ROZETTA TECHNOLOGY

THE EVOLVING LANDSCAPE OF DATA STORAGE

Stephen Johansen - Head of Technology
RoZetta Technology

Many IT Managers and Data & Solutions Architects are constantly faced with the problem of managing large and complex data sets that grow exponentially. We define large data sets as being measured in Petabytes or even Exabytes. This requires database and system architectures beyond traditional single-node relational database management systems (RDBMS), and not all IT organizations have the capability to deal with these complex systems.

Here we outline when these large-scale systems are necessary, the trends in data system design as the data volumes have increased, and how these techniques are used in our proprietary tick history platform that forms the core of our DataHex SaaS solution.

Defining the Problem of Scale

For most of the second half of the 20th century, the single node RDBMS was the data storage workhorse at the heart of every IT architecture. These systems tended to be installed on specialized, high-end hardware, and this required a team of specialist Database Administrators (DBAs) who ensured they remained reliable and fulfilled queries in a reasonable time. Many enterprises would standardize on one database vendor and ensure their teams were fully trained on that single database technology.

RDBMS systems were well understood by Database Administrators, Application Developers and Data Analysts, all of whom interacted with the database using variations of SQL. It was well understood how to structure the data, index it for optimal performance, and tune SQL queries to make the best of the hardware and software.

Data volumes tended to grow linearly once a system was established, so it was feasible to manage growth by buying larger servers, referred to as vertical scaling. These systems were general-purpose, with almost any application data storage or analytical problem solved with the same tool.

This changed completely in the late 90s with the rise of the web and the explosive growth in data generated as consumers connected to it. Data grew exponentially rather than linearly, and there was simply no way to keep buying bigger servers. The new, big web giants like Google required data management systems outside the expertise of most enterprises.

These companies couldn't outsource the engineering of these software and hardware systems to the specialist database vendors, so they hired Computer Science talent directly out of universities to develop systems to manage this new scale of data.

This gave the industry several new ways of managing big data sets:

- Horizontal scaling: Growing beyond a single node to lots of nodes.
- Disaggregation: Splitting components of the database into different specialist systems.
- Data Services: Buying or building these specialist systems independently from each other, often delivered as Cloud services.
- Data Vendors: Buying the data in a useful form directly and letting specialists handle data management.

Growing Beyond One Node

The world got its first real glimpse of this when Google started producing its seminal data management papers in the early 2000s. With the Google File System (GFS), Map Reduce and BigTable papers, the industry realized that it was possible to build systems completely engineered for scale. These systems were built as horizontally scaled systems, where large nMap reduce numbers of low-cost commodity servers were chained together with intelligent software and tended to be specialized for specific workloads. The BigTable paper alone was the catalyst for the non-relational, NoSQL database movement that has dominated the last decade or so of data management thinking.

All parts of a single node database work together to ensure system reliability and good query performance to the end-user. However, as you need to scale beyond what a single node database can handle, the network becomes a critical system component. This introduces latency and reliability issues. This trade-off becomes necessary beyond a certain scale, so these new systems embraced the fact components could fail or become degraded and planned for this in advance.

Eric Brewer's CAP theorem captured this, in which in its current revision has three definitions involved:

- Network Partition (P): This is a situation where not all of the nodes in the system can see each other and agree on the state of the overall system.
- Consistency (C): In the face of a network partition, do we expect all nodes to continue to see every change applied in order (strict consistency) or do some nodes see stale data for some time (eventual consistency)?
- Availability (A): In the face of a network partition do we expect the overall system to remain available for reads, writes or both?

Systems tend to either be CP systems, which become unavailable in the face of network partitions but always show a consistent view of the system state, or AP systems, where the system remains available but clients see different states of the data until the network partition is resolved and the data becomes consistent.

Regardless of which system is chosen, Application Developers need to understand these trade-offs. Some of the work traditionally done by specialist DBAs now needs to reside with the development teams that build on top of these data systems.

The shape of the data also becomes an important factor, as these systems also give up general purpose query performance to become very good at delivering specific query types at scale.

Splitting the Database

A traditional database was made up of several parts:

- Database storage: The files on the disk where the actual rows of data are stored.
- Database buffer or cache: An in-memory copy of the most frequently accessed chunks of data.
- Database indexes: Copies of parts of the database tables that allow for fast access to particular rows based on the values in a field that is indexed.
- Query engine: The software processes that take end-user connections and plan and execute their SQL statements and return results to them.
- Transaction Log: A log of every change to the database that must be written before that data is considered safely stored in the database.

As data management systems scale, it is often necessary to disaggregate these parts into separate, specialist systems. Google showed this was possible by splitting their data storage onto their Google File System (GFS) from their computing layer query engines like Map Reduce and BigTable.

This directly led to the open-source Hadoop project, which was an attempt to build a similar system for anyone that needed to manage big data sets. Hadoop had its separate storage (HDFS), compute (first MapReduce, then later YARN), and database query layers (HBase, Hive, Impala, Spark).

This allowed for much larger systems than would ever have been possible on single node databases. It meant adding networking and system complexity within a system and between systems.

Much of the disappointment arising from the failure of Hadoop and Data Lake projects of the mid-2010s stemmed from the amount of overhead required to manage this complexity and the gap between the promises these systems made and the delivery of actual business value created.

This pattern also led to the rise of specialist platform engineers and data engineers responsible for managing these complex systems and ensuring data flowed into and through them efficiently.

These roles needed to understand distributed systems at a level not necessary for traditional specialist DBAs and ETL (Extract-Transform-Load) Engineers and as a result were much harder to hire for.

Database Cloud Services

At the same time, as data systems were distributed to manage massive volumes of data, a revolution was happening on how IT services were provided and consumed. The introduction of Amazon Web Services (AWS) and their later competitors gave the world a business model where it was possible to only pay for the amount of IT services used. It also had the side effect of pushing a lot of complex system management onto specialist engineers working for those cloud vendors, doing what AWS calls "undifferentiated heavy lifting".

Moving systems to the cloud was a massive mindset shift and initially met with skepticism, however almost 15 years after AWS started there is not an industry that is not heavily invested in this model of IT delivery.

While some workflows are hampered by tight latency requirements, such as High-Frequency Trading (HFT) or compliance and regulatory surveillance, much of the IT world is invested in building on cloud-first and designing their architectures in a way that takes full advantage of the Cloud Native approach.

This had a massive impact on how complex data systems are built. If you only pay for the storage you need, only buy the specialist data systems required for your current scale compute resources, and most importantly, turn them off when not required, it opens a lot of possible system architectures that were not previously feasible to build in-house. As the big Cloud providers invest heavily in building and managing state of the art data systems and specialists, single product Cloud-Native businesses like Databricks and Snowflake build on top of these platforms, the range of choice becomes almost endless.

That choice however also allows for ever more complex systems to be built. While some of the TCO is offloaded to the cloud vendors, and the risks involved in investing heavily in hardware and software upfront are removed, there is still endless scope to build systems that are hard to manage.

Engage Trusted Partners to Manage the Data

The pure complexity of what is required to manage petabytes or exabytes of data is not the only reason it can make sense to let specialists do the work for you. All data requires the knowledge of Subject Matter Experts (SMEs) to ensure that the data is available and understand the business-specific context that data resides in and how to ensure it is accurate, clean, and free from errors.

This has led to many businesses outsourcing the management of specific data sets and then acquiring that data in ready-to-use formats, sometimes referred to as 'application ready'. It has the added benefit of not having data management as a core business competency in a non-technology industry. This is the key concept behind our DataHex platform.

RoZetta's DataHex SaaS Platform

The key idea behind RoZetta's management of tick history is to ensure anyone that needs capital markets data, at the exchange or even instrument level, can do so in a timely manner. The current generation system was built entirely on the cloud and is built on several systems working together.

We manage this complexity on behalf of our customers so they don't require large teams of cloud & data engineers in-house. We marry our large-scale data management knowledge with our in-house Capital Markets Data SMEs, who ensure that the data is accurate, and our technology ensures that a client's SMEs have the tools required to do their job.

Rozetta Technology has a proud history of managing capital markets data, and our ever-evolving management of tick history has made the shift from on-premises to cloud-based technology.

DATAHEX
Data Management Platform

Modules within Our DataHex SaaS platform are designed to:

- Efficiently ingest raw market data and make it available for users in a range of roles.
- Efficiently store multiple petabytes of historical data.
- Allow users to have a subset of this data delivered to them, from the entire data set to the history for a single stock exchange and down to a single time slice of data for a single instrument code.

RoZetta has built DataHex based on several key cloud-based technologies that promise timely and accurate capital markets data. It is an example of a disaggregated database system architecture, heavily tuned to a particular set of use cases.

We store historical capital markets market data in a proprietary binary partitioned data format on the AWS S3 service at the lowest level. We chose S3 primarily for its incredible reliability and relatively low storage cost. Glacier storage service was not an option as users within our client base need all the history readily available. We also build a proprietary distributed index layer on top of S3 to allow for fast retrieval of individual instruments and time slices for both data previews and full data extracts. This role would use indexing within a traditional database, not as a separate service.

The indexing and tuning of this database are critical to the performance of data discovery by our end users.

RoZetta has a separate Amazon Aurora cluster that houses our metadata layer. This layer stores reference data about instruments and other entities that aid with searching for a portfolio of instruments to be extracted.

Our data loading process is built to efficiently populate both the binary storage layer on S3, the indexing layer and the metadata layer in Aurora to ensure there is no disparity between what we store and what we can find. This process is built on custom code executed as a fleet of Docker containers on the Elastic Container Service (ECS). We maintain some pools of compute for this service on constantly running EC2 virtual machines to ensure we can always meet our load SLAs. For historical backloads we often run these container tasks on the serverless Fargate ECS backend, which allows us to only pay for the actual run time of our tasks.

These kinds of trade-offs are common when building cloud-native services and we have taken our business context into account when making these choices.

Finally, our data extraction and fulfillment service run custom pipelines on top of ECS container tasks and makes heavy use of the indexing service to find the most efficient spot on S3 to start scanning for a particular time period, exchange and set of instruments to extract. We define the exact dimensions we need to access the raw data, which lets us optimize this fulfillment process. The loading and extract services act as a specialized subset of the query engines found in traditional database systems.

Conclusion

The capital markets services industry continues to find new and interesting market data to inform their trading strategies. That means platforms must continue to scale to store and serve the demand for this data. We also have trends like the rise of cryptocurrencies, a mind-boggling array of new instrument types and the catch-all term of Alternative Data that only means data volumes will continue to grow. RoZetta's DataHex SaaS platform will continue to evolve to meet this need. Our team of engineers and market data experts can take the challenge of managing these complex systems away from your business and let you focus on maximizing the return on your data assets.

Contact our ***Head of Technology, Stephen Johansen***, via email at **stephen.johansen@rozettatechnology.com** to find out how we can maximize the value from your data assets.

Stephen Johansen
Head of Technology, RoZetta Technology

ROZETTA TECHNOLOGY

www.rozettatechnology.com stephen.johansen@rozettatechnology.com