ENS 491 – Graduation Project (Design)
Progress Report I

Real-Time Supercomputer Monitoring

Ali Eren Ak, Deniz Batu, Muhammed Orhun Gale

Supervisor: Kamer Kaya

June 12, 2022

1 Project Summary

Increasing complexity in the High-Performance Computing (HPC) clusters due to the improvements whose aim is to boost the computational efficiency entails various bottlenecks in these clusters both software-wise and hardware-wise. Therefore, detecting and overcoming bottlenecks in order to use HPC clusters on their true scales is an important research and practice topic nowadays.[KSV+21] [TRQR21] [BG09] To be able to attack bottleneck problems, the first requirement is being able to detect bottlenecks in HPC clusters. Bottleneck detection in HPC clusters can be achieved via detailed monitoring of the cluster and interpreting the gathered data. While monitoring of the cluster requires complex tools, interpretation can be done via using Machine Learning (ML).[IRM+20] [PPG20] [HBK05] [DWKN20] A digital twin is a synchronized computerized model of a physical device which has an active communication with the physical device in order to create a real-time image of the physical device.[Che17] Even though there exist various approaches to detect bottlenecks in HPC clusters, conceptually, creating a Digital Twins of HPC clusters would make this process more structured and reproducible. In this sense our project, "Real-Time Supercomputer Monitoring", aims to establish a framework which can create digital twin of a HPC cluster which will gather clusters data in a structured manner to enable vendors and customers to detect hardware, operating system, and user-program bottlenecks, provide them performance and efficiency insights and perform prediction-based simulations of various hardware/software scenarios via using the models that are trained via the cluster's digital twin.

2 Scientific/Technical Developments

In the project the first aim is to extract as many metrics as possible of an HPC system. This turned out to be a very complicated process since most metrics are not readily available. Accessing various metrics requires multiple libraries, command lines and scripts. Some of the metrics are only accessible indirectly, meaning that they can be interpreted in relation with other metrics.

Since the proposal, we have realized the complications of extracting metrics and focused on ways to extract and store them. We have decided to use Python for scripting over the Docker Container we

created for extracting and displaying the data while simulating the Digital Twin structure. 'likwid', 'lshw', 'CDLL', 'libnuma-dev' libraries were used to construct the data extraction system. C compatible libraries were preferred as they are less likely to cause restraints on other platforms and devices. 'vmstat' , 'diskstats', 'meminfo' are some of the Linux command lines that are used to extract information about processes, memory, paging, block IO, traps, CPU activity, major number, minor number, device name, reads completed successfully, reads merged, sectors read, time spent reading (ms), writes completed, writes merged, sectors written, time spent writing (ms), I/Os currently in progress, time spent doing I/Os (ms), weighted time spent doing I/Os (ms), discards completed successfully, discards merged, sectors discarded, time spent discarding, flush requests completed successfully, time spent flushing etc. 'numastat' Linux library is used for extracting following NUMA information and more:

numa_hit: memory successfully allocated on this node as intended.
numa_miss: memory allocated on this node despite the process preferring some different node.
numa_foreign: memory intended for this node, but actually allocated on some different node.
interleave_hit: interleaved memory successfully allocated on this node as intended.
local_node: memory allocated on this node while a process was running on it.
other_node: memory allocated on this node while a process was running on some other node.

These libraries and command lines give detailed insight on device, GPU, CPU, OS, network types and real time usages. Project's approach is to analyze a HPC system is layer by layer such that the data will be collected, visualized, trained, and interpreted on different layers namely the deepest and most detailed layer being the per-CPU level to highest and most populated being the entire HPC system. To achieve this granularity, relational database diagrams are being drawn to group statistics according to layers such as per-node, per-socket, per-CPU, per-GPU, per-core. The data acquisition step is ongoing, and it gave insight on how to test the project. The eventual goal of the project is a tangible performance increase over a multitude of HPC systems so, every test aims to detect an increase in performance however, to reach the point in the project where a HPC system can fully be tested, there are many more steps that need to be tested to make sure that they yield proper results.

No data acquisition should create a significant tension over various HPC systems over different jobs. Jobs will be run over HPC systems with and without data acquisition, the slowdown that data operations cause should be no more than %3 for any jobs over any HPC systems. If the threshold is exceeded, either acquisition methodology needs to be changed or metrics that give less or no insight to job performances should stop being extracted from HPC systems. This discarding process requires deep understanding and experimentation of metrics and configurations in relation to performance. HPC configurations over different jobs and systems will be changed, recorded with their performance (total running time) systematically and evaluated to detect the job-significant metrics and settings. The election process is also necessary to not overpopulate the visualization part with insignificant metrics. NUMA settings can be changed through 'numactl' Linux library.

In later stages, manual interference commands' accuracy, acquired data consistency, different machine learning methods and their accuracy on both run time and postmortem data, real time visualization accuracy needs to be tested before the Digital Twin is online. After Digital Twin is online, machine learning results and manual interference will be applied to the Twin and the real HPC system to observe the symmetry. At this stage, metrics to collect might be re-evaluated to optimize sameness and overhead. After this point, all the previous steps and tests will be done again to construct further versions of the Digital Twin to achieve better results in terms of robustness, likeness, delay and performance.

3 Encountered Problems

This project aims to establish a framework that can efficiently create a digital twin of an HPC cluster to gather and store data in a structured manner. Collected data will be used to train models that detect HPC clusters' bottlenecks. In this manner, it is a very significant step to create a digital twin of the HPC cluster using the extracted system information. In the first progress of the project, the base

model that represents the HPC cluster in planned format, JSON-LD, is recursively formed. Thus, the project goals are still relevant, and progress will continue in the development. During the design and development, we aim to create software that can be scaled and reproduced among different computers and hardware. For instance, different computer processors are produced by the same manufacturer, such as Intel Atom and Intel Skylake, or by different manufacturers, such as AMD Interlagos and Intel Icelake. Therefore, it is an essential and challenging step to design a digital twin that can form different architectures and be able to record processor metrics. To achieve that, low-level libraries and container technologies have been used. Most of the developed functions depend on low-level libraries that already exist in the system. However, in the development process, the Docker container is used to standardize the programming environment. Thus, both development and production steps are standardized into industry-wide used systems.

In the progress, the development of the digital twin and metric extractor part is very significant and affects other parts such as machine learning and data visualization. Thus, the progress is behind the planned timetable. However, the achievements that we obtain from the current step, which is digital twin establishment and metric extraction, will accelerate other steps. Thus the effort was mostly spent on the design and development of this part. Until the second progress, it is aimed to complete digital twin development and the visualization of the digital object. In order to visualize the digital twin, it is planned to develop software that shows the hierarchical structure and details of the twin. Thus, React-flow is a widely used React-Native package for building interactive node-based UIs, editors, flow charts, and diagrams. In addition to the user interface of the twin, it is planning to add metric charts to the web application to visualize real-time HPC data. In this part, it is needed to develop a programming interface between the web application and the database, which contains digital twin architecture, metric data, etc.

Currently, development mostly follows the design of the project in the proposal. There is not any significant change made until now. Instead, it is aimed to depend on the main design while fixing problems in the development. The most crucial difference made in the structure of the digital twin. We initially tried to design the digital twin structure based on NUMA domains to represent the physical device precisely. However, we decided that a more unsophisticated design would make representation more efficient to comprehend and analyze. Therefore, we moved a design that recursively creates the digital twin structure of the HPC cluster by considering every part of the cluster which have computational power in a top-down fashion. We named each unit with computational power Compute Unit (C-Unit) and recursively created the digital twin structure according to the Azure DTDL. In addition, to do that, we changed some implementation in the system and metric parser scripts to make it more scalable and reproducible throughout using low-level C libraries or Command-Line Interfaces called from Python scripts.

4 Tasks to be Completed Until Progress Report II

4.1 Digital Twin Implementation

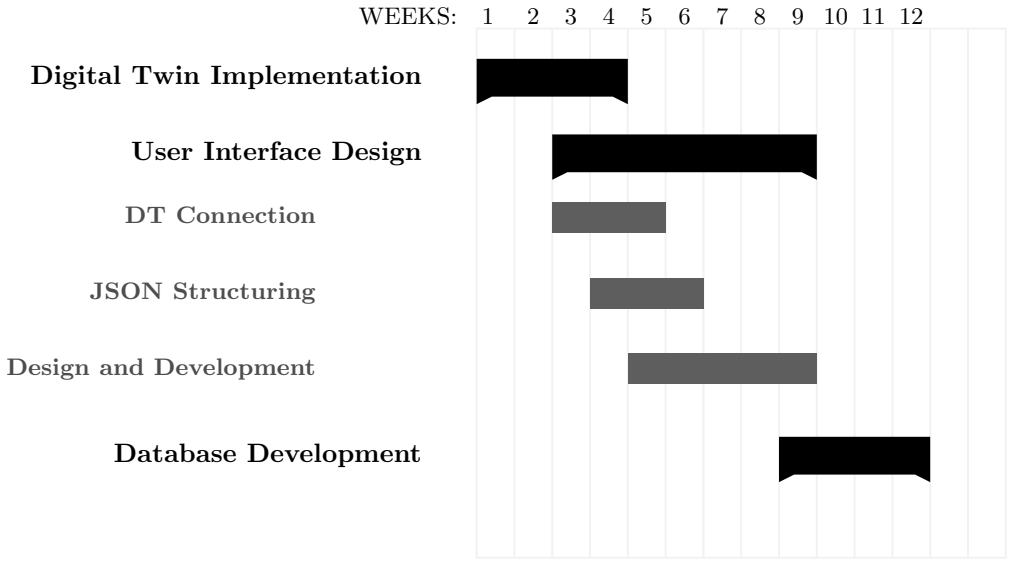
Digital Twin is one of the core parts of the project. We believe a better computer representation could help us to understand and predict the system better. To do that digital twin structure is carefully designed and formed in this progress. In the second part of the project, we plan to develop a service for the digital twin to connect the user with the application. We plan to complete this structure and connect it to the database and user interface in the second progress.

4.2 Database Design and Development

In the second step, we are planning to complete database development powered by a back-end service to connect the digital twin and the data with the user interface. Until this step, the development of the digital twin and the system parser are independent, which means there is no connection. However, we will also design a system in the second step to solve this problem.

4.3 User Interface Design

In the user interface, we will serve our findings, predictions, and system insight to the user. Thus, it is essential to develop an interface that is informative but also easy to use. Firstly, we plan to develop a graph showing the digital twin structure. Since there will be detailed mass information about the system, the optimization is required to align nodes and information inside the graph. That's why we plan to use React-Flow to establish a graph with aligned nodes and edges. Users will be able to expand and collapse nodes and view details about the nodes.



References

[BG09] Nathan Bell and Michael Garland. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, New York, NY, USA, 2009. Association for Computing Machinery.

[Che17] Yubao Chen. Integrated and intelligent manufacturing: Perspectives and enablers. *Engineering*, 3(5):588–595, 2017.

[DWKN20] Robert Dietrich, Frank Winkler, Andreas Knüpfer, and Wolfgang Nagel. Pika: Center-wide and job-aware cluster monitoring. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 424–432, 2020.

[HBK05] Ben Huang, Michael Bauer, and Michael Katchabaw. Network performance in distributed hpc clusters. volume 2, pages 546–549, 01 2005.

[IRM+20] Mihailo Isakov, Eliakin del Rosario, Sandeep Madireddy, Prasanna Balaprakash, Philip Carns, Robert B. Ross, and Michel A. Kinsy. Hpc i/o throughput bottleneck analysis with explainable local models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020.

[KSV+21] Awais Khan, Hyogi Sim, Sudharshan S. Vazhkudai, Ali R. Butt, and Youngjae Kim. An analysis of system balance and architectural trends based on top500 supercomputers. In *The International Conference on High Performance Computing in Asia-Pacific Region*, HPC Asia 2021, page 11–22, New York, NY, USA, 2021. Association for Computing Machinery.

[PPG20] Ivy Peng, Roger Pearce, and Maya Gokhale. On the memory underutilization: Exploring disaggregated memory on hpc systems. In *2020 IEEE 32nd International Symposium on*

Computer Architecture and High Performance Computing (SBAC-PAD), pages 183–190, 2020.

[TRQR21] Erfan Bank Tavakoli, Michael Riera, Masudul Hassan Quraishi, and Fengbo Ren. Fspgemm: An opencl-based hpc framework for accelerating general sparse matrix-matrix multiplication on fpgas, 2021.