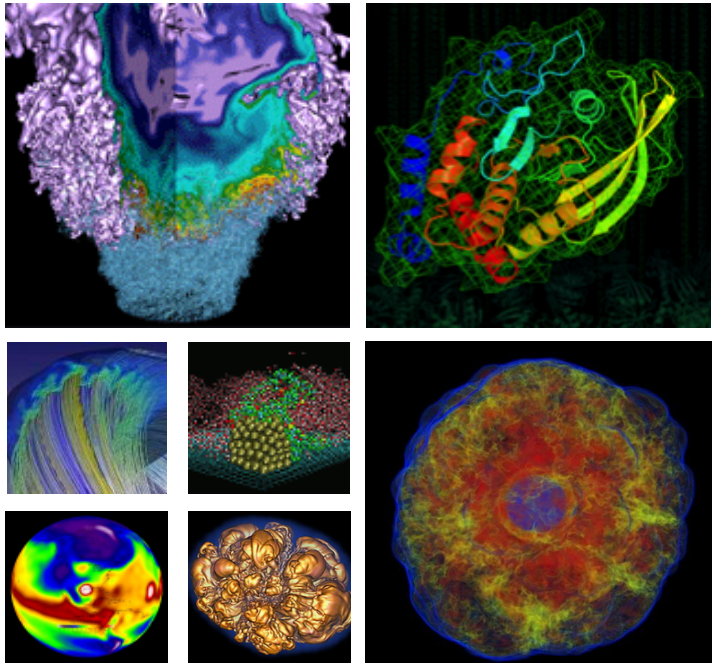
Toward Automated Application Profiling on Cray Systems



Charlene Yang, Brian Friesen, Thorsten Kurth, Brandon Cook
NERSC at LBNL
Samuel Williams
CRD at LBNL

U.S. DEPARTMENT OF ENERGY | Office of Science

I have a dream.. M.L.K.

Collect performance data:

- low **overhead**: little to no overhead to codes' native runtime
- easy to **deploy**: no instrumentation, no extra kernels/modules, no code modification
- as **much useful** information as possible
- **accurate** information
- on **mass** users: run at background

HPC users: optimize their codes, prioritize development efforts

HPC facilities: better understand their user base, guide future procurements

Automated, passive, mass performance data collection!

I have a plan

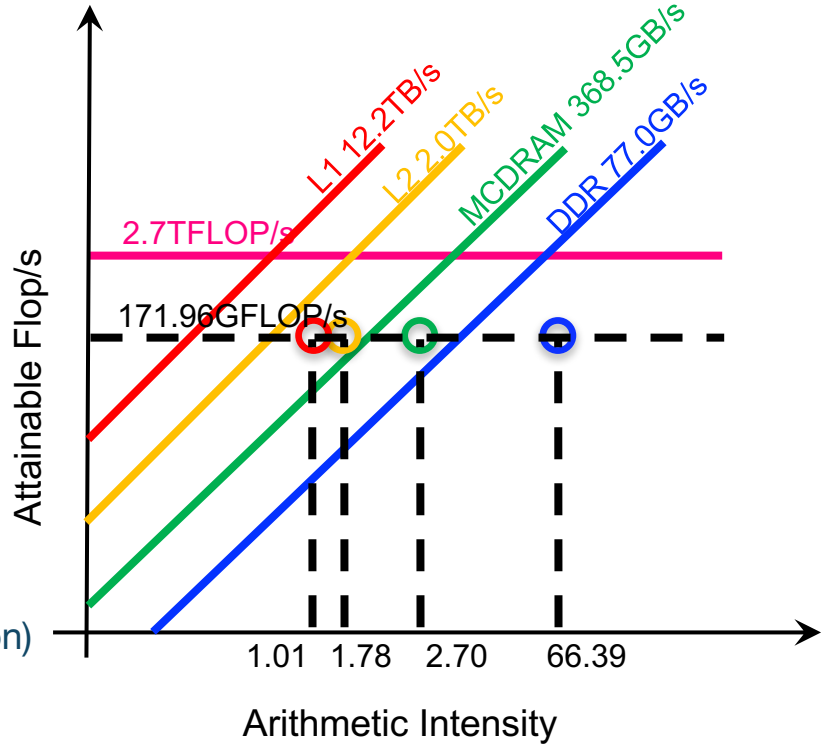
- **6 tools**: **CrayPat** (perftools-lite), **LIKWID** from Regional Computing Center Erlangen, **IPM** from Lawrence Berkeley National Laboratory, Intel **VTune** Amplifier, Intel **SDE** Emulator, and **perf** from the Linux Kernel

- **3 applications**: **HPGMG**, **Nyx**, and **Tiramisu**

	HPGMG	Nyx	Tiramisu
Scale	Kernel	Full Application	Full Application
Lines of Code	20K	2M	3M
Language	C	C/C++, Fortran	Python, C/C++
Parallelism	MPI, OMP, CUDA	MPI, OMP	Python, MKL
Domain	HPC	HPC	Deep Learning
	PDE Solvers	PDE/ODE Solvers	TensorFlow
	Geometric Multigrid	Mesh Refinement	Image Processing

I have a plan

- **5 metrics**:
  - Runtime
  - GFLOP's
  - Memory bandwidth (DRAM/LLC/L2/L1)
  - Memory high watermark
  - Vectorization efficiency
- **4 qualities**:
  - Usability (instrumentation/recompilation/*etc*)
  - Overhead
  - Actionability (amount of actionable information)
  - Accuracy (accuracy of information)



Execute our plan

- Run on **Cori-KNL** at NERSC, LBNL
- Compiled with **Cray** wrappers for **Intel** 18.0.1.163
- Linked **dynamically** *

- **HPGMG** and **Nyx**: 8 MPI ranks and 8 OpenMP threads per rank
- **Tiramisu**: 2 Python processes and 33 threads per process

- Fixed CPU **frequency** 1.401 GHz to avoid timing variation caused by clock difference

Make sure compile time/runtime environment is the same

*except for Nyx when it's profiled with IPM

Results

3 dimensions to our results: metrics (5), applications (3) and tools (6)

How to group these data-points?

- One metric per figure
- One color per application: **HPGMG**, **Nyx**, **Tiramisu**
- One marker per tool: ✳**Baseline, + CrayPat,** ○ **LIKWID,** ◇ **IPM,** ◁ **VTune,** ▷ **SDE,** △ **Perf**

We would like to compare

- different **tools**, same application, same metric
- same tool, different **applications**, same metric
- same tool, same application, different **metrics**

Missing data-points: not available from tools, analytically intractable (e.g. baseline)

Metric 1: Runtime

Low overhead: CrayPat, LIKWID, IPM and **Perf**

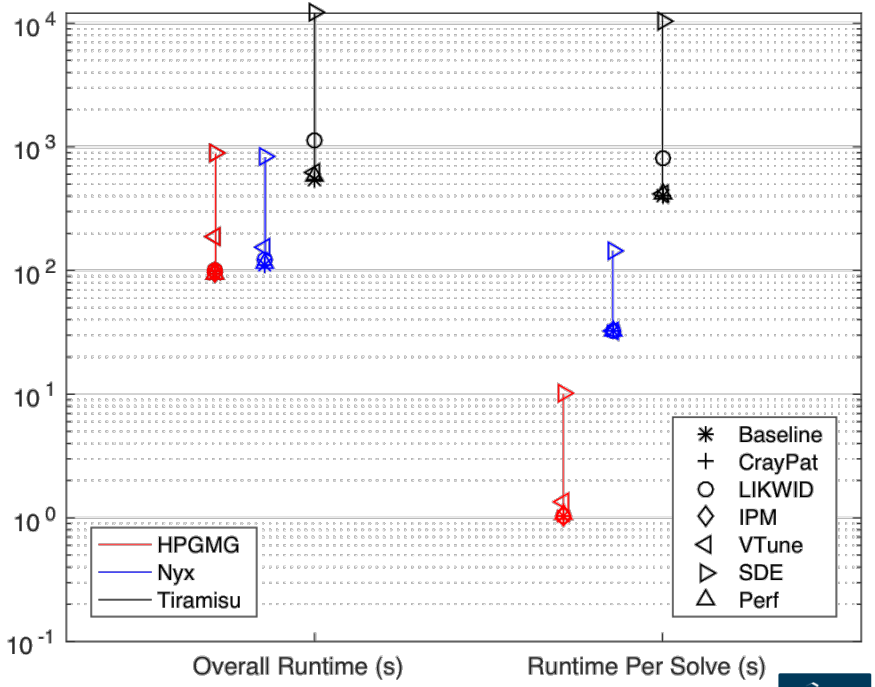
High overhead: SDE and VTune

Tiramisu is not benchmarked with CrayPat/IPM

H: HPGMG; N: Nyx; T: Tiramisu

O: Overall Runtime; P: Per Solve Runtime

	CrayPat	LIKWID	IPM	VTune	SDE	Perf
H-O	1.093	1.091	1.043	2.057	9.56	1.001
H-P	1.023	1.011	1.014	1.306	9.84	1.054
N-O	1.004	1.109	0.975	1.403	7.499	1.043
N-P	1.012	1.001	0.915	1.016	4.464	1.015
T-O	-	2.117	-	1.135	22.281	1.078
T-P	-	2	-	1.012	25.683	1.027



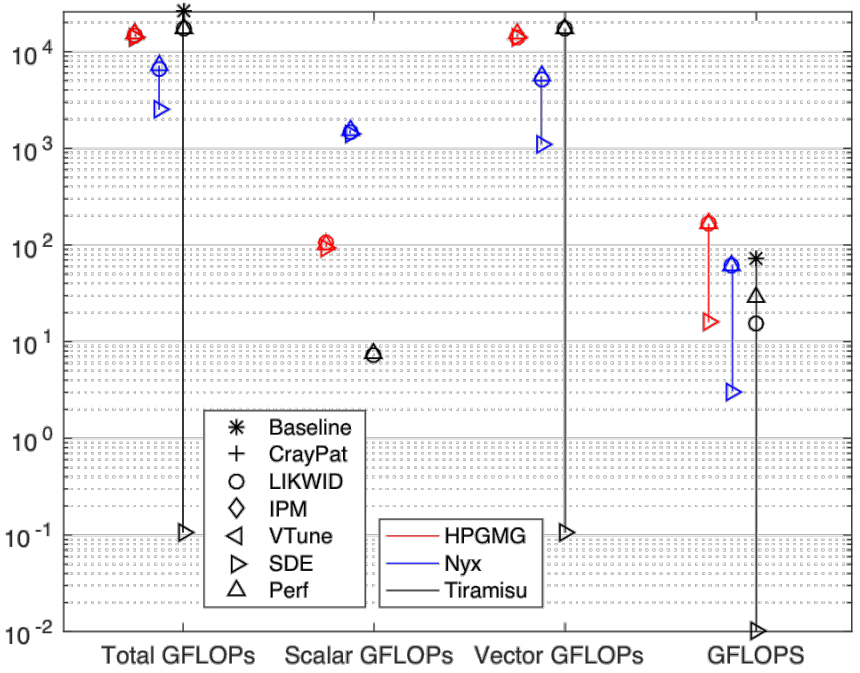
Metric 2: GFLOP's and GFLOP/s

- Count vs Rate

- **LIKWID** and **Perf**
- **SDE**

- KNL doesn't have a FLOP's counter, but...
  - *UOPS_RETIRED.PACKED_SIMD*
  - *UOPS_RETIRED.SCALAR_SIMD*
- Tools work under different mechanisms

- **HPGMG**: **consistent** results
- **Nyx/Tiramisu**: different ***Vector GFLOP's***



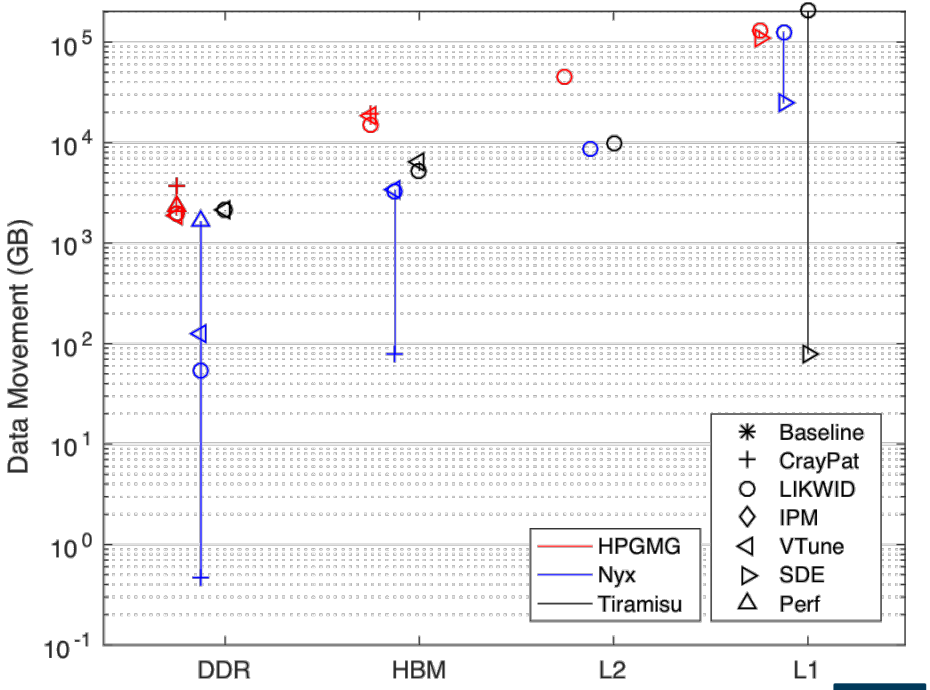
Metric 3: Memory Bandwidth

- data movement -> **average** bandwidth
- instantaneous bandwidth -> **max** bandwidth

- **4 levels**: DDR, LLC, L2, and L1

- **CrayPat** and **VTune**: DDR/HBM, avg./max
- **LIKWID**: all 4 levels, no max
- **SDE**: L1, avg. skewed by runtime overhead
- **Perf**: hexadecimal code; DDR, avg.

- Accuracy?



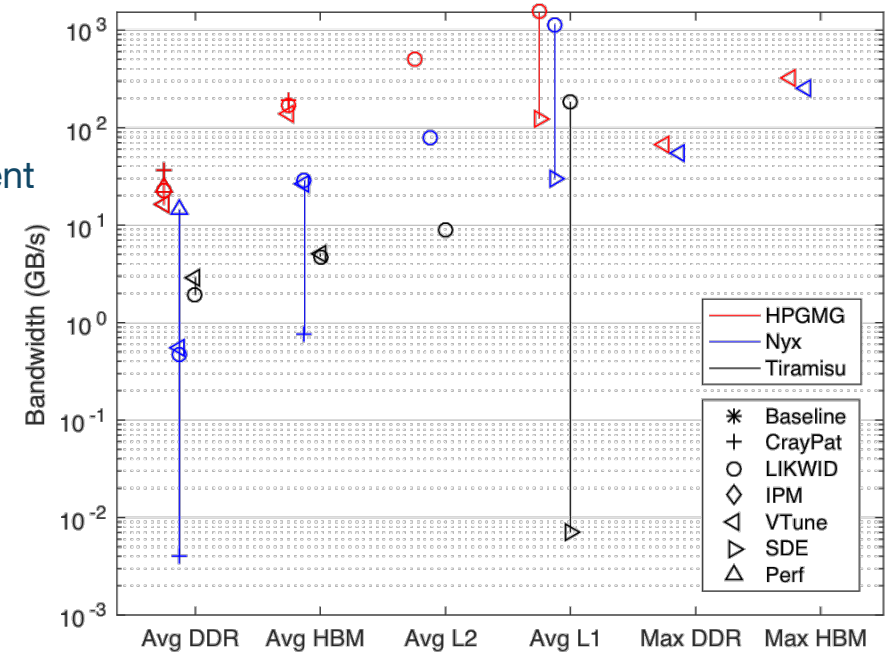
Metric 3: Memory Bandwidth

HPGMG:

- Consistent results on DDR and HBM level from CrayPat, LIKWID and VTune (both data movement and average bandwidth)
- Expected tendency for L2 and L1 data movement from **LIKWID** and SDE
- Reasonable max bandwidth on DDR and HBM from **VTune**

Nyx/Tiramisu:

- CrayPat < LIKWID/VTune < Perf
- SDE unable to introspect precompiled binaries



Metric 4: Memory High Watermark

CrayPat

- uses */proc/self/numa_maps* for reporting
- captured near the end of the program

IPM

- uses */proc/self/status* for reporting
  - *VmHWM*: Peak resident set size ("high water mark").
- more accurate: **13.52GB** vs. **12.3GB** (from Nyx's own memory tracking)

	CrayPat	LIKWID	IPM	VTune	SDE	Perf
HPGMG	0.255	-	11.93	-	-	-
Nyx	2.76	-	13.52	-	-	-
Tiramisu	-	-	-	-	-	-

Metric 5: Vectorization Efficiency

- Ratio:

  packed arithmetic floating-point instructions : total number of arithmetic floating-point instructions

- **HPGMG** well vectorized **<->** high vectorization efficiency reported by tools
- **Nyx** not well optimized **<->** very low vectorization efficiency

  LIKWID and Perf possibly over-reporting due to inclusion of non-arithmetic vector micro-ops
- **Tiramisu** utilizes MKL-DNN and cuDNN (highly optimized) **<->** near-1 vectorization efficiency reported by all tools

	CrayPat	LIKWID	IPM	VTune	SDE	Perf
HPGMG	-	0.945	-	-	0.949	0.949
Nyx	-	0.309	-	-	0.088	0.311
Tiramisu	-	0.993	-	-	1	0.993

Results Analysis

Strengths | Weaknesses

	Usability	Overhead	Actionability	(In-)Accuracy
CrayPat	• Instrumentation required • pat_build	• Negligile for Sampling • Variabile for Tracing	• BW, Hi mem; no FLOP, vec. eff.; but hotspots, load balance, MPI comm., IO • text and graphical reports	• Max BW captured at end of execution; inaccurate
LIKWID	• Elevated user access • No instrumentation except with Marker API	• Low; mainly at finalization • Number of ranks/threads can increase overhead	• FLOP, BW, vec. eff.; no Hi mem; access HW counters • Command line	• Vector FLOP's include non-arithmetic vector uops
IPM	• Prepend to native run command • No instrumentation	• Low	• Limited information; but MPI comm., *etc* • Command line	• Accurate Hi mem.
VTune	• Extra kernel modules • Privileged user access • -g and dynamic linking	• High • Depends on analysis type	• BW, Hi mem; no FLOP, vec. eff.; but hotspots, load balance, concurrency, locks/waits • GUI	• Accurate for supported metrics
SDE	• No instrumentation	• Very high • Startup, during execution and at finalization	• Mainly instruction level code characteristics • Unable to tap into pre-compiled binaries/libraries	• Accurate FLOP's
Perf	• No instrumentation • Lack pre-defined performance groups • Hex codes for HW counters	• Low	• Can access HW counters but lack provision of performance groups	• Vector FLOP's include non-arithmetic vector uops

Conclusion

Suitability for performance data collection by	HPC users?	HPC facilities?
SDE: • incurs too high an overhead; • provides limited actionable information;	✗	✗
IPM: • limited information available	✗	✗
CrayPat and **VTune:** • require certain work to gather a good amount of information • relatively low overhead and high accuracy	✓	✗
LIKWID and **Perf:** • minimal work required to collect data • produce a good amount of information with low overhead	✗	✓

Thank You