HHS Public Access

Author manuscript

Neuroinformatics. Author manuscript; available in PMC 2018 January 01.

Published in final edited form as:

Neuroinformatics. 2017 January ; 15(1): 51–70. doi:10.1007/s12021-016-9315-8.

Automating NEURON Simulation Deployment in Cloud Resources

David B. Stockton[1] and **Fidel Santamaria**[2]

[1]Department of Biomedical Engineering, The University of Texas at San Antonio, San Antonio, TX 78249, USA

[2]Department of Biology, The University of Texas at San Antonio, San Antonio, TX 78249, USA

Abstract

Simulations in neuroscience are performed on local servers or High Performance Computing (HPC) facilities. Recently, cloud computing has emerged as a potential computational platform for neuroscience simulation. In this paper we compare and contrast HPC and cloud resources for scientific computation, then report how we deployed NEURON, a widely used simulator of neuronal activity, in three clouds: Chameleon Cloud, a hybrid private academic cloud for cloud technology research based on the Open-Stack software; Rackspace, a public commercial cloud, also based on OpenStack; and Amazon Elastic Cloud Computing, based on Amazon's proprietary software. We describe the manual procedures and how to automate cloud operations. We describe extending our simulation automation software called NeuroManager (Stockton and Santamaria, Frontiers in Neuroinformatics, 2015), so that the user is capable of recruiting private cloud, public cloud, HPC, and local servers simultaneously with a simple common interface. We conclude by performing several studies in which we examine speedup, efficiency, total session time, and cost for sets of simulations of a published NEURON model.

Keywords

Computer simulation; Cloud computing; Grid computing; NEURON (RRID:SCR_005393); Computational neuroscience; NeuroManager

Correspondence to: David B. Stockton.

[33]OpenStack API: http://developer.openstack.org/api-guide/quick-start/

[34]Rackspace API: https://developer.rackspace.com/docs/cloud-servers/v2/developer-guide/#api-reference

Compliance with Ethical Standards

Conflict of interests The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

We made use of Rackspace's developer program (Rackspace 2016b) which provides $50 free access per month to any developer for one year; we paid for all use above that amount. We also made use of Amazon's Free Tier (https://aws.amazon.com/free/) which gives any developer free access to 750 hours per month of instances built upon the "t2.micro" flavor for one year; we paid for all use above that amount as well as use of all flavors that were not "t2.micro".

Information Sharing Statement The NeuroManager software and instance configuration scripts are available at https://github.com/SantamariaLab/NeuroManager under an open source license that is presented at that location. MATLAB commercial software is available at http://www.mathworks.com/products/matlab/. Other software mentioned in this paper is freely available and has a footnote indicating the URL at which it can be found. The Miyasho model used in all simulations is available at ModelDB (RRID:SCR 007271, Model 17664).

Introduction

The NEURON simulation tool is widely used in computational neuroscience (Brette et al. 2007; Hines and Carnevale 1997). Researchers use NEURON on personal workstations, laboratory servers, and High Perfomance Computing (HPC) clusters to perform single and networked neuron simulations. Increasing complexity in the development of computer models and size of parameter spaces require efficient integration of these resources while keeping track of the origin of data structures and results, which is a significant challenge for both manual and script–assisted operation. For this reason we have recently developed the NeuroManager metasimulation tool (Stockton and Santamaria 2015), which provides workflow automation, input parameter isolation, hierarchical model construction, and automated batch simulation on a combination of laboratory servers, clusters, and super computers, for greater research efficiency and high–throughput parameter space exploration (Teka et al. 2016).

The world of computing is moving towards a software– defined commodity model of computation that provides both on-demand servers and on-demand networks (Buyya et al. 2009). It would be useful to the neuroscience community if our traditional tools could match pace with this new movement. In this paper we discuss cloud computing for neuroscience simulation, show how to run NEURON in the cloud, and present NeuroManager's ability to seamlessly add on-demand cloud–based simulator resources to the user's working simulation power.

The Cloud in Scientific Research

The Cloud is Ubiquitous in Science

Cloud technology has been changing the face of computing, including public–commercial cloud computing services and companies such as Amazon Elastic Cloud Computing (EC2) (Amazon 2016), Google Cloud Platform's Compute Engine (Google 2016), and Rackspace (Rackspace 2016a); private working clouds such as FermiCloud (Wu et al. 2014), CERN's multiple particle physics research clouds,[1] and the private–commercial CQSCS[2] (Cheng et al. 2015); clouds for cloud technology research such as the hybrid Chameleon Cloud (Chameleon Cloud 2016), or the Open Cloud Institute (UTSA 2016); and mobile applications extended with cloud services such as Apple's Siri (Nusca 2011). For the research community, cloud computing offers the opportunity to leverage practically unlimited computing power, as needed, while reducing investment in hardware infrastructure. Yet there are challenges, including the need for specialized expertise, prior investment in other computational facilities, and the need to focus on research topics rather than adapting to new computing mechanisms.

Although there are many views of cloud computing (Geelan 2009), for this paper the essence of cloud computing is that it provides on–demand software servers and clusters, called Infrastructure as a Service (IaaS) (Mell and Grance 2011). The user creates,

[1] https://www.openstack.org/user-stories/cern/
[2] CQSCS = Construction Quality Supervision Collaboration System.

configures, uses, and destroys them as needed with little regard for how or where they are implemented. The cloud server is based on an **image**, a file which is a snapshot of a software server that has a particular configuration of operating system plus other software. The user can also create custom images by configuring an existing server then saving it as an image. A cloud server also has a **flavor**, which determines a server's number of processors/cores, amount of RAM, bulk storage, and other resources.

Modern cloud technology provides performance essentially indistinguishable from hardware servers (Barham et al. 2003; Figueiredo et al. 2003; Keahey et al. 2005b), and can, in some cases, provide the specialized processing hardware such as GPU (Graphics Processing Units) hardware or Infiniband interconnections[3] traditionally associated with hardware servers or clusters. Typically, there is no waiting queue for cloud servers and no layer between the user and the cloud server, though the so–called 'bare–metal' versions, which allow the user exclusive access to hardware, may require reservations.[4] Public clouds and private clouds both provide cloud services, but private clouds tend to have better performance characteristics (Sadooghi et al. 2015).

Hardware cluster computing (often called grid or traditional HPC computing) provides unshared access to subsets of shared networked hardware, negotiated by a local resource manager; different 'queues' provide access to node–specific hardware such as coprocessors, GPUs, or large memory nodes. Often clusters provide high– speed communications such as Infiniband (Texas Advanced Computing Center 2016). Foster et al. (2008) provide extensive comparisons of grid and cloud computing from the aspects of architecture, security model, business model, programming model, virtualization, compute model, data model, locality, and provenance.

Many scientific communities have embraced or investigated the use of cloud resources for their research, including particle physics (Sadooghi et al. 2015), astrophysics (Smith 2011), high-energy physics (Segal et al. 2010; Taylor et al. 2015), computational chemistry (Thackston and Fortenberry 2015b), chemical modeling for high–throughput drug discovery (Moghadam et al. 2015), bioinformatics (Hanson et al. 2014), medical imaging (Kagadis et al. 2013), geophysics (Mudge et al. 2011), social sciences (Wittek and Rubio-Campillo 2012), geochemistry (Huang et al. 2014), genomic analysis (Ban et al. 2015), and various projects at the Department of Energy (Yelick et al. 2011).

Each scientific application presents a unique computational challenge, requiring a specific combination of vertical and horizontal scalability (described below) and thus an individual blend of the intrinsic advantages of HPC, private cloud, and public cloud.

Advantages of Cloud Computing for Simulation

Horizontal scalability—Primarily, cloud computing provides **horizontal scaling** — adding processors to allow more simulations to run simultaneously — but also does offer vertical scaling options such as increased server performance and faster simulation

[3] https://www.chameleoncloud.org/about/hardware-description/
[4] https://www.chameleoncloud.org/docs/bare-metal-user-guide/

completion (Arcitura 2016; Vaquero et al. 2011). **Vertical scaling** — increasing the power or resources of a processor to make a single simulation run faster — is better associated with the kind of traditional HPC seen in the Texas Advanced Computing Center Stampede supercomputer's Xeon coprocessor– equipped nodes (Texas Advanced Computing Center 2016). For the researcher, horizontal scaling tends to improve **makespan** for a *set* of simulations (elapsed time between start of the first simulation to completion of the last simulation) rather than the speed of a single simulation, but involves less investment of programming effort than vertical scaling's focus on improving the internal parallelism of a single simulation. In addition, the server management involved in horizontal scaling tends to be more future– proof, accessible, and portable than vertical scaling, which often involves highly refined hardware–specific tuning of specific internal simulation algorithms. Although cloud servers do offer a form of vertical scaling (Vaquero et al. 2011), it does not always translate into higher performance (Thackston and Fortenberry 2015b). The user must experiment to find the best configuration (Belgacem and Chopard 2015).

No Hardware Investment or Maintenance—The researcher does not need to invest in, maintain, or upgrade hardware resources, and there is no risk of losing hardware to obsolesence (Thackston and Fortenberry 2015b). Beginning researchers can lower their hardware acquisition barrier, and seasoned researchers can follow the computational demands of their projects with fine precision (Armbrust et al. 2010; Creeger 2009). Some companies have significantly reduced their data–center size through cloud computing (Creeger 2009). In addition, hardware expertise is kept where the hardware exists — externally, so the researcher does not need to hire or access hardware experts (Creeger 2009). There is, however, need for cloud software expertise similar to that of HPC centers, though this can be ameliorated with specialized software (Yelick et al. 2011), such as NeuroManager.

Wait–Free Access—In general, cloud servers appear to the user just like standalone hardware servers and are immediately accessible. In contrast, busy HPC clusters can cost the user substantial time sitting in job–waiting queues (Foster et al. 2008; Hoffa et al. 2008).

Better Resilience—Better Resilience of simulation capability. The loss of a computing resource such as an HPC cluster or cloud due to maintenance, business failure, or cash flow glitches can lead to simulation work slowdowns. Dependency on a single point of simulation computing power is a serious weakness in any research scenario. Being able to muster additional resources quickly and easily in order to keep simulation work active is a clear advantage of cloud computing (Grozev and Buyya 2014).

Dynamic Time/Cost Tradeoffs—Given proper software support, cloud computing allows the researcher to directly, immediately, and reversably trade off cost and time/ makespan for a given set of simulations. In many situations, this could involve improvements of days or even weeks of researcher/user time (Mudge et al. 2011). In contrast, the process of investing in hardware to decrease makespan is indirect, delayed, irreversible, and static.

A Flexibly Tailored Runtime Environment—A researcher can produce instances with a variety of operating systems, server capability, support software, simulator configurations, and simulator versions. In this way, researchers are able to reconstruct exact simulation conditions, support legacy software (Figueiredo et al. 2003), interface components designed for different operating systems (Yoginath and Perumalla 2013), resolve difficult configuration bugs without risk, and have complete support for custom operating systems, applications (Rehr et al. 2010), simulators, and/or custom modifications to existing simulators.

Improved Scientific Computing Provenance—In contrast to physical servers and clusters, the researcher can register an exact copy of the virtual machine (cloud server) on which a given simulation was done (Bechhofer et al. 2013; Dudley and Butte 2010; Howe 2012; Sliman et al. 2013). This feature can extend current neuroscience approaches to model sharing and provenance such as ModelDB, a public database of computational neuroscience models (Migliore et al. 2003). With cloud support, then, the ModelDB submission of a model could also include an OpenStack compatible image and/or configuration script that would allow the user to recreate the exact server on which a model was run.

The Academic and Commercial Clouds Used in this Study

We worked with three separate clouds in this study — one academic and two commercial.

Academic Cloud — Chameleon—The Chameleon Cloud (2016) is a cloud testbed designed for open source research into cloud computing. Sponsored by the National Science Foundation (National Science Foundation 2014), Chameleon is a federated cloud that integrates private cloud research clouds hosted by the University of Chicago and the Texas Advanced Computing Center. Chameleon offers an OpenStack KVM[5] Cloud which began in October 2015 and was used in the work described in this paper. Our quotas relevant to this paper were 20 instances, 40 virtual CPUs, 100GB of RAM, 50 floating IP addresses, and 1000GB of storage. We were able to choose from Chameleon–supplied images that included CentOS-6 and -7, Fedora-20, and Ubuntu Server.

Commercial Cloud — Rackspace—Rackspace (2016a) offers commercial public and private cloud services to individuals and businesses, including cloud servers, cloud clusters, and cloud storage; web hosting; and database hosting on a pay–as–you–go basis. We made use of their *developer*+ program (Rackspace 2016b) for the work in this paper. Our default quotas were 100 instances/IPs, 128GB of RAM, and up to 10 TB of SSD and 10TB of storage. We had about 22 stock images to choose from, including 13 images from the CentOS, CoreOS, Debian, Fedora, Red Hat, Ubuntu, and Vyatta varieties of LINUX, and 9 images from 3 types of Windows Server.

Commercial Cloud — Amazon Elastic Computing—Amazon Web Services (AWS) provides public and private cloud services worldwide, such as cloud instances (EC2), load balancing, storage (Elastic Block Storage, or “EBS”), web hosting, networking, and mobile

[5]KVM = Kernel-based Virtual Machine and refers to the type of hyper-visor employed by the cloud to run virtual machines; see http://www.linux-kvm.org.

services (Amazon 2016; Fusaro et al. 2011). We made use of their Free Tier[6] which gave us up to 20 instances and up to 750 hours per month of the "t2.micro" flavor, for a length of one year. We had the choice of over 24 images from Amazon, Red Hat, SUSE, and Ubuntu LINUX, and Microsoft Windows Server.

In this document we refer to Amazon Web Services as "AWS" and the Amazon EC2 cloud as "Amazon EC2" or "EC2" in accordance with current practice (Belgacem and Chopard 2015; Kaminski and Szufel 2015; Sadooghi et al. 2015; Yoginath and Perumalla 2015).

Hardware and Pricing Options—Although cloud computing is complex, in this project we deal primarily with the "Compute" aspect of a cloud — creating, configuring, running, and terminating instances. For a variety of customers, there are many other offerings including networking, webhosting, monitoring, and various types of storage.

For the most part, cloud instances are not tied to specific *hardware*. However, for issues of performance, specialized software, company policies, and restrictive software licenses (DAntoni 2013), cloud technology has evolved to allow a gradient of virtuality.

The Chameleon Cloud's KVM Cloud (used in the majority of this paper's experiments) offers no option to control which hardware hosts a virtual server. Instead, their "Bare Metal Reconfiguration" resources (Cha, 2016) allow user customization through a reservation system that gives the requester the ability to reserve hardware nodes for a specific date/time period with specific characteristics including: site, platform type, number of CPUs, number of cores, compute vs storage, and presence of Infiniband support. This reservation is called a "lease" and, once it begins, the user can launch instances with a flavor of "baremetal" which run directly on the reserved nodes. At that point, the instances are similar in many ways to those on the virtual cloud.

Rackspace offers "OnMetal Cloud Servers" which are created just as Virtual Servers are Laffoon (2016), yet do not share users, and are totally solid state. For example, the "Compute" flavor is a ten core Xeon machine with 32 GB RAM and a 32 GB system disk. The OnMetal servers are more expensive than Virtual Servers, though direct comparison is difficult.

Amazon EC2 provides many options for customizing the performance of cloud instances, networks, and storage (Amazon Web Services 2016; Thackston and Fortenberry 2015a). Instead of separating virtual and hardware servers the way Chameleon and Rackspace do, their instances are flavored using not only the normal number of virtual CPUs, memory, and included storage, but also by the exact type of hardware (processor, storage, GPU) that flavor is hosted on. For example, instances of the "m4.large" flavor are hosted on 2.4 GHz Intel Xeon E5-2676 v3 (Haswell) processors.[7] In addition, EC2 provides the "Dedicated Instances" option that provides physical isolation of the user's instances from other users and the "Dedicated Hosts" option that provides physical control for server–bound software licenses.[8]

[6] https://aws.amazon.com/free/
[7] These details can be seen at https://aws.amazon.com/ec2/instance-types/.

Cloud *pricing* is by instance flavor, bandwidth, and storage. For the Chameleon Cloud academic research cloud, pricing was hidden to us as researchers; what is tracked are Virtual CPU hours, gigabyte hours, and RAM hours.

In the Rackspace pricing scheme,[9] instances are charged for existence time, running or not. Ingoing data ("bandwidth") is not charged, but outgoing bandwidth is charged by the gigabyte in tiers. Image and other storage is also charged by the gigabyte–hour. Rackspace has volume discounts and also levels of technical support.

In the Amazon pricing scheme,[10] instances are charged only if running. Ingoing data is not charged, while outgoing data is charged by the gigabyte in tiers. Storage is charged by the gigabyte–month. Amazon has volume discounts and support levels.

The standard pay–on–demand pricing is common to both; however Amazon has two other purchasing options.[11] With their "Reserved Instances" option, the user reserves not an instance, but subscribes to a discount for one or three years that is specific to the instance flavor desired. Users can sell their Reserved Instances on the Amazon– provided "Reserved Instance Marketplace", which would allow a researcher to perform a period of heavy simulation then recoup costs quickly by selling the remainder of the reservation.

Amazon also has a "Spot Instances" program where customers can bid on potential instances. When their bid exceeds the going rate ("Spot Price"), those instances are created and automatically granted to the customer, until which time that the Spot Price exceeds their bid — then the instances are automatically terminated. This approach can lead to low prices but is only useful for applications in which the automatic external termination is not an issue (Gong et al. 2015).

Running NEURON in the Cloud

For the NEURON simulator, using the cloud means creating a cloud server with a compatible operating system, installing additional software facilities as required, installing NEURON on that server, uploading model and other simulation files, compiling mod files as required,[12] running the simulator, retrieving resulting data files, then terminating the server to avoid additional cost.

In order to avail oneself of the on–demand character of the cloud while minimizing cost and human workload, this process must be fully automated. Ideally, the use of computational resources should be integrated into simulation management software to improve daily workflow and to pare costs of resource use to the minimum. Automation can allow the user to take advantage of sophisticated scheduling algorithms that allow tradeoffs between several complex parameters: total simulation time, the cost of computer resources, the

[8] https://aws.amazon.com/ec2/purchasing-options/dedicated-instances/
[9] Discussion here: https://www.rackspace.com/cloud/public-pricing#cloud-servers and calculator here: https://www.rackspace.com/calculator
[10] Discussion here: https://aws.amazon.com/ec2/pricing/ and calculator here: http://calculator.s3.amazonaws.com/index.html
[11] https://aws.amazon.com/ec2/purchasing-options/
[12] In NEURON, mod files are used to define the simulation program for a biomechanism such as an ion channel. After definition, they are translated into C code and compiled into a biomechanism library before use in actual simulation.

heterogeneity of available resources, and the reliability of grid resources (Li et al. 2016; Panda and Jana 2015).

In this section we discuss the specific steps we found necessary to make use of the NEURON simulator on Chameleon Cloud, Rackspace, and EC2. Although both Chameleon and Rackspace clouds are based on the Open-Stack open source cloud software,[13] there are differences in their purposes, configurations, and even in their Application Programmer's Interfaces (APIs). EC2 has similar functionality but the API is substantially different from OpenStack. We worked manually with EC2 and manually and automatically with Chameleon and Rackspace.

Building a NEURON-Ready Cloud Server

Our experiences with Chameleon, Rackspace, and Amazon were slightly different. Since Chameleon is a cloud research resource, our project intially was completely bare. As a result it was necessary, using the Chameleon browser– based Dashboard, to set up a basic virtual network that included a virtual router to allow the network access to the outside world, which is a stereotypical set of cloud operations known as the "External Virtual Server Accessibility Cloud Design Pattern" (Erl et al. 2015). That network allowed us to create servers that were externally accessible using a public IP address. In contrast, on Rackspace and Amazon those details are, by default, invisible to the user and it was possible to create instances immediately that were externally–accessible. Both Chameleon and Amazon required preliminary setup of a security group that determines what traffic is allowed, and an SSH key pair for communication with the instances. The Chameleon preparations steps were:

- Create a virtual network and name it.
- Create a subnet of the virtual network and name it.
  - Choose IPv4 or IPv6.
  - Select DHCP yes.
  - Assign DNS Name Servers.
  - Select the gateway IP.
  - Select the local network address range.
- Create a virtual router and name it.
  - Attach router to external network.
  - Add a new interface to the router and attach it to the new subnet.

Network setup completed, we were able to create server instances using each of the Chameleon, Rackspace, and Amazon dashboards. On Chameleon, for most experiments (Section "Experimental Methods, Results, and Analyses") we chose a CentOS-6 server to match our in–lab servers, with a medium 'flavor' called 'm1.medium' that gave us 2 Virtual

[13] http://www.openstack.org

CPUs, 4 GB of RAM, and 40 GB of storage. On Rackspace, we chose a CentOS-6 server with a flavor called '2 GB General Purpose v1' with 2 CPUs, 2 GB of RAM, and 40 GB of storage. On Amazon, we used the Amazon Linux AMI image[14] with the c4.large flavor which has 2 CPU and 3.75 GB of RAM.

On the Chameleon Cloud, the user then must allocate a floating IP address from an address pool and assign it to the instance. The user logs into the instance through the Internet using that floating IP address. For Rackspace and Amazon, creating the server automatically allocated and assigned the public IP address. In all cases, the user makes use of dual key SSH-2 authentication (Barrett et al. 2005) to communicate with the servers.

The three clouds are different in the software stack available on the OS upon creation; this is true also of each image. Our goal was to configure the servers for NEURON simulation with Python (Hines et al. 2009). We referred to the basic approach described by the NEURON website[15] and by Andrew Davison[16] to install NEURON without Interviews, since we do not use the GUI. Because each cloud's images are different, we were obliged to scour the installation output logs in detail to determine which elements of the OS were necessary but not present; we then incorporated those elements into our configuration scripts. We credit the NEURON developers with providing the installation feedback that allowed us to do that. The basic steps to create a NEURON 7.4–ready cloud server were:

- Choose name, image, flavor, key pair, and subnet for the new instance.
- Launch the instance.
- If necessary, allocate a floating IP from a floating IP pool and attach it to the instance.
- Install basic terminal and compilation software using **yum** or other software package manager.
- Install Python 2.7 if necessary.
- Create specific directories in preparation for installation of NEURON.
- Download the NEURON 7.4 tar file from the NEURON repository or local source and unpack it.
- Configure the NEURON installer with Python location. – Install NEURON using **make** and **make install**.
- For use with NeuroManager, we also install the appropriate version of the freely available MATLAB (Natick, MA) Compile Runtime (MCR).[17]
- Create working directory for use with NeuroManager.

[14] AMI = 'Amazon Machine Image'.
[15] http://www.neuron.yale.edu/neuron/download/compile_linux
[16] http://www.davison.webfactional.com/notes/installation-neuron-python/
[17] Available free at http://www.mathworks.com/products/compiler/mcr/; the version matches the MATLAB compiler we have available.

Once the server was created and tested for correctness, we were able to save it as an image which can be cloned, manually or automatically, to produce multiple servers ready for simulation use. The actual scripts we used to configure a MATLAB–NEURON–Python server for the three clouds are available at the NeuroManager GitHub site,[18] but the user will need to adapt to his or her own needs and resources. These images could serve as a shareable 'appliance' on compatible clouds (Sharma 2008).[19] The cloud objective of universally–portable images/appliances, however, is still an elusive one (Howe 2012; Cloud Standards Customer Council 2014).

Figure 1 shows details of six servers using the Chameleon Cloud Dashboard. The server named 'dbs-test' was created using the dashboard from an image called 'CC–CentOS7–MCR2013a–NEURON74–Python27' which we created by the procedure listed above. The image supports a specific set of two neuroscience simulators: it runs compiled MATLAB programs that are compatible with MATLAB2013a, and it runs GUI–free NEURON simulations compatible with NEURON 7.4 that may or may not include Python 2.7 code. In NeuroManager, the JSON[20] file for a specific cloud describes the images available on that cloud, and each image description lists the neuroscience simulators it supports so that NeuroManager can check compatibility with the user's chosen neuroscience simulator. The remainder of the servers in Fig. 1 were created programmatically using NeuroManager's Cloud Management classes. The servers are of the m1.medium flavor and are essentially identical except for their external IP address.

We do not need a MATLAB license for any of the cloud servers, since we are making use of the compiled version of MATLAB. Instead, we compile the code on a local server using a single MATLAB license and push the compiled code to each cloud server. The free MATLAB MCR, which allows one to run compiled MATLAB without a license, is part of the configuration of the image upon which each server is based. The end result is that, in contrast to many applications (DAntoni 2013; Microsoft 2016), there is no additional licensing fee for use of MATLAB in the cloud.

For manual use, the cloning of a server is the easiest, most convenient way to add resources. For automatic use, cloning can be slower than actually recreating each server anew, since the use of cloning may involve transferring a large image file (the image 'CC–CentOS7–MCR2013a–NEURON74–Python27' seen in Fig. 1 is 4.7 GB) between servers internally within the cloud's physical network(s). The base OS images are much smaller (the CentOS-7 image is 677.3 MB) and may transfer internally more quickly. In other circumstances, installing from a base image followed by configuration may take far longer than transferring the larger, fully configured image (Keahey et al. 2005a). OpenStack, and other clouds, offer the ability to pass and automatically run configuration scripts and data (called 'user data') as part of launching a new instance, often with the use of the CloudInit software (Automating Openstack with cloud init run a script on VM's first boot 2015). By

[18] https://github.com/SantamariaLab/NeuroManager
[19] https://www.chameleoncloud.org/appliances/
[20] 'JSON' stands for Javascript Object Notation. Please see http://www.json.org/.

automating these processes, simulation management software can handle the full building of servers using a locally–optimal procedure without any additional effort by the user.

Running a Simulation Manually on a Cloud Server

Once a cloud server has been created and is NEURON– ready, the user can run a simulation on it using the same basic manual workflow presented in Stockton and Santamaria (2015).

Automating Cloud Operations

The dashboards provide manual/visual interaction with the cloud. For automated interaction, there are three options. The first is to use the cloud's client programs which, once installed on the user's computer, provide a command-line interface. For dealing with instances/servers in OpenStack, the command line interface is called 'nova'.[21] For example, the command to create a server is

$$\text{nova boot } -\text{image IMAGE } -\text{flavor FLAVOR SRV\_NAME}$$

The Amazon equivalent is called the Command Line Interface.[22]

The second option is to interact directly with the service. OpenStack, like many other cloud types (Foster et al. 2008; Cloud Standards Customer Council 2014), provides a RESTful[23] Application Programming Interface (REST-ful API).[24] Amazon EC2 also provides a RESTful interface (Amazon 2015). By making use of the API through the http protocol,[25] the user's programs (or the user's simulation manager) can create and terminate servers, query status and quotas, inquire about current charges, save and manage images, assign and reassign IP addresses, or perform any of an extensive array of cloud functions. For authentication in each interaction, Chameleon and Rackspace use the Open-Stack token procedure. In this procedure, the user's code requests a token from the cloud using assigned login credentials; for Chameleon the user uses the account password and for Rackspace the user retrieves a long numeric code from the account dashboard which effectively becomes the password. All subsequent API interactions with a given cloud must include that cloud's returned token, which will expire at some time and must be replaced by a new token request. Examples of direct access to the service can be seen in the API Quick Start Guide.[26] Amazon's authentication involves encrypted authorization signatures.[27] In API use on all three clouds, the principle of "Eventual Consistency" is at work, in which the user must verify states before and after issuing a command in order to ensure that the command's results have propagated completely; there is also the possibility that a valid command may fail and require resubmission (Amazon 2015; Rackspace 2016c).

[21] http://docs.openstack.org/user-guide/common/cli_overview.html
[22] https://aws.amazon.com/cli
[23] http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069
[24] http://developer.openstack.org/api-ref.html
[25] See RFCs 7230–7237 at http://tools.ietf.org/rfc/index
[26] http://developer.openstack.org/api-guide/quick-start/api-quick-start.html
[27] http://docs.aws.amazon.com/AWSEC2/latest/APIReference/making-api-requests.html

The third option is to use a Software Development Kit (SDK) that provides a software interface to the cloud either via the client programs or via the API. Several SDKs have been written in Python, Perl, C++, and other languages for use with the OpenStack API.[28] For example, in the Python SDK's *novaclient.v2.servers.ServerManager* class, the

$$\text{create}(\text{name}, \text{image}, \text{flavor}, \ldots)$$

method creates a new server[29]. As discussed below in Section "Managing Cloud Instances Within NeuroManager", we have developed a minimal MATLAB SDK for integrated NeuroManager use that uses the RESTFUL API and makes use of the cURL utility[30] for the http protocol. For example, NeuroManager internally uses the class method

$$\text{serverList} = \text{createMultipleServersNoWait}(\text{obj}, \text{num}-\text{Servers}, \ldots \text{serverNameRoot}, \text{imageName}, \text{flavorName}, \text{networkName})$$

to create a set of cloud servers in parallel; in practice, however, the user uses the interface shown in Fig. 2 and does not deal with the cloud directly. Similarly, EC2 provides a set of SDKs for use[31] and we are in the process of extending our SDK to EC2.

NeuroManager Automates NEURON use on a Blend of Clouds, Clusters, and Standalone Servers

NeuroManager is a metasimulation tool that automates the submission process for neuroscience simulations on a variety of platforms including local servers, clusters, and supercomputers (Stockton and Santamaria 2015). Neuro-Manager forms a Virtual Simulator based upon a SimCore (in this paper, the SimCore is either a MATLAB–based simulator or NEURON), places multiple Simulators on a variety of computational resources to form a Simulator Pool/Farm/Cloud, then schedules Simulations on those Simulators using a modified min–min scheduling algorithm (Braun et al. 2001). Simulations that become stalled on cluster waiting queues will be rescheduled on the fastest available Simulator, which may be on a cloud server. Simulators placed on standalone and cloud servers run as parallel multitasked processes; in contrast Simulators placed on clusters run as individual jobs on nodes using the number of cores requested by NeuroManager. Horizontal scaling in the NeuroManager sense, then, means adding Simulators to current or additional servers, clusters, and/or cloud servers. By ensuring there is an additional processing core associated with each additional Simulator, the user scales both parallelism and processing power. Adding a Simulator without adding an additional processing core still results in increased parallel performance in many situations, but the individual performance of the Simulators on that resource will be reduced.

Because NeuroManager acts as a super user and works with each cloud individually, cloud interoperability and portability are not issues; more formally, NeuroManager falls under the Cloud Standards Customer Council's Scenarios 2 and 4 (Cloud Standards Customer Council

[28] https://wiki.openstack.org/wiki/SDKs
[29] http://docs.openstack.org/developer/python-novaclient/ref/v2/serverser.html
[30] https://curl.haxx.se/
[31] See, for example, the Java SDK: https://aws.amazon.com/sdk-for-java/.

2014). As part of the process of extending NeuroManager to cloud instances, we recruited the cloud concepts of ‘image’ and ‘flavor’ for use with all computational resource types. NeuroManager uses a combination of JSON configuration files[32] and class inheritance to handle intercloud differences in API, images, flavors, methodologies, and workflow. The user places within the Simulator definition its requirements for SimCore and flavor minimums to allow NeuroManager to check them against the machine configuration the user has chosen.

NeuroManager can work with existing, continuously operating cloud servers, but can also use temporary cloud servers as required for a specific user configuration. After initial setup the user has the ability to configure a mixture of local servers, clusters, supercomputers, and cloud servers with minimal work (Fig. 2). For ‘ephemeral’ cloud servers which we call ‘Wisps’ that minimize cost because they exist only during simulation operations, NeuroManager can create, configure, use, and terminate the servers automatically, an example of the ‘Rapid Provisioning Cloud Design Pattern’ (Erl et al. 2015).

Managing Cloud Instances Within NeuroManager

For MATLAB, the language currently employed by Neuro-Manager, there is no existing OpenStack SDK. In addition, there are differences between the APIs of the two Open-Stack clouds we used in our work.[33][34] For these reasons and for better extensibility to other cloud types, we avoided the need to accumulate specialized client programs by developing a minimal object-oriented MATLAB SDK that uses the cURL tool’s http protocol support for interacting directly with the cloud’s instance services. The class hierarchy seen in Fig. 3 implements an instance management interface that hides the differences between the three clouds, focusing primarily on instance creation and termination, information query, and quota determination. We first developed the class hierarchy for the two OpenStack clouds, and the approach is proving extendable to EC2. Neuro-Manager’s instance management interface means that for daily operation the user does not need to deal with the cloud directly. This feature is an example of ‘externally– managed multi–cloud brokering’ (Grozev and Buyya 2014). In addition, the merging of virtual and bare–metal instances by both Rackspace and Amazon (Section “Hardware and Pricing Options”) means that NeuroManager could handle either with little or no change, and even mix the types at will.

Combining Cloud Servers with Other Resources

Figure 4 shows a snapshot of NeuroManager’s session– monitoring webpage in the middle of running a simulation set on mixed resources that include four servers on the Chameleon Cloud and three servers on the Rackspace public cloud, each hosting two simulators. All together, the Simulators created in this example form a ‘Simulator Cloud’ of eighteen independent Simulators. NeuroManager schedules simulations on those Simulators much like a cluster manager schedules jobs on cluster resources. The scheduling algorithm uses individual Simulator characteristics to place simulations efficiently and yet handle grid dynamics, including the ability to move simulations from stalled cluster–hosted simulators

[32]The JSON format is similar to the XML format but much less cluttered. It permits hierarchical data definition, whereas the more familiar INI files do not.

to non–stalled ones on other resources. We also were successful in adding persistent Amazon EC2 instances to this combination.

Managing Cloud Costs with NeuroManager

NeuroManager provides the ability to create and terminate Wisps, allowing the user to minimize the *existence time* of a server, and thus cost (suitable for Rackspace). The ability to start and stop existing cloud servers allows the user to minimize the *runtime* of a server (suitable for Amazon). The ability to distribute Simulators at will on a mixture of heterogeneous resources allows the user to trade off makespan and costs by determining how much of the bulk of a simulation set will be run on the cloud, and to determine what percentage of the cloud portion is on more expensive clouds. NeuroManager's Simulation Set approach, where a number of simulations are processed, scheduled, and run in bulk, permits closer packing of server use and, since the smallest billing time unit of clouds is typically one hour, helps reduce unused server partial hours. Finally, the user can determine on–the–fly how much of the simulation output data to download; thus avoiding data transfer charges for data that is of no interest.[35]

Cloud Clusters and Parallel NEURON

When configured for parallel use using the **paranrn** module (Brette et al. 2007; Hines and Carnevale 2008; Migliore et al. 2006), NEURON uses time–stepped synchronization (D'Angelo and Marzolla 2014) to perform Message Passing Interface (MPI)–based Parallel Discrete Event Simulations (PDESs) (Misra 1986; Yoginath and Perumalla 2013), continuous model simulations (Hines et al. 2008a, b), or continuous–PDES hybrids, and has been run on physical clusters to good effect (Schneider et al. 2015; Silverstein and Lansner 2011). Unfortunately, cloud clusters[36] show significant performance loss due to instance placement (Ballani et al. 2011), interinstance communications issues such as variability and latency (El-Khamra et al. 2010; He et al. 2010; Sadooghi et al. 2015), and heterogeneous workloads among simulation components (Yoginath and Perumalla 2013, 2015). Similarly, MPI–based applications are strongly dependent on low communications latency (Mauch 2015). While HPCs provide less scalability than clouds and often large wait times, they also provide hardware–optimized interinstance communications and non–shared processors. Various groups are working to improve interinstance communications (Ballani et al. 2011; Ban et al. 2015; Branch et al. 2014; Mauch 2015; Peng et al. 2015) and hypervisor[37] scheduling policies (Yoginath and Perumalla 2013, 2015).

Although it is certainly possible to run a distributed, MPI–based application such as parallel-configured NEURON on a cloud cluster (El-Khamra et al. 2010), the performance of cloud clusters has not yet reached that of physical clusters, though this may be changing for some applications (Mossucca et al. 2015). What 'performance' means in cloud services, though, is a complex topic and is neither well-controlled nor well–defined (Mogul and Popa 2012). Performance–wise, investigators may be best served by running parallel NEURON on

[35]By programming the UserSimulation() function accordingly; see Stockton and Santamaria (2015).
[36]Cloud clusters are virtual networks formed from cloud instances.
[37]The *hypervisor* is the program that runs on the physical processor and produces the virtual machines, or cloud servers, that are hosted by the physical processor.

physical clusters, as in Schneider et al. (2015), or possibly bare–metal clusters with Infiniband connections (Rad et al. 2015). Based on these considerations, we have reserved NeuroManager's handling of cloud–hosted clusters for the near future.

The *multisplit* functionality included in the paranrn option (Hines et al. 2008a) provides single–node multi–core vertical scaling of certain types of single simulations by parsing out to multiple processors dendritic trees that fall into specific morphology classes.[38] By requesting multiple cores in the cluster job file through the machine class, or by requesting cloud servers with a flavor that provides multiple processors, the user could leverage both vertical and horizontal scaling on both cluster and cloud.[39]

Experimental Methods, Results, and Analyses

We ran experiments to explore the characteristics of NeuroManager's usage of cloud servers in comparison to HPC resources from the perspective of single–node NEURON simulation. Others have done such from other perspectives (Gupta and Milojicic 2011; Iosup et al. 2011; Ismail and Khan 2015; Jackson et al. 2010; Mauch 2015; Oesterle et al. 2015; Sadooghi et al. 2015; Thackston and Fortenberry 2015b; Yelick et al. 2011; Zaspel and Griebel 2011).

We set NeuroManager up to run a publicly–available NEURON model obtained from ModelDB[40] (Miyasho et al. 2001), which applies a current step to the soma of a modeled Purkinje cell. After adjusting the stop time to achieve a runtime of 0.5 hours on the standalone server, we ran the simulation with identical input parameters for all experiments. We used the "KhStudy" Simulator[41] which modifies one of the model's NEURON mod files, compiles all mod files into a library, runs NEURON with the newly–created library and the other input parameters specified by the user, then generates plots of the resulting membrane voltage signal using MATLAB compiled code. In most of these experiments we used the Chameleon Cloud to host the cloud servers, but we have run other sessions using all three clouds, separately and together.

Runtime Dependence on Flavor

We determined the individual runtimes, $T_{runtime}$, for our chosen NEURON simulation on each resource: standalone server, local cluster, and the Chameleon, Rackspace, and Amazon clouds. $T_{runtime}$ is the actual NEURON simulation runtime of an individual simulation on the resource as gathered by MATLAB's tic/toc facility and does not include file transfer or overheads associated with running multiple simulations. We ran two Simulators (two parallel simulations) on each resource. Our resources were as follows.

- Hardware standalone server: a Dell PowerEdge T620 with two 2.9 GHz processors with 6 cores each and 64 GB RAM;
- Local Sun Grid Engine (SGE) HPC cluster: twenty nodes, each with 2 Intel Xeon E5450 Quad Cores for a total of 8 cores per node and 16GB of RAM.[42]

[38]The approach is also suitable for multi–node cluster–based simulations but we focus on single–node applications here.
[39]Note that cluster and cloud nodes are often limited in cores to eight or fewer each.
[40]Model 17664; see https://senselab.med.yale.edu/modeldb/ShowModel.cshtml?model=17664.
[41]See https://github.com/SantamariaLab/NeuroManager/tree/master/NeurSim/MiyashoMOD/KhStudy.
[42]http://cbi.utsa.edu/hardware/cluster

This cluster has other queues which the scheduler also used depending on number of cores requested and availability;

- Chameleon Cloud: four different flavors of servers, including “m1.small” (1 CPU and 2 GB RAM), “m1.medium” (2 CPU and 4 GB RAM), “m1.large” (4 CPU and 16 GB RAM), and “m1.xlarge” (8 CPU and 16 GB RAM);
- Rackspace cloud: “2 GB General Purpose v1” flavor of server (2 CPU and 2 GB RAM);
- Amazon EC2: servers with flavors “t2.large” (2 CPU and 8 GB RAM) and “c4.large” (2 CPU and 3.75 GB RAM).

The smallest Chameleon cloud flavor (“m1.tiny”) was too small to host the CentOS–6 or 7 operating system which we used for all cloud servers.[43]

The results show that the three largest Chameleon flavors had a shorter simulation runtime than the standalone server, the HPC, and the other two clouds (Fig. 5). The Rackspace and Amazon servers had a comparable run time. Note that the “small” flavor had one core for two Simulators, the “medium” flavor had one core per Simulator, and the HPC had one core for each Simulator, whereas the lab server and larger flavors had multiple cores per Simulator. Since the “cloud small” flavor has fewer cores than Simulators, it runs longer. There was no advantage to running larger flavors where the number of cores exceeds the number of Simulators; accordingly we used the “medium” flavor for most experiments in the rest of this study.

The data presented in Fig. 5 also quantifies the variability of the simulation runtime on each resource. The standalone server, with a fixed hardware configuration and no user competition, was the most consistent in runtime. Of the remaining resources, the HPC showed the highest runtime variance. We suspect this to be due to the fact that we were requesting single core jobs on the cluster’s “all” queue which uses all available hardware. Each cluster node has a varying number of cores, memory, and communications hardware; in addition, some Simulators may have been hosted on nodes also running jobs from other users. This observation is useful because a common criticism of cloud servers is that their performance can vary due to hardware placement which is not under control of the user; in fact “server migration”, moving poorer performing virtual servers to different hardware automatically to improve performance, is a significant aspect of cloud operations (Mishra et al. 2012).[44]

The variability seen in the HPC resource can also be seen in cloud resources. When we used the “t2.large” flavor, which although called “General Purpose”, is intended for burst traffic only, the first few simulations ran in less than 18 minutes (about the same as the CCloud larger flavors), but subsequent simulations ran very slowly (up to 1.5 hours). In contrast, the compute flavors are intended for sustained use and give much faster and more consistent

[43]Except for Amazon; there is a licensing fee for Centos–7 usage on Amazon EC2, and it must be obtained through the AWS Marketplace. See https://aws.amazon.com/marketplace/b/2649367011.

[44]Our experiments did not make use of server migration or load balancing of any kind, however NeuroManager’s scheduler favors faster Simulators.

performance for NEURON simulations. It is important that the cloud user choose the proper flavor for the intended application.

Effects of Horizontal Scaling on Simulation Session Performance

We investigated the characteristics of horizontal scaling using HPC–based Simulators and using cloud–based Simulators.

Horizontal scaling on the HPC was administratively restricted to 50 simultaneous jobs per user, limiting horizontal scaling to a maximum of $R = 50$ Simulators. Additional jobs are also restricted to nodes that have space available, so there is a dynamic point where adding additional HPC–based simulators does not horizontally scale. The user must choose a point between more parallelism (fewer cores per job and hence more room on the cluster for running jobs) and more consistency (more cores per job, reducing node–local timesharing with other users).

Horizontal scaling on the cloud was limited only to the quotas set for each user. In the case of Chameleon Cloud, we were able to double our default quotas by request, resulting in new quotas of 20 instances/40 cores/100 GB RAM total. The public clouds have similar quota systems, which can also be changed through interaction with cloud representatives. In contrast to the HPC, we were able to add Simulators as needed and knew that they would all run in parallel without being placed in a waiting state.

We ran an experiment consisting of three sets of five NEURON/NeuroManager simulation sessions, with the purpose of quantifying the effects of horizontal scaling on simulation session performance. In each set the base session consisted of 8 Simulators running a total of 20 simulations; then we scaled the number of Simulators and simulations by scale factor F = [1, 2, 3, 4, 5], resulting in R = [8, 16, 24, 32, 40] Simulators, and N = [20, 40, 60, 80, 100 simulations.] In Set 1 (HPC only) we ran one Simulator per actual HPC core. In Set 2 (Chameleon Cloud only) we ran one Simulator per virtual core using only medium flavor servers (2 cores per server), resulting in $N_{servers}$ = [4, 8, 12, 16, 20]. In Set 3 (Chameleon Cloud only) we used 8 medium flavor servers in all five scale factors, thus fixing the number of cores at 16 and placing 1, 2, 3, 4, and 5 Simulators on each server. Set 1 sessions were each run five times, Set 2 sessions three times, and Set 3 sessions one time, for a total of 45 sessions and 2700 individual identical simulations, each lasting from less than 20 minutes to more than one hour depending on the resource.

For analysis, we adapted classic parallel processing performance definitions (Censor and Zenios 1997) to Neuro-Manager use by defining the ‘problem’ as a session (one set) of simulations and a ‘processor’ as a Simulator–core.

A NeuroManager session that runs a set of simulations is composed of three stages: setup → running → teardown. The time taken by a session is:

$$T_{session} = T_{setup} + M + T_{teardown} \quad (1)$$

where $T_{session}$ is the total time taken by the session, T_{setup} is the setup time, M is the makespan, and $T_{teardown}$ is the time to remove all session–related structure from the remote resources, including Wisps.

The *relative speedup* S_{rel} of a specific session S is:

$$S_{rel}(S) = \frac{\sum_{i=1}^{N_S} T_{\text{simulation}_i}}{M_S} \quad (2)$$

where N_S is the number of simulations in session S, $T_{\text{simulation}_i}$ is the runtime for single simulation i in S, and M_S is the makespan developed by session S. $S_{rel}(S)$ indicates how many times faster the parallel version is and includes all overhead except initial setup time $T_{setup}(S)$ and post–simulation teardown time $T_{teardown}(S)$.

Similar to relative speedup is the *absolute speedup* S_{abs} of a session S:

$$S_{abs}(S) = \frac{N_S \cdot \min_i(T_{\text{simulation}_i})}{M_S} \quad (3)$$

where the minimum is taken over all simulations in the session. This equation considers the parallelism improvement if one assumes the serial runtime had been performed using the processor that gave the shortest simulation runtime in the session.

We also have the absolute speedup S_{absgroup} of a single session S with respect to a group G of sessions of which it is part; we call this the “AbsGroup Speedup”:

$$S_{absgroup}(G, S) = \frac{N_S \cdot \min_{i,j}\left(T_{\text{simulation}_i^j}\right)}{M_S} \quad (4)$$

where $T_{\text{simulation}_i^j}$ is the ith simulation in the jth session of group G, and the minimum is taken over all i and j. This equation considers the parallelism improvement of a session if one assumes the serial runtime had been performed using the processor that gave the shortest simulation runtime of all the sessions in the group, independent of resource.

Normalizing the relative speedup S_{rel} of a session S by the number R of processors/Simulators/cores used in that session gives the efficiency η of a session, a measure of how effective the parallel configuration was:

$$\eta(S) = \frac{S_{rel}(S)}{R} = \frac{\sum_{i=1}^{N_S} T_{\text{simulation}_i}}{RM_S} \quad (5)$$

The degree to which the efficiency stays constant while the load (total simulation runtime) and capacity (number of processors) are both increased by the same factor is called the *scalability*.

Speedup, Efficiency, and Scalability—For Sets 1 and 2 we calculated the values of S_{rel} and S_{absgroup} (Fig. 6a). We see that the HPC and cloud give a steady, similar S_{rel}, reflecting the increased number of Simulators/cores in play. When S_{absgroup} is calculated relative to the 975.3 second minimum runtime of all of the simulations in Sets 1 and 2 (the minimum occurred on a cloud–based Simulator), we see that the HPC fairs less well than the cloud because of the cloud's faster, more consistent runtimes.

For Sets 1 and 2 we also calculated the values of η and T_{session} (Fig. 6b). We see that cloud and HPC efficiency are similar; both efficiency lines have a shallow downward trend, reflecting serial aspects of makespan operation including NeuroManager's scheduling loop. There is a slight climb in T_{session} for both configurations, probably due to the efficiency loss just mentioned; in every case, however, T_{session} was lower for the cloud than for the HPC.

In contrast, Set 3, which scaled Simulators but not cores, had no increased speedup for $F >$ 2, which is the point where the number of Simulators matched the number of cores (not shown).

Session Time breakdown—Setup time T_{setup} is of concern in the use of Wisps, where time is required to create new servers. NeuroManager uses the cloud's API to initiate creation of the Wisps in a Wisp Set in parallel, then waits for all Wisps in the set to reach the RUNNING state. The much shorter time required by this approach, though still a function of the number of servers, helps reduce total setup time.

To determine the relationship of T_{setup} to session time $T_{session}$ as a function of scale factor, we analyzed Sets 1–3 to break down session time into its component intervals. Figure 7 shows the breakdown of $T_{session}$ and T_{setup} as a function of number of Simulators for Sets 1 and 2. For Set 3 setup times were constant, with an average of 427 seconds (not shown). In contrast to the cloud sessions, the setup associated with the HPC is shorter, reflecting many differences: no need for Wisp Set creation, only a single communications test, vastly fewer file uploads, and the use of local network communications rather than farther–reaching communications through multiple networks. Despite these costs, the cloud setup load is offset by gains in scalability, runtime consistency, speedup, parallel efficiency, and overall session speed.

To ensure that Wisp creation time on Chameleon was comparable to that on a public cloud, we supplemented Set 2 data with additional sessions on Rackspace (five sessions involving creation of ten Wisps each). The average Wisp creation time on Chameleon was 16.1 seconds/Wisp, and on Rackspace 37.4 seconds/Wisp.

Flavor and Cost

We quantified the effects of cloud server flavor, number of simulations, and simulation length on session time and cost. For this purpose we ran a new set of simulations (Set 4) that

consisted of eight sessions (4.1–4.8) ranging from one to four hours each (a total of 560 simulations). We used one Simulator per core with a total of 16 Simulators distributed on the appropriate number of servers of identical flavor. All servers were on Chameleon Cloud (Table 1).

Sessions 4.1–4.4 show that, although setup time decreased as the number of servers decreased, average simulation runtime increased, perhaps due to the lack of timesharing with other Simulators on the same server. The resulting overall session time T_{session} was independent of configuration. When we increased the number of simulations 4× to 160 (4.5–4.6) the overall effect is that the two $T_{session}$ values are identical. Increasing the number of simulation steps per simulation 4× (4.7–4.8) emphasized the faster performance of the smaller flavors, resulting in lower $T_{session}$, despite the resulting larger data downloads.

Although we performed most of our analyses in a no–cost research cloud environment, cost is a concern if using commercial services. We re–examined the Set 4 results from the perspective of cost. We calculated the potential cost using the charge for the Rackspace General Purpose flavor with the same number of cores as the Chameleon Cloud flavor in use, assuming that there are 750 hours per month and that the minimum billable unit is one second, and ignoring any data charges or monthly support charges. For ephemeral servers costs depend on flavor and $T_{session}$; the results are presented in Table 1. The Amazon EC2 c4 compute equivalents (not shown), in comparison, are about 68 % higher than the Rackspace costs, suggesting that it is advantageous to the user that simulation management software supports multiple clouds for avoiding the “vendor lock–in” problem (Opara-Martins et al. 2014).

Related Work

The **Neuroscience Gateway** (NSG) is a portal that provides vetted users web–based access to super–computer installations of specific versions of standard simulators including NEURON (Subhashini et al. 2013a, b, 2015). The portal does not use the cloud for simulations but does offer access to cloud storage.

Yamazaki et al. (2011) report on a service called **Simulation Platform** that is similar to the Neuroscience Gateway, but is cloud–based. The platform provides the user with a web–based interface to virtual machines that offer GENESIS, NEURON, and NEST, as well as various plotting and analysis tools.

The NSG and Simulation Platform facilities are both complementary to NeuroManager in that they offer specific implementations to a researcher who desires a point–and– click interface during sit–down sessions. Both facilities are Software–as–a–Service (SaaS) portals to which the user comes to do work. In contrast, the user employs NeuroManager to form his or her own custom simulator farm on–the– fly, from any or all available IaaS resources, for automated parallel simulation and sophisticated model/simulator configuration and exploration.

NeuroManager‘s integration of cloud, grid, and server does so without being limited by middleware boundaries. A more involved approach to managing instances is called

Dynamic Virtual Environments (DVEs) or **Virtual Workspaces** (VSs) (Keahey et al. 2004, 2005a), and is built upon middleware such as the Globus Toolkit (Foster 2005). NeuroManager uses JSON files for machine configuration that are similar to the XML description files used by the DVE/VS system. A fully automated approach to instance configuration as treated by Keahey et al. (2005a) or Tihfon et al. (2016) appears indispensible for future work.

The **Open Cloud Computing Interface**, or OCCI (http://occi-wg.org/) is representative of efforts to provide a common interface to all cloud types. Another approach is to use a **Cloud Broker** such as STRATOS (Pawluk et al. 2012; Grozev and Buyya 2014), which adds a layer between the customer and multiple (commercial) clouds; that layer manages cloud differences and negotiates prices. Like STRATOS, NeuroManager's common resource interface permits the user to delay deployment decisions until runtime, enabling the user to work around maintenance, communications trouble, or restrictive quotas. The overall broker approach, however, appears to be oriented towards business needs, where fairness is important (Aazam and Huh 2015), in contrast with scientific computing, where fairness can contribute to poor performance, as described above in Section "Cloud clusters and parallel NEURON". NeuroManager uses JSON–format configuration files together with inherited methods to allow multi– cloud operation. There is an effort to standardize the description and publishing of cloud entities and differences in a way that is web–accessible through RESTful services (Smit et al. 2012).

Discussion

In this paper we presented advantages of cloud resources for computational neuroscience. We have verified a selection of them with detailed experiments that show that the cloud has better horizontal scalability and overall performance when combined with automated workflow software such as NeuroManager. In addition, we have shown that we can add cloud servers quickly and easily without additional investment in hardware, addressed the problem of waiting queues in clusters by automated rescheduling of waiting jobs onto cloud servers, shown that we can construct servers with specific versions of simulation software rather than that provided to us by an institutional resource, and demonstrated how to reconstruct the exact server on which our simulations were run.

A mental shift away from considering only single– simulation speed and towards examining the total speed of sets of independent simulations run partially or totally in parallel can save a researcher considerable time and investment. Vertical scaling, which invests effort to improve the speed of a single simulation, tends to be perceived as more sophisticated than horizontal "embarrassingly parallel" operations, which run multiple simulations simultaneously, but both are potential contributors to improved simulation throughput. The cloud is not yet suitable for all types of simulations, especially multinode simulations that are extensively communications–bound, such as large–scale neuronal networks. Not all simulations, however, require the vertical scaling offered by HPC resources. The HPC's limitations on horizontal scaling can reduce simulation throughput in comparison with the cloud's ability to appear limitless. By considering both traditional vertical scaling and the dynamic horizontal scaling now available through the cloud, a researcher can produce higher

quality, more detailed examinations of parameter space with less effort, time, and hardware investment.

Computational neuroscience research can gain advantages such as research speed, quickness, flexibility, and cost savings by incorporating cloud technology. Cloud resources can be used to supplement or even replace existing resources, depending on the researcher's needs for computational power and speed in simulation throughput. These advantages are possible only through automation of the use of computational resources, as can be seen by our outlining of the steps involved. Adding additional resources, no matter how prevalent or accessible, is only helpful if the user's workload does not increase proportionally. Any cloud–capable automation should support multiple clouds to ensure the researcher is not trapped by vendor limitations and pricing.

The NeuroManager metasimulation tool allows automated use of heterogeneous resources, including cloud resources, and by implementing API-based facilities for hands–off on–demand server creation and termination, allows the user to minimize costs while increasing temporary simulation power. NeuroManager's virtualization means that the user has a nearly identical interface to HPC computing resources as to cloud resources, so it is straightforward to develop a simulation set on a local, free resource and then extend the resource set to external resources that may involve costs or to future internal resources that have a cloud structure. By forming an environment in which an established simulator such as NEURON can thrive on multiple types of cutting–edge computing resources, the simulation management software provides continuity in which computational neuroscience investigations can gain new power.

Acknowledgments

NSF-EF1137897, NSF-DBI1451032, NIH-G12MD007591 (for use of computational facilities at UTSA), Texas Advanced Computing Center for providing HPC resources, and the Computational System Biology Core at UTSA for providing access to the Chameleon Cloud facilities.

References

Automating Openstack with cloud init run a script on VM's first boot. 2015. https://raymii.org/s/tutorials/Automating_Openstack_with_Cloud_init_run_a_script_on_VMs_first_boot.html

Aazam M, Huh EN. Cloud broker service-oriented resource management model. Transactions on Emerging Telecommunications Technologies. 2015

Amazon. Amazon Elastic Compute Cloud – API Reference – API Version 2015-10-01. 2015

Amazon. Amazon Web Services. 2016. https://aws.amazon.com/. Accessed 4 January 2016

Amazon Web Services. Amazon EC2 – virtual server hosting. 2016. https://aws.amazon.com/ec2/

Arcitura. WhatIsCloud.com. 2016. www.whatiscloud.com

Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, et al. A view of cloud computing. Communications of the ACM. 2010; 53(4):50–58.

Ballani, H., Costa, P., Karagiannis, T., Rowstron, A. Proceedings of the ACM SIGCOMM 2011 Conference. Association for Computing Machinery; 2011. Towards predictable datacenter networks; p. 242-253.

Ban K, Tan TW, Chrzeszczyk J, Howard A, Li D. InfiniCloud: leveraging global InfiniCortex fabric and openstack cloud for borderless high performance computing of genomic data and beyond. Supercomputing Frontiers and Innovations. 2015; 2:14–27.

Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization. ACM SIGOPS Operating Systems Review. 2003; 37(5):164–177.

Barrett, DJ., Silverman, RE., SSH, RGB. The secure shell: The definitive guide. O'Reilly Media, Inc; 2005.

Bechhofer S, Buchan I, De Roure D, Missier P, Ainsworth J, Bhagat J, Couch P, Cruickshank D, Delderfield M, Dunlop I, et al. Why linked data is not enough for scientists. Future Generation Computer Systems. 2013; 29(2):599–611.

Belgacem MB, Chopard B. A hybrid HPC/cloud distributed infrastructure: Coupling EC2 cloud resources with HPC clusters to run large tightly coupled multiscale applications. Future Generation Computer Systems. 2015; 42:11–21.

Branch R, Tjeerdsma H, Wilson C, Hurley R, McConnell S. Cloud computing and big data: A review of current service models and hardware perspectives. Journal of Software Engineering and Applications. 2014; 2014

Braun TD, Siegel HJ, Beck N, Bolöni LL, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B, Hensgen D, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing. 2001; 61(6):810–837.

Brette R, Rudolph M, Carnevale T, Hines M, Beeman D, Bower JM, Diesmann M, Morrison A, Goodman PH, Harris FC Jr, Zirpe M, Natschläger T, Pecevski D, Ermentrout B, Djurfeldt M, Lansner A, Rochel O, Vieville T, Muller E, Davison AP, El Boustani S, Destexhe A. Simulation of networks of spiking neurons: a review of tools and strategies. Journal Computer of Neuroscience. 2007; 23(3):349–398. DOI: 10.1007/s10827-007-0038-6

Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems. 2009; 25(6):599–616. ISSN 0167-739X. http://www.sciencedirect.com/science/article/pii/S0167739X08001957. DOI: 10.1016/j.future.2008.12.001

Chameleon Cloud Bare Metal User Guide. 2016. https://www.chameleoncloud.org/docs/bare-metal-user-guide/

Censor, Y., Zenios, SA. Parallel optimization: Theory, algorithms, and applications. Oxford University Press on Demand; 1997.

Chameleon Cloud. Chameleon Cloud: A configurable experimental environment for large-scale cloud research. 2016. https://www.chameleoncloud.org/

Cheng Y, Chen Y, Wei R, Luo H. Development of a Construction Quality Supervision Collaboration System based on a SaaS private cloud. Journal of Intelligent & Robotic Systems. 2015; 79(3–4): 613–627.

Cloud Standards Customer Council. Interoperability and portability for cloud computing: A guide. 2014. p. 1-31.http://www.cloud-council.org/deliverables/CSCC-Interoperability-and-Portability-for-Cloud-Computing-A-Guide.pdf

Creeger M. Cloud computing: An overview. ACM Queue. 2009; 7(5):2.

D'Angelo G, Marzolla M. New trends in parallel and distributed simulation: From many-cores to cloud computing. Simulation Modelling Practice and Theory. 2014; 49:320–335. ISSN 1569-190X. http://www.sciencedirect.com/science/article/pii/S1569190X14001014. DOI: 10.1016/j.simpat.2014.06.007

DAntoni, J. The SQL virtualization tax?. 2013. https://joeydantoni.com/2013/02/07/the-sql-virtualization-tax/

Dudley JT, Butte AJ. In silico research in the era of cloud computing. Nature Biotechnology. 2010; 28(11):1181–1185. ISSN 1546-1696. DOI: 10.1038/nbt1110-1181

El-Khamra, Y., Kim, H., Jha, S., Parashar, M. IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom). IEEE; 2010. Exploring the performance fluctuations of HPC workloads on clouds; p. 383-387.

Erl, T., Cope, R., Naserpour, A. Cloud computing design patterns. Prentice Hall Press; 2015.

Figueiredo, RJ., Dinda, PA., Fortes, JAB. A case for grid computing on virtual machines. IEEE; 2003.

Foster, I. Network and Parallel Computing. Springer; 2005. Globus Toolkit version 4: Software for service-oriented systems; p. 2-13.

Foster, I., Zhao, Y., Raicu, I., Lu, S. Grid Computing EnvironmentsWorkshop, 2008 GCE'08. IEEE; 2008. Cloud computing and grid computing 360-degree compared; p. 1-10.

Fusaro VA, Patil P, Gafni E, Wall DP, Tonellato PJ. Biomedical cloud computing with Amazon Web Services. PLoS Computer Biology. 2011; 7(8):e1002147.

Geelan, J. Twenty-one experts define cloud computing. 2009. http://virtualization.sys-con.com/node/612375

Gong, Y., Zhou, AC., He, B. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. ACM; 2015. Monetary cost optimizations for MPI-based HPC applications on Amazon clouds: Checkpoints and replicated execution; p. 32

Google. COMPUTE ENGINE: Scalable, high-performance virtual machines, 2016. 2016. https://cloud.google.com/compute/. Accessed 4

Grozev N, Buyya R. Inter-cloud architectures and application brokering: taxonomy and survey. Software: Practice and Experience. 2014; 44(3):369–390.

Gupta, A., Milojicic, D. Open Cirrus Summit (OCS). Sixth. IEEE; 2011. Evaluation of HPC applications on cloud; p. 2011

Hanson, NW., Konwar, KM., Wu, SJ., Hallam, SJ. IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology. IEEE; 2014. Metapathways v2. 0: A master-worker model for environmental pathway/genome database construction on grids and clouds; p. 1-7.

He, Q., Zhou, S., Kobler, B., Duffy, D., McGlynn, T. Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM; 2010. Case study for running HPC applications in public clouds; p. 395-401.

Hines ML, Carnevale NT. The NEURON simulation environment. Neural Computer. 1997; 9(6):1179–1209.

Hines ML, Carnevale NT. Translating network models to parallel hardware in NEURON. Journal Neuroscience Methods. 2008; 169(2):425–455. DOI: 10.1016/j.jneumeth.2007.09.010

Hines ML, Eichner H, Schürmann F. Neuron splitting in compute-bound parallel network simulations enables runtime scaling with twice as many processors. Journal of Computational Neuroscience. 2008a; 25(1):203–210. [PubMed: 18214662]

Hines ML, Markram H, Schürmann F. Fully implicit parallel simulation of single neurons. Journal of Computational Neuroscience. 2008b; 25(3):439–448. [PubMed: 18379867]

Hines ML, Davison AP, Muller E. NEURON and Python. Frontiers in Neuroinformatics. 2009; 3(1)doi: 10.3389/neuro.11.001.2009

Hoffa C, Mehta G, Freeman T, Deelman E, Keahey K, Berriman B, Good J. On the use of cloud computing for scientific workflows. IEEE Fourth International Conference on eScience, 2008 eScience'08. 2008; :640–645. DOI: 10.1109/eScience.2008.167

Howe B. Virtual appliances, cloud computing, and reproducible research. Computing in Science & Engineering. 2012; 14(4):36–41.

Huang X, Cao G, Liu J, Prommer H, Zheng C. Reactive transport modeling of thorium in a cloud computing environment. Journal of Geochemical Exploration. 2014; 144(Part A):63–73. ISSN 0375-6742. http://www.sciencedirect.com/science/article/pii/S0375674214001009. Computational modeling of fluid flow and geochemical processes in ore-forming and geoenvironmental systems. DOI: 10.1016/j.gexplo.2014.03.006

Iosup A, Ostermann S, Yigitbasi MN, Prodan R, Fahringer T, Epema D. Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Transactions on Parallel and Distributed Systems. 2011; 22(6):931–945. ISSN 1045-9219. DOI: 10.1109/TPDS.2011.66

Ismail L, Khan L. Implementation and performance evaluation of a scheduling algorithm for divisible load parallel applications in a cloud computing environment. Software: Practice and Experience. 2015; 45(6):765–781.

Jackson, KR., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, HJ., Wright, NJ. IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom). IEEE; 2010. Performance analysis of high performance computing applications on the Amazon Web Services cloud; p. 159-168.

Kagadis GC, Kloukinas C, Moore K, Philbin J, Papadimitroulas P, Alexakos C, Nagy PG, Visvikis D, Hendee WR. Cloud computing in medical imaging. Medical Physics. 2013; 40(7):070901.doi: 10.1118/1.4811272 [PubMed: 23822402]

Kaminski B, Szufel P. On optimization of simulation execution on Amazon EC2 spot market. Simulation Modelling Practice and Theory. 2015; 58(Part 2):172–187. ISSN 1569-190X. http://www.sciencedirect.com/science/article/pii/S1569190X15000830. Special issue on Cloud Simulation. DOI: 10.1016/j.simpat.2015.05.008

Keahey, K., Doering, K., Foster, I. Proceedings Fifth IEEE/ACM International Workshop on Grid Computing. IEEE; 2004. From sandbox to playground: Dynamic virtual environments in the grid; p. 34-42.

Keahey K, Foster I, Freeman T, Zhang X. Virtual workspaces: Achieving quality of service and quality of life in the grid. Scientific programming. 2005a; 13(4):265–275.

Keahey, K., Foster, I., Freeman, T., Zhang, X., Galron, D. Euro-Par 2005 Parallel Processing. Springer; 2005b. Virtual workspaces in the grid; p. 421-431.

Laffoon, K. What is new with OnMetal Cloud Servers. 2016. https://support.rackspace.com/how-to/what-is-new-with-onmetal-cloud-servers/

Li, Z., Ge, J., Yang, H., Huang, L., Hu, H., Hu, H., Luo, B. A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. 2016. ISSN 0167-739Xhttp://www.sciencedirect.com/science/article/pii/S0167739X15003982

Mauch V. Deployment of virtual Infiniband clusters with multi-tenancy for cloud computing. Cloud Computing. 2015; 2015:66.

Mell P, Grance T. The NIST definition of cloud computing. 2011

Microsoft. Microsoft SQL Server 2016 Licensing Datasheet. 2016. https://www.microsoft.com/en-us/cloud-platform/sql-server-pricing

Migliore M, Morse TM, Davison AP, Marenco L, Shepherd GM, Hines ML. ModelDB: making models publicly accessible to support computational neuroscience. Neuroinformatics. 2003; 1(1): 135–139. DOI: 10.1385/NI:1:1:135 [PubMed: 15055399]

Migliore M, Cannia C, Lytton WW, Markram H, Hines ML. Parallel network simulations with NEURON. Journal of Computational Neuroscience. 2006; 21(2):119–129. [PubMed: 16732488]

Mishra M, Das A, Kulkarni P, Sahoo A. Dynamic resource management using virtual machine migrations. IEEE Communications Magazine. 2012; 50(9):34–40. ISSN 0163-6804. DOI: 10.1109/MCOM.2012.6295709

Misra J. Distributed discrete-event simulation. ACM Computing Surveys (CSUR). 1986; 18(1):39–65.

Miyasho T, Takagi H, Suzuki H, Watanabe S, Inoue M, Kudo Y, Miyakawa H. Low-threshold potassium channels and a low-threshold calcium channel regulate Ca2+ spike firing in the dendrites of cerebellar Purkinje neurons: a modeling study. Brain Research. 2001; 891(1–2):106–115. [PubMed: 11164813]

Moghadam BT, Alvarsson J, Holm M, Eklund M, Carlsson L, Spjuth O. Scaling predictive modeling in drug development with cloud computing. Journal of Chemical Information and Modeling. 2015; 55(1):19–25. [PubMed: 25493610]

Mogul JC, Popa L. What we talk about when we talk about cloud network performance. ACM SIGCOMM Computer Communication Review. 2012; 42(5):44–48.

Mossucca, L., Zinno, I., Elefante, S., De Luca, C., Goga, K., Terzo, O., Casu, F., Lanari, R. Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS). IEEE; 2015. Performance analysis of the DInSAR P-SBAS algorithm within AWS cloud; p. 469-473.

Mudge, JC., Chandrasekhar, P., Heinson, GS., Thiel, S. IEEE 7th International Conference on E-Science (e-Science). IEEE; 2011. Evolving inversion methods in geophysics with cloud computing—a case study of an eScience collaboration; p. 119-125.

Nusca, A. How Apple's Siri really works. 2011. http://www.zdnet.com/article/how-apples-siri-really-works/

National Science Foundation. Press Release 14–102: Enabling a new future for cloud computing. 2014. https://nsf.gov/news/news.summ.jsp?cntnid=132377. Accessed 05-09-2016

Oesterle F, Ostermann S, Prodan R, Mayr GJ. Experiences with distributed computing for meteorological applications: grid computing and cloud computing. Geoscientific Model Development. 2015; 8(7):2067–2078. DOI: 10.5194/gmd-8-2067-2015

Opara-Martins, J., Sahandi, R., Tian, F. International Conference on Information Society (i-Society). IEEE; 2014. Critical review of vendor lock-in and its impact on adoption of cloud computing; p. 92-97.

Panda, SK., Jana, PK. International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV). IEEE; 2015. A multi-objective task scheduling algorithm for heterogeneous multi-cloud environment; p. 82-87.

Pawluk, P., Simmons, B., Smit, M., Litoiu, M., Mankovski, S. IEEE Fifth International Conference on Cloud Computing. IEEE; 2012. Introducing STRATOS: A cloud broker service; p. 891-898.

Peng Z, Bo XU, Gates AM, Cui D, Lin W. The feasibility and properties of dividing virtual machine resources using the virtual machine cluster as the unit in cloud computing. KSII Transactions on Internet and Information Systems (TIIS). 2015; 9(7):2649–2666.

Rackspace. Rackspace website. 2016a. https://www.rackspace.com/en-us

Rackspace. Rackspace developer home. 2016b. https://developer.rackspace.com/

Rackspace. Rackspace Cloud Files FAQs. 2016c. https://support.rackspace.com/how-to/cloud-files-faq/

Rad P, Chronopoulos AT, Lama P, Madduri P, Loader C. Benchmarking bare metal cloud servers for HPC applications. 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM). 2015; doi: 10.1109/ccem.2015.13

Rehr JJ, Vila FD, Gardner JP, Svec L, Prange M. Scientific computing in the cloud. Computing in Science Engineering. 2010; 12(3):34–43. ISSN 1521-9615. DOI: 10.1109/MCSE.2010.70

Sadooghi I, Martin JH, Li T, Brandstatter K, Zhao Y, Mahesh-wari K, De Lacerda Ruivo TPP, Timm S, Garzoglio G, Raicu I. Understanding the performance and potential of cloud computing for scientific applications. IEEE Transactions on Cloud Computing PP(99). 2015; :1–14. DOI: 10.1109/TCC.2015.2404821

Schneider CJ, Bezaire M, Ivan S. Toward a full-scale computational model of the rat dentate gyrus Structure, function and plasticity of hippocampal dentate gyrus microcircuits. 2015

Segal B, Sanchez CA, Buncic P, Rantala J, Mato P, Blomer J, Quintas DG, Weir DJ, Yao Y, Harutyunyan A. LHC cloud computing with CernVM. PoS, 004. 2010

Sharma, M. A virtual appliance primer. 2008. https://www.linux.com/news/virtual-appliance-primer. Accessed 5-20-2016

Silverstein D, Lansner A. Scaling of a biophysical neocortical attractor model using Parallel NEURON on the Blue Gene/P. BMC Neuroscience. 2011; 12(Suppl 1):191.

Subhashini S, Vadim A, Kenneth Y, Ted C, Maryann M, Amit M, Anita B. A Neuroscience Gateway — software and implementation. Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery. 2013a; doi: 10.1145/2484762.2484816

Subhashini, S., Amit, M., Kenneth, Y., Vadim, A., Anita, B., MaryAnn, M., Nicholas, TC. Introducing the Neuroscience Gateway. IWSG vol .993 of CEUR Workshop Proceedings: Citeseer. 2013b. CEUR-WS.org

Subhashini S, Amit M, Kenneth Y, Vadim A, Anita B, MaryAnn M, Nicholas C, et al. Early experiences in developing and managing the neuroscience gateway. Concurrency and Computation: Practice and Experience. 2015; 27(2):473–488. [PubMed: 26523124]

Sliman L, Charroux B, Stroppa Y. A new collaborative and cloud based simulation as a service platform: Towards a multidisciplinary research simulation support. IEEE 2013 UKSim 15th International Conference on Computer Modelling and Simulation (UKSim). 2013:611–616.

Smit M, Pawluk P, Simmons B, Marin L. A web service for cloud metadata. Services (SERVICES) IEEE Eighth World Congress on IEEE. 2012:361–368.

Smith MS. Nuclear data for astrophysics research: A new online paradigm. Journal of the Korean Physical Society. 2011; 59(2):761–766.

Stockton DB, Santamaria F. Neuromanager: A workflow analysis based simulation management engine for computational neuroscience. Frontiers in Neuroinformatics. 2015; 9(24) ISSN 1662-5196. doi: 10.3389/fninf.2015.00024

Taylor RP, Berghaus F, Brasolin F, Cordeiro CJD, Desmarais R, Field L, Gable I, Giordano D, Di Girolamo A, Hover J, LeBlanc M, Love P, Paterson M, Sobie R, Zaytsev A. The evolution of cloud computing in ATLAS. Journal of Physics: Conference Series. 2015; 664(2):022038. http://stacks.iop.org/1742-6596/664/i=2/a=022038.

Teka W, Stockton DB, Santamaria F. Power-law dynamics of membrane conductances increase spiking diversity in a Hodgkin–Huxley model. PLoS Computer Biology. 2016; 12(3)doi: 10.1371/journal.pcbi.1004776

Texas Advanced Computing Center. TACC Stampede website. 2016. https://www.tacc.utexas.edu/stampede/

Thackston, R., Fortenberry, R. High performance computing: Considerations when deciding to rent or buy. SAIS 2015 Proceedings, page Paper 16. 2015a. http://aisel.aisnet.org/sais2015/16

Thackston R, Fortenberry RC. The performance of low-cost commercial cloud computing as an alternative in computational chemistry. Journal of Computational Chemistry. 2015b; 36(12):926–933. [PubMed: 25753841]

Tihfon, GM., Kim, J., Kim, KJ. Information Science and Applications (ICISA) 2016, chapter A New Virtualized Environment for Application Deployment Based on Docker and AWS. Singapore: Springer Singapore; 2016. p. 1339-1349.Lecture Notes in Electrical Engineering 376

UTSA. Open Cloud Institute website. 2016. http://opencloud.utsa.edu/

Vaquero LM, Rodero-Merino L, Buyya R. Dynamically scaling applications in the cloud. ACM SIGCOMM Computer Communication Review. 2011; 41(1):45–52.

Wittek P, Rubio-Campillo X. Scalable agent-based modelling with cloud HPC resources for social simulations. 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom). 2012; :355–362. DOI: 10.1109/CloudCom.2012.6427498

Wu H, Ren S, Timm S, Garzoglio G, Noh SY. Overhead-Aware-Best-Fit (OABF) resource allocation algorithm for minimizing VM launching overhead. 7th IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS). 2014

Yamazaki T, Ikeno H, Okumura Y, Satoh S, Kamiyama Y, Hirata Y, Inagaki K, Ishihara A, Kannon T, Usui S. Reprint of: Simulation Platform: A cloud-based online simulation environment. Neural Networks. 2011; 24(9):927–932. ISSN 0893-6080. http://www.sciencedirect.com/science/article/pii/S0893608011002255. DOI: 10.1016/j.neunet.2011.08.007 [PubMed: 21944492]

Yelick, K., Coghlan, S., Draney, B., Canon, RS., et al. The Magellan report on cloud computing for science. US Department of Energy, Office of Science Office of Advanced Scientific Computing Research (ASCR); 2011.

Yoginath, SB., Perumalla, KS. Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques, Simu-Tools '13, pages 1–9, ICST. Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering); 2013. Optimized hypervisor scheduler for parallel discrete event simulations on virtual machine platforms. http://dl.acm.org/citation.cfm?id=2512734.2512735

Yoginath SB, Perumalla KS. Efficient Parallel Discrete Event Simulation on cloud/virtual machine platforms. ACM Transactions on Modeling and Computer Simulation (TOMACS). 2015; 26(1):5.

Zaspel P, Griebel M. Massively parallel fluid simulations on Amazon's HPC cloud. Network Cloud Computing and Applications (NCCA) First International Symposium on IEEE. 2011:73–78.

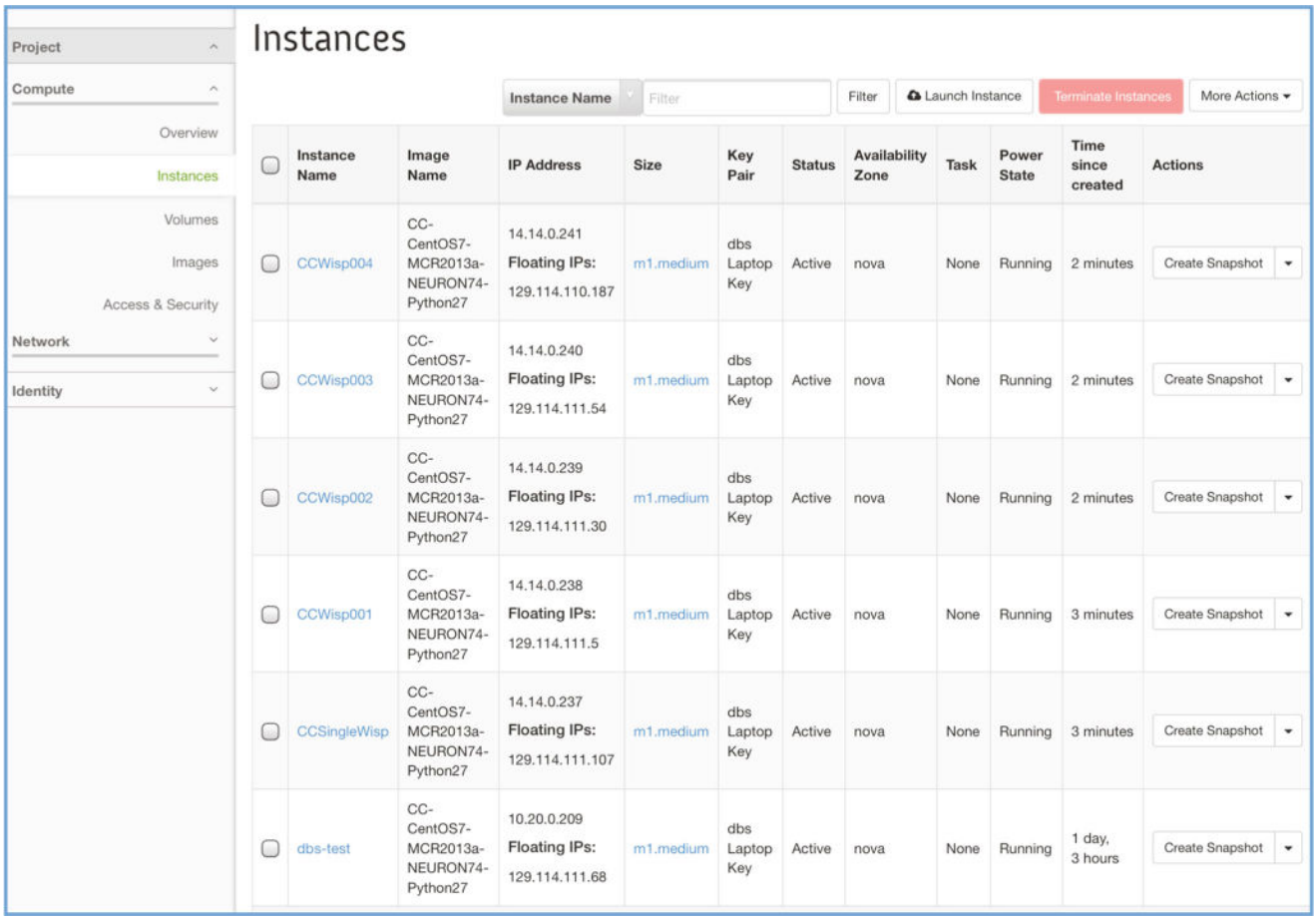

Fig. 1.

Cloud server management via browser-based dashboard. The Chameleon Cloud dashboard's 'Instances' page shows an instance, 'dbs–test', created using the dashboard, and five other instances which were created with NeuroManager's cloud management classes. All were created using the same image as seen in the "Image Name" column; the image itself was produced by configuring an instance based on a stock CentOS-7 image using a NEURON/Python–specific script. All instances have public ("floating") IP addresses and each makes use of the same key pair (here called 'dbs Laptop Key') for dual key authentication and communications encryption

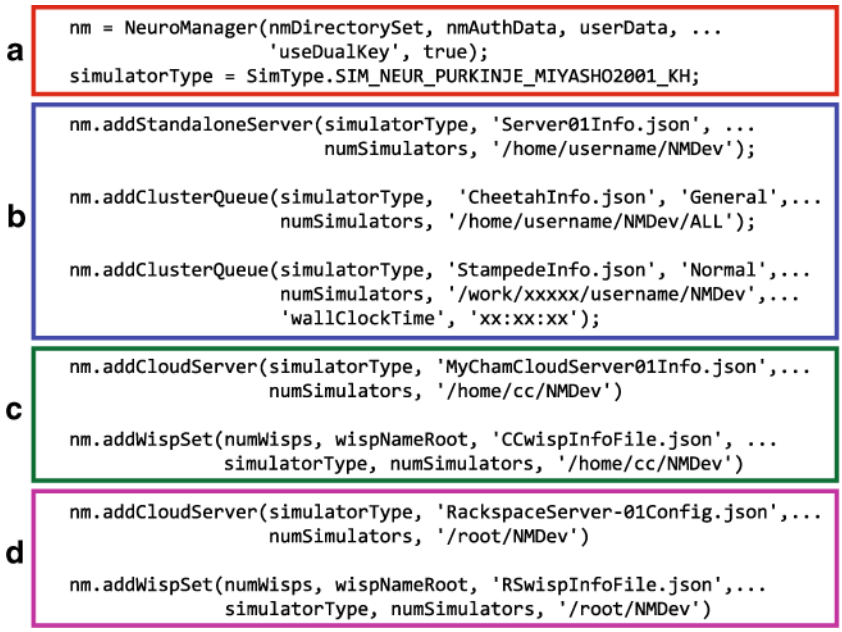

Fig. 2.

NeuroManager configuring resources. **a**) Line 1: Constructing the NeuroManager object. Line 2: Specifying the type of Simulators to be constructed. **b**) Specifying Simulators to be added to a standalone server (Line 1), institutional HPC cluster (Line 2) and remote supercomputer cluster (Line 3, Stampede cluster at Texas Advanced Computer Center). **c**–**d**) Adding cloud servers: Chameleon (**c**) or Rackspace (**d**). The servers can be persistent, created and terminated by user, or ephemeral (Wisps), automatically created and destroyed by NeuroManager. In all cases the "…Info.json" file holds localizing information about the resource. In this illustration, assuming *numSimulators* = 4 and *numWisps* = 3, we have 5 fixed and 6 ephemeral machines, each hosting 4 Simulators, for a total of 44 Simulators. For details, see text and NeuroManager documentation

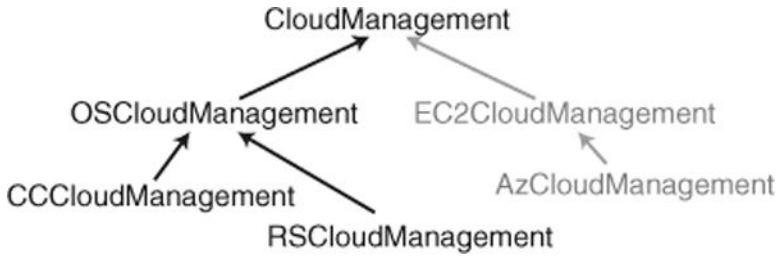

Fig. 3.

Cloud server management classes. NeuroManager implements instance management through use of the cloud's API using JSON configuration files and a class hierarchy that together accommodate differences in settings and methodology. This gives the researcher the ability to ignore cloud differences in day–to–day simulation activities. The *CloudManagement* class specifies the interface that the subclasses must implement; the *OSManagement* class implements the interface for OpenStack clouds, and the *CCCloudManagement* and *RSCloudManagement* classes take specific OS cloud differences into account. The Amazon classes are under implementation and thus shown in grey. Arrows point to a class's superclass

NeuroManager 0.961

Input SimSpec file: ...\AutoSpecs\CloudPaperRun.txt
Size of Simulator Pool: 18
SimSet "CloudPaperRun" status at 17-May-2016 17:18:46

SIMID	State	Submission Time	Run Start Time	Job ID	Simulator	Machine	Result	
CPR0000	FULLYPROCESSED	17-May-2016 17:06:35	17-May-2016 17:06:38	1048	synapse01	synapse	Success	← *Standalone Server*
CPR0001	FULLYPROCESSED	17-May-2016 17:06:53	17-May-2016 17:06:56	2338	synapse02	synapse	Success	
CPR0002	DOWNLOADING	17-May-2016 17:06:56	17-May-2016 17:07:09	97030	CheetahGeneral01	CheetahGeneral	?	← *UTSA Cluster - General Queue*
CPR0003	RUNNING	17-May-2016 17:07:01	17-May-2016 17:07:26	97031	CheetahGeneral02	CheetahGeneral	?	
CPR0004	FULLYPROCESSED	17-May-2016 22:07:06	17-May-2016 22:07:10	14563	dbs-test01	dbs-test	Success	← *Persistent Chameleon Cloud Server*
CPR0005	FULLYPROCESSED	17-May-2016 22:07:08	17-May-2016 22:07:12	15003	dbs-test02	dbs-test	Success	
CPR0006	FULLYPROCESSED	17-May-2016 22:07:11	17-May-2016 22:07:18	3577	CCSingleWisp01	CCSingleWisp	Success	← *Ephemeral Chameleon Cloud Server Single*
CPR0007	FULLYPROCESSED	17-May-2016 22:07:15	17-May-2016 22:07:20	4216	CCSingleWisp02	CCSingleWisp	Success	
CPR0008	FULLYPROCESSED	17-May-2016 22:07:19	17-May-2016 22:07:26	3515	CCWisp00101	CCWisp001	Success	← *Ephemeral Chameleon Cloud Server Set*
CPR0009	FULLYPROCESSED	17-May-2016 22:07:24	17-May-2016 22:07:28	4210	CCWisp00102	CCWisp001	Success	
CPR0010	FULLYPROCESSED	17-May-2016 22:07:29	17-May-2016 22:07:36	3472	CCWisp00201	CCWisp002	Success	
CPR0011	FULLYPROCESSED	17-May-2016 22:07:35	17-May-2016 22:07:39	4188	CCWisp00202	CCWisp002	Success	
CPR0012	RUNCOMPLETE	17-May-2016 22:07:39	17-May-2016 22:07:46	1613	RSdbs-test01	RSdbs-test	?	← *Persistent Rackspace Cloud Server*
CPR0013	RUNNING	17-May-2016 22:07:46	17-May-2016 22:07:53	2290	RSdbs-test02	RSdbs-test	?	
CPR0014	RUNNING	17-May-2016 22:07:55	17-May-2016 22:08:04	2876	RSWisp00101	RSWisp001	?	← *Ephemeral Rackspace Cloud Server Set*
CPR0015	RUNNING	17-May-2016 22:08:02	17-May-2016 22:08:09	3540	RSWisp00102	RSWisp001	?	
CPR0016	RUNNING	17-May-2016 22:08:10	17-May-2016 22:08:19	2893	RSWisp00201	RSWisp002	?	
CPR0017	RUNNING	17-May-2016 22:08:17	17-May-2016 22:08:24	3557	RSWisp00202	RSWisp002	?	
CPR0018	RUNNING	17-May-2016 17:11:09	17-May-2016 17:11:20	29814	synapse02	synapse	?	
CPR0019	RUNNING	17-May-2016 17:12:18	17-May-2016 17:12:28	2861	synapse01	synapse	?	
CPR0020	UNRUN	30-Nov-9999 00:00:00	30-Nov-9999 00:00:00	------	---	---	?	

Fig. 4.

A single NeuroManager session: running a set of simulations on multiple varied computational resources. In this example two Simulators were hosted by each server. SIMID: simulation id; State: current state of the simulation; Simulator: the name of the Simulator the simulation is running on; Machine: name of physical resource the Simulator is hosted on; Result: final status of simulation (? = not known yet). We added annotation on the right to describe the resources: standalone server, cluster, or cloud server. The persistent cloud servers were running before NeuroManager was launched and remained running after NeuroManager concluded. The ephemeral servers were created by on the fly by NeuroManager and autmatically terminated at the end of the session

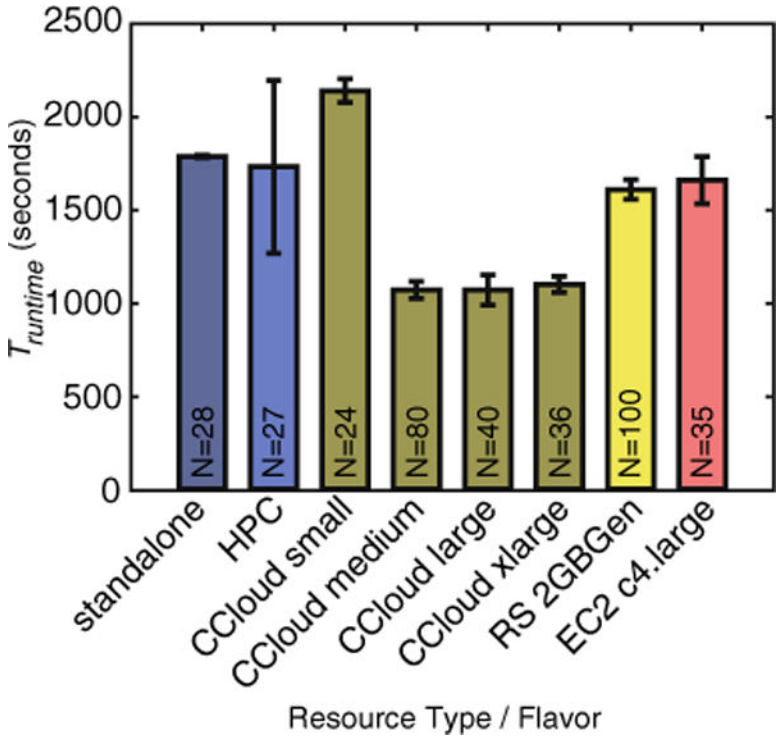

Fig. 5.

Individual simulation runtime dependence on resource and flavor. The mean and standard deviation of $T_{runtime}$ for the NEURON simulation (see text for details) on a standalone server, HPC, four flavors of Chameleon Cloud server (CC), one flavor of Rackspace cloud server (RS), and one flavor of Amazon EC2 (EC2). For these measurements we used two Simulators per cloud server; N is the number of simulations

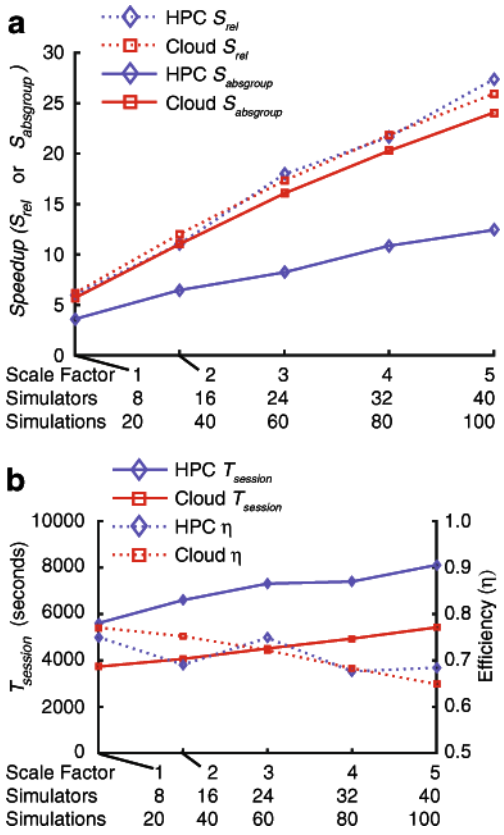

Fig. 6.

Scaling, speedup, and efficiency in cloud and HPC resources. **A** Session speedup vs scale factor. The speedup was calculated relative to the fastest simulation within the specific HPC or cloud session (Srel), or the fastest simulation across all sessions within the group composed of Sets 1 and 2 (Sabsgroup). **b** Session time, $T_{session}$, and efficiency, η, vs scale factor. All values shown are the averages of 5 trials (HPC) or 3 trials (Chameleon Cloud). See text for additional details and discussion

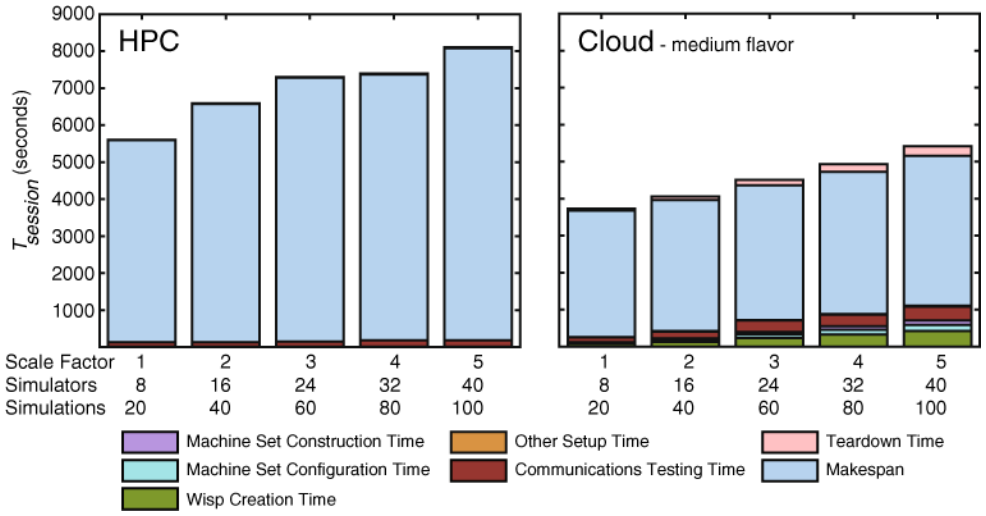

Fig. 7.

$T_{session}$ breakdown vs. scale factor for HPC and Cloud. On the left, all Simulators are located only on a single institutional HPC cluster with one core per Simulator. On the right, all the Simulators are located on two–core Wisps with two Simulators per Wisp; setup involves Wisp creation in parallel, testing of SSH communications and file transfer, and construction of the Simulators on multiple Wisps. All values shown are the averages of 5 trials (HPC) or 3 trials (Chameleon Cloud). See text for additional details and discussion

Table 1

Effects of flavor on $T_{session}$ and cost

Session	Chameleon Cloud flavor	Cores, Simulators per server	Number of servers	Number of simulations	Setup time	Average simulation runtime	$T_{session}$	Core-equivalent Rackspace monthly cost	Session cost	Cost per simulation
4.1	small	1	16	40	905	1020.4	4514	$23.36	$0.58	$0.01
4.2	medium	2	8	40	419	1053.2	4032	$46.72	$0.54	$0.01
4.3	large	4	4	40	272	1073.2	3826	$93.44	$0.52	$0.01
4.4	xlarge	8	2	40	228	1143.6	4030	$186.88	$0.55	$0.01
4.5	small	1	16	160	939	1028.6	12950	$23.26	$1.75	$0.01
4.6	xlarge	8	2	160	196	1149.7	12841	$186.88	$1.77	$0.01
4.7	small	1	16	40	800	3989.2	13262	$23.36	$1.79	$0.04
4.8	xlarge	8	2	40	201	4428.3	13853	$186.88	$1.91	$0.05

Sessions 4.1–4.4: Effects of flavor choice when running fixed numbers of Simulators and simulations. Sessions 4.5–4.6: Effects of increasing number of simulations (4×) on two flavors. Sessions 4.7–4.8: Effects of increasing the simulation and saved files sizes (4×). All simulations run on Chameleon Cloud. Costs were obtained from Rackspace for similar flavors. All times in seconds. See text for more details