Big Data Analytics Options on AWS

January 2016

Notices

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.

Contents

Abstract

This whitepaper helps architects, data scientists, and developers understand the big data analytics options available in the AWS cloud by providing an overview of services, with the following information:

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

This paper concludes with scenarios that showcase the analytics options in use, as well as additional resources for getting started with big data analytics on AWS.

Introduction

As we become a more digital society, the amount of data being created and collected is growing and accelerating significantly. Analysis of this ever-growing data becomes a challenge with traditional analytical tools. We require innovation to bridge the gap between data being generated and data that can be analyzed effectively.

Big data tools and technologies offer opportunities and challenges in being able to analyze data efficiently to better understand customer preferences, gain a competitive advantage in the marketplace, and grow your business. Data management architectures have evolved from the traditional data warehousing model to more complex architectures that address more requirements, such as real-time and batch processing; structured and unstructured data; high-velocity transactions; and so on.

Amazon Web Services (AWS) provides a broad platform of managed services to help you build, secure, and seamlessly scale end-to-end big data applications quickly and with ease. Whether your applications require real-time streaming or batch data processing, AWS provides the infrastructure and tools to tackle your

next big data project. No hardware to procure, no infrastructure to maintain and scale—only what you need to collect, store, process, and analyze big data. AWS has an ecosystem of analytical solutions specifically designed to handle this growing amount of data and provide insight into your business.

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 type of analysis. 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 global infrastructure with access to the many different geographic regions[1] that AWS offers, along with the ability to use other scalable services that augment to build sophisticated big data applications. These other services include Amazon Simple Storage Service (Amazon S3)[2] to store data and AWS Data Pipeline[3] to orchestrate jobs to move and transform that data easily. AWS IoT,[4] which lets connected devices interact with cloud applications and other connected devices.

In addition, AWS has many options to help get data into the cloud, including secure devices like AWS Import/Export Snowball[5] to accelerate petabyte-scale data transfers, Amazon Kinesis Firehose[6] to load streaming data, and scalable private connections through AWS Direct Connect.[7] As mobile continues to rapidly grow in usage, you can use the suite of services within the AWS Mobile

Hub[8] to collect and measure app usage and data or export that data to another service for further custom analysis.

These capabilities of the AWS platform make it an ideal fit for solving big data problems, and many customers have implemented successful big data analytics workloads on AWS. For more information about case studies, see Big Data & HPC. Powered by the AWS Cloud.[9]

The following services are described in order from collecting, processing, storing, and analyzing big data:

- Amazon Kinesis Streams
- AWS Lambda
- Amazon Elastic MapReduce
- Amazon Machine Learning
- Amazon DynamoDB
- Amazon Redshift
- Amazon Elasticsearch Service
- Amazon QuickSight

In addition, Amazon EC2 instances are available for self-managed big data applications.

Amazon Kinesis Streams

Amazon Kinesis Streams[10] enables you to build custom applications that process or analyze streaming data in real time. Amazon Kinesis Streams can continuously capture and store terabytes of data per hour from hundreds of thousands of sources, such as website clickstreams, financial transactions, social media feeds, IT logs, and location-tracking events.

With the Amazon Kinesis Client Library (KCL), you can build Amazon Kinesis applications and use streaming data to power real-time dashboards, generate alerts, and implement dynamic pricing and advertising. You can also emit data from Amazon Kinesis Streams to other AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon Elastic MapReduce (Amazon EMR), and AWS Lambda.

Provision the level of input and output required for your data stream, in blocks of 1 megabyte per second (MB/sec), using the AWS Management Console, API,[11] or SDKs.[12] 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 the stream. Within seconds, data put into a 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 (such as Amazon S3 or Amazon Redshift). The KCL 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 fault-tolerant data processing.

Ideal Usage Patterns

Amazon Kinesis Streams is useful wherever there is a need to move data rapidly off producers (data sources) and continuously process it. That processing can be to transform the data before emitting into another data store, drive real-time metrics and analytics, or derive and aggregate multiple streams into more complex streams, or downstream processing. The following are typical scenarios for using Amazon Kinesis Streams for analytics.

- **Real-time data analytics** – Amazon Kinesis Streams 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 Streams, you can have producers push data directly into an Amazon Kinesis stream. For example, you can submit system and application logs to Amazon Kinesis Streams and access the stream for processing within 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 Streams provides accelerated data intake because you are not batching up the data on the servers before you submit it for intake.
- **Real-time metrics and reporting** – You can use data ingested into Amazon Kinesis Streams 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 wait for data batches to arrive.

Cost Model

Amazon Kinesis Streams has simple pay-as–you-go pricing, with no up-front costs or minimum fees, and you'll only pay for the resources you consume. An Amazon Kinesis stream is made up of one or more shards, each shard gives you a capacity 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 capacity of each shard. There are just two pricing components, an hourly charge per shard and a charge for each 1 million PUT transactions. For more information, see Amazon Kinesis Streams Pricing.[13] Applications that run on Amazon EC2 and process Amazon Kinesis streams also incur standard Amazon EC2 costs.

Performance

Amazon Kinesis Streams allows you to choose throughput capacity you require in terms of shards. With each shard in an Amazon Kinesis stream, you can capture up to 1 megabyte per second of data at 1,000 write transactions per second. Your Amazon Kinesis applications can read data from each shard at up to 2 megabytes per second. 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.

Durability and Availability

Amazon Kinesis Streams synchronously replicates data across three Availability Zones in an AWS Region, providing high availability and data durability. Additionally, you can store a cursor in DynamoDB to durably track what has been read from an Amazon Kinesis stream. In the event that your application fails in the middle of reading data from the stream, you can restart your application and use the cursor to pick up from the exact spot where the failed application 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 using API calls or development tools, you can automate scaling of

your Amazon Kinesis Streams environment to meet demand and ensure you only pay for what you need.

Interfaces

There are two interfaces to Amazon Kinesis Streams: input which is used by data producers to put data into Amazon Kinesis Streams; and output to process and analyze data that comes in. Producers can write data using the Amazon Kinesis PUT API, an AWS Software Development Kit (SDK) or toolkit[14] abstraction, the Amazon Kinesis Producer Library (KPL),[15] or the Amazon Kinesis Agent.[16]

For processing data that has already been put into an Amazon Kinesis stream, there are client libraries provided to build and operate real-time streaming data processing applications. The KCL[17] acts as an intermediary between Amazon Kinesis Streams and your business applications which contain the specific processing logic. There is also integration to read from an Amazon Kinesis stream into Apache Storm via the Amazon Kinesis Storm Spout.[18]

Anti-Patterns

Amazon Kinesis Streams has the following anti-patterns:

- **Small scale consistent throughput** – Even though Amazon Kinesis Streams works for streaming data at 200 KB/sec or less, it is designed and optimized for larger data throughputs.
- **Long-term data storage and analytics** – Amazon Kinesis Streams is not suited for long-term data storage. By default, data is retained for 24 hours, and you can extend the retention period by up to 7 days. You can move any data that needs to be stored for longer than 7 days into another durable storage service such as Amazon S3, Amazon Glacier, Amazon Redshift, or DynamoDB.

AWS Lambda

AWS Lambda[19] lets you run code without provisioning or managing servers. You pay only for the compute time you consume—there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service—all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code

with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

Ideal Usage Patterns

Lambda enables you to execute code in response to triggers such as changes in data, shifts in system state, or actions by users. Lambda can be directly triggered by AWS services such as Amazon S3, DynamoDB, Amazon Kinesis Streams, Amazon Simple Notification Service (Amazon SNS), and Amazon CloudWatch, allowing you to build a variety of real-time data processing systems.

- **Real-time file processing** – You can trigger Lambda to invoke a process where a file has been uploaded to Amazon S3 or modified. For example to change an image from color to gray scale after it has been uploaded to Amazon S3.
- **Real-time stream processing** – You can use Amazon Kinesis Streams and Lambda to process streaming data for clickstream analysis, log filtering, and social media analysis.
- **Extract, transform, and load** – You can use Lambda to run jobs that transform data and load it from one data repository to another.
- **Replace cron** – Use schedule expressions to run a Lambda function at regular intervals as a cheaper and more available solution than running cron on an EC2 instance.
- **Process AWS events** – Many other services, such as AWS CloudTrail, can act as event sources simply by logging in to Amazon S3 and using S3 bucket notifications to trigger Lambda functions.

Cost Model

With Lambda, you only pay for what you use. You are charged based on the number of requests for your functions and the time your code executes. The Lambda free tier includes 1M free requests per month and 400,000 GB-seconds of compute time per month. You are charged $0.20 per 1 million requests thereafter ($0.0000002 per request). Additionally the duration of your code executing is priced in relation to memory allocated. You are charged $0.00001667 for every GB-second used. For more information, see AWS Lambda Pricing.

Performance

After deploying your code into Lambda for the first time, your functions are typically ready to call within seconds of upload. Lambda is designed to process events within milliseconds. Latency will be higher immediately after a Lambda function is created, updated, or if it has not been used recently.

Durability and Availability

Lambda is designed to use replication and redundancy to provide high availability for both the service itself and for the Lambda functions it operates. There are no maintenance windows or scheduled downtimes for either. On failure, Lambda functions being invoked synchronously will respond with an exception. Lambda functions being invoked asynchronously are retried at least 3 times, after which the event may be rejected.

Scalability and Elasticity

There is no limit on the amount of Lambda functions that you can run. However, Lambda has a default safety throttle of 100 concurrent executions per account per region. A member of the AWS support team can increase this limit.

Lambda is designed to scale automatically on your behalf. There are no fundamental limits to scaling a function. Lambda allocates capacity dynamically to match the rate of incoming events.

Interfaces

Lambda functions can be managed in a variety of ways. You can easily list, delete, update, and monitor your Lambda functions using the dashboard in the Lambda console. You can also use the AWS CLI and AWS SDK to manage your Lambda functions.

You can trigger a Lambda function from an AWS event, such as Amazon S3 bucket notifications, DynamoDB Streams, CloudWatch logs, Amazon SES, Amazon Kinesis Streams, Amazon SNS, Amazon Cognito, and more. Any API call in any service that supports CloudTrail can be processed as an event in Lambda by responding to CloudTrail audit logs. For more information about event sources, see Core Components: AWS Lambda Function and Event Sources.[20]

Lambda supports programing languages such as Java, Node.js, and Python. Your code can include existing libraries, even native ones. Lambda functions can easily launch processes using languages supported by the Amazon Linux AMI,[21] including Bash, Go, and Ruby. For more information, see the Node.js,[22] Python,[23] and Java[24] documentation.

Anti-Patterns

Lambda has the following anti-patterns:

- **Long-running applications** – Each Lambda function must complete within 300 seconds. For long-running applications that may require jobs to run longer than five minutes, Amazon EC2 is recommended or you create a chain of Lambda functions where function 1 calls function 2, which calls function 3, and so on until the process is completed.
- **Dynamic websites** – While it is possible to run a static website with Lambda, running a highly dynamic and large volume website can be performance prohibitive. Using Amazon EC2 and Amazon CloudFront would be a recommended use case.
- **Stateful applications** –Lambda code must be written in a "stateless" style; that is, it should assume there is no affinity to the underlying compute infrastructure. Local file system access, child processes, and similar artifacts may not extend beyond the lifetime of the request, and any persistent state should be stored in Amazon S3, DynamoDB, or another Internet-available storage service.

Amazon EMR

Amazon EMR[25] is a highly distributed computing framework to easily process and store data quickly in a cost-effective manner. Amazon EMR uses Apache Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances and allows you to use the most common Hadoop tools such as Hive, Pig, Spark, and so on. 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 Patterns

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) data movement
- Risk modeling and threat analytics
- Ad targeting and click stream analytics
- Genomics
- Predictive analytics
- Ad hoc data mining and analytics

For more information, see the Best Practices for Amazon EMR[26] whitepaper.

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, and so on) and all Amazon EC2 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 information, see Amazon EMR Pricing.[27]

Performance

Amazon EMR performance is driven by the type of EC2 instances you choose to run your cluster on and how many you chose to run your analytics. You should choose an instance type suitable for your processing requirements, with sufficient memory, storage, and processing power. For more information about EC2 instance specifications, see Amazon EC2 Instance Types.[28]

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 customers can monitor the health of nodes and replace failed nodes with CloudWatch.

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,[29] which provides a no-NameNode architecture that can tolerate multiple simultaneous failures with automatic failover and fallback. 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.[30] You can add core nodes which hold the Hadoop Distributed File System (HDFS) at any time to increase your processing power and increase the HDFS storage capacity (and throughput). Additionally, you can use Amazon S3 natively or using EMFS along with or instead of local HDFS which allows you to decouple your memory and compute from your storage providing greater flexibility and cost efficiency.

You can also add and remove task nodes at any time which can process Hadoop jobs, but 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 may not know how much data your clusters will be handling in 6 months, or you may 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 console or by a programmatic API[31] call.

Interfaces

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

Hive

Hive is an open source data warehouse and analytics package that runs on top of Hadoop. Hive is operated by Hive QL, a SQL-based language which allows users to structure, summarize, and query data. Hive QL 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 DynamoDB and Amazon S3. For example, with Amazon EMR you can load table partitions automatically from Amazon S3, you can write data to tables in Amazon S3 without using temporary files, and you can access resources in Amazon S3, such as scripts for custom map and/or reduce operations and additional libraries. For more information, see Apache Hive[32] in the *Amazon EMR Release Guide*.

Pig

Pig is an open source analytics package that runs on top of Hadoop. Pig is operated by Pig Latin, a SQL-like language which allows users to structure, summarize, and query data. As well as SQL-like operations, Pig Latin also adds first-class support for map and 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 (such as "REGISTER s3://my-bucket/piggybank.jar"), and additional functionality for String and DateTime processing. For more information, see Apache Pig[33] in the *Amazon EMR Release Guide*.

Spark

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). For more information, see the Installing Apache Spark on an Amazon EMR Cluster blog post.[34]

HBase

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 Hadoop Distributed File System (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. For more information, see HBase and EMR[35] in the *Amazon EMR Developer Guide*.

Impala

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 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. For more information, see Impala and EMR[36] in the *Amazon EMR Developer Guide*.

Hunk

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 on Hadoop. For more information, see Amazon EMR with Hunk: Splunk Analytics for Hadoop and NoSQL.[37]

Presto

Presto is an open-source distributed SQL query engine optimized for low-latency, ad-hoc analysis of data. It supports the ANSI SQL standard, including complex queries, aggregations, joins, and window functions. Presto can process data from multiple data sources including the Hadoop Distributed File System (HDFS) and Amazon S3.

Other third-party tools

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), 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,[38] an extension of the open source tool DistCp that uses MapReduce to efficiently move large amounts of data.

You can optionally use the EMR File System (EMRFS), an implementation of HDFS which allows Amazon EMR clusters to store data on Amazon S3. You can enable Amazon S3 server-side and client-side encryption as well as consistent view for EMRFS. When you use EMRFS, a metadata store is transparently built in DynamoDB to help manage the interactions with Amazon S3 and allows you to

have multiple EMR clusters easily use the same EMRFS metadata and storage on Amazon S3.

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 be worth it for small data sets that can easily be processed in memory on a single system.
- **ACID transaction requirements** – While there are ways to achieve ACID (atomicity, consistency, isolation, durability) or limited ACID on Hadoop, another database, such as Amazon RDS or relational database running on Amazon EC2, may be a better option for workloads with stringent requirements.

Amazon Machine Learning

Amazon ML[39] is a service that makes it easy for anyone to use predictive analytics and machine-learning technology. Amazon ML provides visualization tools and wizards to guide you through the process of creating machine learning (ML) models without having to learn complex ML algorithms and technology. After your models are ready, Amazon ML makes it easy to obtain predictions for your application using API operations, without having to implement custom prediction generation code or manage any infrastructure.

Amazon ML can create ML models based on data stored in Amazon S3, Amazon Redshift, or Amazon RDS. Built-in wizards guide you through the steps of interactively exploring your data, to training the ML model, to evaluating the model quality and adjusting outputs to align with business goals. After a model is ready, you can request predictions in either batches or using the low-latency real-time API.

Ideal Usage Patterns

Amazon ML is ideal for discovering patterns in your data and using these patterns to create ML models that can generate predictions on new, unseen data points. For example, you can:

- **Enable applications to flag suspicious transactions** – Build an ML model that predicts whether a new transaction is legitimate or fraudulent.
- **Forecast product demand** – Input historical order information to predict future order quantities.
- **Personalize application content** – Predict which items a user will be most interested in, and retrieve these predictions from your application in real-time.
- **Predict user activity** – Analyze user behavior to customize your website and provide a better user experience.
- **Listen to social media** – Ingest and analyze social media feeds that potentially impact business decisions.

Cost Model

With Amazon ML, you pay only for what you use. There are no minimum fees and no upfront commitments. Amazon ML charges an hourly rate for the compute time used to build predictive models, and then you pay for the number of predictions generated for your application. For real-time predictions you also pay an hourly reserved capacity charge based on the amount of memory required to run your model.

The charge for data analysis, model training, and evaluation is based on the number of compute hours required to perform them, and depends on the size of the input data, the number of attributes within it, and the number and types of transformations applied. Data analysis and model building fees are priced at $0.42 per hour. Prediction fees are categorized as batch and real-time. Batch predictions are $0.10 per 1,000 predictions, rounded up to the next 1,000, while real-time predictions are $0.0001 per prediction, rounded up to the nearest penny. For real-time predictions, there is also a reserved capacity charge of $0.001 per hour for each 10 MB of memory provisioned for your model.

During model creation, you specify the maximum memory size of each model to manage cost and control predictive performance. You pay the reserved capacity charge only while your model is enabled for real-time predictions. Charges for data stored in Amazon S3, Amazon RDS, or Amazon Redshift are billed separately. For more information, see Amazon Machine Learning Pricing.[40]

Performance

The time it takes to create models, or to request batch predictions from these models depends on the number of input data records, the types and distribution of attributes within these records, and the complexity of the data processing "recipe" that you specify.

Most real-time prediction requests return a response within 100ms, making them fast enough for interactive web, mobile, or desktop applications. The exact time it takes for the real-time API to generate a prediction varies depending on the size of the input data record, and the complexity of the data processing "recipe"[41] associated with the ML model that is generating the predictions. Each ML model that is enabled for real-time predictions can be used to request up to 200 transactions per second by default, and this number can be increased by contacting customer support. You can monitor the number of predictions requested by your ML models by using CloudWatch metrics.

Durability and Availability

Amazon ML is designed for high availability. There are no maintenance windows or scheduled downtimes. The service runs in Amazon's proven, high-availability data centers, with service stack replication configured across three facilities in each AWS Region to provide fault tolerance in the event of a server failure or Availability Zone outage.

Scalability and Elasticity

You can process data sets that are up to 100 GB in size to create ML models or to request batch predictions. For large volumes of batch predictions, you can split your input data records into separate chunks to enable the processing of larger prediction data volume.

By default, you can run up to five simultaneous jobs and by contacting customer service you can have this limit raised. Because Amazon ML is a managed service, there are no servers to provision and as a result you are able to scale as your application grows without having to over provision or pay for resources not being used.

Interfaces

Creating a data source is as simple as adding your data to Amazon S3 or you can pull data directly from Amazon Redshift or MySQL databases managed by Amazon RDS. After your data source is defined, you can interact with Amazon ML using the console. Programmatic access to Amazon ML is enabled by the AWS SDKs and Amazon ML API.[42] You can also create and manage Amazon ML entities using the AWS CLI available on Windows, Mac, and Linux/UNIX systems.

Anti-Patterns

Amazon ML has the following anti-patterns:

- **Very large data sets** – While Amazon ML can support up to 100 GB of data, terabyte-scale ingestion of data is not currently supported. Using Amazon EMR to run Spark's Machine Learning Library (MLlib) is a common tool for such a use case.
- **Unsupported learning tasks** – Amazon ML can be used to create ML models that perform binary classification (choose one of two choices, and provide a measure of confidence), multiclass classification (extend choices to beyond two options), or numeric regression (predict a number directly). Unsupported ML tasks such as sequence prediction or unsupervised clustering can be approached by using Amazon EMR to run Spark and MLlib.

Amazon DynamoDB

Amazon DynamoDB[43] 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. 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.

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. DynamoDB supports three data types (number, string, and binary), in both scalar and multi-valued sets. It supports document stores such as JSON, XML, or

HTML in these data types. Tables do not have a fixed schema, so each data item can have a different number of attributes. The primary key can either be a single-attribute hash key or a composite hash-range key.

DynamoDB offers both global and local secondary indexes provide additional flexibility for querying against attributes other than the primary key. 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, conditional operations, and increment/decrement.

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 Patterns

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
- 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 DynamoDB, you pay only for what you use and there is no minimum fee. DynamoDB has three pricing components: provisioned throughput capacity (per hour), indexed data storage (per GB per month), data transfer in or out (per GB per month). New customers can start using DynamoDB for free as part of the AWS Free Usage Tier.[44] For more information, see Amazon DynamoDB Pricing.[45]

Performance

SSDs and limiting indexing on attributes provides high throughput and low latency[46] 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; you don't need to think about instances, hardware, memory, and other 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

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. DynamoDB Streams[47] captures all data activity that happens on your table and allows the ability to set up regional replication from one geographic region to another to provide even greater availability.

Scalability and Elasticity

DynamoDB is both highly scalable and elastic. There is no limit to the amount of data you can store in a DynamoDB table, and the service automatically allocates more storage as you store more data using the DynamoDB write API operations. 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"[48] or "dial-down"[49] the read and write capacity of a table as your needs change.

Interfaces

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 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).

While standard SQL isn't available, you may use the 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 DynamoDB using the console.

Anti-Patterns

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 may elect to use either Amazon RDS (Amazon Aurora, MySQL, PostgreSQL, Oracle, or SQL Server), or one of the many pre-configured Amazon EC2 database AMIs. You can also install your choice of database software on an EC2 instance that you manage.
- **Joins or complex transactions** – While many solutions are able to leverage 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 may want to explore Amazon Redshift, Amazon RDS, or Amazon EC2 with a self-managed database.
- **Binary large objects (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, DynamoDB still has a role to play in this scenario, for keeping track of metadata (e.g., item name, size, date created, owner, location, etc.) about your binary objects.
- **Large data with low I/O rate** –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 may be a better choice, such as Amazon S3.

Amazon Redshift

Amazon Redshift[50] is a fast, fully-managed, petabyte-scale data warehouse service that makes it simple and cost-effective to analyze all your data efficiently using your existing business intelligence tools. It is optimized for data sets 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. It automates most of the common administrative tasks associated with provisioning, configuring, monitoring, backing up, and securing a data warehouse, making it easy and inexpensive to manage and maintain. This automation allows you to build petabyte-scale data warehouses in minutes instead of weeks or months taken by traditional on-premises implementations.

Ideal Usage Patterns

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
- 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 provides up to 4 TB of backup storage on 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.[51] There is no data transfer charge for communication between Amazon S3 and Amazon Redshift. For more information, see Amazon Redshift Pricing.[52]

Performance

Amazon Redshift uses a variety of innovations to obtain very high performance on data sets 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 CPUs and drives, and a 10 GigE mesh network to maximize throughput between nodes. Performance can be tuned based on your data warehousing needs: AWS offers Dense Compute (DC) with SSD drives as well as Dense Storage (DS) options.

Durability and Availability

Amazon Redshift automatically detects and replaces a failed node in your data warehouse cluster. The data warehouse cluster is read-only until a replacement node is provisioned and added to the DB, 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 data warehouse cluster remains available in the event of a drive failure; because Amazon Redshift mirrors your data across the cluster, it uses the data from another node to rebuild failed drives. Amazon Redshift clusters reside within one Availability Zone,[53] but if you wish to have a multi-AZ set up 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 console or an API call,[54] you can easily change the number, or type, of nodes in your data warehouse as your performance or capacity needs change. Amazon Redshift enables you to 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. For more information, see the About Clusters and Nodes[55] section, Amazon Redshift Clusters topic, in the *Amazon Redshift Management Guide*.

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 pay only 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 automatically redirects queries to your new cluster and removes the old cluster.

Interfaces

Amazon Redshift has custom JDBC and ODBC drivers that you can download from the Connect Client tab of the console, allowing you to use a wide range of familiar SQL clients. You can also use standard PostgreSQL JDBC and ODBC drivers. For more information about Amazon Redshift drivers, see Amazon Redshift and PostgreSQL.[56]

There are numerous examples of validated integrations with many popular BI and ETL vendors.[57] 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 DynamoDB. You can easily load streaming data into Amazon Redshift using Amazon Kinesis Firehose, enabling near real-time analytics with existing business intelligence tools and dashboards you're already using today. Metrics for compute utilization, memory utilization, storage utilization, and read/write traffic to your Amazon Redshift data warehouse cluster are available free of charge via the console or CloudWatch API operations.

Anti-Patterns

Amazon Redshift has the following 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 all the benefits that Amazon Redshift has to offer and Amazon RDS may be a better solution.
- **On-line transaction processing (OLTP)** – Amazon Redshift is designed for data warehouse workloads producing extremely fast and inexpensive analytic capabilities. If you require a fast transactional system, you may want to choose a traditional relational database system built on Amazon RDS or a NoSQL database offering, such as DynamoDB.
- **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 is unstructured, you can perform extract, transform, and load (ETL) on Amazon EMR to get the data ready for loading into Amazon Redshift.
- **BLOB data** – If you plan on storing large binary files (such as digital video, images, or music), you may 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 (such as item name, size, date created, owner, location, and so on) about your binary objects, but the large objects themselves would be stored in Amazon S3.

Amazon Elasticsearch Service

Amazon ES[58] is a managed service that makes it easy to deploy, operate, and scale Elasticsearch in the AWS cloud. Elasticsearch is a real-time distributed search and analytics engine. It allows you to explore your data at a speed and at a scale never before possible. It is used for full-text search, structured search, analytics, and all three in combination.

You can set up and configure your Amazon ES cluster in minutes using the console. Amazon ES manages the work involved in setting up a domain, from provisioning infrastructure capacity you request to installing the Elasticsearch software.

After your domain is running, Amazon ES automates common administrative tasks, such as performing backups, monitoring instances and patching software that powers your Amazon ES instance. It automatically detects and replaces failed Elasticsearch nodes, reducing the overhead associated with self-managed

infrastructure and Elasticsearch software. The service allows you to easily scale your cluster via a single API call or a few clicks in the console.

With Amazon ES, you get direct access to the Elasticsearch open-source API so that code and applications you're already using with your existing Elasticsearch environments work seamlessly. It supports integration with Logstash, an open-source data pipeline that helps you process logs and other event data. It also includes built-in support for Kibana, an open-source analytics and visualization platform that helps you get a better understanding of your data.

Ideal Usage Patterns

Amazon ES is ideal for querying and searching large amounts of data. Organizations can use Amazon ES to do the following:

- Analyze activity logs, such as logs for customer-facing applications or websites
- Analyze CloudWatch logs with Elasticsearch
- Analyze product usage data coming from various services and systems
- Analyze social media sentiments and CRM data, and find trends for brands and products
- Analyze data stream updates from other AWS services, such as Amazon Kinesis Streams and DynamoDB
- Provide customers with a rich search and navigation experience
- Monitor usage for mobile applications

Cost Model

With Amazon ES, you pay only for the compute and storage resources that you use. There are no minimum fees or upfront commitments. You are charged for Amazon ES instance hours, Amazon EBS storage (if you choose this option), and standard data transfer fees.[59]

If you use EBS volumes for storage, Amazon ES allows you to choose the volume type. If you choose Provisioned IOPS (SSD) storage,[60] you are charged for the storage as well as for the throughput that you provision. However, you are not charged for the I/O you consume. You also have the option of paying for additional storage based on the cumulative size of EBS volumes attached to the data nodes in your domain.

Amazon ES provides storage space for automated snapshots free of charge for each Amazon ES domain. Manual snapshots are charged according to Amazon S3 storage rates. For more information, see Amazon Elasticsearch Service Pricing.[61]

Performance

Performance of Amazon ES depends on multiple factors including instance type, workload, index, number of shards used, read replicas, and storage configurations (instance storage or EBS storage, such as general purpose SSD). Indexes are made up of shards of data that can be distributed on different instances in multiple Availability Zones.

Read replicas of the shards are maintained by Amazon ES in a different Availability Zone if zone awareness is checked. Amazon ES can use either the fast SSD instance storage for storing indexes or multiple EBS volumes. A search engine makes heavy use of storage devices and making disks faster results in faster query and search performance.

Durability and Availability

You can configure your Amazon ES domains for high availability by enabling the Zone Awareness option either at domain creation time or by modifying a live domain. When Zone Awareness is enabled, Amazon ES distributes the instances supporting the domain across two different Availability Zones. Then, if you enable replicas in Elasticsearch, the instances are automatically distributed in such a way as to deliver cross-zone replication.

You can build data durability for your Amazon ES domain through automated and manual snapshots. You can use snapshots to recover your domain with preloaded data or to create a new domain with preloaded data. Snapshots are stored in Amazon S3, which is a secure, durable, highly-scalable object storage. By default, Amazon ES automatically creates daily snapshots of each domain. In addition, you can use the Amazon ES snapshot APIs to create additional manual snapshots. The manual snapshots are stored in Amazon S3. Manual snapshots can be used for cross-region disaster recovery and to provide additional durability.

Scalability and Elasticity

You can add or remove instances, and easily modify Amazon EBS volumes to accommodate data growth. You can write a few lines of code that monitor the state of your domain through CloudWatch metrics and call the Amazon ES API to scale your domain up or down based on thresholds that you set. The service executes the scaling without any downtime.

Amazon ES supports one EBS volume (max size of 512 GB) per instance associated with a cluster. With a maximum of 10 instances allowed per Amazon ES cluster, customers can allocate about 5 TB of storage to a single Amazon ES domain.

Interfaces

Amazon ES supports the Elasticsearch API,[62] so code, applications, and popular tools that you are already using with any existing Elasticsearch environments work seamlessly. The AWS SDKs support all of the Amazon ES API operations, making it easy to manage and interact with your domains using your preferred technology. The AWS CLI or the console can be used for creating and managing your domains as well.

Amazon ES supports integration with several AWS services, including streaming data from Amazon S3, Amazon Kinesis Streams, and DynamoDB Streams. The integrations use a Lambda function as an event handler in the cloud that responds to new data by processing it and streaming the data to your Amazon ES domain. Amazon ES also integrates with CloudWatch for monitoring Amazon ES domain metrics and CloudTrail for auditing configuration API calls to Amazon ES domains.

Amazon ES includes built-in integration with Kibana, an open-source analytics and visualization platform and supports integration with Logstash, an open-source data pipeline that helps you process logs and other event data. You can set up your Amazon ES domain as the backend store for all logs coming through your Logstash implementation to easily ingest structured and unstructured data from a variety of sources.

Anti-Patterns

Amazon ES has the following anti-patterns:

- **Online transaction processing (OLTP)** – Amazon ES is a real-time, distributed search and analytics engine. There is no support for transactions or processing on data manipulation. 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 DynamoDB, are a better choice.
- **Petabyte storage** – With a maximum of 10 instances allowed per Amazon ES cluster, you can allocate about 5 TB of storage to a single Amazon ES domain. For workloads larger than this, consider using self-managed Elasticsearch on Amazon EC2.

Amazon QuickSight

In October 2015, AWS introduced the preview of Amazon QuickSight, a fast, cloud-powered, business intelligence (BI) service that makes it easy for you to build visualizations, perform ad hoc analysis, and quickly get business insights from your data.

QuickSight uses a new, super-fast, parallel, in-memory, calculation engine (SPICE) to perform advanced calculations and render visualizations rapidly. QuickSight integrates automatically with AWS data services, enables organizations to scale to hundreds of thousands of users, and delivers fast and responsive query performance via SPICE's query engine. At one-tenth the cost of traditional solutions, QuickSight allows you to deliver affordable BI functionality to everyone in your organization. To learn more and sign up for the preview, see QuickSight.[63]

Amazon EC2

Amazon EC2,[64] with instances acting as AWS virtual machines, provides an ideal platform for operating your own self-managed big data analytics applications on AWS infrastructure. Almost any software you can install on Linux or Windows virtualized environments can be run on Amazon EC2 and you can use the pay-as-go pricing model. What you don't get are the application-level managed services

that come with the other services mentioned in this whitepaper. There are many options for self-managed big data analytics; here are some examples:

- A NoSQL offering, such as MongoDB
- A data warehouse or columnar store like Vertica
- A Hadoop cluster
- An Apache Storm cluster
- An Apache Kafka environment

Ideal Usage Patterns

- **Specialized Environment** – When running a custom application, a variation of a standard Hadoop set or an application not covered by one of our other offerings, Amazon EC2 provides the flexibility and scalability to meet your computing needs.
- **Compliance Requirements** – Certain compliance requirements may need you to run applications yourself on Amazon EC2 instead of 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.), and different pricing options (On-Demand, Reserved, and Spot). Depending on your application requirements, you may want to use additional services along with Amazon EC2, such as Amazon Elastic Block Store (Amazon EBS) for directly attached persistent storage or Amazon S3 as a durable object store; each comes with their own pricing model. If you do run your big data application on Amazon EC2, you are responsible for any license fees just as you would be in your own data center. The AWS Marketplace[65] offers many different third-party, big data software packages preconfigured to launch with a simple click of a button.

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, 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 does not affect application users. For non-uptime critical applications, you can back up your application to Amazon S3 and restore to any Availability Zone in the region if an instance or zone failure occurs. Other options exist, depending on which application you are running and the requirements, such as mirroring your application.

Scalability and Elasticity

Auto Scaling[66] is a service that allows you to automatically scale your Amazon EC2 capacity up or down according to conditions that you define. With Auto Scaling, you can ensure that the number of EC2 instances you're using scales up seamlessly during demand spikes to maintain performance, and scales down automatically during 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 CloudWatch and available at no additional charge beyond CloudWatch fees.

Interfaces

Amazon EC2 can be interfaced programmatically via API, SDK, or the console. Metrics for compute utilization, memory utilization, storage utilization, network consumption, and read/write traffic to your instances are free of charge via the console or CloudWatch API operations.

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

Anti-Patterns

Amazon EC2 has the following 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, then this "do it yourself" model of managing your own analytics software on Amazon EC2 may not be the correct choice.
- **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 installation for the system in question, you should consider using the AWS equivalent such as Amazon EMR, DynamoDB, Amazon Kinesis Streams, or Amazon Redshift.

Solving Big Data Problems on AWS

In this whitepaper, we have examined some tools at your disposal on AWS for big data analytics. This provides a good reference point when starting to design your big data applications. However, there are additional aspects you should consider when selecting the right tools for your specific use case. In general, each analytical workload will have certain characteristics and requirements, which dictate which tool to use, such as:

- 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?
- 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 helps point you in the right direction for the tool to use. In some cases, you can simply map your big data analytics workload easily into one of the services 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 solve both of the above examples in the same toolset, you end up either over-provisioning and therefore

overpaying for unnecessary response time, 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 is the most cost-effective use of your compute and storage resources.

Big data doesn't need to mean "big costs". So, when designing your applications, it's important to make sure your design is cost efficient. If it's not, relative to the alternatives, then it's probably not the right design. Another common misconception is that having multiple tool sets to solve a big data problem is more expensive or harder to manage than having one big tool. If you 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 specification and not overprovision. With the AWS pay-as-you-go and only for what you use infrastructure as a service model, this equates to a much better 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 may find this approach easier to manage rather 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 (real-time requests) into a round hole (a large data warehouse).

The AWS platform makes it easy to decouple your architecture by having 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. Let's put this into practice by exploring a couple of real world, big data analytics problem scenarios and walk through an AWS architectural solution.

Example 1: Enterprise Data Warehouse

A multi-national 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 all act independently from a technical standpoint. They have different management, point-of-sale systems, and accounting departments. 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 be enabled to do ad hoc analytics when required. Example analytics that the company wants are:

- What trends exist across channels?
- What geographic regions do better across channels?
- How effective are their advertisements and coupons?
- What trends exist across each clothing line?
- What external forces may have an impact its sales, for example unemployment rate, or the weather?
- How does store attribute effect sales, for example tenure of employees/management, strip mall versus enclosed mall, location of merchandise in the store, promotion, endcaps, sales circulars, in-store displays, etc.?

An enterprise data warehouse is a terrific way to solve this problem. The data warehouse needs to collect data from each of the three channels' various systems and from public records for weather and economic data. Each data source sends data on a daily basis for consumption by the data warehouse. Because each data source may be structured differently, an extract, transform, and load (ETL) process is performed to reformat the data into a common structure. Then, analytics can be performed across data from all sources simultaneously. To do this, we use the following data flow architecture:

Enterprise Data Warehouse

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 is used to transform and cleanse the data from the source format into the destination and a 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 warehouses get new data on a nightly basis from its many different sources. Because there is no need for these analytics in the middle of the night, the only requirement around this transformation process is that it finishes by the morning when the CEO and other business users need results. This requirement means that you can use the Amazon EC2 Spot market[67] 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 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 be the exact right power to complete all data transformation jobs for the lowest possible price without contending with the resources of the other jobs.
3. Each transformation job then puts the formatted and cleaned data onto Amazon S3. Amazon S3 is used here again because Amazon Redshift can 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 loads, sorts, distributes, and compresses the data into 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 another node as the data size increases over time and the business expands.
5. For visualizing the analytics, Amazon QuickSight can be used, or one of the many partner visualization platforms via the OBDC/JDBC connection to Amazon Redshift. This is where the reports and graphs can be viewed by the CEO and her staff. This data can now be used by the executives for making better decisions about company resources, which can 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 scaled in a few clicks by increasing the number of nodes in the Amazon Redshift cluster.

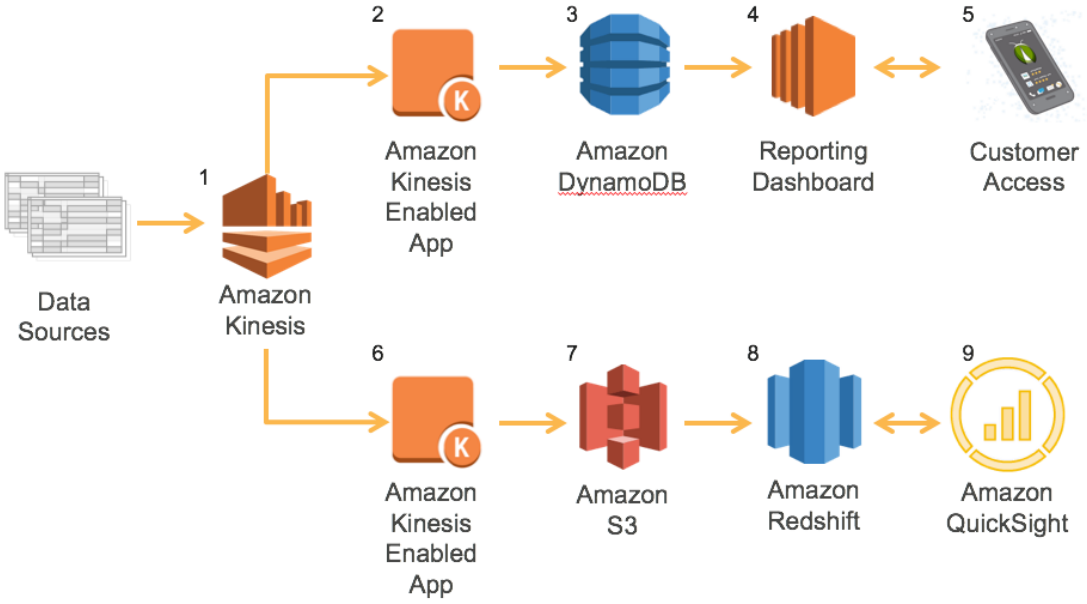
Example 2: Capturing and Analyzing Sensor Data

An international air conditioner manufacturer has many large air conditioners that it sells to various commercial and industrial companies. Not only do they sell the air conditioner units but, to better position themselves against their competitors, they also offer add-on services where you can see real-time dashboards in a mobile application or a web browser. Each unit sends its sensor information for processing and analysis. This data is used by the manufacturer and its customers. With this capability, the manufacturer can visualize the dataset and spot trends.

Currently, they have a few thousand pre-purchased units with this capability. They expect to deliver these to customers in the next couple of months and are hoping that, in time, thousands of units throughout the world will be using this platform. If successful, they would like to expand this offering to their consumer line as well, with a much larger volume and a greater market share. The solution needs to be able to handle massive amounts of data and scale as they grow their business without interruption. How should you design such a system? First, break it up into two work streams, both originating from the same data:

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

The following is the data-flow architecture to solve this big data problem:



Capturing and Analyzing Sensor Data

1. The process begins with each air conditioner unit providing a constant data stream to Amazon Kinesis Streams. This provides an elastic and durable interface the units can talk to that can be scaled seamlessly as more and more A/C units are sold and brought online.
2. Using the Amazon Kinesis Streams-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 Streams, analyze it, and determine if the data warrants an update to the real-time dashboard. It looks for changes in system operation, temperature fluctuations, and any errors that the units encounter.
3. This data flow needs to occur in near real time so that customers and maintenance teams can be alerted as quickly as possible if there is an issue with the unit. The data in the dashboard does have some aggregated trend information, but it is mainly the current state as well as any system errors. So, the data needed to populate the dashboard is relatively small. Additionally, there will be lots of potential access to this data from the following sources:
    - Customers checking on their system via a mobile device or browser
    - Maintenance teams checking the status of its fleet

— Data and intelligence algorithms and analytics in the reporting platform spot trends that can be then sent out as alerts, such as if the A/C fan has been running unusually long with the building temperature not going down.

DynamoDB was chosen to store this near real-time data set because it is both highly available and scalable; throughput to this data can 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. It provides content based on the system status and trends as well as alerting customers and maintenance crews of any issues that may 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 (steps 2-5) that was just described 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 (steps 6-9) that is 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 choose less expensive compute and storage resources.

6. To read from the Amazon Kinesis stream, there is a separate Amazon Kinesis-enabled application that probably runs on a smaller EC2 instance that scales 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 and to host the data set in a data warehouse. This data set ends up being all data sent from the systems and allows a much broader set of analytics to be performed without the near real-time requirements.
7. The data is transformed by the Amazon Kinesis–enabled application into a format that is suitable for long-term storage, for loading into its data warehouse, and storing on Amazon S3. The data on Amazon S3 not only serves as a parallel ingestion point to Amazon Redshift, but is durable storage that will hold all data that ever runs through this system; it 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 is again used as the data warehouse for the larger data set. It can scale easily when the data set grows larger, by 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 lead to A/C units underperforming or breaking.

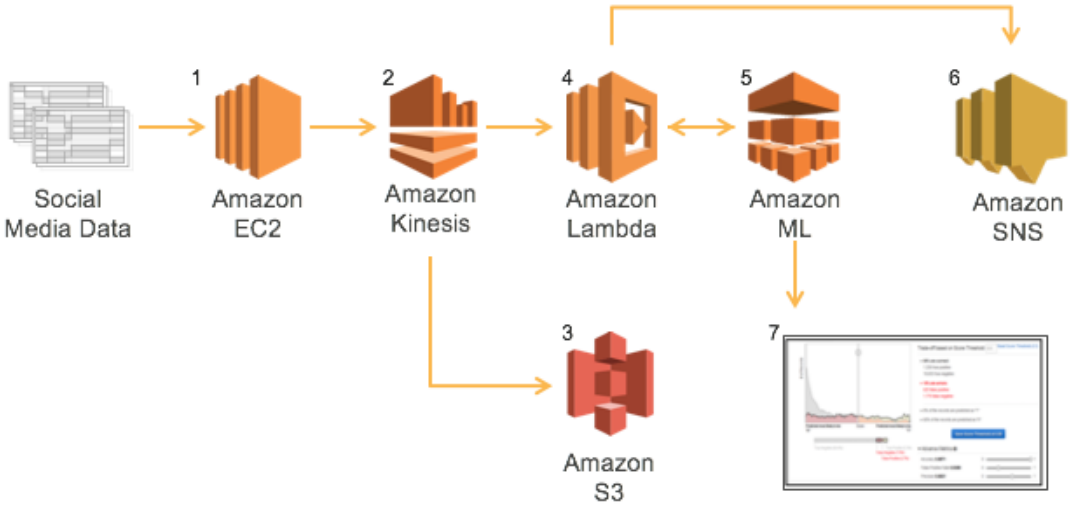
This architecture can start off small and grow as needed. Additionally, by decoupling the two different work streams from each other, they can grow at their own rate depending on need without upfront commitment, allowing the manufacturer to assess the success or failure of this new offering without a large investment. You could easily image further additions such as Amazon ML to be able to predict exactly how long an A/C unit will last and pre-emptively send out maintenance teams based on its prediction algorithms to give their customers the best possible service and experience. This level of service would be a differentiator to the competition and lead to increased future sales.

Example 3: Sentiment Analysis of Social Media

A large toy maker has been growing very quickly and expanding their product line. After each new toy release, the company wants to understand how consumers are enjoying and using their products. Additionally, the company wants to ensure that their consumers are having a good experience with their products. As the toy ecosystem grows, the company wants to ensure that their products are still relevant to their customers and that they can plan future roadmaps items based on customer feedback. The company wants to capture the following from social media:

- Understand how consumers are using their products
- Ensure customer satisfaction
- Plan for future roadmaps

Capturing the data from various social networks is relatively easy but the challenge is building the intelligence programmatically. After the data is ingested, the company wants to be able to analyze and classify the data in a cost-effective and programmatic way. To do this, the following architecture can be used:



Sentiment Analysis of Social Media

1. The first thing to do is decide on which social media sites to listen to. Then create an application that poll those sites via their corresponding APIs and run it on Amazon EC2.
2. Next, an Amazon Kinesis stream is created, because we might have multiple data sources: Twitter, Tumblr, and so on. This way, a new stream can be created each time for each new data source added and you can use the existing application code and architecture. Additionally, in this example, a new Amazon Kinesis stream is created to copy the raw data to Amazon S3.
3. For archival, long term analysis, and historical reference, raw data is stored into Amazon S3. Additional Amazon ML batch models can be run from data located in Amazon S3 to do predictive analysis and track consumer buying trends.
4. As noted in the architecture diagram, Lambda is used for processing and normalizing the data and requesting predictions from Amazon ML. After the

Amazon ML prediction is returned, the Lambda function can take action based on the prediction – for example, to route a social media post to the customer service team for further review.

5. Amazon ML is used to make predictions on the input data. For example, an ML model can be built to analyze a social media comment to determine whether the customer expressed negative sentiment about a product. To get accurate predictions with Amazon ML, start with training data and ensure that your ML models are working properly. If you are creating ML models for the first time, see Tutorial: Using Amazon ML to Predict Responses to a Marketing Offer.[68] As mentioned earlier, if multiple social network data sources are used, then a different ML model for each one is suggested to ensure prediction accuracy.
6. Finally, actionable data is sent to Amazon SNS using Lambda, and delivered to the proper resources via text or email for further investigation.
7. As part of the sentiment analysis, creating an Amazon ML model that is updated regularly is imperative for accurate results. Additional metrics about a specific model can be graphically displayed via the console, such as: accuracy, false positive rate, precision, and recall. For more information, see Step 4: Review the ML Model Predictive Performance and Set a Cut-Off.[69]

By using a combination of Amazon Kinesis Streams, Lambda, Amazon ML, and Amazon SES, we have create a scalable and easily customizable social listening platform. It is important to note that this picture does not depict the action of creating a ML model. This act would be done at least one time, but usually done on a regular basis to keep the model up to date. The frequency of creating a new model depends on the workload and is really only done to make the model more accurate when things change.

Conclusion

As more and more data is generated and collected, data analysis requires scalable, flexible, and high performing tools to provide insights in a timely fashion. However, organizations are facing a growing big data ecosystem where new tools emerge and "die" very quickly. Therefore, it can be very difficult to keep pace and choose the right tools.

This whitepaper offers a first step to help you solve this challenge. With a broad set of managed services to collect, process, and analyze big data the AWS platform makes it easier to build, deploy and scale big data applications, allowing you to focus on business problems instead of updating and managing these tools.

AWS provides many solutions to address big data analytic requirements. Most big data architecture solutions use multiple AWS tools to build a complete solution: this can help meet the stringent business requirements in the most cost-optimized, performant, and resilient way possible. The result is a flexible, big data architecture that is able to scale along with your business on the AWS global infrastructure.

Contributors

The following individuals and organizations contributed to this document:

- Erik Swensson, Manager, Solutions Architecture, Amazon Web Services
- Erick Dame, solutions architect, Amazon Web Services
- Shree Kenghe, solutions architect, Amazon Web Services

Further Reading

The following resources can help you get started in running big data analytics on AWS:

- Visit aws.amazon.com/big-data[70]

  View the comprehensive portfolio of big data services as well as links to other resources such AWS big data partners, tutorials, articles, and AWS Marketplace offerings on big data solutions. Contact us if you need any help.

- Read the AWS Big Data Blog[71]

  The blog features real life examples and ideas updated regularly to help you collect, store, clean, process, and visualize big data.

- Try one of the Big Data Test Drives[72]

  Explore the rich ecosystem of products designed to address big data challenges using AWS. Test Drives are developed by 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[73]

  The Big Data on AWS course introduces you to cloud-based big data solutions and Amazon EMR. We show you how to use Amazon EMR to process data using the broad ecosystem of Hadoop tools like Pig and Hive. We also teach you how to create big data environments, work with DynamoDB and Amazon Redshift, understand the benefits of Amazon Kinesis Streams, and leverage best practices to design big data environments for security and cost-effectiveness.

- View the Big Data Customer Case Studies[74]

  Learn from the experience of other customers who have built powerful and successful big data platforms on the AWS cloud.

Document Revisions

January 2016	Revised to add information on Amazon Machine Learning, AWS Lambda, Amazon Elasticsearch Service; general update
December 2014	First publication

Notes

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

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

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

[4] https://aws.amazon.com/iot/

[5] https://aws.amazon.com/importexport/

[6] http://aws.amazon.com/kinesis/firehose

[7] https://aws.amazon.com/directconnect/

[8] https://aws.amazon.com/mobile/

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

[10] https://aws.amazon.com/kinesis/streams

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

[12] http://docs.aws.amazon.com/aws-sdk-php/v2/guide/service-kinesis.html

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

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

[15] http://docs.aws.amazon.com/kinesis/latest/dev/developing-producers-with-kpl.html

[16] http://docs.aws.amazon.com/kinesis/latest/dev/writing-with-agents.html

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

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

[19] https://aws.amazon.com/lambda/

[20] http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html

[21] https://aws.amazon.com/amazon-linux-ami/

[22] http://docs.aws.amazon.com/lambda/latest/dg/nodejs-create-deployment-pkg.html

[23] http://docs.aws.amazon.com/lambda/latest/dg/lambda-python-how-to-create-deployment-package.html

[24] http://docs.aws.amazon.com/lambda/latest/dg/lambda-java-how-to-create-deployment-package.html

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[39] https://aws.amazon.com/machine-learning/

[40] https://aws.amazon.com/machine-learning/pricing/

[41] http://docs.aws.amazon.com/machine-learning/latest/dg/suggested-recipes.html

[42] http://docs.aws.amazon.com/machine-learning/latest/APIReference/Welcome.html

[43] https://aws.amazon.com/dynamodb

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

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

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

[47] http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html

[48] DynamoDB allows you to change your provisioned throughput level by up to 100% with a single UpdateTable API operation call. To increase your throughput by more than 100%, call UpdateTable again.

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

[50] https://aws.amazon.com/redshift/

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

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

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

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

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

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

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

[58] https://aws.amazon.com/elasticsearch-service/

[59] https://aws.amazon.com/ec2/pricing/

[60] https://aws.amazon.com/ebs/details/

[61] https://aws.amazon.com/elasticsearch-service/pricing/

[62] https://aws.amazon.com/elasticsearch-service/faqs/

[63] https://aws.amazon.com/quicksight

[64] https://aws.amazon.com/ec2/

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

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

[67] http://aws.amazon.com/ec2/spot/

[68] http://docs.aws.amazon.com/machine-learning/latest/dg/tutorial.html

[69] http://docs.aws.amazon.com/machine-learning/latest/dg/step-4-review-the-ml-model-predictive-performance-and-set-a-cut-off.html

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

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

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

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

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