Analyzing the Energy Cost of Data Movement in Scientific Applications

GOKCEN KESTOR, ROBERTO GIOIOSA, DARREN KERBYSON, ADOLFY HOISIE

Pacific Northwest National Laboratory

Richland, WA

Workshop on Modeling & Simulation of Systems and Applications, Seattle, WA

Motivation

- The energy cost of powering a supercomputer is rapidly increasing
  - Will keep increasing in the future → not sustainable
  - Next generation exascale systems should be more power/energy efficient
- Several studies point out that the major energy limiting factor is data movement across memory hierarchy
- No quantitative evaluation of data movement on the energy consumption for scientific applications on current systems
  - Simulation environment:
    - Obtain energy cost per operation
    - Limited-size applications/reduced systems
  - Applications/Systems characterization with external power meter
    - Run full-size application
    - Do not provide the energy cost of moving data

Introduction

What is the amount of energy spent in data movement on current systems?

What is the dominant component of data movement energy?

- We propose a methodology to accurately estimate the energy cost of data movement:
  - Uses highly-tuned micro-benchmarks
  - Follows an incremental-step methodology
  - Derive the energy cost of moving data between any two levels of the memory hierarchy
- We apply our methodology to complex applications and benchmarks (NEKBone, GTC, LULESH and NAS benchmarks)

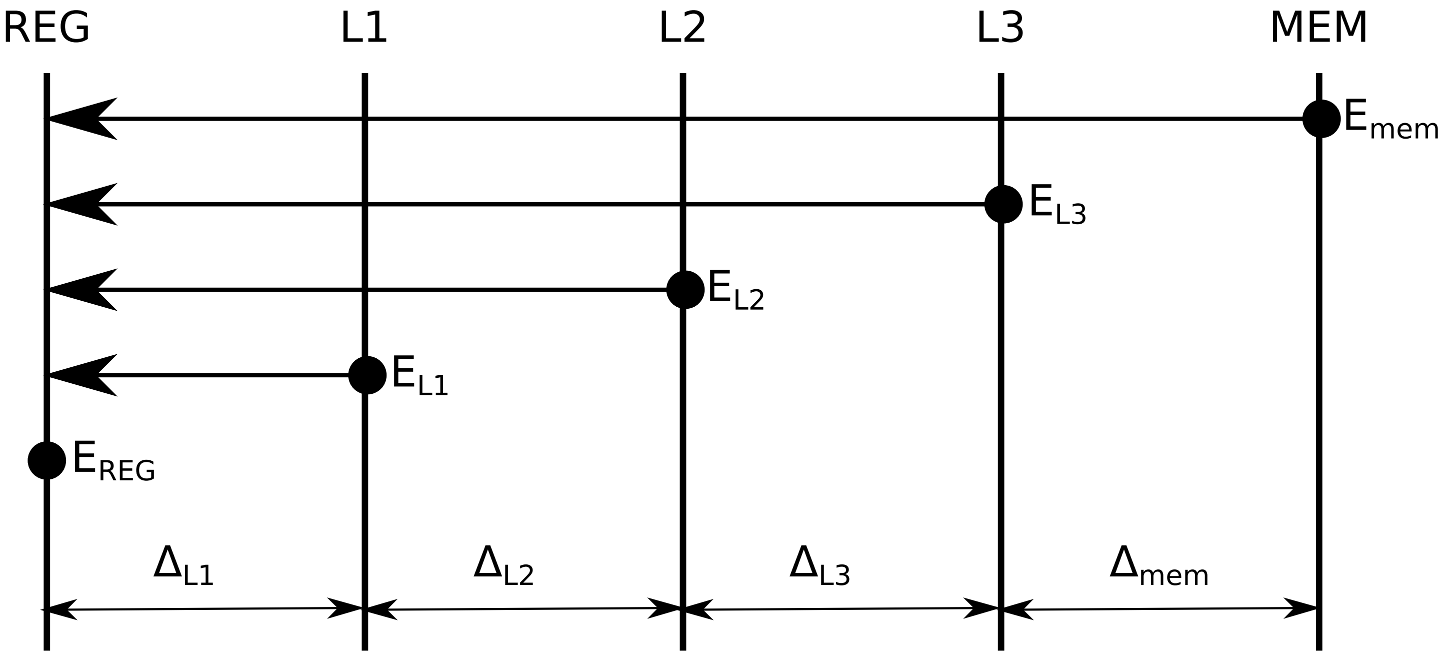
Micro-benchmarks

- Isolating the energy cost of a specific data movement instruction is not trivial due to
  - Out-of-order execution
  - Speculation
  - Memory prefetching, etc.
- We design a new set of well-engineered micro-benchmarks:

MB	L1 miss rate %	L2 miss rate %	L3 miss rate %
MB_{L1}	0.03	0.01	0.01
MB_{L2}	99.96	0.13	0.12
MB_{L3}	99.58	99.47	0.56
MB_{MEM}	99.48	99.48	99.18

```
MB_init();
for (i=0; i<N; i++)
    UNROLL_X{
        <body_loop>
    }
}
MB_finit();
```

Incremental-step methodology



- Measuring the energy cost of single operations:
  - Compute the energy cost of moving data from L1 to processor register
  - Then incrementally determined the energy cost of moving data across the memory hierarchy (ΔE)

How to measure Power?

- We combine the information provided by the internal power sensor and the external power meter:
  - Obtain precise information about socket's power
  - Derive other system components power by difference

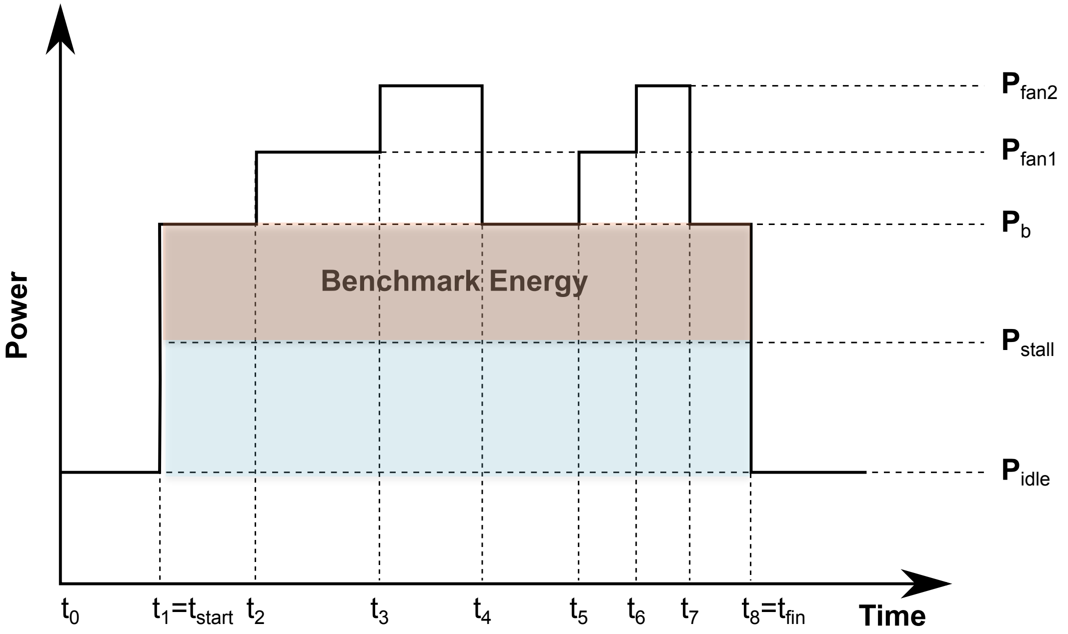
External Power Meter:

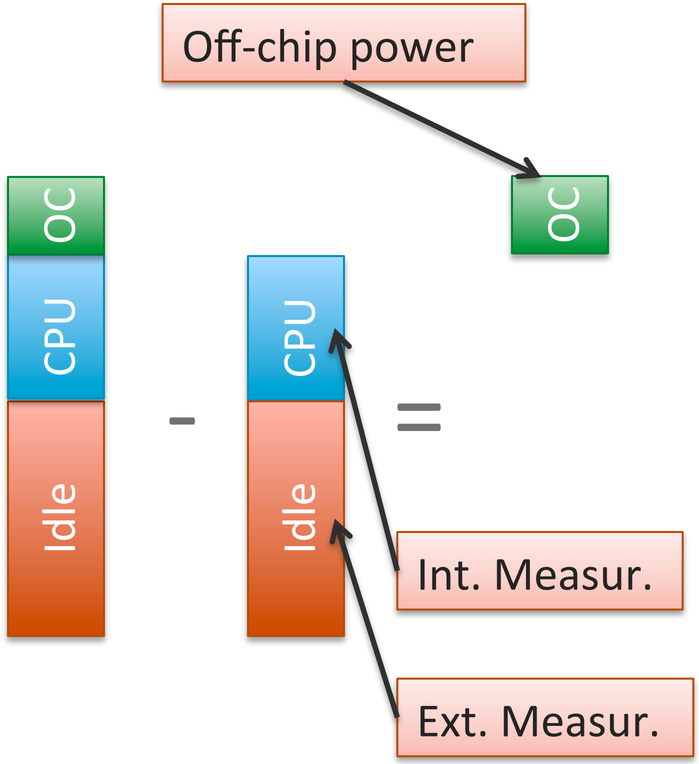
- Measures the power consumption of the entire compute node
- Provides a power sample every 2/3 seconds

Internal Power Sensor:

- Provides power samples at a higher frequency with a higher accuracy
- Measures only the power consumption of the processor chip

Measuring Dynamic Energy





Off-chip power

OC

CPU

Idle

–

CPU

Idle

=

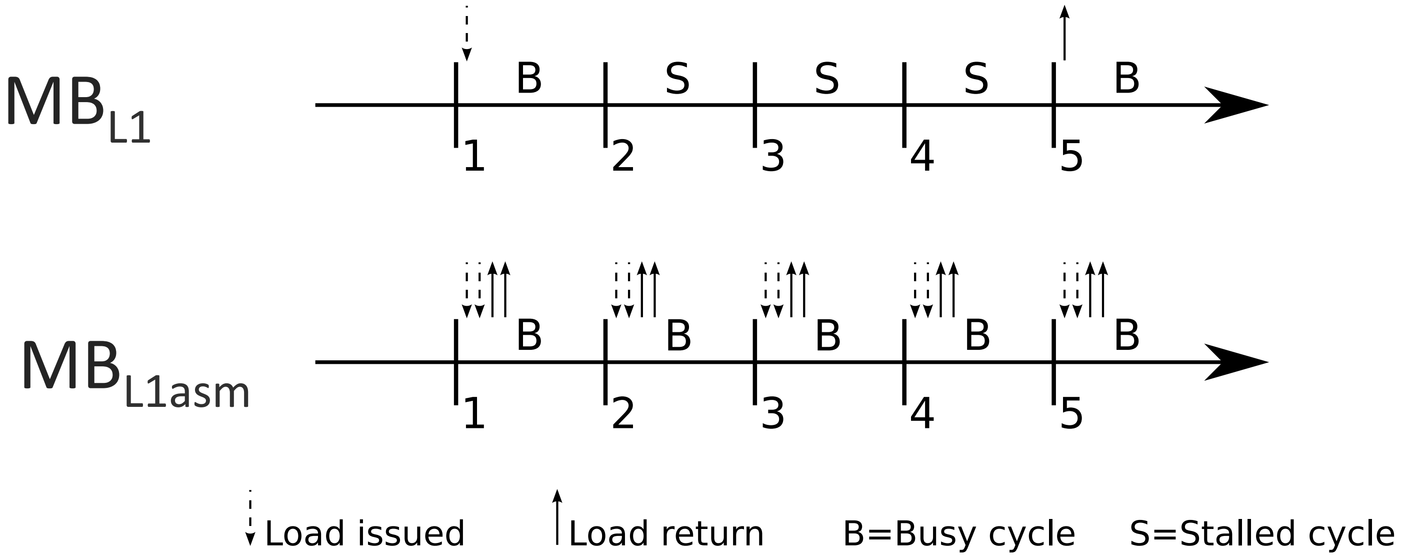
OC

Int. Measur.

Ext. Measur.

- Measure node idle power P_{idle}
- Isolate the dynamic power consumption of off-chip components
  - Processor fans (two speeds -> P_{fan1}, P_{fan2})
  - Memory
- Accurately compute the energy of each micro-benchmark

Energy Cost of Stalled Cycles



- While stalled, processor cores still consume power (P_{stall})
  - Resolve data dependencies, detect memory access patterns, etc.
  - Should not be included in the cost of moving data
- We wrote an alternative version of MB$_{L1}$ that fully utilizes the pipeline:
  - MB$_{L1asm}$ presents no dependencies → no stall cycles
  - We can derive E_{L1} from MB$_{L1asm}$
  - Using MB$_{L1}$ and MB$_{L1asm}$ we derive E_{stall}

Long Latency Memory Operations

- $MB_{L2,}$ MB_{L3}, MB_{MEM} perform long latency memory operations

L1 latency:	4 cycles
L2 latency:	20 cycles
L3 latency:	60 cycles
Memory latency:	150 cycles

- Impossible to implement a version of these micro-benchmarks with no stall cycles:
  - Load-store queue becomes full
  - Core stalls while waiting for the data

- Subtract E_{stall} from the energy consumed by the micro-benchmark

$$E_{L2} = \frac{E_{MB\_L2} - E_{stall} * N_{stall}}{N_{L2}}$$

Data movement by Prefetchers

- To estimate energy cost of data prefetching, we implemented an alternative version of MB_{MEM}:
  - MB_{MEM}: Stride size 512, no data prefetching
  - MB_{MEM64}: Stride size 64, perfect data prefetching

- AMD Interlagos 6227 provides two specific performance counters for data prefetching requests (L1 and L2 prefetcher)

MB	L1 miss rate %	L2 miss rate %	L3 miss rate %	L1 prefetcher %	L2 prefetcher %
MB_{MEM}	99.48	99.48	99.18	0.15	0.04
MB_{MEM64}	99.33	99.94	2.15	97.75	97.60

Summary of Energy Costs

Operation	Operation Energy Cost (nJ)	Equivalent ADD
ADD	0.64	-
L1->REG	1.11	1.8x
L2->REG	2.21	3.5x
L3->REG	9.80	15.4x
MEM->REG	63.64	99.7x
Stall	1.43	-
Prefetching	65.08	-

Data Movement	Data movement Energy (nJ)
-	-
L1->REG	1.11
L2->L1	1.10
L3->L2	7.59
MEM->L3	53.84
-	-
MEM->cache	65.08

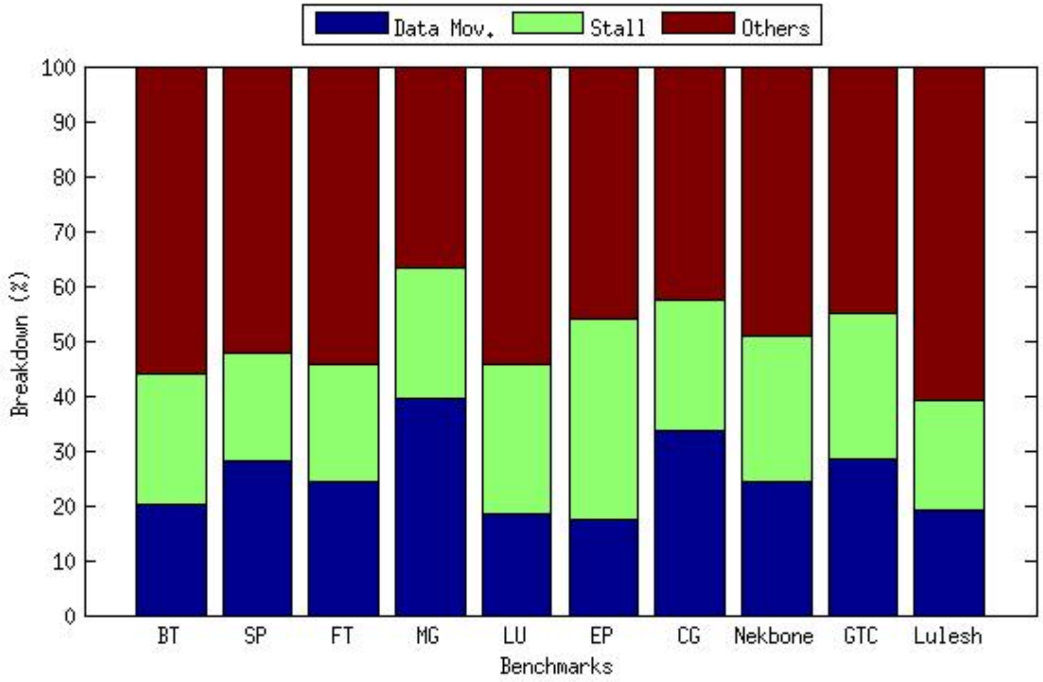
$$E_{DM} = \sum_i \Delta E_i * N_i$$

i = L1, L2, L3, MEM, PRE

ΔE_i = Energy of moving data from *i* to *i-1*

N_i = Number of events

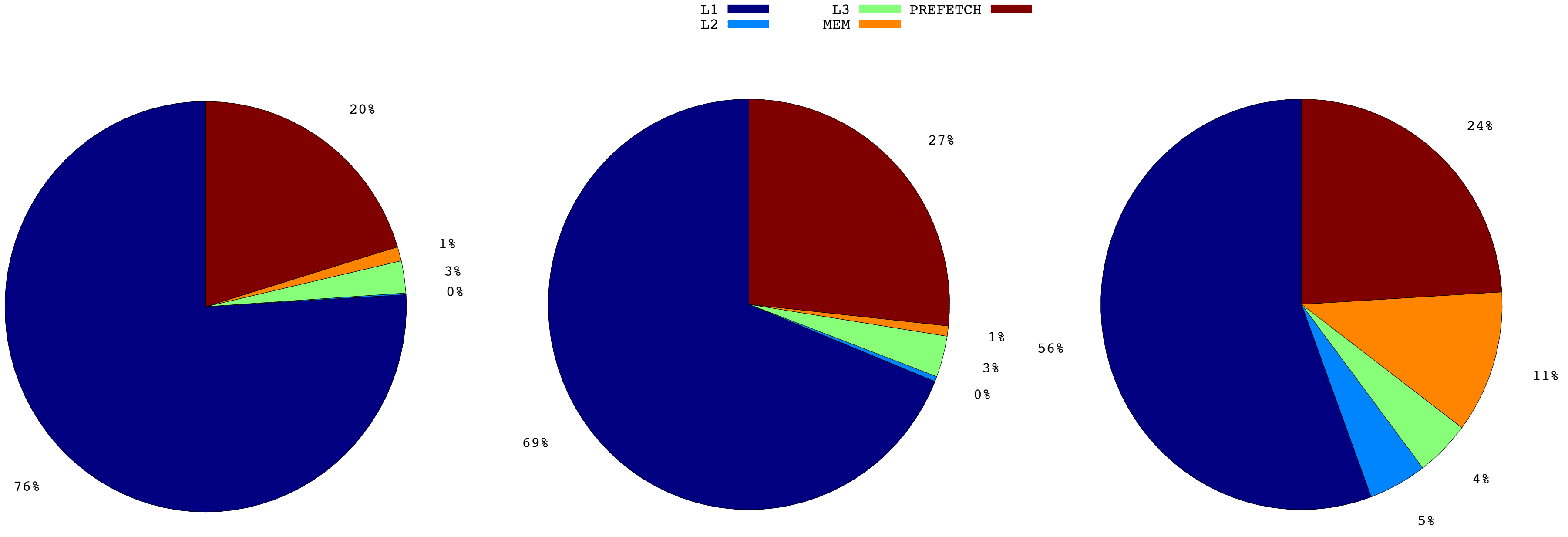
Energy Breakdown



- Others: computing operations, fans, circuitry, etc.
- Total dynamic energy measured from external power meter.
- Stall and Data Movement estimated by our model.

- Various percentage of energy consumed in data movement: 18% (EP) and 40% (MG), 25% on average
- 19-36% of total dynamic energy spent in stall cycles
  - Motivates simpler architectural design

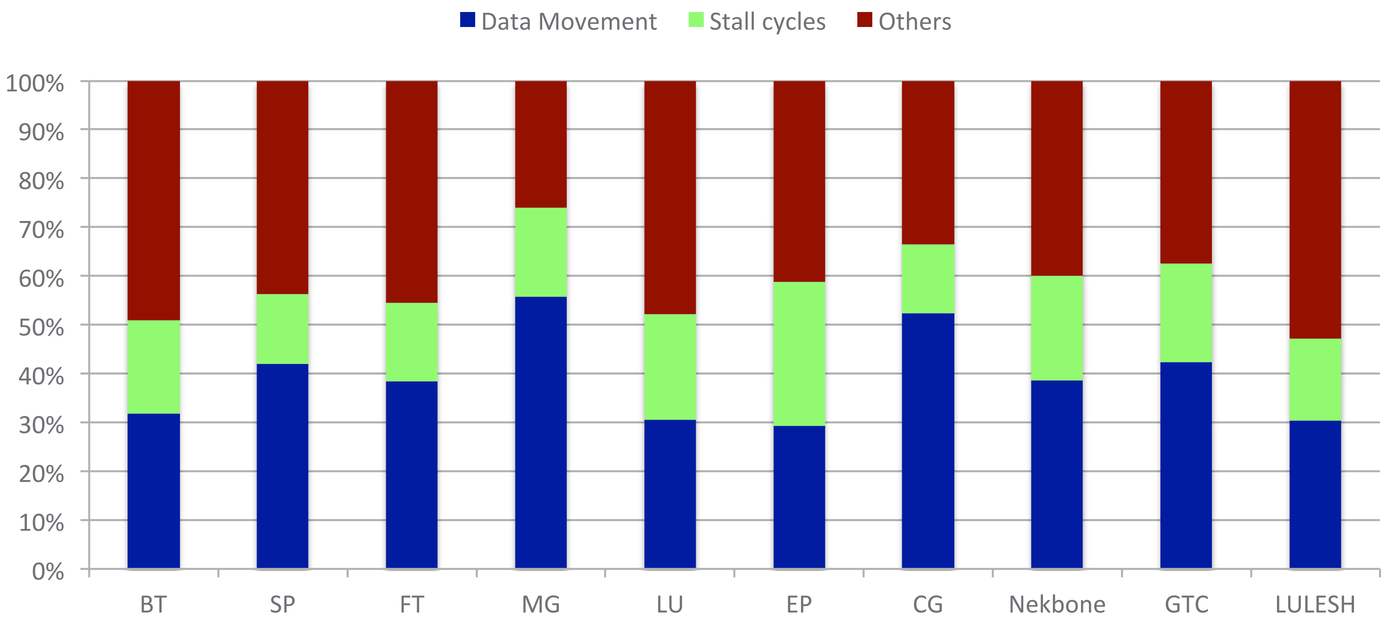
Energy spent into moving data



NEKBone GTC LULESH

- NEKBone and GTC have been optimized ➔ excellent locality
- LULESH:
  - Good locality
  - Prefetchers move most of the data
  - Still needs to bring more data from memory

Let’s assume…



- Energy cost of computation reduces to 1/10
- Energy cost of data movement remains roughly the same
- Processor architectures become simpler and more energy efficient (1/2 stall cycle energy)
- Energy cost of other, non-processor components (fans, circuitry, etc.) remains roughly the same

Conclusions

- Data movement is the key challenge on the road to Exascale

- We accurately estimate the energy cost of moving data across memory hierarchy:
  - Uses a set of highly-tuned micro-benchmarks
  - Follows an incremental-step approach

- Our analysis for scientific applications on current systems:
  - Significant amount energy spent to move data across memory hierarchy, 25% on average
    - Data movement should be reduced in future systems
  - Energy spent in stall cycles is noticeable, 19%-36%
    - Guides simpler architecture design
  - Memory prefetchers also contributes in data movement
    - More precise data prefetcher to avoid prefetching unnecessary data

ModSim Questions

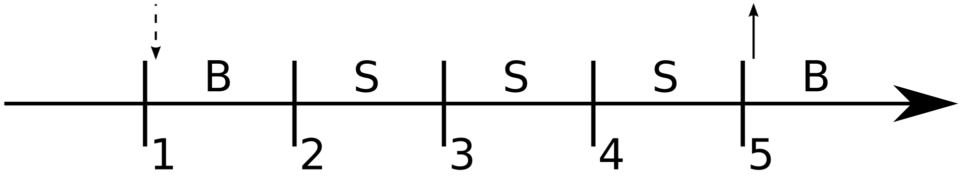
- what is the major contribution of your research?
  - Methodology to evaluate the energy cost of data movement
  - Estimation of the energy cost of data movement in scientific applications running on current systems
- what are the gaps you identify in the research coverage in your area?
  - More precise sensors to measure components' power
  - Better interaction with computer architects
  - Assumption on future architecture components and their energy cost is not clear
- what is the bigger picture for your research area?
  - Data movement
  - Energy efficiency

Thanks!

Backup slides

Empirical Evaluation of Stalled Cycle Energy

- MB$_{L1}$ operations include stalled cycles:



- MB$_{L1}$ operations consume more energy than MB$_{L1asm}$ operations

$$\frac{E\_MB_{L1}}{N_{L1}} > \frac{E\_MB_{L1asm}}{N_{L1}} = E_{L1}$$

- The energy consumption of MB$_{L1}$ is E_MB$_{L1}$ (measured):

$$E\_MB_{L1} = E_{L1} * N_{L1} + E_W * N_{L1} + E_{stall} * N_{stall}$$
$$\cong k * E_{L1} * N_{L1} + E_{stall} * N_{stall}$$

Empirical Evaluation of Stalled Cycle Energy (cont.)

$$E_{stall} = \frac{E\_MB_{L1} - k * E_{L1} * N_{L1}}{N_{stall}}$$

- Where:
  - N_{L1} = number of loads operations issued by MB_{L1}
  - EL1 = energy cost of a load that moves data from L1 (estimated from MB_{L1asm}
  - k = empirical factor that account for the wasted energy E_W.
    - $1 < k <= 2$ A load is issued, 2 is the maximum number of loads/cycle
    - $k < 2$ data does not move from the L1 to register

- We evaluated k in terms of “missing opportunities (loads)”
  - For simplicity, assume that the energy is evenly spread in 4 cycles
  - In 4 cycles there could be 7 loads (8 max–1 not issued) ➔1.75 ld/cycle
- k = 1.75 is an empirical value based on reasoning
  - Compare IPC: 2 (MB_{L1asm}), 0.75 (MB_{L1})
  - Missed load energy = 75% of regular load (energy not consumed when issuing/retrieving data from L1)

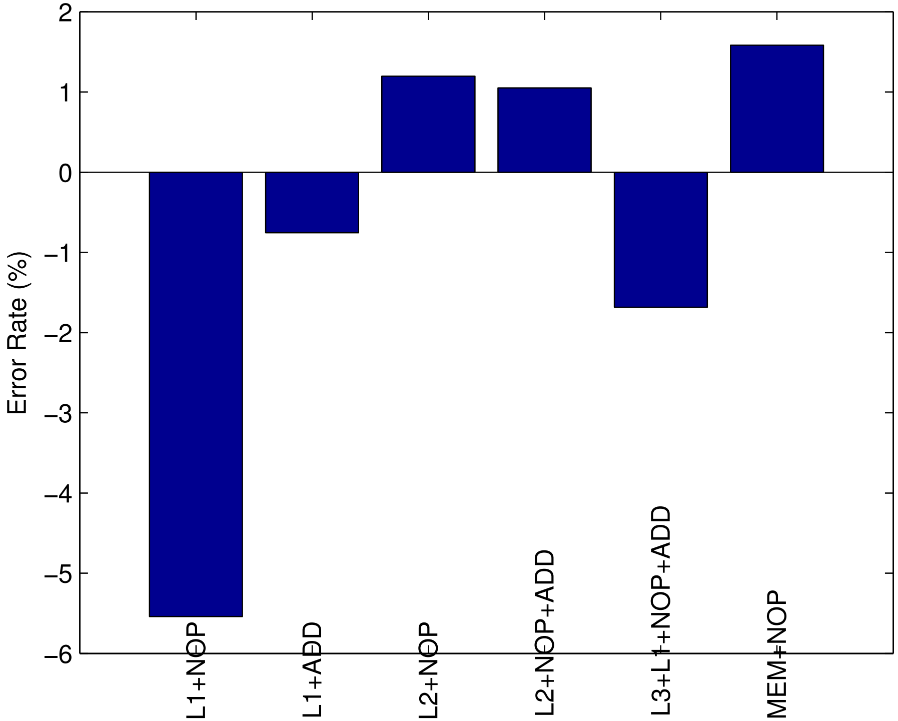
Memory Prefetchers

- Processors proactively prefetch/move data from memory to the processor caches to hide latency/improve performance

- This data movement is
  - Not initiated by a programmer
  - Not reflected in the number of load operations or cache misses

- AMD Interlagos 6227 processor features two prefetchers
  - **L1 prefetcher:**
    - activated by L1 cache misses
    - brings data from memory to the L1 cache
  - **L2 prefetcher:**
    - Reacts to L2 cache misses
    - Coordinates with L1 prefetcher
    - Brings data from memory to the L2 cache

Model Validation

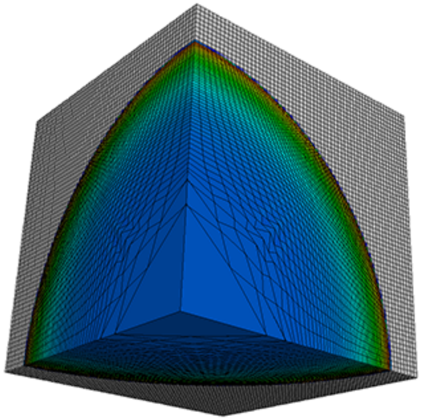


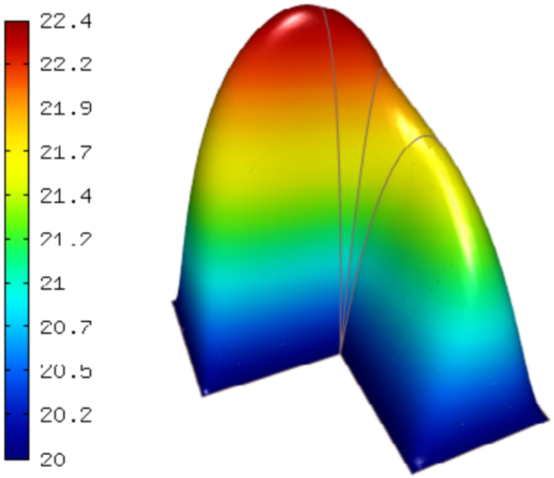
- Validation benchmarks:
  - Combine different operations in the body loop
  - Data movement + computing operations
- Compute the error rate between the estimated energy and the energy obtain from external measurement

$$E_{L1+NOP} = E_{L1} * N_{L1} + E_{NOP} * N_{NOP} + E_{stall} * N_{stall}$$

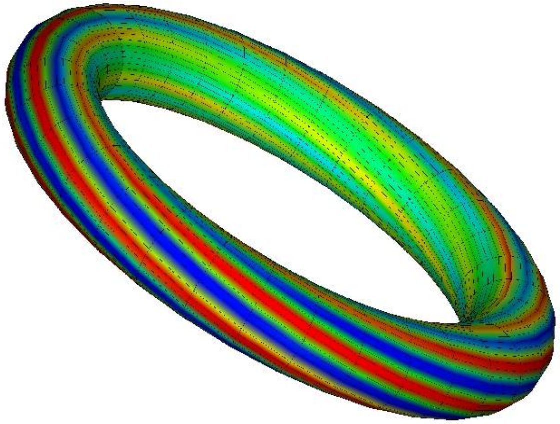
Scientific Applications

- **LULESH:**
  - DOE Co-design center application
  - the Shock Hydrodynamics Challenge Problem
  - solves Sedov blast problem



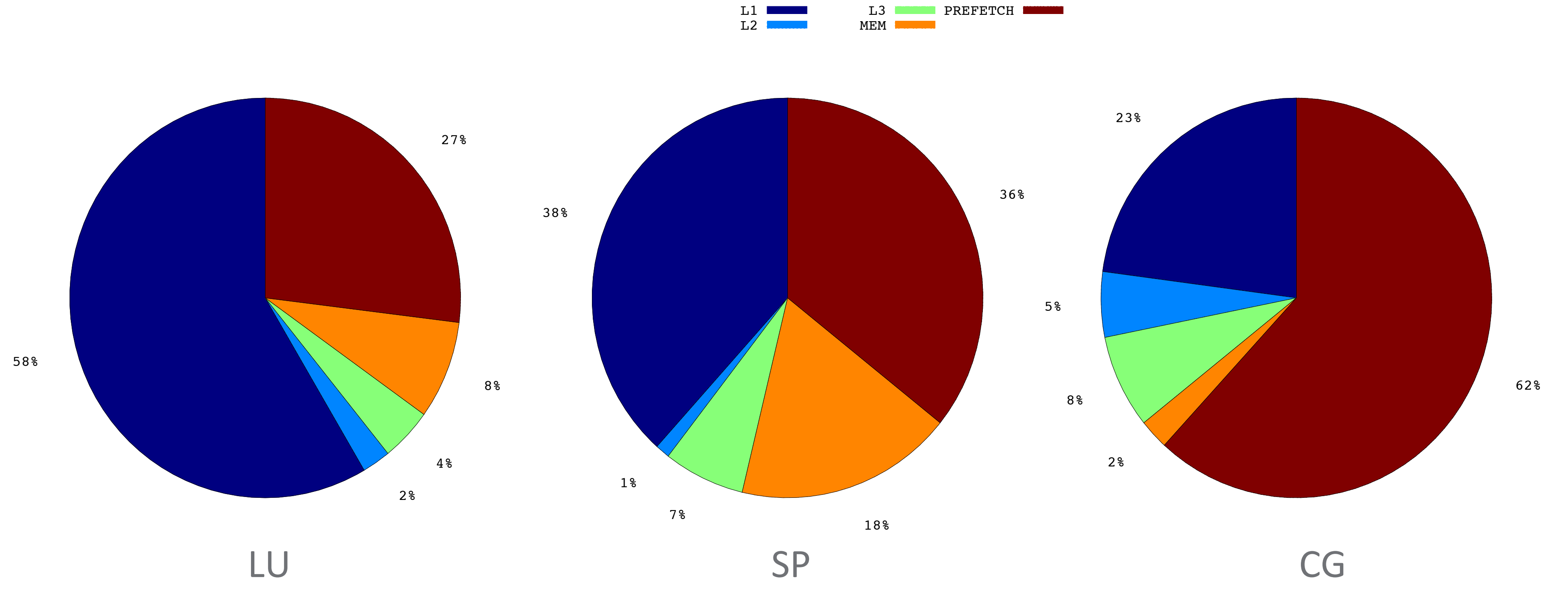


- **Nekbone:**
  - CESAR Co-design center application
  - Proxy application of NEK5000
  - solves Poisson equation using a conjugate gradient



- **GTC:**
  - DOE Office of Science application
  - 3-dimensional code
  - studies microturbulence in magnetically confine toroidal fusion plasmas

Energy spent into moving data



- All data moved to registers must come from the L1
  - ΔE_{L1} is dominant for benchmarks with good locality (LU)
- Memory prefetchers:
  - Capable of capturing access patterns → ΔE_{MEM} low (LU)
  - If not, still need to move data from memory → ΔE_{MEM} high (SP)
  - May waste energy prefetching useless data → $\Delta E_{PRE} >> \Delta E_{L1}$ (CG)