Database Architectures Compared

(/users/3106511/jryan999.html) by John Ryan (/users/3106511/jryan999.html) MVB </>

CORE ·
Sep. 14, 18 · Database Zone (/database-sql-nosql-tutorials-tools-news) · Opinion

The dominance of Oracle, IBM and Microsoft in the anlytics database market was challenged by database appliances in the 1990s from Netezza and Teradata using an MPP architecture, and severely put to the test with the emergence of “Big Data”, and Hadoop with distributed processing.

In this article I’ll summarise how the architecture has changed over time. Starting with a single machine, SMP platform, then the Massively Parallel Processing (MPP) architecture, followed by Hadoop/HDFS, and new cloud based solutions from Amazon, Google and Snowflake.

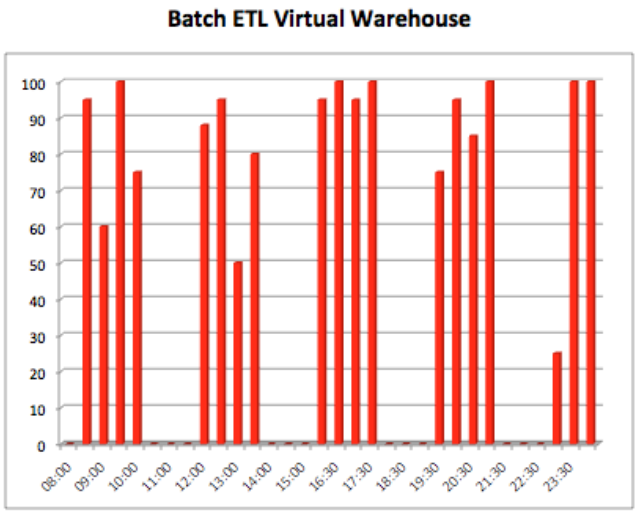
No need to worry if you don't have a PhD in Database Design - this is a "Plain English" article.

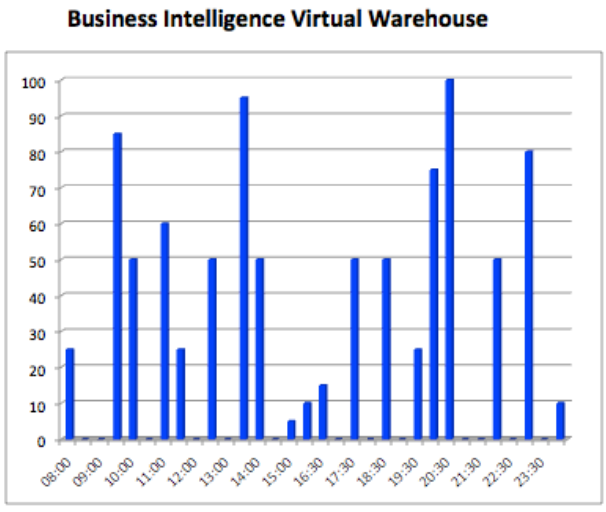
What Problem(s) Are We Trying to Solve?

At its very core, a database simply needs to store data so it can be retrieved later. However,

there are a number of non-functional requirements to consider for a large analytic platform:

- **Performance and Response Time:** Often the most obvious needs: the database must be fast enough. That's a deliberately vague definition because if you are building an online financial trading system, a fifty millisecond response time is way too slow, whereas most users can only dream of response times of 1/20th of a second. The appropriate solution is normally determined by a number of factors including the expected data volume, number of concurrent users, and the type of workload expected. For example, a system to deliver batch reports to 50 concurrent users will have different performance profile to an Amazon-style eCommerce database supporting 10,000 concurrent users.
- **Throughput:** Often confused with performance, this indicates the overall amount of work which can be done in a set time. For example, a system which needs to transform and query massive data volumes needs to process very large data volumes at speed. This will lead to a different solution where a fast response time for individual queries is the priority. Generally, the best way to maximize throughput is to break down large tasks and execute the individual components in parallel.
- **Workload Management:** This describes the ability of the system to handle a mixed workload on the same platform. This is related to performance and response times in that most systems need to find a balance between the two. The diagram below illustrates a typical situation whereby a system must handle both high volume batch operations where throughput is more important, alongside fast response times for online users.

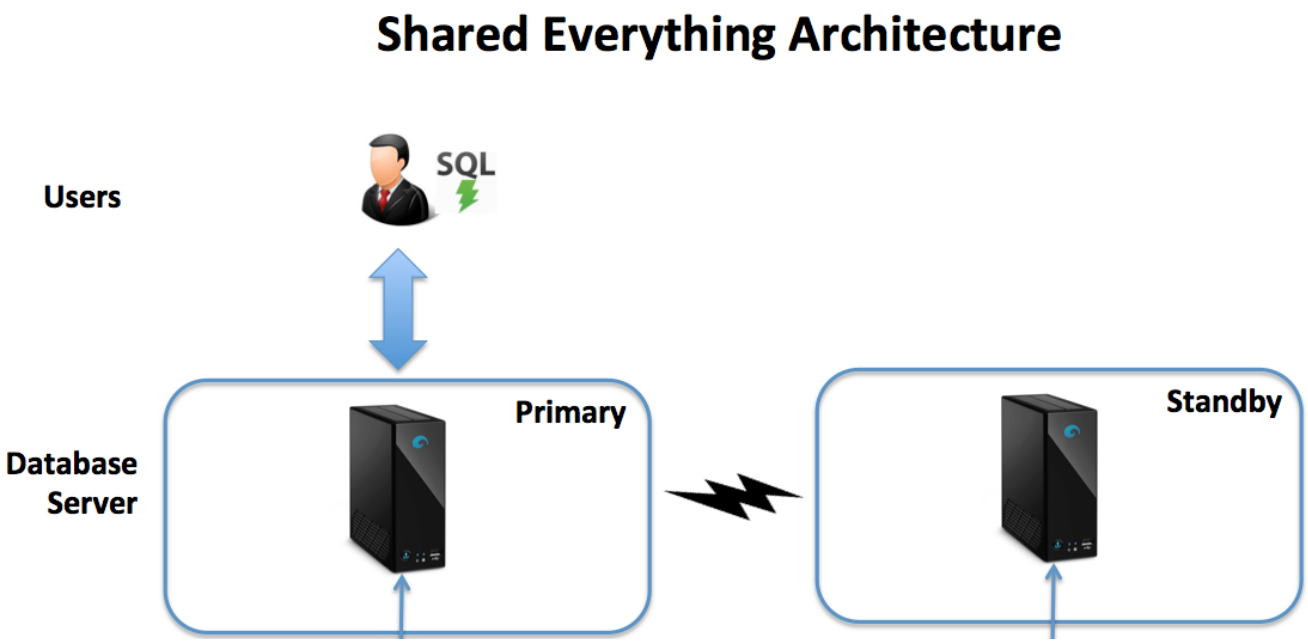




- **Scalability:** The system must have options to grow. Nothing stays still, and if your application is successful, both the data volumes and the number of users is likely to grow. When it does, the database will need to cope with the additional workload. Even within this simple requirement, there are many additional details to decide. Is data volume growth relatively stable or highly unpredictable? Can you accept downtime to add additional compute resources or storage or do you need a 24x7 operation?
- **Concurrency:** Describes the extent to which the system can support multiple users at

the same time. This is related to scalability and performance in that the system must be fast enough to respond quickly but also handle the required number of concurrent users without a significant drop in response times. Likewise, if the number of users exceeds the predefined limits, we need to consider what options are available to scale the system. This highlights an important difference in scalability. That the challenges around handling increasing data volumes are significantly different to the challenges around maintaining response times as the number of users increases. As we'll see later — one size does not fit all.

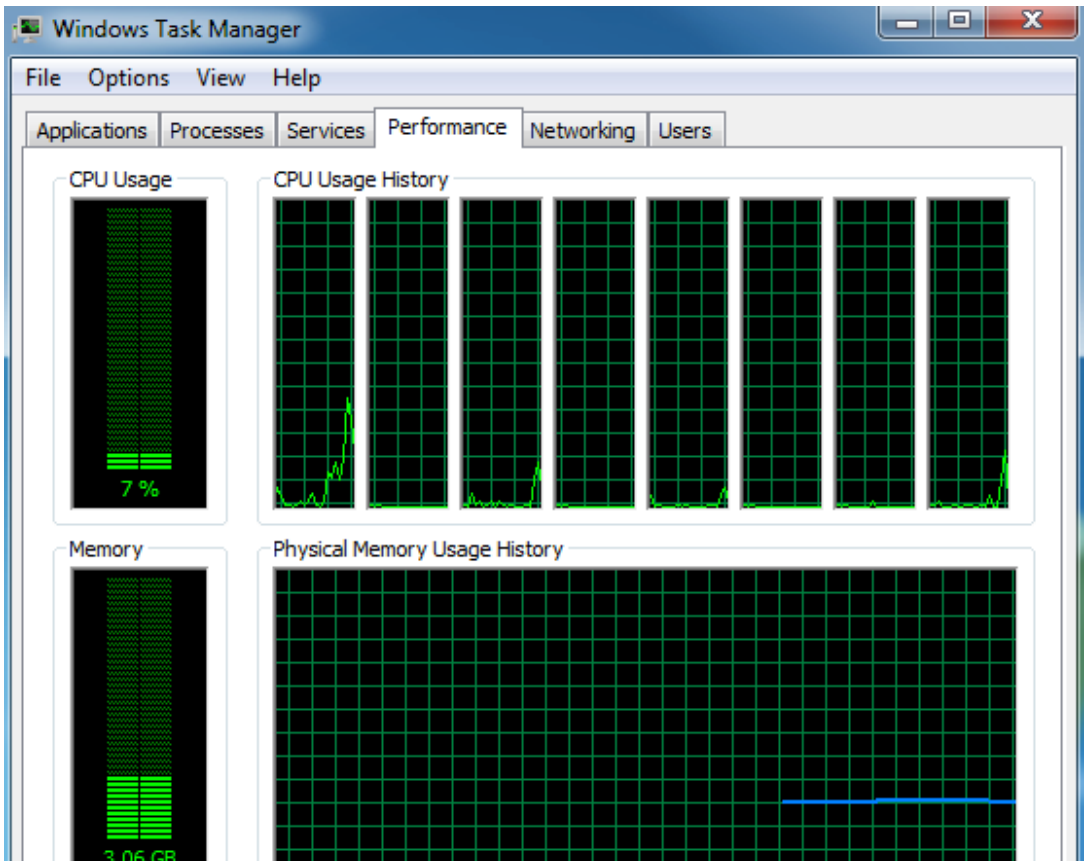
- **Data Consistency:** While the previous requirements may be obvious, the need for consistency is a more nuanced requirement, and one might assume that, of course, you need data consistency. In reality, however, the need for data consistency is flexible, and (for example), an online banking system might need 100% guaranteed consistency, whereas consistency and pin sharp accuracy might be less important on a reporting system which delivers high-level management reports. Data consistency is an important consideration, because it may rule out the use of a NoSQL database solution where consistency is seldom guaranteed.
- **Resilience and Availability:** Describes the ability of the database to keep going despite component, machine or even entire data center failure, and the level of ***Resilience*** is determined by the need for ***Availability***. For example, an online banking system may need to be available 99.999% of the time, which allows for a little over five minutes of downtime per year. This implies a highly resilient (and potentially expensive) solution, whereas a system that allows downtime at weekends has more options available. The diagram below illustrates a common solution, whereby transactions written to the primary system are automatically replicated to an off-site standby system with automatic fail-over built in.



- **Accessible via SQL:** While not an absolute requirement, the SQL language has been around for over forty years and is used by millions of developers, analysts, and business users alike. Despite this, some NoSQL databases (for example HBase and MongoDB) don't natively support access using SQL. While there are several bolt-on tools available, these databases are fundamentally different from the more common relational databases and don't (for example) support relational joins, transactions, or immediate data consistency.

Option 1: The Relational Database on SMP Hardware

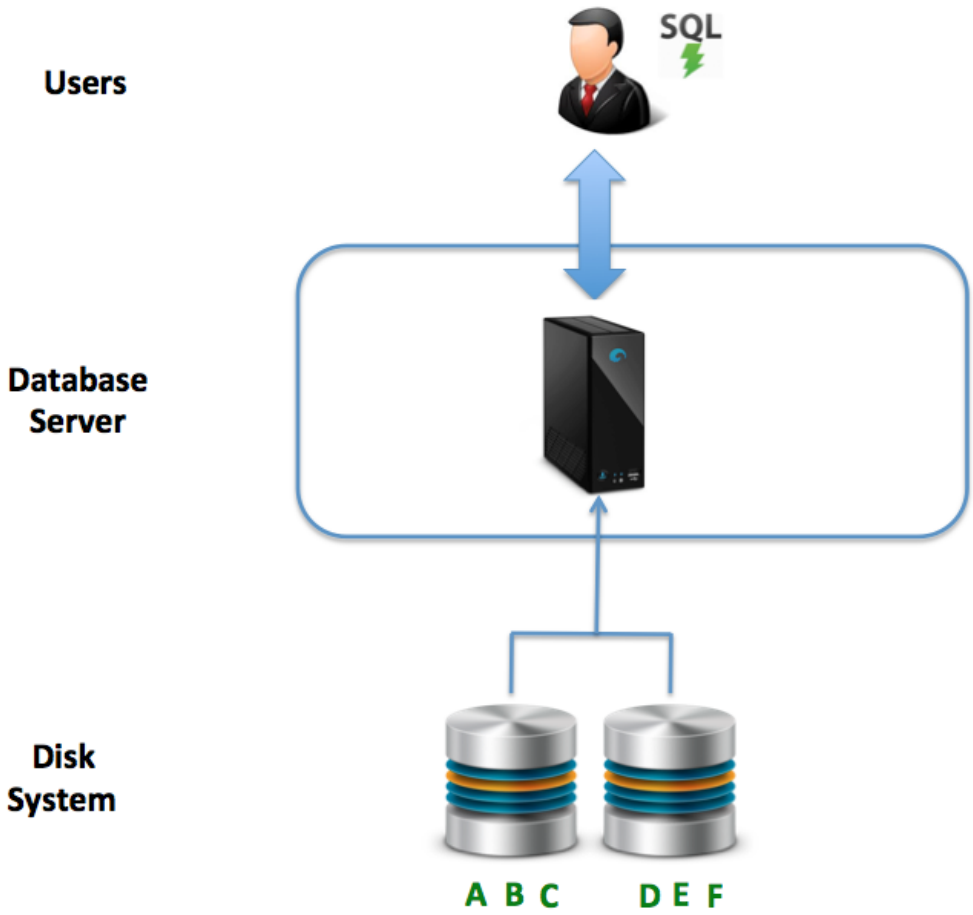
Since the early 1980s, the market has been dominated by Oracle, Microsoft, and IBM who have delivered general purpose solutions designed to deal with the above requirements. The underlying hardware and database system architecture was first developed in the 1970s and is based upon a Symmetric Multiprocessing (SMP) hardware in which a number of physical processors (or cores) execute instructions using shared memory and disks.



The screenshot above shows the Windows Task Manager, which shows eight processors executing instructions on an SMP database server. An SMP based database solution has both advantages and drawbacks as follows:

Advantages

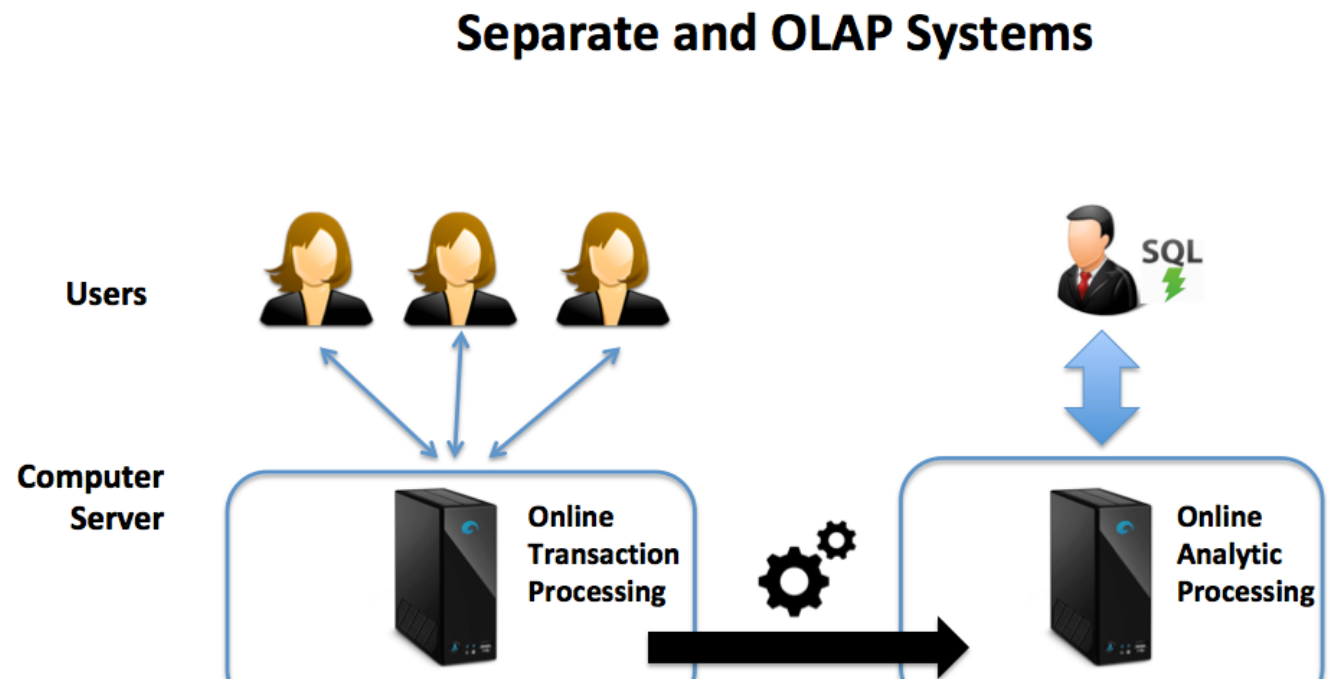
- **It Works:** It is a battle-hardened, proven architecture that is relatively inexpensive to deploy and can run on everything from massive servers to mid-sized commodity hardware. It has a proven track record of delivering reasonable performance and throughput.
- **It is Homogeneous:** This means databases designed for this platform will run on almost any hardware. As it is based upon a number of cores, those can be physical or logical meaning it is also an option to run on a virtual server on a cloud platform.
- **Data Consistency:** The diagram below illustrates the essential nature of this architecture — a single machine connected to either local or network connected disks. Effectively, there is one copy of the data, and therefore, data consistency is not the challenge; it is on distributed systems. This compares well to many NoSQL solutions where the risk of data inconsistency is traded for maximum response times.



Drawbacks

Although this is a popular and widely deployed architecture, it does have the following drawbacks:

- **Difficult to scale:** While it's possible to add CPUs, memory or additional disks, in reality, scalability is limited, and it's likely the system will need to be replaced by a larger machine within a few years.
- **Maximum Sized:** Because of the difficulty in scaling, most architects size SMP platforms to the maximum predicated workload. This means paying for more processing capacity than is initially needed while performance gradually degrades as more load is added over time. A better solution would allow the incremental addition of compute resources over time.
- **Inefficient Scaling:** Adding additional or faster CPUs seldom increases performance on a linear scale. For example, it's highly unlikely adding 100% faster processors will double performance unless the system is entirely CPU bound. In most cases, the bottleneck shifts to another component, and increasing capacity linearly is not possible.
- **Workload Management:** Depending on the database solution and hardware, an SMP based system can deliver reasonable performance or throughput — seldom both at the same time. In common with MPP platforms, many solutions struggle to balance competing demands for fast response times and high batch throughput. The diagram below illustrates a common approach whereby customer facing Online Transaction Processing (OLTP) systems deliver fast response times, while periodically delivering data to a separate Data Warehouse platform for Online Analytic Processing (OLAP). While Oracle claims to resolve the challenges around combining OLTP and OLAP using Oracle12c in Memory, in reality, this is limited small to medium-sized applications.

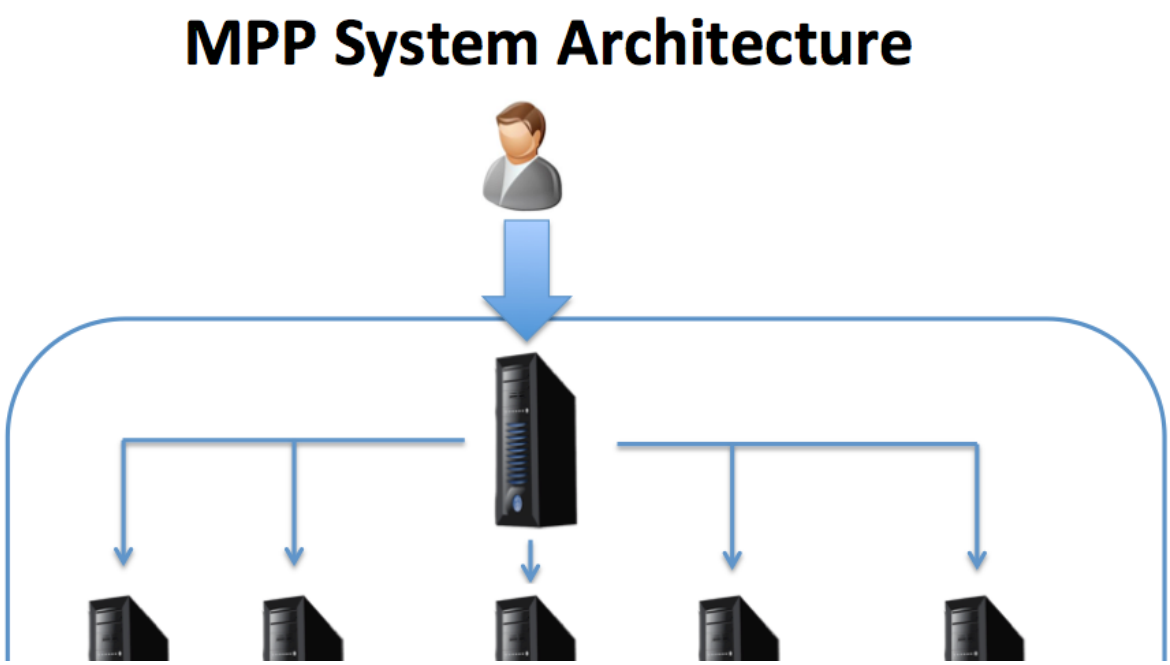


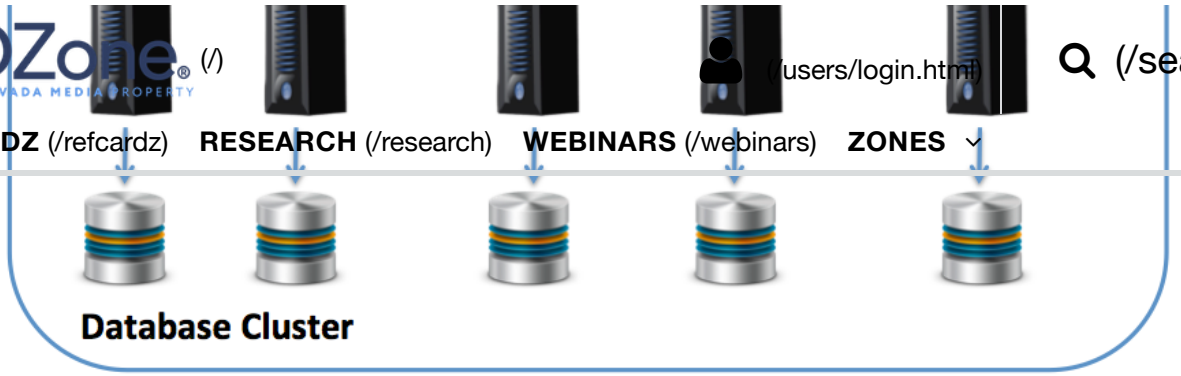
- **Resilience and Availability:** Because SMP systems use a single platform, the solution needs to be extended to provide resilience in the event of component or entire data center failure. This often makes this option expensive, although (in theory), it can be deployed on inexpensive commodity servers, in reality, it's more often deployed on enterprise quality hardware with dual redundant disks, network connections, and power supplies. While this provides resilience to component failure, the solution will also need a separate standby system to guarantee high availability.

Option 2: The Relational Database on MPP hardware

In 1984, Teradata delivered its first production database using a Massively Parallel Processing (https://dzone.com/refcardz/getting-started-with-prestodb) (MPP) architecture, and two years later, Forbes magazine named Teradata "Product of the Year" as it produced the first terabyte-sized database in production. This architecture was later adopted by Netezza, the Microsoft Parallel Data Warehouse (PDW) and HP Vertica among others. Today, Apple, Walmart, and eBay routinely store and process (https://gigaom.com/2013/03/27/why-apple-ebay-and-walmart-have-some-of-the-biggest-data-warehouses-youve-ever-seen/) petabytes of data on MPP platforms.

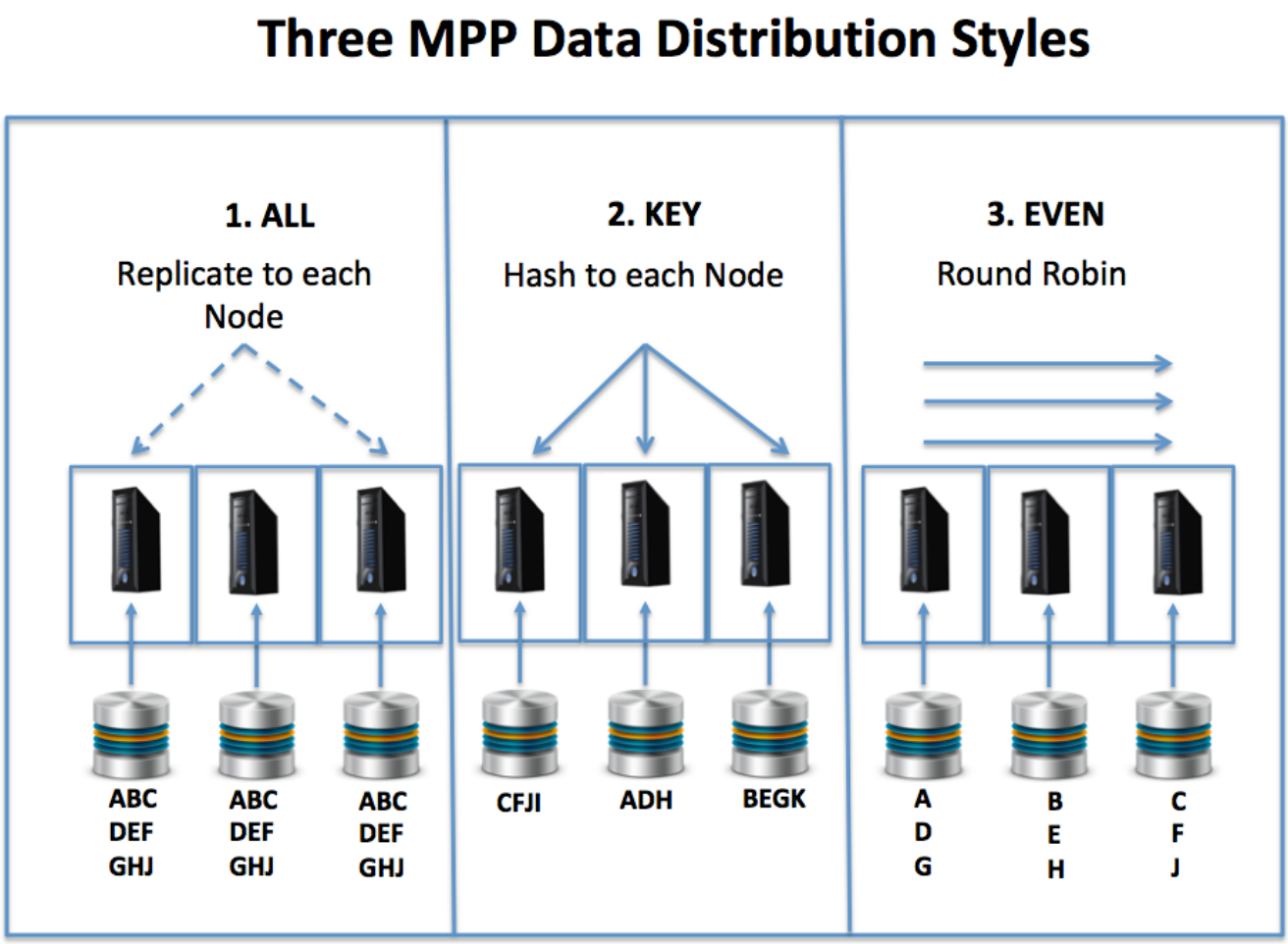
The diagram below provides an illustration of an MPP architecture whereby a coordinating SMP server accepts user SQL statements, which are distributed across a number of independently running database servers that act together as a single clustered machine. Each node is a separate computer with its own CPUs, memory, and directly attached disk.





Using this solution, as data is loaded, a *consistent hashing algorithm* may be used to distribute the data evenly, which (if all goes well) will lead to a balanced distribution of work across the cluster. MPP architectures are an excellent solution for Data Warehouse and analytic platforms because queries can be broken into component parts, and executed in parallel across the servers with dramatic performance gains.

However, unlike SMP systems, where data placement is automatic at the disk level, it's critical to get this step right to maximize throughput and response times. The diagram below illustrates the three MPP data distribution methods available.



The options include:

- **Replication:** Typically used for relatively small tables, using this method, the data is

duplicated on each node in the cluster. While this may seem wasteful, my own experience on several multi-terabyte warehouses indicates there are often a huge number of smaller reference and dimension tables which are often joined with a significantly larger transaction or fact tables. This reference data is a great fit for the replication method, as it means it can be joined locally and in parallel on each node in the cluster, avoiding *data shuffling* between nodes.

- **Consistent Hashing:** Is typically used for larger transaction or fact tables, and involves generating a reproducible key to allocate each row to an appropriate server in the cluster. This method ensures an even load on the cluster, although an incorrect selection of a *clustering key* can lead to hot-spots, which may significantly limit performance in some cases.
- **Round Robin:** This method involves writing each row on the next node in sequence in a round-robin fashion and is typically only used for temporary staging tables, which will be written and read once only. It has the advantage that data is guaranteed to be evenly distributed, and therefore query load likewise, but it is a poor solution unless all related reference data tables are replicated to every node.

Advantages

The MPP architecture has several clear advantages over the SMP solution, and these include:

- **Performance:** This is an area that MPP systems can really excel. Provided data and processing can be evenly executed in parallel across nodes in the cluster, the performance gains over even the largest SMP servers can be significant.

> ***"With a massively parallel processing (MPP) design, queries commonly complete 50 times faster than traditional data warehouses built on symmetric multi-processing (SMP) systems". –*** **Microsoft Corporation.**

- **Scalability and Concurrency:** Unlike SMP solutions, MPP based systems have the option to incrementally add compute and storage resources, and throughput is largely improved at an arithmetic rate. Adding an additional same sized node increases the ability of the system to handle additional queries without a significant drop in performance.
- **Cost and High Availability:** Some MPP based data warehouse solutions are designed to run on inexpensive commodity hardware without the need for enterprise-level dual redundant components which can contain costs. These solutions typically use automatic data replication to improve system resilience and guarantee high availability. This can be a hugely more efficient use of resources, and it avoids the cost of a largely unused

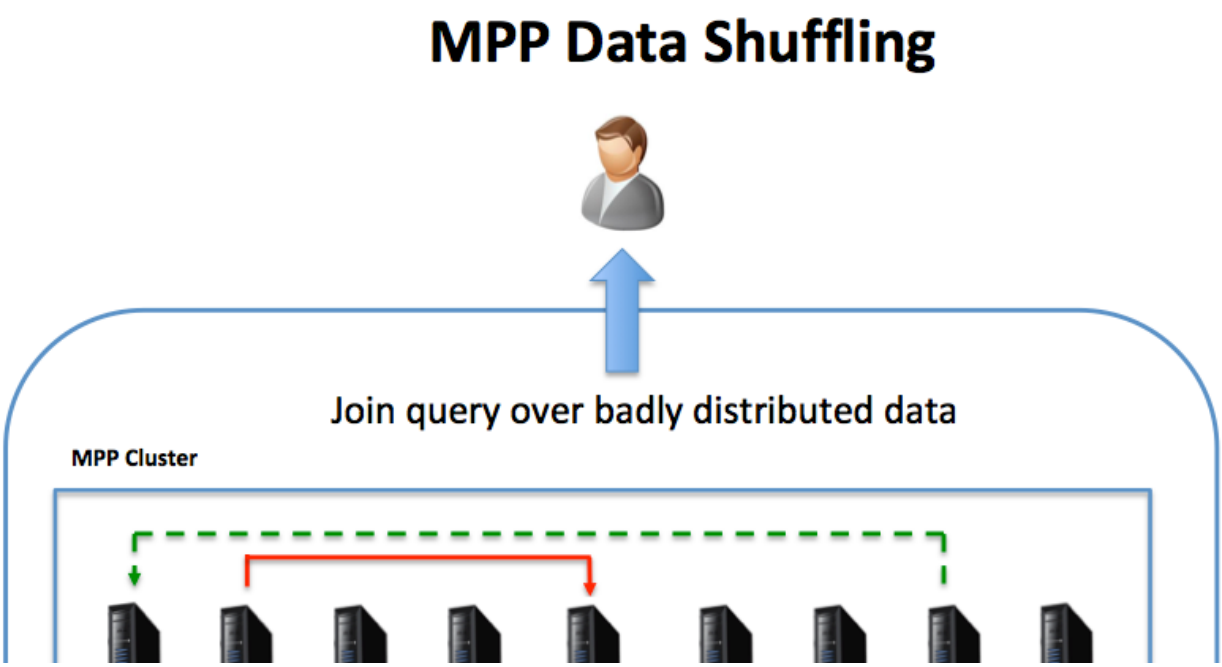
be a hugely more efficient use of resources, and it avoids the cost of a largely unused hot standby used by many SMP based solutions.

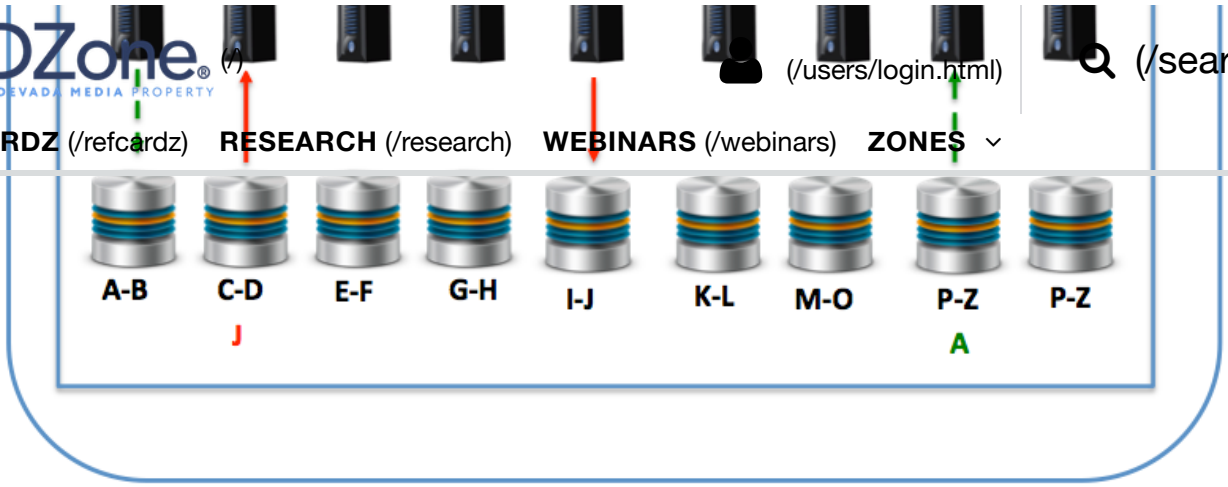
- **Read and Write Throughput:** As data is distributed across the system, this solution can achieve a remarkable level of throughput as both read and write operations can be executed in parallel across independent nodes in the cluster.

Drawbacks

Although MPP systems have compelling advantages over the traditional SMP architecture, they do have the following drawbacks:

- **Complexity and cost:** While the architecture appears simple on the surface, a well designed MPP solution hides a significant level of complexity, and the earliest commercial MPP database systems from Teradata and Netezza were delivered as a hardware appliance at significant cost. However, as this architecture has become more popular for large-scale analytics platforms, prices are beginning to ease.
- **Data Distribution is Critical:** Unlike the SMP solution in which data placement at disk level is simple and can be automated, an MPP platform requires careful design of the data distribution to avoid data skew leading to processing hot-spots. If for example, a poor distribution key is selected, this can lead to a small number of nodes being over-loaded while others lie idle which will limit overall throughput and query response time. Likewise, if a reference table is not correctly placed with the associated transaction data, this can lead to excessive *Data Shuffling* whereby data is transferred between nodes to complete join operations which again can lead to performance issues. This is illustrated in the diagram below where reference data is shuffled between two nodes. While it's possible to fix the problem, it typically needs a massive data reorganization effort, and potentially system downtime.





- **Need for downtime:** Although some MPP solutions have resilience and high availability built in, many require downtime or reduced performance to support the addition of new nodes. In some cases, the entire cluster must be taken off-line to add additional nodes, and even where this is not needed, the adding nodes typically involve the re-distribution of data across the cluster to make use of the additional compute resources. This may not be ideal or even a viable option for some customers.
- **Lack of Elasticity:** Although MPP systems can be scaled out, this typically involves the commissioning and deployment of new hardware which can take days or even weeks to complete. These systems typically don't support elasticity – the ability to extend or reduce the compute resources on demand to meet real-time processing requirements.
- **Scale Out Only:** To avoid an imbalanced system, it's typically only sensible to add nodes of the exact same specification, compute power and disk storage. This means, although adding additional nodes increases the concurrency (the ability for additional users to query the data), it's not possible to significantly increase batch throughput. In short, while it's possible to *scale out*, there are few options to *scale up* the solution to a much more powerful system.
- **Potential Over Capacity:** In theory, MPP systems are perfectly balanced, in that additional nodes add both storage and compute resources to the cluster. However, because these are so closely tied, if storage demands exceed the need for compute capacity, the total cost of ownership can be huge as the *cost per terabyte* rises disproportionately. This is especially prevalent in the popularity of *Data Lake* solutions which potentially store massive volumes of infrequently accessed data. In summary, using a pure MPP solution, you could be paying for 300 times more compute processing than you actually need.

> ***"On analytic MPP platforms data volumes [can] grow disproportionately to analytic workloads (our research showed a variance in some cases of as much as 300:1)" -***

Option 3: Hadoop With SQL Over HDFS

The diagram below illustrates the inexorable growth in interest in *Big Data* between 2010-15. This clearly concerned the database vendors including IBM who became a Hadoop Platform Vendor (http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&appname=gpateam&supplier=897&letternum=ENUS211-196) and Oracle who developed a Big Data Appliance (http://www.oracle.com/technetwork/database/bigdata-appliance/overview/bigdataappliance-datasheet-1883358.pdf).

During this time, there was much discussion on whether the Data Warehouse was dead (http://www.jamesserra.com/archive/2017/12/is-the-traditional-data-warehouse-dead/) and whether Hadoop would replace (https://0x0fff.com/hadoop-vs-mpp/) MPP platforms, although the general consensus appears to indicate that Hadoop is at best a complementary technology to a data warehouse; not a replacement for it.

What Is Hadoop?

Unlike MySQL and PostgreSQL, which are open source databases, Hadoop is not a single product but an open source ecosystem of related projects. As of September 2018, this included over 150 projects (https://hadoopecosystemtable.github.io/) with 12 separate tools for SQL over Hadoop and 17 databases. To illustrate the scale of the ecosystem, Amazon UK sells over 1,000 different books on Hadoop technology, many which cover just a single tool, including Hadoop The Definitive Guide (https://www.amazon.co.uk/Hadoop-Definitive-Guide-Tom-White/dp/1491901632/ref=sr_1_1?ie=UTF8&qid=1535823122&sr=8-1&keywords=hadoop) at over 750 pages.

The diagram below illustrates some of the key components and Hadoop distributors, and each component needs a significant investment in time and expertise to fully exploit.

Primary Use Cases

Hadoop is a massive subject and difficult to summarise in a short article, but the primary problem spaces it addresses include:

1. **Large Volume Data Storage and Batch Processing:** Hadoop and the primary data storage system (Hadoop Distributed File System — HDFS) are often promoted as an inexpensive data storage solution and a suitable platform for a *Data Lake*. Given its ability to scale to thousands of nodes, it is perhaps a good fit for large-scale batch data processing using the SQL based tool *Apache Hive* over HDFS.
2. **Real-Time Processing:** While HDFS is best suited to massive batch processes running for hours, other components including *Kafka, Spark Streaming, Storm* and *Flink* are specifically designed to provide a micro-batch or real-time streaming solution. This is expected to become increasingly important as the Internet of Things (IoT) industry increasingly delivers real-time results from millions of embedded sensors which need real or near real-time data analysis and response.
3. **Text Mining and Analytics:** Another area where the Hadoop platform is strong is its ability to deal with unstructured data including text. While traditional databases work well with structured data in rows and columns, Hadoop includes tools to analyze the meaning (https://www.linkedin.com/pulse/text-mining-1-minute-rajesh-k-gupta-mba-

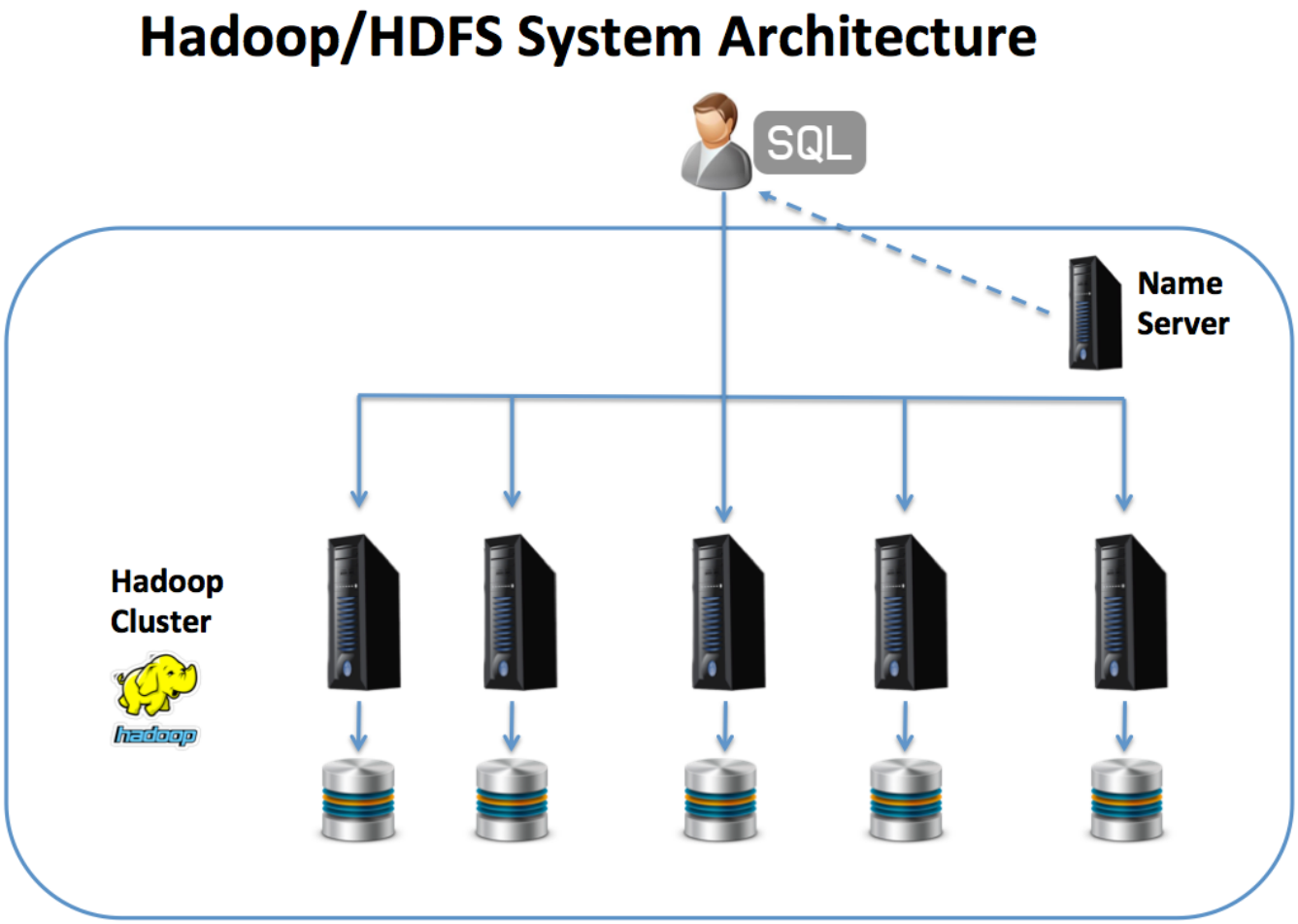
pmp-scpm/) of unstructured text fields, for example, online product reviews or a Twitter feed which can be mined for sentiment analysis.

The above use-cases are often described as the Three V's: Data Volume, Velocity, and Variety.

Hadoop/HDFS Architecture

As the focus of this article on Database Architecture, I will focus on the batch processing use case. A potential real-time processing architecture is described separately in my article on Big Data: Velocity in Plain English (https://www.analytics.today/blog/big-data-velocity).

On first impression, the Hadoop/HDFS architecture appears to be similar to the MPP architecture, and the diagram below illustrates the similarity.



The diagram above shows how data is normally processed using SQL. The *Name Server* acts as a directory lookup service to point the client to the node(s) upon data will be stored or queried from, otherwise, it looks remarkably similar to an MPP architecture.

The biggest single difference, however, is that whereas an MPP platform distributes individual *rows* across the cluster, Hadoop simply breaks the data into arbitrary *blocks*, of which Cloudera recommend are sized at 128Mb, which are then replicated to at least two other nodes for resilience in the event of node failure.

This is significant because it means small files (anything less than 128Mb) are entirely held on a single node, and even a Gigabyte sized file will be distributed over only 8 nodes (plus replicas). This is important because Hadoop is designed to deal with very large data sets and large clusters. However, since small tables are distributed over fewer servers, it is not ideal for data files of under 50-100Gb in size.

Processing *small* data sets is a challenge on Hadoop because in the worse case, processing data on a single node runs entirely sequentially with nothing run in parallel. Indeed, as many Hadoop clusters tend to use a large number of relatively slow and inexpensive commodity servers, performance on small data can be very poor indeed. Furthermore, as the number of small files rises, it increasingly becomes a problem for the Name Server to manage.

To put this in context, my experience demonstrates that on most mid-range data warehouse platforms (around 10Tb of data), only around 10% of tables hold more than 100Gb of data, and 70% of tables held less than 1Gb. These would be a particularly poor fit for deployment on Hadoop, even though the two largest tables exceed 1Tb in size.

> ***"Most of those who expected Hadoop to replace their enterprise data warehouse have been greatly disappointed" - James Serra - Microsoft (http://www.jamesserra.com/archive/2017/12/is-the-traditional-data-warehouse-dead/)***

Advantages

The Hadoop/HDFS architecture has the following advantages as a data storage and processing platform:

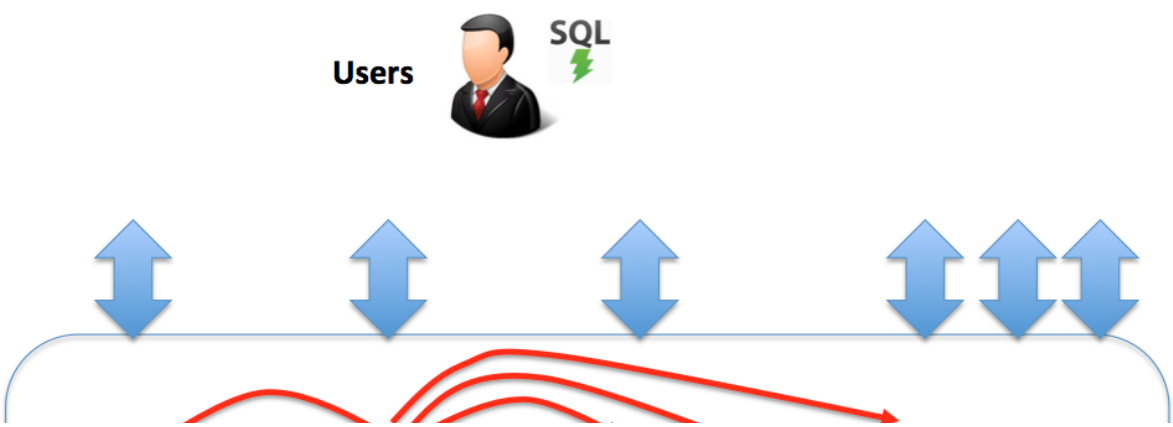
- **Batch Performance:** Hadoop can be a great option to achieve high throughput when processing very large data sets, although processing uses a brute force approach needing jobs executed across multiple nodes in parallel.
- **Scalability:** Similar to MPP systems, additional nodes can be added to extend the Hadoop cluster, and clusters can reach up to. 5,000 nodes (https://www.enterprisetech.com/2013/11/08/cluster-sizes-reveal-hadoop-maturity-curve/) in some cases.
- **Availability and Resilience:** As data is automatically replicated (duplicated) to multiple servers, resilience and high availability are both transparent and built in. This means, (for example), it's possible in production to take a node offline for maintenance without any interruption in service.
- **Cost (hardware and licenses):** As Hadoop tends to be deployed on inexpensive commodity servers running open source software, the cost of hardware and licenses can

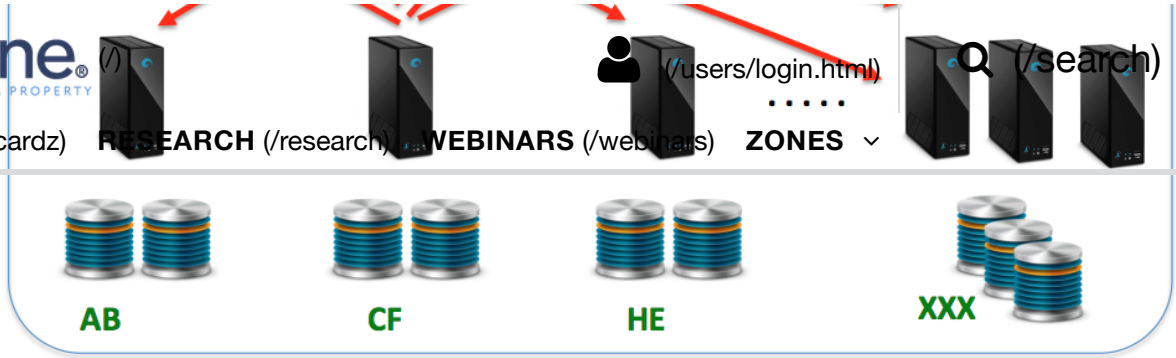
be considerably lower than the traditional enterprise-class data warehouse appliances, and associated license costs.

Drawbacks

Because of the compelling cost and batch performance benefits, Hadoop is often promoted as a replacement (https://www.datanami.com/2018/08/28/cloudera-pivots-from-zoo-animals-to-data-warehousing/) for the Data Warehouse. I would, however, advise caution for the following reasons:

- **Management Complexity:** As described above, Hadoop is not a single product but a massive ecosystem of software, and deployment often needs skilled knowledge of a range of tools including *HDFS, Yarn, Spark, Impala, Hive, Flume, Zookeeper,* and *Kafka*. For organizations whose entire business is managing data (eg. Facebook or LinkedIn), Hadoop may be a sensible option. For many customers, however, it's best avoided as an analytics platform in favor of a database solution. Even a large-scale MPP solution can be considerably less complex to deploy and maintain than Hadoop.
- **Immature query tools:** Relational database management systems include decades of experience in automated query tuning to efficiently execute complex SQL queries. Most Hadoop based SQL tools don't, however, achieve the level of sophistication required, and often rely upon brute force to execute queries. This leads to the highly inefficient use of machine resources, which on a cloud-based (pay as you go) basis can quickly become expensive.
- **The Small Files Issue:** While throughput of very large data processing can be efficient when fully executed in parallel, processing of relatively small files can lead to very poor query response times.
- **Data Shuffling:** Unlike the MPP solution where data can be co-located by either a consistent hashing key or data replication, there is no option to place data on Hadoop nodes. This means that join operations across multiple tables which are (by design) randomly distributed across the cluster, can lead to a massive data shuffling exercise and potentially severe performance issues. This is illustrated in the diagram below.





- **Poor low latency query performance:** Although data caching solutions may help, Hadoop/HDFS is a very poor solution for low latency queries, for example, to serve data to a dashboard. Again, this is related to the architecture being primarily designed to serve very large data sets using brute force batch processing.
- **Other Disadvantages:** Similar to the disadvantages described under the MPP platform, this solution suffers from a lack of elasticity, inability to scale up, and the potential for extreme overcapacity of compute resources, especially when used as a Data Lake.

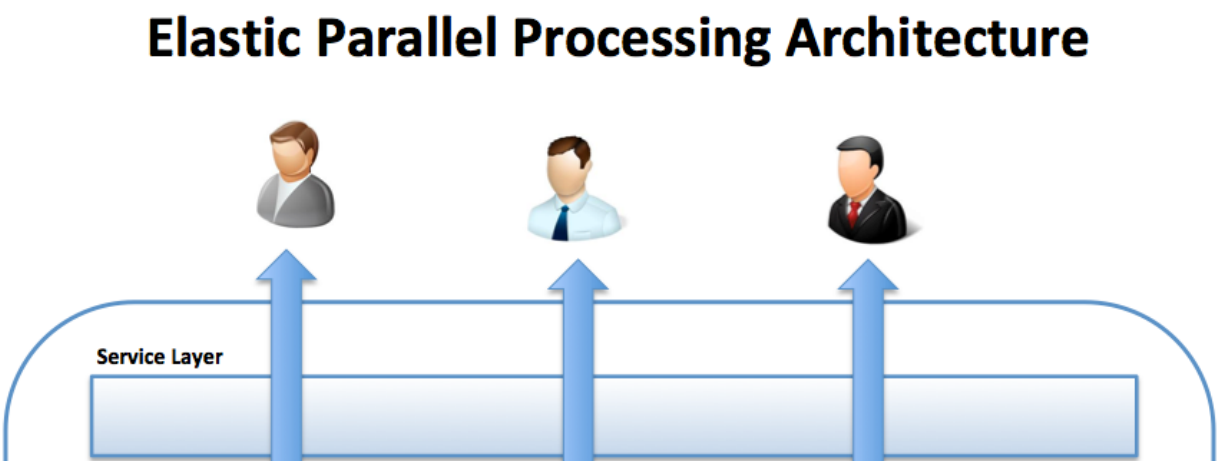
For an excellent lecture on why Hadoop is an inappropriate solution for the marketplace, watch this lecture by Turing Medal Award winner, Dr Michael Stonebraker — Big Data is (at least) Four Different Problems (https://codingvideos.net/michael-stonebraker-big-data-is-at-least-four-different-problems/).

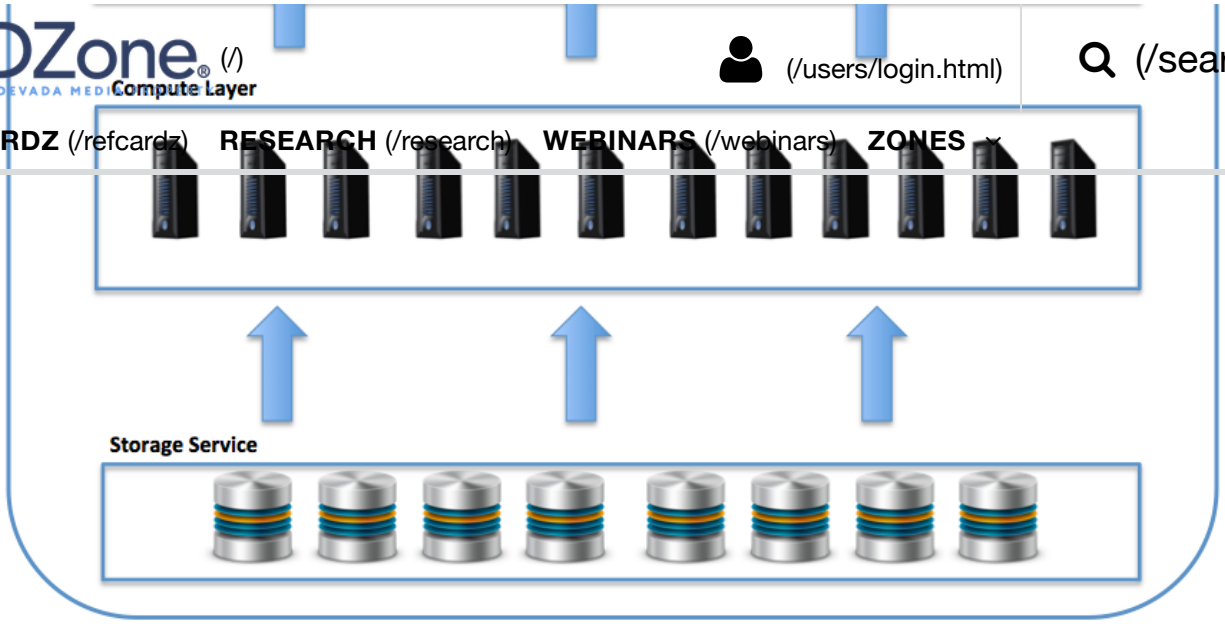
Option 4: EPP: Elastic Parallel Processing

Similar to the MPP solution in which a number of independently running *shared nothing* nodes store and process queries in parallel, the EPP (Elastic Parallel Processing) architecture provides an impressive level of scalability.

However, unlike the MPP cluster in which data storage is directly attached to each node, the EPP architecture separates compute and storage, which means each can be scaled or elastically reduced independently.

This is illustrated in the diagram below:





In the above diagram, the long-term storage is provided by a *storage service*, which is visible from every node in the cluster. Queries are submitted to the *service layer*, which is responsible for overall query coordination, query tuning, and transaction management, and actual work is executed on the *compute layer* — effectively an MPP cluster.

While the *compute layer* typically has directly attached disk or fast SSD for local storage, the use of an independent *storage service* layer means that data storage can be scaled independently of compute capacity. This means it's possible to elastically resize the compute cluster, providing all of the benefits of an MPP architecture while largely eliminating many of the drawbacks.

As of 2018, there are several analytic platforms that (to a varying degree) can be described as supporting *Elastic Parallel Processing*, and these include solutions from Snowflake Computing, Microsoft, HP, Amazon, and Google.

> **"New cloud architectures and infrastructures challenge conventional thinking about big data and collocated storage and compute". - Tripp Smith – CTO Clarity Insights.**

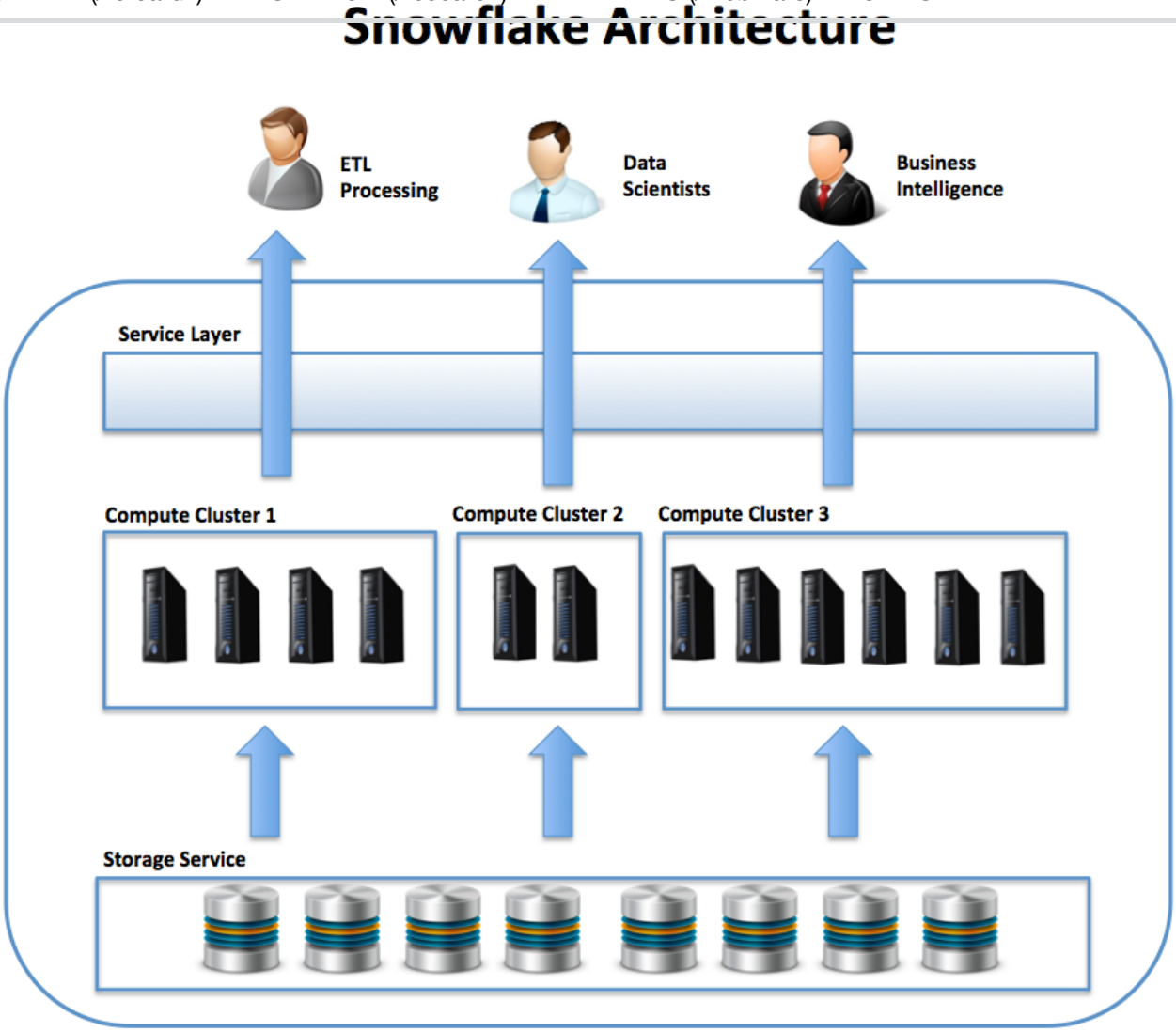
Snowflake: The Elastic Data Warehouse

Image title

The Snowflake Elastic Data Warehouse (https://www.snowflake.com/uk/product/) is by far the best current example of a truly elastic EPP analytics platform, and this section will describe the benefits of this solution.

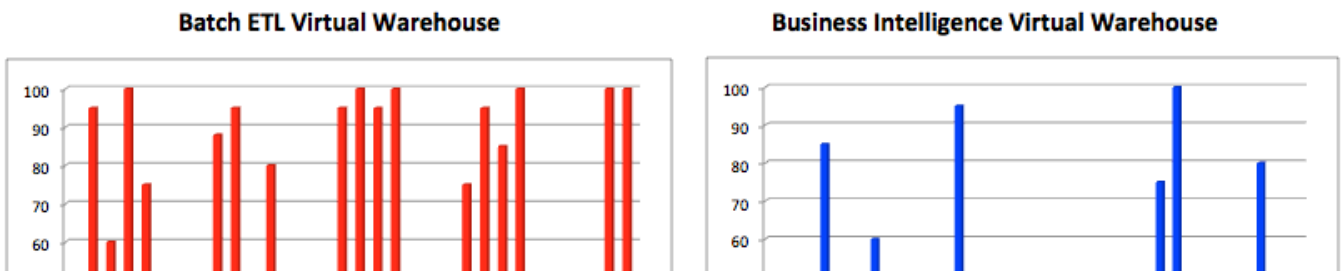
The diagram below illustrates one of the unique benefits of the Snowflake Data Warehouse as a Service solution. Instead of supporting a single MPP cluster over a shared storage service,

it's possible to spin up multiple independent clusters of compute resources, each sized and operating independently, but loading and querying data from a common data store.

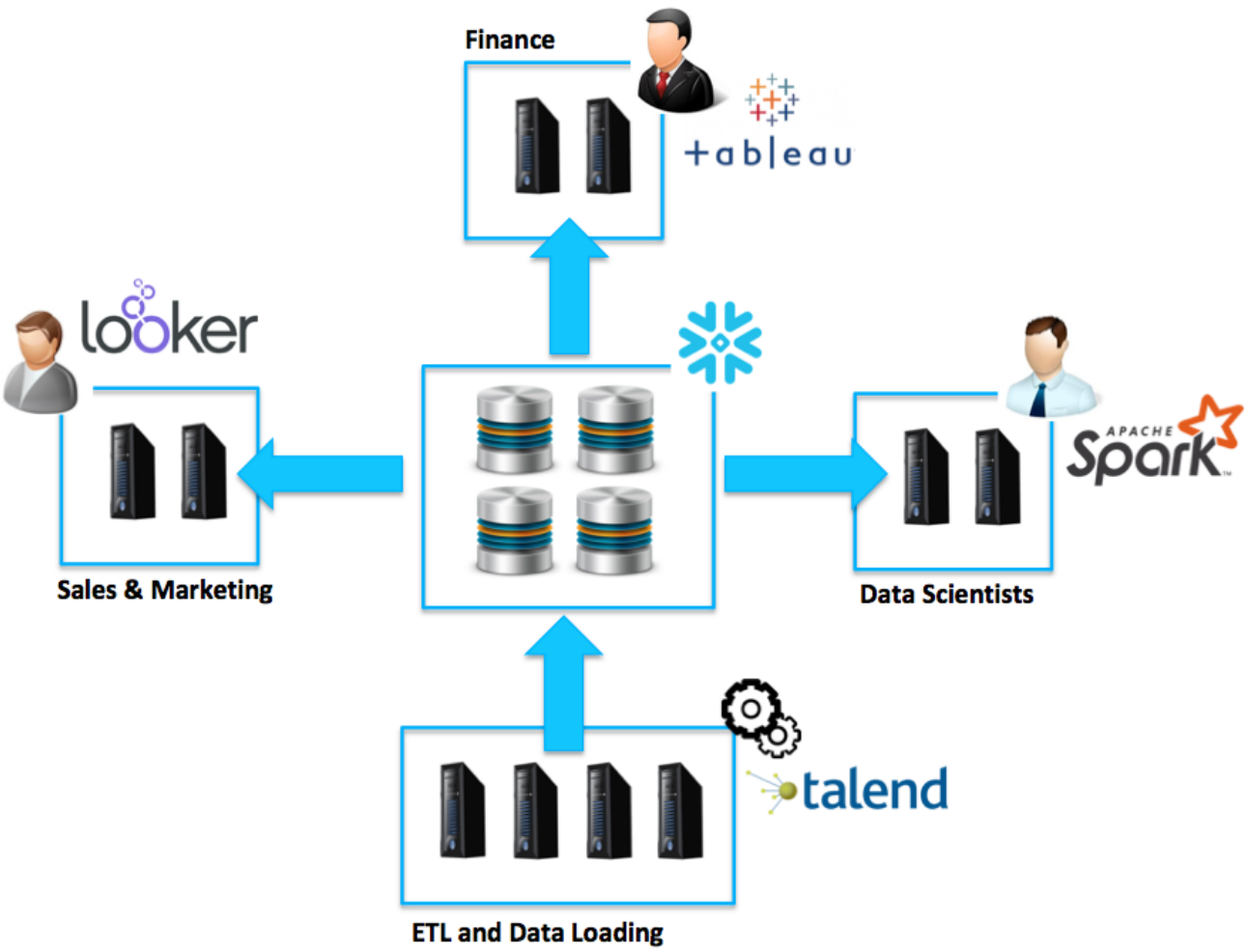


One of the huge advantages this provides is a remarkable level of agility, including the option to start up, suspend or resize any cluster instantly on demand with zero downtime or impact upon the currently executing workload. New queries are automatically started on the resized (larger or smaller) cluster as required.

The diagram below illustrates another key benefit, that it is possible to independently execute potentially competing workloads on the same shared data store, with large throughput workloads running in parallel with low latency, fast response time queries against the same data. This is only possible as a result of the unique ability to run multiple compute resources over a single shared data store.



Because Snowflake can deploy multiple independent clusters of compute resources, there is no tug of war (https://www.analytics.today/blog/the-ideal-data-warehouse-architecture) between low latency and high throughput workloads that you find on almost every other SMP, MPP or EPP system. This means you can run a truly mixed workload with intensive data science operations on the exact same data as batch ETL loading, while also delivering sub-second response times to business user dashboards. This is illustrated in the diagram below:

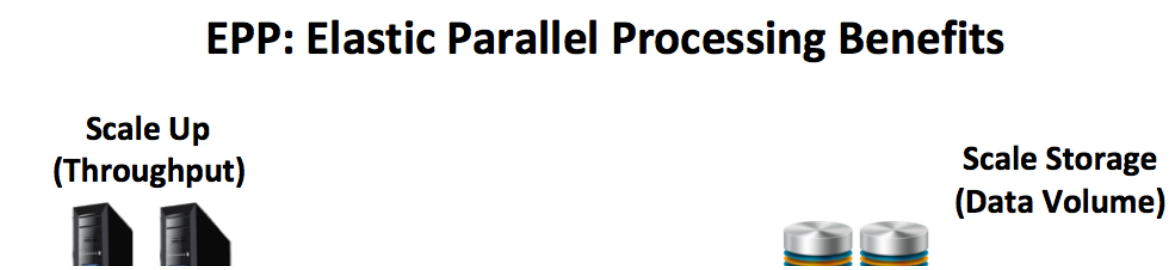


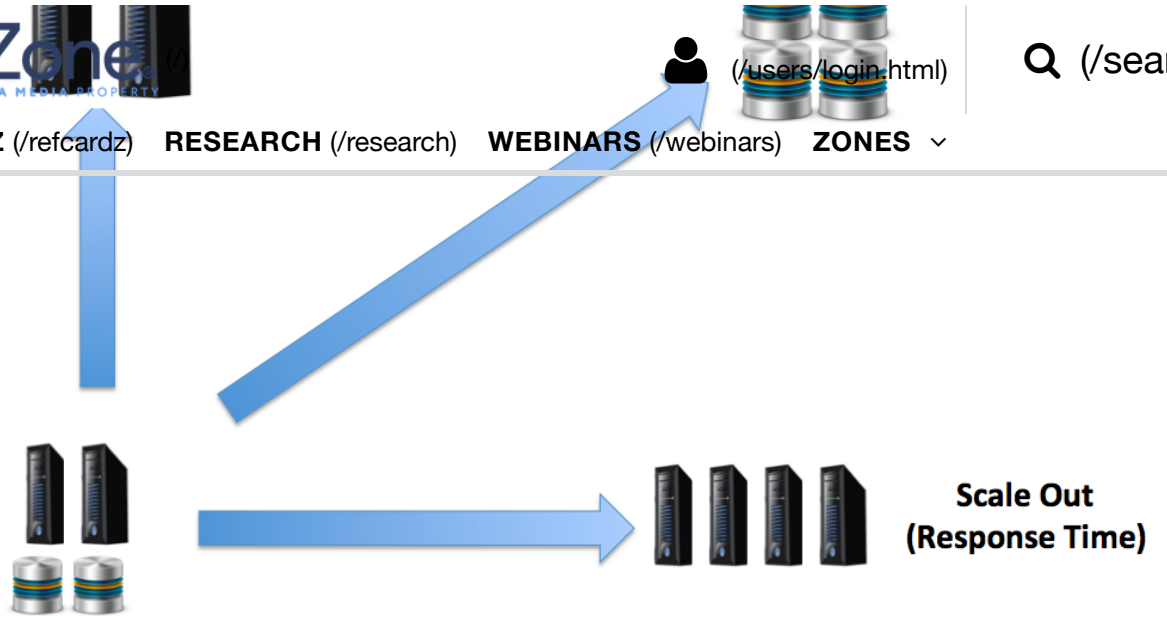
Finally, because there are multiple clusters, it's possible to both scale up and scale out the solution on-the-fly without any downtime or impact upon performance. Unlike some EPP solutions, Snowflake provides genuine elasticity, and can grow from a dual node to 128 node cluster, and back again without any interruption in service.

Advantages

EPP Solutions have similar benefits to the MPP architecture described above, but without many of the disadvantages. These include:

- **Scalability and Concurrency:** In addition to the ability to scale out, by adding additional compute nodes, EPP systems can scale up to perform increasingly demanding workloads or add additional same size nodes to maintain concurrency while the number of users increases.
- **Cost and High Availability:** Some EPP solutions can be deployed on-premises, hybrid, or in a cloud environment. Either way, the solution can in many cases be configured to provide high availability with automatic fail-over as needed. If deployed to a cloud environment, the option also exists to shut down or suspend the database to control costs, restarting when needed.
- **Read and Write Throughput:** As EPP systems are effectively an MPP solution with separate compute and storage, they have the same benefits as MPP in terms of throughput, but with additional benefits of scalability and elasticity.
- **Data Distribution is less critical:** In some cases (eg. Snowflake), data distribution is optional to maximize throughput when processing very large (over a terabyte) data sets, whereas, on other platforms (eg. Microsoft or Amazon Redshift), it's more important to set a correct distribution key to avoid data shuffling.
- **Potentially Zero Down Time:** Unlike MPP solutions, which typically need downtime to resize the cluster, EPP solutions can (for example with Snowflake) scale up or down the cluster size on-the-fly with zero downtime. In addition to the automatic addition or removal of nodes to automatically adjust the level of concurrency required, it's also possible to grow or shrink the compute resources on demand. On other solutions (eg. Redshift), the system needs to re-boot into read-only mode for the duration of the resizing operation.
- **Scaling all three dimensions:** Unlike MPP solutions, which typically only support a scale-out (addition of same size nodes), the EPP solution can independently scale compute and storage. In addition, it is possible to scale up to a larger (more powerful) cluster or add or remove nodes from the cluster. The unique ability of this architecture to scale across the three dimensions is illustrated in the diagram below. This shows the cluster can be scaled up to maximize throughput, scaled out to maintain an agreed response time as additional users are added (concurrency), or by adding data storage.





- **Right Size Hardware and Storage:** Unlike the SMP system — which tends to be inflexible in size — and both Hadoop and MPP solutions — which risk over-provisioning compute resources — an EPP platform can be adjusted to fit the size of the problem. This means a small cluster can be mounted over a petabyte of data or a large powerful system run on a smaller data set as required.

Summary and Conclusion

This article summarised the primary hardware architectures used to support a large analytics or business intelligence platform including the SMP (single node with multiple processors), MPP (multiple nodes with parallel data loading and distributed query processing), and finally EPP (Elastic Parallel Processing), which resolves many of the drawbacks of an MPP platform, and supports true elasticity and agility.

A number of database vendors have either deployed or have EPP architected solutions in development including Amazon Redshift, Google BigQuery, HP Vertica and Teradata, although to date, only one solution, Snowflake, provides full elasticity and on-demand flexibility with a zero downtime solution using multiple independent clusters.

While Hadoop may claim to provide a challenge to the traditional database, in reality, the drawbacks of system complexity and compute over-provisioning make this a poor solution for an analytics platform. Hadoop does, however, provide an excellent framework to deliver real-time processing and text analytics.

Either way, I strongly believe the compelling benefits of agility and cost control will mean increasingly all analytics and indeed all compute processing will eventually be performed in the cloud. I also believe the EPP architecture is the best approach to support an analytics workload, especially when cloud-based.

You can read a comparison of the market-leading options on a free eBook, A Comparison of

Cloud Data Warehouse Platforms (https://www.linkedin.com/pulse/data-warehousing-road-ahead-john-ryan/), although as almost any solution architect will attest, the best way to verify whether a given platform is a good fit for your use-case is to test it using a proof of concept.

Thank You

Thanks for reading this article. If you found this helpful, you can view more articles on Big Data, Cloud Computing, Database Architecture and the future of data warehousing on my web site **www.Analytics.Today (http://www.Analytics.Today).**

Topics: DATABASE, ARCHITECHTURE, HADOOP & BIG DATA, HADOOP, DATA WAREHOUSE, DATA WAREHOUSE ARCHITECTURE, ORACLE

Published at DZone with permission of John Ryan, DZone MVB. See the original article here. (https://www.analytics.today/blog/db-arch-compared)
Opinions expressed by DZone contributors are their own.

Popular on DZone

- **Extending Swagger and Spring Doc Open API (/articles/extending-swagger-and-spring-doc-open-api?fromrel=true)**
- **Memory Wasted by Spring Boot Application (/articles/memory-wasted-by-spring-boot-application?fromrel=true)**
- **Spring Boot: Displaying and Customizing Error Page (/articles/spring-boot-displaying-custom-error-page-customize-1?fromrel=true)**
- **What Do Engineers Really Think About Technical Debt? (/articles/what-do-engineers-really-think-about-technical-debt?fromrel=true)**

Salesforce Application Design

Get an overview of how to design application Salesforce Platform, exploring the no- to-full

Download the Refcard ►

Presented by **Skuid**

Get Started With Static Code Analys

Explore the necessary steps for getting starte analysis, including CI/CD integrations, OWAS and more. Read now ►

Presented by **ShiftLeft**

Getting Started With IaC

Infrastructure as code (IaC) means that you u and manage infrastructure rather than using Explore now ►

Presented by **Pulumi**

The NoSQL vs SQL debate is over

There's no more need to compromise. Couch requirements of modern application developr ►

Presented by **Couchbase**