0.000006	0.202940	0.000007	0.202956	4
remove	-	-	-	-	0.218457	4		

Max Write: 1318.04 MiB/sec (1382.07 MB/sec)

Max Read: 5045.42 MiB/sec (5290.51 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed						
segcnt	blksize	xsize	aggsz	API	RefNum															
write	1318.04	1314.92	1316.78	1.07	0.77766	0	1	1	5	0	0	1	0	0	1	1073741824	2097152	1073741824	POSIX	0
read	5045.42	5041.46	5042.92	1.57	0.20306	0	1	1	5	0	0	1	0	0	1	1073741824	2097152	1073741824	POSIX	0

Finished: Wed Feb 4 17:14:38 2015

B.1.2.3 Posix Direct IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Wed Feb 4 17:14:38 2015

Command line used: /root/CTBS/CTBS_BareMetal/ctbs_1.0.2/ior -b 1G -t 2M -B -i 5

Machine: Linux benchmarks.local

Test 0 started: Wed Feb 4 17:14:38 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = sequential offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 2 MiB
blocksize     = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	74.05	1048576	2048.00	0.000048	13.83	0.000009	13.83	0
read	112.17	1048576	2048.00	0.000005	9.13	0.000006	9.13	0
remove	-	-	-	-	0.014171	0		
write	73.08	1048576	2048.00	0.000048	14.01	0.000008	14.01	1
read	113.93	1048576	2048.00	0.000005	8.99	0.000009	8.99	1
remove	-	-	-	-	0.014241	1		
write	74.38	1048576	2048.00	0.000046	13.77	0.000009	13.77	2
read	122.99	1048576	2048.00	0.000005	8.33	0.000006	8.33	2
remove	-	-	-	-	0.014055	2		
write	72.30	1048576	2048.00	0.000047	14.16	0.000009	14.16	3
read	108.91	1048576	2048.00	0.000005	9.40	0.000008	9.40	3
remove	-	-	-	-	0.014235	3		
write	74.57	1048576	2048.00	0.000054	13.73	0.000008	13.73	4
read	136.75	1048576	2048.00	0.000004	7.49	0.000009	7.49	4
remove	-	-	-	-	0.014278	4		

Max Write: 74.57 MiB/sec (78.19 MB/sec)

Max Read: 136.75 MiB/sec (143.39 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed						
segcnt	blksiz	xsiz	aggsiz	API	RefNum															
write	74.57	72.30	73.67	0.86	13.90099	0	1	1	5	0	0	1	0	0	1	1073741824	2097152	1073741824	POSIX	0
read	136.75	108.91	118.95	10.05	8.66667	0	1	1	5	0	0	1	0	0	1	1073741824	2097152	1073741824	POSIX	0

Finished: Wed Feb 4 17:16:30 2015

B.2 Docker

B.2.1 Random IOPS

B.2.1.1 MPI-IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Sun Feb 8 16:34:47 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 100M -t 4K -z -a MPIIO -i 5

Machine: Linux CTBS

Test 0 started: Sun Feb 8 16:34:47 2015

Summary:

```

api           = MPIIO (version=2, subversion=1)
test filename = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	972.08	102400	4.00	0.000711	0.102086	0.000075	0.102872	0
read	2780.70	102400	4.00	0.000041	0.035889	0.000032	0.035962	0
remove	-	-	-	-	-	0.026892	0	
write	976.71	102400	4.00	0.000124	0.102229	0.000031	0.102384	1
read	2734.99	102400	4.00	0.000040	0.036493	0.000030	0.036563	1
remove	-	-	-	-	-	0.026831	1	
write	978.66	102400	4.00	0.000121	0.102030	0.000030	0.102181	2
read	2731.36	102400	4.00	0.000038	0.036543	0.000029	0.036612	2
remove	-	-	-	-	-	0.026783	2	
write	977.77	102400	4.00	0.000122	0.102119	0.000031	0.102274	3
read	2724.42	102400	4.00	0.000039	0.036637	0.000029	0.036705	3
remove	-	-	-	-	-	0.026821	3	
write	980.91	102400	4.00	0.000117	0.101799	0.000030	0.101946	4
read	2740.78	102400	4.00	0.000038	0.036418	0.000030	0.036486	4
remove	-	-	-	-	-	0.026785	4	

Max Write: 980.91 MiB/sec (1028.56 MB/sec)

Max Read: 2780.70 MiB/sec (2915.78 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt	blksiz	xsiz	aggsize	API	RefNum									
write	980.91	972.08	977.23	2.92	0.10233	0	1	1	5	0	1	0	0	1
read	2780.70	2724.42	2742.45	19.85	0.03647	0	1	1	5	0	1	0	0	1

104857600 4096 104857600 MPIIO 0
104857600 4096 104857600 MPIIO 0

Finished: Sun Feb 8 16:34:48 2015

B.2.1.2 Posix Buffered IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Sun Feb 8 16:09:30 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 100M -t 4K -z -i 5

Machine: Linux CTBS

Test 0 started: Sun Feb 8 16:09:30 2015

Summary:

```

api           = POSIX
test filename  = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file = no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	1044.90	102400	4.00	0.000034	0.095658	0.000011	0.095703	0
read	2948.98	102400	4.00	0.000009	0.033892	0.000008	0.033910	0
remove	-	-	-	-	0.026732	0		
write	1042.91	102400	4.00	0.000079	0.095797	0.000009	0.095886	1
read	2947.07	102400	4.00	0.000009	0.033914	0.000008	0.033932	1
remove	-	-	-	-	0.026700	1		
write	1045.17	102400	4.00	0.000081	0.095588	0.000008	0.095678	2
read	2945.60	102400	4.00	0.000008	0.033932	0.000009	0.033949	2
remove	-	-	-	-	0.026694	2		
write	1048.64	102400	4.00	0.000079	0.095274	0.000009	0.095362	3
read	2947.69	102400	4.00	0.000009	0.033908	0.000008	0.033925	3
remove	-	-	-	-	0.026708	3		
write	1049.34	102400	4.00	0.000078	0.095211	0.000009	0.095298	4
read	2947.94	102400	4.00	0.000009	0.033904	0.000008	0.033922	4
remove	-	-	-	-	0.026819	4		

Max Write: 1049.34 MiB/sec (1100.31 MB/sec)

Max Read: 2948.98 MiB/sec (3092.23 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed						
segcnt blksiz xsize aggsz API RefNum																				
write	1049.34	1042.91	1046.19	2.42	0.09559	0	1	1	5	0	0	1	0	0	1	104857600	4096	104857600	POSIX	0
read	2948.98	2945.60	2947.46	1.11	0.03393	0	1	1	5	0	0	1	0	0	1	104857600	4096	104857600	POSIX	0

Finished: Sun Feb 8 16:09:31 2015

B.2.1.3 Posix Direct IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Sun Feb 8 16:09:31 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 100M -t 4K -B -z -i 5

Machine: Linux CTBS

Test 0 started: Sun Feb 8 16:09:31 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	0.506318	102400	4.00	0.000035	197.50	0.000062	197.50	0
read	0.932593	102400	4.00	0.000006	107.23	0.000008	107.23	0
remove	-	-	-	-	0.001756	0		
write	0.512648	102400	4.00	0.000042	195.07	0.000060	195.07	1
read	0.873737	102400	4.00	0.000005	114.45	0.000009	114.45	1
remove	-	-	-	-	0.002048	1		
write	0.511100	102400	4.00	0.000045	195.66	0.000064	195.66	2
read	0.945465	102400	4.00	0.000006	105.77	0.000008	105.77	2
remove	-	-	-	-	0.002004	2		
write	0.522116	102400	4.00	0.000035	191.53	0.000058	191.53	3
read	0.931020	102400	4.00	0.000005	107.41	0.000015	107.41	3
remove	-	-	-	-	0.002650	3		
write	0.514611	102400	4.00	0.000035	194.32	0.000061	194.32	4
read	0.932350	102400	4.00	0.000005	107.26	0.000009	107.26	4
remove	-	-	-	-	0.001859	4		

Max Write: 0.52 MiB/sec (0.55 MB/sec)

Max Read: 0.95 MiB/sec (0.99 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed						
segcnt	blksize	xsize	aggsz	API	RefNum															
write	0.52	0.51	0.51	0.01	194.81531	0	1	1	5	0	0	1	0	0	1	104857600	4096	104857600	POSIX	0
read	0.95	0.87	0.92	0.03	108.42238	0	1	1	5	0	0	1	0	0	1	104857600	4096	104857600	POSIX	0

Finished: Sun Feb 8 16:34:47 2015

B.2.2 Throughput

B.2.2.1 MPI-IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Sun Feb 8 16:09:24 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 1G -t 2M -a MPIIO -i 5

Machine: Linux CTBS

Test 0 started: Sun Feb 8 16:09:24 2015

Summary:

```

api           = MPIIO (version=2, subversion=1)
test filename = testFile
access        = single-shared-file
ordering in a file = sequential offsets
ordering inter file = no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 2 MiB
blocksize     = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	1203.08	1048576	2048.00	0.000756	0.850319	0.000071	0.851149	0
read	5034	1048576	2048.00	0.000042	0.203350	0.000029	0.203421	0
remove	-	-	-	-	0.224285	0		
write	1187.68	1048576	2048.00	0.000182	0.861969	0.000033	0.862186	1
read	5025	1048576	2048.00	0.000039	0.203719	0.000031	0.203789	1
remove	-	-	-	-	0.224764	1		
write	1186.97	1048576	2048.00	0.000194	0.862471	0.000034	0.862699	2
read	5031	1048576	2048.00	0.000039	0.203485	0.000031	0.203555	2
remove	-	-	-	-	0.224488	2		
write	1188.03	1048576	2048.00	0.000191	0.861703	0.000034	0.861928	3
read	5032	1048576	2048.00	0.000038	0.203421	0.000032	0.203491	3
remove	-	-	-	-	0.224930	3		
write	1196.46	1048576	2048.00	0.016600	0.839223	0.000036	0.855861	4
read	5025	1048576	2048.00	0.000041	0.203723	0.000031	0.203796	4
remove	-	-	-	-	0.224912	4		

Max Write: 1203.08 MiB/sec (1261.52 MB/sec)

Max Read: 5033.90 MiB/sec (5278.42 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt	blksiz	xsiz	aggsize	API	RefNum									
write	1203.08	1186.97	1192.44	6.35	0.85876	0	1	1	5	0	1	0	0	1
read	5033.90	5024.63	5029.22	3.82	0.20361	0	1	1	5	0	1	0	0	1

Finished: Sun Feb 8 16:09:30 2015

B.2.2.2 Posix Buffered IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Sun Feb 8 16:07:23 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 1G -t 2M -i 5

Machine: Linux CTBS

Test 0 started: Sun Feb 8 16:07:23 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = sequential offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 2 MiB
blocksize     = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	1256.28	1048576	2048.00	0.000037	0.815057	0.000011	0.815107	0
read	5029	1048576	2048.00	0.000009	0.203603	0.000009	0.203622	0
remove	-	-	-	-	0.224033	0		
write	1264.71	1048576	2048.00	0.000087	0.809576	0.000009	0.809674	1
read	5028	1048576	2048.00	0.000009	0.203621	0.000008	0.203640	1
remove	-	-	-	-	0.224631	1		
write	1264.47	1048576	2048.00	0.000085	0.809730	0.000009	0.809826	2
read	5019	1048576	2048.00	0.000009	0.204012	0.000009	0.204032	2
remove	-	-	-	-	0.224129	2		
write	1264.62	1048576	2048.00	0.000085	0.809635	0.000009	0.809731	3
read	5025	1048576	2048.00	0.000009	0.203769	0.000008	0.203788	3
remove	-	-	-	-	0.224206	3		
write	1269.08	1048576	2048.00	0.000079	0.806793	0.000009	0.806883	4
read	5024	1048576	2048.00	0.000008	0.203816	0.000008	0.203832	4
remove	-	-	-	-	0.224159	4		

Max Write: 1269.08 MiB/sec (1330.73 MB/sec)

Max Read: 5028.93 MiB/sec (5273.21 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed						
segcnt	blksize	xsize	aggsz	API	RefNum															
write	1269.08	1256.28	1263.83	4.16	0.81024	0	1	1	5	0	0	1	0	0	1	1073741824	2097152	1073741824	POSIX	0
read	5028.93	5018.82	5024.96	3.67	0.20378	0	1	1	5	0	0	1	0	0	1	1073741824	2097152	1073741824	POSIX	0

Finished: Sun Feb 8 16:07:29 2015

B.2.2.3 Posix Direct IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Sun Feb 8 16:07:29 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 1G -t 2M -B -i 5

Machine: Linux CTBS

Test 0 started: Sun Feb 8 16:07:29 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = sequential offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 2 MiB
blocksize     = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	73.16	1048576	2048.00	0.000039	14.00	0.000007	14.00	0
read	113.83	1048576	2048.00	0.000006	9.00	0.000006	9.00	0
remove	-	-	-	-	0.014758	0		
write	72.00	1048576	2048.00	0.000043	14.22	0.000008	14.22	1
read	106.62	1048576	2048.00	0.000007	9.60	0.000006	9.60	1
remove	-	-	-	-	0.014268	1		
write	74.77	1048576	2048.00	0.000037	13.69	0.000008	13.69	2
read	115.32	1048576	2048.00	0.000005	8.88	0.000010	8.88	2
remove	-	-	-	-	0.015945	2		
write	71.92	1048576	2048.00	0.000039	14.24	0.000007	14.24	3
read	107.00	1048576	2048.00	0.000006	9.57	0.000007	9.57	3
remove	-	-	-	-	0.014270	3		
write	74.50	1048576	2048.00	0.000039	13.75	0.000008	13.75	4
read	129.27	1048576	2048.00	0.000005	7.92	0.000008	7.92	4
remove	-	-	-	-	0.014232	4		

Max Write: 74.77 MiB/sec (78.41 MB/sec)

Max Read: 129.27 MiB/sec (135.55 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed						
segcnt																				
blksize																				
xsize																				
aggsize																				
API RefNum																				
write	74.77	71.92	73.27	1.20	13.97950	0	1	1	5	0	0	1	0	0	1	1073741824	2097152	1073741824	POSIX	0
read	129.27	106.62	114.41	8.22	8.99422	0	1	1	5	0	0	1	0	0	1	1073741824	2097152	1073741824	POSIX	0

Finished: Sun Feb 8 16:09:24 2015

B.3 KVM

B.3.1 Random IOPS

B.3.1.1 MPI-IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Thu Feb 12 07:05:41 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 100M -t 4K -z -a MPIIO -i 5

Machine: Linux KVM.localdomain

Test 0 started: Thu Feb 12 07:05:41 2015

Summary:

```

api           = MPIIO (version=2, subversion=1)
test filename = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	900.33	102400	4.00	0.000793	0.110198	0.000078	0.111070	0
read	2071.89	102400	4.00	0.000045	0.048179	0.000039	0.048265	0
remove	-	-	-	-	-	0.031035	0	
write	899.62	102400	4.00	0.000214	0.110909	0.000034	0.111158	1
read	2058.34	102400	4.00	0.000044	0.048504	0.000034	0.048583	1
remove	-	-	-	-	-	0.031059	1	
write	904.16	102400	4.00	0.000173	0.110391	0.000036	0.110600	2
read	2091.44	102400	4.00	0.000042	0.047732	0.000039	0.047814	2
remove	-	-	-	-	-	0.031041	2	
write	897.36	102400	4.00	0.000236	0.111165	0.000033	0.111438	3
read	2054.52	102400	4.00	0.000042	0.048588	0.000042	0.048673	3
remove	-	-	-	-	-	0.031074	3	
write	901.89	102400	4.00	0.000155	0.110684	0.000038	0.110878	4
read	2147.72	102400	4.00	0.000042	0.046484	0.000034	0.046561	4
remove	-	-	-	-	-	0.030027	4	

Max Write: 904.16 MiB/sec (948.08 MB/sec)

Max Read: 2147.72 MiB/sec (2252.05 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt	blksiz	xsiz	aggsize	API	RefNum									
write	904.16	897.36	900.67	2.27	0.11103	0	1	1	5	0	1	0	0	1
read	2147.72	2054.52	2084.78	34.02	0.04798	0	1	1	5	0	1	0	0	1

104857600 4096 104857600 MPIIO 0
104857600 4096 104857600 MPIIO 0

Finished: Thu Feb 12 07:05:42 2015

B.3.1.2 Posix Buffered IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Thu Feb 12 06:40:08 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 100M -t 4K -z -i 5

Machine: Linux KVM.localdomain

Test 0 started: Thu Feb 12 06:40:08 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	937.79	102400	4.00	0.000050	0.106569	0.000009	0.106633	0
read	2166.42	102400	4.00	0.000008	0.046143	0.000008	0.046159	0
remove	-	-	-	-	-	0.029816	0	
write	1007.78	102400	4.00	0.000113	0.099108	0.000006	0.099228	1
read	2199.16	102400	4.00	0.000008	0.045455	0.000007	0.045472	1
remove	-	-	-	-	-	0.029910	1	
write	983.41	102400	4.00	0.000104	0.101573	0.000006	0.101687	2
read	2185.17	102400	4.00	0.000008	0.045747	0.000007	0.045763	2
remove	-	-	-	-	-	0.030342	2	
write	1006.20	102400	4.00	0.000120	0.099256	0.000006	0.099384	3
read	2207.60	102400	4.00	0.000008	0.045282	0.000007	0.045298	3
remove	-	-	-	-	-	0.029853	3	
write	1005.48	102400	4.00	0.000099	0.099349	0.000006	0.099455	4
read	2219.31	102400	4.00	0.000008	0.045043	0.000007	0.045059	4
remove	-	-	-	-	-	0.029942	4	

Max Write: 1007.78 MiB/sec (1056.73 MB/sec)

Max Read: 2219.31 MiB/sec (2327.12 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt														
blksize														
xsize														
aggsize														
API RefNum														
write	1007.78	937.79	988.13	26.72	0.10128	0	1	1	5	0	1	0	0	1
read	2219.31	2166.42	2195.53	18.32	0.04555	0	1	1	5	0	0	1	0	1

Finished: Thu Feb 12 06:40:09 2015

B.3.1.3 Posix Direct IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Thu Feb 12 06:40:09 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 100M -t 4K -B -z -i 5

Machine: Linux KVM.localdomain

Test 0 started: Thu Feb 12 06:40:09 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = random offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 4096 bytes
blocksize     = 100 MiB
aggregate filesize = 100 MiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	0.508830	102400	4.00	0.000053	196.53	0.000074	196.53	0
read	0.917927	102400	4.00	0.000010	108.94	0.000011	108.94	0
remove	-	-	-	-	0.002413	0		
write	0.509118	102400	4.00	0.000121	196.42	0.000065	196.42	1
read	0.899210	102400	4.00	0.000011	111.21	0.000009	111.21	1
remove	-	-	-	-	0.002042	1		
write	0.510913	102400	4.00	0.000126	195.73	0.000069	195.73	2
read	0.894906	102400	4.00	0.000009	111.74	0.000021	111.74	2
remove	-	-	-	-	0.001898	2		
write	0.516384	102400	4.00	0.000121	193.65	0.000065	193.65	3
read	0.924846	102400	4.00	0.000012	108.13	0.000011	108.13	3
remove	-	-	-	-	0.001800	3		
write	0.502623	102400	4.00	0.000143	198.96	0.000064	198.96	4
read	0.905469	102400	4.00	0.000012	110.44	0.000016	110.44	4
remove	-	-	-	-	0.001932	4		

Max Write: 0.52 MiB/sec (0.54 MB/sec)

Max Read: 0.92 MiB/sec (0.97 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt														
blksize														
xsize														
aggsize														
API RefNum														
write	0.52	0.50	0.51	0.00	196.25720	0	1	1	5	0	1	0	0	1
read	0.92	0.89	0.91	0.01	110.09191	0	1	1	5	0	1	0	0	1

Finished: Thu Feb 12 07:05:41 2015

B.3.2 Throughput

B.3.2.1 MPI-IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Thu Feb 12 06:40:02 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 1G -t 2M -a MPIIO -i 5

Machine: Linux KVM.localdomain

Test 0 started: Thu Feb 12 06:40:02 2015

Summary:

```

api           = MPIIO (version=2, subversion=1)
test filename = testFile
access        = single-shared-file
ordering in a file = sequential offsets
ordering inter file = no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 2 MiB
blocksize     = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	1236.72	1048576	2048.00	0.000831	0.827068	0.000090	0.827994	0
read	3489.32	1048576	2048.00	0.000045	0.293381	0.000039	0.293467	0
remove	-	-	-	-	0.223479	0		
write	1220.22	1048576	2048.00	0.000255	0.838894	0.000039	0.839190	1
read	3487.34	1048576	2048.00	0.000046	0.293542	0.000041	0.293633	1
remove	-	-	-	-	0.225073	1		
write	1224.49	1048576	2048.00	0.000380	0.835841	0.000039	0.836264	2
read	3492.41	1048576	2048.00	0.000043	0.293122	0.000040	0.293207	2
remove	-	-	-	-	0.224868	2		
write	1223.73	1048576	2048.00	0.000249	0.836490	0.000041	0.836785	3
read	3500.94	1048576	2048.00	0.000043	0.292406	0.000042	0.292493	3
remove	-	-	-	-	0.256075	3		
write	1214.47	1048576	2048.00	0.000264	0.842856	0.000045	0.843166	4
read	3480.33	1048576	2048.00	0.000044	0.294127	0.000049	0.294225	4
remove	-	-	-	-	0.227380	4		

Max Write: 1236.72 MiB/sec (1296.80 MB/sec)

Max Read: 3500.94 MiB/sec (3671.00 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt	blksiz	xsiz	aggsiz	API	RefNum									
write	1236.72	1214.47	1223.93	7.31	0.83668	0	1	1	5	0	1	0	0	1
read	3500.94	3480.33	3490.07	6.73	0.29341	0	1	1	5	0	1	0	0	1

1073741824 2097152 1073741824 MPIIO 0

Finished: Thu Feb 12 06:40:08 2015

B.3.2.2 Posix Buffered IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Thu Feb 12 06:37:35 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 1G -t 2M -i 5

Machine: Linux KVM.localdomain

Test 0 started: Thu Feb 12 06:37:35 2015

Summary:

```

api                = POSIX
test filename      = testFile
access             = single-shared-file
ordering in a file = sequential offsets
ordering inter file= no tasks offsets
clients           = 1 (1 per node)
repetitions        = 5
xfersize           = 2 MiB
blocksize          = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	1047.72	1048576	2048.00	0.000051	0.977290	0.000011	0.977358	0
read	4014	1048576	2048.00	0.000009	0.255077	0.000010	0.255099	0
remove	-	-	-	-	0.230073	0		
write	1324.50	1048576	2048.00	0.000120	0.772976	0.000008	0.773120	1
read	3988	1048576	2048.00	0.000008	0.256780	0.000010	0.256802	1
remove	-	-	-	-	0.230426	1		
write	1333.30	1048576	2048.00	0.000122	0.767881	0.000009	0.768018	2
read	4013	1048576	2048.00	0.000008	0.255133	0.000010	0.255153	2
remove	-	-	-	-	0.229803	2		
write	1293.55	1048576	2048.00	0.000116	0.791485	0.000010	0.791618	3
read	4083	1048576	2048.00	0.000011	0.250788	0.000010	0.250812	3
remove	-	-	-	-	0.228630	3		
write	1323.83	1048576	2048.00	0.000130	0.773364	0.000010	0.773512	4
read	4014	1048576	2048.00	0.000011	0.255081	0.000009	0.255104	4
remove	-	-	-	-	0.229960	4		

Max Write: 1333.30 MiB/sec (1398.07 MB/sec)

Max Read: 4082.74 MiB/sec (4281.06 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt	blksize	xsize	aggsz	API	RefNum									
write	1333.30	1047.72	1264.58	109.26	0.81673	0	1	1	5	0	0	1	1073741824	2097152
read	4082.74	3987.51	4022.34	31.87	0.25459	0	1	1	5	0	0	1	1073741824	2097152

Finished: Thu Feb 12 06:37:42 2015

B.3.2.3 Posix Direct IO

IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Thu Feb 12 06:37:42 2015

Command line used: /root/CTBS/ctbs_1.0.2/ior -b 1G -t 2M -B -i 5

Machine: Linux KVM.localdomain

Test 0 started: Thu Feb 12 06:37:42 2015

Summary:

```

api           = POSIX
test filename = testFile
access        = single-shared-file
ordering in a file = sequential offsets
ordering inter file= no tasks offsets
clients       = 1 (1 per node)
repetitions   = 5
xfersize      = 2 MiB
blocksize     = 1 GiB
aggregate filesize = 1 GiB

```

access	bw(MiB/s)	block(KiB)	xfer(KiB)	open(s)	wr/rd(s)	close(s)	total(s)	iter
write	52.66	1048576	2048.00	0.000063	19.45	0.000018	19.45	0
read	128.07	1048576	2048.00	0.000010	8.00	0.000008	8.00	0
remove	-	-	-	-	0.019199	0		
write	51.91	1048576	2048.00	0.000116	19.73	0.000010	19.73	1
read	123.36	1048576	2048.00	0.000009	8.30	0.000011	8.30	1
remove	-	-	-	-	0.017154	1		
write	51.88	1048576	2048.00	0.000113	19.74	0.000011	19.74	2
read	119.88	1048576	2048.00	0.000011	8.54	0.000008	8.54	2
remove	-	-	-	-	0.021408	2		
write	52.49	1048576	2048.00	0.000120	19.51	0.000010	19.51	3
read	123.82	1048576	2048.00	0.000008	8.27	0.000012	8.27	3
remove	-	-	-	-	0.013745	3		
write	52.39	1048576	2048.00	0.000116	19.54	0.000010	19.54	4
read	123.07	1048576	2048.00	0.000010	8.32	0.000007	8.32	4
remove	-	-	-	-	0.021695	4		

Max Write: 52.66 MiB/sec (55.22 MB/sec)

Max Read: 128.07 MiB/sec (134.29 MB/sec)

Summary of all tests:

Operation	Max(MiB)	Min(MiB)	Mean(MiB)	StdDev	Mean(s)	Test#	#Tasks	tPN	reps	fPP	reord	reordoff	reordrand	seed
segcnt														
blksize														
xsize														
aggsize														
API RefNum														
write	52.66	51.88	52.27	0.31	19.59252	0	1	1	5	0	0	1	0	0
read	128.07	119.88	123.64	2.62	8.28578	0	1	1	5	0	0	1	0	0

Finished: Thu Feb 12 06:40:02 2015

APPENDIX C

ENVIRONMENT INFORMATION

The following is CTBS generated machine environment output in each of the execution environments.

C.1 Bare-metal

C.1.1 Machine Environment

cpuinfo information

```
processor : 0
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.042
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 0
cpu cores  : 6
apicid     : 0
initial apicid : 0
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4200.08
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 1
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.042
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 1
cpu cores  : 6
apicid     : 2
initial apicid : 2
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
```

```

flags                : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popent
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.08
clflush size        : 64
cache_alignment     : 64
address sizes       : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 2
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.042
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 2
cpu cores  : 6
apicid     : 4
initial apicid : 4
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popent
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.08
clflush size        : 64
cache_alignment     : 64
address sizes       : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 3
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.042
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 3
cpu cores  : 6
apicid     : 6
initial apicid : 6
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popent
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.08
clflush size        : 64
cache_alignment     : 64

```

address sizes : 46 bits physical, 48 bits virtual
power management:

```
processor : 4
vendor_id : GenuineIntel
cpu family : 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id: 0
siblings : 12
core id : 4
cpu cores : 6
apicid : 8
initial apicid : 8
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.08
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 5
vendor_id : GenuineIntel
cpu family : 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id: 0
siblings : 12
core id : 5
cpu cores : 6
apicid : 10
initial apicid : 10
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.08
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 6
vendor_id : GenuineIntel
cpu family : 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
```

```

stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id: 1
siblings : 12
core id : 0
cpu cores : 6
apicid : 32
initial apicid : 32
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.42
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 7
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id: 1
siblings : 12
core id : 1
cpu cores : 6
apicid : 34
initial apicid : 34
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.42
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 8
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id: 1
siblings : 12
core id : 2
cpu cores : 6

```

```

apicid          : 36
initial apicid  : 36
fpu             : yes
fpu_exception   : yes
cpuid level: 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips      : 4199.42
clflush size   : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

```

```

processor      : 9
vendor_id     : GenuineIntel
cpu family    : 6
model         : 62
model name    : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode    : 1045
cpu MHz       : 2100.042
cache size    : 15360 KB
physical id   : 1
siblings      : 12
core id       : 3
cpu cores     : 6
apicid        : 38
initial apicid : 38
fpu           : yes
fpu_exception : yes
cpuid level: 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips      : 4199.42
clflush size   : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

```

```

processor      : 10
vendor_id     : GenuineIntel
cpu family    : 6
model         : 62
model name    : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode    : 1045
cpu MHz       : 2100.042
cache size    : 15360 KB
physical id   : 1
siblings      : 12
core id       : 4
cpu cores     : 6
apicid        : 40
initial apicid : 40
fpu           : yes
fpu_exception : yes
cpuid level: 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf

```



```

pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.42
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

processor : 11
vendor_id : GenuineIntel
cpu family : 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id : 1
siblings : 12
core id : 5
cpu cores : 6
apicid : 42
initial apicid : 42
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.42
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

processor : 12
vendor_id : GenuineIntel
cpu family : 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id : 0
siblings : 12
core id : 0
cpu cores : 6
apicid : 1
initial apicid : 1
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.08
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```
processor : 13
vendor_id : GenuineIntel
cpu family : 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id : 0
siblings : 12
core id : 1
cpu cores : 6
apicid : 3
initial apicid : 3
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.08
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 14
vendor_id : GenuineIntel
cpu family : 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.042
cache size : 15360 KB
physical id : 0
siblings : 12
core id : 2
cpu cores : 6
apicid : 5
initial apicid : 5
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.08
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 15
vendor_id : GenuineIntel
cpu family : 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
```

```

cpu MHz          : 2100.042
cache size      : 15360 KB
physical id: 0
siblings       : 12
core id        : 3
cpu cores      : 6
apicid         : 7
initial apicid : 7
fpu            : yes
fpu_exception  : yes
cpuid level: 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips       : 4200.08
clflush size   : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 16
vendor_id      : GenuineIntel
cpu family: 6
model          : 62
model name     : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode     : 1045
cpu MHz       : 2100.042
cache size    : 15360 KB
physical id: 0
siblings      : 12
core id       : 4
cpu cores     : 6
apicid        : 9
initial apicid : 9
fpu           : yes
fpu_exception : yes
cpuid level: 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips     : 4200.08
clflush size : 64
cache_alignment : 64
address sizes  : 46 bits physical, 48 bits virtual
power management:

processor       : 17
vendor_id      : GenuineIntel
cpu family: 6
model          : 62
model name     : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode     : 1045
cpu MHz       : 2100.042
cache size    : 15360 KB
physical id: 0
siblings      : 12
core id       : 5
cpu cores     : 6
apicid        : 11
initial apicid : 11

```

```

fpu                : yes
fpu_exception      : yes
cpuid level: 13
wp                : yes
flags              : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4200.08
clflush size      : 64
cache_alignment  : 64
address sizes     : 46 bits physical, 48 bits virtual
power management:

```

```

processor   : 18
vendor_id   : GenuineIntel
cpu family : 6
model       : 62
model name   : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping    : 4
microcode   : 1045
cpu MHz     : 2100.042
cache size  : 15360 KB
physical id : 1
siblings    : 12
core id     : 0
cpu cores   : 6
apicid      : 33
initial apicid : 33
fpu         : yes
fpu_exception : yes
cpuid level: 13
wp          : yes
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4199.42
clflush size      : 64
cache_alignment  : 64
address sizes     : 46 bits physical, 48 bits virtual
power management:

```

```

processor   : 19
vendor_id   : GenuineIntel
cpu family : 6
model       : 62
model name   : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping    : 4
microcode   : 1045
cpu MHz     : 2100.042
cache size  : 15360 KB
physical id : 1
siblings    : 12
core id     : 1
cpu cores   : 6
apicid      : 35
initial apicid : 35
fpu         : yes
fpu_exception : yes
cpuid level: 13
wp          : yes
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt

```

```

tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.42
clflush size      : 64
cache_alignment   : 64
address sizes     : 46 bits physical, 48 bits virtual
power management:

processor : 20
vendor_id : GenuineIntel
cpu family: 6
model     : 62
model name: Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping  : 4
microcode : 1045
cpu MHz   : 2100.042
cache size: 15360 KB
physical id: 1
siblings  : 12
core id   : 2
cpu cores : 6
apicid    : 37
initial apicid : 37
fpu       : yes
fpu_exception : yes
cpuid level: 13
wp        : yes
flags     : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.42
clflush size      : 64
cache_alignment   : 64
address sizes     : 46 bits physical, 48 bits virtual
power management:

processor : 21
vendor_id : GenuineIntel
cpu family: 6
model     : 62
model name: Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping  : 4
microcode : 1045
cpu MHz   : 2100.042
cache size: 15360 KB
physical id: 1
siblings  : 12
core id   : 3
cpu cores : 6
apicid    : 39
initial apicid : 39
fpu       : yes
fpu_exception : yes
cpuid level: 13
wp        : yes
flags     : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.42
clflush size      : 64
cache_alignment   : 64
address sizes     : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 22
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.042
cache size : 15360 KB
physical id: 1
siblings   : 12
core id    : 4
cpu cores  : 6
apicid     : 41
initial apicid : 41
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4199.42
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 23
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.042
cache size : 15360 KB
physical id: 1
siblings   : 12
core id    : 5
cpu cores  : 6
apicid     : 43
initial apicid : 43
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4199.42
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

meminfo information

```

MemTotal: 65941420 kB
MemFree: 65294804 kB
Buffers: 8316 kB
Cached: 46440 kB
SwapCached: 0 kB

```

```

Active:      28816 kB
Inactive:    38320 kB
Active(anon): 12444 kB
Inactive(anon): 240 kB
Active(file): 16372 kB
Inactive(file): 38080 kB
Unevictable: 0 kB
Mlocked:     0 kB
SwapTotal:   24313852 kB
SwapFree:    24313852 kB
Dirty:       4 kB
Writeback:   0 kB
AnonPages:   12544 kB
Mapped:      6328 kB
Shmem:       284 kB
Slab:        97404 kB
SReclaimable: 12748 kB
SUnreclaim:  84656 kB
KernelStack: 3336 kB
PageTables:  1508 kB
NFS_Unstable: 0 kB
Bounce:      0 kB
WritebackTmp: 0 kB
CommitLimit: 57284560 kB
Committed_AS: 62824 kB
VmallocTotal: 34359738367 kB
VmallocUsed:  391540 kB
VmallocChunk: 34323885640 kB
HardwareCorrupted: 0 kB
AnonHugePages: 0 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB
DirectMap4k: 4096 kB
DirectMap2M: 2084864 kB
DirectMap1G: 65011712 kB

```

df (file system) information

Filesystem	1M-blocks	Used	Available	Use%	Mounted on
/dev/mapper/vg_benchmarks-lv_root	50269	5427	42282	12%	/
proc	0	0	0	-	/proc
sysfs	0	0	0	-	/sys
devpts	0	0	0	-	/dev/pts
tmpfs	32198	0	32198	0%	/dev/shm
/dev/sda1	477	49	403	11%	/boot
/dev/mapper/vg_benchmarks-lv_home	159346	60	151186	1%	/home
none	0	0	0	-	/proc/sys/fs/binfmt_misc

disk file system information

Linux 2.6.32-504.8.1.el6.x86_64 (benchmarks.local) 02/04/2015 _x86_64_ (24 CPU)

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	32.24	1292.87	22.97	119862	2130
sdh	3.89	31.15	0.00	2888	0
sdi	3.89	31.15	0.00	2888	0
sdf	3.89	31.15	0.00	2888	0
sdg	3.89	31.15	0.00	2888	0
sdh	3.89	31.15	0.00	2888	0
sdb	3.89	31.15	0.00	2888	0
sdd	3.89	31.15	0.00	2888	0
sde	3.89	31.15	0.00	2888	0
dm-0	36.86	1183.84	22.52	109754	2088

dm-1	3.20	25.63	0.00	2376	0
dm-2	2.42	19.01	0.26	1762	24

network file system information

Linux 2.6.32-504.8.1.el6.x86_64 (benchmarks.local) 02/04/2015 _x86_64_ (24 CPU)

Filesystem: rBlk_nor/s wBlk_nor/s rBlk_dir/s wBlk_dir/s rBlk_svr/s wBlk_svr/s ops/s rops/s wops/s

C.1.2 Software Version Environment

gcc information:

gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-11)
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

System information:

Linux benchmarks.local 2.6.32-504.8.1.el6.x86_64 #1 SMP Wed Jan 28 21:11:36 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux

OS information:

CentOS release 6.6 (Final)

C.2 Docker

C.2.1 Machine Environment

cpuinfo information

```
processor : 0
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode : 1045
cpu MHz    : 2100.025
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 0
cpu cores  : 6
apicid     : 0
initial apicid : 0
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4200.05
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 1
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode : 1045
cpu MHz    : 2100.025
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 1
cpu cores  : 6
apicid     : 2
initial apicid : 2
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4200.05
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 2
vendor_id : GenuineIntel
```

```

cpu family : 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.025
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 2
cpu cores  : 6
apicid     : 4
initial apicid : 4
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4200.05
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 3
vendor_id : GenuineIntel
cpu family : 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.025
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 3
cpu cores  : 6
apicid     : 6
initial apicid : 6
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips   : 4200.05
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 4
vendor_id : GenuineIntel
cpu family : 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.025
cache size : 15360 KB
physical id: 0

```

```

siblings      : 12
core id       : 4
cpu cores    : 6
apicid        : 8
initial apicid : 8
fpu           : yes
fpu_exception : yes
cpuid level: 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips     : 4200.05
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor      : 5
vendor_id     : GenuineIntel
cpu family    : 6
model         : 62
model name    : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode     : 1045
cpu MHz       : 2100.025
cache size    : 15360 KB
physical id: 0
siblings      : 12
core id       : 5
cpu cores    : 6
apicid        : 10
initial apicid : 10
fpu           : yes
fpu_exception : yes
cpuid level: 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips     : 4200.05
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor      : 6
vendor_id     : GenuineIntel
cpu family    : 6
model         : 62
model name    : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode     : 1045
cpu MHz       : 2100.025
cache size    : 15360 KB
physical id: 1
siblings      : 12
core id       : 0
cpu cores    : 6
apicid        : 32
initial apicid : 32
fpu           : yes
fpu_exception : yes
cpuid level: 13

```

```

wp                : yes
flags             : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips         : 4199.40
clflush size     : 64
cache_alignment  : 64
address sizes    : 46 bits physical, 48 bits virtual
power management:

```

```

processor       : 7
vendor_id      : GenuineIntel
cpu family     : 6
model          : 62
model name     : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping       : 4
microcode      : 1045
cpu MHz        : 2100.025
cache size     : 15360 KB
physical id    : 1
siblings       : 12
core id        : 1
cpu cores      : 6
apicid         : 34
initial apicid : 34
fpu            : yes
fpu_exception  : yes
cpuid level    : 13

```

```

wp                : yes
flags             : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips         : 4199.40
clflush size     : 64
cache_alignment  : 64
address sizes    : 46 bits physical, 48 bits virtual
power management:

```

```

processor       : 8
vendor_id      : GenuineIntel
cpu family     : 6
model          : 62
model name     : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping       : 4
microcode      : 1045
cpu MHz        : 2100.025
cache size     : 15360 KB
physical id    : 1
siblings       : 12
core id        : 2
cpu cores      : 6
apicid         : 36
initial apicid : 36
fpu            : yes
fpu_exception  : yes
cpuid level    : 13

```

```

wp                : yes
flags             : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips         : 4199.40
clflush size     : 64

```

cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 9
 vendor_id : GenuineIntel
 cpu family : 6
 model : 62
 model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
 stepping : 4
 microcode : 1045
 cpu MHz : 2100.025
 cache size : 15360 KB
 physical id : 1
 siblings : 12
 core id : 3
 cpu cores : 6
 apicid : 38
 initial apicid : 38
 fpu : yes
 fpu_exception : yes
 cpuid level: 13
 wp : yes
 flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
 sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
 pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
 tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
 fsgsbase smep erms
 bogomips : 4199.40
 clflush size : 64
 cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 10
 vendor_id : GenuineIntel
 cpu family : 6
 model : 62
 model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
 stepping : 4
 microcode : 1045
 cpu MHz : 2100.025
 cache size : 15360 KB
 physical id : 1
 siblings : 12
 core id : 4
 cpu cores : 6
 apicid : 40
 initial apicid : 40
 fpu : yes
 fpu_exception : yes
 cpuid level: 13
 wp : yes
 flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
 sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
 pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
 tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
 fsgsbase smep erms
 bogomips : 4199.40
 clflush size : 64
 cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 11
 vendor_id : GenuineIntel
 cpu family : 6
 model : 62

```

model name      : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping       : 4
microcode      : 1045
cpu MHz        : 2100.025
cache size     : 15360 KB
physical id    : 1
siblings       : 12
core id        : 5
cpu cores      : 6
apicid         : 42
initial apicid : 42
fpu            : yes
fpu_exception  : yes
cpuid level    : 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips      : 4199.40
clflush size   : 64
cache_alignment : 64
address sizes  : 46 bits physical, 48 bits virtual
power management:

```

```

processor      : 12
vendor_id     : GenuineIntel
cpu family    : 6
model         : 62
model name    : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode     : 1045
cpu MHz       : 2100.025
cache size    : 15360 KB
physical id   : 0
siblings      : 12
core id       : 0
cpu cores     : 6
apicid        : 1
initial apicid : 1
fpu           : yes
fpu_exception : yes
cpuid level   : 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips     : 4200.05
clflush size  : 64
cache_alignment : 64
address sizes  : 46 bits physical, 48 bits virtual
power management:

```

```

processor      : 13
vendor_id     : GenuineIntel
cpu family    : 6
model         : 62
model name    : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode     : 1045
cpu MHz       : 2100.025
cache size    : 15360 KB
physical id   : 0
siblings      : 12
core id       : 1

```

```

cpu cores : 6
apicid : 3
initial apicid : 3
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.05
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 14
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.025
cache size : 15360 KB
physical id: 0
siblings : 12
core id : 2
cpu cores : 6
apicid : 5
initial apicid : 5
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.05
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 15
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.025
cache size : 15360 KB
physical id: 0
siblings : 12
core id : 3
cpu cores : 6
apicid : 7
initial apicid : 7
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes

```



```

flags                : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.05
clflush size       : 64
cache_alignment    : 64
address sizes      : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 16
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.025
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 4
cpu cores  : 6
apicid     : 9
initial apicid : 9
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.05
clflush size       : 64
cache_alignment    : 64
address sizes      : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 17
vendor_id : GenuineIntel
cpu family: 6
model      : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping   : 4
microcode  : 1045
cpu MHz    : 2100.025
cache size : 15360 KB
physical id: 0
siblings   : 12
core id    : 5
cpu cores  : 6
apicid     : 11
initial apicid : 11
fpu        : yes
fpu_exception : yes
cpuid level: 13
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4200.05
clflush size       : 64
cache_alignment    : 64

```

address sizes : 46 bits physical, 48 bits virtual
power management:

```
processor : 18
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.025
cache size : 15360 KB
physical id: 1
siblings : 12
core id : 0
cpu cores : 6
apicid : 33
initial apicid : 33
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.40
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 19
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.025
cache size : 15360 KB
physical id: 1
siblings : 12
core id : 1
cpu cores : 6
apicid : 35
initial apicid : 35
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.40
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 20
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
```

```

stepping : 4
microcode : 1045
cpu MHz : 2100.025
cache size : 15360 KB
physical id : 1
siblings : 12
core id : 2
cpu cores : 6
apicid : 37
initial apicid : 37
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.40
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 21
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.025
cache size : 15360 KB
physical id : 1
siblings : 12
core id : 3
cpu cores : 6
apicid : 39
initial apicid : 39
fpu : yes
fpu_exception : yes
cpuid level: 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips : 4199.40
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 22
vendor_id : GenuineIntel
cpu family: 6
model : 62
model name : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping : 4
microcode : 1045
cpu MHz : 2100.025
cache size : 15360 KB
physical id : 1
siblings : 12
core id : 4
cpu cores : 6

```

```

apicid           : 41
initial apicid   : 41
fpu              : yes
fpu_exception    : yes
cpuid level: 13
wp              : yes
flags            : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips        : 4199.40
clflush size    : 64
cache_alignment : 64
address sizes    : 46 bits physical, 48 bits virtual
power management:

```

```

processor       : 23
vendor_id      : GenuineIntel
cpu family     : 6
model          : 62
model name     : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping      : 4
microcode     : 1045
cpu MHz        : 2100.025
cache size     : 15360 KB
physical id    : 1
siblings       : 12
core id        : 5
cpu cores      : 6
apicid         : 43
initial apicid : 43
fpu            : yes
fpu_exception  : yes
cpuid level: 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
fsgsbase smep erms
bogomips       : 4199.40
clflush size   : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

```

meminfo information

```

MemTotal:    65941420 kB
MemFree:     63100960 kB
Buffers:     113432 kB
Cached:      2016112 kB
SwapCached:    0 kB
Active:       586396 kB
Inactive:    1571768 kB
Active(anon): 28708 kB
Inactive(anon): 236 kB
Active(file): 557688 kB
Inactive(file): 1571532 kB
Unevictable:    0 kB
Mlocked:        0 kB
SwapTotal:    24313852 kB
SwapFree:     24313852 kB
Dirty:        24 kB
Writeback:     0 kB
AnonPages:    28688 kB

```

```

Mapped:      15456 kB
Shmem:       276 kB
Slab:        195996 kB
SReclaimable: 106328 kB
SUnreclaim:  89668 kB
KernelStack: 3520 kB
PageTables:  2380 kB
NFS_Unstable: 0 kB
Bounce:      0 kB
WritebackTmp: 0 kB
CommitLimit: 57284560 kB
Committed_AS: 251524 kB
VmallocTotal: 34359738367 kB
VmallocUsed:  395740 kB
VmallocChunk: 34323883592 kB
HardwareCorrupted: 0 kB
AnonHugePages: 10240 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB
DirectMap4k: 4096 kB
DirectMap2M: 2084864 kB
DirectMap1G: 65011712 kB

```

df (file system) information

```

Filesystem      1M-blocks  Used Available Use% Mounted on
rootfs          9952 1064   8377 12% /
/dev/mapper/docker-253:0-2622536-6cc5e67e947dd7d56290f40abaa8c9fa92c453b7871e284face8d0e8b33e38a1
9952 1064   8377 12% /
proc            0 0 0 - /proc
tmpfs           32198 0 32198 0% /dev
shm             64 0 64 0% /dev/shm
mqueue          0 0 0 - /dev/mqueue
devpts          0 0 0 - /dev/pts
sysfs           0 0 0 - /sys
/dev/mapper/vg_benchmarks-lv_root
50269 7047 40663 15% /etc/resolv.conf
/dev/mapper/vg_benchmarks-lv_root
50269 7047 40663 15% /etc/hostname
/dev/mapper/vg_benchmarks-lv_root
50269 7047 40663 15% /etc/hosts
/dev/mapper/vg_benchmarks-lv_root
50269 7047 40663 15% /root/CTBS
devpts          0 0 0 - /dev/console
proc            0 0 0 - /proc/sys
proc            0 0 0 - /proc/sysrq-trigger
proc            0 0 0 - /proc/irq
proc            0 0 0 - /proc/bus
tmpfs           32198 0 32198 0% /proc/kcore

```

disk file system information

```
Linux 2.6.32-504.8.1.el6.x86_64 (CTBS) 02/08/2015 _x86_64_ (24 CPU)
```

```

Device:      tps  Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda          6.29   46.10    294.94   818950   5239026
sdh          0.02    0.16     0.00    2888     0
sdi          0.02    0.16     0.00    2888     0
sdb          0.02    0.16     0.00    2888     0
sdc          0.02    0.16     0.00    2888     0
sdd          0.02    0.16     0.00    2888     0
sde          0.02    0.16     0.00    2888     0
sdf          0.02    0.16     0.00    2888     0
sdg          0.02    0.16     0.00    2888     0
dm-0         37.31   45.53    294.94   808826   5238984

```

dm-1	0.02	0.13	0.00	2376	0
dm-2	0.01	0.10	0.00	1762	24
dm-3	24.36	35.30	235.60	627122	4184984
dm-4	0.29	7.39	0.84	131266	14864

network file system information

Linux 2.6.32-504.8.1.el6.x86_64 (CTBS) 02/08/2015 _x86_64_ (24 CPU)

Filesystem: rBlk_nor/s wBlk_nor/s rBlk_dir/s wBlk_dir/s rBlk_svr/s wBlk_svr/s ops/s rops/s wops/s

C.2.2 Software Version Environment

gcc information:

gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-11)
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

System information:

Linux CTBS 2.6.32-504.8.1.el6.x86_64 #1 SMP Wed Jan 28 21:11:36 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux

OS information:

CentOS release 6.6 (Final)

C.3 KVM

C.3.1 Machine Environment

cpuinfo information

```
processor : 0
vendor_id : GenuineIntel
cpu family: 6
model      : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping   : 3
microcode  : 1
cpu MHz    : 2099.998
cache size : 4096 KB
physical id: 0
siblings   : 1
core id    : 0
cpu cores  : 1
apicid     : 0
initial apicid : 0
fpu        : yes
fpu_exception : yes
cpuid level: 4
wp         : yes
flags      : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips   : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 1
vendor_id : GenuineIntel
cpu family: 6
model      : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping   : 3
microcode  : 1
cpu MHz    : 2099.998
cache size : 4096 KB
physical id: 1
siblings   : 1
core id    : 0
cpu cores  : 1
apicid     : 1
initial apicid : 1
fpu        : yes
fpu_exception : yes
cpuid level: 4
wp         : yes
flags      : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips   : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 2
vendor_id : GenuineIntel
cpu family: 6
model      : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping   : 3
microcode  : 1
cpu MHz    : 2099.998
```



```

cache size : 4096 KB
physical id: 2
siblings   : 1
core id    : 0
cpu cores  : 1
apicid     : 2
initial apicid : 2
fpu        : yes
fpu_exception : yes
cpuid level: 4
wp         : yes
flags      : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips   : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

processor   : 3
vendor_id   : GenuineIntel
cpu family : 6
model       : 13
model name   : QEMU Virtual CPU version (cpu64-rhel6)
stepping    : 3
microcode   : 1
cpu MHz     : 2099.998
cache size  : 4096 KB
physical id: 3
siblings    : 1
core id     : 0
cpu cores   : 1
apicid      : 3
initial apicid : 3
fpu         : yes
fpu_exception : yes
cpuid level: 4
wp          : yes
flags       : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips    : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

processor   : 4
vendor_id   : GenuineIntel
cpu family : 6
model       : 13
model name   : QEMU Virtual CPU version (cpu64-rhel6)
stepping    : 3
microcode   : 1
cpu MHz     : 2099.998
cache size  : 4096 KB
physical id: 4
siblings    : 1
core id     : 0
cpu cores   : 1
apicid      : 4
initial apicid : 4
fpu         : yes
fpu_exception : yes
cpuid level: 4
wp          : yes
flags       : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips    : 4199.99

```

clflush size : 64
 cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 5
 vendor_id : GenuineIntel
 cpu family : 6
 model : 13
 model name : QEMU Virtual CPU version (cpu64-rhel6)
 stepping : 3
 microcode : 1
 cpu MHz : 2099.998
 cache size : 4096 KB
 physical id : 5
 siblings : 1
 core id : 0
 cpu cores : 1
 apicid : 5
 initial apicid : 5
 fpu : yes
 fpu_exception : yes
 cpuid level: 4
 wp : yes
 flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
 unfair_spinlock pni cx16 hypervisor lahf_lm
 bogomips : 4199.99
 clflush size : 64
 cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 6
 vendor_id : GenuineIntel
 cpu family : 6
 model : 13
 model name : QEMU Virtual CPU version (cpu64-rhel6)
 stepping : 3
 microcode : 1
 cpu MHz : 2099.998
 cache size : 4096 KB
 physical id : 6
 siblings : 1
 core id : 0
 cpu cores : 1
 apicid : 6
 initial apicid : 6
 fpu : yes
 fpu_exception : yes
 cpuid level: 4
 wp : yes
 flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
 unfair_spinlock pni cx16 hypervisor lahf_lm
 bogomips : 4199.99
 clflush size : 64
 cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 7
 vendor_id : GenuineIntel
 cpu family : 6
 model : 13
 model name : QEMU Virtual CPU version (cpu64-rhel6)
 stepping : 3
 microcode : 1
 cpu MHz : 2099.998
 cache size : 4096 KB

```

physical id : 7
siblings    : 1
core id     : 0
cpu cores   : 1
apicid      : 7
initial apicid : 7
fpu         : yes
fpu_exception : yes
cpuid level: 4
wp          : yes
flags       : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips    : 4199.99
clflush size      : 64
cache_alignment   : 64
address sizes     : 46 bits physical, 48 bits virtual
power management:

```

```

processor    : 8
vendor_id   : GenuineIntel
cpu family   : 6
model       : 13
model name   : QEMU Virtual CPU version (cpu64-rhel6)
stepping    : 3
microcode    : 1
cpu MHz      : 2099.998
cache size   : 4096 KB
physical id : 8
siblings    : 1
core id     : 0
cpu cores   : 1
apicid      : 8
initial apicid : 8
fpu         : yes
fpu_exception : yes
cpuid level: 4
wp          : yes
flags       : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips    : 4199.99
clflush size      : 64
cache_alignment   : 64
address sizes     : 46 bits physical, 48 bits virtual
power management:

```

```

processor    : 9
vendor_id   : GenuineIntel
cpu family   : 6
model       : 13
model name   : QEMU Virtual CPU version (cpu64-rhel6)
stepping    : 3
microcode    : 1
cpu MHz      : 2099.998
cache size   : 4096 KB
physical id : 9
siblings    : 1
core id     : 0
cpu cores   : 1
apicid      : 9
initial apicid : 9
fpu         : yes
fpu_exception : yes
cpuid level: 4
wp          : yes
flags       : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips    : 4199.99
clflush size      : 64

```

cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 10
 vendor_id : GenuineIntel
 cpu family : 6
 model : 13
 model name : QEMU Virtual CPU version (cpu64-rhel6)
 stepping : 3
 microcode : 1
 cpu MHz : 2099.998
 cache size : 4096 KB
 physical id : 10
 siblings : 1
 core id : 0
 cpu cores : 1
 apicid : 10
 initial apicid : 10
 fpu : yes
 fpu_exception : yes
 cpuid level : 4
 wp : yes
 flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
 unfair_spinlock pni cx16 hypervisor lahf_lm
 bogomips : 4199.99
 clflush size : 64
 cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 11
 vendor_id : GenuineIntel
 cpu family : 6
 model : 13
 model name : QEMU Virtual CPU version (cpu64-rhel6)
 stepping : 3
 microcode : 1
 cpu MHz : 2099.998
 cache size : 4096 KB
 physical id : 11
 siblings : 1
 core id : 0
 cpu cores : 1
 apicid : 11
 initial apicid : 11
 fpu : yes
 fpu_exception : yes
 cpuid level : 4
 wp : yes
 flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
 unfair_spinlock pni cx16 hypervisor lahf_lm
 bogomips : 4199.99
 clflush size : 64
 cache_alignment : 64
 address sizes : 46 bits physical, 48 bits virtual
 power management:

processor : 12
 vendor_id : GenuineIntel
 cpu family : 6
 model : 13
 model name : QEMU Virtual CPU version (cpu64-rhel6)
 stepping : 3
 microcode : 1
 cpu MHz : 2099.998
 cache size : 4096 KB
 physical id : 12

```
siblings      : 1
core id       : 0
cpu cores    : 1
apicid        : 12
initial apicid : 12
fpu           : yes
fpu_exception : yes
cpuid level: 4
wp            : yes
flags         : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips     : 4199.99
clflush size  : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 13
vendor_id : GenuineIntel
cpu family: 6
model      : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping   : 3
microcode  : 1
cpu MHz    : 2099.998
cache size : 4096 KB
physical id: 13
siblings   : 1
core id    : 0
cpu cores  : 1
apicid     : 13
initial apicid : 13
fpu        : yes
fpu_exception : yes
cpuid level: 4
wp         : yes
flags      : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips   : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 14
vendor_id : GenuineIntel
cpu family: 6
model      : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping   : 3
microcode  : 1
cpu MHz    : 2099.998
cache size : 4096 KB
physical id: 14
siblings   : 1
core id    : 0
cpu cores  : 1
apicid     : 14
initial apicid : 14
fpu        : yes
fpu_exception : yes
cpuid level: 4
wp         : yes
flags      : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips   : 4199.99
clflush size : 64
cache_alignment : 64
```

address sizes : 46 bits physical, 48 bits virtual
power management:

processor : 15
vendor_id : GenuineIntel
cpu family : 6
model : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping : 3
microcode : 1
cpu MHz : 2099.998
cache size : 4096 KB
physical id : 15
siblings : 1
core id : 0
cpu cores : 1
apicid : 15
initial apicid : 15
fpu : yes
fpu_exception : yes
cpuid level : 4
wp : yes
flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

processor : 16
vendor_id : GenuineIntel
cpu family : 6
model : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping : 3
microcode : 1
cpu MHz : 2099.998
cache size : 4096 KB
physical id : 16
siblings : 1
core id : 0
cpu cores : 1
apicid : 16
initial apicid : 16
fpu : yes
fpu_exception : yes
cpuid level : 4
wp : yes
flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

processor : 17
vendor_id : GenuineIntel
cpu family : 6
model : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping : 3
microcode : 1
cpu MHz : 2099.998
cache size : 4096 KB
physical id : 17
siblings : 1

```

core id          : 0
cpu cores       : 1
apicid          : 17
initial apicid   : 17
fpu             : yes
fpu_exception    : yes
cpuid level: 4
wp             : yes
flags           : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips        : 4199.99
clflush size    : 64
cache_alignment : 64
address sizes    : 46 bits physical, 48 bits virtual
power management:

processor        : 18
vendor_id       : GenuineIntel
cpu family      : 6
model           : 13
model name      : QEMU Virtual CPU version (cpu64-rhel6)
stepping        : 3
microcode       : 1
cpu MHz         : 2099.998
cache size      : 4096 KB
physical id: 18
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 18
initial apicid   : 18
fpu             : yes
fpu_exception    : yes
cpuid level: 4
wp             : yes
flags           : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips        : 4199.99
clflush size    : 64
cache_alignment : 64
address sizes    : 46 bits physical, 48 bits virtual
power management:

processor        : 19
vendor_id       : GenuineIntel
cpu family      : 6
model           : 13
model name      : QEMU Virtual CPU version (cpu64-rhel6)
stepping        : 3
microcode       : 1
cpu MHz         : 2099.998
cache size      : 4096 KB
physical id: 19
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 19
initial apicid   : 19
fpu             : yes
fpu_exception    : yes
cpuid level: 4
wp             : yes
flags           : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips        : 4199.99
clflush size    : 64
cache_alignment : 64
address sizes    : 46 bits physical, 48 bits virtual

```

power management:

```
processor : 20
vendor_id : GenuineIntel
cpu family : 6
model : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping : 3
microcode : 1
cpu MHz : 2099.998
cache size : 4096 KB
physical id : 20
siblings : 1
core id : 0
cpu cores : 1
apicid : 20
initial apicid : 20
fpu : yes
fpu_exception : yes
cpuid level: 4
wp : yes
flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 21
vendor_id : GenuineIntel
cpu family : 6
model : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping : 3
microcode : 1
cpu MHz : 2099.998
cache size : 4096 KB
physical id : 21
siblings : 1
core id : 0
cpu cores : 1
apicid : 21
initial apicid : 21
fpu : yes
fpu_exception : yes
cpuid level: 4
wp : yes
flags : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:
```

```
processor : 22
vendor_id : GenuineIntel
cpu family : 6
model : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping : 3
microcode : 1
cpu MHz : 2099.998
cache size : 4096 KB
physical id : 22
siblings : 1
core id : 0
```



```

cpu cores : 1
apicid      : 22
initial apicid : 22
fpu         : yes
fpu_exception : yes
cpuid level: 4
wp          : yes
flags       : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

```

processor : 23
vendor_id : GenuineIntel
cpu family: 6
model      : 13
model name : QEMU Virtual CPU version (cpu64-rhel6)
stepping   : 3
microcode  : 1
cpu MHz    : 2099.998
cache size : 4096 KB
physical id: 23
siblings   : 1
core id    : 0
cpu cores : 1
apicid     : 23
initial apicid : 23
fpu        : yes
fpu_exception : yes
cpuid level: 4
wp         : yes
flags      : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm
unfair_spinlock pni cx16 hypervisor lahf_lm
bogomips : 4199.99
clflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

meminfo information

```

MemTotal: 64816924 kB
MemFree: 62859704 kB
Buffers: 39472 kB
Cached: 1319656 kB
SwapCached: 0 kB
Active: 831868 kB
Inactive: 537020 kB
Active(anon): 9780 kB
Inactive(anon): 208 kB
Active(file): 822088 kB
Inactive(file): 536812 kB
Unevictable: 0 kB
Mlocked: 0 kB
SwapTotal: 5242876 kB
SwapFree: 5242876 kB
Dirty: 0 kB
Writeback: 96 kB
AnonPages: 10012 kB
Mapped: 9264 kB
Shmem: 220 kB
Slab: 186836 kB
SReclaimable: 120564 kB
SUnreclaim: 66272 kB

```

```

KernelStack:    3184 kB
PageTables:     1668 kB
NFS_Unstable:   0 kB
Bounce:         0 kB
WritebackTmp:   0 kB
CommitLimit:    37651336 kB
Committed_AS:   62432 kB
VmallocTotal:   34359738367 kB
VmallocUsed:    118988 kB
VmallocChunk:   34359609612 kB
HardwareCorrupted: 0 kB
AnonHugePages:  0 kB
HugePages_Total: 0
HugePages_Free:  0
HugePages_Rsvd:  0
HugePages_Surp:  0
Hugepagesize:   2048 kB
DirectMap4k:    9200 kB
DirectMap2M:    65931264 kB

```

df (file system) information

Filesystem	1M-blocks	Used	Available	Use%	Mounted on
/dev/mapper/vg_kvm-lv_root	44733	1630	40825	4%	/
proc	0	0	0	-	/proc
sysfs	0	0	0	-	/sys
devpts	0	0	0	-	/dev/pts
tmpfs	31649	0	31649	0%	/dev/shm
/dev/vda1	477	48	405	11%	/boot
none	0	0	0	-	/proc/sys/fs/binfmt_misc

disk file system information

```
Linux 2.6.32-504.el6.x86_64 (KVM.localdomain)    02/12/2015    _x86_64_ (24 CPU)
```

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
vda	70.16	2597.33	3416.93	11942246	15710668
dm-0	176.60	2595.28	3406.95	11932826	15664792
dm-1	0.07	0.53	0.00	2456	0

network file system information

```
Linux 2.6.32-504.el6.x86_64 (KVM.localdomain)    02/12/2015    _x86_64_ (24 CPU)
```

Filesystem:	rBlk_nor/s	wBlk_nor/s	rBlk_dir/s	wBlk_dir/s	rBlk_svr/s	wBlk_svr/s	ops/s	rops/s	wops/s
-------------	------------	------------	------------	------------	------------	------------	-------	--------	--------

C.3.2 Software Version Environment

gcc information:

gcc (GCC) 4.4.7 20120313 (Red Hat 4.4.7-11)
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

System information:

Linux KVM.localdomain 2.6.32-504.el6.x86_64 #1 SMP Wed Oct 15 04:27:16 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux

OS information:

CentOS release 6.6 (Final)