Global Filesystems in the Real World: Comparing Experiences with AFS and NFS

Phillip Moore
http://www.openefs.org
http://www.yume.org/phil

Overview

- Comparison of Two Global Filesystems:
  - AFS/VMS at Morgan Stanley (MS)
  - NFS/EFS at Another Large Bank (ALB)
- Introduction to EFS

Aurora at Morgan Stanley

- Heterogenous Distributed Systems Architecture
- The key components:
  - Global Filesystem (AFS/VMS)
  - Distributed Systems Database (DSDB)
  - Standardized Desktop (CDE/TED)
  - Dataless UNIX Clients

AFS/VMS at Morgan Stanley: System Architecture

- Single, Enterprise-wide "global virtual cell", built using multiple physical AFS cells.
- Single Authentication Domain (one krb5 realm)
- Automated Systems Management Software (VMS)
- Location-agnostic globally consistent namespace (/ms)
  - RW data (/ms/user, /ms/group, /ms/dev)
  - RO distributed, replicated data (/ms/dist)

AFS/VMS at Morgan Stanley: VMS Software

- Secure, client/server based infrastructure
- Automates all filesystem and namespace operations
  - Creating, moving user home directories (/ms/user)
  - Simple shared group directories (/ms/group)
- Complete software deployment lifecycle management
  - /ms/dev, /ms/dist and the "MPR" namespace
  - Fully automated, global software distribution

AFS/VMS at Morgan Stanley: Historical Timeline

- 1994: AFS Introduced for R&D
- 1995: First Production dataless AFS clients deployed
- 1996: Large scale Aurora/AFS Deployment Begins
- 1999: By year's end, 99% of the production environment runs on AFS

AFS/VMS at Morgan Stanley: Why it Succeeded

- MS Engineering/Technology Culture
  - Truly results and deliverable oriented
  - Global instead of regional teams
  - Pre-existing krb4 realm was essential
  - Started with 2 legacy NFS-based infrastructures that were already centrally managed, and managed fairly well

AFS/VMS at Morgan Stanley: Why it Succeeded

- AFS Features
  - Readonly Volume Replication, Ubik Redundancy
  - Scalability and Security
  - Centrally Manageable (vos, pts, fs, etc)
- VMS Functionality
  - Global /ms Namespace
  - Versionized Release Management, Software Distribution
  - Massively Automated Filesystem Infrastructure

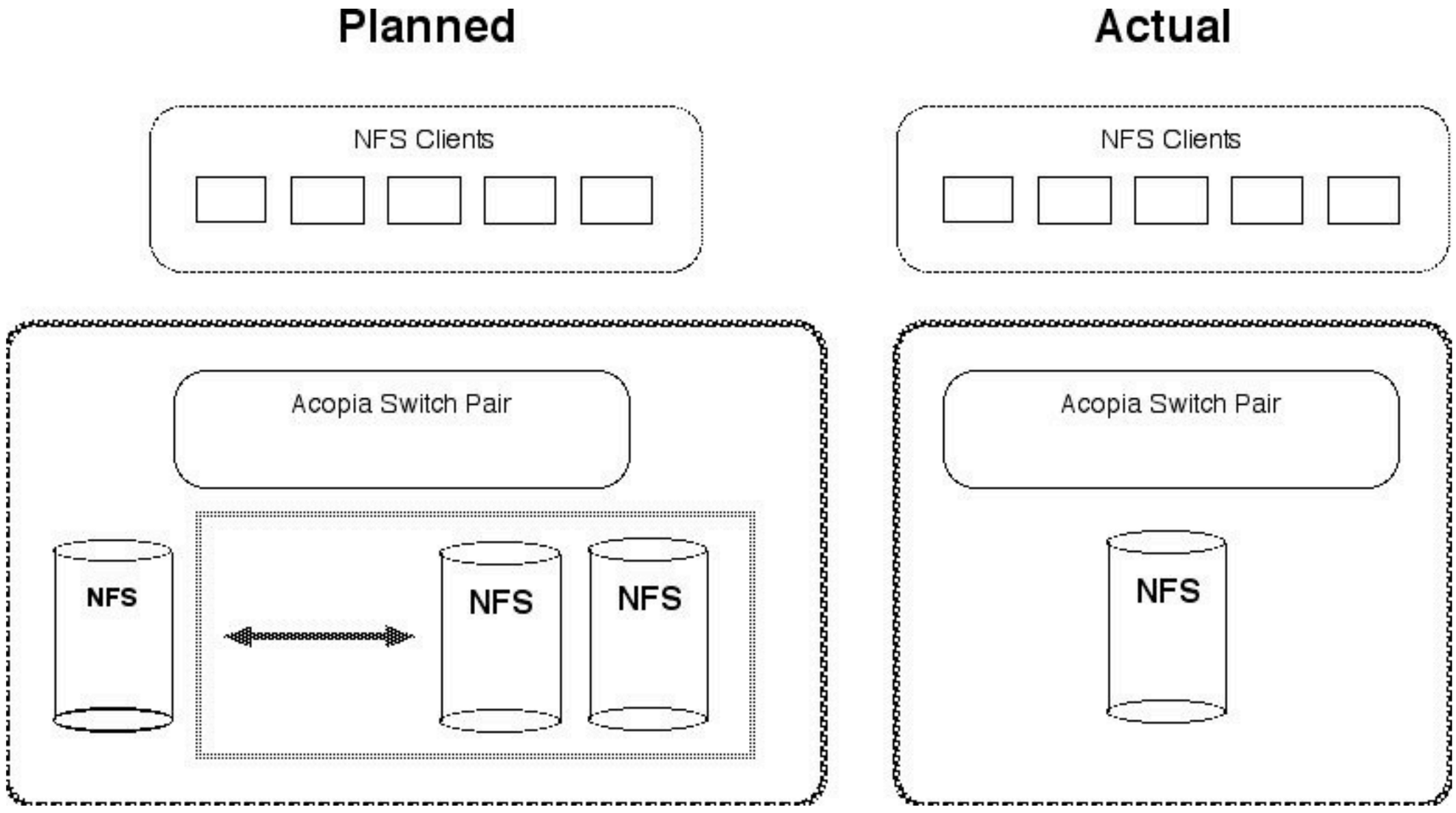
NFS/EFS at ALB:
System Architecture

- EFS was intended to be "VMS for NFS", reimplementing a similar namespace (/ms => /efs) and command line interface (vms => efs)
- AFS was never seriously considered
- ALB had near-zero client-side automation, and integrating and supporting the AFS cache manager would have been too complex
- ALB had a previous (very negative) experience with AFS, 10 years earlier. It was a four-letter word.

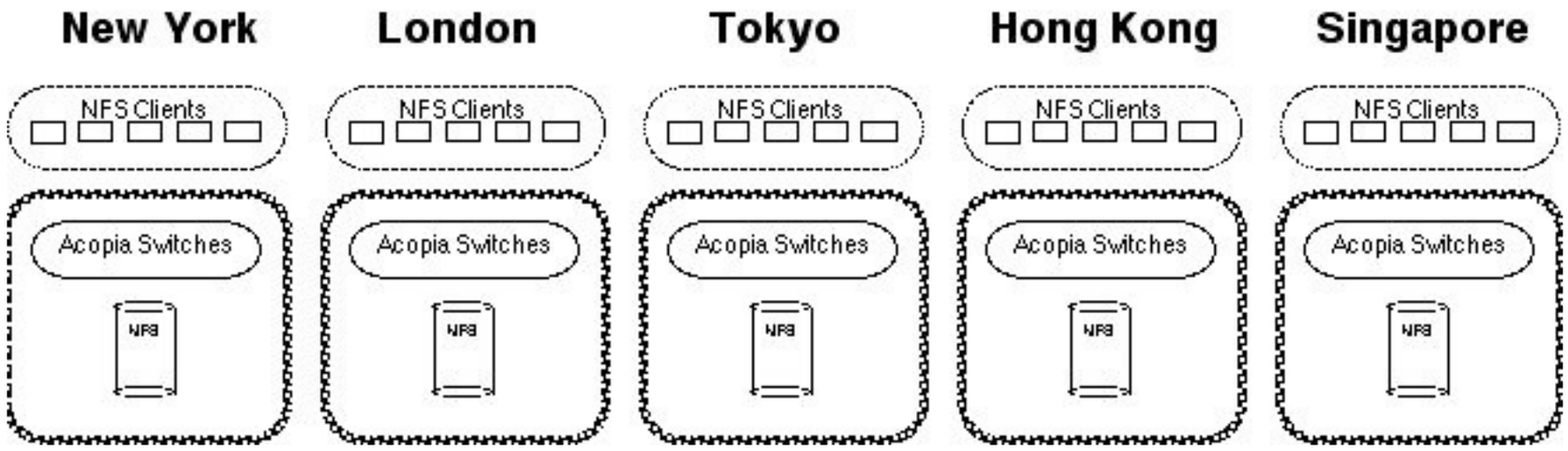
NFS/EFS at ALB:
System Architecture: EFSv1

- NFSv3 was chosen simply because the client was already globally available, and the NFS server was supported.
- EFSv1 implemented the global namespace using Acopia switches:
  - Very expensive, very proprietary
  - Very good when they work, impossible to understand when they break
  - Never truly leveraged their core functionality
    - Automated vos move for NFS
  - "amd on steroids on a proprietary hardware platform"

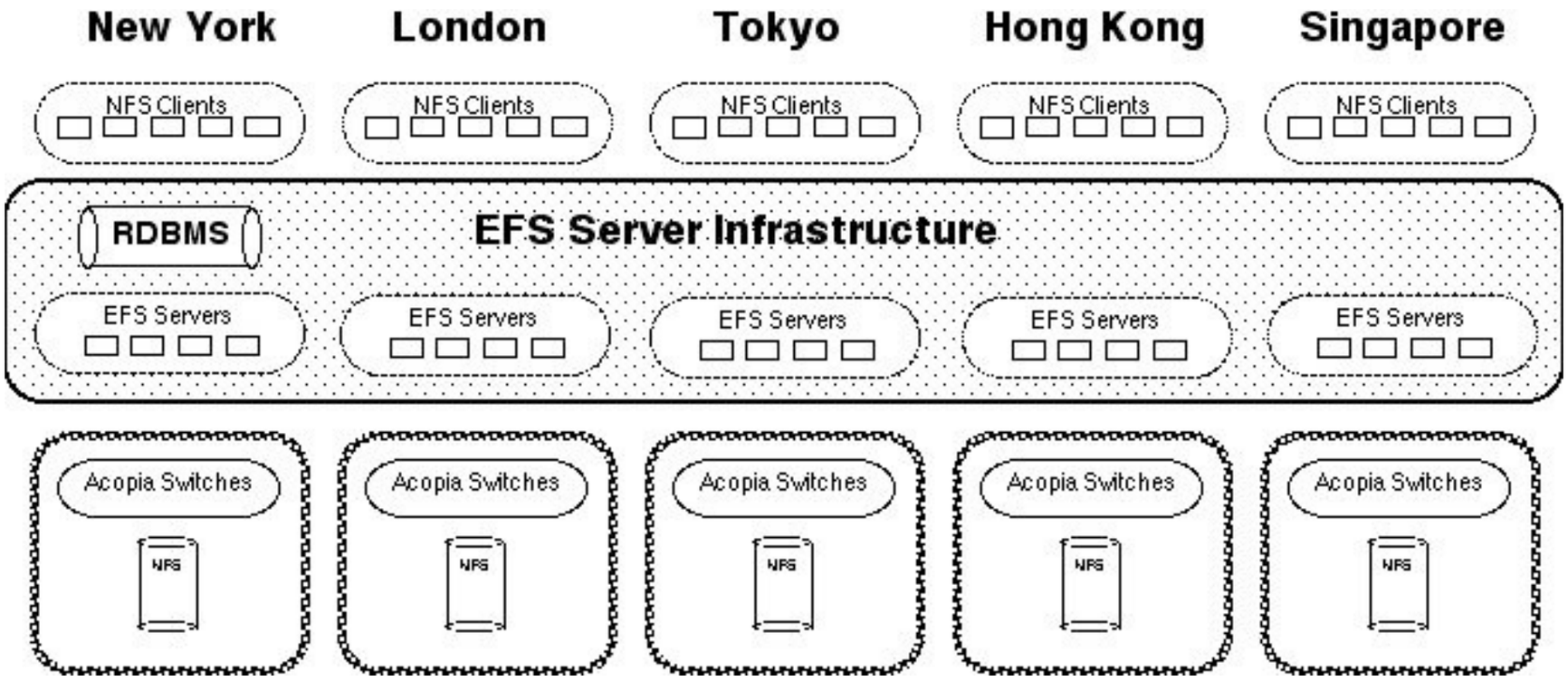
NFS/EFS at ALB:
EFSv1 Acopia-based Cells



NFS/EFS at ALB:
EFSv1 Acopia-based Cells



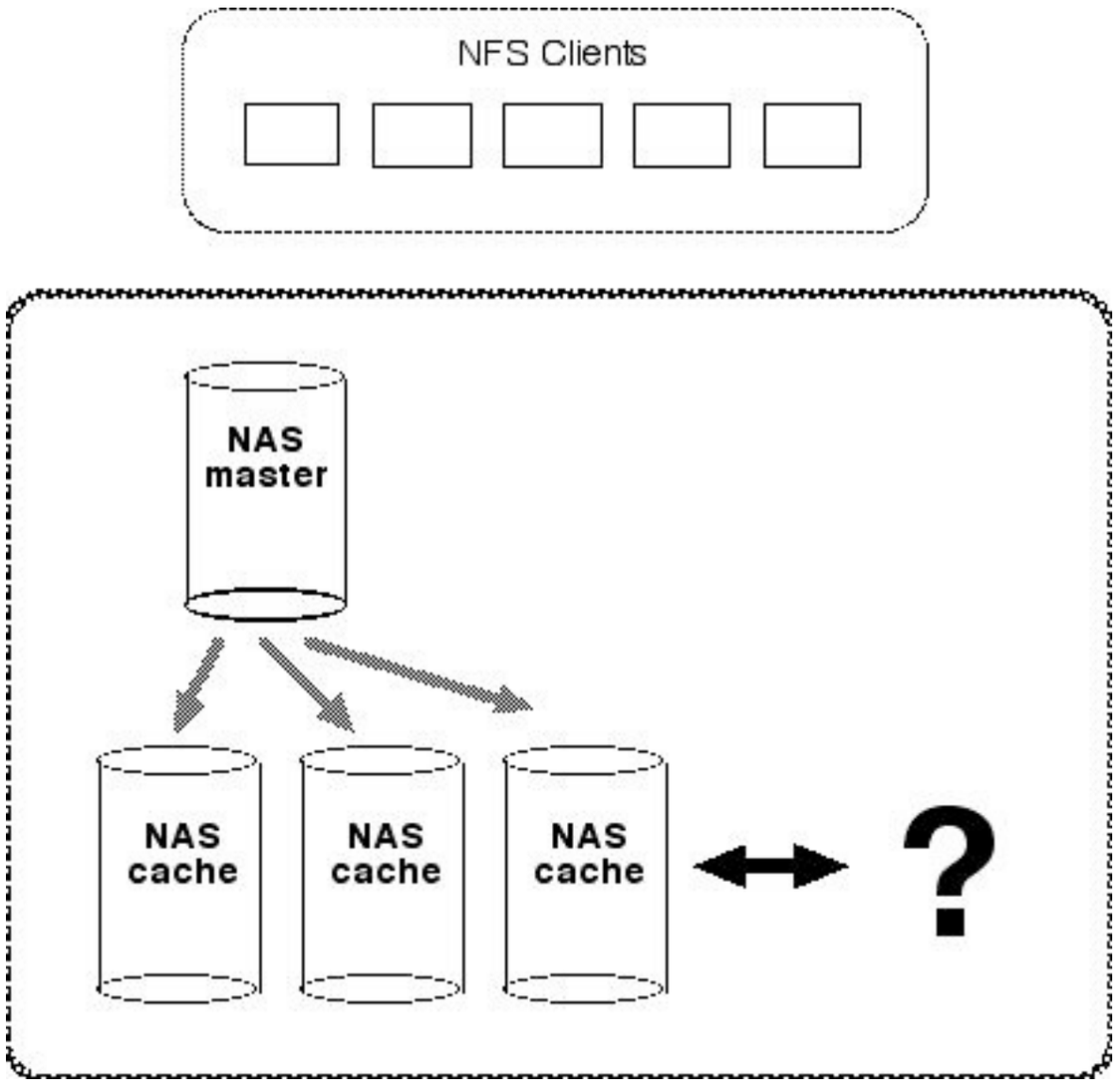
NFS/EFS at ALB:
EFSv1 Acopia-based Cells



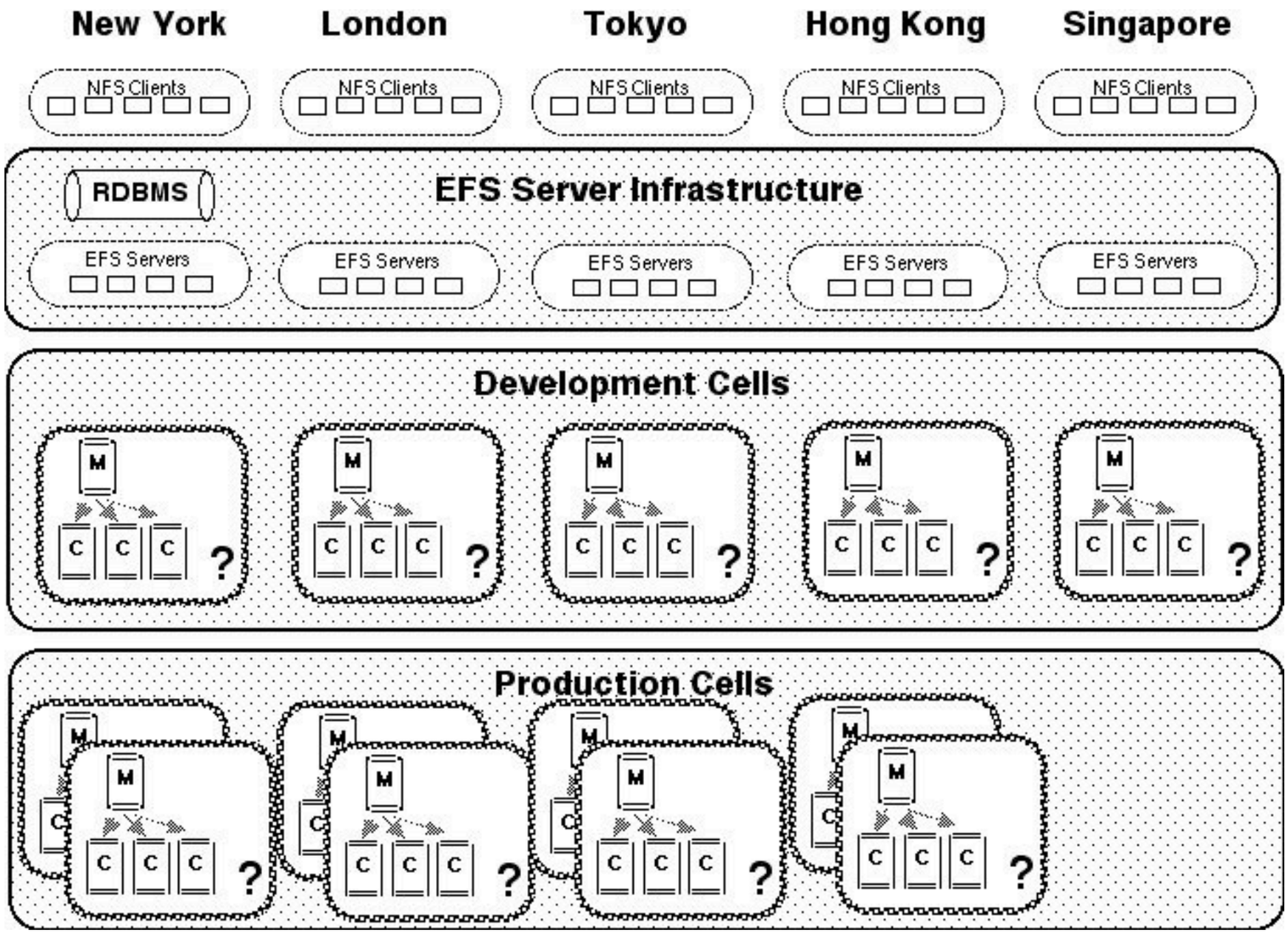
NFS/EFS at ALB:
System Architecture: EFSv2

- EFSv2 implemented the global RW namespace using automounter maps (first amd, then autofs)
  - Migrated from Acopia in less than 3 months
  - Centralized map management via /efs/dist
  - amd proved to be ***very*** unstable, autofs worked fine
- EFSv2 introduced separate dev and prod cells

NFS/EFS at ALB:
EFSv2 NetApp-based Cells



NFS/EFS at ALB:
EFSv2 NetApp-based Cells



NFS/EFS at ALB:
Did it Succeed? Yes

- Deployed in over 20 sites in 10 countries, and considered mission critical trading infrastructure
- Deployment of business applications can be managed by development teams with very little involvement by administrators
- Large amount of open source and vendor software deployed and supported via /efs (1000's of release of 100's of projects)
- Approved for release as an Open Source product

NFS/EFS at ALB:
Did it Succeed? No

- Fully embraced by limited number of development teams
- Not an integrated part of the standard UNIX builds
- Most deployment issues have been political, not technical
- Very difficult to deploy highly automated software in a culture of entirely manual operations
- Deploying vendor products in EFS is very difficult, and adoption by infrastructure teams has been very slow
- Yes, we're Open Source now, but who exactly is ALB again?

NFSv3 Issues

- Nearly impossible to secure
  - Limited to using "system" (in)security, due to complete lack of security infrastructure (no kerberos at all)
  - Requires aggressively managed and audited clients
  - Requires good naming service infrastructure
    - DNS, NIS/NIS+/LDAP
    - Unified UNIX user/group databases
  - UFS semantics for controlling access too primitive

NFSv3 Issues

- Lack of good tools for automation
  - Every hardware vendor has a proprietary interface
    - Acopia: cisco-like manual CLI, no remote management
    - NetApp: GUI based remote management, poorly designed admin CLI
      - Vendor had ZERO interest in NetApp.pm API
- Every operating system has a different NFS implementation
  - Linux, Solaris, FreeBSD, etc all implement /etc/exports, /etc/fstab, autofs, and NFS commands with slight differences.
  - No remote management for any of them

EFS 3: OpenEFS

- Open Source Implementation, based on EFSv2 namespace and CLI
- Apache-based License
- http://www.openefs.org
- Still pre-alpha, plan to be prod this fall

The /efs namespace

- Globally unique, readwrite directories:
  - /efs/home/p/pm/pmoore
  - /efs/dev/fsf/gcc
- Replicated, distributed directories:
  - /efs/dist/fsf/gcc

The /efs namespace

- Global readwrite namespace via autofs

```
/efs/dev/fsf/gcc ->      /.efsglobal/efs_dev_x_yyy/efs_qtree/dev/fsf/gcc
/efs/home/p/pm/pmoore -> /.efsglobal/efs_home_a_bbb/efs_qtree/home/p/pm/pmoore
```

- Map files are maintained in /efs/dist

```
In /etc/auto.master:
/.efsglobal file:/efs/dist/efs/config-autofsmaps/current/common/d.fh.nyc.us.boot.efs/global.map

In .../global.map:
efs_dev_d_fh   -rw,intr,tcp,vers=3   fhnfsd01.boot.efs:/vol/efs_dev_d_fh
efs_home_d_fh  -rw,intr,tcp,vers=3  fhnfsd01.boot.efs:/vol/efs_home_d_fh
```

- Entire set of autofs maps are auto-generated only when NFS volumes are added/removed from the environment globally

The /efs namespace

- Client-side NFS mounts, created dynamically by /etc/init.d/efsclient

```
mount $nfsserver:/vol/efs_dev_x_yyy/efs_qtree/global  /efs/dev
mount $nfsserver:/vol/efs_home_x_yyy/efs_qtree/global /efs/home
```

- Both /efs/dev and /efs/home are centrally managed symlink trees

```
/efs/dev/fsf/gcc -> /.efsglobal/efs_dev_x_yyy/efs_qtree/dev/fsf/gcc
/efs/dev/foo/bar -> /.efsglobal/efs_dev_a_bbb/efs_qtree/dev/foo/bar
etc...

/efs/home/p/pm/pmoore -> /.efsglobal/efs_home_x_yyy/efs_qtree/home/p/pm/pmoore
/efs/home/s/se/setuid -> /.efsglobal/efs_home_a_bbb/efs_qtree/home/s/se/setuid
etc...
```

The /efs namespace

- In VMS, 'exec' leveraged AFS' @sys for heterogenous transparency:

```
/ms/dist/foo/PROJ/bar/1.0/exec -> .exec/@sys
```

- In EFS, the same 'exec' functionality is provided:

```
/efs/dist/foo/bar/1.0/exec -> /.efs/dist/foo/bar/1.0/exec

mount $nfsserver:/vol/efs_dist/efs_qtree/$instance/dist /.efs/dist

/.efs/dist/foo/bar/1.0/exec -> /efs/dist/foo/bar/1.0/.exec/$instance
```

- Platform independent paths for platform specific files:

```
#!/efs/dist/perl5/core/5.10.1/exec/bin/perl
```

The efs CLI

- efs *action object arguments*
  - efs create metaproj $metaproj $user $group
  - efs create project $metaproj $project
  - efs create release $metaproj $project $release
  - efs create install $metaproj $project $release $install
  - efs define location $region $campus $location
  - efs define efsserver $cell $efsserver
  - efs define cell $region $campus $location $cell

EFS Software Distribution
Deploying globally to 'dev'

- Step One: create a release
  - efs create release foo bar 1.0
  - efs create install foo bar 1.0 $platform (repeat for multiple platforms)
- Step Two: populate the install tree
  - make install?? (YMMV)
- Step Three: checkpoint the release
  - efs checkpoint release foo bar 1.0
- Step Four: distribute the release to all dev cells, globally
  - efs dist release foo bar 1.0 --celltype dev

EFS Software Distribution
Deploying globally to 'prod'

- Step Five: promote the release to the prod stage
  - efs setstage release foo bar 1.0 prod
- Step Six: distribute the release to all prod cells, globally
  - efs dist release foo bar 1.0 --celltype prod
- Access can be managed per-metaproj, per-stage
  - Separate teams can manage the release during each distinct "stage" of the lifecycle. In the "dev" stage, the development team has complete access, but once they set the stage to "qa", then access can be restricted to a separate team. Likewise for "prod".

Other EFS Features

- Dependency Management
  - Inter-MPR dependencies, registered by users or build systems
  - Limited support for auto-discovery (RPATH in ELF files)
  - Can’t destroy something that has dependencies
  - Can’t setstage to prod if your dependencies aren’t prod

OpenEFS Security: The Good News

- Inter-server (efsd to efsd) Communications
  - Fully automated, standalone SSL infrastructure, with CRL support
  - All certificates can be rotated using "efs newcerts efsserver"
    - Generates new SSL certs, revokes the old ones, globally
- Client/server (efs to efsd) Communications
  - Privatized using SSL, but...
  - Password-based authentication via PAM only
  - Role-based authorization mechanism

OpenEFS Security:
The Good News, cont'd

- Inter-cell Software Distribution using rsync over ssh
  - All ssh keys are auto-generated, and can be rotated in realtime:
    - efs newkeys efsserver
  - Support for a non-standard ssh port (for network traffic management)
- Complete command history, with centrally managed logfiles
  - Complete forensics for problem solving and trouble shooting

OpenEFS Security: The Bad News

- No NFS Infrastructure Management at all
  - EFS can generate a "recommended" export syntax, but grouping mechanisms are not standardized and thus, export security is left up to the NAS/NFS administrators
  - NFS servers are 100% transparent to EFS
    - Just a hostname and a pathname
    - ZERO knowledge of NetApp flexcache servers
- Limited to plain old UFS semantics for controlling access
  - One uid, one gid :-(
  - ACLs are too non-standard to leverage in a heterogenous environment

OpenEFS Security:
The Bad News, cont'd

- No NFS Infrastructure Management is an OPPORTUNITY
  - Primary reason we don't have it is political. In 2008, we developed the NetApp.pm CPAN modules as the foundation for automating the NFS infrastructure. Politics killed the project, not technical issues. Virtually ***nothing*** at ALB is automated.
  - EFSv3 can be extended to manage the fileserver infrastructure (in fact, interim versions of EFSv3 had some basic NetApp-specific management features, but they were never used).
  - Plan to support NFSv4 and OpenAFS, *based on community input*
    - efs define nfsserver, efs create nfsshare, efs create nfsexport, etc...
  - ***NO*** plans to tie EFS to proprietary hardware, ever again

EFS Futures

- Security
  - Native krb5 support to replace SSL
  - Fileserver management for NFSv4/OpenAFS
  - ACL support for NFSv4/OpenAFS
- Platform Support
  - Currently only supports RHEL/CentOS/Fedora Linux variants
  - Plan to support Ubuntu, Debian, FreeBSD, Solaris clients
  - No plans for Windows support

EFS Futures

- Inter-domain Software Distribution
  - efs upload release: package a locked down release, upload via rsync/ssh
  - efs download release: download pre-packaged releases via http
- Web-based GUI
  - First change control system that actually controls change?
- Users
  - Might be kinda nice.....

Global Filesystems in the Real World: Comparing Experiences with AFS and NFS

Phillip Moore
http://www.openefs.org
http://www.yume.org/phil