Network Support for Resource Disaggregation in Next-Generation Datacenters

Sangjin Han Norbert Egi Aurojit Panda
Sylvia Ratnasamy Guangyu Shi Scott Shenker

UC Berkeley Futurewei Technologies ICSI

Future datacenters will look fundamentally different.

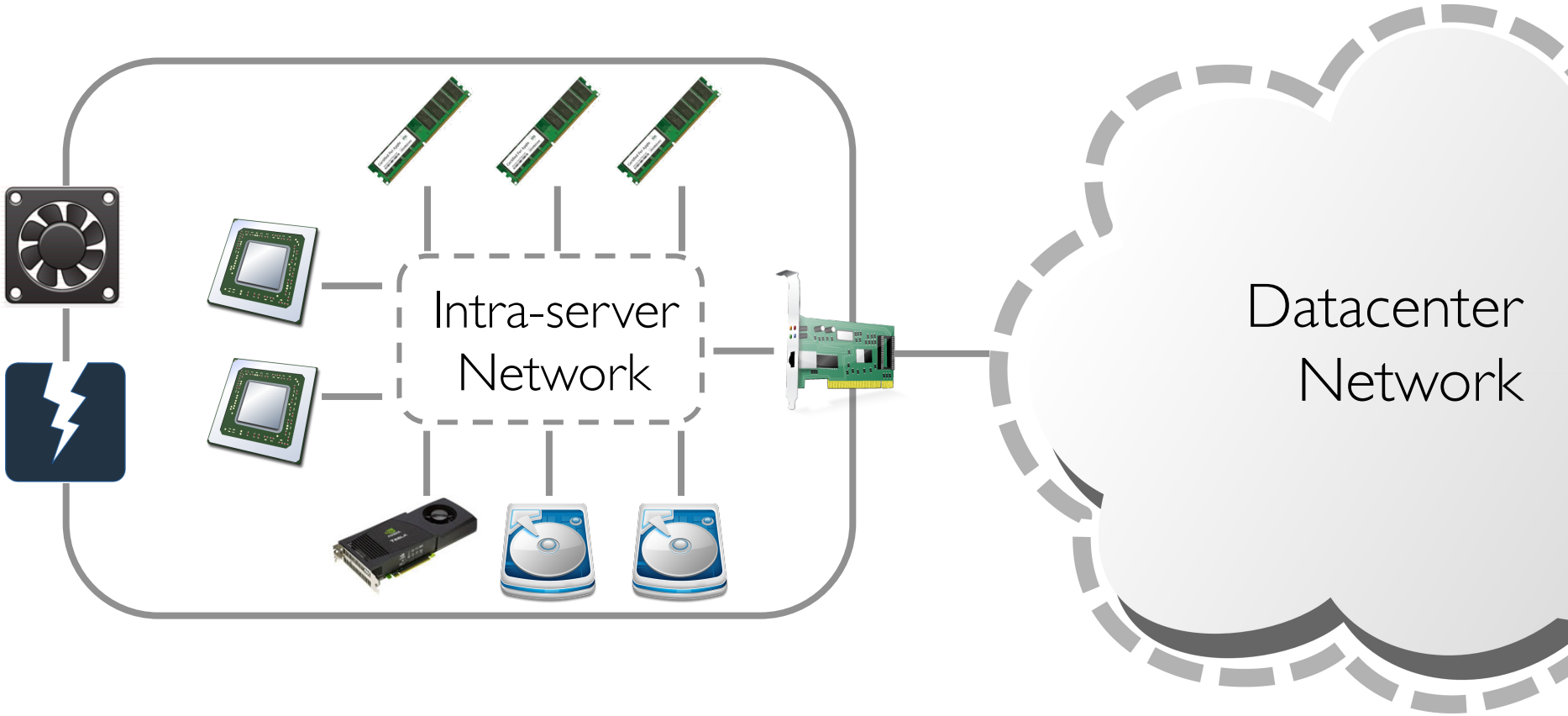
There will be no “servers”.

Like it or not.

Outline:

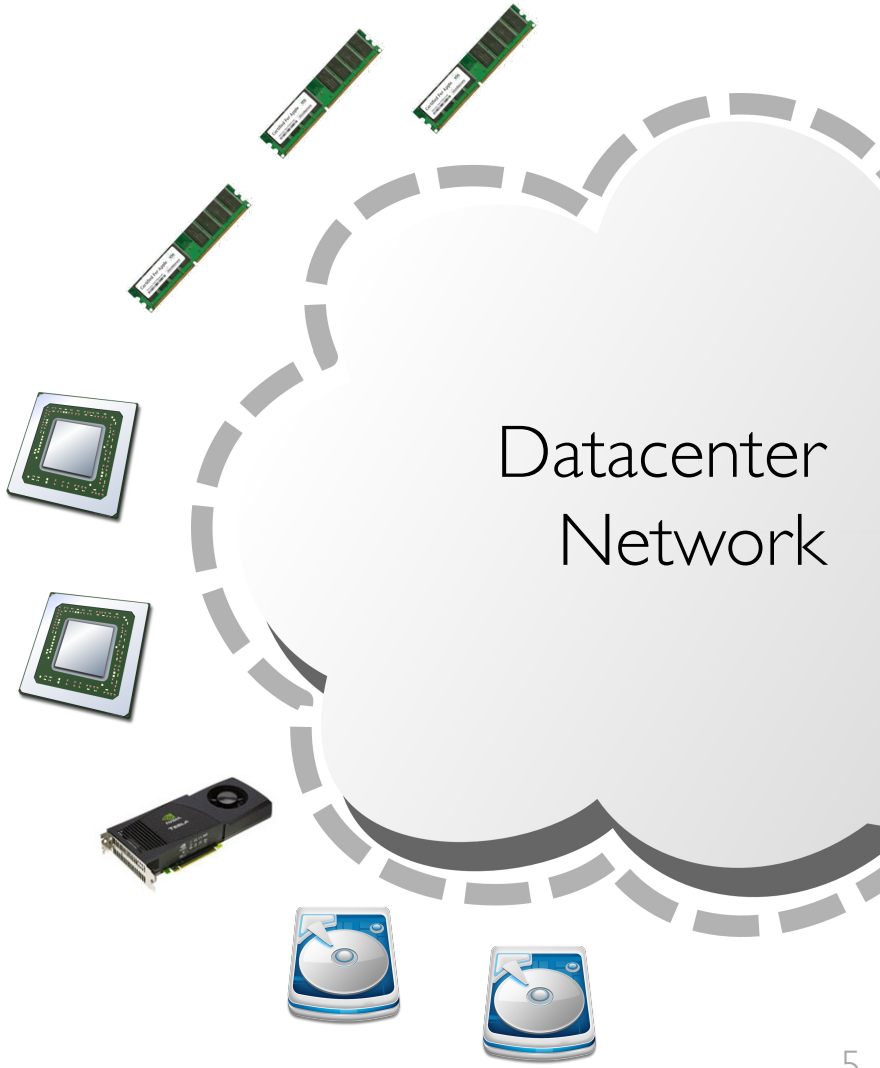
1. What will happen?
2. Why will it happen?
3. How will it happen?

Today: Server-Centric Architecture



Tomorrow: Resource-Centric Architecture

All resources are individually addressable

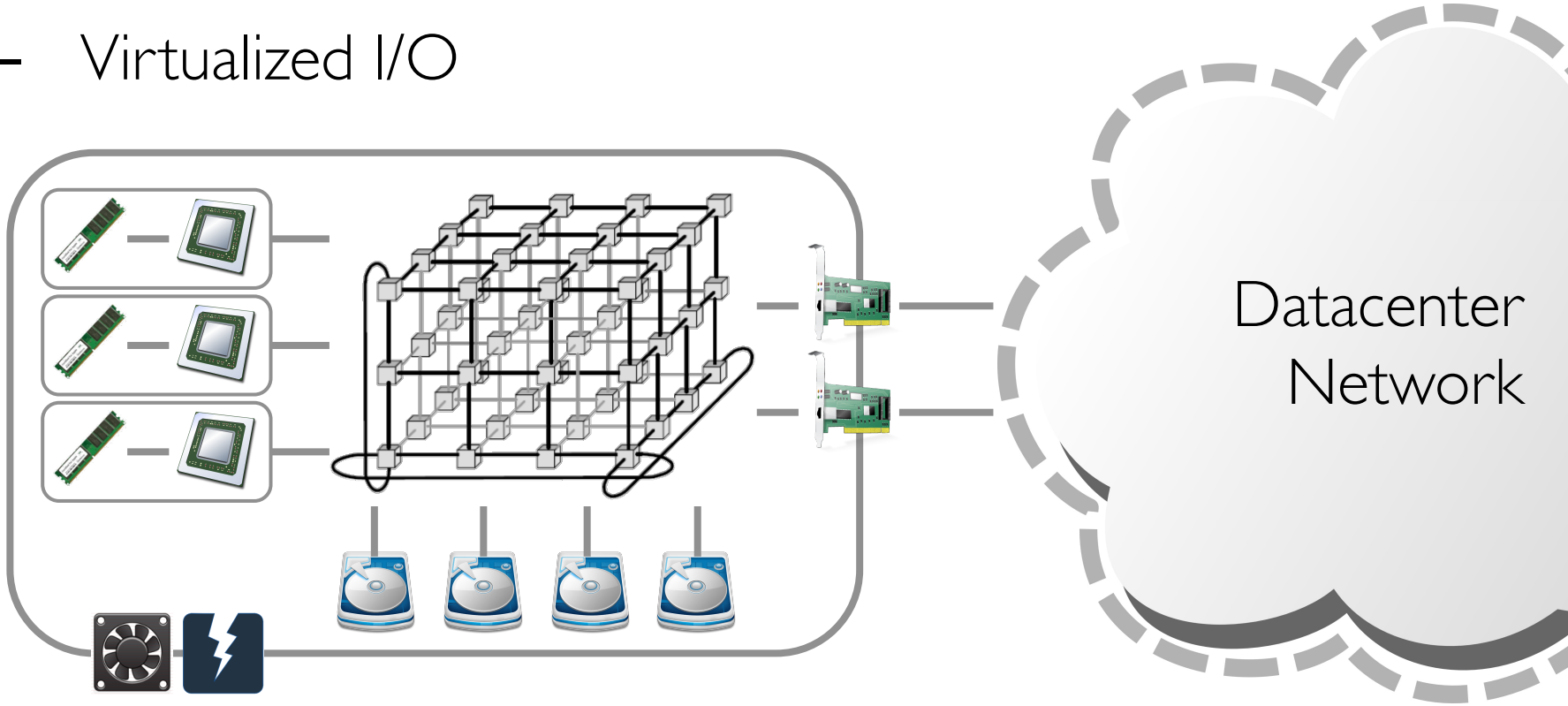


The Trends: Disaggregation

1. HP MoonShot
   - Shared cooling/casing/power/mgmt for server blades

The Trends: Disaggregation

1. HP MoonShot
2. AMD SeaMicro
   - Virtualized I/O

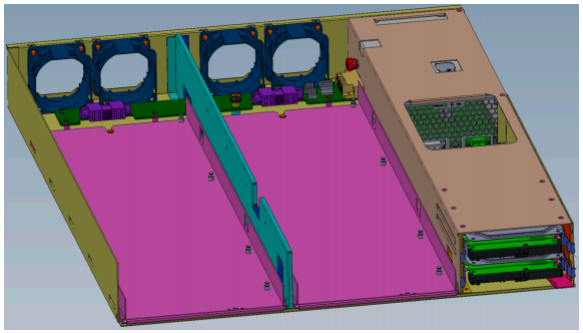


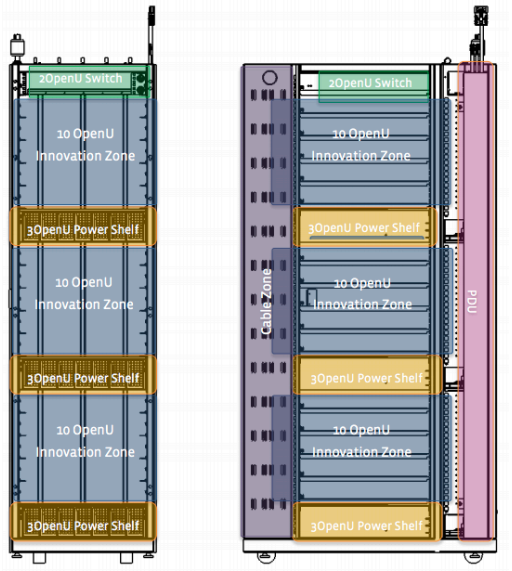
The Trends: Disaggregation

1. HP MoonShot
2. AMD SeaMicro
3. Intel Rack Scale Architecture

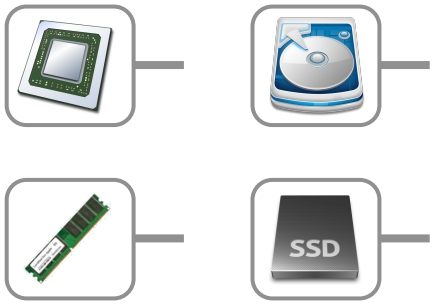
The Trends: Disaggregation

1. HP MoonShot
2. AMD SeaMicro
3. Intel Rack Scale Architecture
4. Open Compute Project

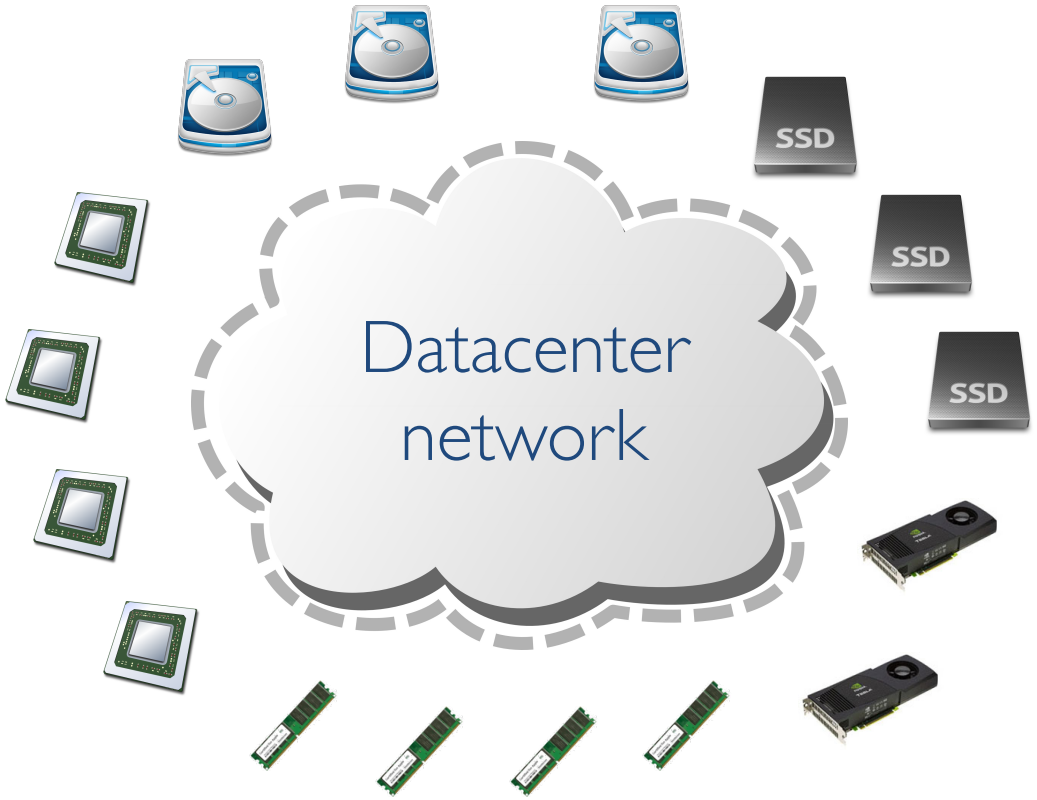




Disaggregated Datacenter



Resource as a standalone blade



Why will it happen?

: Extreme resource modularity

Benefits of Resource Modularity

1. Easier to build & evolve
   - Resources have different cycles/trends/constraints.
     - Tight integration in a server is a huge pain
     - E.g., "Memory capacity per core drops 30% for every 2 years" [Lim et al., ISCA '09]
   - Disaggregation enables independent evolution
   - The biggest driving force from vendor's viewpoint

Benefits of Resource Modularity

1. Easier to build & evolve
2. Fine-grained resource provisioning
   - Current practice: replace/buy an entire server, rack, or even datacenter.
   - e.g., "I want just more processors, not servers!"
     - Go buy some CPU blades at Best Buy® and plug them in.
   - e.g., "I want to try the new NVRAM technology!"
     - Again, go for it.

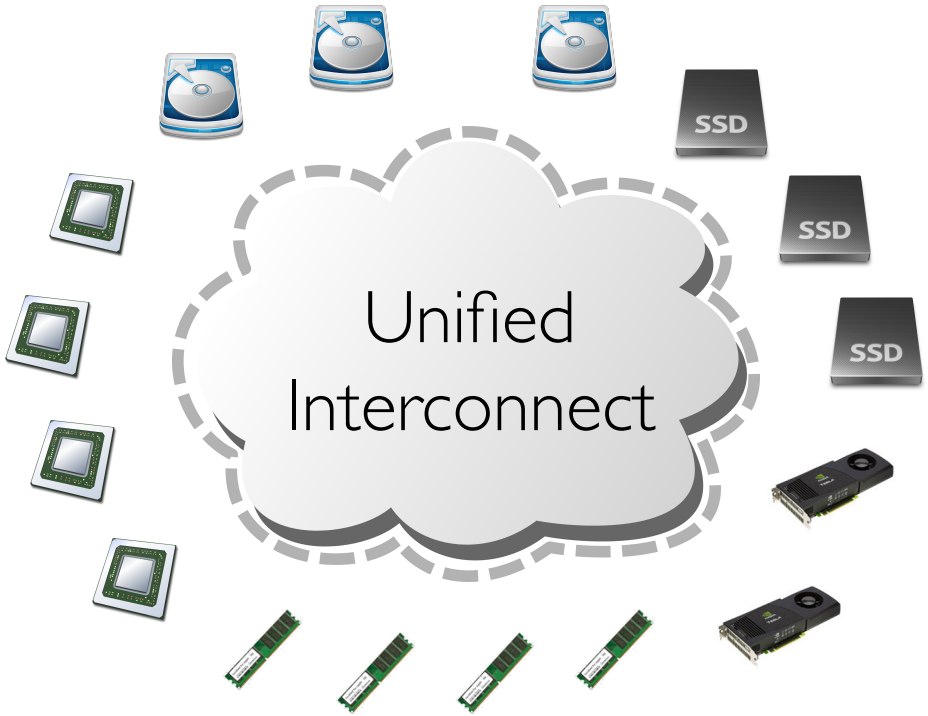
Benefits of Resource Modularity

1. Easier to build & evolve
2. Fine-grained resource provisioning
3. Operational efficiency
   - Datacenter as a single giant computer
   - Higher utilization with statistical multiplexing
   - (I will get back to this)

How will it happen?

: Incrementally and radically.

THIS IS NOT A CRAZY IDEA



- Do we need to change everything? NO.
  - HW change is minimal.
  - SW change is minimal, too.

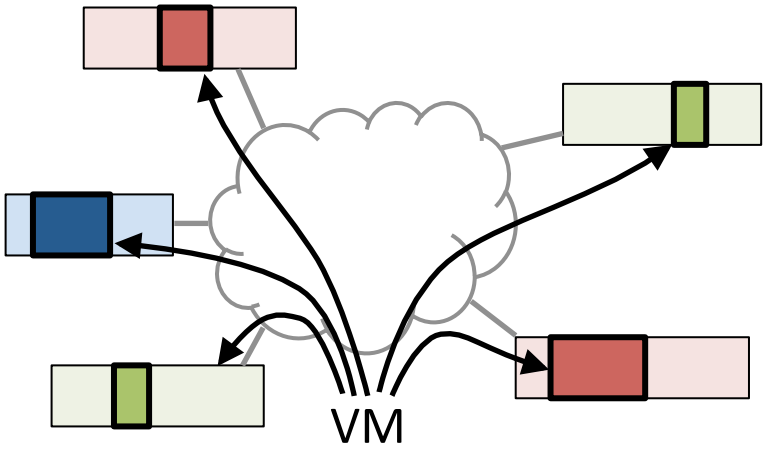
HW Requires Minimal Modification

Resource as a standalone blade

- The internals don't need to change.
- All we need is embedded network controller.
  - They already have: QPI, HT, PCIe, SATA, …
  - Can be very cheap
    - E.g., a whole graphics card w/ 128Gbps for only $50

How About SW?

- Existing SW infrastructure heavily relies on the concept of "server"
  - We don't want to rewrite it from scratch.
  - How to utilize the "giant computer"?



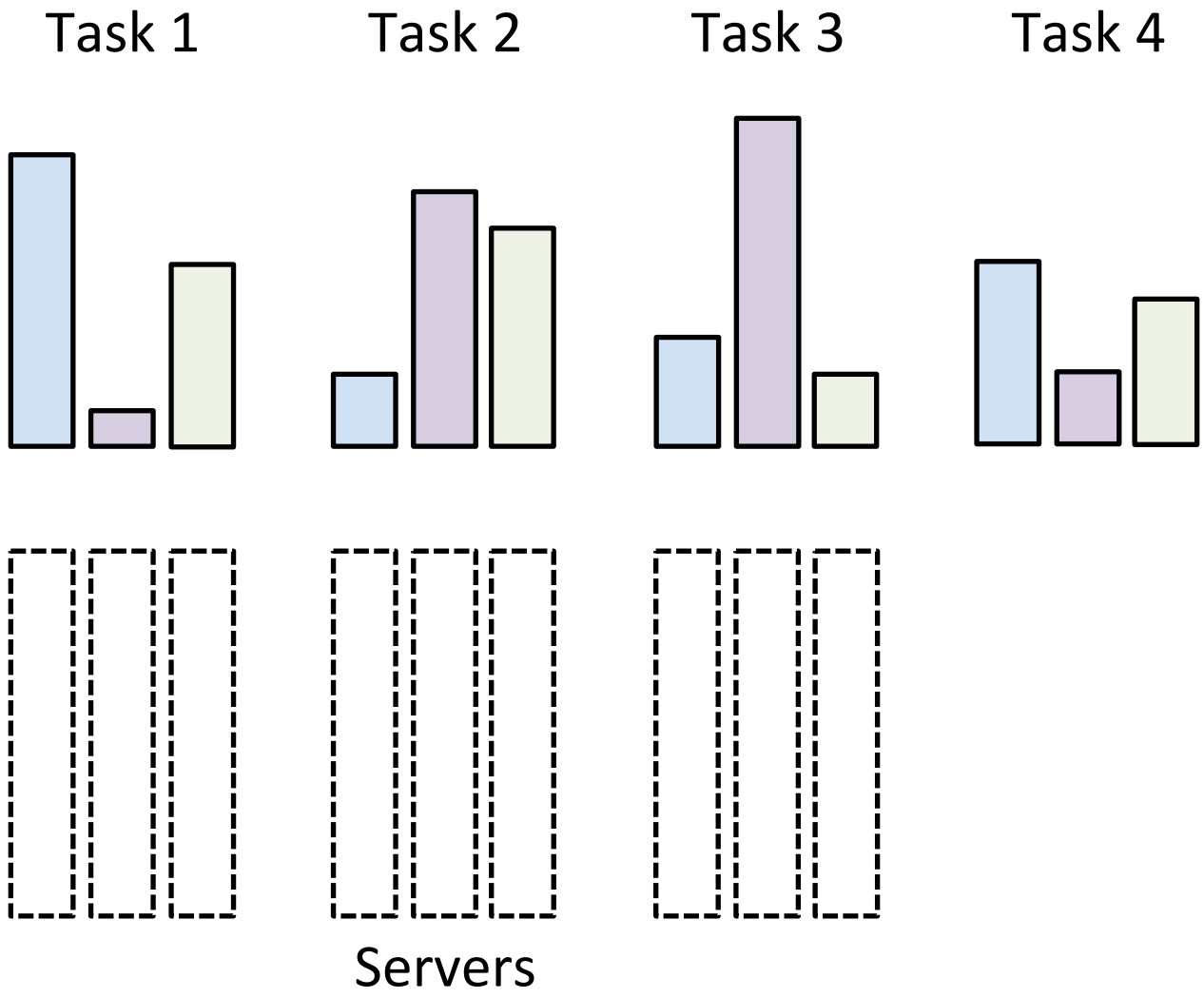
Disaggregated VM

No modification for App/OS

Minor changes in VMM.

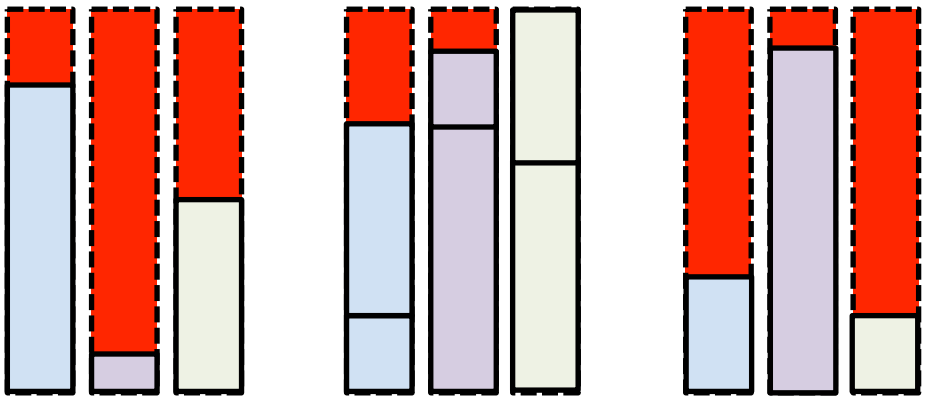
Much higher utilization!

Elastic VMs Achieve High Utilization!



Elastic VMs Achieve High Utilization!

40% of resources are wasted

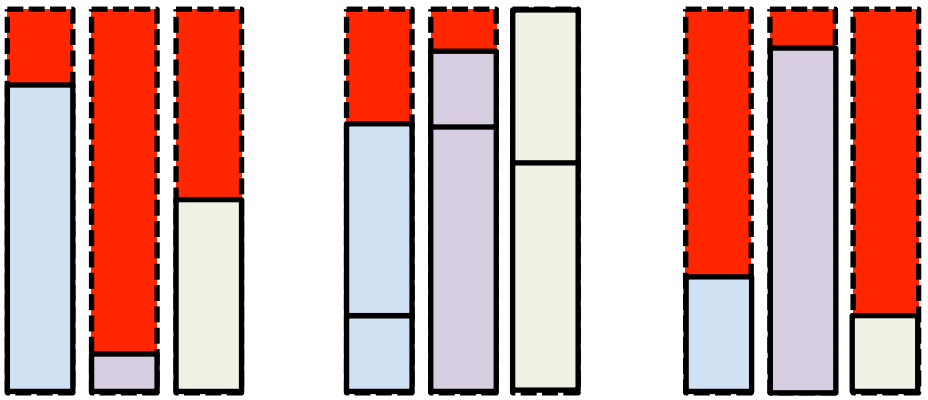


Servers

Elastic VMs Achieve High Utilization!

1. No "server boundary"
2. Statistical multiplexing at a larger scale
3. Higher utilization!

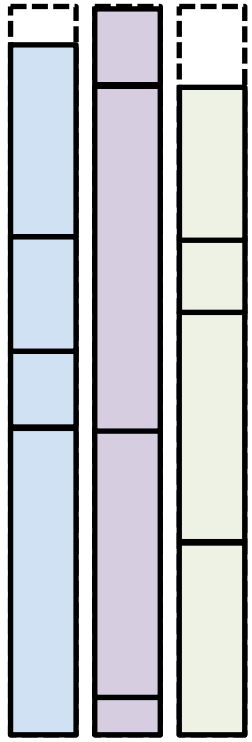
40% of resources are wasted



Servers

vs.

Disaggregated datacenter



(tasks vary greatly)

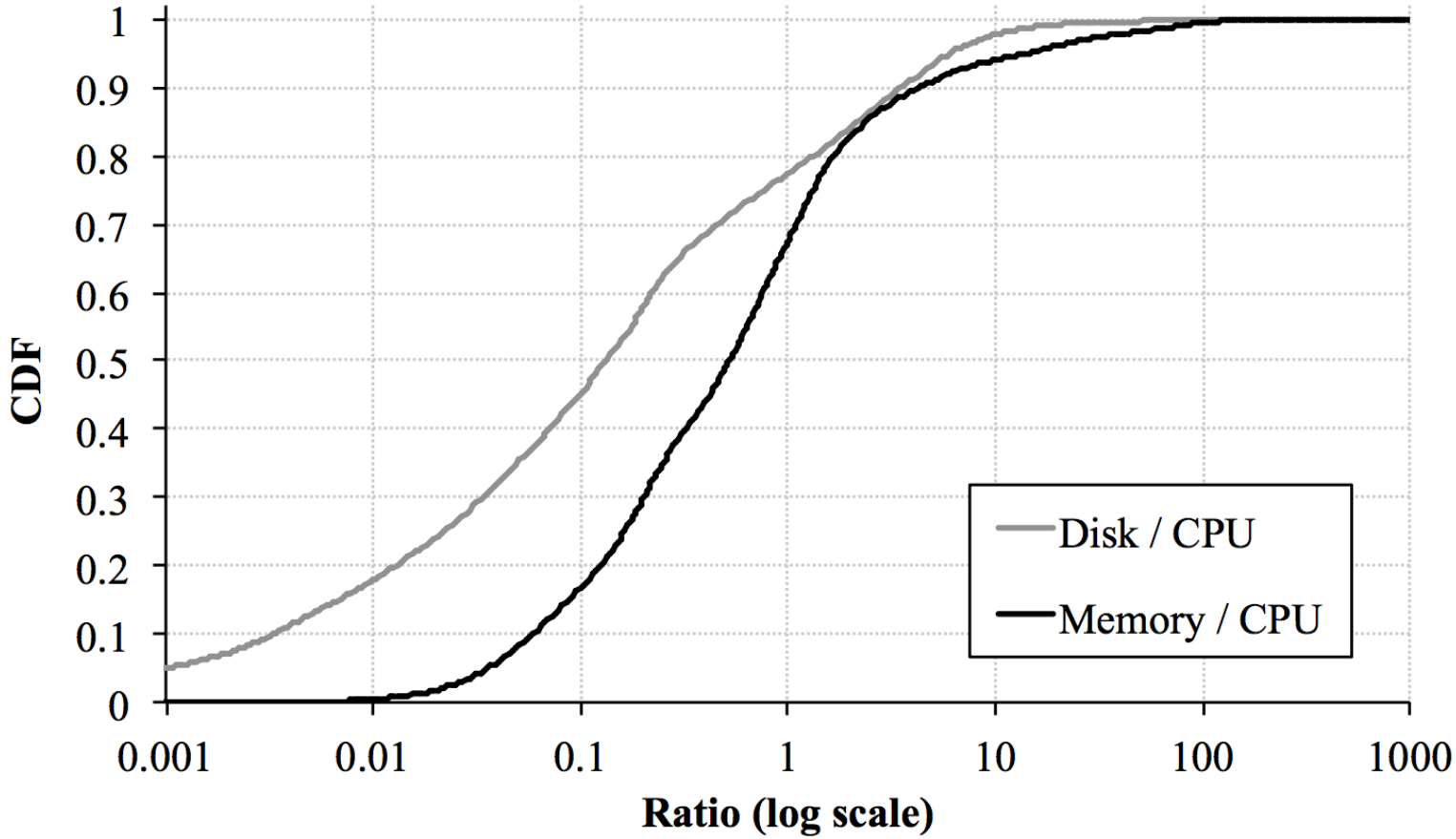


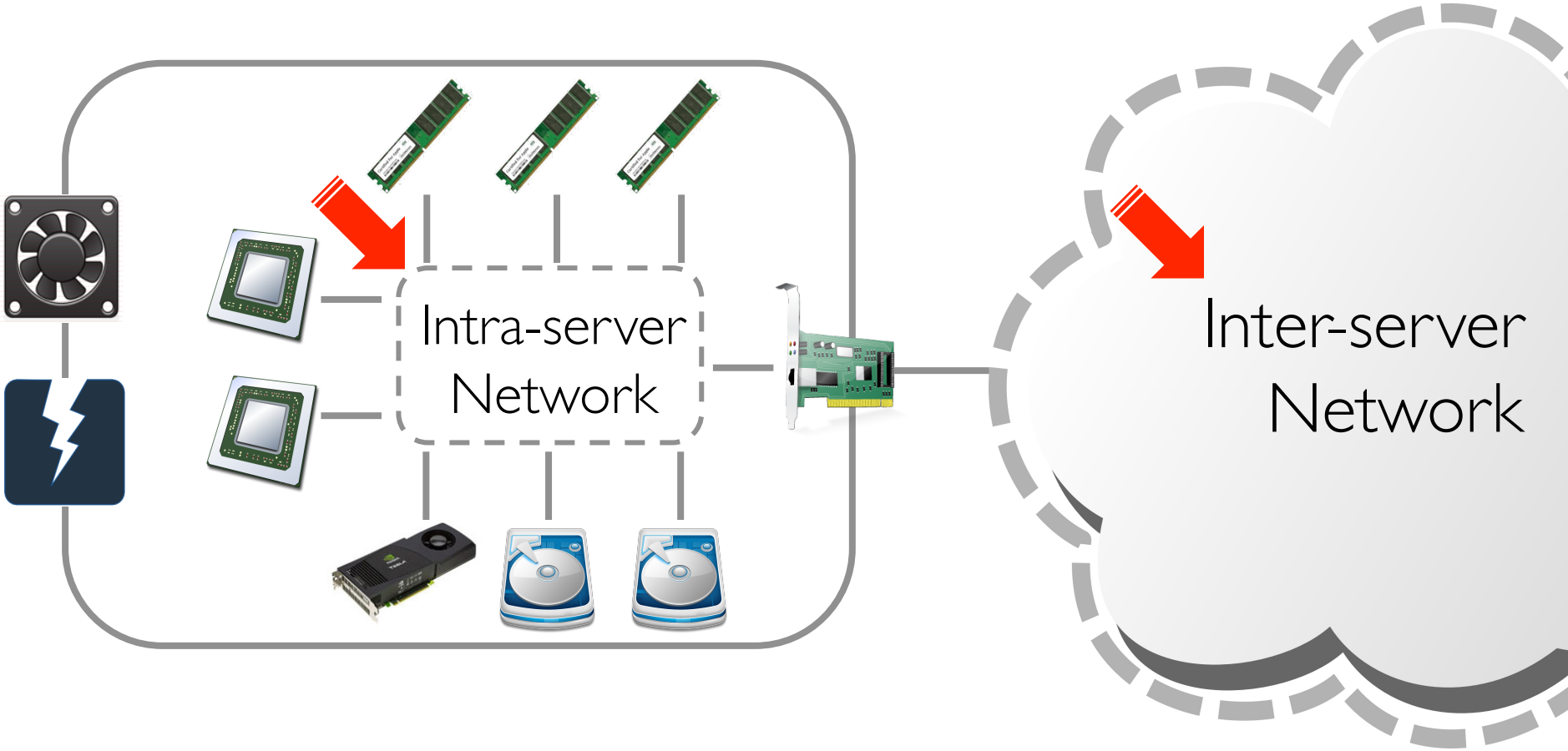
Figure 1: Distribution of disk/memory capacity demand to CPU usage ratio for tasks in Google's datacenter.

THIS IS NOT A CRAZY IDEA, part 2

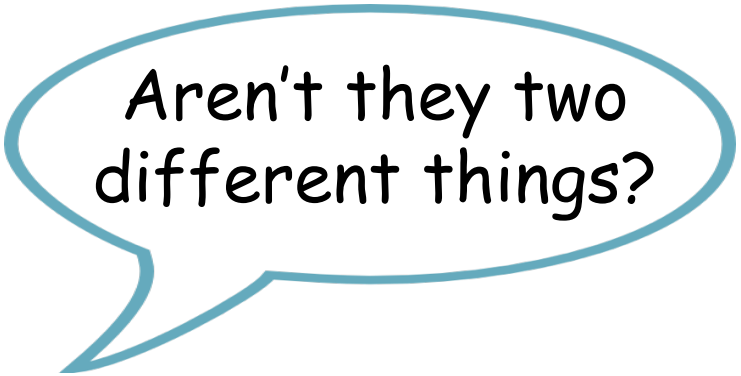
- We don't need to change everything.
  - HW change is minimal.
  - SW change is minimal, too.

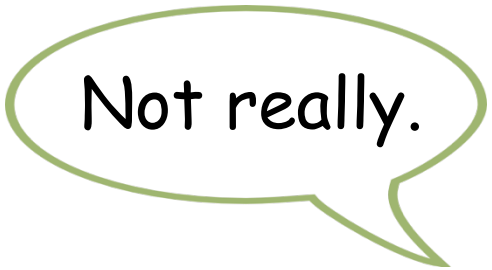
- A unified network is plausible
  - The intra-/inter-server networks can be unified.
  - Bandwidth/latency requirements are within reach.

Two Different(?) Types of Network



Intra- vs. Inter-Server Networking



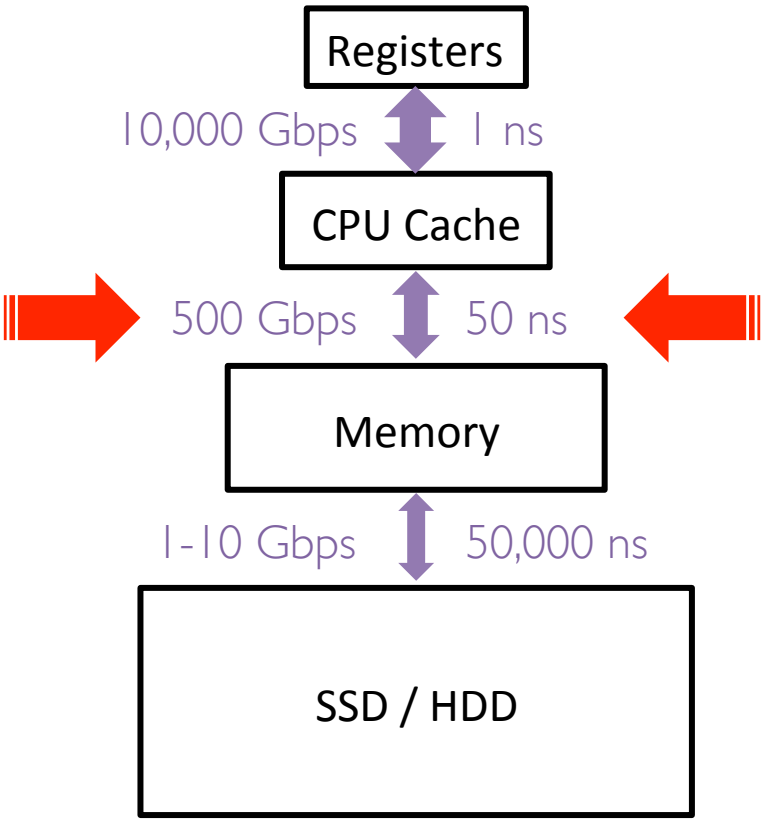


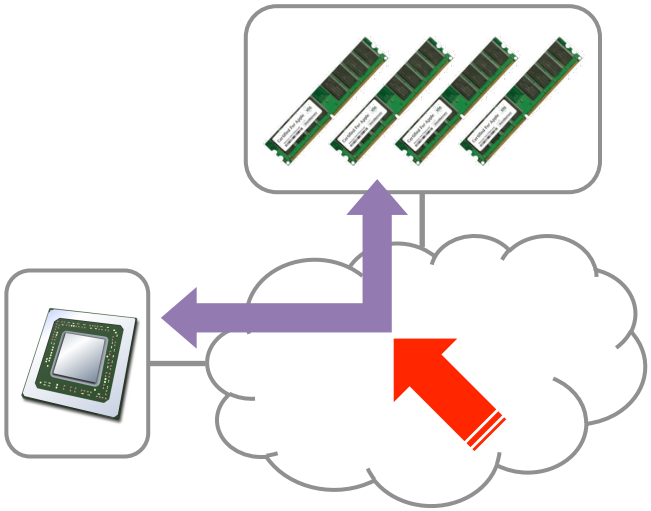
E.g., PCIe and 10GbE

- Serial
- Point-to-point
- Full duplex
- Packet-switched
- Variable packet size
- Supports both message and read/write semantics

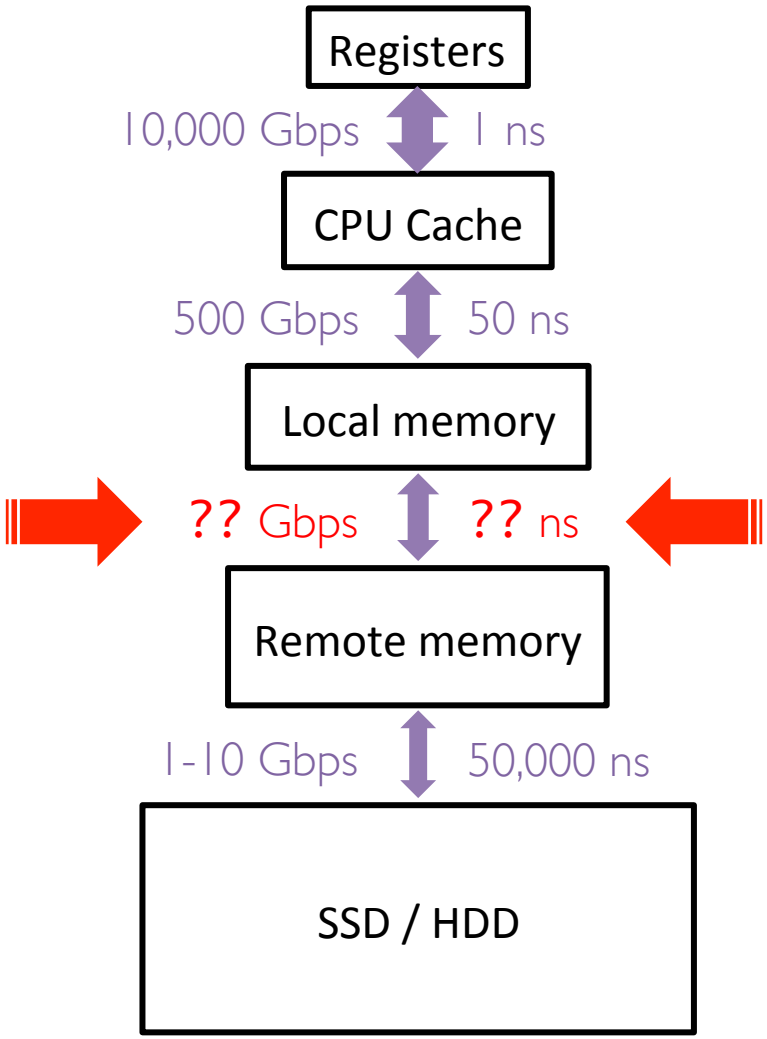
No fundamental™ difference!

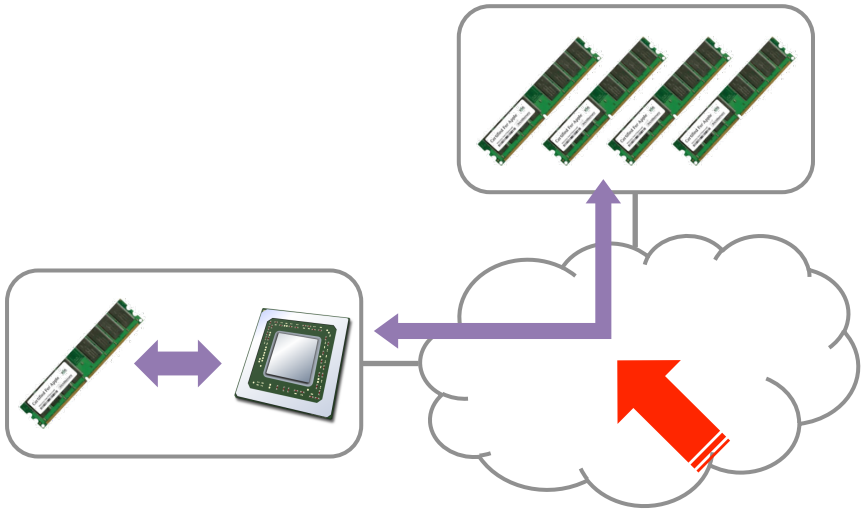
Making Memory Traffic Manageable





Making Memory Traffic Manageable

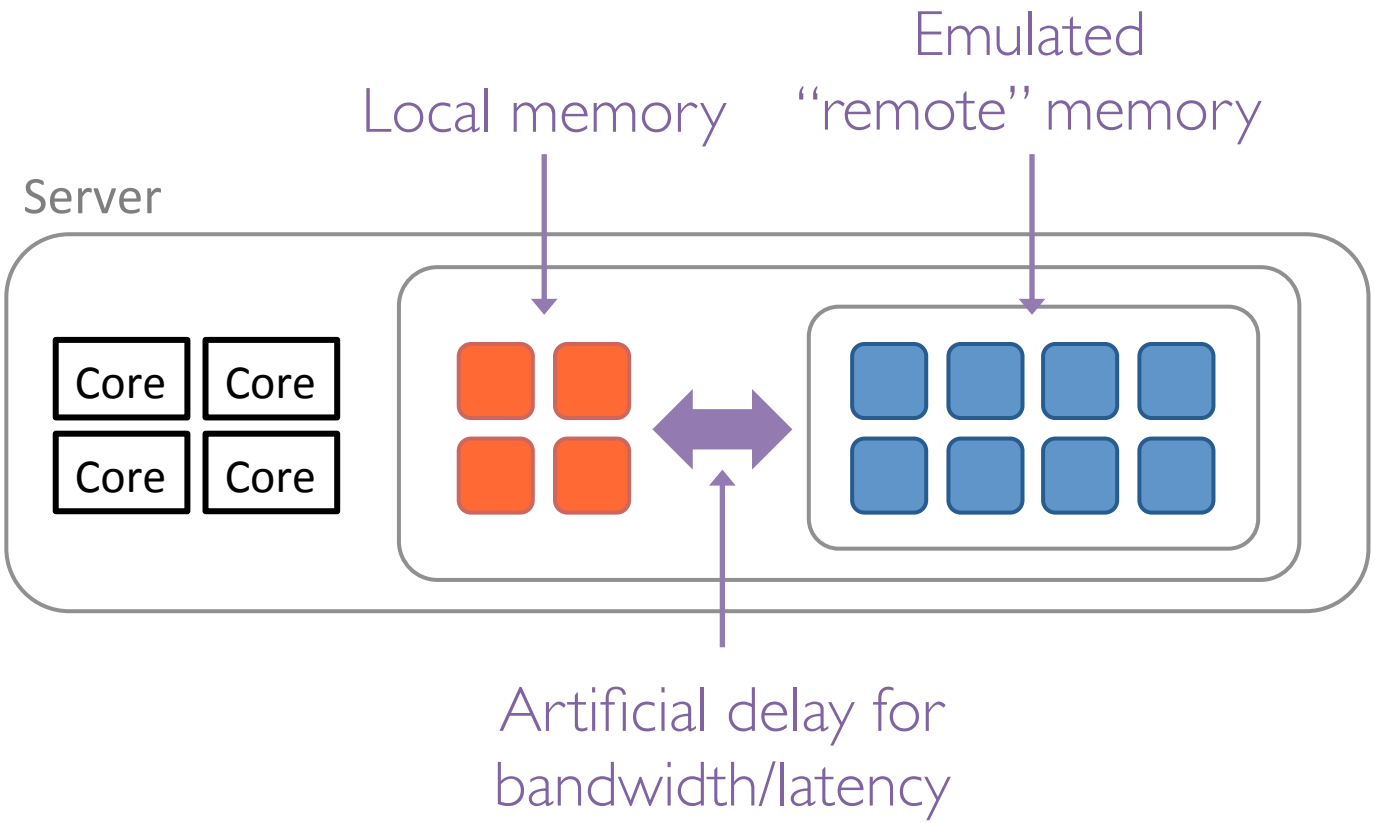




A small amount of local memory as a "cache"

Desirable Network Speed?

- A quick-and-dirty experiment



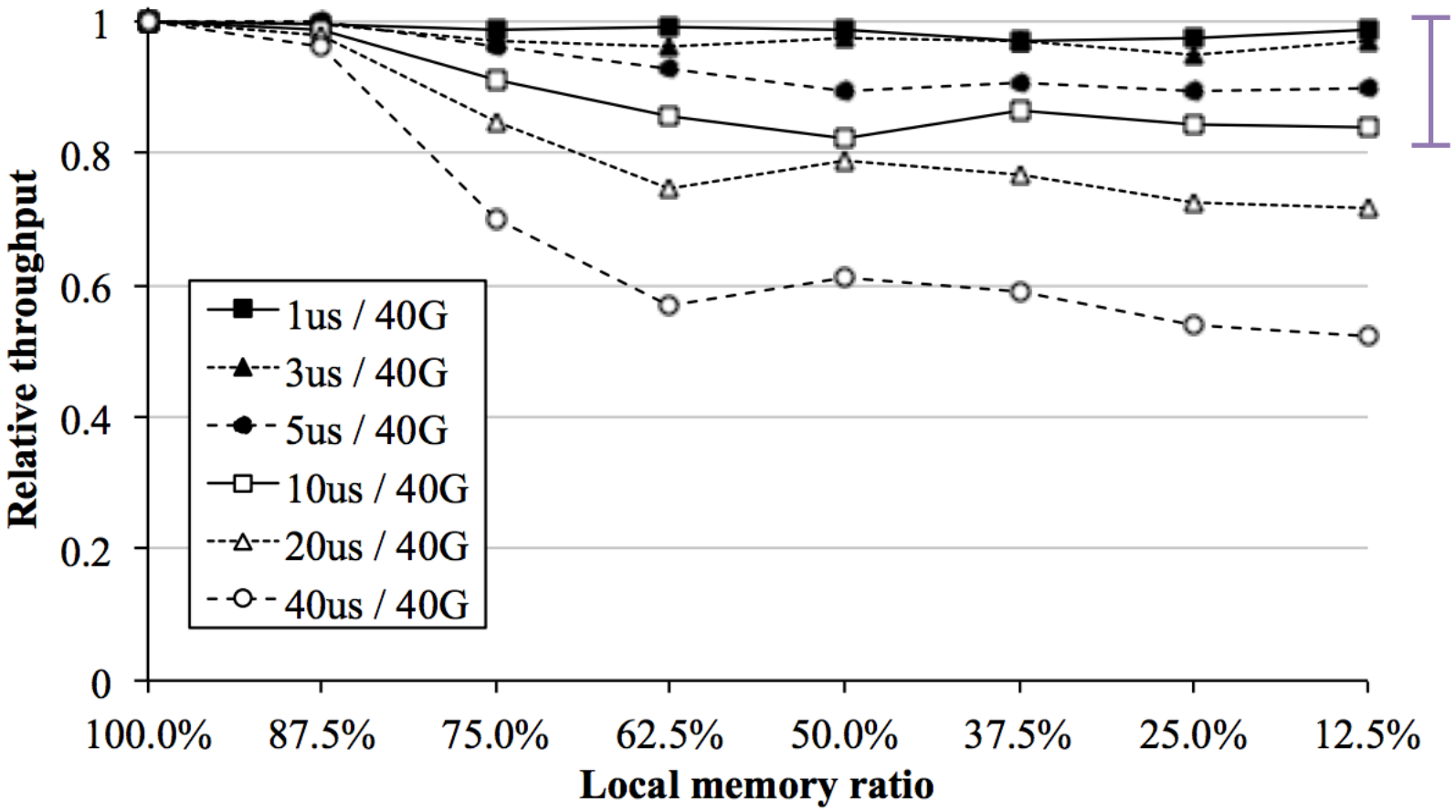
Desirable Network Speed?

- 4 CPU cores, 8GB working set size
- GraphLab, memcached, Pig

- Findings (read the paper)
  1. 10-40 Gbps is enough.
     1. Feasible even today!
     2. Average link utilization: < 1-5Gbps
  2. Latency matters.

WANTED: Low Latency

memcached with varying latency



< 10µs latency, < 20% overhead

Research Questions

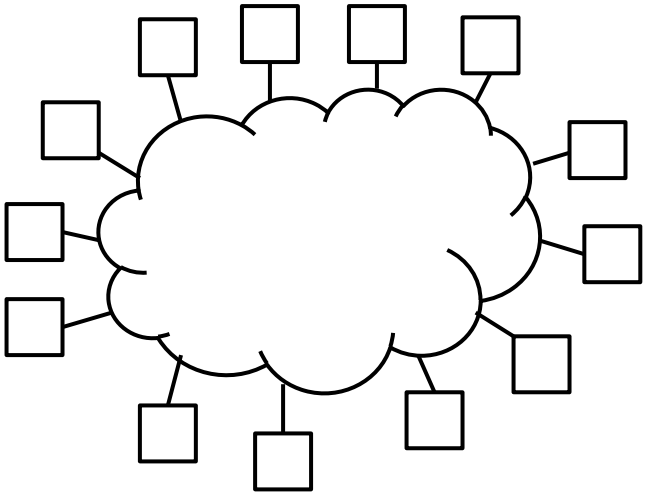
Questions

- Answered:
  - How fast should it be? >10-40Gbps, <1-10μs.

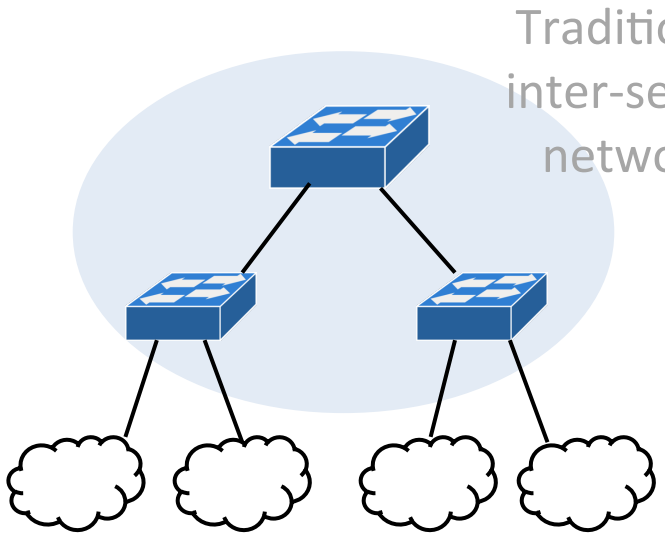
- Unanswered:
  - Scalability?
  - Reliable transfer?
  - QoS?
  - Packet? Circuit?
  - …

1. "Right" Scale of Disaggregation

- Disaggregation scale: where is the sweet spot?



datacenter-scale
(flat)



chassis/rack/pod-scale
(two-tier)

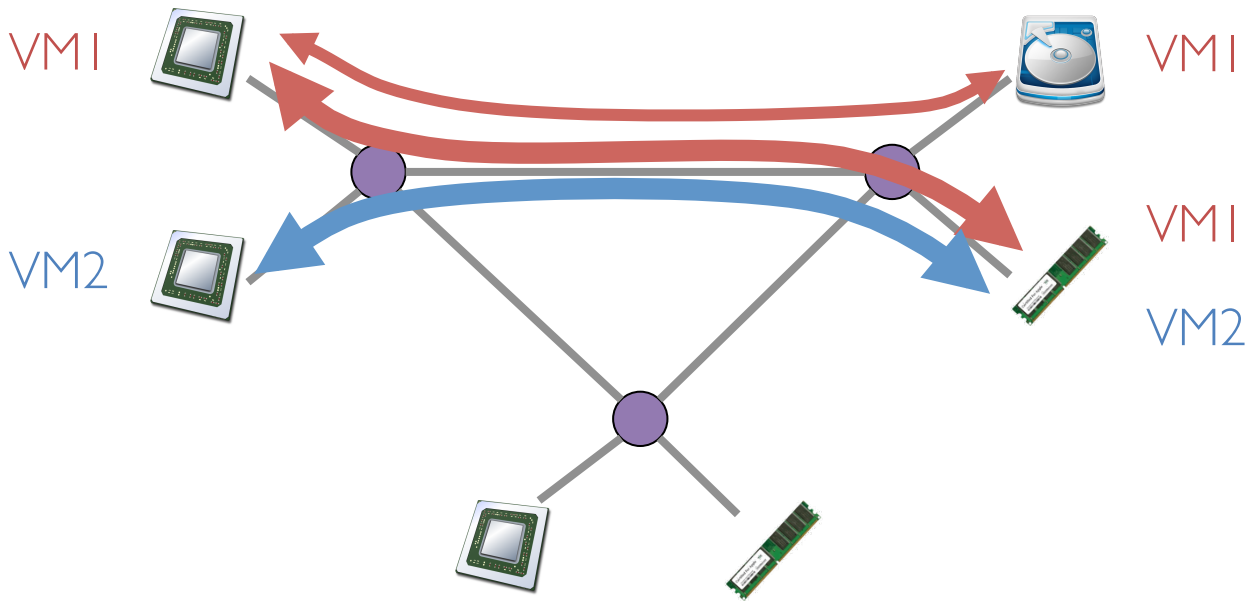
2. Realizing Low Latency

- It's time for low latency [Rumble et al., HotOS '11]
  - "5-10µs latency is possible in the short term"
  - "1µs round-trip times cannot be achieved off-processor"

- Congestion avoidance/control should be "close to the metal".
  - A lot of research efforts are ongoing.
  - Will they be still valid in disaggregated datacenters?

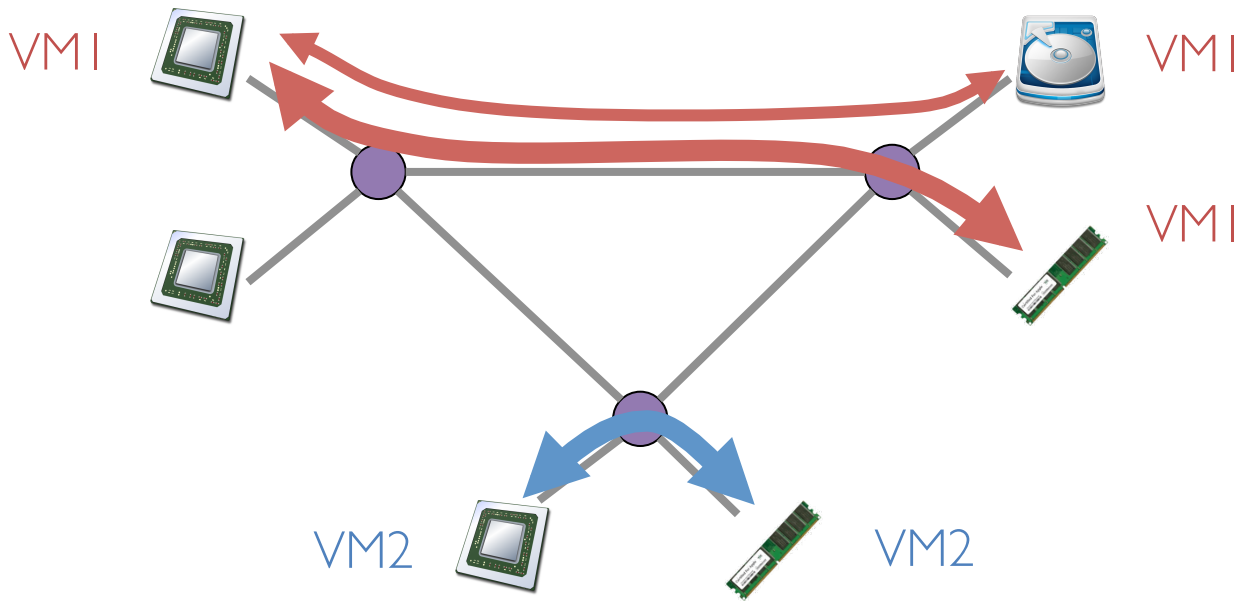
3. Unified Scheduler

- We will need a unified scheduler:
  - Job scheduler + network controller



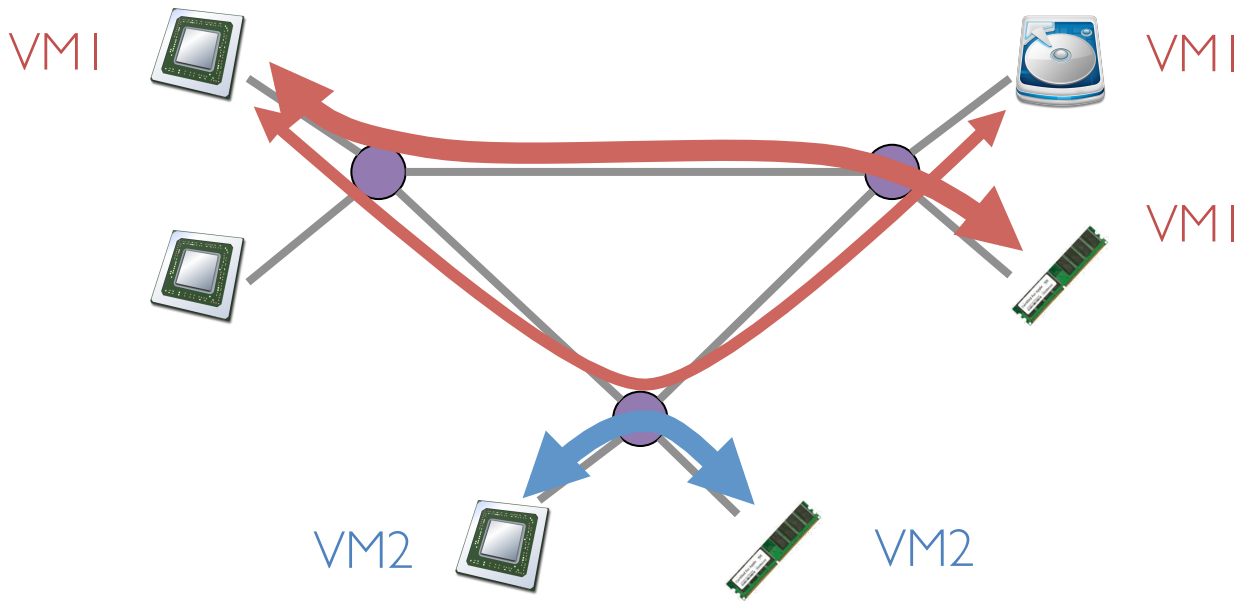
3. Unified Scheduler

- We will need a unified scheduler:
  - Job scheduler + network controller



3. Unified Scheduler

- We will need a unified scheduler:
  - Job scheduler + network controller



Closing Remarks

- Disaggregated datacenter will be "the next big thing"
  - Already happening. We need to catch up!

- We are working on a small-scale prototype.
  - Disaggregated resource blades on existing SW/HW
    - 40 CPUs
    - 6 remote memory blades
    - 8 GPU/NIC/storage blades
  - PCIe as the "unified interconnect"