

# Per-Core Power Estimation for CMPs

---

Bhavishya Goel

Magnus Sjölander

Sally A. McKee

Chalmers University of Technology, SE



**CHALMERS**

# Motivation

- ❑ Power now major design/operational constraint
- ❑ Accurate dynamic power information essential for power aware resource management
- ❑ Little hardware support for core level power measurement

# Statistical Power Model Using PMCs

## □ Why Statistical?

- ◆ **Simple**: requires little analysis of microarchitecture
- ◆ **Portable**: uses generic parameters

## □ Why performance counters?

- ◆ Hooks for peeking into microarchitectural activities
- ◆ Available separately for each core
- ◆ Fairly ubiquitous
- ◆ Easily accessible

# Basic Types of PMCs

## □ Architectural

- ◆ Consistent across processor models
- ◆ Usually small set
- ◆ Examples: Instructions Retired, Last Level Cache References, Unhalted Core Cycles

## □ Non-architectural

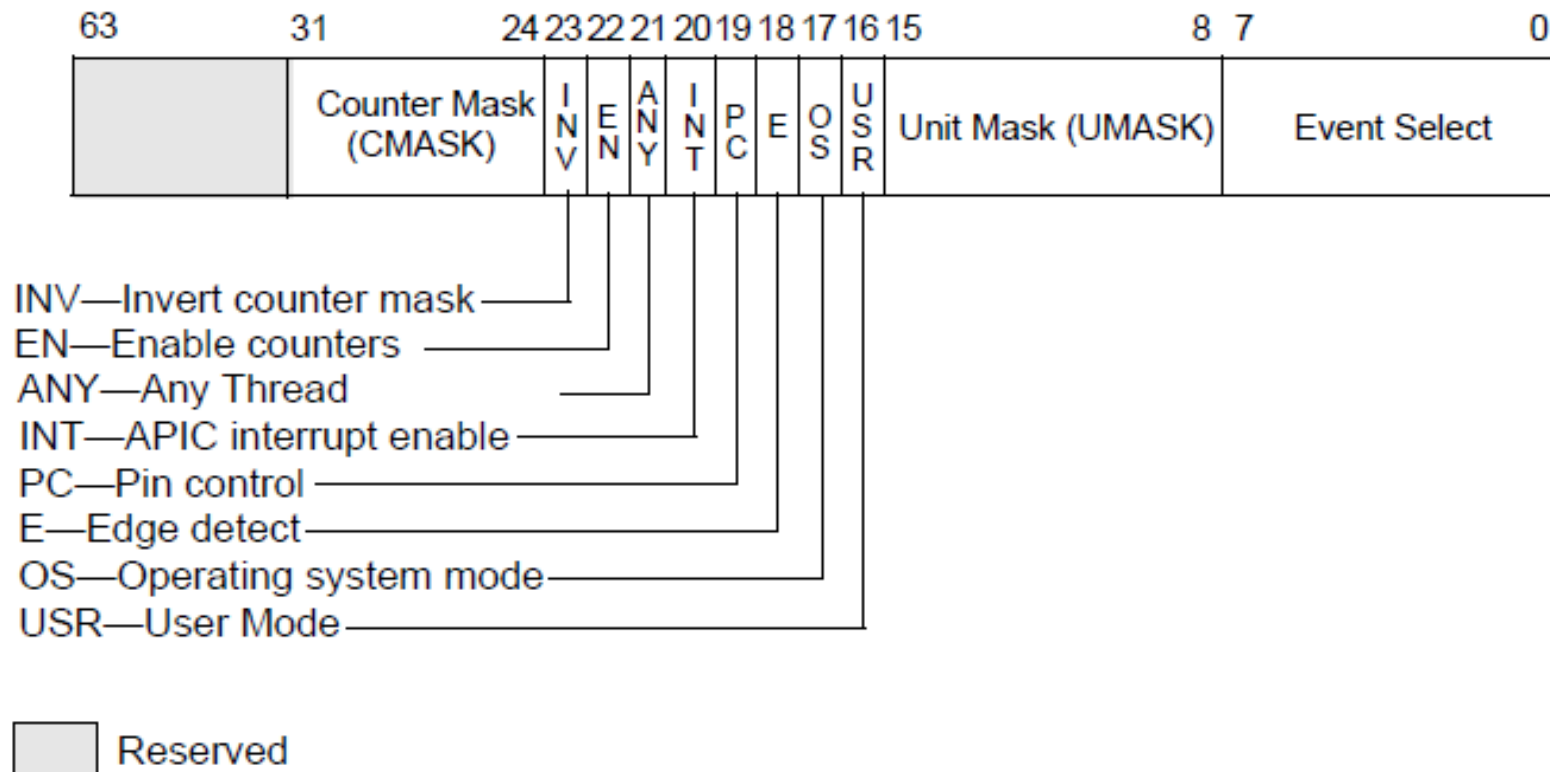
- ◆ Specific to microarchitecture
- ◆ Varies with processor models
- ◆ Larger available set
- ◆ Examples: UOPS Retired, FP Comp Operations Executed, Number of Loads Dispatched

# Reading PMCs

- ❑ PMCs are mapped to model specific registers (MSRs)
- ❑ Limited set of MSRs available
- ❑ Usually 2 (E.g., Intel Core Duo) or 4 (E.g., Nehalem) for general-purpose PMCs
- ❑ MSRs are accessed using `WRMSR`/`RDMSR` – only from privilege level 0
- ❑ PMCs can also be accessed using `RDPMC` – from any privilege level when `CR4.PCE=1`

# Mapping of General Purpose PMCs

- PMCs mapped on `IA32_PMCx` registers
- Mapping architectural PMCs using `PERFVTSELx` MSR



# PERFEVTSELx Configuration Example

<b>Event UOPS_EXECUTED</b>	<b>Umask</b>	<b>Event Code</b>	<b>Cmask</b>	<b>Inv</b>	<b>Edge</b>
Execution Stall Cycles	3FH	B1H	1	1	0
Execution Stall Count	3FH	B1H	1	1	1
Execution Active Cycles	3FH	B1H	1	0	0
Total UOPS Executed	3FH	B1H	0	0	0
Cycles when $\geq 4$ UOPS Executed	3FH	B1H	4	0	0

# Fixed Counters

- In addition to general purpose PMCs
- Fixed mapping to particular event (E.g.,  
INSTRUCTIONS\_RETIRED,  
UNHALTED\_CORE\_CYCLES, and  
UNHALTED\_REFERENCE\_CYCLES on Core i7)
- Mapped on IA32\_FIXED\_CTR<sub>x</sub>
- Configured using IA32\_FIXED\_CTR\_CTRL

# Reading Performance Counters

- Various tools available
  - ◆ Oprofile: Available in mainstream kernels
  - ◆ Perfctr: Requires kernel patch
  - ◆ Pfmon: Requires kernel patch
  
- Kernel Implementation using MSRs: no system calls

```
Configure the PMCs for each core
Initialize the PMC data register
Setup user readable /sys/file
Set the timer interval for reading PMCs
Start the timer
For each timer interrupt, update PMC value in /sys/file
```

# Pfmon Installation

- ❑ Download kernel patch, perfmon2 library, and pfmon from perfmon2 sourceforge page
- ❑ Build and install patched kernel with appropriate CONFIG flags enabled
- ❑ Install perfmon2 library and pfmon application
- ❑ Test pfmon for correctness: E.g., Instructions retired for NAS benchmark EP Class B

```
$ pfmon -e INSTRUCTIONS_RETIRED --system-wide ./ep.B
CPU0          46968223487 INSTRUCTIONS_RETIRED
CPU1          46584964353 INSTRUCTIONS_RETIRED
CPU2          46366578362 INSTRUCTIONS_RETIRED
CPU3          46757719728 INSTRUCTIONS_RETIRED
```

# Reading Temperature Data

- ❑ On-die thermal sensor
- ❑ Im-sensors utility
- ❑ Chip errata report sensor (in)accuracy

```
adt7490-i2c-0-2c
Adapter: SMBus I801 adapter at 3000
in0:          +1.51 V (min = +0.00 V, max = +3.31 V)
Vcore:        +0.87 V (min = +0.00 V, max = +2.99 V)
+3.3V:        +3.30 V (min = +2.96 V, max = +3.61 V)
+5V:          +5.18 V (min = +4.48 V, max = +5.50 V)
+12V:         +12.29 V (min = +0.00 V, max = +15.69 V)
in5:          +2.21 V (min = +0.00 V, max = +4.48 V)
fan1:         1106 RPM (min = 0 RPM)
fan2:         0 RPM (min = 0 RPM)
fan3:         1740 RPM (min = 0 RPM)
fan4:         0 RPM (min = 0 RPM)
temp1:        FAULT (low = -127.0°C, high = +127.0°C) ALARM
               (crit = +100.0°C, hyst = +100.0°C)
M/B Temp:     +38.5°C (low = -127.0°C, high = +127.0°C)
               (crit = +65.0°C, hyst = +61.0°C)
temp3:        +35.0°C (low = -127.0°C, high = +127.0°C)
               (crit = +65.0°C, hyst = +61.0°C)

coretemp-isa-0000
Adapter: ISA adapter
Core 0:       +36.0°C (high = +84.0°C, crit = +100.0°C)

coretemp-isa-0001
Adapter: ISA adapter
Core 1:       +37.0°C (high = +84.0°C, crit = +100.0°C)

coretemp-isa-0002
Adapter: ISA adapter
Core 2:       +32.0°C (high = +84.0°C, crit = +100.0°C)

coretemp-isa-0003
Adapter: ISA adapter
Core 3:       +36.0°C (high = +84.0°C, crit = +100.0°C)
```

# Empirical Power Measurement

- Power meter: Watts Up? PRO
  - ◆ Pros: non-intrusive, off-the-shelf, easily accessible by software
  - ◆ Cons: no isolation of CPU power, lower sampling rate, lower sensitivity to power changes
  
- Current sensors
  - ◆ Pros: higher granularity, potentially higher sampling rates
  - ◆ Cons: custom hardware, relatively complex software interfaces
  
- Digital ammeter
  
- Power management chip output

# Watts Up? PRO

- ❑ Max sampling rate:  
1 sample/sec
- ❑ Accuracy: +/- 1.5%
- ❑ Linux interface: wattsup-daemon
  - ◆ Uses CPAN Perl module
  - ◆ Talks to meter over USB using FTDI serial driver



# Empirical Per-Core Power Measurement

- ❑ Required for training set
- ❑ Per-core power =  $(\text{system power} - \text{uncore power}) / \# \text{cores}$
- ❑ Uncore power = idle system power – idle CPU power
- ❑ Idle CPU power data
  - ◆ Rely on published results
  - ◆ Measure on mother board
  - ◆ Approximate by disconnecting CDROM, DIMMs, HDD, graphics card, NIC and measure the drop in power consumption
- ❑ Assume uncore power remains constant during benchmark runs

# Microbenchmarks

- ❑ Application independent
- ❑ Stress different components of microarchitecture
- ❑ Maintain comprehensive mix of instructions
- ❑ Establish correlation between counters and power consumption
- ❑ Single threaded – one instance runs on each core

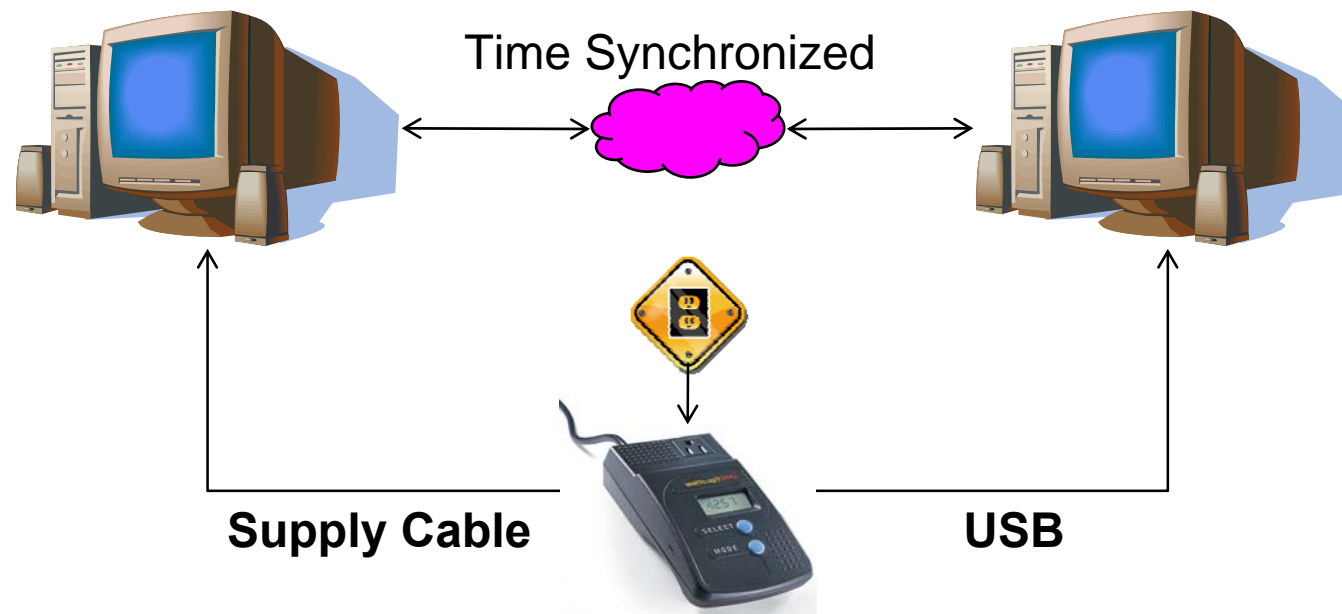
# Setup

## Target Machine

- Runs  $\mu$ benchmarks and test benchmarks
- Collects counter data using pfmon

## Test Machine

- Runs wattsup-daemon
- Collects samples from power meter



# Collecting Correlation Data

- ❑ Run microbenchmarks collecting data for multiple counters
- ❑ Synchronize counter data and power samples using spike in power consumption and time stamp
- ❑ Correlate counter and power values using spearman's rank correlation with R package

## Rscript

```
data <- read.table("ubench.frm", header=FALSE)
cor.test(data$V1, data$V5, method="spearman")
cor.test(data$V2, data$V5, method="spearman")
cor.test(data$V3, data$V5, method="spearman")
cor.test(data$V4, data$V5, method="spearman")
```

# Correlation Results

INSTRUCTIONS_RETIRED	0.81
LAST_LEVEL_CACHE_REFERENCES	0.36
LAST_LEVEL_CACHE_MISSES	0.41
BRANCH_INSTRUCTIONS_RETIRED	0.68
BACLEAR:BAD_TARGET	-0.11
BACLEAR:CLEAR	-0.38
BACLEAR_FORCE_IQ	0.03
BR_INST_DECODED	0.69
BR_INST_EXEC:ANY	0.69
BR_INST_RETIRED:ALL_BRANCHES	0.69
FP_ASSIST:ALL	NA
FP_COMP_OPS_EXE:SSE_FP	0.65
...	...
...	...

# Counter Categorization

- Separate counters into categories
- To make unbiased model

Memory Operations	$\rho$
MEM_INST_RETIRED:LOADS	0.81
UOPS_EXECUTED:PORT2_CORE	0.81
UOPS_EXECUTED:PORT234_CORE	0.74
MEM_INST_RETIRED:STORES	0.74
LAST_LEVEL_CACHE_MISSES	0.41
LAST_LEVEL_CACHE_MISSES	0.41

Total Instructions	$\rho$
UOPS_EXECUTED:PORT1	0.84
UOPS_ISSUED:ANY	0.81
UOPS_EXECUTED:PORT015	0.81
INSTRUCTIONS_RETIRED	0.81
UOPS_EXECUTED:PORT0	0.81
UOPS_RETIRED:ANY	0.78

Stalls	$\rho$
ILD_STALL:ANY	0.45
RESOURCE_STALLS:ANY	0.44
RAT_STALLS:ANY	0.40
UOPS_DECODED:STALL_CYCLES	0.25

Floating Point Operations	$\rho$
FP_COMP_OPS_EXE:SSE_FP	0.65
FP_COMP_OPS_EXE:X87	0.04

# Counter Selection

- Prefer counter masks that cover more comprehensive events

Memory Operations	$\rho$
MEM_INST_RETIRED:LOADS	0.81
UOPS_EXECUTED:PORT2_CORE	0.81
UOPS_EXECUTED:PORT234_CORE	0.74
MEM_INST_RETIRED:STORES	0.74
LAST_LEVEL_CACHE_MISSES	0.41
LAST_LEVEL_CACHE_REFERENCES	0.36

Total Instructions	$\rho$
UOPS_EXECUTED:PORT1	0.84
UOPS_ISSUED:ANY	0.81
UOPS_EXECUTED:PORT015	0.81
INSTRUCTIONS_RETIRED	0.81
UOPS_EXECUTED:PORT0	0.81
UOPS_RETIRED:ANY	0.78

Stalls	$\rho$
ILD_STALL:ANY	0.45
RESOURCE_STALLS:ANY	0.44
RAT_STALLS:ANY	0.40
UOPS_DECODED:STALL_CYCLES	0.25

Floating Point Operations	$\rho$
FP_COMP_OPS_EXE:X87	0.65
FP_COMP_OPS_EXE:SSE_FP	0.04

# Correlation Matrix

Counter	Symbol
UOPS_EXECUTED:PORT234_CORE	A
LAST_LEVEL_CACHE_MISSES	B
FP_COMP_OPS_EXE:X87	C
RESOURCE_STALLS:ANY	D
UOPS_ISSUED:ANY	E
UOPS_EXECUTED:PORT015	F
INSTRUCTIONS_RETIRED	G
UOPS_RETIRED:ANY	H

	A	B
E	0.97	0.14
F	0.88	0.2
G	0.91	0.12
H	0.98	0.08

**MEM vs INSTR**

	C
E	0.44
F	0.41
G	0.49
H	0.43

**FP vs INSTR**

	D
E	0.25
F	0.30
G	0.23
H	0.21

**STALL vs INSTR**

# Counter Selection

Memory Operations	$\rho$
MEM_INST_RETIRED:LOADS	0.81
UOPS_EXECUTED:PORT2_CORE	0.81
UOPS_EXECUTED:PORT234_CORE	0.74
MEM_INST_RETIRED:STORES	0.74
LAST_LEVEL_CACHE_MISSES	0.41
LAST_LEVEL_CACHE_REFERENCES	0.36

Total Instructions	$\rho$
UOPS_EXECUTED:PORT1	0.84
UOPS_ISSUED:ANY	0.81
UOPS_EXECUTED:PORT015	0.81
INSTRUCTIONS_RETIRED	0.81
UOPS_EXECUTED:PORT0	0.81
UOPS_RETIRED:ANY	0.78

Stalls	$\rho$
ILD_STALL:ANY	0.45
RESOURCE_STALLS:ANY	0.44
RAT_STALLS:ANY	0.40
UOPS_DECODED:STALL_CYCLES	0.25

Floating Point Operations	$\rho$
FP_COMP_OPS_EXE:X87	0.65
FP_COMP_OPS_EXE:SSE_FP	0.04

# Model Formation

- Rerun microbenchmarks with selected PMCs to gather counter, power, and temperature data

MEM	INSTR	FP	STALLS	TEMP	POW
5715	6076259299	2290	676985555	65	97.6
72	6106099565	1	690574075	67	128.3
9592	6081882226	1570	688073989	65.25	128.2
3772	6096609181	1544	594153043	67.25	127.9
1415	4931385249	804	1235952822	67.25	128.1
3269	4801320073	1350	1291138418	66.75	127.9
1503	4808931086	1361	1294278197	67.25	127.8
7200	4791204466	2403	1291382390	67.5	127.7
4790	4794755513	2704	1288877215	67.5	127.6
828	4814767108	108	1296655935	68	127.6
995	4810269638	128	1296495400	68	127.5
...	...	...	...	...	...

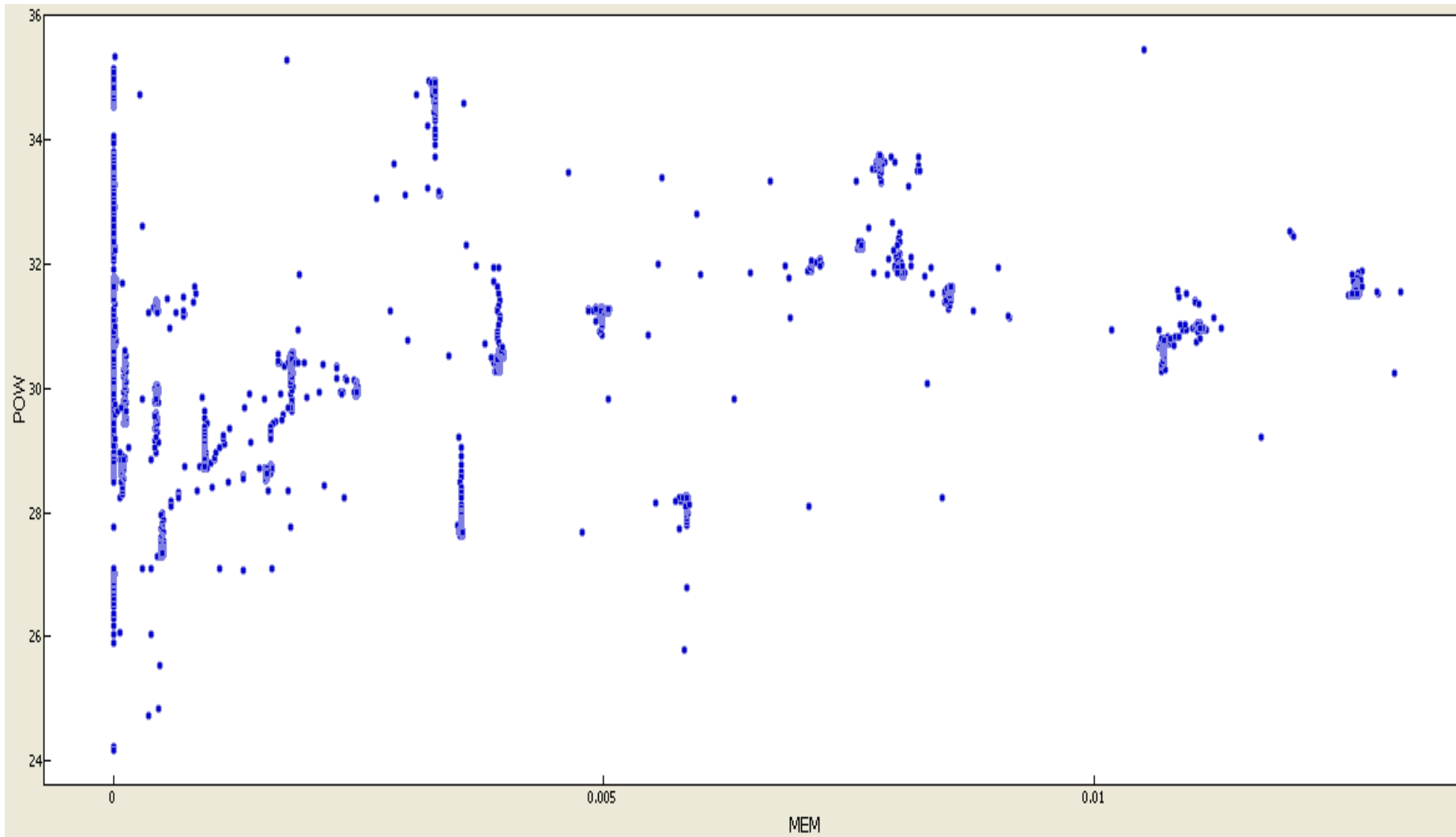
# Model Formation

## □ Normalize values for training

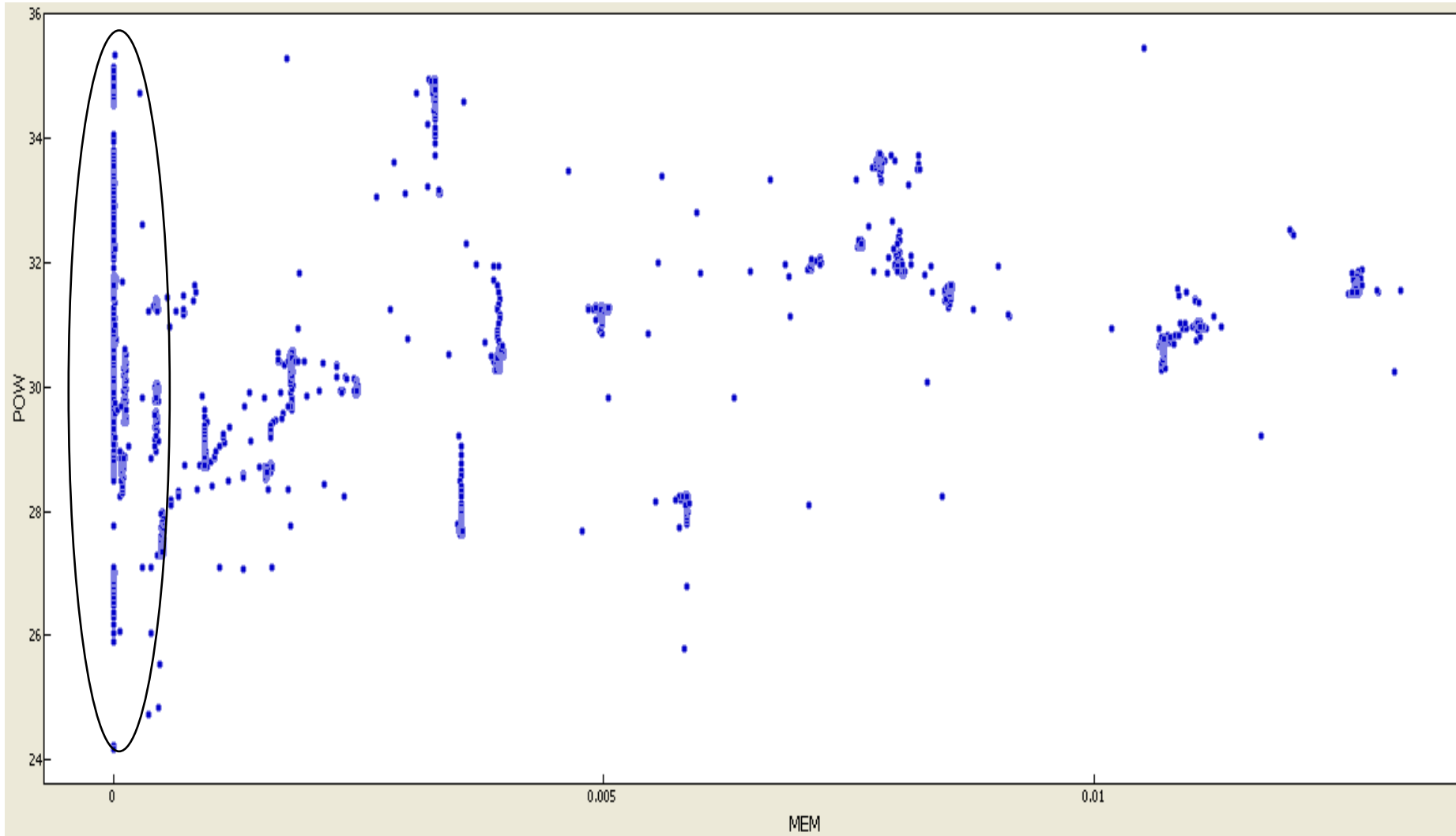
- ◆  $r_{\text{PMC}} = \text{PMC}/\text{Cycles}$
- ◆  $T = \text{TEMP} - \text{TEMP}_{\text{IDLE}}$
- ◆  $P_{\text{CORE}} = (\text{POW} - \text{POW}_{\text{IDLE}})/\#\text{Cores}$

$r_{\text{MEM}}$	$r_{\text{INSTR}}$	$r_{\text{FP}}$	$r_{\text{STALLS}}$	$T$	$P_{\text{CORE}}$
3.28E-06	2.0786	5.37E-07	0.2352	35.25	29.55
1.29E-06	2.0836	5.28E-07	0.2031	37.25	29.475
4.84E-07	1.6854	2.75E-07	0.4224	37.25	29.525
1.12E-06	1.6409	4.61E-07	0.4413	36.75	29.475
5.14E-07	1.6435	4.65E-07	0.4423	37.25	29.45
2.46E-06	1.6375	8.21E-07	0.4413	37.5	29.425
1.64E-06	1.6387	9.24E-07	0.4405	37.5	29.4
2.83E-07	1.6455	3.69E-08	0.4432	38	29.4
3.40E-07	1.644	4.37E-08	0.4431	38	29.375
1.04E-06	1.6377	5.51E-07	0.4404	38.5	29.375
5.06E-06	1.6394	6.51E-07	0.4418	38.5	29.375
...	...	...	...	...	...

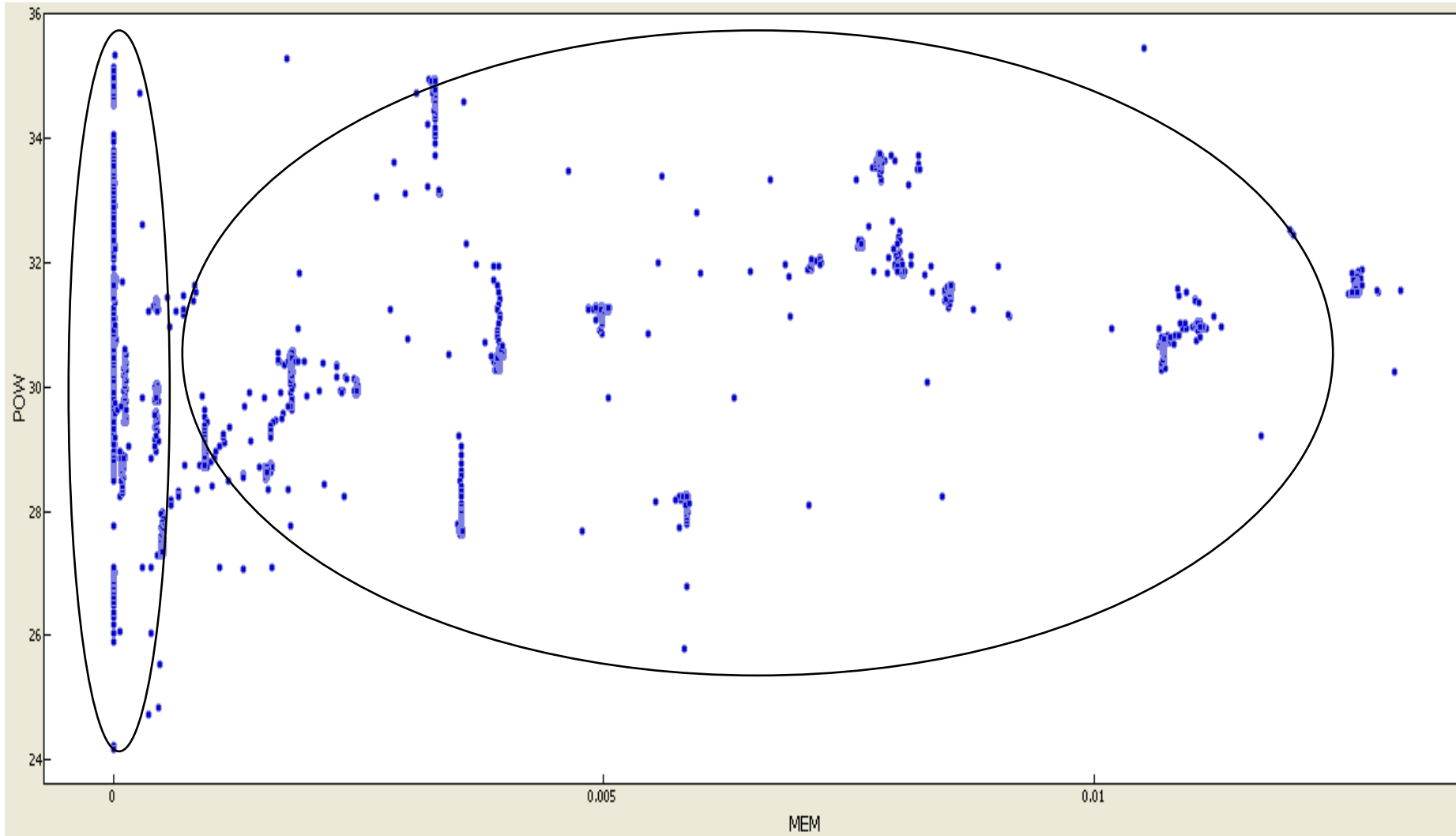
# Piecewise Model



# Piecewise Model



# Piecewise Model



# Optional Transformations

- ❑ Transformations can be applied to some PMCs for better fit
- ❑ Accommodate non-linear relationship between counters and power
- ❑ Mostly logarithmic
- ❑ Example on Core i7:

$$P_{core} = \begin{cases} \text{if } r_{MEM} < BREAK, & F_1(r_{MEM}, r_{INSTR}, \log(r_{FP}), r_{STALL}, T) \\ \text{else,} & F_2(r_{MEM}, \log(r_{INSTR}), \log(r_{FP}), r_{STALL}, T) \end{cases}$$

# Multiple Linear Regression

## □ Use regress tool from STAT package

```
cat ubench.norm | dm "IF x1 < $break THEN INPUT else NEXT" \  
                | dm s6 x1 x2 log(x3) x4 x5 \  
                | regress -e -p POW MEM INSTR FP STALL TEMP \  
> ubench.part1.reg
```

### Output Equation 1:

$$P_{\text{CORE}} = (r_{\text{MEM}} * 0) + (r_{\text{INSTR}} * 3.923) + (\log(r_{\text{FP}}) * 0.031) + (r_{\text{STALL}} * 3.359) + (T * 0.188) + 14.076$$

```
cat ubench.norm | dm "IF x1 >= $break THEN INPUT else NEXT" \  
                | dm s6 x1 log(x2) log(x3) x4 x5 \  
                | regress -e -p POW MEM INSTR FP STALL TEMP \  
> ubench.part2.reg  
mv regress.eqn regress.part2.eqn
```

### Output Equation 2:

$$P_{\text{CORE}} = (r_{\text{MEM}} * 692.930) + (\log(r_{\text{INSTR}}) * 2.765) + (\log(r_{\text{FP}}) * 0.121) + (r_{\text{STALL}} * -1.866) + (T * 0.199) + 21.782$$

# Regression Output : Things to Check

- R-squared
- t statistic
- p values

Significance test for prediction of POW

Multi-R	R-Squared	SEest	F(5,657)	prob (F)
0.9391	0.8819	0.5703	981.2085	0.0000

Significance test(s) for predictor(s) of POW

Predictor	beta	b	Rsq	se	t(657)	p
X1	0.0794	27293.0473	0.1987	5149.5628	5.3001	0.0000
X2	0.9047	3.5767	0.7725	0.1111	32.1845	0.0000
X3	0.1432	0.0456	0.2241	0.0048	9.4107	0.0000
X4	0.3099	2.6651	0.7283	0.2212	12.0469	0.0000
X5	0.3826	0.1897	0.3849	0.0085	22.3828	0.0000

# Regression Output : Things to Check

- R-squared
- t statistic
- p values

Significance test for prediction of POW

Multi-R	R-Squared	SEest	F(5,657)	prob (F)
0.9391	0.8819	0.5703	981.2085	0.0000

Significance test(s) for predictor(s) of POW

Predictor	beta	b	Rsq	se	t(657)	p
X1	0.0794	27293.0473	0.1987	5149.5628	5.3001	0.0000
X2	0.9047	3.5767	0.7725	0.1111	32.1845	0.0000
X3	0.1432	0.0456	0.2241	0.0048	9.4107	0.0000
X4	0.3099	2.6651	0.7283	0.2212	12.0469	0.0000
X5	0.3826	0.1897	0.3849	0.0085	22.3828	0.0000

# Model Validation

- ❑ Comparison of estimated and measured power consumption
- ❑ Three benchmark suites (45 benchmarks)
  - ◆ Single and Multi-threaded
  - ◆ Floating point and Integer
- ❑ Use `dm` to plot the empirical against predicted value using `regress.eqn` output files from `regress`

```
cat test.norm | dm "IF x1 < $break THEN INPUT else NEXT" \  
              | dm s6 x1 x2 log(x3) x4 x5 \  
              | dm Eregress.part1.eqn > test.part1.out
```

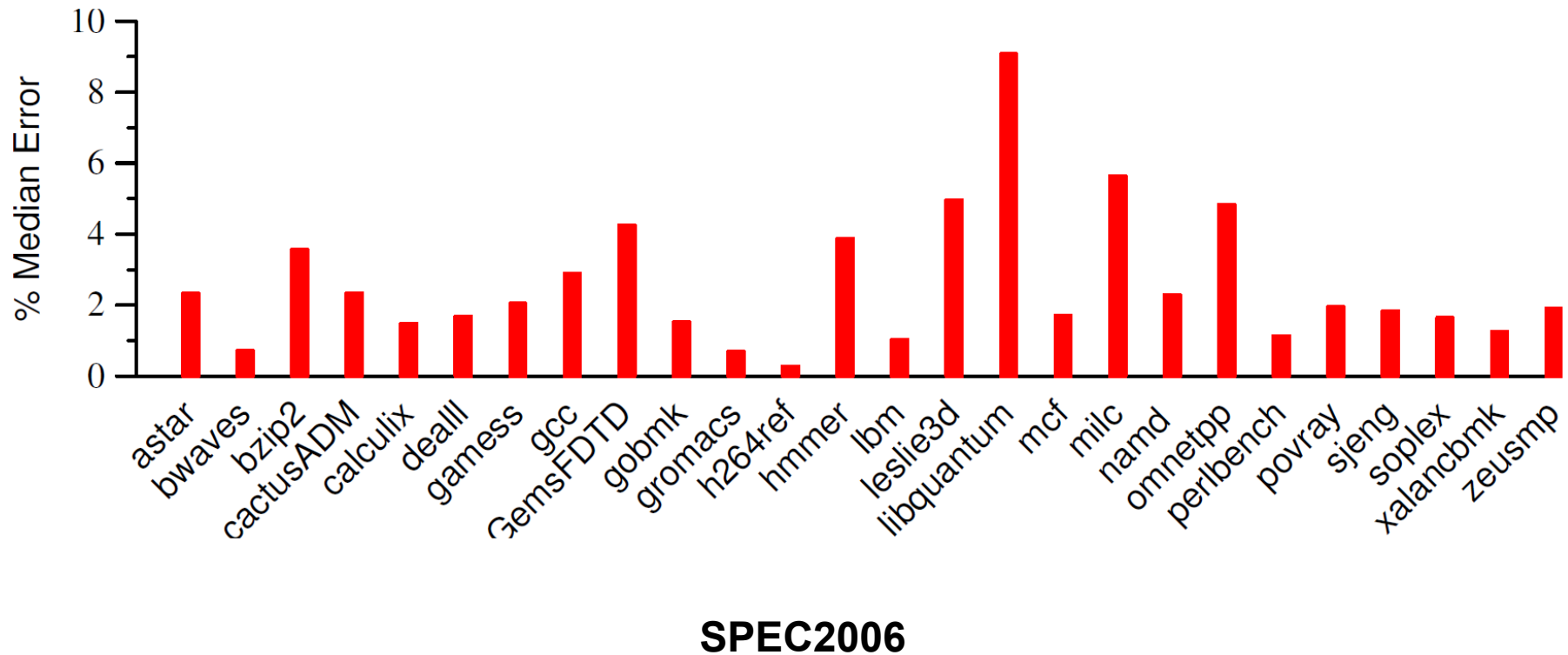
```
cat test.norm | dm "IF x1 >= $break THEN INPUT else NEXT" \  
              | dm s6 x1 log(x2) log(x3) x4 x5 \  
              | dm Eregress.part2.eqn > test.part2.out
```

# Model Validation

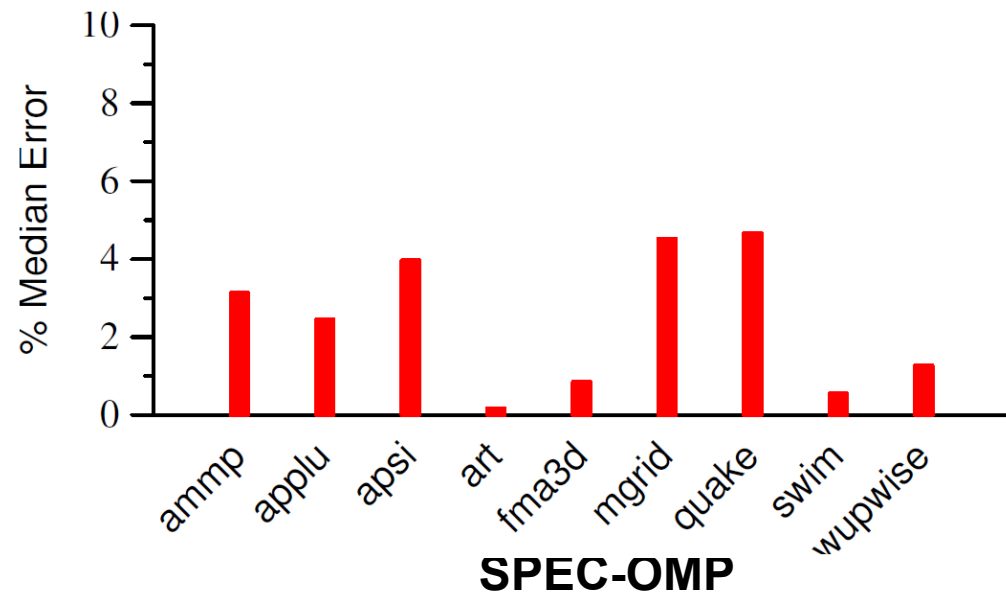
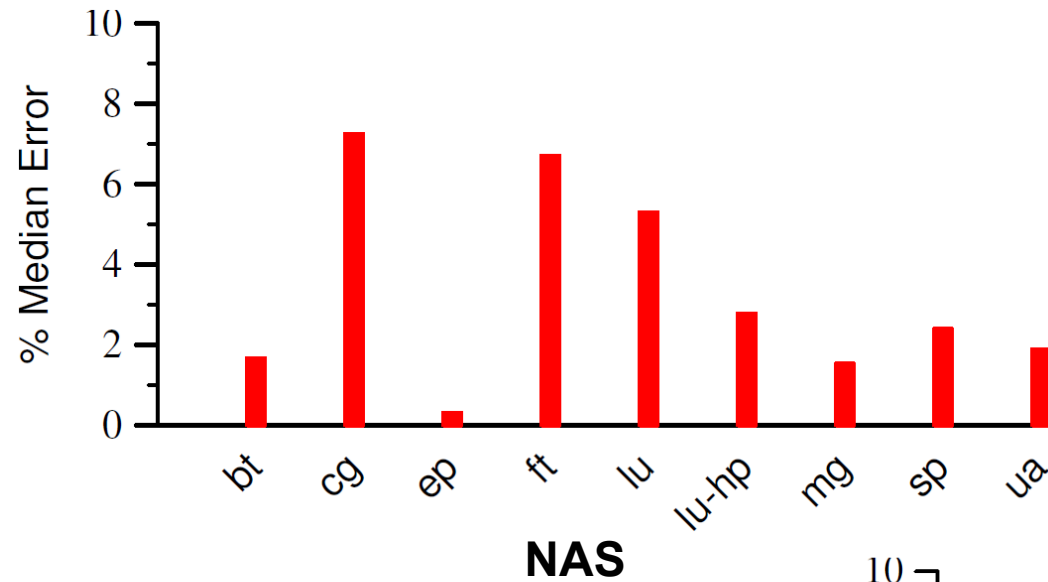
## □ Validation output example: bwaves

Actual	Predicted	% Error
32.70	30.83	5.73
32.73	31.69	3.15
32.80	32.74	0.20
32.70	30.04	8.12
32.80	31.33	4.48
32.80	32.92	0.37
32.68	29.77	8.88
32.80	32.23	1.73
32.95	33.45	1.52
32.85	30.57	6.93
32.88	31.01	5.66
34.20	33.68	1.52
34.20	33.75	1.32
32.10	30.51	4.95
...	...	...
...	...	...

# Core i7 Results



# Core i7 Results



# Website

- [www.cse.chalmers.se/research/group/Fusion/pmc\\_demo](http://www.cse.chalmers.se/research/group/Fusion/pmc_demo)
- README
- scripts (echo.bz2)
  - ◆ Watts Up? PRO
  - ◆ pfmon
  - ◆ temperature collection
  - ◆ regression
  - ◆ validation

**Thank You!**

**Questions?**