White Paper
April 2015

IT@Intel

How Intel IT Successfully Migrated to Cloudera Apache Hadoop*

From our original experience with Apache Hadoop software, Intel IT identified new opportunities to reduce IT costs and extend our BI capabilities.

Nghia Ngo
Big Data Capability Engineer, Intel IT

Sonja Sandeen
Big Data Product Manager, Intel IT

Darin Watson
Platform Engineer, Intel IT

Chandhu Yalla
Big Data Engineering Manager, Intel IT

Seshu Edala
Big Data Capability Engineer, Intel IT

Sinforoso Tolentino
Platform Engineer, Intel IT

Executive Overview

Intel IT values open-source-based, big data processing using Apache Hadoop* software. Until recently, we used the Intel® Distribution for Apache Hadoop (IDH) software to support our original three business intelligence (BI) big data use cases, and it delivered results worth millions of dollars to Intel. However, we realized we could benefit by migrating to the Cloudera Distribution for Apache Hadoop (CDH) software.

From our original experience with Apache Hadoop software, Intel IT identified new opportunities to reduce IT costs and extend our BI capabilities. Forming a strategic partnership with Cloudera, Intel IT adopted CDH, and migrated quickly from IDH. We used CDH enterprise-grade tools to improve performance, management, and ease of use for key Hadoop components.

This paper explains the benefits of migration for IT business groups and these six best practices developed by the Intel IT migration team:

- Do comparative evaluation in a sandbox environment
- Define the implementation strategy
- Split the hardware environment
- Upgrade the Hadoop version
- Create a preproduction-to-production pipeline
- Rebalance the data

Careful planning, thorough testing, proactive communications, and efficient management of scope, schedules, and skills helped enable a successful migration.

Contents

Acronyms

ACL	access control list
BI	business intelligence
CDH	Cloudera Distribution for Apache Hadoop
HDFS	Hadoop Distributed File System
IDH	Intel Distribution for Apache Hadoop
SLA	service-level agreement

Background

Intel IT sees value in open-source–based, big data processing using Apache Hadoop*. We used Intel's own Hadoop distribution, Intel® Distribution for Apache Hadoop (IDH) software.

We later determined that moving to the Cloudera Distribution for Apache Hadoop (CDH) offers significant advantages, shown in Figure 1. CDH contains the full distribution from the Apache Hadoop open-source project, along with key components such as the Apache Hive* data warehouse infrastructure and Apache Pig* data flow language. CDH also provides enterprise-grade management and software development tools to help developers and system administrators improve performance, management, and ease-of-use for the unique Hadoop components such as Cloudera Manager, Cloudera Impala, Navigator, and Sentry. The migration to CDH also allowed us to upgrade from Hadoop 1.0 to 2.0.

Best Practices for Migrating from IDH to CDH

A successful migration requires careful planning to cope with three key challenges:

- **Coping with Hadoop variations.** An unmodified Hadoop installation uses default configuration settings. These settings differ depending on the variations of Hadoop. Before the migration, a comparative analysis can highlight differences (see Best Practice 1: Find Differences with a Comparative Evaluation in a Sandbox Environment).
- **Understanding the scope of the changes.** We must minimize negative impacts from infrastructure changes on our application developers and the customers who use those applications. To invest and participate in such a conversion, business or customer teams expect to have the same experience and better.

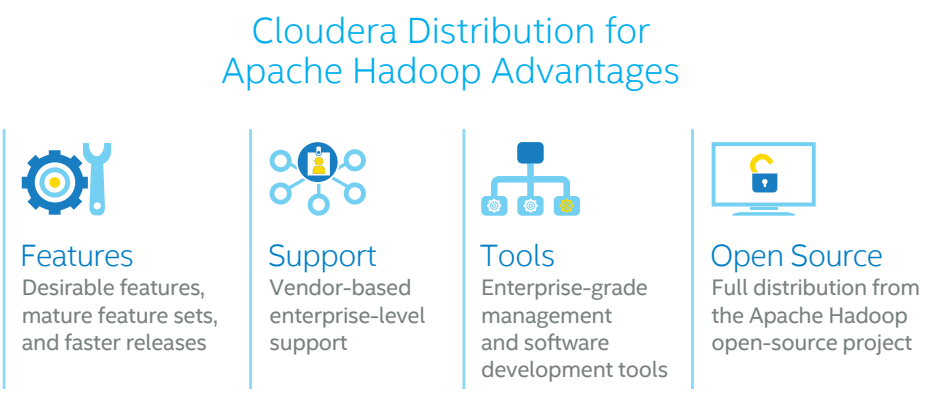


Figure 1. Cloudera Distribution for Apache Hadoop* (CDH) offers significant advantages.

- **Completing the in-place migration in a timely manner.** The migration required us to run IDH and CDH environments in parallel. Our goal was to operate the dual environments in tandem for the shortest possible time and to balance all functional parts between customer projects and platform changes. To do so, the migration team had to avoid idling racks or servers during data transfer and testing.

To address these three challenges, the Intel IT migration team followed six best practices.

Best Practice 1: Find Differences with a Comparative Evaluation in a Sandbox Environment

Using a test environment, or sandbox, to do a complete evaluation of IDH and CDH identified differences between the two environments while providing minimal disruption to operations. The migration team was able to study the proposed solution architecture of the various use cases to determine critical features. We compared features and capabilities, such as business intelligence (BI) frameworks, I/O containers, compression codecs, security and access control lists (ACLs), container allocation limits, and so on, isolating one configuration from another.

From the results, the team constructed a compare-and-contrast table that identified gaps and described key differentiators, as shown in Table 1.

Table 1. Comparison of Features between Cloudera Distribution for Apache Hadoop* (CDH) and Intel® Distribution of Apache Hadoop (IDH)
Note: The comparative study was performed from March to April 2014.

BETTER ON CDH | EQUAL BETWEEN CDH AND IDH | MISSING FROM CDH

CONNECTORS	Data Ingestion	BETTER
DATA STORAGE	Lustre* Integration	MISSING
	Online NoSQL	BETTER
OTHER SOFTWARE AND DRIVER INTEGRATION	Batch Processing	EQUAL
	Analytics SQL	BETTER
	Search	BETTER
	Machine Learning	MISSING
	Stream Processing	BETTER
	Third-party Apps	EQUAL
	Data Management	BETTER
	Workload Management	EQUAL
	System Management	EQUAL
INFRASTRUCTURE	High Availability	EQUAL
	Operating System	EQUAL
	Enterprise Access Management/ Microsoft Active Directory* Integration	EQUAL
	Security	MISSING

6 Best Practices for Migration

This table helped us to determine objectives for an orderly migration. We then consulted with Cloudera to develop a joint roadmap, asking them to state their commitments to implement missing or different capabilities. At this point, the team evaluated the cost of refactoring—that is, the amount of change, testing, and time it would take us to modify the architecture to gain the advantage of the design recommendations for Cloudera's distribution. We opted to make some design and architecture changes, and we selected only mandatory components of the distribution rather than adopting all available Hadoop components.

We made design and code changes to accommodate any feature differences. Take the simple example of support for the lossless compression codec. Using the less efficient CDH codec could impact both downstream code and disk, memory, or CPU capacity. We needed enough time to test new code that uses the CDH codec and evaluate the effects of additional resource usage – otherwise, we would have had a new roadmap gap that Cloudera would need to fill.

We considered differences between the test environment and the path to production for network and hardware as part of the calculations.

Create a Sandbox

In the sandbox environment, the team made sure that everything planned for the migration worked as we anticipated. From the existing IDH implementation, we took a baseline for functional, performance, and data testing benchmarks. Test data included transfer or copy methods as well as transfer speeds. We needed this information to determine implementation timing for the path to production. We considered differences between the test environment and the path to production for network and hardware as part of the calculations. The CDH implementation had to meet or beat prior parametric benchmarks and meet nonparametric criteria.

Use Abstraction Layers to Simplify the Migration

An abstraction layer containing vanity URLs, soft links, and high-availability proxies simplified the migration. For example, because the IDH libraries were not in the same place as the CDH libraries, we used soft links instead of hard coding the library locations.

Use Core Open-Source Hadoop Capabilities, and Minimize Customizations

We learned that most of the core ecosystem, such as Hive, Pig, and others, behaved the same on the CDH implementation as it did on IDH. The code migration had minimal or no impact on customers and their applications. From our own experience, and from what we have seen of customers developing their applications with Hive and Pig, these components required no code modifications to work seamlessly on the CDH environment. Only one project, which used Apache Mahout*, needed a small code change because the older MapReduce* API was deprecated in favor of the newer MapReduce API. Updating code to use the most current API makes any future conversions much easier.

As a result of the sandbox mapping and testing, the team determined that the roadmap was solid and had no gaps. At the least, our roadmap documented when any gaps would be filled.

Best Practice 2: Define Our Strategy for the Cloudera Implementation

After completing the sandbox testing, the team decided which options and key components to include in the initial migration and recorded the benefits and risks with choosing them. Critical options and components included security, user interfaces, APIs, development languages, and version compatibility. We planned to keep the overall environment simple during migration, maintaining stability before adding services or capabilities.

Our component and option selections validated our sandbox findings and were incorporated into our migration strategy. We summarize that strategy, along with benefits, potential risks, and current status, in Table 2.

The information in the table guided the strategy to migrate the Hive metastore service and metadata files. We needed to determine our timeframe for migration with consideration for impact on customers, changes to code, regression testing, and cycle run times.

One component that is easy to overlook during migration is the Hive metastore service. Hadoop provides compute abstraction with MapReduce API and storage abstraction with the Hadoop Distributed File System (HDFS) API. It is easy to focus on transitioning these core components and overlook an ancillary yet important component like the Hive metastore service, which relies on an external database like MySQL or Postgres and is critical for migrating data warehouse tables from one platform to another. The core Hadoop distribution does not provide tools to migrate the Hive metastore service data type, security, compression, ACLs, and extended table information. That part of the migration requires custom tooling (or scripts) to be able to capture, replicate, and migrate over to the new platform.

During the migration, when the environment was split into two parts, we communicated with customers about the possibility of platform slowness. We reset the service-level agreements (SLAs) for cycle processing and deferred any performance benchmarking activities.

Table 2. Considerations for the Migration from Intel® Distribution of Apache Hadoop* (IDH) to Cloudera Distribution for Apache Hadoop (CDH)

Scope	Migrate all Internal Big Data platforms and projects from IDH to CDH.
Strategy	• Split the hardware in half, install CDH, and run the environments in parallel during migration. • Align all customers to complete the testing in CDH and cutover. • Uninstall IDH and add the hardware to the CDH environment to get back to full capacity. • Focus only on the core Hadoop* components needed for current customers. New features and CDH capabilities will be adopted after cutover and environments are returned to full capacity.
Benefits	• Adopt CDH product and vendor support now and align to strategic initiatives. • Single regression testing effort of CDH migration and Hadoop 2.x/YARN. • Less resources now to test current scope of use cases. • Avoid large-scale conversion and significant regression testing of entire projects later. • Take advantage of CDH features sooner (Hue, HCatalog, Impala, Navigator, and Sentry).
Risks and Impacts	• Customers: Minimal risk and impact for testing and migration for two production projects based on initial and early testing. We will know more after receiving application testing results from Customer Insight and the Sales and Marketing Account Recommendation Tool. • Platform: Potential risk if customers delay migration to CDH so that environments remain split, requiring support, half capacity, and delayed start of new Cloudera feature evaluations.
Current Status	• Full migration from IDH to CDH completed in five weeks. • Currently running CDH 5.3 in production. • Completed two version upgrades since initial integration with zero impact to customers.

Split Hardware Environment Features

- Two 10GbE 48-port switches
- 352 cores using Intel® Xeon® processors
- Intel® Distribution of Hadoop 2.51 based on Hadoop Apache 1.x
- 22 data nodes equaling 243 TB (3-way replication)
- High availability with true rack awareness
- Database integration with Teradata, Netezza, and MS SQL
- Integration with Enterprise ETL, Active Directory/EAM, and ITSM

Best Practice 3: Split the Hardware Environment

The Intel IT migration team split the existing Hadoop hardware environment into two clusters (one for IDH and the other for CDH), completing the construction of the two parallel systems using available components, as shown in Figure 2.

We could not do an in-place migration for three reasons. First, we did not want to lose HDFS because it had large amounts of data distributed over the servers. Second, we wanted to move from Hadoop 1.0 to 2.0.

Third, IDH managed Hadoop differently from CDH; each administered clusters with different provisioning and management tools. Thus, we could not easily migrate in place from IDH to CDH without reprovisioning software and configurations. We needed to set up a CDH cluster on the existing hardware and migrate data from IDH to CDH.

Even if an in-place upgrade were possible, we would have needed to back up terabytes of data to another cluster and move it back onto to the CDH cluster. Since we would move data in any case, we built the new CDH cluster on the same network as the IDH cluster and planned to transfer the data directly from the old to the new Hadoop cluster. Parallel systems provided the cleanest and safest approach, keeping in mind that we were working with limited available hardware and we wanted to migrate the data only once.

This split worked out well for the following reasons:

- We had a limited amount of hardware, but we had enough additional compute and storage capacity to support the hardware split, allowing us to decommission nodes from IDH and build a CDH cluster sized to support data migrations.

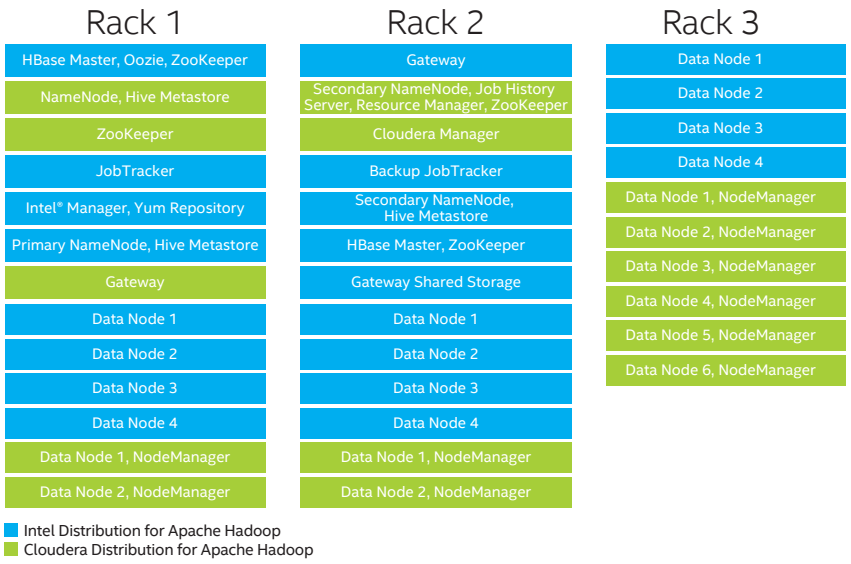


Figure 2. The hardware environment was split into two clusters, one for Intel® Distribution of Apache Hadoop* and the other for Cloudera Distribution for Apache Hadoop.

- By segregating the hardware, we remained operational during data transfer. Our IDH resources were sufficient for the customer workload, but we had to coordinate with projects to ensure minimal impact to existing SLAs and batch performance expectations through the migration process.
- With matching clusters, we built on one and ported to the other.

Minimize Impact on Users

Putting IDH and CDH on the same local network enabled fast data transfer from IDH to CDH. Allocating half of the available servers to each environment helped enable us to complete the migration with minimal negative impact on our customers' applications. Hadoop's distributed nature contributed to this approach.

We built the new CDH cluster on the same network as the IDH cluster and planned to transfer the data directly from the old to the new Hadoop cluster.

The Intel IT team offloaded the data onto a set of data nodes and then turned off hardware that we wanted to prepare for use in the new CDH environment. We decommissioned a few servers at a time, allowing the IDH cluster to redistribute the data to the remaining nodes. By managing the decommissioning process on the IDH cluster, we mitigated data loss and monitored impact on the cluster's workload. We added the freed-up servers from the decommissioned IDH cluster to the CDH cluster.

Building the CDH cluster on the same network as IDH and on the same racks enabled the data to transfer quickly from IDH to CDH. We used implicit failover and fault tolerance of the Hadoop framework to do a phased decommissioning and re-commissioning of cluster nodes between the parallel systems. Using a common network infrastructure made fewer hops between IDH and CDH nodes and leveraged 10G-bonded connections on all data transfer hosts, reducing the data transfer time between clusters and minimizing the impact to our existing IDH workload.

Best Practice 4: Upgrade the Hadoop Version

The move from Hadoop 1.0 to Hadoop 2.0 was simple and straightforward. The team gained desirable features that are not included in IDH, along with the advantages of an open-source platform. Those features include:

- High availability for the HDFS NameNode, which eliminates the previous single point of failure in HDFS and Hadoop distribution–specific, high-availability solutions.
- Support for file system snapshots in HDFS, which supplies native backup and disaster recovery processes to Hadoop.
- Support for federated NameNodes, which allows for horizontal scaling of the file system namespace.
- Support for NFS access to HDFS, which allows HDFS to be mounted as a standard file system.
- Support for Native network encryption, which secures data while in transit.
- Performance enhancements to HDFS.

Hadoop 2.0 Features

- HDFS NameNode High Availability
- HDFS File System Snapshots
- Federated NameNodes
- Performance Enhancements
- YARN

In addition to these features, the YARN resource management system accommodates a wider range of workloads by using one Hadoop cluster to serve emerging frameworks other than MapReduce, such as Spark, Impala, and HBase. This new flexibility expands the use cases for Hadoop, while improving the efficiency of certain types of processing over data already stored there.

Best Practice 5: Create a Preproduction-to-Production Pipeline

After splitting the physical environment into an IDH cluster and a CDH cluster, we then created a preproduction environment that would enable us to validate our processes. The preproduction environment was powered by 128 cores (using Intel® Xeon® processors) and included actual data volumes, as shown in Figure 3. These data volumes consisted of 8 data nodes equaling 160 TB (with 2-way replication).

Creating a preproduction environment enabled us to incrementally migrate data to the CDH environment and to start the data migration process early enough to allow for time for completion and a rapid transition to the new CDH environment.

In this preproduction environment, we took the following steps:

- We assessed the time required to move from IDH to CDH.
- We aligned entitlements, control servers, and users between the two systems without automation.
- To figure out the best methods for moving the data quickly to the new platform, we consulted with Cloudera on a number of topics: data compression standards, data security standards, data transfer standards, metadata backup and restore process, replication factors, user accessibility guidelines (via Hue), Impala memory allocation, and resource management guidelines.
- We identified data tiers, compression codecs, and replication factors that optimized the offline and online data transfers. Then, we communicated any necessary changes to Intel's developers of the STORE and LOAD functions, so we could adequately implement the codec and HDFS namespace changes.
- We used custom scripts for the Hive metastore service and ACL migration. We had to capture the user management, permissions, and partitioning schemes. We manually created the scripts offline before migrating the data.

One week after we implemented the system on the preproduction environment, we repeated these same tasks in the production environment. This process helped minimize the risk and impact to the development and bug-fix efforts that might result from different distributions on the preproduction and production environments.

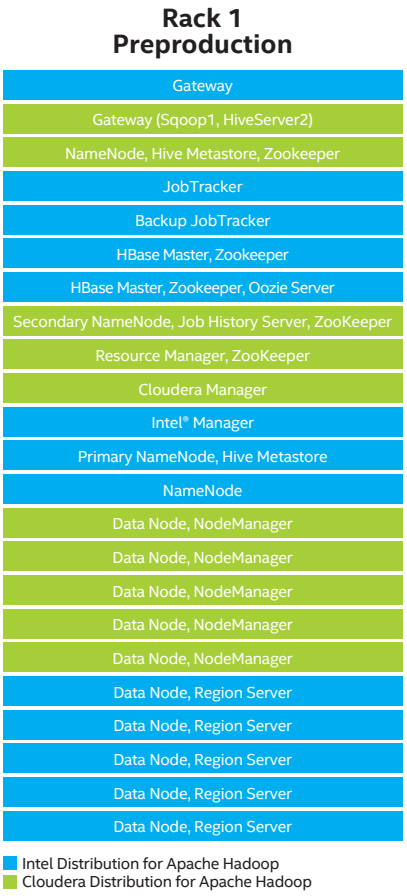
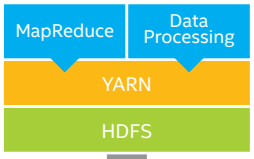


Figure 3. The preproduction environment enabled us to validate our processes. This environment was powered by 128 cores (using Intel® Xeon® processors) and eight data nodes populated with actual data volumes.

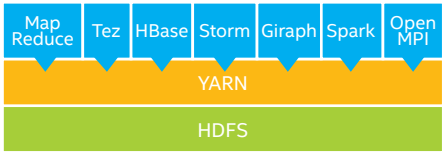
Hadoop's Maturity Path

2005-2012
Hadoop 1.0

2013-2014
Hadoop 2.0
Data processing runs natively



Next-Generation Hadoop
Applications run natively



Best Practice 6: Rebalance the Data

In the production environment, since we had two models of data-node-storage-capacity configurations, part of the migration process required continuous rebalancing of HDFS data across the cluster. HDFS has a utility to balance data block placement across data nodes evenly to help ensure optimal performance.

Mixing older, low-capacity nodes with newer, high-capacity nodes caused uneven data distribution. Low-capacity nodes filled up while data was loaded on the CDH cluster, initiating constant rebalance to redistribute the data from the low-capacity nodes to the high-capacity ones.

We took advantage of a rebalancing feature in CDH that can be executed from the Cloudera Manager user interface. After each rebalance and data migration process, the team reassessed the data distribution and rebalanced as necessary.

With limited nodes, we would have mismatched disk space capacity. Therefore, we paused any data loads during the migration.

Finally, the team documented any middleware or kernel changes and established the following migration best practices:

- Document the container limits for allocations compared to task trackers, HCatalog APIs for I/O between Pig, Hive, and MapReduce. Also document limitations of the Impala framework: UDFs, I/O formats, memory limitations, internal and external tables. These provide Spark guidance and affect positioning of Hive, Spark, and Impala as the data warehouse, bulk ETL, and in-memory analytics.
- Review and understand configuration differences and values that require altering. Some key changes that required tuning were related to resource allocations from MapReduce 1 to MapReduce 2/YARN[1].
- Stabilize the cluster with only necessary migration components prior to enabling additional services or capabilities.
- Use high-bandwidth network backbones to support data migrations to minimize impact of data transfer processes as well as adjusting distributed copy bandwidth configurations.
- Verify that network firewall requirements are configured to support the Hadoop distribution.
- Use DNS vanity names on edge or gateway nodes and key interfaces to reduce development impact.
- Coordinate projects to monitor and help ensure minimal impact to existing SLAs and batch performance expectations through the migration process.
- Make sure that the operating system's tuning parameters, permission requirements, and disk space requirements are in line with each Hadoop distribution's requirements.

[1] MapReduce 2 is an upgrade for scheduling, resource management, and execution in Hadoop. Read the blog for more information at blog.cloudera.com/blog/2013/11/migrating-to-mapreduce-2-on-yarn-for-users.

Conclusion

Intel IT gained valuable experience from this migration. Although this Hadoop migration was complex and involved multiple layers of change, the move was successful. We experienced little dependency on the application teams and did not encounter any significant issues associated with the migration from one distribution to another. Careful planning, thorough testing, proactive communications, and efficient management of scope, schedules, and skills helped us achieve a successful migration.

When performing a Hadoop migration, consider the following tips.

Choose the migration team carefully. Technology is important but software engineers with the right skills are essential. Migration projects can be very complicated, so planning ahead, thoroughly evaluating both products, and understanding the gaps are key to success. Form a small team that comprehends the total solution: the platforms, the code, and the ramifications for migrating data. Make sure the team can communicate, execute, and evaluate finite itemized changes in the lock-stepped mode necessary for a successful migration.

Gain from open-source options. Intel IT benefited from a Hadoop distribution that has open source as its core, not proprietary software. Staying with open-source software components helps minimize total cost of ownership. Starting small helps keep the focus on design and scalability for the platform.

Keep design as simple as possible. Intel IT prioritized integration of components and capabilities according to use cases, rather than thinking that all new features must be enabled. We maintained a balance of scope, timelines, testing, and resources. If it is not clear what features to enable, prioritize only those dictated by the current set of use cases. From there, the team can build and grow feature sets as needed. For our migration, technical, business, and modeling teams used an iterative discovery method to understand the data. The split environments helped simplify the data transfer. In general, keeping the design and migration processes simple was the key to our successful migration.

For more information on Intel IT best practices, visit www.intel.com/IT.

IT@Intel

We connect IT professionals with their IT peers inside Intel. Our IT department solves some of today's most demanding and complex technology issues, and we want to share these lessons directly with our fellow IT professionals in an open peer-to-peer forum.

Our goal is simple: improve efficiency throughout the organization and enhance the business value of IT investments.

Follow us and join the conversation:

- Twitter
- #IntelIT
- LinkedIn
- IT Center Community

Visit us today at **intel.com/IT** or contact your local Intel representative if you would like to learn more.

Related Content

Visit **intel.com/IT** to find content on related topics:

- How Intel Implemented a Low-Cost Big Data Solution in Five Weeks paper
- Intel IT Best Practices for Implementing Apache Hadoop* Software paper
- Integrating Apache Hadoop* into Intel's Big Data Environment paper