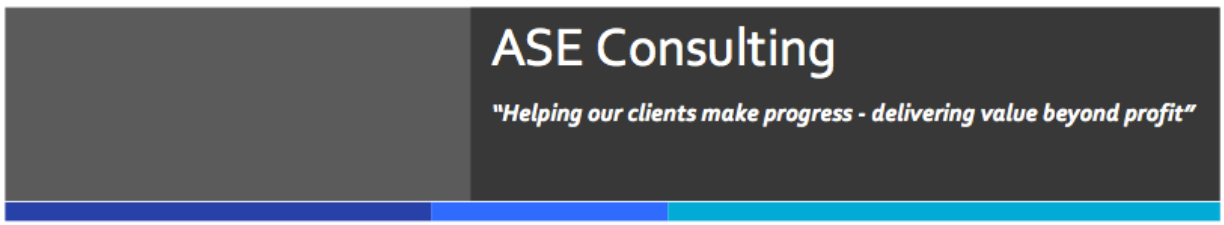
NoSQL – Is It Time To Switch?

Chris King

November 2013



Abstract

There has been an explosion in NoSQL database technologies over recent years. Much of this has been in response to organisations wanting to capitalise on their information stored within unstructured data formats; something that the traditional RDBMS model is limited in supporting. NoSQL technologies offer great benefits through low cost, easily scalable storage, fast performance and high availability. Developers no longer have to be constrained to a structured data model and as such greater agility can be achieved.

Each NoSQL solution offers its own unique selling points, but inevitably there are trade offs to be made, and so understanding the pro's and con's of each technology and how these align to an organisations drivers and architectural priorities can be a large undertaking. Inevitably there will be a shake out of the NoSQL market over the near future but the technology is fast becoming mainstream so the principles are important to IT organisations.

This paper provides an introduction to NoSQL, its key concepts, and examples of its usage. It gives an overview of some of the better-established technologies and discusses key architectural attributes for consideration when selecting such a technology. Finally, it provides practical steps to support an organisation considering embarking upon a NoSQL initiative. It assumes reasonable familiarity of database, infrastructure and application development concepts.

ASE Consulting
73a, Clifton Street
Lytham, Lancashire, FY8 5ER, United Kingdom
e-mail: customerservices@ase.co.uk
web: http://www.ase.co.uk

Background

Traditional relational database management systems (RDBMS) have, since the late 80's, been the enterprise database technology of choice. With a shift in data modelling from the hierarchical model to a structured relational format coupled with improved processing and storage capabilities, hierarchical databases were replaced with RDBMS systems. The likes of Oracle, DB2, MySQL and SqlServer evolved to become the de-facto industry standard. However, such technologies took many years to mature into their current enterprise class solutions. They have had to continually evolve in order to respond to the ever-advancing demands placed by modern application technologies.

Over recent years, organisations have begun to realise potential insight from existing untapped data sources that have not previously been considered of use, in the main due to their unstructured format and a lack of capable technology to store or query them. Whilst RDBMS technologies enable storage of unstructured content such as BLOBs (Binary Large Objects), they offer little usefulness in enabling the interrogation of key aspects of the data within such objects.

At the same time, demands for improvement are being constantly placed on RDBMS technologies in order to support advancements in application technologies and agile programming techniques. Simple RDBMS schema changes can themselves result in major programmes of rework, and as a result organisational agility is all to often stifled. The RDBMS' reliance on vertical scaling presents organisations with ever increasing storage costs and an inability to scale quickly. In summary, the limitations of RDBMS technologies are being exposed.

Enter NoSQL

In an attempt to overcome these drawbacks we have seen an explosion in data storage, search, processing and analytic technologies that attempt to address the limitations of RDBMS solutions. Many of the new developments have adopted the term NoSQL. Of importance here is the 'No' in 'NoSQL'; 'No' equates to 'Not Only', and as such NoSQL databases should not be considered useful only for unstructured data; they are equally at home with structured and semi-structured data and often support standard SQL-like languages. Hence they are able to deal with a variety of formats including JSON, XML, images, audio, video, email, social media, geospatial, structured tables, machine and web logs.

Applying the Technology

NoSQL technologies have enabled organisations to capitalise on these previously unused forms of data to gain valuable insight as well as supporting the implementation and re-architecture of operational systems. Many scenarios have benefited from the new wave of technologies, be it for supporting transactional operations, advanced business intelligence, data mining, predictive modelling or real-time production support. Some typical use case examples include:

- Publishing – to drive unconstrained content management solutions for web publishing, or social media and blogging. Semantic and graph node technologies enable this to be taken further with content linkage such as 'likes', 'people you may know' etc.
- Medical claim Fraud detection – The U.S. Centers for Medicare & Medicaid Services use NoSQL technology to perform data analysis for detecting fraudulent patterns from both medical providers and beneficiaries, e.g. to detect services not actually rendered, or patients that don't exist. Combining data from multiple geographically separated (complex relational) sources, along with external datasets such as diagnostic codes, state policy documents and even Facebook provides an effective mash-up for trend analysis and prediction alerting, and has enabled recovery of $4.2 billion in 2012.

- Geographic Social – used in Foursquare's location-based social networking app and real-time restaurant & venue recommendations.
- Gaming - Angry Birds developer Rovio uses NoSql technology for its gaming platform, to support payments, game state storage, and push notifications.
- Social – with over 400 million tweets per day, Twitter's is able to capitalise from write-intensive optimised NoSQL technology, along with easily scalable cheap horizontal storage.
- Machine generated data analytics – processing and analysing massive amounts of web and machine log data to understand capacity and demand trends and enable rapid deployment of resources to those services in greatest demand.
- Hadron Collider – the European Organization for Nuclear Research (CERN) use NoSQL technology to capture and consolidate unstructured data from distributed locations where minimal latency is critical.

Of course, it could be easy to get carried away listing radical use-cases, but it is important also to highlight the viability of NoSQL technologies for use in the majority of existing scenarios for which RDBMS technologies are currently used.

Benefits of NoSQL

Fundamental to NoSQL technologies is the notion of distributed storage; shared-nothing architectures supporting massively parallel processing (MPP), and utilizing industry standard servers, thus avoiding the need for expensive native storage systems. The main advantages to NoSQL are:

- Highly Available: through distributing and replicating the database across multiple hosts.
- Schema-less: traditional RDBMS databases require that the database schema be well defined at the start of the development. Hence any change to the underlying database structure must be cascaded through to the application, and vice versa. These restrictions are largely removed in NoSQL, and many formats of data can be stored with no prior schema definition, thus benefiting agile development.
- Application developer friendly: modern programming frameworks such as Java and Python evolved long after the introduction of the RDBMS, and their object oriented nature is not a natural fit with the relational model. NoSQL technologies effectively bring the application to the data and enables greater application agility.
- Scalable, using commodity hardware: the ability to easily scale up (and down) and deploy through the addition of industry standard servers to an existing cluster
- Cheap: avoiding expensive proprietary servers and storage systems. Hosted cloud options offer even greater economies of scale for NoSQL storage distributions.

Whilst all these benefits are attractive, it is not enough to fully write-off the RDBMS, which is likely to keep its place providing high value, low-latency workloads for small to medium volumes of structured data. The maturity, reliability, available skilled resource pool, and vendor support will continue to attract the development of RDBMS solutions but in increasingly specialised areas.

Architectural Considerations

There are a number of different categories that NoSQL generally falls under. These include:

- Key/Value store – the most simplistic data model whereby only a set of key value pairs exists. Retrieval of information (the value) is only via the key and might be represented as a blob of information. These are generally highly fault-tolerant and ideal for rapid updating of (via complete replacement of the 'value') and access to data, such as a shopping cart or user identities.

- Document store – well suited for storage of semi-structured XML and JSON data, which fits well with modern programming language data types. The need for data joins is overcome by the use of embedded documents and arrays. These are suited to content oriented applications, such as blog posts, or publishing & content/document management systems.
- Graph database - optimised for managing, traversing and querying highly connected data, such as within social networks, genetic research and online shopping recommendations. Data is stored in structured relational graphs[1] of interconnected key-value pairings, whereby the relationships between nodes can be expressed.
- Column Oriented – synonymous with Big Table storage for petabyte scale, and use with data exploration. The data is represented through its association with columns, rather than a traditional row based model. This provides for a flexible, unconstrained data model, and is beneficial for aggregation operations (through map-reduce) of one or a few columns, but over large numbers of rows. Column oriented data stores exist also in some RDBMS offerings, such as Teradata, however the differentiator lies with the underlying unconstrained data structure and the principles of shared-nothing, horizontally scalable storage & processing.

"There is a multitude of architectural considerations, and it is important to prioritise these when selecting a technology"

The majority of NoSQL technologies have been developed through the open source community, many of which have been taken further and made available as enterprise grade commercial options. Each has brought its own unique selling point, and so selecting the one most appropriate to a given use-case is not a straightforward task. In selecting an appropriate technology, some of the architectural attributes that should be considered include:

- Database Type; Key-value, Document Store, Graph, Column Orientated / Big Table
- Usage; Operational vs. Analytical (Real time decision support, ad-hoc, batch processes)
- Read vs. Write priority and dominance
- Indexing – real time, primary, secondary, full-index support
- In-memory capability
- Clustering – mechanisms and ease by which nodes can be added during live operation, and automated re-allocation of data
- Rapidly changing data vs. accumulating data
- Multi-version concurrency control support
- Replication – consideration to both DR strategy and multi-geographical location low latency mirroring. Automatic failover, rollback, point-in-time recovery.
- Search capabilities – full text, geospatial, context aware, autocomplete
- ACID transactions – (a separate discussion is dedicated to this topic later in this paper)
- Consistency support, optimistic vs. pessimistic document locking
- SQL support – SQL type querying & Sparql
- Language drivers – Java, Python, C, PHP, etc.
- REST vs. proprietary API's
- Integration support - of other data stores such as Hadoop, and business intelligence tools
- Visualisation tools – to support monitoring & management, system configuration, application builders
- Security – role based, LDAP support
- Semantic web – context support for linked open data, the use of RDF[2] triples for facts and relationships
- Tiered storage – support for dynamic re-allocation of data to different storage tiers

[1] In mathematical terms, a graph is simply a network of objects

[2] The Resource Description Framework (RDF) is a general-purpose language for representing information in the Web. See http://www.w3.org/standards/techs/rdf#w3c_all

- Enterprise level product support
- Developer community – maturity and support from both open source and enterprise NoSQL communities

NoSQL Offerings plus Hadoop

One of the biggest emerging data related technologies of late is Hadoop. NoSql and Hadoop are often discussed together, particularly around the subject of Big Data. However Hadoop is not technically a NoSQL database. Whilst being synonymous with Big Data, its focus is predominantly upon the storage and batch processing of massive amounts of data. Hadoop has become the mainstream Big Data technology, and whilst there exist others falling into the Big Data category, a whole range of NoSQL offerings have emerged alongside this, some as part of the Hadoop ecosystem. Whilst these generally apply the same architectural principles of highly scalable, distributed computing, they have been adopted as much within an operational capacity as an analytical one.

The ASE White Paper, 'Navigating the Big Data Space', provides a deep-dive into the history, use-cases, and architectural attributes of Hadoop, along with a look at competing Big Data technologies and supporting analytic tools. This paper also offers practical advice to embarking upon a Big Data initiative, much of which can be applied in testing the water for a NoSQL initiative.

There are over 150[3] known NoSQL database technologies, however we shall take a very brief look at a shortlist of some of the well know NoSQL offerings and a summary of their distinguishing attributes:

- MongoDB – JSON document store, with JavaScript interface for SQL like querying and native API's for Java and Python. Good for dynamically changing data, and popular as an operational backend as well as supporting Big Data analytics.
- CouchDB – JSON document store, supporting multiple document and eventual consistency through MVCC, plus support for disconnected working. Good where data changes occasionally.
- CouchBase – JSON document store, including a built in object level cache enabling sub-millisecond latency and sustained high throughput
- Cassandra – Big Table column-oriented key-value hybrid developed by Facebook, with no single point of failure, and with a SQL like query language (CQL). Ideal where write speed is important. An Apache Hadoop project.
- HBase – Big Table column-oriented store, modelled on Google's BigTable and written on Hadoop's HDFS storage. Ideal for running map/reduce jobs on huge data sets such as analysing log data. An Apache Hadoop project.
- Riak – Highly available and fault tolerant Key/Value store supporting most data formats, strong indexing and search capabilities and a RESTful API. Enterprise class support available to enable dual site replication.
- Redis – Disk backed in-memory Key/Value store, ideal for real-time data collection and analytics where data is rapidly changing but can be stored in memory.
- Neo4J – highly available and massively scalable graph database. Utilises an intuitive graph model for data representation, along with a powerful, human readable graph query language. REST and Java API's and ACID transactions.
- Splunk - a data access tool that collects, indexes and harnesses all the fast moving machine data generated by applications, servers and devices - physical, virtual and in the cloud. Paid license only.

[3] http://nosql-database.org/ provides a basic listing of known NoSQL technologies, a brief overview and links out to the vendors.

- MarkLogic – Enterprise Class NoSQL document store, with full text search, enterprise security, replication and elastic tiered storage support. Uses XML as preferred data format, with XQuery calls over RESTful HTTP requests. Supports semantic web querying through Sparql, plus a SQL like query interface for simplifying map/reduce complexity. Easy Integration with Hadoop and most analytic tools.
- Oracle NoSQL – not overlooking the traditional RDBMS vendors, Oracle (like many of it's traditional competitors) has entered the NoSQL race, utilising and building upon its existing Berkeley DB key/value solution. Offering flexible consistency and durability policies, at the expense of latency and performance, this has the potential for full ACID transactions.

With just a small subset of the available NoSQL solutions being introduced here, it is easy to see how selecting the most appropriate technology for an organisation's needs and architectural priorities is unlikely to be a trivial one.

ACID vs. BaSE

Adding to the complexity of choice against architectural attributes, one of the often touted concerns of the NoSQL distributed architectures is that they fail to meet the four basic ACID transactional requirements without which a database is not considered good, these being; Atomic, Consistent, Isolated and Durable. Without exploring this principle in detail, most NoSQL databases have adopted the more relaxed principle of BaSE;

- Basically Available; availability coming through the clustering of hardware. This is one of the great advantages of NoSQL, but availability often comes at the expense of consistency.
- Soft State; the atomicity, isolation and consistency of an ACID transaction needs to be handled by the application level, hence outside of the databases responsibility. Depending upon the level of consistency required, this may have large implications upon the level of robust code required within the application layer, such as the creation of queues, transaction log backups, conflict resolution rules and two-phase commits.
- Eventual Consistency – at some point in the future data will converge to a consistent state. No guarantees are made as to when this will occur. So this diverges from the ACID requirement that a transaction cannot be executed until the prior one has completed and the database has converged to a consistent state.

The focus of NoSQL generally favours the provision of availability at the expense of consistency, however many solutions come with their own means of implementing certain levels of consistency to meet the needs of its application, even if these are not perfect. Of course, the counter argument to the need for ACID transactions is, does it actually matter? For many cases the answer might be not, particularly for the purpose of batch analysis, or in supporting social media websites. However, where NoSQL databases are used in the context of financial or billing operational systems this will warrant greater importance. For example, it is critical that a bank transfer is considered complete only once both the money has been debited from one account and credited on the other - in other words the transaction is atomic. In such a case, a deeper architectural understanding of the offerings will be necessary to determine the most suitable choice. If the lack of full consistency is not acceptable, but the advantages of NoSQL are still being sought, then there exist a few solutions that address ACID requirements, such as MarkLogic and Neo4J.

Ready to make the Switch?

Switching to a NoSQL architecture is unlikely to be a straightforward task, and so prototyping such technologies within a smaller project may provide an entry point, giving developers and

"Developing a solid information architecture is important in order to understand what assets the organisations possesses, how they interact, what valuable information and insight they contain"

architects an opportunity for familiarity and up-skilling. In terms of making the switch, and determining the most suitable technology, there are some big questions that will first need to be addressed. These considerations are relevant to both existing and new projects & initiatives:

1) What are the compelling reasons – the Business Case - for making the switch? Is our RDBMS stifling our ability to be agile? Are storage costs escalating? If embarking on a new project, should we be planning for flexibility, agility and growth now in order to meet the project's future ambitions and limit future costs of change? Are we unable to take advantage of the information and insights that exists within our unstructured data assets? The key here is to consider the fundamental benefits of NoSQL: highly available, schema-less, programmer friendly, scalable and affordable. If there is no compelling argument in favour of replacing the existing, reliable RDBMS infrastructure, or of approaching a new project differently then don't do it!

2) Do we understand our data? Developing a solid information architecture is important in order to understand what assets the organisation possesses, how they interact, what valuable information and insight they contain. Ownership & accessibility, size, format, accuracy, cross-dependencies, how and at what rate they grow, governance and intellectual property policies; these are all considerations that should be thrown into the mix. Additionally, making use of linked open data sources such as DBpedia and GeoNames might bolster the usefulness of internal data when combined together (provided the chosen technology has the appropriate semantic web support).

3) Can we afford to lose consistency at the expense of high availability and cost? If so, then an open source solution might be suitable, and certainly provide an entry point into discovering what works well and what doesn't. If not, then consideration to a commercially licensed enterprise class NoSQL solution that supports full ACID transactions might be more appropriate. Whilst there exists open source offerings that claim full ACID transactional compliance, it would be advisable to tread carefully if this factor is critical to your organisation, as many vendors have their own interpretations of compliance.

4) Do we have the skills? Assuming you are new to NoSql then the immediate answer is likely to be 'No'. But if an in-house culture of continuing learning exists then encourage and empower staff to familiarising themselves with the many openly available distributions of open-source, and enterprise grade solutions.

5) What are the important architectural attributes that need to be met by a chosen technology? Beyond the ACID transaction debate, there is a multitude of architectural considerations, and it is important to prioritise these when selecting a technology. If prototyping/discovering, then It is worth considering how complex the replacement of one chosen NoSQL database with another might be, e.g. by choosing ones that supports RESTful API's there will be minimal impact to the application when switching.

6) Do we understand the costs? What is the total cost of ownership going to be? Is there enough data to benefit from cheaper distributed storage? How would we transition? What application developer up-skilling is required, and what reduction in existing DBA resource can be made? Is open source, and the potentially complex

configuration and system upkeep, going to be cheaper than a paid, professionally supported Enterprise class solution?

There will inevitably be considerations of the organisation's attitude towards risk and the need to gain senior management & executive level buy-in. Hence the outputs of any prototyping should enable some useful evidence to back up a Business Case. Whatever the case, there is a world of exciting technology waiting to be explored, providing great opportunity for cost reduction, development flexibility and, most importantly, an improved end-user customer experience.

Summary

The capabilities of NoSQL technologies is growing at an alarming rate, and with it the many promises of highly available, affordable and massively scalable storage, along with blindingly fast response times, and a more natural fit for the many programming languages currently constrained by structured data models. Each technology comes with its own unique selling points, and inevitably many of the key features come at the expense of other important fundamental architectural concerns. Much of this falls down to a state of maturity, with most offerings still in their infancy, however the rate at which the NoSQL movement grows and matures should provide comfort in any considerations to adopting such technologies.

Whether or not NoSQL is right for your organisation will fall down to a solid understanding of business objectives, key drivers, openness to change, skills capabilities, architectural priorities and an understanding of the data architecture. It may prove difficult to convince senior management to switch from the trusted and reliable to the lesser known, however testing the water through a prototyping opportunity may offer an ideal route to NoSQL discovery, highlighting the benefits to the organisation and its customers.

If all this is still too daunting, and you need further expertise, then we will be happy to work with and guide you through your NoSQL journey.

About the Author

Chris King is a Principal Consultant at ASE Consulting. The early days of his career focused upon the design, development and administration of large-scale RDBMS systems. He has since worked across multiple sectors within the domains of Enterprise Architecture, Business and Systems Analysis, ICT Strategy, Solution/Technical Architecture design and assurance. Through this period he has continued to maintain a focused interest upon the fast moving world of data related technologies and their applications in the real world.

ASE Consulting
73a, Clifton Street
Lytham, Lancashire, FY8 5ER, United Kingdom
e-mail: customerservices@ase.co.uk
web: http://www.ase.co.uk