Xen Summit™

2010

Update on Transcendent Memory on Xen

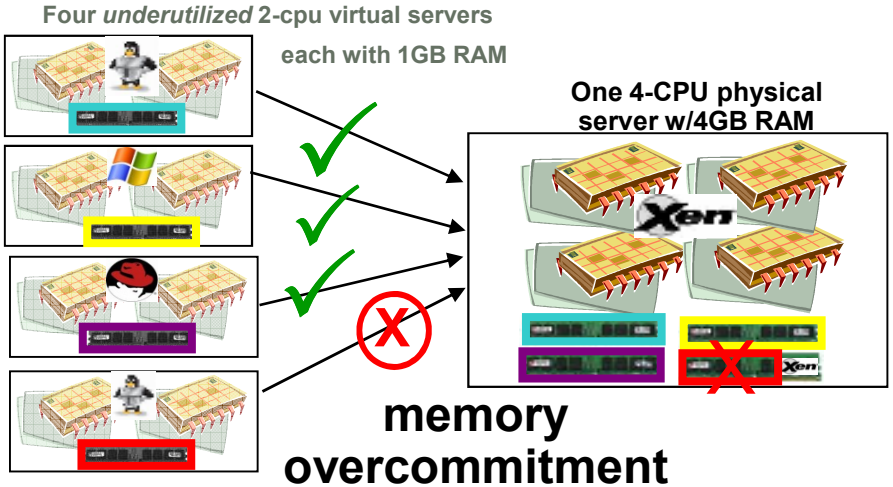
Speaker: Dan Magenheimer
Oracle Corporation

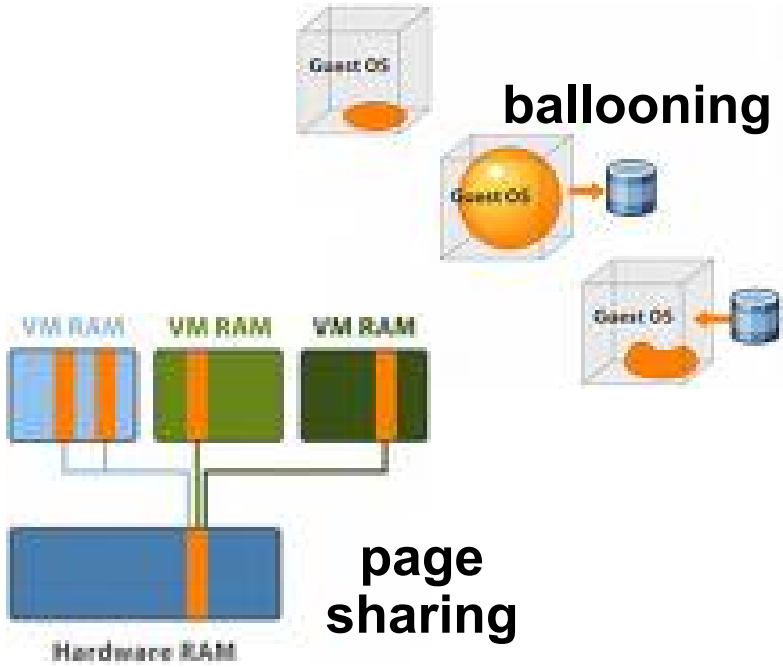
Agenda

- Motivation and Challenge
- BRIEF overview of Physical Memory Management
- BRIEF Transcendent Memory ("tmem") Overview
- Tmem Progress since Xen Summit 2009
- Self-ballooning + Tmem Performance Analysis

Motivation

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





The Virtualized Physical Memory Resource Optimization Challenge

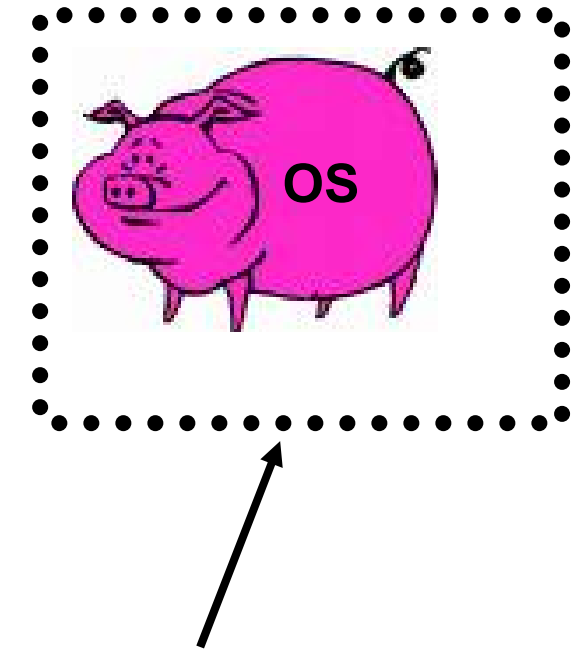
Optimize, across time, the distribution of machine memory among a maximal set of virtual machines by:

- 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

Agenda

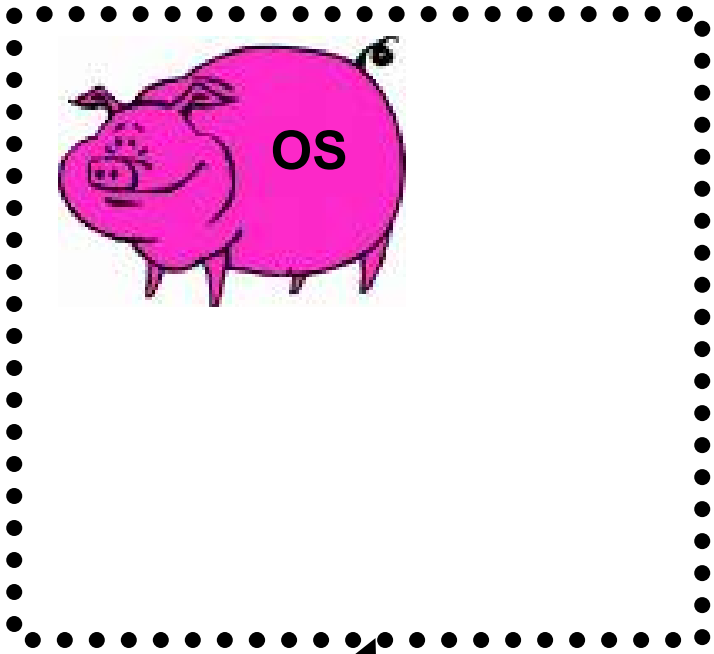
- Motivation and Challenge
- *BRIEF Overview of Physical Memory Management*
  - *in an operating system*
  - in a virtual machine monitor (Xen)
- BRIEF Transcendent Memory ("tmem") Overview
- Tmem Progress since Xen Summit 2009
- Self-ballooning + Tmem Performance Analysis

OS Physical Memory Management



- Operating systems are memory hogs!

OS Physical Memory Management



- Operating systems are memory hogs!

If you give an operating system more memory…..

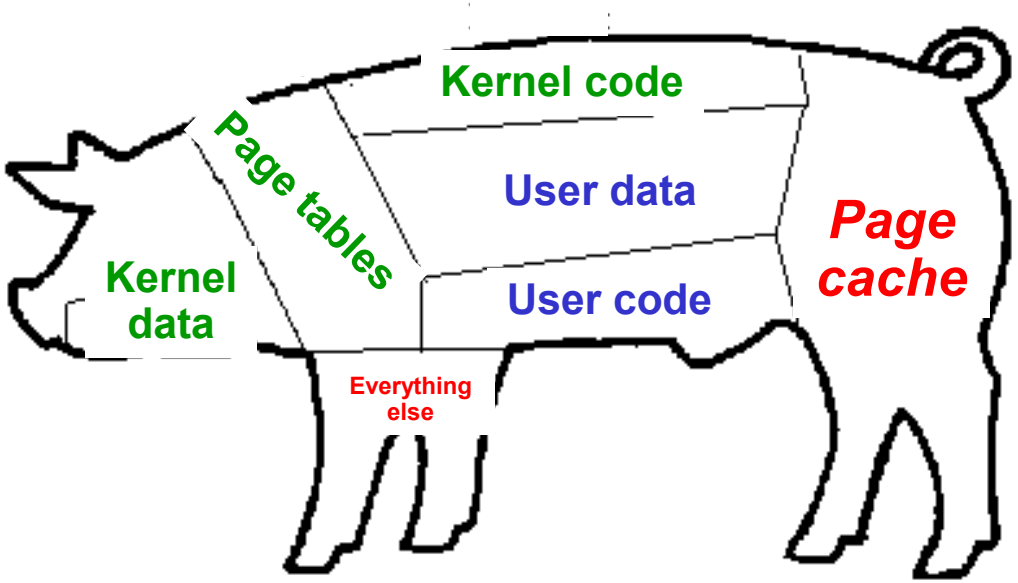
OS Physical Memory Management

- Operating systems are memory hogs!

If you give an OS more memory

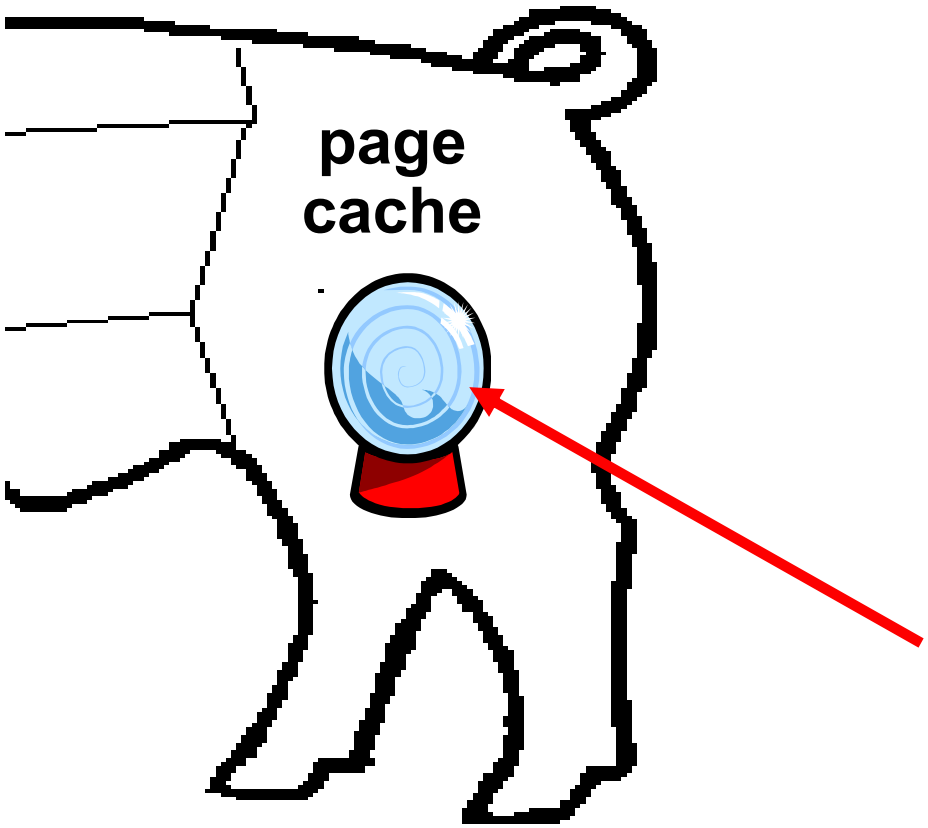
…it uses up any memory you give it!

OS Physical Memory Management



- What does an OS do with all that memory?
  - Kernel code and data
  - User code and data
  - *Page cache!*

OS Physical Memory Management

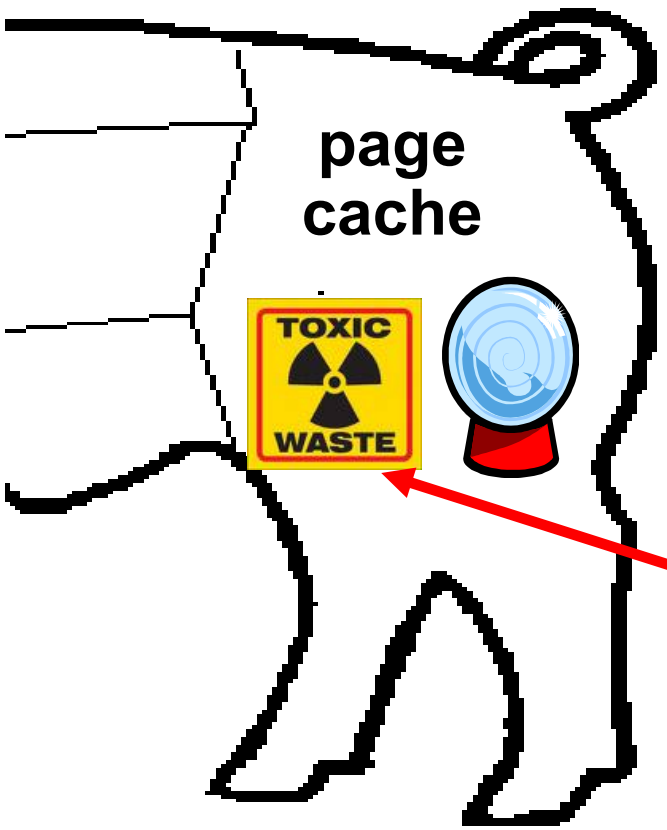


- What does an OS do with all that memory?

Page cache attempts to predict future needs of pages from the disk…

sometimes it gets it right → "good" pages

OS Physical Memory Management



- What does an OS do with all that memory?

Page cache attempts to predict future needs of pages from the disk…

sometimes it gets it wrong → "wasted" pages

OS Physical Memory Management

- What does an OS do with all that memory?

…much of the time mostly page cache

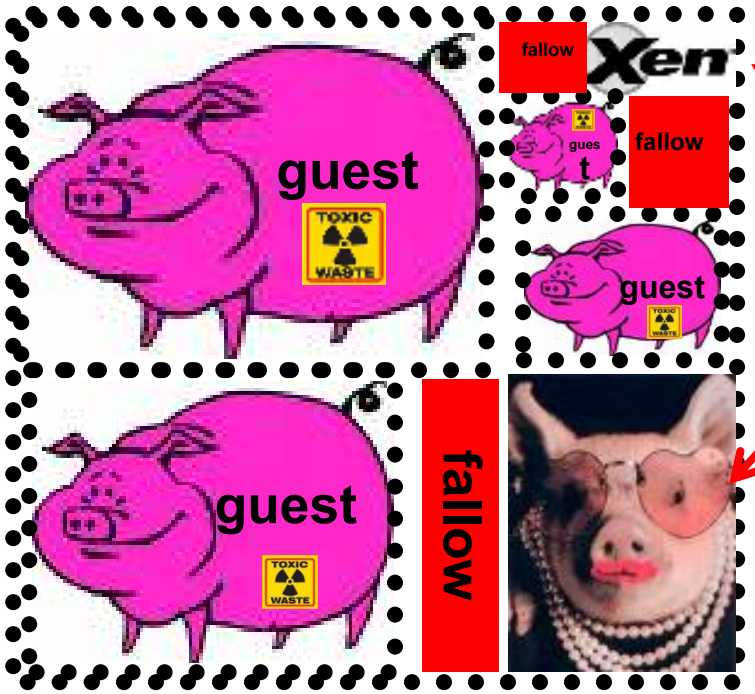
… some of which will be useful in the future

… and some (or maybe most…) of which is **wasted**

Agenda

- Motivation and Challenge
- BRIEF Overview of Physical Memory Management
  - in an operating system
  - *in a virtual machine monitor (Xen)*
- BRIEF Transcendent Memory ("tmem") Overview
- Tmem Progress since Xen Summit 2009
- Self-ballooning + Tmem Performance Analysis

VMM Physical Memory Management



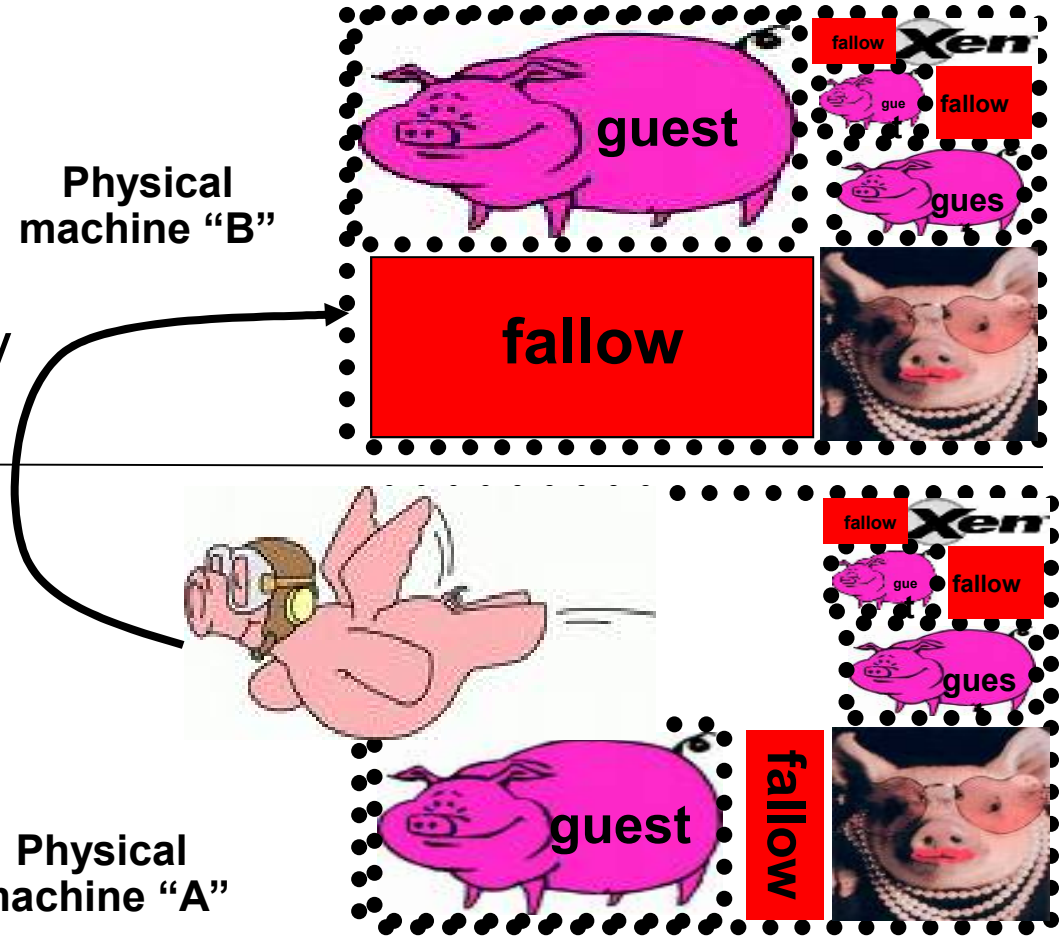
- Xen partitions memory among more guests
  - Xen memory
  - dom0 memory
  - guest 1 memory
  - guest 2 memory
  - guest 3…
- BUT still *fallow memory* leftover

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

VMM Physical Memory Management

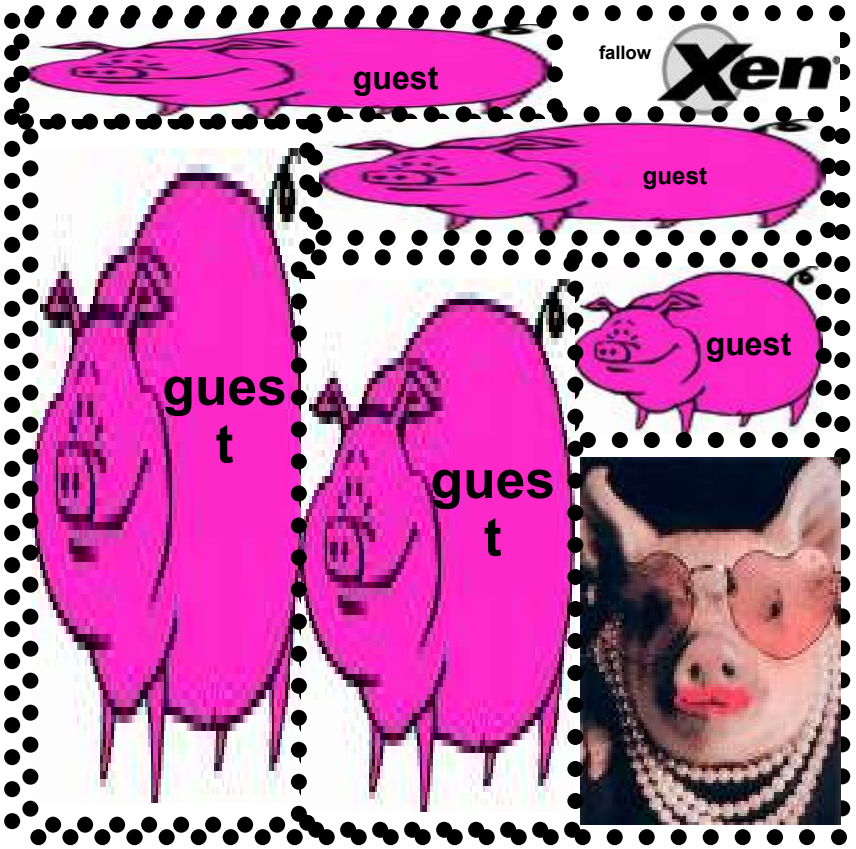
in the presence of migration

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



VMM Physical Memory Management

in the presence of ballooning



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

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

VMM Physical Memory Management

in the presence of ballooning

But *not* so great *to take memory away*

- not instantaneous (*memory inertia*)
- guest can't predict future needs
  - good pages are evicted along with the bad
- host doesn't know how much/fast to balloon
  - too much or too fast has dire results

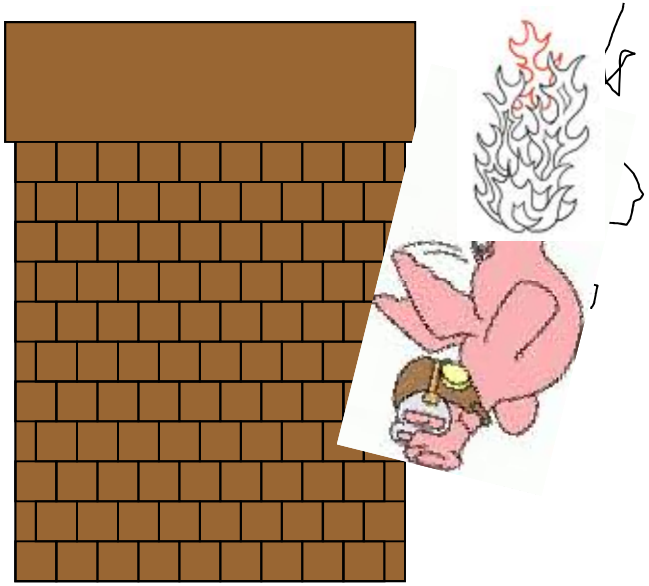
  → thrashing or the dreaded OOM killer

VMM Physical Memory Management

in the presence of migration AND ballooning

- Look ma! No more fallow memory!

….But now live migration crashes and burns



Why this IS a hard problem! Summary

- OS's use as much memory as they are given
  - but cannot predict the future so often guess wrong
  - and often much memory owned by an OS is wasted
- Xen leaves large amounts of memory fallow
  - fixed partitioning results in fragmentation
  - migration *requires* fallow memory to succeed
- Ballooning helps but:
  - can't predict future memory needs of guests
  - memory has inertia
  - the price of incorrect guesses can be dire

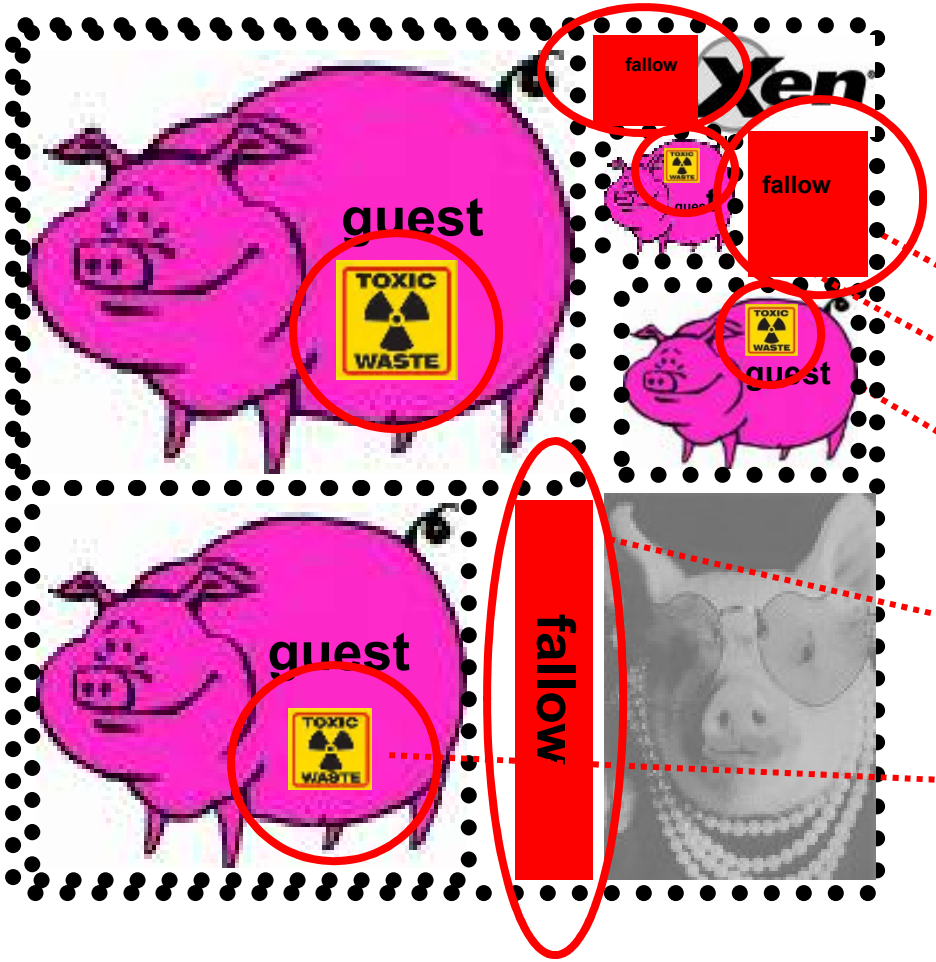
→ **NEED A NEW APPROACH TO VIRTUALIZED PHYSICAL MEMORY MANAGEMENT!!**

ORACLE

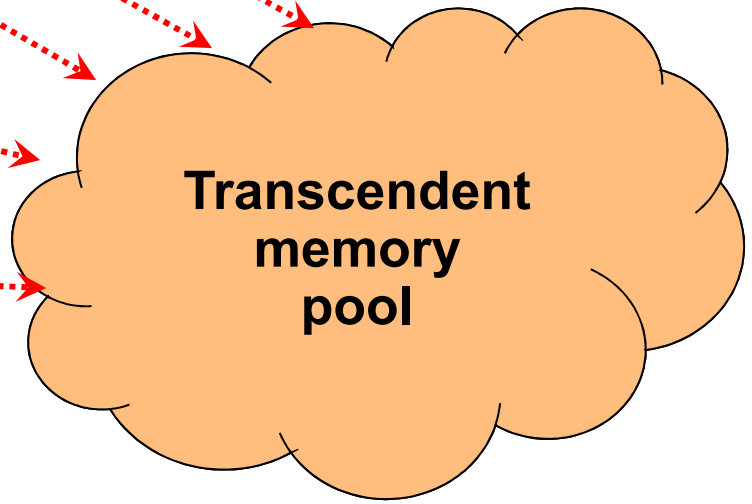
Agenda

- Motivation and Challenge
- BRIEF Overview of Physical Memory Management
- *BRIEF Transcendent Memory ("tmem") Overview*
- Tmem Progress since Xen Summit 2009
- Self-ballooning + Tmem Performance Analysis

Transcendent memory
creating the transcendent memory pool

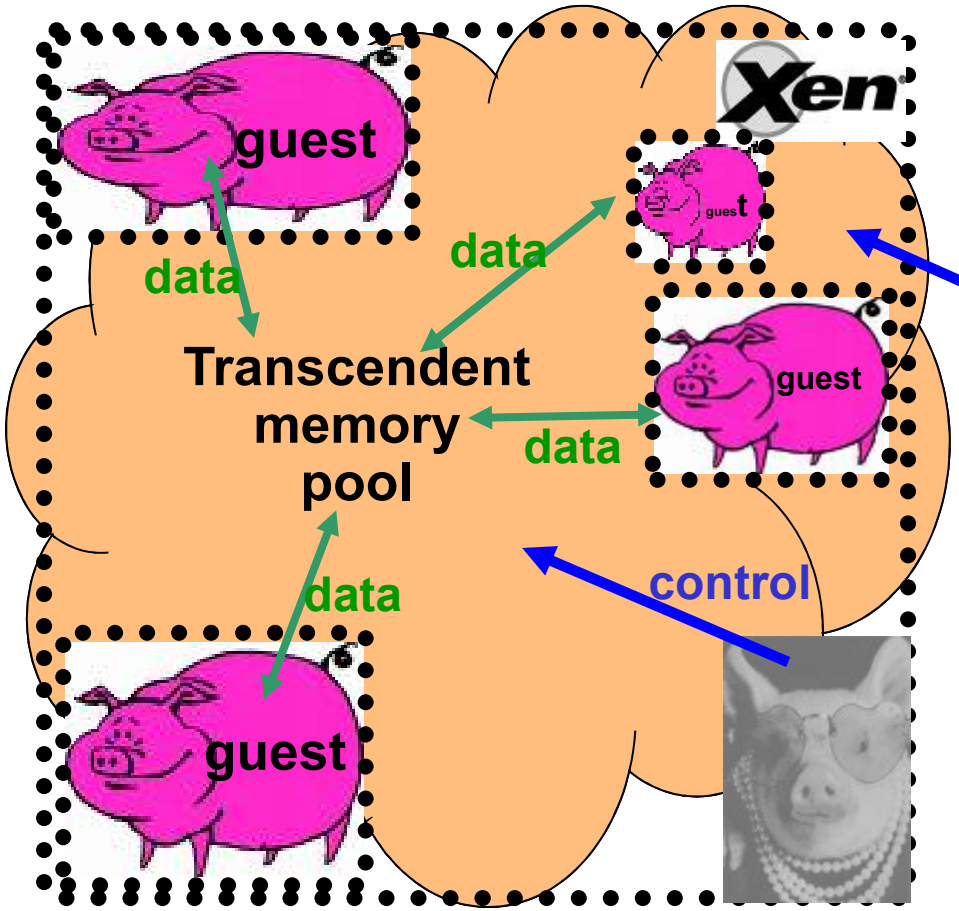


- **Step 1a**: reclaim *all* fallow memory
- **Step 1b**: reclaim wasted guest memory (e.g. via self-ballooning)
- **Step 1c**: collect it all into a pool



ORACLE

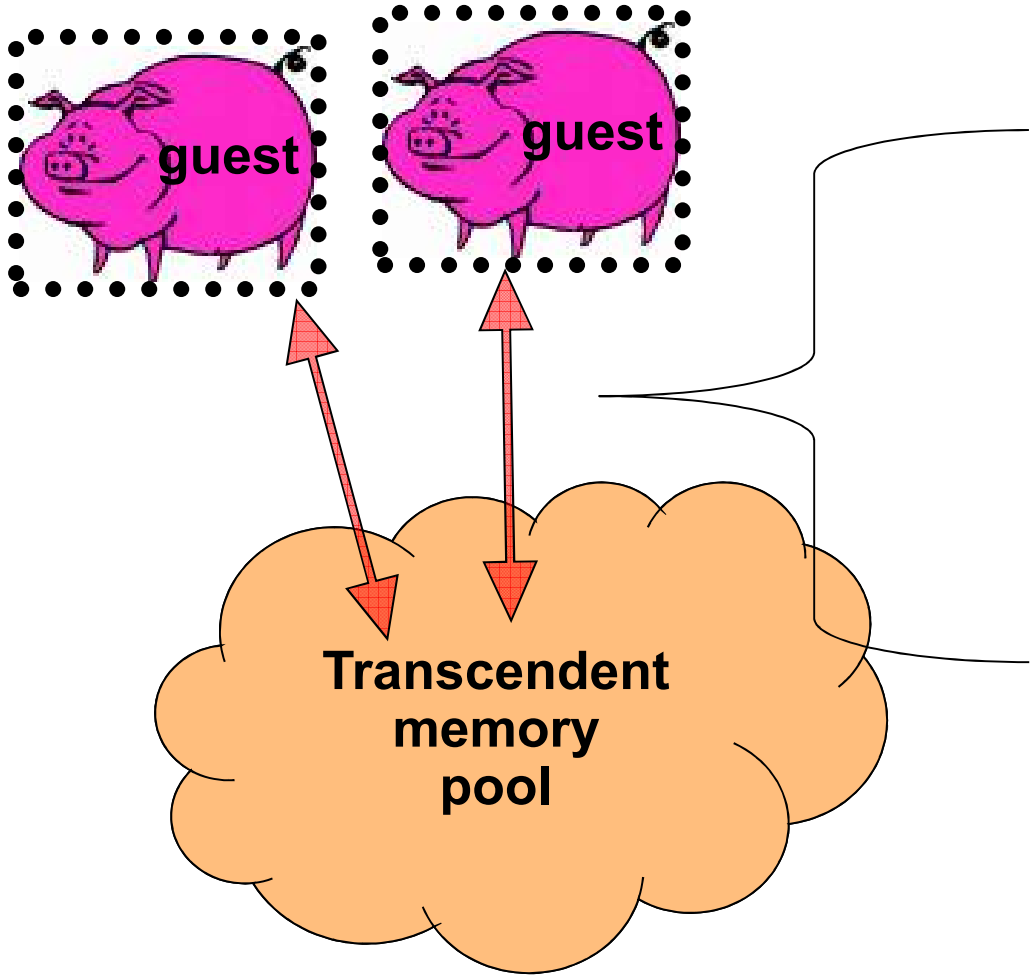
Transcendent memory
creating the transcendent memory pool



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

control

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)

Now in 2.6.32 patch

Under investigation

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

Agenda

- Motivation and Challenge
- BRIEF Overview of Physical Memory Management
- BRIEF Transcendent Memory ("Tmem") Overview
- *Tmem Progress since Xen Summit 2009*
- Self-ballooning + Tmem Performance Analysis

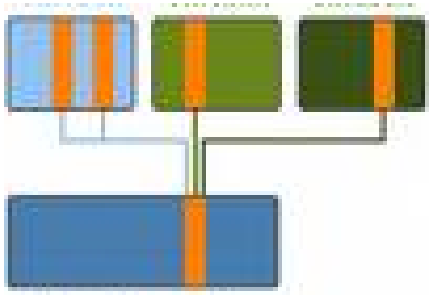
Transcendent Memory
update since Xen Summit 2009 (Feb'09)

- Tmem support officially released in Xen 4.0.0
  - Xen boot option: `tmem`
- Enterprise-quality concurrency
- Complete save/restore and live migration support
- Page deduplication support *(post-4.0.0)*
- Linux-side patches posted, including
  - ocfs2, btrfs, ext4 filesystem support (was ext3 only)
  - sysfs support for in-guest tmem statistics
- Other tmem releases:
  - Oracle VM 2.2 (10/2009)
  - OpenSuSE 11.2 (11/2009); SLE11 SP1 later this year
  - Oracle Enterprise Linux 5 update 4/5 rpm's available soon
  - targeting upstream Linux 2.6.35 (aka *cleancache* and *frontswap*)

ORACLE

Tmem page deduplication

- Now in xen-unstable (for 4.1) and xen-4.0-testing soon (for 4.0.1)
  - Xen boot option: `tmem_dedup`
- Similar in intent to "page sharing" but
  - very different in implementation
  - completely transparent to tmem-enabled guests
  - neither copy-on-write nor host swapping required
  - all tmem ephemeral pages automatically are sharing candidates



- Can optionally be combined with:
  - compression (`tmem_compress`); or
  - "trailing zero elimination" (`tmem_tze`)
- Statistics available through existing "xm tmem-list"
  - e.g., dedup+compression increase time per tmem op by ~10x

Known outstanding issue

Fragmentation! *(ominous music plays here…)*

- Xen page allocator allows "order>0" allocations
  - order==1 → 2 contiguous pages, order==2 → 4 contiguous pages, etc.
- Some Xen features depend on order>0 allocations
  - some are resilient and fall back to multiple one-page allocations
  - some fail badly if not available (e.g., shadow code, domain creation)
- Ballooning+tmem quickly fragments all Xen RAM
- Hacky workaround in place for 4.0, but…

CALL-TO-ACTION:

Post-dom0-boot multi-page allocations are fundamentally incompatible with *all* dynamic memory technologies and must either be made resilient or eliminated from Xen!

Agenda

- Motivation and Challenge
- BRIEF Overview of Physical Memory Management
- BRIEF Transcendent Memory ("Tmem") Overview
- Tmem Progress since Xen Summit 2009
- *Self-ballooning + Tmem Performance Analysis*

Test workload (overcommitted!)

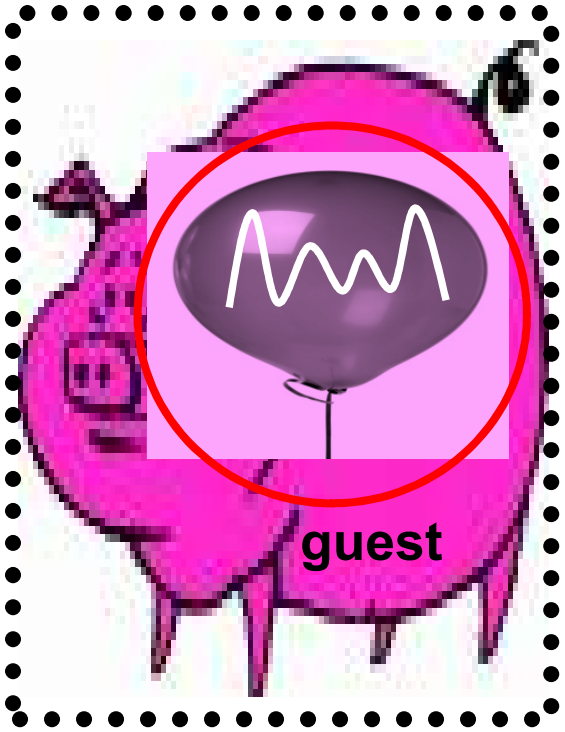
- Dual core (Conroe) processor, 2GB RAM, IDE disk
- Four single vcpu PV VMs, in-kernel self-ballooning+tmem
  - Oracle Enterprise Linux 5 update 4; two 32-bit + two 64-bit
  - mem=384MB (maxmem=512MB)… total = 1.5GB (2GB maxmem)
  - virtual block device is tap:aio (file contains 3 LVM partitions: ext3+ext3+swap)
- 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

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

Self-ballooning recap (see Xen Summit 2008)

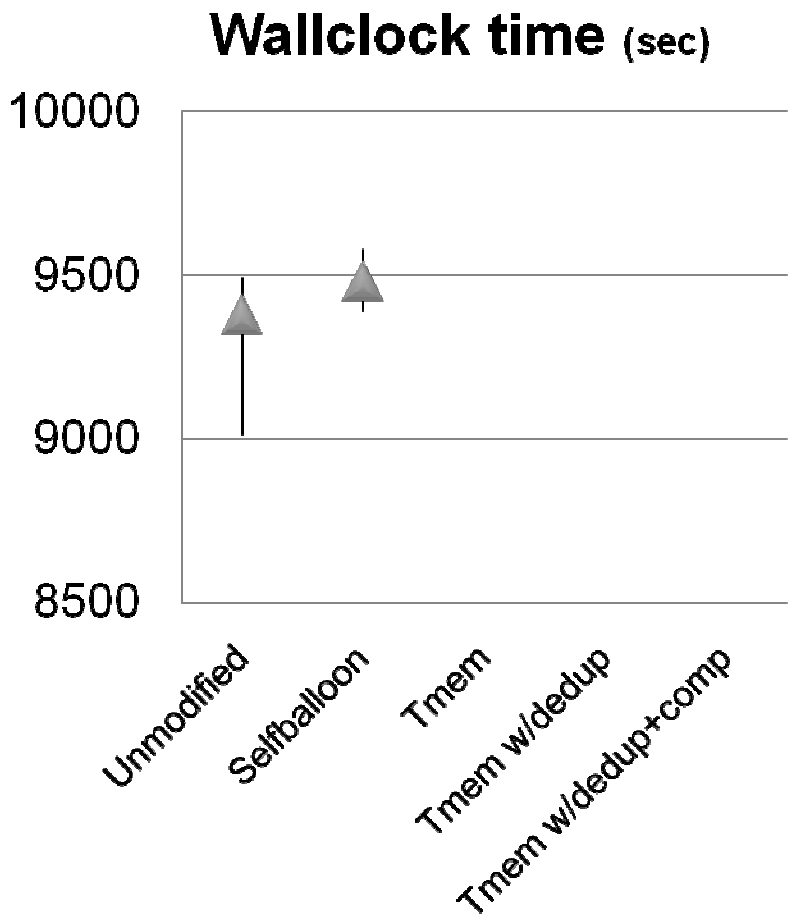


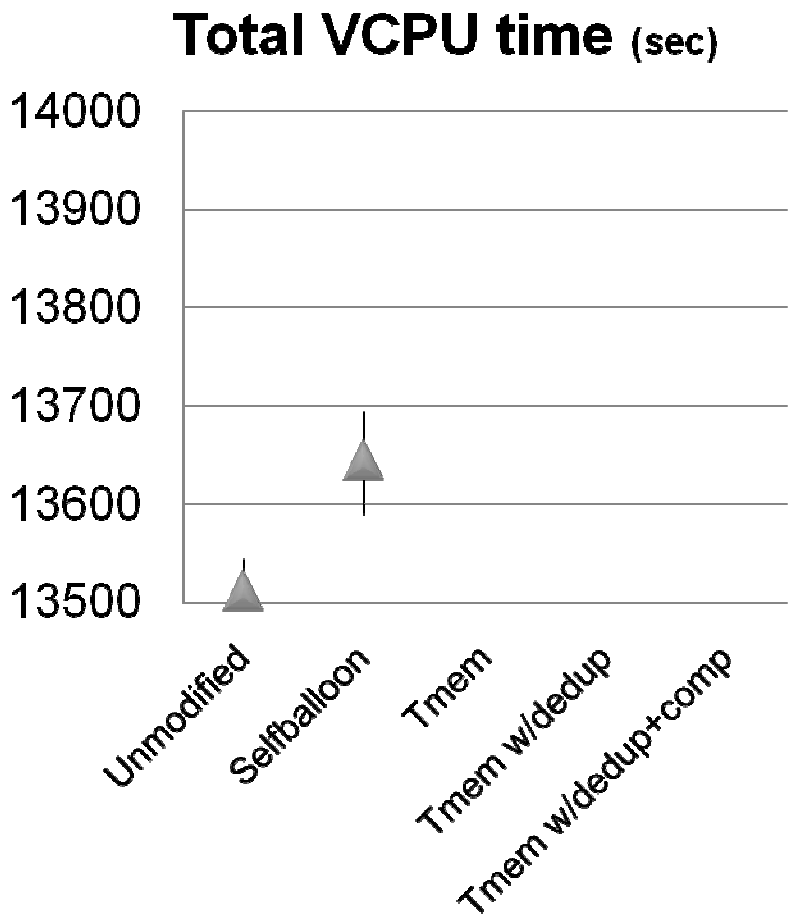
- Goal: Allow memory overcommit
- Use in-guest feedback to resize balloon
  - aggressively
  - frequently
  - independently
  - configurably
- For Linux, size to maximum of:
  - /proc/meminfo “CommittedAS”
  - memory floor enforced by Xen balloon driver
- Userland daemon *or* patched kernel

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**)

Unchanged vs. Self-ballooning only

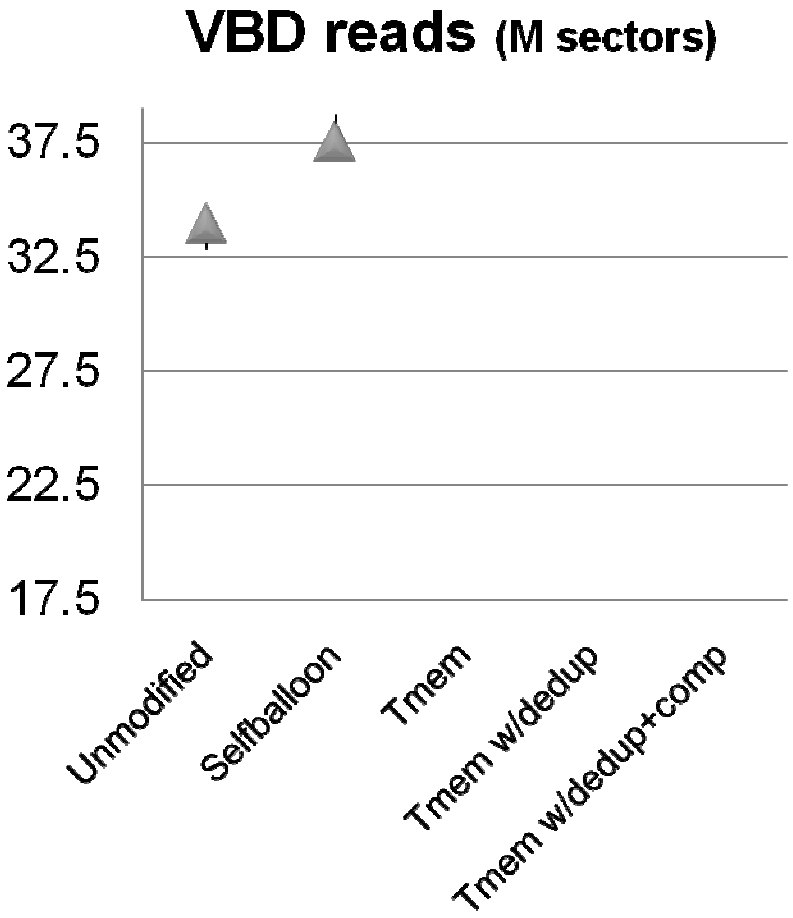
Temporal stats

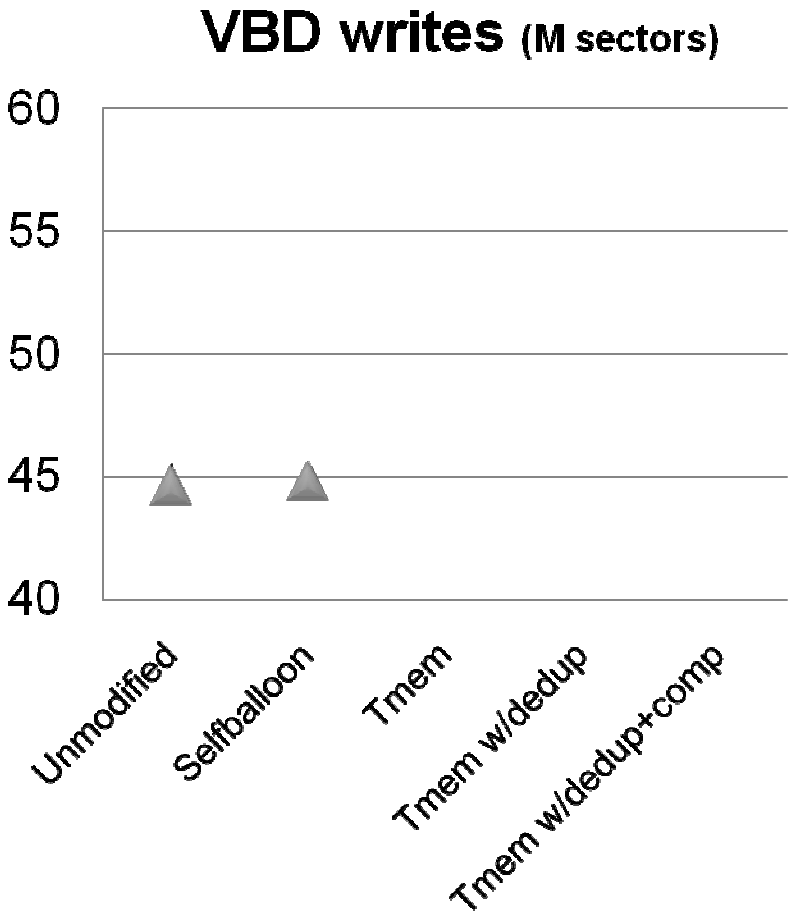




Unchanged vs. Self-ballooning only

Virtual block device stats





Sigh Why is there a performance hit?

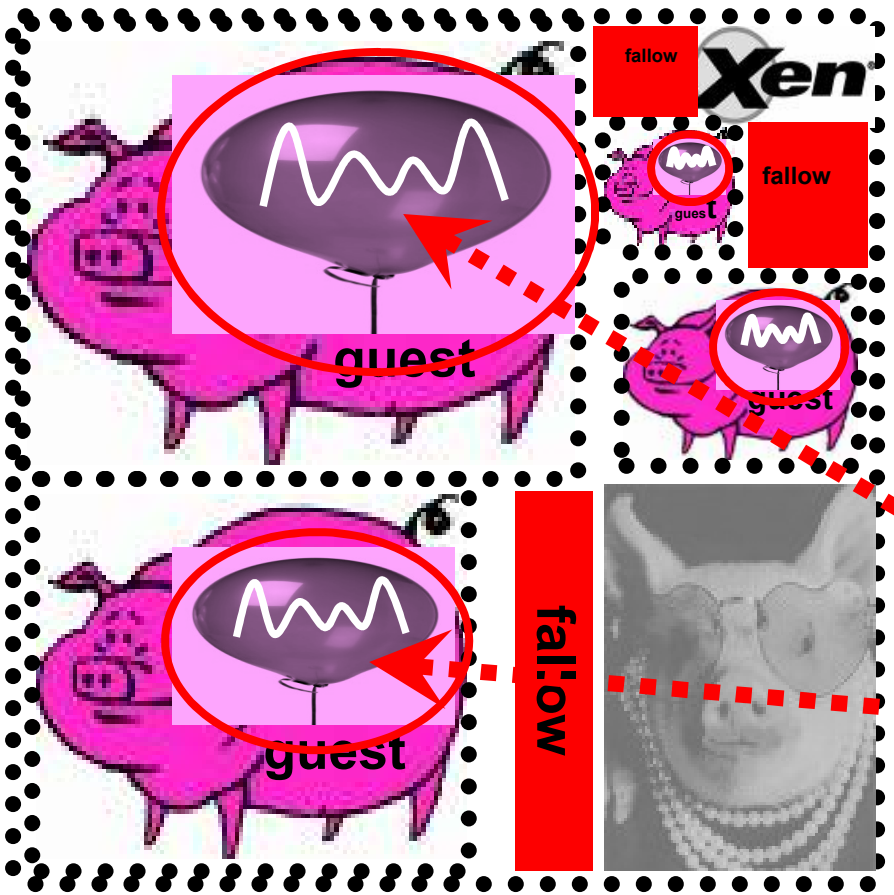
Aggressive ballooning (*by itself)* doesn't work very well!

- (Self-)ballooning *indiscriminately* shrinks the guest OS's page cache, causing *refaults*!
- Insufficiently responsive (self-)ballooning when guest OS needs memory ***now***, results in swapping… or OOMs!

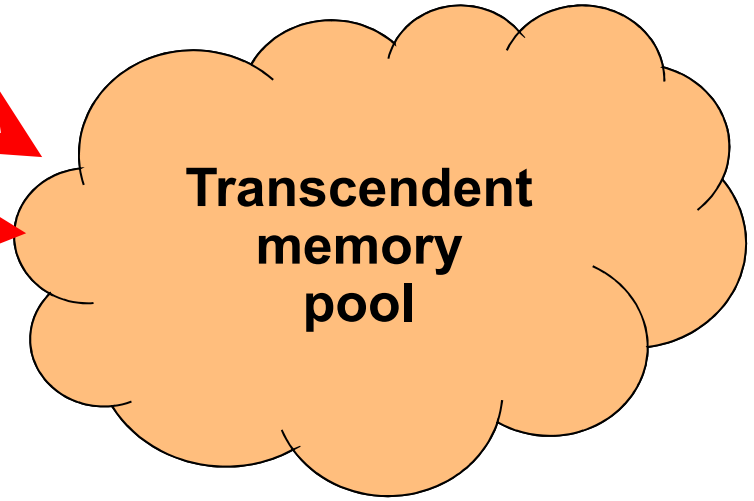
→ PERFORMANCE **WILL GET WORSE** WHEN LARGE-MEMORY GUESTS ARE AGGRESSIVELY BALLOONED

Self-ballooning AND Transcendent Memory

…go together like a horse and carriage



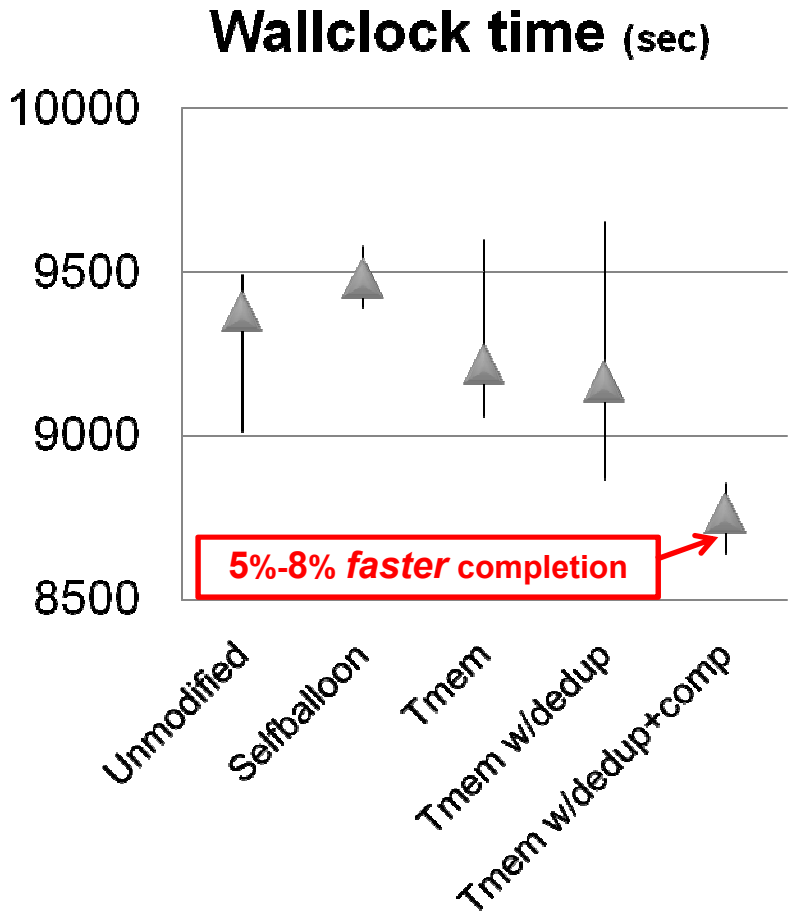
- Self-ballooned memory is returned to Xen and absorbed by tmem
- Most tmem memory can be *instantly* reclaimed when needed for a memory-needy or new guest
- Tmem also provides a *safety valve* when ballooning is not fast enough

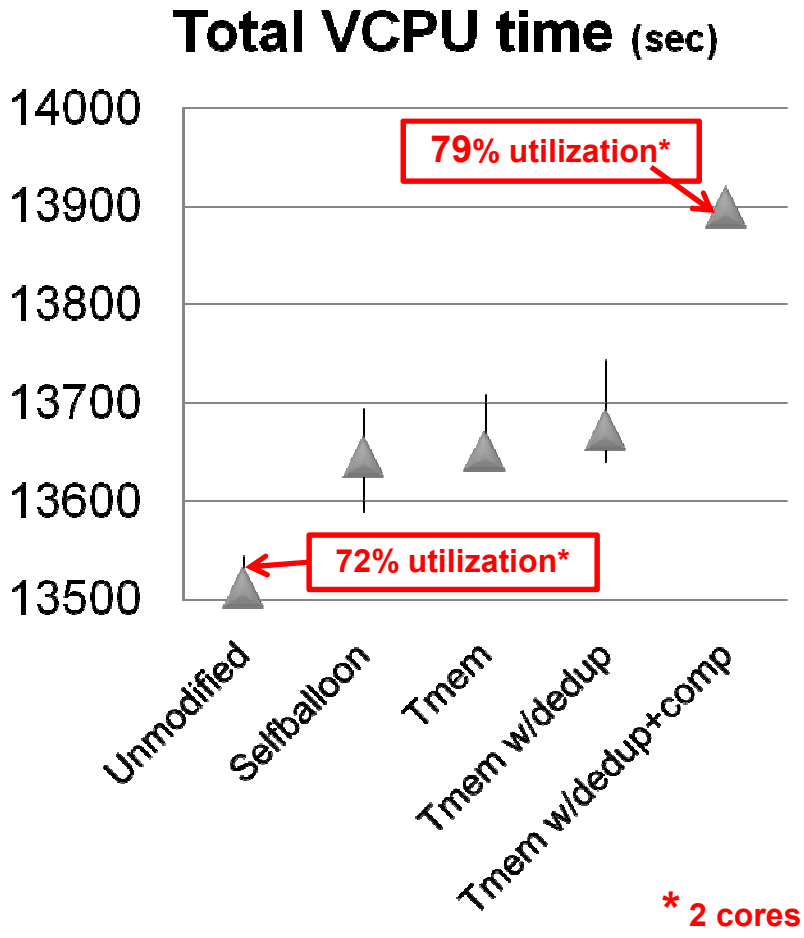


ORACLE

Self-ballooning AND Tmem

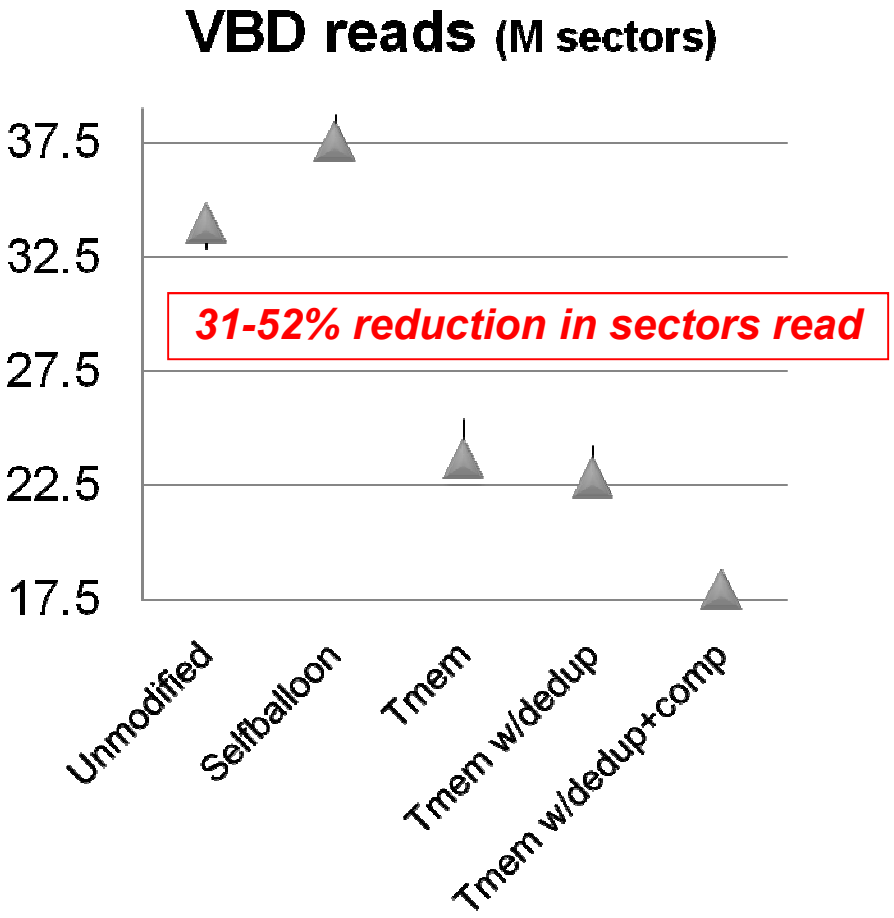
Temporal stats

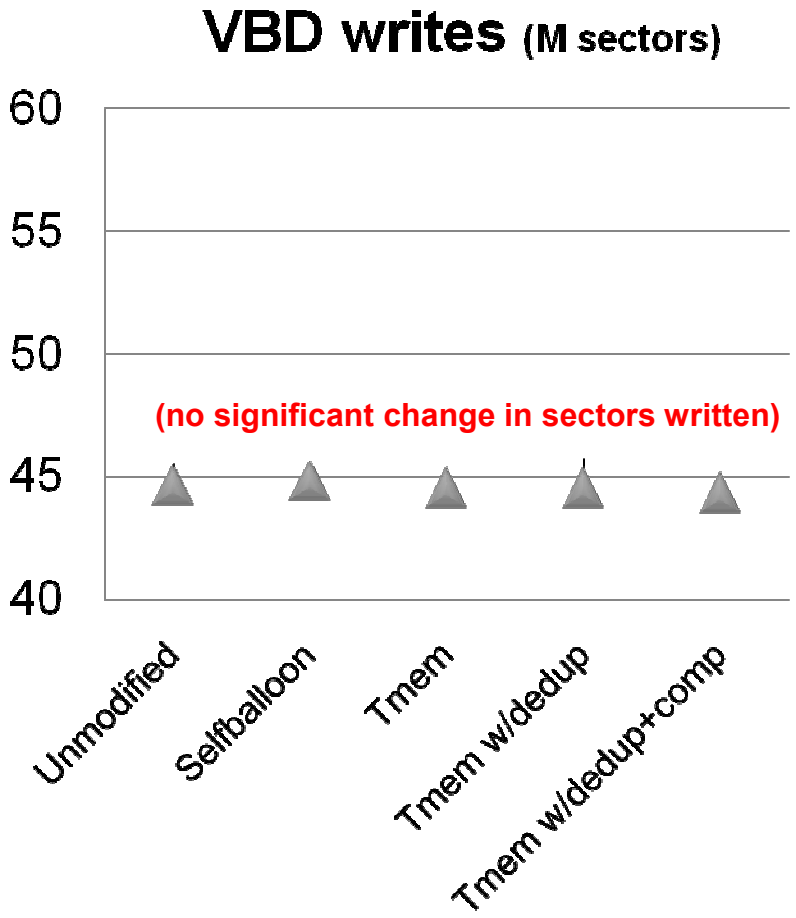




Self-ballooning AND Tmem

virtual block device stats





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

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 (so only Linux now)
- higher power consumption due to higher CPU utilization

Transcendent Memory Update Acknowledgements

Special thanks to Jeremy Fitzhardinge for many reviews and improvements in the Linux-side tmem code!

For more information

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

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

dan.magenheimer@oracle.com