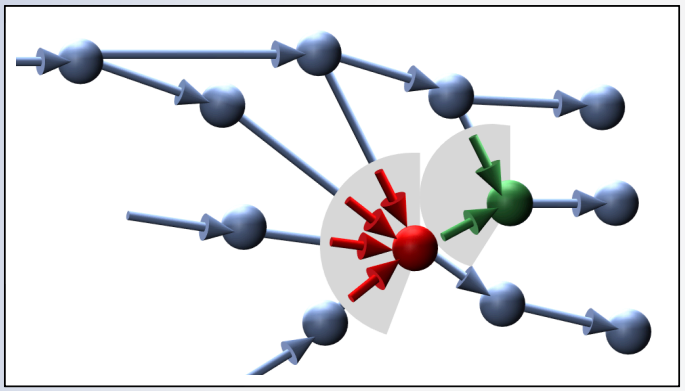


Coordination: The Partial Collapse of Partial Order

Coordination Is the Biggest Remaining Hard Problem
What IS Coordination?

Pat Helland (Oct 2022)

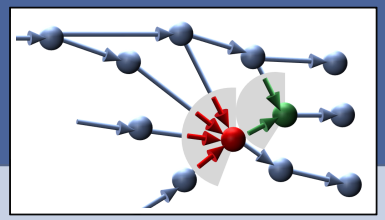
These are my own ideas and opinions

Not necessarily reflective of the position of my employers, past or present

Pat Helland is employed by Salesforce

This is a shortened version of:
"I'm SO Glad I'm Uncoordinated"

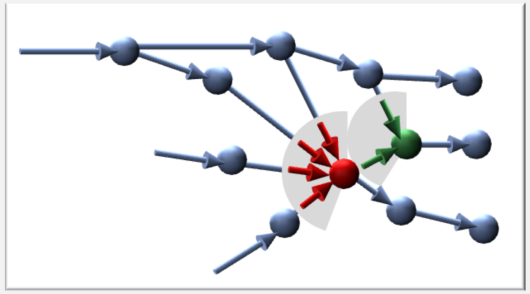
Agenda



How Things Have Changed!

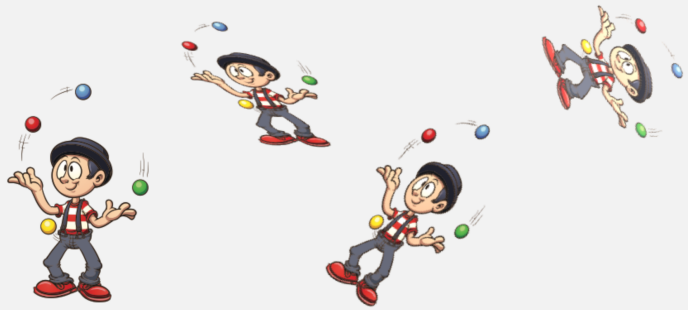
3

The Partial Collapse of Partial Order



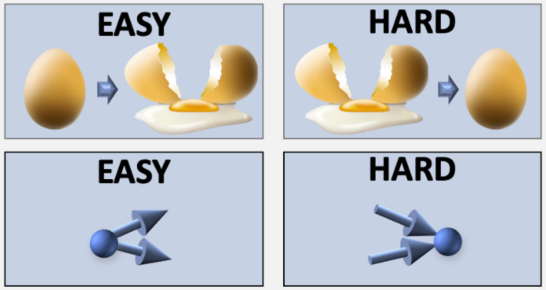
13

Coordination in a Stretched-out World



21

It's Expensive to Decrease Disorder



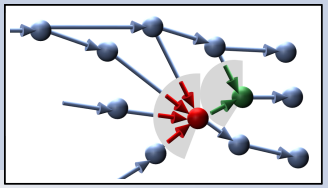
35

How Things Have Changed!

1970s

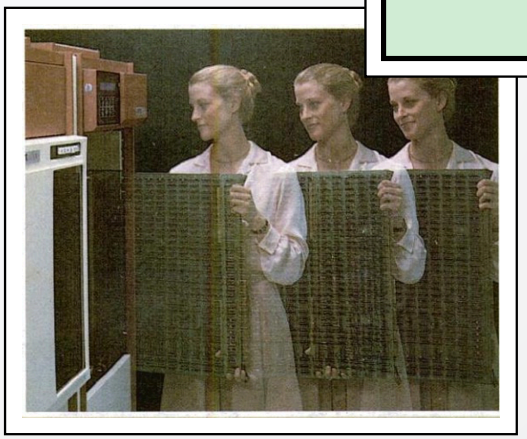
2020s

Latching in My First Programming Job

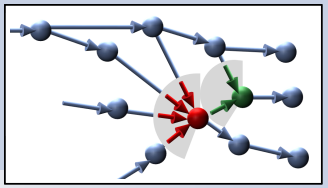


- **1978: My first programming job!**
  - BTI 8000 – SMP (Symmetric Multiprocessor)
  - Shared backplane with pluggable CPU cards
  - No network
- **Scalable Computing!**
  - Plug in more CPUs...
  - 7 CPUs → 5.5 X throughput!
- **15-Megahertz Clock**
  - 12 cycles to access memory (750ns)
  - Coherent memory (no CPU caches)
- **LATCHes, READs, WRITEs → 750ns**

Old *(1978)*	
Cheap	**Expensive**
Coordination *(same cost as Read & Write)*	**CPU**
	Memory
	Storage
	Bandwidth

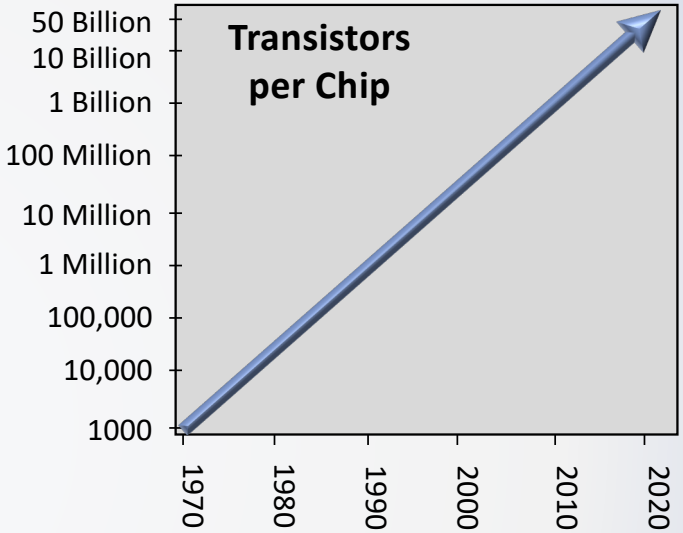


We're Getting "Moore" & "Moore" Stuff!



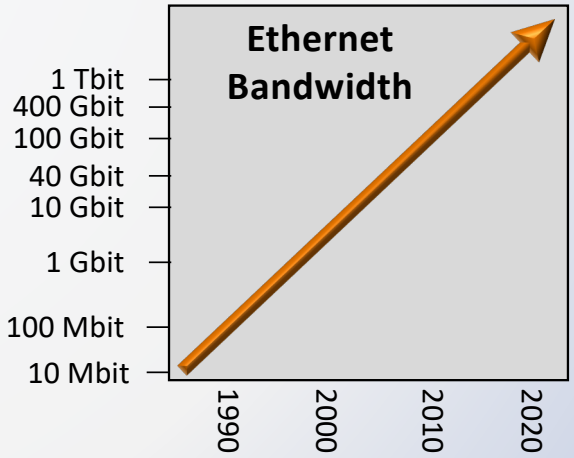
- **Moore's Law:** Transistors double every 18-24 months
  - CPU, SSDs, NVMe, and DRAM

50 billion transistors per chip!

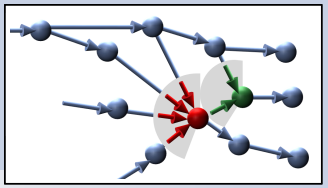


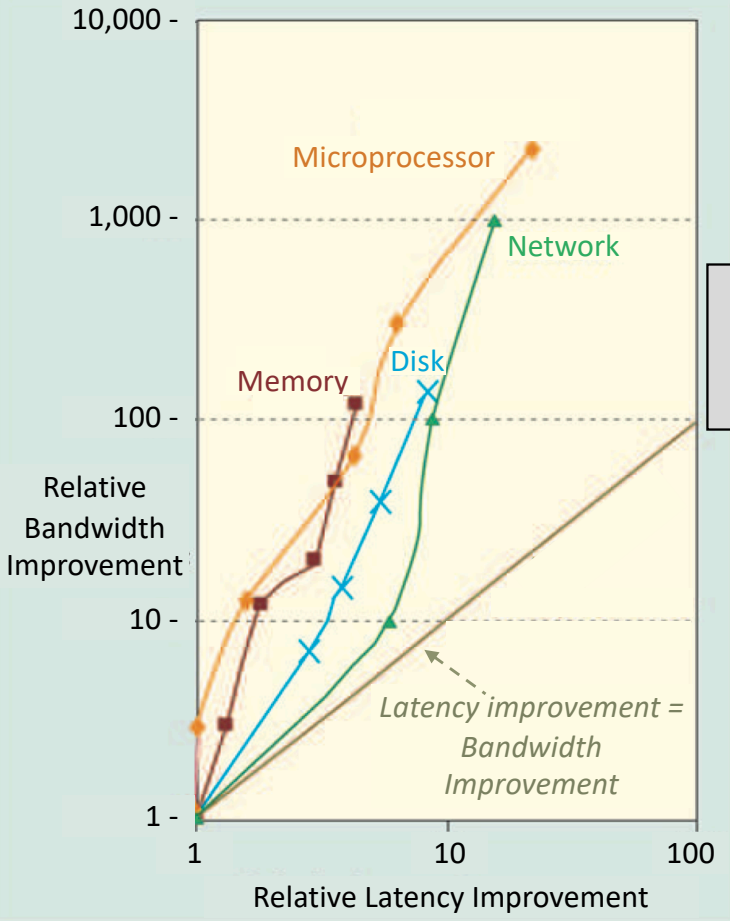
- **Gilder's Law:** Total bandwidth triples per year
  - More connections
  - Faster connections

400Gbit ethernet now
1Tbit ethernet soon



Latency Lags Bandwidth *(Patterson-CACM 2004)*





Note that latency improved about 10X while bandwidth improved about 100X to 1000X

This slide derived from:

"Latency Lags Bandwidth"

By Dave Patterson –

(Communications of ACM - October 2004)

Statistics from about 1982 to 2004

Trend is still true 18 years later!!

"Bandwidth improves by at least the square of the improvements in latency."

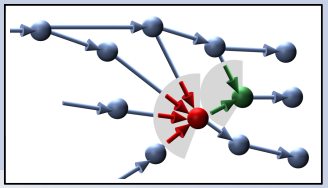
Why Does Latency Lags Bandwidth?

There is an old network saying:

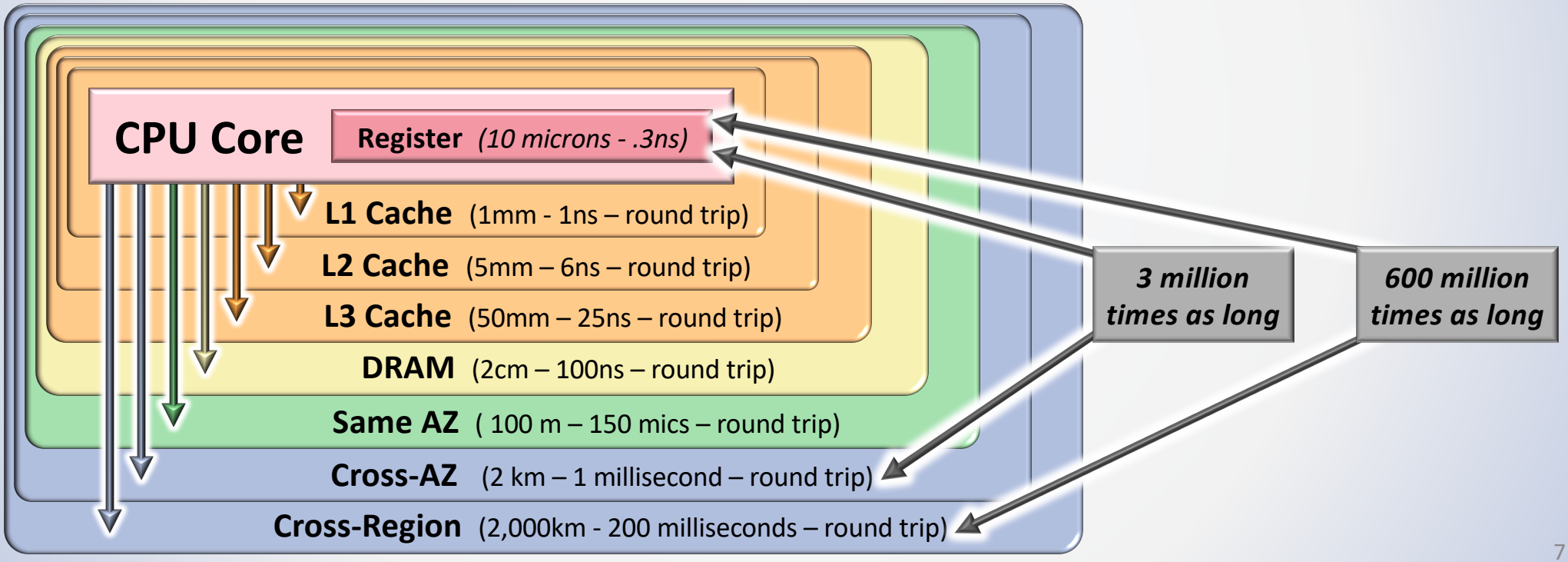
Bandwidth problems can be cured with money.

Latency problems are harder because the speed of light is fixed – You can't bribe God. *-- Anonymous*

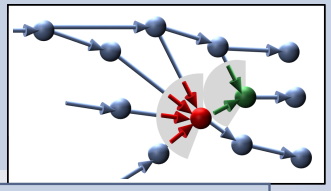
Waiting for Stuff That's Far Away



- **CPU cores need data**
  - If they don't have it, they wait
- **The farther away the data, the longer the CPU core waits**
  - Waiting is inefficient



Putting Latency in Perspective *(2020s)*



Computer Perspective *(3 GHz Clock)*				Ratio	Human Perspective	
Distance	**Latency** *(wall clock)*	**Latency** *(cycles)*	**Instruction Opportunities**		**Distance**	**Time**
CPU Register	**.3 ns**	**1 cycle**	**5 or so** (if all is GREAT)	**1**	**My Pocket**	**1 Second**
L1 Cache *(per core)*	**1 – 1.2 ns** *(per core)*	**3-4 cycles**	**15– 20** (if all is GREAT)	**4**	**My Desk at Work**	**3-4 Secs**
L2 Cache *(multi-core same socket)*	**6 ns**	**20 cycles**	**~ 100 Instructions**	**20**	**File Cabinet** *(my office)*	**20 Secs**
L3 Cache *(cross-socket)*	**24-30 ns**	**72-90 cycles**	**~ 400– 500 Instructions**	**80**	**Another Floor at Work**	**1.5 Mins**
DRAM	**80-100 ns**	**260-330 cycles**	**~ 1000 – 1500 Instructions**	**300**	**Walk 2 City Blocks**	**5 Mins**
Cross-Server RPC/TCP *(local AZ)*	**150 microseconds**	**450,000 cycles**	**~ 2 Million Instructions**	**450,000**	**Walk 100-150 Miles** *(160 – 240 km)*	**5 Days**
Cross-AZ RPC/TCP	**1 millisecond** (1,000,000 ns)	**3,000,000 cycles**	**~ 15 Million Instructions**	**3,000,000**	**Walk 600-900 Miles** (1000-1500 km)	**One Month**

This slide inspired by:

"AlphaSort: a cache-sensitive parallel external sort"

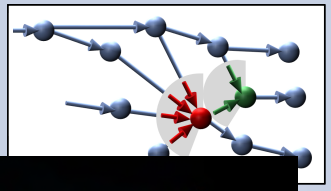
By Chris Nyberg, Tom Barclay, Zarka Cvetanovic, Jim Gray, & Dave Lomet
VLDB – October 1995

	1995	2022
CPU Clock	150 MHz	3 GHz
Wall Clock	80-100 ns	80-100 ns
Cycles	12-15	260-330

DRAM Access Cost

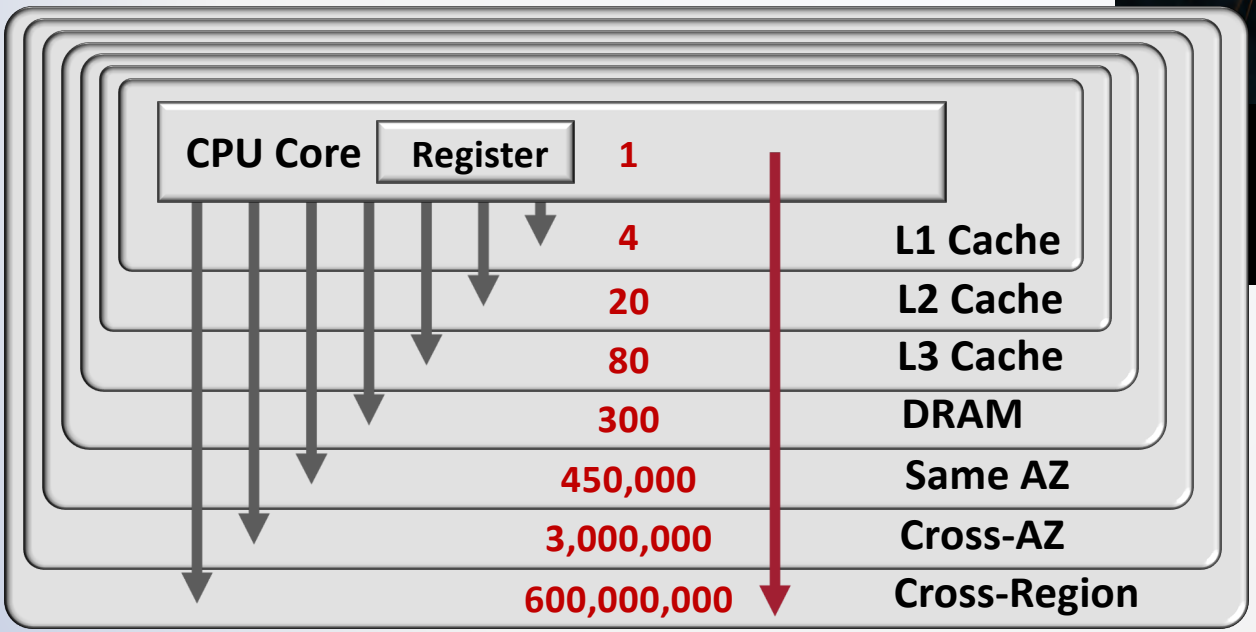
DRAM Is 21 Times Farther *(2020s vs 1995)*

Hubble's Universe: Everything Is Getting Farther Away

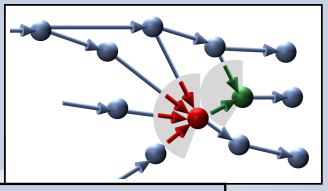


- **More stuff and more thinking…**
  - We can store more stuff
  - We can move more data per millisecond
  - We can think more per millisecond
- **More thinking while waiting for stuff**

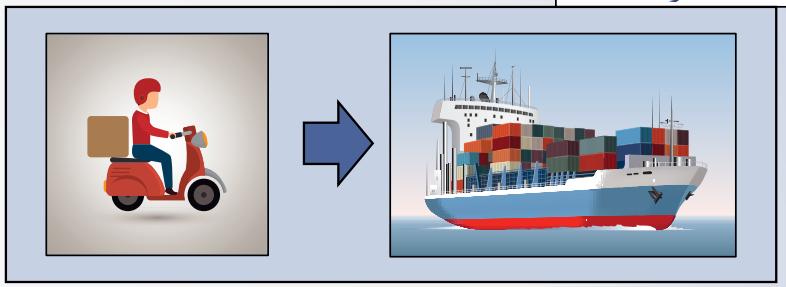
Hubble's Expanding Universe



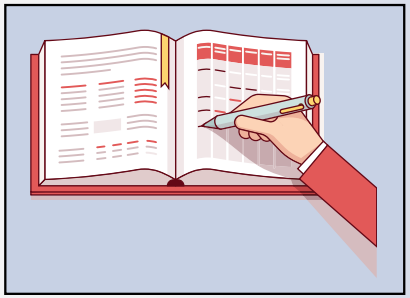
Waiting: Reading, Writing, & Coordination



- **Waiting to READ data from far away?**
  - Immutable data… easy to cache
  - LOTS of data each trip

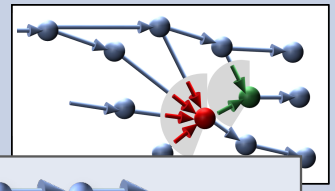


- **Waiting to WRITE data kept far away?**
  - Appending to a log → Batch appends
  - Reorganize for easy reading

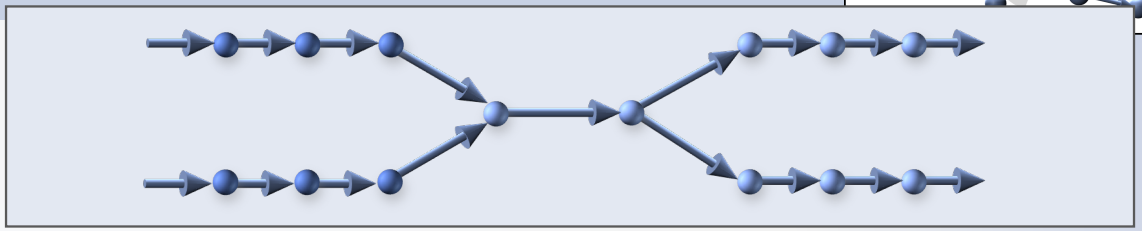


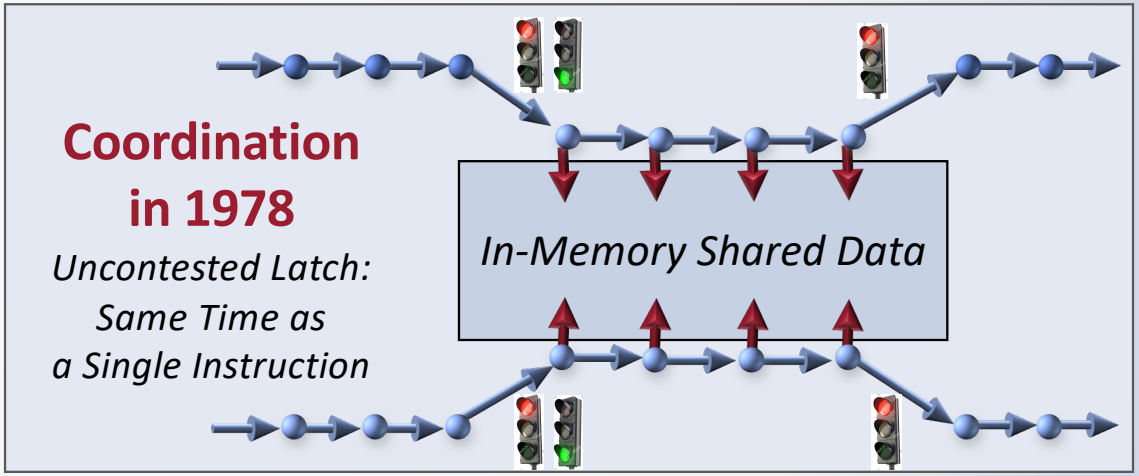
- **Waiting to COORDINATE**
  - Parallel activity is important for many things!
  - When can distant activities remain parallel?
  - ***When do parallel activities need coordination?***

Coordination: Aligning Parallel Work



- **Parallel activity is powerful**
  - Concurrent threads, servers, work
- **Coordination: Align parallel work**
  - Share data, control completion, advance steps
- **Coordination in 1978**
  - Multiple threads & CPUs
  - Latch shared data structures
- **Coordination in 2022**
  - Multi-core latches
  - Zookeeper via RPCs
  - DB via RPCs





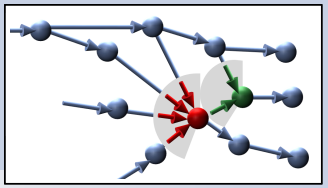
2022: Varying coordination delays

- **Same server latch** 1 cycle to 300 cycles
- **Remote server RPC:** 450K to 3 million cycles!

Uncontested Coordination!

Much worse if you fight for access…

How Things Have Changed!!



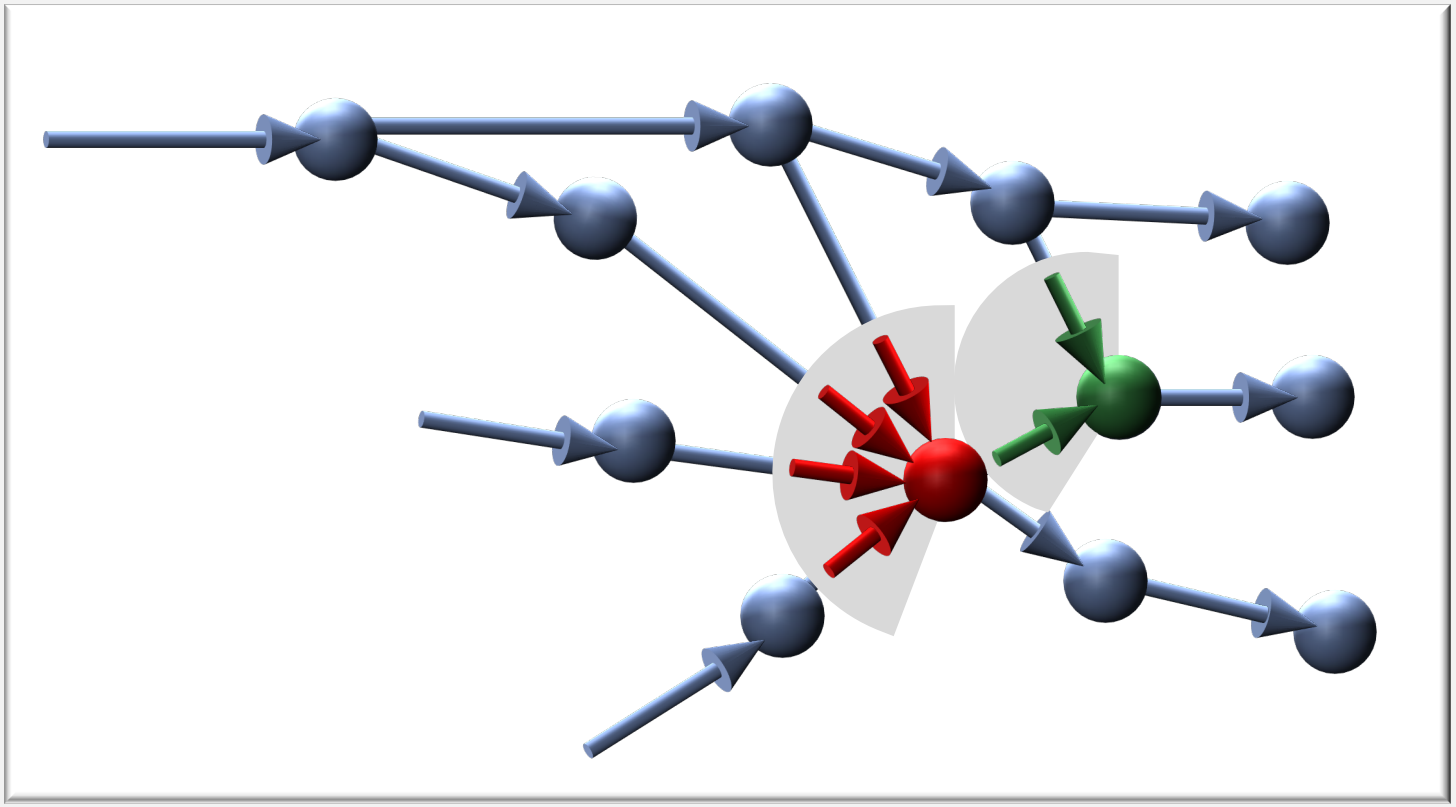
Old *(1978)*	
Cheap	**Expensive**
Coordination *(same cost as Read & Write)*	**CPU**
	Memory
	Storage
	Bandwidth

New *(2022 →)*	
Cheap	**Expensive**
CPU	**Coordination**
Memory	
Storage	
Bandwidth	

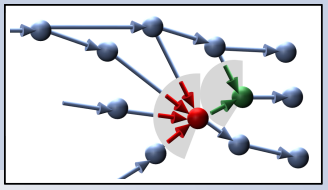
Is There an Abstraction for Coordination?

What Can We Do to Reduce the Pain??

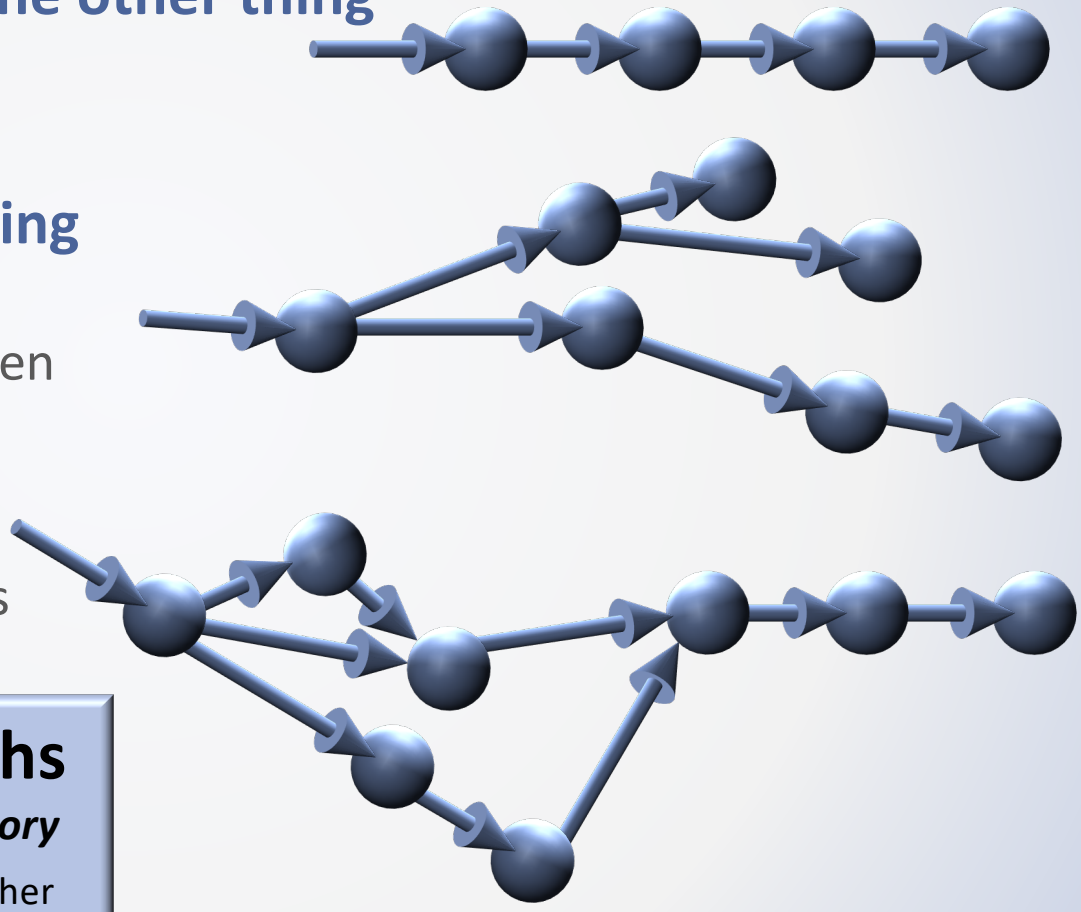
The Partial Collapse of Partial Order



History in a System: Lists, Trees, & DAGs



- **List: One thing follows exactly one other thing**
  - Each thing has one parent
  - Each parent has one child
- **Tree: Many things follow one thing**
  - Each has exactly one parent
  - Each parent may have many children
- **DAG (Directed Acyclic Graph): Many parents & many children**
  - Each thing may have many parents and many children

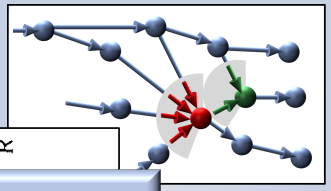


No CYCLIC Graphs

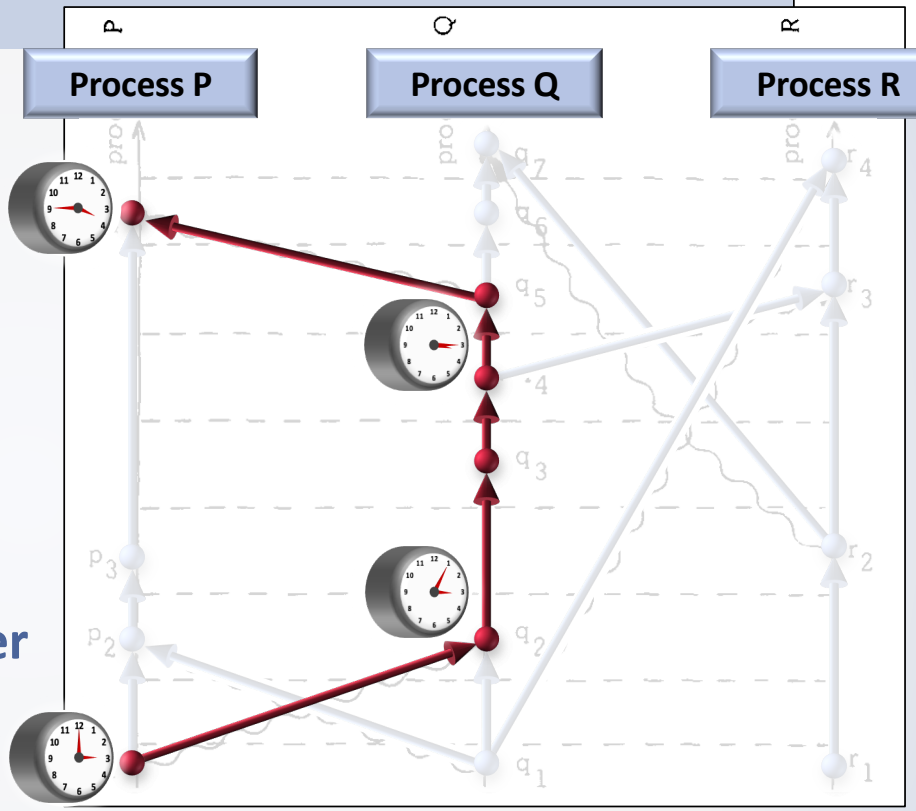
Can't Have Cycles in History

Can't be your own grandfather

Lamport: Ordering of Events in a Distributed System



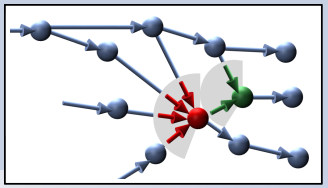
- **Leslie Lamport defined event ordering in a distributed system**
  - Multiple processes
  - Send and receive events
  - Events (messages) define the system order
- **Happened before: the ordering of events**
  - Stuff at sending server "happened before"
  - Includes ***happened before*** of sender
- **Partial Order: Messages form partial order**
  - A DAG of processes and events
  - Your time AFTER all that "happened before"



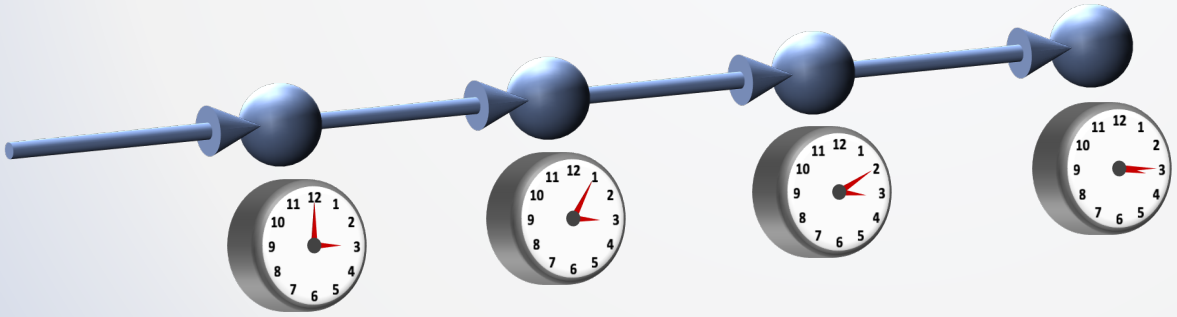
Example: an RPC from Process P to Process Q

"Time, Clocks, and the Ordering of Events in a Distributed System"
By Leslie Lamport – (figure 3)
(Communications of ACM – July 1978)

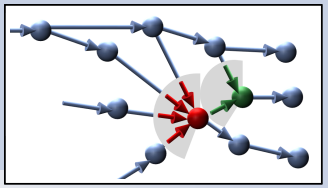
Newton & Von Neumann: Time as Total Order

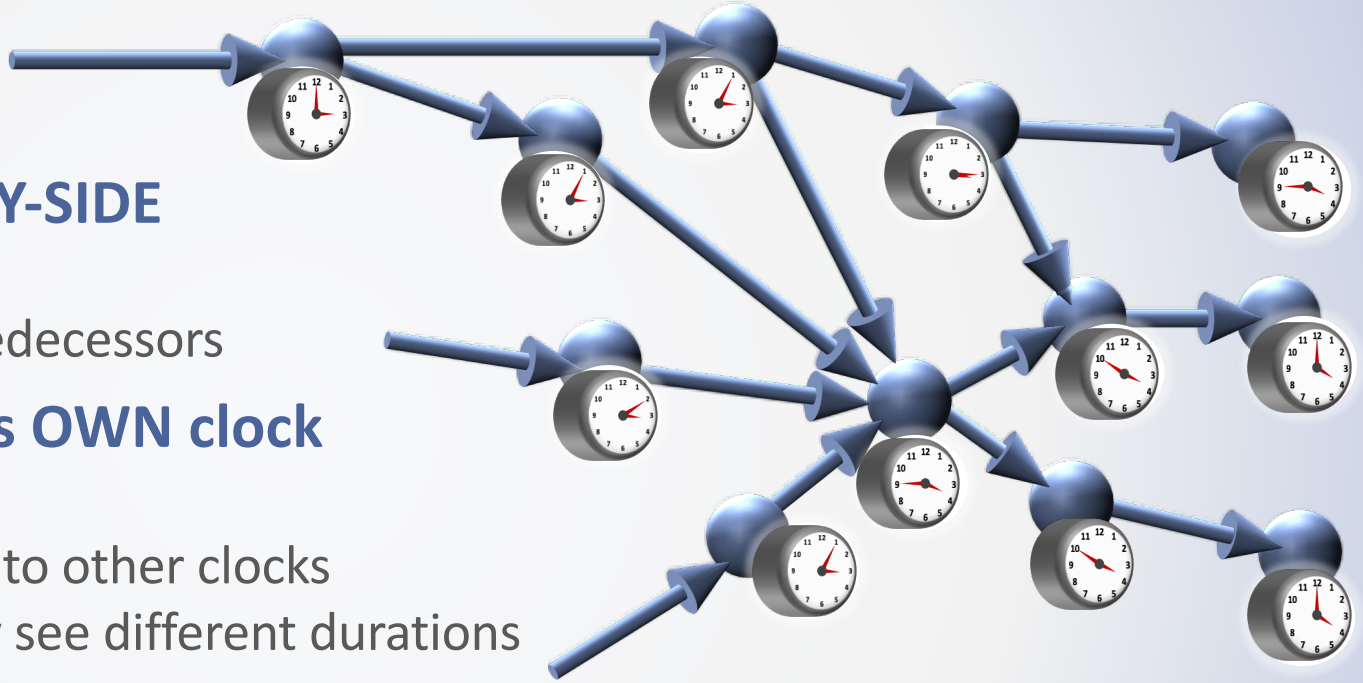


- **Total order:** **One thing happens after ONE other**
  - Follows most recent predecessor
  - Everything either before or after
- **Newton's universe:** **Time marches forward uniformly**
  - Newton: All clocks are the same
  - Time advances at a uniformly everywhere
- **Von Neumann computing:** **One instruction follows another**
  - In a Von Neumann central computer, all clocks are the same clock



Einstein & Lamport: Time as Partial Order

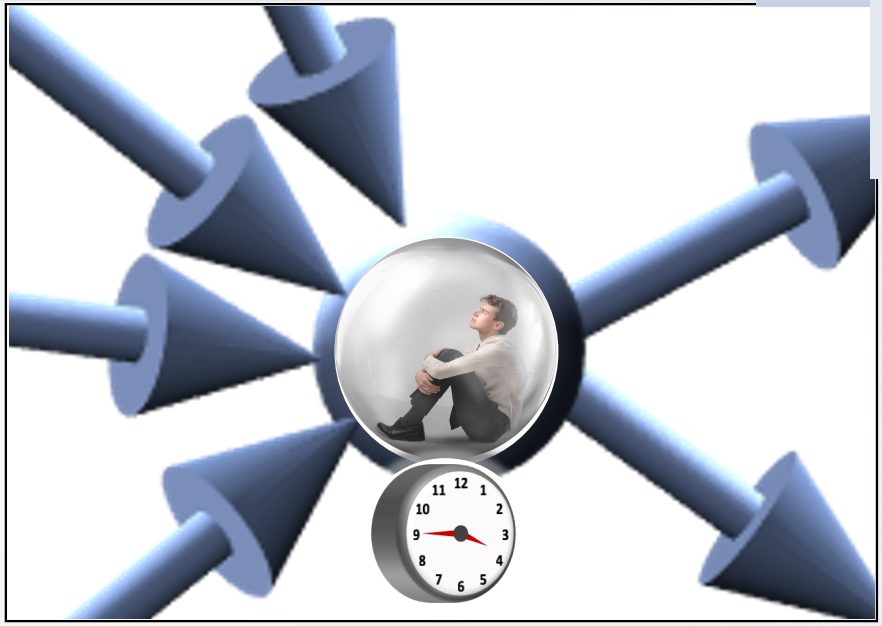




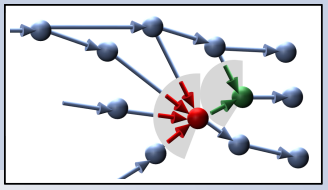
- **Partial order:**
  BEFORE, AFTER, or SIDE-BY-SIDE
  - Partial order is a DAG
  - Things may follow many predecessors
- **Einstein: Each thing has its OWN clock**
  - Follows ALL predecessors
  - Clocks may be slow relative to other clocks
  - Paths thru *the universe* may see different durations
- **Lamport: Your predecessors "happened before"**
  - Follows ALL predecessors
  - Clocks may be slow relative to other clocks
  - Paths thru *the network* may see different durations

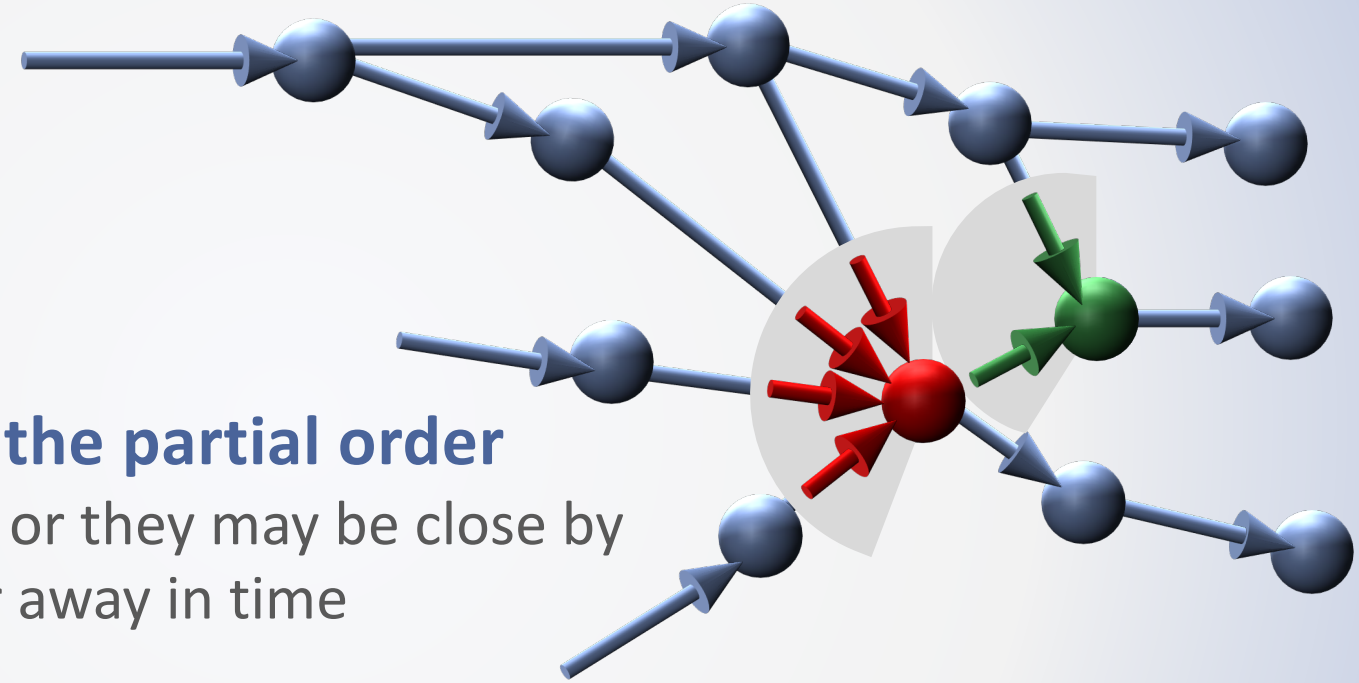
Subjectively, I Really AM the Center of the Universe!

- **Partial order is a DAG**
  - DAG: Directed Acyclic Graph
- **Each piece of the DAG is lonely**
  - Messages come into it
  - Messages go out of it
- **Lamport clocks:**
  After what "happened before"
  - I see what happened before me
  - I don't see what happens after me



Reuniting Branches of the Partial Order

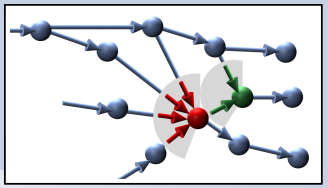




- **Coordination:**
  Reuniting branches of the partial order
  - These may be far away, or they may be close by
  - Far away in space or far away in time

- **Aligning multiple inputs can be painful**
  - **Time:** *Aligning time is hard*
  - **Space:** *Aligning distant servers is hard*

The Partial Collapse of Partial Order

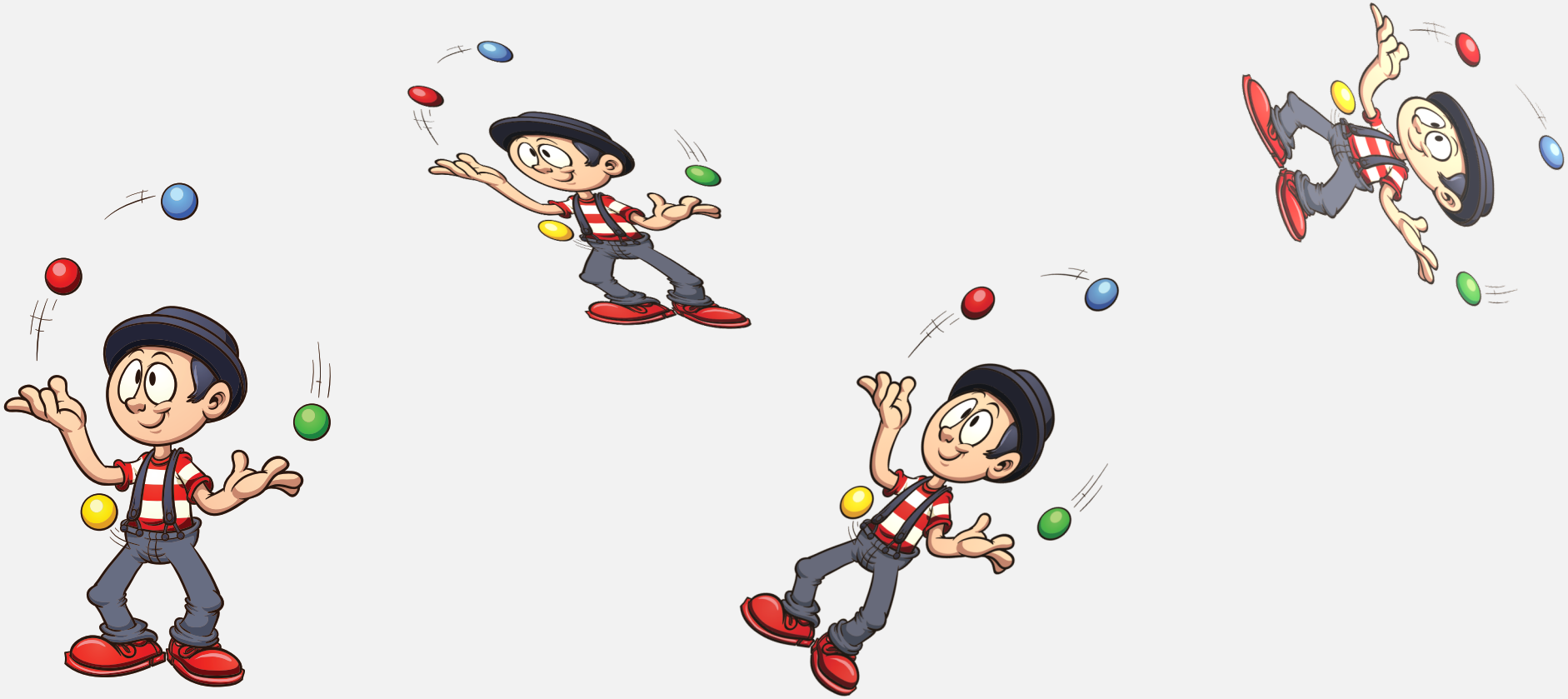


- **The Universe is a big place**
  - Our Earth is a small part of it
  - Computation on Earth is a subset of the Universe
- **Partial order → Smaller than the Universe's Total Order**
  - Stuff in Andromeda is sharded from stuff on Earth
- **Coordination can be small or large**
  - Across threads, cores, servers, departments, companies, etc.
  - Coordination is fractal!

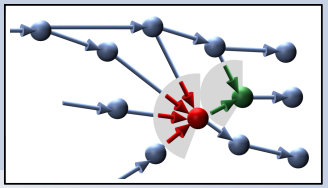
Coordination: The Partial Collapse of Partial Order

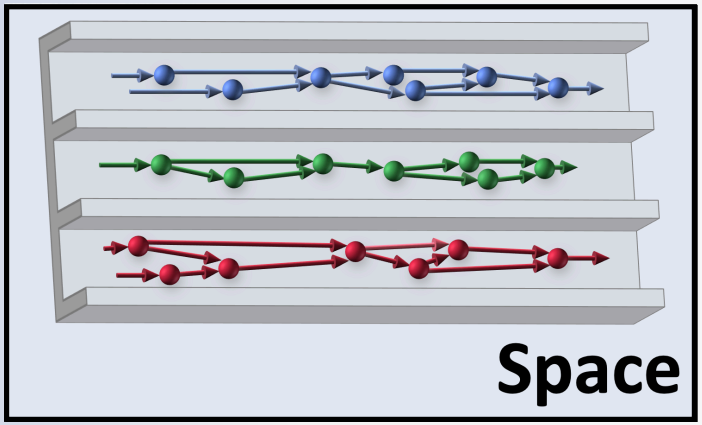
Coordination Reduces the Disorder within a System

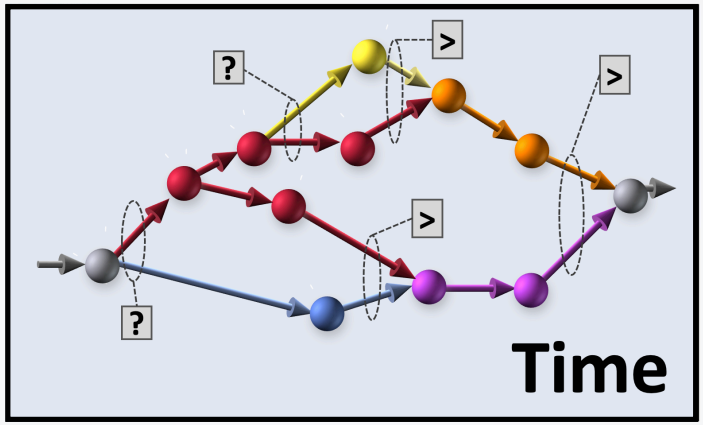
Coordination in a Stretched-out World

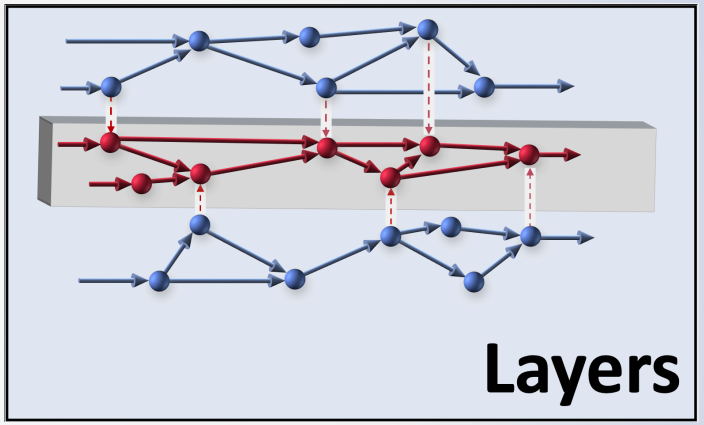


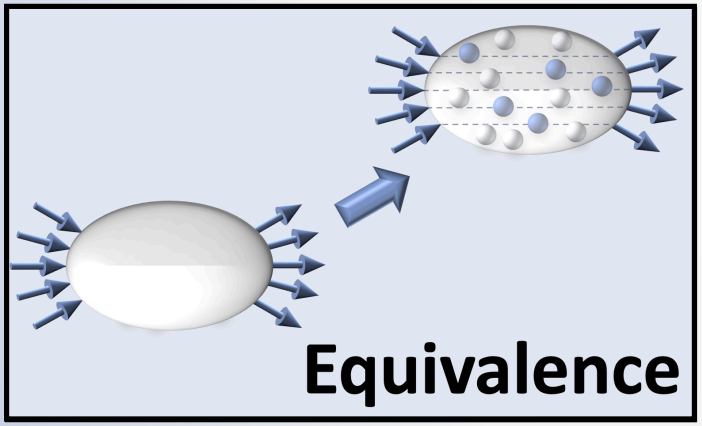
Ways to Relax Coordination

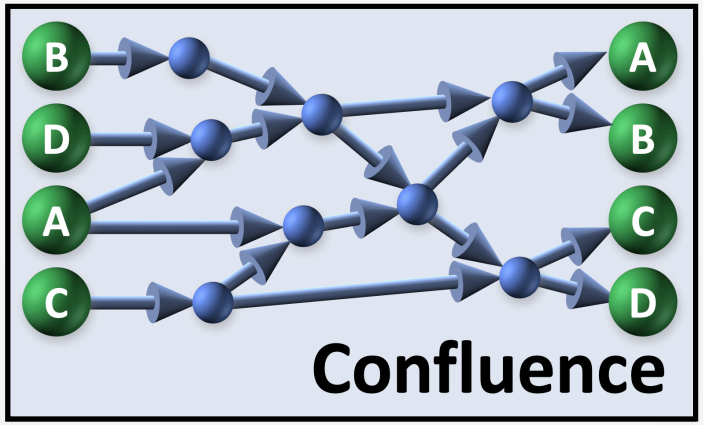




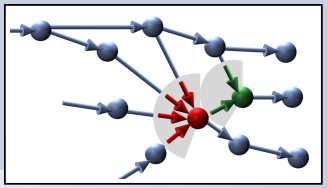


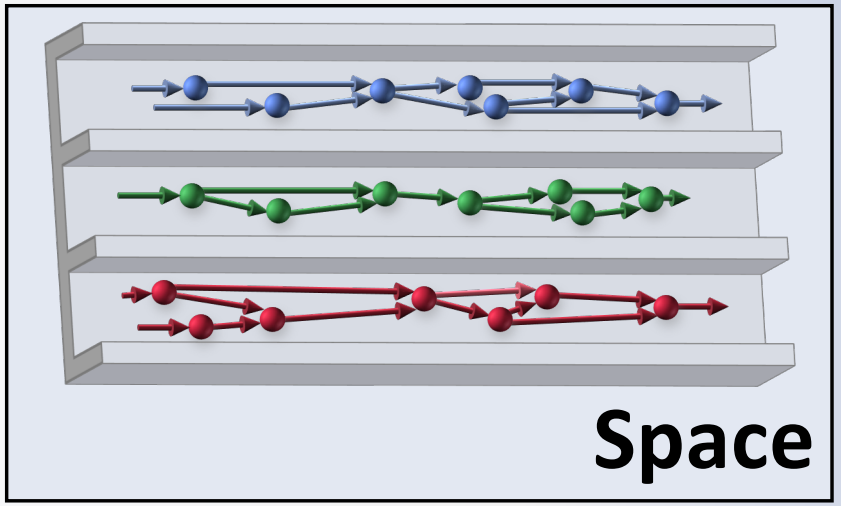






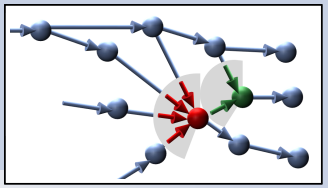
Relax Coordination with Space

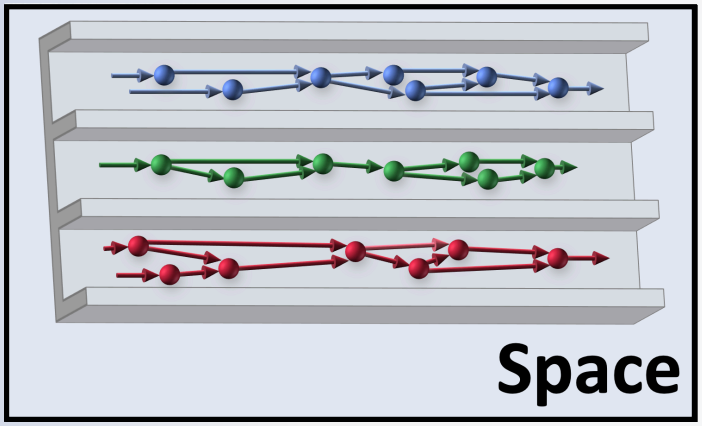


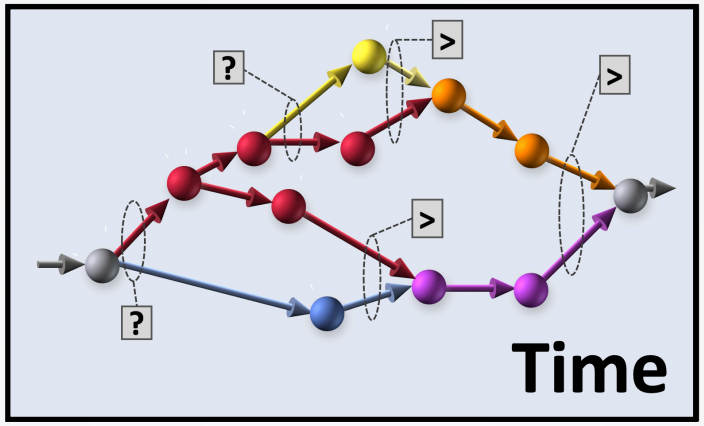


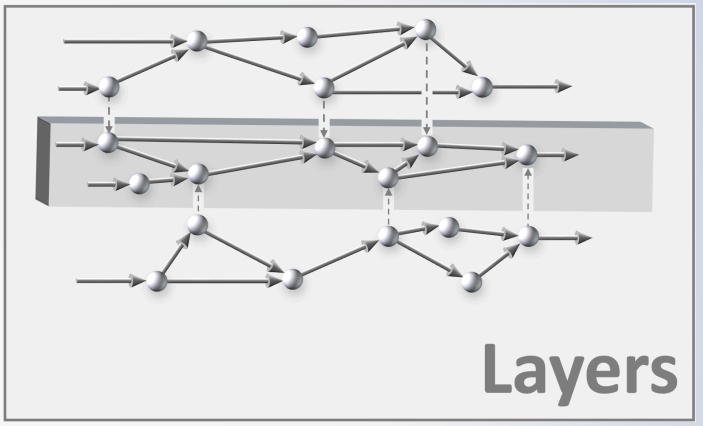
- **Different space → Separate coordination**
  - **Big:** Earth ≠ Andromeda
  - **Small:** Thread private buffer
- **What is separated by space? How do I use it?**
  - **Memory:** Reads and writes to memory
  - **Linearizable objects:** A view from outside
    - *Herlihy & Wing Linearizability*
    - *An object & its operations*
    - *Timing of remote clients' operations to object*
  - **Serializable database:** Looks linearizable from outside & parallel inside!
    - **Outside:** *Transactions APPEAR one at a time*
    - **Inside:** *Parallel execution in database*

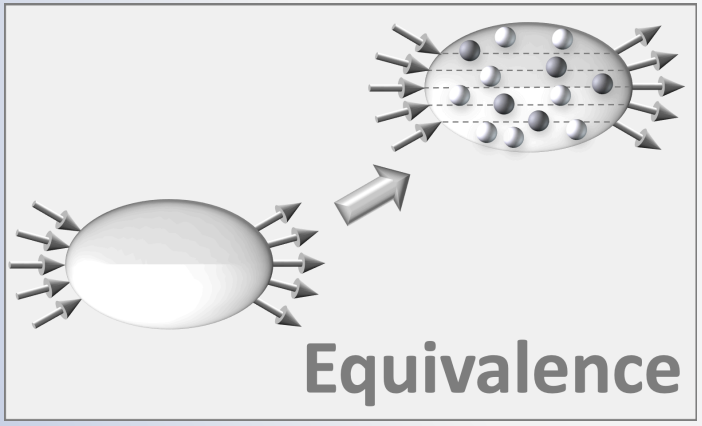
Ways to Relax Coordination

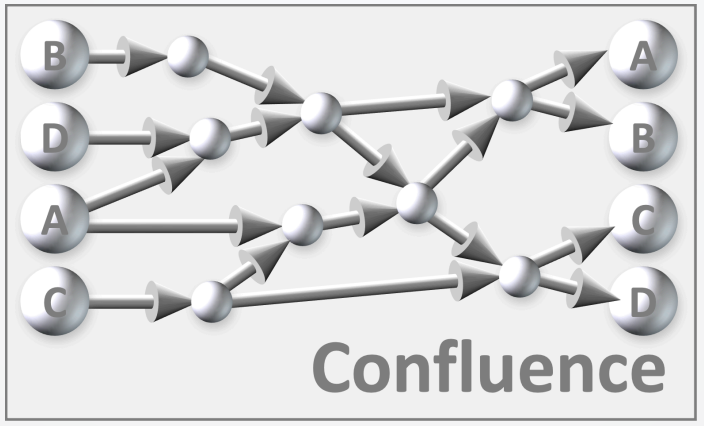




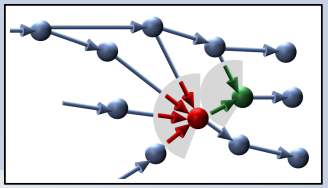




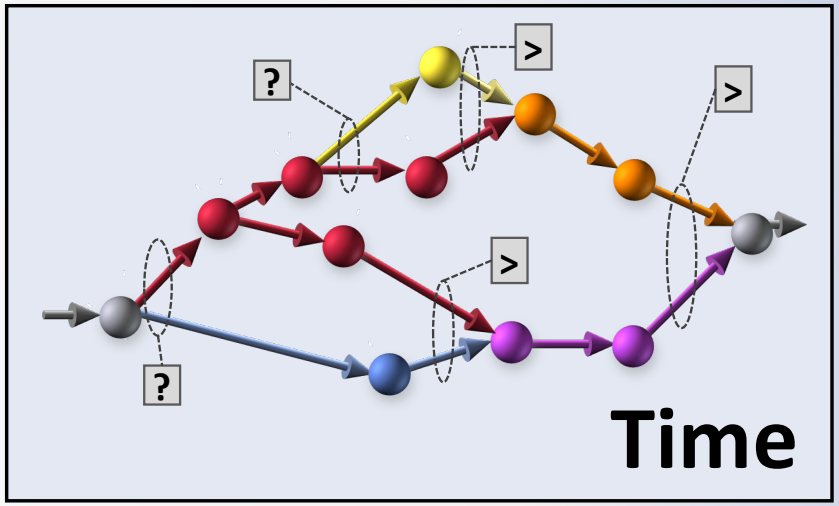




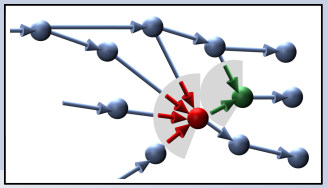
Relax Coordination with Time

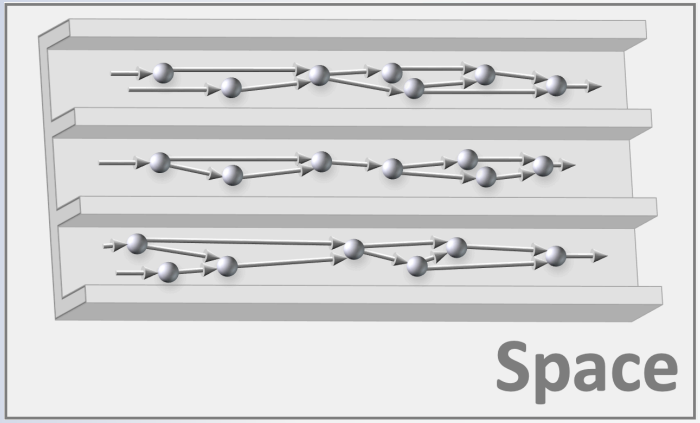


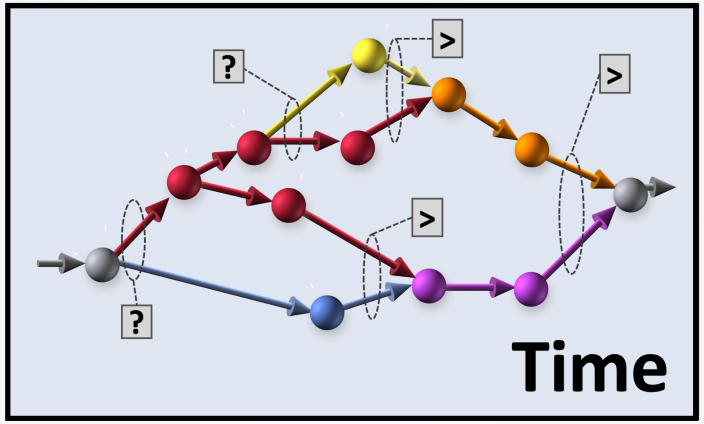
- **Fate sharing reduces coordination**
  - Dependencies on earlier stuff
  - Speculation defers coordination
- **Immutable data → Coalesces speculation**
  - Immutable inputs → Functional programming
- **Pre-allocation: Speculatively get more stuff**
  - Get expensive stuff early & avoid waiting
- **Retries after timeout == Speculation**
  - Retrying idempotent work bounds latency

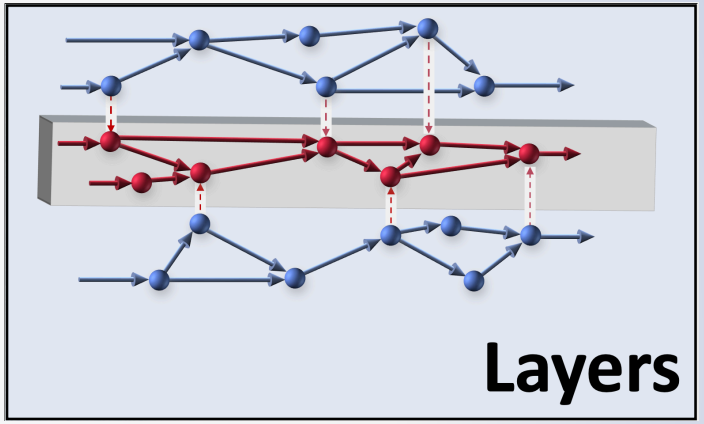


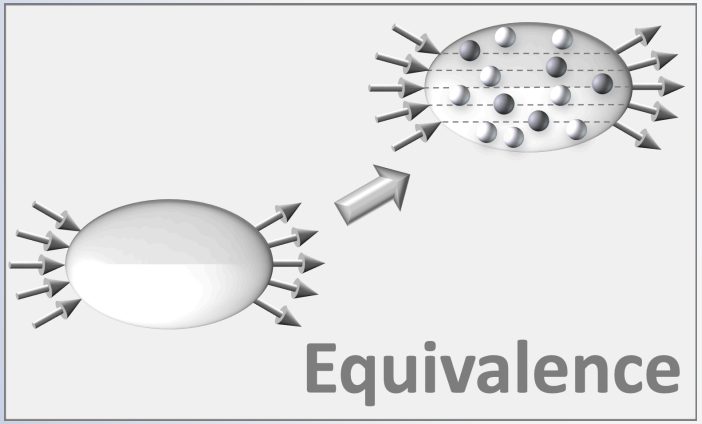
Ways to Relax Coordination

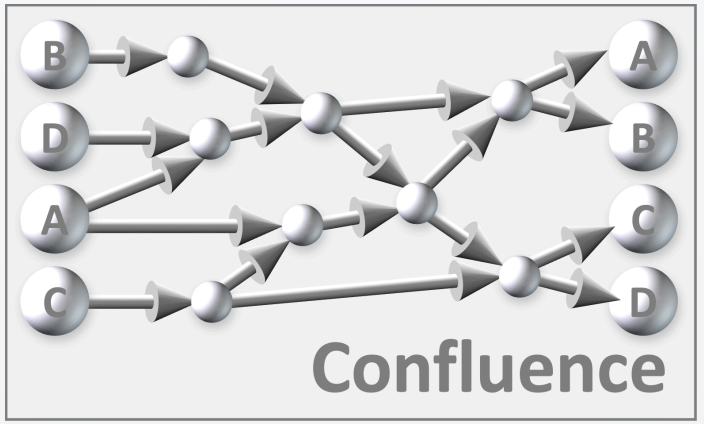




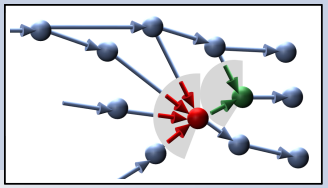




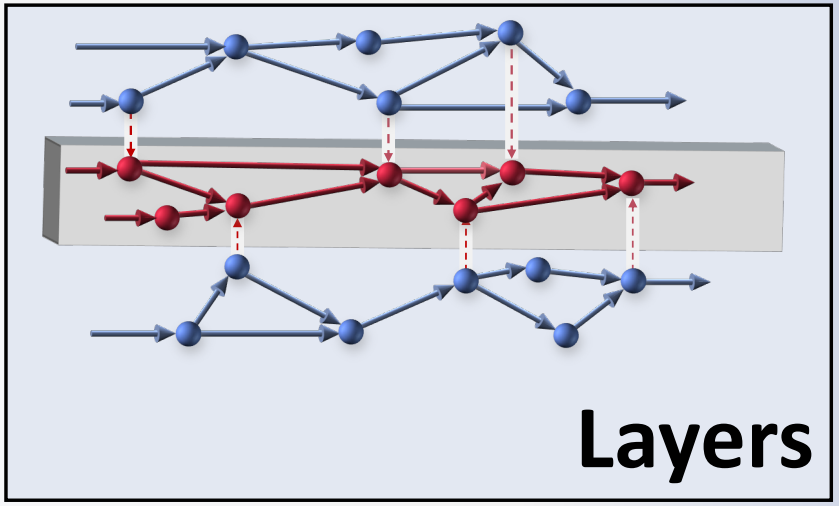




Relax Coordination with Layers



- **Layers of abstraction**
  - Avoid side-effects across layers!
  - Higher-level operations → Reduced coordination

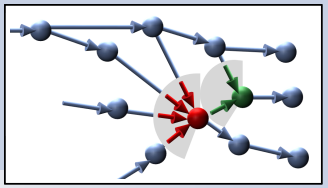


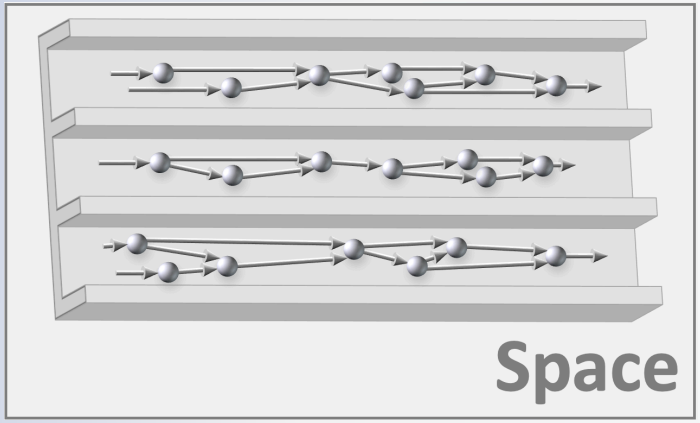
- **Avoid false dependencies**
  - Databases (page locks became records locks)
  - Microservices encapsulated their memory

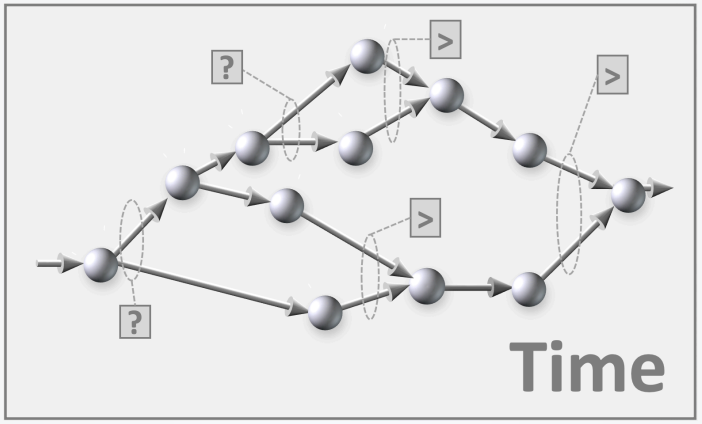
- **Avoid interference**
  - Coordinate externally visible semantics, not internal details

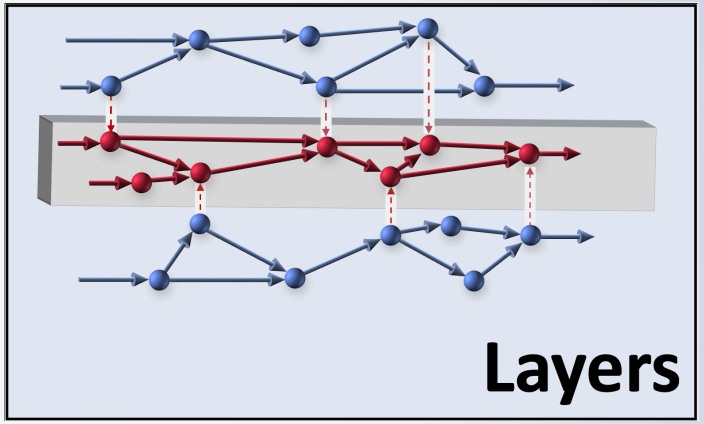
- **Commutative operations → Isolate partial order**
  - CRDTs (Conflict-Free Replicated Data Types) allow reordering

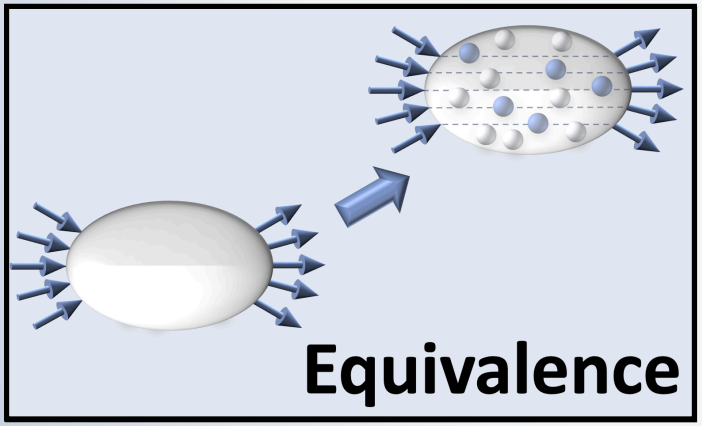
Ways to Relax Coordination

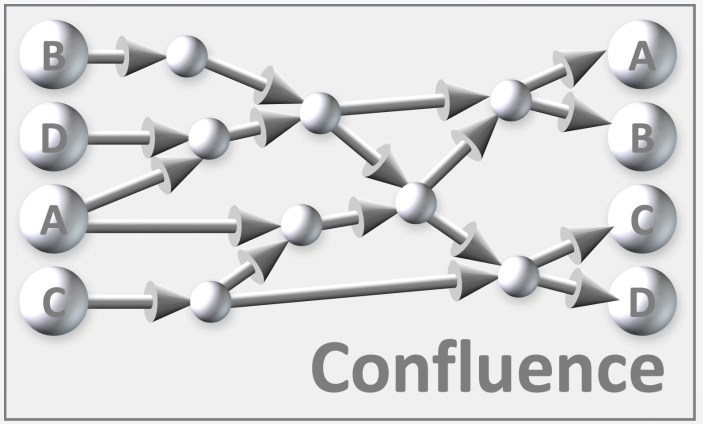




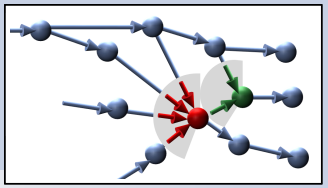




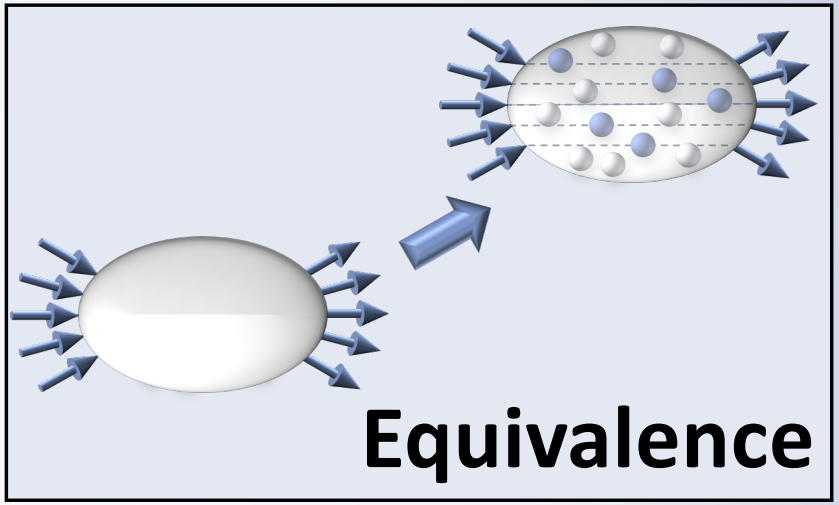




Relax Coordination with Equivalence



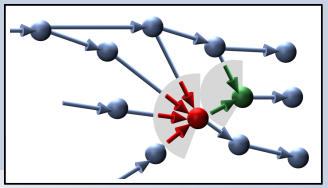
- **Good enough: Don't sweat details**
  - Updates to caches not atomic
  - Relaxed consistency to avoid coordination

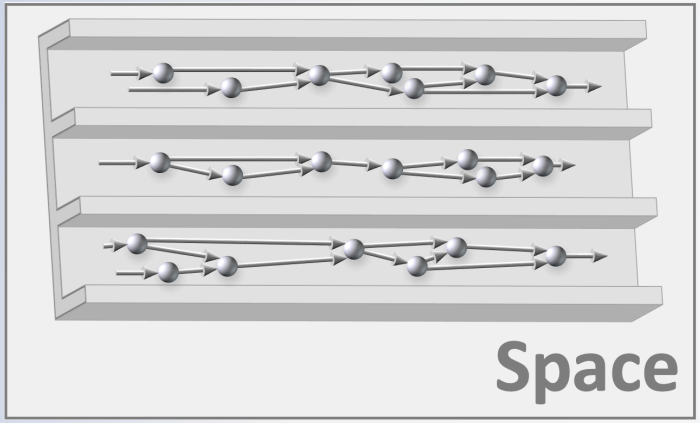


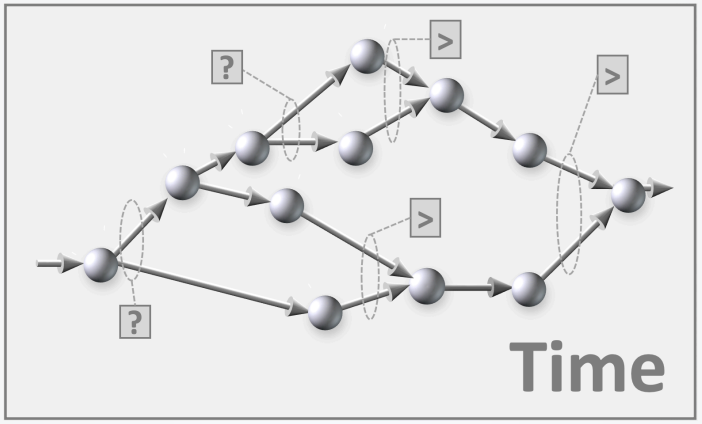
- **Equivalence: Looks the same to me!**
  - Examples: Dollar bills, bushels of wheat, king-sized non-smoking rooms
  - Must ignore differences

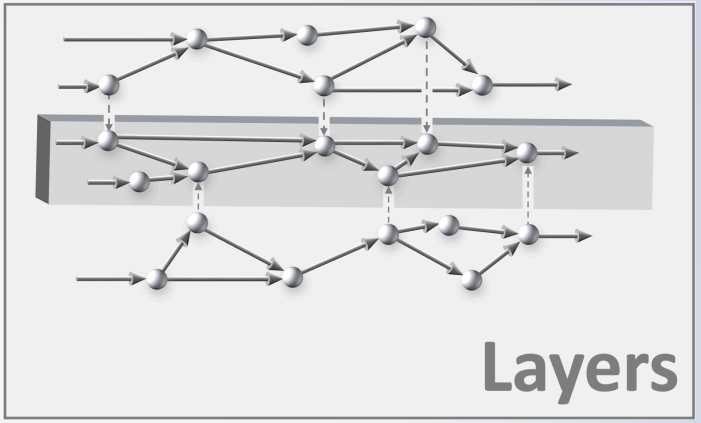
- **Duality: Private versus Shared**
  - **Private work** allocates one of the equivalent things
  - **Shared pool** must manage capacity

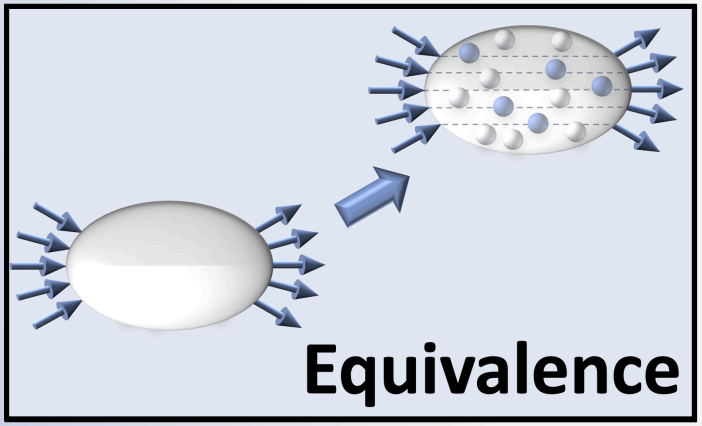
Ways to Relax Coordination

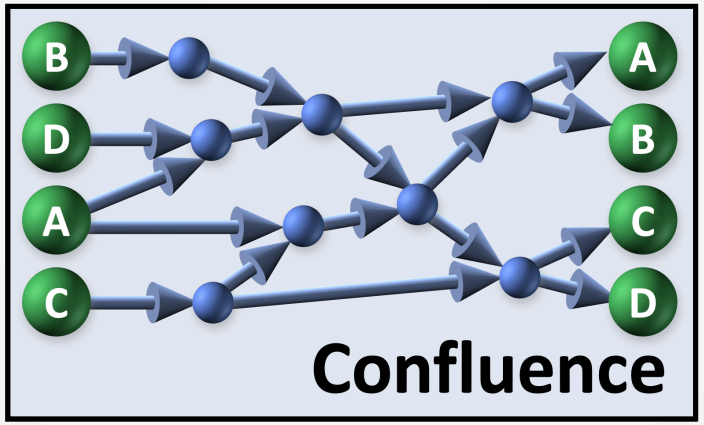




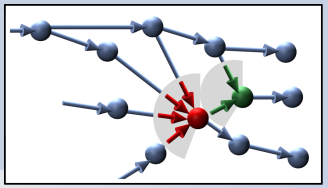




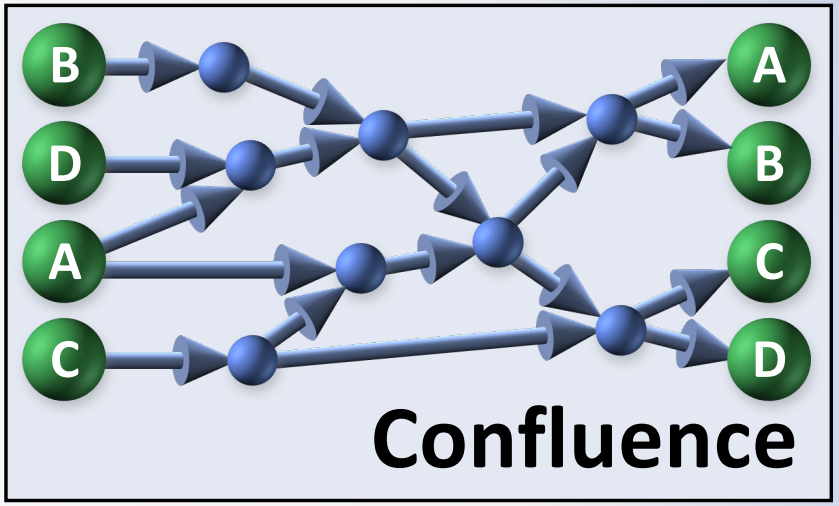




Relax Coordination with Confluence

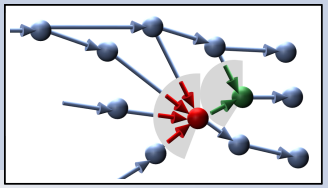


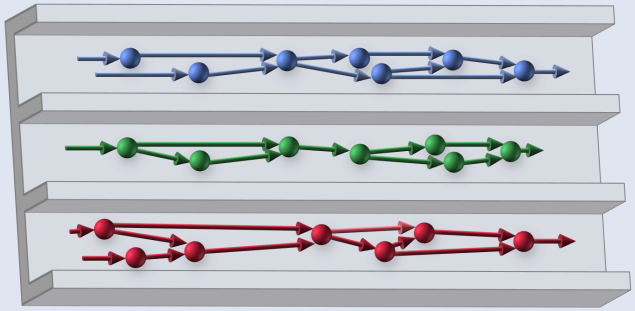
- **Confluence: Order of inputs DOES NOT affect the order of outputs!**
  - Property of SOME programs
  - Same inputs → Same outputs
  - Execution order doesn't matter!



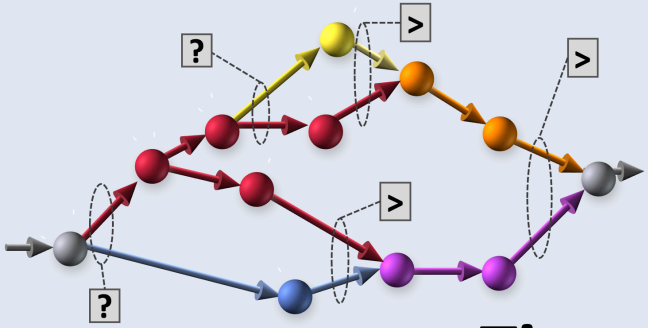
- **Confluence requires some natural order**
  - The output must be naturally ordered
- **Massively parallel work leverages confluence**
  - Break into pieces → Order across pieces doesn't matter → Execute in any order
  - Defer coordination across pieces
  - Examples:
    - *MapReduce, sort, SQL set operations*

Ways to Relax Coordination

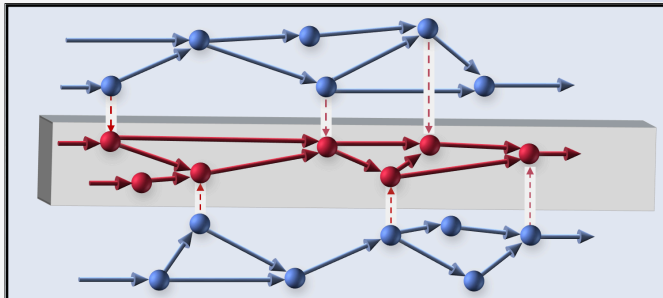




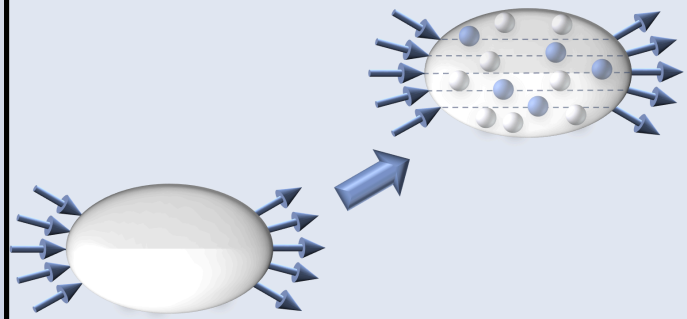
Space



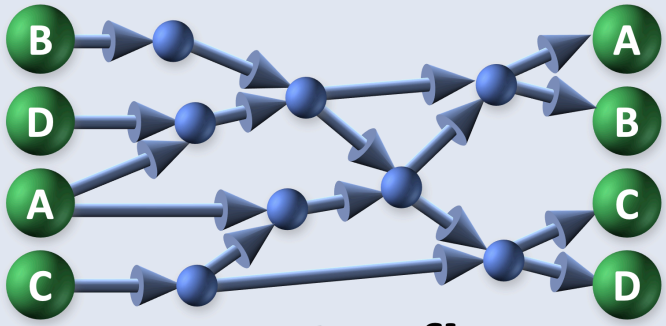
Time



Layers



Equivalence

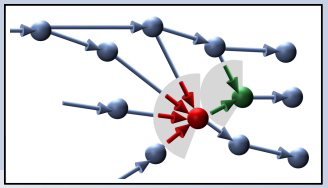


Confluence

These Are Just the Ways I've Imagined

Do you know any more???

Combining Relaxation Techniques (1)



Yes	**Space** (Buffers)
Yes	**Time** (Pre-allocation)
Yes	**Layer** (MALLOC)
Yes	**Equivalence** (Pointers)
No	Confluence

Per-Thread Memory Pools

- Contention for shared memory too painful
- Create per-thread memory pool
- Allocate big blocks from shared memory per-thread

Yes	**Space** (Shards/Items)
Yes	**Time** (Retries for latency)
Yes	**Layer** (Items in Cache)
Yes	**Equivalence** (Stale OK)
No	Confluence

Massive Replication of Caches with Asynchronous Updates

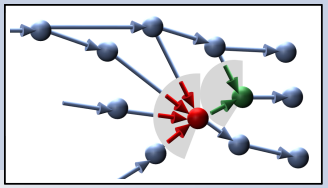
- Examples: ECommerce product descriptions and/or Search NGram-shards
- Updates happen from batch jobs across many replicas
- Read from any replica & retry to another other on timeout
- Answers may vary when retried

No	Space
Yes	**Time** (Speculation)
Yes	**Layer** (Caches vs. Execution)
No	Equivalence
No	Confluence

Reduce CPU Core Cache Misses

- Speculatively predict instruction branches
- Fetch cache lines that MIGHT be needed
- Example: Apple M1 chip 600-800 out-of-order instructions

Combining Relaxation Techniques (2)



Yes	**Space** (Page/record locking)
Yes	**Time** (Pre-caching data)
Yes	**Layer** (Record/block & more)
Yes	**Equivalence** (Memory alloc)
Yes	**Confluence** (set operations)

Record oriented relational databases

- Record locking (transactions) & page locking (internal)
- Caching of data inside the database
- Set oriented operations are disordered in execution
- Allocate big blocks from shared memory per-thread

Yes	**Space** (shards)
Yes	**Time** (Retries to replicas)
Yes	**Layer** (interface to shard)
Yes	**Equivalence** (shard contents)
Yes	**Confluence** (doc set ops)

Internet search operations

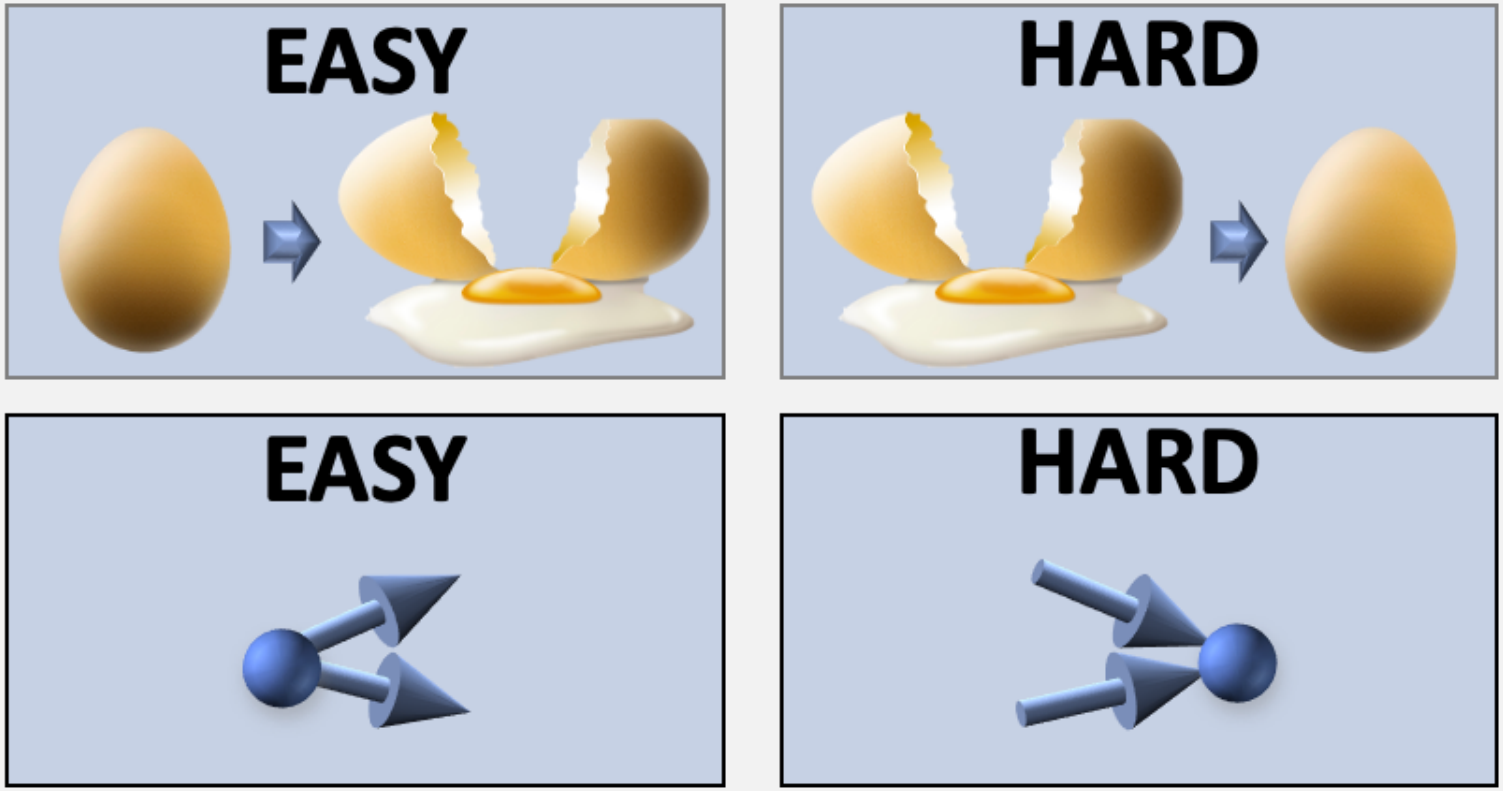
- Calculate NGrams of search terms
- Ask search server shards for docs matching NGrams – *"pretty good result"*
- Retry to shard replicas for latency (Tail at Scale) – *"equivalent result"*
- Intersect docs matching NGrams desired

Yes	**Space** (Application & DB)
Yes	**Time** (Batched changes)
No	Layer
No	Equivalence
No	Confluence

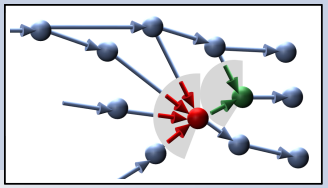
Transactional database optimistic updates

- Application reads database and plans updates to make
- Only if no collision are the updates committed
- Reduces trips to the database by bundling many updates into one trip

It’s Expensive to Decrease Disorder



It's Expensive to Decrease Disorder



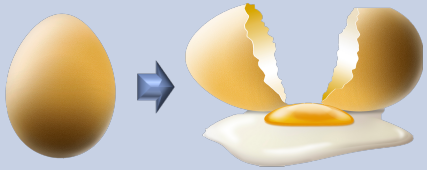
Paraphrasing the Second Law of Thermodynamics

"Disorder in the universe continues to increase"

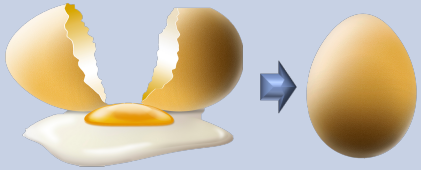
Takes Energy to Reduce Disorder

Less Disorder "Here"
Takes Energy "There"

EASY



HARD

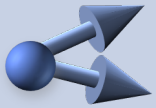


Coordination Is Hard!

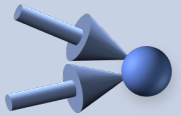
Takes Energy to Decrease Parallelism

Aligning Parallelism Incurs Costs in the System

EASY

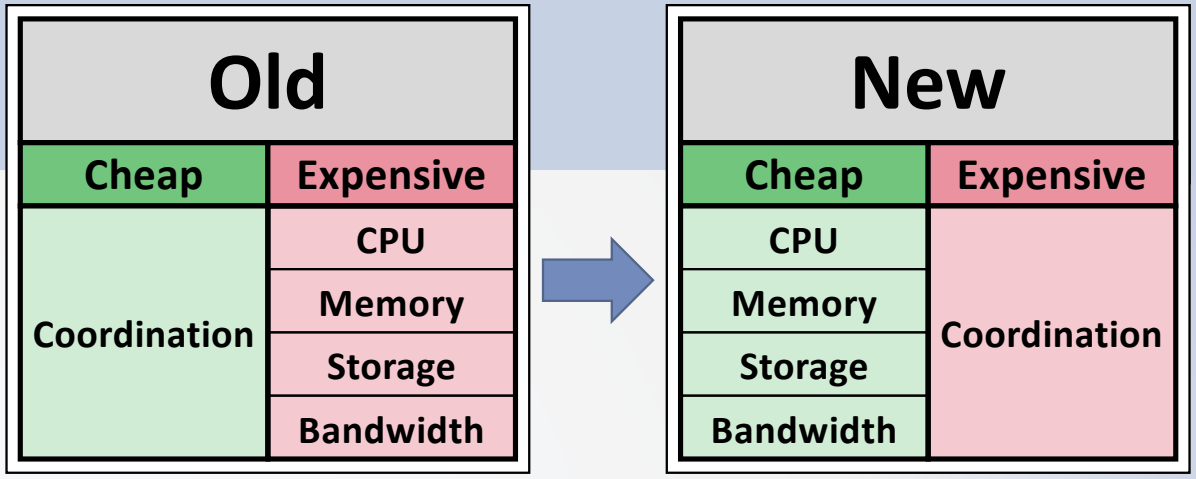


HARD



Don't Coordinate When You Can Avoid It!

Takeaways



- **Computing has changed**
  (and is still changing)
- **Our solutions are changing**
  - ***READ:*** *LOTS of immutable data*
  - ***WRITE:*** *Logs and log-structured*
  - ***COORDINATE:*** *Hard to pick authority & hard to see the latest truth*
- **Coordination: Partial collapse of partial order**
  - **HAPPENED BEFORE:** Partial order in distributed systems → *Parallel execution*
  - **COORDINATION:** Combine parallel stuff to execute together

Coordinate Less Often: *Keep Stuff Isolated in Some Dimension(s)*

Space	Time	Layer	Equivalent	Confluence
• *Shards* • *Buffers* • *Threads* • *Companies*	• *Speculation* • *Retries* • *Optimistic Concurrency*	• *Record vs Block* • *Internal Details* • *Microservices?* • *Modules*	• *King Size Room* • *MALLOC* • *Bushel of Wheat*	• *Leverage existing order* • *Execution may be disodered*

The End