Distributed Systems

fallacies, philosophy and patterns for everyone

KC Braunschweig

SCaLE 21x - March 2024

A breadth first approach to a distributed systems knowledge

```
% whoami
kcb
```

School

University of Southern California

- School of Theatre
- School of Business (Information & Operations Management)

SCaLE volunteer since 1x

```
% whoami
kcb
```

Industry since 2005

- Ticketmaster.com - High speed/volume sales, large queues, unique inventory
- Edmunds.com
- Facebook/Meta since 2012

  Chef - configuration management at scale

  Scribe - log aggregation and stream processing

  Apache Zookeeper - coordination infrastructure

  Public cloud infrastructure - all the things

There are only 2 hard problems in Computer Science:

Phil Karlton

There are only 2 hard problems in Computer Science:

- cache invalidation
- naming things

Phil Karlton

There are only 2 hard problems in Computer Science:

- cache invalidation
- naming things
- off-by-one errors

There are only 2 hard problems in Computer Science:

- cache invalidation - computationally difficult
- naming things - people problems
- off-by-one errors - bugs, solar flares, weird stuff

FAANG interviews and leetcode are not good indicators of real world problem solving

Agenda

fallacies of distributed computing

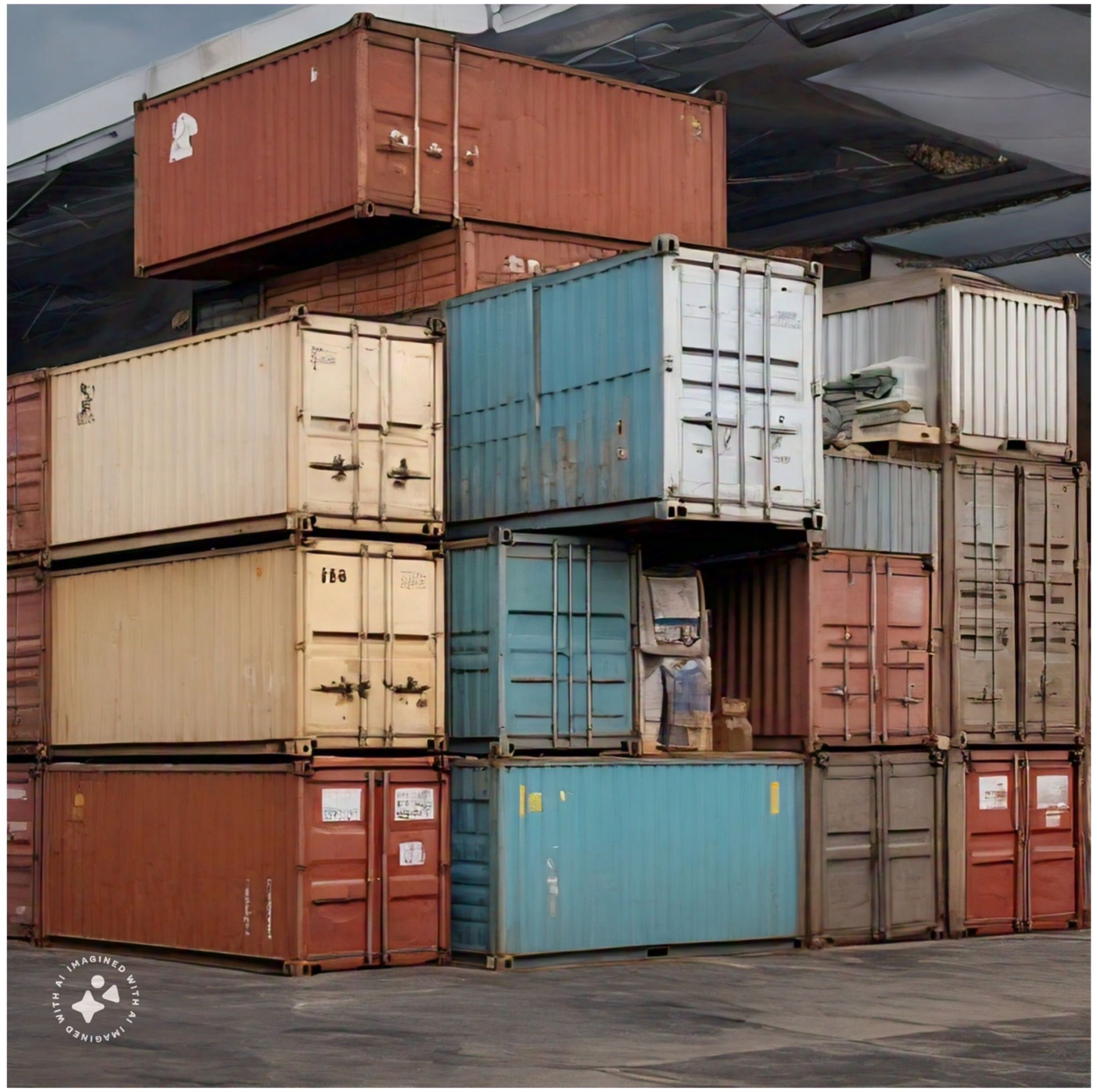
philosophy

algorithms and patterns

now what?

01 fallacies of distributed computing

Everything is a distributed system



https://imagine.meta.com/

The fallacies of distributed computing are a set of assertions made by L Peter Deutsch and others at Sun Microsystems describing false assumptions that programmers new to distributed applications invariably make.

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

The fallacies

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

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

The fallacies are spherical cows

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

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

The fallacies are spherical cows

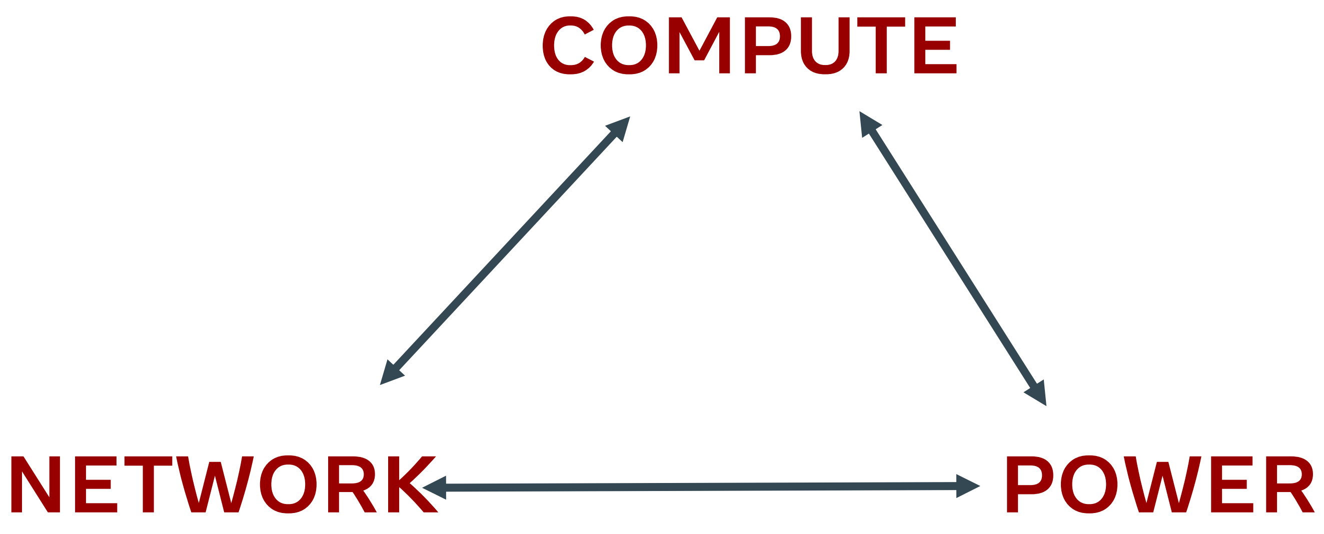
1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Spherical cows are a trade off not a mistake

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

System Optimization - the bottleneck cycle

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous



COMPUTE

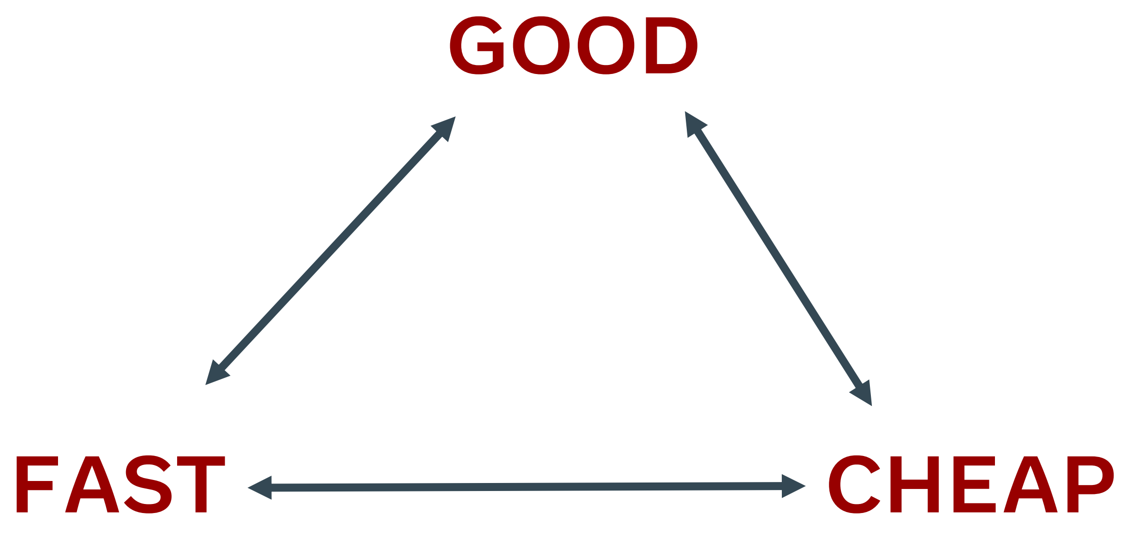
NETWORK

POWER

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

Project Management - the Iron Triangle

Pick 2 - you probably have to pick cheap



02 philosophy

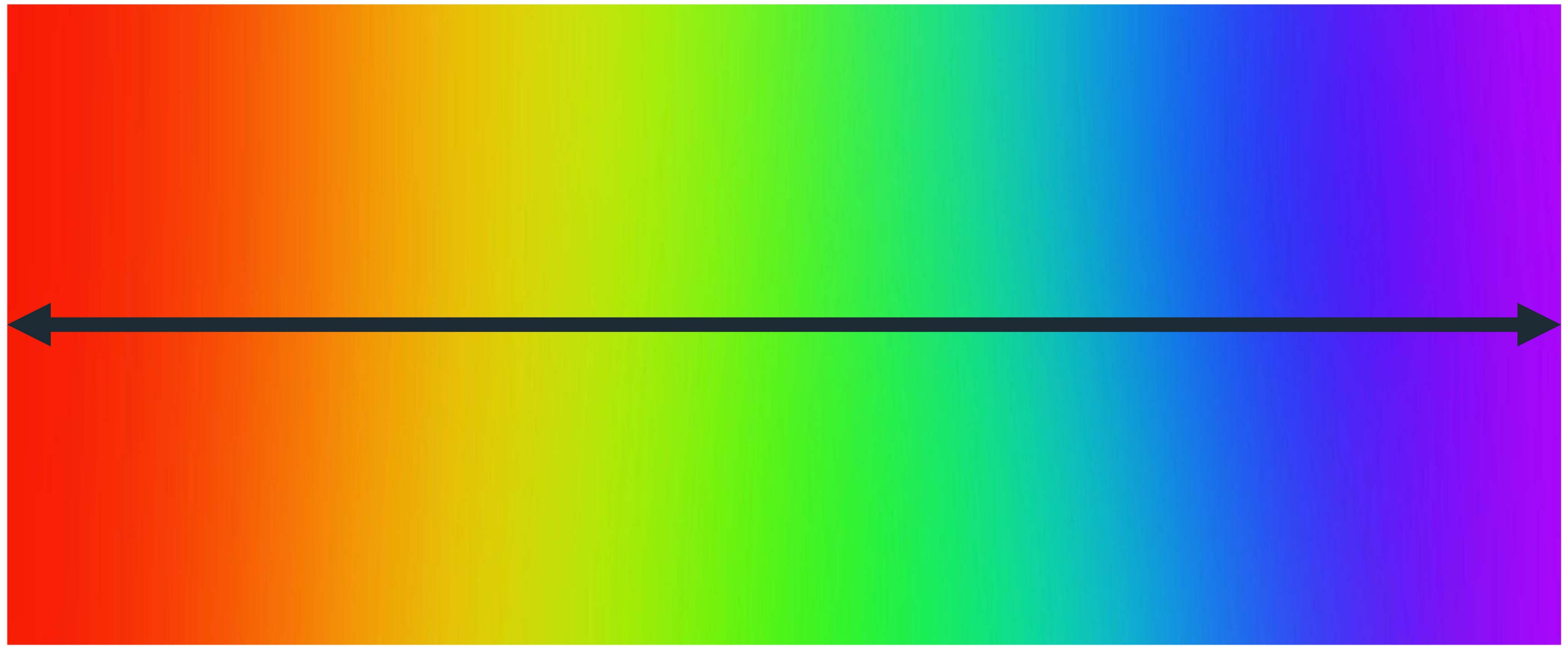
philosophy

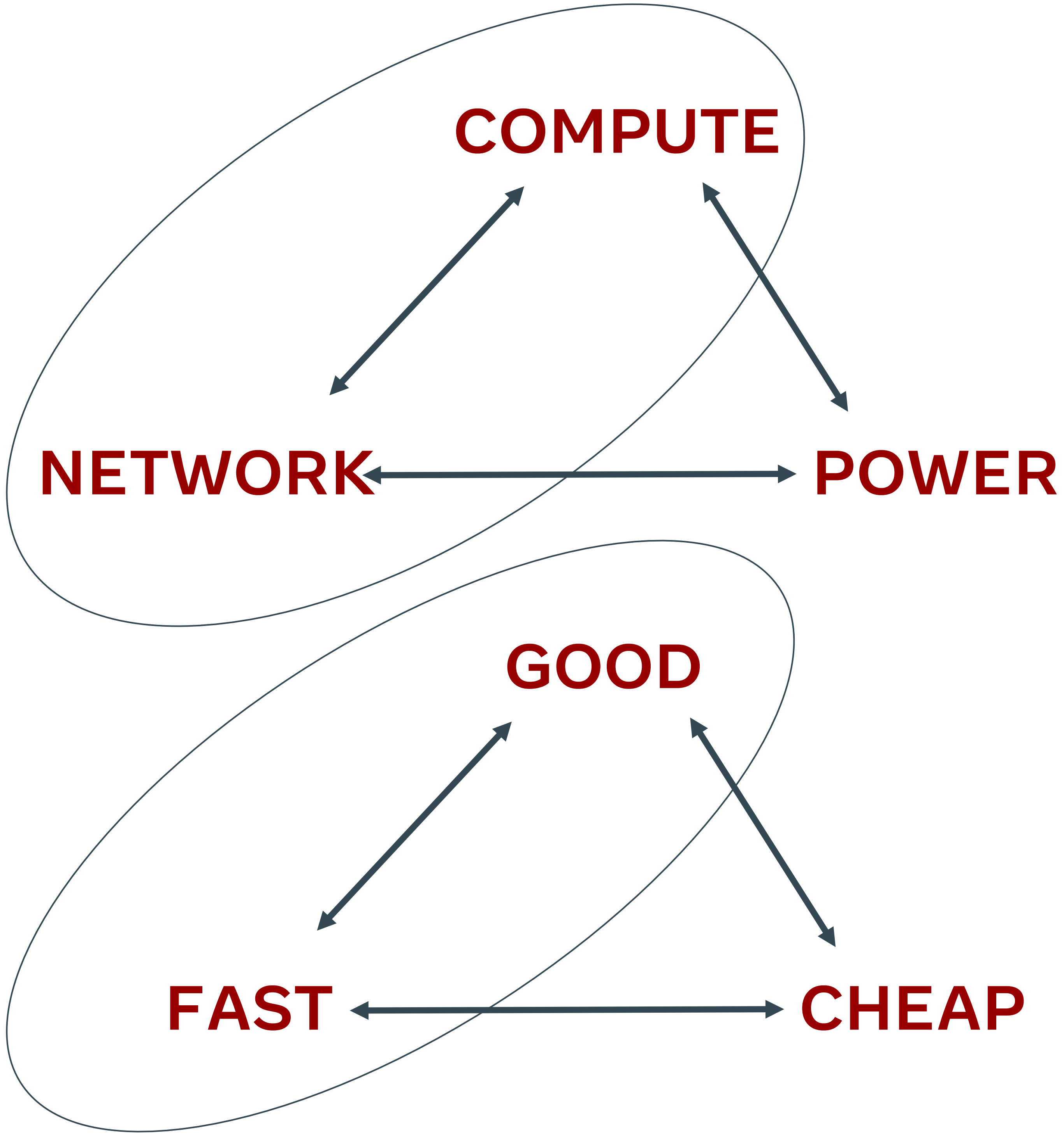
Everything is a trade-off

philosophy

Everything is a trade-off on a spectrum

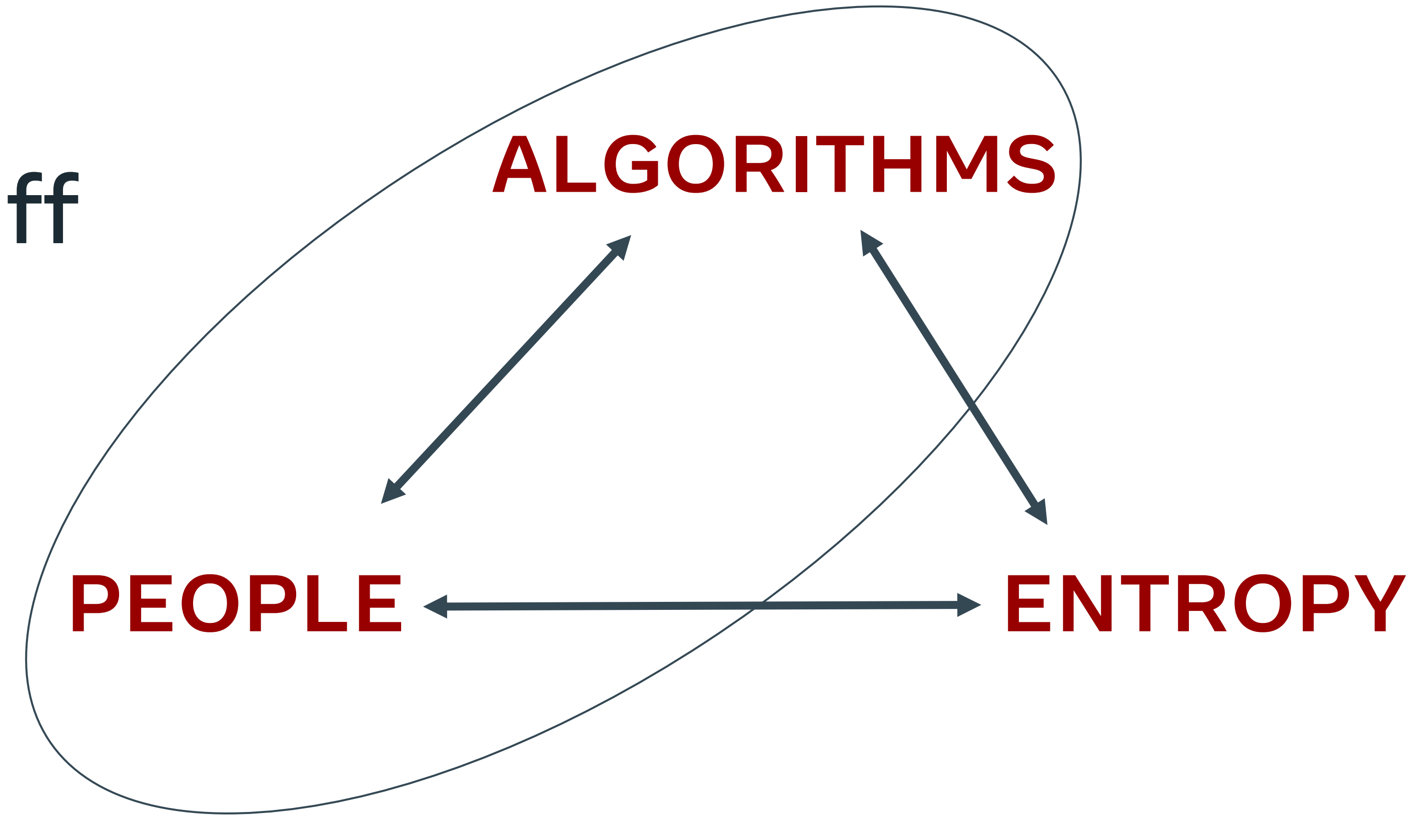
The third thing is probably imposed on you





There are only 2 hard problems in Computer Science:

- cache invalidation - computationally difficult
- naming things - people problems
- off-by-one errors - bugs, weird stuff



How complex systems fail - Richard I. Cook MD

Being a Short Treatise on the Nature of Failure; How Failure is Evaluated; How Failure is Attributed to Proximate Cause; and the Resulting New Understanding of Patient Safety

Why is a doctor’s understanding of patient safety so relevant to us?

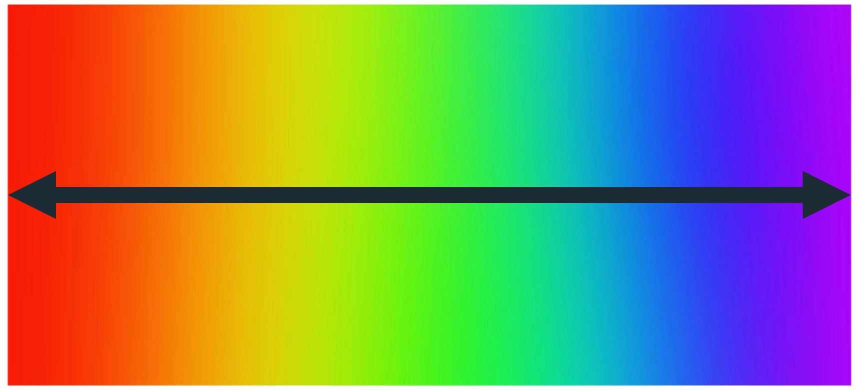
https://how.complexsystems.fail/

Why is a doctor’s understanding of patient safety so relevant to us?

small scale tempts us with the myth of our own (or other’s) intelligence

distributed systems force us to face reality

scale brings inertia we can’t cheat



command & control vs cooperation

push vs pull

imperative vs declarative

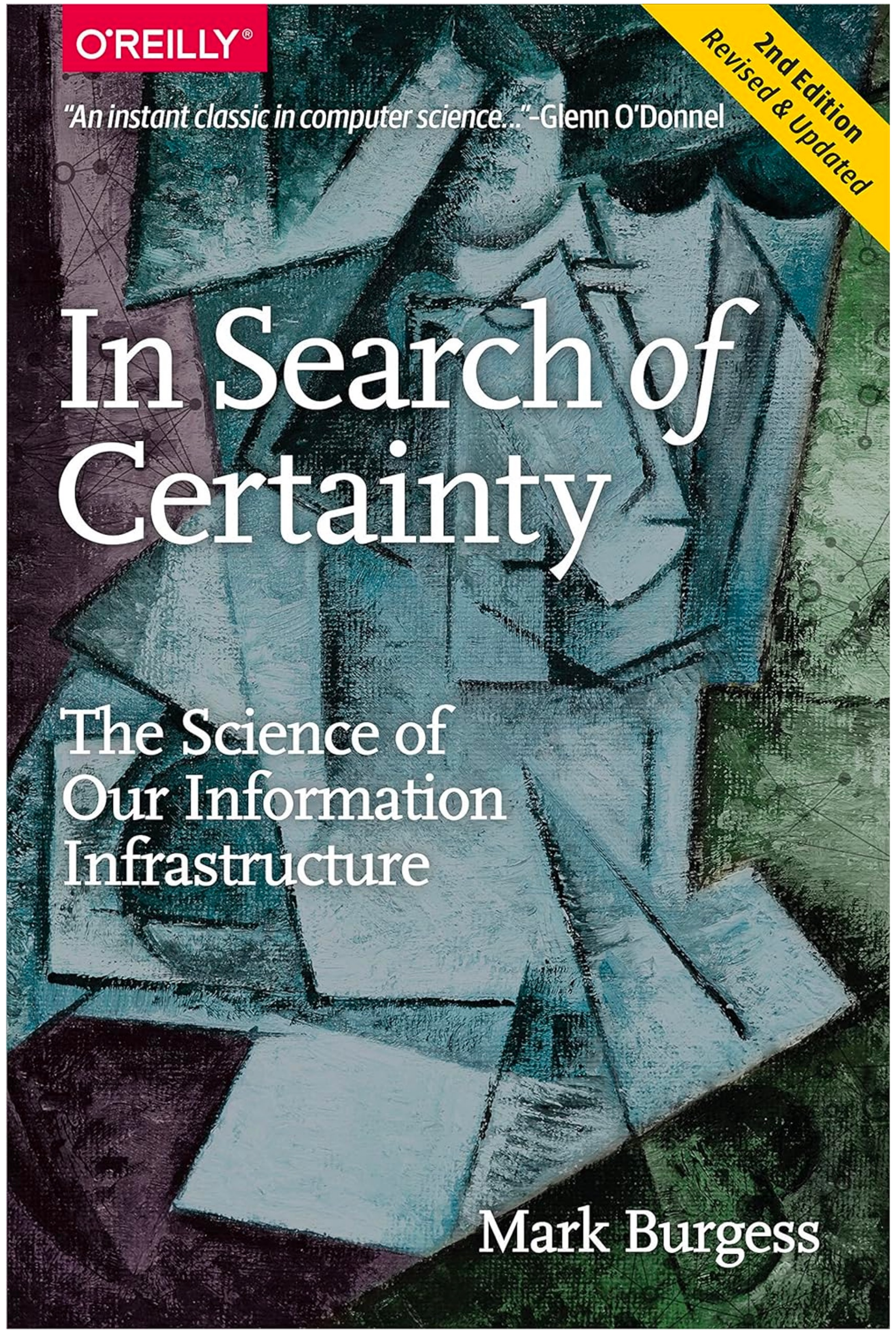
Mark Burgess - Promise Theory

Adam Jacob - Chef

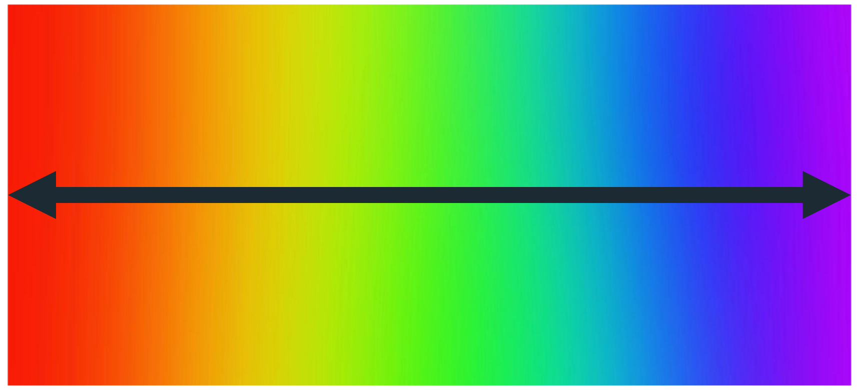
configuration management

command & control vs cooperation

push vs pull


O'REILLY
"An instant classic in computer science..."-Glenn O'Donnel
2nd Edition
Revised & Updated
In Search of Certainty
The Science of Our Information Infrastructure
Mark Burgess

https://www.amazon.com/Search-Certainty-Science-Information-Infrastructure-ebook/dp/B00WL6SPR6

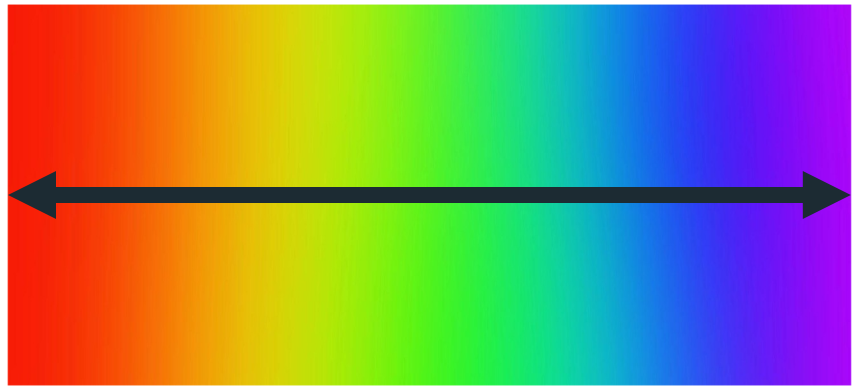


command & control vs cooperation

push vs pull

imperative vs declarative

certainty vs entropy?

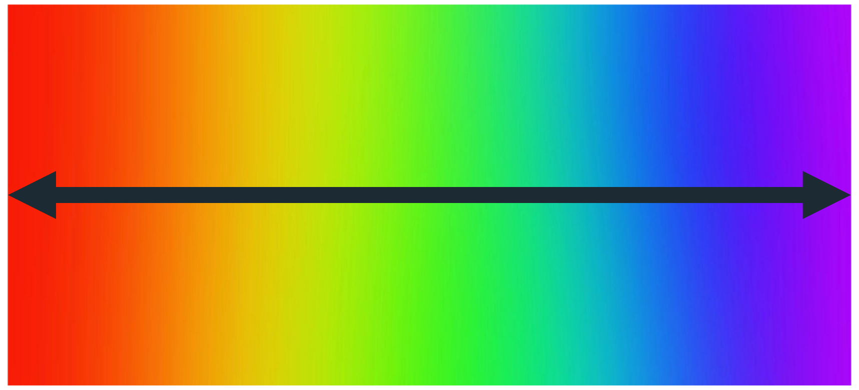


command & control vs cooperation

push vs pull

imperative vs declarative

~~certainty vs entropy~~
the myth of certainty vs acceptance that entropy is not optional



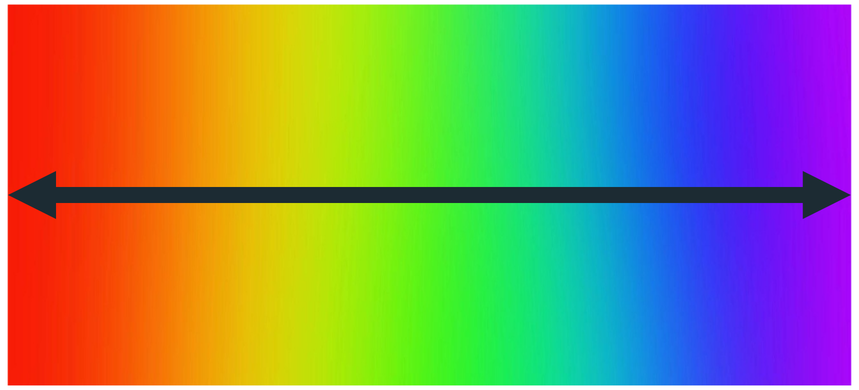
command & control vs cooperation

push vs pull

imperative vs declarative

~~certainty vs entropy~~
the myth of certainty vs acceptance that entropy is not optional

speed vs completeness



command & control vs cooperation

push vs pull

imperative vs declarative

~~certainty vs entropy~~
the myth of certainty vs acceptance that entropy is not optional

speed vs completeness

security vs usability

Time is an illusion
Lunchtime doubly so

Douglas Adams

time is linear

time is monotonically increasing

time is ~~linear~~

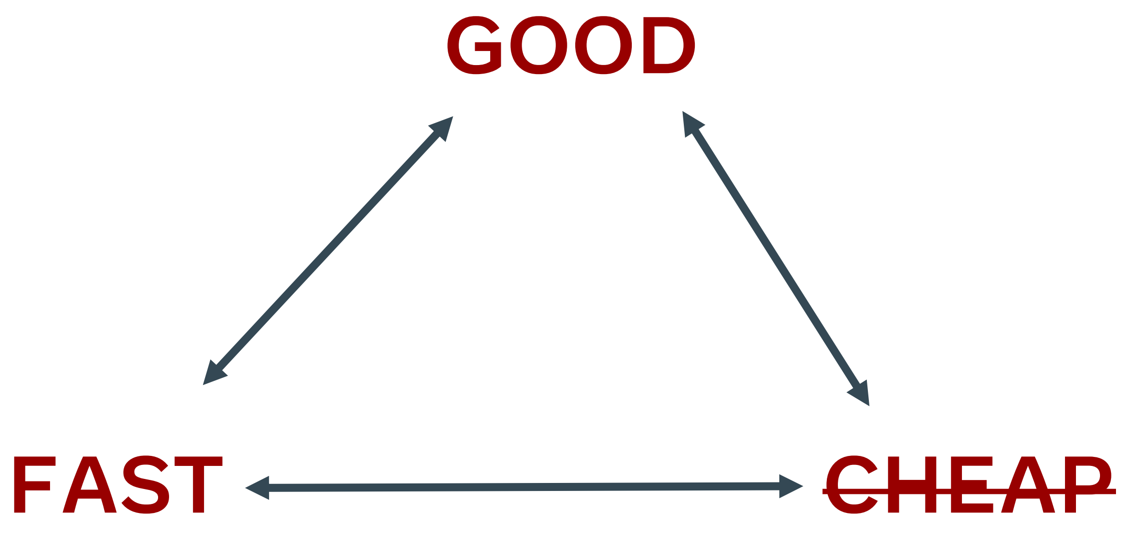
time is ~~monotonically increasing~~
leap seconds, DST, vm snapshots, clock skew
choices

time is ~~linear~~

time is ~~monotonically increasing~~
leap seconds, DST, vm snapshots, clock skew
choices

Spanner uses the Paxos algorithm as part of its operation to shard (partition) data across up to hundreds of servers. It ***makes heavy use of hardware-assisted clock synchronization using GPS clocks and atomic clocks to ensure global consistency****. TrueTime is the brand name for Google's distributed cloud infrastructure, which provides Spanner with the ability to generate monotonically increasing timestamps in data centers around the world.*

https://en.wikipedia.org/wiki/Spanner_(database)



03 algorithms and patterns

we’re taking a wikipedia-level view of algorithms

the point is to recognize the pattern, not learn the math

dive deep in the math if you really need it

Big O notation

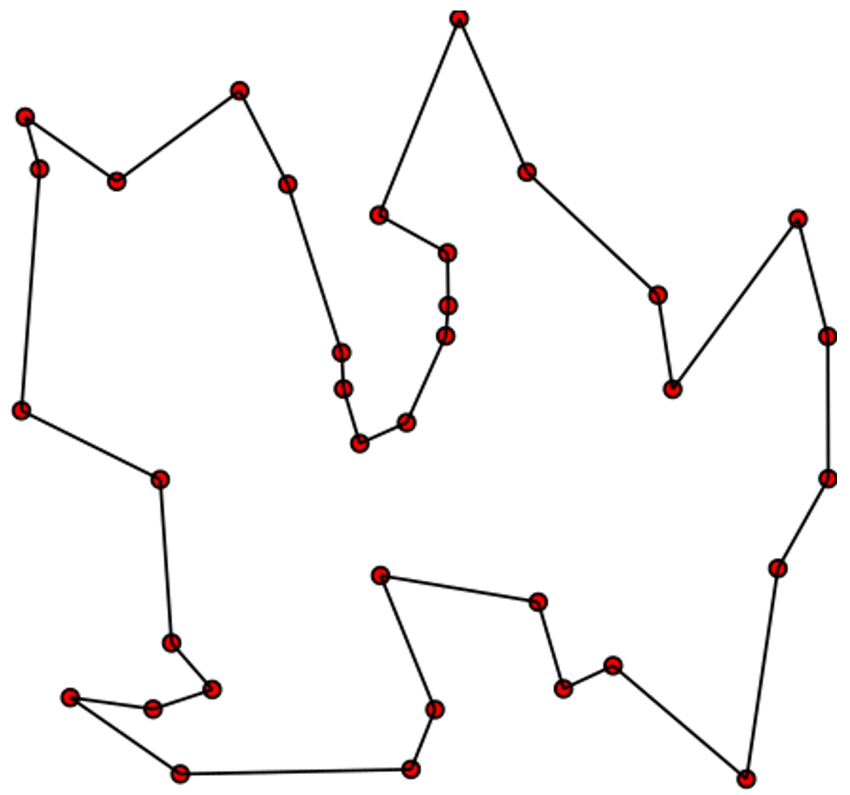
Just a notation for describing algorithmic complexity

$O(1)$

$O(n)$

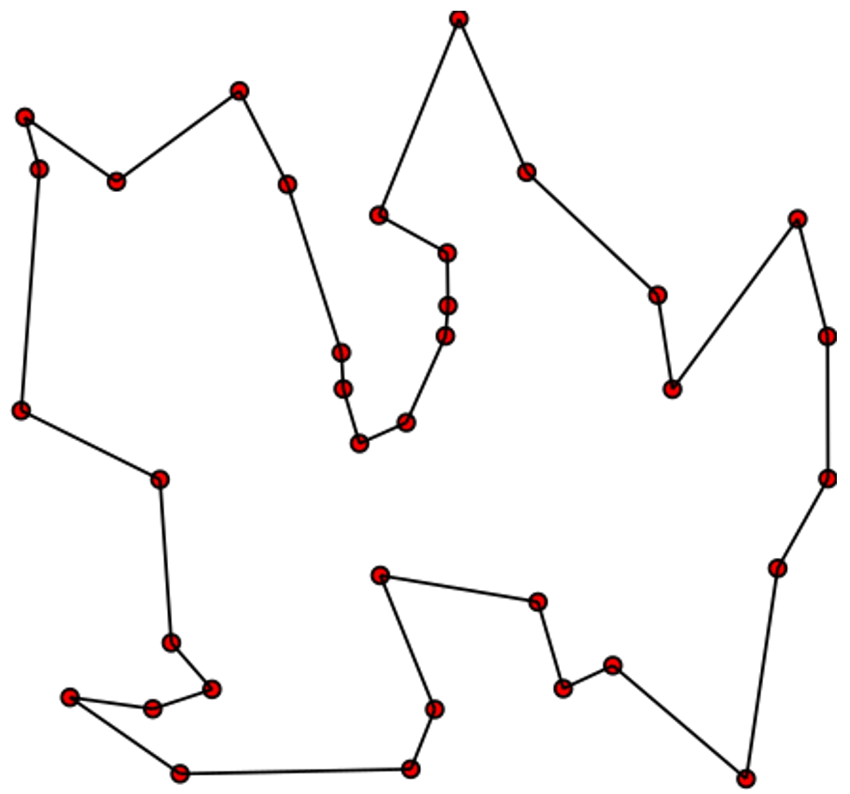
$O(n^2)$

$O(\log n)$

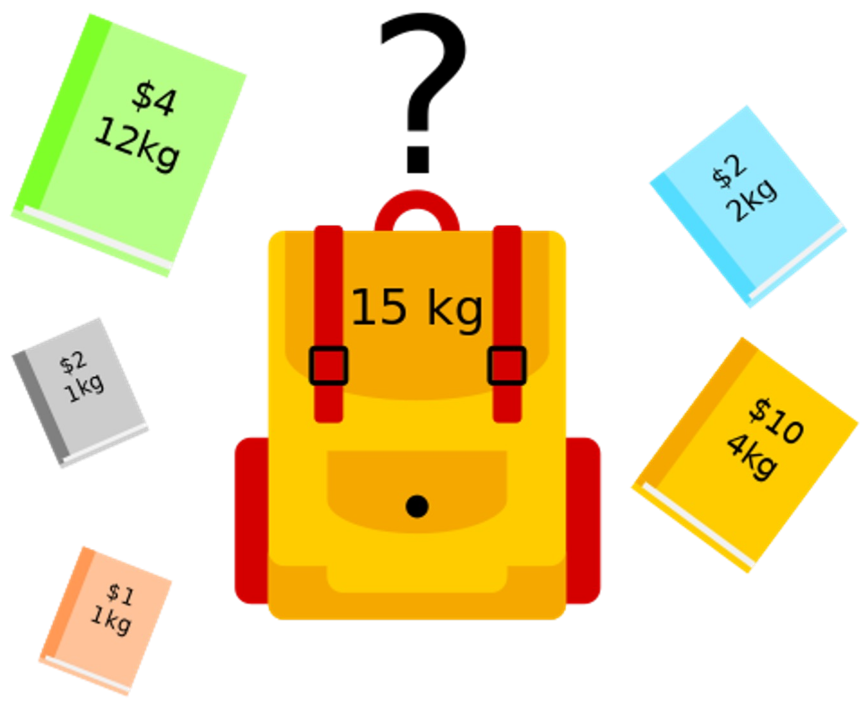


traveling salesman problems

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

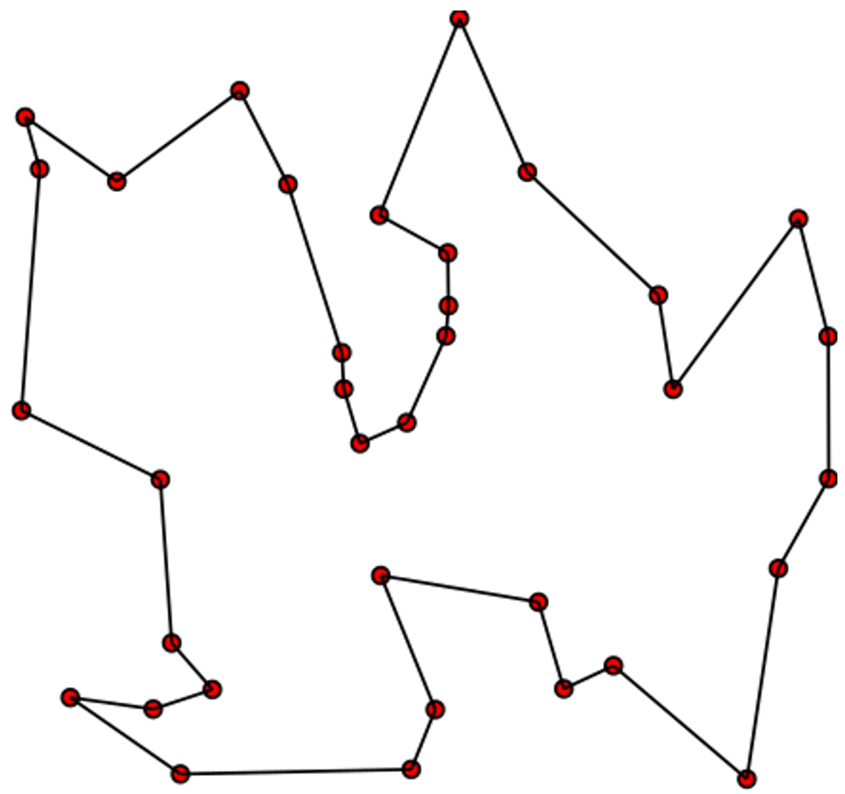


traveling salesman problems

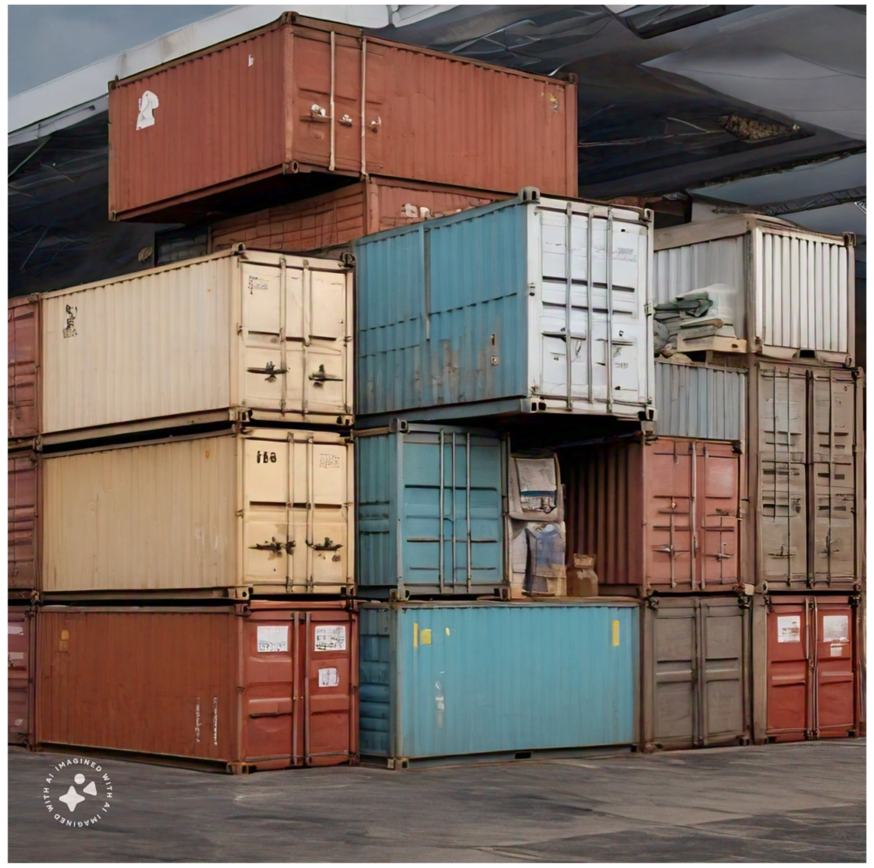


knapsack problems

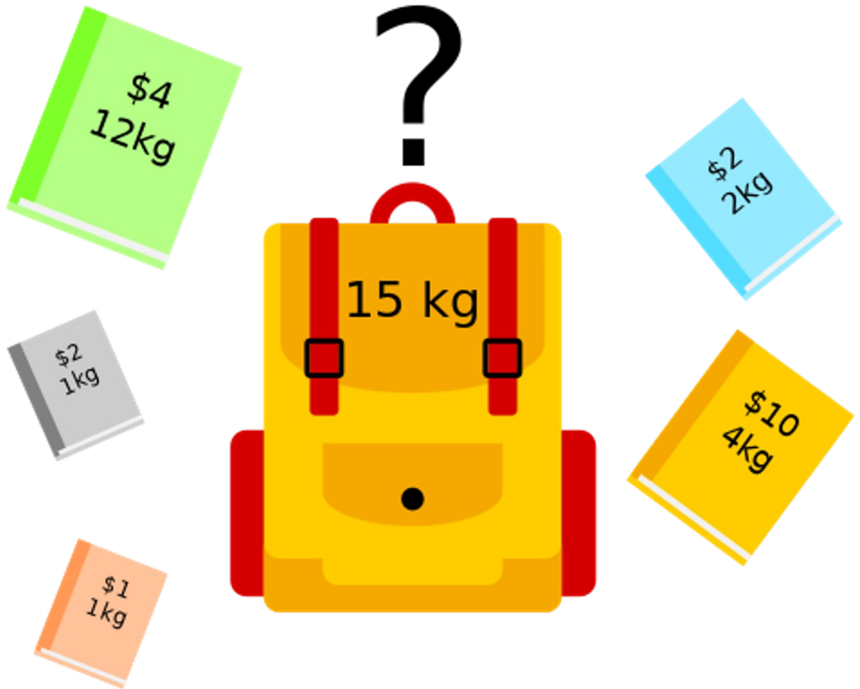
https://en.wikipedia.org/wiki/Knapsack_problem



traveling salesman problems

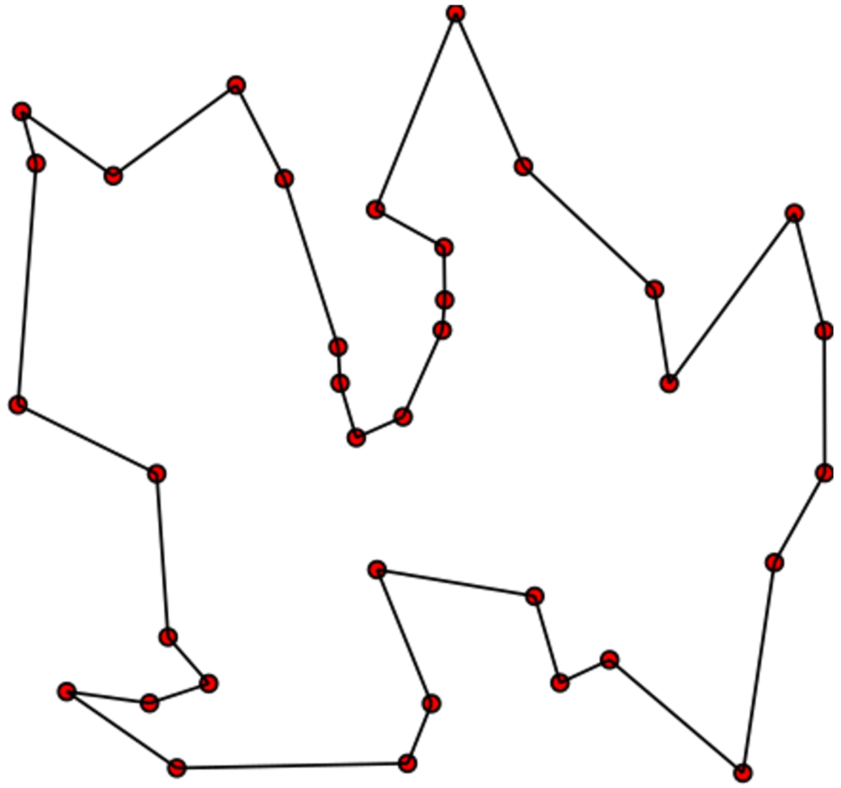


bin packing problems



knapsack problems

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



traveling salesman problems

P vs NP

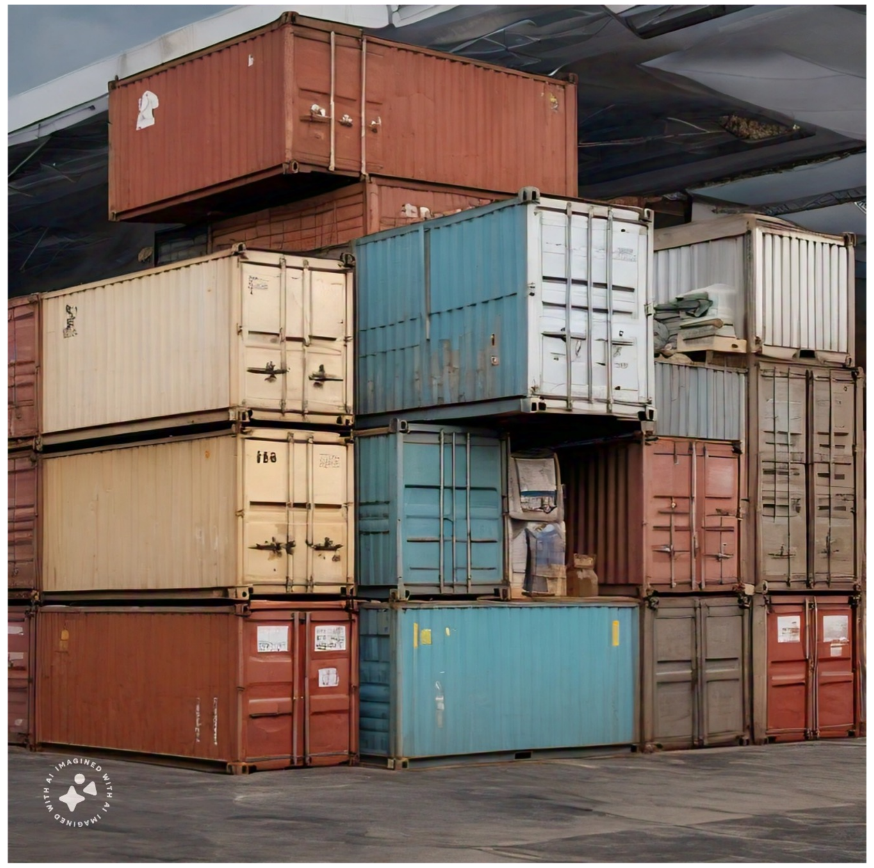
NP hard problems

NP complete problems

1 of 7 Millenium Prize problems

$4
12kg
?
$2
2kg
15 kg
$2
1kg
$10
4kg
$1
1kg

knapsack problems



bin packing problems

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

P vs NP

COPY! STEAL! CHEAT!

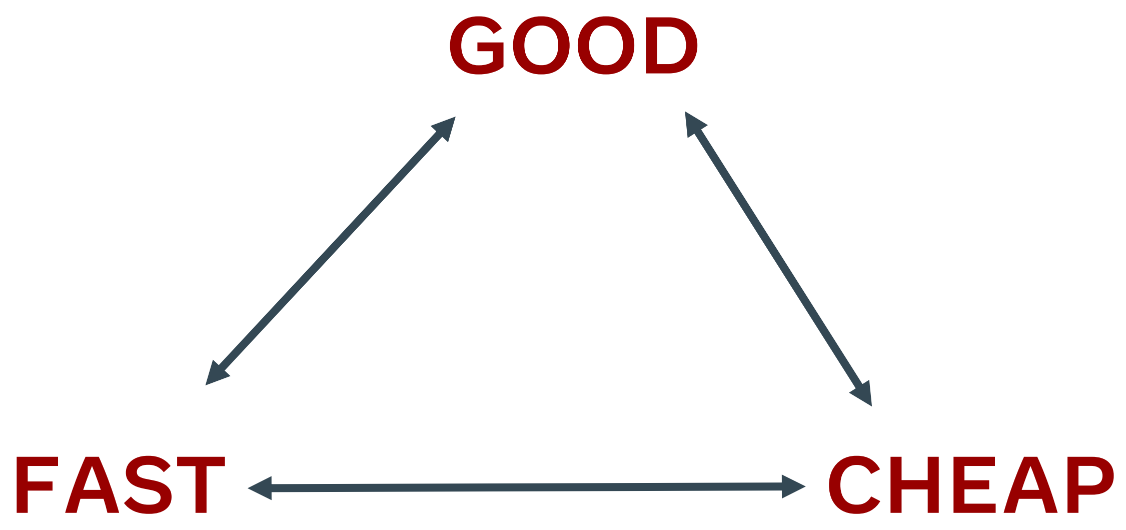
some constrained versions are solved or approximated

utilize requirements gathering

identify constraints

you get the change the problem AND the solution

academics solve generic problems, you’re solving a specific problem speed vs completeness



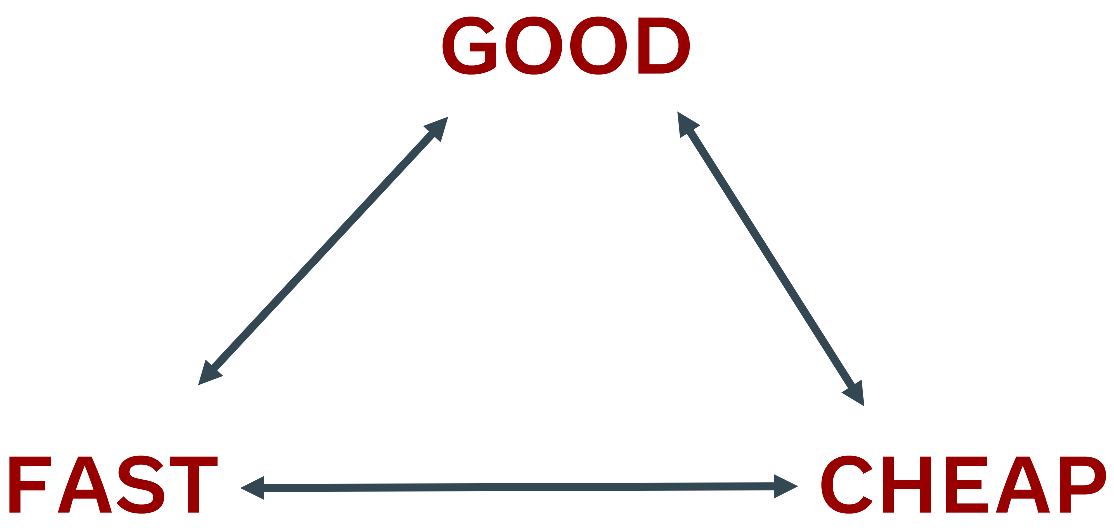
CAP Theorem

pick 2 and you must pick partition tolerance

Consistency

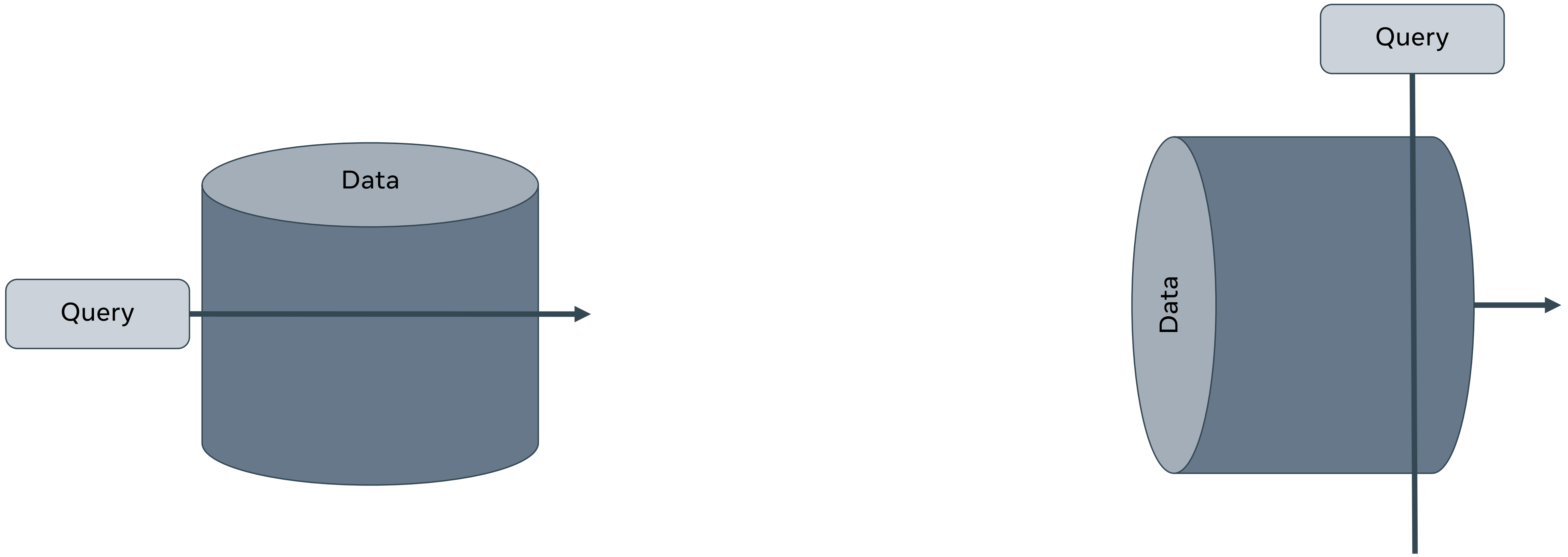
Availability

Partition Tolerance



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

database query vs stream processing



consider:

command & control vs cooperation

push vs pull

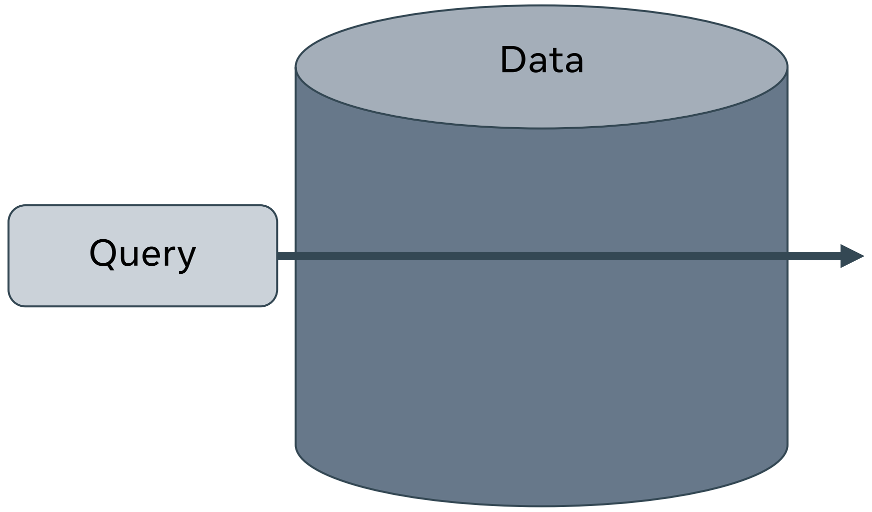
imperative vs declarative

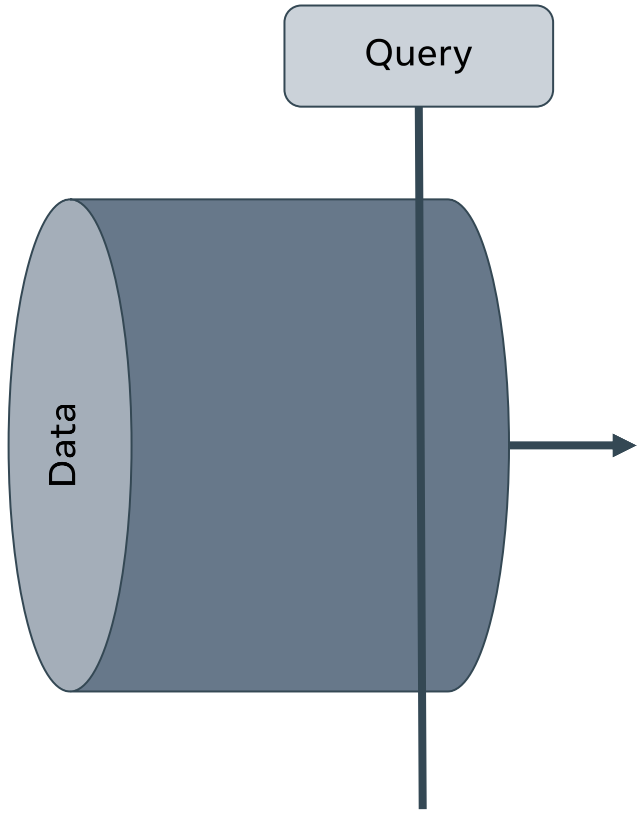
speed vs completeness

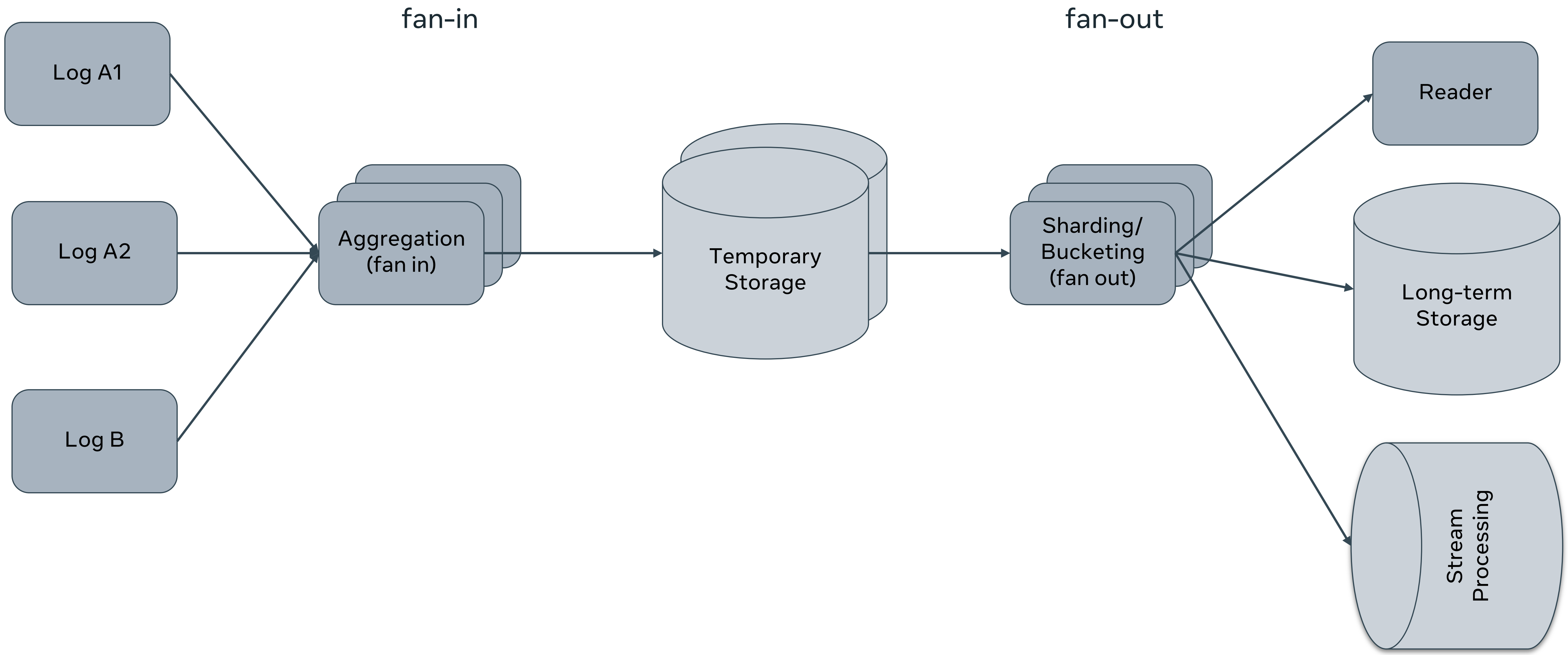
time

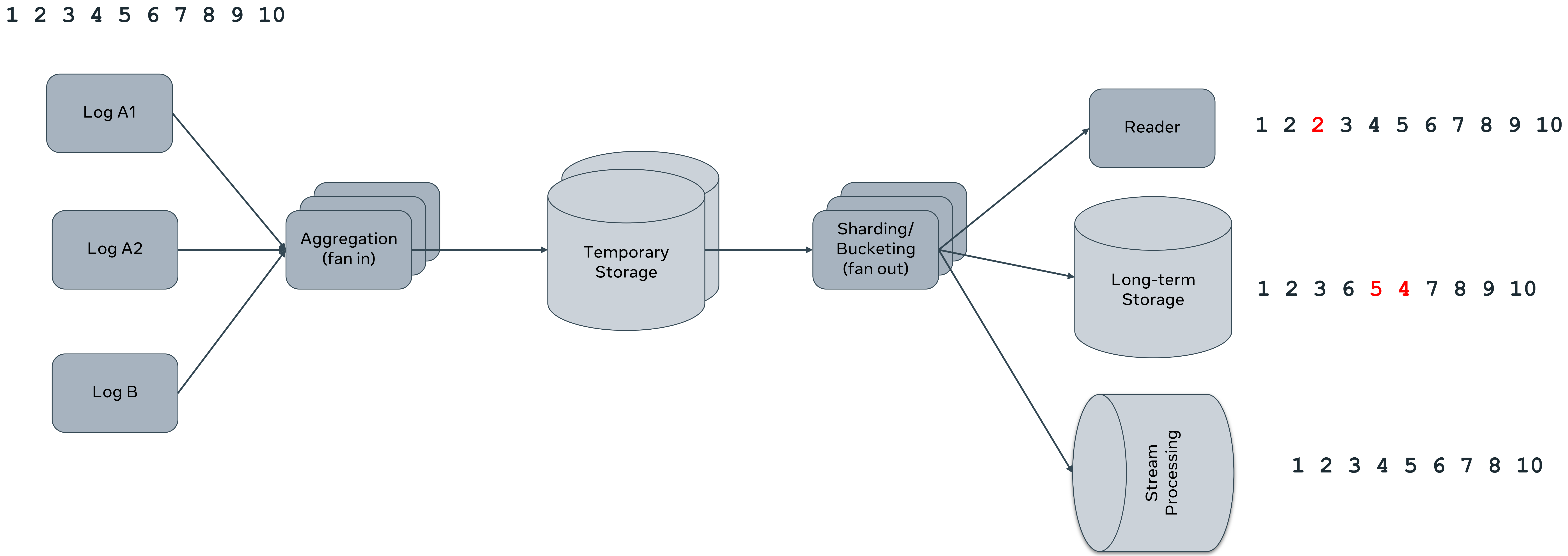
scale

database query vs stream processing






fan-in
fan-out
Log A1
Log A2
Log B
Aggregation (fan in)
Temporary Storage
Sharding/ Bucketing (fan out)
Reader
Long-term Storage
Stream Processing



1 2 3 4 5 6 7 8 9 10

Log A1

Log A2

Log B

Aggregation (fan in)

Temporary Storage

Sharding/ Bucketing (fan out)

Reader

1 2 2 3 4 5 6 7 8 9 10

Long-term Storage

1 2 3 6 5 4 7 8 9 10

Stream Processing

1 2 3 4 5 6 7 8 10

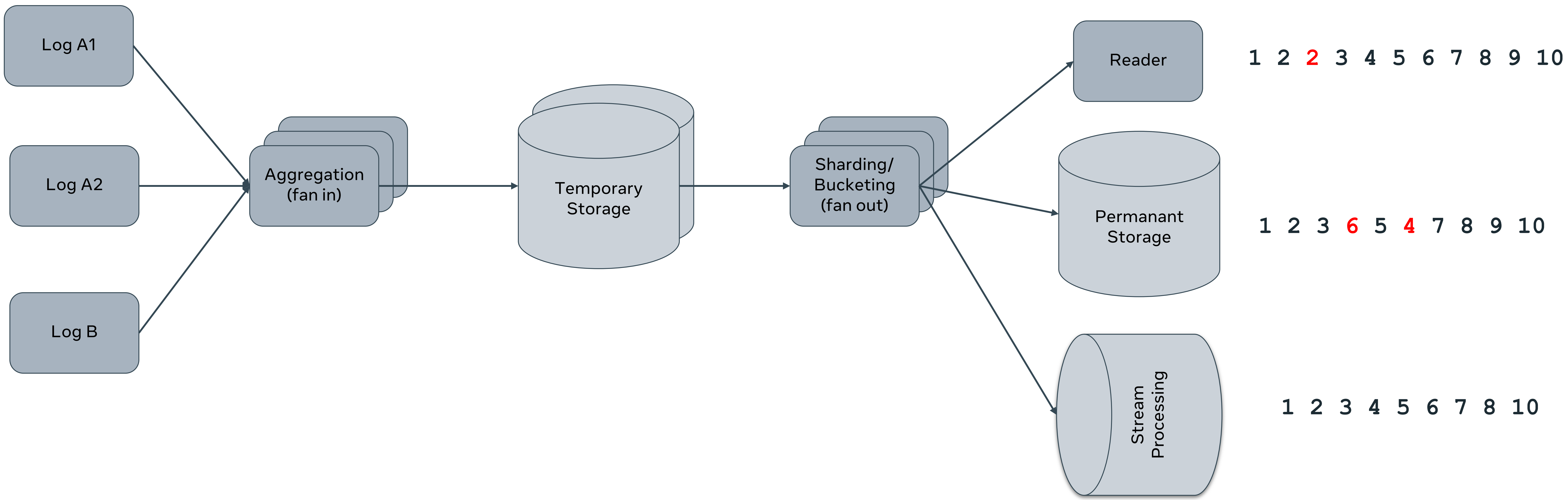
consider:

speed vs completeness

time

scale

1 2 3 4 5 6 7 8 9 10



consider:

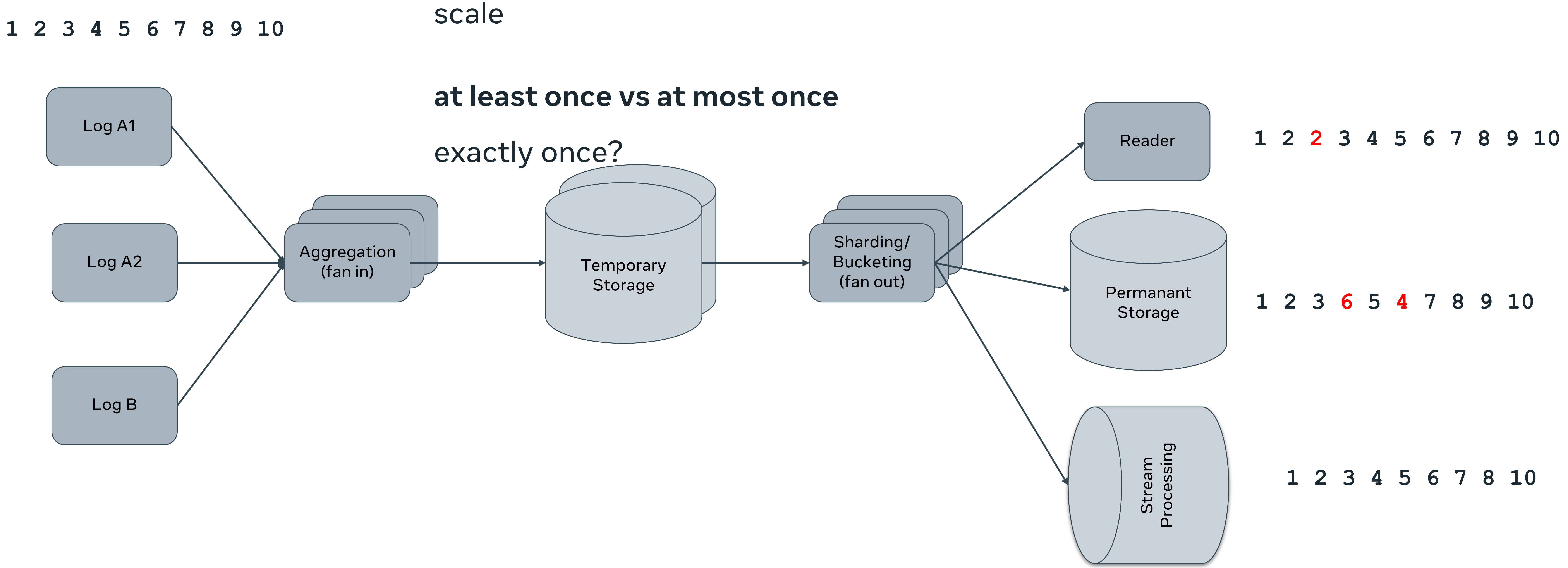
speed vs completeness

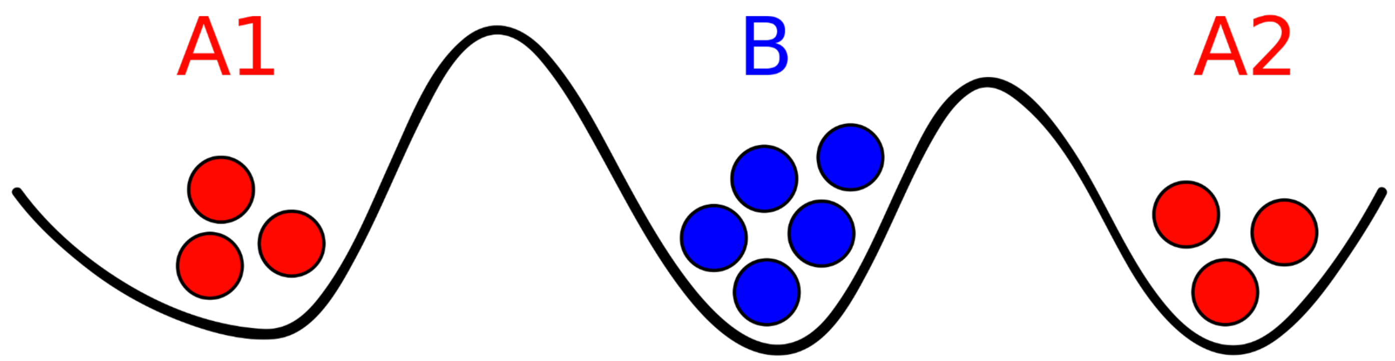
time

scale

at least once vs at most once

exactly once?

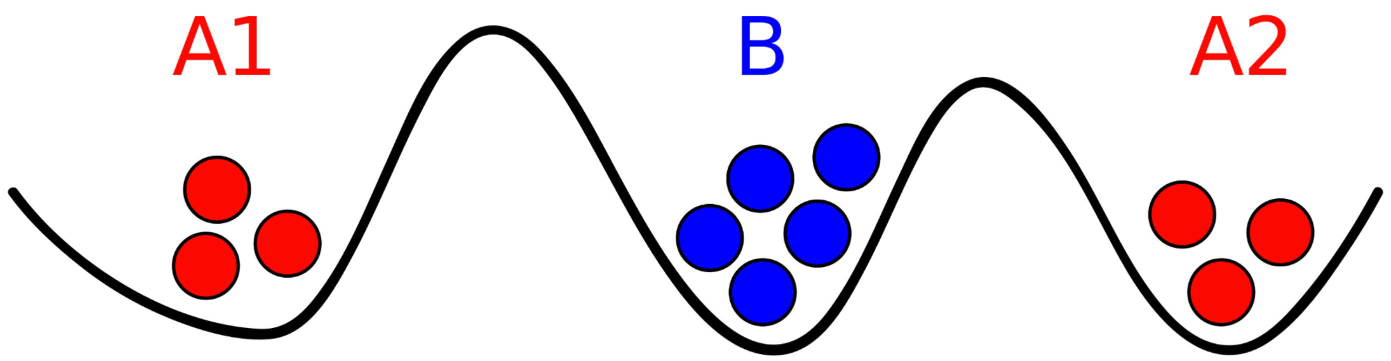




Two Generals' Problem

consensus, distributed locking, leader election

https://en.wikipedia.org/wiki/Two_Generals%27_Problem

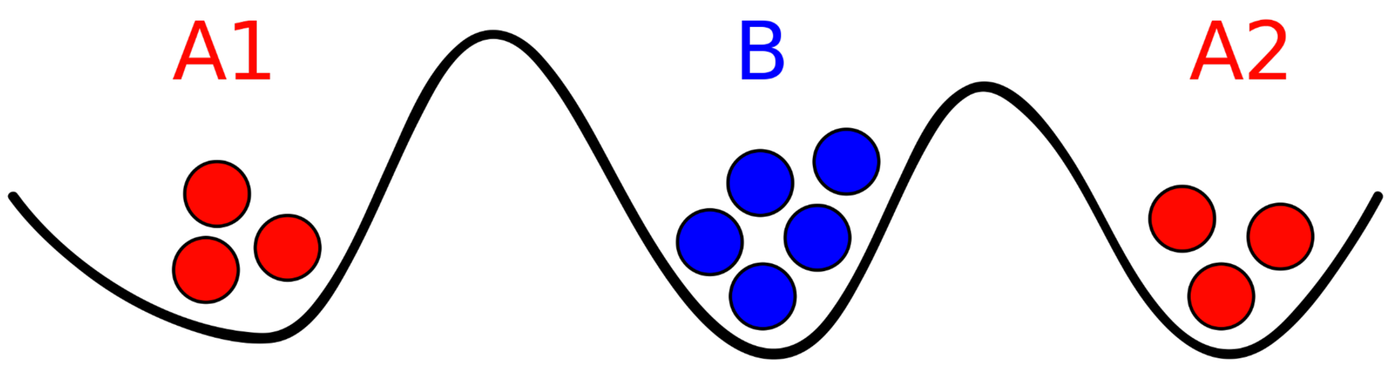


Two Generals' Problem

consensus, distributed locking, leader election

Paxos – e.g. Spanner

https://en.wikipedia.org/wiki/Paxos_(computer_science)

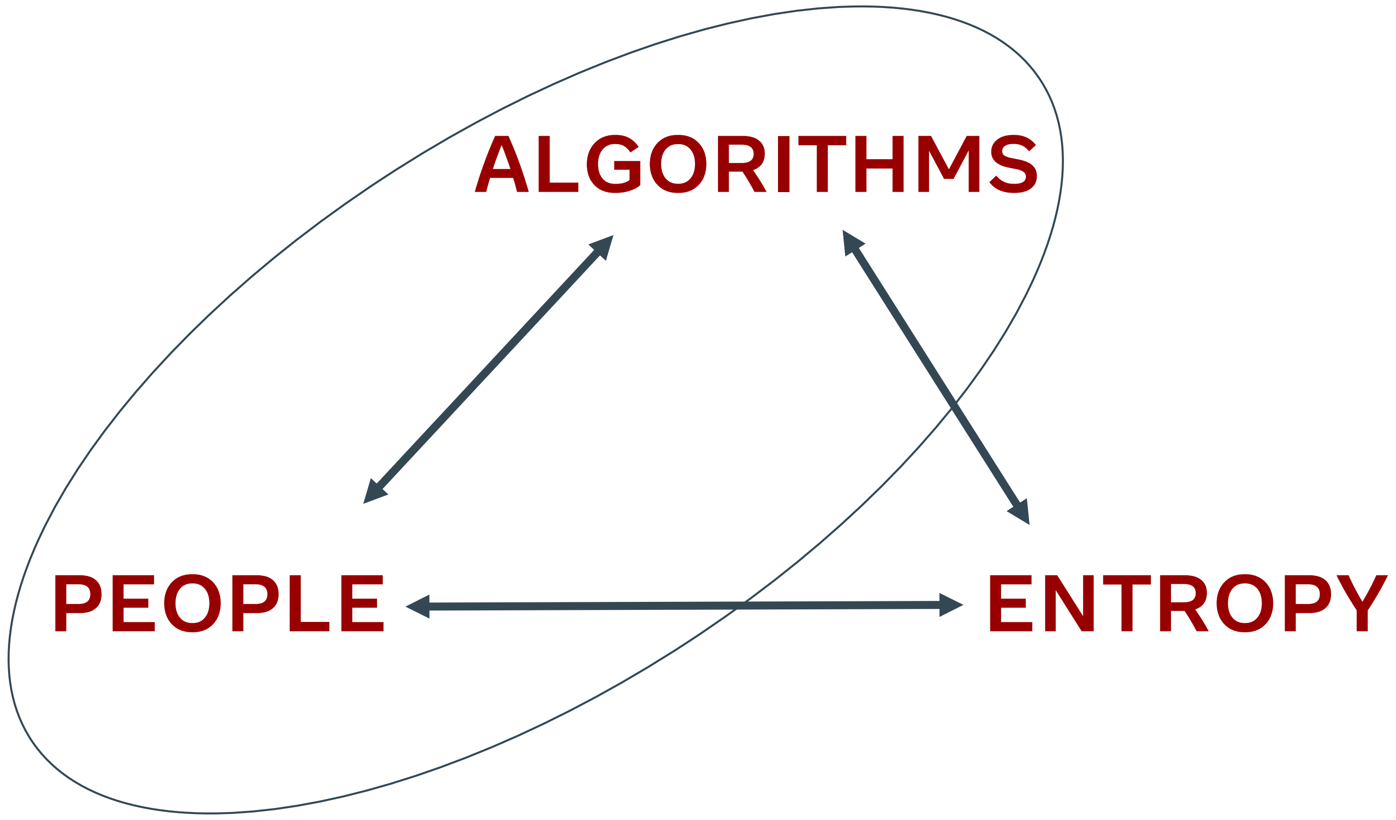


Two Generals’ Problem

consensus, distributed locking, leader election

Paxos – e.g. Spanner

Raft – e.g. Etcd

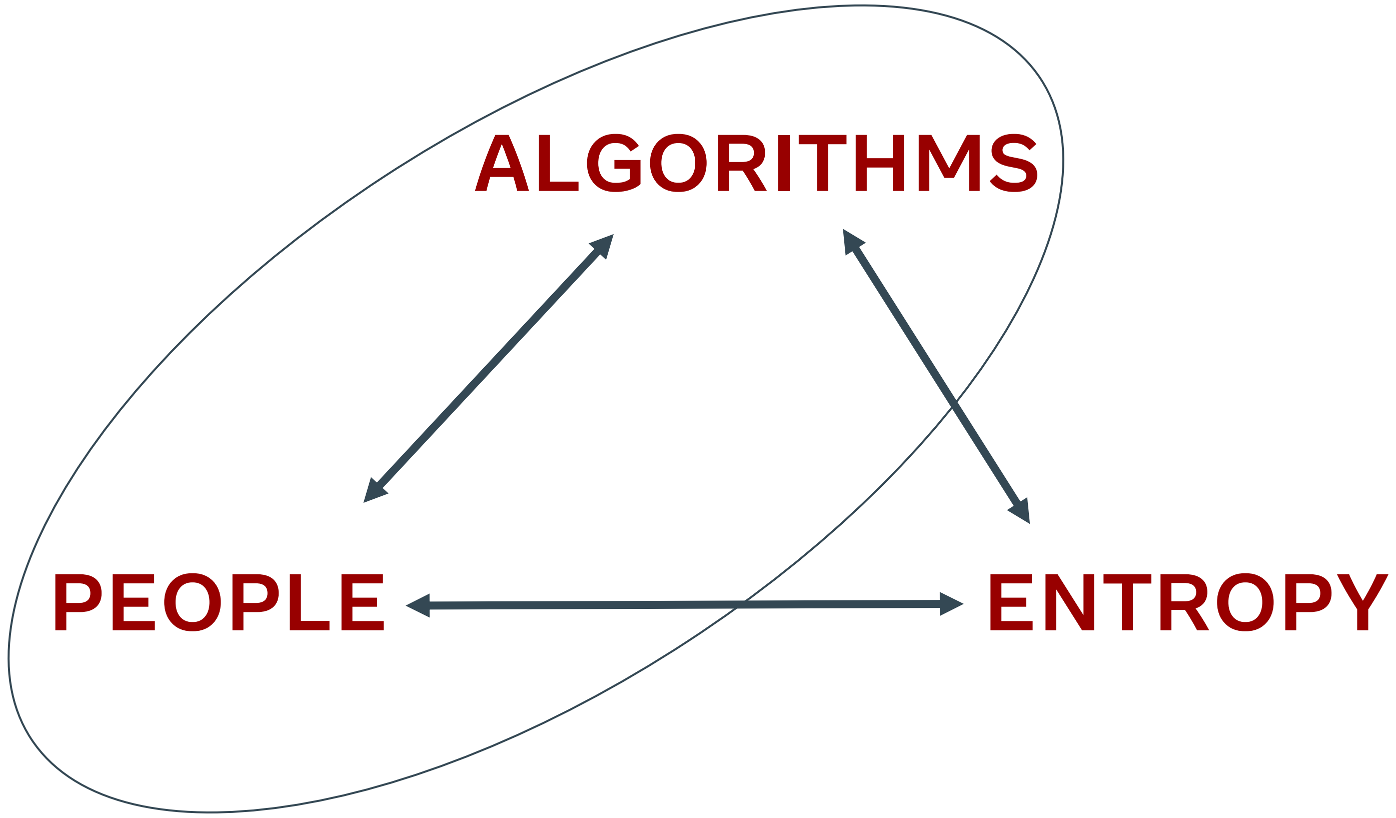


https://en.wikipedia.org/wiki/Raft_(algorithm)

04 now what?

entropy is not optional

if your system isn't designed to mitigate entropy it will tend toward chaos



you're solving a **practical** problem not a theoretical problem

you don't have to solve the problem you're given

you have to solve the actual problem not what you wish the problem was

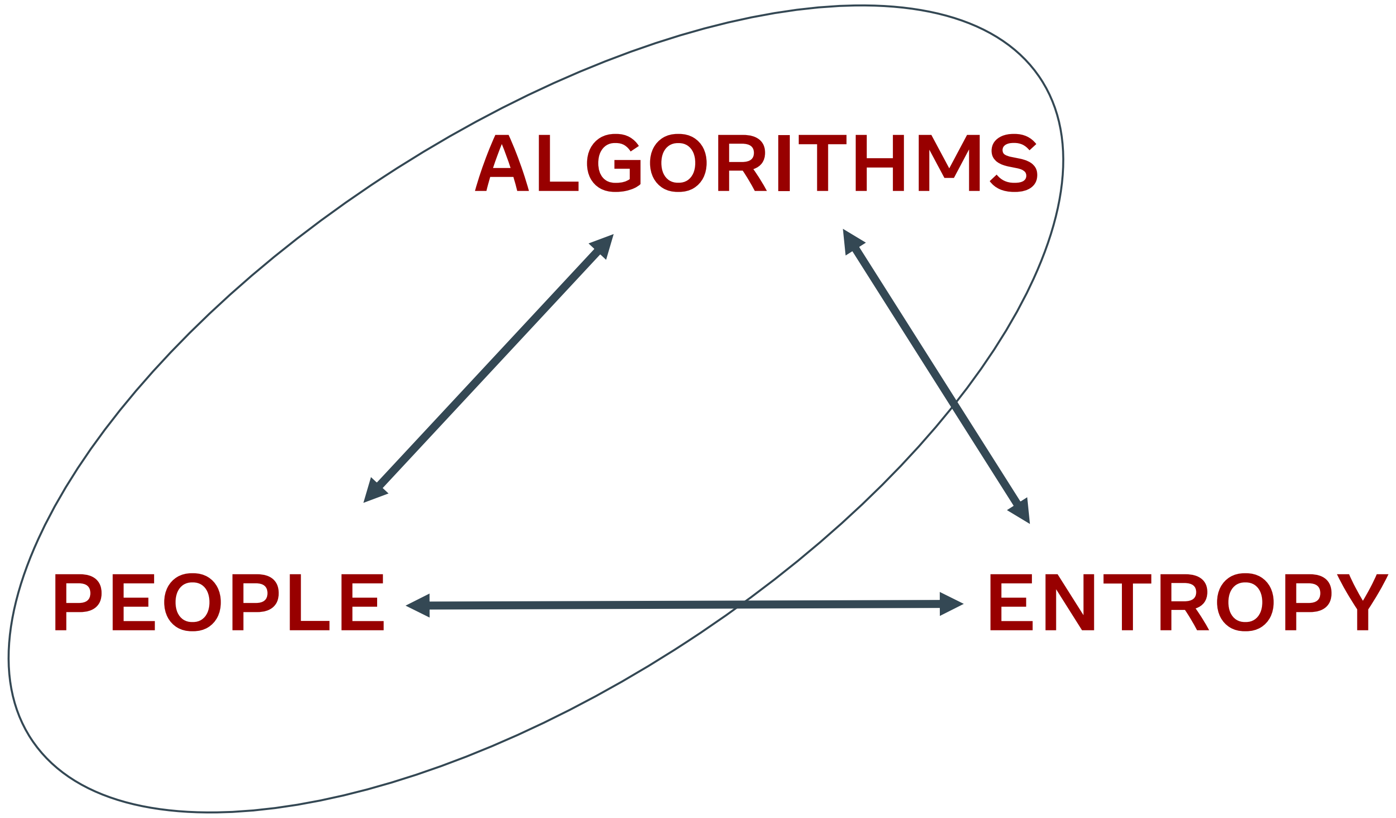
How good is good enough?

How fast is fast enough?

gather more requirements/constraints

can you turn an intractable problem into a simpler more constrained problem?

CHEAT!



it’s ok to take a breadth first approach

find interesting problems worth going into depth on

seek out people you can learn from

don’t try to remember details

remember patterns and do the research when you need it

COPY! STEAL!

have you ever felt like the protagonist in a Greek tragedy?

Hamartia - a fatal flaw leading to the downfall of protagonist

the classic tragic flaw is hubris - replace hubris with humility and discipline

avoid the temptation to be too clever for your own good or too in love with your tools

revisit your assumptions often

iterate

Thanks!

Questions?

Meta