Senior Project I: Attempted Processing of 4K RAW Cinema Data Files in Realtime Via Cluster Computing and Parallel Processing: Interim Report

Jordan Westhoff, RIT 2015

November 15th, 2014

Abstract

This document will outline basic and advanced concepts as they stand currently, as well as technology, budget and intellectual property components that will be contained and exhibited by the finished product. Inherent to the goals of this project, it is a paramount goal to generate a system to benefit the motion picture industry on a global scale, as well as RIT's School of Film Animation free of prohibitive technology limits and prohibitive licensing restrictions.

This paper serves as the first half of a comprehensive, final document to be typeset and presented upon completion of the thesis project in the Spring of 2015.

0.1 Introduction

As the age of cinema has progressed, technology has become more advanced, initially starting only in the field of film and developing, while spreading to the digital domain. As with Moore's law, as time passes, technology becomes increasingly more powerful by a multiplicative factor. This is certainly evident in a host of imaging systems, which have grown exponentially more complex and feature rich since the dawn of digital imaging systems. Today, systems exist that promise images that are both incredibly large and also exceptionally detailed. Images are now being recorded in 2K, 3K, 4K and 6K resolutions in RAW formats, which has proven to be exceeding the processing power of the workstations designated to process them.

In recent years, digital cinema imaging equipment has evolved with leaps and bounds, while the post-production systems required to process these images has remained largely the same. Although processing and GPU power has continued to climb, also by Moore's law, it is still a major process to import and process the complex images that are generated, often in proprietary format, by each major motion picture camera manufacturer. As a result of the variance of each of these formats, different post production workflows are required based on the nuances of the RAW image file sequences and the types of encoding used by each of these specific data cameras.

Processing these workflows has always added an additional level of complexity to the post production processes of the companies and directors that choose to implement them. The conversions from truly RAW encoded formats to viewable and editable formats in the correct color spaces is a process that has plagued those that have adapted RAW from the very beginning. For consumers and independent filmmakers, this process is often crippling both in complexity and cost, with an advanced knowledge of the technologies at hand, required.

This project attempts to tackle multiple technologies simultaneously with beneficial implications for both independent and low-budget filmmakers. The underlying goal of the project aims to provide sound image processing from an imaging science perspective while providing an elegant, scalable solution that is always welcoming of the addition of new hardware. This combination builds the rigid spine of the project and aims to make it marketable for enterprise adaptation although not fully open source (intellectual property retained).

This document is a comprehensive look at the semester project process made in pursuit of completion of the project to the previously agreed terms. Contained within this document will be an exhaustive look at any progress occurring since the project's initial proposal. A detailed look will be made both at hardware and software progress, as well as the systems administration and the units that have been adapted to the project scope as it has grown. Each of these portions will dominate a section of the document since each garners a different focus. De-

tailed specifications and statistics will also be included to benefit the observer and committee. A final section has been added to address the professional branding and any and all individuals that have helped to further the progress of the project. As a result of professional interest in the project, some aesthetic considerations have been made to the project and those will be brought to light at that time.

In the interest in brevity and conciseness, the previous proposal document is available for reference at a provided link.

0.2 Background

Currently, processing RAW cinema footage is a computational nightmare. Single editing systems are powerful, but not powerful enough to edit and process the mass quantities of 4K footage that are generated in a daily shooting environment. While these files can be split up across multiple dedicated workstations, this is cumbersome, requires manual labor for every instance and is inefficient. The focus of this system is to engineer a single, comprehensive system that will ingest and process 4K cinema footage in its RAW state while generating a complete, processed 4K image and an HD video proxy for editing proxies. This allows the system to be intensely valuable to anybody using a modern editing workflow by removing the longwinded process that is preparing 4K cinema files.

In order to do this, and to be advantageous to speed and processing, a cluster of computing nodes has been designed. Common 4K footage process systems rely on the power of a single GPU (graphics processing unit) to process the 4K RAW frames in a linear manner. In order to improve speed and efficiency, processing can be done in parallel across a series of computing nodes, which all serve to process the information like a single commodity GPU.

Systems like this currently do not exist, as they are experimental and require advanced software and developmental programming, often of an open source nature. This software will manage every step of the process and will be engineered in house as the major component of this thesis project. More detail oriented content regarding this code is discussed later in the document. The systems used to run this software are enterprise level blade servers, all of which fall below current tech-cut-line. This is important because it illustrates that parallel processing is advantageous to processing, even on older hardware, where individual node speed can be offset by an increase of the aggregate volume of nodes.

The act of processing 4K files is a wide net, dependent on a great number of

things. All of these aspects define the project and the steps that will have to occur to each image before they can be correctly labeled as completed. Due to the massive amount of fragmentation within the corporate and professional cinema realm, the project has been narrowed to only cater to the 4K files generated by the Sony FS line of cameras. This allows the project to be developed without an extensive dev team and with a focus on correct imaging science practices over corporate marketability.

The goal of the project is for the final, engineered system to accept 4K files, process them and generate final files and proxies in realtime, or as close to it as is possible with the equipment at hand. Each of these aspects will also be addressed deeper in the paper and in relevant sections. In order for the successful completion and academic release to be achieved with the project, a system must be engineered that processes 4K files in an accurate manner and with the required presets by the user. Achieving realtime processing speeds is a bonus, but not to be pursued in the sacrifice of proper image processing or output accuracy. If realtime speeds are not able to be achieved, computation will be run to estimate, based on the current operating potential of the system, how much additional processing would be required to process in realtime based on performance statistics of the current cluster.

0.3 Summer Term Progress

As a result of a summer co-operative opportunity at RIT, significant progress was made over the summer in pursuit of furthering the progress of the project. This progress can mostly be classified as refreshed learning, as well as hardware acquisition and provisioning. In addition, a significant portion of research was accomplished, which helped to cement the decisions for the chosen Operating System, as well as the provisioning of the units.

0.3.1 Acquisition

Through several recycling programs at RIT, in addition to the good will of several departments (primarily Research Computing) - a significant portion of excess hardware was obtained, free of charge under educational application only. Two primary units in the stack, denoted as two Primary Compute Nodes were acquired. These two units formed the spine of summer progress, as they allowed for learning and sandbox testing to occur in a professional enterprise deployment atmosphere. The units, both products from SuperMicro, pack AMD Opteron 2384 Processors (8 cores, single thread) and 128GB of RAM in a 64-bit deployment architecture.

In addition to physical hardware acquisition, a significant amount of software was acquired - most notably the idea to use SLURM to manage the cluster (SLURM will be elaborated and properly explained later in this document). Existing and working in a research environment that also actively manages a cluster computing environment was intensely valuable and aided the acquisition of many ideas, as well as hardware.

0.3.2 Research

Off the record research was a valuable look a wide variety of aspects of the computing world. Downtime over the summer was in abundance, serving as an excellent opportunity to conduct research into servers and their enterprise implementations, as well as software and learning the common skills of a cluster systems administrator.

This research allowed for a comprehensive look at current, industry favorite, operating systems in the Linux sphere, and allowed testing for each. With each of these came a look at specific package managers, compatibility and ease of use with kickstart implementation, as well as other dependency compatibility.

While the project was not officially started until the fall semester, advance research allowed for valuable research and consultation to take place, in addition to becoming with potential pitfalls and difficulties while allowing the scope

of the project to be concretely determined.

UltraGrid and 4K

Work contained within the co-op was also relevant and helped build and maintain necessary skills, as well as introduction to new relevant material and ideas as well. One of the main focuses of the summer work was the application and development of UltraGrid - an open source application to stream and transport 4K uncompressed and compressed data streams over IP networks in a variety of ways. This was of express interest since 4K video files are relevant to this thesis project as well. One of the most valuable assets of working with UltraGrid was the attached networking skills that were gained, as well as focusing on the differences and advantages of CPU vs. GPU computation. UltraGrid focuses on its most efficient method of compression by taking advantage of NVidia's CUDA technology to compress high bandwidth streams into a MotionJPEG format. Working with this was excellent to compare against current DXT and other CPU intensive compression schemes.

0.3.3 Skills and Practices

The summer evaluation period also allowed for time to be dedicated to refreshing personal skills and memory of necessary components of the project. Most notably, these involved some skills that an advanced admin would find basic, yet are integral to the success of the project.

Networking

Networking was one of the most foremost aspects of computing that needed to be reviewed and the summer allowed for testing of networks as well as dealing with hosts, advanced manual networks and fighting with common services such as DHCP and DNS. RIT has an interesting network, especially when it comes to device registrations and getting familiar with all of the inner workings available to a student took some time as well.

Terminal + Shell

In addition to networking, familiarity with the linux terminal and the programming linux scripting language, known as Bash (or shell), also needed to be revived. Scripting in shell allows a great deal of system tasks to be automated, while also allowing complex programs and operations to be executed as well. Shell is the icebreaker back into traditional programming, helping to prepare the way for more advanced, upper level programming languages to be utilized later in the project.

Not every machine in the project utilizes a GUI, or graphic desktop interface, for the user to supply commands and interact with the machine. This can all be handled through Terminal, which comes native on all Linux machines, Mac

OSX UNIX machines and Windows machines through the addition of third-party software such as Putty or Cygwin. Commands can be executed and each machine can be operated fully through the terminal, thus allowing a stack of servers to be configured in a 'headless' manner - in such a way that no keyboard, mouse or display is required for each machine. All of the machines can be controlled remotely, even from different rooms, buildings or cities, through a terminal instance once properly configured. This is a massive performance gain for using Linux machines, as once they are powered on, properly provisioned and connected and registered to the network - they never need to be hooked to a monitor or keyboard again, save for instances requiring individualized maintenance.

0.3.4 Project Management

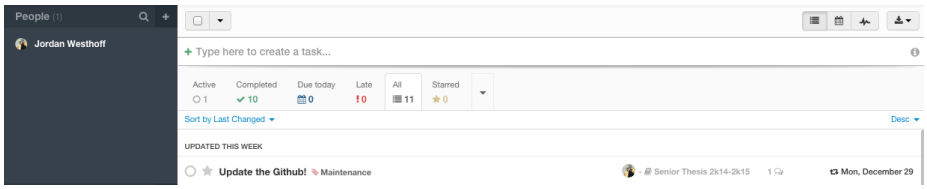
Time over the summer also aided in getting some idea of project management in order. While a dedicated software suite for project management was left undecided until a given time in the Fall semester, the projects timeline was drawn up late in the summer. Generating timelines this early in the project allowed for all considerations to be made in a rough manner while still allowing flex blocks to be inserted in case of OS changes, or other project deviations without causing total destruction of the timeline and cause to build a new one. Goals, milestones and points of interest were possible to determine ahead of time once initial summer research was out of the way which provided this project with a clear advantage in a temporal sense over other projects which were left to preconceive and determine project management techniques during the Fall semester.

0.4 Fall Semester

Fall semester, as it has drawn to a close, was a period of intense continued research and the implementation of the overarching systems administration required by a project of this depth and magnitude. All of the preliminary research has been updated to status completed, as has the vast majority of the cluster building, although continued expansion is still a progressive and continuous phase. The vast majority of software, hardware and the methods to bind them together have been acquired and compiled and the project is still on schedule. Project management has been an integral component of keeping the project on track and will also be discussed as well in an effort to maintain comprehensive coverage of non-proprietary information of the project.

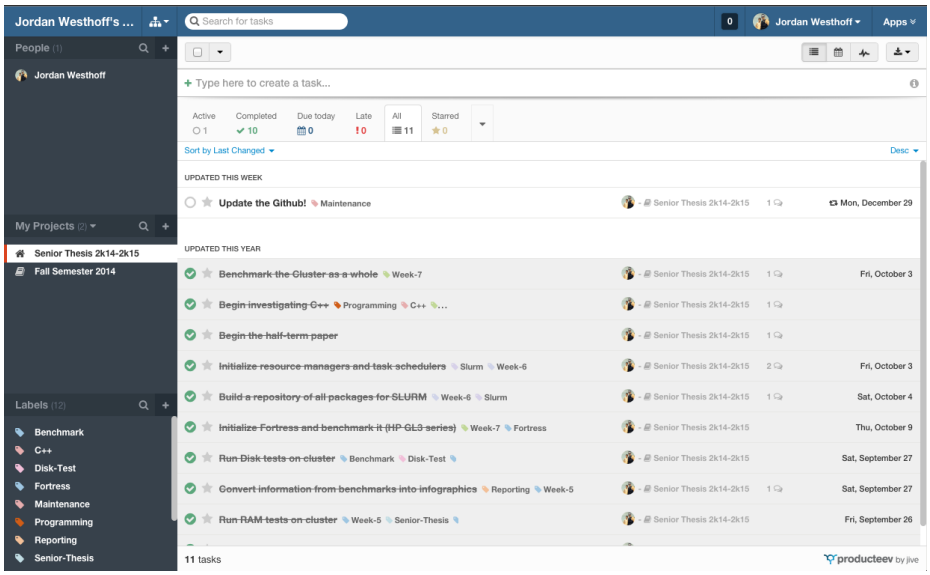
0.4.1 Project Management - Producteev

Managing the project is a significant workload of its own right. Between gathering all of the tasks to be completed and managing all of the time constraints that they require is the most significant facet of project management. After looking through a wide variety of products and services, a free service known as Producteev was implemented, with great success. Producteev spans desktop operating systems, Apple iOS and Android which means it is always accessible regardless of where work is being accomplished. Producteev allows entire projects to be defined within the dashboard, in addition to sub-projects and tasks within those projects.



Producteev's Standard Task Entry Dashboard

All of these can be sourced to different people, or in this case, all tagged to myself. Due dates and notifications as well as intelligent tagging and searching are implemented as well which makes managing tasks, meetings and upcoming deadlines a simple procedure. Producteev is intended to be scalable from single use cases up to enterprise collections of workers. Producteev was also chosen to be implemented here due to its free use case for educational use. This means that all of the features of Producteev are free for use based on the terms of the project and the scale of its implementation - less than ten users and less than 10 open projects. This means that potential collaboration can be opened up between multiple parties and advisors, should it be necessary.



Producteev's Full Task Listing Environment [Example]

Since pre-defined Gandt charts and expected productivity timelines were generated, entering them into Producteev was simple. In addition to this, daily update emails and weekly recap reports were generated by Producteev to help monitor achieved progress and tasks left to do on a daily and weekly timeframe. Producteev was also used to manage co-existing classes and assignments outstanding which made is an excellent, holistic tool for the project.

0.5 Full Timelines - Fall Semester

The following is a complete week by week synopsis of all of the work completed on the thesis during the fall term. Any unfamiliar terms will be discussed forthcoming in their proper sections. All of the information is drawn from internal documentation based on its inclusion into the thesis project.

0.5.1 Week 1

The first week of the semester relied on several small tasks. These included appraising the progress made over the summer and then determining its results and impact on the project. This week was spent mostly moving equipment from my previous labs into new labs as well as reconfiguring current networking. In addition, the week was spent determining goals for the semester and loose ideals as to what acceptable time frames were for deadlines.

0.5.2 Week 2

Week 2 was similar to Week 1 - little physical progress was made but a lot of required ground work and pre-production was conducted and established. Of this, project management software was chosen, and its implementation was started. Producteev was chosen as the primary software package and it has served well over the course of the semester. Producteev allows for desktop and mobile connection, with notifications and time tables. Producteev also features the ability to collaborate project requirements amongst other registered members, all of which is private and unique to each project as set by the user. Producteev also provides the ability to use other tools such as Gandt charts and other organizational tools. Senior Forum discussed the presentations required and I chose the first slot so preperation for that started as well.

0.5.3 Week 3

Week 3 was spent mostly preparing the presentation to give to the MPS senior class. In addition, serveral new systems were added to the stack and had to be configured. In preperation of these nodes, work on a basic benchmarking software set was begun.

0.5.4 Week 4

This week was spent finishing the presentation as well as polishing the initial benchmarking scripts program. This tested CPU speed and disk speed and then copied the results to a server to be logged. It wasn't elegant but it gave some initial idea of performance although it will be replaced later for an elegant solution. This week I also set up a GitHUb repository for the project which now houses all of the code, commits and in future weeks, a wiki page.

0.5.5 Week 5

This week was comprised entirely of memory (RAM) testing, disk testing and some basic infographic generation. In addition to this, with a repo made for the project code, another was made to house SLURM and all of its dependencies. This was not completed this week.

0.5.6 Week 6

This week was an entire comprehensive look at resource managers, SLURM was re-examined, Maui+Torque and other solutions like GEARS, OpenPBS and ROCKS Cluster were examined as well. SLURM won out, and was decided to be the continued choice for cluster resource management. In addition, a new server, Fortress (HP GL3 series) was added. The server is underpowered compared to the rest of the cluster but will still be useful for other aspects of the project. In addition, the primary NAS server was disk tested as well and its results logged.

0.5.7 Week 7

Given inbound midterms, this was an easier week that revolved mostly around investigating C++, learning the basics of compiling code and benchmarking the entire cluster as a whole. Also, passwordless SSH was installed and configured so each node can speak with every other node in the cluster without constant password authentication.

0.5.8 Week 8

The comprehensive thesis paper was started, chosen to cover the architecture, the available benchmarking data as well as the design of the current systems. The github wiki page was also created and this took some time to populate with info (timelines, front page, etc).

0.5.9 Week 9

Continued work on the paper. In addition, common C++ image processing tools are now being tested, these include compiling and running open.CV, dc.RAW and cimg. Lots of testing and trying things, not a lot to write about.

0.5.10 Week 10

This week it was discovered that the distribution of CentOS being used on the head node was not x64 bit but rather the deprecated 32-bit i686 architecture. The week was spent upgrading the head node and testing the full install scripts to ensure easier provisioning. In addition, PTS (Phoronix Testing Suite) was now implemented to replace my older, less elegant home brewed solutions. I also spent a great deal of time installing the PTS Phoromatic server and the web dashboard.

0.5.11 Week 11

More C++ was worked on this week. In addition, PTS has an integrated dc.RAW test profile so I began using that. This operation uses a really basic TIFF $->$ PNG operation though which isn't what I'll be using dc.RAW for so I also began working on a custom benchmark through PTS that uses dc.RAW. Thanks to the Research Computing office I was able to begin adding satellite nodes for testing and comparing enterprise vs desktop hardware.

0.6 Week 12

Five new intruder nodes acquired thanks to the good will of Emelio from the computer engineering department. These units were registered and brought on the network after much stress (thanks RIT RESNET). Cent initialized, updated, provisioned and the benchmarked Phoromatic server was set up to run with each

node. Image processing setup and custom dc.RAW testing built, just needs to be implemented once SLURM is stable. One of the largest compute nodes is also running old i686 architecture - Crawler had to be converted to x64 arch for better RAM and CPU utilization. In addition to this all units were benchmarked with 32-bit and 64-bit performance comparisons.

0.6.1 Week 13

Week 13 brought the implementation of Ansible which is an open-source large scale provisioning tool based on SSH. Ansible was deemed worthwhile and after a full week of testing is now replacing all of the original initialization scripts because Ansible can run operations on the entire cluster at once (like a yum update for instance) and report with detailed information. Ansible saves massive time in a variety of tasks and is now integral to the administration of the cluster.

0.6.2 Week 14

This week Zabbix was installed to provide constant monitoring of every node in the cluster. Zabbix exists based on a server -¿ agent SSH handshake. After a couple of days the server was configured, all of the nodes and all of the satellite nodes were configured as well. This provides around the clock monitoring and alerts and updates if cluster nodes go down or overheat (among 80 other markers and statistics that can be recorded or monitored.)

0.6.3 Week 15

Thanksgiving break and all of the rest of the time before finals were spent cleaning up code, researching C++, and converting old scripts to Ansible YAML play books. This speeds things up.

0.6.4 Week 16

This week was relatively light. I blew a breaker in UC and left the building without power for two hours (cool but university wasn't thrilled). The full cluster is a power monster and I need to look into better power management, or more power. Or both.

0.6.5 Week 17 - Finals

This week was entirely unproductive given the presence of finals. The cluster was prepped and configured to run off of IPMI for break so more work can be done remotely and it can be set to run full bore once intersession starts.

0.7 Full Timelines - Spring Semester

This section is left blank in order to be properly filled out for the papers final release.

0.8 Required Software

This project is entirely dependent on a wide variety of software tools, published and unpublished. Due to the open-source nature of the Linux operating system, as well as its intense catering to developers; it is the perfect operating system to be used. UNIX type operating systems exist on the 'building blocks' mentality, ie. a combination of small tools, being used to collaboratively accomplish the final goal rather than a wide variety of complex tools that serve a single purpose. This exemplifies why Linux is a preferred operating system for this project; it allows for flexibility and optimization by only utilizing the tools chosen with minimal computational overhead.

0.8.1 Operating System - CentOS

CentOS is a particularly useful distribution of the Linux operating system environment. Short for Community Enterprise Operating System, CentOS is an enterprise, lightweight Linux distribution that caters to the needs of development heavy software ecosystems. CentOS is based on the RHEL (Red Hat Enterprise Linux) kernel and package managers. This incorporates all of the engineering and security / enterprise features that RHEL users have access to, but in a free distribution. Cent is still developed with Long Term Stability releases (LTS) in mind which makes is optimal for the project.

CentOS draws from the RPM software package management environment, utilizing the Yum package manager. RPM is short for 'RedHat Package Manager' which means that CentOS users have access to all of the enterprise standardized software packages that the RHEL platform is developed for. In addition to this, EPEL (Extral Packages for Enterprise Linux) and REMI repository package selections can also be added, which allows for the installation of supported GNU/GPL software tools. This project takes advantage of EPEL with REMI support disabled. Custom repositories are also generated but those will be covered later because they are non-standard with CentOS.

CentOS 6.5 and 6.6 are the current supported software versions as supported by the project. Although CentOS Version 7 was released shortly after the passing

of the proposal of this thesis, it was disavowed from potential use here due to its beta like state. CentOS 7 is a finished project but CentOS 6.5 and 6.6 are both still supported under LTS release for many years to come and are known to be bug free in a wide variety of arenas, all of which are required to be stable and efficient for the project to perform well.

CentOS incorporates a valuable software tool known as the Anaconda installer. Often known as a 'kick starter', Anaconda allows a large volumes of machines to all be imaged or freshly installed from a kick start script. This means that Cent can be installed via one of these scripts with little user input, which is the first step in autonomous node addition to the cluster. A valuable kickstart script has been been generated and used widespread and is held in the internal documentation. This script installs the correct architecture release of CentOS 6.6 on nodes while configuring networking, internal drive structures and firewalls and user account creation. From here, other software takes over to finish provisioning new nodes. Currently, provisioning a node takes just under an hour. Networking is dependent on the speed at which RIT RESNET works. In order to finish provisioning the nodes, networking is required, which requires proper registration onto the RIT network. This can take minutes or hours, and is purely at the mercy of the operating speed of RIT RESNET's student employees.

0.9 Software Provisioning - Ansible

Ansible is an excellent provisioning tool, similar to Salt and Puppet. Unlike the others, however, Ansible provisions over SSH which makes it fast and powerful to use. Once nodes are established on the network, they can be added to the Ansible hosts playbook, stored on the Head node and backup up to a repository. From here, the Head node can provision any other connected node, simultaneously.

One of Ansible's best features is its dual functionality. Ansible can either be utilized to run a command, ad-hoc, across an array of nodes from contained headers in the host file, or it can be used to run a specialized YAML syntax playbook which is used to execute a list of commands in order of specification across pre-designated clusters or groups. The first, ad-hoc method, is especially helpful for mass-testing or to simply perform basic tasks across nodes in an elegant manner. This is useful for Yum update operations or modifying the iptables of a series of nodes where it is easier to run a simple ad-hoc command rather than build a playbook for single use operations. The second is the most powerful feature of Ansible, because it is the backbone of elegant provisioning. Ansible allows for a hierarchical Playbook to be constructed. This playbook is formatted for YAML syntax, and acts like a complex to-do list. Modules and sub-modules can be defined and referenced in each playbook with advanced support for error handling and reporting.

Ansible is available free and open-source in its command line state. There are other, additional, enterprise and professional features like Graphical UI's, monitoring and alerting sent from a Proprietary application platform known as

Ansible Tower, but this was entirely unnecessary for the project. Ansible also offers enterprise support, but that was deemed unnecessary as well. Ansible is already incredibly easy to implement based on the command line, and can be installed via a simple Yum install argument so long as the dependencies are met. The dependencies are conveniently installed as part of the CentOS kickstart script, which means that installing Ansible is no problem. Ansible also works on both 32 and 64-bit architectures which makes it compatible with the entire cluster.

```
# Standard CentOS init script using Ansible
#




---
- hosts: fortress.student.rit.edu
  remote_user: root

  tasks:

  - name: Ensure the right version of CentOS is running
    shell: cat /etc/centos-release
    register: cat
  - debug: var=cat.stdout_lines

  - name: Check SSHD
    service: name=sshd state=started

  - name: Update Yum
    yum: name=* state=latest

  - name: Install GitHub
    shell: yum -y install git

  - name: Install Stress
    shell: yum -y install stress

  - name: Install Sysbench
    shell: yum -y install sysbench


```

Example Ansible Initialization Playbook Written In Ansible Native YAML

0.10 Benchmarking

Benchmarking an entire cluster comes with the responsibility of monitoring and performance testing a massive array of system variables. This process is intensely valuable to the progress of the project because the project relies on the cooperative power of a variety of machines. In order to gauge the effectiveness of these machine, there must be some quantitative, qualitative or arbitrary baseline by which to compare all machines. Machines that exceed or score a certain Currently, there are three phases to the benchmark process. One of these phases was removed early on, but will still be covered (albeit briefly) due to the time intensive nature it took to install and configure.

0.10.1 Homebrew Shell Benchmarking

Initially, the construction of an entirely custom benchmarking suite was envisioned. This led to a massive programming campaign that ended with several rudimentary tools being developed and configured to run on the cluster. This basic suite tested disk write speed and the speed at which the network could copy files via SCP. The results were then piped through shell programming to be stored on a centralized server. Another shell script parsed the information and removed the relevant information to be plotted via a service like Excel or MATLAB. A shell script running from the head node was implemented to run each of these processes on each attached node, which was inefficient and introduced before Ansible was implemented.

Eventually, this form of benchmarking was discontinued because it required a tireless amount of manual data analysis that grew exponentially in order to simply visualize the performance of each unit. The method was not efficient and provided no elegant output. As a result, although massive time was devoted for several weeks to home brew this suite, it was scrapped. Time earmarked for future development was devoted to finding open-source benchmark suites that scaled easily and generated elegant outputs in order to grant quick analytics.

0.10.2 Phoronix Test Suite

After searching, a viable benchmarking suite was discovered and implemented across the cluster. Phoronix is an open-source variable test benchmarking suite. Phororonix manages a massive library of tests, arranged by individual test names and custom generated test suites. These test suites contain groupings of individual tests that cater to a specific facet of a system to test (ie CPU, RAM or Disk I/O) or an array of tests chosen to test the entire holistic performance of a system. Phoronix shines, in respect, to the fact that one of the included tests tests the limits and functionality of dc.RAW - one of the key software components of the cluster.

Phoronix tests come with defaults but based on the engineering of the soft-

ware test packages, and one of these benchmarking tools can be modified. As of the end of Fall semester, modifying the dc.RAW test is underway and expected to carry into intersession, the academic period between Fall and Spring semester.

0.10.3 Geekbench 3

While PTS (Phoronix Test Suite) was a massive help in individual test benchmarking, it still did not provide an elegant approach to holistically testing the performance of systems. As a result of this, a third, and final, service was discovered. Geekbench is a proprietary software testing suite that came with massive power at a reasonable price point. Geekbench tests aver 100 facets of the test system in less than 30 minutes and quantifies the results against a reasonable baseline.

Well Known Geekbench 3 Logo

Licensing Geekbench 3 is a lifetime affair and has no limitations to the quantity of tests that can be run. This makes it ideal for a growing cluster. Within the Geekbench branded web dashboard, any two systems can be compared to see where one excels against the other, in any specific category. GB3 also combines all of these scores and averages them internally to generate an arbitrary ranking comprised of two numbers. The first is a comprehensive score of single core performance. The second, as can be inferred, is a meter of multi-core performance.

0.11 Current Hardware Configurations

Sparing the nitty gritty engineering documents that are contained in the internal documentation, this is a comprehensive list of the hardware contained in the presentation ready server rack. Although the generated software is built to be run independently on a linux cluster, there is necessary benchmarking and a token installation is integral to have.