Advances in Memory Management in a Virtual Environment

Speaker: Dan Magenheimer
Oracle Corporation

Linux Plumbers Conference 2010

Agenda

- Motivation, “The Problem” and the Challenge
- Memory Optimization Solutions in a Virtual Environment
- Transcendent Memory (“tmem”) Overview
- Self-ballooning + Tmem Performance Analysis

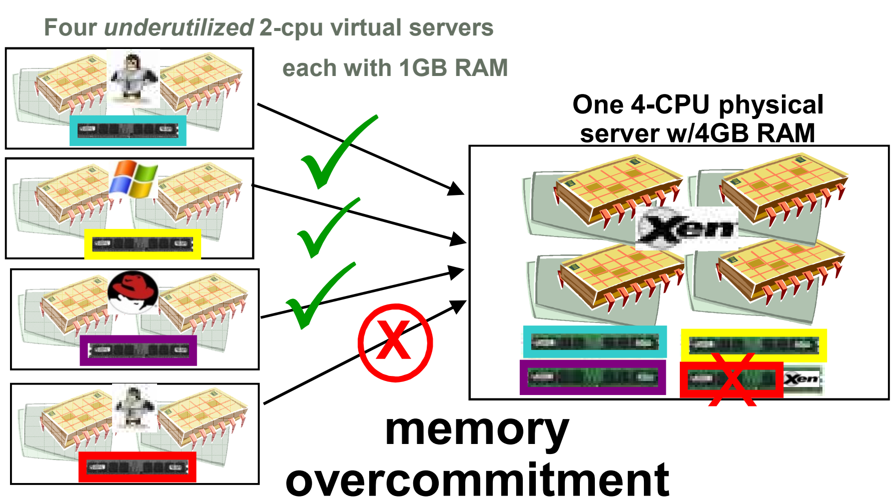
NOTE: FOCUS IS ON Xen ***AND*** KVM

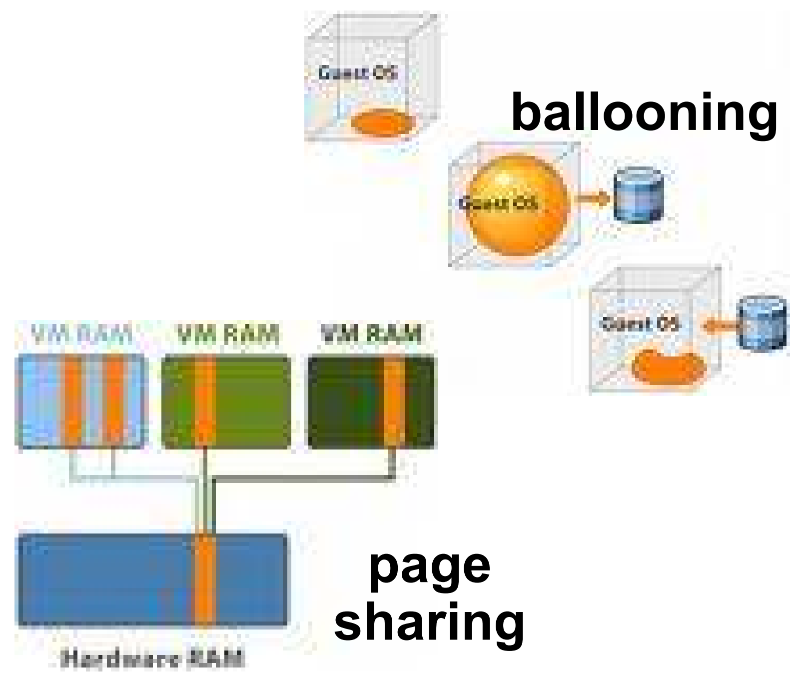
NOT ON: vmware Microsoft Hyper-V Server 2008

ORACLE

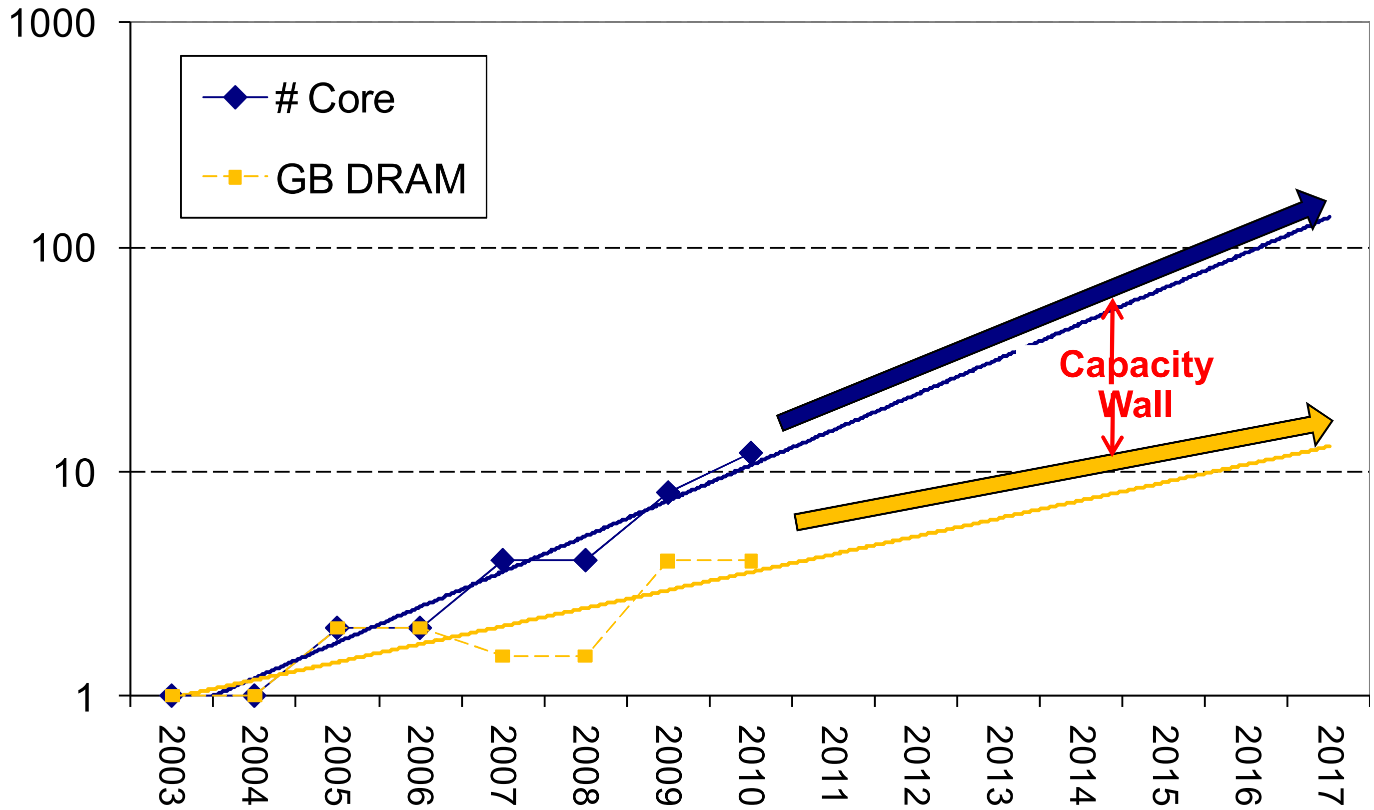
Motivation

- **Memory** is increasingly becoming a **bottleneck** in **virtualized** system
- Existing mechanisms have major holes





More motivation: The memory capacity wall



⇨ Memory capacity per core drop ~30% every 2 years

Source: Disaggregated Memory for Expansion and Sharing in Blade Server
http://isca09.cs.columbia.edu/pres/24.pptx

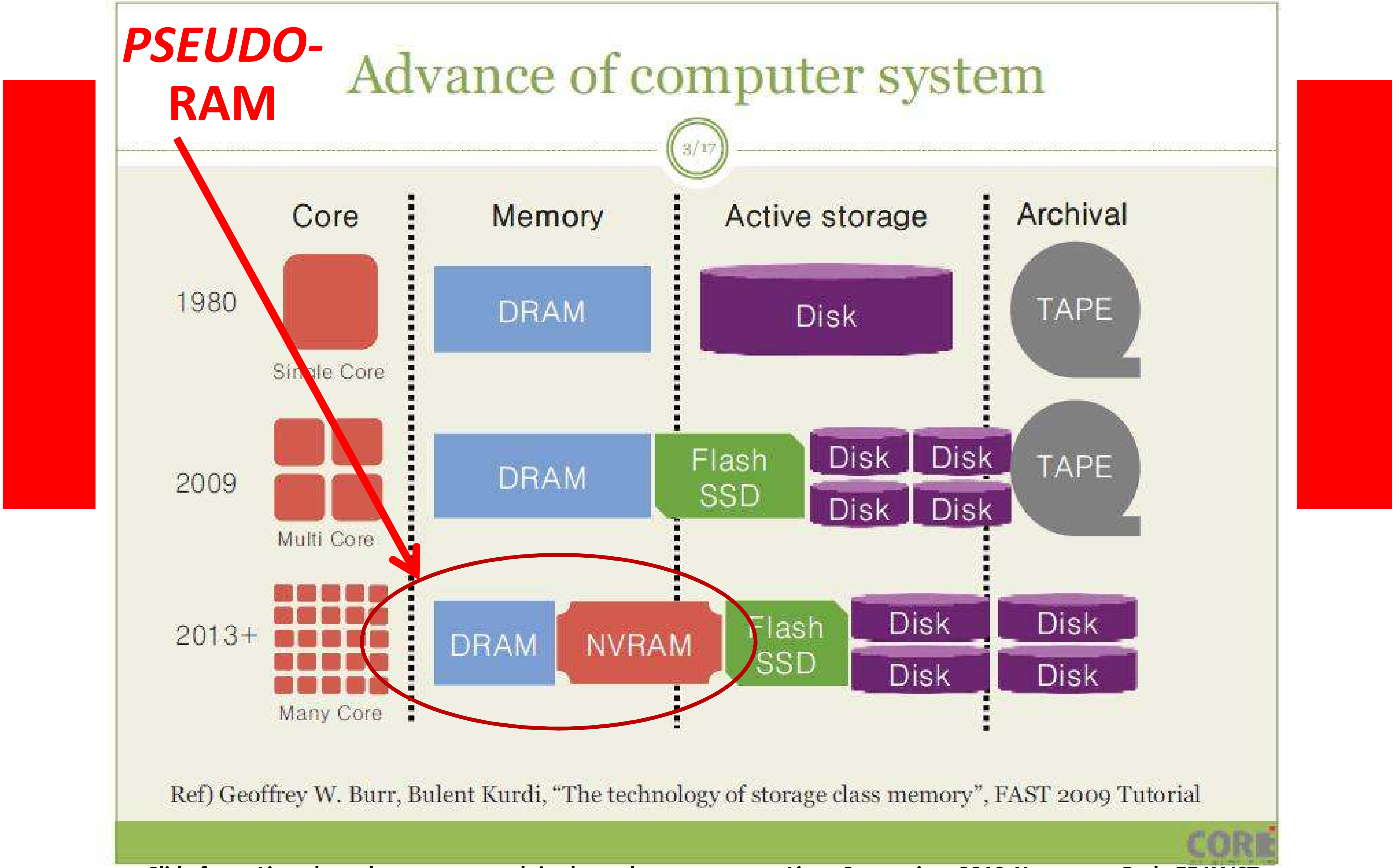
ORACLE

More motivation: Energy Savings

"…several studies show the contribution of memory to the total cost and power consumption of future systems increasing from its current value of about 25%…"

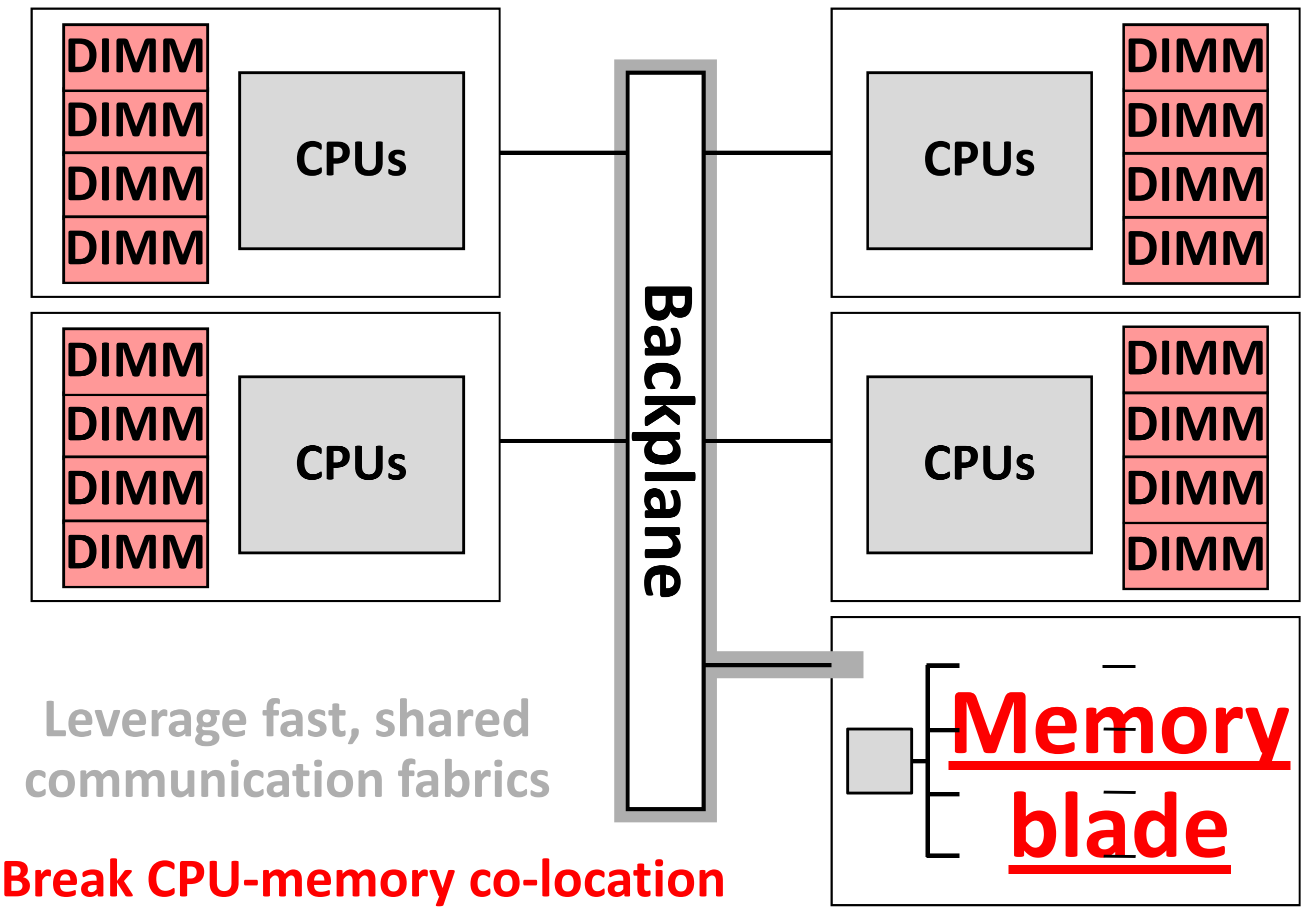
Google Data Center in Belgium

Source: Disaggregated Memory Architectures for Blade Servers, Kevin Lim, Univ Michigan, PhD Thesis



Slide from: Linux kernel support to exploit phase change memory, *Linux Symposium 2010*, Youngwoo Park, EE KAIST

Disaggregated memory concept



Leverage fast, shared communication fabrics

⇨ **Break CPU-memory co-location**

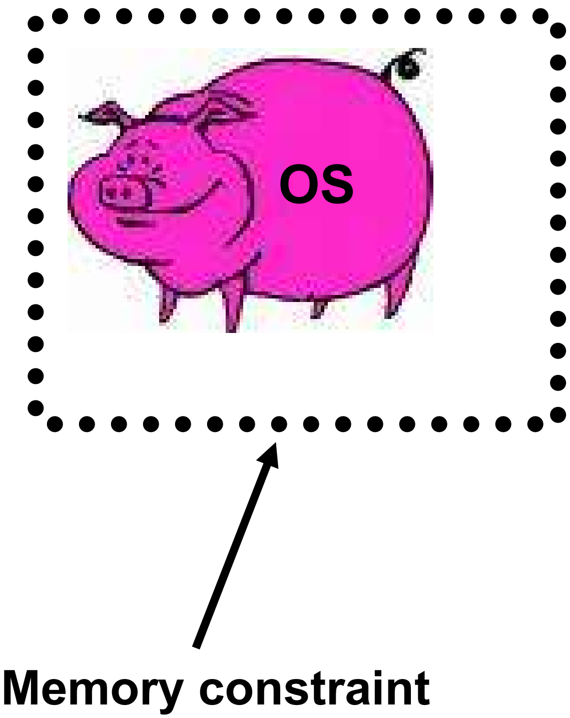
Source: Disaggregated Memory for Expansion and Sharing in Blade Server

http://isca09.cs.columbia.edu/pres/24.pptx

"HARD TO PREDICT THE FUTURE IS" -- Yoda

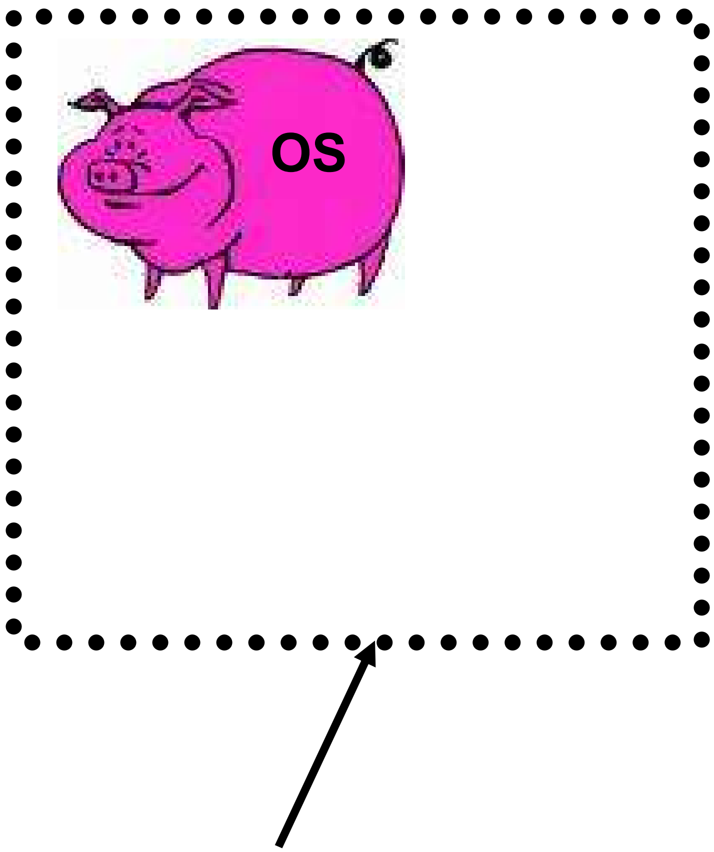
[pictures removed for posted version to get PDF under 2MB]

The “Meat” of the Problem



- Operating systems are memory hogs!

The “Meat” of the Problem



New larger memory constraint

- Operating systems are memory hogs!

If you give an operating system more memory…..

ORACLE

The “Meat” of the Problem

Memory constraint

- Operating systems are memory hogs!

If you give an OS more memory

…it uses up any memory you give it!

The Virtualized Physical Memory Resource Optimization Challenge

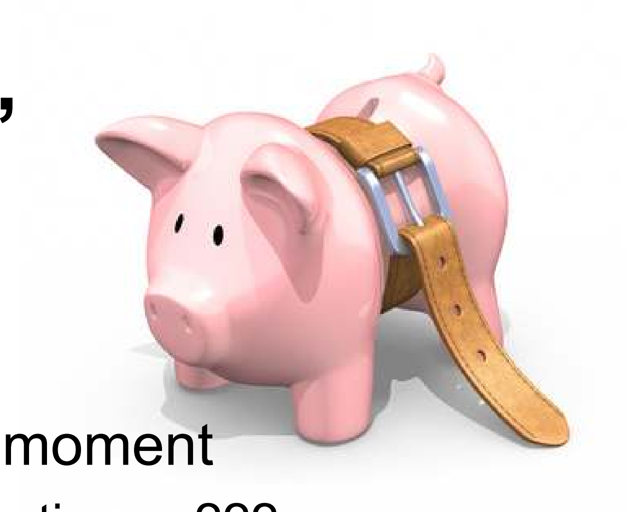
Optimize, across time, the distribution of RAM ***(and future "pseudo-RAM"?)*** among a maximal set of virtual machines by:

- measuring the current and future memory need of each running VM and
- reclaiming memory from those VMs that have an excess of memory and either:
  - providing it to VMs that need more memory or
  - using it to provision additional new VMs.
- *without* suffering a significant performance penalty

First step… put those pigs on a diet?

ORACLE

OS Memory “Asceticism”



ASSUME that it is “a good thing” for the an OS to use as little RAM as possible at any given moment

- motivation may be economic or power or virtualization or ???

SUPPOSE there is a *mechanism* for the OS to ***surrender*** RAM that it doesn’t need at this moment, so it can “pursue goodness”

SUPPOSE there is a *mechanism* for the OS *to* ***ask for*** and obtain a page (or more) of RAM when it ***needs*** more RAM than it currently has

THEN… HOW does the OS decide how much RAM it “**needs**”?

as-cet-i-cism, n. *1.* extreme self-denial and austerity; rigorous self-discipline and active restraint; renunciation of material comforts so as to achieve a *higher state*

ORACLE

Agenda

- Motivation and Challenge
- Memory Optimization Solutions in a Virtual Environment
- Transcendent Memory ("tmem") Overview
- Self-ballooning + Tmem Performance Analysis

VMM Physical Memory Management Solutions

Solution Set A: Just let each guest hog all memory given to it, *but…*

Solution Set B: Guest memory is dynamically adjustable *…somehow*

Solution Set C: Total guest memory is dynamically load-balanced across all guests …*using* some *policy*

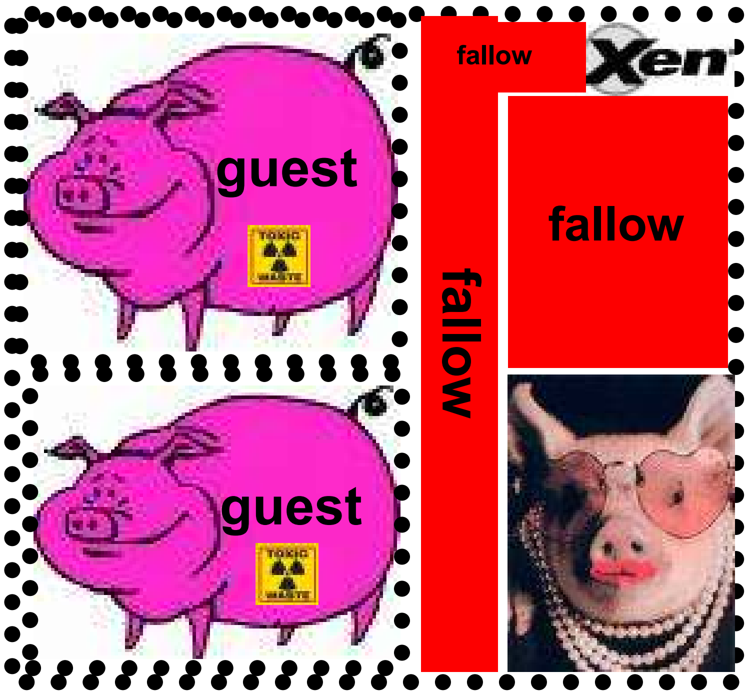
Solution Set D: Host-provided “compensation” … *to correct for insufficiently omniscient policy*

VMM Physical Memory Management
Solution Set A

Solution Set A: Each guest hogs all memory given to it

- Partitioning
- Host swapping
- Transparent page sharing

VMM Physical Memory Management
Partitioning (= NO overcommitment)

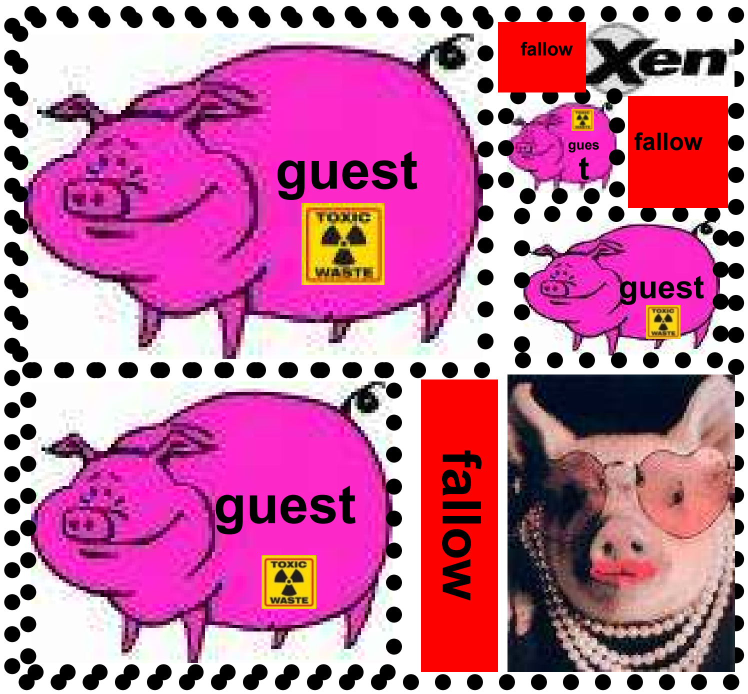


- By default, Xen ***partitions*** memory
  - Xen memory
  - dom0 memory
  - guest 1 memory
  - guest 2 memory
  - whatever's left over: *"fallow"* memory

fallow, *adj., land left without a crop for one or more years*

ORACLE

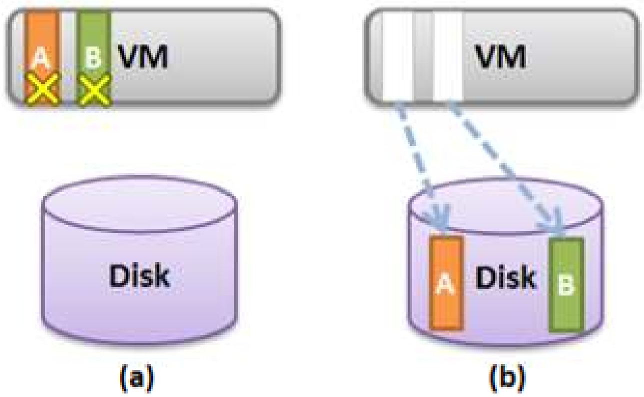
VMM Physical Memory Management
Partitioning (= NO overcommitment)



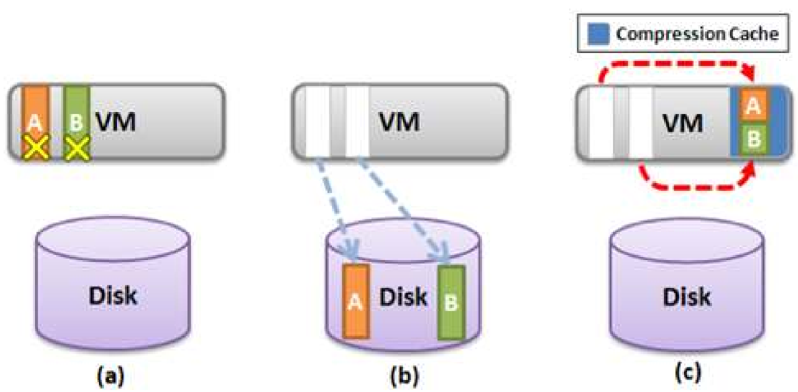
- Xen partitions memory among more guests
  - Xen memory
  - dom0 memory
  - guest 1 memory
  - guest 2 memory
  - guest 3…
- BUT still *fallow memory* leftover

fallow, *adj., land left without a crop for one or more years*

VMM Physical Memory Management
Host Swapping (*SLOW* overcommitment)



- Any page may be either in RAM or on disk
- Tricks like compression can reduce disk writes
- But still…

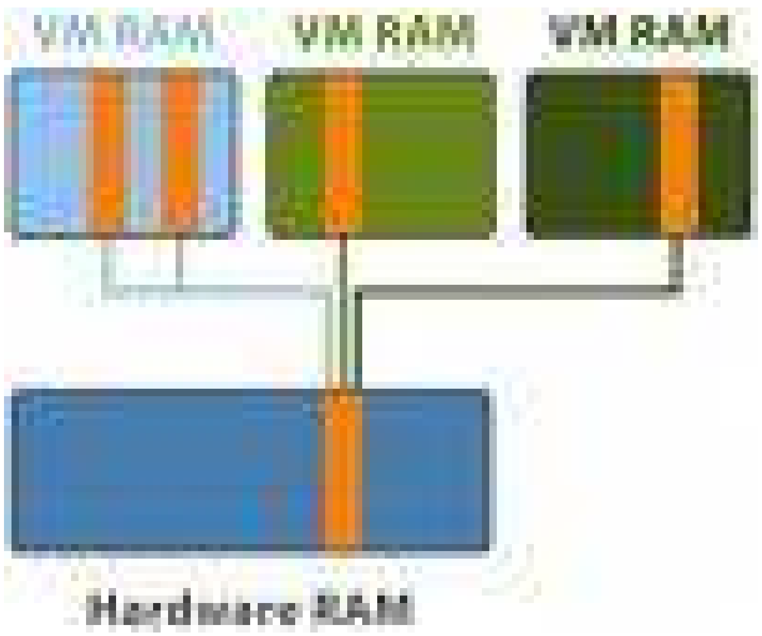


Storage Technology	Response time (ns)
Typical disk (seek)	8000000
DDR3-1600	5

KVM Xen

ORACLE

VMM Physical Memory Management
Transparent Page Sharing (aka "KSM")
("*FAUX*" overcommitment)



- Keep one copy of identical pages
- Scan (huge swaths of memory) periodically for matches
- BUT…
  - very workload dependent
  - sometimes causes host swapping (resulting in unpredictable performance)
  - poor match for 2MB pages

KVM Xen

VMM Physical Memory Management
Solution Set A Summary

Solution Set A: Each guest hogs all memory given to it

- Partitioning
  - NO overcommitment
- Host swapping
  - SLOW overcommitment
    - like living in a swapstorm
- Transparent page sharing
  - “FAUX” (fake) overcommitment, but
    - advantage is very workload dependent
    - inconsistent, variable performance, “cliffs”
    - “semantic gap” between host and guest

ORACLE

VMM Physical Memory Management
Solutions

Solution Set A: Each guest hogs all memory given to it, *but…*

Solution Set B: Guest memory is dynamically adjustable *…somehow*

Solution Set C: Total guest memory is dynamically load-balanced across all guests …*using* some *policy*

Solution Set D: Host-provided "compensation" … *to correct for insufficiently omniscient policy*

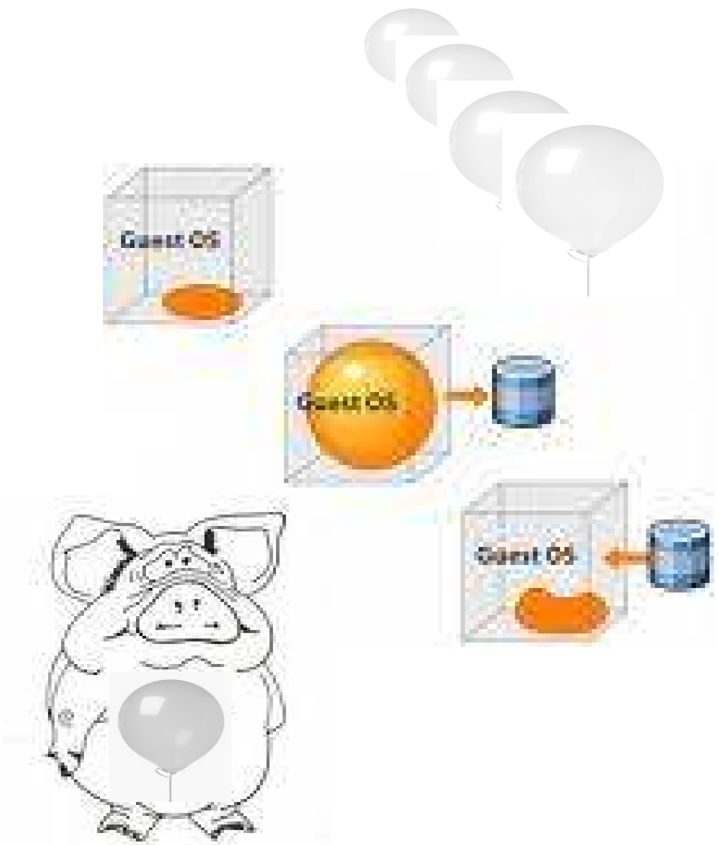
VMM Physical Memory Management
Solution Set B

Solution Set B: Guest memory is dynamically adjustable

- Balloon driver
- “Virtual Hot plug” memory

ORACLE

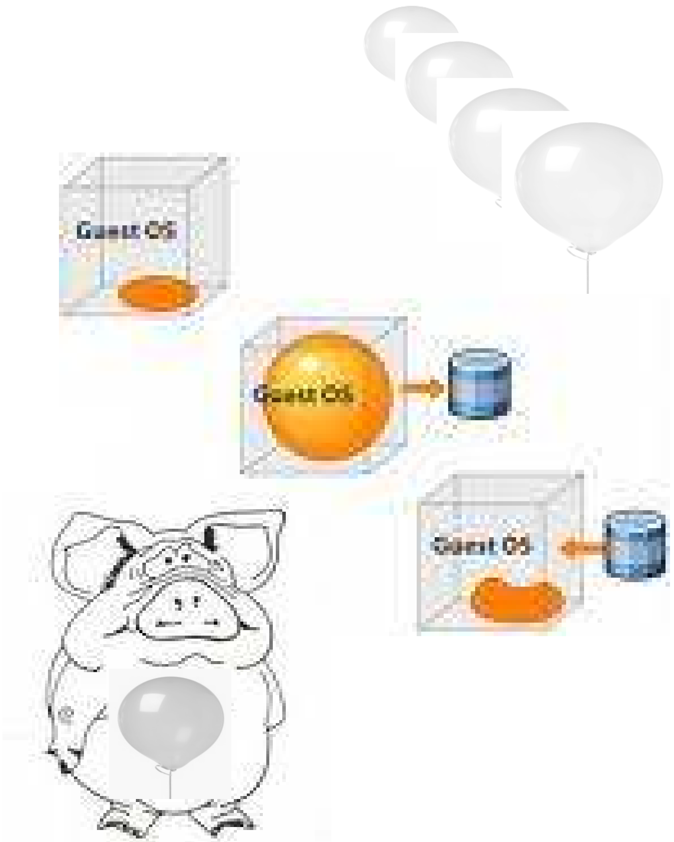
VMM Physical Memory Management
Balloon driver



- In-guest driver under the control of the host
  - a “memory trojan horse”

VMM Physical Memory Management
Ballooning

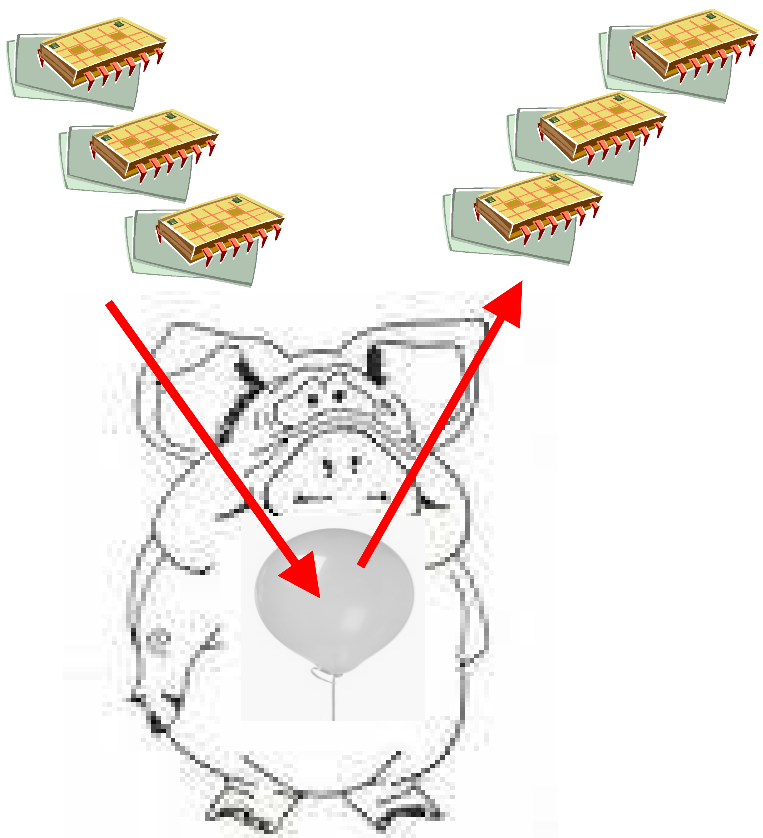
- In-guest driver under the control of the host
  - a “memory trojan horse”
- *BUT…*
  - very workload dependent
  - sometimes causes host swapping (resulting in unpredictable performance)
  - poor match for 2MB pages



KVM Xen

ORACLE

VMM Physical Memory Management
Virtual Hot Plug memory



- Fools the OS's native hot-plug memory interface
- *BUT…*
  - only useful for higher granularity
  - hot-plug interface not designed for high frequency changes or mid-size granularity
  - hot plug delete is problematic

VMM Physical Memory Management
Solution Set B (Summary)

Solution Set B: Guest memory is dynamically adjustable

- Ballooning
  - unpredictable side effects
- Hot plug memory
  - Low granularity

VMM Physical Memory Management
Solution Set B (Summary)

Solution Set B: Guest memory is dynamically adjustable

- Ballooning
  - unpredictable side effects
- Hot plug memory
  - Low granularity

These are *mechanisms,* not solutions!

ORACLE

VMM Physical Memory Management
Solutions

Solution Set A: Each guest hogs all memory given to it, *but…*

Solution Set B: Guest memory is dynamically adjustable *…somehow*

Solution Set C: Total guest memory is dynamically load-balanced across all guests …***using* some *policy***

Solution Set D: Host-provided "compensation" … *to correct for insufficiently omniscient policy*

ORACLE

VMM Physical Memory Management
Solution Set C

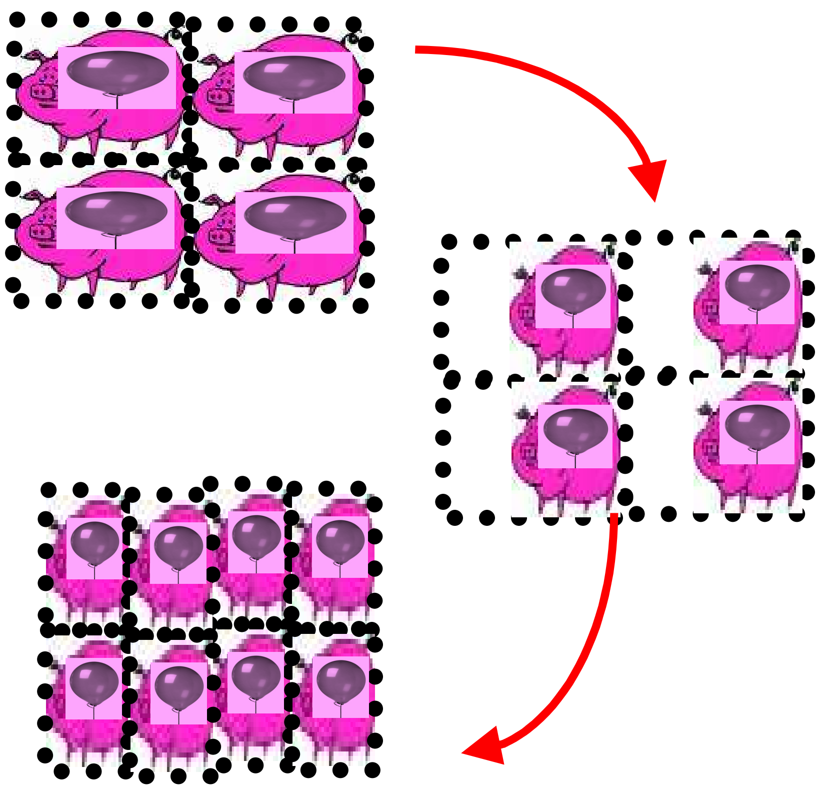
Solution Set C: Guests are dynamically "load balanced" using *some* policy

- Guest-quantity-based policy
- Guest-pressure-driven host-control policy
- Guest-pressure-driven guest-control policy

VMM Physical Memory Management

Citrix Dynamic Memory Control (DMC)

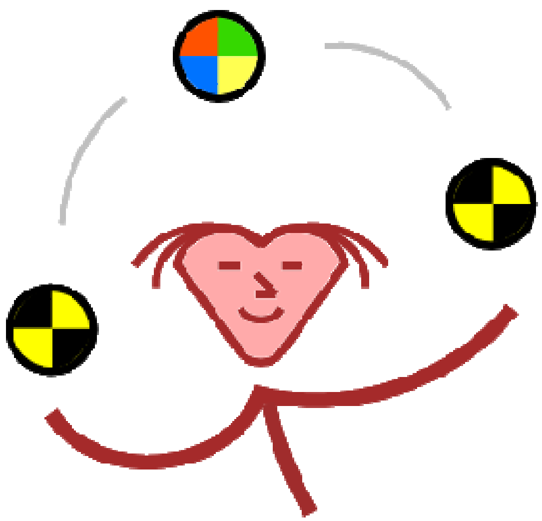
for Xen Cloud Platform (XCP)



- administrator presets memory “range” for each guest
- balloons adjusted based on number of guests
- does NOT respond to individual guest memory pressure

http://wiki.xensource.com/xenwiki/Dynamic_Memory_Control

VMM Physical Memory Management
KVM Memory Overcommitment Manager



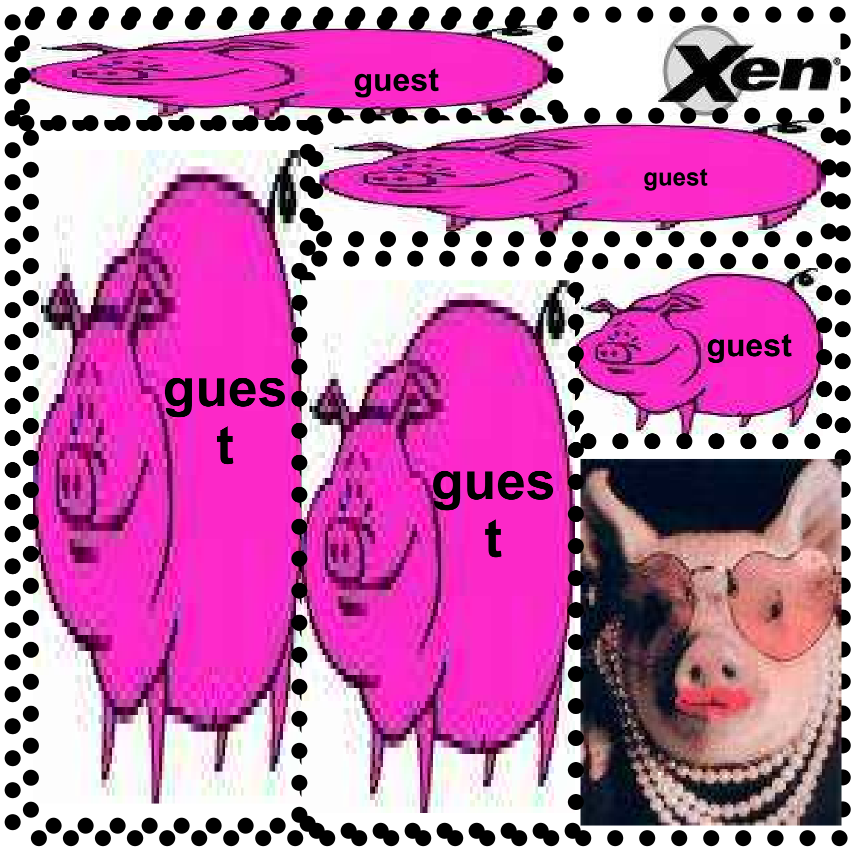
- collects host and guest memory stats, sends to customizable policy engine
- controls all guest balloons, plus host page sharing (KSM)
- shrinks all guests “fairly” scaled by host memory pressure

BUT…

- under-aggressive for idle guests
- issues due to lack of omniscience

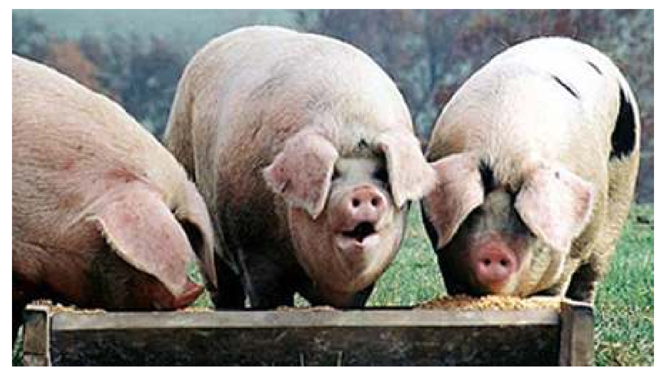
http://wiki.github.com/aglitke/mom

VMM Physical Memory Management
in the presence of under-aggressive ballooning



Ballooning works great for giving more memory TO a guest OS…

*Look ma! No more fallow memory! (*burp*)*

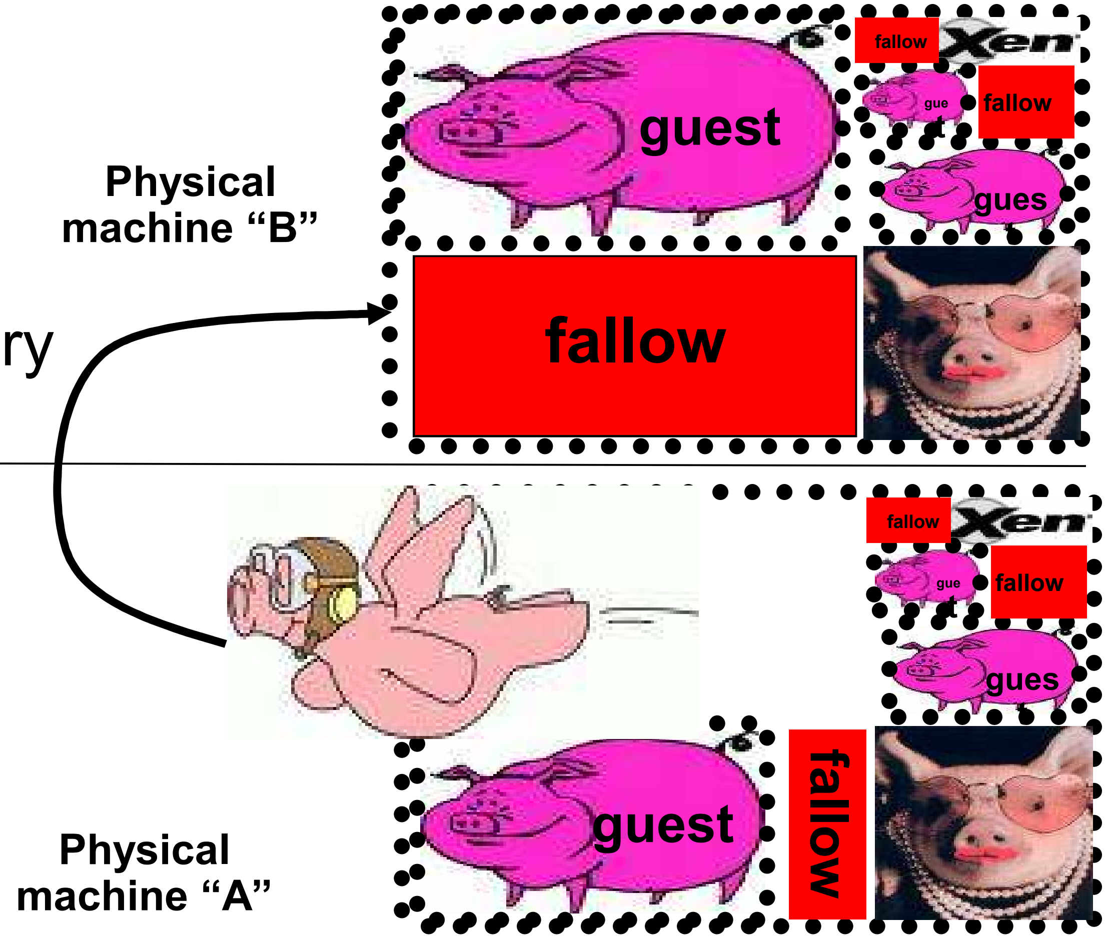


ORACLE

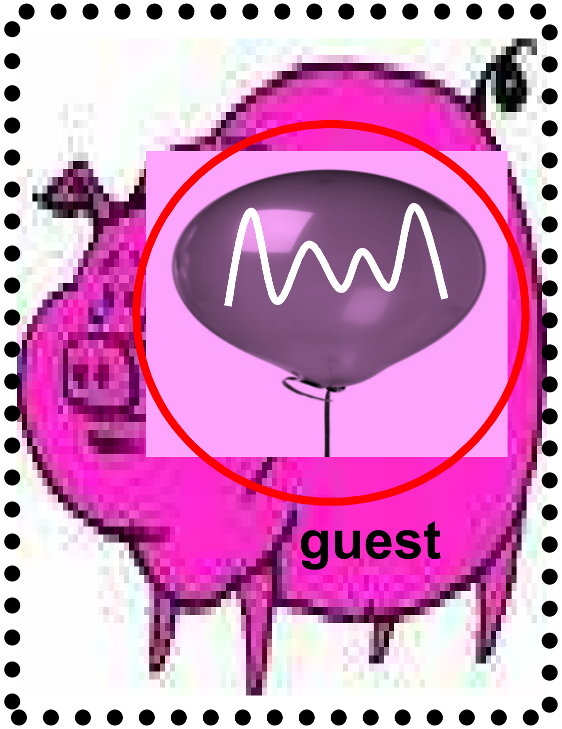
VMM Physical Memory Management

under-aggressive ballooning limits migration

- migration
  - requires fallow memory in the target machine
  - leaves behind fallow memory in the originating machine



VMM Physical Memory Management
Self-ballooning



- In Xen tree since mid-2008
- Use *in-guest* feedback to resize balloon
  - aggressively
  - frequently
  - independently
  - configurably
- For Linux, size to maximum of:
  - /proc/meminfo “CommittedAS”
  - memory floor enforced by Xen balloon driver
- Userland daemon *or* patched kernel

Committed_AS: An estimate of how much RAM you would need to make a 99.99% guarantee that there never is OOM (out of memory) for this workload. Normally the kernel will overcommit memory. The Committed_AS is a guesstimate of how much RAM/swap you would need worst-case. (From **http://www.redhat.com/advice/tips/meminfo.html**)

VMM Physical Memory Management

over-aggressive ballooning

- “enforced memory asceticism”
- ballooning does *not* work well *to take memory away*

Memory Asceticism / Aggressive Self-ballooning *ISSUES*

ISSUE #1: Pages evicted due to memory pressure are most likely to be clean page cache pages. Eliminating these (without a crystal ball) results in refaults → *additional disk reads*

ISSUE #2: When no more clean pagecache pages can be evicted, dirty mapped pages get written … and rewritten… and rewritten to disk → *additional disk writes*

ISSUE #3: Sudden large memory demands may occur unpredictably (e.g. from a new userland program launch) but the "ask for" mechanism can't deliver enough memory fast enough → *failed mallocs, swapping, and **OOM**s*

ORACLE

Memory Asceticism / Aggressive Self-ballooning *ISSUES*

ISSUE #1: Pages evicted due to memory pressure are most likely to be clean pagecache pages. Eliminating these (without a crystal ball) results in refaults → *additional disk reads*

ISSUE #2: When no more clean pagecache pages can be evicted, dirty mapped pages get written … and rewritten… and rewritten to disk → *additional disk writes*

ISSUE #3: Sudden large memory demands may occur unpredictably (e.g. from a new userland program launch) but the “ask for” mechanism can’t deliver enough memory fast enough → *failed mallocs, swapping, and **OOM**s*

ORACLE

Memory Asceticism / Aggressive Self-ballooning *ISSUES*

ISSUE #1: Pages evicted due to memory pressure are most likely to be clean pagecache pages. Eliminating these (without a crystal ball) results in refaults → *additional disk reads*

ISSUE #2: When no more clean pagecache pages can be evicted, dirty mapped pages get written … and rewritten… and rewritten to disk → *additional disk writes*

ISSUE #3: Sudden large memory demands may occur unpredictably (e.g. from a new userland program launch) but the "ask for" mechanism can't deliver enough memory fast enough → *failed mallocs, swapping, and* ***OOM****s*

VMM Physical Memory Management
Solution Set C Summary

Solution Set C: Guests are dynamically "load balanced" using *some* policy

- Guest-quantity-based policy
- Guest-pressure-driven host-control policy
- Guest-pressure-driven guest-control policy

→ ALL POLICIES ~~SUCK~~ HAVE ISSUES BECAUSE:

1) MEMORY PRESSURE IS DIFFICULT TO MEASURE

2) HARD TO PREDICT THE FUTURE IS (Yoda)

ORACLE

VMM Physical Memory Management
Solutions

Solution Set A: Each guest hogs all memory given to it, *but…*

Solution Set B: Guest memory is dynamically adjustable *…somehow*

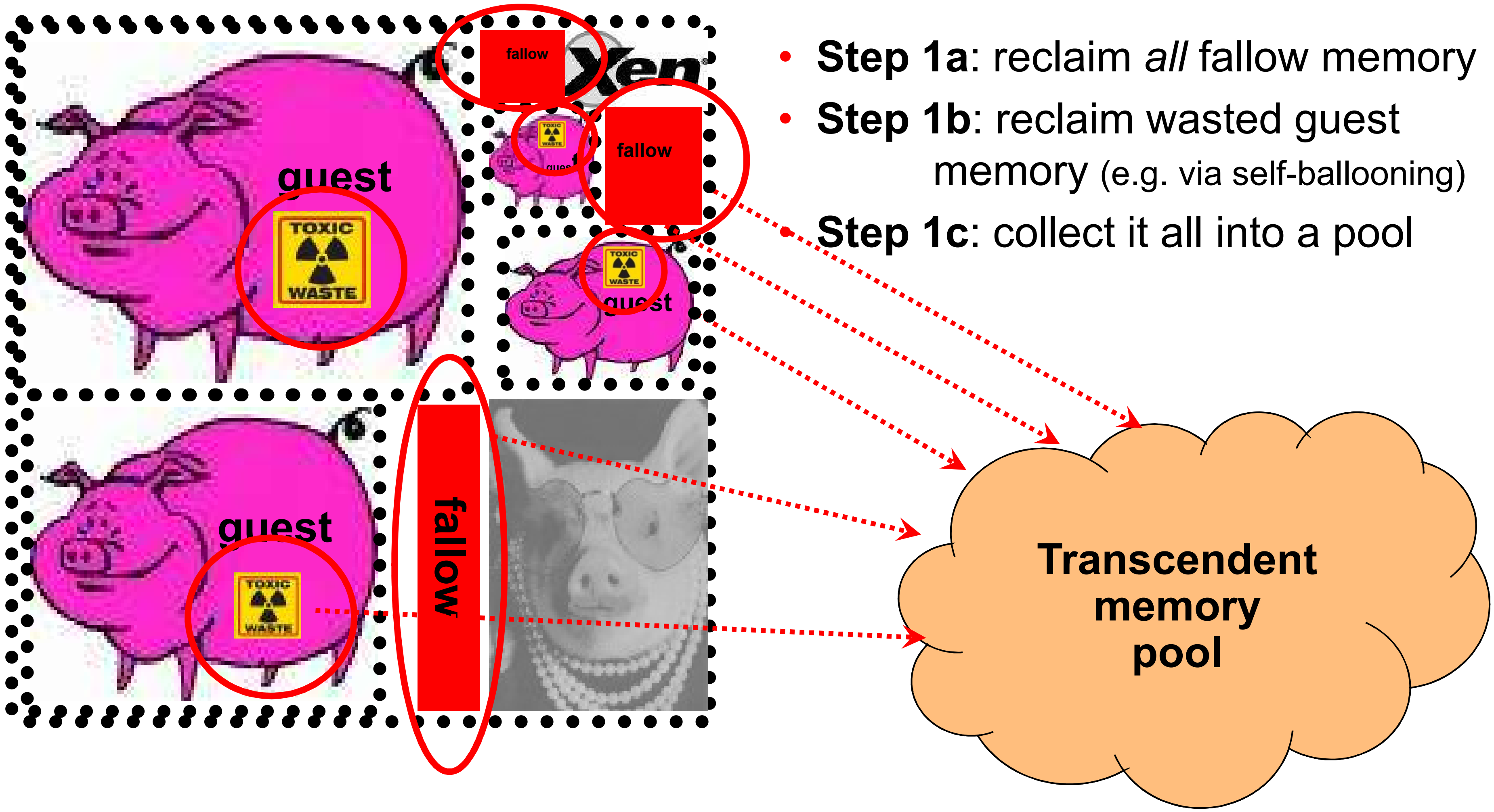
Solution Set C: Total guest memory is dynamically load-balanced across all guests …*using* some *policy*

Solution Set D: Host-provided “compensation” … *to correct for poor or non-omniscient policy*

Agenda

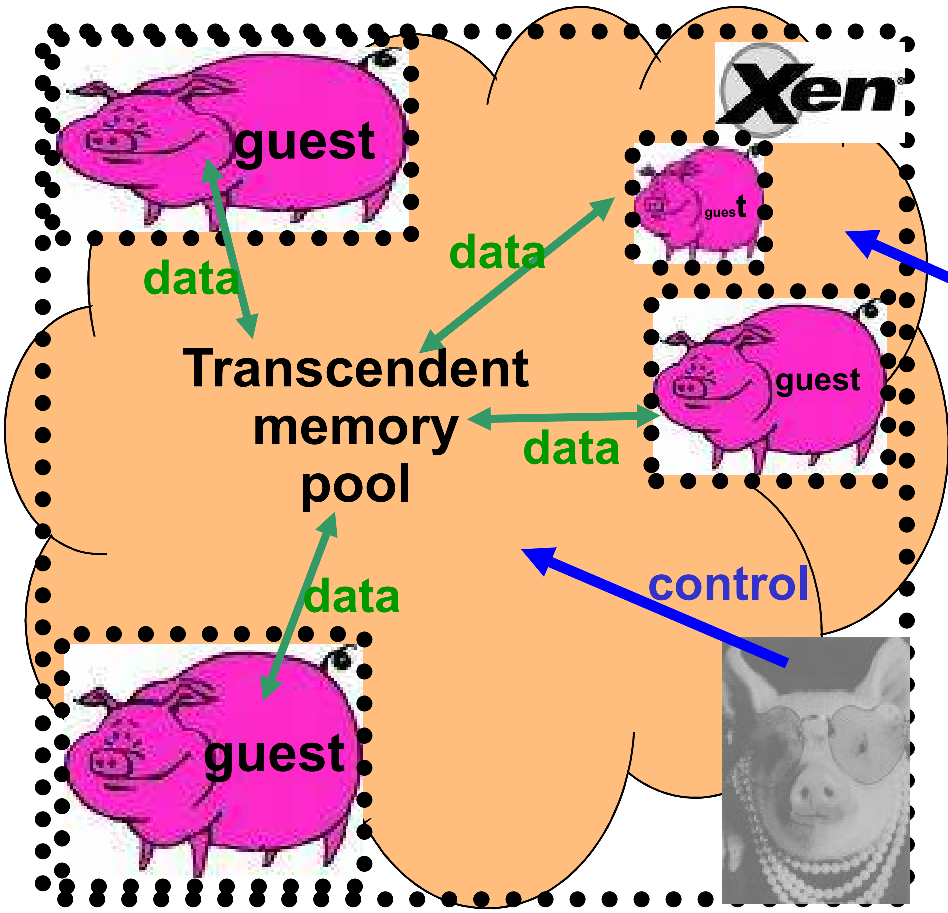
- Motivation and Challenge
- Memory Optimization Solutions in a Virtual Environment
- Transcendent Memory ("tmem") Overview
- Self-ballooning + Tmem Performance Analysis

Transcendent memory
creating the transcendent memory pool

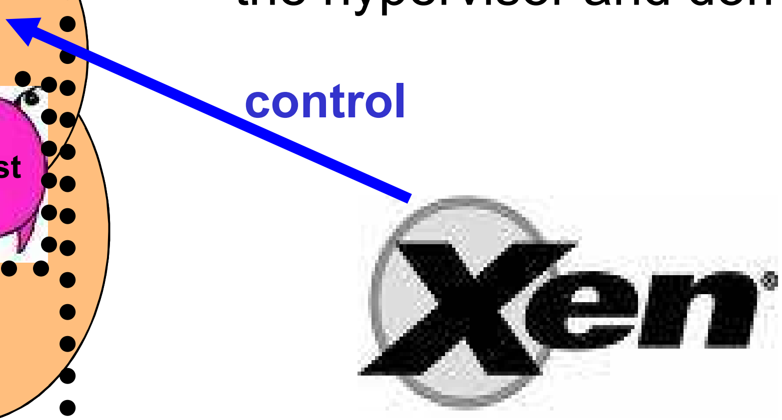


- **Step 1a**: reclaim *all* fallow memory
- **Step 1b**: reclaim wasted guest memory (e.g. via self-ballooning)
- **Step 1c**: collect it all into a pool

Transcendent memory
creating the transcendent memory pool

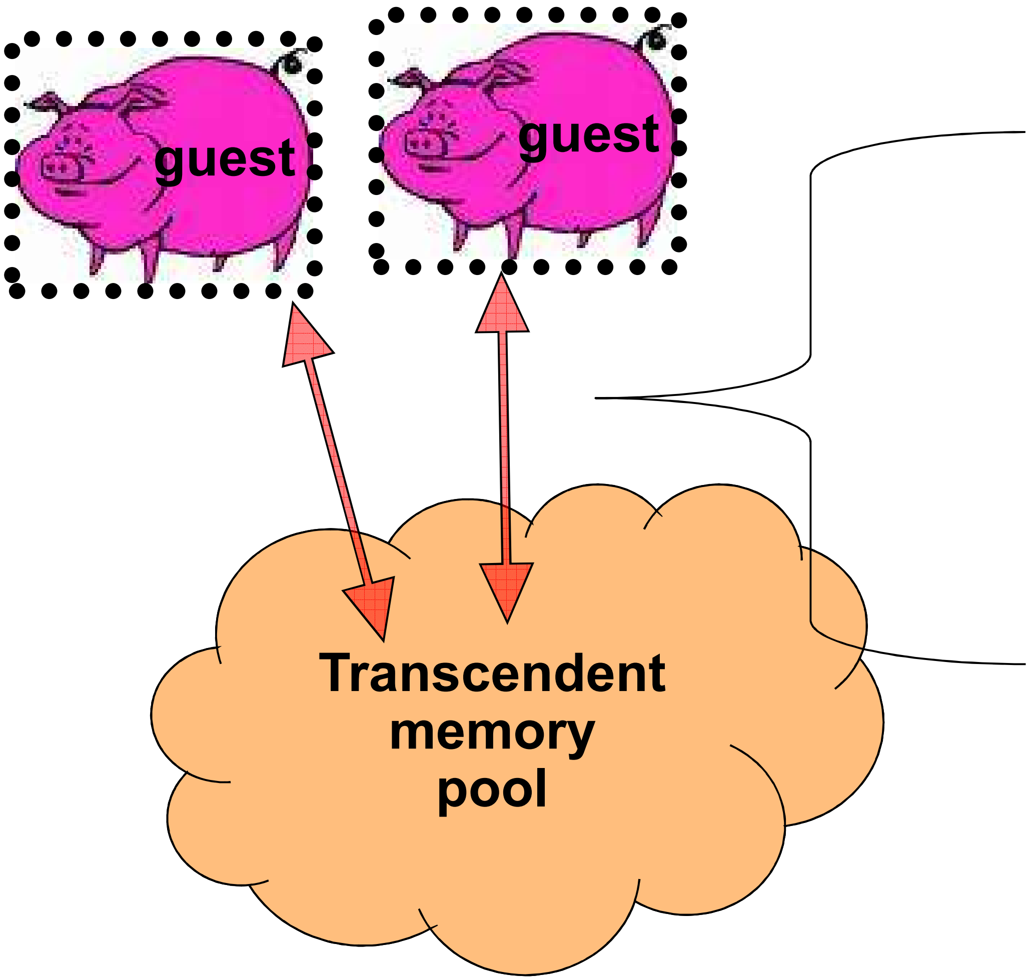


- **Step 2**: provide ***indirect*** access, strictly controlled by the hypervisor and dom0



ORACLE

Transcendent memory
API characteristics



Transcendent memory API

- paravirtualized (lightly)
- narrow
- well-specified
- operations are:
  - synchronous
  - page-oriented (one page per op)
  - copy-based
- multi-faceted
- extensible

Transcendent memory
four different subpool types
→ four different uses

flags	*ephemeral*	persistent
private	"second-chance" clean-page cache!! → "**cleancache**"	Fast swap "device"!! → "**frontswap**"
shared	*server-side cluster filesystem cache* → "**shared cleancache**"	*inter-guest shared memory?*

Legend:

Implemented and working today (Linux + Xen)

Working but limited testing

Under investigation

eph-em-er-al, *adj., … transitory, existing only briefly, short-lived (i.e. NOT persistent)*

ORACLE

Tmem guest kernel paravirtualization
cleancache

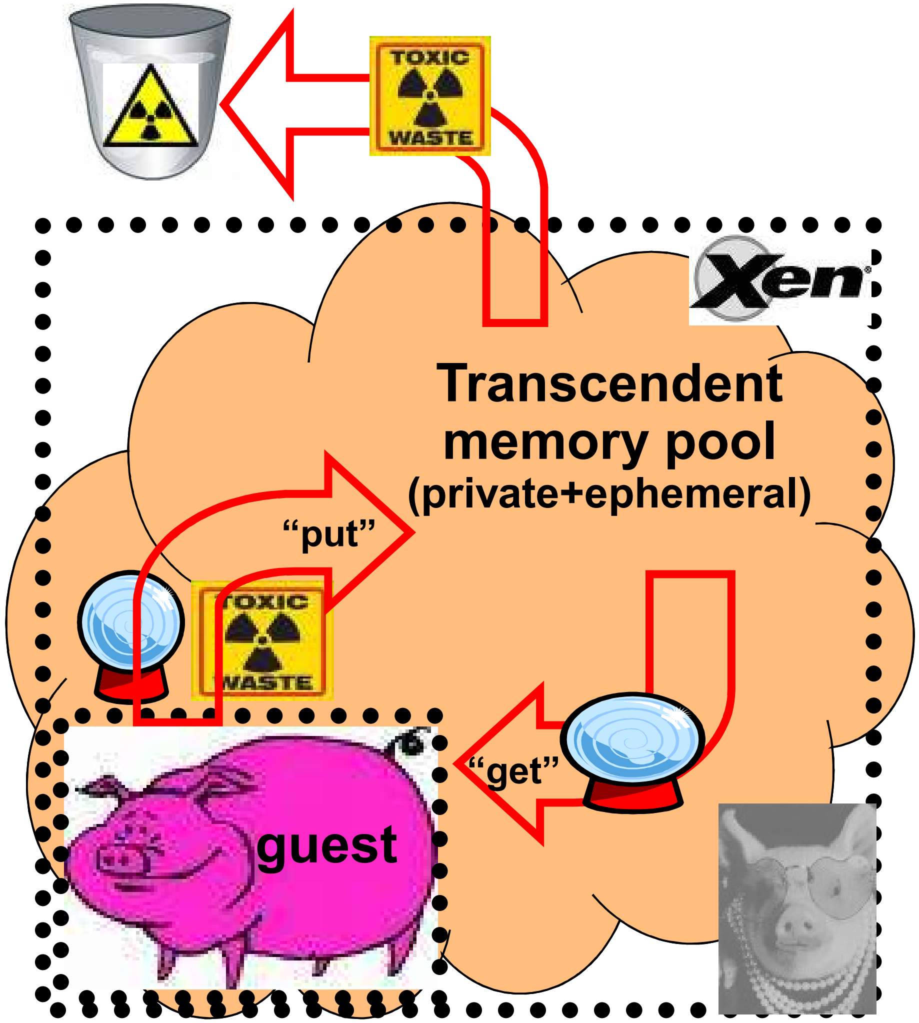
"*Cleancache *is a proposed new optional feature to be provided by the VFS layer that potentially dramatically increases page cache effectiveness for many workloads in many environments at a negligible cost. Filesystems that are well-behaved and conform to certain restrictions can utilize cleancache simply by making a call to* cleancache_init_fs() *at mount time. Unusual, misbehaving, or poorly layered filesystems must either add additional hooks and/or undergo extensive additional testing… or should just not enable the optional cleancache."

Filesystem restrictions to use cleancache

- Little or no value for RAM-based filesystems
- Coherency: File removal/truncation must layer on VFS
  - or FS must add additional hooks to do same (issue in FScache net FS's?)
- Inode numbers must be unique
  - no emulating 64-bit inode space on 32-bit inode numbers
- Superblock alloc/deactivate must layer on VFS
  - or FS must add additional hooks to do same
- Performance: Page fetching via VFS
  - or FS must add additional hooks to do same (e.g. btrfs)
- FS blocksize should match PAGE_SIZE
  - or existing backends will ignore
- Clustered FS should use "shared_init_fs" for best performance
  - on some backends, ignored on others

ORACLE

cleancache



- a ***second-chance*** clean page cache for a guest
  - "put" clean pages only
  - "get" only valuable pages
  - pages eventually are evicted
  - coherency managed by guest
  - exclusive cache semantics

Transcendent Memory Pool types

	ephemeral	persistent
private	"second-chance" clean-page cache!! → "cleancache"	Fast swap "device"!! → "frontswap"
shared	*server-side cluster filesystem cache? → "shared cleancache"*	*inter-domain shared memory?*

ORACLE

Memory Asceticism / Aggressive Self-ballooning *ISSUES*

ISSUE #1: Pages evicted due to memory pressure are most likely to be clean pagecache pages. Eliminating these (without a crystal ball) results in refaults → *additional disk reads*

ISSUE #2: When no more clean pagecache pages can be evicted, dirty mapped pages get written … and rewritten… and rewritten to disk → *additional disk writes*

ISSUE #3: Sudden large memory demands may occur unpredictably (e.g. from a new userland program launch) but the "ask for" mechanism can't deliver enough memory fast enough → *failed mallocs, swapping, and **OOM**s*

Tmem guest kernel paravirtualization
frontswap

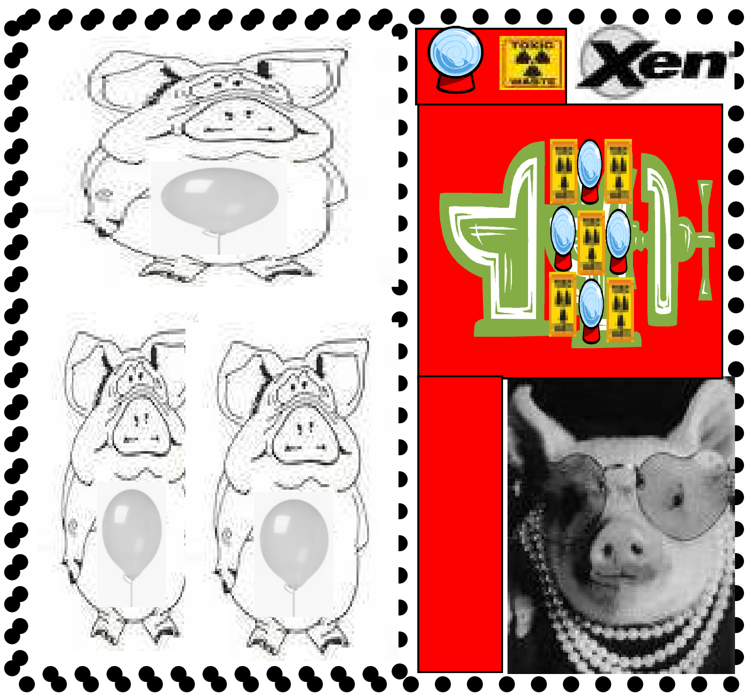
"Frontswap is meant to deal with dirty pages that the kernel would like to get rid of… Like cleancache, frontswap can play tricks with stored pages to stretch its memory resources. The real purpose behind this mechanism, though, appears to be to enable a hypervisor to respond quickly to memory usage spikes in virtualized guests. Dan put it this way:

Frontswap serves nicely as an emergency safety valve when a guest has given up (too) much of its memory via ballooning but unexpectedly has an urgent need that can't be serviced quickly enough by the balloon driver.

-- lwn.net, May 4, 2010,
http://lwn.net/Articles/386090/

ORACLE

frontswap



- over-ballooned guests experiencing unexpected memory pressure have an ***emergency swap disk***
  - much faster than swapping
  - persistent ("dirty") pages OK
  - prioritized higher than hcache
  - limited by domain's maxmem

Transcendent Memory Pool types

	ephemeral	***persistent***
private	"second-chance" clean-page cache! → "cleancache"	Fast swap "device"!! → "frontswap"
shared	*server-side cluster filesystem cache?* *→ "shared cleancache"*	*inter-domain shared memory?*

ORACLE

Memory Asceticism / Aggressive Self-ballooning *ISSUES*

ISSUE #1: Pages evicted due to memory pressure are most likely to be clean pagecache pages. Eliminating these (without a crystal ball) results in refaults → *additional disk reads*

ISSUE #2: When no more clean pagecache pages can be evicted, dirty mapped pages get written … and rewritten… and rewritten to disk → *additional disk writes*

ISSUE #3: Sudden large memory demands may occur unpredictably (e.g. from a new userland program launch) but the "ask for" mechanism can't deliver enough memory fast enough → *failed mallocs, swapping, and* ***OOM****s*

ORACLE

Transcendent Memory Status

- Tmem support officially released in Xen 4.0.0
- Optional compression and page deduplication support
- Enterprise-quality concurrency
- Complete save/restore and live migration support
- Linux-side patches available, including
  - ocfs2, btrfs, ext3, ext4 filesystem support
  - sysfs support for in-guest tmem statistics
  - targeting upstream Linux 2.6.37 (*cleancache), 2.6.38* (*frontswap*)
- Tmem “technology preview” releases:
  - Oracle VM 2.2
  - OpenSuSE 11.2; SLE11 (?)
  - Oracle Linux 5 update 5 rpm

ORACLE

Agenda

- Motivation and Challenge
- Memory Optimization Solutions in a Virtual Environment
- Transcendent Memory (“tmem”) Overview
- Self-ballooning + Tmem Performance Analysis

Test workload (overcommitted!)

- Dual core (Conroe) processor, 2GB RAM, IDE disk
- Four single vcpu PV VMs, in-kernel self-ballooning+tmem
  - Oracle Enterprise Linux 5 update 4; two 32-bit + two 64-bit
  - mem=384MB (maxmem=512MB)… total = 1.5GB (2GB maxmem)
  - virtual block device is tap:aio (file contains 3 LVM partitions: ext3+ext3+swap)
- Each VM waits for all VMs to be ready, then ***simultaneously***
  - two Linux kernel compiles (2.6.32 source), then force crash:
    - `make clean; make -j8; make clean; make -j8`
    - `echo c > /proc/sysrq-trigger`
- Dom0: 256MB fixed, 2 vcpus
  - automatically launches all domains
  - checks every 60s, waiting for all to be crashed
  - saves away statistics, then reboots

ORACLE

Measurement methodology

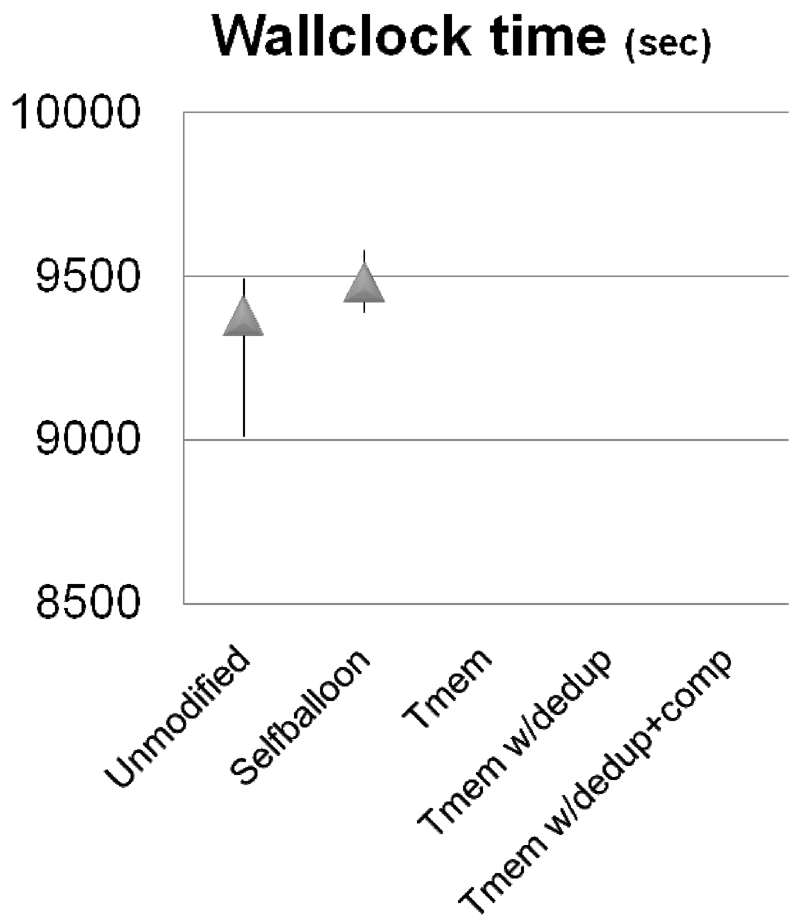
- Four statistics measured for each run
  - Temporal: (1) wallclock time to completion; (2) total vcpu including dom0
  - Disk access: vbd sectors (3) read and (4) written
- Test workload run five times for each configuration
  - high and low sample of each statistic discarded
  - use average of middle three samples for "single-value" statistic
- Five different configurations:

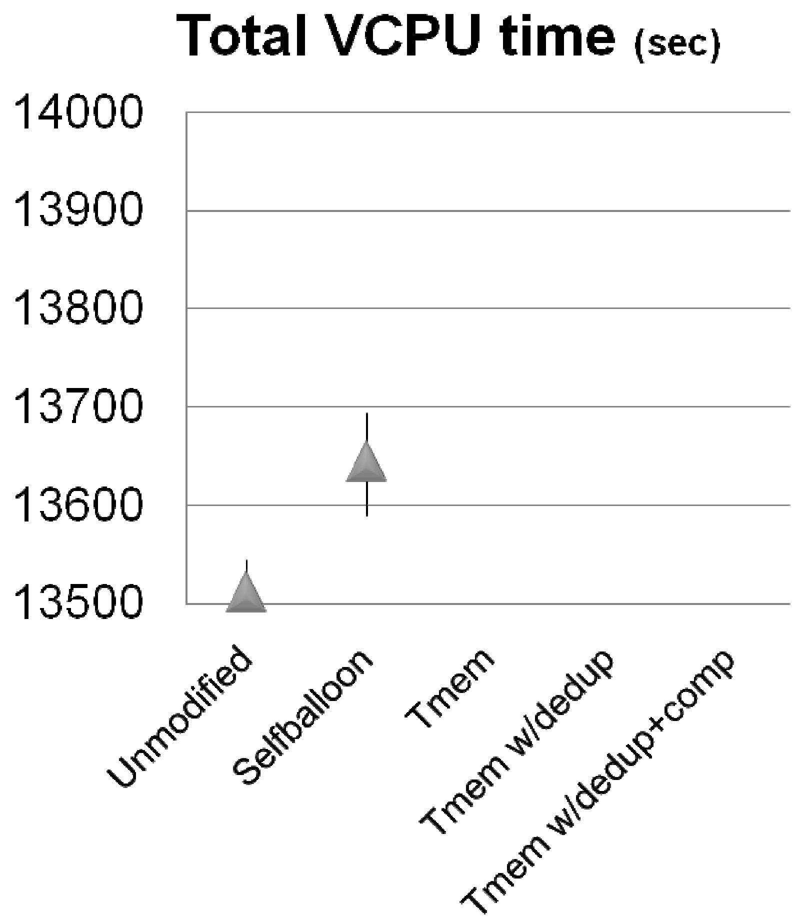
Features enabled / Configuration	Self-ballooning	Tmem	Page Dedup	Compression
Unchanged	NO	NO	NO	NO
Self-ballooning	YES	NO	NO	NO
Tmem	YES	YES	NO	NO
Tmem w/dedup	YES	YES	YES	NO
Tmem w/dedup+ comp	YES	YES	YES	YES

ORACLE

Unchanged vs. Self-ballooning only

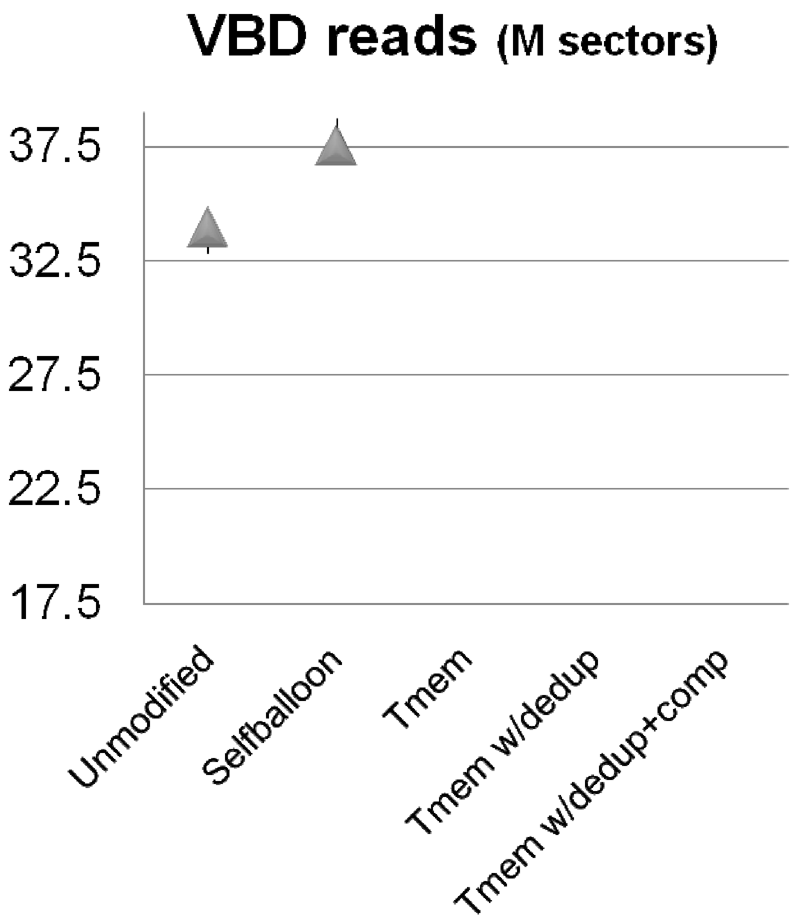
Temporal stats

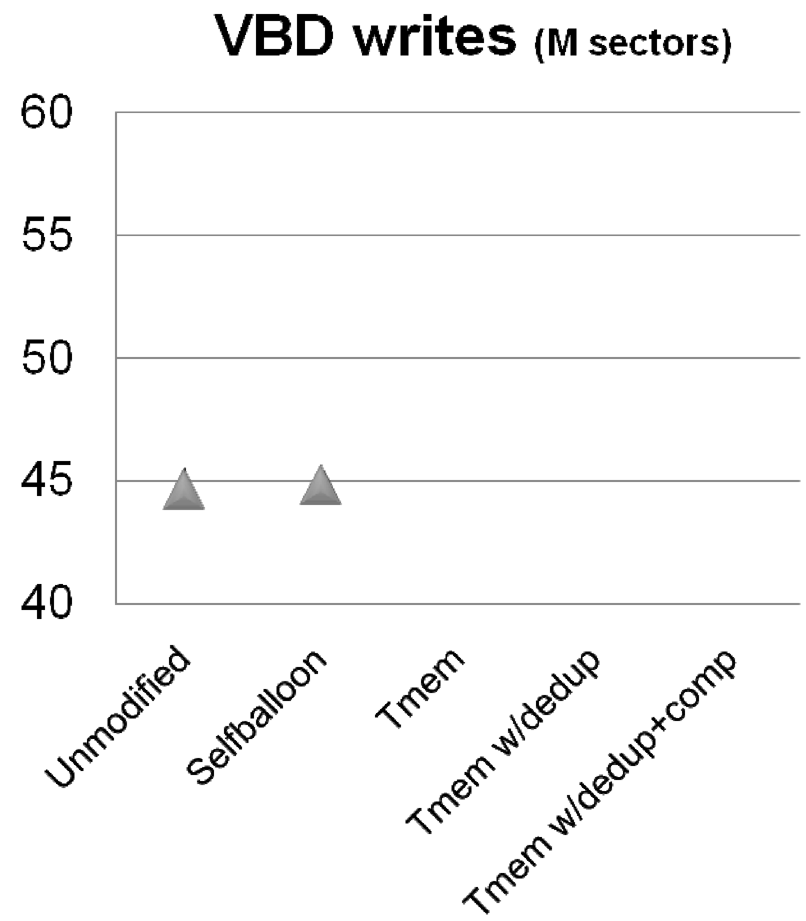




Unchanged vs. Self-ballooning only

Virtual block device stats



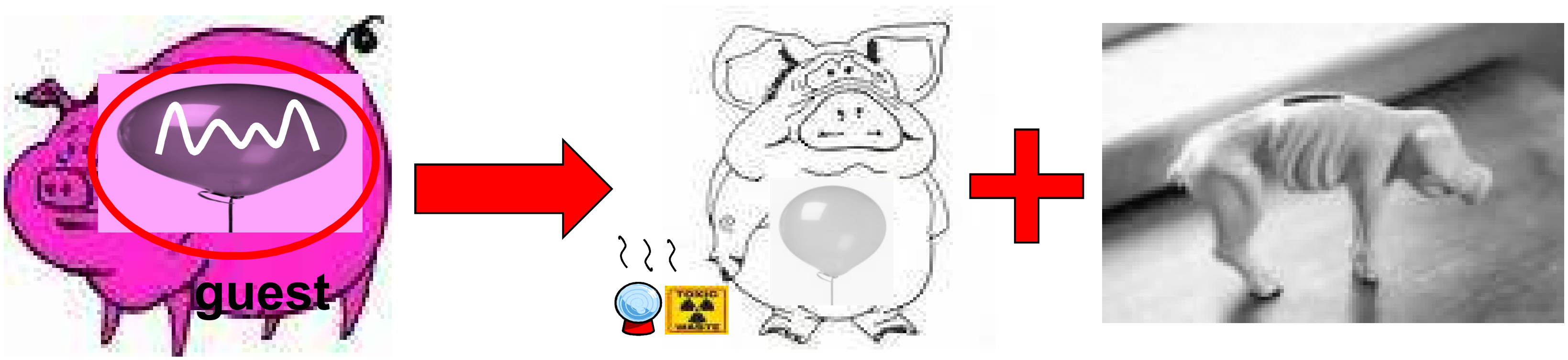


AS EXPECTED: a performance hit!

Aggressive ballooning (*by itself*) doesn't work very well!

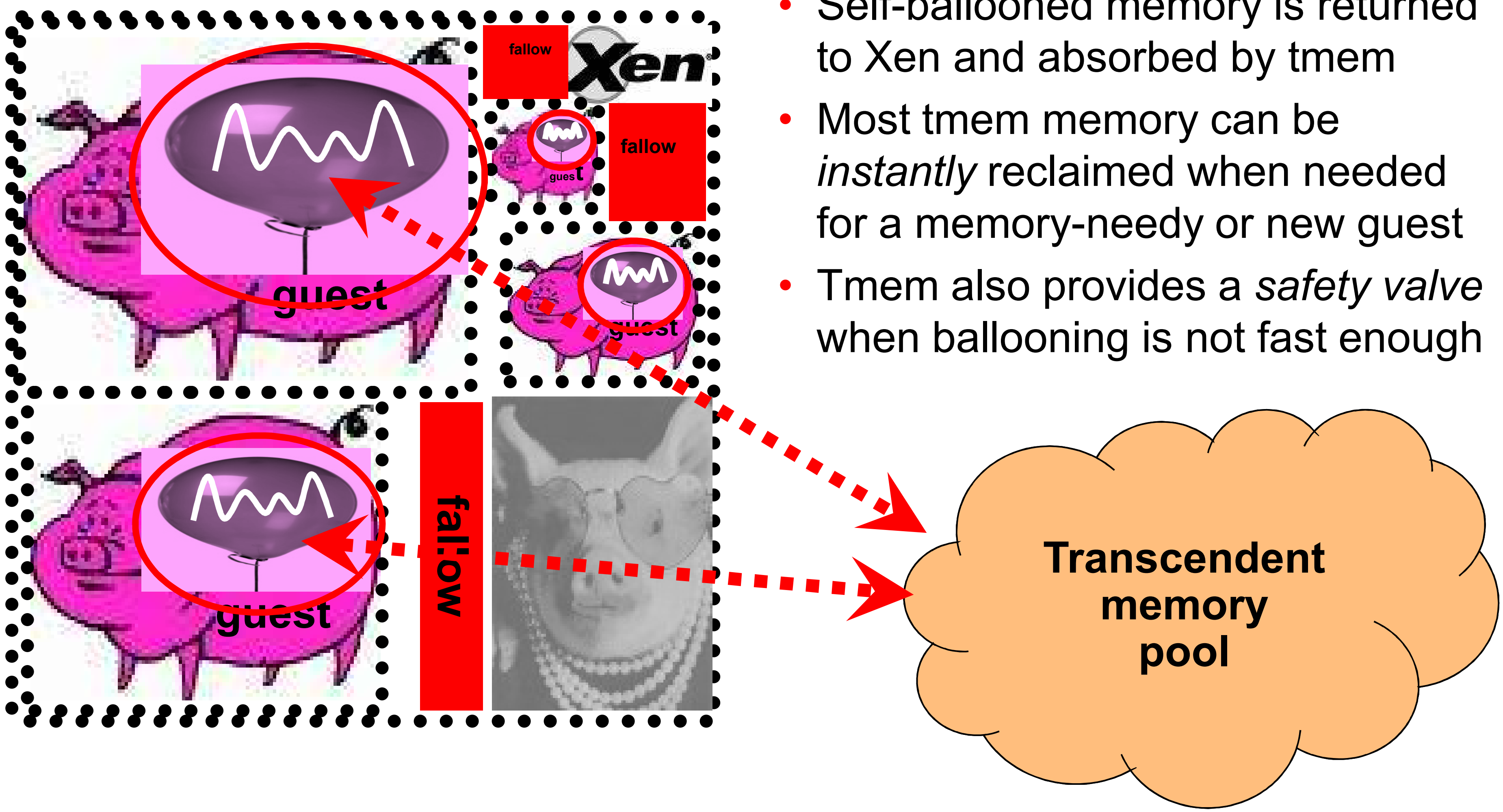
- Self-ballooning *indiscriminately* shrinks the guest OS's page cache, causing *refaults*!

→ PERFORMANCE **WILL GET WORSE** WHEN LARGE-MEMORY GUESTS ARE AGGRESSIVELY BALLOONED



Self-ballooning AND Transcendent Memory

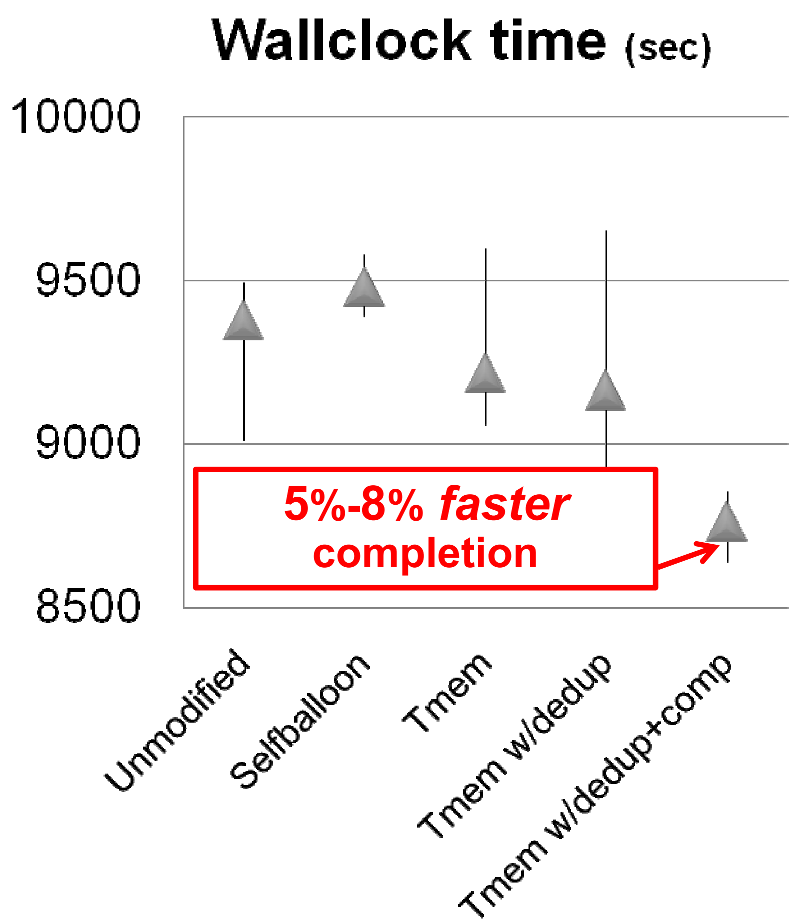
…go together like a horse and carriage

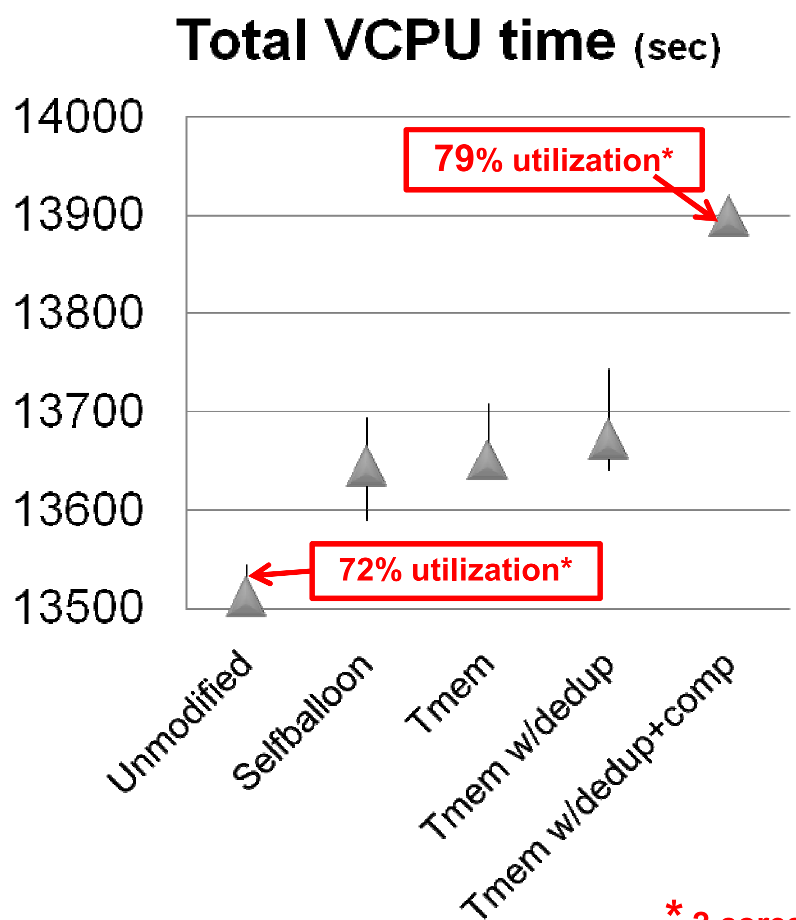


- Self-ballooned memory is returned to Xen and absorbed by tmem
- Most tmem memory can be *instantly* reclaimed when needed for a memory-needy or new guest
- Tmem also provides a *safety valve* when ballooning is not fast enough

Self-ballooning AND Tmem

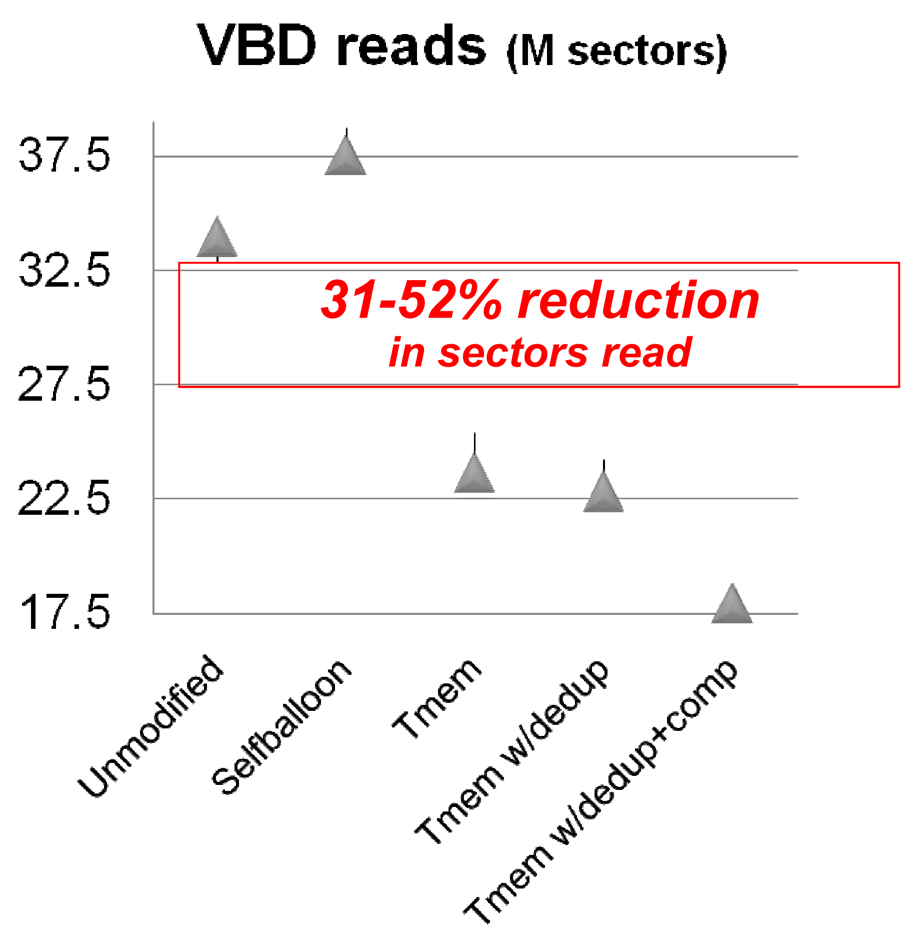
Temporal stats

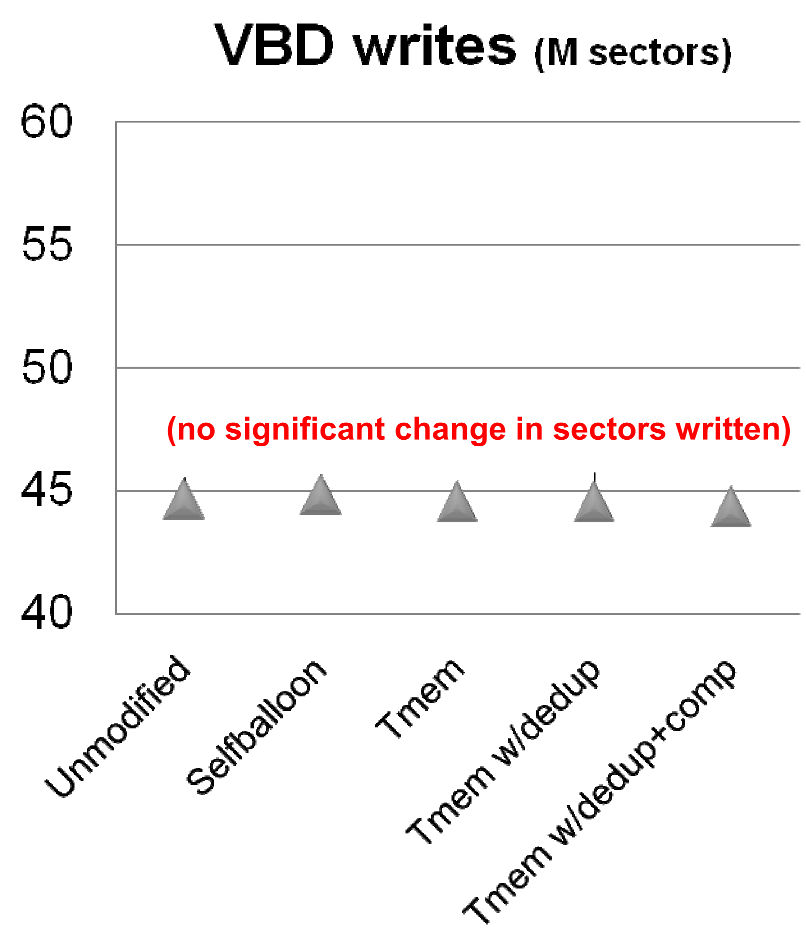




* 2 cores

Self-ballooning AND Tmem
virtual block device stats





ORACLE

WOW! Why is tmem so good?

- Tmem-enabled guests *statistically multiplex* one *shared virtual page cache* to reduce disk refaults!
  - 252068 page (**984MB**) max (NOTE: actual tmem measurement)
- Deduplication and compression together *transparently QUADRUPLE* apparent size of this virtual page cache!
  - 953166 page **(3723MB**) max (actually measured by tmem… on 2GB system!)
- Swapping-to-disk (e.g. due to insufficiently responsive ballooning) is converted to in-memory copies *and* statistically multiplexed
  - 82MB at workload completion, 319MB combined max (actual measurement)
  - uses compression but not deduplication
- CPU “costs” entirely hidden by increased CPU utilization

→ RESULTS MAY BE EVEN **BETTER** WHEN WORKLOAD IS TEMPORALLY DISTRIBUTED/SPARSE

ORACLE

Transcendent Memory Update Summary

Tmem advantages:

- **greatly increased** *memory utilization/flexibility*
- **dramatic reduction** in *I/O bandwidth requirements*
- **more effective** *CPU utilization*
- **faster** *completion* of (some?) workloads

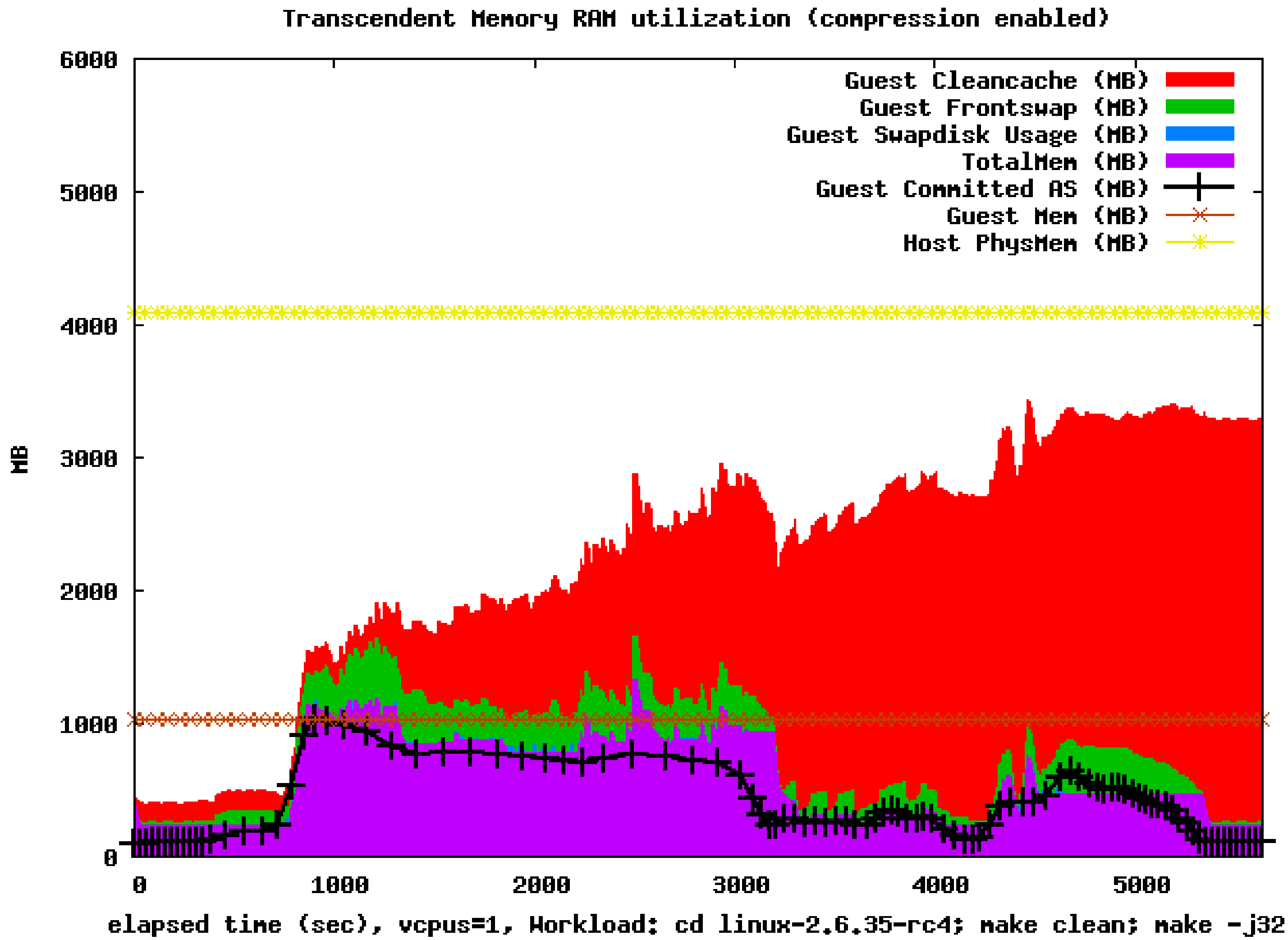
Tmem disadvantages:

- tmem-modified kernel required (cleancache and frontswap)
- higher power consumption due to higher CPU utilization

ORACLE

Cleancache and Frontswap in Action

Oracle Linux 5u5 (with tmem+selfballooning patch) on Xen 4.0



ORACLE

For more information

http://oss.oracle.com/projects/tmem

or xen-unstable.hg/docs/misc/tmem-internals.html

dan.magenheimer@oracle.com

ORACLE