Evaluation of the Jetson TK1 Development Board for Power and Performance

Jee Choi

Mar. 17, 2015

Presented at GPU Technology Conference (GTC) 2015

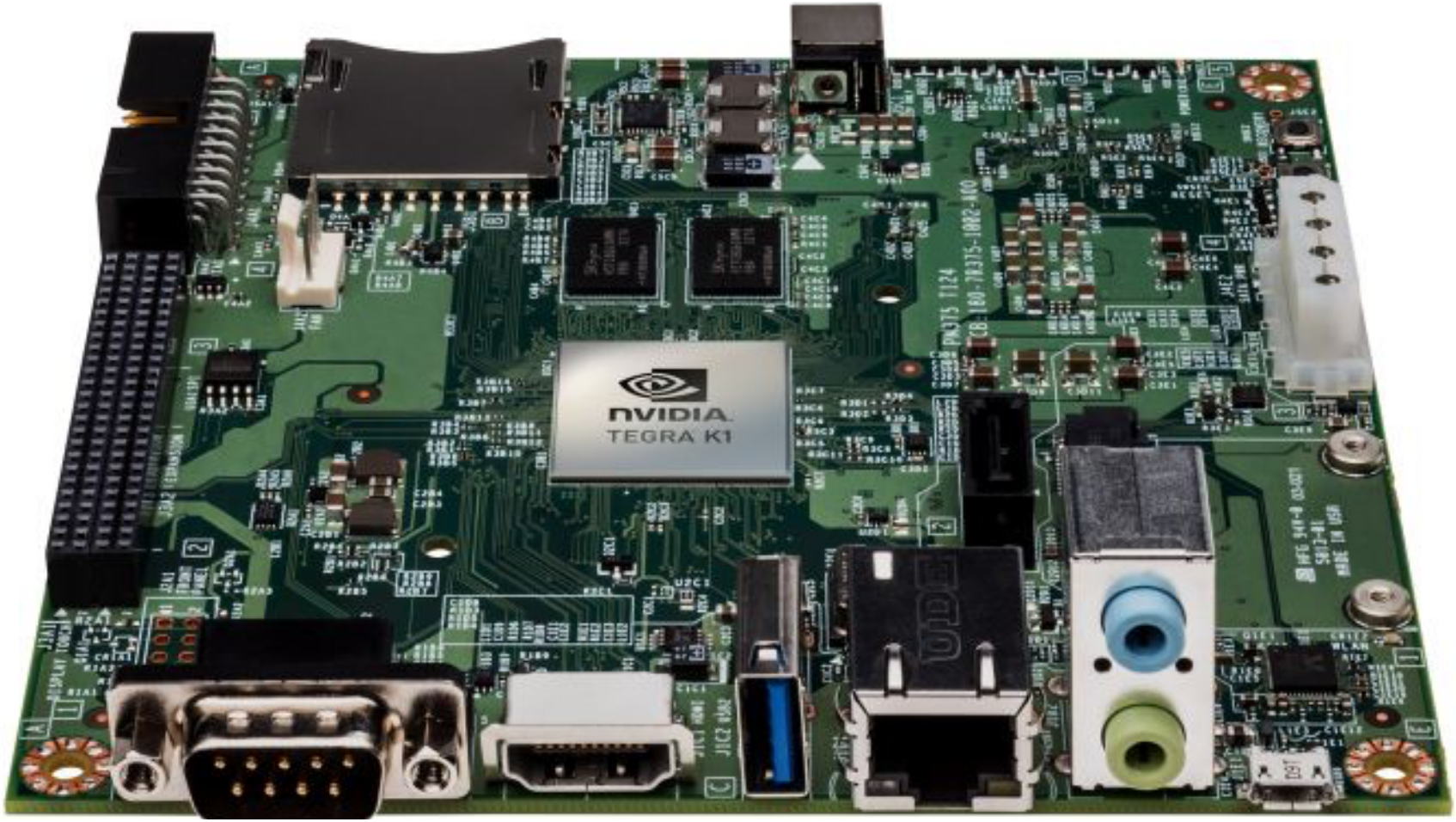
Georgia Tech College of Computing
Computational Science and Engineering

hpcgarage

Disclaimer!!

- I have never worked for NVIDIA
- All information found in this presentation is from the “internet.”
- Some of this information could be incorrect
- If you find errors, please point it out!!
  - Email me at jee@gatech.edu

Jetson TK1



Jetson TK1 Overview

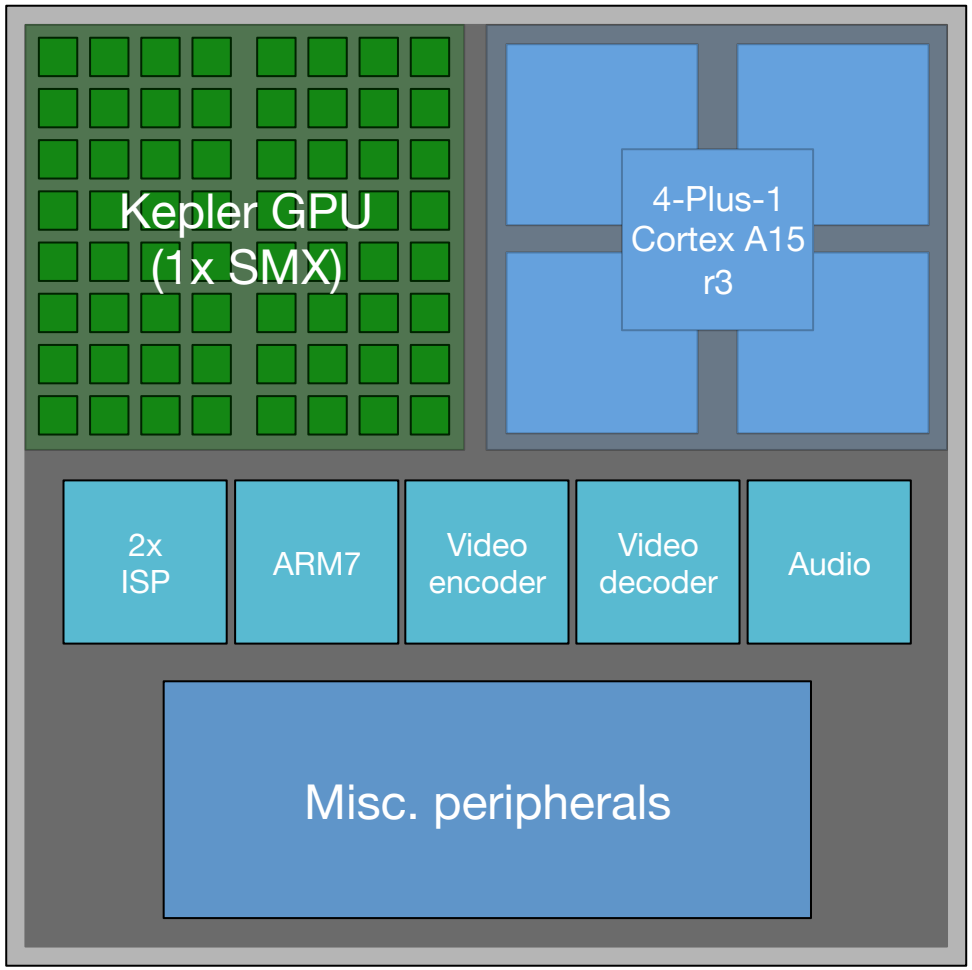
- Hardware features
  - 5”× 5” (127mm × 127mm)
  - Tegra K1 System-on-chip (SoC)
    - GPU – 1× NVIDIA Kepler GK20A SMX (SM 3.2)
    - CPU – NVIDIA “4-plus-1” ARM Cortex A15 quad-core processor with battery-saver core
    - **Typical** power consumption of 1~5 Watts
  - 2GB DDR3L DRAM
    - 64-bit data width
  - 12V DC power supply

Jetson TK1 Overview

- Software features
  - OS – Linux for Tegra (L4T)
    - Modified Ubuntu 14.04 Linux distribution
    - Kernel version 3.10.40
  - Supported media APIs
    - CUDA 6.5
    - Open GL 4.4, Open GL ES 3.3 / 1.1
    - OpenCV4Tegra (CPU + GPU)
    - DirectX 12
    - And more…

Tegra K1 System-on-chip (SoC)

- GPU performance
  - Single-precision
    - ~327 GFLOP/s
  - Double-precision
    - ~13.6 GFLOP/s
  - Memory bandwidth
    - ~14.8 GB/s
- DVFS
  - GPU clock
    - 72 MHz ~ 852 MHz
    - 15 steps
  - GPU memory
    - 12.75 MHz ~ 924 MHz
    - 12 steps



Performance Specifications

- Registers
  - 32K registers per block or SMX
- L1 cache and Shared memory
  - 128 Bytes (32-bit data) or 256 (64-bit data) Bytes per cycle
- L2 cache
  - 1024 Bytes per cycle (GK110)
  - 512 Bytes per cycle (GK104)
  - ? Bytes per cycle (GK20A)
  - 128 KB
- DRAM
  - 64-bit wide
  - 924 MHz DDR3L

Performance Specifications

- Instruction throughput
  - Dual-issue quad scheduler = 8 instructions per cycle (IPC)
  - Due to hardware limitations, 7 is the sustainable peak IPC
  - 6 math instructions + 1 LD/ST or SFU
- Requires ILP and operand sharing to reach theoretical peak math throughput

Tegra K1 vs. K40c

	Tegra K1	Tesla K40c	
# SMX	1	15	15×
Core clock	852 MHz	875[1] MHz	1×
Memory clock	924 MHz	3004 MHz	3.3×
Memory bus width	64-bit	384-bit	6×
Registers per block	32K	64K	2×
L2 cache	128 KB	1.5 MB (102.4 KB per SM)	12× (0.8×)
Peak single-precision throughput	327.2 GFLOP/s	5040 GFLOP/s	15.4×
Peak double-precision throughput	13.6 GFLOP/s	1680 GFLOP/s	124×
Peak memory throughput	14.8 GB/s	288 GB/s	19×
Thermal design power	10 Watts (?)	245 Watts	25×

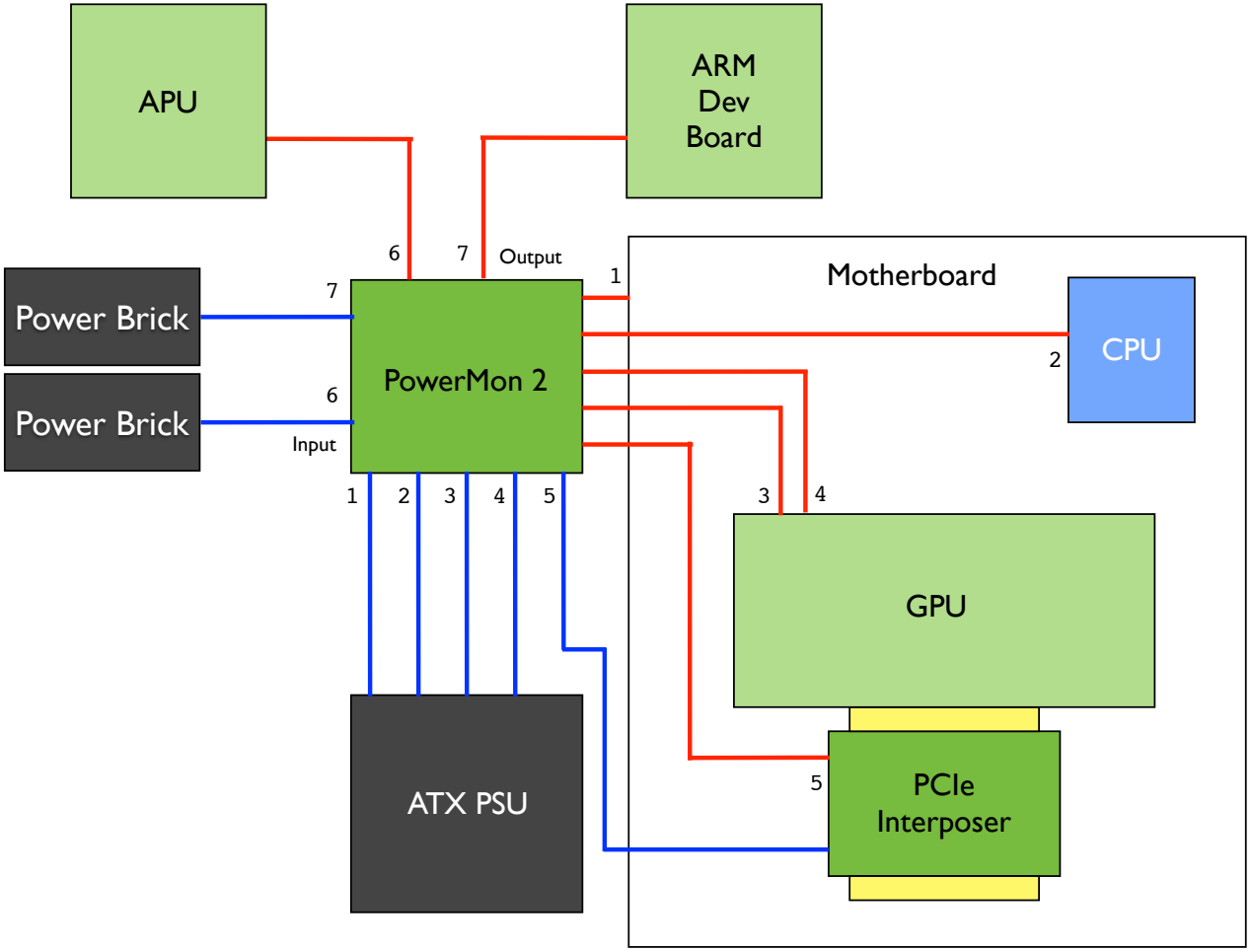
[1]Max clock. Default is 745 MHz

Power Measurement

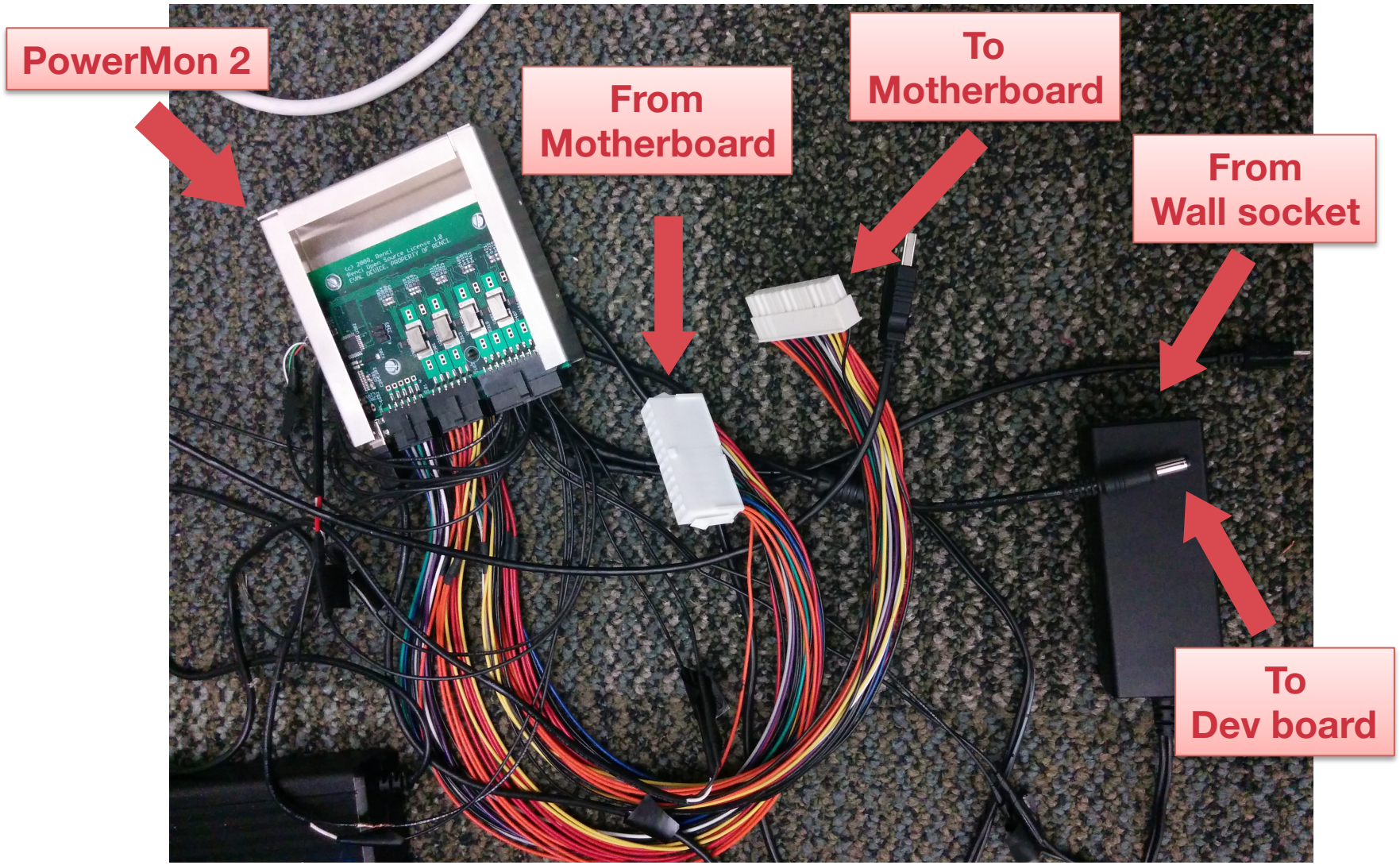
- Measuring platform – PowerMon2
  - Fine-grained, integrated power measurement tool
  - Developed at RENCI by Daniel Bedard, Allan Porterfield, Rob Fowler, and Min Yeol Lim.
  - Open source

    (https://github.com/beppodb/PowerMon/wiki)

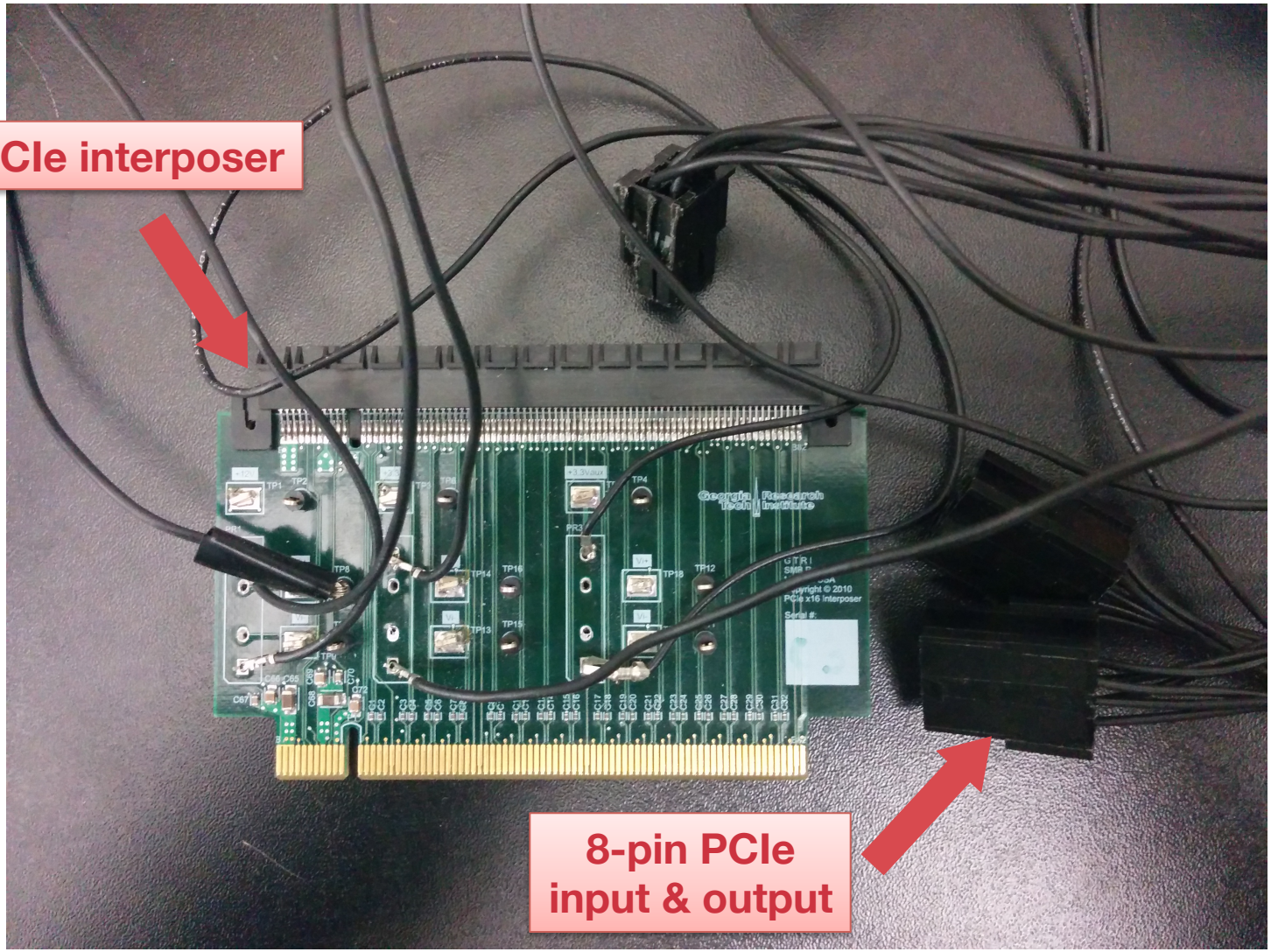
PowerMon 2 is placed between the power source and the target device



PowerMon 2 and interface cards



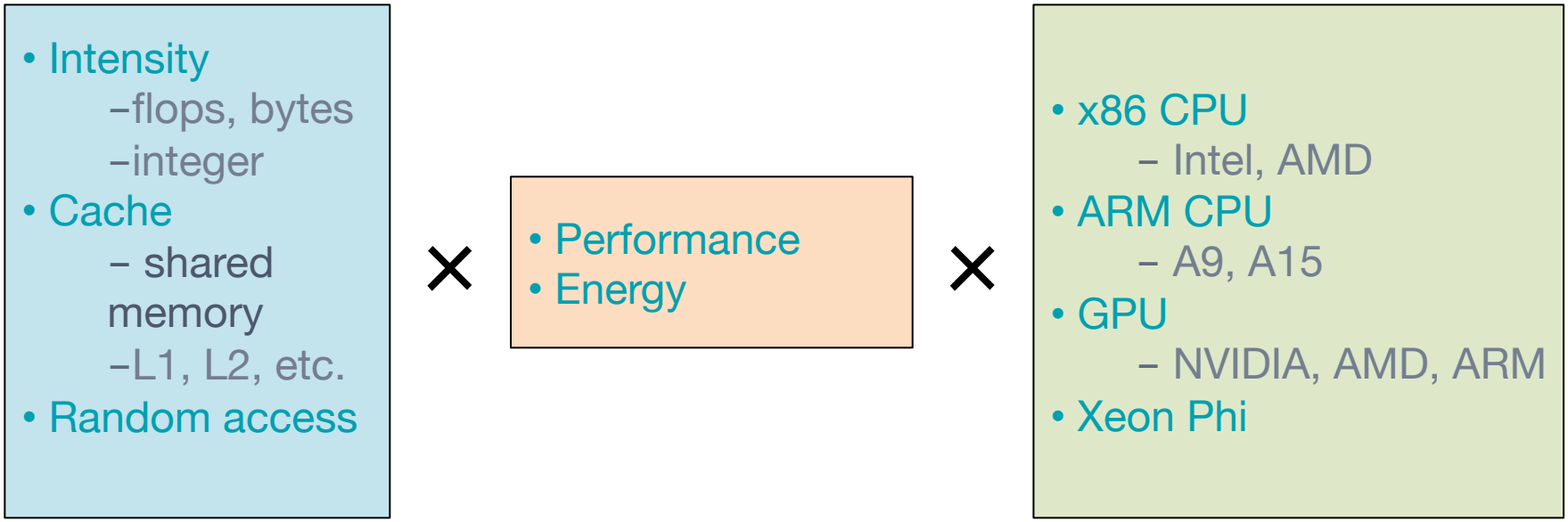
PCIe interposer card and PCIe power



Power measurement 101

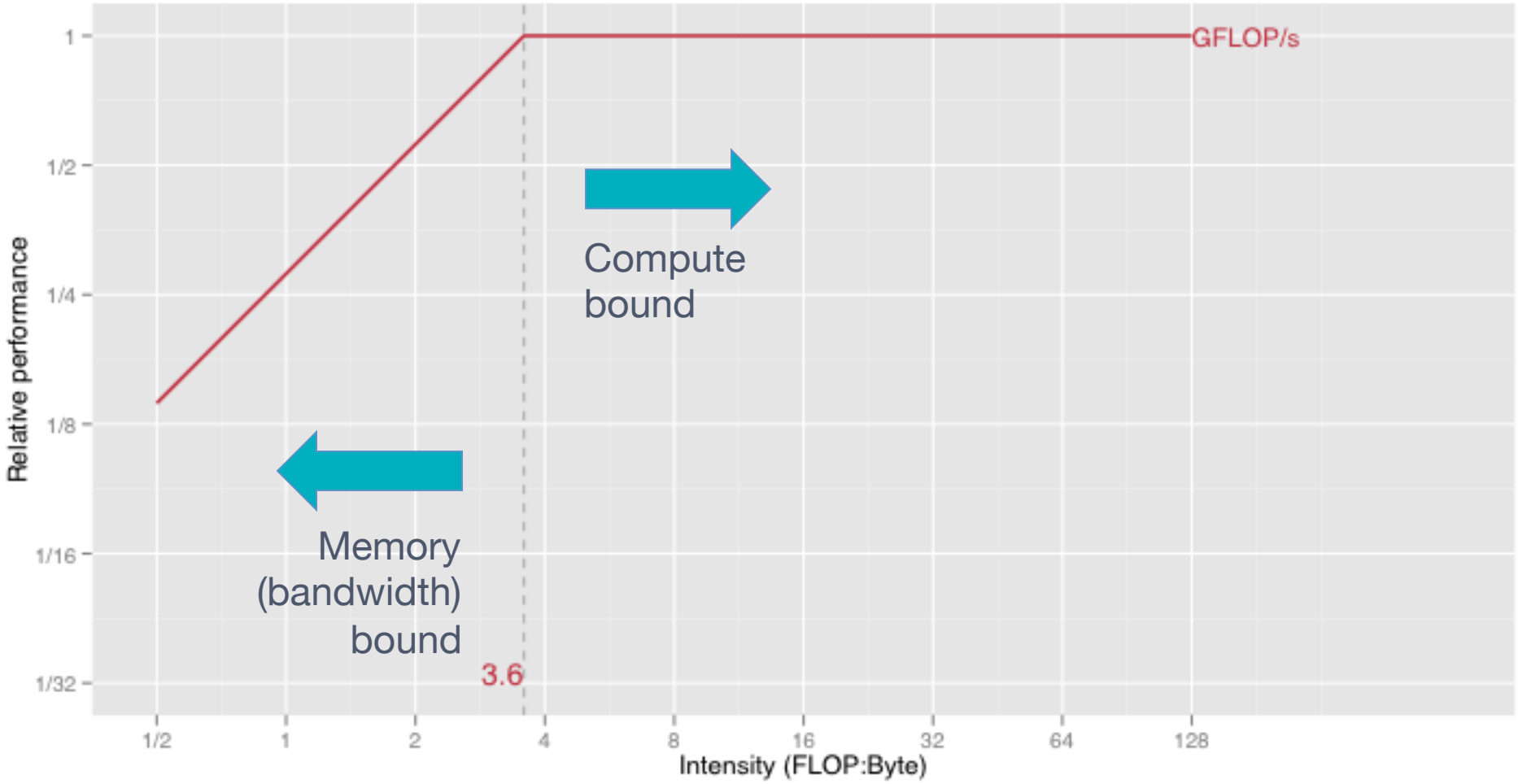
- Tutorial @ Micro'14
  - http://j.mp/micro14tutorial
- Tutorial @ SC'15
  - TBA

µbenchmark Suite



http://hpcgarage.org/archline

Arithmetic intensity



µbenchmark Suite

```
uint tid = threadIdx.x +
blockIdx.x * blockDim.x;
TYPE tmp1;
float x, y, z, w;

if(tid < num_threads) {
  tmp1 = in[tid];
  x = tmp1.x;
  y = tmp1.y;
  z = tmp1.z;
  w = tmp1.w;

  x = x + x * CONST;
  y = y + y * CONST;
  z = z + z * CONST;
  w = w + w * CONST;

  x = x + x * CONST;
  y = y + y * CONST;
  z = z + z * CONST;
  w = w + w * CONST;

  ...

  out[tid] = float4 (x, y, z, w);
```

- GPU Intensity µbenchmark for Kepler
  - Quad warp scheduler
    - vs. 6 warp execution units
    - need ILP in order to saturate pipeline
  - Register bandwidth
    - 4 operands per cycle (6 required to sustain 2 FMAs per cycle)
    - operand sharing between two FMAs or reuse through operand cache is **necessary**
  - Instructions per cycle
    - instruction latency
    - multiple pipelines
    - sustainable IPC of 7

http://hpcgarage.org/archline

Cost of operations

- Derivation method
  - Linear regression on measured data
  - E.g., energy = $W\varepsilon_{flop} + Q\varepsilon_{mem} + \pi_0 T$
- Floating point operation
  - Single
  - Double
- Memory operation
  - Byte from
    - DRAM
    - L2
    - L1
    - Shared memory
- Integer

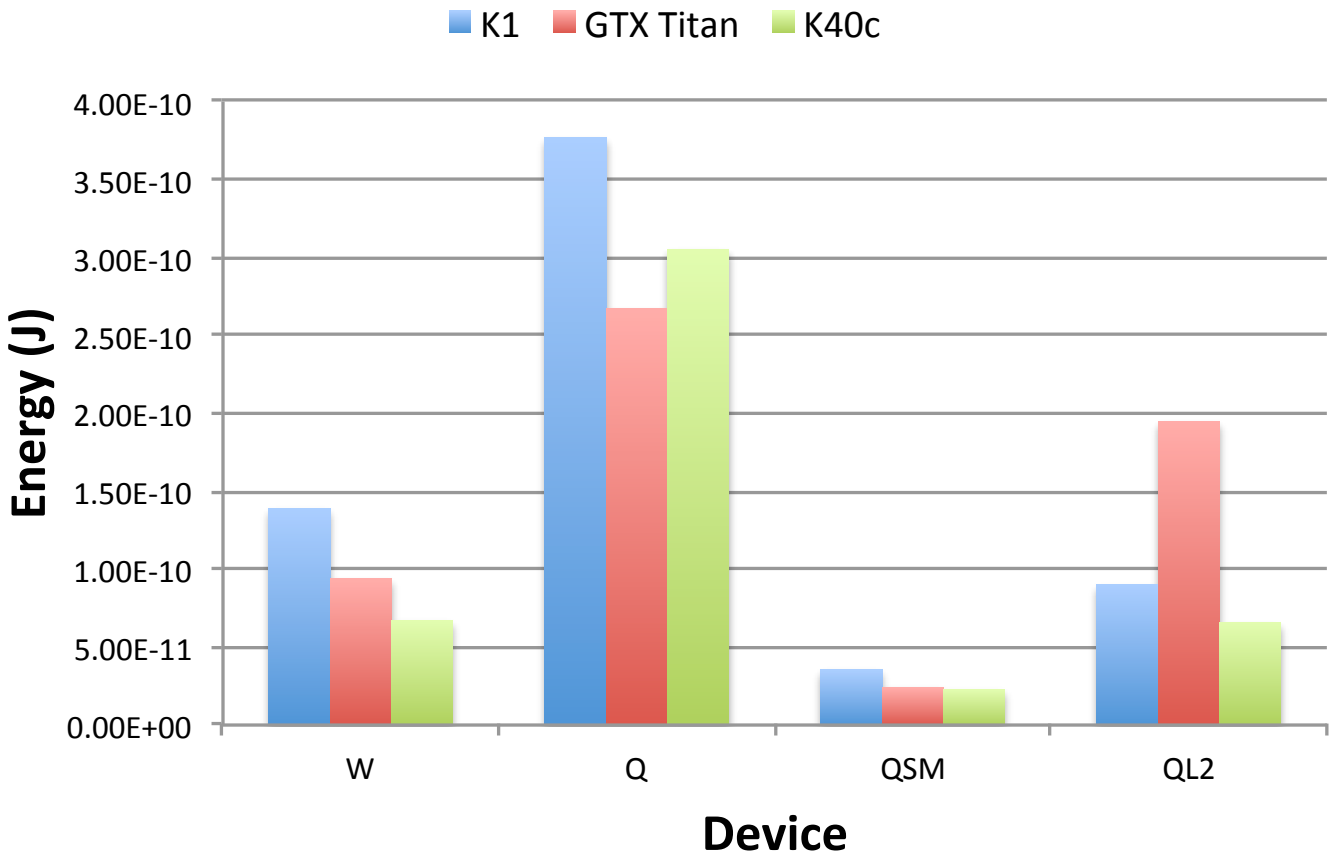
Performance Specifications

- Maximum observed memory throughput
  - Single-precision
    - 13.1 GB/s (/ 14.8 GB/s, ~90%)
    - 15.5 Watts
  - Double-precision
    - 13.2 GB/s (/ 14.8 GB/s, ~90%)
    - 11.4 Watts
- Maximum observed computational throughput
  - Single-precision
    - 304.5 GFLOP/s (/ 327.2 GFLOP/s, ~93%)
    - 17.0 Watts
  - Double-precision
    - 13.6 GFLOP/s (/ 13.6 GFLOP/s, ~99%)
    - 8.6 Watts

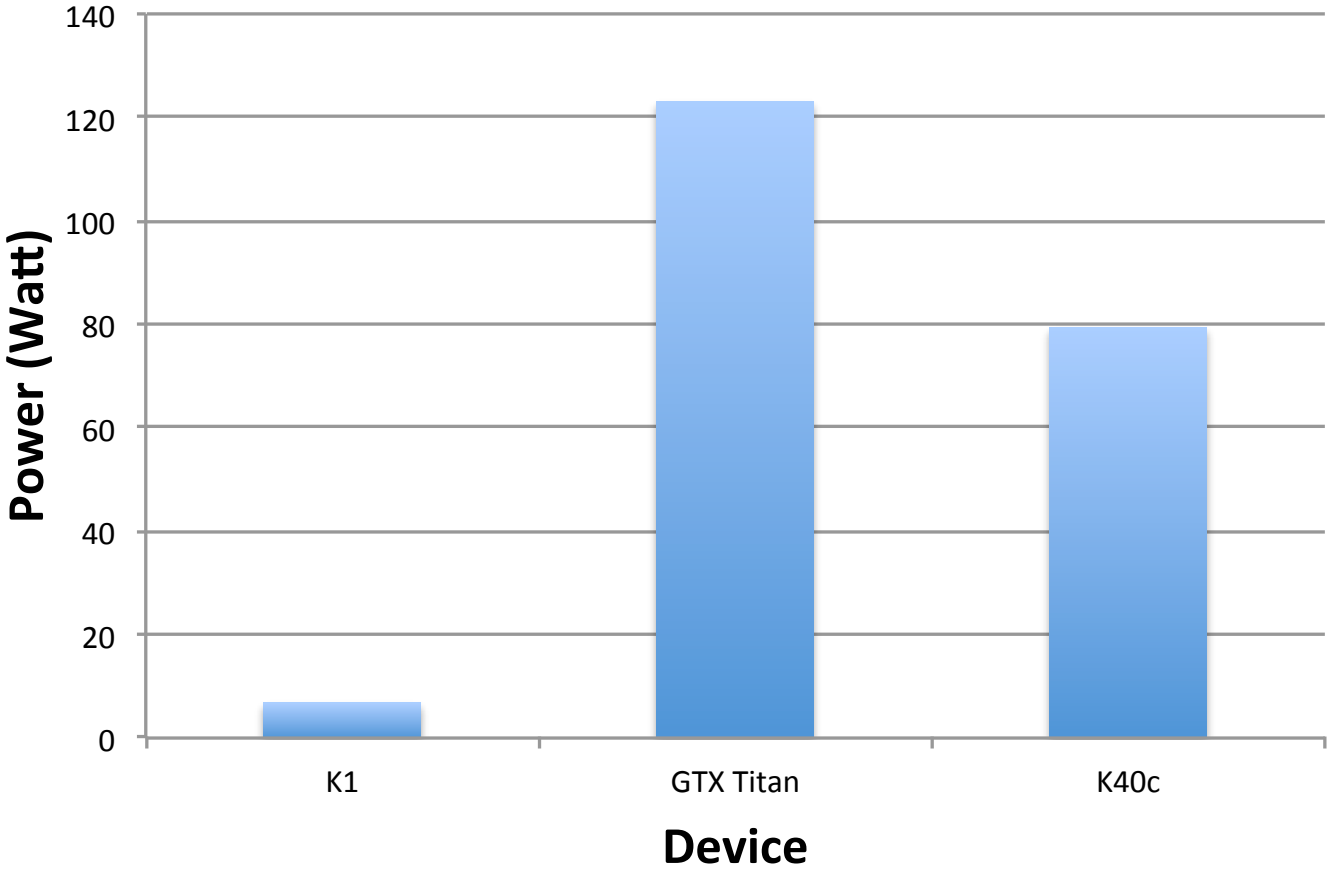
Performance Specifications

- Maximum observed shared memory/L1 cache throughput
  - 32-bit loads
    - 108 GB/s (/ 109 GB/s, ~99%)
  - 64-bit loads
    - 216 GB/s (/ 218 GB/s, ~ 99%)
- Maximum observed L2 cache throughput
  - 27.3 GB/s (/ ? GB/s)
- Maximum observed power
  - Single-precision intensity μbenchmark, arithmetic intensity = 32
    - 17.5 Watts

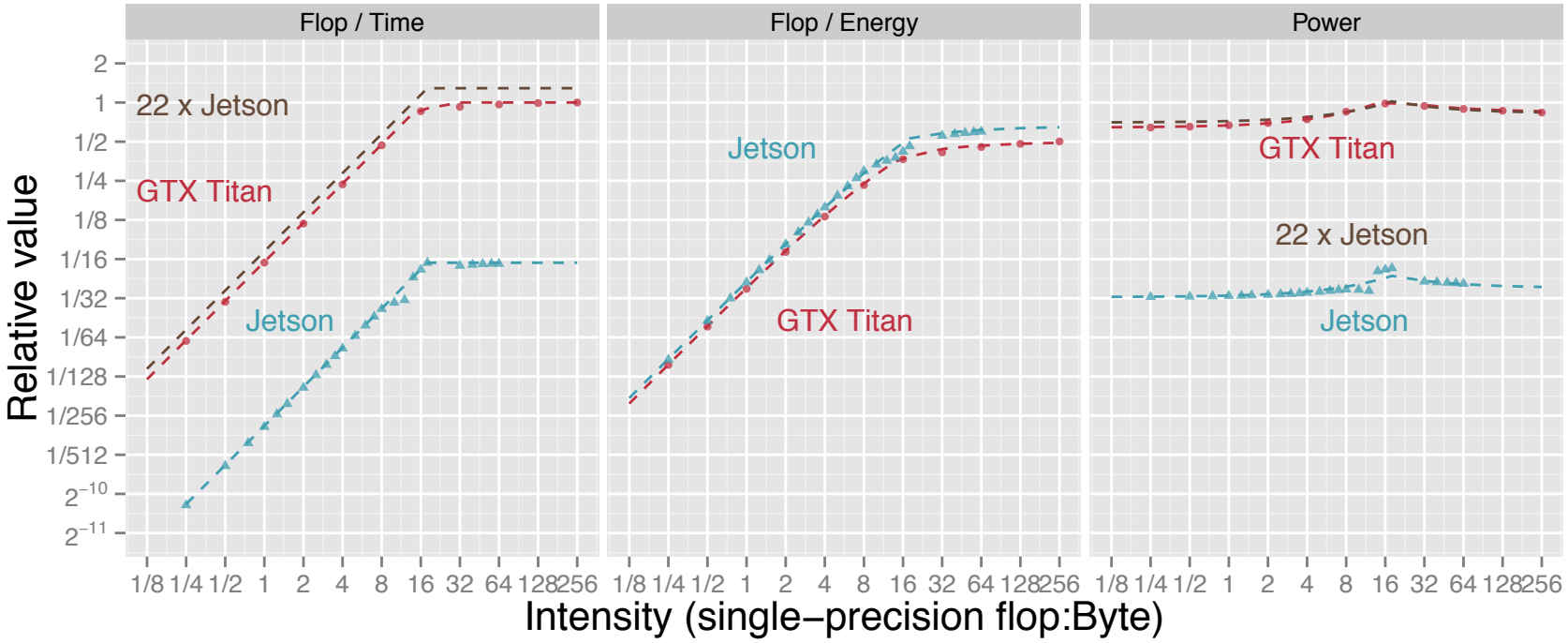
Energy cost comparison



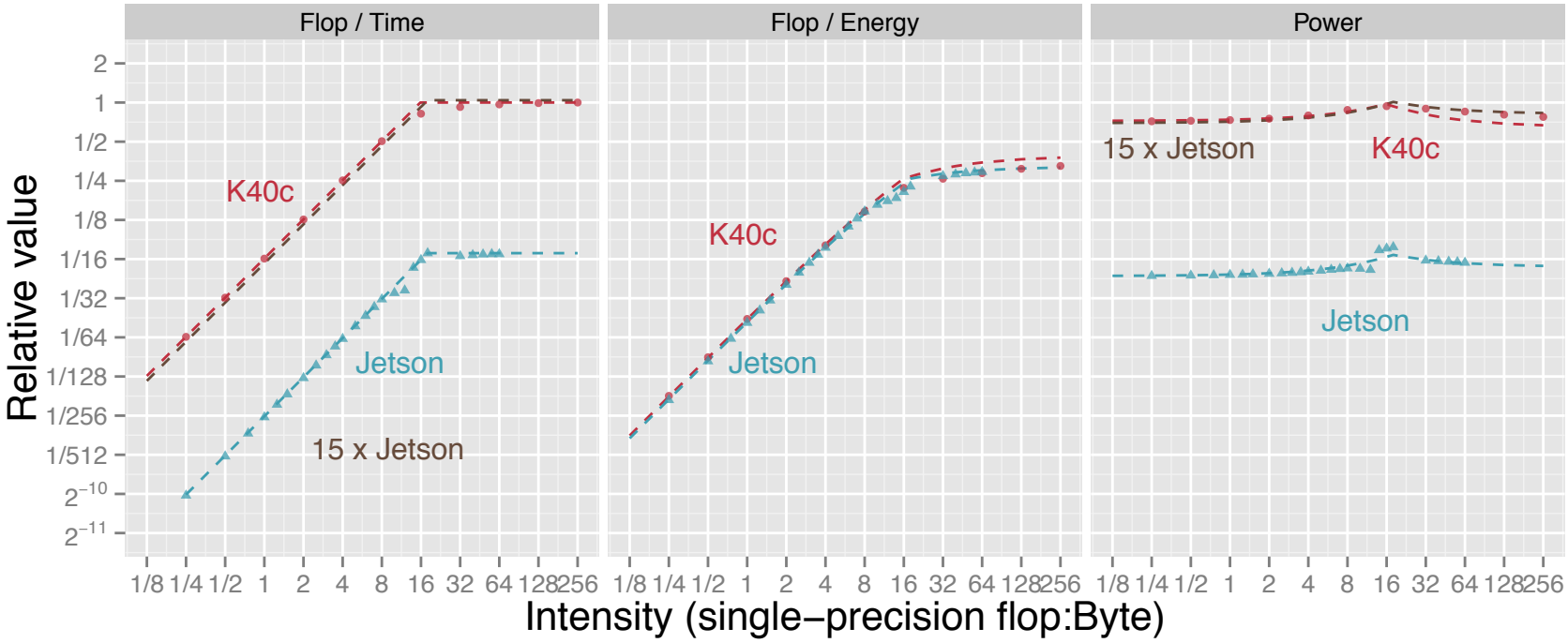
Constant (overhead) power comparison



Jetson TK1 vs. GTX Titan



Jetson TK1 vs. K40c



Dynamic Voltage and Frequency Scaling (DVFS)

GPU		Memory	
Voltage (mV)	**Frequency (MHz)**	**Voltage (mV)**	**Frequency (MHz)**
760	72	800	12.75
760	108	800	20.4
760	180	800	40.8
760	252	800	68
760	324	800	102
770	396	800	204
800	468	820	300
840	540	850	396
870	612	880	528
890	648	910	600
900	684	980	792
920	708	1010	924
950	756		
990	804		
1030	852		

Dynamic Voltage and Frequency Scaling (DVFS)

- Print out the DVFS table

```
sudo cat /sys/kernel/debug/clock/dvfs_table
```

- Change frequency setting
  - For core frequency

```
echo 648000000 > /sys/kernel/debug/clock/override.gbus/rate
echo 1 > /sys/kernel/debug/clock/override.gbus/state
```

  - For memory

```
echo 924000000 > /sys/kernel/debug/clock/override.emc/rate
echo 1 > /sys/kernel/debug/clock/override.emc/state
```

Dynamic frequency & voltage scaling

$$\epsilon_{flop} = C_0$$

$$\begin{aligned} \epsilon_{flop} &= P_{dyn,flop}\tau_{flop} \\ &= C_{0,core}V_{core}^2 f_{core}\tau_{flop} \\ &= C_{0,core}^{''}V_{core}^2 \end{aligned}$$

$$\epsilon_{mem} = C_1$$

$$P_{dyn} = CV^2Af$$

$$\begin{aligned} \epsilon_{mem} &= P_{dyn,mem}\tau_{mem} \\ &= C_{0,mem}V_{mem}^2 f_{mem}\tau_{mem} \\ &= C_{0,mem}^{''}V_{mem}^2 \end{aligned}$$

$$\pi_0 = C2$$

$$P_{leak} = V\left(ke^{-qV_{th}/(ak_aT)}\right)$$

$$\begin{aligned} \pi_0 &= \pi_{core} + \pi_{mem} \\ &= P_{leak,core} + P_{leak,mem} \\ &= C_{1,core}V_{core} + C_{1,mem}V_{mem} \end{aligned}$$

Validation results

- 16-fold cross validation
  - Mean error = 6.56%
  - Standard deviation = 3.8%
- Microbenchmarks
  - Mean error = 2.74%
  - Standard deviation = 2.73%
- Fast Multipole Method (FMM) kernel
  - Mean error = 6.17%
  - Standard deviation = 4.65%

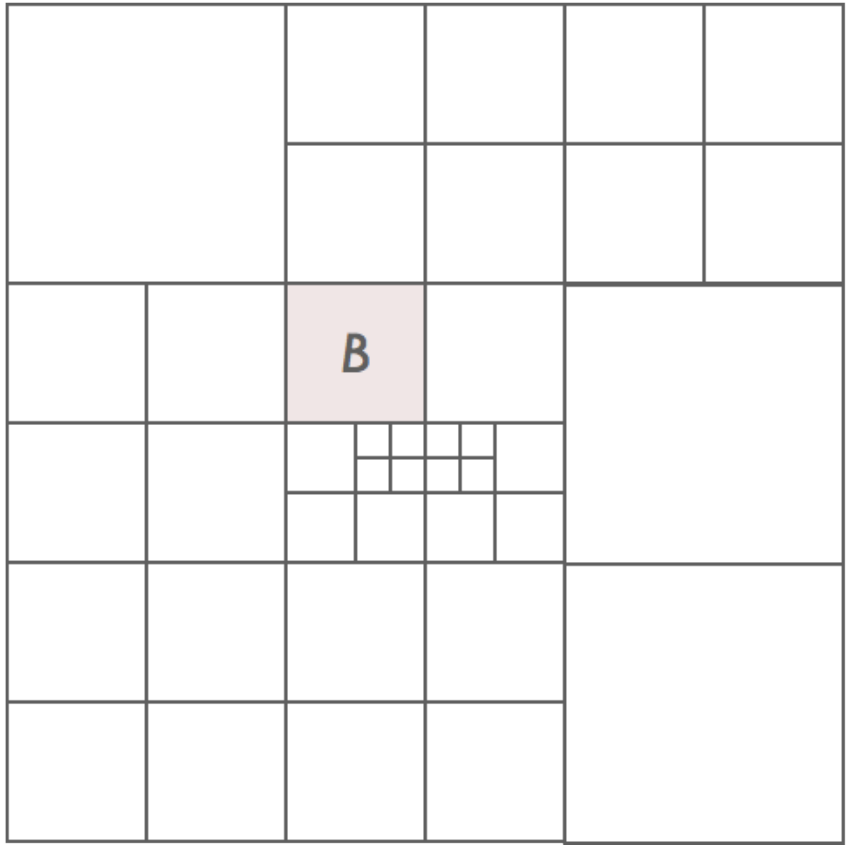
Validation results

- Time vs. model. Which is better at predicting energy?

		Mispredictions	Energy lost (%)		
			Mean	Minimum	Maximum
Single	Model	0 / 25	0	0	0
	Oracle	20 / 25	18.52	7.21	26.52
Double	Model	10 / 36	3.11	0.34	7.30
	Oracle	23 / 36	3.95	0.23	13.90
Integer	Model	6 / 23	2.37	0.32	5.12
	Oracle	23 / 23	3.56	0.44	9.72
Shared memory	Model	7 / 10	3.31	2.92	3.99
	Oracle	10 / 10	10.64	7.07	12.75
L2	Model	9 / 9	1.67	1.12	2.01
	Oracle	9 / 9	10.71	10.49	11.28

Fast Multipole Method (FMM)

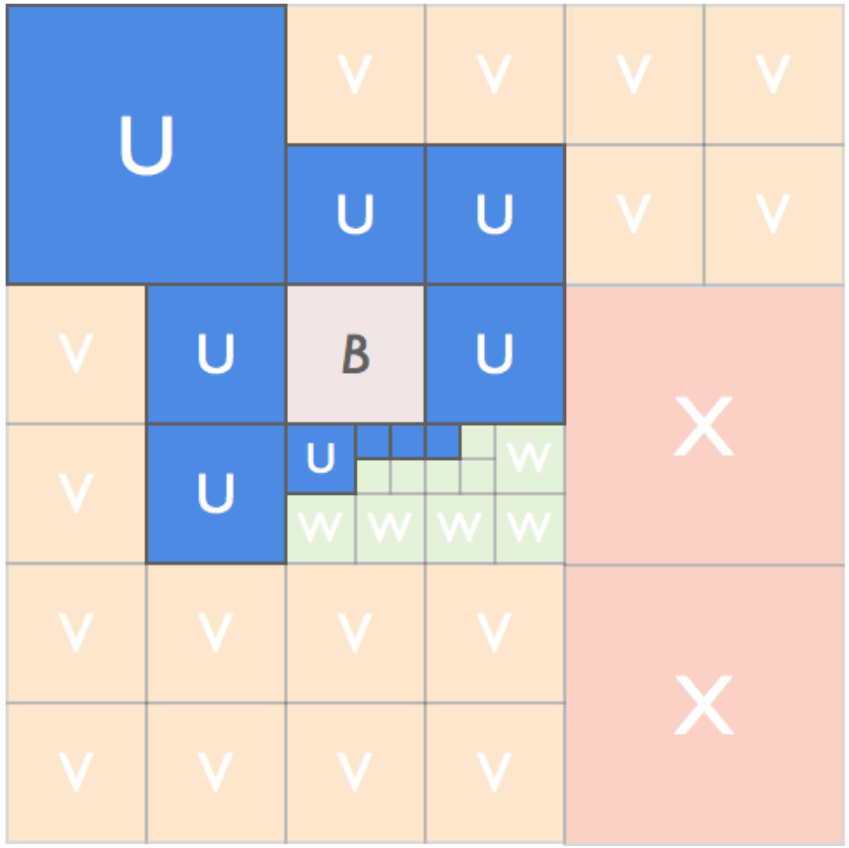
- Tree Construction
  - Recursively divide space until each box has at most ***q*** points
- Evaluation (Uniform)
  - Upward
  - U-List
  - V-List
  - Downward
- Phases vary in:
  - Data parallelism
  - Compute intensity



U-List

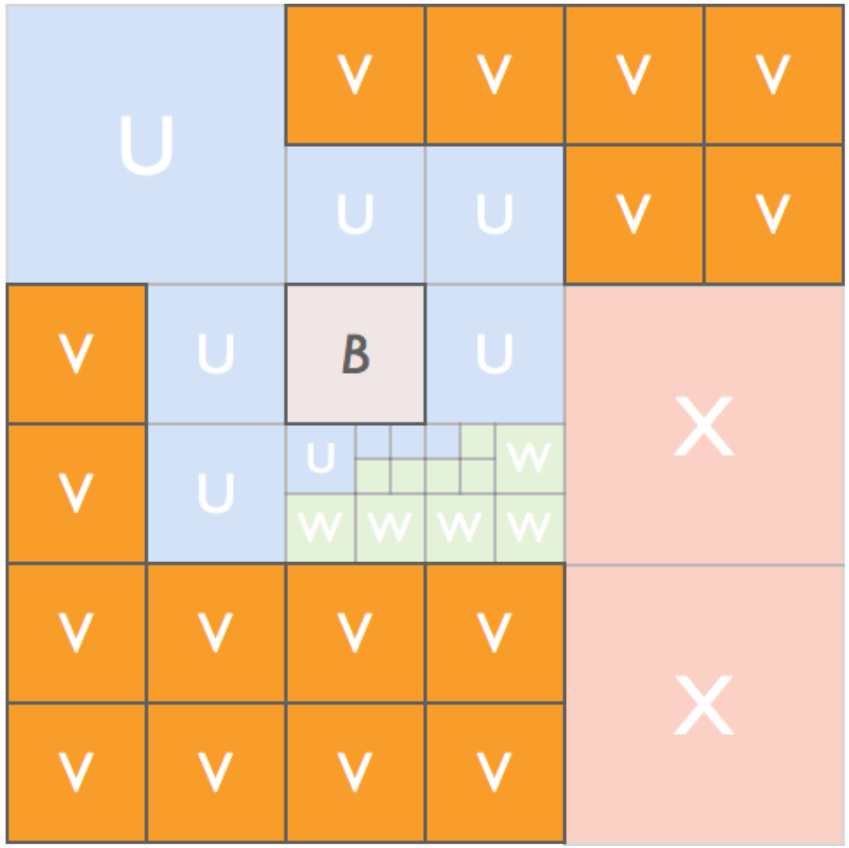
- Direct $B \otimes U$:

→ $O(q^2)$ *flops* : $O(q)$ *mops*

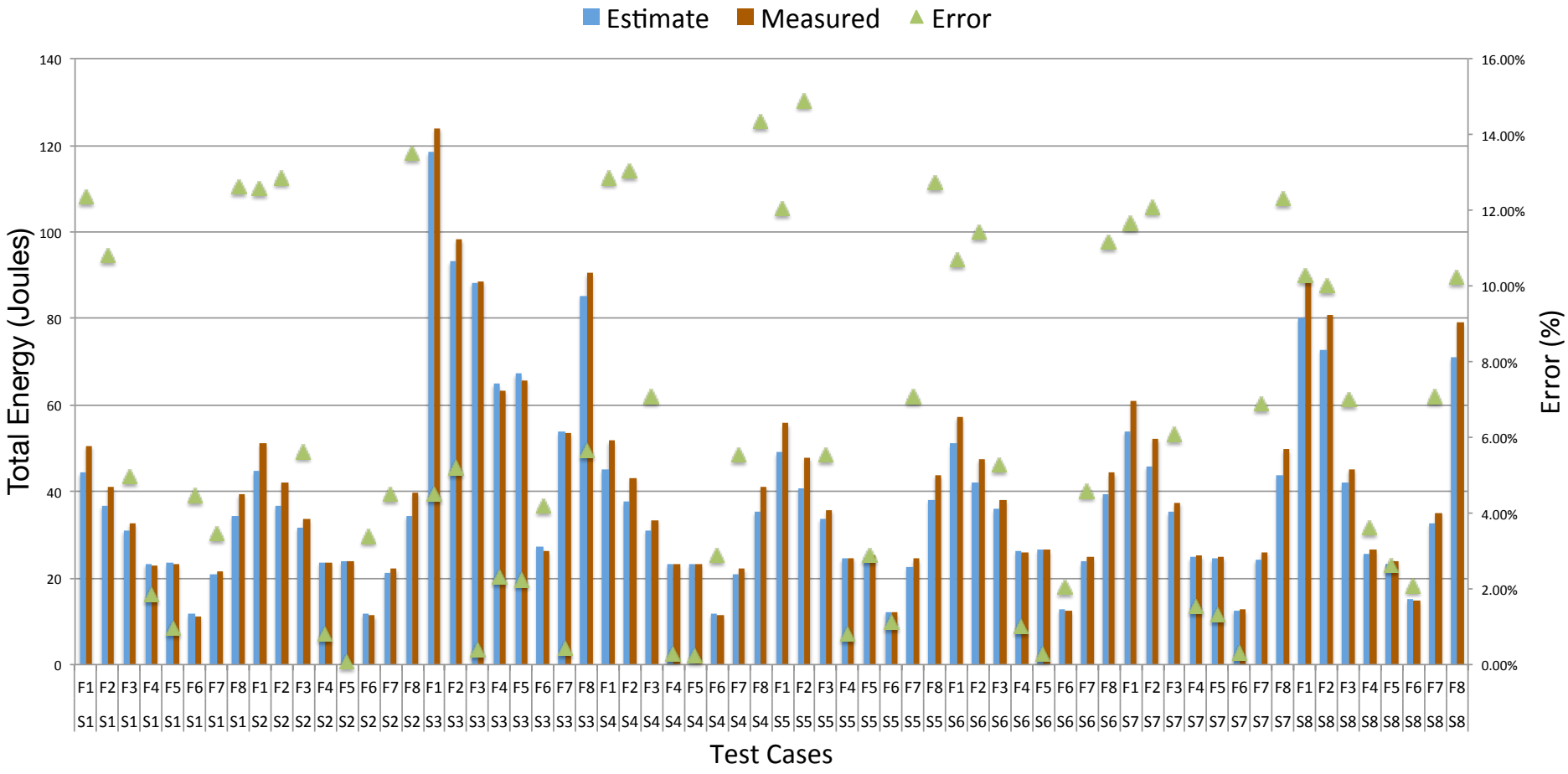


V-List

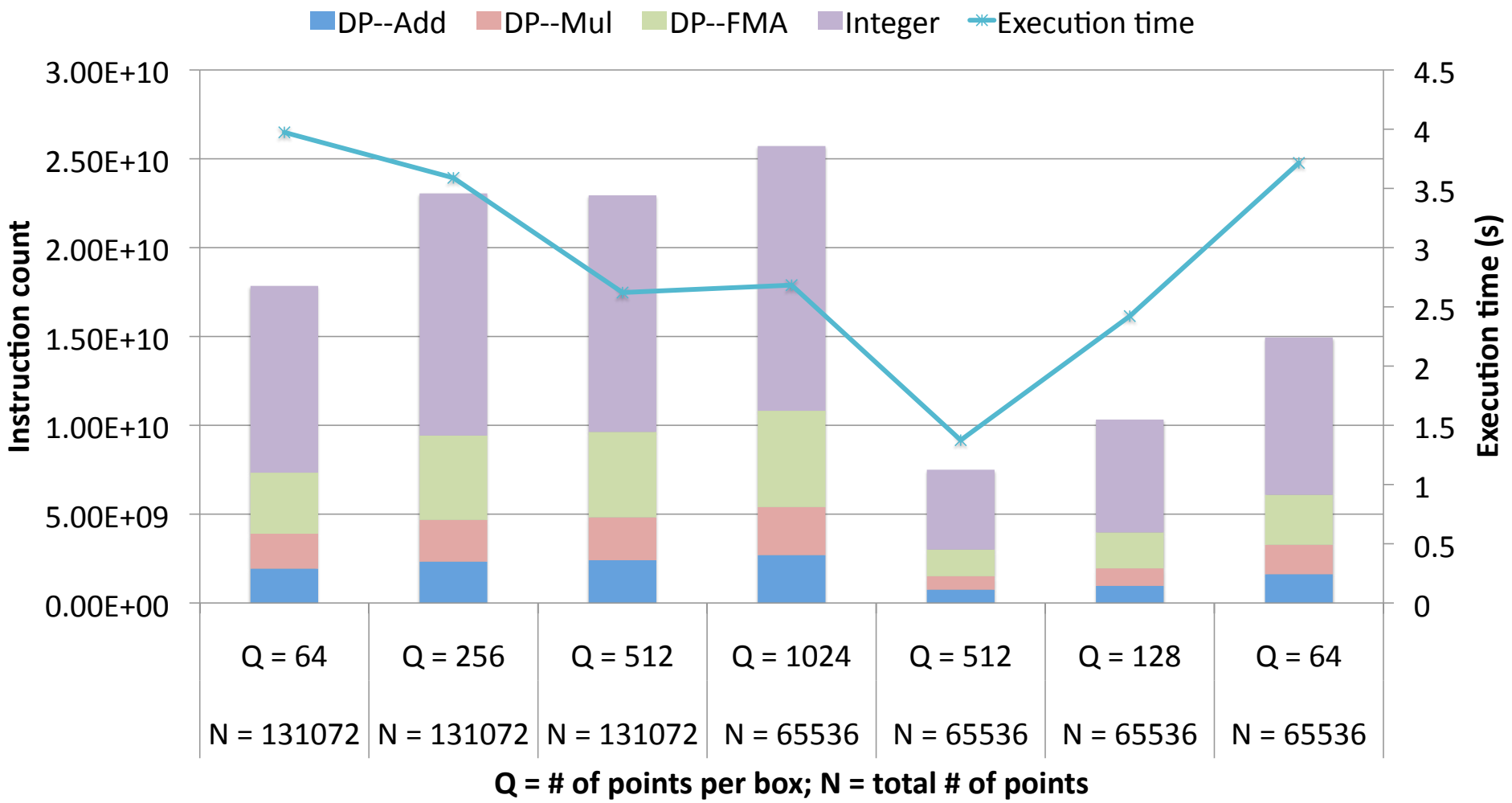
- FFT
- Point-wise multiplication
- IFFT



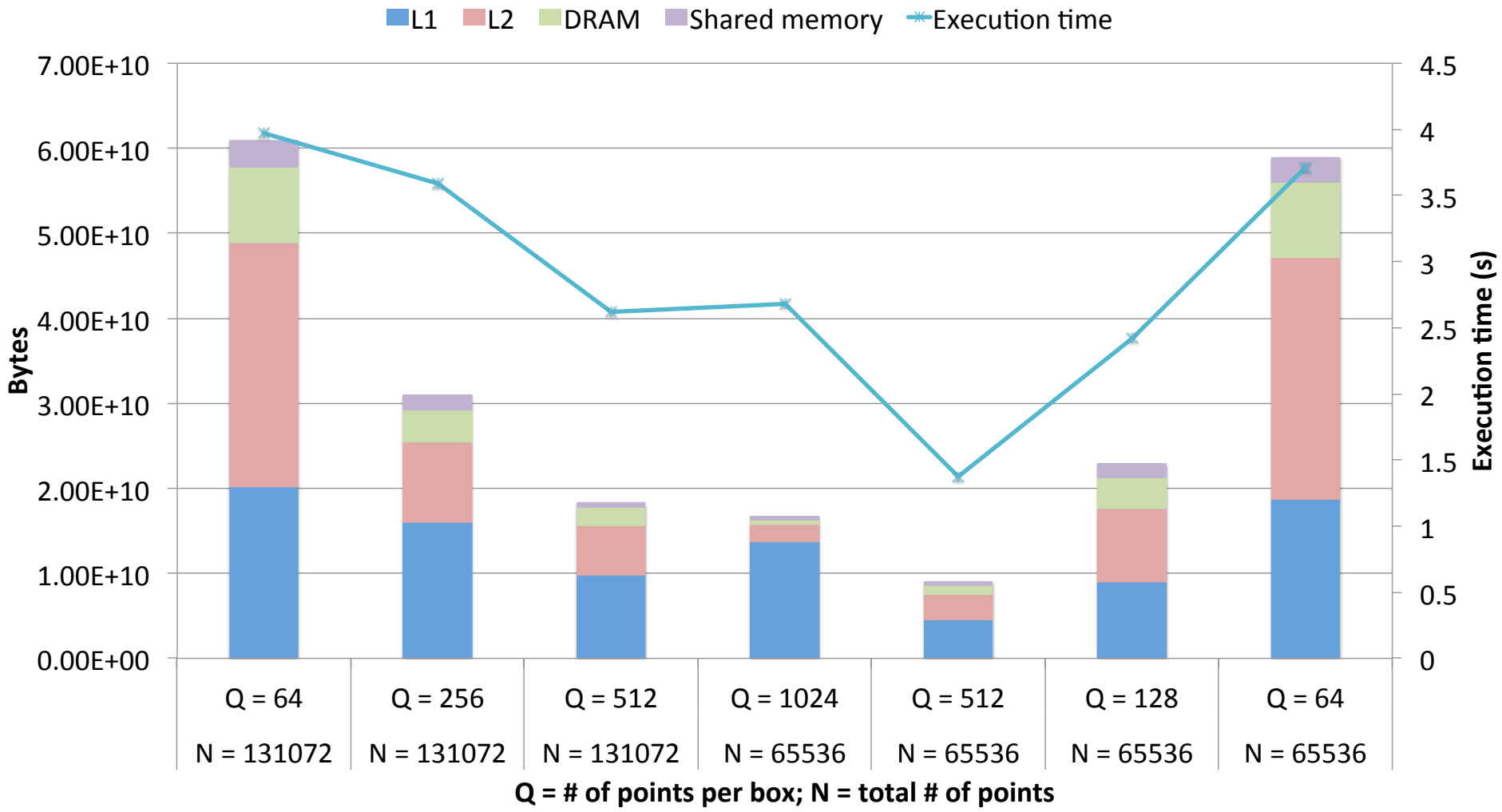
Modeling Error



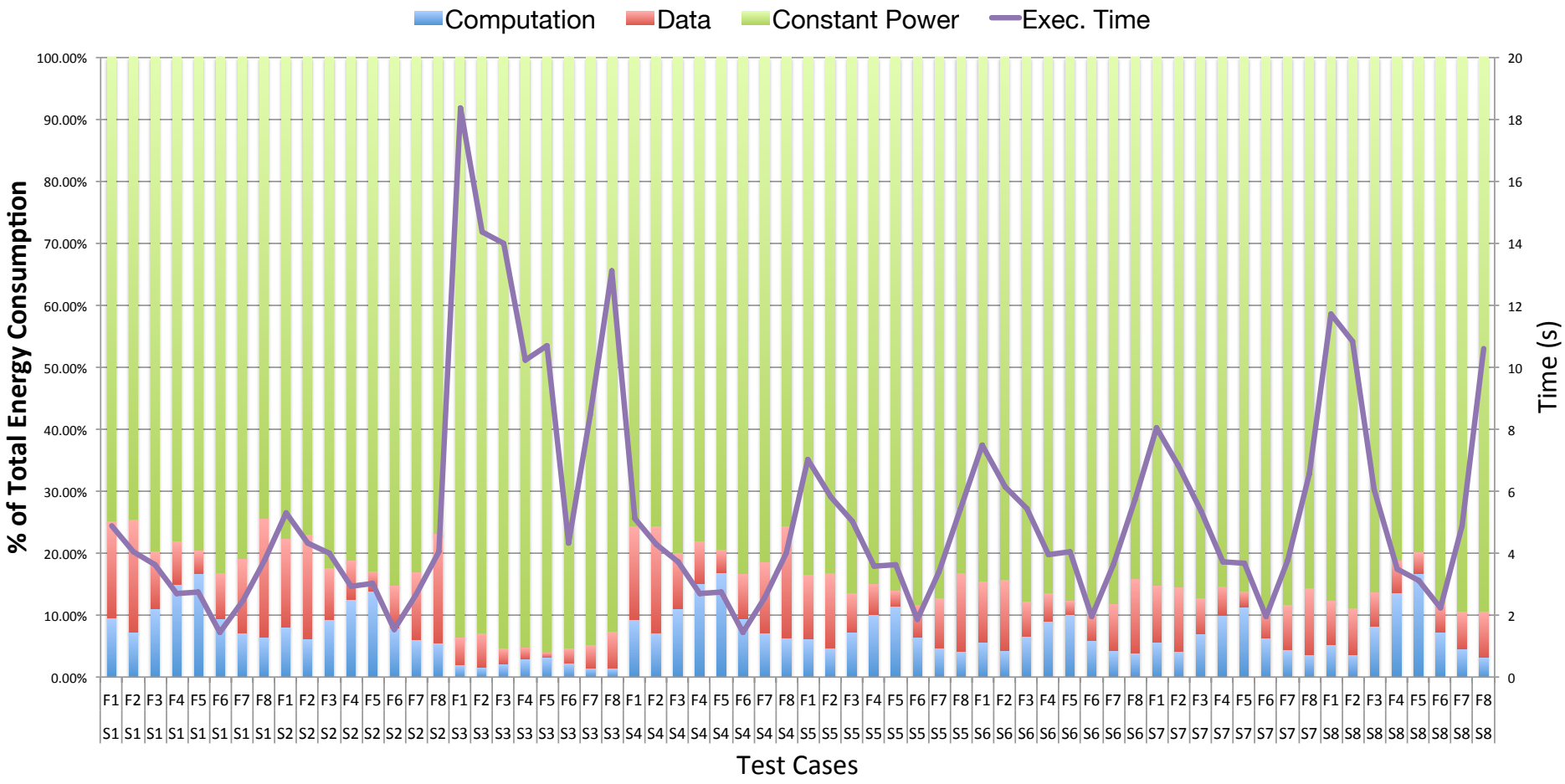
Instruction breakdown (FMM)



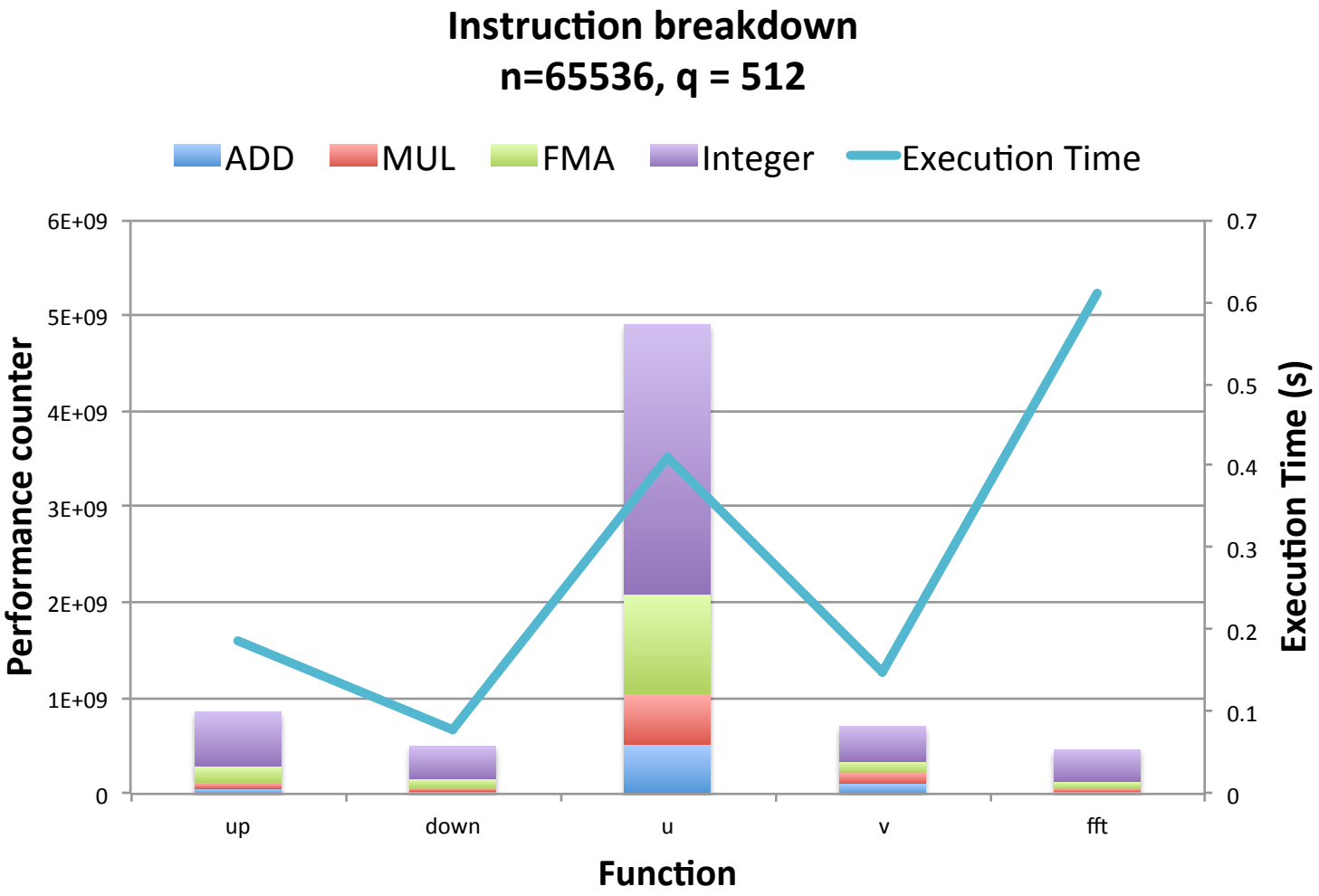
Data access breakdown (FMM)



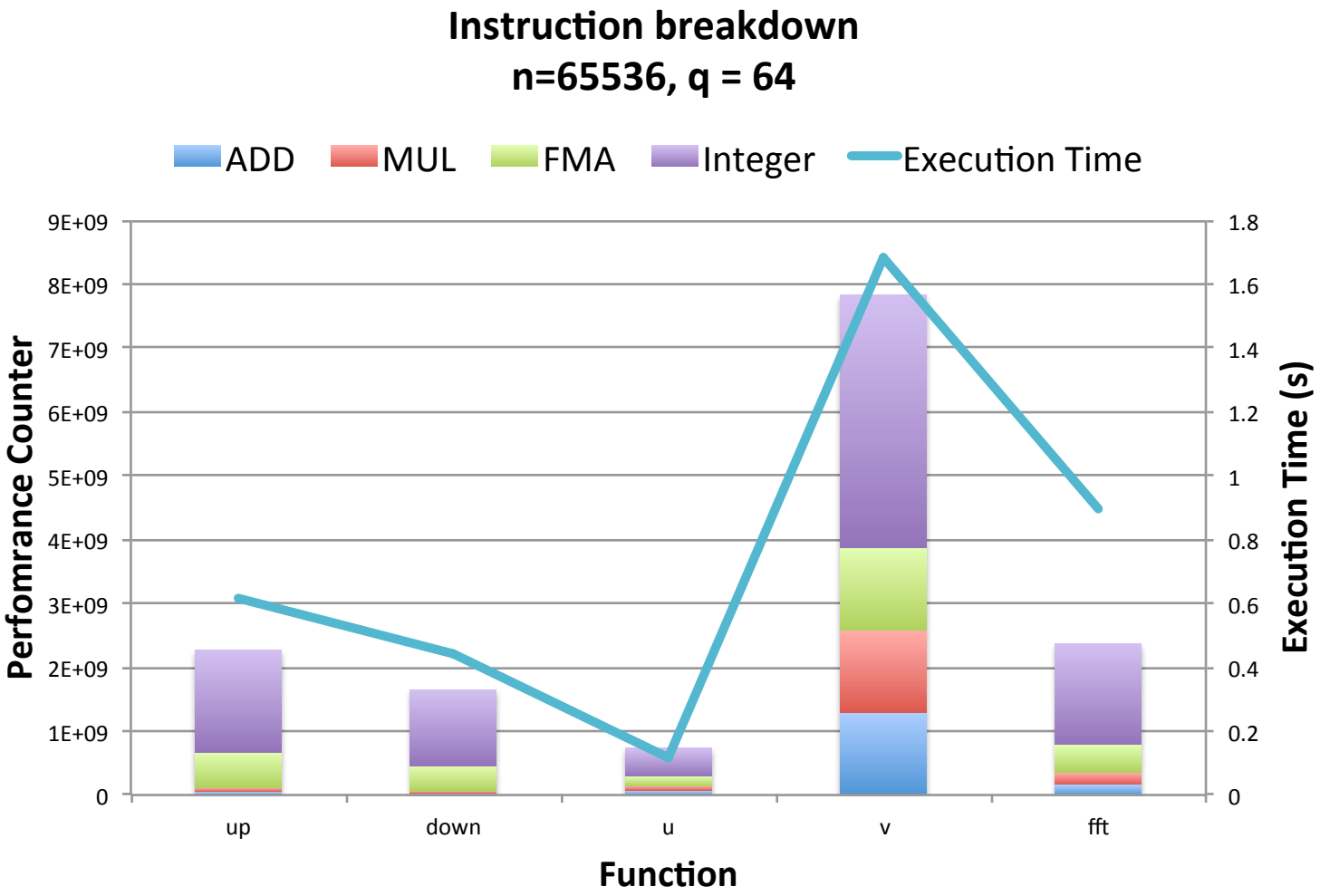
Energy % breakdown



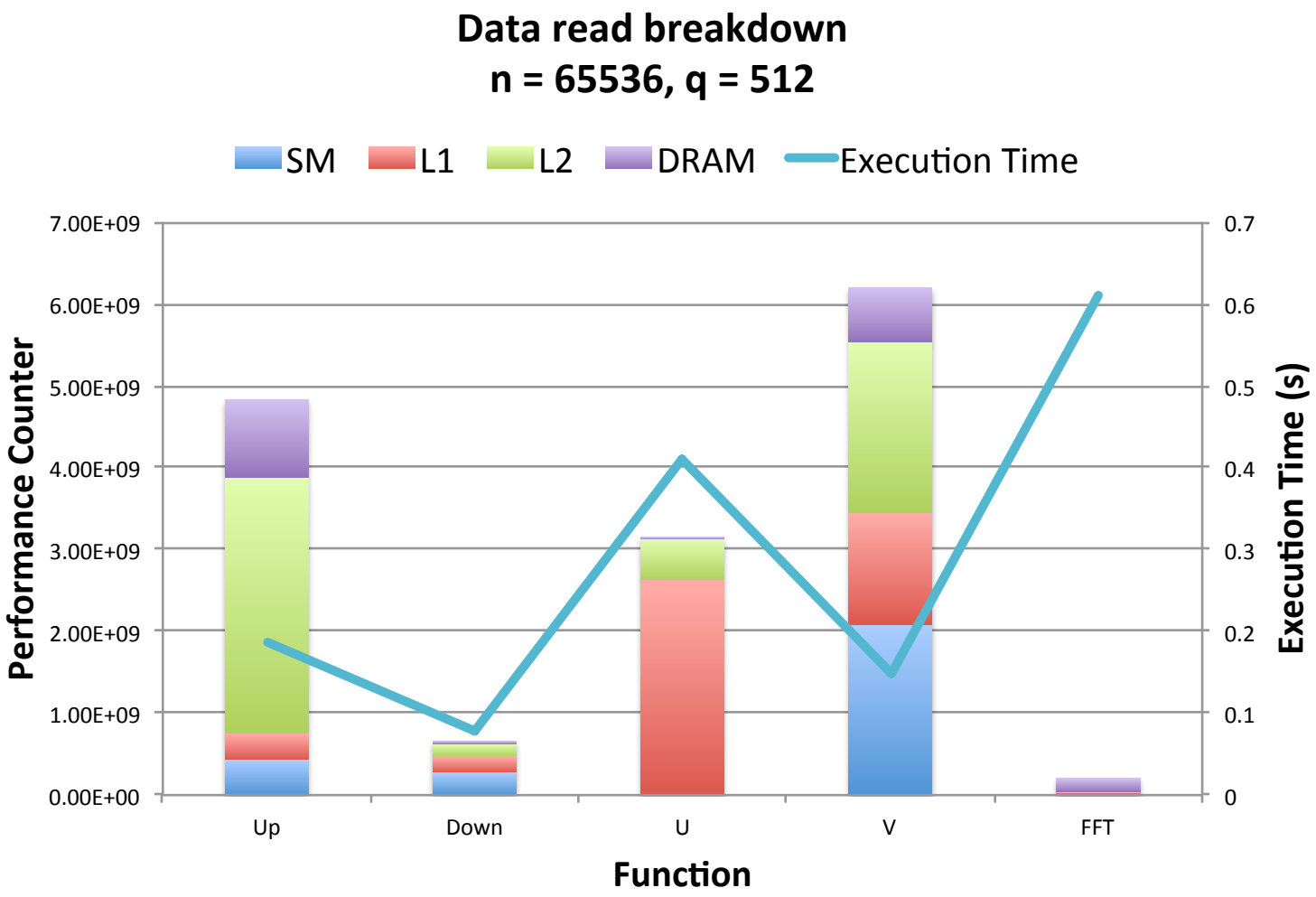
Breakdown by function - instructions



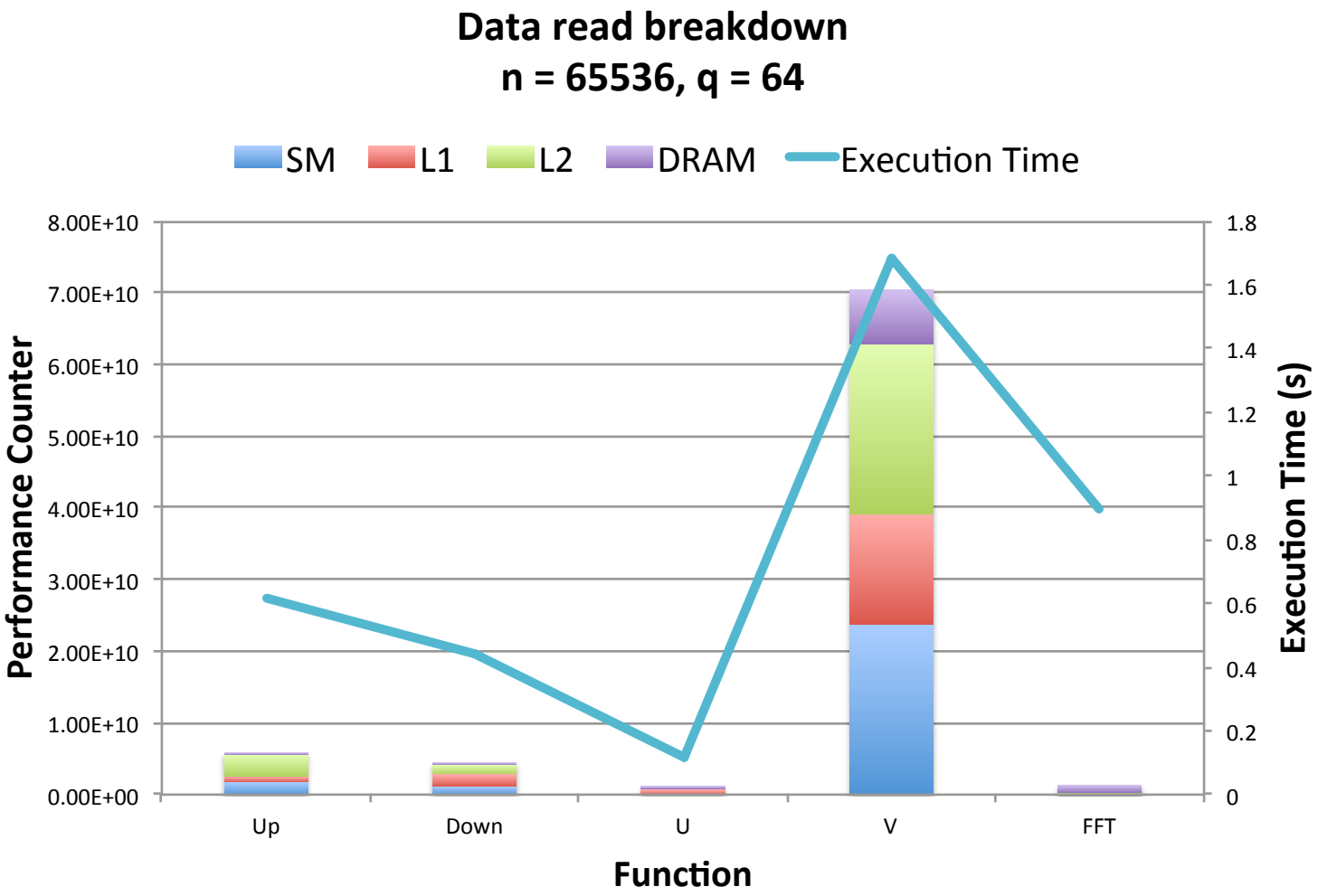
Breakdown by function - instructions



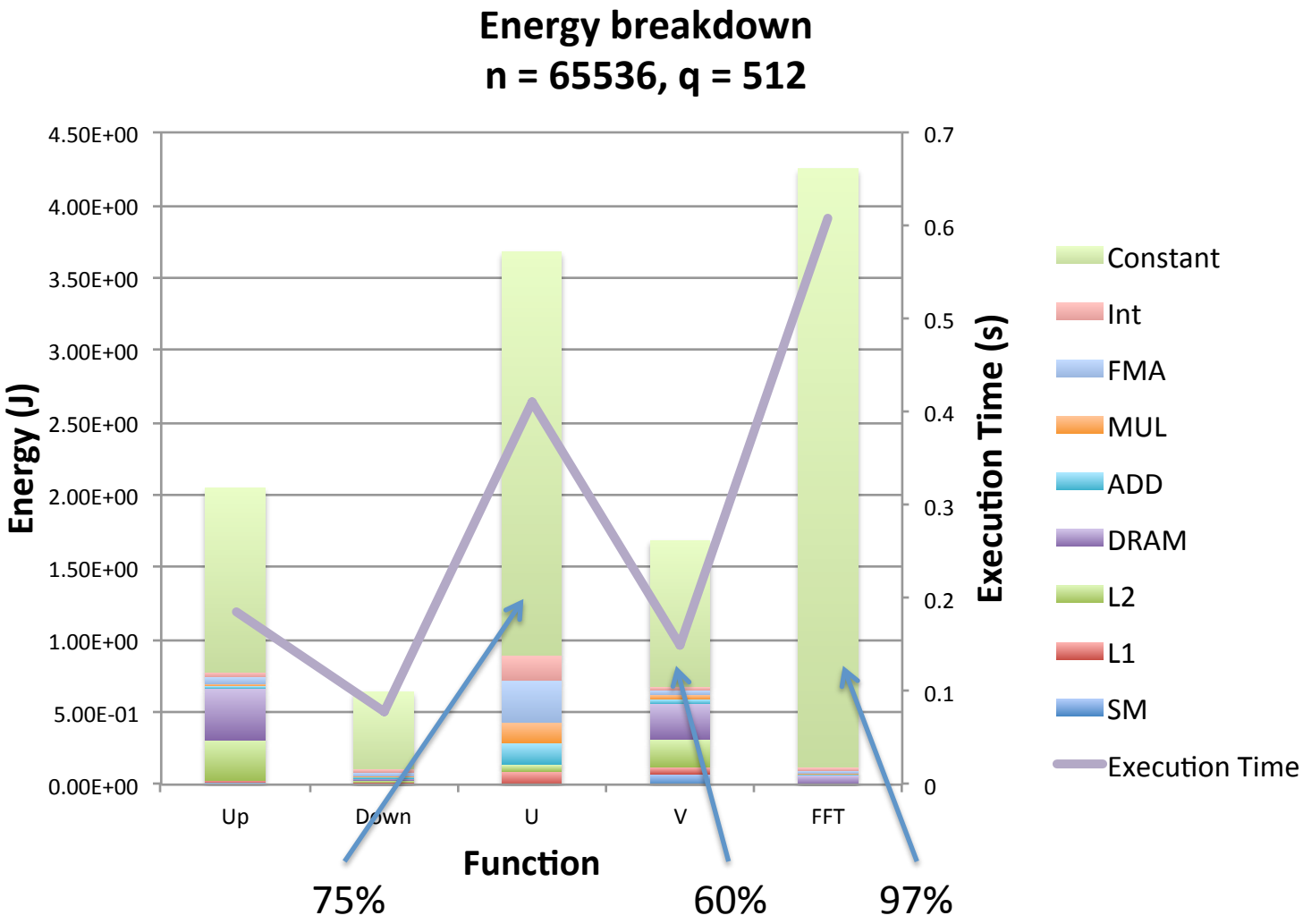
Breakdown by function - data



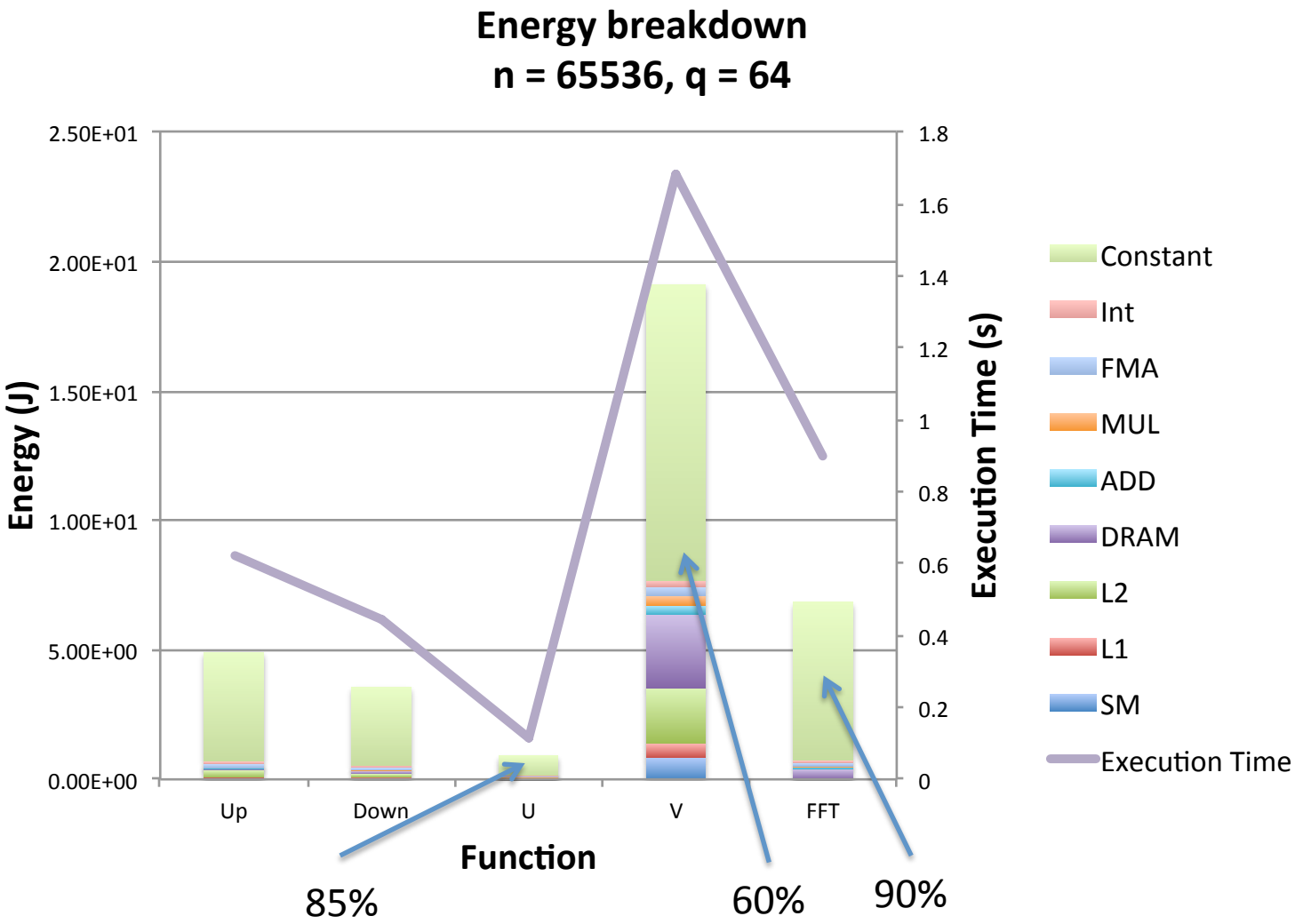
Breakdown by function - data



Breakdown by function - instructions



Breakdown by function - instructions



Contact Information

- Jee Choi
  - jee@gatech.edu
- Richard Vuduc
  - richie@cc.gatech.edu