



Principles of High Load

Peter Milne

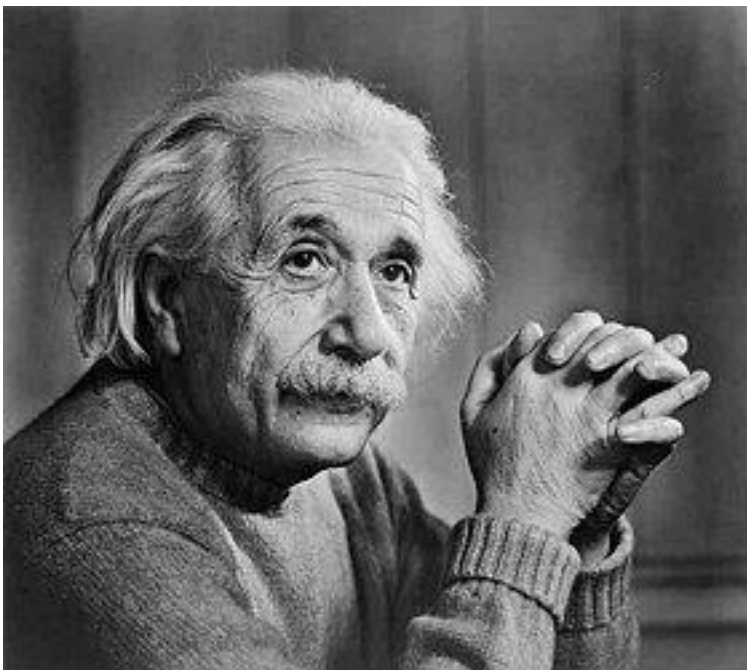
peter@aerospike.com

 @helipilot50

 helipilot50

AEROSPIKE

Wisdom vs Guessing

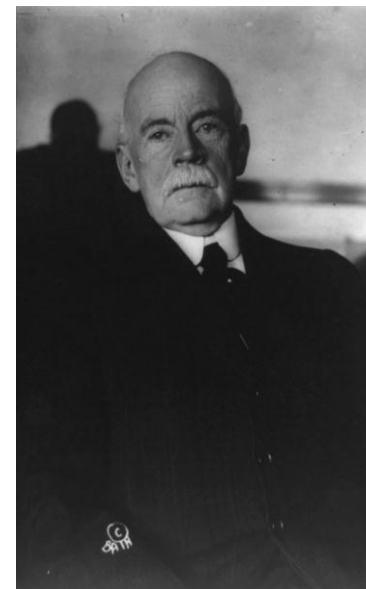


“Insanity is doing the same thing over & over again expecting different results”

- Albert Einstein

“Everything that can be invented has been invented.” -

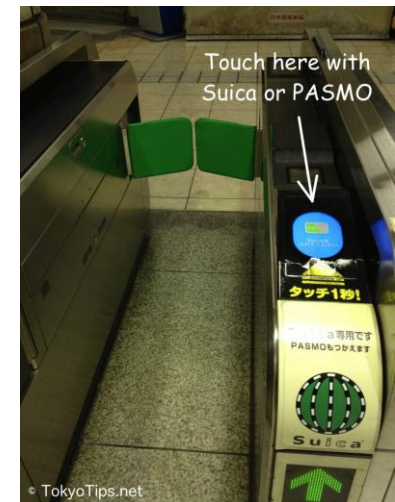
Charles Holland Duell - US Patent Office 1899



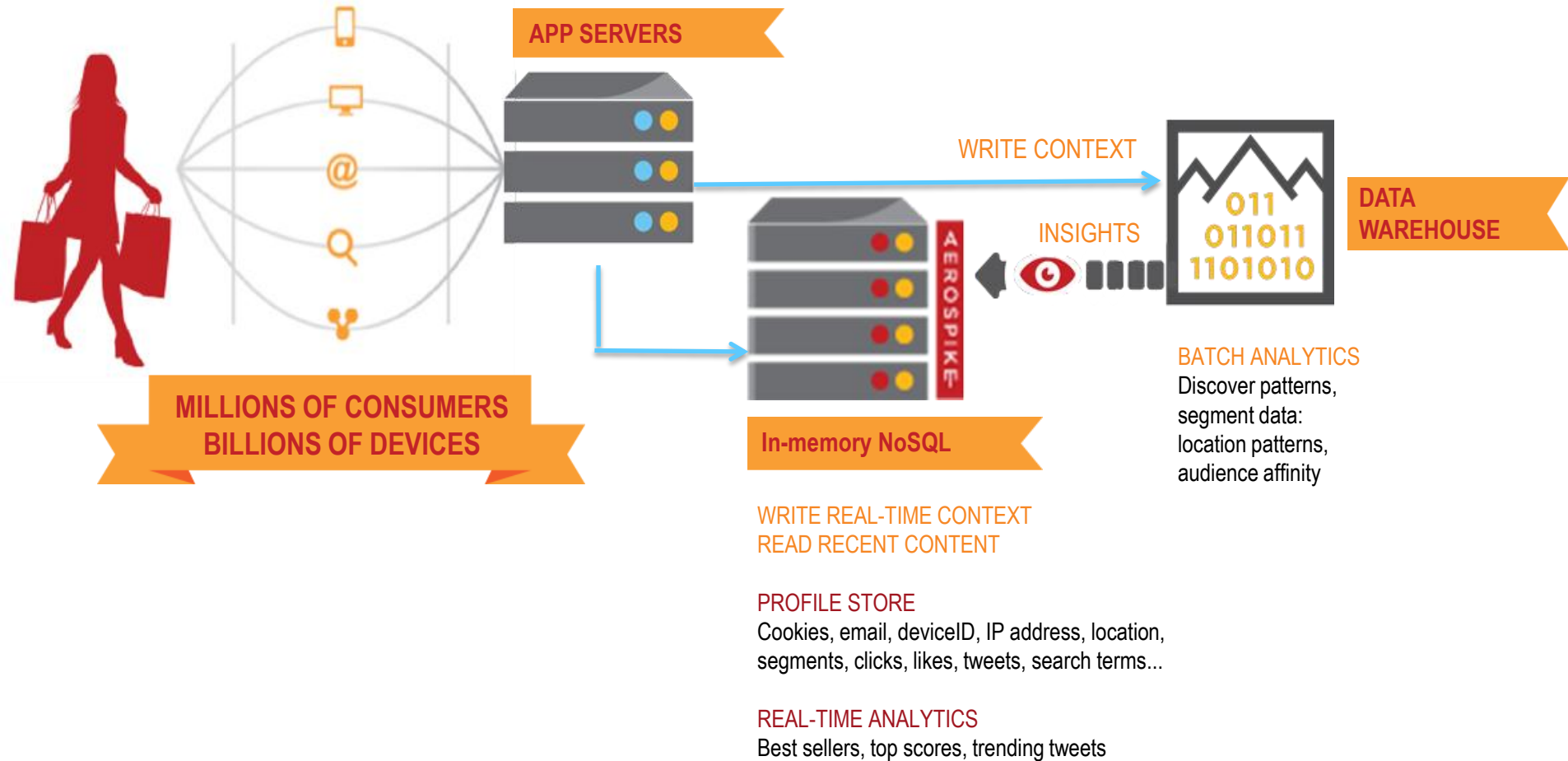
High load

Shinagawa Railway Station - Tokyo, Japan

12 December 2014 08:22 AM

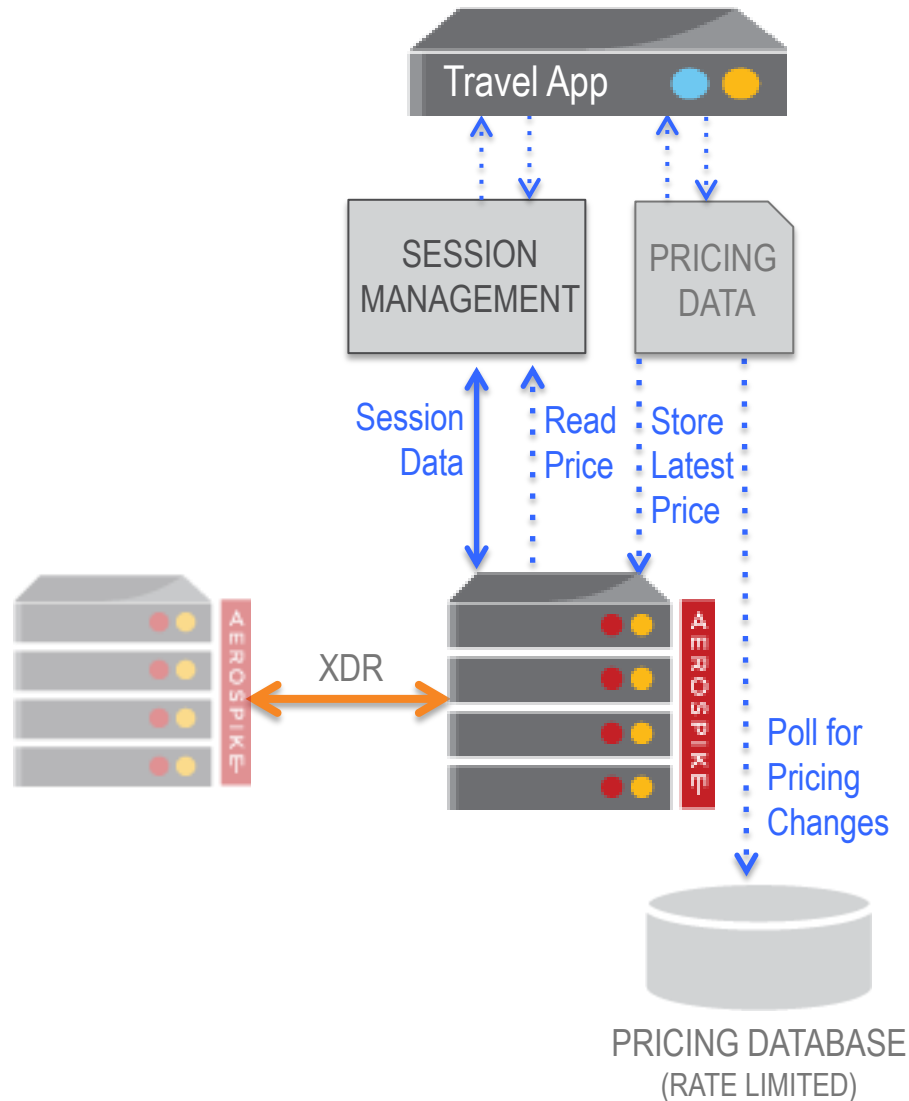


Advertising Technology Stack



Currently about 3.0M / sec in North American

Travel Portal



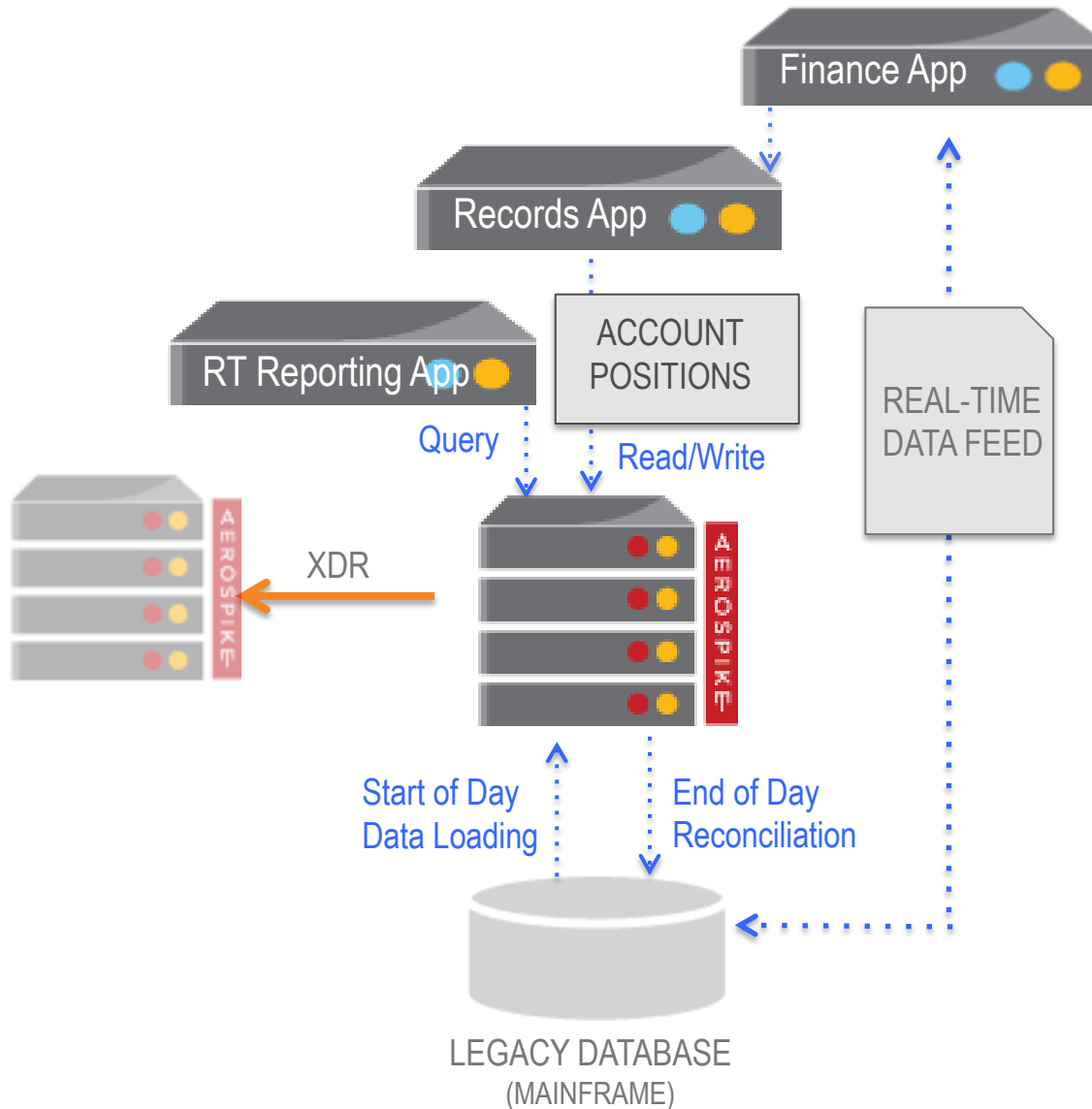
Airlines forced interstate banking

Legacy mainframe technology

Multi-company reservation and pricing

Requirement: **1M TPS** allowing overhead

Financial Services – Intraday Positions



10M+ user records
Primary key access
1M+ TPS



Principles

Little's Law

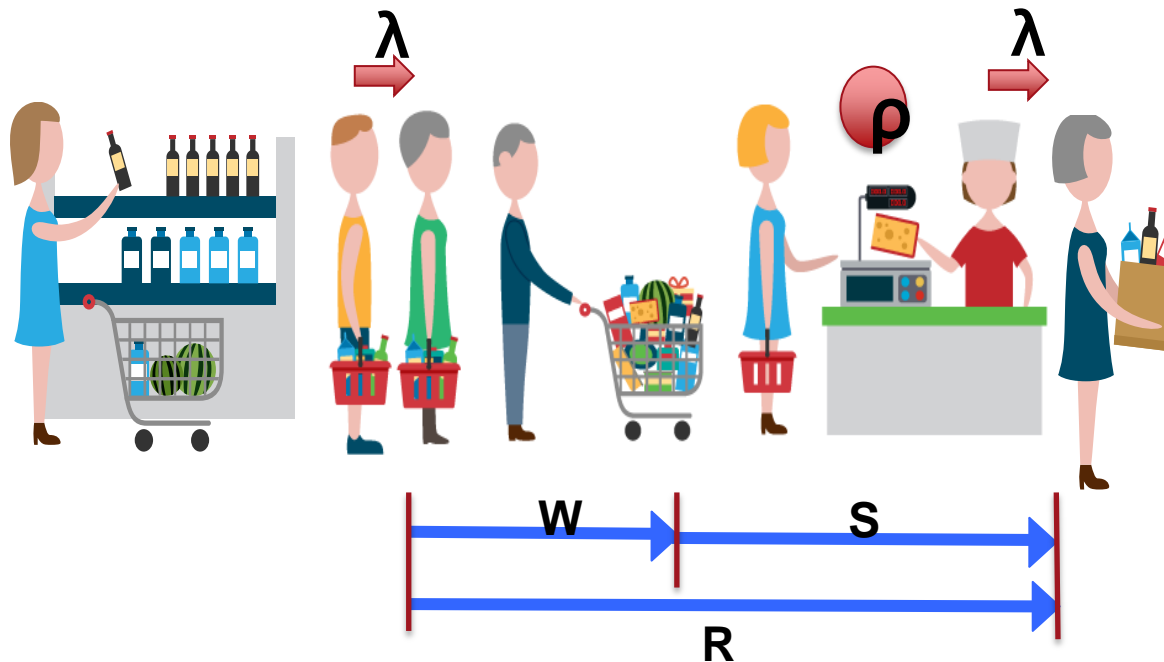
The long-term average number of customers **L** in a stable system is equal to the long-term average effective **arrival rate** λ , multiplied by the **average time W** a customer spends in the system

Length of queue

Average wait time

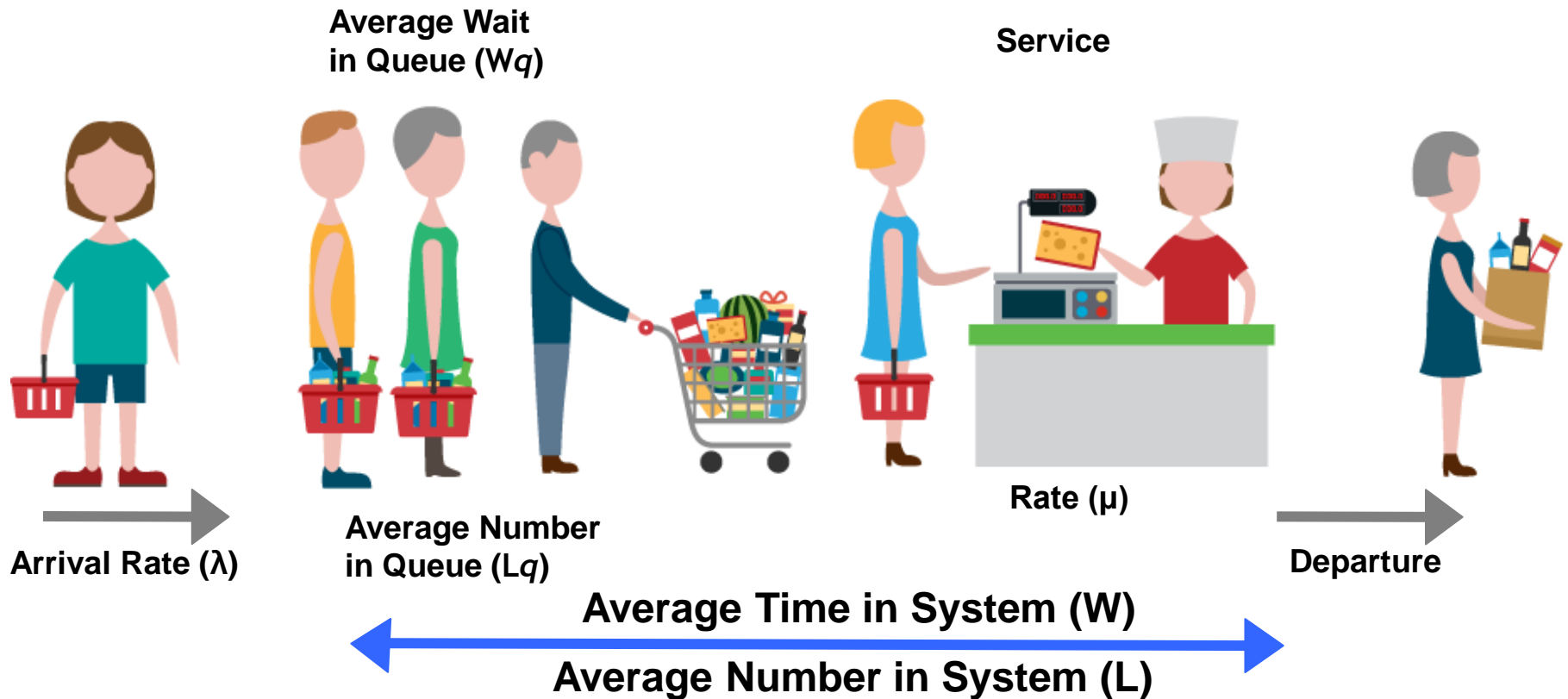
$$L = \lambda W$$

Arrival Rate



Queuing Theory

- **Queuing theory** is the mathematical study of waiting lines, or queues.



Throughput

Throughput is the **rate of production** or the rate at which something can be processed

Similar to **Power**: “work done / time taken”

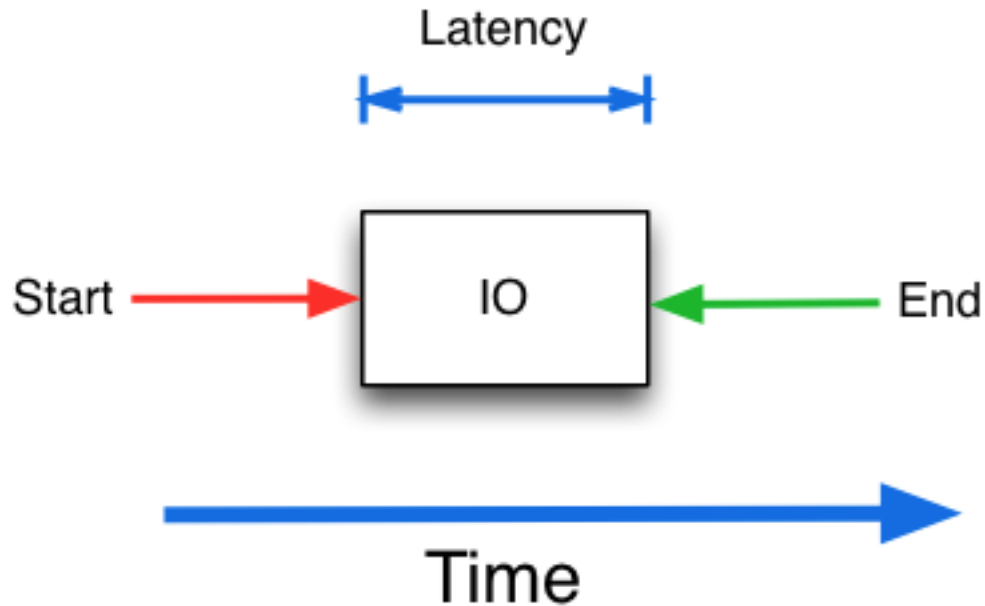
$$P = \frac{W}{\Delta t}$$



The **power** of a system is **proportional to its throughput**

Latency

Latency is a **time interval** between the stimulation and response, or, from a more general point of view, as a time delay between the cause and the effect of some physical change in the system being observed.



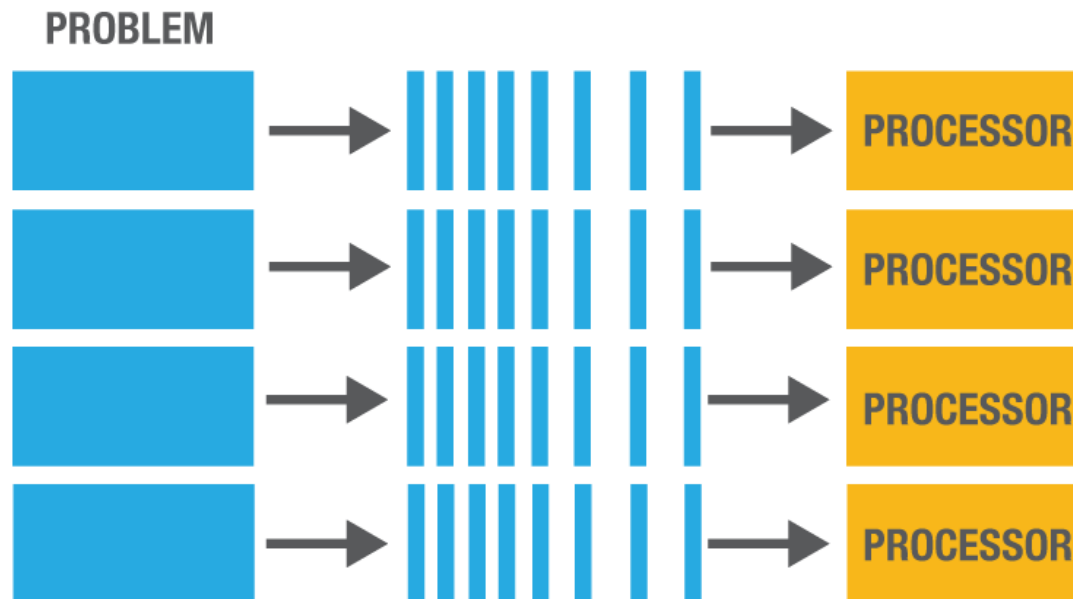
Concurrency

- **Concurrency** is a property of systems in which several computations are executing **simultaneously**, and potentially **interacting** with each other.



Division of labor – Parallel processing

Parallel processing is the **simultaneous use of more than one CPU** or processor core to execute a program or multiple computational threads. Ideally, parallel processing makes programs run faster because there are more engines (CPUs or cores) running it. In practice, it is often difficult to divide a program in such a way that separate CPUs or cores can execute different portions without interfering with each other.

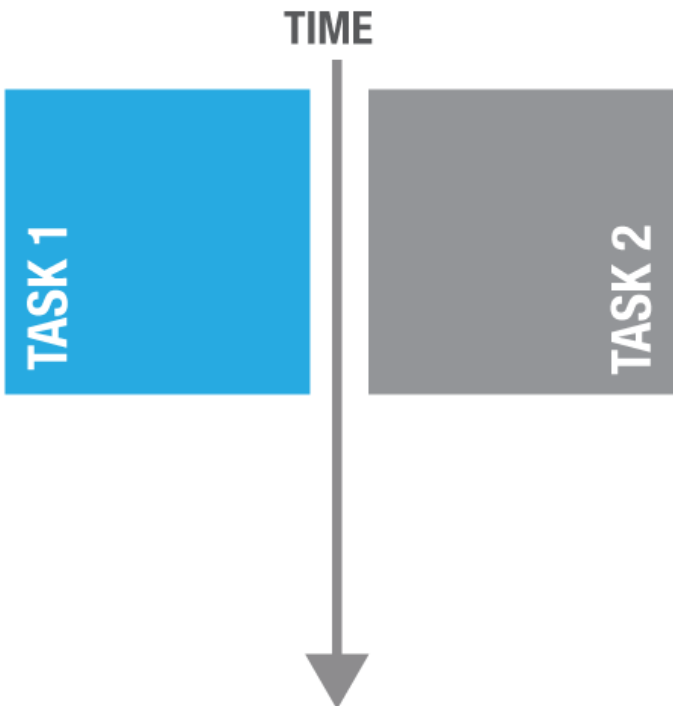


Concurrency vs Parallelism

CONCURRENCY

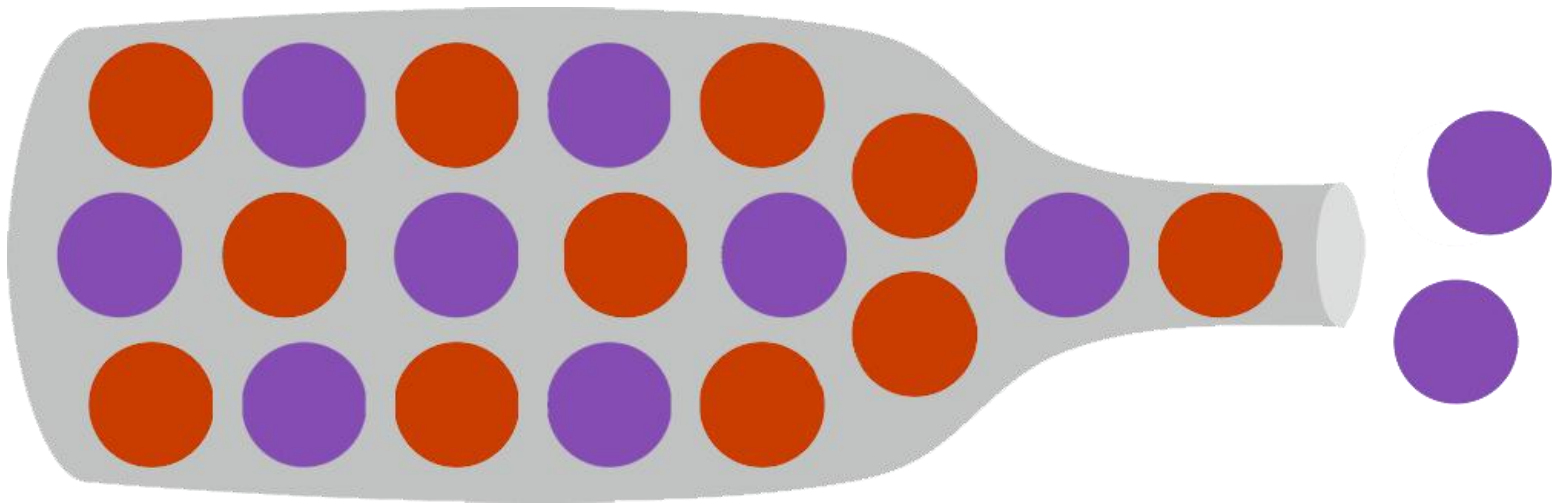


PARALLELISM



Bottle necks

Bottleneck is a phenomenon where the performance or capacity of an entire system is limited by a single or small number of components or resources



Locks, Mutexes and Critical Regions

■ Lock

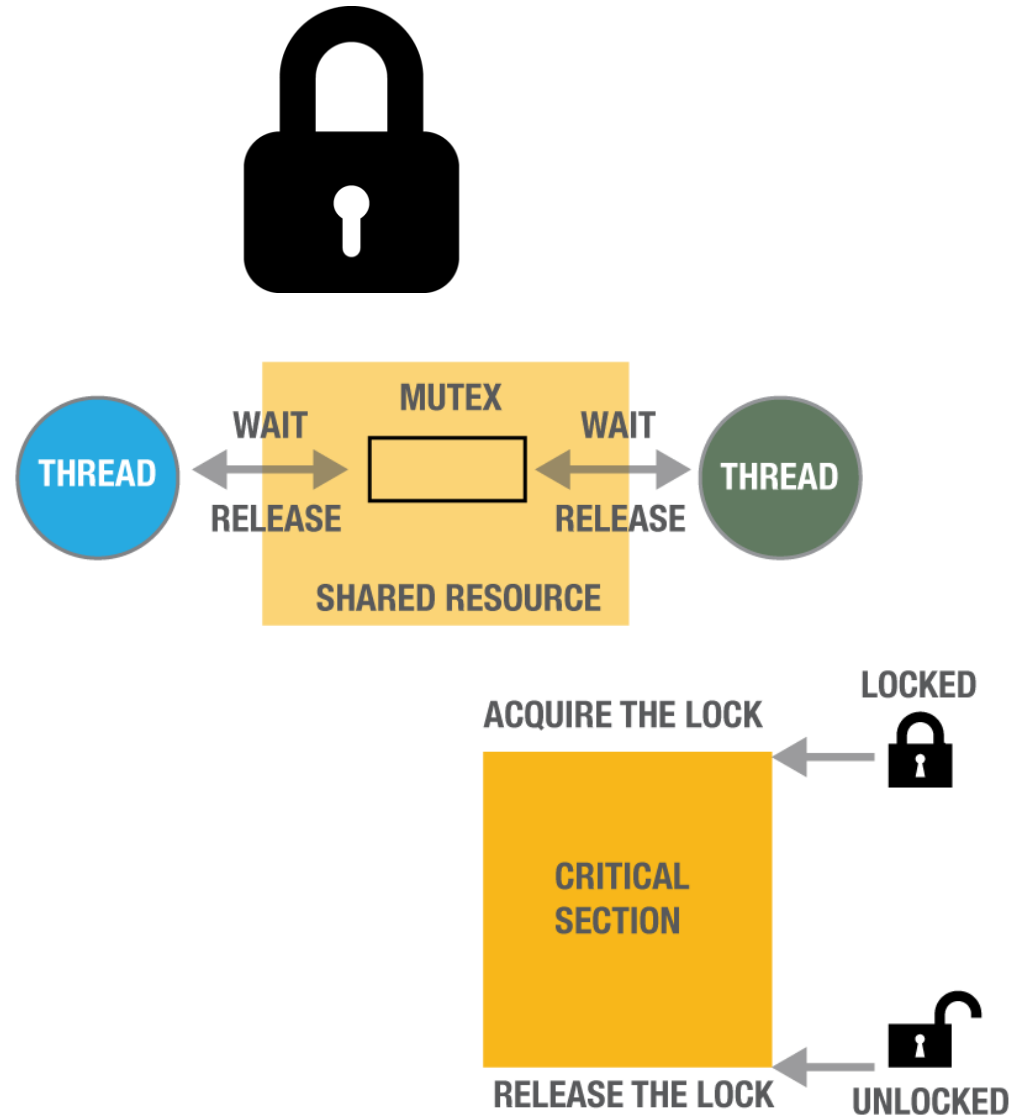
- Atomic Latch
- Hardware implementation
 - 1 machine instruction
- OS system routine

■ Mutex

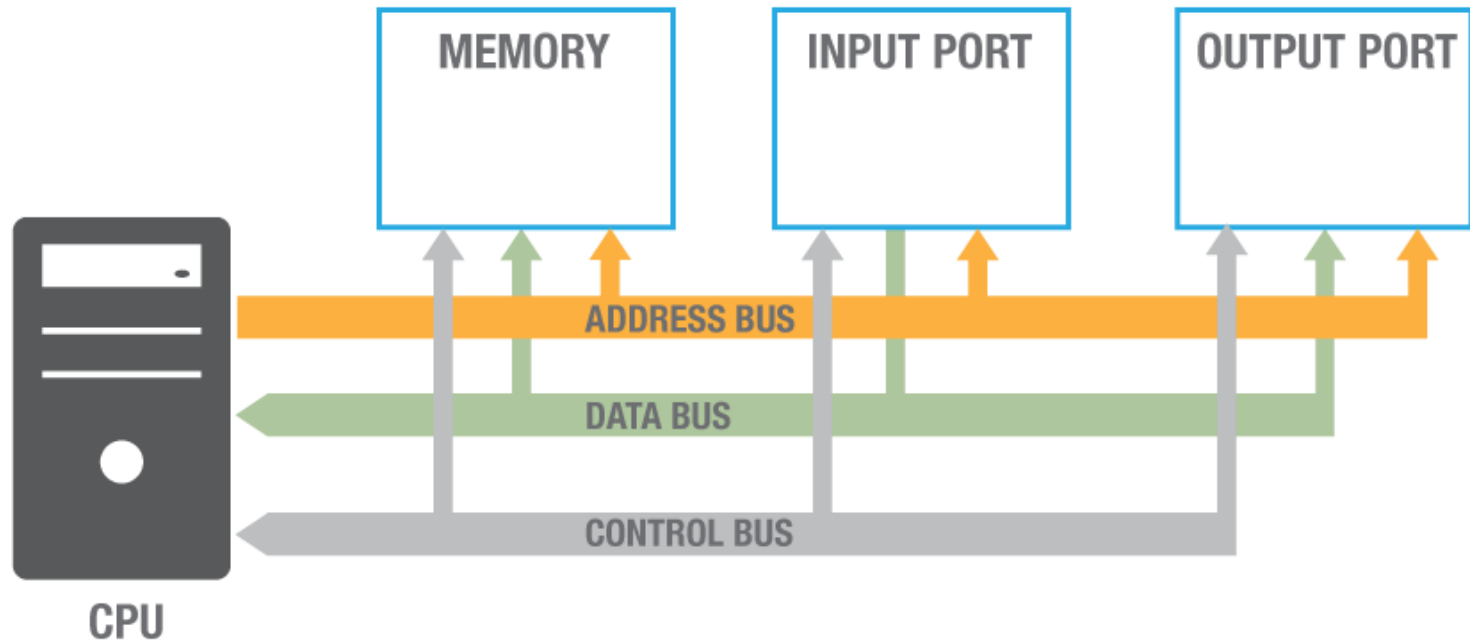
- Mutual exclusion
- Combination of a Lock and a Semaphore

■ Critical section

- Region of code allowing 1 thread only.
- Bounded by Lock/Mutex



Basic computer architecture



Multi-processor, Multi-core, NUMA

■ Multi-processor

- > 1 processor sharing Bus and Memory

■ Multi-core

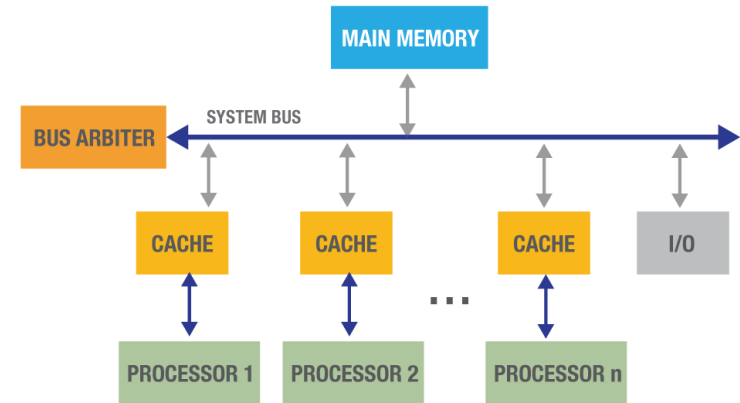
- > 1 processor in a chip
- Each with local Memory
- Access to shared memory

■ Non Uniform Memory Allocation

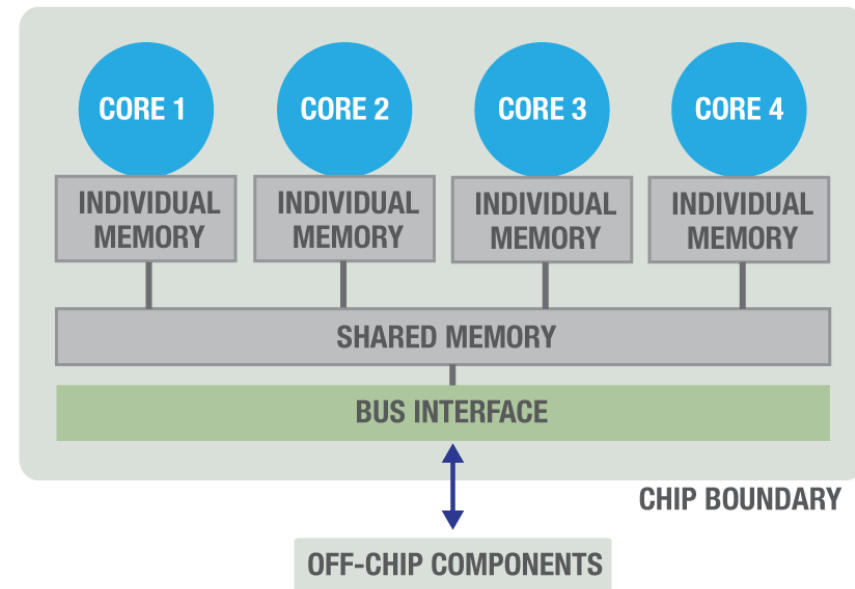
- Local memory faster to access than shared memory

■ Multi-channel Bus

SMP - SYMMETRIC MULTIPROCESSOR SYSTEM

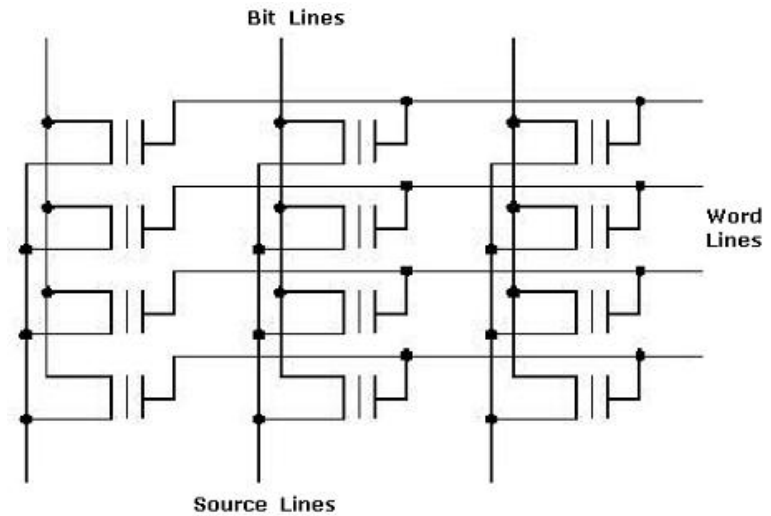
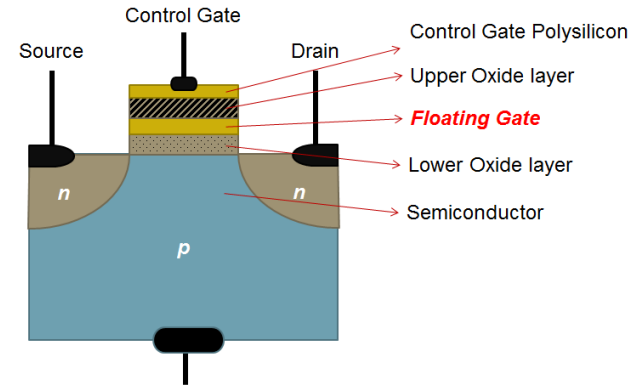


MULTI-CORE PROCESSOR



Flash - SSDs

- Uses Floating Gate MOSFET
- Arranged into circuits “similar” to RAM



- Packaged as PCIe or SATA devices
- **No seek or rotational latencies**





How Aerospike does it

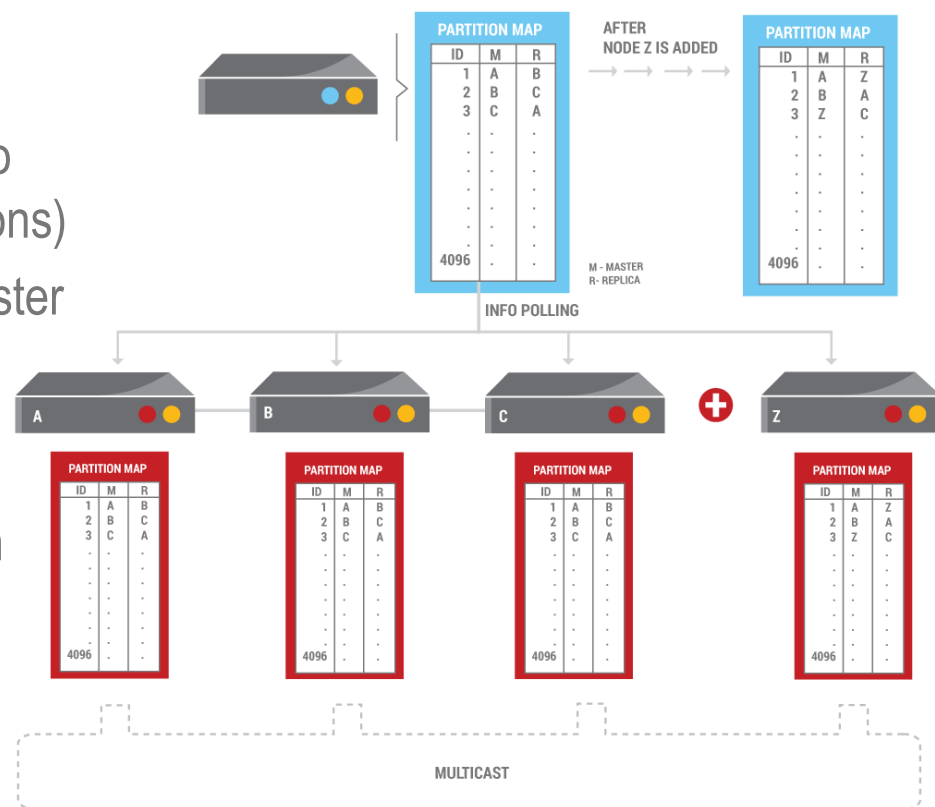
Smart Client -Distributed Hash table

■ Distributed Hash Table with No Hotspots

- Every key **hashed** with **RIPEND160** into an ultra efficient 20 byte (fixed length) string
- Hash + additional (fixed 64 bytes) data forms **index entry** in RAM
- **Some bits** from hash value are used to calculate the **Partition ID** (4096 partitions)
- Partition ID maps to Node ID in the cluster

■ 1 Hop to data

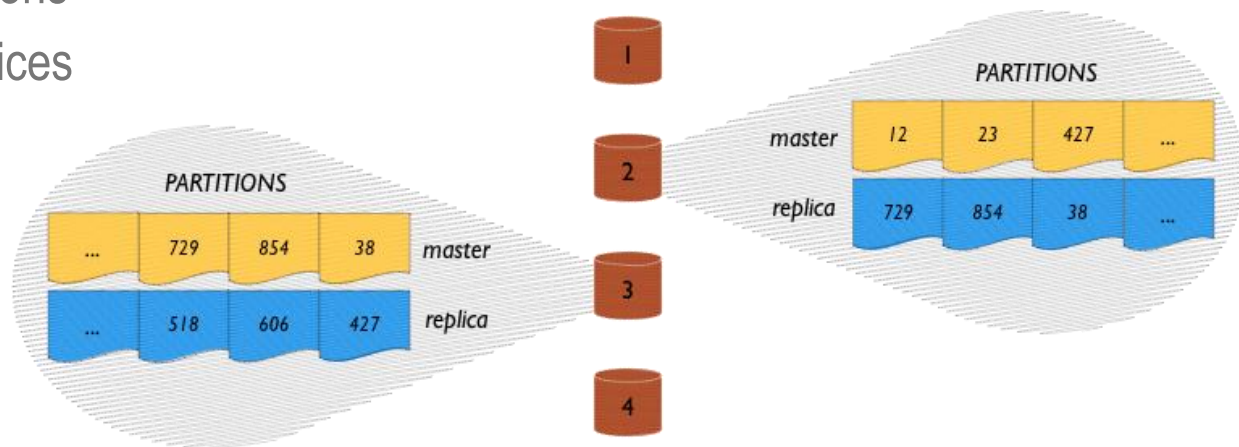
- Smart Client simply calculates Partition ID to determine Node ID
- No Load Balancers required



Data Distribution

Data is **distributed evenly** across nodes in a cluster using the Aerospike Smart Partitions™ algorithm.

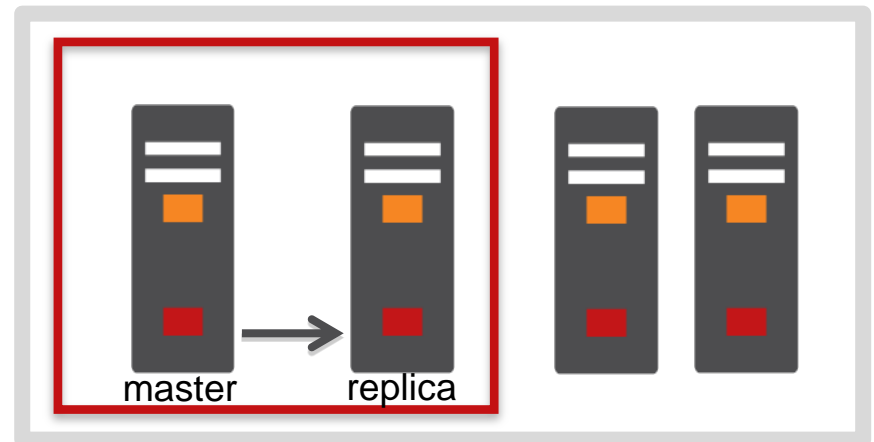
- **RIPEMD160** (no collisions yet found)
- 4096 Data Partitions
- Even distribution of
 - **Partitions** across nodes
 - **Records** across Partitions
 - **Data** across Flash devices
- Primary and Replica Partitions



Consistency

Tunable consistency – **Default: Immediate Consistency**

1. Write sent to row master
2. Latch against simultaneous writes
3. Apply write to **master** memory and **replica** memory **synchronously**
4. Queue operations to storage
5. **Signal completed** transaction
(optional storage commit wait)
6. Master applies conflict resolution policy (rollback/ rollforward)

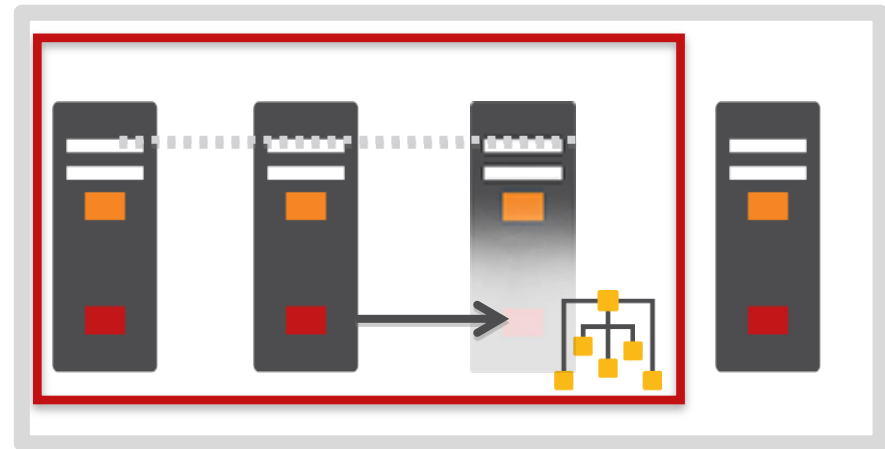


Automatic rebalancing

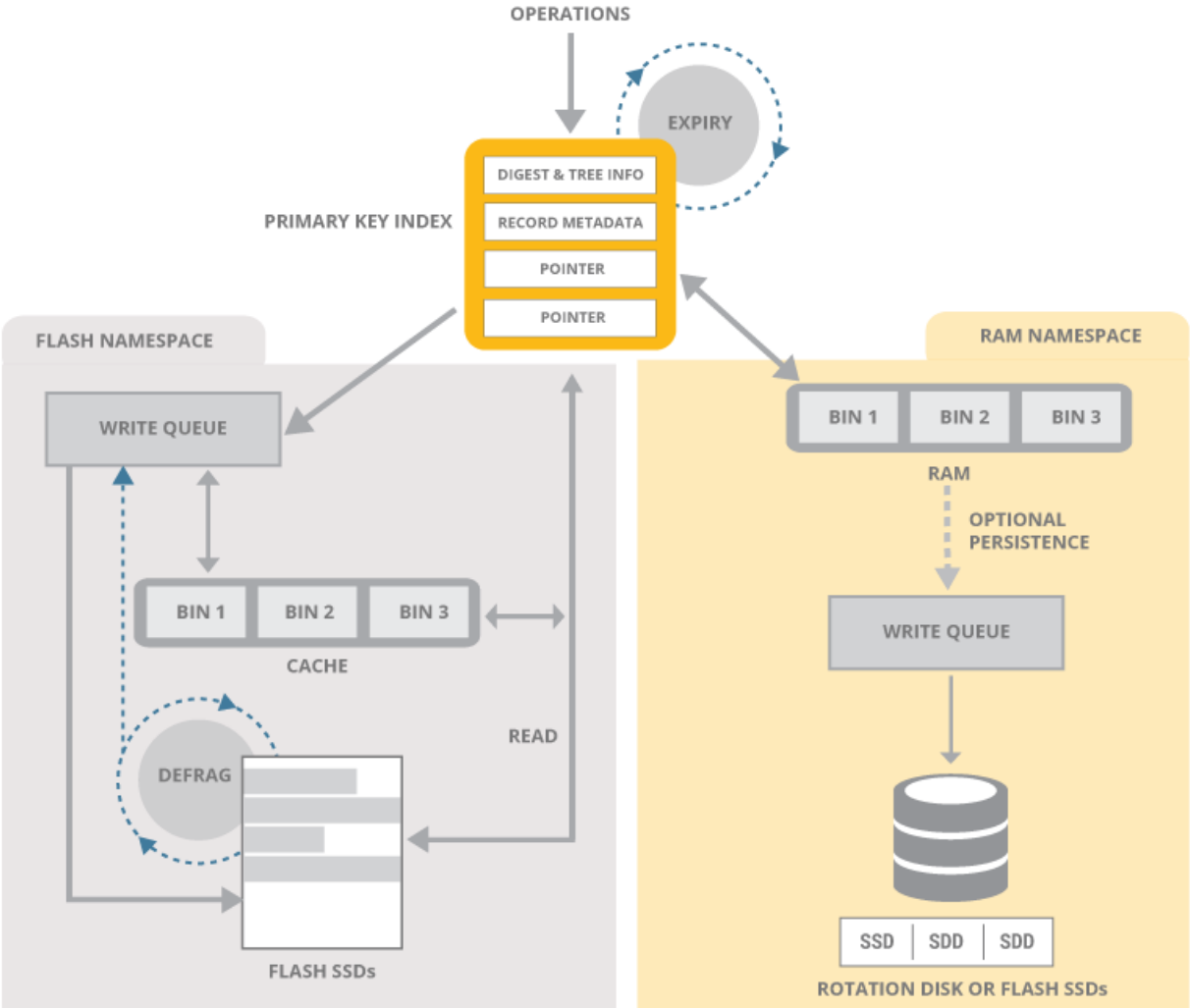
Adding, or Removing a node, the Cluster **automatically rebalances**

1. Cluster discovers new node via **gossip** protocol
2. **Paxos** vote determines new data organization
3. **Partition migrations** scheduled
4. When a partition migration starts, write journal starts on destination
5. Partition moves atomically
6. Journal is applied and source data deleted

After migration is complete, the Cluster is **evenly** balanced.

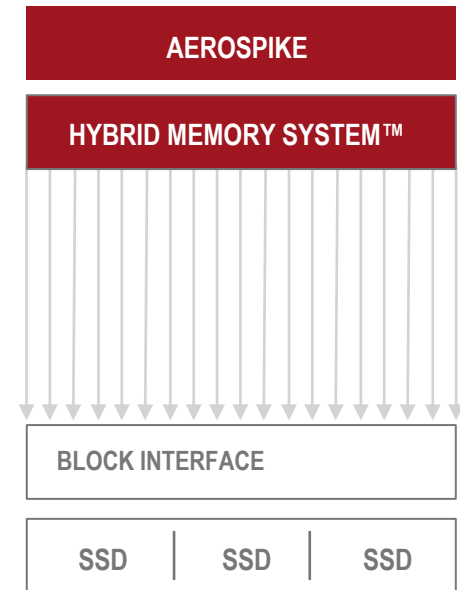


Data Storage Layer



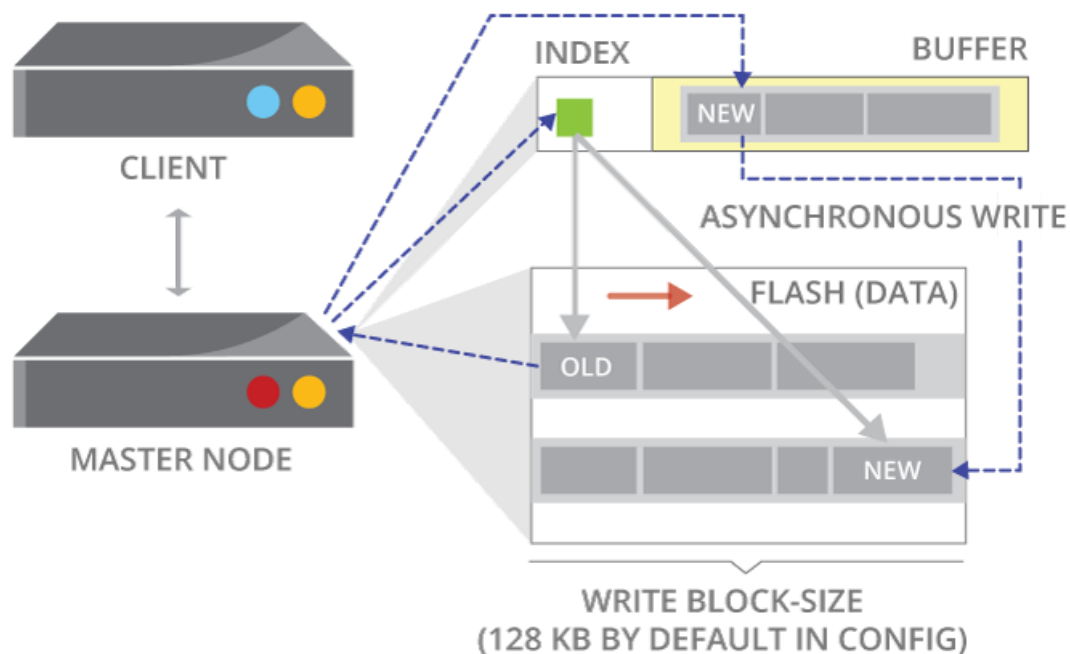
Data on Flash / SSD

- Record data stored contiguously
 - 1 read per record (multithreaded)
- Automatic **continuous defragment**
- Data written in **flash optimal blocks**
- Automatic distribution (no RAID)
- Writes cached

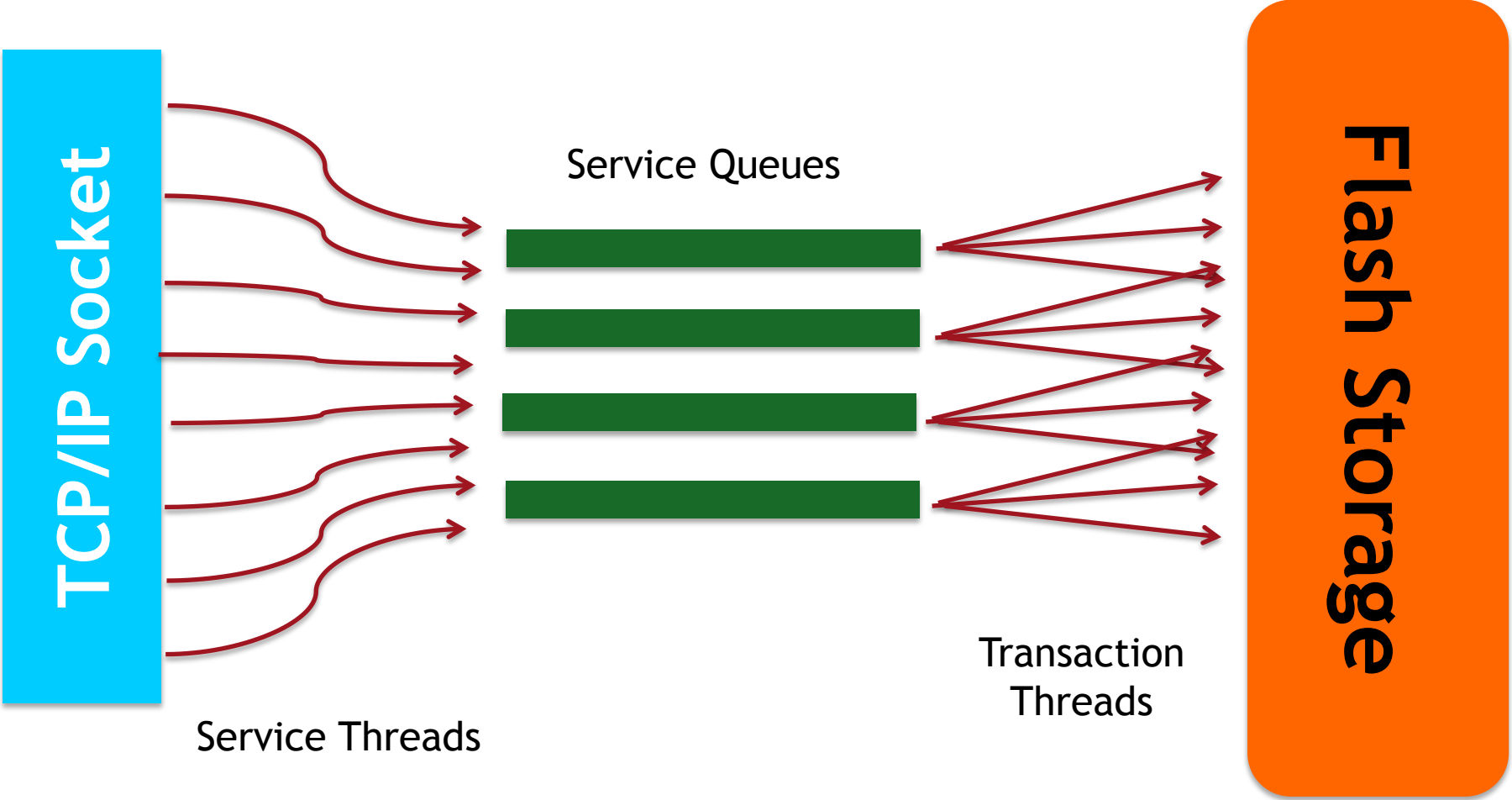


Copy on write – Log structured writes

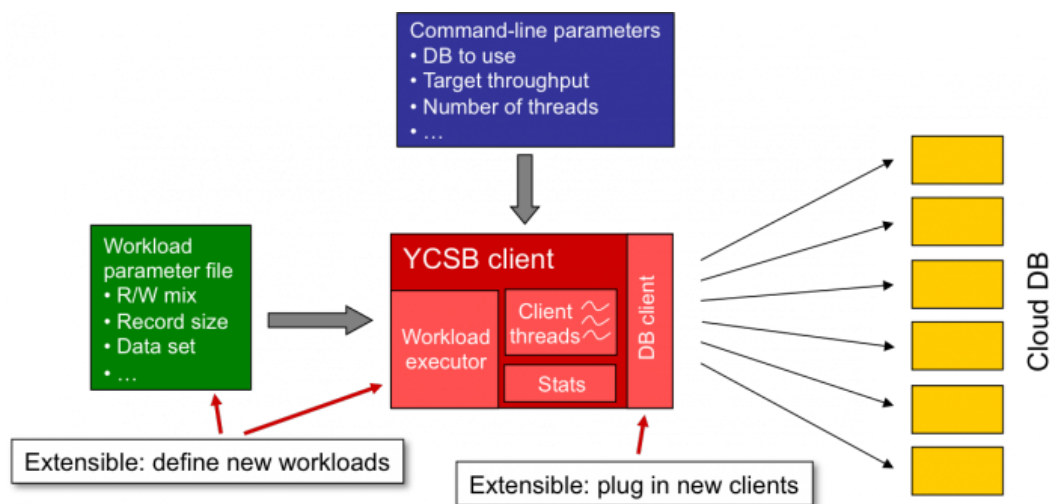
- Record is written to new block
 - Not written in place
 - Much faster
- Even wearing of Flash



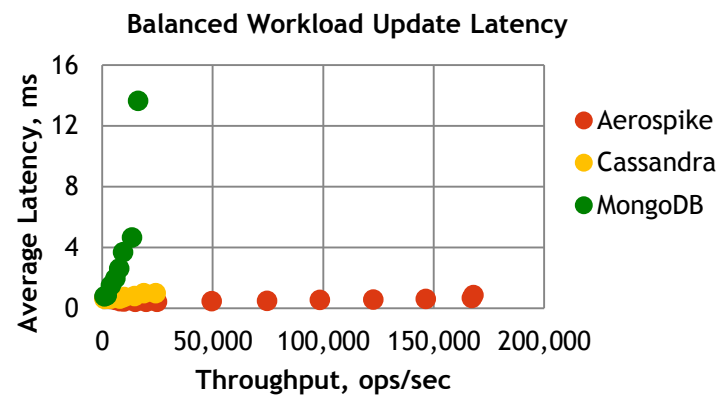
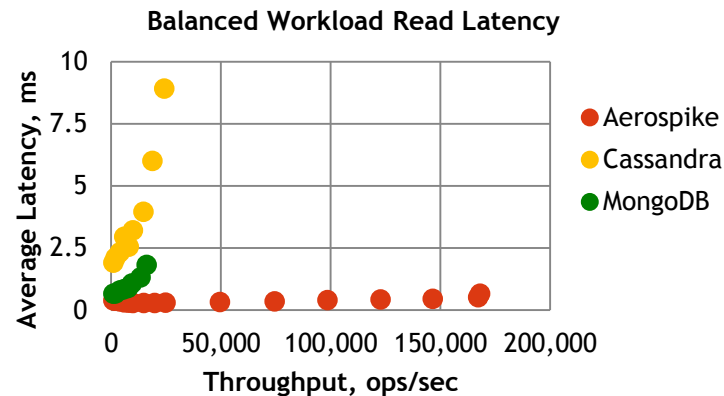
Service threads, Queues, Transaction threads



YCSB – Yahoo Cloud Serving Benchmark



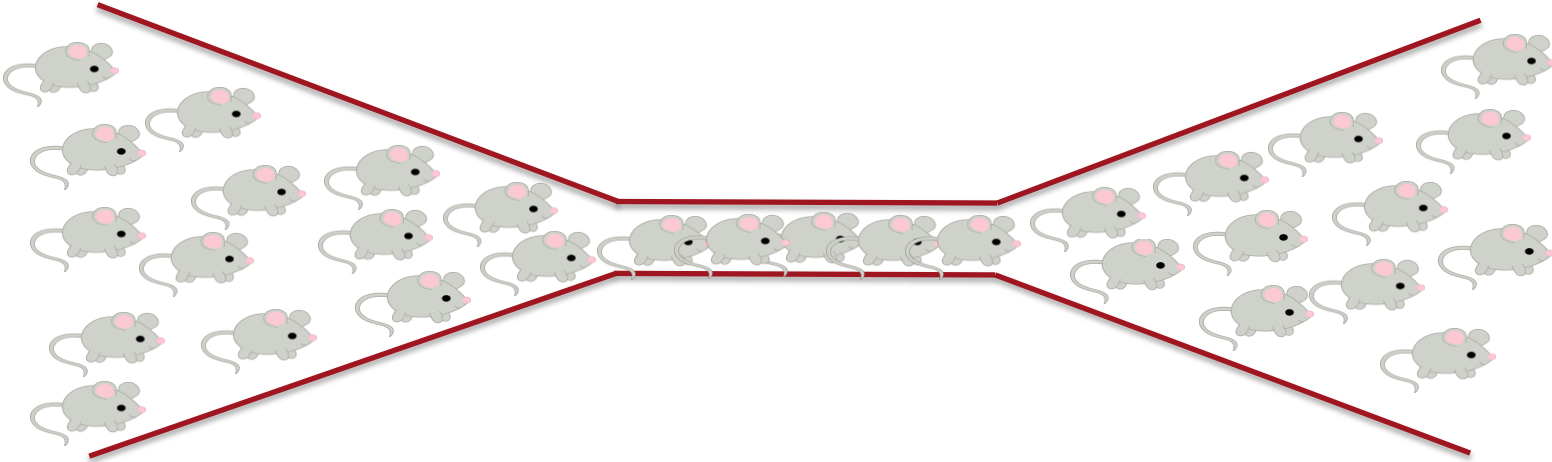
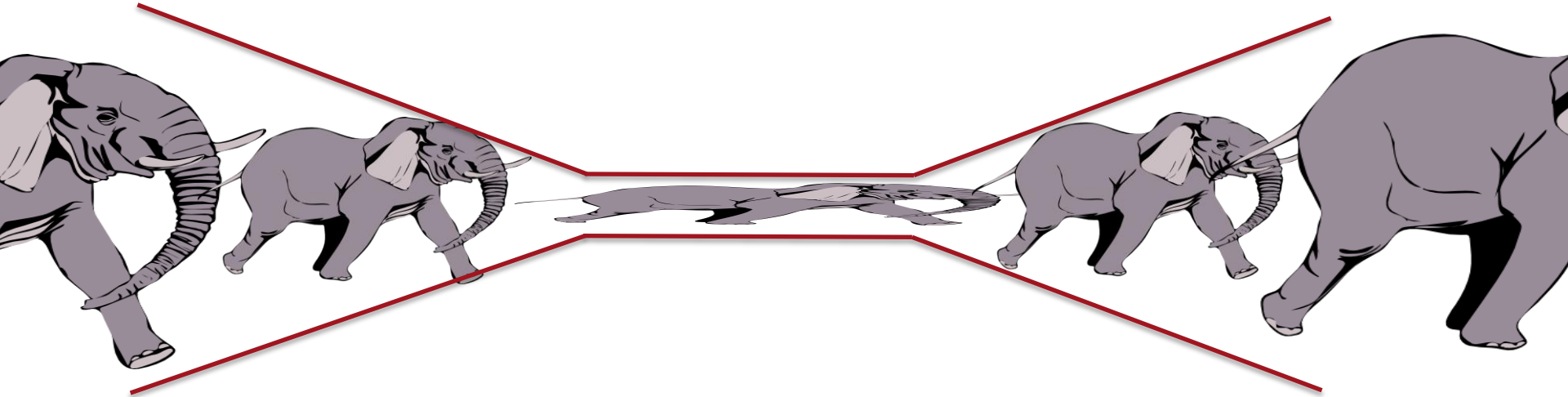
Throughput vs Latency



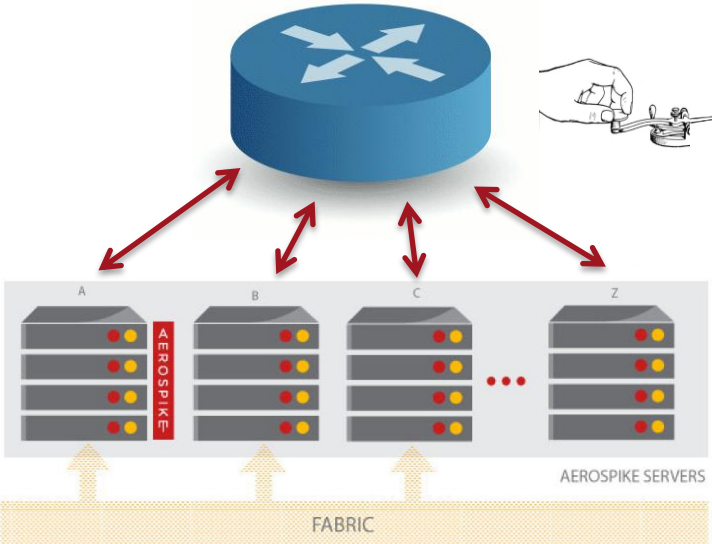
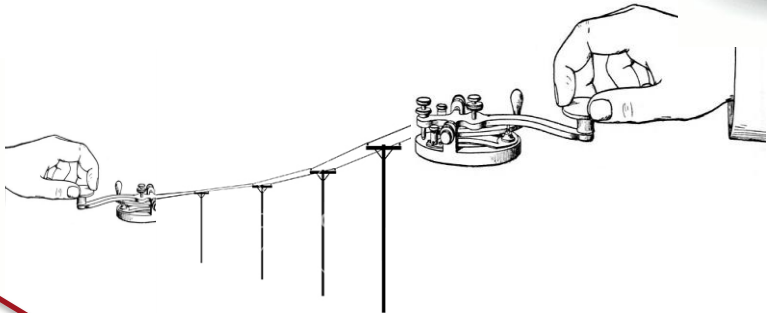
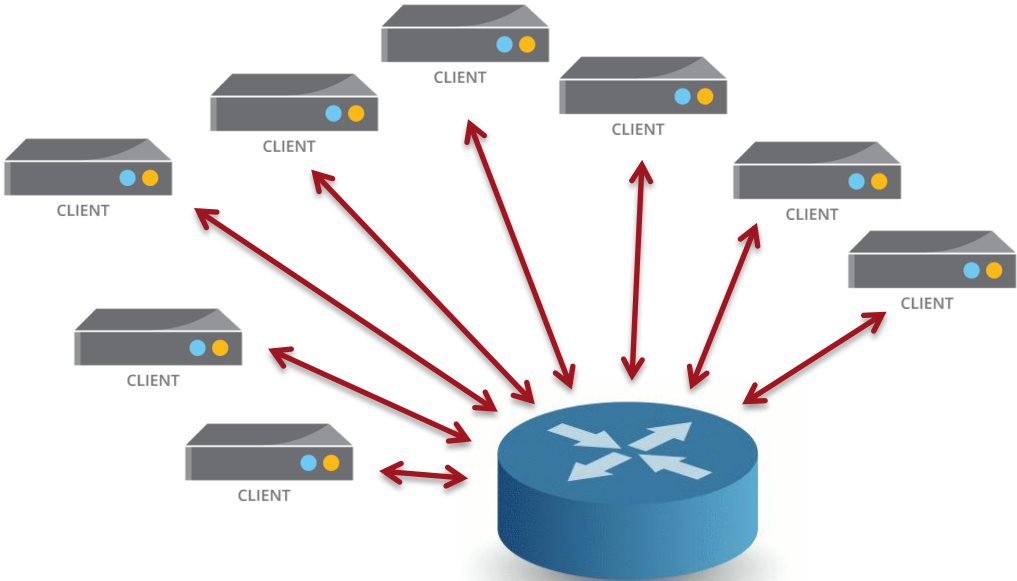


High load failures

Networking – Message size and frequency

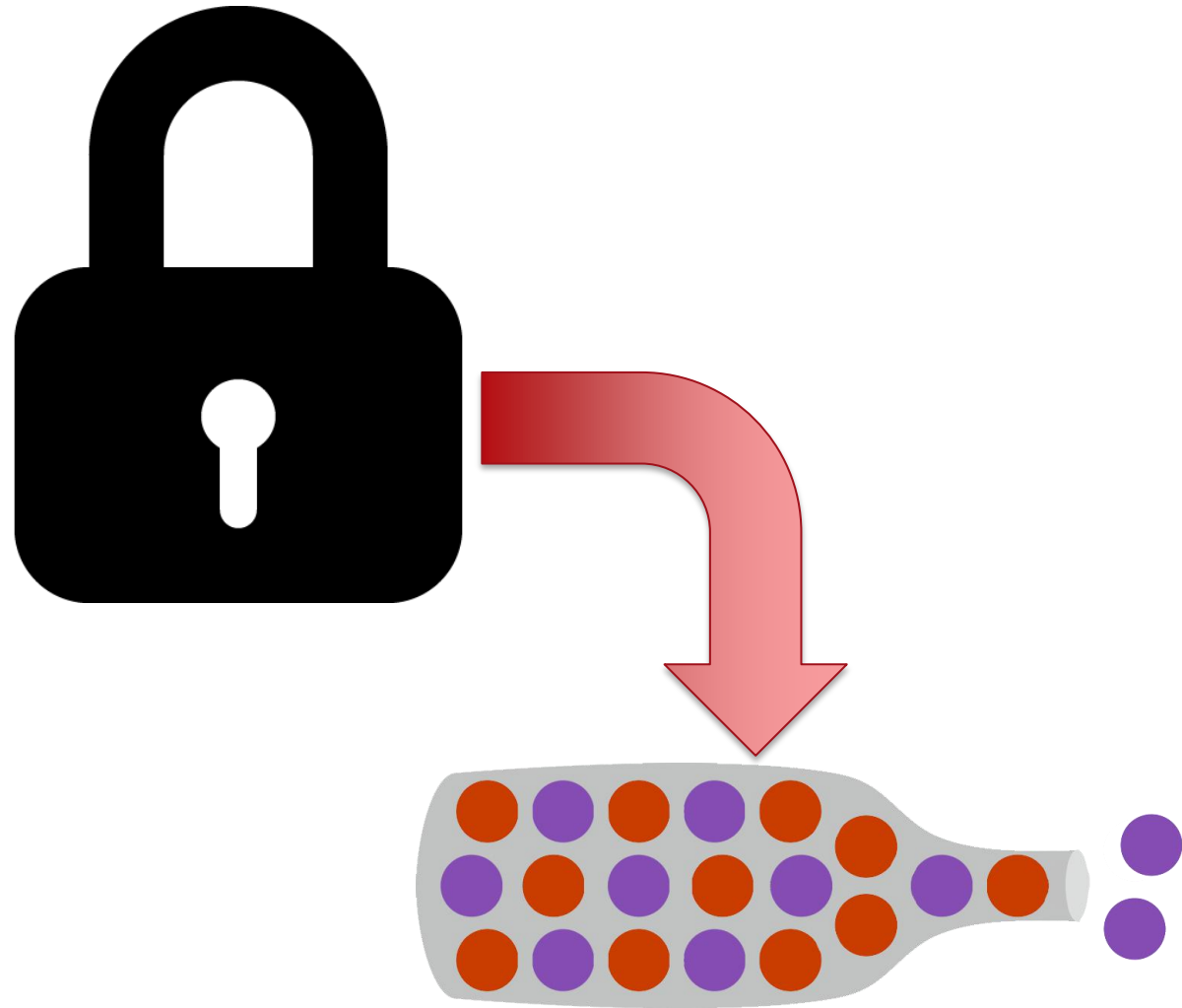


Networking - design



Big Locks

- Locks held for **too long**
- Increases latency
- Decreases concurrency
- Results in a **bottleneck**



Computing power not used

- **Network IRQ** not balanced across all Cores
 - 1 core does all the I/O
- Code does not use **multiple cores**
 - Single threaded
 - 1 core does all the processing
- **Uneven workload** on Cores
 - 1 core 90%, others 10%
- Code not **NUMA aware**
 - Using shared memory



Stupid code

- **1980's programmers** worried about
 - Memory, CPU cycles, I/Os
- **1990's programmers** worried about
 - Frameworks, Dogma, Style, **Fashion**



- **Stupid code**
 - Unneeded **I/Os**
 - Unneeded **object creation/destruction**
 - Poor **memory management**
 - Overworked GC
 - Malloc/Free
 - Loops within **loops within loops**
 - Unnecessary **recursion**
 - **Single tasked**/threaded
 - **Big locks**



Poor load testing

- BAA opened Heathrow's fifth terminal at a cost **of £4.3 billion**.
- Passengers had been promised a "**calmer, smoother, simpler airport experience**".
- The baggage system failed, **23,205** bags required manual sorting before being returned to their owners.



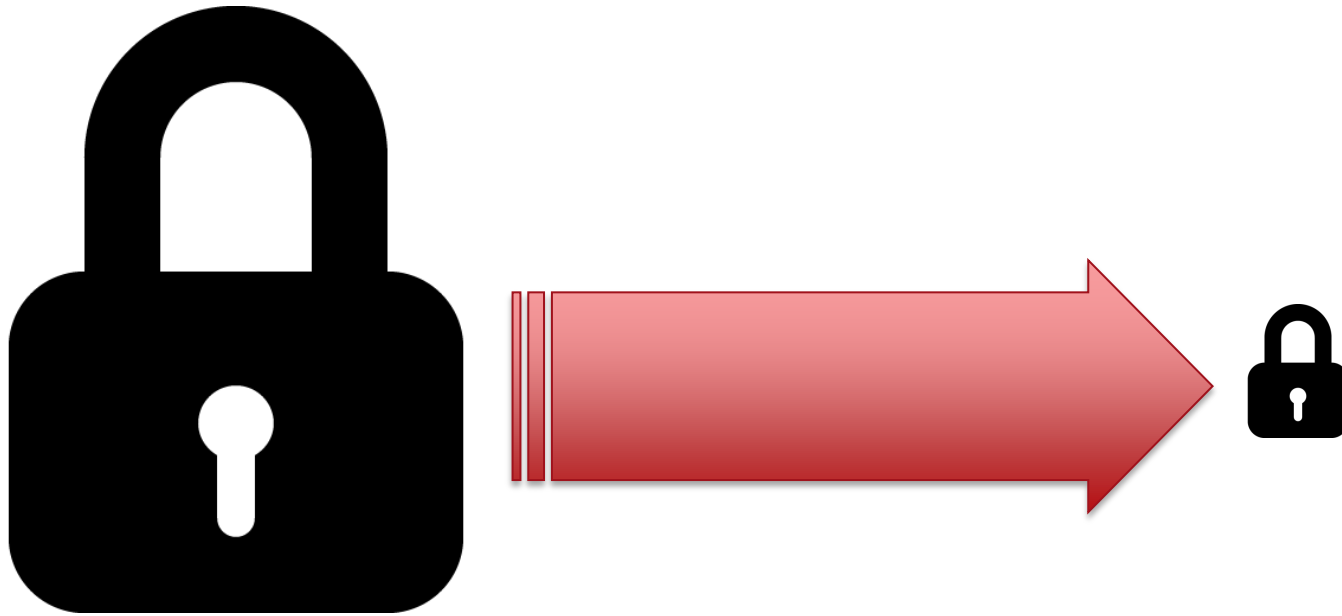


Uncle Pete's advice

Lock size

Make locks small

- Increase concurrency
- Reduce latency



Parallelism at every step

- Multiple machines
- Multiple cores
- Multiple Threads,
 - IRQ balancing
- Multi-channel Bus

CONCURRENCY



PARALLELISM



Efficient and robust partitioning

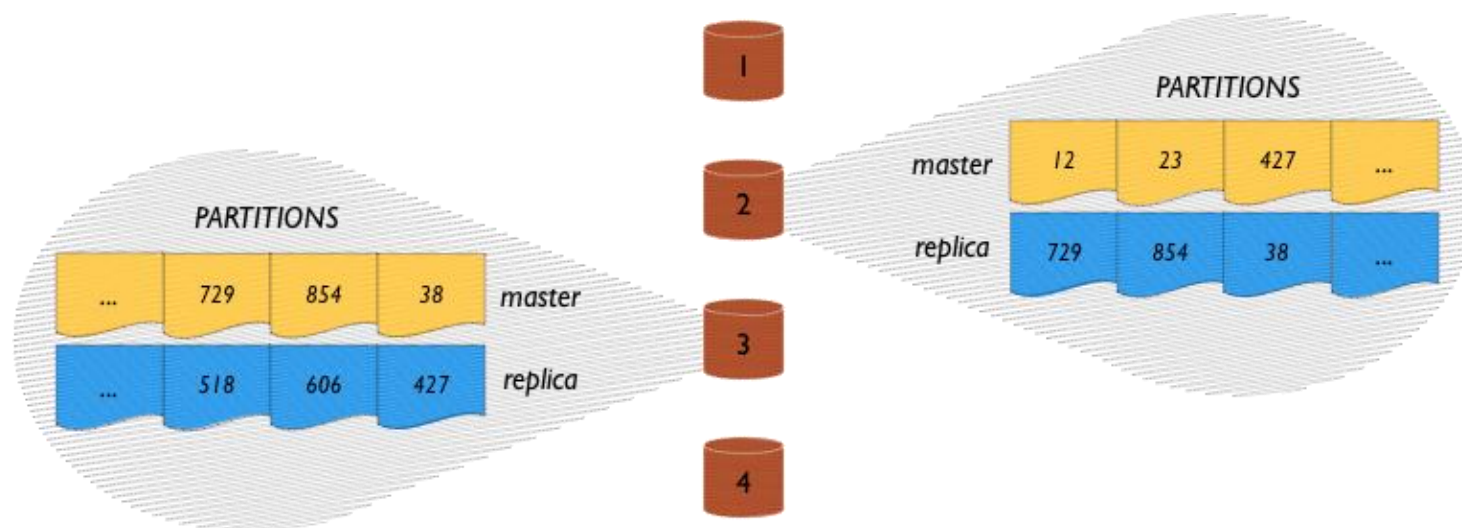
Partition your workload (Application) with

- Reliable, proven Algorithm

- No collisions
- No corner cases



```
03003802 996CB7BA 0EG0161B G0021C06
BA7CE203 G0030200 01208600 37D14D00
1B7125G0 024FG002 53D03C00 AD722500
1BD03C00 887525C1 01A07700 37D14D00
B7125G0 024FG002 53D03C00 AD722500
BD03C00 887525C1 4F553F00 53414241
F4F3D41 4242434E 3D4A6002 64692041
16C2F4F 553D4553 4142434E 4F3D414
425604 00312E30 042434E 0003424
003042 4C000000 024E4E4F 00B1D30
1254F1 21000009 8833B0CC 2957EE
3ECAA CB3EE8EF DF038D7F A14217
2AA4D 04143B75 4F571C83 535C00
7DED9 B57C659E C820EE07 FA49F
196DB 7D7F743D 9A36DD29 454E0
014D 410800C8 9A54E072 5A14C
```



Latency of your application

$$\text{Latency} = \text{Sum}(L_D) + \text{Sum}(L_S)$$

- L_D = Device latency
- L_S = Stupidity latency

- **Minimise** stupidity



Load test

- Simulation
 - Simulate real load
- Nothing is better than **real data**
 - Record live data and playback in testing



Finally..

A well designed and build application should

- Deliver the **correct** result
- **Perform** adequately
- Be **maintainable** by the average Guy or Girl

kérdések Questions
Perguntas Dúvidas
Klausimai Fragen
質問がありますか

AEROSPIKE