Understanding Job Execution in Supercomputers

Alex Moore, UNC Charlotte
Di Zhang, High Performance Computing (HPC)

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

Introduction

Super Computers are built to solve complex problems that require high-speed data processing, such as:

- Spacecraft simulation
- Analyzing cyber attacks and natural disasters

This is done by running programs or jobs, which are then scheduled through a process known as job scheduling.

Why is job scheduling important?

- Minimizes job waiting time, slowdown, or completion time.
- Allows for more complex problems to be solved.

Objectives

Methods to improve job scheduling:

- **TRIP** "Trade-off between Prediction Accuracy and Underestimation Rate in Job Runtime Estimates"
  - ML framework that utilizes data censoring to improve prediction accuracy with a low underestimation rate of job runtimes.
- **SchedInspector** "SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning"
  - Determines the competency of scheduled jobs by examining the decisions of the base job scheduler.

Method

Phase 1: Learning

During this phase I learned:

- Process of estimating HPC Batch Job Schedulers Runtime
- Methods of improving prediction accuracy and job execution performance
  - TRIP
  - SchedInspector
- Fundamentals of Machine Learning
  - Reinforcement Learning
- Pandas
- Jupyter Notebook

Phase 2: Data Mapping

- Utilized Jupyter Notebook and Pandas to map data from previous traces
  - Submit time, Wait time, Run time, Requested Time, User ID, etc

Phase 3: Machine Learning: Feature Extraction

- Extracted features from previous job traces to determine trends amongst the data

TABLE III: The features used for runtime prediction. Here, a class contains jobs with the same user name, project name, and job name.

Feature	Description
t_{last1}	the actual runtime of the last job of the same class
t_{last2}	the actual runtime of the second-to-last job of the same class
$t_{supplied}$	user-supplied job runtime estimate
$n_{supplied}$	the number of nodes requested by the user
$A_{average}$	the average accuracy of the historical jobs of the same class
A_{max}	the maximum accuracy of the historical jobs of the same class
$t_{longest}$	the longest actual runtime of the historical jobs of the same class
$t_{longest10}$	the longest actual runtime of the ten last jobs of the same class
$t_{average}$	the average actual runtime of the historical jobs of the same class
$t_{average10}$	the average actual runtime of ten last jobs of the same class
$t_{percentile25}$	the actual runtime of the 25th percentile historical jobs of the same class

Results

- Mapped data from previous job traces
- Successfully extracted each of the listed features

	Job Number	Submit Time	Wait Time	Run Time	Allocated Processors	Average CPU time used	Used Memory	Requested Processors
0	11	566129	5	28826	1	27758	-1	1
1	12	566290	532	26171	1	24666	-1	1
2	13	567314	15757	8071	8	7334	-1	8
3	14	571164	21568	64832	32	44134	-1	32
4	15	574294	4647	64384	7	58033	-1	7
...	...	...	...	...	...	...	...	...
59710	73492	63562563	11	71	4	40.25	-1	4
59711	73493	63562705	24	75	4	42.00	-1	4
59712	73494	63563061	11	72	4	40.25	-1	4

Mapping of data from SDSC-SP-1998-4.2 (1)

Requested Time	Requested Memory	Status	User ID	Group ID	Application Number	Queue Number	Partition Number
28800	-1	5	153	75	18180	3	-1
28800	-1	1	153	75	18184	3	-1
64800	-1	1	150	6	13592	4	-1
64800	-1	5	6	6	13602	3	-1
64800	-1	1	151	75	13607	3	-1
...	...	...	...	...	...	...	...
1200	-1	1	3	74	56633	1	-1
1200	-1	1	3	74	56636	1	-1
1200	-1	1	3	74	56638	1	-1

Mapping of data from SDSC-SP-1998-4.2 (2)

```
#Feature tlongest
#the longest actual runtime of the historical jobs of the same class

#groups data by class of job
grouped = df.groupby(["User ID"])

#max value of "Run Time" column for each group
max_runtimes = grouped["Run Time"].max()

#longest actual runtime of historical jobs of same class
longest_runtime = max_runtimes.max()

print(longest_runtime)
```

9996

Example of feature tlongest from SDSC-SP-1998-4.2

Hewlett Packard's Frontier; The world's fastest supercomputer

Conclusions

- This data provides valuable, real-world insight into the performance of job scheduling.
- Features will be compared with other past job traces to determine trends in the data.
- These trends will help in the implementation of the TRIP model and SchedInspector.
- This will allow for a comparison of their benefits and look further into other methods of improving job scheduling performance.

References

Yuping Fan, Zhiling Lan, Paul Rich, William E. Allcock, Michael E. Papka Trade-off between Prediction Accuracy and Underestimation Rate in Job Runtime Estimates Table III
Di Zhang, Dong Dai, Bing Xie SchedInspector: A Batch Job Scheduling Inspector Using Reinforcement Learning