Center for GIS

TOWSON UNIVERSITY

White Paper for

Assessing the Potential Benefits of Hosting Maryland's Basemap in the Cloud

Submitted February 28, 2014

Prepared for

NSDI Cooperative Agreements Program
Category 7: Geospatial Platform Cloud Service Testbed

Prepared by the Center for GIS at Towson University

7801 Building Suite 260

Towson, MD 21204

v 410-704-3887

Table of Contents

Executive Summary

In a partnership with the Maryland State Geographic Information Office (GIO), the Center for GIS (CGIS) at Towson University (TU) lead a project to assess the value of making Maryland's GIS data available in Amazon's Cloud-based web services. MD iMap is Maryland's public-facing enterprise GIS infrastructure for providing mapping applications, products, and services online that assist citizens and government employees with managing and presenting data linked to a location. This assessment compares the load handling capability of the current MD iMap hosting environment with an Amazon Elastic Cloud (EC2) hosting environment built around the m1.large server instance size. This instance size was recommended for MD iMap by Amazon technical staff in a proposal unrelated to this grant. The intent of this test was to quantitatively compare the Amazon Virtual core EC2 Compute Units with CPU capacity on MD iMap servers to clarify this important architecting parameter and provide validation for a server instance size that would be suitable as a basis for hosting MD iMap services in EC2. In addition, server scaling capabilities was tested for performance and a cost of ownership analysis. The tests determined that an m1.large EC2 instance was not robust enough to meet the peak demand that MD iMap occasionally responds to. For a variety of methodological reasons the differences between current MD iMap server CPU capacity verses EC2 Compute Units of the m1.large instance was not directly quantifiable. Instead, peak requests per second were used to identify the maximum number of virtual users the systems were able to handle before performance started to degrade. While an estimated cost of ownership was completed, showing MD iMap's traditionally hosted infrastructure having a lower cost over a six year period. If MD iMap's server life cycle was three years instead of 6, then Amazon infrastructure services would have been more cost effective. In either case, there are many ways in which Amazons cost could be brought down by fine tuning testing that was beyond the scope of this project. The most savings in Amazons costs would be wrought by rebuilding GIS services, applications and architecture completely around Amazons hosting services.

Introduction

The Center for GIS at Towson University lead a project to assess the value of making Maryland's latest basemap data (6-inch imagery and Gazetteer) available in Amazon's Cloud-based web services. MD iMap is Maryland's public facing enterprise GIS infrastructure for providing mapping applications, products, and services online that assist citizens and government employees with managing and presenting data linked to a location. The state's 6-inch imagery is maintained by Maryland Department of Natural Resources and is hosted on the MD iMap infrastructure at Towson University. Most of the time MD iMap server resources experience light to moderate utilization. During events such as major storms that trigger a Maryland state of emergency declaration, or during Elections night for example, utilization can jump from tens to hundreds to thousands of users per hour e.g. after Hurricane Irene the public safety mapping application Osprey saw 251,000 unique visits in September 2011.

The purpose of this grant project was to compare the aging MD iMap physical server infrastructure with hosting options available in Amazon EC2 Cloud. Of interest was the overall performance of Amazons EC2 virtual machines and more specifically how to assess the capacity of CPU equivalencies published by Amazon (Virtual core and EC2 compute units) and compare performance with the current MD iMap infrastructure hardware. It was envisioned that the performance would be tested against real world use with the development of mobile applications, which would use GIS basemap data services hosted in Amazon and basic load testing. The outcomes anticipated were deployment costs, performance characteristics, and the estimated cost of ownership for both environments at the current level of utilization in MD iMap.

In the interim between writing the proposed scope of work and launching the project, several unanticipated impediments to progress have occurred, namely, changes in project partners' plans with regards to mobile application development, changes in the planned upgrade

for the MD iMap GIS server software version from 10.0 to 10.1, and delays in testing due to non-availability of the staging servers on MD iMap system. These changes not only delayed progress, but also turned into significant impediments to implementing the grant project as originally envisioned.

Midway through the project a scope change was approved to use a different approach that kept the core project goals intact and assessed the cost and performance equivalencies of the Amazon EC2 environment with the current MD iMap hosting infrastructure. Performance testing would now be done with a web-base load testing platform LoadStorm.com, allowing for the scripting of a typical user's interaction with a website. The intent was to use this simulated GIS web application user (virtual user) script plan to provide a standardized test to load both the MD iMap servers and the Amazon servers which would be setup with the same GIS web services. CPU load, error rate, requests per second would be used to compare the systems. As an outcome of a proposal unrelated to this grant, an Amazon technical team reviewed MD iMap's server infrastructure and recommended using Amazon's m1.large Server instance for the GIS application server to reside on. Since the Amazon's m1.large instance consists of four EC2 compute units (two virtual cores with two EC2 units each – based upon a 2007 1.7 GHz Xeon processor); and MD iMap GIS servers have dual Intel Xeon Quad Core Processors (E5450 @ 3.0 GHz), the intent was to derive a CPU based performance factor differential between the two environments. This factor would be important in understanding the amount of load an m1.large instance could handle in relation to an MD iMap server. This would be an important comparison point for performance and cost of ownership analysis. A second important factor to consider when comparing the two server environments would be to assess EC2 server scaling capabilities. From a cost perspective the desire is to have only enough server capacity to meet the defined

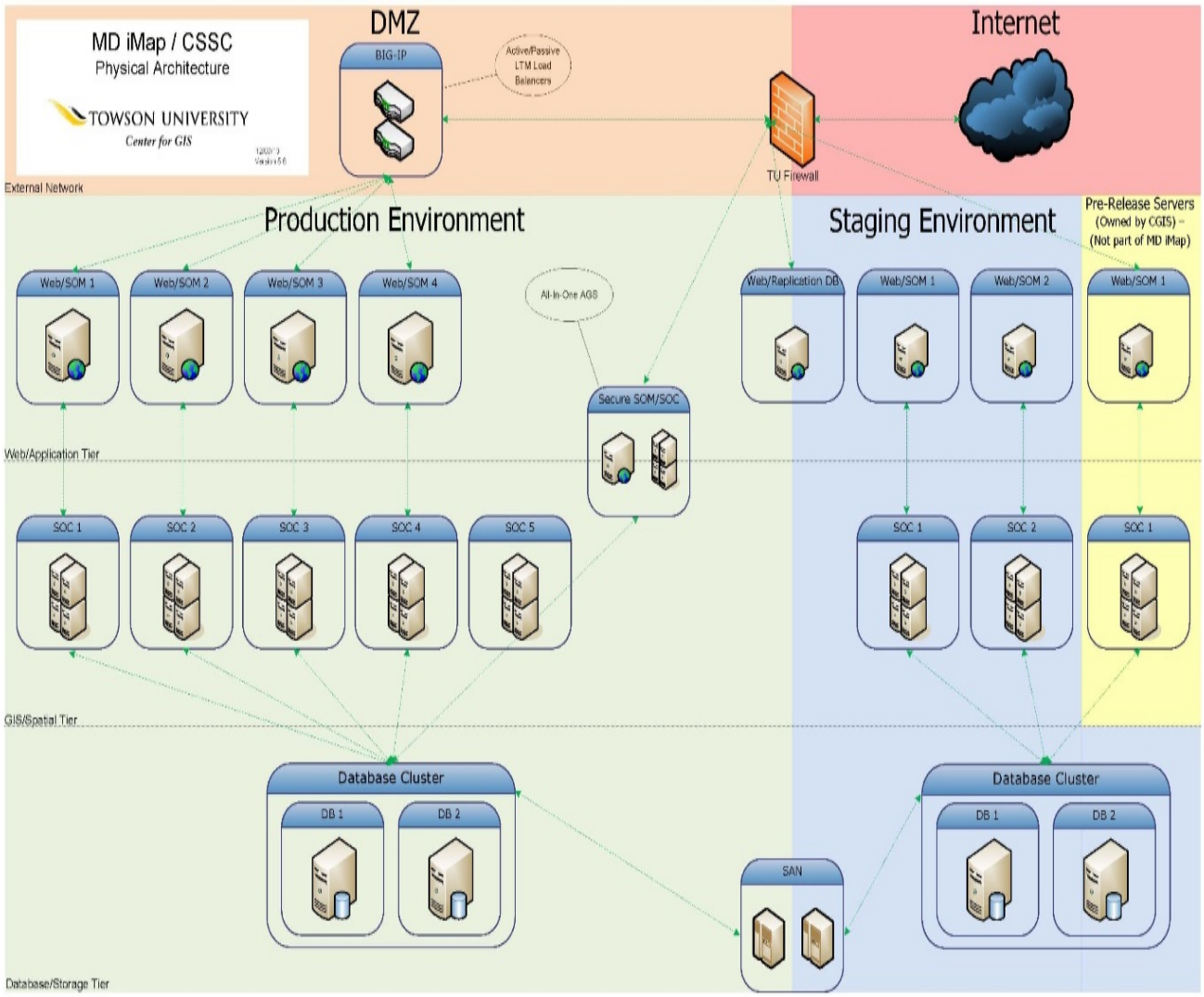
baseline in an Amazon EC2 hosting environment architecture. As utilization rises it requires scaling the environment up by automatically adding more servers to meet that elevated demand and scaling back down as the utilization drops. The tests designed for this grant project were intended to compare capacity and scaling which were setup to mimic the steep loading patterns that MD iMap has experienced in the past. This was required to determine server instance size and scaling parameters needed. The next section will elaborate on the environments tested.

Hosting and Testing Infrastructures

MD iMap Hosting Infrastructure

The MD iMap hosting environment was architected in early 2008 to serve Maryland's framework GIS datasets in the form of desktop services, web services and mapping applications for public and government use. The system was designed to be highly available, redundant and able to handle heavy utilization. IBM was chosen as the prime technology vendor, utilizing their blade center server and fiber channel Storage Area Network (SAN) technology. Load traffic management is handled by two F5 Big-IP load balancers. The system went into production in January 2009. As can be seen from Figure 1 below, the infrastructure consists logically of a production environment, staging environment and a pre-release environment.

Figure 1 - MD iMap Architecture



MD iMap Server hardware consisted of a three-tiered architecture, with one web server, one ArcGIS processing server, and a cluster database. The web tier served web pages through Microsoft IIS 7.0 web server with the ArcGIS Server Object Monitor (SOM) installed. Each ArcGIS Server is connected to the application tier in a one to one relationship (no SOM request balancing) with an "unlimited" capacity. The F5 load balancers monitor and manage all traffic to the production GIS server tiers. Finally, the database cluster used ArcSDE with Microsoft SQL Server 2008. At the time of testing ArcGIS server was at version 10.0.

The web server has the following specifications:

- Dual Intel Xeon Quad Core Processors (E5420 @ 2.5 GHz)
- 8 GB of Ram
- 80 GB System Drive

The application server had the following specifications:

- Dual Intel Xeon Quad Core Processors (E5450 @ 3.0 GHz)
- 8 GB of Ram
- 80 GB System Drive

Each node in the database cluster had the following specifications:

- Dual Intel Xeon Quad Core Processors (E7320 @ 2.13 GHz)
- 16 GB of Ram
- 80 GB System Drive

MD iMap Test Constraints

Testing on MD iMap entailed taking WEB 1 and SOC 1 off-line from the staging environment and configuring them for testing. A testing database was loaded with data and web services were setup. For testing purposes each service was limited to spawning 10 active instances (AGS Server default is two). Scheduling testing time on the MD iMap infrastructure was a challenge. The testing needed to happen between update cycles and in a time of less high volume activity related to emergency management. The testing period had a three week window. Week one was used to reconfigure the servers and make the servers accessible to the internet. Week two (and three additional days) was used for testing. Finally less than one week was dedicated to restore the servers to their original state in the staging environment. This condensed schedule for testing on MD iMap created limitations in the comparison of the two systems. All

tests and metrics observed during the testing time frame were final and there would be no opportunity to make changes.

Amazon Cloud Based Hosting Infrastructure

Independent of this grant Maryland's GIO requested that CGIS work with Amazon to architect and estimate costs required to deploy an equivalent hosting environment to MD iMap in Amazons Cloud environment. Amazon was provided with MD iMap's technical architecture, site utilization and other usage statistics. The recommended architecture included using an m1.large instance size for the GIS servers in production, which is what was subsequently used for testing purposes in this study. Figure 2 represents the production Amazon architecture; Figure 3 represents the staging architecture and Figure 4 represents the development environment that could also be used for prerelease.

The Amazon environment consisted of a collapsed web/application tier and a database server. The combined web/application server used Microsoft IIS as the frontend web server with both the ArcGIS SOM and Server Object Container (SOC). The AMI used for this server also included a version of Microsoft SQL Server Express. However, the services were disabled on the web/application server. A database AMI was used to simulate the same distributed setup with the on-premises hardware. The server had Microsoft SQL Server Express and ArcSDE.

Figure 2 - Amazon production environment

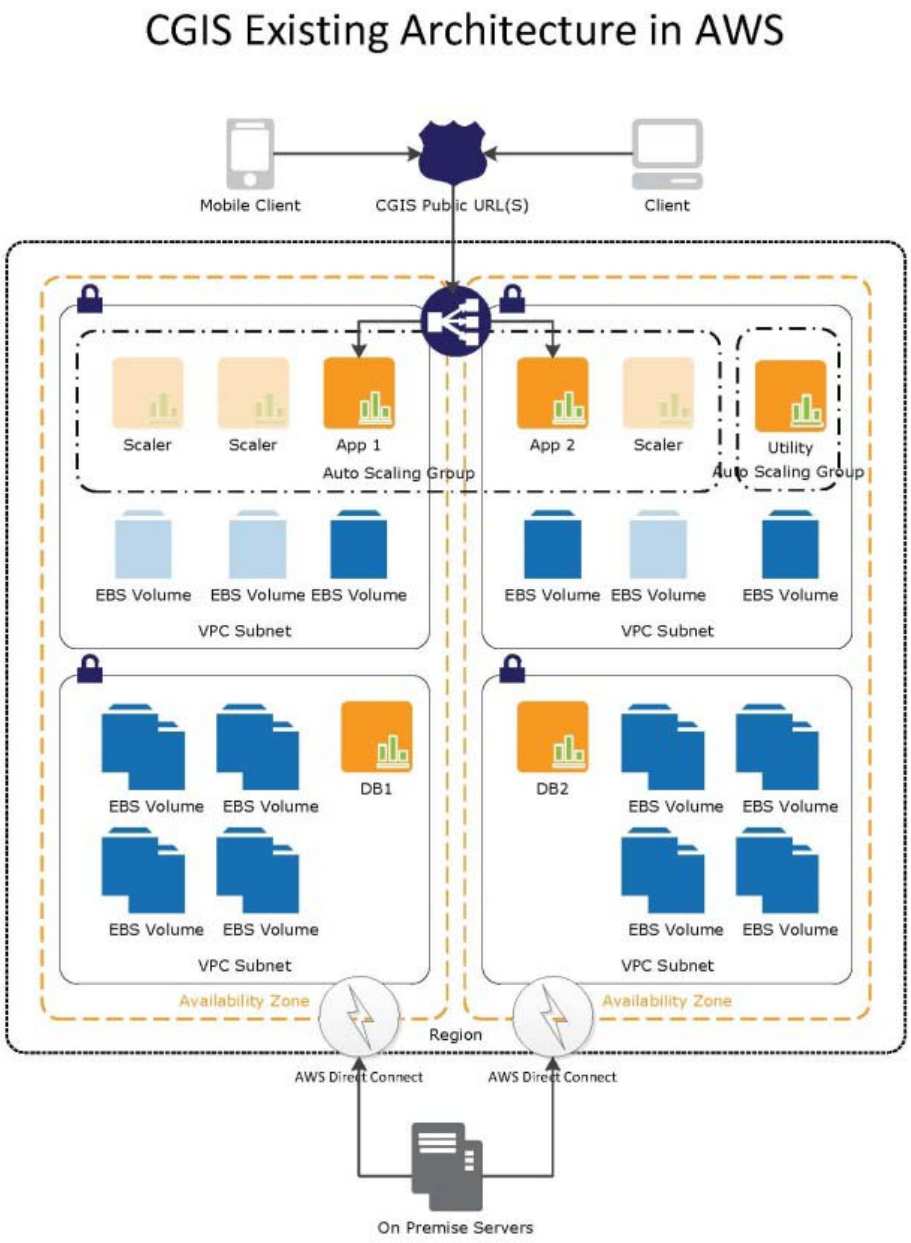


Figure 3 - Amazon staging environment

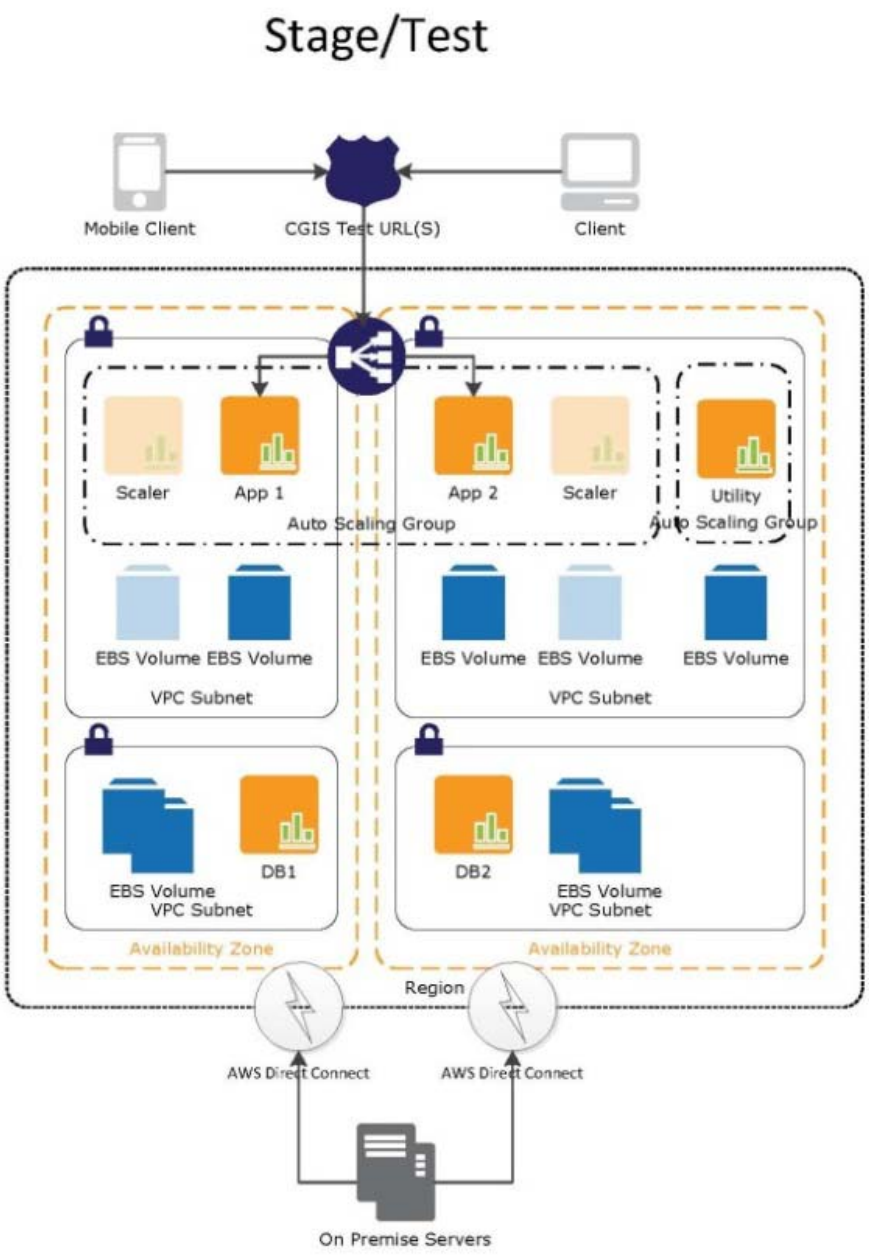
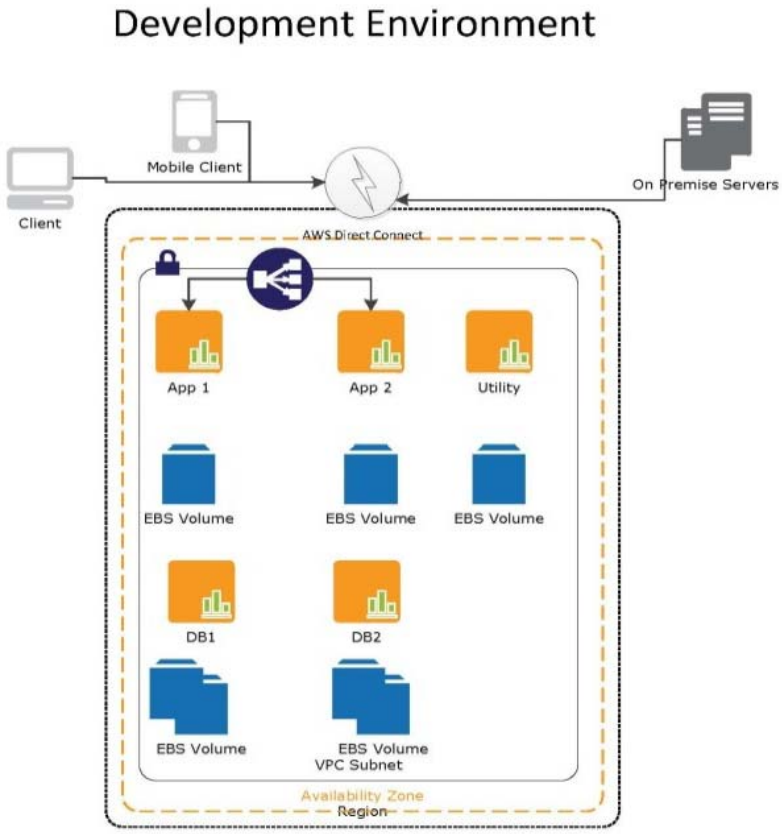


Figure 4 - Amazon development\Pre-release environment



The Amazon web/application server had the following specifications:

- Amazon m1.Large Instance
- Dual Core Processor, 4 EC Units
- 7.5 GB of Ram
- 30 GB System Drive and 100 GB Data Drive

The Amazon database server had the following specifications:

- Amazon m2.2xlarge Instance
- Quad Core Processor, 13 EC Units
- 34.2 GB of Ram
- 30 GB System Drive and 30 GB Data Drive

Amazon Test Constraints

While time constrains were not an issue in working with the Amazon testing environment, there were however several other important constraints that made a direct comparison of CPU and EC2 compute units unattainable. MD iMap's ArcGIS Server configuration was implemented around Esri's high availability enterprise architecture, which splits request processing over three tiers (web processing, application processing and database processing). The Esri ArcGIS Server AMI designed for use in Amazons EC2 environment combines web processing and applications processing together on one server. Setting up the Amazon environment to mimic the MD iMap infrastructure was considered, but it was determined that it would invalidate the desired real world comparison outcome and subsequently estimate of cost evaluation. In addition, logging CPU utilization in EC2 was an issue. In MD iMap, which is not virtualized, Microsoft Performance Mentoring utility (perfmon.exe) was utilized on the servers being tested to capture various metrics of performance, including CPU utilization. Since Amazons server instances are virtualized only a fraction of host servers CPU attention is received. This is presumably the equivalent of the number of virtual cores and EC2 compute units in the size range chosen. While with most hypervisors, the virtualization controller software, has a negligible performance impact; the methods for scheduling resources can impact overall performance and the ability to handle requests. Perfmon.exe cannot be used on an Amazon server because it will not differentiate between hypervisor and guest (VM) requests. This means that it is necessary to monitor resources through the hypervisor's management console for any virtualized server. Amazon provides management tools to see a graph of CPU load and other metrics in real time for their VMs. Unfortunately it is not apparent how to log or export CPU utilization data through the management console, only view it in real time. As with many advanced capabilities in EC2 there may be a way to script a process to log this data. With

multiple issues making a direct CPU load comparison difficult to quantify, it was decided to focus on the number of simulated virtual users supported by the systems being tested as a point of comparison and to determine if the m1.large instance was adequate for the full range of loading MD iMap has had to contend with.

Load Storm Testing Infrastructure

Several options for load testing were evaluated and considered. A top consideration was to choose a package or service that was capable of providing realistic loading patterns that were representative of a user's interacting with a generic web based GIS application. There was an obvious concern with running a load testing application on the same network as the servers being tested. This was especially a concern when comparing results between two systems where one would transvers the internet and the other one would not. LoadStorm (Loadstorm.com) a cloud based load testing tool was chosen because it is an off network service and is built around being able to easily create realistic testing scenarios through a simple user interface. Open source desktop load testing software JMeter was used to do some initial testing and to validate the results seen in LoadStorm, especially during the scaling testing.

The LoadStorm testing plan was derived by monitoring http bowser requests while interacting with MD iMap GIS web applications. The interaction with the web applications resulted in representational state transfer (REST) calls to ArcGIS server for map data, these REST call URLs were captured and analyzed. The server calls were categorized into five applications interaction scenarios that were weight based upon how a typical user works with a generic GIS web application. As can be seen in Figure 5, scenarios included REST calls for an application load or initialization, application interaction (Map Pan, zoom, etc.), feature identify, query request and legend request. Steps represent REST server calls that are iterated though each

time a Loadstorm virtual user is randomly assigned a scenario based upon the associated weighting. Each step has a minimum wait time of 10 seconds before initiating the next step. The minimum wait time was used for all scenarios.

Figure 5 - LoadStorm Test Plan

Scenario	Weighting	Steps	Average Duration	Estimated transfer
Scenario 1 - Application Load	2 (2.0%)	33		0 KB
Scenario 2 - Application Interaction	60 (60.0%)	12		0 KB
Scenario 3 - Identify	20 (20.0%)	3		0 KB
Scenario 4 - Query	10 (10.0%)	1		0 KB
Scenario 5 - Legend	8 (8.0%)	5		0 KB

Five map services were setup with raster and vector data down loaded from MD iMap and setup specifically for this test. Each map service was configured to be able to spawn 10 active instances from the default two instance setting. The full detail of the each scenario step is enumerated in Appendix A.

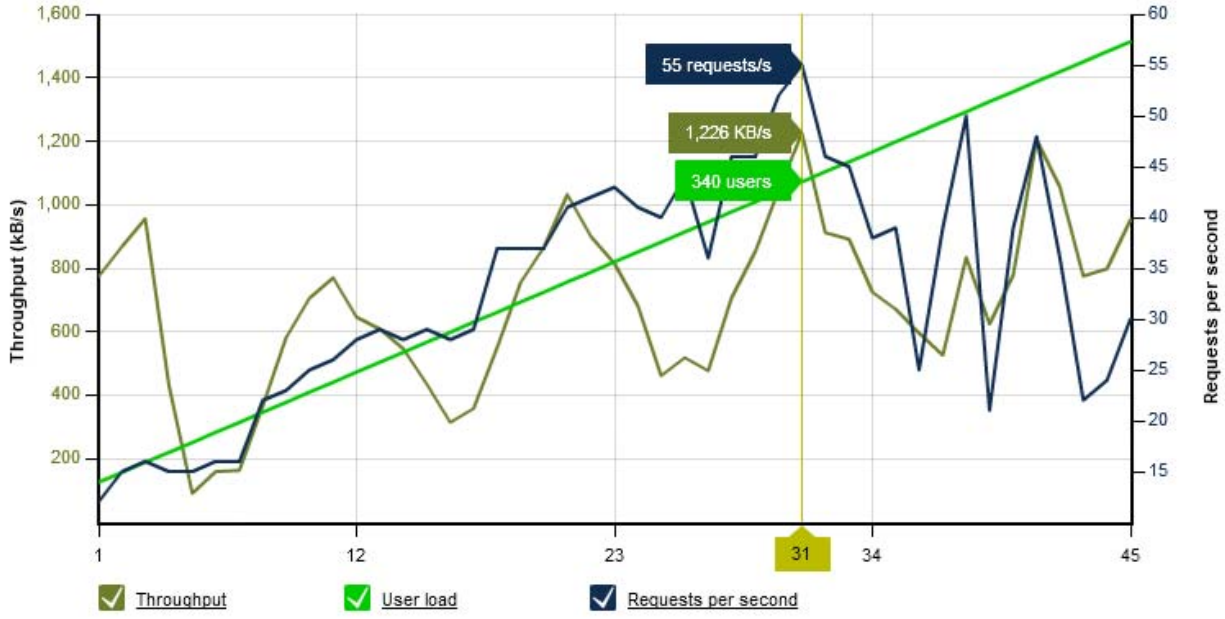
Performance Testing

Setting up a test case that adequately loaded the MD iMap servers, but that did not over whelm the Amazon m1.large instance proved to be more difficult than anticipated. Ideally a test would be setup that could have seen a peak and then drop off in requests handled per second, which is correlated to extreme CPU utilization, on both systems.

An initial heavy load, representative of peak demand seen on the MD iMap server farm was chosen as a challenging starting point. This initial test performed on the MD iMap test instance resulted in using a linear step-up method of increasing load by 10 users every minute. The base load was 40 users. This allowed for a total of 480 virtual users to be load testing the

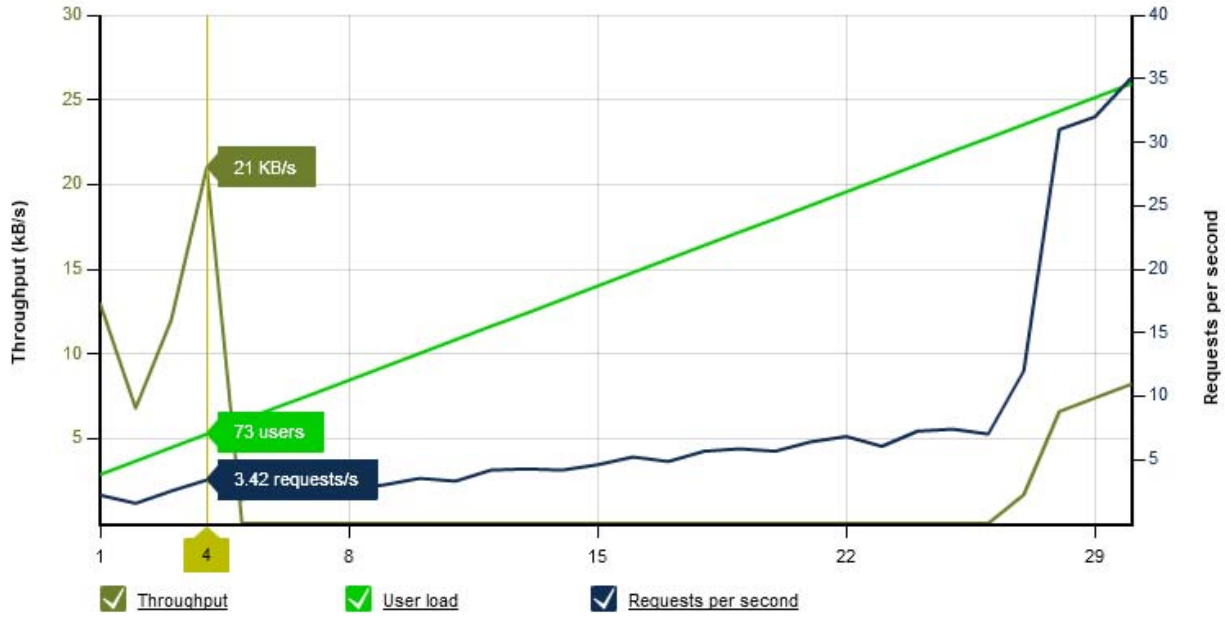
virtual application by the end of the test. Under this loading setup the MD iMap servers were able to handle the load well up to 340 virtual users and 55 requests per second at 31 minutes into the test, see Figure 6 below. After this point the requests per second served dropped and error rate rose along with the response time.

Figure 6 - Initial MD iMap Heavy load test



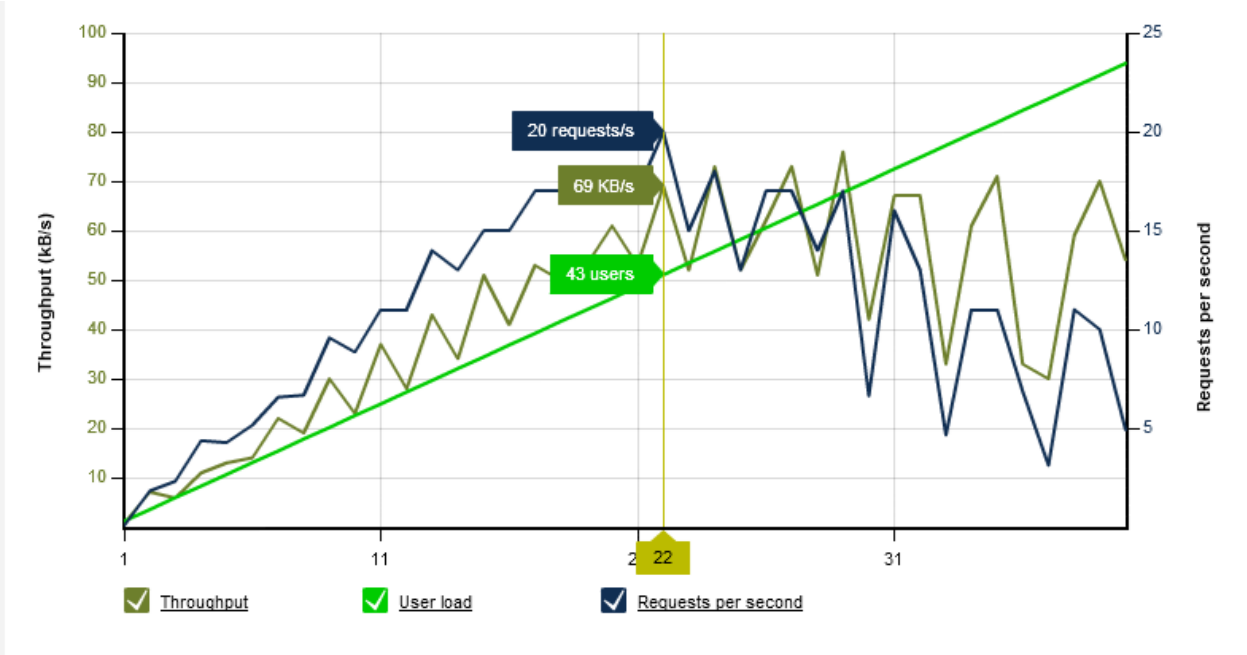
While this test was able to stress the on-premises hardware, it simply overloaded the Amazon virtual machines. Within in the first five minutes, error rates rose to 100 percent. Under this test plan the Amazon m1.large server hit its peak user load at 73 virtual users and 3.42 requests per second at minute four of the test, see Figure 7 below. The load test was terminated early because the server was not responsive. After investigating, it was found that Microsoft IIS could not handle the amount of concurrent requests because of an almost immediate CPU spike to 100 percent. The wait time for a response of web requests exceeded 35 seconds, which generated an HTTP error of 408. The IIS requests became queued, and there were not enough CPU resources to properly handle the requests.

Figure 7 - Initial Amazon Heavy load test (terminated early due to unresponsiveness)



After analyzing this initial round of testing and with less than a week left of testing time available on the MD iMap server, it was necessary to modify the test case to reduce the load generated in order to have a more graduated CPU utilization spike. Various load profiles were tested on the Amazon instances, with the intent of finding a challenging but passing profile. A moderate test plan was created consisting of a linear step of two virtual users every minute, beginning with one user. This allowed for 79 concurrent users over 40 minutes to test the simulated application by the end of the test. Under this loading setup the Amazon m1.large server was able to handle the load well up to 43 virtual users and 21 requests per second at 22 minutes into the test, see Figure 8 below. With the more gradual load pattern the requests per second served peaked at 21 and then declined to plateau where 17 requests per second served was maintained for another seven minutes. At the end of this plateau, 57 virtual users were present with the system handling 17 requests per second at 29 minutes. After this point the requests per second served dropped and error rate rose along with response time.

Figure 8 - Amazon moderate load test



When the more moderate test plan was run the MD iMap Hardware it handled the load without error or having a peak in the number of request per second served. Considering how the MD iMap servers handled the previous more robust test plan, the results for this moderate plan were not a surprise.

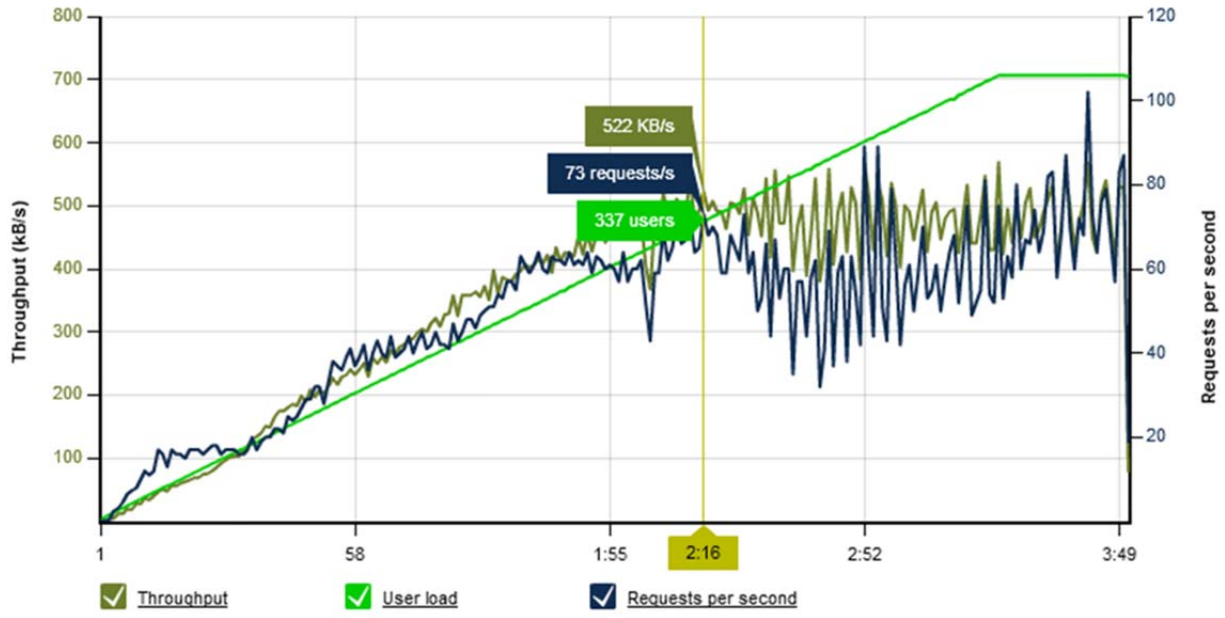
The test was all repeated several times for validity, in part with the help of Towson University computer science student's involvement as part of their senior year Capstone project. Ideally the tests would have been performed on the same test plan, but because of the significantly difference in compute power between the systems being evaluated that was not possible. While the loading pattern was different between the two test plans, the scenarios tested were exactly the same. Looking at the peak number of virtual users between the heavy test plan for MD iMap, some 340 and the peak number of virtual users supported on Amazons m1.large instance when testing with the moderate plan, 57, the rough order of difference is a 6 to 1 ratio.

For the majority of the time the actual load on the MD iMap server farm would equate more to the moderate test plan than the heavy test plan. One of chief architectural principles with Amazon's Cloud hosting environment is the idea that a system needs to scale to demand, only running what is needed for redundancy or availability across datacenters. The next section will elaborate on the tested done to assess scaling of m1.large instance to meet load demand.

Scaling Amazon EC2 on-demand Instances for GIS applications

The LoadStorm moderate test plan used for the Amazon cluster auto scaling differed from the single instance test primarily in duration, to allow for the time needed to build the load to trigger scaling at approximately the same number of virtual users a minute. The moderate cluster test plan started with one virtual user and added two to three virtual users every minute up to 500 virtual users over a period of 240 minutes. A load balancer split the requests between two m1.large instances, with a maximum of six m1.large instances. The cluster was configured to add one instance when the average CPU utilization across the cluster was increased above 50 percent for a period of one minute; this event had a 300 second cooldown period in which another instance has to wait before being added or removed. Under this loading plan the Amazon m1.large cluster was able to scale and handle the load well up to 337 virtual users and 73 requests per second at two hours and 16 minutes into the test, see Figure 9 below. After this point the requests per second served dropped, recovered a bit and then plateaued, error rate rose and climbed steadily after two hours and 34 minutes along with response time.

Figure 9 - Amazon cluster moderate load test



The results of the successful Amazon cluster scaling test are consistent with the virtual user load capacity seen when running the moderate Amazon single instance test, with perhaps a slight capacity gain by the load being distributed, giving each node more time to process before the next request comes in. If each node can handle 57 virtual users in the moderate test plan, it is expected that if scaled properly the cluster would be able to handle 342 virtual users. Interestingly, this closely matched the number of virtual user that the MD iMap server was able to handle, but with a much more gradual loading curve than in the heavy load plan. The heavy load plan is representative of peak utilization curves seen in the past on MD iMap infrastructure.

When more users per minute were added to the Amazon cluster scaling properly became an issue under the moderate loading plan. The time required for a single instance to join the cluster meant the cluster did not recover from the onset of load as more nodes joined the load balancer. Several contributing factors were identified. The idle pool size, cluster resizing rules,

stand up delay, node size, and load balancer interact to create a difficult testing environment. Node size is likely the most important parameter. The capacity of the virtual hardware matters; the larger the Amazon instance, the more load individual nodes can accommodate. Choices here can mitigate difficulties in other areas; a larger node has more headroom (capacity) to wait for another instance to join the cluster, requires fewer base nodes, can wait for higher resource utilization thresholds before scaling, and can scale in smaller increments. These advantages come with a higher hourly running cost.

Of similar importance is the minimum number of nodes on standby. An 80 percent load on a pool of five nodes represents more headroom than an 80 percent load on a single node. This creates a nonlinear behavior; as the number of nodes in the cluster increases, the amount of load that cluster can take on before requiring more nodes also increases. Assuming linear scaling, an increase on a single node of 5 percent CPU utilization is the equivalent of a 1 percent increase on a five node cluster. At the time of writing, it was understood that Amazon provides no parameter that takes into account the number of nodes in the cluster when determining load. This creates a disparity in rule optimization; a cluster can be configured to respond economically to a low base load and scale rapidly, or a large base load and scale slowly. Any given configuration will be cost ineffective if operated outside its design parameters. An ideal solution to this problem would be an implementation of well-understood concepts in control loop theory, rather than static rules that cannot respond to dynamic loads. By using control theory, a scaling policy based on a Proportional-Integral-Derivative controller (PID) loop could accurately choose the number of nodes to bring online and at what thresholds to do so, without user intervention or selection of values beyond tuning the initial gain parameters. Such a mechanism would improve the cost effectiveness of clusters to operate across a varying range of load and use cases.

Related to the minimum pool size are the resizing rules. The criterion by which a cluster gains and loses nodes affects the cluster's performance with respect to rapidly changing load. Some of the factors include: metric, such as CPU or memory utilization; threshold, the ranges of metric that trigger the action; change, the number of nodes to add or remove from the cluster; and cool down, the period of time that must pass before the same rule can trigger another change. As with base load, accurately setting these values requires accurate historical records of the system. Understanding how load fluctuates over time determines how quickly the system must respond. Due to Amazon's load balancing architecture, even a single node becoming unresponsive can be unacceptable for many use cases. It would be advisable to design scaling rules conservatively in these instances.

A factor that the system designer has no control over is the time a node takes to join the cluster. This is a varying amount that was seen to be in the range of three and seven minutes, and must be taken into account. In smaller deployments with low minimum cluster sizes, this can cause a sharp increase in load to make the cluster unresponsive even with conservative scaling rules. The recovery from this condition can be hindered by the load balancer and DNS caching behavior.

The last major consideration is the load balancer. At the time of testing, Amazon's load balancer was a simple DNS round robin. This creates a problem; any node that is loaded to the point of being unusable but not yet unresponsive will remain in the balancer pool. Further, clients can become locked to a specific node through DNS caching. This creates the scaling issue alluded to earlier; if the cluster becomes unresponsive, the users connected to the unresponsive nodes will not load balance over to the new nodes. Using conservative rules that favor removing nodes from the pool can mitigate this problem. For example, rather than checking a simple

service, check a more representative and processor intensive resource. It is often better to have no nodes in the balancer pool and reject new connections for a short period than many overloaded nodes still presented as available. A node leaving the pool is not equivalent to the node being down, as it merely stops new incoming connections; the node is still reachable by IP address. Applications that are sensitive to this problem may be better served through software based load balancers, which can themselves be hosted as a scaling cluster if desired. Amazon recently announced major improvements in the load balancer offered, which came out after the period of testing for this project.

Estimated Cost of Ownership Comparison

This section provides a cost comparison between actual MD iMap infrastructures costs with an estimate of cost to host a similar infrastructure in EC2 as outline previously in the *Hosting and Testing Infrastructures section* of this document. Each solution is broken out in two general categories of infrastructure/software cost and maintenance cost as seen in Figure 11. The infrastructure includes all the required servers, storage, hardware, and network bandwidth. The software includes all required server software and licenses. The maintenance includes monthly and annual upgrades, service packs, patching of serves, and upgrading AGS. Costs do not include Esri software licensing. The State of Maryland has an enterprise license agreement with Esri and while the negotiated price is based upon a complex calculation of all licensing in use every three years, it was decided not to try quantifying this cost in the estimate provided below. Depending upon on how Esri charges for licensing with scaled server instances, this may be a cost saving in Amazons EC2 infrastructure with less 24/7 servers needed. The following section covers a detailed look at what items are considered in the cost comparison.

MD iMap Infrastructure

MD iMAP infrastructure reflects the actual costs for purchasing, hosting and the basic maintenance of the system. The MD iMAP infrastructure was initially designed for a five year period, which was the expected system lifecycle. Where possible all warrantees were purchased for the full five year period and are reflected in the first year's infrastructure/software costs. Several items like the load balancer warrantee's were renewed annually and are reflected in the subsequent infrastructure/software costs. The infrastructure includes all the required servers, SAN storage and hardware required to build and stand up the system. The software includes all required server software and licenses. Year one represent the initial setup costs. Subsequent years are reduced by the initial infrastructure/software costs. The maintenance includes monthly and annual upgrades, service packs, patching of serves, upgrading AGS and network bandwidth. Maintenance is based on a 12 month period totaled annually by year. Maintenance cost decreased in the last three years due to found efficiencies. One year has been added to the intended lifecycle of the current MD iMap infrastructure as part of a transition of application and services to a new system in a state owned facility and is represented in these costs.

Amazon Infrastructure

Independent of this grant Maryland's GIO requested that CGIS work with Amazon to architect and estimate costs required to deploy an equivalent hosting environment to MD iMap in Amazons Cloud environment. Amazon was provided with MD iMap's technical architecture, site utilization and other usage statistics. The recommended architecture included using an m1.large instance size for the GIS servers in production, which is what was subsequently used for testing as outlined in this paper. As an outcome of the preliminary results from this grant work Maryland's GIO asked CGIS to revisit the architecture provided by Amazon and work with them to provide an estimate of cost that included more robust reserved instance, the inclusion of

webservers and utilizing three service pools in a 10.1 deployment. The cost estimate developed by CGIS and reviewed by Amazon includes upgrades to the following instance types: m3.xlarge for the webservers/scalers, m3.2xlarge for the GIS application servers/scalers and databases. For more detail on instance size and cost see the Amazon cost calculate estimate in Appendix B. The cost estimates are based on a three year cost solution which includes 3-yr Heavy RIs (reserved instances) for Production and 3-yr Light RIs (reserved instances) for Staging, which provides a 27 percent savings compared to an using annual reserved instance. In order to make a six year cost comparison with MD iMap three additional years were added for the cost of the environment using the same rate of the original quote. With the reserved instance there is a onetime fee that can be seen as an increase in costs for year one and year three.

The MD iMap cost estimate includes basic maintenance. This support cost includes monthly and annual upgrades, service packs, patching of servers, and upgrading to new AGS AMI's. Maintenance cost is based upon the number of specifically tailor AMI's being supported.

It is important to note that Amazon considered this cost estimate to be a high watermark cost estimate supporting the possible variability with EC2 instance type sizes for all uses and for variability for storage practices and EBS (elastic load balances). Costs are from May 2013.

Figure 11--Cost Comparison

MD iMAP Solution

Year	1	2	3	4	5	6
Infrastructure/Software	600,317	49,878	49,662	49,420	55,460	60,249
Maintenance	85,700	85,700	85,700	76,724	76,724	76,724
Total Cost	686,017	135,578	135,362	126,144	132,184	136,973

Cost for 3 Years	$	956,957	
Total Cost for 6 Years		$	1,352,258.00

Amazon Solution

Year	1	2	3	4	5	6
Infrastructure/Software	320,095	232,152	232,152	320,095	232,152	232,152
Maintenance	35,994	35,994	35,994	32,224	32,224	32,229
Total Cost	356,089	268,146	268,146	352,319	264,376	264,381

Cost for 3 Years	$	892,381	
Total Cost for 6 Years		$	1,773,457.28

Cost Difference (savings)	3 Year Cost	6 Year Cost
MD iMap	---	(421,199.28)
Amazon EC2	(64,576.00)	---

Cost of Ownership Analysis

A comparison of something of a known quantity, like MD iMap actual costs, with an educated estimate of costs for Amazon is at best an approximation comparison. There are many variables that go into the way in which Amazon charges for EC2 services. Optimizing the use of

these services requires extensive planning and testing. The testing completed during this grant project is the first step in fine tuning the architecture and cost estimate, resulting in the EC2 cost calculator estimate in Appendix B. The number of physical and virtual servers represented in the cost comparison are the same, 21. With MD these are all physical servers available 24/7, the Amazon EC2 cost estimate includes 13 reserve instances available 24/7, four reserve instances for 40 hours a week and four are scaler instances for 10 hours a month.

Feedback from Amazon was that cost estimate in Appendix B was a good “high-water mark” that could be used for budgetary purposes and felt confident with optimization and diligent monitoring the cost of ownership could be reduced. One way that this could be achieved is through splitting approximately 269 map services hosted on MD iMap currently, into three pools (new to ArcGIS server at 10.1). Smaller GIS application server instance sizes could potentially be used for a service pool that is less used or requires less resources, like serving up map caches. Another big potential savings, although not possible across all current access types being used on MD iMap, would be to use S3 storage for cached data, over more expensive EBS storage. The last item to consider again in this cost comparison is the fact that the MD iMap physical infrastructure is being compared to Amazons virtual environment. If MD iMap was virtualized with VMware, this would increase its infrastructure cost estimate by approximately $75,070.00 over six years.

The cost of ownership for six years on MD iMap’s physical infrastructure, as estimated in this study, represents a significant savings over Amazons EC2 hosting designed to meet similar peak usage loading. Even when the VMware costs are added to MD iMap actual costs to make the systems more comparable, the difference in cost is approximately $346,000.00. A significant effort of optimization in an Amazons EC2 implementation would be required to cover that gap.

Conversely if the systems life cycle was three years instead of six, the cost comparison would be in Amazons favor. Again, taking into consideration the VMware cost to a future MD iMap implementation, this difference would be roughly $140,000.00. Even without, further optimization this is a substantial savings over owning on premise physical infrastructure. This adds infrastructure lifecycle as another important element to the list of items to be considered when looking at option for hosting framework datasets in a cloud environment.

Conclusion

The comparison between on-premise hardware and cloud-based infrastructure is a hot button topic for technical analysts and managers alike. The difficulty for technical staff remains in the fluid nature of cloud computing. On-premise equipment and specifications will always be a known quantity for analysis; however, Amazon's highly heterogeneous environment causes difficulty in comparison, particularly for those who are new to the cloud hosting paradigm. When first scoping out an Amazon instance, consumers are provided with key specifications, including number of virtual CPUs, amount of memory, and amount of storage. While these specifications seem simple enough, the virtual CPU quantity, in particular, is vague. While not able to derive a CPU based factor difference between the m1.large instance tested and MD iMap's Intel Xeon Quad Core Processors (E5450 @ 3.0 GHz) processors, with some difficulty a method for determining the number of virtual users those systems could support in a simulated real world test was achieved. Knowing that one of MD iMap's application servers could handle the load of approximately six m1.large instances was instructive and helpful in determining the need for a more robust instance type to meet peak loading, as seen in the scaling test. Finally the cost of ownership analysis, while also difficult to do when comparing physical and virtual environments, yielded interesting results based upon the intended lifecycle of the infrastructure.

Appendix A - LoadStorm Plan Steps Detail

Scenario 1 - Application Load

#	Step
1	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Boundaries/MD.State.CongressionalDistricts.2011_WM/MapServer?f=json
2	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ClimatologyMeteorologyAtmosphere/MD.State.StormSurgeAreas_WM/MapServer?f=json
3	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer?f=json
4	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap_Gazetteer_WM/MapServer?f=json
5	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/PlanningCadastre/MD.State.SustainableCommunities_WM/MapServer?f=json
6	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Boundaries/MD.State.CongressionalDistricts.2011_WM/MapServer/layers?f=json
7	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ClimatologyMeteorologyAtmosphere/MD.State.StormSurgeAreas_WM/MapServer/layers?f=json
8	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/layers?f=json
9	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/PlanningCadastre/MD.State.SustainableCommunities_WM/MapServer/layers?f=json
10	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Boundaries/MD.State.CongressionalDistricts.2011_WM/MapServer/0?f=json
11	Open http://ec2-54-204-168-159.compute-

#	**Step**
	1.amazonaws.com/ArcGIS/rest/services/ClimatologyMeteorologyAtmosphere/MD.State.StormSurgeAreas_WM/MapServer/0?f=json
12	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ClimatologyMeteorologyAtmosphere/MD.State.StormSurgeAreas_WM/MapServer/1?f=json
13	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ClimatologyMeteorologyAtmosphere/MD.State.StormSurgeAreas_WM/MapServer/2?f=json
14	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ClimatologyMeteorologyAtmosphere/MD.State.StormSurgeAreas_WM/MapServer/3?f=json
15	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/0?f=json
16	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/1?f=json
17	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/2?f=json
18	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/3?f=json
19	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/4?f=json
20	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/5?f=json
21	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/6?f=json

#	**Step**
22	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/7?f=json
23	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/8?f=json
24	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/9?f=json
25	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/10?f=json
26	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/11?f=json
27	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/12?f=json
28	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/13?f=json
29	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/14?f=json
30	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/15?f=json
31	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/16?f=json
32	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/17?f=json

#	**Step**
33	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/PlanningCadastre/MD.State.SustainableCommunities_WM/MapServer/0?f=json

Scenario 2 - Application Interaction

#	**Step**
1	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Boundaries/MD.State.CongressionalDistricts.2011_WM/MapServer/export?dpi=96&transparent=true&format=png8&bbox=-8684977.236046717%2C4715019.35603861%2C-8393659.099411929%2C4785477.696061908&bboxSR=102100&imageSR=102100&size=1505%2C364&f=image
2	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ClimatologyMeteorologyAtmosphere/MD.State.StormSurgeAreas_WM/MapServer//export?dpi=96&transparent=true&format=png8&bbox=-8468393.749943953%2C4742167.858473372%2C-8463729.79741333%2C4745067.763179177&bboxSR=102100&imageSR=102100&size=1277%2C794&f=image
3	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/export?dpi=96&transparent=true&format=png8&bbox=-8684977.236046717%2C4715019.35603861%2C-8393659.099411929%2C4785477.696061908&bboxSR=102100&imageSR=102100&size=1505%2C364&f=image
4	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/PlanningCadastre/MD.State.SustainableCommunities_WM/MapServer/export?dpi=96&transparent=true&format=png8&bbox=-8684977.236046717%2C4715019.35603861%2C-8393659.099411929%2C4785477.696061908&bboxSR=102100&imageSR=102100&size=1505%2C364&f=image
5	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap_Gazetteer_WM/MapServer/tile/1/258/178
6	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap

#	**Step**
	_Gazetteer_WM/MapServer/tile/1/259/178
7	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap_Gazetteer_WM/MapServer/tile/4/2068/1442
8	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap_Gazetteer_WM/MapServer/tile/4/2069/1442
9	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap_Gazetteer_WM/MapServer/tile/4/2068/1443
10	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap_Gazetteer_WM/MapServer/tile/4/2069/1443
11	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap_Gazetteer_WM/MapServer/tile/4/2068/1444
12	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap_Gazetteer_WM/MapServer/tile/4/2069/1444

Scenario 3 - Identify

#	**Step**
1	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Boundaries/MD.State.CongressionalDistricts.2011_WM/MapServer/identify?geometryType=esriGeometryPoint&geometry=-8821338.1342%2C4804111.682&sr=&layers=0&time=&layerTimeOptions=&layerdefs=&tolerance=1&mapExtent=-8684977.236046717%2C4715019.35603861%2C-8393659.099411929%2C4785477.696061908&imageDisplay=600%2C550%2C96&returnGeometry=false&maxAllowableOffset=&f=json
2	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/identify?geometryType=esriGeometryPoint&geometry=-8821338.1342%2C4804111.682&sr=&layers=0&time=&layerTimeOptions=&layerdefs=

#	Step
	&tolerance=1&mapExtent=-8684977.236046717%2C4715019.35603861%2C-8393659.099411929%2C4785477.696061908&imageDisplay=600%2C550%2C96&returnGeometry=false&maxAllowableOffset=&f=json
3	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/PlanningCadastre/MD.State.SustainableCommunities_WM/MapServer/identify?geometryType=esriGeometryPoint&geometry=-8562228.1733%2C4715913.8137&sr=&layers=0&time=&layerTimeOptions=&layerdefs=&tolerance=1&mapExtent=-8684977.236046717%2C4715019.35603861%2C-8393659.099411929%2C4785477.696061908&imageDisplay=600%2C550%2C96&returnGeometry=false&maxAllowableOffset=&f=json

Scenario 4 - Query

#	Step
1	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/PlanningCadastre/MD.State.SustainableCommunities_WM/MapServer/0/query?geometry=&geometryType=esriGeometryPoint&inSR=&spatialRel=esriSpatialRelIntersects&relationParam=&objectIds=&where=objectid+%3C+29&time=&returnCountOnly=false&returnIdsOnly=false&returnGeometry=true&maxAllowableOffset=&outSR=&outFields=*&f=pjson

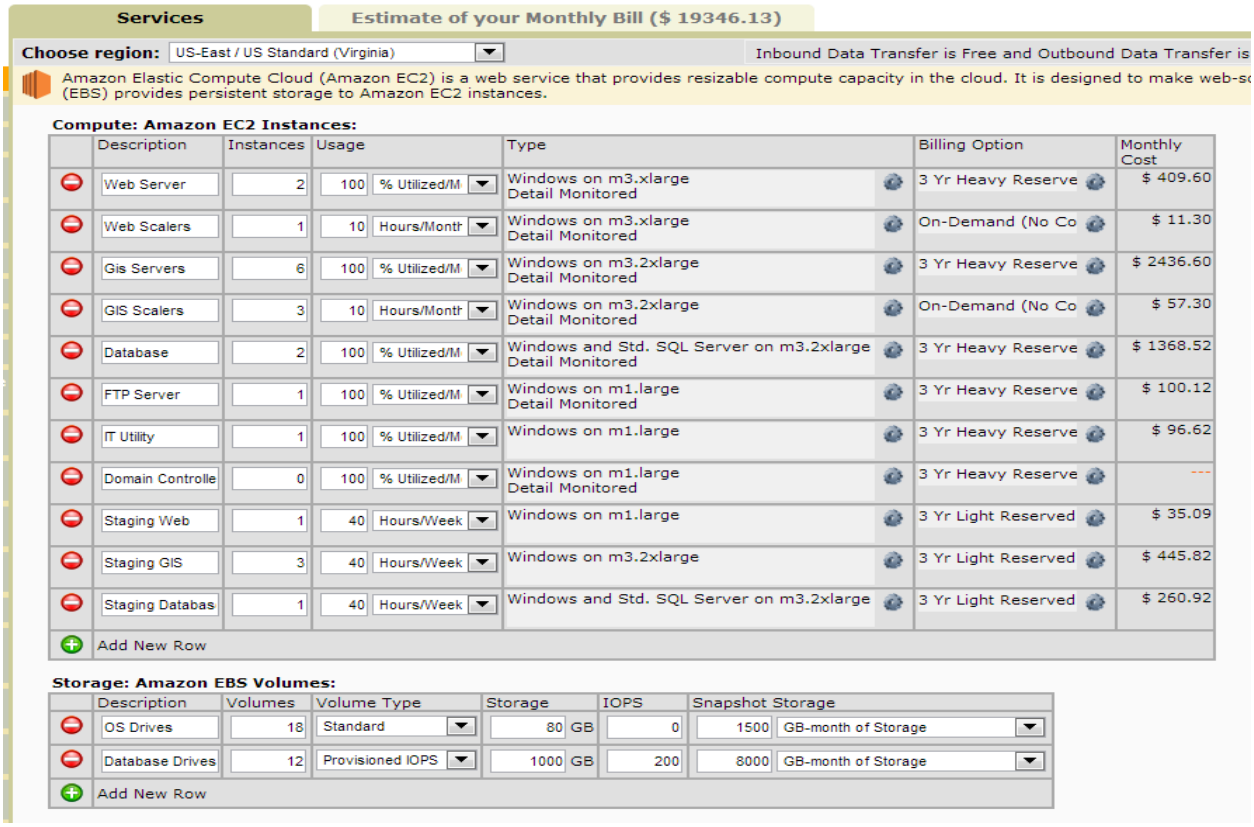
Scenario 5 - Legend

#	Step
1	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Boundaries/MD.State.CongressionalDistricts.2011_WM/MapServer/legend
2	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ClimatologyMeteorologyAtmosphere/MD.State.StormSurgeAreas_WM/MapServer/legend
3	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/Environment/MD.State.ProgramOpenSpace_WM/MapServer/legend
4	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/ImageryBaseMapsEarthCover/MD.State.MDiMap

#	**Step**
	_Gazetteer_WM/MapServer/legend
5	Open http://ec2-54-204-168-159.compute-1.amazonaws.com/ArcGIS/rest/services/PlanningCadastre/MD.State.SustainableCommunities_WM/MapServer/legend

Appendix B – Amazon EC2 Cost Estimate Detail (May 2013)

3-year RIs (Heavy and Light)



Services | Estimate of your Monthly Bill ($ 19346.13)

Choose region: US-East / US Standard (Virginia) — Inbound Data Transfer is Free and Outbound Data Transfer is

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-s
(EBS) provides persistent storage to Amazon EC2 instances.

Compute: Amazon EC2 Instances:

Description	Instances	Usage	Type	Billing Option	Monthly Cost
Web Server	2	100 % Utilized/M	Windows on m3.xlarge Detail Monitored	3 Yr Heavy Reserve	$ 409.60
Web Scalers	1	10 Hours/Month	Windows on m3.xlarge Detail Monitored	On-Demand (No Co	$ 11.30
Gis Servers	6	100 % Utilized/M	Windows on m3.2xlarge Detail Monitored	3 Yr Heavy Reserve	$ 2436.60
GIS Scalers	3	10 Hours/Month	Windows on m3.2xlarge Detail Monitored	On-Demand (No Co	$ 57.30
Database	2	100 % Utilized/M	Windows and Std. SQL Server on m3.2xlarge Detail Monitored	3 Yr Heavy Reserve	$ 1368.52
FTP Server	1	100 % Utilized/M	Windows on m1.large Detail Monitored	3 Yr Heavy Reserve	$ 100.12
IT Utility	1	100 % Utilized/M	Windows on m1.large	3 Yr Heavy Reserve	$ 96.62
Domain Controlle	0	100 % Utilized/M	Windows on m1.large Detail Monitored	3 Yr Heavy Reserve	---
Staging Web	1	40 Hours/Week	Windows on m1.large	3 Yr Light Reserved	$ 35.09
Staging GIS	3	40 Hours/Week	Windows on m3.2xlarge	3 Yr Light Reserved	$ 445.82
Staging Databas	1	40 Hours/Week	Windows and Std. SQL Server on m3.2xlarge	3 Yr Light Reserved	$ 260.92

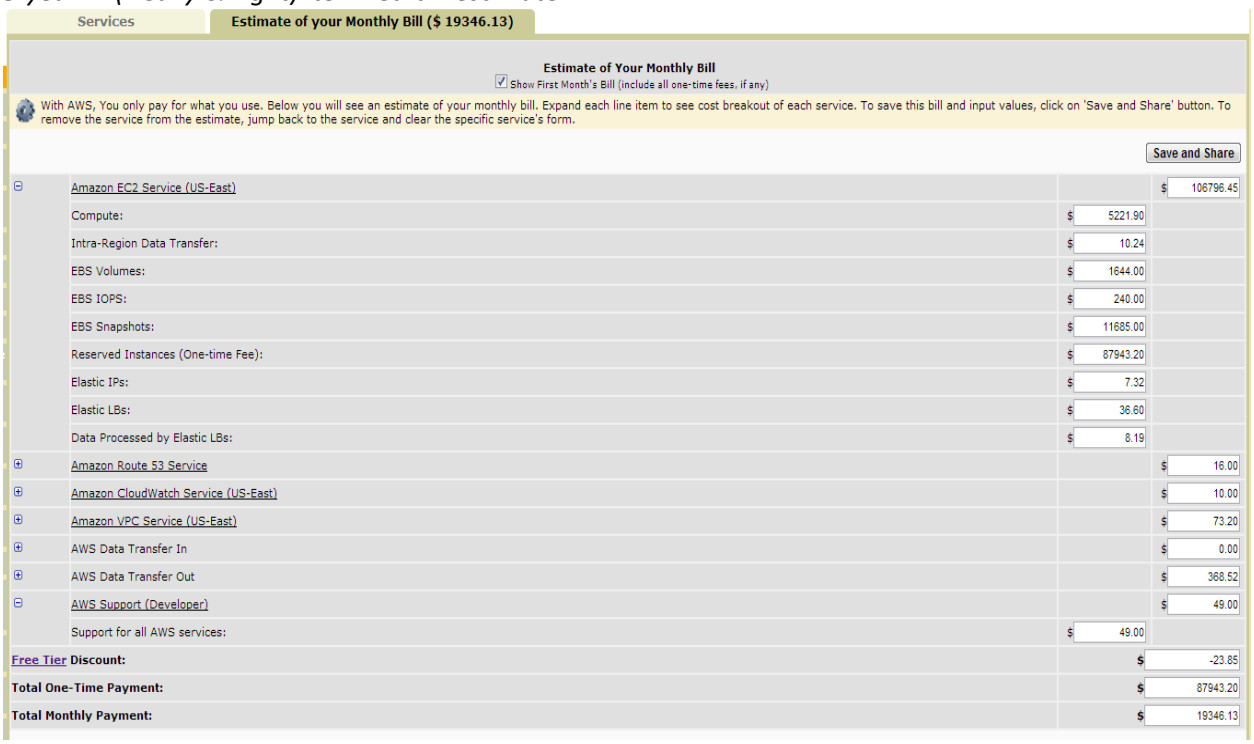
Add New Row

Storage: Amazon EBS Volumes:

Description	Volumes	Volume Type	Storage	IOPS	Snapshot Storage
OS Drives	18	Standard	80 GB	0	1500 GB-month of Storage
Database Drives	12	Provisioned IOPS	1000 GB	200	8000 GB-month of Storage

Add New Row

3-year RI (Heavy & Light) itemized bill estimate



Services | Estimate of your Monthly Bill ($ 19346.13)

Estimate of Your Monthly Bill

Show First Month's Bill (include all one-time fees, if any)

With AWS, You only pay for what you use. Below you will see an estimate of your monthly bill. Expand each line item to see cost breakout of each service. To save this bill and input values, click on 'Save and Share' button. To remove the service from the estimate, jump back to the service and clear the specific service's form.

Save and Share

Amazon EC2 Service (US-East)		$ 106796.45
Compute:	$ 5221.90	
Intra-Region Data Transfer:	$ 10.24	
EBS Volumes:	$ 1644.00	
EBS IOPS:	$ 240.00	
EBS Snapshots:	$ 11685.00	
Reserved Instances (One-time Fee):	$ 87943.20	
Elastic IPs:	$ 7.32	
Elastic LBs:	$ 36.60	
Data Processed by Elastic LBs:	$ 8.19	
Amazon Route 53 Service		$ 16.00
Amazon CloudWatch Service (US-East)		$ 10.00
Amazon VPC Service (US-East)		$ 73.20
AWS Data Transfer In		$ 0.00
AWS Data Transfer Out		$ 368.52
AWS Support (Developer)		$ 49.00
Support for all AWS services:	$ 49.00	
Free Tier **Discount:**		$ -23.85
Total One-Time Payment:		$ 87943.20
Total Monthly Payment:		$ 19346.13