CAP and the Architectural Consequences

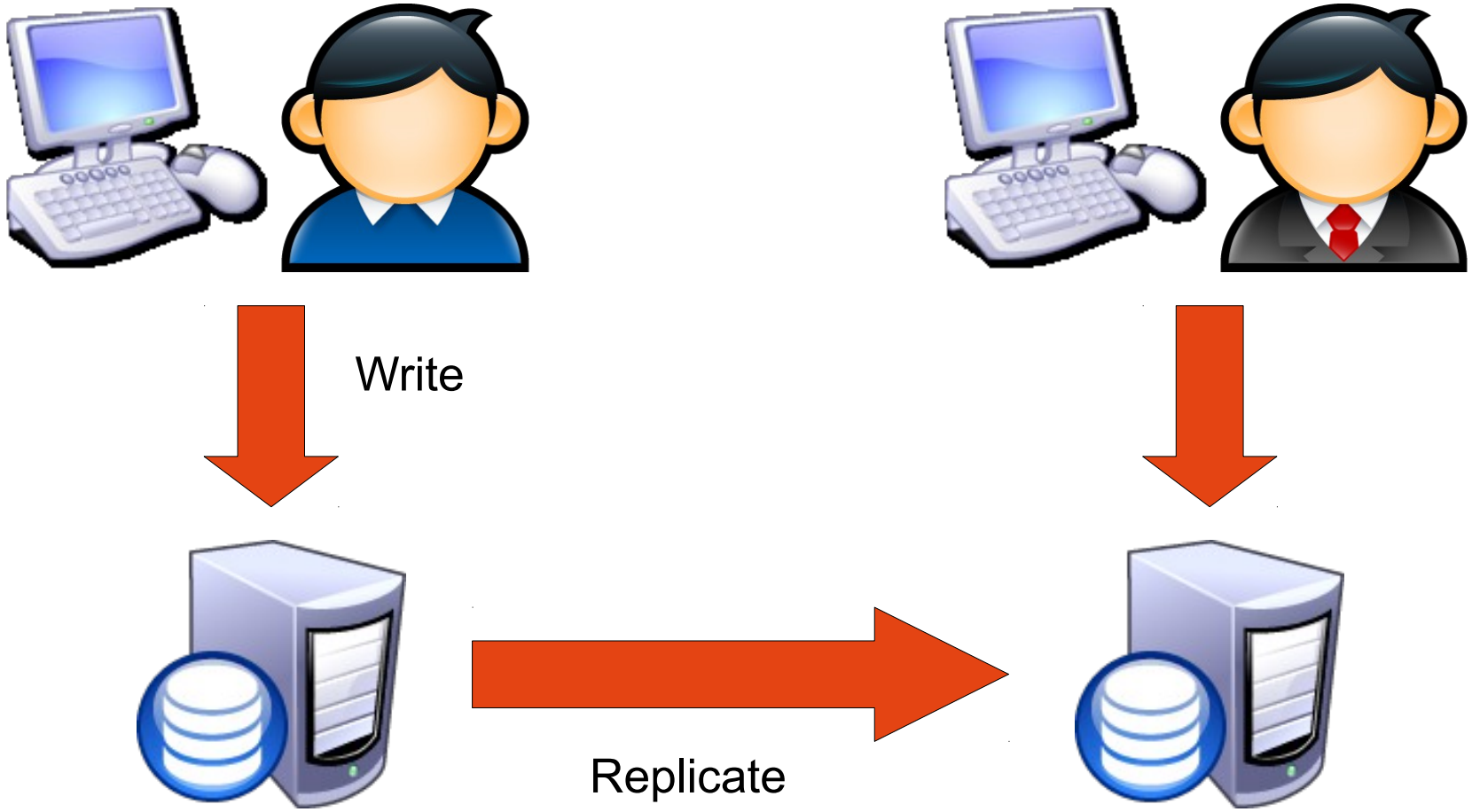
NoSQL matters
Cologne
2013-04-27

martin Schönert (triAGENS)

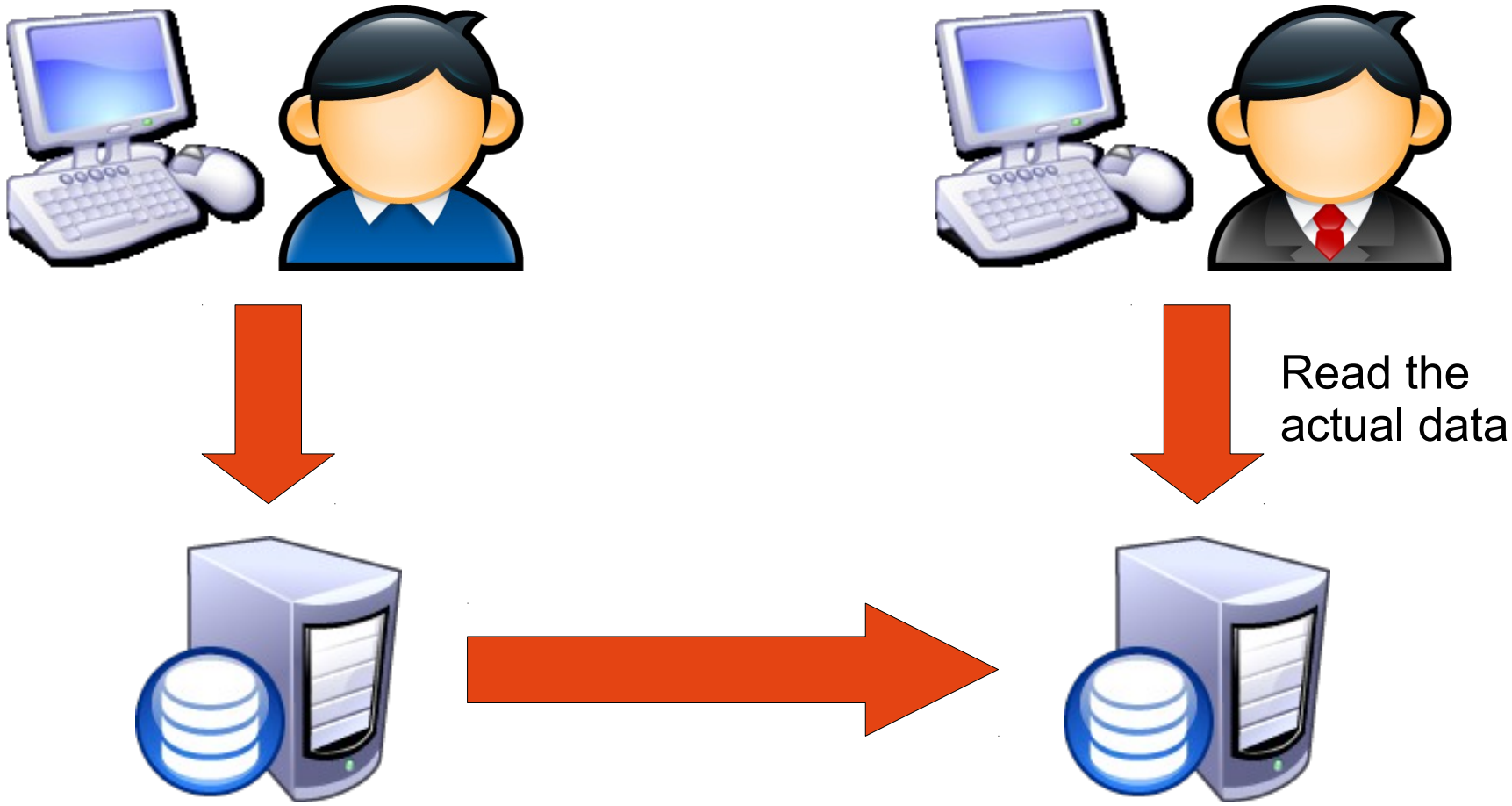
Who am I

- martin Schönert
- I work at triAGENS GmbH
- I have been in software development since 30 years
  - programmer
  - product manager
  - responsible for a data center
  - department head at a large company
  - software architect
- I am the architect of

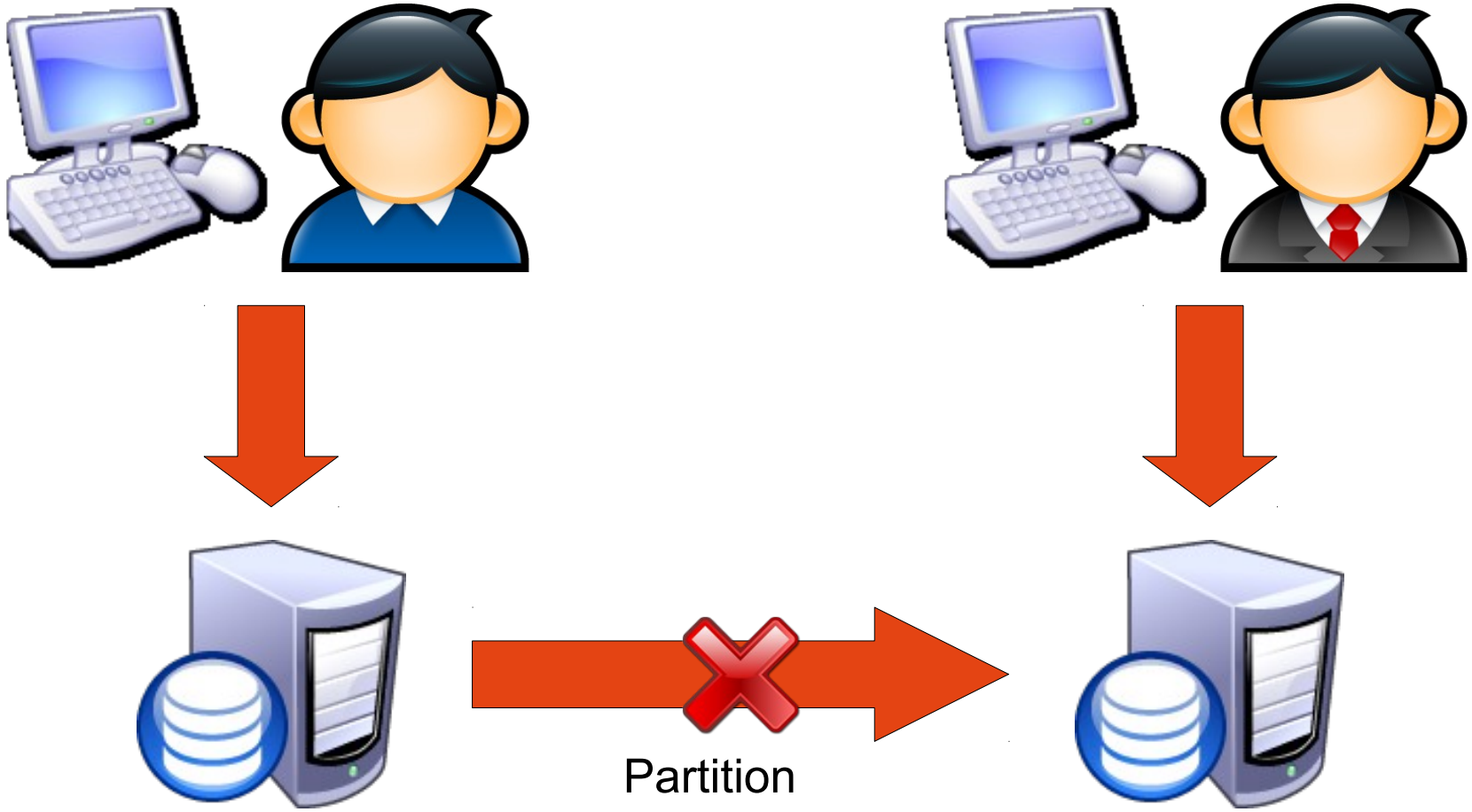
The CAP Theorem: Consistency, Availability, Partition Tolerance



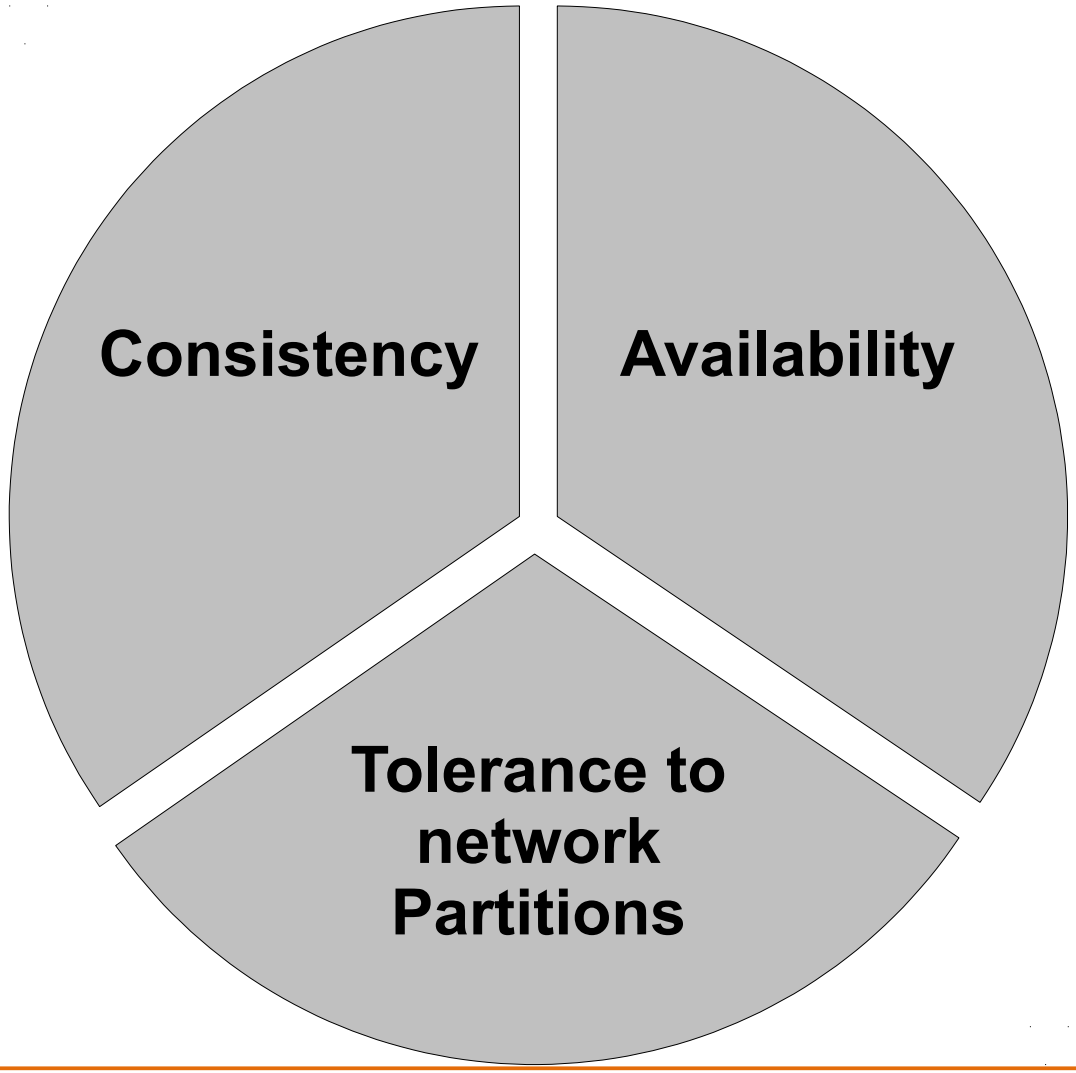
The CAP Theorem:
Consistency, Availability, Partition Tolerance



The CAP Theorem: Consistency, Availability, Partition Tolerance



Theorem: You can at most have two of these properties for any shared data system.



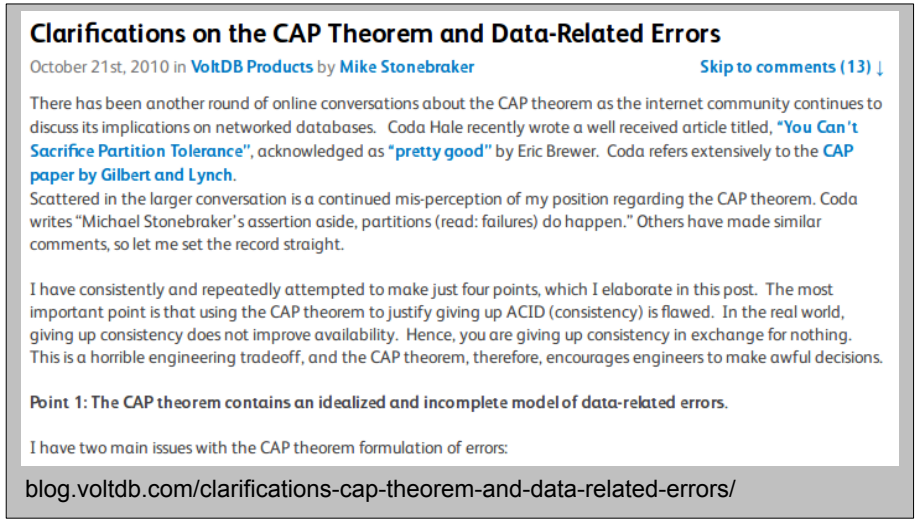
Dr. Eric A. Brewer

Towards Robust Distributed Systems

PODC Keynote, July 19. 2000

Proceedings of the Anual ACM Symposium on the Principles of Distributed Systems, 2000

Which was criticized in many articles and blog entries (below is just a small sample ;-).



Clarifications on the CAP Theorem and Data-Related Errors

October 21st, 2010 in VoltDB Products by Mike Stonebraker Skip to comments (13) ↓

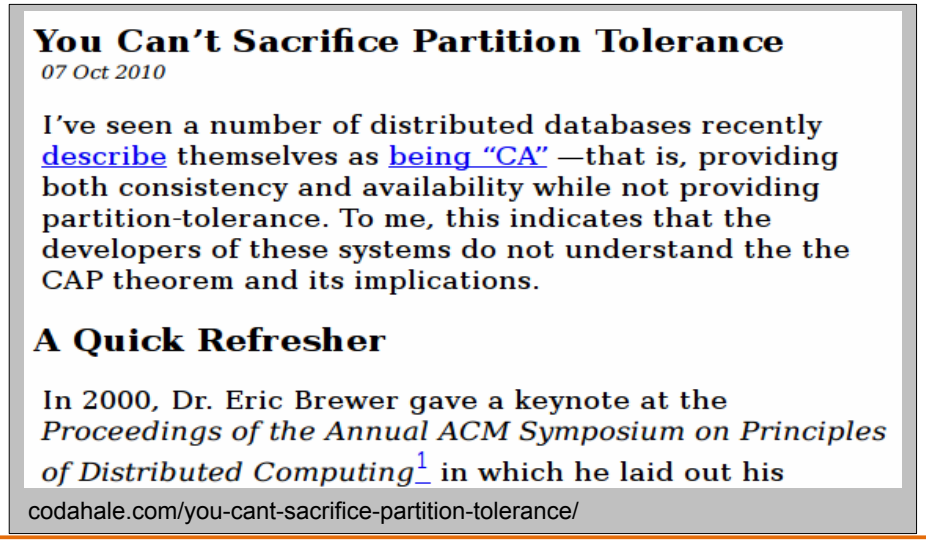
There has been another round of online conversations about the CAP theorem as the internet community continues to discuss its implications on networked databases. Coda Hale recently wrote a well received article titled, "You Can't Sacrifice Partition Tolerance", acknowledged as "pretty good" by Eric Brewer. Coda refers extensively to the CAP paper by Gilbert and Lynch.
Scattered in the larger conversation is a continued mis-perception of my position regarding the CAP theorem. Coda writes "Michael Stonebraker's assertion aside, partitions (read: failures) do happen." Others have made similar comments, so let me set the record straight.

I have consistently and repeatedly attempted to make just four points, which I elaborate in this post. The most important point is that using the CAP theorem to justify giving up ACID (consistency) is flawed. In the real world, giving up consistency does not improve availability. Hence, you are giving up consistency in exchange for nothing. This is a horrible engineering tradeoff, and the CAP theorem, therefore, encourages engineers to make awful decisions.

Point 1: The CAP theorem contains an idealized and incomplete model of data-related errors.

I have two main issues with the CAP theorem formulation of errors:

blog.voltdb.com/clarifications-cap-theorem-and-data-related-errors/



You Can't Sacrifice Partition Tolerance

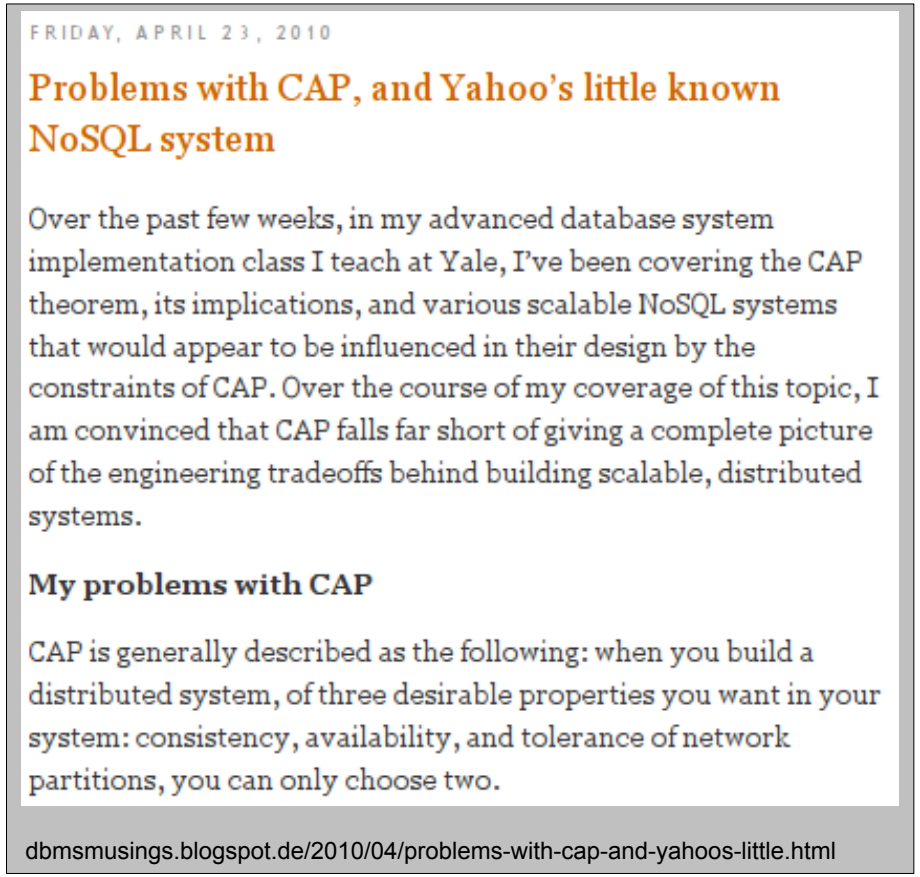
07 Oct 2010

I've seen a number of distributed databases recently describe themselves as being "CA" —that is, providing both consistency and availability while not providing partition-tolerance. To me, this indicates that the developers of these systems do not understand the the CAP theorem and its implications.

A Quick Refresher

In 2000, Dr. Eric Brewer gave a keynote at the *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*[1] in which he laid out his

codahale.com/you-cant-sacrifice-partition-tolerance/



FRIDAY, APRIL 23, 2010

Problems with CAP, and Yahoo's little known NoSQL system

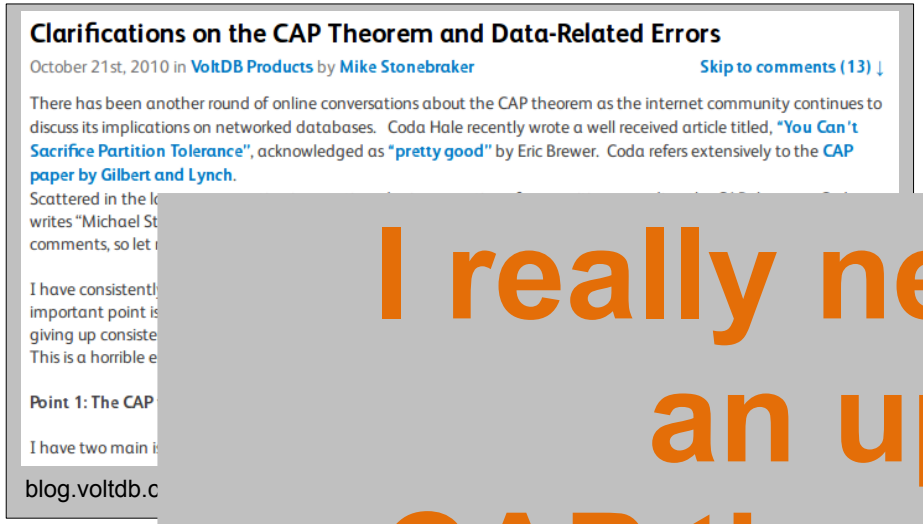
Over the past few weeks, in my advanced database system implementation class I teach at Yale, I've been covering the CAP theorem, its implications, and various scalable NoSQL systems that would appear to be influenced in their design by the constraints of CAP. Over the course of my coverage of this topic, I am convinced that CAP falls far short of giving a complete picture of the engineering tradeoffs behind building scalable, distributed systems.

My problems with CAP

CAP is generally described as the following: when you build a distributed system, of three desirable properties you want in your system: consistency, availability, and tolerance of network partitions, you can only choose two.

dbmsmusings.blogspot.de/2010/04/problems-with-cap-and-yahoos-little.html

Which was criticized in many articles and blog entries (below is just a small sample ;-).



Clarifications on the CAP Theorem and Data-Related Errors

October 21st, 2010 in VoltDB Products by Mike Stonebraker Skip to comments (13) ↓

There has been another round of online conversations about the CAP theorem as the internet community continues to discuss its implications on networked databases. Coda Hale recently wrote a well received article titled, "You Can't Sacrifice Partition Tolerance", acknowledged as "pretty good" by Eric Brewer. Coda refers extensively to the CAP paper by Gilbert and Lynch.

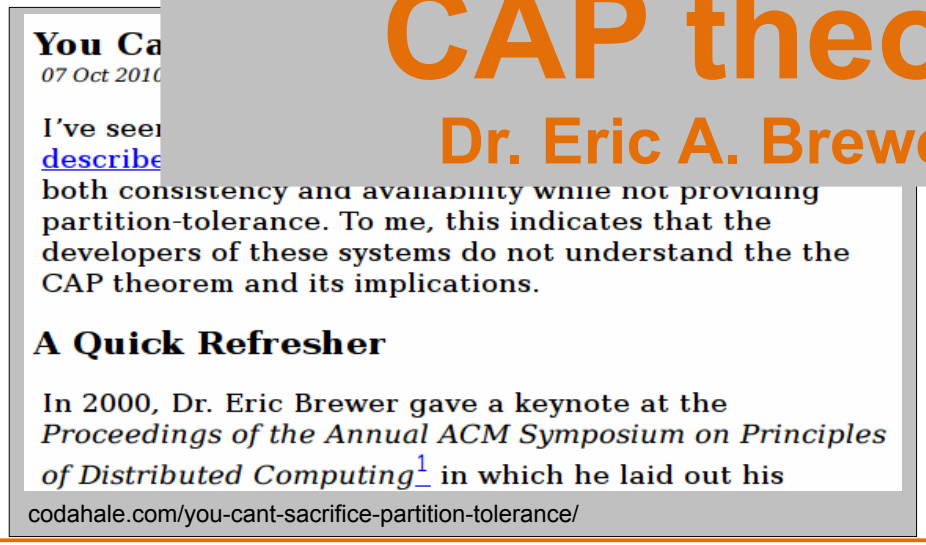
Scattered in the l
writes "Michael St
comments, so let

I have consistentl
important point is
giving up consiste
This is a horrible e

Point 1: The CAP

I have two main i

blog.voltdb.c



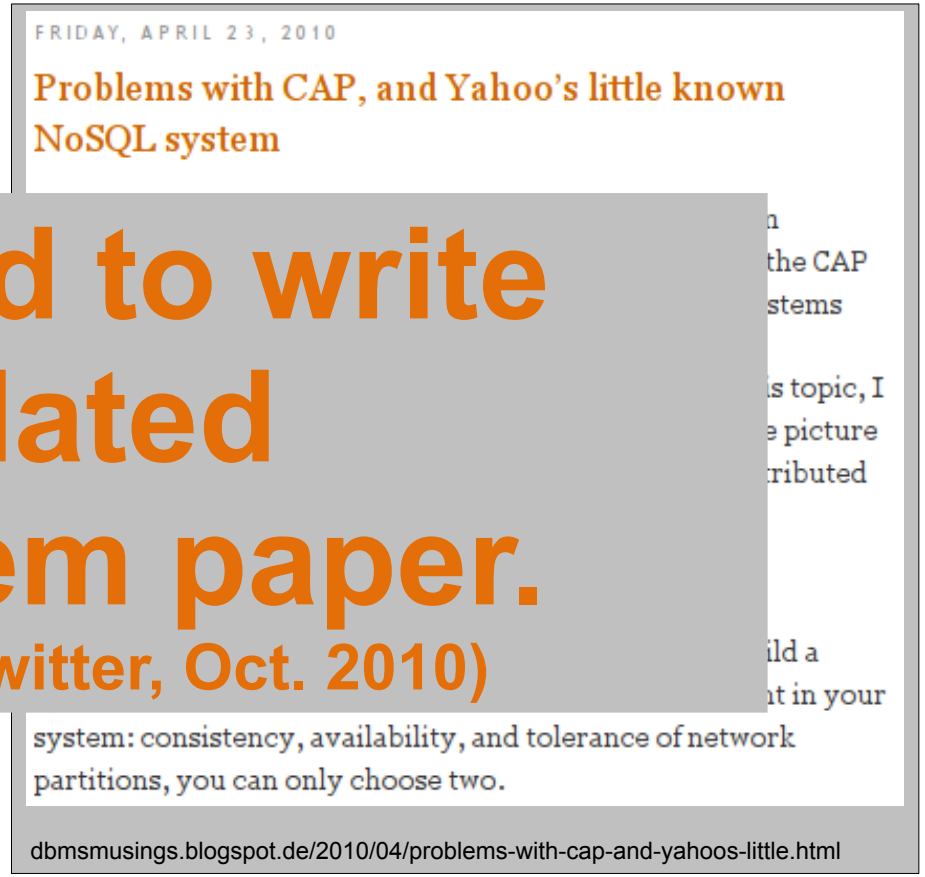
You Ca
07 Oct 2010

I've seer
describe
both consistency and availability while not providing partition-tolerance. To me, this indicates that the developers of these systems do not understand the the CAP theorem and its implications.

A Quick Refresher

In 2000, Dr. Eric Brewer gave a keynote at the *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*[1] in which he laid out his

codahale.com/you-cant-sacrifice-partition-tolerance/



FRIDAY, APRIL 23, 2010

Problems with CAP, and Yahoo's little known NoSQL system

n
the CAP
stems

s topic, I
e picture
ributed

ld a
t in your

system: consistency, availability, and tolerance of network partitions, you can only choose two.

dbmsmusings.blogspot.de/2010/04/problems-with-cap-and-yahoos-little.html

I really need to write an updated CAP theorem paper.

Dr. Eric A. Brewer (twitter, Oct. 2010)

Critique of CAP: CP

- Was basically interpreted as:
- if anything at all goes wrong (real network partition, node failure, ...), immediately stop accepting any operation (read, write, …) at all.

- and was rejected because:
- you can still accept some operations (e.g. reads),
- or continue top accept all operations in one partition (e.g. the one with a quorum),
- ...

Critque of CAP: CA

- Was basically interpreted as:
- the system gives up all of the ACID semantics and
- at no time (even while not partitioned) does the system guarantee consistency.

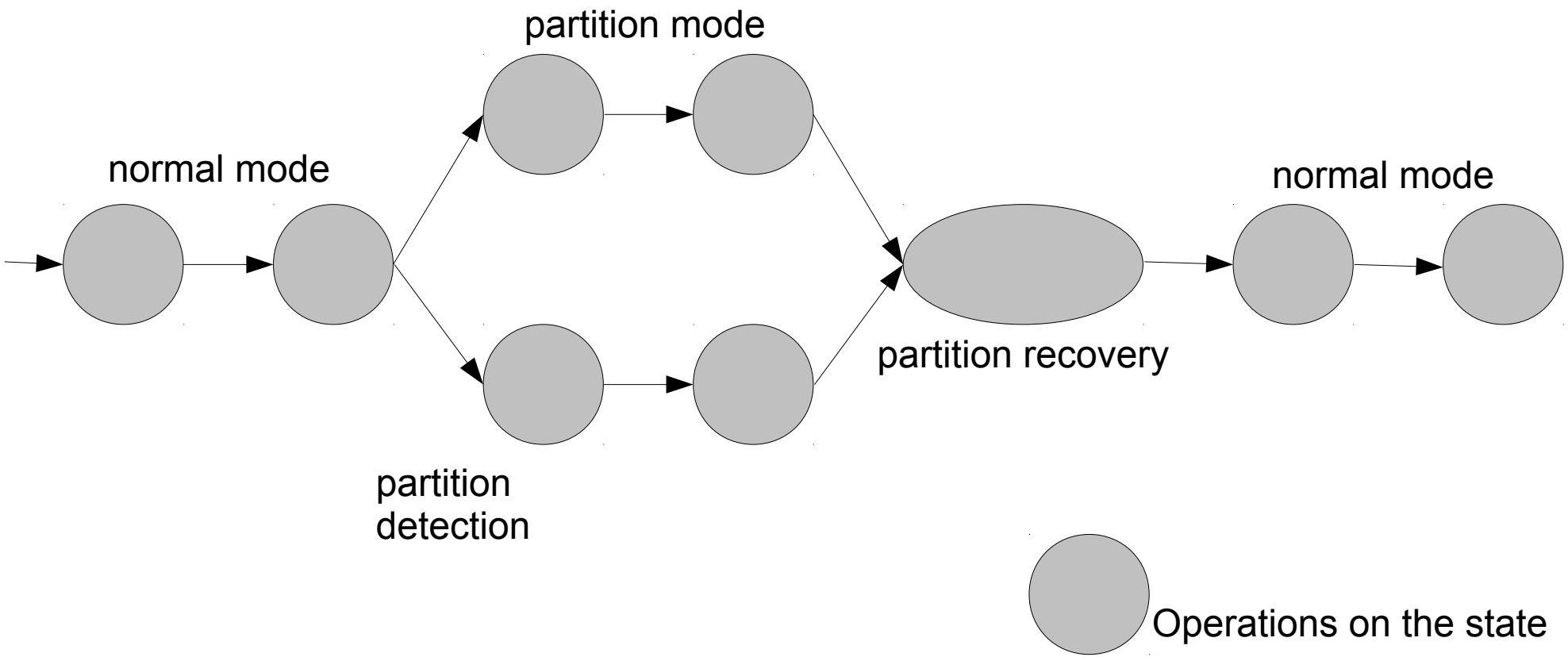
- this confusion is partly because at the same time we had discussions about:
- ACID vs. BASE and
- P(A|C) E(L|C)

Critque of CAP: CA

- Can you actually choose to not have partitions?

- Yes:
  - small clusters (2-3 nodes)
  - in one datacenter
  - nodes and clients are connected through one switch
- No:
  - not for systems with more nodes
  - or distributed over several datacenters

So let us take a better look at the situation:



Detect the partition

- Happens – at the last – when one node tries to replicate an operation to another node and this times out.
- In this moment the node must make a decision:
  - go ahead with the operation (and risk consistency)
  - cancel the operation (and reduce availability)
- Options:
  - separate watchdog (to distuingish failed node from partitions)
  - heartbeats (to avoid that only one side detects the partition)

Partition Mode
Place restrictions on:

- on the nodes that accept operations:
  - quorum
- on the data on which a client can operate:
  - data ownership (MESI, MOESI, …)
  - problems with complex operations
- on the operations
  - read only
- on the semantics:
  - delayed commit
  - async failure
  - record intent
- any combination of the above
- possibly with human intervention
  - (e.g. shut down one partition and make the other fully functional)

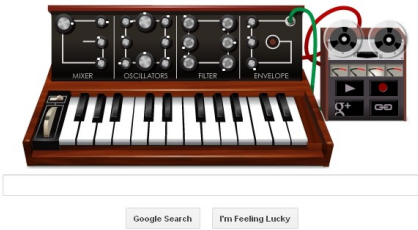
Partition Recovery

- Merging strategies
  - last writer wins
  - commutative operators
  - lattice of operations
  - application controlled
  - opportunistic (read time)
- Fix invariants
  - e.g. violation of uniqueness constraints
- Eventual consistency
  - it IS NOT the fact that every operation is first committed on one node and later (eventually) replicated to other nodes
  - it IS the fact that the system will heal itself, i.e. without external intervention converge to consistent state
- Merkle hash trees
- Hinted handoff

Massively Distributed Systems

- Store so much data that hundreds of nodes are needed just to store it.
- Not that common.
- Main driver behind early NoSQL developments.
- Receive a lot of publicity.

amazon



facebook

twitter

Consequences of CAP for massively distributed systems

- Failures happen constantly
  - Nodes die
  - Network connections die
  - Network route flapping
  - Partitions can be huge
- Must use resources well
  - if a node dies the load must distributed over multiple other nodes
- Partition detection
  - number of possible failure modes and fault lines is HUGE
  - impossible to find out the failure mode quickly is impossible
  - always operate under a worst case assumption

Consequences of CAP for massively distributed systems

- Partition mode
  - restricting operations to nodes with quorum is impossible
  - restricting operations to read only is impossible
  - restricting operation semantics is possible (though always difficult)
  - restricting operations to „own“ or „borrowed“ data is sometimes necessary
- Partition recovery
  - must happen fully automatically
  - must merge states
  - must fix invariants
- Consequences
  - no complex operations
  - resp. only „local“ complex operations

Further properties of massively distributed systems

- Properties
  - Nodes fail often
  - New nodes are added regularly
  - Nodes are not homogenous
- Distribution and redistribution of data must be fully automatic
  - Consistent Hashing
- Consequence:
- No complex operations
  - no scans over large parts of the data
  - no non-trivial joins
  - no multi-index operations
- The marvel is not that the bear dances well, but that the bear dances at all. *Russian Proverb*

My view of the (NoSQL) Database world