CERES: Container-Based Elastic Resource Management System for Mixed Workloads

Jinyu Yu, Dan Feng, Wei Tong, Pengze Lv and Yufei Xiong

Huazhong University of Science and Technology
Outline

• Background and Motivation

• CERES

• Evaluation

• Conclusion
Workload Deployment Method

• Widespread use of emerging technologies
  - Increased difficulty in resource management

• Dedicated Cluster or Resource Reservation
  - Low resource utilization
  - High operation and maintenance costs

• Mixed Workload Deployment (MWD)
  - Deploy multiple workloads in one cluster
  - Widespread use
    ○ Alibaba, Bing, Google.
      □ Latency-Sensitive Services (LSSs) & Batch jobs
  - Workload characteristics
    ○ Resource requirements
    ○ QoS
Details of Workload Processing

• **Job**
  - The logical entity of a workload
  - A job is handled by multiple tasks

• **Task**
  - The basic unit that does the actual work
  - The lifecycle
    o Task Scheduling Latency
      □ Swollen task
        △ Surplus resources
      □ Few idle/allocable resources
        △ Newly coming tasks queuing for resources
    o Task Running Time
      □ Task dependency
      □ Straggler task
        △ Inter-task interferences or insufficient resources
        △ Block the progress of tasks that depend on it

• **How to guarantee the QoS of LSSs in MWD Cluster?**

A. The lifecycle of a task

B. Task dependencies
Related Work

• Solutions to guarantee the task scheduling latency
  - Preempt the resources of batch jobs
    ○ PerfIso, BIG-C
  - Preempt resources without considering surplus resources

• Solutions to eliminate straggler tasks
  - Task Replicas
  - Loss of task progress, increase in resource consumption
Goals and Challenges

**Goals**

- Enough allocable resources to avoid long scheduling latency
- Minimize the performance impact of straggler tasks
- Minimize the performance loss to batch jobs

**Challenges**

- Accurate identification of swollen tasks and on-demand resource reclamation
- Accurately identify and eliminate straggler tasks
- Reduce resource preemptions

Resource Management Mechanism for MWD
CERES

- **Application Status Store (ASS)**
  - Task Status Collector & Time-series Database
  - collecting, processing, and storing task status data
- **Adaptive Policy Builder (APB)**
  - Task Filter
    - screen out swollen tasks and straggler tasks;
  - Adaptive Policy Generator
    - make adaptive resource decisions
- **Node Task Manager (NTM)**
  - Adaptive Policy Executor
    - execute task resource adjustments;
  - Task Status Monitor
    - obtain task status data on the node;
Task Filter

• **Swollen tasks** from batch jobs
  – Get task resource limits and the maximum used resources
  – Compute the actual maximum resource utilization
  – Determine whether the task is swollen or not

• **Straggler task** from latency-sensitive services
  – Get the monitoring data of the last three monitoring time points
  – Compute the current and previous processing speed
  – Estimate the task completion time
  – Determine whether the task is a straggler or not
Adaptive Policy Generator

- Get cluster idle resources;
- Count the total resource requirements of new latency sensitive tasks and straggler tasks;
- Idle resources cannot meet task resource requirements
  - Reclaim resources from swollen tasks;
  - Preempt resources from other batch tasks;
- When there are enough allocable resources
  - If straggler tasks exist, expand resources for them;
  - If there are no reclaiming or preempting operations, restore resources for preempted tasks.
Node Task Manager

• **Adaptive Policy Executor**
  - Receive adaptive policies;
  - Perform the policies on tasks
    ○ Call the Docker Engine API based on Cgroups;
    ○ Achieve container migration with CRIU;

• **Task Status Monitor**
  - Obtain task status information
    ○ Resource usage, processing process, running time, etc.
  - Report the monitoring data to ASS;
Evaluation Setup

• Cluster
  - Composed of 26 servers
    ○ One manager node, 25 worker nodes;
    ○ 32 CPU cores, 128GB memory, 12Gbps Ethernet

• Metrics
  - Task Scheduling Latency (TSL)
  - Task Running Time (TRT)
  - Task Completion Time (TCT)
  - Job Completion Time (JCT)
  - Cluster Resource Utilization (CRU)

• Baselines
  - CS-DP: Capacity Scheduler with resource preemption disabled
  - CS-EP: Capacity Scheduler with resource preemption enabled
  - BIG-C: A container-based preemption solution

• Workloads
  - Latency-sensitive services
    ○ Spark-SQL is used to generate queries as latency-sensitive services (LSSs)
  - Batch jobs
    ○ Select batch jobs from HiBench and BigDataBench, such as wordcount, terasort;
  - Batch jobs account for 10% of the mixed workloads.
Performance of LSSs

- Average task scheduling latency
  - Compared with
    - CS-DP: decreased by 50.87%;
    - CS-EP: decreased by 32.99%;
    - BIG-C: decreased by 16.90%;

- 99th percentile task scheduling latency
  - Compared with
    - BIG-C: decreased by 30.42%;

- 95th percentile task running time
  - Compared with
    - CS-DP: decreased by 36.23%;
    - CS-EP: decreased by 28.04%;
    - BIG-C: decreased by 16.41%;

- 99th percentile task running time
  - Compared with
    - BIG-C: decreased by 18.91%;

- Average task completion time
  - Compared with
    - CS-DP: decreased by 22.42%
    - CS-EP: decreased by 18.00%
    - BIG-C: decreased by 14.07%

- 99th percentile task completion time
  - Compared with
    - BIG-C: decreased by 20.77%
Completion Time of Batch Jobs

- Compared with
  - CS-DP: at most increased by 15.46%
  - CS-EP: at most reduced by 26.06%
  - BIG-C: at most reduced by 17.7%
Resource Utilization of the Cluster

- Cluster resource utilization reached 53.73%;
- Average resource utilization
  - Compared with BIG-C, promoted by 27.06%;
Conclusion

• Problems of MWD
  − Resource contentions and inter-task Interferences lead to severe QoS losses to LSSs
  − Existing solutions guarantee the QoS of LSSs by preempting resources from batch tasks
    ○ Performance loss to batch jobs
  − Tasks do not fully utilize the allocated resources

• We propose CERES to guarantee the QoS with surplus resources
  − Accurate task filters
  − Adaptive resource adjustment policies

• CERES can guarantee the QoS of LSSs and reduce the performance penalty for batch jobs.
Thanks! Q&A

Email: yujinyu@hust.edu.cn