Near-memory Caching for Energy Management

Nevine AbouGhazaleh, Bruce Childers, Daniel Mosse, Rami Melhem.

Power-Aware Real-Time Systems (PARTS)
Introduction

- Memory-CPU performance gap
  - Caches to mask memory latencies
  - Pentium 2 MB / 4 MB

- Memory energy consumption
  - Servers: 42% of total power [IBMp670]
  - Portable systems: 23% of total power [Celebican’04]

- How to target both problems simultaneously?
Near-CPU vs. Near-memory caches

- Need to balance the allocation of the two.
Background:

**Cached-DRAM (CDRAM)**

- On-memory SRAM cache
  - [Hsu’93, Koganti’97]
  - Ex: Mitsubishi, HP, IBM

- access fast SRAM cache
  - Improves performance.

- High internal bandwidth
  - use large block sizes

- How about energy?
Power-Aware CDRAM (PA-CDRAM)

- Objective: reduce power consumption while maintaining high performance
- DRAM-core
- Near-memory caches
DRAM-core

**Objective:** maximize idle time to transition to low power states for longer periods

- Use moderate sized SRAM cache
- Use immediate powerdown
Near-memory caches

Objective: select the best configuration that balances access delay and energy

- Select delay and power efficient block size

![Graph showing energy consumption vs block size for different configurations.](image)
PA-CDRAM design using RDRAM
Evaluation

- Simulation using Simplescalar & integrated RDRAM memory simulator
- System parameters:
  - Cacti-3.0 for near-memory cache: 256 KB, 512B blocks
  - Rambus Datasheets: 32MB X 8 RDRAM
  - Bus energy using models in [Kadayif’01] & [Aghaghiri’04]
Results: Spec00 Benchmarks

Avg. savings compared to trad. memory hierarchy:
E.D = 28%  ,  Energy = 19%

Avg. savings compared to CDRAM [Koganti’97]:
E.D = 56X  ,  Energy = 46X
Conclusion

- PA-CDRAM reduces the memory’s energy consumption by
  - exploiting the high memory bandwidth
  - Distributing cache
  - Increasing the DRAM-core idle periods

- Near-memory v.s. near-processor caching

- Benefits:
  - Saving in energy-delay product 28% on average
  - Higher savings in high spatial-locality and memory-intensive applications.
For more information:

www.cs.pitt.edu/PARTS