Introduction to the Active Everywhere Database

DATASTAX
The power behind the moment.

INTRODUCTION

For almost half a century, the relational database management system (RDBMS) has been the dominant model for database management. This more than 40 years old relational architecture powers most of the database management systems in use today. It provides powerful mechanisms to store and query structured data under strong consistency and transaction guarantees, and has reached an unmatched level of reliability for yesterday’s workloads.

However, today's business needs are forcing data management into new use cases, such as managing interactions with constantly-connected devices and websites, powering and managing social networks, handling multi-structured data and machine-generated files, etc. The amount of useful data in some modern application areas has become so distributed and vast, and the speed at which that data moves so fast, that it cannot be stored or processed by traditional database solutions. User-generated content in social networks and data retrieved from large sensor networks are two of the many such examples where traditional relational databases are being pushed beyond their limits. These data systems were designed for fixed schemas and centralized workloads, and not for more high throughput, always-on, massive scale-out and flexible data needs.

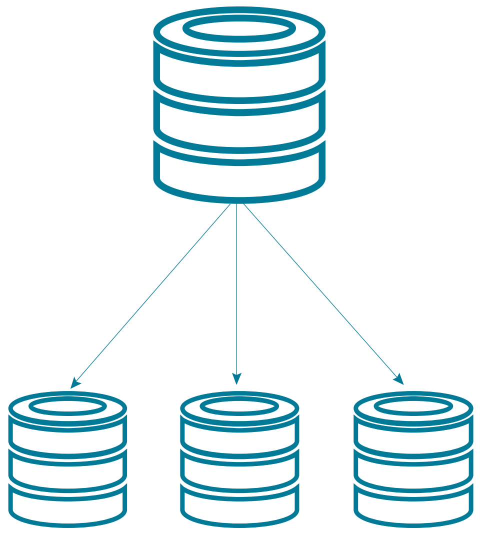
NOSQL DATABASES

More than ever, companies today are relying on data to power their day-to-day operations. Their IT departments must support global access by millions of users on connected devices and by smart machines of all types. This not only requires geographic data distribution, but also the ability to handle massive amounts of reads and writes on the database in real time, without worry of downtime. The rapid rise of these new kind of data requirements led to a new class of data management systems called **NoSQL databases**, which are being used in conjunction with traditional RDBMSs. NoSQL databases offer the promise of horizontal scalability and higher availability than relational databases and have an approach to data management that’s useful for very large sets of distributed data. Some are designed for high volume, web-scale applications in clustered environments delivering the infinite scale and high availability required to meet the growing needs of businesses today.

WHAT IS HIGH AVAILABILITY?

Most databases are designed to deliver “high availability.” But what does that mean? Conceptually, a highly available database uses an architecture that is designed to continue to function normally when there are hardware or network failures within the system. All older (relational) database systems and most of NoSQL database systems are built on a single server and rely on a master/slave architecture to provide availability. In this system, as you add more machines to your database for scale, all write traffic must be controlled by a single control point (the master). No matter how sophisticated you try to make it, this type of architecture inherently introduces bottlenecks to performance and operational complexity at scale. In theory, this model addresses the issue of read availability, but in practice, it does so only in a limited sense as there are still many potential single points of failure as well as scale limits for heavy write applications (e.g. IoT).

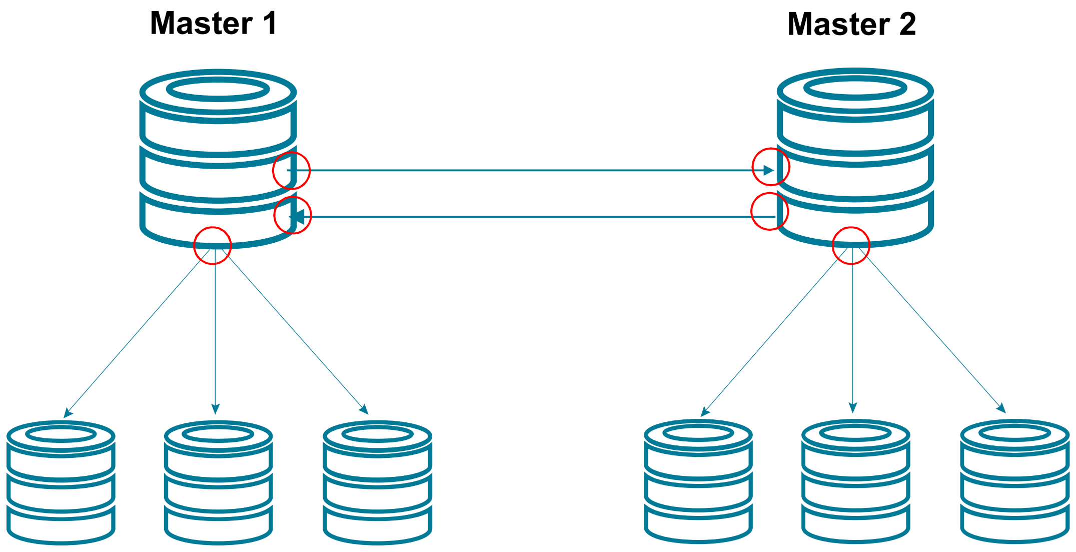
Master Database



These databases can be fine for smaller, centralized workloads, but when it comes to distributed applications, they have numerous fatal flaws—and when you introduce hybrid and multi clouds into the picture, the story gets even worse.

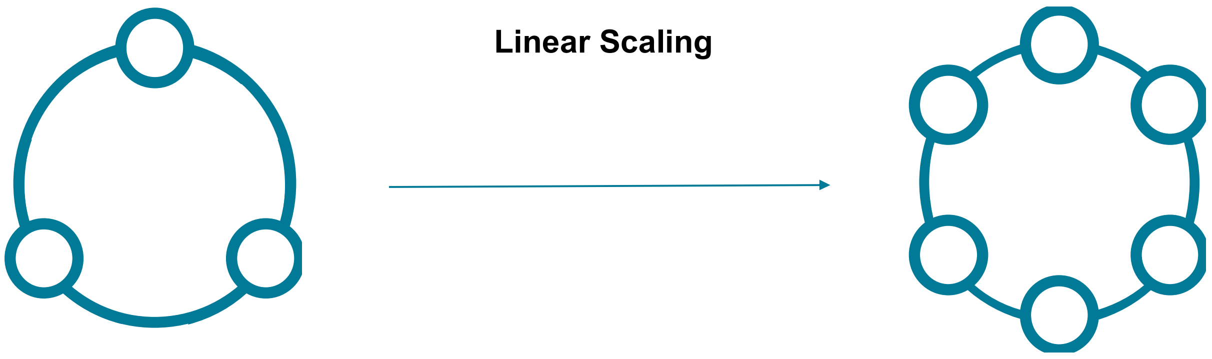
For one thing, how do you avoid downtime? In a master-slave architecture, when the master is out of service, the entire database is no longer fully functional, which means it will lose its ability to continue to accept full read/write workloads until a new master is elected. To resolve this, you can introduce the concept of a multi-master design strategy. However, the multi-master strategy struggles across widely distributed networks, and performance in this model is clunky and anything but straightforward. This introduces **several single points of failure,** and trying to figure out how to keep the whole database fully functional in the event of machines going down is extremely complicated and has a big impact on performance.

Issues compound in widely distributed applications that require more than two geographic locations for the database. When you start to apply the above model to a database spanning 3, 4, 5, or N number of data center locations, the complexity explodes and performance erodes. It is simply not built for a world where data reads/writes have to happen everywhere.



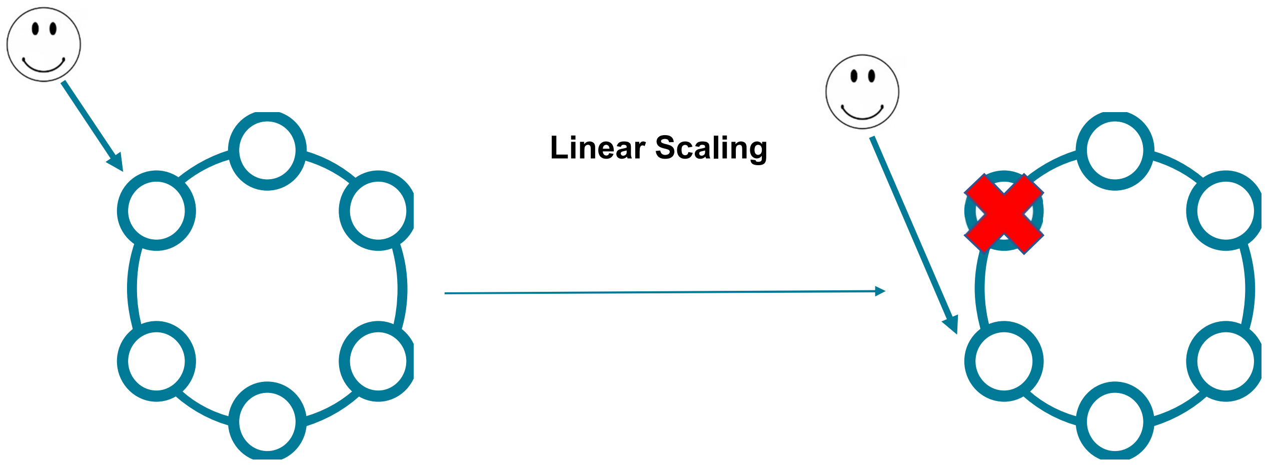
ACTIVE EVERYWHERE DATABASE

DataStax Enterprise (DSE) is the **Active Everywhere Database** built on Apache Cassandra's masterless architecture, where multiple servers (or nodes) are grouped together in a cluster. With DSE, scaling is truly effortless and highly predictable. In fact, it's linear. If you want to double the performance of your cluster, you simply double the number of nodes in the cluster. That's it! And you can do this infinitely. Moreover, as these systems scale, you can have the data closer to the application within any region in the world. This cuts down data transfer costs and increases performance by reducing latency. In master-slave and multi-master models, this is very hard to do and comes with performance penalties, but with DSE, it is easily and automatically configured.

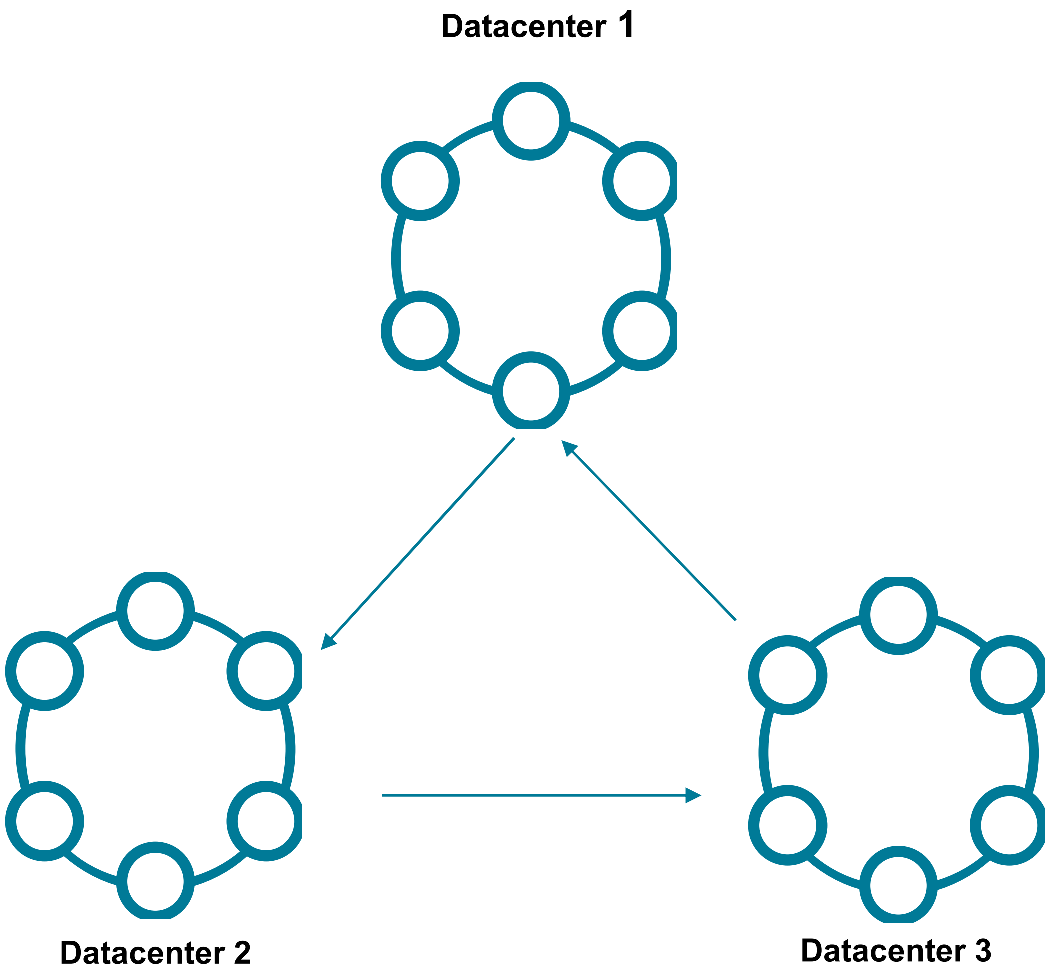


High Availability Database Architecture

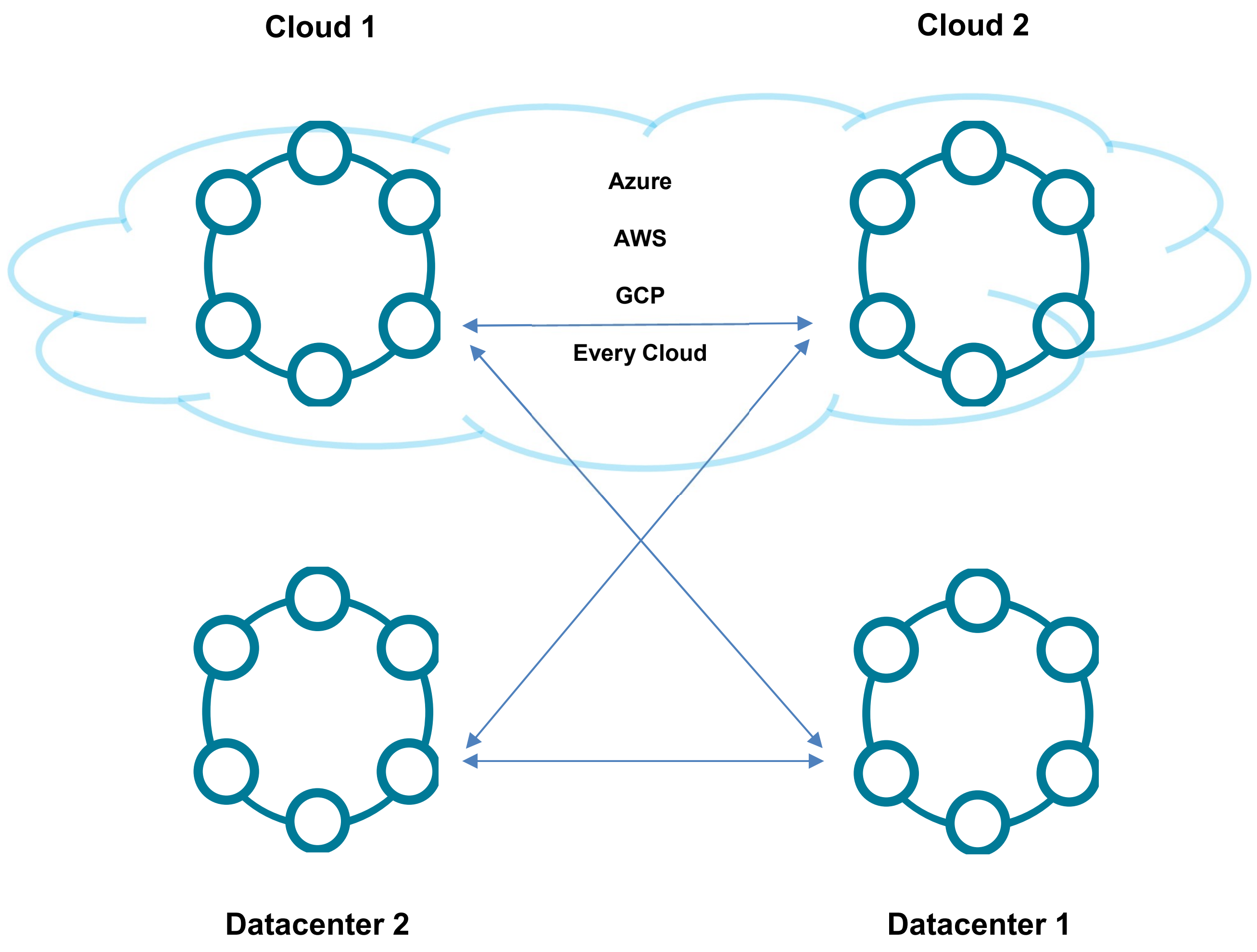
As important as scaling is, today's modern applications need a lot more. A highly available database needs to eliminate any and every single point of failure and should be optimized to ensure no interruption in service for the user when hardware or networks fail. In a master-slave or multi-master database configuration, machines going down can cause massive headaches and halt business. And if you start getting too complicated with failover strategies, the performance drops considerably, especially at scale. In DSE, continuous availability is built into its core architecture. Because there is no master, any node within a cluster can respond to a read or write requests. Each node is identical, autonomous and 100% active in function. This means that any node can serve an incoming request and the data is then replicated across the cluster. The applications connecting to the database automatically get routed to other nodes that are up and functioning—without any special coding by developers and without any special failover procedures by administrators. If a node experiences an outage, other nodes will continue to service read and write requests, providing system redundancy and minimizing the possibility of downtime. When the node comes back online, data is automatically synchronized with other nodes and the node is fully operational to service read and write requests again.



Spanning a DSE database across two or more data centers is very easy. Because there's no master, there are no concerns over which is primary, what needs to be replicated, and how failover happens. It's all simply built-in to the core of the database. When a region fails or a data center goes down, DSE clients are smart enough to automatically reroute requests to the parts of the database that are still running in a different region, and when the first region becomes available again, the clients will re-route the application back to the optimal performance connections. Your database is automatically set for active-active configuration with all your data being hot. And all of this happens WITHOUT ANY SPECIAL CODING needed for database operation continuity!



Let’s now go one step further and consider public cloud infrastructure. With DSE, expanding the same database into the cloud is just as easy as it is to move it to another datacenter. So, if you want to move a part of your database into the cloud, DSE replicates in the cloud and connects with all the cross-data linkages with such incredible ease that it would seem like magic. And taking it further, if you later want to leverage the efficiencies of a multi-cloud deployment, the same database can then be replicated into a different cloud with immense ease and sophistication—no more creation of data silos that come with multi cloud deployments and bring inefficiencies into the system. It’s one database across multiple clouds and the same database for administrators, developers and security teams. Every one of these nodes are fully active, delivering an **Active Everywhere database for hybrid and multi cloud** and enabling you to build and deploy their applications with modern design for today’s business requirements.


Cloud 1
Cloud 2
Azure
AWS
GCP
Every Cloud
Datacenter 2
Datacenter 1