DISTRIBUTED COMPUTING FOR NEW BLOODS

RAYMOND TAY
I WORK IN HEWLETT-PACKARD LABS SINGAPORE
@RAYMONDTAYBL

What is a Distributed System?

What is a Distributed System?

It's been there for a long time

What is a Distributed System?

It's been there for a long time

You did not **realize** it

DISTRIBUTED SYSTEMS ARE THE NEW FRONTIER

whether YOU like it or not

DISTRIBUTED SYSTEMS ARE THE NEW FRONTIER

LAN Games

Mobile

Databases

ATMs

Social Media

whether YOU like it or not

WHAT IS THE <u>ESSENCE</u> OF DISTRIBUTED SYSTEMS?

WHAT IS THE <u>ESSENCE</u> OF DISTRIBUTED SYSTEMS?

IT IS ABOUT THE ATTEMPT TO <u>OVERCOME</u>

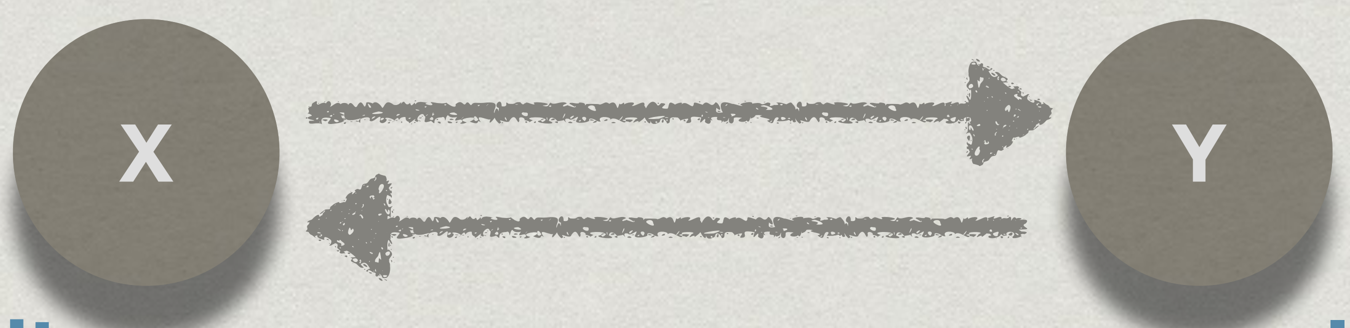
- INFORMATION TRAVEL
- WHEN INDEPENDENT PROCESSES FAIL

Information flows at speed of light !

When independent things **DONT** fail.

I’ve sent it.

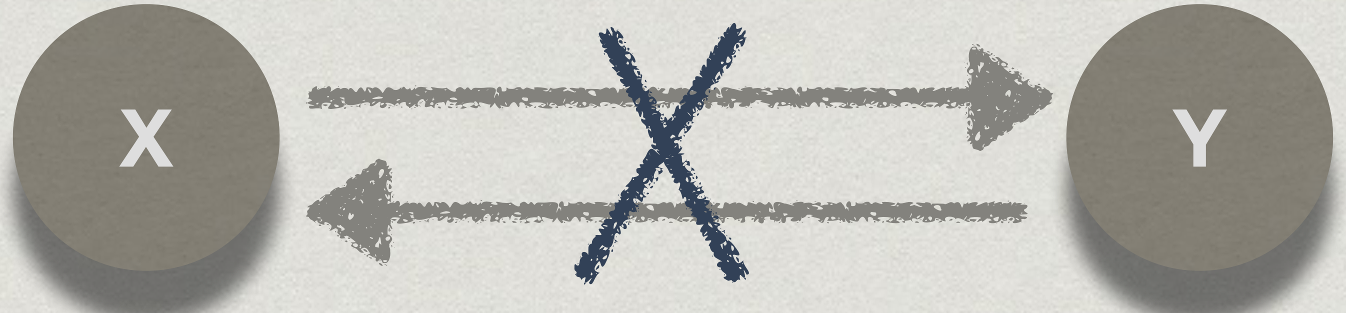
I’ve received it.



I’ve received it.

I’ve sent it.

When independent things fail, independently.



NETWORK FAILURE

When independent things fail, independently.

NODE FAILURE

There is no difference between a slow node and a dead node

Why do we NEED it?

Scalability

when the needs of the system outgrows what a single node can provide

Why do we NEED it?

Scalability

when the needs of the system outgrows what a single node can provide

Availability

Enabling resilience when a node fails

IS IT REALLY THAT HARD?

IS IT REALLY THAT HARD?

The answer lies in knowing what is <u>possible</u> and what is <u>not possible</u>…

IS IT REALLY THAT HARD?

Even widely accepted facts are challenged…

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable

Peter Deutsch and other fellows at Sun Microsystems

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable

AWS EBS Outage

On April 21, 2011, AWS suffered unavailability for 12 hours,[2] causing hundreds of high-profile Web sites to go offline. As a part of normal AWS scaling activities, Amazon engineers had shifted traffic away from a router in the EBS (Elastic Block Store) network in a single U.S. East AZ (Availability Zone), but, due to incorrect routing policies:

> ...many EBS nodes in the affected Availability Zone were completely isolated from other EBS nodes in its cluster. Unlike a normal network interruption, this change disconnected both the primary and secondary network simultaneously, leaving the affected nodes completely isolated from one another.

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable
- The network is secure

Peter Deutsch and other fellows at Sun Microsystems

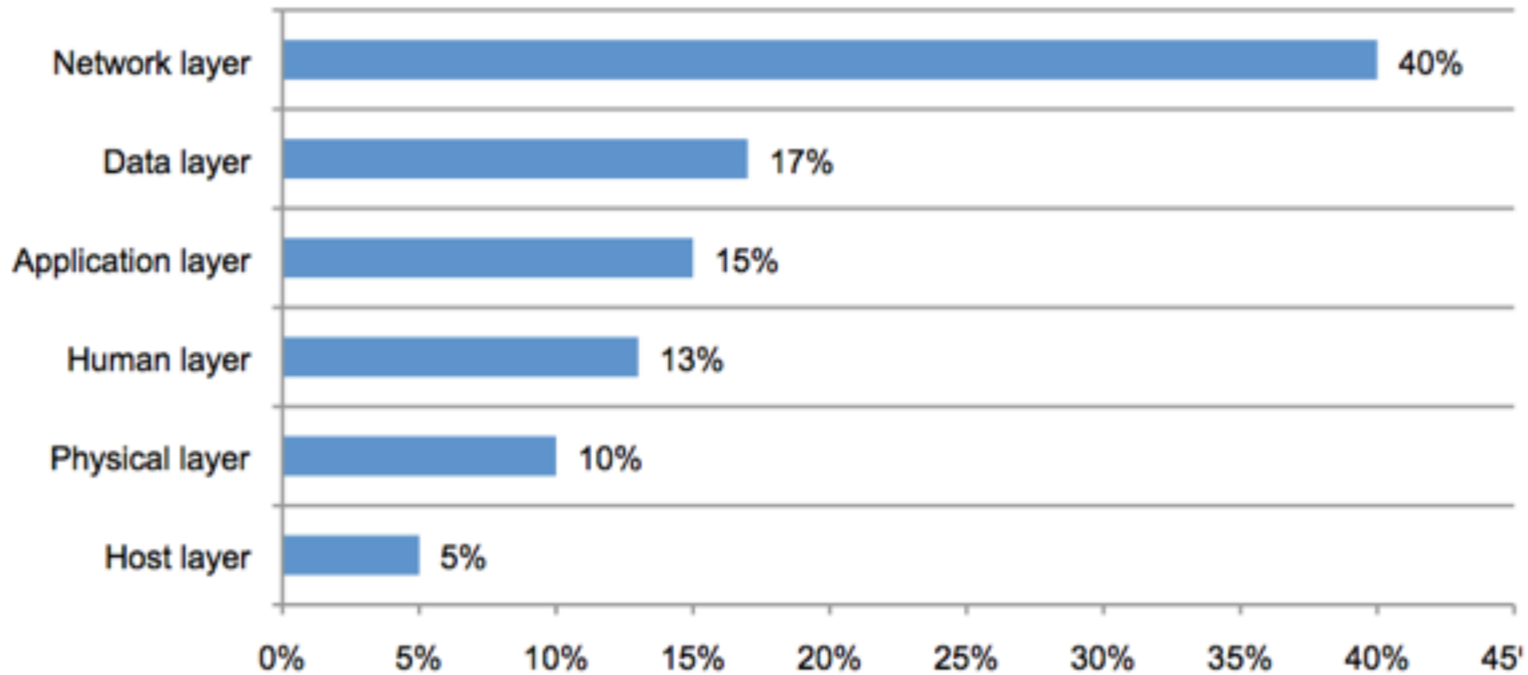
THE FALLACIES OF DISTRIBUTED COMPUTING

Ponemon INSTITUTE

Cyber security budget allocation

Figure 16 summarizes six layers in a typical multi-layered IT security infrastructure for all benchmarked companies. Each bar reflects the percentage dedicated spending according to the presented layer. The network layer receives the highest allocation at 40 percent of total dedicated IT security funding. At only five percent, the host layer receives the lowest funding level. The percentage allocations to physical layer activities is highest for critical infrastructure companies such as communications, energy and utilities and lowest for retail, hospitality and consumer product companies.

Figure 16. Budgeted or earmarked spending according to six IT security layers



2013 cost of cyber crime study: United States

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable
- The network is secure
- The network is homogeneous

Peter Deutsch and other fellows at Sun Microsystems

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable
- The network is secure
- The network is homogeneous
- Latency is zero

Peter Deutsch and other fellows at Sun Microsystems

"*But I think that it's really interesting to see that the end-to-end bandwidth increased by 1468 times within the last 11 years while the latency (the time a single ping takes) has only been improved tenfold. If this wouldn't be enough, there is even a natural cap on latency. The minimum round-trip time between two points of this earth is determined by the maximum speed of information transmission: the speed of light. At roughly 300,000 kilometers per second (3.6 * 10E12 teraangstrom per fortnight), it will always take at least 30 milliseconds to send a ping from Europe to the US and back, even if the processing would be done in real time.*"

Ingo Rammer on latency vs bandwidth

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable
- The network is secure
- The network is homogeneous
- Latency is zero
- Bandwidth is infinite

Peter Deutsch and other fellows at Sun Microsystems

THE FALLACIES OF DISTRIBUTED COMPUTING

* The network is reliable
* The network is secure
* The network is homogeneous
* Latency is zero
* Bandwidth is infinite
* Topology doesn't change

Peter Deutsch and other fellows at Sun Microsystems

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable
- The network is secure
- The network is homogeneous
- Latency is zero
- Bandwidth is infinite
- Topology doesn't change
- There is one administrator

Peter Deutsch and other fellows at Sun Microsystems

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable
- The network is secure
- The network is homogeneous
- Latency is zero
- Bandwidth is infinite
- Topology doesn’t change
- There is one administrator
- **Transport cost is zero**

Peter Deutsch and other fellows at Sun Microsystems

THE FALLACIES OF DISTRIBUTED COMPUTING

- The network is reliable
- The network is secure
- The network is homogeneous
- Latency is zero
- Bandwidth is infinite
- Topology doesn't change
- There is one administrator
- Transport cost is zero

Peter Deutsch and other fellows at Sun Microsystems

WHAT WAS TRIED IN THE PAST?

* DISTRIBUTED OBJECTS
* REMOTE PROCEDURE CALL
* DISTRIBUTED SHARED MUTABLE STATE

WHAT WAS ATTEMPTED?

- DISTRIBUTED OBJECTS
- REMOTE PROCEDURE CALL
- DISTRIBUTED SHARED MUTABLE STATE
  - Typically, programming constructs to “abstract” the fact that there are local and distributed objects
  - Ignores latencies
  - Handles failures with this attitude → “i dunno what to do…YOU (i.e. invoker) handle it”.

WHAT WAS ATTEMPTED?

* DISTRIBUTED OBJECTS
* REMOTE PROCEDURE CALL
* DISTRIBUTED SHARED MUTABLE STATE

- Assumes the synchronous processing model
- Asynchronous RPC tries to model after synchronous RPC…

WHAT WAS ATTEMPTED?

- DISTRIBUTED OBJECTS
- REMOTE PROCEDURE CALL
- **DISTRIBUTED SHARED MUTABLE STATE**

- Found in Distributed Shared Memory Systems i.e. “1” address space partitioned into “x” address spaces for “x” nodes
- e.g. JavaSpaces ⇒ Danger of 2 independent processes to commit successfully.

YOU CAN APPRECIATE THAT IT IS <u>VERY HARD TO SOLVE</u> IN ITS ENTIRETY

The question really is “<u>How</u> can we do <u>better</u>?”

The question really is "**How** can we do **better**?"

To start,we need to understand **2** results

FLP IMPOSSIBILITY RESULT

The FLP result shows that in an asynchronous model, where one processor might crash, there is no distributed algorithm to solve the consensus problem.

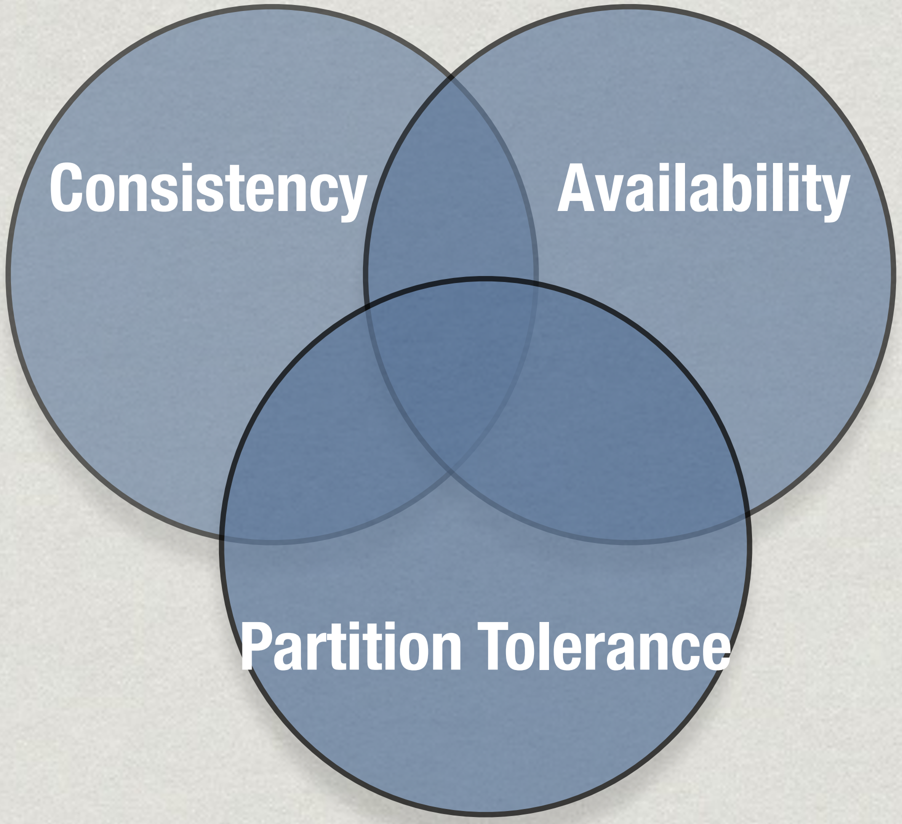
Consensus is a fundamental problem in fault-tolerant distributed systems. Consensus involves multiple servers agreeing on values. Once they reach a decision on a value, that decision is final.

Typical consensus algorithms make progress when any majority of their servers are available; for example, a cluster of 5 servers can continue to operate even if 2 servers fail. If more servers fail, they stop making progress (but will never return an incorrect result).

Paxos (1989), Raft (2013)

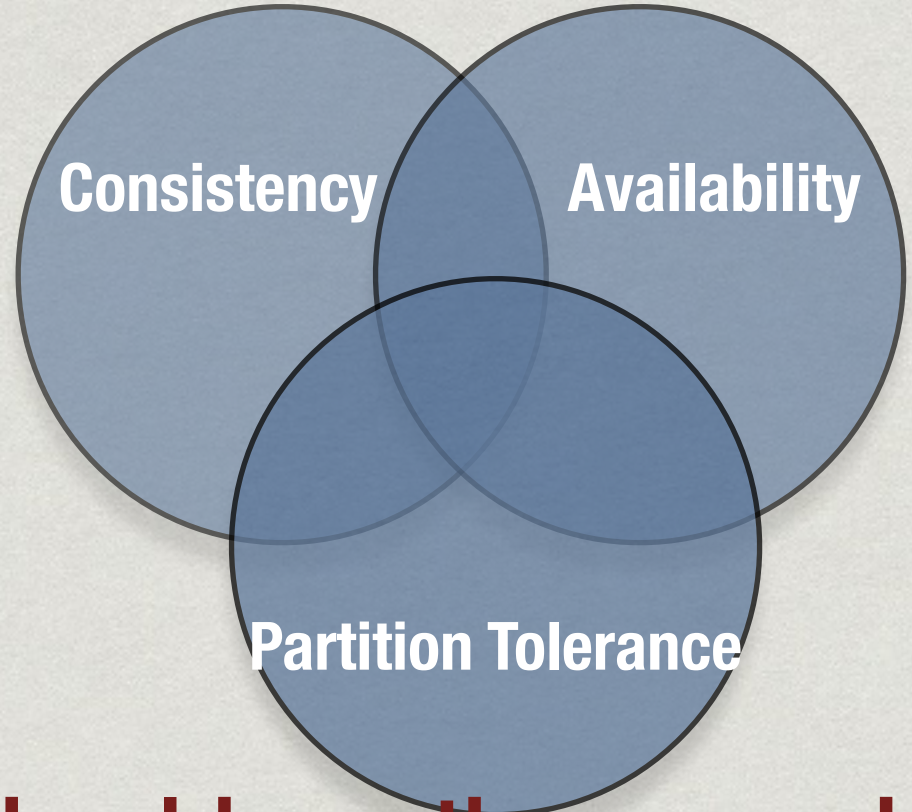
CAP THEOREM

CAP CONJECTURE (2000) established as a <u>theorem</u> in 2002



CAP THEOREM

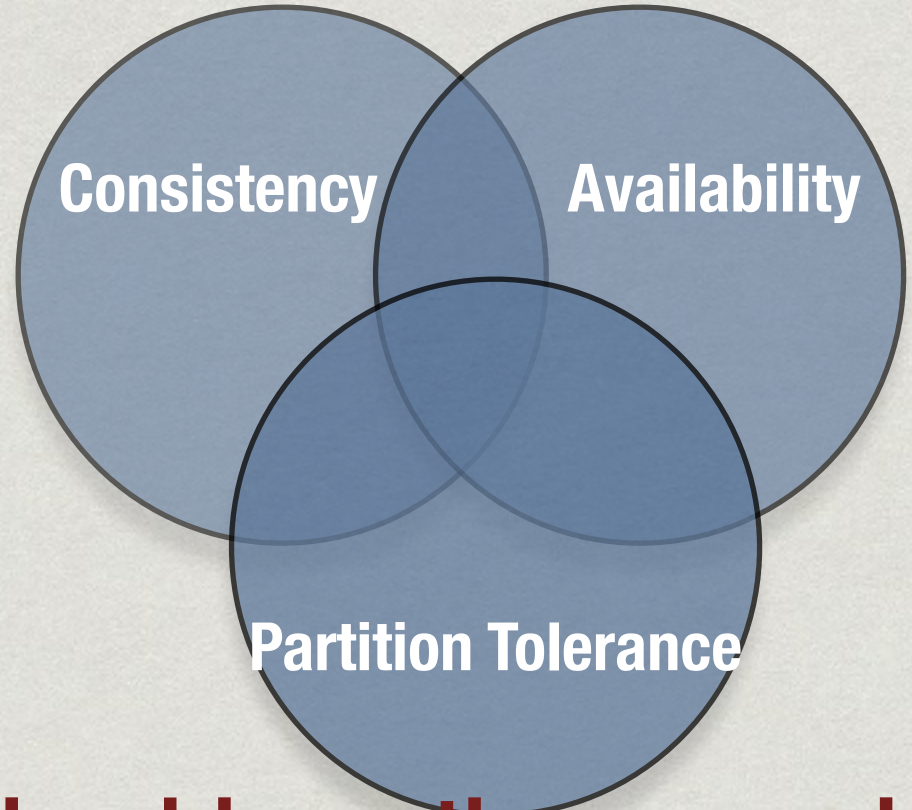
CAP CONJECTURE
(2000) established
as a <u>theorem</u> in 2002



- Consistency ⇒ all nodes should see the same data, eventually

CAP THEOREM

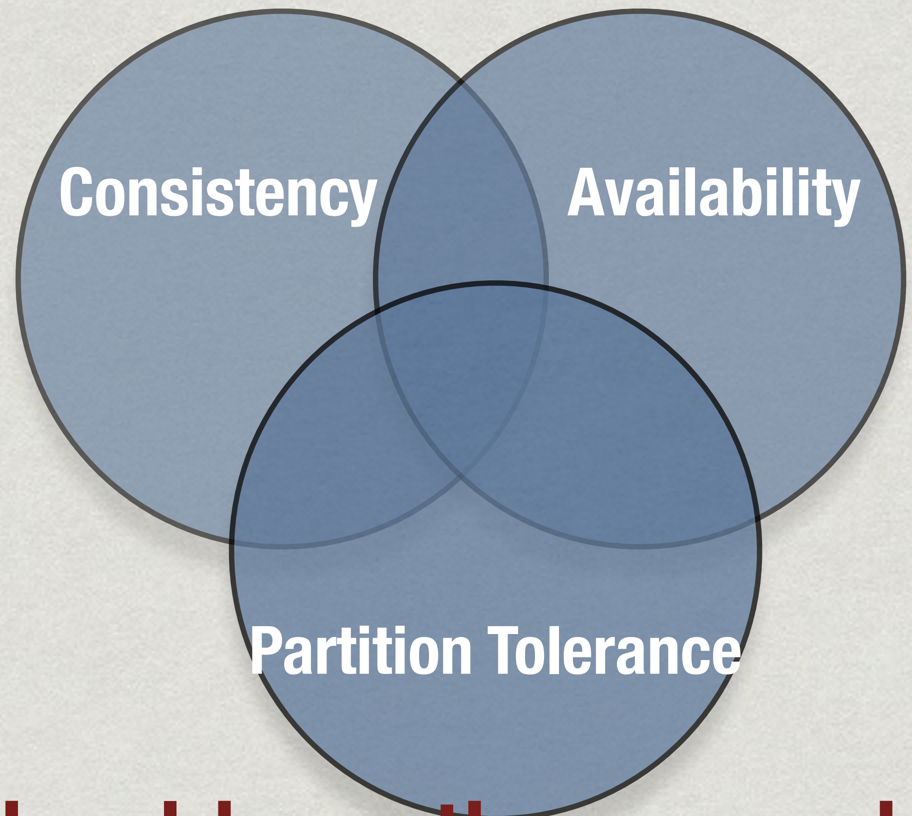
CAP CONJECTURE (2000) established as a <u>theorem</u> in 2002



- Consistency ⇒ all nodes should see the same data, eventually
- Availability ⇒ System is still working when node(s) fails

CAP THEOREM

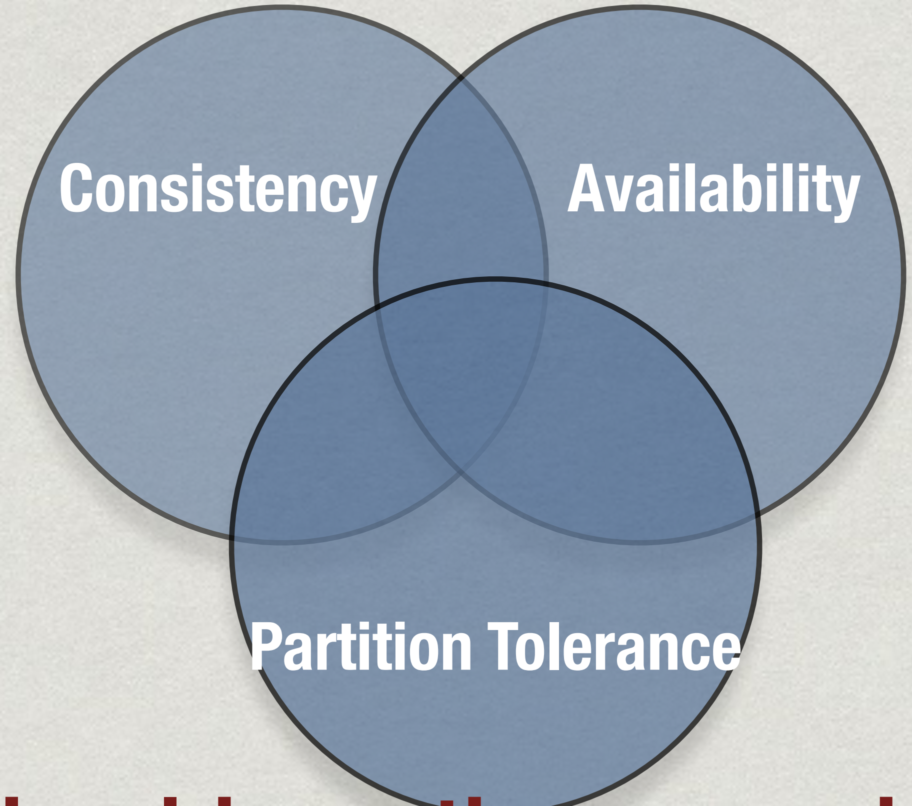
CAP CONJECTURE (2000) established as a theorem in 2002



- Consistency ⇒ all nodes should see the same data, eventually
- Availability ⇒ System is still working when node(s) fails

CAP THEOREM

CAP CONJECTURE (2000) established as a theorem in 2002



- Consistency ⇒ all nodes should see the same data, eventually
- Availability ⇒ System is still working when node(s) fails
- Partition-Tolerance ⇒ System is still working on arbitrary message loss

How does knowing FLP and CAP help me?

How does knowing FLP and CAP help me?

It's really about asking which would you sacrifice when a system fails? **Consistency** or **Availability**?

You can choose C over A

It needs to preserve reads and writes i.e. linearizability

i.e. A read will return the last completed write (on ANY replica) e.g. Two-Phase-Commit

You can choose A over C

It might return stale reads …

You can choose A over C

It might return stale reads …

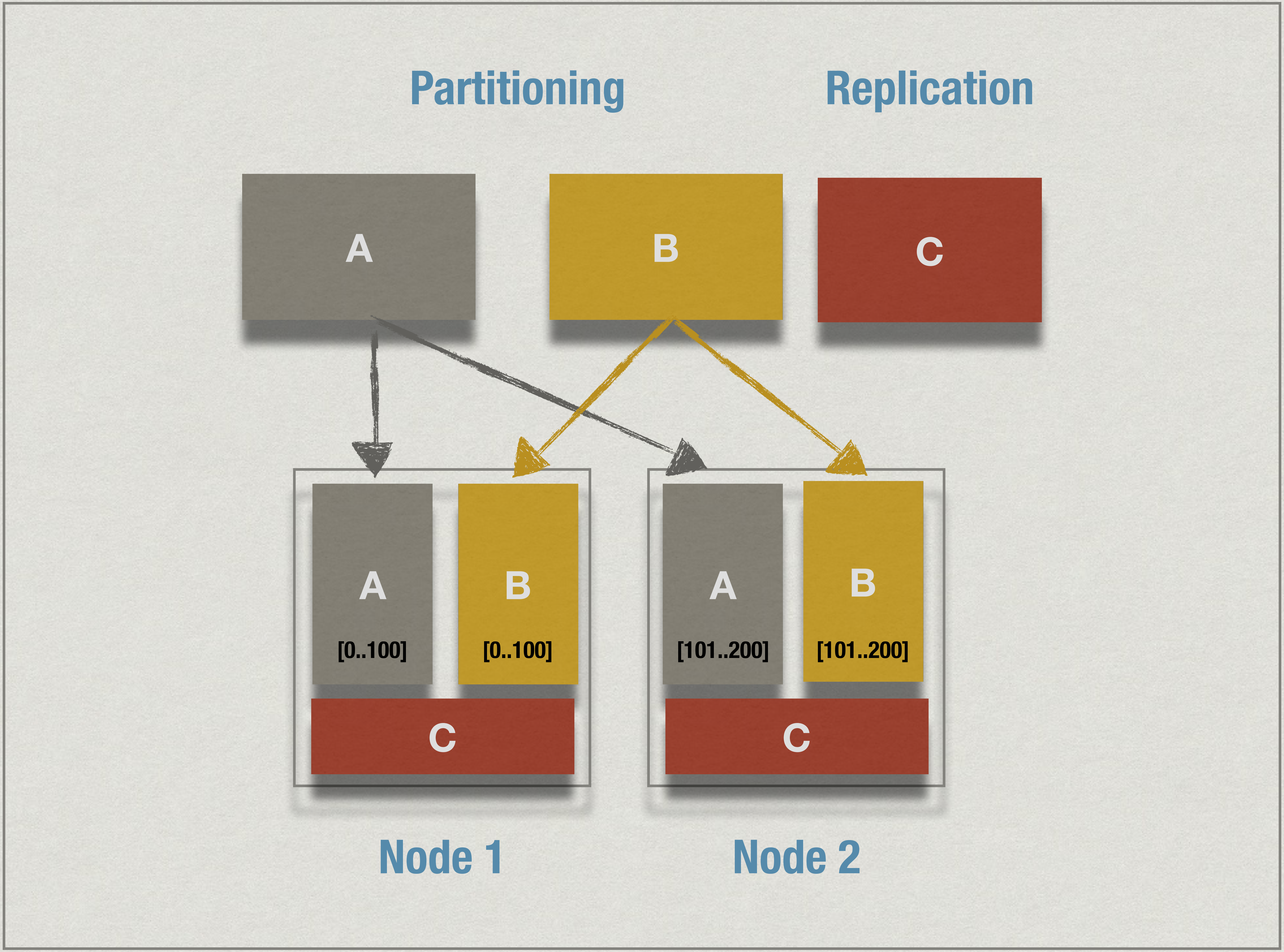
DynamoDB uses vector clocks; Cassandra uses a clever form of last-write-wins

You cannot <u>NOT</u> choose P

Partition Tolerance is <u>mandatory</u> in distributed systems

To scale, partition

To resilient, replicate


Partitioning
Replication
A
B
C
A
[0..100]
B
[0..100]
C
A
[101..200]
B
[101..200]
C
Node 1
Node 2

Replication is ***strongly*** related to fault-tolerance

Fault-tolerance relies on reaching **consensus**

Paxos (1989), ZAB, Raft (2013)

Consistent Hashing is used often PARTITIONING and REPLICATION

C-H commonly used in load-balancing web objects. In DynamoDB Cassandra, Riak and Memcached

- If your problem can fit into memory of a single machine, you don’t need a distributed system
- If you really need to design / build a distributed system, the following helps:
  - Design for failure
  - Use the FLP and CAP to critique systems
  - Algorithms that work for single-node may not work in distributed mode
  - Avoid coordination of nodes
- Learn to estimate your capacity

Jeff Dean - Google

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

REFERENCES

- http://queue.acm.org/detail.cfm?id=2655736 "The network is reliable"
- http://cacm.acm.org/blogs/blog-cacm/83396-errors-in-database-systems-eventual-consistency-and-the-cap-theorem/fulltext
- http://mvdirona.com/jrh/talksAndPapers/JamesRH_Lisa.pdf
- "Towards Robust Distributed Systems," http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf
- "Why Do Computers Stop and What Can be Done About It," Tandem Computers Technical Report 85.7, Cupertino, Ca., 1985. http://www.hpl.hp.com/techreports/tandem/TR-85.7.pdf
- http://www.cs.cornell.edu/projects/ladis2009/talks/dean-keynote-ladis2009.pdf
- http://codahale.com/you-cant-sacrifice-partition-tolerance/
- http://dl.acm.org/citation.cfm?id=258660 "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web"