zSeries Technical Conference

Linux on z/VM Resource Economics

Rob van der Heij
IBM Netherlands
rvdheij@nl.ibm.com

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

BookManager*
DB2*
DFSMS/MVS*
DFSMS/VM*
e-business logo*
Enterprise Storage Server
ESCON*
FICON
GDDM*
IBM*
IBM logo*
Language Environment*
Multiprise*
MVS
NetRexx
OpenEdition*
OpenExtensions
OS/390*
Parallel Sysplex*
PR/SM
QMF
RACF*
RAMAC*
S/390*
S/390 Parallel Enterprise Server
VisualAge*
VisualGen*
VM/ESA*
VSE/ESA
VTAM*
z/Architecture
z/OS
z/VM
zSeries

The following are trademarks or registered trademarks of other companies.

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation; LINUX is a registered trademark of Linus Torvalds; Penguin (Tux) compliments of Larry Ewing; Tivoli is a trademark of Tivoli Systems Inc.; Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries; UNIX is a registered trademark of The Open Group in the United States and other countries; Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation; SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC. * All other products may be trademarks or registered trademarks of their respective companies.

Notes: Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply. All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions. This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area. All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Redbook

Residency: February 2003

Draft: April 14, 2003

Publication: May 2003

Contents

Virtualization and server consolidation

z/VM memory and storage concepts

Linux virtual memory concepts

Tuning memory for z/VM Linux guests

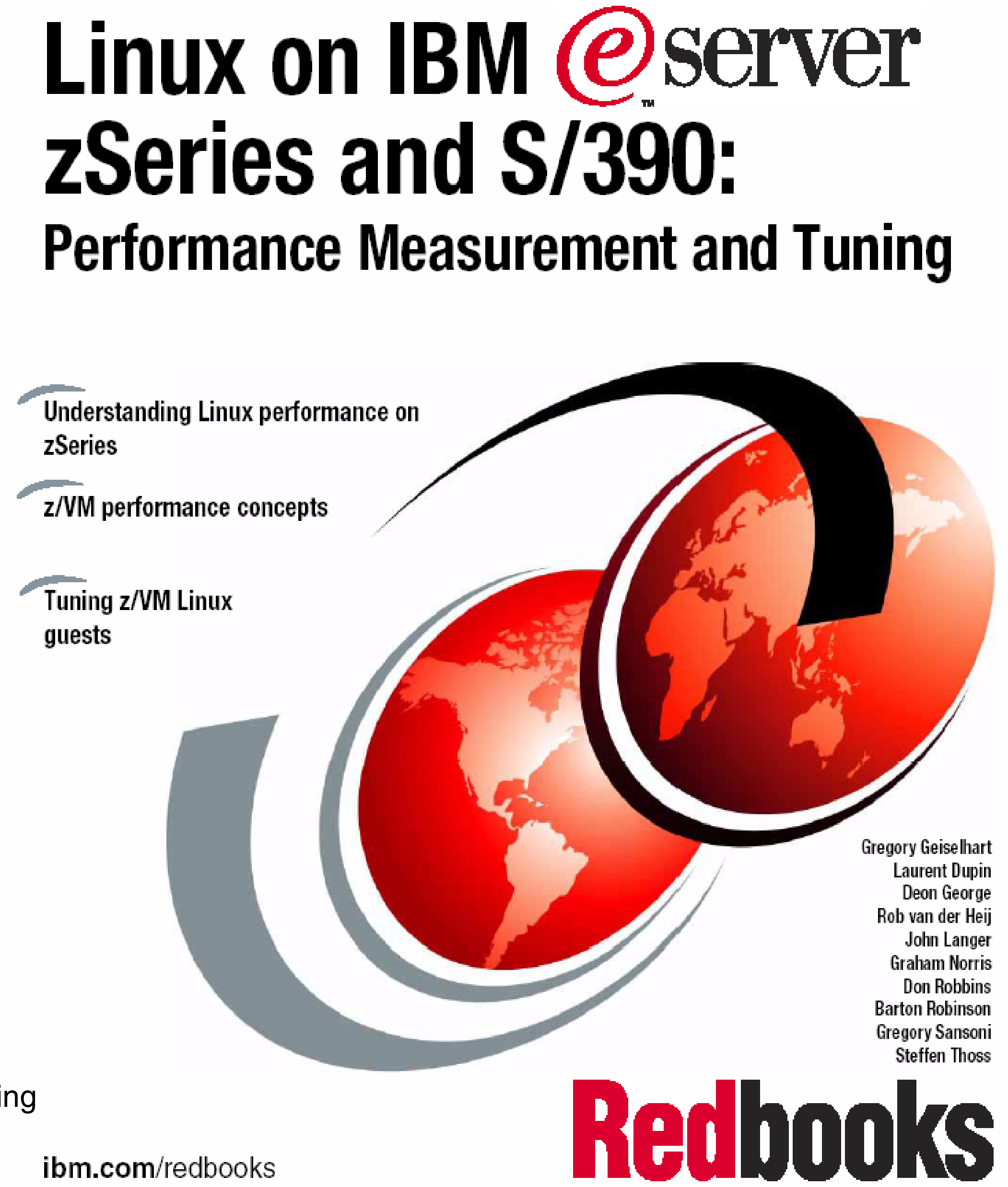
Examining Linux swap device options

CPU resources and the z/VM scheduler

Tuning processor performance for z/VM Linux guests

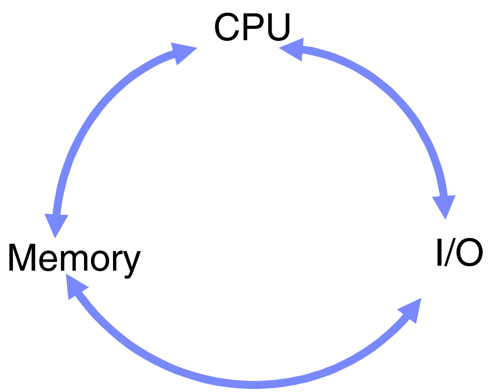
Tuning DASD performance for z/VM Linux guests

Measuring the cost of OSA, Linux, and z/VM networking



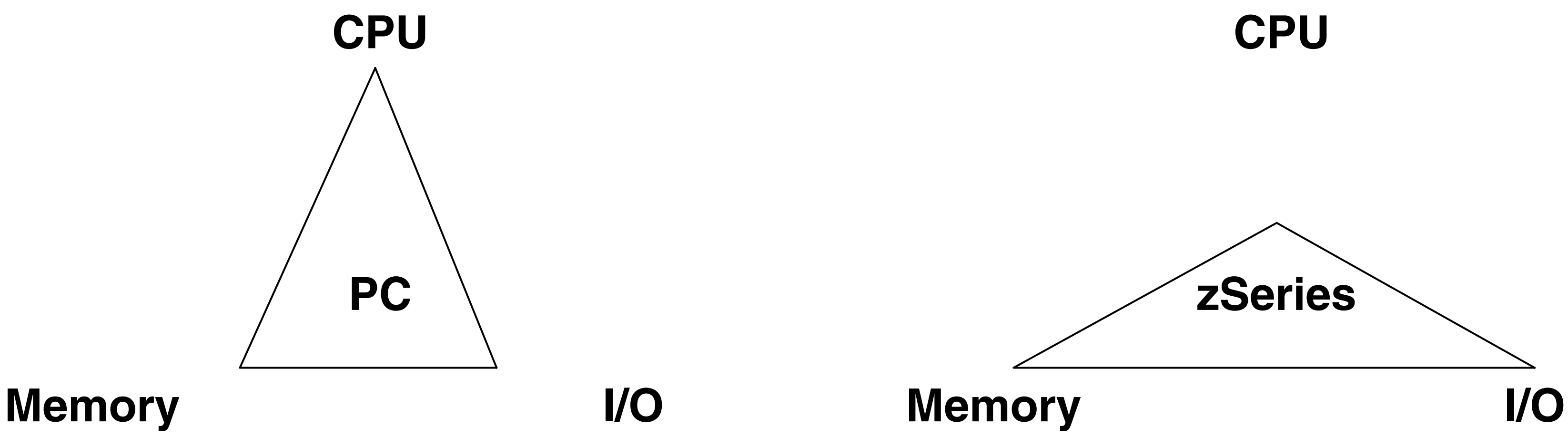
Agenda

- Resource profiles
- Benchmarks
- Performance and tuning
  - CPU resources
  - Memory resources
  - I/O resources



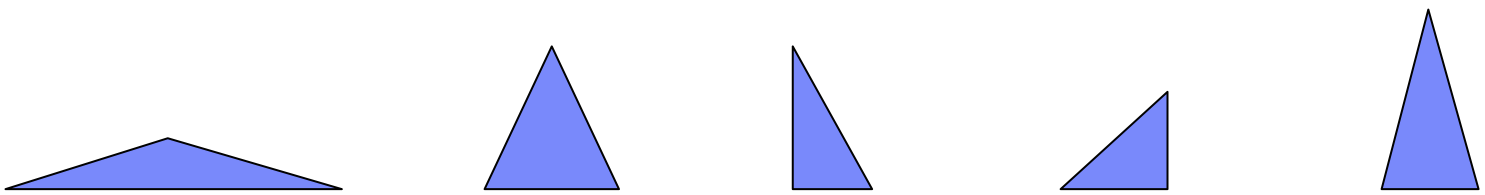
Resource Profiles

- System provides different types of resources
- Capacity for each resource type may be different
- Every architecture has different characteristics
- Some platforms can be extended better than others



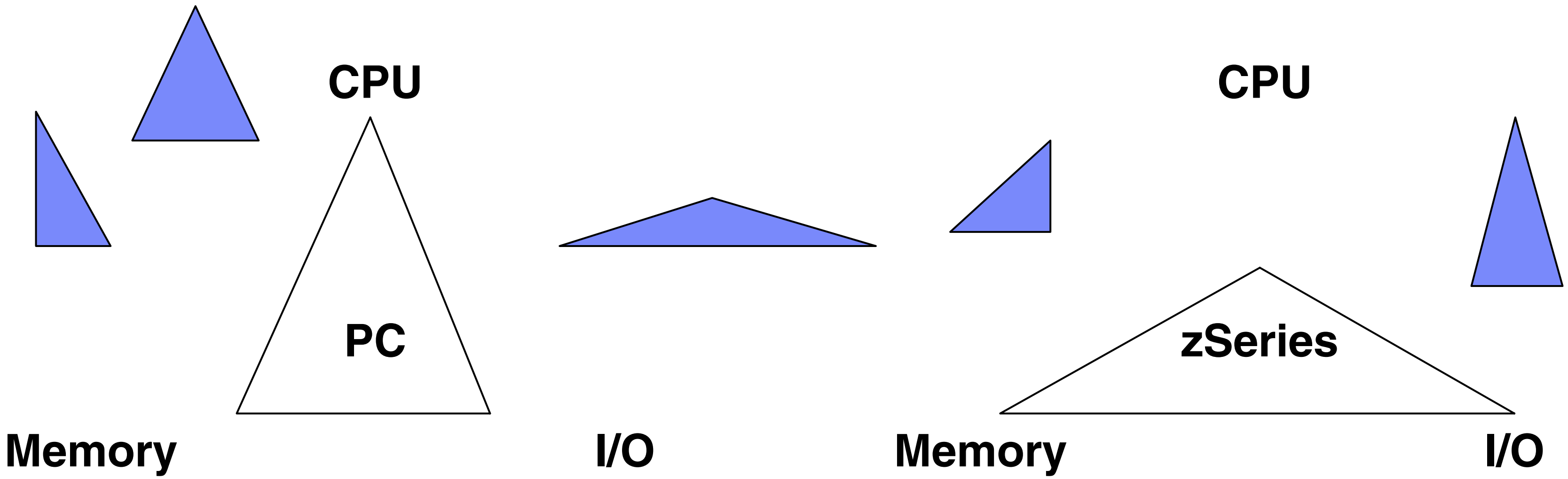
Resource Profiles

- Each application has its specific requirements
  - CPU intensive
  - I/O intensive
  - Memory
- Applications can often be tuned to change the resource profile
  - Exchange one resource for the other
  - Requires knowledge about available resources
  - May be unproductive when the system is upgraded later



Resource Profiles

- Which platform runs the most applications ?
- Not every platform runs every type of application well
- It is not easy to determine the resource profile of an application



Resource Profiles

- Additional resource types
  - Network Bandwidth
  - Availability of software
  - Skilled staff
  - Reliability
  - Money

- The ideal platform requires a mix of resources in right quantity
- Size the system to handle the application requirements

Resource Profiles in Shared Environment

- All applications share the system resources
- One application can use what the others don't need
- It is hard to predict what resources will be available
  - So how will you tune your applications ?
- Provide virtual machines that meet the application requirements
  - Size the virtual machine to deliver peak resource requirements
  - Allocate resources based on actual virtual machine requirements
  - Let VM understand what the resource requirements are
  - Do not waste resources

Benchmarks

- Benchmarks have their own specific resource profile
  - Often exercise only one specific resource type
  - May not be representative for your business applications
- Most benchmarks measure at saturation level
  - Systems behave differently when fully utilized
  - Normal business applications rarely run at saturation level
- Benchmarks concentrate on measuring “top speed”
  - Easy to report
  - Relatively easy to measure
  - Top speed should not be your prime interest

Benchmarks in Shared Environment

- In a shared environment top speed is less relevant
  - Total work achieved is more important than single application speed
  - Unused resources can be used by other applications
- We care more about mileage than top speed
  - Increased efficiency in return for some latency
- Measure the cost per transaction
  - Cost per transaction depends on utilization
  - Run at usage levels comparable to production load
  - Some benchmarks can be used to reproduce the load
  - Find the cheapest way to do the work (with sufficient response)

Benchmarks in Shared Environment

Anyone care to publish Bonnie results ?

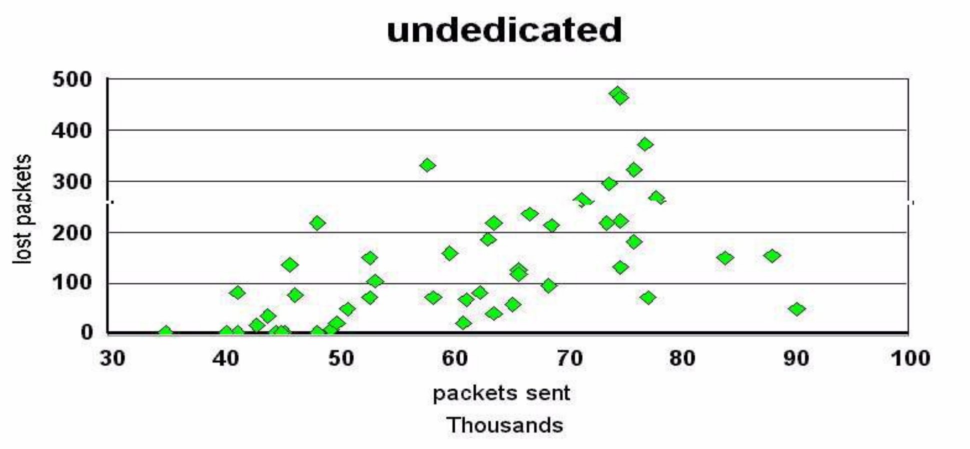
On a virtual machine with single CPU it reports %CPU > 100%

If it fails to tell the time, do we trust the KB/s ?

		-------Sequential Output------------						---Sequential Input----				--Random--	
		-Per Char-		--Block---		-Rewrite--		-Per Char-		--Block---		--Seeks---	
Machine	MB	K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	K/sec	%CPU	/sec	%CPU
	10	6757	97.7	371593	108.9	364283	106.7	6602	99.3	72[illegible]470	**212.0**	53[illegible]80.0	79.6
	20	6840	98.5	356452	69.6	401789	98.1	6623	99.6	740018	72.3	45465.9	79.6
	30	6828	98.7	312369	91.5	246167	72.1	6432	99.3	692156	90.1	42422.3	84.8
	40	6723	98.8	289703	84.9	174891	47.0	6541	99.0	743335	108.9	37362.9	74.7
	50	6690	98.8	300080	87.9	140330	35.6	6498	99.1	732883	100.2	34924.4	69.8
	60	6753	98.8	298957	87.6	225271	55.0	6502	99.3	737902	84.1	33120.0	66.2
	70	6742	98.4	283047	86.9	24793	6.9	6498	99.2	729122	111.9	38043.1	76.1
	80	6786	98.7	79480	24.3	1843	0.7	1793	27.5	8456	2.8	33888.5	76.2
	90	6729	98.7	27031	7.9	1841	0.7	1747	26.9	261546	90.8	23931.0	83.8
	100	6744	98.6	17121	5.4	1820	0.7	1796	27.7	244390	90.7	18255.3	77.6

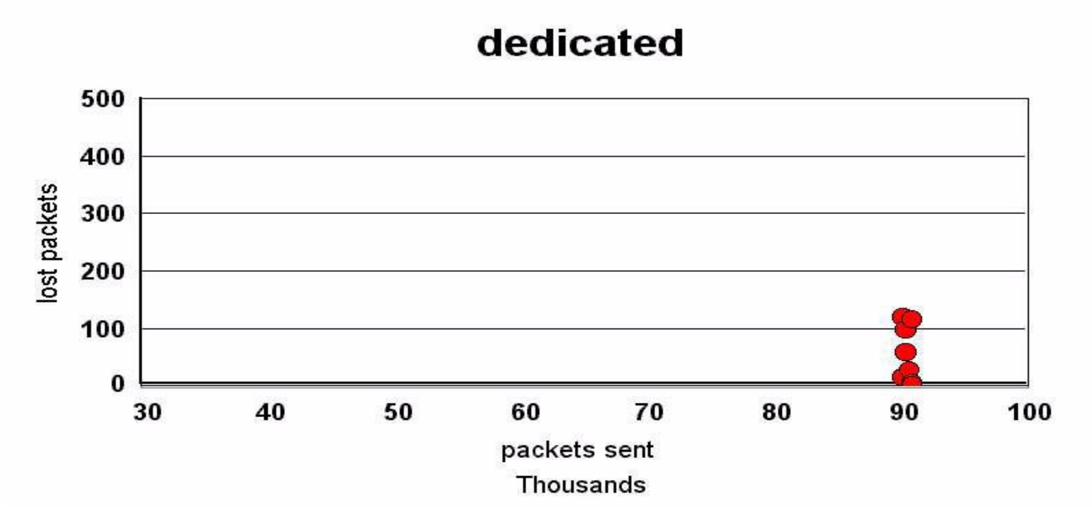
Benchmarks in Shared Environment

- CPU sharing may even break the function of the benchmark
- Running ‘iperf’ to measure bandwidth between two virtual machines
  - Drives the connection with a specified bandwidth
  - Exact rate achieved by “busy wait” between sending packets
  - Adaptive algorithm to adjust the timing loop
  - Noticed erratic packet loss during the tests

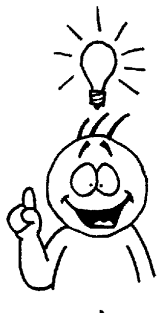


Benchmarks in Shared Environment

- Adaptive algorithm was confused by time slices
  - Instead of constant rate, packets were sent in bursts
  - High packet rates during bursts caused OSA to drop packets
  - Changes to the algorithm fixed most of the problem



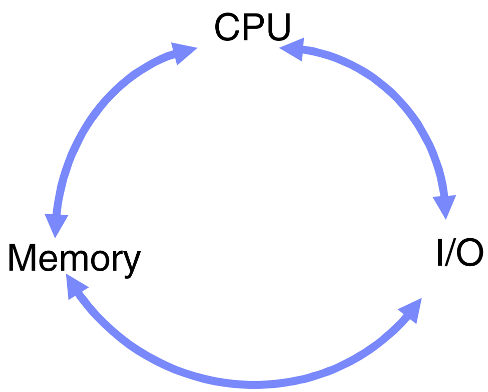
Benchmarks in Shared Environment



- Make sure you understand what you measure
- Find additional places to measure and do cross checks
  - VM provides a lot of instrumentation in Monitor records
  - When necessary capture readings from control blocks
  - This is often a lot of work
- Linux observation of CPU time is wrong
  - Due to use of interval timer rather than CPU timer
  - Affects all instrumentation on Linux
  - Linux measurements can be orders of magnitude off
  - VM can provide numbers to validate (and even correct) Linux numbers

Performance and Tuning

- Look at virtualization issues for each resource type
- Checklist
  - ✓ Size the virtual machine to deliver peak resource requirements
  - ✓ Allocate resources based on actual virtual machine requirements
  - ✓ Let VM understand what the resource requirements are
  - ✓ Do not waste resources



CPU Resources

CPU Resource Management

Scheduler	decides which virtual machines get CPU resources
Dispatcher	hands out time slices to virtual machines

Interactive versus batch

- Interactive virtual machines get short time slices often and soon
- Long-running tasks get longer time slices less often

Balance between

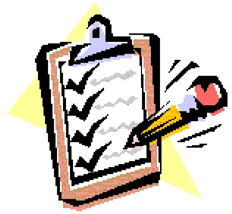
- Enough virtual machines around for work to keep the CPU busy
- Not more virtual machines in memory than we can run at a time

CPU Resources

- Linux systems “out of the box” have large working sets
- Without the “timer patch” it looks like one single long transaction
- Default z/VM tuning settings are very conservative
  - Total working set of all “in queue” virtual machines is kept low
  - Linux virtual machines end up in the eligible list (no fair scheduling)
- Quick and dirty solution: remove the safety pins
  - SRM settings to infinite

> *Our most recent Linux guest addition has pushed us into a storage constrained condition, where the guest typically pages at several hundred per second.*
> *We're looking at adding more storage, but in the mean time, we're wondering if there's anything useful we can do in the way of allocating XStore, additional paging packs, etc.*
>
> Linux-390 mailing list

CPU Resources - Checklist

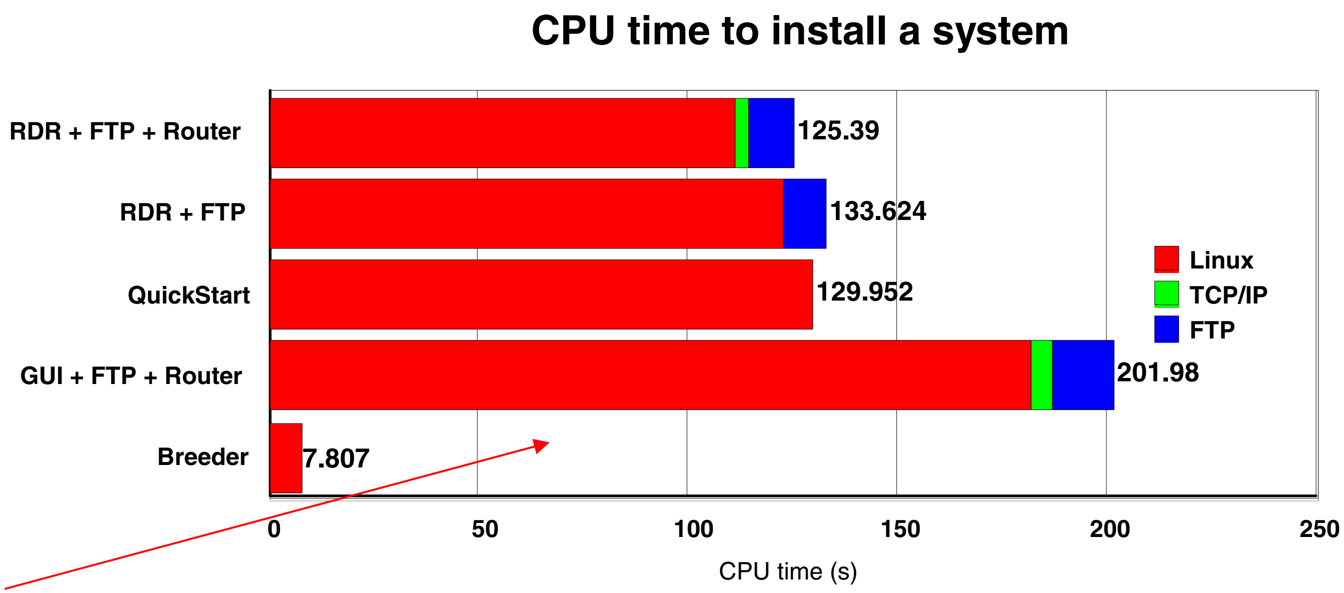


- Size the virtual machines to deliver peak resource requirements
  - In most situations a single virtual CPU is enough
- Allocate resources based on actual virtual machine requirements
  - SET SHARE can be used to further limit resources
  - VM Resource Manager (VMRM) for setting workload requirements
- Let VM understand what the resource requirements are
  - Use the “on demand” timer
  - Do not mix different applications in a single virtual machine
  - Let VM run a lot of small things rather than a few big things
- Do not waste resources
  - Avoid CPU intensive work if you can (e.g. compression and encryption)
  - Do things once, rather than for each system separately

Cost of Installing Systems

- CPU time used for installing new systems is an issue

  Resources are taken away from business applications



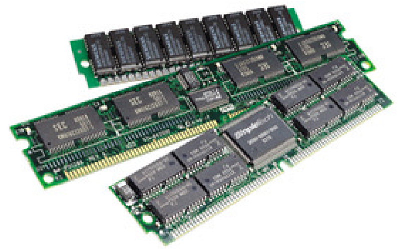
Worth several months running an idle server

Note: Breeder is a home grown Linux cloning procedure

Memory Resources

- z/VM needs enough virtual machines around to keep the CPU busy
  - Be able to service the virtual machines quick enough
- Linux virtual machines require a lot of memory
- Idle Linux virtual machines use little CPU resources (but often)
- This is a challenge for z/VM Memory Management
  - Reducing Linux memory requirements is the only thing that scales

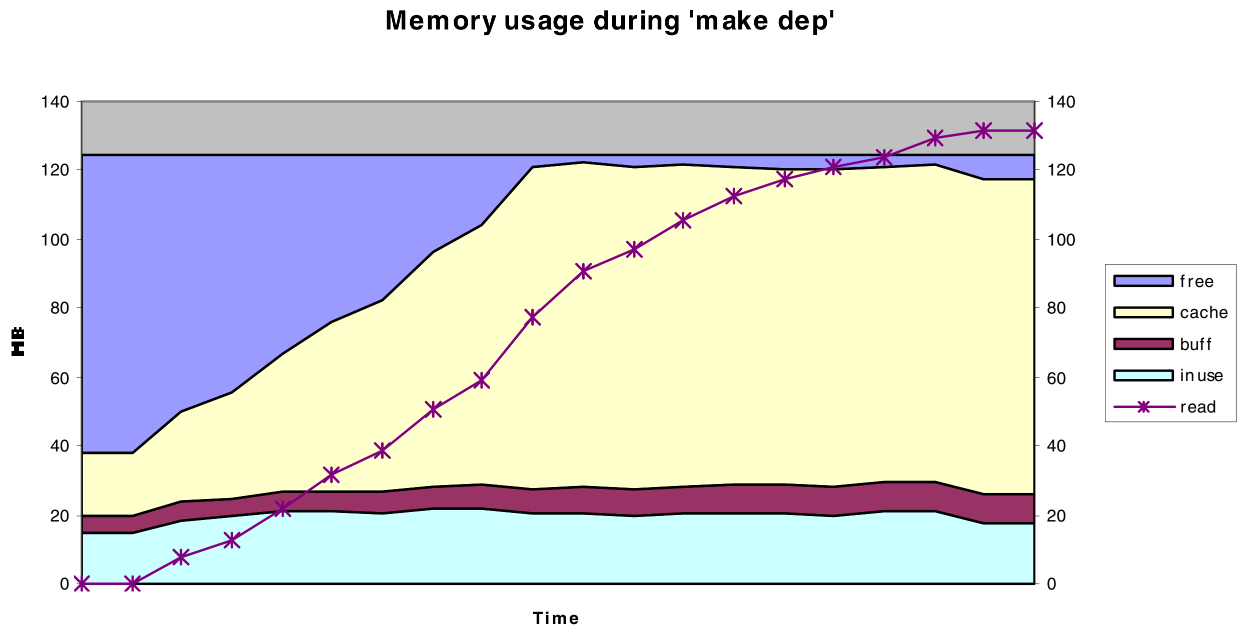
Linux Memory Management



- Design comes from the PC environment
  - Memory is very cheap
  - I/O is slow
- Linux tries to avoid I/O when possible by aggressive caching of data
  - Adding more memory makes it faster
- Too little memory causes swapping (with poor I/O, swapping is bad)
- Given enough time, Linux will use all memory that you give
- Systems are sized after the owner's badge

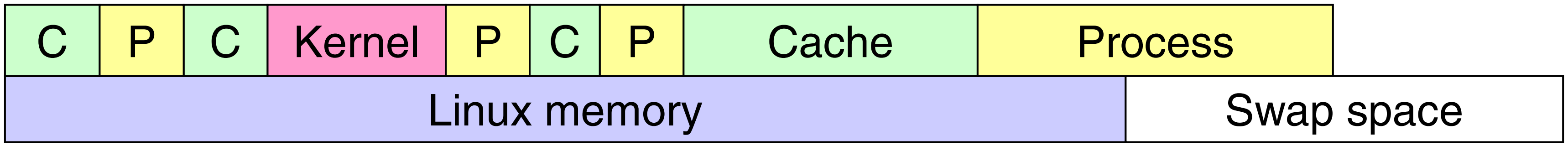
Linux Memory Management

- Excess memory is used for page cache
- Page cache is retained after the program terminates
- Makes an I/O intensive program into a memory hog
- Data is cached just in case it is needed again



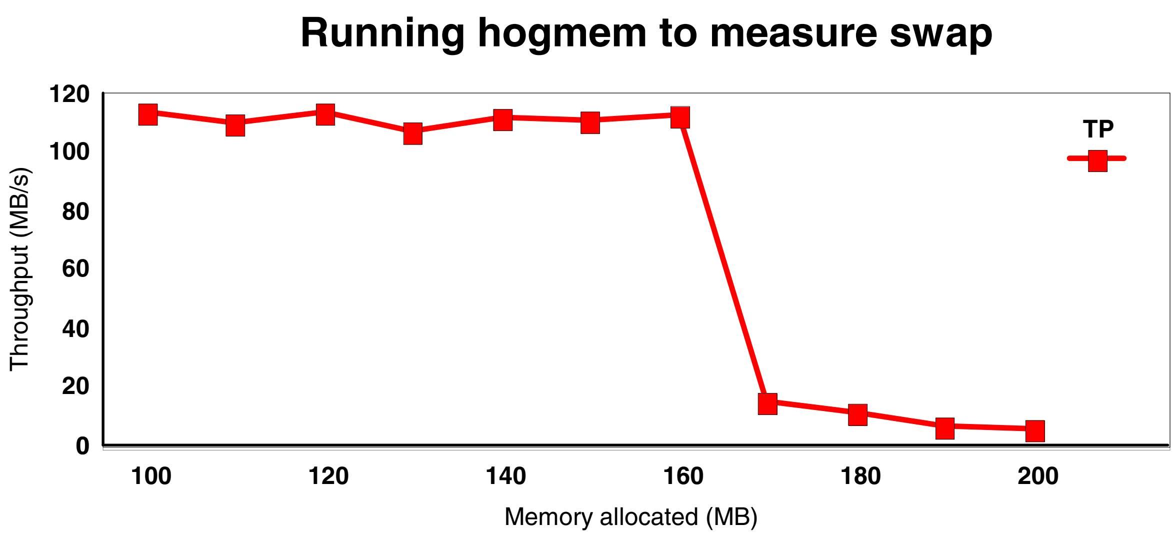
Linux Memory Management

- Swap space is like an extension of memory
  - Slower, cheaper, larger
- Processes are isolated from the details – virtual memory
  - Each process believes to have its own memory all available
- Memory management (page replacement strategy)
  - May prefer frequently used data from disk over process memory
  - Exact algorithms are not widely understood
  - Tuning parameters may be architecture dependent



Linux Memory Management

- Running ‘hogmem’ to measure swapping
  - Allocate virtual memory and run a loop reading and writing memory
- When free memory is not sufficient, swapping makes it very slow
- With such a penalty for swapping, you want to have a lot of memory



Sizing Linux Virtual Machines

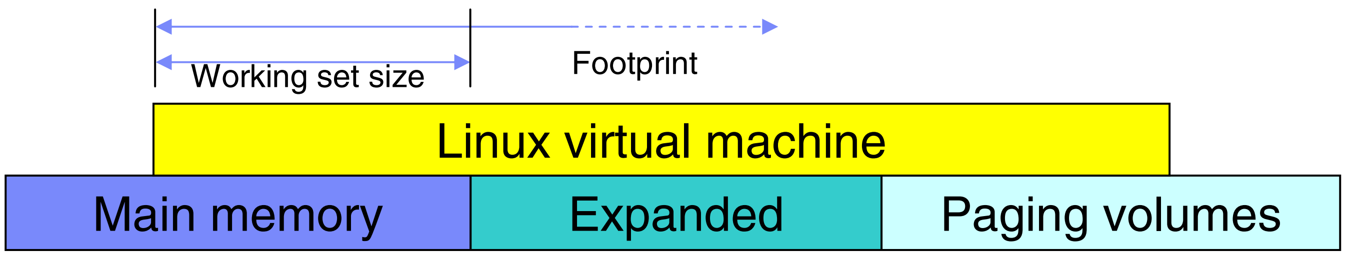
How large do we make the Linux Virtual Machines ?

- Don’t look at the current machine configuration
  - Memory in discrete servers is installed without looking at the application
- Don’t ask the customer
  - The next machine is at least twice as big as the current one
  - Asking more will make it faster (when in doubt, ask more)
- You really have to measure application requirements
  - Unfortunately people are not used to measure this
  - A memory-bound application may turn into an I/O-bound application

Sizing Linux Virtual Machines

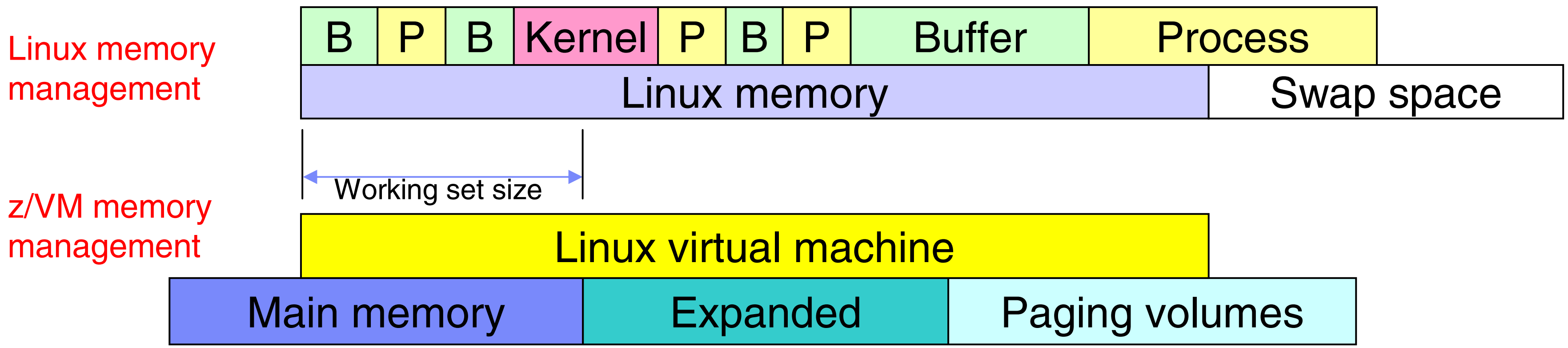
- It may hurt when you make the virtual machine too big
- z/VM uses virtual memory for virtual machines
- Paging subsystem keeps all virtual memory for all virtual machines
- Only a portion of virtual machine memory can reside in memory

  When the Linux footprint is larger than what can stay in memory, frequent page faults will hurt performance



Sizing Linux Virtual Machines

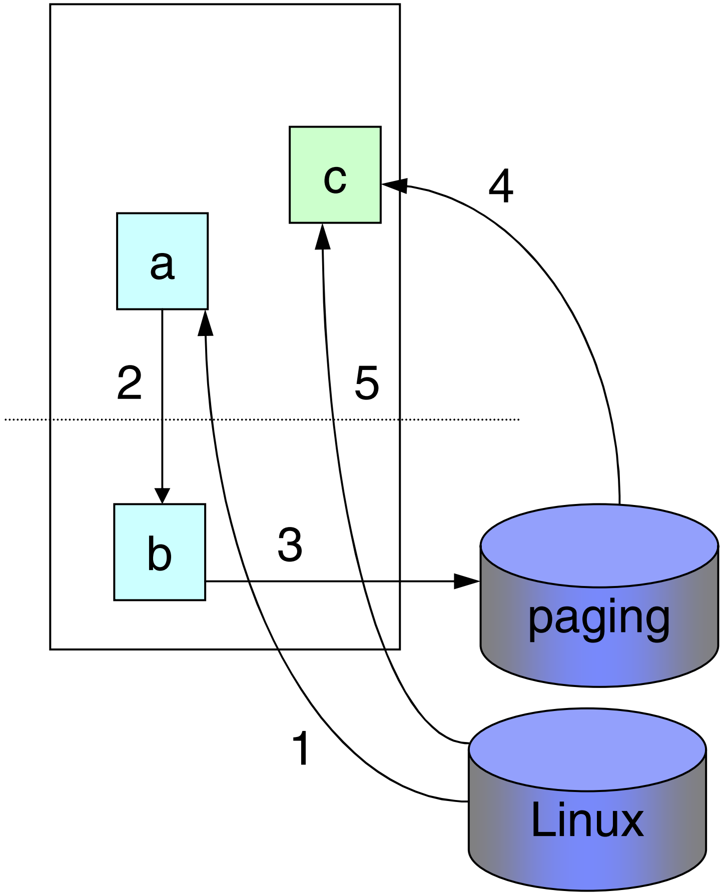
- Two layers of memory management on top of each other
  - Local optimization in Linux
  - Global optimization in z/VM
- Reference pattern in page cache makes this even worse



Page Cache Paging Problem

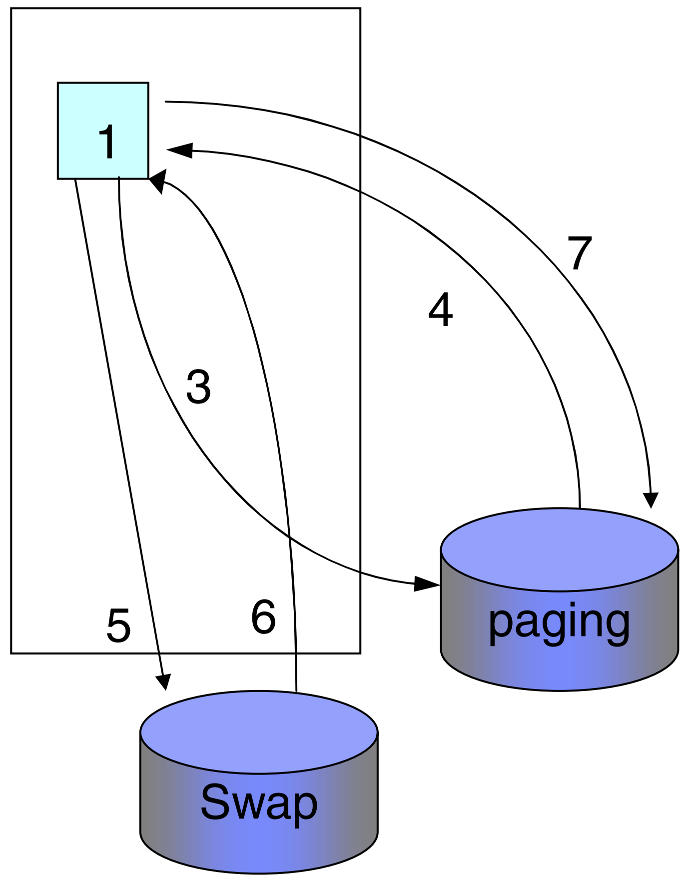
1. Linux application wants to read, new page (a) is taken to read data in. Application continues to read new data in, in new fresh pages.
2. Working set grows and VM moves least recently used page to expanded memory page frame (b).
3. After more time without reference, contents of the page moves to paging volumes.
4. Application reads more data and Linux ran out of unused pages. Linux will use the oldest one which is paged out. The reference causes VM to allocate a frame (c) and page it in.
5. Old contents is disposed, and replaced by new data read by Linux into the page frame.

- Net result: local optimization to reduce I/O resulted in 3 I/O operations for one block of data.



Double Paging Problem

1. Linux allocates page for process virtual memory by referencing it.
2. Other work continues using other pages of Linux memory.
3. VM decides to page out the Linux page that holds the process data.
4. Linux decides to re-use that page, so starts an I/O to swap the page out. The reference causes VM to page it in first.
5. Contents can now be swapped out.
6. Data of other process can be swapped in by Linux.
7. After some time VM will page it out again.



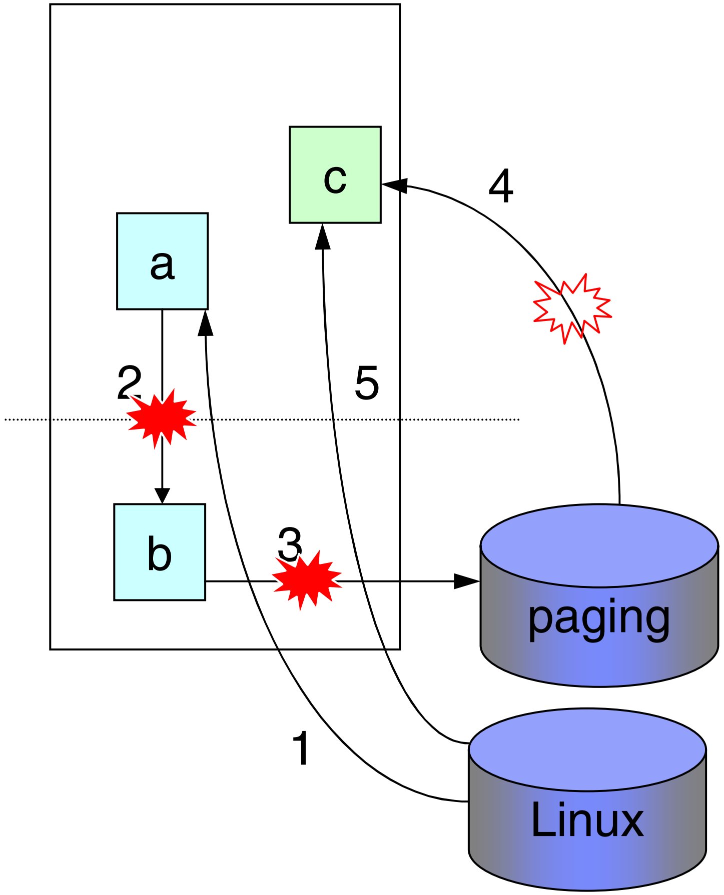
Double paging is a concern for every virtual machine that implements its own virtual memory. Solution requires interaction between the two layers.

Collaborative Memory Management

- Kernel thread that can take memory away from Linux and return to z/VM
  - Triggered by z/VM memory management
  - Pages returned to Linux
    - trigger by external controls
    - timed release

- Can shrink Linux footprint to avoid z/VM having to page it out to expanded (2)
- Can make Linux drop pages that were already out in expanded storage (3)

- Maybe also prevent page-in by z/VM when Linux frees the page (4)



Shared Kernel in NSS

- Linux can IPL from Named Saved System NSS
  - CONFIG_SHARED_KERNEL=yes
- Virtual machines share (R/O) the **same** physical page
  - Reduces total memory requirements (about 2 MB per Linux image)
  - No need to tweak each kernel separately
  - NSS pages less likely to be paged out by VM
  - Faster and cheaper IPL of Linux images (server on demand)
- Room for improvement
  - Further reduction of the EW (Exclusive Write) portion of the NSS

Currently not supported!

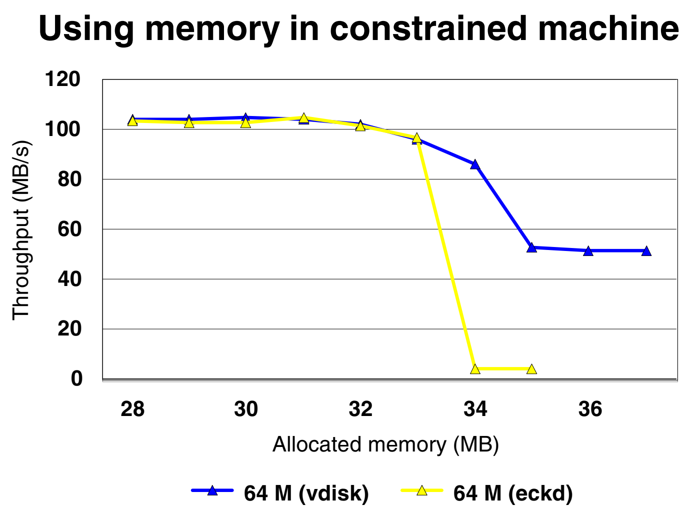
Sizing Linux Virtual Machines

- Optimal virtual machine size is determined by many things
  - Application requirements profile (cyclic patterns and peaks)
  - Resources available on VM (other workload)

- Large virtual machine causes VM page faults
  - If you need to do I/O, it can be better done by VM
  - Pseudo page fault handling allows Linux to continue after page fault
- Small virtual machine causes Linux swapping
  - Occasional swapping is not necessarily bad
  - Performance penalty can be reduced by faster swap device

Linux Swap to VDISK

- VDISK is VM emulated disk in storage
  - Data resides in paging subsystem like virtual machine storage
  - Treat it like memory, not like a disk
- With the Diagnose driver very high swap rates are possible
  - Reduces the performance penalty
  - Allows virtual machines to be made a bit more tight



Linux Swap to VDISK

- Excessive use of VDISK is not recommended
  - z/VM algorithms were designed for small shared disks (VSE lock disk)

- VDISK pages are allocated on first reference
- Oldest pages will migrate to expanded storage and paging volumes
- Linux swap slot allocation is designed to minimize seek times
  - Given enough time, all swap slots will be used
  - Most blocks will be read back in only once, swap slot is freed again
  - VM continues to keep the contents longer than Linux needs it
  - Problems similar to the Buffer Cache paging problem
  - Use multiple VDISK swap devices with different priority
    - Causes Linux to re-use freed swap slots again

Memory Resources

- Size the virtual machines to deliver peak resource requirements
  - Large enough so that you're not constantly swapping under high load
  - As large as you can afford to run
- Allocate resources based on actual virtual machine requirements
  - Use the "on demand" timer (allows VM to trim the working set)
- Let VM understand what the resource requirements are
  - Use multiple VDISK swap devices when necessary
  - Do not mix different applications in a single virtual machine
- Do not waste resources
  - Avoid excessive buffer sizes in applications
  - Do not start more threads than you need
  - Do things once, rather than for each system separately

Timer Tick

- Linux on Intel uses a 100 Hz timer tick to schedule events
  - Recent kernel versions implement the "on demand timer" for S/390
- Some activity every 10 mS suggests VM the guest is active
  - Misleading VM when forced to take resources away
  - Memory will be taken at random from virtual machines
  - Idle Linux images continue to retain a large working set
- Traditionally 300 mS inactive causes a "queue drop"
  - Lowering this value is not practical (increases scheduler overhead)
  - With "on demand timer" ticks can go down to 1 per second
  - Many applications schedule frequent timer interrupts

Timer Tick

What if your Linux virtual machine still stays in queue?

- Redbook shows the process to check for timer interrupts
- Broken kernel code, device drivers and other poor design
  - Bug in ReiserFS with Linux-2.4.7 causes an interrupt every 50 mS
  - Until Linux-2.4.18, kswapd wakes up every second for reference counts
  - Most journaling file systems flush buffers every 5 seconds
  - Some applications (e.g. Samba) wake up to check configuration changes
- The “are you there” pings (e.g. Big Brother, heartbeat)
- Performance reporting on idle Linux images
- Running services that are not needed (e.g. nscd)

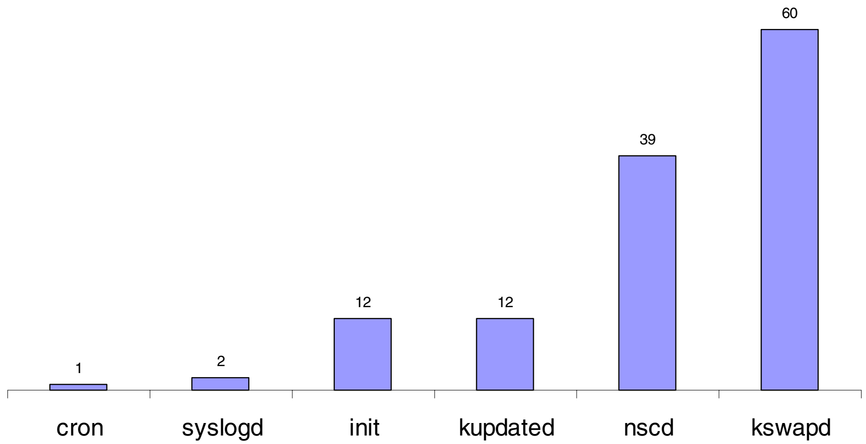
- VM63282 – Guests not dropped from Q3
  - Applies to Linux guests with CTC and QDIO devices

Timer Tick

- Breaking down timer ticks in a Linux 2.4.7 system

- kswapd – fixed in 2.4.18
- nscd – not need in this case
- init – caused by sloppy application

SuSE 7.2 timer ticks per minute



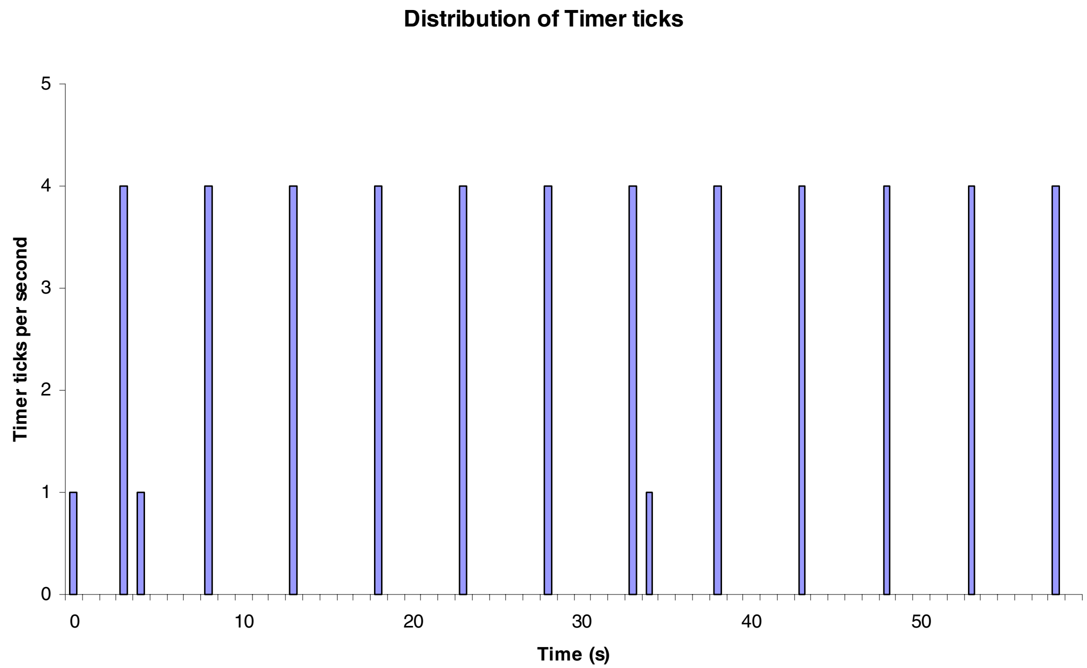
Idle Linux guest can be very cheap:
- 11 mS CPU time per minute
- 0.01% of a CPU
- 5,000 idle Linux guests per CPU

	Before	After
Ticks per minute	126	28
CPU usage	0.1%	0.01%

Timer Tick

- 50 ticks per minute is not always one per second
- Most intervals appear to be aligned
- Results in longer periods with no activity

    Allows VM to spot idle servers and allocate memory better

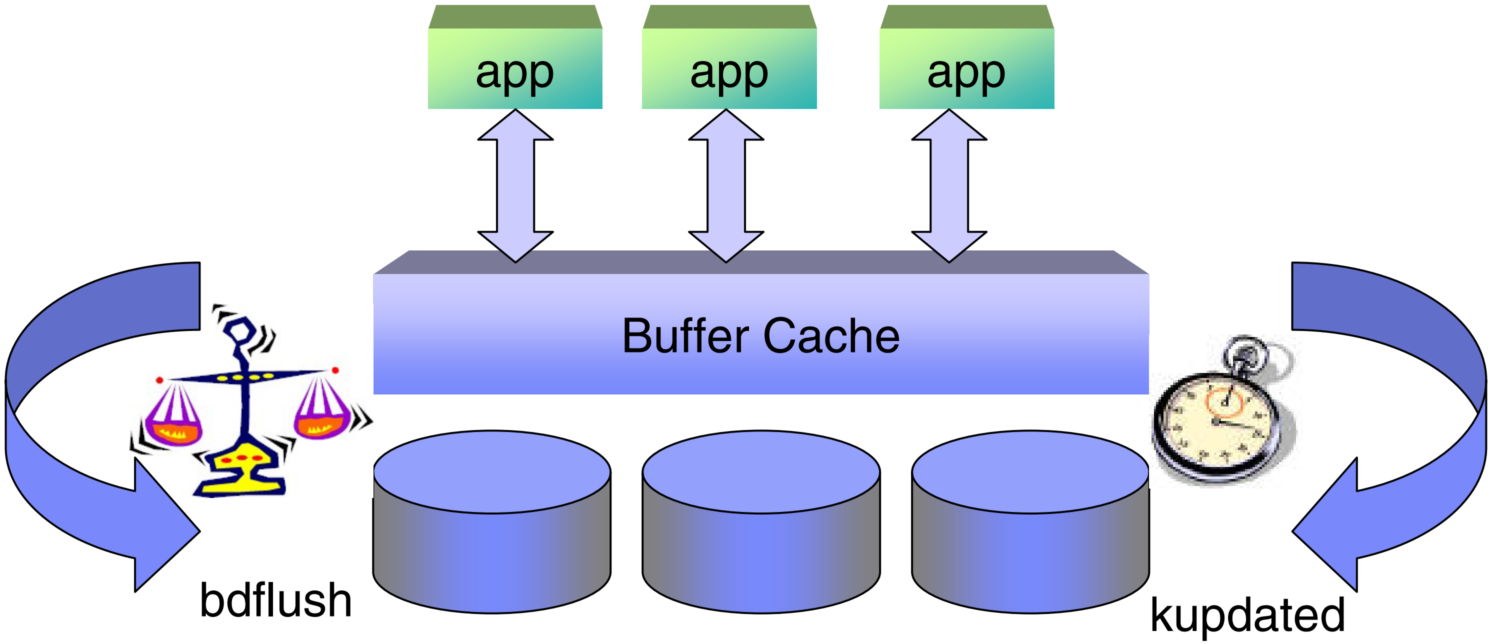


I/O Resources

- Main focus is on disk I/O
- Linux design comes from the PC world
  - Where memory is cheap
  - Disk I/O is slow
  - Disk I/O may hurt interactive performance
- Aggressive caching to avoid I/O where possible
  - Buffer cache is integral part of the memory management
  - Large read-ahead (if we do I/O, let's do all we may ever need)
  - Lazy write

Lazy Write

- Buffer cache between application and disk
- The “dirty blocks” are flushed out asynchronously
- Makes it hard to do I/O benchmarks

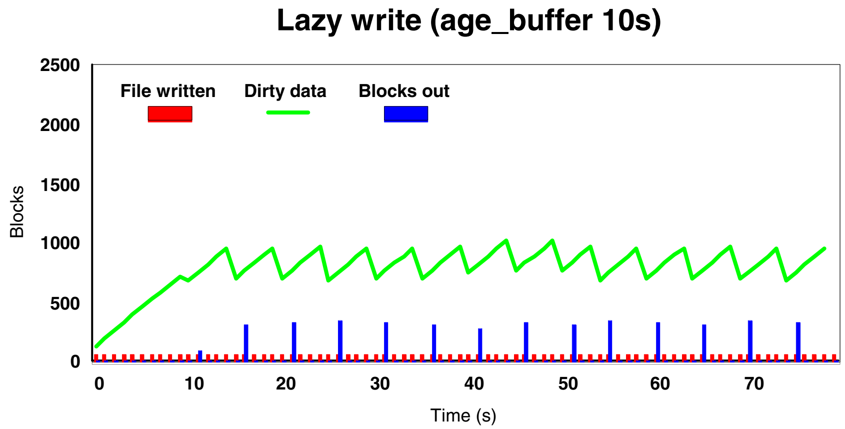


Lazy Write

- The bdflush parameters can be changed through /proc
- The amount of “dirty” data can be a concern for integrity
- Without test cases it is not easy to configure good parameters

- Asynchronous I/O allows Linux to do long channel programs
  - Will lock out others for a long time

- Very attractive for temporary files



I/O Resources

- I/O bandwidth can be increased by using multiple (real) devices
  - LVM with striping
  - Software RAID
  - LVM with PAV
- Not always necessary to configure for maximum bandwidth
  - Effect of striping depends on application, potentially makes it worse
  - Striped volumes can not be extended
  - May not want a Linux virtual machine to monopolize the system
- VM mini disk cache often not attractive for private R/W data

Conclusion

- General recommendations are very hard to give
  - Tuning suggestions from other platforms do not always apply
- z/VM systems with many idle Linux systems run out of memory
  - Use shared kernel
  - Run with on-demand timer
  - Reduce virtual machine size where possible
- Improvements to be expected with z/VM 4.4
  - Virtual switch
  - Memory usage