January 2018

Implementation Of Computational Resources Focusing On Forecast Model Access For Universities

Timothy Wayne See Jr

IMPLEMENTATION OF COMPUTATIONAL RESOURCES FOCUSING ON FORECAST MODEL ACCESS FOR UNIVERSITIES

by

Timothy W See Jr.
Bachelor of Science, Millersville University of Pennsylvania, 2014

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Master of Science

Grand Forks, North Dakota
May
2018

This thesis, submitted by Timothy W See Jr. in partial fulfillment of the requirements for the Degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

Gretchen Mullendore

Aaron Kennedy

Joshua Hacker

This thesis is being submitted by the appointed advisory committee as having met all of the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

Grant McGimpsey
Dean of the School of Graduate Studies

April 5th 2018

PERMISSION

Title	Implementation of Computational Resources Focusing on Forecast Model Access for Universities
Department	Atmospheric Sciences
Degree	Master of Science

In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work or, in her absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my thesis.

Timothy W See Jr.
April 5th, 2018

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

I would like to awknolwedge my committee members, family, and friends for supporting me through my studies at the University of North Dakota. Without their support and guidance this research wouldn't have been possible.

I would like to dedicate this research to my family. Their belief in my studies, while rarely understanding 'what' I do, helped to ensure my progress forward. I also would like to dedicate this work to Leon Osborne. Leon offered drastically different opinions and points of view from other committee members. This required me to think outside my comfort zone and apply my research to more real world applications. Thank you for your constant questioning, Leon.

ABSTRACT

This research combines recent technological advances (i.e. Docker software containers, and cloud computing) in conjunction with the WRF model to create a portable NWP educational module. Incorporating the educational module into a senior level numerical methods course at the University of North Dakota allowed for analysis of the economic, policy, and privacy issues associated with using new software/tools. It also allowed for documentation of the benefits of tool usage within the classroom, both positive and negative. Use of these new tools also allowed for greater computational power to be accessed within the university computer labs. While the university labs were fully capable of running this module, Docker was unable to be installed due to security concerns. Through usage of Amazon Web Services cloud computing platform, the security issues associated with Docker were mitigated. Compliance of policy with the University of North Dakota was no issue, due to their policy not restricting the usage of cloud computing within the educational environment. The economic impact that cloud computing has on the university was found to be minor and easily reduced with proper AWS instance monitoring and use of the AWS Educate program. Use of these tools within the educational environment allowed for students to be exposed to hands on usage of NWP models and emerging tools. While these tools are associated with an increase in cost for the university, that cost is minimal in comparison to the workforce preparedness that can be generated from tool exposure. Due to the benefit associated with exposure to these tools, the recommendation of this work is to use the module within a cloud computing

environment, such as AWS, to help better equip students for their careers following graduation.

CHAPTER 1

INTRODUCTION

Breakthroughs in numerical weather predictions (NWP) often revolve around technology. Some recent technological advances include emerging tools such as software containers and cloud computing. These emerging tools are most frequently being used within the private sector, government funding agencies, and universities. The Big Weather Web (BWW), as part of a National Science Foundation grant, is investigating the use of these new emerging computational tools. The BWW is creating a sustainable big data infrastructure for university members of the NWP community. Discovering the benefits and limitations of cloud computing in an educational setting is the primary goal of the University of North Dakotas inclusion in the BWW.

The exposure to emerging tools is beneficial to a students future career. Private sector companies, such as The Weather Company, have shifted from traditional computing to a more cloud based operation (AWS 2017), and are therefore looking for graduates with cloud computing skills on their resumes. Additionally, as the author learned at a 2017 workshop on atmospheric modeling in the cloud, funding agencies, such as the NSF, are looking to more frequently fund proposals that prioritize a cloud computing research aspect. Educational institutions are also utilizing the cloud; Indiana University has created a cloud computing server know as Jetstream through NSF funding. (Indiana University 2016) While exposure to these new tools is clearly beneficial, the process of how best to integrate these tools into the curriculum remains uninvestigated.

The research conducted in this thesis focuses on the combination of Docker software containers, a NWP model, and cloud computing to create a portable NWP model for classroom use. After successful creation of a portable NWP model environment, it was incorporated into a senior-level numerical methods course, ATSC 405, at the University of North Dakota. The course allowed for testing of the workflow advantages and disadvantages of the portable NWP model, through hands on student use. Students were introduced to the new tools and computing environments, which included: Docker container software, the Weather Research and Forecasting (WRF) NWP model (Skamarock et al. 2008), and cloud computing through both lecture materials and a hands-on homework assignment. The WRF is the community standard mesoscale numerical model for the atmosphere, and represented state-of-the-art NWP.

In addition to the student experience, one must also consider additional concerns of importance to the university. One such concern is the cost . Costs include, but are not limited to hardware purchases, IT personnel, and cloud computing charges. A detailed breakdown of the cost comparison of local versus cloud computing use is conducted to highlight the strengths and weaknesses of both scenarios. Further, the local use of cloud computing and software containers is compared through different university funding scenarios, such as a lab with high funding compared with low or no funding.

Another area of concern is security. Implementing new software into a lab environment must be secure for all computers associated with the network. Interviews with the local IT support are presented to explain their views on introducing these products in the university computer labs. For example, security concerns regarding Docker were discussed when first considering local installation. These concerns, and

their possible solutions, are discussed in depth to ensure that multiple usage pathways are possible.

Along with the security considerations, installation and use of new tools must abide by policies set by the institution. Interviews with personnel associated with various levels of the North Dakota University Systems group were conducted. These interviews are discussed in the context of possible policy compliance concerns associated with new technology use.

While the testing of this process is done locally at the University of North Dakota, this education module can be applied to other institutions. Universities with atmospheric sciences degrees do not necessarily offer undergraduate classes in NWP. Of the 8 BWW university members 6 of them have undergraduate degrees in atmospheric sciences or meteorology. Of those 6 universities, 2 require a NWP course to graduate, 2 offer it as an elective, and 2 do not offer it at all. Some possible reasons for not offering the course are that the staff does not have the ability to teach the course, the course does not fit within the curriculum, or the department lacks equipment to successfully teach the course. The education module has the potential to increase the number of NWP classes offered across undergraduate programs. The module may also be used to augment current NWP classes by adding a hands-on model investigation portion to the class.

The hypothesis behind this project is that combining cloud computing and software containers enables educational institutions, with outdated computational infrastructure and or the lack of Linux systems admin support, to affordably access NWP models in the classroom for the benefit of NWP education. This research is anticipated to increase the availability of NWP models for education, combine emerging tools to create a portable NWP model, and discover the benefits and limitations of applying the emerging tools within an educational setting. First, the tools used

within the research are presented. Then in Chapter 3 the classroom implementation process is reviewed. Chapter 4 presents the outcomes of the classroom implementation process. Economic, privacy, and policy considerations associated with the use of these tools, in an educational environment are presented in Chapter 5. Lastly, Chapter 6 presents discussion and comparison between scenarios.

CHAPTER 2

TOOLS

This research used new and emerging tools to create a new easy workflow for NWP model usage. The Weather Research and Forecasting (WRF) model, Docker containerization software, and the Amazon Web Services cloud computing platform comprised the set of tools. WRF is already utilized as an educational tool within the atmospheric sciences community (University of Albany: Kevin Tyle 2016). However, the daunting process of setting up WRF locally within a computing infrastructure can limit student and faculty use. Installation of WRF can be time consuming and requires a different set of skills than setting up an actual model run. Someone with little to no experience in setting up and executing numerical models may fail numerous times, spending weeks to month on before success (Hacker Et al. 2016). This extraneous process may discourage professors, whose time is limited, from learning how to build a new installation and implementation of the WRF. The combination of these new and emerging tools can alleviate the process of configuration and compilation and allow professors to use hands on models to their advantage for teaching. This section briefly introduces each tool and their components utilized in creation of the portable NWP module.

2.1 Weather Research and Forecasting (WRF) model

WRF is an atmospheric scientist community NWP model (Skamarock Et al. 2008). The model is maintained at the National Center for Atmospheric Research (NCAR)

for the community wide usage for teaching, operations, and research. There are two differing models available for public download, the WRF Non-hydrostatic Mesoscale (NMM) model, and the Advanced Research WRF (ARW) model. The WRF ARW model was selected due to the partnership with NCAR and the ties with the BWW research project. WRF was paired with Docker container software to allow for WRF to be easily transferred between computers and operating systems.

2.2 Docker

Containerization software allows for packaging of applications to help with version control, portability, and reduce limitations due to operating system differences (Docker 2016). Container software is comparable to Virtual Machines (VM); the main difference is the that containers only contain necessary components of the operating system (OS), whereas VM's include a full OS (Docker 2016). Docker is one of many containerization software packages available for public use. Docker was chosen over numerous other companies containerization software. Through personal conversation with Josh Hacker, the selection of Docker was due itsetting the standard in the software container community when the educational model was being developed. Furthermore, the open source community and extensive documentation available allowed for relatively easy use of this new software. This enabled the team working to develop the WRF in a container, hereafter WRF in a box, to access all the necessary information regarding creation of this new tool. Creation of WRF in a box was successful through the use of Dockers images, containers, and online repository DockerHub.

2.2.1 Docker Images

Docker uses 'images' to create containers. Images are essentially a snapshot in time of the operating system necessary for an application to execute, along with the appli-

cation and any libraries necessary for execution. All libraries and binaries associated with that specific application must be properly installed and referenced in order for successful image execution. Referencing wrong library paths, or improperly installing a library will result in a failed state for the application to execute. Figure 1 shows an example library dependency for the wrf.exe executable. The command output shows that in order for WRF to properly execute, there must be roughly 50 libraries installed. Again, any missing libraries will result in failure to run the necessary application.

```
[tsee@W0PR run]$ ldd wrf.exe
        linux-vdso.so.1 =>  (0x00007ffe8e74e000)
        libnetcdff.so.6 => /usr/local/lib/libnetcdff.so.6 (0x00007f19a04b8000)
        libnetcdf.so.11 => /usr/local/lib/libnetcdf.so.11 (0x00007f19a0192000)
        libhdf5.so.7 => /opt/hdf5/lib/libhdf5.so.7 (0x00007f199fcbf000)
        libhdf5_hl.so.7 => /opt/hdf5/lib/libhdf5_hl.so.7 (0x00007f199fa8f000)
        libz.so.1 => /opt/zlib/lib/libz.so.1 (0x00007f199f875000)
        libcurl.so.4 => /lib64/libcurl.so.4 (0x00007f199f600000)
        librt.so.1 => /lib64/librt.so.1 (0x00007f199f3f8000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f199f1db000)
        libnuma.so => /opt/pgi/linux86-64/13.2/lib/libnuma.so (0x00007f199f0da000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f199edd8000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f199ea16000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f199e812000)
        libgfortran.so.3 => /lib64/libgfortran.so.3 (0x00007f199e4f0000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f199e2d9000)
        libquadmath.so.0 => /lib64/libquadmath.so.0 (0x00007f199e09d000)
        libidn.so.11 => /lib64/libidn.so.11 (0x00007f199de69000)
        libssh2.so.1 => /lib64/libssh2.so.1 (0x00007f199dc3f000)
        libssl3.so => /lib64/libssl3.so (0x00007f199d9fd000)
        libsmime3.so => /lib64/libsmime3.so (0x00007f199d7d5000)
        libnss3.so => /lib64/libnss3.so (0x00007f199d4af000)
        libnssutil3.so => /lib64/libnssutil3.so (0x00007f199d283000)
        libplds4.so => /lib64/libplds4.so (0x00007f199d07e000)
        libplc4.so => /lib64/libplc4.so (0x00007f199ce79000)
        libnspr4.so => /lib64/libnspr4.so (0x00007f199cc3b000)
        libgssapi_krb5.so.2 => /lib64/libgssapi_krb5.so.2 (0x00007f199c9ee000)
        libkrb5.so.3 => /lib64/libkrb5.so.3 (0x00007f199c70b000)
        libk5crypto.so.3 => /lib64/libk5crypto.so.3 (0x00007f199c4d9000)
        libcom_err.so.2 => /lib64/libcom_err.so.2 (0x00007f199c2d4000)
        liblber-2.4.so.2 => /lib64/liblber-2.4.so.2 (0x00007f199c0c5000)
        libldap-2.4.so.2 => /lib64/libldap-2.4.so.2 (0x00007f199be73000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f19a0729000)
        libssl.so.10 => /lib64/libssl.so.10 (0x00007f199bc05000)
        libcrypto.so.10 => /lib64/libcrypto.so.10 (0x00007f199b81e000)
        libkrb5support.so.0 => /lib64/libkrb5support.so.0 (0x00007f199b60e000)
        libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00007f199b40a000)
        libresolv.so.2 => /lib64/libresolv.so.2 (0x00007f199b1f0000)
        libsasl2.so.3 => /lib64/libsasl2.so.3 (0x00007f199afd2000)
        libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f199adad000)
        libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007f199ab75000)
        libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f199a914000)
        liblzma.so.5 => /lib64/liblzma.so.5 (0x00007f199a6ef000)
        libfreebl3.so => /lib64/libfreebl3.so (0x00007f199a4eb000)
[tsee@W0PR run]$
```

Figure 1: Library dependency output for the wrf.exe executable, showing all necessary libraries and binaries needed for the script to successfully run.

Creating an image can be done by manually installing everything from the terminal command line (through pip, yum etc.), or through a 'recipe' encoded in a DockerFile. Figure 2 is an example DockerFile used to create a WRF input container used within the module. This specific DockerFile created a container for a specific run configuration based on Global Forecasting System (GFS) input data. Line 1

of the DockerFile designates the OS platform to build from, which was a generic CentOS operating system image. CentOS was chosen due to it being a base image with minimal installations of other unused binaries or libraries. Line 3 simply creates a directory within the images, and line 4 makes that the current working directory. Line 6 deposits a tarball within that directory. A tarball compresses and saves many files into a single compact directory or file structure (Linux Tar 2010). When the tarball is unpacked the directory is populated with all the necessary files. Line 8 designates this container as a volume called /wrfinput. Designating a container as a volume allows for it to be mounted to another existing container for referencing of its content.

DockerFiles give explicit instructions for installation of libraries, creating directories, and or adding files. DockerFiles were created to generate all of the necessary images for this project, thus allowing for them to be reproduced using this recipe on any computer with Docker and internet access.

```
FROM centos:latest
MAINTAINER Tim See <twsee6@gmail.com>
RUN mkdir /wrfinput
WORKDIR /wrfinput

ADD wrfinput.tar.gz /wrfinput

VOLUME /wrfinput
CMD ["true"]
```

Figure 2: An example of the DockerFile responsible for creating the wrf input files container for the main WRF program to access.

2.2.2 Docker Containers

A Docker container is the in-use (i.e. instantiated) version of a Docker image. Images have all the necessary components to properly execute the desired programs, but an image is a static representation of those components. Through the terminal command

line or a script, one can use the image to create a running version of that static environment, known as a container. One benefit of using images and containers is that once a container is started, the environment is unique. Any changes that are made to the environment can be saved as a version that differs from the original. Containers can also be stopped and restarted as new from the same image. Any mistakes that prove to be fatal to the computing environment are of little concern due to the ability to recreate the same environment effortlessly. As will be discussed later, this ability proved to be useful within the classroom as students inevitably made some mistakes in working with the WRF model that were most easily fixed by starting a new container from the original image.

2.2.3 DockerHub

Docker has its own online repository known as DockerHub. DockerHub offers free and private storage of Docker containers. Figure 3 shows an example schematic of how a Docker container can be shared to DockerHub. The image is created on a local workstation and then pushed to DockerHub. Any computer that has the correct linkage to the DockerHub repository can then download it. Images can only be edited by those with proper permissions, but any container that can be accessed can be manipulated and stored in a personal DockerHub repository as a separate image.

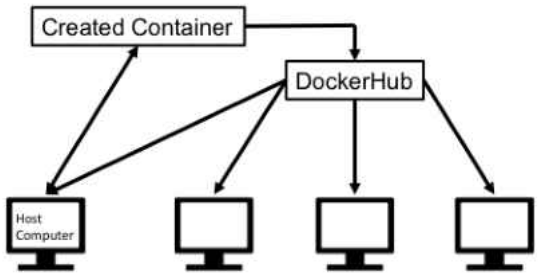


Figure 3: Schematic showing the interconnectivity of local computers, containers, DockerHub, and other computers.

DockerHub enabled easy distribution of the WRF in a box module in the classroom environment. Note that DockerHub is free to use, but that all repositories hosted under free access are public. These repositories may be made private through Docker for an additional cost.

Figure 4 shows an example online repository associated with DockerHub. Users use simple terminal commands to pull down some of the listed repositories, such as twsee/atsci405. An example terminal command would be docker pull twsee/atsci405; this command downloads the image to the users workspace.

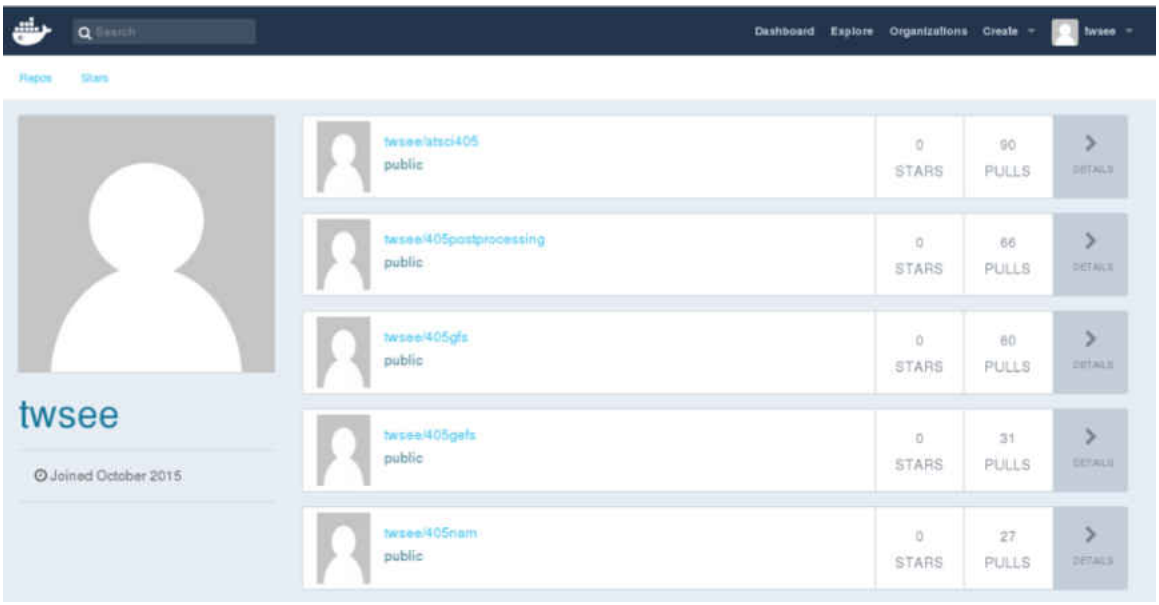


Figure 4: A listing of the containers on DockerHub that were associated with the classroom implementation during the Fall 2016 Semester.

2.3 Cloud Computing

The National Institute of Standards and Technology (NIST) defines cloud computing as an on-demand network with access to a shared pool of customizable computer resources that can be promptly distributed and released with minimal personal management (Mell and Grance 2011). NIST has designated a list of essential characteris-

tics, service models, and deployment models that encompass cloud computing. The essential characteristics are on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured services. The characteristics that applied to this research were the on-demand self-service, broad network access, and measured service. Each student started their own cloud computing instance, needed to have broad network access through some medium, i.e. the internet, and the measured service applied to the charges accrued from each hour of usage from each student.

The service models of cloud computing are Infrastructure as a Service (Iaas), Platform as a Service (Paas), and Software as a Service (SaaS). This research utilized the IaaS portion of cloud computing through Amazon Web Services (AWS). AWS, like many other cloud computing providers, offer these services to the public. The IaaS services allow for the use of a computing environments with complete control over all aspects within the environment (software installation and OS selection), without the need to maintain the physical hardware (Mell and Grance 2011). IaaS would be equivalent to setting up your own computing cluster in the cloud and configuring and compiling WRF from scratch.

NIST also states that cloud computing has four deployment models. The models are private, public, hybrid, and community cloud. This research used the public cloud model. Use of Amazon Web Services is a form of public cloud, which designates that the public must be able to gain access to the cloud in an on demand (Mell and Grance 2011).

There are numerous cloud computing providers available for use, such as Amazon, Google, HP, IBM, and Microsoft. AWS was chosen due to its use within the Big Weather Web project (BWW, http://bigweatherweb.org) that funded this research. AWS also offered financial support in the form of credits to users that were using

its services for educational aspects. AWS Educate allows students and professors to acquire credits through signing up with an education associated e-mail address.

AWS offers many different cloud capabilities, ranging from application development to supercomputing. For this research, the Amazon Elastic Compute Cloud (EC2) service was used. EC2 uses a web service interface that allows users to configure and maintain complete control over their computing resources (Amazon Web Services 2017). EC2 allows a user to designate operating system, storage, and computing power within each instance. An instance is analogous to a personal desktop, where a user can access that desktop via a Secure Shell (SSH) connection. The EC2 service charges per hour for the use of the designated storage, data transfer (based on data amount transferred), and computational infrastructure selected. Supported operating systems varies across cloud providers, and AWS offers several versions of Linux (Ubuntuu, Redhat, etc.) and windows servers dating back to 2003. EC2 offers a variety of different compute-oriented platforms ranging from general computing to storage optimized , which are used for large data sets that require sequential read and write access. (EC2 Storage Optimized Instances 2017) Each platform specializes in a different area of computing; for this research the general computing platforms were used.

CHAPTER 3

CLASSROOM USAGE OF WRF IN A BOX

With the combination of Docker, WRF, and Amazon Web Services, a NWP system was available for use within the classroom. The Fall 2016 Numerical Methods (ATSC 405) course at the University of North Dakota was used as a first test case for the Docker WRF implementation. Students within the classroom had access to their own lab computer and personally started their own AWS cloud instance. Instructions and lectures were given to increase student understanding of the software tools and cloud computing environment.

While the previous chapter presented the software tools in a more general sense, this chapter will cover the specific methodologies by which these tools (and additionally WRF) were implemented by the instructors and then used by the students. All aspects of the classroom implementation process were documented, including the creation of the use case for investigation (Docker Images created), WRF configuration, AWS student access, usage cost analysis, and the issues and successes with implementation.

An assignment was created that required students to change portions of the WRF control run to create an ensemble forecasting system within the classroom. Students ran three separate model implementations for the classroom assignment. One control model run, a data initialization change, and a parameterization change. This generated a class model ensemble with 8 members total. Although the assignment is

referred to in this chapter, a more detailed overview of the entire classroom module and assessment of student learning, including the homework, is given in Chapter 4.

3.0.1 WRF in a Box

Separate Docker containers were created for the WRF model, GFS input data, GEFS input data, and the NCAR Command Language (NCL) post processing portions of the module (see `https://hub.docker.com/u/twsee/`). As stated in the previous chapter, Docker containers are running versions of images, which were generated for this research through the use of DockerFiles.

Besides the ease of use discussed in Chapter 2, Docker containers also guarantee that model output is reproducible (Hacker et al. 2016). Containerization allows for WRF to be used within different operating systems and hardware configurations and produce bit by bit replicable output. Reproducibility is advantageous for research, but also advantageous for education, as the instructors can know a priori the model output, and not worry differences in results that may arise from computational differences such as round-off error or compiler re-ordering of instructions. Having the same reproducible output generated for each student allows for the assignment to focus more on the differences generated from other sources of uncertainty, e.g., parameterizations and initial conditions.

WRF in a box was developed at NCAR, but this study was one of the first use cases and led to a lot of feedback to the NCAR team during the development stages, an overview of the process is given here. The containerized WRF used for this classroom integration differs from the current containerized WRF available from NCAR (`https://github.com/NCAR/container-wrf`) as it is slightly less automated in order to give the students a more hands-on experience.

WRF v3.7.1 was pre-configured to simulation a historical event at the University of North Dakota on a Linux-based research computer. The use case for the model run was a tornadic supercell occurring in North Dakota, on August 3rd 2016. The model domain uses a Lambert conformal projection (figure 5) with 40-km horizontal grid spacing. The model runs begin August 2nd 00 UTC and end August 5th 12 UTC (i.e., an 84-hour simulation).

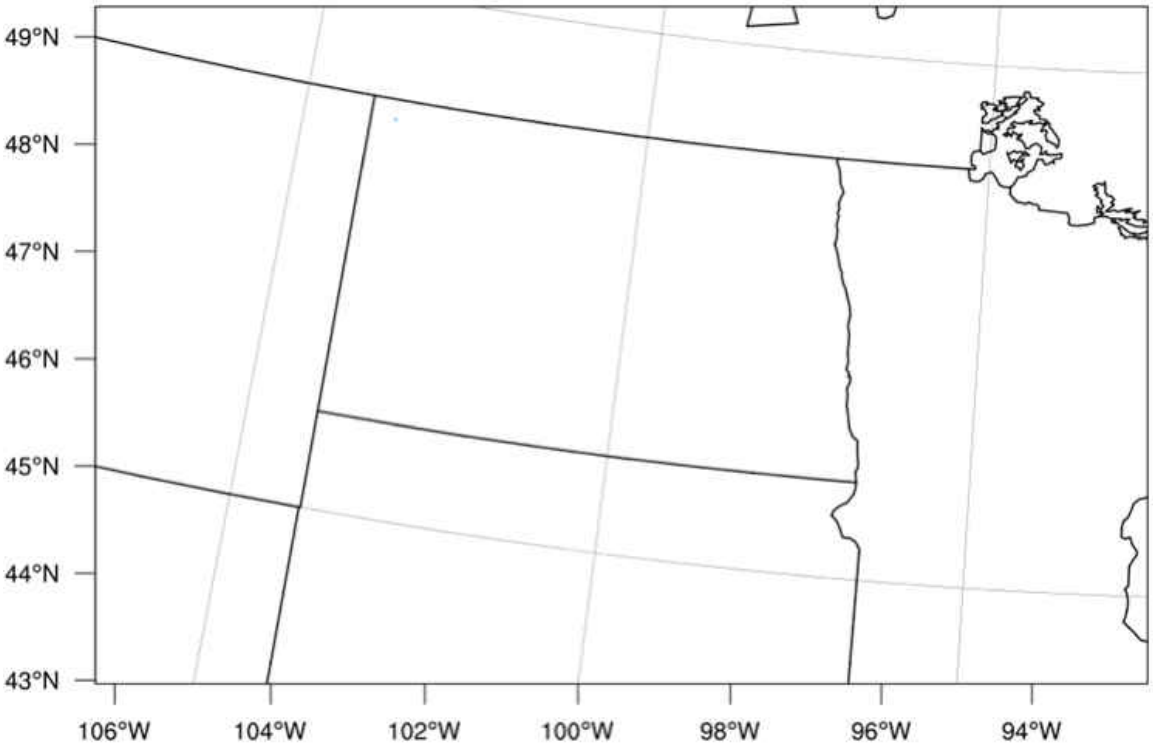


Figure 5: WRF model domain, 40km grid spacing, over North Dakota.

Table 1 shows the associated parameterizations used within the control model run. The selected parameterizations for the WRF control run match the control run used in the BWW project (e.g., `http://schumacher.atmos.colostate.edu/weather/csuwrf_info_bww.php`). The Assigned Changes column shows the parameterization and initialization changes used by the students for the assignment. The modifications to the control run are some of the same modifications chosen to create

Table 1: Full listing of the parameterizations associated with the WRF model run.

Parameterization	Control Run	Assigned Changes
	GFS Input Data	GEFS Input Data
Microphysics	Thompson	WRF Single Moment
Cumulus	Kain-Fritsch	Grell-Devenyi Scheme
Radiation	RRTMG Short and Long	
Boundary Layer	Mellor-Yamada-Janjic	Yonsei University Scheme
Land Surface	Unified Noah Land Surface	
Surface Layer	Eta Similarity Scheme	

the much larger BWW ensemble. Parameterization changes were executed within the Docker containers through VIM text editing of the WRF namelist file.

Two separate input data sets were used within the classroom implementation. The Global Forecasting System (GFS) and a single member of the Global Ensemble Forecasting System (GEFS) were used as initial conditions to generate differences within model runs. The GFS and GEFS initialization data are both half-degree spacing and are produced by the National Center for Environmental Prediction (NCEP). Both of these input data sets were placed in separate Docker containers. These data containers were labeled as volumes, which allowed for mounting onto other existing containers for interaction. This allowed the students to change the initialization data to be pulled into the WRF container. The initialization data (both GFS and GEFS) are used in creation of the BWW ensemble. Note that the GEFS is operationally archived by NCEP at a degraded resolution, but the half-degree data for this case was available through the BWW archive.

Table 2 lists each ensemble member generated by the students. Together the class generated a total of 8 ensemble members. Red cells highlight the parameterization change and/or initialization changes assigned to different groups of students.

The final container created for WRF in a box was the NCAR Command Language (NCL) post-processing container. The NCL software was already being used

Table 2: A list of the ensemble members generated within the classroom. The differences from the control run are highlighted in red and similarities are highlighted in green.

Member Number	Initialization Data	Microphsics	Cumulus	Boundary Layer
Control	GFS	Thompson Scheme	Kain-Fritsch Scheme	Mellor-Yamada-Janjic
2	GFS	Thompson Scheme	Kain-Fritsch Scheme	Yonsei University Scheme
3	GFS	Thompson Scheme	Grell-Devenyi Scheme	Mellor-Yamada-Janjic
4	GFS	WRF Single-Moment	Kain-Fritsch Scheme	Mellor-Yamada-Janjic
5	GEFS	Thompson Scheme	Kain-Fritsch Scheme	Mellor-Yamada-Janjic
6	GEFS	Thompson Scheme	Kain-Fritsch Scheme	Yonsei University Scheme
7	GEFS	Thompson Scheme	Grell-Devenyi Scheme	Mellor-Yamada-Janjic
8	GEFS	WRF Single-Moment	Kain-Fritsch Scheme	Mellor-Yamada-Janjic

by the NCAR team for other WRF in a box cases and was therefore a logical starting point for implementation at UND. The post-processing container is a separate container that references output files from the main Docker WRF container. Five scripts are in the container, four of which generated output plots for each model output file and one that ran all of the scripts in succession. The students used these plots to investigate and reflect on the model results.

A negative aspect of choosing NCL for the post-processing container is that NCL is not taught in the current undergraduate curriculum at UND (students are primarily exposed to C and Python), so students were unlikely to adapt the post-processing scripts to address their own questions. However, the suite of plots provided was sufficient for this implementation, and containerized applications allows

post-processing codes written in other languages to be easily added in future implementations.

3.1 AWS Instance

To access AWS, one can either make a personal account or use a central account, such as a professor account, with generated usernames and passwords for whoever needs access. While creating a personal account can be beneficial for future use, it requires the input of credit card information. To reduce the amount of information required for a student to gain access to AWS, a central account with multiple users was established for the classroom implementation. Dr. Gretchen Mullendore created a master account and separate accounts for each of the students within the classroom. After successful creation of the classroom usernames and passwords students were given a tutorial, located within appendix A Instructions for launching AWS instance, installing Docker, and running Docker WRF, providing step-by-step details for instance set up.

A local computer lab was used to access AWS and utilize Docker in the cloud. The local computational infrastructure used for this project was a university funded lab located in the John D. Odegard School of Aerospace Sciences at UND. The lab computers were unable to have Docker installed due to initial security concerns (see Chapter 5). Docker could then be utilized on AWS without any local security concerns. While this added another layer of difficulty to accessing the workspace, it allowed access to greater computational power and provided easier access for students who might chose to work remotely using their laptops.

AWS offers tiered computing capability and capacity, and the computing instance selected from AWS EC2 was the t2.large. It provided sufficient computing power at a cost that was reasonable for classroom use. The t2.large instance is ideal for processes that usually have a large amount of downtime and then occasionally need

a high amount of computing power in small bursts (Amazon Web Services 2017). The forecast model configuration used for this project was relatively computationally inexpensive; the model run completes on a university funded local lab workstation in roughly 20 minutes . However, should a future model simulation require greater computing power, a different instance may be selected to handle the higher computing needs. The elasticity of cloud computing allows for numerous model configurations to be used regardless of their computational needs (e.g., high or low resolution, added nested grids, global or regional models).

The actual process of signing into and starting an instance can be seen from the tutorial video located at (`http://gretchen.atmos.und.edu/AWS-Instance-Set-Up.mov`). The video demonstrates the steps required to start an AWS EC2 t2. large instance and generate the IP address for subsequent command line access via ssh. This video includes several changes to the default settings needed by for this class, specifically changing the instance type from t2.micro to t2.large and increasing the storage size from 8 GB to 60 GB.

3.1.1 Assignment Cost Analysis

Amazon offers their services based on fees revolving around instance type/uptime, the storage space used, and data transfer fees. As mentioned previously, instance type is chosen based upon the type of computing a user desires to complete the work. Figure 6 represents the different instance types associated with the general computing cloud infrastructure that AWS offers.

Linux | RHEL | SLES | Windows | Windows with SQL Standard | Windows with SQL Web | Windows with SQL Enterprise

Region: US West (Oregon)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
t2.nano	1	Variable	0.5	EBS Only	$0.0059 per Hour
t2.micro	1	Variable	1	EBS Only	$0.012 per Hour
t2.small	1	Variable	2	EBS Only	$0.023 per Hour
t2.medium	2	Variable	4	EBS Only	$0.047 per Hour
t2.large	2	Variable	8	EBS Only	$0.094 per Hour
t2.xlarge	4	Variable	16	EBS Only	$0.188 per Hour
t2.2xlarge	8	Variable	32	EBS Only	$0.376 per Hour
m4.large	2	6.5	8	EBS Only	$0.108 per Hour
m4.xlarge	4	13	16	EBS Only	$0.215 per Hour
m4.2xlarge	8	26	32	EBS Only	$0.431 per Hour
m4.4xlarge	16	53.5	64	EBS Only	$0.862 per Hour
m4.10xlarge	40	124.5	160	EBS Only	$2.155 per Hour
m4.16xlarge	64	188	256	EBS Only	$3.447 per Hour
m3.medium	1	3	3.75	1 x 4 SSD	$0.067 per Hour
m3.large	2	6.5	7.5	1 x 32 SSD	$0.133 per Hour
m3.xlarge	4	13	15	2 x 40 SSD	$0.266 per Hour
m3.2xlarge	8	26	30	2 x 80 SSD	$0.532 per Hour

Figure 6: A chart from Amazon Web Services depicting pricing per computational instance regarding computing power.

The general purpose compute infrastructure, t2.large, is highlighted in yellow within figure 6. The t2.large instance charges $0.094 per hour of instance uptime, which is only charged while an instance is in the running state as shown within the

video tutorial. Note that should an increase in compute power be desired, the cost of the instance per hour uptime would also increase with it.

The use of instances are charged per hour uptime, storage costs increase based on the number of gigabytes (GB) partitioned for use within an instance. As shown in the tutorial video, and as noted in Appendix A: Instructions for launching AWS instance, students were required to change the instance storage configuration from 8 GB to 60 GB. This increases the total cost for each instance. Storage cost is based on a monthly rate, which can be prorated to hourly. The hourly cost changes based upon the type of storage selected. (Since the completion of this classroom implementation, AWS has changed the charging structure to prorate to the second, instead of to the hour. This change is not reflected in this cost analysis.)

AWS storage types range from low to high traffic and differ between Solid State Drives (SSD) and Hard Disk Drives. The class assignment used the Elastic Block Storage (EBS) General Purpose SSD. The use of the EBS General purpose SSD simply was due to it being the default option for the instance selected. No comparisons of the EBS volumes were conducted to assess if one performed better than the other. This was due to the low volume of data generated and the infrequency at which the data was accessed . The cost calculation for the allotted storage used with the EBS is

$$\frac{\$0.1 GB}{Month} * \frac{1 Month}{30 days} * \frac{1 day}{24 hours} = \$0.000139 \frac{GB}{hour} \tag{3.1}$$

and is applied to the account until the instance is terminated. The cost per hour for each instance storage, increased from 8 GB to 60 GB, partitioned increases from \$0.000139 to \$0.0083 per hour.

Within research of atmospheric sciences, the NWP data output files are large, and traditionally stored locally. In contrast, the model output generated from the classroom simulations was kept in the cloud. Students used the post-processing container discussed in Section 3.1 to visualize the supercell evolution and then transferred the images only to their local machine. The visualizations were significantly smaller data files than the model output, which allowed for a lower magnitude of data transfer, reducing the data egress costs. The resulting data egress costs for this study were zero. Data egress costs as per Amazon EC2 Pricing is $0.00 per GB when below 1 GB of data per month (AWS EC2 Pricing 2017). Plots created were 100kb in size, which allows for roughly 1 million of these images to be transferred from AWS to the local machine before costs start to accrue. The classroom as a whole transferred much less than the threshold for charges to begin.

In total, the project tested thirteen students simultaneously creating instances, running the models, and transferring data. Figure 7 shows cost per day during instance use (blue) and storage costs (orange). Billing for EC2 instances only starts when an instance is in the running state (Amazon Web Services FAQs 2016). However, for EBS charges it occurs from the moment the instance is created until termination. In other words, while an instance is stopped there are no hourly instance EC2 charges, but the EBS charges continue. These charges continue to accrue until the instance is terminated (i.e., stopped instances can be started back up for use, but termination wipes them from AWS). Diligent management of instance uptime is crucial to utilizing the economic advantages of Amazon Web Services as an on-demand computing environment.

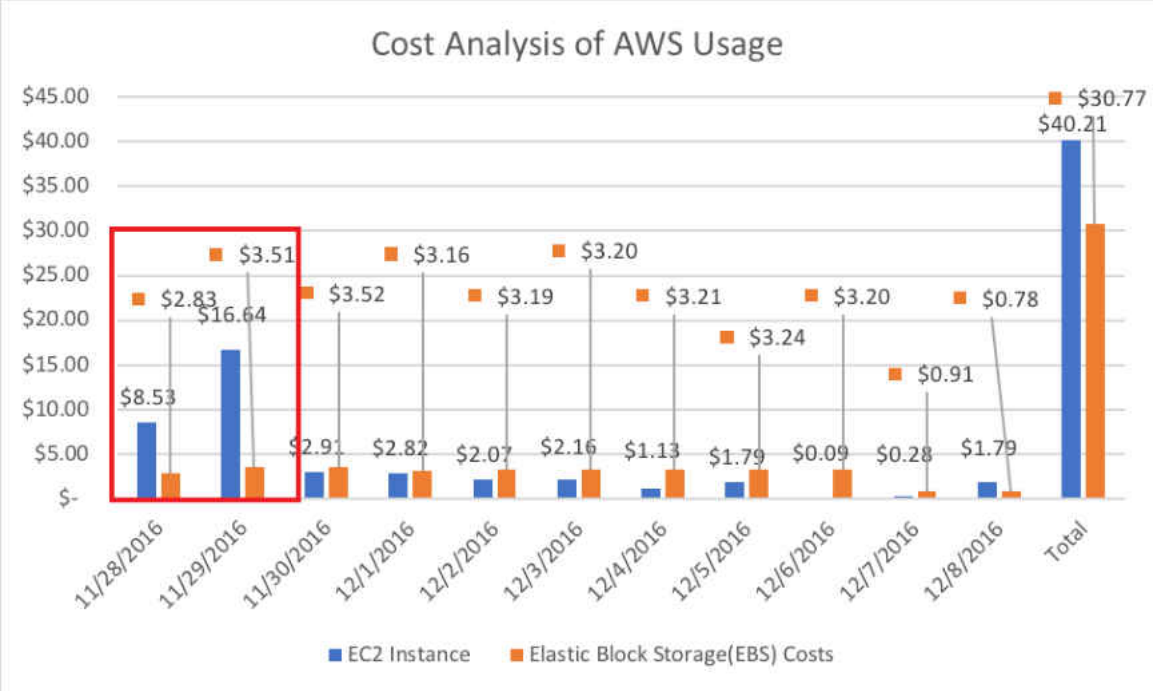


Figure 7: Graph of cost per day of total EC2 (blue) and EBS (orange) usage, including all student generated instances. Y axis is cost in USD, X axis is the date in MM/DD/YYYY format. The red box highlights the large instance uptime charges due to poor instance monitoring.

Most of the days within figure 7 show a low cost per daily use of instance uptime (EC2 costs; blue columns), except for two. November 28th-29th were the days spent introducing cloud computing and Docker. November 28th was the day all instances were created. The class simply created their instance and then connected to them. While the instances were not used after the initial connection, they remained running due to instructor failure to relay the importance of stopping instances after the end of its use. This inflated cost of November 28th-29th is due to the instances running for roughly 36 hours without changing the instance state from running to stopped.

Even with these inflated costs, total cost for AWS integration into the classroom was only $70.98, with the cost per student being $5.46. Future implementations

could easily lower total costs by approximately $20 just by using better instance monitoring. Additionally, the storage allocated for each instance was 60 GB, which in retrospect is larger than the necessary storage for each instance; possible reductions in storage are discussed further in Chapter 5. Proper monitoring of the instances uptime and more appropriate storage allocation size would help to significantly lower the costs of this assignment.

3.1.2 Technical Challenges

As with any implementation of new teaching material and/or new software, issues are certain to be encountered. This section looks at the several specific technical challenges encountered during implementation. Assessment of specific learning outcomes of this module will be discussed within the next chapter.

During instance set up, numerous students had issues connecting to the AWS instances after creation. After some research of the issue, it was found to be related to the AWS security groups associated with the instances. During the creation of each instance, a new security group was created to be referenced for that instance. Students were simultaneously creating their instances. Should a student start creating an instance at the same time as another, it is possible that a single security group could be referenced for multiple instances (particularly as they were all under the same instructor account). This restricted the remainder of classroom time to troubleshooting each of the students instances, rather than introducing them to the new environment and installing Docker. An easy correction was documented so that this should not be an issue for future implementations. By naming the security group of each instance a unique name upon creation of the instance this can reduce the amount of security group overlap when creating instances.

The lack of structured introduction to Docker proved to be detrimental to the students for the remainder of the assignment. The introduction of Docker showed students how to pull images, start containers from images, start and stop containers, and properly remove images and containers. The syntax that was expected to be taught, seen in Appendix A: Docker Introduction and Syntax Tutorial, was necessary for successful completion of the assignment. While step-by-step instructions were given to the students, a mistyped word could break an instruction requiring them to use Docker commands to restart. Without the initial introduction to Docker syntax the students needed extra help for mistakes relating to Docker.

Another unanticipated hurdle was the lack of students experience in command-line text editing. While programming is taught within the curriculum at UND, many students choose to use graphical editors such as gedit and gvim to do their text editing. However, for the software setup used in this module, students could only use command line editing to change text files in the Docker containers. For example, most of the students needed instructor help to edit the namelist.input file within the WRF model. The Virtual Improved (vim) editor was used in this class, and vim has a steep initial learning curve. More instruction is needed on vim to reduce student frustration in future implementations.

CHAPTER 4

STUDENT USAGE ANALYSIS

This chapter assesses the learning goals and outcomes and the student experience of this module. Many of the details of the classroom module were already introduced in Chapter 3; here is presented a brief overview but with added information on how the module fit into the class schedule and the overall undergraduate curriculum. The assessments presented were gathered using various methods including, surveys, homework questions, and instructor observation.

4.1 Classroom Module Overview

The module was incorporated into an the upper level, required, ATSC 405 course at UND in the Fall 2016 semester. The course was designed to introduce numerical methods and their purposes, focusing on numerical problems encountered in the Atmospheric Sciences field. This includes, but is not limited to, how models handle data, model uncertainty, parameterizations, and ensemble forecasting. Pre-requisites for this course were Introduction to Scientific Programming (CSCI 130) and Computer Concepts in Meteorology (ATSC 270). Introduction to Scientific Programming is a computer science course that introduces problem solving, algorithm development, and structure programming using C with an engineering and mathematical focus. Computer Concepts in Meteorology introduces students to programming knowledge needed for manipulating observational and model data in earth sciences (using C, bash, and Python). This includes data visualization, Linux and shell scripting, ad-

vanced file I/O, and memory management. These classes allowed students to utilize Linux environments and some of its tools, such as SSH, to complete the assignment. Having knowledge of a general Linux environment and the use of terminal command line was pertinent to completion of the tasks set for the students. Hence, this particular course was chosen for module testing because the material was a good fit with the existing course learning objectives and the prerequisites ensured a basic level of understanding of the Linux environment.

Two weeks of the sixteen-week class were designated to the classroom module: introduction of Docker, WRF and AWS, one homework assignment, and a classroom discussion. The homework assignment required completion of tasks related to Docker and WRF through the use of AWS. This assignment was used to tie together the use of models, and the associated theory taught in previous weeks. As discussed in Chapter 3, the assignment required students to complete NWP model runs, create output plots, and compare the results. The three tasks for completion were completing the control run, changing the initialization data from GFS to GEFS, and changing a single parameterization. The parameterization change required that the students navigate to the proper directory, provided within the instructions, and replace a value within the namelist.input document. The initialization data change, from GFS to GEFS, was completed through referencing of a separate Docker image that encompassed the associated input data. All of the simulations were completed within the AWS environment.

After completion of all model runs students transferred the output plots, generated within a bash script provided for them, to the local desktop. To assess completion of the simulations and the student experience, students submitted the following items for grading: designated model output plots, feedback regarding ease of use of the module, and a report comparing model outputs to radar observations of the

event. The comparison of model output to radar (and to different model realizations) built upon the causes of model uncertainty learned about previously in the course. The assignment required that students recognize the differences generated from each model setup and apply their knowledge from previous years of classes to understand how the model output responded to changes in parameterizations and initialization (e.g., relating fundamental knowledge of physical meteorology to estimates made in the microphysical parameterizations).

4.2 Learning Outcomes and Goals

Learning objectives of this assignment were to 1) gain knowledge of cloud computing and container software, 2) know how to run a forecast model, and 3) understand model uncertainty. Our expectation was that all three of these learning objectives would benefit from the hands-on nature of the module.

One of the ways the first learning objective was assessed was via a short essay question on the final exam:

> The Weather Research and Forecasting (WRF) model was run in class by using software containers (Docker). Explain a minimum of two advantages of using containerized WRF over running WRF outside of a container.

This question was not answered properly by most students. Many were unable to decipher the differences between Docker and cloud computing. Through instructor discussion and observation, it was clear that many students struggled to differentiate between cloud computing and container software. Equating the Docker software with the AWS instance was common among students. Ultimately, students did not understand why the container software was involved in the exercise. While the students were exposed to these new tools, the learning objective of understanding their usage

and differences failed to be met. In future iterations of this educational module, it should be restructured to establish a distinct difference between the tools being used. This should allow for a better overall understanding by the students.

As all students completed the assignment, the second learning objective, ability to run a forecast model, was met. Having the ability to run a NWP model and interact with its components is greatly beneficial to students (e.g., seeing what is included in the input and output components cements learning more strongly than just hearing about the processes in a lecture). Additionally, having knowledge of running a NWP model is useful for multiple careers and also for graduate school. Several students stated that having experience with WRF prior to going to graduate school was one positive aspect taken away from this assignment. Running the model is not indicative of complete knowledge of the processes behind it, but it does help to lessen the learning curve associated with models should they be exposed to it in the future. The use of this module removes a lot of extensive tasks that need to take place prior to running the model. Data collection, configuration, and compilation are a few steps that can require weeks or months of work to set up based upon a users knowledge level of the processes.

The final learning objective was to gain an understanding of model uncertainty. Students spent the previous weeks in the class learning about the theories behind NWP. This class in previous years only used a 1-D model to investigate numerical instability. However, with the addition of this new assignment, students were also able to apply their knowledge of parameterization and initialization errors by manipulating the model configuration and analyzing the output. Through generating multiple model outputs and analyzing the differences, students were able to identify the changes associated with each of the models. This enabled them to analyze that a

change in a microphysical parameterization has effects on other processes rather than just microphysics.

Every change within the model has broad impacts, related to more than just that specific part of the model. The differences may be widespread, some minute in magnitude and others much more drastic. Students recognized the differences generated within the model outputs within their analysis and comparison section of the homework assignment. Most, if not all students, alluded to the fact that every single measured quantity that had plots generated for it changed to some degree within the model output. Their discussion sections of the homework showed that they did in fact recognize the differences generated within each of the model outputs. Analyzing and deciphering the differences generated from changes allows for uncertainty to be understood through hands-on analysis rather than theory.

4.3 Pre-and-Post Assessment Analysis

A pre-and post self-assessment questionnaire was given to the students to help quantify the success of this classroom module. Table 3 shows the full survey and the pre-and post average scores for the class. The pre-assessment was written before the module itself was finalized and taken by students at the beginning of the semester. Giving the assessment at the beginning of the semester most likely resulted in a lot of the increase in confidence for the students. The materials taught leading up to the module covered much of the content asked about within the survey.

Several of the questions were asked as control questions to ensure that a true self-assessment was conducted on the students behalf. Questions 1 and 3 were asked strictly to demonstrate that their numbers should not increase. Question 1, Your ability to configure and compile a forecast model on a computer from beginning to end, was not demonstrated or taught by any means. Question 3, Ability to locate and

Table 3: A comparison of pre and post assessment averages for the class through the implementation of the WRF in a box module. Confidence ranked from 1 being least confident to 4 being most confident.

	Fall 2016	
	Pre	Post
1) Your ability to configure and compile a forecast model on a computer from beginning to end.	1.2	3.0
2) Confidence to explain why models arent perfect to the general public.	2.9	3.9
3) Ability to locate and download necessary model input data.	2.2	3.0
4) Confidence to explain what contributes to model error.	2.3	3.6
5) Manipulate a forecast models configuration to change parameterizations utilized.	1.2	2.9
6) Manipulate a forecast model to use different input data.	1.3	3.3
7) You have a high level of understanding of instrument error.	2.6	3.4
8) You have a high level of understanding of model truncation error.	1.3	3.1
9) You have a high level of understanding of model parameterizations.	1.3	3.1
10) You have a high level of understanding of a staggered grid and its benefits within models.	1.2	2.6
11) You have a high level of understanding of ensemble forecasting.	2.2	3.7
12) Confidence to interpret ensemble forecast products.	2.3	3.8
13) Ability to explain ensemble forecasting benefits.	2.5	3.6

download necessary model input data, was also not covered in the lessons. Prepared input data was given to the students. The increase of confidence for students for these two portions of the assessment should not have occurred. As discussed in Hacker et al. (2016), compilation and configuration is extraneous to actual model understanding, and is a task that may take weeks to months to perform.

The increase in confidence level indicates an inflated self-assessment. A study by Dunning (2004) states that self-assessments are flawed because people overestimate themselves, holding inflated views of their expertise, skill and character. This can, in part, be attributed to the vast increase of the average in Questions 1 and 3 for the pre-and post assessment comparison. It is likely that students also do not yet understand the process of compilation or input data gathering. Students lack of insight to the configuration, compilation, and gathering of input data processes likely can be attributed to the large increase in confidence.

Questions 2 and 4 show that students increased their ability to explain model uncertainty to the public and their understanding of model uncertainty. While the increase in confidence is not surprising, it is unclear whether the increase is due to the materials taught over the course of the semester or from the model portion of the homework. Questions 5 and 6 show an increase in confidence to manipulate parameterizations and input data. While students did complete the tasks within the assignment, this is with a specific set of instructions for a unique task. Their confidence lies within this WRF in a box model, and not within other forecasting models. The increase is realistic but potentially misplaced in terms of forecast models as a whole. Questions 12 and 13 are associated with the model output generated from the assignment. As part of the assignment, students interpreted multiple plots from a small ensemble forecasting system allowing them to see the differences generated be-

tween all model runs. The increases in confidence associated with these two questions are likely a result of the assignment itself.

Questions 7 through 11 were worded vaguely and offer little insight to the confidence levels. It is unclear as to what a high level of understanding is, and a students threshold of high level is likely variable between students. The overall take away from this assessment is that it was poorly constructed and offers little insight to the actual outcomes of the assignment in terms of student confidence. Conducting the assessment immediately before the module, and consulting with an educational assessment expert, could help create a more useful survey.

CHAPTER 5

ECONOMIC, POLICY, AND PRIVACY CONSIDERATIONS

One of the goals of this research is to investigate the broader implications of using software containers and cloud computing in an academic setting. A university considering a change from local computing to cloud-based computing needs to consider cost differences, and whether such a change complies with university policies on computing and privacy. In order to discuss these considerations, three scenarios are presented: cloud computing (use case and the classroom module presented in this research), a low-budget computer lab such as might be supported within an individual department or research group, and a university-funded (i.e., well-funded) computer lab meant for broad use across the university. Each of the computing scenarios includes a description of computational infrastructure: hardware, operating system, and expected length of use/update time. The time between upgrades is a primary difference between a low-budget lab (8-10 years) and a university-funded facility (roughly every 3-5 years).

As part of the data collection process for this chapter, interviews[1] were conducted with University of North Dakota IT support within the Scientific Computing Center (SCC), Computational Research Center, and the North Dakota University System Core Technology Services (NDUS CTS) regarding the economic, policy, and

[1]Following the interviews the Computer Science department was transferred to the College of Engineering. Interviewee comments that may reflect the change of department were not gathered or used in the analysis here.

privacy considerations for usage of cloud computing within the classroom. The SCC provides support to the John D. Odegard School of Aerospace Sciences (JDOSAS), the Computational Research Center provides high performance computing support to the University of North Dakota researchers, and the NDUS CTS provides secure information management and technology services to the NDUS. Personnel interviewed within the SCC were Andrew Kuelbs (Lead Client Support), Derek Stinchfield (Systems Administrator), and John Wold (Information Systems Manager). Aaron Bergstrom (High Performance Computing Specialist) from the Computational Research Center and Darin King (Chief Information Officer) of NDUS CTS were also interviewed.

5.1 Computational Scenario

5.1.1 Cloud Computing Infrastructure

Amazon Web Services offers a multitude of computing services. For the classroom implementation process we utilized only the Amazon Elastic Compute Cloud (EC2). EC2 provides scalable compute environments that eliminate the need to invest in hardware upfront (Amazon Web Services 2017). EC2 provides a user with virtual computing environments known as instances, access to preconfigured operating system templates known as Amazon Machine Images (AMIs), various configurations of CPU, memory, storage, and networking capacity for instances, secure login information, storage volumes for temporary data, dynamic cloud computing , metadata, and virtual private clouds isolated from the rest of AWS cloud (Amazon Web Services 2017).

The specific cloud computing instance used within the classroom was on the t2.large hardware. The t2.large instance consists of 2 Intel Xeon 2.4 GHz vCPUs (virtual central processing unit), 8GB RAM, and a variable amount of storage set

during creation of the instance (Table 4). T2 is one of several instance choices and it provides a baseline level of CPU performance. The base operating systems available for quickly launching an instance are the Amazon Linux AMI, Red Hat Enterprise Linux 7.3, SUSE Linux Enterprise Server, Ubuntu Server 16.04, and many Windows server versions.

For this specific classroom use, the Ubuntu Server 16.04 was chosen as the operating system. The Ubuntu Server 16.04 was chosen because of its low costs (due to Ubuntu being open source OS) and because the Ubuntu environment was comparable to the operating system used at the university at that time, Mint OS. AMIs such as the Ubuntu AMI are updated by AWS as Ubuntu released new OS versions . Updates to the operating system within a previously created instance must take place by the user, i.e., when an instance is started, it will remain the same until the user personally prompts the upgrade. The storage volume size of the instance used was changed from the default size of 8 GB to 60 GB. This ensured enough storage was allocated for installation of Docker WRF, the generated output files, and plots. Finally, security group configuration ensured that only a student with their unique generated key was able to access their created instance.

5.1.2 Low Budget Lab

At the University of North Dakota, Dr. Leon Osborne leads a computing center known as the Regional Weather Information Center (RWIC). RWIC is used for student computing for several classes taught at both the undergraduate and graduate level. The computing power within RWIC is comprised of completely repurposed machines from the University of North Dakota Streibel Hall computing lab.

RWIC is a prime example of a low budget computational infrastructure. The lab consists of 14 Linux machines. The computing power within the lab is variable

due to computers being acquired at different times throughout RWICs lifetime, but a computer representative of the lab has a single Intel Core 2 Duo @ 3.00Ghz CPU, 250 GB storage, and 8 GB RAM. The machines acquired were excessed by the Scientific Computing Center within the university during an upgrade, and given to the RWIC center. The machines were converted to the Mint OS to ensure homogeneity through the department. The periodicity of hardware upgrades within the lab is variable and relies on the availability of new refurbished equipment.

5.1.3 University Funded Lab

The University of North Dakotas computer science department has numerous computer labs set up for educational use. The specific lab used by the fall 2016 ATSC 405 course contained 36 computers. The lab hardware consisted of Dell Optiplex 990s with 8GB of RAM, 250GB SATA Hard Drives, and 1 Intel core i5 2500 3.30 GHz CPU (Table 4). The most recent upgrades to hardware were in February of 2012. Hardware upgrades are not on a set schedule; they are upgraded when the computer science department has the money available from the university for upgrading. All the desktops are standalone machines, so the computational work is done locally. The lab computers use a shared home directory server hosted in the Scientific Computing Center as primary storage. All the lab computers used Mint OS. Mint is a free open source OS that is community driven while also safe and reliable (Linux Mint 2017). Requests for new software installation, such as Docker, on lab computers can be completed as early as the same day, or may take up to three days. After new software is requested, its installation time is dependent upon the amount of investigation that is necessary to ensure safe incorporation into the network.

Table 4: Computational comparison of each computing scenario, comparing CPUs, cores, computations per second, disk space, RAM, total compute power, and cost.

Computational Comparison	CPUs	Cores	Computations /Second	Disk Space	Memory (RAM)	Total Compute Power	Cost
t2.large Instance	2	8	2.40GHz	Variable	8GB	38.4GHz	0.094/hr
University Funded	1	4	3.30GHz	250GB	8GB	13.2GHz	$986.00
Low Budget	1	2	3.00GHz	250GB	8GB	6GHz	$0.00

5.1.4 Infrastructure Comparison

A comparison between the computing infrastructure in each scenario is necessary to properly assess the impacts of using one scenario over another. To highlight the compute differences, Table 4 compares single machines in the t2.large AWS instance, University Funded lab (the University of North Dakota Streibel Lab), and a low-budget lab (the University of North Dakota RWIC lab). The table shows the differences between each scenario in terms of their compute power, storage, RAM, and cost. Disk space within the cloud is variable and cost increases with storage requirements. Table 4 shows the available raw compute power for each scenario. Note that total compute power is an upper limit; increases in compute power may not be realized if software is not optimized properly.

The low-budget lab has the lowest compute power per machine at 6 GHz, university-funded lab has more than twice as much computing power as low budget (13.2 GHz), and the cloud computing instance has roughly three times as much computing power as the university funded lab (38.4 GHz). Note that there are other various factors and way to determine peak performance. However, due to the inability to measure in other ways, clock speed (Table 4 Computations/Second) will be our deterministic value for computer power. For this specific implementation process the use of cloud computing was not necessary. The university funded lab computers were more than sufficient to run the Docker simulation. While it was possible to run

the simulation, installing and using Docker within the lab environment is prohibited. More information on Docker local installation is provided later in this chapter. There is little to no difference in the operating systems associated with any of the scenarios. However, this may differ for other universities and potentially cause issues due to differences in syntax between operating systems. For example, use of Docker on a Windows machine as compared to a Linux machine will require different referencing file and directory path referencing (i.e. / for Linux and
for Windows).

5.2 Economics of Computational Infrastructures

This section presents a more in-depth assessment of the costs of each computational scenario summarized above. To generalize these results, we assume a classroom size of 30 students for all implementations, and each student completing the assignment within 6 hours of compute time used.

Table 5 shows the costs per classroom breakdown for a university-funded lab using cloud computing, a low-budget lab using cloud computing, a university-funded lab (no cloud component), and a low budget lab (no cloud component). The table includes the costs of IT support, which is assumed to be a quarter year salary of a full time IT support staff member. This quarter time salary assumption is due to the fact that IT staff member would not only support the lab computers, but they would also support the lab servers. It is assumed that the IT staff member is fully engaged in other university duties the other 9 months of the year (i.e., paid by the university to conduct other duties).

These scenarios all utilize Linux OS. Linux OS requires specialized system administrators to configure and maintain hardware, operating systems, and software (James 2016). James states that skilled and reliable Linux system administrators

are hard to find and have a mean salary ($69,380) comparable to that of a tenured professor (James 2016) . The mean salary within an educational institution is based off of a 2,080 yearly working, which translates to a $33.35/hr wage. The required assistance expected from the systems admin is quarter time employee for the duration of the use (1 month) which totals at $66.71 . The total is calculated with 10 hours per week (quarter time employee) for 4 weeks and is split amongst the classrooms associated with the lab for the entire year. For the 2016/2017 school year, the Streibel Hall 115 lab held approximately 20 classes, so the cost is split up between each of those classes for systems admins support. Calculation is shown below.

$$\text{IT Cost} = \frac{\$33.35}{hr} * \frac{10hrs}{week} * 4weeks * \frac{1}{20classes} \tag{5.1}$$

Calculations for cloud costs are based upon 2 weeks of instance set up by a professor (assuming 6 hours of instance compute time, and 2 weeks of storage time),

$$\text{Professor Cost} = \left(\frac{\$0.094}{hr} * 6hrs\right) + \left(60GB * \frac{\$0.1GB}{month} * \frac{1month}{30days} + 14days\right) \tag{5.2}$$

and 2 weeks of student use, with 30 students (assuming 6 hours of instance compute time with 2 weeks of storage time).

$$\text{Student Cost} = \left(\frac{\$0.094}{hr} * 6hr * 30students\right) + \tag{5.3}$$

$$\left(60GB * \frac{\$0.1GB}{month} * \frac{1month}{30days} * 14days * 30students\right)$$

The Class breakdown shows the amount each class pays to use the computing resources for one month out of the year.

Table 5: Cost analysis for each lab environment with and without cloud computing use for per class comparison.

Class Breakdown	IT Support	Hardware Costs	Compute Costs	Storage Costs	Total
University funded using Cloud Computing	$ 66.71	$ 24.65	$ 17.48	$ 89.51	$ 198.36
Low budget lab using Cloud Computing	$ 66.71	$ 12.33	$ 17.48	$ 89.51	$ 186.03
University funded	$ 66.71	$ 24.65	-	-	$ 91.36
Low budget	$ 66.71	$ 12.33	-	-	$ 79.04
Personal laptops	-	Personal	$ 17.48	$ 89.51	$ 106.99

5.2.1 Cloud Computing Infrastructure

The cost of cloud computing is based off instance uptime, hardware selected, memory/storage allocated for use, and the location of the services. AWS provides an online cost calculator (available online at; https://calculator.s3.amazonaws. com/index.html) for estimating costs due to instance type selected, location, operating system, estimated uptime, and any additional desired services paired with EC2 in advance of requesting an allocation. The t2.large instances used for this research charged $0.094 per instance hour uptime. The general expected uptime of each instance, in a proficient class, should not exceed 6 instance hours per student. Using the online Amazon price calculator shows that in an idealized scenario the two weeks of class use of the AWS Docker WRF should cost no more than $17.48 for 31 instances at 6 hours uptime (30 students, 1 professor). This low cost per instance is in part attributed to the current low compute time of the WRF model as configured for the UND classroom (coarse resolution over a limited domain). Any sort of manipulation to the WRF model simulation would likely result in an increase in needed computational power or runtime, which would result in increased cost for the homework assignment.

Storage costs per instance are in addition to the computational charges discussed above. AWS charges for Elastic Block Storage, which is the amount of storage set within the instance created. For the classroom implementation the storage was 60 GB per instance. The storage costs, as discussed in section 3.3 (Assignment Cost

Analysis), are charged whether or not the instance is stopped or started, and recurs until the instance is terminated. Storage costs are per GB-month and prorated to the nearest hour. With the GB-hour cost coming to $0.000139, the total cost for the entire assignment, assuming 2 weeks to complete, increases the total costs by $86.80 to a total of $104.28. The increase from 8GB to 60GB was to ensure the capacity within the instance was more than enough to successfully hold all data. After further investigation, the use of 15GB of storage would suffice for the instance size, and decrease the cost of storage from $86.80 to $21.70. Reducing the total from $104.28 to $39.18.

However, with any new module to be included in the classroom, professors need to test the material. Therefore, total AWS costs for the entire semester would likely increase due to professor testing of the module in the cloud prior to classroom activities. Professor costs have been considered in Table 5, adding an additional 6 hours of instance uptime and 2 weeks of storage.

While the cost of cloud computing is relatively cheap, the use of either lab scenario adds to the cost. The added cost of cloud computing almost doubles the cost of use within the classroom for the single month of use.

5.2.2 University Funded Facility

When the Streibel Hall lab hardware was upgraded in February 2012 the cost per computer was $986 for each of the 30 lab computers. This puts the periodic upgrade cost at roughly $29,580. The depreciated value of the hardware over the upgrade periodicity (5 years) is $5,916.00. While the costs of upgrades are outdated, this can be used as a template to estimate costs of periodic hardware updates. The educational computer lab uses Linux operating systems. The JDOSAS lab currently has a single systems administrator who devotes an estimated quarter of his time to lab upkeep

and the rest of it to the server upkeep. The cost for a class to use the university funded labs for the assignment totals at $91.36 without cloud computing.

As stated previously, the use of AWS within this computational environment was not needed for completion of the homework assignment. This should be taken into account when looking to use cloud computing within a lab environment.

5.2.3 Low Budget Lab

The RWIC computer lab located at UND is comprised of entirely repurposed machines, so the hardware costs are assumed to be zero for its economic assessment. Therefore, the yearly lab cost of hardware and IT support within a low budget lab environment, such as RWIC, depends solely on IT support costs. However, within Table 5, we include hardware costs over a 10-year depreciated value, using the same budget for the February 2012 hardware upgrade of the university funded lab mentioned above. This is to highlight the fact that not all low-budget labs may have the ability to acquire free refurbished or repurposed hardware from elsewhere. The costs of classroom use of one month for a low-budget lab without cloud computing comes to $79.04. We were unable to test Docker in this environment, but these computers are approximately as powerful as the authors university provided research computer which has a total clock speed of 6.00 GHz. The research computer successfully completed the simulation, with longer simulation times associated with the runs.

The use of the containerized WRF within this computational environment may not be viable, and is dependent upon the modeling details and specific hardware. In such a scenario, cloud computing will have to be used. The use of cloud computing increases the cost to $186.03, more than doubling the cost of local hardware only. While the increase in cost is significant, it guarantees the students are able to complete

this module and homogenizes the student experience, a possible issue is a lab filled with a motley assortment of surplus machines.

5.2.4 Economic Analysis Summary

For use of AWS within any computing scenario there is an estimated increase of cost by $106.99. The cost increase is accompanied by greater computing power and introduction to new and beneficial computational tools. While the use of cloud computing increases costs overall, it benefits the students to learn new tools for their future careers. The increased expense versus the benefit to exposure to cloud computing is the fundamental tradeoff that will need to be assessed by each school considering adoption of this cloud-based learning module.

During interviews conducted with university personnel there was a large emphasis on the question Who funds the use within the classroom?. The use of AWS was seen as a great solution to using Docker WRF within the classroom, but this research was funded under a research grant (NSF). For regular use in the classroom, other funding sources must be identified. Within AWS there is AWS Educate, mentioned previously, which allows for students and faculty to apply for educational credits to use toward computing with AWS. As of 5/4/2017, AWS educate offered $200 credits per educator (at member institutions) and $75 dollars at a non-member institution, which can be renewed yearly (UND is a non-member institution). For each student, AWS educate also offered $100 in credits at member institutions and $40 in credits at non-member institutions (AWS Educate 2017). For the class containing 13 students, which is offered every year, an educator doing the same computing as mentioned previously theoretically should not have an issue in gaining the credits to cover a majority of the cost of computing for the homework assignment. To obtain these credits, the students would have to sign up with a credit card, which was not done

for the classroom testing in this project. Additionally, there is no guarantee that free AWS Educate credits will remain available in the future.

5.3 Policy Considerations

This section is focused around speculated changes to the computer labs used within JDOSAS and how the changes may affect the implementation process. Currently the North Dakota University System (NDUS) is moving toward centralization of hardware and services within the universities. This was expected to have some effect on the location of labs and/or on personnel that support the labs. The following information regarding policy reformation is from interviews conducted with Dr. King and Mr. Bergstrom and can be found within Appendix B.

The goal of a more centralized infrastructure through a transition from existing systems to functional consolidation is to realize cost efficiencies and improved quality of service(Feldner 2016). Initially there was concern that the JDOSAS Linux labs would be affected, either by changing their operating system or relocating the labs themselves. The consensus from the interviews is that the computing environment within JDOSAS will remain untouched by centralization. The computing infrastructure within JDOSAS is specialized compared to the rest of university. The scientific computing and educating done within JDOSAS is based within the Linux operating system. JDOSAS needs to have specialized Linux system administrators and staff for maintenance of the computer labs. Mr. King stated that labs as a whole are not affected by the centralization; the closer you are to a system the easier it is to maintain. In Mr. Bergstroms interview, he noted that even if there was a relocation of the IT help for JDOSAS, the centralized IT team would provide the needed Linux administration.

Policy concerning cloud computing within the university was also a topic of discussion during interviews. Mr. King commented that the use of cloud computing is not prohibited within the classroom, and it is viewed as a more campus level decision rather than consolidation. Also, cloud computing is already used by various entities at the university. The usage of cloud computing within the classroom does not conflict with any policies, but the university will not pay for something that potentially could be done locally within their current hardware infrastructure.

5.4 Privacy Consideration

Privacy considerations involve the security concerns associated with local Docker installations and the usage of cloud computing within an educational environment. These security concerns include a discussion of the Family Educational Rights and Privacy Act (FERPA), which was enacted by the U.S. Department of Education to ensure student information confidentiality. To gain a better understanding as to why Docker was unable to be installed on lab computers, interviews with Mr. Stinchfield and Mr. Kuelbs were conducted.

5.4.1 Docker Local Installation

Local installation of Docker was not completed prior to the testing of this teaching module. Originally, this project had meant to test both local and non-local (i.e., cloud) use of Docker. The reason for solely non-local use of Docker was due to the possible security risks for the university. The main security concerns regarding Docker are related to the fact that Docker has root access to the system. While the student may not have root access, they have the ability to download any container from the DockerHub repository online. Instantiating a container online is a security risk because containers can possibly hold malicious software. While the downloading

of a malicious container may be unintentional on the part of the student, it remains a security issue for the network.

A statewide policy mentioned by Mr. Kuelbs is that any computer within the network cannot affect another workstation. A malicious container could contain a script to further the network access of that user through Docker. The script may contain superfluous requests to the network causing a denial of service or create a spam bot, as a few examples. Regardless of the type of malicious software in place within that container, it hinders the ability of another user to utilize that computer efficiently. For Docker to be used safely within a university setting either the images used for running containers must be vetted and stored locally or Docker must be customizable for each user.

Note that after further investigation into Docker installation, Mr. Stinchfield and Mr. Kuelbs stated that they could allow for Docker to only access a certain set of local in-house containers for use by the lab users. This would limit which containers a student could download, making Docker more secure to install on shared lab machines. With that security measure, Docker was deemed a software package that can be implemented into the UND lab environment.

5.4.2 Docker Cloud Computing Installation

One of the benefits to using AWS is that it offloads any security concerns that may arise from locally installed software. By moving Docker into a cloud computing instance, any malicious software that is installed onto the instance remains there and is isolated from the local computing infrastructure. However, while using cloud computing helps to reduce the security issues surrounding malicious containers, it potentially causes another privacy issue. FERPA protects the privacy of student educational records, including but not limited to student name, address, class list,

and grades. The use of AWS within a classroom environment could potentially lead to leaking information such as student name, class name, and university name outside of the network. While this is a concern, an AWS FERPA Whitepaper states that AWS provides services to set security controls that meet the data privacy and data security requirements to ensure protection of education data in compliance with FERPA (AWS FERPA Whitepaper 2015). AWS abides by the policy to protect sensitive information within their cloud computing environment.

While AWS provides the ability to create secure environments, it has proven to be difficult to set up. The way permissions get set is very difficult to understand, even from an IT admin point of view. Mr. Kuelbs stated in his interview that the AWS permissions are tedious and difficult to understand. The initial attempt at creation of security permissions for the classroom failed. Following a white paper published by AWS on example classroom permissions (Amazon ECS IAM Policy Examples 2017) allowed for the author to reduce student access to many AWS applications. The permissions were so restrictive that students users would be unable to launch instances, and were thus not followed because they would have inhibited the students in completing their work. That resulted in additional vulnerabilities. For example, with the permissions set for our instance use, the students had the ability to interact with one anothers instances. One student could start, stop, and terminate another students instance causing disruption to someones work. Use of a more secure set of permissions is recommended for further usage of AWS within the classroom.

The consensus from all interviews is that AWS was a good solution to possible security risks introduced by using of Docker within the classroom environment. The on-demand style of AWS for a short period of time, and off-site computing, ensures that the use of Docker WRF within the class is not detrimental to the security of the local network.

Table 6: Subjective rankings for computing scenarios for policy compliance, economic impact, security and overall categories. 1 being the best and 4 being the worst.

	Policy Compliance	Economic Impact	Security	Rank
University funded using Cloud Computing	1	4	1	2
Low budget lab using Cloud Computing	1	3	1	1
University funded	1	2	2	4
Low budget	1	1	2	3

5.4.3 Summary

To summarize the findings in this chapter simply, a table has been created to reference the ranking of each of the computing scenarios based upon the economic, policy and privacy information gathered. Table 6 shows the rankings of each scenario from 1 to 4. With 1 being the best choice for that particular aspect and 4 being the worst choice.

All scenarios receive equal marks for policy compliance. Due to there being no issues regarding using cloud computing or installing new software that is blessed by IT, they are all an equal decision regarding policy.

The economic impact is directly related to the cost of each computing scenario. The low budget lab receiving the top rank due to these labs, such as RWIC, having the ability to acquire hardware without any upfront purchasing costs. The university funded lab using cloud computing received the worst mark. Overall for economic impacts, the majority of the impact can be reduced due to the use of AWS educate, allowing for each of the cloud computing scenarios to be very competitive with local computing.

Security ranks are tied between the two sets of computing scenarios, local and cloud computing. Cloud computing adds an extra layer of security to the already in place lab securities. It also reduces the issues associated with Docker installation on a computer lab environment. Disallowing for images pulled from the web to have direct access to the local computational infrastructure within the university. The

local usages of the module receives lower security markings due to the vulnerabilities regarding Docker installation locally that were discussed in this chapter.

After reviewing the economic, policy, and privacy considerations for each of the computing scenarios, each has their strengths and weaknesses. The use of university labs, whether they are well funded or not, allows for a reduced cost due to not using cloud computing. The use of Docker within the local environment poses some security threats to the system, but can be alleviated if the IT support has the knowledge to do so. Alleviation of the threats can be done through vetting containers or restricting user access to unknown Docker containers.

While the use of cloud computing adds additional cost, it alleviated the possible security risks associated with Docker and allowed for access to increased computational power. As stated earlier, the usage of cloud computing also promotes use of tools that are being used in future career options. Because of these additional benefits, it is the opinion of the author that provided there is a way to cover expenses of AWS, through AWS Educate or some technology fee associated with the class, the usage of cloud computing is the preferred scenario.

CHAPTER 6

CONCLUSION

Recent technological advances (i.e. software containers, and cloud computing) were used in conjunction with the WRF model to create a portable NWP module. The goals of creating this module were to investigate how the use of these new technologies might reduce barriers to module use in the classroom setting. In addition to simplifying access to WRF, the module would also increase student exposure to modern computing environments. With private sector and even funding agencies gravitating towards these new tools, exposure earlier in a students academic career can be of great benefit. This research created the teaching module, but also investigated the student experience in the classroom, the economic scenarios under which this module might be applied, and compliance with university privacy and computing policies.

The research conducted helps to enlighten the need for introduction and use of emerging tools within education. Use of these new tools during classes allows for workforce preparedness for graduating students. Private sector businesses such as The Weather Company are utilizing these tools within their work environment. Government agencies are also pushing to use more tools such as container software. The National Aeronautics and Space Administration (NASA) is utilizing Docker software containers for some of their numerical models. (NASA Supercomputing 2017). They have containerized their NASA Land Information System Framework due to the dif-

ficulty of non-experts to install. Introducing students to such tools can allow for a wider skillset that may potentially be used within a future career.

Conducting the implementation within a real classroom environment allowed for insight to be provided regarding the issues and successes with the process. The information gathered allows for others, that wish to use such modules, to learn from past processes. This helps to reduce the issues associated with setting up and using these modules and may help to increase the success of others usage.

Major outcomes of this research include creation of WRF in a box, a teaching module and the investigation of economic, security, and policy impacts of AWS on institutions. WRF in a box was made possible through the use of new emerging technologies. Docker paired with AWS enabled WRF to be effectively placed into an educational environment and used without the need for updated computational infrastructure or systems administrator support. Use of these tools proved to be difficult initially for students but their confidence increased as they continued to work on the assignment. Student exposure prior to graduation may help to increase their workforce readiness due to private sector, government agencies, and universities using these tools within their workflow.

A teaching module incorporating hands-on NWP model use allowed for students to experience how models are run and their associated outputs. The learning goals associated with this research were each individually met. First, the students gained knowledge of cloud computing and container software through using them while completing their assignment. Second, each student successfully completed their model runs for analysis, successfully completing the NWP class assignment. Lastly, each student recognized the widespread difference caused by parameterization and initialization differences which are both sources of uncertainty, a significant aspect of NWP.

The use of AWS and Docker within an educational environment had some economic and security impacts on the university. The policy impacts that were considered, mainly centralization, had little to no effect upon implementation of these tools into the classroom setting. Using AWS required funds to be placed aside for assignment completion. The economic impact of using AWS was just over $100. With the utilization of AWS Educate credits the price can be mitigated further. The security concerns associated with use of Docker in the local lab setting were mitigated through by using AWS, which allowed for computing off-site.

Based on the research conducted during the implementation of WRF in a box into the University of North Dakota Fall 2016 ATSC 405 classroom, it is recommended that the use of cloud computing paired with Docker is the best course of action for implementation of WRF in a box. The usage of cloud computing is recommended regardless of the computational infrastructure set in place within the institution. The reason for this decision is that the benefits of introduction to tools needed for career development far outweigh the economic costs of cloud computing. With private sector, graduate schools, and funding agencies moving toward cloud computing and software containers it is beneficial to expose undergraduate students to the tools potentially being used in their future careers. Experience from the in-class exercise suggested that the exposure to new tools also outweighs the burden of increased learning curves for both professor and students.

While the recommendation is to use cloud computing and Docker, there are many other avenues that are possible for use. If the local computing infrastructure allows for Docker to be seamlessly and securely incorporated into the system, then local use is possible. The local use of WRF, without cloud computing and Docker, is also a viable option. One scenario that was not tested, with the exception of a few students within the fall 2016 class, is to require a laptop or access to a personal

computer for enrollment in the class. An assumption can be made, given the technological advances in past years, that most students have access to a home computer or personal laptop. The use of personal laptops removes the financial burden, totaling $91.36 for a funded lab and 79.04$ for a low-budget lab, for that class to utilize the lab computers for class time. Two students in the Fall 2016 classroom were able to complete this module on their personal laptops instead of the computers within the lab. Requiring laptops for class enrollment would reduce the cost for implementation. Only cloud computing costs are incurred, eliminating costs for IT support and physical hardware at the university.

The process outlined for usage provided in this document is not rigid, and there are numerous ways to utilize the module. The authors hope is other universities will be able to follow this plan to integrate emerging tools such as cloud computing and software containers into the undergraduate curriculum. Ultimately, this research provides universities with the basic information needed to decide on the implementation method that will work best for their institution .

The research presented in this thesis has much room for improvement. Delivery and structure of the entire educational module could be manipulated to allow for better delivery to the students. The restructuring of the educational module would allow for more focus to be placed on the theory behind NWP and how it is applied within these tools, rather than deciphering between the tools. With some minor issues fixed through this first application, more time can be allotted for proper introduction of the new tools. This would help with the students ability to separate cloud computing from the concept of containers, which proved to be a challenge during the implementation process. The restructure could move the assessment to a more useful timeline within the class, allowing for more useful results to be collected. Use of a professional

to create the confidence assessment would allow for a better understanding how the module affected the students knowledge, as well as better assessment questions.

Cleaning up of the trivial code would allow for more focus to be place analyzing output rather than stressful syntactical errors frequently generated when following directions. This module can also be expanded to incorporate other investigative aspects of NWP. For example, setting up a model run to fail due to exceeding the Courant-Friedrichs-Lewy condition. This would allow for students to do an investigation of module code and apply mathematical solutions learned in previous class. There are countless ways to improve this module related to NWP for student learning experiences.

Two comparisons that should be investigated is the use of local versus a cloud computing NWP, and the use of personal laptops versus lab computers. Due to the extensive time it took to create this module, comparison of local versus cloud and laptop versus lap computers were unable to be properly assessed and incorporated into this research.

REFERENCES

Amazon EBS Pricing., 2017: Amazon EBS Pricing. Amazon Web Services, INC. [Available online at https://aws.amazon.com/ebs/pricing/]

Amazon EC2 Pricing., 2017: Amazon EC2 Pricing. Amazon Web Services. [Available online at https://calculator.s3.amazonaws.com/index.html]

Amazon EC2 Storage optimized Instances., 2017: Amazon EC2 Instances. Amazon Web Services.[Available online at https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/storage-optimized-instances.html]

Amazon Web Services., 2017: Amazon Elastic Compute Cloud: User Guide for Linux Instances. Amazon Web Services, Inc. 817pp

Amazon Web Services FAQs., 2017: Amazon EC2 FAQs. Amazon Web Services, INC. [Available online at https://aws.amazon.com/ec2/faqs/]

Amazon Simple Storage Service., 2017: Amazon Simple Storage Service: Getting Started Guide. Amazon Web Services, Inc. 19pp

AWS Case Study: The Weather Company, 2017: AWS Case Study. Amazon Web Servics, INC. [Available online at: https://aws.amazon.com/solutions/case-studies/the-weather-company/]

AWS Educate, 2017: Teach Tomorrows Cloud Workforce Today. Accessed 05 May 2017. Amazon Web Services. [Available online at https://aws.amazon.com/education/awseducate/]

AWS FERPA Whitepaper, 2015: FERPA Compliance on AWS. AWS Whitepaper. 17 pp, [Available online at http://d0.awsstatic.com/whitepapers/AWS_FERPA_Whitepaper.pdf]

Docker, 2017; Dockerfile reference. March 2017, [Available online at https://docs.docker.com/engine/reference/builder/]

Docker Docs, 2017; Overview of Docker Hub. March 2017 [Available online at https://docs.docker.com/docker-hub/]

Dunning, D., C. Heath, and J. M. Suls, 2004: Flawed self-assessment: Implications for health, education, and the workplace. Psychol. Sci. Public Interest, 5, 69106, doi:10.1111/j.1529-1006.2004.00018.x.

Feldner, L., 2016: North Dakota University System Information Technology - Strategic Plan 2017-2019 NDUS Information Technology, 30 pp.

Hacker J., Exby J., Gill D., Jimenez I., Maltzahn C., See T., Mullendore G., Fossell K.. 2016: A containerized mesoscale model and analysis toolkit to accelerate classroom learning, collaborative research, and uncertainty quantification. Bull. Amer.Meteor. Soc., accepted for publication.

James C.N., Weber J. 2016: Cloud Computing in Oceanic and Atmospheric Sciences. Cloud Computing in Education, Vance T., Merati N., Yang C., Yuan M., Elsevier, 107-120.

Indiana University 2016; Jetstream, first NSF-supported cloud infrastructure for science & engineering research, to launch September 1. IT News and Events March 2017, [Available online at https://itnews.iu.edu/articles/2016/jetstream,-first-nsf-supported-cloud-infrastructure-for-science–engineering-research,-to-launch-september-1.php]

Linux Mint, 2017: About Us. Linu Mint, 6/13/17, [Available online at https://www.linuxmint.com/about.php]

Linux Tar, 2010: tar(1) - Linux man page, 05/02/18,[Available online at https://linux.die.net/man/1/tar]

Mell, P., Grance, T. 2011: The National Institute of Standards and Technology Definition of Cloud Computing. U.S. Department of Commerce Special Publication, 7pp.

Mullendore G.L., Tilley J.S., 2014: Integration of Undergraduate Education and Field Campaigns: A Case Study from Deep Convective Clouds and Chemistry. Bull. Amer Meteor. Soc., October 2014.

National Oceanic and Atmospheric Administration, cited 2016: Numerical Weather Prediction. [Available online at https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/numerical-weather-prediction.]

National Aeronautics and Space Administration Supercomputing, cited 2018: Containerizing the NASA Land Information System Framework. [Available online at https://www.nas.nasa.gov/SC17/demos/demo29.html]

Singh, KD., Zhadanovsky, L., 2013: Setting Up Multiuser Environments in the AWS Cloud (for Classroom Training and Research). AWS Whitepaper. 21 pp,

https://d0.awsstatic.com/whitepapers/aws-setting-up-multiuser-environments-education.pdf

Skamarock, W. C., and Coauthors, 2008: A description of the Advanced Research WRF version 3. NCAR Tech. Note NCAR/TN-475+STR, 113 pp, doi:10.5065/D68S4MVH.

Sultan, N., April 2010: Cloud computing for education: a new dawn? International Jounral of Information Management 30 (2), 109-116.

Thompson, G. Field, P.R., Rasmussen, R.M., Hall, W.D, 2008: Explicit Forecasts of Winter Precipitation Using an Improved Bulk Microphysics Scheme. Part II: Implementation of a New Snow Parameterization. Mon. Wea. Rev., 136, 5095-5115.

Warner T.T., 2011: Numerical Weather and Climate Prediction, Cambridge, 526 pp.

APPENDIX A

CONTAINS ALL MODULE DOCUMENTS

APPENDIX A

DOCUMENTS FOR LECTURE MATERIAL, HOMEWORK ASSIGNMENT, PRE/POST ASSESSMENT QUESTIONAIRRE, AND INSTRUCTIONS FOR DOCKER/AMAZON WEB SERVICES SET UP

The following pages include the documents used during the Docker WRF implementation into the Numerical Methods Course during the Fall 2016 semester at the University of North Dakota.

Cloud Computing Lecture Notes _____ **1 page**

Introduces cloud computing to the class and touches base on some known challenges and benefits.

Docker Introduction and Syntax Tutorial _____ **4 pages**

Introduces Docker software and what it is used for. After discussing what Docker is, it then goes through a tutorial for setting up AWS, install Docker, and utilize the common commands they will need to complete the assignment.

Weather Research and Forecasting Model (WRF) Notes _____ **2 pages**

Introduces WRF with differences between each type of WRF model available and compares workflows of the models.

Big Weather Web Notes _____ **1 page**

Summarizes the NSF Big Weather Web Project, the primary funding source for development of this classroom module.

Instructions for launching AWS instance, installing Docker, and running Docker WRF ___ **4 pages**

Tutorial for accessing AWS, installing Docker, and running the Containerized WRF.

1

Assignment: Docker WRF Container **2 pages**

Homework assignment given to the students after the lab session that launched AWS and installed Docker and Containerized WRF was completed.

Pre/Post - Assessment Questions **1 page**

Assessment questionnaire for quantification of student experience using cloud computing and Docker WRF.

Instructions for creating classroom environment for AWS usage **4 pages**

Instructions for instructor use of AWS and setting up class usernames, permissions, and access to the cloud environment.

Cloud Computing Lecture Notes

What is Cloud Computing?

Cloud computing is a way to access data, programs, and applications over the internet instead of your computer's hard drive. The cloud is simply a metaphor for the internet. Cloud computing is a way to reduce the need for physical servers, personnel, and hassles of upkeep.

Benefits

Reasonably priced compared to costs of hardware, system support and maintenance.

Provides computing resources that can be expanded or adapted to changing needs.

Does not require hardware maintenance.

Containerization is possible for usage on practically operating system.

Reduces the need for costly and time consuming data backups.

Access to cloud is easy and convenient from anywhere with internet access.

Challenges

Dynamic cost structure requires constant oversight and monitoring.

Parallel processes can be slowed due to lack of physical proximity between cloud networks.

It is still necessary to maintain OS and software.

User insecurity when storing data remotely within the cloud.

Computations are subject to latency when/where internet connectivity is unreliable.

(James and Webber 2016 Poster)

Simple examples of cloud computing

E-mail, Microsoft Office online, general internet browsing.

Amazon Web Services Products

Computation, Storage and Content Delivery, Databasing, Networking, Developer Tools, Management Tools, Security/Identity, Analytics, Game Development, Mobile Services, Application Services, Enterprise Applications.

How are we using cloud computing?

Use of Amazon Web Services to create Linux instances to install Docker and run WRF within the cloud. This way of running WRF highlights the ability of cloud computing to be used within an educational setting. It reduces the time it takes to complete the compilation and configuration process for set up and allows for results to be equivalent amongst all students (reproducibility). Increasing the accessibility to students within universities that may not have a computational infrastructure that supports Numerical Weather Prediction.

Docker Introduction and Syntax Tutorial

What is Docker?

Docker is a way to containerize an application. Containerization of an application means including all of the library dependencies necessary for an application to run.

What is a library?

It is a collection of non-volatile resources used by computer programs often to develop software. It can be anything from configuration data to subroutines and classes. programs that are dependent upon certain libraries will only run if they are all present.

Example output for dependency check for wrf.exe. Not all are listed.

```
[tsee@WOPR run]$ ldd wrf.exe

  libnetcdff.so.6 => /usr/local/lib/libnetcdff.so.6 (0x00007f7f34d25000)
  libnetcdf.so.11 => /usr/local/lib/libnetcdf.so.11 (0x00007f7f349ff000)
  libhdf5.so.7 => /opt/hdf5/lib/libhdf5.so.7 (0x00007f7f3452c000)
  libhdf5_hl.so.7 => /opt/hdf5/lib/libhdf5_hl.so.7 (0x00007f7f342fc000)
  libz.so.1 => /opt/zlib/lib/libz.so.1 (0x00007f7f340e2000)
```

For WRF to run its executable, these libraries must be installed on the machine it is running on.

How does Docker handle the dependencies?

When building an image, you start with a base version of a Linux OS, for our case Centos. From there you add the necessary libraries that you need for the application you want to run. So all of the above listed libraries must be incorporated to the Docker container while building it.

Example installation packages prior to building WRF.

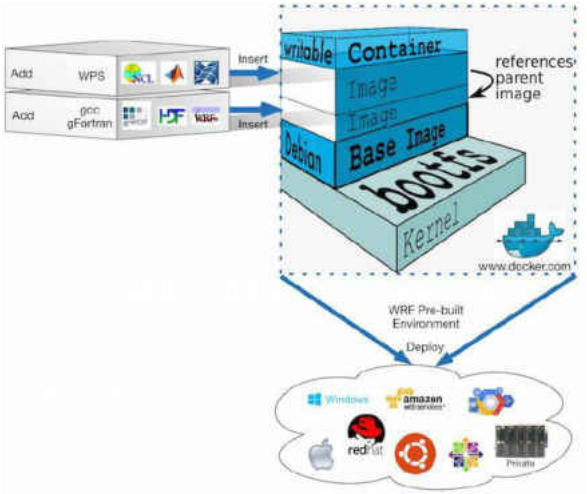
```
yum -y update


yum -y install bzip2 file gcc gcc-gfortran gcc-c++ glibc.i686 libgcc.i686 libpng-devel jasper jasper-devel hostname m4
make openssh-server openssh-clients perl tar tcsh time wget which zlib zlib-devel epel-release


yum -y install netcdf-openmpi-devel.x86_64 netcdf-fortran-openmpi-devel.x86_64 netcdf-fortran-openmpi.x86_64 hdf5-
openmpi.x86_64 openmpi.x86_64 openmpi-devel.x86_64
```

With Docker, you create images and from those images you start containers.

An image is representative of a car engine that is not started. All the necessary parts are there for it to work, but it has not yet been started. The Docker container is representative of the car engine after is has been started.



Once an image is built, it can be pushed to a repository (an online central place for data aggregation) specifically hub.docker.com. When the owner pushes it, it creates a snapshot of that entire container allowing for others to pull it and use it as necessary. This allows for a state to be maintained, so if the container works on the owners' local computer, it will work on any other computer with Docker installed.

Useful Docker Commands

docker ps	lists all running containers
docker ps -a	lists all containers running and stopped

ps and ps -a will have this output form

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES

docker images	lists all images pulled from docker hub

images will have this output form

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE

docker rm "container id"	removes specified container pertaining to the ID
docker rmi "image id"	removes specified image pertaining to the ID
docker pull "repository/name"	pulls the specified repository from dockerhub
docker start "container id"	starts, in the background, a container that was stopped
docker attach "container id"	allows access to the started containers terminal line

Using the "Docker Instructions for Running WRF" we will start an AWS instance, download Docker, and then do some commands to get more comfortable with usage of the Docker syntax.

Complete steps 1, 2, and 3 on that document.

Then we will run these commands:

- docker images
  - no information will be provided under all columns
- docker ps
  - no information will be provided under all columns
- docker ps -a
  - no information will be provided under all columns
- docker pull "repository name"
  - for this example we will replace "repository name" with "centos"
  - this will pull a centos repository and create an image
- docker images
  - there will be information provided under all columns
- docker ps
  - no information provided under all columns
- docker ps -a
  - no information provided under all columns
- docker run -it --name "testcontainer" centos
  - we are within a centos linux instance with the base functions already installed.
  - exit the container by typing "exit"
- docker ps

    - no information provided
- docker ps -a
    - there will be information under each column with the name you gave the container in the "run -it" command. this container is stopped because we exited the bash script
- docker start "testcontainer"
    - this container is now started and available to be accessed.
    - If you ever accidentally exit your container during the homework assignment this will be how you enable it to be accessed again to get the terminal line.
- docker ps
    - there will be information provided in each column referring to the "testcontainer" we had created before.
- docker attach "testcontainer"
    - this will place you back within the terminal line.
    - if you exit the container it will again be stopped and you will need to run "docker start" again to re-access it.
- docker ps -a
    - this will list the container ID again
- docker rm "container ID"
    - this will remove the container, useful if you run into errors within the container due to changes made.
- docker rmi "IMAGE ID"
    - this will remove the container, if something fatal should occur where the container is not working properly even after removal of the old container and reinitiation of a new container from the parent image, use this command to remove the image entirely
    - if you remove the image entirely simply re pull it
        - docker pull "repository name"
    - then follow the directions

Weather Research and Forecasting Model (WRF) Notes

What is WRF?

- A next generation mesoscale numerical weather prediction system designed for atmospheric research and operational forecasting needs.
- It features two dynamical cores, a data assimilation system, and a software architecture facilitating parallel computation.
- It serves a wide range of meteorological applications across scales from tens of meters to thousands of kilometers.
- WRF can generate atmospheric simulations using real data (observations, analyses) or idealized conditions.
- Uses a range of different parameterizations for Microphysics, Convective schemes, Radiation, and the Boundary Layer.

Characteristics of WRF

- Two different versions of WRF, ARW and NMM
    - NMM – Nonhydrostatic Mesoscale Model
        - Fully Compressible, Non-hydrostatic model with a hydrostatic option
        - Sigma-pressure vertical coordinate system
        - Arakawa E-grid Staggering
        - Full physics options for land-surface, planetary boundary layer, atmospheric and surface radiation, microphysics, and cumulus convection.
        - One-way and two-way nesting
    - ARW – Advanced Research WRF
        - Fully Compressible, Non-hydrostatic model with a hydrostatic option
        - Terrain following, dry hydrostatic-pressure, with vertical grid stretching, top of the model is constant pressure surface
        - Arakawa C-Grid Staggering
        - One-way and two-way nesting
            - Nesting is a finer resolution model run that covers a portion of the parent domain.
            - One-way – The information exchange is strictly downscale, the solution of the nest does not feedback into the overall solution
            - Two-way – the nest feedback impacts the coarse-grid domain solution
        - Real and Idealized
            - Real – takes input data such as the NAM, GFS, or GEFS and uses data at the time step to create an input grid to do its calculations on. It requires pre-processing of the input data to properly represent that grid points within the model.
            - Ideal – allows for a user to create a controlled scenario, typically not requiring any input other than the namelist.input which is essentially instructions for the model to run.

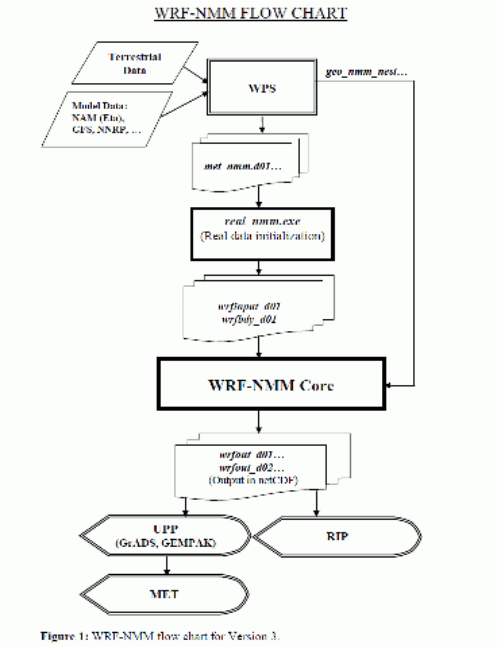
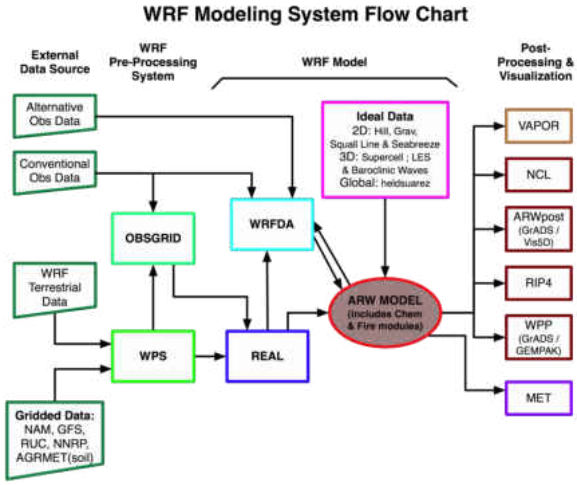


Figure 1: WRF-NMM flow chart for Version 3.

(http://www.dtcenter.org/wrf-nmm/users/overview/model_overview.php)



(http://www2.mmm.ucar.edu/wrf/users/model.html)

Big Weather Web Notes, specifically related to the NSF grant that funded this research.

The Big Weather Web (BWW) project is aimed at the combination of virtualization (AWS and Docker), federated smart storage (AWS), and big data management.

The project is centered around creating an ensemble forecasting system aimed at research. We populated an ensemble matrix that has 48 different members varying from the control run. We are using an AWS instance to store all our data in one central location to remove the necessity of data upload and downloading. The AWS instance allows for post processing to take place within the cloud and reduces separate entities need to download data to their local computers.

The BWW is also aimed at increasing reproducibility within the research environment and "push button" deployment of complex systems (such as our Docker WRF model). Use of Docker is being tested to create an environment where the data is reproducible.

"The Big Weather Web will be developed in the context of numerical weather prediction with the expectation that the resulting infrastructure can be easily adapted to other data-intensive scientific communities"

Instructions for launching AWS instance, installing Docker, and running Docker WRF.

1. Launching an AWS instance
    a. If "Account" is empty, input **und-atsc405**
    b. Go to https://und-atsc405.signin.aws.amazon.com/console and login with your assigned login name and password designated by Gretchen.
    c. Look under All Services
        i. Under "Compute" select EC2
        ii. click the blue "Launch Instance" button
    a. Select **Ubuntu Server 16.04 LTS (HVM), SS Volume Type 64-bit.**
        i. fourth option from the top
    b. The selected Instance type is t2.micro
    c. Change the type to t2.large
        i. click "Next: Configure Instance Details"(Bottom right hand corner of browser)
        ii. click "Next: Add Storage"(Bottom right hand corner of browser)
            1. Under the column "Size(GiB)" change from 8 to 60 GiB (fourth column)
        iii. click "Next:Tag Instance"
            1. under "Value" Type your first initial last name then click Review and Launch
        iv. click "Next: Configure Security Group"
            1. leave the security group name, and description the same
            2. Under the column labeled "Source" select the dropdown menu and choose "My IP"
                a. You will see the IP address to the right change, if you choose to do this on a different computer you will need to change the security group permissions before SSH'ing into the AWS instance
        v. Click "launch" in the bottom right had portion of the screen.
        vi. First time launching an instance
            1. Select "Create a new key pair"
            2. name the keypair, you will need to reference this numerous times
                a. The name of your keypair will replace "Your Key name.pem" in the instructions. (Use something similar to the ATSCI login ex. tsee_aws_key)
            3. Download the key and save the key within the directory you wish to work from.
            4. Then launch the instance
            5. click the blue "View Instances" button
        vii. If you have created one before
            1. Select "Choose an existing key pair"
            2. Then launch the instance
            3. Click the blue "View Instances" button.
    d. This will take you to a dashboard showing you where your instance is with numerous columns of information. Looking at "Instance State" wait until the circle is green and says "running"
    e. Right click the instance and choose connect, there will be a pop up ("Connect to Your Instance") giving you steps on how to ssh into the instance

i. Keep this open you will reference it at a later time

2. Connecting to your AWS instance
    a. Open a terminal and go to the directory that has the key downloaded from earlier steps.
    b. Use this command: chmod 400 Your Key Name.pem
        i. this just limits the access, keeping it more private which allows you to access the instance by agreeing with the privacy settings of amazon.
    c. Then within popup labeled “Connect to Your Instance” on the web browser under the title “Example” copy the line that looks like
        i. “ssh -I “Your Key Name.pem” ubuntu@ec2-IPADDRESS.us-west.compute.amazonaws.com”
    d. Paste that within the terminal to ssh into the instance.
        i. In the terminal it will ask if you want to connect, type “yes”.
    e. You are now within connected to your instance.
3. Install Docker
    a. To install Docker, type in the terminal line
        i. wget -qO- https://get.docker.com/ | sh
        ii. This command accesses a script written by Docker, which alleviates our need to install from source.
    b. After installation is complete, we must join the docker group. this will allow for us to reduce unnecessary syntax when running docker commands.
        i. In the terminal line type
            1. sudo usermod -aG docker ubuntu
        ii. type “groups” in the terminal line will output
            1. Ubuntu adm dialout cdrom floppy sudo audio dip video plugdev netdev
            2. docker will not be listed on this line until we log back in
        iii. Now we must log out then log back into the instance to successfully joint he docker group
            1. In the terminal type “exit”, this will log you out of the instance.
            2. Now repeat the ssh command from above to re log back into the
        iv. After reconnecting to the instance, type “groups” in the terminal line again.
            1. You should see “docker” added to the end of the groups list.
4. Docker instructions from pull to postproc (gfs)
    a. Pulling necessary Docker images
        i. docker pull twsee/atsci405
        ii. docker pull twsee/405gfs
        iii. docker pull twsee/405postprocessing
        iv. docker create -v /wrfinput --name gfsinput twsee/405gfs
            1. #creates the mountable volume of wrfinput for attaching to atsci405
    b. create a directory within the instance and name it wrfoutputgfs, use the command pwd to find its full path and insert below where “/path/to/output/folder” is located
        i. most likely the path will be look like “/home/ubuntu/wrfoutputgfs”
    c. docker run -it --volumes-from gfsinput -v /path/to/output/folder:/wrfoutput --name 405gfswrf twsee/atsci405
        i. places you within the container
        ii. run the command “./run-wrf” to start the model run

iii. Roughly should take ~20 minutes to run and finish
iv. After completion:
1. ls -lR /wrfoutput/*.nc | wc -l
a. the output should be 29
v. type “exit” to get out of the container

5. Creating the post processing container for the model output created above
a. docker run -it -v /path/to/output/folder:/wrfoutput --name postprocgfs twsee/405postprocessing
b. this places you within the NCL container
i. run the command “./ncl_run_all
1. this will loop through all the ncl files and place the output plots within the /wrfoutput directory we mounted to the container.
ii. run the command ls -lR /wrfoutput/*.png | wc -l
1. the output should be 112 .
c. type “exit” to leave the container
6. Moving the data from the AWS instance to your local computer
a. open a terminal on the local computer
i. you will use the same ssh IP address as you did to ssh into the instance previously
ii. go to the directory which has your key you generated from the AWS console
iii. type: scp -i name_of_key.pem ubuntu@ec2-ipaddress.us-west.2.compute.amazonaws.com:/path/to/output/folder/*.png /path/to/local/folder/
1. You must include ***.png**
b. should be output showing the copying progress for each plot
c. Check your local file path you transferred to.
7. To change the input data, simply replace “gfs” with “gefs” for all of the docker commands.
a. this will create a new image to be references, container to be made, and wrfoutput files for post processing
b. I recommend creating a separate wrfoutputgefs in the same location as wrfoutputgfs

After you are done within the instance for the day, go to the webpage that has the instance dashboard and right click the started instance and select the “Instance state” drop menu and choose “Stop”.

To reconnect to that same instance simply right click on your instance, and select start. Once the light is green, you may connect like previously done. However, the IP address will be different so be sure that you’re passing the proper IP address.

1. If you have closed out of the website, here is how to get back to the instance page.
a. Go to cosole.aws.amazon.com
i. if you are still logged in, click on EC2 from the same location as earlier.
ii. instead of clicking on Launch Instance, click on “Running Instances” located above it.
iii. You will see the same columns from earlier wand be able to locate your instance by the Name tag, then reconnect as usual.
iv. **If you are accessing this instance from a different location**
1. Make sure the key you are using is available

2. On the main screen that shows all of the instances, click on yours and in the window below it that shows “Description” “Status Checks” “Monitoring” and “Tags”
    a. Under the “Description” tab, on the right hand side there will be a hyperlink next to “Security Groups” click on it.
    b. On the bottom window it will have “Description” “Inbound” “Outbound” “Tags”
    c. Click the Inbound tab, and click on the “Edit” Button below it
        i. Click “Add Rule” and Change “Type” to SSH and “Source” to My IP

Do not under any circumstances use the <u>TERMINATE</u> option only STOP, this will be done by Gretchen, Tim or Ben after the homework has passed the due date.

Assignment: Docker WRF Container

Instructions: Using the Docker tutorial handout from class complete 3 separate forecast runs.

1. Complete the control run by following the instructions for GFS
2. Change initialization data from GFS to GEFS
3. Change a parameterization, based upon what you were assigned during class.

   All simulations will be done using Amazon Web Services(AWS) environment.

Summary: The tutorial has all the necessary commands to create and access the WRF container within the cloud. For the assignment, you will need the output plots generated from the tutorial between the times of 20160803 12 UTC to 20160804 12 UTC (plot numbers 12 -> 20) for each of the three different runs you will be completing. A report with analysis of these times will be conducted to subjectively highlight the differences between the runs.

Method:

1. After completing the initial control run transfer the .pdf outputs to the local computer, then remove the container
2. Start a new container pointing to a different set of initialization data "twsee/405gefs". Complete the run of WRF and then NCL scripts, and transfer the .pdf output files to the local computer.
3. Depending on which of the parameterization perturbations you have been assigned, you will again need to reinitialize the Docker WRF container. Within the directory, /wrfinput/ there is a file, "namelist.input", this file controls the parameterizations that are specified **before** running WRF. Change the necessary parameterization under the **&physics** section.

mp_physics = microphysical parameterization

sf_sfclay_physics / bl_pbl_physics = surface layer and and planetary boundary layer parameterization (same number)

cu_physics = cumulus parameterization

After completion of this model run, transfer the output plots to your local computer.

Control Run Parameterization Options			**Parameterization Options**	
mp_physics =	8	Thompson Scheme	6	WRF Single-Moment
sf_sfclay_physics =	2		1	
bl_pbl_physics =	2	Mellor-Yamada-Janjic	1	Yonsei University Scheme
cu_physics =	1	Kain-Fritsch Scheme	93	Grell-Devenyi Scheme

All files and report components should be included in a tar file named "WRF_*yourlastname*.tar.gz".

Model Completion: 20 Points

15 points: Provide the output plots for each model run between 0803 12 UTC and 0804 12 UTC.

5 points: Provide feedback relating to the ease of use regarding cloud computing and Docker. Do you feel that use of these tools has helped increase your understanding of modeling? Explain your answer.

Model Validation: 20 Points

Compare and contrast the observed radar and data to the 3 different model outputs. (1-2 pages) Observed radar and data will be provided via a tarbal through ezlms.

GFS Input Data Vs Observed

GEFS Input Data Vs Observed

Parameterization Vs Observed

Summarize relative performance of all three simulations

Assignment Answers:

While there are no specific correct answers, the students are supposed to realize that by changing one parameterization within the model the entire solution changes. Each field represented within the plots has some sort of difference whether minute or extreme and there should be some mention of every field in a students' write up. The main takeaway from this exercise is the model is interconnected and each parameter influences the entire solution, not just that specific process in time.

Pre/Post - Assessment Questions

Rank your confidence from 1-4. 1 being “Not Confident” and 4 being “Very Confident”

1) Your ability to configure and compile a forecast model on a computer from beginning to end.

Not Confident 1 2 3 4 **Very Confident**

2) Confidence to explain why models aren’t perfect to the general public.

Not Confident 1 2 3 4 **Very Confident**

3) Ability to locate and download necessary model input data.

Not Confident 1 2 3 4 **Very Confident**

4) Confidence to explain what contributes to model error.

Not Confident 1 2 3 4 **Very Confident**

5) Manipulate a forecast models’ configuration to change parameterizations utilized.

Not Confident 1 2 3 4 **Very Confident**

6) Manipulate a forecast model to use different input data.

Not Confident 1 2 3 4 **Very Confident**

7) You have a high level of understanding of instrument error.

Not Confident 1 2 3 4 **Very Confident**

8) You have a high level of understanding of model truncation error.

Not Confident 1 2 3 4 **Very Confident**

9) You have a high level of understanding of model parameterizations.

Not Confident 1 2 3 4 **Very Confident**

10) You have a high level of understanding of a staggered grid and its benefits within models.

Not Confident 1 2 3 4 **Very Confident**

11) You have a high level of understanding of ensemble forecasting.

Not Confident 1 2 3 4 **Very Confident**

12) Confidence to interpret ensemble forecast products.

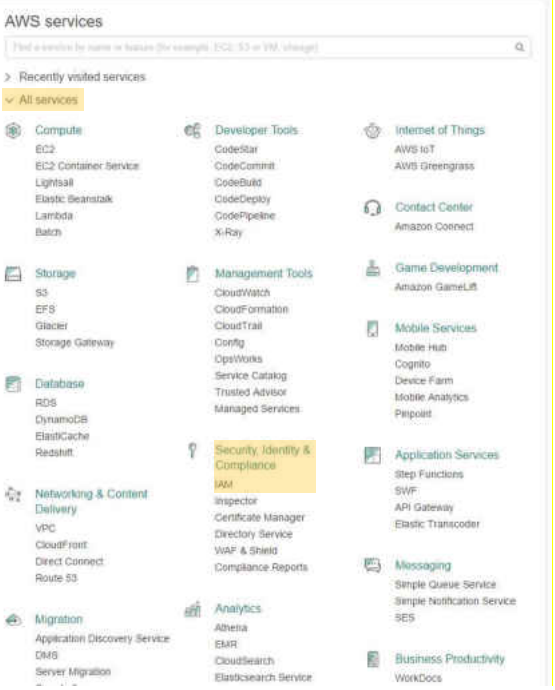
Not Confident 1 2 3 4 **Very Confident**

13) Ability to explain ensemble forecasting benefits.

Not Confident 1 2 3 4 **Very Confident**

Instructions for creating classroom environment for AWS usage

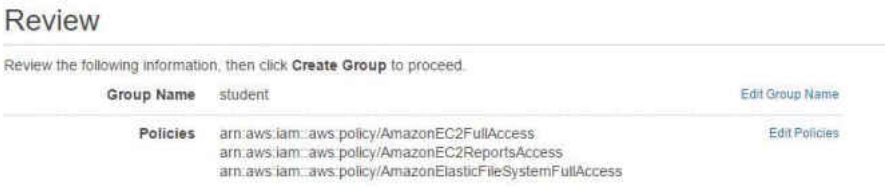
1. Log into your AWS account
2. From the Dashboard show “All Services” shown below
    a. Under the “Security, Identity & Compliance” section, select “IAM” shown below



3. You are now at the Identity and Access Management(IAM) Dashboard
    a. Customization of the classroom URL is possible by selecting customize shown below highlighted in yellow, and changing it to the preferred name. This will change the string of numbers “084685956777” within the URL

4. Creation of group permissions
    a. Under IAM resources shown above, select “Groups: 0”

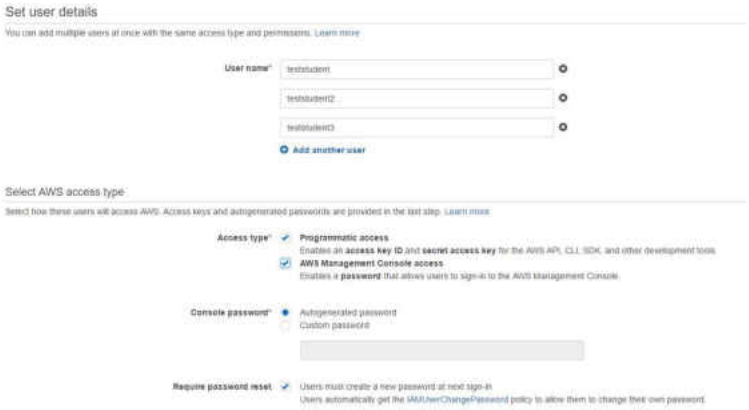
b. Choose the blue “Create New Group” button
    i. Designate the group name to your choosing, likely choice is “student”
    ii. The next page “Attach Policy” allows for you to designate the access each user put within this group is allowed. For the classroom implementation the attached policies were “AmazonEC2FullAccess” “AmazonEC2ReportsAccess” and “AmazoElasticFileSystemFullAccess”. These three policies allowed for students to complete all tasks required within the assignment.
    iii. Screenshot provided below will be what is reviewed prior to creating the group



c. This will take you back to the group portion of the dashboard, by selecting “Dashboard” on the left hand portion of the screen you will return to the initial screen

5. Creation of Users

a. Select “Users: 0”
    i. Then click the blue button in the top left portion of the screen “Add user”
    ii. Below shows an example of 3 test student usernames for creation, make sure that “Access type Programmatic” access and “AWS Management Console access” are both checked
    iii. Password Creation: Select Autogenerated password or Custom password, ideally Autogenerated password would be more secure as Custom password sets that password to each generated user name listed above below. Also require students to choose a new password when logging in for creation of a personal password
b. Click the blue “Next: Permissions” button in the bottom right corner
    i. The next web page will have the group name of permissions created earlier in the instructions if you named it “student” you will see that name, make sure it is check marked
    ii. Click blue button in the bottom right labeled “Next:Review”



iii. Below is the information that should be shown (replacing group name and usernames with what has been inserted)

Review

Review your choices. After you create the users, you can view and download autogenerated passwords and access keys.

User details

User names	teststudent, teststudent2, and teststudent3
AWS access type	Programmatic access and AWS Management Console access
Console password type	Custom
Require password reset	Yes

Permissions summary

The users shown above will be added to the following groups.

Type	Name
Group	student
Managed policy	IAMUserChangePassword

iv. Click the blue button labeled “Create users” in the bottom right hand corner of the web page

v. This next page will show you the created users with their Access key IDs, Secret Access keys, Passwords, and a link to e-mail login instructions

6. Below shows the web page regarding the Users and their passwords
   a. It is recommended to download the .csv file to keep the passwords and usernames handy for student reference should they misplace it

b. On the right hand side there is a “Email login instructions” column, this allows for AWS to send the username and URL to the students e-mail address. Password is not included in this e-mail, so the password per student must be sent manually from the professor to the students e-mail accounts.

7. Students can now access the classroom AWS instance through the URL provided with their username and password generated from this document.

APPENDIX B

CONTAINS INTERVIEW TRANSCRIPTS

APPENDIX B

INTERVIEWS CONDUCTED FOR ECONOMIC, POLICY, AND PRIVACY CONSIDERATIONS

John Wold - Scientific Computing Center University of North Dakota 2 pages

Aaron Bergstrom - Computational Research Center, High Performance Computing Specialist 2 pages

Darin King - North Dakota University Systems, Chief Information Officer 2 pages

Derek Stinchfield - Scientific Computing Center Systems Administrator and Andrew Keulbs - Lead Client Support 2 pages

John Wold - Scientific Computing Center University of North Dakota

Tim: We want every university to be able to take what I did and do it within their university setting

John: So, when you do that in a perfect world without Amazon use we can do it here then ship it out, were not worried about security with what you created. you would want a saved environment of some kind.

Tim: with the push to centralization how would this be possible?

John: Centralization has to do with more services such as e-mail rather than anything else and commodity type services not so much scientific computing. the vast majority of hardware in there is most likely exempt from the centralization.

Tim: most likely the labs we have will not be touched at all?

John: exactly they should stay the same.

Tim: What about some sort of community college or a university with a meteorology department and does not have the IT infrastructure and support that UND does.

John: AWS would make sense for something like that, but who would pay for it?

Tim: do you see the university pushing for cloud use at all?

John: they use a bunch of cloud computing services already; some e-mail is cloud. they are willing to put major services in the cloud.

Tim: is cloud computing services for SCC going to be an option going forward?

John: we find the best possible solution and most affordable solution is also important. we are open to cloud based solutions. especially if we have people outside NDUS accessing our network that. use of cloud computing isolates us and offloads the security to their network. if allows use of certain applications and don't' have security measure that computer can become a virus bot or denial of service bot and go after other machines in that network if they are not secure. if you can log in to use Docker you can do whatever you want with it. scientific discovery requires openness and the bad actors that destroy innovation and scientific discovery because you can't keep it as open as It needs to be. The next best thing is to dump it onto the cloud outside your environment and whatever happens is on their end. they deal with the consequences we do not. They have essentially what we have here but on a grander scale. They know how to isolate assets within their physical data center. the biggest issue for you is to find the funding and continuing funding for this.

Tim: We were thinking about tacking on some sort of fee to the class to help pay for the usage of the cloud computing instance.

John: or you can look into a group of universities that would pay into having it stored somewhere and they then have access to the data. but you will still need some of the expertise to manage the dataset and ensure that it is working. The best way to isolate something is to do it physically, purchasing hardware spending money upfront to have access to it, but that can cost a lot of

money. It seems like the best solution for what you are doing is a cloud based services, especially for universities with lower IT infrastructure.

Tim: We want to either be able to get Docker installed on all lab machine or use it in a different way so that students can access it without harming the local network

John: so being a part of computer science for 15 years I know what undergrads can do to an environment. So, that brings us right back to AWS to create a secure environment, instead of creating our own secure environment which causes a lot of money.

Tim: So I feel that's best solution overall

John: it would be the quickest way to get things going

Tim: I don't think it would cost too much to utilize also

John: if you want to run it all year long that would be much more costly The use is aimed at a specific assignment as a teaching tool as a week or two?

Tim: yes

John: I don't see any reason to not use amazon web services then. there are plenty of providers out there to utilize, you really don't' need much of IT to get involved

Tim: there are some things IT help would be ideal, the setup of the instances allowed for all students to access one another's instances meaning stopping or terminating another instance that is not theirs

John: So, you would need some consulting from us or have us set up permissions, and it seems like it is an on-demand computing service that you use and then once you're done you terminate it and you no longer have access. it is hard to beat that sort of flexibility for something like this. If you're going to use it all semester if might be better to consider local hardware use, if it is one little chunk it seems like you are on the right path then. and just use us as technical consulting if you hit any issues. As far as other universities you just use communication, provide the steps by which you did everything to someone else and they can set it up themselves whether its through GitHub or something else. So if you throw something up in the cloud, unless you want us to open a firewall here is basically your business and there wouldn't be any issues. due to it being a non-mission critical application so it is perfect for the cloud.

Aaron Bergstrom - Computational Research Center, High Performance Computing Specialist

Aaron: (Speaking regarding john Wolds suggestion of using it in the cloud as best practice) He might actually be right the easiest best case scenario for small schools like that would be to deploy it within the class. the problem with that is, as long as you have money to pay for stuff like that its fine, but if you end up having a lot of computing in there it may impact the budget. With a lot of container software out there, such as Docker there is a push to solve some of the security concerns regarding Docker and container software. UND is part of the Big Data Hub which has a lot of resources that are available that could be free for use that could maybe provide some services. The national data service will spin up individual environments for universities to use which is a cloud environment which is run by the national center for supercomputing applications. so that is one option that could be considered due to the lower cost. As for locally we are transitions to a more of a cloud computing environment, but that is a ways in the future. but we would be able to designate the computing to general computing or hpc applications and cloud computing services to professors and students. but just because that could be available, doesn't mean that it is easiest for the project itself. it just may be the easiest for them is to spin up a cloud environment and then run that for a semester or whatever it is needed for. and there are a lot of different options out there to deal with security issues.

Tim: So, what we did for the course of the two weeks in the class cost 40 bucks. that is with many instances up and running without doing any actual computing causing inflated costs.

Aaron: So, that's a cheap reasonable way of doing it and I'd say almost any place can afford it.

Tim: and you can tack on a technological fee for the students to help pay for it.

Aaron: that 40 dollars for everyone in the class? that is obviously a very low cost.

Tim: the only issues we ran into was setting up permissions for the students accounts, allowing for much more access than needed for the assignment. Students could access one another's instances.

Aaron: there is a level of they all want to pass the class, they all want to learn something, why would they go to that? even if they did, if you start from a saved instance, you can go boot it back up. sure someone may have to start over but there is a reasonable level of security because the people in the class are not hacker generally speaking.

Tim: looking forward with consolidation going through, do you know of anything that could happen to the labs we used within the class?

Aaron: computer science is potentially going to be moved to engineering, but as far as I understand they are staying in place indefinitely as far as physical location. how it gets worked out with who would support those systems is not known currently but those systems will stay put.

Tim: there wouldn't' be any push to windows form the current Linux environment?

Aaron: there is no current Linux support within central IT, they know it is something that would be needed and they would hire someone to manage that. they wouldn't' force people to change to windows when computer science has been taught in Linux and always has been.

Tim: lets say the university system moves towards a more cloud based system, how would educators get use of the cloud for something like this(Docker use)

Aaron: if you want to use what the campus purchases for computing then you can use that for free, but if we set up a 128node cloud system, the idea would be that a committee meets 3 times a year and designates the cloud usage over the year. If you wanted to host a class in the fall you would apply for some of the compute environments and the committee would meet and decide who gets what and it would be free. there would be a level of review to get the free services. But purchasing of a node to connect to the environment could be done and then whomever purchased that hardware would have access and dictation over it.

Tim: is there any push against cloud computing within the classroom?

Aaron: honestly, I don't think there is any pushback of something like cloud computing, it comes back to who pays for it. and if it's 40 bucks I don't' think there is much issues along something that low cost. if you have an image already defined in amazon, use this instance and image you created and use that for student works. The thing with classroom use you won't see large data sets for something like what you did, the issue is that maybe a student down the line has a faulty view of data usage and fees within the cloud from using this.

Darin King - North Dakota University Systems, Chief Information Officer

Darin: (Regarding the centralization changes with the upcoming policy reformation, directed at the labs we used) Form the lab perspective the consolidation has nothing to do with them. Labs are labs and we draw the line there because that doesn't lend itself to centralization. The labs must be maintained by local campuses. Trying to manage something from afar does not work. The advantages of AWS and Google Azure is that you can spin up and spin down at will, I don't have any issues with things being in the cloud. Your risks change there is risks in both and my job is built around reducing risk. It's really not anything about the cloud it's a different set of thoughts

Tim: Do you see any security concerns regarding a professor spinning up an AWS cloud instance for use in the classroom?

Darin: Depends on what they are using it for?

Tim: We used it to place Docker on an AWS instance to reduce the security concerns by offloading them onto the cloud platform.

Darin: There is one of the inherent advantages, the ability to spin up and spin down. Microsoft for some higher education has waived the egress fees, so that may be something to consider. Your example of spinning up those AWS instances the data is not an issue this is a perfect use case for this. Some of the security concerns go away, you don't have some of the data feeds intertwined within the network which makes it more secure.

Tim: A lot of it comes down to who pays for it, I know that AWS offers partnering with a university and allot you educational credits per year. Use of that or a technological fee charged to the students would be a good solution to the cost of the compute time.

Darin: What you are talking about is very course specific, it is almost like taking pottery you have to pay for the clay. So making it very class specific costs definitely could be a good solution to payment issues.

Tim: So you don't think there is anything bad about a professor spinning up an instance on campus?

Darin: Well we can't control it, we know it happens. I would view it as a campus level decision, rather than consolidation. Some of the work we do only here only lasts a couple months of the year and we are looking towards using the cloud for that work, due to the small amount of time we use the hardware that otherwise just sits there. Cloud enables us to burst our needs for a shortened period of time and utilize the computing power just when we need it.

Tim: Another reason that we are doing this is that some lab computers may be windows based, it is an advantage to having it in Docker which is reproducible.

Darin: To me that's one of the inherent advantages of being in a container type world as opposed to VM type world. I really think the container concept is maybe the next evolution. The container type methodology makes sense. In a VM world you can take a snapshot and its portable, but you replicate problems that way. In a container, you keep it so tightly wound that

it has less issues. A single advantage for smaller advantages, having a stable technology spend rather than a big spend then nothing for years. This can provide stability to pricing with good management.

Derek Stinchfield - Scientific Computing Center, Systems Admininstrator

Andrew Keulbs - Scientific Computing Center, Lead Client Support

Andrew: (Regarding the security issues regarding Docker in the lab environment) That is a statewide policy, you can't affect the work of other people, like your computer even as a private workspace in your office, is affecting other people's resources you are now impacting their ability to do their work.

Derek: it's not to say we can't use the software and put them places, a lot of times we need to isolate them.

Tim: so, the main security concern is that, that computer can't hinder someone elses computer around it.

Andrew: you can't impact another person's ability to access resources and do their job

Tim: what about Docker is the issue?

Andrew: Docker is just like vm ware, it is treated as if it's a virtual computer.

Derek: you can get root to that system even if you don't have root to the main system.

Andrew: let's say Docker was in the computer lab, you don't have root to that machine, Docker does. so if you put Docker container in there and you have root passwords to that container, you are container on that machine and now you are root in that network. so usually you don't have root access within the computer lab.

Derek: the user that runs the Docker software has elevated and almost root privileges and can affect the system.

Tim: so it is tricky to achieve but possible?

Andrew: the thing with Docker is that containers that would be run within Docker they would have to be vetted to ensure no malicious containers. it is like GitHub you can just tell Docker to download a container no matter what it is. allowing for malicious scripts to possibly accompany it. and it may have a script to act as a bot or spam bot, or act as a denial of service bot.

Derek: depending on what people script, you have root access to that container. it is potentially possible that you can take over the entire machine through the Docker machine if they know how to manipulate privileges

Andrew: a Docker user can elevate another fake user that within the host computer with root access.

Derek: there is enough ways within the container to have it create malicious material within the container so that it can do enough harm and start denial of servicing a computer or probe it to find weaknesses.

Andrew: you can do a scan of the network from the container to see what ip addresses are live and then do a port scan on those live ip addresses and see what ports are open and with the

right code you can find vulnerable spots or denial of services by making requests and never answering them. a generic Docker install can download any hosted Docker container any of those can be run and that's a security issue. it can run malicious and normal ones all the same. they have full rights to download any Docker container on the planet. there are plenty of people that spoof repositories with malicious software. so, that's where security concerns of Docker, how do you know what is legitimate? you can disable the rights to download containers, and you should hold them locally on the machine. if an instructor knew they were going to run these 3 containers we can distribute them onto a lab and just disable the ability to download containers. we would have the authorized containers locally for access.

Derek: even with Docker and things like that, it's not that we don't' want to do it. we just want to mitigate as many of those concerns as possible before we do it.

Andrew: Docker would totally be a valid requested software, just knowing those security concerns would need to be dealt with its just a matter of how do we do it. limiting the containers. we can allow Docker to only access certain subset of containers, or we can allow certain users to install whatever they need. you can't mix match the permissions its one or the other.

Derek: it is essentially putting more than your own data at stake anyone who has got a home directory in the lab could potentially lose their data. professors would not like that if someone wiped all the home directories.

Andrew: with multiple user environments, it is just key that you don't' hinder anyone's ability to access any resource on the computer on any day at any time. which is why we need to know which containers are being used and we know they are not malicious.

Tim: as far as new software requests, what is the process by which you consider it

Andrew: we get the official version of the software that is requested. and for some software some more in-depth security checked within the environment. we may spin up a new isolated system to test the new software in the environment and make sure it is ok.

Tim: how long does it normally take for installation of a software

Andrew: it can be the same day or depending on the accessibility of the software and any complications it may take several days.

Tim: you guys think cloud computing is secure enough to use it with lab environments?

Andrew: connecting to an external service is usually never a problem if what you are doing is handled by the external service. accessing a remote service is never usually a problem. it is not in house it is not here being affected. any virtual computer that is being run somewhere else, that means whatever you are doing is on that computer and not here. if it is a legitimate service. a professor using containers on an amazon hosted instance should never affect the local network or resources. just ensure that password management is held, don't' share the username and passwords with the class. just abide by FERPA. you can't share names and information, if they are personally doing it themselves then its fine but for a professor to request that information within the instance is not allow.