NSFCloud Workshop Report

K. Keahey, R. Ricci,
A. Apon, R. Arpaci-Dusseau, P. G. Bridges, E. Eide, J. Lange, W. Lloyd, A. Majumdar, J. Mambretti, G. Ricart, S. Shetty, W. Smith, and M. Zink

Table of Contents

1. Introduction

The NSFCloud solicitation was established in 2013 as a track within the NSF CRI program with the goal to provide an experimental platform enabling the academic research community to drive research on a new generation of innovative applications of cloud computing and cloud computing architectures. While this effort seeks to build on prior NSF investments, such as the GENI and FutureGrid testbeds, it is unique in that it targets specifically the field of cloud computing and seeks to support and integrate research in this area with innovate solutions in systems, networking, and cyber-physical system (CPS) research and thereby define a broad research agenda for the future.

The NSFCloud workshop was conceived as a vehicle to initiate a dialogue between the research community and the Chameleon and CloudLab projects funded under this initiative, describing the expectations on one side and capabilities on the other. Accordingly, in the first part of the workshop the NSFCloud projects gave extensive presentations to the community describing their planned capabilities; in the second part of the workshop the attending community members reciprocated by providing feedback on the required capabilities in a series of lightning talks as well as via breakout sessions discussing specific and detailed requirements for Computer Science research in cloud computing. This purpose of this report is to summarize that feedback.

The workshop was organized by the PIs of the two funded NSFCloud proposals, Kate Keahey and Rob Ricci, and was held in the Waterview Conference Center in Arlington, VA on December 11-12, 2014. On the first day of the workshop, after presentations from the NSFCloud projects, the workshop included three keynotes and a series of lightning talks describing upcoming challenges and infrastructure required to support research on them. Overall, the workshop was attended by close to one hundred researchers. Participation was based on the submission of position papers that were selected based on their ability to describe testbed requirements corresponding to current and challenging research problems, as well as striving to ensure diversity over different research topics and expected modes of testbed usage. All artifacts presented on the first day, including the agenda and presentations, the proceedings (containing all the position papers submitted to the workshop), video recordings, and list of participants are available on the workshop website at www.chameleoncloud.org/nsf-cloud-workshop.

The second day was spent in a series of breakout sessions discussing research community requirements in detail. The first set of breakouts examined the requirements across different areas containing disruptive factors for cloud computing research; they included work on low-level system software, networking, cyber physical systems, and distributed systems. The second set of breakouts examined cross-area issues including research methodology, educational requirements, and the needs of innovative applications. Each area was assigned a lead and a scribe and provided detailed individual reports. This report presents the findings from those breakouts, including the requested hardware and software capabilities, or any other capabilities deemed interesting by the attendees.

This report is organized as follows. Sections 3 through 9 contain reports from the breakouts as presented by leads and scribes, unedited save for small formatting changes. Section 2 contains the summary of collective insights produced by the breakouts. Section 10 concludes the report.

2. Summary of Findings

This summary was produced by extracting specific requirements from the reports of the breakouts in Sections 3 through 9, grouping similar or related requirements into categories, and then summarizing the combined focus of the groupings. In order to preserve traceability of individual requirements to the original breakout summaries the summaries have been cited as appropriate. For this purpose breakouts were labeled as follows: systems [S], networking [N], cyber-physical

systems [CPS], distributed systems [D], research methodology [M], education [E], and applications [A].

2.1. **Hardware**

This section summarized requirements for experimental hardware.

Diversity of hardware. Almost all working groups pointed to the need for a *diversity of hardware* in experimental testbeds [S,CPS,D,E,A] as it is key to enabling a range of different experiments that study how a given software system behaves across generations of hardware, and in different performance regimes [D]. The variety should accommodate processors of different types (e.g., x86 versus ARM) [S,D], accelerators of different types (e.g., GPUs and FPGAs) [S,CPS, A], different network solutions (e.g., Ethernet, IB) [S,CPS,D], and storage systems (e.g., single disk, RAID, flash-based SSD) [S,D]. Different processor types could be used to study energy trade-offs [D], accelerators are important for applications like encryption and fMRI [CPS], while different networking solutions are essential to study low latency [CPS,D] and different storage solution to study I/O performance [D]. The hardware should also be geographically diverse/distributed [D]. To support research in energy efficient and energy agile clouds the ability to completely spin down disks was requested [CPS].

Cutting-edge character of the hardware. Some groups [S,A,CPS] pointed out that a testbed intended for Computer Science needs to include hardware that is *cutting-edge* so that the research performed on it can be impactful [A]. In addition, testbeds should include a strategy emphasizing the integration of newer hardware [CPS].

Scale and type. Some working groups commented on the need for large-scale hardware [S,N,CPS,E]. The hardware should support scale sufficient for large-scale experimentation [S,N,E] used not just in research but also education [S,E], data-intensive applications [N], and large flows for e.g., applications operating on fMRI data [N,CPS] to support scalability studies. [S] points out that current NSF large-scale testbeds either do not provide deep reconfigurability (e.g., XSEDE) or do not provide hardware that is sufficiently modern to be interesting for research (e.g., PRoBE).

Hardware instrumentation. Some working groups noted that certain types of instrumentation critical to Computer Science research are only available on hardware level and not easily available to users in typical infrastructure [S,CPS,M]. This includes power and thermal metrics [S,CPS,M], as well as network measurements [M].

Programmability of networking hardware. The networking hardware was called out as essential to supporting next generating capabilities, especially *programmability* [CPS,N]. The hardware/firmware needs to support at least some version of Software Defined Networking (SDN), the newest version of OpenFlow, or alternative approaches to support research on programmable networks and latency [N,CPS].

2.2. **Base Testbed Capabilities**

This section summarized requirements for experimental hardware.

Specification, discovery, and verification. Most working groups noted that to enable CS research the information about the testbed must be very *fine-grained, complete/full, and up-to date* [S,N,M,A]. Users should have access to complete [S] and fine-grained information at the level of BIOS versions [S,A], disk models [A] and even serial numbers of individual components [M], essential to know to determine if hardware has been replaced. Users should also have access to information about the testbed capacity, for example network capacity to support experiments [N]. Further, once an environment is created for a researcher, software is needed that verifies the environment to the level of detail needed by the user to make sure that they get the resources they asked for (servers, images, networks, storage, and so on) [M].

Resource allocation. The breakouts articulated three important properties of resource allocation: the need for *on-demand* allocation of resources [CPS,M,E,A] and generally allocation time management, *fine-grained* control over which resources are allocated [S,N,D], and the need for *allocating isolated partitions* of the testbed [N,D,E].

On-demand provisioning of resources is needed to respond to urgent events [CPS], to provide availability at the beginning of a class [E], or respond to demand fluctuations [A]. While the definition of "on-demand" may vary across different applications (seconds may be needed for CPS applications and up to a minute or two may be tolerated by a class) the consensus is that some form of on-demand availability is needed. [M] noted that in the case of large allocations users realize that they will have to wait for the resources to be available but will still want on-demand provisioning for small resources. In addition, [E] noted the need in tesbed allocation mechanisms to provide resources for a semester-long class and [A] noted the need for mechanisms that would allow unused cycles to be offered to applications on a best effort basis.

Fine-grained control allowing precise selection of node and network allocations [S,N,D] is required both for control (i.e., ensuring that enough resources of the required type are available for the experiment) and ensure reproducibility. [M] notes that at a high level, users may want to simply select resources from resource classes, in other situations, researchers will want to specify detailed characteristics of resources (e.g. a server with a specific type of hardware), and finally, researchers may want full control by selecting specific resources depending on the model and type of experiments performed.

The requirement for *strong isolation* is critical for both (1) control over experimental conditions and repeatability and (2) for preventing interference/disruption from other users. [D] noted the need for resources not to be shared in any way and also noted that weaker form of isolation might be useful in some cases when detailed contention information is available. [N] and [E] noted the need for the users to work in a "breakable environment", important especially for students to experiment without hurting anybody else's work.

Configuration. Virtually all groups noted the need for easy and deep reconfigurability at bare metal level, including reconfigurability of systems-level components [S,CPS,N,D,M,E,A]. The fundamental need of a CS testbed is to be able to map a variety of environments such as cloud configurations [D], lightweight virtualization solutions such as Tiny Core Linux VMs [CPS], and a range of generic images for research and education [M,E]. The characteristics of such system-level infrastructures combined with the need for isolation and control mean that *only bare metal reconfiguration* meets all required needs. Furthermore, in some cases researchers need access to *deep reconfigurability* of low-level components of the system such as BIOS settings [S] or network switches [N].

Instrumentation and monitoring. Most breakouts pointed to the need for instrumentation and monitoring capabilities for the testbed [S,N,CPS,D,M,A]. The testbed should support and make available *deep instrumentation at different levels of granularity* [S,N,D], in particular deep monitoring of networking information [N] including precise measurements of latency, queuing, and jitter as well as network delays [CPS,A]. [S,M] pointed out that some of this data (including network traffic, power, and heat) is not available from the operating system of the servers allocated to a user and thus testbeds should provide ways for users to access this information. Further, testbeds should be able to *aggregate and store* this instrumentation information in large-scale databases [S,N,D] and where possible make this information available to tools that would allow easy use of this information and in some cases could even "replay" testbed conditions (including transient errors and all) [N,D,A].

Support for access at large-scale. A few breakouts [N,CPS,E] noted the importance of providing access to thousands of users or clients, especially important in [CPS] where large numbers of

clients may connect to the cloud via cellular or wireless access networks. [E] noted the need to support a large number of requests in a short period of time, e.g., near the due date of an assignment.

2.3. **Higher-level Tools and Services**

This section summarizes higher-level tools and services requested by the breakouts.

System images and image management. Almost all breakouts voiced the need to make a *broad range of diverse system images* (variously referred to as system images, profiles or appliances) available to the user community [S,CPS,D,M,E,A]. This would serve two purposes: (1) so that users can stand up a diverse set of environments or software easily and focus on the algorithms and research they are going to develop [S,D,A], and (2) so that it is easy to reestablish the same environment for reproducibility purposes [M,A]. In addition, some groups pointed to the need for tools capable of converting system images from one format to another, e.g., from virtual machine (VM) images to bare metal images to transition development carried out in VMs into bare metal parts of the testbed where experiments can be conducted [M], or between image formats supported by different providers to ensure portability or to "cloud burst" from the testbed to commercial providers [CPS,E].

Experiment Management tools. Some groups [N,D,M,A] noted the importance of experiment management tools that would allow researchers to describe how testbed resources, system images, and other tools were combined for a specific experiment and automatically enact or re-enact the establishment of such configuration over the testbed. [D,M,A] in particular emphasized that such tools would allow researchers to efficiently evolve and scale an environment and provide ease of use by automating much of the effort currently expended on experiment preparation. [N] also pointed to the need of easy to use "templates" for experiment enactment. [A] pointed out that the ability to thus formalize and re-instantiate an experimental configuration is essential to reproducibility and also suggests that such re-instantiation might for example include injecting faults or re-enacting faults recorded during the original experiment.

User support services and community development. Most groups noted the need for high-quality user support features [S,N,D,M,E]. Specific basic services that were noted as needed were ticket/trouble management system, access portal, documentation, forums for community discussions (email lists, community/discussion forums for different communities within the testbed, social networks), and a longer-term process for addressing any missing platform features [S,D]. The testbeds should also develop *a rich set of mechanisms allowing the research community to share various experimentation and education artifacts*. This includes sharing of workloads, traces, system images, course materials, experiment configurations and data, associating such configurations and data with publications, providing notes to collaborators, benchmarks, sharing techniques for performing data analysis, instructor training sessions, and sharing of tips and tricks in general [D,M,E]. Those shared artifacts should include both materials provided by testbed operators and contributed by community [E]. [E] also outlined a need (that could complement a testbed) for integrated course management console that would support student accounts, resources, grading, and related educational activities in an integrated way.

Experiment Dataset repository. Several groups [S,CPS,M,E] expressed interest in support for an experiment dataset repository that would include datasets that could be shared among the community. Those sets would consist of e.g., application data (road traffic FMRI, climate, etc.) as well as system level data (power, temperature, networks) [CPS]. [M] noted that while ideally those datasets would be available publicly some researchers may want to protect their data while the research should be conducted so that different levels of sharing should be supported.

Workloads & traces repository. Most groups pointed to the need of associating the testbed with a repository of workloads and traces [S,D,M,A] which should be discoverable, contributable and

sharable [D]. To be most useful, traces must describe the configuration the system they were gathered from in detail. In addition, the traces should describe the type of usage the system supports (e.g., experimental versus production) [M]. Both [M] and [A] pointed out that traces of the testbed itself could be useful to replay testbed conditions.

Fault injection tools. Some groups [CPS,D,M] emphasized the need for controlled variation via inserting compute, disk, network (latency and jitter), and other relevant faults into the system, to explore how these parameters affect the system. Some felt that the testbeds should provide a selection of tools and data in relevant formats to simulate failures in order to allow research on fault tolerant cloud architectures [CPS] while others noted that in practice researchers already have their own [M]. [A] noted that the ability to support such controlled variation is critical for reproducibility.

Special services. Some groups expressed the need for special services. [N] highlighted the need for federation of CLoudLab, GENI, and FIRE as well as testbeds including mobile and wireless. [N] also pointed to the need to extend the testbed to include additional types of resources with networking requirements such as power grid radio telescopes, sensors and instruments. Both [CPS] and [E] highlighted the need to potentially "cloud burst" from the testbed onto other infrastructures, such as commercial infrastructures, which in addition to image compatibility mechanisms described above requires also resource management services capable of orchestrating such cloud burst.

2.4. **Cross-Cutting Concerns**

Two cross-cutting concerns, for reproducibility and ease of use, were voiced throughout the workshop. While the specific mechanisms proposed in this context were already discussed in the sections above, here we discuss the concerns themselves.

Reproducibility. Most groups highlighted the ability to control experimental conditions and obtain reproducible results as one of the largest issues in realistic systems research [N,D,M,E,A] with [A] even stating that reproducibility and predictability was the top concern of the working group. Many challenges and solutions in this sphere (such as e.g., the need to upgrade hardware – but in a managed way so that the use is aware of even the smallest upgrades) have already been discussed above in the context of specific testbed services. In addition, [A] noted that reproducibility concerns go beyond the NSF Cloud facilities as users may want to capture and control configurations of end-user behavior, federated systems, data provenance, network routing tables, etc. In addition [D] pointed out that there may be different levels of reproducibility (e.g., "workload" which allows an exact replay of a particular workload, "system" which enables the precise re-creation of a system and its environments, "performance" which re-creates the exact performance of a workload on a system) so that it will be necessary to quantify the different levels of reproducibility in a given experiment (similar insight was formulated in [M]). Further, [D] also advocated for providing an association between published results and the capability to reproduce them.

Ease of use. Some groups [N,D,E] explicitly called out the need for ease of use. While some users are savvy experimenters with strong systems experience, others may be first-year students with little systems expertise. The testbeds should be structured such that both groups can be accommodated and in addition provide easy to use "on ramps" so that inexperienced users can easily get started and transition to more demanding and more complex usage as they go. Here too, specific suggestions, in particular in the area of system image management, experiment management tools, and user services have already been discussed in the context of specific services. [E] additionally highlighted the need for exceptionally good documentation with accurate instructions on how to use the environment.

2.5. **Policy**

The main policy recommendation made by workshop attendees [N,E] was that NSFCloud resources should strongly prioritize use by experimental Computer Science. It was noted that experimental computer Science has strong requirements in particular deep reconfigurability, isolation, instrumentation, and time-controlled allocation mechanisms, as articulated in the requirements above. NSFCloud resources will represent unique capabilities answering those requirements that are not available from other NSF facilities or commercial providers. The guidance was therefore that if some activities could be conducted satisfactorily on public clouds [N] or other mid-scale NSF platforms [E], they should not be using the NSFCloud facilities, which should be reserved for experiments that require those resources.

3. System Software Breakout Summary

This section contains a summary of the NSFCloud Workshop discussion on node-level system requirements and recommendations. The breakout session was led by Patrick G. Bridges, University of New Mexico; Jack Lange, University of Pittsburgh, was serving as a scribe. The participants were Alex Blate, University of North Carolina at Chapel Hill; Steve Crago, USC Information Sciences Institute; Eric Eide, University of Utah; Rajasekhar Ganduri, University of North Texas; Charles Leiserson, Massachusetts Institute of Technology; Rob Ricci, University of Utah; Weisong Shi, NSF.

3.1. **Description of the Area**

This breakout focused on the computing infrastructure needs of researchers working primarily on node-level system software. Each of the participants described the need for scalable testbeds in their research areas. This includes research in subareas such as:

- **OS/VM design, implementation, and optimization** – Research in this area focuses on designing next-generation system software for high-performance scientific system, large-scale data processing systems, and general cloud computing infrastructure. This includes work on new performance isolation mechanisms for cloud systems, optimizing virtual machine access to communication and networking facilities, and supporting emerging hardware capabilities.
- **Node-level performance characterization and engineering** – Research in this area focuses on measuring and optimizing system performance, particularly of applications running on the system. This research must take into account the underlying system software and hardware platforms, which is a significant challenge on modern cloud systems. This area also includes a significant educational component, as performance characterization and engineering is an important educational area.
- **Heterogeneous computing platforms** – Research in this area focus on designing system software to support novel computational resources, including Graphical Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs), and many-core processors in cloud systems. These technologies are computationally powerful, but difficult to share in cloud modern systems and high-performance systems. New system software techniques that support sharing these resources would significantly increase the reach and impact of large-scale cloud and scientific computing systems.
- **Power and energy management** – Power and energy management system software research focuses on matching system power draw to application needs. Power is an over-constrained resource in modern large-scale systems, and techniques that improve the allocation and usage of this resource has high impact. Research in this area is currently generally conducted, however, on small-scale isolated testbeds.

- **System-level resource allocation –** Finally, research in system-level resource allocation focuses on global allocation of resources, including processor, memory, and energy across the entire system. This research area includes both heuristic, analytic, and numerical techniques for optimizing system resource allocation Because of the large-scale nature of cloud systems, these are challenging problems that require large-scale testbeds to evaluate.

Breakout participants in each of these areas focused on determining the hardware, software, and management capabilities they needed from NSFCloud systems to support their research.

3.2. **Hardware Capabilities**

The most important hardware capability that the breakout identified was bare-metal access to a wide range of diverse, modern hardware. In terms of diversity, breakout members identified the need for a low-level access to a wide range of processor, accelerator, storage, and networking hardware; this diversity is important in each research area because of its emerging importance in next-generation systems. Current large-scale cloud testbeds, however, either do not provide low-level access to this hardware (e.g. NSF XSEDE systems) or are older generation hardware (e.g. NSF PRoBE systems) that do not include these resources.

In addition, breakout group members also emphasized the importance of appropriate configuration information, documentation, and instrumentation of this hardware. For reproducibility, this information includes details at the lowest level, for BIOS versions and settings). Having full hardware specification and documentation available is also important, but because it is essential for low-level system research and such documentation is frequently difficult to get for individual researchers. Finally, the systems should be instrumented to provide performance, power, and thermal information; this information is essential in each of the research areas described above.

3.3. **Software Capabilities**

Most of the software capabilities the breakout identified related to system allocation, configuration, and control. In particular, breakout members noted that it is important to have software that can capture complete system information including BIOS version and settings for an allocation. On current systems, this information varies machine to machine, and support software does not generally fully capturing the configuration of the system. This impedes work on resource allocation and performance engineering. Ideally, researchers would also be able to allocate specific hardware nodes to be able to repeat experiments exactly, and have fine-grained control over node and network allocation decisions when possible, though breakout members recognized the challenges inherent in this.

In addition, general software support should include common workloads, system images, and data along with any appropriate licenses. Some current NSF testbed systems include some of these capabilities in terms of system images that can be leveraged. However, larger issues of licensing and workloads are not as well addressed and equally important to enabling system software research.

Finally, several breakout members suggested having large-scale instrumentation databases that aggregated long-term performance information about the system. This information would allow researchers to conduct larger longitudinal performance studies that current systems do not support. Breakout members recognized the privacy issues that such databases would present.

3.4. **Support Capabilities**

In addition to hardware and software requirements, breakout members also pointed out the need for high-quality support features. In particular, they noted the need for a ticket/trouble management system, forums for community discussions, and a longer-term process for

addressing any missing platform features and capabilities that were initially missed. The first two of these (ticket/trouble systems and forums for discussion) are broadly provided in current NSF testbeds; the last issue, a longer-term process for community input into evolving platform features and capabilities is an area in which NSFCloud systems could improve on the current state of the practice.

3.5. **Conclusions**

Breakout members stated that some NSF mid-scale infrastructure systems such as the NSF PRoBE systems partially meet their needs, though not with the diversity and scale proposed in the NSFCloud systems. Overall, breakout members felt that the systems proposed by the two NSFCLoud teams already would meet many of the requirements described above. Some features, for example BIOS-level access, specialized hardware, and power/thermal instrumentation features were close but would not completely support some of the experiments that researchers were interested in. This was particularly true for performance engineering and power/thermal management research.

4. Networking Breakout Summary

This section is a summary of the NSFCloud Workshop discussion on network requirements and recommendations. The charge for the Network Session 1 breakout group was to identify the networking requirements for NSF cloud project. The breakout session well attended – the group included the following participants, Theophilus Benson, Magda El Zarki , J. Bryan Lyles, Eman Mahmoodi, Amitava Majumdar, Joe Mambretti, Sachin Shetty, Martin Swany, Brian Kelley, Rob Ricci, and Zhenhai Duan.

4.1. **Description of the Area**

Participants described a number of enabling applications that could drive specific networking requirements. The participants noted that almost all of the "Lightning Talk" presentations referenced requirements for experimental networking, including topics related to architecture, programmable networking (Software Defined Networking - SDN, etc), network measurements, wireless networks and networking issues arising from challenges in mobility in general as well as related subjects. Many presenters discussed the importance of having access to testbed capabilities for supporting experimentation on large scale data intensive science applications (in addition to enterprise and consumer applications), which implies having network capacity for supporting high volume data flows.

Participants noted the importance of providing support for many types of experiments through federations among testbeds. For example, they noted that it would be important to have a federated integration between Chameleon and CloudLab to allow users to take advantage of resources at multiple sites. A federation with the Global Environment for Network Innovations (GENI, as well as FIRE – the EU Future Internet Research environment - and related testbeds) was also considered important, as well as federations with other testbeds including other mobile and wireless testbeds.

Enabling experiments by extending the testbed environment to include additional types of resources for various applications with specific networking requirements was another topic discussed. Some examples that were identified and discussed included Cyber Physical Systems (CPS), such as power grid, radio telescopes, sensors, and instruments. The appropriate blend of networks and systems for these projects is an interesting issue.

A common requirement among these driver applications is the need to provide appropriate connectivity of devices, including instruments, to the NSF cloud. In addition to connectivity, these applications will require specific computing, bandwidth and latency requirements of different orders of granularity. The hardware and software capabilities to support the experiments

for these applications would include an ability to appropriately interface with different types of devices, including instruments, and ability of the NSF Cloud interface to support deep measurements of networking experiments. To enable some experiments it will be necessary to identify multiple networking requirements, including bandwidth needs and reservation/scheduling mechanisms related to connecting instruments to the NSFCloud testbeds along with identifying computation requirements. Additional experimental research areas discussed included

- **Personalized Networking** – The goal of such investigations would be to determine methods to provide applications with the ability to customize network to their own specific needs. The users may want to configure network settings directly or provide requirements so that the settings could be configured for them. Options for both approaches should be available. A question asked was: could libraries be developed for this area?
- **Networking Adaptable to Applications** – The goal of these investigations would be investigating areas that would minimize manual adjustments at the network edges – shift of automatic configurations to network (e.g., for high quality digital media), for example multi layer path selection and implementation. It was noted that edge users and applications have to undertake the burden of many manual adjustments at the edge to compensate for lack of automated adjustments within the network fabric.
- **Data Center Research** – Data centers are rapidly evolving as a key infrastructure resources, both to support extremely large numbers of users and global enterprises. Data center networking is an important area of research as is private WANs interlinking global data centers.
- **Capability of NSF Cloud to Handle User Intensive Applications**– Questions were asked about the network resources required to support applications with very large user bases, especially those that are highly geographically dispersed. Experimenters should know apriori the ability of the underlying computation and networking resources to support the large user bases. What would be the constraints when the applications have 100s of thousands of users (or millions of users)? How would the experimenter know about the ability of the underlying computation and networking resources to support such large user bases? Is it possible to identify common networking requirements for suite of applications? How many applications with heterogeneous requirements can a single switch handle today? How much capacity would be available for experimentation in this area?
- **Wireless, Mobility, and NSF Cloud** – Mobility is a major research topic. Wireless applications have requirements include the ability to aggregate different wireless links, latency bounds and to explore the synergy between SDR and SDN. Other experiments would investigate the current understanding of implications of wireless network delay and computation delay. Mobility-Cloud research topics include: computational offloading, tradeoffs between energy and delay, next generation LTE and WIFI networks, aggregation of the links in the cloud, adaptive wireless delay and remote processing delay, and others.
- **Network Instrumentation** – The NSF Cloud interface should support deep measurements of networking experiments. A need exists for tracing, recording and especially providing for (ensuring) the repeatability of experiments (Real science experiments must be replicated). Sufficient granularity related to resources, measurements, and reproducibility is currently not provided by public clouds to gain meaningful insights. Hopefully, the NSF Cloud can provide different levels of granularity as required by different experiments.

- **Public Cloud Services on NSF Cloud** – Access to load balancers and security services are available on AWS and should be available on the NSF Cloud. At the same time, having low level access to switches, which can allow changing OS and switching OpenFlow versions should be possible. It was noted that if experiments could be conducted satisfactorily on public clouds, they should not be using the NSFCloud facilities, which should be reserved for experiments that require those resources.
- **Accessibility to Large User Base –** Ability to provide access to an online service on NSF Cloud to 100s of thousands of users to use the service, including at city scale, e.g., US Ignite cities, etc. Synergistic opportunities were recognized between Us ignite activities and the NSF Cloud.
- **Emulation for Difficult Problems**: It is difficult to emulate some behaviors, such as latencies, but it is important to accomplish this.
- **Quick & Easy Access** – Providing easy to use on-ramps, e.g., via templates, was recognized as a requirement
- **Benchmarking** – It was noted that no benchmarking standards exist today for clouds – should there be such standards? If so, where should such standards be developed? Who should develop them? What is the quantified meaning of cloud performance? What are the baselines? What are the algorithms that allow for comparisons among clouds?

4.2. **Hardware Capabilities**

The network hardware required must support next generation capabilities, especially programmability. Given that the environment is being designed to support research experimentation, a distributed platform must be provided to researchers that is free from infrastructure constraints, especially standard production network constraints. The platform must be a "breakable environment." The GENI project provides a model for this approach. However, more effort is required to limit the restrictions of underlying production networks. The network equipment should be based on programmable/highly configurable components, including SDN/OF, but also including alternative approaches. Access over non-standard paths is required, e.g. a "Science DMZ" architectural approach. Switches should be able to support a wide range of heterogeneous requirements.

4.3. **Software Capabilities**

Software is an increasingly key resource, driven by Software Defined Networking (SDN), Software Defined Infrastructure (SDI), Software Defined Systems, Software Defined Components (e.g., Software Defined Radio), etc.

Various software components are required, including an access portal, mechanisms for finding resources (e.g., an RSpec system), integrating resources, accessing resources (e.g., credentials for use), control frameworks for managing resources, monitoring and tracking experiments, etc. Capabilities should be provided to the network equivalent of cloud "bare metal" access. An emerging standard is OpenFlow 1.3, which should be supported, with plans for migration paths to OF 1.4…OF 1.n, 2.0, etc.

4.4. **Capabilities Addressed by Existing Infrastructures**

GENI, FIRE and related experimental research network testbeds have been in operation for several years. An expectation is that basically network oriented research would still use these testbeds, while research that was basically cloud oriented would use the Cloud lab environment. However, it is important to note that increasingly the demarc between these two types of resources is an artificial distinction, both environments use an integration of both resources. For example, all GENI racks support a cloud infrastructure – each rack contains a small cloud.

4.5. **Conclusions and Highlights**

Experimenters conducting empirical research require a large scale highly distributed highly programmable environment that can support many different types of experimentation and within which they can integrate many types of resources, including compute facilities, networks, storage systems, sensors, instruments, CyberPhysical systems, mobile devices, and much more. Networks do not constitute a standalone resource but a fully integrated component within such a distributed environment. Diversity of resources, in part, can be achieved through testbed federations. Programmability is enhanced through macro architectural trends contributing to new levels of virtualizations, including Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), Anything-as-a-Service (XaaS), etc.

5. Cyber-Physical Systems (CPS) and Real Time Breakout Summary

This section contains a summary of the NSFCloud Workshop discussion on CPS and real-time system requirements and recommendations. The breakout session was led by Mike Zink (University of Massachusetts and CloudLab). Warren Smith (TACC and Chameleon) served as scribe. The participants were: Joseph Kizza (University of Tennessee at Chattanooga), Ting Zhu (UMBC), Glenn Ricart (US Ignite and CloudLab), Sekou Remy (Clemson), Mario Gerla (UCLA), William Beksi (University of Minnesota), Arun Ravindran (UNC Charlotte), Muthuramakrishnan Venkitasubramaniam (University of Rochester), and Paul Flikkema (Northern Arizona University).

5.1. **Description of the Area**

The major motivations of the participants can be categorized in three areas:

1. Using clouds to realize cyber-physical systems with real-time constraints.
2. Very interested in how to build a cloud to support (1), but not necessarily working on cyber-physical and real-time systems.
3. Using the cloud for domain specific research.

5.2. **Required Capabilities**

The major discussion points that were addressed during the breakout session are described in the following:

1. *Data set repository:* To support cloud research in the area of cyber-physical and real time systems members of the breakout session thought that it would be useful to have data sets that could be shared amongst the community. In addition, it might be useful to have a repository where these data sets could be stored. To be more precise such sets could, e.g., consist of application data (road traffic, fMRI, climate, etc.) in addition to system level data sets (power, temperature, networks). Researchers could then use a combination of these data sets for their own work on new cloud architectures.
2. *Quick provisioning (seconds) to respond to "physical" events:* One requirement that was expressed during the discussion in the breakout session was the ability to bring servers (both virtual and bare metal) up very quickly (on the order of seconds). This would be important for CPS applications were the need for computation cannot always be planned ahead very well but might be needed instantaneously, if required. This also implies that research clouds would perform a mechanism to (temporarily) deactivate current, lower-priority work if sufficient resources are not available. Potential starting points to address this requirement would be the provisioning of central storage (potentially as SSD) in the cloud data centers and the ability to change hypervisor priorities (e.g., by bringing up an image that is preloaded into memory). In addition, this might require a policy change in how resources in the research clouds are acquired.

3. Latency & jitter investigations (e.g. robotics): Latency is a critical factor for cyber-physical systems, especially for the ones that have real-time requirements. Therefore, it is necessary to investigate how low latency requirements can be fulfilled by the research clouds. To what extend can SDN (in this specific case OpenFlow) and specialized networking hardware support these requirements? In addition, tools to measure and diagnose where in the cloud the latency/jitter is added are required (e.g., which machines, networks, software, etc.). Also tools to inject latency and jitter on the network, compute, disk i/o level would be useful to explore how changes of these parameters would affect cyber-physical and real-time systems and the applications that use these systems.

Besides these three major requirements the participants of the research area suggested a series of additional features that should be offered by the research clouds:

- Experimenters should have the ability to move large data (e.g. fMRI data) into the cloud at a rate of 30 MB/S or higher.
- The research clouds should provide GPUs (and potentially FPGAs) for applications like encryption and fMRI. In addition, it would be important to provide an approach that would allow the integration of newer hardware (that will become available in the next few years) into the clouds.
- To support research in energy efficient and energy agile clouds the ability to completely spin down disks was requested.
- Tools to simulate failures should be provided to allow research on fault tolerant cloud architectures.
- It should be possible to interact with public clouds. E.g., one could imagine a cloud burst from the research cloud in a public cloud if massive amounts of resources are required.
- The support of mobile cloud applications where large amounts of clients are connected to the cloud via cellular or wireless access networks.
- For certain applications like the simulation of a vehicular network representing millions of cars it would be important that the research clouds would support tiny VMs (e.g., such as the ones based on Tiny Core Linux) with limited computational power but the potential to run very large numbers of these VMs in the cloud.
- For data center management it would be critical to ingest external information like cost of power, outside temperature, etc.

6. Distributed Systems Breakout Summary

This section contains a summary of the NSFCloud Workshop discussion on distributed system requirements and recommendations. The breakout session was led by Remzi Arpaci-Dusseau and Wyatt Lloyd, and contains insights from the working group.

6.1. **Description of the Area**

The distributed systems working group discussed a range of topics that will be relevant to distributed systems researchers using CloudLab, Chameleon, and similar research infrastructures. The overall goal of providing this type of research infrastructure is simple: to enable said researchers to perform their science at scale, making it easy to reproduce and share the work that they do.

In this document, we present the major points of discussion of the group, which focused on requirements: what do distributed systems researchers need from the underlying platform to enable their work? We focused on five major points: workload, platform diversity, control, ease

of use and development, and community. We now present each of these focal points, highlighting important facets of our discussion.

6.2. **Required Capabilities**

Workload. One of the most pressing items for many of the researchers in the room was **workload**: the applications, traces, and benchmarks used to evaluate the systems they build. The research platform should make it easy to: (a) discover relevant and popular workloads, traces, and benchmarks (WTBs), (b) create new and interesting WTBs, and (c) share newly-created WTBs with other researchers. Doing so will amplify the work of scientists; instead of spending significant research time on finding suitable workloads for evaluation, the researcher can instead spend the time building their systems and analyzing them with relevant and interesting workloads.

A related point in the space of workload focuses on **reproducibility**; the platform should provide mechanisms to aid scientists in their ability to reproduce experiments. At the heart of science lies this fundamental need: to be able to read about an innovation, and then, in a different experimental setting, re-create the innovation. Shared research infrastructure should provide mechanisms to enable such reproduction. As there may be different levels of reproducibility ("workload" which allows an exact replay of a particular workload, "system" which enables the precise re-creation of a system and its environments, "performance" which re-creates the exact performance of a workload on a system), it will also be necessary to quantify the different levels of reproducibility in a given experiment.

Diversity and Uniformity. The needs from the platform itself dominated much of the discussion of the group. We focused on numerous important topics: diversity, control, fault injection, and isolation levels, and now present more details on each in turn.

A critical need from distributed systems is **hardware diversity**: different types of CPU, network, and storage system. Having different hardware components enables a range of different experiments that study how a given software system behaves across generations of hardware, and in different performance regimes, a key to understanding the scaling properties of a given system. For example, having different types of CPU (x86, ARM) enables one to study energy trade-offs; different networking gear (Ethernet, Infiniband) allows for the analysis of the importance of low-latency; various storage systems (single disk, RAID, flash-based SSD) provides researchers with the opportunity to understand the importance of I/O performance in their system.

A related need is on the software side: the ready ability to install different operating systems, virtual machine stacks, and other critical software components, is also quite useful. This **software diversity** will enable researchers to determine important dependencies of their higher level systems on lower-level pieces of software, and to study their interplay, thus enabling the study of which properties lower-level systems should provide to enable the ready construction of various distributed systems on top of them. For example, in cloud research, there are a growing number of open-source virtual machine environments (OpenStack, Eucalyptus); being able to easily instantiate any one of these options gives researchers the ability to study the importance and interactions with each.

A third requirement in platform comes in the form of **geographic diversity**. While some systems only work well within local clusters, others are designed to span the country and world in their operation (think Google's Spanner); the platform should provide support for running and experimentation with this type of wide-area system.

As related to the last point, while geographic diversity is useful for a range of wide-area systems research, it would be useful to have some **limited forms of uniformity** across data centers. While the software and hardware diversity is useful in many cases, it also would be helpful to provide

the necessary hardware and software across sites to enable a researcher to run on similar platforms across an otherwise geographically diverse system. Thus, while providing heterogeneity across data centers is useful for many of the reasons listed above, it should not be the only aspect considered when building said platforms.

Control. An important aspect of any experimental platform is control: what types of knobs, settings, and configurations does the system provide? We focused on three different aspects of control: resource choice, fault injection, and isolation.

One of the most important needs for distributed systems is **resource choice**: having the precise ability to specify the exact resources needed for a given set of experiments. This might include choice of host, placement of hosts within a datacenter rack as well as across different data centers, and precise selection of networking and storage resources (specific network configurations and storage demands). Having simple (and programmatic) methods to enable such selection is essential. For example, being able to control the creation of a multi-site virtual data center with certain node characteristics as well as specific networking connectivity between the machines within and across sites would be of great utility.

Another critical need is to support a rich and diverse set of **fault injection** tools. Perhaps the most important aspect of a distributed system is its ability to detect and recover from faults. Thus, the platform should provide the ability to insert machine, disk, network, and other relevant faults into the system, enabling the researcher to study their system under these conditions. To the extent possible, the platform should enable realistic fault scenarios, mimicking what happens in real-world datacenters and distributed systems.

One final level of control relates to **isolation level**: what different barriers are placed between experiments running within the platform? Of course, a key need for many researchers is for strong isolation: the platforms should all provide access to bare metal resources that are not shared in any way. This type of isolation enables careful and detailed analysis of systems in ways that a less strongly isolated platform cannot. However, some weaker forms of isolation could also be useful; in some cases, simply making detailed contention information available could allow researchers to better utilize and share hardware while still enabling scientific experimentation to proceed.

Development and Usage. Another important, but sometimes overlooked, aspect of a shared research platform is its ease of use for those developing systems atop it. While sometimes these users are savvy experimenters, sometimes they are first-year graduate students, new to the world of experimentation and analysis. Making the research platform easy to use (and easy to use correctly and well) is thus essential.

The most basic requirement along these lines relate to **ease of experimentation**. How easy is it to compose a simple experiment on small-scale resources? Once put together, how easily can such an experiment be re-run? How can the system enable a frictionless transition from small-scale to large-scale experimentation and analysis? Are there different modes of experimentation available, e.g., interactive when running to learn what is important, and batch when all the pieces are in place and data must be gathered at scale?

A major aspect of usage and development relates to **instrumentation and monitoring**. How readily does the system enable tracing across different resources at different granularities? How easily can information from different sources be collected and integrated? What kind of tools are in place to trace workloads and systems, and archive the results? Given the platforms themselves are fascinating high-end systems, can researchers readily study the platforms, thus making the infrastructure a topic of analysis as well?

Community. A last requirement for any shared research platform for distributed systems are the mechanisms, policies, and guidelines around building a community. Much of the value of shared platforms is that they enable different groups of researchers working on similar problems to share workloads, experiments, analytical tools, and other aspects of their research, thus benefitting each group and accelerating research.

To help build community, the platform should provide a diverse and rich set of **mechanisms for sharing.** Researchers should be able to readily share experiments (how to run a particular workload, inject a particular fault, and measure the particular outcome of a given system), applications and workloads (how to replay a given trace, workload, or benchmark against a different implementation of the same API), and even course materials (how to enable students to work on the same course project across institutions, with the same testing and grading infrastructure). Indeed, a simple mantra that arose from the group was **share everything**.

A related point revolves around other aspects of any community, which are more the **guidelines and expectations for usage.** What incentives are in place to ensure users behave well? What can typical users expect from the system, in terms of what types of resources they can get, and when they can get them? While some systems infrastructure is undoubtedly needed to help encourage good behavior, it would also be useful to think about establishing community norms and expectations, through email lists, online discussion forums for users, and other means of social interaction that can also help shape user behavior.

6.3. **Summary**

Distributed systems researchers are one of the primary users of shared distributed research facilities such as CloudLab and Chameleon. We have identified a number of day-one requirements, including workload, platform diversity, control, ease of use, and building of community. While these represent a solid collection of initial thoughts, as systems researchers know, more can only be learned through the usage and evolution of the platforms over time, as it is built and put into deployment.

7. Research Methodology Breakout Summary

The objective of this breakout session was to discuss research methodology and the support that users of experimental testbeds, such as NSFCloud, need to perform rigorous experiments. There were approximately 20 participants in this session with many of the participants performing systems research, several participants that perform applications research, and participants that are building experimental infrastructure. The session was led by Warren Smith; Eric Eide served as scribed.

7.1. **Hardware Capabilities**

This session did not focus on identifying the specific types of hardware needed for the experiments the workshop attendees want to perform. However, a few general hardware capabilities were discussed.

An important part of performing an experiment is being able to measure properties of interest during the experiment. While many of these properties are available from the servers allocated to a user, other properties require hardware support to measure and are outside of the environments typically visible to users. Examples are network measurements as well as rack-level measurements such as power usage and temperature.

Another important feature of experimental infrastructures is the ability to reproduce experiments. This requires that experimental environments must be able to be described in detail. To support this, hardware must be able to describe itself, even down to component serial numbers so that users know if, for example, failed components have been replaced.

Another area that was discussed was the ability to inject load and faults into an experimental environment. Specialized hardware is needed for some types of load and fault injection, but none of the session attendees indicated a need for it in our very brief discussion. The researchers in the session also indicated that they do not need load or fault injection software because they have their own software that they would use.

7.2. **Software Capabilities**

The discussions in this session mainly focused on the software infrastructure that researchers need to perform experiments. The session therefore identified several different capabilities that should be provided by the software infrastructure of cloud testbeds.

Participants noted that there is a lot of work to perform before an experiment can be run. Researchers need software infrastructure so that they can efficiently evolve an environment, scale the environment, and prepare for their experiment. While researchers understand that they will need to wait to get access to large pools of resources, they want on-demand access to small-scale resources so that they can iterate quickly before they begin to scale. They also want easy and quick access to resources, for example virtual machines. They then wish to be able to easily transition to physical environments. Perhaps by being able to convert virtual machine images to node images, perhaps by adopting low-overhead container technologies like Docker, perhaps by the use of environment configuration tools like ansible, puppet, chef, or similar tools.

Researchers need software so that they can request that an experimental environment be configured for them and so that they can retrieve information about the environment used by a past experiment. The level of detail of these descriptions varies depending on the needs of the researcher. At a high level, the ability to simply select resources (e.g. servers) from classes and apply environments to those resources by identifying them (e.g. images) is useful in many situations. In other situations, researchers want to specify detailed characteristics of resources (e.g. a server with a specific type of hardware) and environments (e.g. a starting image and a configuration procedure). Finally, researchers may want full control by selecting specific resources (by identifier) and exact environments (e.g. OS install image and install parameters). After an experiment is complete, descriptions of the environment at these same different levels of detail are again needed in different situations.

Once an environment is created for a researcher, software is needed that verifies the environment to the level of detail needed by the user. Users expect a basic level of verification built in to the infrastructure – that they get the resources they asked for (servers, images, networks, storage, and so on) and that their requested environment is operating correctly on those resources. Much deeper verification could be performed, but there is a tradeoff between the thoroughness of the verification and the time to perform it. There is also a choice between performing more general verification and performing experiment-specific verification. For example, testbeds could provide a set of general benchmarks to verify performance or a researcher could re-run an experiment that they have already executed. In both situations, the results would be compared to previous results to verify that the environment is performing as expected.

While experiments are running, software is needed to monitor and measure the experimental environment. Monitoring ensures that the testbed services needed by the experiment are operating correctly and measuring records data related to the experiment. There is a variety of experiment-related data that is of interest to researchers. An important point is that some of this data (e.g. network traffic, power, heat) is not available from the operating system of the servers allocated to a user and testbeds should provide ways for users to access this information. In addition to making detailed information about experimental environments available to the researchers performing those experiments, the session had a discussion about providing data to other users and publicly. Such data can be useful, but there are concerns about making data available to

others before a researcher can complete their work and publish. This concern may be mitigated if complete data isn't available, such as which users were using which resources at any given time.

One area that the researchers in this session did not express an interest in was having the testbeds provide tools for simulation and emulation. The feedback from the users that they already have their own ways of doing this and one of the things they want to do is use cloud testbeds to verify their simulations and emulations.

7.3. **Other Capabilities**

The research methodology session discussed several capabilities that are not exactly hardware or software capabilities. The capabilities described in this section are better thought of as services or interfaces.

The participants in the session were very interested in the availability of traces, workloads, and data sets. Traces are useful for comparison and understanding and workloads and data sets are useful for driving experiments. To be most useful, traces must describe the configuration the system they were gathered from in detail. In addition, the traces should describe the type of usage the system supports. The type of usage is important because traces gathered from nonproduction environments can be misleading. While the NSFCloud testbeds are not production cloud infrastructures, traces from these testbeds provide detailed information about experiments and can be used to compare extended or related experiments run on the testbeds.

One of the brief discussions in the session was about data storage and analysis. Researchers will clearly need to store experimental data for analysis. However, in general, the researchers do not require support to analyze their data as they have their own analysis tools and experts.

The participants in this session were very interested in d

7.4. **Existing Capabilities**

Many of the capabilities described above are available in current testbeds or other infrastructure and services. There is significant overlap in the hardware and software capabilities described above with the capabilities needed by academic, government, and commercial data centers so there are many building blocks that cloud testbeds can integrate to create easy-to-use environments. There are also a variety of web-based tools for collaboration and sharing that could be incorporated into testbed web portals.

7.5. **Conclusions and Highlights**

Cloud testbed users require support in a variety of areas so that they can effectively prepare and run rigorous experiments. This session identified a number of these areas such as describing and verifying experimental environments, monitoring and measurement, data storage, and the availability of trace and workload data to drive experiments.

Interestingly, researchers spent a significant amount of the session describing the support they want before and after an experiment. They want to be able to evolve and scale an experiment in an efficient manner with on-demand access to resources and virtual machines and then easily transition to physical machines to run experiments. They want to be able to collaborate across user projects and share information so that all users can make better use of the testbed. Finally, after experiments are complete, they want to be able to share information about their experiments with others.

8. Education Breakout Summary

A group of college-level educators met in the NSF Cloud Education Breakout. Participants included Amy Apon, Patrick Bridges, Charles Leiserson, Renato Figueiredo, Joseph Kizza, Amr El Abbadi, Mike Zink, Jay Akut, Rob Ricci, and Remzi Arpaci-Dusseau. All participants

expressed keen interest in how NSF Cloud may be able to support their activities in teaching, primarily in undergraduate computer science.

8.1. **Description of the Area**

The needs of the education community include support for several types of experimental computer science educational activities. Examples of teaching and learning activities that can benefit from NSF cloud include:

- VM development
- Eucalyptus deployments
- Cloud design classes
- Classes that use FPGAs/Accelerator/specialized hardware
- Large classes with high latency requirements
- Performance engineering, and
- Data center networking classes.

8.2. **Required Capabilities**

The group observed that experimental educational activities place high demands on the experimental environment in several types of use cases, including:

Individual undergraduate student or small group special research projects. These types of projects can be very similar to research done by more experienced researchers, except that students tend to need more guidance than experienced researchers about how to access a system, and may become frustrated more quickly when the environment does not work exactly as expected. Key requirements include exceptionally good documentation of how to use the environment and the ability to provision a very large number of resources for a short period of time in order to execute scalability studies.

Undergraduate classroom exercises. In a classroom setting, the start and end of the class time are unchangeable. Use of experimental resources in this setting requires that the system be up and available at the start of class and remain available throughout the class period. All students have to be able to access the system for the experience to be truly successful. The system environment that is needed for the experiments needs to be ready when students login, or needs to be provisioned very quickly, in order to maximize the time for the actual experiment. A key requirement is that the experimental environment needs to scale to support large classes and a large number of small jobs. The system must be robust to errors in code written by students.

The needs for graduate student classroom exercises are similar, except that graduate students can often be expected to access the environment and complete the exercise outside of the class period much more independently.

Undergraduate homework exercises. The experimental environment for undergraduate homework exercises needs to be provisioned quickly and instructions need to be accurate. Graduate students are more tolerant of errors and glitches in the environment. For both undergraduate and graduate students the environment needs to scale to support a large number of requests in a short period of time, say, near the due date of an assignment. A key requirement is the ability to provision an environment for a long period of time, say, a semester, to allow for homework exercises and course projects to build and extend each other throughout the course. The environment needs to be predictable, and measurable, and to support repeated experimentation.

A summary of the basic features that were identified that would help to meet the needs of these educational use cases includes:

- Isolation – so students can break things without hurting anyone else
- System level hardware access, e.g. VM-level access, introspection
- Predictable, measureable environment
- Minimal setup time, good startup documentation

The resource allocation requirements for meeting the needs of educational activities include:

- Short turn-around allocation time
- Longer-term resource allocation for semester courses
- Sufficient capacity to handle class deadlines, capacity
- Support for scalability exercises
- Hardware Needs and Software Capabilities

In general, no specific hardware needs were identified, except that variety in available hardware will support more types of educational activities.

Educators called out the need for support for collaborations, and for community development and interactions. Software capabilities, learning materials, and educational processes that would support collaboration and community interactions include:

- Interfaces similar to other systems used for education, e.g. GENI
- Lessons learned from GENI
- Portability between NSFCloud/commercial cloud services like AWS
- Shared materials/modules/exercises, both from the facilities and contributed by community
- TA/instructor training sessions
- Community forums for educators
- Good data sets, benchmarks, course modules for educational exercises
- Tie in to SIGCSE, GENI, SC, NSDI etc. related educational events, conferences, planning
- Community events for NSFCloud community (perhaps co-locate with something else? SC, NSDI BOFs?)

A "blue sky" request by educators is the development of an integrated course management console that would support student accounts, resources, grading, and related educational activities in an integrated way. Features of this console might include:

- The ability to control/dole out resources
- Provision starting images, update management
- Testing/grading support
- Hierarchical scheduling and access support (faculty/TA/student):

- Support for team vs. individual allocations
- Account monitoring capabilities
- Control of errant student jobs
- Instructor management of scheduling/deadlines/allocations
- Capabilities Covered by Existing Infrastructures

Nearly all participants in the Education Breakout indicated that they have used various commercial clouds, such as Amazon AWS or Microsoft Azure, in their teaching activities. Most had found these to be generally acceptable. However, certain disadvantages of these systems were called out, including the cost or wait times for experimental access for a large group of students or for large experiments, and lack of access to bare metal. Some resource allocation requirements (e.g., provisioning an environment for a full semester) were difficult to do for some types of exercises, say, requiring a large amount of experimental data, on commercial clouds.

Many participants had also accessed XSEDE or campus computing clusters for experimental activities. Since XSEDE and campus clusters are production environments for computational science, only application-level experimental activity is possible, and often access at scale is limited.

One observation that was expressed in the Education Breakout was the possibility that demands for NSF Cloud by the research community might trump the demand for the educational community. The opinion was mixed about whether there should be some type of set-aside allocation of NSF Cloud for educational purposes. On the one hand, many educational activities can be well-supported by commercial clouds or campus clusters. On the other hand, certain educational activities, such as teaching about cloud design, cannot be supported by these existing infrastructures. However, these types of special requests may also be able to be handled through a normal allocation process. The guidance from the Education Breakout participants is that NSF Cloud should focus on things that cannot be run on commercial or other mid-scale NSF platforms. That is, experimental educational activities should leverage access to XSEDE, AWS, etc., where possible, and use NSF Cloud resources where they are unique.

8.3. **Conclusions and Highlights**

Participants in the Education Breakout were uniformly enthusiastic about the potential teaching and learning activities that can be enabled by NSF Cloud. They are supportive of NSF's funding of these projects and optimistic about the future possibilities for computer science education. Educators are eager to build an experimental computer science educational community that leverages NSF cloud.

9. Applications Breakout Summary

This section reports on the result of the Application Breakout led by Glenn Ricart (version of 2/9/2015).

A brainstorming breakout was held to think about the kinds of novel applications that Chameleon and Cloudlab might enable and to discuss the unique features that may be necessary to support these applications. Most all of the results reported below attracted positive comments from more than one person, but we didn't take any votes or require a majority in order for an idea to make this report. An exception is the "Reproducibility" request which occupied almost half of the time of the breakout and featured many, many speakers in favor.

9.1. **Description of the Area**

Killer Apps. The breakout first discussed "killer apps", applications which would best take advantage of the NSF Cloud capabilities and their differentiation from commercially-available services.

1. Anything Geospatial: Applications which take advantage of spatial diversity in their computation would seem to be well-suited to creating corresponding geospatial clouds in CloudLab and Chameleon. Examples might include vehicular data analysis and traffic management. The NSF Cloud facilities are particularly differentiated if the requirement is for real-time, critical, federated, distributed, global, or hierarchical infrastructure. In these cases, NSF Cloud can be configured to provide for the requirements, and the research areas involved are likely to be quite fruitful.
2. NAS Benchmark: Any applications whose structure resembles the NAS benchmarks for highly parallel architectures is likely to run quite well on the NSF Cloud facilities compared to commercial facilities because of the inter-component communications configured on Chameleon and CloudLab.
3. Anything that Requires Inherent Security: CloudLab and Chameleon can be configured to run applications in an inherently more secure way than commercial clouds because each application can be given its own isolated "cloud" for execution. The related area of mobile-to-cloud security should be a fruitful area for research on the NSF Cloud facilities.
4. Healthcare and other Sensitive Information Applications: Because of the capability for allocating an isolatable cloud for a set of sensitive applications, CloudLab and Chameleon appear to be superior cloud infrastructures for maintaining data security. There are research challenges in extending this isolation to federated clouds, to real-time tasks, and to critical tasks for which high reliability is also needed.
5. Publishable Systems Research: A significant community advantage of CloudLab and Chameleon is the ability to run codes in different cloud configurations greatly advancing publishable systems research. For this purpose, it will be important to completely document the configuration on which each trial was run so that it can be later reproduced.

Differentiation with the Traditional HPC. The Chameleon and CloudLab capabilities will differ significantly from traditional high performance computing (HPC) systems in that CloudLab and Chameleon will have superior capabilities for:

1. Large numbers of real or simulated end-users
2. Real user interaction
3. Real-time
4. Geospatial optimization
5. Tiered or hierarchical clouds (e.g. transportation traffic management at global, regional, and local levels being considered in a hierarchy of clouds)
6. Continuous operations (versus HPC batched or pre-parameterized runs)

9.2. **Requested Capabilities**

The group made a number of requests to the CloudLab and Chameleon teams.

1. Reproducibility and Predictability: The number one request was to add additional features to help with reproducibility and predictability of results. One of the largest issues in realistic systems research is to be able to control experimental conditions and

obtain reproducible results. Results of cloud configurations on any given task are highly dependent on the exact components and their interconnections, and there are challenges when the environment is dynamic allocation of resources from an overall meta-cloud or cloud-pool. New revisions of hardware or software, even when the model number has not changed, can have an effect. It's not reasonable to never upgrade the hardware or software in order to be able to reproduce results exactly over long periods of time. But we should do what we can.

2. Log the exact configuration down to BiOS settings ad disk models so that detailed configuration information is available for each cloud allocated and executed.
3. Log external interactions if possible including network delays. In some cases, being able to replay network interactions (transient errors and all) is desired.
4. Archive "profiles" of clouds so that it's easy to re-invoke the "same" cloud later for reproducibility.
5. Provide the ability to re-instantiate the exact same configuration that was previously used. Perhaps the logging information should be in a form which can be re-submitted to allocate the exact same configuration.

Because faults and delays can trip up reproducibility, the capability of inserting deterministic faults and delays is needed. The determinism may include the seeds used for random number generators used for dynamic fault injection.

Reproducibility concerns go beyond the NSF Cloud facilities. Users would also like to capture and control configurations of end-user behavior, federated systems, data provenance, network routing tables, etc. There is a recognition that one could go too far, but the voice of the room was that we haven't tried to go nearly far enough.

Variation. The flip side of reproducibility and predictability is controlled variation. Users would like to be able to keep an allocated cloud configuration exactly the same but be able to do a sensitivity analysis as to the impact of: changing parallelism, processor speed, disk speed, network speed and configuration and so forth.

Rapid and Dynamic Provisioning. It would be useful to have Chameleon and CloudLab be able to dynamically alter provisioning to accommodate external events as would be common in cyberphysical systems and the Internet of Things.

Dynamic provisioning would also be useful for demand fluctuations and to accommodate algorithm changes or phase changes in an algorithm.

There was also significant demand for being able to utilize "scavenger" cycles that would otherwise be idle. The scientific world is replete with modeling that could be tried with yet another set of parameters or inputs if only the cycles were available. Do CloudLab and Chameleon lend themselves to this purpose? Can we learn something about clouds in this way?

Support for Real-Time Clouds. Commercial clouds are not capable of helping systems researchers determine the sources of delays, latency, and queueing. Support for capturing this information within the CloudLab and Chameleon facilities is requested. Logs will be helpful to see what happened.

Real-time clouds are also desired to permit researchers to shadow production environments with realistic real-time data and compare results to existing production.

Cross-Layer Communication and Adaptation. Applications can optimize themselves better when they understand the physical resources on which they are running. A trivial example is

matching parallelism to cores actually available. So Chameleon and CloudLab should make information about the exact facilities currently allocated available to the application.

Likewise, information from the application may be useful to CloudLab and Chameleon to adapt and optimize their support for that particular application.

Performance Knobs. Researchers would like to see standardized performance knobs they can twist to observe their impact on application performance.

Hardware. Researchers would like to see the CloudLab facilities make available new, cutting-edge, and research hardware so that the impact and benefit can be assessed, and architectures and codes adapted to take advantage of the newest hardware. This includes GPUs and FPGAs.

The datacenter research crowd would like to see an expanded set of datacenter hardware facilities such as WAN optimizers and load balancers.

Software. The end-user community would like pre-packaged or curated libraries of software that allow them to focus on the algorithms and end-user purposes of the applications. These curated libraries will help end-users stay above the infrastructure when they want to do so. Examples include:

- AAA (authentication, authorization, accounting)
- ABAC (attribute-based access control)
- Vetted libraries (e.g. "Technology Insertion Service")
- Toolboxes

The reverse is true as well. Some infrastructure researchers would like standardized or canonical use cases they can try out on different infrastructure configurations.

10. Concluding Remarks

The NSFCloud workshop provided a set of insightful and detailed user requirements to guide all aspect of the development of the NSFCloud mid-scale platforms. Assembling and analyzing such requirements is an essential element of system development methodology.

Most of the requirements noted by breakout leads and scribes were detailed, well articulated, and well justified. We noted a strong convergence and consistency of almost all requirements coming from different areas of research as well as across different areas of focus: they described a need for a deeply reconfigurable and instrumented testbed with user controlled resource access. Such testbed would be unique among existing NSF mid-scale infrastructures and complement the NSF mid-scale set of resources by providing capabilities specifically addressing the needs of Computer Science experiments.

Going beyond the essential needs of providing and operating a testbed, the community feedback points to the fact that NSFCloud projects also have the opportunity to become a focus of cloud computing research and fertilize it, by providing many essential artifacts that can shape and guide it. Workshop attendees expressed the need for repositories of workloads, traces, datasets and other research and education artifacts currently either unavailable or available only in a very fragmented form. Further, the clear message is to not only have some of those artifacts provided by the testbed operators but to create methods for the research community to contribute to them, and ultimately make them available to all. Finally, the much voiced interest in reproducibility points to the fact that the testbeds can provide an essential vehicle for a set of methods enabling and encouraging a desirable element of experimental methodology.

The NSFCloud system is building a unique system that in many ways is a first of its kind. This being the case, the challenge for NSFCloud projects is to not only to build testbeds in themselves but also a community, both in terms of people and in terms of developing a consensus on common set of requirements specific to experimental Computer Science. We expect that both the community's diversity and understanding of those requirements will mature and become more concrete and advanced as the NSFCloud platforms become increasingly used. Therefore, to ensure that the link of testbeds with the community remains fresh, we should consider if the workshop should be repeated at some point in the future.

Acknowledgement

Results presented in this report were obtained using the NSF 1455829 Award "NSFCloud Workshop: Experimental Support for Cloud Computing" supported by the National Science Foundation.