Azerbaijan Journal of High Performance Computing, Vol 3, Issue 2, 2020, pp. 245-254
https://doi.org/10.32010/26166127.2020.3.2.245.254

AzJHPC
Azerbaijan Journal of High Performance Computing

Challenges of Using Big Data in Distributed Exascale Systems

Firuza Tahmazli-Khaligova

Department of General and Applied Mathematics, Azerbaijan State Oil and Industry University, Baku, Azerbaijan, ftahmazli@gmail.com

*Correspondence: Firuza Tahmazli-Khaligova, Department of General and Applied Mathematics, Azerbaijan State Oil and Industry University, Baku, Azerbaijan, ftahmazli@gmail.com

Abstract

In a traditional High Performance Computing system, it is possible to process a huge data volume. The nature of events in classic High Performance computing is static. However, distributed exascale system has a different nature. The processing big data in a distributed exascale system evokes a new challenge. The dynamic and interactive character of a distributed exascale system changes process's status and system elements. This paper discusses the challenge of the big data attributes: volume, velocity, variety; how they influence distributed exascale system dynamic and interactive nature. While investigating the effect of the dynamic and interactive nature of exascale systems in computing big data, this research suggests the Markov chains model. This model constructs the transition matrix, which identifies system status and memory sharing. It lets us analyze convergence of the two systems. As a result both systems are explored by the influence of each other.

Keyword: High Performance Computing, Distributed Exascale System, Dynamic and Interactive, Big Data.

1. Introduction

The traditional way to store data and process them is not sufficient for today's data-intensive computing. When big data appeared in web companies, the field of science met new data processing problems. The digital data are collecting, stored, and analyzed, but it is tough to create real-time data computing. In traditional HPC can cope with a huge amount of data efficiently. (Cheng, P., Lu, Y., Du, Y., & Chen, Z., 2018, March).

The new problem is to find the convergence of big data and distributed exascale system. Big data has a complex anatomy and dictates to HPC compatible with it. We aim to explore the problems on both sides: big data and HPC. This article's primary purpose is to find interconnected relations between distributed exascale systems and big data. In the present day, we are on the road to change the scale of HPC machines. For computing big data, the exascale (one supercomputer can compute at the exaflops scale) system is needed. (Pektürk, M. K., & Ünal, M. (2018, August), (Reed, D. A., & Dongarra, J., 2015).

To find the convergence between two systems, it should be explored nature of them. Big data's nature means its attributes: volume, velocity, and variety; for the distributed exascale system, it means dynamic and interactive environment (Khaneghah, E. M., & Sharifi, M., 2014).

We can look at some computing problems on HPC systems by considering all two systems' nature. When big data systems are running on the distributed exascale system, we notice appearing of the strange events. Resource attributes are dynamic. It influences processes status. The processes are getting slow down. Request to read and write data circuits on many nodes; new processes are forked (created after fork() operation), the parallel topologies of data processing end up system anomalies and latency. The other issue is that the system depends on resources: the resource can join and leave the system without control or it is unpredictable. Therefore, it is impossible to estimate how many data are incoming to the system next per second. The distributed system accept different type of data: structured, semi-structured, unstructured—variety of big data influence data processing in nodes. (Tulasi, B., Wagh, R. S., & Balaji, S. (2015)).

Furthermore, the diversity of data requires multiple approaches to compute them. The primary method is parallel processing on the HPC system. It is difficult to estimate which nodes are busy with different types of data. The other attributes of big data are the volume. All features of big data influence system processing. The asynchronized communication of nodes causes all system or some nodes failures. The obscure nature of these problems begins to create dynamic behavior in processes at runtime. Our purpose is to define the role of big data attributes in the uncertain dynamic system environment.

The processing to investigate mentioned problems, we faced questions as below:

- How volume, velocity, and variety influence a new system fork()?c
- How volume, velocity, and variety influence process intercommunication?
- How volume, velocity, and variety influence interactive processes?
- How can volume, velocity, and variety influence memory (where data come in and waiting for computing)?

We can summarize all issues by mentioning that, there is a problem with processing big data in a distributed exascale system. In real-time computing on HPC system, we are facing with the dynamicity of resources and processes. We propose the Markov chain model as a solution to find the relationship between big data attributes and HPC system. Markov chain model is based on creating transition matrixes for executing process statuses and memory sharing.

2. Big Data Anatomy and Dynamic & Interactive Nature

Big data are a kind of data that contains some dynamic features in itself.

The difference between traditional data and big data is that, big data have some features that are called attributes. They are volume, velocity, variety, shortly denoted as 3Vs. Besides, the researchers investigate other characters (attributes) for big data: veracity and value. However, volume, velocity, and variety influence the executing of big data in the distributed exascale computing system. (Xuan, P., Denton, J., Srimani, P. K., Ge, R., & Luo, F. (2015, November))

Volume: It is the volume of the dataset. The social media, the bank transactions, sensor embedded systems, the research laboratories generate and consume a huge amount of data in a second. The batch of raw data is stored at a cloud computing system containing many petabytes of data. The scientific laboratories routinely process terabytes of data produced by their instruments. (Kumar, M.P., Kumar, S.S., Ramya, I. G. (2018))

Velocity: The velocity can be considered as the speed at which data are generated and transmitted. The data are now continuously streaming into the servers in real-time and make the original batch process break down in distributed system machine to machine process exchange data billions of devices.

Variety: Big data are not just strings. They are also geospatial data, signals, 3D data, audio, video, image, structured and unstructured text. The execution in distributed exascale computing system the variety of big data causes a problem. (Xie, J., Song, Z., Li, Y., et al. (2018)),

Scientific researchers need a distributed exascale system to compute a tremendous amount of data. There is no alternative computing system still now. We have a huge volume of data with complex architecture and dynamic distributed systems. (Jackson, A., Weiland, M., Parsons, M., & Homoelle, B. (2019, June)) Big data is a kind of data that contains some dynamic features in itself. The different traditional data from big data have some features that are called attributes. Besides, the researchers investigate other characters (attributes) for big data: veracity and value.

Nevertheless, volume, velocity, and variety influence the execution of big data in the distributed exascale computing system. Big data's runtime elements are events (sources: remote sensors, people generating data, measurement and transaction systems), data-intensive system, shared cloud infrastructure, and users. (Xuan, P., Denton, J., Srimani, P. K., Ge, R., & Luo, F. (2015, November)). The traditional high performance computing system cope with the huge volume of data. The present-day real-time big data computing in distributed exascale system cause some problems. (Khaneghah, E. M., & Sharifi, M. (2014))

Dynamic distributed exascale system is changing periodically. Some nodes are joining the system or leaving, or they are out of order. When the computing processes are executing, some new processes start to run unpredictably. The nodes of distributed system communicate other nodes of system non-estimated in advance.

In the dynamic, uncertain runtime environments in which big data systems exist, some specific challenges have occurred.

3. Related Work.

Big data and HPC help us manage resources and get high performance to identify the relation between two systems. The resources in the new generation distributed exascale system is dynamic. Our mission is to find extra external impacts on these systems and manage these systems effectively. We assume that big data nature will be able to change executing workloads in a distributed exascale system. It is essential to know what interrupts or causes the latency execution of the HPC system processes.

For this reason, we proposed a stochastic approach based on a Markov model to this problem. In both systems, all processes are complex and mixed. The prediction method will be useful to find some patterns of two systems interconnections. Our model is based on states of processes execution and big data attributes. This model gives us the situations of both systems. The transition matrix describes the relation of big data features and HPC.

Our approach's aim is to increase all attributes of big data separately and create a transition matrix for prediction processes' states that are executing each node of the distributed system. After the implementation of our model, we will get some results:

-if we are increasing any big data attributes, the Markov model creates a transition matrix M_{states} (each element of this matrix contains the probability of states, which each simple process has). M transition matrix is changed after each level of increasing attributes value. Each matrix element describes the probability of moving state positions (states: new, ready, running, waiting, terminate). The variation of each element of the matrix shows the condition of the dynamic system. We can watch the predicting of the process's state. Also, we can manage resource allocation and sharing.

- if we are increasing the attribute's value of big data, the Markov model creates a transition matrix M_{memory} (each element of this matrix contains the probability of write and read states of memory parts). This matrix shows us the status of the memory part after increasing the attributes of big data. We can identify which attributes cause bottleneck in memory or which attributes influence the fault of I/O processing and.

The disadvantage of our model is complexity of implementation. We need to simulate the increasing big data attributes separately in the distributed system and choose the correct data file system and resource managers for the implementation. (Fox, G. C., Qiu, J., Kamburugamuve, S., Jha, S., & Luckow, A., 2015, May). For getting results, we need to implement our model in the different systems: big data processing (Hadoop, Lustre, etc.) and HPC system (Open MPI). (Kim, Y., Park, S., Wang, F., Sun, G., & Wang, S., 2018).

There are different approaches for giving solutions to this problem. One of them is BeBida model (Best-Effort Big Data). They propose a new idea solely based on RJMS (Resource and Job management system) configuration. It makes HPC and big data systems communicate through a simple prolog/epilog mechanism, which leverages big data frameworks' built-in resilience while minimizing the disturbance on the HPC workloads. The BeBida model did not take into account the big data nature influence.

Our Markov chain model is based on the nature of both system and anatomy of them. Bebida model is based on how to manage dynamicity pool resources and gives a solution for it. This solution exposes all nodes not used by HPC jobs to the Big Data resource manager as a dynamic pool of resources. (Mercier, M., Glesser, D., Georgiou, Y., & Richard, O., 2017, December). In our model, we want to ensure that what causes the dynamicity in computing big data in HPC.

The Bebida model will perform well when big data is the most influential in the workload, i.e., when the HPC utilization is low. On the opposite, they expect to have low big data performance on a loaded HPC cluster because the number of idle resources will increase, inducing more re-computation and data movement. (Mercier, M., Glesser, D., Georgiou, Y., & Richard, O., 2017, December).

Both models' purposes are to find a solution for big data computing in HPC systems. Nevertheless, they have a different approach to this problem. The Bedida model suggests managing idle nodes in a dynamicity pool of resources, and our model suggests identifying how big data attributes cause the dynamicity of HPC nature. The information about causing the dynamicity helps us to manage the HPC resource system efficiently.

Markov chains model in the distributed exascale system.

When executing big data in the distributed exascale system, the execution process faced the system's dynamic and interactive nature. Our purpose is to find role of the big data in the distributed exascale system. If we want to interconnect big data and HPC systems, it is important to analyze the two systems' features and internal characters. How big data attributes influence the system's nature. We want to identify if we start to increase the volume, velocity, variety of big data what will system react:

- How volume, velocity, and variety influence a new system fork()?
- How volume, velocity, and variety influence process intercommunication?
- How volume, velocity, and variety influence interactive processes?
- How can volume, velocity, and variety influence memory (where data come in and waiting for computing)?

We want to identify if we start to increase the volume, velocity, variety of big data what will system react:

- create a new process or not,
- the processes start to interact with other process or not
- the processes intercommunication or not.

In the HPC system, many processes are executing simultaneously by parallel programming. We use the Markov chain to sort out this problem and give a model to track all events in a distributed exascale system. Firstly, we take one attribute of big data and increase it. In this situation, we suppose other attributes are stable, or their variations do not impact system’s status (do not cause dynamic and interactive nature). The next step starts to create a transition matrix of process states. All states of the process have a figure, which describes probability of occurrence of process states. We start by taking five states of the process: new, ready, running, waiting, and termination. Each transition probability of states is described in Figure 1 below:

Table 1. Transition matrix of executing process status

The states of processes	New	Ready	Running	Waiting	Termination
New	P(1,1)	P(1,2)	P(1,3)	P(1,4)	P(1,5)
Ready	P(2,1)	P(2,2)	P(2,3)	P(2,4)	P(2,5)
Running	P(3,1)	P(3,2)	P(3,3)	P(3,4)	P(3,5)
Waiting	P(4,1)	P(4,2)	P(4,3)	P(4,4)	P(4,5)
Termination	P(5,1)	P(5,2)	P(5,3)	P(5,4)	P(5,5)

The number i shows the initial state of the process; the number j indicates what state of the process is going to move. The P(i,j) is the probability of transition from state i to j state. The statuses of the process are described in Figure 1

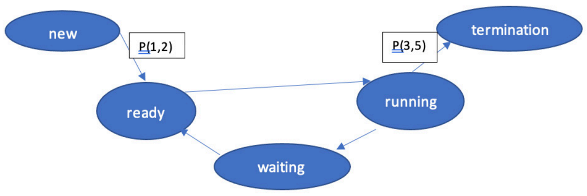

Fig. 1. Statuses of process

Each P(i,j) is calculated by formula (1):

P(i,j)= z/Z , {Z- is how many process states move to i state, z- is how many i states maintain i state) (1)

Our aim is to compare the transition matrix at each one level. We assign level with k variable. Initial level k=0. Then we start to increase the volume of big data. The velocity and variety are still stable, or growth of them is very little. We noted system status for k=0 and next calculated k=1 level status. Maybe after increasing the volume of big data, the transition matrix will change. We can describe them as below :

Mvolume{P(i,j) 1<=i,j<=5;} k=0,1,2,3…; (2)

Mvelocity{P(i,j) 1<=i,j<=5;} k=0,1,2,3…; (3)

Mvariety{P(i,j) 1<=i,j<=5;} k=0,1,2,3…; (4)

kattribute increase ; (<>- the comparison sign) (5)

Mvolume{Pk=0(i,j)}<>Mvolume{Pk=1(i,j)} <>Mvolume{Pk=2(i,j)}<>…. (6)

Mvelocity{Pk=0(i,j)}<>Mvelocity{Pk=1(i,j)} <>Mvelocity{Pk=2(i,j)}<>…(7)

Mvariety{Pk=0(i,j)}<>Mvariety{Pk=1(i,j)} <>Mvariety{Pk=2(i,j)}<>….(8)

In formula (6) if for k=0 and k=1 level the Mk=0 and Mk=1 transition matrix are different then it means the volume of big data cause the system status. The formula (7) and (8) also show comparison of transition matrixes of velocity and variety respectively. We can also compare each Pk=0(i,j) and Pk=1(i,j) then to get information which state probability was increased or decreased. If P(1,1) is changed it means new processes are created. Because figure of "new" state of process are changed only after new fork() operation. We can monitor all element of transition matrix. The same idea also is related to other attributes of Big data. It is possible to display in Figure 2:

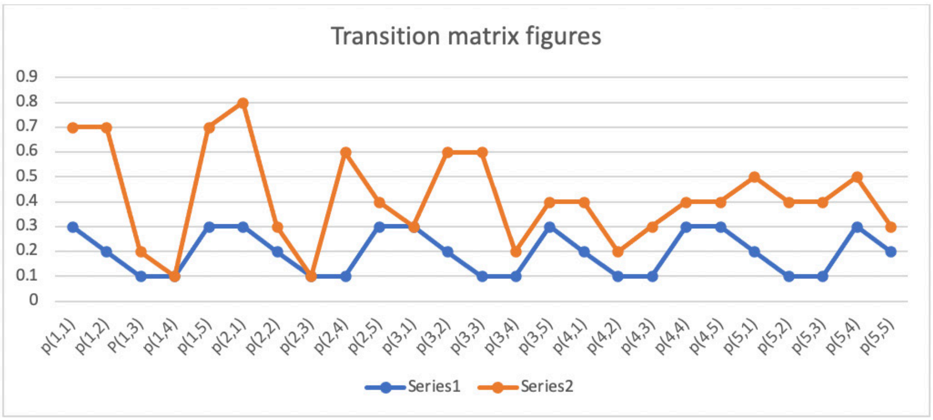

Fig. 2. Elements values of transition matrix by different increasing levels (k=0,1)

In Figure 3, the x axis describes the transition of i status to j; the y axis shows the probability value of this status transition. All P(i,j) are altered between 0 and 1. We have chart lines that show increasing of each level (k=0,1..). We should create the charts for all attributes of Big data. After analysis of all chart figures, we will be able to obtain the information about the system.

For other attributes of big data may cause dynamic and interactive events. Due to the next step, we increase the variety of big data and observe the distributed exascale system processes. We keep the volume and velocity stable, or we suppose that their growth is very little. It is considered that they can not change the dynamic and interactive events. The same calculation process is keeping on. In the end, as a result, we obtain the new transition matrix for variety and velocity.

$$\begin{matrix} M_{volume}\ \{P_k=0(i,j)\} \\ M_{velocity}\ \{P_k=0(i,j)\} \\ M_{variety}\ \{P_k=0(i,j)\} \end{matrix} \quad \rightarrow \quad \text{System <CPU, I/O, Memory>} \quad (9)$$

4. Markov Model for Big Data Storage Problem in the Distributed Exascale System

We know some processes in the operating system write data to memory and other processes take data from this memory. It is called the producer-consumer processes. If we try to represent these events as graph we get an image in Figure 3:

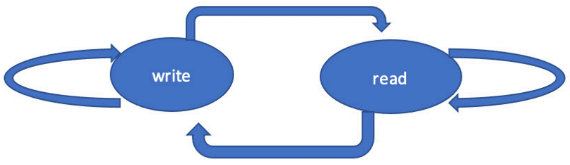

Fig. 3. Producer-consumer process graph

This conception contains 2 nodes - write and read. 4 directed edges show how data comes to memory and goes to processing. We aim to find how big data attributes impact runtime processes in the distributed exascale system in the storage data. We suggest the Markov chain model identify the impact of big data attributes on memory. For that, we create a transition matrix. First, we suppose the volume is being increased, but velocity and variety are stable. In initial level k=0. After increasing volume k=1. It is described in table 2.

Table 2. Transition matrix of producer-consumer process

states	write	read
write	P(1,1)	P(1,2)
read	P(2,1)	P(2,2)

We get the transition matrix for k=0. P(i,j) is the probability of write and read nodes. Each element variation of the matrix shows the process directions. For example, after increasing the volume of big data, if P(1,1) has grown, it means the volume cause I/O bottleneck. P(1,1) means some processes are fetching data to memory, but consumer processes are late to get data from memory. Obviously said that the writing processes are propagating themselves.

Nevertheless, consumer processes are not able to get data from the memory part. For other attributes of big data, the same transition matrix should be created and analyzed. After investigating all transition matrices and P(i,j) elements and elements, we will obtain significant results about internal processes in a distributed exascale system. As a result end of the implementation of this model, we will acquire transition matrixes:

$M_{volume}\{P(i,j);\ i,j=1,2;\ k=0,1,2,3\ldots\}$ M_{volume}-is transition matrix for volume (10)

$M_{velocity}\{P(i,j);\ i,j=1,2;\ k=0,1,2,3\ldots\}$ $M_{velocity}$-is transition matrix for velocity (11)

$M_{variety}\{P(i,j);\ i,j=1,2;\ k=0,1,2,3\ldots\}$ $M_{variety}$-is transition matrix for variety (12)

We will compare results for getting outcomes. By same attributes, the comparison is called Csame(<> is comparison sign)

$M_{volume}\,(k=0) <> M_{volume}\,(k=1) <> M_{volume}(k=2) <> \ldots..$

$M_{velocity}$ (k=0) <>$M_{velocity}$ (k=1)<> $M_{velocity}$ (k=2)<>….. C_{same} (13)
$M_{variety}$ (k=0) <>$M_{variety}$ (k=1)<> $M_{variety}$(k=2)<>…..

Of course, we can compare matrixes by different attributes but for the same level (for k=0,1,2..). It is called C_{dif}.

M_{volume} (k=0) <> $M_{velocity}$ (k=0) <> $M_{variety}$(k=0)
$M_{velocity}$ (k=1) <> $M_{velocity}$ (k=1) <> $M_{velocity}$ (k=1) Cdif (14)
M_{volume}(k=2) <> $M_{velocity}$ (k=2) <> $M_{variety}$(k=2)

C_{same} and C_{dif} comparisons give us a distinctive viewpoint for analyzing of big data attributes to the memory part of resource sharing in HPC. Csame means increasing of which attribute or attributes impact memory sharing. Cdif means at the same level what attributes or attributes impact on memory sharing.

5. Discussion

In the conclusion section, we noted the Markov chain model results for computing big data in a distributed exascale system. A and B results are desirable. Both results give us a new approach to manage the dynamic nature of the system. It helps us raise the performance of CPU or memory sharing. The C result is undesirable. It creates chaos that we cannot identify how big data influences the system's dynamic and interactive nature. Nevertheless, we suppose that the C variant has not a high possibility because we know the dynamic nature of system-independent data anatomy and nature.

6. Conclusion

While executing big data in the distributed exascale system is the dynamic nature of the system, we suggest the Markov chain model. This model is based on creating a transition matrix for process status and memory sharing. All transition matrixes are compared to obtain results, for comparing them we create charts. The figures for each level compared with the previous levels. At a result, we will be able to observe different situations:

A. the figures of each level will be not various or the bias not in significant degree than the previous level

B. the figures of each level will be various in a significant degree than the previous level, and at each level, the figures are steady increasing or decreasing.

C. the figures of each level will acquire mixed variation. The variation trace will be up and low.

Independently from the results of the implementation, we will able to adjust system configurations. As future work we want to apply our model to a different platform for computing big data in High Performance Computing.

References

Cheng, P., Lu, Y., Du, Y., & Chen, Z. (2018, March). Experiences of converging big data analytics frameworks with high performance computing systems. In *Asian Confer-*

ence on Supercomputing Frontiers (pp. 90-106). Springer, Cham.

Fox, G. C., Qiu, J., Kamburugamuve, S., Jha, S., & Luckow, A. (2015, May). Hpc-abds high performance computing enhanced apache big data stack. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (pp. 1057-1066). IEEE.

Jackson, A., Weiland, M., Parsons, M., & Homoelle, B. (2019, June). An architecture for high performance computing and data systems using byte-addressable persistent memory. In *International Conference on High Performance Computing* (pp. 258-274). Springer, Cham.

Khaneghah, E. M., & Sharifi, M. (2014). AMRC: an algebraic model for reconfiguration of high performance cluster computing systems at runtime. *The Journal of Supercomputing, 67*(1), 1-30.

Kim, Y., Park, S., Wang, F., Sun, G., & Wang, S. (2018). HPC software and programming environments for big data applications. *Scientific Programming,* 1-2.

Kumar, M.P., Kumar, S.S., Ramya, I. G. (2018) Big data analytics: A brief survey. *International Journal of Trend in Scientific Research and Development (ijtsrd), 2*(4), 2264-2268.

Mercier, M., Glesser, D., Georgiou, Y., & Richard, O. (2017, December). Big data and HPC collocation: Using HPC idle resources for big data analytics. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 347-352). IEEE.

Pektürk, M. K., & Ünal, M. (2018, August). Performance-aware high-performance computing for remote sensing big data analytics. In *Data Mining* (p. 69). BoD–Books on Demand.

Reed, D. A., & Dongarra, J. (2015). Exascale computing and big data. *Communications of the ACM, 58*(7), 56-68.

Tulasi, B., Wagh, R. S., & Balaji, S. (2015). High performance computing and big data analytics - paradigms and challenges. *International Journal of Computer Applications, 116*(2), 28-33.

Xie, J., Song, Z., Li, Y., et al. (2018). A survey on machine learning-based mobile big data analysis: Challenges and applications. *Wireless Communications and Mobile Computing,* 1-19.

Xuan, P., Denton, J., Srimani, P. K., Ge, R., & Luo, F. (2015, November). Big data analytics on traditional HPC infrastructure using two-level storage. In *Proceedings of the 2015 International Workshop on Data-Intensive Scalable Computing Systems* (pp. 1-8).

Xuan, P., Denton, J., Srimani, P. K., Ge, R., & Luo, F. (2015, November). Big data analytics on traditional HPC infrastructure using two-level storage. In *Proceedings of the 2015 International Workshop on Data-Intensive Scalable Computing Systems* (pp. 1-8).

Submitted: 17.07.2020
Accepted: 27.11.2020