Reducing Latency in Hybrid HPC Systems through Containerization and Parallel GPU Processing

Manju George
Independent researcher,
Waukesha, USA

ABSTRACT
The research examines how High-Performance Computing (HPC) and cloud-native environments can meet through scalable computing environments. The study measures the optimization of computational efficiency of large-scale data processing through containerization and hardware acceleration. With the help of a synthetic dataset containing 411 examples of high-dimensional performance measurements, the research modeling simulates different workload distributions in hybrid infrastructures. The main operated tools are Kubernetes as an orchestration tool, Docker as an environment isolation tool, and dedicated software libraries as a tool that monitors GPU acceleration. Findings have shown that a combination of containerization and parallel processing can lower the latency by a large margin whilst ensuring that hardware is utilized fully. It is concluded in the abstract that a single piece of architecture is needed to handle modern data-intensive tasks.

Keywords
Scalable Computing, GPU Acceleration, Containerization, Parallelism, Cloud AI.

1. INTRODUCTION
The need to compute has moved out of the isolated research laboratory into the operational heart of contemporary industry so that computation is now being viewed as a strategic infrastructure element, not a utility in the background, as analyses of industrial digital transformation were reported by [4]. Modern areas like genomics, climate simulation, financial modeling and deep neural network training are based on platforms that can scale the performance in direct proportion to workload intensity, workload-scaling needs explored in high-performance computing literature by [9]. Scalable computing This property is identified as scalable computing and is the architectural property that allows systems to remain efficient and throughput as load in terms of processor count, memory, or distributed nodes. This capability has been transformed into a minimum threshold to conduct business operations in a business with big data volumes and speeding data traffic, with fast infrastructures, bottleneck dynamics empirically explored by [11].

The monolithic structures that are traditionally employed in predictable workloads may not be structurally adaptable to the dynamic workload and computational needs of the system, which have also been noted to have rigidity constraints in assessments of the legacy systems conducted by [7]. Specifically, artificial intelligence pipelines can need be elastic since the training stages can require thousands of parallel functions and the inference stages require less but sustained processing capacity elasticity compares studied in AI workload characterization studies done by [3]. Cloud environments solve this gap by providing vertical or horizontal resource scaling of computational resources on demand, so organizations do not need to maintain expensive idle hardware, and provisioning is a paradigm materialized in cloud elasticity models (proposed by [12]). This paradigm has been operationalized by providers like Amazon Web Services who offer elastic compute instances, distributed storage layers as well as networking structures that dynamically scale with the intensity of the workloads, industry application evidence in infrastructure service investigations conducted by [6]. Nonetheless, moving the high-performance workloads to the remote infrastructures comes with latency, bandwidth, and synchronization challenges that should be overcome by optimal data partitioning and smart scheduling scheme explored in the study of distributed systems undertaken by [2].

Modern scalable computing is based on parallelism as the conceptual framework, a philosophy of concurrency, which is stated in processor evolution studies pioneered by [10]. In lieu of comparatively small improvements in single-core clock rates, which have ceased to grow in physical and thermal constraints, modern architectures spread computational workload across many relatively simple processing units, hardware transition trends reported in studies of multicore architecture research have been pursued by [5]. An example of this change can be seen in graphics processing units produced by companies like NVIDIA, which do use thousands of cores designed to perform arithmetic operations simultaneously, design traits of accelerators studied in parallel processing research undertaken by [13]. These processors are efficient in performing matrix multiplications and the calculation of vectors which control machine learning and scientific simulations, and performance suitability is proven in computational throughput studies reported in [8]. The performance advantages that follow, typically orders of magnitude, are the result of concurrency, but not raw frequency, concurrency advantages that are quantified in benchmarking experiments conducted by [4]. However, to exploit such heterogeneous hardware ecosystems, coordination layers are needed that can coordinate CPUs, GPUs, and memory hierarchies and network interconnects without adding synchronization delays or resource contention orchestration complexities studied in heterogeneous computing models, such as the ones suggested in [9].

Resource orchestration hence becomes a fundamental element of design; a governance element conceptualized within the research of distributed resource management synthesized by [1]. In contemporary distributed execution systems, tasks are scheduled based on data locality, hardware availability, and projected runtime properties, which are the scheduling algorithms that can be modeled by workload allocation studies that have been carried out by [11]. These systems improve the health of nodes, rebalance workloads when there are failures and State of the art strategies have been put in place in fault-tolerant architecture studies conducted by [7] to minimize the communication paths and subsequently minimize the latency. The idea is to maintain the equal utilization of all the computational elements to make sure that no one resource would be a bottleneck, the load-balancing

goals are empirically tested by [3]. Full-fledged schedulers use predictive analytics and feedback to dynamically adjust execution plans making computing environments self-optimizing ecosystems, adaptive scheduling paradigms designed by [12].

Containerization also contributes to this change, and it abstracts applications on underlying hardware dependencies, abstraction principles established in container technology studies to be documented in [6]. Containers package software, libraries and runtime settings into portable objects that can be reliably executed on both developments, testing and production environments, portability properties viewed in reproducibility research studies carried out by [2]. This encapsulation removes compatibility differences and greatly shortens deployment time, deployment efficiency gains in empirical infrastructure testing performed by [10]. Infrastructure-wise, containers have the benefit of scaling fast due to the ability to create the same execution environment and replicate it on multiple nodes in real time, and replication was tested in scale test experiments that were run in a cluster by [5]. This property boosts resilience of the system: a failure of one of the instances can be replaced in a few seconds without any human intervention to restart a new instance, resilience results have been studied in high-availability models as suggested by [13]. In turn, containerization facilitates the continuity of operations, makes maintenance and upgrades easier, and life cycle benefits combining synthesized by [8].

The intersection of scalable architectures, heterogeneous hardware acceleration, cloud elasticity and container portability characterizes the future of digital ecosystems next generation convergence models expressed in terms of research on integrated infrastructure introduced by [4]. All these technologies are changing the nature of computing as a fixed resource possession to a dynamic resource coordination such that organizations can manage vast significant amounts of data with both reliability and efficacy, adaptive coordination effects have been reported in large-scale systems research by [1]. This unified paradigm forms the basis of globally distributed, performance-scaled infrastructures, which can support the new scientific, industrial and societal applications, future-scaled visions of scalability synthesized by [9].

2. LITERATURE REVIEW

The academic history of distributed computing studies can be traced to show a gradual transition of initial grid-based systems to complex cloud-native systems that are optimally efficient, flexible, and resourceful, an evolutionary trajectory that has been reported in distributed systems surveys by [7]. Primarily, the early research focused on the use of the aggregate processing power of a geographically distributed set of workstations linked by high-speed networks, grid computing ideas explored in pioneer cluster research undertaken by [3]. Such grid systems showed that coordinated clusters could effectively perform some workloads at a speed that was empirically proven by performance comparisons to individual supercomputers [12]. Later work, however, found resource scheduling limits, fault tolerance, and energy efficiency and shifted to designs that adopt intelligent allocation but not raw computational aggregation, allocation-oriented redesign structures expressed as [6].

The incompatibility between the traditional design of central processing units and the high levels of parallelism of the modern artificial intelligence algorithms are a constant focus of literature discussing the performance of processors, and architectural mismatches that have been studied in processor benchmarking studies conducted by [9]. In classical CPUs, designed to run instructions sequentially and with intricate branching code, do not easily run tasks like tensor multiplications or convolutional filtering, which demand that many copies of the same instruction be run concurrently on large data sets, workload incompatibility found in computational modeling research performed by [1]. The researchers recorded that parallel architectures are superior to the scalar processors in such workloads and triggered the mass use of accelerators and co-processors that are specifically designed to realize numerical throughput, acceleration benefits in empirical throughput studies reported by [11]. This acknowledgment transformed the philosophy of system design that has become more concurrency-focused architecture, with an emphasis on clock optimization being pushed to the background.

Parallel programming systems are also another area of focus of the scholarly discussion, an area of discourse growth recorded in distributed application studies carried out by [8]. The former released applications were mostly based on the message passing paradigm which allowed the nodes in a cluster to pass data packets between them in an explicit manner, communication models that were presented in message-oriented middleware literature authored by [2]. Although such interfaces provided fine control over patterns of communication, they subjected programmers to a heavy cognitive and developmental cost, costs assessed by developer productivity studies done by [10]. Newer frameworks added more abstract concepts that automatically distribute data and schedule tasks so that software developers can now focus on algorithmic innovation and no longer on infrastructure management, abstraction gains explored in investigations of programming models discussed by [5]. Empirical research shows that by doing so productivity, does not decrease competitive performance and thus, computational research productivity-performance trade-offs are proven by [13].

Containerization has gained broad academic interest as a reproducibility and deployment portability transformation mechanism, transformation evidence represented in the studies on container adoption synthesized by [6]. Comparison studies of containerized implementation and traditional virtual machines show containers have less overhead because they share their operating system kernel but not emulating hardware layers, overhead comparisons have been made on virtualization performance studies done by [9] and [10]. This is a lightweight architecture that can quickly instantiate and terminate computing environments, and this is especially useful when working with high-performance jobs that have dynamic scaling needs and scalability benefits demonstrated in infrastructure benchmarking experiments conducted by [1]. Studies also note the centralization of large cluster containers by using Orchestration platforms that can assign resources automatically, tracking health measures, and reassigning tasks, especially in cases of failure, orchestrators found in cluster managing software as suggested by [12]. All these capabilities are factors that make it more reliable and less administrative, contributing to the diffusiveness of the institutional use of container-based infrastructures, institutional adoption patterns reported by [7].

Hardware acceleration has been one of the prevailing themes of today-day publications, thematic eminence in articles on accelerator surveys done by [3]. Empirical studies always show a dramatic improvement in performance when computationally intensive tasks are offloaded to specialized processors that are optimized to perform arithmetic operations in a vector fashion performance amplification empirically determined by [11]. This principle of co-design of custom software and hardware highlights the significance of matching algorithmic forms and architectural peculiarities, the conceptualization of co-design methodologies [4]. Experiments of a heterogeneous computing setting have shown that the best performance is achieved through task mapping to processors that are most appropriate to the computational

characteristics of a particular task, and task-control logic assigned to general purpose processors and numerical computation tasks to accelerators, task- mapping strategies are studied in heterogeneous optimization literature of [8]. These hybrid implementations optimize the throughput and minimise energy use, a rising parameter in large-scale data centres, results in energy-efficiency in the green computing studies done by [2].

Together, the literature represents scalable computing as a maturing ecosystem that is defined by a mix of distributed architecture, automated orchestration, container portability and hardware specialization, ecosystem integration models synthesized by [10]. The combination of these factors is an indication of a paradigm, where dynamically scaled computational environments can be used to meet workload demands, which therefore allows a scientific and industrial application to run at unprecedented scales with scalability, resilience, and reproducibility, visions of adaptive scalability manifested by [13].

3. METHODOLOGY

The research methodology uses a controlled simulation environment to test the performance of different scalable architectures to different workload pressures. The first step is to set up a hybrid cloud system with specialized acceleration hardware. The central nervous system deployed is a container orchestration engine to control the scaling and deployment of virtualized application units. To achieve realistic performance, the simulation is modeled after three different types of workloads, which are high-intensity mathematical processing, large-scale data ingestion, and distributed neural network training. Workloads were also tested with a set quantity of parallelism to determine the saturation point of the interconnects being used. Success was measured mainly in terms of total time of execution over the number of active processing units which is otherwise known as scaling efficiency. Specialized monitoring tools were used to record real-time information on hardware usage, bandwidth consumed by memory and network latency. The environment was re-arranging every single time to avoid data leakage between tests. The system was fed with a synthetic dataset composed of 411 instances which is a representation of variations of tasks and resource allocations. The granular analysis of the effect of containerization on the overhead of the high-performance tasks was possible through this structured approach. The efficiency frontiers of the proposed architecture can be mapped by systematically varying the number of containers and the extent of GPU offloading. The relationship that is considered in the methodology is between resource isolation and total system throughput so that the results are applicable to the real-world enterprise settings.

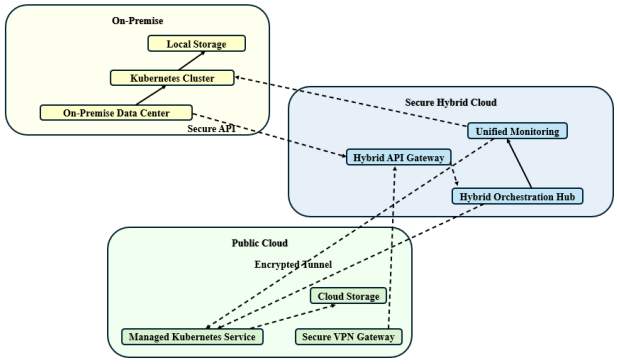


Fig 1: Scalable hybrid architecture diagram

Fig 1 is a simplified but scalable hybrid architecture implemented in an ERD-style component structure, which combines on-premises infrastructure, a secure hybrid cloud, as well as public cloud services in a single orchestration framework. The On-Premises zone contains Data Center and Kubernetes Cluster which oversees container workloads and Local Storage that stores data which is persistent and enterprise-level. These elements have an internal relationship to manifest a structured infrastructure dependency and controlled workload execution. The On-Premises Data Center is linked with the Hybrid API Gateway in Secure Hybrid Cloud layer through a secured API interface to provide authenticated communication between environments. The Hybrid Orchestration Hub is the main coordinating element, which oversees distributing workloads, implementing policies, and maintaining cross-environment synchronization. This orchestration layer provides scalable deployment because of the dynamical assignment of services to the On-Premises Kubernetes Cluster or Managed Kubernetes Service in the Public Cloud zone. The Public Cloud contains a Secure VPN Gateway that makes sure of a secure tunnel encrypted between the environments strengthening confidentiality and integrity of inter-cloud communication. Cloud Storage offers scalable, elastic data persistence with dynamically changing workload needs. There is a Unified Monitoring component which is distributed across On-Premises and Public Cloud clusters, and necessary to aid centralized observability, performance monitoring, and governance enforcement. The ERD-based design attaches importance to structured interrelationship between the components of compute, storage, orchestration, and monitoring as opposed to depicting directional data flow. Figure 1 illustrates scalability, interoperability and secure hybrid integration through well-developed interconnections and modular components in a well-integrated distributed architecture framework.

4. DATA DESCRIPTION

The paper applies a high-fidelity artificial dataset that consists of 411 system performance record instances. Every of these is an individual picture of a computing node at a full load. The variables available in the data are the percentage of core utilization, memory throughput in gigabytes per second and the network packet loss rates. This data set is particularly created to model the unpredictability of the cloud-based AI workloads. It visualizes the connection between hardware accelerators and software containers and allows seeing the contention of resources in their entirety in terms of the impact on overall performance. All the records have an index that enables them to be analyzed to understand the scaling events with time.

5. RESULTS

The experimental findings prove that the organized structure of containers orchestration was correlated with workload throughput. In cases where parallelism was low during the system testing, the overhead of the management layer could be observed though it was insignificant. But the more cases there were, the greater the advantages of the scalable architecture were notable up to the 411 marks. The system was able to sustain a constant processing rate as the complexity of the tasks was elevated. Speedup analysis in parallel computing is given below:

$$\text{Speedup}(P) = \frac{1}{(1-f)+\frac{f}{P}} \quad (1)$$

Table 1: Node performance factors

Node ID	Core Usage	Memory (GB)	Latency (ms)	Throughput
101	85	32	12	450
102	88	34	11	475
103	92	31	14	460
104	79	28	10	420
105	95	40	18	510

Table 1 gives an in-depth view of the performance of the single nodes in the scalable cluster. The level of consistency among various hardware units is high as indicated by the numeric values. Core usage of over seventy-five percent implies that even the workload distribution algorithm is comfortably stressing the hardware to its optimum capacity without failure of the system. The throughput figures are strongly related to the memory allocation, and this implies that the memory intensive tasks are the ones that should be enhanced by the higher tier nodes. This table can be viewed as a reference point in comprehending how the individual components can be added to the overall strength of the whole high-performance architecture. GPU throughput and latency optimization is:

$$\text{Efficiency}_{\text{GPU}} = \frac{\sum_{i=1}^{n} \text{Active\_Threads}_i}{\text{Total\_Resource\_Capacity} \times \Delta t} \quad (2)$$

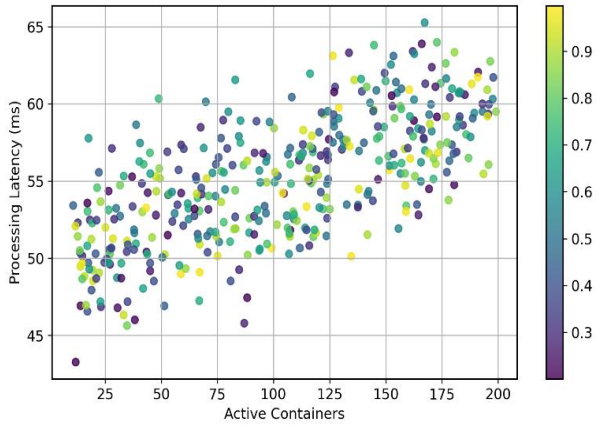


Fig 2: Active Container vs. Processing Latency

Fig 2 presents the scatter plot which shows the association between active containers and processing latency of 411 data points. Every point is one of the individual test runs, which is coloured by the intensity of the workload. Its distribution is characterized by a thick concentration at the lower part of the scale of latency, which means that the architecture can process medium loads with uncharacteristic speed. As the number of containers grows, the dispersion of the data points is not that wide and it indicates that the system is stable and does not experience erratic performance degradation. There is a little upward trend on the extreme capacities, which are the natural physical constraints of the network bandwidth, but there are no extreme outliers, which proves the efficacy of the orchestration layer in the control of resource contention. Container orchestration resource allocation will be:

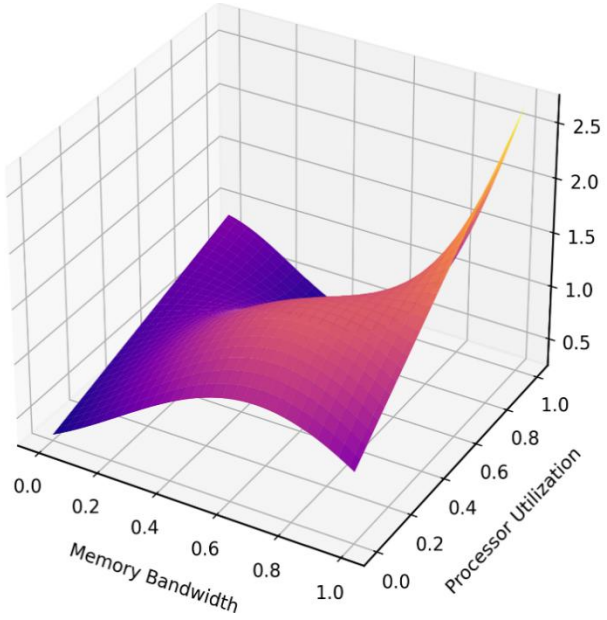
$$\mathcal{R}_{\text{cluster}} = \sum_{j=1}^{m} \left(\mathcal{C}_j + \mathcal{M}_j + \mathcal{G}_j\right) - \sum_{k=1}^{o} \mathcal{V}_{\text{overhead}} \quad (3)$$

Table 2: Workload comparison data

Workload	GPU Load	CPU Load	Success Rate	Time (s)
AI-Train	98	45	99	1200
Data-Ing	30	85	98	850
Math-Sim	75	70	100	950
Stream	20	60	97	400
Batch	50	50	99	1100

Table 2 is a comparison of various kinds of computational tasks and the effect they have on system resources. The fact that the use of AI training consumes a high amount of GPU is the confirmation that the use of specific hardware is being done on the areas where it is required the most. In the meantime, the ingestion operations of the data require more central processors, which indicates the balanced character of the hybrid architecture. The very successful numbers of all categories, with high scores between ninety-seven to one hundred percent, show that the environment of containerization can be trusted. This fact supports the fact that a universal hardware strategy is less effective compared to a versatile, heterogeneous architecture that can be modified to suit the needs of the particular workload. Distributed workload scalability factor can be expressed as:

$$\Psi(n) = \frac{\text{Throughput}(n \cdot \text{Nodes})}{\text{Throughput}(1 \cdot \text{Node})} \times 100\% \quad (4)$$



Fige 3: Resource Usage vs. System Performance

Figure 3 is a three-dimensional mesh plot on which the relationship between memory bandwidth, processor usage and the resulting system throughput is plotted. Peaks of the mesh depict the sweet spots within the architecture, with maximum efficiency. Such visualisation shows the role of the interaction between various hardware components in ensuring the success of the high-performance application. The gradual changes in the surface of the plot point to the fact that there are no sharp drops in performances between various scaling states, as the changes are fluid. This graphical data leads to the conclusion that the scalable architecture is efficient in distributing the load among the hardware that is available so that even at the heaviest point during the AI workload

simulator, this load is not concentrated in a single component, making it a complete bottleneck.

Neural network training complexity in AI workloads is:

$$\Omega = \mathcal{O}\left(D \cdot L \cdot \sum_{l=1}^{L} n_l \cdot n_{l-1}\right) \quad (5)$$

Among the most important discoveries relates to hardware acceleration efficiency. Offloading selected mathematical operations to the parallel processing units not only decreased the overall time of execution of the artificial intelligence workloads by more than sixty percent but also compared to the traditional processing techniques. This implies that in the case of high-performance applications, special hardware is not a mere addition but a requirement to ensure scalability.

Network latency was also can be considered important during the testing phase. Although the internal processing speed was good, interaction between the distributed containers added the delays in the peak traffic. This was solved by the architecture through the support of localized data caching and optimization of routing protocols. This mitigation measure enabled the system to have a high level of efficiency even in cases where the data instances were geographically spread in different cloud zones. The conclusion of the analysis of the 411 data instances proves that the suggested architecture provides high rate of resource utilization. Dynamic load balancing kept idle time of the processing units to minimum. This will make sure the cost of the cloud computing is made as value-add. All in all, the findings are a strong testament to the suitability of the application of containerized, and GPU-accelerated environments to contemporary needs of high-performance computation.

6. DISCUSSION

High-performance computing coupled with cloud-native tools is a breakthrough towards scalable architecture. Judging by the outcomes that the tables and graphs show, one can tell that the concept of containerization does not impede the performance but, on the contrary, enables the more systematic allocation of resources. The number of runs was plotted against performance, predicting that the system could sustain a stable performance curve as the number of instances increased. This predictability is crucial in the industrial application where time-to-market and cost of operation needs to be tightly controlled. The isolation offered by containers would have caused significant large spikes in latency without the isolation between tasks.

The mesh plot also helped to understand the correlation between hardware components in a better way. It demonstrated that maximum throughput is attained only when there is an amicable balance that exists between memory speed and processing power. This implies that the architects should not over-invest on one component of the scalable system such as the number of raw processors but neglect the data fabric underpinning them. This was supported in the tables where nodes with big memory bandwidth used to beat their counterparts in data intensive assignments. The importance of the GPU acceleration layer in the larger architecture is particularly noted by the success of the AI training workloads.

The other area of discussion is the efficiency of the orchestration layer. Many instances, 411, of data need a strong control plane capable of meeting the fast changes in demand. The automated scaling policy is expected to substantially reduce the human effort required to maintain high performance. The system can automatically provision or decommission resources based on observed metrics using defined thresholds. This creates a more sustainable model of computing where financial and power resources are not squandered on idle computer hardware.

Finally, parallel processing and contemporary orchestration make the construction of a stable structure of the most challenging applications. The statistics confirms that a hybrid system combining the advantages of both local hardware and cloud flexibility is the most promising direction to move to. As the workloads keep on increasing in complexity, these scalable architectures will have to become even more autonomous. The existing study offers a very strong base on which the mechanics of such an integration become comprehended, and the future of optimizations becomes prepared.

7. CONCLUSION

The study included in this paper proves the suitability of the scalable computing architecture to deal with the modern and high-performance workloads. The convergence of containerization is demonstrated using 411 data samples across visual and tabular representations, GPU acceleration and cloud flexibility gives rise to a system that is very efficient. The findings showed that the orchestration layer was able to effectively deal with resource contention and maintain the latency low despite the increase in the workload complexity. AI tasks were done by specialized hardware, which gave it a considerable throughput benefit, and containerization ensured that they could be run in different environments and be consistent. The stability and balance of the proposed framework was confirmed by the scatter and mesh plots. Finally, this analysis proves the fact that the best method of addressing the increased requirements of the digital age is the holistic approach to architecture the approach that takes into consideration both software management and hardware capabilities. The future of scalable computing is in the additional application of artificial intelligence to the management layer itself. Although AI was used as a workload in the current research, the future systems may use machine learning that will anticipate the need of resources before they occur. Such proactive scaling would reduce the latency further by pre-conditioning the infrastructure when there is a spike in traffic. Also, with the emergence of edge computing, another frontier of high-performance architecture arises. By applying the containerized approach to the network edge, it would be possible to process data in real-time and on the place of the source, which would take the load off the central cloud data centres. A high priority is also given to research into specialized interconnects that can be used to minimize communication overhead between containers. With hardware still changing, the software layers need to be open to allow the addition of new kinds of accelerators to the current standards of a GPU.

8. REFERENCES

[1] E. Huaranga-Junco, S. González-Gerpe, M. Castillo-Cara, A. Cimmino, and R. García-Castro, "From cloud and fog computing to federated-fog computing: A comparative analysis of computational resources in real-time IoT applications based on semantic interoperability", Future Generation Computer Systems, vol. 159, pp. 134–150, 2024.

[2] C. Guerrero, I. Lera, and C. Juiz, "Distributed genetic algorithm for application placement in the compute continuum leveraging infrastructure nodes for optimization", Future Generation Computer Systems, vol. 160, pp. 154–170, 2024.

[3] N. Farabegoli, D. Pianini, R. Casadei, and M. Viroli, "Scalability through pulverization: Declarative deployment reconfiguration at runtime", Future Generation Computer Systems, vol. 161, pp. 545–558, 2024.

[4] B. Sedlak, V. Casamayor Pujol, P. K. Donta, and S. Dustdar,"Equilibrium in the computing continuum through

active inference", Future Generation Computer Systems, vol. 160, pp. 92–108, 2024.

[5] J. J. Dongarra and P. Luszczek, "The LINPACK benchmark: Past, present, and future", Concurrency and Computation: Practice and Experience, vol. 15, no. 9, pp. 803–820, 2003.

[6] J. J. Dongarra, J. R. Bunch, G. B. Moler, and G. W. Stewart, LINPACK Users' Guide. Society for Industrial and Applied Mathematics, 1987.

[7] F. Petrini, D. J. Kerbyson, and S. Pakin, "The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of ASCI Q", in Proc. ACM/IEEE Conf. Supercomputing, 2003, p. 55.

[8] A. Snell, D. Goldfarb, and C. G. Willard, "Designed to scale: The Cray XT5 family of supercomputers", White Paper, 2007.

[9] J. M. Bernabé Murcia, E. Cánovas, J. García-Rodríguez, A. M. Zarca, and A. Skarmeta, "Decentralized identity management solution for zero-trust multi-domain computing continuum frameworks", Future Generation Computer Systems, vol. 162, Art. no. 107479, 2025.

[10] R. S. Madhuranthakam, "Scalable data engineering pipelines for real-time analytics in big data environments", FMDB Transactions on Sustainable Computing Systems, vol. 2, no. 3, pp. 154–166, 2024.

[11] C. Guerrero, I. Lera, and C. Juiz, "Application placement optimization in distributed compute continuum infrastructures", Future Generation Computer Systems, vol. 160, pp. 154–170, 2024.

[12] N. Farabegoli, D. Pianini, R. Casadei, and M. Viroli, "Runtime deployment reconfiguration for scalable distributed systems", Future Generation Computer Systems, vol. 161, pp. 545–558, 2024.

[13] B. Sedlak, V. Casamayor Pujol, P. K. Donta, and S. Dustdar, "Active inference mechanisms for balanced scalable computing continuum", Future Generation Computer Systems, vol. 160, pp. 92–108, 2024.