A guide to big data workload-management challenges

By George Gilbert

This research was underwritten by DataStax.

Table of contents

Executive summary

The explosive growth in the volume, velocity, variety and complexity of data has challenged both traditional enterprise application vendors as well as companies built around online applications. In response, new applications have emerged, ones that are real-time, massively scalable and have closed-loop analytics. Needless to say, these applications require very different technology underpinnings than what came before.

Traditional applications had a common platform that captured business transactions. The software pipeline extracted, cleansed and loaded the information into a data warehouse. The data warehouse reorganized the data primarily to answer questions that were known in advance. Tying the answers back into better decisions in the form of transactions was mostly an offline, human activity.

The emerging class of applications requires new functionality that closes the loop between incoming transactions and the analytics that drive action on those transactions. Closing the loop between decisions and actions can take two forms: Analytics can run directly on the transactional database in real time or in closely integrated but offline tasks running on Hadoop. Hadoop typically supports data scientists who take in data that's far more raw and unrefined than the type found in a traditional enterprise data warehouse. The raw data makes it easier to find new patterns that define new analytic rules to insert back into the online database.

This paper is targeted at technology-aware business executives, IT generalists and those who recognize that many emerging applications need new data-management foundations. The paper surveys this class of applications and its technology underpinnings relative to more-traditional offerings from several high-level perspectives: the characteristics of the data, the distinctiveness of the new class of applications, and the emerging database technologies — labeled NoSQL, for lack of a better term — that support them. Although the NoSQL label has been applied to many databases, this paper will focus on the class of systems with a rich data model typified

by Cassandra. Other databases in this class include HBase, DynamoDB and Oracle NoSQL.

Understanding the new class of applications

Consider large-scale machine-to-machine monitoring and measurement as an example of applications that require new technology underpinnings. E3 Greentech, a Decatur, Ga.–based startup, has a great example of such an application. Although currently targeted at utilities and others that generate, distribute and consume power, it can ultimately apply to any intelligent grid of machines.

Power is now generated not only by utilities at large, central sites but also at distributed sites with basic generators. The latter can also store power and distribute it back into the grid when necessary. More important, intelligent sensors and devices are making it possible for homes to instrument their electric appliances — from water heaters and air conditioners to dishwashers and entertainment centers — in order to measure and manage electricity consumption.

New applications supporting new business models

One notable aspect of E3 Greentech's application, and potentially similar ones, is how it can enable new business models. E3 Greentech's application makes it possible for utilities to not only provide electricity but also offer home energy management services. Because the application is part of an ongoing relationship between a utility and its end consumers, E3 Greentech can support pricing that captures some of this ongoing value. These pricing models reflect a managed service that goes beyond typical SaaS application pricing, which only captures the value of managing the application itself.

Requirements for new technology underpinnings

The first requirement for this class of application is that the same database has to serve both as the operational data store as well as the basis for real-time analytics. In addition, analytics that are performed offline ideally should be on another cluster of the same database so that moving data back and forth doesn't require the complicated and slow pipeline that typically connects traditional online transaction processing (OLTP) databases and enterprise data warehouses. The database must also deal with data that is far less refined than in traditional SQL databases. It must include not only structured data but also semistructured and unstructured data. A related requirement is the ability to handle potential changes in the format of the data without taking the database or application offline. Finally, the database needs to run on premise, in the cloud or on a combination of both without complicated configuration of the data distribution mechanics.

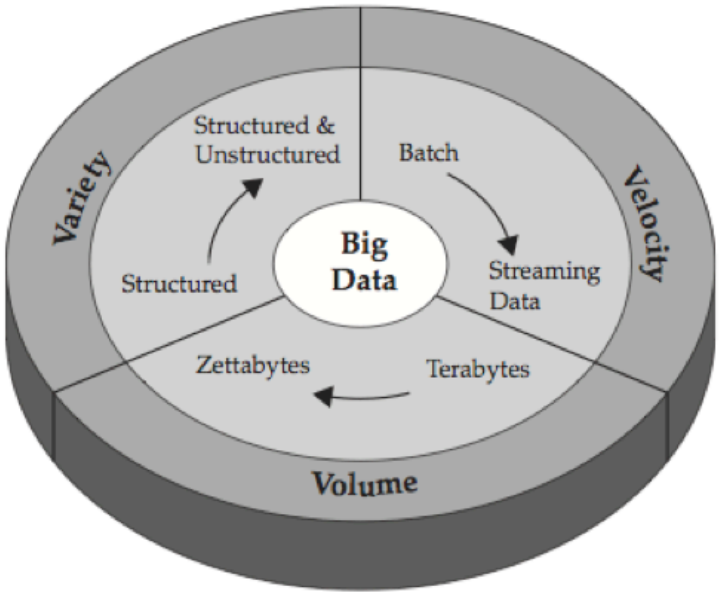
Volume, velocity and variety of data

The previously nebulous definition of "big data" is growing more concrete as it becomes the focus of more applications. As seen in Figure 1 (below), volume, velocity and variety make up three key characteristics of big data:

- **Volume.** Rather than just capturing business transactions and moving samples and aggregates to another database for analysis, applications now capture all possible data for analysis.
- **Velocity.** Traditional transaction-processing applications might have captured transactions in real time from end users, but newer applications are increasingly capturing data streaming in from other systems or even sensors. Traditional applications also move their data to an enterprise data warehouse through a deliberate and careful process that generally focuses on historical analysis.
- **Variety.** The variety of data is much richer now, because data no longer comes

solely from business transactions. It often comes from machines, sensors and unrefined sources, making it much more complex to manage.

Figure 1: The three characteristics of big data: volume, velocity and variety



Source: IBM

Data volume

A more detailed look at data volumes highlights some of the new database requirements. E3 Greentech's application takes in far more data than simply reading a home's meter every 30 minutes to find out aggregate power consumption. In fact, the application reads the consumption data on so many sensors tied to so many appliances and devices that it takes readings roughly 100,000 times per day per residence. As a result, E3 Greentech needs a database that can handle more writes than a traditional database running on one big server or on one shared storage device.

Data velocity

Measuring and managing energy consumption down to the device level requires real-time data and analysis. To look at E3 Greentech again, that means collapsing the traditional pipeline that batch processes data between one database that handles writes and another that handles reads for analysis. E3 Greentech handles data more like a nonstop incoming stream with a set of continuous queries spitting out answers. The company has already tested Cassandra at 1 million writes per second on a cluster of servers. And that doesn't count the simultaneous queries that are analyzing how to manage the demand for electricity for maximum efficiency. That still leaves plenty of headroom.

Part of what makes this speed possible is that the incoming data is primarily new inserts, not updates to existing data, which requires more overhead. In addition, the real-time queries tend to be less complex than what is possible with a traditional SQL database.

To be fair, there are queries in E3 Greentech's application, and with most applications in this class, that require more processing time than a real-time system can handle. Here Cassandra's Hadoop integration becomes relevant. Another cluster holds a replica of the Cassandra data in Hadoop for offline queries while the answers get fed back into the real-time Cassandra cluster to improve its energy-management decisions.

Data variety

The variety of data that the new class of databases has to manage is arguably their most fundamental differentiator.

SQL databases derive much of their strength from the order and discipline they force on their data. In the example of an energy-management application, every device or

entity being measured and monitored would have to be accounted for and described down to its very last attribute. This schema structure requires the application designers to go through a very detailed process of refining the data until everything is classified. The value of this approach is that there is no ambiguity in the data, and the application knows exactly what it's dealing with. The downside is that if the application needs to handle new devices or query the data to find new signals not previously anticipated in the noise of all the data, a lot of additional refining has to happen.

Cassandra makes it much easier to avoid the upfront refining process, as long as the application is intelligent enough to make sense of the updated or new device entities. It also makes it easier to ask many new questions of the data without pumping it through the batch process to get it into the data warehouse (though more-sophisticated questions would still have to go to the Hadoop cluster). In terms of new devices, if the device readings from a house started to include a new type of outlet that also included an occupancy sensor and a temperature sensor, the same column family would start storing the additional attributes without missing a step. Meanwhile the application has already been written to understand that it can now track temperature and occupancy readings from this new device.

From another perspective, it's critical to understand that all the raw data, including these new readings, is available for analysis. The traditional batch process of extracting, transforming and loading data from the online database to the data warehouse removes any variation in the type of data. Again, there is a purpose in this type of refinement. It makes it easier for analysts to make sense of the data, although at the expense of being able to go back into the messy details to find new signal within the noise.

Real-time, massively scalable and closed-loop characteristics of applications

Real time

Continuing with the home energy management example, reducing consumption demands much more than just switching off everyone's air conditioner when the weather is hot and humid. Managing real-time consumption minimizes electricity charges during peak usage, and it can meaningfully reduce the 30 to 40 percent of power-generating capacity that is built just to satisfy peak demand. The application depends on the real-time read/write database to "turn the crank" and evaluate the state of all the assets under management in a few milliseconds as new readings continue to stream in.

That's where real-time applications and databases come into play. The application can compare the trend of the recent readings of indoor and outdoor temperatures and the current thermostat settings for all houses in a certain area that share the same weather. Even though these readings are streaming in, in real time, the database can store the data so it can be accessed instantly and without blocking additional incoming writes. Furthermore, offline analysis from the application running on the replica cluster that runs Hadoop might calculate and periodically update a thermal envelope estimate for each house. By combining these readings plus an occupancy sensor, the application can determine when to raise or lower the thermostat.

For such a system, Cassandra stores and organizes not only the readings but also the components that control the application's mechanics. The application has to put together groups of entities — devices, sensors, homes, neighborhoods and weather regions — dynamically. And as described in the section on data variety, it has to make sense of these groups even when the data coming back from the individual readings changes or as new devices are added. In order to make this happen, Cassandra stores each device and those that are similar in a group of rows called column families. The

values for each sensor reading or row get stored in a flexible number of columns. If the device gets swapped for a similar type, it can store additional or fewer columns for each reading.

The magic is how the system can calculate in real time groups of geographic regions, homes and devices that meet changing criteria and on which the application is going to take action to change energy consumption. The incoming stream of sensor data might not conform exactly to what's stored in the database, but the application still correlates the individual readings from the physical devices with the dynamic collections of classes of devices organized in the database.

Massively scalable

In addition to real-time responsiveness, it takes an application running on a massively scalable database like Cassandra to manage the energy consumption of potentially millions of individual devices and appliances across hundreds of thousands of homes relative to the peak load on the grid. The scalability comes from being able to apply the correct power-management rules for each dynamic group described above (i.e., turn off the water heater for homes when the temperature is above 80 degrees and no one is home).

There are an unlimited number of possible rule combinations as they are applied to progressively more-targeted groupings within the hundreds of thousands of individual houses under management. For example, the application might get a reading that a particular home just became occupied. It then runs a rule to reset the thermostat back to where the occupant originally left it if the temperature outside is more than 10 degrees higher than inside. Once that rule executes, the application might use another to evaluate the same temperature readings to determine just how much additional power the water heater needs to reach a comfortable level. Operating on this scale

while handling reads and writes requires a database that can run on a large cluster of machines without bottlenecking on data stored and shared in a single place.

Most traditional SQL databases for OLTP operate on data that is shared on a common storage array. They are configured this way so that any node in the database cluster can get at any data item or set of items. And for those that do scale out to a cluster of independent machines without shared storage, they still have the requirement to separate the database handling the high volume of updates from the one configured to ask the mostly predefined set of queries. The pipeline that connects these two databases via a batch process is the biggest impediment to real-time operation.

Closing the loop

In traditional applications, analytics haven't been part of a continuous loop that automatically feeds better decisions back into real-time transactions. With Cassandra, there are two mechanisms to accomplish this. The real-time section above describes how the E3 Greentech application dynamically groups devices and homes in order to apply rules that improve energy efficiency. This part of the analytics works in real time because there is no batch-processing pipeline between the online application and the analytics. But there are also patterns of analysis that need access to more data and have to run offline in order to deliver meaningful answers. Cassandra's deep integration with Hadoop supports these scenarios and makes it easy to feed the answers back into the real-time part of the application.

While the real-time part of the application may be operating on current sensor readings, Hadoop works best for finding patterns and correlations across large data sets. For example, a smart-grid application needs to know the average daily temperature for each geography as well as a distribution of readings over time. With this information, the Hadoop-sourced analysis can be fed back periodically into the

real-time application loop. There it can help make decisions based on where the current temperature is relative to its average as well as the highs and lows.

This same type of pattern analysis can be useful for supporting offline decisions. The application might measure how quickly the temperature changes in all homes in a neighborhood relative to weather conditions. Based on the results, it can recommend to the property manager which ones can benefit from weatherization. Alternatively, it can look at the efficiency relative to manufacturer benchmarks of heavy-duty appliances like the water heaters and the air conditioners within each home. After recommending what can be fixed or upgraded, the application can again measure their new performance against the benchmark.

The integration between Cassandra and Hadoop dramatically simplifies these processes. Cassandra replicates its data with a cluster running the Hadoop Distributed File System (HDFS), and that runs the MapReduce application framework on top of it. Because it is a replica cluster, the data gets updated from the online application in real time. And because it is HDFS, any application written to MapReduce or one of the higher-level programming frameworks works without modification.

As documented comprehensively elsewhere, MapReduce simplifies the process of working with massively unrefined data sets such as those from sensors. Rather than going through the rigorous process of modeling each potential device on the smart grid and all of its potential attributes, a MapReduce job can analyze just the available attributes. However, the repository is unrefined and contains all the "noise," in terms of additional stray attributes and new sensors. Having that unrefined data already as part of the repository makes it easier to update the analysis in order to search for new patterns. The parallelism built into MapReduce makes the process highly scalable.

In traditional enterprise data warehouses, the batch process pipeline connecting from the OLTP database needs to be updated to handle additional entities in the form of

new devices or attributes. The data warehouse would also need updating in two dimensions. First, it would need to accommodate the new information in its refined format. And it would need to accommodate the new questions developers want to ask.

It should be noted, however, that Hadoop's flexibility has a price. As described earlier, unrefined data requires more effort on the part of the developer to make sense of it and analyze it.

Approaches to scalability

Traditional SQL DBMS

Traditional SQL databases must trade unlimited scalability in favor of maintaining data consistency and integrity above all else. Managing a bank account is a perfect example. If a customer withdraws cash, the database has to make sure the transaction updates the customer's account balance before any other account activity can take place. Otherwise it could be possible for multiple withdrawals to exceed the account balance. That block on any simultaneous activity is the core of the data integrity features of traditional SQL databases. But that block is also the bottleneck that limits the ultimate scalability of online-transaction-processing applications on SQL databases. All processes in an application ultimately must share access to a single master copy of data in order to ensure there are no reads of "dirty" or out-of-date data.

A clarification on SQL DBMS technical nomenclature:
The full technical requirement with traditional SQL DBMS is for ACID transactions, which means the database must enforce atomicity, consistency, isolation and durability. In this context, consistency refers to data-integrity constraints. These constraints might enforce rules such as requiring all phone numbers to include 10 numeric digits or that a customer record can't be deleted if it still has outstanding orders. This definition of consistency is different from how it is applied with emerging NoSQL databases, as explained in the following section.

Emerging NoSQL databases

NoSQL databases generally relax the consistency described above in favor of scalability or availability. The relevant technical acronym is CAP, which stands for consistency, availability and partition tolerance. These attributes represent the choice of trade-offs that different NoSQL databases make. Theoretically it's only possible to offer two of these three features at any one time in a distributed database.

Because the data doesn't have to be 100 percent consistent all the time, applications can scale out to a much greater extent. By relaxing the consistency requirement, for example, NoSQL databases can have multiple copies of the same data spread across many servers or partitions in many locations. The data is instead eventually consistent when the servers are able to communicate with one another and catch up on any updates one may have missed.

Cassandra

Cassandra lets the application developer dial in the appropriate level of consistency versus scalability or availability for each transaction. This "tunable consistency" is a level of sophistication that other databases such as HBase don't always offer. However, the extra sophistication comes with more of a learning curve for the developer. Cassandra also has an in-memory cache to speed access to the most important data on designated servers in the cluster. However, its current implementation largely leverages the functionality already in the operating system. By contrast, Couchbase, for example, includes a cache that intelligently moves data in and out of memory on each server and across partitions residing on other servers as needed. Despite this feature, Couchbase is not a substitute for Cassandra. Cassandra offers a richer data model that combines elements of tabular data in SQL databases and the flexibility typical of rich key-value stores. E3 Greentech, for example, uses Couchbase to cache and store the

most recent sensor reads for each device, but it runs the rest of the application on Cassandra.

Alternatives to Cassandra

Amazon DynamoDB

Amazon recently introduced a new database, DynamoDB, which offers more sophistication and scale than its earlier, simple key-value store, SimpleDB. Three attributes distinguished it from competitors when it was introduced:

- It is a managed service. Customers don't have to know anything about deploying or managing a database cluster.
- It is both configured and priced by a combination of "tunable" read/write throughput and total storage capacity, so the service itself figures out how to configure the data.
- It is designed exclusively to exploit the high throughput and durability of flash SSD storage. The latest release of Cassandra can also be configured to support SSD.

Oracle NoSQL

Oracle built its NoSQL database on the foundation of Berkeley DB, which the company acquired six years ago. The technology shares some similarities with Cassandra in that it is a distributed, replicated key-value store that runs on a cluster of machines. However, Oracle has made some design decisions that point to different use cases. Rather than accommodating arbitrarily large-scale deployments, the database appears best designed to be deployed in a single rack or an Oracle-designed hardware appliance. Because it is not a fully peer-to-peer system, if a master node goes down and it is configured for relaxed consistency, updates could get lost before they find

their way to a new master node elsewhere on a network when it gets elected. That type of scenario wouldn't happen if it were deployed in a single rack or an appliance.

HBase

HBase is the no NoSQL distributed database that forms part of the Hadoop stack. It provides strongly consistent reads and writes versus an "eventually consistent" data store. It was created at Google and derived from a product called BigTable and Google File System. Cassandra was created at Facebook and while implementing the BigTable data model; it uses a system inspired by Amazon's Dynamo for storing data. (Much of the initial development work on Cassandra was performed by two Dynamo engineers recruited to Facebook from Amazon.) These differing histories have resulted in HBase's being more suitable for data warehousing and large-scale data processing and analysis (i.e., indexing the Web) and Cassandra's being more suitable for real-time transaction processing and serving interactive data.

Sustainable versus nonsustainable differentiators

Some products, such as DynamoDB, have valuable differentiators, but they aren't necessarily permanent. For example, a PaaS vendor such as Heroku could offer Cassandra or HBase as a managed service. Also, in its most recent release Cassandra has been modified to allow certain workloads to target flash SSD as its storage medium along with others targeting traditional spinning disk drives. With that modification, any database can be configured to deliver a promised level of throughput. A managed service provider could then be simply configured and priced based on throughput, just like Amazon's DynamoDB. A customer running it on premise could also achieve much higher and more predictable performance than running it with spinning disk drives.

That Cassandra can offer configurable support for flexibly distributing data across multiple on-premise data centers and public clouds is a critical choice. This flexibility can solve this bottleneck when latency and bandwidth limitations prevent large data

sets — in the billions of rows — from being moved to or from cloud providers like Amazon.

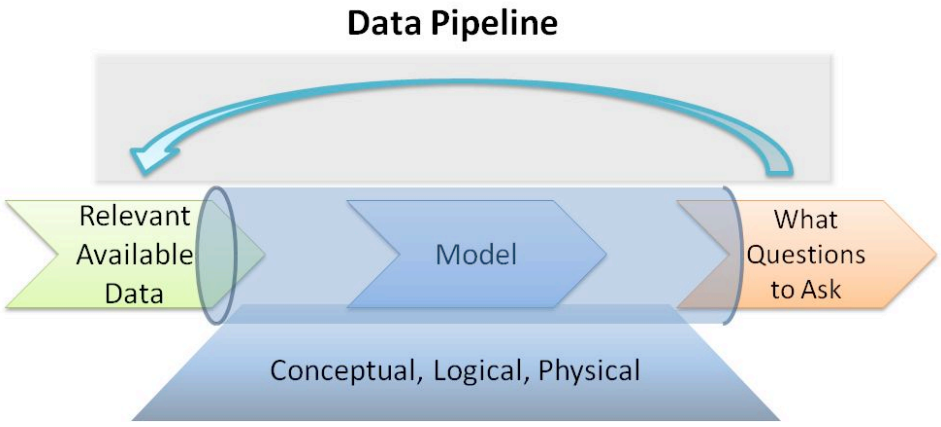
Pricing options are also valuable. Having an option to use different storage media is important also for pricing flexibility. Customers pay dearly for the privilege of running on flash SSD. Once there, using DynamoDB for a 1 TB database with a throughput of 500 writes per second would cost roughly $70,000 per month.

Driving better decisions into online applications

Making analytics an integral part of online applications requires progressively iterating on a model to improve its predictive capability. As the model improves, it becomes easier to benchmark activity captured in the application. Benchmarking makes for more-effective predictions of outcomes. It also makes detecting anomalies more accurate so the application can take remediating actions.

As seen in Figure 2, traditional SQL DBMS refine the data they store so that it is easy for application developers to take advantage of it. However, the refining process creates a requirement to collect just the data relevant for the ultimate questions it will answer.

Figure 2: The data pipeline



Source: Dave Campbell keynote, VLDB 2011

NoSQL databases such as Cassandra don't necessarily replace SQL databases in building analytics into online applications. But the new class of databases definitely enhances the process through complementary functionality in the online database as well as in the Hadoop cluster.

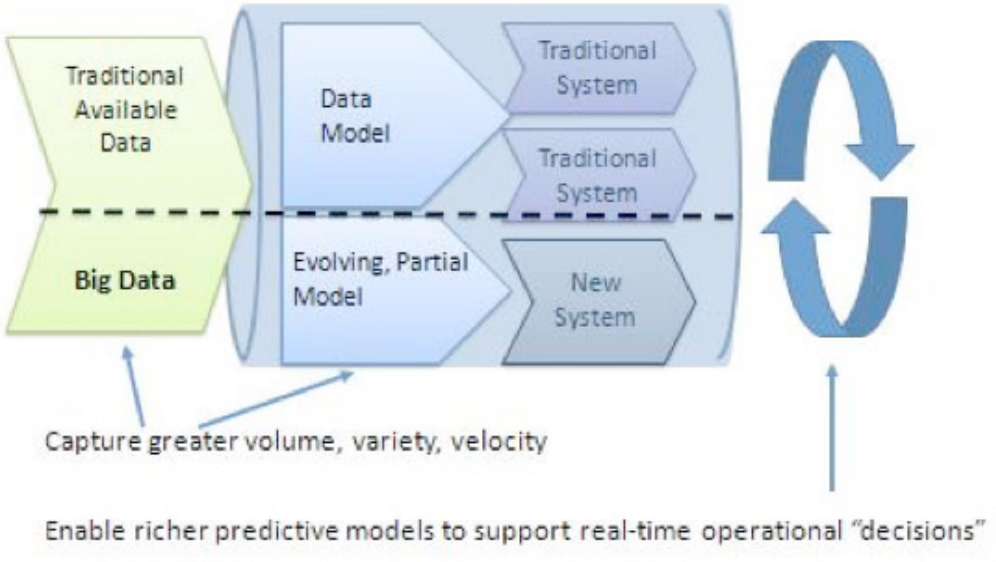
It's worth calling special attention to how this process complements analytics in traditional enterprise data warehouses. Virtually all traditional data warehouses are based on a combination of aggregated data and samples of the full details in the raw data set. That is a natural part of the process of modeling or refining the data to fit into the transactional application and then to batch process it into the data warehouse.

The fundamental trade-off at the heart of this design is to feature data consistency and integrity at the expense of unanticipated flexibility in the data. Big data supported by NoSQL data stores such as Cassandra makes the opposite trade-off. It makes it easy to capture all the detail data in all of its partially structured or unstructured clutter.

Getting all of that unrefined data clutter requires capturing it further back in the data pipeline than the refined data feeding traditional OLTP applications or data

warehouses. Access to that raw data makes it possible to pursue a certain line of analysis that wasn't previously modeled into the traditional data. The raw data also improves the predictive models solely through the availability of a bigger data set against which they are tested. The smart-grid application provides a perfect example.

Figure 3: Systems based on big data



Source: Loosely based on Dave Campbell's keynote, VLDB 2011

The smart-grid application is an early example of the Internet of things, where every electrical device in a home is instrumented to report on its operation. To start out, the production analytic model might measure how much the temperature in each unoccupied house rises when the A/C is off relative to the outside temperature. Later, an analyst might use the Hadoop cluster to factor in additional variables such as humidity and the capacity of the A/C unit and then derive how quickly a particular house can be cooled. Although the rule comes out of an offline, batch process in Hadoop, it would get fed back into the online application cluster. Every time a new reading comes streaming in, the devices, houses, neighborhoods, weather and climate get evaluated to see if the new rule should run. The time to cool a house to the initial

thermostat setting becomes one of the thousands of rules running every few milliseconds as new data from across the system arrives.

The key Cassandra features reviewed here make it an ideal mixed-use database that can support real-time transactions as well as real-time and batch analytics.

A note on the distinction between Cassandra and DataStax Enterprise

DataStax is the company behind the Apache project, employing roughly 90 percent of the committers on the open-source project. The application example in this paper, E3 Greentech, currently uses the open-source version of Cassandra primarily because the company was started before DataStax Enterprise (DSE) was available. This section details the features that could have been utilized if the commercial version had been available in time.

DSE extends Cassandra to simplify deployment and operation and more tightly integrate Hadoop-based analytics. The free, open-source version of Cassandra has real-time, two-way replication with HDFS-based Hadoop clusters, but E3 reports it is extremely complicated to configure. The DSE version, by contrast, works out of the box. An administrator simply distinguishes the online nodes and the Hadoop analytic nodes.

The Hadoop nodes are themselves part of DataStax Enterprise, having added several critical new features. DSE replaces the open-source HDFS file system and introduces instead the HDFS-compatible Cassandra File System (CFS). Until recently CFS eliminated the single point of failure in HDFS. If one central node went down, the Hadoop cluster would be unavailable. HDFS has since fixed this problem. However, having an HDFS-compatible CFS supports MapReduce and other elements of the expanding Hadoop stack without modification.

DSE and its integral version of Hadoop can be deployed with a more configurable topology that enables greater availability or reduced latency. A data center, which is DataStax's name for a cluster, can be configured to have one or more replicas of the online database as well as Hadoop colocated. In addition, other locations themselves can have a configurable number of real-time replicas.

Finally, DSE has a tool called OpsCenter for managing database clusters that's not in the open-source version. It provides detailed, visual configuration information on the whole cluster in one place. It relieves the administrator of having to configure and manage each node through an API.

For more on the innovation driving Hadoop and related technologies, see the companies driving the main distributions, Cloudera and Hortonworks. They are driving a very rapid expansion in the complementary developer and administrator tools and frameworks that make Hadoop accessible to a much broader audience. Many ISVs and corporate developers have little experience in programming and administering highly distributed systems.

Key takeaways

Business considerations

Business managers collaborating with IT have to assess a new class of technologies that can support new business models:

- Traditional enterprise applications have focused on delivering operating efficiencies, principally in the general and administrative line item of profit and loss (P&L) financial statements.
- More recently familiar consumer websites have pioneered businesses where the online service is itself the business and drives the entire P&L. These same sites have also pioneered a new class of foundation technologies that are more real time, more scalable and more closely integrated with online decision making than traditional applications.
- These new foundation technologies are becoming mature enough for more-mainstream deployment. Successful deployment no longer requires the ability to submit patches to the community managing the source code.
- Businesses that have ongoing relationships with their customers should evaluate whether they can deliver new services that complement their existing activities and whether these new foundation technologies can make that possible.

Technology considerations

- Despite the hype cycle around big data and NoSQL databases, they underpin a major new class of applications.
- Real-time, massively scalable applications with closed-loop analytics differ from traditional enterprise applications and webscale applications built on traditional databases in several key ways:
    1. In the characteristics of the data they manage

2. In the combination of real-time and offline closed-loop analytics they require
3. In the need to combine highly scalable writes and analytics in the same database so the application can effect improved operational performance on the real-world systems that it manages itself or via human operators
4. In the way they can run, easily manage and reintegrate offline analytics on a rich set of data types on massively scalable infrastructure

- The database underpinnings of this class of applications have some distinctive characteristics. In particular, in order to support a high volume of simultaneous writes and analysis, these databases put a greater burden on developers to keep track of how their data is stored.
- Relevant databases in this family include Cassandra, Amazon DynamoDB and Oracle NoSQL.

Appendix A: understanding the assumptions in NoSQL databases as a class of systems relative to SQL databases

The debate about how to manage big data revisits arguments about databases previously thought settled for decades. For nearly 25 years relational databases have provided the foundation for enterprise and online applications. What distinguished this foundation from previous database technologies was the careful, modular separation between the data and the applications built on top of it. That modular separation made it possible for applications to worry only about what data they needed, not how to get at it. Each could be changed and updated without breaking the other.

Now the big data meme is reopening the debate about how best to manage data. In some cases, the modular separation of applications and databases will have to give way. For lack of a better term, NoSQL has become the label for many of the new approaches.

Within all the new approaches, including the ones that attempt to stretch the capabilities of RDBMS, there is a common theme: The formal separation that permits applications to avoid having to worry how the data is stored disappears. While that sounds like a major step backward to pre-RDBMS days, in some scenarios such as extreme scalability or managing "messy" data, it makes sense. When performance or functional requirements demand capabilities unavailable in off-the-shelf modules, the application developer is responsible for adding those capabilities through proprietary integration with the data-management layer. Applications have to be more tightly fused with their data. Once again, they have to know how to get at their data physically rather than just say what they need. This trade-off between modularity and integration should be familiar to all readers of Clayton Christensen's books on disruptive innovation.

About George Gilbert

George Gilbert is the co-founder of TechAlpha Partners, a management consulting and research firm that focuses on the business and technical implications of the transition to cloud-based data services. Previously Gilbert was the lead enterprise software analyst for Credit Suisse First Boston, one of the leading investment banks to the technology sector. Prior to that he worked at Microsoft as a product manager and spent seven years in product management and marketing at Lotus Development. He holds a BA in economics from Harvard University.

About GigaOM Pro

GigaOM Pro gives you insider access to expert industry insights on emerging markets. Focused on delivering highly relevant and timely research to the people who need it most, our analysis, reports and original research come from the most respected voices in the industry. Whether you're beginning to learn about a new market or are an industry insider, GigaOM Pro addresses the need for relevant, illuminating insights into the industry's most dynamic markets.

Visit us at: http://pro.gigaom.com

Further reading

Locating data centers in an energy-constrained world

Globally data centers are currently buying $30 billion of power per year, but changes in the power market are rippling through the Internet industry, altering both the location of data centers and their sources of power. This report parses the many complications of picking a data center location, which we call data-center siting. Siting includes considerations beyond the geographic location, such as how to procure energy and green-energy models, all to run better on a new, ever-shifting energy infrastructure. This report will help decision makers like CIOs and data-center planners weigh the most important parameters behind locating new data centers, whether in the U.S. or globally. Doing so will lead to more cost-effective, reliable and sustainable data centers.

Sector RoadMap: Hadoop platforms 2012

For years, technologists have been promising software that will make it easier and cheaper to analyze vast amounts of data in order to revolutionize business. More than one solution exists, but today Hadoop is fast becoming the most talked about name in enterprises. There are now more than half a dozen commercial Hadoop distributions in the market, and almost every enterprise with big data challenges is tinkering with the Apache Foundation–licensed software. This report examines the key disruptive trends shaping the Hadoop platform market, from integration with legacy systems to ensuring data security, and where companies like Cloudera, IBM, Hortonworks and others will position themselves to gain share and increase revenue.

How Amazon's DynamoDB is rattling the big data and cloud markets

The latest AWS offering to rock the technology establishment is DynamoDB, a NoSQL database service that puts the power of NoSQL in the hands of every developer. This research note analyzes the multiple ways in which Amazon's announcement has disrupted the big data and cloud computing market and what that means for other

companies and offerings in the space — from the startups selling Hadoop distributions to public cloud providers like Rackspace and Microsoft, which will have to scramble to keep up and differentiate.