LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-821258

A Holistic View of Memory Utilization on HPC Systems:Current and Future Trends

I. B. Peng, I. Karlin, M. B. Gokhale, K. Shoga, M. Legendre, T. Gamblin

April 8, 2021

The International Symposium on Memory Systems (MemSys'21)
Virtual, DC, United States
October 1, 2021 through October 1, 2021

Disclaimer

A Holistic View of Memory Utilization on HPC Systems: Current and Future Trends

Ivy B. Peng*
peng8@llnl.gov
Lawrence Livermore National Laboratory
USA

Ian Karlin
karlin1@llnl.gov
Lawrence Livermore National Laboratory
USA

Maya B. Gokhale
gokhale2@llnl.gov
Lawrence Livermore National Laboratory
USA

Kathleen Shoga
Shoga1@llnl.gov
Lawrence Livermore National Laboratory
USA

Matthew Legendre
legendre1@llnl.gov
Lawrence Livermore National Laboratory
USA

Todd Gamblin
gamblin2@llnl.gov
Lawrence Livermore National Laboratory
USA

ABSTRACT

Memory subsystem is one crucial component of a computing system. Co-designing memory subsystems becomes increasingly challenging as workloads continue evolving on HPC facilities and new architectural options emerge. This work provides the first large-scale study of memory utilization with system-level, job-level, temporal and spatial patterns on a CPU-only and a GPU-accelerated leadership supercomputer. From system-level monitoring data that spans three years, we identify a continuous increase in memory intensity in workloads over recent years. Our job-level characterization reveals different hotspots in memory usage on the two systems. Furthermore, we introduce two metrics, 'spatial imbalance' and 'temporal imbalance', to quantify the imbalanced memory usage across compute nodes and throughout time in jobs. We identify representative temporal and spatial patterns from real jobs, providing quantitative guidance for research on efficient resource configurations and novel architectural options. Finally, we showcase the impact of our study in informing system configurations through an upcoming NNSA CTS procurement.

CCS CONCEPTS

• **General and reference** → **Metrics**; **Measurement**; *Performance*;
• **Computer systems organization** → *Grid computing*.

KEYWORDS

HPC, system-wide characterization, memory systems, memory characterization, large-scale characterization

1 INTRODUCTION

System co-design of future HPC facilities is increasingly challenging because workloads evolve fast while new architectural options emerge. Therefore, insights derived from realistic system-level data significantly impact system designs and procurement decisions [11]. Previous studies have characterized system-wide I/O, power, memory, and noise on leadership facilities [12, 15, 17, 27]. However, a holistic view of memory utilization is still missing on large-scale HPC systems. Today, many leadership facilities have moved from CPU-only to GPU-accelerated architecture. This work provides a timely and updated view of the current state of memory utilization on HPC systems and its implications for future research opportunities.

Memory subsystem directly impacts the performance and cost of an HPC system. It could account for more than 20% of the total system cost and influences job configurations and efficiency. Further scaling up memory capacity per compute node could be limited by in area, power and cost [21]. Meanwhile, new memory technologies [8, 16, 18–20, 23] continue emerging, expanding the design space. Strategically, we believe that understanding the spectrum of workloads on a system can lead to more efficient architectural choices. Previous works used job logs to understand the jobs' peak and average usage but lack the details of spatial and temporal memory imbalance inside individual jobs. Other works perform detailed profiling of selected applications but lack the quantification of their impact on overall system throughput [25, 31]. For more disruptive architectural changes on future systems, such as disaggregated architecture and dynamic resource configuration, both system-level impact and intra-job memory usage characteristics are crucial for guiding design space exploration.

Today, many leadership facilities have deployed high-resolution system monitoring and tracking infrastructures [13, 14, 17]. Node-level monitoring provides fine-grained information that complements job-level logs. However, there are two main challenges in leveraging the monitoring data. First, the infrastructures provide overwhelming details about a system, and thus, standard metrics are needed to distill information into a normalized high-level basis for comparison across systems. Second, findings on a system reflect the current state, and thus, there requires a systematic way that bridges the current state and future systems. One possible solution is to form an abstraction of utilization patterns from detailed monitoring data and adapt the weight of each pattern to reflect the projection of their composition on a future system.

In this work, we focus on understanding the current state of memory utilization on HPC systems. Therefore, we choose the Lassen supercomputer to represent GPU-accelerated systems and

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

the Quartz supercomputer to represent CPU-based systems. We deploy a system monitoring infrastructure called *Sonar* to collect high-resolution utilization data on the systems. Our system-level characterization indicates that memory intensity in workloads steadily increases in the past three years. We further classify jobs based on their memory intensity (i.e., peak memory usage) and decompose the total system load into different patterns of memory intensity. Although only 10% jobs have moderate to intensive memory intensity, they can account for 30 - 35% of the total system load.

Our characterization of the job-level utilization shows that jobs on Lassen have a hotspot in their memory usage, at 60 GB, which is close to the GPU memory capacity on a node with four GPUs. This finding indicates a massive underutilization of the host memory, which is 256 GB on this GPU-based system. We also compare the correlation between job characteristics and memory utilization on the two systems. While peak memory usage is positively correlated with job sizes (i.e., the number of allocated nodes and time) on Quartz, the correlation is much weaker or even negative on Lassen.

To the best of our knowledge, we provide the first quantitative study of spatial and temporal memory imbalance in jobs on production HPC systems. We propose two metrics, *spatial imbalance* and *temporal imbalance*, to quantify the imbalanced memory usage across compute nodes and time in a job. We identify four temporal patterns – *constant, phased, dynamic and sporadic*, from real jobs. The decomposition of system load into these temporal patterns shows that most jobs have a low temporal imbalance, but their peak usage could be only 10% of the provisioned memory capacity. We also identify three spatial patterns – *batched, grouped, and centralized* – and show that imbalanced usage on a few nodes in a job cause severe underutilization of resources allocated to that job due to homogeneously configured nodes (i.e., all nodes are configured with the same resources) on current HPC systems.

We demonstrate the impact of our study in informing system configurations in an upcoming NNSA CTS procurement of a new system. The results of our analysis are used in procurement decisions considering cost, capacity, and system throughput. We also demonstrate the implication of usage patterns discovered in real jobs for driving emerging system architecture. For instance, workloads with a sporadic temporal pattern may have insignificant performance loss if they leverage disaggregated memory for temporary memory expansion. Also, heterogeneous subclusters of nodes that are configured with different resources may address high spatial imbalance in jobs. Our contributions in this work are summarized as follows:

- the first holistic view of memory utilization at system-level, job-level, temporal and spatial characteristics on CPU-based and GPU-accelerated leadership HPC systems.
- a new metric for quantifying temporal memory imbalance in jobs. We identify four representative temporal patterns on production systems – *constant, phased, dynamic and sporadic* patterns.
- a new metric for quantifying memory imbalance across compute nodes inside jobs. We identify *batched, grouped, and centralized* as the three representative spatial patterns in real jobs.
- Identify the difference in correlation between memory usage and job characteristics on a CPU-based and a GPU-accelerated system. Temporal and spatial imbalances are negatively correlated with job size on the GPU-based system, but positively correlated on the CPU-based system.
- We demonstrate the impact of our study in informing future procurement decisions and emerging system architectural options.

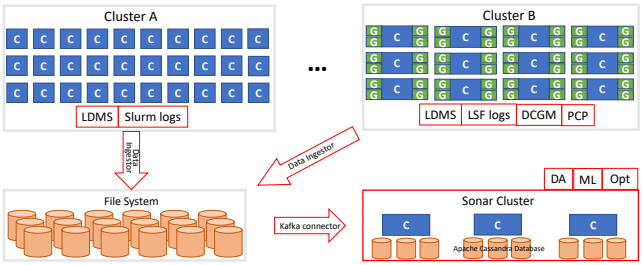


Figure 1: An overview of Sonar monitoring infrastructure. Red boxes indicate the main components.

2 METHODOLOGY

2.1 Data Acquisition

We used the *Sonar monitoring infrastructure* for collecting fine-grained monitoring data from production clusters. Sonar is a central system-monitoring infrastructure deployed at Livermore Computing (LC). Its main components include sample monitoring, data ingestion, staging, archive, and analytics. These components are located at different tiers of the computing infrastructure, as indicated in red boxes in Figure 1. Depending on the architecture of the monitored cluster, different modules are configured at the frontend for sampling. For instance, the lightweight distributed metric service (LDMS) [2] is used to collect system-wide metrics, such as memory usage, I/O read and write sizes, power consumption, and network traffic. On GPU-accelerated clusters, Sonar uses NVIDIA Data Center GPU Manager (DCGM) and Performance Co-Pilot (PCP) to gather GPU-side information, such as GPU utilization and NVlink traffic. Sonar collects job metadata from the respective job scheduling systems on the cluster.

Sonar uses in-house data ingestors to stage samples from the frontend components into the global storage. The raw data is then pre-processed and transformed through Kafka data connectors from the global storage to the Sonar backend. The data staging and ingestion jobs run daily in the background. The Sonar backend consists of a cluster of four compute nodes, each with massive node-local storage. A Cassandra database is deployed over these nodes to provide a central data repository. The massive information captured in the Sonar infrastructure can provide a complete picture of the complex behaviors of various production supercomputers at the system, job, and application levels. On top of the central data repository, multiple data analytics, system monitoring, and optimization efforts can be developed to leveraging the insights from realistic system behaviors. In this work, we leverage the fine-grained monitoring data to understand memory utilization on large-scale HPC systems with temporal and spatial details of each job, which was unachievable in previous works.

2.2 Production HPC Systems in Study

In this study, we focus on two production systems that represent state-of-the-art GPU-accelerated and CPU-only supercomputers.

The Lassen supercomputer is a GPU-accelerated IBM Power system deployed at Lawrence Livermore National Laboratory in 2018 [4]. The system was ranked top 10th in 2019 and is currently ranked 17th in the top 500 list [24]. It is a scale-down open-access version of the pre-exascale Sierra system under the CORAL initiative. Thus, the Lassen system has identical system architecture with the Sierra supercomputer [26]. Lassen has a total of 795 compute nodes interconnected with 2:1 tapered fat-tree network. Each node features two IBM Power9 CPUs (44 cores) and four Nvidia Volta GPUs. In total, the system delivers a peak computing power of 23,047,200 GFlop/s from the 34,848 CPU cores and 3,168 GPUs. The memory subsystem includes 16 GB HBM2 memory per GPU and 256 GB DRAM main memory per node. In total, each node has 320 GB of memory that is globally addressable with hardware-supported coherence from NVLink. Jobs on Lassen are scheduled by the IBM LSM system.

The Quartz supercomputer is a CPU-only system [5]. It was deployed at the Lawrence Livermore National Laboratory in 2016. Each compute node features two Intel Intel Xeon E5-2695 v4 processors and 36 CPU cores. In total, the system consists of 3,018 compute nodes and uses the Cornelis networks Omni-Path interconnect. The system delivers a peak performance of 3,251 TFLOPS from a total of 108,648 CPU cores. Each node has a memory capacity of 128 GB and a peak memory bandwidth of 77 GB/s. In total, the system provides 344,064 GB of memory. The system uses SLURM for job scheduling. Quartz represents CPU-based HPC systems that are popular with workloads not matched to GPU computation.

2.3 Data Processing and Analysis

Sonar infrastructure collects system monitoring data from the compute nodes of Lassen and Quartz every second. We use the total memory and free memory samples to deduce the memory usage on a node at a time point. About 172 million and 60 million daily samples are collected from Quartz and Lassen, respectively. Lassen and Quartz execute 1k to 10k jobs on average every day. For each job, we collect the job name, user group, job partition, job start and end time, and the list of compute nodes assigned to the job. We used the dataset from 2019 to 2021 for the Quartz system and from 2020 to 2021 for the Lassen system. We find that system characteristics evolve over the years, and for a fair comparison across the two systems, we use the same three-month period in 2021. We also present a summary of system-level changes over the years in Section 3.

We correlate node-level samples with job information by the list of assigned nodes and time period to reconstruct each job's fine-grained temporal and spatial behaviors. These monitoring samples may contain corrupted or redundant data points, e.g., system down periods. We performed data quality checks in each period and compared them with the system architecture to ensure that the data used in the analysis are valid. As we are interested in production workloads on the systems, we excluded samples from nodes in the debug partition and jobs that run shorter than one minute. Although job logs could also report peak and average memory usage per job, we use the peak and average usage reconstructed from temporal and spatial behaviors of each job for a fair comparison across the system level and job level.

The monitoring infrastructure generates massive datasets because a single job could generate tens of thousands of data points. Analyzing these datasets to distill high-level insights is a challenging task. Therefore, we devise an interval-peak extraction process to coarsen the granularity into user-defined intervals while maintaining the critical information in time and space. In particular, all data points on a node during a job are binned into intervals of a fixed duration. The data points in each bin are then extracted to the maximum value. We repeat this process on all nodes in a job. We chose to use peak value instead of the average or median value for extraction because resource provision has to satisfy the peak usage. We chose the interval of one minute because jobs on the systems have a maximum 24 hours runtime and one-minute intervals retain sufficient temporal details while limiting data points to reasonable sizes.

2.4 Statistical Methods

We employ several statistical methods to derive meaningful insights from the monitoring data. In addition to common metrics such as mean and standard deviation, we also used probability density function (PDF), cumulative density function (CDF), and correlation coefficients in this work. PDF represents the probability of a random variable X to take value x, while CDF represents the probability of random variable X to have a value equal to or less than x. In this study, we use PDF to identify hotspots in memory usage. We use CDF to quantify the composition of different job groups on the two systems, e.g., jobs classified as low memory intensity compose Y% of the total system load.

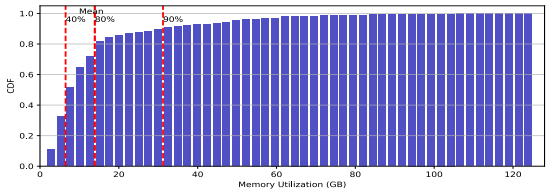
We use the Pearson correlation coefficient to identify potential relationships between two variables, such as memory usage and job sizes, including the number of allocated nodes and job duration. We compare the correlation coefficient on the two HPC systems to uncover their difference in system characteristics. The correlation coefficient (Pearson's r value) quantifies the likelihood of a linear relationship between two random variables. The calculation of the correlation coefficient of two random variables, X and Y, is defined in Equation 1:

$$r(XY) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2(y_i - \bar{y})^2}}, \quad (1)$$

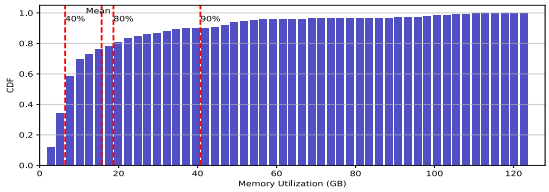
where N is the number of samples, and $\bar{x}$ and $\bar{y}$ represent the mean value of X and Y, respectively. The Pearson's r value is bound to [-1, 1]. A positive value indicates a positive correlation, i.e., the value of the i-th sample in Y always increases when the value of the i-th sample in X increases. A negative r value indicates that the value of the i-th sample of Y always decreases when the value of the i-th sample in X increases. A zero r value indicates no correlation.

3 SYSTEM-LEVEL CHARACTERIZATION

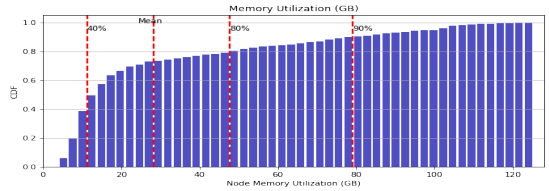
This section provides an overview of the system-level utilization that quantifies memory usage on all nodes during a period. The memory usage includes both user-space and OS memory usage on a node. This analysis aims to understand how the spectrum of workloads on current production systems utilize memory resources



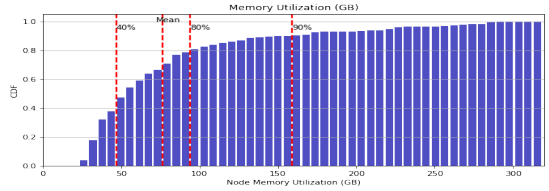
(a) The Quartz system in 2019.



(b) The Quartz system in 2020.

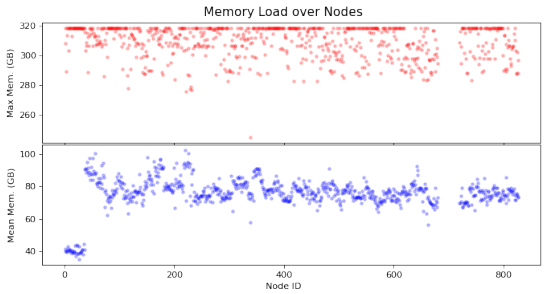


(c) The Quartz system in 2021.

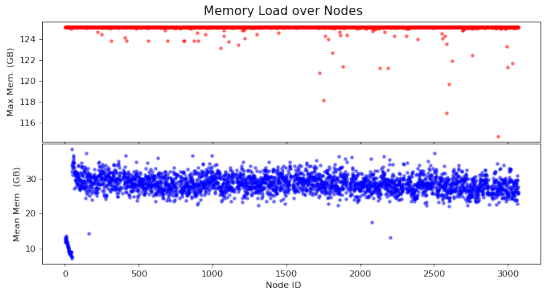


(d) The Lassen system in 2021.

Figure 2: The CDF of memory usage on nodes on Quartz and Lassen. The red dotted lines indicate the 40th, mean, 80th, and 90th percentile. Lassen has 320 GB and Quartz has 128 GB memory per node.



(a) The Lassen system.



(b) The Quartz system.

Figure 3: The mean and max memory usage on all compute nodes ordered by node ID.

and what characteristic differences the GPU-based and CPU-based systems exhibit at the system level.

Figure 2 summarizes the distribution of memory usage on Lassen and Quartz supercomputers. We present the trend of memory usage on Quartz from 2019, 2020, to 2021. In the rest of the paper, we use the monitoring data in 2021 Q1 for analysis to provide the most update-to-date view.

To understand the trend in memory usage over the years, we compare the memory usage on Quartz from 2019 to 2021 (Figure 2a, 2b, 2c). In these years, the 90th percentile of memory usage steadily increases from 30 GB, 41 GB to 110 GB. The 80th percentile of memory usage also increases, from 15 GB, 20 GB to 78 GB. The continuous increase in memory usage is consistent with the observation that more data-intensive workloads are emerging on HPC systems. Changes on the same machine over the years indicate that the state of a system could evolve rapidly, which needs to be factored in system designs.

Both Lassen and Quartz have their 80th percentile in memory usage at about 80GB (Figure 2d and 2c), indicating that for 80% of the time, nodes use less than 80GB memory. Because Lassen has 320 GB memory per node, while Quartz nodes only have 128 GB, their utilization rates differ significantly. In particular, Lassen has a 25% utilization rate, and Quartz has a 61% utilization rate. Both systems experience heavy memory usage for about 10% of time when Lassen nodes use more than 150 GB memory, and Quartz nodes use more than 110 GB memory.

The two systems significantly differ at their 40th percentile in memory usage, where Lassen nodes use a minimum of 30-50 GB memory while Quartz nodes use merely 5-10 GB memory. Quartz nodes have a much wider range of memory usage than Lassen nodes. At the 50th percentile, Lassen nodes use 30 - 80 GB memory, while the usage on Quartz nodes could span from 8 to 100 GB. The broader range of memory usage on Quartz indicates more diverse workloads than those on Lassen. Both systems experience extremely intensive memory usage that nearly exhausts their memory capacity, but such usage accounts for about 2% of the total time.

Figure 3 reports the peak and average memory usage on each node. Overall, the memory usage is balanced over nodes such that all nodes have similar peak and average values. Note that nodes with small node IDs show much lower usage than others because they are in the debug partition. We observe that most nodes have average usage in 60-80 GB on Lassen, while the average usage on Quartz spans from 20 GB to 60 GB. The peak memory usage on Lassen concentrates at 275, 300, and 320 GB, while on Quartz nodes, it spans from 105 to 128GB. This distribution again confirms that workloads on Quartz have more diverse memory usage.

Takeaway: *More memory-intensive workloads continue emerging on Quartz in the past three years. Compute nodes on Lassen and Quartz utilize less than 80 GB for 80% of the time. Workloads on the Quartz (CPU-only system) show more diverse memory usage than workloads on Lassen (GPU-based system).*

4 JOB-LEVEL CHARACTERIZATION

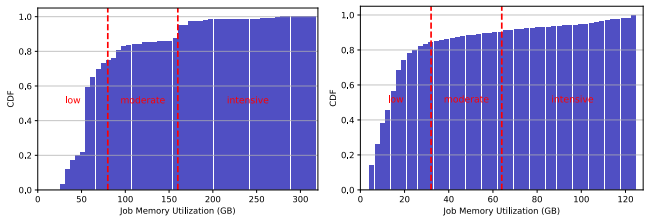
In this section, we classify jobs by their *memory intensity* and decompose the overall utilization into three memory intensity patterns. We quantify the composition of the total system load by each group of jobs. We characterize the memory usage by each group of jobs to evaluate the actual memory needs of workloads on the production systems.

We classify jobs based on a *memory intensity* metric, i.e., the peak usage sampled on all nodes throughout a job. We separate jobs into three groups based on their memory intensity. On a system, if the peak usage of a job is below 25% memory capacity, it is categorized as low intensity. Similarly, jobs that use 25%-50% memory capacity are moderate, and those above 50% are memory-intensive. Accordingly, on the Lassen system, jobs with memory usage below 80 GB are categorized as low intensity, and those using more than 160 GB are memory-intensive. On the Quartz system, the threshold for the three memory intensity groups is 32GB and 64GB.

Most jobs on the two systems have low memory intensity. In particular, 75% of the jobs on Lassen use less than 80 GB, and 85% of the jobs on Quartz use less than 32 GB. About 10% of the jobs on Lassen use a moderate amount of memory between 80 GB and 160 GB. Only 5% of the jobs on Quartz use between 32 GB and 64 GB memory. Memory-intensive jobs take up about 15% of the total jobs on Lassen, and they use more than 160GB of memory. On Quartz, about 10% of the total jobs are memory intensive and use more than 64GB of memory.

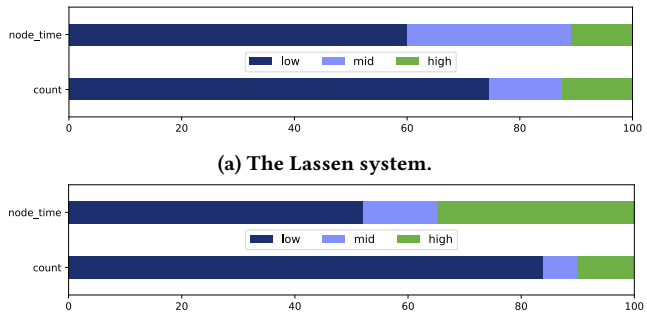
Composition of the system load. Figure 5 presents the breakdown of the overall system load into the three groups of jobs. Although memory-intensive jobs only take up 10% of the total jobs, they consume about 35% node-hours on the Quartz system. This substantial composition in system load indicates that memory-intensive jobs on Quartz likely use more nodes or run longer time than jobs with low or moderate memory intensity. On Lassen, jobs with moderate memory intensity only take 10% of the total jobs but compose 30% of the total system load. On the two systems, jobs of low memory usage compose 50%-60% total system load. From the utilization decomposition, most jobs can be accommodated with reduced node capacity. If the saved cost from memory can be used to facilitate more nodes, the overall system throughput could be increased.

Hot-spots exist in memory usage on both systems. Figure 6 presents the distribution of peak memory usage in jobs, separated into the three groups of memory intensity. A few memory usages occur much more frequently than others, as shown in the spiked bars in Figure 6. For instance, on the Quartz system, jobs with low memory intensity likely use 6 GB memory. Memory-intensive jobs likely use 122 GB memory, nearly exhausting the memory capacity. These jobs are likely constraint by memory capacity. One interesting finding is that the most likely memory usage on Lassen is 60 GB. As each node features four GPUs with a total of 64 GB



(a) The Lassen system. **(b) The Quartz system.**

Figure 4: CDF of memory usage in jobs. The red lines separate jobs into three groups of low, moderate, intensive usage.



(a) The Lassen system.

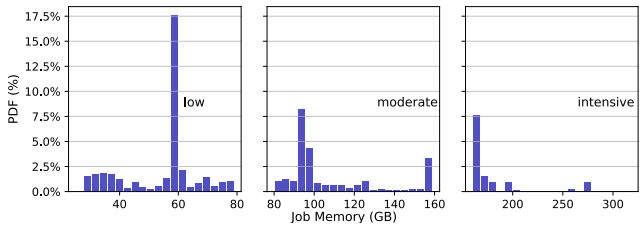
(b) The Quartz system.

Figure 5: Decomposition of the total system load (in node-hours and the number of jobs) by the three job categories.

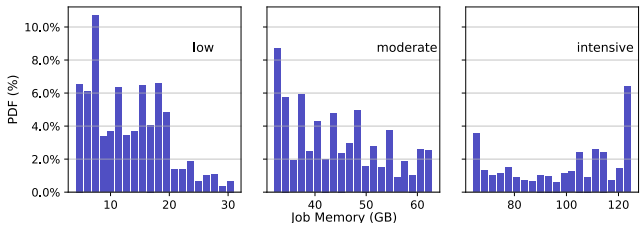
memory, such a hotspot is likely created by the fact that most jobs are configured to fit into the GPU memory for best performance.

Correlation between memory usage and job size. Figure 7 presents the correlation matrix between memory usage and a job's elapsed time and size of allocated nodes. We focus on comparing the characteristic differences between Lassen and Quartz systems. Quartz has a positive correlation between memory usage and job size. The more nodes assigned to a job or longer time elapsed in a job, the job is likely to use more memory. These positive correlations have r-values of 0.33 and 0.2. In contrast, the correlation on Lassen is much weaker, with an r-value of 0.05 and 0.1 only. One reason for the weak correlation on Lassen could be its massive memory capacity, i.e., jobs are likely not constrained by node memory. Another reason could be the high computing power per compute node on Lassen, which can support jobs in a shorter time and with fewer nodes than Quartz.

Takeaway: *Most jobs use less than 25% memory capacity on both systems. The two systems have different hotspots in memory usage – Lassen has a hotspot at 60GB (approximate the total GPU memory) and Quartz has a hotspot at 6GB. Although only 10% jobs are memory-intensive, they compose 35% system load on Quartz. Job's memory usage is positively correlated with job's size, but the correlation is much weaker on Lassen than Quartz.*

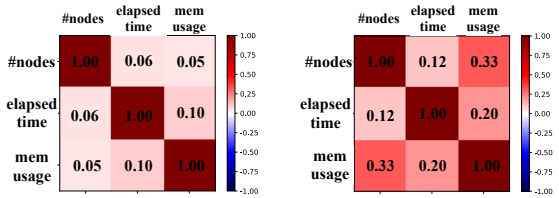


(a) The Lassen system.



(b) The Quartz system.

Figure 6: The PDF of memory usage in the three groups of jobs with low, moderate, and intensive memory usage.



(a) The Lassen system. (b) The Quartz system.

Figure 7: The correlation matrix between job size (the number of nodes and elapsed time) and job's peak memory usage.

5 TEMPORAL CHARACTERISTICS

In this section, we characterize the temporal memory imbalance in jobs on the production systems. Each job can be viewed as two-dimensional, i.e., the time dimension spans over the job's duration, and the space dimension spans over all assigned nodes. The memory utilization on a node changes in time and causes *temporal memory imbalance*. Currently, the allocation of memory resources is static on HPC systems, i.e., the resource is requested based on the peak usage of a job, and resource configuration remains unchanged once allocated to a job. Therefore, temporal imbalance results in temporal under-utilization. We propose a new metric called $RU_{temporal}$ in Equation 2 to quantify the temporal imbalance. Jobs are categorized into four temporal patterns based on $RU_{temporal}$, and we reconstructed real job traces from the Sonar infrastructure in Figure 8 to demonstrate these representative patterns. The distribution of these temporal groups is shown in Figure 9. We analyze the actual memory usage by jobs in each temporal pattern in Figure 10.

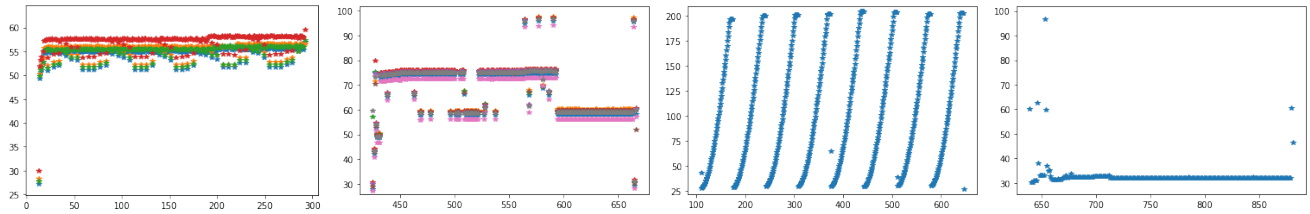
Equation 2 calculates $RU_{temporal}$ for Node n in a job i that runs for T_i time. $S_{n,t}$ represents the actual memory usage sampled at time t on Node n. $RU_{temporal}$ is bounded to the range of [0,1]. Ideal temporal utilization would have $RU_{temporal}$ close to zero and a large value indicates that memory resource is underutilized most of the time.

$$RU_{temporal}(i,n) = 1 - \frac{\sum_{t=0}^{T_i} S_{n,t}}{\sum_{t=0}^{T_i} \max_{0 \le t \le T_1} S_{n,t}} \quad (2)$$

We use $RU_{temporal}$ to summarize temporal characteristics in jobs. Depending on the maximum $RU_{temporal}$ value on all nodes in a job, a job can be categorized into *constant, phased, dynamic, and sporadic* patterns. The constant pattern has minimal changes in memory utilization throughout a job. Jobs with $RU_{temporal}$ values lower than 0.2 are categorized in this pattern. We reconstruct a real job in the constant pattern from system monitoring data and present it in Figure 8a. All samples from one node are depicted in the same color. For instance, green points show the usage on a node at different times, and red points show the usage on another node. Figure 8a shows that throughout the job's time, the changes in memory usage are low. The phased pattern has $RU_{temporal}$ value in 0.2-0.4. Representative jobs in this pattern have their executions divided into multiple phases. Memory usage in each phase could be different. Figure 8b presents one real job composed of four long phases. The dynamic pattern has more temporal imbalance than the previous two and is classified with $RU_{temporal}$ values in 0.4-0.6. This pattern often has frequent and substantial changes in memory usage over time. We show one real job in the dynamic pattern in Figure 8c, where memory usage changes from 25 to 200 GB repeatedly during the job. The sporadic pattern may cause the most severe underutilization and is classified by $RU_{temporal}$ value larger than 0.6. Most jobs in this pattern exhibit spiked memory usage for short periods. For example, the job in Figure 8d has a stable usage at 30 GB but uses up to 100 GB at several time points.

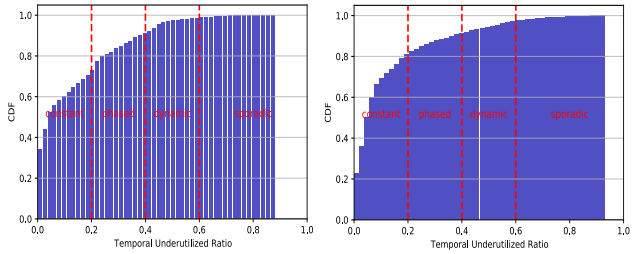
We present the distribution of temporal imbalance in jobs in Figure 9. A job is categorized by the maximum $RU_{temporal}$ value on all nodes in a job. In other words, a job's temporal characteristic is determined by the node with the highest temporal memory imbalance (the difference between nodes will be analyzed in Section 6). Overall, most jobs on the two systems have good temporal balance, i.e., $RU_{temporal}$ values below 0.2. However, the temporal underutilization can reach as high as 0.9 and 0.95 on Lassen and Quartz systems, respectively. About 70% jobs on the GPU-based systems have the constant temporal pattern, while 80% jobs on the CPU-based system are in this category. For the phased pattern, i.e., $RU_{temporal}$ values between 0.2 and 0.4, the GPU-based system has 20% jobs in this category while the CPU-based only has about 10% jobs. Both systems have only about 10% jobs with significantly high temporal underutilization, i.e., dynamic and sporadic patterns with $RU_{temporal}$ above 0.4. The CPU-based system has about 5% jobs in the sporadic pattern, slightly more than the 2.5% jobs on the GPU-based system.

Memory imbalance over time causes a job to require each of its nodes to be configured with memory capacity equal to the peak temporal usage on that node. This constraint is because the memory resource is fixed once a job is scheduled on current HPC systems. Our categorization of jobs in Figure 13 shows the ratio of temporal imbalance in jobs. However, a high $RU_{temporal}$ value unnecessarily indicates a high potential for improvement if its peak memory usage is low. We try to understand the actual node memory



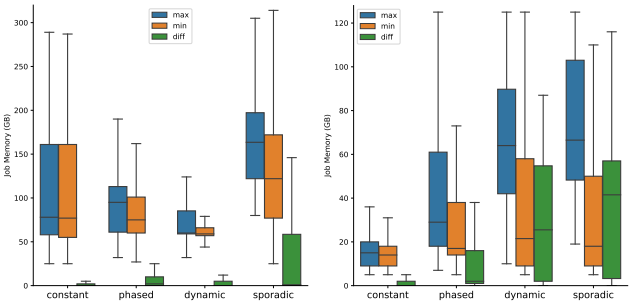
(a) Constant pattern (b) Phased pattern (c) Dynamic pattern (d) Sporadic pattern

Figure 8: Four temporal patterns are identified from system monitoring data. Each scatter plot represents time (in minutes) in the x-axis and memory usage (in GB) in the y-axis. Points in the same color represent samples on the same node.



(a) The Lassen system. (b) The Quartz system.

Figure 9: The distribution of temporal under-utilization in jobs on the two HPC systems. The red dashed lines separate jobs into constant, phased, dynamic, and sporadic patterns.

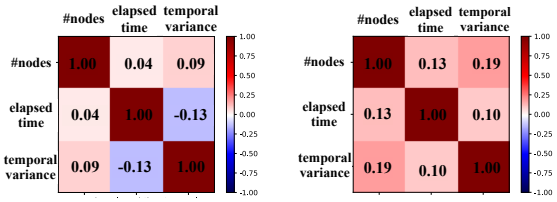


(a) The Lassen system. (b) The Quartz system.

Figure 10: The memory usage in jobs in the four temporal patterns. The box plot summarizes the minimum, lower quartile, median, upper quartile, and maximum of usage.

usage in jobs in Figure 10, which presents the node memory usage in jobs in each temporal category.

On both systems, jobs in the constant group have nearly identical distribution in memory usage on nodes. 75% jobs in this category on Lassen use less than 160 *GB* memory while 75% jobs on Quartz use less than 20 *GB* memory. As 70%-80% total jobs on these two systems are in this group, memory resource is overprovisioned by about 160 and 88 GB for more than 50% total jobs on Lassen and Quartz systems, respectively. In fact, on the Quartz system, this group of jobs uses no more than 40 GB.



(a) The Lassen system. (b) The Quartz system.

Figure 11: The correlation matrix of temporal variance in memory usage with job size and elapsed time on two systems.

One main difference between the two systems is memory usage changes in constant, phased to dynamic groups. For 75% jobs, once sufficient memory is provided on the GPU-based system for the constant group (i.e., 160 GB), jobs in phased and dynamic groups are also supported. However, the required memory on the CPU-based system increases from 20, 60 to 90 GB in these three groups. To support 75% jobs in the sporadic group, at least 200 and 105 GB memory capacity is needed on the Lassen and Quartz system, respectively. The difference in node usage in these jobs is about 50 GB on both systems.

Correlation with job characteristics. We quantify the correlation of temporal variance in memory usage with the job size and duration. Figure 11 visualizes the correlation matrix of the three variables. A drastic change in the correlation is between the temporal variance and job elapsed time from the CPU-based system to the GPU-based system. The temporal variance on Lassen is negatively correlated to the job duration (elapsed time), indicating that jobs that run longer on the GPU-based machine are likely to exhibit lower temporal variance in memory usage. In contrast, on the CPU-based machine, jobs that run longer are likely to show more temporal imbalance. One possible reason for the negative correlation on Lassen is that longer jobs on the GPU-accelerated system likely need to use more memory to supply data for computation and thus likely constantly have higher memory usage throughout the job.

Takeaway: *Temporal imbalance in memory usage is low in* 70%-80% *jobs on the two systems. However, temporal underutilization could reach beyond* 90% *on both systems because* 50% *jobs on Lassen and Quartz are over-provisioned with* 160 *and* 88 *GB memory, respectively.*

Temporal imbalance is negatively correlated with the job's time on Lassen but positively correlated on Quartz.

6 SPATIAL CHARACTERISTICS

The *spatial memory imbalance* in a job comes from the different memory usage on nodes. In this section, we define a new metric called $RU_{spatial}$ to classify jobs based on their spatial imbalance. We follow the utilization decomposition analysis to categorize jobs into different spatial utilization patterns. Three real job traces are reconstructed from monitoring data in Figure 12 to show their signature usage behaviors. The composition of these job groups on the systems is then presented in Figure 13. We characterize the actual memory usage in each spatial group and report in Figure 14.

Eq. 3 computes the $RU_{spatial}$ value for a job i, where J (i) in Eq. 4 computes the peak usage among all nodes in the job and N (n) computes the peak usage from samples (S) throughout the job's lifespan (ts_i–te_i) on Node n (Eq. 5). $RU_{spatial}$ integrates underutilized memory resources on all nodes in a job ($\mathbb{Z}_i$). It serves as an indicator of *memory imbalance* among nodes in a job. The value of $RU_{spatial}$ is bounded to [0,1]. A $RU_{spatial}$ value of zero indicates perfectly balanced memory usage among all nodes. A large value indicates that only few nodes can reach the job's peak memory, representing large improvement potential on the job's spatial usage.

$$RU_{spatial}(i) = \sum_{1}^{N}(J(i) - N(ts_i, te_i, n))/\sum_{1}^{N} J(i) \tag{3}$$

$$J(i) = \max\{N(ts_i, te_i, n) : n \in \mathbb{Z}_i\} \tag{4}$$

$$N(t_0, t_1, n) = \max_{t_0 \le t \le t_1} S_{n,t} \tag{5}$$

We compute $RU_{spatial}$ for each job on the two systems and categorize jobs into three groups–*batched*, *grouped*, *centralized*. The batched pattern has $RU_{spatial}$ values lower than 0.2. Jobs in the batched pattern have similar peak memory usage on all nodes. Figure 12a shows a real job trace captured by the Sonar system. This job is allocated with eight compute nodes, and all the nodes use similar memory throughout the job. Note that this job actually has low temporal utilization as characterized by $RU_{temporal}$ in the previous section and has a dynamic temporal pattern.

The grouped pattern has $RU_{spatial}$ values in 0.2-0.6. Jobs in this group have lower spatial utilization because some nodes cannot reach the peak memory usage as other nodes. Figure 12b shows one representative job, where nodes in the jobs are separated into three groups. These groups have a different peak usage of 25 GB, 60 GB, 70 GB, respectively.

The centralized pattern has $RU_{spatial}$ values higher than 0.6. Jobs in this pattern have very different peak memory usage among nodes and likely have the highest optimization potentials. Figure 12c presents one real job on the Quartz system that exhibits the centralized spatial pattern. Dots in the same color represent samples collected from the same node. In this example, all nodes except the one in blue dots use lower than 10 GB memory throughput the job while one node uses about 70 GB memory. We also evaluate other jobs with such behavior. We find that the first node in the list of assigned nodes is often the one that uses significantly more memory than the others. One explanation could be that the first node is often mapped to be MPI rank 0, which is known to handle extra works, such as I/O and coordination.

We present the distribution of spatial imbalance in jobs in Figure 13. Overall, jobs on the Lassen system are more memory balanced over nodes than those on the Quartz system. About 95% jobs on Lassen have $RU_{spatial}$ values lower than 20% (Figure 13a). The maximum spatial imbalance on Lassen is 88% while it could reach 92% on Quartz. Only a small portion of jobs have grouped or centralized patterns on both systems. However, for a full picture of memory utilization in jobs, we need to combine temporal and spatial metrics because, as shown in the example in Figure 12a, jobs with low $RU_{spatial}$ could have high temporal underutilization. The spatial imbalance in jobs on the Quartz system is more diverse than Lassen. About 10% jobs have $RU_{spatial}$ higher than 20%, and 5% jobs have more than 60% spatial imbalance.

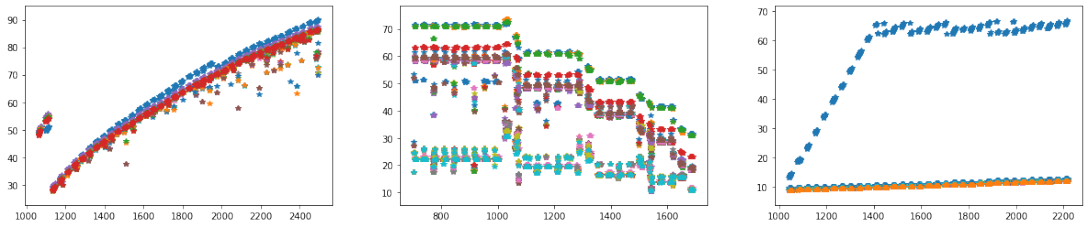
Memory imbalance across nodes in a job could reduce resource utilization significantly because most HPC systems today are composed of nodes configured with the same resources. Based on spatial utilization categorization, we evaluate the actual memory usage in jobs in each category. Figure 14 presents the distribution of memory imbalance among nodes in a job, i.e., the difference between the highest node usage and the lowest node usage in a job (denoted as diff), and the highest and lowest node usage (denoted as max and min), respectively.

Jobs in the batched group have the lowest memory imbalance, i.e., the green bar in the first group is approximately zero in Figure 14a and 14b. 75% jobs in this group use less than 80 GB memory on Lassen and less than 20 GB on Quartz. As more than 90% of total jobs on the two systems fall in this category, they indicate that most jobs are over-provisioned with 240 GB and 108 GB memory on the Lassen and Quartz system, respectively.

Jobs in the grouped pattern behave differently on the two systems. On Lassen, the distribution of memory imbalance is centered at 30 GB (i.e., the green bar in Figure 14a). These jobs on Lassen are over-provisioned with at least 245 GB memory because their maximum node usage is only 75 GB (the blue bar in Figure 14a). However, on Quartz, the memory imbalance could spread from 10 GB to 40 GB at 25th and 75th percentile. Therefore, memory imbalance in jobs of the grouped pattern is more diverse on Quartz than Lassen.

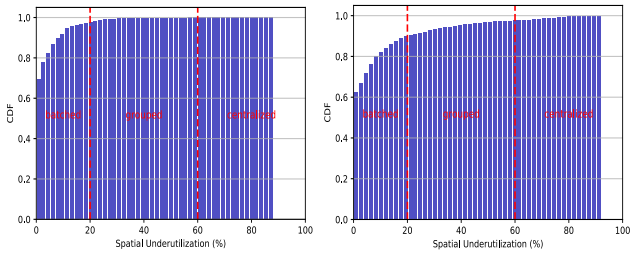
The centralized group presents the highest memory imbalance. The average memory imbalance on Lassen is about 160 GB in these jobs and approximately 58 GB on Quartz. The minimum node usage is centered at 25GB on Lassen and 10 GB on Quartz. Note that jobs in this group mostly have only one or very few nodes that use the maximum usage, and thus, we exclude them to estimate the overprovision to be at least 270 GB on Lassen and 88 GB on Quartz.

Correlation with job characteristics We quantify the correlation of memory imbalance (spatial variance) in a job with the job size (the number of nodes) and the elapsed time. Figure 15 visualizes the correlation matrix of the three variables. On both systems, spatial variance is positively correlated with the number of nodes in a job, which is expected. The most significant difference between the two systems is the correlation between spatial variance and job elapsed time. Spatial variance is positively correlated with job time on Quartz (Figure 15b with an r-value of 0.24, indicating that increased job length is likely to be accompanied by increased memory



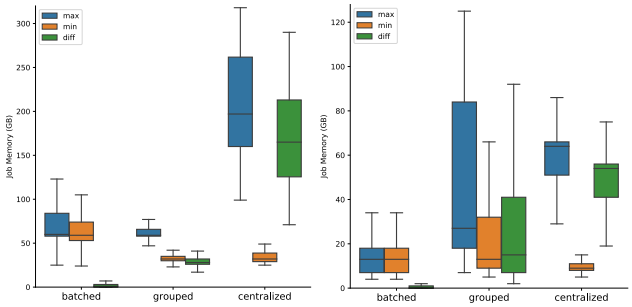
(a) Batched pattern. (b) Grouped pattern. (c) Centralized pattern.

Figure 12: Three spatial patterns are identified from system monitoring data. Each scatter plot represents time (in minutes) in the x-axis and memory usage (in GB) in the y-axis. Points in the same color represent samples on the same node.



(a) The Lassen system. (b) The Quartz system.

Figure 13: The distribution of spatial imbalance ($RU_{spatial}$) in jobs. The red dashed lines separate jobs into batched, grouped, and centralized patterns.

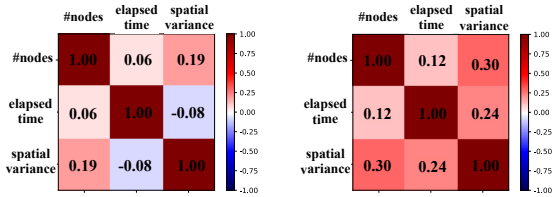


(a) The Lassen system. (b) The Quartz system.

Figure 14: The peak node memory usage in jobs in the three spatial patterns. The box plots the minimum, lower quartile, median, upper quartile, and maximum of usage.

imbalance on Quartz. However, the correlation becomes negative on Lassen, indicating that longer jobs are likely to have reduced spatial memory imbalance. One explanation could be large jobs intend to fit in the fast GPU memory capacity for best performance.

Takeaway: *Spatial imbalance in nodes could result in severe underutilization in jobs exhibiting centralized and grouped patterns. Emerging workloads like workflows deploying diverse tasks onto nodes could exhibit such patterns. Job's spatial imbalance is positively correlated with job time on Quartz but negatively correlated on Lassen.*



(a) The Lassen system. (b) The Quartz system.

Figure 15: Correlation of spatial variance across nodes in a job with the job size (i.e., the number of assigned nodes and the elapsed time).

7 IMPLICATIONS FOR FUTURE SYSTEMS

In this section, we discuss considerations in applying our study and findings. We share a recent procurement decision to show the impact of our study. We also outline promising new architectural designs motivated by the utilization patterns in real jobs.

Our method is generally applicable to HPC facilities with monitoring infrastructure deployed to provide required system samples. A facility may discover different utilization characteristics when applying our analysis approach to their monitoring data. The findings presented in this work reflect the current state of the two leadership supercomputers. We believe that the characteristic differences between Lassen and Quartz are more a result of workloads than architecture, i.e., the main incentive for an application to choose a GPU-based system is likely for speedup by the accelerators, not for memory capacity. Therefore, jobs on Lassen are likely configured to fit in GPU for best performance. Our findings of the average and hotspot memory usage on Lassen support this assumption. When deriving insights for future systems, one needs to project the changes in the workloads.

7.1 Procurement Decisions

System-level data is impactful in making procurement decisions [11]. Memory is one of the most expensive components of a system, making up more than 20% overall cost. Therefore, memory subsystem configuration is critical from the system design perspective.

The data from Sonar is used to inform the machine configuration through an upcoming NNSA Commodity Technologies System (CTS) procurement. Data from Sonar was augmented with data collected in logs from a classified system called Jade, which has

the same architecture as Quartz. The findings showed that the Jade system's workloads are more memory-intensive than Quartz's workloads. While over 80% of jobs on Quartz have low memory intensity, only about 40% of the jobs on Jade are in this category. The same finding holds true for the high memory intensity category, where only 10% of Quartz jobs but 25% of Jade jobs are in this category. These differences impact the design of a new system. While most of the Quartz workloads use less than 1 GB of memory per core, most Jade workloads used at least 1 GB. Still, on both systems, most of the memory resources are not utilized, and there is space to optimize the system design.

For future CTS systems, the data presented in this paper made a strong case to push memory capacity down to 2 GB per core if possible on all systems. Solutions in the 2-3 GB per core range are about 20% cheaper per node than those with double the memory. In the new system designs, some jobs would need to use more nodes, but most jobs would not notice. Also, the extra compute that we could get would make up for the lost memory capacity.

To reduce memory capacity meant to give up Chipkill memory [6] used for reliability in current systems, and/or moving to single rank DIMMs, impacting memory bandwidth. Both have performance impacts, but modeling showed that single-rank DIMMs would cost only 4-5% average application performance. Also, the additional check-pointing needed by removing Chipkill would cost less than the impact of removing a rank from the DIMMs. Therefore, by reducing memory capacity, a procurement decision is made for a machine that is about 5% less capable per node, but about 20% more nodes are configured.

In addition, the data shows that further memory capacity reductions down to 1-1.5GB per core would suffice for most jobs. This opens up the option of HBM [10] only based CPUs, such as the new A64FX used in the Fugaku supercomputer [1]. HBM could enable many bandwidth-bound HPC applications to improve performance significantly. Over 80% of Quartz jobs and 40% of Jade jobs would fit in 1GB of memory per core. Likely for cost reasons and because data staging tends not to be used much in practice on CPU+GPU machines, these machines would be HBM only. Whether an HBM-only machine could serve all workloads or a machine with a hybrid of DDR nodes and HBM nodes is an open question that requires further investigation.

While the GPU data collected here has not been used in procurements yet, it provides useful insight into how the machine is being used and confirms some anecdotal stories from users. For example, Figure 6a shows a significant number of jobs using memory close to the capacity of the GPUs, which is 64GB. These jobs are likely trying to fit within GPU memory either because this is the most efficient operation point or they have not implemented an efficient data migration strategy. These GPU-memory-capacity limited jobs match user stories. Many code teams shared that they get their best throughput and machine efficiency when they use all GPU memory. Many of them tried staging data between host and device and found they did not have enough data reuse to take advantage of the extended memory. Coupling this anecdotal evidence with the data provided by Sonar justifies increasing GPU memory capacity when possible and shows more incentive than increasing CPU memory.

7.2 Emerging System Architecture

Temporal utilization patterns identified in real jobs present opportunities in exploring dynamic resource configurations and disaggregated memory [17, 22] on future systems. The overhead of resource re-configuration depends on the frequency of usage changes in a job. The constant temporal pattern exhibits little changes in memory usage throughout a job, and it composes most system loads. Also, the phased pattern only has a few disruptive usage changes throughout a job. Therefore, both patterns could be candidates for dynamic resource configuration because the overhead of re-configuration would be low. The sporadic pattern has a few short usage spikes throughout a job, which could leverage disaggregated memory for temporary memory expansion. Since remote memory is only used in short periods, the performance impact would be limited.

Spatial utilization patterns reveal the imbalanced memory usage across nodes in one job. Today, most supercomputers are composed of uniformly configured nodes, i.e., each node provides the same amount of resources. In this work, we identified the centralized pattern in jobs, where only one node in a job needs high memory capacity. Codes with the centralized pattern could be forced to adapt to new environments rather than being accommodated. Understanding the tradeoff between the cost of the code changes and the cost of accommodation needs to be address in future designs. As more complex workflows emerge on HPC systems, nodes performing different workflow stages may have very different resource requirements. These spatially imbalanced jobs could be more efficiently supported on systems composed of heterogeneous subclusters [3], where nodes are configured with different resources.

8 RELATED WORK

Previous works used system-level characterization of I/O, power, and memory for performance optimizations [14, 15, 17, 29]. Xie et.al [29] build a prediction model for I/O performance on Titan supercomputer. Patel et.al [15] investigate the I/O behaviors of a parallel file system on NERSC supercomputer and correlate temporal and spatial I/O activities with hardware utilization. Panwar et.al [14] quantified node-level memory usage and proposes to leverage unused memory for boosting system performance. Peng et.al [17] also characterize system-level memory usage and propose to use disaggregated memory to improve utilization. Our work is the first large-scale study of job's temporal and spatial memory usage on production HPC systems.

Several works use job logs to correlate with subsystem logs for analyzing job-level behaviors on HPC systems [7, 12, 27, 30]. Madireddy et.al [12] correlate job logs with storage-system logs to analyze job- and system-level I/O activities. Wang et.al [27] use five years' job logs to characterize the utilization of CPU, GPU, memory, and I/O on Titan supercomputer. Zheng et.al [30] correlate job logs with RAS logs to uncover the characteristics of system failures and job interruptions. LogAider [7] provides a tool for identifying correlations in job logs and RAS logs. Our work correlates high-resolution monitoring samples with job metadata to accurately depict the current state and provide quantitative guidance for system designs.

Prior works often use selected applications to represent workloads on HPC systems. Turner et.al [25] study the memory usage of selected applications at different job sizes. Zivanovic et.al [31] use HPCG and HPL benchmarks to assess the suitability of 3D-stacked memory for HPC systems. Weinberg et.al [28] quantify the spatial and temporal memory locality in a set of selected applications on different architectures. Ji et.al [9] characterize the memory access patterns of 38 applications. Our work monitors all applications running on a system, reflecting their impact on the current system but needs abstraction and projection for future systems.

9 CONCLUSION

Co-designing memory subsystems has become increasingly challenging for HPC facilities because workloads continue evolving and new architectural options continue emerging. We introduce our strategy at a leadership facility that leverages system monitoring data to inform future system designs and procurement decisions. To the best of the authors' knowledge, this is the first large-scale study of memory utilization with spatial and temporal characterizations on production HPC systems. Our study identifies the changes in memory utilization from a CPU-only system to GPU-accelerated supercomputers. We discover representative spatial and temporal utilization patterns in real jobs. The findings in this study provide quantitative guidelines for emerging architectural designs. Finally, we showcase the impact of our study in an upcoming NNSA CTS procurement.

ACKNOWLEDGMENTS

This research was partially supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. LLNL-CONF-821258.

REFERENCES

[1] 2021. the Fugaku supercomputer. https://www.fujitsu.com/global/about/innovation/fugaku/.
[2] Anthony Agelastos, Benjamin Allan, Jim Brandt, Paul Cassella, Jeremy Enos, Joshi Fullop, Ann Gentile, Steve Monk, Nichamon Naksinehaboon, Jeff Ogden, et al. 2014. The lightweight distributed metric service: a scalable infrastructure for continuous monitoring of large scale computing systems and applications. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 154–165.
[3] Faraz Ahmad, Srimat T Chakradhar, Anand Raghunathan, and TN Vijaykumar. 2012. Tarazu: optimizing mapreduce on heterogeneous clusters. *ACM SIGARCH Computer Architecture News* 40, 1 (2012), 61–74.
[4] Livermore Computing. 2021. the Lassen supercomputer. https://hpc.llnl.gov/hardware/platforms/lassen.
[5] Livermore Computing. 2021. the Quartz supercomputer. https://hpc.llnl.gov/hardware/platforms/Quartz.
[6] Timothy J Dell. 1997. A white paper on the benefits of chipkill-correct ECC for PC server main memory. *IBM Microelectronics division* 11 (1997), 1–23.
[7] Sheng Di, Rinku Gupta, Marc Snir, Eric Pershey, and Franck Cappello. 2017. Logaider: A tool for mining potential correlations of hpc log events. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 442–451.
[8] Joseph Izraelevitz, Jian Yang, Lu Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R Dulloor, et al. 2019. Basic performance measurements of the intel optane DC persistent memory module. *arXiv preprint arXiv:1903.05714* (2019).
[9] Xu Ji, Chao Wang, Nosayba El-Sayed, Xiaosong Ma, Youngjae Kim, Sudharshan S Vazhkudai, Wei Xue, and Daniel Sanchez. 2017. Understanding object-level memory access patterns across the spectrum. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12.
[10] Joonyoung Kim and Younsu Kim. 2014. HBM: Memory solution for bandwidth-hungry processors. In *2014 IEEE Hot Chips 26 Symposium (HCS)*. IEEE, 1–24.
[11] Edgar A. León, Ian Karlin, Abhinav Bhatele, Steven H. Langer, Chris Chambreau, Louis H. Howell, Trent D'Hooge, and Matthew L. Leininger. [n.d.]. Characterizing Parallel Scientific Applications on Commodity Clusters: An Empirical Study of a Tapered Fat-Tree. In *SC'16: the International Conference for High Performance Computing, Networking, Storage and Analysis*.
[12] Sandeep Madireddy, Prasanna Balaprakash, Philip Carns, Robert Latham, Robert Ross, Shane Snyder, and Stefan M Wild. 2017. Analysis and correlation of application I/O performance and system-wide I/O activity. In *2017 International Conference on Networking, Architecture, and Storage (NAS)*. IEEE, 1–10.
[13] Alessio Netti, Daniele Tafani, Michael Ott, and Martin Schulz. 2021. Correlation-wise Smoothing: Lightweight Knowledge Extraction for HPC Monitoring Data. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2–12.
[14] Gagandeep Panwar, Da Zhang, Yihan Pang, Mai Dahshan, Nathan DeBardeleben, Binoy Ravindran, and Xun Jian. 2019. Quantifying memory underutilization in HPC systems and using it to improve performance via architecture support. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 821–835.
[15] Tirthak Patel, Suren Byna, Glenn K Lockwood, and Devesh Tiwari. 2019. Revisiting I/O behavior in large-scale storage systems: The expected and the unexpected. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–13.
[16] J Thomas Pawlowski. 2011. Hybrid memory cube (HMC). In *2011 IEEE Hot Chips 23 Symposium (HCS)*. IEEE, 1–24.
[17] Ivy Peng, Roger Pearce, and Maya Gokhale. 2020. 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)*. IEEE, 183–190.
[18] Ivy B Peng, Roberto Gioiosa, Gokcen Kestor, Jeffrey S Vetter, Pietro Cicotti, Erwin Laure, and Stefano Markidis. 2018. Characterizing the performance benefit of hybrid memory system for HPC applications. *Parallel Comput.* 76 (2018), 57–69.
[19] Ivy B Peng, Maya B Gokhale, and Eric W Green. 2019. System evaluation of the intel optane byte-addressable nvm. In *Proceedings of the International Symposium on Memory Systems*. 304–315.
[20] Ivy B Peng and Jeffrey S Vetter. 2018. Siena: Exploring the design space of heterogeneous memory systems. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 427–440.
[21] Milan Radulovic, Darko Zivanovic, Daniel Ruiz, Bronis R de Supinski, Sally A McKee, Petar Radojković, and Eduard Ayguadé. 2015. Another trip to the wall: How much will stacked dram benefit hpc?. In *Proceedings of the 2015 International Symposium on Memory Systems*. 31–36.
[22] Vishal Shrivastav, Asaf Valadarsky, Hitesh Ballani, Paolo Costa, Ki Suh Lee, Han Wang, Rachit Agarwal, and Hakim Weatherspoon. 2019. Shoal: A network architecture for disaggregated racks. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*. 255–270.
[23] JEDEC Standard. 2013. High bandwidth memory (hbm) dram. *Jesd235* (2013).
[24] top500.org. 2021. Top 500 list. https://www.top500.org/lists/top500/list/2020/11/.
[25] Andy Turner and Simon McIntosh-Smith. 2017. A survey of application memory usage on a national supercomputer: an analysis of memory requirements on ARCHER. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. Springer, 250–260.
[26] Sudharshan S Vazhkudai, Bronis R de Supinski, Arthur S Bland, Al Geist, James Sexton, Jim Kahle, Christopher J Zimmer, Scott Atchley, Sarp Oral, Don E Maxwell, et al. 2018. The design, deployment, and evaluation of the CORAL pre-exascale systems. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 661–672.
[27] Feiyi Wang, Sarp Oral, Satyabrata Sen, and Neena Imam. 2019. Learning from Five-year Resource-Utilization Data of Titan System. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 1–6.
[28] Jonathan Weinberg, Michael O McCracken, Erich Strohmaier, and Allan Snavely. 2005. Quantifying locality in the memory access patterns of HPC applications. In *SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*. IEEE, 50–50.
[29] Bing Xie, Yezhou Huang, Jeffrey S Chase, Jong Youl Choi, Scott Klasky, Jay Lofstead, and Sarp Oral. 2017. Predicting output performance of a petascale supercomputer. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*. 181–192.
[30] Ziming Zheng, Li Yu, Wei Tang, Zhiling Lan, Rinku Gupta, Narayan Desai, Susan Coghlan, and Daniel Buettner. 2011. Co-analysis of RAS log and job log on Blue Gene/P. In *2011 IEEE International Parallel & Distributed Processing Symposium*. IEEE, 840–851.
[31] Darko Zivanovic, Milan Pavlovic, Milan Radulovic, Hyunsung Shin, Jongpil Son, Sally A Mckee, Paul M Carpenter, Petar Radojković, and Eduard Ayguadé. 2017. Main memory in HPC: Do we need more or could we live with less? *ACM Transactions on Architecture and Code Optimization (TACO)* 14, 1 (2017), 1–26.