Tools and Techniques for Managing Virtual Machine Images

26 August 2008

Håvard Bjerke, Dimitar Shiyachki,
Andreas Unterkircher, Irfan Habib

Agenda

- Libfsimage
  - A library for creating VM images
- OS Farm
  - Repository of tailored VM images
- Content Based Transfer
  - A technique for efficient transfer of VM images

CERN's use case

- We need to test our Grid middleware on an increasing number of different OS flavours
- Need to increase the efficiency and reduce turnaround time of testing and certification
- Testing is most efficiently done in Virtual Machines
- A repository of unused VM images provides an easy way of setting up clean testing environments

Libfsimage

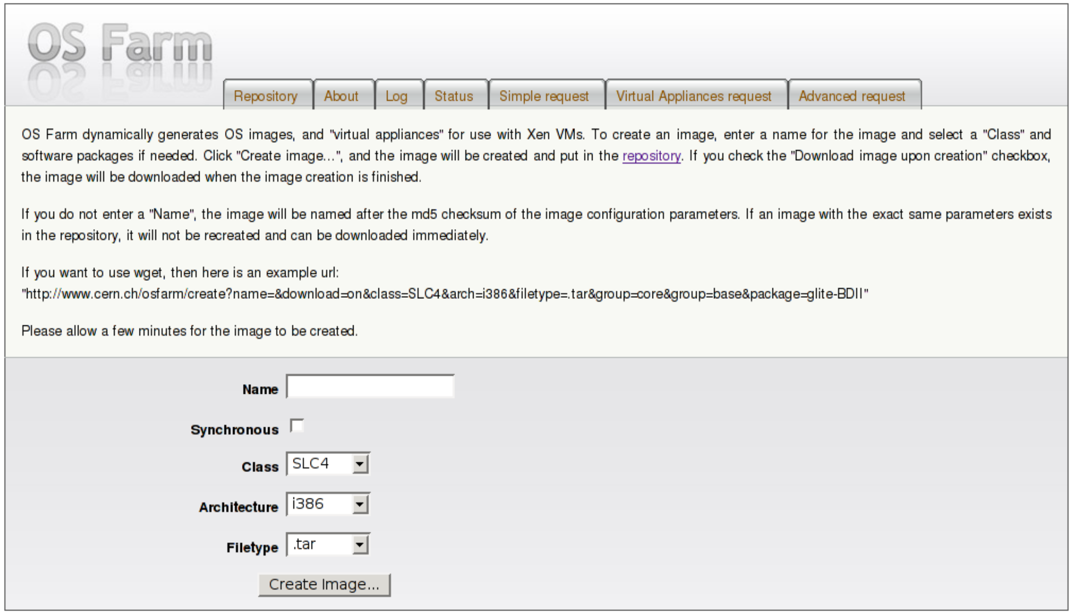
- Driven by the need for testing software in a clean and reproducible environment
- Standalone tool and library for creating VM images
- Uses native package management tools in a chroot environment
- Can create many different Linux flavours
  - i386 and x86_64
  - Ubuntu, Debian, Fedora, Scientific Linux (CERN), CentOS

OS Farm

- Uses libfsimage for creating core images
- Adds a graphical user interface for configuring and adding software to images
- Provides a repository of stored images
- Optimizes the image creation process by caching shared data

OS Farm interfaces

- Web interface



OS Farm dynamically generates OS images, and "virtual appliances" for use with Xen VMs. To create an image, enter a name for the image and select a "Class" and software packages if needed. Click "Create image...", and the image will be created and put in the repository. If you check the "Download image upon creation" checkbox, the image will be downloaded when the image creation is finished.

If you do not enter a "Name", the image will be named after the md5 checksum of the image configuration parameters. If an image with the exact same parameters exists in the repository, it will not be recreated and can be downloaded immediately.

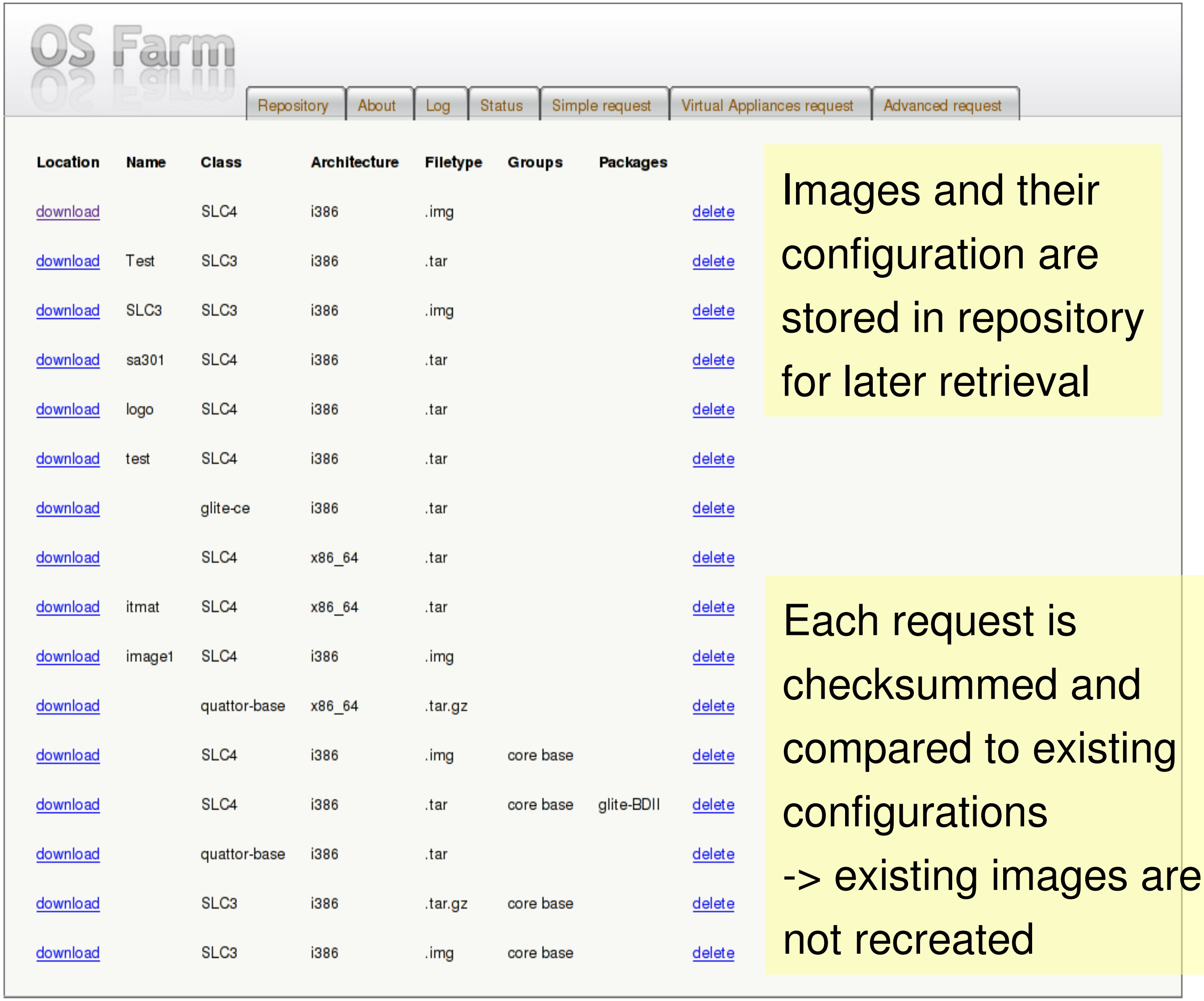
If you want to use wget, then here is an example url:
"http://www.cern.ch/osfarm/create?name=&download=on&class=SLC4&arch=i386&filetype=.tar&group=core&group=base&package=glite-BDII"

Please allow a few minutes for the image to be created.

Name	
Synchronous	□
Class	SLC4
Architecture	i386
Filetype	.tar

- SOAP web service interface
- XML specification

VM Image Repository



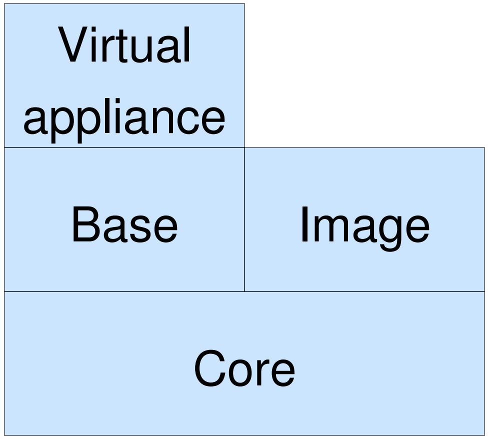
Location	Name	Class	Architecture	Filetype	Groups	Packages	
download		SLC4	i386	.img			delete
download	Test	SLC3	i386	.tar			delete
download	SLC3	SLC3	i386	.img			delete
download	sa301	SLC4	i386	.tar			delete
download	logo	SLC4	i386	.tar			delete
download	test	SLC4	i386	.tar			delete
download		glite-ce	i386	.tar			delete
download		SLC4	x86_64	.tar			delete
download	itmat	SLC4	x86_64	.tar			delete
download	image1	SLC4	i386	.img			delete
download		quattor-base	x86_64	.tar.gz			delete
download		SLC4	i386	.img	core base		delete
download		SLC4	i386	.tar	core base	glite-BDII	delete
download		quattor-base	i386	.tar			delete
download		SLC3	i386	.tar.gz	core base		delete
download		SLC3	i386	.img	core base		delete

Images and their configuration are stored in repository for later retrieval

Each request is checksummed and compared to existing configurations
-> existing images are not recreated

Optimization

- Four different image stages: Core, Base, Image, and Virtual Appliance
- Image generation process is optimized by caching and sharing lower stages of an image



Virtual appliance

Base

Image

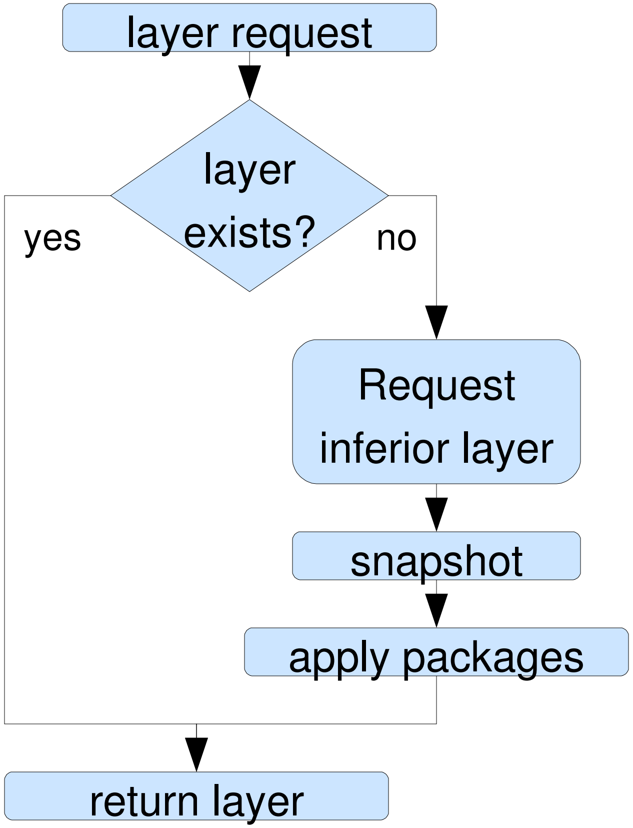
Core

VM image stages

- libfsimage is used to create *core*, or minimal, VM images
- *Core* consists of software critical to boot and access VM
  - Can be shared between VM image configurations
- *Base* consists of software critical for Virtual Appliances
  - Can be shared among Virtual Appliances

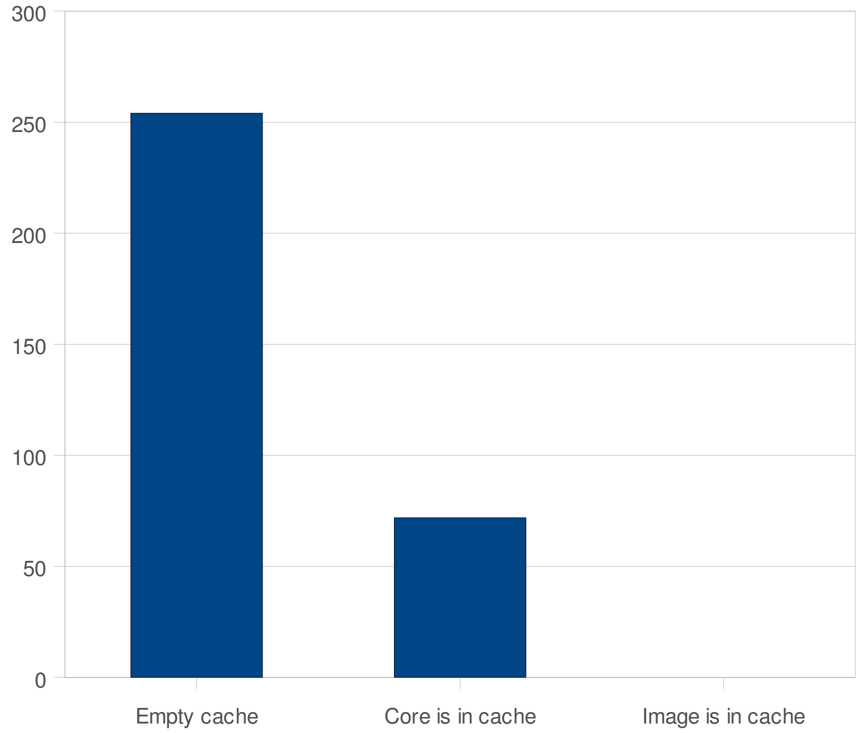
Copy-on-Write staging

- Core and Base stages are kept in cache
- LVM snapshots (copy-on-write) are used for instantaneous staging



Performance

- Any stage can be kept in cache

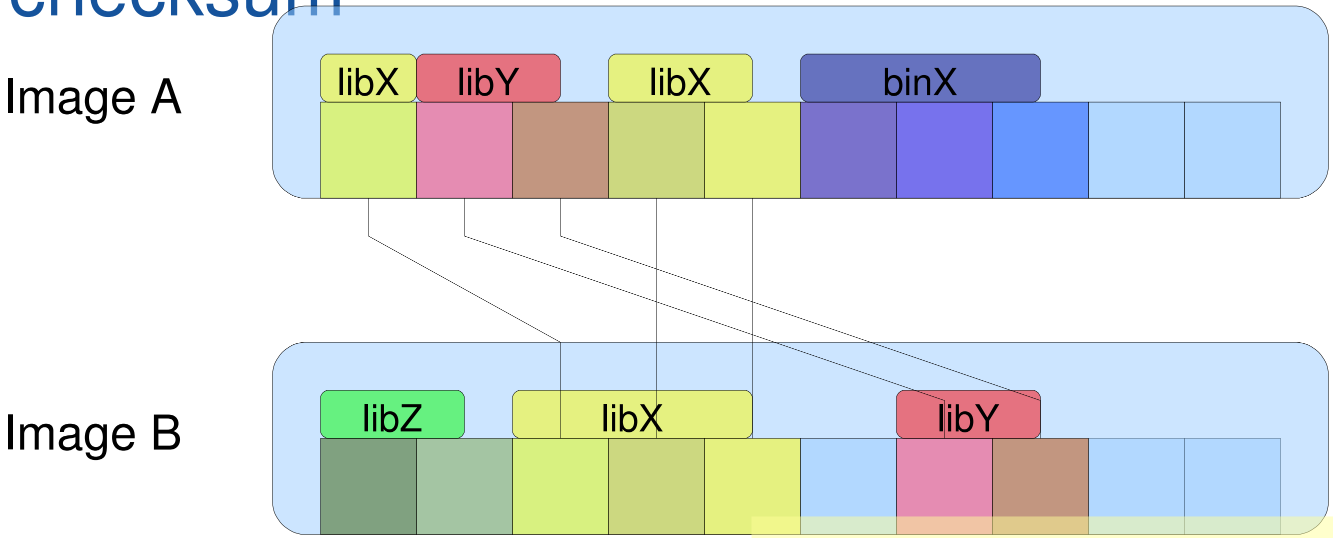


Content-Based Transfer

- VM images are big
  - ~ 300 MB to several GB
  - Congests the network
  - Anything scheduled for a VM will have to wait for the image transfer to finish
- Observation from Content-based Addressing
  - Most images are relatively similar
  - Not always necessary to transfer the whole image; just transfer the delta

Filesystems

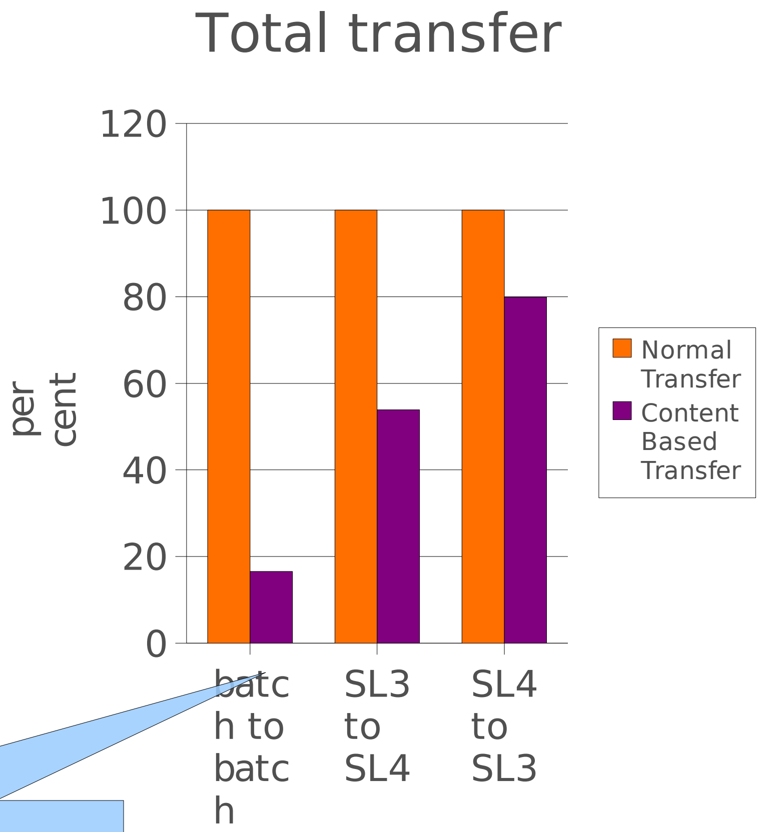
- Each file starts on a block boundary
- Identical blocks can be identified with a hash checksum



Only 50 % of Image A needs to be transferred if Image B is already at the destination

Image comparisons

- Two typical batch machines (5.3 GB)
    - 84 % common blocks
- Scientific 3 (343 MB) and Scientific 4 (762 MB)
    - SL3 -> SL4
        - 48 % common blocks
    - SL4 -> SL3
        - 22 % common blocks

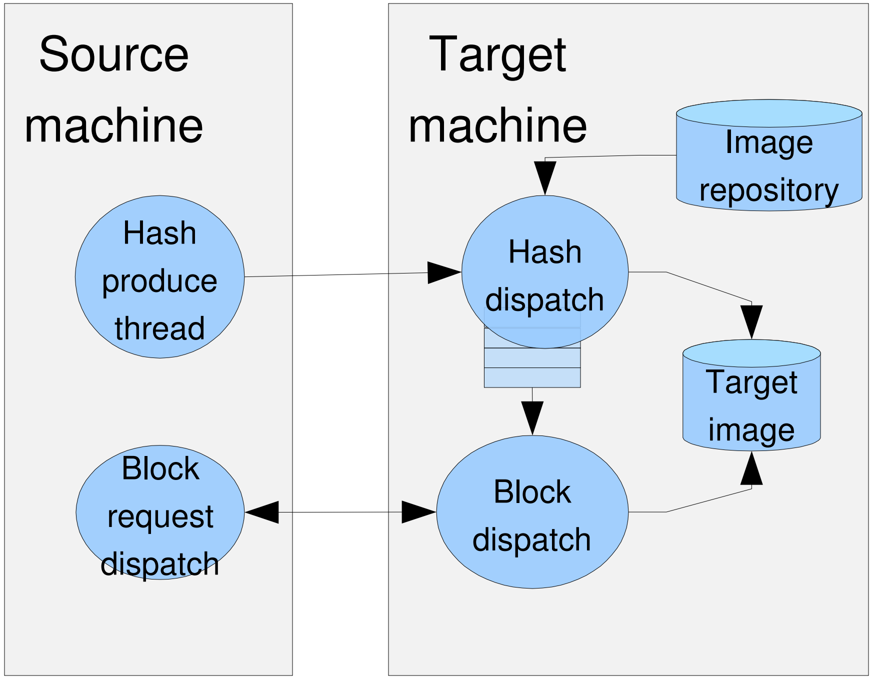


Fraction of full image data needed to transfer, including hash table

Cost

- Generating hash tables for source file and target repository – linear cost
- Accessing hash tables
  - Java has a convenient constant-time hash table
- Hash table data overhead
  - Depends on
    - hash function, e.g. SHA is 20 bytes
    - block size – usually 4096 bytes
  - 0.48 to 2.0 % of the image size

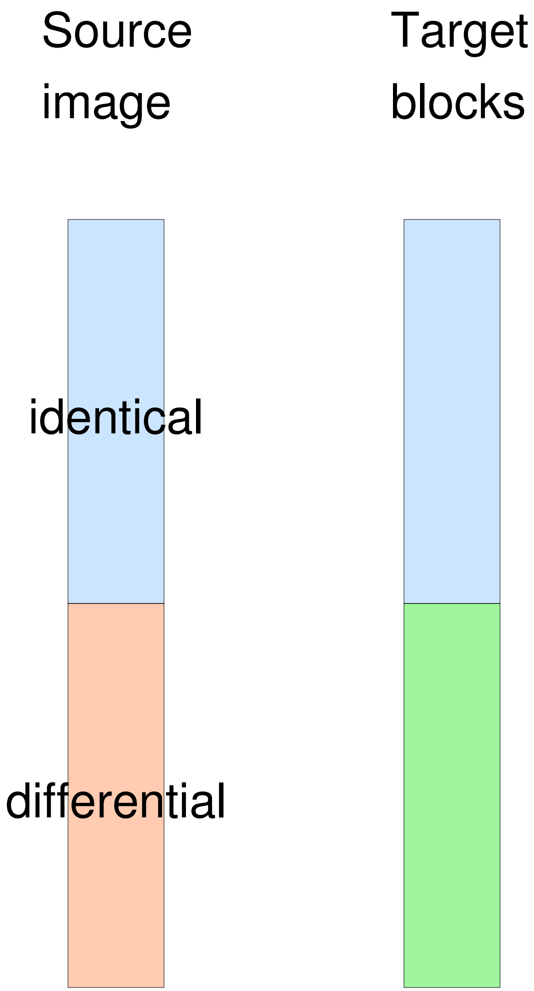
Implementation



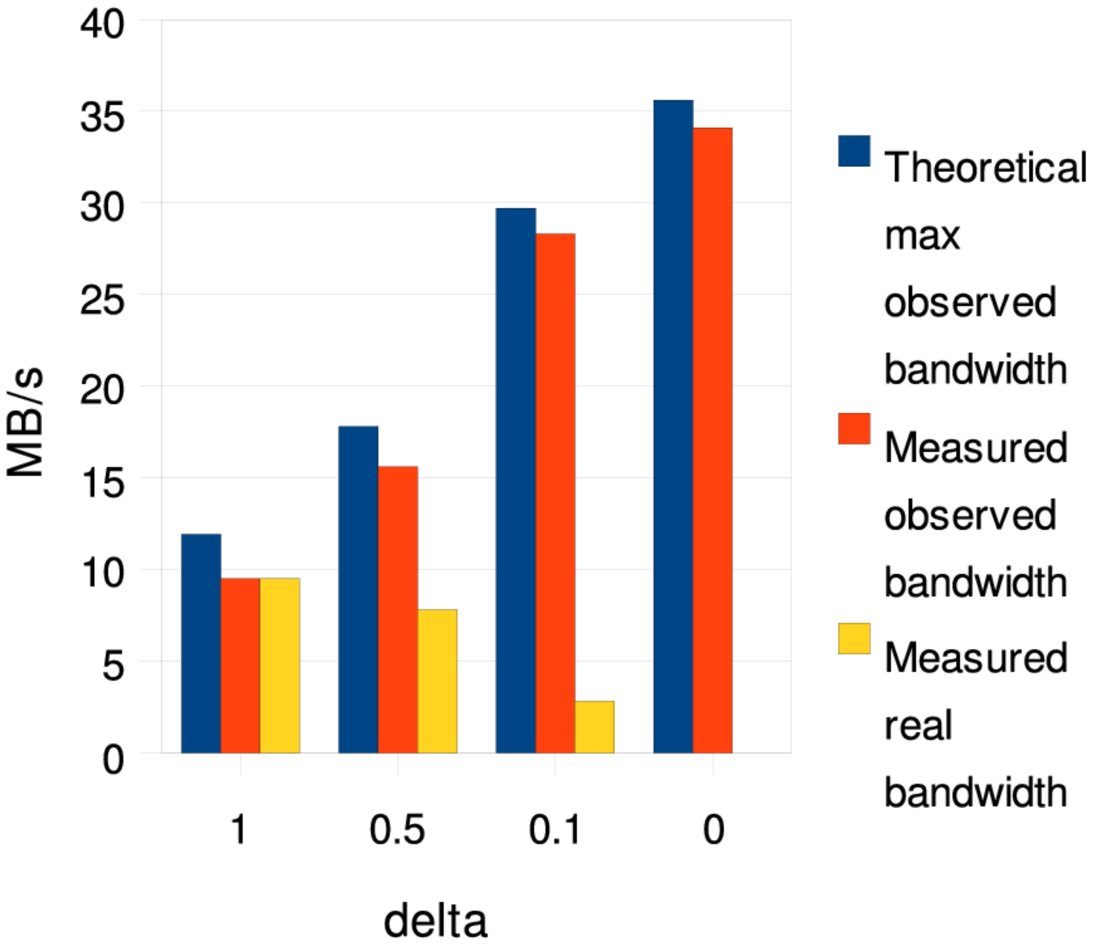
- Multi-threaded
- Hash calculation and data transfer pipelined
- Implemented in Java

Efficiency

- A source image which to transfer consists of a *differential* and an *identical* part
- “Observed Bandwidth”
  - $|image| / time_{diff}$
- Theoretical maximum can be calculated



Measurement



Future goals

- Signed images
  - Signed by image author or image creation service
- Automated image testing
- Contextualization
  - One VM image should be able to run on any infrastructure, e.g. Amazon EC2, LCG, my laptop
- Open Virtual Machine Format
  - Open DMTF standard for VM images

More information & Questions

- OS Farm
  - http://cern.ch/osfarm
- Content Based Transfer
  - http://hbjerke.web.cern.ch/hbjerke/cba/cba.xml