**Introduction**
Moving MapReduce into the cloud is believed to benefit from rapid deployment, high availability, on-demand elasticity and secure multi-tenant support. However, simple migration does not ensure that the flexibility, efficiency and elasticity in the cloud could be fully exploited.

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

The semantic 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.

- Disk I/O Bottleneck due to Data Skew
- CPU Starvation due to Inaccurate Demand Estimation

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

**Preliminary Study:**

- Task runtime vs. cputime
- Task runtime vs. iowattime
- 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:**
![Graphs and charts showing task runtime, I/O, CPU utilization, etc.]

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

**Design:**
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:**
![Graphs and charts showing job completion time, normalized job completion time, CPU utilization, memory utilization, etc.]

**Conclusion:**
Significantly improve job performance in heterogeneous clusters.

**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

**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.

**Evaluation:**
![Graphs and charts showing preemptive, low latency scheduling results, CPU usage, memory usage, etc.]

**Conclusion:**
1. Reduce preemption overhead to an acceptable level.
2. Significant Improvement both in cluster resource utilization and job performance.