accenture

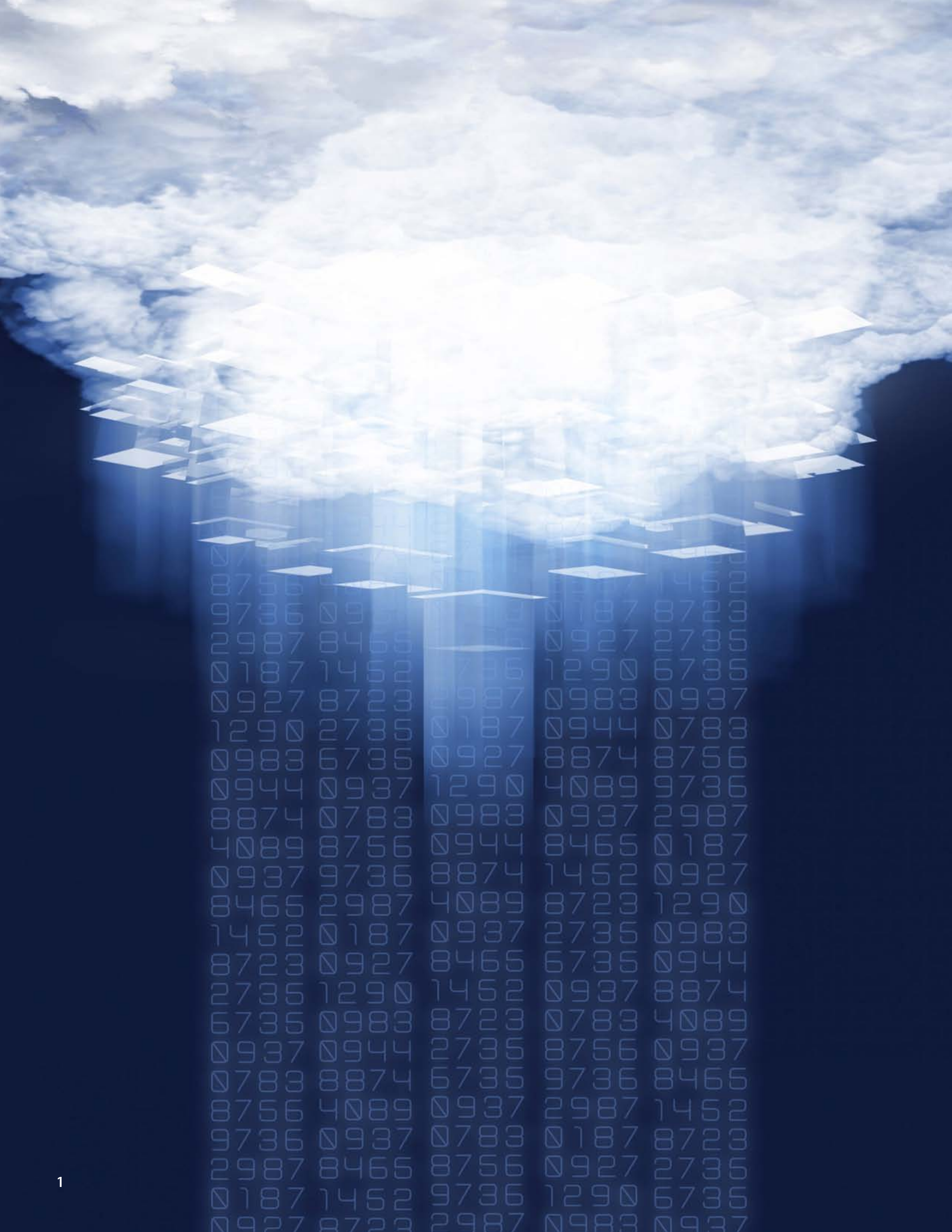
consulting | technology | outsourcing

Accenture Technology Labs

Cloud-based Hadoop Deployments: Benefits and Considerations

An updated price-performance comparison between a bare-metal and a cloud-based Hadoop cluster, including experiences using two cloud providers.

High performance. Delivered.

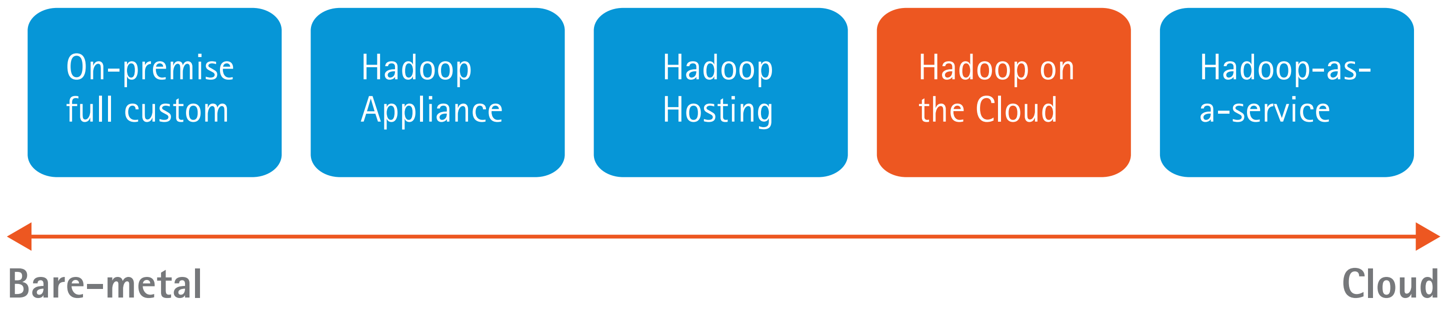


Introduction

Adoption of big data technology has changed many business organizations' perspective on data and its value. Traditional data infrastructure has been replaced with big data platforms offering capacity and performance increases at a linear cost increase, compared with traditional infrastructure's exponential cost increase. This change in how businesses store and process their data has led them to derive more insight from their existing data by combining multiple datasets and sources to yield a more complete view of their customers and operations. The success of businesses using big data to change how they operate and interact with the world has made many other businesses prioritize big data rollouts as IT initiatives to realize similar results.

Apache Hadoop™ ("Hadoop") has been at the center of this big data transformation, providing an ecosystem with tools for businesses to store and process data on a scale that was unheard of several years ago. Two key components of the Hadoop ecosystem are Hadoop Distributed File System (HDFS™) and Hadoop MapReduce®; these tools enable the platform to store and process large datasets (terabytes and above) in a scalable and cost-effective manner.

Figure 1. The spectrum of Hadoop deployment options with the studied deployment option highlighted



When enterprises adopt Hadoop, one of the decisions they must make is the deployment model. In our previous study[1], we identified four deployment options; however, we have now identified a new deployment option called "Hadoop on the cloud." The five deployment options, as illustrated in Figure 1, are as follows:

- **On-premise full custom.** With this option, businesses purchase commodity hardware, then they install software and operate it themselves. This option gives businesses full control of the Hadoop cluster.
- **Hadoop appliance.** This preconfigured Hadoop cluster allows businesses to bypass detailed technical configuration decisions and jumpstart data analysis.
- **Hadoop hosting.** Much as with a traditional ISP model, organizations rely on a service provider to deploy and operate Hadoop clusters on their behalf.
- **Hadoop on the cloud.** This new deployment option, and the focus of our study, allows organizations to create and customize Hadoop clusters on virtual machines utilizing the compute resources of the virtual instances and deployment scripts. Similar to the on-premise full custom option, this gives businesses full control of the cluster.
- **Hadoop-as-a-Service.** This option gives businesses instant access to Hadoop clusters with a pay-per-use consumption model, providing greater business agility.

To determine which of these options presents the right deployment model, we established in our previous study five key areas that organizations must consider: price-performance ratio, data privacy, data gravity, data enrichment, and productivity of developers and data scientists. Focusing on the price-performance ratio in this study, we wanted to confirm our previous result: cloud-based Hadoop deployments offer a better price-performance ratio than bare-metal. Additionally, our goal was to explore the performance impacts of data flow models and cloud architecture on the Accenture Technology Labs' Data Platform Benchmark suite.

Reusing the Accenture Technology Labs' Data Platform Benchmark suite, we continued to explore two divergent views related to the price-performance ratio for Hadoop deployments. A typical view is that a virtualized Hadoop cluster is slower because Hadoop's workload has intensive I/O operations, which tend to run slowly on virtualized environments. The other, contrasting view is that the cloud-based model provides compelling cost savings because its individual server node tends to be less expensive; furthermore, Hadoop is horizontally scalable.

Accenture's studies revealed that cloud-based Hadoop deployments—Hadoop on the cloud and Hadoop-as-a-Service—offer better price-performance ratios than bare-metal. (A bare-metal Hadoop cluster is the most common Hadoop deployment option in production environments; it consists of Hadoop deployed on physical servers without a virtualization layer.) These results confirm our initial dismissal of the idea that the cloud is not suitable for Hadoop MapReduce workloads given their heavy I/O requirements. Furthermore, the benefit of performance tuning is so huge that the cloud's virtualization layer overhead is a worthy investment because it expands performance-tuning opportunities. As a result, despite the sizable benefit, the performance-tuning process is complex and time-consuming, and thus requires automated tuning tools. In addition, we observed that remote storage options provided for better performance than local disk HDFS relying on data locality.

Leveraging our previously developed total-cost-of-ownership (TCO) model and performance-tuning methods of bare-metal Hadoop clusters and Hadoop on the cloud, Accenture Technology Labs was able to conduct the price-performance comparison of a bare-metal Hadoop cluster and Hadoop on the cloud at a matched TCO and using real-world applications. The following sections detail our study, containing information on the TCO model we developed and the Accenture Data Platform Benchmark, explaining the experiment setup and results, discussing the findings, and sharing our experiences with cloud providers while performing these studies.

Total Cost of Ownership

Continuing to focus on the price-performance ratio from our previous study, we found that it is more meaningful to compare the performance at the matched budget rather than at the matched hardware specification. Therefore, it is important to understand the TCO of the Hadoop deployments that we compared.

In this TCO analysis, we list the TCO components along with various factors needed to calculate the cost of each component. Calculating the TCO of a generic Hadoop cluster is a challenging—perhaps even impossible—task, because it involves factors that are unknown or that vary based on time. Given that, we put our best efforts into including representative numbers and being specific about the assumptions we made. Moreover, for comparison, we calculated the monthly TCO and translated capital expenditures into monthly operating expenses.

As stated earlier, we compared two Hadoop deployment options at the matched TCO budget. Table 1 illustrates the methodology we used to match the TCO. We first picked a bare-metal Hadoop cluster as a reference and calculated its TCO, which was $21,845.04 per month. Then, using $21,845.04 as the monthly TCO for Hadoop on the cloud, we allocated that budget to the necessary components and derived the resulting cloud-based capacity so that we could compare the performance of the two deployment options.

We excluded from comparison components that are difficult to quantify and agnostic to the deployment type, such as the staff personnel cost of data scientists and business analysts.

Table 1. TCO matching bare-metal Hadoop cluster and Hadoop on the Cloud

Bare-metal	Monthly TCO		Hadoop on the Cloud
	$21,845.04		
Staff for operation	$9,274.46	$3,091.49	Staff for operation
Technical support (third-party vendors)	$6,656.00	$1,294.94	Technical support (service providers)
Data center facility and electricity	$2,914.58	$1,816.58	Storage services
Server hardware	$3,000.00	**$15,642.03**	Virtual machine instances

Bare-metal Hadoop Cluster

The left half of Table 1 shows the monthly TCO breakdown of bare-metal Hadoop clusters, which is from our original study (TCO details have been included as a convenience for the reader). We picked the cluster size with 24 nodes and 50 TB of HDFS capacity. In practice, it is a reference point for small-scale initial production deployment. The following subsections explain each cost component and the assumptions we used.

Server hardware

In the TCO calculation, we estimated the hardware cost at $4,500 per node based on retail server hardware vendors. The modeled server node assumes four 2 TB hard disk drives, 24 GB memory, and 12 CPU cores. Also, this pricing includes a server rack chassis and a top-of-rack switch. Of course, multiple factors could change the given pricing, such as a different hardware configuration, a volume discount on a purchase, or regional or seasonal price discrimination.

To calculate the monthly TCO, we had to translate the onetime capital expense of the hardware purchase into the monthly operating cost. This translation typically uses a straight-line depreciation method—even distribution of the capital cost across a period of time. For the sake of comparison, we chose three years as the distribution period, which is one of the most commonly used periods for server hardware. However, the best period to use is debatable because of many influential factors, such as the expected lifetime of the hardware as well as the organization's asset-depreciation policy and its technology-refresh strategy

.

Data center facility and electricity

We budgeted $2,914.58 for the data center facility and electricity. For the data center facility, we assumed a tier-3 grade data center with a 10,000-square-foot building space including 4,000 square feet of IT space at a construction cost of $7,892,230. We used a 25-year straight-line depreciation method to translate it to operating cost. For electricity, we budgeted $252,565 per year, assuming 720 kW total power load at $0.09 per kWh. It includes both power and cooling of the entire facility, such as servers, storage, network, and failover power sources. Also, we budgeted $701,546 per year for building maintenance. In total, the annual facility TCO was $1,269,800.

We further assumed that 40 percent of the IT space is allocated for server racks, which is 70 percent actively occupied. With a rack with a capacity of 24 1U servers and a 30-square-foot footprint, the annual facility TCO is shared by 1,344 servers, the cost of which is $944.79. We also budgeted $1,500 per rack hardware that is shared by 24 servers with a five-year depreciation cycle, and $500 per data center switch cost per server per year. Taking all these factors into account, we budgeted $1,457.29 per node per year, which translates into $2,914.58 per month for the targeted 24-node cluster.

The assumptions we made above were heavily based on Gartner reports.[2]

Technical support from third-party vendors

Hadoop is an open-source product. Users may run into bugs or technical issues, or they may desire custom features. Even though anyone can patch the Hadoop project in theory, doing so requires a deep understanding of Hadoop architecture and implementation. For enterprise customers seeking production deployment of Hadoop, this is a significant risk. To meet the need for troubleshooting, Hadoop distributors typically offer technical support in the form of annual subscriptions per server node.

The retail pricing of an annual subscription is typically not publicly shared. However, Cloudera has shared its retail pricing, and we used it in our study, with Cloudera's permission. In particular, we used the retail pricing of Cloudera Enterprise Core: $3,328 per node per year with 24/7 support.

Staff for operation

A Hadoop cluster requires various operational tasks because it comes with the complexity of distributed systems. The Hadoop cluster should be deployed on reasonably chosen hardware and tuned with appropriate configuration parameters. It also requires cluster health monitoring and failure recovery and repair.[3] In addition, as workload characteristics change over time, the cluster also needs to be retuned. Also, the job schedulers should be controlled and configured to keep the cluster productive. Furthermore, because Hadoop is an evolving product, users must keep up with current Hadoop versions and integrate new tools in the Hadoop ecosystem as needed. Finally, the underlying infrastructure itself should be managed and kept available, which typically requires IT and system administration support.

There is no publicly available data point for Hadoop operation staff FTE cost, yet. The closest one we could find was Linux Server FTE cost data published by Gartner.[4] Based on the data, one Linux Server FTE can manage 28.3 servers, and the associated cost is $130,567 on average. Based on these assumptions, we budgeted $9,274.46 for operation staff personnel cost.

Hadoop on the Cloud (Google Compute Engine)

Hadoop on the cloud refers to using virtual instances within the cloud to deploy Hadoop clusters to run MapReduce jobs with the assistance of provided deployment scripts and remote storage services. To prove the value of cloud-based Hadoop deployments, we selected Google Compute Engine as our service provider for this deployment type. Google Compute Engine is a part of the Google Cloud Platform, which consists of many cloud-based offerings that combine to provide businesses with the ability to run their big data rollout in the cloud. In this section, we explain the TCO breakdown of our Hadoop on the cloud deployment using Google Compute Engine, along with the assumptions we used.

Staff for operation (cloud administrator)

Using the Hadoop-as-a-Service TCO as a guideline from the previous study, we retained the budget of $3,091.49 for cloud-related internal operation staff personnel cost, which is one-third of its bare-metal counterpart. Using a service provider like Google Cloud Platform shifts a large portion of operational burden to that provider. For example, Google Compute Engine deploys a fully configured Hadoop cluster with a few inputs from users such as the cluster's instance type and count using command-line.

Google Compute Engine's offerings allow for customized operation of Hadoop clusters; however, the need for an internal role to maintain and monitor these clusters still exists as in our previous study. This internal role takes the form of a cloud administrator, whose responsibilities can include monitoring the health of a company's assets that are deployed to the cloud as well as the cloud itself, making troubleshooting decisions, tuning the Hadoop cluster parameters for performance, owning the technical relationship with cloud service providers, and keeping up with newly offered features.

Technical support from service providers

Although there is a reduced technical risk by operating instances within the cloud, there is still a need for enterprise support if any technical issues are encountered. Google Cloud Platform provides four levels of technical support. Through our research, we found the "gold support" to be the most comparable to the level of support we assumed for our bare-metal TCO. Support charges are calculated by a fee schedule of varying percentages of product usage fees for a Google Cloud Platform account, instead of an annual per-node subscription cost as with our bare-metal deployment. Using the published fee schedule and our anticipated product usage fees, gold support from Google Cloud Platform costs $1,294.94.

Storage services (Google Cloud Storage)

There are many benefits to storing input and output data in Google Cloud Storage, rather than maintaining a Hadoop cluster to store data in HDFS on the cloud. First, users can reduce the server instance cost by tearing down the Hadoop cluster when not running MapReduce jobs. Second, multiple Hadoop clusters can easily run analyses on the dataset in parallel without interfering with one another's performance. This approach limits the data locality; however, the processing capabilities provided by the number of affordable instances and methods for accessing the data efficiently compensate for the need to access the data outside of the cluster.

In calculating the required volume for cloud storage, we assumed 50 percent occupancy of the available HDFS space in the bare-metal cluster, because users need spare room when planning the capacity of bare-metal clusters. First of all, a bare-metal cluster needs extra HDFS space to hold temporary data between cascaded jobs. Second, it needs extra local temporary storage to buffer the intermediate data shuffled between map tasks and reduce tasks. Lastly, it needs to reserve room for future data growth given that a bare-metal cluster does not expand instantaneously. A Google Compute Engine cluster, on the other hand, comes with local storage and HDFS and thus does not need space in Google Cloud Storage for temporary storage. Also, Google Compute Engine clusters do not need to be over provisioned for future growth because permanent storage is outside of the cluster. Based on pricing and our assumption of 50 percent utilization (of the HDFS space within the bare-metal cluster), the storage requirement of 25 TB on Google Cloud Storage costs $1,816.58.

Virtual machine instances (Google Compute Engine)

After considering all the above components, the budget leaves $15,642.03 for Google Compute Engine spending for virtual machine instances.

At the time of our testing, Google Compute Engine offered 12 instance types with or without local disk storage and a single per-minute pricing option. Because it is both time- and cost-prohibitive to run the benchmark on all 24 combinations, we selected four instance types with local disk storage: standard 4 vCPU cores (n1-standard-4-d), standard 8 vCPU cores (n1-standard-8-d), high-memory 4 vCPU cores (n1-highmem-4-d), and high-memory 8 vCPU cores (n1-highmem-8-d). We chose these four instances because their processing capabilities were representative of production Hadoop nodes (bare-metal and cloud deployments), local disk storage allowed for HDFS installation, and Google Compute Engine instances with 4 vCPU cores or more have sole access to a hard disk within the virtual machine, removing the need to share resources with another virtual instance and yielding better performance. We excluded two instance families for specific reasons: high CPU instances have a CPU-memory resource ratio that is not balanced for our MapReduce workloads; shared-core instances do not provide the necessary CPU requirements to be an effective option.

In calculating the number of affordable instances for Google Compute Engine, we assumed 50 percent cluster utilization, which means the percentage of time that the Google Compute Engine cluster is active and operating over the course of one month. In the context of a pay-per-use model, a higher utilization assumption leads to a smaller number of affordable instances for a given budget. This assumption is from our original study and was used to calculate the number of affordable instances for our four chosen instance types listed in Table 2.

After we completed the study, Google Compute Engine released three new instance types, all offering 16 vCPU cores and double the amount of memory in the 8 vCPU core machines, up to 104 GB of RAM. Google Compute Engine also deprecated local disks in favor of Google Compute Engine Persistent Disks,[5] which offer high and consistent performance; low and predictable price; safety in the form of redundancy, encryption and checksum verification; and management simplicity and flexibility that is unavailable with local disks. This resulted in decreased instance costs to help offset any costs incurred by persistent disk usage.

Table 2. Affordable number of instances

Instance type	Number of Instances
n1-standard-4-d	80
n1-standard-8-d	40
n1-highmem-4-d	70
n1-highmem-8-d	35

Accenture Data Platform Benchmark

Utilizing the same benchmark from our previous study, the Accenture Data Platform Benchmark suite comprises multiple real-world Hadoop MapReduce applications. Within Accenture Technology Labs, we have been fortunate to directly monitor enterprise clients' business needs and to solve their real-world business problems by leveraging big-data platform technologies, including Hadoop. On the basis of such client experience, our internal road map, and published literature, we assembled the suite of Hadoop MapReduce applications, which we named Accenture Data Platform Benchmark.

We used the following selection process. First, we categorized and selected common use cases of Hadoop MapReduce applications: log management, customer preference prediction, and text analytics. Then, from each category, we implemented a representative and baseline workload with publicly available software packages and public data. This strategy makes the benchmark agnostic to any of our clients' custom design and thus easier to share, while keeping it relevant. The rest of this section introduces three workloads in the benchmark suite.

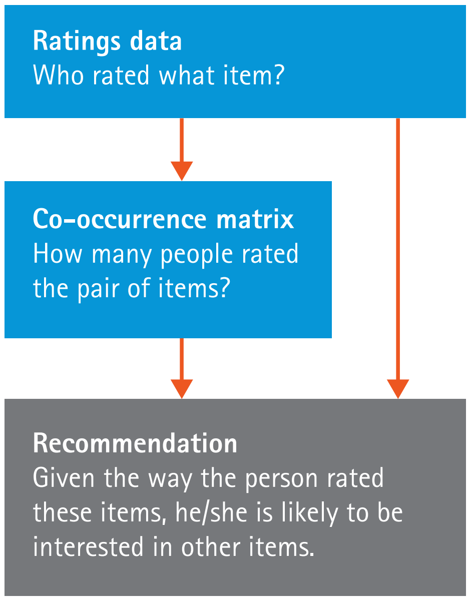
Recommendation engine

A recommendation engine is one of the most popular instantiations of customer preference prediction. Many industries—including retail, media content providers, and advertising—use recommendation engines to predict the unexpressed preference of customers and further stretch revenue potential.

Although there are many algorithms and use cases for recommendation engines, we used an item-based collaborative filtering algorithm and a movie recommendation engine as a reference. It reads a history of movie ratings from multiple users regarding multiple movies. Then, it builds a co-occurrence matrix that scores the similarity of each pair of movies. Combining the matrix and each user's movie-rating history, the engine predicts a given user's preference on unrated movies.

We used the collaborative filtering example in the Apache Mahout project. Moreover, we used synthesized movie ratings data from 3 million users on 50,000 items, with 100 ratings per user on average.

Figure 2. Recommendation engine using item-based collaborative filtering



Ratings data
Who rated what item?
Co-occurrence matrix
How many people rated the pair of items?
Recommendation
Given the way the person rated these items, he/she is likely to be interested in other items.

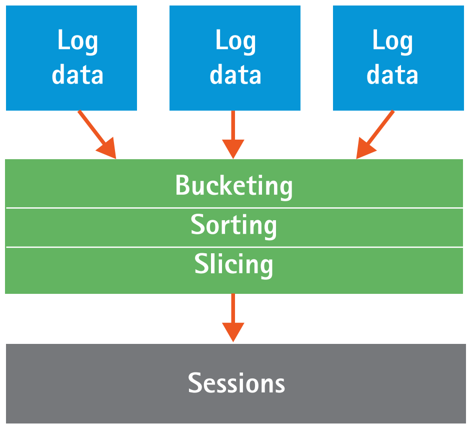
Sessionization

In the context of log analysis, a session is a sequence of related interactions that are useful to analyze as a group. The sequence of web pages through which a user navigated is an example of a session. Sessionization is one of the first steps in many types of log analysis and management, such as personalized website optimization, infrastructure operation optimization, and security analytics.

Sessionization is a process of constructing a session, taking raw log datasets from multiple sources. It reads a large number of compressed log files, decompresses and parses them, and buckets the log entries by a session holder identifier (for example, by user ID). Then, within each bucket, it sorts the entries by time order, and finally slices them into sessions based on a time gap between two consecutive logged activities.

We used synthesized large log datasets whose entries had a timestamp, user ID, and the log information content (140 random characters, in this case). The application relies on user ID for bucketing, timestamp for sorting, and 60 seconds as an implied session boundary threshold. For the study, we used about 150 billion log entries (~24 TB) from 1 million users and produced 1.6 billion sessions.

Figure 3. Sessionization



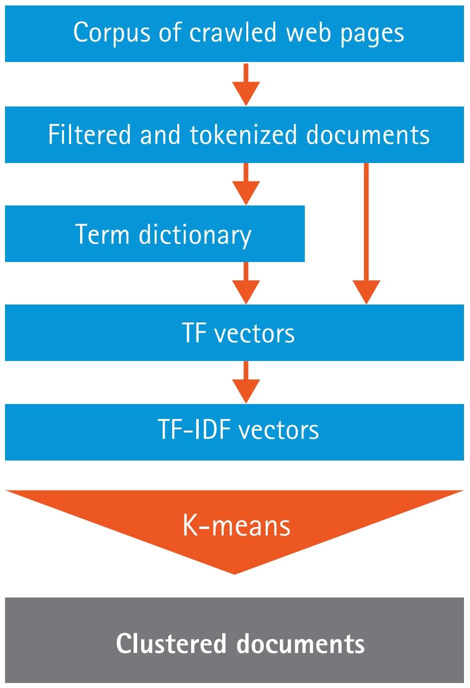
Document clustering

Document clustering is one of the important areas in unstructured text analysis. It groups a corpus of documents into a few clusters. The document clustering, as well as its building blocks, has been popularly used in many areas, such as search engines and e-commerce site optimization.

The application starts with a corpus of compressed crawled web pages. After decompression, it reads out and parses each html document, then extracts tokenized terms but filters out unnecessary words ("stop words"). Next, it builds a term dictionary: a set of pairings of a distinct term and its numerical index. Using this term dictionary, it maps each tokenized document to its corresponding term frequency (TF) vector, which lists the occurrence of each term in the document. To enhance the precision of the clustering, it normalizes these TF vectors into term frequency—inverse document frequency (TF-IDF) vectors. Finally, taking these TF-IDF vectors, it runs a k-means clustering algorithm to cluster the documents.

We used a crawled web page dataset publicly available from the Common Crawl project hosted in Amazon Web Services, Amazon S3TM. Given the size of clusters undergoing testing, we used 3 TB of compressed data (10 TB uncompressed) or 300 million web pages.

Figure 4. Document clustering



Experiment Setup

Bare-metal deployment

For the bare-metal Hadoop deployment, we used the Hadoop hosting service from MetaScale,[6] a Sears Holdings subsidiary. The cluster we used for study has a client node, a primary NameNode, a secondary NameNode, a JobTracker node, and 22 worker-node servers, each of which runs a DataNode for HDFS as well as TaskTracker for MapReduce. Table 3 lists the detailed hardware specification of the cluster. We preloaded the input data to its HDFS and stored the output and the intermediate data between cascaded jobs, also in HDFS.

Table 3. Hardware specification of the bare-metal cluster

Cluster in total	
Client node	1
Master nodes	3
Worker/Data nodes	22
Cores	264
Memory (GB)	528
Raw TB	176
HDFS TB Available	148.7
Usable TB (w/3 replicas)	49.6

Worker/Data node summary	
Model	Dell R415
CPU type	Opteron 4180
# of cores	12
Clock speed (GHz)	2.6
Memory bus speed (MHz)	1333
# of disks	4
Each disk's capacity (TB)	2
Total Capacity (TB)	8

Cloud-based deployment

For our cloud-based deployment, we used the number of affordable instances from Table 2 to size our Hadoop clusters on Google Compute Engine. Each cluster contained one master node to serve as the NameNode and JobTracker, and the remaining number instances acted as worker nodes, each of which ran a DataNode and a TaskTracker. For example, our TCO gives us 80 nodes for instance type n1-standard-4-d; this results in 1 master node and 79 worker nodes for our Hadoop on the cloud cluster deployment.

Architecture setups

We split the benchmarks into two architectures to match the available configurations on Google Compute Engine:

- **Local Disk HDFS** – Google Compute Engine instances using local disks for HDFS
- **Google Cloud Storage connector for Hadoop** – Google Compute Engine instances using local disks for HDFS plus a new Google Cloud Storage connector for Hadoop; allowing direct Google Cloud Storage access and eliminating the need to copy data from Google Cloud Storage to HDFS

As discussed in the next section, both architectures utilize distinct data-flow methods; however, the number of affordable instances remained constant and only configuration changes were made to support the data-flow methods used.

Data flow methods

In order to run the benchmarks, we preloaded the input data into Google Cloud Storage for permanent storage during the benchmarking process. Once the input data was stored in Google Cloud Storage, we chose the appropriate data flow model for each architecture:

For the "Local Disk HDFS" architecture, we chose the data-flow method detailed in Figure 5. Input data was copied by a streaming MapReduce job (provided by Google) to the HDFS of the Google Compute Engine Hadoop cluster before starting the MapReduce job, then output data was copied by another streaming MapReduce job (provided by Google) to Google Cloud Storage for permanent storage once the MapReduce job completed. For some jobs, this increased the execution times given that the copy times were a part of the final numbers. In the next section, we will see the impact on execution time using this data-flow method.

"Google Cloud Storage connector for Hadoop" benchmarks use the method detailed in Figure 6. With the availability of direct access to Google Cloud Storage via the Hadoop connector, there was no longer a need for copying input or output data from the Google Compute Engine cluster. We were able to minimize execution times and saw a performance increase from using this data flow method despite data locality concerns. In our previous study, we used the data flow method in Figure 6 for our Hadoop-as-a-Service deployments.

Figure 5. Local Disk HDFS data-flow model

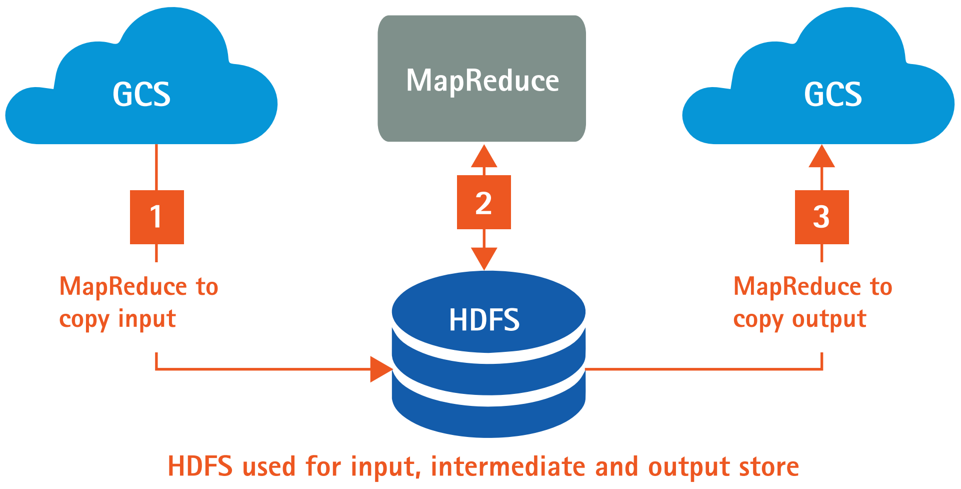


Figure 7 was not used in this experiment; however, there are key advantages to this data-flow method. First, instance failures will not result in a loss of data because all data is stored in Google Cloud Storage. Second, by using Google Cloud Storage, we are using the distributed nature of the storage similar to HDFS providing high throughput. Finally, we can create Google Compute Engine clusters that are dynamically sized using this method. This allows us to use a varying number of TaskTrackers to complete difficult workloads in less time with the additional computing power (in the form of extra map/reduce slots). The added complexity of managing HDFS while adding and removing instances is eliminated by using Google Cloud Storage connector for Hadoop and Google Cloud Storage for all storage.

Figure 6. Google Cloud Storage connector for Hadoop data-flow model

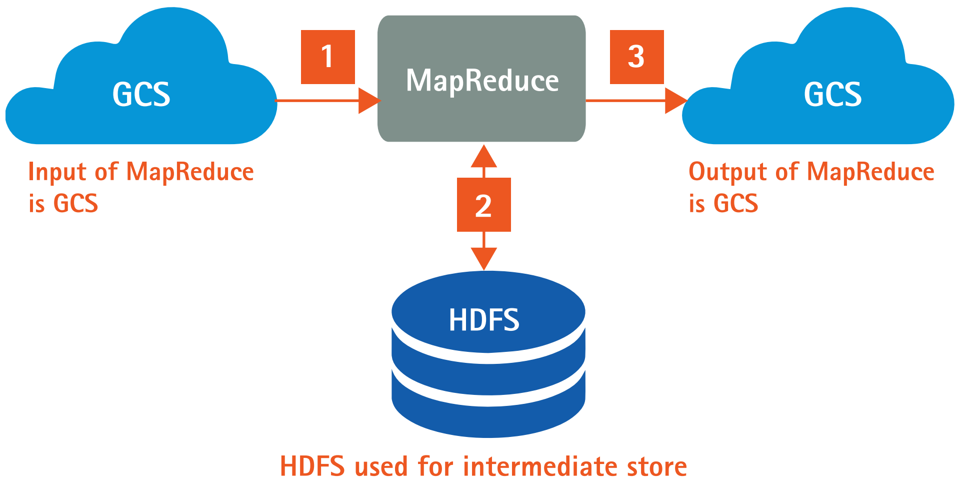
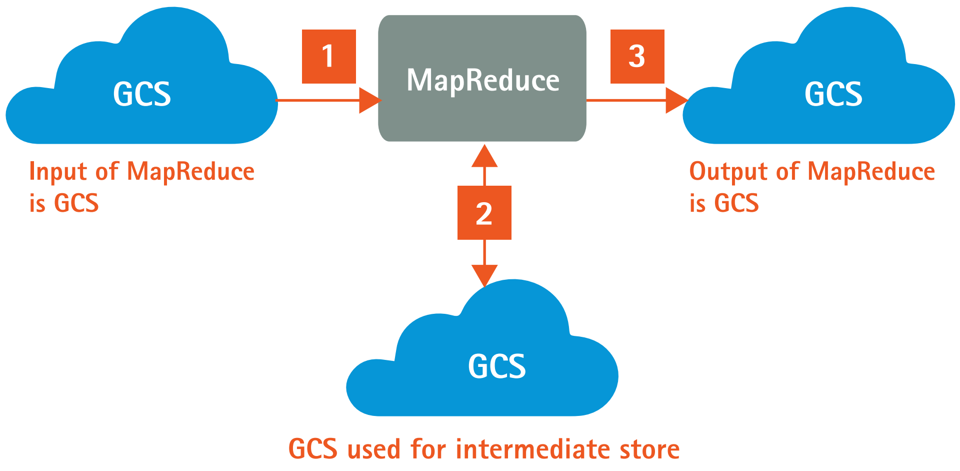

GCS
1
MapReduce
3
GCS
Input of MapReduce is GCS
2
Output of MapReduce is GCS
HDFS
HDFS used for intermediate store

Figure 7. Google Cloud Storage based data-flow model


GCS
1
MapReduce
3
GCS
Input of MapReduce is GCS
2
Output of MapReduce is GCS
GCS
GCS used for intermediate store

Hadoop configuration

Utilizing the provided deployment scripts from Google allowed us to quickly deploy Hadoop clusters of the instance type and number of nodes we required. The customizable scripts allowed us to configure the Hadoop clusters choosing the number of map/reduce slots and the heap sizes of the Hadoop daemons for each instance type. Our goal was to keep a map-slots-to-reduce-slots ratio of 3:1 when planning because this is typical in most Hadoop deployments, as shown in Table 4.

Owing to the memory demands of the sessionization workload we had to reduce the number of map/reduce slots (Table 5) to ensure that the large heap size demands could be met for each slot and the CPU-intensive operations did not result in CPU thrashing that would lead to processing delays.

For worker node Hadoop daemons, we used 512 MB for TaskTracker and DataNode daemons on all nodes regardless of instance type. The JobTracker and NameNode daemons on the master nodes were configured to use 60 to 70 percent of available memory for the JobTracker and 20 to 24 percent for the NameNode, with the remainder free as shown in Table 6.

By configuring each instance type to the above parameters, we ensured that we were able to take advantage of the available CPU and memory resources for each instance, just as we would when configuring a bare-metal cluster.

Test setup

Combining the four different cluster setups with the two architecture configurations gave us eight platforms upon which to test each of our three benchmark workloads. We applied the same performance-tuning methods used in our previous study for each combination of the workload and the cluster setup. We tuned them by both applying manual tuning techniques and getting help from an automated performance-tuning tool. The tuned performance results are shown and discussed in the next section.

Table 4. Map and Reduce slot configurations for Recommendation Engine and Document Clustering

Instance types	Map slots	Reduce slots
n1-standard-4-d	6	2
n1-standard-8-d	10	3
n1-highmem-4-d	8	3
n1-highmem-8-d	18	6

Table 5. Map and Reduce slot configurations for Sessionization

Instance types	Map slots	Reduce slots
n1-standard-4-d	3	2
n1-standard-8-d	8	2
n1-highmem-4-d	4	2
n1-highmem-8-d	9	3

Table 6. JobTracker and NameNode heap configurations

Instance types	JobTracker	NameNode	Free memory	Total memory
n1-standard-4-d	9,216 MB	3,072 MB	3,072 MB	15,360 MB
n1-standard-8-d	21,473 MB	6,175 MB	3,072 MB	30,720 MB
n1-highmem-4-d	17,306 MB	6,246 MB	3,072 MB	26,624 MB
n1-highmem-8-d	37,274 MB	12,902 MB	3,072 MB	53,248 MB

Experiment Result

Continuing from our previous study, we conducted the price-performance comparison of a bare-metal Hadoop cluster and Hadoop on the cloud at the matched TCO using real-world applications.

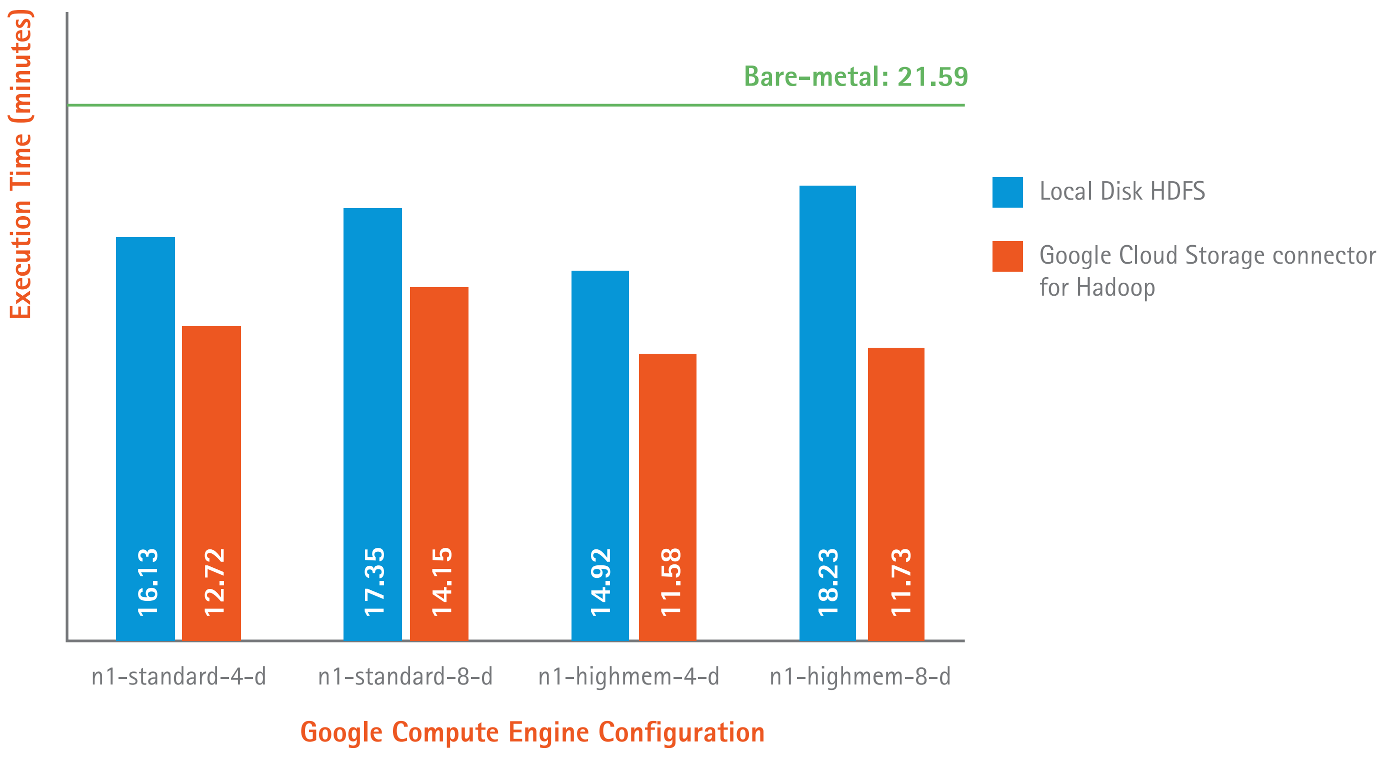
In the following figures, we show the execution time comparison of a bare-metal Hadoop cluster and eight different options from Google Compute Engine. All cloud-based Hadoop clusters resulted in a better price-performance ratio with the Google Compute Engine pricing option. The results of this study verify the claims of our original study: the idea that the cloud is not suitable for Hadoop MapReduce workloads given their heavy I/O requirements has been debunked; cloud-based Hadoop deployments provide a better price-performance ratio than the bare-metal counterpart.

Recommendation engine

The recommendation engine workload comprises ten cascaded MapReduce jobs, completing in 21.59 minutes on the bare-metal cluster. In Figure 9, we can see that all Google Compute Engine instances, regardless of type and architecture, were able to outperform the bare-metal cluster. For this workload, the n1-highmem-4d instance type outperformed all other Google Compute Engine instance types when using the Google Cloud Storage connector for Hadoop. Using the connector resulted in an average execution time savings of 24.4 percent. Of the savings, 86.0 percent (or 21.0 percent overall) came from removing the need for input and output data copies, and 14.0 percent of the savings (or 3.4 percent overall) by using the connector. This reduction of time has decreased compared with the sessionization workload because this workload has several cascading jobs.

The opportunity for speed-up using the connector is dependent on the amount of reads and writes to Google Cloud Storage. Because the workload reads input data only in the first job and writes output data only in the last of the ten cascaded jobs, there is limited opportunity to improve the execution time using the connector. Also the relatively small dataset (5 GB) for the recommendation engine is able to be processed more quickly on the Google Compute Engine instances and results in less data that need to be moved between Google Cloud Storage and the cluster. Despite the MapReduce framework overhead for launching the ten cascaded MapReduce jobs, the cloud-based Hadoop instances are able to outperform their bare-metal counterparts.

Figure 8. Recommendation engine execution times



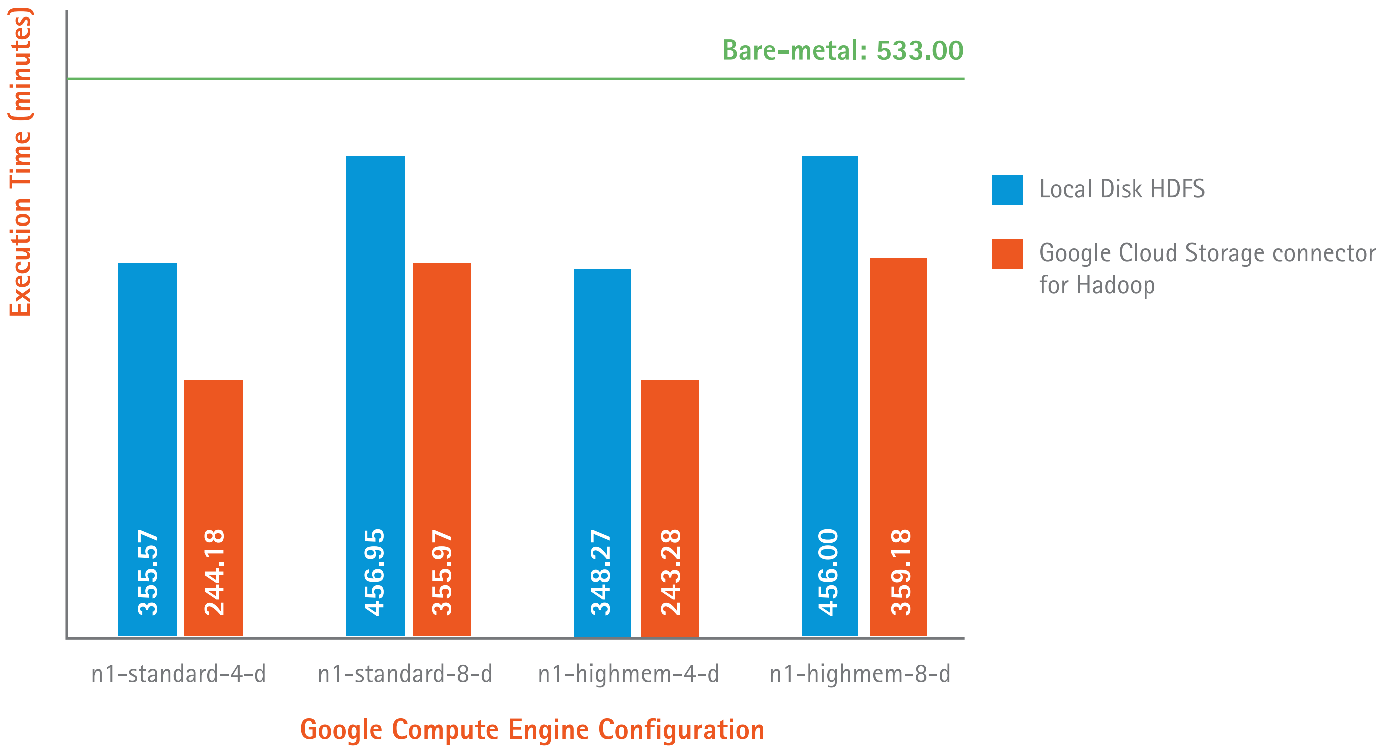
Sessionization

For the sessionization workload, all eight Google Compute Engine configurations outperformed the bare-metal Hadoop cluster. There are two key takeaways from this result.

First, we can see the impacts of choosing instance types that complement the workload. Sessionization rearranges a large dataset (24 TB uncompressed; ~675 GB compressed), which requires a large heap size per task slot in addition to computing resources for managing the decompression of input files and compression of output files. As mentioned earlier, this large heap size required us to change the number of map/reduce slots for this workload (see Table 5). For this workload, n1-standard-4-d and n1-highmem-4-d both provide the fastest results as well as the best price-performance ratio. The number of affordable instances for each of these instance types—80 and 70, respectively—helps this workload by providing a large number of map/reduce slots to tackle the 14,800 map tasks and 1,320 reduce tasks as well as balancing the number of slots with the available CPU resources to prevent thrashing. The CPU-intensive reduce tasks took 1.5-2 times as long to complete versus the map tasks; being able to distribute the CPU-intensive reduce tasks over more nodes reduced the execution time significantly.

Second, we can see that data locality is not a critical factor when using the Google Cloud Storage connector for Hadoop. Using the connector resulted in an average execution time savings of 26.2 percent. Of the savings, 25.6 percent (or 6.7 percent overall) came from removing the need for input and output data copies, and 74.4 percent of the savings (or 19.5 percent overall) by using the connector. This large speed-up from the connector is thanks to the nature of the workload as a single MapReduce job. Overhead with the NameNode and data locality issues such as streaming data to other nodes for processing can be avoided by using the connector to supply all nodes with data equally and evenly. This proves that even with remote storage, data locality concerns can be overcome by using Google Cloud Storage and the provided connector to see greater results than using traditional local-disk HDFS.

Figure 9. Sessionization execution times

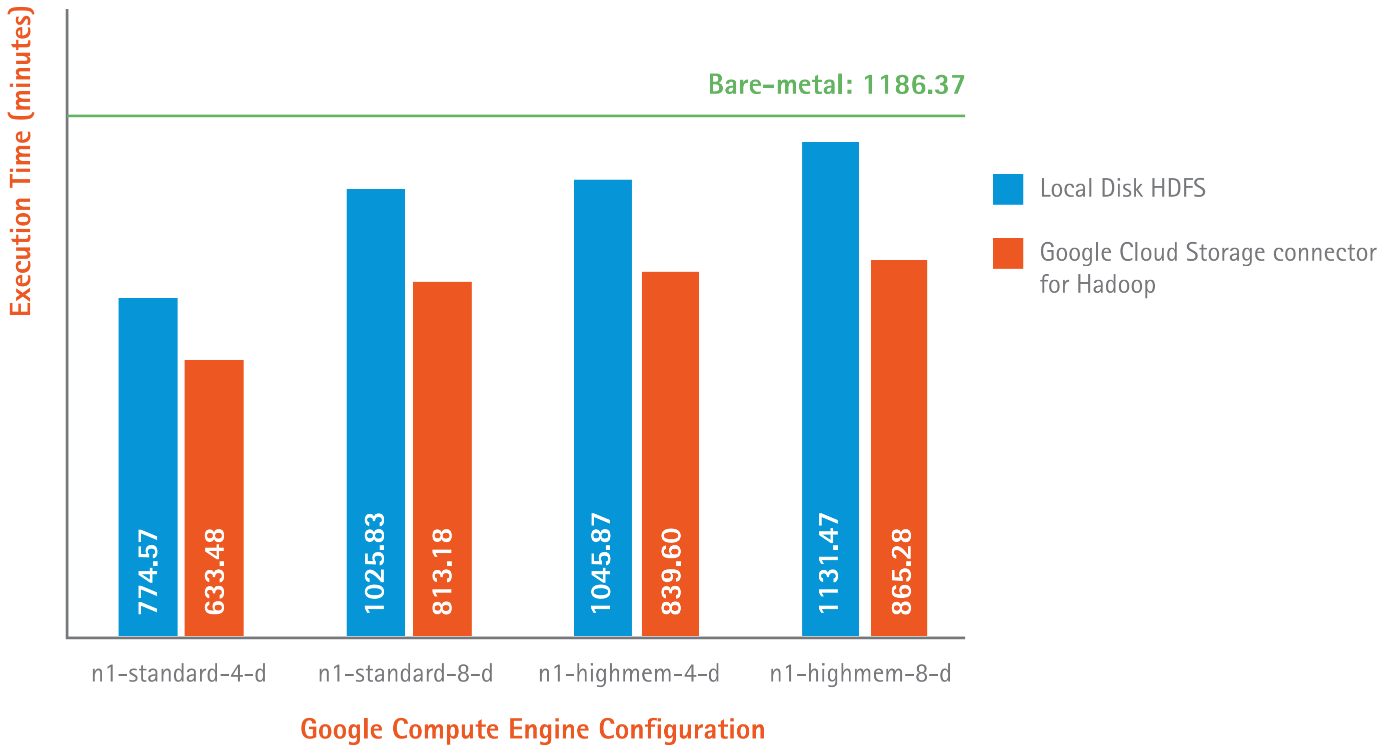


Document clustering

Similar speed-ups from the connector were observed with the document-clustering workload. Using the connector resulted in an average execution time savings of 20.6 percent. Of the savings, 26.8 percent (or 5.5 percent overall) came from removing the need for input and output data copies, and 73.2 percent of the savings (or 15.0 percent overall) by using the connector. Owing to the large amount of data processed (~31,000 files with a size of 3 TB) by the first MapReduce job of the document-clustering workload, the connector is able to transfer this data to the nodes much faster, resulting in the speed-up.

As in sessionization, we were able to see that n1-standard-4-d was able to perform better than other configurations owing to the balance of slots to CPU resources to prevent thrashing and the number of nodes to distribute the CPU-intensive vector calculations across multiple nodes. However, n1-highmem-4-d did not perform as well as in the sessionizer because the additional map/reduce slots combined with the more memory- and CPU-intensive workload strained the virtual instance. Again as in sessionization, we can see that there are benefits to choosing the instance type for a given workload to maximize efficiency and price-performance ratio.

Figure 10. Document clustering execution times



Discussion of Results

Performance impact: I/O virtualization overhead and performance tuning

As in our previous study, the I/O virtualization overhead did not cause a noticeable reduction in performance during our experiment. Our results from the document-clustering workload reinforced this observation in that the runtimes of local-disk and network I/O bound non-MapReduce Java tasks were better or slightly above those on bare-metal, as shown in Table 7.

The performance can also be attributed to the fact that the instances we tested had sole access to their local disks, reducing the risk of contention with another virtual instance. This bare-metal-like performance within the cloud coupled with a larger number of affordable instances to bare-metal nodes accounts for much of the performance increases we observed.

Through virtualization, cloud offers a variety of virtual machine instances, each with hardware configurations to support different types of workloads. In our results, we saw that the n1-highmem-4-d was a good instance type for two of the three workloads whereas n1-standard-4-d performed consistently well across all three workloads. Unlike bare-metal clusters, cloud instances can be configured to meet the demands of the workload, not only from the software level of Hadoop parameters but also at the hardware level.

In an effort to address this resource limitation Hadoop YARN™ aims to provide the flexibility of customizing CPU and memory resources for each MapReduce job across the cluster. As YARN matures and more features are added to the resource manager, it is possible for YARN to offer performance comparable to that of the cloud on bare-metal hardware; however, the scale and resource management issues of the hardware will still exist. By using cloud-based Hadoop deployments, these issues and overhead are shifted to the cloud providers, freeing organizations to invest time and effort into existing and new ventures.

Table 7. Runtimes of Disk/Network I/O bound Non-MapReduce Java tasks

Non-MapReduce Task	Bare-metal Hadoop (min)	Avg. of Cloud-based Hadoop (min)
Dictionary Creation	6:30	6:37
Frequency File Creation	3:30	2:41
K-means Centroid Selection	22:30	21:17

Automated performance tuning

Selecting a virtual instance that supports our workloads is one method of performance tuning. Additionally, we can use other tuning techniques to improve performance; however, the cost to obtain this performance increase is typically a time-consuming and iterative process that requires deep knowledge of Hadoop. In our previous study, we attempted manual and automated tuning practices to compare the two approaches. Manual tuning of the sessionization workload took more than two weeks overall; each iteration took about a half to a full day including performance analysis, tuning, and execution. This intensive process did result in significant savings, reducing the time from more than 21 hours to just over 9 hours, although at the cost of time and labor.

Most organizations do not have the resources or time to perform manual tuning, resulting in the need for automated performance-tuning tools. We used an automated performance-tuning tool called Starfish to achieve the performance-tuning results for this and the previous study. Starfish operates in two phases: first profiling the standard workload to gather information and second analyzing the profile data to create a set of optimized parameters, executing the result as a new workload. With Starfish, we could minimize the manual analysis and tuning iterations and achieve significant performance improvement. The optimized recommendation engine workload improved the performance by eight times relative to the default parameter settings within a single performance-tuning iteration by using Starfish.

As cloud architectures change, automated performance-tuning tools are necessary because they handle changes in underlying architectures, simplifying the performance-tuning operations. Without a tool like Starfish, organizations are forced to perform multiple time-consuming guess-and-check iterations with the hope of seeing large performance increases.

Data locality and remote storage

When MapReduce and HDFS were introduced, the replicated nature of the data ensured that data would not be lost as the result of a node failure; more importantly, it provided MapReduce with the ability to process the data in multiple locations without needing to send the data across the network to another node. This notion of data locality is exploited within MapReduce today, to minimize the network traffic between nodes copying data and to process data more quickly by accessing the data from the local disks. In most distributed systems, data locality is exploited for performance advantages; therefore, it is easy to see why remote storage is viewed as a less desirable alternative given that the data is outside of the machine.

From our study, we can see that remote storage powered by the Google Cloud Storage connector for Hadoop actually performs better than local storage. The increased performance can be seen in all three of our workloads to varying degrees based on their access patterns. Workloads like sessionization and document clustering read input data from 14,800 and about 31,000 files, respectively, and see the largest improvements because the files are accessible from every node in the cluster. Availability of the files, and their chunks, is no longer limited to three copies[7] within the cluster, which eliminates the dependence on the three nodes that contain the data to process the file or to transfer the file to an available node for processing.

In comparison, the recommendation engine workload has only one input file of 5 GB. With remote storage and the connector, we still see a performance increase in reading this large file because it is not in several small 64 MB or 128 MB chunks that must be streamed from multiple nodes in the cluster to the nodes processing the chunks of the file. Although this performance increase is not as large as the other workloads (14.0 percent compared with 73.2 to 74.4 percent with the other workloads), we can still see the value of using remote storage to provide faster access and greater availability of data when compared with the HDFS data locality model. This availability of remote storage on the scale and size provided by Google Cloud Storage and other cloud vendors unlocks a unique way of moving and storing large amounts of data that is not available with bare-metal deployments. In addition, remote storage is able to grow and adapt to business needs seamlessly without the cost of additional infrastructure.

Comparison with previous study

Examining the results from our both our previous study and this study, it could be tempting to compare the execution times directly between cloud providers. However, this would not be an accurate comparison because two different deployment options are studied, time has passed from the original benchmarks, architectures may have changed, newer versions of Hadoop exist, and instance types do not match one-to-one. Our study's goal is to compare each platform separately at the matched TCO level to a bare-metal Hadoop cluster to examine the price-performance ratio and to detail the benefits of cloud-based Hadoop clusters.

Experiences with Cloud Providers

Performing this study on multiple cloud platforms has yielded several key takeaways for businesses to consider when deciding to use cloud-based Hadoop deployments.[8]

Workload utilization and demands

To take advantage of the performance that the cloud offers, businesses must understand their workload needs. Our studies assumed 50 percent cluster utilization for the TCO analysis; however, the actual usage will depend on the workloads run and the service-level agreements (SLA) for each workload. When utilization is greater than the assumed 50 percent, this reduces number of affordable instances by requiring more budget to run instances, which could lead to longer SLAs.

Utilization below 50 percent gives businesses three main options. First, the savings allows for a greater number of affordable instances to obtain faster SLAs with more instances and/or using more powerful instance types. Second, unused budget can be used for R&D purposes to discover new insight from the data, optimization testing, and pilot environments for on-boarding additional workloads. Third, the savings can be used elsewhere within the business or result in money saved.

Evaluating each instance type along with a variable number of instances is both time- and cost-prohibitive for businesses. This is why automated performance tuning tools like Starfish allow estimations to be made once a job is profiled. Starfish can take the profiled data and estimate the changes in the workload when the number of nodes is changed in addition to the size of the instances. With this information, the overall budget, and SLAs, businesses can craft their TCO analysis to a utilization percentage that is appropriate for their needs.

Pricing structure

Differences in pricing structure can have a significant impact on the monthly TCO costs, and understanding the available pricing options along with anticipated use can help select the option that works best. Cloud providers tend to bill for usage of their services by time (or quantity) used, with some providers offering multiple tiers of pricing to suit different customers.

Google Compute Engine offers per-minute billing, which can yield many advantages over the per-hour billing offered by other services. For workloads that complete in under an hour, like the recommendation engine, we can see significant savings with the per-minute billing option. Running the recommendation engine for 30 minutes (including provisioning time) results in a 50 percent cost reduction using per-minute billing. Per-minute billing can also take advantage of the 30-minute runtime by doubling the original cluster size and paying the same cost as with per-hour billing. Google Compute Engine charges a minimum billing time for all instances of at least ten minutes which is one-sixth less than the typical one-hour minimum charge in per-hour billing. This difference in billing can also reduce costs during initial setup and testing for when clusters need to be quickly redeployed owing to scripting and other errors.

Currently, Google Compute Engine offers only a single pricing tier, competitively priced below some providers. Other cloud providers offer several pricing tiers that allow businesses to reserve compute resources through an up-front contract fee over a specified time period for discounted hourly rates, or take advantage of compute resources not in use for a deep discount with the risk of losing them to another customer paying a higher price. Multiple pricing tiers can be more beneficial to some longer-running workloads, taking advantage of the discounted rates to accommodate the increased cluster utilization.

Pricing structure is a major component in the choice of a cloud provider. Google Compute Engine offers many advantages relative to other providers while currently lacking the additional discounted pricing tiers found elsewhere.

Cloud architecture

Underlying architecture can significantly affect performance for clients; understanding the current technology available from cloud providers as well as their upgrade plans will help to make the choice easier. Unlike bare-metal refresh cycles of several years, most cloud providers upgrade their architecture multiple times a year. This presents a challenging task for many organizations because they have to actively manage how they optimize for the platform to take advantage of the latest available performance. While this may seem daunting, allowing cloud providers to manage the hardware frees companies from the logistics and complications of upgrading and maintaining this ever-changing hardware.

A business should not only understand the architecture provided by the cloud providers but also know its requirements for the workloads it anticipates running. Memory and CPU requirements that a business would have used to determine a bare-metal cluster configuration are used to select virtual instances instead. Google Compute Engine instances provide a broad mix of options to choose instances that match with the demands of a given workload. An instance with more than 64 GB of RAM was missing from the available instances during testing, leading to slower execution time with the memory-intensive sessionization workload. The high-memory demands reduced the number of available slots that could be used for the map and reduce phases, leading to unused CPU on n1-highmem-8-d and full memory. After testing completed, new instances were offered with 64 GB and 104 GB of RAM, but we were unable to test these instance types.

In contrast, the document-clustering workload showed bare-metal like I/O performance on Google Compute Engine for key tasks in between MapReduce jobs, detailed in the "Discussion of Results" section. The increased performance can be attributed to an architectural feature of Google Compute Engine where instances with four or more vCPUs do not share hard disks with other instances. This leads to faster disk I/O and faster performance for disk bound activities since there are no "noisy neighbor" instances to contend with. Understanding internal data I/O is as crucial as grasping external data I/O. Moving and storing data are key in a cloud environment because this will be the input used to process and deliver insights from the data for organizations. In the "Experiment Setup" section, we described multiple ways of transferring data between the cloud-based Hadoop cluster and Google Cloud Storage. These same data-flow methods are generally available on other cloud platforms, connecting to their respective distributed data stores. From our experiment, we saw that the Google Cloud Storage connector for Hadoop offers better performance for moving input and output data between MapReduce jobs and Google Cloud Storage than other data-flow methods. Understanding the key differences in data flows and available connectors is vital for being able to select the one that best performs and fits the needs of the workload, resulting in better performance.

Choosing the cloud architecture that best supports the needs of the business guarantees a better price-performance ratio than bare-metal clusters. Google Compute Engine's architecture offers performance advantages while providing a broad mix of instances to choose from in a competitive landscape.

Operator usability

Cloud platforms are limited by the tools given to operators; different platforms target operators with varying skillsets. Google Cloud Platform is more similar to bare-metal clusters than other cloud providers in performance, control, and accessibility. This results in performance advantages such as owning a disk within an instance but also includes fast instance start times, typically less than 30 seconds for most instances. The decrease in instance start time allows operators to do more in less time, getting results faster. The added level of control allows for user-defined DNS names of instances, which makes management much simpler, allowing users to refer to and access instances with a name of, say, "hadoop-master" instead of a string of random numbers and letters.

These features give operators much more control over the systems, allowing operators to "roll their own" code when customizing clusters. Depending on the operator, this may or may not be a desirable feature. In our testing, we had to customize provided deployment scripts to also install our own necessary tools like Starfish for automated performance tuning and Ganglia cluster monitoring for troubleshooting. Other platforms include a bootstrap framework that can install Ganglia and other tools as the cluster is being configured and started with a single additional command. As the Google Compute Engine platform matures, this is sure to be an option.

Once clusters are provisioned, managing the workflow can be a challenging task for operators as they build clusters, run workloads, and destroy clusters in an effort to create more useful data. Currently, a workflow-management tool is not available on Google Compute Engine, in contrast to other cloud providers. However, the provided deployment scripts and tools allow for custom scripting to automate many tasks.

Conclusion

Through our study, we conducted a price-performance comparison of a bare-metal Hadoop cluster and cloud-based Hadoop clusters. Using the TCO model we developed, we created eight different cloud-based Hadoop clusters utilizing four virtual machine instance types each with two data-flow models to compare against our bare-metal Hadoop cluster. The Accenture Data Platform Benchmark provided us with three real-world Hadoop applications to compare the execution-time performance of these clusters.

Results of this study reinforce our original findings. First, cloud-based Hadoop deployments—Hadoop on the cloud and Hadoop-as-a-Service—offer better price-performance ratios than bare-metal clusters. Second, the benefit of performance tuning is so huge that cloud's virtualization layer overhead is a worthy investment as it expands performance-tuning opportunities. Third, despite the sizable benefit, the performance-tuning process is complex and time-consuming and thus requires automated tuning tools.

In addition to our original findings, we were able to observe the performance impact of data locality and remote storage within the cloud. While counterintuitive, our experiments prove that using remote storage to make data highly available outperforms local disk HDFS relying on data locality.

Choosing a cloud-based Hadoop deployment depends on the needs of the organization: Hadoop on the cloud offers more control of Hadoop clusters, while Hadoop-as-a-Service offers simplified operation. Once a deployment model has been selected, organizations should consider these four key areas when selecting a cloud provider: workload utilization and demands, pricing structure, cloud architecture, and operator usability. Careful consideration of these areas will ensure that businesses are successful and are able to maximize their performance on the cloud.

References

1. "Where to Deploy Your Hadoop Clusters?" Accenture, June 2013. http://www.accenture.com/us-en/Pages/insight-hadoop-deployment-comparison.aspx

2. David J. Cappuccio, "Use a TCO Model to Estimate the Costs of Your Data Center," June 2012, Gartner.

3. Hadoop tolerates failures but does not necessarily fix the failures.

4. Jamie K. Guevara, et al., "IT Key Metrics Data 2013: Key Infrastructure Measures: Linux Server Analysis: Current Year," December 2012, Gartner.

5. "Compute Engine Disks: Price, Performance and Persistence," Google, December 2013. https://cloud.google.com/developers/articles/compute-engine-disks-price-performance-and-persistence

6. http://www.metascale.com

7. Standard replication factor for HDFS is 3.

8. The following discussion is based on information available at the time of publication.

Contact

Michael E. Wendt
R&D Associate Manager –
Accenture Technology Labs
michael.e.wendt@accenture.com

About Accenture

About Accenture Technology Labs

mc587