Advanced Operating Systems: Three Easy Pieces

Datacenter OS

Outline

- **The Datacenter needs an OS**
  - Motivation, why, what is needed..
- **What is Apache Mesos**
- **Twitter DC/OS based on Mesos**
  - Master Node
  - Agent Node
  - Frameworks
  - Others

The Datacenter needs an OS

The Datacenter is the new Computer

- **Datacenter is running today's most popular consumer applications:**
  - Facebook, Google, iCloud, etc.
- Needed for big data in business & Science
- Widely accessible through cloud computing/Internet

Claim: **this new computer needs an operating system**

Why Datacenter needs an OS

- **Growing diversity of applications:**
  - **Computing frameworks:** MapReduce, Dryad, Pregel[1], Percolator, Dremel[2]
  - **Storage systems:** GFS, BigTable, Dynamo, etc.
- **Growing diversity of users:**
  - 200+ Hive users at Facebook
- Same reasons computers needed one!

1 **Pregel**: https://kowshik.github.io/JPregel/pregel_paper.pdf
System for large-scale Graph Processing

2. DremeL: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36632.pdf
Interactive Analysis of Web-scale Dataset

What Operating System Provide

Resource Sharing

time-sharing, virtual memory, …

Data Sharing

files, pipes, IPC, …

Programming Abstractions

libraries, languages

Debugging & Monitoring

ptrace, DTrace, top, …

What Operating System Provide

Resource Sharing

time-sharing, virtual memory

Most importantly: **an ecosystem**

Dat

files

ing

ns

ages

…enabling independently developed software to interoperate seamlessly

Debugging & Monitoring

ptrace, DTrace, top, …

Today's Datacenter Operating System

- **Platforms like Hadoop are well-aware of these issues:**
  - **Inter-user resource sharing**, but at the level of MapReduce jobs (though this is changing - YARN)
  - **InputFormat API for storage systems** (but what happens with the next hot platform after Hadoop?)
- **Other examples:** Amazon services, Google stack

The **problems** motivating a datacenter OS are well recognized, but solutions are **narrowly targeted**

Can researchers take a longer-term view?

Tomorrow’s Datacenter OS

Resource Sharing

time-sharing, virtual memory, …

Data Sharing

files, pipes, IPC, …

Programming Abstractions

libraries, languages

Debugging & Monitoring

ptrace, DTrace, top, …

Resource Sharing

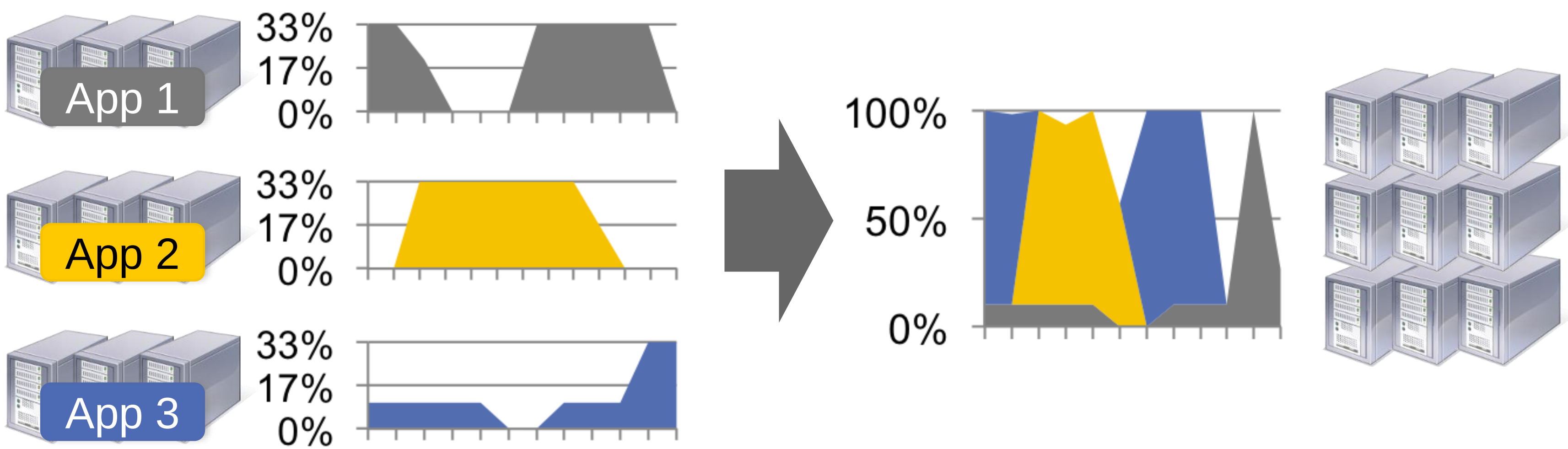
"To solve these interaction problems we would like to have a computer made simultaneously available to many users in a manner somewhat like a telephone exchange. Each user would be able to use a console at his own pace and without concern for the activity of others using the system."

– Fernando J. Corbató, 1962

Resource Sharing

- **Today**, cluster apps are built to run independently and assume they own a fixed set of nodes
- **Result:** inefficient static partitioning

- **What's the right interface for dynamic sharing?**



Memory Management

- **Memory is an increasingly important resource:**
  - In-memory iterative processing (AllegroGraph “Graph DB”, Pregel, Spark, etc.)
  - DFS cache for MapReduce cluster could serve 90% of jobs at Facebook (HotOS ‘11)

- **What are the right memory management algorithms for a parallel analytics cluster?**

Programming and Debugging

- Although there are new programming models for applications, **system programming remains hard**:
  - Can we identify useful common abstractions? (Chubby[1] "lock service for distributed system / Big Table", Sinfonia[2], Mesos are some examples)
  - How much can languages (e.g. Go, Erlang) help?

- **Debugging is *very* hard:**
  - Magpie[3], X-Trace, Dapper[4] are some steps here

- Can a clean-slate design of the stack help?

1 **Chubby**: https://static.googleusercontent.com/media/research.google.com/en//archive/chubby-osdi06.pdf

2 **Sinfonia**: new paradigm to building scalable distributed systems: http://www.sosp2007.org/papers/sosp064-aguilera.pdf

3 **Magpie**: online modelling & performance-aware systems: https://www.usenix.org/legacy/publications/library/proceedings/hotos03/tech/full_papers/barham/barham_html/paper.html

4 **Dapper** is a large scale Distributed Systems tracing infrastructure: https://research.google.com/pubs/pub36356.html

What is Needed

- **Focus on paradigms, not only performance:**
  - Industry is spending a lot of time on performance
- **Explore clean-slate approaches:**
  - Much datacenter software is written from scratch
  - People using Erlang, Scala, functional models (MR)
- **Bring cluster computing to non-experts:**
  - Most impactful (datacenter as the new workstation)
  - Hard to make a Google-scale stack usable without a Google-scale ops team

What is Apache Mesos

1. Mesos Overview

1. Overview

- **Mesos**[1] is a platform for Fine-Grained resource sharing in the data center – Cluster Manager.

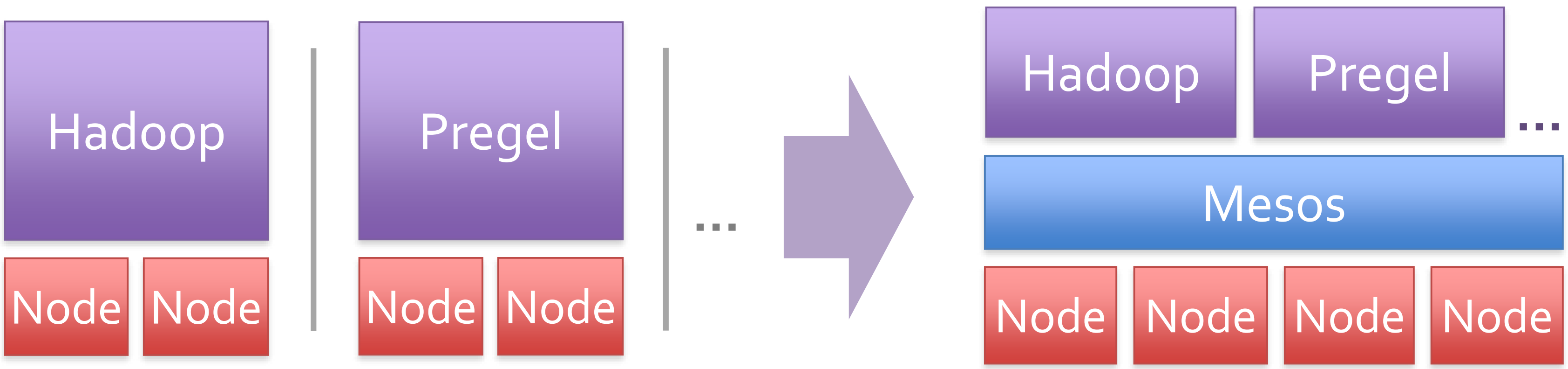
- While there is rapid innovation in cluster computing frameworks, there is no single framework that is optimal for all applications.

- **An Alternative** is to run multiple frameworks in a single cluster, **without static partitioning**, to maximize utilization and to share data between frameworks ← Mesos

1 **Mesos**: Cluster Manager: http://mesos.apache.org/

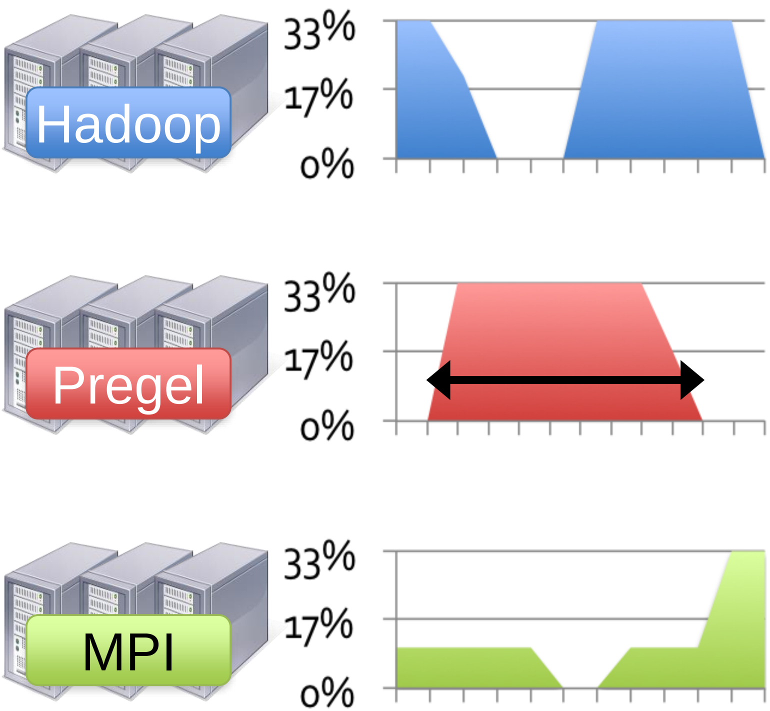
1. Static vs. Dynamic Partitioning

- **Mesos** is a common resource sharing layer over which diverse frameworks can run

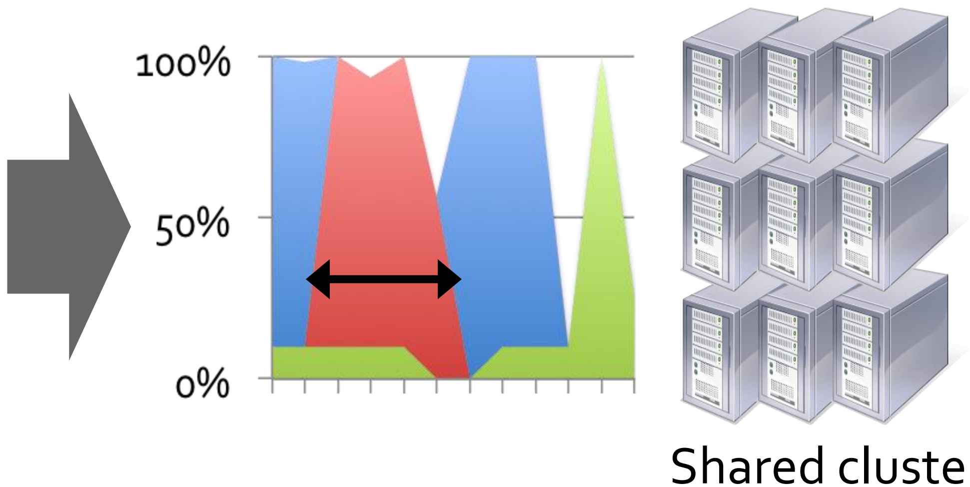


1. Static vs. Dynamic Partitioning

Today: static partitioning



Mesos: dynamic sharing



1. Other Benefits

- **Run multiple instances of the *same* framework:**
  - Isolate production and experimental jobs
  - Run multiple versions of the same framework concurrently

- **Build *specialized frameworks* targeting particular problem domains:**
  - Better performance than general-purpose abstractions

2. Mesos Goals

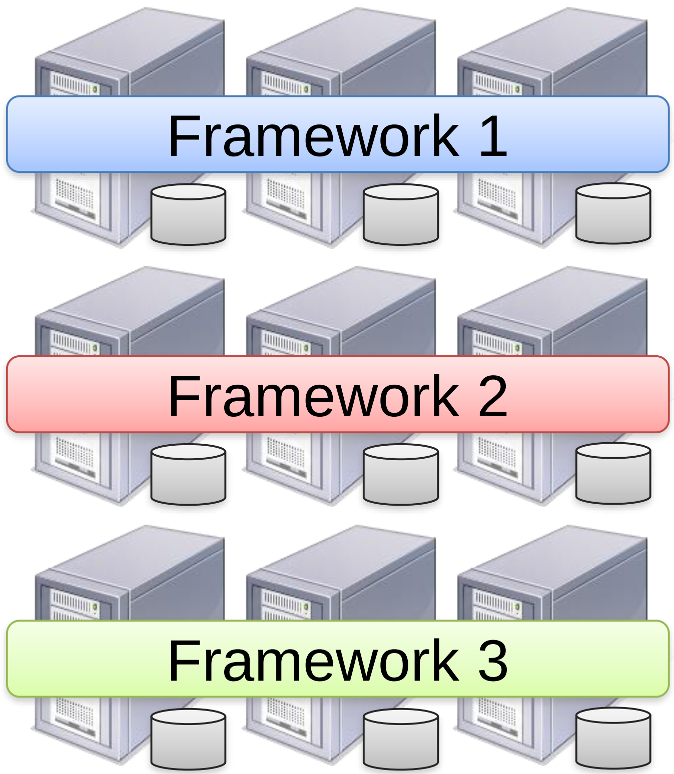
2. Mesos Goals

- **High utilization** of resources
- **Support diverse frameworks** (current & future)
- **Scalability** to 10,000's of nodes
- **Reliability** in face of failures
- **Fine-grain sharing**
- **Support the Resource Offers Model:** simple application-controlled scheduling mechanism

Resulting design: Small microkernel-like core that pushes scheduling logic to frameworks

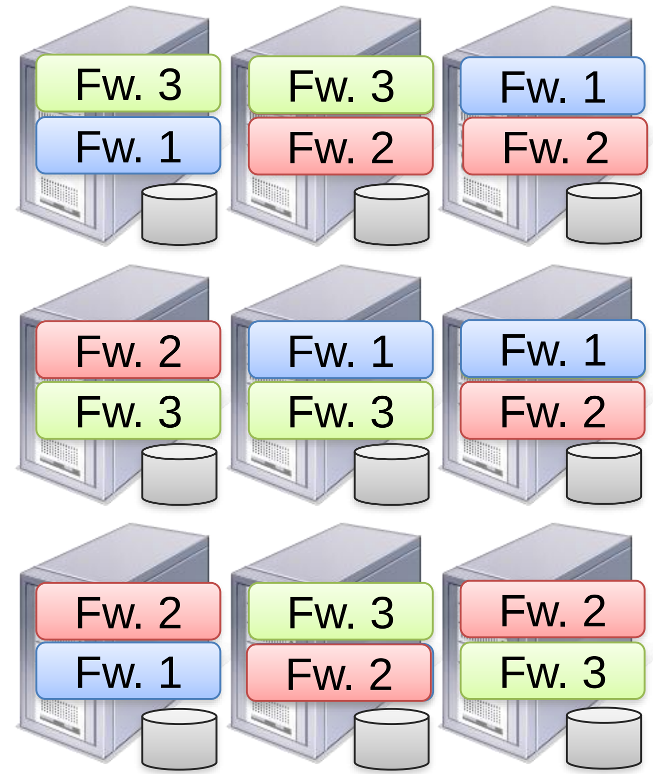
2. Fine-Grained Sharing

Coarse-Grained Sharing (HPC):



Storage System (e.g. HDFS)

Fine-Grained Sharing (Mesos):



Storage System (e.g. HDFS)

+ Improved utilization, responsiveness, data locality

2. Resource Offers Model

- **Option:** Global scheduler
  - Frameworks express needs in a specification language, global scheduler matches them to resources

  \+ Can make optimal decisions

- **Complex:** language must support all framework needs:
  - Difficult to scale and be robust
  - Future frameworks may have unanticipated needs

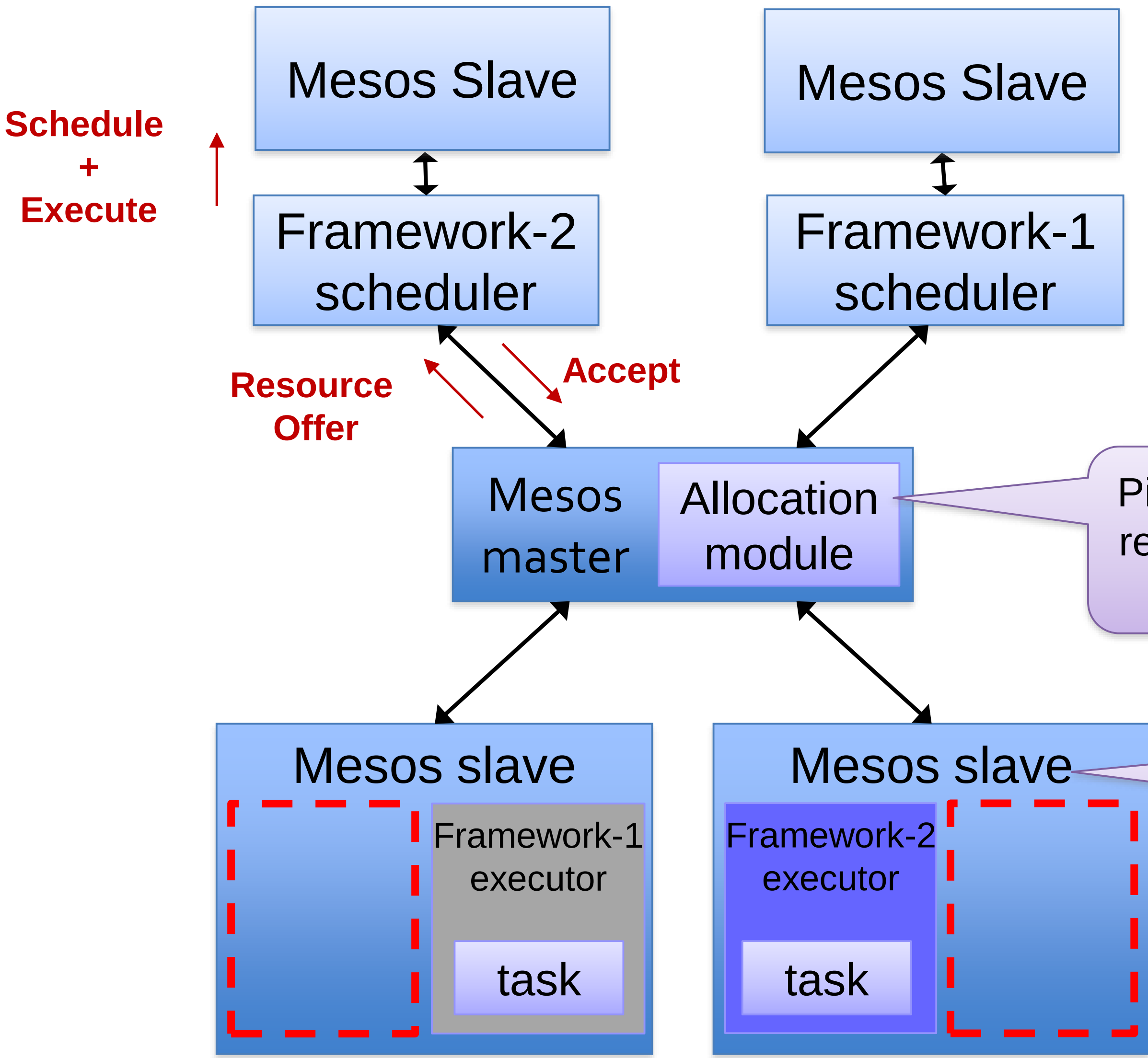
2. Mesos Resource Offers

- **Mesos:** Resource offers
  - Offer available resources to frameworks, let them pick which resources to use and which tasks to launch

  \+ Keeps Mesos simple; makes it possible to support future frameworks

  \- Decentralized decisions might not be optimal

2. Mesos Architecture



Resource offer =
list of (node, availableResources)

E.g. { (node1, <2 CPUs, 4 GB>),
(node2, <3 CPUs, 2 GB>) }

Pick framework to offer resources to, i.e., send Resource offers

Launches and isolates executors

3. Results

3. Mesos vs. Static Partitioning

- Compared performance with statically partitioned cluster where each framework gets 25% of nodes

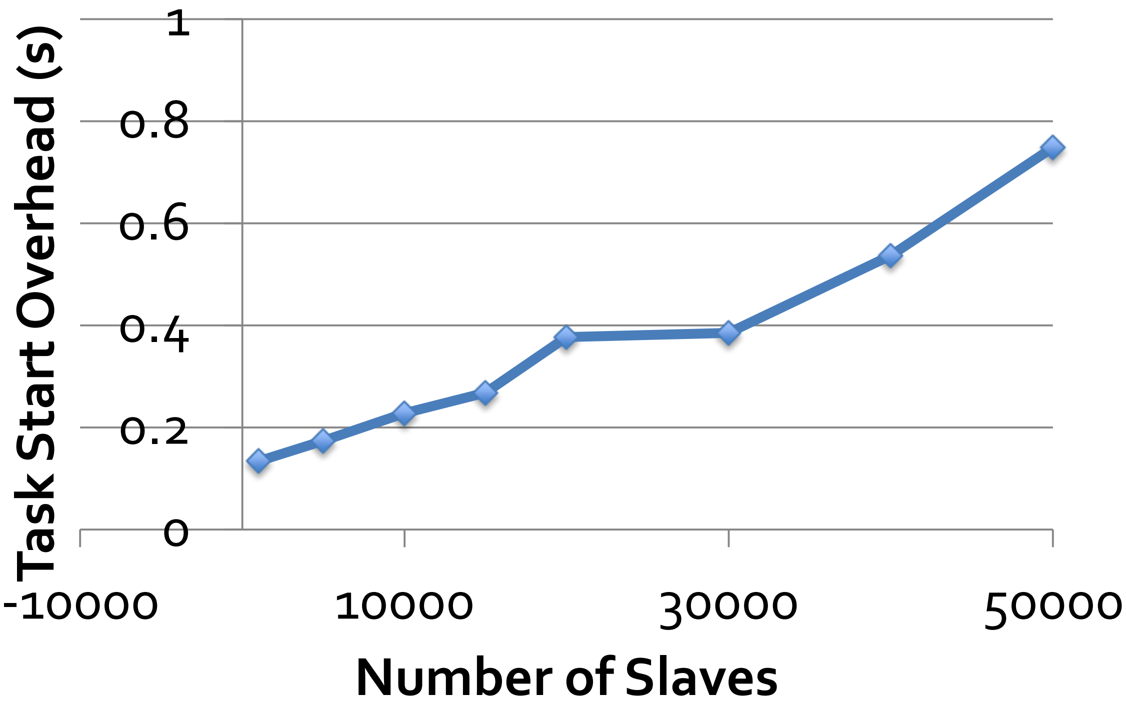
Framework	Speedup on Mesos
Facebook Hadoop Mix	1.14 ×
Large Hadoop Mix	2.10 ×
Spark	1.26 ×
Torque / MPI	0.96 ×

- Overall performance improvement = (1.14 + 2.1 + 1.26 + 0/96)/4 = 1.365 ➔ 36.5% improvement

3. Mesos Scalability

- Mesos only performs *inter-framework* scheduling (e.g. fair sharing), which is easier than intra-framework scheduling

Result:
Scaled to 50,000 emulated slaves, 200 frameworks, 100K tasks (30s len)

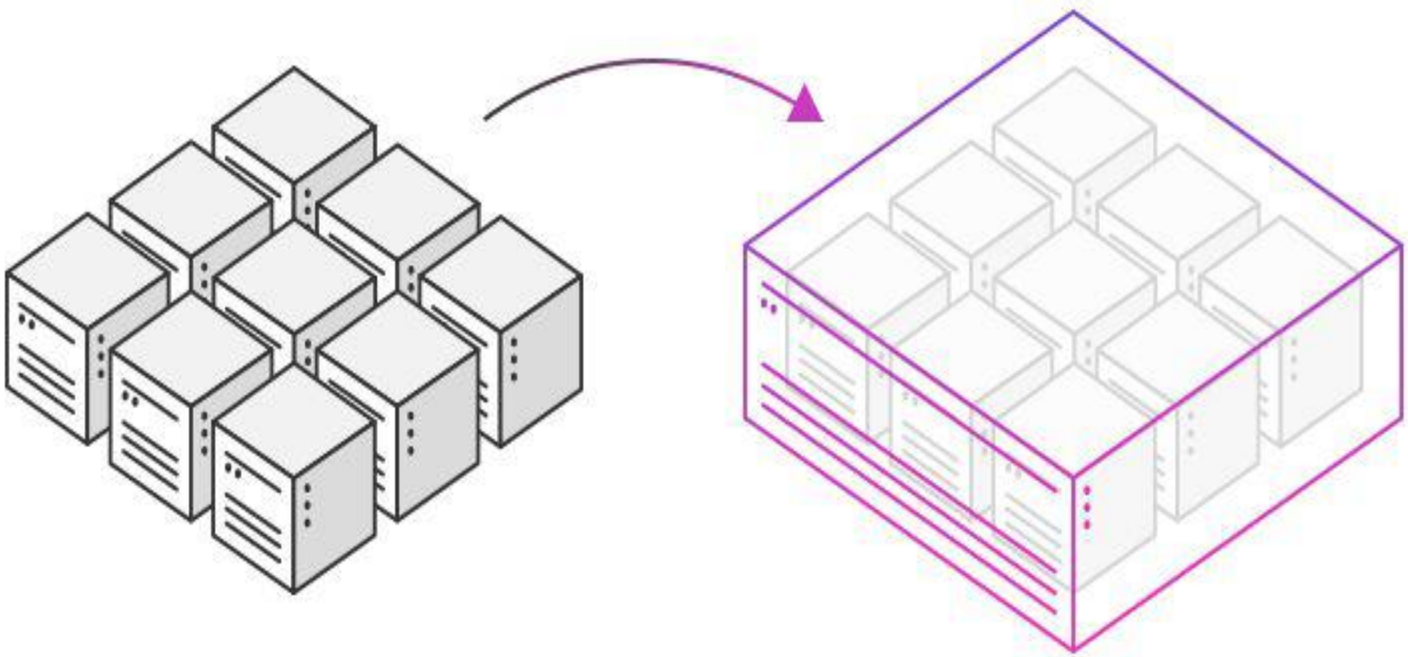


3. Fault Tolerance

- **Mesos master has only *soft state*:** list of currently running frameworks and tasks
- **Rebuild** when frameworks and slaves re-register with new master after a failure
- **Result: fault detection and recovery in ~10 sec**

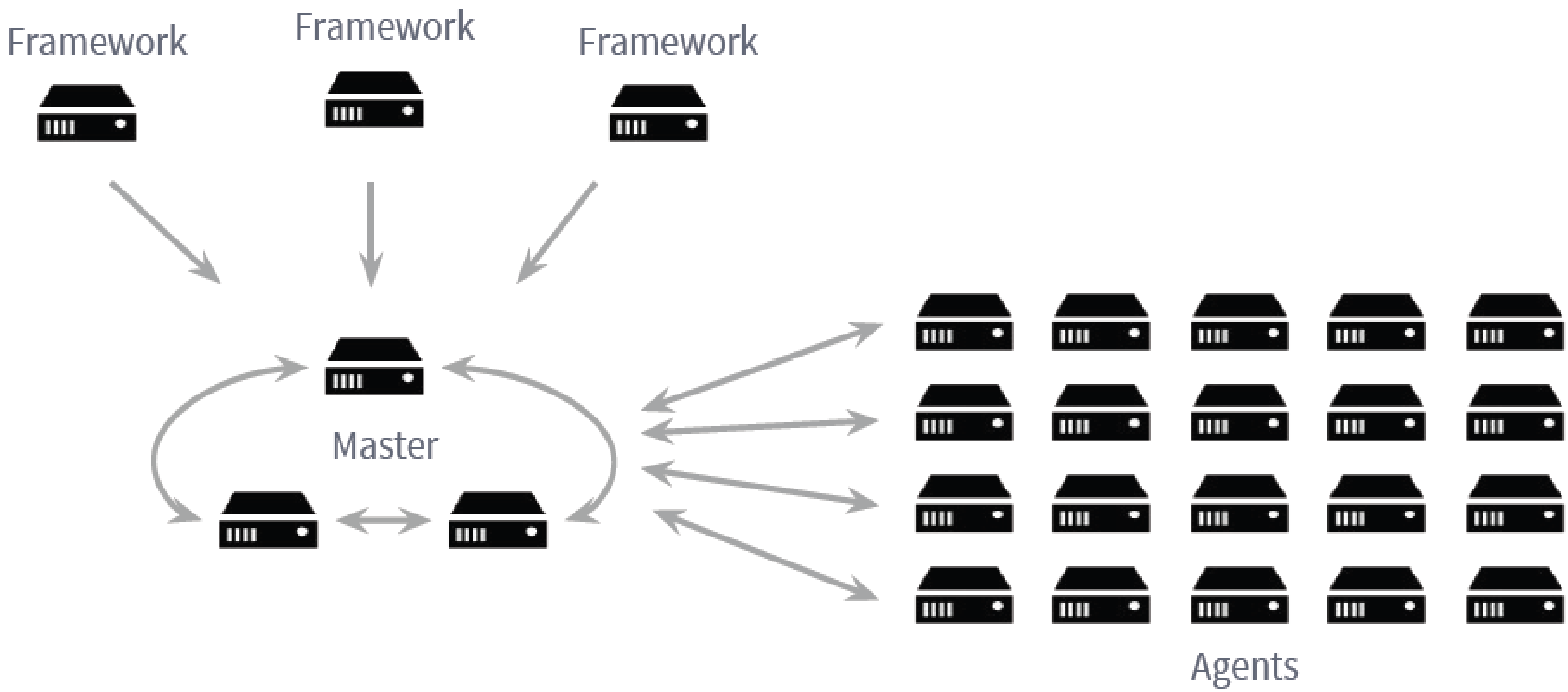
Twitter DC/OS based on Mesos - Aurora

The Datacenter Computer

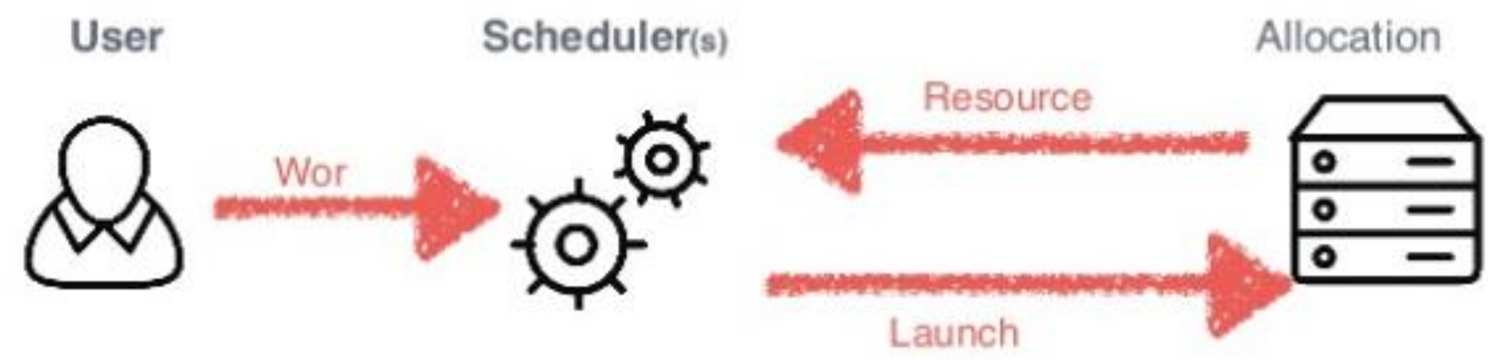


* Do not confuse Twitter's Aurora with Amazon's Aurora. Amazon Aurora is an RDBMS that is compatible with MySQL and PostgreSQL and is built for the cloud.

Two-levels Scheduling

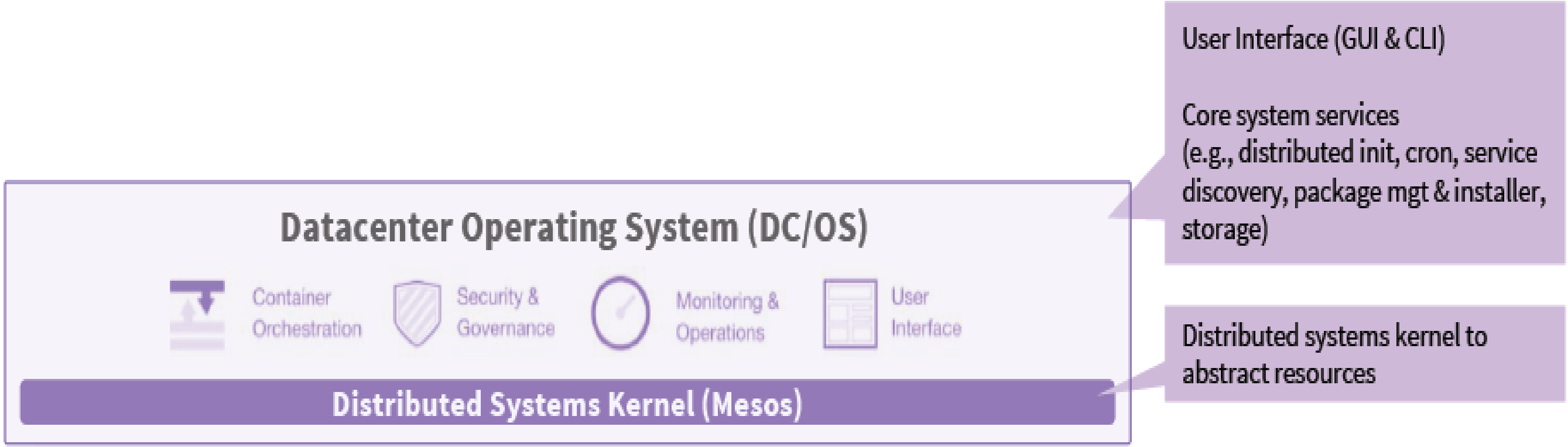


Two-levels Scheduling



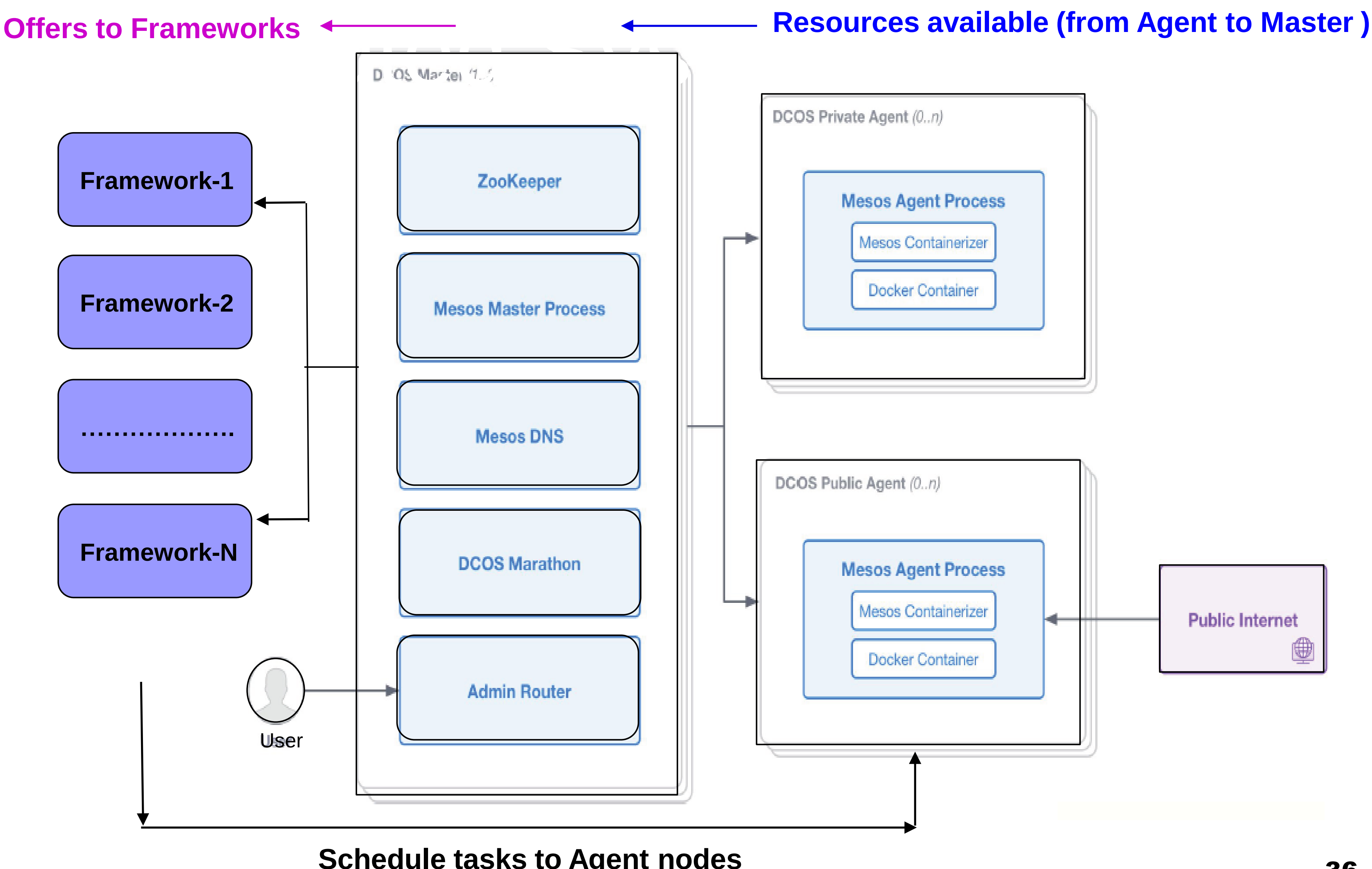
1. Master (Mesos+) schedule global resources to a given framework
2. Framework schedule jobs to agents (Mesos agent+)

Datacenter Operating System (DC/OS)



- **What is included:**
  - Mesos!
  - API, CLI and GUI
  - Service discovery & Load balancing
  - Storage Volumes
  - Package Manager
  - Installer on Premise and on Cloud

Datacenter OS Architecture



Datacenter OS Features

- **Kernel == Apache Mesos,** scaling up to 10,000 nodes
- **Fault tolerance in all components, rolling upgrades throughout**
- **Containers first class citizens** (LXC, Docker)
- **Local OS per node** (container enabled)
- **Scheduling** (long-lived, batch)
- **Service discovery, monitoring, logging, debugging**

Datacenter Components

1. **Master Node:** one master is assigned as the leader, manage the metadata for the whole system and propagate the metadata to other masters using quorum replication
2. **Agent Node:** can support multiple tasks executing concurrently belonging to different frameworks with complete isolation, using cgroups (Linux control groups), and managing resources consumption by each task to be limited to what is assigned to that task
3. **Frameworks:** representing different application paradigms, e.g., time sharing, batch, etc. An individual Framework will receive resource offers from the master and schedule tasks to agent nodes accordingly.

Datacenter Components

1. **Master Node:**
   - Resource allocation
   - Resource de-allocation
   - Resource reservation
   - Resource isolation
   - Resource monitoring
   - Failure detection
   - Package distribution
   - Task starting, killing, cleanup
   - Volume management
   - ….

2. **Agent Node:**
   - Notify Master with available resources
   - Execute tasks assigned to them by different Frameworks
   - Use cgroups to isolate tasks from each other and to ensure that no task consume resources beyond what was assigned to them

3. **Framework:**
   - Receive resource offers from the master and assign tasks to be executed on agent nodes
   - Handle task failure or loss of agent node

Datacenter Components: Master

- Master is the cluster coordinator (cluster configuration).
- The cluster will have multiple master nodes (for fault tolerance). One master functions as the **leader** and propagates the metadata updates to the other masters using quorum replication. If the leader dies, one of the masters is elected as the new leader.
- The leader monitor the state of the cluster.
- Leader receives resources available from the agent nodes and generates "resource offers" to the available frameworks based on some policy; framework scheduler may accept (which of the offered resources to use) or reject the offer!
- The most significant factor of a leader to manage large cluster is available RAM in the leader node.
- Twitter with cluster of 30,000 nodes they use 5 master nodes.

Datacenter Components: Agent

- **Agent node** notifies the leader node with available resources (CPU, RAM, Storage, Ports)
- **Agent node** execute tasks assigned to them by any of the frameworks supported in the cluster
- **Two tasks belong to different frameworks** are completely isolated from each other as well the agent node guarantee that a task can use only resources assigned to them, i.e., no task can abuse the resources on that node.
- **You can assign resource to a specific role**, i.e., restricting this resource to a specific framework; otherwise resources can be used by any application.

Datacenter Components: Framework

- Tasks are assigned to Mesos on the agent node by framework.
- Framework is an application that uses Mesos APIs to receive resource offers from the Master and replies to instruct agent nodes to execute appropriate tasks.
- Framework handles task failure or loss of an agent node.

Datacenter Components: Others

- Zookeeper
- Admin Router
- Metronome
- Marathon
- Messos-DNS: service discovery
- Cosmos: package manager; run on all master nodes

END