Readings:

- 80% thoroughly read “Understanding ML driven HPC: Applications and infrastructure”
- 80% thoroughly read “A taxonomy for the integration of machine learning and simulations
- 80% thoroughly read “Up to two billion times acceleration of scientific simulations with deep neural architecture search”
- Reread “Learning everywhere: Pervasive machine learning for effective High-Performance computation.”
- Reread “Smart Surrogates: Uncertainty-Aware Efficient General Surrogates for Simulations”

RCT:

Below are the results and analysis of the Run Time for 6 jobs executed on Comet supercomputer:

A) 1 Task that “sleeps” (suspends execution) for 1 second

B) 1-5 tasks that run concurrently.

C) 1-5 tasks that run sequentially.

D) 128 tasks concurrently, where each task is 1 core.

E) 16 concurrent batches of 8 tasks (each of 1 core), but where in each batch each task runs sequentially, i.e., one after the other.

F) 8 concurrent batches of 16 tasks (each of 1 core), but where in each batch only 8 task runs concurrently.

Job A: 1 Task that “sleeps” (suspends execution) for 1 second

The goal of Job A is to get a sense of the overhead resulting of using EnTK. The results are as follows

Conditions	Run Time
Local machine without EnTK	1.001136064529419
Comet supercomputer with EnTK	394.97047686576843

Job B: 5 tasks that run concurrently → 1 pipeline with one stage containing 5 tasks

Figure 1 shows the results of Job B when run on comet supercomputer. The excutable was a)'/bin/bash' or b)a simple program that sleeps for 100 seconds

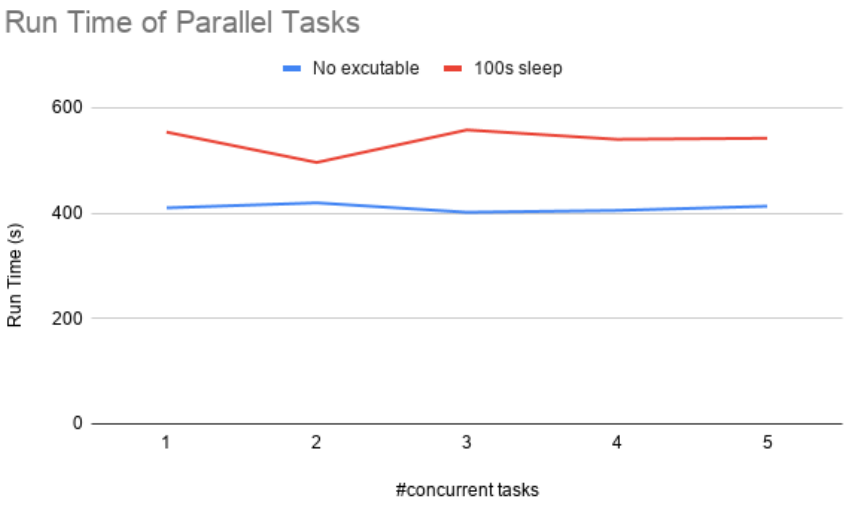

Figure 1

As shown in the figure, the overhead for using EnTK and running on comet is ~400s. When the task's executable is to "sleep" for 100 seconds, we get an (almost) constant run time regardless of the number of tasks running concurrently. The average for the 5 points is 537.8741436s and the standard deviation is 24.57320116.s The standard deviation is within 5% of the average.

Job C: 5 tasks that run sequentially → one pipeline with 5 stages; each has 1 task

Figure 2 shows the results of Job C when run on comet supercomputer. The excutable was a)'/bin/bash' or b)a simple program that sleeps for 100 seconds.

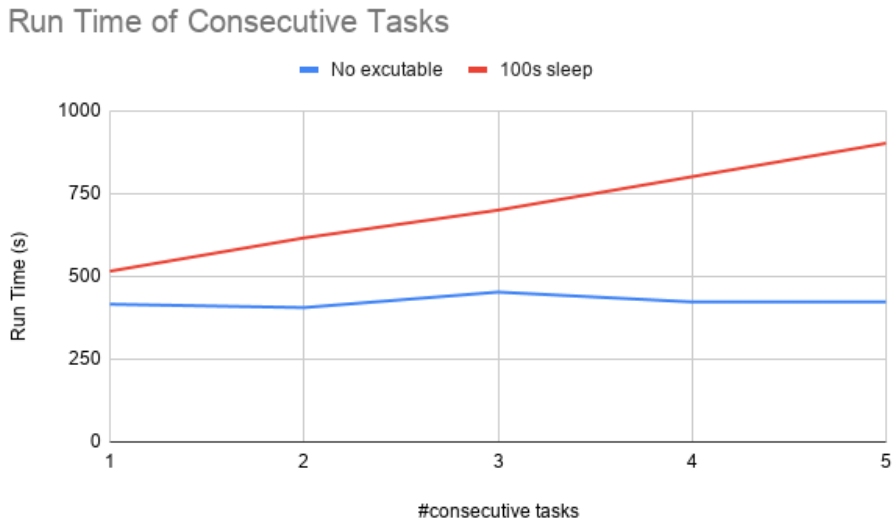

Figure 2

As shown in the figure, the overhead is ~400s. Increasing the number of tasks by 1 increases the run time by ~100s. This is expected sine the tasks are running sequentially such that task i-1 has to excute before task i.

Job D: 128 tasks concurrently, where each task is 1 core → 128 tasks in one stage

Executable	Run Time (s)
Sleep for 100 seconds	610.2394454479218

Job E: 16 concurrent batches of 8 tasks (each of 1 core), but where in each batch each task runs sequentially, i.e., one after the other → 16 concurrent pipelines, each has 8 stages; each stage has 1 task

Executable	Run Time (s)
Sleep for 100 seconds	1360.406531572342

I expected the run time for job E should be 700 seconds longer than that for Job D. This is because in job E, there are 8 tasks running sequentially at a time while in job D there were no two tasks running sequentially. The expiremnt met my expectations.

Job F: 8 concurrent batches of 16 tasks (each of 1 core), but where in each batch only 8 task runs concurrently. → 8 concurrent pipelines, each has 2 stages; each stage has 8 tasks

Executable	Run Time (s)
Sleep for 100 seconds	726.0380449295044

I expected the run time for job E to be 100 seconds longer than that for Job D due to having two conseqcutive stages in each pipeline. The expiremnt met my expectations.

For all of the jobs, the overhead for using EnTK and running on comet supercomputer is ~300s-550s. The time to terminate the application manager on average ≈ 185.8s