TRANSCLOUD:
Design Considerations for a High-Performance Cloud Architecture Across Multiple Administrative Domains

Rick McGeer
HP Labs
Palo Alto, California, USA

I. INTRODUCTION

The dramatic trend of the first decade of the 21st century in the information technology industry was the emergence of *society-scale systems*: online services such as Google, eBay, iTunes, Yahoo!, Twitter, and Facebook that routinely served millions of simultaneously-connected users. These systems gave rise to entirely new programming models and systems problems: management of the data center as a single, unified, "warehouse-scale" computer, each of which had more raw computing power than existed on the planet as late as 1990; programming models for loosely-coupled, data-intensive parallel operations ("data-intensive supercomputing"), most concretely realized in the MapReduce architecture from Google and its open-source cousin, Hadoop. vast, highly-efficient distributed data stores such as PNUTS and Cassandra; the re-emergence of virtualization of time, space, and computing, to permit services to migrate instantly around the globe and radically new notions in networking to support the new programming and management models.

As revolutionary as the last decade has been, the coming decade promises a far more profound transformation: the twin emergence of the Computation Cloud and the Internet of Things. For all of its power and promise, the Cloud today is little more than a massive, well-indexed repository of text, videos, photos, and music, and a vast, universal transaction engine. The Cloud has merely automated and made vastly more efficient traditional human communications and commerce. Over the next decade, widespread availability of massive computation – the Computation Cloud -- will give to everyone the ability not only to look up what someone knows, but to discover things that no one knows.

Paired with the Computation Cloud is the Internet of Things – a world where every object houses a computer, sensor or sensors, and a connection to the network. The applications range from the trivial to the profound; milk that senses when it's going bad and tells its owner to drink it up, and buy more; automobiles that drive themselves and automatically (in coordination with other vehicles on the road, and a network of smart highways) avoid traffic jams; homes that pre-cool themselves when power is cheap and plentiful and turn off air conditioning when it is dear; smoke alarms that can tell the difference between a real fire and a ruined dinner, and call the firefighters for the former; intelligent buildings that vector people to a safe escape route in the event of an emergency; fine-grained climate sensors that can predict severe weather and evacuate people before the storm hits; and many more.

The computation cloud is tightly coupled to the Internet of things. The deployed sensors will range in capacity and bandwidth from a few bytes transmitted every few seconds to every few hours. Extracting information from all of that data is a formidable computational task, will beyond current capabilities. Only a vast new computational infrastructure will suffice to process all that data. The challenges are vast:

- Computational infrastructure. Reduction of the data requires a flexible, universal programming interface. Most data will need to be reduced at or near the point of collection; the sheer volume of data and real-time requirements ensure that. Therefore, an open, standard, and sufficiently powerful computational infrastructure ill need to be proximate to any collection of sensors.
- Data management. The Internet of Things envisions a widely-distributed set of sensors, data consumers, and fusion of information from many disparate, distributed sources. *In-situ* reduction of data at the source on a per-query basis, coordination of widely distributed queries, and a wide variety of data access mechanisms will be required, including new distributed computation mechanisms. Some early innovations are already present: MapReduce and Hadoop in the data center, and Sector and Sphere in the distributed environment.
- Networking. Efficiently connecting the Internet of Things and the Computational Cloud will require new networking stacks and protocols, both wired and wireless

- Networks must be adaptive in the face of changing conditions, using one of a number of radio frequencies and choosing routing on an adaptive basis.

In this paper, we focus on three problems:

- Ensuring that Computation Cloud users can run computation jobs wherever they have access, as simply and transparently as they now download files from multiple computers across the web
- Ensuring that execution of remote queries is done efficiently and safely for both remote user and data host
- Designing a simple, efficient, network-aware architecture for queries over geographically-distributed heterogeneous data.

II. Scalable Lightweight Federation

The computation cloud offers individuals, small companies, and researchers the ability to develop, test and deploy Internet-scale services easily and at low cost. However, many of these services require the use of multiple facilities, or users wish to transparently move their services across multiple facilities. This gives rise to the desire to federate facilities. We view facility federation not as a set of agreements between federated facilities, but rather as a set of services to developers and facilities. This approach scales easily across heterogeneous facilities, operating in different environments.

Our goal is to design a system whereby users can manage their jobs on facilities to which they have independently obtained access. This involves designing two central components.

1. An architecture and set of interfaces which permit users to easily and rapidly upload, configure, and run virtual machines
2. A service which manages a user's access to and use of facilities.

The first component is the analogue in our system to a web server (more precisely, to the *specification* of a webserver); the second, to a web browser.

The architecture we choose is the Slice-Based Facility Architecture (SFA). It is an open, standardized set of facilities to manage individual VMs and networks of VMs. In order to demonstrate its utility for this purpose, we have added support for the SFA into the Eucalyptus cluster management system. These efforts have demonstrated that the functionality in the SFA is a superset of the functionality supported by Eucalyptus; in particular, the SFA offers the ability control *slices*, or sets of virtual machines, and the topology of the network of virtual machines.

We also introduce a cloud service which manages cloud services. The user registers with the cloud service, and registers his public key with the service itself, as well as the URI of the services to which it is delegating permission. The cloud service then uses the standard SFA calls to instantiate slices, slivers, initialize and run VMs, allocate resources, and control jobs.

III. Safe Execution of Remote Jobs and Queries

This general problem: large, heterogeneous data, spread over a distributed computing infrastructure with varying connectivity and no common administrative interface – is ubiquitous through the natural, social, and engineering sciences. We are designing and implementing a computing infrastructure which addresses the distributed data management and query problem, and deploy it in a live service.

The service we will deploy is the State of the Internet service, deployed over the TransCloud infrastructure.

The basis of the query engine is a sandboxed environment which permits the user to run programs safely and efficiently at remote sites, and is based on two fundamental architectural building blocks:

1. Restricted Python (*Repy)*, a sandboxed execution environment originally used in the *Seattle* project
2. Google Native Client (*NaCl)*, a sandboxed native-code execution environment distributed with the Firefox and Chrome browsers, with x86 implementations.

The two central elements work together to provide a secure, efficient execution environment where side effects are tightly controlled. NaCl offers an efficient execution environment in a secure sandbox for computation-intensive code; safety is guaranteed by severely restricting access to system services. However, any real job requires more system services than NaCl provides. In particular, jobs in our context require access to network connectivity and resources. NaCl relies on a trusted service on the client in order to provide these services. RePy is the mechanism we choose: it has been widely deployed on a number of platforms, and offers secure access to a restricted but adequate set of system resources.Optionally, Lind can run inside a virtual machine for added isolation and security. Our initial deployment of Lind inside the TransCloud environment offers this.

IV. A Wide-Area Query infrastructure

Lind merely offers a safe execution environment. Above that, we require a distributed query/data reduction environment that is network-aware, processes and reduces data optimally with consideration of latency, bandwidth, and available processing capacity. Such an environment must support common data types, and must be extensible to new data types on an on-demand basis.

There are two central themes of the data processing environment. The first is a distribution mechanism, and the second an extensible data extraction and processing mechanism. For the first, we turn to early attempts to provide services of this form, notably Astrolabe, Hadoop, and Hadoop's wide-area cousins: Sector and Sphere. Hadoop has achieved widespread use and popularity in a cluster environment. However, extension to the wide area is still an unresolved issue. There are two major issues to be addressed:

1. Use of addressable subnets, easy in a data center environment but challenging in the wide area

2. Restrictions on bandwidth and large latencies in the wide area.

Sector, Sphere, and Astrolabe have addressed some of these issues. We will develop a hybrid approach and report on it as a deliverable from this research.

V. AN ARCHITECTURAL STACK

We have enumerated a number of issues and our solutions to them. Some (the Slice-based Facility Architecture, delegation, TCP acceleration over private networks) we have brought into being in our experimental cloud application. The others, including our distributed query architecture and secure, bare-metal execution environment, are under intense development.

We have largely built our prototype system from existing components: cluster managers such as Eucalyptus and PlanetLab, distributed programming environments (Hadoop, Sector/Sphere) and distributed query interfaces (Pig, Astrolabe). This is deliberate, both for ease of implementation and, far more important, ease of adoption.

The Computation Cloud must be ubiquitous; in order for it to be ubiquitous, it must be based on standards, *de jure* if possible, *de facto* by necessity. Every successful infrastructure has grown by standardizing and formalizing existing practice. The Web threw a hypertext skin over ftp; PlanetLab canonized virtual machines on Linux; SMTP/POP standardized sendmail. Our stack is no more than codification of common practice.

Though different in application, this resembles the classic wedding cake of the TCP/IP stack. We show the Strawman Architecture in figure 1.

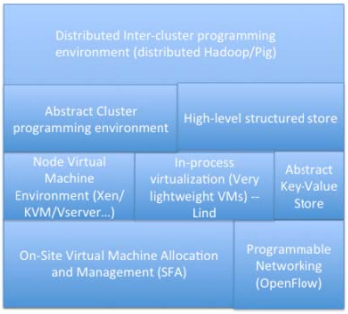

Figure 1: Strawman Architecture.

The Strawman Architecture has been instantiated and demonstrated at GEC-10 in March, 2011. Specifically, the following elements of the architecture have been deployed at a multi-site Cloud cluster spanning TU-Kaiserslautern, Northwestern University, HP Labs, Palo Alto, and the University of California, San Diego:

- An SFA-based cloud allocation and management scheme, using the PlanetLab tools and Eucalyptus as an underlying management infrastructure
- A KVM- and Xen-based node virtualization environment
- An intracluster programming environment based on Hadoop
- An intercluster, distributed programming environment

The remaining elements will be deployed over the next few months.

VI. ACKNOLEDGMENTS

Though this abstract has a single author for this panel, the work described is the work of the TransCloud team: Marco Yuen and Andy Bavier of PlanetWorks; Jessica Blaine, Eric Weu, Peter Haddad, and Alvin AuYoung of HP Labs; Yvonne