Scaleout to Scaleup: From Traditional Supercomputer to NextGen Bigcomputer for Data-intensive Scientific Applications

Arghya Kusum Das, Seung-Jong Park
School of Electrical Engineering and Computer Science
Center for Computation and Technology
Louisisna State University
Baton Rouge, LA, 70801
Email: {adas7, sjpark} @lsu.edu

***Abstract*—**

The enormous growth of bigdata produced by different experimental facilities is rapidly changing the model of computation in the domain of high performance computing (HPC). Many HPC aficionados, in order to efficiently manage their data intensive workload started using the current state of the bigdata analytics softwares like Hdoop, giraph etc. devieting from traditional parallel programming models like MPI etc. Consequently, traditional supercomputers with lots of processing power are found to provide suboptimal performance due to several limitations in the underlying hardware infrastructure creating new opportunities for hardware manufacturers as well as the cloud service providers.

In this paper, we compare the performance of a traditional suprcomputer named SuperMikeII located in LSU and our state of the art bigdata analytics cluster called SwatIII located in Samsung, Korea) while handling a data-intensive workload. and show how we achieve x-speedup using only 1/x nodes and subsequently x% performance to price benefit in SwatIII. Our analysis is based upon our benchmark large-scale Parallel Genome Assembler (PGA) based upon Hadoop and Giraph. Our assembly pipeline consists of a huge amount of short read analysis using Hadoop, followed by a large de Bruijn graph analysis using Giraph, thus serving as a very good real world example of data as well as compute intensive workload.

I. Introduction

Scientists in different fields are increasingly handling huge amount of bigdata produced by different experimental facilites which make the so called compute intensive scientific applications a severe data intensive endeavor. Starting from astronomical data analysis to coastal simulation, from social data analysis to genome assembly, this huge volume of data poses several challneges from effectively storing and managing to optimally processing it. The fundamental model of computation involved in the scientific applications is rapidly changing in order to address these challenges. Deviating from the decade old compute intensive programming paradigm like MPI, Grid etc. many HPC aficionados has started using the current state of the art big data analytics software like Hadoop, Giraph etc. for their data intensive scientific workloads.

Consequently, the traditional supercomputers even with tera to peta FLOP scale processing power are found to yield suboptimal performance concluding the processing power is not the only factor of actual performance for these data intensive workloads. The cumulative effect of CPU, memory, disk and network architecture on the over all performance make the job of providing efficient yet cost-effective hardwares more challenging, however, opening new opportunities for the hardware manufacturers. As a result, there is a growing interest in both HPC community as well as the hardware companies to address the challenges involved in providing cost-efective high performance testbeds that will drive the next generation data intensive scientific research. Furthermore, in the last few years, the scientific community has also experienced several benefits of pay-as-you-go cloud services (eg. Amazon-Cloud, Penguin, R-HPC etc) including the elasticity of resources with reduced setup cost and time which also has a catalytic effect on this interest. As a consequence, commercial cloud service providers started investing a lot to update and upgrade their infrastructures. Also, millions of dollars are being spent in programs like NSFCloud where several academic organizations and manufacturing companies collaborated to enable the academic research community to develop and experiment with novel cloud architectures.

Despite of this growing interest in both scientific as well as industrial community, there is very limited understanding of the performance characteristics of the current state of the art bigdata analytics softwares when applied for high performance data intensive scientific workloads on different hardware infrastructure. Thus, we found it extremely important to evaluate different types of distributed cyber infrastructure to understand the limitation in traditional supercomputers as well as the impact of different types of storage media, networking technologies and the overall cluster architecture and organization on the state of the art bigdata analytics softwrares in the context of a real world data intensive high performance complex scientific workload.

In this work, we use large scale de novo genome assembly as one of the most challenging and complex real world example of high performance computing that recently made its way to the forefront of bigdata challenges. De novo genome assembly reconstructs the entire genome from fragmented parts called short reads when no reference genome is available. The assembly pipeline consists of huge amount of short read analysis followed by a complex analysis on a largescale graph (refer to **??**), thus, serving as a very good example of both

data as well as compute intensive scientific workload.

Specifically, in this paper, we juxtapose the performance of different distributed cyber infrastructure with a large scale parallel genome assembler, called PGA, that we developed using Hadoop and Giraph. We present the performance result of PGA atop three different types of clusters including LSU supercomputer, SuperMikeII and two different clusters SwatIII and CeresII located in Samsung, korea, which we built as a prototype of state-of-the-art bigdata analytics clusters.

Our performance comparison is divided into two parts. In the first part, we provide insights on several architectural perspective including number of disk, type of storage media, amount of memory etc. in order to point out the limitations in SuperMikeII, and how we alleviate those by scaling up SwatIII and achieve almost x-speedup while using only 1/y nodes than SuperMikeII. However, we believe that price, power and cost are as important as raw execution time for a long term deployment. So, in the next part, we focused on comparing performance to price, space and power. We found almost x% improved performance to price in SwatIII than SuperMikeII considering long term deployment.

The rest of the paper is organized as follows: Section-II describes the prior works related to our study. In section-III we discuss the programming model offered by Hadoop and Giraph as well as provide a brief overview of the limitation in traditional supercomputers. Section-IV describes the assembly workload. In section-V we evaluate different distributed cyber infrastructure for the assembly workload.

II. Related Work

Earlier studies showed that Hadoop can be useful for data intensive scientific workloads [42]. Consequently, a growing number of codes in several scientific areas such as bioinformatics, geoscience are currently being written using open source state of the art bigdata analytics software like Hadoop, Giraph etc. [24]. Many of the traditional supercomputers also started using myHadoop [24] to provide the scientists an easy interface to configure Hadoop on-demand. However, there is very limited prior work that evaluated different distributed cyber infrastructures for these softwares when applied for data intensive scientific workload. This leaves a fundamental question yet to be answered: *how does a next generation high performance computation cluster should look like to handle data intensive scientific workload*. In this section we provide the related works for our study.

BigData analytics softwares: Hadoop [4] offers a simple, easily scalable disk-based map-reduce abstraction. HBase [6] is a NoSQL-based distributed linearly scalable key-value store targetted the applications that need random, realtime read or write access to tera/peta byte scale data residing in disk. Similarly, Hive [7], Impala [8] etc. are some of the popular disk-based NoSQL Database which provide the users with an SQl like query interface. On the other hand, Piccolo [17] and Redis [16] are two in-memory distributed key-value store, aimed at applications that need low-latency finegrained random access. Giraph [18] is a synchronous, vertex centric, in-memory graph processing framework originated as the open-source counterpart to Google's Pregel [2] that we analyzed in our work. GraphLab [20] is a faster asynchronous graph processing framework mainly motivated to provide the users a framework to write correct machine learning algorithms. Resilient Distributed Datasets (RDDs) [12] in the Spark system, offers a unified in-memory solution for all batch processing, Stream prcessing [13], SQL query [15] and graph processing [21]. Although, the computation model has evolved enough in the last few years to handle data intensive complex scientific workload, the choice of underlying hardware infrastructure still remains a major challenge.

Evaluation of Hadoop for scientific workload on existing superComputers: With growing number of scientific applications written in Hadoop, many different groups studied the performance of Hadoop on different existing supercomputers that they have access to. Jha [43] observed the convergence between traditional HPC and current state of the art bigdata analytics softwares and evaluated both of them in different supercomputing environment with k-means clustering as an example. Fadika [42] studied the performance of Hadoop for common HPC workload namely filter, merge and append. Guo [45] analyzed different graph processing framework with graph500 [36] BFS workload. Although, thse studies provide excellent insights on performance of current state of the art bigdata analytics softwares for different scientific applications, their analysis is confined into the domain of existing supercomputers, thereby, unable to address whether or not we can get better performance in other cyber infrastructure.

Evaluation of Hadoop for enterprise workload on different cyber infrastructure: Several performance analysis studies have been made with Hadoop atop different types of storages (SSD and HDD) and highspeed network interconnects (Infiniband and Ethernet etc). Moon [25] showed significant cost benefit by storing intermediate Hadoop data in SSD, leaving the HDDs to store Hadoop Distributed File System (HDFS [5]) source data. Wu [28] found that Hadoop performance can be increased almost linearly with the increasing fraction of SSDs in the storage system. Ahn [29] identified in a virtual environment overhead of virtualization is minimized with SSDs. Tan [26] analyzed the performance of SSD and HDD of different type of workloads involving different IO patterns and found better performance in using SSD. Vienne [30] evaluated the performance of Hadoop on different high speed interconnects such as 40GigE RoCE and Inifiniband FDR and found InfiniBand FDR yields the best performance for HPC as well as cloud computing applications. Similarly, Yu [31] found improvedperformance of Hadoop in traditional supercomputers due to high speed networks. From the perspective of a cost effective deployment, Appuswamy [32] studied the scale-out and scale-up performance for different enterprise level Hadoop job and found better perfromance price to performance in scaled up system. On the contrary, Michael [33] reached entirely different conclusion for interactive query. All of the above studies have been performed either with existing benchmarks like HiBench[] or for enterprise level analytics workloads such as log processing etc, thus, unable to address the HPC aspect of Hadoop in terms of efficient hardware provisioning. Furthermore, very limited studies consider the in-memory graph processing frameworks like Giraph, although, graph analysis is a core part of many analytics workloads.

From the above survey, we found a gap in the existing studies in terms of evaluating different distributed cyber infras-

tructure for current state of the art bigdata analytics software for a real world data intensive scientific workloads. On the other hand, millions of dollars are being invested in programs like NSFcloud with the goal "to provide an experimental platform enabling the academic research community to drive research on a new generation of innovative applications of cloud computing and cloud computing architectures" [46] . Hence, we found it extremely important to find out the limitations in traditional supercomputers in terms of underlying hardware infratructure and present a study addressing how to alleviate those limitations in an efficient yet cost-effective manner.

III. Bigdata Softwares on Traditional Supercomputers

A. Hadoop

Hadoop was originated as the opensource counter part of Google's Map-Reduce [1]. Hadoop has two different components: Hadoop Distributed File System (HDFS) and a mapreduce programming abstarction. HDFS splits huge volume of data into small disjoint sets called blocks (typically of size 64mb to 128mb) and distributes those accross the cluster. A user defined map function is applied to each blocks parallely in order to extract information from each records in the form of key-value pair. These intermidiate key-value pairs are then partitioned on the basis of keys where each key gets a list of values. Finally, a user defined reduce function is applied to the value-list of each key independently and the final output is wrtten to HDFS.

B. Giraph

Large scale graph analysis is a core part of many supercomputing workload. Apache Giraph is an iterative in-memory grpah processing framework that is implemented on top of Hadoop's map-reduce implementation. It is originated as the open-source counterpart to Google's Pregel [2]. Giraph is inspired by Bulk Synchronous Parallel model [3] where computation proceeds in supersteps. In each superstep all vertices of the graph excutes different istances of the same progam called vetrex-program simultaneosly without ineracting with other verties which is similar to map tasks of Hadoop After each superstep all the vertices send messages to other vertices normally cotaining the output of its vertex-progam-instance. Once all the messages are recieved by the intended vertices, the next supesrtep starts and the pocess iterates until all the vertices vote to halt simultaeously.

C. Limitations in traditional supercomputers

Earlier studies [42], [44] as well as our experience show that Hadoop and other softwares in its eco system like Giraph can be useful for data-intensive scientific applications, however, the underlying storage, memory as well as the computation model differs severely from other parallel processing frameworks like MPI. The challenges involved in optimal processing of these data-intensive workload needs to be addressed possibly by changing the underlying hardware infrastructure. In this section we provide a brief overview on the limitations in existing supercomputers.

Storage: In order to provide high io-bandwidth, Hadoop colocates data and computation. Unlike other parallel file system like Lustre which stores data in dedicated io-servers, HDFS relies on local file system. It stores the data in the same nodes where the computation takes place requiring high storage space in the compute nodes. Furthermore, the intermidiate output of each mapper is temporarily stored in the local filesystem of the corresponding node, which may be a magnitude higher than the final output especially in the case of a shuffle intensive job. On the other hand, in a traditional supercomputing environment each compute node is provided with less amount of storage typically provided with one disk ranging from 250gb to 500gb. This small amount of storage not only limits data size to be handled but also slow down the process because of lower IOPS. Although, scaling out in terms of compute nodes may alleviate both of these issues, it does in the cost of lower CPU utilization. In the subsequent sections, we show how number of disk per node, as well as the type of the storage media (SSD/HDD) impact the performance and price of a Hadoop workload in the context of large scale genome assembly.

Memory: Graph processing typically involves many iteration and random access to the data which is conventionally addressed with in-memory solutions. Memory system that is used in most of the supercomputers shows lower capacity per core and fewer independent channels [36]. Complicating the scenario, the performance is again hindered by high message passing over network for Giraph, which is designed to facilitate BSP model. In our study, we show the impact of provisioning more memory per core in a Giraph workload.

Swapping/Paging: Programs handling huge amount of data frequently perform swap in and out between memory and swap-space of the disk especially when the memory per core is small. Traditional supercomputers with small amount of memory per core and small amount of storage per compute node frequently run out of memory as well as swap-space, consequently failing the entire job. Furthermore, normal HDD with less throughput (than SSD) spends lots of time for io in case of swapping which adversely affects the performance of the application. In our analysis, we tweak the in-memory java-heap-space and the swap-space (both both SSD and HDD) in our prototype bigdata analytics cluster to understatnd its impact on the performance as well as the price.

IV. Evaluation Methodology

A. Genome Assembly Workload

De novo genome assembly refers to construction of an entire genome sequence from a large amount of short read sequences when no reference genome is available. De Bruijn graph construction and removal of sequencing errors (tips and bubble) from this graph is central to de novo sequencing. Finally, resolving repeated regions followed by a scaffolding phase produces long size scaffolds that represents a region in the actual genome.

We classified de novo sequencing in three different phases like other assemblers. *a*) De Bruijn graph construction *b*) Graph simplification and *c*) Scaffolding. We store short reads in fastq format in hdfs as input to PGA. In the first phase, we use Hadoop in order to build de Bruijn graph from

these short reads. Once the graph is constructed we use Giraph in the subsequent phases to analyze the graph in order to construct appreciably long contigs and scaffolds. In this section we provide a brief overview of each stage of the assembler.

1) De Bruijn graph construction: This phase is inspired by Contrail, another Hadoop based genome assembler. The de Brujin graph is constructed using MapReduce by scanning each read in the mapper. In the map phase, each read is divided into several short fragments of length k known as k-mer. Each two subsequent k-mers are emitted as key-value pairs where the first one (key) represents a vertex in the graph and the second one (value) represents the outgoing edge from the key. Similar process is repeated for the reverse complement of the reads are also considered. After the map function completes, the shuffle phase partitions the intermidiate key-value pairs on the basis of key which effectively collects the edges of the graph emitted from the same source k-mer. Finally, the reduce function aggregates the edges (value-list) of each source k-mer and saves the graph structure in HDFS.

2) Graph Simplification: This phase invokes a series of Giraph jobs each of which perform the following and write the output to the HDFS which seves as the input to the next Giraph job. The process continues until no linear chains are found in the graph.

Compression: The first step that follows after building the graph is compressing the linear chain of nodes in the graph. Giraph reads the graph from HDFS in a predefined adjacency list format where the source k-mer serves both as vertex id and value. The non-branching linear paths of vertices are compressed into single vertex without the loss of any information. In one superstep each compressible vertex is tagged as either head or tail with equal probability and send a meassage containing the tag to the immidiate predecessor and successor. In the next superstep the head-vertices are merged with corresponding tail-vertices. The value of the head is updated accordingly. This process continues for i supersteps until there is no compressible vertex remaining in the graph.

Tip removal: Tips are formed because of errors in the end of the short reads. Removing the tips from the de Bruijn Graph is a straight forward process. After compressing the graph, in a single superstep the vertices with no outgoing edge and the value-length less than a threshold (normally set to $2k$) are deleted from the graph.

Bubble removal: Bubbles are introduced in the DBG because of errors in the middle of the short reads. Bubbles are formed when two paths start and end at the same vertices. After compressing the linear chain, the objective of bubble removal is to group the vertices by the same predecessor and successor in the entire graph and from each group keep only the node which has the highest frequency support. In one superstep every node matching this criteria sends the cumulative frequency to their immidiate successor. In the next superstep successor nodes compute difference in frequency and delete all the nodes with lower frequency. Remember we calculated the frequency during the compression phase.

3) Scaffolding: The first step of scaffolding determines which contigs are linked by matepairs, and their relative orientation and separation. By convention, mated reads have the same name except for their suffix (either 1 or 2). PGA therefore finds all mate-linked contigs using a single MapReduce cycle by emitting from the mapper mate messages consisting of the read name without the suffix as the key, and the contig name, read orientation, and read offset as the value.

Next, we developed a graph hop method to find the exact path between the linked nodes

B. Input Data

High throughput next generation DNA sequencing machines like Illumina Genome Analyzer produce huge amount of short read sequences typically in the scale of several GigaBytes to Terabytes. Furthermore, the size of the de Bruijn graph built from these vast amount of short reads may be another magnitude higher than the reads itself making the entire assembly pipe line severely data-intensive. In this paper, we use the bumble bee genome and human genome sequence as a representative data sets. The first one, bumble bee genome is available in Genome Assembly Gold-standard Evaluation (GAGE [41]) website in fastq format. The data size is 85GB containing almost 1billion reads. The size of the de Bruijn graph produced by it is 90GB. The second data set, the human genome is openly avalible in NCBI web site with accession no. SRX10639 in a compressed SRA format. We decompress it using NCBI SRA toolkit which produce 452GB of short read sequence data in fastq format. The size of the de Bruijn graph produced by it is almost 3.8TB which is almost 9 times higher than the reads itself. The variation between the size of the input reads and the corresponding de Bruijn graph depends upon the sequencing-quality or pre-assembly-processing [39], [40] that is out of the scope of this paper.

C. Experimental Testbeds

We compare the result of three different clusters with varying architecture. In this section we prvide the details of the clusters. The first one is SuperMikeII, the LSU HPC resource that represents the domain of traditional supercomputer. This LSU supercomputer offers total 382 computing nodes, however, maximum of 128 can be allocated at a time. Each node has two 2.6GHz 8-core Sandy Bridge Xeon 64-bit processors, 32GB RAM, thus offering maximum 2GB of memory per core. Each node has 500GB local storage (normal Hard Disk Drive) available as filesystem data storage for Hadoop. All nodes are connected through 40Gb/s infiniband network with 2:1 blocking ratio.

On the other hand, we built a prototype of the current state of the art bigdata analytics cluster, SwatIII in Samsung, Korea with HDD and SSD variant to observe the impact of different storage media on Hadoop and Giraph job. Each node of SwatIII is equipped with sixteen 2-core Intel Xeon 64bit processors, 256GB RAM, thus yielding maximum 8GB memory per core. Each node in this cluster has 1 TB of local storage in either variant. Observe, each node in SwatIII offers almost double processing power than each node in SuperMikeII in terms of number of cores. However, in all the experiments discussed in this paper we use the same number of total cores in both the clusters in order to eliminate the impact of procesing power and do a fair comparison in terms of the effect of io-bandwidth, network-bandwith and memory-bandwidth/core which is the main focus of this paper.

	Super MikeII @LSU	Big CRON @LSU	SwatIII @Samsung	CeresII @Samsung
Number of nodes	128	128	15	33
Processor	2*2.6GHz 8-Core Sandy Bridge Xeon 64-bit	2*2.4GHz 4-Core Intel Xeon 64-Bit	16*2.6GHz 2-core Xeon 64-bit	2.3GHz 2-core Xeon 64-bit
Number of cores per node	16	8	32	2
Total number of cores	2048	1024	512	66
RAM per node (GB)	32	32	256	16
Total memory	4TB	4TB	4TB	528GB
Storage per node	500GB HDD	1TB HDD	1TB HDD/ SSD	256GB HDD
Total Storage	64TB HDD	64TB HDD	16TB HDD/ SSD	7TB HDD
Network	40Gb/s Inifiniband 2:1 bloking	40Gb/s Infiniand no blocking	10Gb/s Ethernet	10Gb/s virtualized Ethernet

TABLE I. TETBEDS

V. RESULT AND ANALYSIS

In this section we present the performance result of different clusters to understand the impact of different storage and network architechitecture. All experiments are performed as many times until 95% confidence level is reached.

A. Understanding the workload

Figure-**??** shows the average disk and memory utilization of PGA for assemblying bumble bee and human genome for three different phases of PGA. The first phase, building de Bruijn graph is a shuffle intensive mapreduce job which writes almost xGB intermidiate data for bumble bee and almost xGB for human genome however the final output of the reducer that is written to HDFS is 85GB and 3.8TB respectively for bumble bee and human genome. In the second phase, the entire graph is taken in the memory and analyzed using Giraph making it severely memory intensive. At the same time, the output of each giraph job is written to HDFS which serves as the input to the next job making the perfromance of this phase disk-bound also.

B. Measurement baseline

In order to do both a fair comparison as well as pinpointing the limitations we changed SwatIII environent according to SuperMikeII specifications. The parameters that we changed are listed in table-**??**. Figure-**??** shows performance of assemblying bumble bee genome in using 256 cores, 16 disks and an aggregated 1TB of RAM space. Then in the subsequent sections we turn on each feature one by one and compare its benefit in order to provide a guideline which needs to be taken care of.

C. Effect of number of disk per node

. In this experiment we added more(??) number of storage drive in the SWATIII cluster and distributed the intermidiate output of Hadoop according to that. Since the total IOPS increases in the cluster Hadoop was expected to yield better performance. Figure-**??** shows the result of adding more number of storage drive in the cluster. Also it is important to notice that, adding each SSD to the cluster shows almost ??% benefit over adding HDD.

D. Effect of Hyper threading

Figure-**??** shows the execution time of each phase of PGA when we enable hyperthreading in SWATIII. In each variant of SWATIII we noticed almost ?? speedup simply enabling the hyperthreading. On the other hand, we observed almost ?? speedup when normalized the performance with SuperMikeII.

E. Effect of Java Heap Space and Virtual Memory

The impact of java heap space and virtual memory is a long standing issue. Although, the performance is found to vary according to the java heap space allocated to each worker the proper explanation is still unknown. In this section we tweak the java heap space and the Linux virtual memory configuration to notice the effect of swapping on different types of storage media. Figure-**??** shows that SSD with more IO throughput performs better than the HDD in SWATIII with lower amount of Virtual H. Also,

F. Effect of adding more memory per node

1) Discussion:

G. Performance to price comparison

VI. CONCLUSION

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51, no. 1 (2008): 107-113.

[2] Malewicz, Grzegorz, Matthew H. Austern, Aart JC Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. "Pregel: a system for large-scale graph processing." In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 135-146. ACM, 2010.

[3] Cheatham, Thomas, Amr Fahmy, Dan Stefanescu, and Leslie Valiant. "Bulk synchronous parallel computinga paradigm for transportable software." In Tools and Environments for Parallel and Distributed Systems, pp. 61-76. Springer US, 1996.

[4] White, Tom. Hadoop: the definitive guide: the definitive guide. " O'Reilly Media, Inc.", 2009.

[5] Borthakur, Dhruba. "The hadoop distributed file system: Architecture and design." Hadoop Project Website 11, no. 2007 (2007): 21.

[6] George, Lars. HBase: the definitive guide. " O'Reilly Media, Inc.", 2011.

[7] Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. "Hive: a warehousing solution over a map-reduce framework." Proceedings of the VLDB Endowment 2, no. 2 (2009): 1626-1629.

[8] Wanderman-Milne, Skye, and Nong Li. "Runtime Code Generation in Cloudera Impala." IEEE Data Eng. Bull. 37, no. 1 (2014): 31-37.

[9] Chen, Rishan, Xuetian Weng, Bingsheng He, Mao Yang, Byron Choi, and Xiaoming Li. "On the efficiency and programmability of large graph processing in the cloud." Microsoft Research TechReport (2010).

[10] Kang, U., Charalampos E. Tsourakakis, and Christos Faloutsos. "Pegasus: A peta-scale graph mining system implementation and observations." In Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, pp. 229-238. IEEE, 2009.

[11] Kang, U., Hanghang Tong, Jimeng Sun, Ching-Yung Lin, and Christos Faloutsos. "Gbase: a scalable and general graph management system." In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1091-1099. ACM, 2011.

[12] Zaharia, Matei, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pp. 2-2. USENIX Association, 2012.

[13] Zaharia, Matei, Tathagata Das, Haoyuan Li, Scott Shenker, and Ion Stoica. "Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters." In Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing, pp. 10-10. USENIX Association, 2012.

[14] Gonzalez, Joseph E., Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. "Graphx: Graph processing in a distributed dataflow framework." In Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI). 2014.

[15] Xin, Reynold S., Josh Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Shark: SQL and rich analytics at scale." In Proceedings of the 2013 ACM SIGMOD International Conference on Management of data, pp. 13-24. ACM, 2013.

[16] Carlson, Josiah L. Redis in Action. Manning Publications Co., 2013.

[17] Power, Russell, and Jinyang Li. "Piccolo: Building Fast, Distributed Programs with Partitioned Tables." In OSDI, vol. 10, pp. 1-14. 2010.

[18] Avery, Ching. "Giraph: Large-scale graph processing infrastructure on hadoop." Proceedings of the Hadoop Summit. Santa Clara (2011).

[19] Tian, Yuanyuan, Andrey Balmin, Severin Andreas Corsten, Shirish Tatikonda, and John McPherson. "From" think like a vertex" to" think like a graph." Proceedings of the VLDB Endowment 7, no. 3 (2013): 193-204.

[20] Low, Yucheng, Joseph E. Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E. Guestrin, and Joseph Hellerstein. "Graphlab: A new framework for parallel machine learning." arXiv preprint arXiv:1408.2041 (2014).

[21] Xin, Reynold S., Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. "Graphx: A resilient distributed graph system on spark." In First International Workshop on Graph Data Management Experiences and Systems, p. 2. ACM, 2013.

[22] Roy, Amitabha, Ivo Mihailovic, and Willy Zwaenepoel. "X-stream: Edge-centric graph processing using streaming partitions." In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, pp. 472-488. ACM, 2013.

[23] Kyrola, Aapo, Guy E. Blelloch, and Carlos Guestrin. "GraphChi: Large-Scale Graph Computation on Just a PC." In OSDI, vol. 12, pp. 31-46. 2012.

[24] Krishnan, Sriram, Mahidhar Tatineni, and Chaitanya Baru. "myHadoop-Hadoop-on-Demand on Traditional HPC Resources." San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego (2011).

[25] Moon, Sangwhan, Jaehwan Lee, and Yang Suk Kee. "Introducing SSDs to the Hadoop MapReduce Framework." In Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on, pp. 272-279. IEEE, 2014.

[26] Tan, Wei, Liana Fong, and Yanbin Liu. "Effectiveness Assessment of Solid-State Drive Used in Big Data Services." In Web Services (ICWS), 2014 IEEE International Conference on, pp. 393-400. IEEE, 2014.

[27] Kang, Yangwook, Yang-suk Kee, Ethan L. Miller, and Chanik Park. "Enabling cost-effective data processing with smart ssd." In Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on, pp. 1-12. IEEE, 2013.

[28] Wu, Dan, Wenhai Luo, Wenyan Xie, Xiaoheng Ji, Jian He, and Di Wu. "Understanding the Impacts of Solid-State Storage on the Hadoop Performance." In Advanced Cloud and Big Data (CBD), 2013 International Conference on, pp. 125-130. IEEE, 2013.

[29] Ahn, Sungyong, Sangkyu Park, Jae-Ki Hong, and Wooseok Chang. "Performance Implications of SSDs in Virtualized Hadoop Clusters." In Big Data (BigData Congress), 2014 IEEE International Congress on, pp. 586-593. IEEE, 2014.

[30] Vienne, Jerome, Jitong Chen, Md Wasi-Ur-Rahman, Nusrat S. Islam, Hari Subramoni, and Dhabaleswar K. Panda. "Performance analysis and evaluation of infiniband fdr and 40gige roce on hpc and cloud computing systems." In High-Performance Interconnects (HOTI), 2012 IEEE 20th Annual Symposium on, pp. 48-55. IEEE, 2012.

[31] Yu, Jie, Guangming Liu, Wei Hu, Wenrui Dong, and Weiwei Zhang. "Mechanisms of Optimizing MapReduce Framework on High Performance Computer." In High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on, pp. 708-713. IEEE, 2013.

[32] Appuswamy, Raja, Christos Gkantsidis, Dushyanth Narayanan, Orion Hodson, and Antony Rowstron. "Scale-up vs Scale-out for Hadoop: Time to rethink?." In Proceedings of the 4th annual Symposium on Cloud Computing, p. 20. ACM, 2013.

[33] Michael, Maged, Jose E. Moreira, Doron Shiloach, and Robert W. Wisniewski. "Scale-up x scale-out: A case study using nutch/lucene." In Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, pp. 1-8. IEEE, 2007.

[34] Chen, Yanpei, Sara Alspaugh, and Randy Katz. "Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads." Proceedings of the VLDB Endowment 5, no. 12 (2012): 1802-1813.

[35] Huang, Shengsheng, Jie Huang, Yan Liu, Lan Yi, and Jinquan Dai. "Hibench: A representative and comprehensive hadoop benchmark suite." In Proc. ICDE Workshops. 2010.

[36] Murphy, Richard C., Kyle B. Wheeler, Brian W. Barrett, and James A. Ang. "Introducing the graph 500." Cray Users Group (CUG) (2010).

[37] Marathe, Aniruddha, Rachel Harris, David K. Lowenthal, Bronis R. de Supinski, Barry Rountree, Martin Schulz, and Xin Yuan. "A comparative study of high-performance computing on the cloud." In Proceedings of the 22nd international symposium on High-performance parallel and distributed computing, pp. 239-250. ACM, 2013.

[38] Pevzner, Pavel A., Haixu Tang, and Michael S. Waterman. "An Eulerian path approach to DNA fragment assembly." Proceedings of the National Academy of Sciences 98, no. 17 (2001): 9748-9753.

[39] Medvedev, Paul, Eric Scott, Boyko Kakaradov, and Pavel Pevzner. "Error correction of high-throughput sequencing datasets with non-uniform coverage." Bioinformatics 27, no. 13 (2011): i137-i141.

[40] Yang, Xiao, Sriram P. Chockalingam, and Srinivas Aluru. "A survey of error-correction methods for next-generation sequencing." Briefings in bioinformatics 14, no. 1 (2013): 56-66.

[41] Salzberg, Steven L., Adam M. Phillippy, Aleksey Zimin, Daniela Puiu, Tanja Magoc, Sergey Koren, Todd J. Treangen et al. "GAGE: A critical evaluation of genome assemblies and assembly algorithms." Genome research 22, no. 3 (2012): 557-567.

[42] Fadika, Zacharia, Madhusudhan Govindaraju, Richard Canon, and Lavanya Ramakrishnan. "Evaluating hadoop for data-intensive scientific operations." In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, pp. 67-74. IEEE, 2012.

[43] Jha, Shantenu, Judy Qiu, Andre Luckow, Pradeep Mantha, and Geoffrey C. Fox. "A tale of two data-intensive paradigms: Applications, abstractions, and architectures." In Big Data (BigData Congress), 2014 IEEE International Congress on, pp. 645-652. IEEE, 2014.

[44] Matsunaga, Andra, Maurcio Tsugawa, and Jos Fortes. "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications." In eScience, 2008. eScience'08. IEEE Fourth International Conference on, pp. 222-229. IEEE, 2008.

[45] Guo, Yong, Marcin Biczak, Ana Lucia Varbanescu, Alexandru Iosup, Claudio Martella, and Theodore L. Willke. "How well do graph-processing platforms perform? an empirical performance evaluation and

analysis." In Parallel and Distributed Processing Symposium, 2014 IEEE 28th International, pp. 395-404. IEEE, 2014.

[46] https://www.chameleoncloud.org/nsf-cloud-workshop/.