

Power Aware Page Allocation

Alvin R. Lebeck



Milly Watt

Department of Computer Science
Duke University
alvy@cs.duke.edu
<http://www.cs.duke.edu/~alvy>

The Milly Watt Team

 **Faculty:** Alvin R. Lebeck, Carla S. Ellis, Amin Vahdat

 **Students:** Xiaobo Fan, Heng Zeng, Todd Cignetti, Kirill Komarov, Prachi Thakar, Jaidev Patwardhan

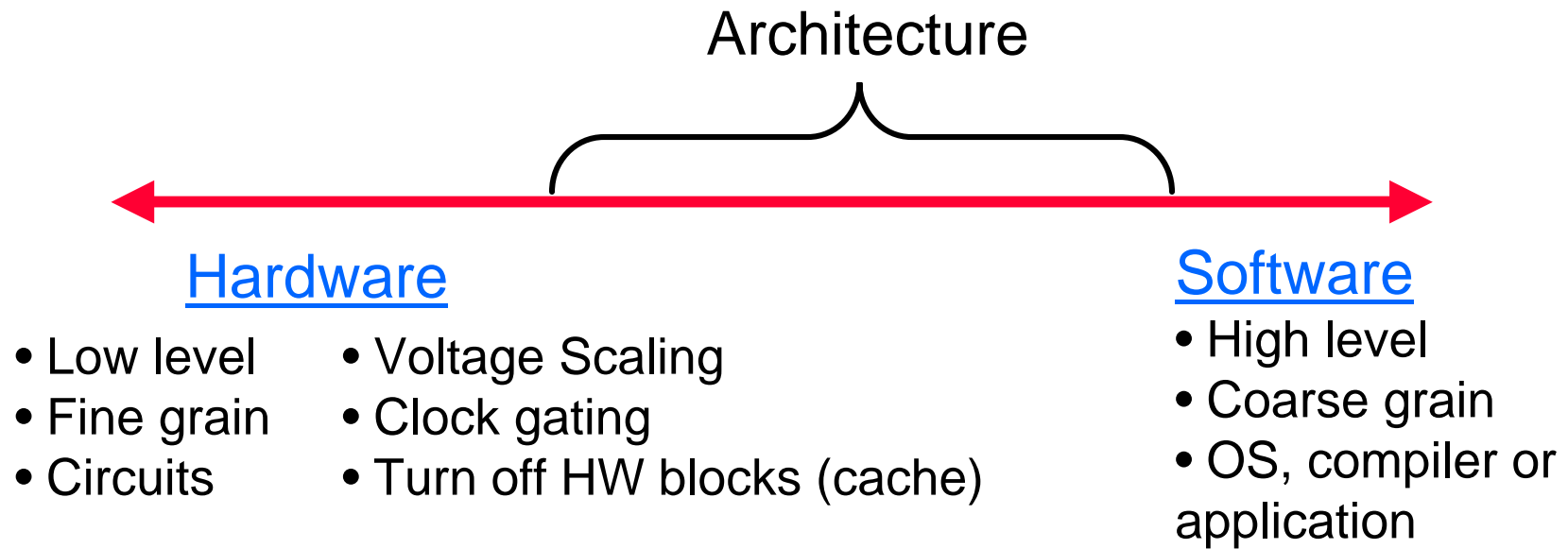
 **Staff:** David Becker

 **Funding:** NSF ITR, NSF Research Instrumentation, Intel, Microsoft

The Milly Watt Project

- ✍ Traditional system design targets increased performance
- ✍ Post-PC world has many battery powered devices
- ✍ Energy optimization becomes increasingly important
 - Functionality of devices constrained by available power
- ✍ Today, little understanding of energy ramifications of application/system design
- ✍ **Goal: Power Aware System Design**
 - Need: Revisit all aspects of system design [SIGOPS EW '00]
 - Focus on Architecture, Operating System, Networking, Applications

The Energy Saving Spectrum



 **Architecture: Re-examine interactions between HW and SW, particularly the Operating System**

The Unturned Stone

Traditional OS resources

- Processor, Memory, Disk, Network

Power Studies

 Disk spindown policies [Baker, Douglass, Li]

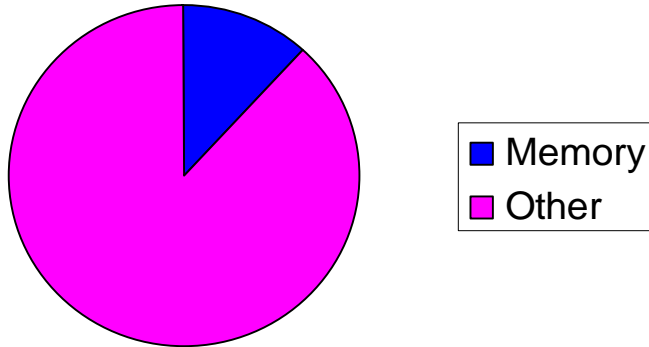
 Processor voltage and clock scaling [Weiser, Pering, Lorch]

 Network Interface [Stemm, Kravets, Imielinski]

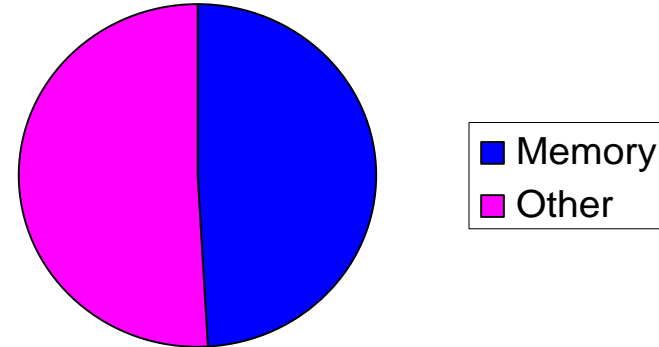
 **Where is main memory?**

Memory System Power Consumption

Laptop Power Budget
9 Watt Processor



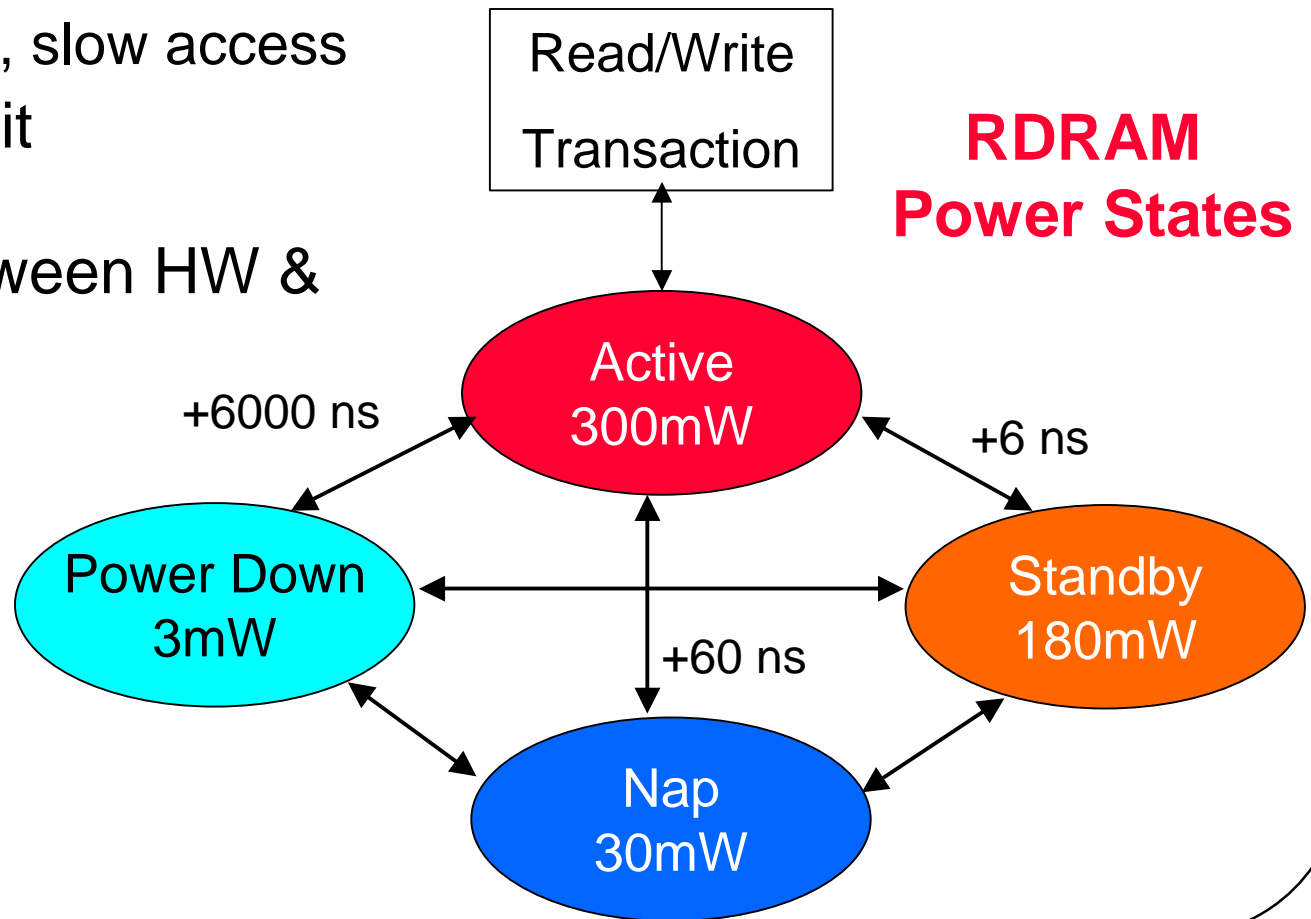
Handheld Power Budget
1 Watt Processor



- ✍ Laptop: memory is small percentage of total power budget
- ✍ Handheld: low power processor, memory is more important
- ✍ Memory benchmark variance larger for Pocket Computer
 - 9% Thinkpad vs. 100% Itsy [Farkas '00]

Opportunity: Power Aware DRAM

- ✍ Multiple power states
 - Fast access, high power
 - Low power, slow access
- ✍ How to exploit opportunity?
- ✍ Interplay between HW & SW



Outline

✍ Motivation

✍ The opportunity

✍ Hardware power management policies

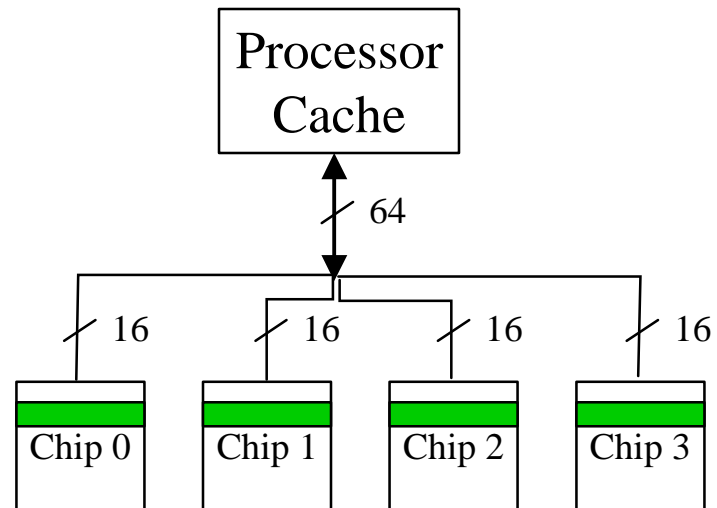
✍ Operating system page allocation policies

✍ Results

- Need both HW and OS support to maximize benefits
- Simple HW does very well
- Sophisticated HW w/ OS 6% to 55% improvement over simple hardware

Conventional Main Memory Design

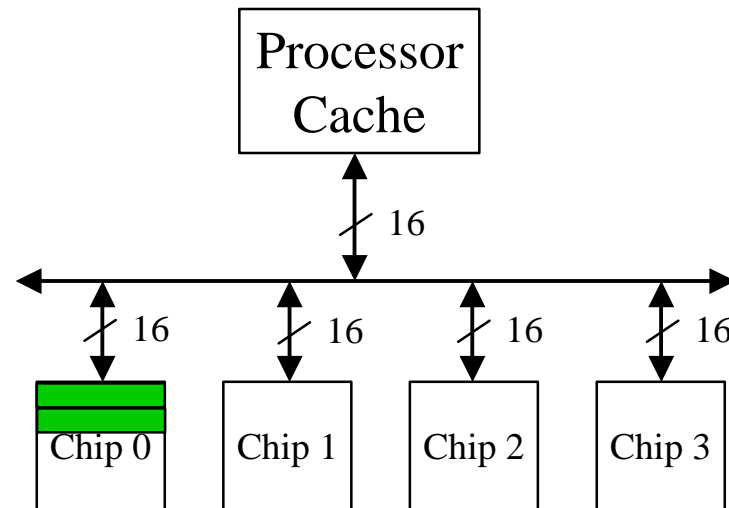
Part of Cache Block



- ✍ Multiple DRAM chips provide high bandwidth per access
 - Wide bus to processor
 - Few internal banks
- ✍ Energy implication: **Must activate all those chips to perform access at high bandwidth**

RAMBUS Main Memory Design

Cache Block



- ✍ Single RDRAM chip provides high bandwidth per access
 - Novel signaling scheme transfers multiple bits on one wire
 - Many internal banks: many requests to one chip
- ✍ Energy implication: **Must activate only one chip to perform access at same high bandwidth as conventional design**
 - **Cluster accesses to already powered up chip**

Exploiting the opportunity

✍ Main memory is becoming more important

✍ Opportunity: RAMBUS properties

- Multiple power states
- Single chip high bandwidth
- Can transition chips individually

✍ Interaction between state transitions and data locality

✍ Q1: How do we manage the power state transitions?

- Quantify benefits of power states

✍ Q2: What role does software have?

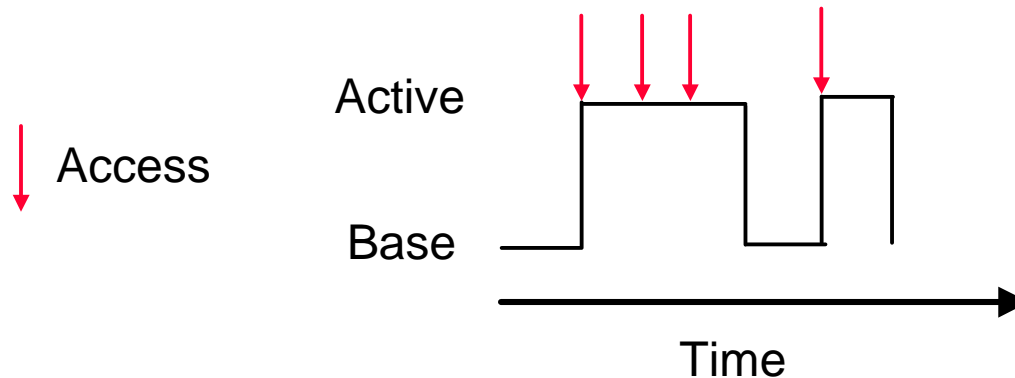
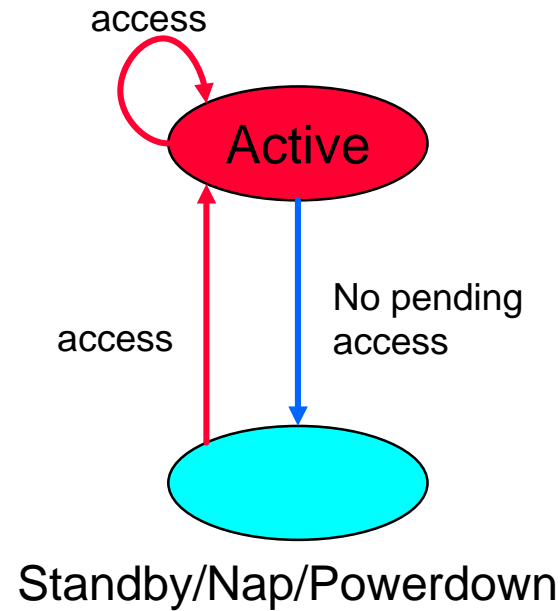
- Does allocation of data/text to memory affect energy?

Outline

- ✍ Motivation
- ✍ The opportunity
- ✍ Hardware power management policies
- ✍ Operating system page allocation policies
- ✍ Results

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



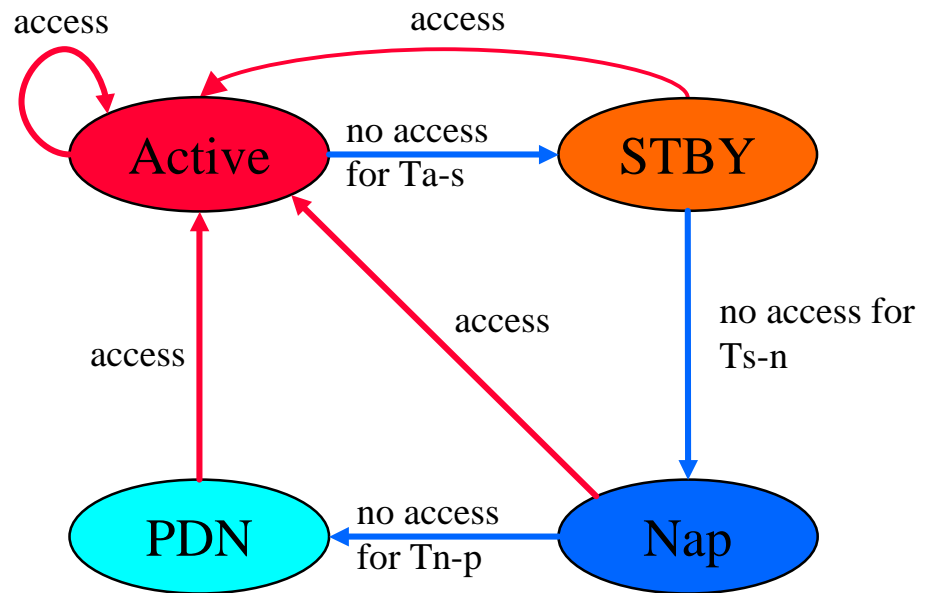
Quad-state (Dynamic) HW Policies

✍ Downgrade state if no access for **threshold** time

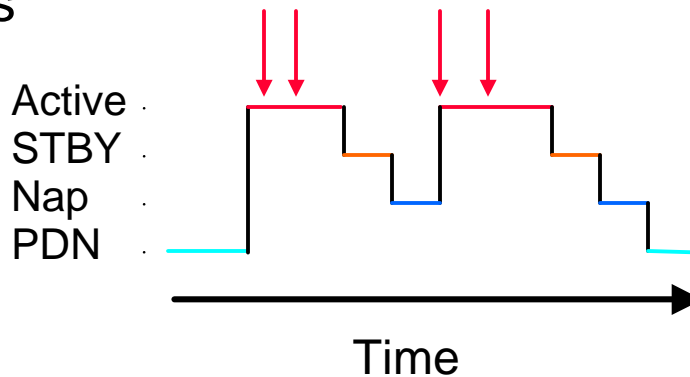
✍ Independent transitions based on access pattern to each chip

✍ Competitive Analysis

- rent-to-buy
- Active to nap 100's of ns
- Nap to PDN 10,000 ns



↓ Access



Exploiting PDRAM Power States

Hardware power management policies

- Exploit locality to reduce energy consumption
- Dual-state model
- Quad-state model (could set to dual-state)

Want to increase DRAM chip-level locality

Performance penalty with conventional DRAM

- Limited single chip bandwidth to processor
- Limited number of internal banks (simultaneous requests)

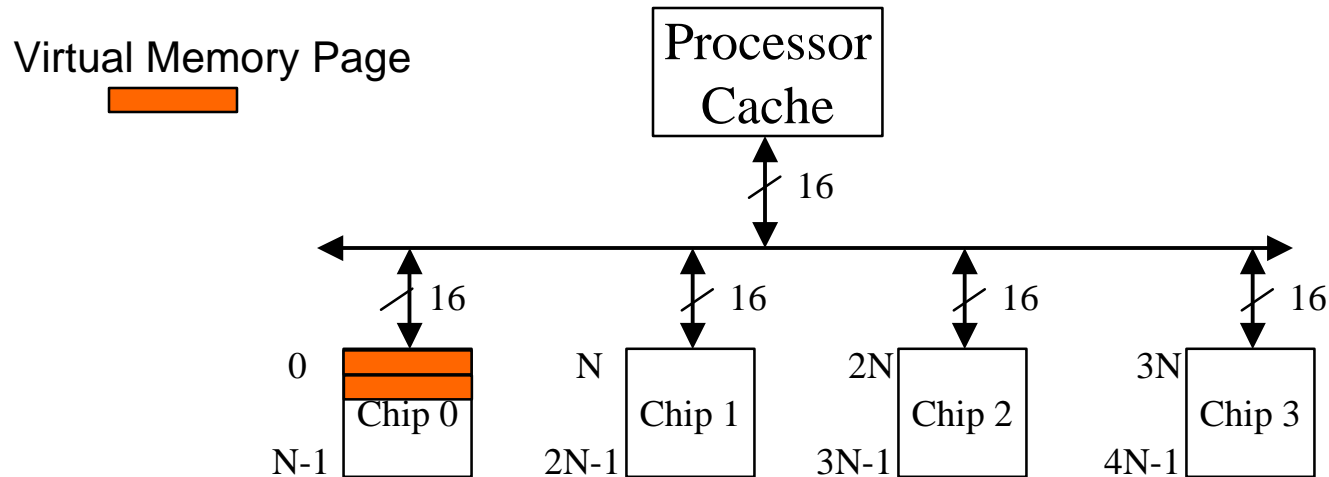
RDRAM properties

- Many internal banks (can handle many requests)
- High bandwidth connection to processor

Outline

- ✍ Motivation
- ✍ The opportunity
- ✍ Hardware power management policies
- ✍ Operating system page allocation policies
- ✍ Results

Page Allocation and PADRAM



- ✍ Physical address determines which chip is accessed
- ✍ Assume non-interleaved memory
 - Addresses 0 to N-1 to chip 0, N to 2N-1 to chip 1, etc.
- ✍ Entire virtual memory page in one chip
- ✍ **Virtual memory page allocation influences chip-level locality**

Page Allocation Policies

Random Allocation

- Pages spread across chips

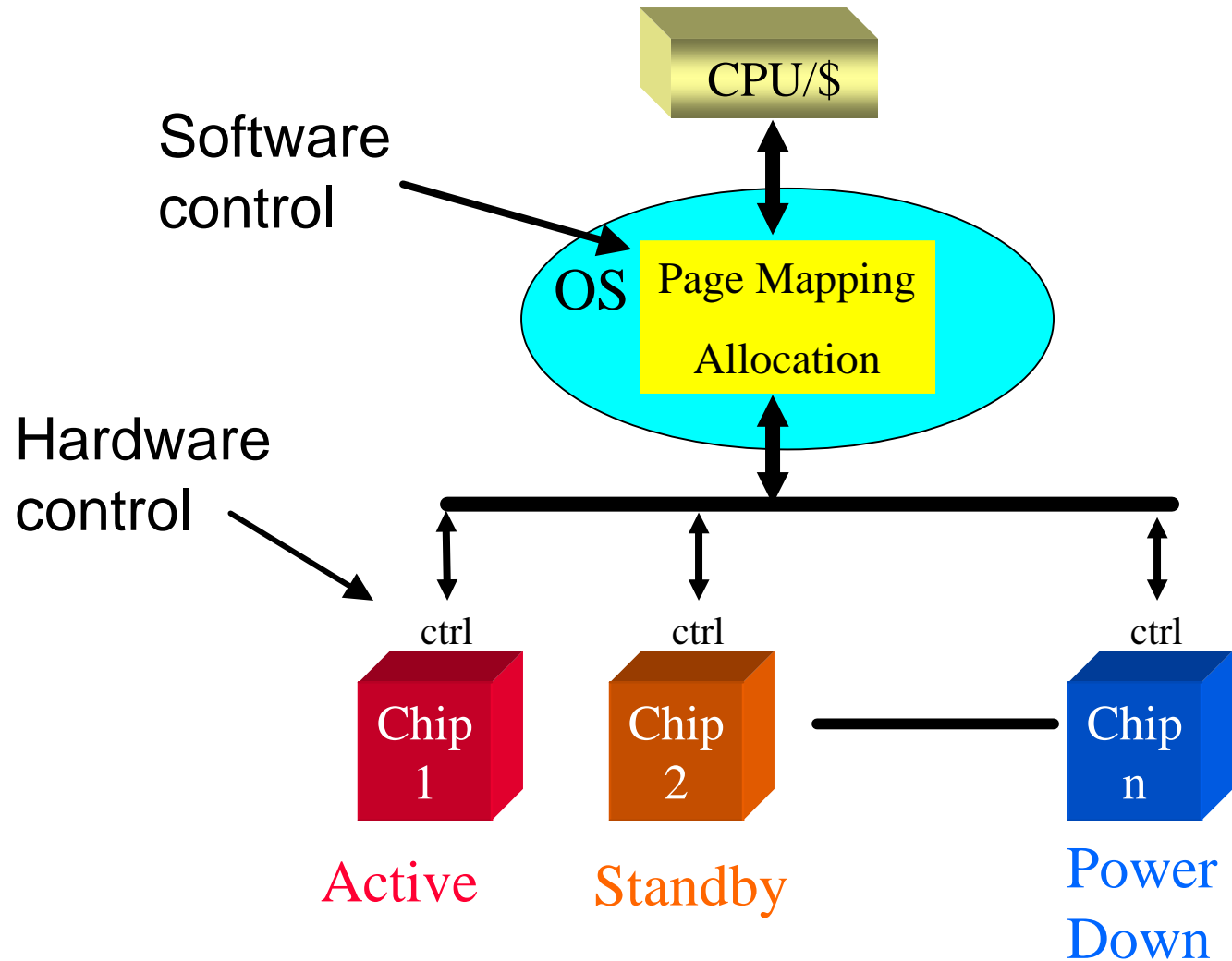
Sequential First-Touch Allocation

- Consolidate pages into minimal number of chips
- One shot

Frequency-based Allocation

- First-touch not always best
- Allow movement after first-touch
- Preliminary results

Two Dimensions to Control Energy



Outline

✍ Motivation

✍ The opportunity: Power Aware DRAM

✍ Hardware power management policies

✍ Operating system page allocation policies

✍ Results

- Methodology
- Hardware and Software Policies (alone and combined)
- Frequency-based Allocation

Methodology

✍ Metric: Energy*Delay Product

- Avoid very slow solutions

✍ Energy Consumption (DRAM only)

- Processor & Cache affect runtime
- Runtime doesn't change much in most cases

✍ 8KB page size

✍ L1/L2 non-blocking caches

- 256KB direct-mapped L2
- Qualitatively similar to 4-way associative L2

✍ Average power for transition from lower to higher state

✍ Trace-driven and Execution-driven simulators

Methodology Continued

Trace-Driven Simulation

- Windows NT personal productivity applications (Etch at Washington)
- Simplified processor and memory model
- Eight outstanding cache misses
- Eight 32Mb chips, total 32MB, non-interleaved

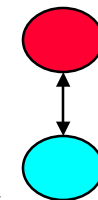
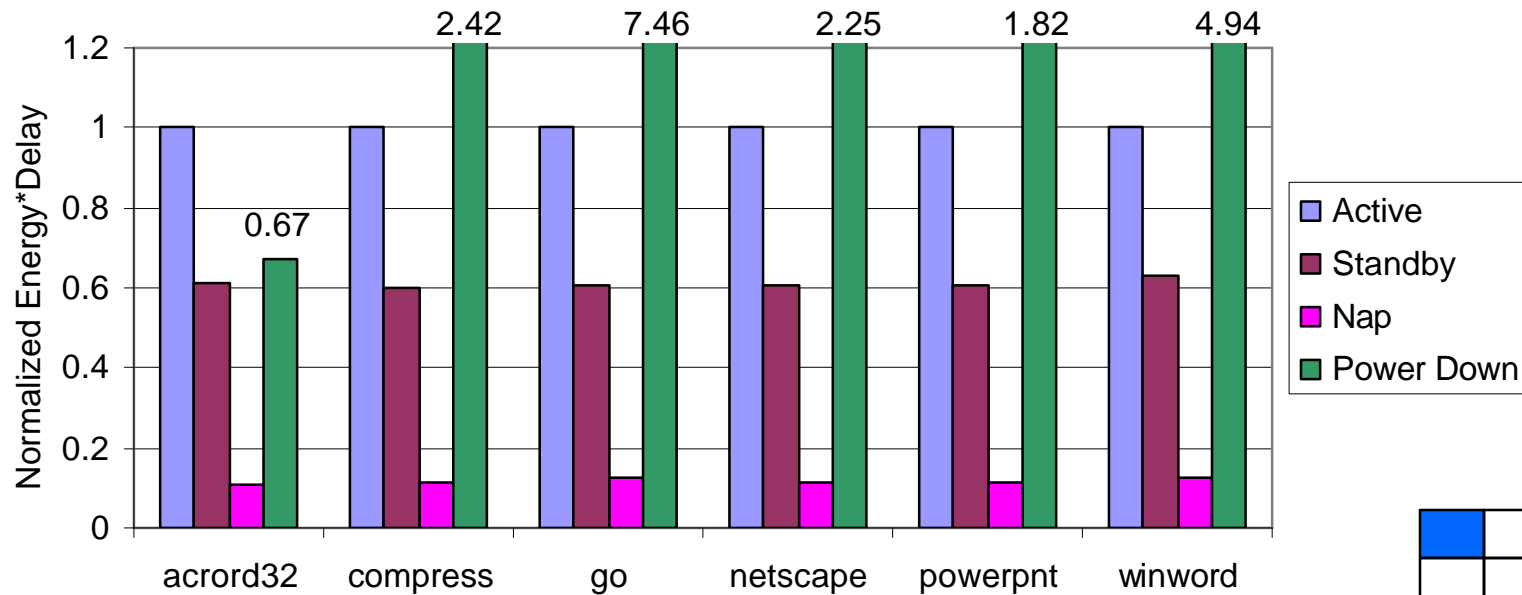
Execution-Driven Simulation

- SPEC benchmarks (subset of integer)
- SimpleScalar w/ **detailed RDRAM timing and power models**
- Sixteen outstanding cache misses
- Eight 256Mb chips, total 256MB, non-interleaved

The Design Space

	Random Allocation	Sequential Allocation
Dual-state Hardware (static)	1 Simple HW	2 Can the OS help?
Quad-state Hardware (dynamic)	3 Sophisticated HW	4 Cooperative HW & SW

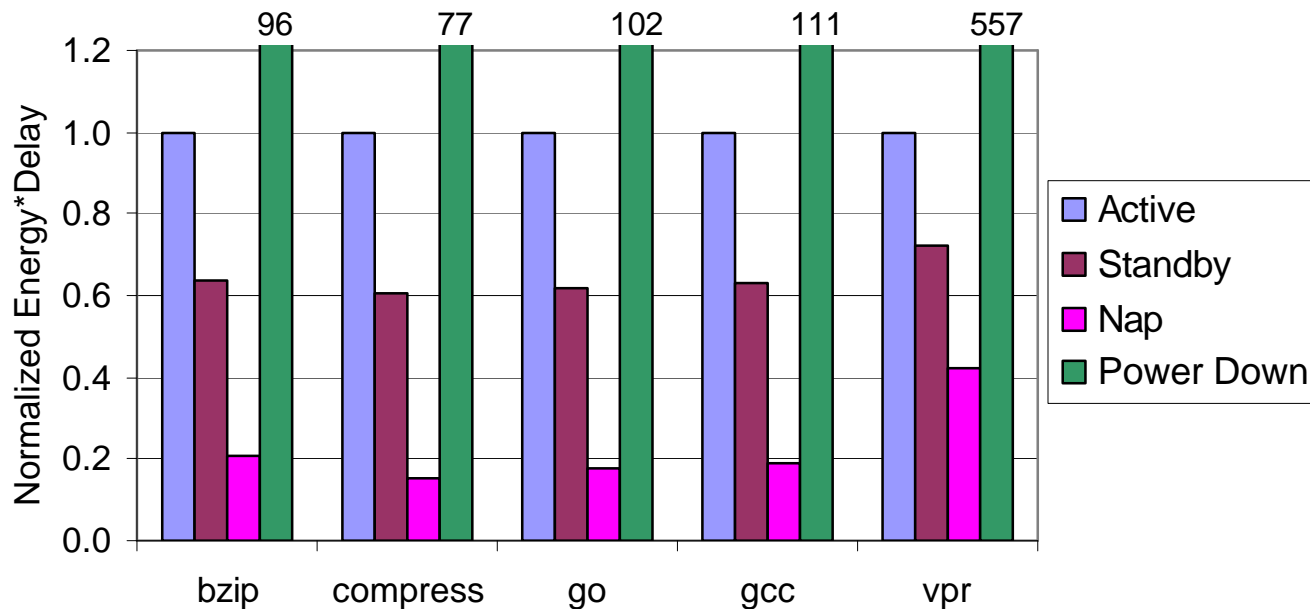
Dual-state + Random Allocation (NT Traces)



2 state model

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

Dual-state + Random Allocation (SPEC)



- ✍ All chips use same base state
- ✍ Nap is best 60% to 85% reduction in $E \cdot D$ over full power
- ✍ Simple HW provides good improvement

The Design Space

Random
Allocation

Sequential
Allocation

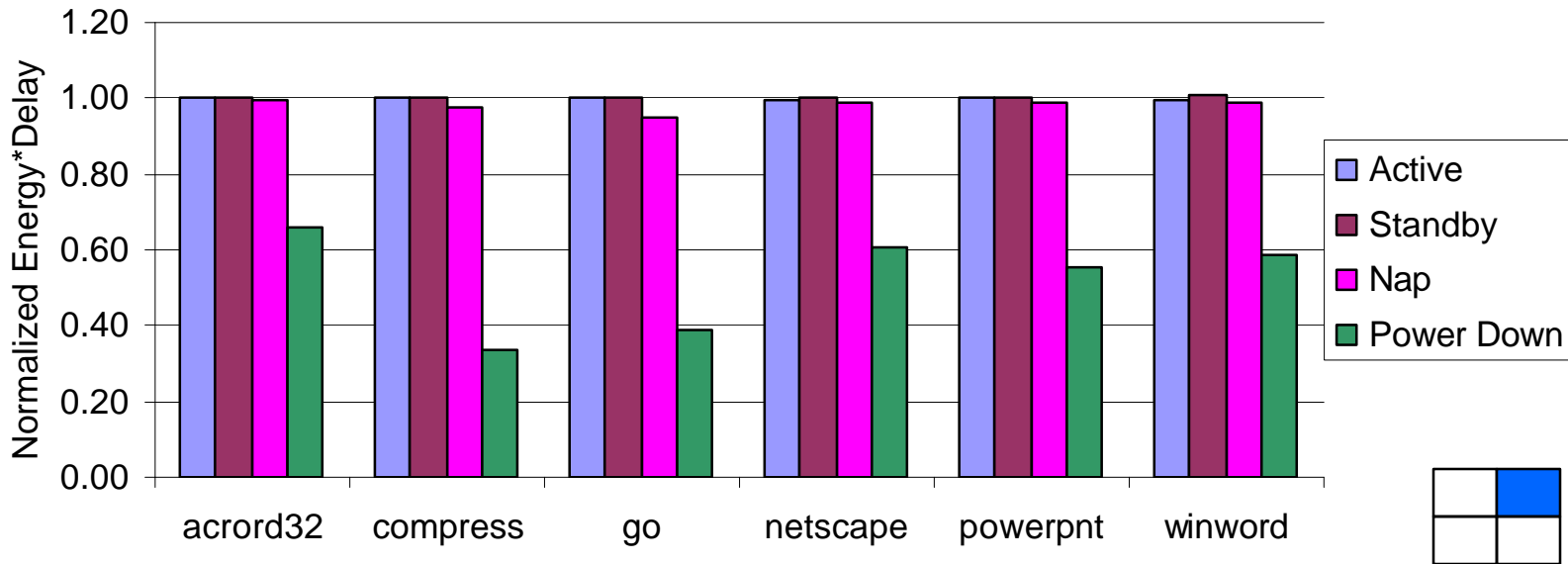
Dual-state
Hardware
(static)

Nap is best
60%-85%
improvement

Can the OS help?

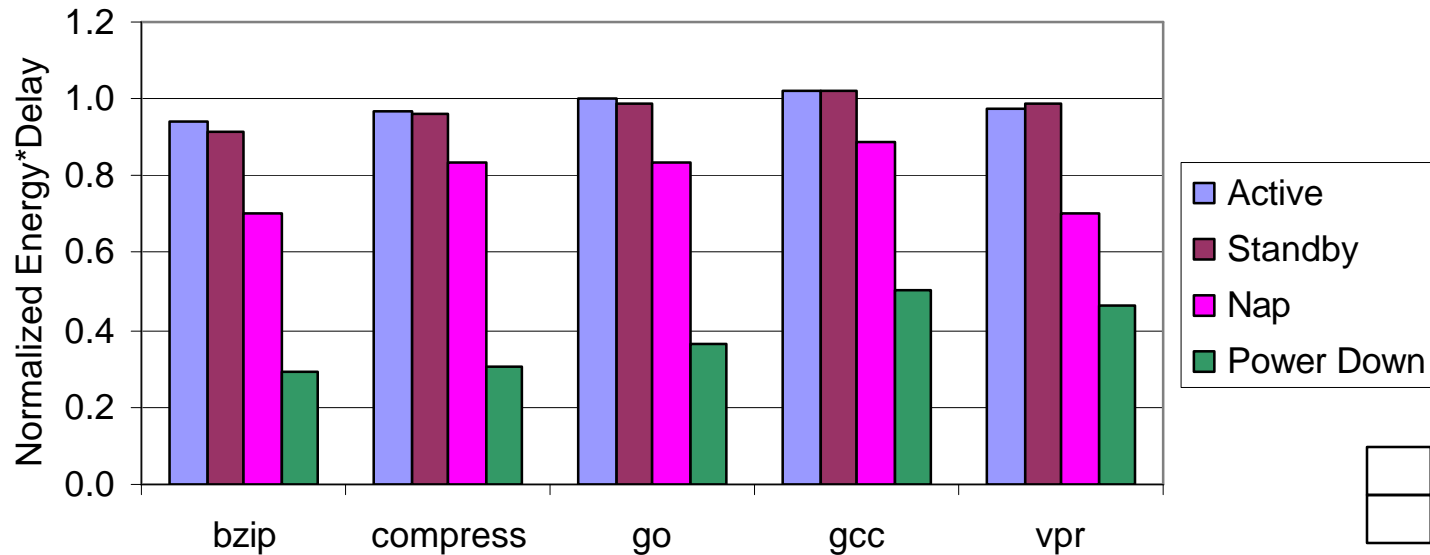
Quad-state
Hardware
(dynamic)

Benefits of Sequential Allocation (NT Traces)



- ✍ Sequential normalized to random for same dual-state policy
- ✍ Very little benefit for most modes
 - Helps PDN, but it's still really bad

Benefits of Sequential Allocation (SPEC)



- ✍ Sequential normalized to random for same dual-state policy
- ✍ 10% to 30% additional improvement for dual-state nap
- ✍ Some benefits due to cache effects

The Design Space

Random
Allocation

Sequential
Allocation

Dual-state
Hardware
(static)

Nap is best
60%-85%
improvement

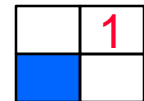
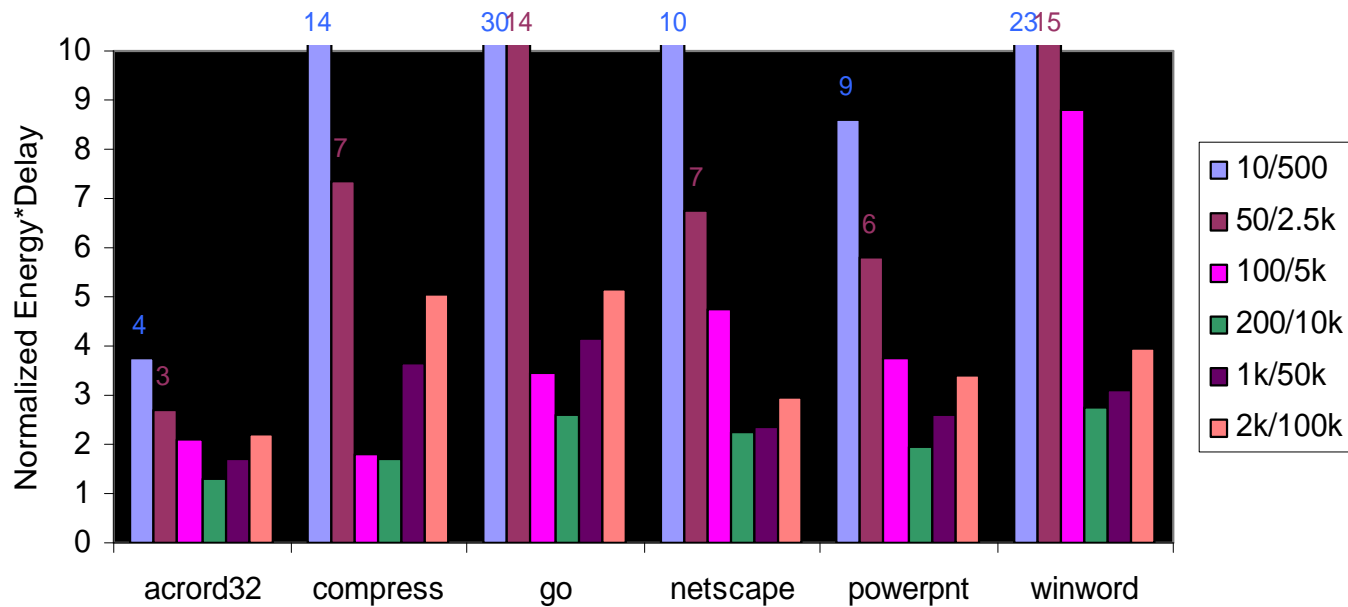
10% to 30%
improvement for
nap. Base for
future results

Quad-state
Hardware
(dynamic)

What about
smarter HW?

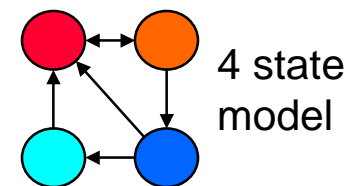
Smart HW and
OS support?

Quad-state HW + Random Allocation (NT)

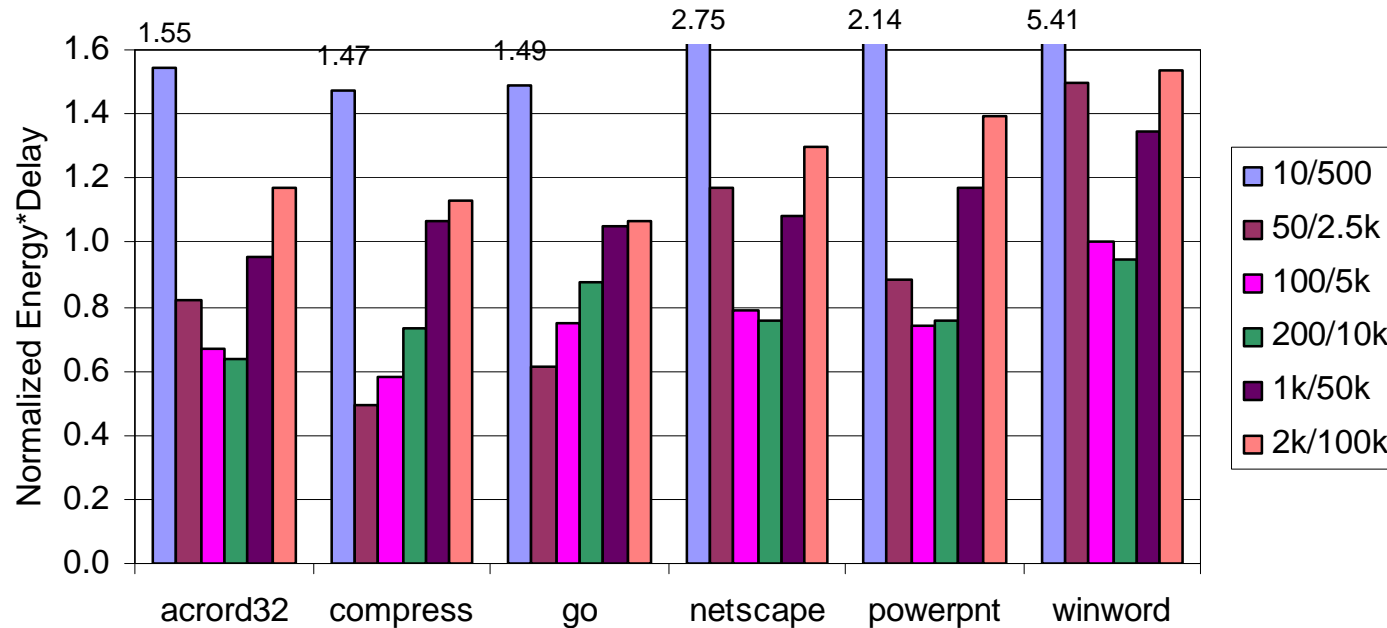


✍ Quad-state random vs. Dual-state nap sequential

✍ Sophisticated HW not enough



Quad-state HW + Sequential Allocation (NT)

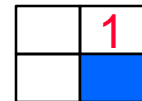


✍ Quad-state vs. Dual-state nap sequential

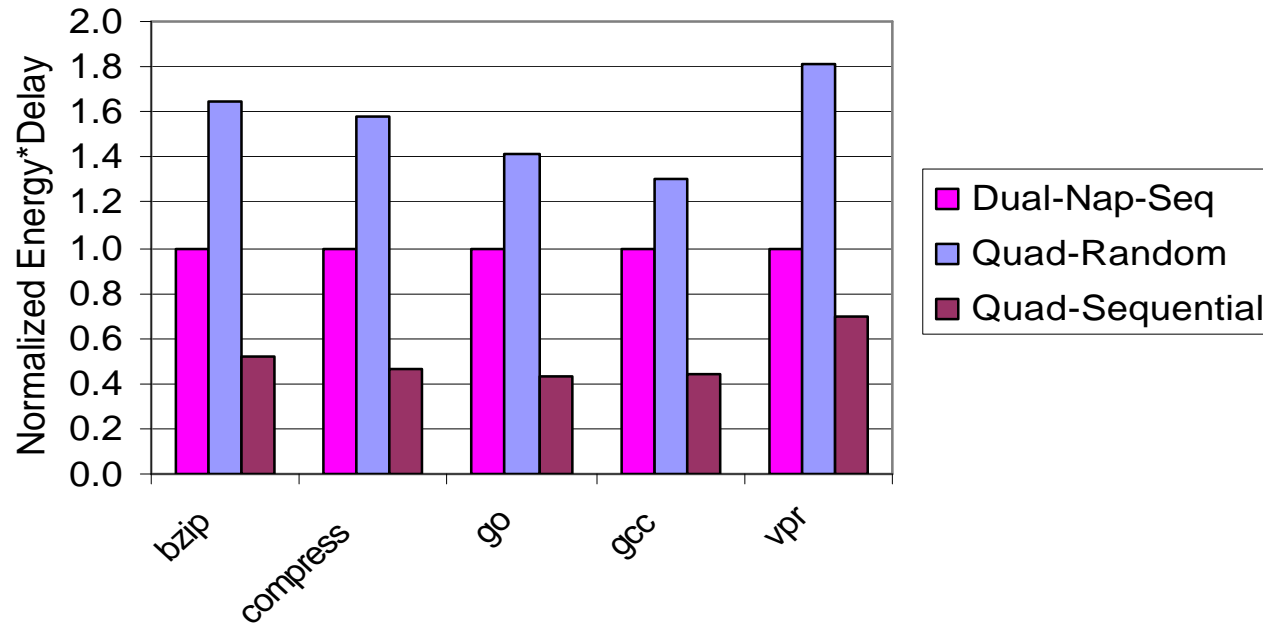
✍ Bars: Active to Nap / Nap to PDN threshold values

- Best thresholds match general results of analysis

✍ 6% to 50% improvement over best dual-state

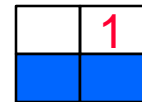


Quad-state HW (SPEC)



✍ Base: Dual-state Nap Sequential Allocation

✍ Thresholds: 0ns A->S; 750ns S->N; 375,000 N->P



- Detail: Can hide STBY to Active Delay

✍ Quad-state + Sequential 30% to 55% additional improvement over dual-state nap sequential

✍ Sophisticated HW not enough

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)	Thresholds not obvious, Could be equal to dual-state	Best Approach: 6% to 55% over dual-nap-seq, 99% to 80% over all active.

Outline

✍ Motivation

✍ The opportunity: Power Aware DRAM

✍ Hardware power management policies

✍ Operating system page allocation policies

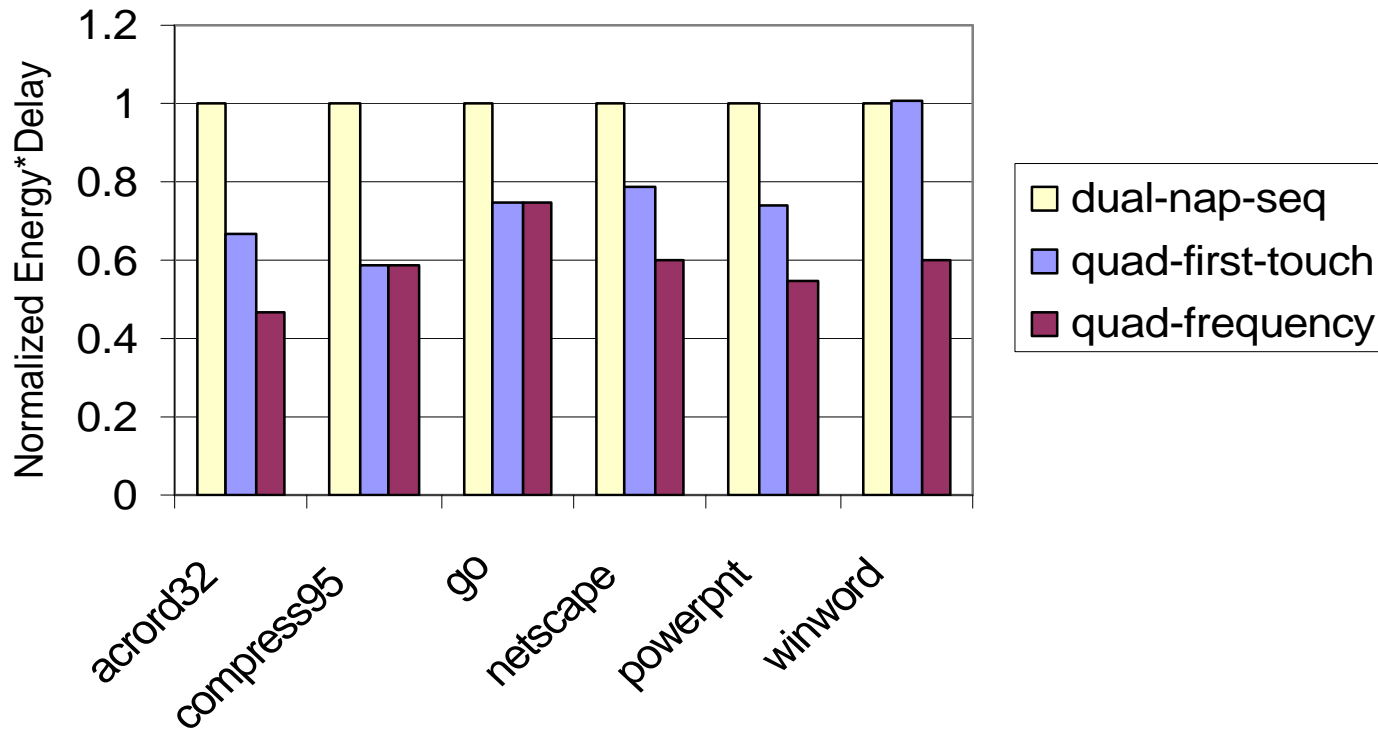
✍ Results

- Need both HW and OS support to maximize benefits
- Frequency-based Allocation

Better Page Allocation?

- ✍ First-touch not always best
 - Allow movement after first-touch
 - Preliminary results
- ✍ Frequency-based allocation
- ✍ Offline algorithm: sort by page count
- ✍ Allocate sequentially in decreasing order
 - Packs most frequently accessed pages into first chip
 - Gain insight into potential benefits of page movement

Frequency vs. First-Touch (NT)



✍ Base: dual-state nap sequential

✍ Thresholds: 100 A->N; 5,000 N->PDN

✍ Opportunity for further improvements beyond first-touch

Hardware Support for Page Movement

- ✍ Reserve 128 pages in chip 0
- ✍ 10-bit saturating counter per physical page
- ✍ Warmup for 100ms, sample accesses for 2ms
- ✍ Interrupt OS, sort counts, move 128 most frequently accessed to chip 0, repack others into minimum chips
- ✍ Use 0.011ms and 0.008mJ for page move
 - Measured from execution-driven bcopy
- ✍ **10% improvement for winword**
- ✍ Need to add to execution-driven simulator

Conclusion

- ✍ Energy is an important metric for Post-PC computing
- ✍ Memory is unexplored, but important
- ✍ New DRAM technologies provide opportunity
 - Multiple power states
- ✍ Hardware power mode management
- ✍ Effects of operating system page allocation
- ✍ **Cooperative hardware / software solution is best**

Future Work

- ✍ Page movement
- ✍ Automatically determine thresholds
- ✍ Multi-programmed workload
- ✍ Software control of power states
- ✍ Reducing DRAM refresh (idle power savings)
- ✍ Power aware operating system
 - Energy is first-class resource
 - Architectural support where appropriate

Recent Results

- ✍ **Power Aware Page Allocation**, Alvin R. Lebeck, Xiaobo Fan, Heng Zeng, Carla S. Ellis, in Proceedings of Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX), November 2000.
- ✍ **Every Joule is Precious: A Case for Revisiting Operating System Design for Energy Efficiency**, A. Vahdat, A. R. Lebeck, C. S. Ellis, in the 9th ACM SIGOPS European Workshop, September 2000.
- ✍ **Energy Estimation Tools for the Palm**, Todd L. Cignetti, Kirill Komarov, Carla Schlatter Ellis, in ACM MSWiM 2000: Modeling, Analysis and Simulation of Wireless and Mobile Systems, August 2000
- ✍ **Managing the Storage and Battery Resources in an Image Capture Device (Digital Camera) using Dynamic Transcoding**, Surendar Chandra, Carla Schlatter Ellis and Amin Vahdat, in the Third ACM International Workshop on Wireless and Mobile Multimedia (WoWMMoM'00), August 11 2000.
- ✍ **The Case for Higher Level Power Management**, C. Ellis Proceedings of HotOS, March 1999.

Other Projects

TUNE

- Memory friendly programming (Transforms)
- Non-linear data layouts

ICE (Informed Caching Environment)

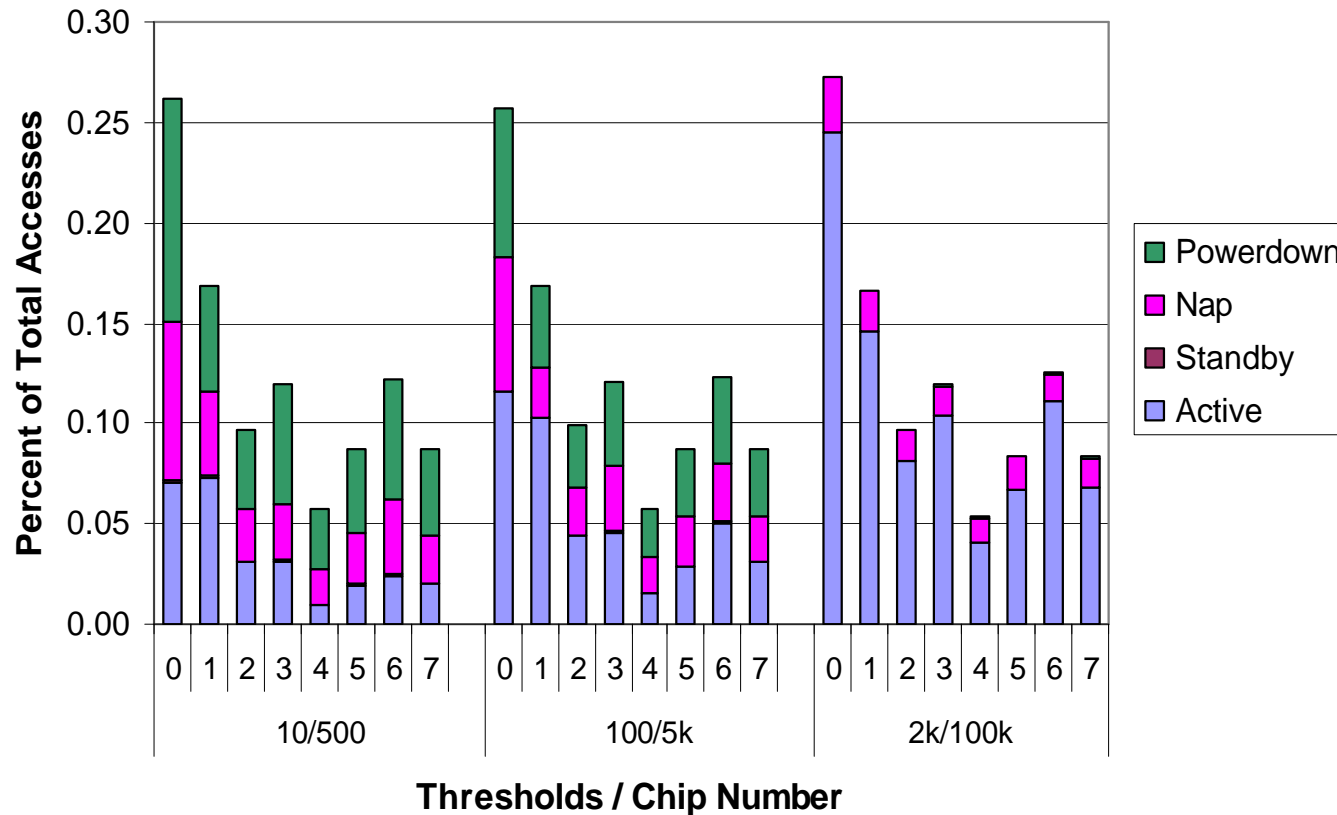
- Choreograph placement and movement of data
- Load Latency Tolerance
- Push architecture for linked data structures

Interconnection Networks

- Trapeze: Myrinet Messaging System (1Gb/s TCP)
- Self-Tuned Congestion Control for Multiprocessor Interconnection Networks

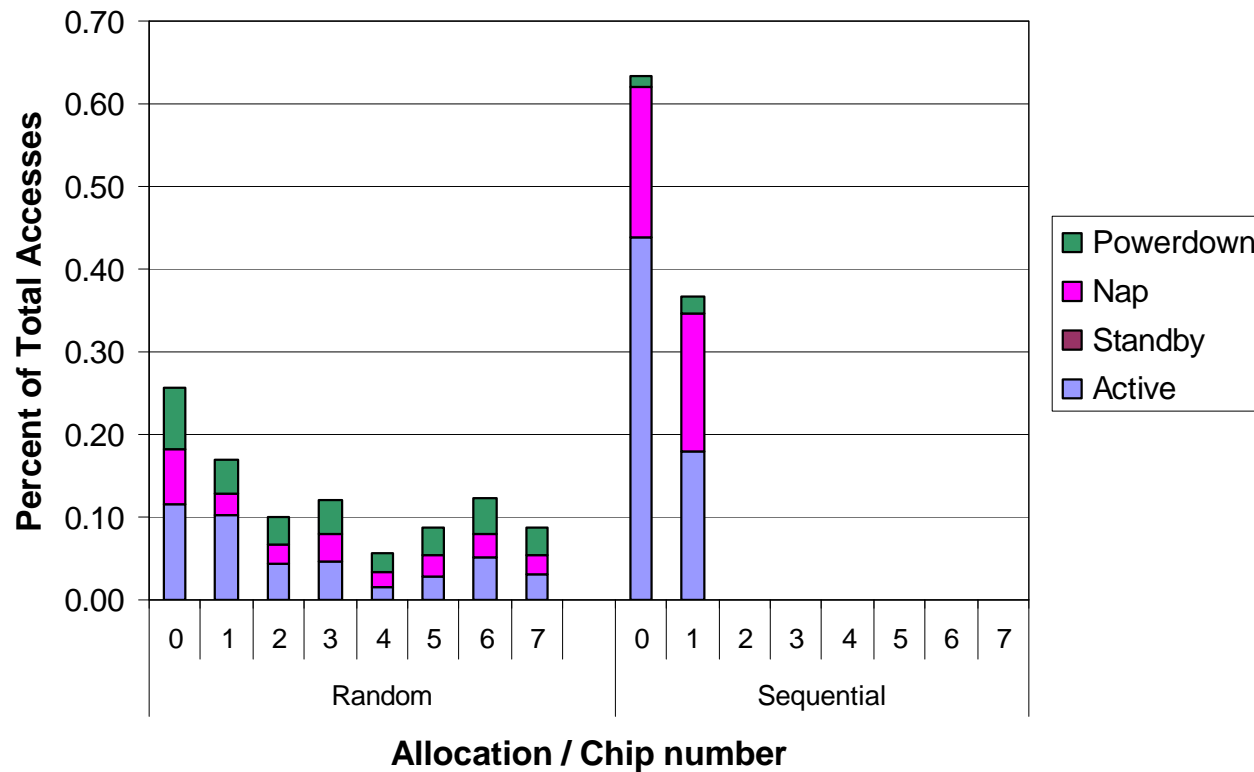
Automatically avoid saturation for variety of communication patterns

Access Distribution: Netscape



- ✍ Quad-state random
- ✍ Low threshold: too many power down accesses
- ✍ High threshold: too many active

Allocation and Access Distribution: Netscape



- ✍ Quad-state threshold 100/5k
- ✍ Random: spread across all chips
- ✍ Sequential: Increased locality, fewer chips, more active

DRAM Architecture: Interleaving

Interleaving

- Map of physical addresses to DRAM chips
- Currently assume non-interleaved
- Want page allocation to affect chip allocation
- Minimum page-grain interleaving

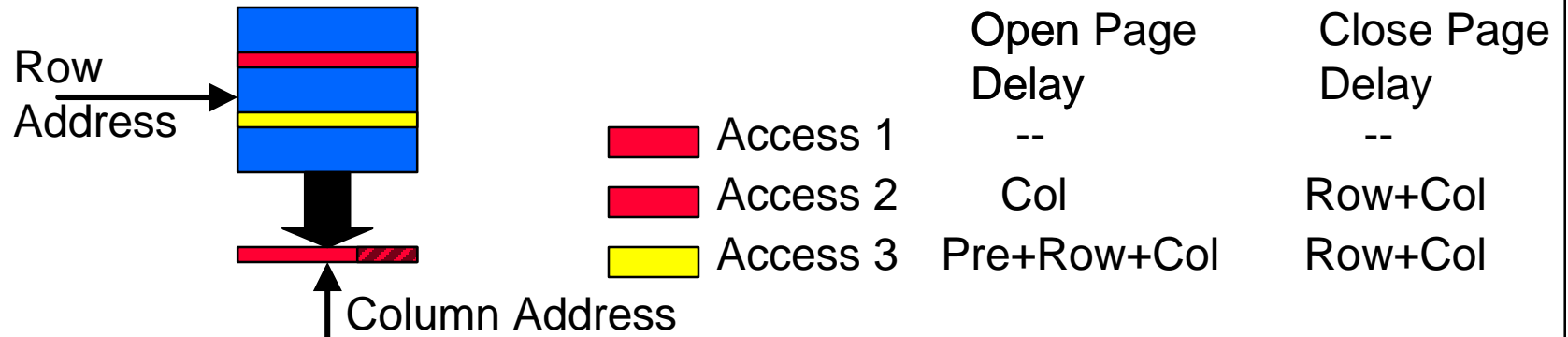
Page Interleaving

- Behaves like random allocation
- Lose part of L2 cache

Cache Block Interleaving

- Spread across internal RDRAM banks
- No change in results

DRAM Architecture: Page Policy



✍ Internal DRAM bank is 2D array

- Delay: Precharge + row + column
- DRAM Page = Row

✍ Open page: retain (cache) row data for future access

- Hit fast, no row access delay
- Miss incurs precharge delay + row access delay

✍ Close page: precharge immediately after access

- No hits, can reduce/eliminate precharge delay

Energy and DRAM Page Policy

- ✍ If any page is open, chip is Active
- ✍ All pages must be closed to enter Standby
 - Close page can enter Standby immediately
 - Open page must issue command to close pages---delay
- ✍ Close Page: Standby no extra delay over Active
 - Detail: Row and Col command busses, Data bus
 - Row cmd bus remains synchronized
 - Resynch delay of Column cmd and Data bus overlapped with Row command delay
- ✍ Close page 20% lower E*D than Open page
 - Execution-driven simulator