AFRL-RI-RS-TR-2011-211

HIGH PERFORMANCE COMPUTING (HPC) INNOVATION SERVICE PORTAL PILOTS CLOUD COMPUTING (HPC-ISP PILOT CLOUD COMPUTING)

UNIVERSITY OF SOUTHERN CALIFORNIA – INFORMATION SCIENCES INSTITUTE

AUGUST 2011

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2011-211 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
CHRISTOPHER J. FLYNN
Work Unit Manager

/s/
RICHARD MICHALAK, Acting Technical Advisor
Advanced Computing Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
AUG 2011	Final Technical Report	MAR 2010 – MAR 2011

4. TITLE AND SUBTITLE

HIGH PERFORMANCE COMPUTING (HPC) INNOVATION SERVICE PORTAL PILOTS CLOUD COMPUTING (HPC-ISP PILOT CLOUD COMPUTING)

5a. CONTRACT NUMBER
FA8750-10-C-0112

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
63760E / 62304E

6. AUTHOR(S)

Lorin Hochstein

5d. PROJECT NUMBER
B011

5e. TASK NUMBER
US

5f. WORK UNIT NUMBER
CE

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California – Information Sciences Institute
3811 North Fairfax Drive, Suite 200
Arlington, VA 22203-1714

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITB
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2011-211

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2011-4328
Date Cleared: 10 AUG 2011

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The objective of part 1 of the research effort was to perform three capability demonstrations on a brokerage infrastructure in a commercial cloud computing environment, to engage multiple computing service providers and application software vendors and collect comparative metrics to evaluate the return-on-investment (ROI) argument for high performance computing within the DoD supply chain and to evaluate the startup costs saved of using a broker. The demos were:1)AltaSim Ghost on Ohio Supercomputer Center, with an estimated ROI of 252X and brokerage startup cost savings of 90% and ten months of time, 2)ANSYS on R Systems, with an estimated ROI of 14.7X and brokerage startup costs savings of 91% and seven weeks of time, 3)Mathematica and Blender on Amazon EC2, with an estimated ROI of 27.4X and brokerage startup cost savings of 96% and one month of time. The objective of part 2 of this research effort was to investigate the use of probabilistic computer architectures to solve computational challenge problems at orders of magnitude lower cost and increased processing capability relative to conventional computer architectures. 1000x+ speedups with 10-30x less power usage using a software-reprogrammable probabilistic video processor were documented.

15. SUBJECT TERMS
High Performance Computing, DoD Manufacturing Supply Chains, Modeling and Simulation, Cloud computing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			CHRISTOPHER J. FLYNN
U	U	U	UU	30	19b. TELEPHONE NUMBER (*Include area code*) N/A

Table of Contents

List of Figures

List of Tables

1 Summary

This is the final report for *HPC-ISP PILOTS CLOUD COMPUTING* contract #FA8750-10-C-0112, which includes the period of performance from March 24, 2010 to March 23, 2011. The first objective was to select three of the four pilot capability demonstrations from the High Performance Computing-Innovative Service Portal (HPC-ISP) - PILOTS project (FA8750-08-C-0184) [Schott11] and migrate them to a brokerage infrastructure in a commercial cloud- computing environment. The purpose of this activity is to engage multiple computing service providers and applications software vendors and collect comparative metrics to further strengthen the Return-On-Investment (ROI) argument for high performance computing within the DoD supply chain. The computing and software providers engaged under HPC-ISP-PILOTS will be utilized but instead of dedicated hosted service, the companies will be connected through an HPC brokerage service.

The three capability demonstrations were:

- AltaSim Ghost, a parallel circuit simulation package, using Ohio Supercomputer Center as an HPC hardware provider. The metrics analysis suggested an estimated ROI of 252X. The use of a brokerage service yielded estimated startup savings of $12,713 (99%) and a reduction of about ten months in startup time.
- ANSYS, a computational structural mechanics package, using R Systems as an HPC hardware provider. The metrics analysis suggested an estimated ROI of 14.7X. The use of a brokerage service yielded estimated startup savings of $2,918 (91%) and a reduction of about seven weeks in startup time.
- Mathematica, a computational mathematics package, and Blender, an image rendering package, using Amazon's Elastic Compute Cloud (EC2) as an HPC hardware provider. The metrics analysis suggested an estimated ROI of 27X. The use of a brokerage service yielded estimated startup savings of $1,837 (96%), and a reduction of about one month in startup time.

The research outcomes from the first objective are documented in Sections 2-5.

The second objective was to investigate the use of probabilistic computer architectures to solve computational challenge problems at orders of magnitude lower cost and increased processing capability relative to conventional computer architectures. The computational problems addressed in this effort were:

- Real-time motion detection
- Real-time clustering and classification of streaming data

The research outcomes from the second objective are described in Section 6.

2 Introduction

Cloud computing has recently emerged as a business model to allow users to temporarily gain access to a large set of computational resources by renting resources from a cloud computing provider [Armburst09]. The most prominent example of a cloud computing provider is Amazon's Elastic Compute Cloud [EC2], although many other providers have emerged. Cloud computing emerged from the IT industry and was based on commodity hardware. While certain computational problems are very amenable to being run in this fashion [Dean04], other applications (e.g., finite element analysis, computational fluid dynamics) do not perform as well on EC2 because of the absence of a high-speed interconnect. A number of cloud computing providers oriented towards high-performance computing (HPC) have emerged, such as R Systems [RSys], Penguin on Demand [PoD], and SGI Cyclone [Cyc].

There are several cloud-computing models for providing access to computing resources as a service. In the infrastructure as a service (IaaS) model, the end-user gets access to low-level computing resources and is responsible for installing and configuring all of the software. This is the most flexible model but requires the most amount of IT expertise on the part of the end-user. In the platform-as-a-service (PaaS) model, the end-user is a software developer who can deploy Internet-based applications onto the cloud.

Examples of the PaaS model include Google App Engine [GAE] and Windows Azure [Azure]. In the software-as-a-service (SaaS) model, the software is hosted in the cloud and the user accesses the software over the Internet, typically over a standard web browser. The canonical example of SaaS is Salesforce [Salesforce], a customer relationship management (CRM) software solution.

For manufacturing companies, the two models of interest are IaaS and SaaS, since PaaS is intended only for software developers. However, IaaS requires substantial IT expertise to configure the systems, in addition to the issues involved in acquiring the licenses for use on a third-party system. The effort described in this report examines the use of an SaaS-based cloud access model, where the end-users access systems that have already been configured with the engineering software. In particular, this effort focused on a brokerage-based approach where the broker provides service-level access to HPC resources, and contracts separately with both the hardware and software vendors to obtain the resources. The broker develops the web interfaces needed to provide access to the HPC software as a service. The broker involved in this study was Nimbis Services.

This effort follows from HPC-ISP-PILOTS (FA8750-08-C-0184), a set of pilot studies to demonstrate the benefits of the use of HPC for modeling and simulation in small and medium sized companies in the DoD supply chain [Schott11]. In HPC-ISP-PILOTS, we were able to quantify the potential benefits of companies leveraging HPC to do modeling and simulation, but all of the resources (hardware, software, labor) were provided by the DoD funding. A consequence of this approach is that the true costs were not captured.

Three of the four pilot studies from HPC-ISP-PILOTS were selected to show that the software involved could be made accessible through a brokerage service when running on an HPC-based backend. The three demonstrations were:

1. AltaSim Ghost on Ohio Supercomputer Center (AltaSim Technologies)
2. ANSYS on R Systems (Woodward FST)
3. Blender, Mathematica on Amazon EC2 (ACE Clearwater)

3 Methods, Assumptions, and Procedures

To support the three demonstrations, Nimbis Services provided access to the software through a custom website at http://metrix.nimbis.net.

3.1 AltaSim Ghost on Ohio Supercomputer Center

In the original HPC-ISP-PILOTS effort, AltaSim Technologies and Ohio Supercomputer Center (OSC) used the Xyce parallel circuit simulator tool [Hutchinson01] from Sandia National Laboratories to simulate the electromagnetic interference generated by power node control (PNCC) circuits intended for use in naval vessels. AltaSim Technologies has licensed the Xyce technology from Sandia and re-branded it as AltaSim Ghost. Nimbis Services ensured that Ghost was installed on an HPC provider: in this case, it was OSC.

Figure 1 shows the system architecture for this setup. The setup consists of a persistent login node managed by Nimbis located at OSC, as well as compute nodes that can be allocated on-demand as needed. The login node is connected to the cluster through a gigabit Ethernet (1GbE) network connection. The login node runs Redhat Enterprise Linux (RHEL) 5, and has several responsibilities. It manages the Ghost licenses, runs the compute node scheduler, and serves as temporary storage for user job data.

The compute nodes, which actually execute the software, are referred to as OSC Glenn 1 or G-1 nodes. The user may allocate from 1 to 256 nodes, each with dual 2.6 GHz Opteron 64-bit dual core processors, 4 cores per node, 8 GB memory, 48 GB file storage, RHEL 5, and 4X SDR (10 Gbps) Infiniband node interconnect, running 64-bit GHOST for SPICE based simulations of electrical and electronic circuits.

Nimbis provided a web interface where users can upload a Ghost file (simple example shown in Figure 2), and download a results file.

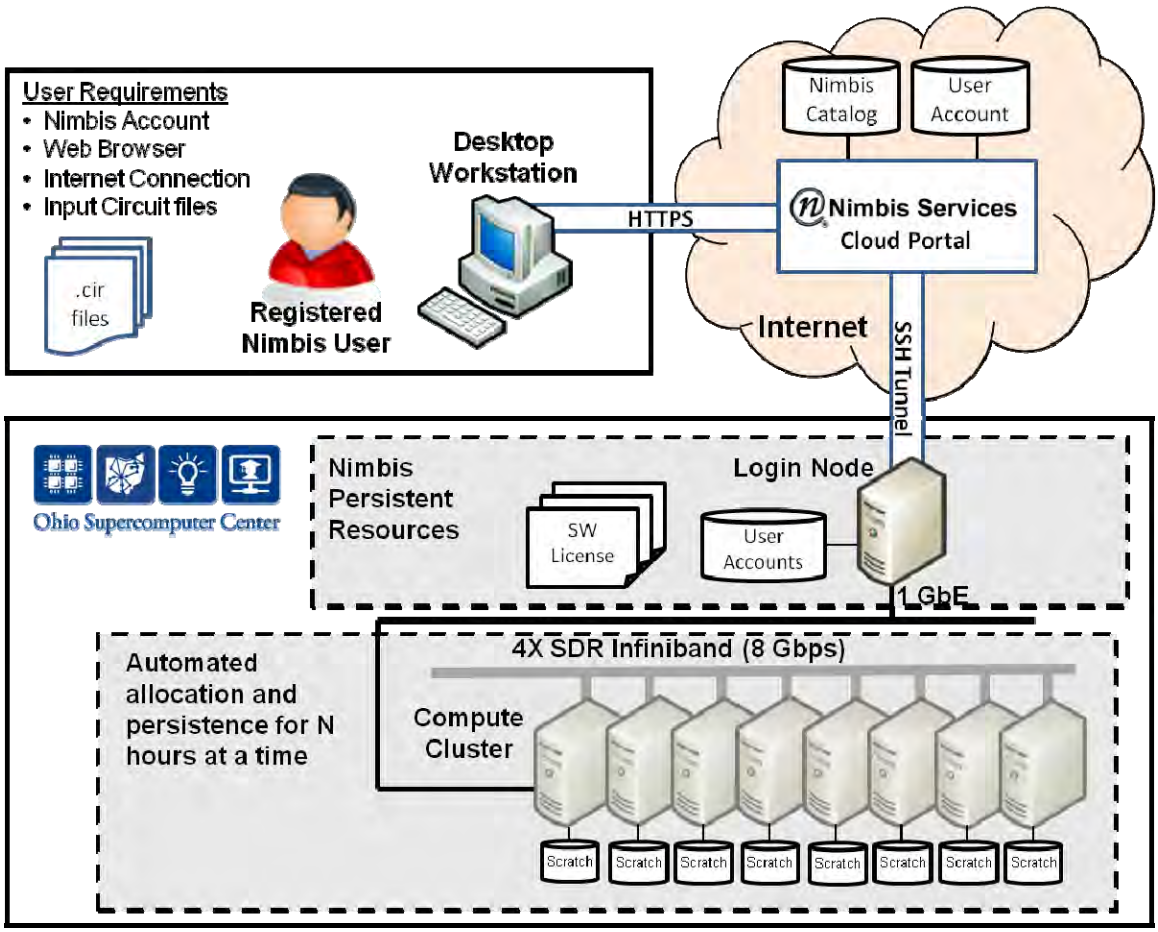

Figure 1 Architectural diagram of running Ghost on OSC through Nimbis

```
Diode Clipper Circuit with transient analysis statement
*
* Voltage Sources
VCC 1 0 5V
VIN 3 0 SIN(0V 10V 1kHz)
*Analysis Command
.TRAN 2ns 2ms
* Output
.PRINT TRAN V(3) V(2) V(4)
* Diodes
D1 2 1 D1N3940
D2 0 2 D1N3940
* Resistors
R1 2 3 1K
R2 1 2 3.3K
R3 2 0 3.3K
R4 4 0 5.6K
* Capacitor
C1 2 4 0.47u
*
* GENERIC FUNCTIONAL EQUIALENT = 1N3940
* TYPE: DIODE
* SUBTYPE: RECTIFIER
.MODEL D1N3940 D(
+     IS = 4E-10
+     RS = .105
+     N = 1.48
+     TT = 8E-7
+     CJO = 1.95E-11
+     VJ = .4
+     M = .38
+     EG = 1.36
+     XTI = -8
+     KF = 0
+     AF = 1
+     FC = .9
+     BV = 600
+     IBV = 1E-4)
*
.END
```

Figure 2 Example of a simple Ghost input file

3.2 ANSYS on R Systems

In the original HPC-ISP-PILOTS effort, engineers from Woodward FST ran structural analysis simulations using the ANSYS [ANSYS] software package on IBM's Computing on Demand service.

The original plan in this research effort was to obtain access to IBM's Computing on Demand service through the Nimbis web portal. Unfortunately, during the period of performance of this project, IBM discontinued their Computing on Demand service, and we were forced to identify an alternative HPC provider that could run ANSYS in an ITAR environment. Ironically, this event demonstrated one of the benefits of using a broker: the Nimbis front-end interface to the system remains the same to the user, even though the back-end cycle provider changed.

Figure 3 shows the system architecture for this setup. The setup consists of a persistent login node managed by Nimbis located at R Systems, as well as dedicated compute nodes that are allocated on a week-to-week basis. Users access the login node through a VPN connection using OpenVPN [Feilner06]. The login node runs RHEL 5.4, has eight 2.8 GHz cores, 32 GB of RAM, and a 4X quad data rate (QDR) Infiniband connection to the cluster. The login node manages the ANSYS licenses, runs the VNC and web server interface, and provides access to the Torque [Torque] job queue that is used to launch jobs.

There are eight compute nodes, which are used for executing the ANSYS software. Each node has eight 2.8 GHz cores, 32 GB RAM, 250 GB local storage (with 127 GB user scratch storage) and a 4X QDR Infiniband node interconnect.

Nimbis offers several different mechanisms for allowing users to run ANSYS on the remote system. A web-based portal is available to submit jobs, download results files, and do some limited post-processing. In addition, a VNC client can be used to run ANSYS remotely and do interactive post-processing directly on the remote machine. The user can install a VNC client locally, or can use a web-based VNC client provided by Nimbis.

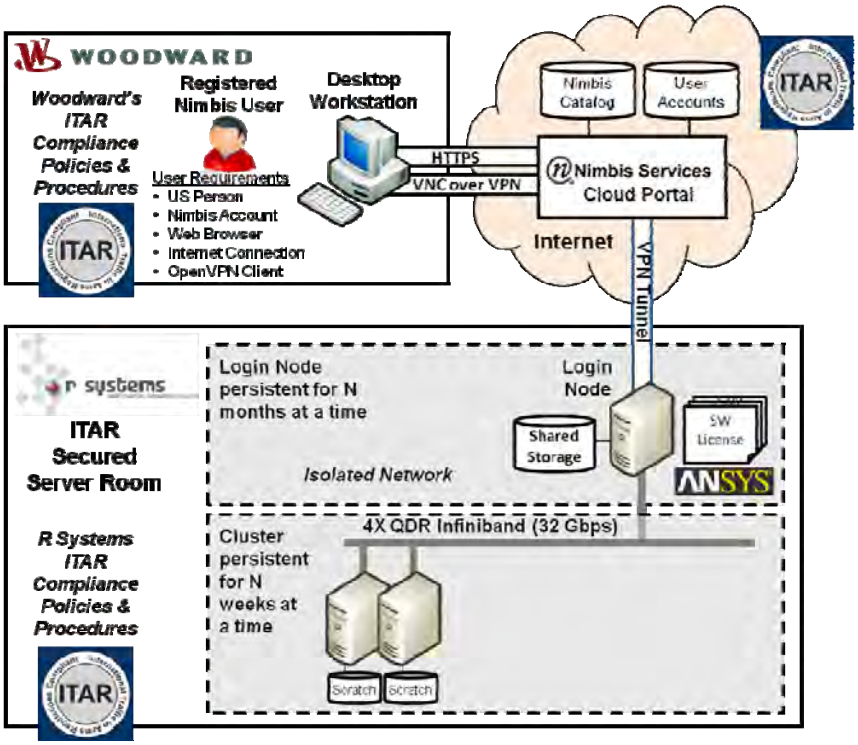

Figure 3 Architectural diagram of running ANSYS on R Systems through Nimbis

3.3 Mathematica and Blender on Amazon EC2

In the original HPC-ISP-PILOTS effort, engineers from ACE Clearwater worked with graduate students at University of Southern California's Western Research Applications Center (WESRAC) on a design clinic approach.

As a parallel activity to this effort, WESRAC continued working with ACE Clearwater as a model for design clinics. By combining Hollywood modeling/simulation with manufacturing modeling/simulation, new synergistic job opportunities are created for small manufacturers, software vendors, graduate students and Hollywood animators. This approach provides small manufacturers with access to an existing well-trained Southern California modeling talent pool. The ability to combine desktop pre- and post-processing platforms with high performance cloud computing resources eliminates the need for high capital investments in hardware and software. This approach creates a new computing utility which can be scaled to meet the dynamic demand of production schedules.

Two post-processing tools were used in this study: Mathematica and Blender. Mathematica would be used to create the simulations. Blender would be used to create rendered animation and visual representations of data collected from simulations.

Figure 4 and Figure 5 show the system architecture for this setup, for Blender and Mathematica, respectively. In both cases, the number of (virtual) cores and the size of the memory will vary depending on the Amazon EC2 instance type selected by the user when the job is launched. Information about the different instance types available can be found on Amazon's website[1].

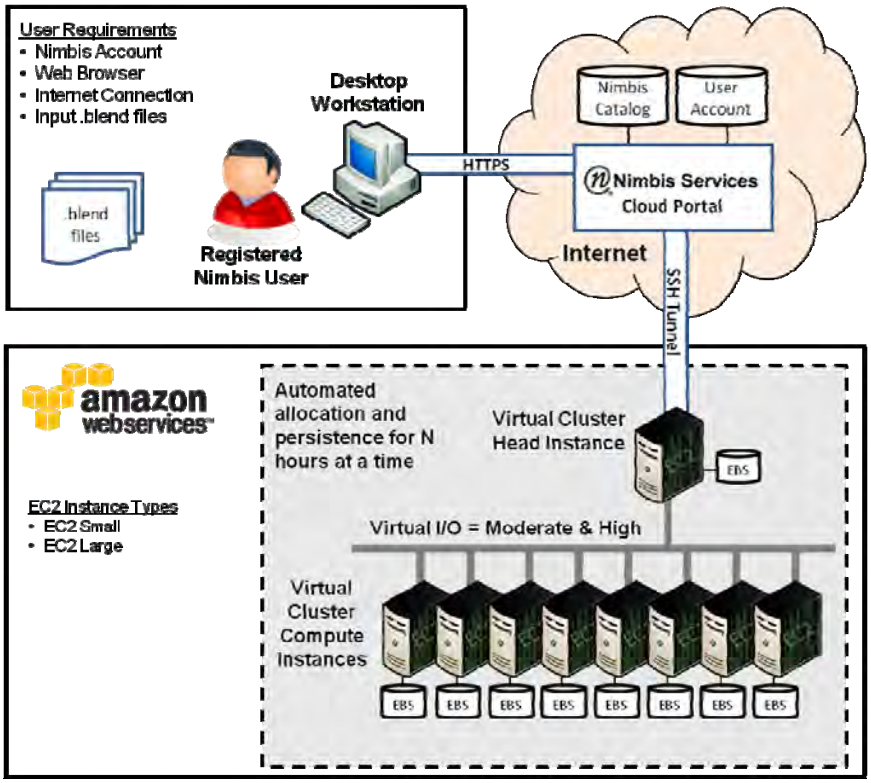

Figure 4 Architetural diagram of running Blender on Amazon EC2 through Nimbis

[1] http://aws.amazon.com/ec2/instance-types

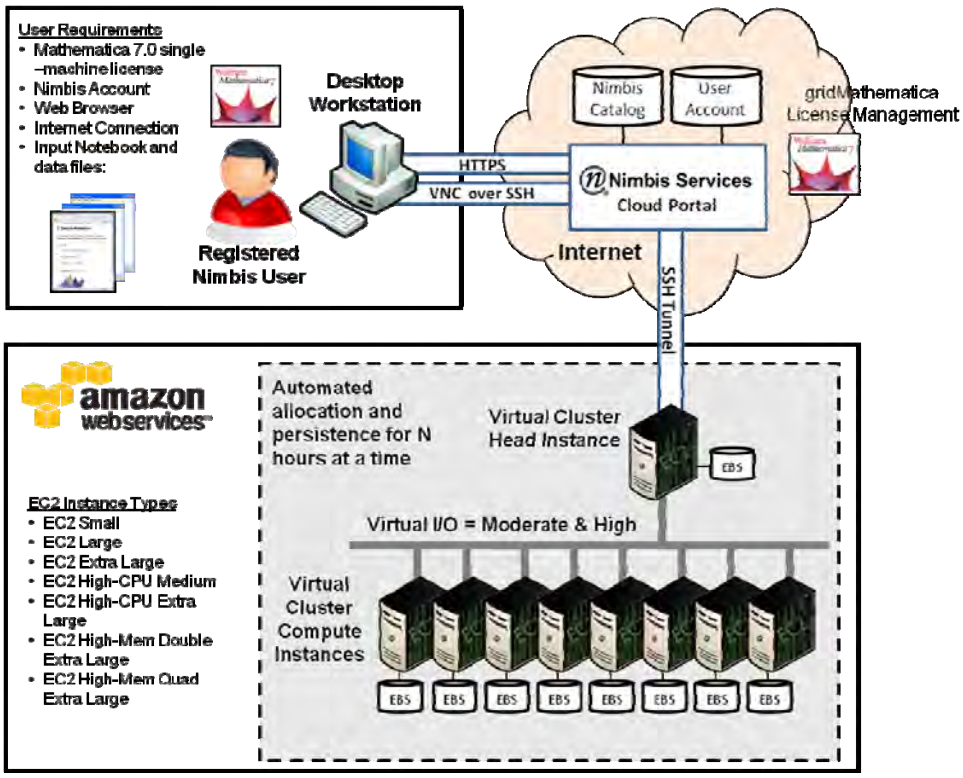

Figure 5 Architectural diagram of running Mathematica on Amazon EC2 through Nimbis

3.4 Evaluation methodology

For each capability demonstration, we evaluated whether the Nimbis interface supported running the appropriate HPC job on the remote HPC provider. In addition, we performed user studies to identify how long it would take to use the interface for the metrics activities. A byproduct of this evaluation was the identification of usability issues, which we fed back to Nimbis to help improve the interface. The results of these evaluations are described in Section 4.

3.5 Metrics analysis

The main goal of this project was to provide estimates of the return on investment (ROI) for companies to use HPC, assuming a brokerage interface. In particular, one goal was to contrast the startup time and cost of using a brokerage service to provide access to the hardware and software, versus the cost of an end-user negotiating directly with the hardware cycle provider, software provider, and then providing the required IT labor to set up the system for use.

To compute the ROI, we use the follow equation:

$$ROI = \frac{Benefits}{Costs} - 1 = \frac{Benefits}{Startup\ costs + Variable\ costs} - 1$$

The desire is to achieve a positive ROI: if benefits are equal to costs, then the ROI is 0.

We break the cost down into two components: startup costs and variable costs. The startup costs incorporate all of the costs incurred before running the first HPC job, and the variable costs are a function of the amount of compute time used.

Figure 6 shows a flowchart that captures the activities required to get started running engineering software on a remote HPC system, for the direct case (contracting directly with hardware and software providers), as well as going through a broker.

This startup activity requires both time (e.g., waiting to get access) as well as labor. The labor is highlighted in yellow. For the direct method, the end-user needs access to IT labor, specialized in HPC, that can install and configure the software on the hardware. For the brokerage method, the end-user needs to expend engineering labor to learn the Nimbis web portal interface.

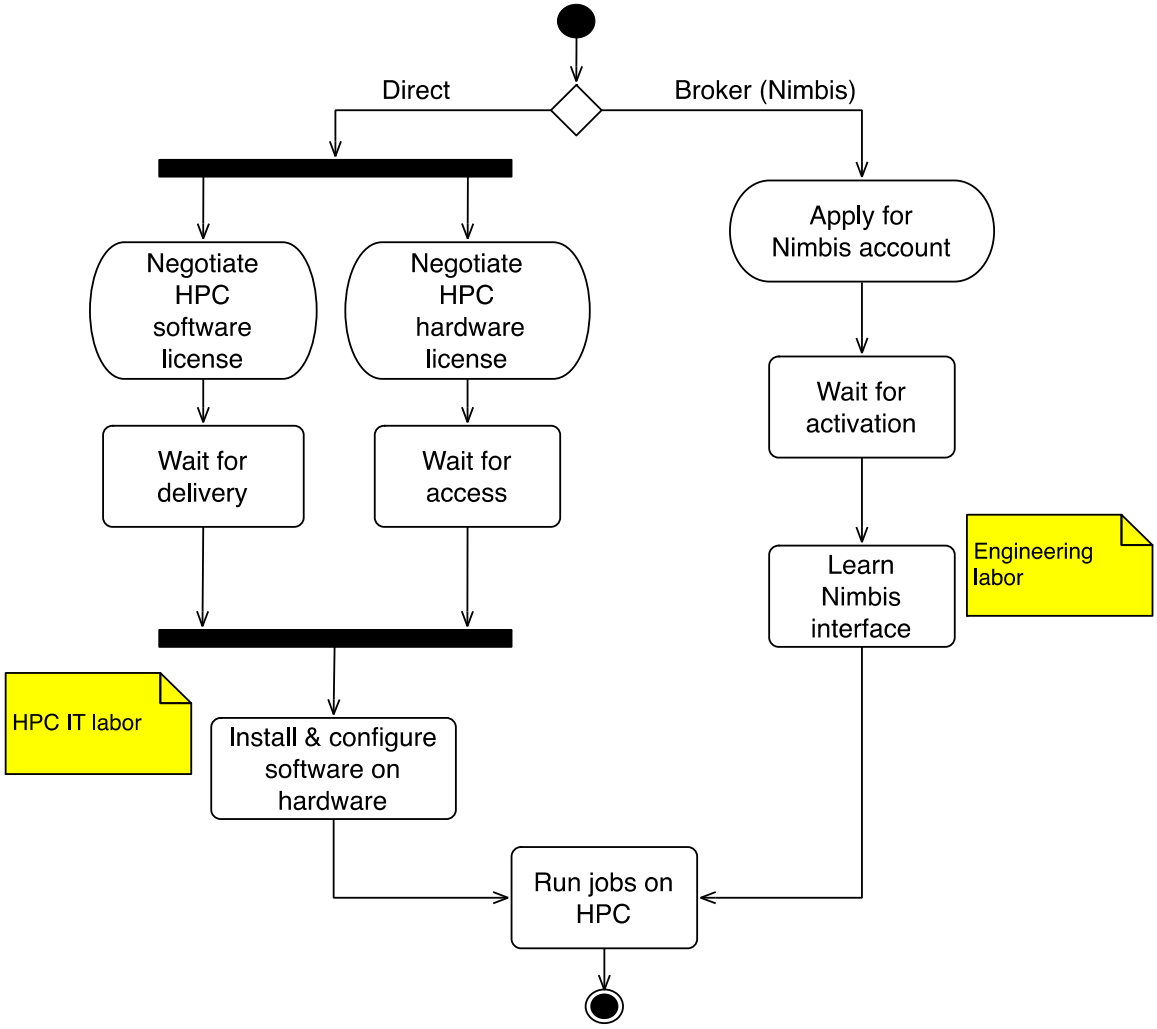

Figure 6 Startup workflow: direct versus broker

Our initial assumption at the onset of this work was that a broker could substantially reduce the startup time, because the time to acquire and configure the hardware and software pieces had already been done by the brokerage service. We hypothesized that the savings would be on the order of 80%.

4 Results and Discussion

4.1 Capability demonstrations

The capability demonstrations were recorded by using Snapz Pro [Snapz] software to do a "screencast" that shows a user submitting jobs for each of the three demonstrations, as well as downloading the results. In all three cases, the capability demonstration was successful.

4.2 ROI metrics analysis

Table 1 shows a summary of the rates that we used to estimate labor costs. In all cases, the salary was obtained from Indeed[2], a job search website that provides salary estimates. The position listed in the table was used as the search term for Indeed, and we used national salary trend for the estimate. For example, to obtain the salary estimate for an electrical engineer, we searched for "electrical engineer", and did not specify a particular location.

For the ANSYS use case, we assumed that a senior mechanical engineer would be the typical user, because it takes years of experience to become productive in a finite element analysis (FEA) tool such as ANSYS. In contrast, for Blender and Mathematica we assumed "mechanical engineer", because the target audience here is users at manufacturing companies who are just learning how to use these tools.

Table 1 Rates used for estimating labor costs

Position	Salary	Hourly
HPC IT consultant (Software installation on HPC)	$80,000	$80
Electrical engineer (Ghost user)	$87,000	$87
Senior mechanical engineer (ANSYS user)	$94,000	$94
Mechanical engineer (Blender/Mathematica user)	$83,000	$83

To calculate hourly costs from salary, we estimated the fully burdened cost to the employer was twice the salary, and then assumed a 40-hour work week, with fifty work-weeks per year. This allows us to estimate an hourly cost rate using the following equation:

$$Hourly = \frac{Salary \times 2}{(40\,hr/wk) \times (50\,wks/yr)} = \frac{Salary}{1000}$$

4.2.1 AltaSim Ghost on OSC

Benefits. From the previous study (HPC-ISP-PILOTS), we estimated the benefits of using HPC at $488,000 cost savings / PNCC unit, with an additional benefit of $233,000 for reducing the probability of design escapes, for a total estimated benefit of **$721,000.** This analysis assumed 128 design iterations were involved.

Simulation time. The relevant simulation was a Xyce simulation of EMI in a 4U PNCC circuit across a frequency range of 30 MHz. The simulation was run on 4 nodes, with 4 cores per node, and ran in two hours and thirty-three minutes.

Nimbis rates. The initial Nimbis pricing for Ghost is to charge $1.40 / node-hour of usage for access to the OSC hardware, and $0.40 / node-hour of usage for access to the Ghost software.

[2] http://www.indeed.com

Usage costs. For our usage scenario, we assume simulation usage similar to HPC-ISP-PILOTS analysis: 4 nodes used per simulation, 3 hours per simulation, 128 simulations. Based on these numbers, Table 2 shows the estimated simulation cost for production usage.

Table 2 Ghost simulation costs

Hardware compute costs	$2,150.40
Software compute costs	$614.40
Total simulation cost	**$2,764.80**

Startup cost. The only startup cost associated with using the Ghost interface is the labor of the electrical engineer to learn how to use the interface for the first time. The author ran a usability study with two electrical engineers at ISI, observing them as they performed tasks with the Nimbis interface. The time to perform all tasks was about half an hour. Therefore, we assume here a startup cost of 1 hour of labor from an electrical engineer. From Table 1, we estimate this as **$87.**

Table 3 ROI calculations for Ghost

Benefits	
Physical cost savings/PNCC unit	$488,000
Reduction in design escapes	$233,000
Total benefits	$**721,000**
Costs	
Startup cost (1 hour EE labor)	$87
Cost for 128 simulations	$2,765
Total costs	$2,852
ROI	
Physical cost savings/PNCC unit	170X
Reduction in design escapes	81X
Total ROI	**252X**

ROI calculations. Table 3 shows the ROI calculations for the Ghost usage, broken down by the different categories of benefits identified in HPC-ISP-PILOTS [Schot11], with a final estimated ROI of **252X,** which is very high.

Brokerage savings. Table 4 shows estimates for the startup costs and time associated with contracting directly with the hardware and software provider, as shown in Figure 6. These estimates were provided by Nimbis Services based on their experiences in acquiring access to the resources. The OSC hardware acquisition time required one month of business time and one week of technical time. Note the significant amount of time to acquire the Ghost software. This large period of time is due to the time required to negotiate the related license agreements to obtain access to the software. For traditional software, such latencies are not typical. However, in the case of Ghost, the original software was developed by Sandia National Labs, and was sub-licensed to AltaSim Technologies. This situation substantially increases the time to acquire the licenses. Nimbis provided an estimate of six to twelve months to negotiate an agreement with Sandia; we chose the median value of 9 months (180 business days) for our calculations. Table 5 shows estimates when interacting through a brokerage service. The resulting startup savings for using a brokerage service are estimated at $12,713 (a savings of 99%), and reduces startup time by about ten months.

Table 4 Direct startup costs for Ghost

OSC hardware acquisition	25 business days
Ghost software acquisition	180 business days
Install & configure software	20 business days IT labor: $12,800
Time to run first job	**200 business days (10 months)**

Table 5 Brokerage startup costs for Ghost

Nimbis account activation	1 business day
Learn Nimbis interface	1 hour EE labor: $87
Time to run first job	**1 business day**

4.2.2 ANSYS on R Systems

Benefits. From the previous study [Schott11], we estimated the benefits of using HPC at $275,000 labor savings per engineer per year, with an additional benefit of $815,000 for reducing the probability of design escapes, for a total estimated benefit of **$1,090,000**. This analysis assumed 8 design iterations.

Simulation time. The relevant simulation was a thermal transient analysis, followed by 81 static structural analyses, for a total simulation involving 486 MDOF (million degrees of freedom). This required a simulation time of seven hours and fifty six minutes. The thermal analysis ran on 14 nodes, and the structural analyses ran on only 10 nodes (due to licensing restrictions). The original cluster had four cores per node, which meant that the simulation used 56 cores for the thermal analysis, and 40 cores for the structural analysis.

Nimbis rates. The ANSYS pricing offered by Nimbis is more complex than their other pricing models. For hardware costs, there is a one-time setup fee, described in the next section. The login node costs $2,500 per month, and the compute nodes cost $1,317 per 16 core-week (i.e., it costs $1,317 to rent 16 cores for one week). For software costs, an ANSYS mechanical license seat costs $1,083 per month, and an ANSYS Mechanical HPC license seat costs $100 per month. Running an ANSYS simulation in HPC mode requires one ANSYS Mechanical license seat, and one additional HPC license seat for every core (except for the first two). For example, to run on 16 cores requires one ANSYS Mechanical license seat, and 14 ANSYS HPC license seats.

Usage costs. For our ANSYS usage scenario, we assume a similar setup to the original simulations done under HPC-ISP-PILOTS. We assume the login node will be rented for 3 months, and that 8 compute weeks of time will be rented, to correspond to 8 one-week iterations. We assume the customer will rent 32 cores, therefore requiring 2 16-core machines. Based on these numbers, Table 6 shows the estimated simulation cost for production usage, with a total usage cost of $65,072.

Table 6 ANSYS simulation costs

Hardware: 3 months login, 8 weeks of 32-core compute	$23,572
Software: 3 months of 10 ANSYS Mechanical licenses + 30 Mechanical HPC licenses	$41,500
Total	**$65,072**

Startup costs. Nimbis charges a one-time setup fee of $4,000. In addition, the ANSYS web interface is more complex than other interfaces: it provides access to VNC, as well as a web portal interface. We

were not able to directly measure the startup time for this interface. However, based on our interactions with engineers at Woodward who are the intended users, we estimate an upper limit of about 3 hours to learn how to use the interface once the user has access to the web portal. Based on Table 1, three hours of senior mechanical engineer labor would cost $282.

ROI calculations. Table 7 shows the ROI calculations for ANSYS usage, with a final estimated ROI of **14.7X.**

Table 7 ROI calculations for ANSYS

Benefits	
Labor savings per engineer per year	$275,000
Reduction in design escapes	$815,000
Total benefits	$1,090,000
Costs	
Startup costs (one time fee + 3 hrs of senior mech eng labor)	$4,282
Cost for simulations (3 months login, 8 weeks of 32-core compute, licenses)	$65,072
Total costs	$69,354
ROI	
Labor savings per engineer per year	3.0X
Reduction in design escapes	10.8X
Total ROI	**14.7X**

Brokerage savings. Table 8 shows estimates for the startup costs and time associated with negotiating directly with the provider. We obtained estimates of the R Systems hardware acquisition for Nimbis Services, who has experience acquiring their hardware. For the ANSYS software acquisition, we consulted with engineers at Woodward who have experience acquiring ANSYS software licenses. For installing and configuring software, we were able to estimate this since we were directly involved in configuring the software to run on R Systems. Table 9 shows estimates when interacting through a brokerage service. The setup process for using Nimbis is a little more complex in this scenario because of the additional ITAR restrictions: VPN software must be configured to access the system, and account activation takes about a week instead of a day. The resulting startup savings for using a brokerage service are estimated at $2,918 (a savings of 91%), and reduces startup time by about seven weeks.

Table 8 Direct startup costs for ANSYS

R Systems hardware acquisition	35 business days
ANSYS software acquisition	5 business days
Install & configure software	5 business days IT labor: $3,200
Time to run first job	**40 business days (2 months)**

Table 9 Brokerage startup costs for ANSYS

Nimbis account activation	5 business days
Install VPN software & learn Nimbis interface	3 hours Senior mech eng labor: $282
Time to run first job	**6 business days**

4.2.3 Mathematica and Blender on Amazon EC2

Benefits. From the previous study [Schott11], we estimated the benefits of using HPC at $12,420/yr by reducing the amount of physical prototyping required. This analysis assumes four design loops.

Simulation time. Because the previous HPC-ISP-PILOTS study did not result in a particular simulation job, we did not have hard numbers to use as the basis of our simulation time. Instead, we chose computational simulation numbers that we felt were plausible. We assumed that each design loop would involve eight hours of Mathematica simulation, running on four extra-large Amazon instances (four cores per instance), resulting in sixteen application kernels per simulation. Similarly, we assumed that each design loop would require eight hours of Blender rendering running on four extra-large Amazon instances.

Nimbis rates. For both Mathematica and Blender jobs, Nimbis charges $0.89/node-hour for an extra-large instance, as well as an additional fee of $3/job. In addition, running Mathematica jobs costs $0.20/kernel-hour, where running Blender does not incur any additional software costs.

Usage costs. Table 10 shows the estimated simulation costs for the usage scenario described above, with a total usage cost of $354.24.

Table 10 Mathematica & Blender simulation costs

Mathematica	
Hardware: $/node-hour	$0.89
Software: $/kernel-hr	$0.20
Nimbis fee (per job/session)	$3
Hardware & software compute cost (4 iterations)	$228.32
Blender	
Hardware: $/node-hour	$0.89
Software: $/node-hour	Free
Nimbis fee (per job/session)	$3
Hardware & software compute costs (4 iterations)	$125.92
Total compute cost (4 iterations)	**$354.24**

Startup cost. The only startup cost associated with using the Mathematica and Blender interfaces is the labor of the mechanical engineer to learn how to use the interface for the first time. We collected this information from a user at WESRAC who was interacting with the system. It took about half an hour for the user to figure out how to use the interface, so for labor estimation purposes, we round up and assume an hour. From Table 1, we estimate this as **$83.**

Table 11 ROI calculations for Mathematica and Blender

Benefits	
Less physical prototyping / yr	$12,420
Costs	
Startup cost (1 hr of mech eng labor)	$83
Compute cost	$354
Total costs	$347
ROI	
Total	**27.4X**

ROI calculations. Table 11 shows the ROI calculations for Mathematica and Blender usage, with a final estimated ROI of **27.4X**.

Brokerage savings. Table 12 shows estimates for the startup costs and time associated with contracting directly with the hardware and software provider. Obtaining access to Amazon EC2 can be done in the span of a single business day, probably on the order of about an hour. The Blender software is open source and easily downloadable from the web, similarly in about the order of an hour. The long pole in the

tent is obtaining access to the Mathematica software, since it is commercially licensed. In particular, the parallel version of Mathematica cannot be purchased directly from Wolfram's website the way the desktop version can. Instead, the end-user must negotiate directly with a Wolfram sales associate. Nimbis Services estimated about a month (20 business days) is typically required to obtain a personalized quote for a license. They also estimated that it took about 3 days of labor to install and configure Blender and Mathematica in Amazon EC 2 instances. Table 13 shows estimates for the startup costs and time associated with using the Nimbis brokerage service. The resulting startup savings for using a brokerage service are estimated at $1,837 (a savings of 96%) and reduces startup time by about a month.

Table 12 Direct startup costs for Mathematica and Blender

Amazon EC2 hardware acquisition	<1 business day
Blender software acquisition	<1 business day
Mathematica software acquisition	20 business days
Install & configure Blender & Mathematica software	3 business days IT labor: $1,920
Time to run first job	**23 business days (~1 month)**

Table 13 Brokerage startup costs for Mathematica and Blender

Nimbis account	1 business day
Learn Nimbis interface	1 hour Mech eng: $83
Time to run first job	**1 business day**

4.3 Qualitative usability analysis

In the course of this study, the author had the opportunity to observe and work with the Nimbis user interface, both directly and through interactions with potential users. Through these interactions, we identified a number of usability issues with the current Nimbis user interface that caused some users to struggle with the system. Here we described some of them.

Shopping cart metaphor. Nimbis uses a "shopping cart" metaphor for allowing users to purchase compute resources the way one might buy a book on Amazon. Some users had trouble forming a mental model of the behavior of the Nimbis web portal: they were confused by the shopping cart metaphor, and the need to "check out" their jobs.

Instructional flowchart. When logging in to the portal for the first time, several users misinterpreted the block diagram (see Figure 7) as a set of buttons to click. When they clicked it, they were confronted with a more complex flowchart that was not clickable (see Figure 8).

Feedback about job progress. For some of the jobs, there was insufficient feedback to the user about the progress of the job. For example, when running a Blender job, the user was used to having some progress information about the results to date, and that information was not provided in the Nimbis Services web portal implementation, although at the time the study was done, there were plans to add more feedback.

More detailed information about failures. Currently, when a job fails, there is limited information available to the user about why the job failed. This makes it difficult for the user to determine what the problem is and what should be changed before re-submitting the job.

Resources to devote to job. Under the Nimbis model, the user must identify in advance how many computational resources they need, in terms of the size of the nodes and the amount of time required. However, the users often had no idea how many resources to devote to a job, and what time limit to use.

Charges in case of failure. The users were not sure if their money would be refunded in the event that their job failed. In fact, they would be refunded in this event, but this information was not clearly displayed to them. In particular, if the job was terminated, it was not clear to the user what the charges would be.

Nimbis points exchange rate. Nimbis allows the user to purchase credits called *Nimbis points* for use in running simulations. Each Nimbis point is worth one penny, but this exchange rate was not obvious to the users. "The users' resources were denominated in Nimbis points, but the job costs were denominated in dollars, and so the users were not sure how many of their Nimbis points resources would be expended when running a job.

Application kernels. The user must specify how many hardware nodes (called "parallel instances", and how many licenses (called "application kernels") when running jobs such as Ghost or Mathematica (see Figure 9). Some users found this nomenclature confusing. They did not understand what was meant by "application kernels", and did not realize that it was associated with the number of cores being run.

Charges for unused time. Some users were not sure if they were charged for time they had requested but had not used. For example, if they requested four hours and ran their simulation in one hour, they did not know if they would be charged for one hour or four hours. One expected to get some kind of confirmation about a rebate for time that was not used.

Interface elements present but not prominent. In some cases, there were user interface elements that the users failed to notice because they were not displayed prominently. For example, some users had a hard time finding the different HPC software packages available through Nimbis under the "Catalog" heading at the top-left. In another case, after the user uploaded a file, the user did not see the file upload had been successful, because the feedback about the successful file upload was too subtle.

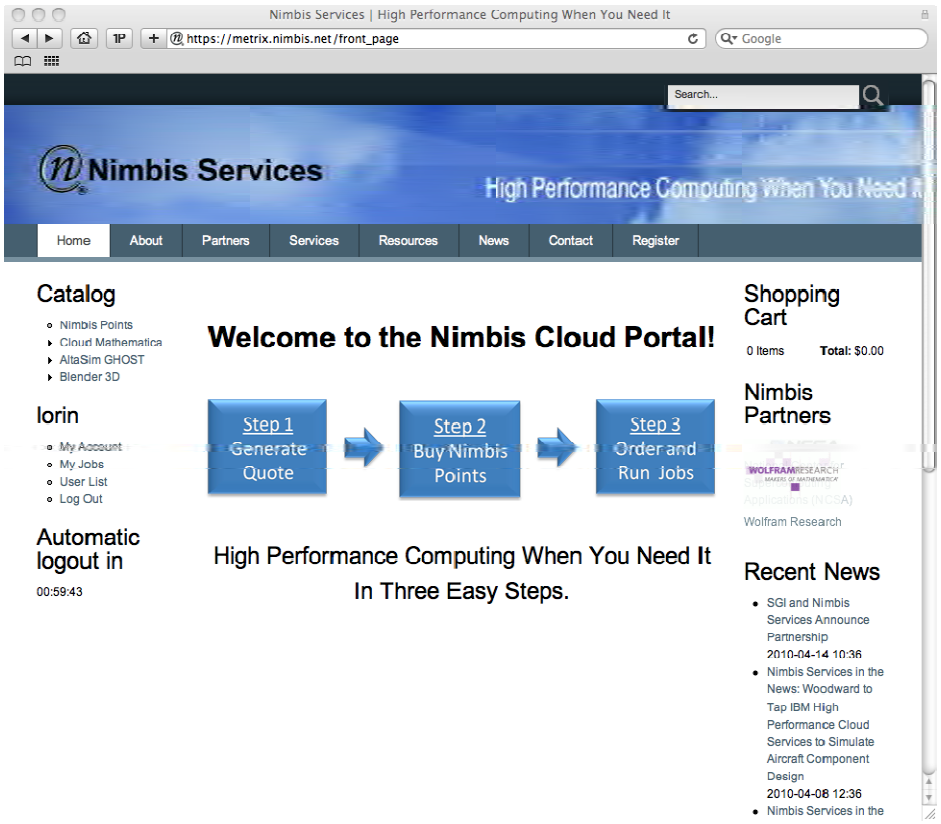

Figure 7 Initial page after login

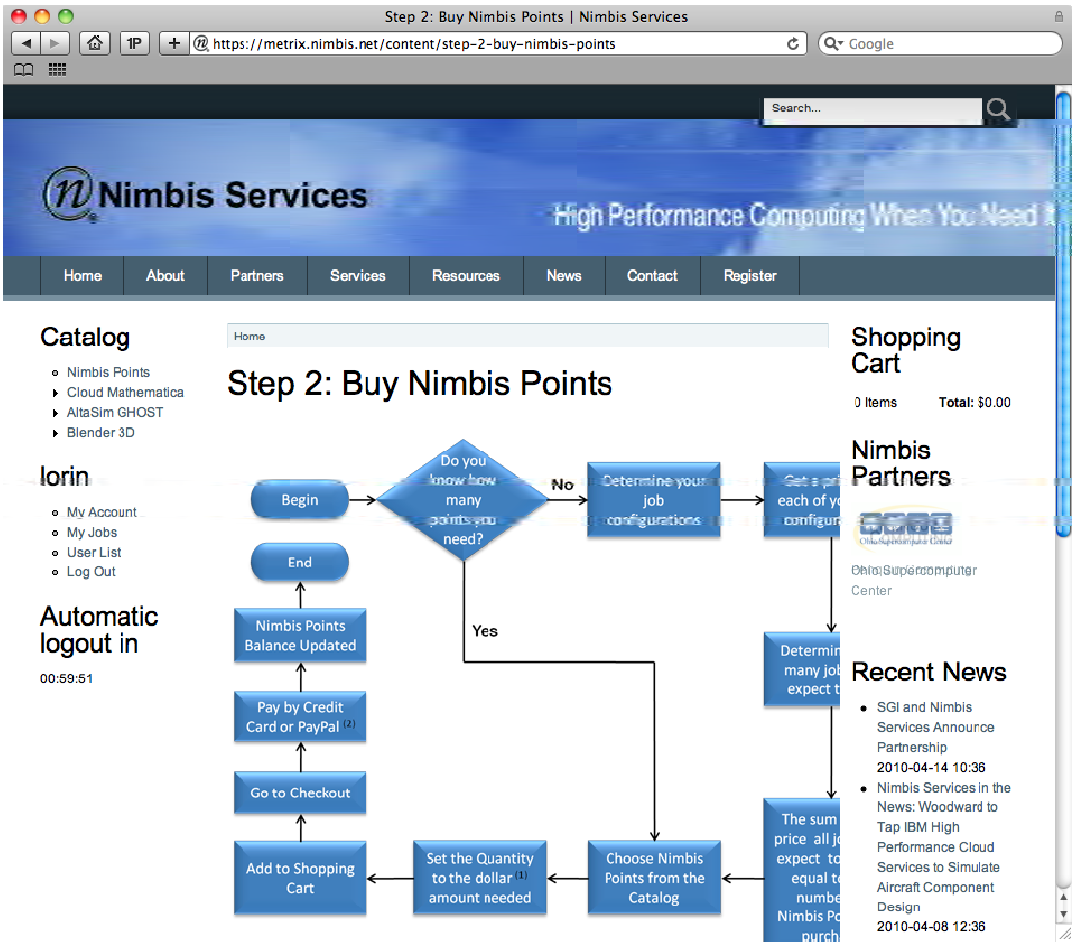

Figure 8 Flowchart displayed after clicking block diagram

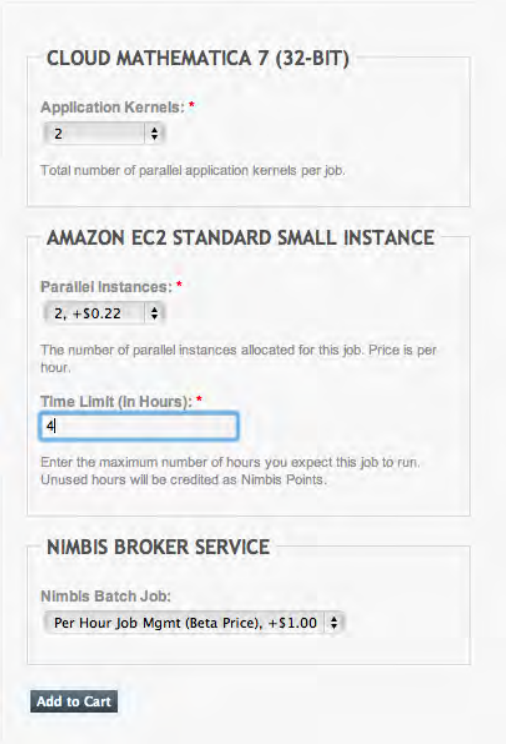

Figure 9 Choosing resources for Mathematica jobs

5 Conclusion

The results of the study complete the picture initially sketched by the HPC-ISP-PILOTS study and suggest that there is significant potential for small and medium-sized companies in the DoD supply chain to leverage remote HPC resources to do modeling and simulation, in order to reduce physical prototyping and reduce the likelihood of design escapes. The metrics analysis performed here suggests that the potential return on investment is very high, in the range of 15X-250X.

The startup costs metrics suggests that there is a significant advantage of using a brokerage service in order to get up and running as quickly as possible. We estimated startup cost savings on the order of 90%-99%, where we had hypothesized savings on the order of 80%. More significant than the cost savings is really the time savings, which was reduced from ten months to one month. One of the major use cases of a cloud computing approach to modeling and simulation for HPC is to use it when engineers need a quick turnaround time for a large simulation. A large startup time penalty presents a significant adoption barrier, and the reduction of this barrier through a brokerage approach should reduce the barrier and increase the adoption rate. Even for data that is subject to ITAR processing restrictions, this study has shown that there are remote HPC providers that can provide ITAR compliant environments, such as R Systems, that can be made accessible through a brokerage service.

While these results suggest that the brokerage approach holds promise, there are other social and technical factors that will slow adoption. Networking issues will continue to cause problems. One simple issue is bandwidth: if large files need to be transferred back and forth, the performance will depend on the bandwidth available to the end-user. For small and medium-sized companies, they may have poorer network connectivity, which will reduce turnaround time. Another network issue is network policy, security and firewalls. Firewalls may block access to the VNC client, which plays a significant role in providing access to ANSYS. We saw both of these issues when observing engineers at Woodward try to use ANSYS on R Systems through Nimbis in a production environment.

Finally, there are user interface issues. End users are very unforgiving of web applications with poor user interfaces, because they have very little invested in them up-front. It is therefore critical to ensure that the interface is properly usable. Proving access to engineering software over web interfaces is a new type of model that users are not yet familiar with, which requires a learning curve for both the interface designer and the user. Interfaces for services like Nimbis will improve over time through user feedback, but the speed at which these interfaces are able to adopt to the needs of the user will likely be critical in determining overall adoption.

6 Real-time Processing via Natively Probabilistic Computation

6.1 Introduction

Computers were designed to solve problems of logical deduction and arithmetic, but are increasingly being used across the DoD and in industry to interpret large streams of noisy, ambiguous data and make good guesses under extreme uncertainty. They can manage databases with billions of entries, but fail at probabilistic problems with just hundreds of unknown quantities. This severely limits the performance of DoD computing systems, especially in estimation and optimization problems key for ISR and logistics.

This project aimed to close this gap, by evaluating Navia's, natively probabilistic computer architectures on key problems of DoD interest. The main goal of the project was to provide quantitative evidence showing that state-of-the-art probabilistic inference based solutions to problems of video processing can be delivered at 100-1000x greater throughput and 10x less power than solutions on commodity digital hardware, making it possible to apply otherwise impractical but highly accurate methods in real time. The project empirically verified Navia's theoretical estimates of expected performance benefits on two real-time video processing and inference problems of DoD interest:

- Motion detection
- Clustering and classification of streaming data

The project leveraged Navia's pre-existing, probabilistic computing technology, including probabilistic circuit primitives (exclusively licensed from MIT), along with Navia's probabilistic circuit compiler. It also used commodity off-the-shelf reconfigurable logic chips (Field Programmable Gate Arrays FPGAs) as the substrate on which Navia's probabilistic circuitry was implemented. Navia has already used these to deliver 1000x speed improvements on problems from computational statistical mechanics, despite their 10-50x disadvantages versus ASICs.

The key features of Navia's hardware prototypes enabling these improvements are the use of extremely fine grained (i.e. variable, pixel and bit-level) parallelism and very low bit precision to pack a very large number of stochastic computing devices on a single chip.

6.2 Technical approach

Structured stochastic processes play a central role in the design of approximation algorithms for probabilistic inference and nonlinear optimization. Markov chain [Metropolis53, Geman87] and sequential [Doucet01] Monte Carlo methods are classic examples, addressing probabilistic inference problems arising in application areas such as image and speech processing, object tracking, supervised and unsupervised learning, and protein folding, among many others. However, these widely used algorithms, and approximate Bayesian inference in general, can seem unacceptably inefficient when simulated on current general-purpose computers. This high apparent cost should not be surprising. Computers are based on deterministic Boolean circuits that simulate propositional deduction according to the Boolean algebra, while problems of inference under uncertainty (and many stochastic algorithms for solving these problems) are best described in terms of the probability algebra. To perform probabilistic inference on computers based on all-or-none, deterministic logic circuits, one typically rewrites algorithms in terms of generic real-number arithmetic, which is then approximated by general-purpose Boolean circuits for floating-point arithmetic. This indirection has many disadvantages: it obscures fine-grained parallelism, complicates algorithm analysis, and is needlessly costly in both time and space.

Navia has developed an alternative approach that directly models probability in digital hardware. They base this approach on a novel abstraction called combinational stochastic logic, which stands in relation to the probability algebra as Boolean gates do to the Boolean algebra. In earlier work, they have made three contributions. First, Navia has shown how combinational stochastic logic circuits generalize Boolean logic, allowing construction of arbitrary propositional probabilistic models. Second, they have combined our stochastic logic gates with ideas from contemporary digital design, showing how to build stochastic finite state machines that implement useful sampling algorithms. In particular, they have directly implemented widely-used Markov chain Monte Carlo (MCMC) algorithms for arbitrary discrete Markov random fields (MRF) [2] in hardware in a massively parallel fashion. Although they have focused on MRF models, widely used in image processing, our basic approach is based on general primitives directly usable for many other applications. Finally, they have estimated the performance of our approach when implemented on commodity FPGAs, finding substantial improvements in time efficiency, space efficiency and price. They have also shown that stochastic logic circuits can perform robustly in the presence of a range of transient and persistent faults, suggesting interesting possibilities for distributed computing on unreliable substrates.

The central, novel abstraction behind Navia's probabilistic circuits is called *combinational stochastic logic*. It generalizes combinational (i.e., stateless) Boolean circuits to the stochastic setting, recovering Boolean gates and composition laws in the deterministic limit. Figure 10 compares Navia's abstraction to traditional digital logic and shows an example natively probabilistic gate: (a) shows the combinational Boolean logic abstraction, and one example: the AND gate and its associated table; (b) shows the *computational stochastic logic* abstraction. On each work cycle, samples are drawn on OUT from P(OUT|IN), consuming *h* random bits on RAND to generate nondeterminism; (c) shows how an AND gate can be viewed as a combinational stochastic logic gate that happens to be deterministic; (d) shows the conditional probability table and schematic for a Θ gate, which flips a coin whose weight was specified on IN as a binary number (e.g. for IN=01111, P(OUT=1|IN)=7/16). Θ gates can be implemented by a comparator that outputs 1 if RAND ≤ IN.

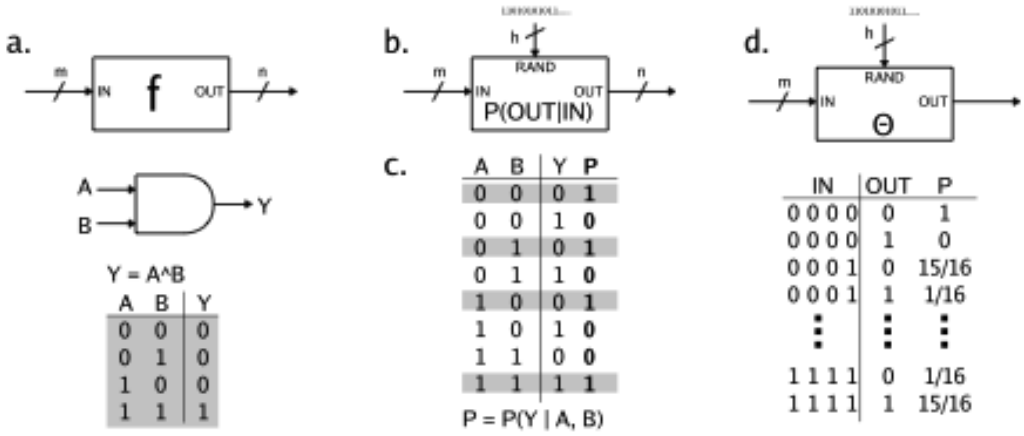

Figure 10 Combinational stochastic logic

Feeding the output of one gate into the input of another results in samples from the joint distribution of the two elements, allowing construction of samplers for complex distributions from simpler pieces. Furthermore, one can abstract away the details of a complex stochastic circuit, viewing it as a single combinational stochastic gate that simply generates samples from the marginal distribution of the original output gate's value given the original input gate's input value. Taken together, these laws support the construction of arbitrarily complex probabilistic (and Boolean) systems out of reusable components. State machines built using this logic naturally carry out Markov chain Monte Carlo algorithms, and can often be implemented in a massively parallel fashion, leveraging conditional independencies [Mansinghka08].

Figure 11 shows an example circuit, a synthesized circuit for a parallel 16-bit Gibbs sampler. The middle and right plots show price/performance measurements for tilings of this design, suitable for large binary

lattice problems, comparing Spartan 3 family FPGAs (with XOR-SHIFT PRNGs) to estimates for a software sampler. The comparison is intended to be as meaningful as possible, comparing off-the-shelf price for both commodity x86 hardware and FPGAs. Note that MCMC runs for image processing typically use ~100,000 samples, achieved in real-time (at competitive marginal cost).

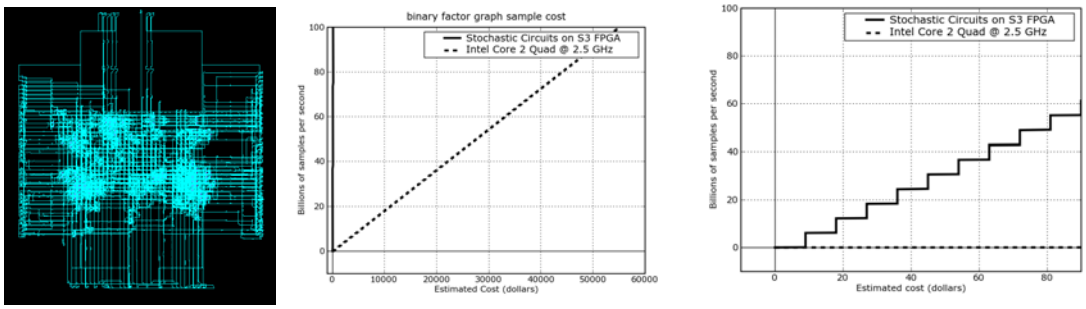

Figure 11 Synthesized stochastic circuit for massively parallel MCMC in an Ising Lattice

Figure 12 shows architectures for stochastic state machines. Unlike deterministic, digital FSMs, which carefully execute a precise sequence of steps, stochastic FSMs iterate Markov chains to convergence. (right) A distributed stochastic circuit for Gibbs sampling in a lattice probability model, reflecting the node-level parallelism in a factor graph, where all nodes of the same color in a given coloring can be sampled simultaneously.

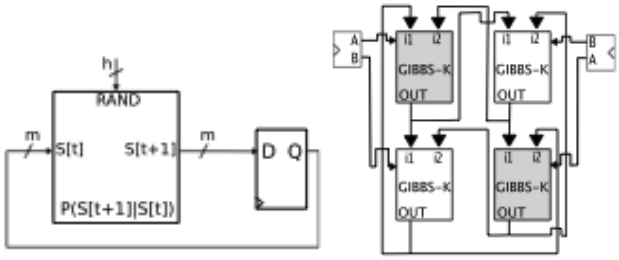

Figure 12 Stochastic digital state machines

Stochastic digital circuits can leverage the low bit precision requirements of sampling algorithms to save space and to be robust to bit-error rates as high as 1 fault in every 100 bit transitions in realistic settings [Mansinghka08]. Ultimately, Navia's design wins arise from the freedom the primitives give them to exploit the low bit precision requirements of sampling algorithms combined with their massive, fine-grained (but non-embarrassing) per-iteration parallelism. In fact, a wide range of sampling algorithms, including both other Markov chain based methods and sequential Monte Carlo methods for problems like target tracking can be built using the same primitives [Doucet01, Mansinghka08].

Navia has also prototyped a compiler that produces finite-precision parallel automata for sampling-based inference in factor graphs, substantially reducing development time (leaving tabular factors to be straightforwardly implemented by hand). The compiler operates by first coloring the factor graph to identify opportunities for parallelism and then constructing a graphical representation of the abstract stochastic automaton that is the desired Gibbs sampler.

The apparent intractability of inference has hindered the use of Bayesian methods in the design of intelligent systems in very large scale signal processing and statistical computing applications, and in the explanation of computation in the mind and brain. We hope stochastic logic circuits help to address this concern, by providing new tools for mapping probabilistic inference onto existing digital computing machinery, and suggesting a new class of natively stochastic digital computing machines.

Current state-of-the-art approaches to performing inference in probabilistic graphical models (and stochastic simulation more generally) rely on laborious design of optimized software systems implementing either highly approximate deterministic calculations or very slow stochastic simulators. Both of these approaches currently involve a large amount of per-problem mathematical work as well as custom high performance numerical software development, and typically yield systems that substantially exceed the accuracy of non-probabilistic approaches (e.g. in video processing problems like stereo reconstruction, dense optical flow, and clustering) but which are far too inefficient to be practical. GPU acceleration has been attempted for some of these methods, but typically only yields a single order of magnitude speedup, and cannot be fruitfully applied to the stochastic simulation methods for video because of the inability to do accurate branch prediction for stochastic methods.

In contrast, Navia's approach relies on general compilation technology to turn a declarative description of the problem being solved (represented as a probabilistic graphical model) into a massively parallel, low precision, fault tolerant probabilistic digital circuit for solving it. Navia has previously observed 1000x improvements in latency and throughput over software simulation and 10x-100x improvements in power envelope, making high resolution offline and low resolution embedded processing feasible for the first time. These improvements seem to be sufficient to make probabilistic techniques for video processing practical in real DoD settings for the first time.

6.3 Key findings

In this research effort, Navia demonstrated the application of this approach on two computational challenge problems, using their implementation of a software-reprogrammable probabilistic video processor (Figure 13). The key findings from this research effort were:

- Documented 1000x+ speedups over software using a software-reprogrammable probabilistic video processor
- Documented video processing in real time, streaming data for real-time time series clustering and classification, with no loss of accuracy.
- 10-30x less power usage than a CPU.
- 300-1000x faster than software MCMC.
- 30-100x faster than restricted domain software solvers (e.g. graph cuts), using general purpose stochastic hardware.

Figure 13 Software-reprogrammable probabilistic video processor

6.4 Problem 1: Real-Time Motion Detection

The first step in many EO stream processing pipelines is *motion extraction*: identifying the motion vector for every incoming pixel that maps it to its best match in the next frame. This underpins segmentation, tracking, and compression, as well as much of modern robotics, and requires resolving an ill-posed problem. Best-in-class solutions use probabilistic inference to resolve the ambiguities in the frame-by-frame match, but take minutes or worse per frame on expensive server hardware. Navia's probabilistic video processor can solve this problem in real time in an embedded form factor, by doing massively parallel inference using stochastic digital circuits and architectures (Figure 14).

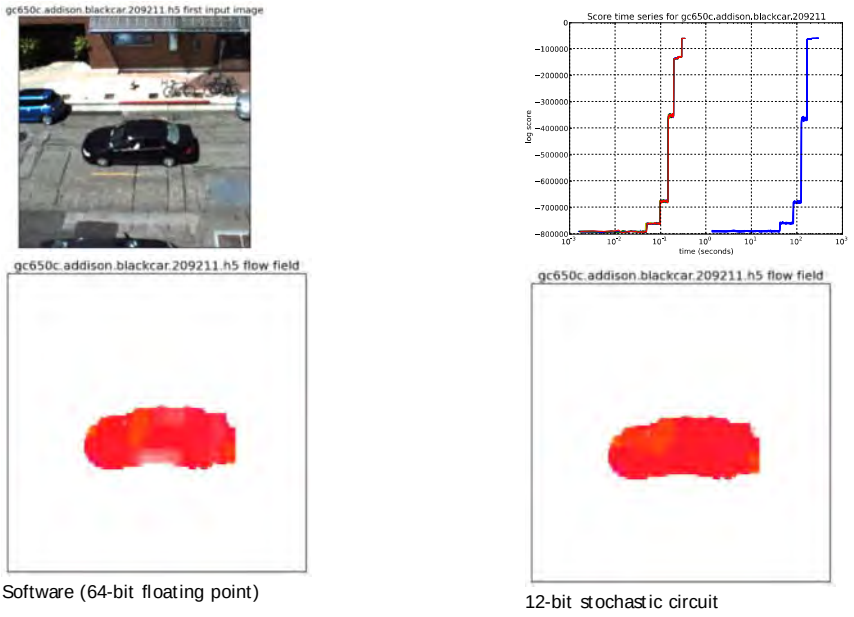

Figure 14 Real-time optical flow, comparing 64-bit floating point software to a 12-bit stochastic circuit. The red line represents a stochastic circuit, the blue line represents a CPU.

6.5 Problem 2: Real-Time Clustering and Classification of Streaming Data

A second major bottleneck in many EO stream processing pipelines is in pattern recognition: constructing a model of the streaming data that can be used to identify clusters and predict missing values, such as image categories. Navia has documented the performance of a massively parallel, stochastic circuit based accelerator for a Dirichlet process mixture model on a dataset of handwritten images, showing comparable speed and power benefits to video processing with no appreciable loss in accuracy over software simulation. It automatically discovers the number of clusters that best explain the data using fully Bayesian inference, allowing that number to change as more streaming data arrives, and enables streaming classification (Figure 12).

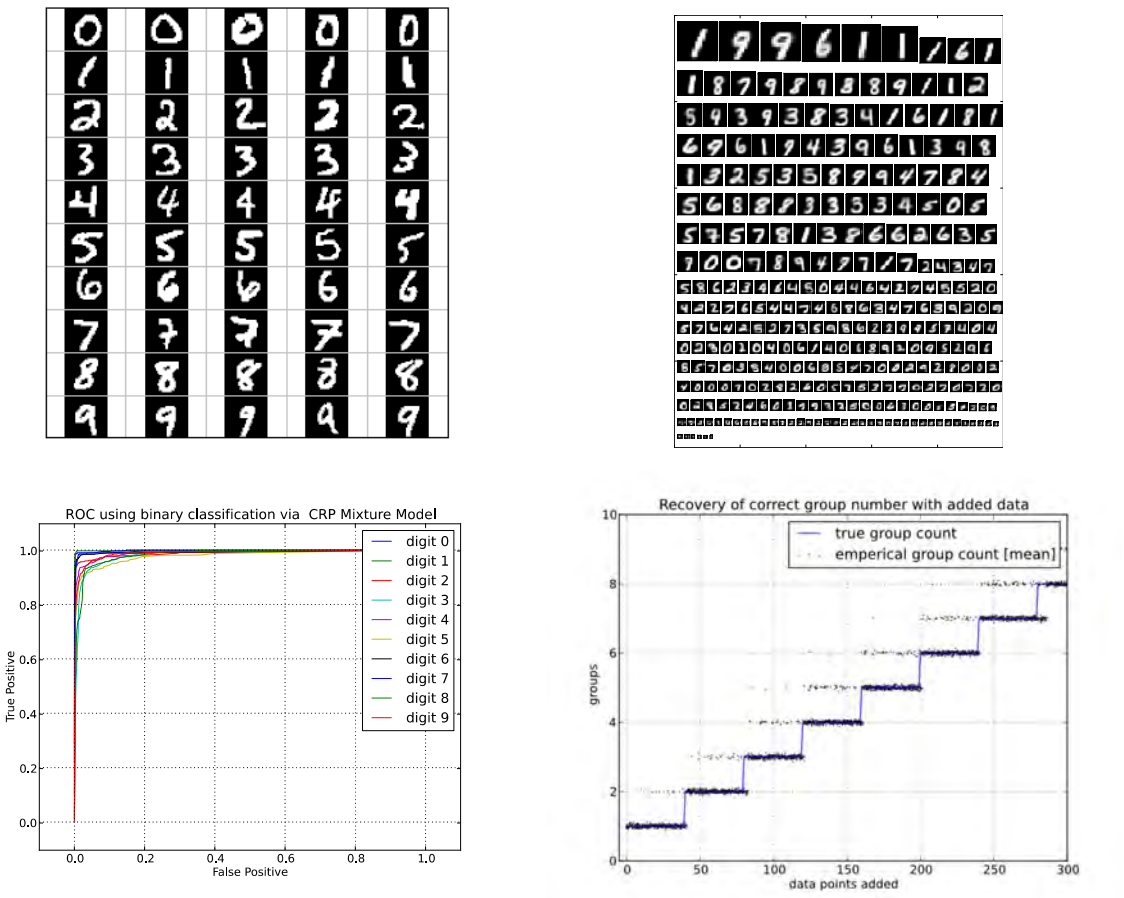

Figure 15 Clustering and classification of streaming data. Example input images (top left). All digit prototypes (cluster centers) found, with size proportional to frequency (top right). Classification accuracy for real-time circuit (bottom left). Tracking the number of clusters as the distribution changes on-line (bottom right).

7 References

[Ambrust09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica and Matei Zaharia, *Above the Clouds: A Berkeley View of Cloud Computing*, Technical Report No. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb. 10, 2009.

[ANSYS] ANSYS, Inc. http://www.ansys.com/

[Azure] Windows Azure http://www.microsoft.com/windowsazure/

[Cyc] SGI Cyclone http://www.sgi.com/products/hpc_cloud/cyclone/

[EC2] Amazon Elastic Compute Cloud (EC2). 2011. 7 July 2011 <http://aws.amazon.com/ec2/>.

[Dean04] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Proceedings fo the Sixth Symposium on Operating System Design and Implementation (OSDI'04),San Francisco, CA, December, 2004.

[Doucet01] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo in Practice*, 2001.

[GAE] Google App Engine. 2011. 7 July 2011 <http://code.google.com/appengine/>.

[Geman87] S. Geman and D. Geman. *Stochastic Relaxation, Gibbs distributions and the Bayesian restoration of images.* Readings in Computer Vision, pp. 564-584, 1987.

[Hutchinson01] Scott A Huctchinson, Eric R. Keiter, Robert J. Hoekstra, Herman A. Watts, Arlon J. Waters, Regina L. Schells and Steven D. Wix, "The Xyce Parallel Electronic Simulator – An Overview", Proceedings of the IEEE International Symposium on Circuits and Systems, May 6—9, 2001.

[Feilner06] Markus Feilner, *OpenVPN: Building and Integrating Virtual Private Networks*, O'Reilly, 2006.

[Mansinghka08] V. Mansinghka, E. Jonas and J. Tenenbaum. *Stochastic Digital Circuits for Probabilistic Inference*. MIT CSAIL Technical Report 2008-069.

[Metropolis53] N. Metropolis, A. W. Rosenbluth M.N. Rosenbluth, A.H. Teller and E. Teller. *Equations of State Calculation by Fast Computing Machines*. Journal of Chemical Physics, 1953.

[PoD] Penguin Computing On Demand. 2011. 7 July 2011. <http://www.penguincomputing.com/POD>.

[RSys] R Systems NA. 2011. 7 July 2011 <http://rsystemsinc.com/>.

[Schott11] Brian Schott, High Performance Computing Innovation Service Portal Pilots (HPC-ISP-PILOTS) Final Technical Report, Air Force Research Laboratory Information Directorate, AFRL-RI-RS-TR-2011-002, January 2011.

[Snapz] Snapz Pro X, Ambrosia Software, http://www.ambrosiasw.com/utilities/snapzprox/

[Torque] Torque Resource Manager, Cluster resources. 2011. 7 July 2011 <http://www.clusterresources.com/products/torque-resource-manager.php>.

8 Acronyms

Acronym	Definition
ASIC	Application-specific integrated circuit
CRM	Customer relationship management
EC2	Elastic Compute Cloud
EMI	Electromagnetic interference
FPGA	Field programmable gate array
GbE	Gigabit ethernet
Gbps	Gigabits per second
HPC	High performance computing
IaaS	Infrastructure as a service
IB	Infiniband
ISR	Intelligence, surveillance and reconnaissance
ITAR	International Traffic in Arms Regulations
MCMC	Markov chain Monte Carlo
MDOF	Millions of degrees of freedom
MHz	Megahertz
MRF	Markov random field
OSC	Ohio Supercomputer Center
PaaS	Platform as a service
PNCC	Power node control center
QDR	Quad data rate
RHEL	Redhat Enterprise Linux
ROI	Return on investment
SaaS	Software as a service
SDR	Single data rate
SW	Software
VNC	Virtual network computing
VPN	Virtual private network