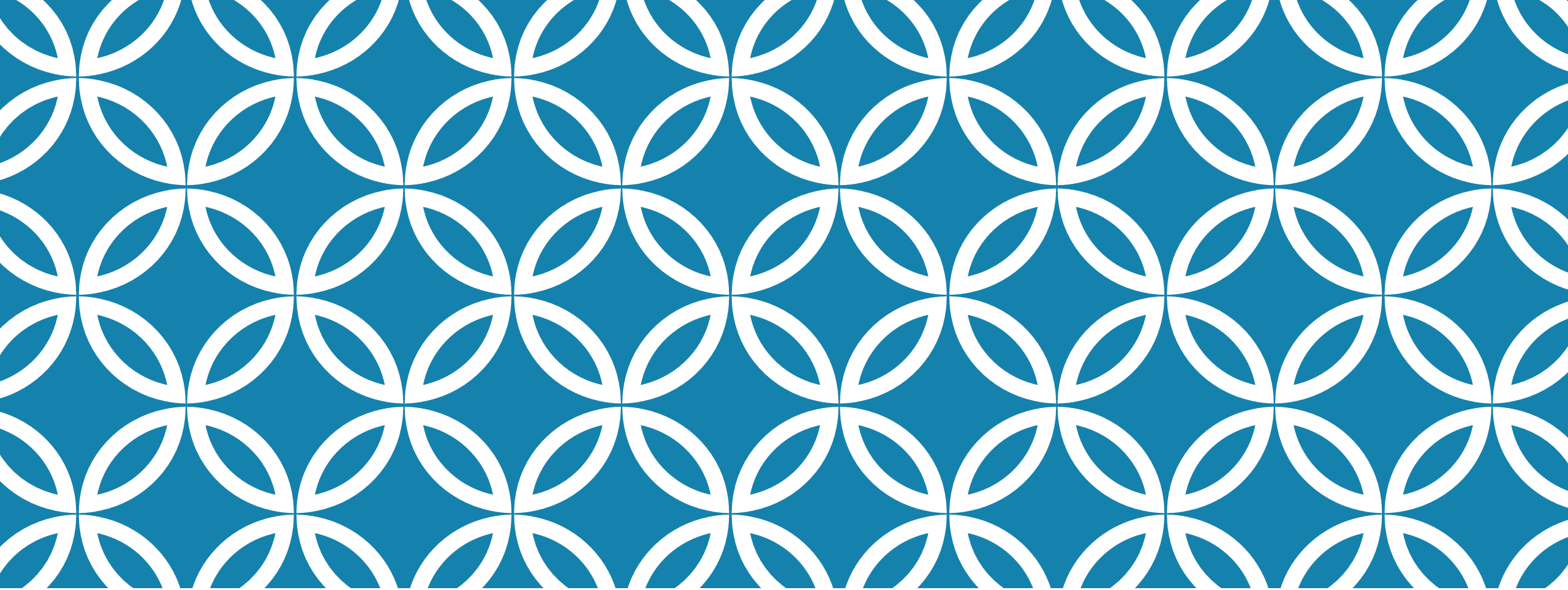


THE CHALLENGES OF INTRODUCING RDMA INTO CLOUD DATACENTERS

CS4414/5416
Fall, 2025

CONTEXT FOR THIS LECTURE

We saw how the need for performance has pushed some very fancy machine learning components closer to the edge, like Facebook TAO. Hardware accelerators improve performance and reduce costs for these components.

As we connect the cloud to sensors, we'll get an even greater demand for real-time updates (hence replication), consistency and coordination at the edge. Cascade and Derecho are examples of a response to that need.

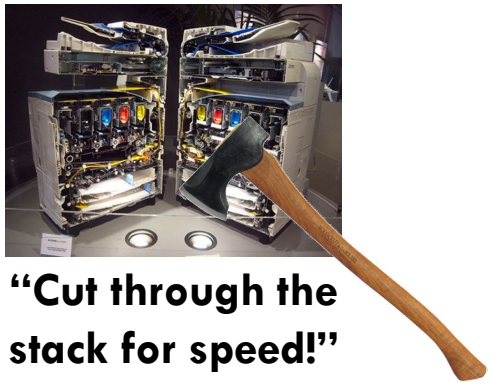
CONTEXT FOR THIS LECTURE

Then we learned about how accelerators of all forms are central to performance and also eco-cost-effectiveness in the cloud.

In the case of Derecho this speed centered on RDMA.

But the real edge with the actual sensors will be 5G. And even inside the cloud itself, Derecho would often live on an edge-cloud, like Azure IoT. Can we just deploy RDMA everywhere?

LIFE ON THE EDGE



"Cut through the stack for speed!"

The edge demands disruptive changes.

… early adopters tend to experience a lot of pain.

Nothing works… the hardware may lack programming tools… is undocumented… may even have *hardware bugs*. And "cutting through the stack" may have unexpected consequences elsewhere.

- None of the rosy predictions are as easy to leverage as you might expect.
- Hint: Start by duplicating some reported result for the same setup!

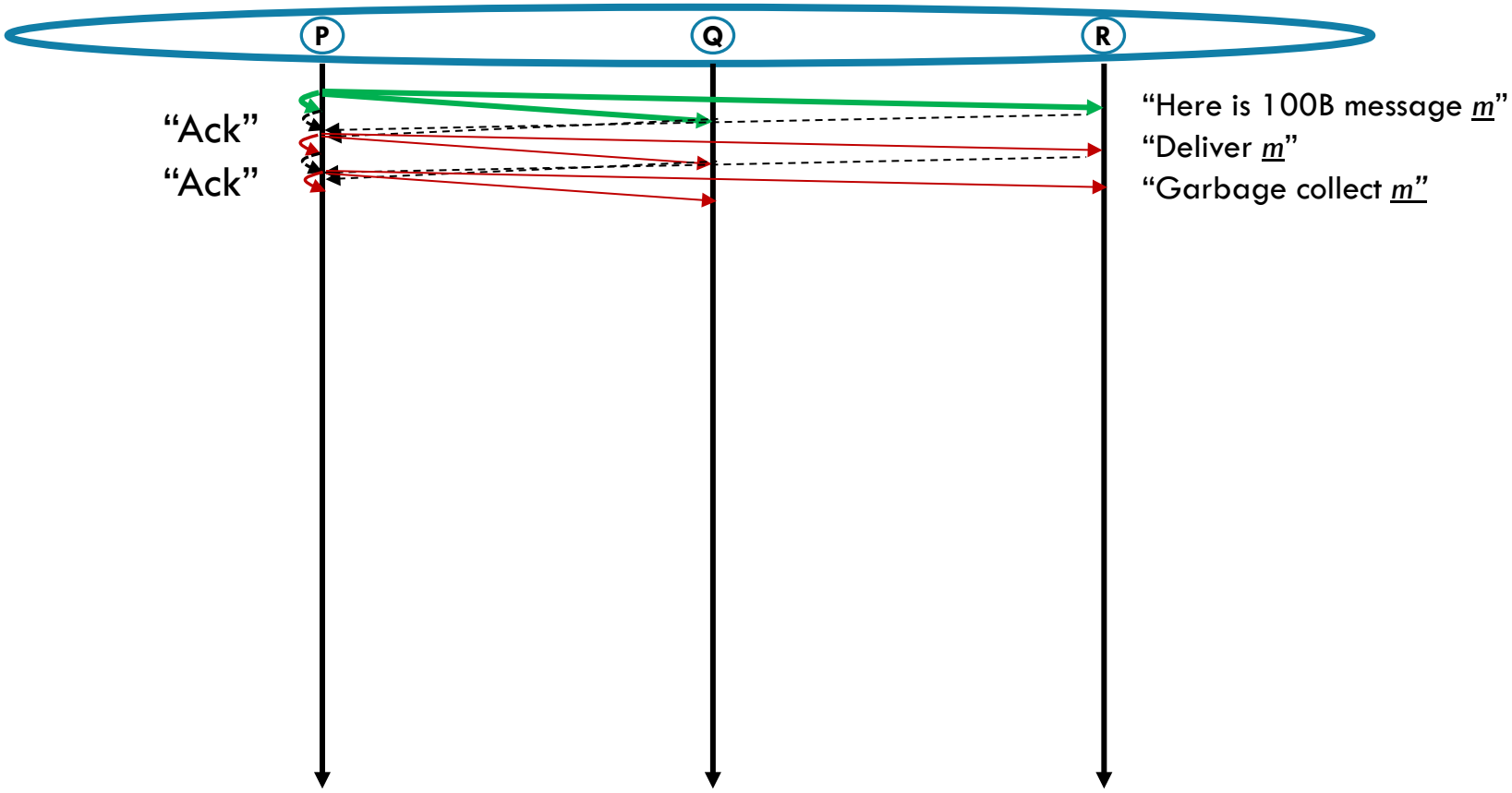
RECAP OF SOME (OLD) LECTURES

By now we've seen accelerators a few times, such as Derecho on RDMA

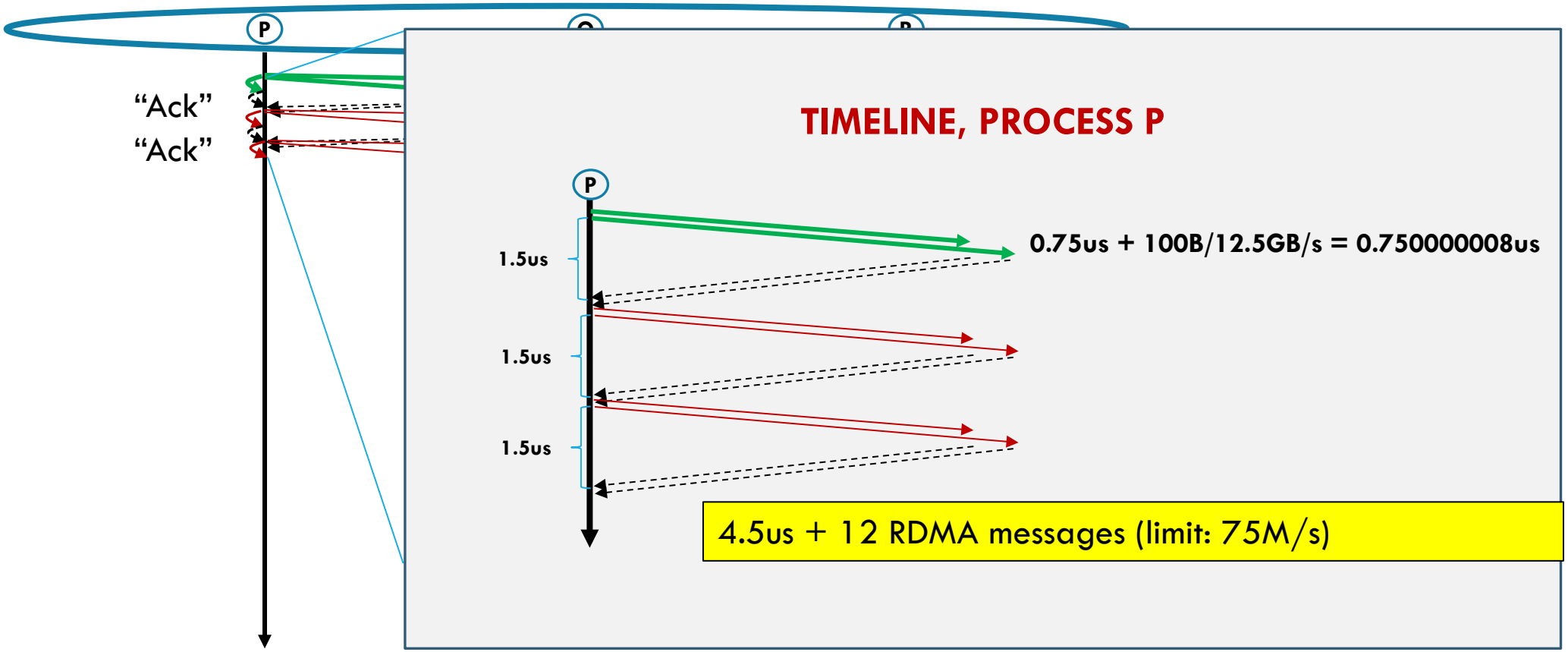
We'll do a quick refresh just to remind ourselves how hard it was to move Paxos to an RDMA framework.

Not a new lecture on Derecho, just a quick peek at some slides from early in the semester

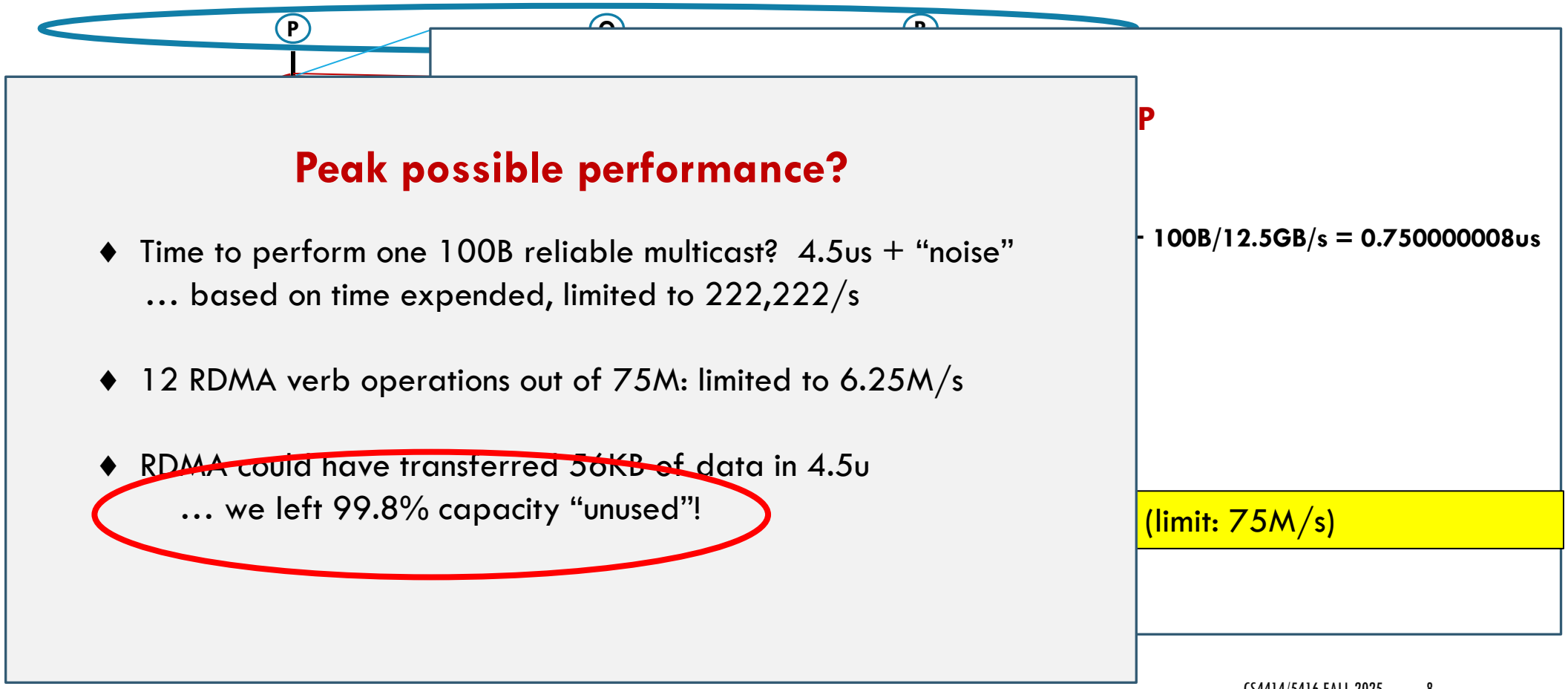
INTUITION: CONSIDER A SIMPLE RELIABLE BARRIER PROTOCOL ON RDMA



A SIMPLE RELIABLE PROTOCOL ON 100G RDMA



A SIMPLE RELIABLE PROTOCOL ON 100G RDMA



Peak possible performance?

- Time to perform one 100B reliable multicast? 4.5us + "noise"
  … based on time expended, limited to 222,222/s
- 12 RDMA verb operations out of 75M: limited to 6.25M/s
- RDMA could have transferred 56KB of data in 4.5u
  … we left 99.8% capacity "unused"!

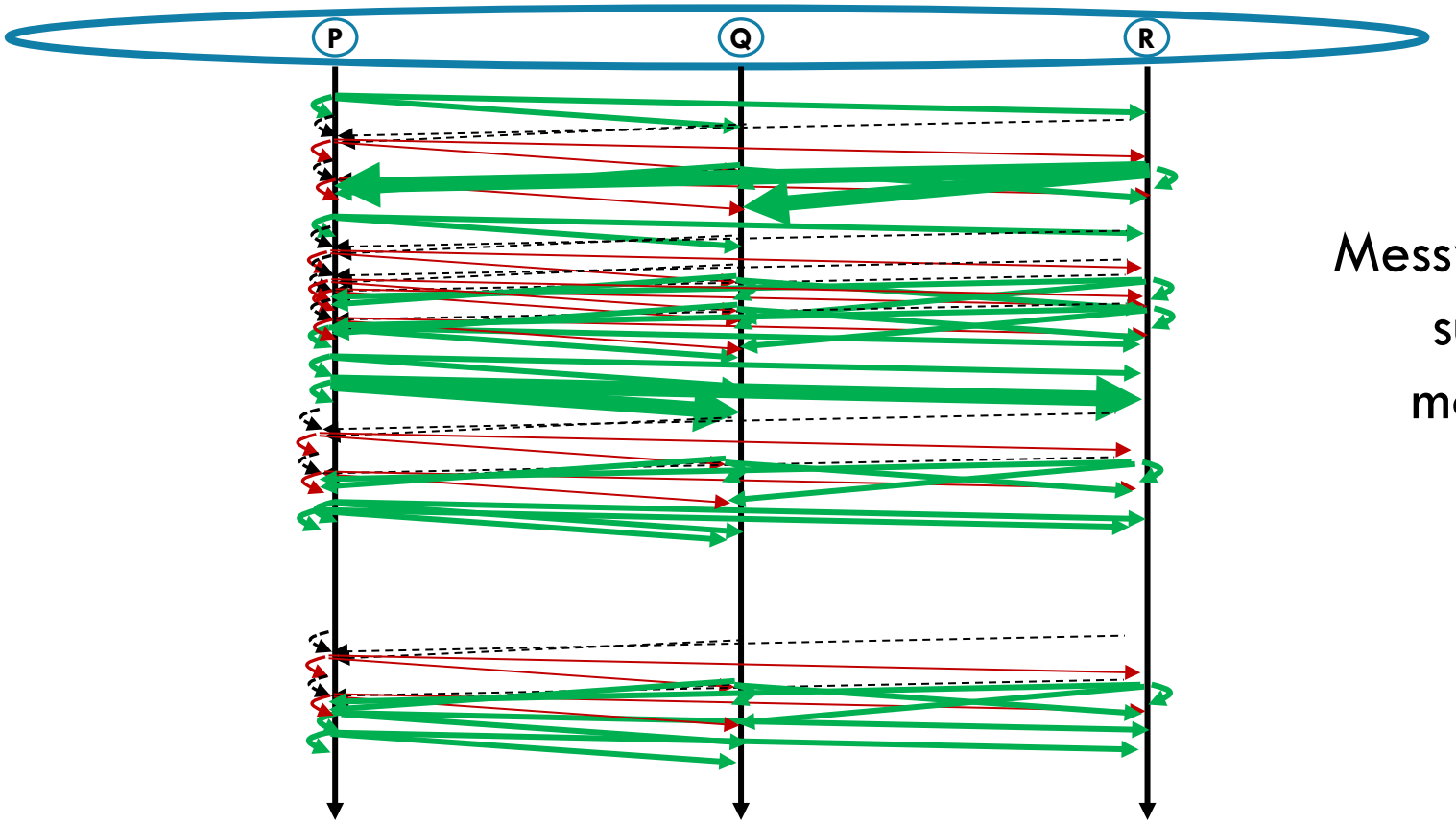
A FEW IDEAS

Have all the 3 members perform concurrent updates… now we might get some overlap and push our efficiency… to 0.6% ☹

Run lots of threads… maybe 10 per process. We aim for 6% efficiency (but locking and scheduling delays will cut this sharply) ☹

Batch 1000 messages at a time. ☺ But now the average message waits until 500 more have turned up. Latency soars to 2.25ms ☹

AT BEST, YOU GET SOMETHING LIKE THIS…



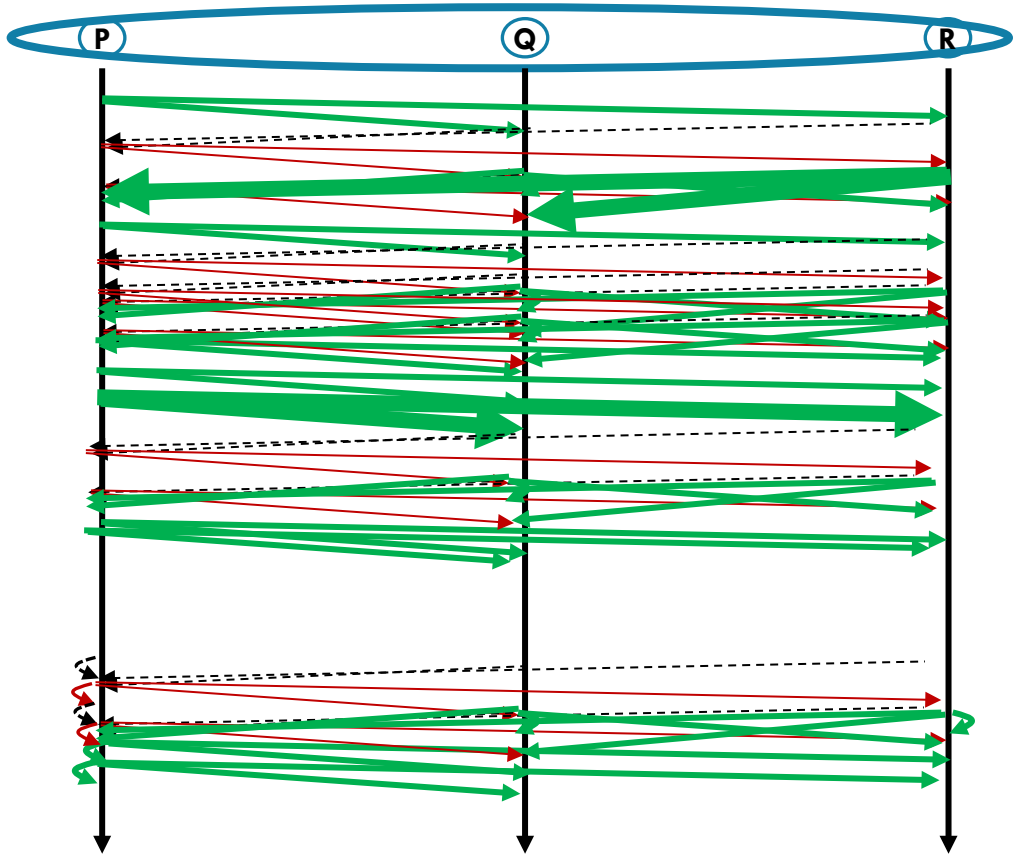
Messy and unpredictable with sudden bursts of data movement… Unlikely to perform well ☹

BETTER: SEPARATE DATA PLANE AND CONTROL PLANE, MAKE THEM LOCK-FREE

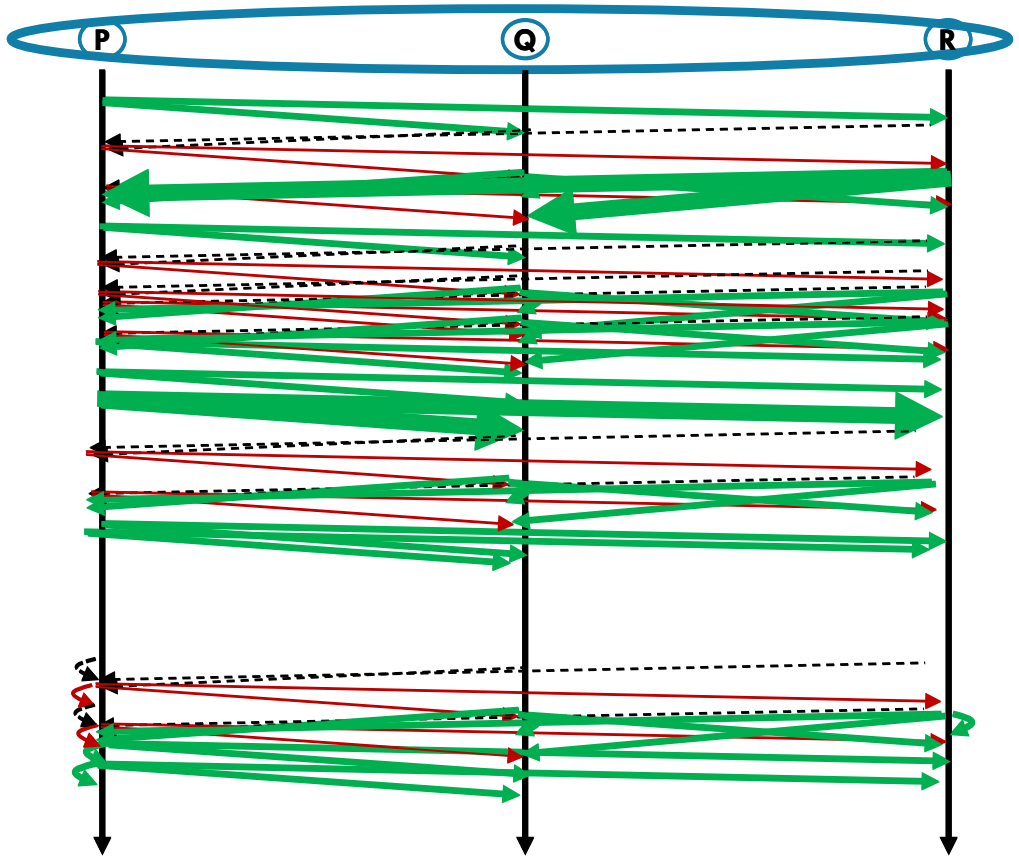
Send continuously, as soon as new updates show up

Receivers continuously report their acks, in an all-to-all pattern. This way every process can deduce delivery/garbage collection.

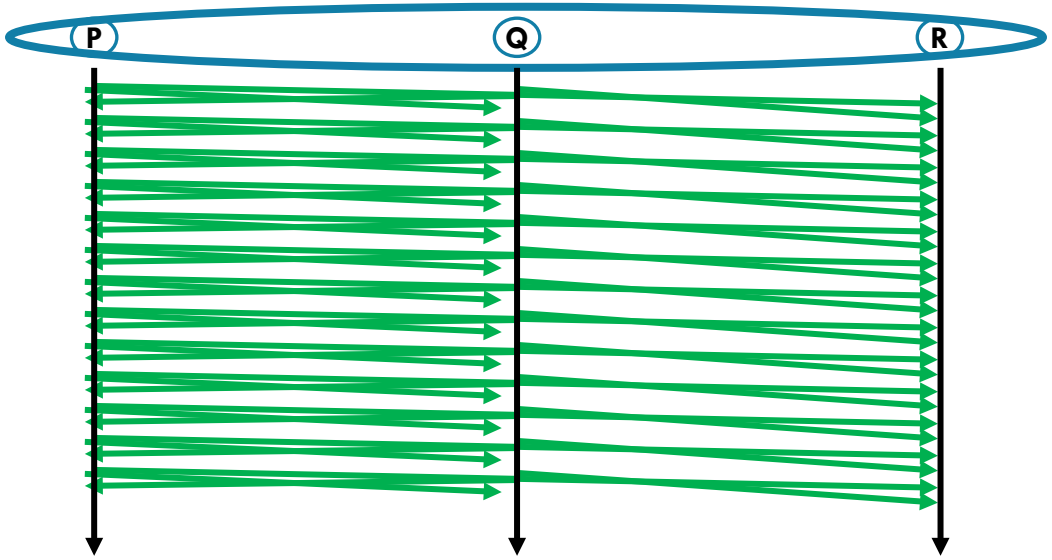
BETTER: SEPARATE DATA PLANE AND CONTROL PLANE, MAKE THEM LOCK-FREE



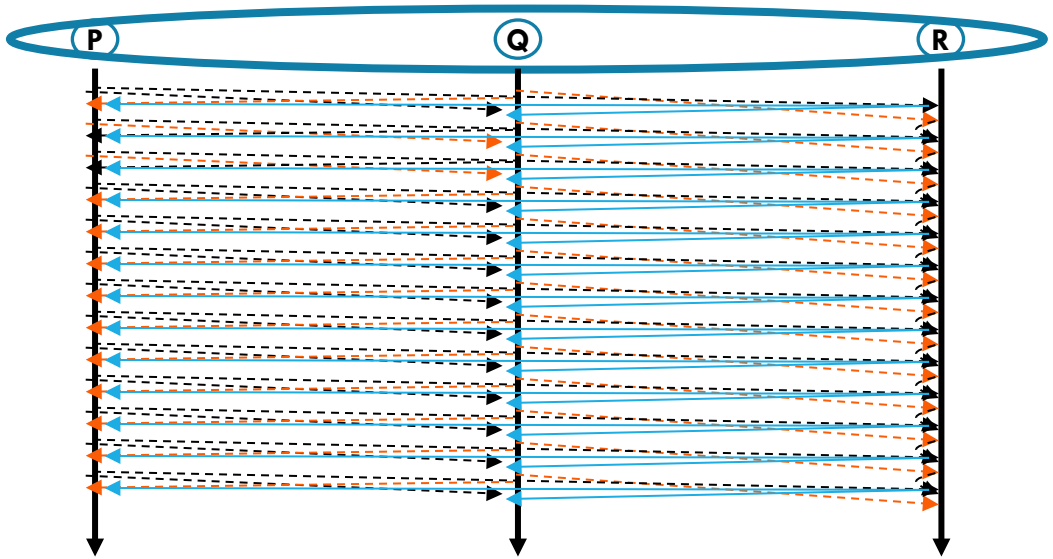
BETTER: SEPARATE DATA PLANE AND CONTROL PLANE, MAKE THEM LOCK-FREE



BETTER: SEPARATE DATA PLANE AND CONTROL PLANE, MAKE THEM LOCK-FREE

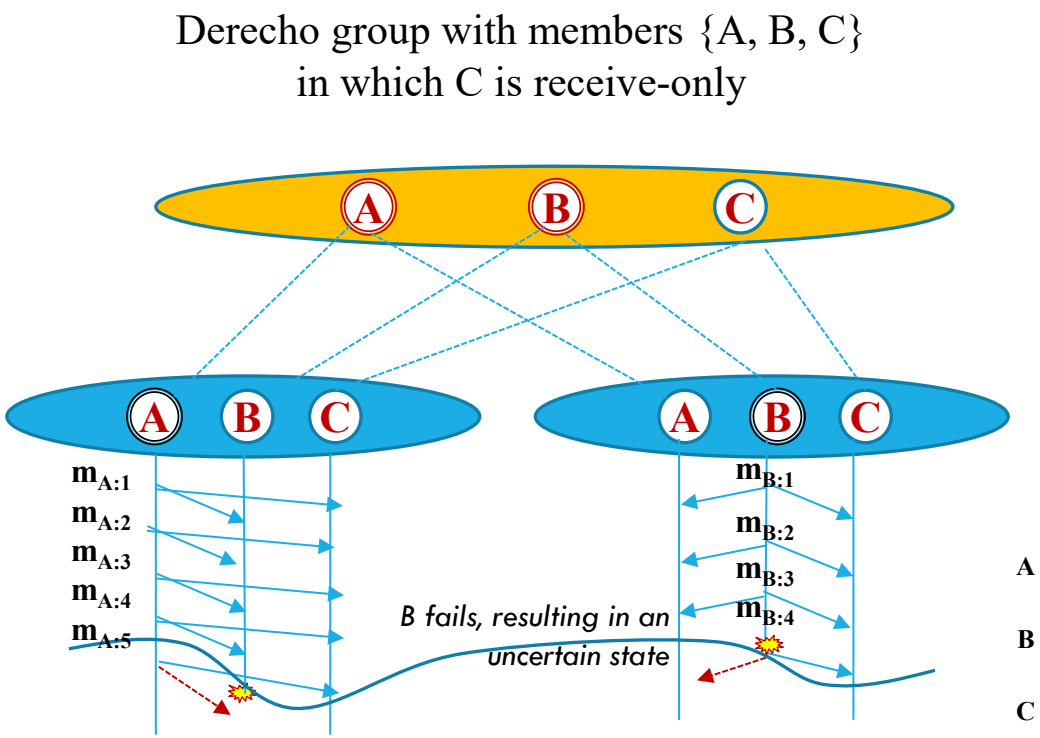


Data plane runs steadily (Derecho's RDMC or SMC multicast)



Control information exchanged continuously (one-sided RDMA writes via SST)

DERECHO'S DATA PLANE = RDMC/SMC. DERECHO'S CONTROL PLANE = SST



	Suspected			Proposal	nCommit	Acked	nReceived		Wedged
A	F	T	F	4: -B	3	4	5	3	T
B	F	F	F	3	3	3	4	4	F
C	F	F	F	3	3	3	5	4	F

Data moved on RDMA multicast

Control is done using knowledge programming on the SST

THIS IS NOT AN OBVIOUS WAY TO PROGRAM!

Notice how ***the hardware forced us to program differently:***

- The hardware is very fast, but only if used in a certain way
- To use it in that way, at that speed, we couldn't do "normal" things,
  like sending messages and waiting for acknowledgements, or votes
- So we had to invent this new shared table abstraction, and had to
  rewrite the standard Paxos protocols in a totally new way

… NOT UNUSUAL WITH NEW HARDWARE!

New hardware often results in ideas like Derecho

Specialty hardware can be extremely fast, but often requires that you
use it in some very unfamiliar way.

If we just run the old style of algorithm on the new hardware, but in the
old way, we wouldn't benefit

… OR EVEN SOME OLD HARDWARE

After building it, we realized that Derecho is actually faster on TCP too, although not quite as fast as with RDMA.

This is because modern TCP in a datacenter is incredibly fast, only about 4x slower than RDMA if you use it "just right" (TCP won't hit this rate out of the box, it takes a lot of tweaking the application to get those speeds)

Also, TCP has pretty high "lowest delay" numbers (latency)

INITIAL CONCLUSION

We managed to make Derecho very fast, but to do so:

- Had to come up with a way to move bytes at crazy speed.
- Then had to come up with a "control plane" that can run separate from the data plane.
- Turned out it needed a lot of 2PC kinds of mechanism. To get those to be fast we invented this whole way to program the SST.
- A lot of work, but the payoff was extreme speed.

DERECHO: THE REMAINING PUZZLES

To get the full speed of the technology

- You need to work in C++. But many people prefer Python, Java…
- Programs need to be "zero copy". But most people have no idea if the packages they use do copying
- Your code needs to be nearly lock free and rather pipelined. Few people are used to coding this way

Is it worth it? Derecho is as much as 15,000x faster than other options…
but only if you use it properly!

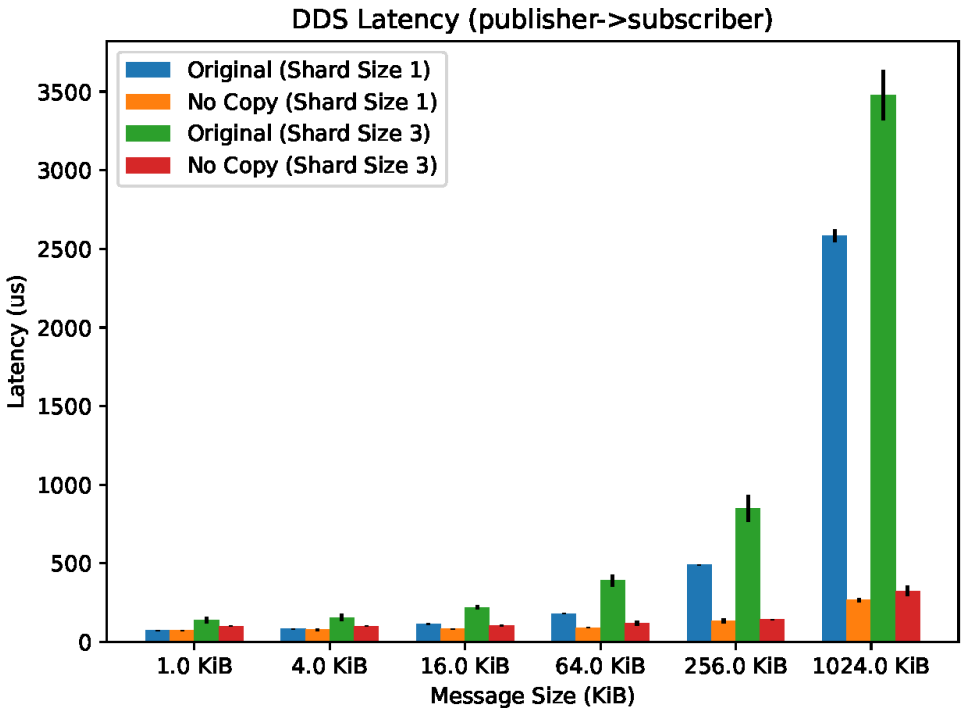
ZERO COPY REQUIREMENT

Moreover, to get the full possible speed of Derecho, you need to write code that won't involve any copying (even using memcpy, or even automated copying done in the programming language runtime).

Copying is slower than RDMA!

This is quite tricky: Your application needs to use RDMA "everywhere" for large objects where performance will be critical.

ZERO COPY OPPORTUNITY



This shows two runs of the Cascade DDS (message bus) with various numbers of servers (shard size) and various messages sizes.

In each case we just measured delay from when a message is published to when it is received.

Yellow and red used a zero-copy approach. This one change dropped 1MB delay from 3.1ms to just 230us: a 13.5x speedup!

… THERE ARE MORE PUZZLES, TOO

Modern DRAM is internally concurrent: each cache line is independent.

If we write, say, 1MB we are writing into 156,000 cache lines. Intel guarantees "total store order" (TSO) but ARM and AMD are different.

When the RDMA sender is told "send complete", have these writes really finalized? And will the remote system's cache coherency hardware ensure that if an application looks at that message, it will see the new bytes?

MEMORY FENCING

A memory fence tells the CPU that memory may have been updated and cached data must be refreshed.

An example: lock acquire and release are memory fenced. Suppose Y is data in the Derecho SST, and X guards Y. We update Y, then flip X to true

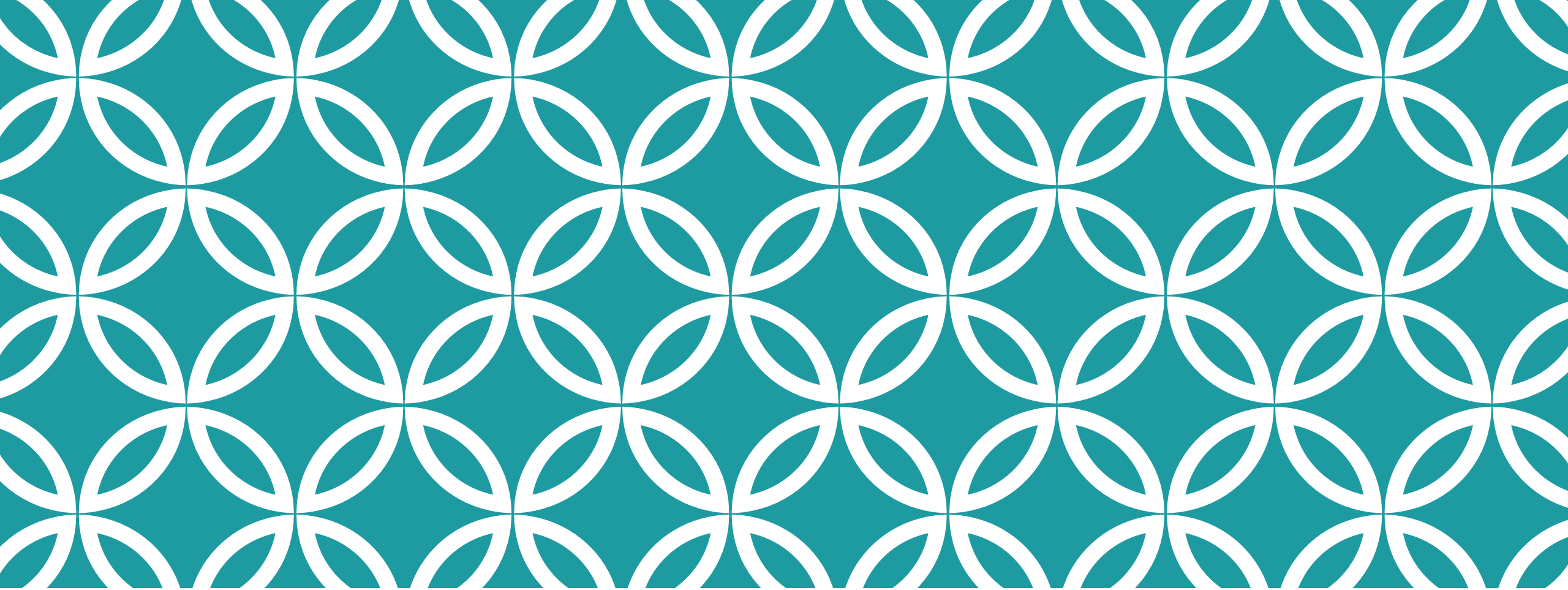
The reader sees X become true. Will it see the new Y? Without a fence, the reader might see a corrupted version of Y due to old cached data.

OTHER ACCELERATORS: SIMILAR STORIES

GPU units require entire special programming languages (CUDA) and a really peculiar programming style (move object into GPU, load program, press “run”, move results back into CPU memory)

FPGA accelerators have to be coded in a gate-level language, like the ones used for VLSI chip design.

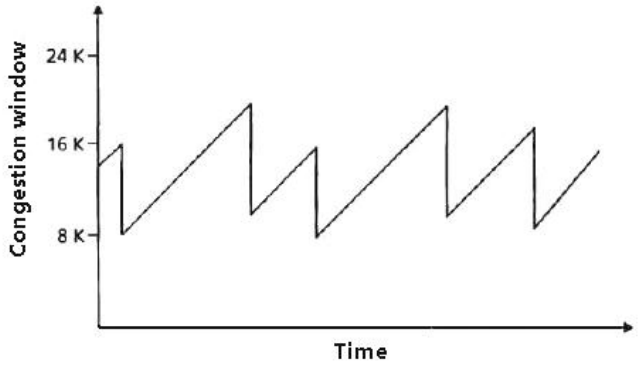
Network interfaces can be programmed, but the run a very strange kind of code focused on moving messages (P4, but it isn’t yet a standard).



NOW WE KNOW ALL ABOUT DERECHO.

Should everyone switch to it in all their edge systems?

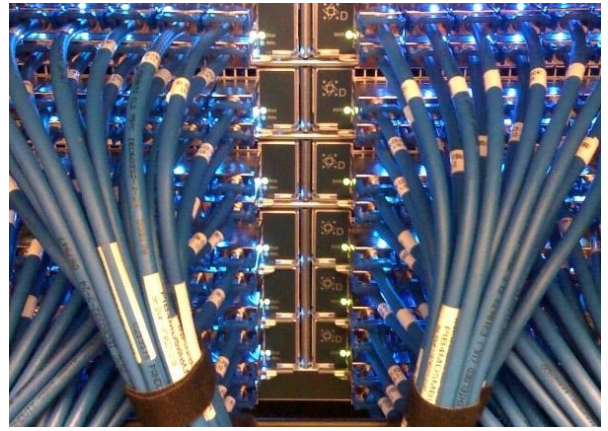
THERE IS A SMALL ISSUE…



Not so fast… RDMA doesn't really work in datacenters! A long history…

- RDMA was invented in connection with a novel networking approach called Infiniband. It competes with optical ethernet
- In ethernet, senders send packets, and packets are dropped if congestion (overload) occurs.
- This causes loss if the packets are part of UDP messages, but TCP retransmits missing chunks.
  - It "backs off" (slows down) if loss occurs
  - Additive increase, multiplicative backoff

INFINIBAND ISN'T ETHERNET



In Infiniband, no device sends unless it has permission to send from the receiving device first

So when a router transmits a packet to another router, for example, the receiver has granted "credit" for the sender to send B bytes.

This is true for every step along the route!

- Hop by hop, no data is moved without assurance of a place to put it
- The optical network layer is so reliable that Infiniband is **lossless!**
  - More precisely: Loss is incredibly rare and usually caused by some form of crash

SUPERCOMPUTERS LOVE INFINIBAND

The “market share” for Infiniband in HPC systems is extremely high

Ethernet is unpopular because those packet drops aren’t rare, and this causes erratic performance.

So we now have 15 years of experience with Infiniband with as many as hundreds of thousands of datacenter computers!

RDMA was born in this world: DMA transfer over Infiniband works because hop by hop, no loss ever occurs. Every piece of data is moved reliably at “optical network” speed. Today: 200Gbps (bi-directional) is available

- In contrast, memcpy with a single core is more like 36 Gbps for large transfers that can’t leverage the L2 cache.

RDMA ON CONVERGED ETHERNET: RoCE

The idea emerged of running RDMA over optical ethernet, around 2010

Puzzle: RDMA doesn't do retransmits over Infiniband, and a full TCP-style
solution wouldn't be nearly as fast

So, how to get RDMA to run in a setting without sender credits?

They introduced a concept of "Priority Pause Frames"

RDMA ON CONVERGED ETHERNET: RoCE

The idea emerged of running RDMA over optical ethernet, around 2010

They introduced a concept of "Priority Pause Frames"

- An overloaded router or switch or NIC sends PPFs to the sender of a flow if it becomes overloaded by incoming data
- The RDMA NIC pauses, then restarts the transfer (the entire transfer) if it receives even a single PPF

… but unfortunately, PPF didn't work very well

- It can generate PPF "storms" and RDMA performance collapse

"BUT DOES ROCE WORK??"

There are many stories of datacenter technologies that didn't work well

They include optical Ethernet multicast (broadcast), early web server technologies, early packet routing solutions, early ways of connecting browsers to web servers, early DDoS attack filters

Often, they disabled entire datacenters when they malfunctioned!

… so datacenter operators are not eager to embrace RoCE yet, because "a little unstable" can mean "my datacenter could be toast"

REMINDER: BROADCAST STORMS

Every machine suddenly receives 50,000 msgs/s

This is another slide related to an old topic

When we talked about Bimodal Multicast, we asked whether UDP multicast could accelerate gossip. But a hardware "bug" blocks can trigger meltdowns that wipe out the whole data center

Kind of like this storm over Austin Texas…

COULD RDMA TRIGGER SUCH A STORM?

No, not in a literal sense

- The multicast storm was caused by a feature of routing and NICS specific to the way that Ethernet class-D multicast forwarding is done
- In fact the "cause" is that a hardware hash-table fills up and overflows (underlying limitation: "Bloom filter" hash tables are too small)

But in the larger sense, the story is about how a technology used by one subsystem can overwhelm the whole datacenter and disrupt other systems

- So you need to ask: ***"Could enabling use of RDMA disrupt the cloud?"***

TURNING THIS INTO A TECHNICAL QUESTION

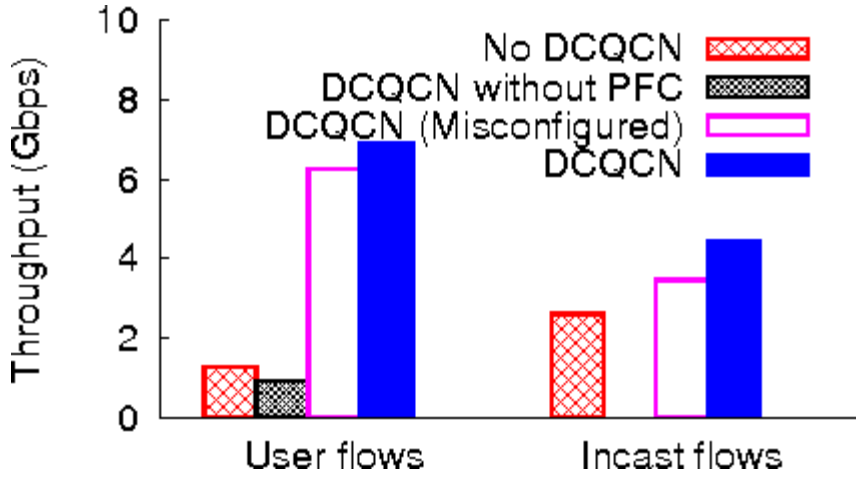
The cloud is stable because TCP congestion control is stable.

RDMA doesn't use TCP congestion control.

Does this imply that if we use RDMA heavily, our TCP traffic will be starved and our data center will become unstable?

In fact… yes. RDMA can be destabilizing in this way! Enter… DCQCN

DCQCN



Data Center Quantum Congestion Notification is a new idea from Microsoft and Mellanox that uses end-to-end congestion notification

Basically, these are the same as credits in Infiniband

Experiments on modest datacenter clusters worked well!

- RDMA + DCQCN enabled RDMA to work in normal datacenters!
- A disruptive and transformational development!

Not so fast… does it *really* work? At full scale?

MICROSOFT AZURE SET UP A RED TEAM

Goal: Deploy DCQCN *side by side* with normal TCP/IP in a new Microsoft datacenter, during the burn-in testing period (about six weeks long)

Test aggressively – try and see if they can trigger problems.

- Includes misconfiguration, but not "breaking the logic"
- They want to know: are we getting into trouble here?

Guess what? It didn't work!

ISSUES THAT WERE IDENTIFIED

RDMA and normal TCP/IP interfered with each other.

PPF standard requires "Enterprise VLAN" feature on switches. Azure doesn't use this technology.

But they solved these

- They repurposed another (also unused) feature called "DiffSrv"
- There was a DiffSrv packet format that could be reused for PPFs
- DiffSrv also allowed TCP/IP to run side-by-side with RDMA, isolated

… IT STILL DIDN'T WORK

"Jim, it dinna work. Antimatter containment will fail in 3 minutes!"

They discovered that RDMA + DCQCN + DiffSrv + PPF could cause a new kind of routing / congestion loop

It was triggered by a form of resource exhaustion because the TCP/IP layer and the RDMA layer were sharing buffers inside NICs, switches and routers.

By dividing the resources into pools, this could be solved. But their hardware lacked a way to do that. They solved it by "overprovisioning"

- They bought more memory for the routers and switches than needed
- Then configured to make sure that the memory never gets used up

MORE LIMITATIONS

RDMA only works if the application is working from "pinned" memory pages, and works best if the memory pages are huge.

- Seems to be at odds with virtualization
- Advances in RDMA hardware may help reduce these issues

When an RDMA transfer finishes, the program can definitely access the data. But if you instantly do a DMA transfer to disk, or try to display it on a graphics device, caches and pipelines may need to be flushed first.

- There is no standard way to actually do this.

MORE LIMITATIONS

RDMA NIC is directly accessed from user-space code, and has direct visibility into the datacenter network.

But very little user-space code can be trusted! Cloud vendors are in a continuous state of attack by hackers and even naïve users just after speed

So the very idea of trusting RDMA applications is a *big* source of stress! Vendors are hoping to do encryption/decryption at line rate and to virtualize RDMA within a few years – but this is hard for them to do!

IN THE END, THEY GOT IT RUNNING!

Today, Azure uses RDMA, but only for system services (for now)

Azure HPC actually does have application-layer RDMA, but only for people who use a package called MPI.

- In fact, MPI doesn't share the devices with normal TCP/IP
- Instead, Azure HPC computers have an entire extra Infiniband network

In the future, Microsoft may expand use of RDMA to allow some trusted subsystems to also use it. Probably it will never be free for general use.

… SO, CAN YOU USE DERECHO ON AZURE?

Yes, over TCP or DPDK, but not with RDMA except on Azure HPC.

Today, virtualization seems to be incompatible with RD

And even where Microsoft has RDMA, they don't allow you to access it yet. AWS does offer an accelerated data path, similar to RDMA, but we haven't experimented with it. So this is coming, but not there yet!

RDMA IS JUST ONE OF MANY "PLAYERS"

This story of RDMA as an accelerator in the cloud is just one of a few

Each technology has many hurdles to overcome!

- Is it *way* faster than not using it?
- Will it save the cloud owner money?
- Is it stable? Really stable?
- How hard will it be to "manage"?
- How hard is it to program?

EDGE VERSION OF THIS

The actual sensors are on 5G platforms. We can use a layer called the data plane developer's toolkit, DPDK, to uniformly talk to the network whether it is 5G or other technology.

And there are ways to simulate RDMA on DPDK such as "urdma", which is a library some graduate student built in Switzerland a few years ago.

But would Derecho run on such a layer? And would it make sense?

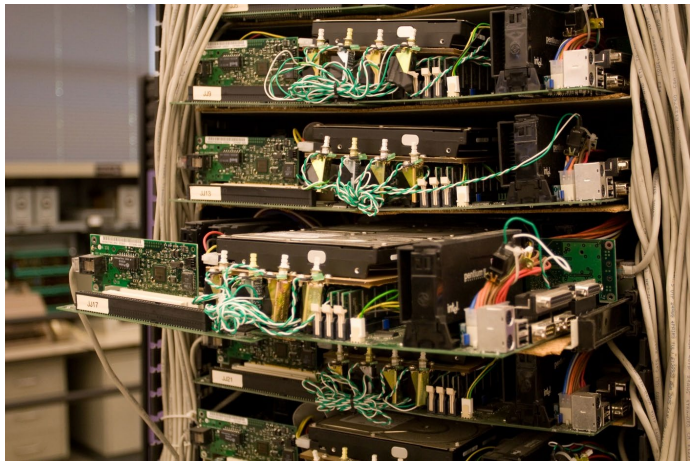
MORAL OF OUR STORY?

Modern hardware enables big steps

… but when you take them you potentially stumble.

… and even after the big step many questions still remain open!

PROGRAMMABLE COMPONENTS



General purpose cores on NUMA machines

Network Interface Card (NIC) for modern RoCE (RDMA-capable Converged Optical Ethernet). Optical network itself.

Storage components (SSD)

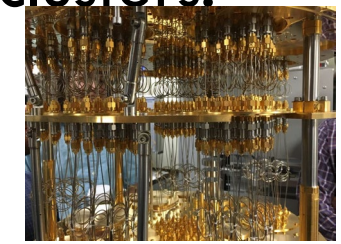
Network Switches and Routers

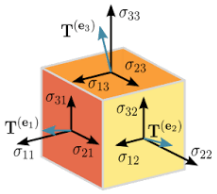
Field Programmable Gate Array (FPGA), FPGA clusters, ASICs.

GPU (graphics accelerator) and GPU clusters.

TPU (tensor processor unit) and TPU clusters.

Quantum computing hardware





EACH WOULD HAVE A SIMILAR STORY

Every one of these has amazing potential, but to leverage it can require changing all sorts of things that used to be standard in Linux

As the modern data center evolves, we will run into that issue often.

Yes, we want accelerators. But the pain level is substantial!

ROUGH ROAD AHEAD!

The latest new thing always sounds amazing…

Paradise awaits!

Road to Paradise

The pitch. What could go wrong?

The boss: Paid to say "no"!

100x Speedups have a chance of adoption, if your pain tolerance is high!