Big Data Analytics Options on AWS

Erik Swensson

December 2014

Contents

Abstract

Amazon Web Services (AWS) is a flexible, cost-effective, easy-to-use cloud computing platform. The AWS Cloud delivers a comprehensive portfolio of secure and scalable cloud computing services in a self-service, pay-as-you-go model, with zero capital expense needed to handle your big data analytics workloads, such as real-time streaming analytics, data warehousing, NoSQL and relational databases, object storage, analytics tools, and data workflow services. This whitepaper provides an overview of the different big data options available in the AWS Cloud for architects, data scientists, and developers. For each of the big data analytics options, this paper describes the following:

- Ideal usage patterns
- Performance
- Durability and availability
- Cost model
- Scalability
- Elasticity
- Interfaces
- Anti-patterns

This paper describes two scenarios showcasing the analytics options in use and provides additional resources to get started with big data analytics on AWS.

Introduction

As we become a more digital society the amount of data being created and collected is accelerating significantly. The analysis of this ever-growing data set becomes a challenge using traditional analytical tools. Innovation is required to bridge the gap between the amount of data that is being generated and the amount of data that can be analyzed effectively. Big data tools and technologies offer ways to efficiently analyze data to better understand customer preferences, to gain a competitive advantage in the marketplace, and to use as a lever to grow your business. The AWS ecosystem of analytical solutions is specifically designed to handle this growing amount of data and provide insight into ways your business can collect and analyze it.

The AWS Advantage in Big Data Analytics

Analyzing large data sets requires significant compute capacity that can vary in size based on the amount of input data and the analysis required. This characteristic of big data workloads is ideally suited to the pay-as-you-go cloud computing model, where applications can easily scale up and down based on demand. As requirements change you can easily resize your environment (horizontally or vertically) on AWS to meet your

needs without having to wait for additional hardware, or being required to over-invest to provision enough capacity. For mission-critical applications on a more traditional infrastructure, system designers have no choice but to over-provision, because a surge in additional data due to an increase in business need must be something the system can handle. By contrast, on AWS you can provision more capacity and compute in a matter of minutes, meaning that your big data applications grow and shrink as demand dictates, and your system runs as close to optimal efficiency as possible. In addition, you get flexible computing on a world-class infrastructure with access to the many different geographic regions that AWS offers[1], along with the ability to utilize other scalable services that Amazon offers such as Amazon Simple Storage Service (S3)[2] and AWS Data Pipeline.[3] These capabilities of the AWS platform make it an extremely good fit for solving big data problems. You can read about many customers that have implemented successful big data analytics workloads on AWS on the AWS case studies web page.[4]

Amazon Redshift

Amazon Redshift is a fast, fully-managed, petabyte-scale data warehouse service that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools.[5] It is optimized for datasets ranging from a few hundred gigabytes to a petabyte or more, and is designed to cost less than a tenth of the cost of most traditional data warehousing solutions. Amazon Redshift delivers fast query and I/O performance for virtually any size dataset by using columnar storage technology while parallelizing and distributing queries across multiple nodes. As a managed service, automation is provided for most of the common administrative tasks associated with provisioning, configuring, monitoring, backing up, and securing a data warehouse, making it very easy and inexpensive to manage and maintain. This automation allows you to build a petabyte-scale data warehouse in minutes, a task that has traditionally taken weeks, or months, to complete in an on-premises implementation.

Ideal Usage Pattern

Amazon Redshift is ideal for online analytical processing (OLAP) using your existing business intelligence tools. Organizations are using Amazon Redshift to do the following:

- Analyze global sales data for multiple products
- Store historical stock trade data
- Analyze ad impressions and clicks
- Aggregate gaming data
- Analyze social trends

---

[1] http://aws.amazon.com/about-aws/globalinfrastructure/

[2] http://aws.amazon.com/s3/

[3] http://aws.amazon.com/datapipeline/

[4] http://aws.amazon.com/solutions/case-studies/big-data/

[5] http://aws.amazon.com/redshift/

- Measure clinical quality, operation efficiency, and financial performance in health care

Cost Model

An Amazon Redshift data warehouse cluster requires no long-term commitments or upfront costs. This frees you from the capital expense and complexity of planning and purchasing data warehouse capacity ahead of your needs. Charges are based on the size and number of nodes of your cluster. There is no additional charge for backup storage up to 100% of your provisioned storage. For example, if you have an active cluster with 2 XL nodes for a total of 4 TB of storage, AWS will provide up to 4 TB of backup storage on to Amazon S3 at no additional charge. Backup storage beyond the provisioned storage size, and backups stored after your cluster is terminated, are billed at standard Amazon S3 rates.[6] There is no data transfer charge for communication between Amazon S3 and Amazon Redshift. For details about Amazon Redshift pricing go to the pricing web page.[7]

Performance

Amazon Redshift uses a variety of innovations to obtain very high performance on datasets ranging in size from hundreds of gigabytes to a petabyte or more. It uses columnar storage, data compression, and zone maps to reduce the amount of I/O needed to perform queries. Amazon Redshift has a massively parallel processing (MPP) architecture, parallelizing and distributing SQL operations to take advantage of all available resources. The underlying hardware is designed for high-performance data processing, using local attached storage to maximize throughput between the Intel Xeon E5 processor and drives, and a 10GigE mesh network to maximize throughput between nodes. Performance can be tuned based on your data warehousing needs; AWS offers high density compute with solid-state drives (SSDs) as well as high-density storage options.

Durability and Availability

An Amazon Redshift cluster will automatically detect and replace a failed node in your data warehouse cluster. The data warehouse cluster will be read-only until a replacement node is provisioned and added to the database, which typically only takes a few minutes. Amazon Redshift makes your replacement node available immediately and streams your most frequently accessed data from Amazon S3 first to allow you to resume querying your data as quickly as possible. Additionally, your Amazon Redshift data warehouse cluster will remain available in the event of a drive failure. Because Amazon Redshift mirrors your data across the cluster, it will use the data from another node to rebuild failed drives. Amazon Redshift clusters reside within one Availability

[6] http://aws.amazon.com/s3/pricing/

[7] http://aws.amazon.com/redshift/pricing/

Zone (AZ).[8] If you want to have a multi-AZ setup for Amazon Redshift you can set up a mirror, and then self-manage replication and failover.

Scalability and Elasticity

With a few clicks in the AWS Management Console,[9] or a simple API call,[10] you can easily change the number or type of nodes in your data warehouse as your performance or capacity needs change. Using Amazon Redshift you can start with as little as a single 160 GB node and scale up all the way to a petabyte or more of compressed user data using many nodes. To learn more about Amazon Redshift node types go to the Amazon Redshift documentation.[11] While resizing, Amazon Redshift places your existing cluster into read-only mode, provisions a new cluster of your chosen size, and then copies data from your old cluster to your new one in parallel. During this process you only pay for the active Amazon Redshift cluster. You can continue running queries against your old cluster while the new one is being provisioned. After your data has been copied to your new cluster, Amazon Redshift will automatically redirect queries to your new cluster and remove the old cluster.

Interfaces

Amazon Redshift uses industry-standard SQL and is accessed using standard JDBC, or ODBC, drivers (for more information on Amazon Redshift drivers, go to the documentation).[12] There are numerous examples of validated integrations with many popular business intelligence (BI) and extract, transform, and load (ETL) vendors.[13] Loads and unloads are attempted in parallel into each compute node to maximize the rate at which you can ingest data into your data warehouse cluster as well as to and from Amazon S3 and Amazon DynamoDB. Metrics for compute utilization, memory utilization, storage utilization, and read/write traffic to your Amazon Redshift data warehouse cluster are available for no additional charge via the AWS Management Console or Amazon CloudWatch APIs.[14]

Anti-Patterns

- **Small data sets—**Amazon Redshift is built for parallel processing across a cluster. If your data set is less than a hundred gigabytes, you are not going to get

[8] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html

[9] http://aws.amazon.com/console/

[10] http://docs.aws.amazon.com/redshift/latest/APIReference/Welcome.html

[11] http://docs.aws.amazon.com/redshift/latest/mgmt/working-with-clusters.html#rs-about-clusters-and-nodes

[12] http://docs.aws.amazon.com/redshift/latest/dg/c_redshift-and-postgres-sql.html

[13] http://aws.amazon.com/redshift/partners/

[14] http://aws.amazon.com/cloudwatch/

all the benefits that Amazon Redshift has to offer; Amazon Relational Database Service (Amazon RDS) may be a better solution.[15]

- **Online transaction processing (OLTP)—**Amazon Redshift is designed for data warehouse workloads producing extremely fast and inexpensive analytic capabilities. If your requirement is for a fast transactional system, then a traditional relational database system built on Amazon RDS, or a NoSQL database offering functionality such as Amazon DynamoDB, is a better choice.

- **Unstructured data—**Data in Amazon Redshift must be structured by a defined schema, rather than supporting arbitrary schema structure for each row. If your data are unstructured you can perform extract, transform, and load (ETL) processes on Amazon Elastic MapReduce (Amazon EMR) to get the data ready for loading into Amazon Redshift.

- **Binary large object (BLOB) data—**If you plan on storing large data files (such as BLOB data, which includes digital video, images, or music), you'll want to consider storing the data in Amazon S3 and referencing its location in Amazon Redshift. In this scenario, Amazon Redshift keeps track of metadata (e.g., item name, size, date created, owner, and location) about your binary objects, but the large objects themselves would be stored in Amazon S3.

Amazon Kinesis

Amazon Kinesis provides an easy way to process fast moving data in real time.[16] You simply provision the level of input and output required for your data stream, in blocks of one megabyte per second (MB/sec), using the AWS Management Console, API,[17] or SDKs.[18] The size of your stream can be adjusted up or down at any time without restarting the stream and without any impact on the data sources pushing data to Amazon Kinesis. Within 10 seconds, data put into an Amazon Kinesis stream is available for analysis. Stream data is stored across multiple Availability Zones in a region for 24 hours. During that window, data is available to be read, re-read, backfilled, and analyzed, or moved to long-term storage (e.g., Amazon S3 or Amazon Redshift). The Amazon Kinesis client library enables developers to focus on creating their business applications while removing the undifferentiated heavy lifting associated with load-balancing streaming data, coordinating distributed services and handling fault-tolerant data processing.

[15] http://aws.amazon.com/rds/

[16] http://aws.amazon.com/kinesis/

[17] http://docs.aws.amazon.com/kinesis/latest/APIReference/Welcome.html

[18] http://docs.aws.amazon.com/aws-sdk-php/guide/latest/service-kinesis.html

Ideal Usage Pattern

Amazon Kinesis is useful wherever there is a need to move data rapidly off producers (i.e., data sources) as it is produced, and then to continuously process it. There are many different purposes for continuous processing including transforming the data before emitting it into another data store, driving real-time metrics and analytics, or deriving and aggregating multiple Amazon Kinesis streams into either more complex Kinesis streams or into downstream processing. The following are typical scenarios for using Amazon Kinesis for analytics.

- **Real-time data analytics**: Amazon Kinesis enables real-time data analytics on streaming data, such as analyzing website clickstream data and customer engagement analytics.
- **Log and data feed intake and processing**: With Amazon Kinesis producers can push data directly into an Amazon Kinesis stream. For example, system and application logs can be submitted to Amazon Kinesis and be available for processing in seconds. This prevents the log data from being lost if the front-end, or application server, fails and reduces local log storage on the source. Amazon Kinesis provides accelerated data intake because you are not batching up the data on the servers before you submit the data for intake.
- **Real-time metrics and reporting**: You can use data ingested into Amazon Kinesis for extracting metrics, and generating KPIs to power reports and dashboards, at real-time speeds. This enables data-processing application logic to work on data as it is streaming in continuously, rather than waiting for data batches to be sent to the data-processing applications.

Performance

An Amazon Kinesis stream is made up of one or more shards; each shard gives you a capacity of 5 read transactions per second, up to a maximum total of 2 MB of data read per second. Each shard can support up to 1000 write transactions per second and up to a maximum total of 1 MB data written per second. The data capacity of your stream is a function of the number of shards that you specify for the stream. The total capacity of the stream is the sum of the capacities of its shards. You can provision as many shards as you need to get the throughput capacity you want, for instance a 1 gigabyte per second data stream would require 1024 shards.

Cost Model

Amazon Kinesis has simple pay-as-you-go pricing, with no up-front costs and no minimum fees. You'll only pay for the resources you consume. There are just two pricing components, an hourly charge per shard and a charge for each 1 million PUT transactions. For detailed pricing information, go to the Amazon Kinesis pricing page.[19]

[19] http://aws.amazon.com/kinesis/pricing/

Applications running on Amazon Elastic Compute Cloud (EC2) that process the Amazon Kinesis stream would also incur standard Amazon EC2 pricing.

Durability and Availability

Amazon Kinesis synchronously replicates data across three Availability Zones in an AWS region, providing high availability and data durability. Additionally, a cursor in Amazon DynamoDB can be used to follow what has been read from Amazon Kinesis in Amazon DynamoDB. In the event of system failure in any system that was in the middle of reading, the cursor is designed to keep the spot where the reading left off in durable storage so that a healthy system can pick up from the exact spot where the failed system left off.

Scalability and Elasticity

You can increase or decrease the capacity of the stream at any time according to your business or operational needs, without any interruption to ongoing stream processing. By utilizing simple API calls or development tools you can automate scaling of your Amazon Kinesis environment to meet demand and ensure that you only pay for what you need.

Interfaces

There are two interfaces to Kinesis: the first is the input interface, which data producers use to put data into Amazon Kinesis, and the second is the output interface, which is used to process and analyze the data that comes in. Producers write data using a simple PUT via an API directly or abstracted by an AWS Software Development Kit (SDK) or toolkit.[20]

For processing data that has already been put into Amazon Kinesis client libraries are provided to build and operate real-time streaming data processing applications. The Amazon Kinesis Client Library (KCL) acts as an intermediary between your business applications, which contain the specific logic needed to process Amazon Kinesis stream data, and the Amazon Kinesis service itself.[21] There is also integration to read from a Kinesis stream into Apache Storm via the Kinesis Storm Spout.[22]

Anti-Patterns

- **Small-scale consistent throughput—**If you are streaming data, throughput is a consistent couple hundred kilobytes per second or less. While it will technically

[20] http://aws.amazon.com/tools/

[21] https://github.com/awslabs/amazon-kinesis-client

[22] https://github.com/awslabs/kinesis-storm-spout

function just fine with such a small amount of data, Amazon Kinesis is designed and optimized for large amounts of data.

- **Long-term data storage and analytics—**Amazon Kinesis is a robust ingestion platform, not a storage platform, thus data is only kept for a rolling 24-hour period. The design pattern to follow to store data that you want to keep or analyze for longer than a 24-hour period is to move the data into another durable storage service such as Amazon S3,[23] Amazon Glacier,[24] Amazon Redshift,[25] or Amazon DynamoDB.[26]

Amazon Elastic MapReduce

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data.[27] Amazon EMR uses Apache Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances. Hadoop provides a framework to run big data processing and analytics; Amazon EMR does all the heavily lifting involved with provisioning, managing, and maintaining the infrastructure and software of a Hadoop cluster.

Ideal Usage Pattern

Amazon EMR's flexible framework reduces large processing problems and data sets into smaller jobs and distributes them across many compute nodes in a Hadoop cluster. This capability lends itself to many usage patterns with big data analytics. Here are a few examples:

- Log processing and analytics
- Large extract, transform, and load (ETL) and data movement
- Risk modeling and threat analytics
- Ad targeting and click stream analytics
- Genomics
- Predictive analytics
- Ad-hoc data mining and analytics

For a more in-depth discussion on Amazon EMR, go to the Amazon EMR Best Practices whitepaper.[28]

---

[23] http://aws.amazon.com/s3/

[24] http://aws.amazon.com/glacier/

[25] http://aws.amazon.com/redshift/

[26] http://aws.amazon.com/dynamodb/

[27] http://aws.amazon.com/elasticmapreduce/

[28] https://media.amazonwebservices.com/AWS_Amazon_EMR_Best_Practices.pdf

Cost Model

With Amazon EMR you can launch a persistent cluster that stays up indefinitely or a temporary cluster that terminates after the analysis is complete. In either scenario you only pay for the hours the cluster is up. Amazon EMR supports a variety of Amazon EC2 instance types (standard, high CPU, high memory, high I/O, etc.) and all Amazon EC2 instance pricing options (On-Demand, Reserved, and Spot). When you launch an Amazon EMR cluster (also called a "job flow"), you choose how many, and what type, of Amazon EC2 Instances to provision. The Amazon EMR price is in addition to the Amazon EC2 price. For more detailed information, go to the Amazon EMR pricing page.[29]

Performance

Amazon EMR performance is driven by the type of EC2 instances you choose to run your cluster on. You should choose an instance type suitable for your processing requirements—memory, storage, or processing power. For more information on EC2 instance specifications, go to the Amazon EC2 instance types web page.[30]

Durability and Availability

By default, Amazon EMR is fault tolerant for core node failures and continues job execution if a slave node goes down. Currently Amazon EMR does not automatically provision another node to take over failed slaves, but using Amazon CloudWatch customers can monitor the health of nodes and replace failed nodes.[31]

To help handle the unlikely event of a master node failure we recommend that you back up your data to a persistent store such as Amazon S3. Optionally you can choose to run Amazon EMR with the MapR distribution, which provides a no-NameNode architecture that can tolerate multiple simultaneous failures with automatic failover and fallback.[32] The metadata is distributed and replicated, just like the data. With a no-NameNode architecture, there is no practical limit to how many files can be stored, and also no dependency on any external network-attached storage.

Scalability and Elasticity

With Amazon EMR, it is easy to resize a running cluster.[33] You can add core nodes which hold the Hadoop Distributed File System (HDFS) at any time to increase your processing power and increase HDFS storage capacity (and throughput). Additionally, you can use Amazon S3 natively, or you can use Elastic Map Reduce File System (EMRFS) along with or instead of local HDFS. This allows you to decouple your memory

---

[29] http://aws.amazon.com/elasticmapreduce/pricing/

[30] http://aws.amazon.com/ec2/instance-types/

[31] http://aws.amazon.com/cloudwatch/

[32] http://aws.amazon.com/elasticmapreduce/mapr/

[33] http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-manage-resize.html

and compute from your storage, providing greater flexibility and cost efficiency.[34] At any time, you can add and remove task nodes that can process Hadoop jobs, but that do not maintain HDFS. Some customers add hundreds of instances to their clusters when their batch processing occurs, and remove the extra instances when processing completes. For example, you might not know how much data your clusters will be handling in six months, or you might have spikey processing needs. With Amazon EMR you don't need to guess your future requirements or provision for peak demand because you can easily add or remove capacity at any time.

Additionally, you can add all new clusters of various sizes and remove them at any time with a few clicks in the AWS Management Console or a by a programmatic API call.[35]

Interfaces

Amazon EMR supports many tools on top of Hadoop that can be used for big data analytics and each tool has its own interface. Here is a brief summary of the most popular options:

Hive is an open source data warehouse and analytics package that runs on top of Hadoop. Hive is operated by HiveQL, a SQL-based language which allows users to structure, summarize, and query data. HiveQL goes beyond standard SQL, adding first-class support for map/reduce functions and complex extensible user-defined data types like JSON and Thrift. This capability allows processing of complex and unstructured data sources, such as text documents and log files. Hive allows user extensions via user-defined functions written in Java. Amazon EMR has made numerous improvements to Hive, including direct integration with Amazon DynamoDB and Amazon S3. For example, with Amazon EMR you can load table partitions automatically from Amazon S3, write data to tables in Amazon S3 without using temporary files, and access resources in Amazon S3, such as scripts for custom map and/or reduce operations and additional libraries. See the Amazon EMR documentation to learn more about analyzing data with Hive.[36]

Pig is an open source analytics package that runs on top of Hadoop. Pig is operated by Pig Latin, a SQL-like language that allows users to structure, summarize, and query data. In addition to SQL-like operations, Pig Latin also adds first-class support for map/reduce functions and complex extensible user-defined data types. This capability allows processing of complex and unstructured data sources, such as text documents and log files. Pig allows user extensions via user-defined functions written in Java. Amazon EMR has made numerous improvements to Pig, including the ability to use multiple file systems (normally Pig can only access one remote file system), the ability to load customer JARs and scripts from Amazon S3 (e.g., "REGISTER s3://my-bucket/piggybank.jar"), and additional functionality for String

[34] http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-plan-file-systems.html

[35] http://docs.aws.amazon.com/ElasticMapReduce/latest/API/Welcome.html

[36] http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-hive.html

and DateTime processing. See the Amazon EMR documentation to learn more about processing data with Pig.[37]

Spark is an open source data analytics engine built on Hadoop with the fundamentals for in-memory MapReduce. Spark provides additional speed for certain analytics and is the foundation for other power tools such as Shark (SQL driven data warehousing), Spark Streaming (streaming applications), GraphX (graph systems), and MLlib (machine learning). See the Amazon blogs to learn how to run Spark on Amazon EMR.[38]

HBase is an open source, non-relational, distributed database modeled after Google's BigTable. It was developed as part of Apache Software Foundation's Hadoop project and runs on top of HDFS to provide BigTable-like capabilities for Hadoop. HBase provides you a fault-tolerant, efficient way of storing large quantities of sparse data using column-based compression and storage. In addition, HBase provides fast lookup of data because data is stored in-memory instead of on disk. HBase is optimized for sequential write operations, and it is highly efficient for batch inserts, updates, and deletes. HBase works seamlessly with Hadoop, sharing its file system and serving as a direct input and output to Hadoop jobs. HBase also integrates with Apache Hive, enabling SQL-like queries over HBase tables, joins with Hive-based tables, and support for Java Database Connectivity (JDBC). With Amazon EMR, you can back up HBase to Amazon S3 (full or incremental, manual or automated) and you can restore from a previously created backup. See the Amazon EMR documentation to learn more about HBase and Amazon EMR.[39]

Impala is an open source tool in the Hadoop ecosystem for interactive, ad hoc querying using SQL syntax. Instead of using MapReduce, it leverages a massively parallel processing (MPP) engine similar to that found in traditional relational database management systems (RDBMS). With this architecture, you can query your data in HDFS or HBase tables very quickly, and leverage Hadoop's ability to process diverse data types and provide schema at runtime. This makes Impala a great tool to perform interactive, low-latency analytics. Impala also has user-defined functions in Java and C++, and can connect to business intelligence (BI) tools through ODBC and JDBC drivers. Impala uses the Hive metastore to hold information about the input data, including the partition names and data types. See the Amazon EMR documentation to learn more about Impala and Amazon EMR.[40]

Hunk was developed by Splunk to make machine data accessible, usable, and valuable to everyone. With Hunk you can interactively explore, analyze, and visualize data stored in Amazon EMR and Amazon S3, harnessing Splunk analytics

---

[37] http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-pig.html

[38] http://blogs.aws.amazon.com/bigdata/post/Tx15AY5C50K70RV/Installing-Apache-Spark-on-an-Amazon-EMR-Cluster

[39] http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-hbase.html

[40] http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-impala.html

amazon web services

on Hadoop. Go to the Amazon EMR website to learn more about Hunk and Amazon EMR.[41]

Other: Amazon EMR also supports a variety of other popular applications and tools in the Hadoop ecosystem, such as R (statistics), Mahout (machine learning), Ganglia (monitoring), Presto (distributed SQL engine), Accumulo (secure NoSQL database), Hue (user interface to analyze Hadoop data), Sqoop (relational database connector), HCatalog (table and storage management), and more.

Additionally you can install your own software on top of Amazon EMR to help solve your business needs. AWS provides the ability to quickly move large amounts of data from Amazon S3 to HDFS, from HDFS to Amazon S3, and between Amazon S3 buckets using Amazon EMR's S3DistCp, an extension of the open source tool DistCp, which uses MapReduce to efficiently move large amounts of data.[42]

Anti-Patterns

Amazon EMR has the following anti-patterns:

- **Small data sets—**Amazon EMR is built for massive parallel processing; if your data set is small enough to run quickly on a single machine in a single thread, the added overhead to map and reduce jobs may not worth it for small data sets that can easily be processed in memory on a single system.
- **ACID (atomicity, consistency, isolation, durability) transaction requirements—**While there are ways to achieve ACID or limited ACID on Hadoop, if this requirement is very stringent another database, such as Amazon RDS or a relational database running on Amazon EC2, might be a better option for workloads with this profile.

Amazon DynamoDB

Amazon DynamoDB is a fast, fully-managed NoSQL database service that makes it simple and cost-effective to store and retrieve any amount of data, and serve any level of request traffic. Amazon DynamoDB helps offload the administrative burden of operating and scaling a highly available distributed database cluster. This storage alternative meets the latency and throughput requirements of highly demanding applications by providing single-digit millisecond latency and predictable performance with seamless throughput and storage scalability.

Amazon DynamoDB stores structured data in tables, indexed by primary key, and allows low-latency read and write access to items ranging from 1 byte up to 400 KB. Amazon DynamoDB supports three data types: number, string, and binary, in both scalar and multi-valued sets. Tables do not have a fixed schema, so each data item can have a

[41] http://aws.amazon.com/elasticmapreduce/hunk/

[42] http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/UsingEMR_s3distcp.html

different number of attributes. The primary key can either be a single-attribute hash key or a composite hash-range key. The global and local secondary indexes that DynamoDB offers provide additional flexibility for querying against attributes other than the primary key. Amazon DynamoDB provides both eventually-consistent reads (by default), and strongly-consistent reads (optional), as well as implicit item-level transactions for item put, update, delete, and conditional operations, and increment/decrement.

Amazon DynamoDB is integrated with other services, such as Amazon EMR, Amazon Redshift, AWS Data Pipeline, and Amazon S3, for analytics, data warehouse, data import/export, backup, and archive.

Ideal Usage Pattern

Amazon DynamoDB is ideal for existing or new applications that need a flexible NoSQL database with low read and write latencies, and the ability to scale storage and throughput up or down as needed without code changes or downtime.

Common use cases include: mobile apps, gaming, digital ad serving, live voting, audience interaction for live events, sensor networks, log ingestion, access control for web-based content, metadata storage for Amazon S3 objects, e-commerce shopping carts, and web session management. Many of these use cases require a highly available and scalable database because downtime or performance degradation has an immediate negative impact on an organization's business.

Cost Model

With Amazon DynamoDB, you pay only for what you use and there is no minimum fee. Amazon DynamoDB has three pricing components: provisioned throughput capacity (per hour), indexed data storage (per GB per month), and data transfer in or out (per GB per month). New customers can start using Amazon DynamoDB for free as part of the AWS Free Usage Tier.[43] Pricing information can be found on the Amazon DynamoDB pricing web page.[44]

Performance

The use of SSDs and limiting indexing on attributes provides high throughput and low latency[45] and drastically reduces the cost of read and write operations. As the datasets grow, predictable performance is required so that low-latency for the workloads can be maintained. This predictable performance can be achieved by defining the provisioned throughput capacity required for a given table. Behind the scenes, the service handles the provisioning of resources to achieve the requested throughput rate, which takes the burden away from you to have to think about instances, hardware, memory, and other

[43] http://aws.amazon.com/free/

[44] http://aws.amazon.com/dynamodb/pricing/

[45] Single-digit milliseconds typical for average server-side response times

factors that can affect an application's throughput rate. Provisioned throughput capacity reservations are elastic and can be increased or decreased on demand.

Durability and Availability

Amazon DynamoDB has built-in fault tolerance that automatically and synchronously replicates data across three data centers in a region for high availability and to help protect data against individual machine, or even facility, failures.

Scalability and Elasticity

Amazon DynamoDB is both highly-scalable and elastic. There is no limit to the amount of data you can store in an Amazon DynamoDB table, and the service automatically allocates more storage as you store more data using Amazon DynamoDB write API calls. Data is automatically partitioned and re-partitioned as needed, while the use of SSDs provides predictable low-latency response times at any scale. The service is also elastic, in that you can simply "dial-up"[46] or "dial-down"[47] the read and write capacity of a table as your needs change.

Interfaces

Amazon DynamoDB provides a low-level REST API, as well as higher-level SDKs for Java, .NET, and PHP that wrap the low-level REST API and provide some object-relational mapping (ORM) functions. These APIs provide both a management and data interface for Amazon DynamoDB. The API currently offers operations that enable table management (creating, listing, deleting, and obtaining metadata) and working with attributes (getting, writing, and deleting attributes; query using an index; and full scan). Although standard SQL isn't available for Amazon DynamoDB, you can use the Amazon DynamoDB select operation to create SQL-like queries that retrieve a set of attributes based on criteria that you provide. You can also work with Amazon DynamoDB using the AWS Management Console.

Anti-Patterns

Amazon DynamoDB has the following anti-patterns:

- **Prewritten application tied to a traditional relational database**—If you are attempting to port an existing application to the AWS Cloud and need to continue using a relational database, you can elect to use either Amazon RDS (MySQL, PostgreSQL, Oracle, or SQL Server), or one of the many preconfigured Amazon

[46] Amazon DynamoDB allows you to change your provisioned throughput level by up to 100% with a single UpdateTable API call. If you want to increase your throughput by more than 100%, you can simply call UpdateTable again.

[47] You can increase your provisioned throughput as often as you want, however there is a limit of two decreases per day.

EC2 database Amazon Machine Images (AMIs). You can also install your choice of database software on an Amazon EC2 instance that you manage.

- **Joins and/or complex transactions**—Although many solutions are able to leverage Amazon DynamoDB to support their users, it's possible that your application may require joins, complex transactions, and other relational infrastructure provided by traditional database platforms. If this is the case, you might want to explore Amazon Redshift, Amazon RDS, or Amazon EC2 with a self-managed database.

- **BLOB data**—If you plan on storing large (greater than 400 KB) BLOB data, such as digital video, images, or music, you'll want to consider Amazon S3. However, Amazon DynamoDB still has a role to play in this scenario, for keeping track of metadata (e.g., item name, size, date created, owner, and location) about your binary objects.

- **Large data with low I/O rate**—Amazon DynamoDB uses SSD drives and is optimized for workloads with a high I/O rate per GB stored. If you plan to store very large amounts of data that are infrequently accessed, other storage options, such as Amazon S3, might be a better choice.

Application on Amazon EC2

Amazon EC2 provides an ideal platform for you to operate your own self-managed big data analytics on AWS's world class infrastructure.[48] Almost any software you can install on Linux or Windows virtualized environments can be run on Amazon EC2. You get the pay-as-you-go pricing model by running applications on Amazon EC2. What you don't get is application-level managed services that come with the other options mentioned in this paper. There are many options for self-managed big data analytics; the following are a few examples:

- Manage your own NoSQL offering such as MongoDB
- Manage your own data warehouse or columnar store like Vertica
- Manage your own Hadoop cluster
- Manage your own Apache Storm cluster
- Manage your own Apache Kafka environment

[48] http://aws.amazon.com/ec2/

Ideal Usage Pattern

- **Specialized environment:** If you are running a custom application, a variation of a standard Hadoop set, or an application not covered by one of our other offerings then Amazon EC2 provides the flexibility and scalability to meet your computing needs.

- **Compliance requirements:** Certain compliance requirements might require you to run applications yourself on Amazon EC2 instead of using a managed service offering.

Cost Model

Amazon EC2 has a variety of instance types in a number of instance families (standard, high CPU, high memory, high I/O, etc.). There are different pricing options (On-Demand, Reserved, and Spot) for instances. Depending on your application requirements you might want to use additional services along with Amazon EC2, such as Amazon Elastic Block Store (EBS) for directly attached persistent storage or Amazon S3 as a durable object store; each comes with its own pricing model. If you run your big data application on Amazon EC2 you will be responsible for any license fees, just as you would in your own data center. The AWS Marketplace offers many different third-party big data software packages preconfigured to launch with a simple click of a button.[49]

Performance

Performance in Amazon EC2 is driven by the instance type that you choose for your big data platform. Each instance type has a different amount of CPU, RAM, storage, input/output operations per second (IOPS), and networking capability so that you can pick the right performance level for your application requirements.

Durability and Availability

Critical applications should be run in a cluster across multiple Availability Zones within an AWS region so that any instance or data center failure will not affect users of the application. For non-uptime critical applications you can back up your application to Amazon S3 and restore from Amazon S3 to any Availability Zone in the region if an instance or Availability Zone failure occurs. Other options exist depending on which application you are running and your requirements, such as mirroring your application.

Scalability and Elasticity

Auto Scaling is a service that allows you to automatically scale your Amazon EC2 capacity up or down according to conditions you define.[50] With Auto Scaling, you can ensure that the number of Amazon EC2 instances you're using scales up seamlessly during demand spikes to maintain performance, and scales down automatically during

[49] https://aws.amazon.com/marketplace

[50] http://aws.amazon.com/autoscaling/

demand lulls to minimize costs. Auto Scaling is particularly well suited for applications that experience hourly, daily, or weekly variability in usage. Auto Scaling is enabled by Amazon CloudWatch and available at no additional charge beyond Amazon CloudWatch fees.

Interfaces

Amazon EC2 can be interfaced programmatically using the API, SDK, or the web-based AWS Management Console. Metrics for compute utilization, memory utilization, storage utilization, network consumption, and read/write traffic to your instances are no additional charge via the AWS Management Console or Amazon CloudWatch APIs.[51]

The interfaces for your big data analytic software that you run on top of Amazon EC2 will vary based on the characteristics of the software you choose.

Anti-Patterns

- **Managed service—**If your requirement is a managed service offering where you abstract the infrastructure layer and administration, from the big data analytics offering then this "do it yourself" model of managing your own analytics software on Amazon EC2 might not be the correct choice for your use case.

- **Lack of expertise or resources—**If your organization does not have, or does not want to expend, the resources or expertise to install and manage a high availability system, you should consider using an AWS equivalent managed system such as the following:

  Amazon EMR, Amazon DynamoDB, Amazon Kinesis, or Amazon Redshift

Solving Big Data Problems Using AWS

This white paper has examined some of the AWS tools at your disposal for big data analytics. However, when should you choose one tool over the other? How do you design a good architecture for a specific problem statement? Your analytical workload will have certain characteristics and requirements which will dictate which tool to use. Consider the following:

- How quickly do you need analytic results—in real-time, in seconds, or is an hour a more appropriate time frame?
- How much value will these analytics provide your organization, and what budget constraints exist?
- How large is the data, and what is its growth rate?
- How is the data structured?
- What integration capabilities do the producers and consumers have?
- How much latency is acceptable between the producers and consumers?

[51] http://aws.amazon.com/cloudwatch/

- What is the cost of downtime, or how available and durable does the solution need to be?
- Is the analytic workload consistent or elastic?

Each one of these characteristics or requirements will help point in you in the right direction for which tool to use. The following figure depicts the AWS big data analytics options side-by-side against different requirements:

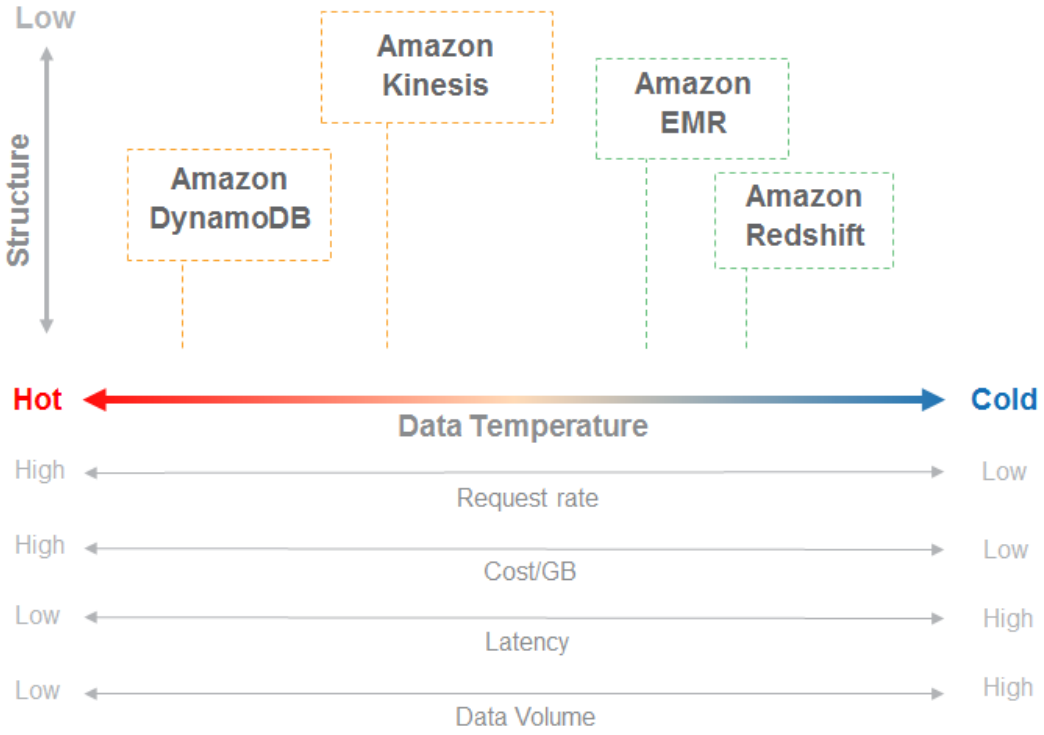

Figure 1: Big Data Analytics Tools on AWS

In some cases, you can simply map your big data analytics workload easily into a service based on a set of requirements. However, in most real-world big data analytic workloads, there are many different, and sometimes conflicting, characteristics and requirements on the same data set. For example, some result sets may have real-time requirements as a user interacts with a system while other analytics could be batched and run on a daily basis. These different requirements over the same data set should be decoupled and solved by using more than one tool. If you try to satisfy these different requirements using the same toolset you end up either over-provisioning and therefore overpaying for response time you do not need, or you have a solution that is not quick enough to respond to your users in real-time. By matching the best-suited tool to each

individual analytical problem set the results will be the most cost-effective use of your compute and storage resources.

A common misconception is that having multiple tool sets to solve a big data problem will be more expensive or harder to manage than having one big tool. If we take the same example of two different requirements on the same data set, the real-time request may be low on CPU but high on I/O, while the slower processing request may be very compute intensive. Decoupling can end up being much less expensive and easier to manage because you can build each tool to exact specifications and not overprovision. With AWS you pay for what you use. This infrastructure as a service model equates to a good value because you could run the batch analytics in just one hour and therefore only have to pay for the compute resources for that hour. Also, you might find this approach easier to manage than leveraging a single system that tries to meet all requirements. Solving for different requirements with one tool results in attempting to fit a square peg (such as real-time requests) into a round hole (such as a large data warehouse).

The AWS platform makes it easy to decouple your architecture and have different tools analyze the same data set. AWS services have built-in integration so that moving a subset of data from one tool to another can be done very easily and quickly using parallelization. The following examples put this into practice by exploring a couple of real-world big data analytics problem scenarios and walking through what an architecture would look like on AWS to solve each problem.

Example 1: Enterprise Data Warehouse

A multinational clothing maker has over a thousand retail stores, sells certain lines through department and discount stores, and has an online presence. Currently these three channels are independent of each other. They have different management, point of sale systems, and accounting departments. Currently there is no system that merges all of these data sets together to allow the CEO to have insight across the business. The CEO wants to have a company-wide picture of its channels, and she wants the ability to do ad-hoc analytics when required. Some of the analytics questions the company needs answers to include the following:

- What trends exist across channels?
- What geographic regions do better across channels?
- How effective are the company's advertisements and coupons?
- What trends exist across each clothing line?
- What external forces might have an impact on its sales—for example, the unemployment rate or the weather?

- How do store attributes affect sales—for example, the tenure of employees/management, strip mall versus enclosed mall, location of merchandise in the store, promotion, endcaps, sales circulars, in-store displays, etc.?

You might consider using an enterprise data warehouse to solve this problem, but how would you implement one on AWS? This data warehouse will need to collect data from each of the three channel's various systems and from public records for weather and economic data. Each data source will send its data on a daily basis for consumption by the data warehouse. Because each data source may be structured differently, an ETL process will be performed to reformat the data into a common structure. Then, analytics can be performed across the data from all sources simultaneously. To do this, we will use the following data flow architecture:

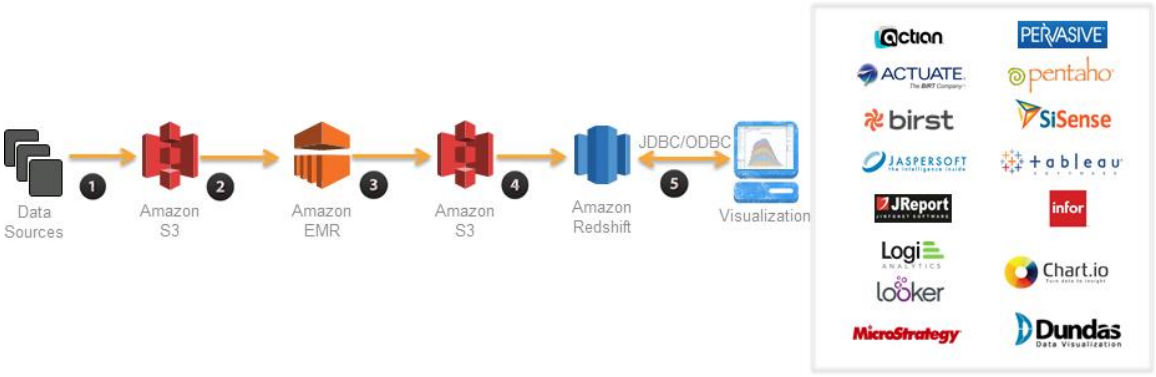

Figure 2: Data Warehouse Data Flow Architecture

1. The first step in this process is getting the data from the many different sources onto Amazon S3. Amazon S3 was chosen because it's a highly durable, inexpensive, and scalable storage platform that can be written to in parallel from many different sources at a very low cost.

2. Amazon EMR will be used to transform and cleanse the data from the source format into the destination format. Amazon EMR has built in integration with Amazon S3 to allow parallel threads of throughput from each node in your cluster to and from Amazon S3. Typically data warehouse workloads get sent data on a nightly basis from their many different sources. Since there is no need for these analytics in the middle of the night, the only requirement for this transformation process is to finish by morning, when the CEO needs

results. This means that you can utilize the Amazon EC2 Spot Market[52] to further bring down the cost of transformation. A good Spot strategy could be to start bidding at a very low price at midnight, and continually increase your price over time until capacity is granted. As you get closer to the deadline, if Spot bids have not succeeded, you can fall back to On-Demand prices to ensure that you still meet your completion time requirements. Each source may have a different transformation process on Amazon EMR, but with AWS's pay-as-you-go model you can create a separate Amazon EMR cluster for each transformation and tune it to complete this transformation for the lowest possible price without contending with the resources of the other jobs.

3. Each transformation job will then put the formatted and cleaned data onto Amazon S3. Amazon S3 is used here because Amazon Redshift can then consume this data on multiple threads in parallel from each node. This location on Amazon S3 also serves as a historical record and is the formatted source of truth between systems. Data on Amazon S3 can be consumed by other tools for analytics if additional requirements are introduced over time.

4. Amazon Redshift will load, sort, distribute, and compress the data into it its tables so that analytical queries can execute efficiently and in parallel. Amazon Redshift is built for data warehouse workloads and can easily be grown by adding more nodes as the data size increases over time and the business expands.

5. For visualizing the analytics, the company could use one of the many partner visualization platforms via the OBDC/JDBC connection to Amazon Redshift. This allows reports and graphs to be viewed by the CEO and her staff. Executives can use the data to make informed decisions about how to utilize company resources, which could ultimately increase earnings and value for shareholders.

This architecture is very flexible and can easily be expanded if the business expands, more data sources are imported, new channels are opened, or a mobile application is launched with customer-specific data. At any time additional tools can be integrated, and the warehouse can be scaled in a few clicks by increasing the number of nodes in the Amazon Redshift cluster.

Example 2: Capturing and Analyzing Sensor Data

An international manufacturer of air conditioners sells many large air conditioners to various commercial and industrial companies. In addition to selling air conditioner units, the company offers add-on services, so customers can see real-time dashboards in a mobile application or a web browser. Each unit sends its sensor information for

[52] http://aws.amazon.com/ec2/purchasing-options/spot-instances/

processing and analysis. This data lets both customers and the manufacturer know what is going on. With this capability the manufacturer can predict possible problems before they occur or know when a system needs urgent maintenance so it can send out a mechanic before the customer even realizes there is a problem.

Currently the manufacturer has a few thousand pre-purchased units with this capability. It expects to deliver these units to customers in the next couple of months, and it hopes that in time thousands of units throughout the world will be using this platform. If successful the company would like to expand this offering to their consumer line as well with a much larger volume and a greater market share. The solution will need to handle massive amounts of data and to scale without interruption as the business grows.

How should you design such a system? First you should break it up into two work streams, both originating from the same data:

- The air conditioning unit's current information with near real-time requirements with a large number of customers consuming this information.
- All historical information on the air conditioning units for use internally to run trending and analytics for internal use.

The following diagram illustrates the data-flow architecture for solving this big data problem:

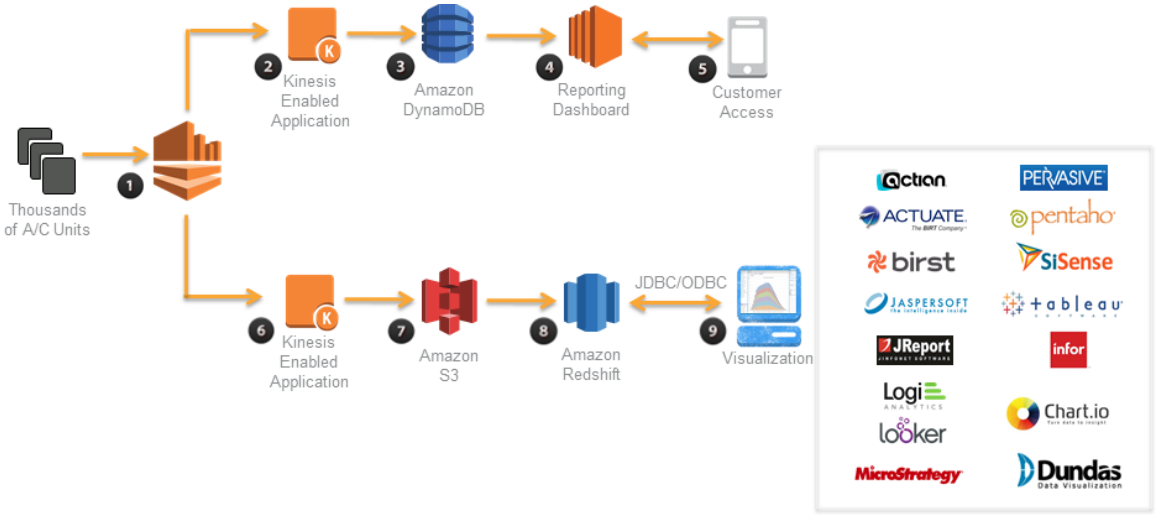

Figure 3: Collecting and Analyzing Sensor Data

1. The process begins with each air conditioner unit providing a constant data stream to Amazon Kinesis. This provides an elastic and durable interface that can be scaled seamlessly as more and more units are sold and brought online.

2. Using the Amazon Kinesis-provided tools such as the Kinesis Client Library or SDK, a simple application is built on Amazon EC2 to read data as it comes into Amazon Kinesis, and then to analyze it and determine if the data warrants an update to the real-time dashboard. The application will be looking for changes in system operation, temperature fluctuations, and any errors the units encounter.

3. This data flow will need to occur in near real-time so that customers and maintenance teams can be alerted quickly if there is an issue with the unit. The data in the dashboard does have some aggregated trend information, but it mainly reflects the current state as well as any system errors. So, the amount of data needed to populate the dashboard is relatively small. Potentially, the following sources will need access to this data:

    - Customers checking on their system using a mobile device or browser
    - Maintenance teams checking the status of the fleet
    - Data and intelligence algorithms and analytics in the reporting platform to spot trends that can be then sent out as alerts (for example, when an air conditioning fan has been running unusually long, but the building temperature has not gone down)

    Amazon DynamoDB was chosen to store this near real time data set because it is both highly available and scalable. Throughput to this data needs to be easily scaled up or down to meet the needs of its consumers as the platform is adopted and usage grows.

4. The reporting dashboard is a custom web application that is built on top of this data set and run on Amazon EC2. The dashboard provides content based on the system status and trends, and it alerts customers and maintenance crews of any issues that might come up with the unit.

5. The customer accesses the data from a mobile device or a web-browser to get the current status of the system and visualize historical trends.

   The data flow in steps 2 through 5 is built for near real-time reporting of information to human consumers. It is built and designed for low latency and can scale very quickly to meet demand. The data flow in steps 6 through 9 (depicted in the lower part of the diagram) does not have such stringent speed and latency requirements. This allows the architect to design a different solution stack that can hold larger amounts of data at a much smaller cost per byte of information and to choose less expensive compute and storage resources.

6. To read from Amazon Kinesis there will be a separate Kinesis-enabled application that could run on smaller Amazon EC2 instances that scale at a slower rate. While this application is going to analyze the same data set as the upper data flow, the ultimate purpose of this data is to store it for long-term record keeping and to host the data set in a data warehouse. This data set will end up being all data sent from the systems and allow a much broader set of analytics to be performed without the near real-time requirements.

7. The data will be transformed by the Amazon Kinesis-enabled application into a format that is suitable for long term storage and for loading into its data warehouse and storing it on Amazon S3. This data on Amazon S3 not only serves as a parallel ingestion point to Amazon Redshift, but it is durable storage that will hold all data that ever runs through this system and can be the single source of truth. It can be used to load other analytical tools if additional requirements arise. Amazon S3 also comes with native integration with Amazon Glacier if any data needs to be cycled into long-term low-cost cold storage.

8. Amazon Redshift will again be used as the data-warehouse for the larger data set. It can scale easily when the data set grows larger by just adding another node to the cluster.

9. For visualizing the analytics, one of the many partner visualization platforms can be used via the OBDC/JDBC connection to Amazon Redshift. This is where the reports, graphs, and ad-hoc analytics can be performed on the data set to find certain variables and trends that can indicate performance issues with the air conditioning units. The goal is to predict how long a unit will last, and pre-emptively send out maintenance teams based on its prediction algorithms. This level of service would be a differentiator from the competition and lead to increased future sales.

This architecture can start off small and grow as needed. Additionally, by decoupling the two different work streams from each other they can grow at their own rate depending on need. The manufacturer doesn't need to make an upfront commitment. This allows the company to assess the success or failure of the new offering without a large investment.

Conclusion

As more and more data are collected, the analysis of these data requires scalable, flexible, and high-performing tools to provide analysis and insight in a timely fashion. AWS provides many solutions to solve big data analytics problems. Most big data architecture solutions utilize multiple AWS tools to build a complete solution that meets business requirements in the most cost-optimized, performant, and resilient way possible. The result is a flexible big data architecture that scales along with your business on the AWS global infrastructure.

Further Reading

The following AWS resources will help you get started running big data analytics on AWS.

- Visit aws.amazon.com/big-data to view the comprehensive portfolio of big data services[53] as well as links to other resources such AWS big data partners, tutorials, articles, AWS Marketplace offerings on big data solutions,[54] and contact us if you need any help.[55]
- Try one of the Big Data Test Drives:[56]

  Quickly and easily explore the rich ecosystem of products designed to address big data challenges using AWS. Test Drives are developed by the AWS Partner Network (APN) Consulting and Technology partners and are provided free of charge for education, demonstration, and evaluation purposes.
- Take an AWS training course on big data:[57]

  Big data on AWS introduces you to cloud-based big data solutions and Amazon EMR, the AWS big data platform. In this course, you learn how to use Amazon EMR to process data using the broad ecosystem of

[53] http://aws.amazon.com/big-data

[54] https://aws.amazon.com/marketplace

[55] http://aws.amazon.com/big-data/contact-us/

[56] https://aws.amazon.com/testdrive/bigdata/

[57] http://aws.amazon.com/training/course-descriptions/bigdata/

Hadoop tools like Pig and Hive. You also learn how to create big data environments, work with Amazon DynamoDB and Amazon Redshift, understand the benefits of Amazon Kinesis, and leverage best practices to design big data environments for security and cost-effectiveness.

- See the Big Data Customer Case Studies: Learn from the experience of other customers who have built powerful and successful big data platforms on the AWS Cloud.[58]
- Read the AWS Big Data Blog: Real life examples and ideas updated regularly to help you collect, store, clean, process, and visualize big data.[59]

[58] http://aws.amazon.com/solutions/case-studies/big-data/

[59] http://blogs.aws.amazon.com/bigdata/

Notices

© 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.