DataStax

Welcome to Apache Cassandra 1.0

An Overview for Architects, Developers, and IT Managers

WHITE PAPER

Welcome to Apache Cassandra 1.0

An Overview for Architects, Developers, and IT Managers

By DataStax Corporation

October 2011

Table of Contents

Introduction

For a number of years, modern businesses have been looking for an alternative to the legacy relational model for storing and accessing data. Information management requirements have undergone subtle to radical changes in these organizations, such that the practice of forcing the proverbial square peg into the round hole is no longer a viable solution.

The rising interest of using NoSQL databases in the enterprise has demonstrated that developers and CTOs alike have recognized the fact that many progressive companies are now using NoSQL to manage both operational and analytical data. Businesses that were early adopters of NoSQL already have progressed to the point where NoSQL is powering a number of their key production systems, with both new development and RDBMS replacement projects being run in parallel.

The release of Cassandra 1.0 in October 2011 is a significant milestone, both in the history of Cassandra itself and in the NoSQL data management movement in general. It is atypical for companies to run pre-1.0 software in production, yet because of a need for the features Cassandra offers, that is exactly what many businesses have been doing. With the release of 1.0, companies with software policies that negate the use of non-GA software can now also begin to enjoy the benefits Cassandra provides.

This paper provides an overview of Cassandra for new users, discusses why and where Cassandra is being used, describes a number of key enhancements that have gone into the product prior to the 1.0 release, and details a number of major new features that have just appeared in version 1.0.

A Basic Overview of Cassandra

What exactly is Cassandra and why are so many companies using it for their prominent applications? The following sections provide an overview of Cassandra for those unfamiliar with the database and describe the various application use cases and technology differentiators that make Cassandra a standout in the area of data management.

What Is Apache Cassandra?

Apache Cassandra™ is a highly scalable and high-performance distributed database management system that can serve as both an operational datastore (the “system of record”) for online/transactional applications, and as a read-intensive database for business intelligence systems. Cassandra is able to manage the distribution of data across multiple data centers and offers incremental scalability with no single points of failure.

Cassandra is a logical choice for enterprises that need high degrees of uptime, reliability, and very fast performance. Leading companies like Netflix, Twitter, Cisco, HP, Motorola, Rackspace, Ooyala, Openwave, and many more rely upon Cassandra to manage the data needs of their critical production applications.

Cassandra was originally incubated at Facebook and is based upon Google’s BigTable and Amazon’s Dynamo software. The end result is an extremely scalable and fault-tolerant data infrastructure that solves small to big data problems, handles write intensive user traffic, delivers sub-millisecond caching layer reads, and supports demanding workloads involving petabytes of data.

Why Cassandra?

A progressive database like Cassandra overcomes many of the limitations inherent in legacy RDBMSs. The relational model was designed primarily to handle complex and involved transactional operations, very structured and rigid data, moderate data volumes in a single location, and works very well for standard data input/output use cases.

However, many modern businesses have outgrown the typical RDBMS use case and are in need of data management software that offers more. Sharding was a stop-gap measure, but architectural limitations, and the management complexity it requires, make it unacceptable for many mainstream organizations. Successful web companies like Facebook, Yahoo, Google, and others like them, first exposed the need for a more forward-thinking method beyond sharding that managed all types of data, but it wasn’t long before that need became prevalent in nearly every industry.

“We spent a lot of time evaluating different databases for our viewership data which would extend our current analytics solution and allow us to scale as publishers deliver more video content. *Cassandra ended up being the best choice* and is allowing us to continue to build innovative products around analytics and monetization.”

—Edmond Lau, Technical Lead, Ooyala

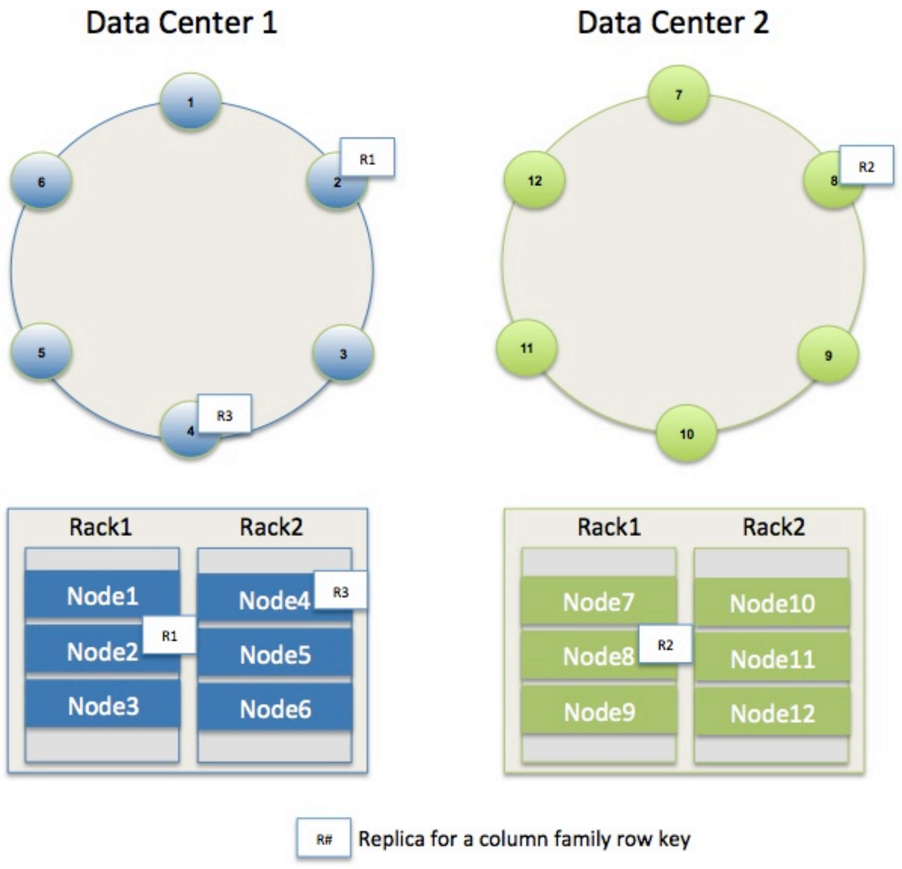
A Brief Look at the Cassandra Architecture

Cassandra can satisfy many data-driven application use cases through a carefully thought-out architecture designed to manage all forms of modern data, scale to meet the requirements of “big data” management, offer linear performance scale-out capabilities, and deliver the type of high availability that most every online, 24x7 application needs.

At its foundation, Cassandra is a peer-to-peer distributed data management system where every node is essentially the same with respect to how it functions in the cluster. In Cassandra, there is no concept of a “master node” or anything similar, with the benefit being derived that no single point of failure exists for any key process or function.

The scale-out aspect of Cassandra allows node additions to occur with no disruption to application uptime. Capacity to handle increasing I/O traffic or incoming data volumes is added easily and requires no special ETL processes or other data movement work to be performed manually. Instead, Cassandra automatically partitions data across nodes once one or more nodes have been added to a cluster and “seeds” the new nodes from existing machines in the cluster.

Data redundancy to protect against hardware failure and other data loss scenarios is also built into and managed transparently by Cassandra. Further, this capability can be configured to be quite sophisticated so data can be distributed across multiple, geographically dispersed data centers, between different physical racks in a data center, and between public cloud providers and on-premise managed data centers.



An administrator, architect, or developer only has to specify a replication and data-partitioning strategy. From there, Cassandra takes care of everything.

All nodes in the cluster communicate with each other through the gossip protocol. If a node goes down, the cluster detects the failure and automatically routes user requests away from the failed machine. Once the failed node is operational again, it rejoins the cluster, and its data is brought back up to date via the other nodes.

Application Use Cases

Because of its progressive architecture, Cassandra excels in the following application use cases:

- Serving as the operational datastore or system of record for web online applications or other systems needing around-the-clock transactional input capabilities
- Applications needing “network independence,” meaning systems that cannot worry about where data lives. This often equates to widely dispersed applications that need to serve numerous geographies with the same fast response times
- Applications needing extreme degrees of uptime and no single point of failure
- Retailing or other such systems that need easy data elasticity, so capacity can be added to service peak workloads for various periods of time and then shrink back when user traffic reduction allows – all done in an online fashion
- Write-intensive applications that must take in large volumes of data continuously (e.g. credit card systems, music download purchases, device/sensor data, web clickstream data, archiving systems, event logging)
- Real-time analysis of social media or similar data that requires tracking user activity, preferences, and so on
- Management of large data volumes (terabytes-petabytes) that must be kept online for query access and business intelligence processing
- Caching functionality that delivers caching tier performance response times without resorting to separate caching (e.g., memcached) and database tiers
- Software as a Service (SaaS applications that utilize web services to connect into a distributed, yet centrally managed database, and then display results to SaaS customers
- Cloud applications that require elastic data scale, easy deployment, and a need to grow through a data-centric, scale-out architecture
- Systems that need to store and directly deal with a combination of structured, unstructured, and semi-structured data, with a requirement for a flexible schema/data storage paradigm that allows for easy and online structure modifications

Technology Differentiators

Key technical differentiators of Cassandra over its RDBMS predecessors, as well as other NoSQL offerings, include the following:

- A built-for-scale architecture that can handle petabytes of information and thousands of concurrent users/operations per second as easily as it can handle data volumes and user traffic of lesser volume
- Online capacity additions that deliver linear performance gains for both read and write operations
- Read/write anywhere capabilities that equate to a truly network-independent method of storing and accessing data
- Tunable data consistency that allows Cassandra to offer comparable data durability and protection like an RDBMS, but with the flexible choice of relaxing that consistency when application use cases allow
- Flexible schema design that accommodates structured, semi-structured, and unstructured data
- Peer-to-peer design that offers no single point of failure for any database process or function
- Simplified replication that provides data redundancy and is capable of being multi-data center and cloud-focused in nature
- An SQL-like language (CQL) that lessens the learning curve for developers and administrators coming from the RDBMS world
- Support for key developer languages (e.g., Java) and operating systems
- No requirement for any special equipment; Cassandra runs on commodity hardware

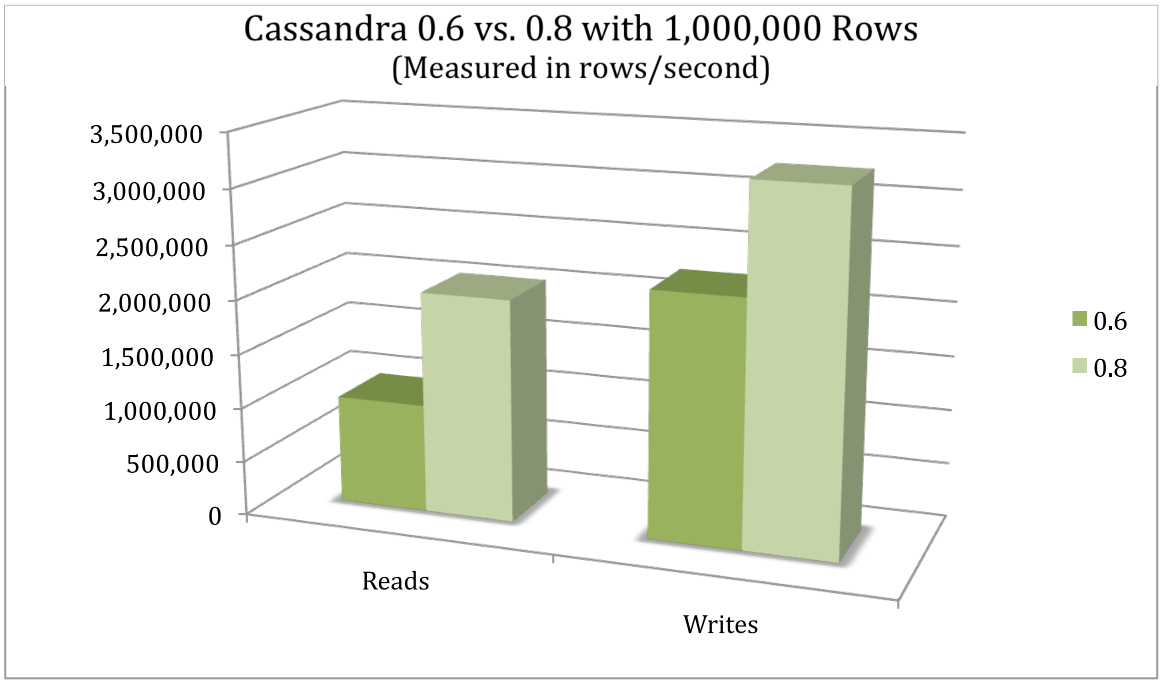
Features and Benefits of Cassandra 1.0

Even though version 1.0 of Cassandra has just been released, many companies have been using Cassandra for years in production environments. That said, there are other enterprises that have policies in place preventing them from using pre-1.0 software in production. While these companies have heard of Cassandra, they have yet to do any feature review and do not have an understanding of the core features found in the data management system.

For those unfamiliar with Cassandra, it will be helpful to first briefly touch on some key enhancements introduced in prior versions that have led up to the 1.0 release. Afterwards, for others who have experience with Cassandra, an overview of the enhancements that are new for version 1.0 will be presented.

Review of Key Enhancements Prior to 1.0

There have been many important enhancements in Cassandra that were delivered up to and including the 0.8.x version, which was the last pre-1.0 release of the database. Some of the most notable include the following:



Faster Performance for Reads and Writes

A hallmark of Cassandra for some time has been nearly unmatched performance for write operations, with linear performance gains resulting from the addition of new nodes to a cluster. This strength of Cassandra doesn't negate its ability in queries/reads; the database is quite capable of servicing stringent read response time requirements.

With the release of 0.8.0 of Cassandra, a number of performance enhancements were made such that the read/query throughput was improved with the end result being the current read performance of Cassandra now matches the previous version's impressive write response time's ability (a gain of over 100 percent).

In addition, write speeds also improved by nearly 45 percent.

The end result is that Cassandra supplies outstanding response times for all forms of application I/O demands.

CQL Language

Prior to version 0.8.0, interacting with objects and data in Cassandra was handled only via commands that were non-SQL-like (e.g., get to retrieve data; set to write data). That all changed with the introduction of CQL (Cassandra Query Language) in version 0.8.0.

CQL provides a very similar syntax to that used in all RDBMSs, making it very easy for developers and administrators coming from the relational world to begin working with Cassandra. DDL, DML, and SELECT functionality all can be found in CQL.

As an example, creating a column family in Cassandra is much like creating a table in any RDBMS:

```
cqlsh> CREATE COLUMNFAMILY users (
 ...  KEY varchar PRIMARY KEY,
 ...  password varchar,
 ...  gender varchar,
 ...  session_token varchar,
 ...  state varchar,
 ...  birth_year bigint);
```

Inserting and manipulating data is also much like an RDBMSs DML:

```
cqlsh> INSERT INTO users (KEY, password) VALUES ('jsmith', 'ch@ngem3a');
```

And querying data uses the familiar SELECT syntax:

```
cqlsh> SELECT * FROM users
 ...   WHERE gender='f' AND
 ...   state='TX' AND
...    birth_year='1968';
```

CQL is supported in the CQL command line client installed with Cassandra. Driver support for development languages up to version 1.0 included Java and Python.

Additional syntax support for CQL is planned for upcoming Cassandra releases.

Batched Operations

Cassandra supports tunable consistency on a per-operation basis, meaning developers can choose how strong or loose they want data consistency to be for a particular request. If a developer wants to apply a certain consistency level for a number of different requests, he or she can encase them in a BEGIN and APPLY BATCH statement. An example would be:

```
BEGIN BATCH USING CONSISTENCY QUORUM

INSERT INTO users (KEY, password) VALUES (‘user1’, ‘mypass’)

UPDATE users SET password = ‘newpass’ WHERE KEY = ‘user1’

INSERT INTO users (KEY, password) VALUES (‘user2’, ‘user2pass’)

DELETE name FROM users WHERE key = ‘user5’

APPLY BATCH
```

Batched operations allow a developer to retry (if necessary) a group of changes in an idempotent fashion.

Secondary Indexes

Before 0.7.0 of Cassandra, there was no way to easily index and access data on non-primary key columns in a column family. There was a method that entailed use of super columns; however, this approach proved to be a lot of work and was not considered a long-term solution.

Starting with version 0.7.0, secondary index support was introduced, which allows easy search operations on requests that aren’t performed by a row’s key/primary index. Secondary indexes are supported via CQL:

```
CREATE INDEX ON users (state);

SELECT * FROM users WHERE state = “TX”;
```

Typically, secondary indexes perform best with low cardinality data.

Expiring Data

Much of today's data has a life cycle to it, meaning that it needs to be present in a database for a particular time period. However, once that expiration period has occurred, the data can or should be removed from the system. An example might be a historical sales table that tracks monthly sales information over many years.

Enforcing data life cycle policies in most databases is difficult, with developers often having to write various purge routines that are scheduled to run every so often (e.g., deleting rows, dropping various partitions or columns of a table). Cassandra, however, offers a more elegant solution to implementing a data life cycle policy through the use of expiring columns, sometimes referred to as "time to live" data. Such an implementation has a substantial performance advantage of other "scan and purge" data removal processes.

When a column is established in a column family, an optional "time to live" value can be assigned to that column's data in a CQL INSERT or UPDATE statement so that the database automatically removes the column's data when its assigned life cycle amount has expired. This negates the need for any programmatic actions to be introduced for purging unwanted data from the database. Note that "time to live" options also are available via other Cassandra access methods (e.g., set).

Counters

Counters, which allow for incrementing columns needed for real-time analytical operations, were added in version 0.8.0 of Cassandra. To use counters, a developer creates a column family of CounterColumnType and then can either increment or decrement counter columns in the column family with either the incr or decr commands in the Cassandra core command line interface.

Counters also are supported via CQL, and represented in a typical SQL <column + value> fashion. For example, to increment a column counter by five, a CQL command may be:

```
cqlsh> UPDATE counters SET c1 = c1 + 5 WHERE key = row1;
```

Bulk Loader

The sstableloader utility, introduced in version 0.8.1, allows a developer or administrator to perform two key tasks:

1. Bulk load external data into a Cassandra cluster
2. Load existing SStables, typically generated from a backup/snapshot on one Cassandra cluster, into another cluster

As its name implies the sstableloader utility loads data only in SStable format, which means any external files will first need to be converted into SStables before being bulk loaded. (This can be accomplished via a Java routine, with examples being provided on the DataStax website.)

Other Notable Feature Additions Prior to Version 1.0

- Online modifications of keyspace and column family definitions
- Secured SSL communication between nodes

New Features in Version 1.0

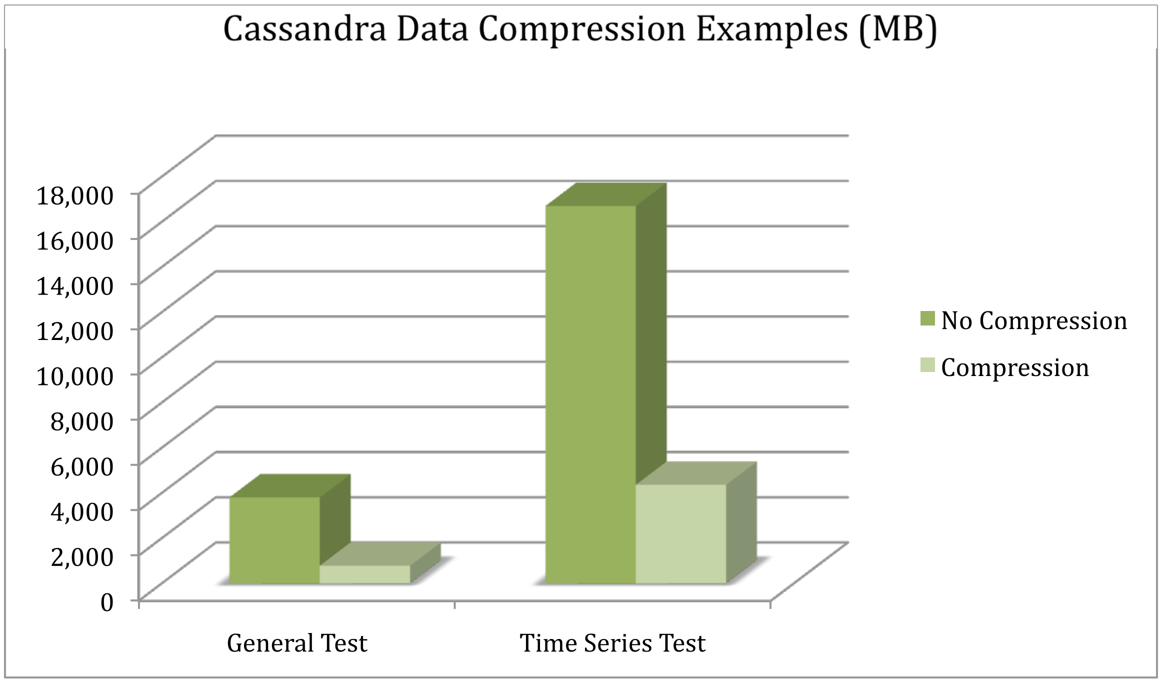
The following are some of the major new enhancements added to version 1.0 of Cassandra.

Data Compression

Cassandra is often chosen as a database management system because it can comfortably scale into the petabyte data range, while still offering very fast performance for both read and write operations. However, until version 1.0, it stored data without reducing the data's original footprint. So, for example, a terabyte of raw data equated to a terabyte being stored on disk across a cluster.

In version 1.0, data compression has been introduced to reduce the size of data stored on disk. The compression is based on Google's snappy data compression technology and can be set per column family in a keyspace.

The amount of data compression achieved can be quite high in some situations. Internal data tests at DataStax have shown achieved compression rates to range anywhere from 70 to 80 percent in a number of use cases.



Lastly, the use of compression on column families does not negatively impact either read or write performance. In truth, because of the reduction in physical I/O, compression can actually increase performance in a number of use cases.

Compaction Enhancements

When writes are performed in Cassandra, they are first written to the commit log and then to a memory structure called a Memtable. Writes continue to flow into a Memtable until it becomes full and then the data is flushed to disk in a sequential manner as an SStable.

The above process produces exceptionally fast write operations; however it also can lead to data fragmentation across the disk. Read requests may have to combine data from many SStables as well as Memtables to satisfy end user requests for data, and this can increase query response times.

To reduce data fragmentation and reclaim space taken by obsolete data, Cassandra performs “compactions” that merge the most recent data from many different SStables on disk into a new one.

Prior to version 1.0 of Cassandra, compaction operations were relatively primitive. In the worst case, compaction required up to 50 percent of a node's storage capacity to accommodate the copying of data during the compaction operation. Additionally, for update-intensive workloads, there was no upper bound on how many SStables might be required to satisfy a read.

Version 1.0 of Cassandra introduces a new and improved compaction algorithm inspired by LevelDB, Google's key-value datastore used in Chrome. This design groups SStables into "levels," where SStables within each level are a constant size contain non-overlapping ranges of data, and each subsequent level contains 10 times as much data as the previous.

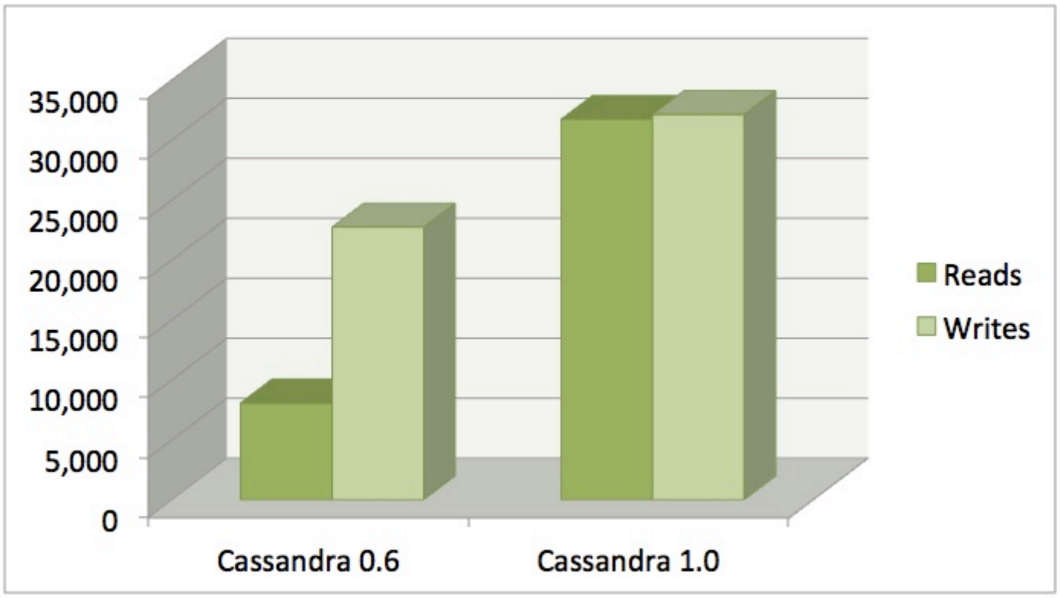
This solves both problems with the older approach to compaction: temporary space during a compaction is bounded to a small number of megabytes, and the number of SStables that a row may be fragmented across is bounded at one per level. The exponentially increasing level size also means that disk space overhead compared to an overwrite-in-place design is only about 11 percent (compared to the previous need for up to 50 percent).

Another side benefit is that data is de-duplicated much more elegantly in the compaction process, which has the ability to reduce the overall data size for column family keys that are being constantly rewritten by the host application.

The new compaction process in Cassandra 1.0 is enabled on a per column family basis and is not the default. It must be specified either when creating or updating a particular column family.

Additional Performance Enhancements
Compared to the Cassandra 0.6 release that was made available in October 2010, performance enhancements have been included in version 1.0 that have resulted in overall write performance increasing 40+% and read performance increasing an incredible 400%.

Specifically, for read performance, Cassandra 1.0 optimizes queries by using a lighter-weight data structure for representing a row fragment from a read, than for a row fragment in a memtable into which updates accumulates. Also, with named reads, Cassandra 1.0 includes enhancements for deserializing the most recent versions of requested columns. Combined with the other optimizations, this makes reads in Cassandra as fast as writes for many workloads.



Improved Memory Management

Cassandra provides a built-in row cache for very fast access to frequently requested data that is competitive with standalone caching products (e.g. memcached) but without the cache coherence problems that come from using a separate system, such as data in the cache becoming temporarily or even permanently out of sync with the database.

Cassandra 1.0 adds the ability to store cached rows in native memory, outside the Java heap. This results in both a smaller per-row memory footprint and reduced JVM heap requirements, which helps keep the heap size in the sweet spot for JVM garbage collection performance.

This off-heap row cache was introduced in version 0.8 but it didn’t become the default until 1.0. Note that the row cache requires the JNA library to be installed. If JNA is not installed, Cassandra will automatically fall back to the old on-heap cache provider.

Having now briefly covered the features and benefits found in Cassandra 1.0, let’s examine other related software released by DataStax along with 1.0 to assist with the development and management of Cassandra.

An Overview of DataStax

DataStax is the leading provider of enterprise NoSQL software products and services based on Apache Cassandra. Through its offerings, DataStax supports modern businesses that need a progressive data management system that can serve as an operational database as well as deliver built-in analytic capabilities for analyzing that data once it is in Cassandra.

Industries Served by DataStax

The industries currently using DataStax to support key applications include:

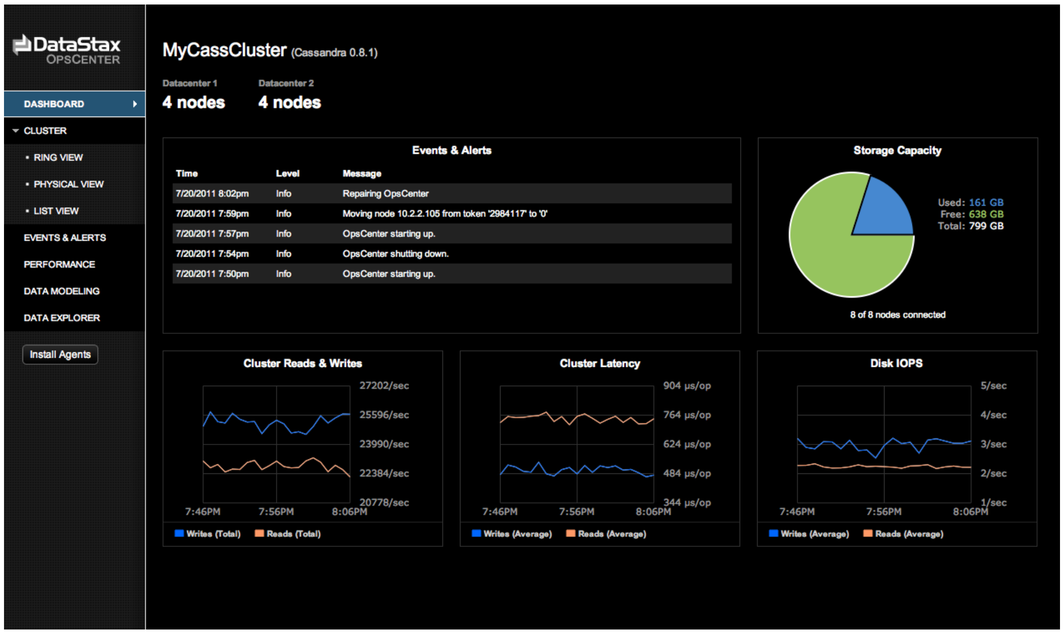
- Communications
- Consulting
- Consumer Electronics
- E-Commerce
- Entertainment
- Energy
- Financial
- Government
- Healthcare
- Hosting
- Marketing/Advertising
- Messaging
- Mobile Applications
- Online Gaming
- Retail
- Security
- Social Media
- Social Networking
- Software
- Travel

“Customers turn to us for highly complex analysis. ***The best way for us to deliver the experience our users demand is to employ extremely fast, scalable distributed computing based on Cassandra.”***

—Harry Schultz, Digital Reasoning

Software Supplied by DataStax for Cassandra 1.0

To accompany the release of Cassandra 1.0, DataStax has produced a number of management tools and developer aids that may be used free of charge with the database.



MyCassCluster (Cassandra 0.8.1)

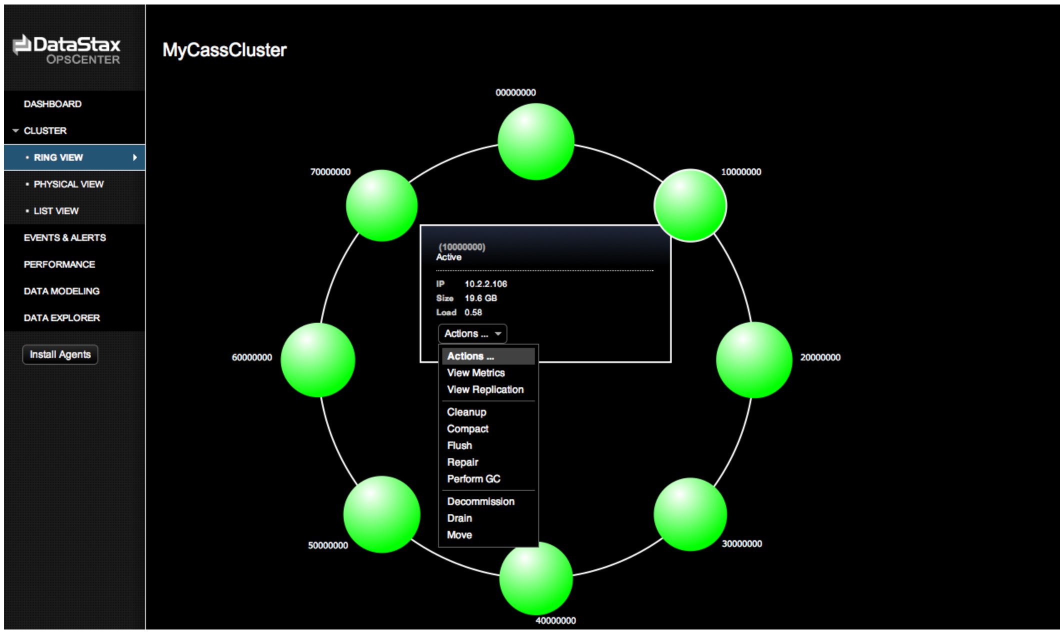
Datacenter 1: 4 nodes
Datacenter 2: 4 nodes

Events & Alerts

Time	Level	Message
7/20/2011 8:02pm	Info	Repairing OpsCenter
7/20/2011 7:59pm	Info	Moving node 10.2.2.105 from token '2984117' to '0'
7/20/2011 7:57pm	Info	OpsCenter starting up.
7/20/2011 7:54pm	Info	OpsCenter shutting down.
7/20/2011 7:50pm	Info	OpsCenter starting up.

OpsCenter Community

DataStax makes available OpsCenter Community, a free version of its visual management and monitoring solution for Cassandra. OpsCenter Community allows a developer or administrator to manage and monitor the health of an entire Cassandra cluster from a centralized web console.



(10000000)
Active

IP 10.2.2.106
Size 19.6 GB
Load 0.58

OpsCenter uses an agent-based architecture to monitor and carry out tasks on each node in a Cassandra cluster. Through an intuitive point-and-click interface, a user can understand the state of a cluster, what nodes are up and down, and what type of performance users are experiencing. Key events are reported into a centralized dashboard that is displayed along with other vital statistics.

CQL Drivers and Client Interfaces

DataStax supplies a number of drivers/connectors for popular developer languages such as Java and more. All drivers are provided free of charge and may be downloaded from the DataStax website.

DataStax Smart Start Installers

In addition to providing default TAR software packages for Cassandra, DataStax supplies text-based and GUI installers that help new users install everything needed to get started with Cassandra. The DataStax installers also help users configure Cassandra so the best possible out-of-the-box performance will be experienced.

DataStax Smart Start installers can be used for stand-alone implementations as well as multi-machine cluster installations. Defaults are provided for all options so a default installation on a single machine literally can be completed in under a minute.

Conclusions

The release of Cassandra 1.0 provides a number of eagerly anticipated enhancements (e.g., data compression) to an already strong and compelling enterprise-class database feature set. With version 1.0 of Cassandra, modern businesses can confidently roll out operational and analytic applications that are supported by a database foundation designed to scale and support both increasing user loads and large data volumes.

To find out more about Cassandra, along with products and services offered by DataStax, or to download and get started with Cassandra today, please visit www.datastax.com or send an email to info@datastax.com.

About DataStax

DataStax is the developer of DataStax Enterprise, a distributed, scalable, and highly available database platform that delivers optimal performance either on premise or in the cloud for modern enterprise applications that manage both real-time and analytic workloads. The company has over 100 customers, including leaders such as Netflix, Cisco, Rackspace and Constant Contact, and spanning verticals including web, financial services, telecommunications, logistics and government. DataStax is backed by industry leading investors, including Lightspeed Venture Partners and Crosslink Capital and is based in Burlingame, CA with offices in Austin, TX and Stamford, CT. For more information, visit www.datastax.com.