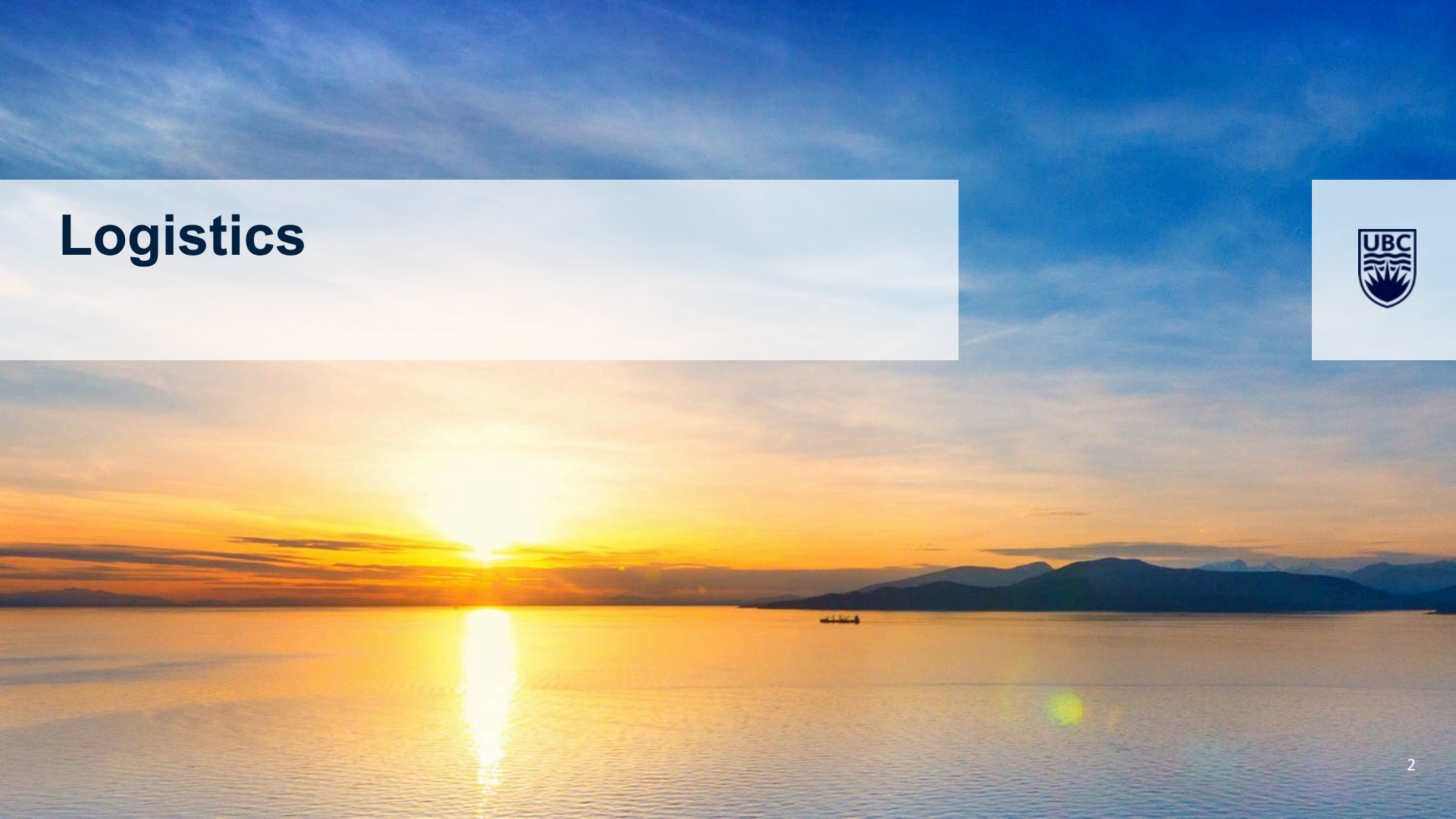
CPSC 416 Distributed Systems

Winter 2022 Term 2 (March 28, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer

Logistics



Deadlines

Project 4 Released. Late Due: April 13, 2023.

Project 5 Released Due: April 13, 2023. **No extensions**.

All project work is due April 13, 2023. Late projects are scaled to 75% of the on-time max.

Final Exam: April 20, 2023, DMP 310, 08:30-11:00. Format TBA.

Note: EC opportunity for Final Exam. Piazza 133, Canvas. Closes April

Deadlines

Alternate Path 1 & 2: Review in progress

- Piazza private threads need TLC
  - **Weekly updates due each Monday @ 23:59 PT**
- Final reports due no later than Thursday April 13, 2023 @ 23:59 PT
- Optional 10 min presentation April 13, 2023, up to 10 minutes.

Instructor Office Hours:

- Zoom Office Hours (Tuesday) @ 13:00-14:00
- Discord (Casual) Office Hours (Thursday) @ 14:00-15:00

TA Office Hours:

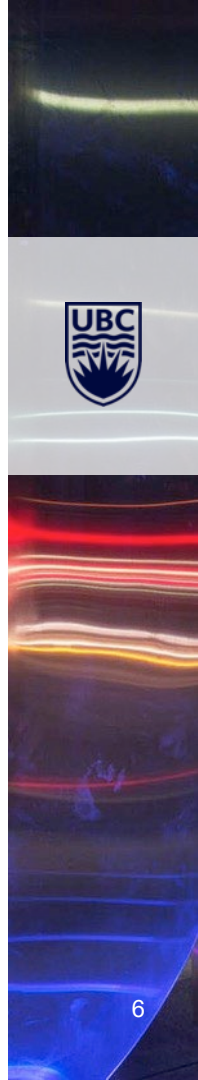
- Eric: Friday 9-11 am (in-person and Zoom)
- Japraj: Wednesday 3-5 pm (Zoom)
- Yennis: Thursday 2-4 (Zoom), Friday 2-4 (in-person)

Questions?

Questions about the class?

Questions about the previous lecture?

Funny stories to share?



Today’s Failure

Microsoft Azure

Date: September 4, 2018

Time: 08:42 UTC

Source: [Azure status history | Microsoft Azure (archive.org)]

Lightning storm caused “sags and swells” to voltage feeds in Azure’s South Central US Region (Texas)

By design, this caused the air conditioning to shut down:

At 08:42 UTC, lightning caused a large swell which was immediately followed by a large sag in one of the region’s datacenters, which fell below the required voltage specification for the chiller plant. By design this sag triggered the chillers to power down and lock out, to protect this equipment.

Microsoft Azure (September 4, 2018)

Backup cooling was in *manual* mode:

In the impacted datacenter, the redundant cooling system had been switched to manual mode for invoking the failover of cooling. It was set to a manual failover following the installation of new equipment in the facility where all testing had not been completed.

The warnings were not noticed, bad things happened.
Onsite engineers in the impacted datacenter received multiple alarms so were investigating several situations reported in the facility. Cooling alerts which should have been manually actioned were not. Consequently, temperatures began to rise, eventually reaching temperatures that triggered infrastructure devices and servers to shutdown.

Microsoft Azure September 4, 2018

Things got bad faster than expected:

This shutdown mechanism is intended to protect infrastructure from overheating but, in this instance, temperatures increased so quickly in parts of the datacenter that a number of storage servers were damaged, as well as a small number of network devices.

Recovery is hard:

The decision was made to work towards recovery of data and not fail over to another datacenter, since a failover would have resulted in limited data loss due to the asynchronous nature of geo replication. Recovering the storage services involved replacing failed infrastructure components, migrating customer data from damaged servers to healthy servers, and validating the integrity of the recovered data. This process took time due to the number of servers that required manual intervention, and the need to work carefully to maintain customer data integrity above all else.

Microsoft Azure September 4, 2018

Takeaways:

- Unexpected things can happen at the worst of times
- Failures compound
- Recovery is slow and painful
- Core service failures lead to higher level service failures
  - VS Code/Visual Studio Marketplace
  - Azure Active Directory
  - Azure Service Manager
  - Etc.

Lesson Goals

Data Center Services

Trends and Services

High-speed RDMA and programmable networks

Resource heterogeneity

Resource disaggregation

Resource management and orchestration

Trends

Moore's Law:

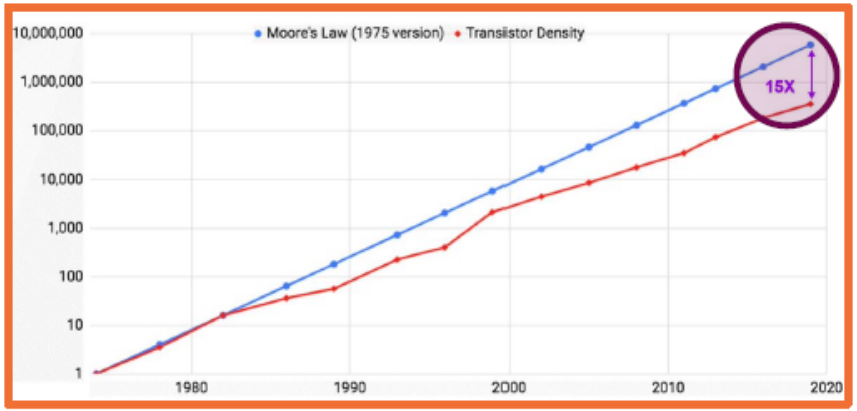
- Economy of scale
- Performance and scale with commodity components

Specialization/heterogeneity

- GPUs, TPUs, etc.
- New memory
- New Storage classes

Disaggregation

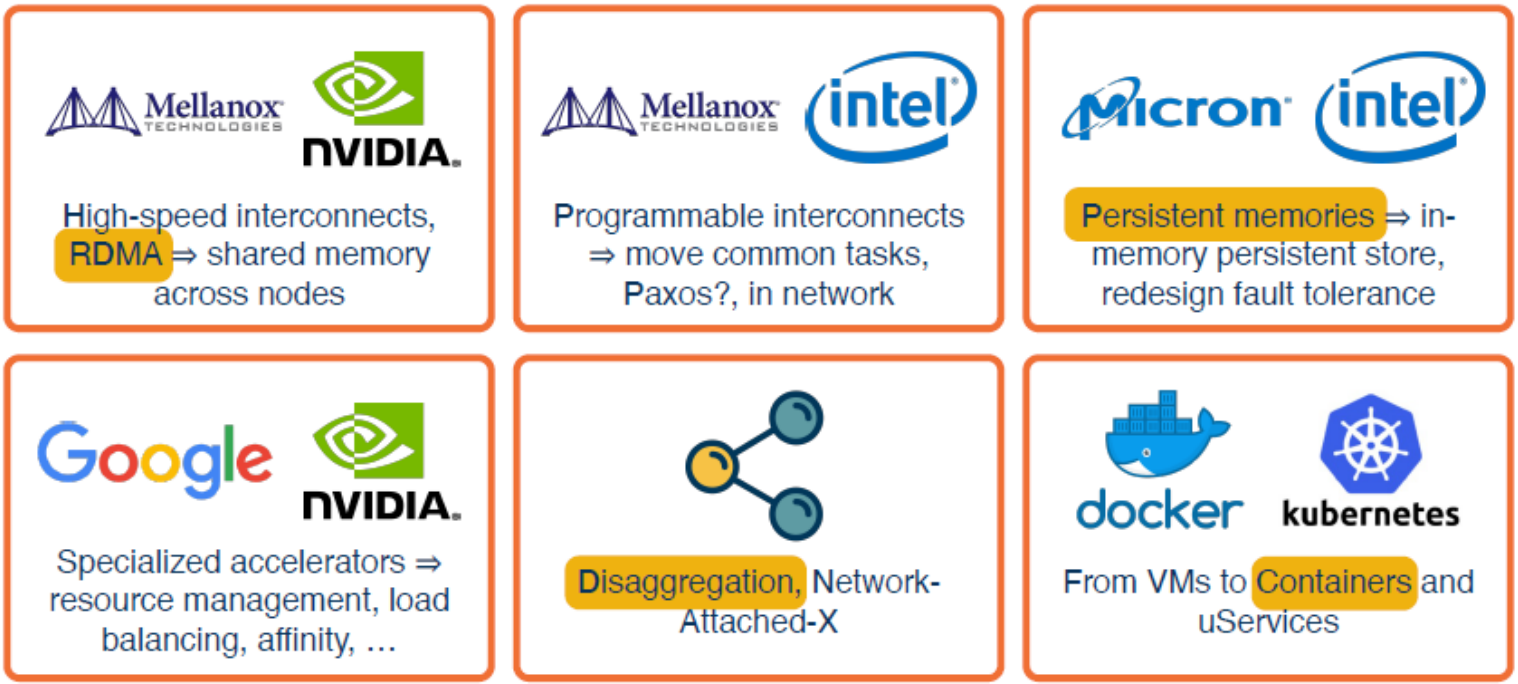
- Independently scaled tiers of different resources



Moore, Gordon E. "No exponential is forever: but 'Forever' can be delayed!" Solid-State Circuits Conference, 200.

New Technologies

Limitation of (x86+DRAM+Ethernet) + scale and requirements of emerging workloads:



Remote Direct Memory Access (RDMA)

Remote DMA

Bypass CPU

- Data access/communications via interconnect support (“network”)

Benefits (versus commodity Ethernet):

- Higher Bandwidth
- Lower latency

Tradeoffs:

- Costs

RDMA

Original idea: 1990s research

Virtual Interconnect Architecture (VIA): 2000s

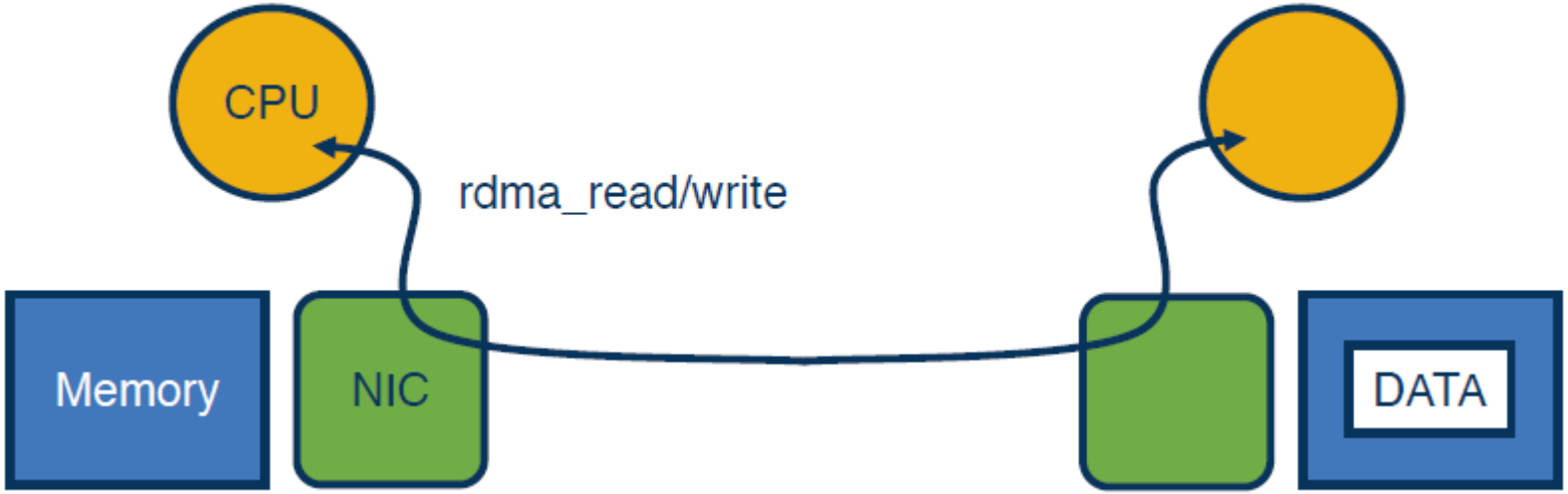
Mellanox: Infiniband

- Today: approx. 50% of top 500 machines, many data centers

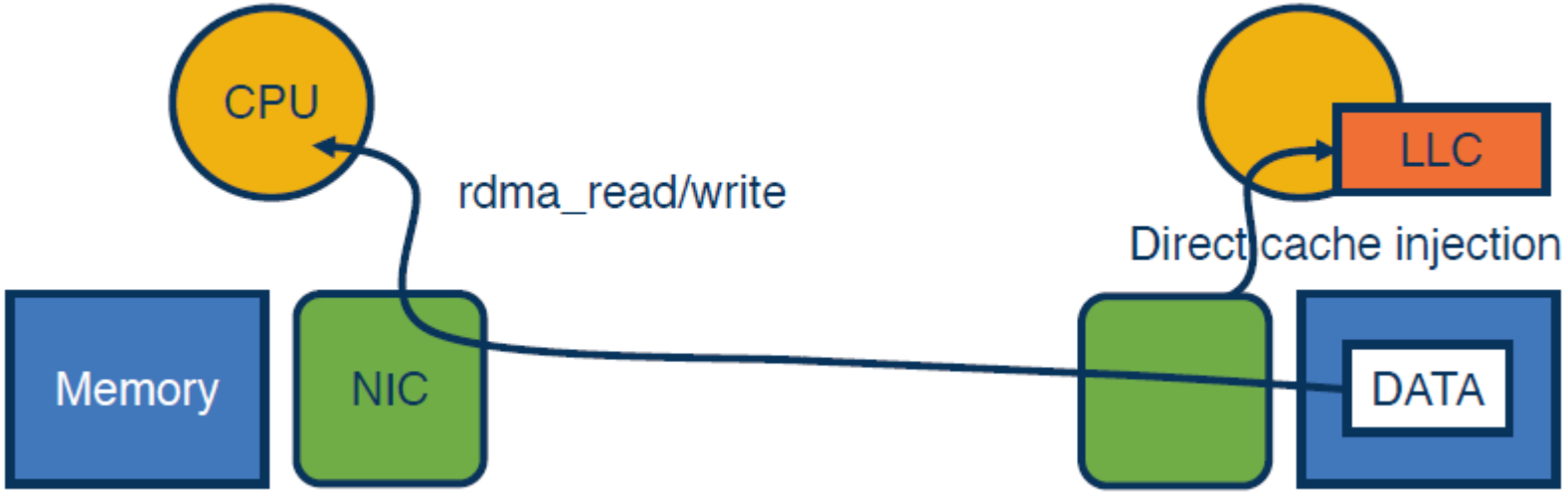
Remote DMA over Converged Ethernet (RoCE)

Internet Wide Area RDMA Protocol (iWARP)

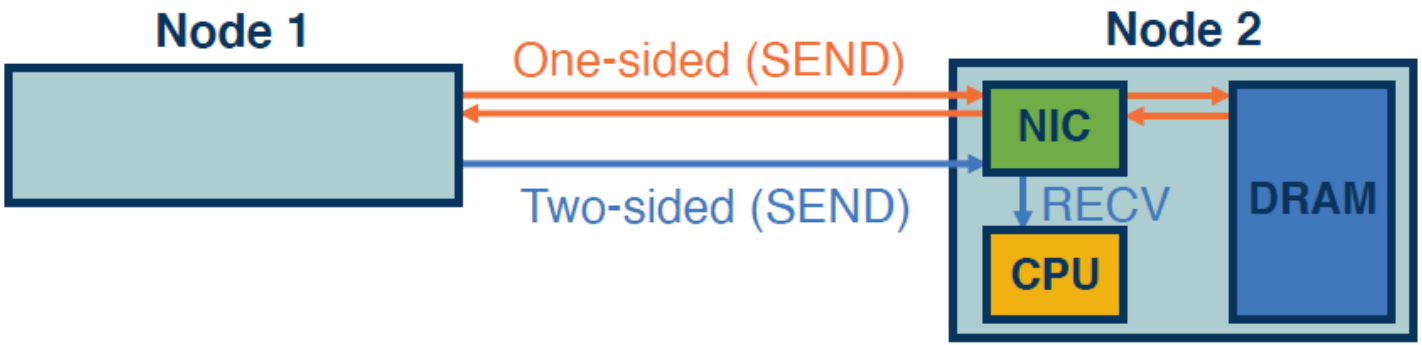
Two-sided RDMA



One-sided RDMA



RDMA Specialized Remote Procedure Call (RPC)



One-sided:

- Low CPU utilization, multiple RTTs
- Redesign to match new API

Two-sided:

- Single RTT, simple integration in existing stack

Figure adapted from: https://www.usenix.org/sites/default/files/conference/protected-files/osdi16_slides_kalia.pdf

RPC with RDMA Options

Leverage RDMA features:

- Connection vs. connection-less protocol
- Shared Receive Queues
- Datagram communication for small RPCs
- …

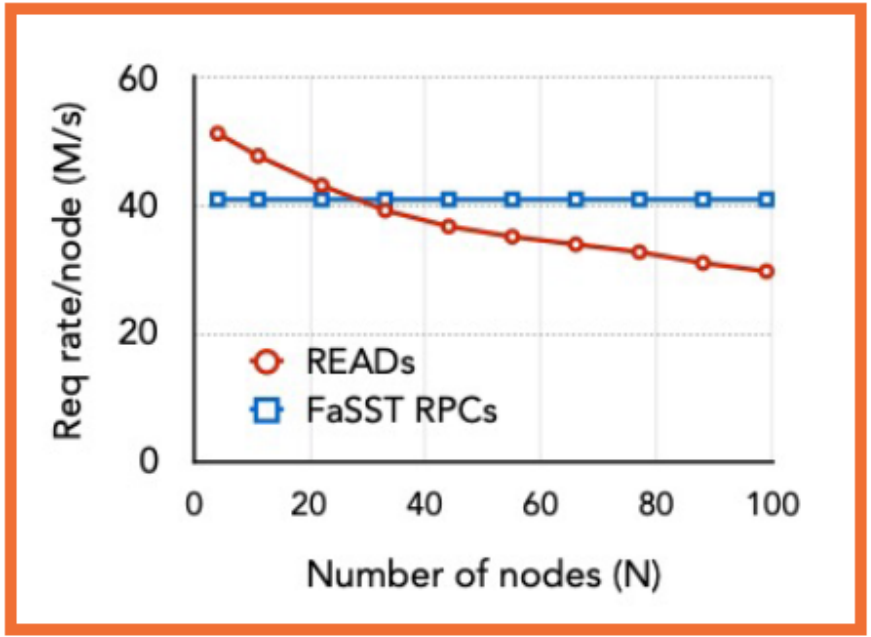
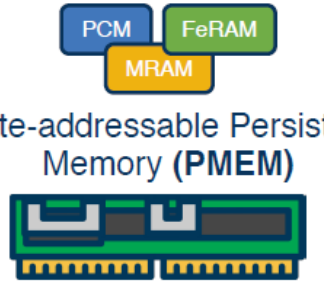
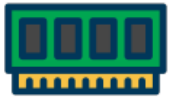


Figure from: FaSST Paper, OSDI'16

Persistent Memory

	PCIe SSD	Byte-addressable Persistent Memory (PMEM)	Main Memory (DRAM)
			
Interface	block I/O	**byte**	byte
Durability	yes	**yes**	no
Latency	30us	~100ns -- 300ns	60ns
Bandwidth	-	$\frac{1}{8}x$ *to* $\frac{1}{4}x$	$1x$
Capacity	Terabytes	Terabytes	Gigabytes

RPC + Persistent Memory

Persistent data operations require **flush** to persistent memory

Must complete **before client is Acknowledged**

Removes advantage of RDMA over send/receive RPC

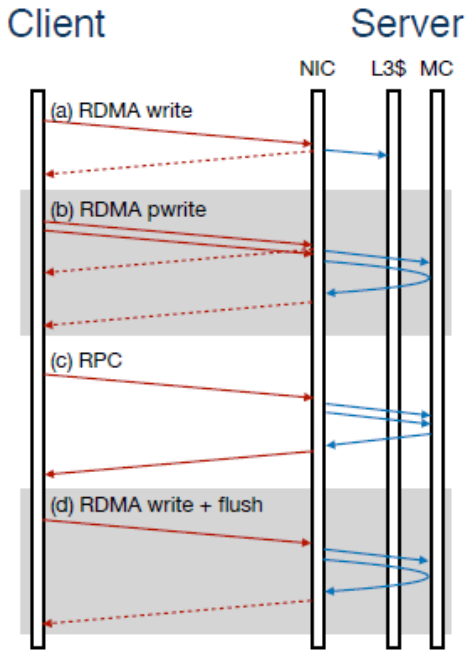


Figure 2 from SOCC'20 paper: *http://anujkalia.com/doc/socc20/kalia.pdf*

Network and PCIe operations involved in writing to remote NVMM with different methods. Red arrows from the client to the server's NIC are network packets. The dotted arrows are NIC-generated RDMA acknowledgments. Straight blue arrows between the server's NIC and its cache (L3) or memory controller (MC) are PCIe DMA or MMIO writes; the curved ones are DMA reads. The server's CPU (not shown) is in-volved in persisting RPC requests.

Disaggregation

Server configurations

- Different memory components
- Different compute components
- Storage
- Etc.

Choose specific amounts based on workload

- Workloads evolve

Monolithic server configurations

- Inflexible, cannot elastically scale
- Imbalances lead to inefficiency

Monolithic servers with fixed configuration of CPU, memory, storage, devices, PCIe slots, …

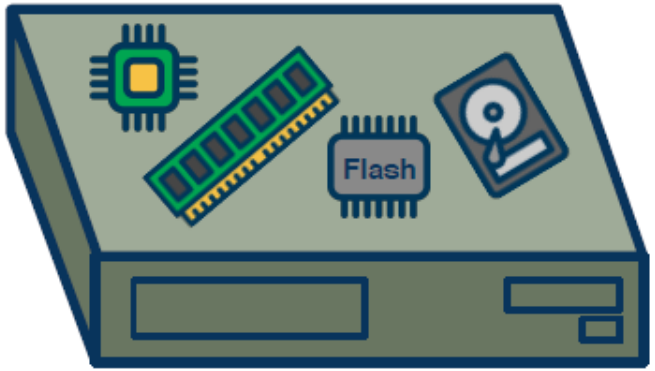


Figure adapted from the LegoOS presentation: https://www.usenix.org/sites/default/files/conference/protected-files/osdi18_slides_shan.pdf

Disaggregation

Resource Disaggregation

- Pools of different resource types
- Network attached
- Independently scaled

Motivation

- Fast Networks
- Integrate compute with devices
  - Smart NICs
  - In-memory compute

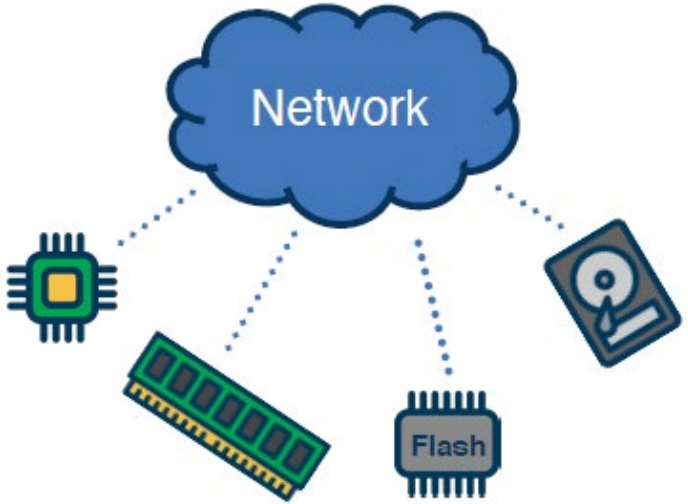
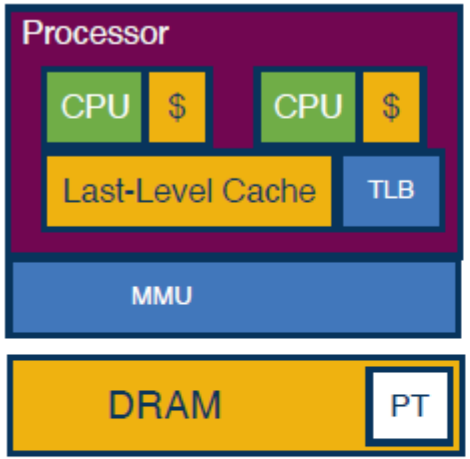


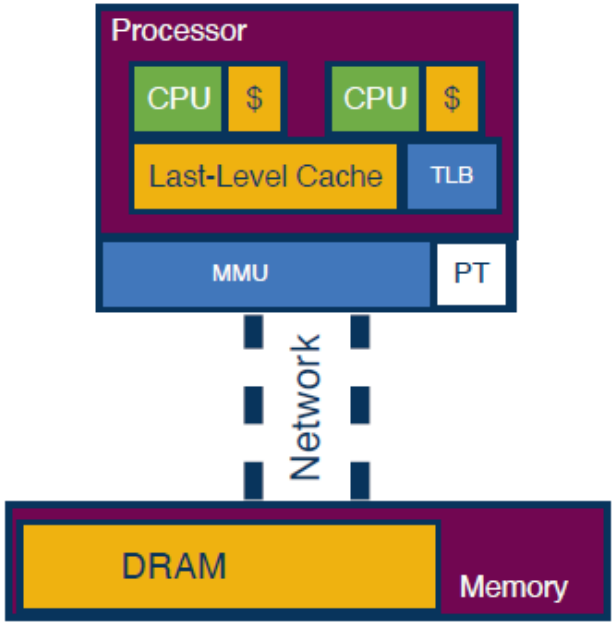
Figure adapted from the LegoOS presentation: https://www.usenix.org/sites/default/files/conference/protected-files/osdi18_slides_shan.pdf

Separating Processor and Memory



Figures adapted from the LegoOS Presentation at OSDI'20: https://github.com/WukLab/LegoOS

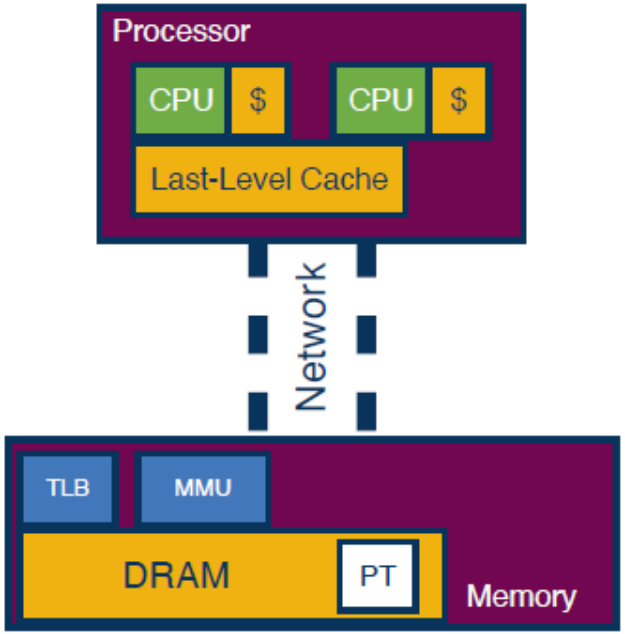
Separate Processor and Memory



Disaggregating DRAM

Figures adapted from the LegoOS Presentation at OSDI'20: https://github.com/WukLab/LegoOS

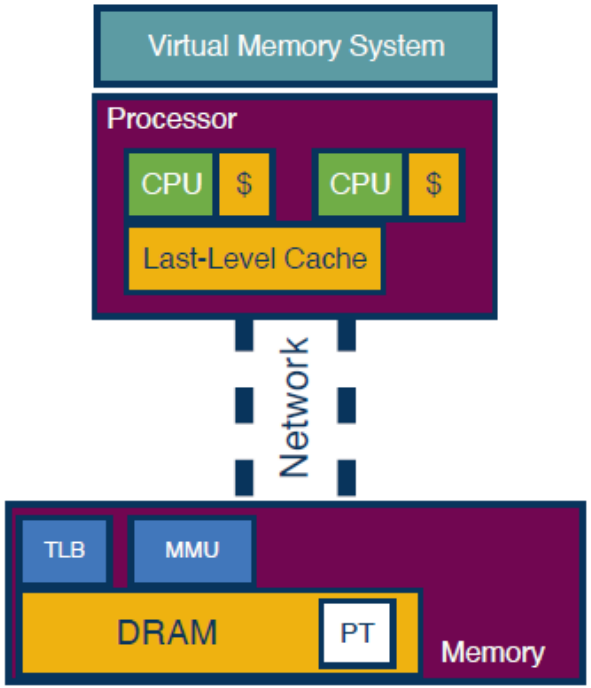
Separate Processor and Memory



Separate and move
hardware units
to memory component

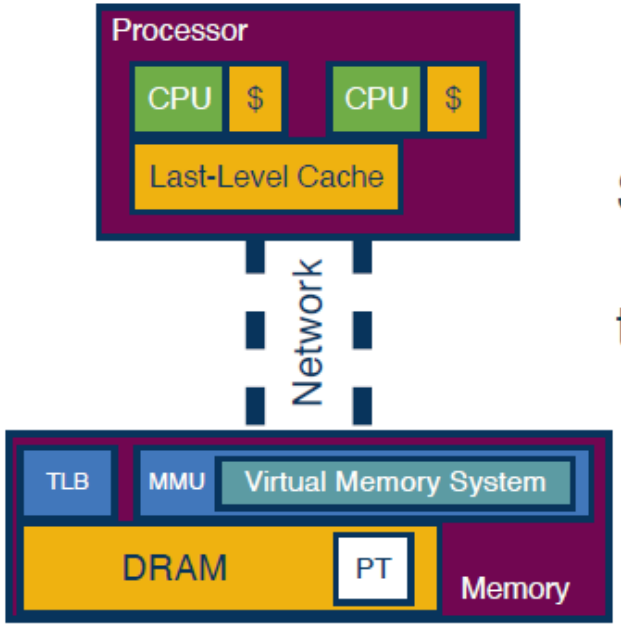
Figures adapted from the LegoOS Presentation at OSDI'20: https://github.com/WukLab/LegoOS

Separate Processor and Memory



Figures adapted from the LegoOS Presentation at OSDI'20: https://github.com/WukLab/LegoOS

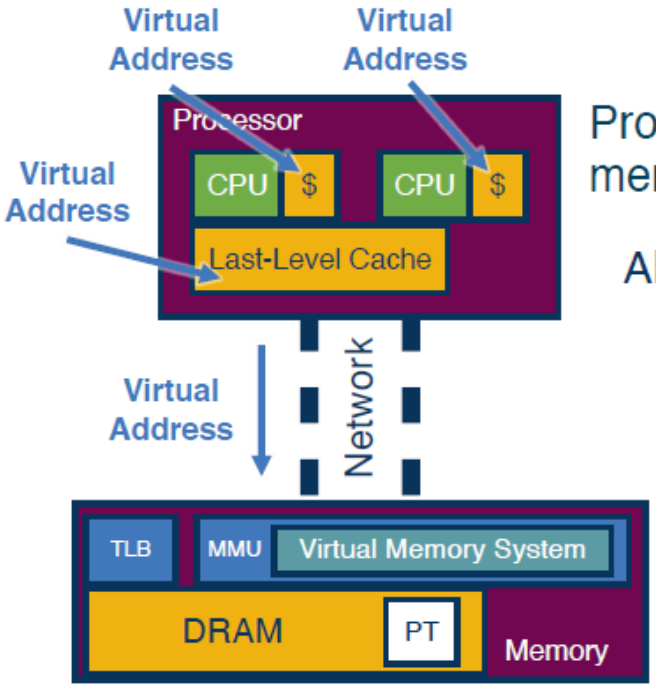
Separate Processor and Memory



Separate and move
virtual memory system
to memory component

Figures adapted from the LegoOS Presentation at OSDI'20: https://github.com/WukLab/LegoOS

Separate Processor and Memory



Processor components only see virtual memory addresses

All levels of cache are ***virtual cache***

Memory components manage virtual and physical memory

Figures adapted from the LegoOS Presentation at OSDI'20: https://github.com/WukLab/LegoOS

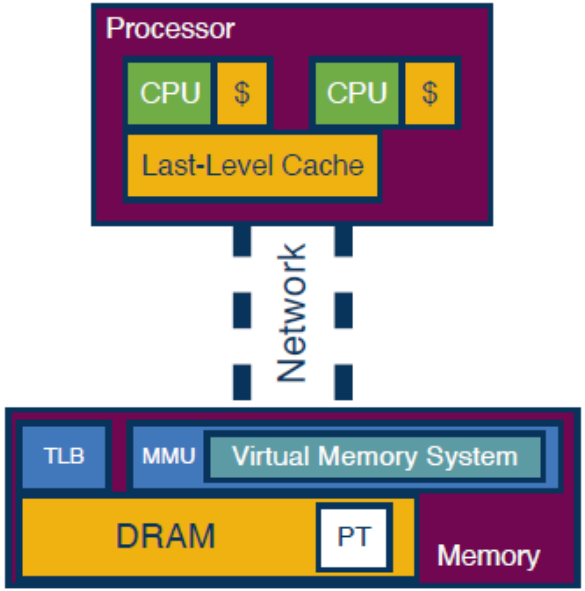
Disaggregation Challenges

Network is slower than local memory bus

- Bandwidth: 25-50% capacity
- Improving quickly (800Gb/s Ethernet now exists)

- Latency: 12x longer
- Improving slowly (“speed of light”)

Use Extended Cache in CPU

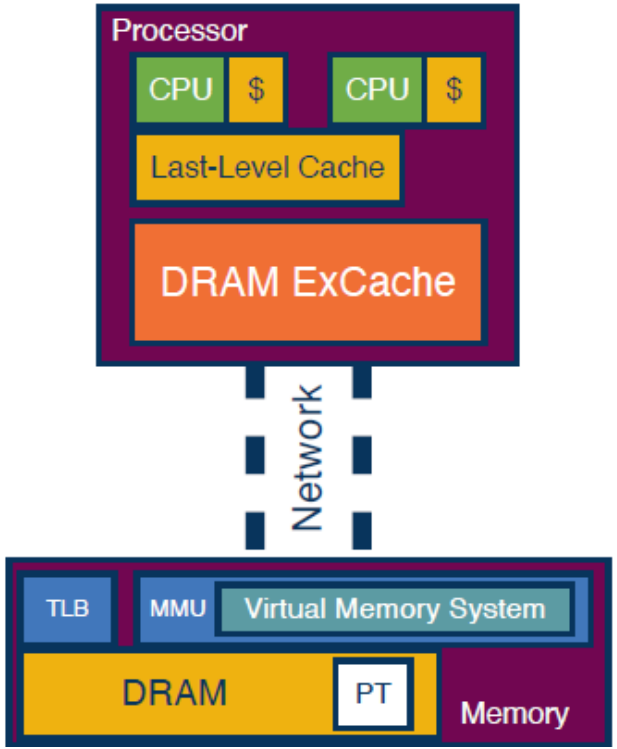


Figures adapted from the LegoOS Presentation at OSDI'20: https://github.com/WukLab/LegoOS

Add small DRAM/HBM with CPU

Extend Cache

- Software/Hardware co-managed
- Inclusive
- Virtual memory cache



Figures adapted from the LegoOS Presentation at OSDI'20: https://github.com/WukLab/LegoOS

Datacenter Scale

Thousands of components

General purpose server components

Specialized configurations for specific workloads

Hyperscale sizes (order of magnitude larger!)

https://blog.google/around-the-globe/google-asia/growing-our-data-center-in-singapore

Management Challenges

Application

Long running services
Batch jobs
Production vs. non-production

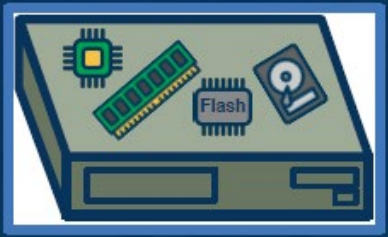
Multi-tenancy

Processes/tasks

Differ in end-to-end metrics:
- Latency-sensitive
- Throughput-intensive

Differ in resource requirements:
- Compute-intensive
- Demand accelerators
- Data-intensive
- Data access speed vs. data capacity

Orchestrate resource allocation and deployment



Service-level Agreement (SLA)
Service-level Objective (SLO)

Managing Resources

Omega (Eurosys 2013)

Borg (Eurosys 2015)

Kubernetes

Borg Terminology

Cell: a collection of machines

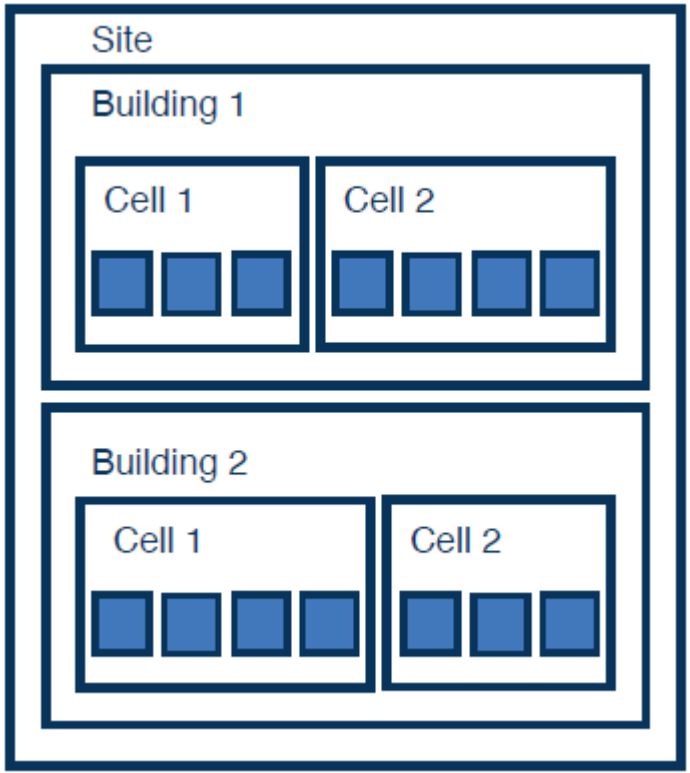
- Unit of management

Machines in a cell are part of one cluster

- High-performance “network fabric”

Cluster: set of machines in a single datacenter building

Site: a set of datacenter buildings



Application Job & Task States

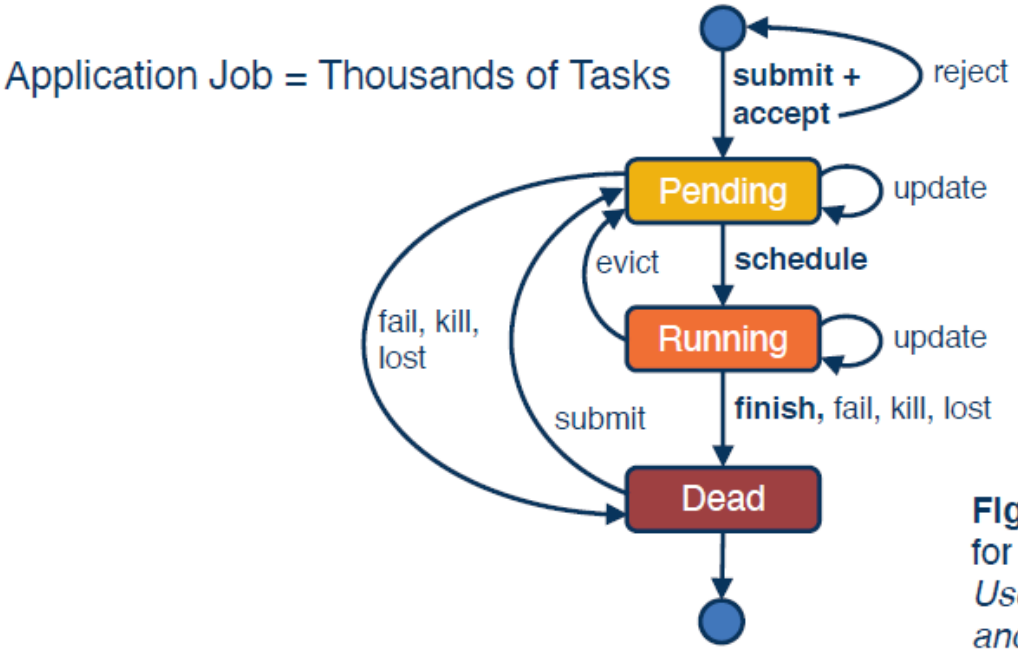


Figure 2: The state diagram for both jobs and tasks. *Users can trigger submit, kill, and update transitions.*

Adapted from: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/43438.pdf

Borg Architecture

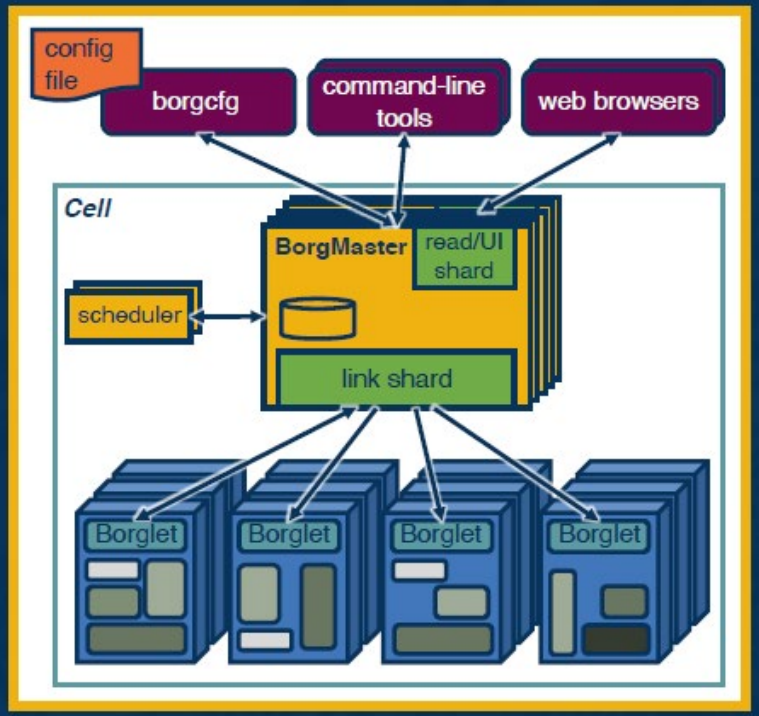


Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

Figure adapted from: https://res.infoq.com/news/2015/04/google-borg/en/resources/borg.png

Borg Architecture

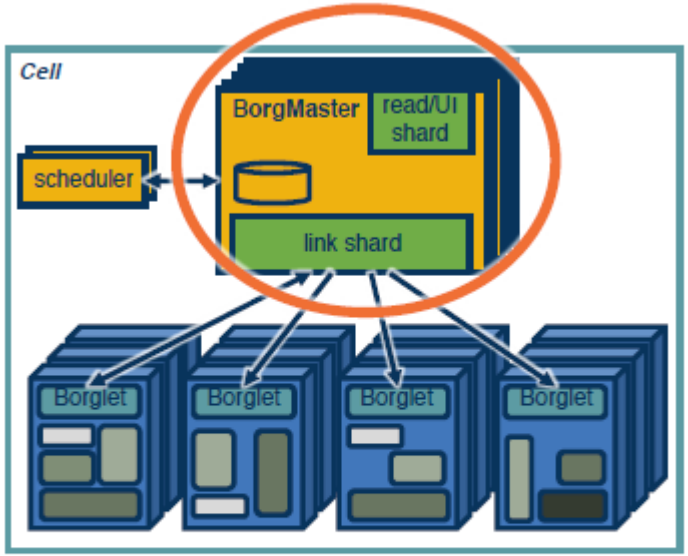


Figure adapted from: https://res.infoq.com/news/2015/04/google-borg/en/resources/borg.png

- Borgmaster is the brain of the Borg system
- One Borgmaster per cell
- Handles requests to execute/check job status
- Cell state maintained in memory
- Maintains **pending queue**
  - If job exceeds quota, not admitted
  - Quota assignment is *policy,* not Borg
- Assigns **tasks** to machines
- Monitors all machine state within the cell

Borg Architecture

Task Scheduler:

- Scans pending queue in priority order
- Checks feasibility, finds set of machines where tasks can run
- Finds best fit
- Forwards assignment to Borgmaster

Borgmaster:

- May pre-empt lower priority tasks
- Pre-empted tasks are returned to pending queue
- **Production** priority tasks not pre-empted.

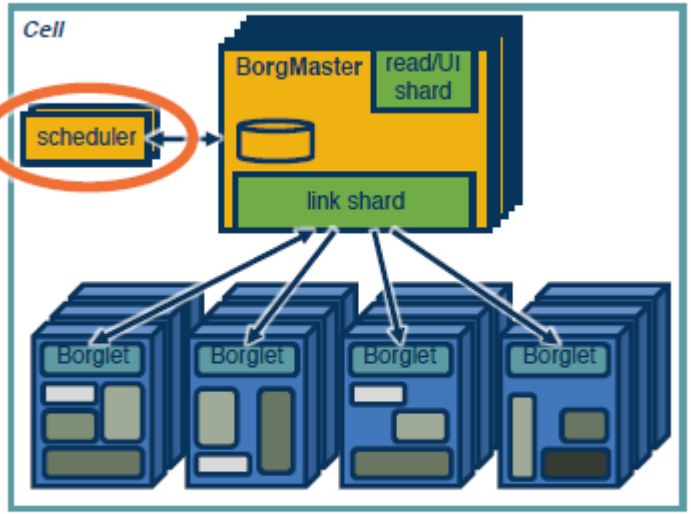


Figure adapted from: https://res.infoq.com/news/2015/04/google-borg/en/resources/borg.png

Borg Architecture

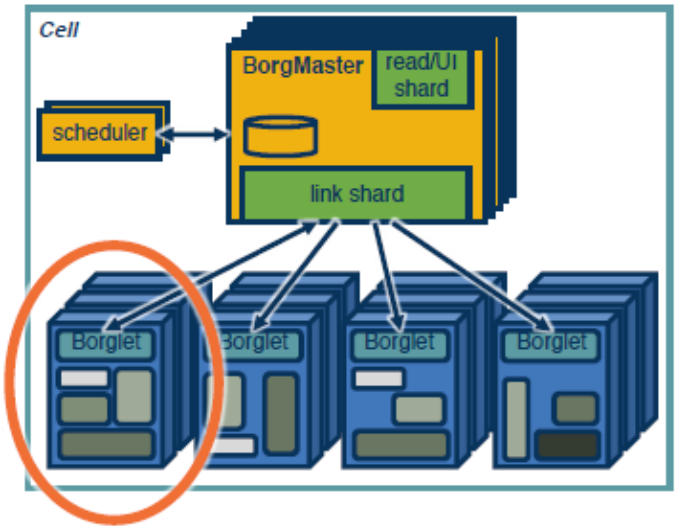


Figure adapted from: https://res.infoq.com/news/2015/04/google-borg/en/resources/borg.png

Borglet: local Borg agent on machine

- Start/stop tasks
- Restart tasks after failure
- Manages local resources
- Reports machine state to Borgmaster

Borgmaster polls Borglet

- Machine info
- Updated cell state
- No response = machine marked down

Borgmaster Reliability

Borgmaster has 5 replicas

Chubby lock acquire by leader; other replicas use lock collision info to find leader.

Only leader changes cell state.

Cell leader is *also* Paxos leader for replicated data store

Each replica contains cell state (Paxos-based key-value store)

Failover time is approximately **10 seconds**.

Borgmaster Availability

Reschedule pre-empted tasks

Reduces correlated failures

- Spreads job tasks across failure domains
  - Separate machines
  - Separate racks
  - Separate power domains

Limits task disruption rates

Avoids repating task/machine pairings that cause task/machine crashes

- “Learning”

Borg Scalability

Decouples task assignment from scheduling

- Asynchronous update/read from pending queue
- Permits different schedulers

Efficient communications

- Separate threads for RPCs and Borglet communications
- Use link shards to summarize information from Borglets

Optimize scoring of machine/task pairs

Efficient resource utilization

- Spread job tasks across machines
- Allow mixing production/non-production workloads

Scheduler Optimizations

Score Caching:

- Scores for task assignment cached until machine/task priorities change
- Small changes in resource quantities on machines are ignored

Equivalence classes:

- Group tasks with identical requirements
- Score computed once **per equivalence class**

Relaxed randomization:

- Scheduler examines machines in random order to finds enough to score
- Reduces amount of scoring/cache-invalidations when tasks enter/leave
- Speeds up assignment of tasks to machines

Performance Isolation

Tasks run in containers

Borglets manipulate container properties

Borg tasks have an application class

- Latency sensitive application classes: high priority
- Batch application classes: low priority

Compressible resources

- CPU cycles, rate based disk I/O bandwidth
- May be reclaimed with lower QoS, but continues running

Non-compressible resources

- Memory, disk space
- Reclaim requires halting task

Lesson Review

Data Center Services

Trends and Services

High-speed RDMA and programmable networks

Resource heterogeneity

Resource disaggregation

Resource management and orchestration

Questions?


THE UNIVERSITY OF BRITISH COLUMBIA
THE UNIVERSITY OF BRITISH