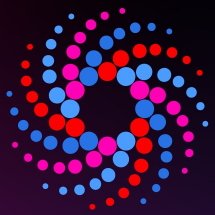


SC22
Dallas, TX | hpc accelerates.

Performance Analysis with Unified Hardware Counter Metrics

Brian J Gravelle
William D Nystrom
Boyana Norris

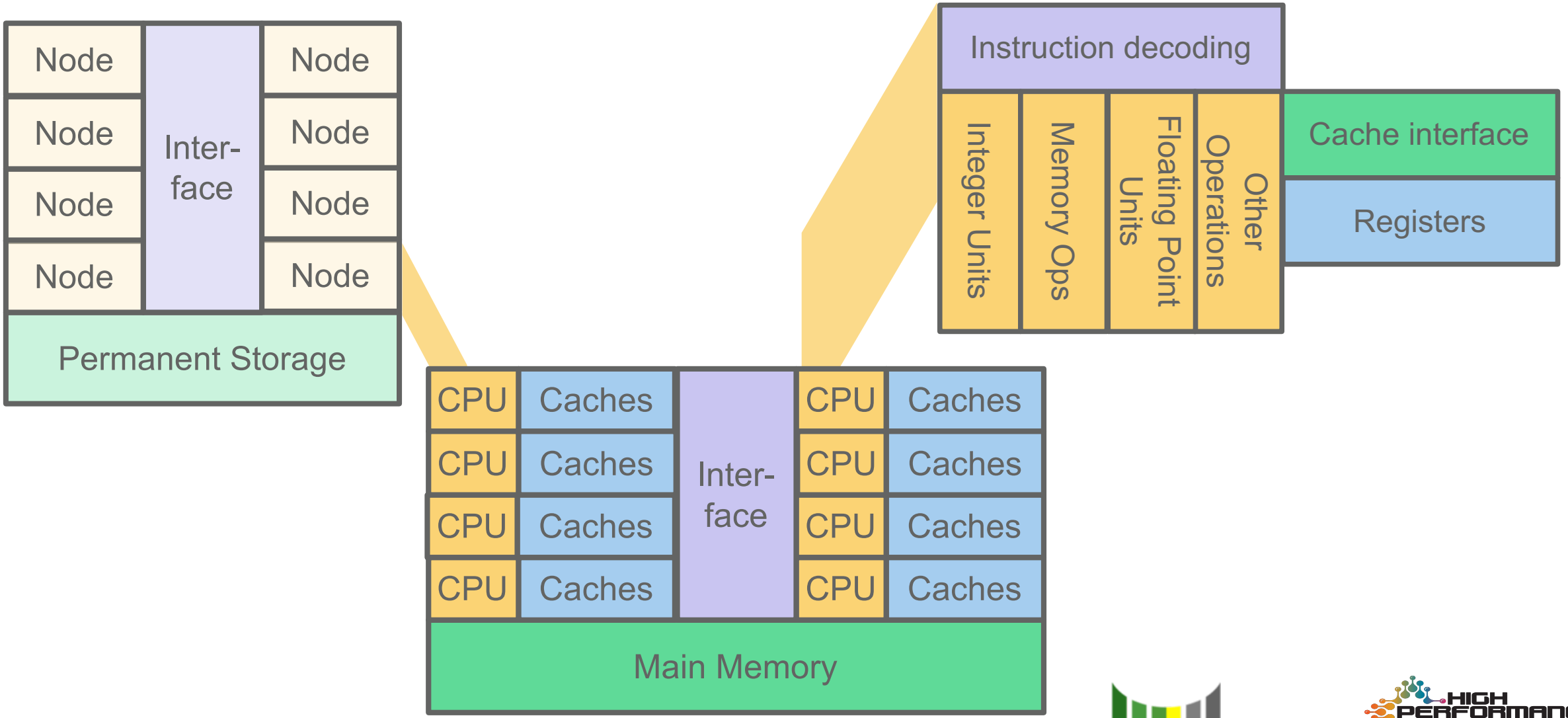
What is performance analysis?

How does an application make use of the Supercomputer?

```
!$OMP DO PRIVATE(right_flux,left_flux,top_flux,bottom_flux,total_flux,min_cell_volume, &
!$OMP             energy_change,recip_volume,volume_change_s)
DO k=y_min,y_max
  DO j=x_min,x_max

    left_flux=  (xarea(j  ,k  )*(xvel0(j  ,k  )+xvel0(j  ,k+1)                    &
      +xvel0(j  ,k  )+xvel0(j  ,k+1)))*0.25_8*dt*0.5
    right_flux= (xarea(j+1,k  )*(xvel0(j+1,k  )+xvel0(j+1,k+1)                    &
      +xvel0(j+1,k  )+xvel0(j+1,k+1)))*0.25_8*dt*0.5
    bottom_flux=(yarea(j  ,k  )*(yvel0(j  ,k  )+yvel0(j+1,k  )                    &
      +yvel0(j  ,k  )+yvel0(j+1,k  )))*0.25_8*dt*0.5
    top_flux=   (yarea(j  ,k+1)*(yvel0(j  ,k+1)+yvel0(j+1,k+1)                    &
      +yvel0(j  ,k+1)+yvel0(j+1,k+1)))*0.25_8*dt*0.5
    total_flux=right_flux-left_flux+top_flux-bottom_flux

    volume_change_s=volume(j,k)/(volume(j,k)+total_flux)

    min_cell_volume=MIN(volume(j,k)+right_flux-left_flux+top_flux-bottom_flux &
      ,volume(j,k)+right_flux-left_flux                      &
      ,volume(j,k)+top_flux-bottom_flux)

    recip_volume=1.0/volume(j,k)

    energy_change=(pressure(j,k)/density0(j,k)+viscosity(j,k)/density0(j,k))*total_flux*recip_volume

    energy1(j,k)=energy0(j,k)-energy_change

    density1(j,k)=density0(j,k)*volume_change_s

  ENDDO
ENDDO
!$OMP END DO
```

And how can we make it run faster?

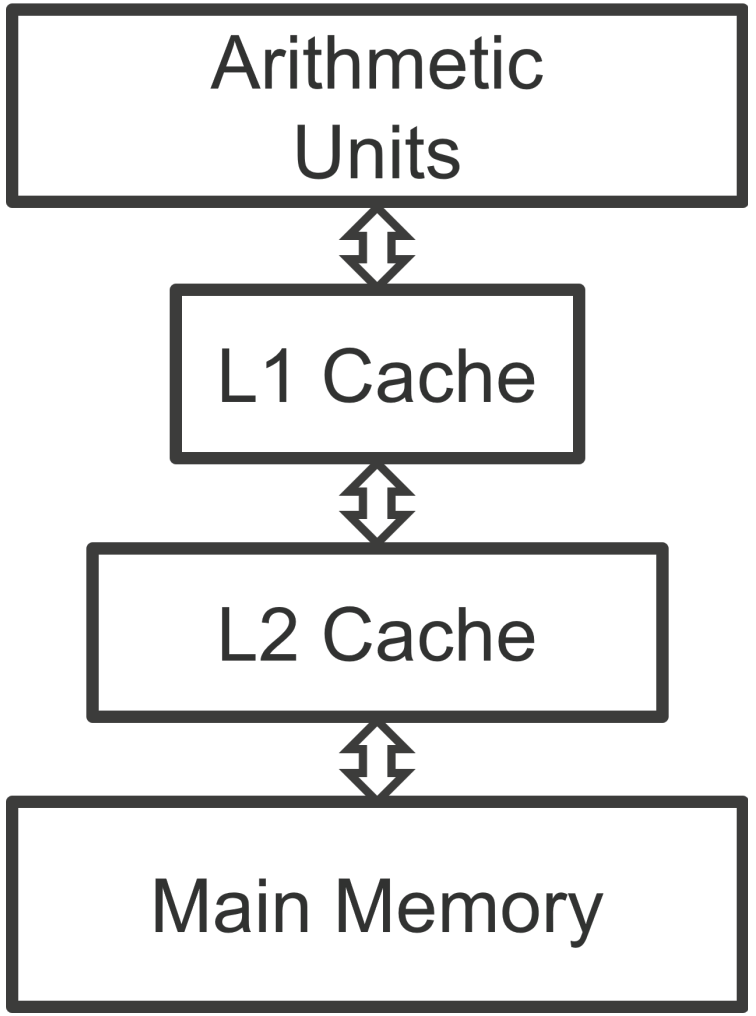
Scope of Our Research



Hardware Counter Analysis

- Counters in CPUs
- Collect counts of various events
- Provide insight into the performance

- Different between each vendor
- Tied to microarchitecture features



Current Counter Challenges

- Sometimes tied to specific CPUs
  - Microarchitecture features
  - Available Counters

- Difficulty applying information to application
  - Idealized peaks can be misleading
  - Measurements of specific CPU features

Hardware Counter Analysis

- Example measuring Main Memory accesses

- A64FX – measured on each core

```
L2D_CACHE_REFILL
L2D_CACHE_WB
```

- Cascadelake – measured on each memory controller

```
skx_unc_imc[0...6]::UNC_M_CAS_COUNT:RD:cpu=[0,25]
```

Main hypothesis

Hardware counters from different CPU types can be used to measure consistent performance metrics which provide information about how a computational kernel uses the CPU microarchitecture.

Additional Hardware Counter Metrics

- Are there additional metrics to provide more performance information to the user?

- Metrics that:
  - Provide information about common architecture features
  - Inform the user about application performance
  - Can be measured on both systems

Overall performance metrics

- Time – execution time of the kernel
- flops/s – floating point operations per second
- IPC – Instructions per cycle

Data movement metrics

- LS Bytes – Bytes moved based on the count of load and store instructions
- [L2, L3] Bytes – Bytes moved by cache misses
- Mem Bytes – Bytes moved to and from main memory

Cache Efficiency

- [L1, L2, L3] Miss Rates – Cache misses per total accesses
- [L2, L3] Bytes / LS Bytes – Bytes moved at a cache layer per LS Bytes
- Mem Bytes / LS Bytes – Bytes moved to and from the main memory per LS Bytes
- LD Ins / ST ins – loads per stores

Computation

- flops – number of floating point operations used

- flops / fp ins – flops per floating point instruction to measure the amount of SIMD use

Computation Data Rates

- Arithmetic Intensity – flops per LS Bytes
- flops / [LD, ST] Bytes – total flops per just the load bytes or just the store bytes
- flops / [LD, ST] Ins – total flops per the load instructions or the store instructions

Examples

- Nbody Kernel
  - “textbook” algorithm
  - Scalar vs SIMD operations
- XS Bench – mini-application
  - With and without sorting
- VPIC – particle in cell simulation

- Fujitsu A64FX and Intel Cascadelake

Systems

- Fujitsu A64FX
  - ARM CPU with 512 bit SVE
  - 48 Cores, 32 GB of HBM
- Intel Cascade Lake Xeon
  - x86 CPU with AVX 512 bit SIMD
  - 48 Cores, 188 GB of DDR

NBody

- Fewer bytes moved -> higher AI
  - More exaggerated with stores
- Increased L1 Miss Rate
- Use of SIMD operations

	A64FX Scalar	A64FX SIMD	CLake Scalar	CLake SIMD
LS Bytes (GB)	20,170	2,202	16,191	2,581
L1 Miss Rate (%)	0.159	3.12	0.098	50.0
Load ins / Store ins	5.65	3750	2.26	80.8
AI (flops / LS Bytes)	0.068	4.56	0.085	0.53
flops / load bytes	0.08	4.56	0.12	0.54
flops / store bytes	0.45	32,700	0.28	43.56
flops / fp ins	0.95	6.08	2.26	7.9997

XSBench

- LS Bytes stays within ~5%
- Cache miss rates fall dramatically
- L2 Bytes/ LS Bytes doesn't change much
- L3 and Mem Bytes/ LS Bytes do change

	A64FX			Cascadelake		
	event	event opt 1	event opt 2	event	event opt 1	event opt 2
LS Bytes (GB)	201	190	194	234	226	230
L1 Miss Rate (%)	1.7	1.8	1.9	24.1	16	18
L2 Miss Rate (%)	43	2.0	0.98	88	9.0	9.1
L2 Bytes / LS Bytes	4.54	3.29	3.61	0.92	0.61	0.66
L3 Bytes / LS Bytes	NA	NA	NA	0.80	0.055	0.60
Mem Bytes / LS Bytes	3.49	0.066	0.036	0.55	0.050	0.032

VPIC

- Significant reduction in amount of data moved
- Cache performance appear to be worse

	AoS A64FX	AoSoA cell A64FX	AoS CLake	AoSoA cell CLake
Time (s)	23.5	9.55	21.1	15.3
IPC	0.71	0.55	0.43	0.20
LS Bytes (GB)	13,800	5,220	22,400	3,900
flops / fp ins	23.4	23.4	15.9…	15.9…
L1 Miss Rate (%)	4.1	9.7	9.1	41.0
L2 Miss Rate (%)	26	27	134	165
L2 Bytes / LS Bytes	0.49	1.84	0.091	0.41
L3 Bytes / LS Bytes	NA	NA	0.12	0.69
Mem Bytes / LS Bytes	0.33	1.13	0.18	0.75

Conclusions

- ✓ Counter metrics can be unified across systems
- ✓ Metrics can provide information which users can reason about in conjunction with application changes

Future research questions

- Are there portable metrics for instruction mix?
- Is this method portable to other CPUs?
- Is this method portable to GPUs and other accelerators?

Questions?

Over 70 years at the forefront of supercomputing

Contact: Brian J Gravelle, gravelle@lanl.gov

Data Movement – LS Bytes

- LS Bytes are computed from load store instructions
- A64FX has counters for different data sizes
- Cascadelake doesn’t differentiate sizes
  - We assume same ratio as floating point instructions

Data Movement – Cache and Memory

- Cache Level bytes
  - Based on misses from the level above
  - Multiply number of misses by the cache size

- Memory Bytes
  - A64FX – reads and writes from L2 multiplied by cache line
  - Cascadelake – sum of counters on the memory controllers

Cache Efficiency Metrics

- Miss Rates

$$AI = \frac{Acccesses\ to\ Cache\ Level}{Misses\ at\ that\ Level}$$

  - L1 accesses are load and store instructions
  - Other levels are misses from the level above

- Byte ratios are calculated as written

Arithmetic Intensity

- Ratio of FLOPs to Bytes

$$AI = \frac{FLOPs}{LS\ Bytes}$$

- Can be main memory or other Byte metrics

- Other computation data rates are similar
  - Change denominator to appropriate value

Over 70 years at the forefront of supercomputing