AMPERE®

Hadoop on Ampere® Arm Processors

Reference Architecture

October 24, 2023

Document Issue 0.75

AMPERE.

Contents

AMPERE.

Contents (continued)

1. Overview

The influence of Arm technology has expanded beyond mobile and even desktop computing. On-premises and cloud servers now use Arm processors for a variety of workloads. These processors offer improved performance per rack, reduced power consumption, and lower costs, leading to capital expenditure (Capex) and operational expenditure (Opex) optimization. Arm servers are widely deployed in datacenters, and in cloud and edge computing environments. This paper contrasts Big Data Hadoop performance on clusters of Ampere® Altra® Max and Intel Ice Lake servers.

1.1 Scope and Audience

The scope of this document includes setting up, tuning and evaluating the performance of Hadoop on a test bed with Ampere Altra Max processors. This document covers the architectural concepts of Arm and Big Data Hadoop, then steps to installing and tuning Hadoop on an Altra Max multi-node cluster. However, note that these are general guidelines for setting up the cluster and the values and parameters are in no way to be considered as final and optimized values.

This document is intended for a diverse audience, including sales engineers, IT and cloud architects, IT and cloud managers, and customers seeking to leverage the performance and power efficiency benefits of Ampere Arm servers in their datacenters. It aims to provide valuable insights and technical guidance to these professionals who are interested in implementing Arm-based Hadoop solutions and optimizing their infrastructure.

1.2 Ampere Altra Max Processors

Ampere server processors offer comprehensive System-on-Chip (SoC) solutions designed specifically for Cloud Native applications. Ampere Altra Max processors provide powerful features to meet the demands of modern datacenters and increase performance per rack in modern datacenters. This processor supports up to 128 high performance Arm 64-bit cores. Additionally, this processor features eight channels of DDR4 memory for efficient data processing and storage. Altra Max processors support an impressive 128 channels of PCIe Gen 4 interfaces, enabling fast, reliable data transfer between various components and peripherals. Overall, Ampere server processors deliver a robust and scalable solution for a wide range of cloud-native applications deployed anywhere from hyperscale cloud environments to dense edge cloud setups.

Ampere Altra Max processors excel in handling intensive tasks such as data analytics, artificial intelligence (AI), database storage, telecommunications, edge computing, and web hosting. Leveraging the performance and capabilities of Ampere Altra processors enables datacenter operators to enhance the efficiency of their infrastructure and aggressively meet the performance demands of today's cloud-based applications and services.

Ampere server processors are specifically engineered to deliver exceptional energy efficiency, resulting in industry-leading performance per watt (Perf/Watt) at the individual processor level. This means that each server equipped with Ampere Altra Max processors delivers high performance while consuming less power than a comparable legacy server. Additionally, when scaled out, Ampere server processors exhibit outstanding performance per rack (Perf/Rack), maximizing energy efficiency and reducing overall power consumption.

Optimizing Perf/Rack not only reduces datacenter operating costs but also contributes to significant reductions in carbon footprint. Ampere Altra Max processors align datacenter operations with sustainability goals and reduced environmental impact. Overall, Ampere Altra Max processors provide a compelling solution for datacenters aiming to achieve both high performance and energy efficiency, leading to cost savings, improved sustainability, and a greener approach to computing.

2. Big Data Architecture

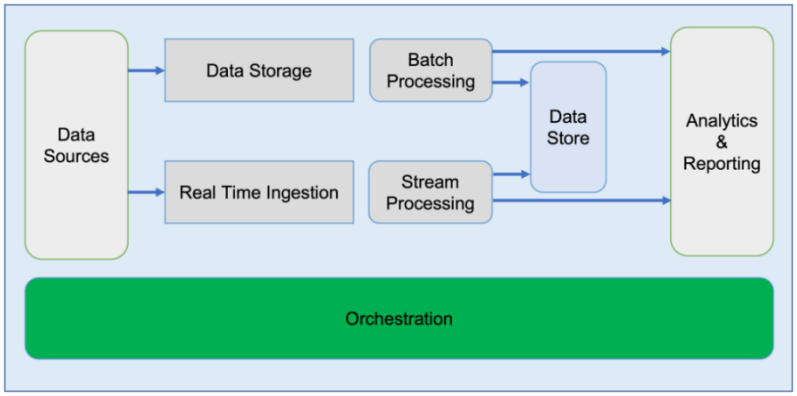
Big Data describes structured, semi-structured, and unstructured data that is mined by using advanced analytics.

Big Data deployments involve terabytes and petabytes of data, which is created and collected overtime. Big data domain encompasses ingestion, processing, and analysis of data sets that are too large to be handled by traditional systems.

Big Data solutions involve workloads such as:

- Batch processing of Big Data sources at rest
- Real-time processing of Big Data in motion
- Interactive exploration of Big Data
- Predictive analytics and machine learning

2.1 Components



A Big Data solution involves various components, and processes to handle data from various sources and enable efficient analysis. Key components involved in a Big Data solution include:

- Data sources
  Data can be obtained from a variety of sources, including application data stores such as databases (for example, DBMS systems), static files (for example, application log files), and real-time data sources (for example, IoT devices). Such sources provide raw data for further processing and analysis.
- Data storage
  The cost of storage has decreased significantly, leading to the use of faster storage devices like NVMEs. The data is typically stored in a distributed file store known as a Data Lake. A data lake supports the storage of large volumes of data in raw format, providing flexibility for future analysis and processing.
- Batch processing
  Batch processing involves filtering and aggregating the data for analysis. Frameworks like Apache Hadoop, MapR, Spark, and Hive are commonly used for batch processing. These frameworks use programming languages such as Java, Scala, and Python to process source files and generate output in new files. Batch processing is suitable for scenarios for which real-time analysis is not required.
- Real Time processing
  In real-time processing, the system captures and stores real-time messages for immediate or near-immediate processing. To facilitate message ingestion and act as a buffer for messages, streaming frameworks like Kafka are often employed. Kafka enables scalable and distributed message handling. After real-time messages are captured, they can be processed using frameworks like Spark, which handle real-time data streams efficiently.
- Data store and reporting
  After processing, data is typically stored in a structured format that facilitates querying and analysis. NoSQL databases like HBase and tools like SparkSQL are commonly used for data storage and reporting. These tools can query and analyze processed data, enabling users to derive insights and generate reports based on the data analysis.

2.2 Hadoop Ecosystem

The Apache Hadoop software library is a powerful framework that facilitates the distributed processing of massive data sets across clusters of computers. Hadoop employs straightforward programming models, making it accessible and user-friendly. Hadoop is designed to seamlessly scale from single servers to thousands of machines, with each machine capable of providing local computation, storage, or both.

Built-in resiliency is a notable strength of Hadoop, which can effectively handle various failures in a cluster ranging from individual drive failures to complete server failures. Instead of relying solely on hardware redundancy, Hadoop employs resilient techniques that ensure high availability and fault tolerance. This approach enables data processing and storage to continue uninterrupted even when hardware fail.

By leveraging distributed computing and resilient data management, Hadoop enables organizations to efficiently process and analyze large data sets. Hadoop supports a wide range of applications and use cases, including data-intensive tasks such as data mining, machine learning, and large-scale data analytics. Hadoop can distribute computation and storage across a cluster of machines to provide the scalability and performance required to handle complex data processing tasks.

The main elements of the Hadoop ecosystem include:

- Hadoop Distributed File System (HDFS)
- MapReduce
- Yet Another Resource Negotiator (YARN)
- Hadoop Common

2.2.1 Hadoop Distributed File System (HDFS)

HDFS is a distributed file system designed to store and manage large volumes of data across multiple machines. It provides fault tolerance, high scalability, and data redundancy. HDFS divides files into blocks and distributes them across a cluster of machines, enabling parallel processing and efficient data storage.

2.2.2 MapReduce

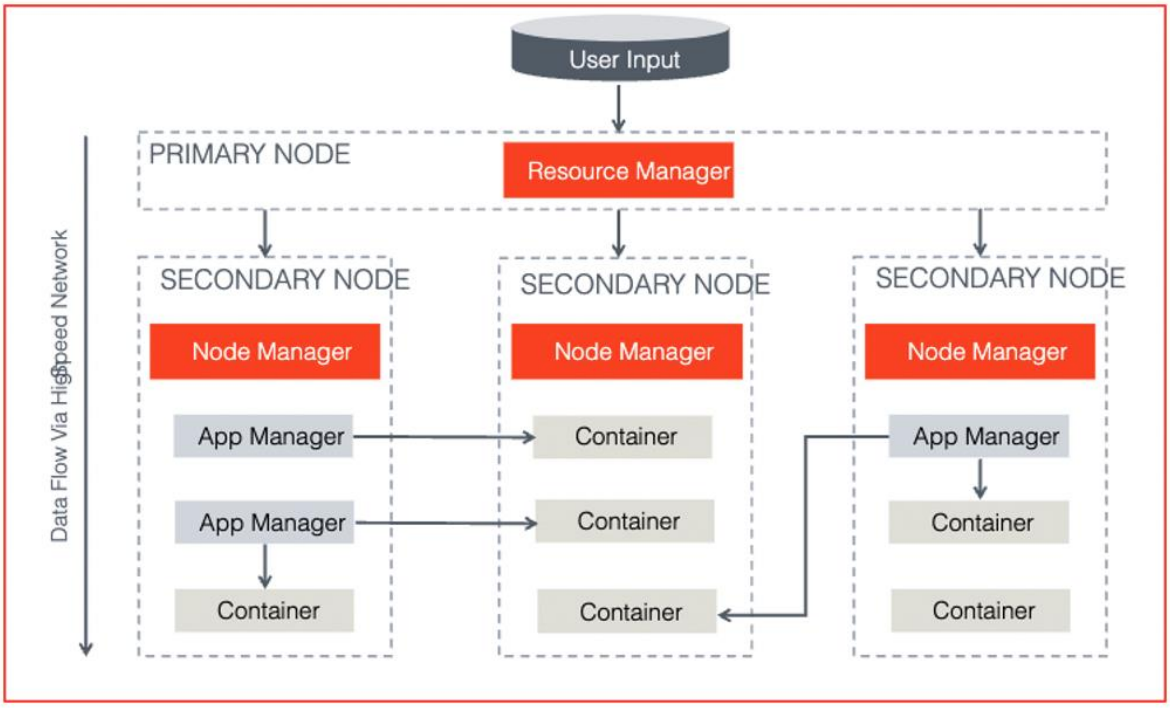
MapReduce is a programming model and processing framework used for distributed data processing in Hadoop. To support parallel processing, MapReduce splits processing tasks into smaller subtasks and distributing the subtask across a cluster. The Map phase processes data in parallel, and the Reduce phase aggregates and summarizes the results. MapReduce is widely used for batch processing and large-scale data analysis.

2.2.3 Yet Another Resource Negotiator (YARN)

YARN is a cluster management system that acts as a resource manager in Hadoop. It oversees resource allocation and scheduling of tasks across the cluster, managing resources efficiently. YARN allows different data processing frameworks, such as MapReduce, Apache Spark, and Apache Flink, to run simultaneously on the same Hadoop cluster, enabling diverse workloads to coexist.

2.2.4 Hadoop Common

Hadoop Common, which provides essential libraries and utilities that support the other components of the Hadoop ecosystem, includes common software modules and utilities required for Hadoop functioning. These include authentication, security, and file system interfaces. Hadoop Common is a foundation for other Hadoop components and ensures compatibility across the ecosystem.



2.3 Hadoop Test Bed

Two clusters were set up for performance benchmarking. The first cluster was equipped with HPE RL300 servers powered by Ampere Altra Max processors, while the second cluster used Dell PowerEdge R650 servers powered by Intel Ice Lake processors.

2.3.1 Equipment Under Test

ARCHITECTURE	AMPERE ALTRA MAX	INTEL ICE LAKE
Make & Model	HPE RL300	Dell PowerEdge R650
Cluster Nodes	3	3
CPU	Ampere M128-30	Intel Xeon SP 6342
Sockets/Node	1	2
Cores/Socket	128	24
Threads/Socket	128	48
CPU Speed	3.0 GHz	2.8 GHz/3.50 GHz
Memory	512 GB	512 GB
Network Card	1 x Mellanox CX-6 Dx	1 x Mellanox CX-6 Dx
Storage	4 x Micron 7450 Gen 4 NVME	4 x ScaleFlux CSD 3010 Gen 4 NVME
Kernel Version	4.18.0-348.7.1	5.15.0-60-generic
Operating System	CentOS 8.5	Ubuntu 22.04 LTS
Hadoop Version	3.3.4	3.3.4
JDK Version	JDK11	JDK11

3. Hadoop Installation and Cluster Setup

3.1 Configure Unified Extensible Firmware Interface (UEFI)

Before installing Hadoop, see the *Appendix* for UEFI settings for Ampere servers. Boot the servers to UEFI and make required changes.

3.2 Install the OS

Most modern open source or enterprise Linux distributions support AArch64. You can choose any OS supported on Arm. Map your CD/DVD in the Keyboard, Video, and Mouse (KVM) console and install the supported OS on the servers.

3.2.1 Set Up Networking

Set up a public network on an available interface. This interface can be used to log into any server in the cluster and where client communication is needed. Set up a private network for communication between cluster nodes.

3.2.2 Set Up Storage

Choose a drive for OS install. Clear any old partitions, reformat the drive, and install the OS. Samsung 960 GB drives are used for the OS install in this setup example.

3.2.3 Create the Hadoop User

Create a user named 'hadoop' as part of the OS Install.

3.2.4 After the OS Install

After the OS install, perform these steps for all nodes in the cluster:

1. Run yum or apt update on the nodes.
2. For monitoring, install packages such as dstat, net-tools, lm-sensors, linux-tools-generic, python, and sysstat.
3. Set up ssh trust among all nodes.
4. Update the **/etc/sudoers** file for nopasswd for the hadoop user.
5. Update **/etc/security/limits.conf**, using the *Appendix* as an example.
6. Update **/etc/sysctl.conf**, using the *Appendix* as an example.
7. Update scaling governor and hugepages, using the *Appendix* as an example.
8. If necessary, change **/etc/rc.d** to make the preceding changes persistent across reboots.
9. Setup NVMe disks as an xfs file system for HDFS.
   a. Create a single partition on each NVMe disk with fdisk or parted.
   b. Create a file system on each of the created partitions as `mkfs.xfs -f /dev/nvme[0-n]n1p1`.
   c. Create directories to mount on root.
   d. `mkdir -p /root/nvme[0-n]1p1`.
   e. Update **/etc/fstab** with entries mount the file system. The `blkid` command can extract the Universally Unique Identifier (UUID) of each partition for update in fstab.
   f. Change ownership of these directories to the 'hadoop' user.

3.3 Install Hadoop

Download Hadoop 3.3.4 from the Apache web site along with JDK11 for Arm and AArch64. Extract the tarballs in the Hadoop home directory.

Update Hadoop configuration files in **~/hadoop/etc/hadoop/** and environment parameters in .bashrc as described in the *Appendix*. Depending on the hardware specifications on cores, memory and disk capacities, these may have to be altered. Update the workers file to include the set of data nodes.

Run these commands:

```
hdfs namdenode -format
scp -r ~/hadoop <datanodes>:~/hadoop
~/hadoop/sbin/start-all.sh
```

This should start the NodeManager, ResourceManager, NameNode and DataNode processes on the nodes. Note that NameNode and Resource Managers start only on the primary node.

3.4 Verify the Installation

1. Run the `jps` command on each node to check the status of the Hadoop daemons.
2. Verify that `-ls`, `-put`, `-du`, and `-mkdir` commands can run on the cluster (`hdfs dfs <arg>`)

4. Performance Tuning

Several components interact across multiple systems in the Hadoop framework. Factors such as UEFI settings, operating system parameters, network and disk subsystems, and Hadoop stack configuration influence Hadoop performance. To optimize configuration settings, experience with Hadoop is helpful. Note that performance tuning is an iterative process. Parameters listed in the *Appendix* provide reference values obtained after a few iterations.

4.1 UEFI

When it comes to tuning systems for Hadoop, UEFI is a good starting point. Note that parameter names and options may differ slightly depending on the system manufacturer. The *Appendix* provides some UEFI tuning parameters that help optimize Hadoop performance.

4.2 Linux

Occasionally, conflicts occur between subcomponents in a Linux system, such as the network and disk, that impact overall performance. The objective is to optimize the system to achieve optimal disk and network throughput and identify and resolve bottlenecks that may arise.

4.3 Network

To evaluate the network infrastructure, the iperf utility can conduct stress tests. To help optimize performance, adjust the TX/RX ring buffers and the number of interrupt queues to align with the cores in the NUMA node where the NIC is located. Note that if the UEFI setting is already configured as chipset-ANC in a NUMA monolithic mode, these modifications may not be necessary.

4.4 Disks

When working in a Hadoop environment, pay attention to aspects such as aligned partitions, queue depth, number of requests, and filesystem mount options. Aligned partitions can be created using the parted utility. Queue depth and nr_requests can be configured through the /sys/block/<device>/device|queue directory. For Hadoop, the noatime file system option (fstab) is crucial. To test the disk subsystem, the fio tool can be used.

4.5 HDFS, YARN and MapReduce

In HDFS, the primary parameters to consider are the block size and replication factor. By default, the block size is set to 128 MB. Files in HDFS are divided into multiple chunks, each corresponding to the block size, and these chunks can be stored on different data nodes.

In the test bed uses a larger block size of 512 MB. With a replication factor of three, data is written once by applications but replicated to different data nodes based on this factor. As a result, three copies of the data are always available, ensuring high availability. The required storage space is directly proportional to the replication factor.

HDFS 3.x introduced a new feature called Erasure Coding (EC) that significantly reduces the storage required for the cluster. For example, a 6-3 EC configuration provides redundancy like a replication factor of three but has less storage overhead. However, note that EC puts an additional load on the network. The test bed used a replication factor of three.

YARN, the resource management framework in Hadoop, offers two scheduler options: Fair scheduler and Capacity scheduler. The Fair scheduler (the default) distributes resources evenly among jobs. The Capacity scheduler ensures that each user or job is allocated a fixed capacity, leaving any excess capacity unused. The key resource settings include minimum, maximum, and stepping allocation of memory and virtual core (vcore) resources.

In MapReduce, a job is divided into numerous tasks; each task has a smaller memory footprint and uses one core. Container allocations are based on these task footprints, considering the total memory available to the YARN node manager and the number of vcores. These settings can be directly modified in the yarn-site.xml file. The *Appendix* contains a few parameters used as examples in the test bed.

5. Benchmarking Tools

We used the HiBench benchmarking tool. HiBench is a popular benchmarking suite specifically designed for evaluating the performance of Big Data frameworks, such as Apache Hadoop and Apache Spark. The suite comprises benchmarks that simulate real-world Big Data processing workloads. Refer to this link for more information.

After running HiBench on both clusters, you can assess and compare their performance in handling various Big Data workloads. The benchmark results can provide insights into factors such as data processing speed, scalability, and resource utilization for each cluster.

The benchmarking exercise conducted using the HiBench tool enables comparative analyses of the performance characteristics of the HPE RL300 three-node cluster with Ampere Altra Max processors and the Dell PowerEdge R650 three-node cluster with Intel Ice Lake processors.

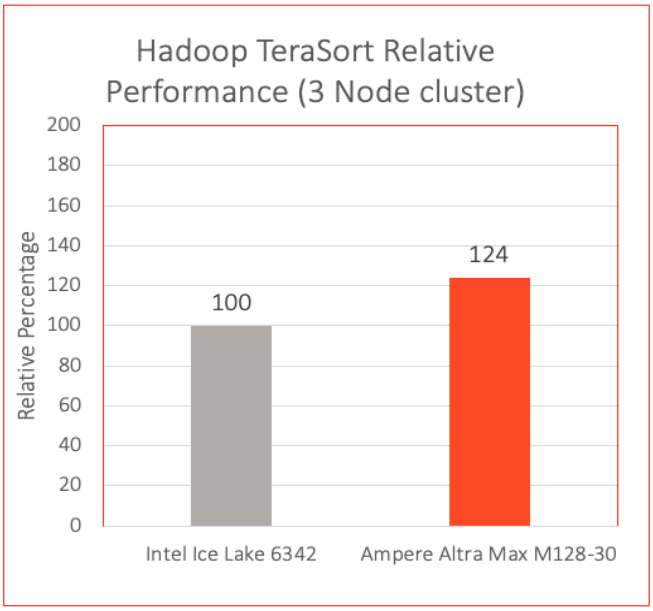
1. Update the hibench.conf file, such as the scale, profile, parallelism parameters and list of master and slave nodes.
2. Run **~HiBench/bin/workloads/micro/terasort/prepare/prepare.sh**
3. Run **~HiBench/bin/workloads/micro/terasort/Hadoop/run.sh**

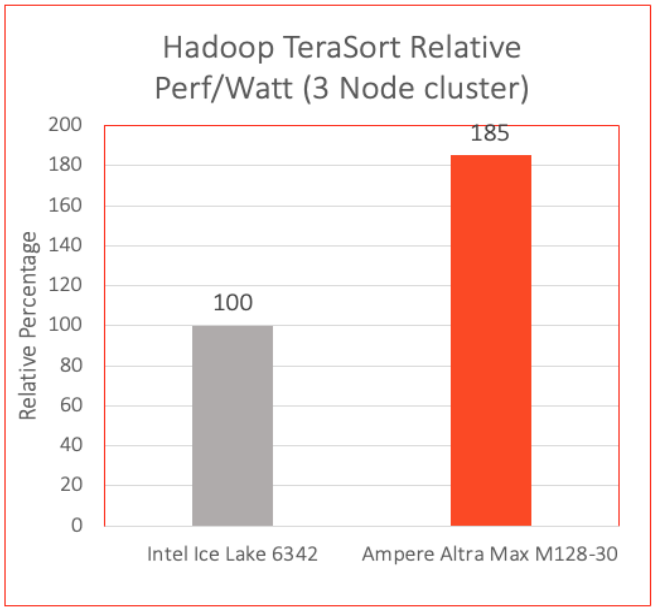
These steps generate a hibench.report file in the report directory. The bench.log file provides details of the run.

Throughput from the cluster is captured using HiBench for TeraSort and Wordcount benchmarks on these clusters, each using a 3 TB data set. We measure the total power consumed, CPU power, CPU utilization, and other parameters such as disk and network utilization using Grafana, IPMI and Redfish tools.

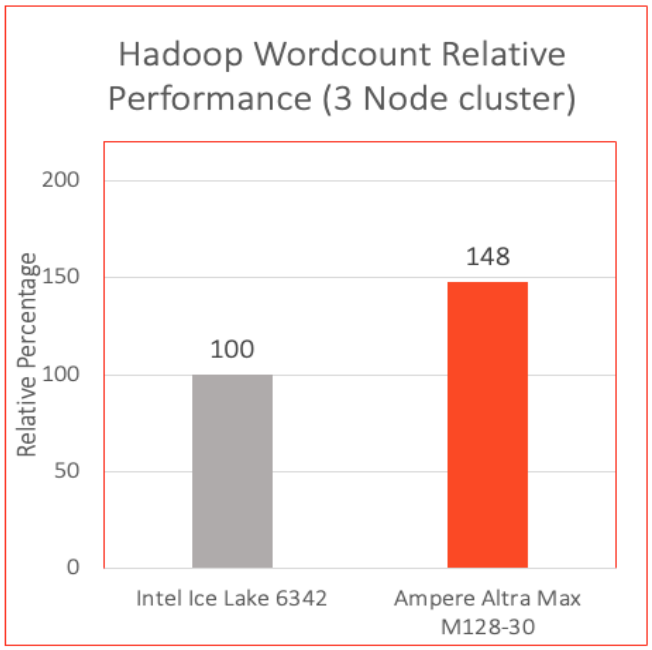
6. Performance Tests on Three-Node Clusters

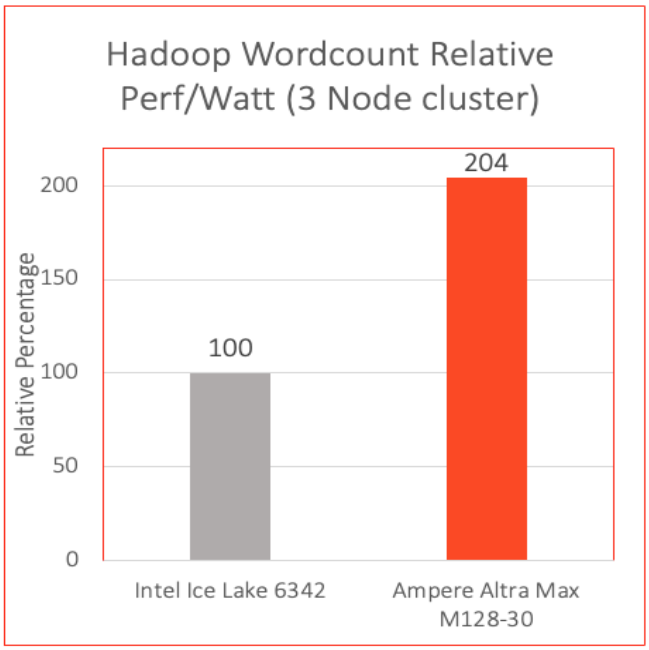
6.1 TeraSort Performance





6.2 Wordcount Performance



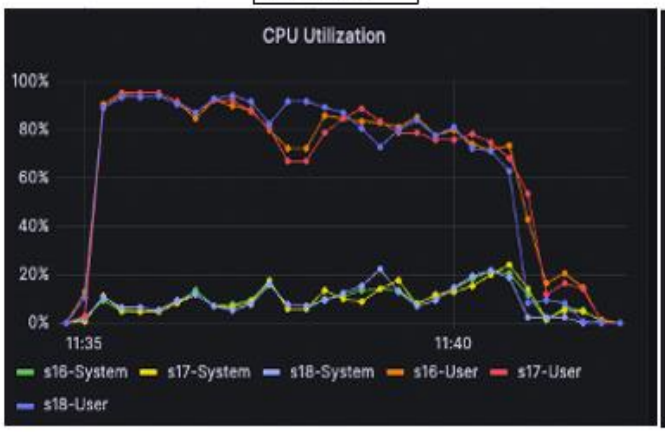


While benchmarking of the clusters, we find that the Ampere Altra Max systems delivered higher performance in both TeraSort and Wordcount benchmarks. Specifically, the TeraSort raw throughput was 24% higher, and the Wordcount raw throughput was 48% higher compared to the Intel Ice Lake systems.

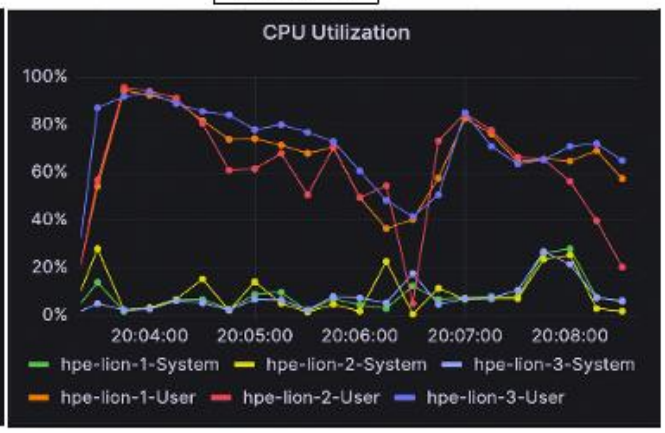
To assess the energy efficiency of each cluster, we calculate the Perf/Watt ratio by dividing the cluster throughput (in MB/s) by the power consumed by the cluster (in watts) during the benchmarking interval. The Ampere Altra Max cluster outperformed the Intel Ice Lake cluster in Perf/Watt, with an improvement of 85% in TeraSort and 200% in Wordcount.

6.3 CPU Utilization

Ice Lake



Altra Max

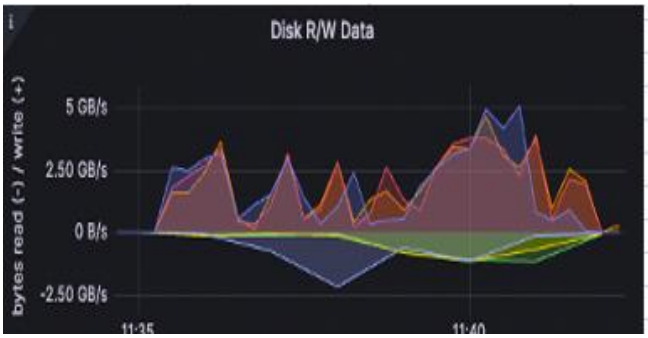


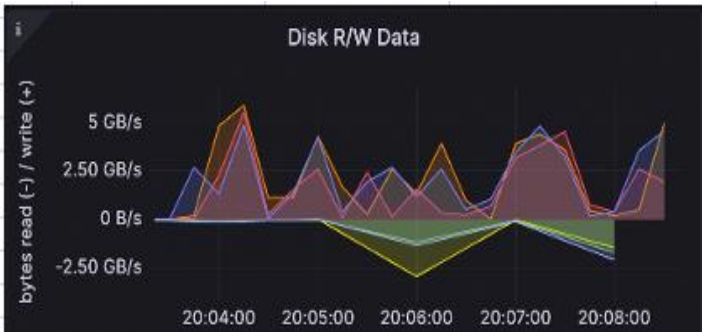
Based on the Grafana graphs presented, both systems are pushed to maximum CPU utilization while optimizing the TeraSort output.

6.4 Disk and Network Utilization

Ice Lake

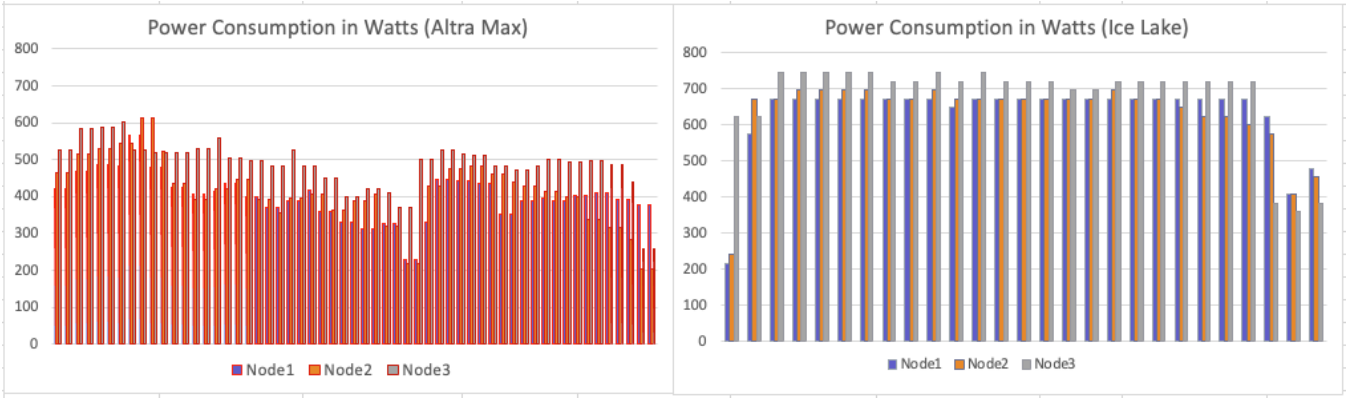
Altra Max





Data indicates that both the HPE RL300 and Dell PowerEdge R650 servers achieved approximately 5 GB/s disk throughput and 10 Gb/s network throughput.

6.5 Power Consumption



The preceding graphs illustrate the power consumption of the clusters. It is evident that the HPE RL300 servers, equipped with Ampere Altra Max processors, consumed significantly less power compared to the Dell PowerEdge servers featuring Intel Ice Lake processors.

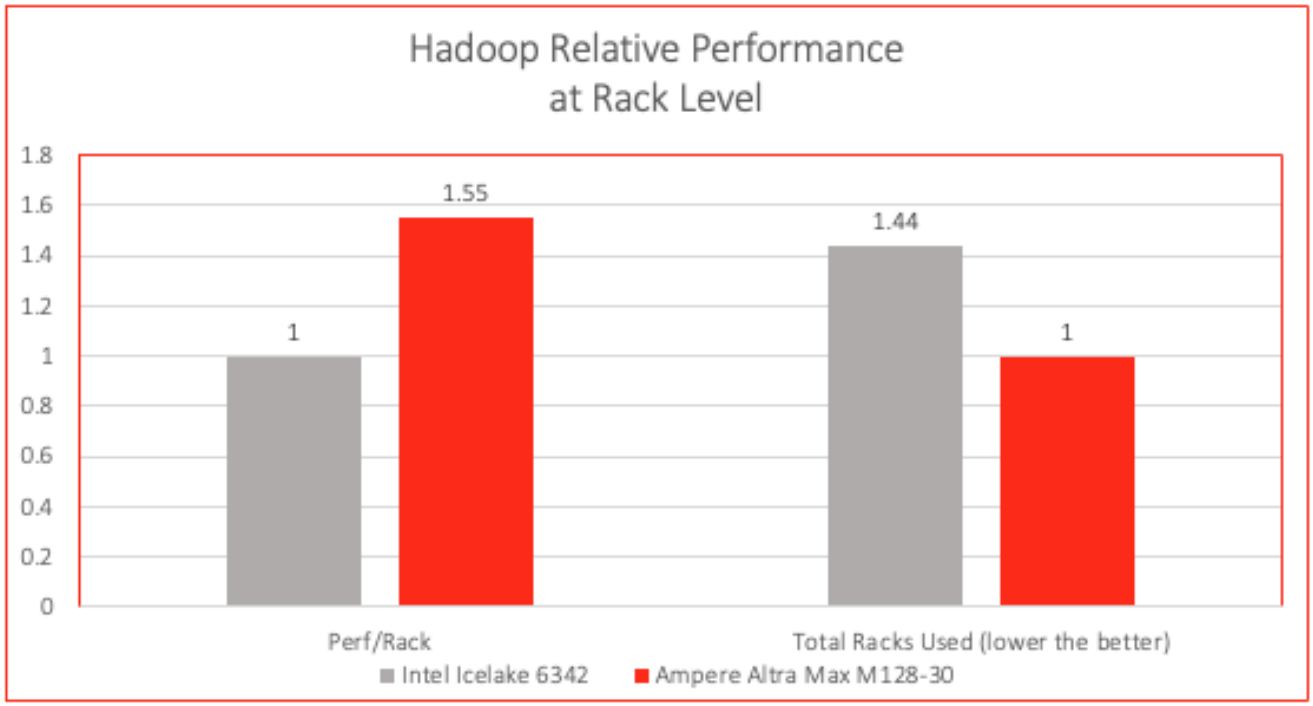
6.6 Rack and Datacenter Level Efficiency

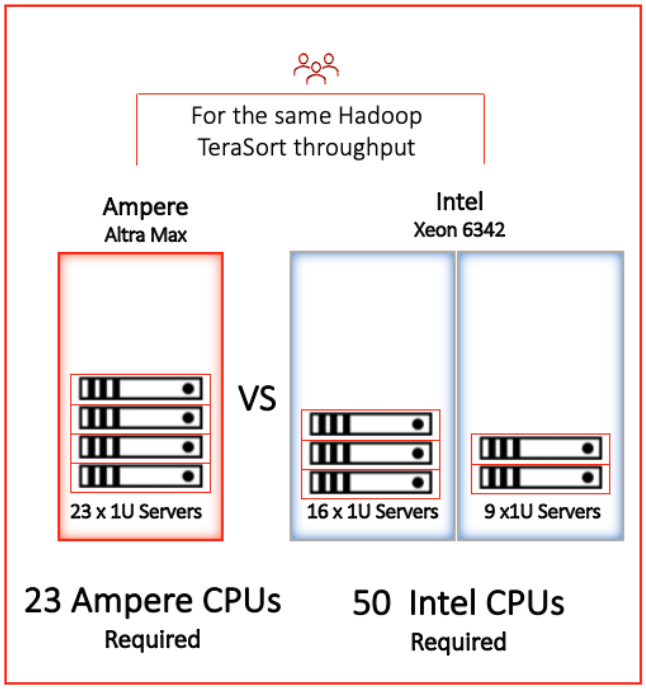
Dealing with large-scale Big Data processing using services like Hadoop often requires many servers, depending upon data set size. The infrastructure must be both scalable and sustainable.

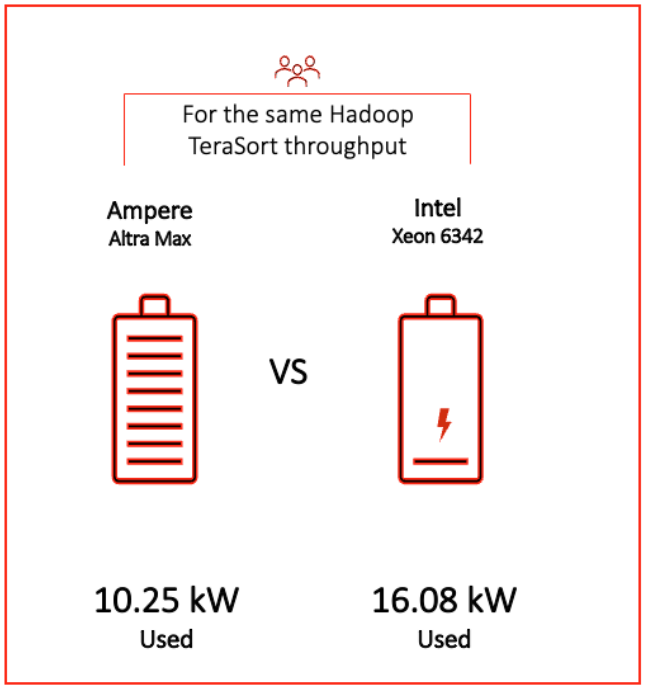
To assess the performance efficiency at the rack level, we extrapolate our three node cluster data to rack level (42U rack with a 12 KW power budget providing space for network equipment and other components). Ampere Altra Max CPUs not only provide superior performance but also consume less power, resulting in a smaller number of required Ampere racks compared to x86 racks to achieve the same level of performance.

To measure performance at the rack level, we divide the TeraSort throughput by the total power consumed to calculate the Perf/Rack metric. Our findings demonstrate that Ampere Altra Max servers outperform Intel Ice Lake servers by 55% in the Perf/Rack metric.

We can accommodate more Ampere Altra Max servers within the power limits of a rack. While a rack running a Hadoop workload could accommodate a maximum of 16 Intel servers before reaching the power limit, we can fit 23 Ampere Altra Max servers, resulting in a 44% improvement in rack-level density with HPE RL300 servers equipped with Ampere Altra Max processors.







Leveraging 23 Ampere Altra Max processors enables performance levels equivalent to those of 50 Intel Ice Lake processors. This performance equivalence results in substantial power savings of 57% when using Altra Max servers for Hadoop workloads compared to Intel servers.

The Ampere architecture is significantly more sustainable, boasts industry-leading performance per watt and performance per rack, and can reduce the overall resource footprint of Hadoop deployments by over 50%. This advantage is truly disruptive, making the Ampere Altra Family the most sustainable processor for your on-premises and cloud deployments.

7. Conclusions

This document describes a reference solution architecture for deploying Hadoop on a multi-node cluster, powered by Ampere Altra Max processors. It contrasts this solution to a similar setup based on Intel Ice Lake processors. The objective is to achieve maximum scalability at the rack level. Ampere Altra Max processors offer outstanding power efficiency, linear scalability, and high performance.

Big Data solutions demand substantial computational power and persistent storage. By running these applications on Ampere Altra Max processors, the benefits of both scale-up and scale-out architectures are harnessed. This approach enables a densely packed core configuration per rack, resulting in reduced power consumption per rack while maintaining the same level of throughput.

Appendix

UEFI changes

```
System Configuration -> BIOS Platform Configuration -> Processor Options -> ANC Mode Monilithic

System Configuration -> Power and Performance -> Ampere Max Performance -> Enabled
```

/etc/sysctl.conf

```
kernel.pid_max = 4194303
fs.aio-max-nr = 1048576
net.ipv4.conf.default.rp_filter=1
net.ipv4.tcp_timestamps=0
net.ipv4.tcp_sack = 1
net.core.netdev_max_backlog = 25000
net.core.rmem_max = 2147483647
net.core.wmem_max = 2147483647
net.core.rmem_default = 33554431
net.core.wmem_default = 33554432
net.core.optmem_max = 40960
net.ipv4.tcp_rmem =8192 33554432 2147483647
net.ipv4.tcp_wmem =8192 33554432 2147483647
net.ipv4.tcp_low_latency=1
net.ipv4.tcp_adv_win_scale=1
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv4.conf.all.arp_filter=1
net.ipv4.tcp_retries2=5
net.ipv6.conf.lo.disable_ipv6 = 1
net.core.somaxconn = 65535
#memory cache settings
vm.swappiness=1
vm.overcommit_memory=0
vm.dirty_background_ratio=1
```

/etc/security/limits.conf

```
*           soft    nofile          65536
*           hard    nofile          65536
*           soft    nproc           65536
*           hard    nproc           65536
```

Miscellaneous Kernel changes

```
#Disable Transparent Huge Page defrag
echo never> /sys/kernel/mm/transparent_hugepage/defrag
echo never > /sys/kernel/mm/transparent_hugepage/enabled

ifconfig enp1s0 mtu 9000
ifconfig enP1p1s0 mtu 9000
```

bashrc file

```
export JAVA_HOME=/home/hadoop/jdk
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=.:$JAVA_HOME/lib:$JRE_HOME/lib:$classpath
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

#HADOOP_HOME
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
export PATH=$PATH:/home/hadoop/.local/bin
```

core-site.xml

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://<server1>:9000</value>
        </property>

        <property>
                <name>hadoop.tmp.dir</name>
       <value>/data/data1/hadoop, /data/data2/hadoop, /data/data3/hadoop, /data/data4/hadoop
</value>
        </property>

        <property>
                <name>io.native.lib.available</name>
                <value>true</value>
        </property>
```

```
        <property>
                <name>io.compression.codecs</name>
<value>org.apache.hadoop.io.compress.GzipCodec, org.apache.hadoop.io.compress.DefaultCodec,
org.apache.hadoop.io.compress.BZip2Codec, com.hadoop.compression.lzo.LzoCodec,
com.hadoop.compression.lzo.LzopCodec, org.apache.hadoop.io.compress.SnappyCodec</value>
        </property>

        <property>
                <name>io.compression.codec.snappy.class</name>
                <value>org.apache.hadoop.io.compress.SnappyCodec</value>
        </property>

</configuration>
```

hdfs-site.xml

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>

        <property>
                <name>dfs.blocksize</name>
                <value>536870912</value>
        </property>

         <property>
                <name>dfs.namenode.name.dir</name>
                <value>file:/home/hadoop/hadoop_store/hdfs/namenode</value>
        </property>

        <property>
                <name>dfs.datanode.data.dir</name>
        <value>/data/data1/hadoop, /data/data2/hadoop, /data/data3/hadoop, /data/data4/hadoop
</value>
        </property>

</configuration>
```

yarn-site.xml

```
<configuration>
<!-- Site specific YARN configuration properties -->
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>

        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value><server1></value>
        </property>

        <property>
                <name>yarn.scheduler.minimum-allocation-mb</name>
                <value>1024</value>
        </property>

        <property>
                <name>yarn.scheduler.maximum-allocation-mb</name>
                <value>49152</value>
        </property>

        <property>
                <name>yarn.scheduler.minimum-allocation-vcores</name>
                <value>1</value>
        </property>

        <property>
                <name>yarn.scheduler.maximum-allocation-vcores</name>
                <value>112</value>
        </property>

        <property>
                <name>yarn.nodemanager.vmem-pmem-ratio</name>
                <value>4</value>
        </property>

        <property>
                <name>yarn.nodemanager.resource.memory-mb</name>
                <value>409600</value>
```

```
        </property>

        <property>
                <name>yarn.nodemanager.resource.cpu-vcores</name>
                <value>112</value>
        </property>

        <property>
                <name>yarn.log-aggregation-enable</name>
                <value>true</value>
        </property>
</configuration>
```

mapred-site.xml

```
       <configuration>
               <property>
                       <name>mapreduce.framework.name</name>
                       <value>yarn</value>
               </property>

               <property>
                       <name>yarn.app.mapreduce.am.env</name>
                       <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
               </property>

               <property>
                       <name>mapreduce.map.env</name>
                       <value> HADOOP_MAPRED_HOME=$HADOOP_HOME,
       LD_LIBRARY_PATH=$LD_LIBRARY_PATH </value>
               </property>

               <property>
                       <name>mapreduce.reduce.env</name>
                       <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
               </property>

                <property>
                 <name>mapreduce.application.classpath</name>
                 <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*,
$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib-examples/*,
$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/sources/*,
```

```
$HADOOP_MAPRED_HOME/share/hadoop/common/*, $HADOOP_MAPRED_HOME/share/hadoop/common/lib/*,
$HADOOP_MAPRED_HOME/share/hadoop/yarn/*, $HADOOP_MAPRED_HOME/share/hadoop/yarn/lib/*,
$HADOOP_MAPRED_HOME/share/hadoop/hdfs/*, $HADOOP_MAPRED_HOME/share/hadoop/hdfs/lib/*</value>
                </property>

                <property>
                        <name>mapreduce.jobhistory.address</name>
                        <value><server1>:10020</value>
                </property>

                <property>
                        <name>mapreduce.jobhistory.webapp.address</name>
                        <value><server1>:19888</value>
                </property>

                <property>
                        <name>mapreduce.map.memory.mb</name>
                        <value>2048</value>
                </property>

                <property>
                        <name>mapreduce.map.cpu.vcore</name>
                        <value>1</value>
                </property>

                <property>
                        <name>mapreduce.reduce.memory.mb</name>
                        <value>4096</value>
                </property>

                <property>
                        <name>mapreduce.reduce.cpu.vcore</name>
                        <value>1</value>
                </property>

                <property>
                        <name>mapreduce.map.java.opts</name>
                        <value>-Xmx1536m -XX:+UseParallelGC -XX:ParallelGCThreads=32</value>
                </property>

                <property>
                        <name>mapreduce.reduce.java.opts</name>
```

```
        <value>-Xmx3072m -XX:+UseParallelGC -XX:ParallelGCThreads=32</value>
</property>

<property>
        <name>mapreduce.task.timeout</name>
        <value>6000000</value>
</property>

<property>
        <name>mapreduce.map.output.compress</name>
        <value>true</value>
</property>

<property>
        <name>mapreduce.map.output.compress.codec</name>
        <value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>

<property>
        <name>mapreduce.output.fileoutputformat.compress</name>
        <value>true</value>
</property>

<property>
        <name>mapreduce.output.fileoutputformat.compress.type</name>
        <value>BLOCK</value>
</property>

<property>
        <name>mapreduce.output.fileoutputformat.compress.codec</name>
        <value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>

<property>
         <name>mapreduce.reduce.shuffle.parallelcopies</name>
         <value>32</value>
</property>

<property>
          <name>mapred.reduce.parallel.copies</name>
          <value>32</value>
```

```
        </property>

</configuration>
```

hibench.conf

```
hibench.default.map/shuffle.parallelism 336
# 112 per node in a 3 node cluster
hibench.scale.profile bigdata
# the bigdata size configured as hibench.terasort.bigdata.datasize
30000000000 in ~/HiBench/conf/workloads/micro/terasort.conf
```

Revision History

ISSUE	DATE	DESCRIPTION
0.75	October 24, 2023	Updated: • *Section 2.3.1, Equipment Under Test*. • *Appendix*.
0.70	June 2, 2023	Initial issue.

October 24, 2023

Ampere Computing reserves the right to change or discontinue this product without notice.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

The information contained in this document is subject to change or withdrawal at any time without notice and is being provided on an "AS IS" basis without warranty or indemnity of any kind, whether express or implied, including without limitation, the implied warranties of non-infringement, merchantability, or fitness for a particular purpose.

Any products, services, or programs discussed in this document are sold or licensed under Ampere Computing's standard terms and conditions, copies of which may be obtained from your local Ampere Computing representative. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Ampere Computing or third parties.

Without limiting the generality of the foregoing, any performance data contained in this document was determined in a specific or controlled environment and not submitted to any formal Ampere Computing test. Therefore, the results obtained in other operating environments may vary significantly. Under no circumstances will Ampere Computing be liable for any damages whatsoever arising out of or resulting from any use of the document or the information contained herein.

Ampere Computing
4655 Great America Parkway, Santa Clara, CA 95054
Phone: (669) 770-3700
https://www.amperecomputing.com

Ampere Computing reserves the right to make changes to its products, its datasheets, or related documentation, without notice and warrants its products solely pursuant to its terms and conditions of sale, only to substantially comply with the latest available datasheet.

Ampere, Ampere Computing, the Ampere Computing and 'A' logos, and Altra are registered trademarks of Ampere Computing.

Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All other trademarks are the property of their respective holders.