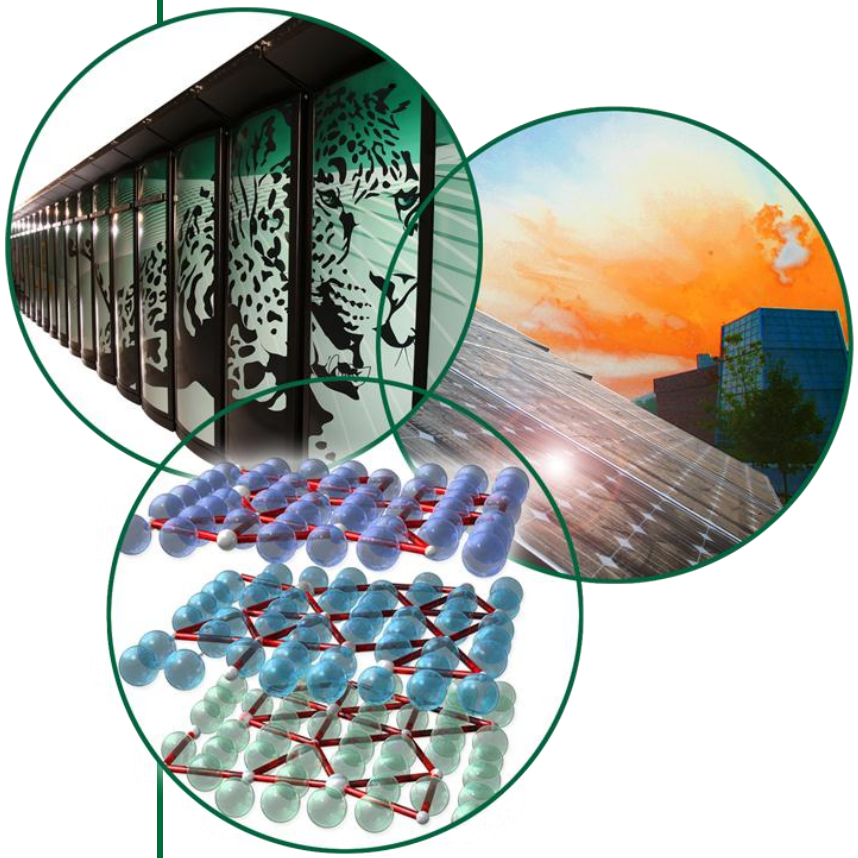
Brief Overview of DOE Blackcomb and Vancouver Projects



U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY

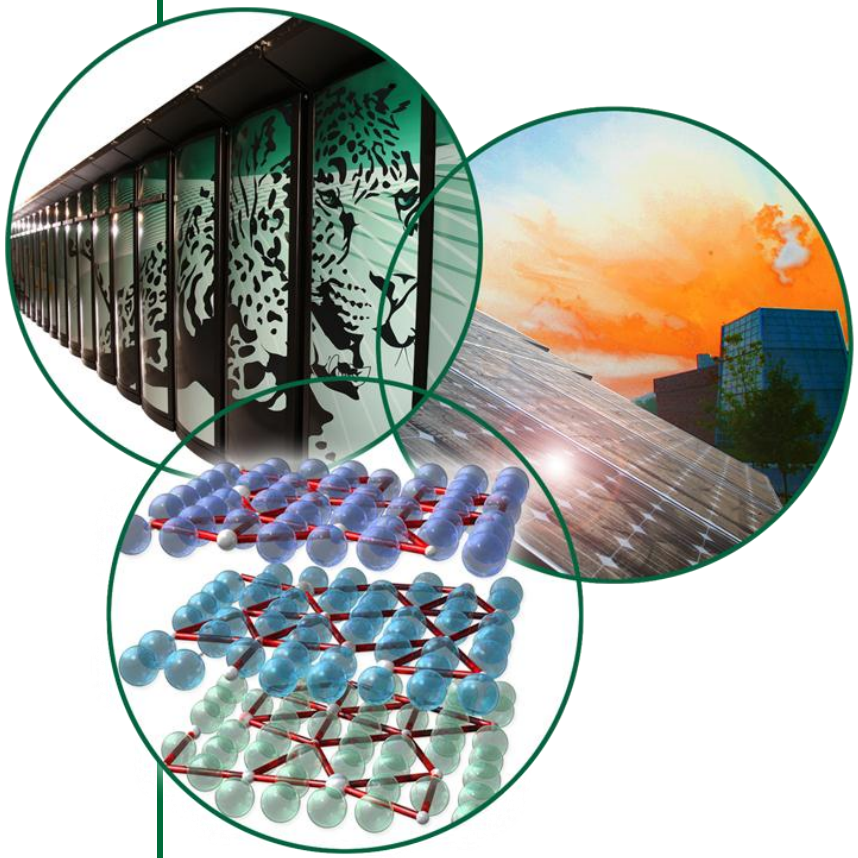
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Blackcomb: Brief Overview

Presented to X-Stack PI Meeting, LBL

21 Mar 2013

Jeffrey S. Vetter, ORNL
Robert Schreiber, HP Labs
Trevor Mudge, University of Michigan
Yuan Xie, Penn State University



U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Blackcomb: Hardware-Software Co-design for Non-Volatile Memory in Exascale Systems

Jeffrey Vetter, ORNL
Robert Schreiber, HP Labs
Trevor Mudge, University of Michigan
Yuan Xie, Penn State University

http://ft.ornl.gov/trac/blackcomb

FWP #**ERKJU59**

Objectives

- Rearchitect servers and clusters, using nonvolatile memory (NVM) to overcome resilience, energy, and performance walls in exascale computing:
  - Ultrafast checkpointing to nearby NVM
  - Reoptimize the memory hierarchy for exascale, using new memory technologies
  - Replace disk with fast, low-power NVM
  - Enhance resilience and energy efficiency
  - Provide added memory capacity

Approach

- Identify and evaluate the most promising (NVM) technologies – STT, PCRAM, memristor.
- Explore assembly of NVM and CMOS into a storage + memory stack.
- Propose an exascale HPC system architecture that builds on our new memory architecture.
- New resilience strategies in software.
- Test and simulate, driven by proxy applications.

Established and Emerging Memory Technologies – A Comparison

	SRAM	DRAM	eDRAM	NAND Flash	PCRAM	STTRAM	ReRAM (1T1R)	ReRAM (Xpoint)
Data Retention	N	N	N	Y	Y	Y	Y	Y
Cell Size (F^2)	50-200	4-6	19-26	2-5	4-10	8-40	6-20	1- 4
Read Time (ns)	< 1	30	5	10^4	10-50	10	5-10	50
Write Time (ns)	< 1	50	5	10^5	100-300	5-20	5-10	10-100
Number of Rewrites	10^{16}	10^{16}	10^{16}	10^4-10^5	10^8-10^{12}	10^{15}	10^8-10^{12}	10^6-10^{10}
Read Power	Low	Low	Low	High	Low	Low	Low	Medium
Write Power	Low	Low	Low	High	High	Medium	Medium	Medium
Power (other than R/W)	Leakage	Refresh	Refresh	None	None	None	None	Sneak

Device level investigations: MLC ReRAM

- Material composition, device organization, programming, etc

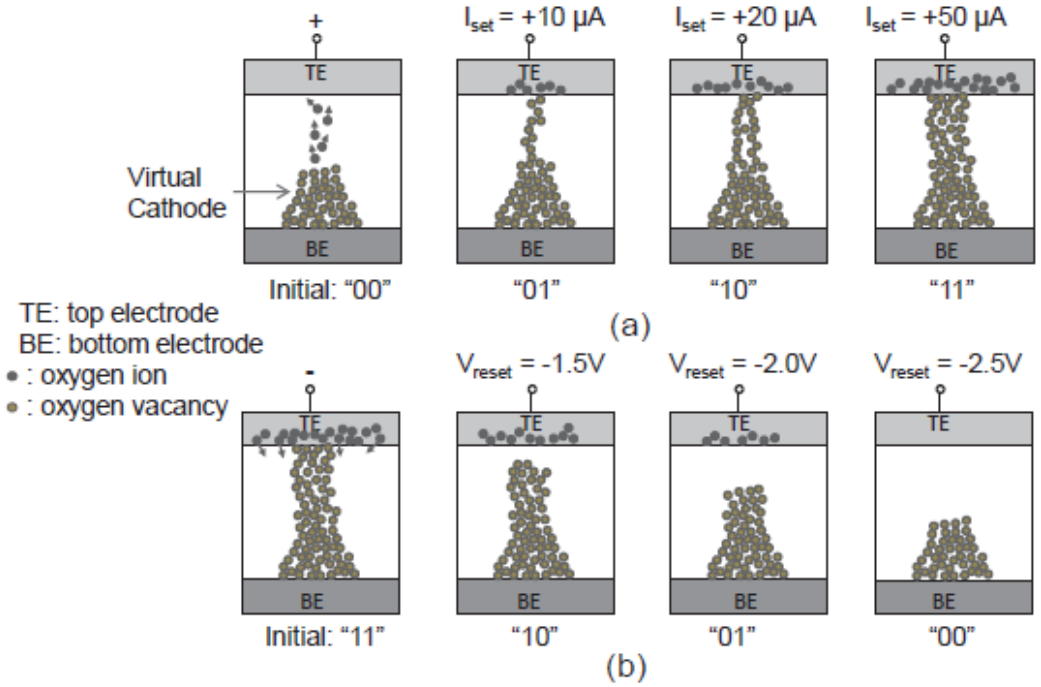
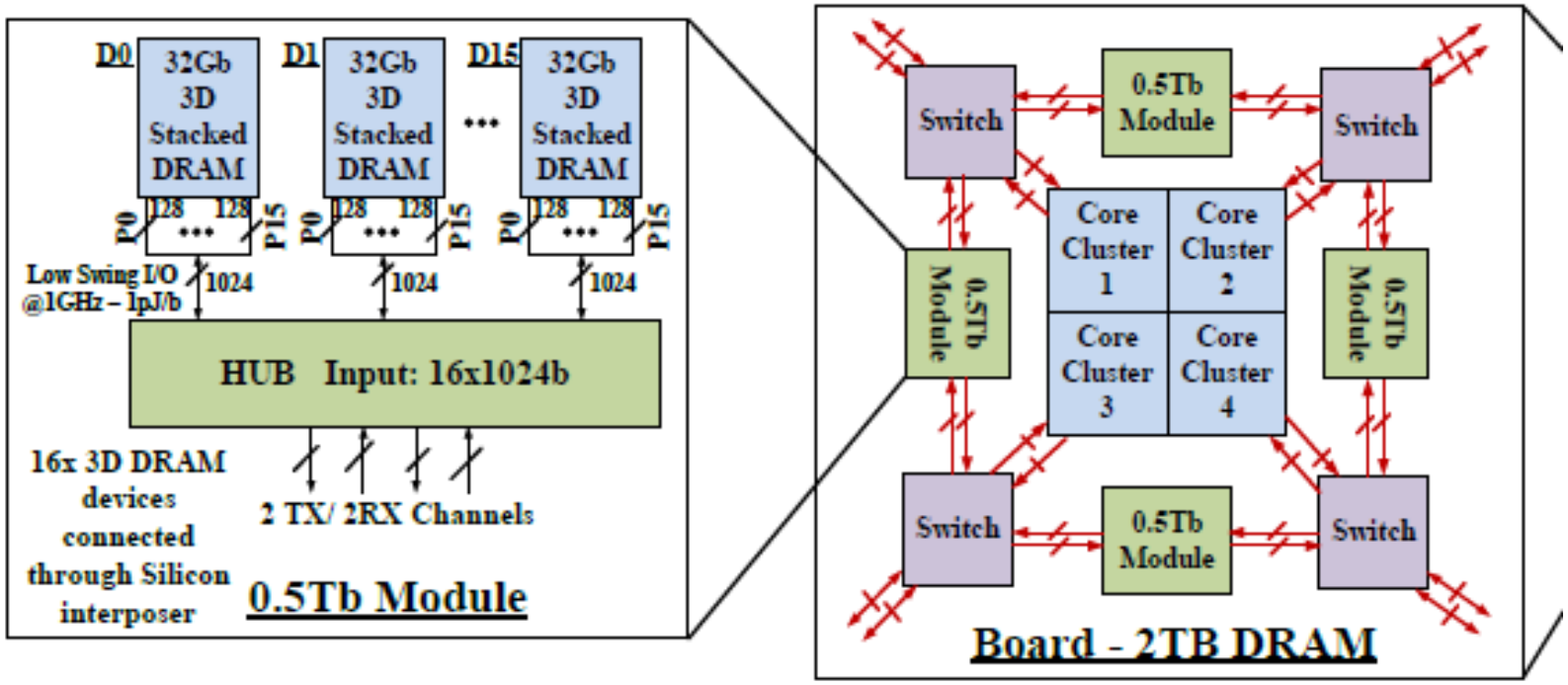


Fig. 1. Multi-level switching in ReRAM: (a) H2L and (b) L2H programming.

N. Muralimanohar et al., "Understanding the Trade-offs in MLC ReRAM Design," in *DAC, 2013*

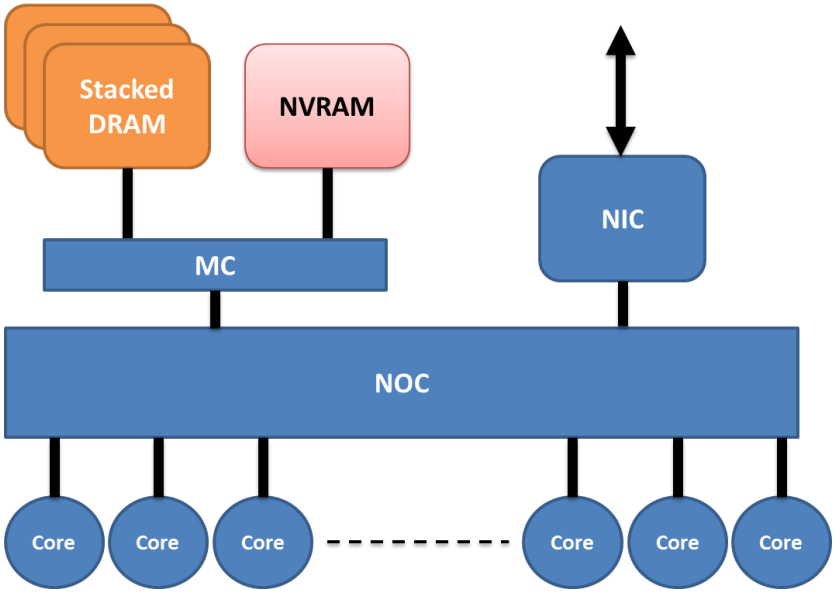
Tradeoffs in Exascale Memory Architectures

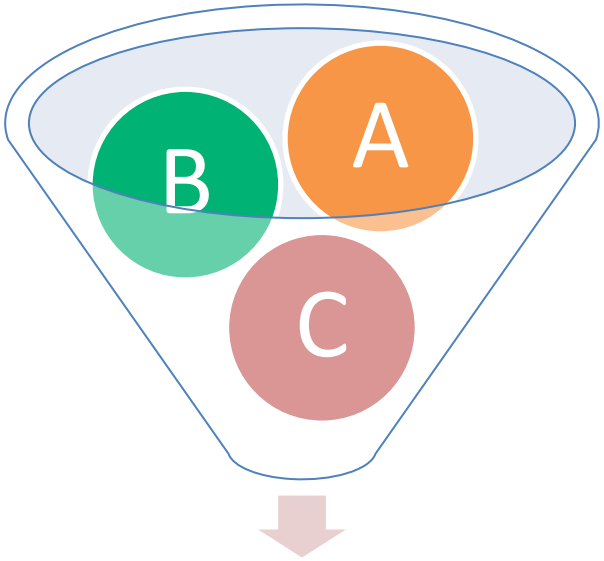


- ECC type, row buffers, DRAM physical page size, bitline length, etc

T. Mudge et al., "Optimizing DRAM Architectures for Energy-Efficient, Resilient Exascale Memories," (*submitted), 2013*

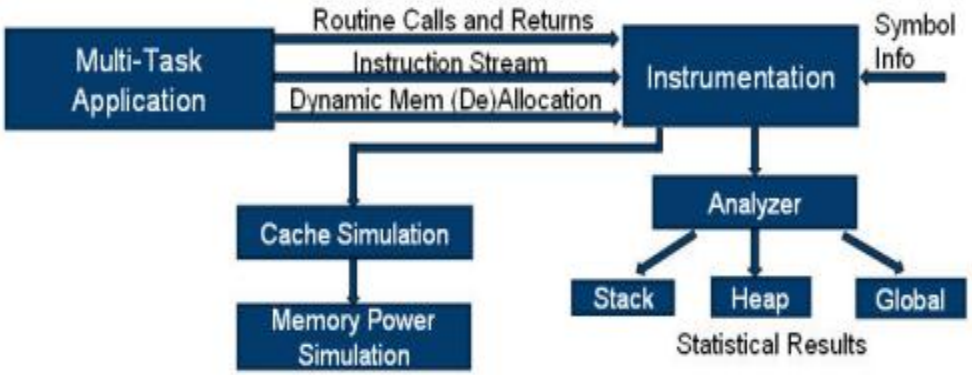
New hybrid memory architectures: What is the ideal organizations for application scenarios?





Identifying Opportunities for Byte-Addressable Non-Volatile Memory in Extreme-Scale Scientific Applications

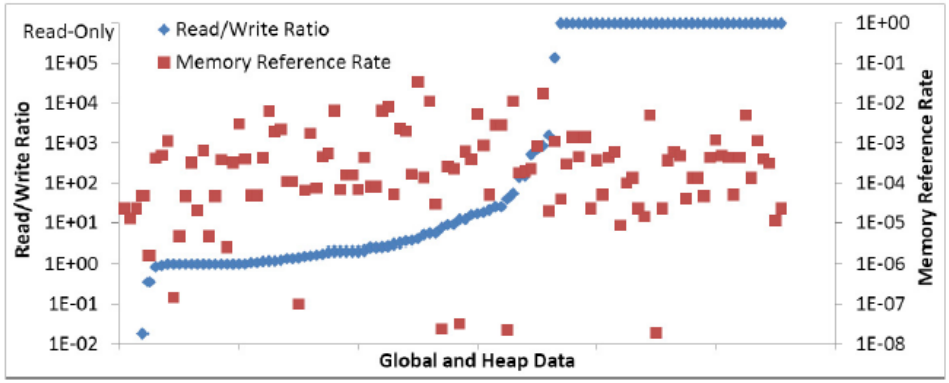
- **Problem**
  - Do specific memory workload characteristics of scientific apps map well onto NVRAMs' features?
  - Can NVRAM be used as a solution for future Exascale systems?



- **Solution**
  - Develop a binary instrumentation tool to investigate memory access patterns related to NVRAM
  - Study realistic DOE applications (Nek5000, S3D, CAM and GTC) at fine granularity
- **Impact**
  - Identify large amount of commonly existing data structures that can be placed in NVRAM to save energy
  - Identify many NVRAM-friendly memory access patterns in DOE applications
  - Received attention from both vendor and apps teams

D. Li, J.S. Vetter, G. Marin, C. McCurdy, C. Cira, Z. Liu, and W. Yu, "Identifying Opportunities for Byte-Addressable Non-Volatile Memory in Extreme-Scale Scientific Applications," in *IEEE International Parallel & Distributed Processing Symposium (IPDPS). Shanghai: IEEEE, 2012*

Measurement Results



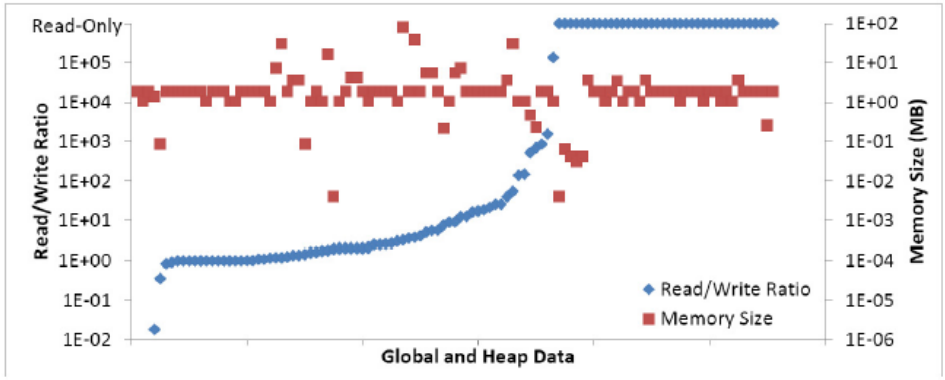
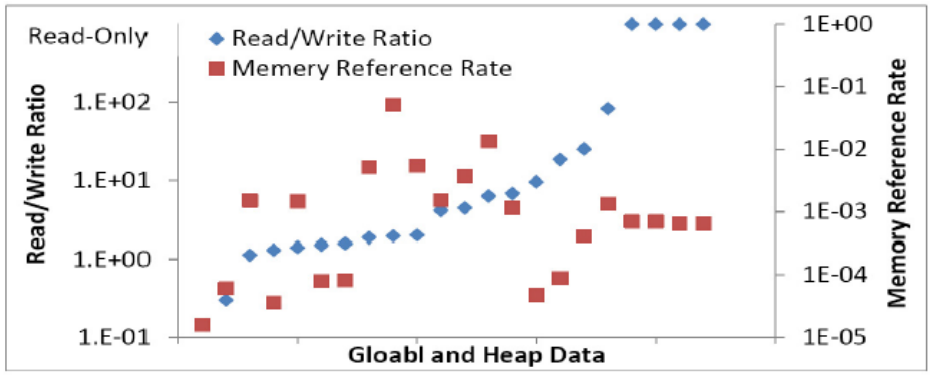


Figure 3: Read/write ratios, memory reference rates and memory object sizes for memory objects in Nek5000



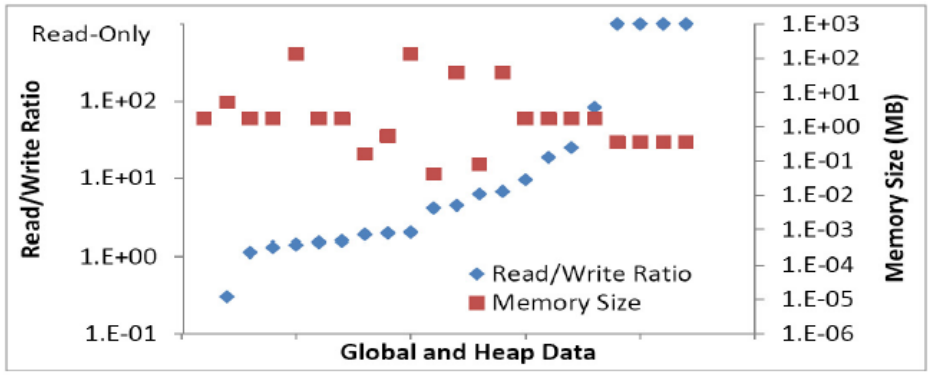
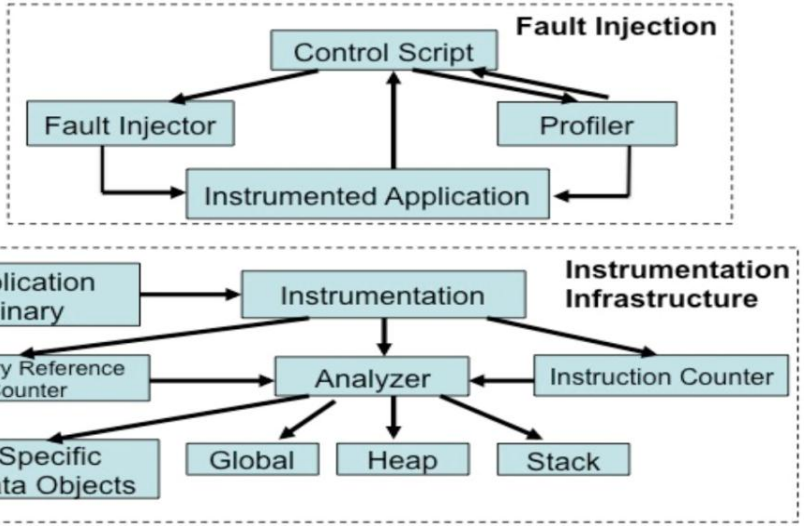


Figure 6: Read/write ratios, memory reference rates and memory object sizes for memory objects in S3D

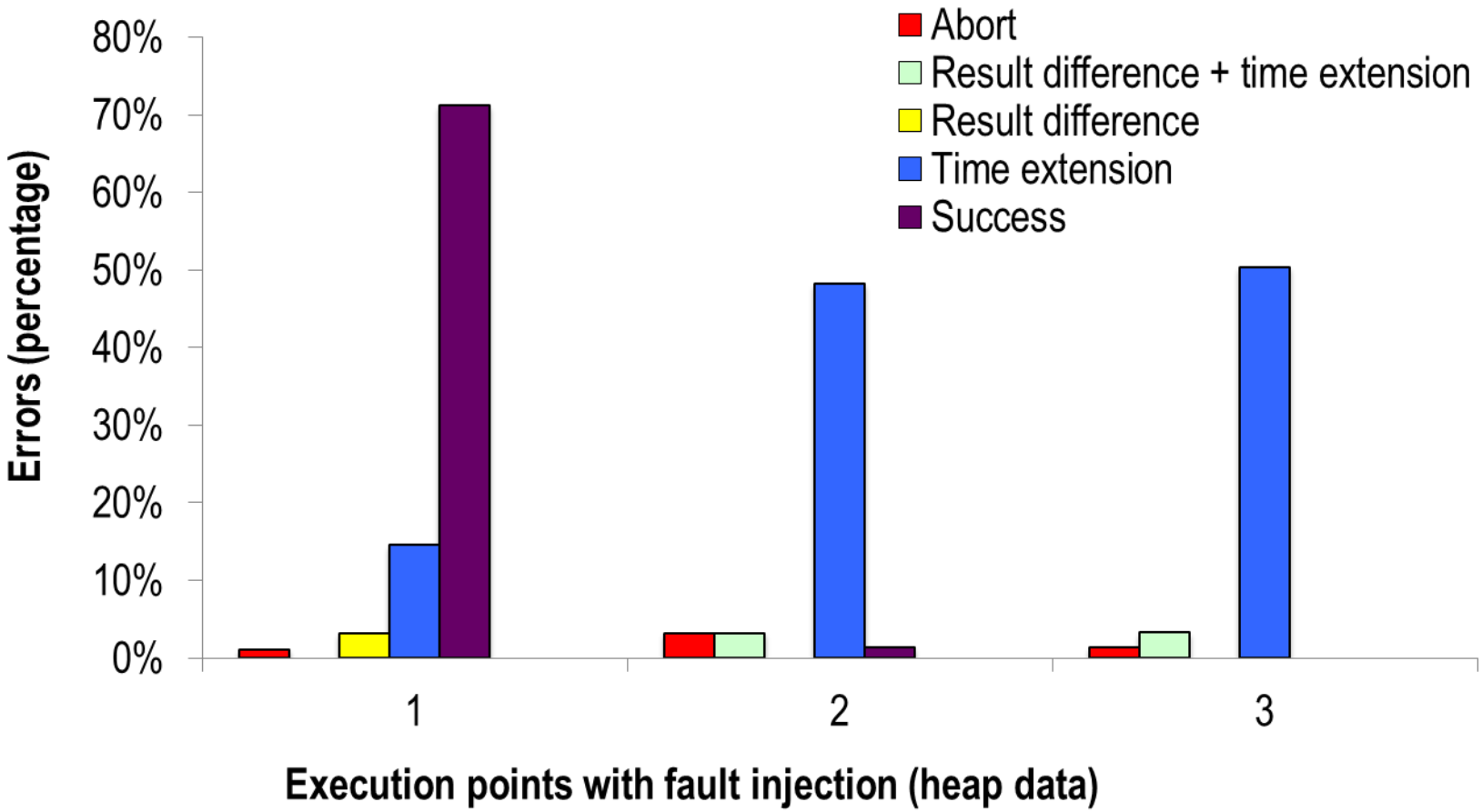
Classifying Soft Error Vulnerabilities in Extreme-Scale Scientific Apps Using a Binary Instrumentation Tool



- **Problem**
  - We lack deep understanding of apps vulnerability with complete fault coverage
- **Solution**
  - Build an empirical fault injection and consequence analysis tool to evaluate how soft errors impact apps
  - classify each applications individual data structures based on their sensitivity to these vulnerabilities, and generalize these classifications across applications
- **Impact**
  - Reveal intrinsic relationships between application vulnerability and specific data objects for mission-critical DOE applications (Nek5000, S3D, and GTC)
  - Motive innovation in applications, architectures, and programming models

D. Li, J.S. Vetter, and W. Yu, "Classifying Soft Error Vulnerabilities in Extreme-Scale Scientific Applications Using a Binary Instrumentation Tool," in *SC12, 2012*

Application Results (S3D): Heap



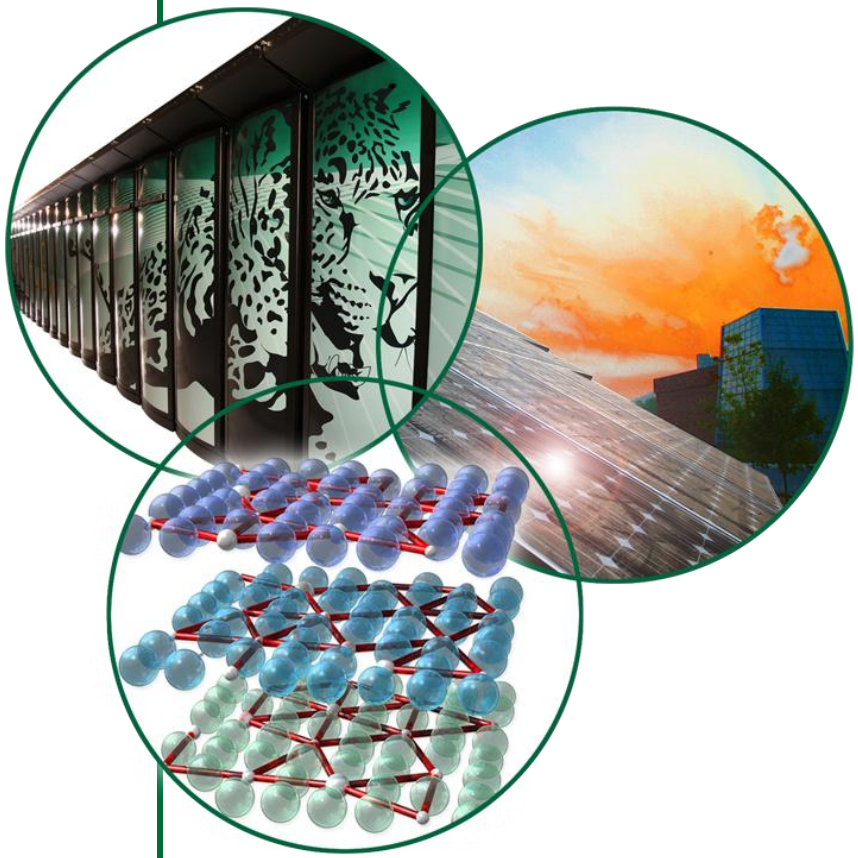
Observation: the application is very sensitive to when the fault is injected.

Vancouver: Brief Overview

Presented to X-Stack PI Meeting, LBL

21 Mar 2013

Jeffrey S. Vetter, ORNL
Wen-Mei Hwu, UIUC
Allen Malony, University of Oregon
Rich Vuduc, Georgia Tech



U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Vancouver: A Software Stack for Productive Heterogeneous Exascale Computing

Jeffrey Vetter, ORNL
Wen-Mei Hwu, UIUC
Allen Malony, University of Oregon
Rich Vuduc, Georgia Tech

ERKJU44

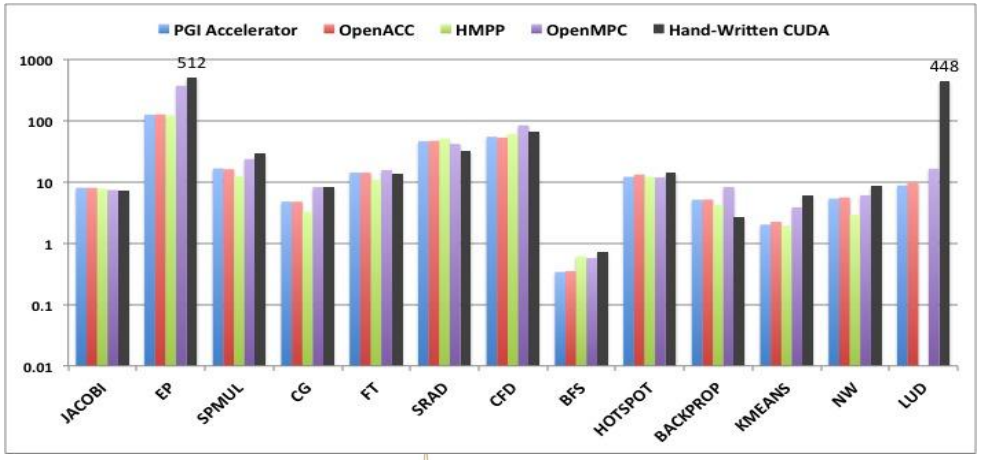
Objectives

- Enhance programmer productivity for the exascale
  - Increase code development ROI by enhancing code portability
  - Decrease barriers to entry with new programming models
- Create next-generation tools to understand the performance behavior of an exascale machine
- Automate common routines to improve performance portability and programmability

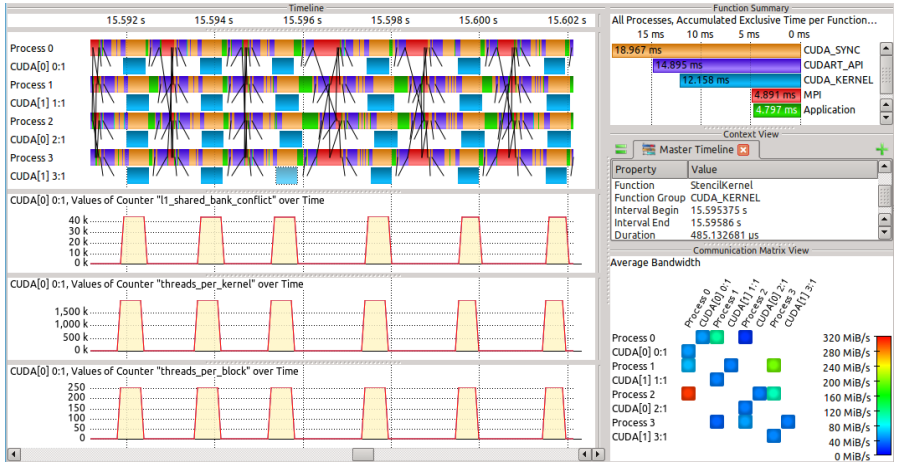
Approach

- Understand emerging heterogeneous architectures
- Programming tools
  - Compilers
  - GAS programming model
- Software libraries: autotuning
- Performance tools targeting appropriate, new abstractions
- Impact on DOE Applications

Programmer Productivity



Performance



Georgia Tech College of Computing
Computational Science and Engineering
ILLINOIS
UNIVERSITY OF OREGON
OAK RIDGE National Laboratory

The Scalable HeterOgeneous Computing (SHOC) Benchmark Suite

PI: Jeffrey S. Vetter, ORNL
Future Technologies Group

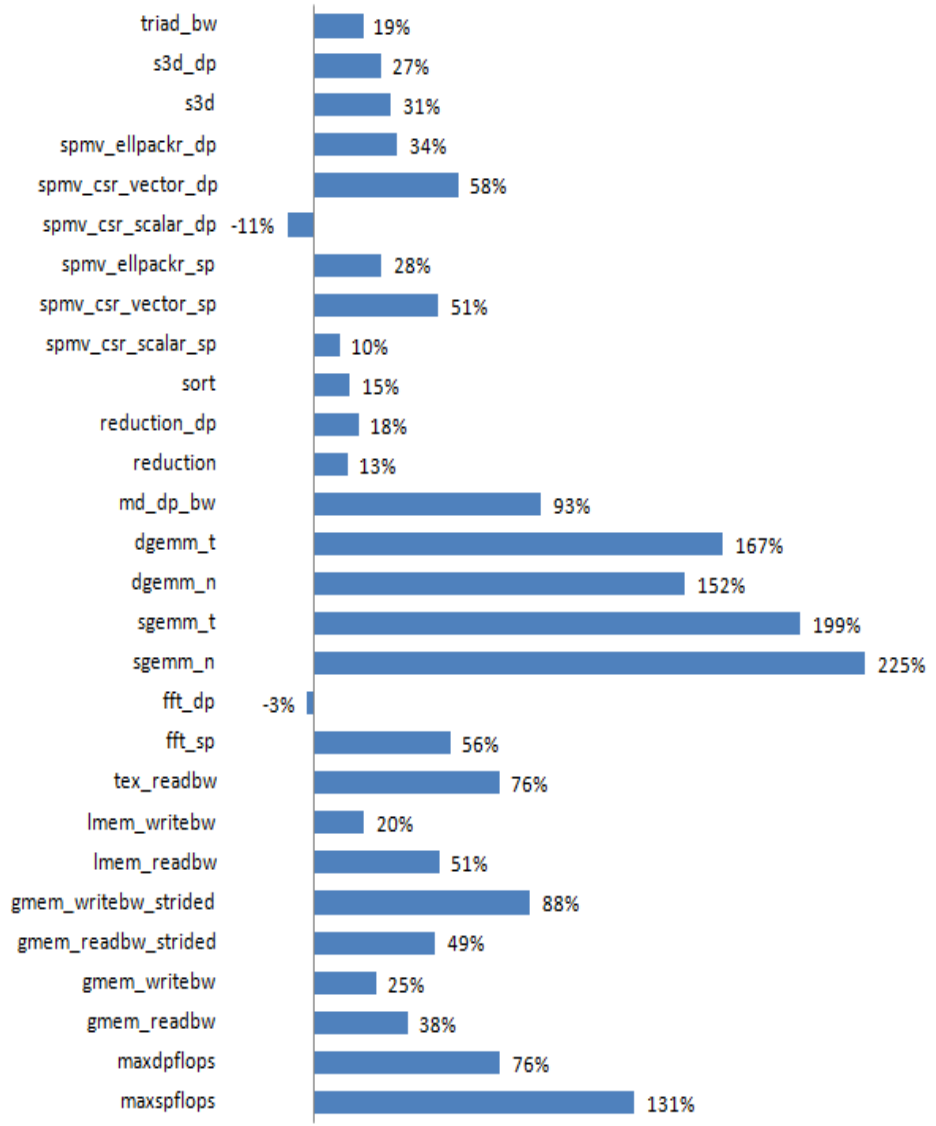
http://j.mp/shocmarks

Objectives

- *Design and implement a set of performance and stability tests for HPC systems with **heterogeneous** architectures*
- *Implemented each test in MPI, OpenCL, CUDA to*
  - *Evaluate the differences in these emerging programming models*
  - *MIC to be released shortly*
  - *OpenACC coming later this spring*
- Sponsored by NSF, DOE

Accomplishments

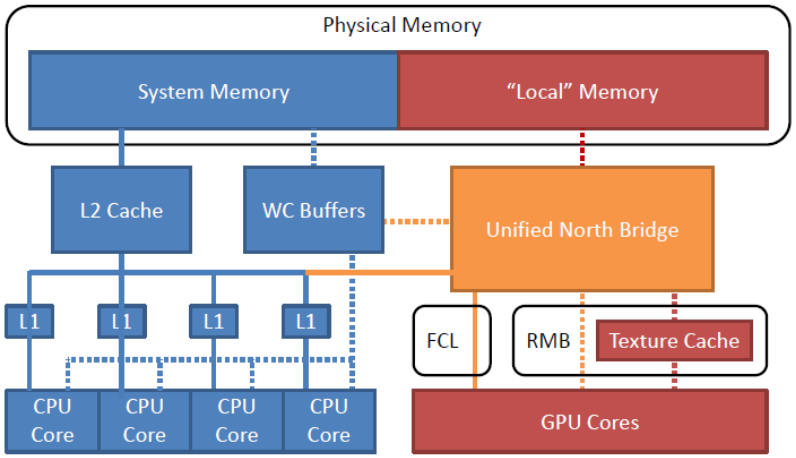
- *Consistent open source software releases*
  - *Over 10000 downloads internationally since 2010*
  - *Used in multiple procurements worldwide*
  - *Used by vendors and researchers for testing, understanding*
- *Across diverse range of architectures: NVIDIA, AMD, ARM, Intel, even Android*
- *Overview published at 3rd Workshop General-Purpose Computation on Graphics Processing Units (GPGPU '10): ~80 citations to date*



A. Danalis, G. Marin, C. McCurdy, J. Meredith, P.C. Roth, K. Spafford, V. Tipparaju, and J.S. Vetter, "The Scalable HeterOgeneous Computing (SHOC) Benchmark Suite," in Third Workshop on General-Purpose Computation on Graphics Processors (GPGPU 2010)`. Pittsburgh, 2010

This chart shows the "out of the box" improvement from NVIDIA Fermi (M2090) to Kepler (K20m). Measured using CUDA 5.0 with an identical host system. Largest improvements observed in compute intensive workloads. Modest increases for memory bound kernels. No increase in DP FFT, suggests CUFFT not completely optimized for Kepler in release 5.0.

AMD Llano's fused memory hierarchy



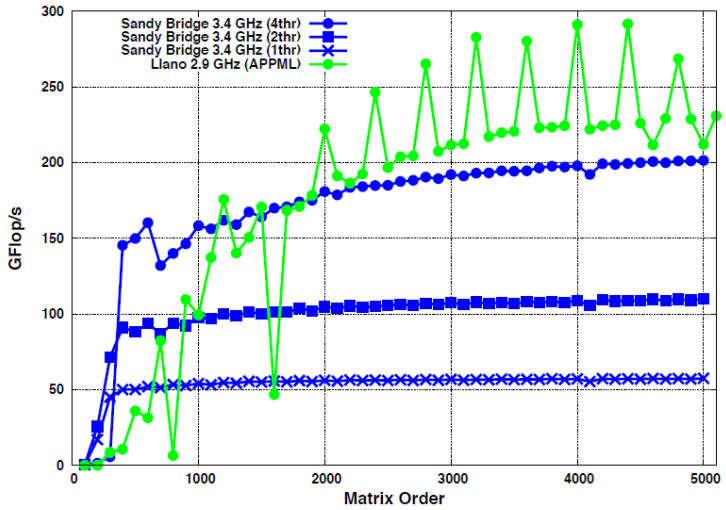
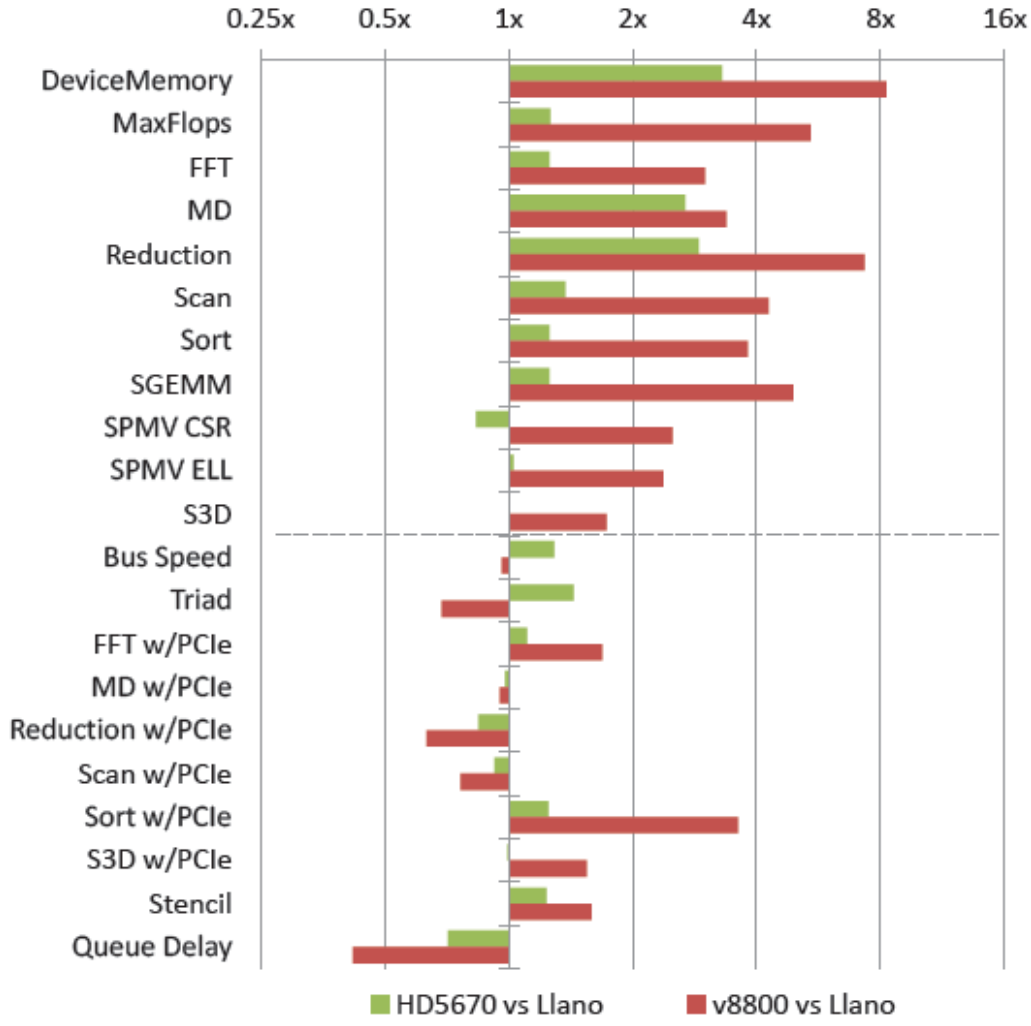


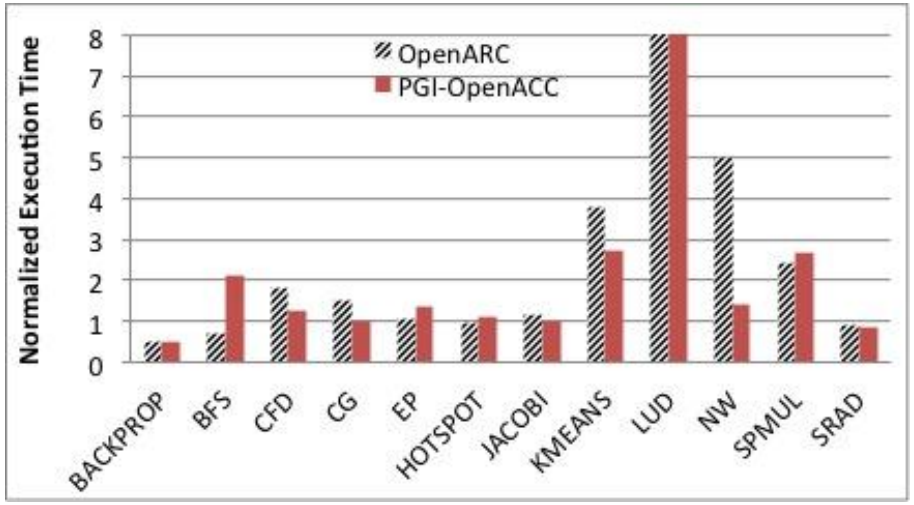
Figure 3: SGEMM Performance (one, two, and four CPU threads for Sandy Bridge and the OpenCL-based AMD APPML for Llano's fGPU)



K. Spafford, J.S. Meredith, S. Lee, D. Li, P.C. Roth, and J.S. Vetter, "The Tradeoffs of Fused Memory Hierarchies in Heterogeneous Architectures," in ACM Computing Frontiers (CF). Cagliari, Italy: ACM, 2012. Note: Both SB and Llano are consumer, not server, parts.

OpenARC: Open Accelerator Research Compiler

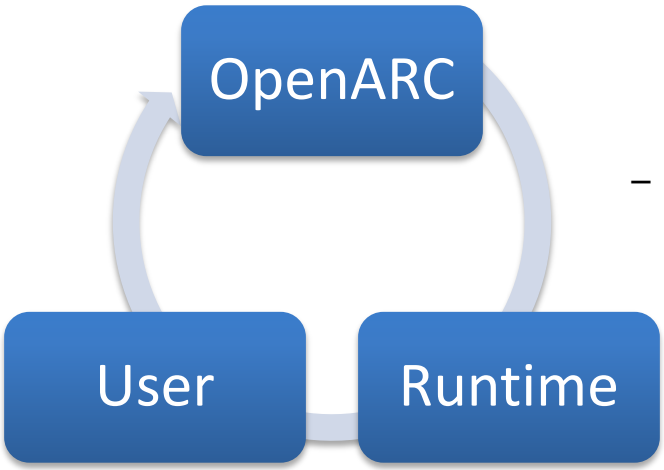
- Problem
  - Directive-based GPU programming models provide abstraction over complex language syntax of low-level GPU programming and diverse architectural details. However, too much abstraction puts significant burdens on programmers regarding debugging and performance optimizations.
- Solution
  - OpenARC is *an open-sourced, very High-level Intermediate Representation (HIR)-based, extensible compiler framework*, where various performance optimizations, traceability mechanisms, fault tolerance techniques, etc., can be built for better debuggability/performance/resilience on the complex accelerator computing.
- Impact
  - OpenARC is the first open source compiler *supporting full OpenACC features*.
  - HIR with a rich set of directives in OpenARC provides a powerful research framework for *various source-to-source translation and instrumentation experiments*, even for porting Domain-Specific Languages (DSLs).
  - Additional OpenARC directives with its built-in tuning tools allow users to control overall OpenACC-to-GPU translation in a fine-grained, but still abstract manner.



Performance of OpenARC and PGI-OpenACC compilers relative to manual CUDA versions (Lower is better.)

Optimization and Interactive Program Verification with OpenARC

- Problem
  - *Too much abstraction* in directive-based GPU programming!
    - Debuggability
      - Difficult to diagnose logic errors and performance problems at the directive level
    - Performance Optimization
      - Difficult to find where and how to optimize
- Solution
  - Directive-based, interactive GPU program verification and optimization
    - OpenARC compiler:
      - Generates runtime codes necessary for *GPU-kernel verification* and *memory-transfer verification and optimization*.
    - Runtime
      - Locate trouble-making kernels by comparing execution results at kernel granularity.
      - Trace the runtime status of CPU-GPU coherence to detect incorrect/missing/redundant memory transfers.
    - Users
      - Iteratively fix/optimize incorrect kernels/memory transfers based on the runtime feedback and apply to input program.



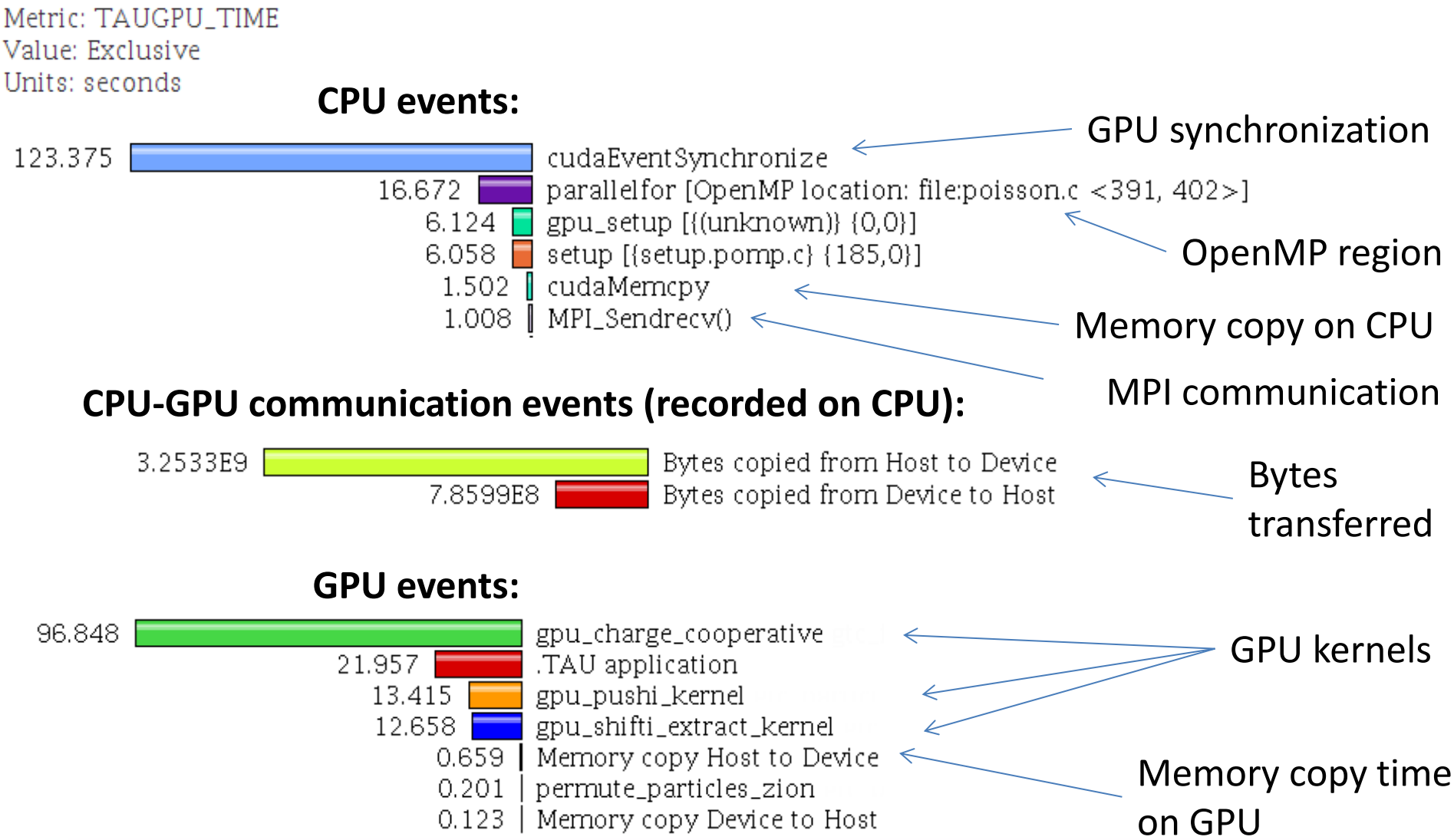
Iteratively find where and how to fix/optimize

Clause	Description
accglobal(list)	contains global symbols
accexplicitshared (list)	contains user-specified shared symbols
accreadonly(list)	contains R/O shared symbols
kernelConfPt (kernel)	indicates where to put kernel-configuration statements
gangconf(list)	contains sizes of each gang loop in nested gang loops
iterspace(exp)	contains iteration size of the loop

TAU for GPU Measurement

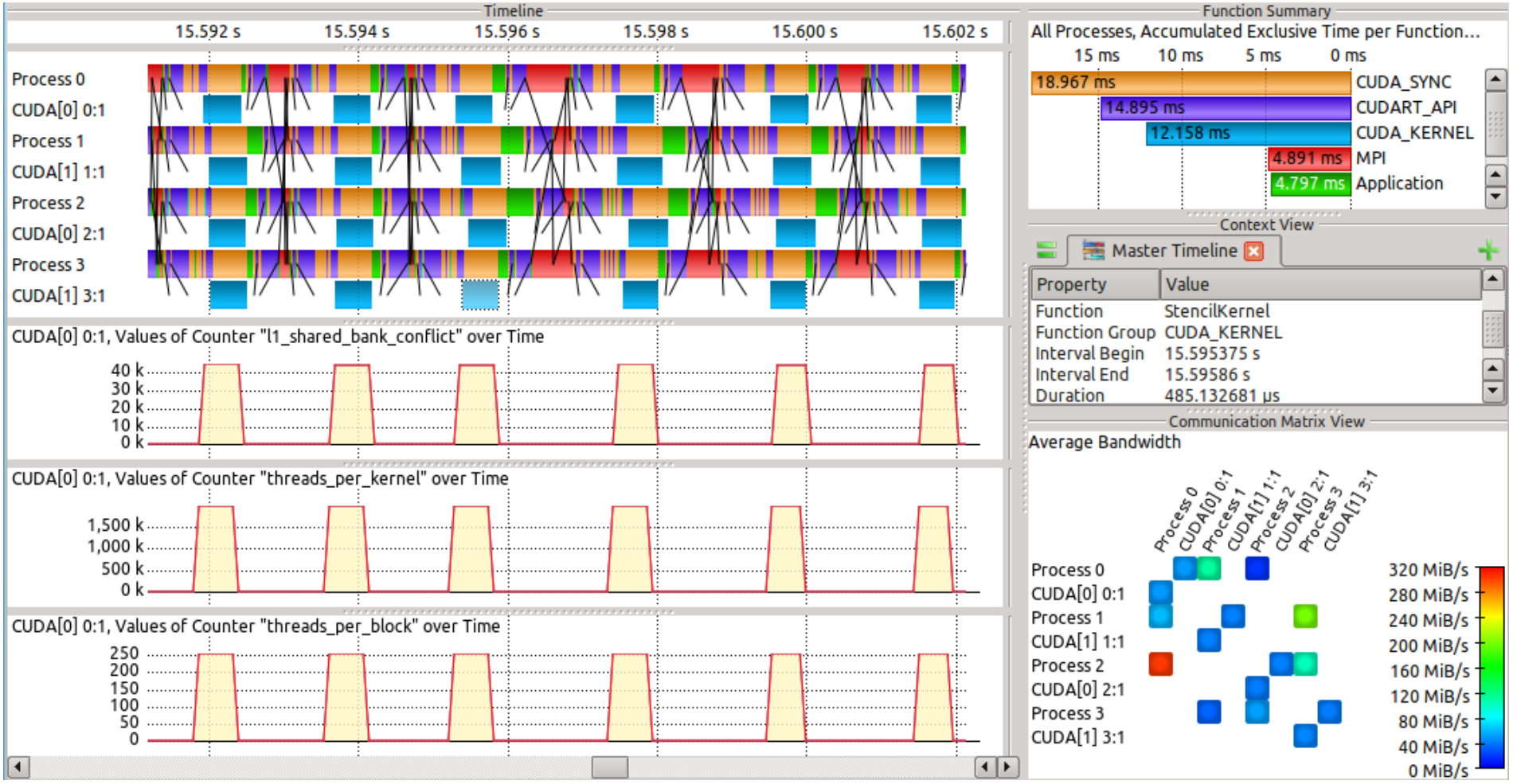
- **TAU Performance System® (http://tau.uoregon.edu)**
  - Instrumentation, measurement, analysis
  - Extended to support heterogeneous performance analysis
- **Integrate Host-GPU support in TAU measurement**
  - Enable host-GPU measurement approach
    - CUDA, OpenCL, PyCUDA as well as support for PGI and HMPP accelerator code generation capabilities
    - utilize PAPI CUDA and CUPTI
  - Provide both heterogeneous profiling and tracing
    - contextualization of asynchronous kernel invocation
- **Additional support**
  - TAU wrapping of libraries (tau_gen_wrapper)
  - Work with library preloading (tau_exec)

Guide to a TAU Profile



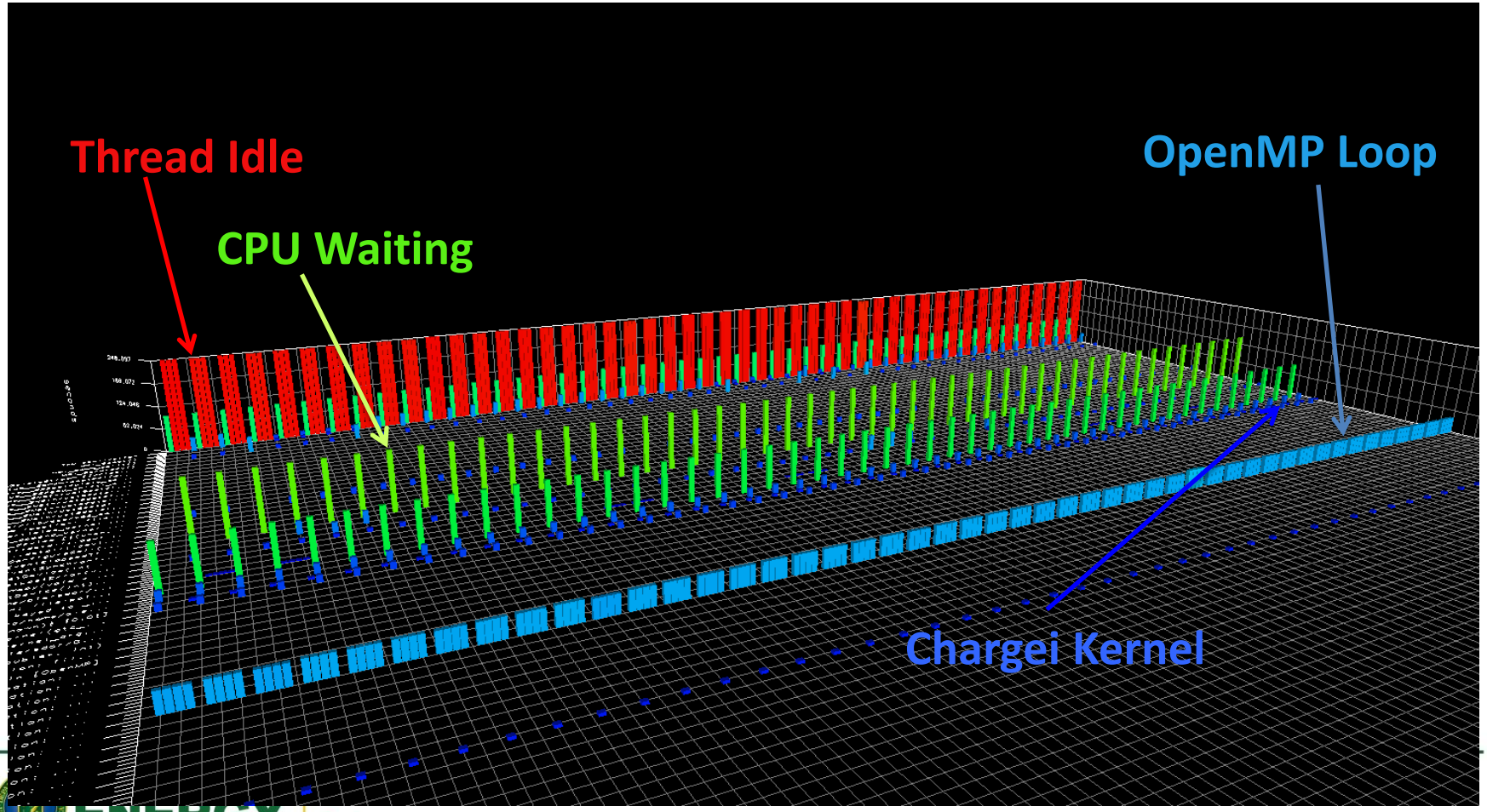
Stencil2D Trace (Vampir/VampirTrace)

- **Four MPI processes each with one GPU**
- **VampirTrace measurements**



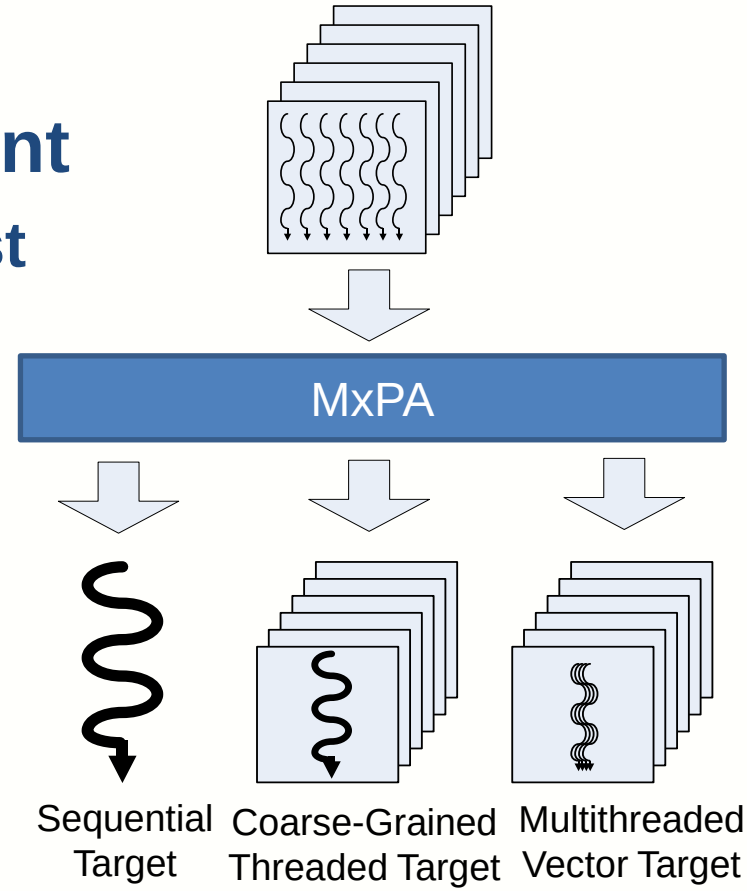
GTC on 16 Keeneland Nodes (48 MPI ranks)

- **48 MPI ranks**
  - 198 OpenMP threads (240 total threads), 48 GPUs



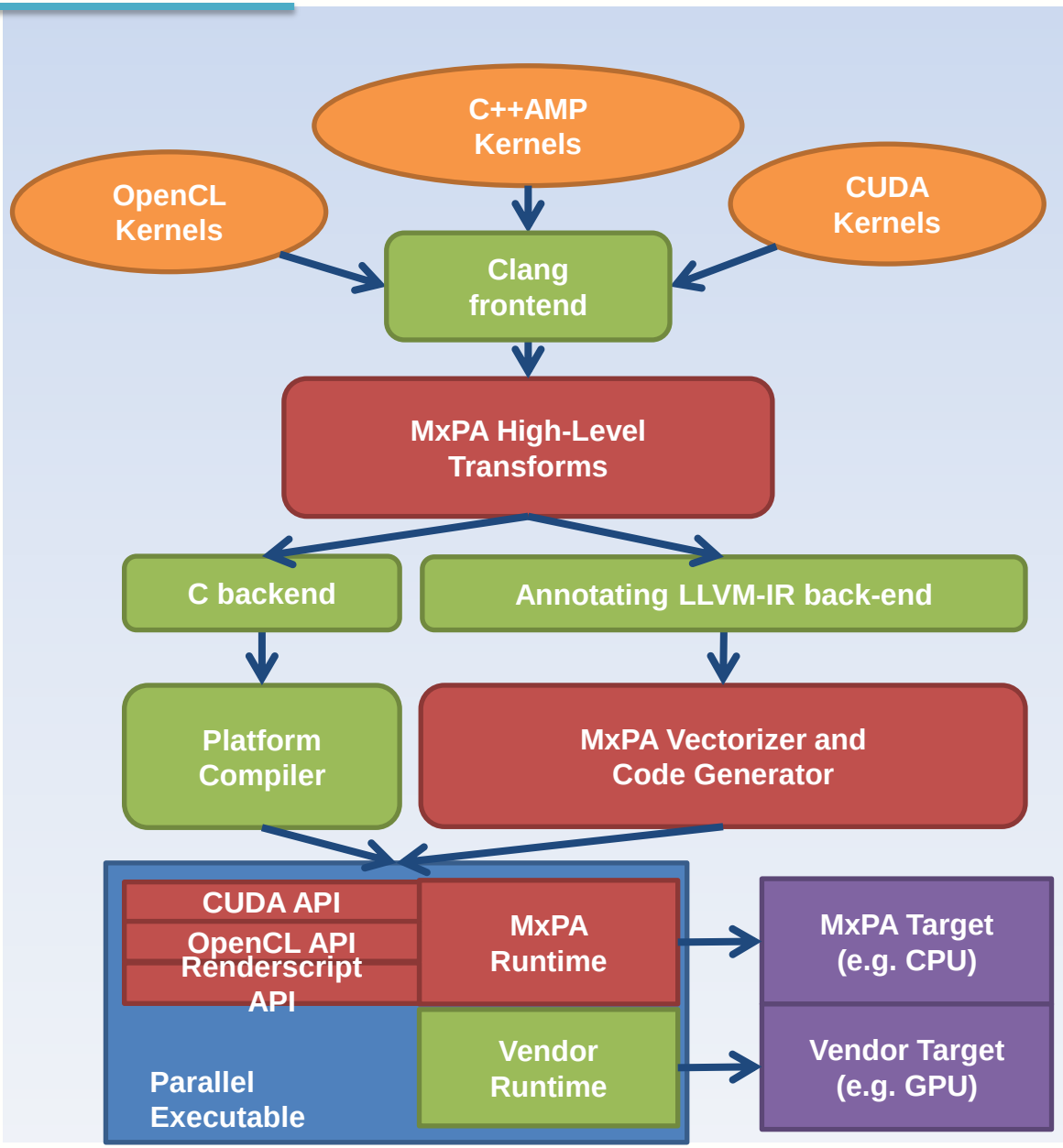
MxPA Performance Portability

- Single-source OpenCL (or CUDA, C++AMP) development
  - Control exascale software cost
- Many-target deployment
  - Maximize impact
- High-performance requirement
  - Adopting explicitly parallel programming models worthwhile



MxPA Transforms

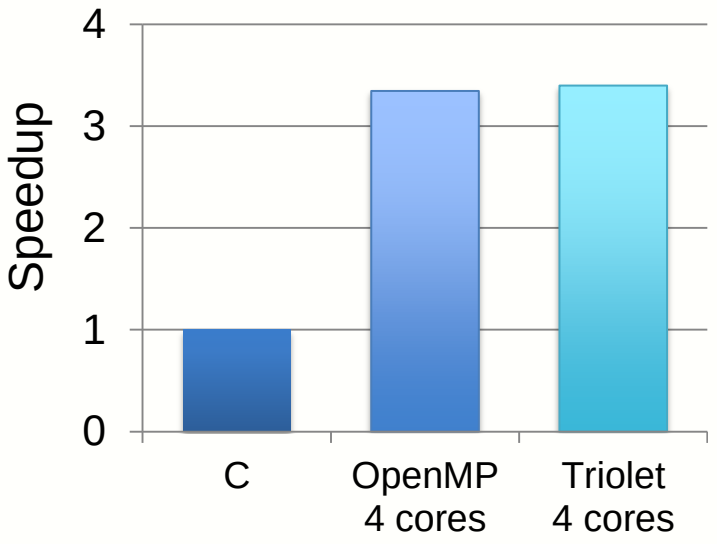
- Flexible compiler architecture based on the Clang LLVM frontend
- Full support for OpenCL currently, easily integrated with other frontends
- Can generate C code and a variety of parallelism and alignment annotations
- Adjusts thread parallelism granularity upward as necessary for target platform



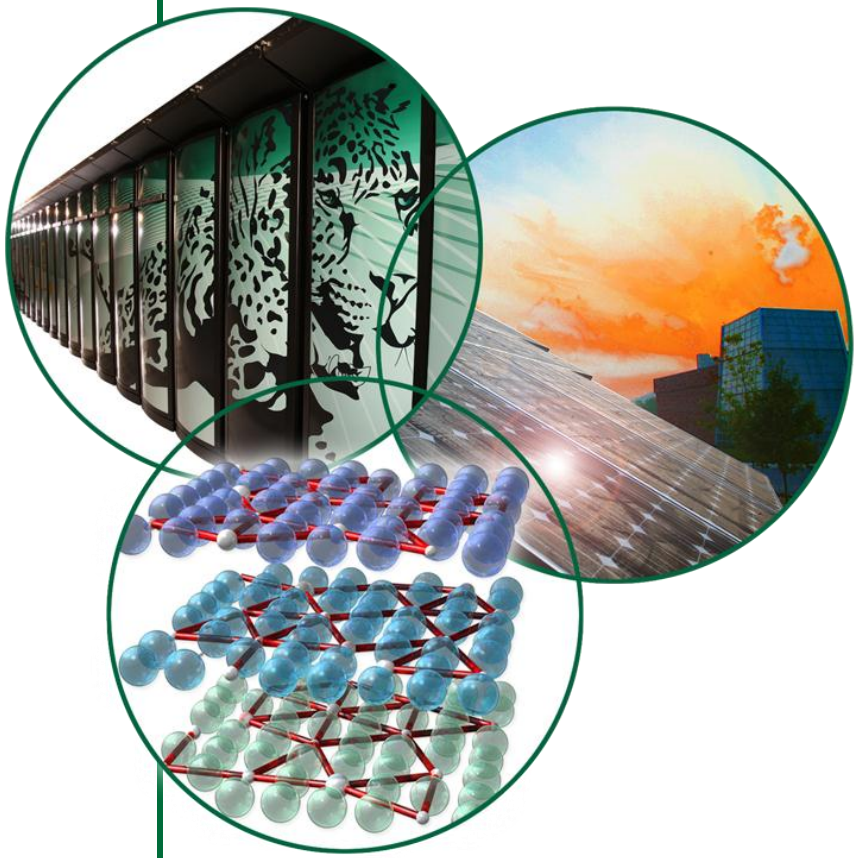
Triolet as a High-Level Interface

- High-level language features enhance programmability
  - Polymorphism, first-class functions, ...
  - Parallel loops written using parallel pattern library functions
- Optimizations remove most high-level overheads
  - Outputs sequential tasks and primitive parallel looping constructs
- Multicore implementation exists
  - Performance often similar to C

```
#  Histogramming problem from Parboil
#  Doubly nested loop
#  Library parallelizes outer loop using
#  histogram privatization
def autocorrelate(S):
  xs = (f(a, b)
        for (i, a) in enumerate(S)
        for b      in S[i+1:])
  return histogram(20, xs)
```



Other projects of interest



Aspen: A Domain Specific Language for Performance Modeling

PI: Jeffrey S. Vetter, ORNL
Kyle Spafford, ORNL

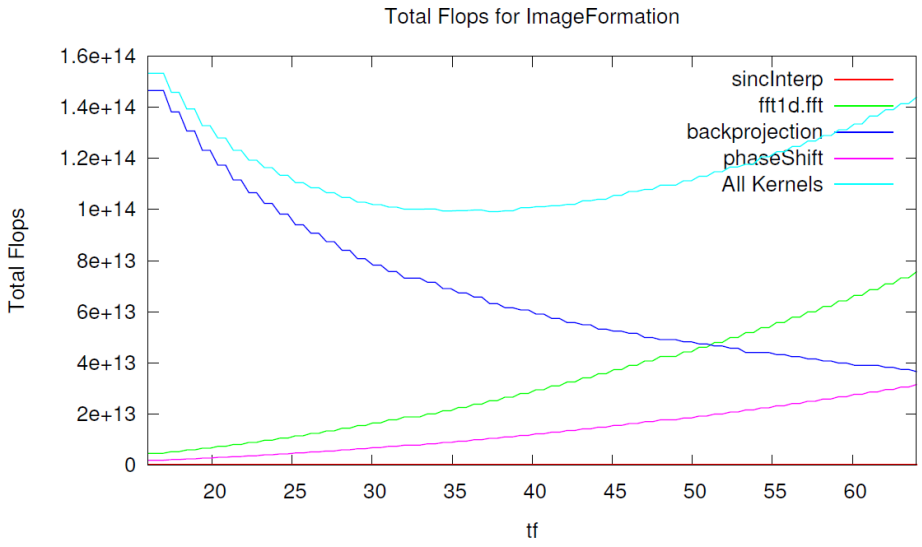
Objectives

- *Design and implement a new language for analytical performance modeling*
- *Use the language to create machine-independent models for important applications and kernels*
- *Develop a suite of analysis tools which operate on the models and produce key performance metrics like available parallelism, arithmetic intensity, and message volume*

Accomplishments

- *Developed a new language, compiler, and set of analysis tools*
- *Constructed models for important apps and mini-apps: MD, UHPC CP 1, Lulesh, 3D FFT*

K. Spafford and J.S. Vetter, "Aspen: A Domain Specific Language for Performance Modeling" To appear in the Proceedings of the ACM\IEEE Conference on High Performance Computing, Networking, Storage, and Analysis. (SC 12).



Example: Studying how the floating point requirements changed based on TF, an application-specific tiling factor in UHPC CP#1

Impact and Champions

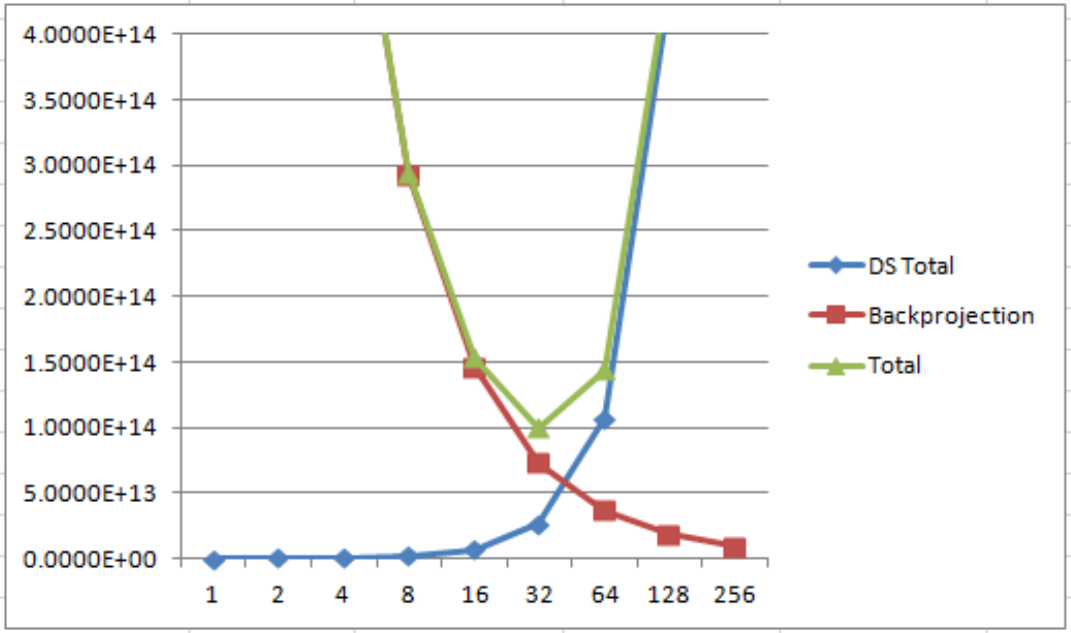
- *Increase understanding of application performance requirements*
- *Facilitate early-stage performance planning*
- *Sponsored by DoE – ExMatEx CoDesign Center, DARPA UHPC Echelon Team*

```
1 kernel localFFT {
2   exposes parallelism [n^2]
3   requires flops [5 * n * log2(n)] as dp,
        complex, simd
4   requires loads [a * n * max(1, log(n)/
        log(Z)) * wordSize] from fftVolume
5 }
```

Listing 2. Aspen statements for the local 1D FFTs

Example: Ad-Hoc Excel Files

	A	B	C	D	E	F	G	H	I	J
1	DS = Digital Spotlighting									
2	Tile Factor	DS Pulses/Sec	DS Samples/Pulse	DS FFT Flops	DS Range/Pulse	DS Total	Backprojection		Total	
3	1	2809	80636	1.85E+10	1.0442E+11	1.2288E+11	2.3378E+15		2.3380E+15	
4	2	1405	40318	1.85E+10	3.1889E+10	2.0140E+11	1.1693E+15		1.1695E+15	
5	4	703	20159	1.85E+10	1.3753E+10	5.1539E+11	5.8509E+14		5.8560E+14	
6	8	352	10080	1.85E+10	9.2163E+09	1.7712E+12	2.9296E+14		2.9473E+14	
7	16	176	5040	1.85E+10	8.0800E+09	6.7941E+12	1.4648E+14		1.5327E+14	
8	32	88	2520	1.85E+10	7.7960E+09	2.6885E+13	7.3240E+13		1.0013E+14	
9	64	44	1260	1.85E+10	7.7249E+09	1.0725E+14	3.6620E+13		1.4387E+14	
10	128	22	630	1.85E+10	7.7072E+09	4.2871E+14	1.8310E+13		4.4702E+14	
11	256	11	315	1.85E+10	7.7027E+09	1.7146E+15	9.1550E+12		1.7237E+15	
12										
13	* Note: The DS FFT flops category is missing the initial FFT in range for each pulse. However, this only needs to be done once at a cost of ~2e10 flop									

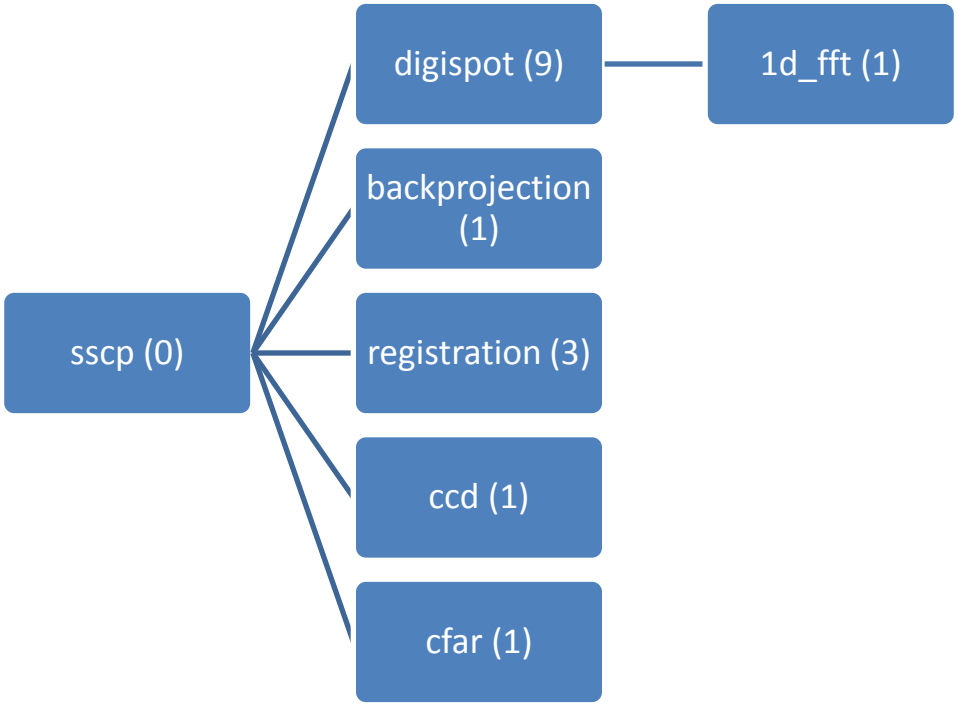


Prediction Techniques Ranked

	Speed	Ease	Flexibility	Accuracy	Scalability
Ad-hoc Analytical Models	1	3	2	4	1
Structured Analytical Models	1	2	1	4	1
Aspen	1	1	1	4	1
Simulation – Functional	3	2	2	3	3
Simulation – Cycle Accurate	4	2	2	2	4
Hardware Emulation (FPGA)	3	3	3	2	3
Similar hardware measurement	2	1	4	2	2
Node Prototype	2	1	4	1	4
Prototype at Scale	2	1	4	1	2
Final System	-	-	-	-	-

DARPA UHPC CP #1 Model Hierarchy

- Synthetic Aperture Radar
- Modeled pipeline for 1 image (v1)
  - Resources
    - Flops (including fmad vs. sin/cos vs. sqrt)
    - Loads/Stores
  - Control Flow
  - Data Sizes
- By The Numbers
  - 7 .aspen files
  - 16 kernels
  - ~350 lines of Aspen
    - including comments, spacing



Each block is a model/file, edges are imports, and the number of kernels is shown in parenthesis.

K. Spafford, J.S. Vetter, T. Benson, and M. Parker, "Modeling Synthetic Aperture Radar Computation with Aspen," *International Journal of High Performance Computing (IJHPC), (to appear), 2013,*

CCD: analytical model (1)

```
for(m=0; m<Mwins; m++) {  mIndex = (Ncor-1)/2 + m;        iter=57018
  for(n=0; n<Nwins; n++) {  nIndex = (Ncor-1)/2 + n;      iter=57018
    for(i=mIndex-(Ncor-1)/2, k=mu_f.real=mu_f.imag=mu_g.real=mu_g.imag=0;
        i<=mIndex+(Ncor-1)/2; i++)                        iter=5
      for(j=nIndex-(Ncor-1)/2; j<=nIndex+(Ncor-1)/2; j++, k++) {  iter=5
        f[k].real= curImage[i][j].real; f[k].imag= curImage[i][j].imag;
        g[k].real= refImage[i][j].real; g[k].imag= refImage[i][j].imag;
        mu_f.real+= curImage[i][j].real; mu_f.imag+= curImage[i][j].imag;
        mu_g.real+= refImage[i][j].real; mu_g.imag+= refImage[i][j].imag;}
    mu_f.real /= Ncor*Ncor; mu_f.imag /= Ncor*Ncor;
    mu_g.real /= Ncor*Ncor; mu_g.imag /= Ncor*Ncor;
    for(k=num.real=num.imag=den1=den2=0; k<Ncor*Ncor; k++) {   iter=25
      f[k].real -= mu_f.real; f[k].imag -= mu_f.imag;
      g[k].real -= mu_g.real; g[k].imag -= mu_g.imag;
      num.real += f[k].real*g[k].real + f[k].imag*g[k].imag;
      num.imag += f[k].real*g[k].imag - f[k].imag*g[k].real;
      den1 += f[k].real*f[k].real + f[k].imag*f[k].imag;
      den2 += g[k].real*g[k].real + g[k].imag*g[k].imag;}
    corr_map[m][n].x = nIndex; corr_map[m][n].y = mIndex;
    if(den1 != 0.0 && den2 != 0.0)
      corr_map[m][n].p=sqrtf((num.real*num.real+num.imag*num.imag)/(den1*den2));
    else corr_map[m][n].p = 0.0;
}}
```

4

4

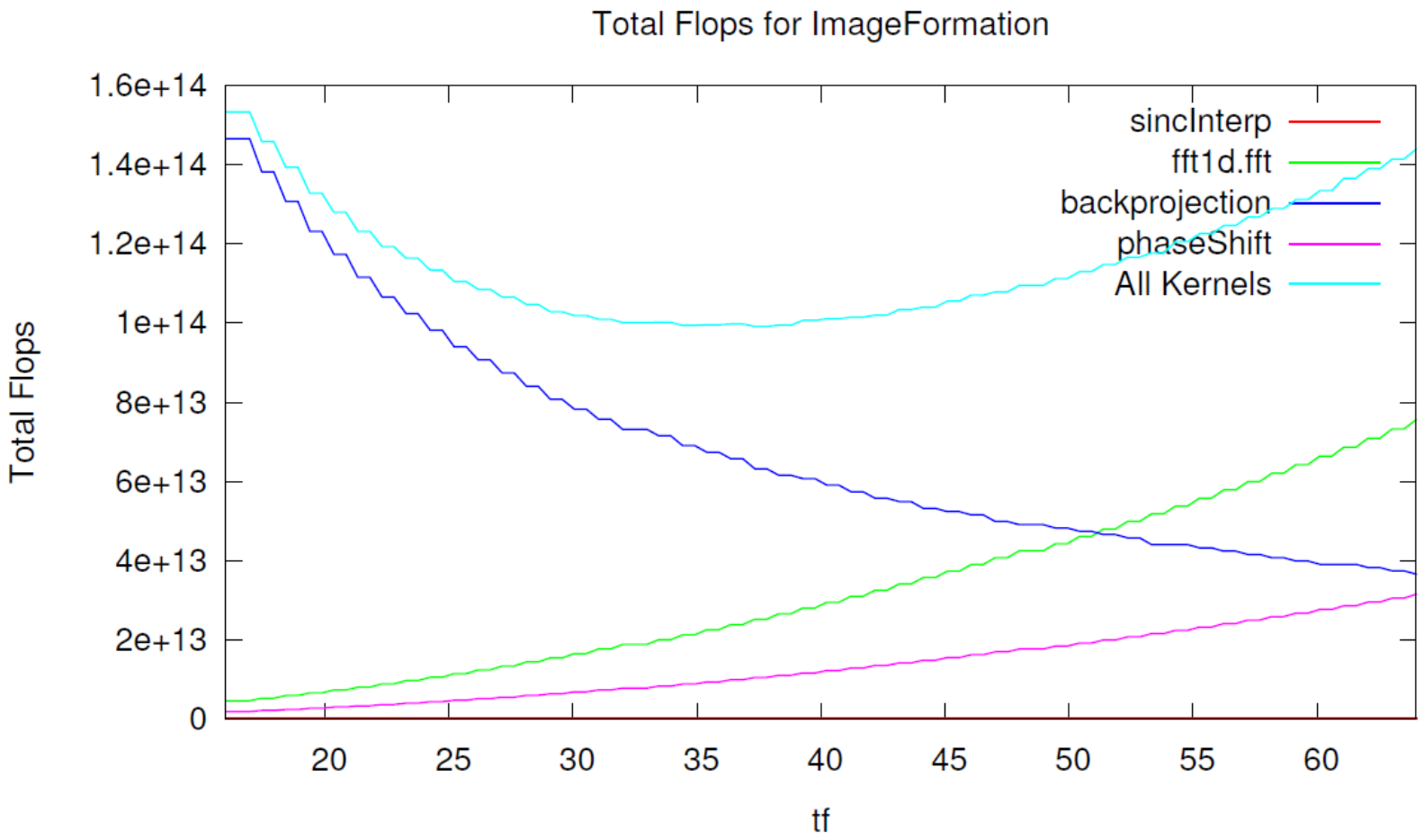
20

6

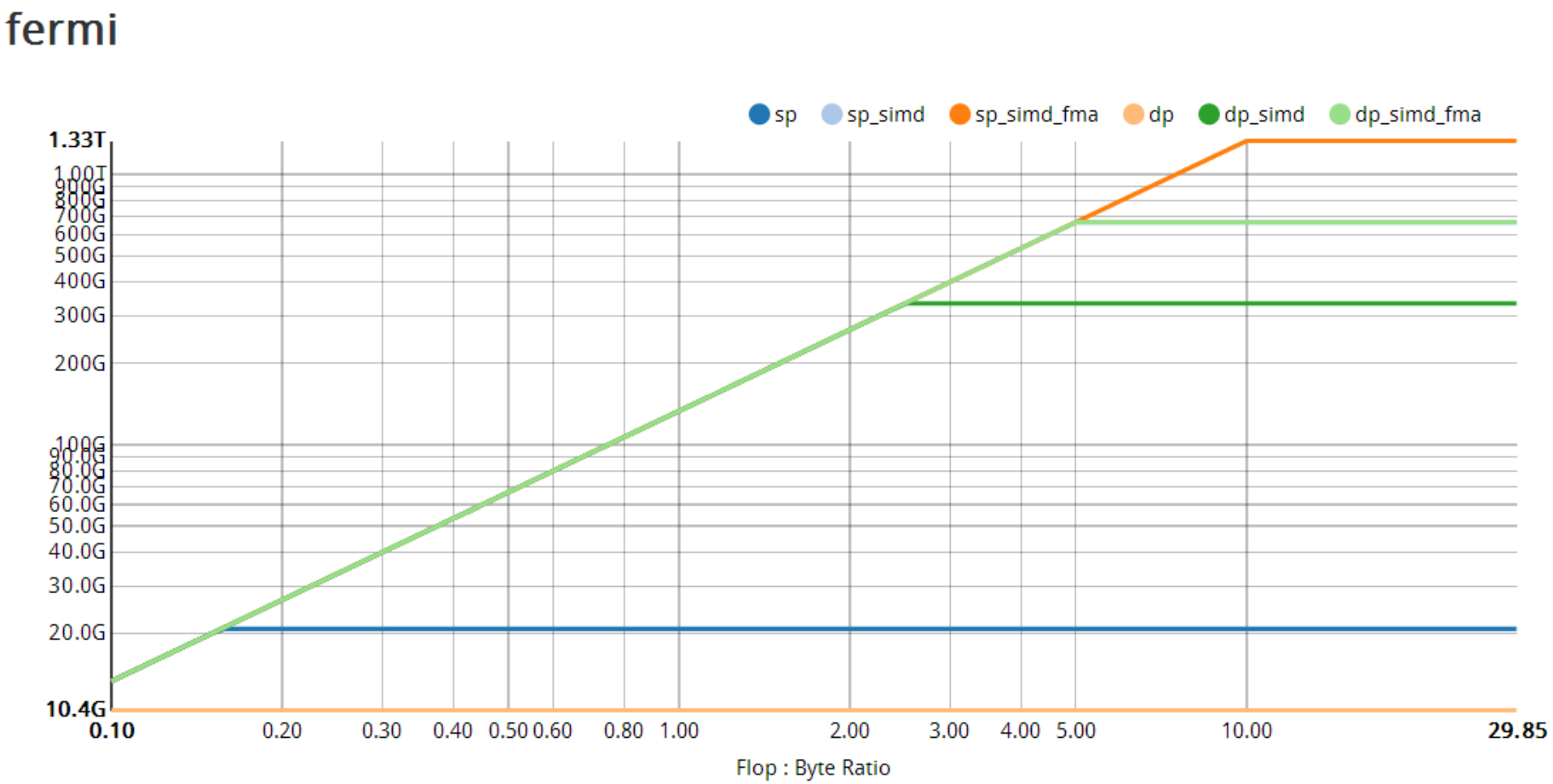
CCD: Aspen model

```
model ccd {

  param tileEdge = 5 // neighborhood size (5x5), aka nCor
  param tileSize = tileEdge * tileEdge
  param imgWidth = 57018
  param imgHeight = 57018
  param wordSize = 8

  param numXTiles = imgWidth - (tileEdge + 1)
  param numYTiles = imgHeight - (tileEdge + 1)

  kernel ccd {
    exposes parallelism [numXTiles * numYTiles]
    // First tile loop
    requires loads [2 * tileSize * wordSize] // from currImg, refImage,
        cached per tile
    requires flops [4 * tileSize] as simd // accumulate into mu_f, mu_g
    // Scale mu
    requires flops [4] as simd // scale mu
    // Second tile loop
    requires flops [4 * tileSize] as simd
    requires flops [16 * tileSize] as simd, fmad
    // Update 3 scalar values in corr_map
    requires flops [1] as simd, sqrt
    requires flops [2] as simd, fmad
    requires flops [3] as simd
    requires stores [3 * 4] // to corr_map
  }

  control main {
    ccd
  }
```

Understanding application specific tradeoffs in CP1



Automatic Rooflines, Powerlines, etc



Contributors and Sponsors

- Future Technologies Group: http://ft.ornl.gov
- US Department of Energy Office of Science
  - DOE Vancouver Project: https://ft.ornl.gov/trac/vancouver
  - DOE Blackcomb Project: https://ft.ornl.gov/trac/blackcomb
  - DOE ExMatEx Codesign Center: http://codesign.lanl.gov
  - DOE Cesar Codesign Center: http://cesar.mcs.anl.gov/
- Scalable Heterogeneous Computing Benchmark team: http://j.mp/shocmarks
- US National Science Foundation Keeneland Project: http://keeneland.gatech.edu
- US DARPA NVIDIA Echelon
- NVIDIA CUDA Center of Excellence at Georgia Tech
- DOE Exascale Efforts: http://science.energy.gov/ascr/research/computer-science/
- International Exascale Software Project: http://www.exascale.org/iesp/Main_Page

Q & A

More info: vetter@computer.org