jamk | IT

HPC-Performance tests

Jyväskylän ammattikorkeakoulu

Lassi Partamies, Joni Korpihalkola, Juha Jokinen, Mika Rantonen & Janne Alatalo
9.12.2020

1 DEEP LEARNING BENCHMARK FOR HIGH PERFORMANCE COMPUTING SYSTEM

While a standard deep learning benchmarking framework does not exist, this document describes a few datasets and neural network models, where metrics from training can be used to to measure the deep learning performance of the HPC system. The main metrics are throughput, which describes how many images or tokens are trained on the neural network per second and batch and epoch execution times.

2 TESTING SYSTEM SPECIFICATIONS

The relevant hardware and software specifiations for the high performance computing system are as follows:

- Four 16-core Intel Xeon Gold 6130 processors, 64 cores in total
- 768 GB DDR4 memory, error-correcting
- Four NVIDIA Tesla V100 32GB GPUs, though NVLink is not available
- CentOS operating system
- Tensorflow 2.1.0
    - Python 3.7.4
    - CUDA 10.1

3 DATASET AND MODELS

In order to create a benchmark, several scripts with Python were created. These scripts load a preprocessed dataset, which is converted into a TensorFlow dataset class. This high-performance data pipeline is used to train a neural network, where the structure of the network is loaded from TensorFlow libraries. The first benchmark is focused on image recognition datasets and models. The datasets are DeepWeeds, where eight different weed species are captured in 17,509 images. The second dataset is ImageWang, which is a combined version of Imagenette and Imagewoof datasets. The dataset classifies 14,669 images into 20 classes.

The tensorflow backend uses float32 datatypes in all of the examples. The first neural network is a deep convolutional neural network InceptionV3, which was the first runner up for image classification in ImageNet Large Scale Visual Recognition Competition in 2015. Deep residual neural network ResNet50V2 is the second tested neural network model.

The inference throughput and time of OpenAI's GPT-2 language generation model was also tested. The tested GPT-2 models were the 124 and 1558 million parameters versions. The dataset contains 216,930 questions from Jeopardy. The GPT-2 model generates 30 tokens long extensions to the questions in Jeopardy dataset. The dataset was parsed to 100,000 questions and the dataset was divided into four parts, each GPU then performed inference into the divided dataset. Below are listed average throughput and time elapsed values.

4 BENCHMARK RESULTS

Below are the training metrics for neural networks trained on image datasets.

Model	Dataset	Training throughput (batch size 64)	Inference throughput (batch size 1)	Mean batch time	Mean epoch time	Training time
InceptionV3	DeepWeeds	239 images/s	64 images/s	0,2676 seconds	58,71 seconds	2935 seconds
InceptionV3	ImageWang	160 images/s	59 images/s	0,3968 seconds	73,17 seconds	3658 seconds
Resnet50V2	DeepWeeds	244 images/s	92 images/s	0,2616 seconds	57,40 seconds	2870 seconds
Resnet50V2	ImageWang	163 images/s	80 images/s	0,3920 seconds	72,28 seconds	3614 seconds

GPT-2 benchmark results are listed below.

Model	Dataset	Mean length of input	Mean Inference throughput	Mean Inference time
GPT-2 (128M)	Jeopardy	14,58 words	56,81 words/s	0,256 seconds
GPT-2 (1558M)	Jeopardy	14,58 words	16,26 words/s	0,896 seconds

The results can be compared to NVIDIA's own benchmarks listed on Nvidia's website[1]. The tests are performed on NVIDIA's docker containers, with batch sizes ranging from 256 to 384 in image datasets. The inference benchmarks use NVIDIA's TensorRT to enhance performance, TensorRT was not used in benchmarks above.

5 BENCHMARK CONCLUSION

Nvidia has benchmarked it's V100 GPUs single and converged training performance on multiple datasets, the benchmarks are available at https://developer.nvidia.com/deep-learning-performance-training-inference. The exact setup for these benchmarks could not be replicated due to the lack of dataset preprocessing and neural network training scripts. The benchmarked datasets and models however are similar to the ones in Nvidia's benchmarks, and the results are similar when the differences in batch sizes are taken into account.

6 SPARK BENCHMARK FOR HIGH PERFORMANCE COMPUTING SYSTEM

Spark-Bench (https://codait.github.io/spark-bench/) is system for benchmarking and simulating Spark jobs and is used to measure HPC's performance. Spark-Bench is used to generate datasets and run tests with that data. Mainly focus of this test is to create workloads to HPC's CPU and see how long running certain task with different setups will take.

Tests are split between two categories: Generating test data and Kmeans clustering test. During tests data is generated with Spark-Bench Data Generator – KMeans[2]. In the test multiple datasets are generated with different row count and changing available CPU cores. As a result we will get the amount of time it took in different tasks (generating-, converting- and saving data).

With KMeans test we test how long it takes to run KMeans clustering algorithm in dataset created in Data Generator test. Test's output will show the amount of time that algorithm took with loading-, testing and training the model. All tests are ran 10 times and compared between means of other tests with different setups.

7 SPARK BENCHMARK TEST DESCRIPTIONS

Generating data:

- TEST1
    - 1M Rows, 13 Columns

[1] https://developer.nvidia.com/deep-learning-performance-training-inference

[2]https://codait.github.io/spark-bench/workloads/data-generator-kmeans/

  - 3 Tests with 1, 10 and 64 CPU cores available

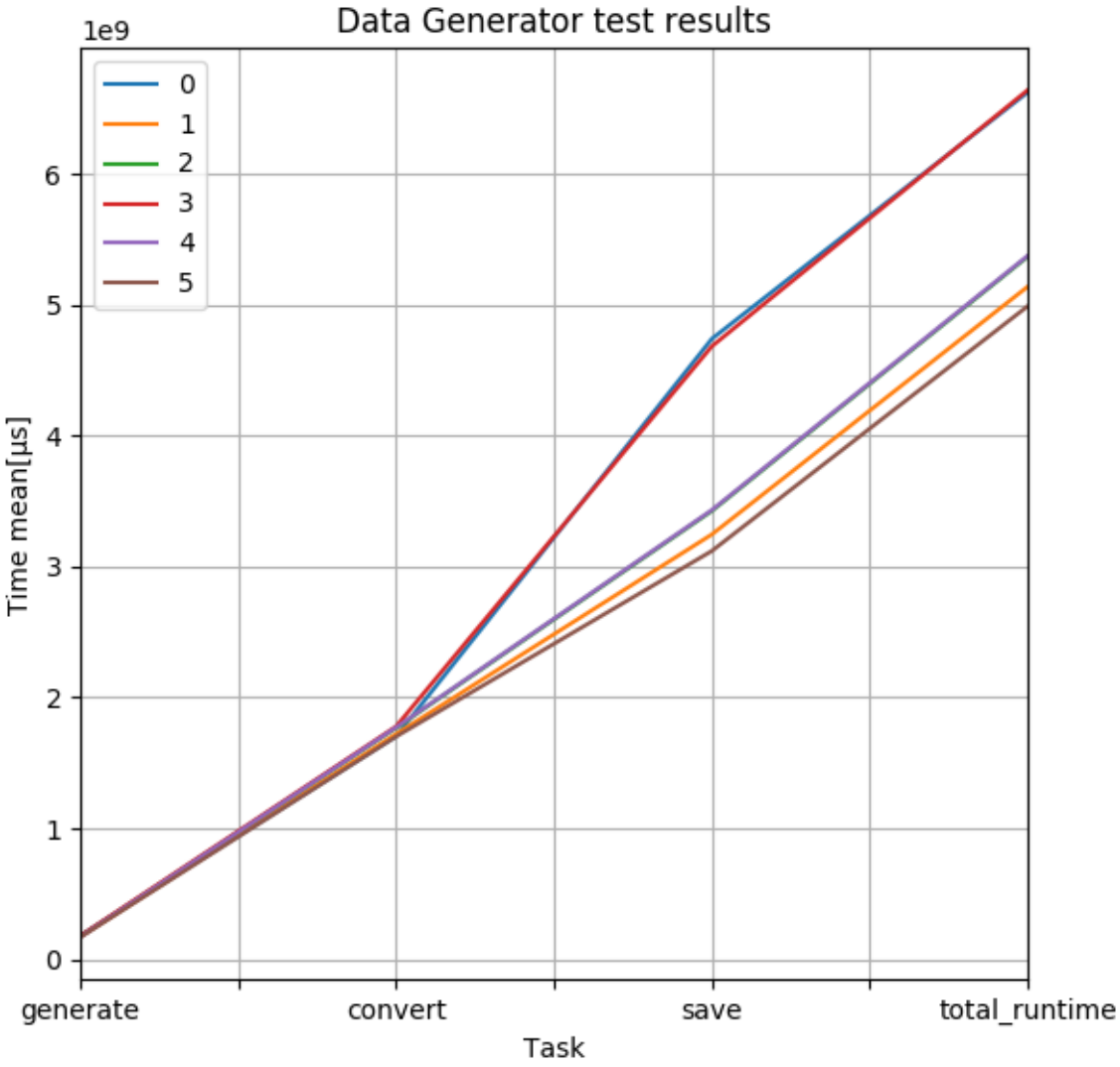
- TEST2
  - 100M Rows, 13 Columns
  - 64 CPU cores available

Kmeans:

- TEST1
  - 2 Clusters
  - 3 Tests with 1, 10 and 64 CPU cores available
  - 1M Rows, 13 Columns
- TEST2
  - 10 Clusters
  - 3 Tests with 1, 10 and 64 CPU cores available
  - 1M Rows, 13 Columns
- TEST3
  - 2 Clusters
  - 64 Cores
  - 100M Rows, 13 Columns

8 SPARK BENCHMARK DATA GENERATING RESULTS

TEST1



Legend explained:

0: 1 Core

1: 64 Cores

2: 10 Cores

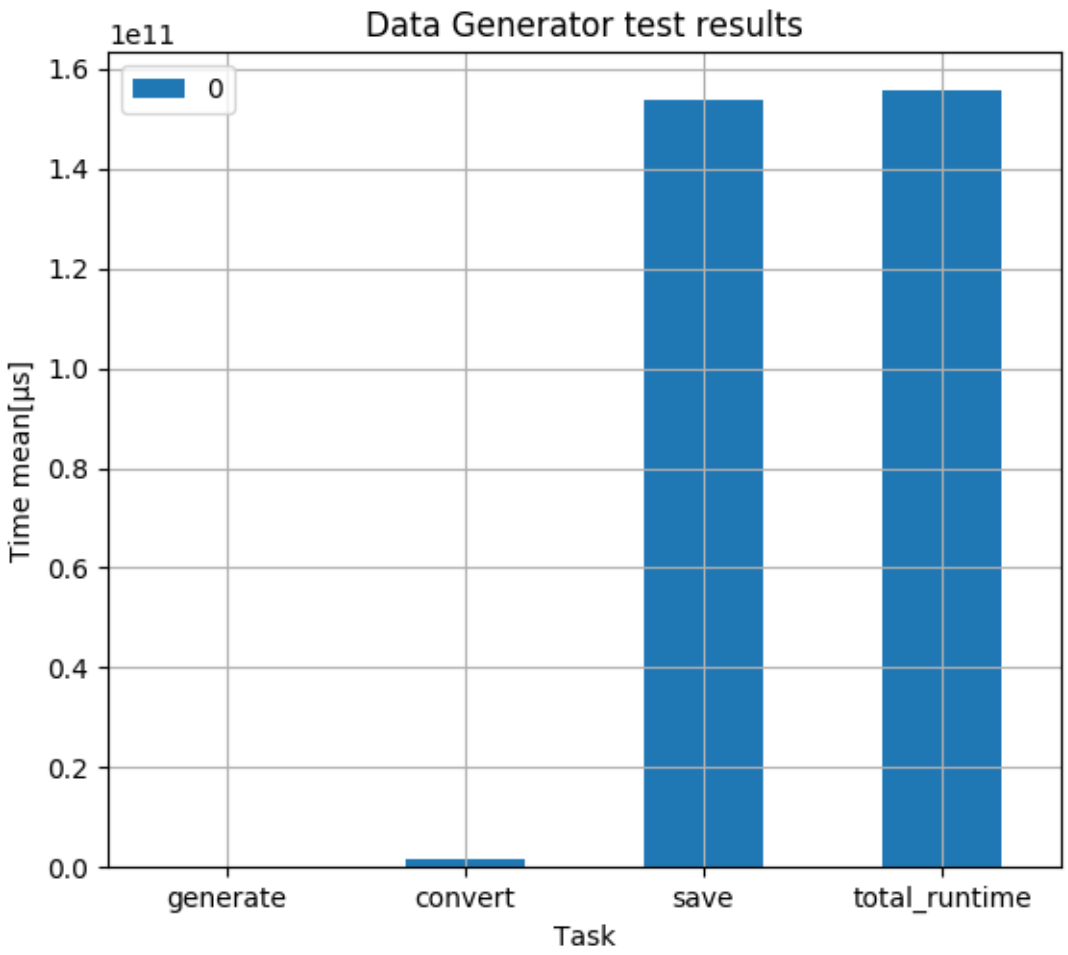
3: Cores 1

4: Cores 10

5: Cores 64

In TEST1 we can see the amount of time that took to create dataset with 100M rows and 13 columns with different setups. In figure we can see that time used in saving data is reduced when operating with more CPU cores available.

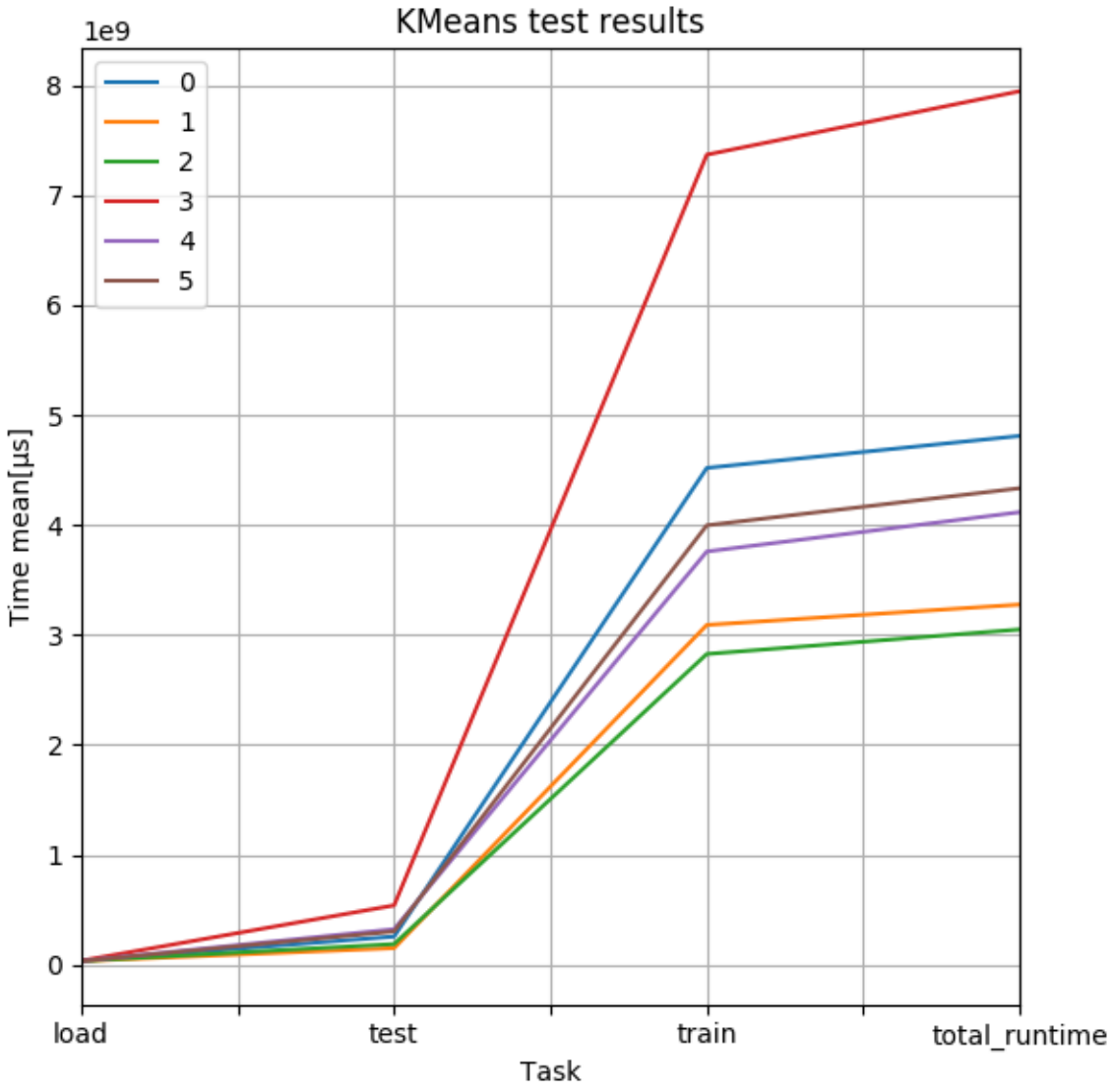
TEST2:



In TEST2 dataset with 100M rows and 13 columns is created in 2.59 minutes. Most of the time was spent on saving data into a disk in as parquet file format. Genereting data only took 0.17 seconds and converting took 1.7 seconds. Total filesize 10.3G.

9 SPARK BENCHMARK KMEANS RESULTS

TESTS 1&2:



Legend explained:

0: 1 Core, 2 Clusters

1: 64 Cores, 2 Clusters

2: 10 Cores, 2 Clusters

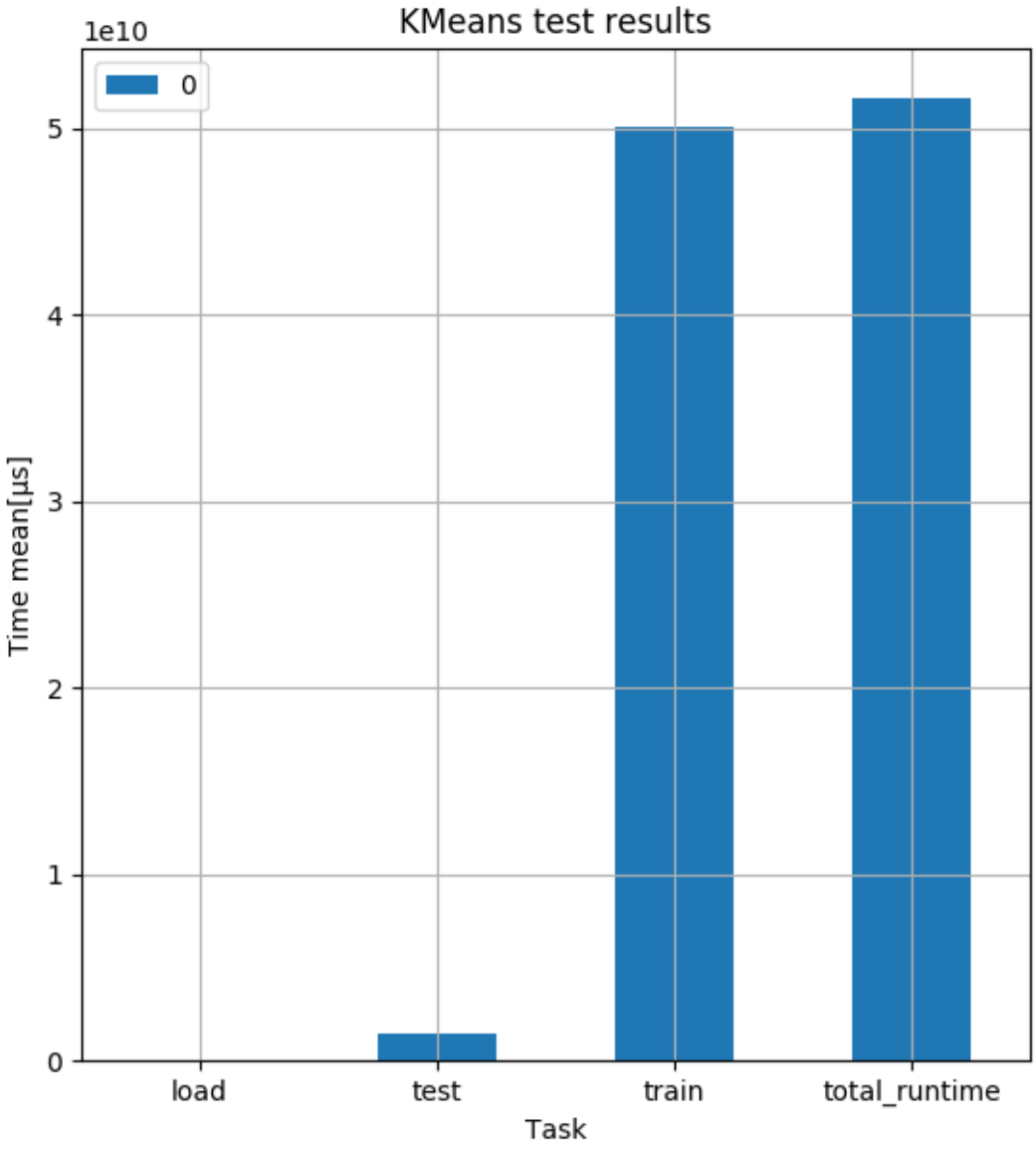
3: 1 Core, 10 Clusters

4: 10 Cores, 10 Clusters

5: 64 Cores, 10 Clusters

In figure we see run times of tests 1 and 2. Most of the time is used on model's training. Again we can see that time is reduced by adding more than 1 CPU cores as available.

TEST 3:



In TEST 3 we can see the time took by training KMeans model with 2 clustes in dataset containing 100M rows.

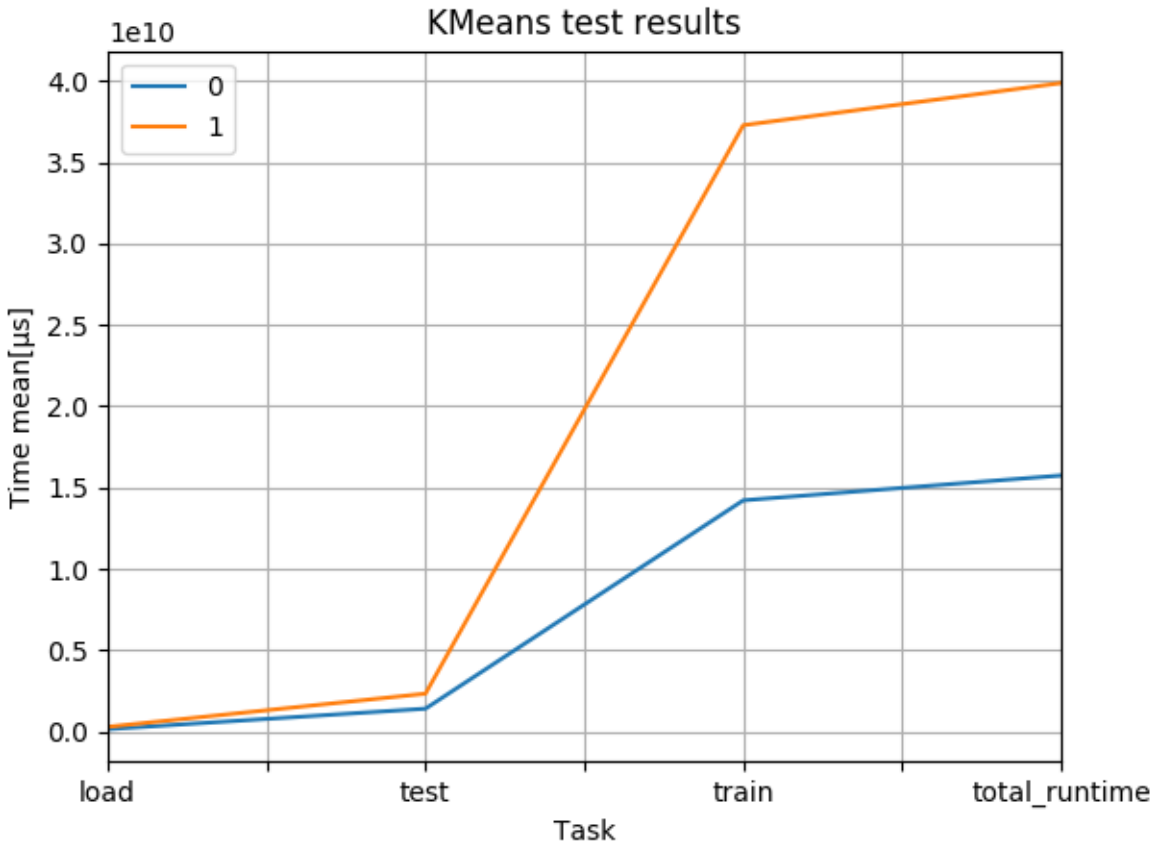
10 SPARK BENCHMARK COMPARISON

Since there is no any specific comparison chart in Spark-Bench we do comparison between HPC and ordinary business laptop. Specification of laptop used in comparison:

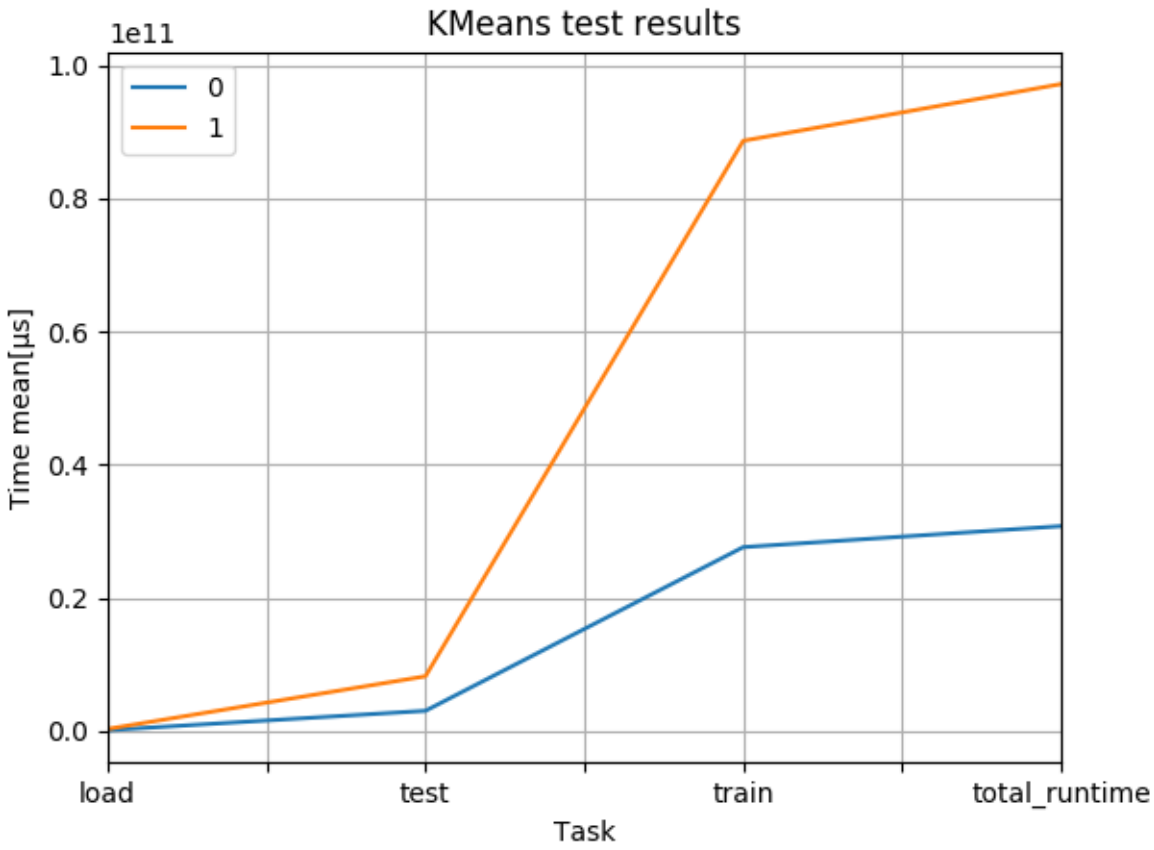
- Model: HP EliteBook 830 G6
- CPU: Intel i5-8265U (4 Cores)
- Memory: 8GB DDR4

In comparison we use Spark-Bench KMeans test with dataset of 10M rows and 14 columns. All tests are ran once and with all cores and memory available. Tests with larger datasets could not be used due the lack of memory on laptop.

In first comparison test was ran with 10 clusters:



Legend explained: 0 = HPC, 1 = Laptop. As seen on comparison, HPC is significantly faster with larger datasets. Next test is ran with 50 clusters and same dataset:



11 SPARK BENCHMARK CONCLUSION

In these test we can see how HPC handles large masses of datas with machine learning algorithms. HPC is capable of handling and creating large groups of data and can be used to implement complex machine learning models into those masses of data. During tests HPC was functioning seamlessly without any issues during these stress tests.

Performance can not be rated universally due the lack of comparison charts in Spark-Bench tests, but when comparing with business laptop we can see that it is significantly faster when operating with HPC. Also HPC is capable of handling large data masses which compared computer could not.