ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #27890

VM High-Performance Computing for Undergraduate Engineering Projects

Forrest Mobley, Embry - Riddle Aeronautical University

I am a junior level aerospace engineering student who has a passion for research and computational simulations. My goal is to develop the skills I need as an engineer to improve society through the advancement of aerospace technologies and understanding. Something that I have a particular interest in is developing a system for aerial refueling for unmanned aerial vehicles, particularly for search and rescue operations. I enjoy cycling, backpacking, and computer gaming.

Dr. Shigeo Hayashibara, Embry-Riddle Aeronautical University, Prescott

Associate Professor, Department of Aerospace Engineering, College of Engineering

VM High-Performance Computing for Undergraduate Engineering Projects

INTRODUCTION

Parallelized processing, or the process of solving multiple parts of a single problem simultaneously through the use of many processors (see Fig. 1), is essential for many engineering and scientific disciplines as projects and mathematical models continue to reach beyond the scope of what can be done by hand. From their inception in the 1960's, these grid computing (or supercomputing) systems have advanced so as to come in a variety of sizes and are maintained by a variety of entities. Governments have built the largest and most powerful supercomputers in the world, particularly the U.S., which has produced 5 of the top 10 supercomputers[2]. These massive systems not only help scientists around the world solve increasingly challenging problems, but also provide for the national security of their owners. Universities are also common developers of these machines, for both academic and national research. Academic supercomputers, although not as powerful as most government systems, can also perform at very high speeds and accomplish many large jobs.

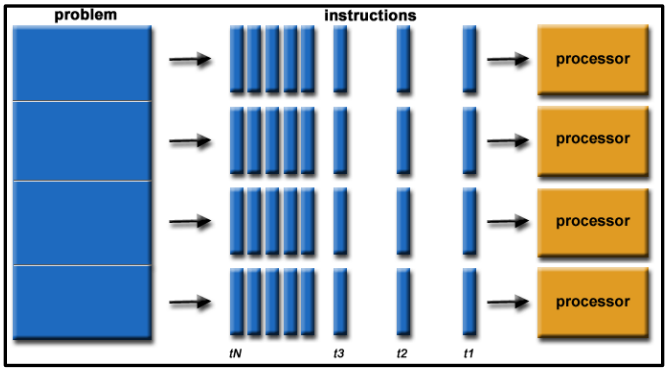

Figure 1: Parallel Processing[1]

As useful as these systems are, they generally come with high costs. Summit, the most powerful supercomputer in the world, cost the U.S. Department of Energy more than 200 million dollars[3]. China's most powerful supercomputer and third-most powerful in the world, the Sunway TaihuLight system, was built for around 270 million dollars[4]. Both of these systems regularly consume in excess of 10MW of power[3,4]. Regardless of their size and power, supercomputers are not cheap. Their high-performance computer parts, along with high storage space and upkeep costs, quickly add up and become a large undertaking. Even the smallest of what would be termed a "supercomputer" can easily cost hundreds of thousands to millions of dollars to obtain, and smaller entities, such as small companies or undergraduate-level research labs, generally do not have the resources to purchase such equipment if needed.

For these smaller entities, there are two options available to gain access to large parallel processing systems: renting computer time through various companies or building their own system. Renting supercomputer time can be quite cheap and is a great option for companies who need simulations or computing jobs done quickly. This solution does not generally work for academic settings, however, due to the number of jobs that must potentially be done; building a system is more useful and economical in the long run. This option can present some challenges but can be accomplished readily in an academic setting. To contribute to the available options for building a system for academia, this paper outlines a parallel processing system build based on virtual machines (VM).

PARALLEL GRID COMPUTING SETUP

Parallel grid computing systems can vary in terms of hardware and software, but generally have the same layout: a “head node” which operates as the center of operations for the system; “slave nodes” that compute what the head node distributes to them; and a network to connect the head node to the slave nodes. The slave nodes are generally just copies of each other, both in terms of hardware and software; however, hardware can vary as long as software remains constant. Several software components are required for parallelized computing, such as Secure Shell (SSH), Network File Sharing (NFS), and some form of Message Passing Interface (MPI). SSH allows for the head node and other nodes to access directories stores on a particular node without the use of a password; this is especially useful for parallel processing as access can be required hundreds of times per job. NFS creates a directory dedicated for each job, which can be shared over the network with each node (this requires SSH as well). MPI is a model set up to allow communication between nodes to happen smoothly and efficiently while a job is being run. All of these work together, along with the parallelized software running the job, to increase computing speed exponentially.

The system detailed in this paper is centered on VM technology. This focus allows the system to become part of a set of network-attached computers that are already set up and running. Virtual machines can run in the background and are easily customizable to use different levels of power based on desire; thus, the computers that this system is a part of are free to perform other tasks while a job is being performed (providing each virtual machine is not set to use all the resources of each node). In all, the system developed consists of 1 head node, which is a separate physical machine; 10 virtual machine nodes, which are housed in the 10 workstations currently set up in the Advanced Computing and Simulations Lab (ACSL) on the Embry – Riddle Aeronautical University (ERAU) Prescott campus (see Fig. 2); and one additional physical machine devoted to the cluster. This comes to 30 processors of varying specs, 104 Gb of RAM, and 10 Tb of storage space housed within the head node.

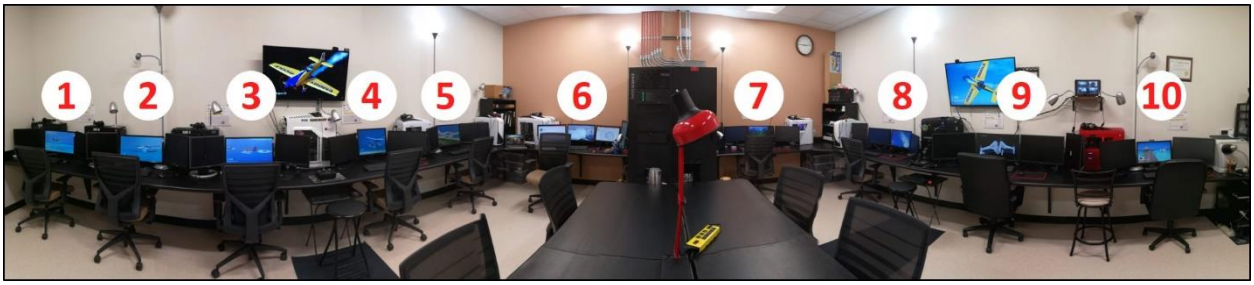

Figure 2: ACSL Virtual Machine Cluster

SYSTEM ECONOMICS

Due to the implementation of VM nodes, the economics of this cluster can vary; if host machines are not available for each VM, the cost of each node becomes much higher. Because this machine is purposed directly for undergraduate engineering projects, the assumption that a set of network-attached computers, such as a computer lab, is available for use is applied. Thus,

the cost of parts is completely made up of hard drives for VM housing; however, this is not required, but makes organization of the cluster easier for the builder. For a 10-node cluster, the total cost of hardware, with the previous assumption, can be as low as $230[5]. If a computer lab is not available for use and every node and the head must be built, the following table details the average costs[6] to build the system with comparable parts as those of the cluster tested in this paper.

Table 1: Itemized Cost for Sample 10-Node Cluster

Item	**Cost (Cost per unit)**	**Wattage Use**	**Description**
CPU	$1309 ($118.99)	715W (65W[7])	i3-8100 3.6 GHz
Motherboard	$732.60 ($66.60)	330W (~30W, depends on loading[8])	ASUS H110M-E Micro ATX
Memory	$604.89 ($54.99)	Negligible	G.SKILL Ripjaws V 8GB
Power Supply	$571.89 ($51.99)	NA	EVGA 500W 80+ Bronze
System Drive & VM Drive	$505.78 ($22.99)	~33W (~3W[9])	Kingston A400 2.5” 120GB SSD
Accessories (Keyboard, Mouse, Case, Monitor)	$1336.94 ($14.99, $39.99, $66.56)	~327W (~2.5W[8], 11.9W[10])	Logitech MK120 Wired USB Keyboard & Mouse, CoolerMaster MasterBox Q300L mATX Case, Acer V6 V196HQL Ab Black 18.5” Monitor
Total	$5061.10 ($437.11)	~1400W	

Although components prices will vary, as well as components themselves, the cluster components listed in Table 1 represent a very capable machine, consisting of 44 cores and 88GB of DDR4 2133MHz RAM. Components of a typical cluster that are missing from this table due to builder preference and the variety available are any sort of data storage beyond the system drive and a network to allow communication between nodes and the head machine. However, this build represents a comparatively very economic option for undergraduate-level research, especially considering how much power is consumed. As will be seen in the following section, this sort of cluster build could never hope to compete in terms of computing power with large supercomputers, but for small scale projects such as those common in an undergraduate setting, it is more than capable.

COMPUTING PERFORMANCE

This VM-grid computing machine was tested in a couple of ways. It was first benchmarked using the NAS Parallel Benchmarks (NPB) developed by NASA[11]. The NPB consist of various tests which are used to benchmark computing systems in several different ways. One of the tests, known as "Lower-Upper Gauss-Seidel solver", was used to benchmark the VM-grid system. This test involves solving a large system of nonlinear PDEs using a symmetric successive over-relaxation algorithm and is very similar to some computational fluid dynamics (CFD) algorithms. Using this benchmark, the cluster system achieved a speed of 21.8 GFLOPS, or 21.8 billion floating point operations per second. Comparatively, the single physical node, when benchmarked using the same test but only using 8 processors, performed about 4.7 GLOPS. These results show that through parallelization over several nodes, performance increased by over 4.5 times with only 3.75 times the number of processors.

The other method of testing performed with this system was through direct CFD simulations, using the open source software known as OpenFOAM. Two CFD simulations were set up for testing: a smaller 2D test with about 300,000 cells and a larger 3D test with about 7 million cells. Every other aspect of these tests was identical, with a 2-equation realizable k-ε turbulence model[12] and simpleFOAM solver[13]. As CFD is an iterative process, both of these simulations were run for a baseline 1000 iterations, disregarding the level of convergence. Time results of these tests, which consist of running the simulations with 1, 12, and 30 processors, are shown in the graph below. Times are shown in both execution time and clock time. Execution time is the time during which the processor(s) were performing floating point operations; clock time is the physical time (the time experienced by the user) the simulation took to complete. Clock time will always be larger than execution time due to other tasks needed to run the simulation, because of tasks such as reading or writing data and communications.

These time results show that, as expected, time elapsed generally decreased with increased number of processors. A single processor was able to finish the 2D simulation in about 30 minutes, which results in roughly 6 milliseconds per cell over the course of the simulation. The 12-processor run spanned the least amount of time at 16 minutes and only 3 milliseconds per cell. Using 30 processors for the 2D simulation resulted in poor performance, at about 27 minutes and around 5 milliseconds per cell of execution time. There are many factors that could be contributing to this decrease in performance like limits in processor speed, memory capabilities and network speeds. However, the main reason behind the slow down is the size of the simulation; a certain computation can only be parallelized so much. Particularly for CFD simulations, which are parallelized by dividing the computational mesh between each processor, serial communication can become an overwhelming factor[14]. Certain sizes of simulation require a certain number of processors for peak optimization (which is also limited by the level of optimization of the CFD code[15]). This is why there is a noticeable gap of about 5-8% between execution time and clock time for the parallelized runs.

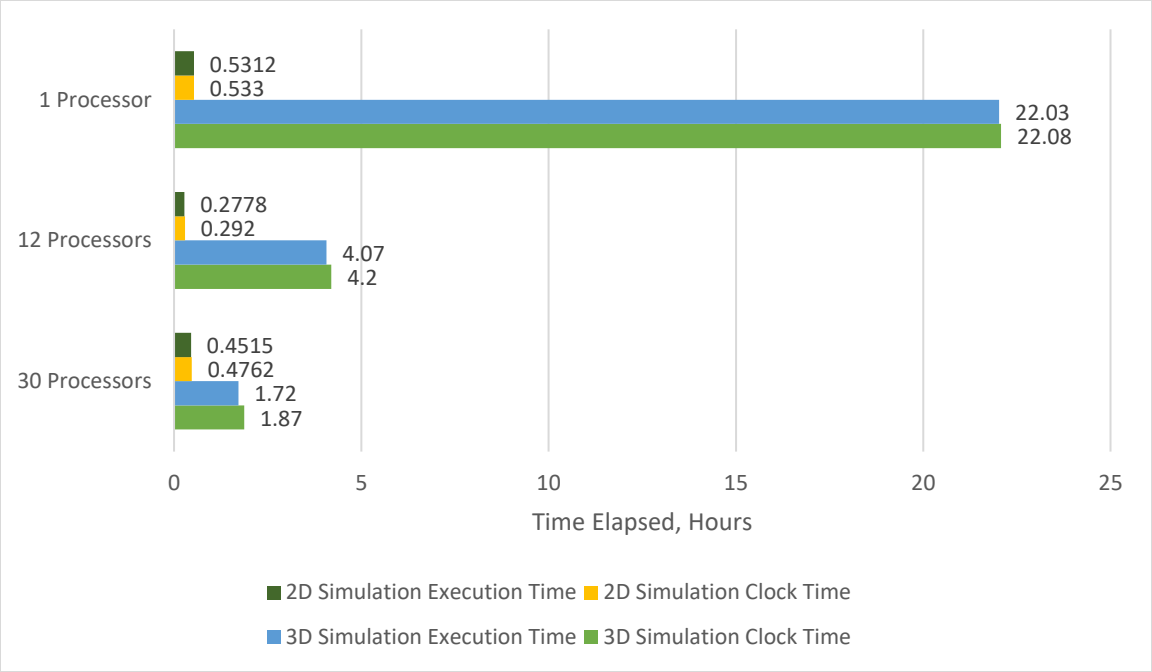

Figure 3: 2D and 3D Simulation Time Results

The perks of parallelization became very apparent in the time results for the 3D simulation. The single processor finished 1000 iterations for the 3D mesh in about a day or about 11 milliseconds per cell, while the 30-processor set up finished it in less than two hours and performing at 0.9 milliseconds per cell. As can be seen in Figure 1, the increase in speed does not linearly correlate with number of processors but is significant between the single and 30-processor runs. This non-linearity is due to the setup of this system, which depends on the standard ethernet network for intercommunication.

CLUSTER USAGE

As part of an ongoing research project at ERAU, this cluster was used to conduct CFD simulations of an aerial refueling system known as a paradrogue, which is used to refuel aircraft mid-flight. Due to the complexity of the flow within the wake field of this system, 3D simulations were all but impossible on a single machine. However, with the cluster built these simulations were accomplished in a reasonable amount of time. These simulations are very insightful on how the wake field is operating and how it affects the stability and drag characteristics of the paradrogue (see Fig. 4). This information can in turn help uncover many new things about CFD simulation algorithms, turbulent flow, and bluff body aerodynamics. As more simulations are needed, the VM cluster at ERAU will continue to be used in this project until greater computing power can be acquired or built.

One potential issue with the VM cluster is the human factor associated with it. As each workstation/node is still available for use by other users while simulations are running in the background, if a user were to unknowingly shut off or restart the node, the simulation would be interrupted. Although the option to power off a Windows machine can be disabled, it is always

necessary at some point to restart a system. Thus, stringent communication is needed between every user in the lab. This issue has caused several incidents but is not enough to cause concern. Human factors will always be the leading cause of problems in this sort of system due to its implementation; however, concise communication should alleviate any issues between users.

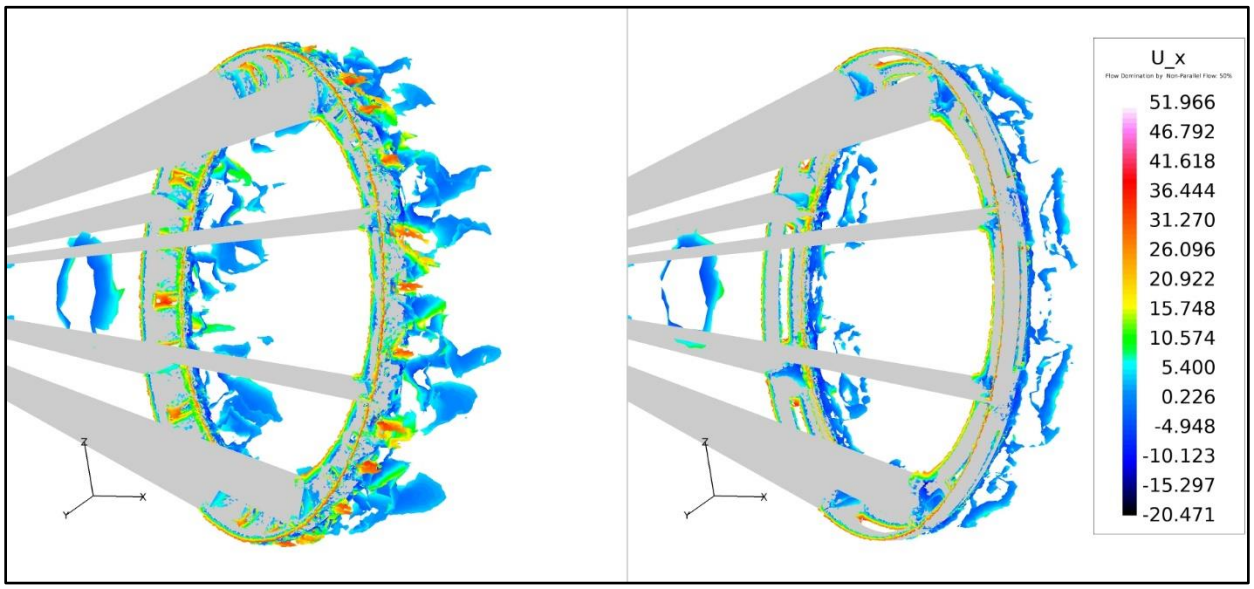

Figure 4: Paradrogue CFD Simulation Results

CONCLUSION

A grid computing system was developed using virtual machine technology to increase usability and economic practicability. The nodes of this system were built into each workstation computer housed in the ACSL at Embry – Riddle Aeronautical University and used portions of the resources available in each computer, thus allowing the grid computer to run in the background and each workstation to still be used while parallel processing occurred. Though not very powerful in comparison to commercial and government supercomputers, this system is extremely well suited for undergraduate level research due to its versatility and economic value.

BIBLIOGRAPHY

1: Barney, B. (2019, February 11). Introduction to Parallel Computing. Retrieved from Lawrence Livermore National Laboratory: https://computing.llnl.gov/tutorials/parallel_comp/

2: Dongarra, J., Strohmaier, E., Simon, H., & Meuer, M. (2019, January 14). *November 2018*. Retrieved from Top500: https://www.top500.org/lists/2018/11/

3: U.S. Department of Energy. (2017, May 23). *FY 2018 Congressional Budget Justification.* Retrieved from Energy.gov: https://science.energy.gov/~/media/budget/pdf/sc-budget-request-to-congress/fy-2018/FY_2018_SC_ASCR_Cong_Budget.pdf

4: Dongarra, J. (2016, June 24). Report on the Sunway TaihuLight System. Retrieved from netlib.org: http://www.netlib.org/utk/people/JackDongarra/PAPERS/sunway-report-2016.pdf

5: newegg.com. (2019, February 12). Kingston A400 2.5" 120GB SATA III TLC Internal Solid State Drive. Retrieved from newegg.com: https://www.newegg.com/Product/Product.aspx?item=N82E16820242399

6: newegg.com. (2019, February 12). Retrieved from newegg.com: https://www.newegg.com/

7: Intel. (2019, February 13). INTEL CORE i3-8100 PROCESSOR. Retrieved from Intel: https://www.intel.com/content/www/us/en/products/processors/core/i3-processors/i3-8100.html

8: buildcomputers.net. (2019, February 13). Power Consumption of PC Components in Watts. Retrieved from buildcomputers.net: http://www.buildcomputers.net/power-consumption-of-pc-components.html

9: Kingston. (2019, February 13). A400 SSD. Retrieved from Kingston: https://www.kingston.com/datasheets/SA400S37_us.pdf

10: Acer. (2019, February 13). *V6*. Retrieved from Acer: https://www.acer.com/ac/en/US/content/professional-model/UM.XV6AA.A01

11: NASA. (2016, March 21). *NAS Parallel Benchmarks*. Retrieved from NASA Advanced Supercomputing Division: https://www.nas.nasa.gov/publications/npb.html

12: OpenCFD, Ltd. (2018, September 10). Realizable k-epsilon. Retrieved from OpenFOAM: The open source CFD toolbox: https://www.openfoam.com/documentation/cpp-guide/html/guide-turbulence-ras-k-epsilon.html

13: OpenCFD, Ltd. (2018, September 10). simpleFoam. Retrieved from OpenCFD: The open source CFD toolbox: https://www.openfoam.com/documentation/cpp-guide/html/guide-applications-solvers-incompressible-simpleFoam.html

14: Keough, S. (2014). Optimising the Parallelisation of OpenFOAM Simulations. Fishermens Bend, Victoria: Australian Government Department of Defense: Defence Science and Technology Organisation

15: Cummings, M, R., Mason, W. H., McDaniel, D. R., & Morton, S. A. (2015). Applied Computational Aerodynamics: A Modern Engineering Approach. New York: Cambridge University Press