Moving MapReduce into the Cloud: Flexibility, Efficiency, and Elasticity

PI Xiaobo Zhou* and CoPI Jia Rao◆

NSF CNS-1422119

* The University of Colorado, Colorado Springs ◆The University of Texas, Arlington

Introduction

Moving MapReduce into the cloud is believed to benefit from rapid deployment, high availability, on-demand elasticity and secure multi-tenancy.

However, simple migration does not ensure that the flexibility, efficiency and elasticity in the cloud could be fully exploited.

The sematic gap between the MapReduce runtime and the virtualization layer, and the lack of MapReduce-aware cloud management impede the wide adoption of *Big Data Cloud*.

- We propose para-virtualized MapReduce (para-MR), an enhancement of MapReduce to actively adapt job to the cloud dynamics, including interference and hardware heterogeneity.
- We propose MapReduce cloud (MR-cloud), a collection of cloud optimizations for MapReduce workloads to truly realize the flexibility and elasticity of virtualization.

Background and Motivation

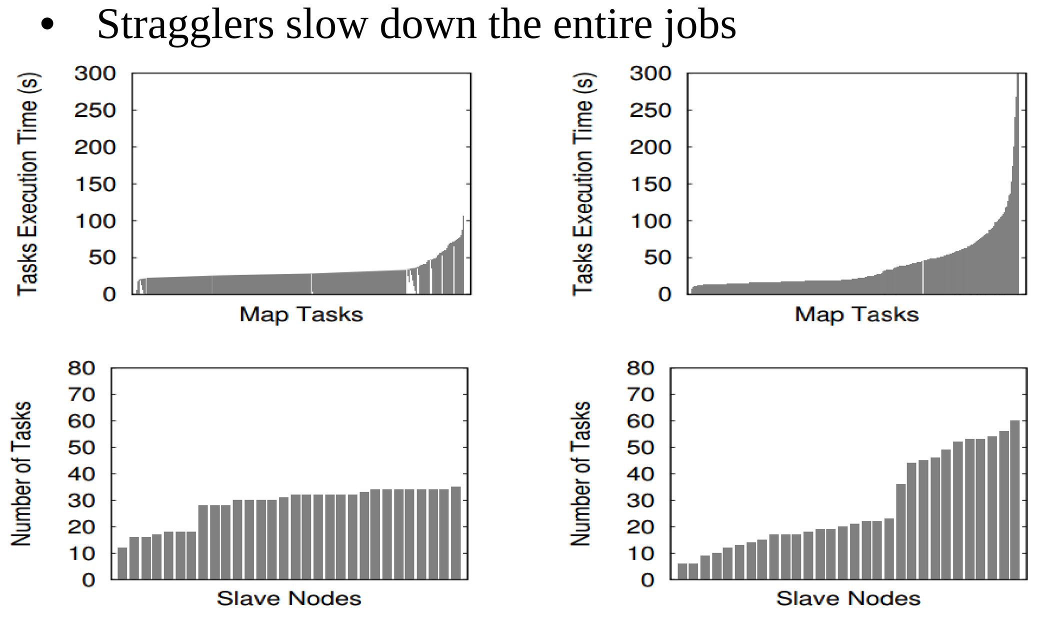
Issues in MapReduce: low cluster utilization, suboptimal scalability and poor multi-tenant support.

The worker based resource allocation in MapReduce makes it hard to fully utilize cluster resource .

Skew caused by uneven data distribution or non-uniform data processing cost creates stragglers.

Challenges in fully unlocking cloud potential.

- Hidden hardware capabilities and performance discrepancies
- Stragglers slow down the entire jobs



Virtualization causes task performance heterogeneity

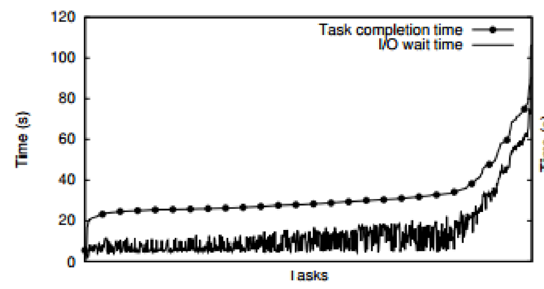
Results: We developed techniques that successfully detect stragglers, resize node capacity and achieve load balancing in virtual cluster. As a result, both job performance and cluster utilization are improved

Future work: We are aiming to apply light-weight virtualization in Big Data analytics.

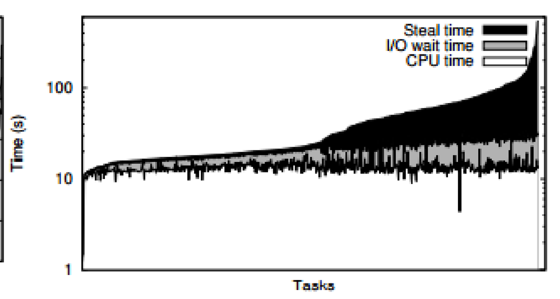
Approaches: FlexSlot

FlexSlot: Moving Hadoop into the Cloud with Flexible Slot Management (SC 2014)

Preliminary Study:



Task runtime vs. iowattime



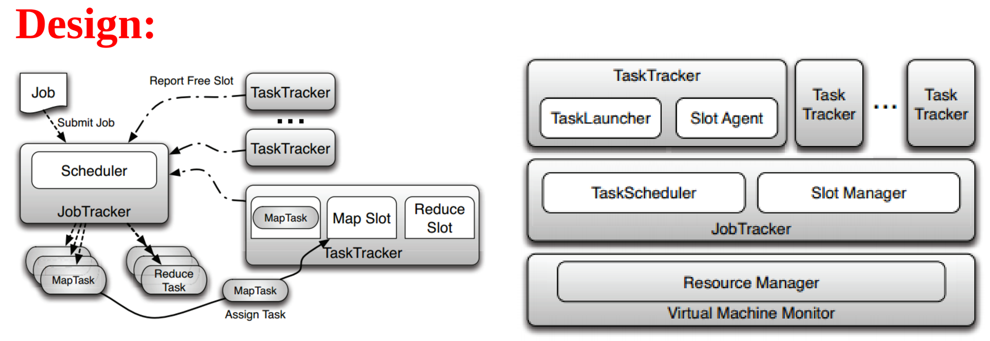
Task runtime vs. cputime

Category the reasons cause stragglers:

•Disk I/O Bottleneck due to Data Skew

•CPU Starvation due to Inaccurate Demand Estimation

Design:



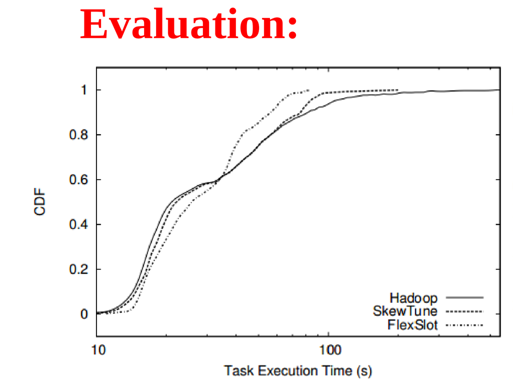
MapReduce design and FlexSlot design based on MapReduce

1.Identifying stragglers: continuously monitors two task-specific metrics during task execution: progress rate and input processing speed

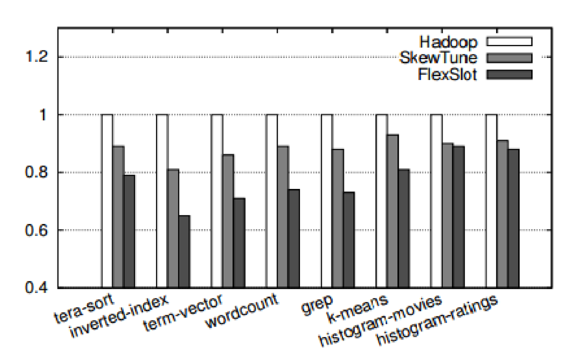
2.Proactively changing the size of slots: If a straggler's performance is bottlenecked by I/O operations, it proactively terminates the straggler and restarts it with a larger slot size.

3.Adaptively adjusting the number of slots: bridges the semantic gap between Hadoop tasks and the demand-based resource allocation by adaptively changing the number of slots on Hadoop nodes.

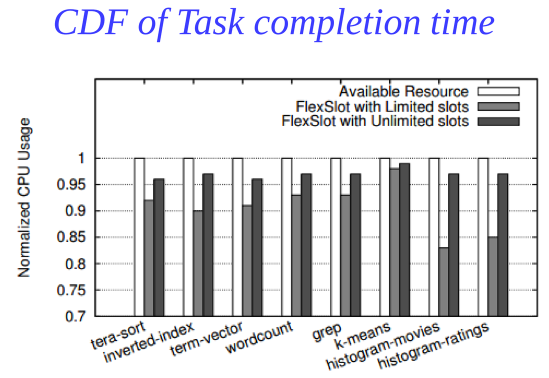
Evaluation:



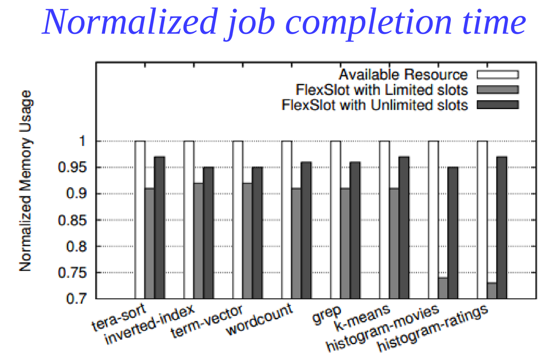
CDF of Task completion time



Normalized job completion time



Normalized CPU utilization



Normalized memory utilization

Conclusion:

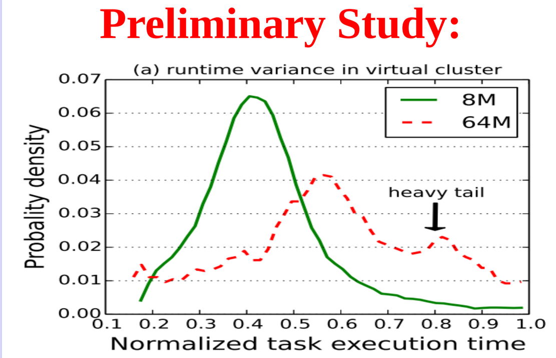
1. Significant reduction in job completion time.

2.Significant Improvement in cluster resource utilization.

Approaches: FlexMap

Addressing Performance Heterogeneity in MapReduce Clusters with Elastic Tasks (IPDPS 2017)

Preliminary Study:



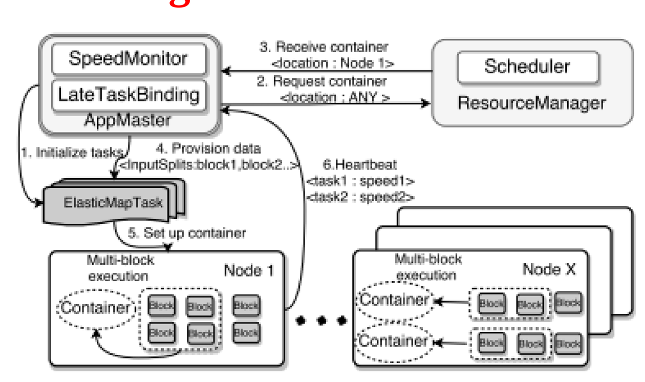
Task runtime probability

Job runtime & efficiency vs. bock size

•Load balancing should be performed at fine granularity to mitigate performance heterogeneity but tasks should be run at coarse granularity to avoid execution overhead.

•The optimal task size depends on the interplay between the execution overhead, such as container and JVM startup time, the computation needed by a particular job, and the degree of performance heterogeneity.

Design:



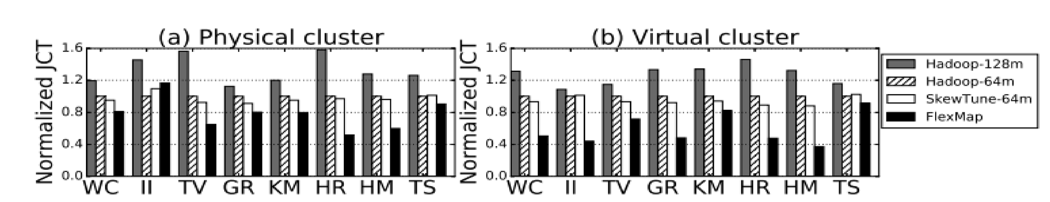
- *Multi-block execution*
- *Late task binding*
- *Monitoring node speed*
- *Task sizing*

Heterogeneous-aware task sizing:

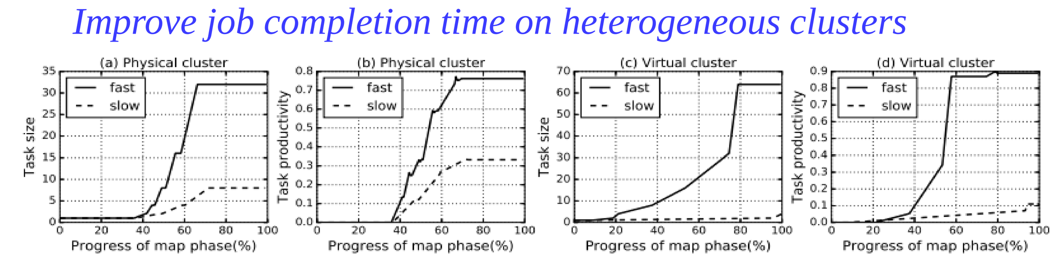
•***Vertical scaling:*** fast scaling to jump small block size causing inefficiency.

•***Horizontal scaling:*** adjust map size horizontally across machines depends on their relative speed.

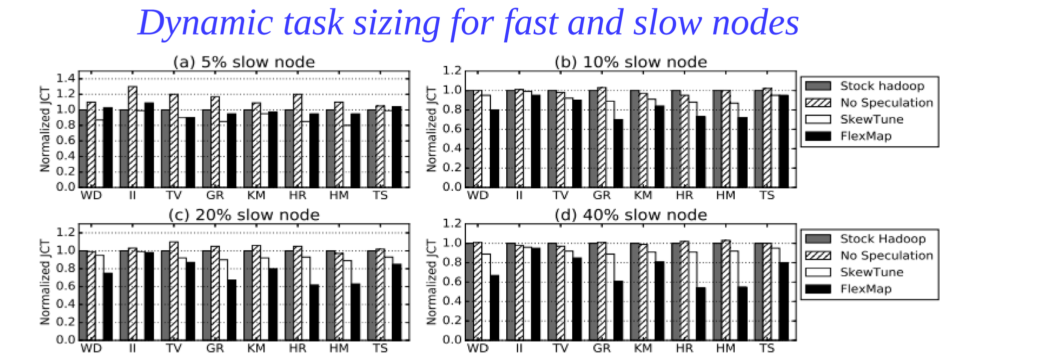
Evaluation:



Improve job completion time on heterogeneous clusters



Dynamic task sizing for fast and slow nodes



Large scale results

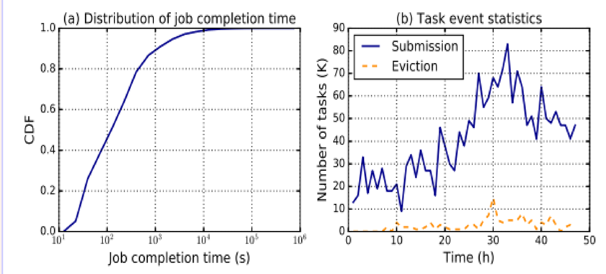
Conclusion:

Significantly improve job performance in heterogeneous cluster.

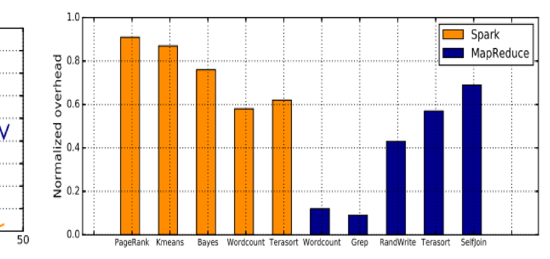
Approaches: BIG-C

Preemptive, Low Latency Datacenter Scheduling via Lightweight Virtualization (USENIX ATC 2017)

Preliminary Study:



Data center trace analysis



Overhead of default killing preemption

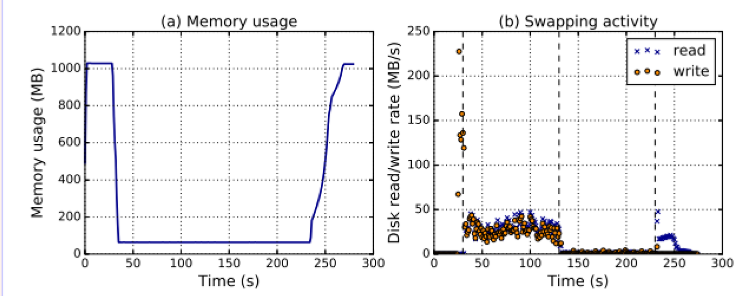
•Data centers are dominated by short jobs and eviction are frequent to attain scheduling policy.

•Default killing based preemption causes significant overhead, especially long running workloads.

Design:

Task suspension: deprive CPU resource and save task context onto disk while keeping task heartbeat(e.g., set CPU usage to 1% and memory usage to 64MB)

Task resumption: re-activating the container by restoring its deprived resources.

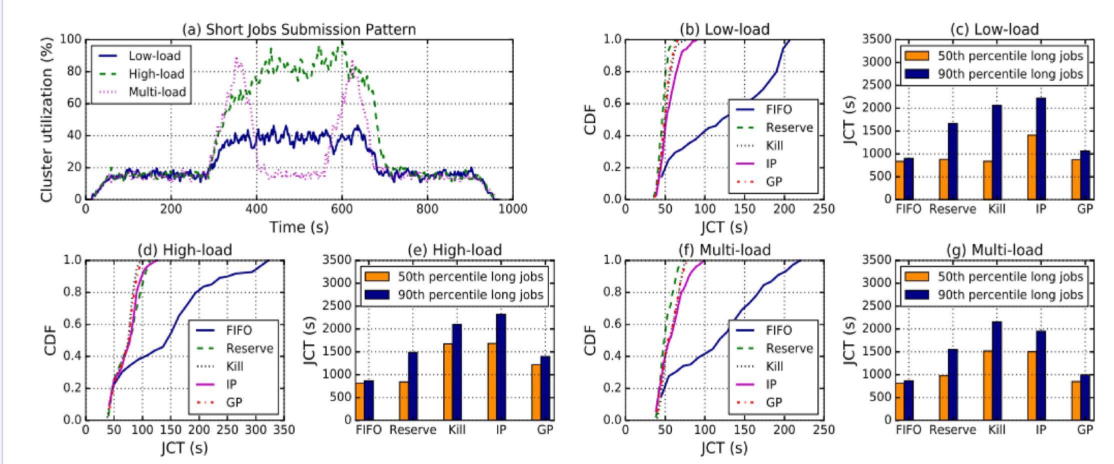


- *Throttle CPU*
- *Shrink memory*
- *Resume memory*
- *Resume CPU*

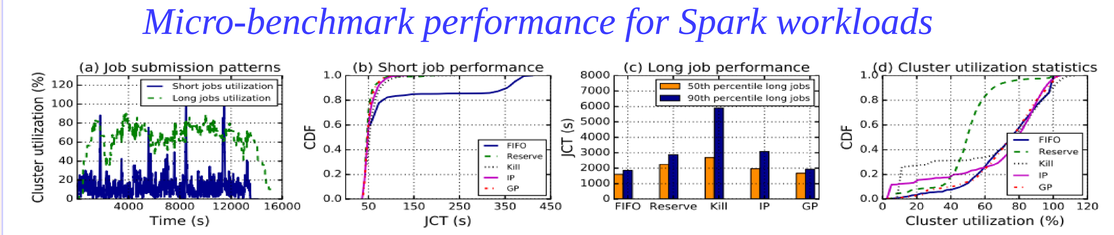
Immediate task preemption: immediately deprive both CPU and memory resource for a task.

Graceful task preemption: Reclaim a task's resource at pre-defined step based on preemptive fair share scheduling.

Evaluation:



Micro-benchmark performance for Spark workloads



Google trace results

Conclusion:

1. Reduce preemption overhead to an acceptable level..

2. Significant Improvement both in cluster resource utilization and job performance.