Network Capabilities of Cloud Services for a Real Time Scientific Application:CloudCast

Shruti
Dept of Computer Science and Engineering,
SJB Institute of Technology
Bangalore,India

Mrs Archana R A,Asst.prof
Dept of Computer Science and Engineering
SJB Institute of Technology
Bangalore,India

Shruti.sajjanshetty@gmail.com

Abstract:- **Dedicating high-end servers for executing scientific applications that run intermittently, such as severe weather detection or generalized weather forecasting, wastes resources.**
While the Infrastructure-as-a-Service (IaaS) model used by today's cloud platforms is well-suited for the bursty computational demands of these applications, it is unclear if the network
capabilities of today's cloud platforms are sufficient. In this paper, we analyze the networking capabilities of multiple commercial (Amazon's EC2 and Rackspace) and research (GENICloud and ExoGENI cloud) platforms in the context of a Nowcasting application, a forecasting algorithm for highly accurate, nearterm, e.g., 5-20 minutes, weather predictions. The application has both computational and network requirements. While it executes rarely, whenever severe weather approaches, it benefits from an IaaS model; However, since its results are time-critical, enough bandwidth must be available to transmit radar data to cloud platforms before it becomes stale. We conduct network capacity measurements between radar sites and cloud platforms throughout the country. Our results indicate that ExoGENI cloud
performs the best for both serial and parallel data transfer with an average throughput of 110.22 Mbps and 17.2 Mbps, respectively. We also found that the cloud services perform
better in the distributed data transfer case, where a subset of nodes transmit data in parallel to a cloud instance. Ultimately,we conclude that commercial and research clouds are capable of providing sufficient bandwidth for our real-time Nowcasting application.

I. INTRODUCTION

Dedicating high-end servers for executing scientific applications that run intermittently, such as severe weather detection or generalized weather forecasting, wastes resources. The infrastructure- as-a-service (IaaS) model used by today's cloud platforms is well suited for the bursty computational demands of these applications.Clouds are emerging as the primary host platform for a variety of applications, such as DropBox, iCloud, and Google Music.These applications let users store data in the cloud and access it from anywhere in the world. Commercial clouds are also well suited for renting high-end servers to execute applications that require computation resources sporadically.Cloud users pay only for the time they actually use resources and the data they transmit to and from the server, which has the potential to be more cost effective than purchasing, hosting, and maintaining dedicated hardware.With this in mind, we present CloudCast, a new application for short-term, location-based weather prediction using cloud platforms. If severe weather is approaching the user's location, CloudCast automatically sends personalized notifications. To enable this functionality, CloudCast combines two major components. The first component is an algorithm that produces fine-grained, short-term weather forecasts—called *Nowcasting*1,2—up to 15 minutes in the future for areas as small as 100 m2. Nowcasting has the potential to personalize severe weather alerts by programmatically transmitting highly targeted, location-specific warnings to mobile devices based on their GPS coordinates. The second component is a new architecture that allows the execution of Nowcasting on cloud platforms, such as Amazon's Elastic Compute Cloud (EC2; http://aws.amazon.com/ec2). Nowcasting is compute-intensive, requiring high-memory systems for execution. Hence, we use cloud services to execute the Nowcasting algorithm for our CloudCast application. Cloud computing platforms lower the cost of computation by leveraging economies-of-scale. Generating Nowcasts on a dedicated infrastructureis economically infeasible due to its enormous computational demands, which scale quadratically with spatial resolution. Thus, to understand the feasibility of hosting Nowcasting on cloud platforms, we analyze the network and computation capability of four different cloud services: two commercial cloud services (Amazon EC2 and Rackspace [www.rackspace.com]) and two research cloud testbeds (the Global Environment for Network Innovations' GENICloud3,4 and ExoGENI
[http://wiki.exogeni.net]). Commercial clouds use the pay-as-you-use model, charging users for resource usage on an hourly basis. In contrast, research clouds such as GENICloud and ExoGENI cloud provide free resources for the research community. These research platforms offer additional advantages, beyond being free for the research community. First, researchers can use them to develop prototypes of scientific cloud applications. Second, research clouds such as the ExoGENI allow for dynamic configuration of the network

IJERT

topology within the cloud, a feature that isn't provided by commercial clouds. Finally, research clouds are often connected via next-generation research networks—such as the National LambdaRail (NLR) FrameNet (www.nlr.net/framenet.php) or Internet2 ION (https://geni-orca.renci.org/ trac/wiki/flukes)—which allow the provisioning of dedicated, isolated network resources. The latter will help researchers better understand how distributed applications that run in the cloud can benefit from new network technologies. The primary reason weather services cite for not leveraging the cloud is data staging costs. We evaluate these costs for the extreme case of Nowcasting, which requires realtime radar data uploads to predict conditions tens of minutes in the future. Compared to most cloud applications, CloudCast's Nowcasting algorithm has stricter time constraints. Timely execution of the algorithm is critical, because the Nowcast data must be made available to end users before it becomes obsolete. For example, in a severe weather scenario, we can use Nowcast information to warn the public, as well as to guide spotters and other emergency management personnel. Because Nowcasting predicts weather only in the very near-term future, it's important that the algorithm produces results fast. For instance, if it takes 12 minutes to generate and disseminate a 15-minute Nowcast, that leaves just three minutes for users to take action. Our hypothesis is that the connectivity and diversity of current cloud platforms mitigate the impact of Nowcasting's staging data and computation latency, enabling the platforms to perform the atmospheric modeling and personalized weather forecasting required by Cloud- Cast. In evaluating our hypothesis, we present the CloudCast architecture, which links weather radars to cloud instances, allowing the architecture's components to request computational and storage resources required to generate forecasts based on real-time radar data feeds as requested from clients. We emulate a radar network usingPlanetLab sites and conduct extensive bandwidth measurements between each site and cloud instances. We quantify average bandwidth and its variability to determine if the public Internet and today's clouds are sufficient for real-time Nowcasting. We also analyze the computation time and cost of Nowcasting in the cloud for instances offered by cloud services and demonstrate CloudCast live using a deployed prototype radar as a proof-of-concept.

Cloud computing provides scalable computing and storage resources via the Internet. It also enables users to access services without regard to where the services are provided and how they are delivered, similar to water, electricity, gas, and telephony utilities. With the flexible and transparent features in the resource allocation and service provisioning, more and more data-intensive applications are developed in the cloud computing environment. The dataintensive applications devote most of their execution time in disk I/O for processing a large volume of data, e.g. data mining of commercial transactions, satellite data processing, web search engine, etc. Apache Hadoop is an emerging cloud computing platform dedicated for data-intensive applications. Due to a large number of nodes in the cloud computing system, the probability of hardware failures is non-trivial based on the statistical analysis of hardware failures in. Some hardware failures will damage the disk data of nodes. As a result, the running data-intensive applications may not read data from disks successfully. To tolerate the data corruption, the data replication technique is extensively adopted in the cloud computing system to provide high data availability. However, the QoS requirement of an application is not taken into account in the data replication. When data corruption occurs, the QoS requirement of the application cannot be supported continuously. The reason is explained as follows. With a large number of nodes in the cloud computing system, it is difficult to ask all nodes with the same performance and capacity in their CPUs, memory, and disks. For example, the Amazon EC2 is a realistic heterogeneous cloud platform, which provides various infrastructure resource types to meet different user needs in the computing and storage resources .The cloud computing system has heterogeneous characteristics in nodes. Due to the node heterogeneity, the data of a high-QoS application may be replicated in a low-performance node (the node with slow communication and disk access latencies). Later, if data corruption occurs in the node running the high-QoS application, the data of the application will be retrieved from the low-performance node. Since the low-performance node has slow communication and disk access latencies, the QoS requirement of the high-QoS application may be violated. Note that the QoS requirement of an application is defined from the aspect of the request information. For example, in the response time of a data object access is defined as the QoS requirement of anapplication in the content distribution system.

IJERT

This paper investigates the *QoS-aware data replication* (QADR) problem for data-intensive applications in cloud computing systems. The QADR problem concerns how to efficiently consider the QoS requirements of applications in the data replication. The main goal of the QADR problem is to minimize the data replication cost and the number of QoSviolated data replicas. With minimizing the data replication cost, the data replication can be completed quickly. This can significantly reduce the probability that the data corruption occurs before completing data replication. Due to limited replication space of a storage node, the data replicas of some applications may be stored in lower-performance nodes. This will result in some data replicas that cannot meet the QoS requirements of their corresponding applications. These data replicas are called the QoS-violated data replicas. The number*]* of QoS-violated data replicas is expected to be as small as possible.
To solve the QADR problem, we first propose a greedy algorithm, called the *high-QoS first replication* (HQFR) algorithm. In this algorithm, if application i has a higher QoS requirement, it will take precedence over other applications to perform data replication. However, the HQFR algorithm cannot achieve the above minimum objective. Basically, the optimal solution of the QADR problem can be obtained by formulating the problem as an *integer linear programming* (ILP) formulation. However, the ILP formulation involves complicated computation. To find the optimal solution of the QADR problem in an efficient manner, we propose a new algorithm to solve the QADR problem. In this algorithm, the QADR problem is transformed to the *minimum-cost maximumflow* (MCMF) problem. Then, an existing MCMF algorithm is utilized to optimally solve the QADR problem in

polynomial time. Compared to the HQFR algorithm, the optimal algorithm takes more computational time. However, the two proposed replication algorithms run in

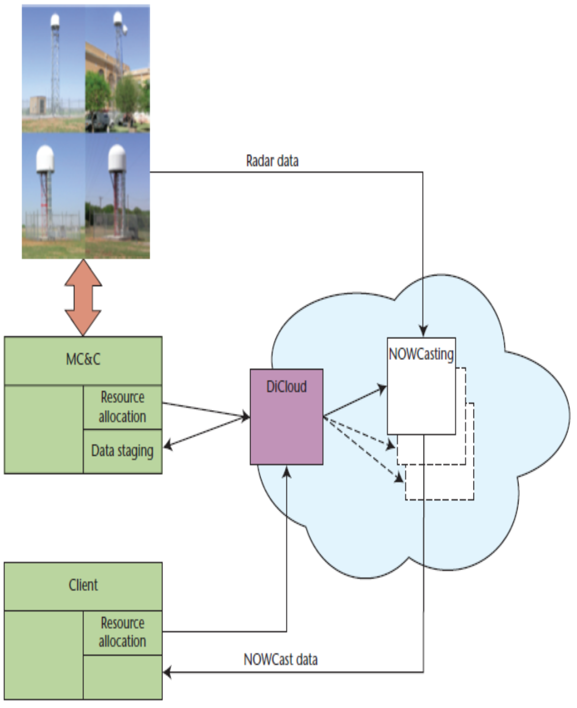


Figure 1. Overview of the CloudCast system architecture. The application is composed of meteorological command and control (MC&C), cloud instances, and the Nowcasting short-term weather forecasting algorithm.

polynomial time. Their time complexities are dependent on the number of nodes in the cloud computing system. To accommodate to a large-scale cloud computing system, the scalable replication issue is particularly considered in our QADR problem. We use node combination techniques to suppress the computational time of the QADR problem without linear growth as increasing the number of nodes. To the best of our knowledge, this is the first paper to investigate the QADR problem in the cloud computing system. Overall, the main contributions of this paper are summarized as follows.

• Unlike previous data replication schemes of the cloud computing system, our data replication algorithms consider the QoS requirements of applications.

• For optimally solving the QADR problem, we present how to formulate the QADR problem as an ILP formulation. Due to considering the computational complexity in solving the ILP formulation, we transform the QADR problem to the MCMF problem to obtain the polynomialtime optimal solution.

a. Existing system

Dedicating high-end servers for executing scientific applications that run intermittently, such as severe weather detection or generalized weather forecasting, wastes resources.

b. Proposed System

The infrastructure-as-a-service (IaaS) model used by today's cloud platforms is well suited for the bursty computational demands of these applications. Clouds are emerging as the primary host platform for a variety of applications, such as DropBox, iCloud, and Google Music. These applications let users store data in the cloud and access it from anywhere in the world. Commercial clouds are also well suited for renting high-end servers to execute applications that require computation resources sporadically

CloudCast, a new application for short-term, location-based weather prediction using cloud platforms. If severe weather is approaching the user's location, CloudCast automatically sends personalized notifications.

MC&C Architecture

MC&C is the control part of the Collaborative Adaptive Sensing of Atmosphere (CASA) network that determines how the radars will scan the atmosphere in each 60-second heartbeat. The MC&C architecture considers these different factors to determine how to control the radars. For our CloudCast approach, we use the MC&C's detection features. The algorithms detecting the existence and current location of precipitation are ingested into the MC&C's blackboard architecture; it's then able to classify the situation on multiple levels. On a high level, it differentiates between clear air, stratiform rain, and convective regimes, and each regime has a set of tasks associated. In clear air mode, the need for computational resources diminishes, whereas convective mode has strict requirements for data collection and heavily utilizes computers and networks. Rain sensed
by the radars will be detected by the MC&C's algorithms; CloudCast then uses the information to determine when to initiate Nowcasts in the cloud. Thus, cloud-based Nowcasts are automatically initiated and halted without user intervention.

NOWCASTING

Nowcasting1,2 refers to short-term (less than 30 minutes) weather forecasting. The Nowcasting algorithm predicts high-impact weather events, such as flood-producing rainfall, severe storms, and hail, in a specific region with sufficient accuracy within a time frame such that appropriate actions can be taken to effectively mitigate the loss of life and property. Because Nowcasting is a short-term weather-prediction system, its applications include warning-decision support for detecting potentially severe weather. Its performance is typically measured in terms of a categorical yes/no (such as rain/ no-rain) detection relative to a predetermined measurement threshold representative of a desired threat. This model of measuring performance is well suited for our Nowcasting application, because the Nowcasting algorithm's ability to predict a sufficiently high reflectivity value in a given region is important for end user emergency decision support.

II. MEASUREMENTS

a. Experiment

Now that we have a better understanding of CloudCast's architecture, let's investigate the network feasibility of cloud services for Cloud- Cast. As you can see in Figure 1, a potential bottleneck for the real-time operation of Nowcasting

IJERT

in the cloud is the link between the radars and the cloud instances.

To understand the network capability of cloud services for Nowcasting, we perform a series of measurements in a large-scale setting by replicating a distribution system that, at least on the network level, is similar to the Next- Generation Radar (Nexrad) system. Kevin Kelleher and his colleagues8 give an overview on how data from Nexrads are disseminated by using the National Oceanic and Atmospheric Administration's NOAANet backbone and Internet2. We perform our measurements in two different radar settings: one measurement emulates Nexrad radars and the other uses our four-node radar network that's located in southwestern Oklahoma.5,9 Both Nexrad and CASA radars use Internet2 as their backbone network and thus have similar network settings. Because we don't have access to Nexrad nodes for our measurement, we use PlanetLab,6 a global research network that supports large-scale, distributed experiments. Previous work showed that radars generate data at a constant rate of roughly 5 megabits per second (Mbps).5 For the remainder of our analysis, we use 5 Mbps as the minimum required throughput between a radar node and the cloud instances to allow real-time data transmission for Nowcasting operation. We conduct four different measurement scenarios to depict the network capabilities of cloud services considered for our Nowcasting application. We conduct a serial measurement where data are transmitted to cloud instances from each individual PlanetLab node (replicating Nexrad radars). We conduct this measurement to verify how the cloud instances perform without any competing traffic. We also conduct a parallel measurement, where data from all the PlanetLab nodes transmit data to a cloud instance at the same time. We perform this experiment to verify if the cloud instances can handle the large traffic from all the Nexrad radars at once and still maintain the required minimum threshold throughput of 5 Mbps for our CloudCast application. Because not all radar nodes around the country would transfer their data to one central instance simultaneously, a more likely scenario is the case in which a particular geographic region's radar nodes will transmit their data to a cloud instance that's close to this subset of radar nodes. To investigate this scenario, we conducted a distributed measurement where only 10 PlanetLab nodes transmit data to a cloud instance in parallel. The measurements explained so far consider data sent from the radar continuously, but in a real scenario, the data is sent in bursts every minute. From the Nexrad data collected, we've seen that 7.5 Mbytes of data is sent from the radar every minute. Hence, we perform a bursty traffic measurement to understand how the cloud services perform for bursty data sent from the radars. For our measurements, we use Iperf (http://iperf.sourceforge.net) to transit data from radar nodes to cloud instances.

Table 1 gives an overview of the average throughput measurement from Nexrad radar nodes (PlanetLab nodes) and CASA radar nodes to the commercial cloud instances and research cloud instances. To investigate if the location of the EC2 instance has an impact on throughput, we performed the measurement twice—once with an EC2 instance in a West Coast data center and another in the EC2 East Coast data center. The results of serial measurements for both the Nexrad radar nodes and CASA radar nodes in Table 1 show that both the research cloud testbed nodes and the commercial cloud service nodes perform well without competing traffic, with an average throughput above the required threshold of 5 Mbps.

b. *Measurement Result*

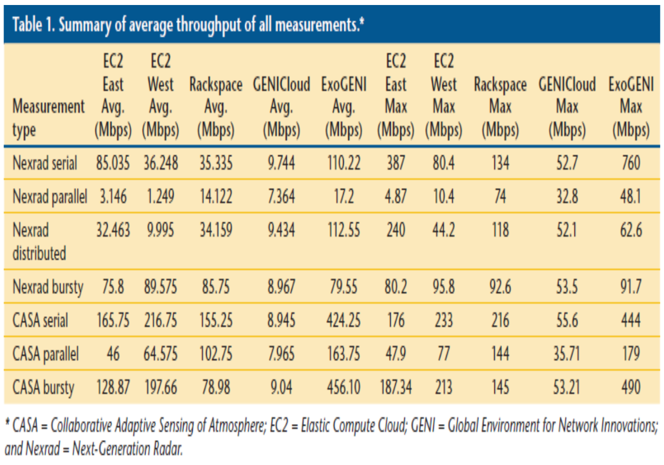


Table 1. Summary of average throughput of all measurements.*

Measurement type	EC2 East Avg. (Mbps)	EC2 West Avg. (Mbps)	Rackspace Avg. (Mbps)	GENICloud Avg. (Mbps)	ExoGENI Avg. (Mbps)	EC2 East Max (Mbps)	EC2 West Max (Mbps)	Rackspace Max (Mbps)	GENICloud Max (Mbps)	ExoGENI Max (Mbps)
Nexrad serial	85.035	36.248	35.335	9.744	110.22	387	80.4	134	52.7	760
Nexrad parallel	3.146	1.249	14.122	7.364	17.2	4.87	10.4	74	32.8	48.1
Nexrad distributed	32.463	9.995	34.159	9.434	112.55	240	44.2	118	52.1	62.6
Nexrad bursty	75.8	89.575	85.75	8.967	79.55	80.2	95.8	92.6	53.5	91.7
CASA serial	165.75	216.75	155.25	8.945	424.25	176	233	216	55.6	444
CASA parallel	46	64.575	102.75	7.965	163.75	47.9	77	144	35.71	179
CASA bursty	128.87	197.66	78.98	9.04	456.10	187.34	213	145	53.21	490

* *CASA = Collaborative Adaptive Sensing of Atmosphere; EC2 = Elastic Compute Cloud; GENI = Global Environment for Network Innovations; and Nexrad = Next-Generation Radar.*

Out of the cloud instances investigated, the ExoGENI cloud instance performs best without competing traffic, yielding an average throughput of 110.22 Mbps for the Nexrad radar links and 424.25 Mbps
for CASA radar links. ExoGENI was followed by Amazon's EC2 East Coast data center cloud instance with 85.03 Mbps; the EC2 West Coast data center
cloud instance with 36.24 Mbps; the Rackspace cloud instance with 35.33 Mbps; and, finally, the GENICloud instance with a mere average throughput of 9.71 Mbps for Nexrad radar links. For the CASA radar links, the EC2 West Coast instance performs better than the EC2 East Coast instance, but the average still remains above our required threshold throughput of 5 Mbps for both radar links. The parallel measurement rows for Nexrad and CASA radars shown in Table 1 provide results that are quite different from the aforementioned serial measurement results. The ExoGENI, Rackspace, and GENICloud cloud instances yield a better average throughput during the parallel measurement than EC2 cloud instances. ExoGENI, Rackspace, and GENICloud cloud instances yield an average throughput of 17.2, 14.12, and 7.68 Mbps, respectively, for Nexrad radar links, which is greater than the threshold throughput of 5 Mbps required for our Nowcasting application. EC2 cloud instance links to Nexrad radar nodes yield an average throughput of 3.14 Mbps and 1.24 Mbps for the East and West Coast data centers, respectively, which is well below the threshold throughput of 5 Mbps. CASA radar links to cloud instances perform better for parallel measurements, as Table 1 shows, yielding an average throughput of more than 5 Mbps for each cloud instance.

These cloud instances perform better when only a subset of nodes is transmitting data in parallel, as shown in the distributed measurement row for Nexrad radars in Table 1. As in the serial measurement scenario, the average throughput results from all the cloud instances are greater than the threshold throughput of 5 Mbps. The ExoGENI cloud instance performs better than the other three cloud instances, with an average throughput of 112.55 Mbps, while the Rackspace

IJERT

cloud instance provides an average throughput of 34.15 Mbps. The GENICloud instance provides an average throughput of 9.43 Mbps, and the EC2 cloud service provides an average throughput of 32.46 and 9.99 Mbps in the East and West Coast data centers, respectively. The results from the bursty traffic measurement experiment, where data are transmitted in bursts of 7.5 Mbytes every minute, yield an average throughput greater than the required threshold throughput of 5 Mbps for each of the cloud instances considered, for both Nexrad radar nodes and CASA radar links. Table 1 summarizes the results from the bursty traffic measurement. From our measurement, we conclude that the networking capabilities of the cloud instances are sufficient for our real-time Nowcasting application. We also infer that the network performance of research cloud testbeds are on par with that of the commercial cloud services, and we can use them as a test instance to execute the Nowcasting application without incurring any additional cost. Additional data from our measurements can be found elsewhere.10

c. *Mobile bandwidth*

The goal to develop an architecture that allows the execution of on-demand Nowcast algorithms in the cloud to provide short-term weather forecasts to mobile devices. So far we have investigated the network characteristics for data that is transmitted from weather sensors to the cloud. Another important link in the overall system is the network from an instance in the cloud to the end user's mobile device. In order to characterize that link we performed an iperf measurement from a mobile device to EC2 cloud data centers. For our measurement we make use of a laptop that connects to the Internet via a 3G modem. We perform the measurements in 3 different locations (at work, at home and in the car traveling on a major highway) for the duration of an hour. Similar to earlier measurements, we perform this experiment in both East and West EC2 instances. The results from the measurements are shown in Figure 2. The measurement results show a surprisingly high variance in throughput, from tens of Kbps (home) up to slightly over 1100 Kbps (car) in both measurements. In addition to the high variance, it is somewhat surprising that the scenario in which the highest and lowest throughput are obtained are from a measurement performed while traveling in a car on a highway. The huge differences in throughput lead to significant differences in download time of the Nowcasting images from the central web server in CloudCast. Based on the data from this measurement we derive that for the highest throughput it would only take 0.8 seconds to download the images while it would take 35.8 seconds in the worst throughput case.

d. *Nowcasting Analysis*

Having investigated the network feasibility of commercial and research cloud services for our CloudCast application, we can present the cost and computation time analysis of cloud-based Nowcast generation on the cloud instances considered. As a proof-of-concept of our application, we present the results of the live analysis performed with our own radar located on our campus. We also perform an analysis that compares the cost of running Nowcast in the cloud to the cost of using dedicated hardware.

Table 2. Nowcast algorithm execution time for live measurement.

Instances	Memory (Gbytes)	Disk (Gbytes)	Cost/hour (US$)	Total cost (US$)	Execution time (s)	Total time (s)
Amazon EC2	7.5	850	0.34	1.13	74.34	95.08
Rackspace	8	320	0.48	1.63	96.53	120.33
GENICloud	8	20	–	–	67.45	78.60
ExoGENI	8	20	–	–	56.83	72.07

e. *Computation Time*

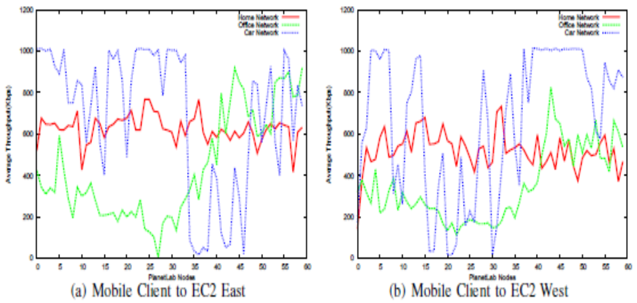
Calculate Nowcasting's cost and computation time for one hour of weather data for the similar instance types offered by Amazon EC2 andRackspace cloud services. We also calculateNowcasting's computation time for the sameweather data with the instances offered by GENICloud and ExoGENI research cloud services. The weather data used was collected during a severe weather event in May 2011. For each of the cloud services mentioned in
Table 2, we bring up the instances with the Nowcasting image and start the ingest of weather data from the radars. Once the cloud-based Nowcast instance receives the first set of radar scans, it starts to generate 1- to 15-minute Nowcasts, which are kept on the instance's storage. We carry out this operation for one hour of weather data and determine the cost for running this 1-hour Nowcast operation using the cost-tracking services provided by Amazon EC2 and Rackspace in their instance-management console. In addition, the execution time for 15 minute Nowcasting of each weather data from the radar in the cloud instances is measured and the mean is calculated over the whole 1-hour interval.

As Table 2 shows, to generate 15-minute Nowcasts on the commercial cloud services, EC2's average computation time is 74.34 seconds and Rackspace's average is 96.53 seconds. The bootup time of the instances is approximately 7 minutes on average for both EC2 and Rackspace. Hence, generating the first 15-minute Nowcast takes about 8 minutes and 14 seconds for EC2 and 8 minutes and 36 seconds for Rackspace, whereas the subsequent 15 minute Nowcast generation takes only about 74 and 96 seconds for EC2 and Rackspace, respectively. Comparing their cost versus computation time to generate 15-minute Nowcasts, we see that EC2's computation time and cost is less than that of Rackspace.
We also perform the computation time analysis on the research cloud service instances with GENICloud and ExoGENI for the same weather data used to analyze the computation time of commercial cloud services. Table 2 shows the results of our analysis. Both research cloud services offer only one type of instance, which is sufficient for the standard operation of Nowcasting application. As Table 2 shows, both research cloud instances take less time (67.45 and 56.83 seconds, respectively) to compute 15-minute Nowcasts than EC2 and Rackspace. ExoGENI computes the 15-minute Nowcasts the fastest (just 56.83 seconds), and the bootup time for GENICloud and ExoGENI cloud instances are about 2 minutes on average compared to 7 minutes for EC2 and Rackspace. As a proof-of-

IJERT

concept for our CloudCast application on cloud services, we carry out a live measurement on each of the four cloud instances to calculate the overall time taken for the Nowcasting process—that is, data is generated by the radar, the data is transmitted to the instance executing the algorithm, the 15-minute Nowcast images are generated, and the images are sent to a central webserver to be used by clients. The overall duration of the sum of the individual steps determines how much time a user has between when a severe weather situation is indicated by the Nowcast and when it actually occurs. Obviously, the goal is to maximize that time interval. For the live measurement analysis, we use the data from our own radar on campus, which is a prototype CASA radar.9 The last column in Table 2 shows the results from the live measurement carried out on the cloud instances. The average overall time taken for the wholeNowcasting process was about 95.08 seconds for the EC2 cloud instance, of which 71.98 seconds is consumed in generating 15-minute Nowcasts by the algorithm running on the cloud instance.
Thus, it takes about 23.10 seconds for the data to be sent from the radar to the receiving instance, create the predicted images, and transfer the images back to the central server to be accessible by clients. Similarly, the total time taken for the whole Nowcasting process on Rackspace, GENICloud, and ExoGENI cloud instances is 120.33, 78.60, and 72.07 seconds, respectively. Thus, with our live measurement for the potential and feasibility of performing short-term weather forecasts for mobile devices in the cloud, we found that, for 15-minute Nowcasting, it takes only approximately two minutes to generate the Nowcast images and disseminate them to the client. That gives the clients 13 minutes to take any necessary action based on the 15-minute prediction. Additional information on Nowcasting accuracy on cloud services can be found elsewhere.

f.Mobile bandwidth



(a) Mobile Client to EC2 East (b) Mobile Client to EC2 West

Fig. 2. Mobile Device to EC2 Bandwidth Measurement

The Measurement results show a surprisingly high variance in throughput, from tens of Kbps (home) up to slightly over 1100 Kbps (car) in both measurements. In addition to the high variance, it is somewhat surprising that the scenario in which the highest and lowest throughput are obtained are from a measurement performed while traveling in a car on a highway. The huge differences in throughput lead to significant differences in download time of the Nowcasting images from the central web server in CloudCast. Based on the data from this measurement we derive that for the highest throughput it would only take 0.8 seconds to download the images while it would take 35.8 seconds in the worst throughput case.

III. CONCLUSION

Commercial and research cloud services are feasible for the execution of our real-time CloudCast application and can provide accurate, short-term weather forecasts to end users. We believe that CloudCast has the potential to support emergency managers and the general public in severe weather events by promptly providing them with potentially life-saving information. In the future, we would like to build a prototype of our architecture in collaboration with cloud services to run Nowcasting on demand for a longer experimentation.
With this approach accurate, short-term weather forecasts can be provided to mobile users. We believe that CloudCast has the potential to support emergency managers and the general public in severe weather events by promptly providing them with potentially life-saving information.

IV. BIBLIOGRPHY

1. E. Ruzanski, Y. Wang, and V. Chandrasekar,Development of a Real-Time Dynamic and Adaptive Nowcasting System," *Proc. Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, 2009; https://ams.confex.com/ams/89annual/techprogram/paper_149633.htm.
2. E. Ruzanski, V. Chandrasekar, and Y. Wang, "The CASA Nowcasting System," *J. Atmospheric and Oceanic Technology*, 2011.
3. M. Yuen, "GENI in the Cloud," master's thesis, Dept. of Computer Science, Univ. of Victoria, 2010.
4. A.C. Bavier et al., "Transcloud—Design Considerations for a High-Performance Cloud Architecture Across Multiple Administrative Domains," *Proc. 1st Int'l Conf. Cloud* 5. M. Zink et al., "Closed-Loop Architecture for Distributed
Collaborative Adaptive Sensing: Meteorogolical Command & Control," *Int'l J. Sensor Networks*, vol. 7, nos. 1–2, 2010, pp. 4–18.
6. B. Chun et al., "PlanetLab: An Overlay Testbed for Broadcoverage Services," *SIGCOMM Computer Comm. Rev.*, vol. 33, no. 3, 2003, pp. 3–12.
7. D. Nurmi et al., "The Eucalyptus Open-Source Cloud-Computing System," *Proc. 2009 9th IEEE/ACM Int'l Symp. Cluster Computing and the Grid*, IEEE CS, 2009, pp. 124–131.
8. K.E. Kelleher et al., "Project CRAFT: A Real-Time Delivery System for NEXRAD Level II Data via the Internet," *Bull. Am. Meteorological Soc.*, vol. 88, 2007, pp. 1045– 1057; http://dx.doi.org/10.1175/BAMS-88-7-1045.
9. D. McLaughlin et al., "Short-Wavelength Technology and the Potential for Distributed Networks of Small Radar Systems," *Bull. Am. Meteorological Soc.*, vol. 90, 2009, pp. 1797–1817; http://dx.doi.org/10.1175/2009BAMS2507.1.
10. D.K. Krishnappa et al., "Network Capabilities of Cloud Services for a Real Time Scientific Application," *Proc. Local Computer Networks*, IEEE, 2012, pp. 487–495.

IJERT