Morgan Stanley

OpenAFS and the Dawn of a New Era

Alistair Ferguson, Vice President
Morgan Stanley Information Technology

Presented at the AFS and Kerberos Best Practices Workshop, 2008
New Jersey Institute of Technology

May 21, 2008

Dawn of a New Era

- Brief history of AFS at Morgan Stanley
- The dark ages ?
- Dawn of a new era ?
- Recently completed projects and their impact
- Concluding remarks

A Brief History of AFS at Morgan Stanley

- Before AFS
  - only regional NFS
  - different configurations used by different business units
  - developers in multiple regions would pass tarballs from region to region daily
- Aurora project (see "*Morgan Stanley's Aurora System: Designing a Next Generation Global Production Unix Environment*" presented at LISA '95)
  - sought to consolidate technologies across business units
  - identified the need for a global filesystem for:
    - redundancy and automated replication
    - global access to shared files
    - more efficient use of the network
    - better security

Why AFS ?

- AFS was chosen
  - Local disk cache
  - Guaranteed cache consistency
  - Logical volume management
  - Automated data replication
  - Transparently available redundant data
  - Superior performance over WAN links
- Constraints highlighted
  - UBIK protocol meant one cell / building instead of one global cell
  - No inter-cell data distribution
  - No byte-level locking
  - Backups
  - Lack of per-file permissions
  - Significant departure from UFS semantics

AFS vs. NFS/CIFS

	AFS	NFS/CIFS
Mount Points	One per client	One per filesystem
Hierarchy	Single global hierarchy	Local, requires auto-mounter / AMD
Caching	Consistent client-side caching	Minimal, usually implemented by 3rd-party products
Management	Online data migration	Offline data migration
Scalability	Highly scalable	Not scalable
Load Balancing	Automatic	None
Client fail-over (server failure)	Automatic	None
Performance	Multi-threaded server/client	Multi-threaded server/client
Security	Strong (Kerberos)	Advisory in NFS v3 and earlier (May be improved in NFS v4)

What does Morgan Stanley use AFS for ?

- Read/Write
  - Shared development areas
  - Application data storage
  - User home directories
- Read-Only
  - Operating systems
    - `/usr` is a symlink to AFS space
    - Minimal local footprint allows fast rebuilds and rapid change deployment
  - Application executables (binaries, libraries, scripts)
  - Configuration files
  - Data

The Good, The Bad, ...

- Advantages:
  - Consistent client-side caching
  - Excellent WAN performance
  - Read-only replication
  - Hierarchical namespace
  - Online volume management
  - Highly scalable
  - Security
  - `@sys` client-side platform abstraction

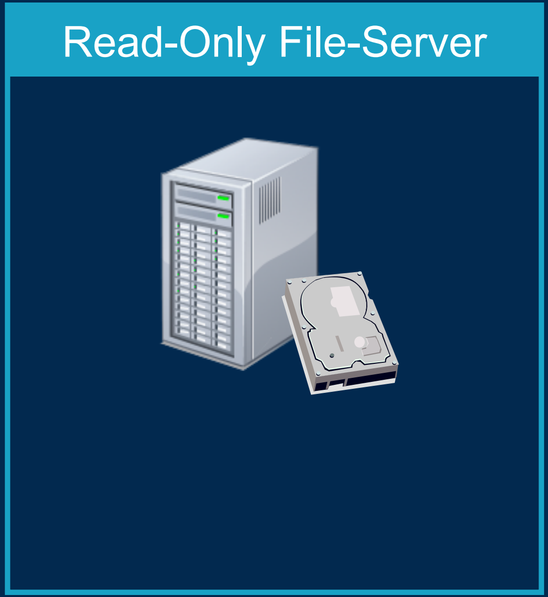
- Disadvantages:
  - Very complex
  - Read/write performance & stability concerns
    - `afs_global_lock`
  - Non-standard semantics
    - ACL's
    - Write-on-close
  - Lack of byte-level locking
  - VLDB doesn't scale

Replication vs. Distribution

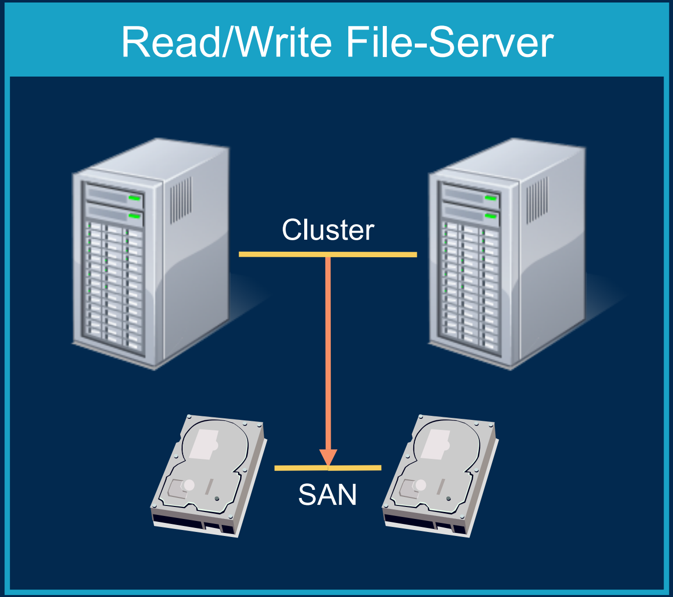
- AFS has built-in read-only replication, but…
- Limits on cell scalability
  - File-servers scale infinitely (maybe)
  - Database servers do not (UBIK protocol limitations)
- Application servers need reliable local access to AFS
  - Boundaries between cells determined by bandwidth and connectivity
  - Originally, one cell / building and branch offices had smaller cells
  - Now determined by the number of clients in that building and few branch office cells remain
  - No cell-to-cell failover; loss of a cell means loss of all clients in that cell

Oodles of Cells

- Read-only cell: 63 cells serving ~3.5TB unique data (10.5TB replicated)
  - > 430TB RO data globally
  - > 400 file-servers, each with around 1.6TB
- Read/write cells: 6 dedicated serving ~15TB
  - > 29 clusters, each with around 512MB
- Why separate them ?



Read-Only File-Server



Read/Write File-Server

The /ms Namespace

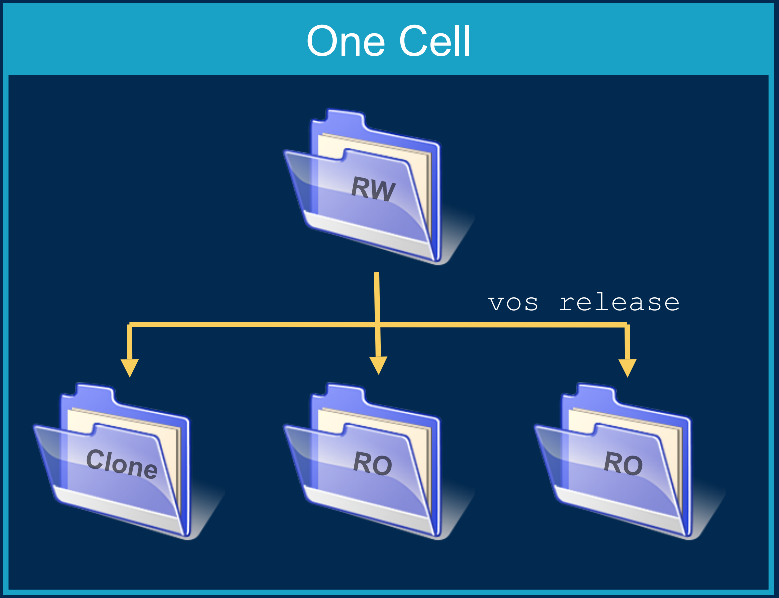
- One top-level AFS mount point (/ms)
- Traditional AFS namespace exposes individual cells; /ms hides them

Traditional AFS	MS Namespace
/afs/transarc.com	/ms/dev
ibm.com	dist
nasa.gov	group
uiuc.edu	user
...	.local
...	.global/ny.a
	ny.b
	...

- Read-only data (/ms/dist) served from local primary cell
  - Consistent paths can be used across the plant
  - @sys further helps mask differences between platforms
- Read/write data is shared globally (/ms/dev, /ms/group, /ms/user)

How Does It Get There?

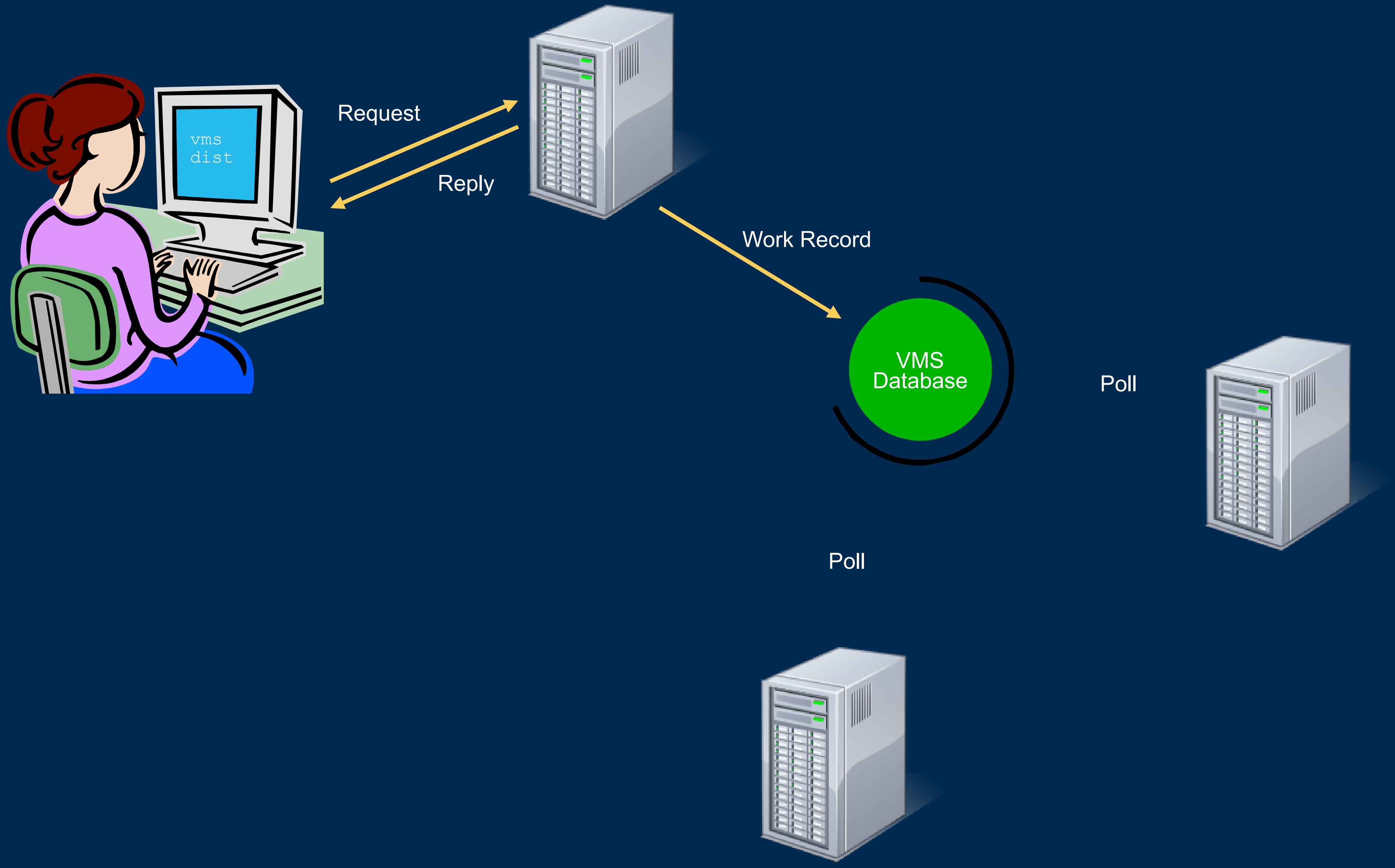
- Normal AFS replication:
  - `vos restore`, `vos addsite`, `vos release`
  - Requires admin rights
  - No built-in support for multi-cell distribution
- Volume Management System (VMS)
  - Volume-based distribution system written at Morgan Stanley
  - Client/server architecture with entitlements, logging and other enterprise features
  - Based on AFS `dump/restore/release` operations



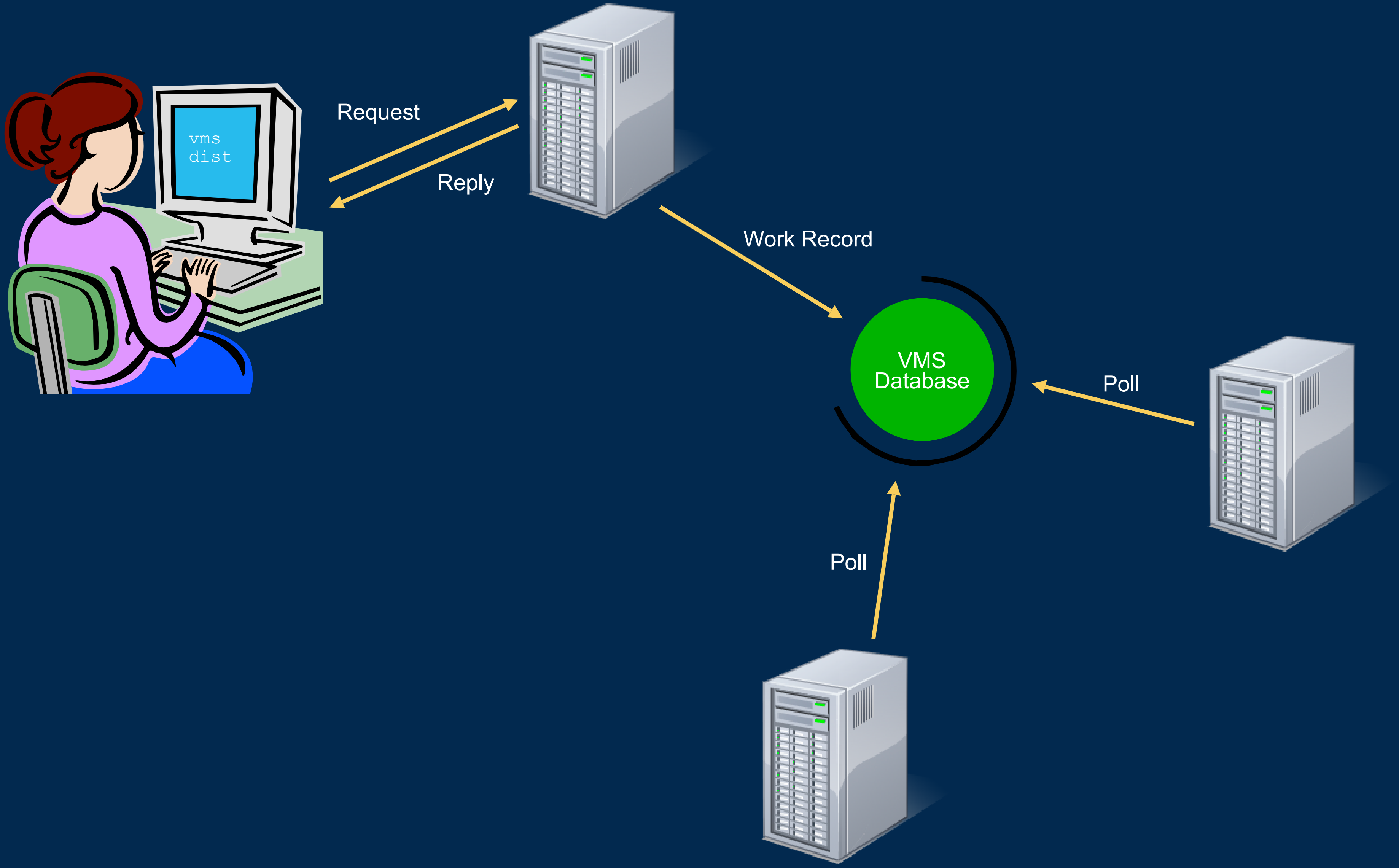
VMS, Under the Hood

- Automates the creation and management of the AFS namespace (i.e., AFS *volumes*)
- Client/server command-line syntax modeled after AFS utilities (`fs`, `pts`, etc.)
- Master data stored in database
  - Queries and reports done via regional read-only replicas
- Written entirely in Perl
  - POE framework, SOAP messaging between client & server
  - VMS Servers manage interaction with clients, write to-do's to database
  - VMS Queue Servers read work records from database, do the heavy lifting
    - Allows asynchronous operations, automatic retry, centralized monitoring, workload throttling
- Servers distributed globally

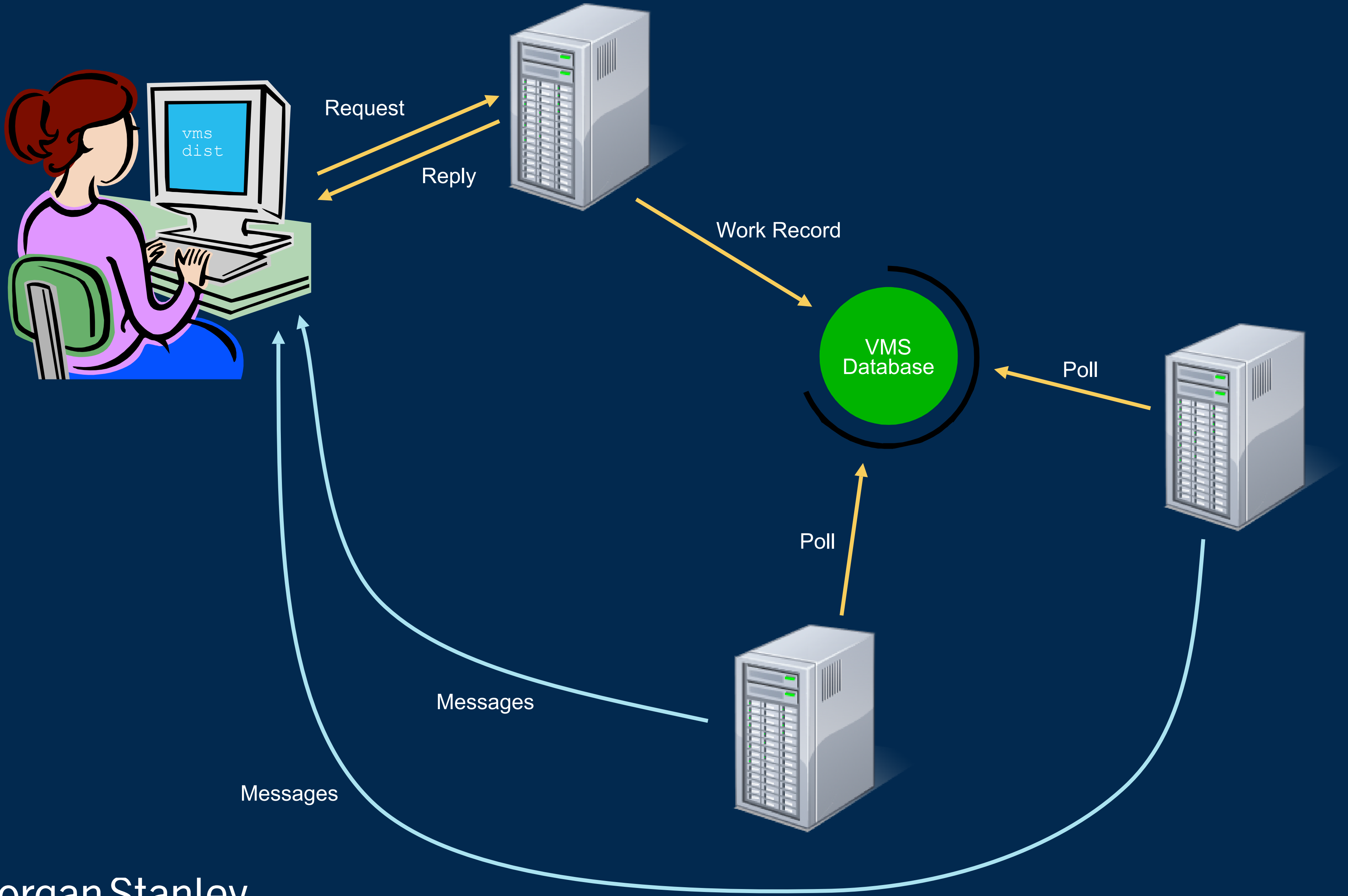
A Typical VMS Session



A Typical VMS Session



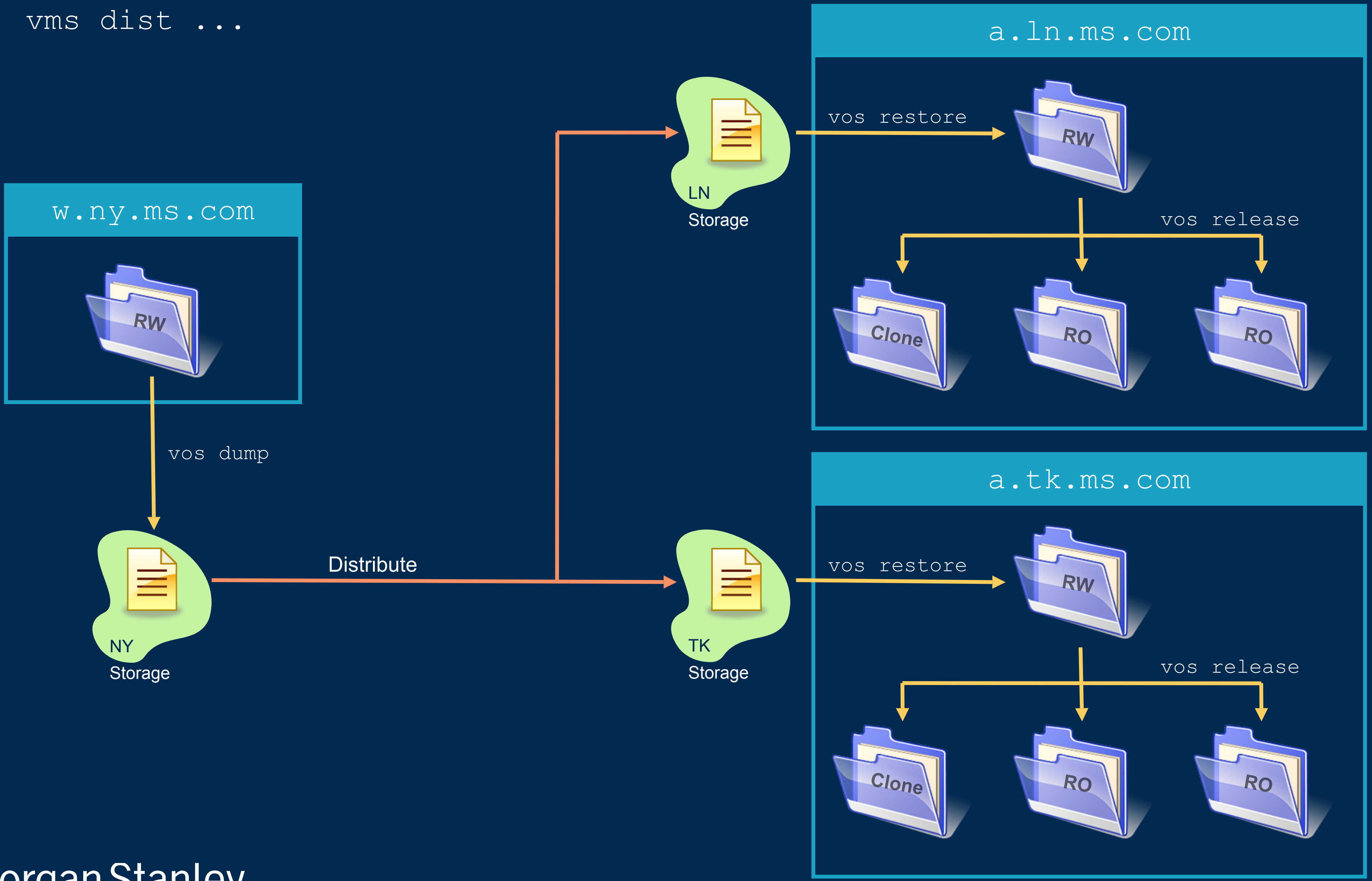
A Typical VMS Session



Morgan Stanley

VMS In Action

```
vms dist ...
```



Morgan Stanley

Problems - OpenAFS

- Transition from TransARC AFS to OpenAFS (feature freeze)
- Bugs in the OpenAFS 1.2 code-base → continual need to upgrade
- File-servers take along time to restart making upgrades difficult / painful
- Support for new platforms was slow
- Keepalives mean a single hung file-server can (eventually) hang all clients in a cell
- Call-backs are too coarse-grained, causing clients to re-fetch consistent data
- Cells "*dive off a cliff*" rather than degrade gracefully
- Clients can request the same volume look-up multiple times and hence increase VLDB load
- UBIK (and hence the vlserver and ptserver) are effectively single-threaded, which particularly affects the sync site
- Single mount-point (to address > 72-bit address space)

Problems - AFS at Morgan Stanley

- Move to Linux (RedHat AS 3)
  - OS less stable (than Solaris)
  - NAMEI file-server slower compared to inode fileserver (on-disk volume format sub-optimal for file-systems)
  - `ext3` file-system too slow, `vxfs` file-system was faster, but a (still unresolved) bug meant the file-server could hang
- OpenAFS 1.2 caused a lot of instability, particularly for read/write file-servers
- Takes > 2 hours to bounce a file-server, longer if the file-system needs checking
- Too many clients in some cells
- Legacy clients with small caches (512MB) / cache settings and old AFS releases (TransARC 3.5 onwards) cause higher file-server and VLDB load
- Bugs in the AFS client (Linux) meant cache settings remained lower than required
- Increased usage meant more volumes / files → wrapped around 32-bit address space → unpredictable inode clashes (affects libraries and Java)

Problems - VMS

- VMS was becoming slow
  - Slower Linux file-servers meant every volume operation took longer
  - Original volume hierarchy means container volumes are getting large (> 600MB in cases)
  - Non-incremental `vos release`
  - Entire directory structure present in incremental dumps
  - Coarse-grained locking (partly due to volume hierarchy)
  - Larger number of requests
  - Requests themselves are getting larger, due to more architectures, larger binaries, ...
  - More cells, so more work for VMS

➡ Larger number of requests in-transit

➡ Higher load on VMS servers

➡ Requests get slower, causing even more to be in-transit ...

and the Compute Model is Changing !

- More hosts: Aurora was designed for ten of thousands of hosts, now need to accommodate hundreds of thousands
- Larger scalability: 1.5K hosts / cell now (750 hosts / FS), need more than 10K (5K hosts / FS) to accommodate the increase without having six times as many cells
- Larger file-servers: 1.6TB (170K volumes) / FS now, need more like 8TB (850K volumes) as the amount of data increases
- Less downtime: 2.5 hours to upgrade a file-server is no longer feasible, > 650 file-servers globally means 1,625 hours / upgrade !
- More changes: < 100 changes / day a decade ago, now around 3K / day and growing
- Larger changes: 1GB changes used to be exceptional, now they are the norm
- Time critical: even large changes (> 1 GB) need to happen in minutes, not hours
- Client cache needs have increased: 512MB used to be enough, now > 4GB is required

It Wasn’t Any One Of These Problems ...

it was all of them !

Dawn of a New Era - OpenAFS

- OpenAFS 1.4
  - Code-base more stable
  - Bugs found and fixed in a shorter time-frame
  - Much better file-server performance
  - File-servers more immune to *bad* clients
  - Platform releases more timely
  - Easier / faster to get changes incorporated
    - Fewer upgrades required; only two release of OpenAFS 1.4 have been deployed !
- Increased interest ?
  - Traffic on openafs-info has tripled
  - This is the largest OpenAFS conference to date
- Where next ?

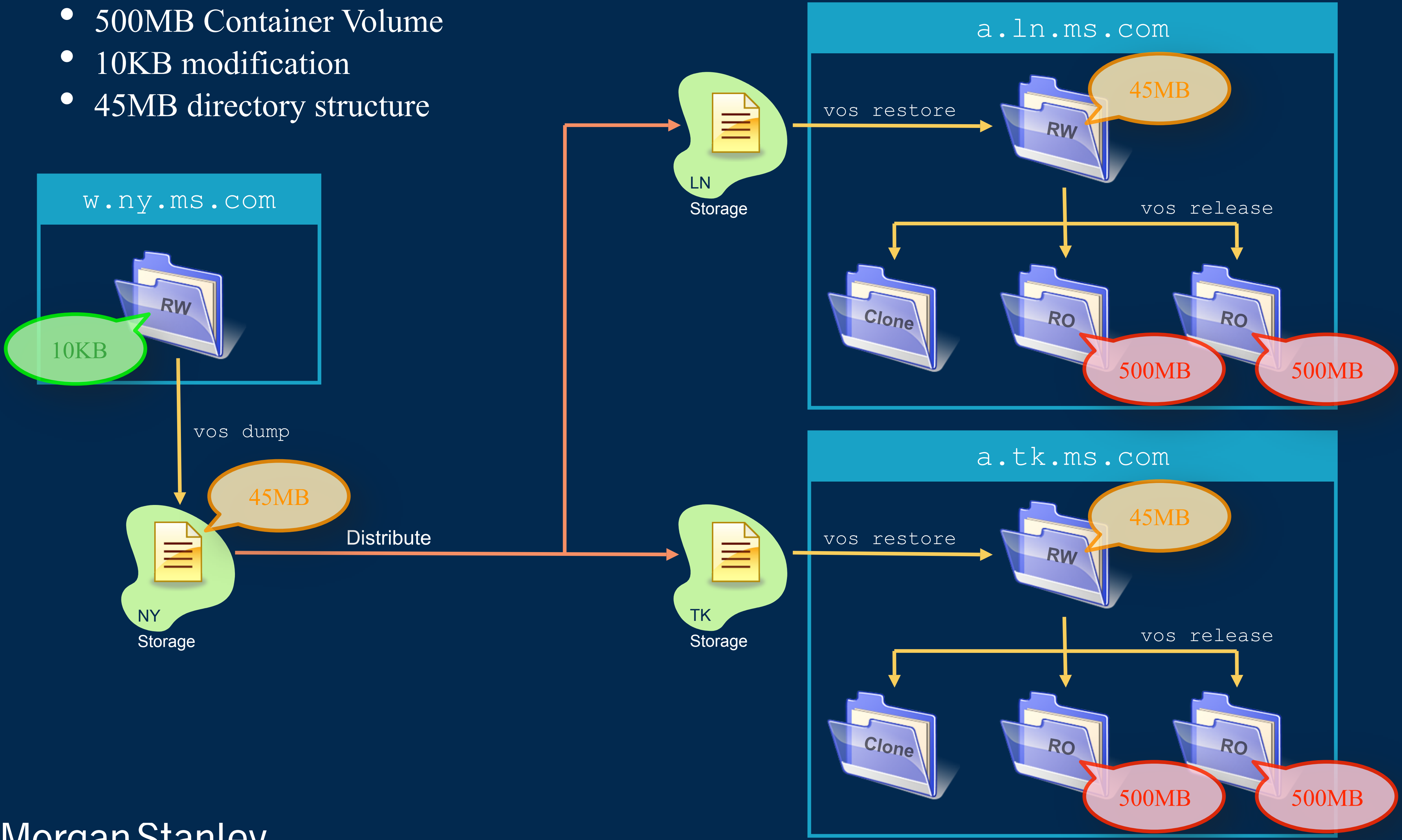
Dawn of a New Era - AFS at Morgan Stanley

- OpenAFS 1.4 deployment
- RedHat AS 4
  - More stable and better performance
  - Better `ext3` file-system performance (far superior to `vxfs`)
- Completed projects[1]
  - Incremental `vos release`
  - True incremental `vos dump`
  - Demand Attach File-server (DAFS)
  - Pthreaded UBIK (phase I): port from LWP to pthread
  - Keepalives
  - md5 inodes
- More projects underway

[1] Morgan Stanley contracts with an external vendor to enhance OpenAFS.
All work is contributed to open-source mainline.

Incremental Volume Dumps and Restores

- 500MB Container Volume
- 10KB modification
- 45MB directory structure



Morgan Stanley

Incremental `vos Release`

- Incremental `vos release` has a long history at Morgan Stanley
  - at least 6 attempts over a > 8 year period
  - exposed some of the most obscure bugs !
- VMS dist involves
  - dumping the canonical volume in the source (RW) cell
  - pushing the dump file to VMS server(s) local to the target (RO) cell(s)
  - restoring the incremental dump file into the RW copy of the volume
  - full `vos release`, which had come to dominate VMS dist times
- 500MB canonical volume:

	Non-incremental	Incremental	* Faster
`vos release` (seconds)	5,696.00	74.00	76.97
VMS dist times (max. seconds)	6,626.00	624.00	10.62
AFS file-server load (mean CPU)	6.83	4.87	1.40

True Incremental `vos dump`

- Incremental dumps include the entire directory structure
- VMS has to distribute the dump files globally
- Incremental releases don't include the directory structure, since AFS knows exactly when the copy was taken. True incremental `vos` dump exposes this feature.
- VMS knows exactly when a restore / dump was done (and verifies it)
- e.g. incremental dump for a 500MB *container* volume can exceed 45MB even for a 10K change ! VMS distributes 45MB, but AFS only releases 10K.
- 500MB volume with 45MB directory structure (all times are in seconds):

	Non-incremental	Incremental	* Faster
`vos dump`	24.87	0.52	47.83
Distribute	26.09	2.09	12.48
`vos restore`	4.30	1.35	3.19
VMS dist times (mean)	90.00	40.00	2.25

Morgan Stanley

Why Demand Attach ?

- AFS file-server with around 170K volumes and 1.6TB disk space takes approximately an hour to salvage, an hour to attach and an hour to shut-down
- Regardless of whether the restart was clean / unclean, a restart takes > 2 hours !
- Restarts can take even longer if a significant number of clients are down
- On a traditional AFS file-server, attached volumes need to be salvaged on unclean start. Volumes are always attached.
- All volumes need salvaging, regardless of whether they were in use or not
- Off-line volumes are not being used

Results of Demand Attach File-Server (DAFS)

- Demand Attach File-Server (DAFS) changes
  - Volume finite-state automata
  - Attach volumes on demand and hence salvage volumes only when required
  - Perform all I/O outside the global lock
  - Parallelize file-server shutdown process
  - Callbacks are not broken during shutdown
  - Host and callback states are saved on shutdown and restored on start-up
  - Volumes are garbage-collected (off-lined if not accessed)
  - Modified vnode package means read-only volumes are almost never salvaged
- > 90% Read-only file-servers across Morgan Stanley now run DAFS !

	Disk-space (MB)	Volumes	Non-DAFS	DAFS	* Faster
Read-only	1,752	170,000	7,200	4	1,800
Read/write	512	70,000	6,300	38	166

Pthreaded-UBIK

- UBIK is effectively single-threaded
- Sync site holds DB lock throughout writes and while propagated
- Single-thread can do nothing else while this is being done
- Performance of the sync site is dictated by the speed of the slowest VLDB server
- Sync site can easily become overloaded and hang clients (while the remaining VLDB servers are under-utilised)
- Cannot throw additional hardware at the problem !
- Unfortunately, it is difficult to

Why Stick With AFS ?

- Aggressive caching
- Guaranteed cache coherency
- User-perspective: when `vms dist` completes, all clients have the update !
- File-servers can handle far more clients than similar technologies, e.g. NFS/CIFS
- Online data migration, allows automated space balancing with no client impact
- Automatic load balancing
- Scalability
- DAFS means AFS file-servers restart as fast as NFS/CIFS !
- From Morgan Stanley’s perspective
  - Every server is the same; no local installs
  - Changes are all or nothing
  - When VMS returns success, it means everything

Challenges / What Next ?

- Eliminate the salvager
- No more salvaging
- Death to the salvager
- Automated test suite
- Volume-level FetchStatus (vFetch)
- Incremental DB propagation after quorum election
- Extended (finer-grained) call-backs so clients only need to re-fetch changed data
- Byte-level locking
- Revise time-outs, which were set when networks were slow and unreliable
- Better file-server performance
- RxTCP

Questions ?

Morgan Stanley