

How To Turn a Penguin Into a Dog

...or...
Things To Do
That Will Avoid
Linux on z Success



Phil Smith III
Voltage Security, Inc.
SHARE 111
August 2008
Session 9284

Copyright Information

SHARE Inc. is hereby granted a non-exclusive license to copy, reproduce or republish this presentation in whole or in part for SHARE activities only, and the further right to permit others to copy, reproduce, or republish this presentation in whole or in part, so long as such permission is consistent with SHARE's By-laws, Canons of Conduct, and other directives of the SHARE Board of Directors

Goals

- ▶ Examine Linux on z historical roadmap
- ▶ Learn from others' hard-won experiences
- ▶ Understand some things ***not*** to do—and why

**BEWARE OF
ATTACK PENGUIN.**

Linux on z: Ancient History

- ▶ 1999: Linux released for (then) System/390
 - IBM “skunkworks” effort
 - Works, but not a “real” IBM product
- ▶ 2000: “41,000 Linux guests on a VM system”
 - Proof-of-concept, no relation to reality
 - Garnered tremendous press attention
 - Vendors jump in: Linuxcare, Aduva, BMC...

LINUXCARE

aduva

bmcsoftware

Linux on z: Where We've Been

- ▶ 2001–2006: z/Linux growth slow
 - IBM pushes Linux on z hard (IFL loaners, etc.)
 - Many failed pilots, ROI not realized in many cases
 - zSeries CPUs not fast enough to compete with Intel
 - Levanta (Linuxcare), BMC, Aduva(?) quit market
 - Rocket enters with Linux Provisioning Expert
 - IBM adds Director for z

- ▶ **The Dirty Little Secret:**

*An **untuned** penguin can be a **dog!***

- But they can be trained, with some tools and effort



Linux on z: Where We Are

- ▶ 2006–present: z/Linux starts to grow up
 - New, faster processors (z9) made z competitive
 - Nationwide, Wells Fargo, Citi, other “poster children” validate ROI
- ▶ “Now it gets real...”
 - ...and now performance **must** be tamed!



WELLS
FARGO



citi

Important History

- ▶ Mainframes have been around for a while...
 - z/OS (OS/390, MVS/ESA, MVS/XA, MVS, MVT, MFT): **43 years** (OS/360, 1964)
 - z/VM (VM/ESA, VM/XA, VM/SP, VM/370, CP/67): **43 years** (CP/40, 1964)
 - z/TPF (TPF, ACP): **43 years** (PARS, 1964)
 - z/VSE (VSE/ESA, VSE/SP, DOS/VSE, DOS/VS): the youngster, **42 years** (DOS/360, 1965)
- ▶ We're spoiled by decades of experience
 - We expect that someone, somewhere has done it all

The New Kid on the Block

- ▶ Linux is just sixteen years old
 - Elderly in penguin years...
 - ...still immature as an OS
- ▶ Only seven years of mainframe Linux
 - Adult in dog or penguin years...
 - Progress made, but many apps still **not** well-behaved!
- ▶ z/Linux tuning and capacity planning still largely unknown territory to many
 - Each new kernel level offers new opportunities (and old opportunities return with kernel changes!)



Still a Brave New World

▶ **Nobody** really knows all the answers yet

- This is like tuning MVS circa 1980
- ...or maybe more like tuning VM/370 circa 1975

▶ **Not** a reason to avoid Linux!

- Just something to be aware of
- You **cannot** believe everything you hear, good or bad



Linux Success Requirements

- ▶ Management buy-in and distributed support group support
 - Without both of these, either:
 - Management won't care about success
 - Distributed folks will protect their turf and torpedo you
 - Management can force distributed folks' support
- ▶ Appropriate application choices
 - No fractal reductions, SETI@home
 - Java OK in moderation (many apps are evil, though)
 - VMware has similar constraints (plus no memory overcommitment)



More Success Requirements

- ▶ A willingness to say “I was wrong”
 - Some applications may turn out to be poor choices
 - Some tuning choices will have the opposite effect
 - **Requires a political climate that lets you say so**
- ▶ Monitoring, tuning, and capacity planning
 - IYDMIYWGS*
 - Many Linux apps are **not** well-behaved, mature!
 - Must make **correct** tuning choices

* If You Don't Measure It You Will Get Screwed

Reasons Linux POCs Fail

- ▶ Lack of management buy-in leading to distributed group non-support
 - “They just didn’t show up for the meetings”
- ▶ Inappropriate application choices
 - “The application we chose just didn’t perform”
 - “Management lost patience”
- ▶ Disappointed by performance
 - Without tools, no way to understand
 - “There is no think, only do” — Master Yoda



The Real Cause

- ▶ Inappropriate expectations
 - Running thousands of Linuxen on one system
 - “Just port it and it will run”
 - “Mainframes are large and fast”
- ▶ The reality
 - Plan dozens or hundreds of Linuxen per system, **tops**
 - Porting requires understanding, (maybe) rearchitecting
 - Mainframes are **fairly** large and **fairly** fast—now (z10)



www.dvdrewinder.com

How To Guarantee Failure

•Copyright 2007 Velocity Software, Inc. All Rights Reserved.
Other products and company names mentioned herein may be
trademarks of their respective owners.

Unmeasured Equals Unsuccessful

- ▶ Make unjustified assumptions
 - “Tune it like MVS” (aka “Linux apps are well-behaved”)
 - “The app needs 4GB on Intel, so we’ll give it 4 on z”
 - “More CPUs are good”
 - “Swapping is bad”
 - “z/VM is 64-bit, so we should run 64-bit Linux”
- **Critical requirement: You *must* measure it!**
 - I’ve believed this since long before joining Velocity

Performance Tuning “Back in the day”

- ▶ VM in days of old
 - Hundreds (or thousands!) of CMS users
 - Relatively small, well-behaved applications
 - Performance degradation was typically gradual
- ▶ Performance tuning was easier **and** harder
 - **Easier**: smaller problems, smaller changes
 - **Harder**: smaller changes, smaller effects



Why Linux is Different

- ▶ z/VM today
 - Tens (or hundreds) of z/Linux guests
 - Very large, often poorly behaved Linux applications
 - Performance degradation can be precipitous
- ▶ Performance tuning is harder *and* easier
 - **Harder**: bigger problems, bigger changes
 - **Easier**: bigger changes, bigger effects





Herding Penguins

The single
most important
lesson in
this presentation

(but easier
than
herding cats)

Your Penguins Must Sleep!*

- ▶ Your idle Linux guests **must** go **truly** idle
 - This is a **memory** (storage) management issue, **not** a CPU usage issue
- ▶ What does “idle” mean?
 - Means “transaction” complete, guest drops from queue
 - CP defines 300ms of idle time = end of transaction
 - Theoretically represents interactive user “think time”
 - Less meaningful for servers, but what better metric?



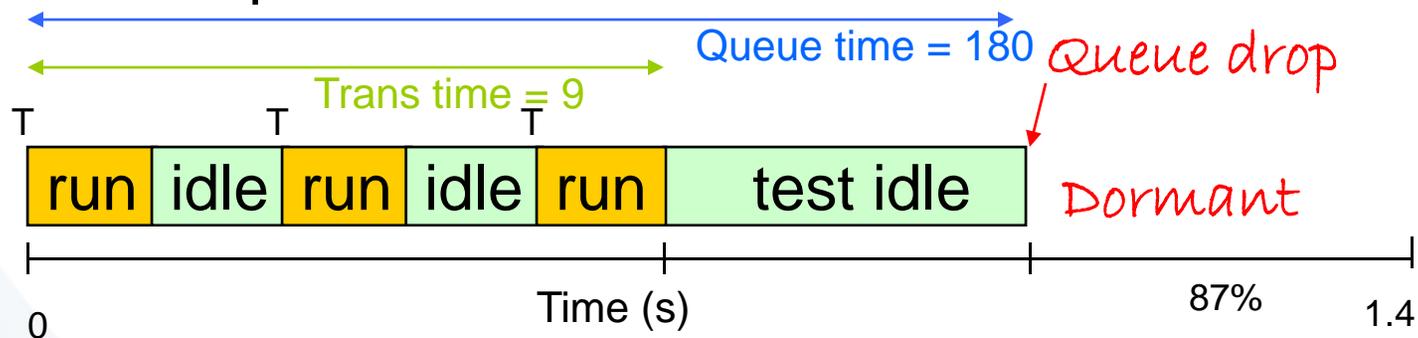
* Thanks to Rob van der Heij for this line!



What's a "Transaction"?

- ▶ Anatomy of the average transaction
 - Periods of activity with short idle time between them
 - Starts with a timer interrupt (or perhaps I/O interrupt)
 - Longer idle period at end is followed by queue drop

- ▶ Example:



Scheduler and Dispatcher 101

▶ Some critical concepts

- Guests must be *runnable* to do work
- CP must be willing to schedule the guest
- CP must be willing to dispatch the guest
- ▶ A guest is always in one of three lists:
 - **Dormant** list: guest has no work to do
 - **Dispatch** list: guest is active, CP is allowing it to run
 - **Eligible** list: guest is active, CP is not allowing it to run
 - (Can also be **running**...special case of Dispatch list!)

Scheduler and Dispatcher 101

- ▶ **CP scheduler** analyzes resources, decides whether enough to give guest service
 - Entirely storage-related (memory)
 - If not enough available, guests get put on the E-list
- ▶ **CP dispatcher** gives guests access to CPUs
 - If multiple guests are active, they take turns
 - VM is very good at this — supports tens of thousands of active users with excellent response time

Dispatch Classes – Class 1

- ▶ When first dispatched, guest is Class 1 (“Q1”)
 - CP waits one Class 1 Elapsed Timeslice (C1ETS) to see if it goes idle voluntarily
 - Guests that do not go idle within that timeslice are preemptively stopped from execution— sent back to the scheduler
 - C1ETS is dynamically calculated to keep a fixed % of guests in class 1
 - C1ETS should be enough for short, interactive transactions (minor CMS commands)

Dispatch Classes – Class 2

- ▶ If guest does not go idle in one C1ETS, it enters Class 2 (“Q2”)
 - Next time CP runs it, given 8x C1ETS
 - Guests that do not go idle within that amount of time are rescheduled
 - Such guests are presumed to be running a command, but not necessarily doing something “major”

Dispatch Classes – Class 3

- ▶ If guest does not go idle within class 2 C1ETS multiple, it enters Class 3 (“Q3”)
 - Next time CP runs it, given 6x Class 2 = 48x C1ETS
 - Guests that do not go idle within that amount of time are rescheduled
 - Such users are presumed to be running a long-running command

Dispatch Classes – Class 0

- ▶ **QUICKDSP ON** bypasses some rules
 - Still get rescheduled, but never held in eligible list
- ▶ Interactive guests (on terminals, hitting keys) also get Q0 stays (“hotshot” stays)
 - Still get rescheduled, but “go to head of line” briefly
 - Return to their previous queue level after Q0 stay
 - Virtual machines holding certain short-term system locks are also considered to be in Q0

Leaving the Dispatch List

- ▶ Guests leave dispatch list because they:
 - Use up their current *CnETS* multiple
 - Go idle voluntarily (load a wait PSW)—see below
- ▶ 300ms **test idle timer** set when guest loads wait PSW
 - Guest resuming activity within that period are reinserted into previous place in queue
 - Guests that don't go idle never get queue dropped!

How This Plays Out...

- ▶ CP scheduling is based on storage analysis
 - If not enough, guests are held in **Eligible list (E-list)**
 - Assumption: other guests will go idle, storage will become available soon
 - If not, E-listed guests never get scheduled
 - There are actually a host of other bad side-effects of too-large Linux guest virtual storage sizes



Why This Goes Wrong

- ▶ Linux real storage requirements higher than CMS guests because Linux guests:
 - Are quite large (virtual storage size)
 - Use all storage (working set = virtual storage size)
 - Don't interact with CP to release unused storage
 - Stay active (rarely/never go idle)
- ▶ If enough Linux guests are logged on, CP notices it will overcommit real storage
 - One or more such guests “lose”, are E-listed — and stay there!

How Does This Manifest?

- ▶ System is running along fine
 - One guest too many is started
 - Things “just stop”!
- ▶ Dispatched guests “should” go idle
 - Linux guests typically don’t, stay runnable all the time
- ▶ Historically, guests doing I/O were “active”
 - Recent releases have mostly eliminated this
- ▶ Remember the test idle timer
 - Guests never go idle (as far as CP can tell)
 - Never get scheduled properly, so E-listing permanent!



Detection

▶ CP INDICATE QUEUES EXPANDED shows:

```
LINUX902      Q3 PS  00013577/00013567  .... -232.0 A00
LINUX901      Q3 PS  00030109/00030099  .... -231.7 A00
VSCS          Q1 R   00000128/00000106  .I.. -208.7 A00
VMLINUX3      Q3 IO  00052962/00051162  .... -.9398 A00
VMLINUX3 MP01 Q3 PS  00000000/00000000  .... .0612 A00
LINUX123      E3 R   00177823/00196608  .... 5255. A00
```

- **HELP INDICATE QUEUES** shows meaning of output
- CP privilege class E required
- **Note:** “deadline time” (sixth column) indicates when CP thinks the guest will run
- Guest **LINUX123** is not running any time soon...

Remediation

- ▶ Buy lots more storage ($\$ < 6K/GB$ — cheap!)
- ▶ Tune applications so guests do queue drop
 - Obviously only meaningful if guests are nominally idle
 - Remember cron et al. may wake them anyway
- ▶ Log off some guests
 - You didn't need that WAS application, did you?
- **Tune guest storage sizes**
 - Linux uses “extra” storage for file buffers
 - Smaller guests may actually perform ***better***

Why Idle Guests are Important

- ▶ CP analyzes storage use when guests go idle
 - Avoids taking pages from active guests
- ▶ Three-pass process
 - First pass analyzes users on dormant list—never happens if Linux guests never go idle!
 - Result: CP must steal pages, makes wrong guesses
 - Causes thrashing—pages go out, come right back in
- ▶ Linux and z/VM paging algorithms collide
 - When Linux wants a page, where does it look? (LRU)
 - Where is that page most likely to be?



Care and Feeding of Aptenodytes

Keeping
your penguins
from
becoming dogs

“Jiffies”

- ▶ “Jiffies”: Frequent Linux timer pops
 - Controlled via setting in `/proc`
- ▶ “Correct” setting is perhaps unintuitive
 - 0 is what you want:
`echo 0 > /proc/sys/kernel/hz_timer`
- ▶ Why do “jiffies” hurt?
 - 10ms is a lot less than the CP idle timer of 300ms
 - Guests with the timer ON never go idle

Make sure “jiffies” are off!



Virtual Multiprocessors

- ▶ Don't use virtual MPs without **good** reason
 - Most Linux applications don't exploit MP
 - Exception: apps that use more than one CPU of MIPS
- ▶ Bogus advice, frequently heard:
"Define as many vCPUs as real CPUs"
 - Valid **only** in lab, single-Linux-guest environment
- ▶ Note: Linux doesn't report MP usage
 - Harder to prove MP need (or lack thereof)

Virtual Multiprocessors

- ▶ Why does this hurt?
 - Guest isn't idle until all vCPUs are idle
 - Virtual MP spreads timer events over vCPUs
 - Thus MP = more transactions = more in-queue time
 - ▶ Bigger problem: significant CPU wastage
 - Inter-vCPU management isn't free
 - Linux spin locks can use an **entire CPU**
- Use virtual MP only if proven need

Extra Services

- ▶ Be careful about `cron` and friends
 - Services such as `cron` wake guests up from idle
 - Obviously necessary in some cases, but examine, understand, and experiment!

Understand requirement for every service

Update Services and Friends

- ▶ Watch for the “thundering herd” phenomenon
 - Things like Red Hat Network tend to wake guests up
 - All your guests waking up at once is **not** a good thing!
 - Examine, understand, and stagger the wakeups
- ▶ Avoid/aggregate services such as updates
 - Why check for updates on **every** guest?
 - Use a single update server!



64-bit Linux

- ▶ z/VM no longer runs on 31-bit hardware
 - 31-bit guests still supported, but...
- ▶ Natural assumption: 64-bit guests “better”
 - 64-bit guests require significantly more resources
 - Page tables alone are twice as large (16MB per GB)
 - Other control structures can also be significant

Use 64-bit guests only when > 2G virtual memory or specific application requirement

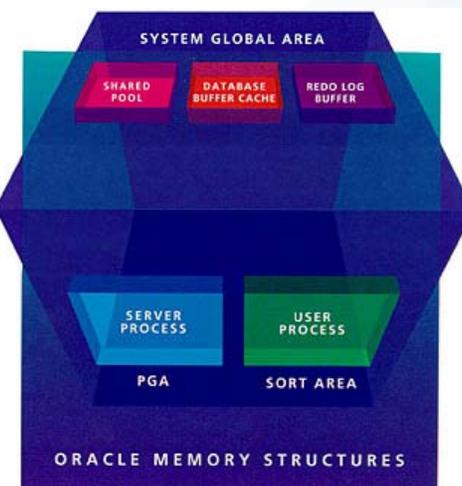
Swapping and VDISK

- ▶ Intel boxes have fast CPU, RAM; slow disk
 - Conventional wisdom: “Swapping is bad”
- ▶ Swapping to DASD is slow
 - **But** z/VM has VDISK (virtual disk in storage)
 - “Minidisks” that exist in z/VM paging subsystem
- ▶ z/VM paging subsystem is pretty darned fast
 - Conventional wisdom thus **mostly wrong** under z/VM
- ▶ Swapping to VDISK is **way** fast
 - Linux still does I/O, but CP intercepts and handles
 - CP can manage VDISK better (LRU problem again)

Swapping and VDISK

- ▶ Most applications can stand to swap some
 - Exception: Oracle Shared Global Area (SGA) **must** stay in-memory for reasonable performance
 - Other exceptions surely exist
- ▶ **Use small virtual storage + Swap to DASD to slow down guest that's too fast ;-)**

ORACLE®



VDISK Myths and Realities

- ▶ Fear: “VDISK will use too much real storage”
 - Reality: VDISK lives in VM paging subsystem
 - Linux virtual storage lives in VM paging subsystem
 - Real storage use not really affected
- ▶ Reality: VM does better managing both
 - Use smaller Linux virtual storage + VDISK for swap
 - VM controls both, rather than Linux caching data, causing VM paging through LRU mismatch
- ▶ Myth: “VDISK pages never migrate to DASD”
 - Fact: CP Monitor records prove otherwise

VDISK Notes and Recommendation

- ▶ VDISK notes:
 - Max size: 2G-1 page (4194296 512-byte blocks)
 - Control via CP `SET VDISK` command (privileged)
- ▶ Use **two** VDISKs, prioritized
 - Linux “moving cursor” algorithm wanders across disk
 - With one, large VDISK, entire disk winds up “dirty”
 - With two, Linux will use higher priority first
 - Avoids old, “dirty” pages lingering in VM paging space
 - Note: “higher priority” is numeric — 10 is higher than 1 (unlike your tasks at work!)

Large Virtual Storage (Memory)

- ▶ Example: 256MB virtual storage vs. 1024MB
 - 8MB more real storage required just for page tables
 - 16MB if 64-bit guest!
 - Significant even if not actually **using** the storage!
- ▶ Recommendation: Tune virtual storage size
 - “Squeeze until it hurts”
 - Then give it a bit more (or not — let it Swap, to VDISK)



Virtual Storage and Linux Caching

- ▶ Linux caches data (read *and* write)
 - Data may be replicated five times:
 1. Linux file buffers
 2. z/VM minidisk cache/paging subsystem
 3. Controller cache
 4. Device cache
 5. “Brown, round, & spinning”

Multiply cached data probably not helpful!

- Tuning virtual storage size controls this

Minidisk Cache

- ▶ Minidisk cache (MDC) is a powerful tool
 - But only for data that actually gets reread
 - And not if the data is cached by Linux too...
- ▶ Default: MDC uses both main and XSTORE
 - CP “Arbiter” that controls XSTORE use seems broken
 - MDC can use huge amounts of XSTORE for no gain
 - Even decent MDC hit ratio may not justify increased paging load due to reduced main/XSTORE available
- ▶ **CP SET MDCACHE XSTORE 0M 0M**

QUICKDSP ON

- ▶ CP SET QUICKDSP ON *sounds* good
 - “This guest is important, we want it to run fast!”
- ▶ Reality: makes guest avoid scheduler, *not* “run faster”
 - Circumvents scheduler “smarts”
 - Result: when storage overcommitted, CP thrashes
 - Result: worse performance for everyone
- ▶ Use QUICKDSP only by prescription*

* And for MAINT, when you’re doing performance tuning...!

SHAREs

- ▶ ABSOLUTE SHAREs **sound** good
 - “We can ensure that this machine gets xx% of a CPU!”
- ▶ Reality: Difficult to manage with many guests
 - With one or two, quite feasible—but at that point, RELATIVE SHAREs work just as well
 - Use ABSOLUTE for TCPIP et al (machines others depend on) to ensure service even when system busy
 - Note ABSOLUTE SHAREs are % of **entire system**
- ▶ Leave SHARE at **RELATIVE 100** unless addressing **specific** performance problem

SRM

- ▶ CP SRM settings provide some system performance management “knobs”
 - Be careful: These are **big** knobs
 - **Misapplied, they *will hurt!***
- ▶ Default SRM settings based on CMS users
 - Most are still OK for z/Linux
 - Be careful of “lore” suggesting changes unsupported by measured results

SRM LDUBUF

- ▶ Some “lore” suggests raising **SRM LDUBUF** is a good idea
 - Actual measured results suggest otherwise
 - Controls the number of “loading” users (users with significant paging activity) allowed in-queue
- ▶ **Never never increase this with z/Linux!**
 - In large shops, may actually want to **lower** it
 - E.g., 50 page packs on 8 CHPIDs—CP probably can’t really support that many loading users

SRM STORBUF and XSTOR

- ▶ **STORBUF** controls CP's storage usage calculations by queue
 - Linux guests are always Q3, so default incorrect
 - Best to essentially disable its function
 - Default: `SET SRM STORBUF 125 95 75`
 - Suggest: `SET SRM STORBUF 300 300 300`
- ▶ Also: `SET SRM XSTORE 50%`
 - Includes 50% of expanded storage in calculations
- ▶ **Measure** results on your system!

z/VM 5.2 and 5.3

- ▶ IBM has done **tons** of work to make z/VM a better host for Linux
 - Example: fixes allow queue drop when I/O outstanding
- ▶ z/VM 5.2/5.3 continue the tradition
 - Many small enhancements that make Linux run better
 - z/VM upgrades aren't a big deal any more
- ▶ If you aren't on 5.2 or 5.3, get there ASAP!
 - 5.3 is best choice

CMM

- ▶ CMM: Collaborative Memory Management*
 - Allows dynamic Linux storage tuning
- ▶ Driver from IBM Böblingen
 - Accepts commands via CP `smsg`, allocates storage within Linux, tells CP “fuhgeddaboutit”
 - CP no longer has to manage those pages
- ▶ Lets you “inflate a balloon” within Linux
 - Linux continues operation, working set greatly reduced
 - If swapping becomes a problem, release some pages!

* Or possibly “Cooperative Memory Management” — nobody seems to be sure!

CMM In Action

▶ Linux without CMM

4GB
virtual
storage

▶ Linux with CMM

4GB
virtual
storage
minus nn
pages

Linux still *thinks* it has 4GB

“Rest” of storage *not managed by VM*

Multiply savings by n guests...

CMM Benefits

- ▶ CMM avoids most of the complaints about storage tuning
 - “We don’t want to reboot”
 - “This isn’t peak load, and we can’t reboot when it is!”
- ▶ Critical for Linux success in some shops
 - Real example: Oracle said “App needs 4GB”; Linuxen have 4GB, but only 1GB *really* available!
 - Apps folks still *think* they have 4GB
 - Without CMM, $n \times 4\text{GB} = \$\$\$$ for more real storage (or unacceptable performance)

CMM2

- ▶ z9 added hardware support for “CMM2”
 - Cooperative z/VM–z/Linux page management
 - Intended to reduce double paging, LRU thrashing
- ▶ Adds **CP SET** and **QUERY MEMASSIST**
 - Requires z/VM 5.2 with PTFs UM31784, UM31868
 - SLES 10 SP1 supports via `cmma=on` IPL option
 - No support in RHEL4 or RHEL5 (yet?)
- ▶ No proven success in the field
 - Stick with CMM(1) for now

XIP

- ▶ XIP = eXecute-In-Place
 - DCSSs under Linux, containing stored, shared data
 - Manifest as special filesystem type
- ▶ Use XIP when possible to share static data
 - Common applications can save significant real storage
 - Requires some management and care
 - Evolving area, stay tuned!
- ▶ Explore for common apps (SNMP, etc.)

Summary



(C) ON-NO.NET

Summary

- ▶ Linux on System z is reaching adolescence
 - Much progress made, lots more to do
- ▶ Tuning Linux on z is an emerging science
 - We're still learning, and it's a moving target
- ▶ As always, use the community
 - z/Linux mailing list: `LINUX-390@marist.edu`
 - z/VM mailing list: `IBMV@listserv.uark.edu`
- **Measure, test, prove — don't rely on rumor, innuendo, and lore!**

Questions?



Contact Info

Phil Smith III

703.476.4511 (direct)

phil@voltage.com

Thanks to

Barton Robinson

Rob van der Heij