Scaling Applications on High Performance Computing Clusters and Analysis of the System Performance

Rusif Eyvazli[1], Sabrina Mikayilova[2] , Mike Mikailov[3] , Stuart Barkley[4]
FDA/CDRH/OSEL/DIDSR
US Food and Drug Administration, Silver Spring, MD, USA
{[1]Rusif.Eyvazli, [2]Sabrina.Mikayilova , [3]Mike.Mikailov, [4]Stuart.Barkley}@fda.hhs.gov

ABSTRACT

FDA/CDRH High-Performance Computing (HPC) team provides training and expert consultations to the HPC users in migration and scaling scientific applications on the HPC clusters. These applications include large-scale modeling and simulations, genomic analysis, computational physics and chemistry, molecular and fluid dynamics, and others which overwhelm even the most powerful scientific computers and workstations. Software scaling techniques for simulation programs written in C/C++ and Java programming languages are presented in this work. Python programming language also was used to do performance analysis.

Application Scaling Techniques

We use *array job* facility of the cluster job schedulers for scaling applications across the computing nodes of the clusters and adapt L'Ecuyer's RngStream package [1] to provide a quality Random Number Generator (RNG) for the massively parallel computations. Simulation iterations of an application are divided into subsets of iterations which are delegated as array job tasks to computing nodes of the clusters. Since tasks are independent the array job starts as soon as resources are available even for a single task. This avoids *job starvation* problem encountered when using the most dominant software parallelization technique – Message Passing Interface.

The quality of simulation applications strongly depends on the quality of employed random numbers across tasks running independently on different computing nodes. Studies show that traditional RNGs are not adequate for parallel simulations. L'Ecuyer's RngStream package provides 2^64 independent random number streams, each with the period of the 2^127. A unique task ID of a task is used to compute the unique stream for the task.

Why Array jobs?

Traditional software parallelization (scaling) techniques include Multi-threading, OpenMP, Message Passing Interface (MPI), Single loop parallelization, Scientific workflows, MapReduce, Spark, and Hadoop. The table below shows the advantages and disadvantages of these techniques.

Array job based techniques combine advantages of the traditional techniques. The disadvantage of the array job based techniques is the introduction of set up and convergence phases. These overheads are insignificant compared to those associated with traditional techniques. Some specific advantages include:

(a) Natural checkpointing: Every task of an array job is independent and system failures affect only subsets of the tasks.

(b) Automated identification of incomplete partial result files by counting and sorting numbers of lines in the partial result files.

(c) Automated identification of missing partial result files by comparing the list of expected partial result file names with the actual ones;

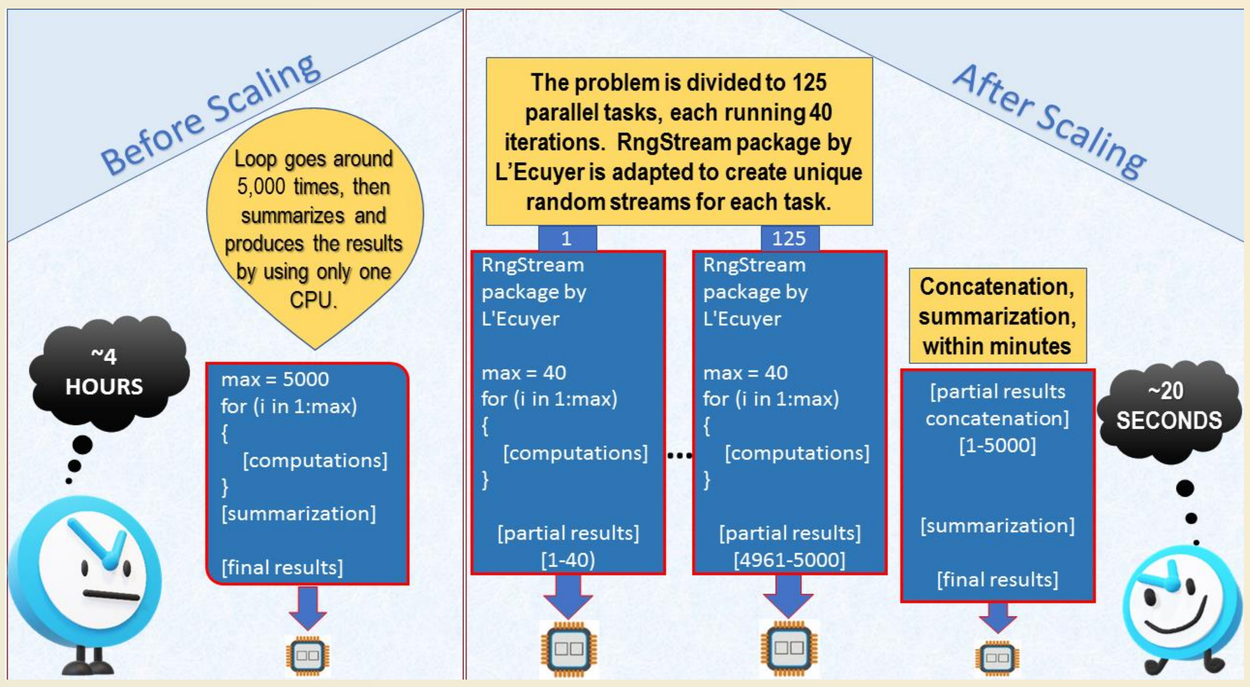
(d) Rerunning only the failed tasks.

Scaling Techniques Comparison

Scaling Technique	Advantages	Disadvantages
Multi-threading, OpenMP	More than one execution threads working in parallel within a node.	• Scaling is limited to cores on one computing node.
MPI	More than one execution threads working in parallel on more than one node.	• Increased likelihood of I/O and load balancing problems. • In practice, all computational resources requested by all parallel threads must be available for an MPI application to start. This may lead to job starvation. • No checkpointing.
Single loop parallel	All of the above.	• Does not parallelize multilevel nested loops.
Scientific workflows, MapReduce, Spark, Hadoop	Series of computational or data manipulation tasks run on more than one node in scalable manner.	• Incomplete approach for Modelling and Simulation (M&S)applications scaling and parallelization. • Not integrated with Son of Grid Engine and similar widely used job schedulers.
Array job	**All of the above**	**• Setup and convergence phase**

Table 1: Loop unrolling across HPC using array job technique [2]

Scaling Simulation Loops



System Performance Plots

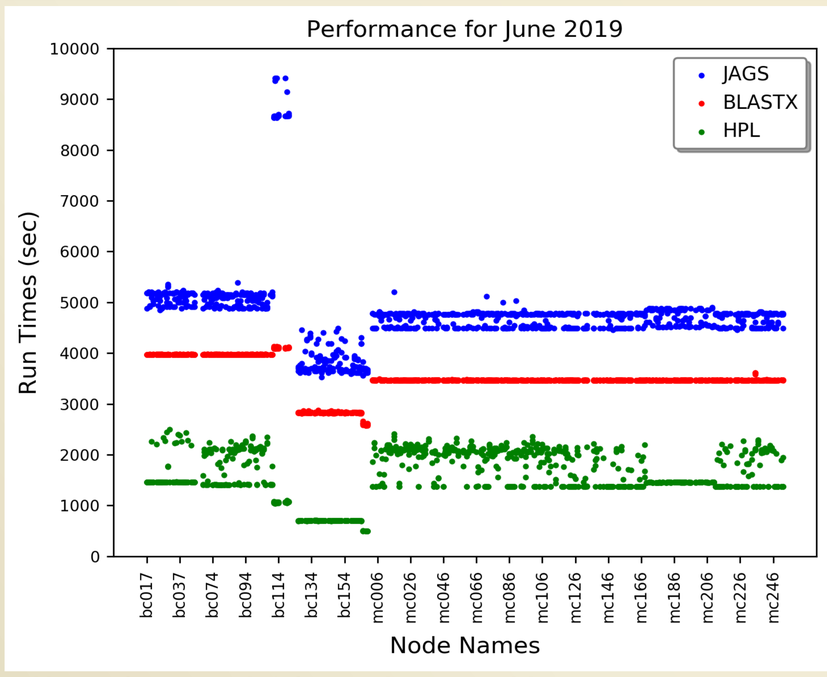


Table 2: Performance analysis for HPC cluster system [3]

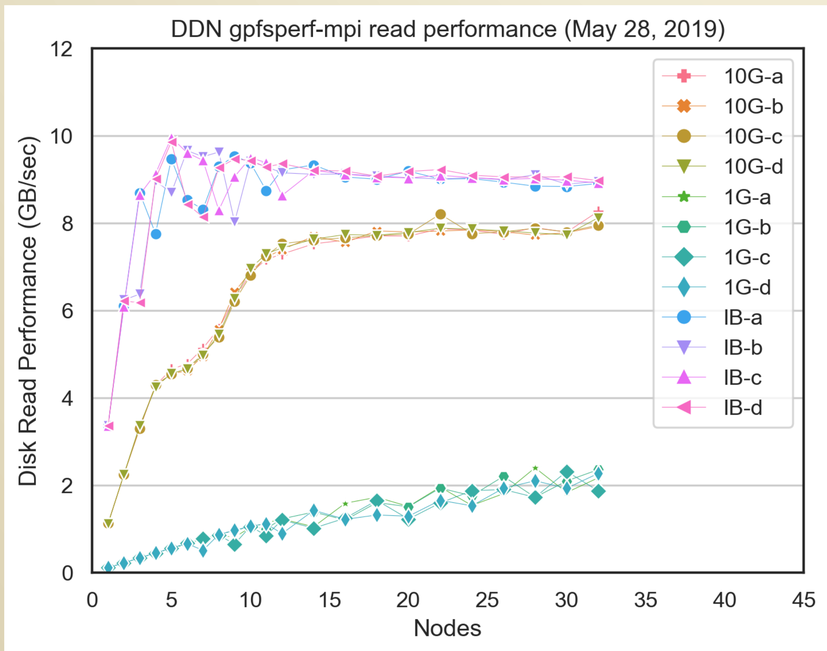


Table 3: Disk read performance analysis for DataDirect Network storage used by FDA [4]

Computing Nodes Performance Analysis

A Python program is created to help with analysis of the system performance. An input file named hostnum.csv contains the node names with their unique IDs, and the other three CSV input data files contain the system performance logs accumulated while running test application programs on the Betsy cluster. Based on these files, the Python program produces plots which show bc111-bc120 do better on the BLASTX and HPL problems but performs significantly worse on JAGS problem for June 2019. Likewise, the other nodes with differences are bc121-168. We have known that different applications behave differently, but this plot shows a very impressive example that these problems are different. The plots can be zoomed in to discover more detail information about system performance.

Disk Performance Analysis

The Python program was further enhanced to process other sets of data files for measuring DataDirect Storage read/write performance. The challenge in this task was to read the data from the input files in which the records were mixed with the notes by the data producer (application). To be able to read the data and get the resulted graph, the data were first filtered out by first two and last columns, then important three columns derived from the resulted data frame and plotted using Seaborn Python package

Conclusion

The run times of the applications are drastically reduced after using the techniques based on the array job facility of the job schedulers.

Application Performance Measurement graphs help the HPC team for preventive maintenance and system capacity management.

Network Storage Performance plots help the HPC team for further analysis of the performance of InfiniBand, 10Gb, and 1Gb ethernet with an increasing number of node counts. In this case, 10Gb performance levels off at about 10 and 8 Gb/sec. The 1Gb performance continues with a constant linear improvement with additional nodes.

Future Work

Research is needed to automate steps in setting up and convergence phases of the array job based scaling techniques. Experimental and theoretical works need to be conducted to estimate real speed up compared to ideal linear speedup.

References

[1] Reference for L'Ecuyer et al: https://cran.r-project.org/web/views/HighPerformanceComputing.html

[2] Table 1: Loop unrolling across HPC using array job technique: https://scl-wiki.fda.gov/wiki/images/6/6f/MS_Tasks_Parallelization-V2_FINAL.pdf

[3] Sample Python project for Table #2 can be found at: https://rusife.github.io/Performance-analysis-for-HPC-cluster-system-/

[4] Sample Python project for Table #3 can be found at: https://rusife.github.io/Performance-Analysis-for-Network-Storage/

Acknowledgements

Authors appreciate great support of FDA/CDRH/OSEL/DIDSR management, mentors, Dr. Mike Mikailov, Stuart Barkley, and all HPC team members, including Fu-Jyh Luo.