Use cases of NoSQL

Software Languages Team
University of Koblenz-Landau
Ralf Lämmel and Andrei Varanovich

What The Heck Are You Actually Using NoSQL For?

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Bigness. big data, big numbers of users, big numbers of computers, big supply chains, big science, and so on. When something becomes so massive that it must become massively distributed.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Massive write performance. Facebook needs to store billions of messages a month. Twitter needs to store TBs of data per day. This implies key-value access, MapReduce, replication, fault tolerance, and consistency issues.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Fast key-value access. When latency is important it's hard to beat hashing on a key and reading the value directly from memory or in as little as one disk seek.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Flexible schema and **flexible datatypes**. Complex objects can be easily stored without a lot of mapping. Developers love avoiding complex schemas and ORM frameworks. Lack of structure allows for much more flexibility. We also have program and programmer friendly compatible datatypes likes JSON.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Schema migration. Schemalessness makes it easier to deal with schema migrations without so much worrying. Schemas are in a sense dynamic, because they are imposed by the application at run-time, so different parts of an application can have a different view of the schema.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Write availability. Writes need to succeed no mater what.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

No single point of failure. There is a convergence on relatively easy to configure and manage high availability with automatic load balancing and cluster sizing. A perfect cloud partner.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Generally available parallel computing. We are seeing MapReduce baked into products, which makes parallel computing something that will be a normal part of development in the future.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Programmer ease of use. The response to a database problem can't always be to hire a really knowledgeable DBA, get your schema right, denormalize a little, etc.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Use the right data model for the right problem. Isn't it better to solve a graph problem in a graph database? We are now seeing a general strategy of trying find the best fit between a problem and solution.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Avoid hitting the wall. Many projects hit some type of wall in their project. They've exhausted all options to make their system scale or perform properly. We are now seeing usable out-of-the-box products that a project can readily adopt.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Distributed systems support. What is often needed is a distributed system that can span datacenters while handling failure scenarios without a hiccup. NoSQL systems are well positioned to operate in distributed scenarios.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

General Use Cases of NoSQL

Tunable CAP tradeoffs. Are a few drops OK? Does your app need strong or weak consistency? Is availability more important or is consistency? Will being down be more costly than being wrong? It's nice to have products that give you a choice.

CAP =

Consistency: Sets of operations appear to occur at once.
+ ***Availability***: Operations terminate in intended responses.
+ ***Partition tolerance***: Operations complete despite unavailable components.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

More Specific Use Cases
(of NoSQL and SQL)

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to **push** the **technological** envelope in a direction nobody seems to be going then build it yourself because that's what it takes to be great sometimes.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

other business units to **run quick relational queries** so you don't have to reimplement everything then use a database that supports SQL.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

complex transactions because you can't afford to lose data or if you would like a simple transaction programming model then look at a Relational or Grid database.

- Example: an inventory system that might want full ACID. I was very unhappy when I bought a product and they said later they were out of stock. I did not want a compensated transaction. I wanted my item!

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

support for **secondary indexes** so you can look up data by different keys then look at relational databases and Cassandra's new secondary index support.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

creates an **ever-growing set of data** (really BigData) that rarely gets accessed then look at Bigtable Clone which will spread the data over a distributed file system.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to implement **social network operations** then you first may want a Graph database or second, a database like Riak that supports relationships. An in- memory relational database with simple SQL joins might suffice for small data sets. Redis' set and list operations could work too.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

a very **deep join depth** then use a Graph Database because they support blisteringly fast navigation between entities.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to **always** be able to **write** to a database because you need high availability then look at Bigtable Clones which feature eventual consistency.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to **span multiple data-centers** then look at Bigtable Clones and other products that offer a distributed option that can handle the long latencies and are partition tolerant.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to work on a **mobile platform** then look at CouchDB/ Mobile couchbase.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to **integrate with other services** then check if the database provides some sort of write-behind syncing feature so you can capture database changes and feed them into other systems to ensure consistency.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to scale then NoSQL or SQL can work. ***Look for*** systems that support scale-out, partitioning, live addition and removal of machines, load balancing, automatic sharding and rebalancing, and fault tolerance.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

fault tolerance, ***check how*** durable writes are in the face power failures, partitions, and other failure scenarios.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to handle lots of **small continuous reads and writes**, that may be volatile, then look at Document or Key-value or databases offering fast ***in-memory*** access. Also consider ***SSD***.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to build **CRUD** apps then look at a Document database, they make it easy to access complex data without joins.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

programmer friendliness in the form of programmer friendly data types like JSON, HTTP, REST, Javascript then first look at Document databases and then Key-value Databases.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to operate over a **wide variety** of **access patterns** and **data types** then look at a Document database, they generally are flexible and perform well.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to **dynamically** build **relationships** between objects that have **dynamic properties** then consider a Graph Database because often they will not require a schema and models can be built incrementally through programming.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

an **easier upgrade path** then use a fluid schema system like a Document Database or a Key-value Database because it supports optional fields, adding fields, and field deletions without the need to build an entire schema migration framework.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

powerful **offline reporting** with **large datasets** then look at Hadoop first and second, products that support MapReduce. Supporting MapReduce isn't the same as being good at it.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

built-in **search** then look at Riak.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to operate on **data structures** like lists, sets, queues, publish-subscribe then look at Redis. Useful for distributed locking, capped logs, and a lot more.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to be as **simple as possible** to operate then look for a hosted or PaaS solution because they will do all the work for you.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

If Your Application Needs...

to move **behavior close to the data** so the data doesn't have to be moved over the network then look at stored procedures of one kind or another. These can be found in Relational, Grid, Document, and even Key-value databases.

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

etc.

Poor Use Cases

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

Poor use cases

- OLTP
- Data integrity
- Data independence
- SQL
- Ad-hoc queries
- Complex relationships
- Maturity and stability

[http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html]

Resources

- 35+ Use Cases For Choosing Your Next NoSQL Database:

  http://highscalability.com/blog/2011/6/20/35-use-cases-for-choosing-your-next-nosql-database.html

- NoSQL in the Enterprise

  http://www.odbms.org/download/WP-DataStax-NoSQL.pdf