NoSQL

Advantages and Limitations

Advantage – Elastic Scaling

For years, database administrators have relied on scale up — buying bigger servers as database load increases — rather than scale out — distributing the database across multiple hosts as load increases.

However, as transaction rates and availability requirements increase, and as databases move into the cloud or onto virtualized environments, the economic advantages of scaling out on commodity hardware become irresistible.

RDBMS might not scale out easily on commodity clusters, but the new breed of NoSQL databases are designed to expand transparently to take advantage of new nodes, and they're usually designed with low--cost commodity hardware in mind.

Advantage – Dealing with Big Data

Just as transaction rates have grown out of recognition over the last decade, the volumes of data that are being stored also have increased massively. This is sometimes called this the "industrial revolution of data."

RDBMS capacity has been growing to match these increases, but as with transaction rates, the constraints of data volumes that can be practically managed by a single RDBMS are becoming intolerable for some enterprises. Today, the volumes of "big data" that can be handled by NoSQL systems, such as Hadoop, outstrip what can be handled by the biggest RDBMS.

Advantage – Lower Maintenance Requirement

Despite the many manageability improvements claimed by RDBMS vendors over the years, high--end RDBMS systems can be maintained only with the assistance of expensive, highly trained Database Administrators(DBAs). DBAs are intimately involved in the design, installation, and ongoing tuning of high--end RDBMS systems.

NoSQL databases are generally designed from the ground up to require less management: automatic repair, data distribution, and simpler data models lead to lower administration and tuning requirements — in theory. In practice, it's likely that rumours of the DBA's death have been slightly exaggerated. Someone will always be accountable for the performance and availability of any mission--critical data store.

Advantages - Cheaper

NoSQL databases typically use clusters of cheap commodity servers to manage the exploding data and transaction volumes, while RDBMS tends to rely on expensive proprietary servers and storage systems. The result is that the cost per gigabyte or transaction/second for NoSQL can be many times less than the cost for RDBMS, allowing you to store and process more data at a much lower price point.

Advantage – Flexible Data Model

Change management is a big headache for large production RDBMS. Even minor changes to the data model of an RDBMS have to be carefully managed and may necessitate downtime or reduced service levels.

NoSQL databases have far more relaxed — or even non-existent — data model restrictions. NoSQL Key Value stores and document databases allow the application to store virtually any structure it wants in a data element. Even the more rigidly defined BigTable--based NoSQL databases (Cassandra, HBase) typically allow new columns to be created without too much fuss.

The result is that application changes and database schema changes do not have to be managed as one complicated change unit. In theory, this will allow applications to iterate faster, though,clearly, there can be undesirable side effects if the application fails to manage data integrity.

Advantage – Support Agile Development

Agile development is heavily used for application development to quickly iterate on features and continuously improve the application based on user feedback. NoSQL products can further accelerate this process by using a set of features that **make development faster**, such as:

- Flexible data schema to allow quick modifications and adjustments based on business requirements without the need for an expensive maintenance window
- The ability to represent data in application-native formats such as JSON and XML, eliminating the need for an additional layer to transform the data into a format understood by the application
- Operational tools that enable quick deployment of a NoSQL environment on a global scale, as well as to support DevOps teams through index analysis, data inspection, or backup management
- Elastic scalability that can handle the growing needs of the enterprise by leveraging commodity hardware to scale out
- Automated query optimization, among many other features, to help accelerate the time to launch new applications or modify existing ones

Limitation – Less Mature

RDBMS systems have been around for a long time. NoSQL advocates will argue that their advancing age is a sign of their obsolescence, but for most Chief Information Officers(CIOs), the maturity of the RDBMS is reassuring. For the most part, RDBMS systems are stable and richly functional. In comparison, a lot of NoSQL alternatives are in early versions with some key features yet to be implemented.

With this immaturity there can also be security issues.

Living on the technological leading edge is an exciting prospect for many developers, but enterprises should approach it with extreme caution.

Limitation – Lack of Support

Enterprises want the reassurance that if a key system fails, they will be able to get timely and competent support. All RDBMS vendors go to great lengths to provide a high level of enterprise support. In contrast, most NoSQL systems are open source projects, and although there are usually firms offering support for each NoSQL database, these companies can be smaller without the global reach, support resources, or credibility of an Oracle, Microsoft, or IBM.

Limitation – Lack of Consistency

Most NoSQL databases do not perform ACID transactions, a tried and true technique for ensuring that data remains consistent across the entire database as it is moved around. Instead, NoSQL relies on the principle of "eventual consistency." This provides some performance advantages, but it poses the risk that data on one database node may go out of sync with data on another node.

Some NoSQL implementations, like FoundationDB, try to provide the best of both worlds by doing ACID-like transactions while retaining the fluidity of NoSQL design. But in general, data consistency remains a fundamental challenge for NoSQL.

Limitation – Lack of Standardisation

NoSQL is not a specific type of database or programming interface. The design and query languages of NoSQL databases vary widely between different NoSQL products -- much more widely than they do among traditional SQL databases.

This isn't an inherent design flaw. It means, however, that the learning curve for NoSQL databases is steeper, since a programmer who knows one type of NoSQL database may not be prepared to work with a different one. For organizations keen to maximize the usefulness of staff expertise, this is a barrier to NoSQL adoption.