Global Knowledge®

Expert Reference Series of White Papers

An Introduction to Amazon Redshift

An Introduction to Amazon Redshift

Rich Morrow, Global Knowledge Instructor, Cloud and Big Data Analyst

Introduction

Like any "big data" initiative, deploying and operating a data warehouse of any size used to be limited to only large enterprises with deep budgets for proprietary hardware and multi-year software licenses. Pay-as-you-go cloud products like Google's BigQuery and AWS's Amazon Redshift change all of that, putting a fully blown, fully managed data warehouse within reach of even the smallest business.

But do commodity costs equal cut-rate performance? Can even large enterprises rely on these for workloads that currently run out of commercial solutions like Oracle Exadata and Teradata? How is security and latency affected in a hybrid environment? In this article, we focus on Amazon Redshift, with an introduction into what it is (and is not), as well as how it compares with the costs, performance, support for third-party visualization tools, scale, and reliability.

What is a Data Warehouse?

Data storage and analysis is typically bifurcated into two types of systems, known as Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP), both of which are terrible acronyms that need a bit of explanation. OLTP is about serving real-time (low latency), high concurrent connections—traditionally served by Relational DataBase Management Systems (RDBMSes) like MySQL and Oracle. OLAP systems, on the other hand, are characterized by longer-running, more complicated queries against probably much larger volumes of data. An OLTP system could provide the data layer for a customer-facing web or mobile app so that the customer could complete a **transaction;** whereas an OLAP system serves internal users who are typically doing data mining in order to extract business intelligence **analytics**. Some examples of OLAP queries are "tally and sort all sales by salesperson by region by product for all of 2013," or "find the lowest margin products by category by region for Q3 2014"—data which could be used to either calculate salesperson bonuses or decide which product lines to terminate. For a reasonably busy retailer, either of those queries could involve sorting through terabytes of data (which probably exist in a couple of disparate, siloed, and perhaps even proprietary systems), and could take hours or maybe even days to run.

The hardware and software underneath these OLAP systems is what we refer to as a data warehouse (DW), which is sometimes also referred to as an enterprise data warehouse (EDW). Technically, OLAP refers to the operation being performed, while an EDW is the static hardware and software to support OLAP operations, although many refer to EDW systems as OLAP systems as well. These systems have very different considerations than OLTP systems—they need to be able to store and process dozens, hundreds, or maybe even thousands of TB of data, as cost-effectively as possible, and they need to be able to handle very long-running, complicated queries across those large data sets. EDW administrators also need to be able to easily and cost-effectively grow the storage and processing capabilities as their business grows.

Data generated from the OLTP systems is periodically (hourly, daily) copied into the EDW system via Extract, Transform and Load (ETL) operations meaning that the amount of data in an EDW grows very quickly (see figure 1). For this reason, the "cost per TB" of storage in an EDW has always been a big consideration.

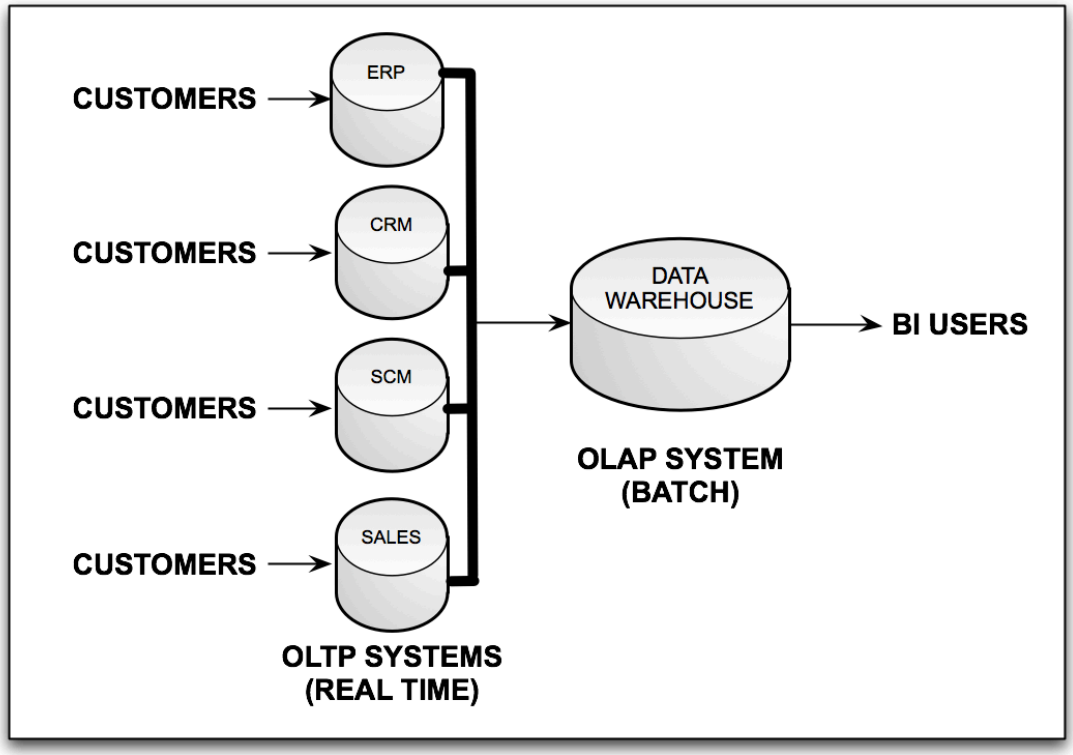

Figure 1: Internal and external customer needs served by OLTP, then moved (via ETL) into DW, which is used by internal customers for Business Intelligence.

In the pre-cloud days, any business hoping to do BI queries had to fork out millions to purchase the EDW hardware and software from a limited number of companies. The makers of these EDW systems could (and did) pretty much charge the maximum they could, and because of their high cost, businesses had to be very selective about what they chose to put into their EDW.

To serve queries as fast as possible, the EDW systems also needed to "pre-materialize" the data into what is known as "OLAP Cubes"— essentially cells representing the data for one of many dimensions. Figure 2 shows an example of such a cube, where we can see that the **customer** (dimension 1) "Fred Smith" purchased 234 digital cameras (**product**, dimension 2) in the **month** (dimension 3) of January. The downside of this pre-materialization meant that should the BI users also maybe like to add a fourth dimension (say region) to the cube, all of the data would need to be re-calculated—a process that could burn up hours or days (taking the system offline in the process), and result in much, much more data needing to be stored. Because adding an additional dimension often doubled or tripled the size of the stored data, some businesses had to keep their cubes limited to many fewer dimensions than they would have liked to analyze.

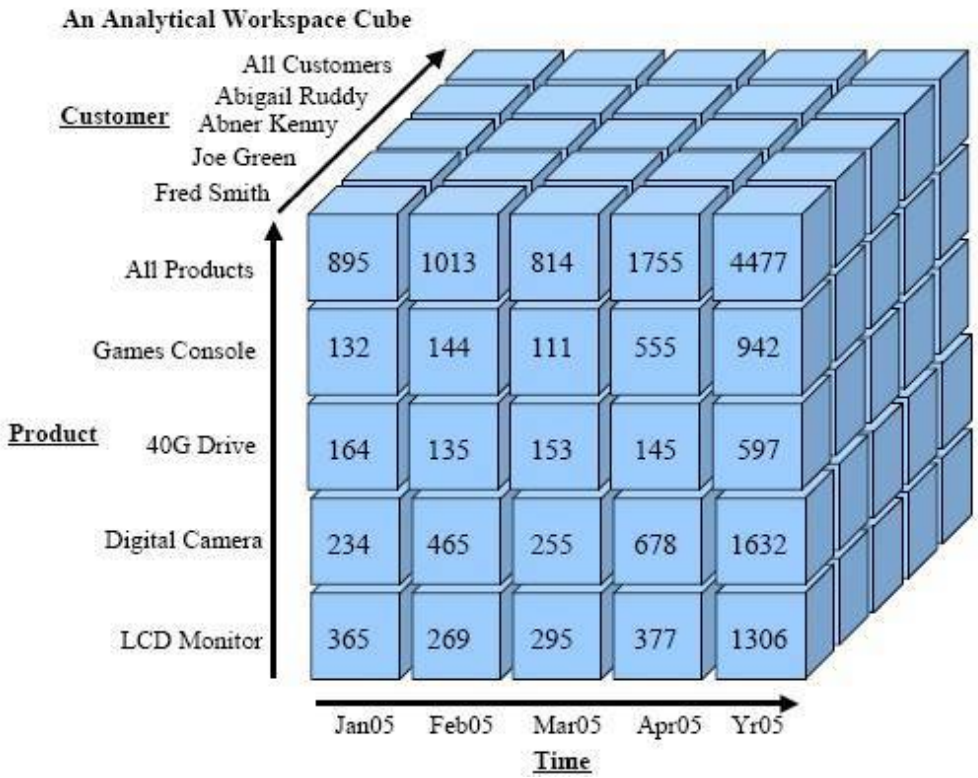

Figure 2: An OLAP cube showing the number of products purchased by specific customers for specific months

What is Amazon Redshift?

Perhaps one of the most exciting outcomes of the public cloud was addressing the shortcomings of traditional EDW storage and processing. The fast provisioning, commodity costs, scalable, and pay-as-you-grow pricing of public cloud are a natural fit for EDW needs, providing even the smallest of users the ability to now get valuable answers to BI questions. Amazon Redshift is one such system built to address EDW needs, and it boasts low costs, an easy SQL-based access model, easy integration to other AWS services, and most importantly, high query performance.

Amazon Redshift gets its name from the astronomical phenomenon noticed by Hubble, which explained the expansion of the universe. By adopting the Amazon Redshift moniker, AWS wanted to relay to customers that the service was built to handle the perpetual expansion of their data.

An Amazon Redshift cluster consists of one leader node (which clients submit queries to) and one or more follower (or "compute") nodes, which actually perform the queries on locally stored data. By allowing for expansion of follower nodes, Amazon Redshift ensures that customers can continue to grow their cluster as their data needs grow. Customers can start with a "cluster" as small as a single node (acting as both leader and follower), and for the smallest supported instance type (a DW2), that could be as low cost as $0.25/hour or about $180/month. By using "Reservations" (paying an up-front fee in exchange for a lower hourly running cost) for the underlying instances, Amazon Redshift can cost as little as $1,000/TB/year—upwards of one-fifth to one-tenth of the cost of a traditional EDW.

Because Amazon Redshift provides native ODBC and JDBC connectivity (in addition to PostgresSQL driver support), most third-party BI tools (like Tableu, Qlikview, and MicroStrategy) work right out of the box. Amazon Redshift also uses the ubiquitous SQL language for queries, ensuring that your current resources can quickly and easily become productive with the technology.

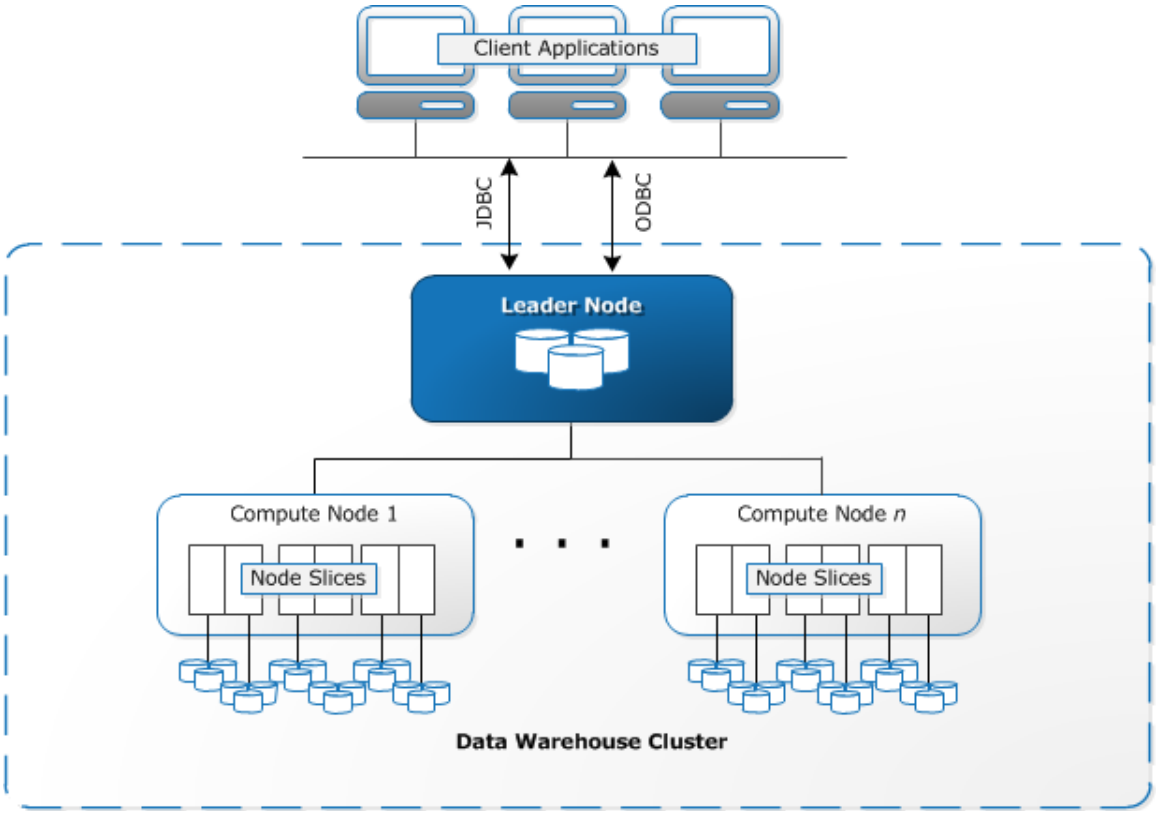

Figure 3: Amazon Redshift Architecture (Source: docs.aws.amazon.com/redshift)

Amazon Redshift was custom designed from the ParAccel engine—an analytic database which used columnar storage and parallel processing to achieve very fast I/O. Columns of data in Amazon Redshift are stored physically adjacent on disk, meaning that queries and scans on those columns (common in OLAP queries) run very fast. Additionally, Amazon Redshift uses 10GB Ethernet interconnects, and specialized EC2 instances (with between three and twenty-four spindles per node) to achieve high throughput and low latency. For even faster queries, Amazon Redshift allows customers to use column-level compression to both greatly reduce the amount of data that needs stored, and reduce the amount of disk I/O.

Amazon Redshift, like many of AWS's most popular services, is also fully managed, meaning that low-level, time-consuming administrative tasks like OS patching, backups, replacing failed hardware, and software upgrades are handled automatically and transparently. With Amazon Redshift, users simply provision a cluster, load it with their data, and begin executing queries. All data is continuously, incrementally, automatically backed up in the highly durable S3, and enabling disaster recovery across regions can be accomplished with just a few clicks. Spinning a cluster up can be as simple as a few mouse clicks, and as fast as a few minutes.

A very exciting aspect of Amazon Redshift, and something that is not possible in traditional EDWs, is the ability to easily scale a provisioned cluster up and down. In Amazon Redshift, this scaling is transparent to the customer—when a resize is requested, data is copied in parallel from the source cluster (which continues to function in read-only mode) to a new cluster, and once all data is live migrated, DNS is flipped to the new cluster and the old cluster is de-provisioned. This allows customers to easily scale up and down, and each scaling event nicely re-stripes the data across the new cluster for a balanced workload. In a traditional, hosted EDW environment, a resize would typically involve weeks of preparation and days of downtime, along with a hefty six- or seven-figure bill.

Amazon Redshift offers mature, native, and tunable security. Clusters can be deployed into a Virtual Private cloud (VPC), and encryption of data is supported via hardware accelerated AES-256 (for data at rest) and SSL (for data on the wire). Compliance teams will be pleased to learn that users can manage their own encryption keys via AWS's Hardware Security Module (HSM) service, and that Amazon Redshift provides a full audit trail of all SQL connection attempts, queries, and modifications of the cluster. The AWS CloudTrail service additionally logs all API calls against Amazon Redshift, and both the native SQL logs and AWS CloudTrail logs can be exported and queried.

Now that we know what Redshift is, let's also clarify what it is not. It is not a NoSQL engine. It is not a suitable solution to search through large collections of text documents. It is not an RDBMS, nor is it intended to serve OLTP for external customers. It is not a real-time analysis engine. It is not a good place to store anything but structured data. It is not the fastest way to analyze data, nor is it the cheapest way to store data. Instead, it is a cloud-based EDW that allows internal users to quickly perform business analytics on large collections of both rolled-up and granular data.

Amazon Redshift vs. Other Solutions

When building an overall big data architecture, an EDW is just one piece of the puzzle, and to add to the confusion, some of the other pieces in that architecture overlap in some ways. In the public cloud alone, you'll often find EDW products like Amazon Redshift compared or contrasted against Hadoop, text search engines like Solr, NoSQL engines like DynamoDB, and real-time analysis engines like Kinesis. To really understand where Amazon Redshift fits among these other technologies, we need to understand the raison d'etre of each.

As a batch analysis data platform, Hadoop is probably the product that compares most directly to Amazon Redshift. Although both exploit horizontal scaling to perform parallel batch (non-real-time or "offline") processing, the approaches and benefits of each are quite different. Front and center is the storage paradigm. Amazon Redshift requires users to impose RDBMS-like table structure around their data, but Hadoop imposes no such structure. In Hadoop, users can load structured (think tables), semi-structured (think e-mails, with some structure but a big blob of text in the body), or unstructured (think graphics or CLOBs) data in the cluster.

For processing, Hadoop uses the somewhat dated MapReduce paradigm, which is now coming up on its tenth birthday. Although programming in MapReduce requires knowledge of both MapReduce and a programming language like Java, users can leverage Hive— another Apache project which imposes table-like structure on Hadoop data, and exposes SQL as the access method. Under the hood though, Hive just takes the SQL queries and converts them to the somewhat slow MapReduce processing model.

At a high architectural level, Hadoop is much, much more than Amazon Redshift. It has a very rich ecosystem, comprising many other Apache projects that can do everything from real-time data analytics to workflows to providing alternate Domain Specific Languages or DSLs like the Pig project. Whereas Amazon Redshift is focused on providing an easily managed EDW, Hadoop is focused on providing a general purpose "data hub," which can service many different types of workloads. Hadoop is generally hailed as the cheapest way to store and process data, but many workload comparisons show Amazon Redshift as comparable. Depending on the workload, Amazon Redshift can even be both significantly faster and cheaper than Hadoop.

At AWS, Hadoop can be provided by either custom-building your own cluster on top of EC2 (virtual servers), or by leveraging the Elastic MapReduce (EMR) service—a fully managed Hadoop cluster with many of the "managed service" benefits of Amazon Redshift. If you have a true EDW need, however, Amazon Redshift will serve you much better and more cost effectively than even EMR.

Text-based search is a very different need than an EDW, and to fill this need, many mature products like Apache Lucene, Apache Solr, and Elastisearch already exist. In AWS, text search is provided via the Amazon CloudSearch service, which is essentially a fully managed, pay-as-you-go deployment of Solr. When you need to allow your users to perform detailed text searching (faceting, highlighting, weighing, sorting, etc.) on large collections of text documents, choose one of these text search technologies.

Amazon Redshift is sometimes incorrectly referred to as a NoSQL database, but make no mistake, it is most definitely not. NoSQL databases like Apache CouchDB, Apache Cassandra, and AWS's own DynamoDB are a different beast—mainly aimed at providing real-time, low-latency responses to a high number of external user queries. You'll often find NoSQL used in combination with an RDBMS, where the "hot" data (with high throughput or high volume requirements) resides in the NoSQL solution and data requiring more transactional capabilities (rollbacks, ad-hoc querying needs) is stored in the RDBMS. Many AWS users leverage DynamoDB for transient, high throughput needs like user sessions, mobile gaming statistics, and the like. Again, Amazon Redshift differs dramatically from this use case in that it focuses on read-heavy analytical processing. For an EDW, Amazon Redshift is very fast (returning results in seconds or minutes), but it's never going to be as fast as a NoSQL solution (returning results in milliseconds) that was designed specifically for this purpose.

Real-time data analysis is all the rage these days, and for very good reason—businesses find real value in being able to respond more quickly to trends. Real-time analysis engines like Spark/Shark, Storm, and AWS Kinesis typically analyze data in memory as it flows through the ingestion phase. Often just the results of that analysis itself are persisted to disk, and the raw data is discarded. The number one benefit of these "analysis on ingestion" engines is that they can accomplish much faster analysis on extremely large volumes of data. For example, take a company wishing to monitor its brand reputation by analyzing tweets that contain their corporate name and select products. This company could filter the Twitter fire hose, just saving off all relevant tweets and then analyzing the results either hourly or weekly in something like Amazon Redshift. However, if it's really important that the company respond quickly to negative tweets, they may want to use the Kinesis service, which would provide such insight (and the ability to respond) in real time.

Where Amazon Redshift Fits in the Enterprise

With the recent explosion of data storage and processing technologies, many of which overlap in some ways, it's easy to be confused about how, where, and when to leverage an EDW in general or Amazon Redshift in particular. Having many options can be both a blessing and a curse, but a cloud-based EDW like Amazon Redshift still has a very important and distinct role in an Enterprise.

The sweet spot that any EDW occupies is business analytics. Any business has some very important metrics that it needs to collect and report on for historical determinations of which areas/sectors/products/divisions are doing well and which are not. Obviously, the metrics we need to collect in such a system are but a small subset of the overall metrics that other divisions like marketing, IT, and app development teams need.

From a raw dollars per terabyte cost perspective, Hadoop will likely remain unbeatable—it's hard to get lower cost than commodity hardware running open source software. For this reason, a lot of organizations are moving their "big data" architectures to resemble figure 4 below.

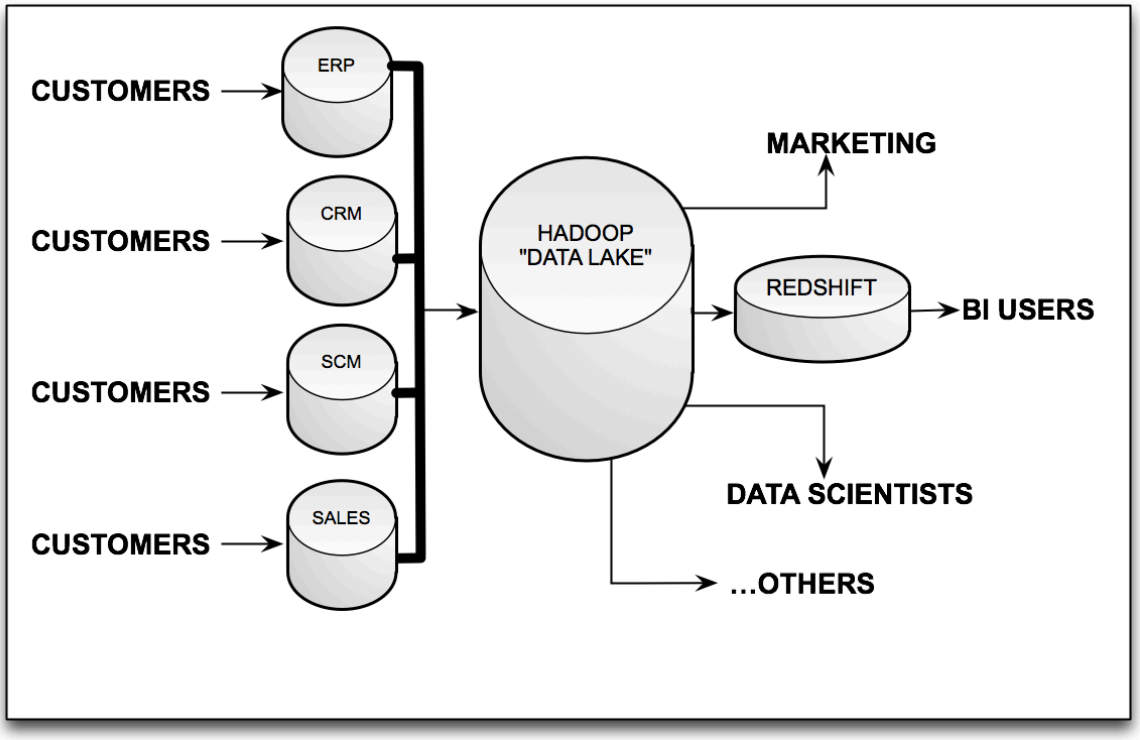

Figure 4: Amazon Redshift as a downstream consumer of Hadoop.

In this architecture, data from our OLTP systems are no longer put directly into the EDW (Amazon Redshift in this case), but instead first flow directly into Hadoop. Because Hadoop is a more general, lower cost tool, we can both store huge amounts (petabytes) of data in it, and allow multiple downstream systems and users to pull data from it, or analyze data directly in it.

In this architecture, our BI-specific metrics would be copied from the Hadoop "data lake" into Amazon Redshift, where our BI folks could run their workloads. Hadoop can act as a consolidation area for the data before it goes into Amazon Redshift, whereby users get much more responsive results by querying smaller amounts of data, and the business saves costs by storing the bulk of their data in Hadoop. Users such as marketing, still wishing to extract value from either BI or non-BI specific data, can still go fishing in the data lake, and remove some of the load from Amazon Redshift.

With AWS Direct Connect capabilities (the ability to rent a dedicated, secured 1 or 10Gb line directly from a corporate datacenter into AWS), and VPC (a secured, cordoned-off private cloud which allows full control of network topologies, routes, and NACLs), enterprises can run really any part of such an overall architecture either on-premises or in the public cloud. Even if an organization has significant internal investments in Hadoop and runs all other aspects of their business from their own cost-effective data center, flowing the BI subset of that data into Amazon Redshift and pointing BI users to that new endpoint is incredibly easy to do, and almost guaranteed to save dollars—possibly a lot of dollars—in the form of avoiding future EDW hardware and software purchases.

Third-Party Tool Support

One of the main reasons it's so easy to transition from a traditional on-prem EDW to Amazon Redshift is the third-party tool support. The vast majority of BI users don't run their queries directly against the EDW. Instead, they leverage tools like Tableau, QlikView, MicroStrategy, TIBCO Jaspersoft, and others which provide point and click, visual interfaces that make the EDW a whole lot more accessible, easy to navigate, and easy to extract value out of. In addition to providing visualization capabilities, most of these tools also allow for easy loading and data transformations as well.

Because of its open standards JDBC and ODBC connectivity, Amazon Redshift natively supports all of the aforementioned leaders in the BI visualization tool space as well as many, many others.

Transitioning to Amazon Redshift can literally be a couple of days of effort, the very last step of which would be repointing the BI users to the Amazon Redshift endpoint. From that point forward, it would be completely transparent to the BI users that the back-end EDW is now hosted in public cloud (assuming, of course, that they have a good, stable, secure low-latency connection, like Direct Connect, to that public cloud). AWS's generous two-month free trial of Amazon Redshift also makes it very low risk for companies to test and compare Amazon Redshift by migrating some of their EDW data into Amazon Redshift and then running their own cost and performance benchmarks.

Conclusion

Amazon Redshift is one of the most talked about and fastest growing services that AWS has ever released, and hopefully we've been able to relay to you some of the big reasons why. Amazon Redshift opens up EDW capabilities to even the smallest of businesses, yet its costs, security, and flexibility also make it appealing to the largest of enterprises.

It allows companies to easily, conveniently scale their EDW needs both up and down, and as a managed service, allows your team to offload all of the "undifferentiated heavy lifting" of building and maintaining an EDW. Its raw storage costs are about one-fifth to one-tenth of traditional in-house EDW, and AWS has taken great care to ensure its performance is still competitive with those in-house solutions.

Before deciding to use Amazon Redshift, however, it's important to understand what it is and is not. There is a great deal of misunderstanding and outright misinformation on the web, and one needs to take great care in where and from whom they are getting their information. In addition, public cloud services like Amazon Redshift continue to evolve, and there is great risk in getting outdated information as well—all the more reason to ensure you're getting information either directly from AWS or from an AWS-approved partner like Global Knowledge.

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge through training.

- Architecting on AWS
- Architecting on AWS - Advanced Concepts
- Systems Operations on AWS
- Developing on AWS
- Big Data on AWS

Visit **www.globalknowledge.com** or call **1-800-COURSES (1-800-268-7737)** to speak with a Global Knowledge training advisor.

About the Author

Rich Morrow is a 20-year open-source technology veteran who enjoys coding and teaching as much as writing and speaking. His current passions are cloud technologies (mainly AWS and Google Cloud Platform) and big data (Hadoop and NoSQL), and he spends about half of his work life traveling around the country training the Fortune 500 on their use and utility. He leads the Denver-Boulder cloud computing group, as well as quicloud, a cloud and big data consultancy firm.