BaBar LTDA Architecture and Security

S. Luitz

PPA SCA Meeting

3/18/2011

The BaBar Long Term Data Access System

- Provide and maintain the capability to perform BaBar data analyses until at least end of 2018
  - code & repositories, data, databases, documentation, storage, CPU
- Minimize the effort needed needed to support and maintain the system
  - Think of: hardware, operating system upgrades, tool upgrades, code validation, maintaining documentation, interaction with data center infrastructure, etc., etc.
- Puting everything "in a box" seems a very reasonable approach
  - A controlled environment even simplifies documentation and user support

LTDA Design Premise

- It is "easier" (== cheaper, more practical) to maintain the complete BaBar analysis environment on a frozen operating system infrastructure than to actively migrate the software environment forward as needed
  - Note: Different expertise required for the two alternatives
- Premise not automatically valid – areas of concern:
  - Hardware support
    - Does the old O/S still run on the next-gen hardware?
    - H/W life cycle in contemporary data centers ~4-5 years. Need to replace failed hardware sooner.
  - System management
    - Maintaining back-versioned O/S comes at a potentially high cost
  - Security
    - Critical exploits **will** be discovered in the unsupported O/S
    - Fixes will be labor- and knowledge intensive
- Can we provide an environment where this premise is valid?

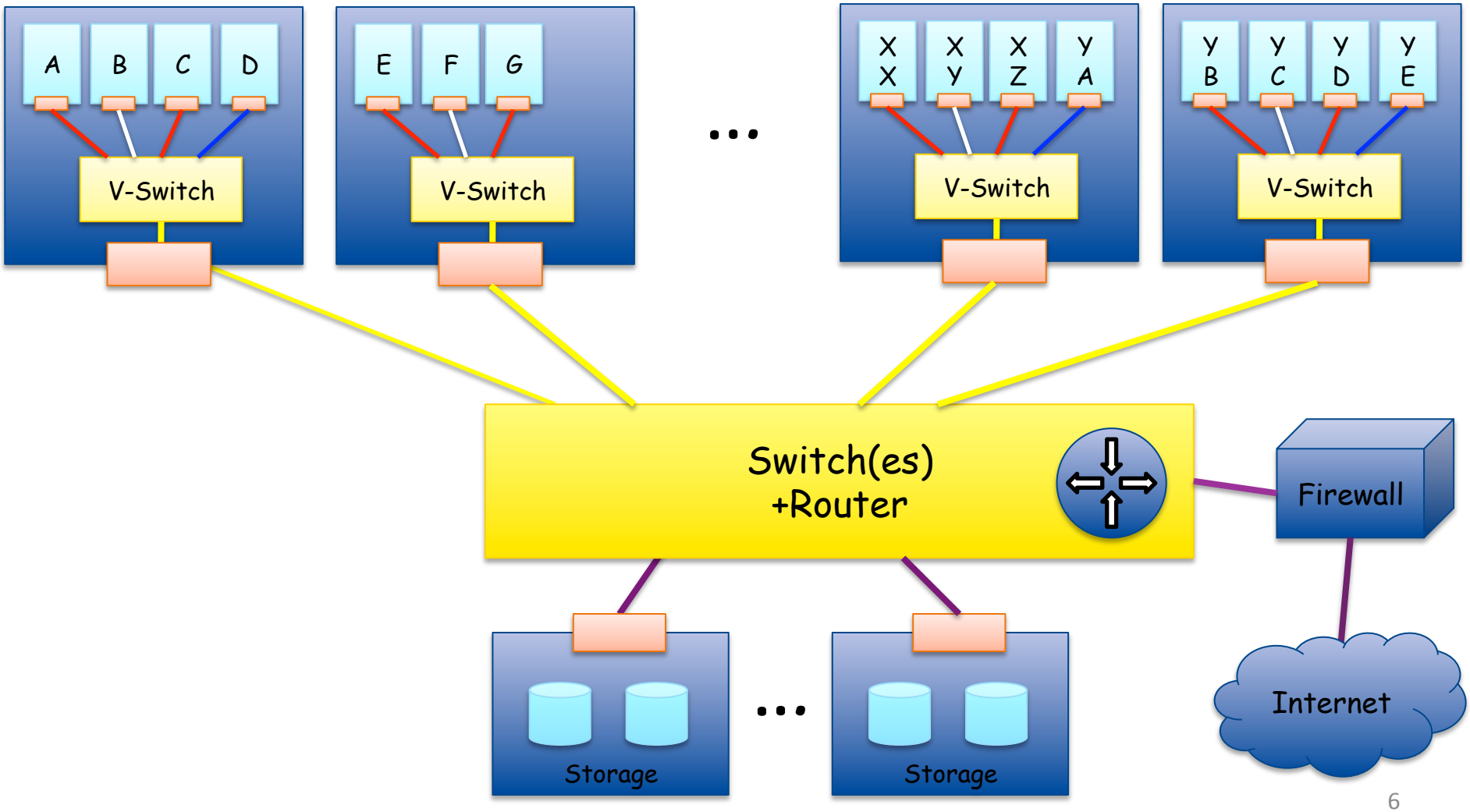
Virtualization to the Rescue

- Virtualization = Providing an abstracted "virtual" hardware environment (containers) in which you run operating system instances (virtual machines, VMs)
  - In the commodity computing world performant virtualization has only become available a few years ago (with Intel/AMD CPU hardware & instruction set support for virtualization)
- Hardware virtualization
  - Solves the hardware support problem for the foreseeable future by providing a frozen hardware environment to the frozen O/S
- System images (an application of storage virtualization)
  - Solves the system administration problem by replacing management of a live large cluster with management of a small number of O/S images

LTDA – Virtual Data Center

- LTDA system needs 100s of (virtual) nodes
- To operate a "virtual data center" on this scale you need tools:
  - Create and destroy virtual computers, configure them, assign network address, etc.
  - The software equivalent of the CD TechCoords
- Software layer known as Infrastructure as a Service (IaaS)
  - Buzzword-compliant: "Private Cloud"
- LTDA uses Nimbus as IaaS layer
  - Open Source (http://www.**nimbus**project.org)
- After IaaS has instantiated virtual nodes, one can use a batch system (LRM) to run jobs on them
  - LTDA uses Condor (http://www.cs.wisc.edu/**condor**)
    good at managing intermittently available resources

Virtual Data Center – Example



Security (1)

- Unfortunately, virtualization does ***NOT*** add (a lot of*) security:
  - It doesn't make a difference whether a real or virtual machine runs an exploitable or compromised operating system!
  - … and that's exactly what we want to do!
  - There are also lots of unpleasant SLAC policies wrt. outdated systems
- Security doesn't deal in absolutes → **risk-based approach**
  - Note: for secure design purposes it helps to assume that systems that ***can be*** (easily) compromised ***are actually*** compromised.
- Threats, Vulnerabilities, Mitigations/Controls
  - Methodology:
    - Scope, determine threats & vulnerabilities, design mitigations and controls to reduce risk to an acceptable level, implement, test/review/monitor/audit, improve
  - If you remember your SLAC safety training (hopefully!) – computer security methodology is quite similar to ISEMS:
    - Scope – analyze hazards – apply controls – work – review & improve
- Let's look at some examples (won't do the full analysis & cycle) …

*) Dynamic creation of VMs from read-only images adds a *small* amount of "security" by preventing compromises from persisting across VM destruction

Threats & Mitigations in the LTDA System

- Examples (non-exhaustive):
  - "Someone" compromises a machine and …
    - … uses it to attack other machines in the world.
      - Mitigation: isolate from the world (firewall & "DMZ")
    - … steals a credential (e.g. password) and uses it on other machines at SLAC or in the world.
      - Mitigation: no internal LTDA credentials usable elsewhere
    - … impersonates other users and deletes their data within LTDA
      - Mitigation: e.g. frequent backups
    - … uses it to do password cracking (brute force).
      - Mitigation: e.g. resource monitoring to detect anomalous behavior
    - … impersonates another user and tweaks their code so that it generates incorrect results
      - Mitigation: history in source code control system, self-consistency of our data, internal scientific review process, etc.
  - And there are different kinds of "Someone"
- Condense this into a few very simple "ground rules" for LTDA security …

Security(2)
The Ground Rules

- Common sense:
  - Assuming that (easily) compromisable components are compromised, the LTDA system shall not be able to do harm to (non-LTDA) systems at or outside of SLAC
    - Isolation (devil is in the detail!), monitoring
  - The LTDA system shall prevent *accidental* modification or deletion of data
    - Assuming that it contains components that can (easily) be compromised it is close to impossible to protect against intentional/malicious modification or deletion of data
    - Maintaining user identity for access to back-versioned O/S within the LTDA system is not "security-critical". Can be done with simple mechanisms, e.g. the LRM or the new ssh .shosts. Significant simplification!
  - Compromise of components of the LTDA system shall be detectable
    - Logging, monitoring, etc.
- Input to the architecture

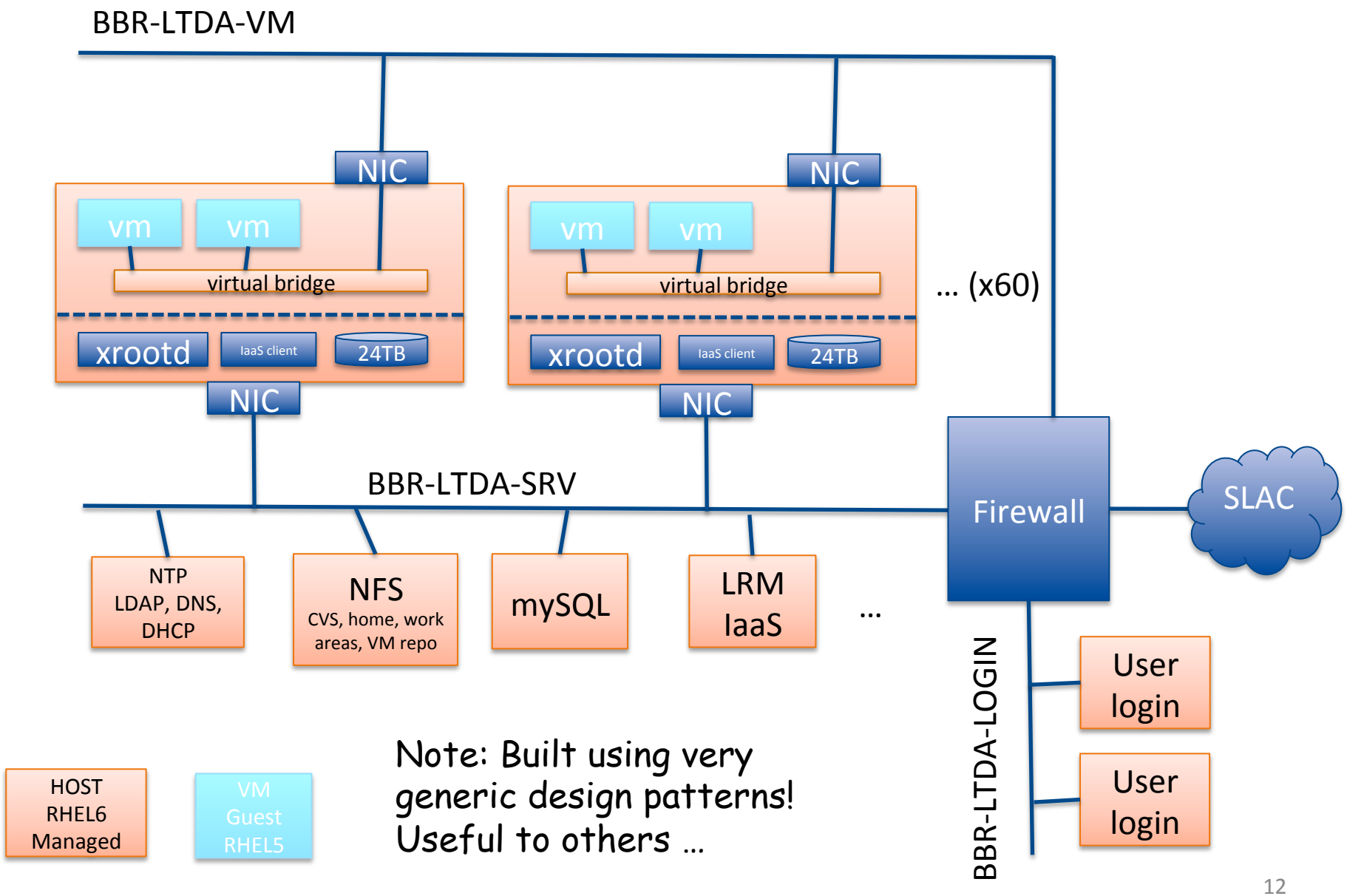
Integration into Infrastructure

What needs to be managed & patched?

- Systems that need to be trusted (because they enforce security policy)
  - User login machines (authenticate users, authorize access to the system, enforce security policy)
  - Firewall (enforces security policy)
  - Virtual machine hosts (enforce isolation, security policy)
  - Xrootd servers (tape storage backend needs to trust them)
- What do we want managed centrally?
  - File servers and database servers (we want backups done by SLAC CD)
  - IaaS & batch scheduler servers (because we don't want to manage them ourselves)
  - Infrastructure replicas (NTP, DNS, …, don't want to manage them ourselves)
- → Everything on these two lists needs to be up-to date and maintained
  - Standard SLAC computing management tools can (and will) be used for that
  - Would easily fit into any other (e.g. non-SLAC) managed infrastructure
- The only back-versioned components are batch workers and a few interactive hosts for debugging
  - All in VMs

Hardware Platform

- Plan:
- 60 Servers: Intel Dual-Hex-Core, 3GHz, 48GByte RAM, 24TByte Disk
  - 4 existing systems in prototype
  - Purchase remaining systems in 2 stages
- Cisco 6500-720 network switch (Gbit Ethernet)
- NFS server
- A few dedicated infrastructure & login servers


BBR-LTDA-VM
NIC
vm
vm
virtual bridge
xrootd
IaaS client
24TB
NIC
NIC
vm
vm
virtual bridge
xrootd
IaaS client
24TB
NIC
... (x60)
BBR-LTDA-SRV
Firewall
SLAC
NTP
LDAP, DNS,
DHCP
NFS
CVS, home, work
areas, VM repo
mySQL
LRM
IaaS
...
BBR-LTDA-LOGIN
User
login
User
login
HOST
RHEL6
Managed
VM
Guest
RHEL5
Note: Built using very
generic design patterns!
Useful to others ...

Use Case

- User logs in to load-balanced "login" cluster on the LOGIN network
  - Prepares code on login machines
  - Submits jobs to compile and run analysis
    - LRM asks IaaS to create VMs, LRM then executes jobs on the VMs
    - VMs access data through xrootd and files through NFS
    - VMs write output data
  - Uses login machines to do interactive analysis or copies result data out of the LTDA system for interactive analysis elsewhere
  - Can create VMs for interactive login (e.g. compile, debug)

Storage

- NFS
  - Home directories, AWG / group space, build areas, CVS repository
  - VM image repository
  - Not directly accessible from outside the LTDA system
  - Backed up as appropriate
  - Accessible from VMs as appropriate (through firewall)
    - A bit more about that later
- Xrootd
  - Accessible from VMs (through firewall)
  - VM hosts double-act as storage servers
  - HPSS backend
- MySQL
  - Backed up
  - Precise access controls still TBR/D (to be revisited/designed)
- General issue for storage (since it has so much state ;-)
  - Graceful migration of all BaBar analysis into LTDA – how does the "inside" interact with the "outside" during the transition

What the Firewall will block/allow Connection Matrix

- Protect SLAC & World from direct access by VM network (block all)
- Protect LOGIN network from VM network
- Allow ssh login from LOGIN network to VM machines
- Allow well-defined services between VM and SRV network
  - Infrastructure (DHCP, LDAP, NTP)
  - File service (xroot, NFS)
  - LRM (Batch scheduling)
- Allow USER & SRV networks to use SLAC infrastructure services
  - including backup and tape access
- Allow ssh login to machines on USER and SRV networks
- Tradeoff restrictions vs ease of management. To be resolved

From / To	SLAC	BBR-LTDA-LOGIN	BBR-LTDA-SRV	BBR-LTDA-VM
SLAC	1	ssh, infrastructure	Infrastructure, tape access, backup	X
BBR-LTDA-LOGIN	infrastructure	1	ssh, NFS, infrastructure	ssh (for interactive VM)
BBR-LTDA-SRV	infrastructure, tape access, backup	NFS	1	LRM, DHCP, xroot, NFS, LDAP
BBR-LTDA-VM	X	X	LRM, DHCP, xroot, NFS	1

Security (3)
The Devil is in the Detail

- Or: Where security could be at odds with usability
  - E.g. protection of home directories
    - Security-sensitive files: login scripts, ssh keys
    - Problematic if a compromised VM can overwrite these
    - Can affect outside world through login machines
  - Mitigations:
    - Login machines are not allowed to execute code or scripts from areas that are writable by the VMs. At least for critical areas such as home directories.
    - No outgoing connectivity from login machines (can ssh to login machines but not out)?
  - Many other issues, some similar
    - Scripts in areas writable by the VMs
    - Database access from outside the LTDA system
    - Permission/authorization granularity much more coarse inside LTDA (no AFS ACLs, NFS ACLs much less granular). How to map?
  - Approach
    - Prototype system
    - Start with "default deny"
- Current design looks approvable to SLAC computer security
  - we will need to document the design, have a security review and write a security plan

User Acceptance -or- Managing Expectations

- Probably the toughest problem – we need to be very clear
  - The LTDA system is designed to do "one thing" well and with acceptable risk: BaBar analysis in a limited and controlled environment
  - It is not a general-purpose computing platform, you probably won't be able to do things like reading your e-mail on the system (for that we have general-purpose systems at SLAC)
  - Tools in the VM environment will be limited to the bare minimum that is needed to build, run and debug BaBar code (don't expect the latest and greatest ROOT version there). After all the software environment there is frozen!
- My experience from years of trying to remove external dependencies in the IR2 Online system – it's very hard
- → Start out with a minimum system and default deny stance
  - Try to use the system
    - Open specific access if really necessary and justified
    - Add software and tools if really necessary and justified

Summary

- There now is a concrete system, network and security design for the BaBar LTDA system
  - Fits into the SLAC computing infrastructure
    - Including system and network management
    - Based on generic design patterns – useful for others @ SLAC and elsewhere (that's another R&D project…)
    - Will need a technical design review
  - Mitigates security risks to acceptable levels
    - → need security plan & review
- Implementation in progress
  - Well advanced
  - But still "exploring as we go" in some areas
    - E.g. file systems, directories, autofs vs. amd, etc., etc.