TECHILA

TECHILA TECHNOLOGIES
SCHEDULER BENCHMARK STUDY

25 FEBRUARY 2022

Executive summary

Techila Distributed Computing Engine (TDCE) is a highly efficient middleware system providing excellent APIs and a world class scheduler that ensures maximal throughput in even the most demanding high-performance computing (HPC) situations. This document draws on the performance tests conducted and published by MIT, presenting TDCE performance statistics of similar test setup for comparison purposes. As can be seen from the results, TDCE outperformed all competing solutions and ensured that system utilization remained close to 100% in all test cases, significantly reducing both runtime and capacity costs.

Introduction

In *Scalable System Scheduling for HPC and Big Data* by Reuther et al. of Massachusetts Institute of Technology (MIT), several popular HPC schedulers - Slurm, Son of Grid Engine, Mesos, and Hadoop YARN - were benchmarked using workloads consisting of sleep tasks. These benchmarks were used to measure efficiency and system utilization to determine how well the schedulers perform when processing workloads of different durations.

The study also stated the following:

> *High performance data analytics (HPDA) jobs have characteristics of both HPC and big data types of jobs. This paper explores the features that are necessary to execute high performance data analytics jobs because we need the best of both worlds to best accommodate such jobs.*

This statement continues to be accurate, meaning the need for highly efficient scheduling operations is paramount when the goal is to deploy a highly responsive and efficient HPC environment. TDCE includes a proprietary state-of-the-art scheduler, capable of meeting all these requirements.

TDCE also includes extensive APIs for several popular programming languages, such as Python, R, MATLAB, Java and C#/.NET. These programming language APIs make integrating TDCE with HPDA and other demanding applications easy and straightforward.

Inspired by the reported benchmark results in the original study, Techila Technologies ran a similar set of performance tests to compare the TDCE scheduler performance with the previously tested schedulers.

Brief description of the methodology

The duration of the sleep tasks varied in each test, ranging from 1 second up to 60 seconds. The number of sleep tasks was changed respectively, keeping the theoretical ideal runtime at 240 seconds using 1408 CPU cores. Two examples below for clarification:

- *Example #1. If task time is set to **1** second, the number of tasks will be **337920** (1 * 337920 / 1408 = 240 second runtime)*
- *Example #2. If task time is set to **5** seconds, the number of tasks will be **67584** (5 * 67584 / 1408 = 240 second runtime)*

Thus, anything exceeding 240 seconds may be considered a result of underutilization of the computing environment.

Comparing earlier study results and new TDCE performance data

Table 1 below shows a summary of the performance test results conducted in the original study. TDCE performance statistics can be found at the end of the table.

Configuration	Rapid tasks	Fast tasks	Medium tasks	Long tasks
Tasktime (sec)	1	5	30	60
Job time per processor (sec)	240	240	240	240
Tasks per processor (#)	240	48	8	4
Processors (#)	1408	1408	1408	1408
Total tasks (#)	337920	67584	11264	5632
Total processor time (h)	93.7	93.7	93.7	93.7
Runtimes (sec)				
Slurm	2774 2787 2790	622 603 606	280 278 255	287 264 300
GE	3057 3073 3082	622 634 623	278 279 277	275 281 274
Mesos	1794 1795 1792	366 364 367	280 280 281	306 306 305
Hadoop YARN	-	2013 1798 1710	479 472 510	342 445 347
TDCE	**275.5** **254.6** **254.1**	**242.3** **245.7** **242.1**	**240.7** **240.8** **241.0**	**240.5** **240.7** **240.6**

Table 1. Performance statistics from the original study. TDCE performance statistics added to the last row.

Figure 1 and Table 2 below show the system utilization for Slurm, GE, Mesos, YARN in a similar manner as in the MIT study. TDCE utilization rates have been added for comparison purposes. Utilization rates were determined by comparing the theoretical ideal runtime of 240 seconds to the fastest test run for each scheduler.

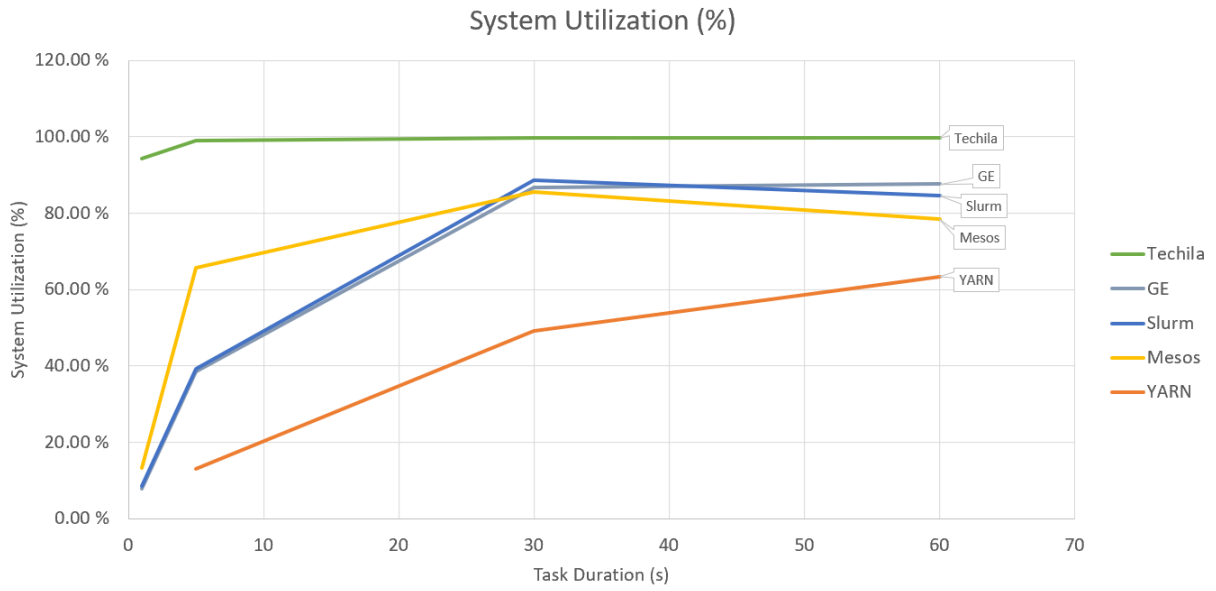

Figure 1. System utilization rates.

Case		Overhead seconds	Total overhead (%)	System utilization (%)
1 second	**Slurm**	2544	1059.86 %	8.62 %
	Mesos	1554	647.36 %	13.38 %
	YARN	N/A	N/A	N/A
	GE	2817	1173.75 %	7.85 %
	Techila	14.32	5.97 %	94.37 %
5 seconds	**Slurm**	370	154.31 %	39.32 %
	Mesos	126	52.36 %	65.63 %
	YARN	1600	666.81 %	13.04 %
	GE	382	159.17 %	38.59 %
	Techila	2.44	1.02 %	99.00 %
30 seconds	**Slurm**	31	12.92 %	88.56 %
	Mesos	40	16.81 %	85.61 %
	YARN	247	102.92 %	49.28 %
	GE	37	15.42 %	86.64 %
	Techila	0.84	0.35 %	99.65 %

60 seconds	**Slurm**	44	18.19 %	84.61 %
	Mesos	66	27.36 %	78.52 %
	YARN	138	57.50 %	63.49 %
	GE	34	14.17 %	87.59 %
	Techila	0.59	0.25 %	99.75 %

Table 2. System utilization rates in numerical format.

Analyzing and summarizing the results

Using sleep tasks to measure system responsiveness and throughput is a valid approach, as it limits the number of moving parts and allows test cases to be carefully designed to measure the desired metric. In the referenced MIT study, this approach was used to measure distinct differences between several popular HPC schedulers. Alternative approaches such as LLMapReduce and task grouping were also proposed to improve system utilization and overall scheduler performance.

The usage of sleep tasks makes this test well suited for reproduction, as the advancements in processor architecture will not have had any impact on the task execution time on the compute nodes.

The goal of this benchmark was to see how well the TDCE scheduler performs when compared to other popular schedulers that were benchmarked by MIT in the original study. When measuring scheduler performance, the system utilization rate is a valuable and well suited metric as it gives insight about both the actual runtime and the overall cost of the computation.

TDCE achieved a system utilization of **over 94%** in all test cases, rising **above 99%** when the task duration reached 5 seconds or longer. This is significantly higher when compared to results achieved by the schedulers in the original study, where the system utilization rates were **between 8,62% and 88,56%.**

To further highlight the differences in scheduler performance, we can compare the absolute overheads between TDCE and the schedulers in the original study. The absolute overhead for TDCE ranged from **14.32 seconds** (for 1 second tasks) **to 0.59** seconds (for 60 second tasks). These overheads are significantly smaller when compared to the best competing schedulers in the original study where the overhead ranged from **1554 seconds** (Mesos when processing 1 second tasks) **to 44 seconds** (Slurm when processing 60 second tasks).

Practical implications

TDCE provides excellent resource utilization combined with minimal overhead, even when processing extremely short computational workloads. This translates to the following benefits:

Time savings. TDCE completes the workloads in a shorter wall clock time than any of the competing traditional HPC schedulers in the original study. If you are an end-user, this means you will not need to spend your time waiting for results and can focus on your actual work.

Infrastructure cost savings. If you are processing your computations using on-demand cloud capacity, you will typically have to pay by the second, even when the computational nodes are idling. This means that the computations get more expensive as the wall clock time increases. TDCE ensures that the costs stay low by getting the computations done in the shortest possible time.

High capacity utilization. While end-users are not necessarily concerned about the efficient utilization of computational capacity, this can be a relevant metric for anyone working in HPC IT management. TDCE ensures that the HPC capacity you are managing is used to its fullest potential and that it does not spend time idling between workloads.

Low learning curve. The original study presented multilevel scheduling as an alternative approach that could be used to keep the capacity utilization high. TDCE scheduler performs extremely well even with the most granular workloads, meaning that the user does not need to artificially increase the computational work done in a single task. However, for situations where the user still prefers to increase the computational work done in each task, TDCE provides built-in functionality for several popular programming languages that can increase the task length automatically, without user intervention.

Disclaimer and parting words

TDCE benchmark was done by Techila Technologies. MIT was not involved in these tests in any way. Computing capacity used was 44 x 32-CPU core machines.

Benchmarking studies conducted by companies about their own product are understandably difficult to take at face value and their scientific value may well be disputed. If you are looking to improve the utilization and throughput of your computing environment but are hesitant about the results or methods presented here, **please feel free to contact sales@techilatechnologies.com for additional information about the benchmark methods used or to schedule a demo.**

References and further material

Albert Reuther*, Chansup Byun, William Arcand, David Bestor, Bill Bergeron, Matthew Hubbell, Michael Jones, Peter Michaleas, Andrew Prout, Antonio Rosa, Jeremy Kepner (2017). Scalable System Scheduling for HPC and Big Data. Journal of Parallel and Distributed Computing, Vol. 111, January 2018. Preprint available online: https://arxiv.org/pdf/1705.03102.pdf

More information about TDCE can be found in the TDCE product description. Additional details about the scheduler functionality can be found in the TDCE scheduler document.