ENS 491 – Graduation Project (Design)
Project Proposal

Real-Time Supercomputer Monitoring

Ali Eren Ak, Deniz Batu, Muhammed Orhun Gale

Supervisor: Kamer Kaya

April 30, 2022

Abstract

An increase in architectural and operational complexities of High-performance Computing (HPC) clusters makes bottleneck detection harder in those systems since sufficient monitoring of HPC clusters is challenging. Even though there are various solutions for different bottlenecks, these solutions are applicable only after a diagnosis of the bottleneck. This project aims to establish a framework that can efficiently create a digital twin of a HPC cluster to gather and store data in a structured manner. Collected data will be used to train models that can detect bottlenecks in HPC clusters, understand performance characteristics and provide hardware/software insights for efficiency and energy consumption and perform prediction-based simulations of various hardware/software scenarios.

1 Introduction

In recent years, developments in High-performance Computing (HPC) which significantly improve the performance of HPC clusters, are accompanied by to increase in the architectural and operational complexity of HPC systems. [KSV+21] Because of the increased complexity, detecting and overcoming bottlenecks that occur becomes one of the crucial challenges in HPC. For instance, sparse data-oriented computation, which is one of the most important practices in HPC, has hardware and software-related bottlenecks. [TRQR21][BG09] Even though there exist a variety of solutions to overcome bottlenecks such as the implementation of different architectures, improving memory utilization, improving I/O utilization and enhancing network traffic, to make them applicable the first step is detecting the bottleneck. [IRM+20][PPG20][HBK05] Detection of the bottlenecks can be achieved via comprehensive monitoring of the HPC clusters with sophisticated tools to collect various metrics. An elaborate analysis of the collected metrics can uncover hardware and software-related alterations in performance and can be used to build Machine Learning models to establish a framework. However, the detection process, which includes monitoring, analysis, and learning overheads, may cause a burden that can overshadow the actual performance of HPC clusters.[DWKN20] In this sense, creating a digital twin of HPC clusters, which creates real-time digital models of clusters that make it possible to employ various tools, may provide flexibility for bottleneck detection and detection model production. [Che17] “Real-Time Supercomputer Monitoring” project 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.

1.1 High-Performance Computing

High-performance computing (HPC) is used to solve computation problems that would take infeasible amounts of time on singular commercial computers. HPC uses supercomputers, and computer clusters are made of many compute servers to combine the computing capabilities of many computation units such as CPUs.

In the last decade, the industry and academia focus has been shifted from making faster processors to clustering because the increased power consumption on faster processing units generates more heat which means the end of frequency scaling[ASF16]. Compute servers of clusters are networked to run algorithms and software programs simultaneously. The cluster is networked to the data storage to capture the output. It is critical for computing, network, and storage sub-systems to keep pace with each other for the HPC to work efficiently.

HPCs are used in many industries and academic research since they have the capabilities of processing huge amounts of data. Some application areas are machine learning, artificial intelligence, drug and gene research, computational fluid dynamics simulations, hydrocarbon exploration, and risk analysis. Due to the number of processors and sub-systems that need to work in harmony for performance efficiency, an accurate and detailed analysis of HPC systems is necessary. For increasing the performance of HPCs, the sub-system that causes the bottleneck needs to be detected; however, due to the complex structure of HPCs, detection becomes a considerable problem. Approaches, models, and applications have been developed for analyzing HPCs and their performances. For example, The Linpack Benchmark is used to grade performances of HPCs, which reflect the performance of a dedicated system for solving a dense system of linear equations[top] however, there are no industry standards for performance criteria or the analysis approach.

1.1.1 Performance Monitoring

There are a variety of tools that were developed by academia and companies for HPC monitoring with online/postmortem configurations over job/system/application layers. These tools help users and owners to analyze and adjust HPC performance. Most of these tools focus on specific layers and metrics of performance with some of them offering automated adjustments and some of them helping professionals to pinpoint the performance pitfalls while our project aims to collect data of all available, more than 4000, metrics in real time to detect hidden patterns that are infeasible to be analyzed without Machine Learning. Error handling and non-intrusive data collecting operations are critical aspects of such tools.

PIKA [DWKN20] is conceptually similar project to this project in the sense that it collects and visualizes run time metrics but only few select metrics are chosen. PIKA offers InfluxDB databases for storing temporary data without intrusion. NVIDIA GPU metrics can be acquired using the NVIDIA management library. To collect run-time data, collectd1 daemon is used. LIKWID is used to capture IPC, FLOPS, main memory bandwidth and power consumption via hardware counters. SLURM does the job scheduling; it is stated that Slurm plug-in architecture for node and job control (SPANK) could be used for this purpose.Writing to InfluxDB was found to be optimal using HTTP. The main difference between our project and PIKA is the number of metrics chosen to be analyzed and that our monitoring structure is Digital Twin.

Simulation-Based Performance Prediction of HPC [XIJ+20] has came to the following deductions and observations: Communication on supercomputers on node level, Message Passing Interface (MPI) is the industry standard. Every layer of a HPC should be modeled to achieve an acceptable accuracy. Analyzing source code is inefficient because the time consumption and the fact that with each update, the source code analysis needs to be repeated. Exponential increase of core counts in exascale causes need for modeling application performance to ensure continued scalability on future supercomputer architectures which our project aims to achieve with complete mapping of HPC systems.

Application Heartbeats [HES+10] describes a general use interface to monitor Autonomous Computing Environments performances. Their work proves the performance increasing and optimizing capabilities of Machine Learning and manual observer interference made possible by collecting performance data regularly with intervals called Heartbeats. Most applications on HPCs do not offer high-level data and analyzing performance over low-level data can yield inaccurate results in performance analysis. For example, counting the executed instructions over a period does not yield information about the actual work that was done; considering CPU utilization or cache miss rates causes similar problems. If an application achieves more performance than it requires, its resources can be allocated to other applications and if an application is under performing, more resources can be allocated for performance optimization.

1.1.2 HPC and Sparse Data

Sparse data is a term which refers to distribution of the informative data (for example having a non-zero value for a matrix or having an edge for a graph) of a data structure such as matrix, tensor or graph. Sparse computations, especially General sparse matrix-matrix multiplication (SpGEMM) and Sparse matrix-vector multiplication (SpMV), have fundamental functions in a large variety of HPC applications, such as solving large scale lineer systems, finite element simulations based on domain decomposition and parsing of context-free languages, and corresponds to the biggest portion of the computational cost. [BG08][TRQR21][BG09] There are several reasons why sparse computations have high computational cost. Tavakoli et al. argues that inadequate cache locality and regular chip memory access on CPUs and GPUs as well as loop-carried data dependencies of in-use algorithms cause performance limitations while Buluç and Gilbert identify data load imbalance among processors and low computation to communication ratio as performance challenges in sparse computation. [BG08][TRQR21]

1.2 Digital Twin

Digitalization is at the center of Industry 4.0 with recent technological developments in AI, IoT, Cloud Systems, and Digital Twin. Digital Twin, a relatively new and emerging concept compared to the others, was defined by Grieves [Gri14] in 2003 and later on introduced as a white paper in 2014. That describes Digital Twin as a virtual representation of what has been produced. There have been many unique definitions is evolved; however, a not yet standard description of a digital twin has been developed. Chen [Che17] defines a digital twin as a synchronized computerized model that can communicate with a physical device, share real-time data and represent operation status, position, environment situation, and knowledge of the physical device. Liu et al. [LMM18] focuses on predictive maintenance in the aerospace industry and sees the digital twin as a living model that forecasts the future of its physical counterpart. Even though some descriptions are domain-specific, they agree on the digital twin's functionality. It keeps information that is updated continuously to represent the current condition of the physical device.

Similar Concepts More digital conceptions are defined in the literature, such as digital models and digital shadow. They are mostly confused with digital twins; however, they have specific differences between each other and digital twins. A digital model is a generic definition of digital representations. The more specific feature is that there is no established data exchange between physical and digital devices. A digital model is just a representation of a physical device, and it can be seen as static, with no automatic data exchange. Similarly, a digital shadow is a digital representation that has a one-way data flow between a physical device and a digital shadow. One-way flow represents the physical object's changes, which means the digital shadow is connected with the physical object and follows changes in it. From the simulation point of view, this concept enables industries to create a digital representation of a product that can be used in data analytics, simulations, and marketing.

Although they are similar concepts, they have certain differences. The digital twin can be seen as the consequence of developments in information and communication technologies (i.e., IoT, 5G) and data-intensive technologies (i.e., AI and Big Data). It became cost-effective and feasible to leverage previous digital representations within the new concept, called digital twin [LMM18].

Use Cases In recent years, digital twins have been at the center of attention by academia and industry. Fuller et al. reviewed the literature and described three key application concepts for digital twin [FFDB20]. Firstly, developments in smart city technologies come with more connected communities. Increasing the number of IoT devices used throughout smart cities, there will be a considerable amount of data that enables future planning, development, and data analytics. More importantly, It will give us more insight about the city itself, such as how smart city utilities are distributed and consumed. Secondly, manufacturing concepts are worked mainly by top industry 4.0 companies. They are releasing patents, standards, and services about digital twins. Siemens uses digital twins for the power system, wastewater plant, and machine-human interface. The platform called MindSphere [Sie19] is developed to connect machines and physical infrastructure to a digital twin. The ready-to-use DT systems, open-source software, and platforms are being developed. The reason why the industry has attention to a digital twin is its ability to track, monitor, and optimize physical devices. It has the potential to give feedback about product-cycle such as machine performances, production line conditions, and device readabilities. As a third application case, healthcare systems in the digital twin are being studied. The digital twin can predict the effects of certain drugs under certain conditions. It also has potential for hospital management, predictive maintenance, and monitoring medical equipment [FFDB20]. However, there are more areas in which digital twins are applied, and more areas are being developed. Tao et al. stated that more than 14 application of digital twin is patented or released by academia and industry. With recent development, it will be much more and can be applied to different domains [TZLN18]. Current studies remain domain-specific, and no standard approach has been developed.

Machine Learning and Data Analytics Machine learning and AI algorithms have a crucial role in digital twinning. The role consists of getting insight from the digital twin, forecasting, and decision making. Rathore et al. gathered recent applications of digital twin and machine learning in a systematic survey [RSS$^+$21]. Machine learning applications in the digital twin help manufacturers by providing forecasts about degradation in entities. Early detection of anomalies could decrease faults and cracks in production. Furthermore, Lei et al. propose a digital twin system powered unmanned aerial vehicle (UAV) intelligent cooperation system where continuous feedback and forecasting mechanism is deployed between physical object and digital representation [LSZL20]. Digital twin might be very efficient in these cases where directly applying machine learning prediction to physical objects can cause risky and expensive outputs. Hence, first applying prediction to digital representation reduce risk and increases efficiency.

SuperTwin Structure A digital twin consists of nodes which are processing components of varying granularity of a HPC system. A node's granularity can range from CPU cores to CPU clusters. Non-uniform memory access(NUMA) is the memory access design of multiprocessing. In SuperTwin's architecture, two different approaches will be used. In one, NUMA domains will be established inside a node to collect metrics. These metrics will give detailed insight on per-CPU level to simulate the physical structure and the real-time performance of the Twin. In the other approach, nodes will be clustered as NUMA domains together with their communication and memory subsystems. This approach will give insight of inter-node communication and domain total processing performance. On the processor side, each domain will consist of sockets that consist of cores and threads per cores. On the memory side, domains will include the caches and their levels for each core which will yield information on the memory subsystem. Caches will be addressed per socket and per core. Both of these approaches will be used to represent the physical structure of a HPC system. When a SuperTwin is initialized, peak performances will be collected to establish a reference point for performance, core ID's will be given. Job instances, NUMA free space, latencies will be collected run-time to visualize performance. In the future, MPI system that works over the nodes will be integrated into the architecture as well.

[HTML]C0C0C0 Source	Source Description	[HTML]C0C0C0Metric	[HTML]C0C0C0Metric Description
/proc	Kernel statistics	kernel.all.intr	Context switches metric from /proc/stat
		kernel.all.pressure.cpu.some.total	Total time processes stalled for CPU resources
		kernel.all.pressure.memory.some.total	Total time processes stalled for memory resources
		kernel.all.pressure.memory.full.total	Total time when all tasks stall on memory resources
		kernel.all.pressure.io.some.total	Total time processes stalled for IO resources
		kernel.percpu.interrupts.PMI	Performance monitoring interrupts for each core
		kernel.percpu.interrupts.TRM	Thermal event interrupts for each core
		kernel.percpu.interrupts.line*	Number of interrupts caused by each IO device
/proc/meminfo	System memory statistics	mem.util.used	Used system memory
		mem.util.free	Free system memory
		mem.util.directMap4k	Amount of memory that is directly mapped in 4kB pages
		mem.util.directMap2M	Amount of memory that is directly mapped in 2MB pages
		mem.util.directMap1G	Amount of memory that is directly mapped in 1GB pages
		swap.pagesin	Pages read from swap devices due to demand for physical memory
		swap.pagesout	Pages written to swap devices due to demand for physical memory
	NUMA statistics	mem.numa.util.free	Per-node free memory
		mem.numa.util.used	Per-node used memory
		mem.numa.alloc.hit	Per-node count of times a task wanted alloc on local node and succeeded
		mem.numa.alloc.miss	Per-node count of times a task wanted alloc on local node but got another node
		mem.numa.alloc.local_node	Per-node count of times a process ran on this node and got memory on this node
		mem.numa.alloc.other_node	Per-node count of times a process ran on this node and got memory on another node
/proc/vmstat	Virtual memory statistics	mem.vmstat.kswapd_low_wmark_hit_quickly	Count of times low watermark reached quickly
		mem.vmstat.kswapd_high_wmark_hit_quickly	Count of times high watermark reached quickly
/proc/net/dev	Network interface statistics	network.interface.in.bytes	Network recv read bytes per network interface
		network.interface.out.bytes	Network send bytes per network interface
/proc/diskstats	Disk statistics	disk.dev.read	Per-disk read operations
		disk.dev.write	Per-disk write operations
		disk.dev.read_merge	Per-disk count of merged read requests
		disk.dev.write_merge	Per-disk count of merged write requests
/proc/<pid>/*	Per process statistics	proc.psinfo.ngid	NUMA group identifier
		proc.psinfo.threads	Number of threads
		proc.psinfo.nvctxsw	Number of non-voluntary context switches
		proc.psinfo.processor	Last CPU the process was running on
		proc.psinfo.cmaj_flt	Count of page faults other than reclaims of all exited children
		proc.psinfo.maj_flt	Count of page faults other than reclaims
		proc.io.wchar	write(), writev() and sendfile() send bytes
		proc.io.rchar	read(), readv() and sendfile() receive bytes

[HTML]C0C0C0 [HTML]C0C0C0Hardware Counter	Metric	Metric Description
CPU_CLK_UNHALTED.THREAD	Cycles	Counts the number of core cycles while the logical processor is not in halt state
MEM_INST_RETIRED.ALL_LOADS	Loads	Counts the number of retired loads
MEM_INST_RETIRED.ALL_STORES	Stores	Counts the number of retired stores
L1D.REPLACEMENT	L1 Data Misses	Counts the data line replacements that occur on L1D cache
LLC_REFERENCE - L2-RQSTS.CODE_RD_MISS	L2 Data Misses	Number of data requests that miss L2D cache. Corresponds to the difference between every core request that references a cache line in LLC and the L2 code misses
CAS_COUNT.RD+CAS_COUNT.WR	LLC Misses	Sum between all DRAM reads and all DRAM writes
CYCLE_ACTIVITY.STALLS_L1D_MISS	L1 Data Stalls	Stalls that occur due to outstanding loads that miss L1D cache
CYCLE_ACTIVITY.STALLS_L2_MISS	L2 Stalls	Stalls that occur due to outstanding loads that miss L2 cache
CYCLE_ACTIVITY.STALLS_L3_MISS	L3 Stalls	Stalls that occur due to outstanding loads that miss L3 cache
CYCLE_ACTIVITY.STALLS_MEM_ANY	Memory Stalls	Stalls that occur due to outstanding loads in the memory subsystem
CYCLE_ACTIVITY.CYCLES_L1D_MISS	Cycles with misses on L1 Data	Cycles while there are outstanding loads that miss L1D cache
CYCLE_ACTIVITY.CYCLES_L2_MISS	Cycles with misses on L2	Cycles while there are outstanding loads that miss L2cache
CYCLE_ACTIVITY.CYCLES_L3_MISS	Cycles with misses on L3	Cycles while there are outstanding loads that miss L3 cache
CYCLE_ACTIVITY.CYCLES_MEM_ANY	Cycles with outstanding loads	Cycles while there are outstanding loads in the memory subsystem
FP_ARITH_INST_RETIRED.SCALAR_DOUBLE	FP Scalar Double	Double-precision scalar FP instructions
FP_ARITH_INST_RETIRED.SCALAR_SINGLE	FP Scalar Single	Single-precision scalar FP instructions
FP_ARITH_INST_RETIRED.128B_PACKED_DOUBLE	FP 128-bit SIMD Double	Double-precision 128-bit packed FP instructions
FP_ARITH_INST_RETIRED.128B_PACKED_SINGLE	FP 128-bit SIMD Single	Single-precision 128-bit packed FP instructions
FP_ARITH_INST_RETIRED.256B_PACKED_DOUBLE	FP 256-bit SIMD Double	Double-precision 256-bit packed FP instructions
FP_ARITH_INST_RETIRED.256B_PACKED_SINGLE	FP 256-bit SIMD Single	Single-precision 256-bit packed FP instructions
FP_ARITH_INST_RETIRED.512B_PACKED_DOUBLE	FP 512-bit SIMD Double	Double-precision 512-bit packed FP instructions
FP_ARITH_INST_RETIRED.512B_PACKED_SINGLE	FP 512-bit SIMD Single	Single-precision 512-bit packed FP instructions

Table 1: Kernel performance metrics from /proc and hardware counter metrics that used model sparse computation performance.

2 Proposed Solution and Methods

High-performance Computing (HPC) cluster's are expensive machines in terms of both while building and operating for vendors and HPC clusters are employed for modeling and solving complex systems/problems of customers. [KSV+21] Therefore, any other task that may obstruct the executed task is risky to execute from both vendor and customer's business plans. However, again for both, detecting bottlenecks in hardware/software and developing solutions to problems is crucial. Moreover, even if the possible bottlenecks are detected, halting the operations for trying out hardware improvements is costly since cluster will not be available and to be able to try new configurations the specific component must be accessible physically. Therefore, having a digital twin of the HPC cluster which will store the status and performance data in a structured manner that would enable vendors and customers to monitor the system efficiently, train machine learning models with the cluster data, which may reveal hidden patterns which provide performance insights and evaluate new hardware configurations could be a promising facility. In this sense, the framework that will be established in this project promises creating digital twins of HPC clusters and providing performance insights such as existing bottlenecks via using machine learning models and prediction-based simulations.

Building the framework which is aimed by this project can be described as a complex problem for various reasons. First of all, designing and implementing proposed features of this system, which are detailed monitoring of a HPC cluster, building a digital twin of it and creating models that will detect anomalies and perform prediction-based simulations, involve in wide-ranging technical and engineering issues such as database systems to create and manage digital twins, operating systems to gather and interpret cluster data, machine learning to create detective and predictive models, parallel programming and high-performance computing. Next, there is no canonical way of building such a system. There are various tools to create digital twin, daemons to gather cluster data, algorithms and approaches for anomaly detection. This leads to the third reason, to be able to find and design the suitable system, a detailed research and development process must be conducted. Then, this problem has two major stakeholders, vendors and customers, who have clashing interests. A vendor value a sustainable, high throughput and energy efficient (which decreases the operation cost) system therefore, she may dare to have a digital twin that cause overhead. However, a customer most probably does not want to have any overheads while her task is executed. Lastly, solving this problem would have significant consequences on HPC clusters' environmental effects and sustainability since bottleneck detection would increase energy efficiency of clusters and prediction-based simulation enable vendors to try various hardware configurations without purchasing them.

HPC monitoring is a complex and important issue as of today and as computing technologies progress. Collecting data from HPC systems is a complicated task itself as performance metrics are not readily available on the layers and units of the HPC architecture. Data needs to be extracted from hardware units without creating significant tension on the system. To achieve this the general approach is to export the data from components with as little computation as possible. The extracted minimal data is then transferred to applications to be reconstructed and moved to the database. We aim to extract IPC, FLOPS, main memory bandwidth and power consumption via hardware counters using LIKWID. Extracted data will be stored on InfluxDB database. HTTP will be used for data transfer. Message Passing Interface (MPI) will be used for node-to-node communications. This project aims to collect all possible performance monitoring metrics of HPC systems for run-time and offline observation and analysis. Collected data will be visualized on a Digital Twin for real time observer interference and later trained using Machine Learning algorithms to detect performance bottlenecks and crucial metrics. ClusterCockpit[EGA+19] web application and its specific node agents might be used for collecting, monitoring, and forwarding performance metrics.

Digital twin implementation in high-performance computing is challenging. The first challenge is making the digital twin of the HPC cluster proceed with essential functionalities such as the job-scheduling mechanism, relationship between compute nodes and home nodes, each hardware of the cluster, etc. As it is stated, there is no standard digital twin implementation method developed; current studies have domain-specific approaches and will not work in the HPC domain. Azure Digital Twin Description Language (DTDL) is a platform as a service (PaaS) helping customers to create digital models of physical environments using graphs based approach. In twin graphs, each node represents a digital object which connects other digital objects with edges. Edges could be defined with relations. Azure proposes a DTDL backed with JSON-LD, a unique JSON format for linked data. The sample JSON file provided by the Azure team can be seen below.

```
{
  "@id": "dtmi:com:adt:dtsample:home;1",
  "@type": "Interface",
  "@context": "dtmi:dtdl:context;2",
  "displayName": "Home",
  "contents": [
    {
      "@type": "Property",
      "name": "id",
      "schema": "string"
    },
    {
      "@type": "Relationship",
      "@id": "dtmi:com:adt:dtsample:home:rel_has_floors;1",
      "name": "rel_has_floors",
      "displayName": "Home has floors",
      "target": "dtmi:com:adt:dtsample:floor;1"
    }
  ]
}
```

In the HPC context, a graph-based approach could be useful. JSON-LD gives the flexibility to create any digital object while preserving relations, fields, and context. Azure DTDL offers 5 model fields to identify the type of digital information. For example, the telemetry field, which represents measurements or events, can handle instant events in HPC such as running jobs, hardware metrics, etc. The component field is used to create a hierarchy between interfaces. This field can be used to represent the relation between compute nodes and their hardware units. In addition to Azure DTDL, Twinbase, which is an open-source platform to manage and distribute digital twin documents using git protocol, is also using JSON-LD format to link digital documents [AST21]. The JSON-LD format looks proper in this case. We plan to define our ontology and digital twin format to create HPC digital twin called SuperTwin.

2.1 Objectives/Tasks

The project's objective is to create a framework for monitoring HPC clusters by collecting hardware unit metrics, which will help in the prediction and detection of bottlenecks and other hardware-related issues. In addition to monitoring, the project proposes different metrics to reveal hidden patterns using machine learning algorithms. Moreover, the digital twin will represent HPC in a graph-based structure to empower our understanding of supercomputers. There are four challenging steps consisting of collecting hardware data via back-end software, visualizing the metrics using front-end frameworks, learning patterns from data using machine learning algorithms, and developing a digital twin of the cluster:

1. To develop a back-end software, we need to gather all hardware metrics related to HPC overall performance and cluster overload. In the back-end part, there are extra challenges to be considered. For example, the proper database must be chosen to store different data types i.e., time-series data. The system's efficiency should also be considered because HPC clusters are designed to provide high computing power for large teams. There can not be any space and time complexity bottleneck, which is additional overload.

2. The gathered data should be visualized and shown well in the front-end part. Data visualization is a challenging task; the visualizations should be easily interpreted for analytic purposes, and the user interface should be smooth and responsive. Multiple methods should be tried, and a proper one should be chosen to increase user experience.

3. A machine learning pipeline will be developed to detect anomalies and predict performances. To build an accurate pipeline, correct features should be extracted, a proper algorithm should be chosen, and the machine learning model should be correctly evaluated

4. The digital twin will help to construct a digital representation of a supercomputer. Without applying machine learning predictions to the physical supercomputer, the digital twin can be used to observe possible outcomes. Therefore, the digital twin should be developed in a way that it can represent physical supercomputers correctly. All dependencies and relations between hardware components should be represented. Each part of this project will help us understand supercomputers, detect anomalies in runs, and predict performances for future jobs.

2.2 Realistic Constraints

This project requires access to a HPC system whose users accept sharing metadata, job description, and cluster information with our project team for real world development and testing of our monitoring tool and sub-systems. This is a concern since users such as companies might not want to share inside knowledge with us due to security concerns. Changes in HPC architecture, black-box systems, new chip models and the general evolution of HPC systems might render our tools obsolete. For example, a new chip model that does not have hardware counters will render us unable to collect metadata from it. We aim to develop our data acquisition tool such that it checks for available metrics in systems and pulls the metric data to the digital twin.

2.3 Engineering/Scientific Standards

To create a system that is appropriate to use in any HPC cluster, the monitoring tool's computational cost must not slow down the normal operations of the HPC system that it is monitoring more than %2. All possible metrics must be collected for the identification of significant metrics before deciding on which metrics the digital twin of a HPC cluster would require. Data collection must at least be done on the processor (CPU, GPU) level of a node and all nodes must be accessible by the Digital Twin. Also, collected performance metrics must be visualized for observers on run-time. Collected data must be convertible to the proper format for developing and training machine learning algorithms.

3 Risk Management

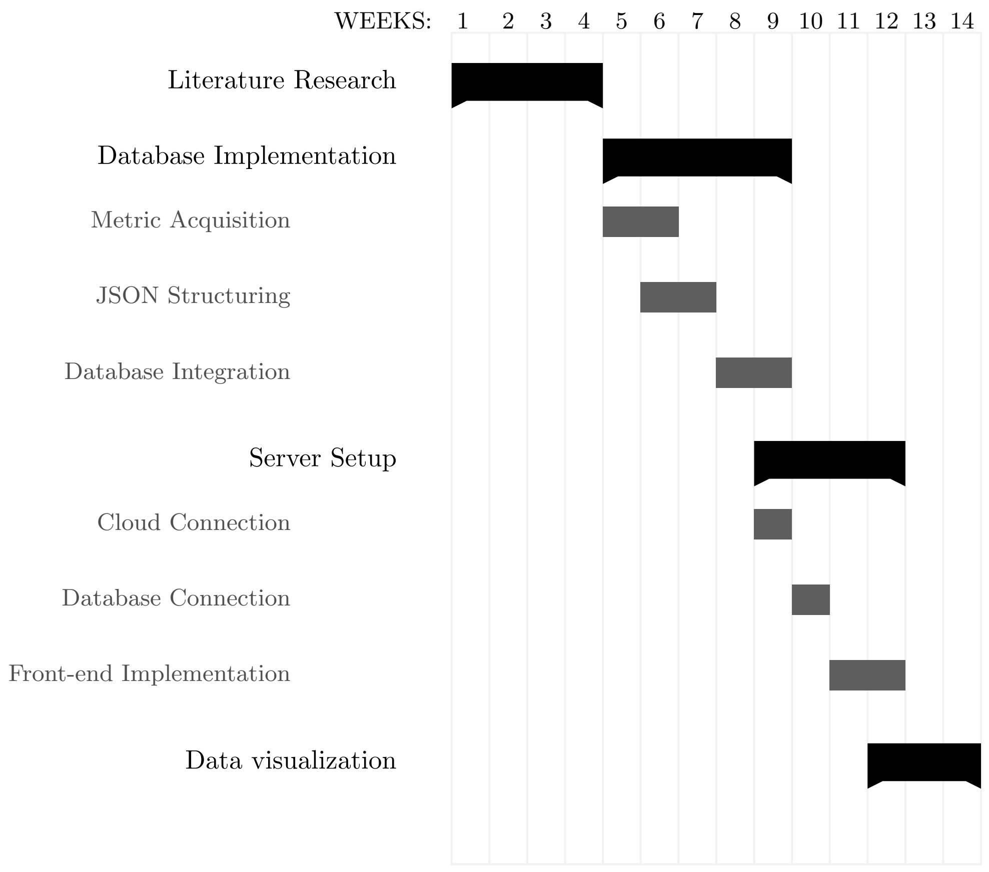
For this project two main risks are identified which are resource-related problems and technical challenges that would hinder the progress of the project. Resource-related problems are mainly due to the availability of the HPC clusters. TRUBA is providing us access to a HPC system, in case the system becomes unavailable to us, we can ask our SparCity partners for HPC access.

Technical challenges that would emerge during the project would be caused by mainly the maturity of the technologies that will be used in the system. Since the digital twin concept and related technologies are recent, insufficient facilities regarding these technologies would affect or even block the various parts of the process. In case our expertise on a topic hinders our progress, we can ask our project supervisor and our partners in SparCity for support. Also, we can develop custom software according to the needs of the project.

4 Project Schedule

In the first 4 weeks literature research was done on the topics of digital twins, HPC and, sparse data. Database implementation will finish at the 9th week. Implementation consists of metric acquisition, JSON structuring, database integration. The metrics were collected and stored in the JSON structures designed by us. Starting from this week(week 8), database integration will be completed in 2 weeks to store JSON data. Server setup process will be finished at 12th week. Server setup consists of cloud connection, database connection, front-end implementation. Lastly, while the server is running and database is connected, visualization process will begin at week 11 and last until the project progress is reported at the end of week 14.

Each project step will be led by one of the group members. Other group members should be organized to keep track of project steps and respond to the corresponding task. Database implementation task will be led by Ali Eren Ak; server setup task will be led by Muhammed Orhun Gale; data visualization task will be led by Deniz Batu.



5 Ethical Issues

The project is planning to develop in a way that it can be used in any supercomputer in the world. It will be open-source licensed and we will release our findings transparently. It is a non-profit software, no patent protected designs and software will be used. During this project, we will use other open-source projects/concepts or develop our own software. In short, there are not any ethical concerns.

References

[ASF16] 2016 1:22 am UTC Ars Staff Feb 11. Parallel computing research at illinois the upcrc agenda, Feb 2016.

[AST21] Juuso Autiosalo, Joshua Siegel, and Kari Tammi. Twinbase: Open-source server software for the digital twin web. *IEEE Access*, 9:140779–140798, 2021.

[BG08] Aydın Buluç and John R. Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. *2008 37th International Conference on Parallel Processing*, pages 503–510, 2008.

[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.

[EGA$^+$19] Jan Eitzinger, Thomas Gruber, Ayesha Afzal, Thomas Zeiser, and Gerhard Wellein. Clustercockpit — a web application for job-specific performance monitoring. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–7, 2019.

[FFDB20] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. Digital twin: Enabling technologies, challenges and open research. *IEEE access*, 8:108952–108971, 2020.

[Gri14] Michael Grieves. Digital twin: manufacturing excellence through virtual factory replication. *White paper*, 1:1–7, 2014.

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

[HES$^+$10] Henry Hoffmann, Jonathan M. Eastep, Marco D. Santambrogio, Jason E. Miller, and Anant Agarwal. Application heartbeats: a generic interface for specifying program performance and goals in autonomous computing environments. In *ICAC '10*, 2010.

[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.

[LMM18] Zheng Liu, Norbert Meyendorf, and Nezih Mrad. The role of data fusion in predictive maintenance using digital twin. In *AIP conference proceedings*, volume 1949, page 020023. AIP Publishing LLC, 2018.

[LSZL20] Lei Lei, Gaoqing Shen, Lijuan Zhang, and Zhilin Li. Toward intelligent cooperation of uav swarms: When machine learning meets digital twin. *IEEE Network*, 35(1):386–392, 2020.

[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.

[RSS$^+$21] M Mazhar Rathore, Syed Attique Shah, Dhirendra Shukla, Elmahdi Bentafat, and Spiridon Bakiras. The role of ai, machine learning, and big data in digital twinning: A systematic literature review, challenges, and opportunities. *IEEE Access*, 9:32030–32052, 2021.

[Sie19] AG Siemens. Mindsphere the cloud-based, open iot operating system for digital transformation, 2019.

[top] top500.org. The linpack benchmark.

[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.

[TZLN18] Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415, 2018.

[XIJ+20] Gen Xu, Huda Ibeid, Xin Jiang, Vjekoslav Svilan, and Zhaojuan Bian. Simulation-based performance prediction of hpc applications: A case study of hpl. In *2020 IEEE/ACM International Workshop on HPC User Support Tools (HUST) and Workshop on Programming and Performance Visualization Tools (ProTools)*, pages 81–88, 2020.