CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2016
Doc 27 NoSQL
Dec 6, 2016

Relational Databases and SQL

Database consists of a number of tables

Table is a collection of records

Each Column of data has a type

firstname	lastname	phone	code
John	Smith	555-9876	2000
Ben	Oker	555-1212	9500
Mary	Jones	555-3412	9900

Use Structured query language (SQL) to access data

Common SQL Statements

SELECT	Retrieves data from table(s)
INSERT	Adds row(s) to a table
UPDATE	Changes field(s) in record(s)
DELETE	Removes row(s) from a table Data Definition
CREATE TABLE	Define a table and its columns(fields)
DROP TABLE	Deletes a table
ALTER TABLE	Adds a new column, add/drop primary key
CREATE INDEX	Create an index
DROP INDEX	Deletes an index
CREATE VIEW	Define a logical table from other table(s)/view(s)
DROP VIEW	Deletes a view

SQL is not case sensitive but data is case sensitive

Creating a Table

```
CREATE  TABLE  IF NOT EXISTS SampleTable (
    name TEXT UNIQUE ,
    age INTEGER check(typeof(age) = 'integer') ,
    is_student BOOL,
    description TEXT check(typeof(description) = 'text') ,
    id INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL )
```

name	age	is_student	description	id

ACID

Atomicity

Transactions are all or nothing

Consistency

Transactions change the database from one valid state to valid state

Isolation

Transaction's affect is not seen until transaction is done

Concurrent transactions will have the same affect as if they were done serially

Durability

Once a transaction is done its changes remain

https://en.wikipedia.org/wiki/ACID

Issues with SQL Databases

Need to know structure of data in advance

Difficult to modify structure after created

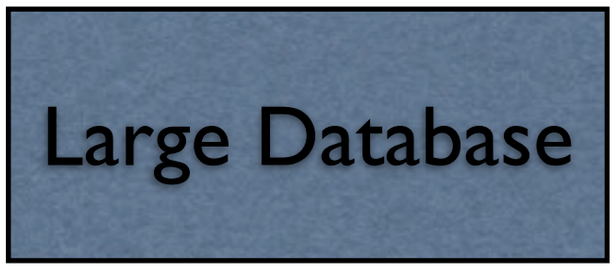
Difficulty in scaling

Performance

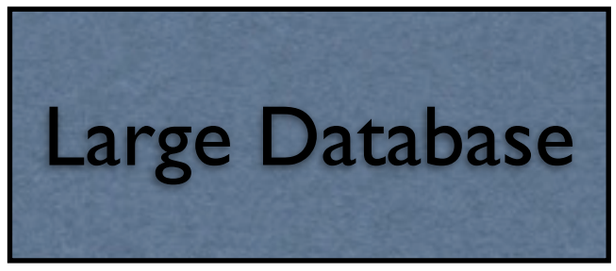
Scaling to Multiple Machines - Replication

Each machine has complete copy

Machine 1

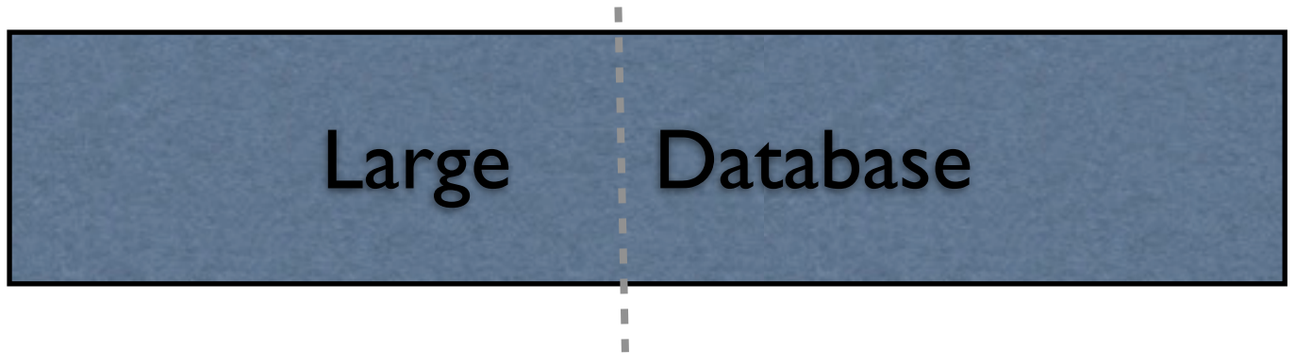


Machine 2



Scaling to Multiple Machines - Sharding

Each machine has only part of the data



Machine 1

Large

Machine 2

Database

CAP Theorem

CAP theorem says in a distributed system you can not have all three

Consistency

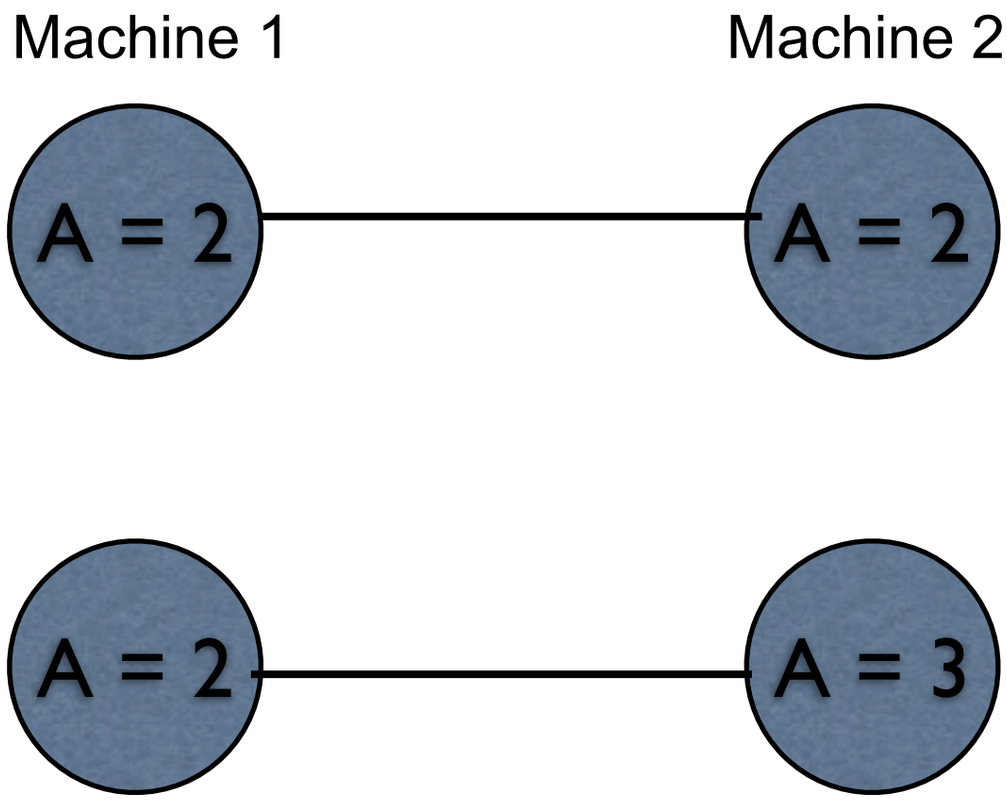
Availability

tolerance to network **P**artitions

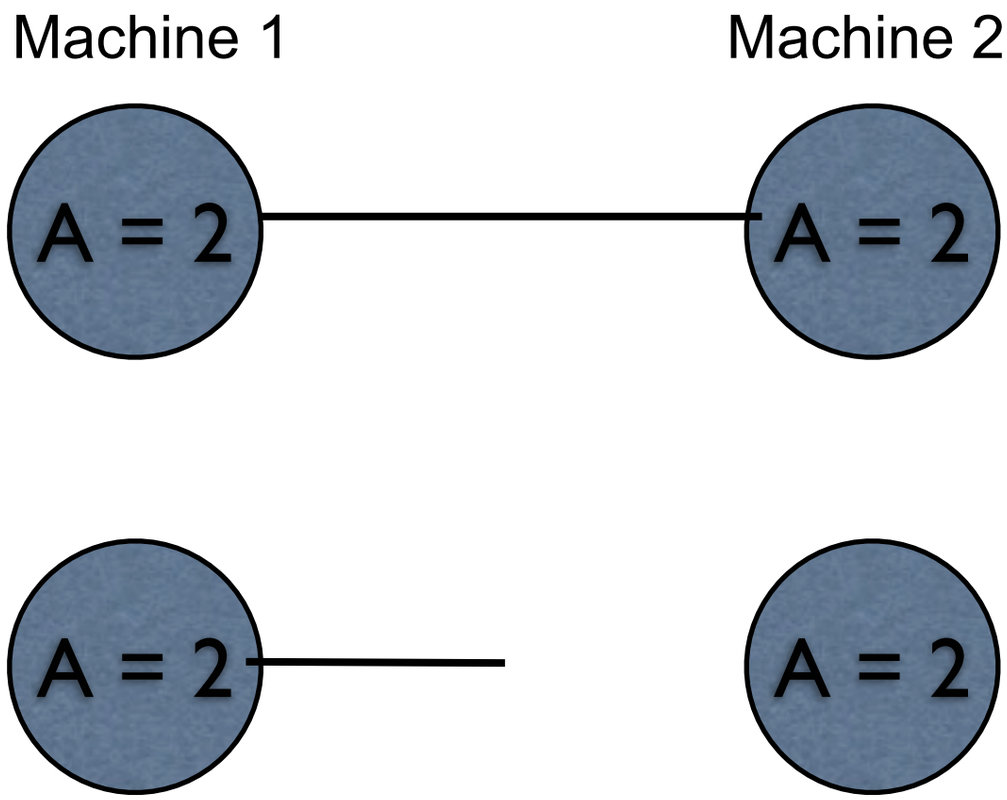
Tuesday, December 6, 16

https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

Consistency



Partition



Machine 1 cannot talk to machine 2

But how does machine 1 tell the difference between no connection and a very slow connection or busy machine 2?

Latency

Latency

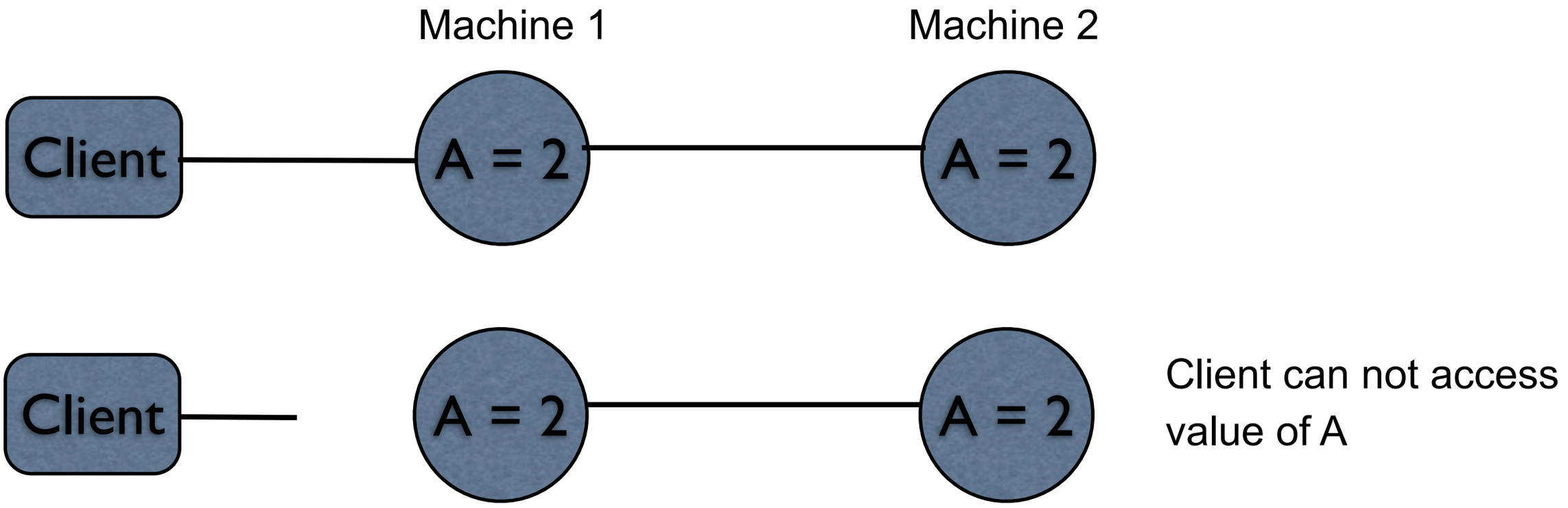
Time between making a request and getting a response

Distributed systems always have latency

In practice detect a partition by latency

When no response in a given time frame assume we are partitioned

Available



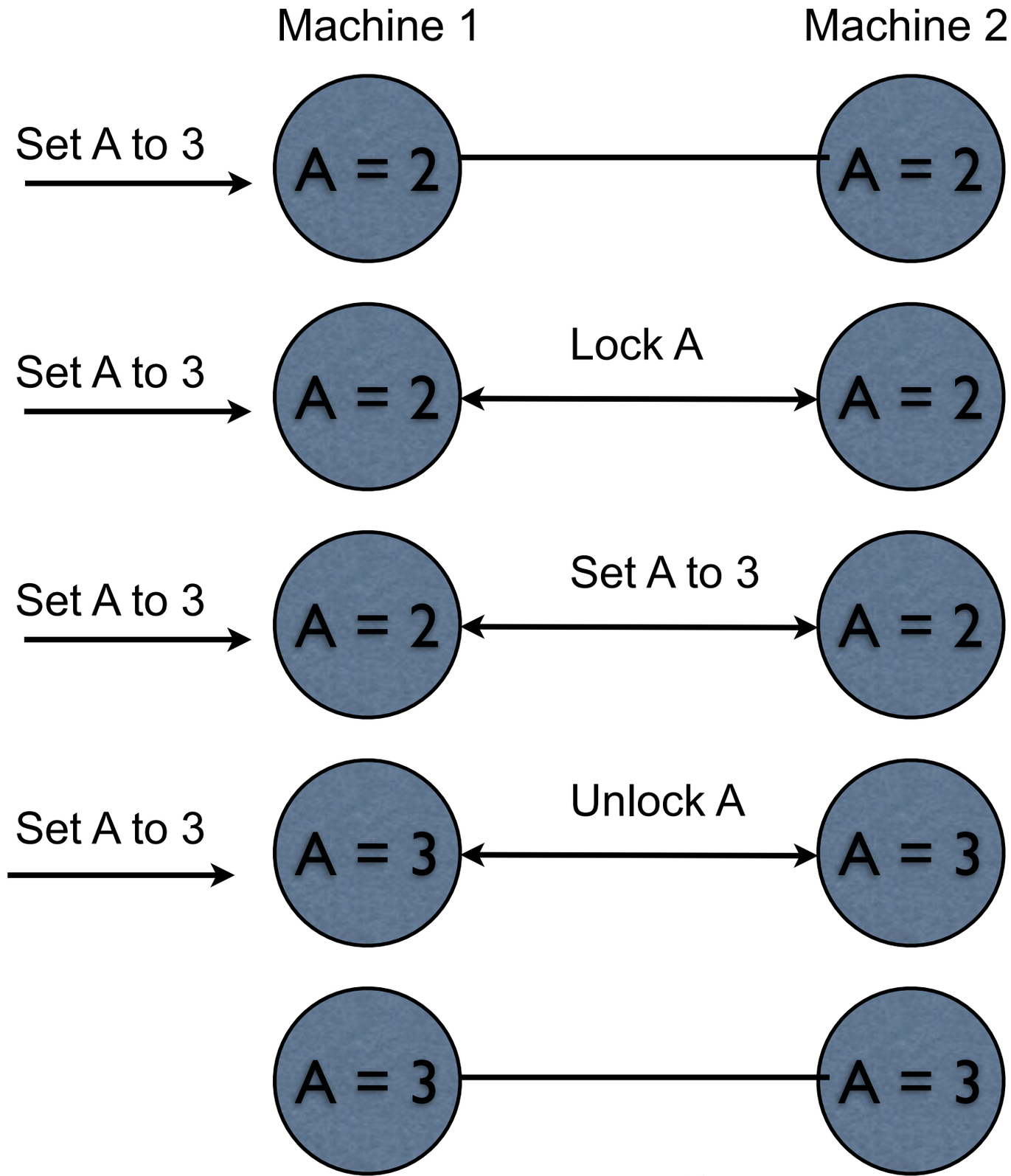
What does not available mean?

- No connection
- Slow connection
- What is the difference?

Some say high available - meaning low latency

In practice available and latency are related

Consistency over Latency

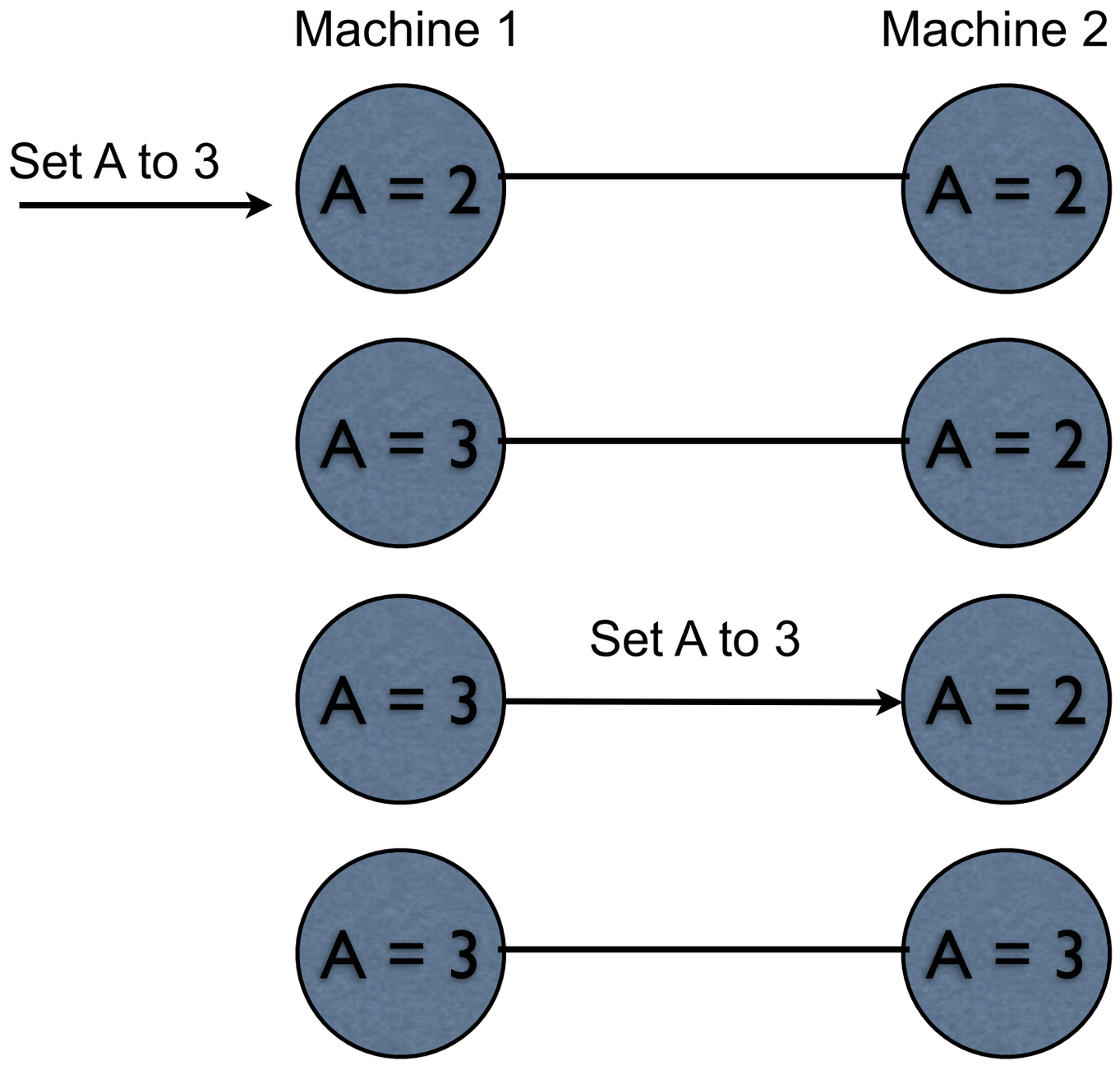


Write requests queued until unlocked

Increased latency

System still available

Latency over Consistency

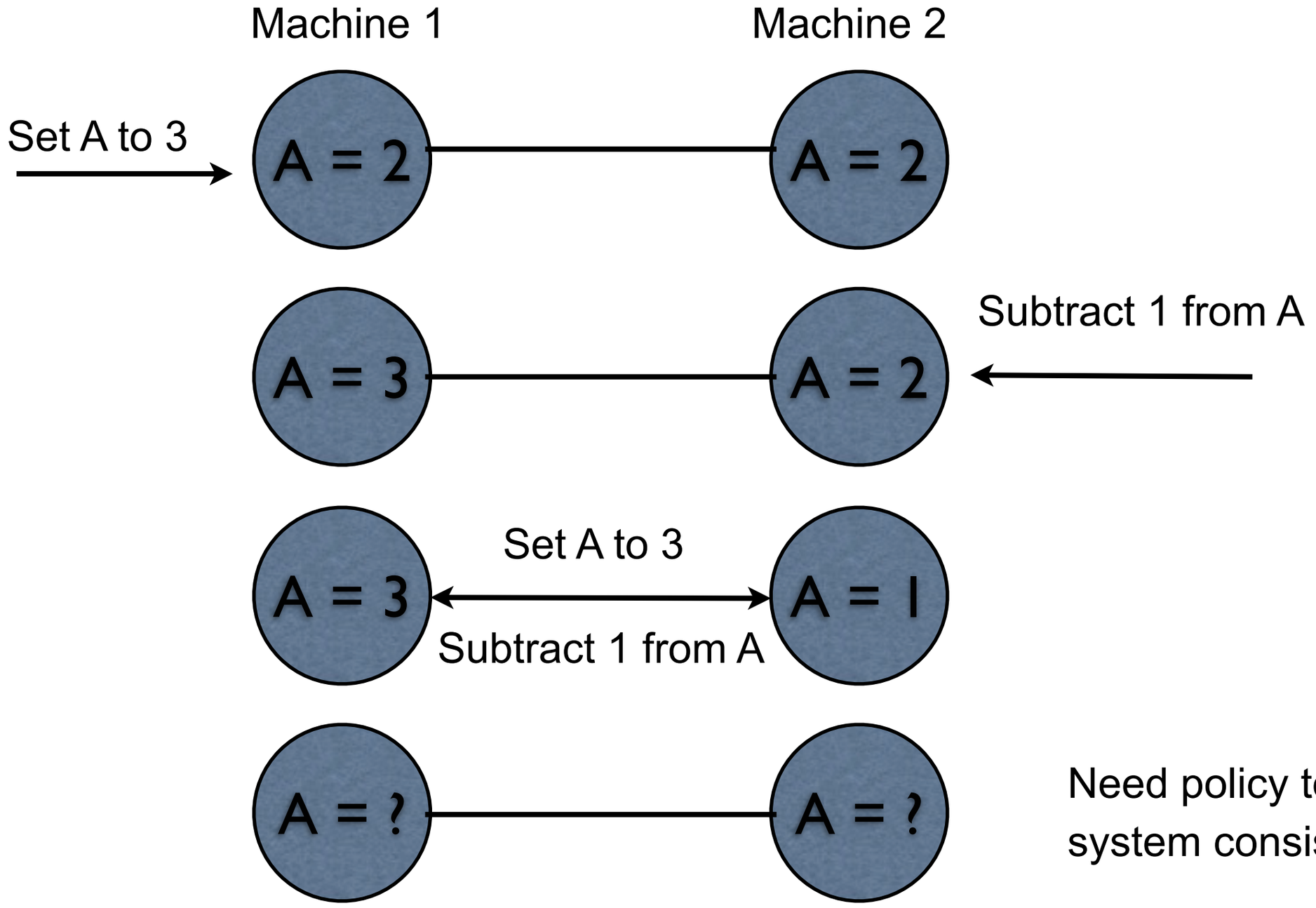


Write requests accepted

Low latency

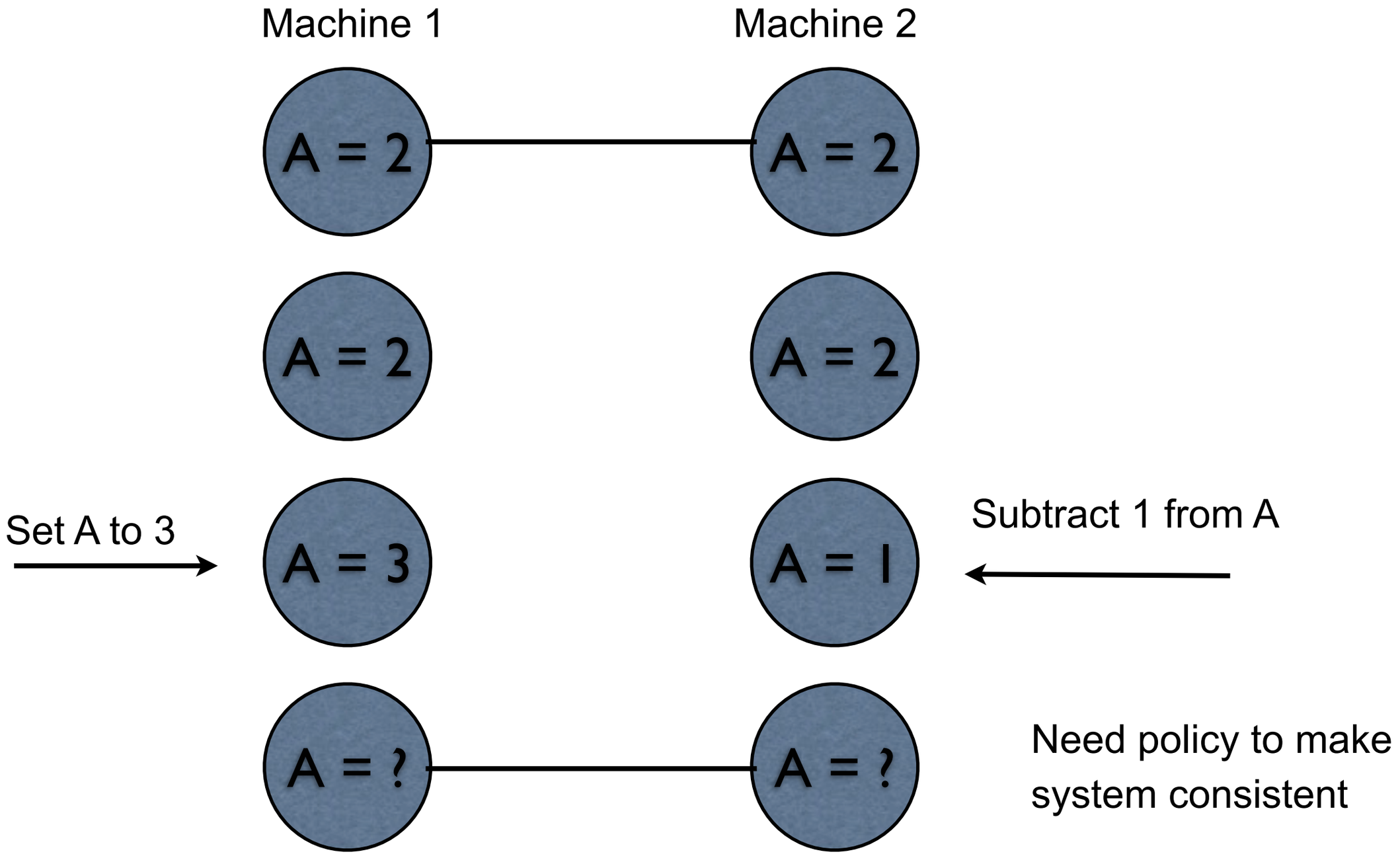
System inconsistent

Latency over Consistency - Write Conflicts



Need policy to make system consistent

Partition


Machine 1
Machine 2
A = 2
A = 2
A = 2
A = 2
Set A to 3
A = 3
A = 1
Subtract 1 from A
A = ?
A = ?
Need policy to make system consistent

CAP Theorem

Not a theorem

Too simplistic

- What is availability
- What is a partition of the network

Misleading

Intent of CAP was to focus designers attention on the tradeoffs in distributed systems

- How to handle partitions in the network
- Consistency
- Latency
- Availability

https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

NoSQL verses ACID SQL Database

NoSQL databases

Tend to prioritize low latency over consistency

ACID SQL databases

Prioritize consistency over low latency

This is why NoSQL databases tend to scale better than SQL databases

More Terminology

BASE - **B**asically **A**vailable, **S**oft state, **E**ventual consistency

Systems that favor low latency over consistency

NewSQL

Modern databases that

- SQL as a primary interface
- ACID support for transactions
- Non-locking architecture
- High per-node performance
- Scalable, shared nothing architecture

NewSQL databases become competitive with NoSQL

Examples

- Google Spanner
- VoltDB
- MySQL Cluster
- CockroachDB

NoSQL Database

Don’t use

- SQL
- Tables

NoSQL database types

- Document (XML, YAML, JSON, BSON)
  - MongoDB, CouchDB, Couchbase, ...
- Graph
  - Neo4J, AllegroGraph, ...
- Key-value
  - SimpleDB, DynamoDB, Redis, Oracle NoSQL, ...
- Column
  - Accumulo, Cassandra, HBase, ...

Issues

If no SQL how to access/modify database

How to structure the data

HBase

Problem with HDFS - no random access to data

How do you store 4+ billion web pages so you can search them fast

2006 - Google BigTable

Fault-tolerant way of storing large quantities of sparse data

2007 - HBase started as part of Hadoop

2010 - Hbase becomes Apache top-level project

2015 - Hbase 1.0 released

HBase

Written in Java

Runs in

Standalone, Pseudo-distributed, Cluster

Stores data in HDFS

Has interactive shell

Hadoop & Spark can read/write to Hbase

In CAP theorem

C - consistency

P - partition tolerance

HBase - Users

Facebook uses HBase for its messaging platform

Spotify uses HBase as base for Hadoop and machine learning jobs

Airbnb uses HBase as part of its AirStream realtime stream computation framework

Twitter

Yahoo

HBase - Parts

Namespace

- Group of tables - like database in relational databases

Table

- Group of rows

Row

- One or more columns
- A row has a row key - just bytes
- Rows are sorted by row key

Column-families

- Group of columns
- When create table must specify column families
- Each column family can contain any number of columns
- HBase does not do well with more than three column families per table

HBase - Parts

Column

- Consists of a column family name & column qualifier
- Different rows can have different columns

Cell

- At a given Row and Column Hbase stores a cell
- Each cell contains a value and a timestamp
- Value - bytes

Some Shell Operations

create

create a table

get

get a given row

get a value at a row and column

put

Add/modify a value at a row & column

scan

Show the rows in a table

list

List the tables

```
hbase(main):002:0> create 'testtable', 'names'
0 row(s) in 1.5660 seconds
=> Hbase::Table - testtable

hbase(main):003:0> list
TABLE
testtable
1 row(s) in 0.0320 seconds
=> ["testtable"]


hbase(main):004:0> put 'testtable', 'person-1', 'names:first', 'Roger'
0 row(s) in 0.4690 seconds

hbase(main):005:0> put 'testtable', 'person-2', 'names:first', 'Peter'
0 row(s) in 0.0190 seconds

hbase(main):006:0> put 'testtable', 'person-2', 'names:last', 'Rabbit'
0 row(s) in 0.0180 seconds
```

```
hbase(main):008:0> scan 'testtable'
ROW                    COLUMN+CELL
 person-1              column=names:first, timestamp=1480995132309, value=Roger
 person-2              column=names:first, timestamp=1480995947448, value=Peter
 person-2              column=names:last, timestamp=1480995995526, value=Rabbit
2 row(s) in 0.0380 seconds
```

```
 hbase(main):010:0> get 'testtable', 'person-2'
COLUMN                 CELL
 names:first           timestamp=1480995947448, value=Peter
 names:last            timestamp=1480995995526, value=Rabbit
2 row(s) in 0.0470 seconds
```

```
hbase(main):011:0> put 'testtable', 'person-3', 'names:last', 'Gates'
0 row(s) in 0.0160 seconds

hbase(main):012:0> put 'testtable', 'person-3', 'names:first', 'William'
0 row(s) in 0.0040 seconds

hbase(main):013:0> put 'testtable', 'person-3', 'names:Middle', 'Henry'
0 row(s) in 0.0030 seconds

hbase(main):014:0> put 'testtable', 'person-3', 'names:knickname', 'Bill'
0 row(s) in 0.0050 seconds

hbase(main):015:0> get 'testtable', 'person-3'
COLUMN                          CELL
 names:Middle                   timestamp=1480996890990, value=Henry
 names:first                    timestamp=1480996861847, value=William
 names:knickname                timestamp=1480996931417, value=Bill
 names:last                     timestamp=1480996838910, value=Gates
4 row(s) in 0.0220 seconds
```