CloudCast: Cloud Computing for Short-term Mobile Weather Forecasts

Dilip Kumar Krishnappa, David Irwin, Eric Lyons, Michael Zink
Electrical and Computer Engineering Department
University of Massachusetts Amherst

***Abstract*—Since today's weather forecasts only cover large regions every few hours, their use in severe weather is limited. In this paper, we present CloudCast, an application that provides short-term weather forecasts depending on users current location. Since severe weather is rare, CloudCast leverages pay-as-you-go cloud platforms to eliminate dedicated computing infrastructure. CloudCast has two components: 1) an architecture linking weather radars to cloud resources, and 2) a Nowcasting algorithm for generating accurate short-term weather forecasts. We study CloudCast's design space, which requires significant data staging to the cloud. Our results indicate that serial transfers achieve tolerable throughput, while parallel transfers represent a bottleneck for real-time mobile Nowcasting. We also analyze forecast accuracy and show high accuracy for ten minutes in the future. Finally, we execute CloudCast live using an on-campus radar, and show that it delivers a 15-minute Nowcast to a mobile client in less than 2 minutes after data sampling started.**

I. Introduction

The emergence of smart phones has led to the proliferation of a variety of innovative data-driven mobile applications or "apps".[1] One useful application is mobile weather forecasting, which provides hour-to-hour forecasts on a coarse-grained geographical scale. While today's forecasting apps (e.g., mycast [8], The Weather Channel App [15], AccuWeather [1]) are useful for long-term weather prediction, they do not provide the type of precise, small-scale, short-term predictions necessary for reacting to fast-changing severe weather. For instance, informing mobile users 10-15 minutes in advance of a tornado hitting their exact location would be a major step forward for emergency management systems.

In this paper, we present CloudCast, a new mobile application for short-term, location-based weather prediction. CloudCast is a mobile application that generates short-term forecasts based on a user's specific location. 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 (up to 15 minutes in the future) for areas as small as $100m^2$, called Nowcasting [36], [35]. 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. For example, rather than displaying an hourly forecast for a whole region, a personalized Nowcast is capable of issuing specific close-term predictions, such as whether or not it will rain in a few minutes, for a mobile device's specific location.

The second component is a new architecture that allows the execution of Nowcasting on cloud platforms, such as Amazon's Elastic Compute Cloud (EC2). Nowcasting is compute-intensive, requiring high-memory systems for execution. Hence, we use cloud services to execute Nowcasting algorithm for our CloudCast application. Cloud computing platforms lower the cost of computation by leveraging economies-of-scale. Generating Nowcasts on dedicated infrastructure is economically infeasible due to its enormous computational demands, which scale quadratically with spatial resolution. For instance, a 4x increase in resolution results in (roughly) a 16x increase in computation. As a result, a service that generates Nowcasts, similar to the National Weather Service (NWS) that generates coarse-grained forecasts for every three hours in the future up to three days, requires massive upfront capital costs for servers. Yet, since severe weather is rare, dedicating significant infrastructure to Nowcasting "wastes" server capacity most of the time. Weather services, such as MetService in New Zealand and the U.S. NWS, already invest millions of dollars in server capacity for today's coarse-grained atmospheric modeling and forecasting [3], [7], [11].

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 rapid real-time radar data uploads to predict conditions tens of minutes in the future. Compared to most cloud applications, CloudCast's Nowcasting algorithm has stricter real-time constraints. Timely execution of the algorithm is critical since the Nowcast data has to be made available to end users before it becomes obsolete. For example, in a severe weather scenario Nowcast information can be used to warn the public, as well as guide spotters and other emergency management personnel. Since Nowcasting predicts weather only in the very near-term future (on the order of minutes), it is important that the algorithm produce results fast. For instance, if it takes 12 minutes to generate and disseminate a 15-minute Nowcast, that leaves just 3 minutes for the users to take action.

Our hypothesis is that the connectivity and diversity of current cloud platforms mitigate the impact of staging data and computation latency for Nowcasting, enabling them to perform atmospheric modeling and personalized weather forecasting that CloudCast requires. In evaluating our hypothesis, this paper makes the following contributions.

[1]This work is supported by, in part, by NSF grant OCI-1032765.

- We propose CloudCast, an architecture that links weather radars to cloud instances, allowing the components in the architecture to request computational and storage resources required to generate forecasts based on real-time radar data feeds as requested from mobile clients.
- We emulate a radar network using PlanetLab 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 analyze the computation time and cost of Nowcasting in the cloud for various instance types offered by the cloud services, to quantify the trade off between faster execution of the Nowcast algorithm and higher cost that arise from renting more powerful cloud instances.
- We use statistical metrics to analyze Nowcast prediction accuracy for various aggregation levels to ensure accurate results given changing Internet conditions. We show that our system is able to achieve high accuracy using existing networks and clouds for up to 10 minute Nowcasts.
- Finally, we demonstrate CloudCast live using a deployed prototype radar as a proof-of-concept.

In Section II, we provide background on cloud services considered in our analysis, while Section III presents an overview of our CloudCast architecture. Section IV evaluates the network efficacy of cloud services for Nowcasting using an extensive measurement study on PlanetLab. Section V then presents a cost and computation analysis of Nowcasting on the cloud using cloud services and quantifies the effect of data compression on Nowcasting accuracy. Section VI presents the results of live radar-to-cloud network measurements, while Section VII discusses related work and Section VIII concludes.

II. Cloud Services

In this paper we have chosen four cloud services to analyze the compute feasibility of cloud services for our real-time application of short-term weather forecasting. We have considered two commercial cloud services—Amazon's EC2 and Rackspace Cloud Hosting—as well as two research cloud testbeds—GENICloud and ExoGENI cloud.

Commercial clouds use the pay-as-you-use model where the users are charged for resource usage on an hourly basis. In contrast, research clouds like GENICloud [41], [16] and the ExoGENI cloud [4] provide free resources for the research community. Apart from the fact that the usage of these cloud platforms is free for the research community, they bear the following additional advantages. First of all, researchers can use them to develop prototypes of scientific cloud applications. Large-scale implementation of these applications will still have to happen in commercial clouds since the research clouds provide only a limited number of resources (e.g., available compute nodes or overall storage space). Second, research clouds such as the ExoGENI (with its NEuca extensions [9]) allow for dynamic configuration of the network topology within the cloud, a feature that is not provided by commercial clouds. Third, specific research clouds are connected via next-generation research networks (NLR FrameNet [10] or Internet2 ION [5]) that allow the provisioning of dedicated, isolated network resources. The latter will help researchers to better understand how distributed applications that run in the cloud can benefit from new network technologies. This will, e.g., allow us to investigate how a dedicated layer 2 connection between the source and the receiving instance in the cloud will impact the overall performance of the application. In this section, we give a brief description of these cloud services before explaining our application architecture in Section III.

A. Elastic Compute Cloud (EC2)

Amazon's Elastic Compute Cloud (EC2) [2] is a cloud service which provides resizable compute capacity to execute applications on demand. Amazon EC2 provides a variety of services including cloud servers, storage, Virtual Private Cloud, and CloudWatch. Amazon provides an easy-to-use web service interface which allows users to obtain and configure cloud resources at any of Amazon's AWS data centers. It provides users with complete control over their computing resources and lets users run applications on Amazon's 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.

EC2 provides on-demand resources with pricing depending on the type of resources used and the duration of the usage. The cost of using commercial cloud services also depends on additional factors such as the amount of I/O performed and the amount of storage used, both of which can incur significant costs for researchers using cloud resources. Wang et. al [38] provide a list of example applications that can be executed on Amazon's Elastic Compute Cloud (EC2).

B. Rackspace Cloud

Rackspace Cloud [13] is one of Amzon's competitors in the area of commercial cloud hosting. Rackspace offers services including cloud servers, cloud storage and cloud-based website hosting. Cloud servers are available in eight different sizes (with respect to available RAM and disk space) and support a variety of Linux and Windows operating systems. In [23] and [28], the authors provide a brief description of Rackspace and compare its services with other cloud providers.

C. GENICloud

GENICloud [41], [16] is an open source research cloud testbed which is based on the Slice-Based Facility Architecture (SFA) used by PlanetLab [17]. It supports the management of individual VMs or clusters of VMs. GENICloud uses the Eucalyptus [31] open source cloud platform as a base and federates it with SFA to provide a slice-based architecture to acquire cloud instances (virtual machines) as slivers, similar to acquiring virtual machines on PlanetLab. GENICloud as a platform consists of a small set of nodes at various sites connected internally to provide a cloud testbed for trusted researchers. The GENICloud resources can be acquired using

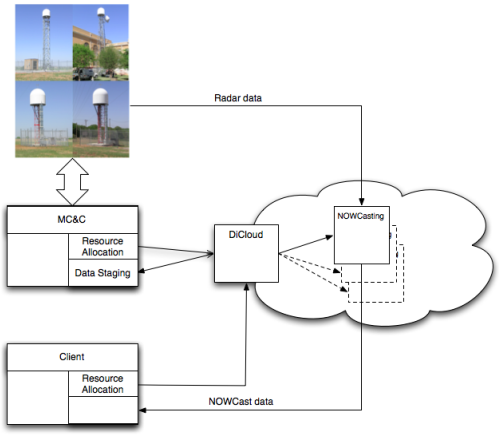


Fig. 1. Overview of the CloudCast system architecture.

the Sface [14] GUI providing valid credentials, or a Web based GUI similar to the one for PlanetLab.

D. ExoGENI Cloud

ExoGENI cloud [4] is a software framework and an open-source cloud platform, which allows users to programmatically manage a controllable, shared substrate. Based on this substrate, researchers can create their own cluster infrastructure by combining servers, storage, and network links in an arbitrary fashion. An ExoGENI deployment is a dynamic collection of interacting control servers that collaborate to provision and configure resources for each experimenter according to the policies of the participants. ORCA (ExoGENI's control framework) helps to provision virtual networked systems via secure and distributed management of heterogeneous resources over federated substrate sites and domains. ORCA allows users to create global topologies of nodes connected via layer 2 QoS-provisioned links. Based on these features ExoGENI cloud offers a variety of opportunities for experimentation and research and also for developing new resource control and management policies via plugins. The ExoGENI cloud, similar to GENICloud, uses a slice-based architecture on top of OpenStack [12] or Eucalyptus [31]. ExoGENI gives researchers more flexibility than other research clouds, as well as commercial clouds, since it allows them to *i)* create their own network topology for a compute cluster, and *ii)* choose between several geographically distributed clusters.

III. CloudCast Architecture

In this section, we describe the architecture of our CloudCast application. We designed this architecture for a new type of radar sensor network developed by the center for Collaborative Adaptive Sensing of the Atmosphere (CASA) [30], [42]. CASA has developed this new type of radar based weather observation systems to better observe the lowest part of the atmosphere and in turn improve the detection and prediction of severe weather. We have chosen CASA since we have full access to its radars (compared to the operational radars run by the NWS) which allows us to evaluate CloudCast with real radars and weather as shown in Section VI. While we evaluate CloudCast by using the CASA system we designed its architecture in a way that will allow its use with other radar networks such as, e.g., NEXRAD.

Figure 1 shows the components of the CloudCast architecture which is composed of the MC&C, described below, which controls the scanning of the radars; DiCloud, an environment that allows users to conduct data-intensive experiments in commercial clouds (Amazon EC2 in this case); and a short term weather forecasting algorithm called Nowcasting.

The CloudCast architecture allows the initiation of Nowcasts in the cloud in two different ways. The first way starts a Nowcast the moment weather enters the radar networks area. To enable this functionality, the CloudCast architecture makes use of Meteorological Command and Control (MC&C). The MC&C includes several detection algorithms that run on radar data and can detect certain weather phenomena (e.g., rain). These detections can be used to initiate a Nowcast in the cloud. The alternative way to initiate a Nowcast is more end-user centric. In this case, a Nowcast process is started if the conditions in the case described above are met and if a mobile user has requested a Nowcast service for a specific area. E.g., one can imagine that a user has subscribed to a Nowcast service and would like to obtain a Nowcast if weather is in the vicinity of their current location. In this case, a Nowcast will only be initiated if the MC&C indicates weather and if a user who has registered for a Nowcast service is in that area.

A. MC&C Architecture

Meteorological Command and Control (MC&C) [42] is the control part of the CASA network that determines how the radars will scan the atmosphere in each 60 second heartbeat. The scanning of each radar is determined by several factors, including 1) detections from data obtained in present heartbeat, 2) historical detections from earlier heartbeats, and 3) end-user policies. The MC&C architecture takes these different factors into consideration to determine how to control the radars. For our CloudCast approach we are making use of the detection features the MC&C offers. For example, rain sensed by the radars will be detected by the algorithms in the MC&C. CloudCast uses the information to determine when to initiate Nowcasts in the cloud. Thus, cloud-based Nowcasts are automatically initiated and halted without user intervention.

B. DiCloud

DiCloud [21] is a platform that enables controlled access to EC2 cloud resources. Though cloud services such as Amazon EC2 provide details of the usage for the operations carried out in the cloud, they perform this usage tracking on a per account basis (which may have multiple users) rather than individual application or user basis. For example, if one envisions that an entity would offer Nowcasting (executed in the cloud) as a service to mobile users, all that Amazon would provide the entity with would be the total cost for the execution of Nowcasts in the cloud. Based on this information it could not be determined how much it costs to execute an individual Nowcast for a mobile user. DiCloud on the other hand enables tracking the

costs that occur by executing individual Nowcasts and, thus, the final receiver of the Nowcast data (E.g., mobile client) can be precisely charged. The DiCloud server communicates with EC2 to track each operation carried out in the DiCloud console. Since the DiCloud console is scriptable, pre-defined scripts can be initiated from the MC&C to start or stop a Nowcast instance in the cloud.

C. Nowcasting

Nowcasting [36], [35] refers to short term (0 - 30min) weather forecasting. Nowcasting is an algorithm for the prediction of 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. Since Nowcasting is a short term weather prediction system, its applications involve warning decision support for detection of potentially severe weather. Its performance is typically measured in terms of categorical yes/no (e.g., 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 where the ability of the Nowcasting algorithm to predict a sufficiently high reflectivity value in a given region is important for end-user emergency decision support.

In order to measure the quality of a Nowcast we make use of three statistical metrics used by NWS to assess warning programs [39]. The three metrics are False Alarm Rate (FAR), Probability of Detection (POD) and threat score or Critical Success Index (CSI) and there definition is as given below:

$$FAR = \frac{\text{false alarms}}{\text{hits + false alarms}} \tag{1}$$

$$POD = \frac{\text{hits}}{\text{hits + misses}} \tag{2}$$

$$CSI = \frac{\text{hits}}{\text{hits + misses + false alarms}} \tag{3}$$

As shown in [36] these metrics can be used to determine the quality of a Nowcast. One way to assess the quality of a Nowcast is to calculate the three metrics mentioned above by comparing a Nowcast for time t with the actual observation at time t. For example, in the case of a five minute Nowcast one would calculate FAR, POD, and CSI by comparing the Nowcast for time t (created at $t-5$ minutes) with the actual observation at t. The actual comparison is performed as follows. A *hit* is defined as the case in which the pixels for both the image created from the Nowcast and the image created from the actual observation are either both active or inactive. Where *active* describes a pixel value that is equal or greater than a predefined threshold level. Similarly, *inactive* is defined as a pixel value being below the predefined threshold level. A *false alarm* is defined as the case where an *active* pixel is presented by the Nowcast image, while an *inactive* pixel is presented by the actual observation image. The opposite case defines a *miss* in the above equations.

In Section V, we use the above metrics to analyze the Nowcasting case study for two different weather conditions for various data aggregation factors in the cloud.

IV. Measurements

In the previous section, we explained the architecture of our CloudCast application in detail. As seen in Figure 1, a potential bottleneck for the real-time operation of Nowcasting in the cloud is the link between the radar nodes and the cloud instances. Hence, in this section we investigate the network feasibility of cloud services for the CloudCast operation. (Detailed description can be found in [27])

In this section, we describe a series of measurements we performed to investigate to what extent the cloud services may be used for real-time weather applications. To perform measurements in a large-scale setting we *replicate* a distribution system that, at least on the network level, is similar to the NEXRAD system. Kelleher [26] et al. give an overview on how data from the NEXRADs is disseminated by using NOAA's NOAANet backbone and Internet2. Since we do not have access to NEXRAD nodes for our measurement, we make use of PlanetLab [17] a global research network that supports large-scale, distributed experiments. According to [42], radars generate data at a constant rate of roughly 5 Mbps. In the rest of the paper, we use 5 Mbps as the minimum required throughput between a radar node and the cloud instances to allow real-time transmission for Nowcasting operation. This threshold can be varied based on the application's need.

A. Experiment

To make our experiment as realistic as possible we went through the exercise of choosing PlanetLab nodes that are close in physical location to the NEXRAD radars. Unfortunately, close proximity between NEXRADs and PlanetLab nodes is not always given due to the fact that locations for weather radars are chosen based on parameters like coverage and beam blocking, which often places them in remote areas. Although there are 159 NEXRAD radar sites in US, we could find only 103 PlanetLab nodes close to those locations, out of which there were around 60 PlanetLab Nodes active at any given time. Hence, in our measurements we use results from around 60 PlanetLab nodes compared to 159 NEXRAD radars.

We have conducted three different measurement scenarios to depict the network capabilities of cloud services considered for our Nowcasting application and any real-time scientific application in general. We conducted a serial measurement where data is transmitted to cloud instances from each individual PlanetLab nodes (replicating the NEXRAD radars). We conducted this measurement to verify how the cloud instances perform without any competing traffic. We also conducted a parallel measurement, where data from all the PlanetLab nodes transmit data to a cloud instance at the same time. We performed 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 operation. Since not all radar nodes around the country would transfer their data to one central instance simultaneously a more likely scenario is the case where a group of radar nodes that belong to a geographic region will transmit their data to a cloud instance that is 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.

B. Measurement Results

The results from the measurements are shown in Tables I and II. 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. Also, to investigate if the time of the day has any impact on our measurement results we perform our measurements twice for each cloud instance, once during the day and once at night time. Approximate time for the day measurement was around noon and for night measurement was around midnight (PST). For these measurements we used Iperf [6] to transmit data.

Table I and Table II give an overview of the average throughput measurement from PlanetLab nodes to the commercial cloud instances and research cloud instances, respectively. As it can be seen from the results of serial measurements row in Tables I and II, 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. Out of the cloud instances we investigated, ORCA cloud instance performs best without competing traffic yielding an average throughput of 110.22 Mbps followed by EC2 East Coast data center cloud instance with 85.03 Mbps, EC2 West Coast data center cloud instance with 36.24 Mbps, Rackspace cloud instance with 35.33 Mbps and then GENICloud instance with a mere average throughput of 9.71 Mbps. We also note that there is not much improvement in the average throughput for the nighttime measurements and the order remains almost the same.

The parallel measurement row shown in Tables I and II provides results that are very different to the serial measurement results described above. ORCA, Rackspace and GENICloud instances yield a better average throughput during the parallel measurement than EC2 cloud instances. ORCA, Rackspace and GENICloud instances yield an average throughput of 17.2 Mbps, 14.12 Mbps and 7.68 Mbps respectively which is greater than the threshold throughput of 5Mbps required for our Nowcasting application. EC2 cloud instances yield an average throughput of 3.14 Mbps and 1.24 Mbps for East and West Coast data center, respectively, which is well below the threshold throughput of 5 Mbps for our application.

The distributed measurement row in Tables I and II shows that each of the cloud instances considered perform better when only a subset of nodes are transmitting data in parallel. 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 ORCA cloud instance performs better than the other three cloud instances with an average throughput of 112.55 Mbps while the Rackspace cloud instance provides an average throughput of 34.15 Mbps. GENICloud instance provides an average throughput of 9.43 Mbps and EC2 cloud service provides an average throughput of 32.46 Mbps and 9.99 Mbps in the East and West Coast data centers, respectively. It can be observed from the measurement results that GENICloud instance is the most consistent irrespective of the number of nodes and transmission scenarios (serial, parallel, distributed) used.

C. Mobile Bandwidth

As mentioned in Section I, it is our overarching 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. Summary

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 can be used as a test instance to execute the Nowcasting application without incurring any additional cost.

The results from the mobile measurement show that the throughput on a 3G wireless link is very volatile. This volatility can lead to the fact that the final Nowcast result is delivered to a mobile client with an additional delay of up to 30 seconds. This is a large delay considering the average data staging and algorithm execution time is around 70 seconds (see Section V).

The measurement results presented in this paper can also be used to verify which cloud services (either commercial or research) offer sufficient network capacity for other applications that require a certain throughput. One example would be a camera sensor network for security or monitoring for which data are transmitted from a set of distributed cameras

TABLE I
SUMMARY OF AVERAGE THROUGHPUT OF ALL MEASUREMENTS IN COMMERCIAL CLOUD SERVICES

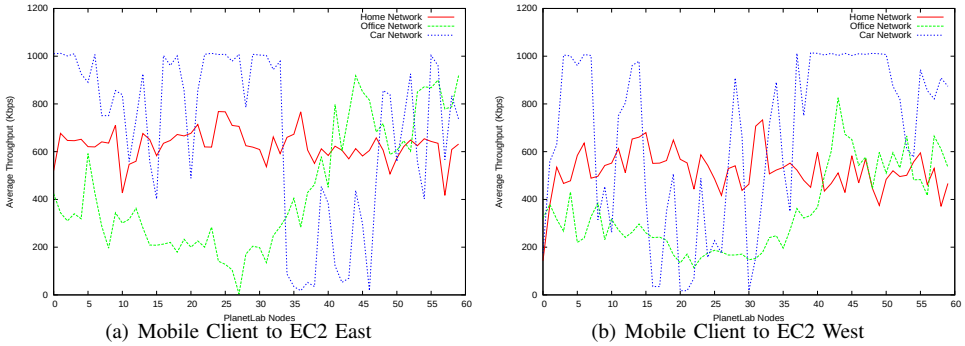
Measurement type	Average (Mbps)						Maximum (Mbps)					
	EC2 East		EC2 West		Rackspace		EC2 East		EC2 West		Rackspace	
	Day	Night	Day	Night	Day	Night	Day	Night	Day	Night	Day	Night
Serial	85.04	86.80	36.25	37.06	35.34	50.66	387.00	360.00	80.40	84.80	134.00	145.00
Parallel	3.15	4.28	1.25	1.06	14.12	12.46	4.87	10.80	10.40	11.20	74.00	43.90
Distributed	32.46	35.26	10.00	9.98	34.16	32.81	240.00	245.00	44.20	64.20	118.00	105.00

TABLE II
SUMMARY OF AVERAGE THROUGHPUT OF ALL MEASUREMENTS IN RESEARCH CLOUD TESTBEDS

Measurement type	Average (Mbps)				Maximum (Mbps)			
	GENICloud		ORCA		GENICloud		ORCA	
	Day	Night	Day	Night	Day	Night	Day	Night
Serial	9.744	9.92	110.22	115.40	52.70	53.00	760.00	764.00
Parallel	7.36	8.46	17.20	41.19	32.80	51.10	48.10	417.00
Distributed	9.43	9.89	112.55	98.53	52.10	52.00	626.00	631.00

TABLE III
PERCENTAGE OF PLANETLAB NODES WITH THROUGHPUT BELOW 5MBPS THRESHOLD.

Measurement type	EC2 East		EC2 West		Rackspace		GENICloud		ExoGENI	
	Day	Night	Day	Night	Day	Night	Day	Night	Day	Night
Serial	12.9%	14.5%	12.5%	18.7%	18.7%	15.6%	15.6%	14.6%	9.8%	8.2%
Parallel	100%	68.8%	100%	97.3%	48.8%	17.7%	22.2%	15.5%	17.7%	24.4%
Distributed	12.6%	16.6%	21.8%	35.9%	15.6%	15.6%	17.1%	15.6%	5.7%	13.4%



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

Fig. 2. Mobile Device to EC2 Bandwidth Measurement

to a central processing node. Assuming the processing would be performed on one of the cloud instances investigated in this paper and the minimum throughput requirement for a single camera stream is known one can simply use the results presented in this paper to determine which scenarios and cloud instances can support such an application.

V. NOWCASTING CASE STUDIES

In the previous section, we investigated the network feasibility of commercial and research cloud services for our CloudCast application. In this section, we present case studies of cloud-based Nowcast generation for two weather scenarios and study the prediction accuracy based on the metrics described in Section III-C. We also investigate how insufficient resources impact Nowcast prediction accuracy.

A. Weather Data

Since the performance of the Nowcasting algorithm may be affected by the type of weather, we have selected two very different cases to evaluate the performance of a Nowcast algorithm that runs in the cloud. The first case from May 10th, 2010 depicts a supercell thunderstorm forming in the CASA radar domain, strengthening as it moves ENE. This storm went on to produce an EF4 tornado in the city of Norman, OK shortly after it exited the field of view of the radars. Supercells tend to be discrete; They rob moisture from their surroundings and precipitation is contained in a relatively small area. The intense forces of a supercell may also result in non-linear movement over time, deflecting right or left as the storm reaches the upper levels of the atmosphere and is affected by the coriolis force. Additionally, supercells generally have slower movement than other types of storms. From the perspective of Nowcasting, the slower movement may lead to reduced spatial error, however it is partially balanced by the difficulties predicting non-linear trajectories.

The second case from May 19th 2010 was a squall line moving west to east across the domain. Squalls are fast moving, long lived, elongated areas of precipitation. They may stretch hundreds of miles and reach forward speeds of 100 MPH in rare cases. Squalls do not generally produce large tornadoes as do supercells, but are known to cause weak spin-ups and straight line wind damage. In this specific case, winds were measured at approximately 60 MPH, which qualifies

TABLE IV
NOWCAST ALGORITHM EXECUTION TIME.

Instance Type	Memory (GB)	Disk (GB)	Cost/hr ($)	Total Cost ($)	Exec. Time (Sec)
EC2 Large	7.5	850	0.34	1.13	74.34
EC2 X-Large	15.0	1690	0.68	2.17	73.77
EC2 High Memory X-Large	17.1	420	0.50	1.63	55.90
EC2 High Memory Double X-Large	34.2	850	1.00	3.54	55.40
Rackspace Large	8.0	320	0.48	1.63	96.53
Rackspace X-Large	15.5	620	0.96	3.22	96.80
Rackspace Double X-Large	30.0	1200	1.8	5.39	96.38
GeniCloud	8.0	20	-	-	67.45
ExoGENI	8.0	20	-	-	56.83

as low-end severe. As squalls are associated with larger air masses, the movement tends to be linear and is thus easier to model, however the fast advection speeds can lead to enhanced spatial errors if network or computation latency is introduced.

We used these *canned* weather data sets for our analysis described in the following sections. The use of identical data sets is necessary to allow for a valid comparison of the performance of different cloud service, since individual weather events can be very different and thus the computational requirements of Nowcasting. In Section VI, we will present results from an experiment where *live* data is used.

B. Cost and Computation Time

In this section, we discuss the measurement procedure to calculate the cost and computation time of the Nowcasting operation for one hour of weather data described in Section V-A with various instance types offered by Amazon EC2 and Rackspace cloud services. We also calculate the computation time of the Nowcasting operation for the same weather data with the instances offered by GENICloud and ExoGENI research cloud services. As mentioned in Section III-B, we use DiCloud as an interface to start the instances, attach storage, and calculate cost of operation for Nowcasting in EC2 cloud service. For each of the EC2 instance types in Table IV, DiCloud is used to bring up the instances with the Nowcasting image, attach EBS storage 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 stored in EBS.

We have carried out this operation for one hour of weather data and with DiCloud's cost tracking mechanism (see Section III-B) we determine the cost for running this 1-hour Nowcast operation. In addition, the execution time for 15 minute Nowcasting of each weather data from the radar in the EC2 instances is measured and the mean is calculated over the whole 1-hour interval. We carry out the same operation on other EC2 instance types and show the results in Table IV. As it can be seen from Table IV, the computation time for Nowcasting on EC2 cloud service decreases as the instance types selected becomes larger and more expensive. The last column in Table IV shows that, the execution time of the 15-minute Nowcast algorithm decreases by $\sim 30\%$ by executing it on a high-performance instance while the cost almost triples. Thus, choosing an appropriate EC2 instance for executing a Nowcast is a trade-off between faster execution time and cost.

Similar to EC2 cloud service, Rackspace commercial cloud service also offers different types of instances as shown in Table IV. We carried out the Nowcasting operation on each of the Rackspace instance types mentioned in Table IV for one hour of weather data and calculated the computation time taken by the Rackspace instances to generate Nowcasts for the weather data. We also noted the cost for Nowcasting operation on Rackspace cloud instance using the web interface offered by Rackspace which allows the users to keep track of their instances.

As it can be observed from Table IV, the computation time taken by Rackspace cloud instances is about 96.50 secs on average to generate 15-minute Nowcasts. Comparing the computation time taken by the two commercial cloud instances (EC2 and Rackspace) to generate 15-minute Nowcasts, it can be seen that the computation time taken by EC2 cloud instances are lesser than those taken by Rackspace cloud instances. Another interesting result shown in Table IV is that with high performance instance types in EC2, the computation time decreases, while the computation time remains about the same on Rackspace cloud service. This observation shows that it might not be beneficial to pay more for high performance instances on Rackspace for computation intensive applications.

We also performed the computation time analysis on research cloud service (GENICloud and ExoGENI) instances for the same weather data used to analyze the computation time of commercial cloud services. The result from our analysis is presented in Table IV. Both the research cloud services offer only one type of instance which is sufficient for the standard operation of Nowcasting application. From the computation time results presented in Table IV for GENICloud and ExoGENI instances, it can be seen that both the research cloud instances take less time (67.45 secs and 56.83 secs respectively) to compute 15-minute Nowcasts compared to the EC2 and Rackspace cloud instances (74.34 secs and 96.53 secs respectively) of the same characteristics. It can also be noted that, ExoGENI research cloud instance is the fastest instance to compute 15-minute Nowcasts in just 56.83 secs compared to any of the other cloud instances considered.

C. Data Aggregation

The results from the measurements described in Section IV shows that the link capacity from the radar nodes and the cloud instances is not always feasible for our Nowcasting operation with certain link throughput always lower than the threshold throughput of 5 Mbps (See Table III). One way to encounter

TABLE V
NOWCAST ALGORITHM EXECUTION TIME FOR LIVE MEASUREMENT

Instances	Memory (GB)	Disk (GB)	Exec. Time (s)	Total Time (s)
EC2	7.5	850	71.98	95.08
Rackspace	8	320	102.48	120.33
GeniCloud	8	20	67.37	78.60
ExoGENI	8	20	56.10	72.07

for the link capacity limitations between the radar nodes and the cloud is to reduce the amount of data through aggregation. In this section, we look at a very simple, lossy aggregation method to reduce the amount of data that has to be transmitted between the radars and an instance in the cloud. The method takes a standard radar moment data file and down-samples it to a specified amount. The down-sampling simply merges a specified number of *gates* in a *radial*. For example, a down-sampling factor of two averages two gates into one and, thus, reduces the file size and the required bandwidth for real-time transmission by a factor of two. Figure 3 shows an example where this aggregation method is applied to single radar data. We observe that even with a relatively high aggregation rate (75% in Figure 3(c)) the visible degradation is marginal.

To analyze the impact of lossy data aggregation with an objective metric, we use FAR, POD, and CSI, described in Section III-C. In the next section, we analyze the impact of lossy aggregation for our two weather scenarios.

D. Lossy Prediction

We now look into Nowcast prediction accuracy from the aggregated radar data for the two weather scenarios explained in Section V-A. We compare the scores of the 15-minute Nowcast data for the aggregated data with that of the non-aggregated data for the two weather scenarios in the following.

Figures 4(a), 4(b) and 4(c) show the POD, FAR and CSI respectively for the 15-minute Nowcasts generated for weather data collected on May 19th 2010. Similarly, Figures 5(a), 5(b) and 5(c) shows the POD, FAR and CSI respectively for the 15-minute Nowcasts generated for weather data collected on May 10th 2010. The metrics are calculated for Nowcasts generated for weather data aggregated up to 128 gates. From the two weather scenarios presented in the Figures, we can infer that the reduction in prediction accuracy is minimal due to radar data aggregation up to 32 gates. From the CSI or threat scores for the two scenarios, we observe that for an aggregation factor of 32 gates, the prediction scores are similar to the non-aggregated radar data, while further aggregation degrades prediction accuracy. Our observation allows us to aggregate radar data up to 32 gates for those radars whose link bandwidth is below threshold, and still maintain high prediction accuracy.

VI. LIVE MEASUREMENT

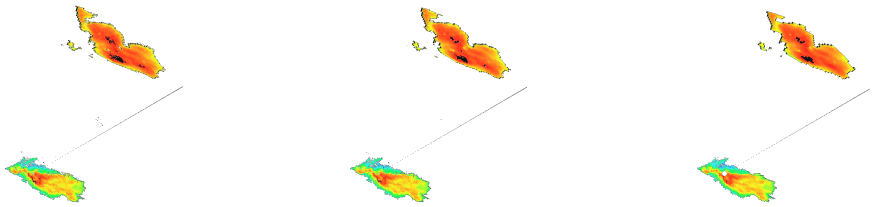
In this section, we present the results from a live, end-to-end measurement that was performed with our own radar on campus as a proof-of-concept for our CloudCast application.

We carried out a live measurement on each of the cloud instances considered to calculate the overall time taken for the Nowcasting process from the time data is generated by the radar, transmitted to the instance executing the algorithm, generating 15-minute Nowcast images and sending the predicted images to a central web server to be used by clients. The overall duration of the sum of the individual steps mentioned above determines how much time a user has between when a severe weather situation is indicated by the Nowcast and when it actually occurs. Obviously, it is the goal to maximize that time interval. For the live measurement analysis we used the data from our own radar on campus which is a prototype CASA radar [30]. Table V shows the result from the live measurement carried out on the cloud instances. The average overall time taken for the whole Nowcasting process was about 95.08 seconds for the EC2 cloud instance, out of which 71.98 seconds were consumed by the generation of 15-minute Nowcasts by the algorithm running on the cloud instance. This means, it takes about 23.10 secs 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, 72.07 seconds, respectively. The overall time taken for the whole Nowcasting process on ExoGENI cloud instance is much lower than the time taken on the other cloud instances considered which is also the case for the time taken only for the generation of 15-minute Nowcasts as explained in the Section V-B.

Figure 6 shows the POD, FAR and CSI metrics (defined in Section III-C) calculated for the live weather data analysis carried out on each of the cloud services. The difference in the values for each cloud instance is due to the different weather that occurred during each live measurement. Since we performed the measurement for each cloud instance at different points in time atmospheric condition can be quite different. In summary, we demonstrated with this live measurement the potential and feasibility of performing short-term weather forecasts for mobile devices in the cloud. From the timing analysis we found that, for 15-minute Nowcasting it takes only approximately 2 minutes to generate the Nowcast images and disseminate it to the client, which leaves the clients with 13 minutes to take any necessary action based on the 15-minute prediction.

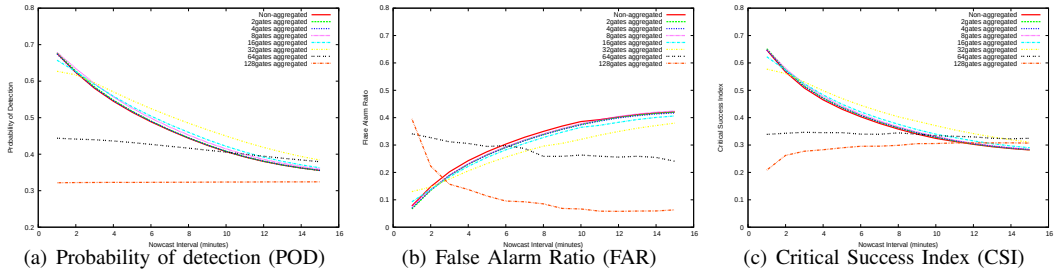
VII. RELATED WORK

A substantial amount of research has been carried out to investigate the feasibility of running scientific applications in commercial clouds such as Amazon's AWS. Hazelhurst examines the performance of the bioinformatics application WCD [20]. Deelman et al. provide details of performance and storage costs of running the Montage workflow on EC2 [18]. The High-Energy and Nulclear Physics (HENP) STAR experiment has examined the costs and challenges associated with running their analysis application in the EC2 cloud [24], [25]. Ramakrishnan et al. have examined the usefulness of cloud computing for e-Science applications [32], [29]. In addition, standard benchmarks have also been evaluated on Amazon EC2. Rehr et al. show that Amazon EC2 is a feasible platform



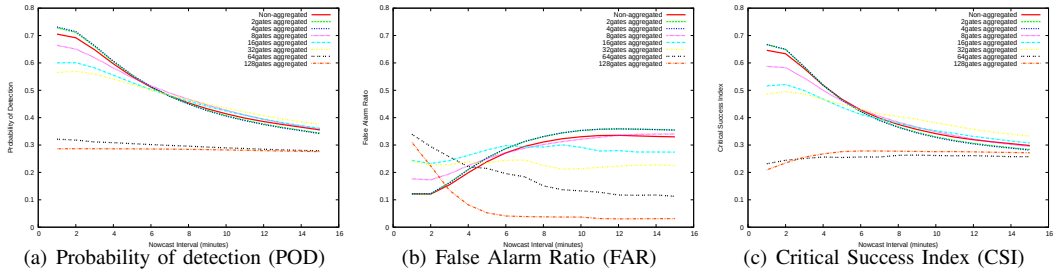
(a) Non-aggregated radar data (b) 50% aggregated radar data (c) 75% aggregated radar data

Fig. 3. Example for simple, lossy radar data aggregation



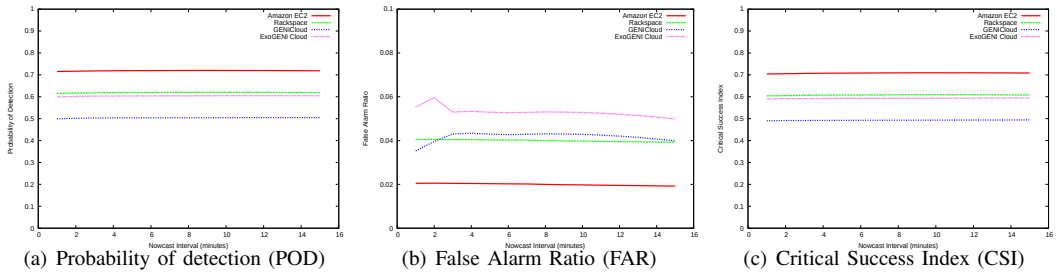
(a) Probability of detection (POD) (b) False Alarm Ratio (FAR) (c) Critical Success Index (CSI)

Fig. 4. Nowcasting analysis with various aggregation factor for weather data from May19th 2010.



(a) Probability of detection (POD) (b) False Alarm Ratio (FAR) (c) Critical Success Index (CSI)

Fig. 5. Nowcasting analysis with various aggregation factor for weather data from May10th 2010.



(a) Probability of detection (POD) (b) False Alarm Ratio (FAR) (c) Critical Success Index (CSI)

Fig. 6. Nowcasting analysis with live weather data on each of the cloud services.

for applications that do not need advanced network performance [33]. Wang et al. [37] study the impact of virtualization on network performance in the cloud. Ramakrishnan et al. perform a comprehensive comparison of the performance of EC2 with HPC platforms, using real applications representative of the workload at a typical supercomputing center [22].

The Nowcasting algorithm used in this paper is based on the Nowcasting system operating in the CASA Distributed Collaborative Adaptive Sensing network. [35], [36] provide details on the Dynamic and Adaptive Radar Tracking of Storms (DARTS) Nowcasting method. The DARTS Nowcasting falls under the area-based Nowcasting methods which estimate a motion vector field over the entire radar coverage domain.

Other examples of area-based Nowcasting methods are presented in Tracking of Radar Echoes by Correlation (TREC) by Rinehart et al. [34], Growth and Decay Storm Tracker(GDST) by Wolfson et al. [40], and McGill's Algorithm for Precipitation Nowcasting by Lagrangian Extrapolation (MAPLE) presented by Germann et al. [19].

To the best of our knowledge, the work presented in this paper is the first to look into the feasibility of commercial and research clouds for real-time application of weather forecast-

ing and the first work to introduce a mobile application for short-term weather prediction in the cloud.

VIII. Conclusion

In this paper, we present CloudCast, a mobile application for personalized short-term weather forecasting. CloudCast uses instances from commercial and research cloud services to execute a short-term weather forecasting application which is based on a Nowcasting algorithm. We demonstrate the network feasibility of using cloud services for our CloudCast application by performing a series of measurements.

We also compare the compute feasibility of Nowasting in the cloud with real weather data on various instance types offered by cloud services. We calculate the computation time and the cost to generate 15-minute Nowcasts in the cloud. Our results show that computation time for generating 15-minute Nowcasts reduces by $\sim 30\%$ if executed on a high-performance instance, but with an almost 300% higher cost than a low-performance instance in EC2 cloud service. It can be observed from the results that ExoGENI cloud service provides lower computation time to generate 15-minute Nowcasts compared to other cloud services considered. We also performed a live experiment with our CloudCast architecture based on data from a weather radar that is located on our campus. The results from our live measurement show a very high prediction accuracy whereas the delay between data generation at the radar to the delivery of 15-minute Nowcast image to a mobile client is less than 2 minutes on average.

We have shown that commercial and research cloud services are feasible for the execution of our real-time CloudCast application. 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.

References

[1] AccuWeather. http://www.accuweather.com/downloads.asp.

[2] Amazon's Elastic Compute Cloud. http://aws.amazon.com/ec2/.

[3] Clouds Can't Move Fast Enough for Weather Service. http://www.hpcinthecloud.com/blogs/Clouds-Cant-Move-Fast-Enough-for-Weather-Service-100353244.html.

[4] ExoGENI. http://wiki.exogeni.net.

[5] Internet2 ION. https://geni-orca.renci.org/trac/wiki/flukes.

[6] Iperf. http://sourceforge.net/projects/iperf/.

[7] MetService Rejects Cloud. http://computerworld.co.nz/news.nsf/technology/metservice-rejects-cloud.

[8] My-Cast. http://www.digitalcyclone.com/products/mobile-my-cast/.

[9] NEuca. https://geni-orca.renci.org/trac/wiki/NEuca-overview.

[10] NLR Framenet. http://www.nlr.net/framenet.php.

[11] NOAA Supercomputer Solicitation. https://www.fbo.gov/utils/view?id=cc031d6f3b30b8e6f872e4e53136c851.

[12] OpenStack. http://www.openstack.org.

[13] Rackspace Cloud. http://www.rackspace.com.

[14] Sface. http://svn.planet-lab.org/wiki/SfaceGuide.

[15] The Weather Channel App. http://www.weather.com/mobile/.

[16] A. C. Bavier, M. Yuen, J. Blaine, R. McGeer, A. A. Young, Y. Coady, C. Matthews, C. Pearson, A. Snoeren, and J. Mambretti. Transcloud - Design Considerations for a High-performance Cloud Architecture Across Multiple Administrative Domains. In *CLOSER*, May 2011.

[17] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-coverage Services. *SIGCOMM CCR.*, July 2003.

[18] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The Cost of Doing Science on the Cloud: The Montage Example. In *SC*, November 2008.

[19] U. Germann, I. Zawadzki, and B. Turner. Predictability of Precipitation from Continental Radar Images. Part IV: Limits to Prediction. *Journal of the Atmospheric Sciences*, December 2006.

[20] S. Hazelhurst. Scientific Computing Using Virtual High-performance Computing: A Case Study Using the Amazon Elastic Computing Cloud. In *SAICSIT*, October 2008.

[21] D. Irwin, P. Shenoy, E. Cecchet, and M. Zink. Resource Management in Data-Intensive Clouds: Opportunities and Challenges (invited paper). In *Workshop on Local and Metropolitan Area Networks*, May 2010.

[22] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. Wasserman, and N. Wright. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. In *CloudCom*, December 2010.

[23] H. Jin, S. Ibrahim, T. Bell, W. Gao, D. Huang, and S. Wu. Cloud Types and Services. In *Handbook of Cloud Computing*. Springer US, 2010.

[24] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. In *CCA*, October 2008.

[25] K. Keahey, T. Freeman, J. Lauret, and D. Olson. Virtual Workspaces for Scientific Applications. *Journal of Physics: Conference Series*, 2007.

[26] K. E. Kelleher, K. K. Droegemeier, G. Qualley, J. J. Levit, C. Sinclair, D. E. Jahn, S. D. Hill, L. Mueller, T. D. Crum, S. D. Smith, S. A. Del Greco, S. Lakshmivarahan, L. Miller, M. Ramamurthy, B. Domenico, and D. W. Fulker. Project CRAFT: A Real-Time Delivery System for NEXRAD Level II Data Via the Internet. *Bulletin of the American Meteorological Society*, 2007.

[27] D. K. Krishnappa, E. Lyons, D. Irwin, and M. Zink. Network Capabilities of Cloud Services for a Real Time Scientific Application. Technical report, University of Massachusetts Amherst. Available from http://server.casa.umass.edu/žink/cloudmeas.pdf.

[28] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: Shopping for a Cloud Made Easy. In *HotCloud*, June 2010.

[29] J. Li, M. Humphrey, D. Agarwal, K. Jackson, C. van Ingen, and Y. Ryu. eScience in the Cloud: A MODIS Satellite Data Reprojection and Reduction Pipeline in the Windows Azure Platform. In *IPDPS*, April 2010.

[30] D. McLaughlin, D. Pepyne, V.Chandrasekar, B. Philips, J. Kurose, and M. Z. et al. Short-Wavelenth Technology and the Potential for Distributed Networks of Small Radar Systems. *Bulletin of the American Meteorological Society (BAMS)*, 90(12):1797–1817, December 2009.

[31] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System. In *CCGRID*, May 2009.

[32] L. Ramakrishnan, K. R. Jackson, S. Canon, S. Cholia, and J. Shalf. Defining Future Platform Requirements for e-Science Clouds. In *SoCC*, October 2010.

[33] J. J. Rehr, J. P. Gardner, M. Prange, L. Svec, and F. Vila. Scientific Computing in the Cloud. *ArXiv e-prints*, Dec. 2009.

[34] R. Rinehart and E. Garvey. Three-dimensional Storm Motion Detection by Conventional Weather Radar. *Nature*, 1978.

[35] E. Ruzanski, V. Chandrasekar, and Y. Wang. The CASA Nowcasting System. *Journal of Atmospheric and Oceanic Technology*, May 2011.

[36] E. Ruzanski, Y. Wang, and V. Chandrasekar. Development of a Real-time Dynamic and Adaptive Nowcasting System. In *Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, January 2009.

[37] G. Wang and T. Ng. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. In *INFOCOM*, March 2010.

[38] L. Wang, J. Tao, M. Kunze, A. Castellanos, D. Kramer, and W. Karl. Scientific Cloud Computing: Early Definition and Experience. In *HPCC*, September 2008.

[39] D. Wilks. *Statistical Methods in the Atmospheric Sciences*. Elsevier, 2nd edition, 2005.

[40] M. Wolfson, B. E. Forman, R. G. Hallowell, and M. P. Moore. The Growth and Decay Storm Tracker. In *Aviation, Range, and Aerospace Meteorology*, January 1999.

[41] M. Yuen. GENI in the Cloud. Master's thesis, University of Victoria, 2010.

[42] M. Zink, E. Lyons, D. Westbrook, J. Kurose, and D. Pepyne. Closed-loop Architecture for Distributed Collaborative Adaptive Sensing: Meteorogolical Command & Control. *IJSNET*, February 2010.