ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 4, No. 1, 40-44 (2019)

www.astesj.com

Special Issue on Recent Advances in Engineering Systems

ASTESJ

ISSN: 2415-6698

A Proposed Architecture for Parallel HPC-based Resource Management System for Big Data Applications

Waleed Al Shehri*, Maher Khemakhem, Abdullah Basuhail, Fathy E. Eassa

Department of Computer Science, King Abdul-Aziz University, Jeddah, KSA

ARTICLE INFO

Article history:
Received: 02 October, 2018
Accepted: 11 January, 2019
Online : 20 January, 2019

Keywords:
High-Performance Computing
Big Data
Load-Balancing
Data-Locality
Resource Management
Parallel Programming models
Power Consumption

ABSTRACT

Big data can be considered to be at the forefront of the present and future research activities. The volume of data needing to be processed is growing dramatically in both velocity and variety. In response, many big data technologies have emerged to tackle the challenges of collecting, processing and storing such large-scale datasets. High-performance computing (HPC) is a technology that is used to perform computations as fast as possible. This is achieved by integrating heterogeneous hardware and crafting software and algorithms to exploit the parallelism provided by HPC. The performance capabilities afforded by HPC have made it an attractive environment for supporting scientific workflows and big data computing. This has led to a convergence of the HPC and big data fields.

However, big data applications usually do not fully exploit the performance available in HPC clusters. This is so due to such applications being written in high-level programming languages and do not provide support for exploiting parallelism as do other parallel programming models.

The objective of this research paper is to enhance the performance of big data applications on HPC clusters without sacrificing the power consumption of HPC. This can be achieved by building a parallel HPC-based Resource Management System to exploit the capabilities of HPC resources efficiently.

1. Introduction

The amount of data produced in the scientific and commercial fields is growing dramatically. Correspondingly, big data technologies, such as Hadoop and Spark, have emerged to tackle the challenges of collecting, processing, and storing such large-scale data.

There are different opinions on the definition of big data resulting from different concerns and technologies. One definition applies to datasets that cannot be realized, managed and analyzed with traditional IT software. This definition reflects two connotations: data volume that is growing and changing continuously; and, this growing volume is different from one big data application to another [1]. A more specific definition based on the multi-V model by Gartner in 2012: ‘‘Big Data are high-volume, high-velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization’’ [2].

While the focus of big data applications is on handling enormous datasets, high-performance computing (HPC) focuses on performing computations as fast as possible. This is achieved by integrating heterogeneous hardware and crafting software and algorithms to exploit the parallelism provided by HPC [3]. The performance capabilities afforded by HPC have made it an attractive environment for supporting scientific workflows and big data computing. This has led to a convergence of the HPC and big data fields.

Unfortunately, there is usually a performance issue when running big data applications on HPC clusters because such applications are written in high-level programming languages. Such languages may be lacking in terms of performance and may not encourage or support writing highly parallel programs in contrast to some parallel programming models like Message Passing Interface (MPI) [4]. Furthermore, these platforms are designed as a distributed architecture, which differs from the architecture of HPC clusters [5].

* Waleed Al Shehri, : Email: waleed.ab2@gmail.com

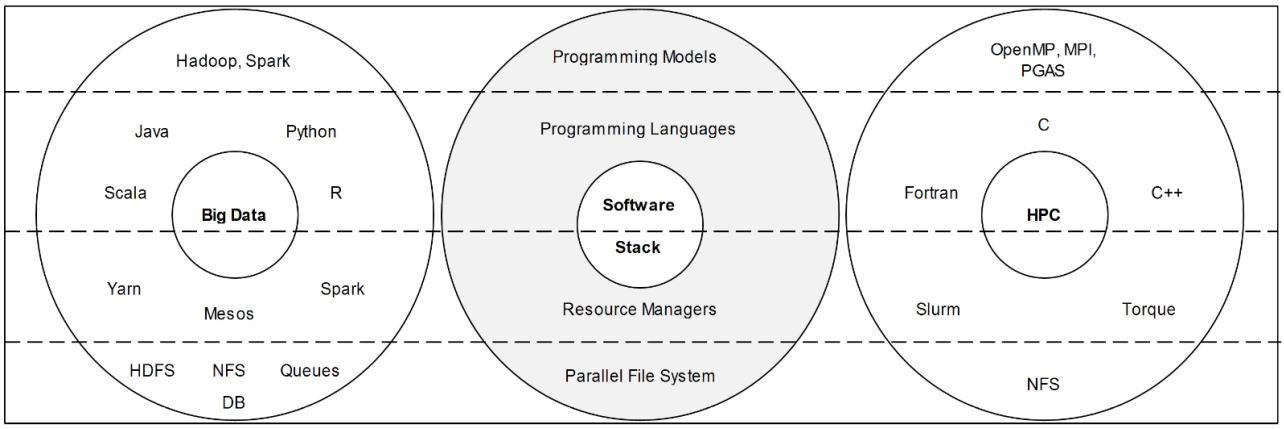

Figure 1. HPC and Big Data Software Stacks.

Additionally, the large volume of big data may hinder parallel programming models such as Message Passing Interface (MPI), Open Multi-Processing (OpenMP) and accelerator models (CUDA, OpenACC, OpenCL) from supporting high levels of parallelism [1].

Furthermore, resources allocation in HPC is one of the prime challenges, especially since HPC and big data paradigms has a different software stack [6] as shown in (**Figure 2**):

2. Related Work

The related work can be organized based on the different aspects required to fulfill the architecture requirements of this research. The job scheduler concept will be highlighted by considering its features and functionality. Moreover, different comparative studies will be introduced covering big data programming models and parallel programming models to establish the performance gap between them. Other works are presented to show how the performance of these programming models can be enhanced. Finally, research involving data locality approaches and decomposition mechanisms covering the same context of this research is reviewed.

A job scheduler can play an essential role in modern big data platforms and HPC systems. It manages different compute jobs related to different users on homogenous or heterogeneous computational resources. It can have different names that reflect the same mechanism such as scheduler, resource manager, resource management system (RMS), and a distributed resource management system (D-RMS) [7]. Despite significant growth in terms of heterogeneity of resources and job complexity and diversity, job schedulers still have the main core function of job queuing, scheduling and resource allocation, and resource management [8][9]. In [7], many features are analyzed of the most popular HPC and big data schedulers including Slurm, Son of Grid Engine, Mesos, and Hadoop YARN.

Additionally, there are two primary job types: job arrays and parallel jobs. In job arrays, multiple independent processes for a single job identifier can be run with different parameters for each process. In parallel jobs, it is possible to launch each of the processes simultaneously, allowing communication between them during the computation. While HPC schedulers support both types, big data schedulers can support only job arrays. Furthermore, there are many important features of HPC schedulers, generally not available with big data schedulers, such as job chunking, gang scheduling, network aware scheduling and power-aware scheduling.

Big data jobs are usually considered to be network-bound regarding a large amount of data movement between different nodes among clusters. In [10], traffic forecasting and job-aware priority scheduling for big data processing is proposed by considering the dependencies of the flows. The network traffic for flows of the same job is forecasted via run-time monitoring, then a unique priority for each job is assigned by tagging every packet in the job. Finally, it uses a FIFO order for scheduling flows of the same priority.

In [11], a new backfilling algorithm, known as fattened backfilling, is proposed to provide more efficient backfilling scheduling. In this algorithm, short jobs can be moved forward if they do not delay the first job in the queue. A Resource and Job Management System (RJMS) based on a prolog/epilog mechanism has been proposed in [12]. It allows communication between HPC and Big Data systems by reducing the disturbance on HPC workloads while leveraging the built-in resilience of Big Data frameworks.

Processing tremendous volumes of data on dedicated big data technology is not as fast as processing the data on HPC infrastructure. This fact is recognized when comparing the efficiency of low-level programming models in HPC, which supports more parallelism, with big data technologies that are written with high-level programming languages. Many practical case studies and research have confirmed this fact. In [13], sentiment analysis on Twitter data was conducted for different dataset sizes using an MPI environment that showed better performance than using Apache Spark.

The enhancement of big data programming models can be achieved by integrating them with parallel programming models such as MPI. This approach can be seen in [4] that showed how to enable the Spark environment using the MPI libraries. Although this technique indicates remarkable speedups, it must use shared memory, and there are other overheads as a potential drawback. In [14], a scalable MapReduce framework, named Glasswing, is introduced. It is configured to use a mixture of coarse- and fine-grained parallelism to obtain high performance on multi-core

CPUs and GPUs. The performance of this framework is evaluated using five MapReduce applications with the indication that Glasswing outperforms Hadoop in terms of performance and resource utilization.

Data locality is a critical factor that affects both performance and energy consumption in HPC systems [15]. Many big data frameworks such as MapReduce and Spark can support this concept by sending the computation to where the data resides. In contrast, parallel programming models such as MPI lack this advantage. A novel approach by Yin et al. in [16], named DL-MPI, is proposed for MPI-based data-intensive applications to support data locality computation. It uses a data locality API that allows MPI-based programs to obtain data distribution information for compute nodes. Moreover, it proposes a probability scheduling algorithm for heterogeneous runtime environments that evaluate the unprocessed local data and the computing ability of each compute node.

In [17], a data distribution scheme is used by abstracting NUMA hardware peculiarities away from the programmer and delegating data distribution to a runtime system. Moreover, it uses task data dependence information, which is available with OpenMP 4.0RC2, as a guideline for scheduling OpenMP tasks to reduce data stall times.

Partitioning or decomposition is the first step for designing a parallel program by breaking down problems into small tasks. It includes two main types: domain or data decomposition and function decomposition [18]. These two types can be combined as mixed parallelism that employs an M-SPMD (multiple-single program multiple data) architecture, which includes both task parallelism (MPMD) and data parallelism (SPMD) [19].

The choice of decomposition type and parallelism paradigm is determined by resource availability. Furthermore, these resources may define the granularity level that the system can support [20]. There have been few empirical studies for performing data decomposition in the HPC field, [21] investigated this approach when designing parallel applications. Additionally, the state-of-practice was studied with probing tools used to perform this function. Moreover, a set of key requirements was derived for tools that support data decomposition and communication when parallelizing applications.

Based on this previous related work and to the best of our knowledge, there is no contribution yet that employs all the previous factors in terms of using a hybrid parallel programming model, decomposition technique and granularity approach to build a HPC-based Resource Management System for enhancing the performance of big data applications and optimizing HPC resource utilization. The contribution of this paper is to address this gap.

3. The Proposed Architecture

The proposed architecture will be built based on different techniques that will be integrated together to constitute a parallel HPC-based Resource Management System that enhances the performance of big data applications on HPC clusters without sacrificing the power consumption of HPC. In more details, the system will have the following techniques (Figure 2):

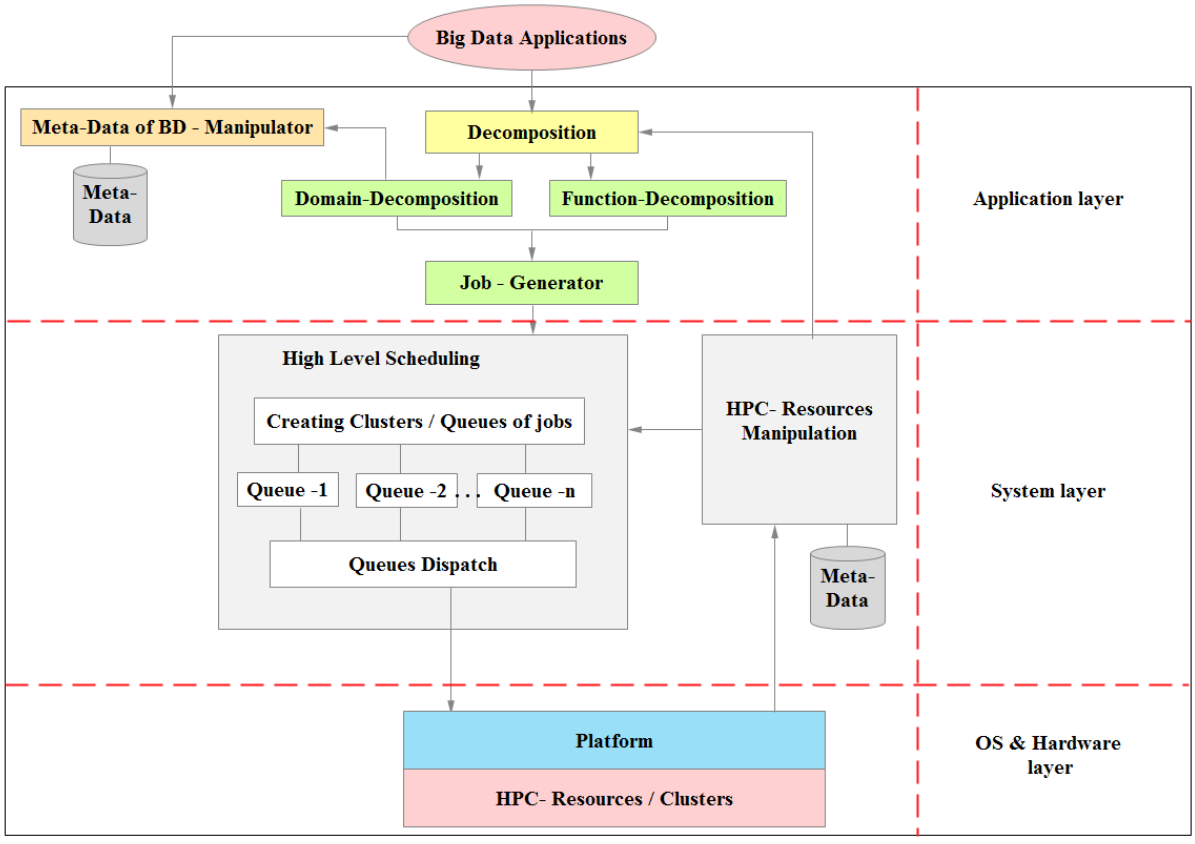

Figure 2: High-level architecture for the proposed system.

- **A new technique for HPC resources manipulation**

This dynamic technique will have some functionality related to HPC resources including:

  - Collecting a resource metadata to constitute a repository containing HPC resources with their capabilities and availabilities.
  - Tracking status of HPC resources.
  - Updating the metadata repository from time to time.

- **New decomposition techniques**

In this technique, we will provide a domain decomposition technique and/or a function decomposition technique for the given big data applications targeting both parallel computing architectures: Single Program Multiple Data (SPMD) and/or Multiple Instruction Multiple Data (MIMD). The availability of resources and system architecture can determine the decomposition paradigm and granularity options that efficiently support the resource management system.

- **A new high-level scheduling technique**

This technique will receive the decomposition results and creates clusters/queues of jobs during scheduling time based on the metadata about HPC resources/ clusters. It will consider data locality and load balancing while performing this task. Once clusters/queues of jobs are created, they will be dispatched to being executed by OS-platform on HPC resources/clusters.

4. Evaluation and comparative study

By comparing our architecture to other techniques, it is noticeable that the proposed architecture considers all the critical factors to achieve the performance and scalability attributes. Primarily, focusing on the topology awareness and building a metadata repository about the availabilities and capabilities of HPC resources can play a critical role to support the decomposition and high-level scheduling techniques. Such metadata can enhance the decision-making about choosing suitable granularity options and parallelism paradigm. Furthermore, the high-level scheduling technique can exploit the HPC resources positively by taking in account data locality and load balancing.

Instead of Integrating some big data and parallel programming models, this architecture constitutes an independent big data platform that employ hybrid parallel programming to support high parallelism for CPUs and GPUs accelerators.

The scalability can be seen from adding more dedicated clusters as needed. Adding more clusters will not affect the essence of each technique in particular, and resource management as a whole system integrating these techniques.

Different performance metrics have to be considered to implement the proposed architecture efficiently. Big data is the primary stream of this architecture, thus data building time is a significant metrics that may affect the performance. This time is required to construct a data structure used for computation and to perform the decomposition technique. Furthermore, employing parallel programming models such as MPI and OpenAcc can affect the computation time positively. From the part of HPC, hardware utilization metrics is also a cornerstone of the proposed architecture particularly for improving high-level scheduling technique

The novelty of this architecture can be arisen from having metadata about both big data applications and HPC resources, which leads to scheduling current jobs to the most suitable and available resources or cluster.

5. Conclusion

HPC has become an attractive environment for supporting scientific workflows and big data computing due to its performance capabilities. Unfortunately, big data applications usually do not fully exploit these capabilities afforded by HPC clusters, because such applications were written in high-level programming languages that do not encourage parallelism as parallel programming models. Another reason is that the architecture of big data platforms defers from the HPC architecture. A parallel HPC-based Resource Management System is proposed in this paper to enhance the performance of big data applications on HPC clusters without sacrificing the power consumption of HPC. For the future work, the High-level architecture for the proposed system will be developed and evaluated by running some big data applications. Moreover, some performance benchmarks will be provided to reflect the efficiency of our system.

References

[1] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mob. Networks Appl.*, vol. 19, no. 2, pp. 171–209, 2014.

[2] C. L. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Inf. Sci. (Ny).*, vol. 275, pp. 314–347, 2014.

[3] D. A. Reed and J. Dongarra, "Exascale computing and big data," *Commun. ACM*, vol. 58, no. 7, pp. 56–68, 2015.

[4] M. Anderson *et al.*, "Bridging the gap between HPC and big data frameworks," *Proc. VLDB Endow.*, vol. 10, no. 8, pp. 901–912, 2017.

[5] P. Xuan, J. Denton, P. K. Srimani, R. Ge, and F. Luo, "Big data analytics on traditional HPC infrastructure using two-level storage," *Proc. 2015 Int. Work. Data-Intensive Scalable Comput. Syst. - DISCS '15*, pp. 1–8, 2015.

[6] H. R. Asaadi, D. Khaldi, and B. Chapman, "A comparative survey of the HPC and big data paradigms: Analysis and experiments," *Proc. - IEEE Int. Conf. Clust. Comput. ICCC*, pp. 423–432, 2016.

[7] A. Reuther *et al.*, "Scalable system scheduling for HPC and big data," *J. Parallel Distrib. Comput.*, vol. 111, pp. 76–92, 2018.

[8] P. Jones, "NAS for Job Requirements Queuing / Scheduling Checklist Software," 2018.

[9] W. Saphir, L. A. Tanner, B. Traversat, W.~Saphier, L.A.~Tanner, and B.~Traversat, "Job Management Requirements for {NAS} Parallel Systems and Clusters," *IPPS Work. Job Sched. Strateg. Parallel Process.*, no. 949, pp. 319–336, 1995.

[10] Z. Wang and Y. Shen, "Job-Aware Scheduling for Big Data Processing," *Proc. - 2015 Int. Conf. Cloud Comput. Big Data, CCBD 2015*, pp. 177–180, 2016.

[11] C. Gómez-Martín, M. A. Vega-Rodríguez, and J. L. González-Sánchez, "Fattened backfilling: An improved strategy for job scheduling in parallel systems," *J. Parallel Distrib. Comput.*, vol. 97, pp. 69–77, 2016.

[12] M. Mercier, D. Glesser, Y. Georgiou, and O. Richard, "Big Data and HPC collocation : Using HPC idle resources for Big Data Analytics," pp. 347–352, 2017.

[13] D. S. Kumar and M. A. Rahman, "Performance evaluation of Apache Spark Vs MPI: A practical case study on twitter sentiment analysis," *J. Comput. Sci.*, vol. 13, no. 12, pp. 781–794, 2017.

[14] I. El-Helw, R. Hofman, and H. E. Bal, "Glasswing," *Int. Symp. High-Performance Parallel Distrib. Comput.*, pp. 295–298, 2014.

[15] D. Unat *et al.*, "Trends in Data Locality Abstractions for HPC Systems," *IEEE Trans. Parallel Distrib. Syst.*, pp. 1–1, 2017.

[16] J. Yin, A. Foran, and J. Wang, "DL-MPI: Enabling data locality computation

for MPI-based data-intensive applications," *Proc. - 2013 IEEE Int. Conf. Big Data, Big Data 2013*, pp. 506–511, 2013.

[17] A. Muddukrishna, P. A. Jonsson, and M. Brorsson, "Locality-aware task scheduling and data distribution for OpenMP programs on NUMA systems and manycore processors," *Sci. Program.*, vol. 2015, pp. 156–170, 2015.

[18] B. Ren, S. Krishnamoorthy, K. Agrawal, and M. Kulkarni, "Exploiting Vector and Multicore Parallelism for Recursive, Data- and Task-Parallel Programs," *Proc. 22nd ACM SIGPLAN Symp. Princ. Pract. Parallel Program. - PPoPP '17*, pp. 117–130, 2017.

[19] V. Boudet, F. Desprez, and F. Suter, "One-step algorithm for mixed data and task parallel scheduling without data replication," *Proc. - Int. Parallel Distrib. Process. Symp. IPDPS 2003*, no. October 2015, 2003.

[20] L. Silva and R. Buyya, "Parallel programming models and paradigms," *High Perform. Clust. Comput. Archit. ...*, pp. 4–27, 1999.

[21] A. Meade, D. K. Deeptimahanti, J. Buckley, and J. J. Collins, "An empirical study of data decomposition for software parallelization," *J. Syst. Softw.*, vol. 125, pp. 401–416, Mar. 2017.