

# Automatic Installation and Deployment of Network File System and On-Demand Caching Service on Dynamically Instantiated Large Scale Batch of Virtual Machines

## On Private and Public Clouds

Sandeep Palur\* Steven Timm† Dr. Ioan Raicu\*

[psandeep@hawk.iit.edu](mailto:psandeep@hawk.iit.edu) [timmm@fnal.gov](mailto:timmm@fnal.gov) [iraicu@cs.iit.edu](mailto:iraicu@cs.iit.edu)

\*Department of Computer Science, Illinois Institute of Technology, Chicago IL, USA

†Scientific Computing Division, Fermi National Accelerator Laboratory, Batavia IL, USA

**Abstract** – *Big scientific experiments usually need a lot of virtual machines for running scientific workflows on private and public clouds. Virtual machine images can be large and the software they need inside them can be constantly changing. The best solution is to use CERN Virtual Machine File System - read only, runtime download, and caching http file system. It is a read-only network file system that provides access to files from a CVMFS Server over HTTP. When CVMFS client runs on a groups of worker nodes that share the same cloud, a HTTP web proxy, inside the same cloud, can be used to cache the file system contents, so that all subsequent requests for that file will be delivered from the local HTTP web proxy) and doesn't have to hit the Internet. Typically, a High Energy Physics (HEP) computing site has a local or regional Squid HTTP web proxy, with the central CVMFS servers located at the main laboratory, such as CERN for the LHC experiments. Each VM has a list of the available Squid servers and, in most cases, the Squids are remote. The optimal Squid may be different depending on the location of the cloud. Further, one can imagine dynamically instantiating Squid servers in an opportunistic cloud environment to meet application demand. As a result, we use Shoal as a service that can dynamically publish and advertise the available Squid servers. In this work, we automate installation and deployment of CVMFS(network file system), Squid(on-demand caching service) and Shoal(squid cache publishing and advertising tool designed to work in fast changing environments) on dynamically instantiated large scale batch of virtual machines on Fermi Cloud (private cloud) and Amazon Web Services (Public cloud).*

## I. INTRODUCTION

The CERN Virtual Machine File System (CVMFS) [1] is widely adopted by the High Energy Physics

(HEP) community for the distribution of project software. CVMFS is a read-only network file system that provides access to files from a CVMFS Server over HTTP. When CVMFS is used on a cluster of worker nodes, a HTTP web proxy can be used to cache the file system contents, so that all subsequent requests for that file will be delivered from the local HTTP proxy server. Typically, a HEP computing site has a local or regional Squid HTTP web proxy [2], with the central CVMFS servers located at the main laboratory, such as CERN for the LHC experiments.

The use of IaaS cloud resources is becoming a realistic solution for HEP workloads [3, 4], and CVMFS is an effective means of providing the software to the virtual machines (VMs). Each VM has a list of the available Squid servers and, in most cases, the Squids are remote. The optimal Squid may be different depending on the location of the cloud. Further, one can imagine dynamically instantiating Squid servers in an opportunistic cloud environment to meet application demand. However, there is currently no mechanism other than Shoal for locating the optimal Squid server. As a result, we use Shoal as a service that can dynamically publish and advertise the available Squid servers. Shoal is ideal for an environment using both static and dynamic Squid servers.

### A. CERN Virtual Machine File System

The CernVM File System (CernVM-FS) provides a scalable, reliable and low maintenance software distribution service. It was developed to assist High Energy Physics (HEP) collaborations to deploy software on the worldwide-distributed computing infrastructure used to run data processing applications. CernVM-FS is implemented as a POSIX read-only file system in user space (a FUSE module). Files and directories are hosted on standard web servers and mounted in the universal namespace /cvmfs. Internally, it uses content-addressable storage and Merkle trees in order to maintain file data and meta-data. CernVM-FS uses outgoing HTTP connections only, thereby it avoids most of the firewall issues of other network file systems. It is actively used by small and large HEP collaborations.

In many cases, it replaces package managers and shared software areas on cluster file systems as means to distribute the software used to process experiment data.

### *B. Squid*

Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages. Squid has extensive access controls and makes a great server accelerator. It runs on most available operating systems, including Windows and is licensed under the GNU GPL.

### *C. Shoal*

Shoal is divided into three logical modules, a server, an agent, and a client. Each package is uploaded to the Python Package Index [5] (the standard method of distributing new components in the Python language).

Each component is designed to provide the functionality of different parts of the system as follows:

**Shoal Server** - is responsible for the following key tasks:

1. Maintaining a list of active Squid servers in volatile memory and handling AMQP messages sent from active Squid servers.
2. Providing a RESTful interface for Shoal Clients to retrieve a list of geographically closest Squid servers.
3. Providing basic support for Web Proxy Auto-Discovery Protocol (WPAD).
4. Providing a web user interface to easily view Squid servers being tracked.

**Shoal Agent** - is a daemon process run on Squid servers to send an Advanced Message Query Protocol (AMQP) [6] message to Shoal Server on a set interval. Every Squid server wishing to publish its existence runs Shoal Agent on boot. Shoal Agent sends periodic heartbeat messages to the Shoal Server (typically every 30 seconds).

**Shoal Client** - is used by worker nodes to query Shoal Server to retrieve a list of geographically nearest Squid servers via the REST interface. Shoal Client is designed to be simple (less than 100 lines of Python) with no dependencies beyond a standard Python installation.

Shoal Server runs at a centralized location with a public IP address. For agents (i.e. Squid servers), Shoal Server will consume the heartbeat messages sent and maintain an up-to-date list of active Squids. For clients, Shoal Server will return a list of Squids organized by geographical distance and load. For regular users of Shoal Server, a web server is provided. The web server generates dynamic web pages that display an overview of Shoal. All of the tracked Squid servers are displayed and updated periodically on Shoal Server's web user interface, and all client requests are available in the access logs.

AMQP forms the communications backbone of Shoal Server. All information exchanges between Shoal Agent (Squid Servers) and Shoal Server are done using this protocol, and all messages are routed through a RabbitMQ [7] Server.

## II. DESIGN AND IMPLEMENTATION

This project aims to automate installation and deployment of CVMFS(network file system), Squid(on-demand caching service) and Shoal(squid cache publishing and advertising tool designed to work in fast changing environments) on dynamically instantiated large scale batch of virtual machines on Fermi Cloud (private cloud) using Puppet Master/Agent and Amazon Web Services (Public cloud) using Serverless Puppet.

As a part of this work, we developed puppet modules and scripts for installing and deploying shoal client, agent and server, script to dynamically update the proxy address. We also fixed potential bugs in Shoal Server and made it suitable to publish Squid Servers running on EC2 instances that does not have a static public IP.

### *A. Architecture*

The architecture of large scale batch of dynamically instantiated FermiCloud and EC2 worker nodes provided with network file system, on-demand caching service and a cache publishing and advertising tool is shown in Figure1. It consists of the following components:

a) **Worker Node Installed with Shoal Client** – When a worker node is instantiated dynamically, CVMFS client and Shoal Client are installed on start up of the machine. Shoal Client is a cron job that queries the Shoal Server using the REST interface to get the closest Squid Server and is configured to run every 2 hours. Shoal Client updates the proxy address in the CVMFS configuration file. So that when

CVMFS client tries to download any software from CVMFS server, the request passes through the Squid Server, whose IP address is configured in CVMFS configuration file.

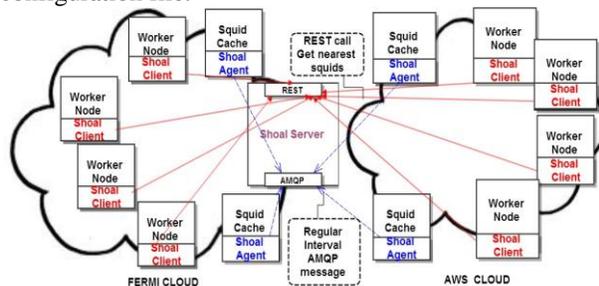


Figure1: Architecture of Dynamically Instantiated Fermi Cloud and EC2 Instances with On-Demand Caching Service and a Cache Publishing Service

**b) Squid Server Installed with Shoal Agent –**

When a server node is instantiated dynamically, Squid Server and Shoal Agent are installed on the start up of the machine. Shoal Agent running alongside with Squid Server, sends periodic heartbeat messages (IP Address, Load, etc) to the Shoal Server typically every 30 seconds.

**c) Shoal Server Installed with RabbitMQ Server and Apache Server-**

When a server node is instantiated dynamically, Shoal Server is installed on start up of the machine. Shoal Server provides two major functions: provides a RESTful interface (hosted on Apache server) for Shoal Clients to retrieve a list of geographically closest Squid servers and maintains a list of active Squid servers (active squid servers send AMQP messages to RabbitMQ server) in volatile memory.

Since we have our Worker Nodes split over in two different clouds (AWS and FermiCloud), the idea is to install sufficient Squid Servers on both the clouds and restrict the Worker Nodes to use the Squid Servers in their local cloud for the following reasons:

- 1) We don't want to open Fermilab cache servers to outside internet
- 2) Reduce data transfer from Internet.
- 3) Faster data transfers.
- 4) Reduce Latency

When a Shoal Client on any Worker Node queries for nearest Squid Servers, it is responded back with the IP addresses of Squid Servers running inside the local cloud because the Shoal Server finds the closest Squid server to the Worker Node. Thus only the first

Worker Node in each cloud downloads the software from Internet (CVMFS server) and rest of the Worker Nodes that needs the same software, takes it from the local Squid Server.

**III. CONCLUSION AND FUTURE WORK**

We automate installation and deployment of CVMFS(network file system), Squid(on-demand caching service) and Shoal(squid cache publishing and advertising tool designed to work in fast changing environments) on dynamically instantiated large scale batch of virtual machines on Fermi Cloud (private cloud) and AWS(public cloud) by installing and deploying all the required software on start up of the instances and also restrict the Worker Nodes to use the Squid Servers running on the local cloud thereby reducing the number of hits to the Internet.

Our future work includes:

- a) Installation and deployment on a sum of 1000 virtual machines on both AWS and Fermi Cloud
- b) Benchmarking this work at high scales.

**IV. REFERENCES**

[1] J. Blomer et al, Status and future perspectives of CernVM-FS J. Phys.: Conf. Ser. 396052013, doi:10.1088/1742-6596/396/5/052013  
 [2] Squid - HTTP proxy server <http://www.squid-cache.org>  
 [3] F. H. B. Megino et al. Exploiting Virtualization and Cloud Computing in ATLAS J. Phys.: Conf. Ser. 396032011, doi:10.1088/1742-6596/396/3/032011  
 [4] I. Gable et al, A batch system for HEP applications on a distributed IaaS cloud J. Phys.: Conf. Ser. 331062010, doi:10.1088/1742-6596/331/6/062010  
 [5] Python Package Index <https://pypi.python.org/>  
 [6] S.Vinoski, Advanced Message Queuing Protocol, IEEE Internet Computing 10 87, doi:10.1109/MIC.2006.116  
 [7] RabbitMQ - AMQP Messaging software, <http://www.rabbitmq.com>