**Literature Revie**w

In recent years, computer systems with better processing power has become a necessary need in various sectors of the industry such as the financial, engineering, education and science aspects as they need to process a vast amount of data. With Artificial intelligence aiding more in data processing, the complexity of models and algorithms are becoming more heavily dependent on the process power. Without the appropriate equipment, certain research cannot be addressed or may take a lot of time to produce a result. Thus, missing out on potential contributions(Pfalzgraf & Driscoll, 2014) .

As the need for computation power grows, instead of investing in super computers which are expensive and have a difficult learning curve for non-expert programmers to use, one can invest in having multiple lesser systems working in parallel.

1. **Parallel Computing**

High performance computing (HPC) according to (Mauch et al., 2013) is currently the leading-edge disciplines in information technology that caters to demanding application in economy, science and engineering.

Parallel computing is considered to fall within these three main paradigms: 1) Cluster computing; 2) Grid computing; 3) Cloud computing. Cluster computing utilises a system of networked systems designed to solve a common task~\cite{valentini2013overview}. Thus, allowing the resulting system to share its resources to solve a computation complex problem that acts as one whole system. On the other hand, grid computing is a diversified system, with systems found in different locations instead on the same local area network and with the use of cloud computing, the evolutionary next step of grid computing, is a system which is offered as a service with dynamically allocated resources based on the demand and allow users to use such systems remotely.

1. **Multiple Environments and Virtualisations**

According to (Mauch et al., 2013) High performance computing systems are managed and operated by individual organizations in private. In cases, computational resources are underutilized or overloaded as the execution environment to run certain applications depend on specific libraries, schedulers and operating systems. Such a scenario limits other applications to be implemented on the same machine as it is only catering for a specific application to run with its libraries and operating system.

With the aid of virtual machines or using the concept of creating containers, one can create multiple environments which allows full computer utilization and can have multiple applications executed on the same machine. This allows full computer utilization and have multiple environments on the same machine.

(Chung et al., 2016) support the claim that was made by (Mauch et al., 2013) about specific application environment issues as they stated in their research that physical machines cannot handle a huge range of users with different operation environments simultaneously and that cloud computing and virtualization techniques such as virtual machines and containers can be used to tackle such issue.

**2.1 Virtual Machines**

Virtual machines, which are known as hypervisor-based, is a layer that allocates operation space of instances. As it follows the architecture of a real computer, there needs to be an interaction between software and hardware. There are three important interfaces which contribute to the role of manging resources which are the instruction set architecture (ISA), application binary interface (ABI) and application programming interface (API). (Huang et al., 2006)

**{ADD VIRTUAL MACHINE DIAGRAM}**

In virtual machines, a process or system running is on a ‘guest’ user protocol while the underlying platform that is executing the virtual machine is the ‘host’. Viewing the operating system and the application it supports, a virtual machine has a whole execution environment that enable it to perform many processes simultaneously as it can individually allocate memory and I/O resources to the process. However, it must emulate the instruction set architecture that the guest software can execute. Thus, allow complete isolation and security on the hosting device as virtual devices, privileged instructions and virtual memory address spaces are constructed.(Huang et al., 2006)

As the execution of the virtual machine is in complete isolation, every malicious code that is run within this guest OS only crashes the virtual machine, leaving the host machine intact without any repercussions.

For high performance computing utilization, virtual machines can offer such systems with complete isolation and security as:

1) hypervisor executes an entire environment to an instance. This provides complete isolation and as it is operated at a guest level, if a malicious code is run within that environment it will only affect that virtual environment, leaving the host machine intact. Whilst if the virtual machine crashes, the virtual machine monitor (VMM) can recover automatically;

2) Ease of management as virtual machines are created or deleted much easier than operating a physical machine.

3) Enable image customization as the virtualized software and emulated hardware can be modified before creating the virtual environment (Huang et al., 2006)

**2.2 Containers**

A second virtualisation approach would be the use of Linux Virtual Containers (LXC). Linux containers also addresses the requirements that are handled by virtual machines such as customization and isolations. However, it does not need to use the three interfaces (ISA, ABI and API) layers as containers operate in the kernel level to implement such features making it more light-weight and reduce overhead compared to virtual machines (Chung et al., 2016).

**{ADD CONTAINER DIAGRAM to show the difference}**

However, (Jacobsen & Canon, 2015) stated that operating at the kernel level can have consequences since the process running the containers on a system is running in a common Linux kernel. He stated that if a process or user disrupts the kernel, it can impact the other containers running on that system. In contrast, virtual machines isolate the hardware layer and as stated previously by (Huang et al., 2006), if a virtual environment crashes it does not affect the host machine or any other virtual machines running on that machine. Thus, virtual machines typically offer greater protection between applications running on a shared system.

On the other hand, as stated previously by (Chung et al., 2016), containers are more light-weight and has less overhead than virtual machines as each container only needs to run a single copy of the kernel whereas a virtual machine has its own instance of the kernel as can be seen in figure: **(Quote image for vm against containers).**

Although (Jacobsen & Canon, 2015) previous claim regarding kernel issues, he also agrees with (Chung et al., 2016) that using containers is ‘lighter’ in contrast to virtual machines. Also, they stated that as the kernel level is not virtualised, containers allow full I/O integration than virtual machines as there is no virtualization layer between them.

When (Jacobsen & Canon, 2015) compared these technologies in his research, they also stated that containers are created / reloaded on the fly in contrast to virtual machines as it only requires starting one process instead of initializing and booting an entire kernel which can take several minutes. This fast start-up can be very useful for high dynamic workloads that may need to shift resources between components or even fast re-execution of the container if it crashes or fails.

In addition, they also claim that accessing shared resources is also different between the two virtualisation models in high performance computing. As most applications need to have access to data stored in a global high-performance or in a share state between steps in a workflow, virtual machines must run an instance of the file system client. If a vast number of virtual machines are running on each node, this leads to an increase in the number of file system clients that the file system must serve. This adds complexity to manage and configure these virtual machines to access such file systems, as well as it can cause security concerns if the user has elevated privileges in the virtual machine environment.

On the other hand, a container can map through a file system from the host system into individual containers. Thus, there is only one instance of the file system client per host system. However, security still must be kept in mind depending on the implementation and user right. According to (Manu et al., 2016), If the container is compromised, then hackers/attackers would be able to get complete access to host OS and resources.

1. **Application Deployment and security**

As stated previously, both virtualisation models have their advantages and disadvantages that were stated by (Chung et al., 2016), (Jacobsen & Canon, 2015). In high performance computing according to (Higgins et al., 2015), Resource isolation is a problem in HPC systems as they execute a user’s job request within the same operating system environment that utilises the hardware directly to gain the best performance. This is problematic for application portability especially in grid systems where a remote resource may not have the required libraries or software required by the given job. To solve this issue, Docker containers offer an opportunity to create a cloud-like flexibility in the cluster without limiting performance.

As Linux Container system gained more popularity, Docker is an open source project that provides a systematic way to automate and deliver faster deployment of Linux applications inside containers. Docker makes use of container technology with kernel-end application levels as also was previously stated by (Jacobsen & Canon, 2015) and (Chung et al., 2016). These kernel-end applications run processes in isolation as docker uses namespaces to completely isolate an application’s view of the underlying operating environment, aiding in security between the container and the host machine.

According to (Manu et al., 2016), as docker containers are used for thin sandboxing they have become a suitable core candidate from majority of Enterprise ready to use real-time production environment used by Red, hat, Microsoft, IBM, HP, and google. Docker Inc discovered orchestration, a standard approach to manage the container. It is used as a hosting platform in clustering application and increases level of security that provides good performance in production deployment.

1. **Distributed training using TensorFlow**

Deep learning has become a powerful tool for solving problems not only in the industry but also in many scientific fields. As datasets continue to grow rapidly in size as well as in complexity, high performance computing is being utilised to address these challenges by adding more computational power as a resource. As an approach, distributed training has been introduced for deep learning models (Kurth et al., 2019)

According to (Kurth et al., 2019), he also stated that one unchallenged mantra of the HPC community is that to obtain scalable high-performance code, one must resort to classical HPC languages such as Fortran or C/C++ that utilises the efficient communication layers such as the Message Passing Interface (MPI). However, that has changed as most deep learning frameworks not utilize python as it provides flexibility to the model developer. Nevertheless, those frameworks still incorporate highly optimized back-ends written in C/C++ for good single node performance but still falls short as some frameworks are inefficient, if at all available.

**4.1 Tensorflow framework**

TensorFlow (Abadi et al., 2016), an open-sourced API, is one of the most widely used frameworks in recent years. Developed by Google in 2015 and is always maintained and updated by implementing results of recent deep learning research. Therefore, it allows support for a large variety of neural network layers, activation functions, optimizers and analytical tools for profiling and debugging. For TensorFlow to be able to deliver a good single node performance, compute heavy kernels such as convolutions and perform dense matrix multiplication. (add citation to google) (Kurth et al., 2019)

According to preliminary white paper published by google, the main components in a TensorFlow system are the client, which utilises the Session interface to be able to create communication between the *master*, and one or more worker processes. Each worker process is responsible for accessing one or more computational device (such as CPU cores or GPU cards) to be able to execute graph nodes on those devices as instructed by the master.

The TensorFlow interface also allows local and distributed implementations where;

1. The local implementation is used when the client, master and the worker all are executed on a single machine in the context of a single operating system process implementing multiple devices such as multiple installed GPU cards
2. The distributed implementation utilizes most of the code as local implementation however, it extends it by allowing support for an environment where the client, master and workers can all be implemented in different process on different machines. In the distributed environment, these different tasks are created as containers where jobs are managed by a cluster scheduling system. (add diagram below)

**{ADD TENSORFLOW DIAGRAMS for local implementation and distributed implementations}**

However, according to (Kurth et al., 2019), although TensorFlow provides a built-in mechanism for distributed training based on Google’s GRPC protocol. This protocol is not suited for HPC architectures as it requires at least one server and utilizes network protocols such TCP/IP which is designed for internet traffic. This messaging protocol has a larger header and thus require more bandwidth than traditional HPC interconnected messages. Furthermore, they are forced into 64K frames and always rerouted through the kernel.

Google’s white paper does in fact mention that the communication layer that communicated with worker processes use remote communication mechanisms such as TCP or RDMA to move data across machines. However, this allows for devices not needing to be on the same network and can implement scalability through cloud computing.

On the other hand, TensorFlow also implements fault tolerance where failures in a distributed execution can be detected in a variety of places. When a failure of any kind is detected, the entire graph execution is aborted and restarted from the beginning. However, variable nodes refer to tensors that persist across executions. Also, consistent checkpointing can be used by connecting to a Save Node. These saved nodes are executed periodically every N iteration. When executed, the contents of the variables are written to persistent storage like a distributed file system. Similarly, each variable can be connected to a restore node that is only enabled in the first iteration after a restart.

1. **Data Parallel Training**

Data parallelism has become the predominant parallelization due to its simplicity; it works by splitting the input data across the nodes and each node computes the forward pass fully independently on his shard of the batch data. With the use of the loss obtained from the forward pass, each node will then perform the back-propagation process as it computes the relevant gradients locally. Before being incorporated into the model by the solver, the gradients will be averaged across all nodes and in each update process the local gradient computation can process without waiting for the reduction to be complete. Only then, before starting the next forward pass for the given layer, the gradient reduction and incorporation must be finished. In deep and compute intensive models, it is possible to hide a big fraction of the communication time behind computation. This method of data parallel training is called fully synchronous training, the model and gradients are always kept in sync between the nodes.(This is like the all-reduce method, all nodes must be in sync with each other and has to wait for all to be finished before continuing)(Abadi et al., 2016)

(Add diagram to explain this process)

Another method of distributed data parallel training is called asynchronous training. This method requires a parameter server that keeps track of the model weights and received gradient updates from the individual nodes and applies them into the model and distributes the updated nodes in the order as they are received. This approach allows much more scalability than the previous stated method as it does not need global synchronization involved. However, the stochastic convergence of the loss can be impacted negatively if there are many nodes that are participating in the asynchronous update. This is because the number of outdated gradients incorporated in the model can grow linearly where ***N*** is the number of participating nodes. (Kurth et al., 2019)

1. **Parallel Computing in education**

As discussed previously, parallel computing is a need in industries that handle big data and solve complex algorithms. However, it is a complex topic and is not implemented in most academic institutions(Pfalzgraf & Driscoll, 2014). The reason being that the costs related to setup and maintenance of such a system are the causing factors for an academic institution not to implement such a system for students to use.

The University of Helsinki successfully created a private cloud computing cluster that cost them up to one million euro (Abrahamsson et al., 2013). Apart from also having a technical complexity, the space also must be considered as such systems need a substantial amount of spacing, cooling and the constant maintenance needed that add to the hindering factors.

(Goscniski et al., 2013) remarked that academic institutions already have “readymade parallel computers” as they have computers which are very often idle or lightly loaded that are not used after school hours. Thus, clustering such machines together can provide students and staff the ability to potentially raise more workload and produce far better results. However, he also noted that it is not easy to manage such a system as developers must identify the computer of a cluster that are suitable and available for such application while also tracking fault computers.

On the other hand, instead of using computers, (Abrahamsson et al., 2013), (Cox et al., 2014), (Kaewkasi & Srisuruk, 2014), (Doucet & Zhang, 2019) have presented single-board device based solution by using Raspberry PIs, a credit-card sized minicomputer that utilises a low-power ARM chipset. The use of such systems allows the potential for institutions to implement parallel computing as failure of such a device leads to a replacement of a low-cost components rather than very expensive equipment such as super computers and servers.

However, (Abrahamsson et al., 2013) stated that even though there is a cost benefit to implementing such a solution, there are also limitations in the overall performance that can be achieved by such single-board devices. Other limitations that these devices present are the read/write operations as they use an SD-Card instead of a hard-disk and the network transfer is not the same as using a dedicated NIC (Network Card) on machines and servers.

# Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*.

Abrahamsson, P., Helmer, S., Phaphoom, N., Nicolodi, L., Preda, N., Miori, L., Angriman, M., Rikkilä, J., Wang, X., Hamily, K., & others. (2013). Affordable and energy-efficient cloud computing clusters: The bolzano raspberry pi cloud cluster experiment. *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, *2*, 170–175.

Chung, M. T., Quang-Hung, N., Nguyen, M., & Thoai, N. (2016). Using Docker in high performance computing applications. *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, 52–57. https://doi.org/10.1109/CCE.2016.7562612

Cox, S. J., Cox, J. T., Boardman, R. P., Johnston, S. J., Scott, M., & O�brien, N. S. (2014). Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing*, *17*(2), 349–358.

Doucet, K., & Zhang, J. (2019). The Creation of a Low-cost Raspberry Pi Cluster for Teaching. *Proceedings of the Western Canadian Conference on Computing Education*, 7:1--7:5. https://doi.org/10.1145/3314994.3325088

Higgins, J., Holmes, V., & Venters, C. (2015). Orchestrating Docker Containers in the HPC Environment. In J. M. Kunkel & T. Ludwig (Eds.), *High Performance Computing* (pp. 506–513). Springer International Publishing.

Huang, W., Liu, J., Abali, B., & Panda, D. K. (2006). A Case for High Performance Computing with Virtual Machines. *Proceedings of the International Conference on Supercomputing*, 125–134. https://doi.org/10.1145/1183401.1183421

Jacobsen, D. W., & Canon, S. (2015). *Contain This , Unleashing Docker for HPC*.

Kaewkasi, C., & Srisuruk, W. (2014). A study of big data processing constraints on a low-power Hadoop cluster. *2014 International Computer Science and Engineering Conference (ICSEC)*, 267–272.

Kurth, T., Smorkalov, M., Mendygral, P., Sridharan, S., & Mathuriya, A. (2019). TensorFlow at Scale: Performance and productivity analysis of distributed training with Horovod, MLSL, and Cray PE ML. *Concurrency and Computation: Practice and Experience*, *31*(16), e4989. https://doi.org/10.1002/cpe.4989

Manu, A. R., Patel, J. K., Akhtar, S., Agrawal, V. K., & Murthy, K. N. B. S. (2016). A study, analysis and deep dive on cloud PAAS security in terms of Docker container security. *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 1–13. https://doi.org/10.1109/ICCPCT.2016.7530284

Mauch, V., Kunze, M., & Hillenbrand, M. (2013). High performance cloud computing. *Future Generation Computer Systems*, *29*(6), 1408–1416. https://doi.org/https://doi.org/10.1016/j.future.2012.03.011

Pfalzgraf, A. M., & Driscoll, J. A. (2014). A low-cost computer cluster for high-performance computing education. *IEEE International Conference on Electro/Information Technology*, 362–366.

# 