ViPZonE: Exploiting DRAM Power Variability for Energy Savings in Linux x86-64

Mark Gottscho

M.S. Project Report

Advised by Dr. Puneet Gupta

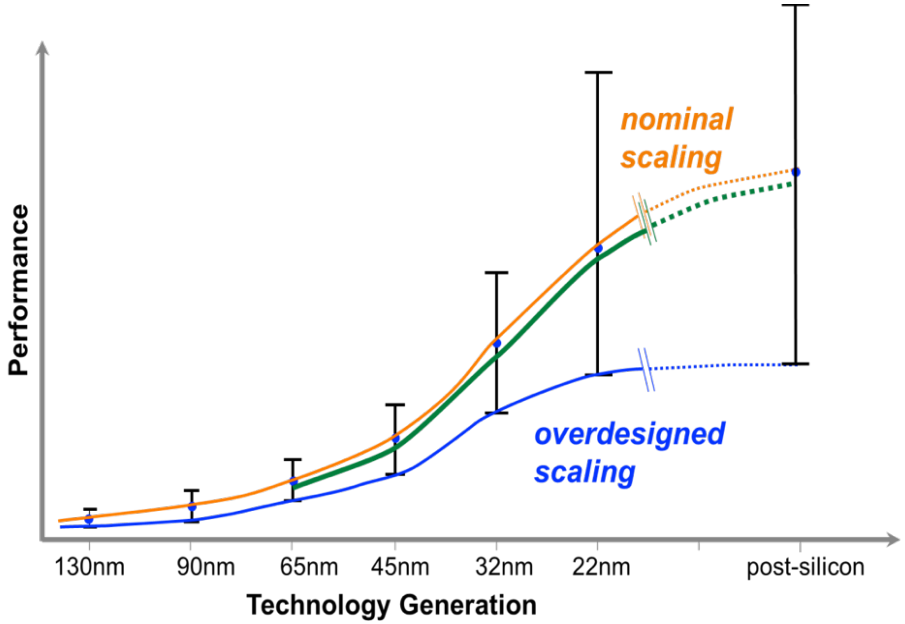
NanoCAD Lab, UCLA Electrical Engineering

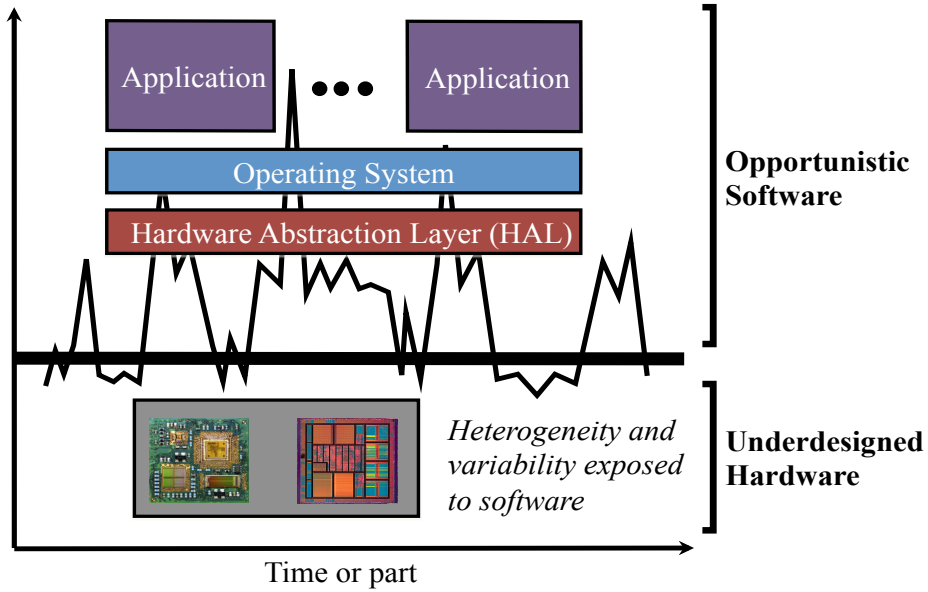
UCI Research Collaborators: Dr. Nikil Dutt, Dr. Alex Nicolau, Dr. Luis A. D. Bathen

March 5, 2014

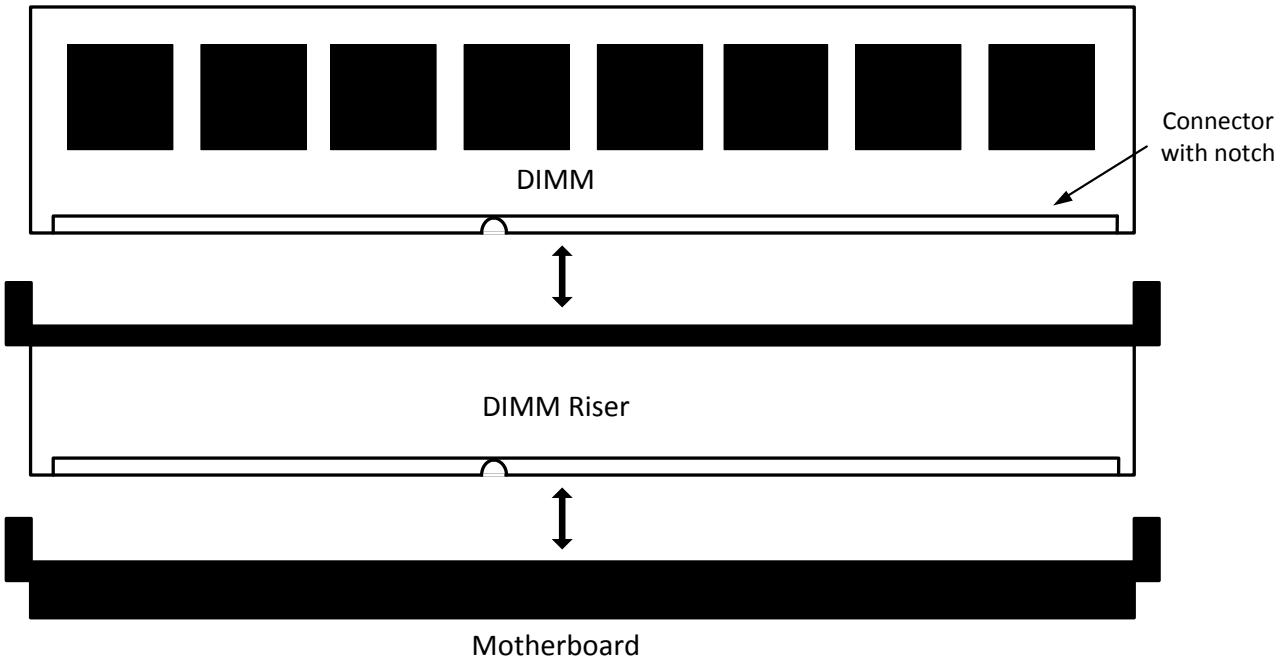
Motivation: Worsening Process Variability

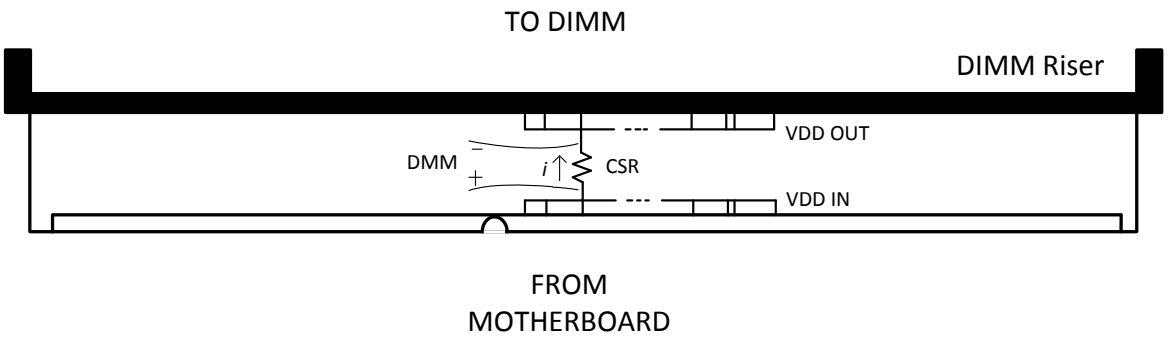
- IC variability is increasing due to technology scaling
- Hardware over-design costs are worsening (performance, power, energy, etc.)
- *Idea:* Flexible hardware/software interface, expose some variability/heterogeneity to software, restore scaling benefits
- **"Underdesigned and Opportunistic Computing (UnO)"**
- **Can memory variations be exploited?**



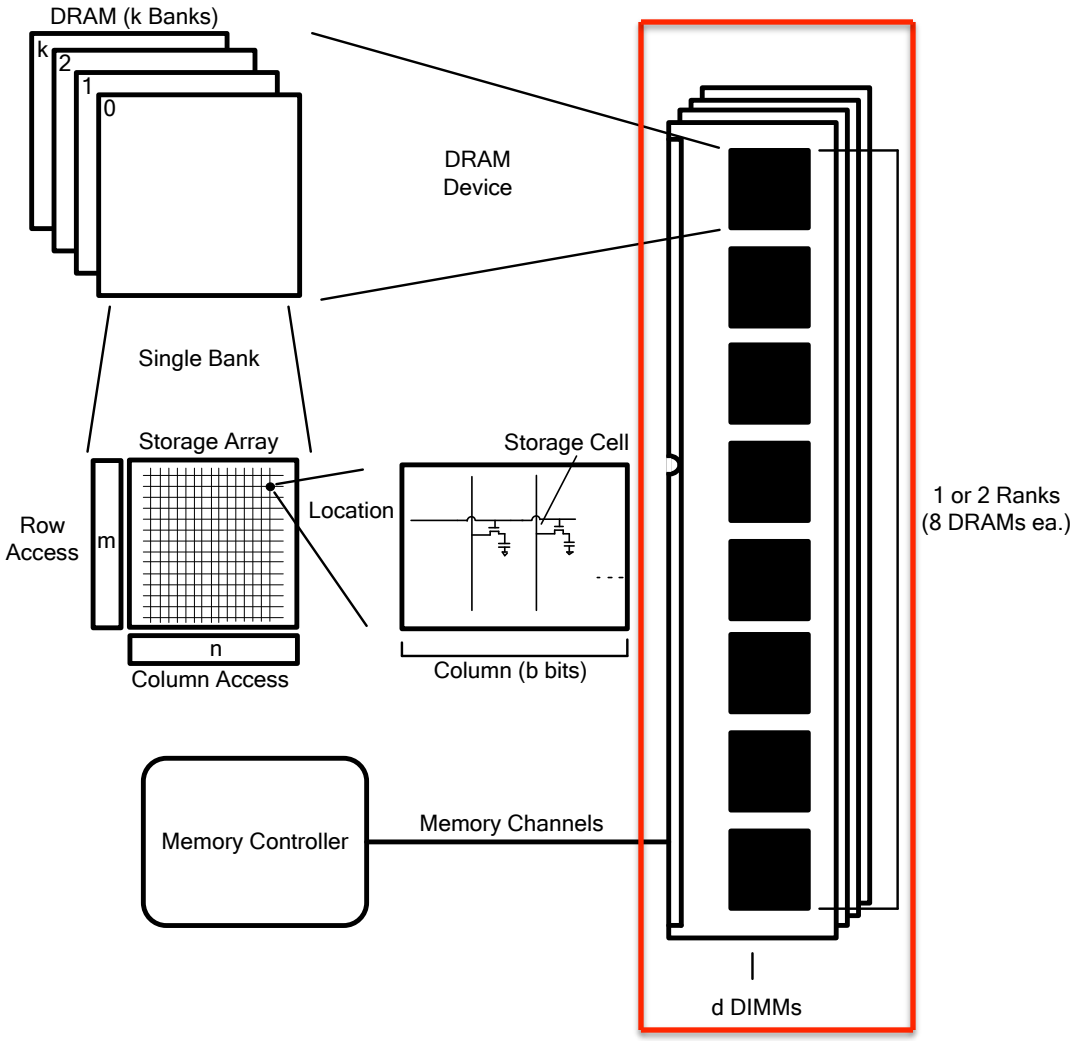


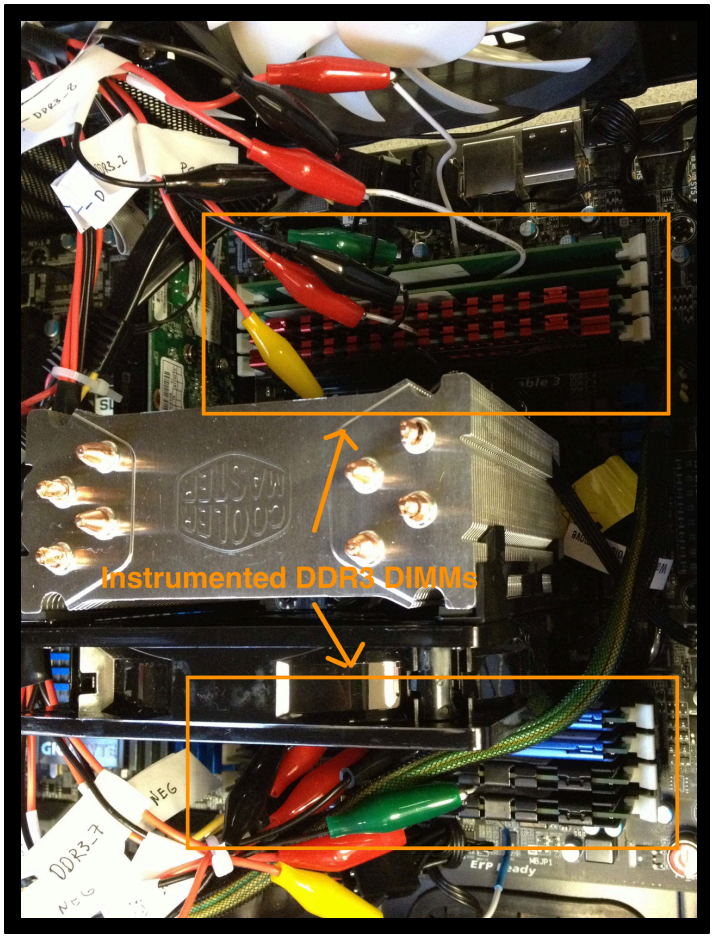
DRAM Power Measurement Setup





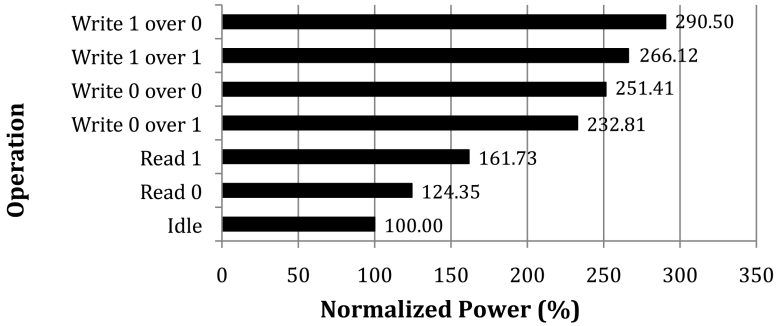
DDR3 Memory System

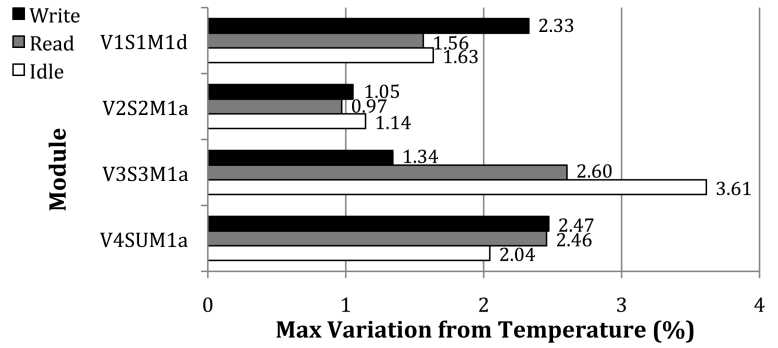


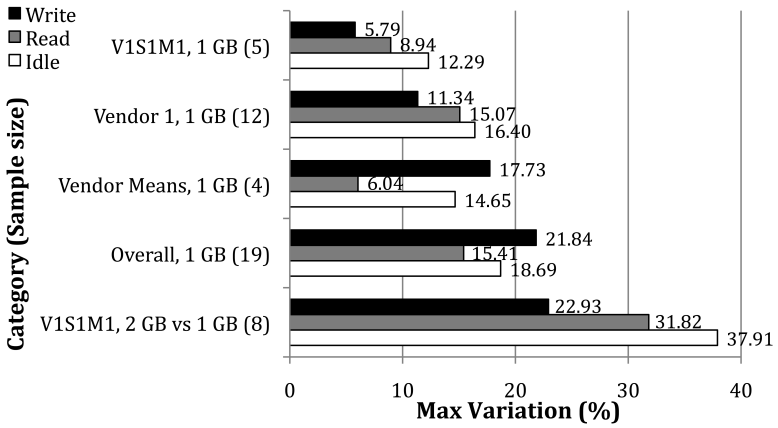


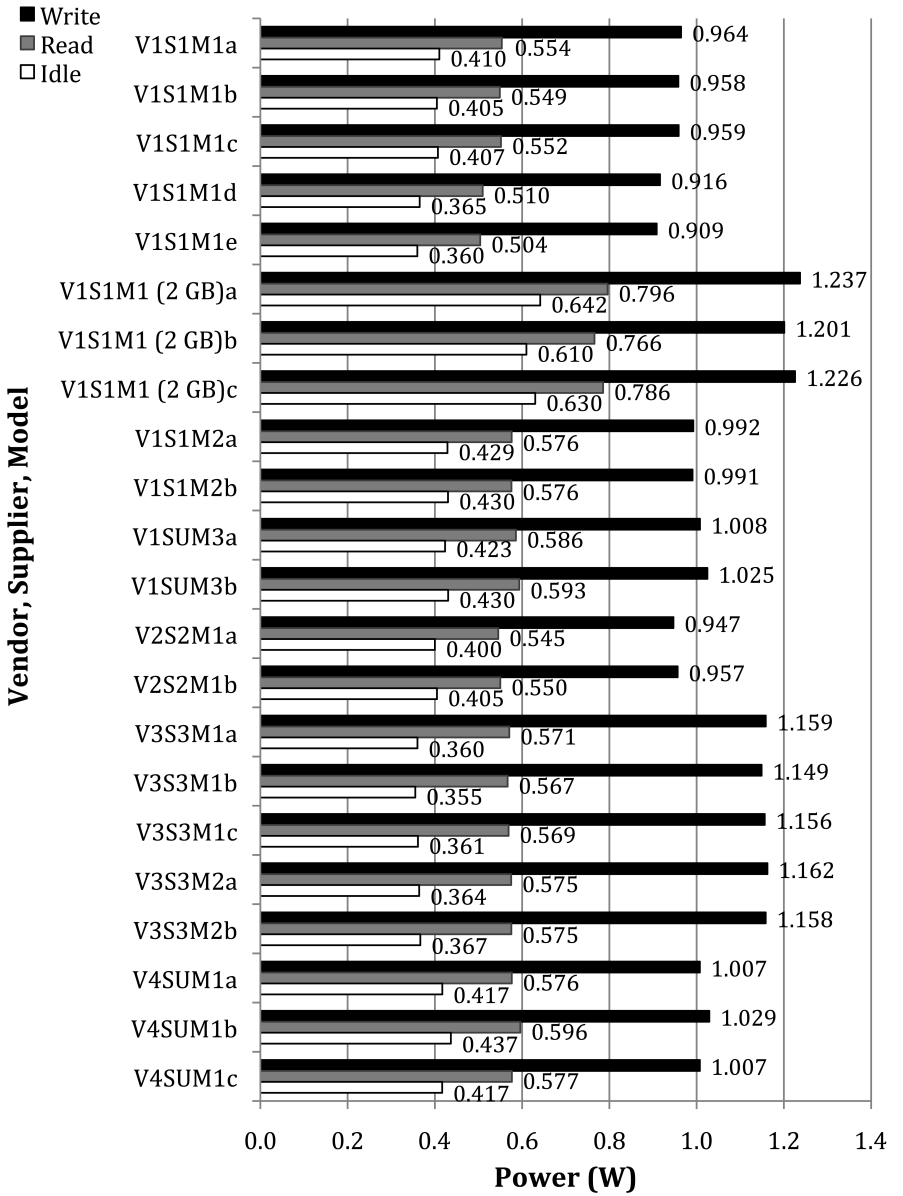
Instrumented DIMMs in the testbed

Measured Power Variability in DRAMs



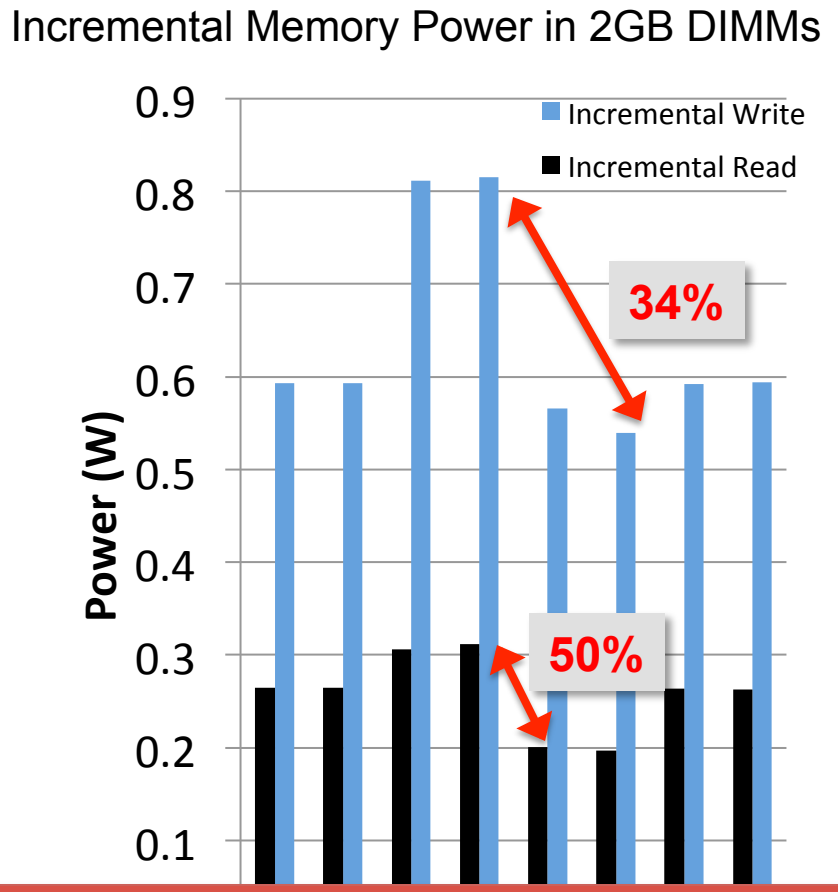






Memory Power Variability is Significant

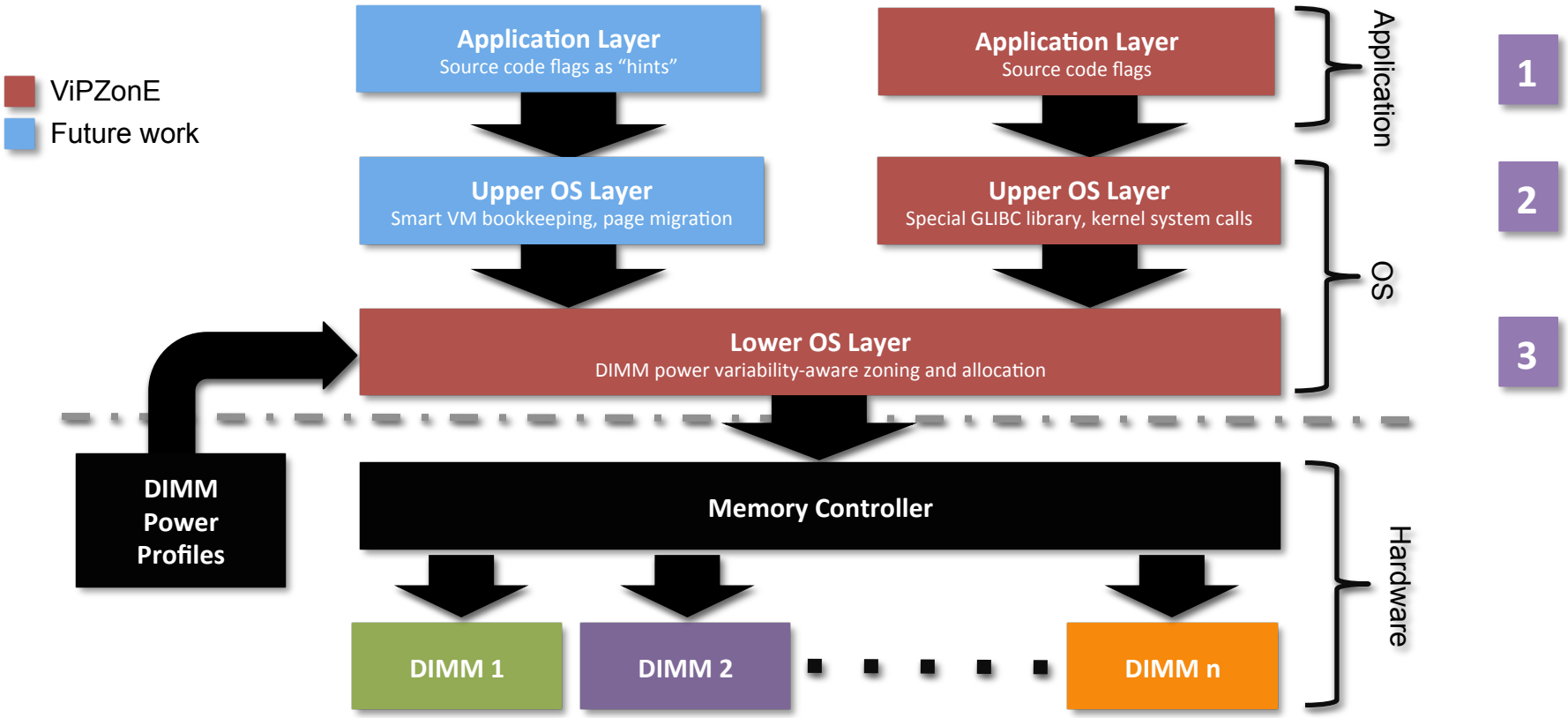
- Significant power variations in DDR3 memory
  - Up to ~20% var. in 19 1GB DIMMs
  - Up to ~55% var. in 8 2GB DIMMs
  - Up to ~50% "usable" variability
    - e.g. incremental power over idle



Systems could save energy by exploiting active memory power variability!

ViPZonE Architecture: App Flags With OS Support

- Apps pass flags to the OS indicating their memory behavior
  - Via special variants of standard system calls
- OS back-end makes power variability-aware allocations using the flags
  - Low-level page allocator uses "zoned" physical address space for each DIMM



Implementation: Linux Kernel, GLIBC on x86-64

1. • Application layer: ViPZonE-enabled apps
   – App flags indicate memory power preferences to the OS
2. • Upper OS layer: modified GLIBC 2.15
   – New *vip_malloc()* function
3. • Lower OS layer: modified Linux 3.2 kernel
   – "Zoning by DIMM"
   – Power variability-aware physical page allocator
   – New *vip_mmap()* syscall

• System hardware assumptions
   – x86-64 architecture (not a requirement for general applicability)
   – DIMM channel & rank interleaving disabled
      • Not necessarily a performance loss, but gives memory power savings
   – Multiple DIMMs with known power consumption

Approach top-down

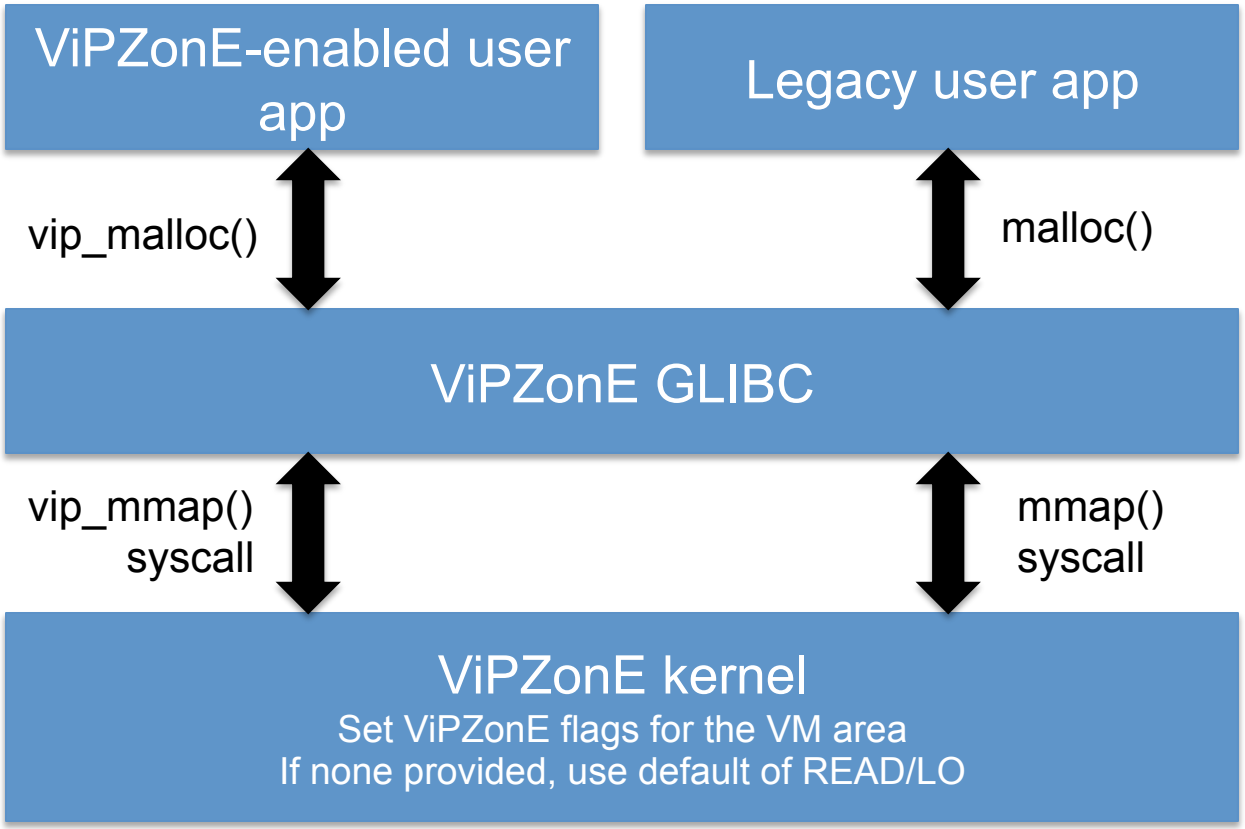
Our approach works on off-the-shelf hardware

1 Implementation: Application Layer

```
#include <stdlib.h> //Special ViPZonE GLIBC with ViPZonE Linux kernel

//…some code…

void foo(size_t arraySize) {
        int *data_ptr = NULL;

        /* Possible vip_malloc() flags:
         * One of: VIP_WRITE or VIP_READ
         * One of: VIP_HIGH_UTIL or VIP_LOW_UTIL
         * Programmer is responsible to decide
         */
        data_ptr = (int *) vip_malloc(sizeof(int)*arraySize,
                VIP_WRITE | VIP_HIGH_UTIL);

        //…some write-heavy operations…
```

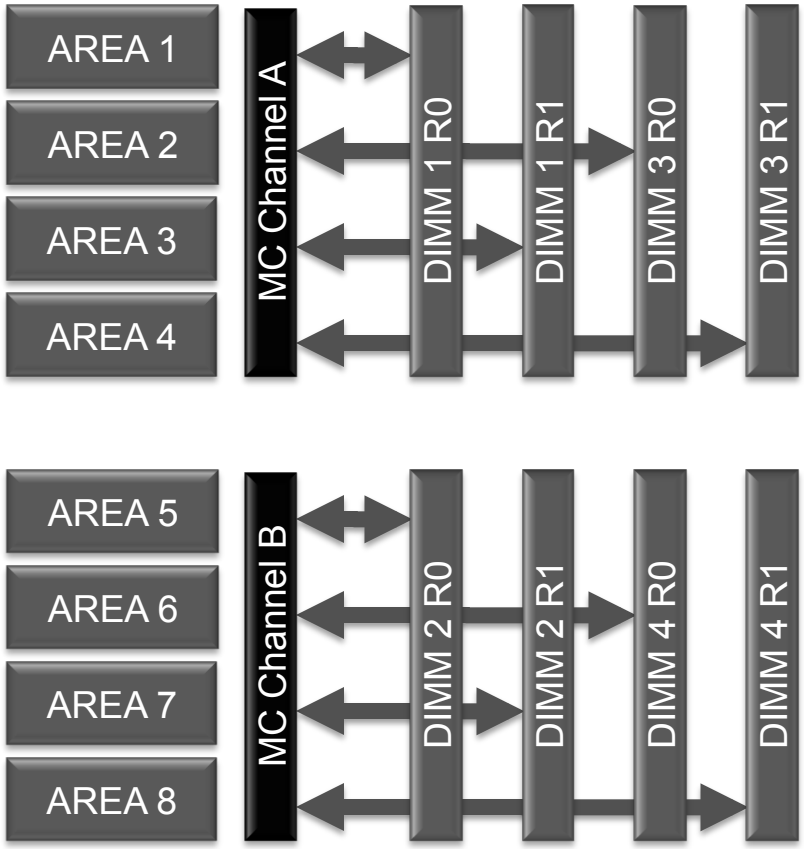
vip_malloc() abstracts power variability in a user-friendly way

2 Implementation: Upper OS Layer



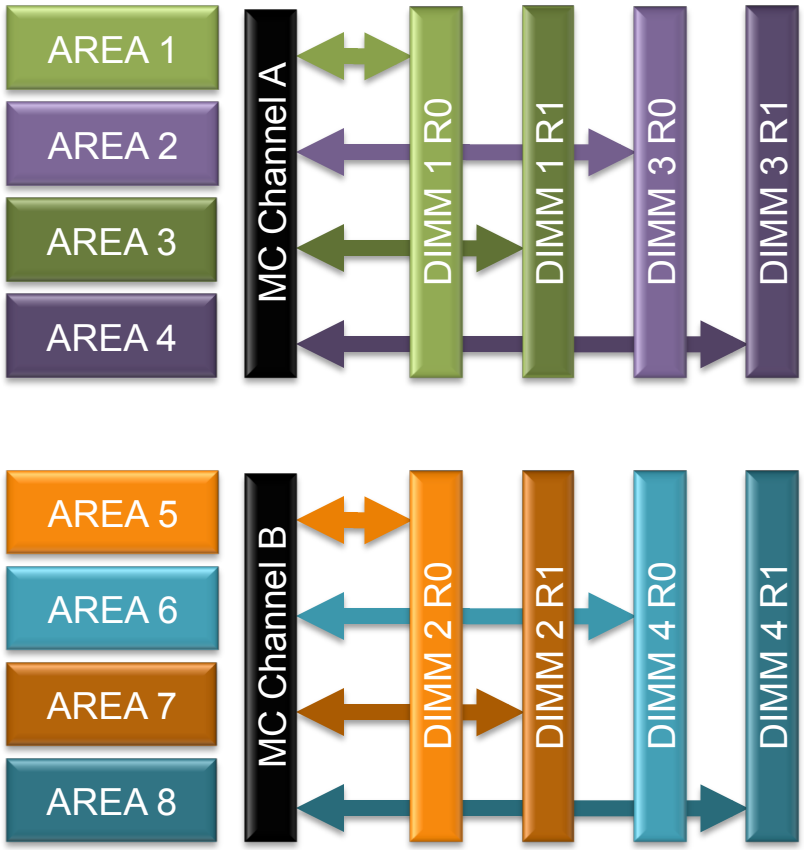
New apps can exploit ViPZonE, legacy apps work the same

Aside: Channel & Rank Interleaving ON



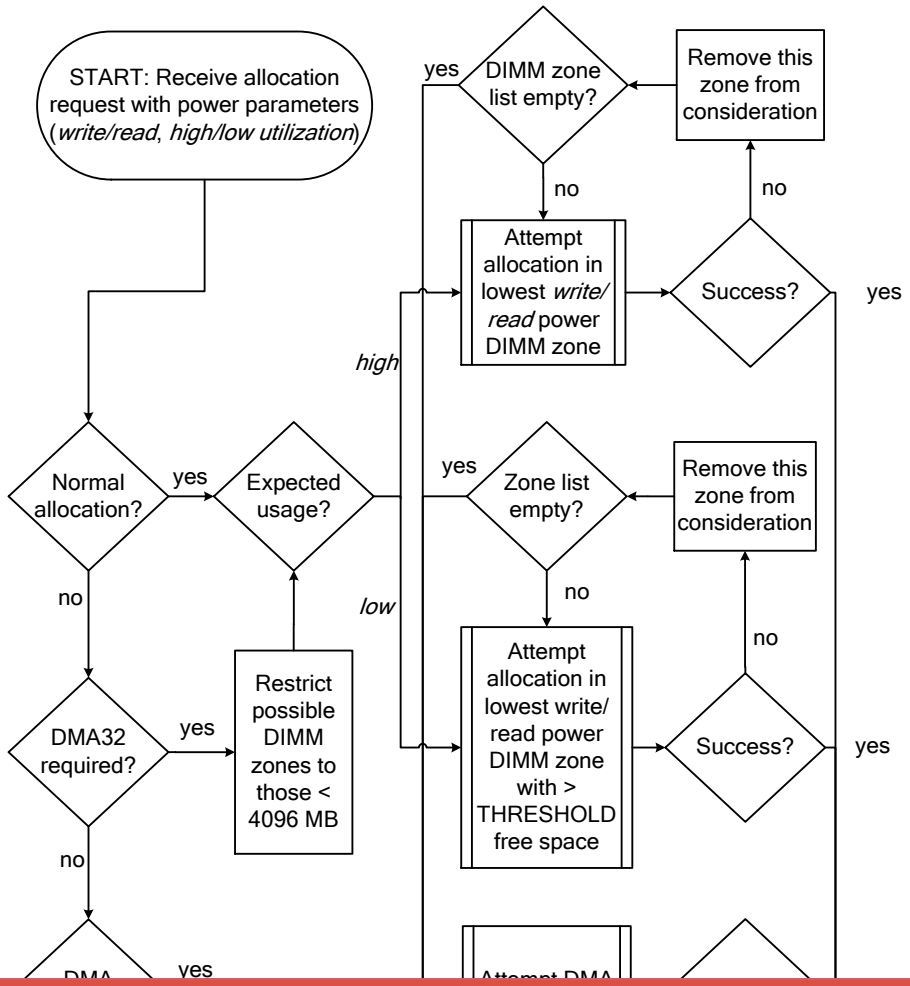
- Using the testbed as example:
  - 2 channels
  - 2 DIMMs per channel
  - 2 ranks per DIMM
  - All rank capacities equal
- Mappings use this scheme:
  - Data striped (interleaved) across all channels, DIMMs, and ranks
  - Stripe size is << size of DIMM or rank

Aside: Channel & Rank Interleaving OFF (ViPZonE)



- No striping
  - If we access only 1 DIMM, the others will be idle (low power)
- If we want benefits of multi-channel bandwidth
  - Need to access different regions of address space in parallel

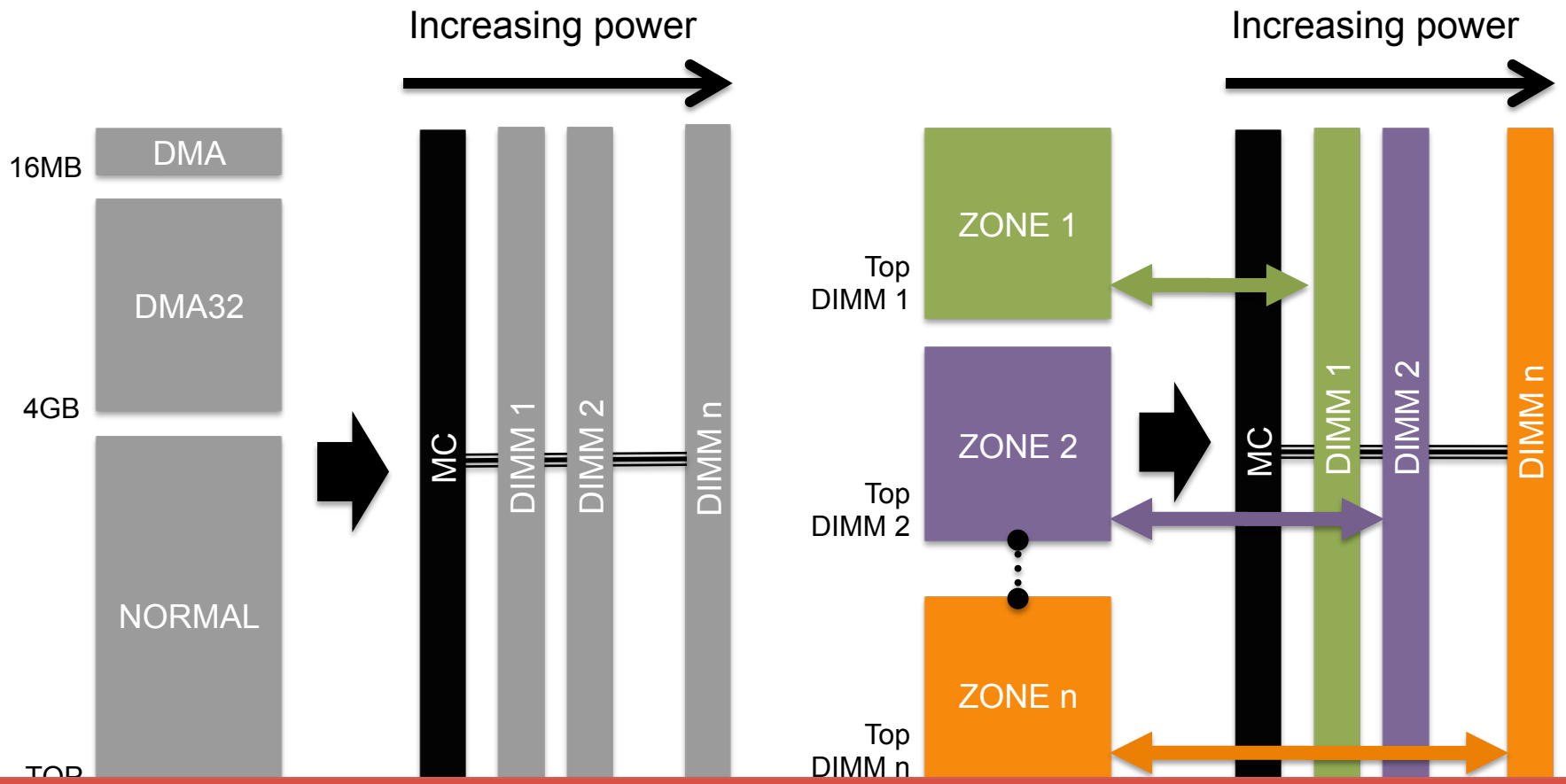
3 Implementation: Lower OS Layer



Physical page allocator

In the kernel, simplicity → high performance

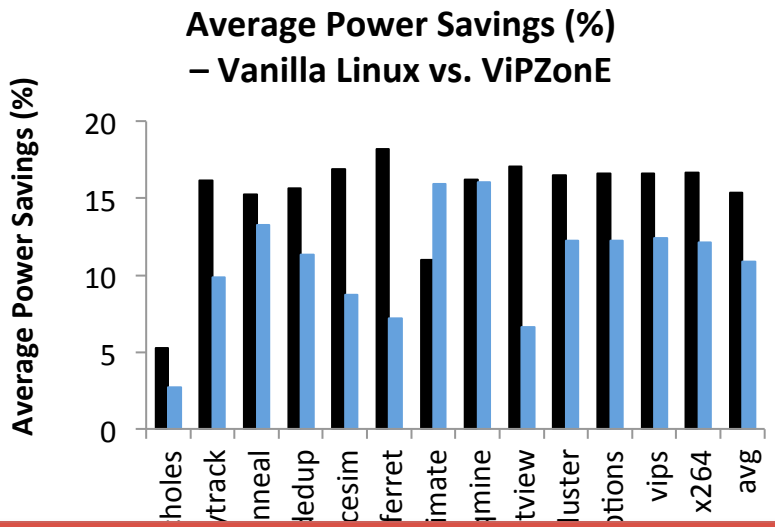
3 Implementation: Lower OS Layer

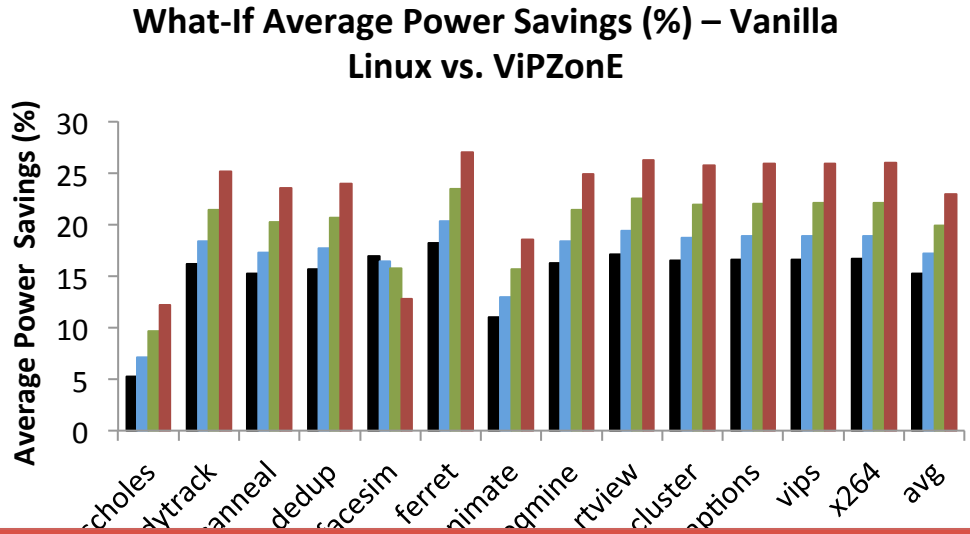


Our zones have different power characteristics by directly mapping them to DIMMs

Simulation Results: Promising Power Savings

- Simulation showed that memory power savings could be up to ~20%
  - Using the 1GB DIMM variability data shown earlier
- Memory power savings could increase to ~30% if DIMM variability increases to 100%
- Performance overhead should be modest

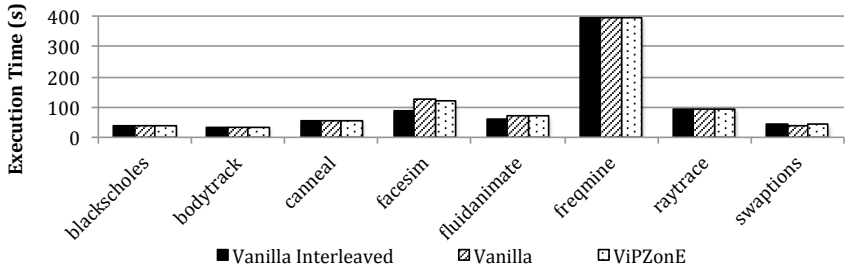


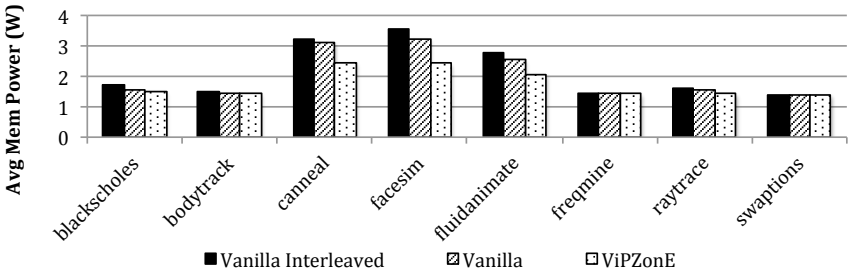


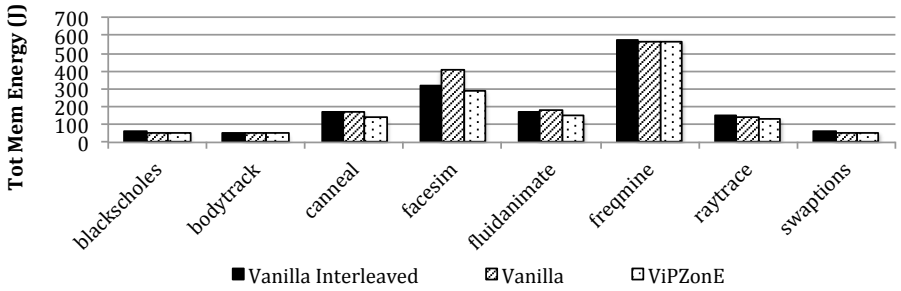
Detailed simulations indicate promising power savings

Measured Testbed Results

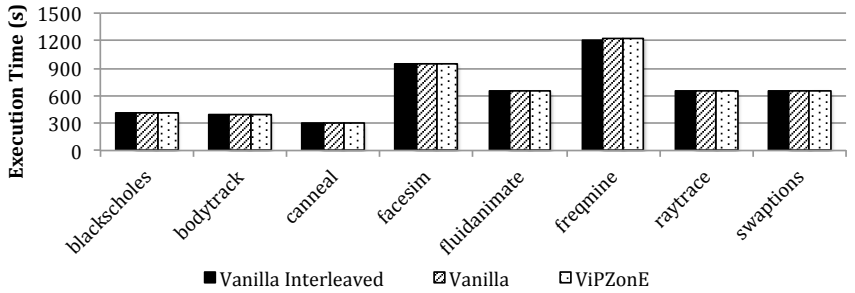
Config. A

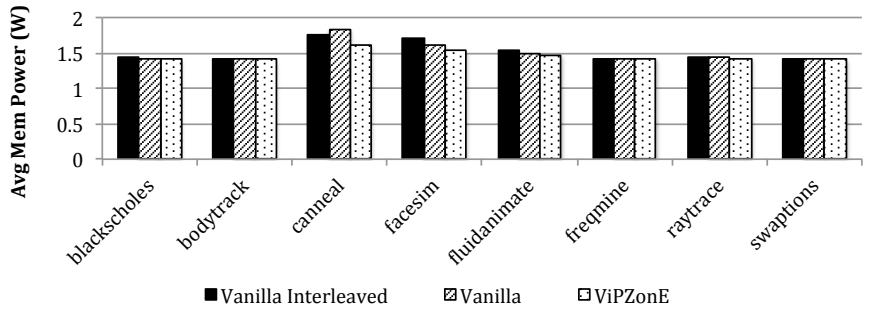


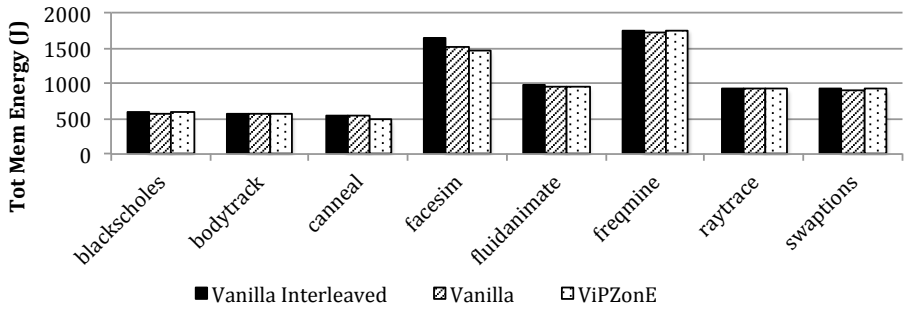




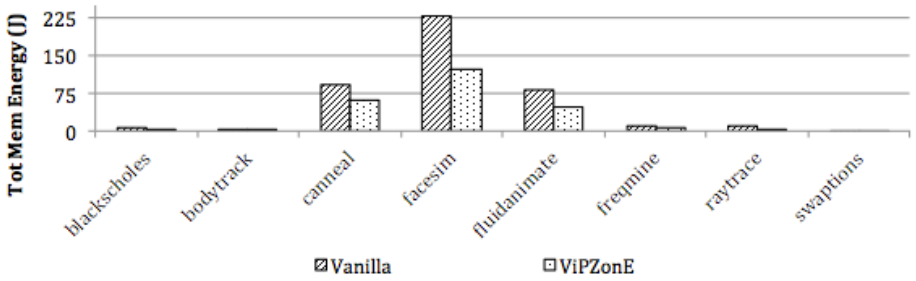
Config. B



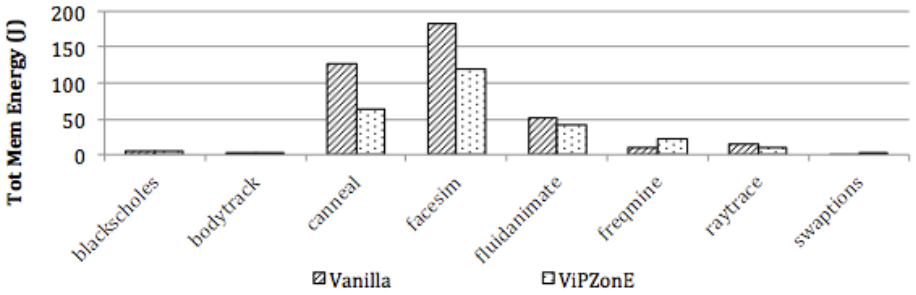




More Measured Testbed Results



(a) *Fast2* Memory Energy, idle energy removed



(b) *Slow2* Memory Energy, idle energy removed

Metric	Value (Benchmark)	Metric	Value (Benchmark)
Max memory power savings, *Fast2* config	25.13% (*facesim*)	Max memory power savings, *Slow2* config	11.79% (*canneal*)
Max execution time overhead, *Fast2* config	4.80% (*canneal*)	Max execution time overhead, *Slow2* config	1.16% (*canneal*)
Max memory energy savings, *Fast2* config	27.80% (*facesim*)	Max memory energy savings, *Slow2* config	10.77% (*canneal*)
Max memory energy savings, *Fast2* config estimated, "NVM" (no idle power)	46.55% (*facesim*)	Max memory energy savings, *Slow2* config estimated, "NVM" (no idle power)	50.69% (*canneal*)

Summary

- ViPZonE can reduce energy consumption in systems with many variable memory devices (e.g., servers and desktops)
- This can be applied in *today's systems* with minimal hardware changes
- ViPZonE enables app-level exploitation of variability with minimal programming changes
- On-going work aims to improve OS sophistication with bookkeeping
  - Kernel could automatically promote hot pages to low power, etc.
  - App flags become "hints" rather than strict guidelines
    - Leaves optimal memory scheduling policy to OS
- Results through simulation & real testbed demonstrate good energy savings
- Underdesigned and Opportunistic Computing is a promising direction for further research