Transcendent Memory
and Friends
(Not just for virtualization anymore!)

Dan Magenheimer,
Oracle Corp.

ORACLE

Transcendent Memory's "Friends"

zcache

RAMster

frontswap

cleancache

SSmem

page-accessible memory

persistent

ephemeral

handle

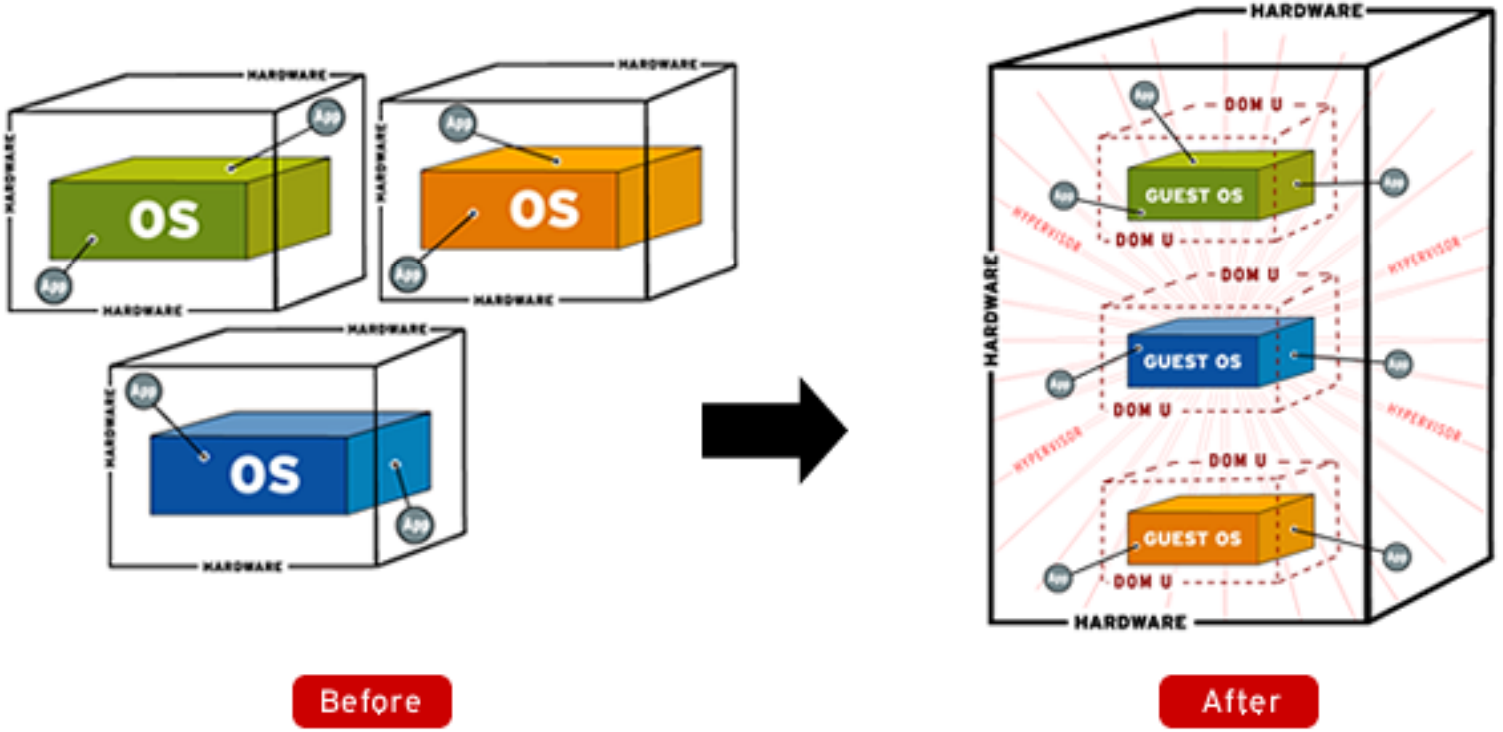
Transcendent Memory
Objectives:

- ***Utilize RAM more efficiently*** **to obtain**
  - **--*Lower capital costs* in the data center**
  - **--*Lower power utilization* in the data center**
  - **--Less I/O resulting in *better performance* on many workloads**

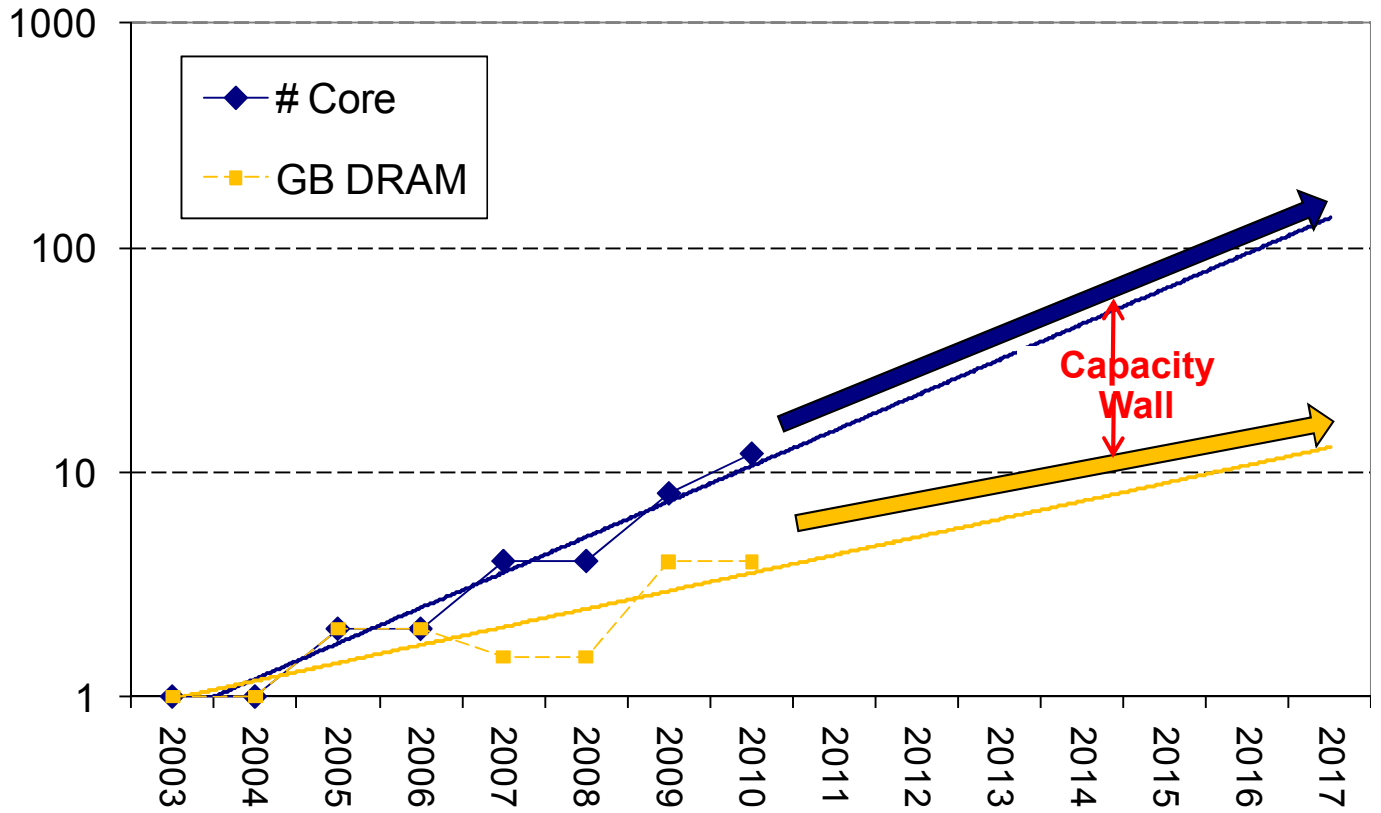
    (with negligible loss on other workloads)

ORACLE

Motivation: Memory-inefficient workloads



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

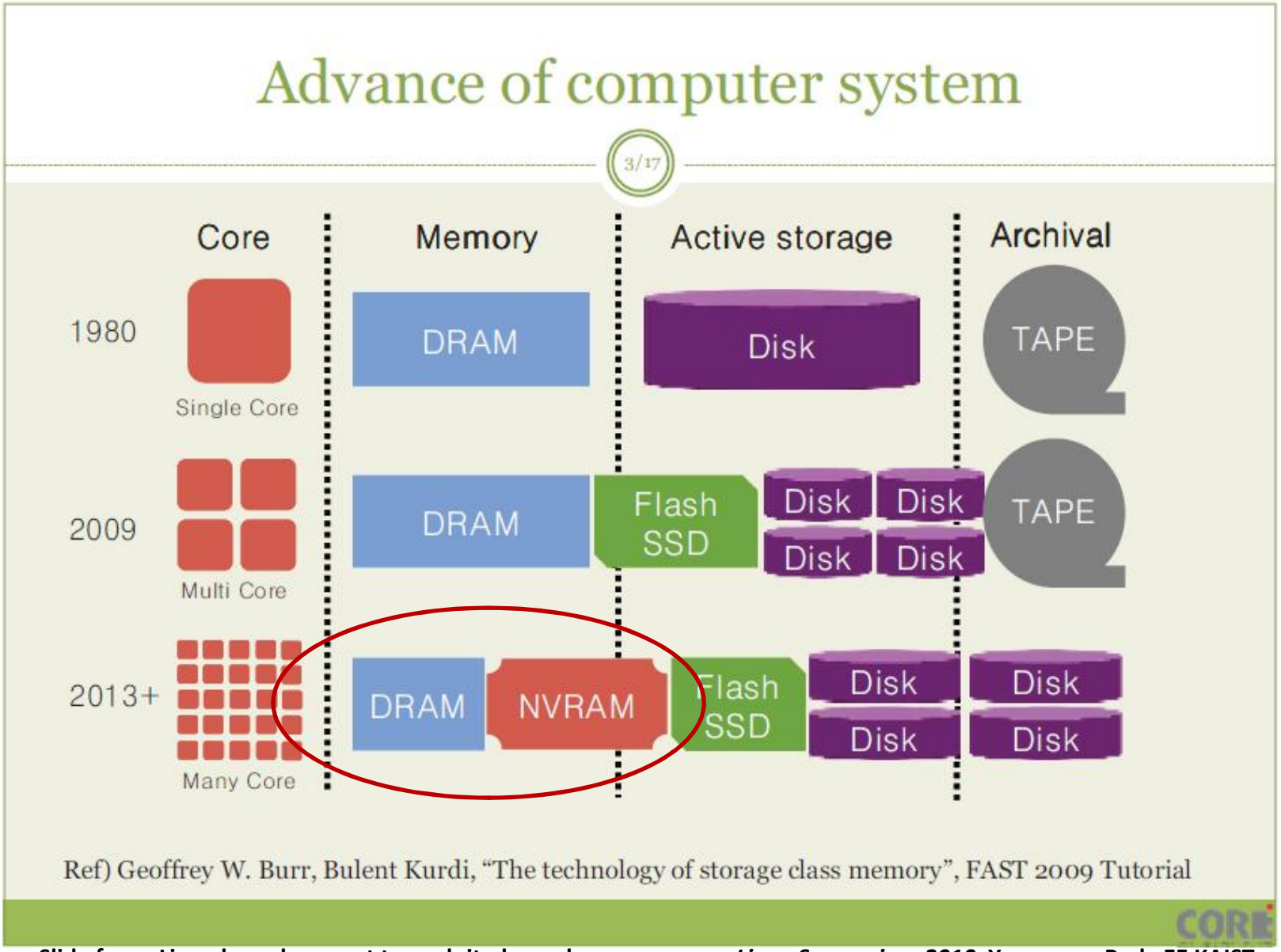
More motivation: Energy Savings

"…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

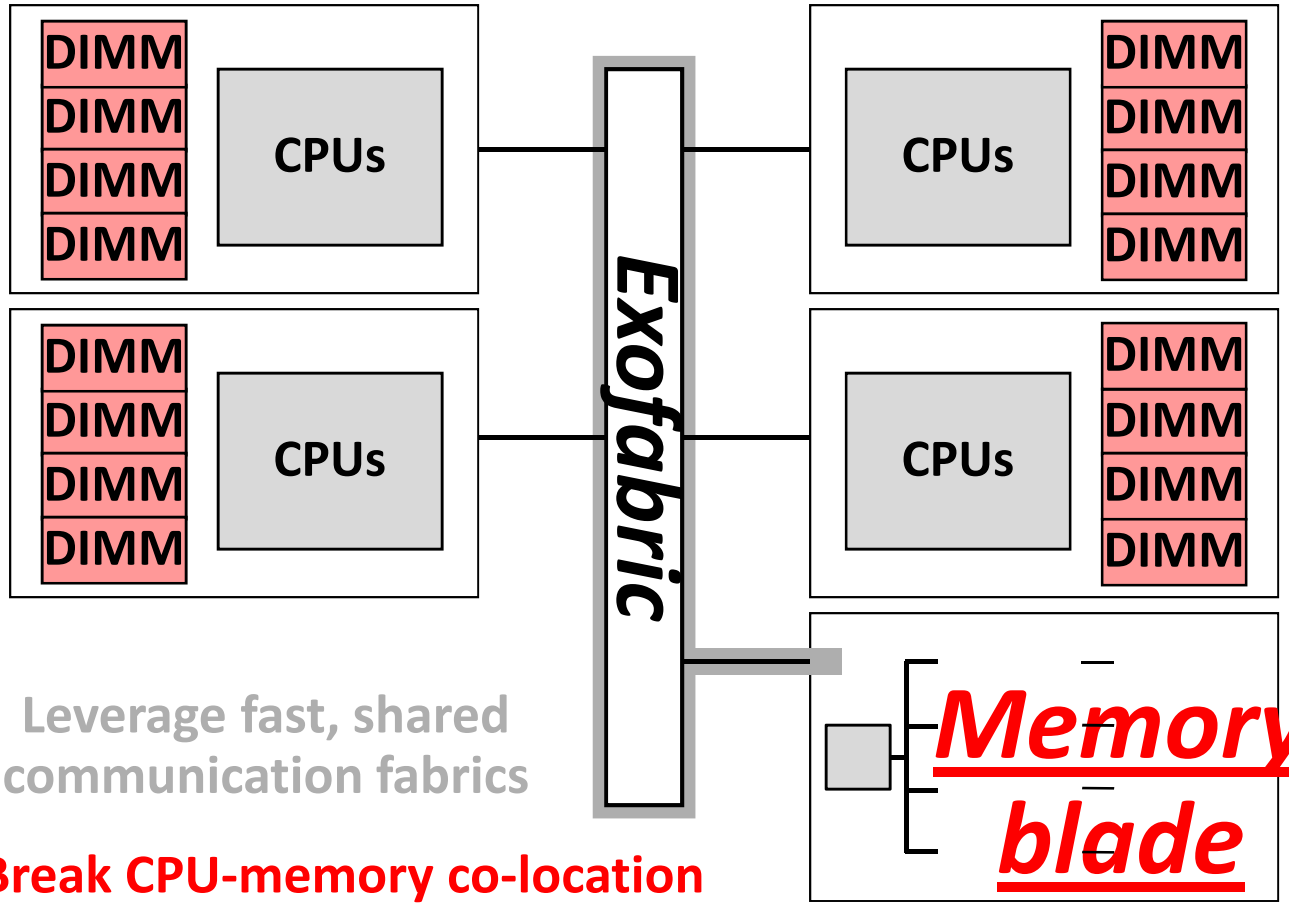
Source: Disaggregated Memory Architectures for Blade Servers, Kevin Lim, Univ Michigan, PhD Thesis

ORACLE



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

Disaggregated memory concept



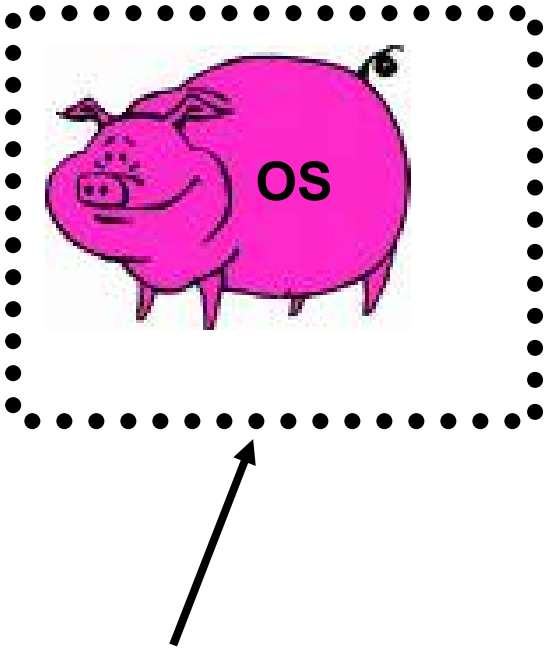
Leverage fast, shared communication fabrics

⇨ **Break CPU-memory co-location**

Source: Disaggregated Memory for Expansion and Sharing in Blade Server

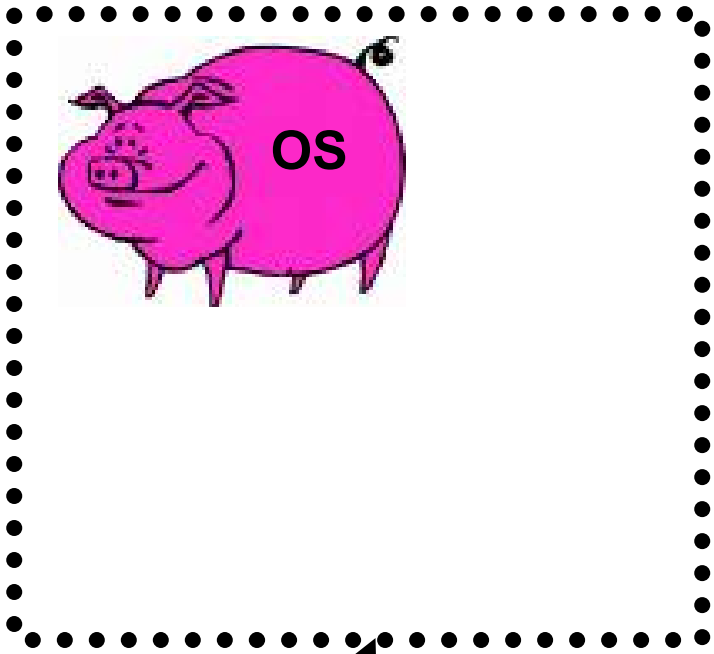
http://isca09.cs.columbia.edu/pres/24.pptx

OS Memory "Demand"



- Operating systems are memory hogs!

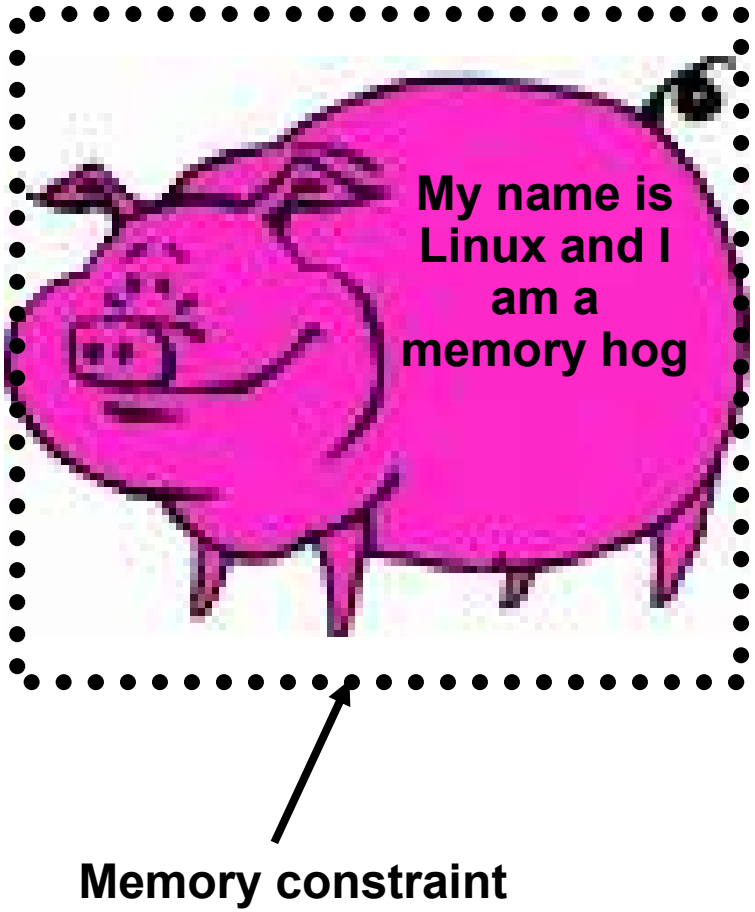
OS Physical Memory Management



- Operating systems are memory hogs!

If you give an operating system more memory…..

OS Physical Memory Management



- Operating systems are memory hogs!

…it uses up any memory you give it!

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"

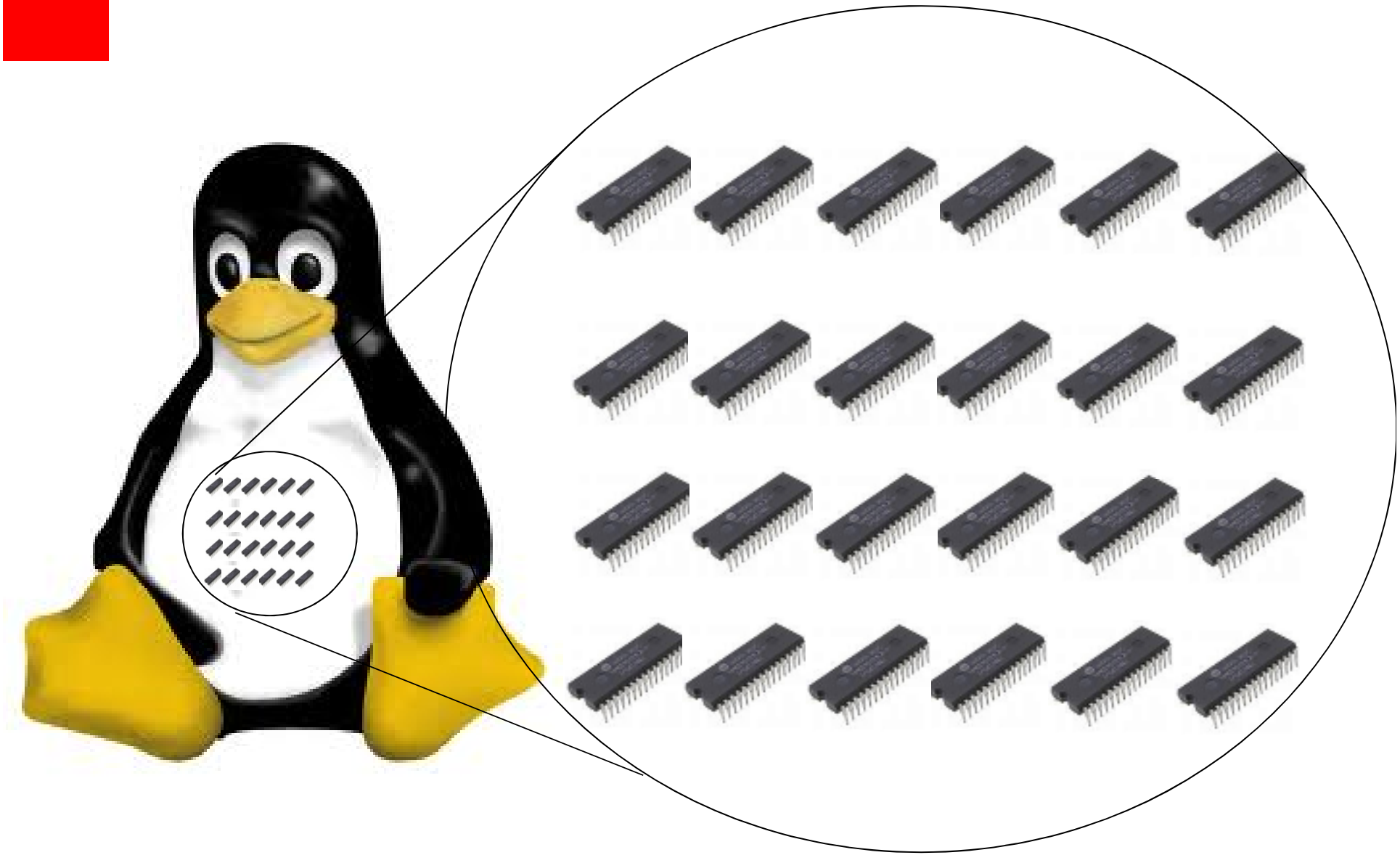
SUPPOSE there is a *mechanism* for the OS *to* ***ask for*** and obtain a page (or more) of RAM when it ***needs*** more RAM than it currently has

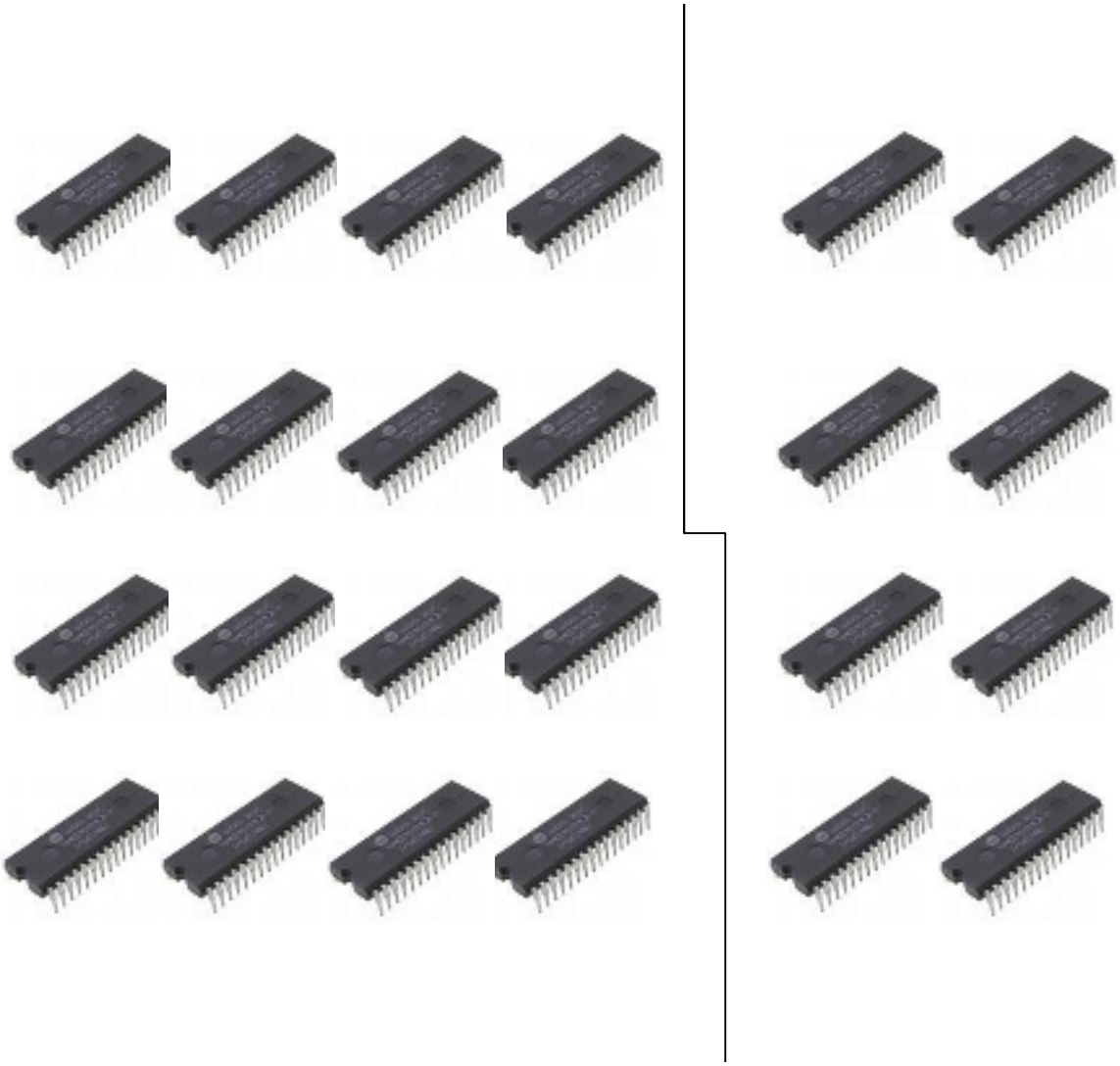
THEN… **HOW** does an OS decide

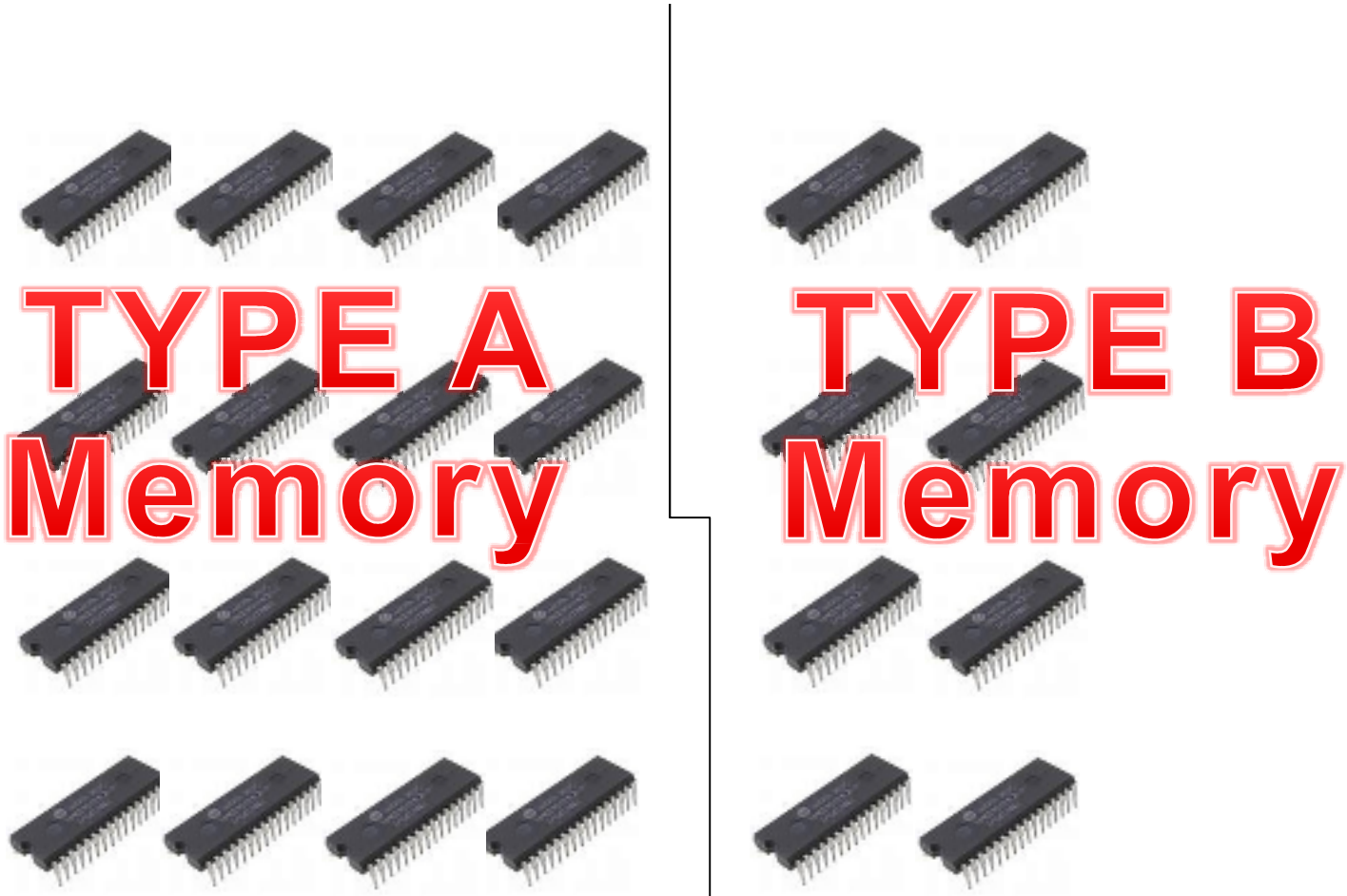
how much RAM it "***needs***"?

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






TYPE A
Memory
TYPE B
Memory

TYPE A Memory

ORACLE

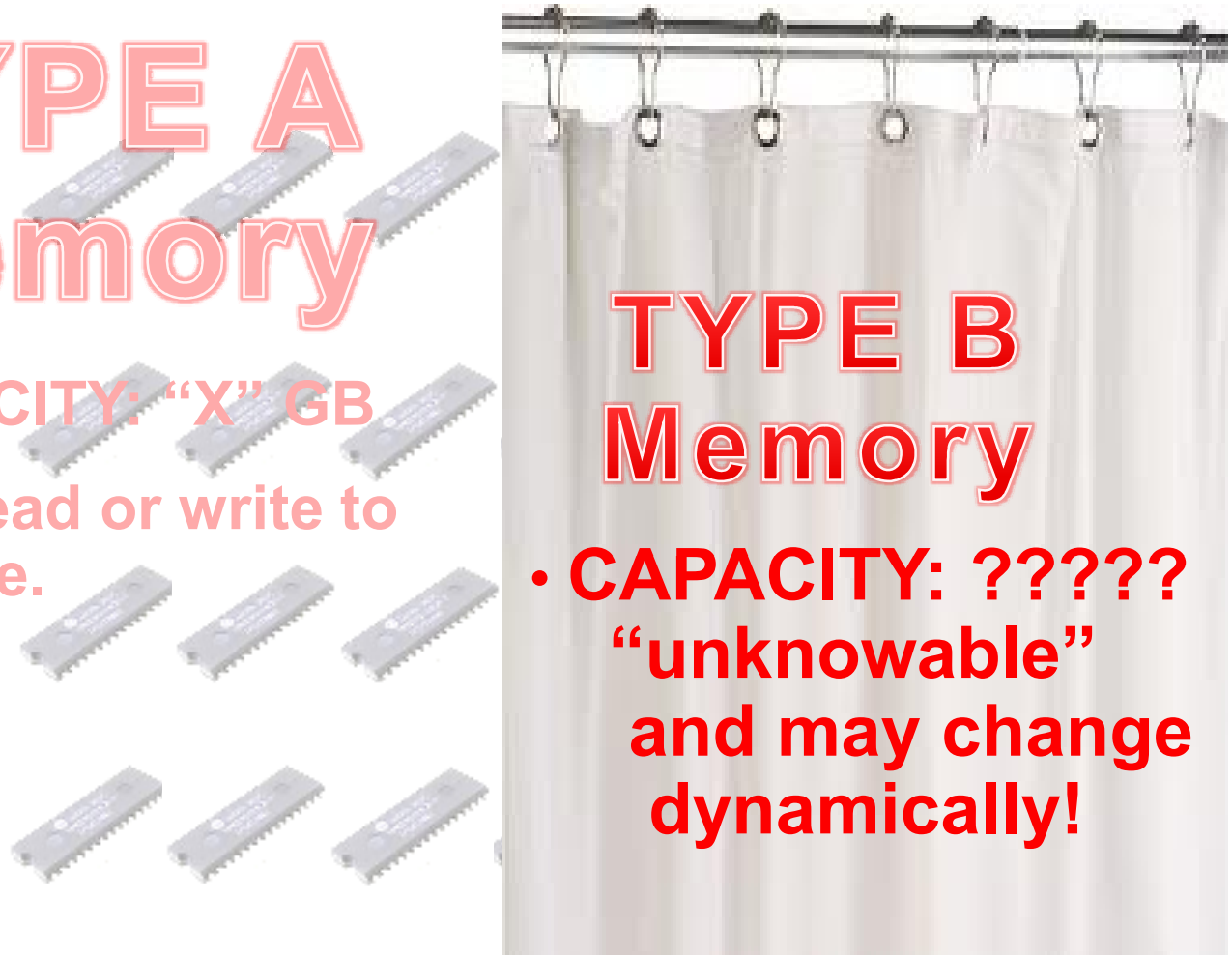
TYPE A Memory

- CAPACITY: “X” GB
- Can read or write to any byte.

ORACLE

TYPE A Memory

- CAPACITY: "X" GB
- Can read or write to any byte.



TYPE B Memory

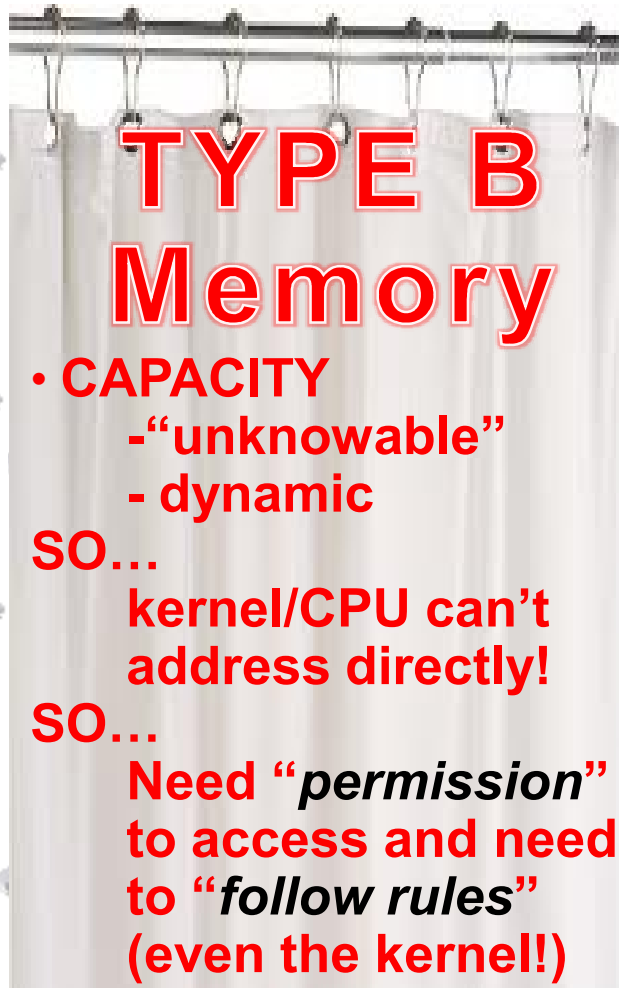
- CAPACITY: ????? "unknowable" and may change dynamically!

(Normal) RAM

- CAPACITY: known
- USES:
  - -kernel memory
  - -user memory
  - -DMA
  - -etc
- ADDRESSABILITY:
  - Read/write any byte

(Normal) RAM

- CAPACITY: known
- USES:
  - -kernel memory
  - -user memory
  - -DMA
  - -etc
- ADDRESSABILITY:
  - Read/write any byte



TYPE B Memory

- CAPACITY
  - -"unknowable"
  - - dynamic

SO…

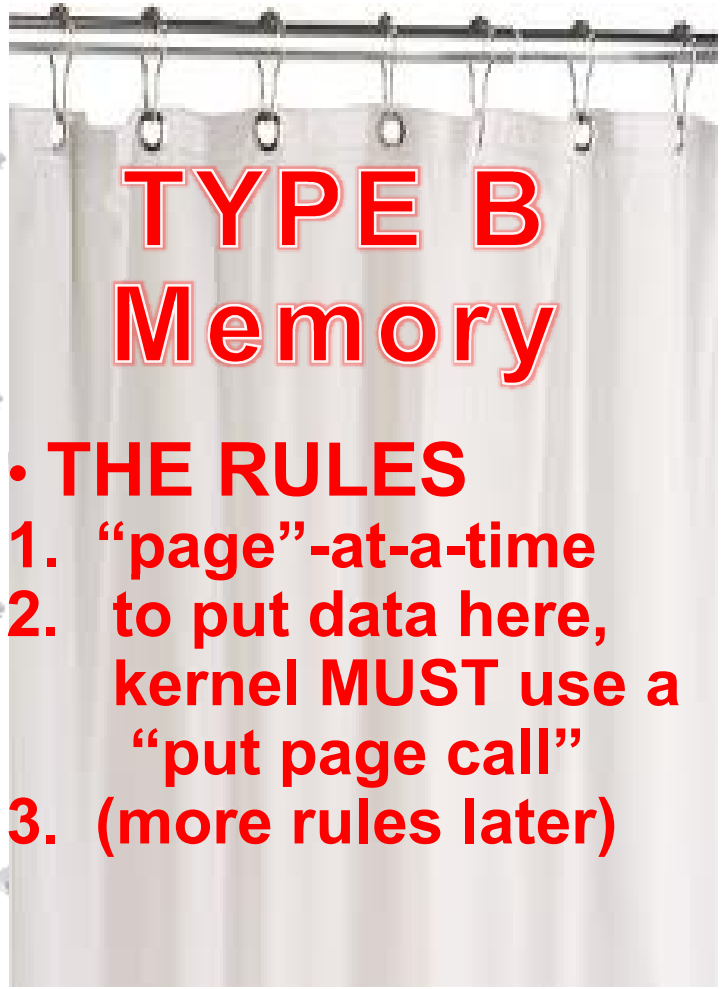
kernel/CPU can't address directly!

SO…

Need "*permission*" to access and need to "*follow rules*" (even the kernel!)

(Normal) RAM

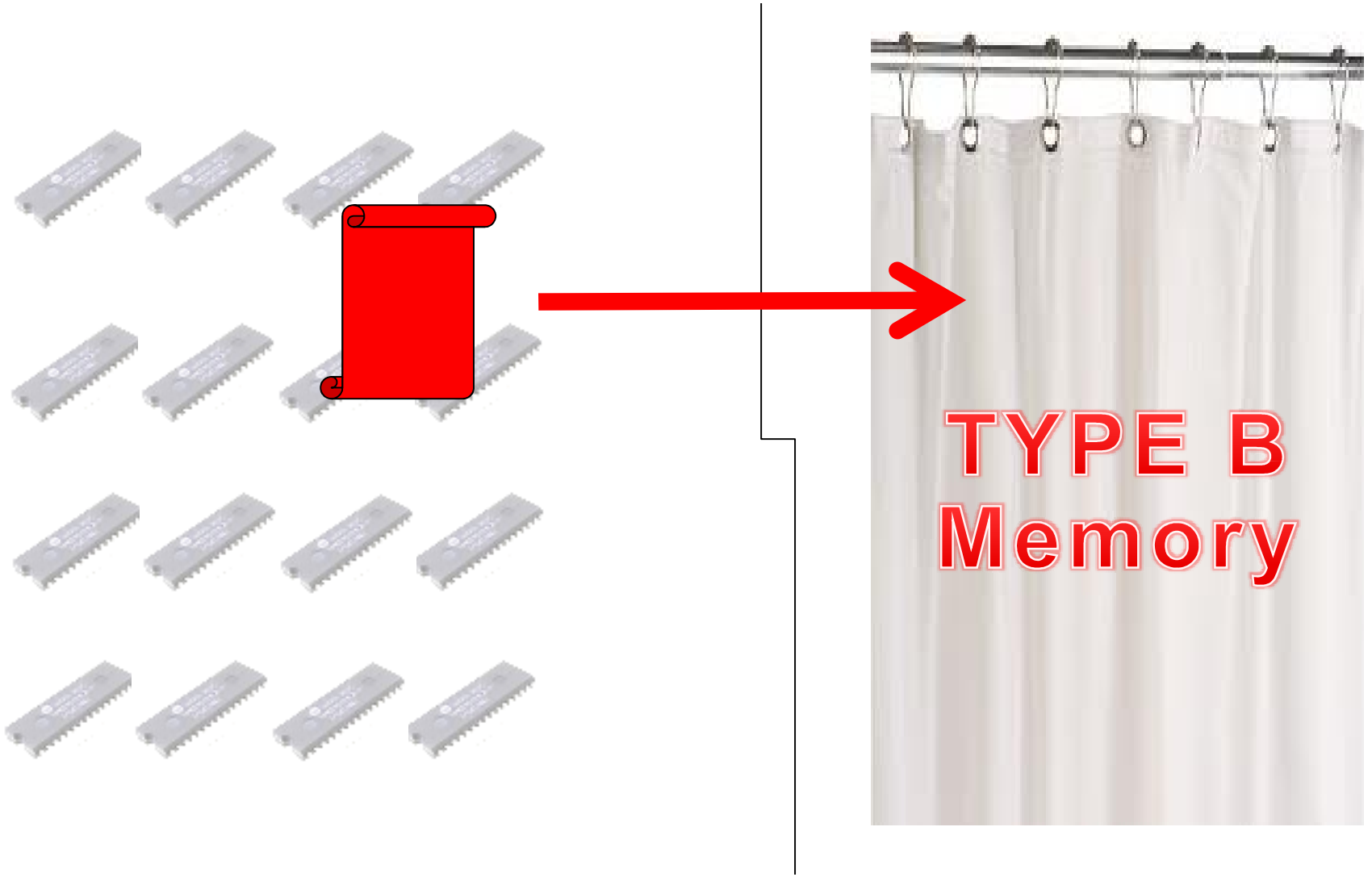
- CAPACITY: known
- USES:
  - -kernel memory
  - -user memory
  - -DMA
  - -etc
- ADDRESSABILITY: Read/write any byte



TYPE B Memory

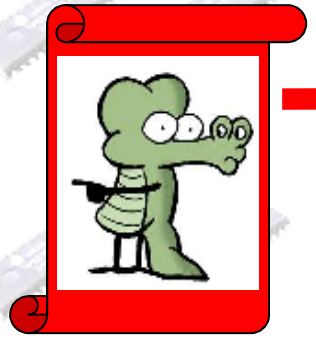
- THE RULES

1. "page"-at-a-time
2. to put data here, kernel MUST use a "put page call"
3. (more rules later)

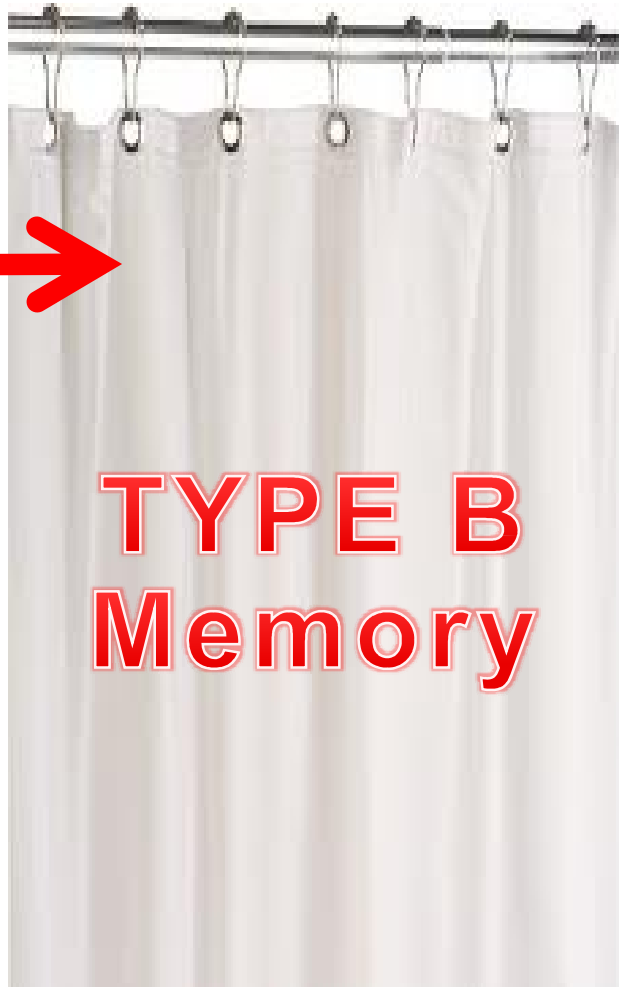

TYPE B
Memory

We have a page that contains:

Larry



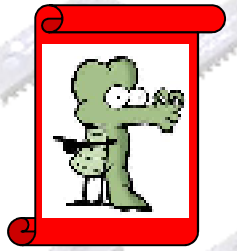
And the kernel wants to "preserve" Larry in Type B memory.



Note: All images of Larry the Crocodile are copyright Stephan Pastis
From the "Pearls Before Swine" comic strip dist. by United Feature Syndicate, Inc.

We have a page that contains:

Larry

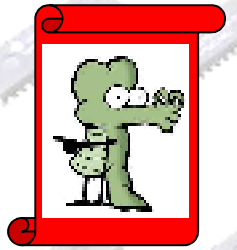


And the kernel wants to "preserve" Larry into Type B memory… *but…*

Kernel MUST ask permission and may get told NO!

We have a page that contains:

Larry

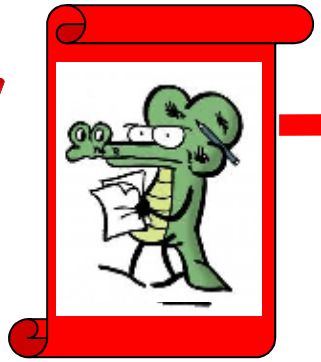


And the kernel wants to “preserve” Larry into Type B memory.

The kernel has *two choices*…

We have a page that contains:

Dirty Larry



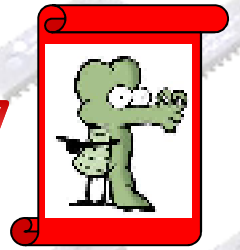
And the kernel wants to "preserve" Larry into Type B memory.

Two choices…

1. **DEFINITELY want Larry back**
   (e.g. "dirty" page)

We have a page that contains:

Clean Larry



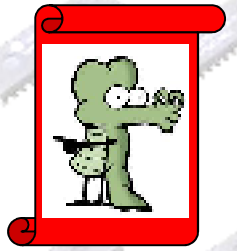
And the kernel wants to "preserve" Larry into Type B memory.

Two choices…

1. DEFINITELY want Larry back
2. PROBABLY want Larry back
   (e.g. "clean" pages)

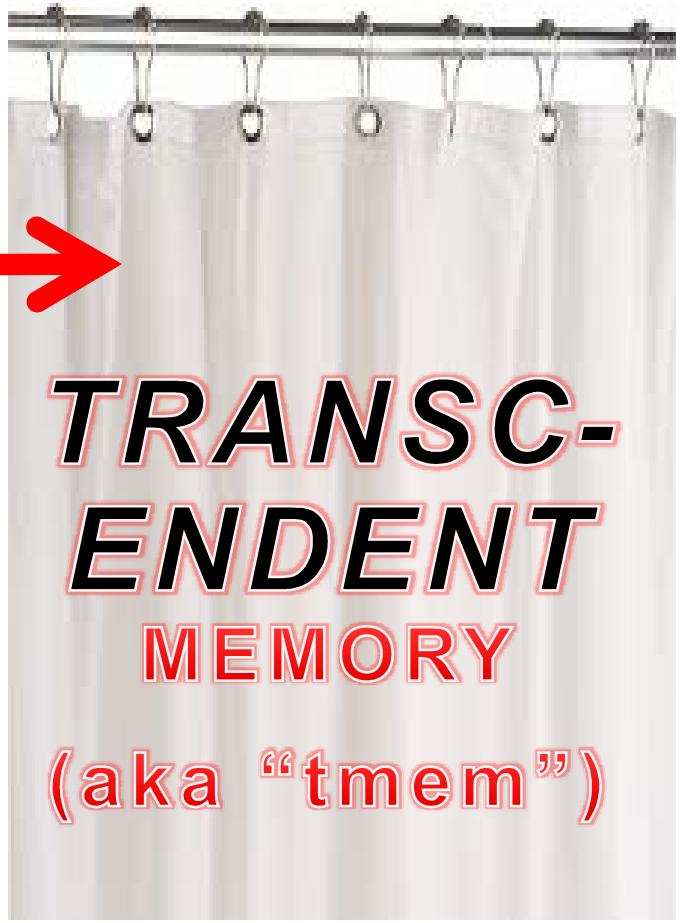
We have a page that contains:

Larry



Two choices…

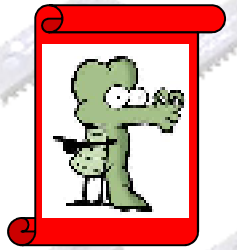
1. DEFINITELY want Larry back
2. PROBABLY want Larry back



tran-scend-ent, *adj., … beyond the range of normal perception*

We have a page that contains:

Larry

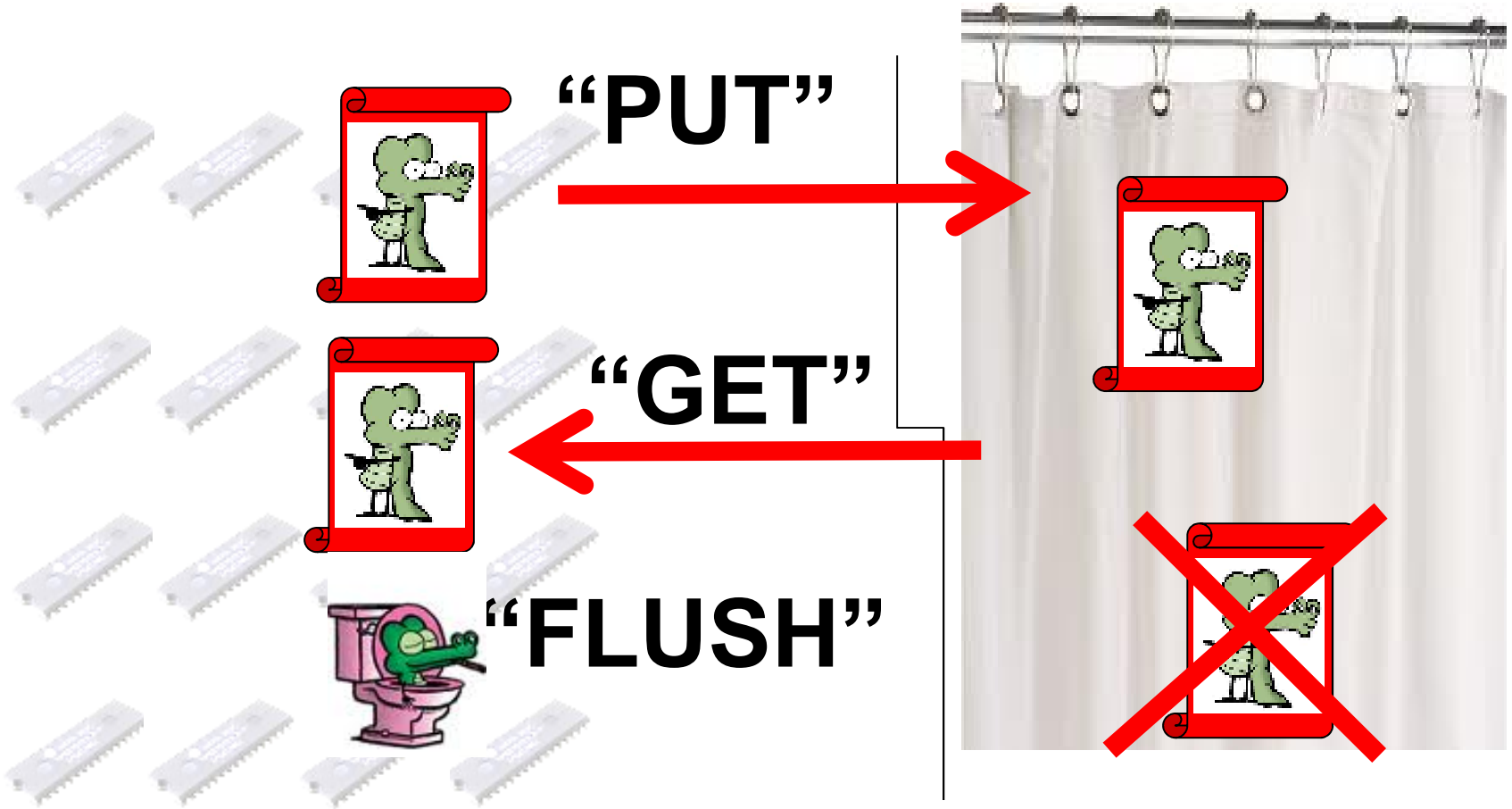


Two choices…

1. DEFINITELY want Larry back
   "*PERSISTENT* PUT"
2. PROBABLY want Larry back
   "*EPHEMERAL* PUT"

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

tran-scend-ent, *adj., … beyond the range of normal perception*



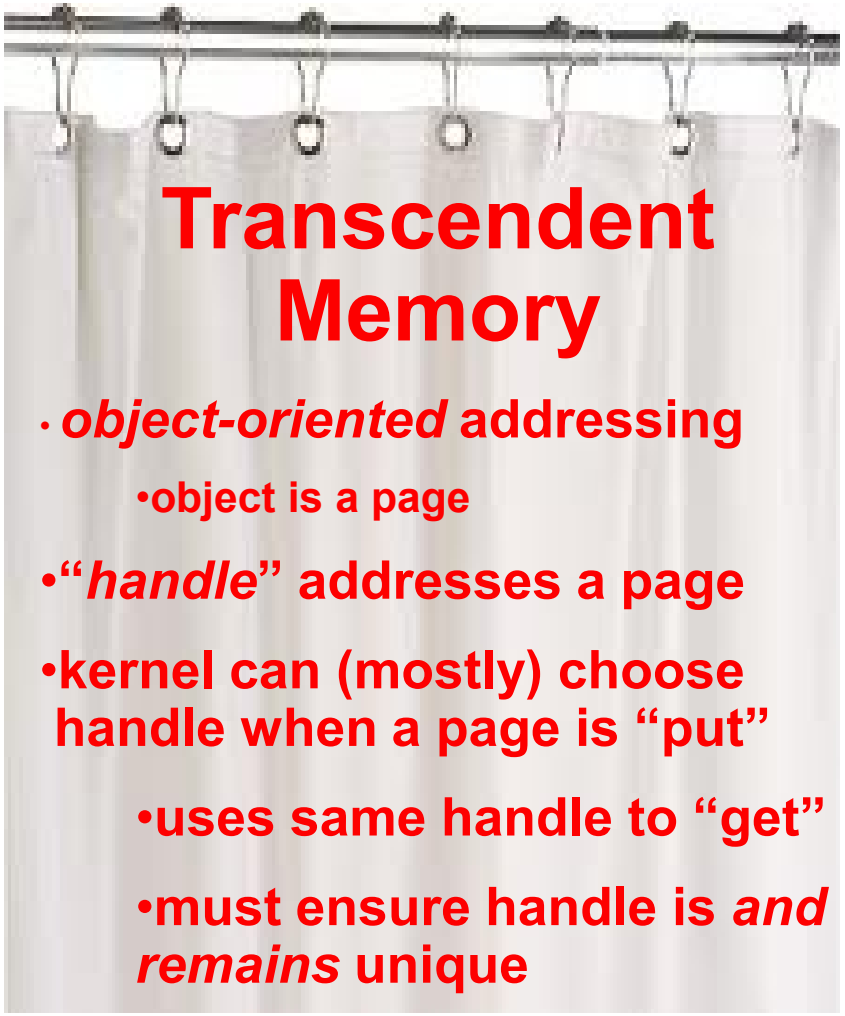
Core Transcendent Memory Operations

“Normal” RAM addressing

- byte-addressable
- virtual address: `@fffff80001024580`

"Normal" RAM addressing

- byte-addressable
- virtual address: `@ffffff80001024580`

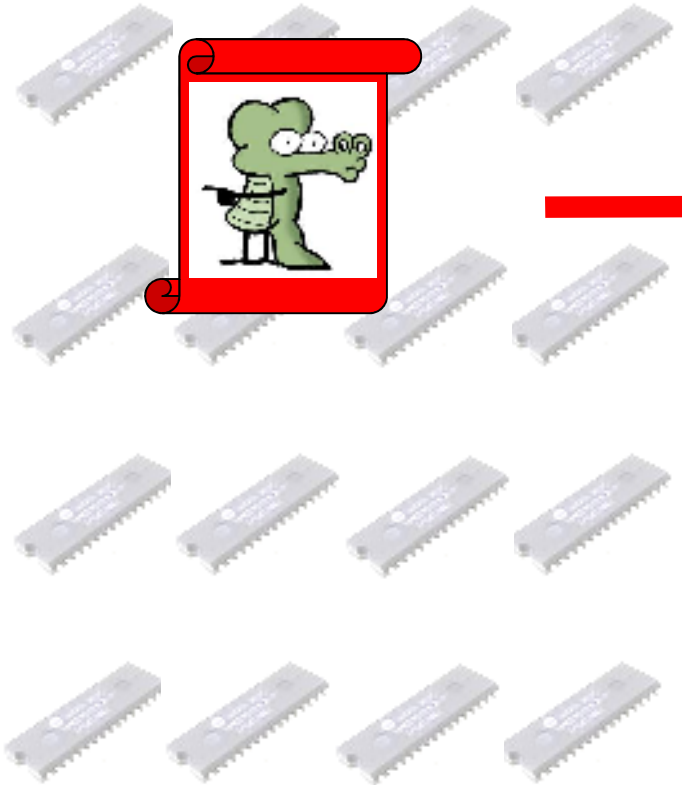


Transcendent Memory

- *object-oriented* addressing
  - object is a page
- "*handle*" addresses a page
- kernel can (mostly) choose handle when a page is "put"
  - uses same handle to "get"
  - must ensure handle is *and remains* unique

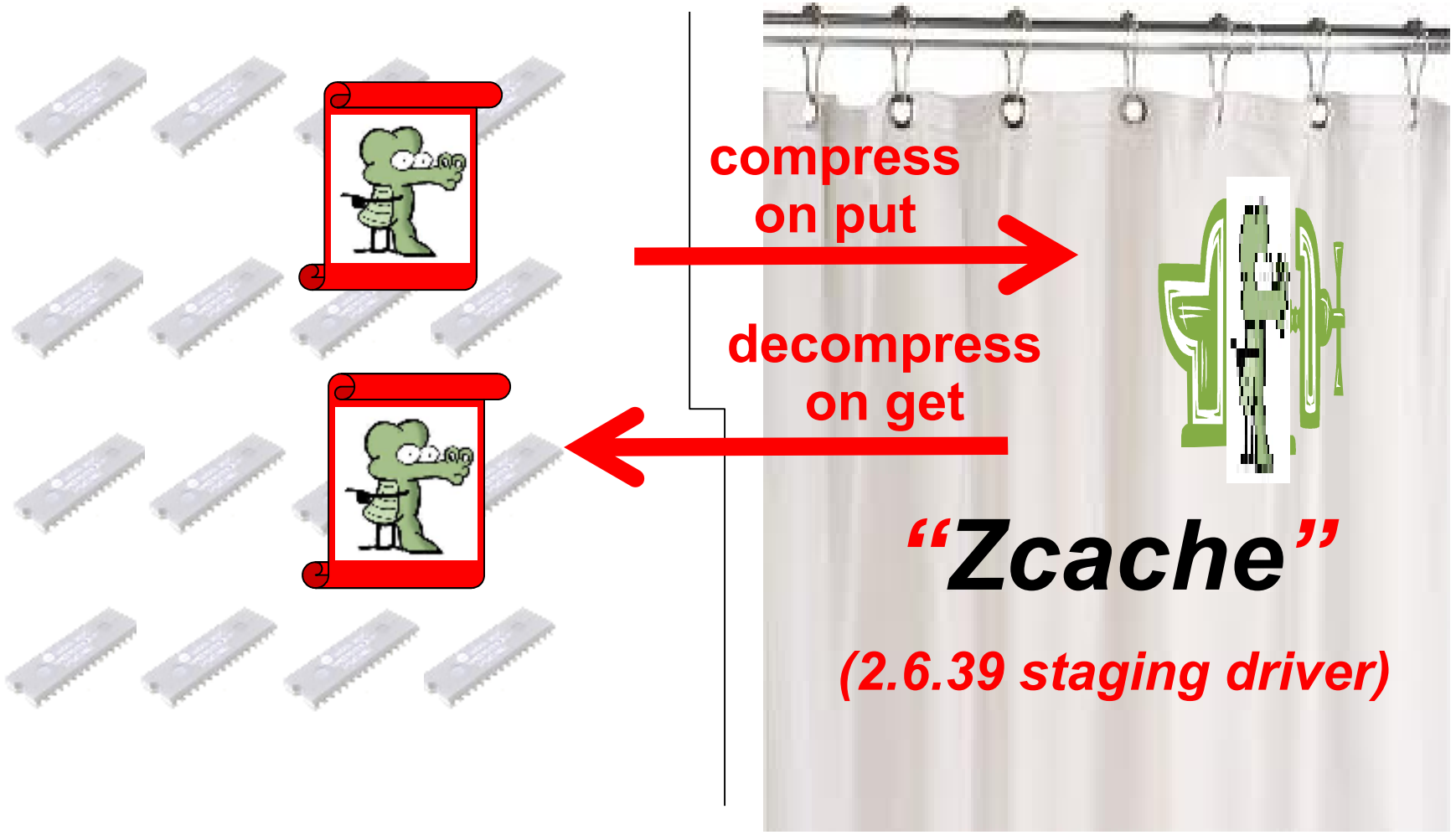
Why bother??

Why bother?? Because…

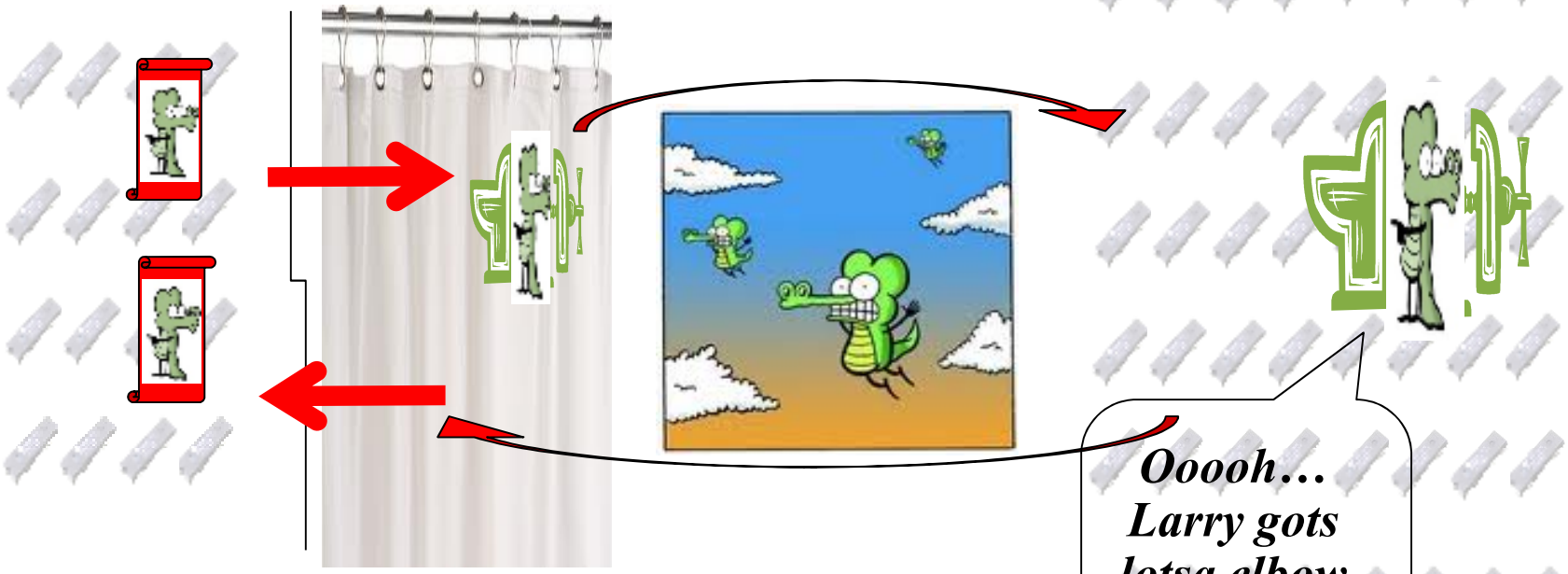


Because once behind the curtain, we can do* interesting *things…

Interesting thing #1: Zcache

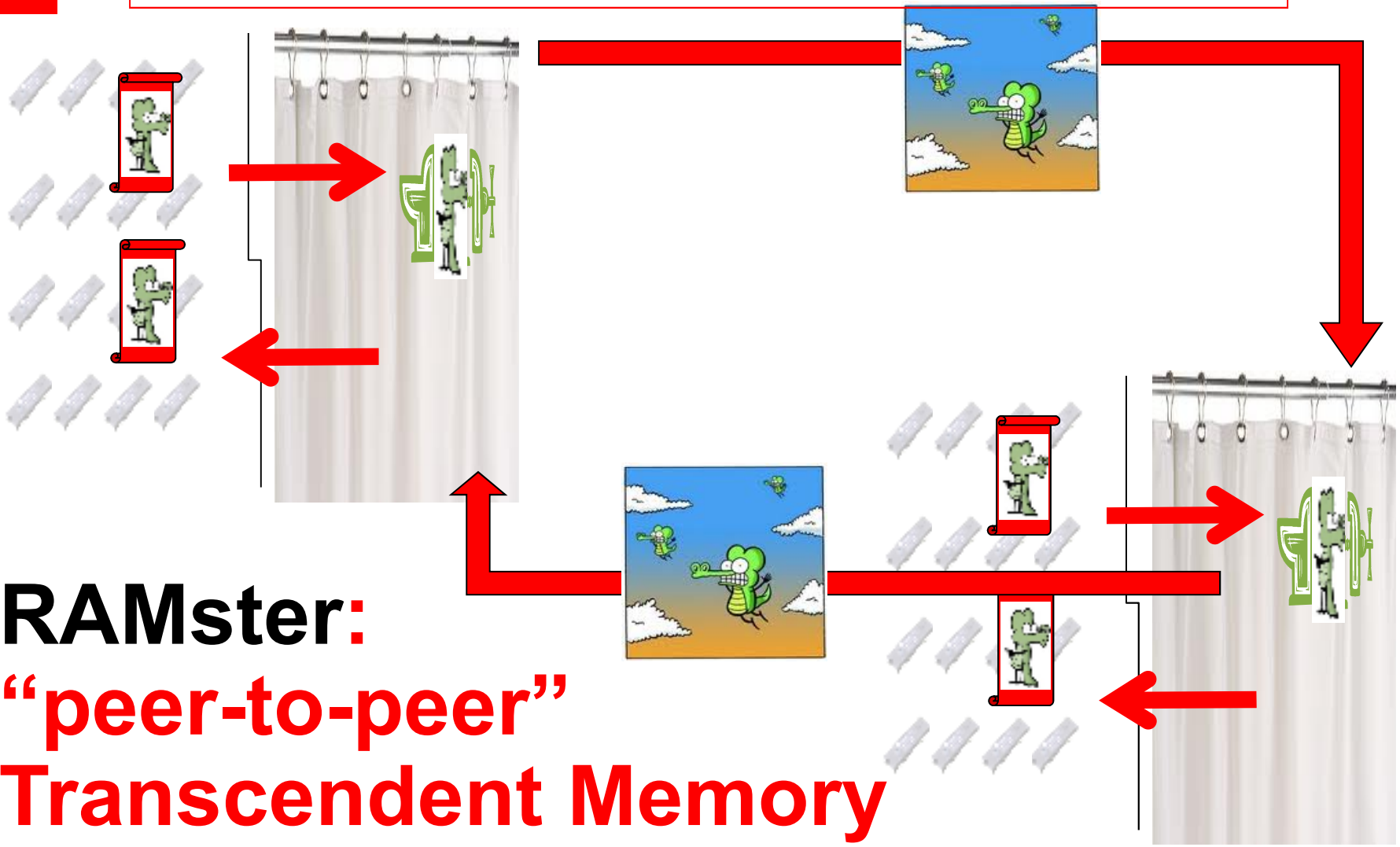


Interesting thing #2:



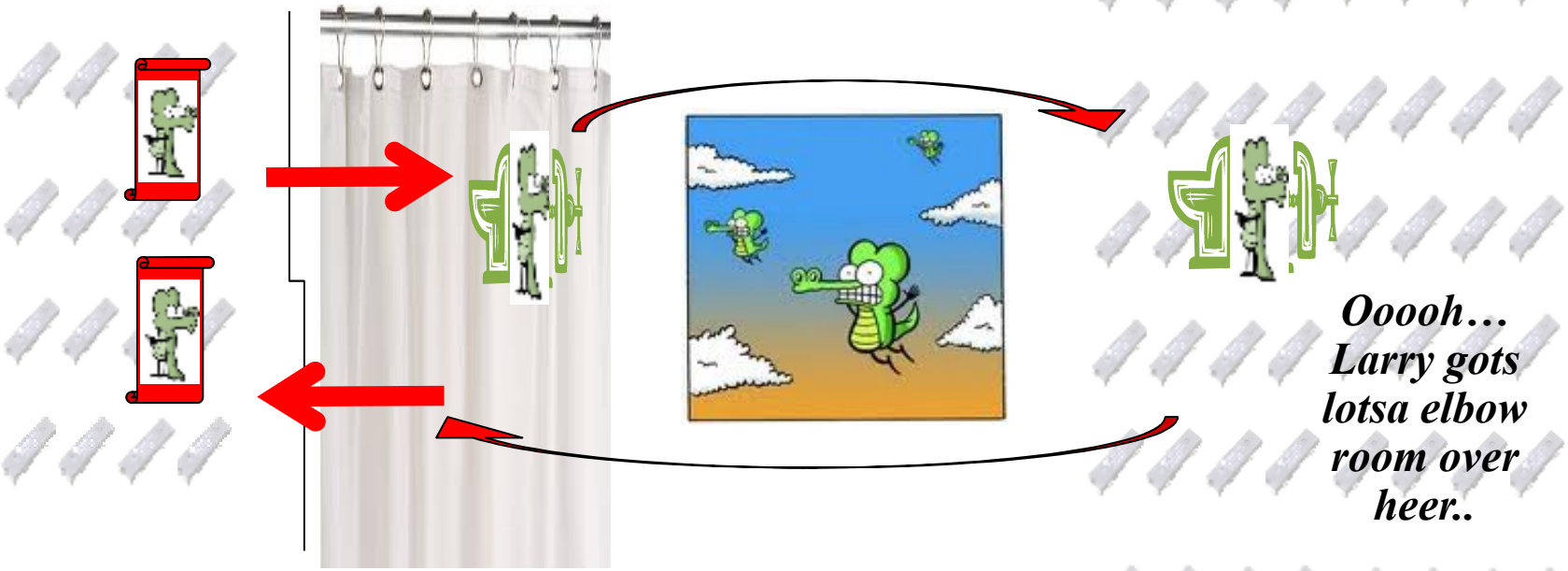
Transparently move pre-compressed pages across a high(??)-speed coherent(?) interconnect

Interesting thing #2A:



RAMster:
"peer-to-peer"
Transcendent Memory

Interesting thing #2B:



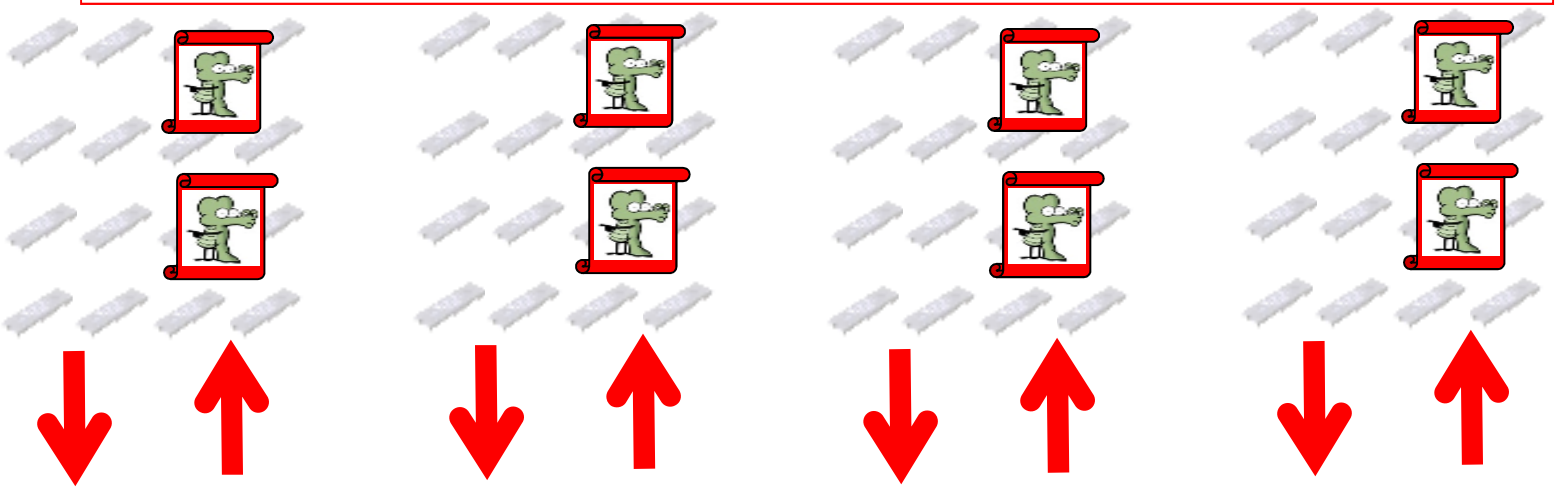
Ooooh… Larry gots lotsa elbow room over heer..

…maybe only one large "memory server" shared by many machines?

Interesting thing #3:

SSmem: Transcendent Memory as a "safe" access layer for SSD or NVRAM e.g. as a "RAM extension" *not* I/O device

Interesting thing #4:



virtual machines (aka "guests")

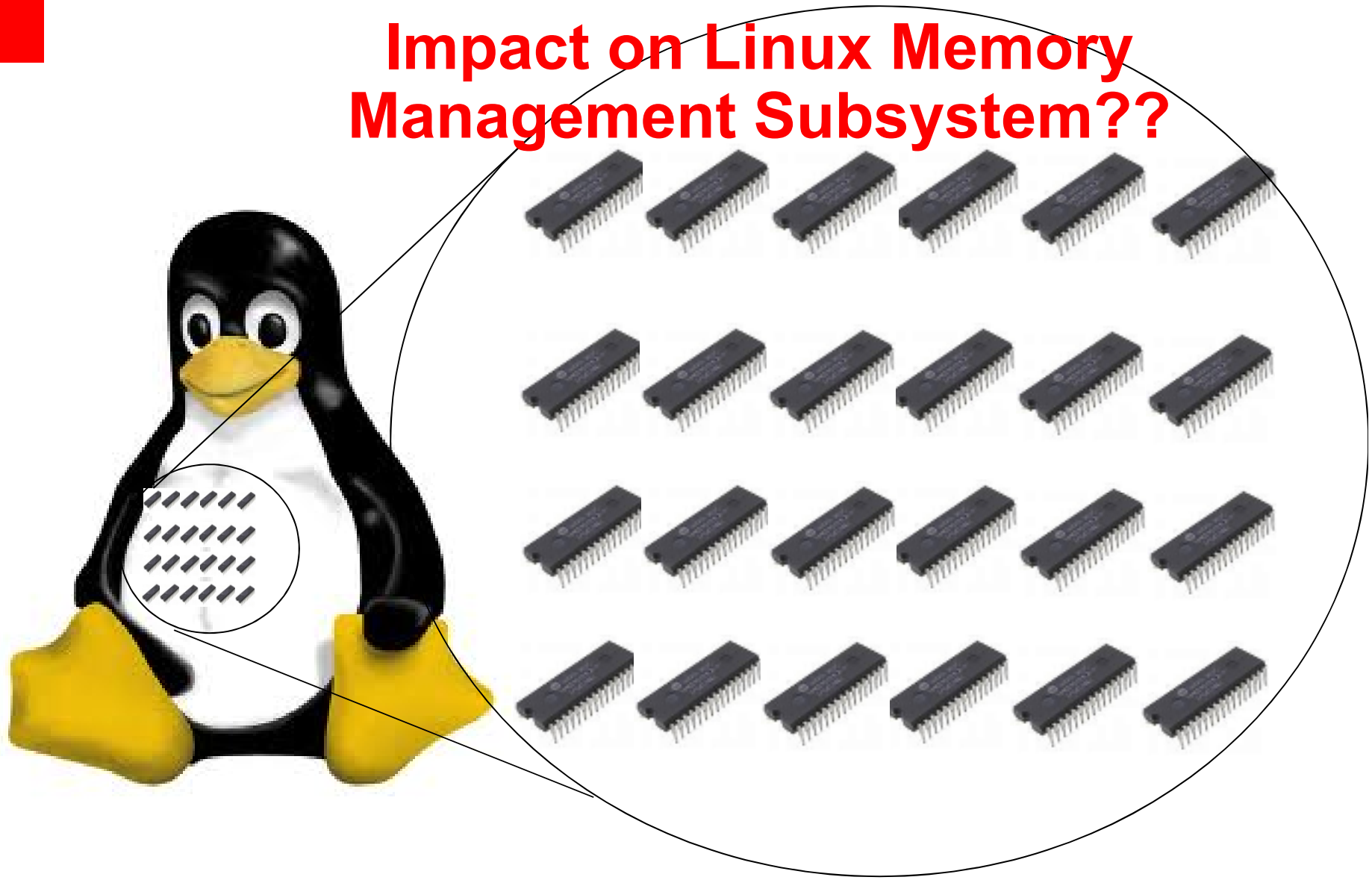
hypervisor (aka "host")

hypervisor RAM

Tmem supported in Xen since 4.0 (2009)

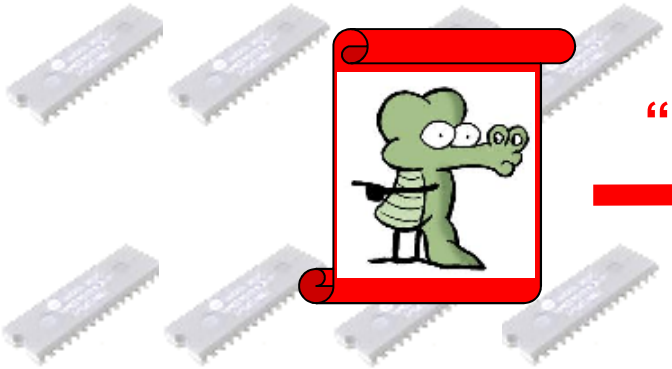
KVM *future?*

Impact on Linux Memory Management Subsystem??

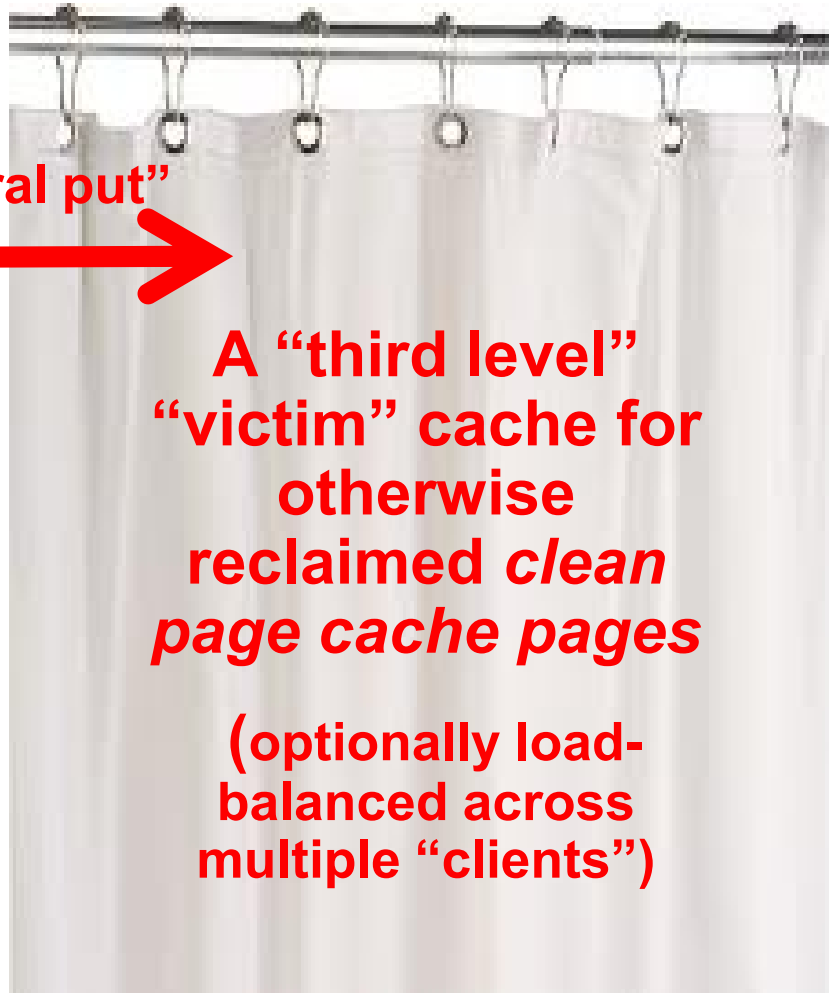


Memory pressure?
So what's a kernel to do?

Cleancache (merged for 3.0)



"ephemeral put"



Cleancache patchset:

- vfs hooks to put clean page cache pages, get them back, and maintain coherency
- per-filesystem opt-in hooks
- shim to zcache in 2.6.39
- shim to Xen tmem in 3.0

A "third level" "victim" cache for otherwise reclaimed *clean page cache pages*

(optionally load-balanced across multiple "clients")

Frontswap (target merge 3.2)

"persistent put"

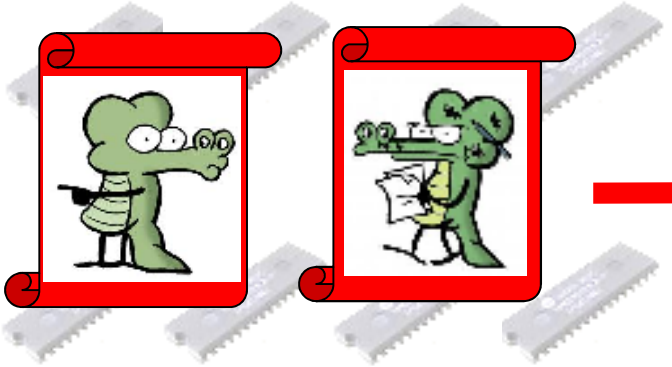
Temporary emergency FAST *swap page* store

(optionally load-balanced across multiple "clients")

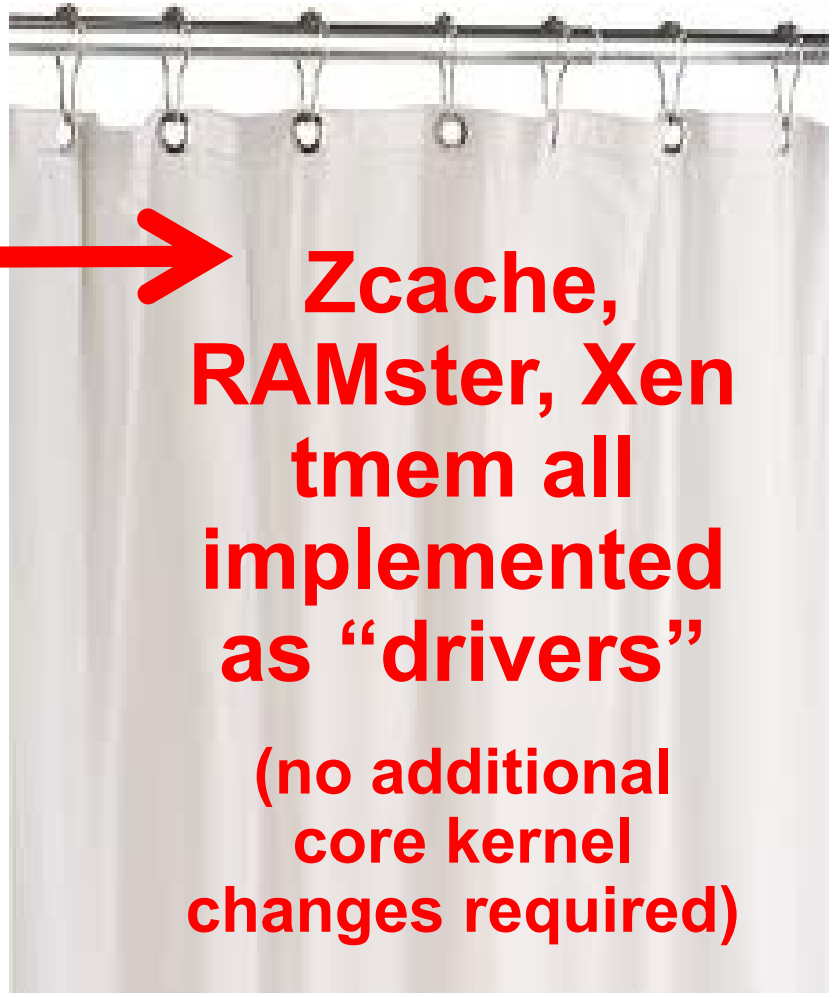
Frontswap patchset:

- **swap subsystem hooks to "*put*" swap cache pages and "*get*" them back, and maintain coherency**
- **manages tracking data structures (1 bit/page)**
- **"partial swapoff"**
- **shim to zcache in 2.6.39**
- **shim to Xen tmem merged in 3.1**

Cleancache & Frontswap

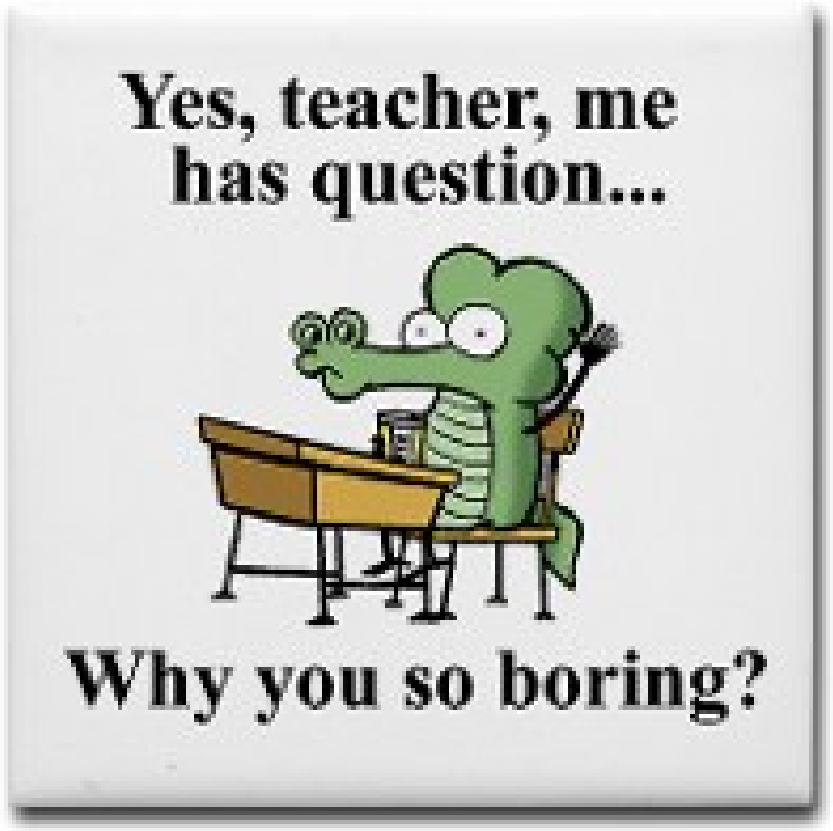


Cleancache and frontswap patches are the only core changes necessary to support ALL of Transcendent Memory's "friends"!!



Zcache, RAMster, Xen tmem all implemented as "drivers"

(no additional core kernel changes required)



Questions?

BACKUP SLIDES

Transcendent Memory in Linux
(status Aug 2, 2011)

Xen	non-Xen	name of patchset	Linux version	
N	Y	zcache	2.6.39	staging driver
Y	Y	cleancache	3.0	Linus decided!
Y	N	selfballooning	3.1	
Y	?	frontswap-selfshrinking	3.1	
Y	Y	frontswap	3.2?	linux-next
?	Y	RAMster	?	In development