Nathan Hurst's Blog

Thoughts on Software, Technology, and Startups

« Back to blog

Nathan Hurst

I'm on the technical side of entrepreneurship in NYC. I love programming, board games, and my wife. I lead engineering at the world's largest educational marketplace, Teachers Pay Teachers.

Find out more about me at nahurst.com.

Posted 7 years ago
March 15, 2010 at 4:17 AM

Tags

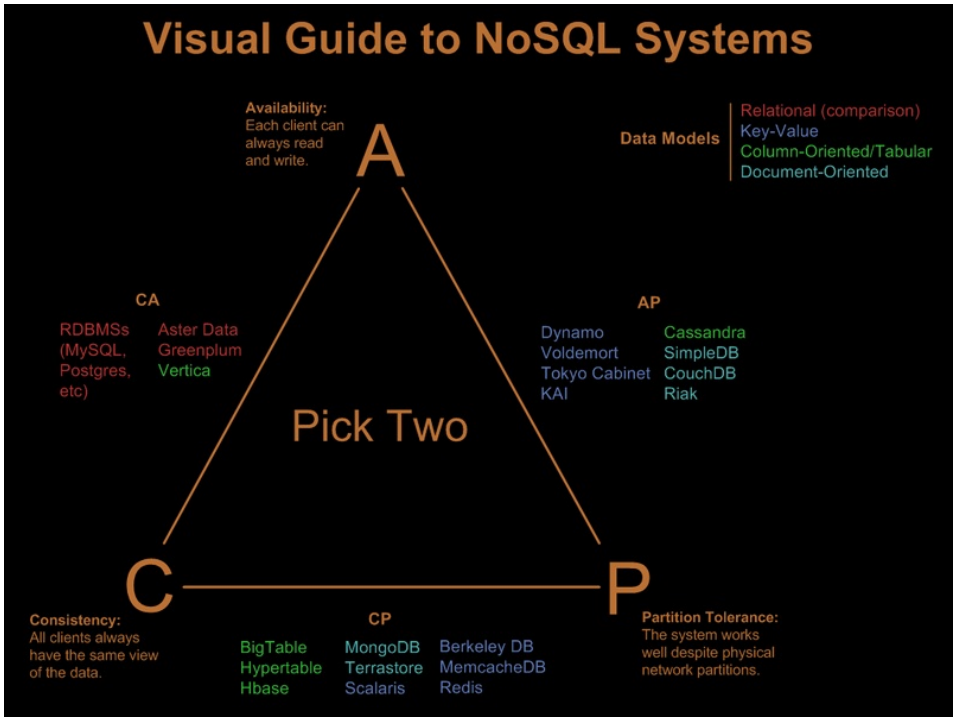
visual
cap
data
comparison
nosql
database
guide

Visual Guide to NoSQL Systems

There are so many NoSQL systems these days that it's hard to get a quick overview of the major trade-offs involved when evaluating relational and non-relational systems in non-single-server environments. I've developed this visual primer with quite a lot of help (see credits at the end), and it's still a work in progress, so let me know if you see anything misplaced or missing, and I'll fix it.

Without further ado, here's what you came here for (and further explanation after the visual).

Note: RDBMSs (MySQL, Postgres, etc) are only featured here for comparison purposes. Also, some of these systems can vary their features by configuration (I use the default configuration here, but will try to delve into others later).



As you can see, there are three primary concerns you must balance when choosing a data management system: consistency, availability, and partition tolerance.

- **Consistency** means that each client always has the same view of the data.
- **Availability** means that all clients can always read and write.
- **Partition tolerance** means that the system works well across physical

network partitions.

According to the CAP Theorem, you can only pick two. So how does this all relate to NoSQL systems?

One of the primary goals of NoSQL systems is to bolster horizontal scalability. To scale horizontally, you need strong network partition tolerance which requires giving up either consistency or availability. NoSQL systems typically accomplish this by relaxing relational abilities and/or loosening transactional semantics.

In addition to CAP configurations, another significant way data management systems vary is by the data model they use: relational, key-value, column-oriented, or document-oriented (there are others, but these are the main ones).

- **Relational** systems are the databases we've been using for a while now. RDBMSs and systems that support ACIDity and joins are considered relational.
- **Key-value** systems basically support get, put, and delete operations based on a primary key.
- **Column-oriented** systems still use tables but have no joins (joins must be handled within your application). Obviously, they store data by column as opposed to traditional row-oriented databases. This makes aggregations much easier.
- **Document-oriented** systems store structured "documents" such as JSON or XML but have no joins (joins must be handled within your application). It's very easy to map data from object-oriented software to these systems.

Now for the particulars of each CAP configuration and the systems that use each configuration:

Consistent, Available (CA) Systems have trouble with partitions and typically deal with it with replication. Examples of CA systems include:

- Traditional RDBMSs like Postgres, MySQL, etc (relational)
- Vertica (column-oriented)
- Aster Data (relational)
- Greenplum (relational)

Consistent, Partition-Tolerant (CP) Systems have trouble with availability while keeping data consistent across partitioned nodes. Examples of CP systems include:

- BigTable (column-oriented/tabular)
- Hypertable (column-oriented/tabular)
- HBase (column-oriented/tabular)
- MongoDB (document-oriented)
- Terrastore (document-oriented)
- Redis (key-value)
- Scalaris (key-value)
- MemcacheDB (key-value)
- Berkeley DB (key-value)

Available, Partition-Tolerant (AP) Systems achieve "eventual consistency" through replication and verification. Examples of AP systems include:

- Dynamo (key-value)
- Voldemort (key-value)
- Tokyo Cabinet (key-value)
- KAI (key-value)
- Cassandra (column-oriented/tabular)
- CouchDB (document-oriented)
- SimpleDB (document-oriented)
- Riak (document-oriented)

Self promotion and Credits

- If you're a developer and looking for a job or if you're hiring developers and these data systems are important to you, consider coming to Hirelite: Speed Dating for the Hiring Process on Tuesday.
- This guide draws heavily from a recent Ruby meetup (by Matthew Jording and Michael Bryzek) and a recent MongoDB presentation (given by Dwight Merriman).
- Thanks to DBNess and ansonism for their help with validating system categorizations.
- Thanks to those who helped shape the post after it was written: Stan, Dwight, and others who commented here and on this Hacker News thread.

Update: Here's a print version of the Visual Guide To NoSQL Systems if you need one quickly (warning: it's not all that pretty and I may not keep it updated, but as of 3/17/2010, it's current).

93 responses

I'm new to the NoSql movement, but I'm wondering where a graph database such as neo4j fits in.

— ariejdl

I believe Neo4J locks nodes and edges until commit, so it would be a CP system with a graph data model.

— Nathan Hurst

MondoDB has consistency? How can that be true if MondoDB does "lazy writes"? http://ivoras.sharanet.org/blog/tree/2009-11-05.a-short-time-with-mongodb.html

— Stan Harris

Stan, I am on the fence about MongoDB. Dwight Merriman, CEO of 10gen (the commercial MongoDB backer), says that MongoDB is headed in the CP direction http://www.leadit.us/hands-on-tech/MongoDB-High-Performance-SQL-Free-Database

However, from what I understand, MongoDB does not currently use Paxos or 2PC to provide consistency. I'll look in to this a bit more (thanks for the link) and update as necessary.

— Nathan Hurst

For me sonesDB is the hidden champion ;) for complex data scaling. real Graph based DB, object persistence (own Graph File-system), complete query language.. maybe its the missing part for connecting objects (in an external index?).. take a look: sones.com

— Steven Bailey

Nathan, since CouchDB fully supports ACID properties, shouldn't it be under consistency? http://couchdb.apache.org/docs/overview.html

— Stan Harris

Without further _ado_

— Association for the Preservation of the English Language

Thanks APEL

— Nathan Hurst

This is a great topic that has not sufficiently been covered yet. A few comments:
- mysql isn't really a distributed system, so I don't know where it really belongs on here. anyone else have thoughts? also the answer is complex; master/slave replication in mysql, if you do reads on the slave, is eventually consistent. but normally as a single server, it's "strong consistent".
- not sure but i thought Riak is dynamo-base and thus AP
- with CouchDB, it depends. single server it's strong consistent, like almost single server systems. with master-master replication in action, it's AP.

— dmerr

Regarding MongoDB:
@stan: i think of CAP as someone orthogonal to ACID durability. They are related, but fairly separate topics: CAP is about distributed portion of the system. You could argue a system without full durability is theoretically inconsistent period, in which case, there is no where to put it on this chart. So I think best thing is to make this chart about distribution. Plus, single server durability is on the mongodb roadmap: http://blog.mongodb.org/post/381927266/what-about-durability

"CP" is right for MongoDB - it's a lot like BigTable in terms of sharding, with a few twists. Currently the metadata storage is updated with 2 phase commits (so strong consistent). Further on each shard, at a given point of time, all writes are going first to one server (although over time the server in charge can vary) -- so that's strong consistent too.

— dmerr

One more nuance. "A" is labelled as "available for reads". It may be useful to differentate between available for reads and available for writes.
For example suppose I am using MySQL and 3 data centers: 1 master and 2 slaves, 1 instance at each DC. The network partitions. I can still read the local slave's data, but I can't write as i can't reach the master. So reads are available, and writes aren't. Now, perhaps this qualifies as eventually consistent as the data i'm reading may not be the most current. It is however a consistent snapshot from the past. It's also easy to do with a conventional db; what Dynamo added was availability for writing.

I'm not sure the best way to articulate the above, but throwing up the topic for discussion.

— dmerr

@dmerr Actually ... MongoDB sharding is about as far as you can get from a BigTable architecture. :)

— LusciousPear

Hi Nathan,
thanks for including Terrastore in your overview.

I just wanted to point out that Terrastore is a CA or CP system depending on server-to-master reconnection parameters: that is, Terrastore can be configured to try to reconnect servers to master(s) for a given time window: in such a window, the system will behave as CA, rather than CP.
Anyways, the default configuration is to behave as CP (no reconnection attempts).

Hope that helps.
Cheers,

Sergio B.

— Sergio Bossa

@LusciousPear why do you say that?
it's exactly the same in how it shards -- it has "chunks" ("tablets"?) which have key ranges that are stored as metadata.

it is very different from bigtable in some other respects - such as the storage engine, and data model (JSON rather than tabular)

— dmerr

With the recently announced consistent-read features, SimpleDB actually lives in both the AP and the CP segments.

— Mitch Garnaat

@dmerr Thanks for your comments. I do need to think of a way to show the distinction between consistent reads and writes. I also need to depict how systems can choose either CP or AP based on their configuration.

— nahurst

@dmerr MySQL clusters allows for a distributed shared nothing architecture
Also would be worth mentioning that AsterData, Vertica, and Greenplum are database appliances; and are highly optimized for data distribution; all three employ a shared nothing architecture; maintaining consistency during loads.

— ryanprociuk

I am not sure I understand how ACID fits to all this. My educational guess would be no one of this system supports transactions as any financial organisation would demand.

— Roman

Moreover, Riak is better categorized as AP, with optional consistency thanks to dynamic tuning of required reads and writes.
Hope that helps.
Cheers,

Sergio B.

— Sergio Bossa

Nathan, your previous comment saying you were going to update CouchDB has disappeared! ...Now I don't mind if you delete a comment as long as you explain why, so that us kind readers who took the time to read and comment on this post will understand. Thanks!

— Stan Harris

@dmerr You are right. Distributed data stores are fundamental to the CAP theorem. But help me understand this: Because of MongoDB "lazy writes" a client could be informed of a successful write, but then later that write fails. How does MongoDB recover from this and become consistent. "Eventual consistency" still means consistency at some point. The link I included in my previous comment about MongoDB does make a good point regarding this. And I'm not trying to bash MongoDB at all. I want very much to like it. But "lazy writes" really concern me.

— Stan Harris

Stan, thanks for keeping me honest :-). I'm working on an update (including CouchDB), but I'm having trouble figuring out how to change images on existing Posterous posts (I just didn't want people expecting an update immediately if I can't figure it out). Let me know if you how. I'm awaiting a response from Posterous.

— Nathan Hurst

You might even add http://neo4j.org to the CA side of things. Graph Databases are hard to partition, but read scale-out with eventual consistency is there.

— peterneubauer

Riak should be on the AP side of the triangle, it is influenced by Dynamo.

— Sean Cribbs

Nathan, have you tried using Posterous web interface for editing? Just go to "manage"

then click on the post. You have to hover your mouse near the title to get the "edit" link.

— Stan Harris

@stan MongoDB is adding a feature where you can block on the write until its replicated to N slaves. In conjunction with replica sets, this guarantees that write be saved somewhere. More info here: http://blog.mongodb.org/post/381927266/what-about-durability

— ehwizard

From the comments across multiple sources and emails today, I'm making the following updates:
- text updates to clarify that this diagram is for non-single-server environments (thus I'm going to keep CouchDB on AP per @dmerr's comments)
- text updates to clarify that systems are categorized in their default configurations. In the future, I'll try to delve into multiple configurations (ex: SimpleDB, Terrastore, Riak)
- moving Riak to AP
- updating the definition of A to "each client can always read and write"
- removing the sharding reference on CA systems

— Nathan Hurst

@ehwizard That is a good direction for MongoDB. But until they get to that point it shouldn't really be listed on this chart under consistency. To say that MongoDB exhibits consistency and CouchDB does not on this chart is really misleading. At least in my opinion :)

— Stan Harris

@stan in CAP context, strong consistency means "one copy serializability" [1]. durability is different; the A and C of ACID i think relate to consistency, and the D to durability. albeit it's a messed up acronym with lots of overlap.
[1] Bernstein, Goodman. http://portal.acm.org/citation.cfm?id=806714

— dmerr

Bigtable is actually a lot more like AP than CP. Each individual Bigtable is CP, but a group of Bigtables in a replication setup is AP. Even with a complete network partition between Bigtables, clients can continue to happily read/write from their local Bigtables, with eventual consistency later.
And of course since "A" is almost always the most important thing, basically every production service at Google uses multiple replicated bigtables (and is designed for this), thus for this graphic I think it would be misleading for Bigtable to be anywhere but AP.

— Dan

@Dan very interesting. how are conflicting updates reconciled?

— dmerr

Can anyone explain me what consistency means in this chart ? I am working with RDBMS systems for more than 20 years. And consistency always meant that if you update more than one object then either all changes will be saved or none. In other words you won't get a system where object one is updated and object2 is not. This is guaranteed by transactions. And as far as i know, MongoDB does not have trasnactions spanning more than one object. So how can it be labeled consistent ?

— Vagif Verdi

IIRC, there is no concept of a conflicting update, since update isn't a Bigtable operation. You are either setting or retrieving data, so if 2 people perform a "set" operation on different Bigtables, they will be replayed in some consistent order for all the Bigtables, and thus the latest operation will win. Of course since timestamp syncing is a major problem, "latest" doesn't necessarily guarantee it will be the operation that actually happened latest in the real world, but it normally does. What is guaranteed is that after all is said and done, one of the operations will win across all the Bigtables.... IE eventual consistency.
So basically in a worst case scenario where 2 different "set" operations occur on the same row at the exact same time, one of them will essentially randomly win and that will be reflected in all of the Bigtables after the operation is replicated.

— Dan

I think AP Systems are quite appealing since they provide great scalability. I have little experience with those so my question is which databases among these forces the updates of replication to be atomic?

— huangjs

no no no ... Your diagram is wrong. Almost no one chooses CP - who wants a database that is unavailable? BigTable is CA, and so are most of the items listed under CP. BigTable is highly consistent and highly available (machines can die at anytime) but all machines have to be in the same datacenter.

— aaroncordova

@aaroncordova
There's a kind of blurry line between availability and partition-tolerance, but Nathan is right here: many systems choose CP over CA because choosing CA would mean to potentially block the whole cluster in case of network partitions. Choosing CP instead means that the system is allowed to put partitioned nodes "out" of the cluster (making them unavailable), allowing so the remaining nodes to continue working.
For a more detailed explanation: http://pl.atyp.us/wordpress/?p=2521

Cheers,

Sergio B.

— Sergio Bossa

In response to the first reply - afaik neo4j doesn't (yet) natively support any form of distribution, so CAP is entirely irrelevant. There is a hint on the site that it supports sharding, but a brief googling session didn't turn up any details.

— Peter

Chad Walters (@chad_walters) did follow up on this article in some tweets, compilation of those tweets is in his blog post: http://chadwa.wordpress.com/2010/03/16/twitter-conversation-about-cap/

My follow-up comment:

I believe the notion of Fault-Tolerance lies between both the availability and partition tolerance. Best systems that show case them are definitely AP systems, which are actually distributed and decentralized systems providing us an eventual consistency.

You are definitely write that diagram does imply somewhat that CA, systems are more available vs CP, but they do so relying on a fact that the high end systems used to serve them without cost of partitioning makes system more available vs. a grid of low-end servers aka a distributed way.

Summarizing, I second Bradford (@LusciousPear), that consistency is your the main choice, while partitioning and availability are both knobs contributing towards different level/ kind of faults tolerance. Best to treat them like knobs, and NoSQL systems like Dynamo, Cassandra and Riak, allow us to do that. Depending on your context of application and requirement of service you can tune those knobs and have best of all three.

— Ali Sohani

Thanks for the link Ali. That's a great exchange. I really appreciate @LusciousPear's point "CAP is more like knobs to turn, not blocks to build from." I'll try to think of a way to incorporate that notion.

— Nathan Hurst

Dean, from a quick glance, Ingres VectorWise looks like a column-oriented CA system (same location and color as Vertica).

— Nathan Hurst

Re: key-value systems... there is a Python implementation called y_serial: http://yserial.sourceforge.net/ -- serialization + persistance :: in a few lines of code, compress and annotate Python objects into SQLite; then later retrieve them chronologically by keywords without any SQL. Most useful "standard" module for a database to store schema-less data.

Probably most suited for single server environment, but it does not require a server

daemon. Dead simple to use for Python apps.

— code43

Nathan, do you offer printer-friendly versions of your blog pages? The print is completely messed up.

— Alok

Alok, thanks for letting me know. I posted a pdf at the bottom of the post.

— Nathan Hurst

Hehe! :-) The PDF does not have the visuals or the comments! While the website being not print-friendly is most likely not your fault, I think people should pressurize the CMS software developers to keep these little things in mind when writing their software! I wonder why they do this to begin with -- If the page looks right in the browser then it should also print right without any extra effort from the CMS software developer. The state however is so messed up that stuff that should be working without any effort from them also does not work!

— Alok

I agree that the relational model and the key-value model are different data models. However, I cannot have any clear image about your definition of the "column-oriented" data model. "they store data by column as opposed to traditional row-oriented databases" is just an approach to implement a "physical" data storage. "Logical" data model built on the top of a data storage is a completely different story.
For instance, Sybase IQ also uses such a "vertically fragmented" data storage, but it is also an RDBMS that supports SQL. More interesting example is MonetDB. The core of the MonetDB server is just a vertically fragmented data storage. However, on the top of this data storage MonetDB developers have developed a RDBMS, a native XMLDB, an RDF storage and a spatial DB ! How do you categorize this polymorphic DBMS?

I think it is better to clearly distinguish between logical data models and physical data models. Otherwise you should introduce a formal definition of the column-oriented "logical" data model.

— sugibuchi

I propose you the missing Wakanda which may be considered as an object-oriented data store from its SSJS API and maybe also as a Document-store one when looking at its REST API (some people we shown it compared its data store to CouchDB). Here some talks about it:
- http://www.wakandasoftware.com/blog/nosql-but-so-much-more/
- http://www.slideshare.net/alexandre_morgaut/wakanda-js-conf-eu-09-slideshare
- http://jsconfeu.blip.tv/ (third video)

— amorgaut

What about a NoSQLite initiative?
I really miss the embedded side. AFAIK, only BerkeleyDB (both Java and C editions), Tokyo/Kyoto Cabinet, Neo4J fill this gap for different requirements. Of course, I know about good'ol GDBM and the various fliesystem-based serialization mechanisms dynamic languages provide, but some points such as replication, speed and safety are always nice to have.

— Nando Sola

...Not to mention that a JSON document-oriented embedded database à la CouchDB would be awesome.

— Nando Sola

Nice Post Nathan. This is a great summary that will be very useful in helping me evaluate data store systems.

— MightyByte

@MightyByte Thanks! Glad it was helpful.

— Nathan Hurst

This is a great discussion, thanks to all. +1 on sugibuchi's request for better

distinction between logical and physical data models. As a big fan of the relational *data model*, I wince when I see generalizations about traditional relational DB implementations that imply more than they should about the relational model itself. And I smile at the irony of the NoSQL movement's unofficial slogan ("select fun, profit from real_world where relational=false;") because it implies a relational model itself (ref. http://en.wikipedia.org/wiki/Nosql). In fact, relational purists know SQL barely qualifies as a relational query language and it is merely one of those quirks of history that SQL is the dominant query language. This consideration for distinguishing between data model and implementation applies equally to non-relational DBs, so I think it would help everyone if we could try to keep them separate in our conversations. I personally would love to see the relational data model creatively applied to other parts of the CAP universe. And I would love for critics of the relational model to familiarize themselves with the old 1970s debates about navigational and relational data models, so that we avoid rehashing old arguments.

Changing topics... consistency. Am I right that the C in CAP is a combination of A/C/I/D in ACID? If so that is unfortunate. In various discussions there seem to be at least three interpretations of what Consistency means. One is is about whether multiple updates (that are related to each other and thus have to all be kept consistent with each other) happen in an all-or-nothing (atomic) fashion. Another is about whether multiple clients see the same data values modulo time, geography, and partitioning. And another is whether data that is stored can violate various rules like referential constraints or value constraints. This seems to be a source of wasted energy, but I don't know the solution.

— STH

@STH - great points. The whole NoSQL movement is a conglomerations of many different principles. It would be interesting to see more relation systems balance on different sides of the CAP universe.
I don't have a good answer to your consistency question. Anyone else? I'll have to get back to you on it.

— Nathan Hurst

STH, Nathan,
consistency in distributed systems (hence in NOSQL systems) is all about the order of read/write operations as seen by the clients.
That is, in a consistent system (such as MongoDB or Terrastore), clients are guaranteed to read/write the latest version that has been previously written (or read in case it was unmodified); in eventually consistent systems (such as Cassandra or Riak) clients have no such a guarantee, so they may actually read/write stale data.

So it's pretty different from the C in ACID, which refers to data constraints, and is maybe more a mix of A and I, where atomicity and isolation must be taken in the context of a fully distributed system.

HTH,
Cheers,

Sergio B.

— sbtourist

http://danweinreb.org/blog/improving-the-pacelc-taxonomy

— shrusamira

This is a great resource. Can you *please* add a paragraph to defilne ACID (like you did for CAP). Then it's all on one page.
Thanks!

— still learning

This is a great discussion - thanks to all of you!!!
I guess this discussion makes clear that there is still a lot to clarify in categorizing or comparing NoSQL-approaches / products. I have just stated one on bigdata.de too (http://www.bigdata.de/2011/02/22/datenbank-technologien-fur-bigdata/ - follow the links to find the englisch original).

Furthermore I wonder where to locate ParStream (www.parstream.com) in your category-system. ParStream is an analytical database with a hybrid-data-store (row and/or column), using a highly compressed bitmap-index, operates MPP on distributed environments inlcuding redundancy and automatic rebalancing and import and updates of distributed data, provides interfaces for SQL, JDBC and a C++API and

offers JOINS. Currently, it does not offer full ACID-support as known from RDBMS. Anybody who can help me with the classification.
Thanks

— Michael Hummel

Very nice graph! Clear but informing, I like :-)

— Christian

great post ! and good intro to CAP theorem. thanks!

— mathieuel

I would dare to put on few points that have been "miss-visualized" in this post. I have not ready any of comments so I may be repeating some issues mentioned above. Cassandra gives you total power for consistency level. This diagram totally lies about it. HBase and Riak when quoted here also suffers with same issue as Cassandra. Tokyo cabinet should be renamed to Tokyo Tyrant. I would recommend any author to have hands on experience before writing such stuff. For what I see from your post is that only RDBMS are CA but I bet I can produce the CA effect out of some NoSQL stores (not satisfying C's in your diagram).

— Zohaib

I'm new to NoSql. But I see that joins must be done within the application. So why go for the NoSql databases if Oracle for example is optimised its database for joins? Where can I read more whats makes it so scalable and faster. What misconceptions should a SQL user like me be aware of? What mistakes should I watch out for when learning the new NoSql?

— m_raoul

If your database makes it hard to work with your data, than use another datastore. If you have tables to join, use SQL. But if these are only a few tables and you normalized your data to have more tables to index, you could denormalize your data, store it on a document-oriented DB and forget about joins.

Or do you store lists in a SQL DB? I once used a table like a fifo queue and it was a really stupid decision, because I had to rewrite the index every few minutes. Use Redis for something like that.

— Richard Metzler

Print hack: invert colors.

— steakknifesteak

Be great to see MarkLogic on this diagram.

— mustard57

You seem to be missing Aerospike http://www.aerospike.com/

— Ægir Örn

I was just researching this subject and noticed a difference between your classification of Vertica (CA) and what Vertica's blog states (CP). I know it's a delicate balance between the 3, with Partitioning typically being a guaranteed option in most NoSQL databases. Just wanted to clarify!

Availability – Vertica is willing to sacrifice availability in pursuit of consistency when failures occur.

^http://www.vertica.com/2012/09/13/a-feather-in-...

— Tom Kierzkowski

Another great example of a NoSql database is CryptonorDB (cloud - mobile database), which manages the storage of encrypted data and the key is managed only by the client.
See more information here: http://cryptonordb.com/

— Andreea

27 visitors upvoted this post.

Your Name

Email

Add Website URL »

Your Comment

Notify me by email when new comments are added

Comment