International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

High Performance Computing (Hpc) in the Cloud: A Proactive Fault Tolerance (Pft) Strategy

[1]Sunil Sharma, [2]Garima Jain, [3]Preethi D, [4]Shambhu Bhardwaj

Submitted:20/04/2023 **Revised:**12/06/2023 **Accepted:**23/06/2023

Abstract: The High Performance Computing (HPC) applications benefit from the new paradigms for computers, capacity, and adaptable responses provided by cloud computing. For instance, the Hardware as a Service (HaaS) paradigm enables individuals to provide several Virtual Machines (VMs) for applications that need a lot of computing. Any execution error would require re-running applications, which would waste time, money, and energy since the HPC system on the cloud uses a lot of VMs and electrical components. In this research, the execution time on the clock and the cost when mistakes occur, we provided a Proactive Fault Tolerance (PFT) strategy to High Performance Computing systems in the cloud. Additionally, we created an enhanced PFT technique for cloud-based HPC systems. Before predicting a failure, our approach does not depend on a spare node. Also, we created a model cost for running computing-heavy apps on cloud HPC servers. To evaluate the effectiveness of our strategy, we looked at the monetary costs associated with supplying spare nodes and checkpointing PFT. Our experimental findings from a genuine cloud execution environment demonstrate that executing computation-intensive apps in the cloud may lower costs and execution times by up to 30%. Our PFT technique for HPC in the cloud may minimize the occurrence of checkpointing of computation-exhaustive applications by up to fifty percent when compared to existing PFT approaches.

Keywords: *High Performance computing, Cloud computing, computation-intensive, Proactive Fault Tolerance.*

1. Introduction

Computing resources are made available as services by cloud service providers like Amazon and bare-metal cloud. Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Hardware as a Service (HaaS) are different categories that apply to these services [1]. The full computational capacity of the device is accessible to consumers thanks to HaaS. Some of the VMs running on the hardware and the operating system (OS) are both controlled by the users. For applications requiring a lot of processing and data, research groups may simply lease HaaS and customize HPC servers to meet their specific requirements [2]. As a result, programs that need a lot of processing and were previously only performed on specialized supercomputers may now be operated in shared cloud settings. When not in use, these resources may be given up. But one of the biggest problems that cloud services now confront is Proactive Fault Tolerance (PFT) [3]. A failure happens when a piece of hardware breaks and has to be replaced, when a node must be stopped or restarted, or when a piece of software cannot finish running [4]. This will fail any applications using the failed component.

However, differential counting of blood cells requires a lot of time and effort if it is exclusively done by humans [6]. Additionally, the outcomes may vary based on each expert's own subjective viewpoint. Although the automated cell counters that are now on the market are based on the concepts of flow cytochemistry and laser light scattering, of all analyzed blood samples, 23% need expert microscopic analysis [7]. Consequently, several attempts to create automated cell analysis systems employing image processing have already been established. Hence In this paper we introduce Enhanced Naive Bayes-Ant Colony Optimization (ENBACO) for classified Blood cells.

Additionally, VMs which are used to run HPC applications in cloud settings, have a higher failure rate owing to resource sharing and competition. Because PFT technology can prevent restarting, it is crucial for HPC systems operating in cloud settings because doing so lowers operational expenses and energy use [5]. To offer PFT in the event of hardware failures, hardware redundancy is employed. Another element that is in excellent working condition continues to function in the case of a hardware fault of one component up until the defective part is replaced. To enable the HPC systems to withstand failures,

1Assistant Professor, Department of Computer Science & Engineering, Vivekananda Global University, Jaipur, India, Email Id: sunil.sharma@vgu.ac.in
2Assistant Professor, Department of Computer Science and Business Systems (CSBS), Noida Institute of Engineering and Technology, Greater Noida, Uttar Pradesh, India, Email id: garimajain@niet.co.in
3Assistant Professor, Department of Computer Science and IT, Jain (Deemed-to-be University), Bangalore-27, India, Email Id: preethi.D@jainuniversity.ac.in
4Associate Professor, College of Computing Science and Information Technology, Teerthanker Mahaveer University, Moradabad, Uttar Pradesh, India, Email id: shambhu.bharadwaj@gmail.com

redundant computing nodes are added using hardware redundancy PFT methods [6]. A redundant computing idea for HPC systems. All of the compute nodes are duplicated in redundant computing. The application of the message-passing interface (MPI) is the main topic of this study. A parallel programming standard called MPI enables messages between jobs running concurrently on several processors or virtual machines [7]. It has two operating modes: running and failing. Because scalable and widely accessible HPC systems may be created on the cloud, MPI applications like the Groningen Molecular Simulation and Modeling Machine apps can substantially advantage from them. The findings of a thorough dollar cost study of several PFT approaches are presented in this research.

2. Related Works

The research [8] employed an approach to High Performance computing known as evolutionary multi-objective optimization (EMO) methods to address these MOPs. To show that our method may be used in the real world, we give a case study including floor layout. The study [9] proposed a preliminary task-focused taxonomy for High Performance computing (HPC) technologies, including a classification of programming interfaces and runtime mechanisms. By characterizing advanced task-based contexts, we show how our taxonomy might be beneficial. The article [10] presented the Mochi technique and framework. The fundamental building blocks and microservices of Mochi are outlined. It describes four case studies in which the Mochi approach was used to provide highly specialized services.

The research [11] introduced the theories of Diffusion of Innovation (DOI) and Human-Organization-Technology fit (HOT-fit) to investigate the influence of 10 variables on cloud computing adoption choices in the High Performance Computing (HPC) setting. The paper [12] analyzed the efficacy and efficiency of novel audio-video early fusion, slicing, and sampling techniques, and presents a side-by-side comparison of multiple 2-Dimensional Convolutional Neural Network (2D-CNN) video action detection algorithms. The study [13] offered a new method that makes use of the ACS algorithm developed for use in cloud computing. By using Map Reduce, classical ACS may be made parallel, the solution issue can be processed in a distributed parallel manner, and ACS's flaws can be addressed. The research [14] explored how a supercomputing or cluster-based computing environment may be used to create a DNA cryptography-based secured weather prediction model. The work [15] discussed the use of High Performance computing on the Google Cloud Platform to quickly and effectively analyze massive volumes of traffic data on demand during a crisis.

3. Materials and Methods

The application layer, platform layer, infrastructure layer, and hardware layer are the four levels that make up the architecture of cloud computing, as shown in Figure 1. Through the Internet, each of these levels offers users of the cloud a variety of services.

Users may access SaaS via the application layer. Most cloud users use this tier, which is why. User-owned devices like laptops and iPads may access applications over the Internet.

Platform layer implementation of PaaS. Platforms like Visual Studio are made available to Azure developers by a PaaS cloud provider. PaaS enables developers to create, distribute, and test cloud-based software. IaaS is utilized as a virtualization tool on top of the hardware layer. To supply computer resources like VM and storage, it takes advantage of virtualization technologies like the Xen hypervisor.

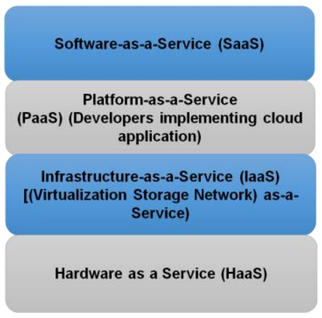
Software-as-a-Service (SaaS)
Platform-as-a-Service (PaaS) (Developers implementing cloud application)
Infrastructure-as-a-Service (IaaS) [(Virtualization Storage Network) as-a-Service)
Hardware as a Service (HaaS)

Fig.1. Architecture of cloud computing

The physical hardware that contains the operating system and other components makes up the hardware layer. The virtualization features are component of HaaS. Users may sign up for HaaS since it allows them complete control over the server and the number of VMs to run on it to optimize performance.

The architectures of cloud computing enable cloud providers to provide consumers with specific services. Utilizing the aforementioned cloud services might result in considerable savings for certain consumers. Particularly, HPC systems in the cloud may be used to run computation-intensive applications, with advantages including scalability, a pay-as-you-go pricing mechanism, and accessibility. Additionally, these services may be accessible via the Internet in a variety of ways and at any time.

3.1. Image PFT for cloud-based HPC systems

The cloud reduces initial expensive capital expenditures in hardware and infrastructure purchases by offering pools of computer services through networking utilizing a pay-as-you-go service pricing model. Up until recently, most research communities lacked access to HPC systems. For their computationally demanding programs that formerly ran in specialized HPC settings, research, and academic groups may now make use of the cloud pricing model. Cloud customers are in charge of setting and maintaining the services, whereas HaaS suppliers rent out the 'bare bones' gear, like data servers, computers, and storage. When users desire complete control of the OS, over the server, the software stack, and the several VMs, they may use HaaS. When there is additional comprehensive control over the programs that run in the HPC management, performance and other sorts of trade-offs may be made more readily. At this level, PFT methods are not often provided by cloud service providers.

PFT employs an avoidance strategy to put up with flaws. This is accomplished by using health monitoring and system log tools. Data on the status of the software and hardware is available through the system log and health monitoring. PFT communities have lately been interested in hardware health monitoring since sensors are now put on current gear to track things like CPU temperature and fan speeds. Future failures are predicted using this data. Four different kinds of modules are needed for our proactive PFT for cloud-based HPC systems: a controller module, a PFT policy module, a failure predictor, and a node monitoring module with an lm-sensor. The sections that follow provide explanations of these.

3.2. Monitoring nodes with lm-sensors

Modern CPUs include sensors built in that may be used to keep track of many characteristics, including CPU temperature, memory use, fan speeds, and another hardware issues. The performance and dependability of systems might suffer from variations in the monitored parameters. The lm-sensors package provides tracking tools, libraries, and drivers these metrics, are what we utilize. The monitored parameters' values are accessed using the lib sensors library. It offers user-space support for the console tools that provide sensor readings as well as the hardware monitoring drivers. Sensor limitations may be easily established using Lm-sensors. We chose lm sensors because they employ Linux OS kernel drivers, which are present in the majority of HPC systems. We created an FTDaemon with lm-sensors that is simple to install on a cloud-based HPC server. However, our techniques are readily transferable to different OS systems.

Centralized node health monitoring adds substantial network overhead to an HPC system with more than 100,000 cores. Our approach was designed to lessen the need for constant monitoring by requiring just periodic readings of hardware characteristics from each node. In our prototype, the FTDaemon on every compute node polls lm-sensors every 600 ms. Every time the observed parameters go beyond the most defined values, an alert is sounded off. The alert triggers the calculation to ascertain if a failure is likely to occur and the reading of the sensor readings.

Algorithm 1: The rule-based prediction method

```
Step 1: start timing True
Step 2: every computing node's FTDaemon while timer === True #Step 3: do
Step 4: parameters read $D_j U$
Step 5: computing for $D_j U_x$
Step 6: if $D_j U_x = 1$then;
Step 7: Break
```

3.3. Fault predictor

Each node's user space hosts the FTDaemon process. It employs strategies for rule-based prediction. Reading the sensor data regularly allows for the prediction of potential failure scenarios. The values at this moment are contrasted with the predetermined maximum operating conditions. For instance, we gave the weights of the normal, maximum, and critical values of 21, 0, and 1, respectively. To evaluate if a failure is likely to happen soon, current sensor readings are compared to maximum predefined criteria. In Algorithm 1, the rule-based prediction method is shown.

Let's pretend we've bored a hole in every surface minimum and are going to use that hole to fill several catch basins. Dams are constructed when additional immersion would cause water from separate catchment basins to mix. The water level will rise to a point when just the dam's peak is visible. To solve the problem of overlapping cells, the space transform of the binary mask of the cells with the largest area is applied to the watershed transform. The outcome of the watershed segmentation of the blood cell picture is shown in Figure 5.

3.4. PFT Proactive Fault Tolerance (PFT) policy

The PFT policy seeks to minimize how a failure may affect the running of a computationally demanding application. We created and put into effect three policies:

(1) the service provider for a further node to rent

(2) Get rid of the bad node

(3) Send a request for action to the administration.

When failure is anticipated, the FTDaemon may either go through with renting a new node or notify the administrator. The standard procedure is to lease a second node and notify the head host of the new node's specifics. All nodes are listed in a database kept by the head host. In the case that the head host fails as expected, the functionality is moved to a freshly leased node. After moving the unhealthful VMs to the recently leased node, the second policy, "relinquish the unhealthy node," is put into action.

3.5. Controller module

A controller module carries out the aforementioned regulations. On every node, a controller module is inserted. The node that is going to fail may now take prompt action thanks to this. When a failure is expected, the FTDaemon activates this regulating module. The supplier of services is contacted by the controller module and given the necessary credentials (such as a username and password) for the leasing procedure. After leasing the extra node, it performs a live transfer of the virtual machines from the malfunctioning node to the just rented node. Additionally, it records information about the extra node on the head host. A controller module additionally installs the newly leased node's FTDaemon when the VMs have been successfully migrated. Each node has the virtualization program Xen hypervisor loaded. This enables the installation of numerous par-virtualized OSs on every node. The host OSs is Dom00... Dom0n and domain zero. They control the administration interface and have unique access rights to the hardware. The host OSs is where FTDaemon runs. Through the drivers, FTDaemon and the backend talk to the hardware. The guest VMs are located in the unprivileged domains DomU0 and DomUn. The guest virtual machines are set up to create a cluster. Applications that need a lot of computing are run by guest VMs.

4. Result and Discussion

In cloud computing, the cost-oriented PFT in the cloud is crucial. Users may choose an appropriate PFT strategy for operating the apps in cloud management at the lowest possible cost by having a solid grasp of the cost implications and dependability of HPC systems in cloud computing. From the perspective of cost management, it is a technique utilized to reduce expenses and choose PFT by a Project cost. It is also beneficial to evaluate various PFT solutions and cloud computing resources to reach a certain dependability level. To run computationally demanding apps on cloud HPC servers, we construct the D_{db} cost model. With the help of the following cloud computing characteristics, the price may be calculated. We examine the aforementioned parameters and demonstrate that we derived the model cost in the Configuration Cost and Execution Cost sections below.

4.1. Configuration cost

The term "configuration cost" usually refers to the costs of setting up and maintaining the hardware and software infrastructure necessary for HPC operations. This cost may vary greatly based on a variety of variables, including the size of the HPC system, the complexity of the design, and the unique needs of the HPC applications. The installation and setup costs are modeled as

$$D_{jm} = D_0 + \sum_{h=1}^{n} D_{jh} + \sum_{g=1}^{m} D_{jg}, \qquad (1)$$

The initial cost for setting up a typical High Performance computing system environment in the cloud is D_0.where D_jg (h = 1,2,3,...,n) when they surpass the requirement, the unit costs to install and configure each computational node, and D_jg (g = 1,2,3,...m) are the unit costs to set up and deploy the storage resources. High Performance computing tests are often carried out on systems with 4 to 32 compute nodes. In this instance, the "standard" included Linux OS, 32 computing nodes, and MPI management. To properly investigate scalability, we may need a system with thousands of nodes. Figure 2 and Table 1 show that our proposed method PFT is lower when compared to some of the current approaches, including CNN, SVM, and LSTM.

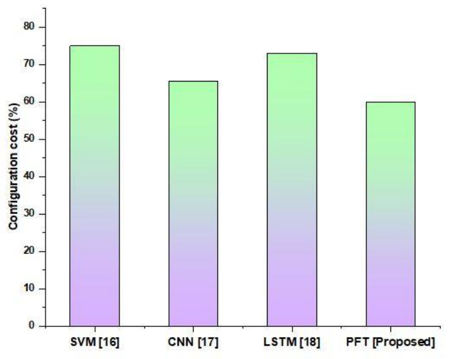

Fig.2. Comparison of configuration cost with our proposed and existing methods.

Table 1. Comparison of configuration cost

Methods	Execution cost (%)
SVM [16]	85.76
CNN [17]	82.72
LSTM [18]	60.78
PFT [Proposed]	52.75

4.2. Execution cost

The Execution cost is the expense spent when running a computationally expensive program on cloud computing resources. Typically, the cost of execution is calculated by the hour or the month. This may be expressed as the product of the price of a compute node that is leased and the length of time it takes for a computation-intensive application to execute in a cloud-based HPC system. The execution cost is represented by:

$D_f = \sum_{j=1}^{o} D_{fj} \,.\, F_{sj}$ (2)

Where $j = 1, 2, 3, \ldots and\ o$ is the total quantity of compute nodes utilized to run the computation-intensive apps D_f stands for the dollar rate per computing node. The execution time of the computationally demanding application on the HPC system D_{fi} is denoted by the notationF_{sj}.

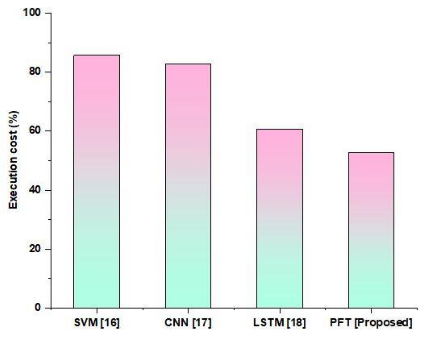

Fig.3. Comparison of Execution cost with our proposed and existing methods.

Table 2. Comparison of Execution cost

	Execution cost (%)
SVM [16]	85.76
CNN [17]	82.72
LSTM [18]	60.78
PFT [Proposed]	52.75

Figure 3 and Table 2 show that our proposed method PFT is lower when compared to some of the existing methods such as CNN, SVM, and LSTM.

4.3. Failure cost

Failure costs in High Performance computing (HPC) refers to the expenditures spent when unanticipated faults or mistakes occur in the HPC system or during the execution of computational operations. These expenditures might come from a variety of sources and have both direct and indirect effects on HPC operations. Here are some elements that contribute to HPC failure costs:

$D_e = D_{loss}[1 - Q_y]$ (3)

$D_{db} = D_0 + \sum_{h=1}^{n} D_{jh} + \sum_{g=1}^{m} D_{jg} + \sum_{j=1}^{O} D_{fj}.F_{sj} + \sum_{i=1}^{o} D_{di}^{jm} + \sum_{l=1}^{q} D_{dl}^{out} + \sum_{k=1}^{t} D_{tk}.D_{sk} + \sum_{v=1}^{w} D_{qv} \,.\, D_{sv} + D_{loss}\,[1 - Q_y]$ (4)

The following may be inferred from Equation (4):

- The cost of operating computation intensive apps in HPC systems in the cloud computing is increased by the use of compute node redundancy FT methods, which are ubiquitous in conventional HPC systems.
- The cost of operating the computation-intensive apps rises when checkpoint and restart FT methods are used in cloud-based high-performance computing (HPC) systems, yet this enhances system dependability. This is because using checkpoints and restarts in FT causes the application's wall-clock execution time to grow. These accords with previous findings about the impact of checkpoint and restart FT methods.
- It is possible to increase the availability of HPC systems and lower the cost of running computationally demanding applications by performing a live migration of Virtual Machines (VMs) from sick nodes to unneeded nodes before a failure prediction is generated.

4.4. The Algorithm and Economic Impact of Proactive Fault Tolerance

We present our method and offer a numerical mathematical analysis of its characteristics in this part. The condition of the monitored parameters is ascertained using the most recent data available from the sensors. The algorithm anticipates failure in the future and takes measures to lessen its effects on the application. It then installs an FTDaemon on the newly leased node and releases the unhealthy node as well. Here is the description of Algorithm 2**.**

Algorithm 2

```
# Every computer node is running FTDaemon. $D_j (j = 0, 1, \ldots, m)$;
# Observable parameters: $U =$fan speed, temperature voltages;
# Factors ¼ functions $U_X$: weight (– 1, 0, 1),
# Where
#–1 = functions with all parameters set to their default settings. U;
# 0 = functions at the maximum levels of one or more values;
# 1 ¼ functions at the difficult cost; $D_j U = 1$
# For compute node$D_j U$;
Step 1: FTDaemon:
Step 2: Start
Step 3: note the hostnames of all active guest VMs on the node. $D_j$
Step 4: set difficult values of$D_j U$;
Step 5: set timer True;
Step 6: while True do;
Step 7: Factors  read$D_j U$;
Step 8: computing for $D_j U_x$;
Step 9: if $D_j U_x = 1$then;
Step 10: break; # exit loop
Step 11: else if $D_j U_x = 0$ then;
Step 12: Record the max U;
Step 13: Get delayed;
Step 14: else
Step 15: see whether an alarm trigger has been received;
Step 16: end while;
Step 17: controller module:
Step 18: lease another node;
Step 19: actual migration of, $UN_1, \dots, UN_m$
Step 20: FTDaemon installation on a freshly leased node;
Step 21: provide the head host information about a freshly leased node.;
Step 22: Get rid of the unhealthy node;
Step 23: End
```

Case 1

First, we utilize Equation (4) to determine the overall cost of the proactive PFT technique employed when a spare node is supplied before failure prediction. Due to the price of the spare nodes, using this architecture will result in a somewhat high cost for operating computation-intensive apps in computing.

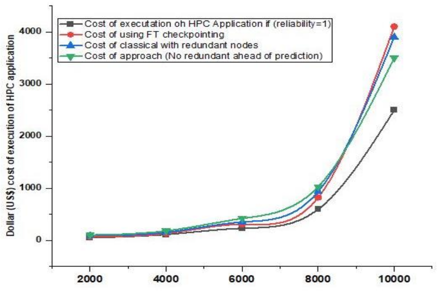

Fig.4. Performance of HPL benchmarks with and without check pointing and with FTDaemon.

Figure 4 and Table 3 display the cost, which has been calculated.

Table 3. Check pointing, non-check pointing, and FTDaemon-enabled HPL benchmark performance.

	Dollar (US$) cost of execution of HPC application			
	Cost of execution of HPC Application if (reliability=1)	Cost of using FT checkpointing	Cost of classical with redundant nodes	Cost of approach (No redundant ahead of prediction)
2000	55	75	95	105
4000	115	135	155	185
6000	235	310	355	423
8000	600	825	936	1032
10000	2500	4100	3900	3500

Case 2

A setup is created using the compute node $D_j\, U_x$ =1 (as previously mentioned). There is no need to have a spare node in this condition. The majority of the time, according to observances and documentation, HPC systems function in this zone, except when the fault is likely to happen [when a node approaches its critical state, or CiVw 14 1].

$$D_q = \sum_{w=1}^{x} D_w \,.\, Q_{sx} \approx 0 \qquad (5)$$

$$D_{db} = D_0 + \sum_{h=1}^{n} D_{jh} + \sum_{g=1}^{o} D_{fj} + \sum_{i=1}^{r} D_{ig}^{jm} + \sum_{l=1}^{q} D_{dl}^{out} + \sum_{k=1}^{t} D_{tk} \,.\, D_{sk} + D_{loss}[1 - Q_v] \qquad (6)$$

Only in this stage does the controller model give up the sick node and lease a new node from the service provider. We assume that there is ample time to move virtual machines from the sick node to the recently leased node between the forecast of node failure and actual failure. The running cost of the backup node is almost nil since the sick node is abandoned right away after the VM transfer. Our test system's supply of a node and live VM migrations within 30 seconds, according to the findings of our experiments.

5. Conclusion

In this work, we detail the planning and execution of a proactive PFT strategy for HPC in the cloud, with a focus on minimizing costs. The cost model for executing applications requiring extensive computational resources on cloud-based High Performance computing systems was developed by us. We looked at how much money it would cost to have extra nodes available before failure was predicted. We demonstrated that a pre-emptive supply of spare nodes is not necessary for our strategy to work. We showed the outcomes of our experiments conducted in a genuine cloud setting. The experimental findings demonstrate that the suggested proactive PFT approach to HPC systems in the cloud may dramatically increase the execution time of computation-intensive applications, resulting in a reduction of the dollar cost for operating them by as much as 30%. Our FTDaemon can also cut in half the time it takes for computation-intensive programs to do checkpointing. In the event of the breakdown of one or more computing nodes, our method may assist minimize energy usage by decreasing the clock execution time of computationally expensive HPC programs.

References

[1] Wada, I., 2018. Cloud computing implementation in libraries: A synergy for library services optimization. International Journal of Library and Information Science, 10(2), pp.17-27.

[2] Negru, C., Mocanu, M., Cristea, V., Sotiriadis, S. and Bessis, N., 2017. Analysis of power consumption in heterogeneous virtual machine environments. Soft Computing, 21, pp.4531-4542.

[3] Kumari, P. and Kaur, P., 2021. A survey of fault tolerance in cloud computing. Journal of King Saud University-Computer and Information Sciences, 33(10), pp.1159-1176.

[4] Jadhav, S. B. ., & Kodavade, D. V. . (2023). Enhancing Flight Delay Prediction through Feature Engineering in Machine Learning Classifiers: A Real Time Data Streams Case Study. International Journal on Recent and Innovation Trends in Computing and Communication, 11(2s), 212–218. https://doi.org/10.17762/ijritcc.v11i2s.6064

[5] Gunawi, H.S., Suminto, R.O., Sears, R., Golliher, C., Sundararaman, S., Lin, X., Emami, T., Sheng, W., Bidokhti, N., McCaffrey, C. and Srinivasan, D., 2018. Fail-slow at scale: Evidence of hardware performance faults in large production systems. ACM Transactions on Storage (TOS), 14(3), pp.1-26.

[6] Bharany, S., Badotra, S., Sharma, S., Rani, S., Alazab, M., Jhaveri, R.H. and Gadekallu, T.R., 2022. Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy. Sustainable Energy Technologies and Assessments, 53, p.102613.

[7] Ashraf, R.A., Hukerikar, S. and Engelmann, C., 2018, March. Shrink or substitute: handling process failures in HPC systems using in-situ recovery. In 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP) (pp. 178-185). IEEE.

[8] Raguntha r, T., Ashok, P., Gopinath, N. and Subashini, M., 2021. A strong reinforcement parallel implementation of k-means algorithm using message passing interface. Materials Today: Proceedings, 46, pp.3799-3802.

[9] Wang, G.G., Cai, X., Cui, Z., Min, G. and Chen, J., 2017. High Performance computing for cyber-physical social systems by using an evolutionary multi-objective optimization algorithm. IEEE Transactions on Emerging Topics in Computing, 8(1), pp.20-30.

[10] Thoman, P., Dichev, K., Heller, T., Iakymchuk, R., Aguilar, X., Hasanov, K., Gschwandtner, P., Lemarinier, P., Markidis, S., Jordan, H. and Fahringer, T., 2018. A taxonomy of task-based parallel programming technologies for High Performance computing. The Journal of Supercomputing, 74(4), pp.1422-1434.

[11] Ross, R.B., Amvrosiadis, G., Carns, P., Cranor, C.D., Dorier, M., Harms, K., Ganger, G., Gibson, G., Gutierrez, S.K., Latham, R. and Robey, B., 2020. Mochi: Composing data services for High Performance computing environments. Journal of Computer Science and Technology, 35, pp.121-144.

[12] Hutchinson, M.S., 2020. Applying High Performance computing to early fusion video action recognition (Doctoral dissertation, Massachusetts Institute of Technology).

[13] Goar, D. V. . (2021). Biometric Image Analysis in Enhancing Security Based on Cloud IOT Module in Classification Using Deep Learning- Techniques. Research Journal of Computer Systems and Engineering, 2(1), 01:05. Retrieved from https://technicaljournals.org/RJCSE/index.php/journal/article/view/9

[14] Tosson, A., 2020. The way to a smarter community: exploring and exploiting data modeling, big data analytics, High Performance computing, and artificial intelligence techniques for applications of 2D energy-dispersive detectors in the crystallography community.

[15] Li, C. and Zhao, Y., 2019. Traffic route optimization based on cloud computing parallel ACS. International Journal of Information and Communication Technology, 14(2), pp.204-217.

[16] Kairi, A., Gagan, S., Bera, T. and Chakraborty, M., 2019. DNA Cryptography-Based Secured Weather Prediction Model in High Performance Computing.

In Proceedings of International Ethical Hacking Conference 2018: eHaCON 2018, Kolkata, India (pp. 103-114). Springer Singapore.

[17] Posey, B., Deer, A., Gorman, W., July, V., Kanhere, N., Speck, D., Wilson, B. and Apon, A., 2019, November. On-demand urgent High Performance computing utilizing the google cloud platform. In 2019 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC) (pp. 13-23). IEEE.

[18] Catak, F.O. and Balaban, M.E., 2013. CloudSVM: training an SVM classifier in cloud computing systems. In Pervasive Computing and the Networked World: Joint International Conference, ICPCA/SWS 2012, Istanbul, Turkey, November 28-30, 2012, Revised Selected Papers (pp. 57-68). Springer Berlin Heidelberg.

[19] Dogani, J., Khunjush, F., Mahmoudi, M.R. and Seydali, M., 2023. Multivariate workload and resource prediction in cloud computing using CNN and GRU by attention mechanism. The Journal of Supercomputing, 79(3), pp.3437-3470.

[20] Arif, M., Ajesh, F., Shamsudheen, S. and Shahzad, M., 2022. Secure and Energy-Efficient Computational Offloading Using LSTM in Mobile Edge Computing. Security And Communication Networks, 2022, pp.1-13.