CS 7172
Parallel and Distributed Computation

Centralized Structure

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Outline

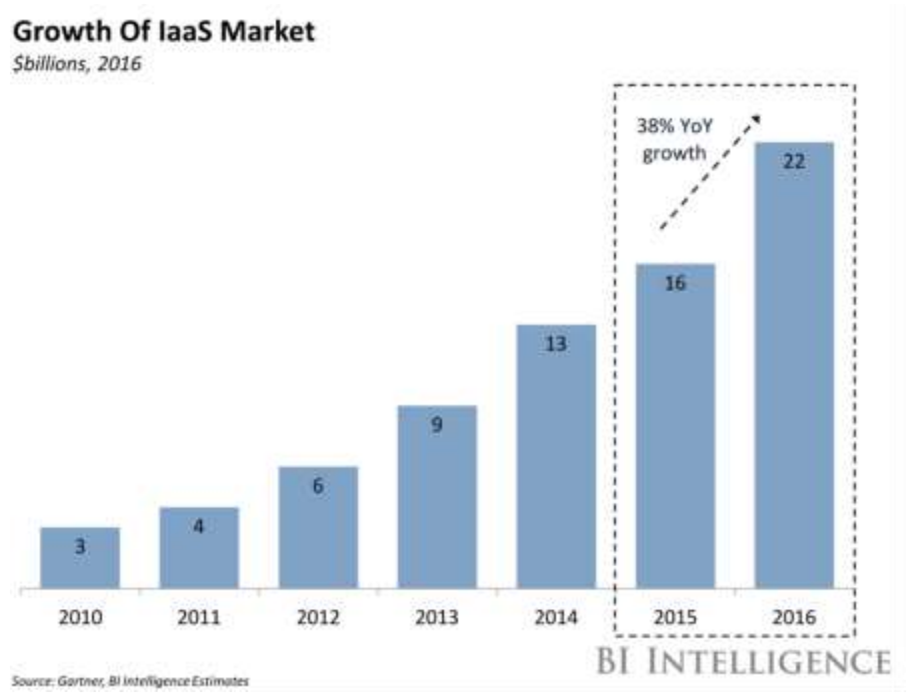
- Computer networks, primarily from an application perspective
- Protocol layering
- Client-server architecture
- End-to-end principle
- TCP
- Socket programming

Cloud

- Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user.

- Today, almost every services are built on the cloud
  - Video
  - Social network
  - News
  - Entertainment
  - …

Scale of the Cloud

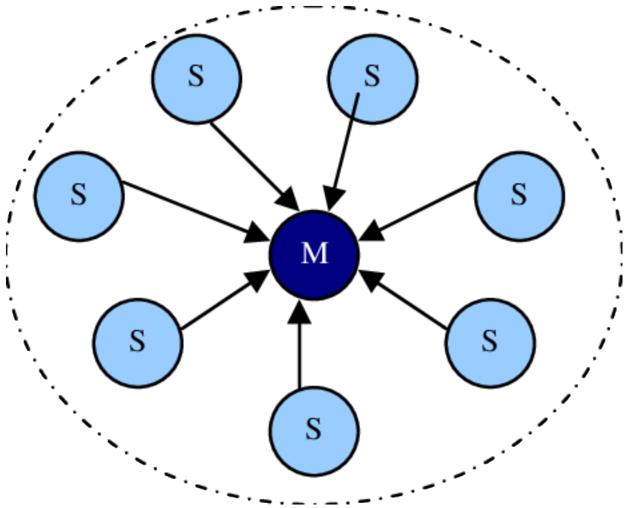


How to manage millions of servers?

How to manage millions of servers?

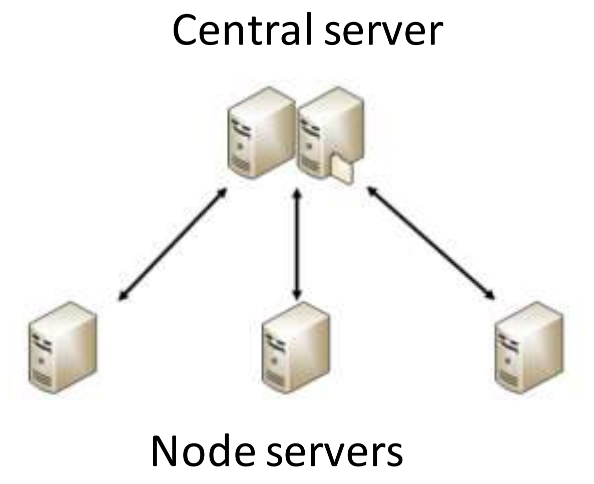
- In many scenarios, our requests are aggregated on one server, and this server coordinates the relationship between our requests and other servers.

- The way that one server manages other servers in a unified manner is a ***centralized*** structure in a distributed architecture



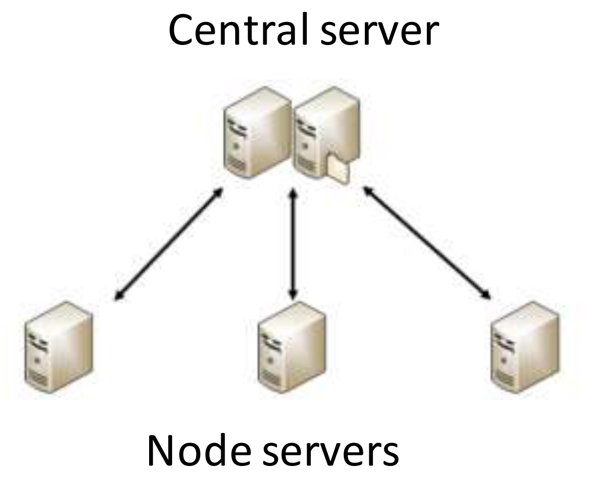
What is Centralized Structure?

- The central server is composed of one or more servers.
- All data in the system are stored in the central server, and all business in the system is processed by the central server first. The central server performs resource and task scheduling.
- Multiple node servers are connected to the central server and report their information to the central server. The central server sends tasks to the node server based on the information; the node server executes the tasks and sends the results & feedback to the central server.



What is Centralized Structure?

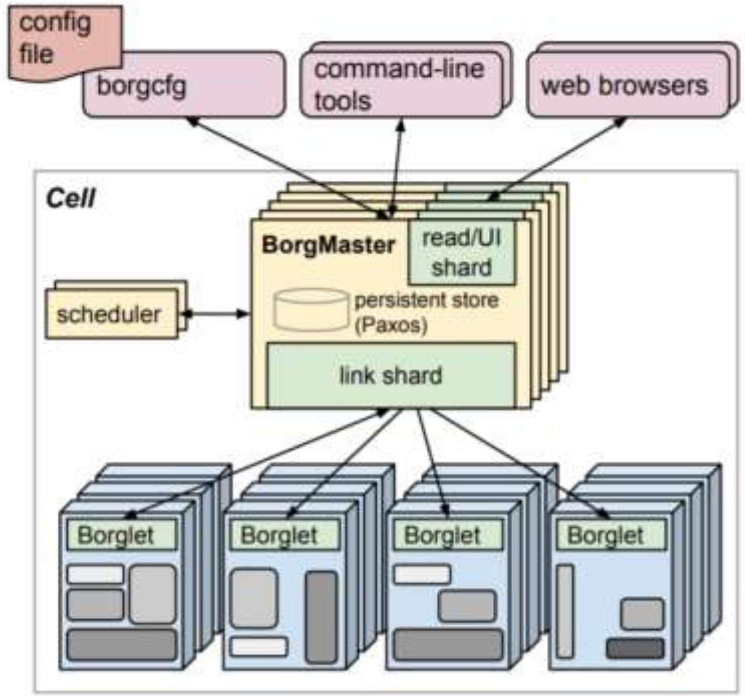
- Advantages:
  - Simple structure
  - Centralized manage and scheduling, the node servers do not need to communicate with each other and only need to communicate with the central node



Example 1 of Centralized Structure: Google Borg

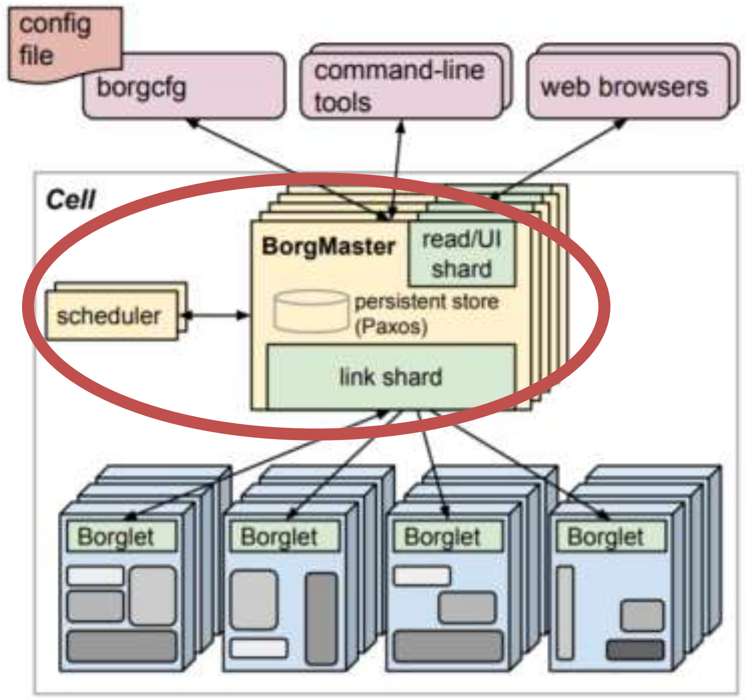
- Borg is a cluster management system used internally by Google. It adopts a typical centralized structure and is responsible for submitting, scheduling, starting, restarting, and managing all applications running in Google.

- In Borg, a cluster is called a cell, and each cell has a central server called *BorgMaster*; other servers are node servers or slave servers called *Borglet*



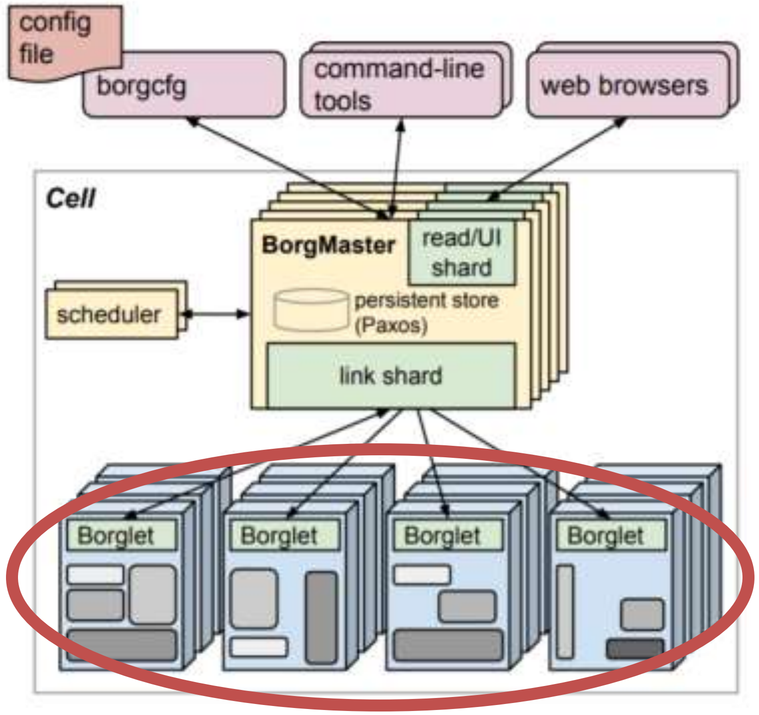
Example 1 of Centralized Structure: Google Borg

- BorgMaster contains two processes:
  - Borgmaster main process:
    - Handle RPCs from clients, e.g., query, update, etc.
    - Manage all entity states and communicate with Borglet
  - Scheduler process:
    - Responsible for scheduling, e.g., find a node server for one task execution



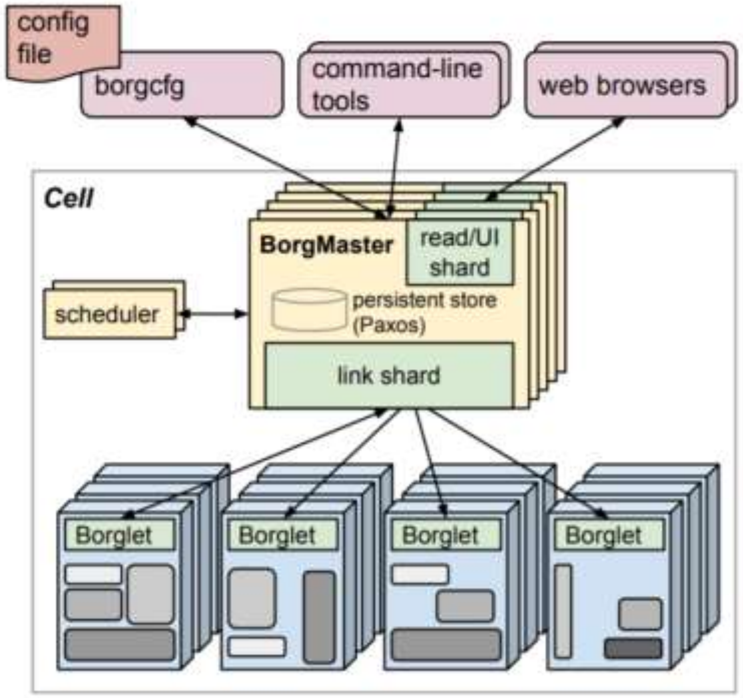
Example 1 of Centralized Structure: Google Borg

- Borglet, agent on each node server:
    - Responsible for task start, stop, reboot, etc.
    - Manage the server on local node server
    - Send task status, server status, etc. to the BorgMaster



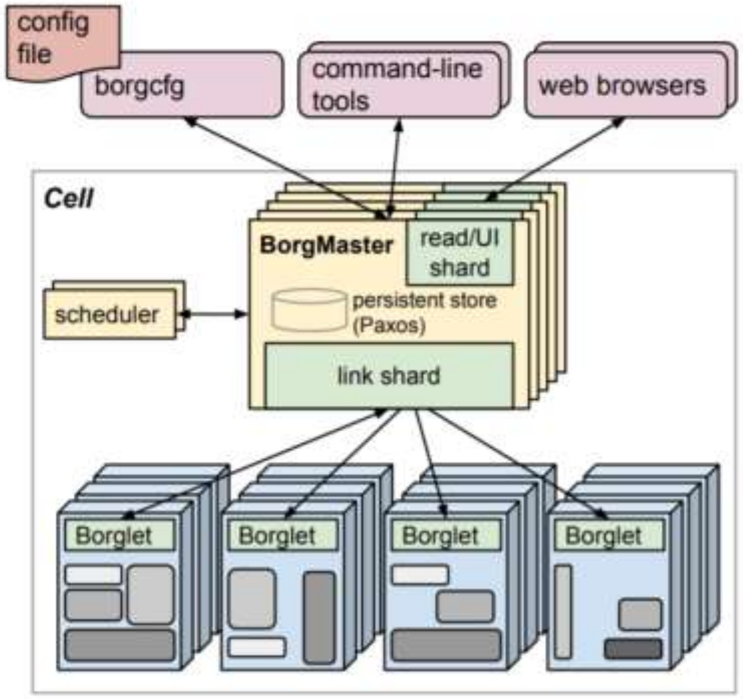
Example 1 of Centralized Structure: Google Borg

- Tasks on Borg
  - Batch jobs: take several seconds to several days to complete. These jobs are not sensitive to latency and short-term performance fluctuations
  - Long jobs: running forever and is responsible for handling short, latency-sensitive requests (ranging from microseconds to hundreds of milliseconds)



Example 1 of Centralized Structure: Google Borg

- Advantages of Borg
  - Developers only need to focus on the application, not on the underlying resource management. It hides resource details and users can focus on developing applications.
  - High reliability and availability to support millions of applications.
  - Support the management and operation of thousands of servers.



What is Docker container?

- https://www.youtube.com/watch?v=RyxXe5mbzIU

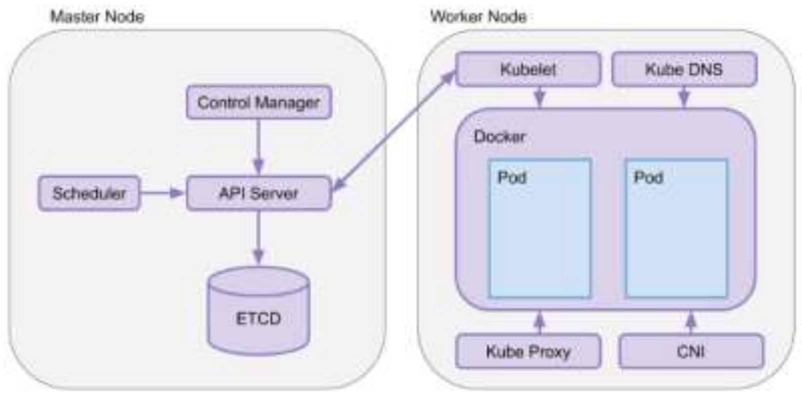
Example 2 of Centralized Structure: Kubernetes

- Kubernetes is Google's open source container cluster management system

- Kubernetes supports running containerized applications on the nodes of the cluster, which can automate container operations, such as deployment, scheduling, and elastic scaling between nodes, etc.

kubernetes

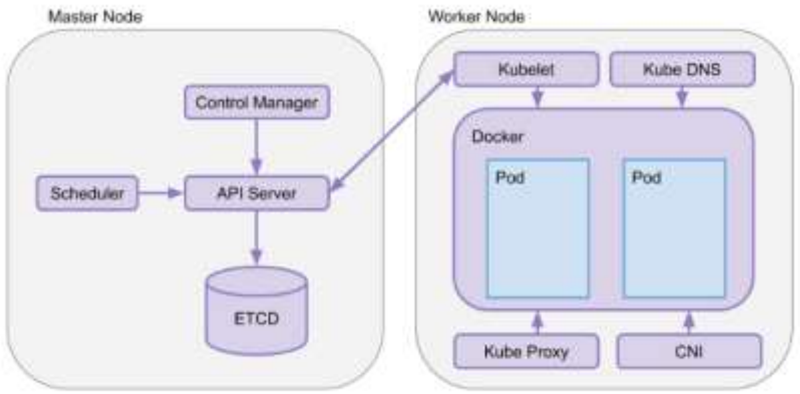
https://kubernetes.io/



Example 2 of Centralized Structure: Kubernetes

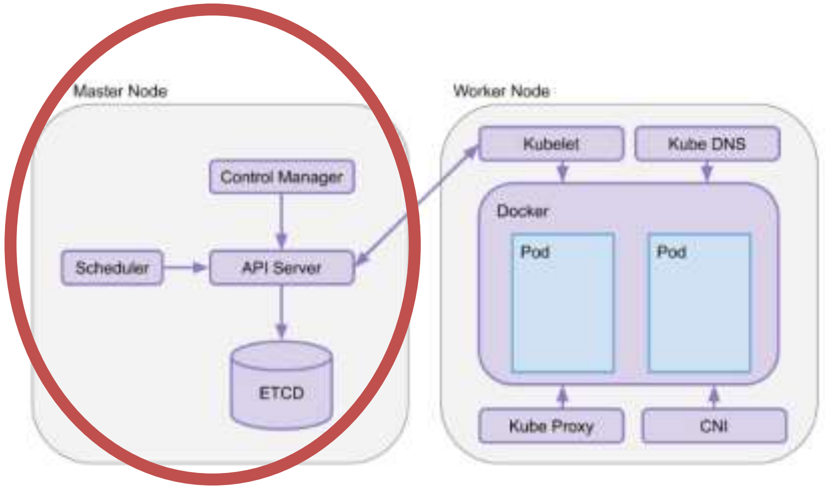
- Kubernetes is a typical centralized structure.

- A Kubernetes cluster is composed of Master nodes and Worker nodes, as well as the client command line tool kubectl and other additional items.



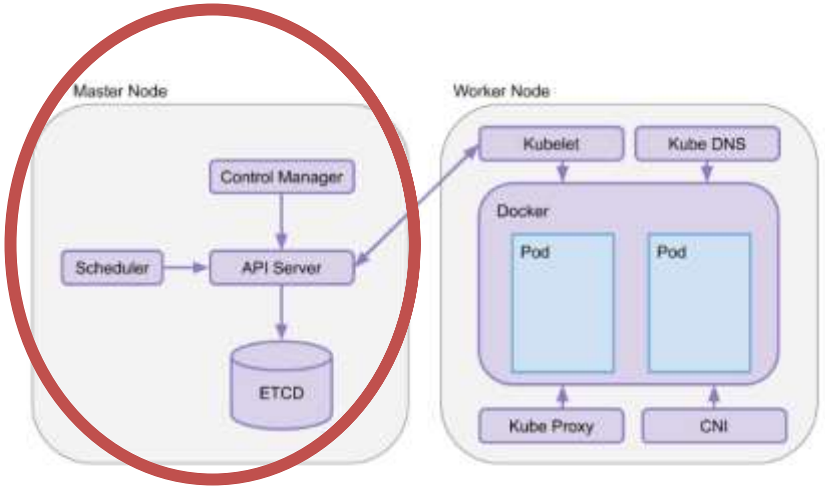
Example 2 of Centralized Structure: Kubernetes

- Master node:
  - It runs on a central server.
  - The master node is composed of API Server, Scheduler, Cluster State Store, and Control Manger Server
  - Master node is responsible for scheduling and management of the cluster.



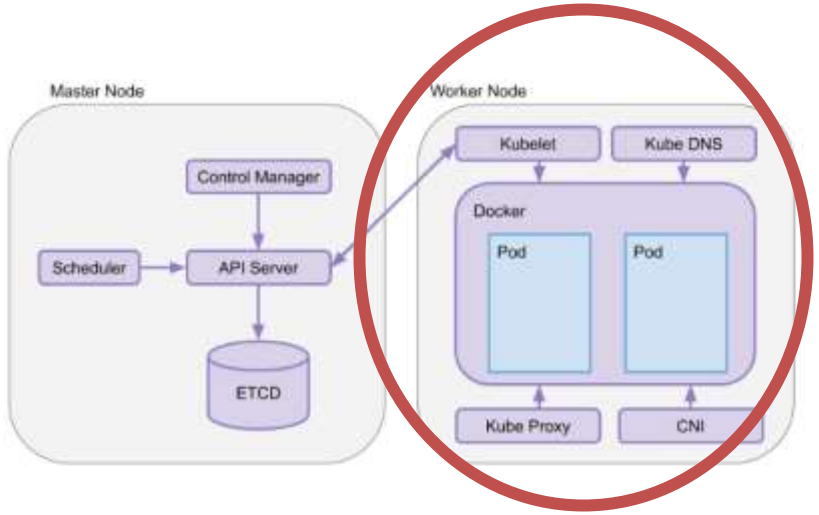
Example 2 of Centralized Structure: Kubernetes

- Master node:
  - API Server: entrance to all commands and is responsible for processing operations and executing related business logic
  - Scheduler: based on the resources required by the container and the resource information of the node server, it automatically select the appropriate node server for the container
  - Cluster State Store: it uses etcd by default. Etcd is a distributed key-value store that is mainly used for shared configuration and service discovery.
  - Control Manager: used to perform most of the cluster-level functions, such as lifecycle functions (event garbage collection, etc.) and API business logic.



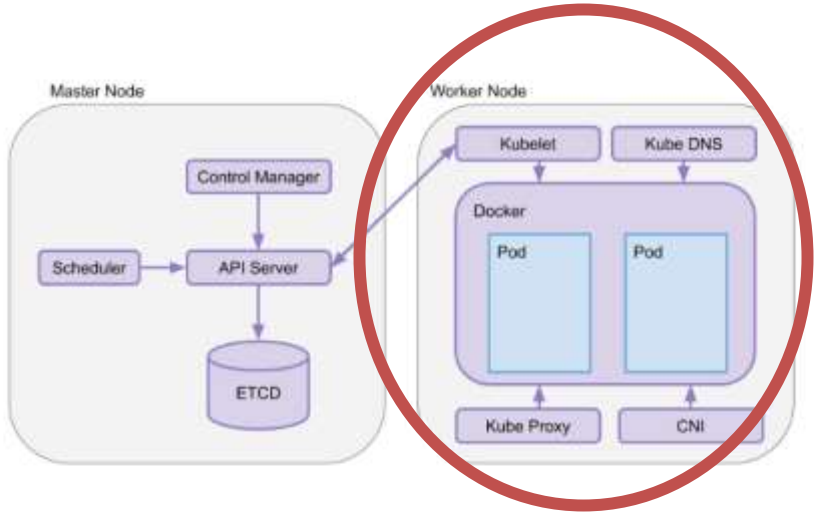
Example 2 of Centralized Structure: Kubernetes

- Worker node:
  - runs on the node server, including kubelet and kube-proxy core components
  - is responsible for running the application container.



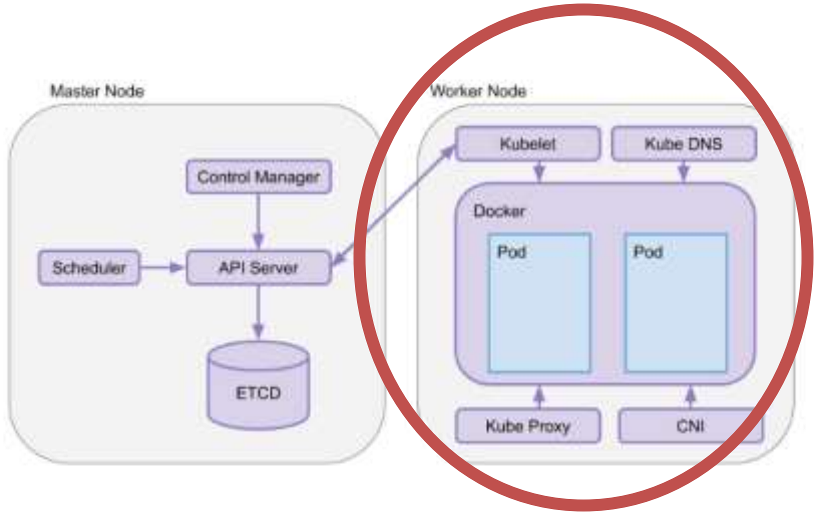
Example 2 of Centralized Structure: Kubernetes

- Worker node:
  - kubelet: used to interact with the API Server through the command line, and operate the Worker node according to the received request. By communicating with the API server, it receives requests or command from the Master node and control the containers on the Worker node (for example, restarting the failed container).
  - kube-proxy: responsible for creating a network proxy and load balancing service for the container.



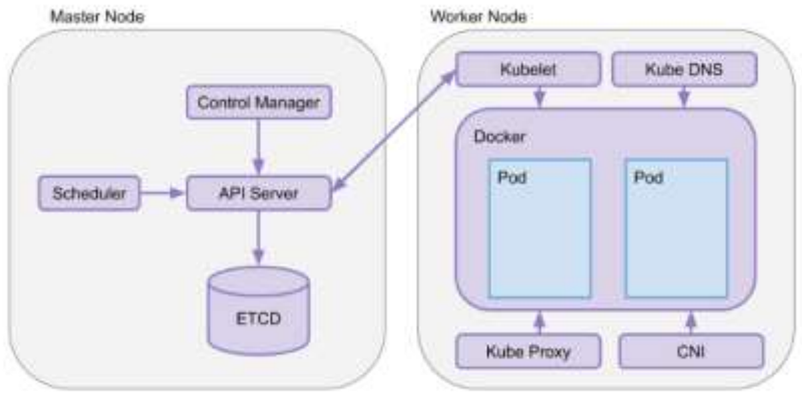
Example 2 of Centralized Structure: Kubernetes

- Worker node:
  - Kube DNS: is responsible for providing DNS services for the entire cluster
  - CNI is a standard universal interface for the Container Network Interface, which is used for container management systems and network plug-ins.



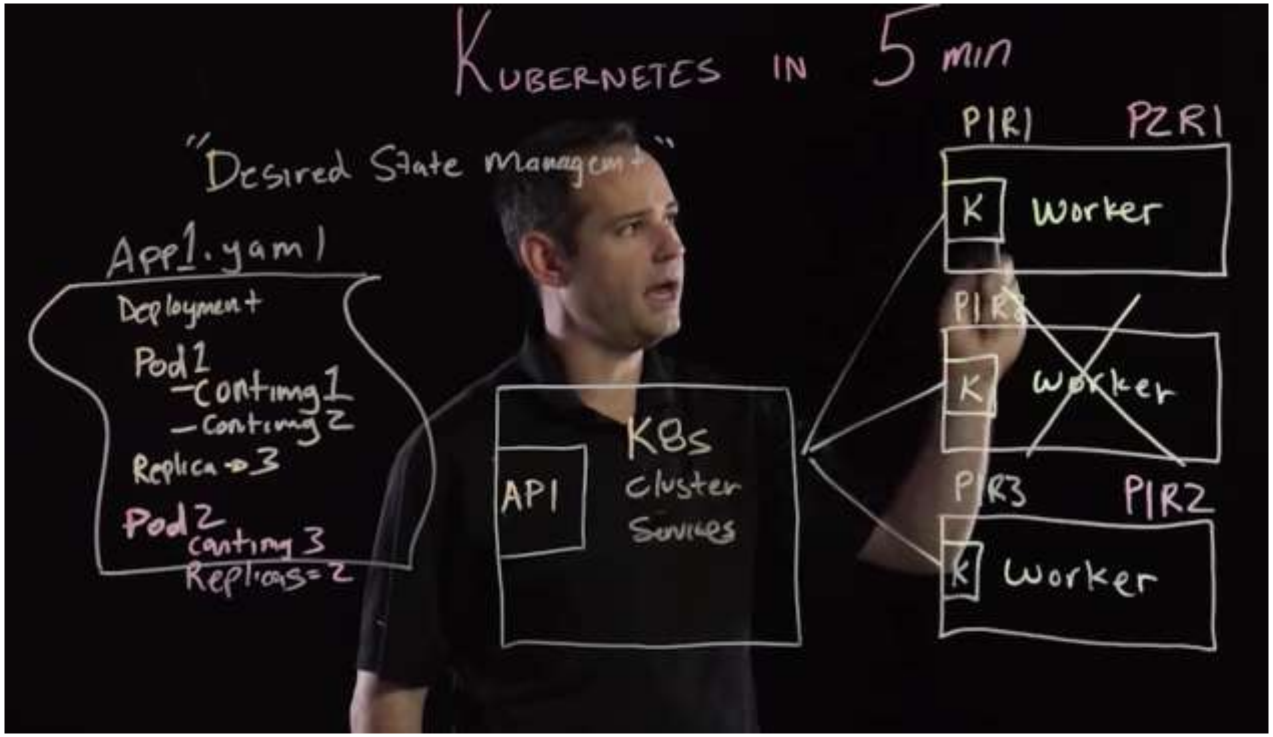
Example 2 of Centralized Structure: Kubernetes

- Advantages:
  - Automate container deployment and replication
  - Organize containers into groups (pods)
  - Load balancing among containers
  - Easy version control and rolling updates



Example 2 of Centralized Structure: Kubernetes

- https://youtu.be/PH-2FfFD2PU

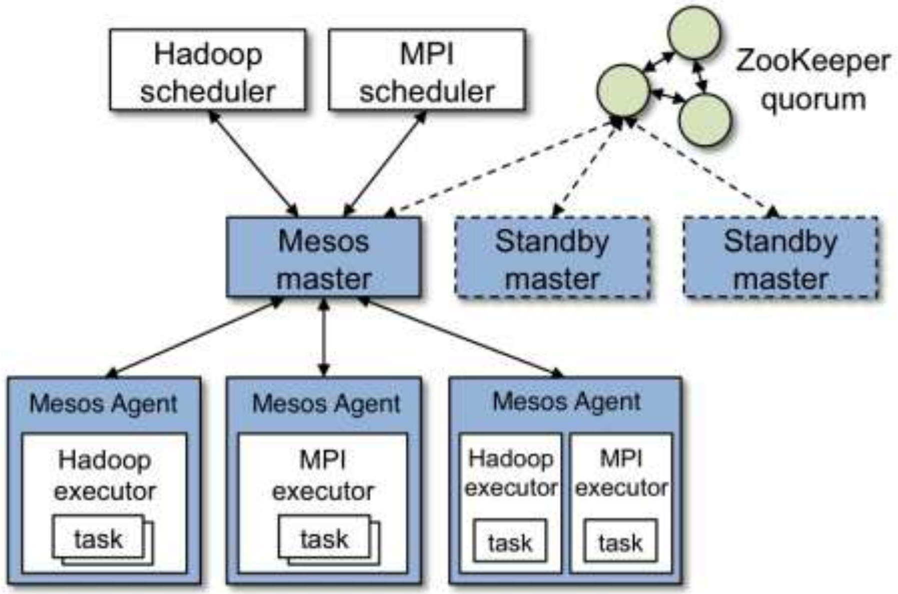


Example 3 of Centralized Structure: Mesos

- Open source distributed resource management framework by Apache

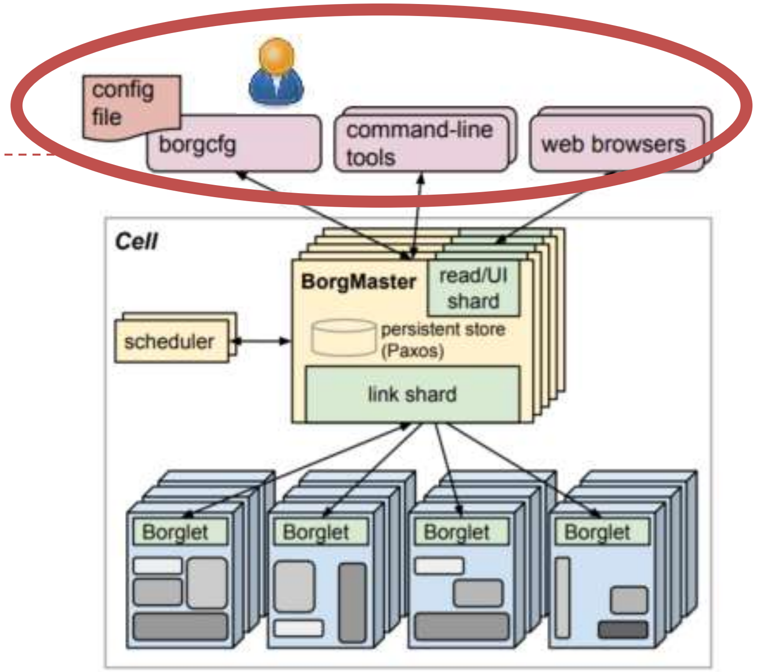
http://mesos.apache.org/

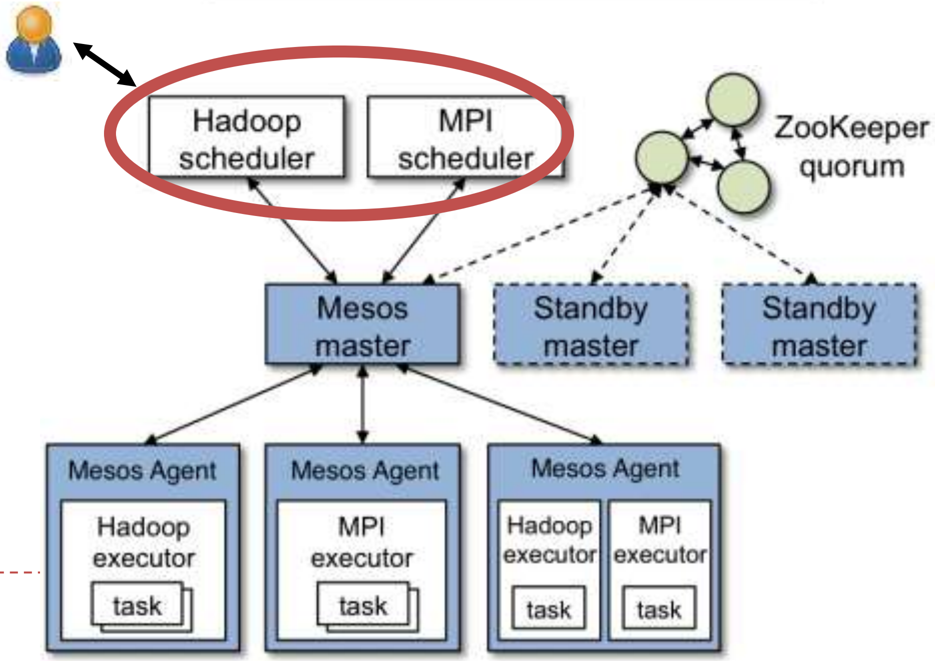
- Mesos is in centralized structure



Example 3 of Centralized Structure: Mesos

- Difference between Borg and Mesos
  - Borg's Master directly interfaces with user applications, which means that users can directly send tasks to Borg's Master
  - Mesos is only responsible for the management and allocation of the underlying resources and does not involve functions such as storage and task scheduling. Therefore, Mesos Master interfaces with frameworks such as Spark, Hadoop, and Marathon. User tasks need to be submitted to these frameworks.

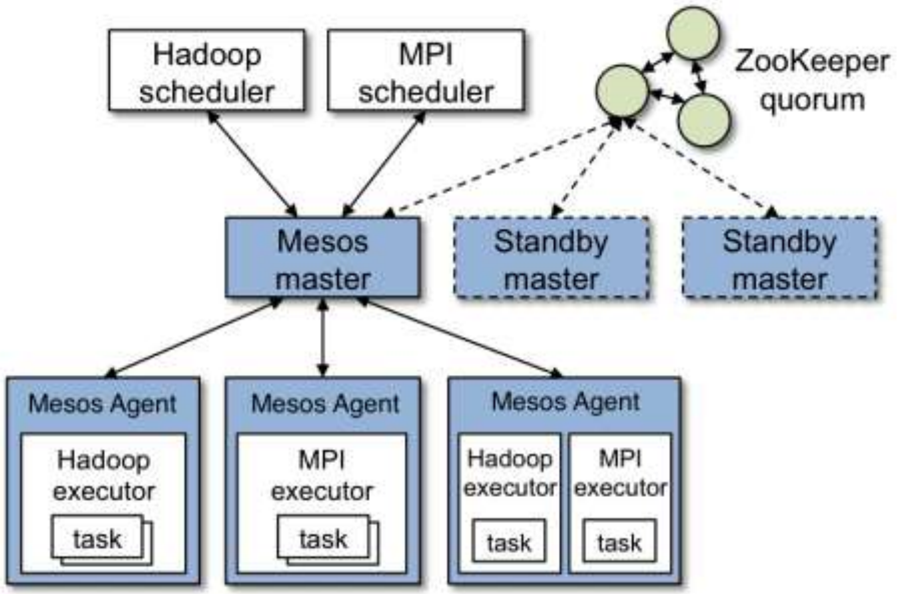




Example 3 of Centralized Structure: Mesos

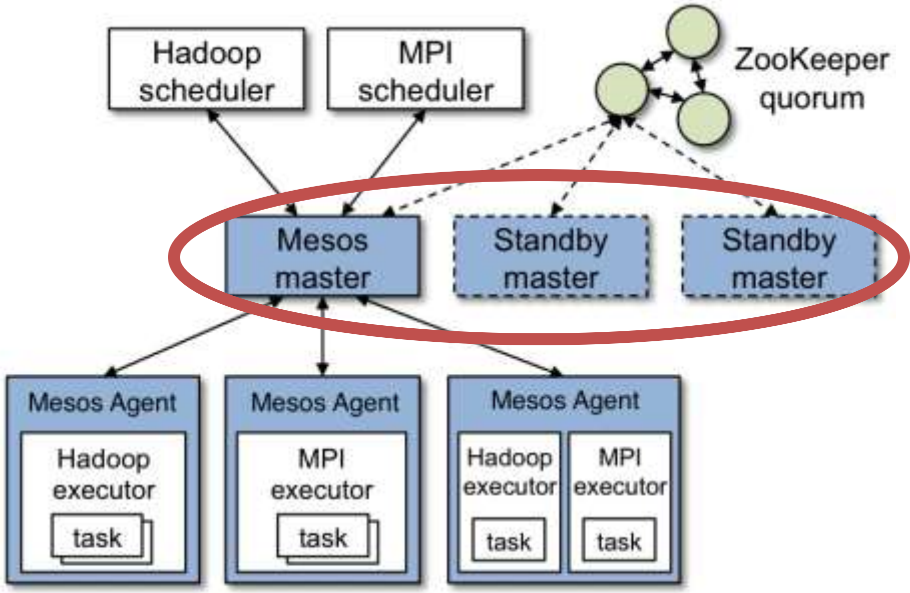
- In Mesos, a cluster includes a Mesos Master and multiple Mesos Agents.

- Mesos Master runs on the central server, and Mesos Agent runs on the node server.



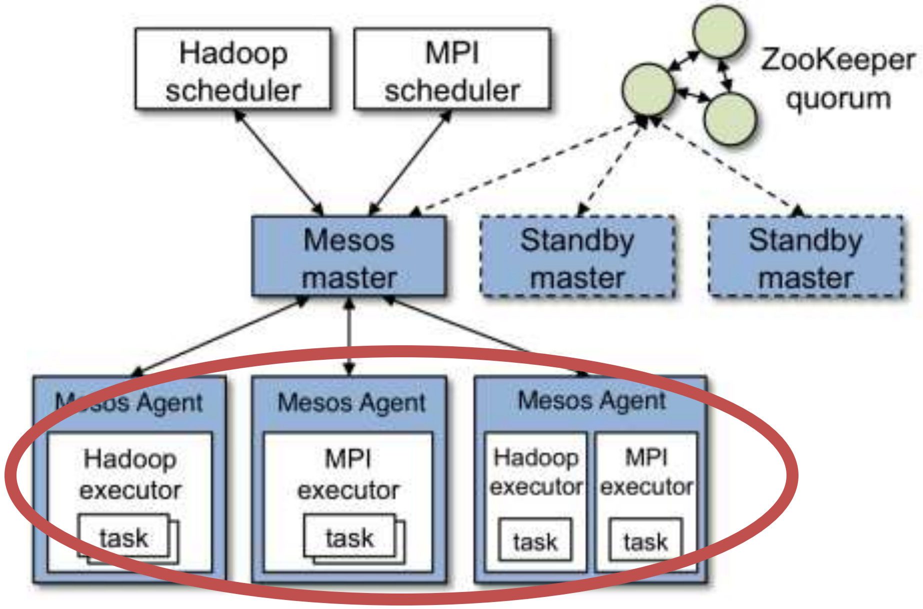
Example 3 of Centralized Structure: Mesos

- The Mesos Master:
  - Responsible for collecting and managing the resources and status of the server where the Agent is located
  - Interface with frameworks such as Spark and Hadoop and notify these frameworks of the resource information of the servers in the cluster for task resource matching and scheduling.
  - Mesos Master usually uses one master and two backup standbys for fault handling and recovery.



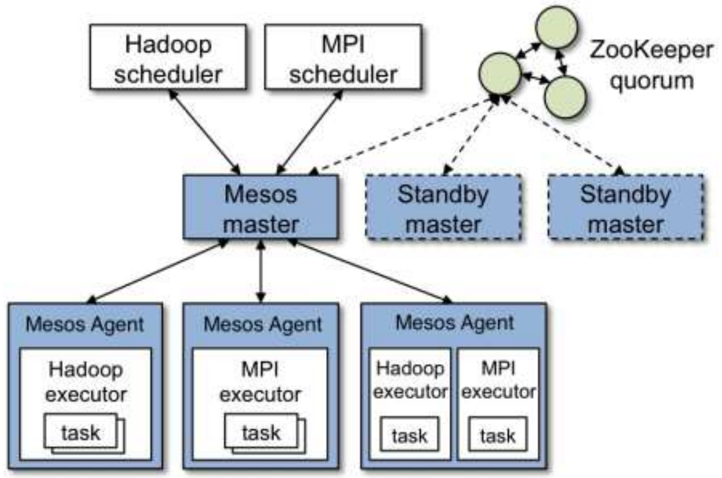
Example 3 of Centralized Structure: Mesos

- The Mesos Agent:
  - Responsible for starting, stopping, and restarting tasks
  - Responsible for collecting information and status of the server's resources (such as CPU, memory, etc.) and reporting it to the Mesos Master.



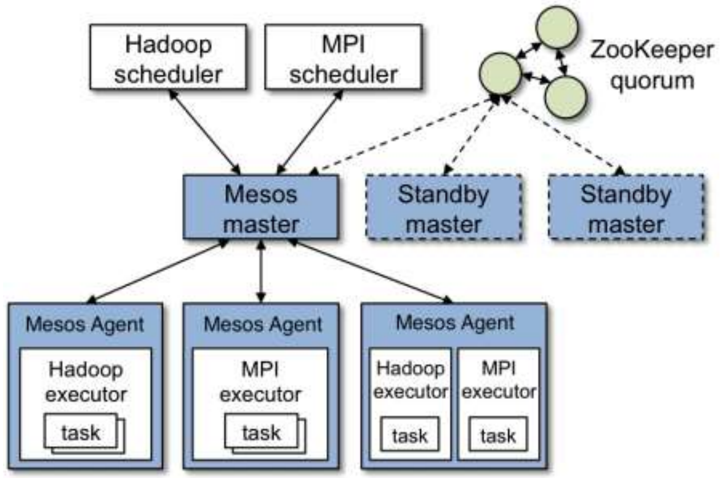
Example 3 of Centralized Structure: Mesos

- Advantages of the Mesos: Effectiveness
  - Mesos abstracts physical resources, allocates resources at the application layer instead of the physical layer, and allocates tasks through containers.
  - The application's scheduler (e.g., Hadoop) knows how to make the most efficient use of resources, therefore, allocating resources at the application layer can benefit the special needs of each application.
  - Allocating tasks through containers is more lightweight and benefits from migration



Example 3 of Centralized Structure: Mesos

- Advantages of the Mesos: Scalability
  - Two-level scheduling architecture: Mesos Master allocates resources and corresponding Framework schedules tasks.
  - Since the Master does not have to know the complex logic behind each type of application and does not have to schedule each task, it can be implemented with very lightweight code and it is easier to expand the cluster size.

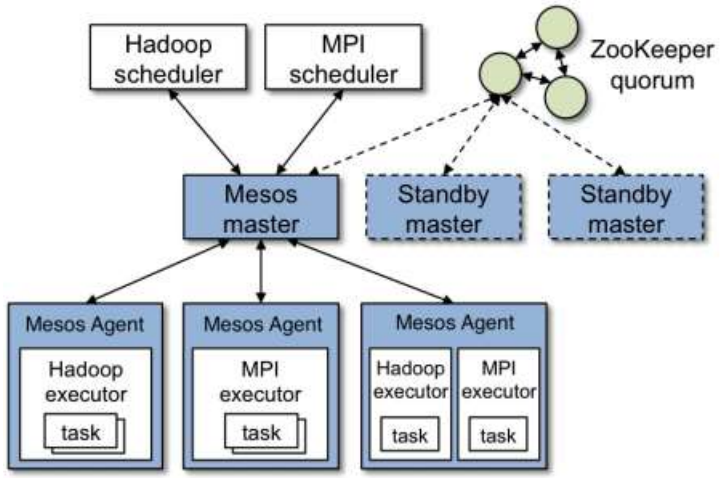


Example 3 of Centralized Structure: Mesos

- Advantages of the Mesos:

  Modular

  - Each time a new framework (e.g., Hadoop, spark, etc.) is added, the Master does not need to add new code, and the Agent module can be reused.
  - Developers can focus on the choice of applications and frameworks. This allows Mesos to support multiple frameworks and adapt to different application scenarios.



Comparison

	Borg	Kubernetes	Mesos
Information	Close source, used by Google inside	Open source	Open source
Design purpose	Allow user focusing on applications instead of resource management	Provide support to large scale of container management	Provide dynamic resource allocation for data center users
Architecture	Master/Slave	Master/Slave	Master/Slave
Scheduling	Single layer scheduling	Single layer scheduling	Two-layer scheduling
Interface to outside	End user task	End user task	Framework (e.g., Hadoop)
Scheduling unit	Job	Pod (group of containers)	Container and job
Operating unit	Task	Service	Task
User representative	Google	Google, Amazon, Microsoft, …	Twitter, Apple, …