ECE 473/573
Cloud Computing and Cloud Native Systems
Lecture 15 Resource Management I

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

October 14, 2024

Outline

Resource Management

Mesos

Reading Assignment

- This lecture: Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center https://static.usenix.org/events/nsdi11/tech/full_papers/Hindman_new.pdf
- Next lecture: Large-scale cluster management at Google with Borg https://storage.googleapis.com/pub-tools-public-publication-data/pdf/43438.pdf

Outline

Motivation

- VMs and containers make it easy to deploy multiple services and applications onto the same server.
  - Ease the complexity to support various OS, library, programming langauges, and application frameworks.
- Overprovisioning by packing more VMs and containers to a single server helps to improve resource utilization and to reduce cost under dynamic loads.
- However, that is not ideal to meet performance demands.
- In a multi-tenancy cloud environment with sufficient amount of computational resources,
  - How to use them more effectively for better performance?
  - How to provide quality of service guarantees to meet business need under large swings in demand?
  - Maintain high resource utilization for cost reduction.

Resource Management

- Resource allocation: where to run an application or service.
- Resource scheduling: when an application or service has access to a piece of resource.
- Resource types: CPU, memory, storage, networking, etc.
- Max-min fairness
  - Everyone receives at least a predefined share of resources.
  - Any excess is distributed among users for better utilization.
- Complexities
  - Location of data may limit allocation choices.
  - Quality of service requirement may limit scheduling choices.
  - Need to fulfill requested resources from an application or service all at once.

An Example of CPU Scheduling

- Assume there are two applications A and B that both need 50% CPU on a server.
- Use round-robin scheduling to assign each application a fixed time slice at a time.
  - A and B is guaranteed to receive 50% CPU time.
  - If any of them is less busy and gives up its current time slice, the other is able to use more.
- What should be the length of the time slice?
  - Longer: less context switching overhead, better cache performance, but higher latency to handle requests – better for computation heavy applications.
  - Shorter: the opposite – better for applications facing end users but reduce actual utilization because of the overhead.
- How could we build a scheduler for these and to meet even more diverse needs?

Outline

Apache Mesos

- An open-source platform to sharing commodity clusters for diverse cluster computing frameworks.
  - These frameworks like Hadoop and MPI simplify programming of the cluster and become popular.
  - Partitioning the cluster to allocate resources for individual frameworks is not ideal because of dynamic loads and the needs to move data around.
  - Instead, sharing the same cluster with multiple frameworks improves resource utilization.
  - Expensive data movement can also be avoided by allowing frameworks to take turn to write and read data.
- Resource offers: distributed two-level scheduling mechanism.
  - Mesos decides amount of resources to offer to each framework.
  - Frameworks decide which resources to accept and which computations to run on them.

Mesos Design Challenges

- Each framework will have different scheduling needs.
  - Depending on many factors like programming model, communication pattern, task dependencies, and data placement.
- Must be able to scale.
  - Tens of thousands of nodes running hundreds of jobs from many applications and frameworks.
  - Each job may consist of thousands of tasks that need resource allocation and scheduling individually.
- Must be fault-tolerant and highly available.
- A centralized scheduler making global scheduling decisions cannot fulfill all these requirements.
  - Difficult to support all scheduling needs, in particular from future frameworks.
  - Waste of efforts as many frameworks have their own scheduler.
  - Overly complicated for scalability and resilience.

Distributed Scheduling via Resource Offer

- Mesos uses a master process to manage slave daemons running on each cluster node.
- Slaves interact with frameworks to run tasks.
- The master generates a resource offer as a list of free resources on multiple slaves.
  - Depending on an organizational policy, e.g. fair sharing, or prioritizing a production framework over a testing one.
- Each framework interfaces with Mesos via two components.
  - A scheduler to register with the master to receive resource offers, and to decide which offered resource to use.
  - An executor to launch tasks on slaves base on the decision of the scheduler.
- Frameworks don't need to specify their resource requirements
  - Instead, frameworks reject offers not satisfying their need and wait for Mesos to provide ones that do.
  - This helps to keep Mesos simple and scalable.

Filters

- However, it is not efficient for frameworks to simply wait for offers that satisfy their needs.
- Filters allow frameworks to specify what will always be rejected as Boolean prdicates.
  - E.g. a list of nodes that a framework wants to avoid.
- Filters capture easy-to-represent constraints for performance optimization, while frameworks still have full control via rejections.
- Empirical studies with many fine-grained tasks show resource offers to work very well even without filters.

Scalability, Fault Tolerance, and Isolation

- Mechanisms for better scalability.
  - Filters to reduce communications.
  - Encourage frameworks to respond to offers quickly by counting resources offered towards their usages.
  - Cancel offers if frameworks fail to respond within a deadline.
- Handling faults in Mesos
  - Stateless master: states holding by the master can be reconstructed from that of slaves and frameworks.
  - Node failures and task exceptions are reported to frameworks' schedulers for them to take further actions.
  - Frameworks can register multiple schedulers with Mesos in case their schedulers fail.
- Isolation between frameworks, i.e. for tasks from different frameworks running on the same slave, is provided via containers.

Summary

- Resource management is an essential part of cloud computing to make better use of tremendous amount of computational resources.
- Mesos provides a two-level scheduling mechanism to support cluster computing frameworks.