

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
- ✍ 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]

The Unturned Stone

Traditional OS resources

- Processor, Memory, Disk, Network

Power Studies

 Disk spin down [Baker, Douglass, Li]

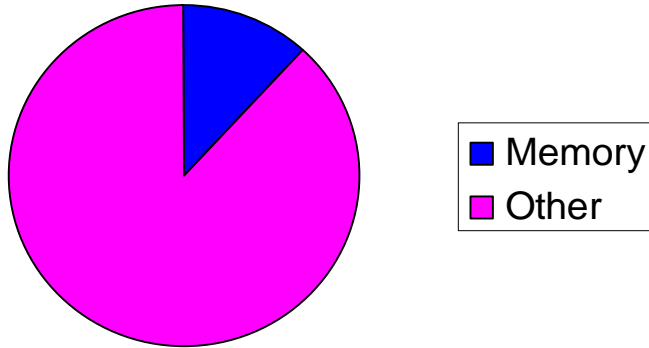
 Processor voltage/clock [Weiser, Pering, Lorch, Grunwald]

 Network interface [Stemm, Kravets, Imielinski]

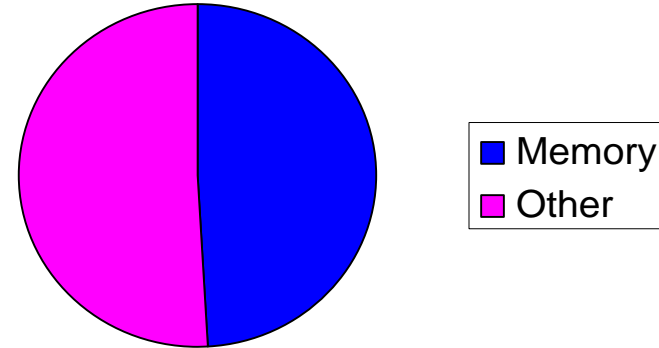
 **Where is main memory?**

Memory Power Budget

Laptop Power Budget
9 Watt Processor



Handheld Power Budget
1 Watt Processor



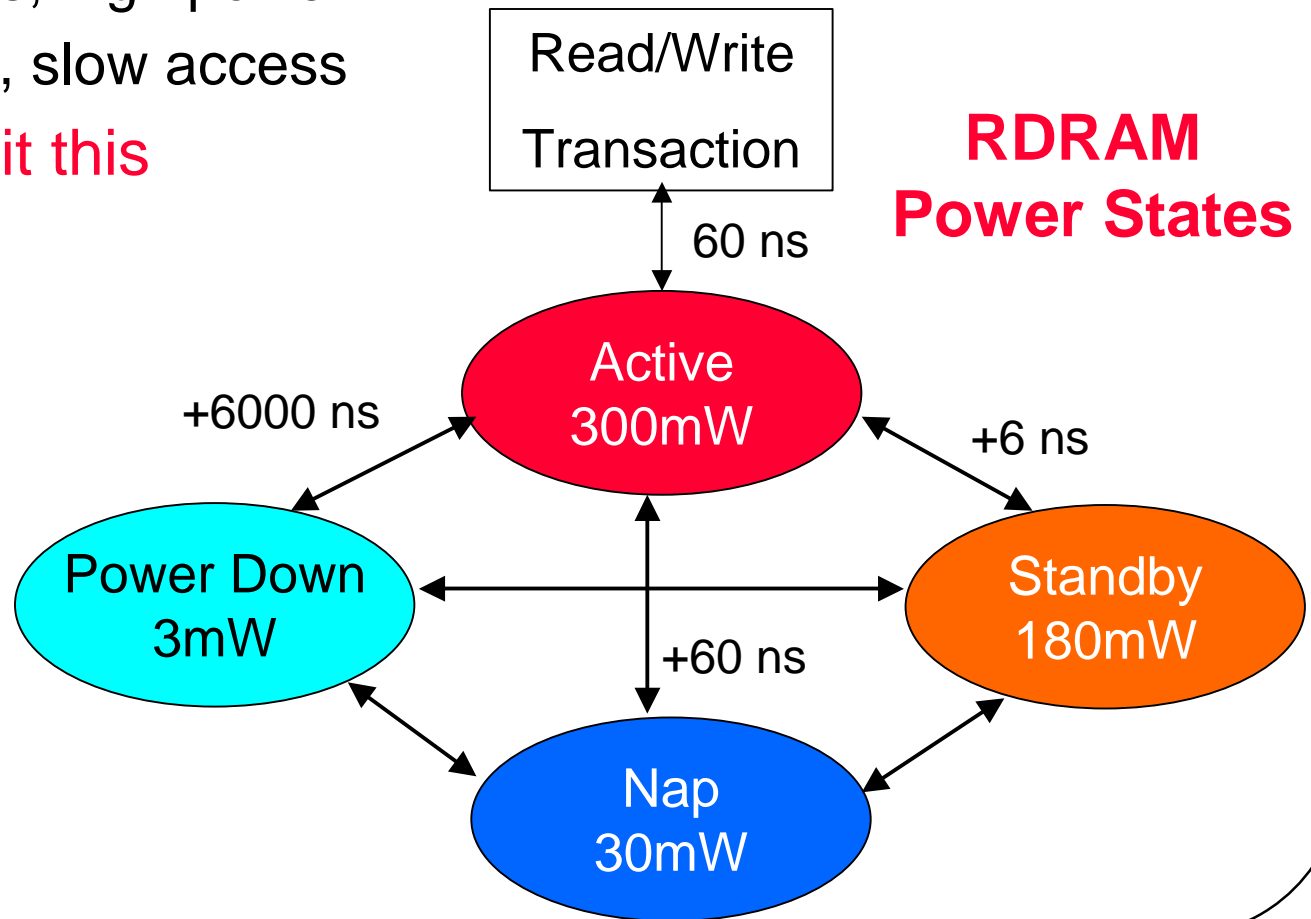
- ✍ Laptop: memory is small percentage of total power budget
- ✍ Handheld: low power processor, memory is more important

Opportunity: Power Aware DRAM

✍ Multiple power states

- Fast access, high power
- Low power, slow access

✍ How to exploit this opportunity?



Outline

 Motivation

 **The opportunity**

 Hardware power management policies

 Operating system page allocation policies

 Results

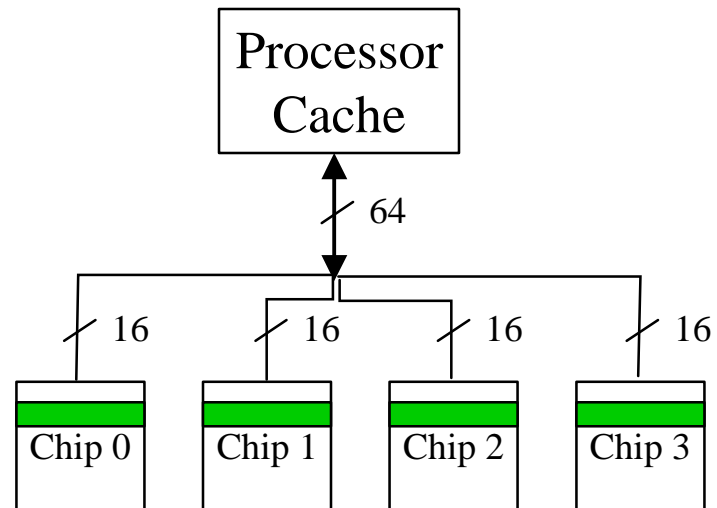
- **Need both HW and OS support to maximize benefits**
- Simple HW does very well

 Future Work

 Conclusion

Conventional Main Memory Design

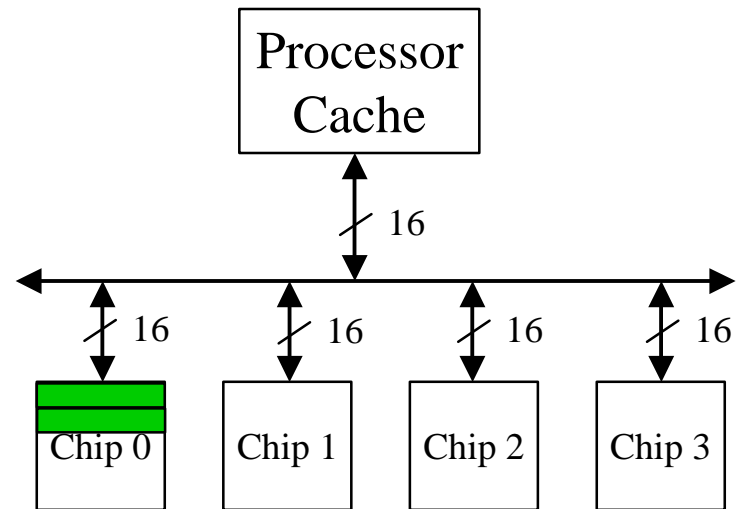
Cache Block



- ✍ 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

 Interaction between state transitions and access locality

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

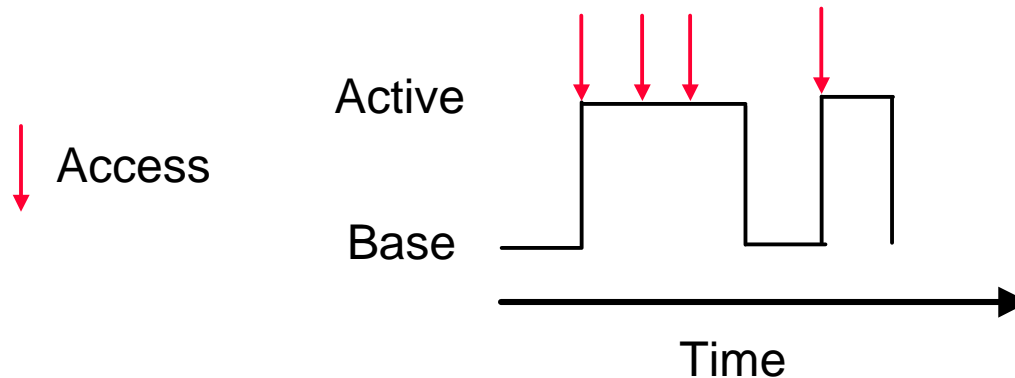
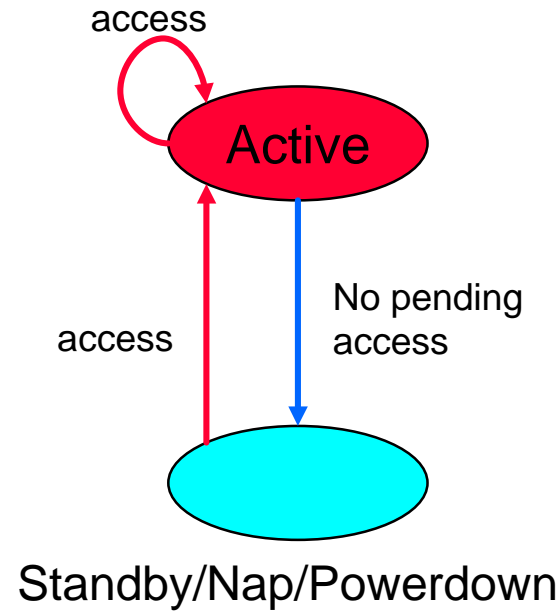
- Memory controller policies
- Quantify benefits of power states

 Q2: **What role does software have?**

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

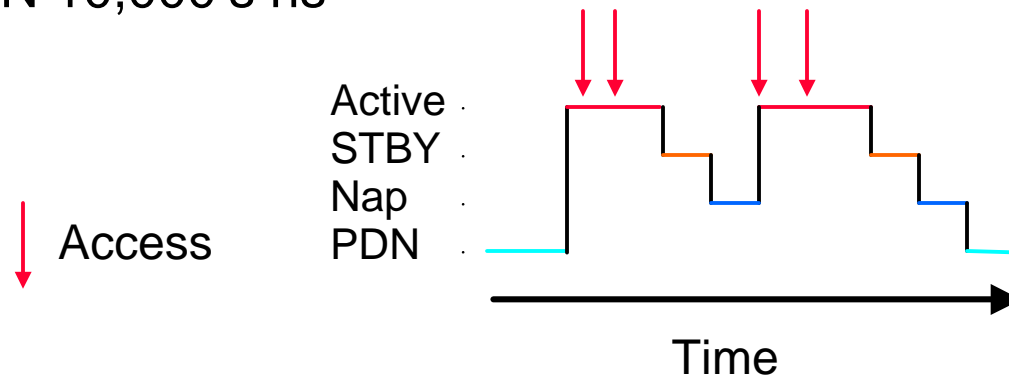
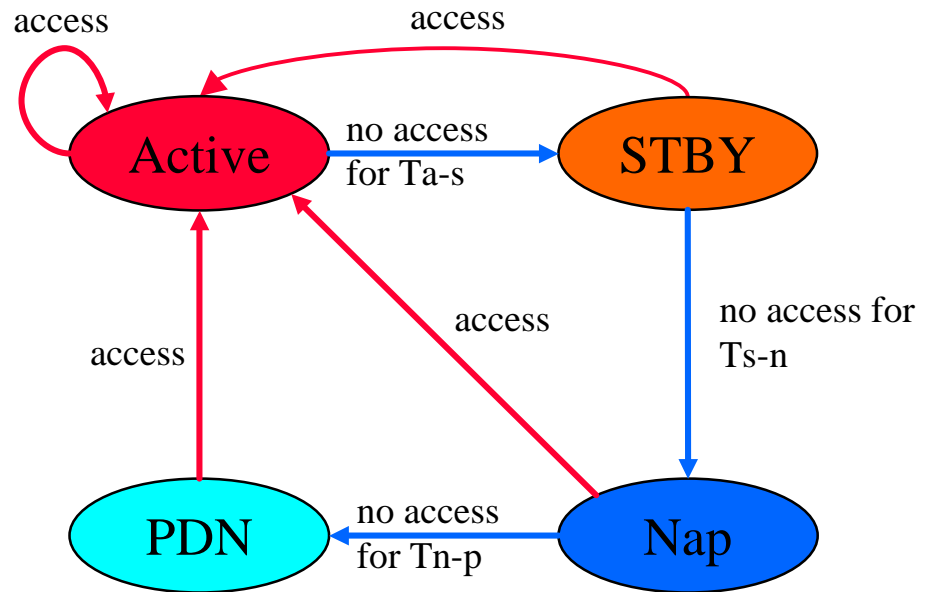
Dual-state (Static) HW Power State Policies

- ✎ All chips in same 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
- ✍ Analysis
 - Active/Stby to nap 100's of ns
 - Nap to PDN 10,000's ns



Exploiting PDRAM Power States

Hardware power management policies

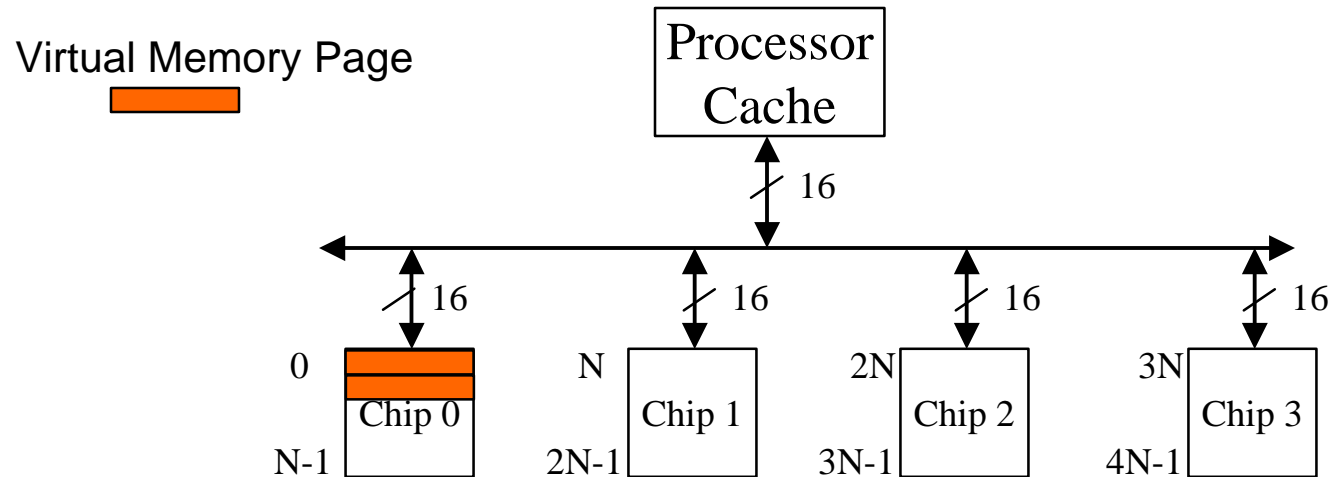
- Exploit locality to reduce energy consumption
- Dual-state model
- Quad-state model

Want to increase DRAM chip-level locality

Performance penalty with conventional DRAM

- Can ignore BW tradeoff w/ RDRAM properties

Page Allocation and PADDRAM



- ✍ 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

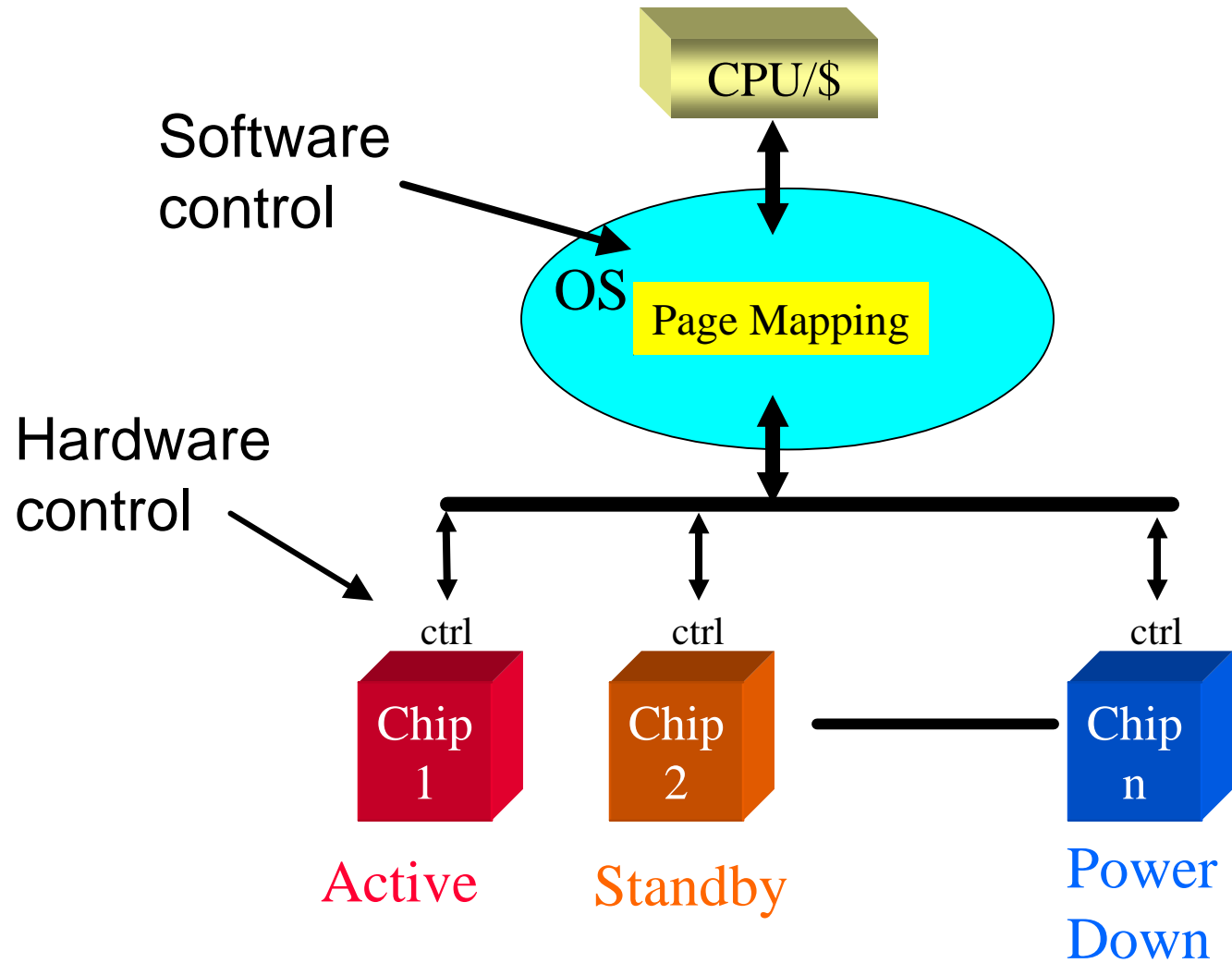
Sequential First-Touch Allocation

- Consolidate pages into minimal number of chips
- One shot

Frequency-based Allocation

- 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
- Closer look at Hardware

 Conclusion

Methodology

✍ Metric: Energy*Delay Product

- Avoid very slow solutions

✍ Energy Consumption (DRAM only)

- Processor & Cache affect runtime

✍ 8KB page size

✍ L1/L2 non-blocking caches

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

✍ Average power for transition from lower to higher state

Methodology Continued

Trace-Driven Simulation

- NT applications from U. of Washington Etch group
- Simple processor
- Eight 32Mb chips, total 32MB, non-interleaved
- See ASPLOS paper for full results

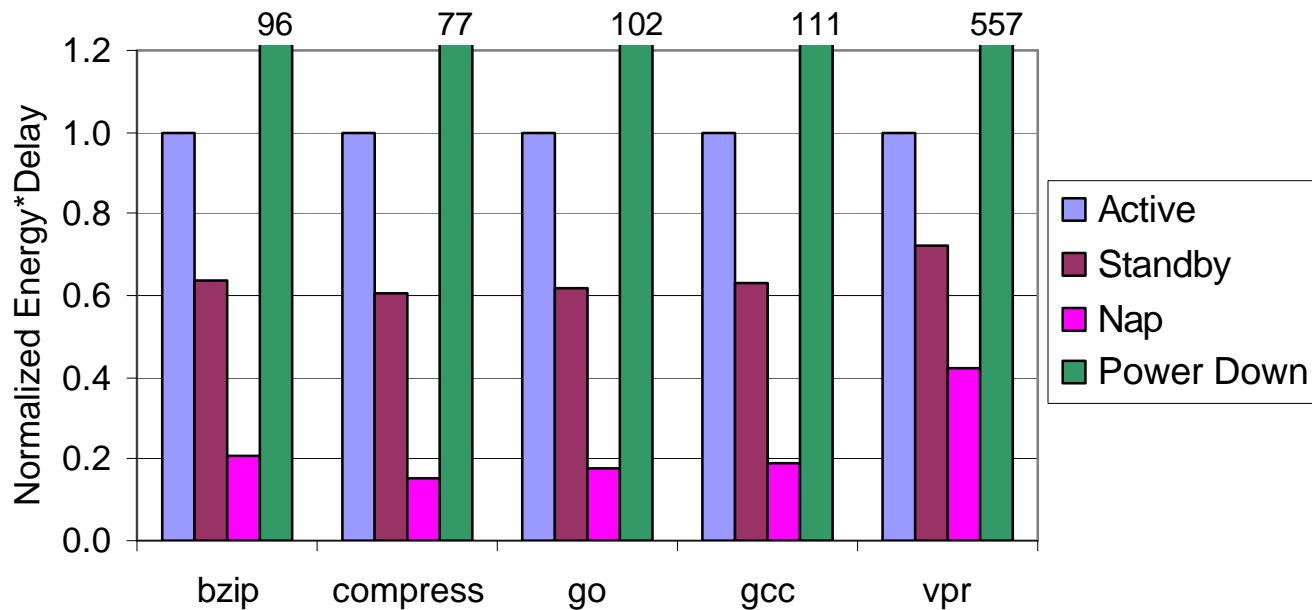
Execution-Driven Simulation

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

The Design Space

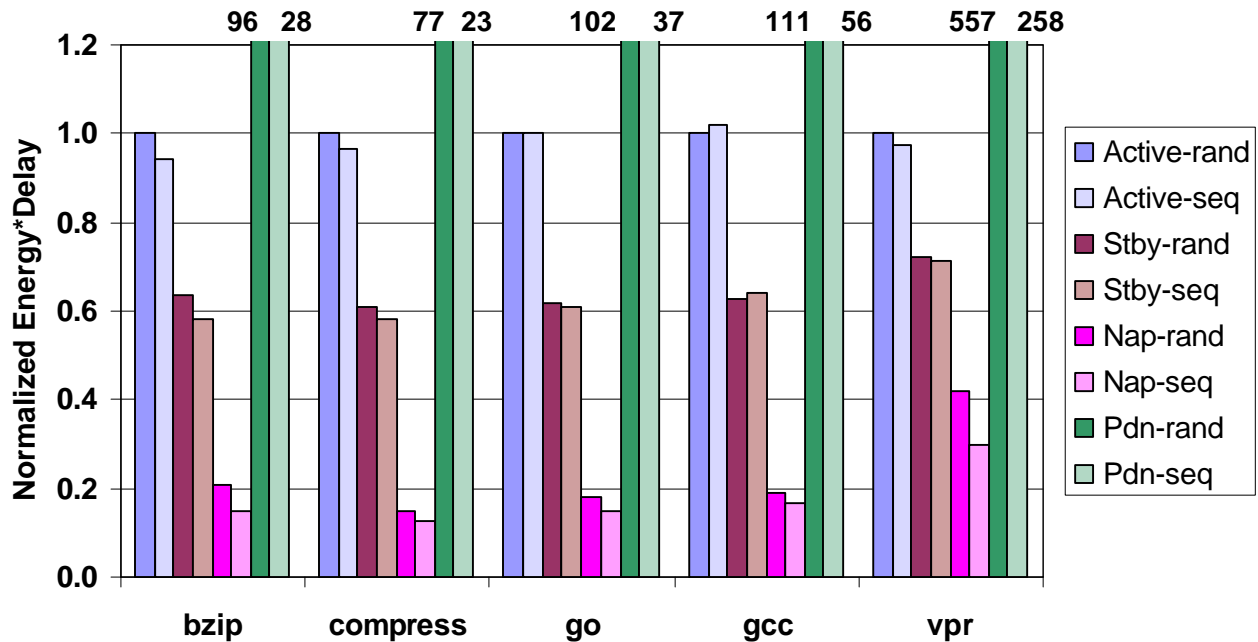
	Random Allocation	Sequential Allocation
Dual-state Hardware (static)	1 Simple HW	2 Can the OS help?
Quad-state Hardware (dynamic)	3 Sophisticated HW	4 Cooperative HW & SW

Dual-state + Random Allocation



- ✍ 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

Benefits of Sequential Allocation



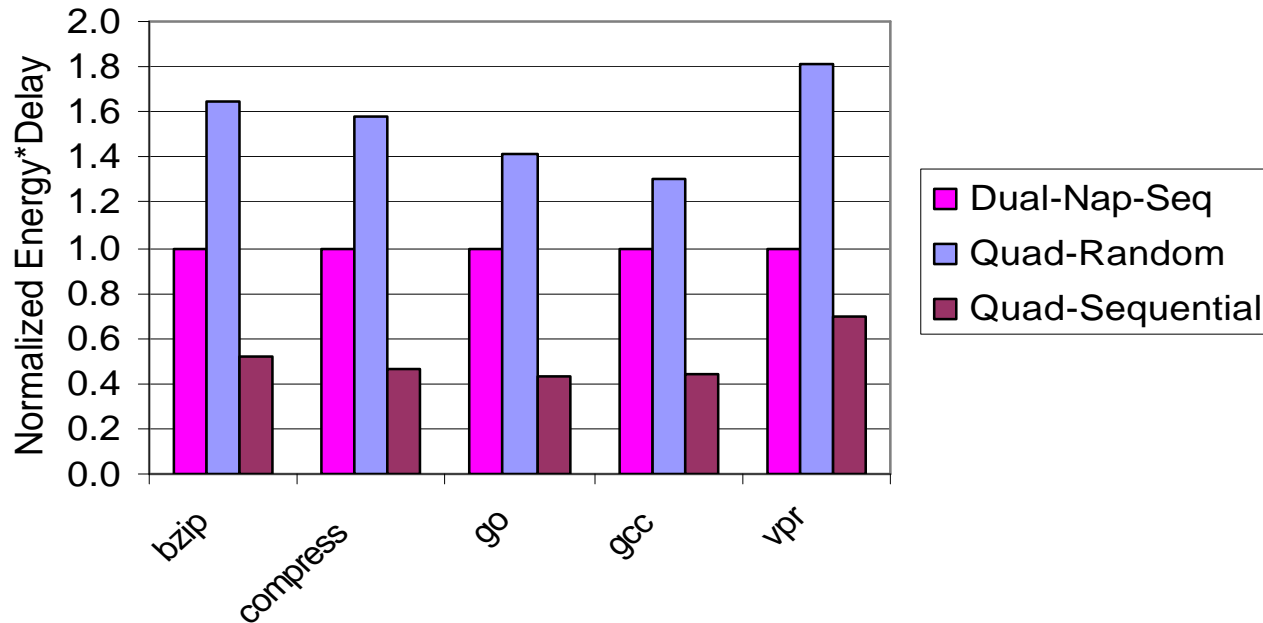
✍ 10% to 30% relative improvement for dual-state nap

✍ Some benefits due to cache effects

The Design Space

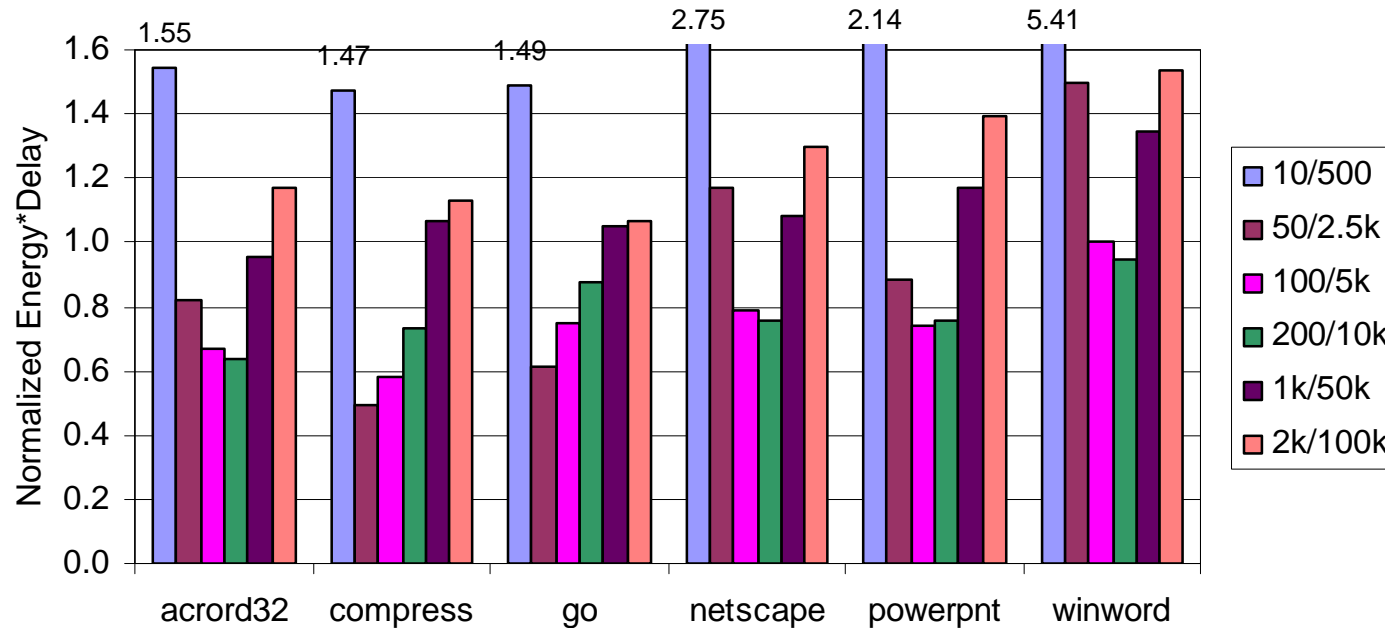
	Random Allocation	Sequential Allocation
Dual-state Hardware (static)	Nap is best 60%-85% improvement	10% to 30% improvement for nap. Base for future results
Quad-state Hardware (dynamic)	What about smarter HW?	Smart HW and OS support?

Quad-state Hardware



- ✍ Base: Dual-state Nap Sequential Allocation
- ✍ Thresholds: 0ns A->S; 750ns S->N; 375,000 N->P
- ✍ Quad-state + Sequential 30% to 55% additional improvement over dual-state nap sequential
- ✍ Sophisticated HW must get thresholds correct

Threshold Sensitivity (NT Traces)



✍ Quad-state seq. vs. Dual-state nap seq.

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

- Best thresholds match general results of analysis

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

The Design Space

Random
Allocation

Sequential
Allocation

Dual-state
Hardware
(static)

Nap is best
dual-state policy
60%-85%
improvement

Additional
10% to 30%
improvement
over Nap

Quad-state
Hardware
(dynamic)

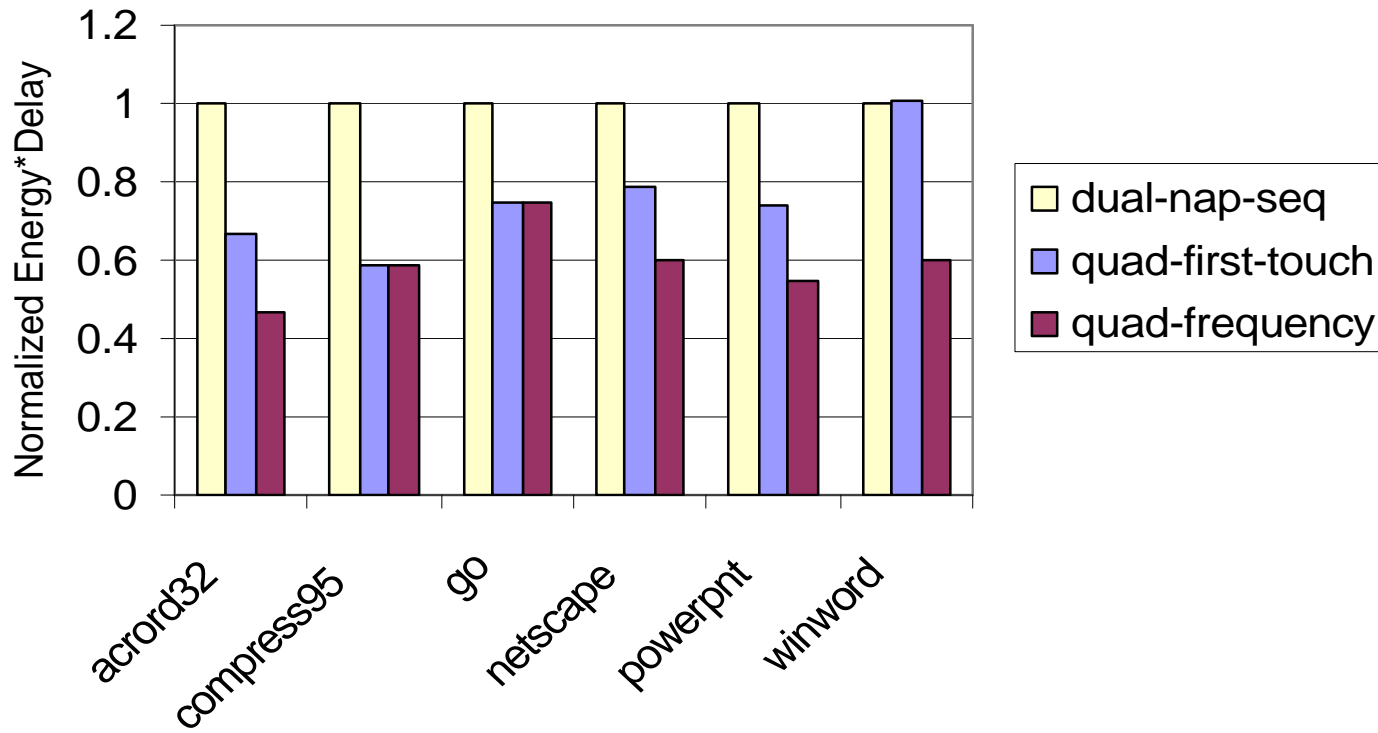
Thresholds not
obvious,
Could be equal
to dual-state

Best Approach:
6% to 55% over
dual-nap-seq,
80% to 99% over
all active.

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

Outline

✍ Motivation

✍ The opportunity: Power Aware DRAM

✍ Hardware power management policies

✍ Operating system page allocation policies

✍ Results

- Simple Hardware works well
- Need both HW and OS support to maximize benefits
- Closer look at HW policies

Hardware Policies [ISLPED '01]

✍ Without OS support

- Thresholds not obvious

✍ With OS support

- Idle DRAM chips power down
- Can we decouple thresholds for Nap & Power down?

✍ Recent studies advocate “smarter” policies [[Delaluz HPCA '01](#)]

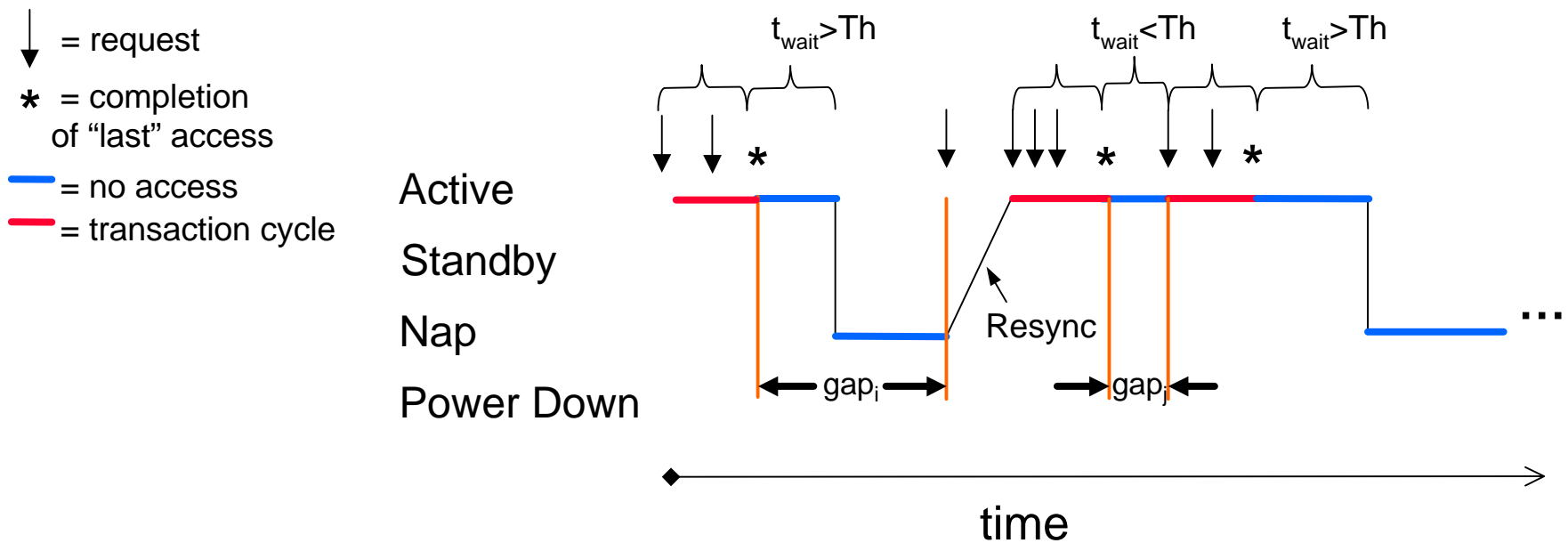
- No caches, no virtual memory

✍ Are “smarter” policies required for cache-based systems?

✍ Use analytic modeling to evaluate space

✍ Validate with simulation

Key Parameters for Model



✍ **Gap** (?): time between clusters of accesses

✍ **Threshold** (Th): time to remain in high power before transition

✍ How does gap affect threshold selection?

Analytic Model

Assume the density function of the gap distribution is $p(g)$, then mean gap is

$$\int_0^{\infty} p(g) dg$$

Mean time of staying in low power

$$t_{nap} = \int_{Th}^{\infty} (g - Th) p(g) dg$$

Energy saving

$$e_1 (P_{active} - P_{nap}) t_{nap}$$

Energy cost for resync

$$e_2 P_{nap} t_{nap} = e_2 T_{nap} \text{Prob}(gap > Th)$$

Mean Energy cost

$$e_1 + e_2 T_{nap} \text{Prob}(gap > Th)$$

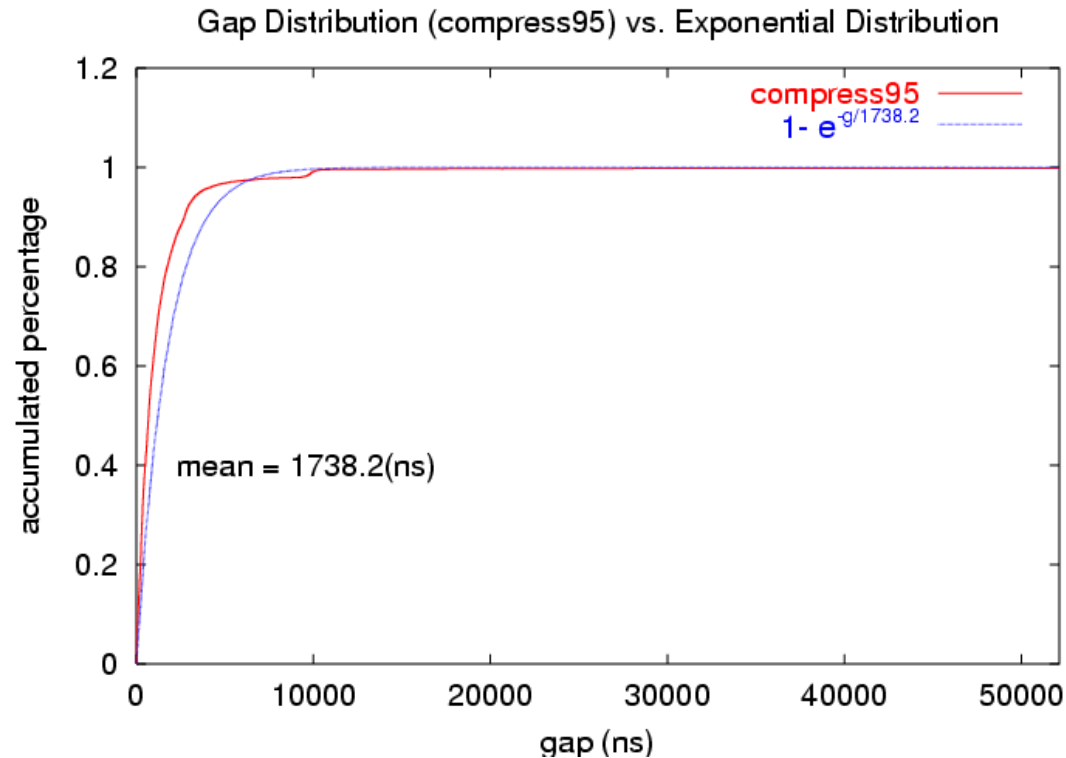
Mean Delay cost

$$d T_{nap} \text{Prob}(gap > Th)$$

Mean per-gap change of $e \cdot d$ is a function of mean gap and threshold

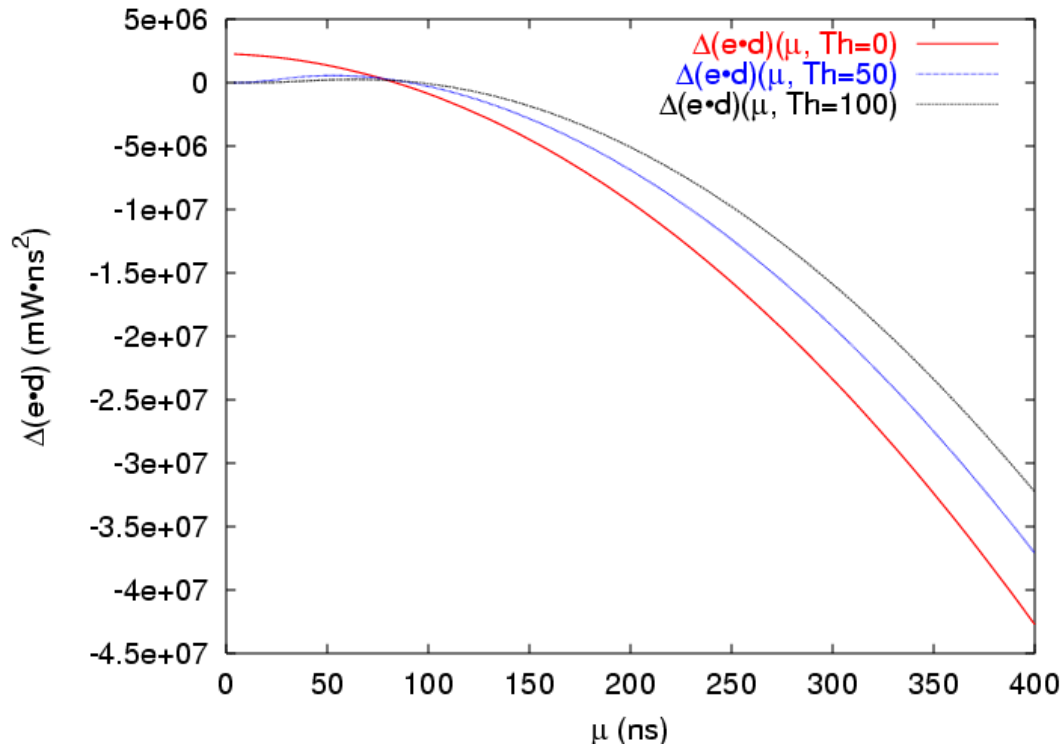
$$(e \cdot d) = f(\text{mean gap}, Th)$$

Distribution of Accesses



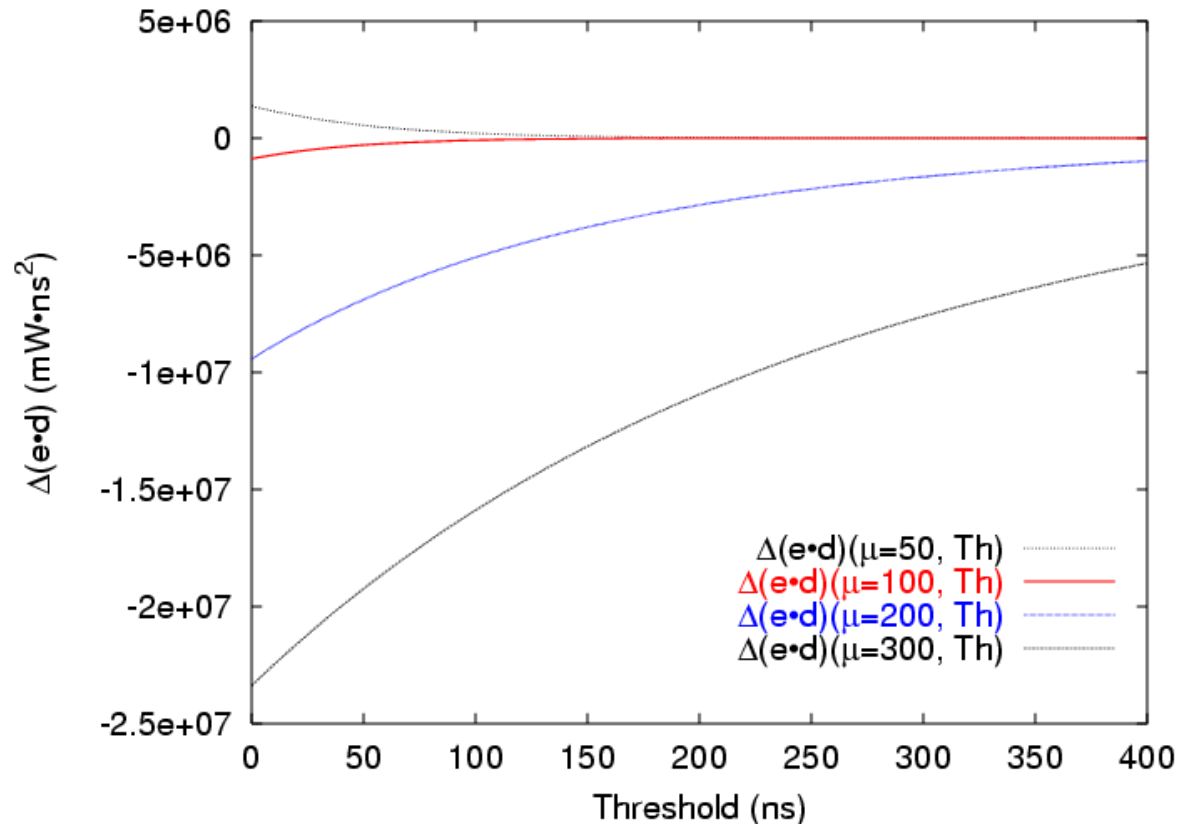
- ✍ Can approximate gaps using exponential
- Same mean (μ)
 - Use analytic model instead of many many simulations

Change in E^*D vs. Gap



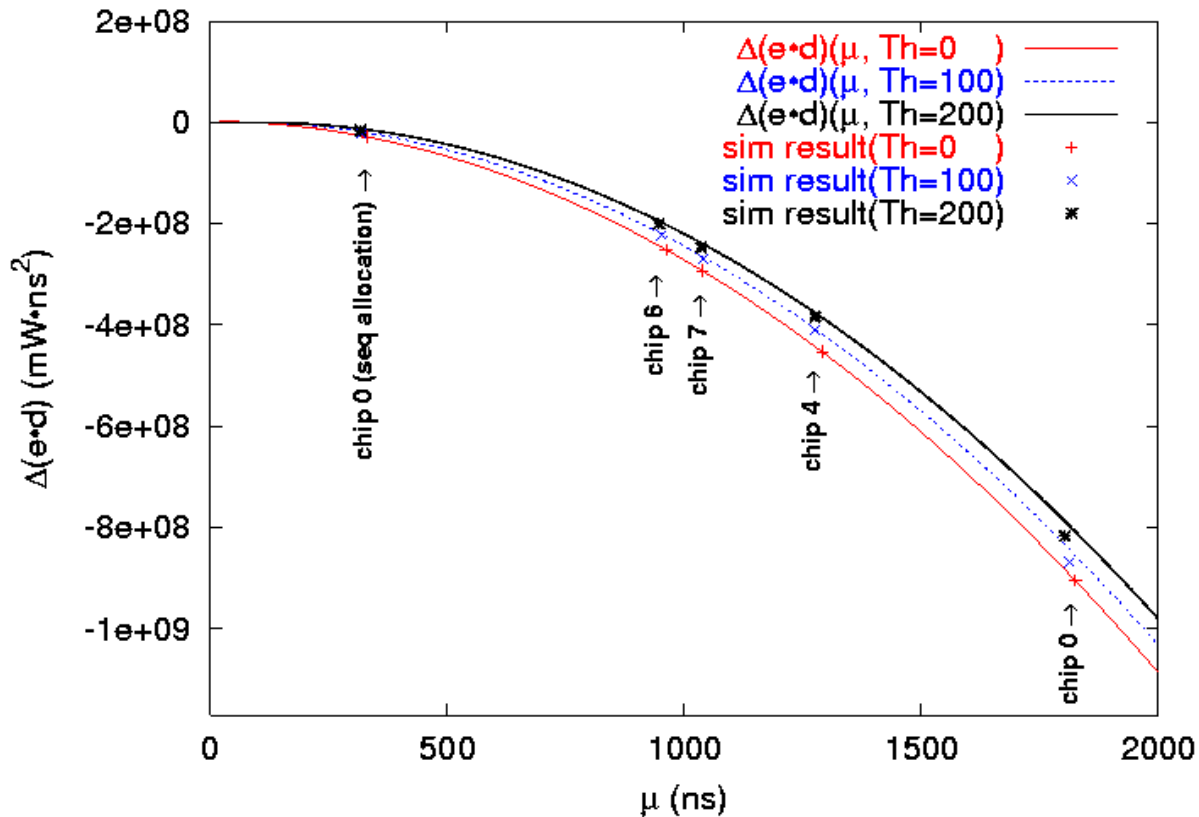
- ✍ Relative E^*D savings for one DRAM chip (lower better)
- ✍ Transition immediately to lower power state ($Th = 0$) is best for reasonably large average gap
 - Resynch cost for small gaps

Change in E*D vs. Threshold



✍ For fixed average gap, increasing threshold reduces amount of time in lower power state

Model Validation



 Simulation within 5% for most cases

- 20%-50% for small average gap (small values)

 Long wait time for PDN, 0 threshold for Active -> Nap

Future/Current Work

Memory Power

- Interaction of voltage scaling and thresholds
- Page movement
- Multi-programmed workload

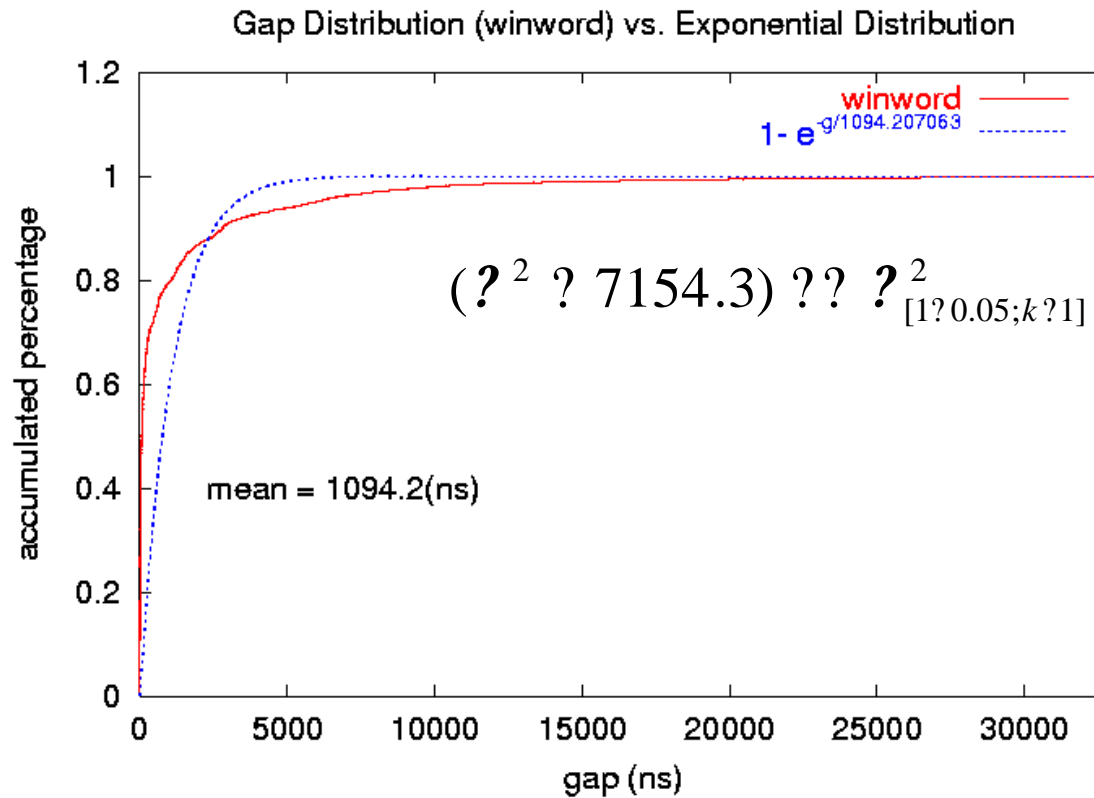
Power Aware Operating System

- Energy is first-class resource
- Unified energy accounting and allocation model
- Linux prototype
- Architectural support where appropriate

Conclusion

- ✍ Energy is an important metric for Post-PC computing
- ✍ Memory is unexplored, but important
- ✍ New DRAM technologies provide opportunity
 - Multiple power states
- ✍ **Cooperative hardware / software solution is best**
- ✍ **Simple HW for cache-based systems**
- ✍ Power Aware Operating System

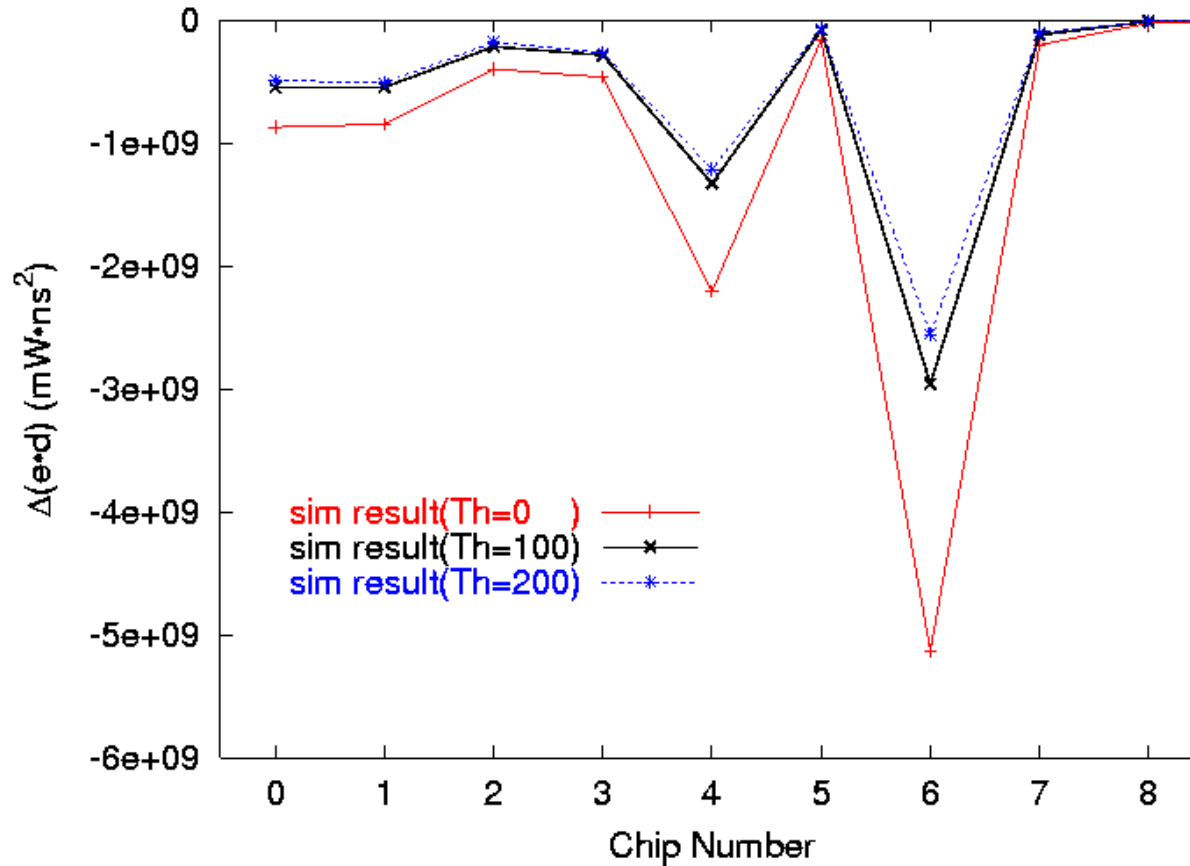
DRAM Access Characteristics



✎ Can not be considered as exponential

✎ Need to find other distribution that matches it

Winword Model Validation

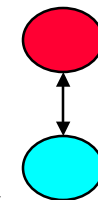
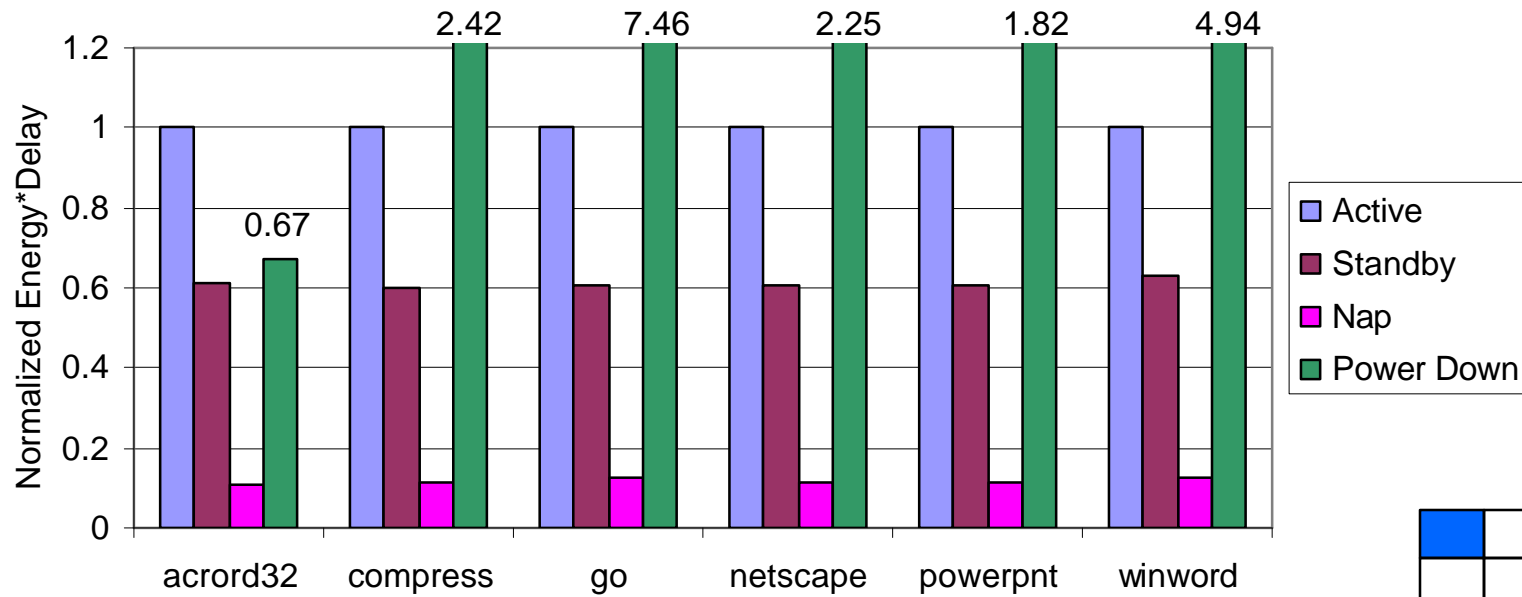


- Threshold 0 is still the best
- SimpleScalar/Spec2k qualitatively similar results

Typical Gap Estimation

- ✍ 500MHz PIII ~ 225.2 MIPS
- ✍ 1 instruction/4 ns
- ✍ Assume: 1 memory access / 2 instructions
- ✍ 1 memory access / 8 ns
- ✍ Even with a high 10% cache miss rate, there is 80ns on average between memory accesses even they are all targeted to one chip

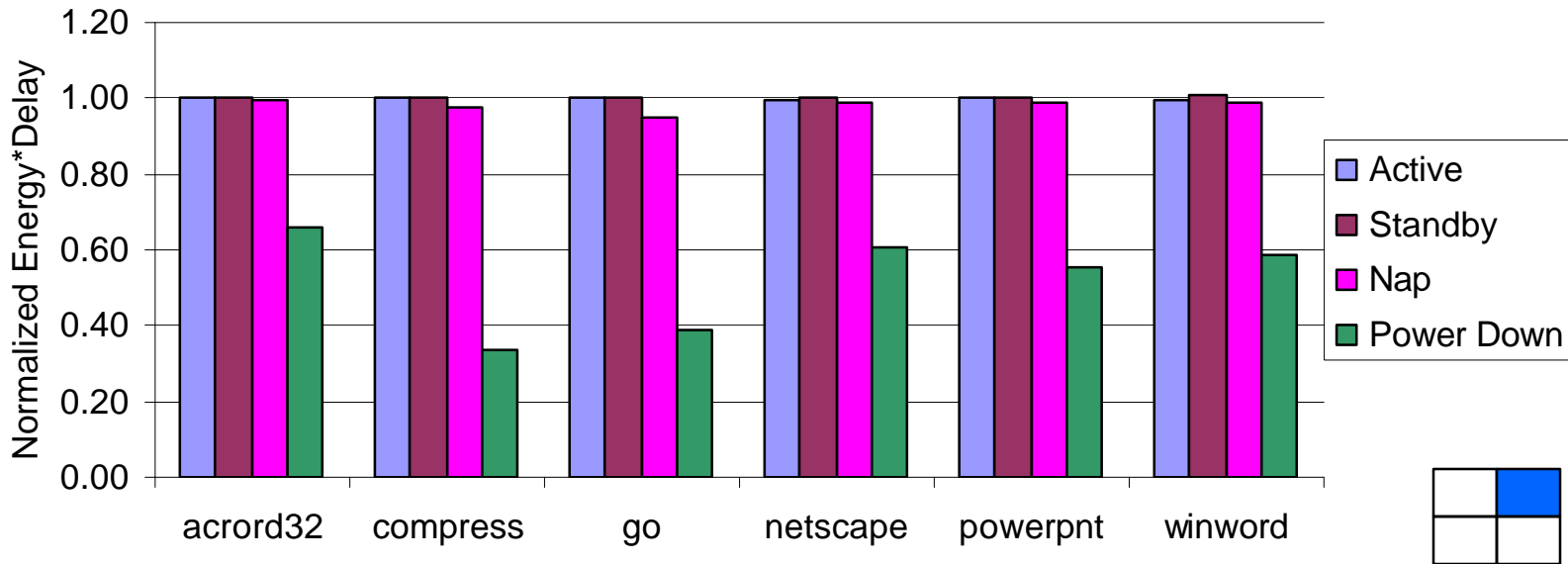
Dual-state + Random Allocation (NT Traces)



2 state model

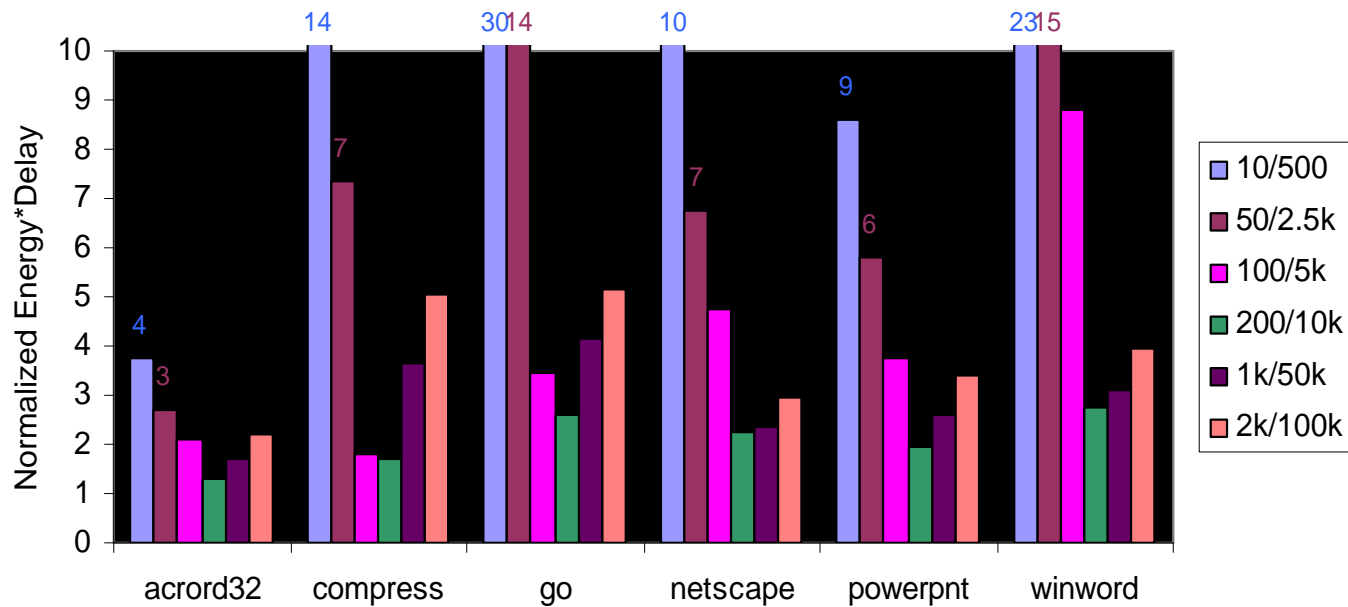
- ✍ Active to perform access, return to **base state**
- ✍ Nap is best ~85% reduction in $E \cdot D$ over full power
- ✍ Little change in run-time, most gains in energy/power

Benefits of Sequential Allocation (NT Traces)



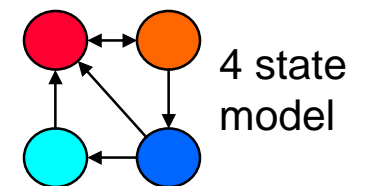
- ✍ Sequential normalized to random for same dual-state policy
- ✍ Very little benefit for most modes
 - Helps PDN, but it's still really bad

Quad-state HW + Random Allocation (NT)

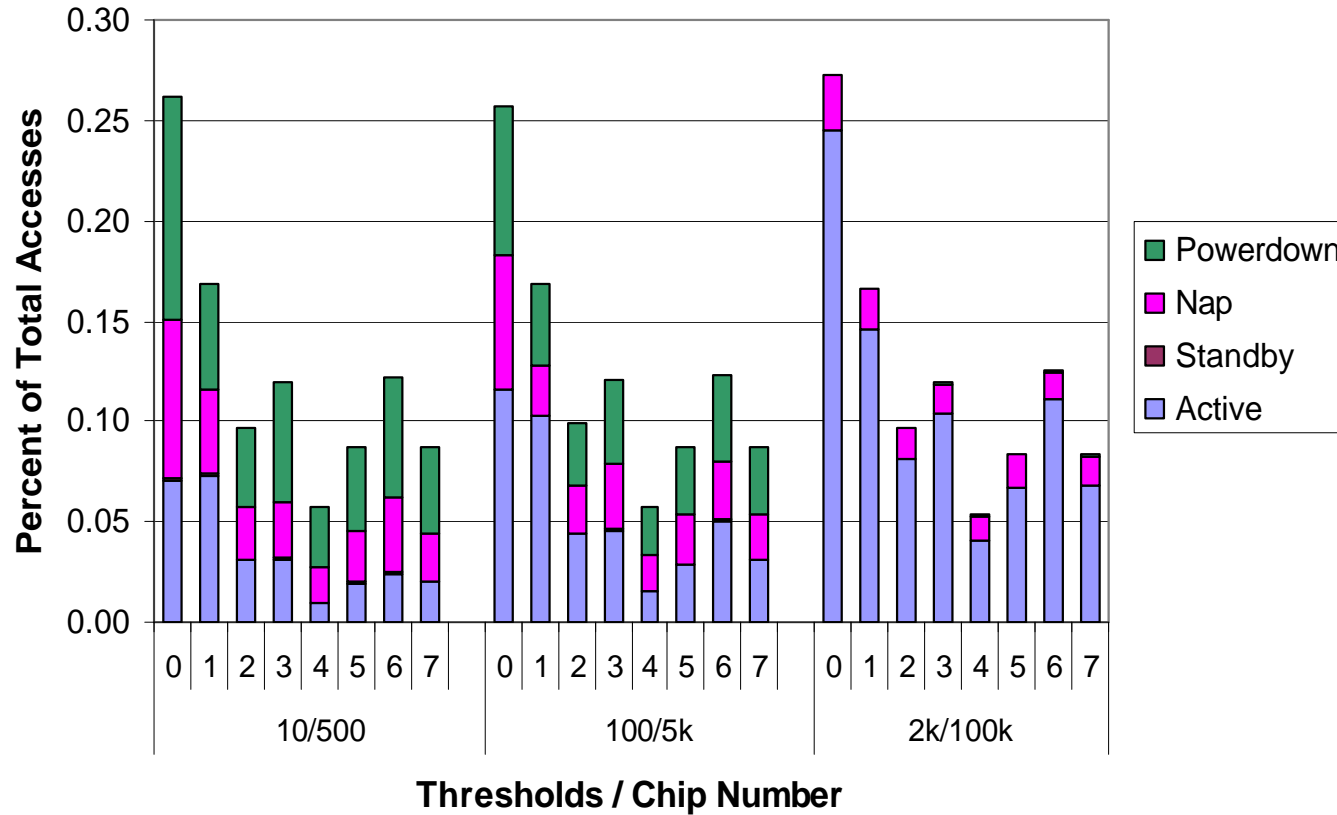


✍ Quad-state random vs. Dual-state nap sequential

✍ Sophisticated HW not enough

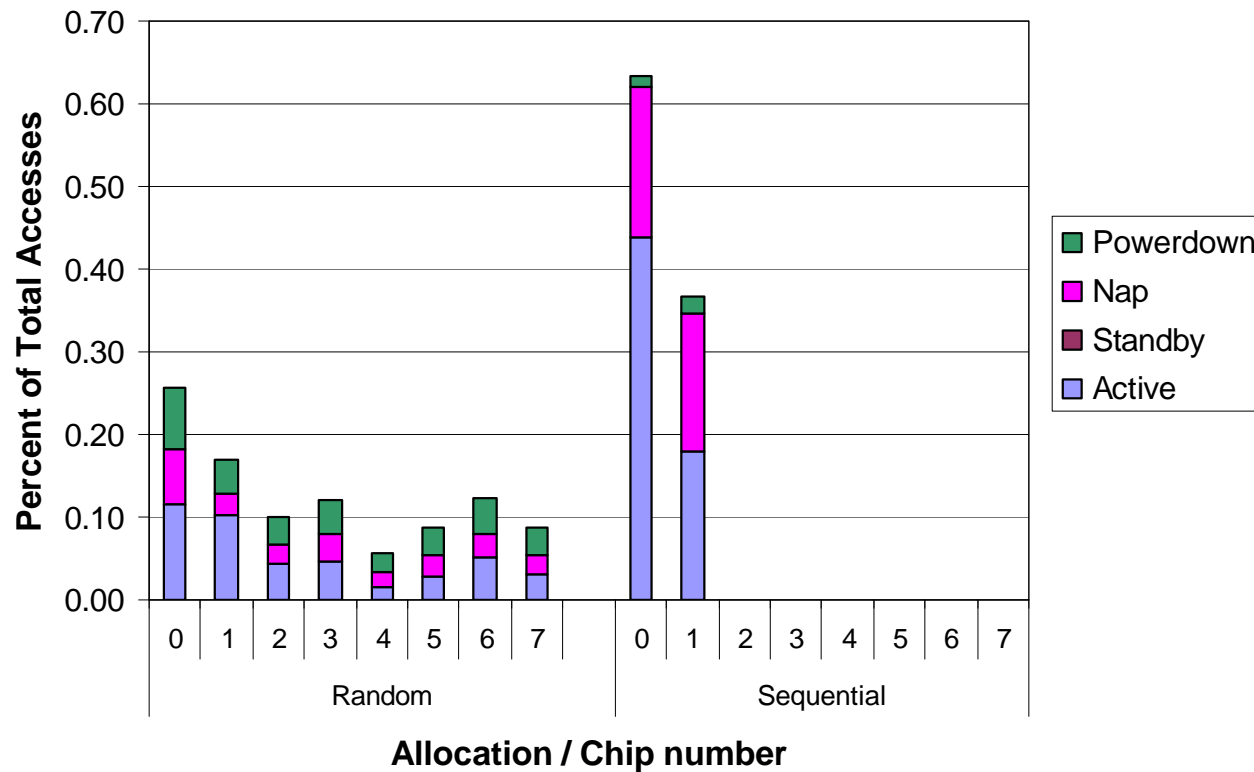


Access Distribution: Netscape



- ✍ 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

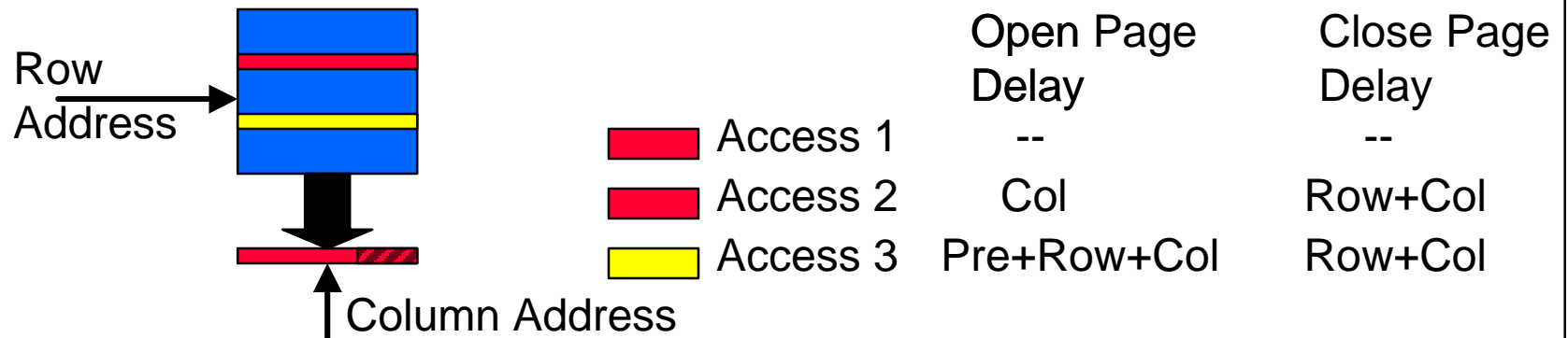
Page Interleaving

- 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

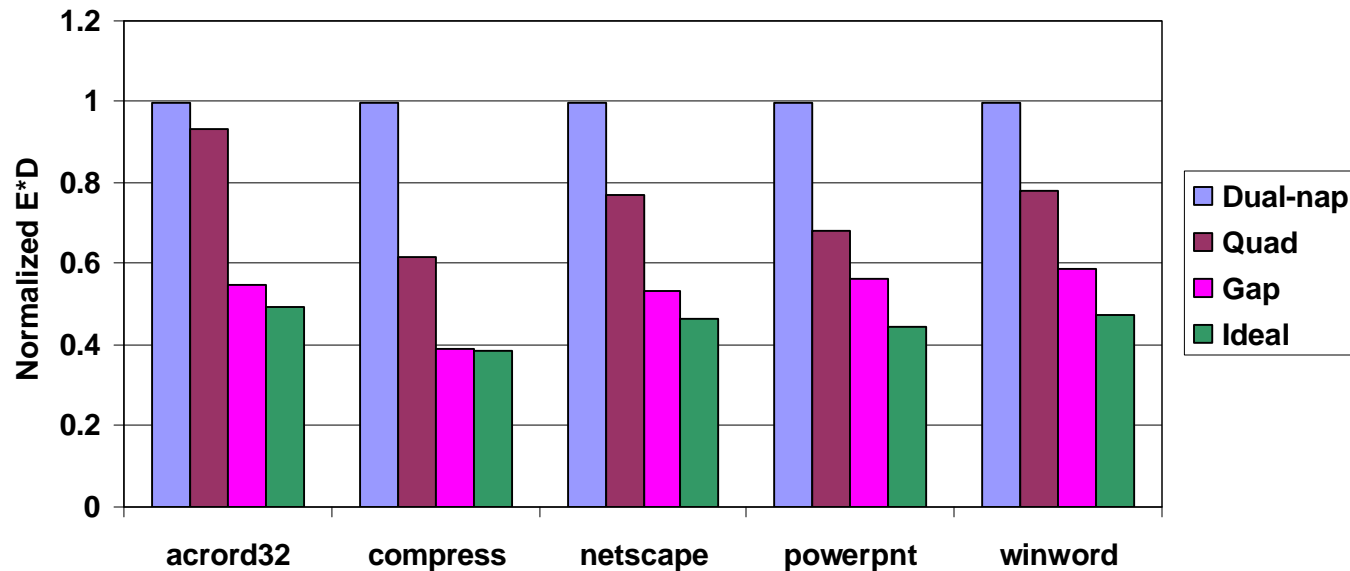
✍ Close page: precharge immediately after access

- 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

Locality Aware Memory Controller Policy



✍ Sequential Page Allocation, NT Traces

✍ Ideal: Offline, Delay = all Active, minimize Power

✍ **Gap: History-based prediction for next access time**

- If gap > benefit boundary, immediately transition
- Stable for random allocation