SCEEM: A Stealing and Chunking Execution Engine for Mesos

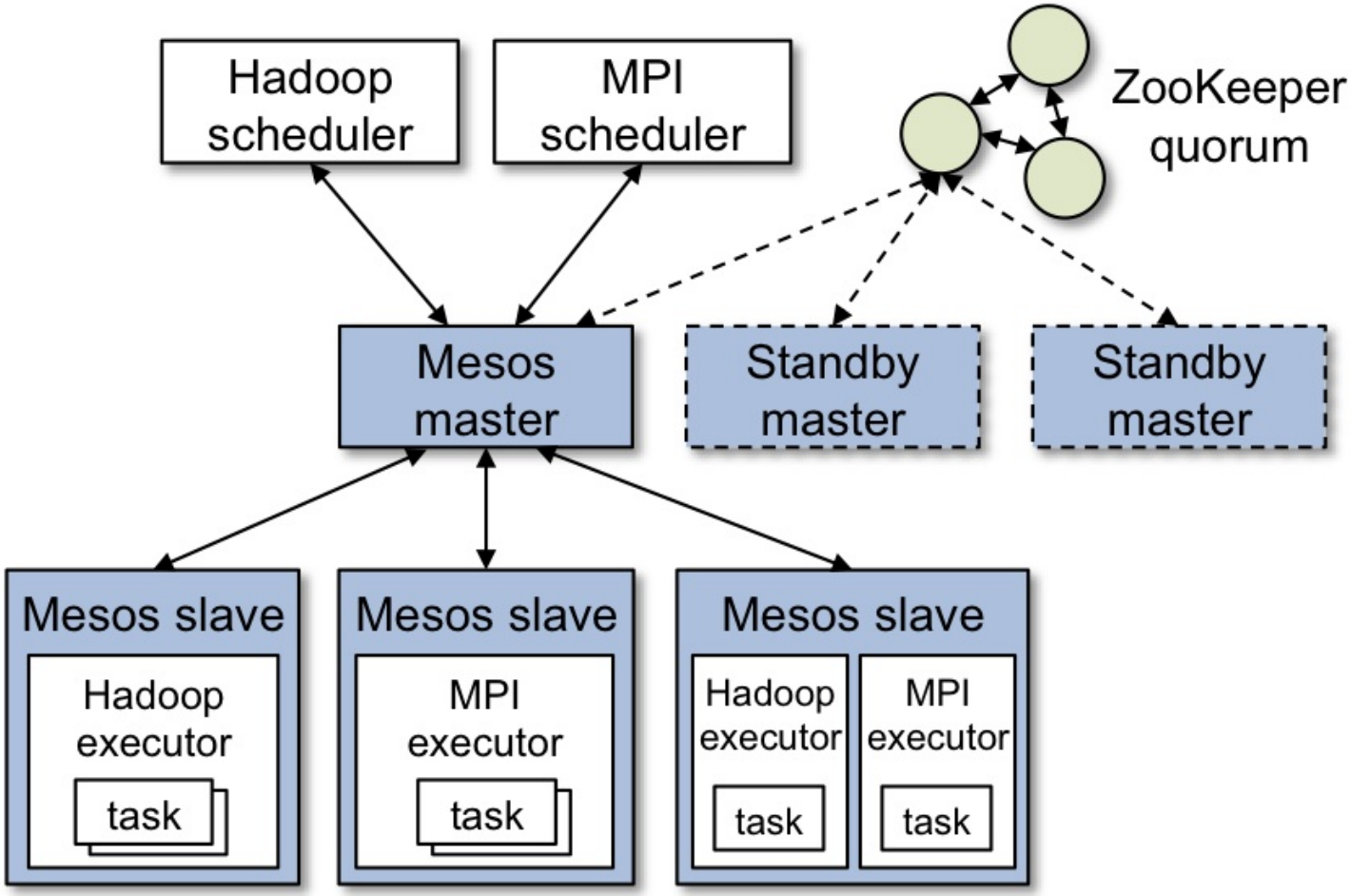
Alex Degtiar and Apoorva Sachdev

Problem

- Latency:
  - Distributed task-based execution engines suffer a latency hit when submitting tasks for execution.
  - This effect is especially pronounced when the task time length is small.
  - **Task chunking** can minimize this cost by submitting tasks fewer times, but each time in bulk.
- Load balancing:
  - Unevenly distributed tasks leave idle nodes.
  - **Task stealing** is an effective means of dynamically balancing groups of pending tasks between nodes.

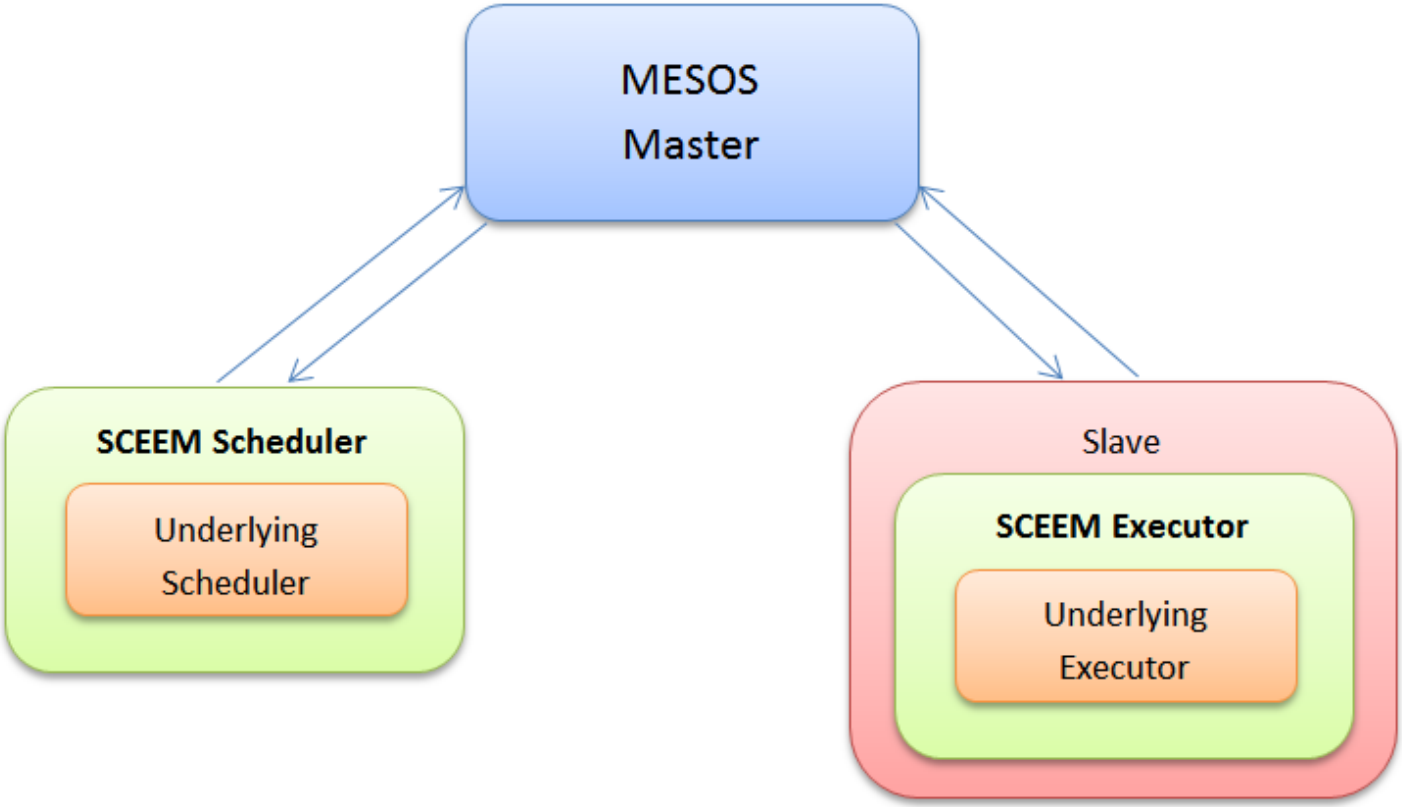
Mesos

- A cluster manager providing resource sharing among distributed applications.
- Defines a platform on which to build distributed frameworks.
- A framework implements a centralized task scheduler + a (distributed) task executor.


Hadoop scheduler
MPI scheduler
ZooKeeper quorum
Mesos master
Standby master
Standby master
Mesos slave
Hadoop executor
task
Mesos slave
MPI executor
task
Mesos slave
Hadoop executor
task
MPI executor
task

Where our module fits in

- It forms a wrapper around the underlying scheduler and executor modules implemented by the framework.
- Like Mesos, it is framework-agnostic.
- It provides abstractions to the framework to group tasks and submit them as task chunks.
- It enhances the Mesos framework by implementing task stealing on the existent driver.
- It is an abstract scheduler that the framework can use to take advantage of task stealing without implementing it.

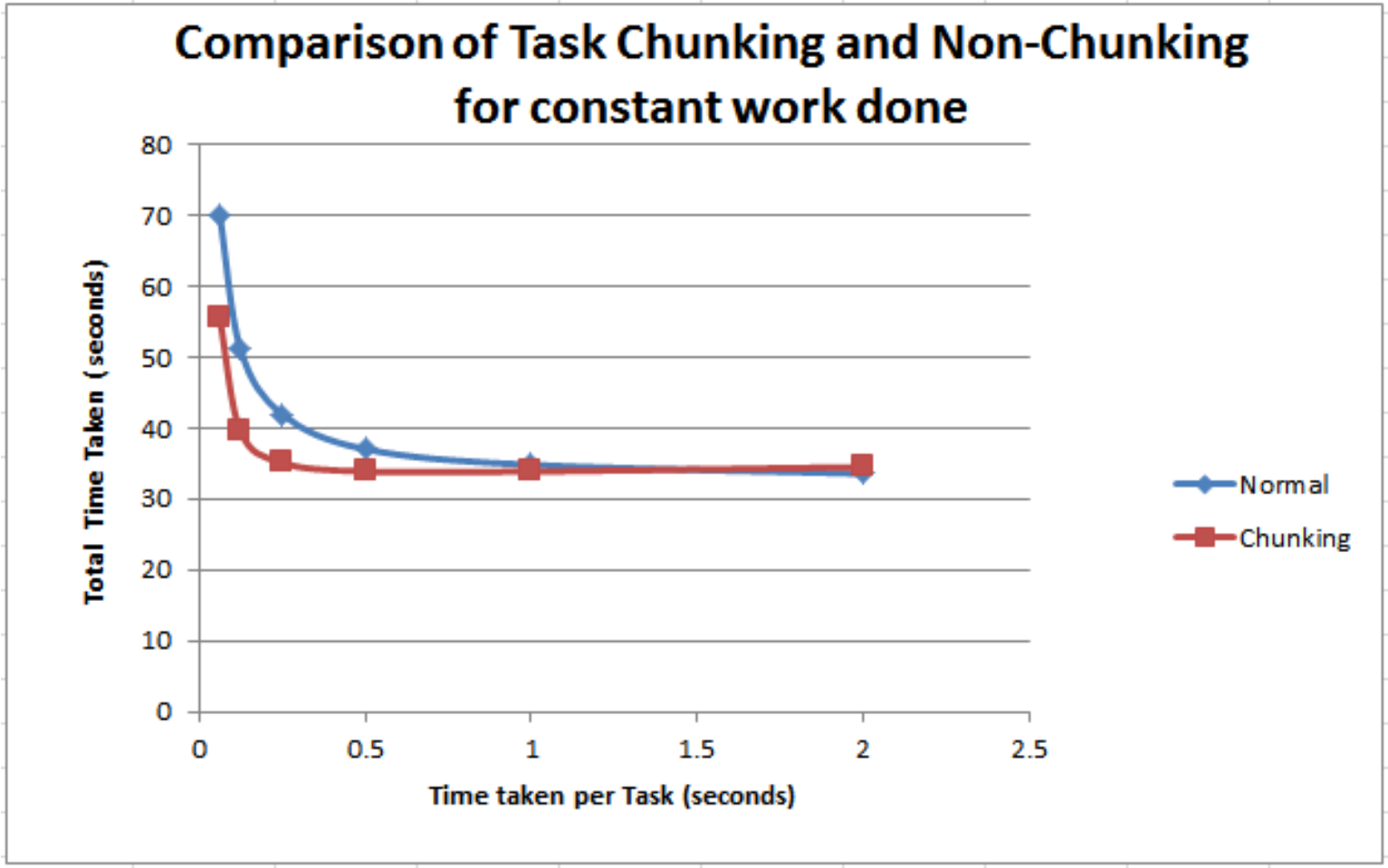

MESOS
Master
SCEEM Scheduler
Underlying
Scheduler
Slave
SCEEM Executor
Underlying
Executor

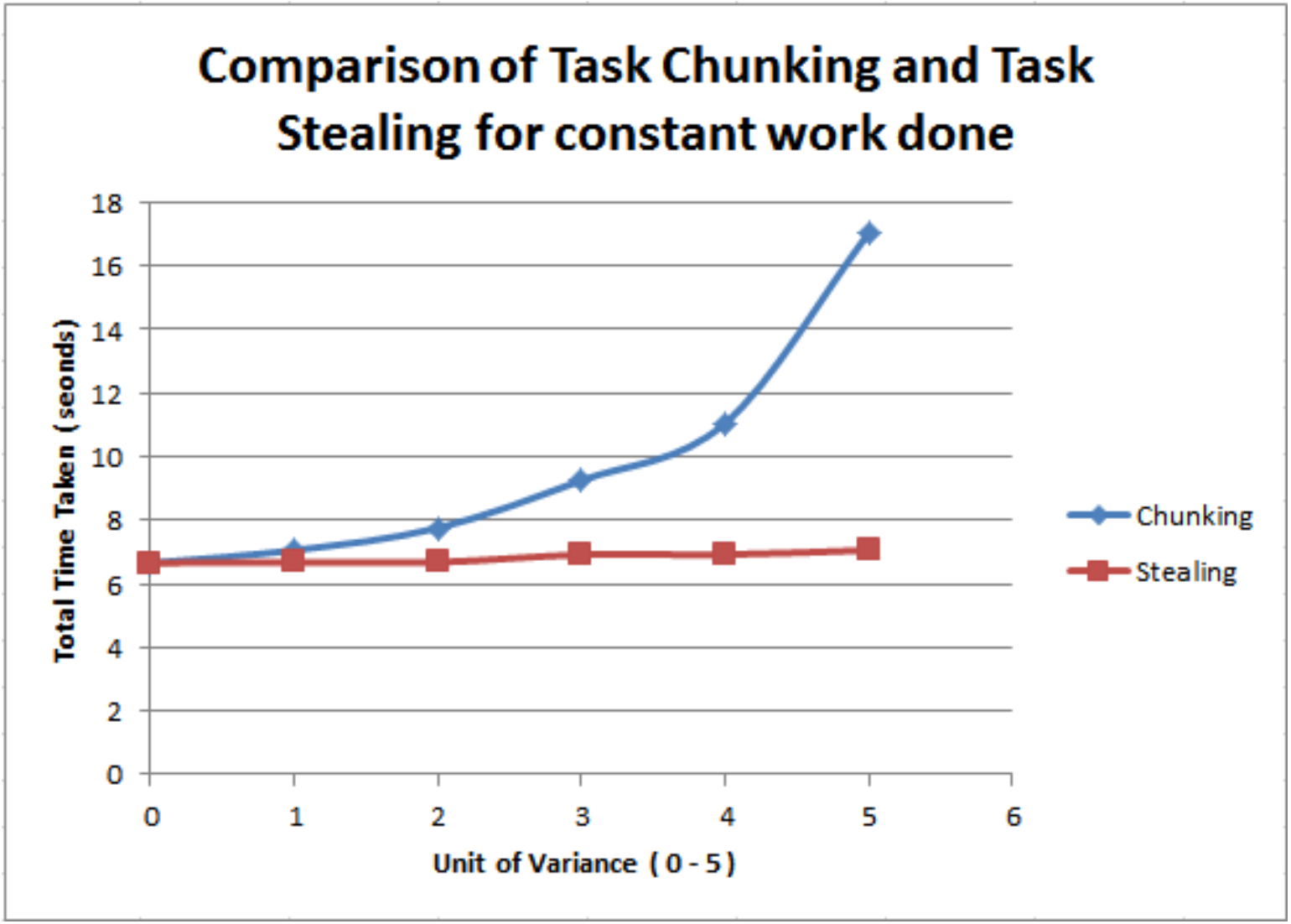
Design and Implementation

- Task Chunking:
  - A chunk is a DAG/queue of pending tasks.
  - Executor-side event-driven loop.
  - A table of chunks keeps internal queue state.
  - Scheduler-side hook for updates and control.
- Task Stealing:
  - Scheduler-side copy of pending task state.
  - Scheduler hook for resource offers, which selects subtasks to steal and kills/re-launches them.

Task Stealing Algorithms

- Greedy algorithm: select highest-priority task and steal half its sub-tasks. Priorities:
  - Largest group of subtasks.
  - Longest-waiting idle task.
- Task chunks are split to maintain locality and dependency grouping. Pairing a stolen split with resources on the same node may also help in this respect.
- Resources matched with tasks in order of smallest to largest to avoid excessive splitting of resources.
- Task stealing attempts are prioritized over offers to underlying framework (configurable). Unused offers are passed along.


Comparison of Task Chunking and Non-Chunking for constant work done
Total Time Taken (seconds)
80
70
60
50
40
30
20
10
0
0
0.5
1
1.5
2
2.5
Time taken per Task (seconds)
Normal
Chunking


Comparison of Task Chunking and Task Stealing for constant work done
Total Time Taken (seonds)
18
16
14
12
10
8
6
4
2
0
0
1
2
3
4
5
6
Unit of Variance (0 - 5)
Chunking
Stealing

Results

1. Task chunking reduces the latency overhead and gives significantly better performance for small tasks.
2. Task stealing gives significantly better performance when task chunk size variability is high.
3. Overhead of Python implementation causes a measurable difference vs. the native implementation when ignoring latency.

Conclusion and Future Work

We have shown that employing a chunking and stealing scheduler can significantly improve the aggregate performance of executing many small tasks in system like Mesos.

For future work, we can improve the task stealing algorithm, implement SCEEM natively, implement a distributed work stealing model instead of a centralized scheduler, and add full DAG support.