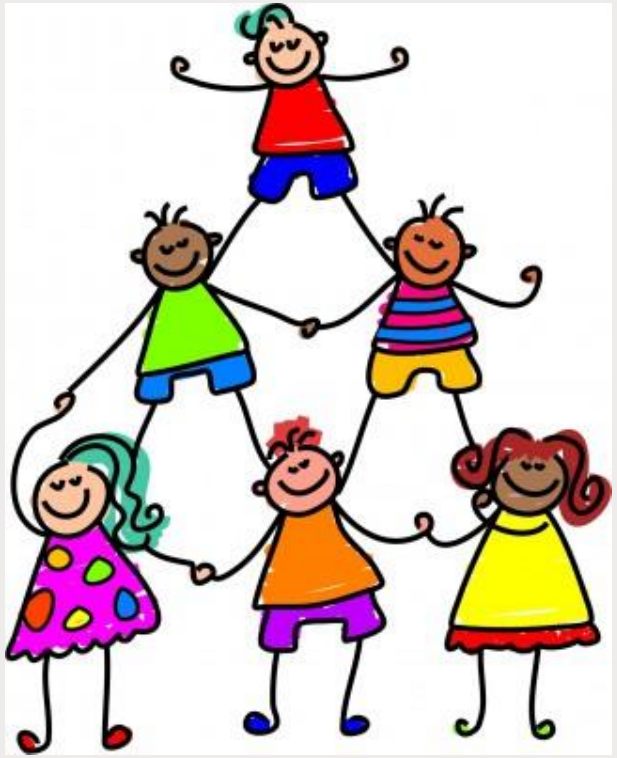
CHOOSING YOUR DATABASE

Integration considerations

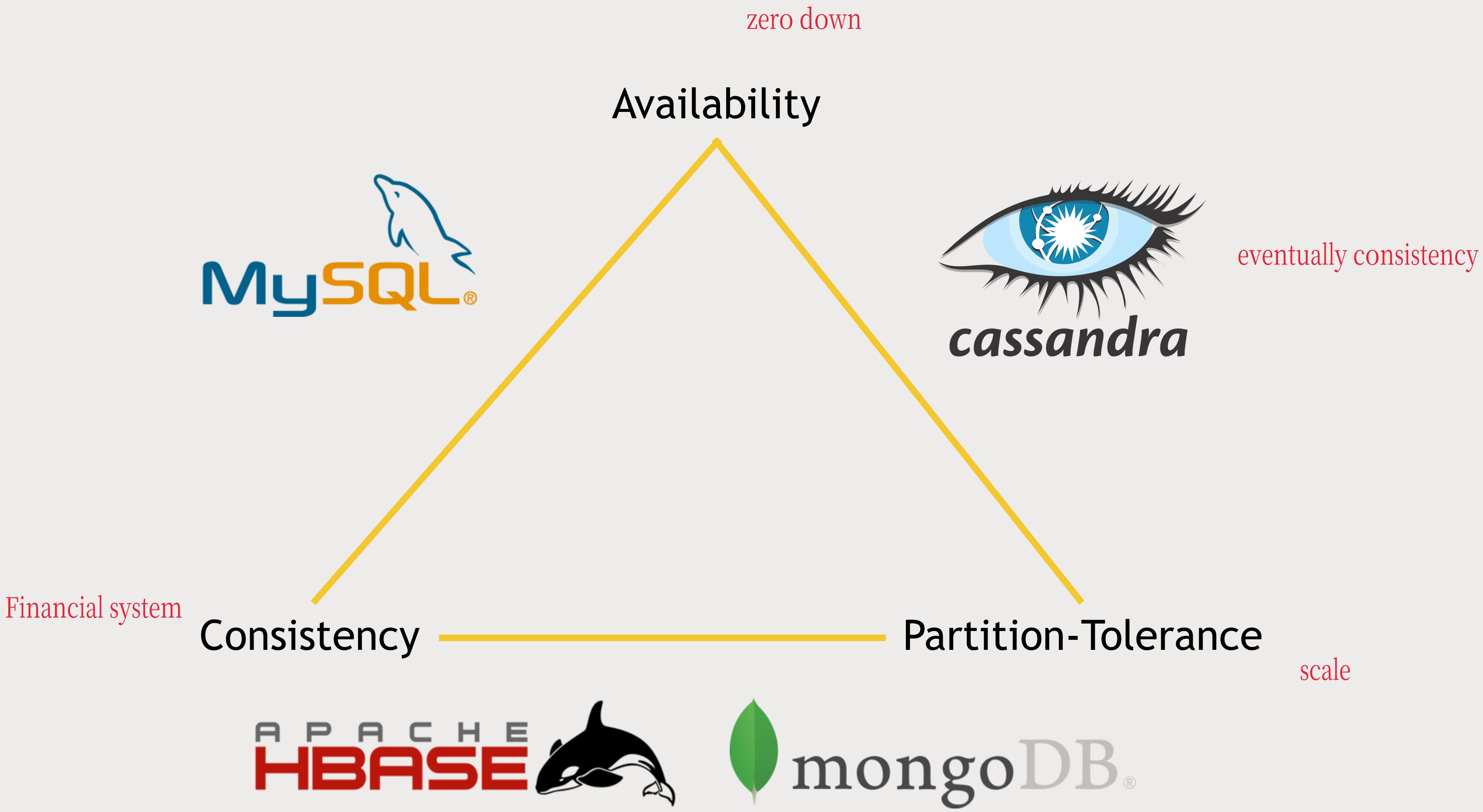
Scaling requirements

Support considerations



Budget considerations? Probably not.

CAP considerations



zero down

Availability

eventually consistency

Financial system

Consistency

Partition-Tolerance

scale

Simplicity

keep it
simple

An example

- You’re building an internal phone directory app
  - *Scale: limited* data amount no partition-tolerance required
  - *Consistency: Eventual is fine*
  - *Availability requirements: not mission critical*
  - *MySQL is probably already installed on your web server…*

MySQL®

Another example

- You want to mine web server logs for interesting patterns
- What are the most popular times of day? What’s the average session length? Etc.

Hadoop and spark are built for analysis

NoSQL is built for quickly access to a same query over and over again, thousands times/second. That means vend data to a large audience at once.
if not then just hbase/kudu is fine and with spark or impala as analyst tool.

None of
the above

Another example

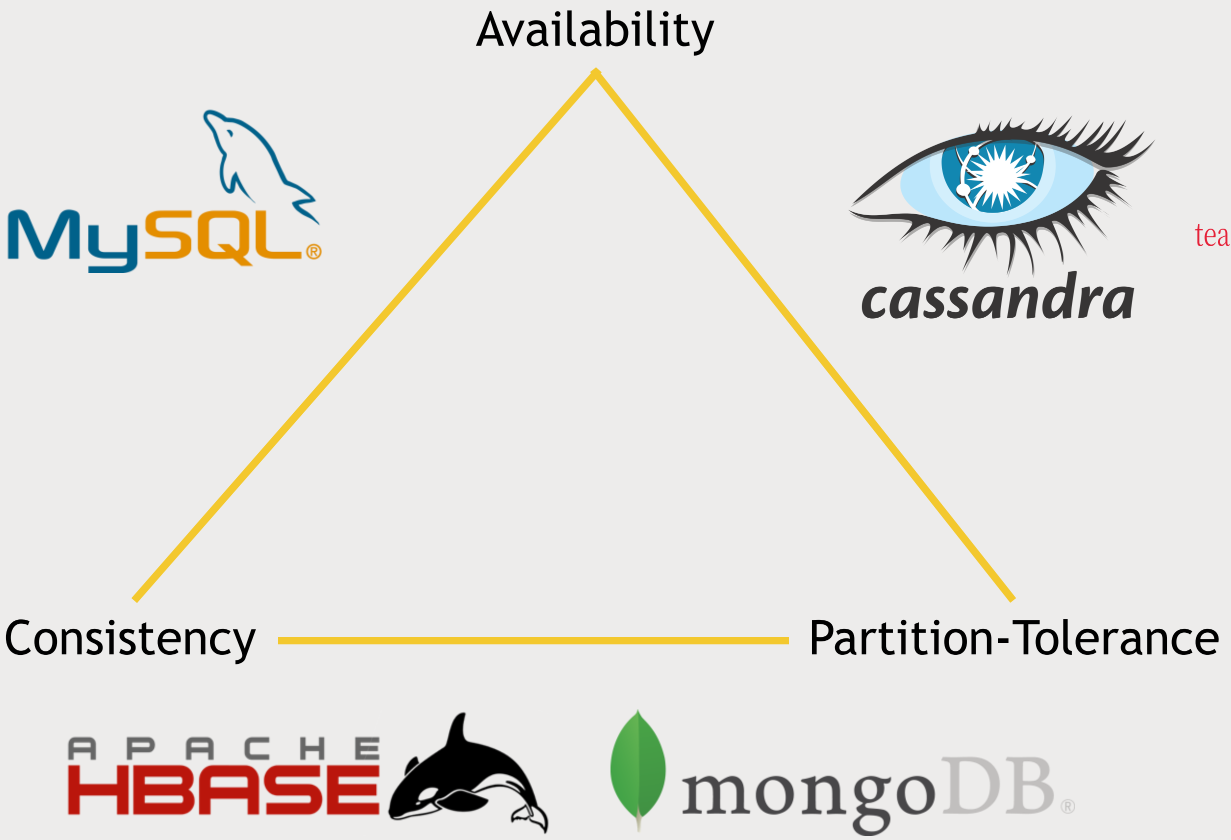
- You have a big Spark job that produces movie recommendations for end users nightly
- Something needs to vend this data to your web applications
- You work for some huge company with massive scale partition-tolerance
- Downtime is not tolerated availability
- Must be fast
- Eventual consistency OK – it’s just reads

cassandra

You try it!

- You're building a massive stock trading system scale, big data
- Consistency is more important than anything
- "Big data" is present partition-tolerance
- It's really, really important – so having access to professional support might be a good idea. And you have enough budget to pay for it.

CAP considerations



team behind as mongodb

sit on top of hdfs

enterprise, professional support behind