IBM POWER8® HPC System Accelerates Genomics Analysis with SMT8 Multithreading

Highlights

- Results of Hadoop enabled Genome Analysis on an IBM POWER8 (40 Node) HPC Cluster
- Performance results utilizing IBM SMT8, Simultaneous Multi-Threading
- Use of the Spectrum Scale file system for data and compute intensive workloads
- De Bruijn graph construction

Contents

Executive summary

Current compute technologies and scientific methods for Big Data analysis today are demanding more compute cycles per processor than ever before, with extreme I/O performance also required. The capabilities of Intel Hyper-Threading that offers only two simultaneous threads per core limit the results of these recent advances in Big Data analysis techniques.

Despite multiple prior Hadoop Genome analysis attempts on an existing Intel based LSU HPC cluster, a large metagenome dataset could not be analyzed in a reasonable period of time on existing LSU resources. Knowing the extraordinary capabilities for big data analysis offered by IBM Power Systems, the LSU Center for Computational Technologies staff and researchers turned to IBM for help.

The existing Hadoop based method was ported to an IBM Customer Center 40 Node POWER8® cluster running Ubuntu 14.1 and the Spectrum Scale GPFS file system. The result was astonishing: ***The 1st phase of the Hadoop analysis on the huge 3.2TB metagenome dataset was rendered in 6.25 hours*****, using only 40 nodes**.

LSU and IBM are excited to report on these promising advances in this data and compute intensive scientific use case of Hadoop.

Further research work with IBM includes the use of NOSQL key value stores for the graph analysis.

Introduction and Science Challenge

The genome input data size has already outpaced Moore's law and exceeded terabytes. The data size will continue to grow as sequencing technologies improve. This unprecedented volume of data and complex computational requirement creates a dire need for a scalable software solution, as well as an efficient distributed cyber infrastructure that can handle this data and compute intensive workload. Recently, Hadoop has emerged as the big data analytics framework. Genome scientists increasingly use Hadoop for their data and compute intensive scientific applications. In this work, we focus on evaluating underlying distributed cyber infrastructure to accelerate a Hadoop-based data- and compute- intensive genome assembly workload.

In particular, in this white paper, we evaluate the IBM POWER8 processor with respect to our Hadoop-based benchmark genome assembler. On a cluster of 40 POWER8 nodes, we analyze a 3.2TB of input metagenome data set with Hadoop producing results in 6 hours rendering 8.6TB of graph data structure (called de Bruijn graph).

Hadoop Programming Model

Hadoop was originated as the open-source counterpart of Google's MapReduce. It has two distinct modules: a distributed file system, and a MapReduce programming abstraction.

Hadoop reads the input data from the underlying distributed file system in the form of disjoint sets or partitions of records. Normally, Hadoop Distributed File System (HDFS) is used for this purpose. However, other parallel file systems (e.g., Lustre and GPFS) are frequently used for this purpose as well. In our IBM POWER8 cluster evaluation, we used IBM Spectrum Scale (formerly known as General Parallel File System, or GPFS) as the underlying parallel file system.

Then, in the MapReduce programming abstraction, a user-defined map function is applied to each disjoint set concurrently to extract information from each record in the form of intermediate key-value pairs. These key-value pairs are then grouped by the unique keys and shuffled to the reducers. Finally, a user-defined reduce function is applied to the value-set of each key, and the final output is written to the HDFS. The MapReduce framework enables data-and compute-intensive applications to run large volumes of distributed data sets over distributed compute nodes with local storage.

The 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. The problem of de novo genome assembly can be interpreted as a simplified de Bruijn graph traversal problem. Construction of this simplified de Bruijn graph from huge volumes of genome sequence data is the central workload of the de novo genome assembly pipeline. The graph construction process is severely data- and compute-intensive. We use Hadoop for this purpose.

Hadoop-based De-Bruijn graph construction (Data and Compute intensive workload)

Figure 1: Hadoop-based de Bruijn graph construction job

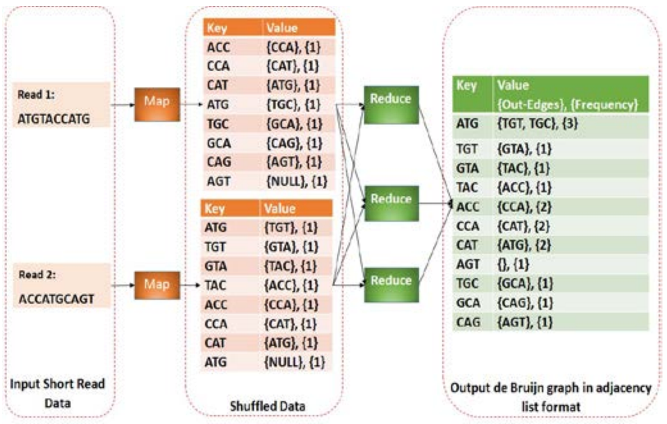


Figure 1 shows the overview of the MapReduce job. The map function scans through each line of the data file (written in fastq format) and filters out the lines containing only A, T, G, and C, i.e., the nucleotide characters. These lines are called short reads, or simply reads, which represent a small fragment of the entire genome. Then the same map task divides each read into several short fragments of length k, known as k-mers. Two adjacent k-mers are emitted as an intermediate key-value pair that represents a vertex and an edge (emitted from that vertex) in the de Bruijn graph. The reduce function aggregates the edges (i.e., the value-list) of each vertex (i.e., the k-mer emitted as a key) and, finally, writes the graph structure in the file system in an adjacency-list format. Based upon the value of k (determined by biological characteristics of the species), the job produces huge amounts of shuffled data. For example, for a read-length of 100 and k of 31, the shuffled data size is more than 6-times that of the original short read input.

Experimental Testbeds

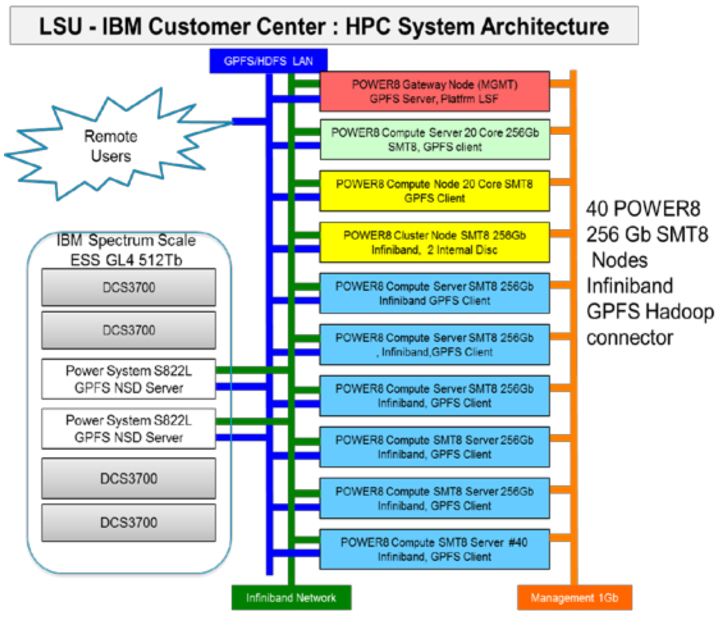
Table 1: Cluster configuration

System Type	IBM	LSU SuperMikeII
Processor	Two 10-core IBM Power Systems with POWER8	Two 8-core Intel SandyBridge Xeon
Maximum #Nodes used in various experiments	40	120
#Physical cores/node	20 (8 Simultaneous Multi-Thread)	16 (Hyper threading disabled)
#vcores/node	160	16
RAM/node (GB)	256	32
#Disks/node	5	3
#Disks/node used for shuffled data	3	1
Total Storage space/node used for shuffled data	1.8	0.5
Network	56Gbps InfiniBand(no blocking)	40Gbps InfiniBand (2:1 blockings)

Table 1 compares two different types of clusters that we used in our experiments. The first one is the IBM POWER8 based HPC cluster. Each Power System S824L server was equipped with 20 IBM POWER8 cores, each with maximum 8 SMTs available. Each server had 5-HDDs for improved sequential I/O bandwidth. Each node had 256GB RAM. All the POWER8 nodes are inter-connected with an InfiniBand connection. We used 40 nodes for this benchmark evaluation.
In contrast, each LSU SuperMikeII node is equipped with two 8-core Intel SandyBridge Xeon processors, 32GB of RAM, and 1 500GB-HDD. All the nodes of SuperMikeII are connected with a 40Gbps InfiniBand network with 2:1 blocking ratio.

Physical Compute/Storage Cluster Schematic

Figure 2: 40 Nodes IBM Power System S824L, SMT8 enabled. IBM Spectrum Scale Elastic Storage, 512TB.


LSU - IBM Customer Center : HPC System Architecture
GPFS/HDFS LAN
Remote Users
POWER8 Gateway Node (MGMT) GPFS Server, Platfrm LSF
POWER8 Compute Server 20 Core 256Gb SMT8, GPFS client
POWER8 Compute Node 20 Core SMT8 GPFS Client
POWER8 Cluster Node SMT8 256Gb Infiniband, 2 Internal Disc
POWER8 Compute Server SMT8 256Gb Infiniband GPFS Client
POWER8 Compute Server SMT8 256Gb , Infiniband,GPFS Client
POWER8 Compute Server SMT8 256Gb Infiniband, GPFS Client
POWER8 Compute SMT8 Server 256Gb Infiniband, GPFS Client
POWER8 Compute SMT8 Server 256Gb Infiniband, GPFS Client
POWER8 Compute SMT8 Server #40 Infiniband, GPFS Client
IBM Spectrum Scale ESS GL4 512Tb
DCS3700
DCS3700
Power System S822L GPFS NSD Server
Power System S822L GPFS NSD Server
DCS3700
DCS3700
40 POWER8 256 Gb SMT8 Nodes Infiniband GPFS Hadoop connector
Infiniband Network
Management 1Gb

Processor

As mentioned in Table 1, each IBM POWER8 node is equipped with 20 IBM POWER8 cores operating at 3.42 GHz. Each core can utilize 8 SMT threads (Simultaneous Multi-Threading), which equates to a total of 160 (20*8) virtual cores (vcores) visible to the OS. Conversely, each SuperMikeII node has two 8-core Intel Sandy Bridge Xeon processors with 2.6 GHz core speed with hyper-threading disabled.

The high number of vcores enabled by the IBM POWER8 SMT technology can accelerate data and compute-intensive scientific workloads tremendously, especially, when used in conjunction with YARN resource management. For example, a data-intensive Hadoop job can use a high number of concurrent YARN containers (mappers/reducer) while also using fewer vcores per container. On the other extreme, a severely constrained compute-intensive workload can launch fewer concurrent YARN containers while using more vcores per container. In our genome assembly workload, we optimized these numbers (i.e., the number of concurrent YARN containers, and the number of vcores per container) by observing prior job profiles.

Storage

To provide enough I/O bandwidth to each IBM POWER8 node, we used 5 hard disc drives (HDDs) (1 x 300GB and 4 x 600GB) per node, which cumulatively provide almost ~500MBps I/O throughput (assuming ~100MBps per HDD). One 600GB drive was used for O/S and one 300GB HDD was configured as /scratch, where Apache Hadoop was installed. The other three 600GB drives were used for shuffle space (mounted as /shuffle1, /shuffle2, /shuffle3). This high I/O throughput is exactly in contrast with SuperMikeII, which only has available one HDD per node.

A system's performance is always constrained by its slowest component, such as the I/O subsystem, or network, or both. A big data application, such as, our genome analytic application, on the other hand normally involves a huge number of I/O operations. In order to take advantage of the POWER8 data processing capabilities, I/O throughput needs to also be maximized. The throughput of each compute node can be increased linearly by scaling up the I/O subsystem with more numbers of Direct Attached Storage (DAS) devices. However, it should be noted that after a certain threshold on I/O throughput, the disk controller can be saturated, i.e., adding more disks does not improve the application's performance. We found that three discs for shuffle space provide enough I/O throughput to optimize the performance.

Memory

Each IBM POWER8 node had 256GB memory (RAM) as compared to only 32GB RAM in the LSU SuperMikeII system. The IBM node provides 16 independent, high bandwidth memory controllers, which allowed us to provide 16GB per memory controller. With the Intel system, due to the direct attached memory interface, adding more memory did not improve the memory bandwidth. It is worthy to mention here, that the high computational speed of the POWER8 processor eliminates the need for tuning Hadoop parameters related to memory buffer size (e.g., io.sort.mb, io.sort.factor, etc.) provided enough I/O bandwidth is available. These parameters are commonly increased for performance reasons.

For our Hadoop-based genome analytic workload, 256GB RAM per IBM POWER8 node (with 20 IBM POWER8 cores) delivered a good balance between the high processing speed and the memory bandwidth required.

Network

The IBM customer center system uses two 108-port Mellanox SX6506 FDR 56Gbps InfiniBand switches with non-blocking I/O for a balanced architecture. The high network bandwidth (non-blocking 12.1Tbps) provided by the SX6506 switch enables the high throughput needed for the analysis and capabilities of IBM POWER8 nodes. Comparatively, the LSU SuperMikeII cluster uses a 40Gbps InfiniBand network. Because the SuperMikeII cluster uses a 2:1 blocking factor, (a tradeoff between performance and price), the effective bandwidth between any two SuperMikeII:nodes is significantly reduced to only 950Mbps.

***The choice of network architecture is critical to the performance of Hadoop*.** In particular, the shuffle phase of a Hadoop job involves a huge data movement across the cluster. At this point, the data network is a critical path, and its performance and latency directly impact the execution time of the entire job-flow. The choice of a high performance, low latency, InfiniBand network with non-blocking I/O successfully addresses the following issues:

1) The current programming model, i.e., MapReduce,(in particular the shuffle phase) emphasizes bandwidth. Hence, the use of blocking is avoided.

2) With the rapid evolution of processor and storage technologies, each host (or, compute node) is capable of processing data in multiple GB per sec. rates. Therefore, in order for a compute node to take the advantage of the increased I/O bandwidth and processing power, the

network capacity should also be increased correspondingly.

Hadoop Configuration

In the entire benchmark evaluation, we used Apache Hadoop-2.5.2. The file system for the analysis was IBM Spectrum Scale 4.1.1 (GPFS with Hadoop connector) solution that has been on the market for many years. In this case we also used a new Hadoop GPFS connector. The GPFS Hadoop Connector provides a liaison between Hadoop MapReduce and the scalable parallel file system. The GPFS solution allows the system to utilize millions of high speed parallel files.

In this section, we briefly describe the most important Hadoop parameters that we configured to explore the application behavior atop IBM hardware.
Table 2 shows the Hadoop and YARN parameters that we used in the POWER8 cluster and SuperMikeII for the most optimized performance in each of the clusters.

Table 2: Hadoop parameter

Hadoop Parameters	Power Systems	SuperMikeII
Yarn.nodemanager.cpu.resource.vcore	120	16
Yarn.nodemanager.memory.mb	231000	29000
Mapreduce.map/reduce.cpu.vcore	4	2
Mapreduce.map/reduce.memory.mb	7000	3500
Mapreduce.map/reduce.java.opts	6500m	3000m

1) **Number of concurrent YARN containers:** As Table 2 suggests, for our data- and compute-intensive genome analytic application, we launched 30 concurrent Yarn containers (mappers/reducers) per IBM POWER8 node, where each container used 4 vcores. That amounts to a total of 120 vcores (out of 160) per node utilized, which was found to yield the most optimized performance. On the other hand, in SuperMikeII, we could only achieve the most optimized performance with 8 concurrent YARN containers per node with 2 vcores per container.

2) **Amount of memory and Java heap space per container:** As it can be seen in Table 2, in each node of any cluster, we kept almost 10% of the memory for the system use. The rest of the memory was equally divided among the concurrently launched YARN containers. The Java heap space per worker is always set to lower than the memory per container, as per normal recommendation.

3) **Total number of reducers:** Based on the observation of job profiles and prior experiences, we observed that 2-times the # of reducers than number of concurrent containers produced good performance in general cases.

Discussion: From the Hadoop configuration parameters, it is clear that the POWER8 SMT technology helps increase the level of parallelism, both in the job, as well as, individual task level verses the Intel Xeon processor. Even though only 4 more cores exist in POWER8 versus our Intel implementation, we found that turning on SMT8 thread capability accelerated the job significantly more.

Input Data

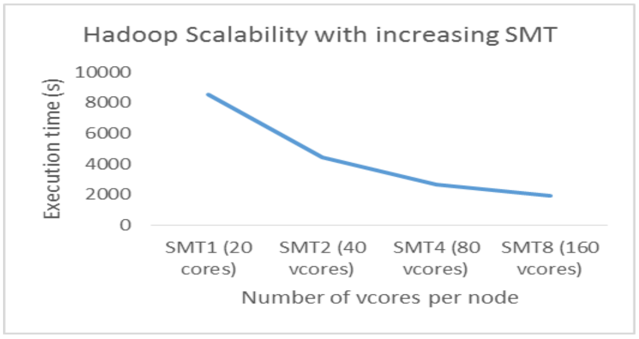
Table 3: Data set

Genome data set	Input size	Shuffle data size	Output size
Rice genome	12GB	70GB	50GB
Bumble bee genome	90GB	600GB	95GB
Metagenome	3.2TB	20TB	8.6TB

Table 3 shows the details of the input data set that was used in the benchmark evaluation. We used three different data sets. The first one is a small size rice genome data set. The second one was a 90GB data set of a moderate size Bumble Bee genome data. We used this data set to make a comparative study between SuperMikeII and the IBM POWER8 cluster. The data set is openly available on the public website of genome assembly gold standard evaluation (http://gage.cbcb.umd.edu/data/index.html). The second dataset is a metagenome data set generated by the Joint Genome Institute (JGI). The huge volume of the data with complex computation make this analysis challenging. We were only able to analyze the second data set in a reasonable amount of time due to the availability of SMT8 technology.

Scalability Test of Hadoop atop POWER8 SMT

Figure 3: Scalability of Hadoop with increasing SMT



In this experiment, we tested the scalability of our Hadoop application with increasing SMT of POWER8 processors. It is worthy to mention here that we performed this experiment on a 2-node IBM POWER8 cluster with the smaller size (12GB) rice genome data set before the actual benchmark evaluation at the Poughkeepsie customer center took place. The motivation of this experiment was to analyze the job profiles of our genome assembler atop POWER8 processor. In fact, we configured Hadoop (discussed earlier in this paper) based upon these experiments to get the most optimized performance.

Figure 3 shows the scalability of our Hadoop application with increasing the threads of each POWER8 based node. We started with disabling SMT on all compute nodes. Then we increased it until SMT8, the maximum threading available per compute node. At the same time, we changed the number of concurrent Yarn containers (thus, number of mappers and reducers) according to the total number of vcores visible to the OS.

Evaluation of IBM POWER8 Cluster for Hadoop

In this section, we compare the performance of the IBM Power Systems S824L and Intel SandyBridge Xeon processor with respect to our benchmark genome assembler with varying size of data (shown in Table 3) atop two different clusters (shown in Table 1). For a fair comparison, the impact of I/O should be minimized. To do that we used the same number of disks in both of the clusters for each of the experiments, thus keeping the aggregated I/O throughput similar across both clusters.

As shown in Table 1, each IBM node uses 3 disks and each SuperMikeII node uses 1 disk for shuffled data. Hence, for each of the experiments (which are all shuffle intensive as mentioned before), we used 3x more nodes in SuperMikeII than those used in the IBM POWER8 cluster. Even though, this configuration always results in 2.4x fewer IBM POWER8 cores than the Intel Xeon physical cores, the POWER8 cluster combined with SMT8 always showed 2.5x to 3x performance increase over SuperMikeII.

In summary, the IBM POWER8 cluster combined with SMT8 rendered 7.5x to 9x performance increase per compute server versus each Intel Xeon based LSU HPC cluster node.

This high performance gain implies that even if HyperThreading is enabled (i.e., 2 threads/core), and we assume Hadoop performance improves linearly with the number of vcores, POWER8 with SMT8 will show almost 3.75x to 4.5x performance increase per server compared to Intel Xeon nodes. In our opinion, performance of Hadoop is sub-linear most of the time, therefore POWER8 SMT8 will likely produce even better results based on the results we have seen.

Analyzing Small Size Rice Genome

Figure 4: Execution time of Rice genome

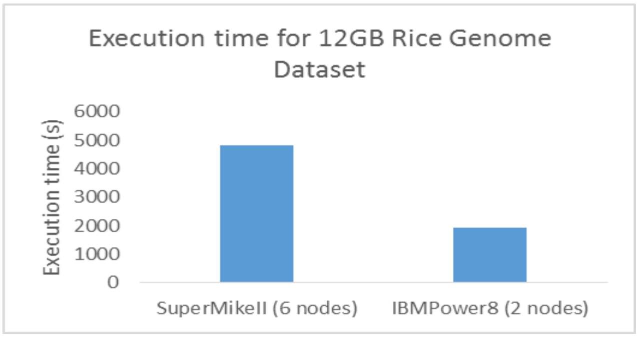


Figure 4 compares the execution time fo the IBM POWER8 cluster and the Intel Xeon based SuperMikeII cluster for a small size (12GB) rice genome data set using 2 and 6 nodes respectively. The small size fo data and small number of nodes minimizes the impact of the network. On the other hand, 3x more SuperMikeII nodes than IBM POWER8 nodes were required to keep the total I/O throughput similar across the clusters. The IBM POWER8 cluster rendered 2.5x performance improvement over the LSU SuperMikeII HPC cluster, while using 3x fewer nodes. In other words, the IBM cluster shows almost 7.5x improvement over SuperMikeII in terms of performance per server.

Analyzing Moderate Size Bumble Bee Genome

Figure 5: Execution time for 90GB Bumble Bee genome

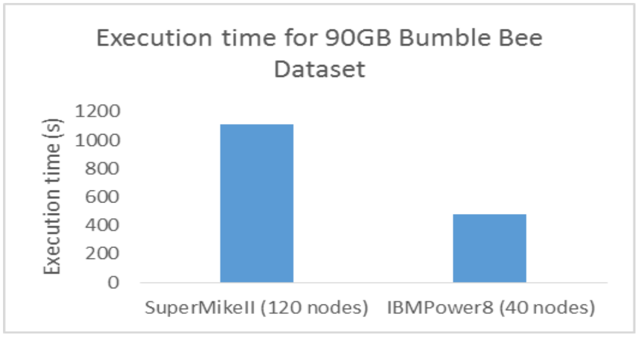
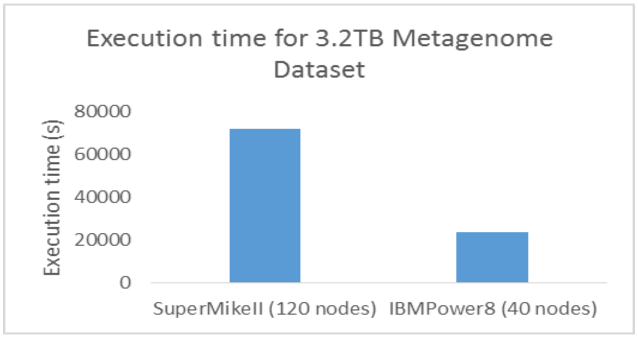


Figure 5 shows the execution time for graph construction in both IBM POWER8 cluster and Intel Xeon based LSU HPC cluster (SupermikeII). We used 40 nodes in IBM POWER8 cluster as opposed to 120 nodes in SuperMikeII. The IBM POWER8 cluster combined with SMT8 rendered almost 3x performance improvement over the LSU SuperMikeII HPC cluster while using 3x fewer nodes and 2.4x fewer physical cores.

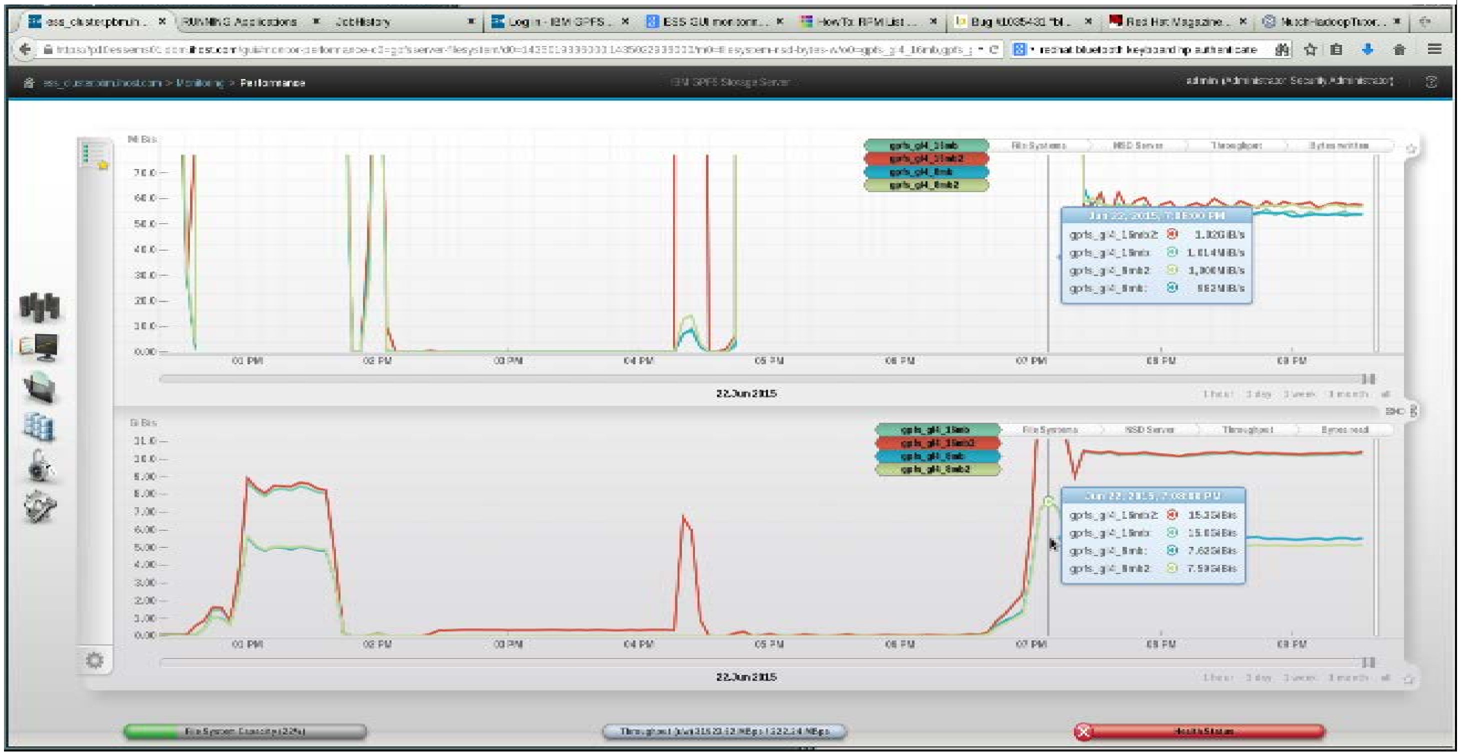
Analyzing Large Size 3.2TB Metagenome

Figure 6: Execution time for 3.2TB



To explore the capability of IBM POWER8 processor with large scale data, we analyzed a dataset size of 3.2TB of metagenome data. The process completed in 6 hours and 22 minutes on the IBM POWER8 cluster using only 40 nodes. This same process takes more than 20 hours to complete on 120 Intel nodes available at LSU. ***That is, for this huge data set also the IBM cluster produced more than 3x performance improvement using 3x fewer nodes, thereby producing more than 9x improvement in terms of performance/server.***

Figure 7: GPFS I/O Throughput



Evaluating GPFS for Hadoop

Benefit of IBM Spectrum Scale over HDFS

As mentioned earlier, we used IBM Spectrum Scale (formerly named GPFS) as the underlying distributed file system. All Hadoop applications run as-is, or with very minimal changes if Spectrum Scale is used as the underlying file system instead of HDFS. Spectrum Scale is a global parallel file system offering several features that benefit large scale big data clusters in terms of easy and cost effective data management. Some of the advantages of using GPFS over HDFS are as follows:

1) **Integrating Different Analytic Environments:** IBM Spectrum Scale is POSIX compliant. It provides a seamless environment for Hadoop and other traditional HPC analytics environments. For example, the output of an MPI application can easily work as an input to a Hadoop application, and vice versa.

2) **Information Life-cycle Management (ILM):** Spectrum Scale ILM policies can enable automatic archiving of unused or aging data to low performance disks. This is a major advantage over HDFS that minimizes the continually growing storage cost.in a big data analytic cluster.

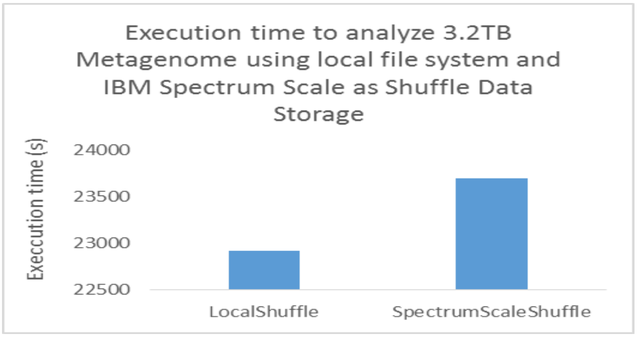
3) **Fault Tolerance:** Spectrum Scale does not have any single points of failure, whereas HDFS has a single point of failure (i.e., the Hadoop NameNode).

4) **Storage Space:** Furthermore, HDFS is mounted on local storage of compute nodes which may be limited in terms of storage capacity in an HPC cluster. GPFS uses dedicated I/O servers and normally provides a huge amount of storage space required in a big data HPC environment. Performance can also be designed to scale linearly with additions of NSD servers and scalable network bandwidth.

IBM Spectrum Scale as Temporary Shuffled Data Storage

Figure 7 shows GPFS yielding high I/O throughput (Peak 11GiB/s*4 = 44GiB/s) during the shuffle phase of our Hadoop job when configured properly.
Figure 8 shows the execution time of the metagenome analysis executing in similar time scale for both the cases (6hrs35min for GPFS shuffle comparing to 6hrs22min for local shuffle).

Figure 8: Performance comparison between local file system and GPFS as shuffle data storage



In this section, we explore the possibility of IBM Spectrum Scale as a storage space for temporary shuffled data. In general, Hadoop uses the local file system that is mounted on the DAS device of the compute node to keep the temporary shuffled data. For a shuffle intensive Hadoop job there are two major bottlenecks in an HPC cluster:

1) **Limited storage space:** Traditionally the HPC cluster has a lower amount of storage space attached to compute nodes as compared to that in the dedicated I/O servers. This may be a bottleneck for shuffle intensive Hadoop jobs

2) **Limited sequential I/O throughput causes I/O wait:** Because of lower number of DAS devices (normally HDD) the shuffle phase becomes I/O bound, resulting in a huge amount of I/O wait.

If GPFS can be configured properly to store this temporary shuffled data, then a shuffle intensive Hadoop job (such as, our Hadoop-based de Bruijn graph construction which produces more than 6 times shuffled data than the original input) can take advantage of the huge storage space and the optimized I/O throughput. ***We configured the Spectrum Scale file system to utilize*** <u>***one directory for each*** **compute node, with 6 shuffle directories within each for that node's shuffle data.**</u> In both of the cases, we used the default round robin scheduling of YARN to spread the temporary shuffle data across multiple directories.

What Next?

Based on these results, LSU joined the OpenPOWER Foundation. This enables IBM and LSU-CCT staff to have deeper technical collaboration and provide unique science innovations using even more technologies from IBM and members of the OpenPOWER Foundation. The 2nd phase of the Genome analysis is being refined through the use of Key-Value-Store NOSQL structures and analysis. This analysis will take advantage of the patented IBM CAPI® interface (Coherent Accelerator Processor Interface) attached to an IBM Flash Array system. This CAPI interface and the IBM Flash storage array currently will provide the POWER8 processors with 40TB of main memory in a single array chassis. Preliminary results are already very promising with a 20TB Processor memory CAPI® extension.

Article authors and contributors:

LSU: Seung-Jong, Jay Park, Ph.D.
Ram Ramanajam, PhD. LSU CCT Director.
Gus Kousoulas, PhD.
Arghya Kusum Das, PhD Student. Author
Richard Platania, PhD Student, Author
Sayan Goswami, PhD Student, Author

IBM: Frank Lee, PhD., IBM Life Sciences Industry Leader, SDFS.
Ravi Arimilli, IBM Fellow, IBM Research
Terry Leatherland, Systems Architect, IBM Systems Group
Joanna Wong, PhD. IBM Software Defined Infrastructure Architect.
John Simpson, IBM Power Systems Technical Computing Team lead, IBM Customer Center.
Grace Liu, IBM Client Technical Specialist
JinChun Wang, IBM Client Technical Specialist