



Antenna Toolbox™ Parallelization on Amazon EC2 Cloud Cluster

BY:

VISHAL K. RATHI

Submitted: March 2017

Approved by:

PROFESSOR SERGEY MAKAROV

A Major Qualifying Project Report submitted to the faculty of WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Computer Engineering.

Abstract

The goal of this project was to run Antenna Toolbox™ solutions on Amazon EC2 cloud cluster and benchmark the performance on both local and cloud clusters. The first section will cover some background knowledge about the products used in this project. After that, the report talks about the methodology to create a cloud cluster and use it. Next, the report shows some performance results obtained for the cloud cluster running on Amazon EC2. Finally, the report ends with a conclusion section which talks about the things learned in this project.

Acknowledgements

I would like to thank the following people and institutions:

The RF and Antenna Toolbox™ team members, Dr. Vishwanath Iyer, Dr. Shashank Kulkarni, Dr. Da Huang, Dr. Don Orofino, and Dr. Mark Reichelt at The MathWorks, Inc. They guided me throughout this project and made this project a success.

I would like to thank my advisor for this project, Professor Sergey Makarov. He provided me with much guidance and advice, and my project would not have been what it is without him.

Finally, I would like to thank Worcester Polytechnic Institute and The MathWorks, Inc. for making this MQP project possible.

Table of Contents

Abstract.....	I
Acknowledgements.....	II
List of Figures	V
1. Introduction	1
1.1 Project Description.....	1
2. Background	2
2.1 Cloud Computing	2
2.2 Amazon Elastic Compute Cloud (EC2).....	2
2.3 Parallel Computing	3
2.4 Antenna Toolbox™	3
2.5 Parallel Computing Toolbox™	4
2.6 Using parfor loops to parallelize Antenna Toolbox™ solutions	4
3. Methodology.....	6
3.1 Step by step guide for setting up Amazon EC2 cloud cluster	6
3.2 Local and Cloud clusters details	22
3.2.1 Local cluster	22
3.2.2 Amazon EC2 cloud cluster.....	23
4. Results	24
4.1 LU Factorization Benchmarking	24
4.2 Use Cases	28
4.2.1 Case 1: Dipole Array	28
4.2.2 Case 2: Spiral antenna.....	30
4.2.3 Case 3: Spiral array.....	33
4.2.4 Case 4: Patch antenna on a substrate.....	35
4.3 Running batch jobs on Amazon EC2 cloud cluster	38
4.4 AWS memory optimized instance.....	39
4.4.1 RadarBookColumn Problem.....	40
5. Conclusion.....	42
References	43

Appendix	44
LU Scaling code	44
Linear dipole array	45
Spiral antenna	46
Spiral array	46
Patch antenna	47
Script with the parfor loop.....	48
RadarBookColumn	48

List of Figures

Figure 1 Adding users to the MDCS license on the MathWorks license manager	6
Figure 2 Sign up screen for Amazon Web Services	7
Figure 3 Register screen for Amazon Web Services	8
Figure 4 MathWorks cloud center login window	9
Figure 5 Home screen of MathWorks cloud center	9
Figure 6 User preferences screen of the cloud center	10
Figure 7 Home screen of AWS console/dashboard	11
Figure 8 AWS services	11
Figure 9 Identity and Access Management dashboard	12
Figure 10 Creating a new role	12
Figure 11 Specifying the role name	13
Figure 12 Selecting role type	13
Figure 13 Adding MathWorks cloud center account ID and External ID to the new role	14
Figure 14 Attaching policy to the role	14
Figure 15 Review screen of the new role created	15
Figure 16 Machine types available on the MathWorks cloud center	17
Figure 17 Create new key on MathWorks cloud center	17
Figure 18 Creating an Amazon EC2 cloud cluster on MathWorks cloud center	19
Figure 19 Typical cluster on the cloud center	20
Figure 20 View the available clusters on cloud center	21
Figure 21 Cluster profile on the MATLAB® client	22
Figure 22 Amazon EC2 cloud cluster details	23
Figure 23 Amazon EC2 compute optimized (C3) instances details	23
Figure 24 LU Factorization speed up plot for cloud cluster (5000 matrix size)	24
Figure 25 LU Factorization speed up plot for local cluster (5000 matrix size)	25
Figure 26 LU Factorization speed up plot for cloud cluster (9552 matrix size)	26
Figure 27 LU Factorization speed up plot for local cluster (9552 matrix size)	27
Figure 28 Dipole Linear Array object details	28
Figure 29 Speed up plot for frequency sweep on a linear dipole array	29
Figure 30 Time taken in minutes for frequency sweep of linear dipole array	30
Figure 31 Spiral antenna object details	31
Figure 32 Speed up plot for spiral antenna	31
Figure 33 Time taken in minutes for frequency sweep on a spiral antenna	32
Figure 34 Spiral antenna 2x2 array object details	33
Figure 35 Speed up plot for spiral antenna 2x2 array	34
Figure 36 Time taken in minutes for frequency sweep on a spiral antenna 2x2 array	35
Figure 37 Patch antenna object details	36
Figure 38 Speed up plot for Patch antenna	36
Figure 39 Time taken in minutes for frequency sweep on a patch antenna	37
Figure 40 AWS memory optimized instance details on cloud center	39
Figure 41 EC2 memory optimized (R3) instance details	39
Figure 42 Patch antenna image	40
Figure 43 Radar antenna object details	40

Figure 44 Impedance plot for the radar antenna 41

1. Introduction

Antenna Toolbox™ users run both time and memory intensive simulations in the process of antenna design. Some of the computations take days and weeks to complete. In the current world which is so fast, the importance of time is much greater than money. To solve this problem, we investigated parallelization of Antenna Toolbox™ solutions on both local multicore computers and cloud clusters. The cloud cluster was created on Amazon Web Services (AWS). In addition to Antenna Toolbox™, Parallel Computing Toolbox™ and MATLAB® Distributed Computing Server™ were also used in this project. The goal of this project was to benchmark and compare the performance of Antenna Toolbox™ parallelization on both the local machine and cloud cluster. The comparison is made between the performance on local machine and cloud cluster on Amazon EC2. The methodology and results are discussed in this paper.

1.1 Project Description

The goal of this project will be to run Antenna Toolbox™ solutions on Amazon EC2 cloud cluster and benchmark the performance on both local and cloud clusters. The first step to achieving this goal will be to create a remote cluster of multiple workers on Amazon EC2. After the cluster is up and running, the next step will be to study the performance of LU factorization on both local and cloud clusters. LU matrix factorization is done to solve the antenna structure and is the most time and memory intensive step in the process. Therefore, the overall performance of parallelization of antenna solution will depend on the performance of LU factorization of the corresponding matrix size. Once that is done, multiple use cases will be run on the Amazon EC2 cloud cluster to measure both the speed up and time taken to solve different frequency sweeps on different antenna objects. This report will start off with the background knowledge first and then will go into details of creating the remote cluster and establishing the performance of different problems.

2. Background

The purpose of this section is to provide the reader with a general overview of cloud computing as well as an introduction to Amazon Elastic Compute Cloud and its services. This section also gives readers an introduction to parallel computing, Antenna Toolbox™ and Parallel Computing Toolbox™.

2.1 Cloud Computing

The idea of cloud computing, a virtual form of network-based processing, was first introduced in the 1960s. Now, it has become a public utility. The idea was to allow users to purchase and use virtual resources so that they don't have to invest heavily in a physical computing infrastructure.

Cloud computing is essentially made up of a set of data centers located at different geographic locations that provide users with scalable remote computational resources. These data centers can be either private or public. Private data centers are commonly used by organizations or corporations for private use. Public data centers are available to the general public for personal use. The biggest advantage of cloud computing is the on-demand nature of these services. It allows customers to get immediate acquisition of any amount of computing power, without requiring any prior commitment to the resources.

2.2 Amazon Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. [9]

Amazon EC2's simple web service interface allows users to obtain and configure capacity with minimal friction. It provides users with complete control of your computing resources and lets users run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing users to quickly scale capacity, both up and down, as their computing requirements change. Amazon EC2 changes the economics of computing by allowing users to pay only for capacity that users actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give users the flexibility to choose the appropriate mix of resources for their applications. Each instance type includes one or more instance sizes, allowing users to scale their resources to the requirements of their target workload. [3]

2.3 Parallel Computing

Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time. Parallel computing is an opposite approach to serial computing where an algorithm is constructed implemented as a serial stream of instructions which are executed on one computer. On the other hand, in parallel computing, multiple processing elements are used to simultaneously solve a problem. The processing elements can be diverse and include resources such as a single computer with multiple processors, several networked computers, specialized hardware, or any combination of the above. [6]

Ideally, the speedup for parallelization should be linear meaning that doubling the number of processing elements should halve the runtime, and doubling it a second time should again halve the runtime. Since, we don't live in an ideal world, achieving linear speedup isn't possible. Most of the algorithms have close to linear speedup for smaller number of processors and then the speedup saturated as the number of processors become large.

2.4 Antenna Toolbox™

Antenna Toolbox™ provides functions for the design, analysis, and visualization of antenna elements and arrays. Users can design standalone antennas and build linear, rectangular, and conformal arrays of antennas using predefined elements with parameterized geometry or custom elements.

Antenna Toolbox™ uses the method of moments (MoM) to compute port properties such as impedance, surface properties such as current and charge distribution, and field properties such as the near-field and far-field radiation pattern. Users can visualize antenna geometry and analysis results in 2D and 3D. The

goal is to parallelize Antenna Toolbox™ functions since the antenna design process can both be time and memory intensive. The details and results of the parallelization is explained in this paper. [43]

2.5 Parallel Computing Toolbox™

Parallel Computing Toolbox™ lets users solve computationally and data-intensive problems using multicore processors, GPUs, and computer clusters. The toolbox lets users use the full processing power of multicore desktops by executing applications on workers (MATLAB® computational engines) that run locally. Without changing the code, users run the same applications on a computer cluster or a grid computing service. With the help of this toolbox and MATLAB® Distributed Computing Server™, the Antenna Toolbox™ functions are run in parallel on both local and cloud clusters in this project. [6]

2.6 Using parfor loops to parallelize Antenna Toolbox™ solutions

Parallel Computing Toolbox™ allows users to run their loops in parallel using a parfor loop. When a parfor loop is used, the MATLAB® client coordinates with the workers comprising a parallel pool to execute iterations in parallel on the pool. The required data is sent to the workers from the client and once the task is completed, the results are sent back to the client from the workers. Since the loop iterations are run simultaneously, parfor loop can provide significant improvement in the performance.

Each execution of the body of the parfor loop is an iteration. There is no particular order in which workers evaluate each iteration. Each execution of the parfor loop is independent of each other. This is a very important property of the parfor loop because it doesn't require workers to communicate with each other for the data since each iteration is independent of each other. Parfor loops cannot be used when an iteration in the loop depends on the results of other iterations. [8]

The independence of each iteration makes it ideal to parallelize Antenna Toolbox™ solutions. Typically, while designing the antenna, users run a frequency sweep consisting of several data points or some other kind of similar sweep to analyze the performance. Each data point in the frequency sweep is independent of each other and therefore, each loop iteration doesn't depend on the results from other iterations. This creates an ideal situation for users to parallelize their solutions to save on the computation times which can be quite significant. This project capitalizes off of this to explore the possibility of parallelizing Antenna

Toolbox™ solutions on both local and cloud clusters with the aim of reducing the computation times significantly.

3. Methodology

This chapter discusses the methodology used to create the Amazon EC2 cloud cluster. It also gives details of the clusters. [7] The next chapter discusses the results obtained.

3.1 Step by step guide for setting up Amazon EC2 cloud cluster

This guide lists all the steps to set up Amazon EC2 cloud cluster on MathWorks cloud center. The user needs both MathWorks cloud center login and AWS login in order to successfully set up the cluster. The set up process is one time. After the cluster is set up, Antenna Toolbox™ code can be run on that cluster in parallel from the client.

1. The first step is to have valid MATLAB® Distributed Computing Server™, Parallel Computing Toolbox™, and Antenna Toolbox™ licenses. MATLAB® Distributed Computing Server™ is needed to run computationally intensive MATLAB® programs on computer clouds such as Amazon EC2. Users develop their programs on multicore desktop computers using the Parallel Computing Toolbox™ and scale up to many computers by running it on cloud clusters using MATLAB® Distributed Computing Server™. The server includes a built in cluster job scheduler for Amazon EC2 which is why it is needed to parallelize Antenna Toolbox™ solutions on Amazon EC2.
2. The next step is to add a user to the MDCS license on the MathWorks license manager page. This step is important before the user can successfully use the license. Here is a screenshot:

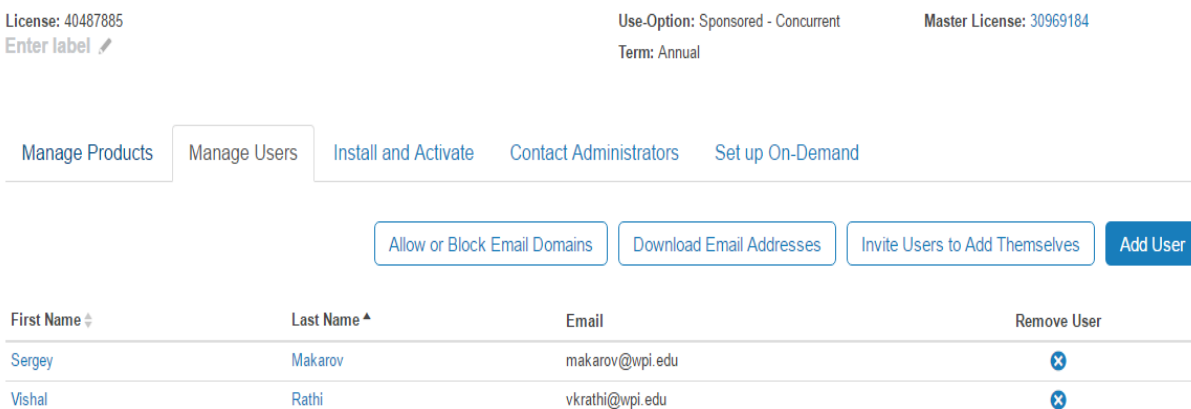


Figure 1 Adding users to the MDCS license on the MathWorks license manager

3. After adding a user, the MDCS license has to be added to the MATLAB® Hosted License Manager in order to login to the MathWorks cloud center where the user can set up the cluster. Without this, the user won't be able to log in to the MathWorks cloud center. To do this, every MDCS license holder has to contact the installation team / technical support at MathWorks and they should be able to do it quickly after verifying some license information. This step is important in order to be able to login to the MathWorks cloud center to begin the cluster formation process. Login to MathWorks cloud center to verify this change. You should be able to successfully login once your license has successfully been added to the MATLAB® Hosted License Manager by the technical support team at MathWorks.
4. Once you are able to login to the MathWorks cloud center, let's set up an account on AWS so that you can connect them in order to be able to successfully run MATLAB® code on Amazon EC2. In order to create a cloud cluster on AWS, an account with AWS is needed. If you have an Amazon.com account already then you can use the same credentials for AWS. Therefore, visit <https://aws.amazon.com/> to sign up. Here is the screenshot of the sign up screen:



Figure 2 Sign up screen for Amazon Web Services

If you don't have an Amazon account already then choose the new user option. Put your email address and click on the sign in button. The next screen is shown below:



Login Credentials

Use the form below to create login credentials that can be used for AWS as well as Amazon.com.

My name is:

My e-mail address is:

Type it again:

note: this is the e-mail address that we will use to contact you about your account

Enter a new password:

Type it again:

Figure 3 Register screen for Amazon Web Services

Once all the information is entered, you will have an account with AWS. You can then login to verify if your credentials work or not. Once you are logged in, you will be asked to enter your billing address and credit card information on your first login. You will only be charged once you start using the services. It might take up to 24 hours for amazon to fully activate the account. Therefore, it is recommended for you to wait for up to 24 hours until you get the confirmation email from amazon. The confirmation email would say that you can now successfully use the services and launch instances from your AWS account.

5. The next step is to access the MathWorks cloud center to create the Amazon EC2 cloud cluster and connect MathWorks cloud with your AWS account. To access your MathWorks® Cloud Center account, navigate in a Web browser to the website: <https://cloudcenter.mathworks.com/login>.

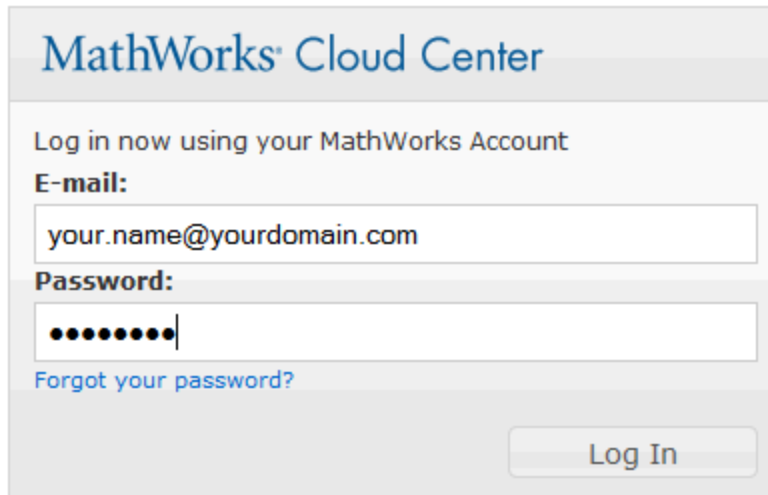
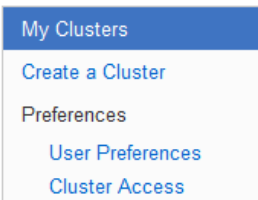


Figure 4 MathWorks cloud center login window

6. Login with your MathWorks account and password to login to the cloud center.

Cloud Center Navigation

When you are logged in to the Cloud Center, the left-hand navigation pane determines what view and actions are available:



- My Clusters — Lists your clusters and provides access to cluster details.
- Create a Cluster — Allows you to create and start a new cluster.
- User Preferences — Allows you to modify your Cloud Center account and access information, including your AWS credentials.
- Cluster Access — Allows you to control which machines can access your cluster by IP address range.

Figure 5 Home screen of MathWorks cloud center

7. The next step is to set up your Amazon Web Services (AWS) credentials before you can create a cluster. Click User Preferences in the navigation pane. Then follow the instructions on the dialog box which shows up.

- My Clusters
- Create a Cluster
- Preferences
- User Preferences**
- Cluster Access

Add Amazon Web Services Credentials

i For detailed steps refer to "AWS Identity and Access Management (IAM)" section of [Cloud Center User's Guide](#).

Step 1: Sign in to the **Amazon Web Services (AWS)** Console and click on **Identity & Access Management (IAM)** services.

Step 2: In the navigation pane of the console, click **Roles** and then click **Create New Role**. For Role name, type a role name to help identify the purpose of this role. For example: CloudCenterApp.

Step 3: Expand "**Role for Cross-Account Access**" and select "**Allows IAM users from a 3rd party AWS account to access this account**".

Step 4: Use the following information when asked to enter 3rd party AWS account ID and external ID.

Account ID

*External ID

Require MFA (Checkbox remains unselected)

Step 5: On Attach Policy screen, select the managed policy named "**AdministratorAccess**".

Step 6: Copy and enter the **Role ARN** from the review screen below.

*Role ARN

Step 7: On review page click "**Create Role**" to save your role.

Step 8: Add description (optional).

Description

*Indicates required information

Figure 6 User preferences screen of the cloud center

8. Enter data for the following fields from your AWS account.
 - a. External ID – A unique ID that Cloud Center uses when requesting access to your AWS account.
 - b. Role ARN – The Amazon Resource Name (ARN) that uniquely identifies the IAM Role which defines the set of permissions that you are granting Cloud Center for access to actions and resources in your AWS account.
 - c. Description — you can enter any text here for a description of your account or credentials.

To get the above data, follow the steps shown below.

Sign in to the AWS Console with your credentials. The home screen is shown below:

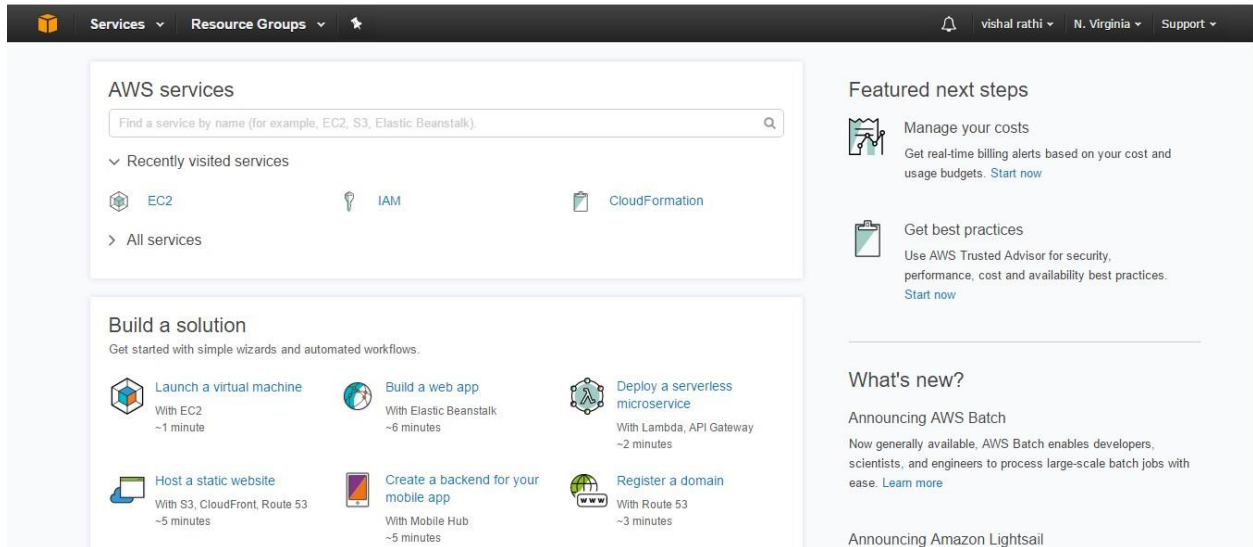


Figure 7 Home screen of AWS console/dashboard

Click on the Services on top left of the home screen which will bring you to the following screen:

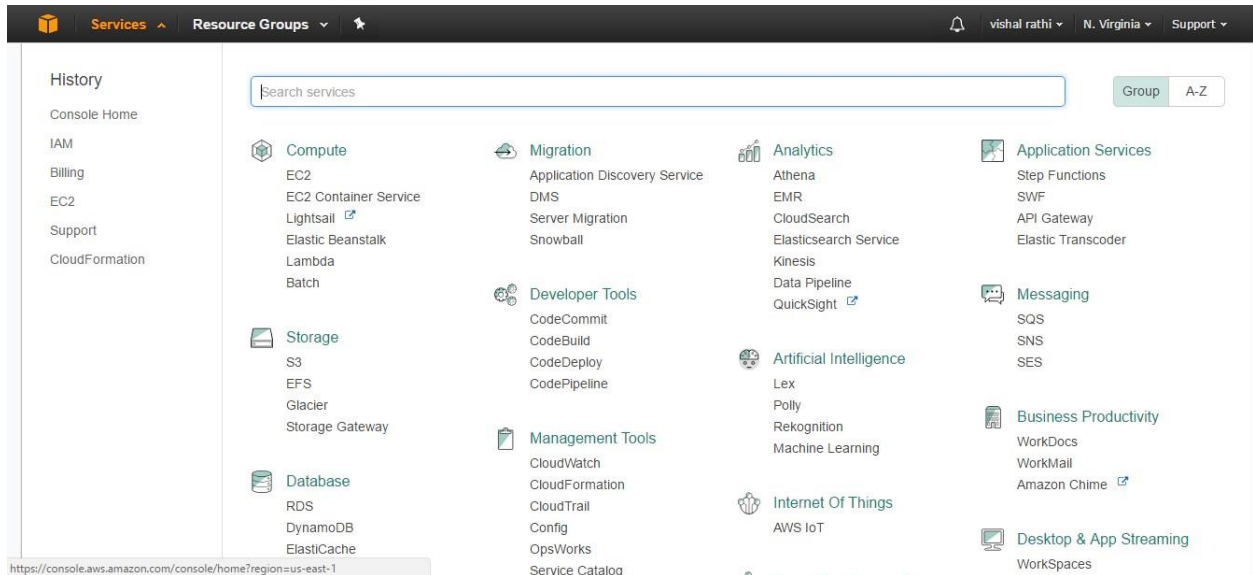


Figure 8 AWS services

The next step is to click on **IAM** option shown in the navigation bar on the left. This brings the following screen:

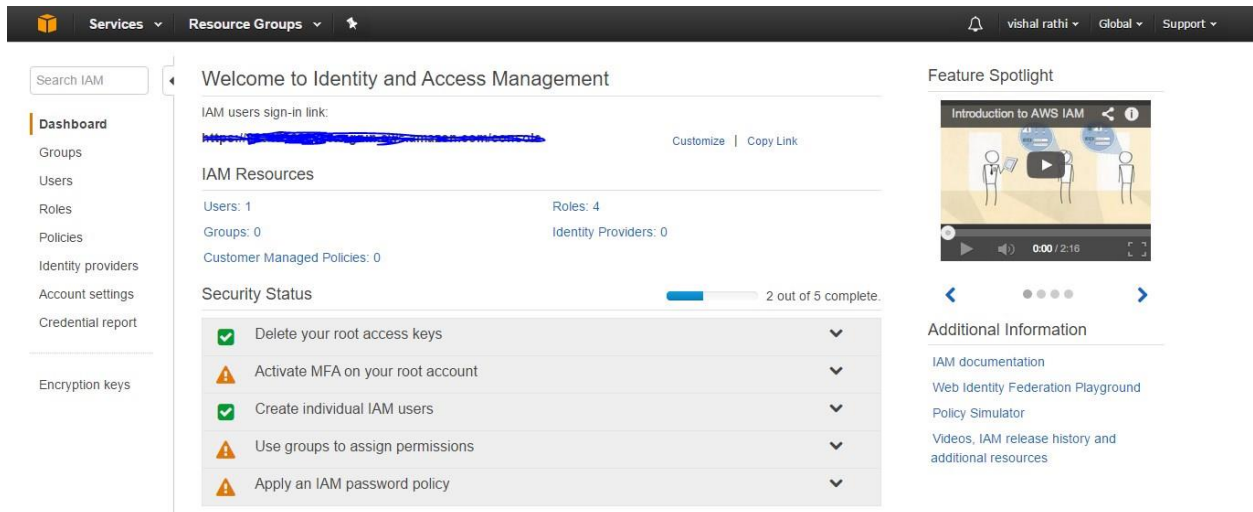


Figure 9 Identity and Access Management dashboard

Click on the **Roles** option shown on the left in the navigation bar.

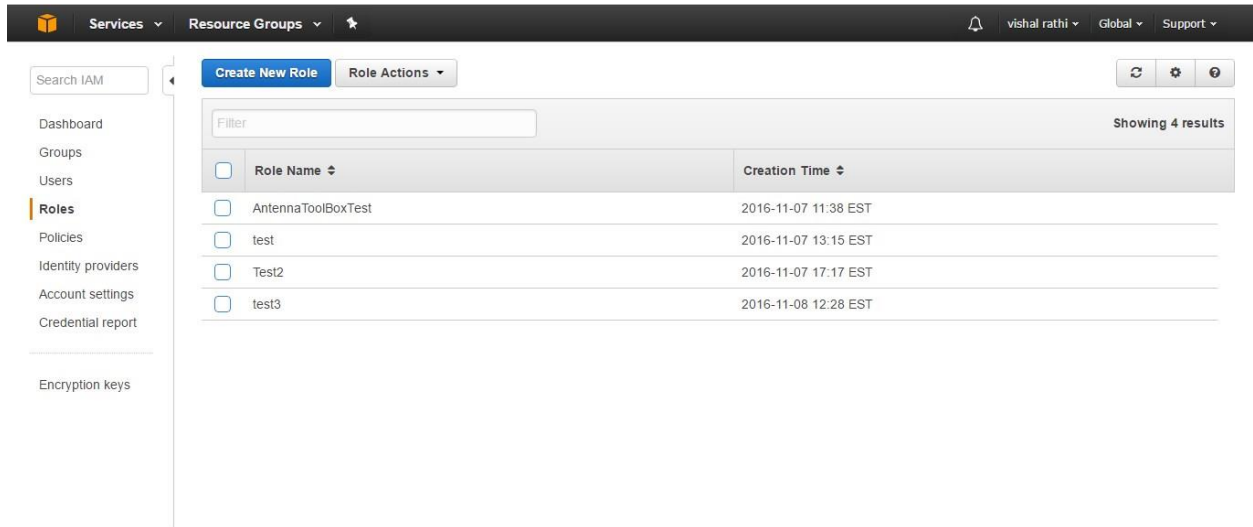


Figure 10 Creating a new role

The next step is to create a new role. Therefore, click on **Create New Role** button on the top. This will bring you to the following screen:

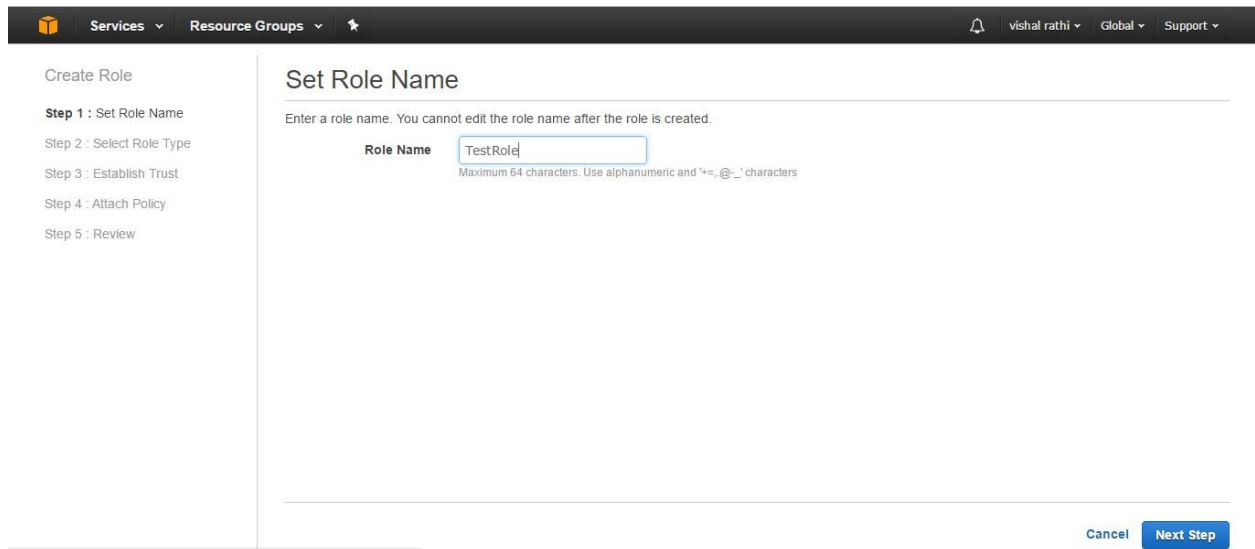


Figure 11 Specifying the role name

Give a name to your role and click on the next step.

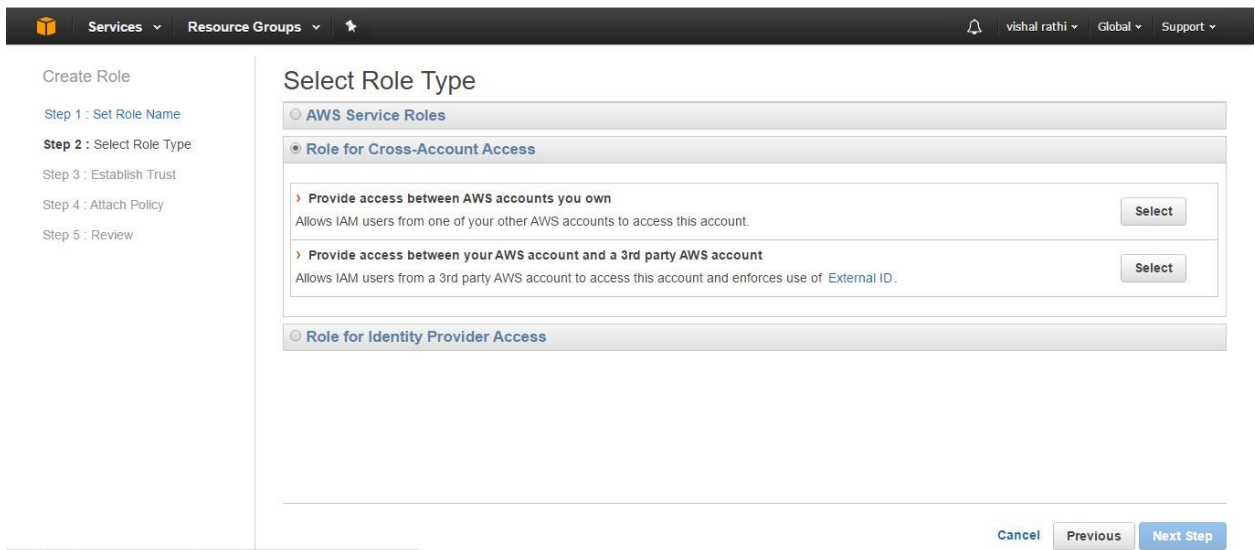


Figure 12 Selecting role type

Expand **Role for Cross-Account Access** option as shown above and select **Allow IAM users from a 3rd party AWS account to access the account and enforces use of External ID**. Once this is done, the next step screen shows up.

Enter the ID of the 3rd party AWS account whose IAM users will be able to access this account. Enter the external ID provided by the 3rd party. For details, see [About the External ID](#).

Account ID:

External ID:

Require MFA:

Cancel Previous Next Step

Figure 13 Adding MathWorks cloud center account ID and External ID to the new role

Use information from figure 13 to provide your **Account ID** and **External ID** from the MathWorks cloud center to the new role that you are creating. The **Require MFA** checkbox remains unselected. After the data is entered, click on the next step button.

Select one or more policies to attach. Each role can have up to 10 policies attached.

Filter: Policy Type Showing 252 results

	Policy Name	Attached Entities	Creation Time	Edited Time
<input checked="" type="checkbox"/>	AdministratorAccess	4	2015-02-06 13:39 EST	2015-02-06 13:39 EST
<input type="checkbox"/>	AmazonAPIGatewayAdministr...	0	2015-07-09 13:34 EST	2015-07-09 13:34 EST
<input type="checkbox"/>	AmazonAPIGatewayInvokeFul...	0	2015-07-09 13:36 EST	2015-07-09 13:36 EST
<input type="checkbox"/>	AmazonAPIGatewayPushToCL...	0	2015-11-11 18:41 EST	2015-11-11 18:41 EST
<input type="checkbox"/>	AmazonAppStreamFullAccess	0	2015-02-06 13:40 EST	2015-02-06 13:40 EST
<input type="checkbox"/>	AmazonAppStreamReadOnlyA...	0	2015-02-06 13:40 EST	2016-12-07 16:00 EST
<input type="checkbox"/>	AmazonAppStreamServiceAcc...	0	2016-11-18 23:17 EST	2016-11-18 23:17 EST
<input type="checkbox"/>	AmazonAthenaFullAccess	0	2016-11-30 11:46 EST	2016-11-30 11:46 EST

Cancel Previous Next Step

Figure 14 Attaching policy to the role

On the Attach Policy screen, always select **AdministratorAccess** managed policy and go to the next step.

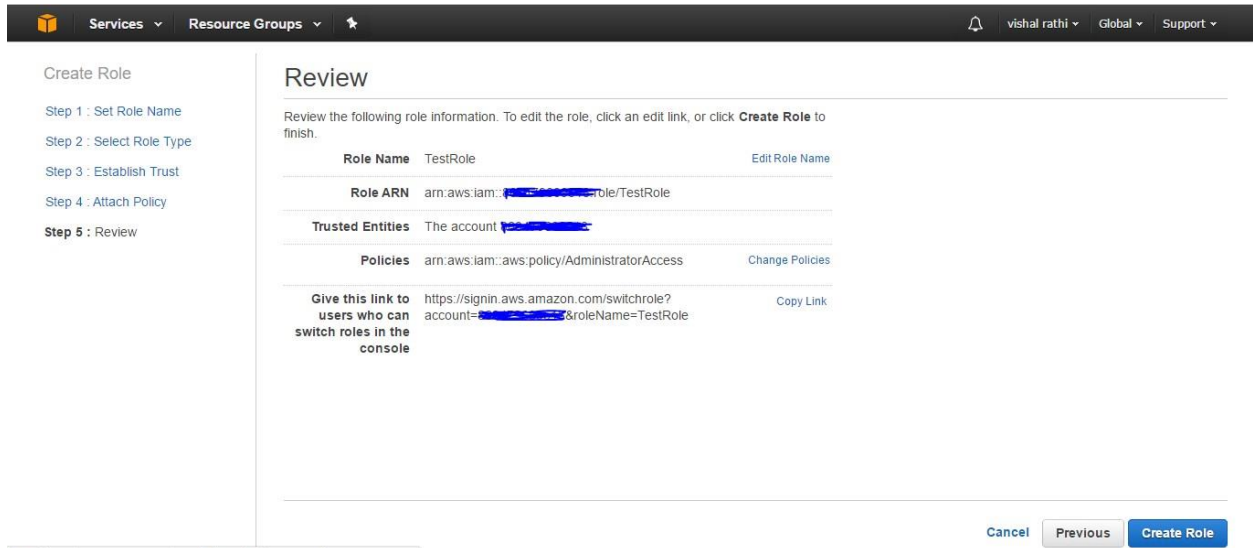


Figure 15 Review screen of the new role created

You will now see the review screen for the role that you just created. Copy and enter the **Role ARN** on MathWorks cloud center screen from figure 5. Click on the **Create Role** button to create the role. This will successfully add the new role to your AWS account. Add the optional description.

Click Save to save your settings. After you have entered your AWS credentials, when you click User Preferences you get the options to edit your credentials and time zone.

9. The next step is to create the cloud cluster after you have linked your MathWorks account with your AWS account. Click on Create a cluster in the navigation pane on MathWorks cloud center. Specify your cluster characteristics, including:
 - a. Cluster name
 - Specify the name for your cluster
 - b. MATLAB® version
 - The MATLAB® version that you are running
 - c. Cluster termination

- i. Automatically terminate cluster: an optional timeout for the cluster so that it shuts down automatically.
- ii. When cluster is idle: When the cluster no longer has any jobs to process, it will eventually shut itself down after a few minutes if no more work is submitted.
- iii. After a set time period: The cluster shuts down after the specified amount of time, whether busy or idle.
- iv. Never: The cluster continues to run until you manually shut it down.
 - For this, it is recommended to select **When cluster is idle** to save money when the cluster is not doing anything. A timeout of 15 mins is set by default. This timeout can be changed to fit the needs of the user.
- d. Cluster Log Level: Select a cluster log level. The cluster log contains MATLAB® job scheduler and worker log output. The default log level is Low. If you need to diagnose cluster issues with support engineers, you can increase the log level for more detail. Log levels above Medium can negatively impact performance.
 - For this, it is recommended to stick to the default log level which is low.
- e. Region where your cluster will run. In selecting a region, consider your location and connectivity.
 - For users in the United States, please select **US East (N. Virginia)**.
- f. Network: Cloud Center enables customers to leverage EC2-Classic or Amazon Virtual Private Cloud (VPC) for networking. Select a network that meets the requirements for Connecting a Desktop Computer (Client Machine) to MATLAB® Distributed Computing Server™ running on the Amazon EC2 Cloud. For information on configuring a VPC for use with Cloud Center, see Network Configurations.
 - Please stick to the default option.
- g. Machine type: types vary by hardware specification including number of cores, memory, and GPU support.
 - The machine type depends on the application of the user. The options are discussed in detail later in this report. The

three categories available are memory optimized, compute optimized and GPUs as shown below:

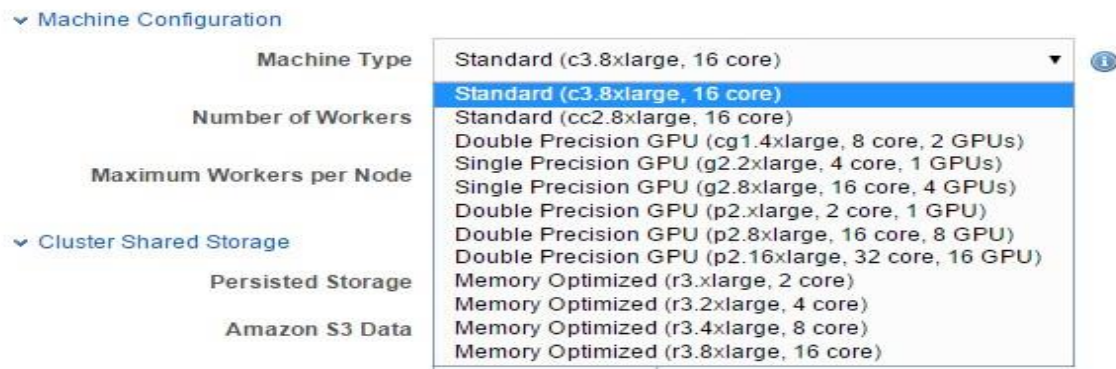


Figure 16 Machine types available on the MathWorks cloud center

- h. Total number of workers
 - The number of workers needed for this cluster. The maximum limit is 256.
- i. Number of workers per node
 - Specify the number of workers per node. The maximum limit is 16.
- j. SSH key name: The SSH key is required to start and login as root to your cloud cluster nodes. Cluster nodes have no password, so you use a key to login using SSH. When you create a cluster, you can select from the existing keys for the specified region of your AWS account, or you can request that Cloud Center create a new key. If you click create a new key, the following dialog appears for you to provide a name.

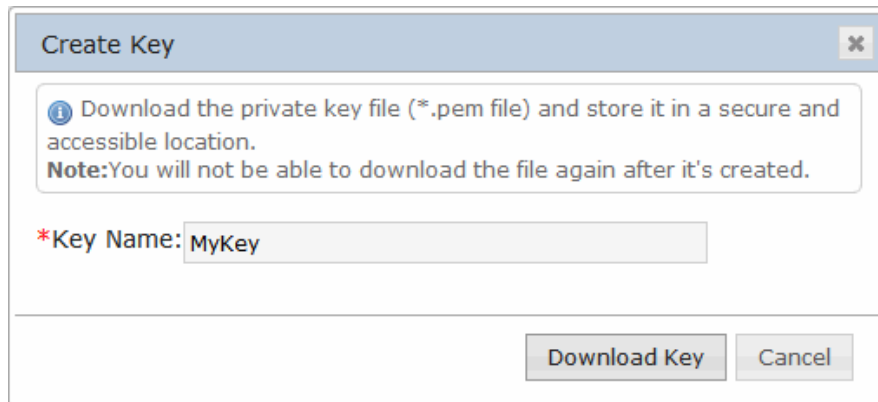


Figure 17 Create new key on MathWorks cloud center

- k. Enter a name, and click Download Key. Your browser might require you to identify a location for the download. This is a root access key file having the extension .pem. Do not lose this file, because you cannot download it again. (However, you can always create a new key, and download its key file)
- l. You can specify the same SSH key for multiple clusters. Cloud Center also provides a non-root user access key file, unique to each cluster. For information about downloading the user access key file, see Download SSH Key Identity File.
- m. Persisted storage space. For more information on persisted storage, see Cluster Shared File System.
- n. Data files to add to the worker machines. If you want to transfer files from your Amazon S3 account to the cluster nodes when the cluster starts up, click Add Files. You can specify S3 files only when creating your cluster and starting it for the first time. See Transfer Data from Amazon S3 Account.

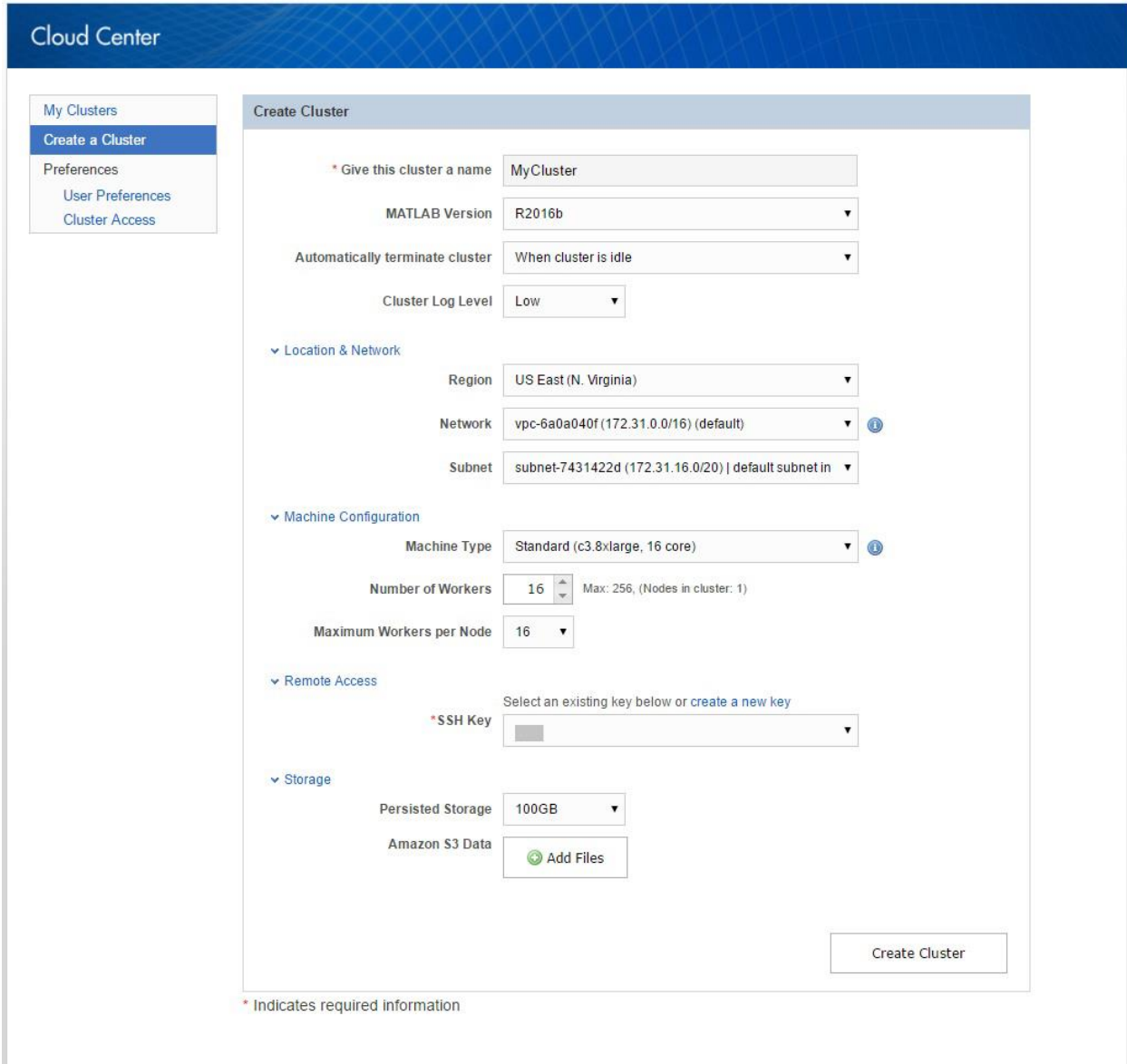


Figure 18 Creating an Amazon EC2 cloud cluster on MathWorks cloud center




After all the information is entered on figure 7, click on the **Create Cluster** button shown. At the end of this step, we have successfully created our cluster.


The figure shows typical settings for a standard 32-worker cluster with a 2-hour time limit. Click Create Cluster to create and start your cluster nodes. The cluster starts a number of nodes (instances) determined by your choices of number of workers and workers per node. During the time it takes for your cluster to start, the Cloud Center indicates the cluster status as Starting, and indicates the interim status of all the cluster nodes:

MyCluster

Do you need to access this cluster from another computer or location?
[Add cluster access for other locations.](#)

Status: Online

 Edit  Shut Down  Delete


Cluster Summary  MATLAB Cluster Profile

MATLAB Job Scheduler Host: ec2-174-129-103-221.compute-1.amazonaws.com
Started: 2016-08-04 @ 11:19AM EDT
Expires: 2016-08-04 @ 1:19PM EDT
Default Run Time: 2 Hours
Workers Requested: 32
Region: US East (N. Virginia)
Network: vpc-d45953b1 (172.31.0.0/16) (default)
Subnet: subnet-dfa1a8e5 (172.31.32.0/20) | default subnet in us-east-1e

[▼ More Details](#)

Machine Type: Standard (c3.8xlarge, 16 core)
Persisted Storage: 100GB
Cluster Nodes Requested: 2
Cluster Nodes Available: 2
Maximum Workers per Node: 16
Cluster Log Level: Low
Security Group: cluster-access-3904399
MATLAB Version: R2016a
SSH Keys: [User Access](#) (Username: clouduser)
Root Access (Keyname in EC2:joss | Username: ubuntu)
Operating System Image (AMI): Use MathWorks Image
Amazon S3 Data: None

Cluster Details

[> Headnode](#)  Online


[> Worker](#)  Online

Figure 19 Typical cluster on the cloud center

It can take up to several minutes for a cluster to completely start up, with the status indicating the particular stages of the process. You can click More Details to see further information about your cluster, including any status messages. To get further status information on any individual cluster node, click the appropriate Head node or Worker expanders. When the cluster is started and ready for use, the Cloud Center indicates the cluster status as Online. If the cluster fails to start completely, its status will indicate

that. For information on the failure, click the appropriate head node or Worker expander to read the respective log. Often you can shut down your failed cluster and attempt to start it again.

View Clusters

You can have more than one cluster, some running (online) and some shut down (offline). Click My Clusters to see a list of your clusters. The following listing shows a pair of clusters, one currently online and ready, and the other offline:



Cluster Name	Region	Workers	Status	Date Created	MATLAB Version	Actions
myCluster2		2	Offline	2016-06-10	R2015a	<input type="button" value="Start Up"/> <input type="button" value="Delete"/>
MyCluster		1	Online	2016-04-27	R2016a	<input type="button" value="Shut Down"/> <input type="button" value="Delete"/>

Figure 20 View the available clusters on cloud center

For detailed information about a particular cluster, click its name in the list.

Once the cluster is created, this cluster can be accessed by the MATLAB® client. In the Home tab on MATLAB® desktop, click on Parallel and then click on Manage Cluster Profiles. In the screen that shows up, click on the Add button on the top left corner.

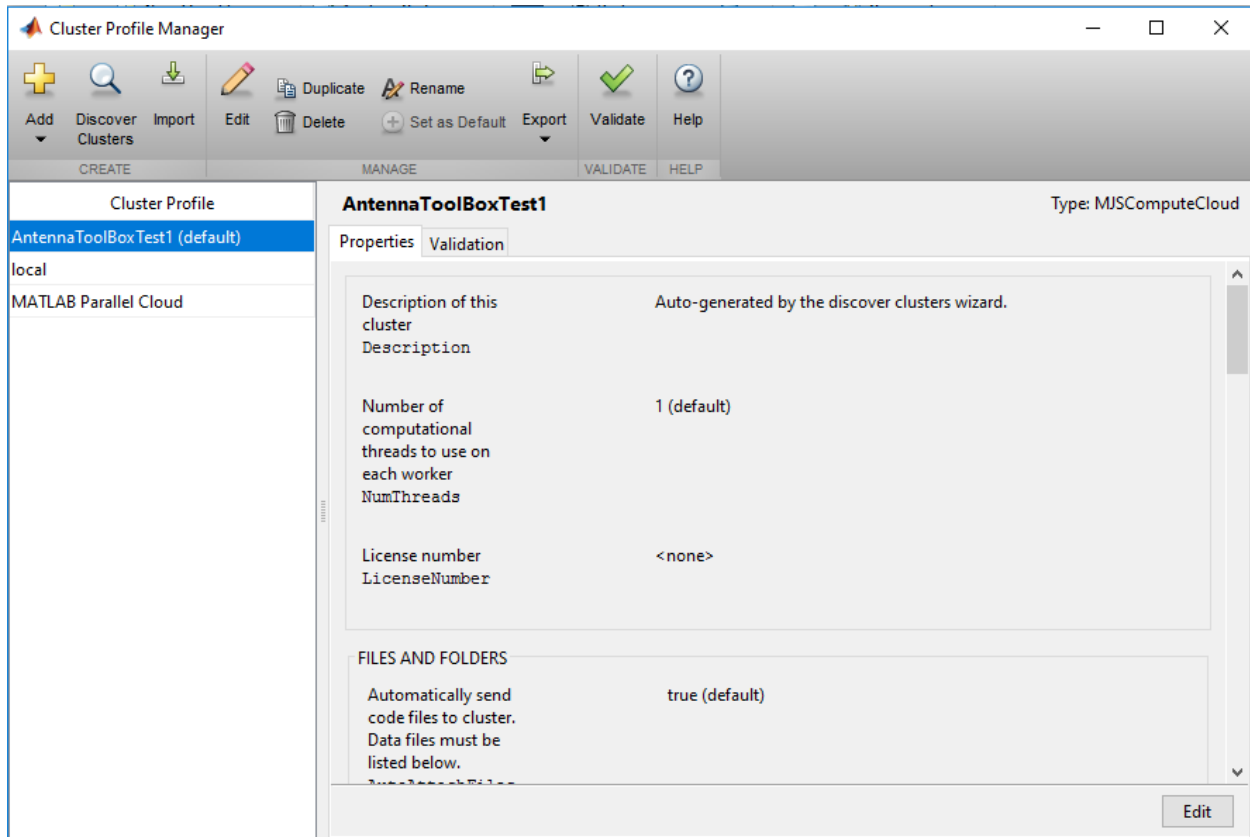


Figure 21 Cluster profile on the MATLAB® client

After clicking Add, click on Discover Clusters to add clusters on the cloud center. It will ask for MathWorks account details to access the cluster. Once added, the cluster can be seen in the Default Cluster list.

The cloud cluster works in the same way as the local machine. A parpool can be created with the cloud cluster and a parfor loop will use that parpool to run code in parallel.

3.2 Local and Cloud clusters details

The parallelization of Antenna Toolbox™ solutions was done on both the local machine and the Amazon EC2 cloud cluster. The configuration of the local machine and cloud cluster was kept constant.

3.2.1 Local cluster

256 GB RAM AMD Opteron Processor 6348 (2.80 GHz) with 48 Physical Cores

3.2.2 Amazon EC2 cloud cluster

Cluster Summary

MATLAB Cluster Profile

MATLAB Job Scheduler Host: Not yet assigned
Default Run Time: Until cluster is idle
Workers Requested: 32
Region: US East (N. Virginia)
Network: vpc-8950b6ef (172.31.0.0/16) (default)
Subnet: subnet-0777bc3b (172.31.32.0/20) | default subnet in us-east-1e

[▼ More Details](#)

MATLAB Version: R2016b
Machine Type: Standard (c3.8xlarge, 16 core)
Cluster Nodes Requested: 2
Cluster Nodes Available: 0
Maximum Workers per Node: 16
Persisted Storage: 100GB
Cluster Log Level: Low

Figure 22 Amazon EC2 cloud cluster details

The cloud cluster has a total of 4 nodes with 16 physical cores each. The machine type is c3.8xlarge with configuration as shown below:

C3

Features:

- High Frequency Intel Xeon E5-2680 v2 (Ivy Bridge) Processors
- Support for [Enhanced Networking](#)
- Support for clustering
- SSD-backed instance storage

Model	vCPU	Mem (GiB)	SSD Storage (GB)
c3.large	2	3.75	2 x 16
c3.xlarge	4	7.5	2 x 40
c3.2xlarge	8	15	2 x 80
c3.4xlarge	16	30	2 x 160
c3.8xlarge	32	60	2 x 320

Figure 23 Amazon EC2 compute optimized (C3) instances details

This machine is a compute optimized machine with use cases such as high performance front-end fleets, web-servers, batch processing, distributed analytics, high performance science and engineering applications, ad serving, MMO gaming, and video-encoding.

4. Results

The major part of solving an antenna is the LU factorization of the complex symmetric matrix. “LU factorization factors a matrix as the product of a lower triangular matrix and an upper triangular matrix”. [1] Therefore, to benchmark the performance of Antenna Toolbox™ solutions, benchmarking of LU decomposition was done on both the local machine and the cloud cluster. Different size matrices were used to establish the performance of LU decomposition on both the local and cloud cluster. The code used is included in the appendix of this report.

4.1 LU Factorization Benchmarking

Size of the matrix

5000 (complex symmetric)

LU speed up plot for cloud cluster (64 physical cores)

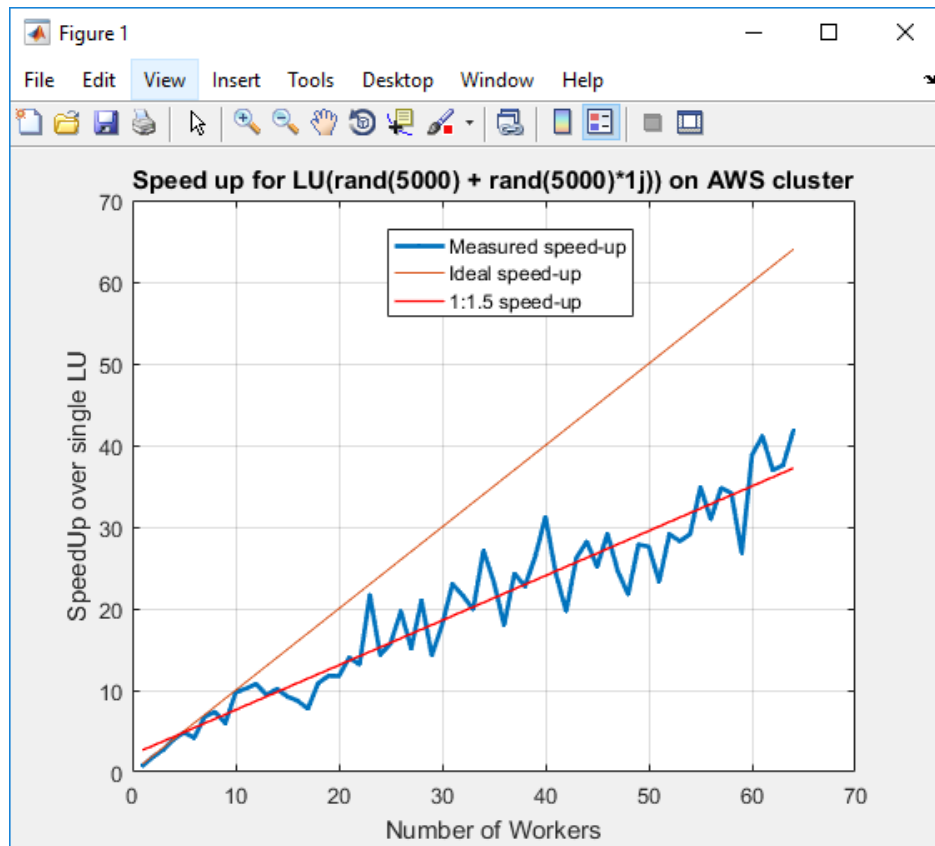


Figure 24 LU Factorization speed up plot for cloud cluster (5000 matrix size)

The plot above shows the ideal speed-up 1:1 that one should be getting in the ideal conditions. But, of course, we don't live in an ideal world. As discussed, the most time and memory intensive step in solving an antenna is the LU factorization of a complex symmetric matrix. Therefore, the performance of parallelization of Antenna Toolbox™ solutions would depend on the performance of LU factorization. Therefore, the benchmarking for LU decomposition was done to establish a speed-up reference that is more practical to achieve for the Antenna Toolbox™ solutions. As shown in figure 13, the speed up obtained is very close to 1:1.5 instead of the ideal 1:1. The maximum speed up that is obtained with 64 physical workers is ~48.

LU speed up plot for local machine (32 physical cores)

Here is the corresponding LU speed up plot obtained on the local machine:

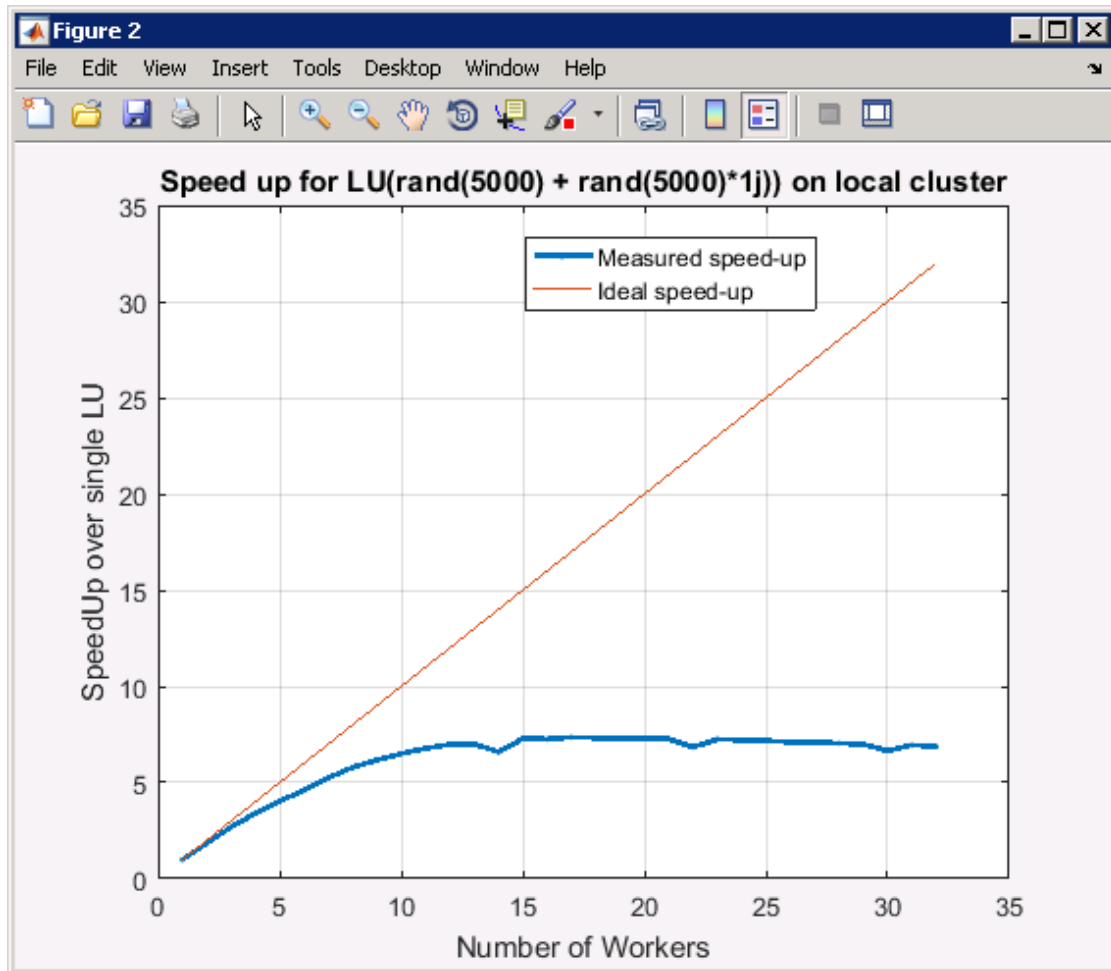


Figure 25 LU Factorization speed up plot for local cluster (5000 matrix size)

Size of the matrix:

9552 (complex symmetric)

LU speed up plot for cloud cluster (64 physical cores)

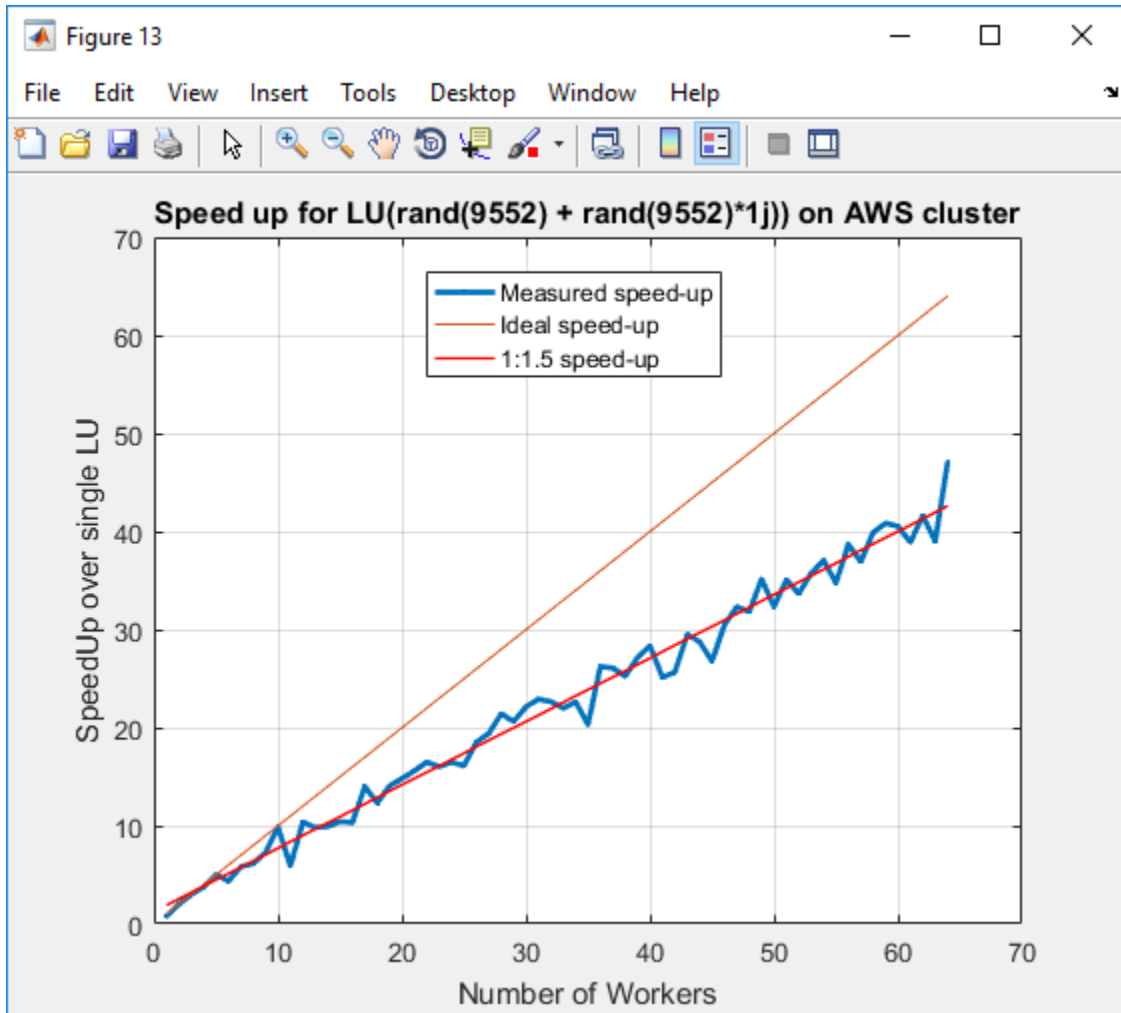


Figure 26 LU Factorization speed up plot for cloud cluster (9552 matrix size)

To further establish the performance of LU factorization, the size of the complex symmetric matrix was changed to 9552 which is \sim double the size used before. As shown in figure 15, the speed up plot obtained is quite similar to one in figure 13. The speed up obtained is very close to 1:1.5 instead of the ideal 1:1. The maximum speed up that is obtained with 64 physical workers is \sim 48. Therefore, for a similar size problem, the speed up obtained will be based off the LU speed up plot. This gives us a reference to what should be expected instead of comparing it to an ideal speed up.

LU speed up plot for local machine (32 physical cores)

Here is the corresponding LU factorization speed up plot on the local machine:

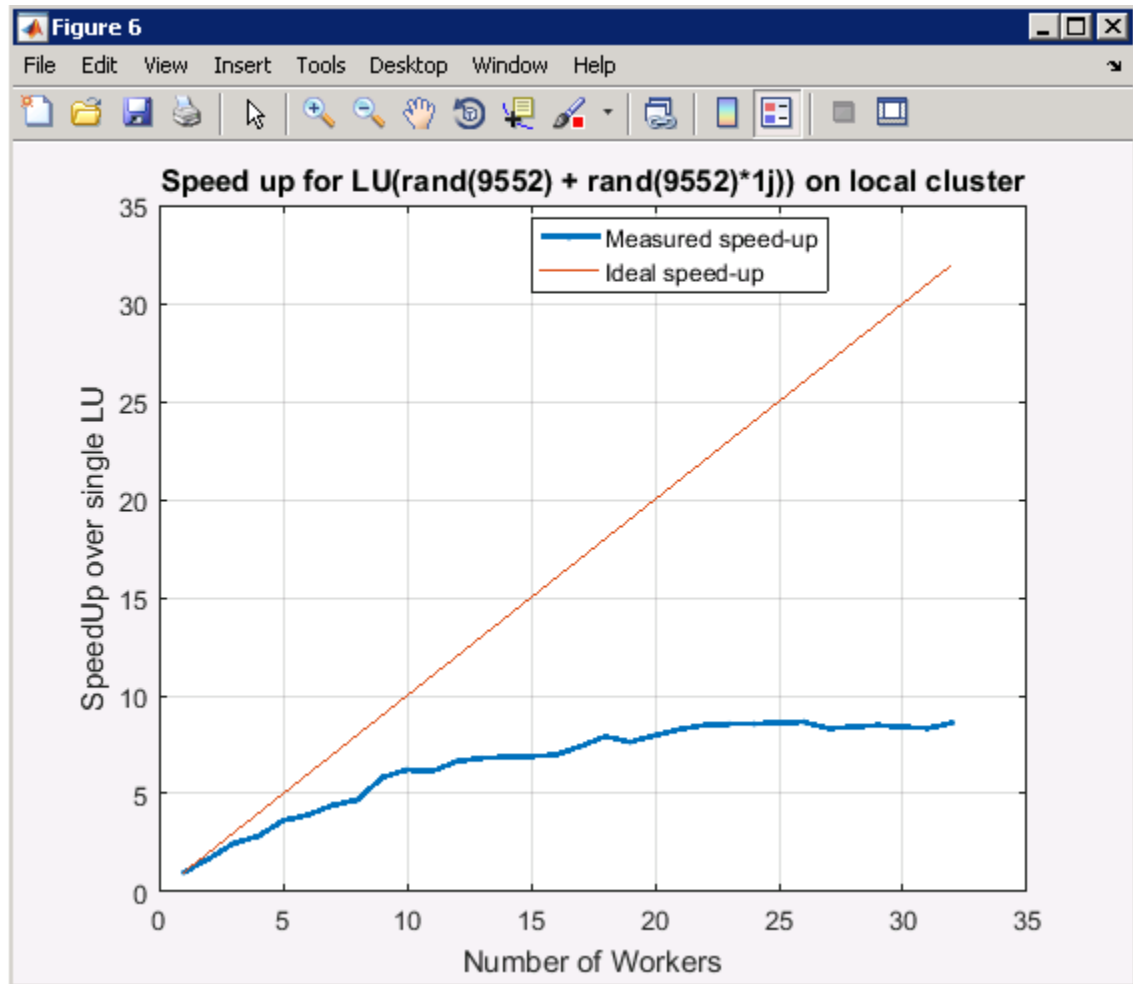


Figure 27 LU Factorization speed up plot for local cluster (9552 matrix size)

Looking at the results above, the performance comparison between the local machine and the Amazon EC2 cloud cluster can be established. The performance obtained on the cloud cluster is much better than the local machine. With the local machine, the speed up obtained saturates beyond a certain point. As seen in figure 16, the speed up saturates at ~ 8 and beyond that any additional number of workers don't add to the performance. The difference in performance between the local machine and the Amazon EC2 cloud cluster is a result of several factors. The hardware configuration of the local machine is the biggest limitation in the performance. Memory bandwidth is the rate at which data can be read from or stored

into a semiconductor memory by a processor. This can be thought of as the biggest limitation in the local machine as the cache sizes are pretty small. With the cloud cluster, the cache sizes are bigger and the instances have Solid State Drives (SSD) which makes data retrieval faster. Therefore, the performance obtained is far better than the performance on the local machine.

4.2 Use Cases

After benchmarking the performance of LU factorization, different use cases were tried on cloud clusters to verify whether the speed up results are consistent with the LU factorization of similar size problems. The code was run on the cloud cluster after starting an interactive parallel pool on the MATLAB® client session.

4.2.1 Case 1: Dipole Array

Frequency sweep of a linear array of dipoles with 40 elements

The first case that was run on the cloud cluster was a linear array of dipoles with 40 elements. Dipole is the simplest antenna structure. Here are the antenna object details:

```
dp_mesh =  
  
struct with fields:  
  
    NumTriangles: 2240  
    NumTetrahedra: 0  
    NumBasis: 2200  
    MaxEdgeLength: 0.0749  
    MeshMode: 'auto'
```

Figure 28 Dipole Linear Array object details

The number of unknown basis functions for this object are 2200. Frequency sweep was run on this object with varying number of frequency points. The script for creating the object and running it on the cloud cluster is given in the appendix section of the report. Here are the results obtained on the cloud cluster:

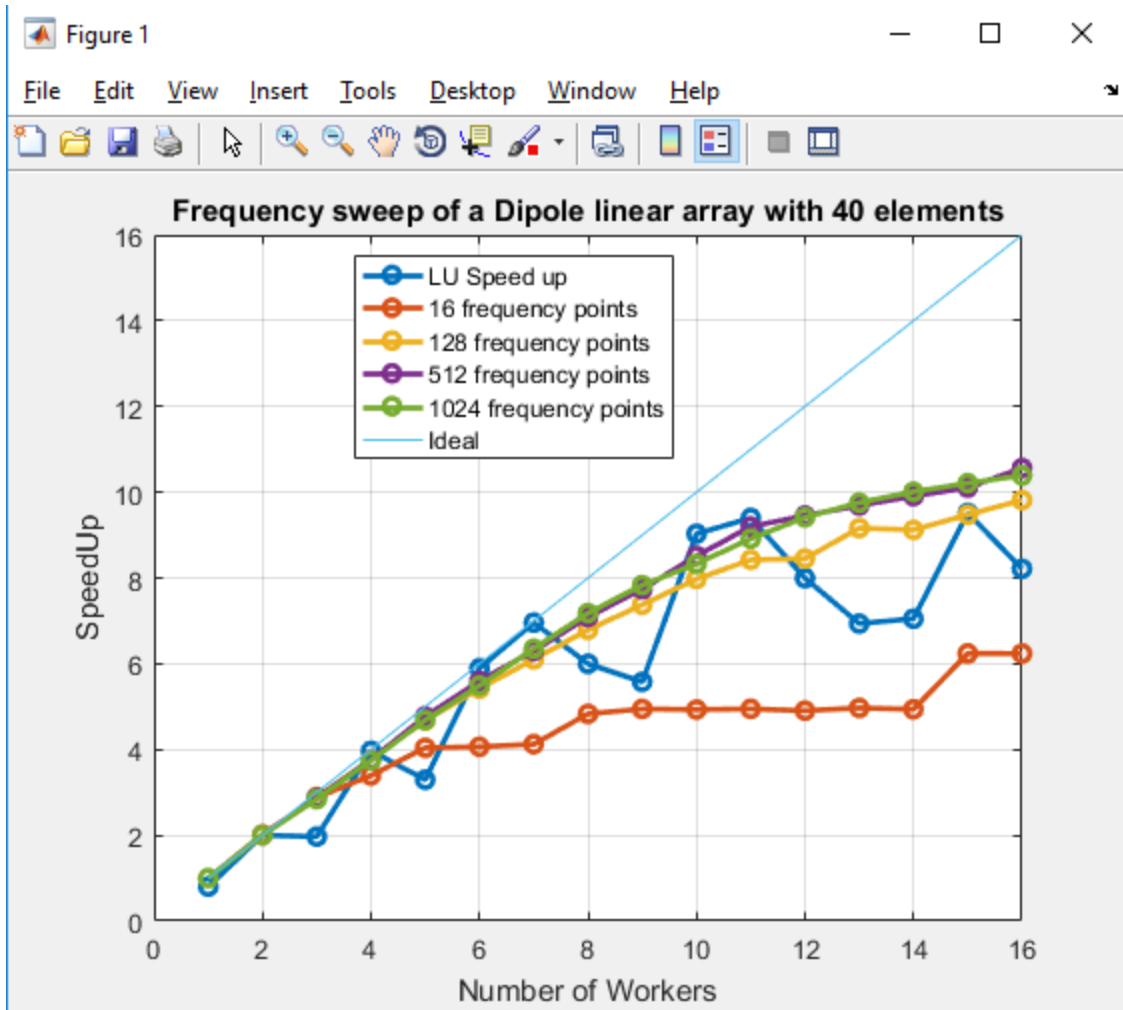


Figure 29 Speed up plot for frequency sweep on a linear dipole array

The above plot shows the speed up obtained for the linear dipole array of 40 elements on the Amazon EC2 cloud cluster. Number of workers were set to 16 and multiple frequency sweeps, each with different number of frequency points were run. The blue line in the above plot shows the LU factorization results for a problem of this size i.e. with 2200 basis functions. As shown in the plot, the speed up obtained is very comparable to the LU factorization speed up. This proves that LU factorization takes up the most time in solving the antenna.

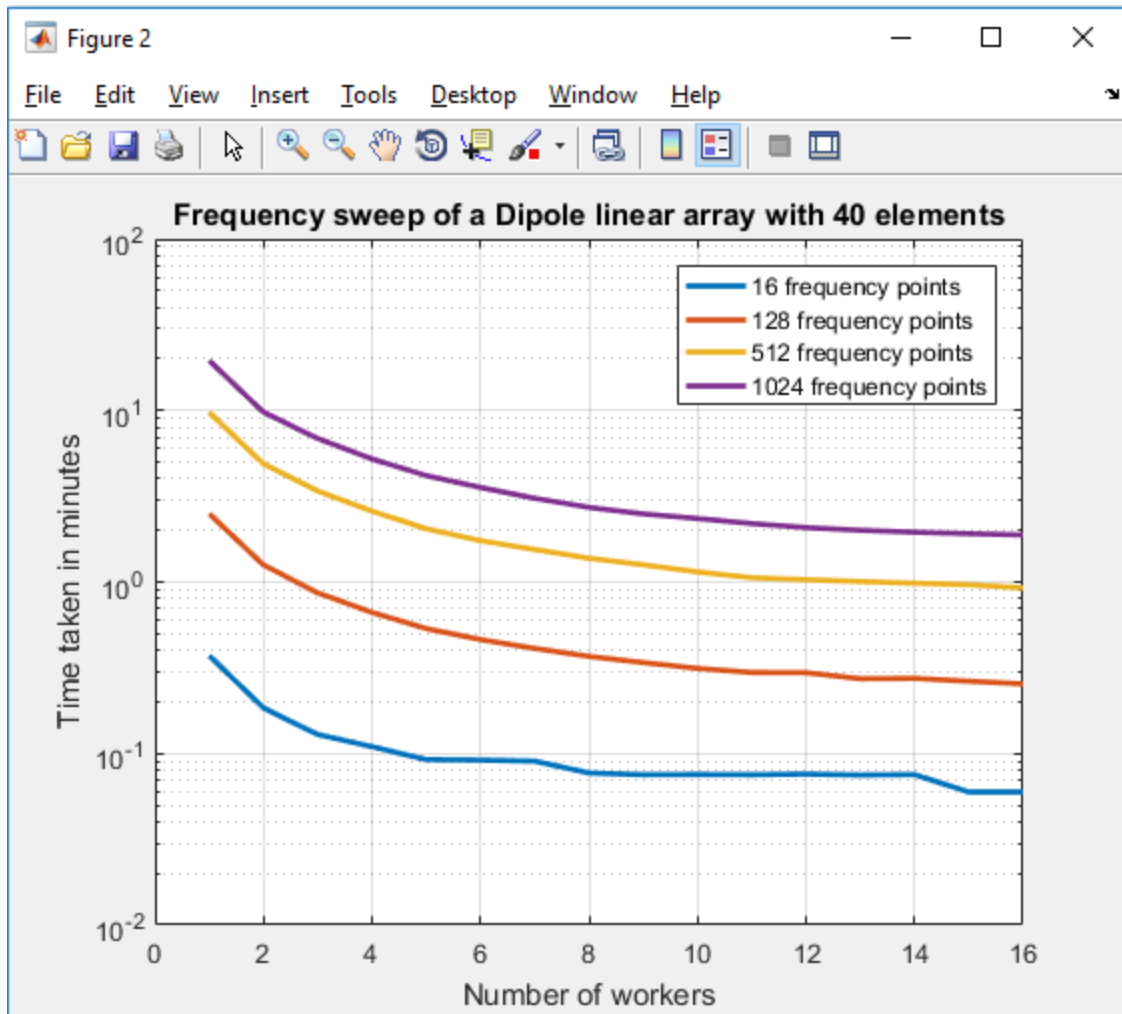


Figure 30 Time taken in minutes for frequency sweep of linear dipole array

The above plot shows the time taken in minutes to solve the linear dipole array of 40 elements over different frequency sweeps. For the case of 16 frequency points, the time taken to solve the problem is reduced from ~22 seconds to ~3.5 seconds. For the case of 1024 frequency points, the time taken to solve the entire problem on 16 workers is reduced from ~19 minutes on 1 worker to ~2 minutes on 16 workers.

4.2.2 Case 2: Spiral antenna

Frequency sweep of a spiral antenna

The second case that was run on the cloud cluster was a spiral antenna. Here are the antenna object details:

```

sp_mesh =

  struct with fields:

    NumTriangles: 1952
    NumTetrahedra: 0
    NumBasis: 2388
    MaxEdgeLength: 0.0258
    MeshMode: 'auto'

```

Figure 31 Spiral antenna object details

The number of unknown basis functions for this object are 2388. Frequency sweep was run on the object with varying number of frequency points. The script for creating the object and running it on the cloud cluster is given in the appendix section of the report. Here are the results obtained on the cluster:

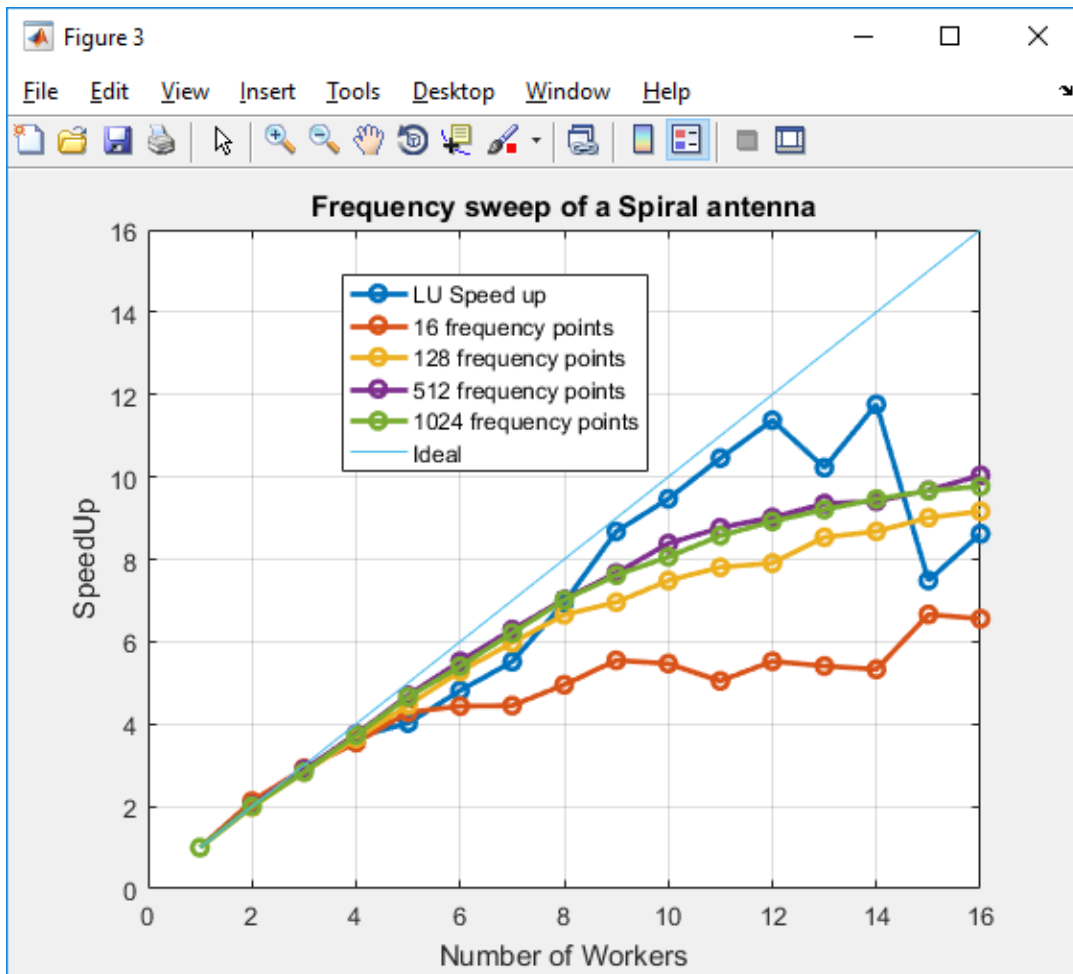


Figure 32 Speed up plot for spiral antenna

The above plot shows the speed up obtained for the spiral antenna on the Amazon EC2 cloud cluster. Number of workers were set to 16 and multiple frequency sweeps, each with different number of frequency points were run. The blue line in the above plot shows the LU factorization results for a problem of this size i.e. with 2388 basis functions. As shown in the plot, the speed up obtained is very comparable to the LU factorization speed up. This further proves that LU factorization takes up the most time in solving the antenna.

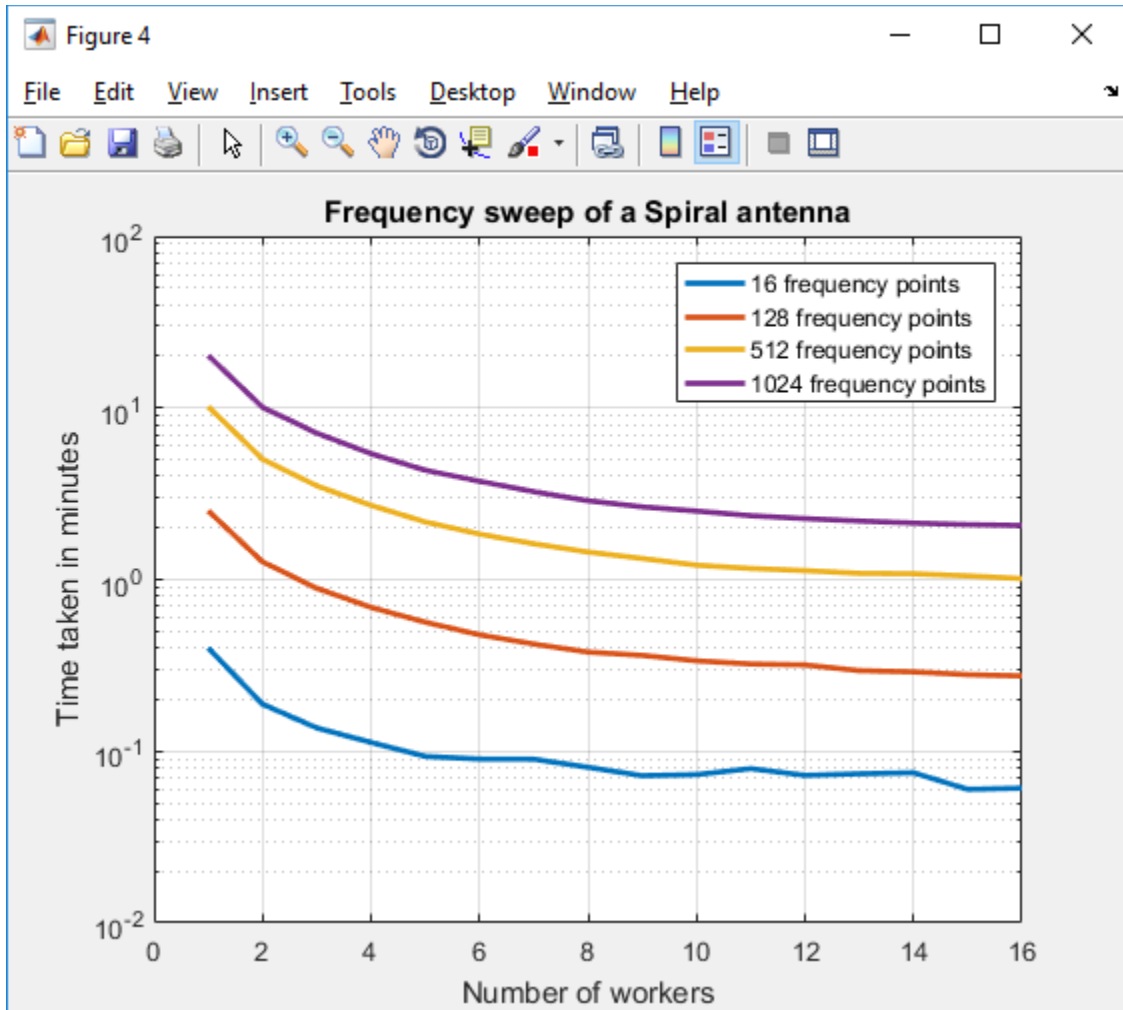


Figure 33 Time taken in minutes for frequency sweep on a spiral antenna

The above plot shows the time taken in minutes to solve the spiral antenna over different frequency sweeps. For the case of 16 frequency points, the time taken to solve the problem is reduced from ~24 seconds to ~3.6 seconds. For the case of 1024 frequency points, the time taken to solve the entire problem on 16 workers is reduced from ~20 minutes on 1 worker to ~2 minutes on 16 workers.

4.2.3 Case 3: Spiral array

Frequency sweep of a spiral antenna 2x2 array

Another case that was run on the cloud cluster was a 2x2 spiral antenna array. Here are the antenna object details:

```
mesh_r =  
  
  struct with fields:  
  
    NumTriangles: 7808  
    NumTetrahedra: 0  
    NumBasis: 9552  
    MaxEdgeLength: 0.0258  
    MeshMode: 'auto'
```

Figure 34 Spiral antenna 2x2 array object details

The number of unknown basis functions for this object are 9552. Frequency sweep was run on the object with varying number of frequency points. The script for creating the object and running it on the cloud cluster is given in the appendix section of the report. Here are the results obtained on Amazon EC2 cloud cluster:

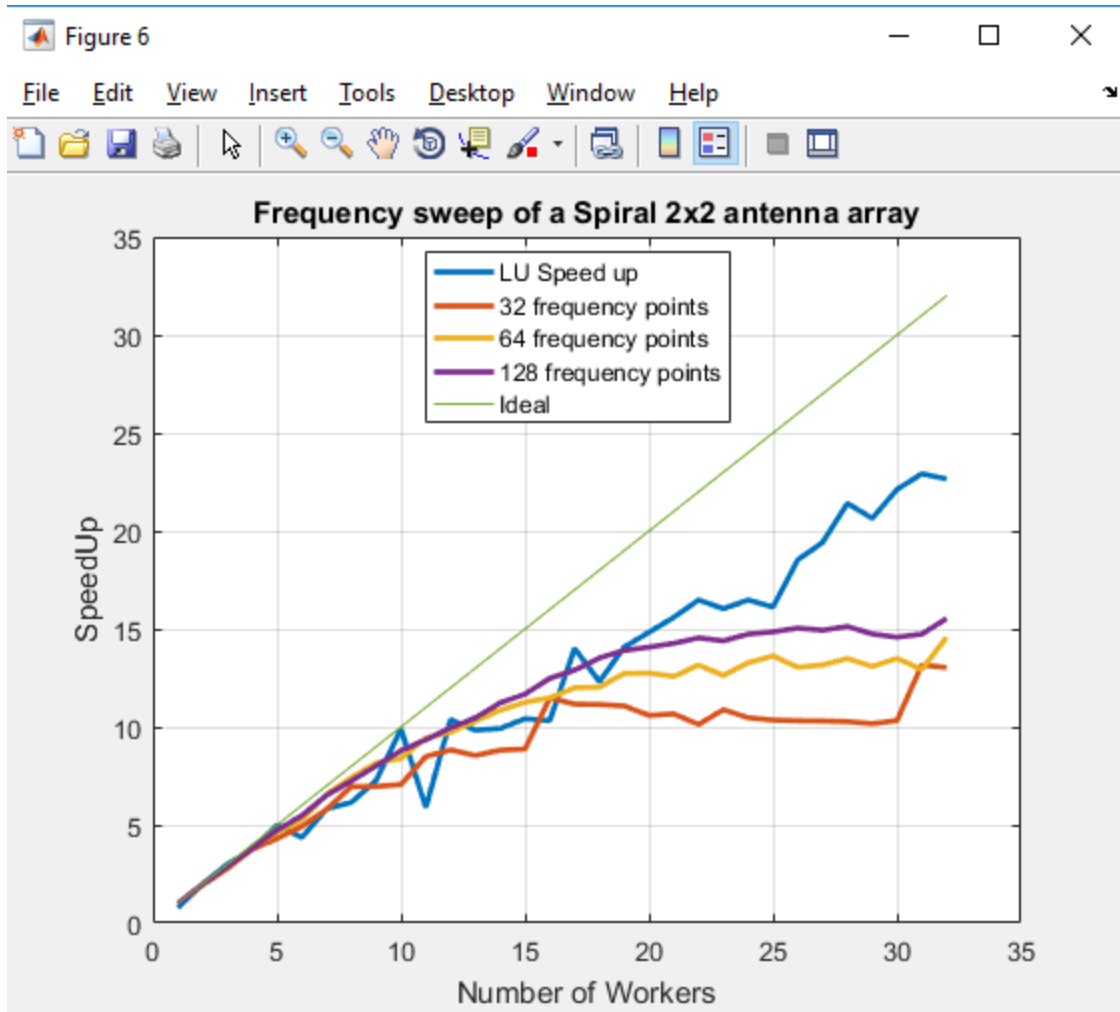


Figure 35 Speed up plot for spiral antenna 2x2 array

The above plot shows the speed up obtained for the spiral antenna 2x2 array on the Amazon EC2 cloud cluster. Number of workers were set to 32 and multiple frequency sweeps, each with different number of frequency points were run. The blue line in the above plot shows the LU factorization results for a problem of this size i.e. with 9552 basis functions. As shown in the plot, the speed up obtained is very comparable to the LU factorization speed up. This further proves that LU factorization takes up the most time in solving the antenna.

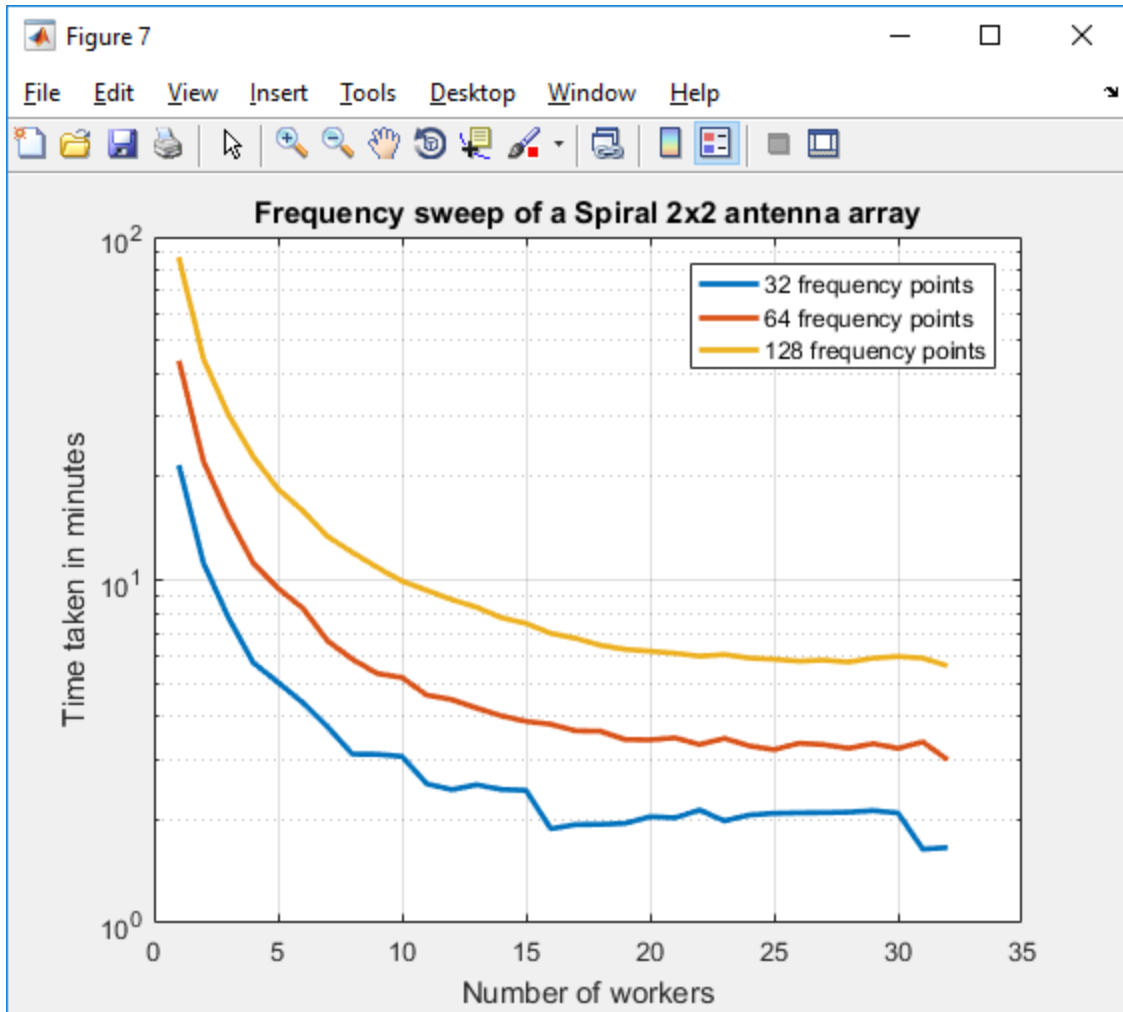


Figure 36 Time taken in minutes for frequency sweep on a spiral antenna 2x2 array

The above plot shows the time taken in minutes to solve the spiral antenna over different frequency sweeps. For the case of 32 frequency points, the time taken to solve the problem is reduced from ~21 minutes to ~1.6 minutes. For the case of 128 frequency points, the time taken to solve the entire problem on 32 workers is reduced from ~87.24 minutes on 1 worker to ~5.6165 minutes on 32 workers.

4.2.4 Case 4: Patch antenna on a substrate

Frequency sweep of a patch antenna on a Taconic TLC substrate

Another case that was run on the cloud cluster was a patch antenna on a Taconic TLC substrate. Here are the antenna object details:

```

p1_mesh =

  struct with fields:

    NumTriangles: 3150
    NumTetrahedra: 7218
    NumBasis: 14363
    MaxEdgeLength: 0.0042
    MeshMode: 'auto'

```

Figure 37 Patch antenna object details

The number of unknown basis functions for this object are 14363. Frequency sweep was run on the object with varying number of frequency points. The script for creating the object and running it on the cloud cluster is given in the appendix section of the report. Here are the results obtained on the Amazon EC2 cloud cluster.

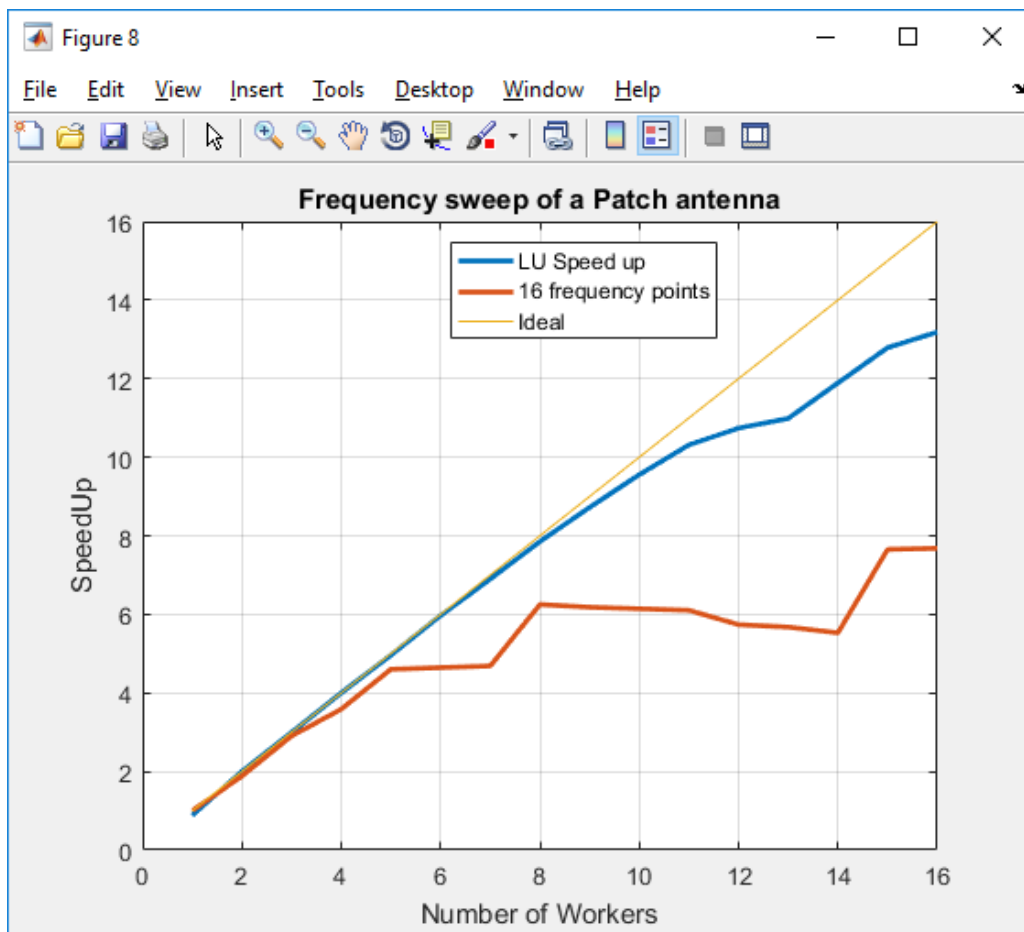


Figure 38 Speed up plot for Patch antenna

The above plot shows the speed up obtained for the patch antenna on the Amazon EC2 cloud cluster. Number of workers were set to 16 and number of frequency points was also set to 16. The blue line in the above plot shows the LU factorization results for a problem of this size i.e. with 14363 basis functions. As shown in the plot, the speed up obtained is comparable to the LU factorization speed up.

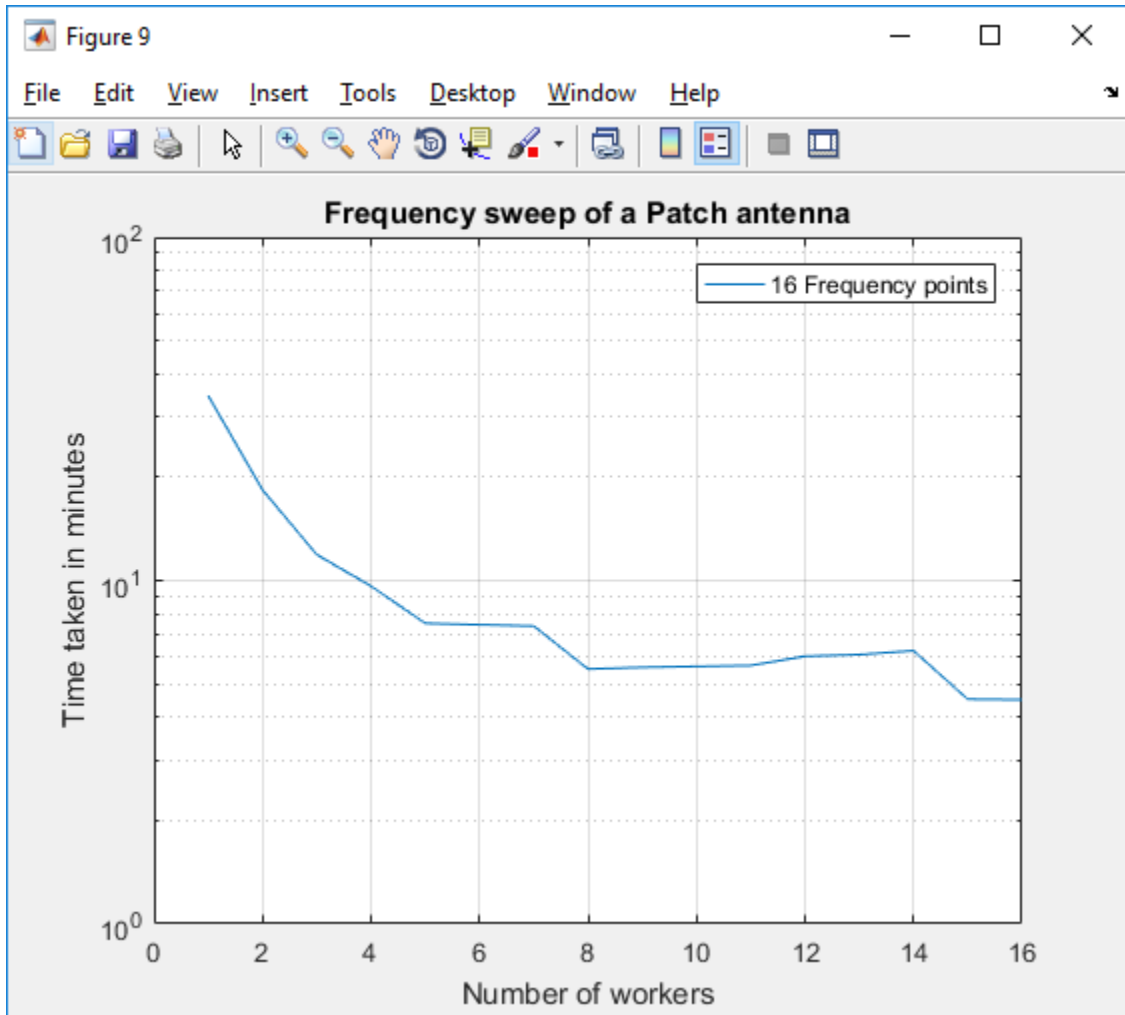


Figure 39 Time taken in minutes for frequency sweep on a patch antenna

The above plot shows the time taken in minutes to solve the patch antenna over a frequency sweep. The time taken to solve the problem is reduced from ~34.5 minutes on 1 worker to ~4.5 minutes on 16 workers.

4.3 Running batch jobs on Amazon EC2 cloud cluster

Batch jobs can be run on the cloud cluster to offload execution of functions to run in the background. When working interactively in a MATLAB® session, work can be offloaded to a MATLAB® worker session to run as a batch job. The command to perform this job is asynchronous. This doesn't block the client MATLAB® session which is very helpful. The client session can be simultaneously used for further development as the batch job runs on either the same machine as the client or on cloud cluster if using the MATLAB® Distributed Computing Server™.

A batch job can be run with a parallel pool. The first step is to create a script with a parfor-loop. An example of such script is included in the appendix of this document. After the script is saved, it can be run in MATLAB® with the batch command. It is important to indicate that the script should use a parallel loop for the loop. Here is the batch command:

```
job = batch('testWithParRadar','Profile', 'AntennaToolBoxRadarBookCluster', 'Pool', 31 , 'AttachedFiles',  
{'TxRadiator.mat'});
```

The command above specifies that 31 workers in addition to the one running the batch script are to evaluate the loop iterations. Therefore, a total of 32 workers are used. Files from the client workspace can also be sent with the batch command using the AttachedFiles attribute. To view the results:

```
wait(job)
```


```
load(job)
```

After the job is completed, the data can be permanently deleted and the reference to the job can be removed from the workspace.

4.4 AWS memory optimized instance

As part of the MathWorks cloud center, memory optimized instances can also be created to solve memory intensive problems. R3.8xlarge EC2 instance can be created for memory intensive antenna solutions. The cluster details are given below:

Cluster Summary

 [MATLAB Cluster Profile](#)

MATLAB Job Scheduler Host: ec2-54-146-46-53.compute-1.amazonaws.com
Started: 2017-02-26 @ 12:10AM EST
Expires: When cluster is idle
Default Run Time: Until cluster is idle
Workers Requested: 32
Region: US East (N. Virginia)
Network: vpc-8950b6ef (172.31.0.0/16) (default)
Subnet: subnet-0777bc3b (172.31.32.0/20) | default subnet in us-east-1e

[▼ More Details](#)

MATLAB Version: R2016b
Machine Type: Memory Optimized (r3.8xlarge, 16 core)
Cluster Nodes Requested: 2
Cluster Nodes Available: 2
Maximum Workers per Node: 16
Persisted Storage: 100GB
Cluster Log Level: Low

Figure 40 AWS memory optimized instance details on cloud center

This cluster has a total of 2 nodes with 16 physical cores each. The machine type is r3.8xlarge with configuration as shown below:

R3

R3 instances are optimized for memory-intensive applications and offer lower price per GiB of RAM.

Features:

- High Frequency Intel Xeon E5-2670 v2 (Ivy Bridge) Processors
- SSD Storage
- Support for [Enhanced Networking](#)

Model	vCPU	Mem (GiB)	SSD Storage (GB)
r3.large	2	15.25	1 x 32
r3.xlarge	4	30.5	1 x 80
r3.2xlarge	8	61	1 x 160
r3.4xlarge	16	122	1 x 320
r3.8xlarge	32	244	2 x 320

Figure 41 EC2 memory optimized (R3) instance details

4.4.1 RadarBookColumn Problem

Another problem that was investigated as part of this project was the radar book column problem. The antenna is created from the following image:

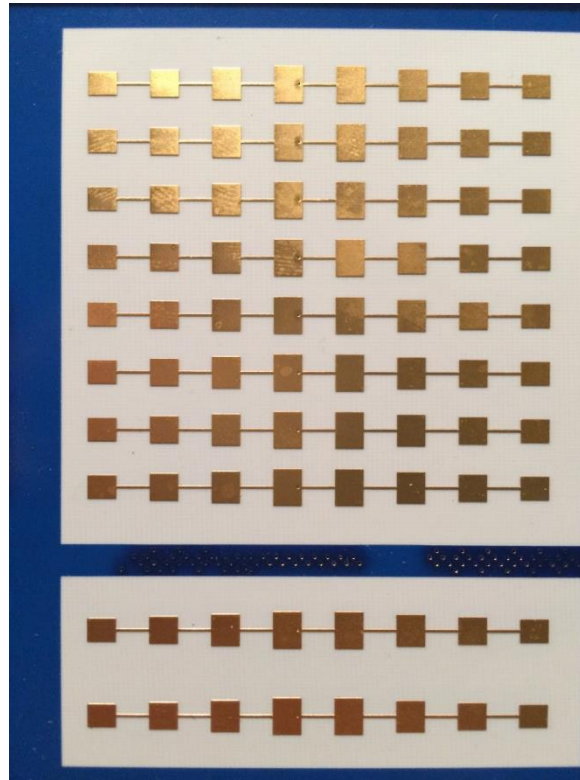


Figure 42 Patch antenna image

Reflector and a dielectric are also added to the antenna object. The antenna object details are:

```
mm1 =  
  
struct with fields:  
  
    NumTriangles: 5564  
    NumTetrahedra: 11190  
    NumBasis: []  
    MaxEdgeLength: 8.0000e-04  
    MeshMode: 'manual'
```

Figure 43 Radar antenna object details

With 0.8×10^{-3} edge length, the object has a 23K number of basis functions. For a frequency sweep consisting of 24 frequency points, the total amount of memory needed to hold the matrices is ~ 96 GB. Therefore, the compute optimized machine cloud cluster which has a total memory of 60 GiB can't be used for this problem since it will throw an out of memory error. Therefore, the memory optimized cluster was created to solve this problem. With a memory of 256 GiB, this cluster can easily handle a frequency sweep of 24 points.

This problem was also run as a batch job to completely isolate the MATLAB® client. The time taken to solve the entire problem is ~ 26 minutes. The impedance plot shows the results obtained.

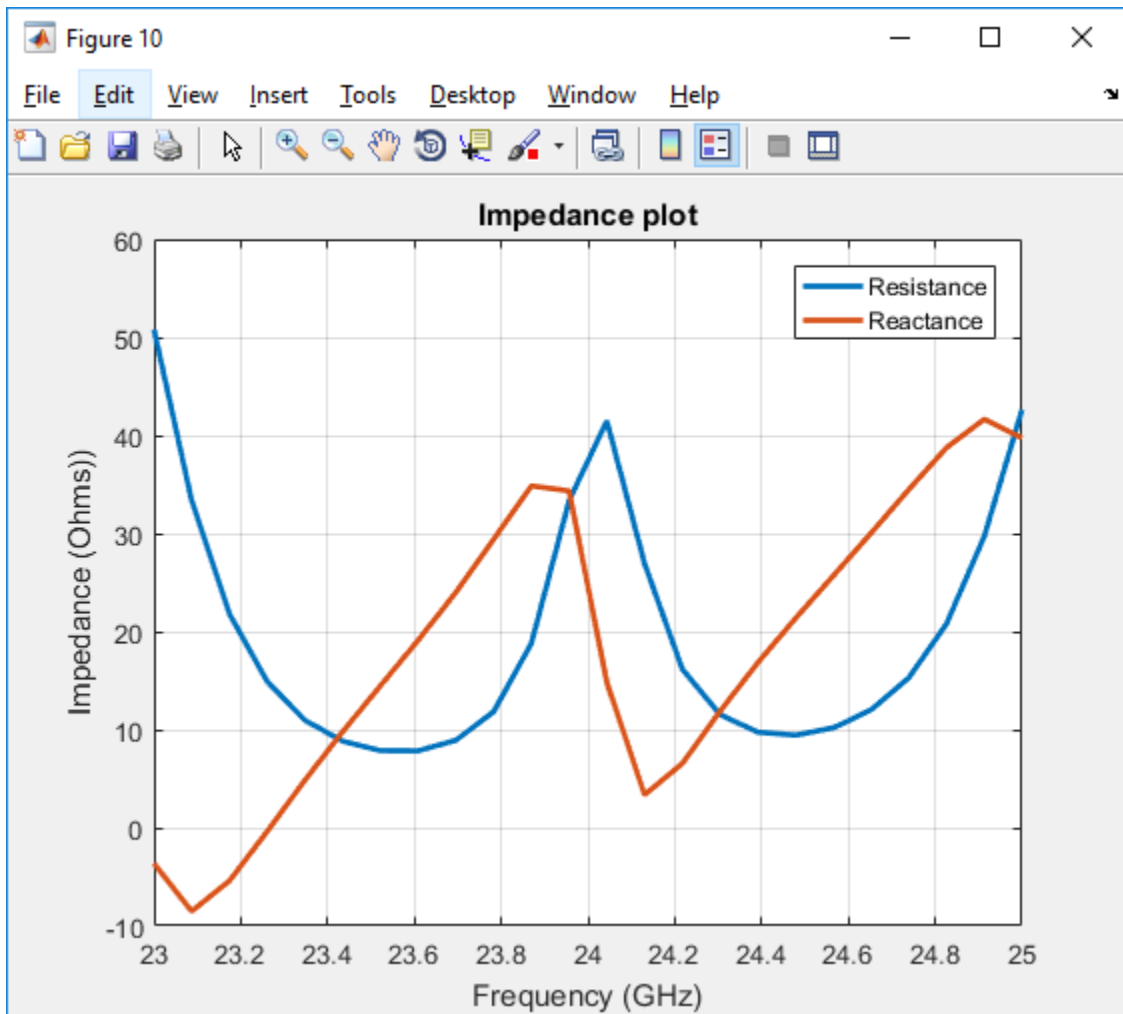


Figure 44 Impedance plot for the radar antenna

5. Conclusion

This project was designed to explore the parallelization of Antenna Toolbox™ solutions on both the local and cloud clusters. The local machine which was used for this project was a WPI server called sunfire14. The cloud cluster was created on Amazon EC2. Performance of LU factorization was established on both the local and cloud cluster. After the performance of LU factorization was benchmarked, it became very clear that parallelization was much better on the cloud cluster. The performance was significantly better on the Amazon EC2 cloud cluster as shown in the results section in this report.

After benchmarking the performance of LU factorization, several use cases were run on the Amazon EC2 cloud cluster. By looking at the results, it can be concluded that LU matrix decomposition is the most time and memory intensive step in the antenna solution since the performance i.e. the speed up obtained follows the corresponding LU factorization speedup of the similar sized matrix very closely. The computation times to solve the antenna are significantly reduced after parallelizing them on Amazon EC2 cloud cluster.

The biggest advantage of using a cloud cluster on Amazon EC2 is the tremendous amount of flexibility that it offers. Users don't have to worry about setting up their own hardware at all. The desired hardware configuration is at their fingertips with Amazon EC2. There are variety of instances that can be created with Amazon EC2 to match the problem needs. If users need more memory then they can launch an instance with a lot of RAM like r3.4x large to solve their memory intensive problem. If the problem is computation intensive then compute optimized instance like c3.4x large can be launched to solve the problem. In addition to this, multiple instances can be created at the same time to divide the problem size into parts to further decrease the computation time. Moreover, running jobs on cloud cluster also doesn't block the user's MATLAB® client which allows the user to continue with his/her development work. Users don't have to worry about the local hardware. This makes Amazon EC2 very desirable for the users to parallelize their Antenna Toolbox™ solutions.

The results obtained in this project are really encouraging and make Amazon EC2 really feasible for our application. Given the nature of problems that users of Antenna Toolbox™ solve, parallelization is really important to reduce the time taken to get the results. Without parallelization, sometimes the users wait for days and weeks to get the results. With Amazon EC2, the problem can definitely be solved. Overall, I was able to demonstrate the feasibility of Amazon EC2 for Antenna Toolbox™ solutions parallelization.

References

- [1]. https://en.wikipedia.org/wiki/LU_decomposition
- [2]. <https://aws.amazon.com/>
- [3]. <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
- [4]. https://en.wikipedia.org/wiki/Parallel_computing
- [5]. <https://www.mathworks.com/products/antenna.html>
- [6]. <https://www.mathworks.com/products/parallel-computing.html>
- [7]. <https://www.mathworks.com/programs/cloud-center/cloud.pdf>
- [8]. <https://www.mathworks.com/help/distcomp/introduction-to-parfor.html>
- [9]. <https://aws.amazon.com/ec2/>

Appendix

LU Scaling code

```
%This function measures the LU factorization speed up on a cluster of
workers for a particular size of complex symmetric matrix.

function [s, n] = testLUScale

N = 14363; % Size of the matrix

spmd(32)
n = 1:32;
s = [];
f = createTimedFunctionCall(N);
for ii = n
labBarrier
t = myTimeIt(f, ii);
labBarrier
tN = gop(@max, t);
fprintfOn1('Time for %d 1-threaded single LU: %f\n', ii, tN);
s(end+1) = tN; %#ok<AGROW>
end

if labindex == 1

s(1) = timeOneFun(f);

end

end

s = s(1);

n = n(1);

end

function t = timeOneFun(fun)

t = timeit(@() fun());

end

function t = myTimeIt(fun, labs)

if labindex <= labs

aT = tic;

N = 1;

for ii = 1:N

feval(fun);
```

```

end

t = toc(aT)/N;

else

t = -Inf;

end

end

function fun = createTimedFunctionCall(N)

r = rand(N) + rand(N)*1i;

fun = @() lu(r);

end

function fprintfOn1(varargin)

if labindex == 1

fprintf(varargin{:});

end

end

```

Linear dipole array

```

% This function creates a dipole antenna array and meshes it at the
% highest frequency manually
function dp = createDipoleAntenna(fmax, num)
    % Create the dipole antenna
    dp = linearArray('NumElements', num);
    % Solve the antenna to mesh it manually at the highest frequency
    temp = axialRatio(dp, fmax, 0, 90);
    % Get the meshdata
    meshdata = mesh(dp);
    % Mesh the antenna with the same edge length at the highest
    % frequency
    [~, ~] = mesh(dp, 'MaxEdgeLength', meshdata.MaxEdgeLength); end

```

Spiral antenna

```
% This function creates a spiral antenna and meshes it at the highest
frequency manually
function sp = createSpiralAntenna(fmax)
    % Create the spiral antenna
    sp =
    spiralArchimedean('Turns',4,'InnerRadius',5.5e-3,'OuterRadius',50e-3);
    % Solve the antenna to mesh it manually at the highest frequency
    temp = axialRatio(sp,fmax,0,90);
    % Get the meshdata
    meshdata = mesh(sp);
    % Mesh the antenna with the same edge length at the highest
frequency
    [~,] = mesh(sp,'MaxEdgeLength',meshdata.MaxEdgeLength); end
```

Spiral array

```
% This function creates a spiral array and meshes it at the highest
frequency manually
function r = createSpiralArray(fmax)
    % Create the spiral antenna
    sp =
    spiralArchimedean('Turns',4,'InnerRadius',5.5e-3,'OuterRadius',50e-3);
    % Create the array
    r = rectangularArray('Element', sp, 'RowSpacing',
    200e-3, 'ColumnSpacing', 200e-3, 'Size', [2, 2]);
    % Solve the antenna to mesh it manually at the highest frequency
    temp = axialRatio(r,fmax,0,90);
    % Get the meshdata
    meshdata = mesh(r);
    % Mesh the antenna with the same edge length at the highest
frequency
    [~,] = mesh(r,'MaxEdgeLength',meshdata.MaxEdgeLength);
end
```

Patch antenna

```
% This function creates a patch antenna and meshes it at the highest
frequency manually
function pl = createPatchAntenna(fmax)
    % A basic Microstrip-patch antenna
    SubstrateThickness = 0.003;
    Er = 2.3;
    Substrate = dielectric('Name', 'Taconic_TLC', 'EpsilonR',
Er, 'LossTangent', 0.0, 'Thickness', SubstrateThickness);
    pl = patchMicrostrip('Height', SubstrateThickness, 'Substrate',
Substrate);
    % Solve the antenna to mesh it manually at the highest frequency
    temp = impedance(pl, fmax);
    % Get the meshdata
    meshdata = mesh(pl);
    % Mesh the antenna with the same edge length at the highest
frequency
    [~, ~] = mesh(pl, 'MaxEdgeLength', meshdata.MaxEdgeLength);
end
```

Script with the parfor loop

```
% Function to run script on a parallel pool
function [twithpar, ARparfor, bytes] = testWithPar(sp, freq, workers)
    ARparfor = zeros(1, numel(freq));
    tic;
    ticBytes(gcp);
    parfor (m = 1:numel(freq), workers)
        maxNumCompThreads(1);
        ARparfor(m) = axialRatio(sp, freq(m), 0, 90);
    end
    twithpar = toc;
    bytes = tocBytes(gcp);
end
```

RadarBookColumn

```
% Script to create the radara antenna object and run it on the cluster
as a batch job.

TxRad = load('TxRadiator.mat');
c = TxRad.c;

% Put it on a GP
c.Tilt = 0;
c.TiltAxis = [0 0 1];

radar = design(reflector, 24e9);
radar.Exciter = c;
radar.GroundPlaneLength = 58e-3;
radar.GroundPlaneWidth = 6e-3;
radar.Spacing = 0.25e-3;
radar.EnableProbeFeed = 1;

radar.Substrate = dielectric('Name', 'R04350',
    ...
    'EpsilonR', 3.48,
    ...
    'LossTangent', 0.0040);

mml = mesh(radar, 'MaxEdgeLength', 0.8e-3);
freq = linspace(23e9, 25e9, 24);
ZparforRadar_24 = zeros(1, numel(freq));
tic;

% Par for loop
parfor (m = 1:numel(freq))
    maxNumCompThreads(1);
    ZparforRadar_24(m) = impedance(radar, freq(m));
end
twithparRadar_24 = toc;
```