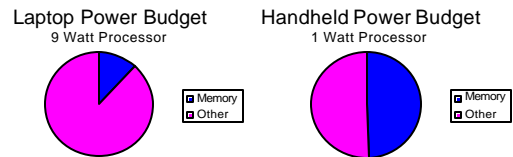


Outline for Today's Lecture

- Administrative:
Extension on Nachos assignment 4 until Thursday Nov 1
- Objective for today:
Advanced topics in memory management
 - Power-aware memories
 - Distributed shared memory (DSM)
- Next time:
 - File systems

Motivation: What can the OS do about the energy consumption of memory?



Energy Efficiency Metrics

- Power consumption in *watts* (mW).
- Battery lifetime in *hours* (seconds, months).
- Energy consumption in *Joules* (mJ).
- Energy * Delay
- Watt per megabyte

Physics Basics

- Voltage is amount of energy transferred from a power source (e.g., battery) by the charge flowing through the circuit.
- Power is the *rate* of energy transfer
- Current is the *rate* of flow of charge in circuit

Relationships

Energy (Joules) = Power (watts) * Time (sec)

$$E = P * t$$

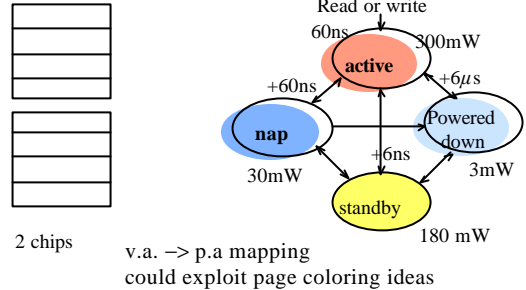
Power (watts) = Voltage (volts) * Current (amps)

$$P = V * I$$

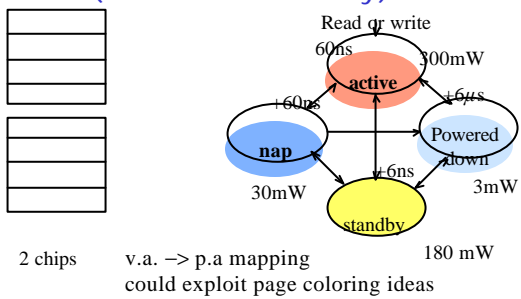
Current (amps) = Voltage (volts) / Resistance (ohms)

$$I = V / R$$

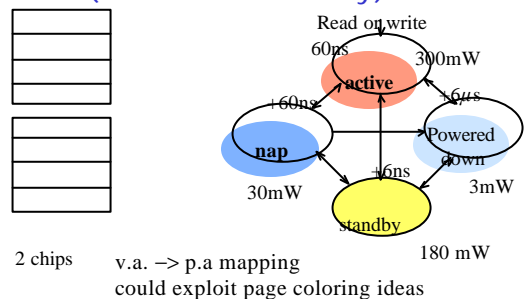
Power Management (RAMBUS memory)



Power Management (RAMBUS memory)

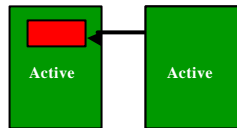


Power Management (RAMBUS memory)



RDRAM as a Memory Hierarchy

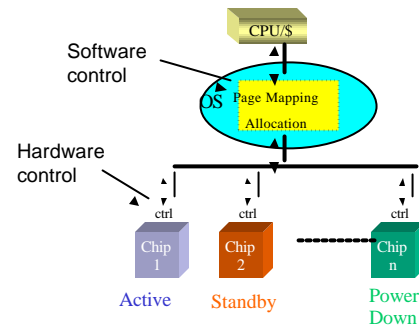
- Each chip can be independently put into appropriate power mode
- Number of chips at each "level" of the hierarchy can vary dynamically.



Policy choices

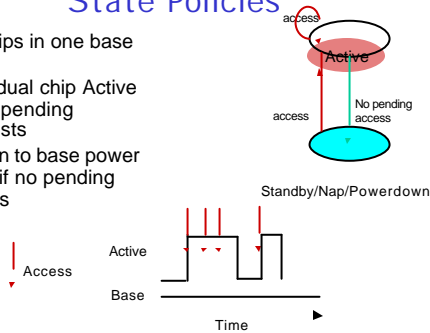
- initial page placement in an "appropriate" chip
- dynamic movement of page from one chip to another
- transitioning of power state of chip containing page

Two Dimensions to Control Energy



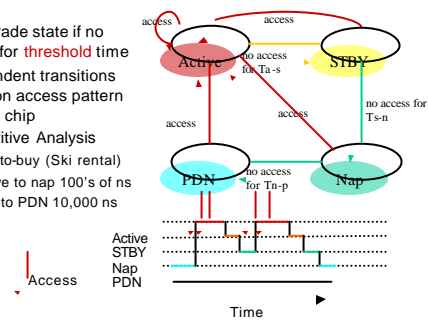
Dual-state (Static) HW Power State Policies

- All chips in one base state
- Individual chip Active while pending requests
- Return to base power state if no pending access



Quad-state (Dynamic) HW Policies

- Downgrade state if no access for **threshold time**
- Independent transitions based on access pattern to each chip
- Competitive Analysis
 - rent-to-buy (Ski rental)
 - Active to nap 100's of ns
 - Nap to PDN 10,000 ns



Ski Rental Analogy

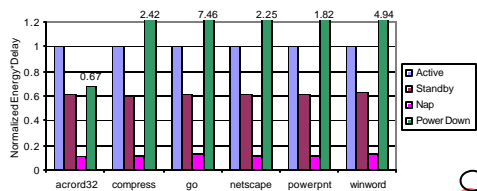


- Week 1: "Am I gonna like skiing? I better rent skis."
- Week 10: "I'm skiing black diamonds and loving it. I could've bought skis for all the rental fees I've paid. I should have bought them in the first place. Guess I'll buy now."
- If he buys exactly when $\text{rentalsPaid} == \text{skiCost}$ then pays 2*optimal

Page Allocation Policies

- Random Allocation
 - Pages spread across chips
- Sequential First-Touch Allocation
 - Consolidate pages into minimal number of chips
 - One shot
- Frequency-based Allocation
 - First-touch not always best
 - Allow movement after first-touch

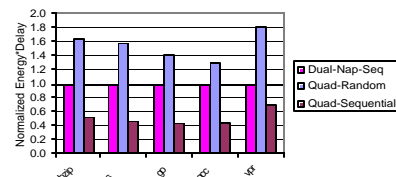
Dual-state + Random Allocation*



- Active to perform access, return to base state
- Nap is best ~85% reduction in E*D over full power
- Little change in run-time, most gains in energy/power

* NT Traces

Quad-state HW + Sequential Allocation*



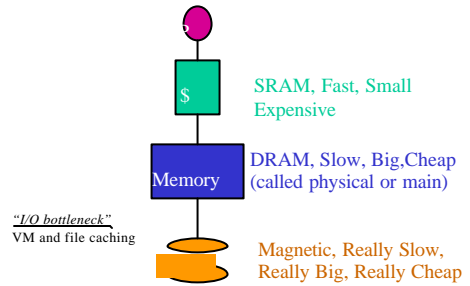
- Base: Dual-state Nap Sequential Allocation
- Thresholds: 0ns A->S; 750ns S->N; 375,000 N->P
Subsequent analysis shows 0ns threshold S->N is better.
- Quad-state + Sequential 30% to 55% additional improvement over dual-state nap sequential
- Sophisticated HW not enough

* SPEC

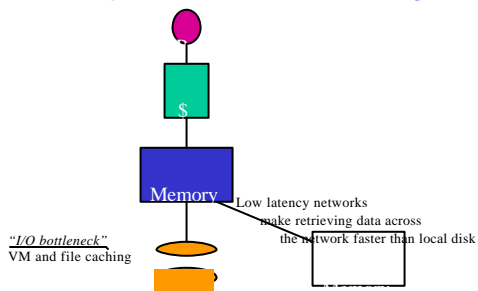
The Design Space

	Random Allocation	Sequential Allocation
Dual-state Hardware (static)	Nap is best dual-state policy 60%-85%	Additional 10% to 30% over Nap
Quad-state Hardware (dynamic)	No compelling benefits over simpler policies	Best Approach: 6% to 55% over dual-nap-seq, 99% to 80% over all active.

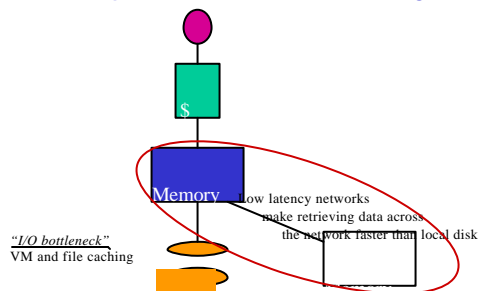
Motivation: Traditional Memory Hierarchy

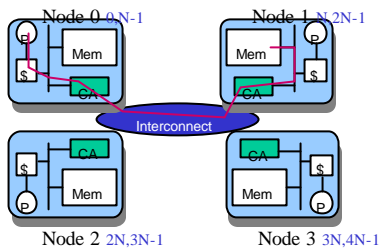


Motivation: How to exploit remote memory?



Motivation: How to exploit remote memory?





- Each node owns some of physical memory
- OS can allocate physical memory anywhere in system
- May or may not be hardware support for shared memory across nodes (does the MMU recognize remote addresses?)
 - NUMA (non-uniform memory access) vs. distributed memory architectures.

NUMA

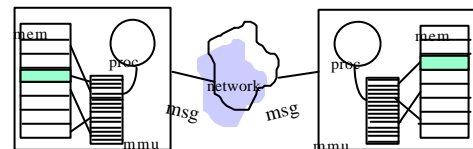
- Unlike distributed memory architectures, remote memory access is possible (HW support).
- Remote access cost $n * local\ access\ time$.
- For each local fault there is a choice:
 - install mapping to remotely resident data
 - migrate data and install mapping to local data
 - replicate data and install mapping to local copy of data
 - if non-resident, determine initial placement
- Valid bit -> non-local,
- Write-protect -> consistency issues to be addressed

Distributed Memory

- Remote memory access is not possible.
- Remote access via message-passing.
- For each local fault there is a choice:
 - install mapping to remotely resident data
 - migrate data and install mapping to local data
 - replicate data and install mapping to local copy of data
 - if non-resident, determine initial placement
- Valid bit -> non-local,
- Write-protect -> consistency issues to be addressed

Distributed Shared Memory (DSM)

Allows use of a shared memory programming model (shared address space) in a distributed system (processors with only local memory)



DSM Issues

- Can use the local memory management hardware to generate fault when desired page is not locally present or when write attempted on read-only copy.
- Locate the page remotely - current "owner" of page (last writer) or "home" for page.
- Page sent in message to requesting node (read access makes copy; write migrates)
- Consistency protocol - invalidations or broadcast of changes (update)
 - directory kept of "caches" holding copies

DSM States

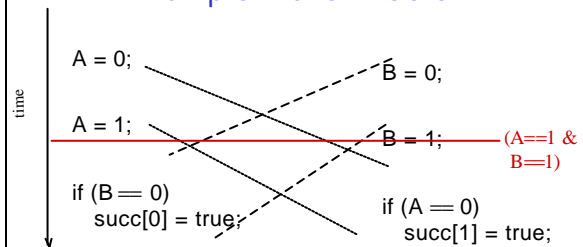
Forced faults are key to consistency operations

- Invalid local mapping, attempted read access
 - data flushed from most recent writer,
 - set write-protect bit for all copies.
- Invalid local mapping, attempted write access
 - migrate data, invalidate all other copies.
- Local read-only copy, write-fault -
 - invalidate all other copies

Consistency Models

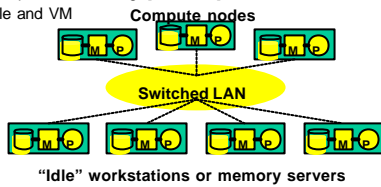
- Sequential consistency
 - All memory operations *appear* to execute one at a time. A write is considered *done* only when invalidations or updates have propagated to all copies.
- Weaker forms of consistency
 - Guarantees associated with synchronization primitives; at other times, it doesn't matter
 - For example:
 - acquire lock - make sure others' writes are done
 - release lock - make sure all my writes are seen by others

Example - the Problem



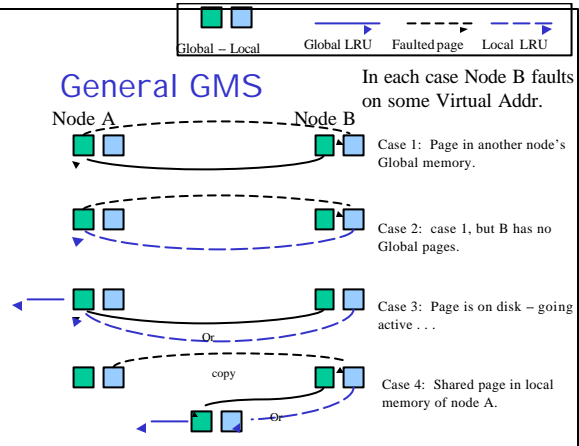
The Global Memory Service (GMS)

- GMS kernel module for Digital Unix and FreeBSD [Feeley95, Voelker98]
 - remote paging [Comer90, Felten91, Li/Petersen93]
 - cooperative caching [Dahlin93]
 - file and VM



General GMS

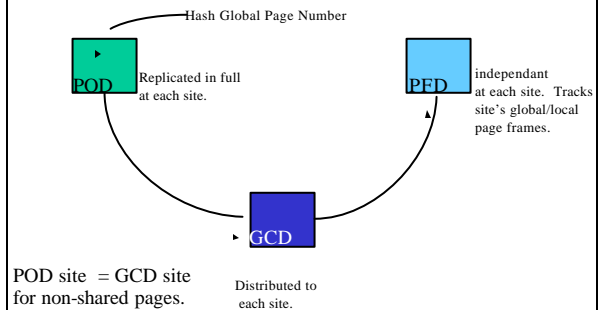
In each case Node B faults on some Virtual Addr.



GMS: The Big Problems

- How to uniquely name and identify pages/blocks?
 - 128-bit Unique Identifiers (UIDs [Leach85]). Global Page Numbers (GPN).
- How to maintain consistent information about the location of pages in the network?
 - Each node keeps a **Page Frame Directory** of pages cached locally.
 - A distributed **Global Cache Directory** identifies the caching site(s) for each page or block.
 - Hash on page ID's to get GCD site.
- How to keep a consistent view of the membership in the cluster and the status of participating nodes?
- How to globally coordinate memory usage for the page cache?
 - "Thing globally, act locally." - choice of replacement algorithm

Finding a Page in GMS



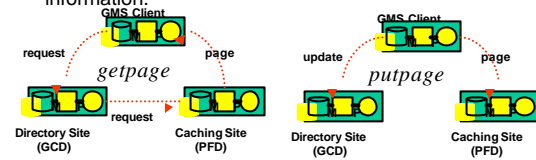
Global Memory Management

- **Fantasy**: on a page fetch, choose the globally “coldest” page in the cluster to replace (i.e., global LRU).
 - may demote a local page to global
- **Reality**: approximate global LRU by making replacement choices across epochs of (at most) **M** evictions in **T** seconds.
 - trade off accuracy for performance
 - nodes send summary of page values to initiator at start of epoch
 - initiator determines which nodes will yield the epoch’s evictions
 - favor local pages over global pages
 - victims are chosen probabilistically

Global Memory System (GMS) in a Nutshell

GCD: Global Cache Directory, PFD: Page Frame Directory, P: Processor, M: Memory

- Basic operations: *putpage*, *getpage*, *movepage*, *discardpage*.
- **Directory site** is self for private pages, hashed for sharable.
- **Caching site** selected locally based on global information.

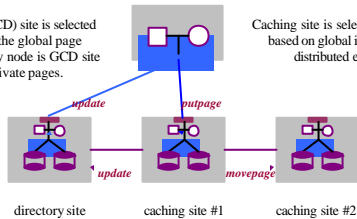


GMS Putpage/Movepage

Page/block evictions trigger an asynchronous *putpage* request from the client; the page is sent as a *payload* piggybacked on the message.

Directory (GCD) site is selected by a hash on the global page number; every node is GCD site for its own private pages.

Caching site is selected locally based on global information distributed each epoch.

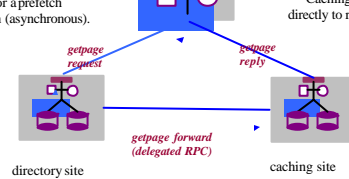


GMS Page Fetch (*getpage*)

Page fetches are requested with a *getpage* RPC call to page directory site.

Getpage can result from a demand fault or a prefetch policy decision (asynchronous).

Caching site sends page directly to requesting node.



Directory site delegates request to the correct caching site.