White Paper

Big Data Analytics

Extract, Transform, and Load Big Data with Apache Hadoop*

ABSTRACT

Over the last few years, organizations across public and private sectors have made a strategic decision to turn big data into competitive advantage. The challenge of extracting value from big data is similar in many ways to the age-old problem of distilling business intelligence from transactional data. At the heart of this challenge is the process used to extract data from multiple sources, transform it to fit your analytical needs, and load it into a data warehouse for subsequent analysis, a process known as "Extract, Transform & Load" (ETL). The nature of big data requires that the infrastructure for this process can scale cost-effectively. Apache Hadoop* has emerged as the de facto standard for managing big data. This whitepaper examines some of the platform hardware and software considerations in using Hadoop for ETL.

– We plan to publish other white papers that show how a platform based on Apache Hadoop can be extended to support interactive queries and real-time predictive analytics. When complete, these white papers will be available at http://hadoop.intel.com.

The ETL Bottleneck in Big Data Analytics

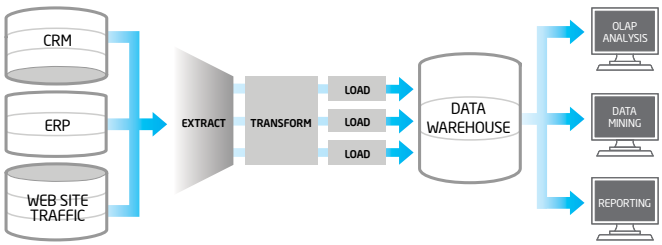
Big Data refers to the large amounts, at least terabytes, of poly-structured data that flows continuously through and around organizations, including video, text, sensor logs, and transactional records. The business benefits of analyzing this data can be significant. According to a recent study by the MIT Sloan School of Management, organizations that use analytics are twice as likely to be top performers in their industry as those that don't.[1]

Business analysts at a large company such as Intel, for example, with its global market and complex supply chain, have long sought insight into customer demand by analyzing far-flung data points culled from market information and business transactions. Increasingly, the data we need is embedded in economic reports, discussion forums, news sites, social networks, weather reports, wikis, tweets, and blogs, as well as transactions. By analyzing all the data available, decision-makers can better assess competitive threats, anticipate changes in customer behavior, strengthen supply chains, improve the effectiveness of marketing campaigns, and enhance business continuity.

Many of these benefits are not new to organizations that have mature processes for incorporating business intelligence (BI) and analytics into their decision-making. However, most organizations have yet to take full advantage of new technologies for handling big data. Put simply, the cost of the technologies needed to store and analyze large volumes of diverse data has dropped, thanks to open source software running on industry-standard hardware. The cost has dropped so much, in fact, that the key strategic question is no longer what data is relevant, but rather how to extract the most value from all the available data.

Rapidly ingesting, storing, and processing big data requires a cost-effective infrastructure that can scale with the amount of data and the scope of analysis. Most organizations with traditional data platforms—typically relational database management systems (RDBMS) coupled to enterprise data warehouses (EDW) using ETL tools—find that their legacy infrastructure is either technically incapable or financially impractical for storing and analyzing big data.

A traditional ETL process extracts data from multiple sources, then cleanses, formats, and loads it into a data warehouse for analysis. When the source data sets are large, fast, and unstructured, traditional ETL can become the bottleneck, because it is too complex to develop, too expensive to operate, and takes too long to execute.



By most accounts, 80 percent of the development effort in a big data project goes into data integration and only 20 percent goes toward data analysis. Furthermore, a traditional EDW platform can cost upwards of USD 60K per terabyte. Analyzing one petabyte—the amount of data Google processes in 1 hour—would cost USD 60M. Clearly "more of the same" is not a big data strategy that any CIO can afford. So, enter Apache Hadoop.

Apache Hadoop for Big Data

When Yahoo, Google, Facebook, and other companies extended their services to web-scale, the amount of data they collected routinely from user interactions online would have overwhelmed the capabilities of traditional IT architectures. So they built their own. In the interest of advancing the development of core infrastructure components rapidly, they published papers and released code for many of the components into open source. Of these components, Apache Hadoop has rapidly emerged as the de facto standard for managing large volumes of unstructured data.

Apache Hadoop is an open source distributed software platform for storing and processing data. Written in Java, it runs on a cluster of industry-standard servers configured with direct-attached storage. Using Hadoop, you can store petabytes of data reliably on tens of thousands of servers while scaling performance cost-effectively by merely adding inexpensive nodes to the cluster.

Central to the scalability of Apache Hadoop is the distributed processing framework known as **MapReduce** (Figure 1). MapReduce helps programmers solve data-parallel problems for which the data set can be sub-divided into small parts and processed independently. MapReduce is an important advance because it allows ordinary developers, not just those skilled in high-performance computing, to use parallel programming constructs without worrying about the complex details of intra-cluster communication, task monitoring, and failure handling. MapReduce simplifies all that.

The system splits the input data-set into multiple chunks, each of which is assigned a **map** task that can process the data in parallel. Each map task reads the input as a set of (key, value) pairs and produces a transformed set of (key, value) pairs as the output. The framework shuffles and sorts outputs of the **map** tasks, sending the intermediate (key, value) pairs to the **reduce** tasks, which group them into final results. MapReduce uses **JobTracker** and **TaskTracker** mechanisms to schedule tasks, monitor them, and restart any that fail.

The Apache Hadoop platform also includes the **Hadoop Distributed File System** (HDFS), which is designed for scalability and fault-tolerance. HDFS stores large files by dividing them into blocks (usually 64 or 128 MB) and replicating the blocks on three or more servers. HDFS provides APIs for MapReduce applications to read and write data in parallel. Capacity and performance can be scaled by adding **Data Nodes**, and a single **NameNode** mechanism manages data placement and monitors server availability. HDFS clusters in production use today reliably hold petabytes of data on thousands of nodes.

Hadoop Architecture

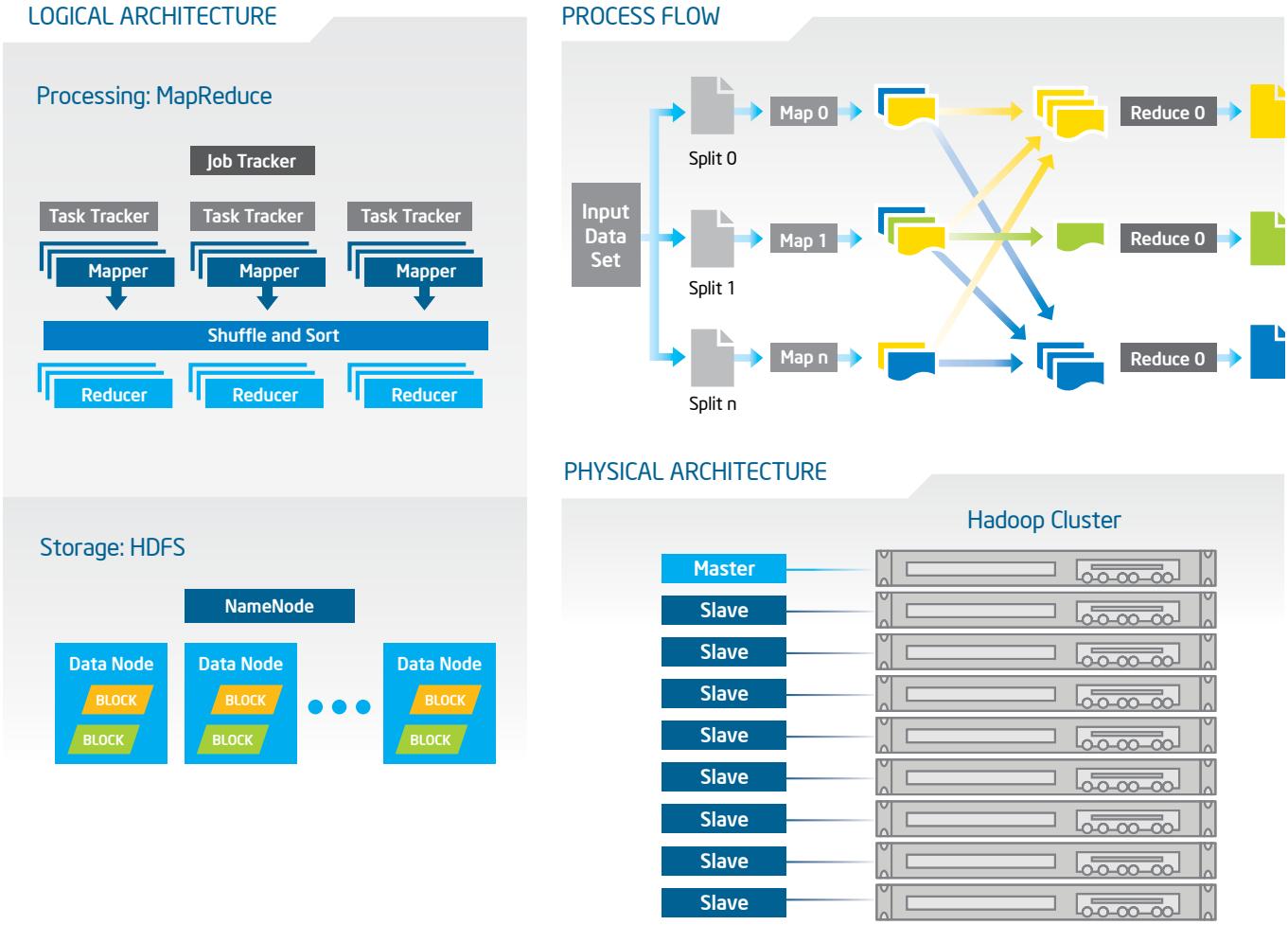


Figure 1. MapReduce, the programming paradigm implemented by Apache Hadoop, breaks-up a batch job into many smaller tasks for parallel processing on a distributed system. HDFS, the distributed file system stores the data reliably.

In addition to MapReduce and HDFS, Apache Hadoop includes many other components, some of which are very useful for ETL.

- **Apache Flume*** is a distributed system for collecting, aggregating, and moving large amounts of data from multiple sources into HDFS or another central data store. Enterprises typically collect log files on application servers or other systems and archive the log files in order to comply with regulations. Being able to ingest and analyze that unstructured or semi-structured data in Hadoop can turn this passive resource into a valuable asset.
- **Apache Sqoop*** is a tool for transferring data between Hadoop and relational databases. You can use Sqoop to import data from a MySQL or Oracle database into HDFS, run MapReduce on the data, and then export the data back into an RDBMS. Sqoop automates these processes, using MapReduce to import and export the data in parallel with fault-tolerance.
- **Apache Hive* and Apache Pig*** are programming languages that simplify development of applications employing the MapReduce framework. HiveQL is a dialect of SQL and supports a subset of the syntax. Although slow, Hive is being actively enhanced by the developer community to enable low-latency queries on Apache HBase* and HDFS. Pig Latin is a procedural programming language that provides high-level abstractions for MapReduce. You can extend it with User Defined Functions written in Java, Python, and other languages.
- **ODBC/JDBC Connectors** for HBase and Hive are often proprietary components included in distributions for Apache Hadoop software. They provide connectivity with SQL applications by translating standard SQL queries into HiveQL commands that can be executed upon the data in HDFS or HBase.

Hadoop is a powerful platform for big data storage and processing. However, its extensibility and novelty renew questions around data integration, data quality, governance, security, and a host of other issues that enterprises with mature BI processes have long taken for granted. Despite the many challenges of integrating Hadoop into a traditional BI environment, ETL has proven to be a frequent use case for Hadoop in enterprises. So what explains its popularity?

ETL, ELT, and ETLT with Apache Hadoop

ETL tools move data from one place to another by performing three functions:

- **Extract data from sources such as ERP or CRM applications.** During the extract step, you may need to collect data from several source systems and in multiple file formats, such as flat files with delimiters (CSV) and XML files. You may also need to collect data from legacy systems that store data in arcane formats no one else uses anymore. This sounds easy, but can in fact be one of the main obstacles in getting an ETL solution off the ground.
- **Transform that data into a common format that fits other data in the warehouse.** The transform step may include multiple data manipulations, such as moving, splitting, translating, merging, sorting, pivoting, and more. For example, a customer name might be split into first and last names, or dates might be changed to the standard ISO format (e.g., from 07-24-13 to 2013-07-24). Often this step also involves validating the data against data quality rules.
- **Load the data into the data warehouse for analysis.** This step can be done in batch processes or row by row, more or less in real time.

In the early days, before ETL tools existed, the only way to integrate data from different sources was to hand-code scripts in languages such as COBOL, RPG, and PL/SQL. Antiquated though it seems, about 45 percent of all ETL work today is still done by such hand-coded programs. Even though they are error-prone, slow to develop, and hard to maintain, they have loyal users who seem impervious to the charms of ETL tools, such as Oracle Warehouse Builder,* that can generate code for databases.

Code generators also have limitations, since they work with only a limited set of databases and are often bundled with them. In contrast, the next generation of ETL tools includes a general-purpose engine that performs the transformation and a set of metadata that stores the transformation logic. Because engine-based ETL tools such as Pentaho Kettle* and Informatica Powercenter* are independent of source and target data stores, they are more versatile than code-generating tools.

A traditional ETL architecture (Figure 2) accommodates multiple ETL iterations as well as an intermediate step, performed in the "staging area," that gets data out of the source systems as quickly as possible. A staging area may use a database or just plain CSV files, which makes the process faster than inserting data into a database table. Additional ETL iterations may be implemented to transfer data from the EDW into data marts, which support specific analytic purposes and end-user tools.

Much has changed in data warehousing over the past two decades. Chiefly, databases have become vastly more powerful. RDBMS engines now support complex transformations in SQL, including in-database data mining, in-database data quality validation, cleansing, profiling, statistical algorithms, hierarchical and drill-down functionality, and much more. It has become more efficient to perform most types of "transformation" within the RDBMS engine.

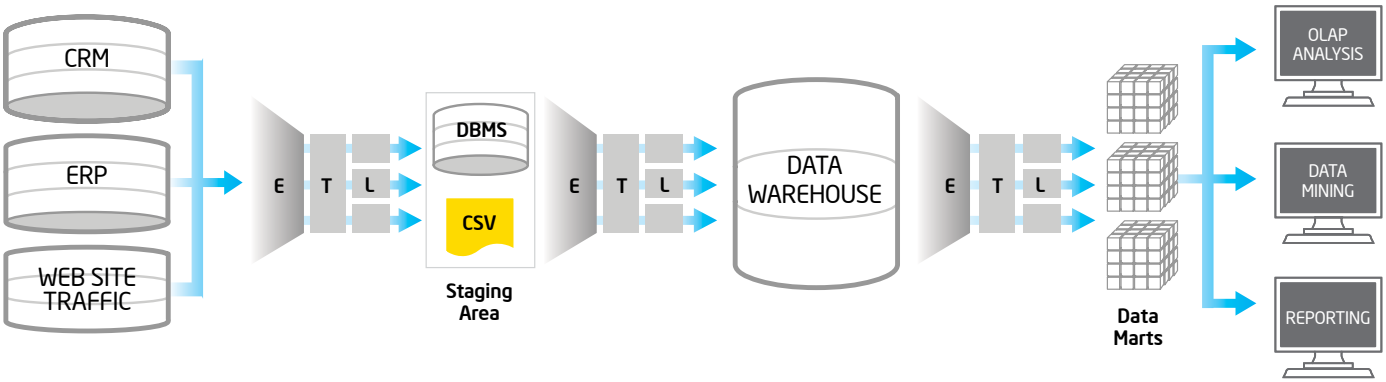


Figure 2. The traditional ETL architecture has served enterprises well for the past 20 years and many deployments still use it.

As a result, ELT emerged as an alternative approach in which data is extracted from the sources, loaded into the target database, and then transformed and integrated into the desired format. All the heavy data processing takes place inside the target database. The advantage of this approach is that a database system is better suited for handling large workloads for which hundreds of millions of records need to be integrated. RDBMS engines are also optimized for disk I/O, which increases throughput. And, as long as the RDBMS hardware scales up, the system performance scales with it.

But ETL is not dead yet. Traditional ETL vendors have enhanced their tools with pushdown SQL capabilities in which transformation can take place either in the ETL engine (for operations not supported by the target database) or after loading inside the database. The result is the ETLT approach, supported equally well by leading database vendors such as Microsoft (SQL Server Integration Services) and Oracle (Oracle Warehouse Builder).

None of these solutions is cheap or simple, and their cost and complexity are compounded by big data. Consider eBay, which in 2011 had over 200 million items for sale, separated into 50,000 categories, and bought and sold by 100 million registered users—all of which entailed roughly 9 petabytes of data. Google reportedly processes over 24 petabytes of data per day. AT&T processes 19 petabytes through their networks each day, and the video game World of Warcraft uses 1.3 petabytes of storage. All of these figures are already out-of-date as of this writing because online data is growing so fast.

Under these circumstances, Hadoop brings at least two major advantages to traditional ETLT:

- **Ingest massive amounts of data without specifying a schema on write.** A key characteristic of Hadoop is called "no schema-on-write," which means you do not need to pre-define the data schema before loading data into Hadoop. This is true not only for structured data (such as point-of-sale transactions, call detail records, general ledger transactions, and call center transactions), but also for unstructured data (such as user comments, doctor's notes, insurance claims descriptions, and web logs) and social media data (from sites such as Facebook, LinkedIn, Pinterest, and Twitter). Regardless of whether your incoming data has explicit or implicit structure, you can rapidly load it as-is into Hadoop, where it is available for downstream analytic processes.
- **Offload the transformation of raw data by parallel processing at scale.** Once the data is in Hadoop (on a Hadoop-compatible file system), you can perform the traditional ETL tasks of cleansing, normalizing, aligning, and aggregating data for your EDW by employing the massive scalability of MapReduce.

Hadoop allows you to avoid the transformation bottleneck in your traditional ETLT by off-loading the ingestion, transformation, and integration of unstructured data into your data warehouse (Figure 3). Because Hadoop enables you to embrace more data types than ever before, it enriches your data warehouse in ways that would otherwise be infeasible or prohibitive. Because of its scalable performance, you can significantly accelerate the ETLT jobs. Moreover, because data stored in Hadoop can persist over a much longer duration, you can provide more granular, detailed data through your EDW for high-fidelity analysis.

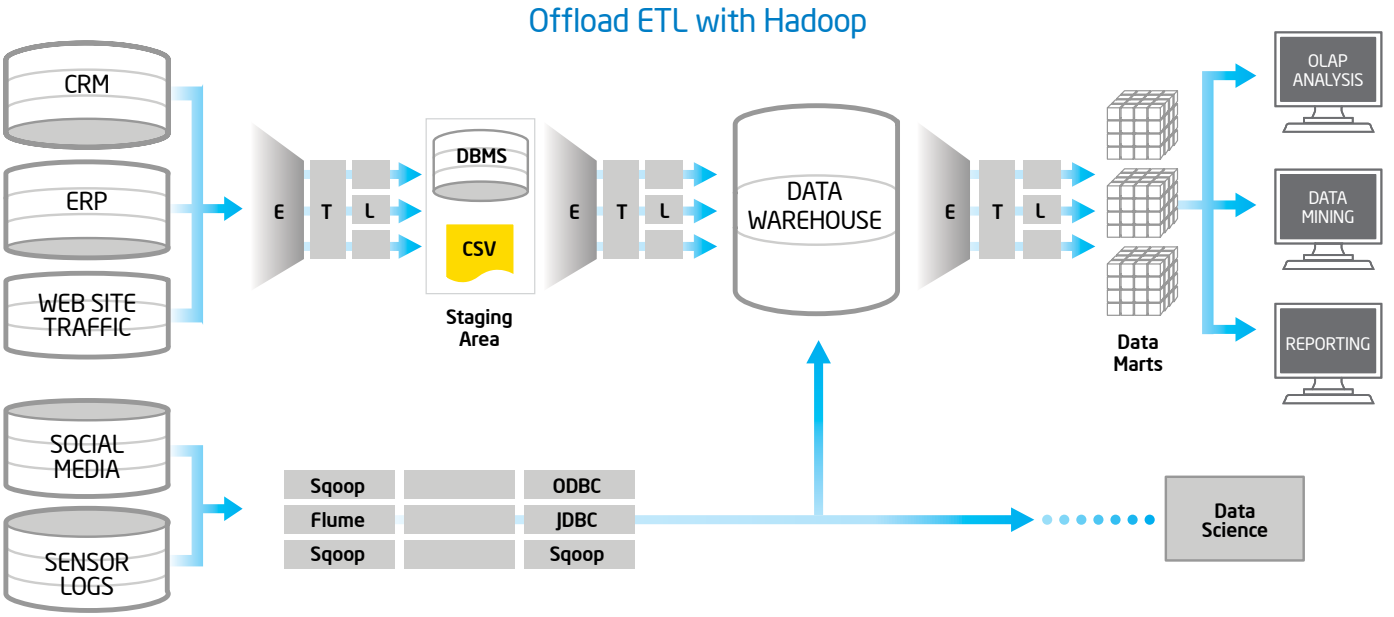


Figure 3. Using Apache Hadoop,* organizations can ingest, process, and export massive amounts of diverse data at scale.

Using Hadoop in this way, the organization gains an additional ability to store and access data that they "might" need, data that may never be loaded into the data warehouse. For example, data scientists might want to use the large amounts of source data from social media, web logs, or third-party stores (from curators, such as data.gov) stored on Hadoop to enhance new analytic models that drive research and discovery. They can store this data cost effectively in Hadoop, and retrieve it as needed (using Hive or other analytic tools native to the platform), without affecting the EDW environment.

Regardless of whether your enterprise takes the ETL, ELT, or ETLT approach to data warehousing, you can reduce the operational cost of your overall BI/DW solution by offloading common transformation pipelines to Apache Hadoop and using MapReduce on HDFS to provide a scalable, fault-tolerant platform for processing large amounts of heterogeneous data.

Choosing the Physical Infrastructure for ETL with Hadoop

The rule of thumb for Hadoop infrastructure planning has long been to "throw more nodes at the problem." This is a reasonable approach when the size of your cluster matches a web-scale challenge, such as returning search results or personalizing web pages for hundreds of millions of online shoppers. But a typical Hadoop cluster in an enterprise has around 100 nodes and is supported by IT resources that are significantly more constrained than those of Yahoo! and Facebook. Organizations managing these clusters must adhere to the capacity planning and performance tuning processes typical of other IT infrastructures. Sooner or later, they need to provision, configure, and tune their Hadoop cluster for the specific workloads their businesses run.

Workloads vary widely, so it is important to select and configure compute, storage, network, and software infrastructure to match specific requirements. But before considering each of these issues, let's examine the common misconception that all Hadoop workloads are I/O-bound. For the past 3 years, developers at Intel have been testing the performance of successive releases of Hadoop on successive generations of Intel processor-based servers using a reference set of test workloads. Many of these workloads, in aggregate, resemble real-world applications, such as ETL and analytics. Based on in-depth instrumentation of the test clusters, we noticed that I/O and CPU utilization vary significantly across workloads and within the stages of MapReduce in each workload.

- **TeraSort** *(Map: CPU Bound; Reduce: I/O Bound)* TeraSort transforms data from one representation to another. Since the size of data remains the same from input through shuffle to output, TeraSort tends to be I/O bound. However, when we compress map output so as to minimize disk and network I/O during the shuffle phases, TeraSort shows very high CPU utilization and moderate disk I/O during the map and shuffle phases, and moderate CPU utilization and heavy disk I/O during the reduce phase.
- **WordCount** *(CPU Bound)* WordCount extracts a small amount of interesting information from a large data set, which means that the map output and the reduce output are much smaller than the job input. As a result, the WordCount workload is mostly CPU bound (especially during the map stage), with high CPU utilization and light disk and network I/O.
- **Nutch Indexing** *(Map: CPU Bound; Reduce: I/O Bound)* The input to this workload is about 2.4 million web pages generated by crawling Wikipedia. Nutch Indexing decompresses the crawl data in the map stage, which is CPU bound, and converts intermediate results to inverted index files in the reduce stage, which is I/O bound.
- **PageRank Search** *(CPU Bound)* The Page Rank workload is representative of the original inspiration for Google search based on the PageRank algorithm for link analysis. This workload spends most of its time on iterations of several jobs, and these jobs are generally CPU bound, with low to medium disk I/O and memory utilization.
- **Bayesian Classification** *(I/O Bound)* This workload implements the trainer part of the Naïve Bayesian classifier, a popular algorithm for knowledge discovery and data mining. The workload contains four chained Hadoop jobs, which are all mostly disk I/O bound, except for the map tasks of the first job, which also have high CPU utilization and happen to be the most time consuming.
- **K-means Clustering** *(CPU Bound in iteration; I/O Bound in clustering)* This workload first computes the centroid of each cluster by running a Hadoop job iteratively until iterations converge or the maximum number of iterations is reached. This is CPU-bound. After that, it runs a clustering job (I/O bound) that assigns each sample to a cluster.

Even a cursory scan of various representative workloads shows that the system resource utilization of Hadoop is more critical for enterprises than it has been for early adopters and more complex than many have assumed. Intel has worked closely with a number of customers to help develop and deploy a balanced platform for various real-world deployments of Hadoop. Our assessments and benchmarking efforts have led us to make several recommendations when customers consider infrastructure hardware and distributions of Apache Hadoop software.

Compute

The Intel® Xeon® processor E5 family provides a strong foundation for many Hadoop workloads. A number of features built into the Intel Xeon processor E5 family are particularly well suited for Hadoop. One feature is Intel® Integrated I/O, which reduces I/O latency by up to 32 percent and increases I/O bandwidth by as much as 2x.[2,3] Another is Intel® Data

Direct I/O Technology (DDIO), which allows Intel® Ethernet adapters to communicate directly with processor cache, rather than only with main memory. This feature delivers more I/O bandwidth and lower latency, which is particularly beneficial when processing large data sets. Another feature that has captured the attention of Hadoop users is Intel® Advanced Encryption Standard New Instructions (AES-NI), which accelerates common cryptographic functions to erase the performance penalty typically associated with encryption and decryption of big-data files.

In several surveys conducted by Gartner, more than 70 percent of CIOs reported that power and cooling issues were the largest challenge they faced in the data center.[4] Maximizing energy-efficiency can be particularly important for Hadoop clusters, which grow with data volume.

With its system on a chip (SoC) design and power envelopes as low as 6 watts, the Intel® Atom™ processor offers better density and energy-efficiency for some workloads. Because Intel Atom processors are based on the industry-standard x86 instruction set, they are compatible with all application code that runs on Intel Xeon processors. This uniformity helps to reduce total cost of ownership by reducing software development, porting, and system management costs.

In addition, Intel offers a power and thermal management software product called Intel® Data Center Manager (Intel® DCM). Intel DCM uses the instrumentation in Intel Xeon and upcoming Intel Atom processors and integrates with existing management consoles using standard APIs. You can use it to monitor power and thermal data in real time for individual servers and blades, and also for racks, rows, and logical server groupings. You can cap power using set limits or workload-related policies, configure alerts for power and thermal events, and store and analyze data to improve capacity planning.

Memory

Sufficient system memory is essential for high-throughput of large numbers of parallel MapReduce tasks. Hadoop typically requires 48 GB to 96 GB of RAM per server, and 64 GB is optimal in most cases. Always balance server memory across available memory channels to avoid memory-bandwidth bottlenecks.

Memory errors are one of the most common causes of server failure and data corruption, so error-correcting code (ECC) memory is highly recommended.[5] ECC is supported in all servers based on Intel Xeon processors and in micro-servers based on the Intel Atom processor family.

Storage

Each server in a Hadoop cluster requires a relatively large number of storage drives to avoid I/O bottlenecks. Two hard drives per processor core typically deliver good results. However, a single solid-state drive (SSD) per core can deliver higher I/O throughput, reduced latency, and better overall cluster performance. Intel® SSD 710 Series SATA SSDs provide significantly faster read/write performance than mechanical hard drives, and this extra performance can be valuable for latency-sensitive MapReduce applications. It can also accelerate jobs as intermediate result files are shuffled between the map and the reduce phase.

Intel tests show that replacing mechanical drives with Intel SSDs can increase performance by as much as 80 percent.[6] You can also use mechanical drives and SSDs together, using Intel® Cache Acceleration Software. This tiered storage model provides some of the performance benefits of SSDs at a lower acquisition cost.

If you use hard drives, 7,200 RPM SATA drives provide a good balance between cost and performance. Run the drives in the Advanced Host Controller Interface (AHCI) mode with Native Command Queuing (NCQ) enabled to improve performance when multiple read/write requests are invoked simultaneously. Although you can use RAID 0 to logically combine smaller drives into a larger pool, Hadoop automatically orchestrates data provisioning and redundancy across nodes, so RAID is not recommended.

Network

A fast network not only allows data to be imported and exported quickly, but can also improve performance for the shuffle phase of MapReduce applications.

A 10 Gigabit Ethernet network provides a simple, cost-effective solution. Intel tests have shown that using 10 Gigabit Ethernet rather than 1 Gigabit Ethernet in a Hadoop cluster can improve performance for key operations by up to 4x when using conventional hard drives. The performance improvement is even greater when using SSDs—up to 6x.[7] The greater improvement with SSDs can be attributed to faster writes into the storage subsystem.

As a Hadoop cluster grows to include multiple server racks, you can scale network performance by connecting each of the 10 Gigabit Ethernet rack-level switches to a 40 Gigabit Ethernet cluster-level switch. As requirements continue to increase, you can interconnect multiple cluster-level switches and add an uplink to a higher-level switching infrastructure.

Software

Apache Hadoop is open source software that is freely available from the apache.org source code repository. A number of early adopters and test deployments directly download the source and build their platform based on the Apache Hadoop distribution. Enterprises and other organizations that need a vendor-supported platform, however, look to one of several independent software vendors (ISVs) to provide a complete product with software, updates, and services.

The Intel® Distribution for Apache Hadoop software (Intel® Distribution) is an enterprise-grade software platform that includes Apache Hadoop along with other software components. The Intel Distribution has a number of unique features.

- **Built from the silicon up for performance and security.** Running the Intel Distribution on Intel® processors enables Hadoop to fully utilize the performance and security features available in the x86 instruction set in general and the Intel Xeon processor in particular. For example, the Intel Distribution includes enhancements that take advantage of Intel AES-NI, available in Intel Xeon processors, to accelerate cryptographic functions, erasing the typical performance penalty of encryption and decryption of files in HDFS.
- **Automated performance tuning.** This integrated mechanism saves time while delivering a more optimized configuration. The Intel Distribution includes a management console that provides an intelligent mechanism for configuring the Hadoop cluster. The Intel Active Tuner uses a genetic algorithm to try various parameters and converge rapidly on the optimal configuration for a given Hadoop application.
- **Support for a wide range of analytic applications.** These applications come from leaders in the field, most of whom are already collaborating with Intel at the hardware layer. For example, Intel is working closely with Dell Kitenga, SAP, SAS, Revolution Analytics, and several other analytics ISVs to optimize the performance and scalability of the overall solution.

The Intel Distribution is sold with enterprise-grade support, training, and professional services at competitive prices. It is backed by a software team that has real-world experience with Hadoop deployments as well as deep expertise in the full platform stack of Apache Hadoop on Intel hardware. Download a free 90-day evaluation version of the software from http://hadoop.intel.com and contact us for help with complex proof-of-concept deployments.

Conclusion

The newest wave of big data is generating new opportunities and new challenges for businesses across every industry. The challenge of data integration—incorporating data from social media and other unstructured data into a traditional BI environment—is one of the most urgent issues facing CIOs and IT managers. Apache Hadoop provides a cost-effective and massively scalable platform for ingesting big data and preparing it for analysis. Using Hadoop to offload the traditional ETL processes can reduce time to analysis by hours or even days. Running the Hadoop cluster efficiently means selecting an optimal infrastructure of servers, storage, networking, and software.

Intel provides software as well as hardware platform components to help you design and deploy an efficient, high-performing Hadoop cluster optimized for big data ETL. Take advantage of Intel reference architectures, training, professional services, and technical support to speed up your deployment and reduce risk.

For more information visit

hadoop.intel.com | intel.com/bigdata | intel.com/microservers

[1] Source: MIT Sloan study; results published in MIT Sloan Management Review. "Big Data, Analytics and the Path From Insights to Value," Steve LaValle, Eric Lesser, Rebecca Shockley, Michael S. Hopkins, Nina Kruschwitz, December 21, 2010.

[2] Source: The claim of up to 32% reduction in I/O latency is based on Intel internal measurements of the average time for an I/O device read to local system memory under idle conditions for the Intel® Xeon® processor E5-2600 product family versus the Intel® Xeon® processor 5600 series.

[3] 8 GT/s and 128b/130b encoding in PCIe 3.0 specification enable double the interconnect bandwidth over the PCIe 2.0 specification. Source: http:www.pcisig.com/news_room/November_18_2010_Press_Release/.

[4] Source: Gartner 2013 CIO Survey. http://www.gartner.com/technology/cio/cioagenda.jsp

[5] A multi-year study by researchers from Google and the University of Toronto verified the importance of ECC memory. According to the team's report "memory errors are one of the most common hardware problems to lead to machine crashes." They found that DRAM errors occurred in about a third of systems and more than 8 percent of DIMMs. "The conclusion we draw is that error correcting codes are critical for reducing the large number of memory errors to a manageable number of uncorrectable errors." Source: "DRAM Errors in the Wild: A Large-Scale Field Study," by Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. www.cs.utoronto.ca/~bianca/papers/sigmetrics09.pdf

[6] TeraSort Benchmarks conducted by Intel in December 2012. Custom settings: mapred.reduce.tasks=100 and mapred.job.reuse.jvm.tasks=-1. http://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/terasort/package-summary.html For a detailed description of the tests, read the Intel white paper. http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/big-data-apache-hadoop-technologies-for-results-whitepaper.pdf

[7] TeraSort Benchmarks conducted by Intel in December 2012. Custom settings: mapred.reduce.tasks=100 and mapred.job.reuse.jvm.num.tasks=-1. Cluster configuration: One head node (name node, job tracker), 10 workers (data nodes, task trackers), Cisco Nexus* 5020 10 Gigabit switch. Performance measured using Iometer* with Queue Depth 32. Baseline worker node: SuperMicro SYS-1026T-URF 1U servers with two Intel® Xeon® processors X5690 @ 3.47 GHz, 48 GB RAM, 700 GB 7200 RPM SATA hard drives, Intel® Ethernet Server Adapter I350-T2, Apache Hadoop* 1.0.3, Red Hat Enterprise Linux* 6.3, Oracle Java* 1.7.0_05. Baseline storage: 700 GB 7200 RPM SATA hard drives, upgraded storage: Intel® Solid-State Drive 520 Series (the Intel® Solid-State Drive 520 Series is currently not validated for data center usage). Baseline network adapter: Intel® Ethernet Server Adapter I350-T2, upgraded network adapter: Intel® Ethernet Converged Network Adapter X520-DA2.Upgraded software in worker node: Intel® Distribution for Apache Hadoop* software 2.1.1. Note: Solid-state drive performance varies by capacity. More information: http://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/terasort/package-summary.html

Printed in USA 0713/DF/HBD/PDF Please Recycle 329261-001US