

MULTI-CHANNEL RADAR DEPTH SOUNDER (MCRDS) SIGNAL PROCESSING:  
A DISTRIBUTED COMPUTING APPROACH

by

JE'AIME HENRI POWELL

A Thesis submitted to the Graduate Faculty of  
Elizabeth City State University  
In partial fulfillment of the  
Requirements for the Degree of  
Master of Science in Applied Mathematics

Elizabeth City, North Carolina

April

2010

APPROVED BY

---

**Dr. Linda Hayden – Chair of Thesis Committee**

---

Dr. Eric Akers  
Committee Member

---

Dr. Paula Viltz  
Committee Member

---

Dr. Benjamin Branch  
Committee Member

---

Richard Knepper  
Committee Member

Copyright by  
JE'AIME HENRI POWELL  
April 2010

## MULTI-CHANNEL RADAR DEPTH SOUNDER (MCRDS) SIGNAL PROCESSING: A DISTRIBUTED COMPUTING APPROACH

In response to problems surrounding measuring ice sheet thickness in high attenuation areas of Greenland and the Antarctic, the Center for the Remote Sensing of Ice Sheets (CReSIS) created a Multi-Channel RADAR Depth Sounder (MCRDS). The MCRDS system was used to measure ice thicknesses of up to five kilometers in depth. This system produced large datasets, which required greater processing capabilities-in the field. The purpose of this project was to test processing performance on a 32-core cluster through distributed computing resources. Testing involved a six-node cluster with an attached storage array and use of the CReSIS Synthetic Aperture RADAR Processor (CSARP) through the MATLAB Distributed Server Job Manager. Performance testing was derived from average run times collected once CSARP jobs completed. The run times were then compared using an ANOVA test with a five percent significance level.

## **ACKNOWLEDGEMENTS**

I would like to thank the following individuals for their guidance and assistance in the formation of this paper.

### **Family**

Mrs. Sonya Powell, Wife  
Mrs. Mary Powell, Mother

### **Elizabeth City State University Faculty and Staff**

Dr. Linda Hayden, Professor/Associate Dean, CERSER  
Mr. Jeff Wood, Multi-Media Technician, CERSER  
Dr. Eric Akers, Assistant Professor, CERSER  
Mr. Timothy Barclift, Networking Analyst, Network Services  
Mr. Kuchumbi Hayden, Networking Analyst, Network Services  
Mr. Shelton Spence, Director, Network Services  
Mr. Randy Saunders, Facility Maintenance Supervisor, Facilities Management  
Mrs. Sharonda Walton, Director, IT Client Services  
Mrs. Doris Johnson, Administrative Support Associate, IT Client Services  
Dr. Vinod Manglik, Professor, Mathematics & Computer Science

### **Indiana University Staff**

Mr. Richard Knepper, Staff/Student, UITS  
Mr. Matthew Link, Staff, UITS  
Mr. Corey Shields, Staff/Student, UITS  
Mr. Jefferson Davis, Staff, UITS

### **The University of Kansas Staff**

Mr. William Blake, CSARP Lead Programmer, CReSIS

### **Carnegie Mellon University Faculty**

Dr. David S. Touretzky, Research Professor, ARTSI

## TABLE OF CONTENTS

<b>Acknowledgements .....</b>	<b>iv</b>
<b>Table of Contents .....</b>	<b>v</b>
<b>List of Tables .....</b>	<b>vii</b>
<b>List of Figures.....</b>	<b>viii</b>
<b>Chapter I: Introduction.....</b>	<b>1</b>
Purpose of the Study .....	2
Research Questions .....	2
<b>Chapter II: Literature Review.....</b>	<b>5</b>
CReSIS Data Collection .....	5
<i>MCRDS in relation to CReSIS</i> .....	5
<i>MCRDS correlation with Synthetic Aperture Radar</i> .....	6
Distributed Computing.....	7
<i>Polar Grid in support of CReSIS</i> .....	7
<i>The MATLAB® Bottle Neck</i> .....	9
<i>Grid and Clustering in Cyber-Infrastructure</i> .....	11
<i>Grid Job Schedulers</i> .....	13
<i>MATLAB® Integration into Grid Technology</i> .....	14
<b>Chapter III: Materials and Methodology .....</b>	<b>17</b>
Experimental Delimiters .....	17
Cluster Hardware Requirements .....	17
Environmental Cluster Requirements .....	21
Cluster Networking Requirements.....	23
Operating System Installation and Configuration.....	26
<i>RAID Array Configuration</i> .....	27
<i>Operating System Initial Installation and Configuration</i> .....	27
<i>Network File System Configuration</i> .....	28
MATLAB Distributed Computing Server Installation and Configuration .....	29
CSARP Installation and Configuration.....	31
<i>CSARP Installation</i> .....	31
<i>CSARP Configuration</i> .....	32
<i>MATLAB Job Scheduler</i> .....	33
Running CSARP Jobs .....	34
<i>Initial testing on the Polar Grid Unit at Indiana University</i> .....	34
<i>PGU Sample Worker Testing</i> .....	37
Madogo Job Manager Configuration .....	38
Data Population Definition and Analysis .....	40
<i>Statistical Methods and Tests Used to Analyze the Data</i> .....	41

<b>Chapter IV: The Results .....</b>	<b>42</b>
What MATLAB toolkits and/or expansion kits are necessary to run CSARP? .....	42
What hardware requirements are necessary to store and process CReSIS collected data? .....	42
What facility environmental requirements are there to house a cluster of at least 32 cores to process a data set? .....	44
What is the process to prepare a cluster from a middle-ware stand-point? .....	45
Can an open-source job scheduler replace the MATLAB proprietary Distributed Computing Server currently required by CSARP? .....	46
Does this study prove within a 5% level of significance that the addition of computing cores increases the performance of the CSARP algorithm .....	47
<i>Data Set</i> .....	47
<i>Statistical Hypothesis Testing</i> .....	50
<i>Step 1: Parameters and Statistical Hypothesis</i> .....	51
<i>Step 2: Definition of the Level of Significance</i> .....	51
<i>Step 3: Test Statistic and ANOVA Testing</i> .....	52
<i>Step 4: Critical Value</i> .....	52
<i>Step 5: Decision</i> .....	53
<i>Step 6: Conclusion</i> .....	53
<b>Chapter V: Discussion .....</b>	<b>54</b>
Further Worker Performance Results .....	55
CSARP Recommendations .....	59
Future Work .....	60
<b>References .....</b>	<b>62</b>
<b>Appendix.....</b>	<b>A</b>
ECSU Param file .....	A
Startup.m .....	D
Stage1a.m (1 Worker Example).....	E
Stage1b.m (1 Worker Example) .....	G
Stage2.m (1 Worker Example) .....	I
IU Worker Collected Times.....	P
Madogo Worker Collected Times.....	HH
Cluster Power and Cooling Table .....	YY

## LIST OF TABLES

Table 1: MATLAB 2009b Linux System Requirements .....	18
Table 2: Parallel Computing Toolbox and MATLAB Distributed Server Network Requirements .....	18
Table 3: Cluster unit configurations .....	20
Table 4: Equipment Power Needs at Maximum Usage .....	22
Table 5: NFS Host General Configuration .....	28
Table 6: Parameter file variable functions .....	33
Table 7: Cluster setup section .....	33
Table 8: Worker Times in Seconds.....	51
Table 9: One-way ANOVA p-value results.....	52

## LIST OF FIGURES

Figure 1: Cluster Topography .....	3
Figure 2: Cluster to Grid Comparison .....	12
Figure 3: Integrated MATLAB to Condor Web Services Workflow .....	15
Figure 4: Client-Side Task Support in MATLAB for Concurrent Distributed Execution	16
Figure 5: Madogo Cluster Complete Topography .....	21
Figure 6: Network Interconnections .....	25
Figure 7: Storage Array Network Interconnections.....	26
Figure 8: Starting Client Worker .....	30
Figure 9: MATLAB Distributed Server Topography .....	31
Figure 10 Startup.m CReSIS Toolbox configuration .....	32
Figure 11: "Stage" file line to designate the number of workers .....	34
Figure 12: Initial CSARP Workflow .....	36
Figure 13: CSARP command line execution .....	37
Figure 14: PGU Worker Mean Run Times .....	38
Figure 15: Top CPU display during CSARP execution.....	39
Figure 16: Final Madogo cluster topography.....	40
Figure 17: ECSU's Madogo verses IU's BCS mean run times .....	44
Figure 18: Image generated from 20080715B data .....	48
Figure 19: 1 Worker Job Display.....	49
Figure 20: Worker mean total run times as derived from stage times .....	50
Figure 21: Madogo worker mean run times graph.....	56
Figure 22: MATLAB "Better Fit" equations .....	57
Figure 23: Shape-preserving interpolant.....	58
Figure 24: Interpolant with 128 worker estimate.....	59



## **CHAPTER I: INTRODUCTION**

The idea of global sea level rise forced many scientist and government representatives to search for hard data to prove or disprove the idea of ice sheet regression (Mank, 2005). In response to this request, the Center for the Remote Sensing of Ice Sheets (CReSIS) (historically the University of Kansas Remote Sensing Laboratory) set out to design a RADAR Depth Sounder that could accurately measure the thickness of large sea and land ice masses. From ice-core simulations performed between the years of 2002 and 2003 it was suggested that a RADAR system tuned between UHF 300-1300MHz for sounding thin ice and VHF 50-250MHz for sounding thick ice would measure ice thicknesses with less than a 20cm variance (V. Ramasami & S. K. Namburi, 2003). This finding led to the creation of a Multi-Channel RADAR Depth Sounder (MCRDS) for use in the Arctic and Antarctic regions in an effort to provide missing ice thickness information in high attenuation areas including calving fronts. Beginning in 2006, both Arctic and Antarctic aerial missions utilized MCRDS radar technology to map ice sheet depths. The vast amount of computing power needed to store and process the collected data led to the 2007 funding of the National Science Foundation's Polar Grid: Cyber infrastructure for Polar Science Major Research Instrumentation (Geoffrey Fox, 2007). Polar Grid funding allowed clustered computing power to be purchased for the purpose of processing CReSIS radar data in-situ.

## **Purpose of the Study**

The purpose of this study was to verify the viability of distributed computing principles with the use of the CReSIS Synthetic Aperture RADAR Processor (CSARP). In short CSARP required a large amount of computing power and time to perform needed operations on MCRDS collected data. Although it was suggested that a decrease in processing time would occur with the addition of computing cores, the true benefits or lack thereof was neither explored nor quantified using true data collected from the field during initial cluster deployments in Greenland of the year 2008.

## **Research Questions**

This study intended to prove within a 5% level of significance that the addition of computing cores increases the performance of the CSARP algorithm. This study also was designed to answer the following sub-questions:

- What hardware requirements are necessary to store and process CReSIS collected data?
- What facility environmental requirements are there to house a cluster with 32-cores to process a data set?
- What is the process to prepare a cluster from a middle-ware stand-point?
- What MATLAB toolkits and/or expansion kits are necessary to run CSARP?
- Can an open-source job scheduler replace the MATLAB proprietary Distributed Computing Server currently required by CSARP?

The methodology behind the study began with the installation and configuration of a 32-core cluster. The cluster consisted of one International Business Machines (IBM) System x3650 2 unit (U) server, and three IBM System x3550 1U servers as seen in Figure 1: Cluster Topography. Both the System x3650 and the System x3550 servers contained dual quad-core 2.33GHz Intel Xeon processors. Therefore each server contained a total of eight cores for a product of 32- cores. The System x3650 was given the role of head node. Within this defined role. both the random access memory (RAM)

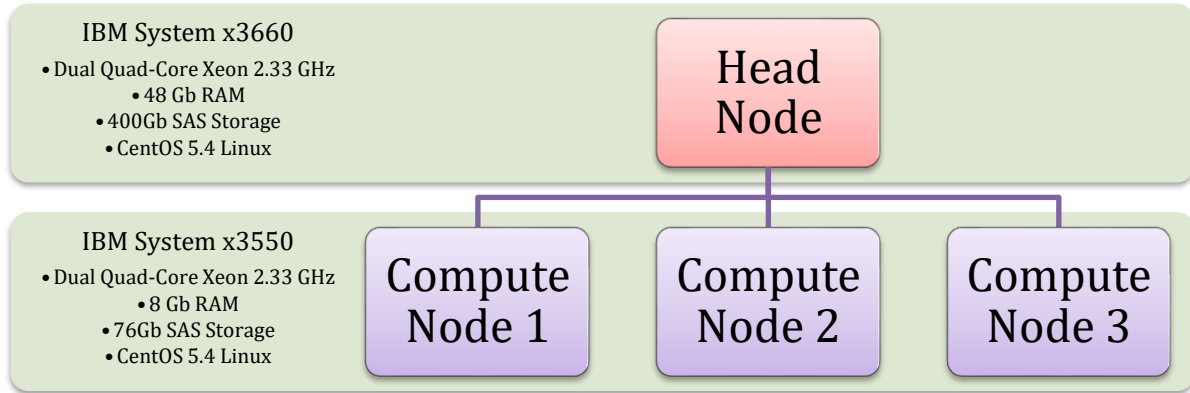


Figure 1: Cluster Topography

and physical spinning disc storage were increased to 48Gb and 400Gb respectively. The System x3550 compute nodes each were configured with 8Gb of RAM and 76Gb of spinning disc storage. All of the storage in the form of spinning discs were 2.5” serial attached small computer system interface (SAS) hard drives. The operating system CentOS 5.4 x86\_64, a Linux distribution was utilized on all the computers. In this configuration the cluster was named “Madogo” which means “junior” in Swahili. The

included cluster management tools allowed the head server and three compute servers to be administered from the console of the head node through a secure shell. MathWorks MATLAB Distributed Server (MDS) was then installed on all four servers as a prerequisite for running the CSARP scripts. The MathWorks MDS Job Manager was also installed and configured as prerequisites for CSARP. The CReSIS toolbox and processing scripts along with data from the 2009 field season were transferred to the head node and the MCRDS “Params” file was configured to point to the location where the data was stored. Within the staging files (i.e. stage1a.m, stage1b.m and stage2.m) the number of workers (cores) was also adjusted as needed for the trials. Processing time was collected from the results of the standard output and saved into a text file after submitting a CSARP job into the prescribed scheduler. Each trial was repeated 20 times to establish a viable mean for 1,2,4,8,16 and 32 workers. These values were then analyzed with an analysis of variance (ANOVA) to determine if a mean change in performance within 5% level of significance occurred. An ANOVA test in combination with the level of significance was used to determine how likely the results were to have occurred by chance.

## CHAPTER II: LITERATURE REVIEW

### CRISIS Data Collection

#### *MCRDS in relation to CRISIS*

The principles behind measuring ice sheet thickness using radar originated in the year 1933. Pilots flying around Admiral Byrd's Base camp in Little America, Antarctica reported that radar altimeters were "useless" over ice (Allen, 2006). An investigation into the matter led by researchers from the U.S. Army found that polar ice was transparent in the UHF and VHF bands. These findings led Armory Waite to show that the radar altimeter SCR 718 could measure the thickness and other features of polar glaciers at 440MHz in 1957. Waite's success observed led to the technical advancements in radar use for the creation of radio-echo sounding systems for the sounding of ice masses.

In order to model ice sheet masses an accurate measurement of ice sheet thickness was needed. In an attempt to measure missing data in areas where high attenuation of radar signal occur, CRISIS designed a MCRDS system. MCRDS was an airborne radar that combined a highly sensitive receiver, an array of folded dipole antennas, pulse compression, and a transmit power of 800W to provide depth sounding data of the polar ice sheets. The radar was developed between October 2005 and April 2006. Design aspects included radio frequency and digital printed circuit boards, system mounting and housing, and digital acquisition software. Upon completion, the MCRDS system was mounted on a Twin Otter aircraft and flight tested in both Northern Canada and Greenland in 2006 (Lohofener, 2006). Further revisions of the MCRDS system allowed

the CReSIS team to peer through ice sheet layers up to three kilometers in thickness (Oberthaler, 2009).

### *MCRDS correlation with Synthetic Aperture Radar*

Through the use of MCRDS technology the depths of large ice masses were measured; however, the processing of the raw telemetry was what truly aided scientists. When MCRDS flew over a target, multiple hard drives collect individually sampled signals at a rate of up to 470MHz or  $4.07 \times 10^8$  times per second. Assuming no data loss, an average flight time of two hours, and a storage size of one byte per collection, data set estimates range up to 3TB ( $2.9304 \times 10^{12}$  bytes) per flight, per collector. Though this presents a great deal of data, it only presents the profile of a slice from one angle. In order to generate full three-dimensional images, use of synthetic aperture radar (SAR) processing was necessary.

SAR geometry involved a platform moving at a velocity and altitude with a “side looking” radar antenna that illuminated an object with pulses of electromagnetic radiation (Oliver & Quegan, 2004). From that geometry the resulting signals returned to the platform and were recorded. Objects were detected due to varied magnitudes and phases in the signal. After a given number of cycles, the signals were combined with respect given to Doppler effect incurred. That data was then utilized to create a high resolution images of the terrain (Wolff, 2009).

CReSIS simulated SAR processing (which was generally a hardware-based process) through the use of software code. In a perfect world this data would be clean and could

be plotted directly in order to display an echogram of an ice bed. Realistically there were many environmental elements that created distortion in the collection of the radar data. The distortion could result from small things such as an antenna misalignment, to cockpit radios within the aircraft. To clean and filter the data, a program entitled the CReSIS SAR Processor (CSARP) was written to both combine the MCRDS swaths as well as image the data. SAR processing often used a single antenna to act as an array. In the case of CSARP combined with MCRDS, there was the combination of a large swath through a true antenna array along with the high-resolution imagery created through the overlaying of signal slices.

## **Distributed Computing**

### *Polar Grid in support of CReSIS*

To increase productivity and expand the capabilities of MCRDS SAR processing the Polar Grid Major Research Instrumentation grant was awarded by the National Science Foundation in the year 2007. Indiana University and Elizabeth City State University in partnership received funding to acquire and deploy the computing infrastructure needed to investigate the urgent problems in glacial melting (Geoffrey Fox, 2007). Previously, data collected during CReSIS field deployments could not be processed in either real or near-real time. Data could not be processed until received back at the CReSIS cluster through physical delivery to Lawrence, Kansas. Polar Grid provided the funding to purchase computing capacity, which would allow an data processing capability in the field.

The initial configuration of the Polar Grid cluster was deployed and tested in July of 2008 in Illulisat, Greenland (Hayden, Powell, & Akers, 2009). The cluster was composed of one IBM x3550 1U server named C01, one IBM DS4200 3U RAID Controller and two IBM EXP420 3U RAID Storage devices. The controlling server (named C01) enclosed two quad core 2.33GHZ Xeon processors. In the initial configuration the server contained 16GB (8 x 2 GB DDR DIMMs) of RAM and three 76 GB internal Serial-Attached SCSI (Small Computer System Interface) (SAS) drives for running the operating system (RedHat Enterprise v.5.1) and user home directories. When transferring data from CReSIS RADAR equipment, a proprietary compact peripheral control interface (CPCI) developed by the University of Kansas (KU) was utilized. The CPCI box ran Cent OS to simply mount the ext3 partition of the radar drive, and to allow connectivity to the data through the built-in 10/100/1000 Ethernet port. The IBM DS4200 RAID controller and IBM EXP420 storage expanders added an additional 13TB of usable storage connected through dual 1Gb fiber channel connections to the server. The added benefit of the RAID storage units was that the combined storage of all 48 hard drives could be accessed as one partition, while redundant storage was handled simultaneously.

Previous missions determined the data storage needs of the cluster. Each standard flight averaged 302GB of data. With a predetermined 22 flights, it was calculated that a minimum of seven tera-bytes of archival storage would be required. An additional two tera-bytes were added to accommodate storage used for processing. Once those specifications were known, the RAID array was ordered, assembled, and configured in a RAID 10 scheme.



It was found in the field that if more than four workers were utilized to run CSARP, input/output (I/O) limitations would create a failure. In the field, upgrading the RAM in C01 from 16Gb to 32Gb solved this. The primary difference between the expected results in this project was the addition of more cores. Initially only one node was utilized with eight cores as opposed to the combination of four nodes with a total of 32-cores.

### *The MATLAB® Bottle Neck*

As previously stated, CSARP was the primary tool for processing CReSIS data. This tool was a custom created script and toolbox which uses MathWorks™ MATLAB® as the interface. MathWorks™ describes MATLAB as, “a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran (“MATLAB - The Language Of Technical Computing,”).” Though MATLAB® was designed to be truly scientist-friendly in the creation of prototypes, production use was thought to require translation into a “traditional” programming language. The inclusion of multiprocessor principles was not an option commercially until later in the development of the programming environment. Before that time only single sessions of MATLAB® jobs could be run on a computer. This produced a huge limitation in the types of jobs MATLAB® could be assigned to perform. Third-party scientist frustrated with these limitations began to build support for parallel computing into the software. Custom written toolboxes such as MITMATLAB (Parry Husbands, Jr., & Edelman, 1998), MATLAB\*p (P. Husbands & C.Isbell, 1999), MATLAB\*G (Chen & Tan, 2002), and

MultiMATLAB (Trefethen, et al., 1996) led early solutions to limitations within the software. The popularity of multi-processor capabilities led MathWorks™ to release the Parallel Computing Toolbox™ which allowed users to “solve computationally and data-intensive problems using MATLAB® and Simulink® on multi-core and multiprocessor computers(" Parallel Computing Toolbox 4.2: Perform parallel computations on multicore computers and computer clusters," 2010).” In combination with a job scheduler, not limited to the MATLAB Distributed Computing Server (DCS), MATLAB was then able to natively handle the large problems through concurrent asynchronous processes.

The proliferation of the MATLAB Distributed Computing Server had taken hold within CReSIS due to the limitations of the Parallel Computing Toolbox. According to MathWorks, the parallel toolbox only allowed up to eight worker cores with the MATLAB 2009b version (Edelhofer, 2010). Prior to that MATLAB only allowed up to four workers therefore, the DCS was required. The distribution of CSARP through the DCS presented challenges in economical terms, in I/O throughput and in greater resource usage. Economically, DCS is extremely expensive. A MathWorks quote from October 2009 for a DCS 128 core pack with maintenance was \$22,399.84 with academic pricing (Sheridan, 2009). When compared against free open-source alternatives, the funds could be spent on the acquisition of a much larger amount of computing power. Due to the fact that DCS required the MATLAB front-end to process scripts, I/O becomes an immediate problem with running large data sets. In the Greenland 2008 deployment, it was found that even with eight Xeon processors, dual fiber-channel connected storage arrays and 16Gb of RAM, CSARP would only function consistently with four or less workers and

required system restarts between jobs to clear the memory (Hayden, et al., 2009). A preliminary goal of the Polar Grid Grant was to leverage the resources of the national TeraGrid (Geoffrey Fox, 2007), one of the world's largest distributed cyberinfrastructures. The issues lied within the fact that DCS nor MATLAB were considered standard software packages for large data sets and were therefore not commonly available on larger resources such as TeraGrid.

TeraGrid had three primary focus areas. Its deep goal was to support the most challenging computational science activities—those that could not be achieved without TeraGrid facilities. TeraGrid's wide mission was to broaden its user base. The project's open goal was to achieve compatibility with peer grids and information services that allowed development of programmatic interfaces to TeraGrid. A part of TeraGrid's wide initiative was the TeraGrid Science Gateways program (Wilkins-Diehr, Gannon, Klimeck, Oster, & Pamidighantam, 2008).

### *Grid and Clustering in Cyber-Infrastructure*

Though often debated, there was a general acceptance that a computer cluster was a group of computers centrally located performing either parallel or distributed tasks. Clusters are composed of commercially available components. A grid generally consists of clusters or supercomputers connected through Internet based network infrastructure, which was geographically distributed, to perform parallel or distributed tasks. The key difference between clusters and grids was the distance between physical locations. For example, ECSU is in the process of installing a 512-core cluster named Umfort (Summer

2010) however when it is connected with IU's Quarry cluster the two will be considered to be a part of the Polar Grid as seen in Figure 2: Cluster to Grid Comparison.

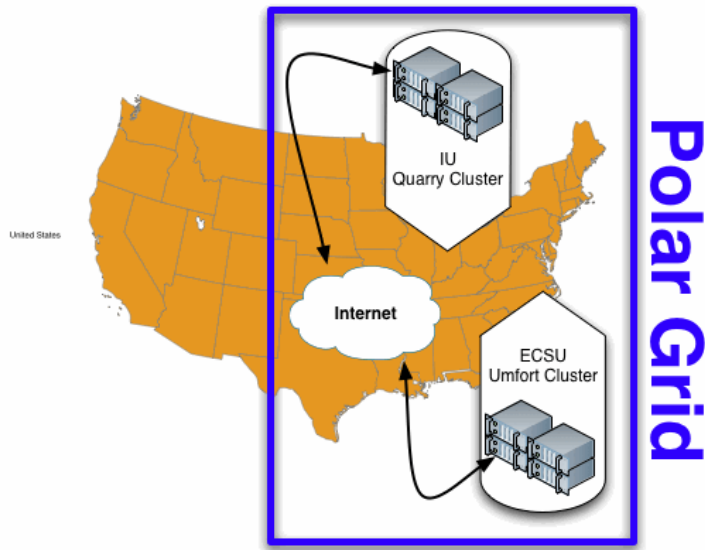


Figure 2: Cluster to Grid Comparison

A second element of topography for a grid was that of individual desktops connected via a network and some form of middleware, which allows jobs to be distributed. The grid could either be local or geographically separated. An example of this was the ECSU VikeGrid that consisted of a dynamic number of laboratory desktop workstations and server-grade computers in various areas of the university campus. The majority of the nodes on the VikeGrid are student workstations during normal business hours, which are then utilized as computing resources during off peak hours. The computers, though geographically close, are utilized as independent compute resources on the campus grid.

## *Grid Job Schedulers*

Once any cluster is established, a job scheduler becomes a primary focus. Cluster management software gives an administrator the ability to see what jobs are active, suspended or pending (Tittel, 2010). Though many open source and commercial variations exist, the MATLAB Job Manger is the scheduler considered for this project. With this in mind, open-source schedulers were explored as possible replacements for the used scheduler.

Condor HTC was the product of the Condor Research Project developed at the University of Wisconsin-Madison University in 1987 (Litzkow, 1987). The purpose behind the creation of Condor was to utilize workstations that were productive during the workday, but were unused at other times. Those unused times left “free” clock cycles at a time when processing power was at a premium. These “wasted” cycles were seen as an unused resource to which Condor was developed to cultivate. The “cycle-stealing” method of the Condor job scheduler allowed pools of dynamically designed computers to work together as a statically constructed supercomputer. The primary control system components consisted of a "central resource manager", which gathers general information about all the machines, and a "local scheduler" which makes decisions affecting only a particular workstation. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion (Project). To extend the capabilities of a local pool, condor uses the concept of “flocking” to create grids. Condor flocking is based on the premise that across different Condor pools that are owned by various departments there is much idle time that can be joined together though

managed separately (Evers, de Jongh, Boontje, Epema, & Van Dantzig, 1994). The primary reason Condor was viewed as a possible solution to this project was flexibility within the architecture of the scheduler. The dynamic ability of Condor to operate on varied types of computers added the possibility of including non-clustered resources in order to process MCRDS data.

As previously stated in section “The MATLAB® Bottle Neck,” the MATLAB Distributed Server (MDS) had already been used to batch CSARP generated jobs. Through the use of MDS and CSARP the raw data files were shared out between processors in order to create the CReSIS formatted files in stage 1a, add the height data in stage 1b, and lastly complete the F-K migration in stage 2. Without the inclusion of the MATLAB Parallel Computing (MPC) toolkit CSARP was limited to the use of only one processing core. With MPC the core density was increased to four and then again to eight with MATLAB 2009b. With MDS the density could be increased up to 128 cores. The key feature of MDS was the minimum number of code additions to expand the distribution capabilities of CSARP.

#### *MATLAB® Integration into Grid Technology*

Prior information regarding grid integration has been presented in this paper. However; specific use cases were a requirement before beginning any type of implementation. One such example used a custom designed computational toolkit to engineer design and search optimization. The process involved identifying design parameters that the engineer wished to optimize, computing a measure of the quality of a particular design (the objective function) using an appropriate model, and using a number

of design search algorithms to generate additional information about the behavior of a model over the parameter space, as well as to optimize the objective function to improve the design quality (Xue, Fairman, Pound, & Cox, 2003). The team's toolbox consisted of a low-level java interface that utilized Condor web services. The workflow of the model is shown in Figure 3: Integrated MATLAB to Condor Web Services Workflow. The integration of Condor like commands into the actual application interface of MATLAB made this project very unique when compared with other solutions which created custom scripts within the operating system. The team focused on ensuring the user had a unified environment in which to qualify, submit, check and analyze Condor jobs. The use of this particular all-in-one formula was an ideal way to processes CSARP data without the additional programming knowledge being required of prospective users.

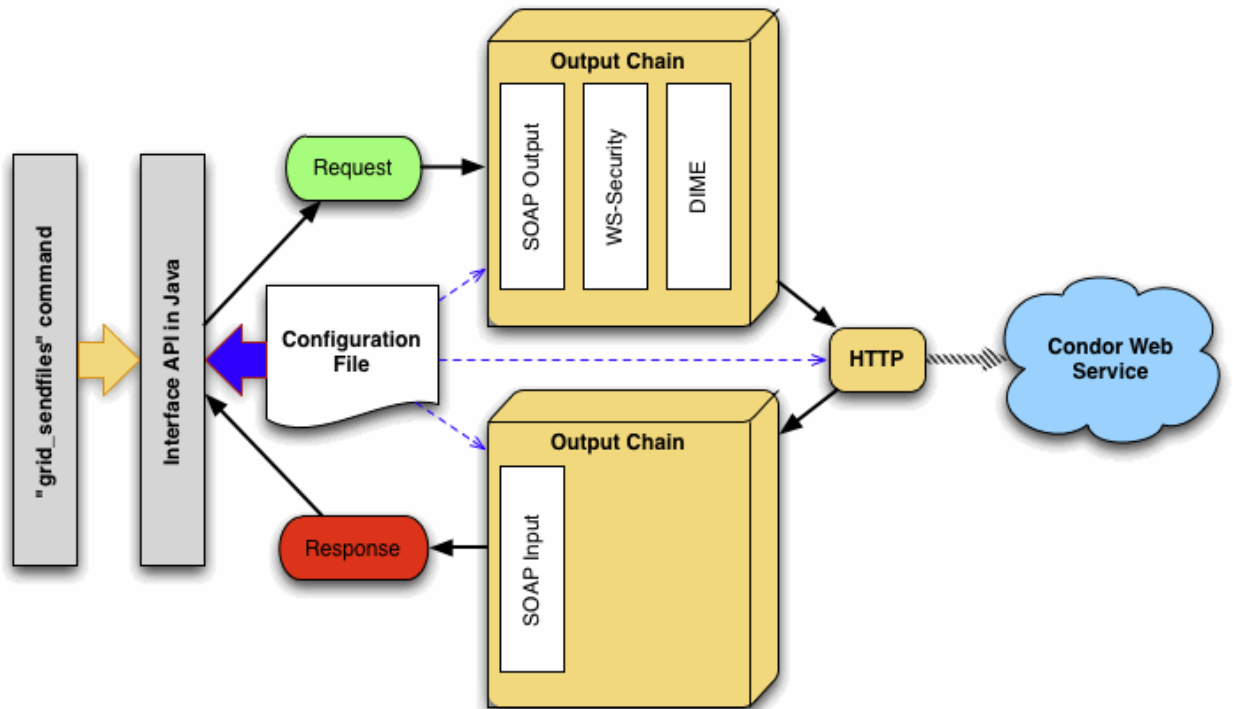


Figure 3: Integrated MATLAB to Condor Web Services Workflow

This project was not the first to prove the idea that grid principles could improve processing periods for data analysis. This research team leveraged the strength's of MATLAB's interactive environment, along with its integrated java virtual machine to utilize simple object access protocol (SOAP) web services to perform task-based parallelism (Christian Hoge, Dan Keith, & Allen D. Malony, 2007). SOAP was designed to be a new protocol for the decentralized, distributed environments utilizing the power of the Internet and XML to pass typed information between nodes (Suda, 2003). As seen in

Figure 4: Client-Side Task Support in MATLAB for Concurrent Distributed Execution, the authors found a decrease in processing time of 67% from the use of a single processor to a cluster with 83 processors. These findings added proof that the possibility of paralyzing CSARP should not only be researched, but that there would be significant leaps in performance.

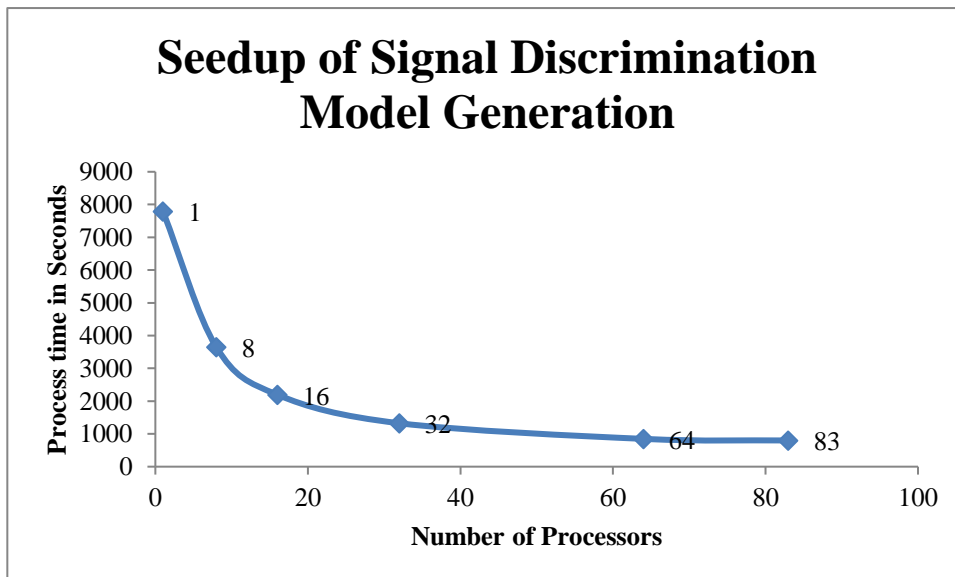


Figure 4: Client-Side Task Support in MATLAB for Concurrent Distributed Execution



## **CHAPTER III: MATERIALS AND METHODOLOGY**

### **Experimental Delimiters**

The following is a list of the areas, which are beyond the scope of this report and as such will not be highlighted.

- Operating System installation walkthrough
- Linux server administration
- Linux distributions other than Cent OS v.5.4
- Cluster management software
- File server configuration other than physical storage array to network connections
- Network switch management
- Storage Array software installation
- ECSU network (wide area network) configuration
- MCRDS RADAR detailed configuration
- CSARP complete source code and/or complete workflow description
- CReSIS Toolbox complete source code and/or complete workflow description
- Comparison with various hardware vendor performance profiles other than IU's Polar Grid (subsection of Quarry) and ECSU's Madogo clusters
- CSARP compilation into any other programming language.
- Networking basic concepts
- Network connection concepts
- Job scheduler implementation other than MATLAB Distributed Server Job Manager
- Cluster security profiles
- Network files system configuration
- Analysis of variance historical origins
- Regression historical origins
- T-test historical origins
- P-value historical origins
- Dat file use to create visualizations
- MiniTab use other than ANOVA
- MATLAB version 2009b

### **Cluster Hardware Requirements**

With the understanding that multiple computers would be used to conduct research, a list of those computers was generated. Power, cooling, networking, and server

side software were identified to correlate with both grid topography needs and CSARP needs.

Due to the fact that CSARP utilized MATLAB 2009b along with MDS and both the parallel computing and signal processing toolboxes, the system requirement for those packages were the initial conditions that had to be met. With the pre-condition that the cluster would be Linux-based MathWorks determined that MATLAB 2009b alone required at minimum (MathWorks) the information located in Table 1: MATLAB 2009b Linux System Requirements.

Table 1: MATLAB 2009b Linux System Requirements

Operating Systems	Processors	Disk Space	RAM
64-Bit MathWorks Products			
<ul style="list-style-type: none"> <li>• Debian 4.0 and above</li> <li>• Red Hat Enterprise Linux v.4 and above</li> <li>• OpenSuSE 9.3 and above</li> <li>• Ubuntu 8 and above</li> </ul>	Intel Pentium 4 and above Intel Celeron Intel Xeon Intel Core AMD64	500 MB (MATLAB only)	1024 MB (At least 2048 MB recommended)

Table 2: Parallel Computing Toolbox and MATLAB Distributed Server Network Requirements

Requirement	Reason
Hostnames mapped within the dynamic naming server	Distributed computing processes must be able to identify each other by hostname, which requires the presence of a DNS or equivalent service on the

	network.
The hostnames must be bound to an IP address the must match the host's ethernet card	Distributed computing processes will work correctly on machines with multiple NICs.
For the computer running the MathWorks job manager process we recommend setting a high limit for the number of per-process file descriptors.	This is especially recommended for users running 64 or more workers with the job manager. Failure to do so can result in unpredictable, hard to detect failures. Typical symptoms include failed job manager database operations.

Though the signal processing toolboxes only requirement was MATLAB (MathWorks), both the parallel processing toolbox and MDS required additional hardware and network items. The addition of the two packages changed the 2048MB RAM recommendation to a requirement because each local worker was running the equivalent of a full-fledged MATLAB session (MathWorks). Each package also required an additional 5GB of storage for the local scheduler to operate. The network requirements for the distributed portions were very specific in nature to allow multiple systems to utilize the same information. Table 2: Parallel Computing Toolbox and MATLAB Distributed Server Network Requirements lists the applicable requirements and their importance to the server configuration. Including the operating system the minimum requirements were increased and the configuration of the server and compute nodes in Table 3: Cluster unit configurations, was utilized.

To accommodate a complete data set for CSARP to process a set field season data for processing had to be identified. Whereas the original equipment was only utilized in the 2008 field deployments, it was decided the 2008 Greenland Air deployment would be utilized. In order to house the complete 2008 Greenland data set a storage amount

equivalent to 9TB would be needed. To hold this amount of data the same IBM 4200 with two IBM EXP420 RAID arrays which were used in the 2008 field season to house the data. In order to reduce the local storage requirements of the data set, the specific flight date of August 1, 2008 was chosen. This particular data was MCRDS calibration data, which totaled 55GB. With the addition of battery backup units, network switching, additional compute nodes and external RAID storage the complete Madogo cluster was mounted into three 12U and one 27U styled racks.

Table 3: Cluster unit configurations

<b>Head node - Hostname: Madogo.ecsu.edu</b>	<b>Compute Nodes (4) – Hostnames: clustN(3 – 6).ecsu.edu</b>
Model: IBM System x3560 Operating System: CentOS 5.4 (Opensource RedHat) Processor: Dual Quad-core Intel Xeon 2.33GHz RAM: 48 Gb Diskspace: 400Gb Array, 76Gb Array Bound IP address: 10.16.3.49	Model: IBM System x3550 Operating System: CentOS 5.4 (Opensource RedHat) Processor: Dual Quad-core Intel Xeon 2.33GHz RAM: 8 Gb Diskspace: 76 Gb Bound IP Addresses: 10.16.3.42 - 45

## Madogo Cluster

### Complete Topography

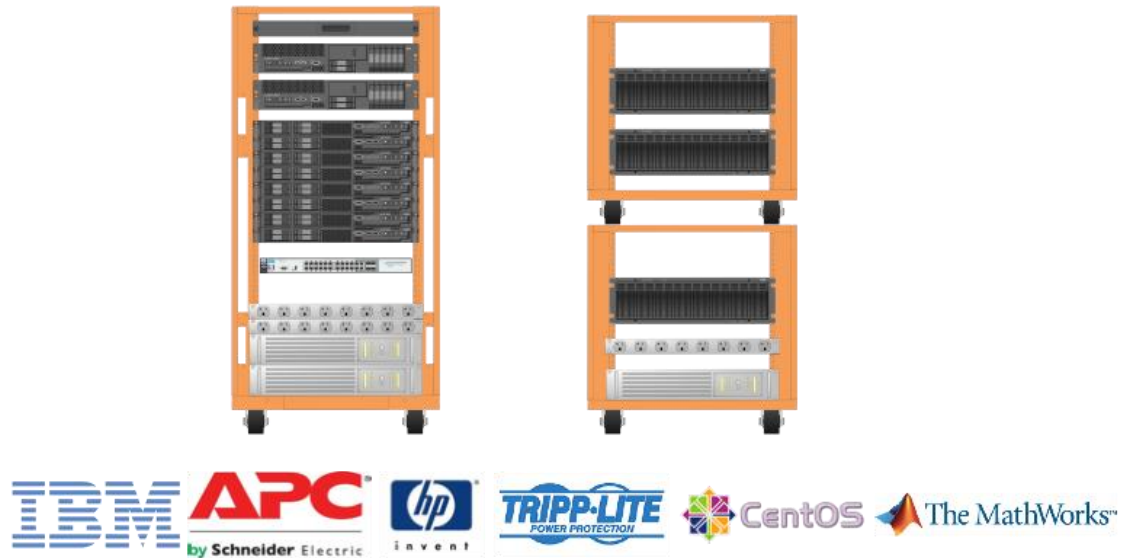


Figure 5: Madogo Cluster Complete Topography

### Environmental Cluster Requirements

The completed Madogo cluster presented power and cooling concerns due to the large amount of resources that were connected together within a small area. In order to ensure the cluster had proper power and cooling available an analysis of the requirements was conducted.

Power requirements were the first items to receive analysis. The power needs of the cluster were gathered utilizing the specifications of the equipment provided by their manufactures. Table 4: Equipment Power Needs at Maximum Usage shows that with all equipment working, there would be a need for 15,105W of power. These needs were split

over three APC Smart-UPS battery backup devices and connected to three separate ~230 VAC circuits at 30 amps each. These connections were made through National Electrical Manufacturers Association (NEMA) L6-30 locking connectors. Qualified electricians at ECSU installed both the outlets and required breaker box.

Table 4: Equipment Power Needs at Maximum Usage

Equipment Name	Units	Power Draw	Unit * Power Draw
IBM System x3650 (IBM)	2	675W *2	2700W
IBM System x3550 (IBM)	8	675W *2	10,800W
IBM DS4200 ("IBM System Storage DS4200 Express Storage Subsystem: Installation, User's and Maintenance Guide," 2007)	1	443W	443W
IBM EXP420 ("IBM System Storage DS4000 EXP420 Storage Expansion Enclosure: Installation, User's and Maintenance Guide," 2006, 2007)	2	443W	886W
Tripp-Lite NetDirector Console KVM (Lite, 2009)	1	22.8W	22.8W
HP 24-port ProCurve (Networking, 2006)	1	254W	254W
<b>Total</b>			<b>15,105.8W</b>

Once the power requirements were derived, the cooling requirements could be calculated from those numbers. This can be done because heat is a form of energy. Heat generated by equipment was commonly expressed as a rate of output in British thermal units (BTU) per hour (hr), tons per day, and joules per second where joules per second is equal to watts (Rasmussen, 2003). Though the beginnings of a global trend away from the use of BTU per hour had started near the time of construction, it was the standard way to describe the needed cooling at the time. To convert between the two, the product of

power in watts and the conversion factor 3.41 was taken. The result was of this calculation was that the cluster would produce 51,510.8 BTU/hr. To further evaluate, the conversion to tons was made by taking the product of the power in watts and the conversion factor .000283. The result was 4.27 tons of cooling would be required.

$$BTU/hr = Power_{Watts} \times 3.41$$
$$Tons = Power_{Watts} \times .000283$$

#### Equation 1: Power to Cooling Conversion

In order to gain a better understanding of these figures they were compared to cooling needs for a standard home. An average central air conditioner size would be between 3-6 ton or 36,000 - 72,000 BTU's/hr ("Central Air Conditioning Unit Size (AC)," 2010). With this in mind it was then understood that to cool a 10' X 10' room housing the Madogo cluster, the power and cooling used in a standard full sized home would be required. As a solution to this problem an in wall air-conditioning unit was temporarily installed with a sump pump to remove any condensation to the outside of the building. With both power and cooling concerns addressed, the configuration of the server interconnections could proceed.

#### **Cluster Networking Requirements**

The interconnections between the cluster head unit, compute units, storage arrays, and the Internet played a vital role in the performance of the cluster when distributing job related data. The primary platform for these connections was a HP

ProCurve 2900-24G 10/100/1000 MB switch capable of using both category six (Cat6) copper wiring and fiber through the addition of a Gigabit-SX-LC Mini-GBIC. In the case of the fiber optic connection on the switch, the primary function was to connect to the wide area network available on the campus of ECSU. Though cluster management was not within the scope of this project, the connections for the networking of the cables were organized as well so as to allow such software to be added at a later date.

The networking for the IBM System x3550s and x3650s were relatively simple. There were three RJ-45 network ports on each unit. Two of those ports were configurable gigabyte Ethernet ports, “eth0” and “eth1.” The third port was an Intel® Integrated baseboard management controller (BMC) for the remote management of the unit through a cluster management application. Eth0 was utilized as the port for the connection to the external wide area network for all units. Each eth0 port was assigned a static IP address within the 10.16.3.0 scheme as per ECSU network requirements and connected via Cat6 to the HP ProCurve switch. For future management applications, the Eth1 and BMC ports respectively were connected to an unpowered 10/100 MB switch. The connections can be visually identified in Figure 6: Network Interconnections.



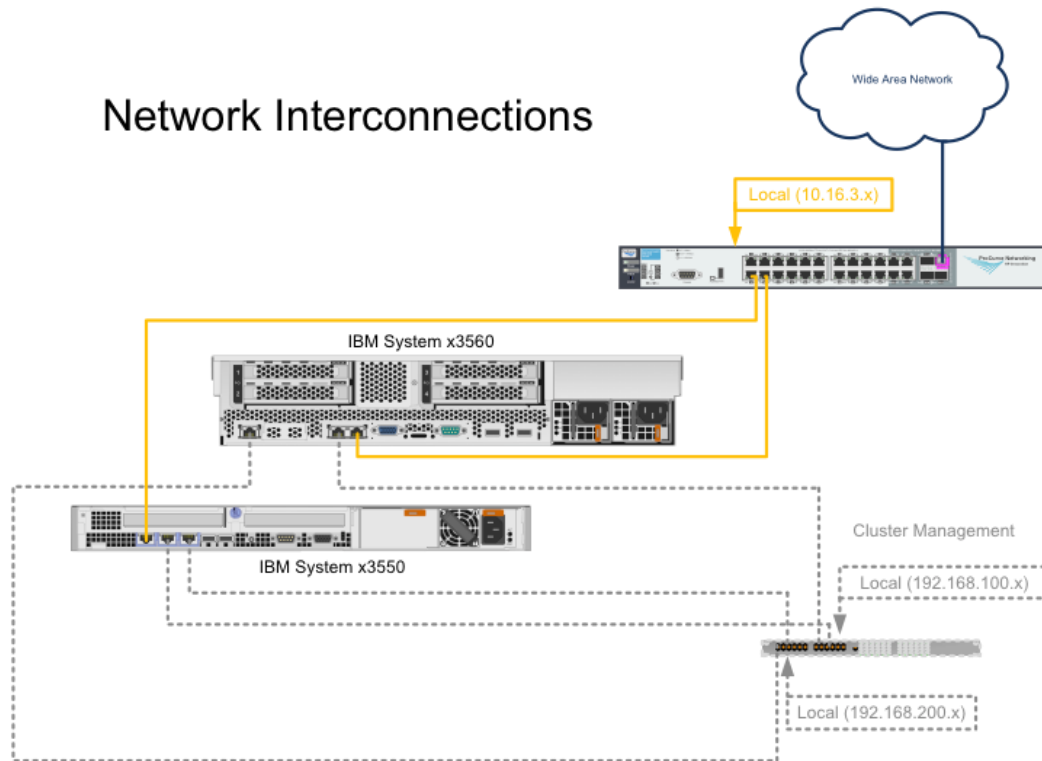


Figure 6: Network Interconnections

As previously stated in Cluster Hardware Requirements, there was a deficiency in the ability for Madogo to house the complete data set for the 2008 field season. In order to solve this deficiency in the same manner as utilized in the field, a storage array was connected to a second IBM System x3650 to serve the data through a shared file system. The connection to this “file server” was through a dual fiber channel connections along with redundant Cat6 connections to access the data. The dual fiber channel connection to the file server was established with the addition of two fiber channel cards, which required a proprietary multiport driver to be added to the operating system of the unit.

The associated connections are visualized in Figure 7: Storage Array Network Interconnections.

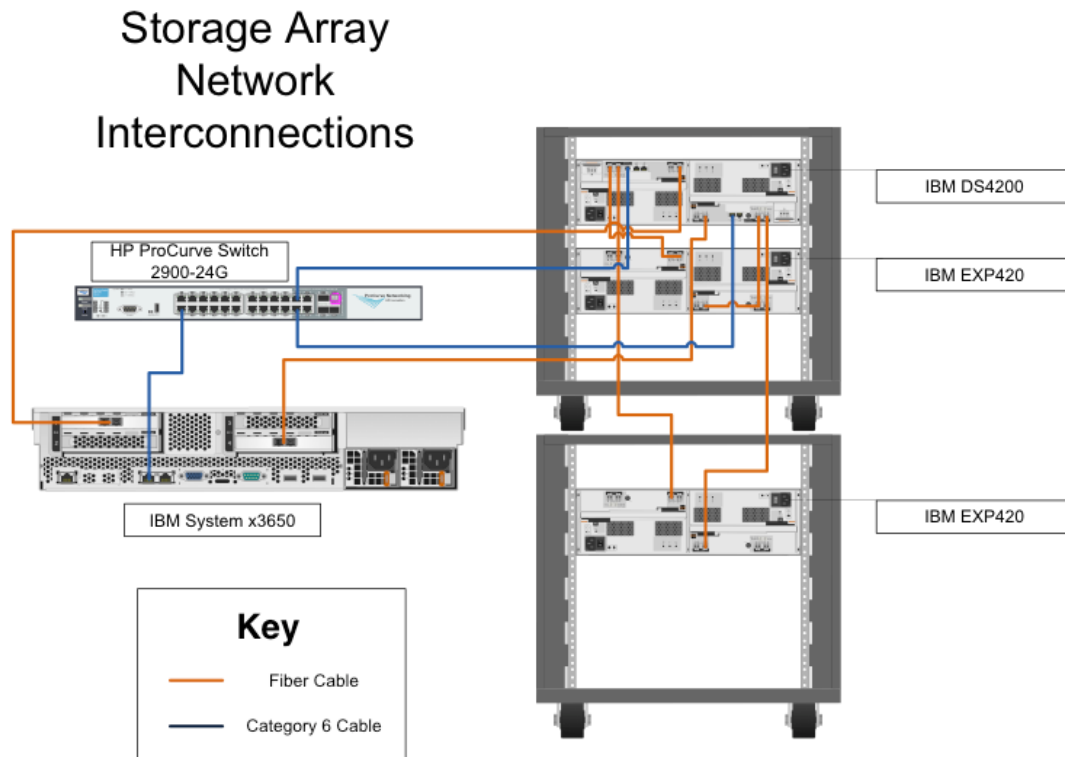


Figure 7: Storage Array Network Interconnections

## Operating System Installation and Configuration

In order to allow flexibility in server configuration, it was decided early in the Polar Grid project that the Linux operating system would be the primary foundation from which all other applications would run. The distribution chosen to install for this project was CentOS which was an open source version of RedHat Enterprise. As previously

stated the static IP addresses and hostnames were appropriated from the ECSU network administration staff prior to attempting the operating system installation process.

### *RAID Array Configuration*

In the case of Madogo (the head node), two disk arrays were configured to separate the operating system from the user “/home” directory data. This was done prior to beginning the installation from within the “RAID Array Utility” built into the x3650 node. The two arrays consisted of one 76GB RAID 1 array and one 400GB RAID 0 array. The RAID 1 array allowed the operating system to operate in a mirrored environment with two 76GB (68.2GB usable) drives. The second 409.5GB array stripped six 76GB drives to create the home directory area with no redundancy allotted. Compute units only contained a single 76GB drive to run the operating system and needed applications. This form of installation was called a “state-full” installation with each unit housing a hard drive with the operating system CentOS installed.

### *Operating System Initial Installation and Configuration*

The CentOS 5.4 Linux distribution was installed from DVD locally to each individual node through the built-in CD/DVD drive. The utilized generated media was created from an image download located on the vendor’s web site. No additional packages or additional software was added during the initial installation. In the case of the head node, allotments were made in the directory setup to have the “/home” and “/” directories located on the pre designated arrays. There was also an election to disable “SELinux” in order to reduce security complexity between the units. A common user

was also designated initially to aide in the installation administration of all nodes. The network port “eth1” was configured utilizing the predetermined static IP and hostname provided by the ECSU network administration team. Once all installation processes were complete, an operating system update was run to update all packages and patches on the cluster node.

### *Network File System Configuration*

As previously stated in Cluster Hardware Requirements, to house the complete 2008 Greenland Air data set 9TB of storage would be required. To accommodate the storage requirements the IBM RAID arrays had to be utilized. To prevent additional processing demands on the head node, it was decided a third server would be brought online to serve the data through a shared network file system (NFS). The second host was a second IBM System x3650 connected to the HP ProCurve switch via a 1Gb connection. The host unit required “nfs” and “portmap” services running in order to serve the data to the cluster nodes. Both the firewall and the “/etc/exports” configurations required additional information to allow the client cluster computers to connect. For network security reasons the exact configuration of the host unit will not be released; however, the process to create the nfs share in general is located in

Table 5: NFS Host General Configuration. With the host serving the storage with the flight data, the client nodes simply had to mount the shared directory.

Table 5: NFS Host General Configuration

<b>NFS Host General Configuration</b>
1. Determine the location of the requested directory

<ol style="list-style-type: none"> <li>2. *Edit the /etc/exports file to include the following <ol style="list-style-type: none"> <li>a. /&lt;dir-path&gt; host-ip/subnet (rw)</li> </ol> </li> <li>3. *Run the command “exportfs”</li> <li>4. *Edit the /etc/hosts.deny file to include the following <ol style="list-style-type: none"> <li>a. portmap:ALL</li> </ol> </li> <li>5. *Edit the /etc/hosts.allow file to include the following <ol style="list-style-type: none"> <li>a. portmap: host-ip/subnet</li> </ol> </li> <li>6. *Run the command “service nfs start”</li> <li>7. *Run the command “service portmap start”</li> </ol>
* Administrative rights required

## **MATLAB Distributed Computing Server Installation and Configuration**

Once the server operating system and network storage was configured, the installation of the MATLAB software as required by CSARP began. The application required the installation a network license manager to “lease” licenses to all nodes utilizing MATLAB (The MathWorks, 2005 - 2010). The chosen host was provided by the ECSU Network Administration team and had previously been utilized to lease MATLAB 2009b classroom concurrent licenses. In order to add to the serving pool of the license manager a license key had to be generated on the MathWorks license manager site. The key was specifically generated based on the host id (physical machine address) of the host machine. Once completed a concatenation of the previous “license.dat” file and the newly generated license key took place. The “Flexlm” service was then restarted within the services of the host machine. With that, the installation of the MATLAB Distributed Server v.4.2 began on the Madogo head node. During the installation it was found that the “libXp.so” library was missing from the machine. Though this did not hinder the completion of the installation process. Once completed the error was corrected with the

command “sudo yum install libXp” to add the deficient library. In progression, the MATLAB Distributed Computing Engine (MDCE), job manager and available workers were started on the head unit. During the initial start of the MDCE service, it was found that “/var/run/mdce”, “/var/log/mdce” and “/var/lib/mdce” had to allow write permissions to the current user in order to start. A “mdce status” was then executed to verify operation. Once completed, a similar process was followed on the compute nodes. In difference to the head unit, the clients utilized the license file from the “matlabroot/etc/license.dat” generated from the head node installation as opposed to the concatenated version. On the clients each worker was manually started with the remote host designated. In MATLAB syntax each processor core was referred to as a worker.

Figure 8: Starting Client Worker is an example of the syntax needed to add a single worker (clust5.worker1) to the madogo.ecsu.edu job manager.

```
[jeaimehp@clustn5 ~]$ ./startworker -name  
clust5.worker1 -remotehost madogo.ecsu.edu
```

Figure 8: Starting Client Worker

With the addition of workers, the “nodestatus” command was used to verify the additional worker resources were both connected and available to the job manager. With that, the configuration of the MATLAB Distributed Server was complete and operated as displayed in Figure 9: MATLAB Distributed Server Topography.

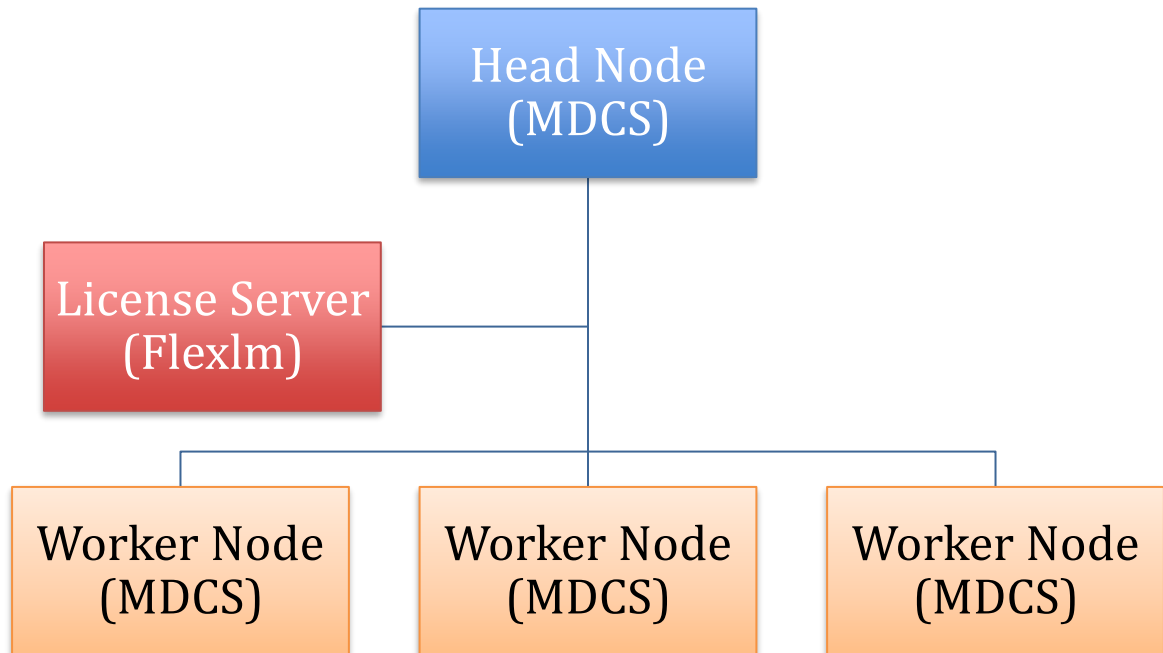


Figure 9: MATLAB Distributed Server Topography

## **CSARP Installation and Configuration**

### *CSARP Installation*

The installation of CSARP was performed in accordance with the informal instructions provided by the lead programmer William Blake of CReSIS (Blake, 2009). Per his instruction, two main code components were identified. The two components were the CReSIS Toolbox and the CSARP source code. The toolbox added proprietary scripts which CReSIS created in support of the MCRDS radar system. The installation of

CSARP consisted of extracting the CReSIS toolbox into a users home directory. Additionally CSARP was expanded into the users home directory.

### *CSARP Configuration*

Within the expanded CReSIS Toolbox a “startup.m” script file required editing to identify the current location of the CReSIS Toolbox on the file-system as seen in Figure 10 Startup.m CReSIS Toolbox configuration. In the script file the variable “isunix”

```
if isunix
    ct_path = '/home/jeaimehp/CReSIS-Toolbox.r16';
```

Figure 10 Startup.m CReSIS Toolbox configuration

identified any Linux based or UNIX based operating system. The startup script needed to be run prior to the execution of any CSARP code to insure the CReSIS Toolbox scripts were identified in the MATLAB path. The next step in configuration required the parameter file (i.e. ParamMCRDS\_year\_type.m) to be altered in order to define the input and output storage locations. This parameter file was located deep within the file system of the CSARP code file directory (/home/jeaimehp/GOAP.r201/code/Params/MCRDS/2008\_Air). The functions of these variables are located in

Table 7: Cluster setup section

The variables “start\_file” and “stop\_file” numerically allowed the number of files selected for processing to be defined and adjusted as needed. In the case of this study



files 100 through 115 were chosen to give a static pool of 16 files with individual sizes of 211MB each, totaling 3.376GB of data. The last configuration change from within the parameters file required the ECSU MDS job manager to be identified. These changes from within the “Cluster Setup” section are identified within

Table 6: Parameter file variable functions

<b>CSARP Parameter File Modified Variables List</b>		
<b>Variable Name</b>	<b>Function</b>	<b>Description</b>
raw_dir	Input	The raw MCRDS data files
cff_dir	Input	CRISIS file format data directory
proc_dir	Input	Processed data directory
gps_name	Input	Name of GPS data file for the raw radar data
param.raw_dir	Input	Directory where raw MCRDS data files are located
param.cff_dir	Output	Directory for CRISIS formatted files
param.proc_dir	Output	Directory where processed files are located
param.pos_dir	Output	Directory of GPS data files

Table 7: Cluster setup section

<b>CSARP Parameter File Modified Variables List – Job Scheduler</b>	
<b>Variable Name</b>	<b>Description</b>
param.sched_type	Supported scheduler application in use (i.e. MathWorks Job Manager, TORQUE, Platform LSF)
param.sched_url	Qualified hostname and domain of the head node running MDS
param.sched_name	Name given to the MathWorks Job Manager when initiated

### *MATLAB Job Scheduler*

It was noted that the variables “param.sched\_type”, “param.sched\_url” and “param.sched\_name” actually acted as attributes to a MATLAB built-in function entitled “findResource.” This findResource function was used to identify the type of scheduler and to create objects representing the scheduler in the local MATLAB client session (The

```
set(job, 'MaximumNumberOfWorkers', Inf);
```

Figure 11: "Stage" file line to designate the number of workers

MathWorks). With these parameter variables the option to utilize a scheduler other than the MathWorks Job Manger would be possible. In order to limit the number of workers in which CSARP could be distributed, it was described that within the individual “Stage” files the number of workers could be designated. Figure 11: "Stage" file line to designate the number of workers displays the default state in which the variable “Inf” indicated the maximum number of available workers that could be used. By replacing “Inf” with an Arabic number, worker utilization was defined.

## **Running CSARP Jobs**

### *Initial testing on the Polar Grid Unit at Indiana University*

The method to run a CSARP job required the use of the MATLAB graphical user interface through a display forwarding connection. Testing of this workflow was completed on Indiana University’s Polar Grid Unit (PGU). The IU University Information Technology Services (UITs) provided user credentials for access through a ssh connection. This cluster was previously configured to run CSARP and could therefore be utilized to test the CSARP workflow. The PGU CSARP workflow utilized

two major components, the IU data capacitor for the wide area network (DCWAN) and the blade center server (BCS). The DCWAN was used to store both raw, and processed data. In particular, the DCWAN was designed for researchers at institutions other than IU to store data to be utilized on a cluster. What made the DCWAN usable was the amount bandwidth, which directly connected to the BCS. According to the IU Knowledge Base, the DCWAN contained 34 servers with up to 10Gb Ethernet connections to the network serving a total of 678TB through a Lustre file system ("At IU, what is the Data Capacitor?,"). The head node (PGU) of the BCS was an IBM x3550 with dual Quad-core Intel Xeon 2.33GHZ processors with 16 GB of RAM. Connected to the head unit through a bonded 5Gb connection were six IBM HS21 blades (pg1 – pg6) within an IBM BladeCenter S chassis (Shields, 2010).

The workflow to run a CSARP job began with initiating a MATLAB session from within the folder that contained the configured parameter file. Once opened through an X window session, the startup.m script was run to set the path to the CReSIS Toolbox directory in the user home directory. Once completed the parameter file was opened and altered to only allow 16 raw files to be processed. This step was done due to the fact that only 16 workers were made available to the MDS Job Manager. The stage files were not altered initially to insure CSARP functioned. The parameter file was then executed without errors. The output from the initial test displayed three time components in relation to the completion of each stage along with the output of “dat” files as the final product. From previous conversations with the head programmer at CReSIS it was suggested that each of these times should be compared there by displaying time differences in high I/O stages verses compute intense stages. This paradigm was apparent

in the relatively fast “Stage1a” verses the much slower “Stage2.” After the initial test, the stages were altered to only allow eight workers to be utilized. The CSARP workflow was repeated, and completed without error. From this initial testing the CSARP workflow became set the number of workers in the stage files, run startup.m, run the parameters file, and save the output times as shown in Figure 12: Initial CSARP Workflow.

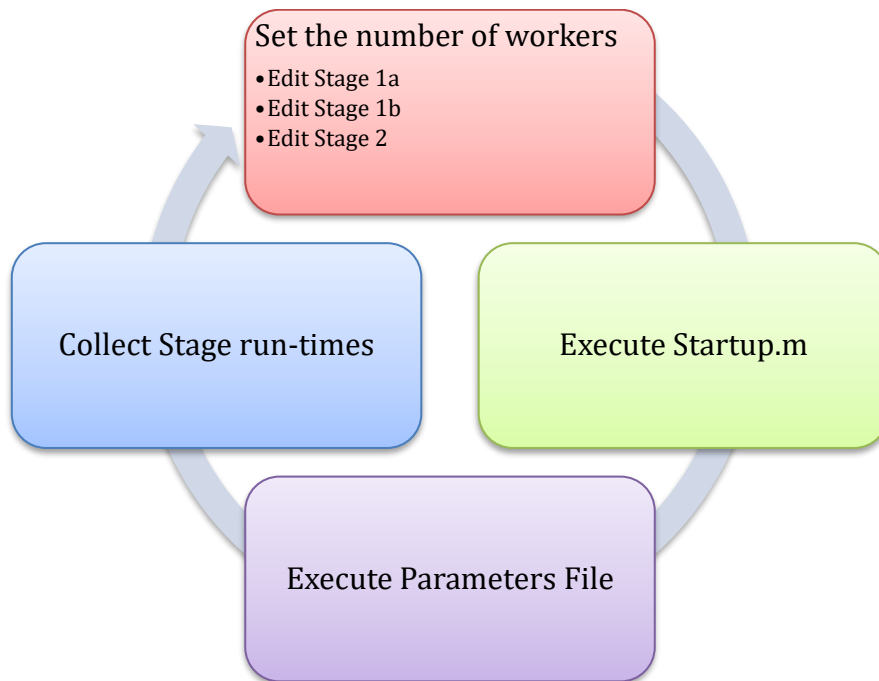


Figure 12: Initial CSARP Workflow

The next change in the CSARP workflow was to remove the GUI component from the running of CSARP. Prior conducted research suggested that CReSIS flight data could be plotted in using MATLAB through a command line interface (Burney & Evans, 2009). Through the command line the GUI process was duplicated as seen in Figure 13: CSARP command line execution. By concatenating the command execution through

semicolons multiple runs of the code could be executed. One change in the stage1b code was made to elevate the need for user confirmation to remove previous CReSIS formatted files thereby making the CSARP execution fully automated once initiated.

```
$ matlab -nodesktop -nosplash -r "startup; paramecs; quit"
```

Figure 13: CSARP command line execution

### *PGU Sample Worker Testing*

Initial testing focused on running CSARP with worker variation at 1, 2, 4, 8, 12 and 16 levels to collect a set of sample performance data. Each run utilized the BCS as the compute cluster and the DCWAN as the storage component. All worker variations were run three times to provide a mean run time for each section. Though not analyzed statically nor used as a metric, a linear trend of reduced processing times with increased workers was visually observed when the mean run times were plotted.

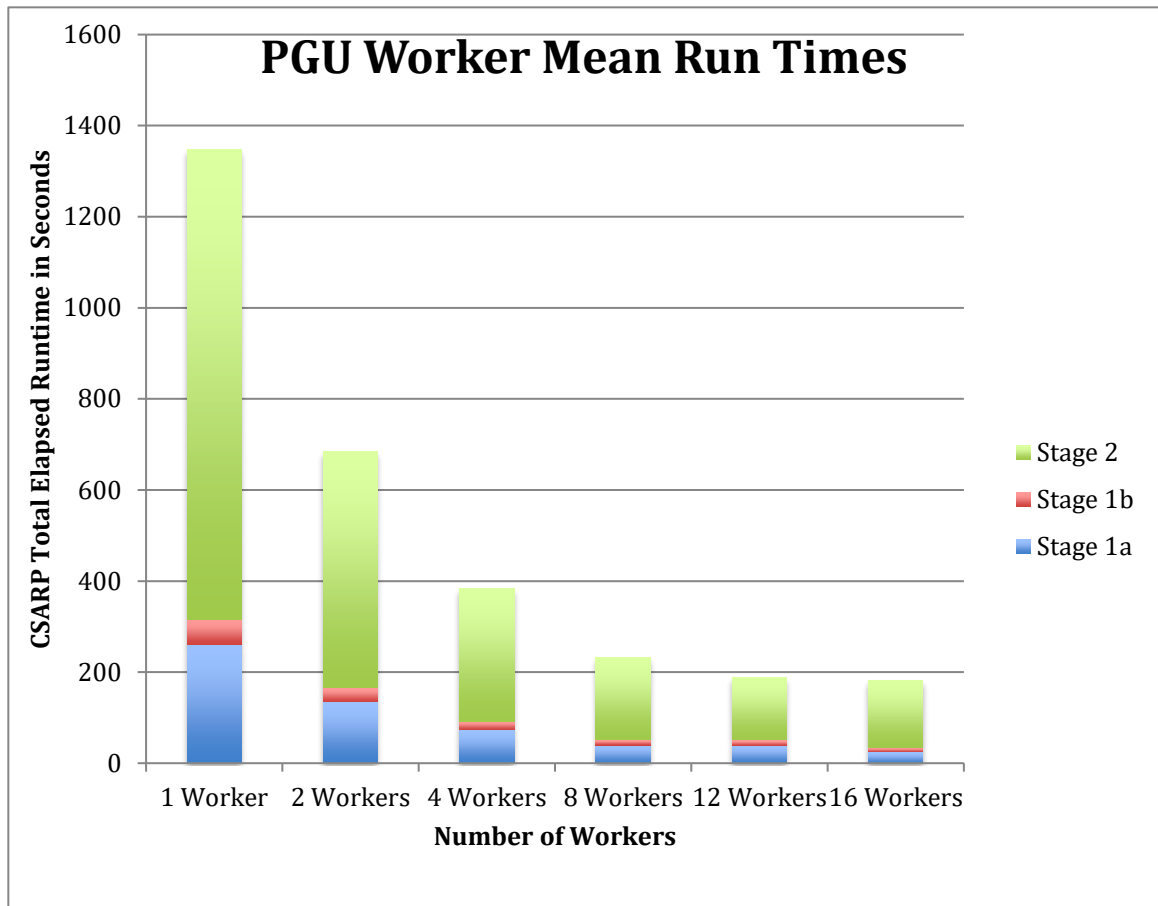
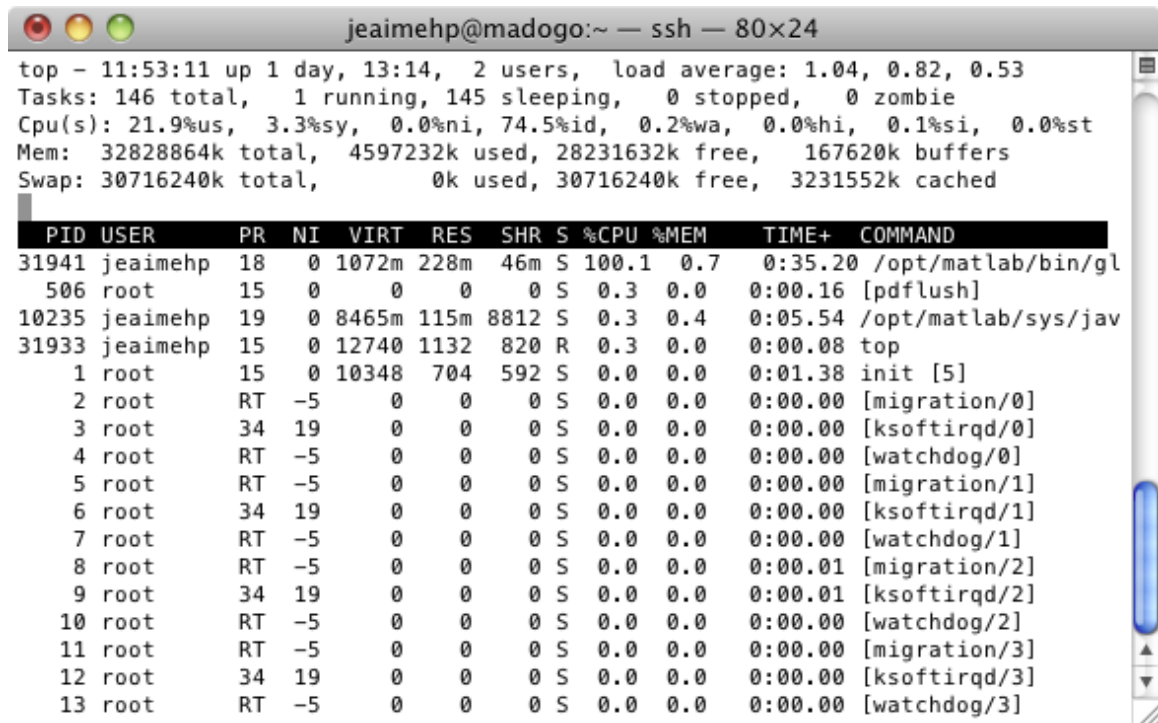


Figure 14: PGU Worker Mean Run Times

### Madogo Job Manager Configuration

The data collection procedure on Madogo began with the mounting of the RAID array to the /PGData location on the client. This was done on each of the cluster compute nodes and head node. Once the mount was completed the mdce service was started and job manager initiated. Initially all eight available cores on the head unit were assigned as workers. Per provenience data collected from the PGU workflow testing, the same 16 files from the 2008 Greenland data set were chosen as the static pool for testing. The

initial execution of CSARP on Madogo duplicated an issue previously seen in the field. During the execution of the job the head unit froze due to insufficient computing resources (Powell, 2008). Due to the fact that Madogo had 48GB of memory installed when compared to the original Polar Grid C01 cluster's 16GB, the issue of memory limiting the processing capability of the cluster was again recognized as a possible concern. As seen in Figure 15: Top CPU display during CSARP execution the MATLAB application used 100.1% of the CPU while the memory use was only .7% of the system resources.



```

top - 11:53:11 up 1 day, 13:14,  2 users,  load average: 1.04, 0.82, 0.53
Tasks: 146 total,   1 running, 145 sleeping,   0 stopped,   0 zombie
Cpu(s): 21.9%us,  3.3%sy,  0.0%ni, 74.5%id,  0.2%wa,  0.0%hi,  0.1%si,  0.0%st
Mem:  32828864k total, 4597232k used, 28231632k free,  167620k buffers
Swap: 30716240k total,   0k used, 30716240k free,  3231552k cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
31941	jeaimehp	18	0	1072m	228m	46m	S	100.1	0.7	0:35.20	/opt/matlab/bin/gl
506	root	15	0	0	0	0	S	0.3	0.0	0:00.16	[pdflush]
10235	jeaimehp	19	0	8465m	115m	8812	S	0.3	0.4	0:05.54	/opt/matlab/sys/jav
31933	jeaimehp	15	0	12740	1132	820	R	0.3	0.0	0:00.08	top
1	root	15	0	10348	704	592	S	0.0	0.0	0:01.38	init [5]
2	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	[migration/0]
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	[ksoftirqd/0]
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	[watchdog/0]
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	[migration/1]
6	root	34	19	0	0	0	S	0.0	0.0	0:00.00	[ksoftirqd/1]
7	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	[watchdog/1]
8	root	RT	-5	0	0	0	S	0.0	0.0	0:00.01	[migration/2]
9	root	34	19	0	0	0	S	0.0	0.0	0:00.01	[ksoftirqd/2]
10	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	[watchdog/2]
11	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	[migration/3]
12	root	34	19	0	0	0	S	0.0	0.0	0:00.00	[ksoftirqd/3]
13	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	[watchdog/3]

Figure 15: Top CPU display during CSARP execution

To correct the system freezing issue the number of workers was reduced to six. CSARP was restarted and functioned without further resource incidents. This created an issue in

reference to the 32 cores with four units topography. To correct this an additional IBM System x3550 was added to the cluster for a total count of four x3550 compute nodes and two x3560s. The two x3650s served as the network file server and head node respectively. To distribute the number of Job Manager workers, two of the compute nodes initiated seven workers and two initiated six workers. In addition to the six workers from the head node a total of 32 workers were initiated over the six nodes. The final cluster topography was as shown in Figure 16: Final Madogo cluster topography. With the final topography solidified the mdce services were started and workers added to the Madogo job manager on each of the compute units.

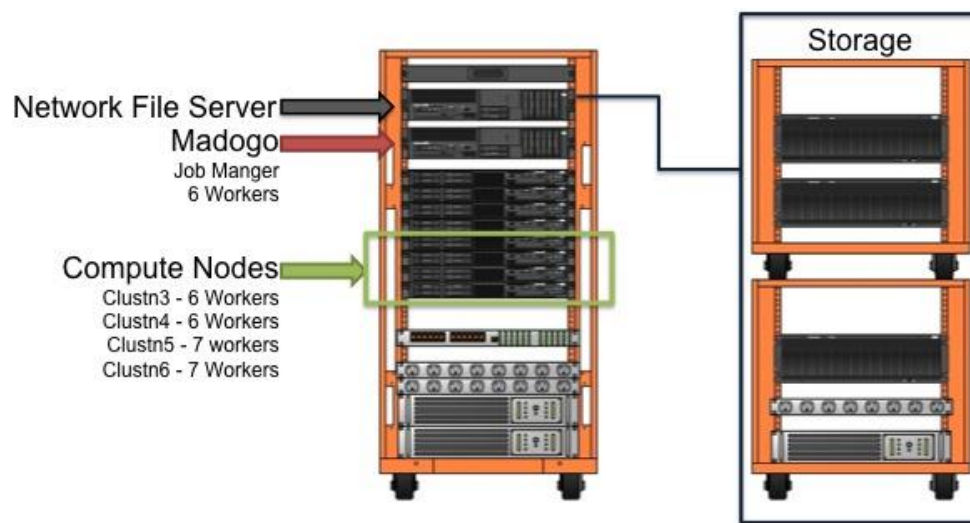


Figure 16: Final Madogo cluster topography

### Data Population Definition and Analysis

The selected raw data samples were collected August 1, 2008, which originated from calibration data of the MCRDS radar. Each data set consisted of setting the number



of usable workers through the stage files, and then running the parameter file with 16 files specified for each of the three trials. Each trial contained three enumerated run times in seconds per the CSARP application. There were 16 files used during each PGU test and this was continued for the Madogo cluster tests. Times for Stage1a (conversion from RAW to CFF process), Stage1b (adding height data to the CFF) and Stage2 (SAR Processing into dat files) were collected. The experiment was repeated three times for 1, 2, 4, 8, 16 and 32 nodes with accompanying times being collected after each run.

#### *Statistical Methods and Tests Used to Analyze the Data*

The research question posed whether within a 5% level of significance that the addition of computing cores increased the performance of the CSARP algorithm. To complete the statistical testing, the means of all worker trials were compared using an analysis of variance (ANOVA). The ANOVA test was a formal test to find the probability of a mean occurring (Fisher, 1925)..ANOVA checks the variability both between groups of data and within groups of data. If the attained p-value was less than the 5% level of significance, it would have indicated there was a significant difference. If greater, it would have indicated that there was no significant difference.

## **CHAPTER IV: THE RESULTS**

### **What MATLAB toolkits and/or expansion kits are necessary to run CSARP?**

In order to run CSARP additional functions only found in specific toolboxes and expansions had to be added to the computers that were utilized. Though un-documented within the included code comments or help files, the required toolboxes for CSARP were MATLAB 2009b, the Signal Processing Toolbox, the Parallel Computing Toolbox, the MATLAB compiler and the CReSIS Toolbox. Recommended toolboxes also included the MATLAB Distributed Server in order to use greater than eight workers. The code designer provided this information informally.

### **What hardware requirements are necessary to store and process CReSIS collected data?**

For the 2008 Greenland Air Calibration data collection of August 1<sup>st</sup>, there were a total of 279 raw files with an average file size of 211MB. The sum of the files totaled 59GB for that single flight collection of data. For the entire 2008 Greenland field season there were 40 flight collections with a total of 9.9TB of data needed to store the samples. To expand this out over the five-year term of the grant, a total of 50Tb of data would be needed for Greenland deployments alone.

The computing hardware requirements as needed by CSARP were primarily focused on the needs of MATLAB and the MATLAB Distributed Server. The MathWorks specified minimum requirements were a processor of Intel Pentium 4 class or

above, 500MB of storage, and 1GB of RAM. The 2008 field-work had determined that a minimum of 16GB of memory with four workers was necessary to run CSARP with multiple cores successfully. In order to fully utilize that amount of RAM a 64-bit version of the CentOS operating system and a 64-bit version of MATLAB was necessary. This added to the hardware requirements a processor capable of 64-bit processing and a minimum of 2GB of RAM. Though CentOS was chosen for the project, any supported 64-bit version of Windows, Macintosh, or Linux as prescribed by MathWorks would be suitable for CSARP.

In order to expand beyond the supported eight cores supported within the MATLAB Parallel computing toolkit, the MATLAB Distributed Computing Server (MDS) was needed with a network to interconnect the multiple computers. Though not specifically defined by the MDS, the network bandwidth amount became the bottle neck of the project. This was highlighted in the reduced performance of ECSU's Madogo cluster verses the tested speeds seen on IU's PolarGrid Blade Center System (BCS). Madogo contained a 1Gb network connection whereas the BCS contained 10Gb connections to both the storage and compute nodes. As seen in Figure 17: ECSU's Madogo verses IU's BCS mean run times, as the number of workers increased, the performance increase, wanes in relation to the two servers. The additional network bandwidth appeared to have played a key role in the relationship though this was not in the scope of this project.

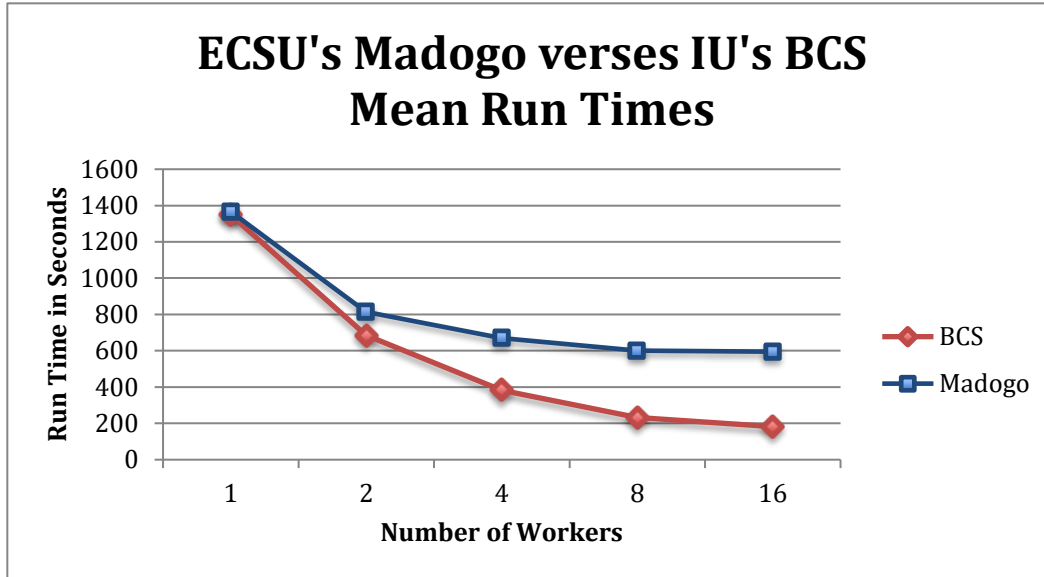


Figure 17: ECSU's Madogo verses IU's BCS mean run times

**What facility environmental requirements are there to house a cluster of at least 32 cores to process a data set?**

In order to house the 32-core cluster there were power and cooling requirements that were a segment of the entire Madogo cluster. The final configuration of the cluster required two IBM System x3650's, four IBM System x3550's, one IBM DS4200 RAID head node, two IBM DS420EXP expansion enclosures, one Tripp-Lite KVM console, and a HP 24-port ProCurve Switch. The power required by each component added together was a total of 9,705.80W or 9.7kW. Similarly the heat output was calculated to be 33,096.78 BTU / hr for the cluster requiring a minimum of 2.7 tons of cooling.

### **What is the process to prepare a cluster from a middle-ware stand-point?**

Broadly speaking, the middleware for a cluster is a job scheduler. Through the course of this study this question was narrowed to focus on the needs of CSARP on a cluster. In this function this question becomes how to prepare a cluster for the implementation of MATLAB Distributed Server's Job Manager. In this respect there were several steps to complete the task. The first was the installation of the license server. In the case of this project the licenses server had been previously configured for the purpose of leasing MATLAB concurrent licenses on the ECSU campus. With the support of the network services manager of ECSU, the license file from the configured server was concatenated with the license file for MDS. Both licensees were generated on the MathWorks license manager site and required the Host ID (machine address) of the licensing machine. This was not a necessary step in that the MDS host machine could have also served as the license manager and was a part of the normal installation procedure. Specifically this was done to not have dual MathWorks Flex License Managers on the ECSU campus network.

The second step of the middle-ware installation was the actual MDS installation on the head node. Aside from using the newly concatenated license file from the license manager, there were no customizations for the cluster. The end of the installation did produce an error announcing the “libXp.so” library was not found and that MATLAB would not function correctly until the library was added. The requested library provided public application interfaces that allowed client applications to render to non-display devices. This issue was resolved by performing a yum install for the libXp library. To test the installation, MATLAB was executed to produce the application interface.

The third step in the setup process required the MATLAB Distributed Computing Environment (MDCE) to start. This process required several folder permissions to allow write access to the user. This had to be performed due to the fact that the MDCE was written to deny the root user to start the service, which would forgo such requirements. Specifically the “/var/run/mdce”, “/var/log/mdce” and “/var/lib/mdce” folders required write permissions added. Additionally the MDCE service was added to the startup of the Madogo server though this was not required.

The fourth MDS installation step required the Job Manager and workers to be added. The primary challenge with this step was the inclusion of other client machines. The MDS with the license file generated on the head unit was utilized for this process. The workers were then individually added to the job manager both locally on the head node and remotely on the clients. Once complete the “nodestatus” command confirmed operability.

### **Can an open-source job scheduler replace the MATLAB proprietary Distributed Computing Server currently required by CSARP?**

It was found during CSARP parameter file configuration that there was a variable entitled “param.sched\_type.” Upon further research it was found that the MDS does support third-party schedulers natively. The supported third party schedulers were Platform LSF (Load Sharing Facility), Microsoft Windows HPC Server (including CCS), PBS Pro, and TORQUE schedulers. Specifications for unsupported schedulers such as Condor were also discussed in the MDS documentation. Through this research path it was also discovered that the MDCE portion of the MDS would still be necessary on all

client machines. This information there by solidified the need for MDS to run CSARP in MATLAB script form even if a different job scheduler was selected. The only way to alleviate that dependency was through the creation of a CSARP executable binary. CSARP binary creation in the current form would not be possible due to the need for coded script changes in order to process data.

**Does this study prove within a 5% level of significance that the addition of computing cores increases the performance of the CSARP algorithm**

#### *Data Set*

Once complete, the CSARP run generated output to both the display and the file system. The resulting file system output was in the form of “dat” files. The dat files were in a format that could then be utilized within filters or other visualization software. As stated in the Experimental Delimiters section, the process to create these visualizations will not be covered, however an example is shown in Figure 18: Image generated from 20080715B data.

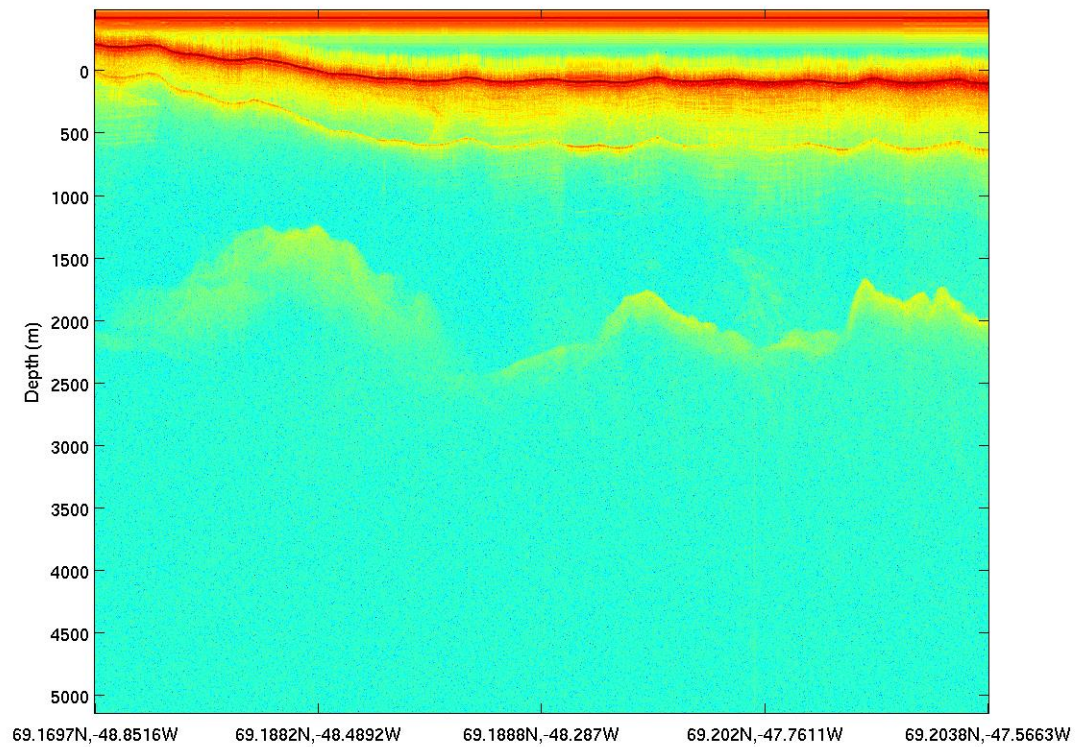


Figure 18: Image generated from 20080715B data

Collected for testing was the display output. An example of the collected data is displayed in Figure 19: 1 Worker Job Display. Beneath each stage completion was an elapsed time, which was saved and inserted into a spreadsheet for analysis.



```

<MATLAB(R)>
Copyright 1984-2009 The MathWorks, Inc.
Version 7.9.0.529 (R2009b) 64-bit (linux64)
August 12, 2009

This is a Classroom License for instructional use only.
Research and commercial use is prohibited.

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

Adding crisis toolbox to path at: /home/jeanmehp/CReStS-Toolbox.r16
Setting crisis_toolbox environment variable: /home/jeanmehp/CReStS-Toolbox.r16
Adding crisis toolbox to path at: /home/jeanmehp/CReStS-Toolbox.r16
Setting crisis_toolbox environment variable: /home/jeanmehp/CReStS-Toolbox.r16

Scheduling Creating CFFs
Queued Stage 1a Processing
Stage 1a Complete

Stage 1a ← Elapsed time is 257.165021 seconds.
            Getting Header and Position Filenames
            Elapsed time is 0.015393 seconds.
            Creating GetHeights Jobs
            Queued Stage 1b Processing
            Stage 1b Complete

Stage 1b ← Elapsed time is 132.404736 seconds.
            Files already exist in this folder. Would you like to delete them?: n
            Getting Header and Position Filenames
            Elapsed time is 0.014707 seconds.
            Getting Data Filenames
            Elapsed time is 0.051228 seconds.
            Creating Chunks of data to process
            Elapsed time is 0.832706 seconds.
            Creating Processing Jobs
            Elapsed time is 1.165571 seconds.
            Queued Stage 2 Processing
            11 | 1243
            22 | 1076
            28 | 997
            39 | 843
            50 | 686
            61 | 534
            72 | 381
            78 | 305
            89 | 152
            100 | 0
            Stage 2 Complete

Stage 2 ← Elapsed time is 1370.436145 seconds.

```

Figure 19: 1 Worker Job Display

Whereas the total elapsed time was comprised of the sum of the displayed times from the stages, Figure 20: Worker mean total run times as derived from stage times shows the recorded mean times of all the workers.

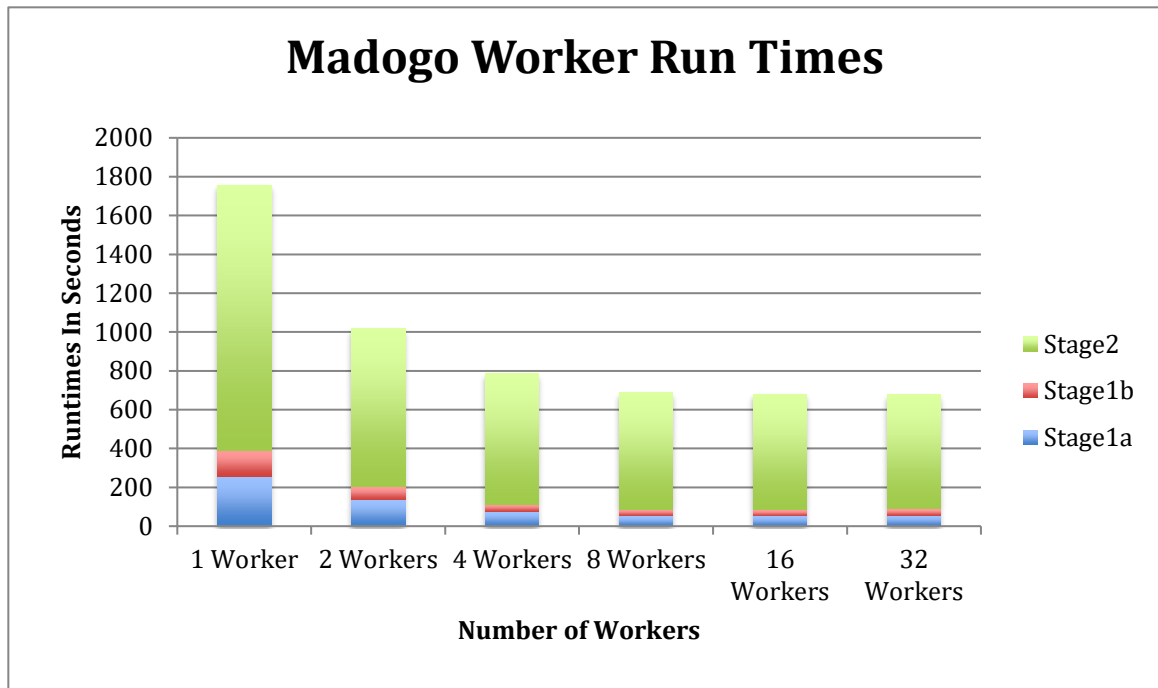


Figure 20: Worker mean total run times as derived from stage times

### *Statistical Hypothesis Testing*

In order to prove or disprove the hypothesis that increasing the number of workers also increased CSARP performance as enumerated by time, an ANOVA test was performed. In accordance with statistical testing there were six steps that were followed. Step one defined both the parameters and statistical hypotheses. The statistical hypotheses were completely separate from the hypotheses related to the research in that they were determined according to ANOVA. The second step defined the level of significance that was an indicator of probability. The third step both decided the test statistic as well as performed the test. The fourth step defined the critical value and the fifth step allowed the decision to accept or reject the statistical hypothesis based upon the

critical value. The sixth and final step contained the conclusion, which restated the decision in more analytic terms.

*Step 1: Parameters and Statistical Hypothesis*

The statistical hypothesis of  $H_0$  was that all worker mean times ( $\mu_1 - \mu_{32}$ ) were equal.  $H_1$  was set that some mean was not equal.

$$H_0 : \mu_1 = \mu_2 = \mu_4 = \mu_8 = \mu_{16} = \mu_{32}$$

$$H_1 : \mu_1 \neq \mu_2 \neq \mu_4 \neq \mu_8 \neq \mu_{16} \neq \mu_{32}$$

Equation 2: ANOVA Hypothesis and Level of Significance

The worker times as collected from the CSARP testing were as listed in Table 8: Worker Times in Seconds.

Table 8: Worker Times in Seconds

<b>Madogo Worker Times</b>					
<b>1 Worker</b>	<b>2 Workers</b>	<b>4 Workers</b>	<b>8 Workers</b>	<b>16 Workers</b>	<b>32 Workers</b>
1760.01	1014.35	783.978	687.050	674.692	675.347
1753.65	1014.74	784.407	683.035	678.568	680.911
1756.54	1018.20	783.482	691.444	688.630	678.669

*Step 2: Definition of the Level of Significance*

Whereas the research question specified a 5% level of significance, the probability was defined. Mathematically speaking, the level of significance subtracted

from 100% yielded the confidence level of 95%, which was more appropriate for an ANOVA within applications.

### *Step 3: Test Statistic and ANOVA Testing*

The ANOVA function was performed through the MiniTab Statistical software package as a One-Way Un-stacked variation. The one-way test statistic was selected due to the fact that only the runtime variable was utilized as well as the data was independent. If another metric such as memory use had also been collected, then a two-way ANOVA would have been appropriate. In this function each column of worker times were seen as the responses with 90 entered as the confidence level. The partial result of the ANOVA is listed in Table 9: One-way ANOVA p-value results.

Table 9: One-way ANOVA p-value results

<b>One-way ANOVA: 1 Worker, 2 Workers, 4 Workers, 8 Workers, 16 Workers, ...</b>					
<b>Source</b>	<b>DF</b>	<b>SS</b>	<b>MS</b>	<b>F</b>	<b>P</b>
<b>Factor</b>	5	2689755	537951	35048.57	0.000
<b>Error</b>	12	184	15		
<b>Total</b>	17	2689939			

### *Step 4: Critical Value*

The critical value for a hypothesis test was a threshold to which the value of the test statistic in a sample was compared to determine whether or not the null hypothesis was rejected. In the case of ANOVA the “P-value” was the critical value. The p-value generated was 0.000. In statistical hypothesis testing, the p-value is the probability of

obtaining a test statistic at least as extreme as the one that was actually observed (Davidson & MacKinno, 1993).

#### *Step 5: Decision*

The decision phase compares the critical value to the test value. This decision will either reject or accept the statistical null hypothesis of there being no significant change. In this experiment the null hypothesis ( $H_0$ ) of all the means being equal was rejected. Simply, the p-value 0.000 was less than the test value .05 (5%) therefore the null hypothesis was rejected.

#### *Step 6: Conclusion*

There was significant evidence to indicate there was a difference in the performance times of CSARP due to the inclusion of additional workers within a 5% level of significance. Moreover the run time difference from ~30 minutes with one worker to ~10 minutes with 32 workers constituted an ~67% increase in performance.

## CHAPTER V: DISCUSSION

The purpose of this paper was to answer the question of whether the addition of computing cores would increase the performance of the CSARP algorithm processing within a 5% level of significance. Several sub-questions were also addressed including the hardware requirements necessary to store and process CReSIS collected data, facility environmental requirements for a 32-core cluster, processes necessary to prepare a cluster from a middleware-stand-point, MATLAB toolkits necessary to run CSARP, and lastly if an open-source job scheduler could replace the MathWorks proprietary MATLAB Distributed Computing Server as required by CSARP.

The methodology of the paper followed the process of first defining the needs of CSARP including all supporting software and hardware. Once defined the needs of the cluster computing hardware was addressed within the location housing the cluster. Once completed the cluster computer roles were defined and the operating system was both installed and configured. Next the MATLAB Distributed Server (MDS) components including the Flex License Manager, MATLAB Distributed Computing Environment (MDCE), MDS Job Manager, and workers were installed and configured within the cluster. At this point the CSARP code was installed and tested. Data collection then began with the CSARP algorithm being tested and repeated three times for 1, 2, 4, 8, 16, and 32 workers. Lastly the collected performance means were statistically tested through an analysis of variance (ANOVA) within a significance level of 5%.

Through the installation process it was noted what hardware and environmental needs were necessary to house a 32-core cluster capable of running CSARP including all additional toolboxes and MATLAB expansions. Through this line of research it was also

found that though the Job Manager could be replaced with a third-party job scheduler, the change would be mute without the capability to go above the eight workers supported without the MDCE service. The final results of the ANOVA testing found that within a 5% level of significance, there was enough evidence to suggest that there was an increase in performance with the addition of workers. In total there was a calculated ~67% increase in performance between 1 worker and 32 workers.

What the results of the experiment presented was the distribution of CSARP performance benefits warrant the investment of further research into code refinement. The reason for that statement was due to an apparent limit of performance increases in the resulting graphs generated by the means.

### **Further Worker Performance Results**

Once it was statistically evident that additional workers did improve CSARP performance, the run times were viewed to determine if there was a definable plateau of performance increase. In grid terminology, was there a point at which the network overhead would begin to increase the time it took CSARP to complete? To perform this extrapolation the means of the worker times would be utilized in combination with finding the equation of the accompanying curve. As shown in Figure 21: Madogo worker mean run times graph, the run times reduced as the number of workers increased; however, it was also noted that as the number of workers increased the amount of run time did not decrease linearly. For this fact a simple regression could not be utilized to find the equation of a line in order to locate a point of inflection.

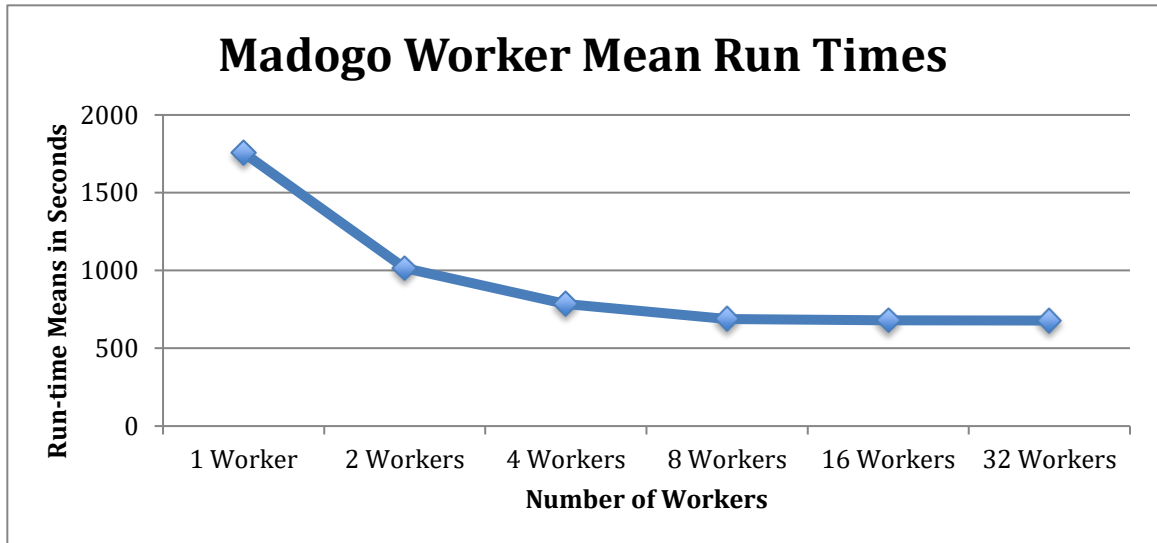


Figure 21: Madogo worker mean run times graph

To generate an equation for the curve, MATLAB's "Better Fit" built in function was utilized on the graph of the mean worker times verses the number of workers. As seen in Figure 22: MATLAB "Better Fit" equations neither linear, quadratic, nor cubic equations generated a viable profile of the curve.



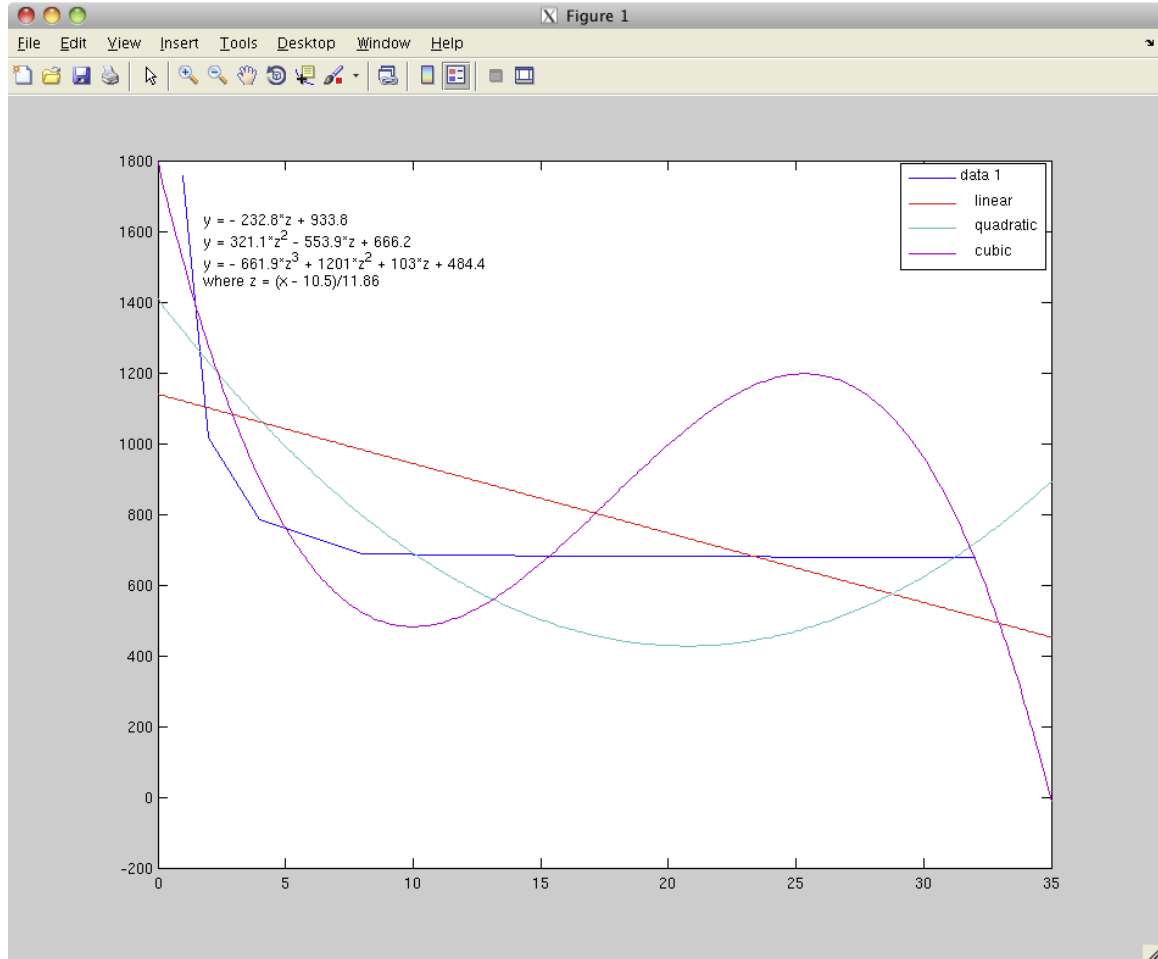


Figure 22: MATLAB "Better Fit" equations

The more viable recreation of the curve utilized the shape-preserving interpolant. As seen in Figure 23: Shape-preserving interpolant

Figure 23: Shape-preserving interpolant, a profile of the data was generated though no true equation could be derived.

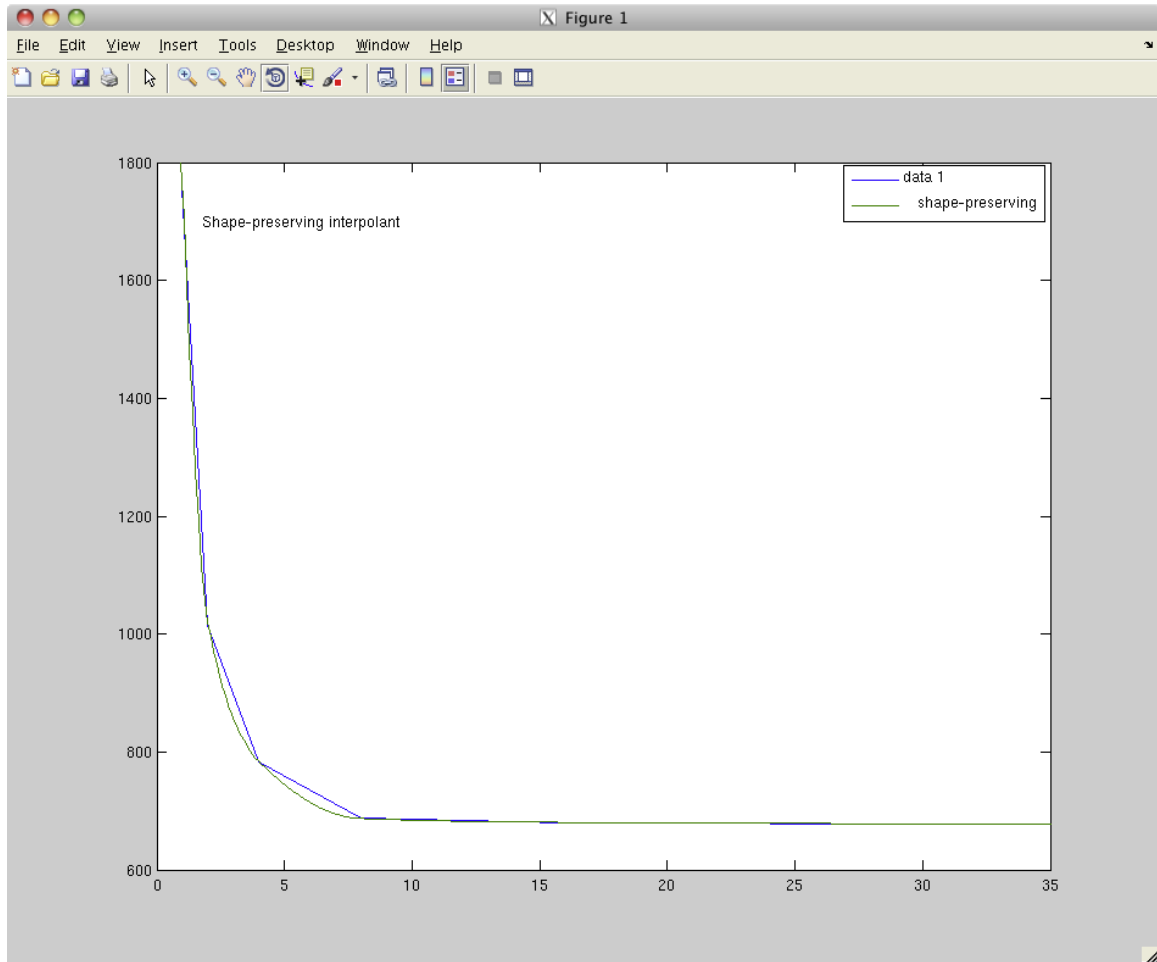


Figure 23: Shape-preserving interpolant

With the inclusion of the interpolant, an estimated expansion of the model was performed with 128 workers. As seen in Figure 24: Interpolant with 128 worker estimate, the overhead of the network decreased the performance to ~854 seconds or a performance increase of only ~30% where as 32 workers had an increase of ~67%. The point of inflection appeared to occur at the 50 worker marker, which illuminated a theoretical limit of benefits relating to CSARP job distribution on Madogo. The immediate solution to this concern would be an increase in available network bandwidth; however, without

CSARP code refinement the network adjustment would yet yield another performance plateau.

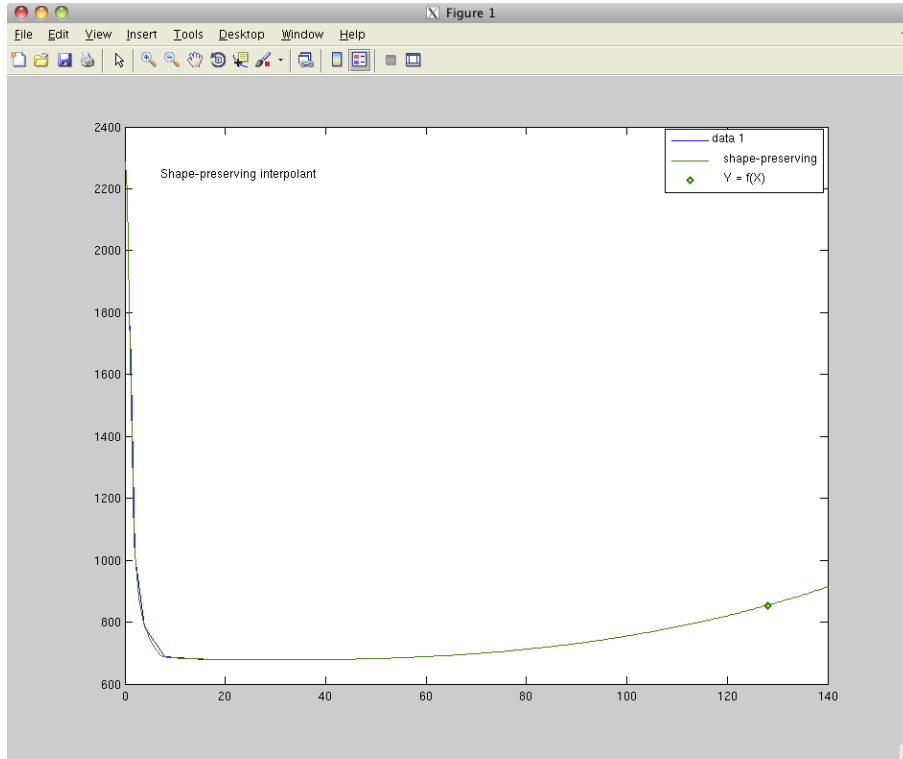


Figure 24: Interpolant with 128 worker estimate

## CSARP Recommendations

Through the progression of the project several issues within CSARP were noted as points of interest for future work. One example on possible study lies within the I/O performance of the CSARP application within MATLAB. This project did not account for either memory or CPU utilization in performance testing. The distribution of workers on nodes, bandwidth increases, memory adjustments, different hardware configurations, and compiled code verses scripts should all be explored as possible areas of

improvement. One area of deficiency within the CSARP algorithm was that of documentation. The revision of CSARP utilized for this project was 201. Within the CSARP code there were very few comments; further there were no written instructions or references to software prerequisites. Without the aide of the CReSIS lead designer an understanding of the software could not have been reached. There was also a vast amount of code that was no longer active within the scripting files. One example of this was the `raw_dir`, `cff_dir`, `proc_dir` and `gps_name` lines within the parameters file. It was discovered that these lines were added for a specific set of testing and were never removed though no commenting was left to make users aware. Another glaring example of limited documentation was the issue revolving what CSARP actually did within stages or what was created as an output? Initial testing of CSARP revision 16 in the field-produced echogram images where as the 201 revision simply produced dat files. For the future expansion of CSARP into other schools or even businesses these issues should be corrected prior to the transition of CSARP to a stand-alone commercial application.

## **Future Work**

Though listed within this paper, testing of CSARP conversion to a third party job scheduler would be a logical step with time trials to determine if a specialized scheduler would outperform the MATLAB Job Manger. The possible conversion of CSARP to a lower level language such as C++ or JAVA may increase the performance capabilities of CSARP by removing the overhead created by MATLAB. This would also remove the requirement of the MATLAB Distributed Computing Environment. This would lower implementation costs for CSARP associated with the purchase and operation a cluster.

Lastly this would provide greater access to other grid system such as TeraGrid for expanded computing ability.

## REFERENCES

- Allen, C. (2006, September 26th, 2008). A Brief History Of Radio – Echo Sounding Of Ice. *earthzine* Retrieved January 20, 2010, from <http://www.earthzine.org/2008/09/26/a-brief-history-of-radio-echo-sounding-of-ice/>
- . At IU, what is the Data Capacitor? *Knowledge Base* Retrieved March 27, 2010, 2010, from <http://kb.iu.edu/data/avvh.html>
- Blake, W. (2009, December 17, 2009). [Fwd: latest version of matlab code].
- Burney, J., & Evans, R. (2009). A Comparative Analysis of Localized Command Line Execution, Remote Execution through Command Line, and Torque Submissions of Matlab(R) Scripts for the Charting of CReSIS Flight Path Data (pp. 4). Elizabeth City, NC: Elizabeth City State University.
- . Central Air Conditioning Unit Size (AC). (2010) Retrieved March 11, 2010, 2010, from <http://www.homeimprovementhelper.com/airconditioner/size.htm>
- Chen, Y., & Tan, S. F. (2002). MATLAB\*G: A Grid-Based Parallel MATLAB. Cambridge, MA: Massachusetts Institute of Technology.
- Christian Hoge, Dan Keith, & Allen D. Malony. (2007). Client-Side Task Support in Matlab for Concurrent Distributed Execution. In Péter Kacsuk, Thomas Fahringer & Zsolt Németh (Eds.), *Distributed and Parallel Systems* (pp. 133-122): Springer.
- Davidson, R., & MacKinno, J. G. (1993). *Estimation and Inference in Econometrics*: Oxford University Press.
- Edelhofer, T. (2010, March 4th). Number of Tasks in Distributed Computing. Forum Retrieved from [http://www.mathworks.com/matlabcentral/newsreader/view\\_thread/269955](http://www.mathworks.com/matlabcentral/newsreader/view_thread/269955)
- Evers, X., de Jongh, J., Boontje, R., Epema, D., & Van Dantzig, R. (1994). Condor flocking: load sharing between pools of workstations. *NLUUG Voorjaarsconferentie*, 111ñ126.
- Fisher, R. A. (1925). *Statistical Methods for Research Workers* (14 ed.): Oliver & Boyd.
- Geoffrey Fox, C. S., Marlon Pierce, Linda Hayden, Malcolm LeCompte. (2007). PolarGrid: Cyberinfrastructure for Polar Science. In C. S. Department (Ed.). Bloomington, IN: Indiana University.

- Hayden, L., Powell, J., & Akers, E. (2009). *Establishing Field and Base Camp Servers for Remote Sensing of Ice Sheets in Ilulissat, Greenland*. Paper presented at the IGARSS 09, Cape Town, South Africa.  
<http://www.igarss09.org/Papers/AbstractSearch.asp?show=search>
- Husbands, P., & C. Isbell. (1999). *MATLAB\*P: A Tool for Interactive Supercomputing*. Paper presented at the Ninth SIAM Conference on Parallel Processing for Scientific Computing. [http://crd.lbl.gov/~parry/text/ppserver/ppserver-papers/SIAM/siamt\\_0.htm](http://crd.lbl.gov/~parry/text/ppserver/ppserver-papers/SIAM/siamt_0.htm)
- Husbands, P., Jr., C. L. I., & Edelman, A. (1998). *Interactive Supercomputing with MITMatlab*. Cambridge, MA Massachusetts Institute of Technology.
- IBM. IBM System x3550 M2 Retrieved March 11, 2010, 2010, from <http://www-03.ibm.com/systems/x/hardware/rack/x3550m2/specs.html>
- IBM. IBM System x3650 M2 Retrieved March 11, 2010, 2010, from <http://www-03.ibm.com/systems/x/hardware/rack/x3650m2/specs.html>
- . IBM System Storage DS4000 EXP420 Storage Expansion Enclosure: Installation, User's and Maintenance Guide. (2006, 2007). In I. B. M. Corporation (Ed.), (3 ed., Vol. GC27-2050-02): International Business Machines Corporation.
- . IBM System Storage DS4200 Express Storage Subsystem: Installation, User's and Maintenance Guide. (2007). In I. B. M. Corporation (Ed.), (3 ed., Vol. GC27-2048-02): IBM.
- Lite, T. (2009). Model #: B020-016-17
- Console KVM Switch - 16-Port NetDirector 1U Rackmount Console KVM Switch w/17" LCD Retrieved March 11, 2010, 2010, from <http://www.tripplite.com/en/products/model.cfm?txtSeriesID=675&EID=15477&txtModelID=3132>
- Litzkow, M. (1987). *Remote UNIX: turning idle workstations into cycle servers*.
- Lohofener, A. (2006). *Design and Development of a Multi-Channel Radar Depth Sounder*. Lawrence, KS: University of Kansas.
- Mank, B. (2005). Standing and Global Warming: Is Injury to All Injury to None? *Environmental Law*, 35(1), 1-85.
- MathWorks. Parallel Computing Toolbox 4.3 Retrieved March 10, 2010, 2010, from <http://www.mathworks.com/products/parallel-computing/requirements.html>
- MathWorks. Signal Processing Toolbox 6.13 Retrieved March 10, 2010, 2010, from <http://www.mathworks.com/products/signal/requirements.html>

- MathWorks. System Requirements - Release 2009b Retrieved March 10, 2010, 2010, from <http://www.mathworks.com/support/sysreq/release2009b/linux.html>
- . MATLAB - The Language Of Technical Computing. Retrieved 1-28-10, 2010, from <http://www.mathworks.com/products/matlab/>
- Networking, P. (2006). Installation and Getting Started Guide (Vol. 5991-4737). Roseville, California: Hewlett-Packard Development Company.
- Oberthaler, K. (2009). Scientists Contend with Elements During Spring Survey of Greenland Glaciers. *ICEBREAKER*.
- Oliver, C., & Quegan, S. (2004). *Understanding synthetic aperture radar images*: SciTech Publishing.
- . Parallel Computing Toolbox 4.2: Perform parallel computations on multicore computers and computer clusters. (2010) Retrieved 1-29-2010, 2010, from <http://www.mathworks.com/products/parallel-computing/>
- Powell, J. a. (2008). Greenland 2008 Field Issues and Solutions (pp. 3). Illulissat, Greenland: Elizabeth City State University.
- Project, T. C. What is Condor? Retrieved March 7, 2010, 2010, from <http://www.cs.wisc.edu/condor/description.html>
- Rasmussen, N. (2003). Calculating total cooling requirements for data centers. *American Power Conversion*.
- Sheridan, P. (2009). MathWorks Quote #2496043. In MathWorks (Ed.), (Vol. 2496043, pp. 2). Natick: MathWorks.
- Shields, C. (2010). [PGU/BCS specs question].
- Suda, B. (2003). *SOAP Web Services*. Master of Science, The University of Edinburgh, South Bridge Edinburgh.
- The MathWorks, I. Using a Fully Supported Third-Party Scheduler. *Parallel Computing Toolbox* Retrieved March 24, 2010, 2010, from <http://www.mathworks.com/access/helpdesk/help/toolbox/distcomp/bqur7ev-20.html>
- The MathWorks, I. (2005 - 2010). MATLAB® Distributed Computing Server™ 4: System Administrator's Guide (Vol. 4): The MathWorks, Inc.
- Tittel, E. (2010). *Clusters for Dummies*. Mississauga, ON: John Wiley and Sons Canada, Ltd.



- Trefethen, A. E., Menon, V. S., Chang, C.-C., Czajkowski, G. J., Myers, C., & Trefethen, L. N. (1996). MultiMATLAB: MATLAB on Multiple Processors.
- V. Ramasami, S. G., B. Holt, P. Kanagaratnam, K. Gurumoorthy, & S. K. Namburi, J. H., D. Braaten, A. Mahoney, V. Lytle. (2003). *A low frequency wideband depth sounder for sea ice*. Paper presented at the IEEE International Geoscience and Remote Sensing Symposium 2003, Toulouse, France.
- Wilkins-Diehr, N., Gannon, D., Klimeck, G., Oster, S., & Pamidighantam, S. (2008). TeraGrid Science Gateways and Their Impact on Science. *Computer*, 41(11), 32-41.
- Wolff, C. (2009). Synthetic Aperture Radar. *Radar Basics* Retrieved 1-23-2010, 2010, from <http://www.radartutorial.eu/20.airborne/ab07.en.html>
- Xue, G., Fairman, M. J., Pound, G. E., & Cox, S. J. (2003). *Implementation of a Grid Computation Toolkit for Design Optimisation with Matlab and Condor*. Paper presented at the Parallel Processing: 9th International Euro-Par Conference, Klagenfurt, Austria.  
[http://www.geodise.org/files/Papers/Europar03\\_%20G%20XUE.pdf](http://www.geodise.org/files/Papers/Europar03_%20G%20XUE.pdf)

## APPENDIX

### ECSU Param file

```
%-----%
% Example Param For MCRDS data                                     %
%-----%
% This script sets up a param struct with appropriate
values
% This should be used as an example to process the given
type of data

clear all; close all; clc;
format short; format compact;
start_file = [100];
stop_file  = [115]; %279
raw_dir    = {'20080801Calibration'};
cff_dir    = {'20080801Calibration/CFF/'};
proc_dir   = {'20080801Calibration/Proc/'};
gps_name   = {'20080801'};
height     = [500];

cur_dir = cd;
for i=1:length(start_file)
    param.raw_dir      = ['/PGData/',raw_dir{i},'/'];
    param.cff_dir      =
['/home/jeaimehp/scratch/',cff_dir{i}];
    if ~exist(param.cff_dir,'dir')
        mkdir(param.cff_dir);
    end
    param.proc_dir     =
['/home/jeaimehp/scratch/',proc_dir{i}];
    if ~exist(param.proc_dir,'dir')
        mkdir(param.proc_dir);
    end
    param.pos_dir      = '/home/jeaimehp/pos_data/';
    param.ref_dir      = '';
    param.pos_name     =
['MCRDS_',gps_name{i},'_ALL_pos.mat'];
    param.prefix       = 'data.'; %
Prefix for the day's data
    param.type         = 'MCRDS';

    %% Files to process
```

```

    param.file_idx_start    = start_file(i);    % Start file
idx
    param.file_idx_stop     = stop_file(i);     % End file
idx

    %% Waveform/Tx/Rx Triplets to setup processing
    param.wf_tx_rx{1}      = ([1 1 1 1 1 1]);
    param.wf_sched          = ([1 1 1 1 1 1]);
    param.process{1}       = ([1 1 1 1 1 1]);

    %% Processing steps
    param.convert2CFF       = true;
    param.get_height        = true;
    param.select_path       = false;
    param.use_index_file    = false;
    param.pulse_comp        = true;
    param.use_ref           = false;
    param.ft_dec            = true;
    param.st_dec            = true;
    param.mo_comp           = false;
    param.azimuth_comp      = true;

    %% System Parameters
    param.proc_type         = 'fk';
    param.system_delay      = 1.785e-6;        % s - System
Delay
    param.chunk_len         = 4000;            % m - F-k migration
length
    param.fk_bw             = 5;                % degrees - F-k
migration beamwidth
    param.approx_height     = height(i); % m - Approximate
height used in F-k
    param.surf_wf           = 1;

    %% Cluster setup
    param.sched_type        = 'local';
    param.sched_url         = 'localhost';
    param.sched_name        = 'default_jobmanager';

    %% Execute StartScript
    cd('../..../');
    job_sla = Stagela(param); tic;
    waitForState(job_sla, 'finished'); fprintf('Stage 1a
Complete\n\n'); toc;
    job_slb = Stagelb(param); tic;
    waitForState(job_slb, 'finished'); fprintf('Stage 1b
Complete\n\n'); toc;

```

```
        job_s2 = Stage2(param);  
%        StartScriptCluster(param);  
        cd(cur_dir);  
  
end
```

## Startup.m

```
%-----%
% Startup.m File      %
%-----%
% By: William blake

if isunix
    ct_path = '/home/jeaimehp/CReSIS-Toolbox.r16';
elseif ispc
    ct_path = '\\emperor\wblake\William Documents\cresis-
toolbox\';
end

fprintf('Adding cresis toolbox to path at: %s\n',ct_path);
addpath(genpath(ct_path));
fprintf('Setting cresis-toolbox environment variable:
%s\n',ct_path);
setenv('cresis-toolbox',ct_path)
clear ct_path;
format short; format compact;
```

## Stage1a.m (1 Worker Example)

```
function [job] = Stagela(param)
%-----%
% Stage 1a (Create CFF) %
%-----%
% Written by: William Blake
% This Script should provide the inputs to the functions
that follow

addpath(param.type)
%% Check paths and files for existences
if ~exist(param.raw_dir,'dir');
error('Path:NotFound','Input path not found'); end;
if ~exist(param.pos_dir,'dir');
error('Path:NotFound','Position path not found'); end;
if ~exist([param.pos_dir param.pos_name],'file');
error('File:NotFound','Position data not found'); end;
if ~exist(param.cff_dir,'dir');
error('Path:NotFound','Output path not found'); end;
ct_path = getenv('cresis-toolbox');
if isempty(ct_path);
warning('CReSIS_Toolbox:NotFound','cresis-toolbox path not
set'); end

%% Create CFF Files
if param.convert2CFF
    fprintf('Scheduling Creating CFFs\n');
    tic
    if strcmp(param.sched_type,'jobmanager')
        jm =
findResource('scheduler','type',param.sched_type,'LookupUrl
',param.sched_url,'name',param.sched_name);
        % job =
createJob(jm,'RunningFcn',@clientJobRunning); % Create job
on scheduler;
        job = createJob(jm); % Create job on scheduler;
        set(job,'RestartWorker',true);
    else
        jm =
findResource('scheduler','type',param.sched_type,'LookupUrl
',param.sched_url);
        job = createJob(jm); % Create job on scheduler;
    end
end
```

```

        set(job,'MaximumNumberOfWorkers',1);
        type_files = GetFilenames(param.type,'','','.m');
        dir_files = GetFilenames(cd,'','','.m');
        ct_files =
GetFilenames(ct_path,'','','','recursive');
        fd = [type_files.' dir_files.' ct_files.'];
        set(job,'FileDependencies',fd)

        fh = eval(sprintf('@%s2CFF',param.type));
        start_idx = param.file_idx_start;
        stop_idx = param.file_idx_stop;
        arg = cell(1,1);
        for a = start_idx:stop_idx
            param.file_idx_start = a;
            param.file_idx_stop = a;
            arg{1} = param;
            % [success] = fh(arg{1});
            if strcmp(param.sched_type,'jobmanager')
%
createTask(job,fh,1,arg,'MaximumNumberOfRetries',2,'Finishe
dFcn',@clientTaskCompleted);

createTask(job,fh,1,arg,'MaximumNumberOfRetries',2);
            else
                createTask(job,fh,1,arg);
            end
        end
        submit(job);
%        waitForState(job,'finished');
        cur_dir = cd;
        cd(param.cff_dir)
        sysCmd = 'rm -f *NaN*';
        system(sysCmd);
        cd(cur_dir)
    end

    fprintf('Queued Stage 1a Processing\n');
    return;

```

## Stage1b.m (1 Worker Example)

```
function [job] = Stage1b(param)
%-----%
%  Stage1b (GetHeights)          %
%-----%
% Written by: William Blake
% This Script should provide the inputs to the functions
% that follow

addpath(param.type)
%% Check paths and files for existences
if ~exist(param.cff_dir,'dir');
error('Path:NotFound','Output path not found'); end;
ct_path = getenv('cresis-toolbox');
if isempty(ct_path);
warning('CReSIS_Toolbox:NotFound','cresis-toolbox path not
set'); end

%% Get header and position filenames
disp('Getting Header and Position Filenames'); tic;
hdr_fns =
GetFilenames(param.cff_dir,param.type,'','header.txt');
pos_fns =
GetFilenames(param.cff_dir,param.type,'','pos.mat');
toc
num_files = length(hdr_fns);
if num_files == 0; error('Output:Error','No Files in output
directory'); end;

%% Add Heights to CFF
if param.get_height
    disp('Creating GetHeights Jobs');
    tic
    % Setup job manager etc
    if strcmp(param.sched_type,'jobmanager')
        jm =
findResource('scheduler','type',param.sched_type,'LookupUrl
',param.sched_url,'name',param.sched_name);
    %
        job =
createJob(jm,'RunningFcn',@clientJobRunning); % Create job
on scheduler;
        job = createJob(jm); % Create job on scheduler;
```



```

        set(job, 'RestartWorker', true);
    else
        jm =
findResource('scheduler', 'type', param.sched_type, 'LookupUrl
', param.sched_url);
        job = createJob(jm); % Create job on scheduler;
    end
    set(job, 'MaximumNumberOfWorkers', 1);
    dir_files = GetFileNames(cd, '', '', '.m');
    ct_files =
GetFileNames(ct_path, '', '', '', 'recursive');
    fd = [dir_files.' ct_files.'];
    set(job, 'FileDependencies', fd)
    fh = @GetHeight;
    wf = max(1, param.surf_wf);
    [tx_all, rx_all] = find(param.wf_tx_rx{wf});
    tx = tx_all(1); % Grab first valid tx
    rx = rx_all(1); % Grab first valid rx
    wf_tx_rx_name =
sprintf('wf_%02d_tx_%02d_rx_%02d.mat', wf, tx, rx);
    radar_fns = GetFileNames(param.cff_dir, param.type,
'radar', wf_tx_rx_name);
    arg = cell(1, 4);
    for a = 1:length(radar_fns)
        arg{1} = param;
        arg{2} = radar_fns{a};
        arg{3} = pos_fns{a};
        arg{4} = hdr_fns{a};
        % fh(arg{1}, arg{2}, arg{3}, arg{4});
        if strcmp(param.sched_type, 'jobmanager')
%
createTask(job, fh, 1, arg, 'MaximumNumberOfRetries', 2, 'Finishe
dFcn', @clientTaskCompleted);

createTask(job, fh, 1, arg, 'MaximumNumberOfRetries', 2);
        else
            createTask(job, fh, 1, arg);
        end
    end
    submit(job);
%    waitForState(job, 'finished');
end

fprintf('Queued Stage 1b Processing\n');
return;

```

## Stage2.m (1 Worker Example)

```
function [job] = Stage2(param)
%-----%
%   Stage 2 Process Data           %
%-----%
% Written by: William Blake
% This Script should provide the inputs to the functions
% that follow

global num_tasks comp_tasks
addpath(param.type)
%% Check paths and files for existences
if ~exist(param.cff_dir,'dir'); error('Path:NotFound','CFF
path not found'); end;
if ~exist(param.proc_dir,'dir');
error('Path:NotFound','Proc path not found'); end;
if param.use_ref && ~exist(param.ref_dir,'dir');
error('Path:NotFound','Reference data path not found');
end;
ct_path = getenv('cresis-toolbox');
if isempty(ct_path);
warning('CReSIS_Toolbox:NotFound','cresis-toolbox path not
set'); end

% Check if there are already files in the output folder
[s r] = system(sprintf('[ "$(ls -A %s)" ] && echo "Not
Empty" || echo "Empty"',param.proc_dir));
[s r2] = system(sprintf('du %s', param.proc_dir));
if strcmp(r(1:end-1),'Not Empty') && str2num(strtok(r2)) >
1e3
    %%ch_del = input('Files already exist in this folder.
Would you like to delete them?: ','s');
    %%if ch_del == 'y'
        cur_dir = cd;
        cd(param.proc_dir)
        system('find . -name '*fk*' -exec rm -r {} \;');
        system('find . -name '*pc*' -exec rm -r {} \;');
        cd(cur_dir)
    %%end
end
```

```

%% Get header and position filenames
disp('Getting Header and Position Filenames'); tic;
hdr_fns =
GetFilenames(param.cff_dir,param.type,'','header.txt');
pos_fns =
GetFilenames(param.cff_dir,param.type,'','pos.mat');
toc
num_files = length(hdr_fns);
if num_files == 0; error('Output:Error','No Files in output
directory'); end;

%% Get data filenames
fprintf('Getting Data Filenames\n'); tic
idx = 0;
for wf = unique(param.wf_sched)
    [tx_all,rx_all] = find(param.process{wf});
    for tx_rx_idx = 1:length(tx_all)
        tx = tx_all(tx_rx_idx);
        rx = rx_all(tx_rx_idx);
        idx = idx+1;
        wf_tx_rx_name =
sprintf('wf_%02d_tx_%02d_rx_%02d.mat',wf, tx, rx);
        radar_fns{idx} = GetFilenames(param.cff_dir,
param.type, 'radar', wf_tx_rx_name);
    end
end
toc

if param.azimuth_comp || param.pulse_comp ||
param.select_path || param.use_index_file
    %% Concatonate position files
    % [lat_all, lon_all, breaks] =
ExtractVariable(pos_fns,'lat','lon');
    lat_all = cell(1,length(pos_fns));
    lon_all = cell(1,length(pos_fns));
    breaks = zeros(1,length(pos_fns)+1);
    breaks(1) = 0;
    for a = 1:length(pos_fns)
        load(pos_fns{a},'lat','lon')
        lat_all{a} = lat;
        lon_all{a} = lon;
        breaks(a+1) = length(lat)+breaks(a);
    end
    breaks = breaks(2:end);

    %% Select Path if choosen
    if param.select_path

```

```

        [start_idx, stop_idx, fk_seg, pc_seg] =
SelectPath(param, lon_all, lat_all);
        save([param.proc_dir
'indexes.mat'], 'start_idx', 'stop_idx', 'fk_seg', 'pc_seg', '-
v7.3');
        elseif isfield(param, 'breaks')
            if ~isempty(param.breaks)
                fk_seg = [];
                pc_seg = [];
                start_idx(1) = 1;
                for aa=1:length(param.breaks)
                    stop_idx(aa) = breaks(param.breaks(aa));
                    start_idx(aa+1) = stop_idx(aa)+1;
                    if strcmp(param.seg_id(V. Ramasami & S. K.
Namburi), 'fk')
                        fk_seg(end+1) = aa;
                    elseif strcmp(param.seg_id(V. Ramasami & S. K.
Namburi), 'pc')
                        pc_seg(end+1) = aa;
                    end
                end
                start_idx = start_idx(1:end-1);
                save([param.proc_dir
'indexes.mat'], 'start_idx', 'stop_idx', 'fk_seg', 'pc_seg', '-
v7.3');
            end
            elseif param.use_index_file
                if ~isfield(param, 'indexes_name') ||
~strcmp(param.indexes_name, '')
                    load([param.proc_dir 'indexes.mat']);
                else
                    load([param.indexes_dir param.indexes_name]);
                end
            else
                fk_seg = [];
                pc_seg = [];
                if param.azimuth_comp
                    fk_seg = 1;
                elseif param.pulse_comp
                    pc_seg = 1;
                end
                start_idx(1) = 1;
                stop_idx(1) = length([lat_all{:}]);
                save([param.proc_dir
'indexes.mat'], 'start_idx', 'stop_idx', 'fk_seg', 'pc_seg', '-
v7.3');
            end
end

```

```

        if param.azimuth_comp || param.pulse_comp
%% Create chunks of fk data from selected segments
            hdr      = ReadHeader(hdr_fns{1});
            % Calculate overlap length for fk
            sample_stop_time = 0;
            for wf = unique(param.wf_sched)
                sample_start_time = hdr.wf(wf).sample_delay(1) -
hdr.system_delay-hdr.wf(wf).pulse_width;
                sample_stop_time =
max(sample_stop_time, (sample_start_time +
hdr.wf(wf).num_sam(1)/hdr.samp_freq+hdr.wf(wf).pulse_width)
);
            end
            param.overlap =
(sample_stop_time*3e8/2)*tand(param.fk_bw); % m - Ammount
overlap for azimuth compression
            disp('Creating Chunks of data to process');
            tic
            [chunk_file_start, chunk_file_stop,
chunk_offset_start, chunk_offset_stop] = ...
                GetChunkIdxs(param, lat_all, lon_all,
start_idx, stop_idx, fk_seg, pc_seg);
            toc

%% Process Chunks of data
            fprintf('Creating Processing Jobs\n');
            tic
            if strcmp(param.sched_type, 'jobmanager')
                jm =
findResource('scheduler', 'type', param.sched_type, 'LookupUrl
', param.sched_url, 'name', param.sched_name);
                % job =
createJob(jm, 'RunningFcn', @clientJobRunning); % Create job
on scheduler;
                job = createJob(jm); % Create job on
scheduler;
                set(job, 'RestartWorker', true);
            else
                jm =
findResource('scheduler', 'type', param.sched_type, 'LookupUrl
', param.sched_url);
                job = createJob(jm); % Create job on
scheduler;
            end
            set(job, 'MaximumNumberOfWorkers', 1);
            dir_files = GetFileNames(cd, '', '', '.m');

```

```

        ct_files      =
GetFileNames(ct_path, '', '', '', 'recursive');
        fd = [dir_files.' ct_files.'];
        set(job, 'FileDependencies', fd)
        fh = @ProcessChunk;
        arg = cell(1,12);
        tasks = 0;
        for seg = 1:length(chunk_file_stop)
            if find(seg == fk_seg) % If it's an fk segment
                param.azimuth_comp = true;
            else % Otherwise it's a pc segment
                param.azimuth_comp = false;
            end
            rf_idx = 0;
            for wf = unique(param.wf_sched) % Iterate
through the wfs
                [tx_all, rx_all] = find(param.process{wf});
                for tx_rx_idx = 1:length(tx_all) % Iterate
through the tx/rx pairs
                    tx = tx_all(tx_rx_idx);
                    rx = rx_all(tx_rx_idx);
                    rf_idx = rf_idx+1;
                    for idx =
1:length(chunk_file_stop{seg})
                        tasks = tasks + 1;
                        rf_idx = 7; seg = 1; idx = 58;
% wf = 2; tx = 1; rx = 2;
%
                        warning('TESTING!!!')
                        arg{1} = param;
                        arg{2} = wf;
                        arg{3} = tx;
                        arg{4} = rx;
                        arg{5} =
chunk_offset_start{seg}(idx);
                        arg{6} =
chunk_offset_stop{seg}(idx);
                        arg{7} = seg;
                        if idx == 1; arg{8} = true;
                        else arg{8} = false; end
                        if idx ==
length(chunk_file_stop{seg}); arg{9} = true;
                        else
arg{9} = false; end
                        arg{10} =
{radar_fns{rf_idx}{chunk_file_start{seg}(idx):chunk_file_st
op{seg}(idx)}};

```

```

                                arg{11} =
{hdr_fns{chunk_file_start{seg}(idx):chunk_file_stop{seg}(id
x)}};
                                arg{12} =
{pos_fns{chunk_file_start{seg}(idx):chunk_file_stop{seg}(id
x)}};
                                %
                                [success] = fh(arg{1}, arg{2},
arg{3}, arg{4}, arg{5}, arg{6}, arg{7}, arg{8}, arg{9},
arg{10}, arg{11}, arg{12});
                                if
strcmp(param.sched_type,'jobmanager')

createTask(job,fh,1,arg,'MaximumNumberOfRetries',2,'Finishe
dFcn',@clientTaskCompleted);
                                %
createTask(job,fh,1,arg,'MaximumNumberOfRetries',2);
                                else
                                createTask(job,fh,1,arg);
                                end
                                end
                                end
                                end
                                submit(job);
                                toc
                                % results = getAllOutputArguments(job);
                                end
                                end

                                fprintf('Queued Stage 2 Processing\n');
                                num_tasks = tasks;
                                comp_tasks = 0;
                                tic
                                waitForState(job,'finished');   fprintf('Stage 2
Complete\n\n');
                                toc
                                return;

```

```

function clientJobRunning(job,eventData)
    global num_tasks;
    global comp_tasks;
    comp_tasks = 0;
    num_tasks = length(job.Tasks);
    disp(['Job started with ' num2str(length(job.Tasks)) '
tasks'])
    disp('% Done | Time left (s)');

```

```

return;

function clientTaskCompleted(task,eventData)
    global num_tasks;
    global comp_tasks;
    every_10 = round(num_tasks*(0:0.1:1));
    if task.AttemptedNumberOfRetries == 0
        comp_tasks = comp_tasks+1;
    end
    if any(comp_tasks == every_10)
        elapsed_time = toc;
        frac_comp = comp_tasks/num_tasks;
        disp(sprintf('%3.0f      | %6.0f',frac_comp*100, ...
            (elapsed_time/frac_comp)-elapsed_time));
    end
return;

```



## IU Worker Collected Times

1 worker

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 263.169995 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.032639 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 55.565447 seconds.  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023440 seconds.  
Getting Data Filenames  
Elapsed time is 0.066996 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.053062 seconds.  
Creating Processing Jobs  
Elapsed time is 1.306517 seconds.  
Queued Stage 2 Processing

11		1258
22		1078
28		996
39		833
50		677
61		528
72		375
78		301
89		150
100		0

Stage 2 Complete

Elapsed time is 1350.653484 seconds.

-----

Scheduling Creating CFFs  
Queued Stage 1a Processing

Stage 1a Complete

Elapsed time is 256.934506 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.032633 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 55.481458 seconds.  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023458 seconds.  
Getting Data Filenames  
Elapsed time is 0.067101 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.037204 seconds.  
Creating Processing Jobs  
Elapsed time is 1.321434 seconds.  
Queued Stage 2 Processing

11		1259
22		1075
28		993
39		835
50		678
61		528
72		376
78		301
89		150
100		0

Stage 2 Complete

Elapsed time is 1349.069837 seconds.

-----

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 258.996669 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.033282 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

```
Elapsed time is 55.150270 seconds.  
find: ./fk_data_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023884 seconds.  
Getting Data Filenames  
Elapsed time is 0.067772 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.050691 seconds.  
Creating Processing Jobs  
Elapsed time is 1.307984 seconds.  
Queued Stage 2 Processing  
  11      |      1251  
  22      |      1069  
  28      |       992  
  39      |       832  
  50      |       677  
  61      |       526  
  72      |       375  
  78      |       300  
  89      |       150  
 100      |         0  
Stage 2 Complete  
  
Elapsed time is 1346.057017 seconds.
```

2 workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 136.033402 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.033096 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 31.115552 seconds.  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023517 seconds.  
Getting Data Filenames  
Elapsed time is 0.067407 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.028050 seconds.  
Creating Processing Jobs  
Elapsed time is 1.310219 seconds.  
Queued Stage 2 Processing

11		658
22		554
28		613
39		489
50		384
61		292
72		206
78		154
89		77
100		0

Stage 2 Complete

Elapsed time is 686.575861 seconds.

-----

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 136.078328 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.033154 seconds.

Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 31.963880 seconds.  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023423 seconds.  
Getting Data Filenames  
Elapsed time is 0.067584 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.045468 seconds.  
Creating Processing Jobs  
Elapsed time is 1.307736 seconds.  
Queued Stage 2 Processing

11		659
22		556
28		600
39		483
50		381
61		291
72		205
78		153
89		76
100		0

Stage 2 Complete

Elapsed time is 684.695795 seconds.

-----

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 131.888079 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.033244 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 30.932078 seconds.  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023407 seconds.  
Getting Data Filenames

Elapsed time is 0.067614 seconds.

Creating Chunks of data to process

Elapsed time is 1.040379 seconds.

Creating Processing Jobs

Elapsed time is 1.322447 seconds.

Queued Stage 2 Processing

11		663
----	--	-----

22		558
----	--	-----

28		597
----	--	-----

39		484
----	--	-----

50		381
----	--	-----

61		291
----	--	-----

72		205
----	--	-----

78		153
----	--	-----

89		76
----	--	----

100		0
-----	--	---

Stage 2 Complete

Elapsed time is 683.494320 seconds.

4 Workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 75.486038 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.032863 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 18.973784 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023759 seconds.  
Getting Data Filenames  
Elapsed time is 0.067778 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.039744 seconds.  
Creating Processing Jobs  
Elapsed time is 1.323601 seconds.  
Queued Stage 2 Processing

11		663
22		294
28		409
39		251
50		231
61		150
72		118
78		88
89		39
100		0

Stage 2 Complete

Elapsed time is 383.474079 seconds.

-----

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

```

Elapsed time is 74.102562 seconds.
Getting Header and Position Filenames
Elapsed time is 0.033288 seconds.
Creating GetHeights Jobs
Queued Stage 1b Processing
Stage 1b Complete

Elapsed time is 19.042741 seconds.
Files already exist in this folder.  Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.023553 seconds.
Getting Data Filenames
Elapsed time is 0.067753 seconds.
Creating Chunks of data to process
Elapsed time is 1.018677 seconds.
Creating Processing Jobs
Elapsed time is 1.308272 seconds.
Queued Stage 2 Processing
 11      |      654
 22      |      289
 28      |      406
 39      |      248
 50      |      230
 61      |      148
 72      |      118
 78      |       88
 89      |       39
100      |       0
Stage 2 Complete

Elapsed time is 382.556132 seconds.

-----

Scheduling Creating CFFs
Queued Stage 1a Processing
Stage 1a Complete

Elapsed time is 70.963758 seconds.
Getting Header and Position Filenames
Elapsed time is 0.033014 seconds.
Creating GetHeights Jobs
Queued Stage 1b Processing
Stage 1b Complete

```



```
Elapsed time is 18.730993 seconds.
Files already exist in this folder.  Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.023752 seconds.
Getting Data Filenames
Elapsed time is 0.067016 seconds.
Creating Chunks of data to process
Elapsed time is 1.051241 seconds.
Creating Processing Jobs
Elapsed time is 1.301494 seconds.
Queued Stage 2 Processing
 11      |      651
 22      |      292
 28      |      405
 39      |      250
 50      |      230
 61      |      149
 72      |      118
 78      |       88
 89      |       39
100      |        0
Stage 2 Complete

Elapsed time is 382.166817 seconds.
```

8 Workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 40.828068 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.024737 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 12.656842 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.024708 seconds.  
Getting Data Filenames  
Elapsed time is 0.067604 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.043085 seconds.  
Creating Processing Jobs  
Elapsed time is 1.389920 seconds.  
Queued Stage 2 Processing

11		663
22		291
28		237
39		149
50		156
61		101
72		68
78		52
89		23
100		0

Stage 2 Complete

Elapsed time is 233.155440 seconds.

-----  
-----

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 39.043730 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.029269 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 12.630410 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023569 seconds.  
Getting Data Filenames  
Elapsed time is 0.067402 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.032675 seconds.  
Creating Processing Jobs  
Elapsed time is 1.319506 seconds.  
Queued Stage 2 Processing

11		659
22		290
28		236
39		146
50		157
61		101
72		69
78		51
89		25
100		0

Stage 2 Complete

Elapsed time is 232.052556 seconds.

-----

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 39.113816 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.024888 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

```
Elapsed time is 12.387591 seconds.
Files already exist in this folder.  Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.023554 seconds.
Getting Data Filenames
Elapsed time is 0.068193 seconds.
Creating Chunks of data to process
Elapsed time is 1.024386 seconds.
Creating Processing Jobs
Elapsed time is 1.387105 seconds.
Queued Stage 2 Processing
 11      |      657
 22      |      288
 28      |      237
 39      |      148
 50      |      156
 61      |      101
 72      |       69
 78      |       52
 89      |       23
100      |        0
Stage 2 Complete

Elapsed time is 230.802283 seconds.
```

12 Workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 40.449667 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.024903 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 12.119444 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.024749 seconds.  
Getting Data Filenames  
Elapsed time is 0.066759 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.042889 seconds.  
Creating Processing Jobs  
Elapsed time is 1.404768 seconds.  
Queued Stage 2 Processing

11		659
22		344
28		270
39		166
50		116
61		76
72		61
78		46
89		23
100		0

Stage 2 Complete

Elapsed time is 188.907026 seconds.

-----

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 37.539038 seconds.

Getting Header and Position Filenames

Elapsed time is 0.029515 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 12.343576 seconds.

Files already exist in this folder. Would you like to delete them?: y

find: ./fk\_data\_01: No such file or directory

Getting Header and Position Filenames

Elapsed time is 0.023497 seconds.

Getting Data Filenames

Elapsed time is 0.067640 seconds.

Creating Chunks of data to process

Elapsed time is 1.028220 seconds.

Creating Processing Jobs

Elapsed time is 1.387598 seconds.

Queued Stage 2 Processing

11		668
----	--	-----

22		342
----	--	-----

28		254
----	--	-----

39		159
----	--	-----

50		112
----	--	-----

61		75
----	--	----

72		61
----	--	----

78		46
----	--	----

89		23
----	--	----

100		0
-----	--	---

Stage 2 Complete

Elapsed time is 186.807848 seconds.

-----

Scheduling Creating CFFs

Queued Stage 1a Processing

Stage 1a Complete

Elapsed time is 39.009672 seconds.

Getting Header and Position Filenames

Elapsed time is 0.024994 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 12.601885 seconds.

```
Files already exist in this folder.  Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.023801 seconds.
Getting Data Filenames
Elapsed time is 0.067476 seconds.
Creating Chunks of data to process
Elapsed time is 1.016598 seconds.
Creating Processing Jobs
Elapsed time is 1.324072 seconds.
Queued Stage 2 Processing
 11      |      669
 22      |      336
 28      |      252
 39      |      158
 50      |      116
 61      |       79
 72      |       61
 78      |       46
 89      |       22
100      |        0
Stage 2 Complete

Elapsed time is 186.851162 seconds.
```

16 Workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 22.375646 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.024960 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 9.070295 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023849 seconds.  
Getting Data Filenames  
Elapsed time is 0.067509 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.032532 seconds.  
Creating Processing Jobs  
Elapsed time is 1.303295 seconds.  
Queued Stage 2 Processing

11		787
22		372
28		276
39		171
50		113
61		76
72		46
78		35
89		16
100		0

Stage 2 Complete

Elapsed time is 180.852529 seconds.

-----

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete



Elapsed time is 27.292390 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.025474 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 9.205106 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.023578 seconds.  
Getting Data Filenames  
Elapsed time is 0.067755 seconds.  
Creating Chunks of data to process  
Elapsed time is 1.045964 seconds.  
Creating Processing Jobs  
Elapsed time is 1.289169 seconds.  
Queued Stage 2 Processing

11		749
22		336
28		258
39		172
50		112
61		75
72		47
78		35
89		16
100		0

Stage 2 Complete

Elapsed time is 180.000216 seconds.

-----

cheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 25.981820 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.024602 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

```
Elapsed time is 10.745828 seconds.
Files already exist in this folder. Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.023584 seconds.
Getting Data Filenames
Elapsed time is 0.067818 seconds.
Creating Chunks of data to process
Elapsed time is 1.054346 seconds.
Creating Processing Jobs
Elapsed time is 1.306283 seconds.
Queued Stage 2 Processing
 11      |      848
 22      |      371
 28      |      283
 39      |      174
 50      |      119
 61      |       78
 72      |       49
 78      |       38
 89      |       17
100      |        0
Stage 2 Complete

Elapsed time is 185.270140 seconds.
```

## Madogo Worker Collected Times

1 worker

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 257.165021 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.015393 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 132.404736 seconds.  
Files already exist in this folder. Would you like to  
delete them?: n  
Getting Header and Position Filenames  
Elapsed time is 0.014707 seconds.  
Getting Data Filenames  
Elapsed time is 0.051228 seconds.  
Creating Chunks of data to process  
Elapsed time is 0.832706 seconds.  
Creating Processing Jobs  
Elapsed time is 1.165571 seconds.  
Queued Stage 2 Processing

11		1243
22		1076
28		997
39		843
50		686
61		534
72		381
78		305
89		152
100		0

Stage 2 Complete

Elapsed time is 1370.436145 seconds.

Scheduling Creating CFFs  
Queued Stage 1a Processing

Stage 1a Complete

Elapsed time is 256.548482 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.307455 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 132.502838 seconds.  
Files already exist in this folder. Would you like to  
delete them?:y  
u find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.013834 seconds.  
Getting Data Filenames  
Elapsed time is 0.051114 seconds.  
Creating Chunks of data to process  
Elapsed time is 0.900855 seconds.  
Creating Processing Jobs  
Elapsed time is 1.153336 seconds.  
Queued Stage 2 Processing

11		1225
22		1066
28		989
39		836
50		684
61		533
72		380
78		304
89		152
100		0

Stage 2 Complete

Elapsed time is 1364.602501 seconds.

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 257.443236 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.013799 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

```
Elapsed time is 132.048523 seconds.
Files already exist in this folder. Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.014322 seconds.
Getting Data Filenames
Elapsed time is 0.052934 seconds.
Creating Chunks of data to process
Elapsed time is 0.871610 seconds.
Creating Processing Jobs
Elapsed time is 1.193518 seconds.
Queued Stage 2 Processing
 11      |      1231
 22      |      1071
 28      |       994
 39      |       838
 50      |       683
 61      |       533
 72      |       380
 78      |       304
 89      |       152
100      |         0
Stage 2 Complete

Elapsed time is 1367.048807 seconds.
```

2 workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 136.656722 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.013813 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 67.574748 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.014240 seconds.  
Getting Data Filenames  
Elapsed time is 0.052997 seconds.  
Creating Chunks of data to process  
Elapsed time is 0.956613 seconds.  
Creating Processing Jobs  
Elapsed time is 1.131056 seconds.  
Queued Stage 2 Processing

11		741
22		644
28		702
39		568
50		449
61		343
72		242
78		181
89		90
100		0

Stage 2 Complete

Elapsed time is 810.120854 seconds.

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 135.355412 seconds.

Getting Header and Position Filenames

Elapsed time is 0.014788 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 67.858275 seconds.

Files already exist in this folder. Would you like to delete them?: y

find: ./fk\_data\_01: No such file or directory

Getting Header and Position Filenames

Elapsed time is 0.013842 seconds.

Getting Data Filenames

Elapsed time is 0.051300 seconds.

Creating Chunks of data to process

Elapsed time is 0.903044 seconds.

Creating Processing Jobs

Elapsed time is 1.143052 seconds.

Queued Stage 2 Processing

11		738
----	--	-----

22		644
----	--	-----

28		699
----	--	-----

39		568
----	--	-----

50		450
----	--	-----

61		343
----	--	-----

72		243
----	--	-----

78		181
----	--	-----

89		90
----	--	----

100		0
-----	--	---

Stage 2 Complete

Elapsed time is 811.526948 seconds.

Scheduling Creating CFFs

Queued Stage 1a Processing

Stage 1a Complete

Elapsed time is 135.554320 seconds.

Getting Header and Position Filenames

Elapsed time is 0.016652 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 67.126349 seconds.

```
Files already exist in this folder.  Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.014308 seconds.
Getting Data Filenames
Elapsed time is 0.050892 seconds.
Creating Chunks of data to process
Elapsed time is 0.829008 seconds.
Creating Processing Jobs
Elapsed time is 1.067212 seconds.
Queued Stage 2 Processing
 11      |      738
 22      |      645
 28      |      708
 39      |      572
 50      |      453
 61      |      345
 72      |      244
 78      |      181
 89      |       91
100      |       0
Stage 2 Complete

Elapsed time is 815.517563 seconds.
```



4 Workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 74.339780 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.014838 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 38.942780 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.013815 seconds.  
Getting Data Filenames  
Elapsed time is 0.051844 seconds.  
Creating Chunks of data to process  
Elapsed time is 0.835245 seconds.  
Creating Processing Jobs  
Elapsed time is 1.087836 seconds.  
Queued Stage 2 Processing

11		1182
22		527
28		750
39		464
50		429
61		280
72		220
78		167
89		73
100		0

Stage 2 Complete

Elapsed time is 670.694962 seconds.

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 74.967272 seconds.  
Getting Header and Position Filenames

Elapsed time is 0.012302 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 39.080960 seconds.

Files already exist in this folder. Would you like to delete them?: y

find: ./fk\_data\_01: No such file or directory

Getting Header and Position Filenames

Elapsed time is 0.013856 seconds.

Getting Data Filenames

Elapsed time is 0.051658 seconds.

Creating Chunks of data to process

Elapsed time is 0.881859 seconds.

Creating Processing Jobs

Elapsed time is 1.095371 seconds.

Queued Stage 2 Processing

11		1184
----	--	------

22		525
----	--	-----

28		753
----	--	-----

39		467
----	--	-----

50		429
----	--	-----

61		280
----	--	-----

72		221
----	--	-----

78		166
----	--	-----

89		73
----	--	----

100		0
-----	--	---

Stage 2 Complete

Elapsed time is 670.359159 seconds.

Scheduling Creating CFFs

Queued Stage 1a Processing

Stage 1a Complete

Elapsed time is 74.465996 seconds.

Getting Header and Position Filenames

Elapsed time is 0.012230 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 39.406272 seconds.

Files already exist in this folder. Would you like to delete them?: y

```

find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.014120 seconds.
Getting Data Filenames
Elapsed time is 0.052919 seconds.
Creating Chunks of data to process
Elapsed time is 0.893295 seconds.
Creating Processing Jobs
Elapsed time is 1.133300 seconds.
Queued Stage 2 Processing
 11      |      1186
 22      |      527
 28      |      754
 39      |      465
 50      |      431
 61      |      280
 72      |      222
 78      |      165
 89      |       73
100      |       0
Stage 2 Complete

```

Elapsed time is 669.609683 seconds.

## 8 Workers

```

Scheduling Creating CFFs
Queued Stage 1a Processing
Stage 1a Complete

```

```

Elapsed time is 52.415833 seconds.
Getting Header and Position Filenames
Elapsed time is 0.018534 seconds.
Creating GetHeights Jobs
Queued Stage 1b Processing
Stage 1b Complete

```

```

Elapsed time is 33.045271 seconds.
Files already exist in this folder. Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.018012 seconds.
Getting Data Filenames
Elapsed time is 0.045211 seconds.

```

Creating Chunks of data to process

Elapsed time is 0.783654 seconds.

Creating Processing Jobs

Elapsed time is 1.131179 seconds.

Queued Stage 2 Processing

11		2040
22		912
28		681
39		417
50		507
61		324
72		201
78		150
89		66
100		0

Stage 2 Complete

Elapsed time is 601.589055 seconds.

Scheduling Creating CFFs

Queued Stage 1a Processing

Stage 1a Complete

Elapsed time is 53.621562 seconds.

Getting Header and Position Filenames

Elapsed time is 0.015535 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 33.570699 seconds.

Files already exist in this folder. Would you like to delete them?: y

find: ./fk\_data\_01: No such file or directory

Getting Header and Position Filenames

Elapsed time is 0.019432 seconds.

Getting Data Filenames

Elapsed time is 0.054506 seconds.

Creating Chunks of data to process

Elapsed time is 0.869925 seconds.

Creating Processing Jobs

Elapsed time is 1.083244 seconds.

Queued Stage 2 Processing

11		2022
22		906
28		693

39		421
50		477
61		321
72		201
78		150
89		66
100		0

Stage 2 Complete

Elapsed time is 595.842584 seconds.

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 54.721518 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.018149 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 33.404025 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.019356 seconds.  
Getting Data Filenames  
Elapsed time is 0.054826 seconds.  
Creating Chunks of data to process  
Elapsed time is 0.865485 seconds.  
Creating Processing Jobs  
Elapsed time is 1.091189 seconds.  
Queued Stage 2 Processing

11		2059
22		923
28		686
39		418
50		500
61		328
72		200
78		150
89		66
100		0

Stage 2 Complete

Elapsed time is 603.318446 seconds.

16 Workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 45.361289 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.015630 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 32.630831 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.019510 seconds.  
Getting Data Filenames  
Elapsed time is 0.054945 seconds.  
Creating Chunks of data to process  
Elapsed time is 0.857649 seconds.  
Creating Processing Jobs  
Elapsed time is 1.182144 seconds.  
Queued Stage 2 Processing

11		3857
22		1722
28		1286
39		816
50		522
61		333
72		202
78		150
89		66
100		0

Stage 2 Complete

Elapsed time is 596.700018 seconds.

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 54.532870 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.021627 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 32.385430 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.019551 seconds.  
Getting Data Filenames  
Elapsed time is 0.055368 seconds.  
Creating Chunks of data to process  
Elapsed time is 0.883319 seconds.  
Creating Processing Jobs  
Elapsed time is 1.151601 seconds.  
Queued Stage 2 Processing

11		3711
22		1751
28		1302
39		802
50		513
61		330
72		201
78		151
89		66
100		0

Stage 2 Complete

Elapsed time is 591.649293 seconds.

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 58.334409 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.017616 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 32.448797 seconds.

```
Files already exist in this folder.  Would you like to
delete them?: y
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.019123 seconds.
Getting Data Filenames
Elapsed time is 0.054986 seconds.
Creating Chunks of data to process
Elapsed time is 0.851165 seconds.
Creating Processing Jobs
Elapsed time is 1.044356 seconds.
Queued Stage 2 Processing
 11      |      3889
 22      |      1749
 28      |      1313
 39      |       808
 50      |       517
 61      |       330
 72      |       201
 78      |       151
 89      |        66
100      |         0
Stage 2 Complete

Elapsed time is 597.846686 seconds.
```



32 Workers

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 52.546661 seconds.  
Getting Header and Position Filenames  
Elapsed time is 0.018926 seconds.  
Creating GetHeights Jobs  
Queued Stage 1b Processing  
Stage 1b Complete

Elapsed time is 34.736004 seconds.  
Files already exist in this folder. Would you like to  
delete them?: y  
find: ./fk\_data\_01: No such file or directory  
Getting Header and Position Filenames  
Elapsed time is 0.019047 seconds.  
Getting Data Filenames  
Elapsed time is 0.054954 seconds.  
Creating Chunks of data to process  
Elapsed time is 0.877092 seconds.  
Creating Processing Jobs  
Elapsed time is 1.151921 seconds.  
Queued Stage 2 Processing

11		4268
22		1896
28		1410
39		857
50		550
61		367
72		224
78		166
89		73
100		0

Stage 2 Complete

Elapsed time is 588.064684 seconds.

Scheduling Creating CFFs  
Queued Stage 1a Processing  
Stage 1a Complete

Elapsed time is 56.618169 seconds.  
Getting Header and Position Filenames

Elapsed time is 0.017847 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 34.191422 seconds.

Files already exist in this folder. Would you like to delete them?: y

find: ./fk\_data\_01: No such file or directory

Getting Header and Position Filenames

Elapsed time is 0.019046 seconds.

Getting Data Filenames

Elapsed time is 0.055747 seconds.

Creating Chunks of data to process

Elapsed time is 0.930826 seconds.

Creating Processing Jobs

Elapsed time is 1.149208 seconds.

Queued Stage 2 Processing

11		4225
----	--	------

22		1907
----	--	------

28		1442
----	--	------

39		892
----	--	-----

50		576
----	--	-----

61		369
----	--	-----

72		224
----	--	-----

78		166
----	--	-----

89		73
----	--	----

100		0
-----	--	---

Stage 2 Complete

Elapsed time is 590.101044 seconds.

Scheduling Creating CFFs

Queued Stage 1a Processing

Stage 1a Complete

Elapsed time is 54.701255 seconds.

Getting Header and Position Filenames

Elapsed time is 0.016395 seconds.

Creating GetHeights Jobs

Queued Stage 1b Processing

Stage 1b Complete

Elapsed time is 33.558612 seconds.

Files already exist in this folder. Would you like to delete them?: y

```
find: ./fk_data_01: No such file or directory
Getting Header and Position Filenames
Elapsed time is 0.019830 seconds.
Getting Data Filenames
Elapsed time is 0.055455 seconds.
Creating Chunks of data to process
Elapsed time is 0.865359 seconds.
Creating Processing Jobs
Elapsed time is 1.079317 seconds.
Queued Stage 2 Processing
 11      |      4102
 22      |      1908
 28      |      1435
 39      |       890
 50      |       569
 61      |       367
 72      |       222
 78      |       166
 89      |        73
100      |         0
Stage 2 Complete

Elapsed time is 590.408716 seconds.
```

## Cluster Power and Cooling Table

Environmental Needs for a 32-Core CSARP Cluster (Requirements as defined from Madogo Components)				
Power Requirements				
Item	# of Item	Power Supply Multiplier	Power Draw (Watts)	Power (Watts)
IBM x3650	2	2	675	2700
IBM x3550	4	2	675	5400
IBM DS4200	1	1	443	443
IBM DS420EXP	2	1	443	886
Tripplite KVM	1	1	22.8	22.8
HP ProCurve	1	1	254	254
		Total Power in Watts		9705.80
Cooling Requirements				
BTU / hr = Power X 3.41				
BTU / hr =		33096.78		
Tons = Power X .000283				
Tons =		2.746741		