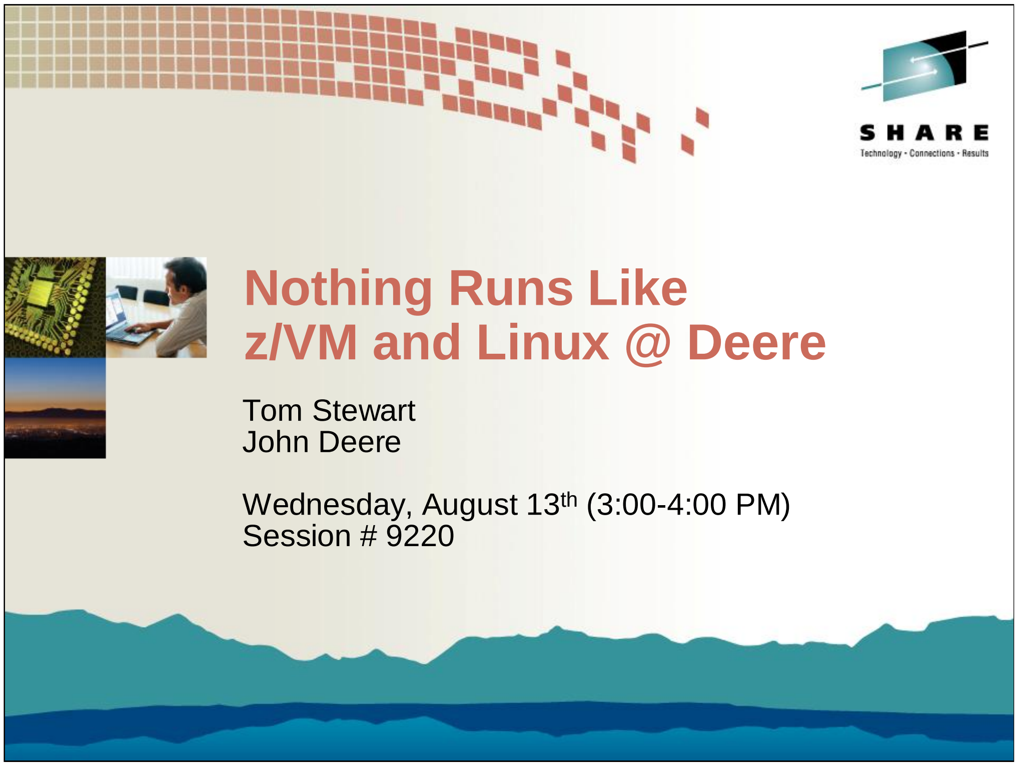


Title: 9220 - Nothing Runs Like z/VM and Linux at John Deere
Schedule: Wed 3:00 PM - Room C3 Concourse Level Convention Center

Abstract: The speaker will share some of the z/VM administrator tasks he's learned being new-ish to z/VM, covering things between where the cookbooks end and ongoing production support begins. For example:

- Managing multiple z/VM LPARs by sharing disk for config files and scripts
- Using your new best friend (Rexx) to automate tasks for Linux servers
- Quickly building Red Hat servers via kickstart files
- Creating a z/VM web admin interface with a free IBM download (lxcms) and PHP running on Linux
- Architecting for WebSphere Application Servers to run on Linux for System z, including disaster recovery planning

The intended audience is newer z/VM administrators running Linux workloads, especially those who don't yet have full commitment to purchase 3rd party provisioning and system management tools.

Speaker: Thomas Stewart
Company: John Deere
Address: 400 19th Street Moline IL 61265 USA
E-Mail: StewartThomasJ@JohnDeere.com

Disclaimer

- The material in this presentation is based on our experience at John Deere. This may or may not be indicative of what you will experience in your company.
- John Deere does not endorse any of the products or companies mentioned in this presentation.
- All products mentioned are trademark or copyright of their respective owners.

Our Run

- Stopwatch *(Timeline)*
- Runner's Log *(Details)*
- Marathon Planning *(Future)*

Session #9220

3

Today's journey will start with a timeline outlining the path of Linux on z/VM @ John Deere, discussing where we came from and where we are going.

Following that, we'll take a look back at some of the things we saw along the journey, documented in our "runner's log".

We'll wrap up with how we are planning to continue running into the future, hopefully for a lengthy "marathon".

Being Born Was Easy

- Internal Whitepaper
  - Researched by z/OS team
    - Customer Experiences, IBM resources
    - Magazines, Websites, SHARE presentations
  - Estimated Startup Costs
  - Targeted Workloads
  - Long Term Vision
- Proof of Concept
  - Communication Server for Linux (TN3270)
  - DB2 Connect Gateway

2005	2006	2007	2008

Being born is easy – someone else does all the work!

Our journey started with recognizing the value of virtualization on z hardware early on. The team that supports z/OS and related products did the initial research and created a whitepaper that was presented to management. It was compiled using information from other customers, magazines, SHARE presentations, and IBM resources. We identified the typical 'good fit' target workloads that we should investigate and created a long term vision of virtualization on System z within our company.

The research effort was rewarded with an opportunity to do a standard Proof of Concept. We chose Communication Server for Linux as a potential replacement for hardware that was approaching end-of-life, and DB2 Connect as a secondary test workload.

Standing Up Was Fairly Natural

- Hardware Configuration
  - Memory, IFL, OSA, disk
- Followed Cookbooks
  - Getting Started with Linux on System z
    - SC24-6096-02
  - z/VM and Linux on IBM System z
    - The Virtualization Cookbook for RHEL 4 (SG24-7272-00)
- Early Results
  - Worked as expected
  - Laundry list of follow up questions and decisions to make

2005	2006	2007	2008

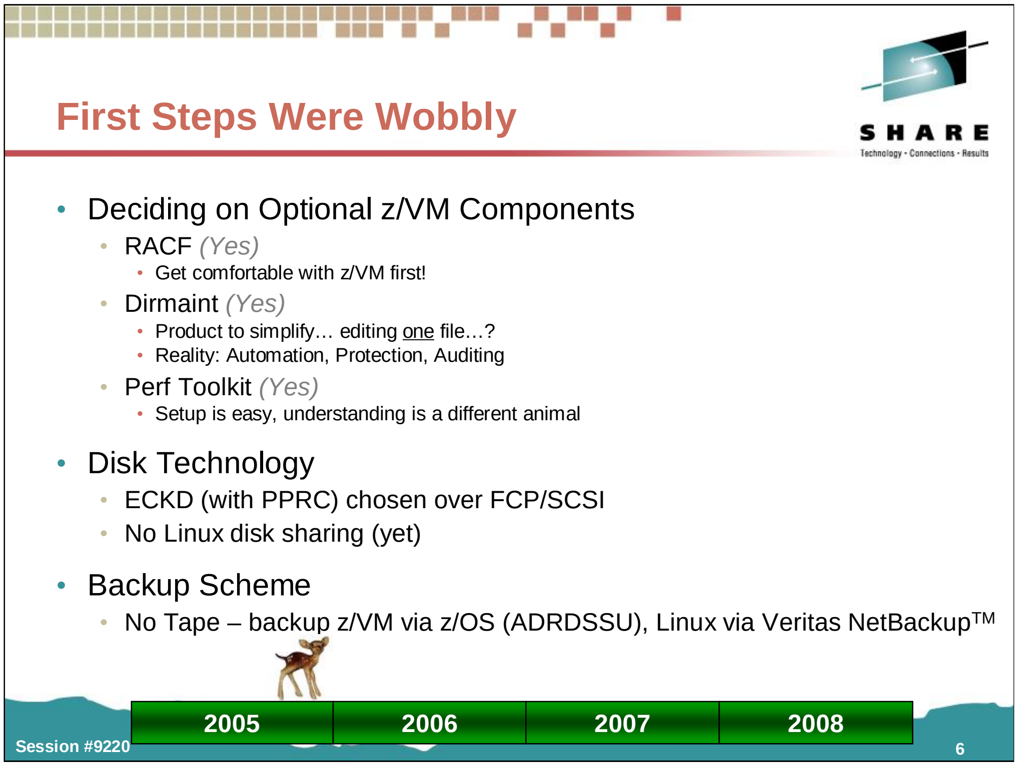
Session #9220 5

We carved out a couple LPARs and installed memory, IFL's, and OSA network cards. We squeezed into some existing ECKD disk ranges. This took some lead time because of the outages to perform some of these items, and there was a learning curve to do something different than z/OS on this hardware.

We chose the Red Hat Linux distribution early on because we had a fairly decent size installation of it already on x86 hardware.

z/VM installation was very easy using the available cookbooks, but it generated a large list of "how does that work" questions, along with a number of "we should test this and that" items. Linux installation was also simple and installed quickly.

Ended up with a few outstanding items that we'll look at in the next slide.



One of the early wobbles was making a determination on which optional z/VM components to install. As a z/OS shop, using an ESM like RACF was given – luckily the broad similarities made it a cinch to use. However, we recommend getting comfortable with z/VM before installing RACF.

Perusing around on the z/VM listservers gave mixed feelings about Dirmaint, but we installed it to try it out. At first, it appears bulky for the job it is doing, and we had our share of problems in the beginning where the configuration files got out of whack (until we got all of the RACF security set up correctly). In the end, DIRMAINT is useful software, particularly for automating user changes. Don't understand all the bells and whistles yet, but we know enough to automate what we need.

Performance Toolkit was a no brainer - the cost was low and we needed some type of monitoring tool. We did look at a few other products but decided to start simple until a need arose for something fancier. Favorite part is the web interface and extensive help.

For disk, we chose ECKD over SCSI because we already had PPRC licenses for the box we squeezed Linux into. We might re-evaluate when we start doing databases but for now we are solely ECKD. In another move to limit what we bit off to start, we also decided to dedicate disk to each Linux guest and not share any filesystems. We merge a handful of mod-9 volumes into a LVM group and then logically partition filesystems from there.

For backups, we take full volume dumps of z/VM disks and just use a distributed solution to backup Linux over the network.

We call these first steps wobbly because there was a balancing act between keeping things simple regarding disk/backups yet adding complexity with the extra z/VM components.

Growing Stronger Legs

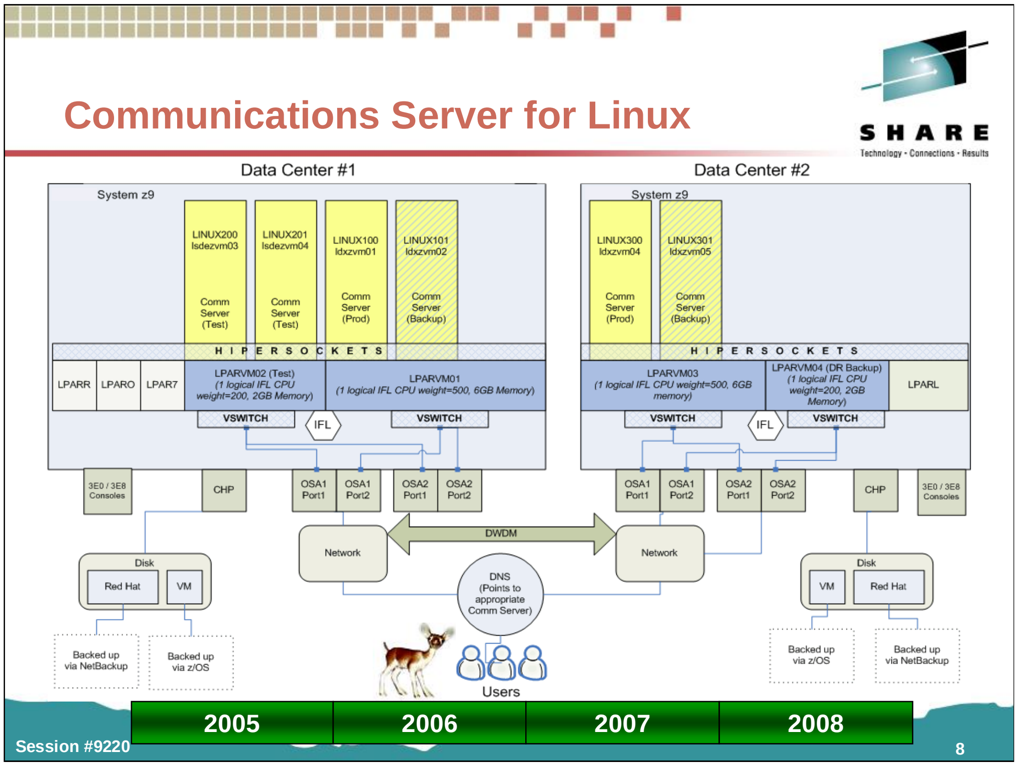
- Proof of Concept Wrap-Up
  - Conclusions Whitepaper
  - Approved for Communications Server for Linux
  - Hold for DB2 Connect
  - Approved for mainframe SFTP gateway

2005	2006	2007	2008

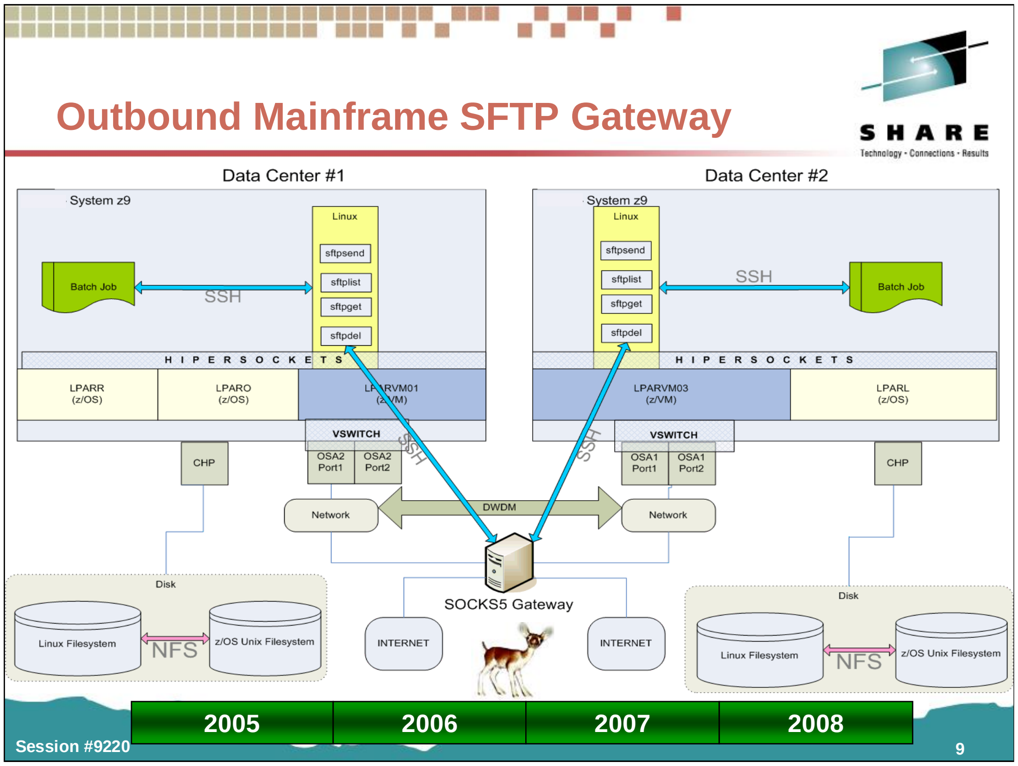
Session #9220 7

At the conclusion of our Proof of Concept, we got the green light to move forward with a production implementation of Comm Server for Linux. We held moving DB2 Connect until CSL was proven in production so we wouldn't get in a situation where any growing pains affected more than a single project.

During the CSL implementation, we also added a side project to see how easily we could put up a small special purpose or multi-purpose server. It addressed a need for a new SFTP gateway to get from the mainframe to the Internet.



This is a diagram of our Communication Server for Linux configuration. We started by splitting the load between two CSL servers at different data centers. DNS points the users to the correct side. Failover is available using a hot standby server at each site that can assume the workload of the dead server via some DNS updates.

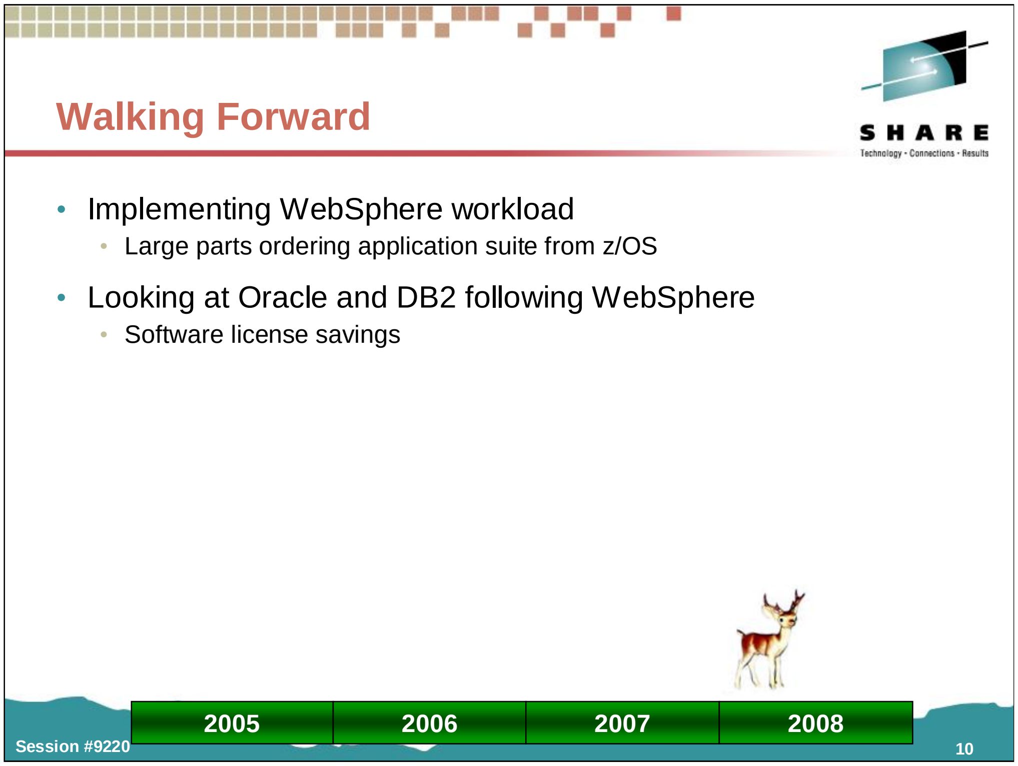


This diagram shows our mainframe SFTP gateway setup. It was for a specific business requirement, but was also used to verify NFS and Hipersocket connectivity between zLinux and z/OS.

Other items of note are how processing on zLinux can be driven using a z/OS batch job from scheduling software that runs SSH as a job step. In this case we are doing an SFTP passthrough because the OpenSSH port on z/OS couldn’t navigate our SOCKS gateway.

To enable SSH between z/OS and zLinux in batch mode, we set up keys on both sides to the application ID’s don’t get a password prompt. Perl scripts in Linux wrapper the SFTP processing that goes to the Internet, using the Expect module on zLinux to drive the SFTP in batch mode.

Just to note: there are products available to do this same functionality, whether it be SFTP straight from z/OS or remote script calls from z/OS to z/Linux, but this isn’t real difficult to implement with homegrown code.



You'll notice our Deere took a big leap here over 2007. About a year was spent migrating TN3270 users from the old solution to the new CSL solution, working out bugs along the way that were causing a number of Linux server crashes during peak loads. New Linux projects were put on hold until all of the issues were overcome. None of the problems were related directly to Linux or z/VM, it was mainly a load related issue with an open-source package used by CSL.

Our current project is to move a large WebSphere application suite from z/OS to z/Linux for support and software savings. We have completed testing and are working through the implementation schedule.

Also on tap in the next year is our Oracle and DB2 database systems due to the potential license savings. This is where we really expect to grow the environment to a much larger size.

Lastly, you'll notice that our Deere never got up to a real high speed on our run thus far, he's been a little distracted by other things to get moving too fast!

Our Run

- Stopwatch *(Timeline)*
- Runner's Log *(Details)*
  - z/VM Admin
  - Linux Admin
- Marathon Planning *(Future)*

During this next section we'll look at some of the things we have seen and learned during our run to this point, mostly items pertaining to z/VM and Linux administration we did to set up our environment for easier management in lieu of purchasing any provisioning tools.

Reading the Road Signs

- First things to do “post cookbook”
- CMS Primer & Users Guide, Virtual Machine Operation
  - http://www.vm.ibm.com/library/
- CMS/CP Commands and Utilities manuals
  - http://www.vm.ibm.com/library/
- IBM Downloads
  - http://www.vm.ibm.com/download/
  - Get and learn VMARC
- Rexx and CMS Pipelines
  - Numerous manuals and tutorials
- Mailing lists
  - http://www.vm.ibm.com/techinfo/listserv.html

Before just taking off at a full sprint, we took a little time to read the road signs to make sure we were on course. This slide references some of those materials.

Shortly after installing z/VM and Linux using available cookbooks, you should spend some time on the items listed here. To make your life easier, get familiar with Rexx and CMS pipelines – those are your scripting tools in this environment. Check out IBM’s developer downloads for some neat examples and also keep updated by learning from others on the available listservers.

Watching For Traffic

- Accounting
  - Accounting and Monitoring for z/VM Linux guest machines
    - REDP-3818-00
  - DISKACNT
    - Collects *ACCOUNT records to 191 disk - enlarged over default
    - Periodically bounced to purge old records – keep 30 days
    - 191 disk linked by ACCSRV userid
  - ACCSRV
    - FTP *ACCOUNT records to z/OS via nightly batch job
    - Rolled into accounting processes and archived in GDG's
- Monitor Data
  - MONWRITE
    - Set up and kept 7 days but never used for anything yet

The next couple slides cover some of the items we learned about from our reading, detailing what we chose to set up for getting z/VM into a more production ready state.

We enlarged the DISKACNT.191 disk to 50 cylinders to hold 30 days of accounting records. By default when we installed it was 1 cylinder.

We FTP our accounting records over to z/OS for long term storage in GDG's and for reporting and planning using our existing processes.

We do collect some monitor data with a MONWRITE service machine, but haven't had an occasion to use it yet.

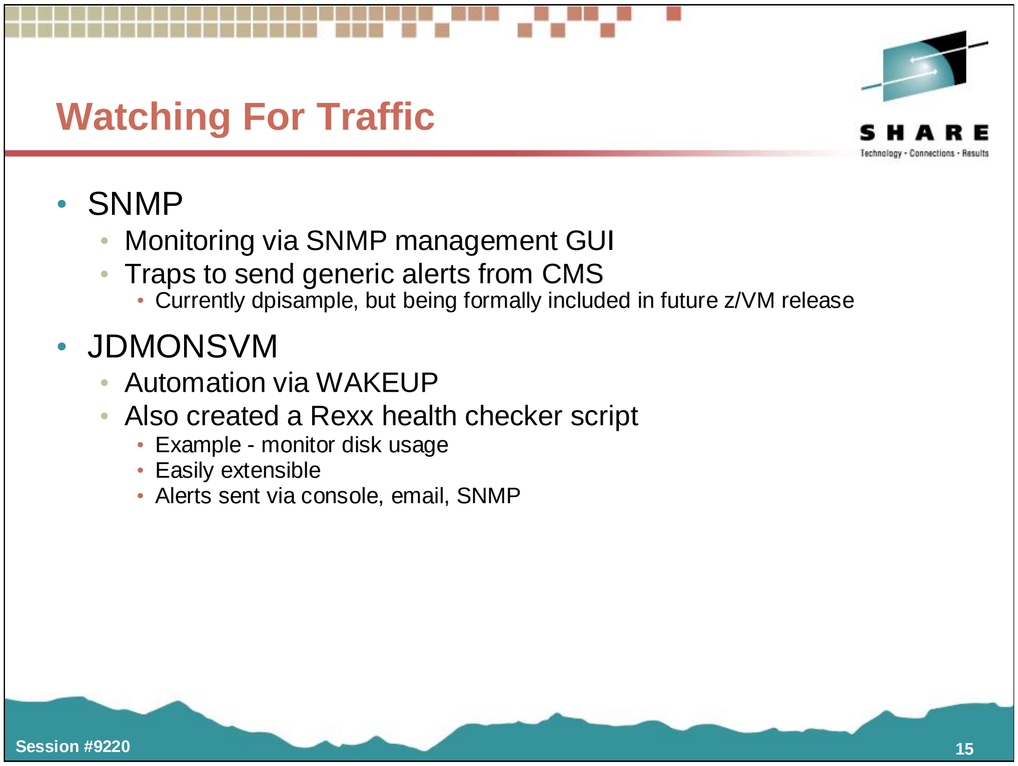
Watching For Traffic

- Cleanup routines
  - SFPURGER to delete old spool files
- PROP (Programmable Operator)
  - Setup per SHARE presentations and manuals
  - Linux guests SET SECUSER OPERATOR
    - Console logs go to PROP
  - Send it commands from scripts
    - So caller doesn't need authorization
    - Example - ATTACH/DETACH SYSASCII
  - Script to delete logs > nn days
  - Script to browse logs by nn days ago
- AUDITOR
  - Monitors/restarts various service machines
  - Found useful for quickly recovering hung Linux servers

Once we got more comfortable with spool interaction, we noticed a buildup of files out on the system. Found SFPURGER in the *CMS Commands and Utilities* manual that cleans up old files. We run this daily out of our JDMONSVM to handle cleanup based on information in an SFPURGER CONTROL file.

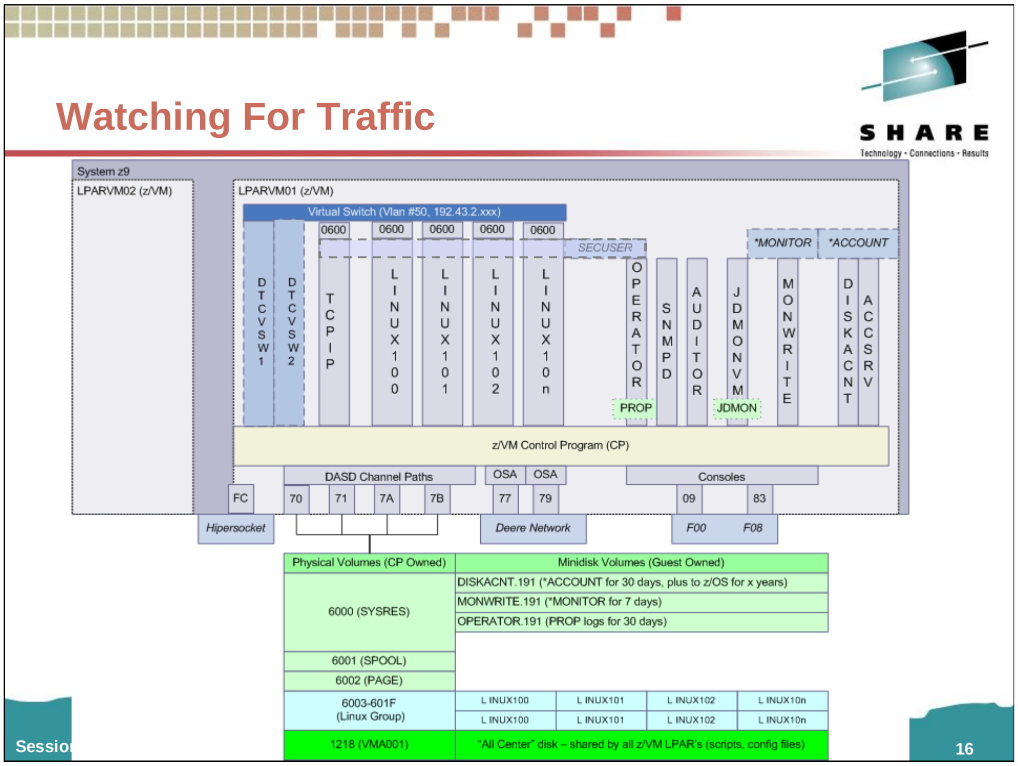
PROP was set up primarily to get some form of centralized logging, similar to what z/OS has. We also found it useful for routing script commands through it instead of granting more cmd authority to some userids. We set up some scripts for managing and viewing the PROP logs.

AUDITOR was one of those surprise finds that wasn't really in any of the initial cookbooks we followed, but fit the bill as a piece of our current low budget monitoring setup. It has actually proven quite useful in rebooting Linux guests that hang.



One of our monitoring decisions was how to externalize alerts from z/VM into one of our existing z/OS or network monitoring tools. We chose SNMP as a standard possibility but ran into some difficulties being able to send generic traps with message text of our choosing. After working with IBM, we were able to build a REXX exec that could trigger an SNMP trap to our tooling.

For automation, we got turned onto the WAKEUP utility and how it can be used inside a service machine to drive tasks. We set up our own userid to drive our automation, which included some homegrown health checking, such as fullness of some critical minidisks. Once this structure was put in place it was easy to add new scripts, and gave us a fairly standard environment because it works similar to our z/OS automation.



This shows a logical representation of how some of our components just discussed connect together.

This graphic also shows something that leads us into our next topic: sharing the road. You'll notice the VMA001 disk volume at the bottom. This is accessible by all of our z/VM LPARs to easily share data between them.

Sharing The Sidewalk

SHARE
Technology · Connections · Results

- Dedicated "All Center" disk volume
  - Write access from one LPAR only
  - Master copies of all customized config files, scripts
  - Backup to local z/VM disk in case access is lost
  - Backup to z/OS tape weekly
- SYSTEM CONFIG shared across LPARs
  - System_Identifier
  - User_Volume_Include
  - Devices Accepted & Not_Accepted
- DIRMAINT
  - EXTENT CONTROL
  - Unique volume groups for each LPAR
- Shared RACF Database
  - Wish list - coming soon
  - May synchronize instead

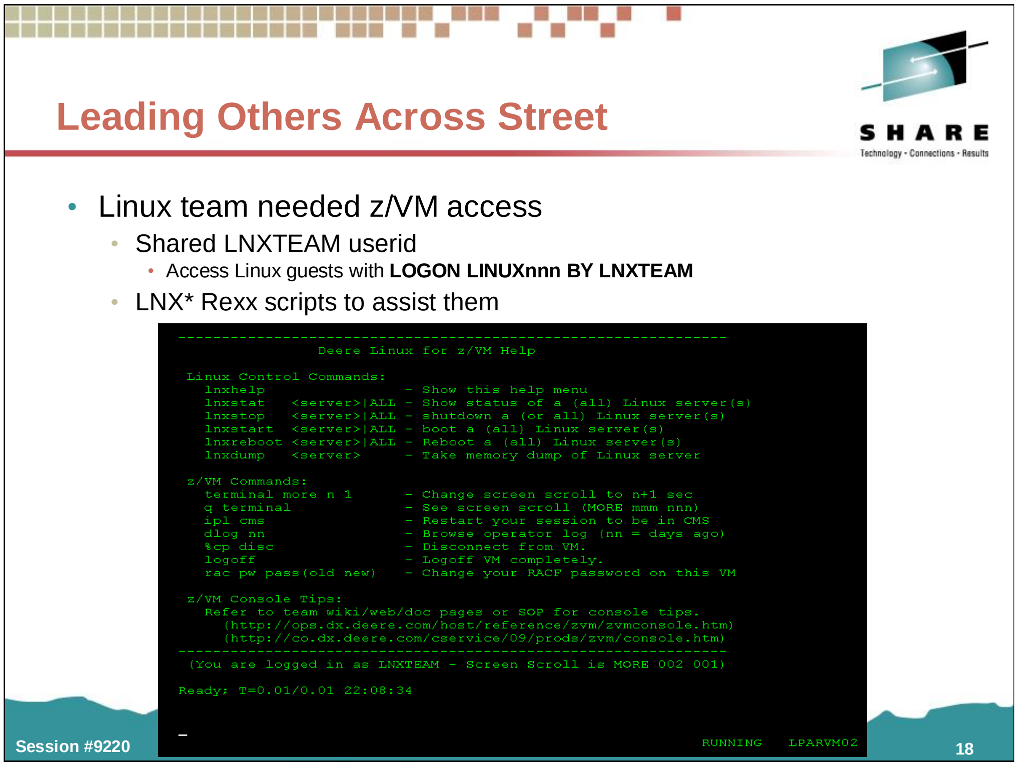
Session #9220 17

We took the idea of this “all center” volume from our z/OS environment where we share a small pool of disks between a number of sysplexes. To protect from corruption, we allow write access from only one z/VM LPAR which we designated as our configuration system, so we drive most of our z/VM administration from that LPAR. The disk houses the source copies for all of our config files and scripts. We do whole disk backups of it from z/OS weekly, but also added file backup to a local minidisk on each z/VM LPAR. We this offered a much faster recovery because we just point the owning userid’s minidisk to the backup minidisk and we are ready to rock.

Again pulling from our z/OS experience, where we really like sharing, we were able to come up with a single SYSTEM CONFIG that can be used by all of our z/VM LPARs using the statements listed.

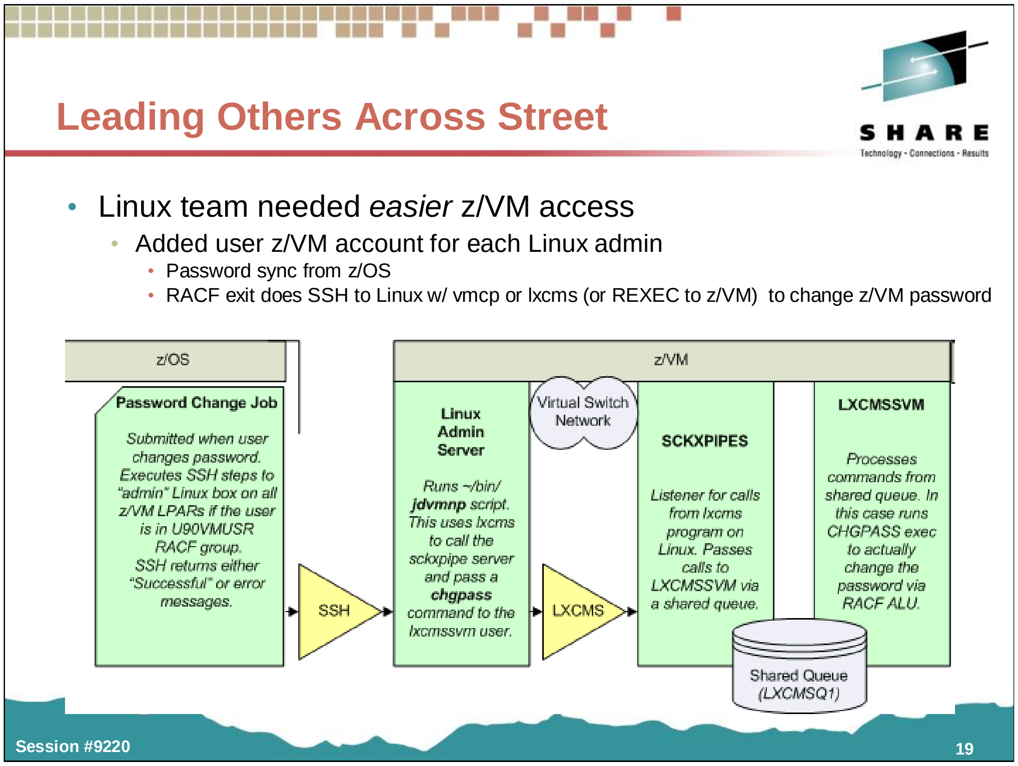
DIRMAINT keeps so many of it’s own internal control files that it isn’t really straightforward at sharing it’s configuration across LPARs, however we recently got to using a single EXTENT CONTROL file. Each LPAR has it’s own volume group that we allocate from. This is a recent test and we haven’t decided if there are enough ramifications to revert back. For example the dirm dirmap command output must be paired down to ignore the disk from the other systems.

Eventually we hope to share our RACF database between some of our LPARs, but haven’t gotten to that point yet.



Reading all the manuals for CMS and CP helped a lot, but nothing was as good as getting our hands dirty. One of our tasks was to get the Linux guys and operators access into z/VM to be functional when administering Linux. For this we created a few scripts to control our Linux guests - like status/reboot/dump. This taught us a ton about interacting between z/VM users, Rexx, and Pipelines.

This slide shows our help menu that displays when the Linux guys or operators log into z/VM.



One of the more challenging things we did to get the Linux guys easier access to z/VM was to give them individual logons so they didn't have to remember the password for LNXTEAM. Because we have a centralized credential store, we also set up password synchronization driven from z/OS. The diagram shows the process we set up to handle this. It works very well to keep human userids synchronized between our z/VM LPARs.

Leading Others Across Street

SHARE
Technology · Connections · Results

- Linux team needed *more* z/VM access
  - Configured HMC ASCII console
  - Tested and documented booting into single user / rescue mode
    - Proved useful during a couple 2am phone calls
  - BLDLNX EXEC to drive Linux install
    - Kickstart each Linux instead of cloning (matched distributed processes)
    - Works with existing dynamic web page to build Red Hat kickstart file
    - Creates internal config files for reporting, monitoring, etc.
    - Defines user to z/VM
    - RACF setup
    - Creates PARM and CONF files
    - IPLs to initiate kickstart process

Session #9220 20

We further simplified things for the Linux team by setting up the HMC ASCII console. This took a bit of work as our Red Hat version didn't play well with our HMC. In the end we got it working with IBM's assistance and a workaround.

We also tested and documented bootup problems like getting into single user mode and rescue mode, or linking a Linux system's disks underneath another guest for file recovery. Practice this so you have it documented when a real problem arises.

Although we haven't grown very large we've spent a bit of time testing out a process to build config files that drive scripts to define userids for Linux guests and then build kickstart files that can be used with a Red Hat install. Cloning might be used at some point, likely after upgrading to a 3rd party process. But for us there was value in building a simple mechanism ourselves to understand some of the steps involved in automating a build process.

Tripping Up

- Hardware config changes from z/OS
  - Linux isn't as forgiving as z/VM
    - Vary paths/chpids off from Linux before removing from IOCDS *(obvious in hindsight)*
  - Get familiar with I/O config commands
    - Controlling paths, devices, chpids, subchannels, etc.
    - On both z/VM and Linux
  - "Hey, move to a new disk array!"
    - Build procedure for moving Linux to new disk or between z/VM systems
  - Monitor to verify device status and pathing
    - Part of our JDMONSVM package

Session #9220 21

Now to share a couple lessons learned (trip-ups) from our experiences.

We do IOCDS changes from z/OS and got hit a couple times early on trying to make dynamic changes. Make sure that you take paths offline to Linux when changing them around with a dynamic IOCDS.

Get familiar with the various I/O config commands to be on top of things when asked to do cabling changes on short notice. This includes controlling paths, devices, chpids, subchannels, etc. Some kind of monitor to verify all of your devices are configured correctly and in the right status is recommended.

There are a number of tools/utilities for copying the disks around so practice this as well before you need to do it quickly.

Tripping Up

- Use a utility to maintain/version SYSTEM CONFIG updates
- Need something to handle DST or run GMT (for z/VM)
  - Linux guests all use NTP to z/OS
- Protect the shutdown command and logoff command
- Practice accessing one VM LPAR's disks from another VM LPAR
- Don't allocate new users on the RES volumes
  - Define your own user volumes
- Practice and document z/VM maintenance/upgrade process
  - Practice running VM second level
  - Different than what you may have done elsewhere
  - No cookie-cutter approach that works for everyone
    - Start with service/put2prod and IBM manuals then customize to what works for you

Session #9220 22

Develop a process or create/download a utility to perform system config updates. This includes using CF1/CF2/CF3 as you see fit and mixing that with any kind of individual file backup scheme among these minidisks. We do everything on CF1 and do filename backups to handle versioning. We have our source on our shared "all center" disk.

Accidentally shut down a Linux server once using the LOGOFF command and you'll quickly realize that you need to protect this. We configure this command with 'CP MODIFY CMD LOGOFF IBM * PRIV ABCDEFO' to not allow class G users to logoff (Linux runs class G). Class O is then added to all userids that need the LOGOFF command.

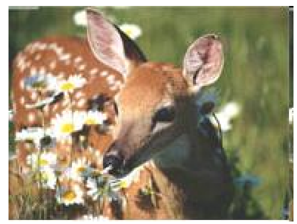
Early on we started allocating user disks onto the SYSRES volume, not realizing how the z/VM maintenance / upgrade process doesn't like having other minidisks on these. Resolved by dedicating a couple dasd volumes to each z/VM LPAR for internal userids and minidisks.

Test out a process for being able to access any z/VM's disks from another z/VM LPAR. This includes cabling and maintaining the necessary config or command documentation on how to get access.

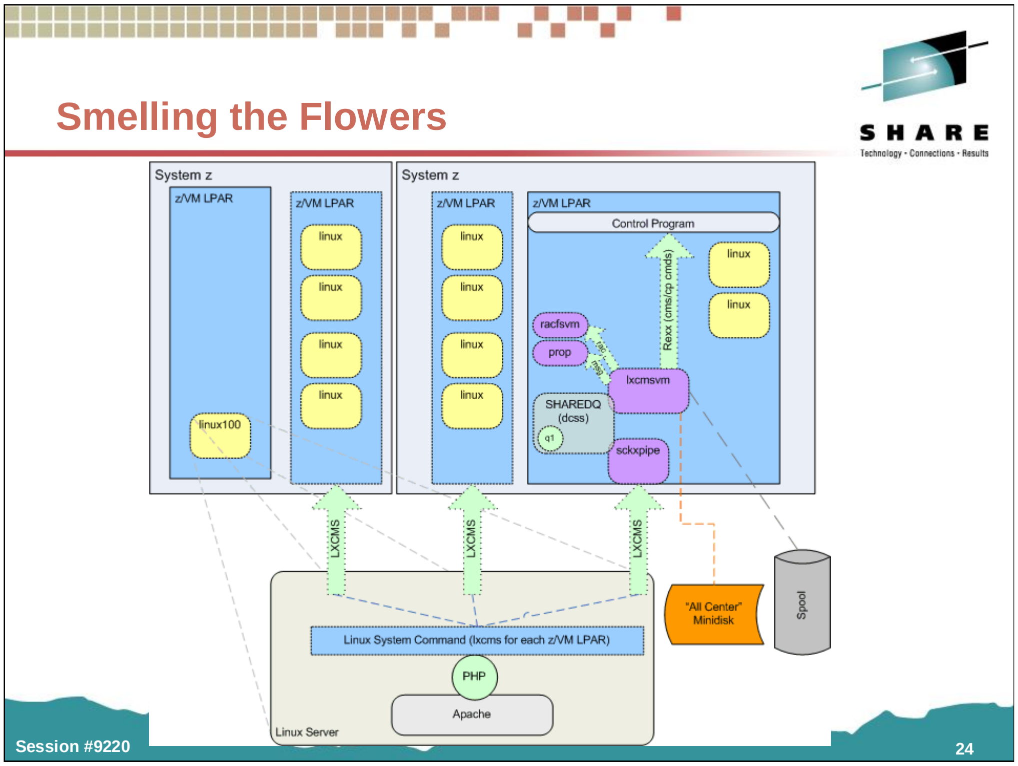
We run our systems with daylight savings time and found we had to configure automation to drive the commands to load in the new time.

Smelling the Flowers

- vmadmin Web Application
  - Single admin point for multiple z/VM LPARs
    - Rolled our own
  - Configuration
    - Linux – Apache/PHP
    - LXCMS talks to z/VM LPARs from Linux
    - Front-ends our Rexx scripts
  - Advantages
    - “Free”
    - Educational
    - Customizable to our exact needs
    - No limitations for number of LPARs
    - Avoided some features - CSE, RSCS, SFS, SMAPI
  - Disadvantages
    - Not as robust or feature rich as a purchased application
    - Currently single-threaded to one LXCMSSVM worker

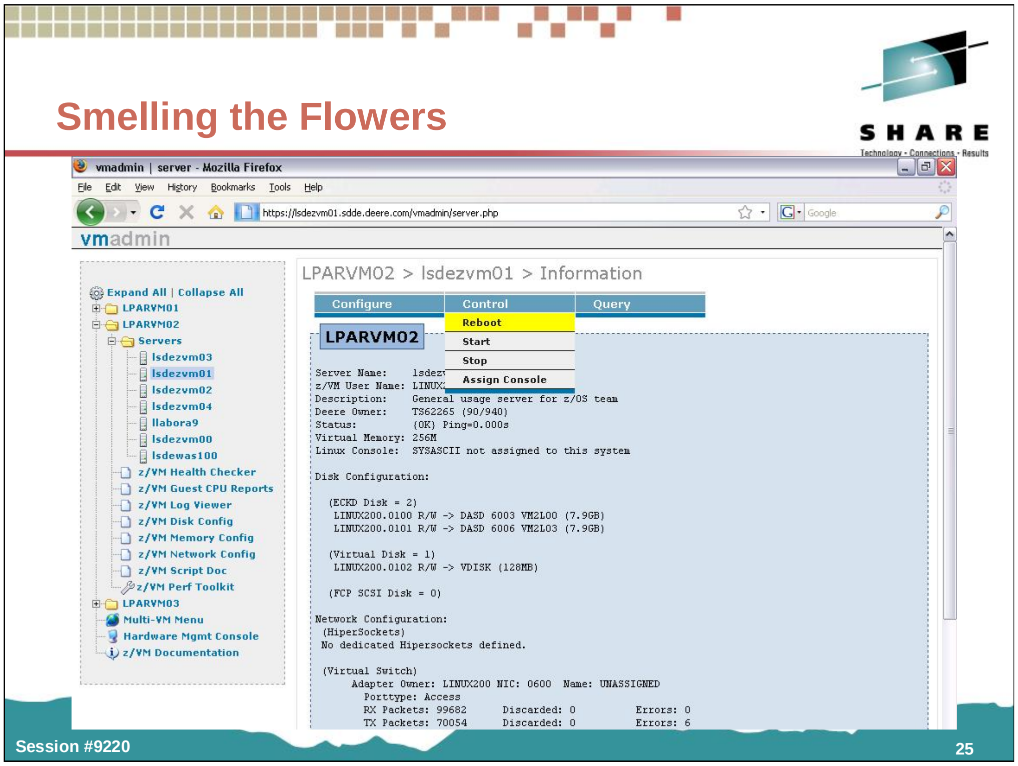


During our paced rollout for CSL, we found some time to play around with areas of z/VM administration that we wanted to automate and customize to our liking. Part of this was the concept to make a web interface to front-end the scripts we’d been writing, and make it easy to control any number of z/VM LPARs from a central point. I’ll describe what we set up to do this and show some screenshots of our tool.

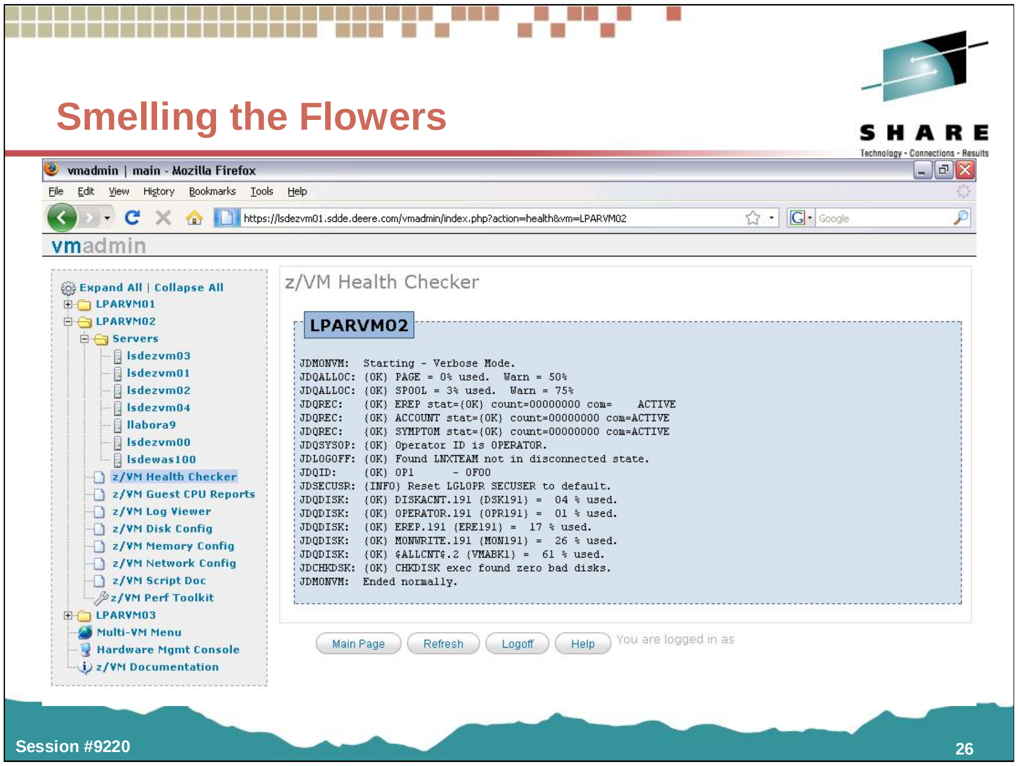


This diagram shows the logical architecture for our admin application. It was developed under PHP since it is pretty simple to work with and we had prior experience. There were a few options for interacting between Linux and z/VM, but we chose a single package called LXCMS that is available at http://www.vm.ibm.com/download/packages/.

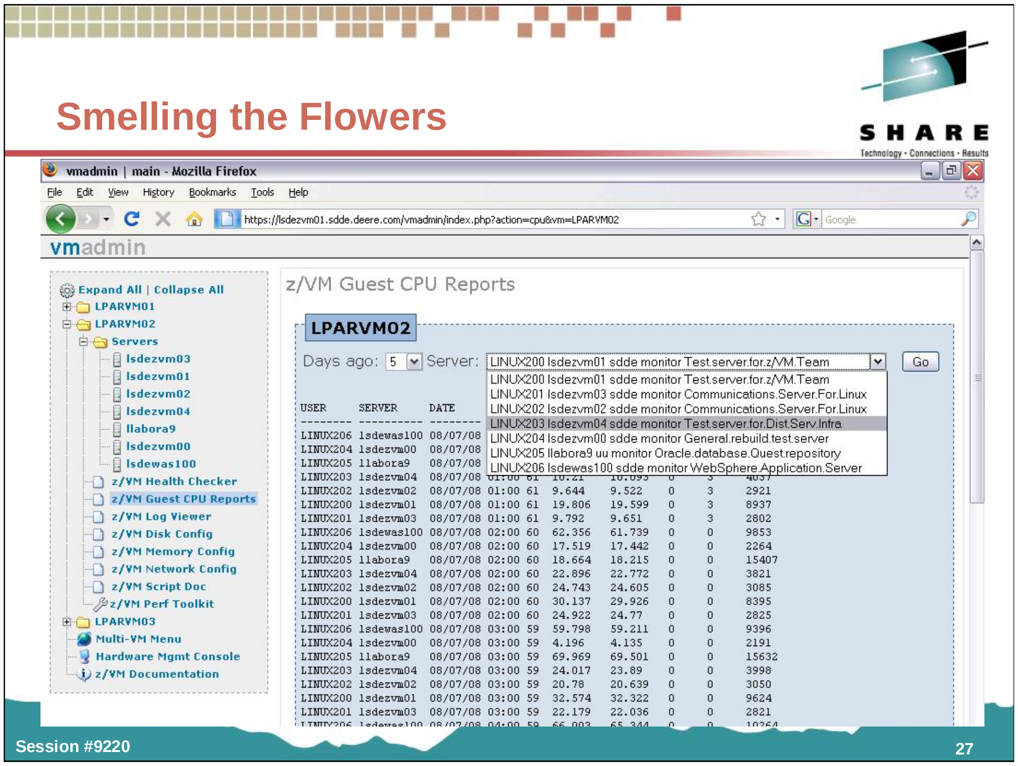
LXCMS bundles a few other packages together, using sckxpipe as the listener and SHAREDQ to talk between this listener and the user called lxcmssvm that actually executes our scripts. This lxcmssvm userid has the ability to run various cms and cp commands (including racf), as well as access a minidisk with our scripts and config files. It can route other commands through PROP if it needs more authority and can also act against spool files. This is all driven from a C program (lxcms) on the Linux server that acts as the client.



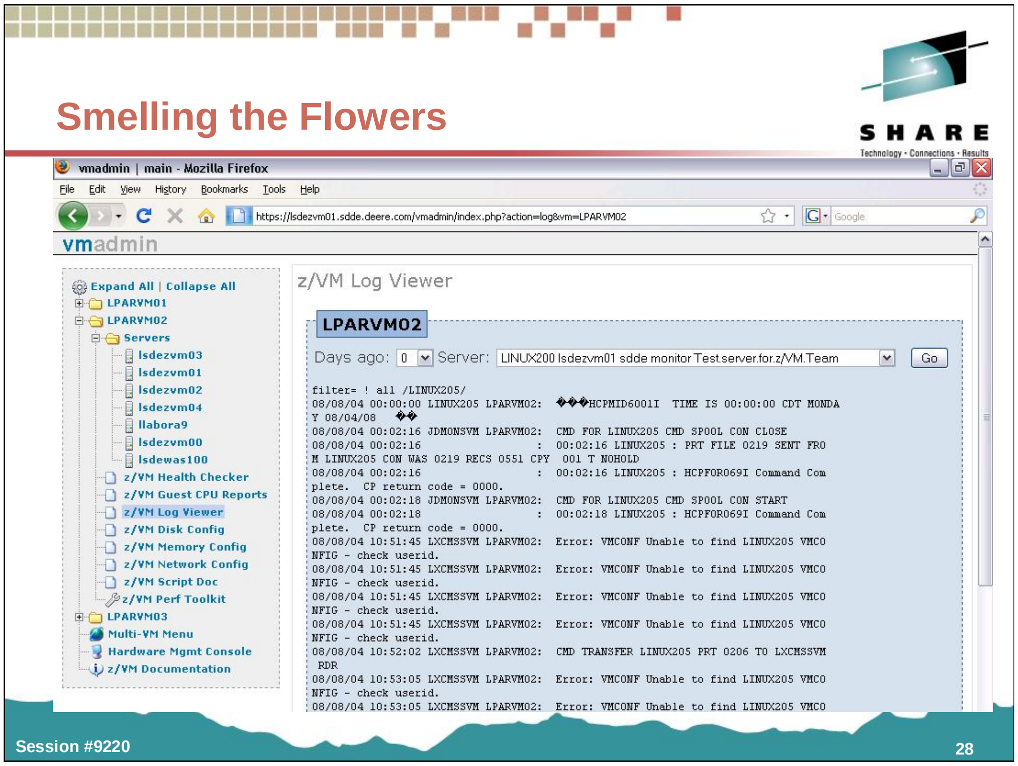
I'll just point out a few of the pages we created. We dynamically build a menu using output from one command that lists all of our Linux servers for each LPAR. From here we can perform actions on each LPAR, each Linux, or across all z/VM LPARs. This is a snapshot of what we have thus far for Linux manipulation.



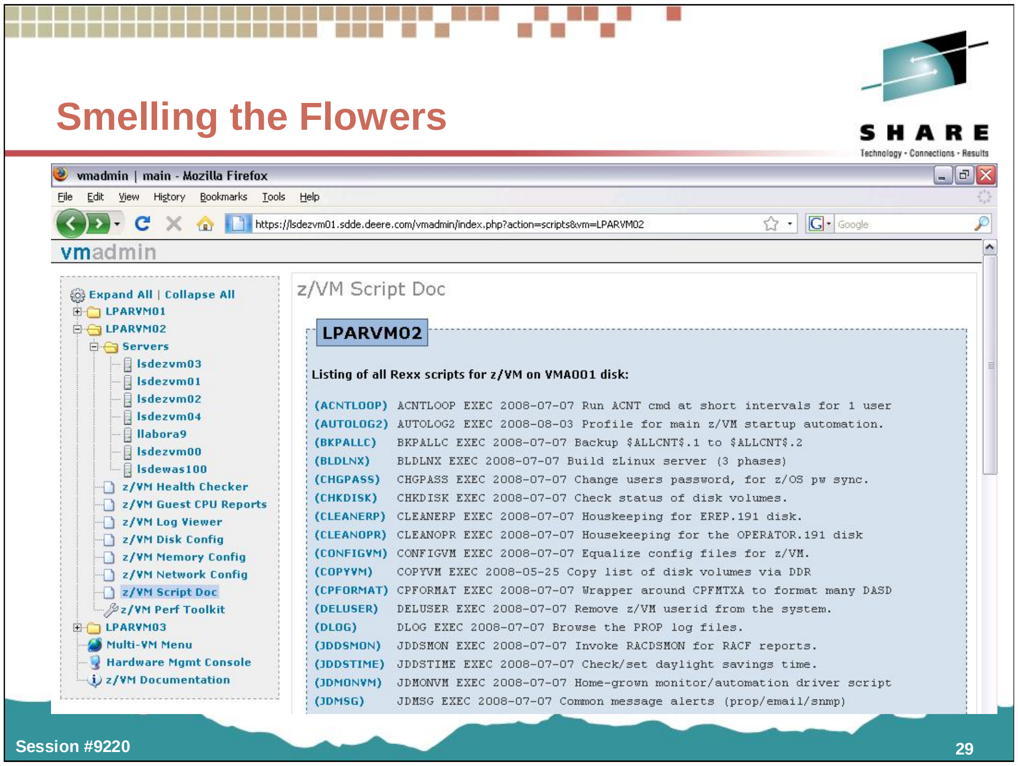
An example showing our simple health checker. We just made a single exec to drive various functions and other execs to echo out a text based status report. This can be run manually, otherwise our automation user runs it hourly and can email or snmp trap alerts to admins or operators.



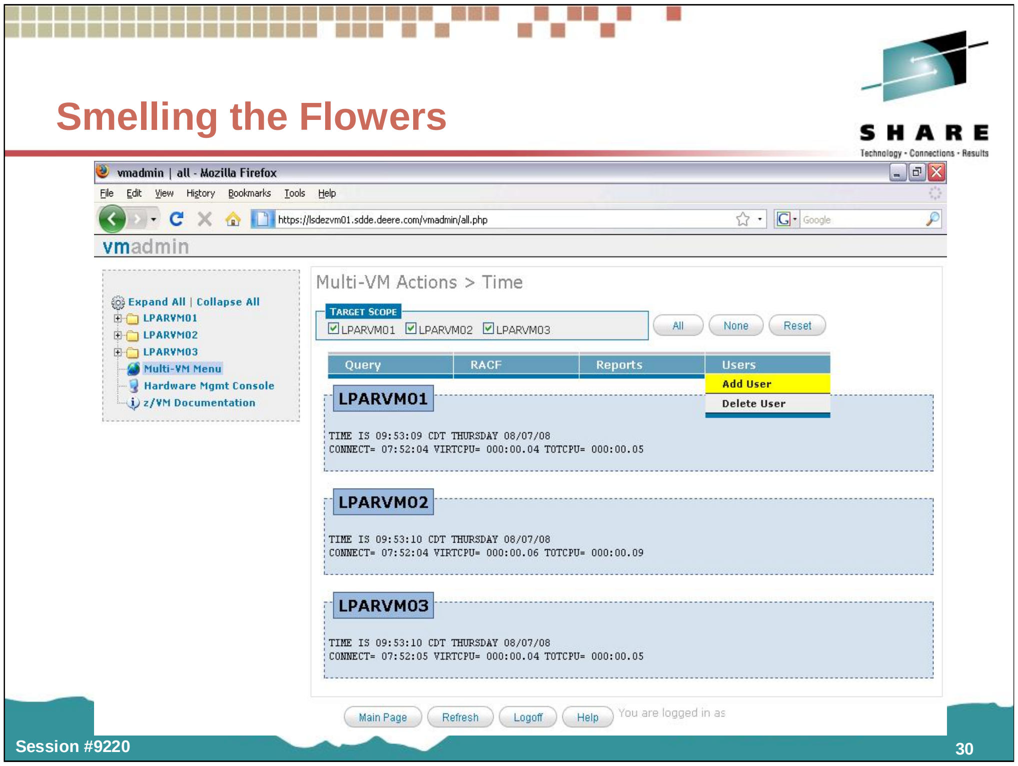
Simple report of the ACCOUNTING records for our Linux guests. This same data is exported to z/OS for long term storage, capacity planning, graphing, etc.



Log viewer for our PROP logs. We use secuser to route the Linux consoles to this central log file, then we can parse out individual servers during this viewing process.



Just another example of what we can do from a web interface. This is a trimmed down file viewer, just listing our internally created execs. We can click on one to bring up the header from it. Could be expanded to view config files or whatever we want.



Beyond controlling Linux instances, the secondary reason for this web interface was to have a single place to interact with multiple z/VM LPARs. We don't run RSCS so we wanted a way to issue a command across all systems. This interface was made generic enough that we can just add a link to the page and we can run any command across our LPARs, like adding/removing userids. We just started with this part of the application and haven't thought through all the things we want to use it for yet.

Today's Journey

- Stopwatch *(Timeline)*
- Runner's Log *(Details)*
- Marathon Planning *(Future)*
  - WebSphere

Session #9220

31

So far we've discussed where we've come from, now let's take a look at where we are going.

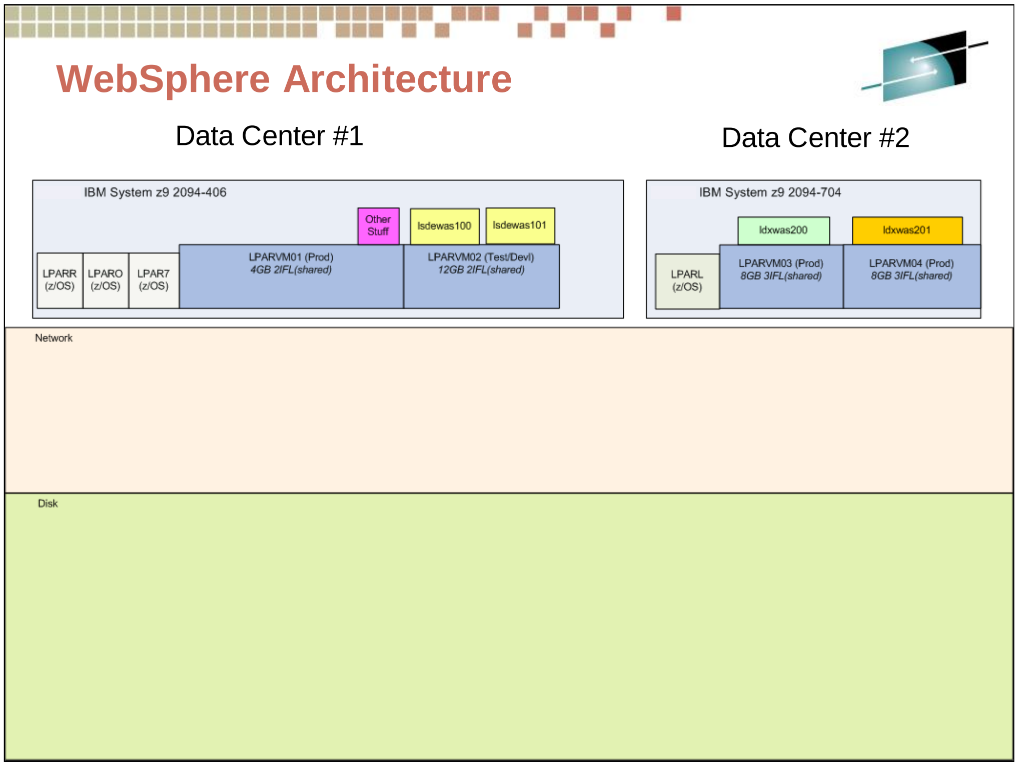
Our Next Jog

- WebSphere Application Server
  - Migration from z/OS
  - Large parts ordering application suite
  - In production build phase
    - Moving users end of 2008

Session #9220 32

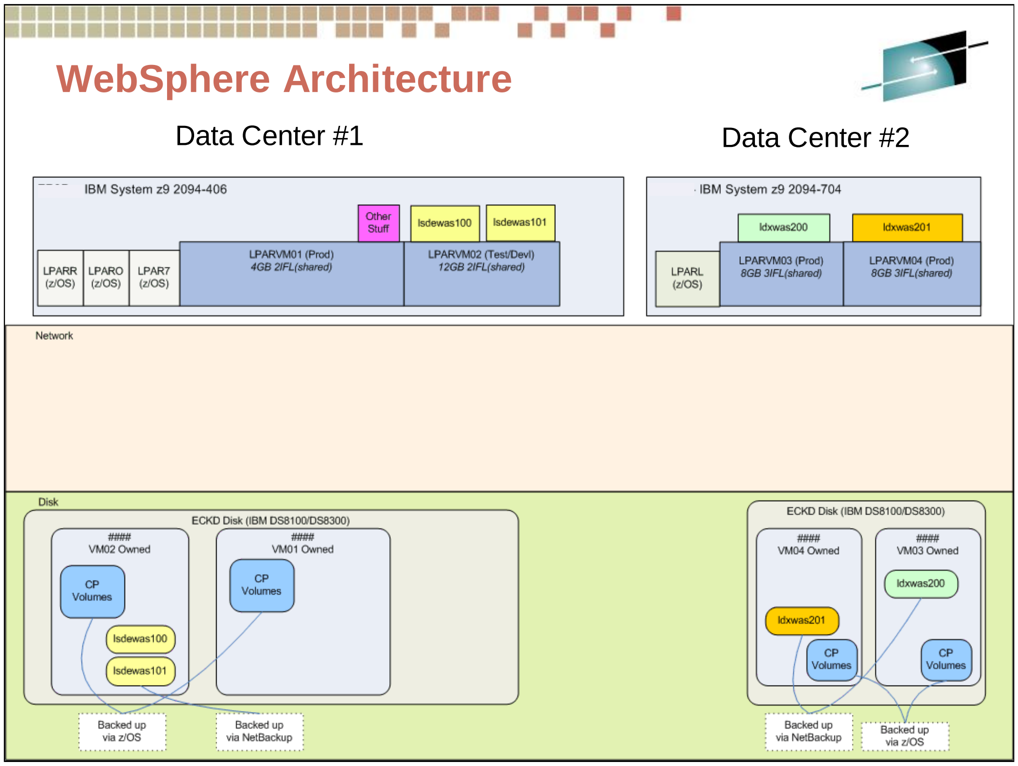
Our current project has been to evaluate WebSphere for Linux on System z. Obviously we knew it works but we spent most time doing performance/cost comparisons between z/OS, z/Linux, and x86. Anyone who’s been through this knows it isn’t apples-to-apples and takes time to agree on what to look at. Once complete though we decided to make the move and exercise more zLinux workloads to see how it really handles for us.

No major successes to report yet since we are mid-project, so I’ll just run through our diagram of what we are setting up.



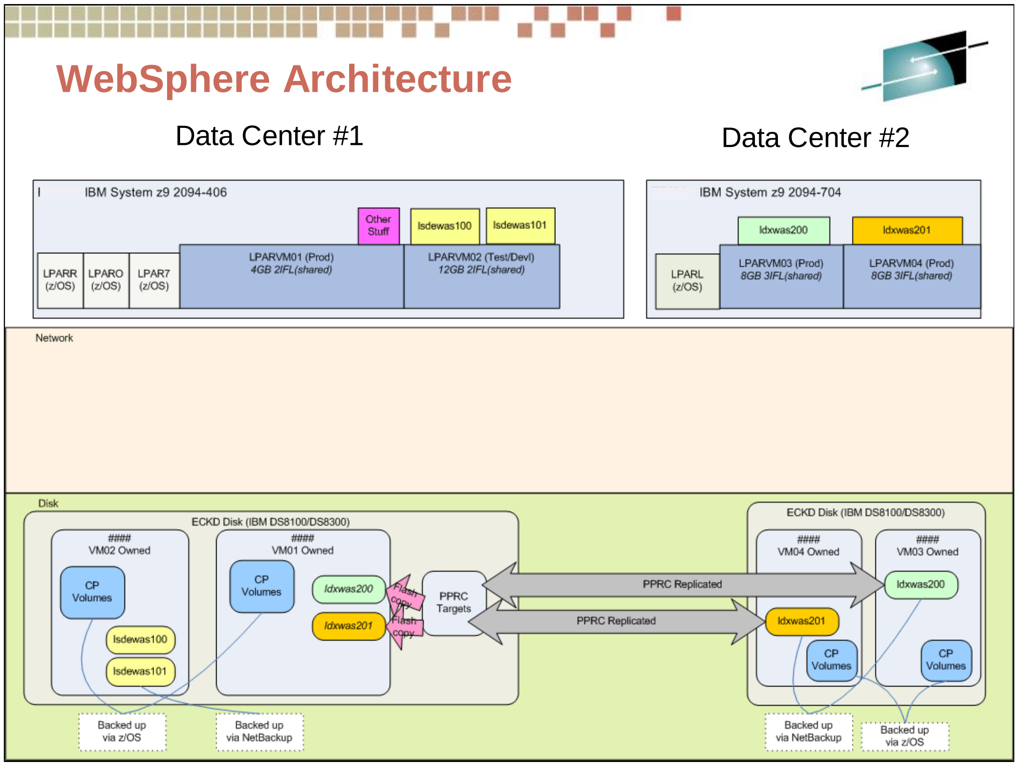
Our configuration begins with z/VM on IBM System z hardware, shown in blue. The development environment would run at Data Center #1 and the production environment would run at Data Center #2. Development Linux instances are all on a single z/VM LPAR and production is split across two z/VM LPARs.

The backend database systems for this setup are on LPARL and can use hipersockets connectivity for performance and security.



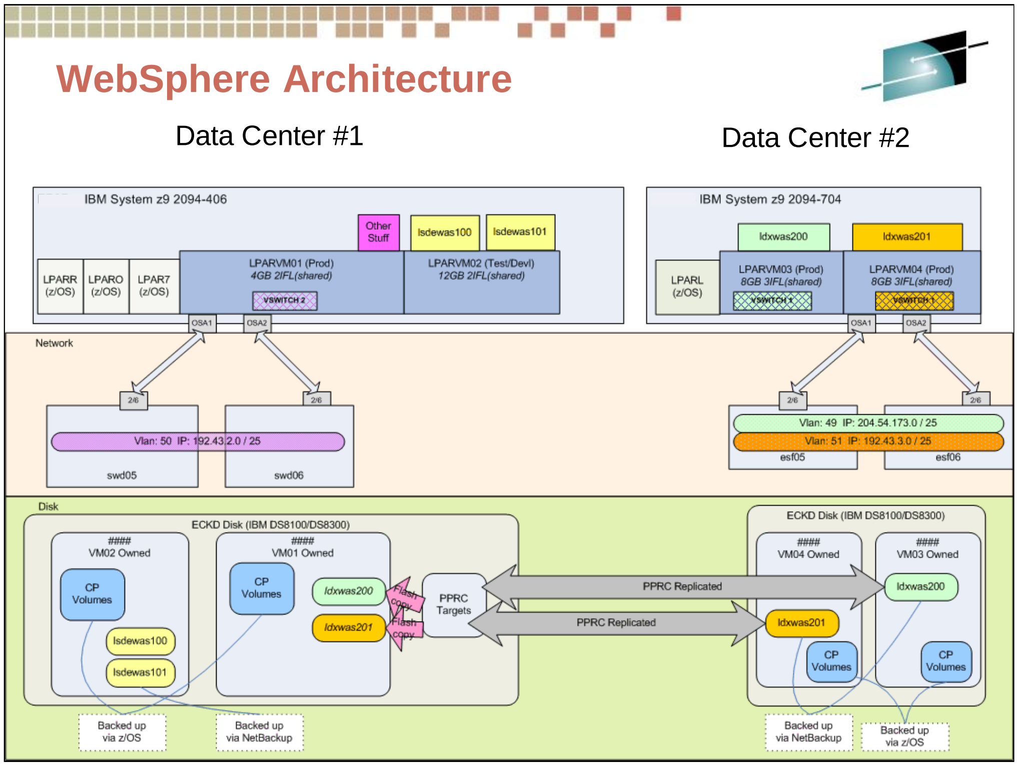
The disk layout uses a simple configuration. z/VM requires a few volumes to house configuration data, spool, and page data sets - these are shown in blue and are backed up via z/OS tape. The Linux volumes are backed up via NetBackup through the Linux operating system. This mimics the file recovery and backup/restore for single servers that is used in our current distributed environment.

Netbackup provides backup/recovery for a single server - process owned by Linux guys.

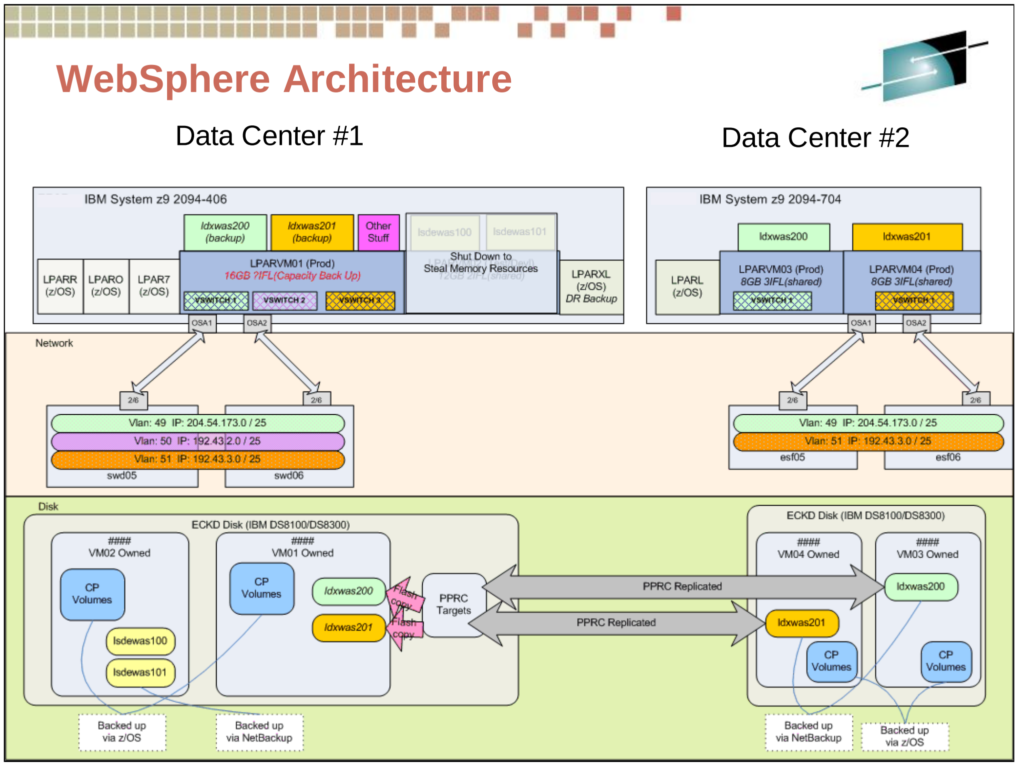


For disaster recovery, PPRC will be used to mirror the Linux guest volumes between our data centers. This is also done with all of the data accessed by z/OS LPARL.

The PPRC targets can then be flashcopied to another set of volumes that are owned by VM01. Cold standby guest definitions on VM01 will exist so these servers can be brought up there.

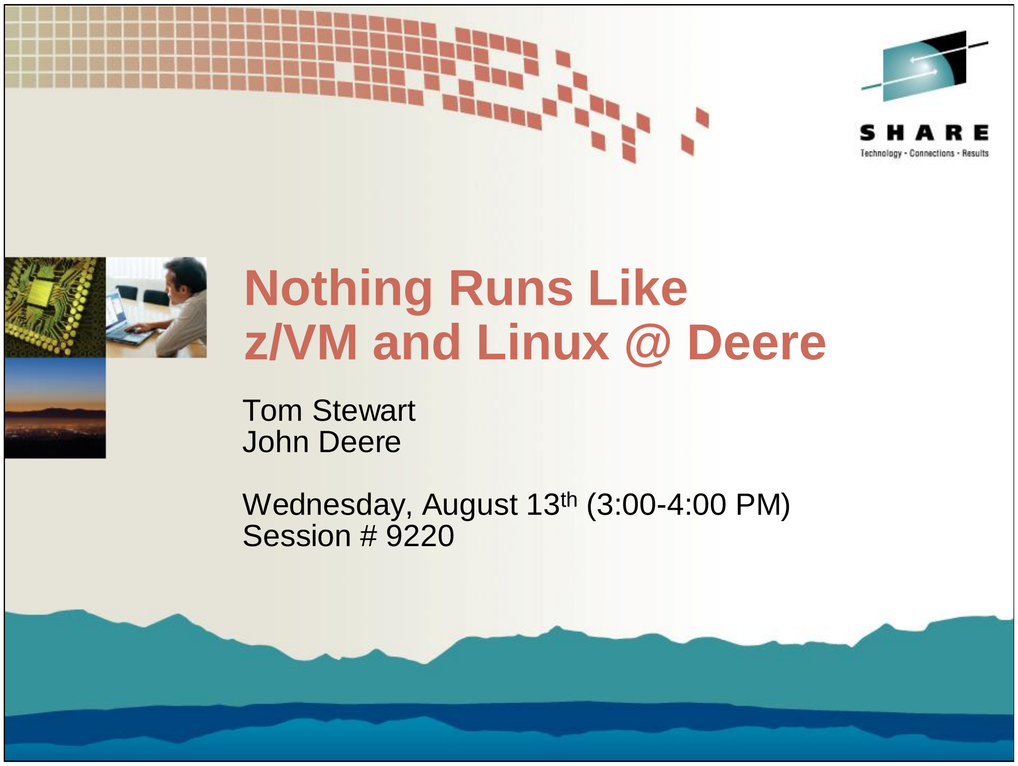


Networking for the virtual Linux environment uses the VSWITCH functionality in z/VM that extends out to VLAN's in our Cisco equipment. This is cabled in a similar fashion at both sites. The neat part is that each VSWITCH can handle traffic for multiple vlans, and 802.1Q (tagging) on the switch ports can also allow them to be shared by multiple vlans. This is crucial for easily moving vlans between sites during a DR.



In the event of a disaster at Data Center #2, this diagram shows the recovery plan.

- •PPRC replication is broken by the disk hardware and human intervention performs a flashcopy.
- •LPARL is failed over to LPARXL. This is the main data backend for WebSphere applications running in this Linux environment.
- •Network administrators deactivate the vlans at DC #2 and activate them at DC #1. No cabling is needed as it is shared with existing z/VM LPARVM01.
- •LPARVM02 is deactivated so memory can be stolen for use by LPARVM01.
- •Capacity Backup would be used to activate enough IFL's to handle the Linux workload from both sites.
- •Pre-positioned definitions for virtual switches and virtual servers are activated on LPARVM01.
- •Individual server testing can be done periodically by simply not activating the vlan at DC #1. Full tests can be done which include VLAN failover.
- •Easily modifiable to do full z/VM LPAR recovery of VM03 and VM04 into standby LPARs instead of just recovering Linux guests into VM01.



Title: 9220 - Nothing Runs Like z/VM and Linux at John Deere
Schedule: Wed 3:00 PM - Room C3 Concourse Level Convention Center

Abstract: The speaker will share some of the z/VM administrator tasks he's learned being new-ish to z/VM, covering things between where the cookbooks end and ongoing production support begins. For example:

- Managing multiple z/VM LPARs by sharing disk for config files and scripts
- Using your new best friend (Rexx) to automate tasks for Linux servers
- Quickly building Red Hat servers via kickstart files
- Creating a z/VM web admin interface with a free IBM download (lxcms) and PHP running on Linux
- Architecting for WebSphere Application Servers to run on Linux for System z, including disaster recovery planning

The intended audience is newer z/VM administrators running Linux workloads, especially those who don't yet have full commitment to purchase 3rd party provisioning and system management tools.

Speaker: Thomas Stewart
Company: John Deere
Address: 400 19th Street Moline IL 61265 USA
E-Mail: StewartThomasJ@JohnDeere.com