

Application Characterization using Oxbow Toolkit and PADS Infrastructure

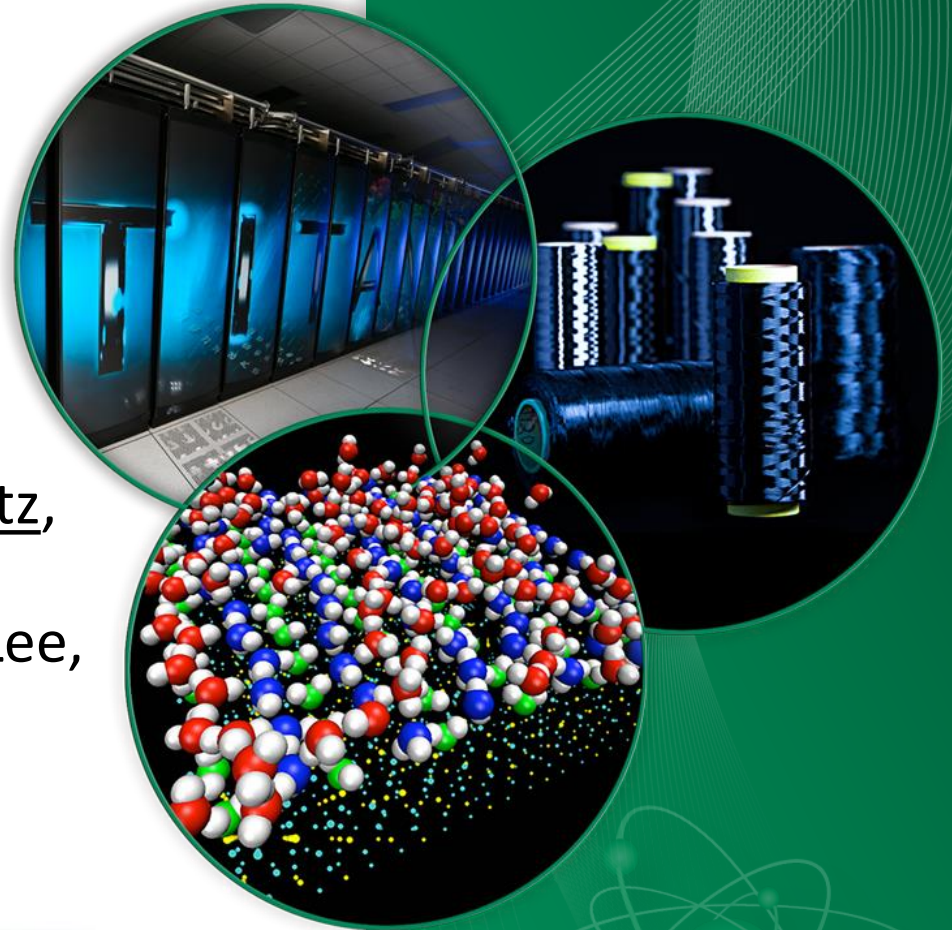
Sarat Sreepathi, Megan Grodowitz,
Robert Lim, Philip Taffet, Philip
Roth, Jeremy Meredith, Seyong Lee,
Dong Li, Jeffrey S. Vetter (PI)

Co-HPC Workshop

November 17 2014



<http://ft.ornl.gov>



Oxbow Project Overview

- Exascale architects and software developers need specific information about current and future DOE workloads
- Previous work developed a systemic process to investigate workload properties, resulting in new insights
- Oxbow project work pursues several interconnected goals
 - To characterize applications and proxy application in new ways and from many angles
 - To create a shared, open data store so that a community of researchers can share and compare results
 - To solicit feedback from the community on methods, metrics, applications, and tools
 - To evolve the toolkit to capture new application features

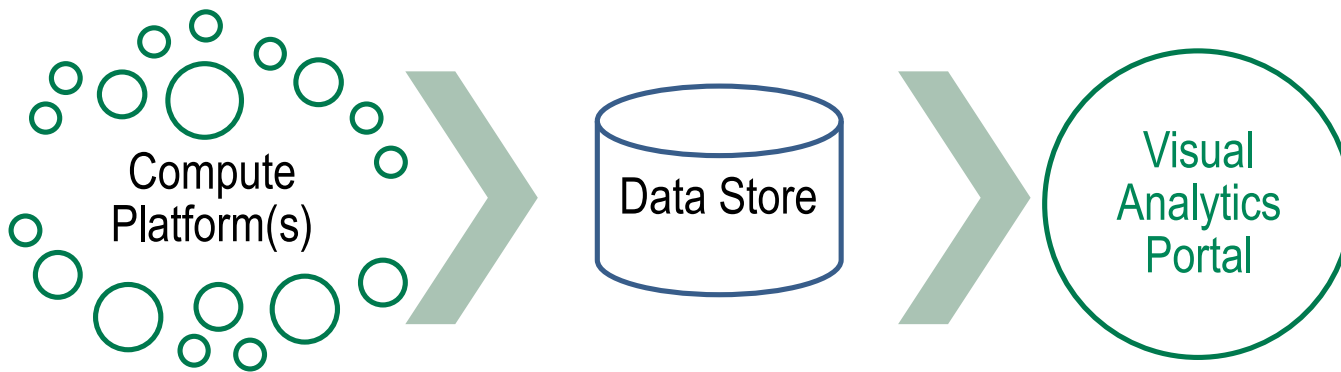
Oxbow Workflow Overview

Application Metrics:

- Computation
 - Instruction mix
- Communication
 - MPI Point to Point
 - MPI Collective
- Memory
 - Reuse Distance
 - Bandwidth estimate

System Statistics:

- Processor Type
 - Speed
 - Vendor/Architecture
- Memory Hierarchy
 - Cache Sizes/Layout
 - DRAM sizes
- Network
 - Protocol
 - Speed



Execute application(s) using Oxbow toolset.

Store results with various meta-data: **application, version, job size, date...**

Explore **dynamic visualization of experiment data** with desktop or mobile browser.

Collect application metrics: **computation, memory, communication...**

Automate uploads as part of Oxbow tool-chain (optional).

Download experiment data, or save data plot images.

Collect system statistics: **processor type, memory hierarchy, network**

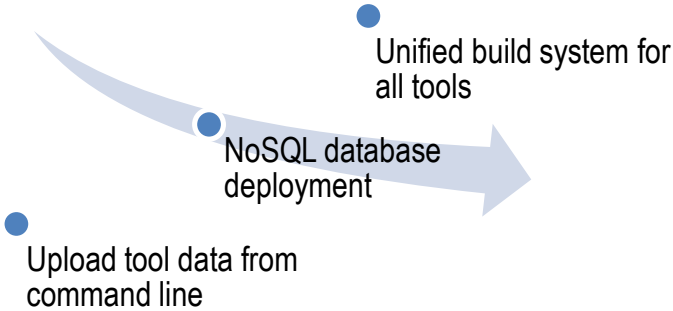
Retrieve previous experiment data by metadata.

Upload experiment data through web interface.

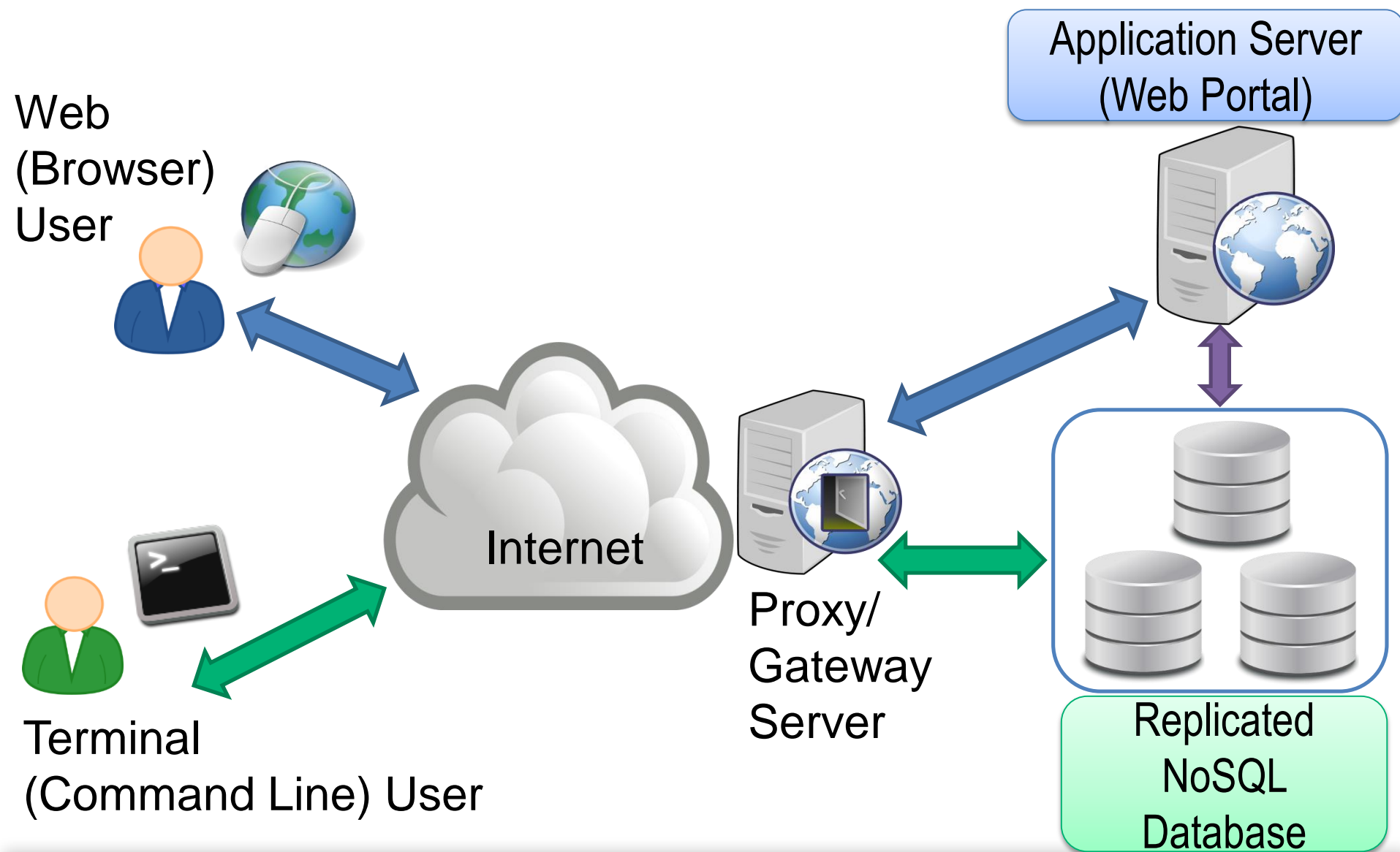
Tools produce results with human readable summary plus raw data files.

Collaborate to share results with other users.

Application Coverage

- Phase 1 (tools only): Q3 2012 – Q2 2013
 - HPCB Benchmark kernels
 - AMG unstructured grid linear solver
 - Nekbone Fluid dynamic proxy application of Nek5000
 - MOCFE neutron transport simulation
 - LULESH shock hydrodynamics code
 - S3D turbulent combustion numerical modelling
 - SPASM short range molecular dynamics
 - GTC particle-in-cell method
 - ddcMD classical molecular dynamics
 - LAMMPS large scale atomic/molecular dynamics simulation
 - Nek500 computational fluid dynamics solver
 - POP Ocean circulation model
 - Phase 2 (web infrastructure): Q3 2013 – Q3 2014
 - XSBench reactor core particle transport cross section lookup proxy application for OpenMC
 - HPCG sparse matrix solver, new top500 benchmark
 - AMGmk kernels from AMG application perform sparse matrix vector multiply, mesh relaxation, vector dot product
 - NEKbonemk microkernel from Nekbone and SIMD compiler challenge
 - UMT / UMTmk performs three dimensional, non linear, radiation transport calculations using deterministic (Sn) method
 - KMI Hash generate and do series of lookups on database of genome sequences
 - QMCPack quantum monte carlo simulation code
 - miniAMR adaptive mesh refinement miniapp
 - BoxLib adaptive mesh refinement proxy application set
 - LSMS first principles ground calculations of solid state systems using WL Monte Carlo walkers method
 - MPAS Climate modeling
 - EAVL Big data visualization and analysis
 - Visit visualization tool
 - MCB simple heuristic transport modeling using Monte Carlo methods
 - RSBench neutronics proxy application
- 
- Upload tool data from command line
- NoSQL database deployment
- Unified build system for all tools

Web Portal Design



Computation

Computation Characterization

- Instruction Mix Tool

- Decode native x86 instructions into RISC-like micro-operations

- Benefits

- Actual instruction stream after compiler optimizations

- Captures compiler generated auxiliary instructions (address arithmetic, spill/unspill, etc.)

- Caveats

- Mix impacted by compiler optimizations / architecture choices

- Can be difficult to attribute results to application structures, components

- Group instructions into coarse categories:

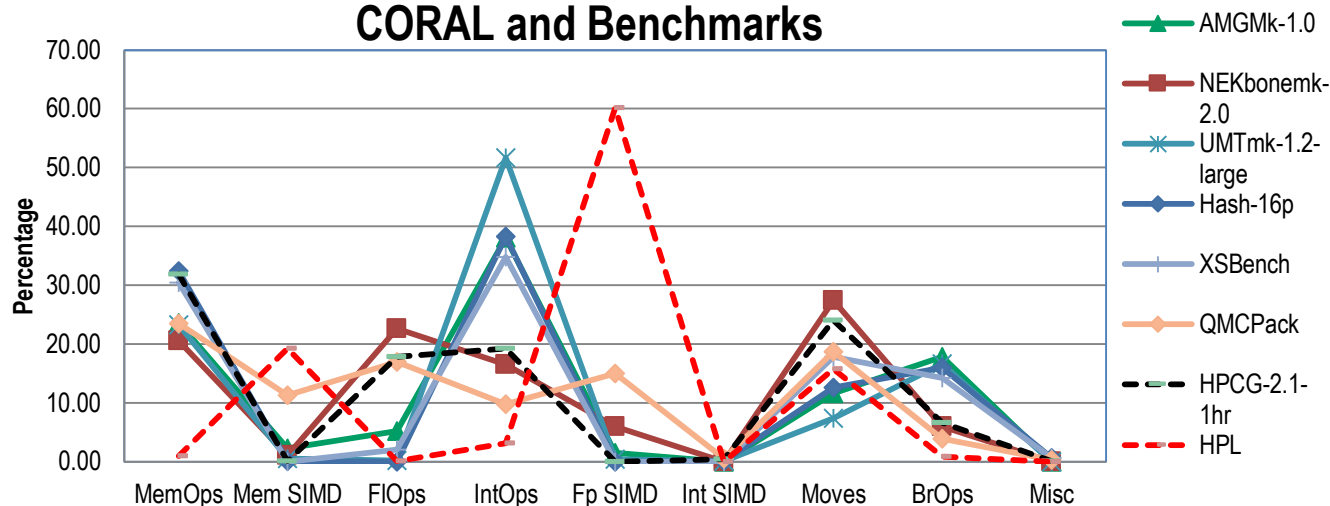
- Memory, integer & floating-point arithmetic, register moves, branches, other

Computation Characterization

Recent Results

Nekbone is markedly different from the other microkernels. It more closely matches QMCPack and HPCG.

Instruction Mix Comparison
CORAL and Benchmarks



HPCG shows a good match for typical applications, heavy on memory and moves, with moderate floating point and integer operations.

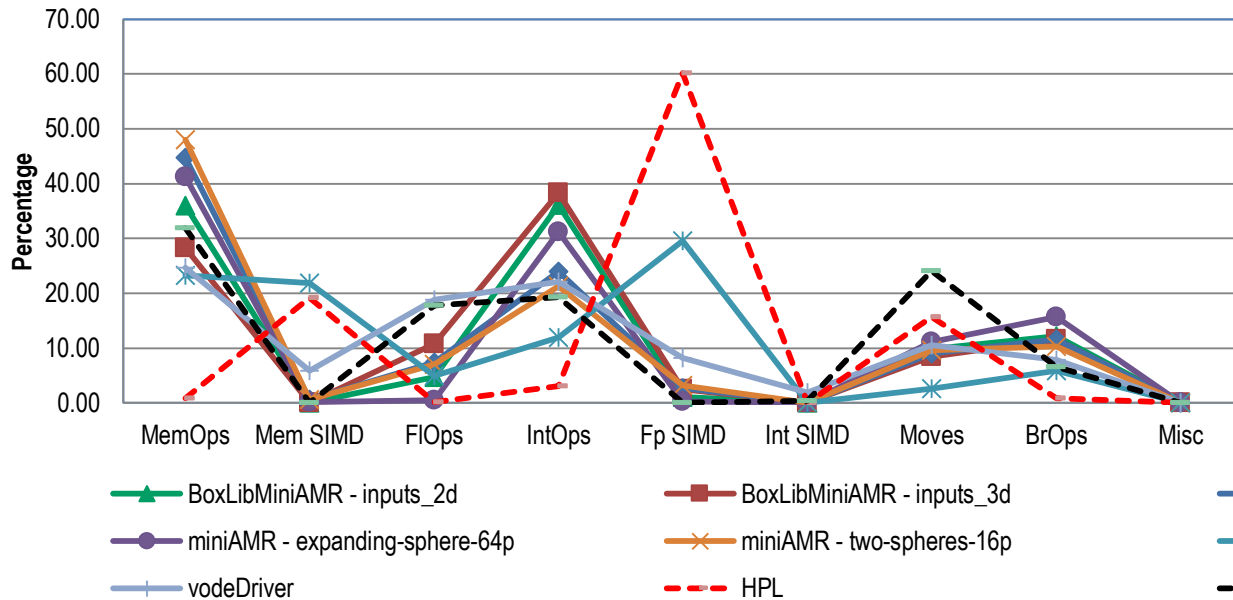
HPCG matches well with QMCPack, despite implementing very different problem type.

	MemOps	Mem SIMD	FIOps	IntOps	Fp SIMD	Int SIMD	Moves	BrOps	Misc
AMGMk-1.0	23.58	2.25	5.16	38.10	1.45	0.00	11.65	17.81	0.00
NEKbonemk-2.0	20.55	1.09	22.61	16.53	5.94	0.00	27.36	5.93	0.00
UMTmk-1.2-large	23.22	0.50	0.24	51.69	0.37	0.03	7.33	16.62	0.01
Hash-16p	32.38	0.08	0.00	38.31	0.00	0.02	12.63	15.92	0.64
XSBench	30.48	0.00	2.02	34.75	0.00	0.00	17.82	14.27	0.66
QMCPack	23.50	11.34	16.92	9.76	15.01	0.56	18.70	3.94	0.28
HPL	0.9	19.2	0.1	3.1	60.2	0	15.7	0.8	0
HPCG-2.1-1hr	31.85	0.040	17.76	19.23	0.002	0.40	24.034	6.62	0.036

Surprisingly, HASH and UMTmk are very similar, despite one being a microkernel and the other being a data-centric application

Computation Characterization Recent Results + AMR codes

Instruction Mix Comparison - AMR apps



HPCG does a reasonable job at representing vodeDriver.

HPL represents EXP_CNS_NoSpec instruction mix

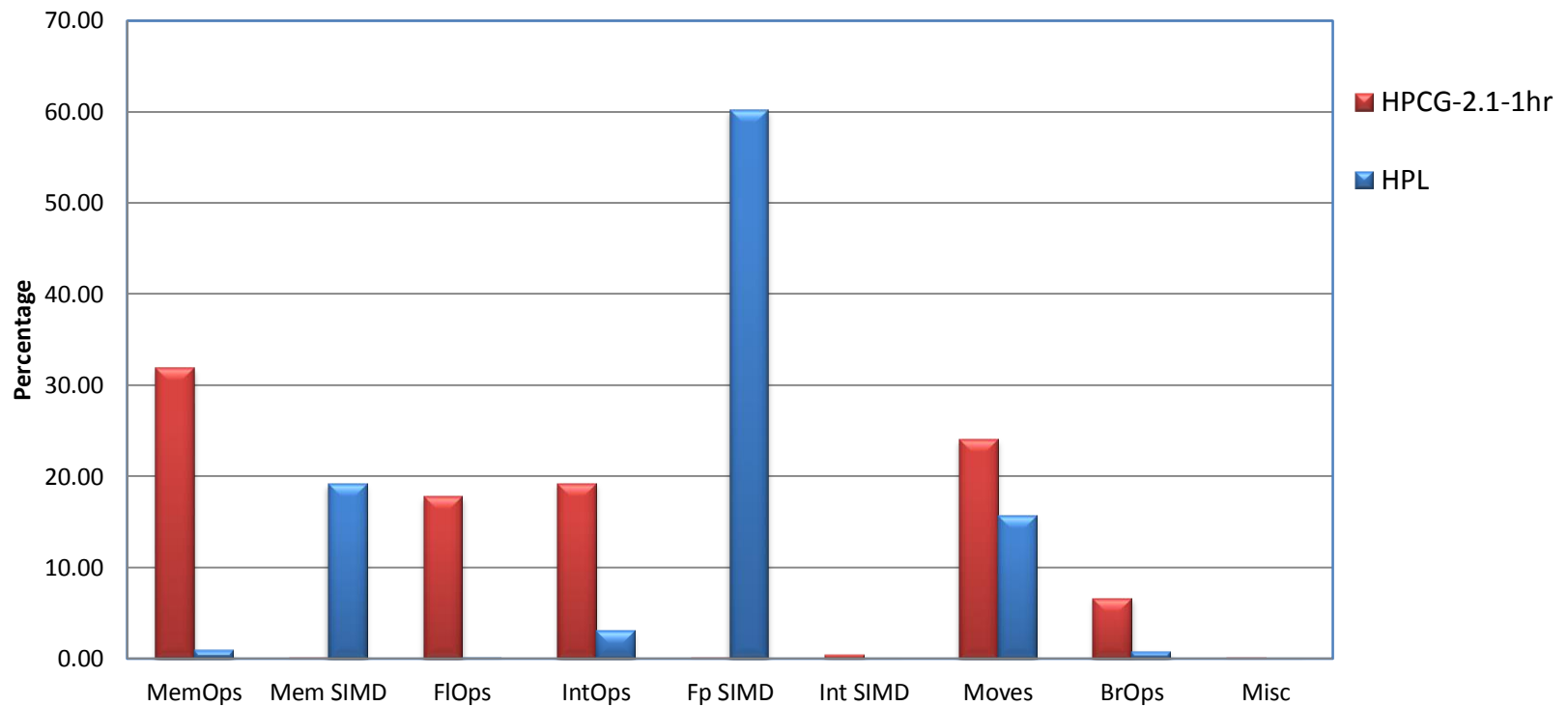
Other AMR miniapps are so Integer intensive that neither top500 benchmark matches particularly well to computational requirements

	MemOps	Mem SIMD	FIOps	IntOps	Fp SIMD	Int SIMD	Moves	BrOps	Misc
BoxLibMiniAMR - inputs_2d	35.94	0.06	4.73	36.08	1.12	0.03	9.78	12.15	0.11
BoxLibMiniAMR - inputs_3d	28.30	0.07	10.73	38.26	2.53	0.00	8.48	11.56	0.07
Exp_CNS_NoSpec	23.22	21.91	4.96	11.99	29.53	0.00	2.56	5.83	0.00
miniAMR - two-spheres-16p	48.01	0.60	7.02	21.26	3.19	0.00	9.66	10.26	0.00
miniAMR - sphere-diagonal-27p	44.69	0.56	7.36	23.95	2.75	0.00	9.04	11.64	0.01
miniAMR - expanding-sphere-64p	41.18	0.15	0.51	31.15	0.24	0.02	11.08	15.66	0.01
vodeDriver	24.64	5.91	18.77	22.05	8.22	1.87	10.57	7.91	0.06
HPL	0.9	19.2	0.1	3.1	60.2	0	15.7	0.8	0
HPCG-2.1-1hr	31.86	0.04	17.77	19.24	0.00	0.40	24.03	6.62	0.04

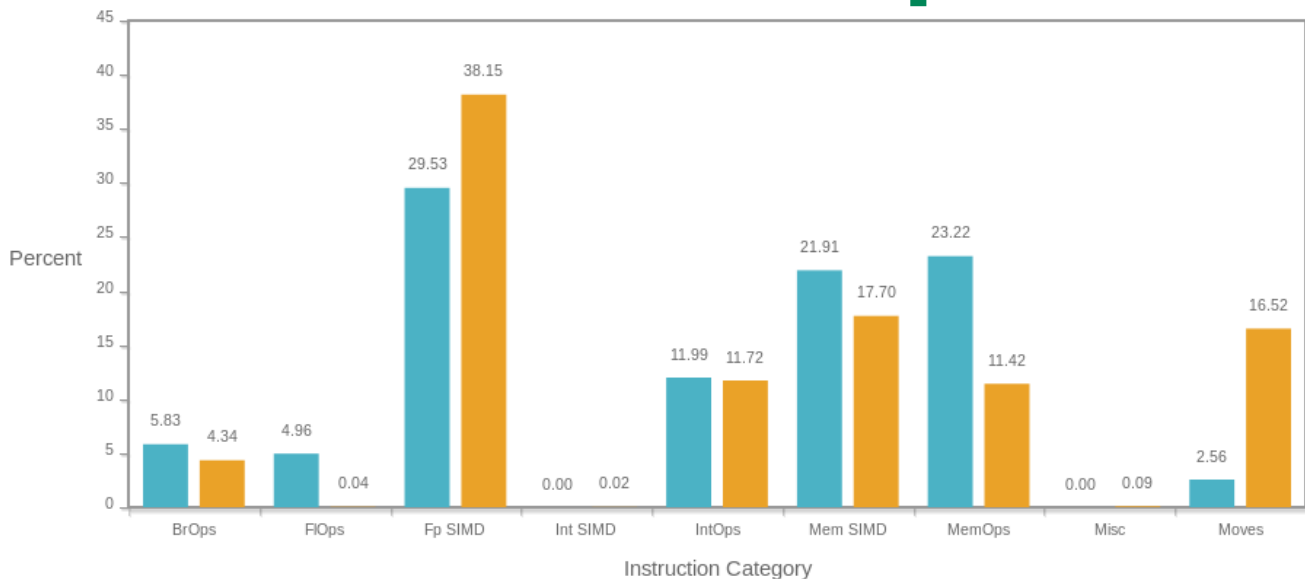
Comparing Top 500 benchmarks

Instruction Mix Comparison HPCG 2.1 vs. HPL

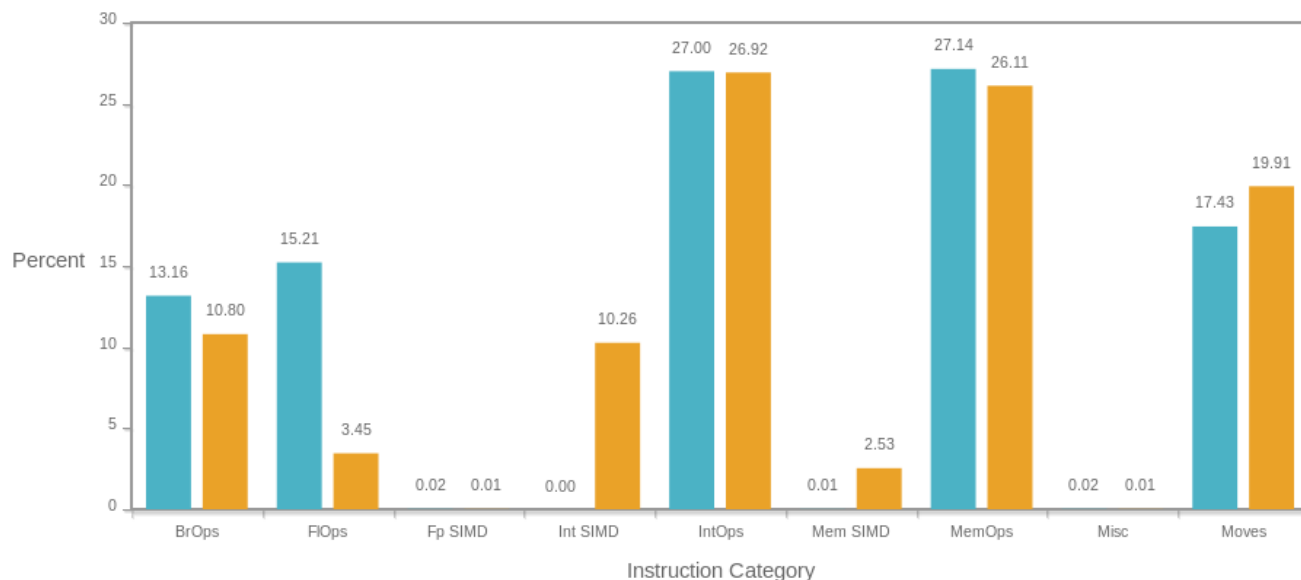
A closer look at the differences between HPL and HPCG shows that HPCG uses much fewer SIMD optimizations, and increases memory operations



Instruction mix comparisons



Exp_CNS_NoSpec (CFD:Compressible Navier-Stokes) and HPL are similar

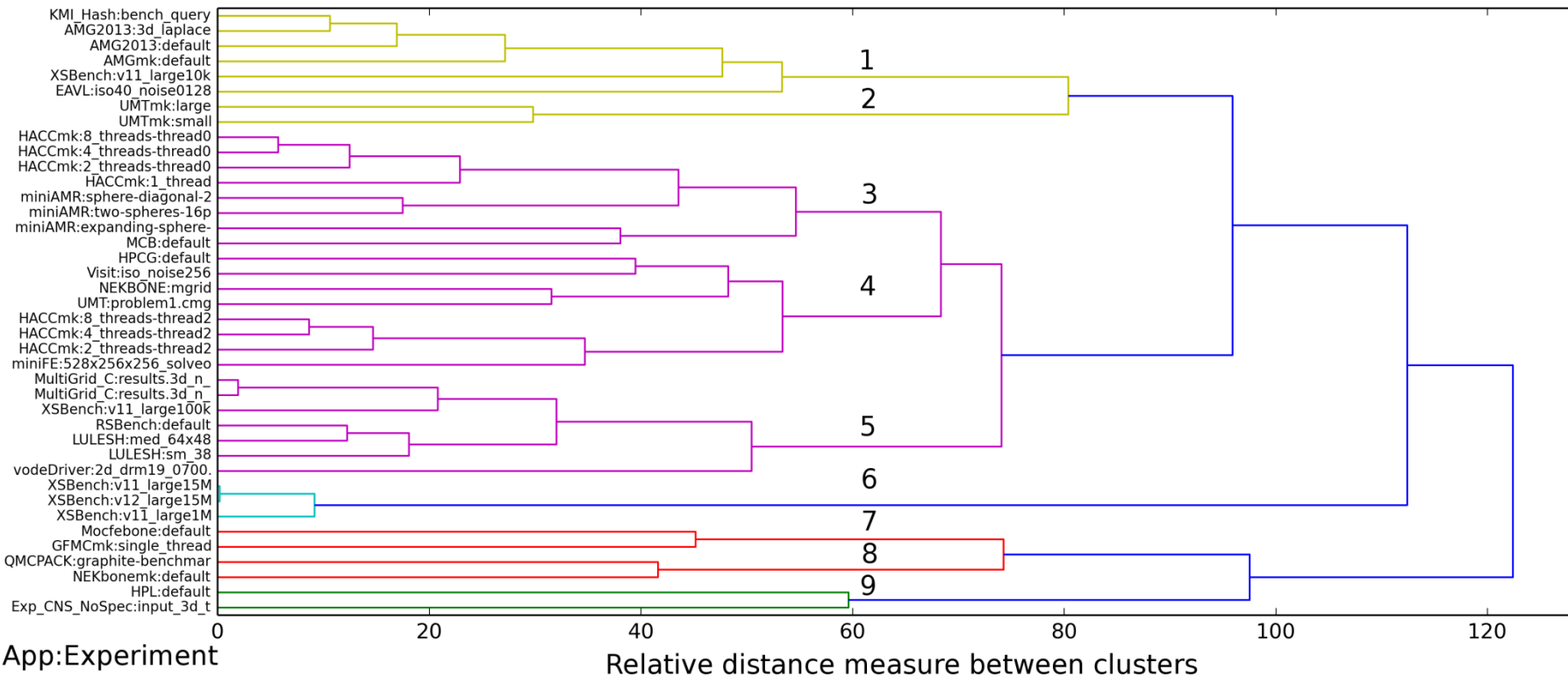


HPCG and VisIt mostly differ in the percentage of Int SIMD and FLOps instructions.

Moreover, VisIt is an outlier among the instruction mix profiles with 10.26% Int SIMD instructions

Instruction Mix Clustering

Dendrogram: Hierarchical Clustering

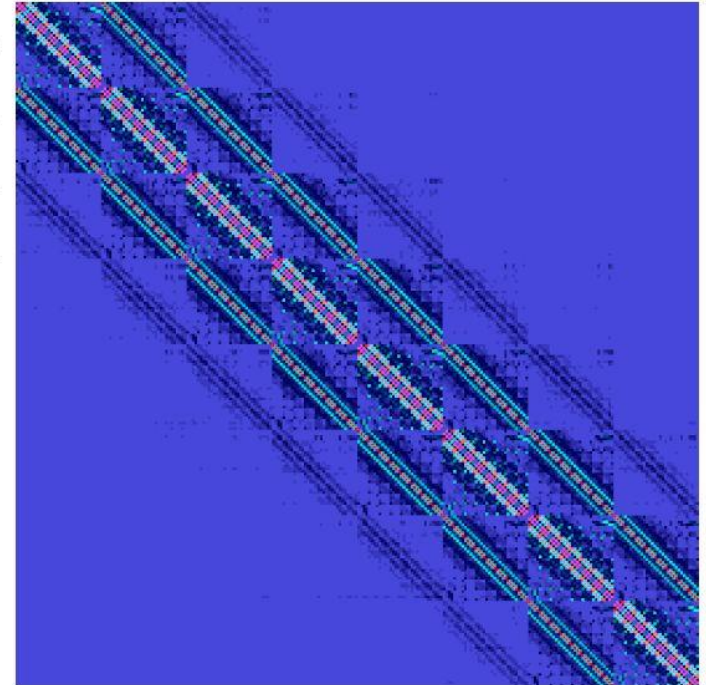
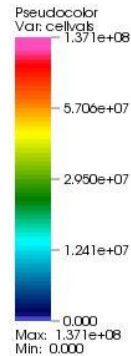


As the amount of disparate experimental data grows, traditional method of comparing results become more difficult, less revealing of larger relationships. Clustering reveals relationships between many different experiments with different experimental designs.

Communication

Communication Characterization with Enhanced mpiP

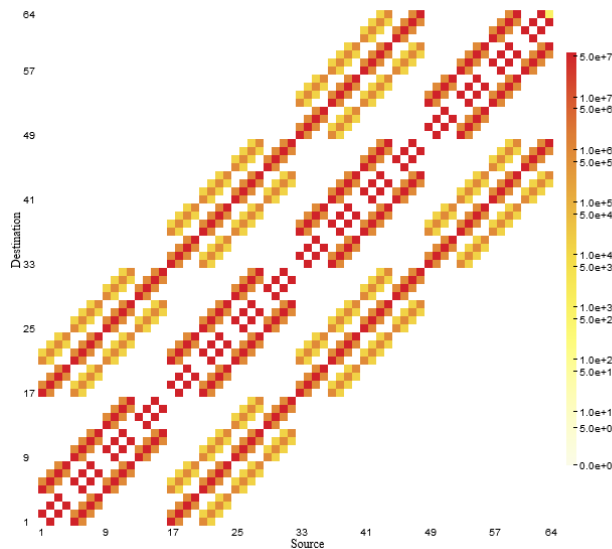
- Lightweight tool for profiling MPI operations
 - Supports MPI communications and I/O
 - Link-time library, uses PMPI profiling interface
 - Measures time spent, number of calls to MPI operations aggregated by call site
 - <http://sourceforge.net/projects/mpip>
- Used enhanced mpiP
 - Captures communication topology, message size histograms for point-to-point operations
 - Captures message size histograms for collective operations



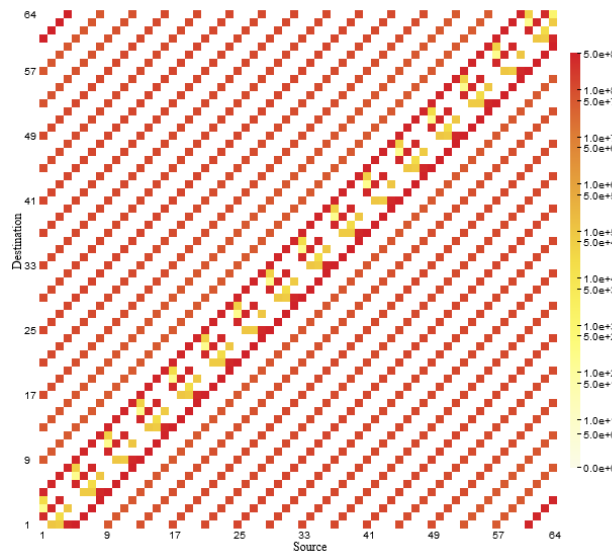
Data from AMG2000 benchmark running on Keeneland Initial Delivery System

Pattern indicates a mostly nearest neighbor communication pattern with some extended communications

HPCG & HPL Communication Volume



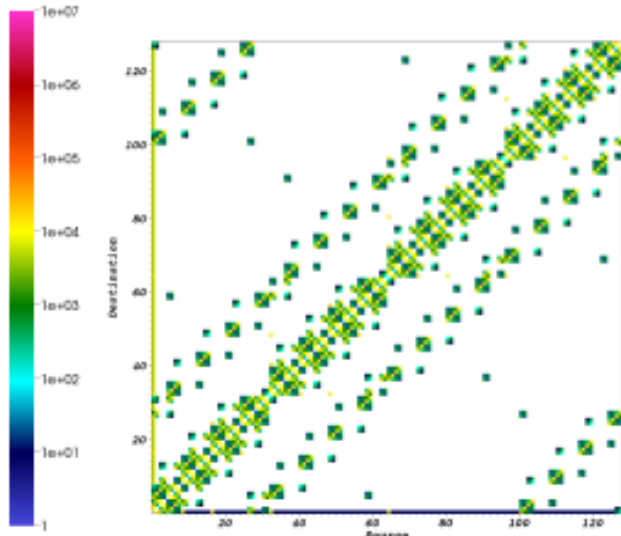
HPCG



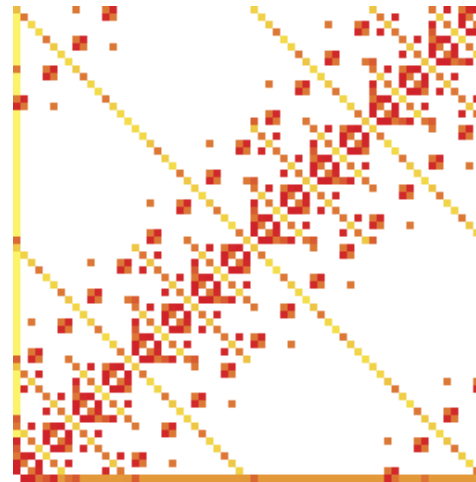
HPL

- Pattern characteristics:
 - HPCG: Wider spread indicates communication beyond just nearest neighbors
 - HPL: Communication spread throughout
- Compared to other apps
 - Lack of random scattering is similar to other benchmarks. Real applications are not so orderly
 - HPCG is more similar to other apps than HPL, which does not resemble realistic communication
- Observations from using multiple tools
 - HPCG instruction mix matches applications, but differences appear when looking at communication
- Implications for system design
 - System optimized for HPL comm pattern would try to balance communication between all nodes
 - System optimized HPCG comm pattern would emphasize tight locality of communication
 - Applications with scattered communication (like Nek5000) might need different system capabilities

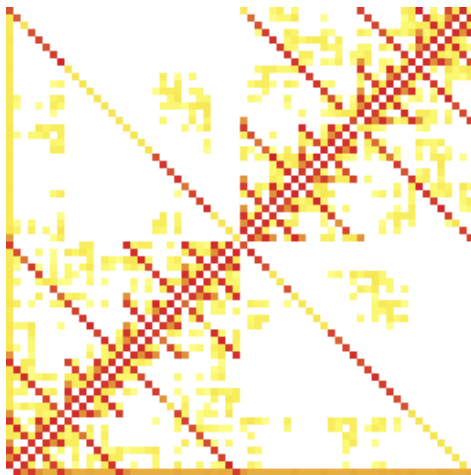
Nek: Communication patterns



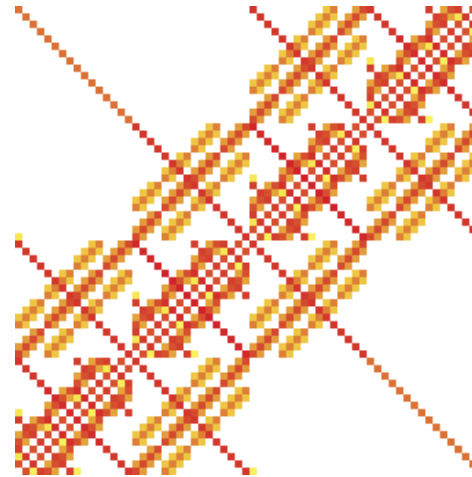
Nek5000, 128 tasks (from PBMS13 results)



Nek5000 eddy, 64 tasks



Nek5000 vortex, 64 tasks



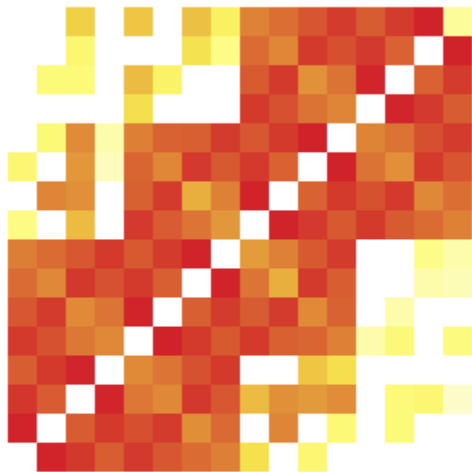
Nekbone multigrid preconditioner, 64 tasks

Nekbone : communication behavior doesn't change much between scenarios

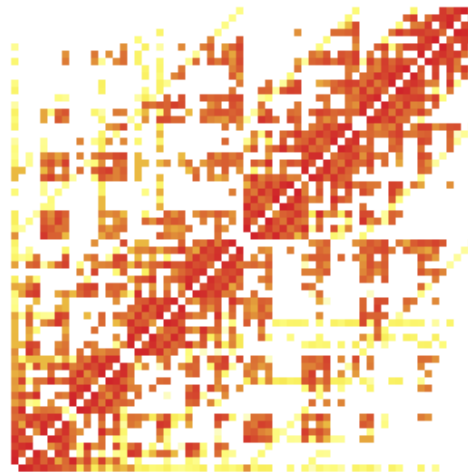
Nek5000: Some configurations resemble Nekbone very closely. Other configurations (vortex) show very different communication patterns.

As with other results, real applications can create more asymmetric patterns, whereas proxy apps tend to be very symmetrical.

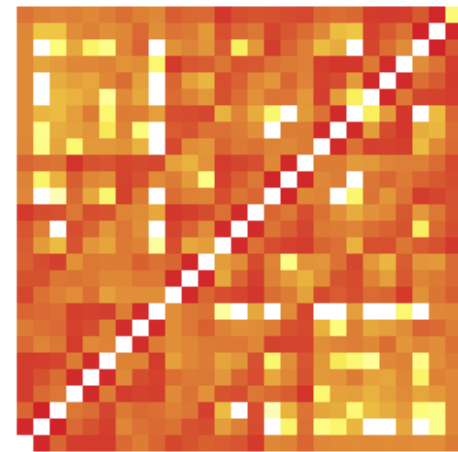
AMR proxy applications



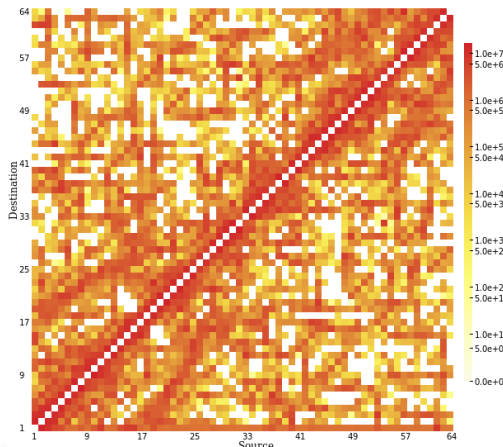
miniAMR two spheres
(16 ranks)



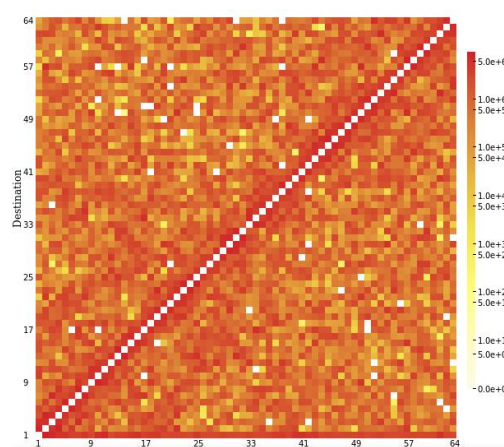
miniAMR expanding sphere
(64 ranks)



miniAMR sphere diagonal
(27 ranks)



BoxLib: AMR_Adv_Diff
inputs_3d_regrid_none (64
rank)



BoxLib: AMR_Adv_Diff
inputs_3d_regrid_4ts (64
rank)

- Different inputs generate very different communication patterns
- Neither HPL nor HPCG represent the communication patterns of AMR apps with objects in motion
- Aids in validation of expected communication behavior

Memory Tools Overview

Memory Access Characterization

- **Memory Bandwidth**

$$Bdwth = \frac{M_{req} S_{req}}{T}$$

- Mreq – # memory requests, Sreq – request size, T – time
- Memory request count and execution time measured using hardware performance counters
- Using two start() and stop() caliper functions
 - Based on PAPI
 - Caliper calls inserted around code region of interest

Memory Access Characterization

- **Reuse Distance:** the number of distinctive data elements accessed between two consecutive references to the same element

time: 1 2 3 4 5 6 7 8 9 10 11 12
access: **d a c b c c g e f a f b**
distance: |← 5 *distinct accesses* →|

(a) An example access sequence.

- Fig courtesy of the paper “Predicting Whole-Program Locality through Reuse Distance Analysis”

- Benefits of using reuse distance

- Quantify program locality
- Allows direct comparison of data behavior across applications

- Challenges

- High time cost ($O(N^2)$) for a memory trace of length N
- High storage cost ($O(N)$) for a memory trace of length N

Portal



Welcome to Oxbow PADS Portal.

It facilitates exploration of performance data in our data store.
Please visit the [main Oxbow website](#) for an overview of the project.

Applications 41

Number of applications: 41

- * AMG2013
- * AMGmk
- * BoxLibMiniAMR
- * CAM-SE
- * CIAN
- * CoEVP
- * CoMD
- * ddcMD
- * EAVL
- * Exp_CNS_NoSpec
- * GFMCmk
- * GTC
- * HAOCmk
- * HPCG
- * HPCG
- * HPL
- * KML_Hash
- * LAMMPS
- * LSMS
- * LULESH
- * MCB
- * miniAMR
- * miniLFE
- * MOCFE
- * MPAS
- * MultiGrid_C
- * Nek5000
- * NEKBONE
- * OpenMC
- * POP
- * QMCPACK
- * RSBench
- * S3D
- * SNAP
- * SPaSM
- * UMT
- * UMTmk
- * Visit
- * vodeDriver
- * VPFPT
- * XSBench

Application Comparison

Start typing application names (autocomplete supported) and hit "Go" to compare.

Tools 5

- imix
- membw
- mpip
- reuse
- tau

Experiments 135

Total number of experiments: 135

Recent Experiments

Exp ID	App	AppVer	Tool	ToolVer	Exp	Exp Date	Upload User	Platform
5452638c53c4282d71c341c8	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-16agg-8stride-io_30d	2014-10-30 00:00:00	sarat	Keeneland
5452547853c428017b4f856b	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-32agg-4stride-io_30d	2014-10-30 00:00:00	sarat	Keeneland

Apps – Static analysis data

Applications

The following table shows the list of applications including static analysis metrics ([Cyclomatic Complexity](#), programming languages, types of parallelism, lines of code, etc.)

Click on a column name to sort. Hover over the Cyclomatic Complexity value for details.
Caveat: This metric may be inaccurate if the program uses languages other than C/C++.

App	Version	Web	Domain	Origin	Prog. Lang	Parallelism	LOC	#Files	#Functions	Cyclomatic Complexity	Details
AMG2013	2.3	Web	Parallel Algebraic Multigrid Solver	CORAL	C	MPI, OpenMP	74901	203	1229	10028	Details
AMGmk	1.0	Web	Algebraic multigrid solver (unstructured grids)	CORAL	C	OpenMP	1812	13	49	270	Details
BoxLibMiniAMR	BoxLib v2...	Web	miniApp - simple advection diffusion example	ExaCT Co-design center	Fortran 90, Perl, C	MPI, OpenMP	33767	99	40	185	Details
CAM-SE	1.3.6	Web	Climate - Community Atmospheric Model	CORAL Throughput Benchmark	Fortran 90, C	MPI, OpenMP	161540	750	855	5391	Details
CIAN	0.4	Web	Coupling and Data Analytics	CESAR	C++, C	MPI	14903	65	505	2803	Details
CoEVP	2014-08-21	Web	ViscoPlasticity Scale-Bridging	ExMatEx	C++	MPI, OpenMP	35203	221	2904	5102	Details
CoMD	1.1	Web	Molecular Dynamics	ExMatEx	C	MPI	4571	42	248	775	Details
ddcMD	r1845-201...	Web	Molecular Dynamics	ExMatEx	C	MPI, OpenMP	114925	593	2367	13857	Details
EAVL	2014-06-17	Web	Big Data analysis and visualization		C++	MPI, CUDA	68289	231	1587	4294	Details
Exp_CNS_NoSpec	BoxLib_co...	Web	computational fluid dynamics	ExaCT	Fortran 90, C, Fortran 77, C++	MPI, OpenMP	3140	21	18	121	Details
GFMCmk	Apr-30-20...	Web	Micro-kernel	CORAL benchmark	Fortran 77, Fortran 90	OpenMP	188	13			Details
GTC	2	Web	Fusion		Fortran 90	MPI	5208	28			Details
HACCmk	1.0	Web	Cosmology	CORAL benchmark	C	MPI, OpenMP	150	13	4	20	Details
HPCC	1.4.3	Web	Benchmark		C	MPI	26915	347	414	3934	Details
HPCG	2.4	Web	Benchmark		C++	MPI, OpenMP	2424	72	76	386	Details
HPCG	2.1	Web	Benchmark		C++	MPI, OpenMP	2364	71	75	362	Details
HPL	2.1	Web	Benchmark		C	MPI	15741	287	194	2007	Details
KML_Hash	1.1	Web	Genome assembly and mapping analysis	CORAL	C	MPI	4232	16	117	574	Details
LAMMPS	1 Feb 2014	Web	Molecular dynamics		C++	MPI	503894	3164			Details
LSMS	3_rev237	Web	Ground state calculations of solid state systems and statistical physics calculations with a focus on magnetic systems	Coral benchmark suite	C, Fortran 77, C++	MPI, OpenMP, CUDA	102623	669	2504	9822	Details
LULESH	2.0.3	Web	Shock Hydrodynamics	ExMatEx	C++	MPI, OpenMP	5432	8	297	1059	Details
MCB	20130723	Web	Modeling the solution of a simple heuristic transport equation using a Monte Carlo technique	Computation Co-design at LLNL	C++	MPI, OpenMP	13458	139	2504	9822	Details

s.ornl.aov...

Instruction Mix



Instruction Mix Data

The following table shows the distribution of instructions (percentages) for an application across various categories. It is dynamically generated from live performance data in PADS.

This information is obtained using [MIAMI](#), which is based on [Intel's PIN](#), a dynamic binary instrumentation tool. We discard NOP and Prefetch instructions while calculating total instructions.

The table heatmap is generated by categorizing a cell's value into six categories (represented by different colors). So, every cell is colored based on the magnitude of the value it contains.

Higher magnitudes are denoted by darker colors.

Click anywhere on a row to select for comparison.

Click on a column name to sort.

Click on the application name to see a summary chart.

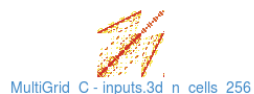
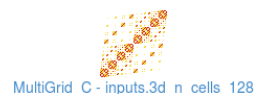
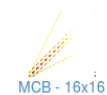
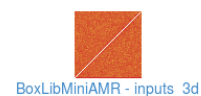
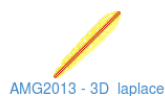
App	Exp	Platform	BrOps	FLOps	Fp SIMD	Int SIMD	IntOps	Mem SIMD	MemOps	Mis	Moves	Details	Compare
AMG2013	3d_laplace	KIDS	17.87	0.00	0.00	0.24	38.79	0.84	29.76	0.09	12.40	Details	<input type="checkbox"/>
AMG2013	default	KIDS	21.70	0.02	0.01	0.07	34.38	0.25	33.54	0.03	9.99	Details	<input type="checkbox"/>
AMGmk	default	KIDS	17.81	5.16	1.45	0.00	38.10	2.25	23.58	0.00	11.65	Details	<input type="checkbox"/>
BoxLibMiniAMR	inputs_3d	Neer	11.56	10.73	2.53	0.00	38.26	0.07	28.30	0.07	8.48	Details	<input type="checkbox"/>
BoxLibMiniAMR	inputs_2d	Neer	12.15	4.73	1.12	0.03	36.08	0.06	35.94	0.11	9.78	Details	<input type="checkbox"/>
BoxLibMiniAMR	inputs_3d-64p	Neer	19.86	0.42	0.10	0.01	37.89	0.01	37.41	0.02	4.28	Details	<input type="checkbox"/>
BoxLibMiniAMR	inputs_2d-64p	Neer	21.09	0.00	0.00	0.01	38.40	0.01	37.08	0.00	3.41	Details	<input type="checkbox"/>
BoxLibMiniAMR	inputs_3d	Keeneland	10.69	2.18	3.36	0.28	35.79	2.89	35.78	0.04	8.99	Details	<input type="checkbox"/>
BoxLibMiniAMR	inputs_2d	Keeneland	12.13	0.34	0.51	0.12	32.38	0.79	43.87	0.21	9.65	Details	<input type="checkbox"/>
EAVL	iso40_noise0128	KIDS	8.76	0.18	0.00	0.00	38.99	0.32	24.62	0.00	27.12	Details	<input type="checkbox"/>
Exp_CNS_NoSpec	input_3d_timesteponly	KIDS	5.83	4.96	29.53	0.00	11.99	21.91	23.22	0.00	2.56	Details	<input type="checkbox"/>
GFMCmk	single_thread	KIDS	9.73	0.13	21.54	0.28	28.23	20.09	18.77	0.00	1.22	Details	<input type="checkbox"/>
HACCmk	1_thread	KIDS	6.36	0.01	7.18	0.97	20.83	1.48	43.05	0.00	20.12	Details	<input type="checkbox"/>
HACCmk	8_threads-thread0	KIDS	6.72	0.01	1.03	0.14	23.04	0.21	48.88	0.00	19.96	Details	<input type="checkbox"/>
HACCmk	8_threads-thread2	KIDS	17.84	0.00	14.30	1.94	24.76	2.94	25.65	0.00	12.57	Details	<input type="checkbox"/>
HACCmk	2_threads-thread0	KIDS	6.72	0.01	3.85	0.52	22.12	0.79	46.06	0.00	19.92	Details	<input type="checkbox"/>
HACCmk	2_threads-thread2	KIDS	17.06	0.00	16.28	2.18	23.67	3.35	24.39	0.00	13.06	Details	<input type="checkbox"/>
HACCmk	4_threads-thread0	KIDS	6.74	0.01	2.01	0.28	22.73	0.41	47.88	0.00	19.93	Details	<input type="checkbox"/>
HACCmk	4_threads-thread2	KIDS	18.52	0.00	12.63	1.70	25.68	2.60	26.73	0.00	12.15	Details	<input type="checkbox"/>
HPCG	default	KIDS	13.16	15.21	0.02	0.00	27.00	0.01	27.14	0.02	17.43	Details	<input type="checkbox"/>
HPL	default	KIDS	4.34	0.04	38.15	0.02	11.72	17.70	11.42	0.09	16.52	Details	<input type="checkbox"/>
s.oml.aov...													

Communication

Communication Behavior

The following figures provide a preview of the communication patterns of various applications. They are dynamically generated from live performance data in PADS.

An application's message volume and counts are obtained using [mpiP](#), a lightweight profiling library for MPI applications. It is used to collect basic communication characteristics (point-to-point and collective calls) as well as the communication topology.



s.ornl.aov...

Communication Details

Communication Volume

Application: MultiGrid_C

Experiment: inputs.3d_n_cells_512

The figure below shows the communication pattern (message volume in bytes) for the above application.

The axes denote the source and destination MPI ranks.

Hover over the plot area to see specific statistics for a (source, destination) pair.

The boxplot (right) shows the distribution of message volume in bytes.

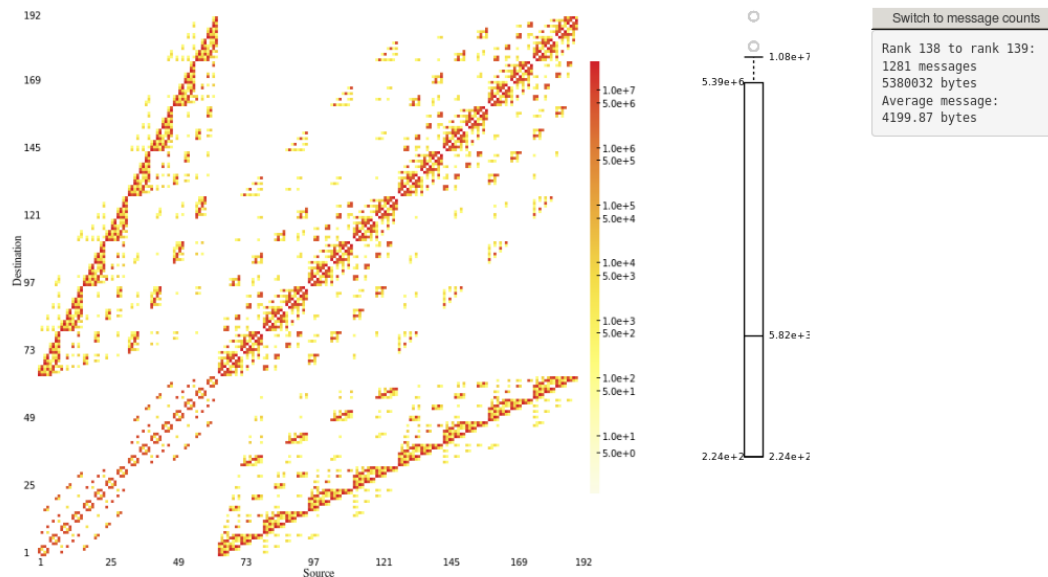
The box is enclosed by the ends of first and third quartiles of data. So, the box contains the middle 50% of the data.

The horizontal line inside the box denotes the median.

The whiskers (when visible) show the farthest points that are not outliers (within 3/2 times Q1-Q3 interquartile range). Any outliers are indicated by circles.

See [this](#) for an introduction to boxplot.

Details



Clustering



Clustering based on Instruction Mix Data

Clustering is performed on the instruction mix data across the following nine dimensions (various instruction categories).

BrOps FLOps Fp SIMD Int SIMD IntOps Mem SIMD MemOps Misc Moves

We use the hierarchical/agglomerative clustering approach. This is dynamically generated from live performance data in PADS.

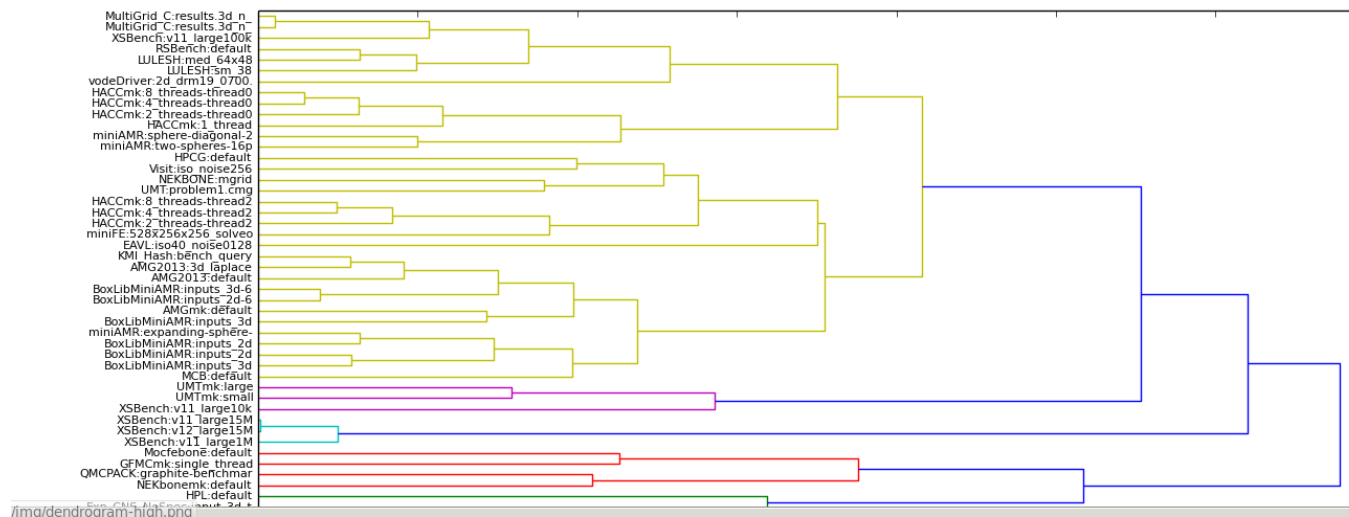
The algorithm begins with a forest of clusters that will finally form a hierarchical tree. At the outset, each data point forms its own cluster. When two clusters a, b from this forest are combined into a single cluster c, a and b are removed from the forest, and c is added to the forest. When only one cluster remains in the forest, the algorithm stops, and this cluster becomes the root.

We use the euclidean distance as the distance metric between any two data points. At every iteration, each cluster is joined together with neighboring clusters (using average distance between cluster members) while combining clusters.

The figure below shows the hierarchical clusters using a dendrogram. Please see [this](#) for an introduction to dendrogram interpretation.

The horizontal axis refers to a distance measure between clusters.

Click on the image below to get a high resolution version.



Memory Bandwidth



Memory Bandwidth Data

The following table shows the memory bandwidth information for the following applications. It is dynamically generated from live performance data in PADS.

This information is obtained using the memory bandwidth tool that is based on [PAPI](#) hardware counters. The statistics represent the mean value for each metric across the different tasks.

Click on a column name to sort.

App	Num Tasks	Exp	BW_Read_CPU(B/Cycle)	BW_Read_Wall(MB/s)	BW_Write_CPU(B/Cycle)	BW_Write_Wall(MB/s)	CpuCycles	MemReads	MemWrites	WallClock
BoxLibMiniAMR	8	inputs_3d_timesteponly	0.56	928.58	0.00	0.00	3889044173.78	33839580.44	0.00	233213
BoxLibMiniAMR	8	inputs_2d_timesteponly	0.17	169.52	0.00	0.00	437123724.56	1175823.78	0.00	44388
EAVL	1	iso40_noise0128	0.10	283.92	0.04	109.62	2415024670.00	3913667.00	1511002.00	88220
Exp_CNS_NoSpec	2	inputs_3d	0.78	1465.03	0.16	291.68	8700806824.67	106291529.00	21162151.33	464335
HACCmk	1	1_thread	0.02	68.69	0.00	4.15	68012485.50	37121.04	1300.48	2138
HACCmk	1	2_threads	0.01	28.53	0.00	6.21	42084770.15	7514.72	1067.29	1344
HACCmk	1	4_threads	0.01	38.29	0.00	11.70	25729291.90	5011.18	1090.02	833
HACCmk	1	8_threads	0.02	62.41	0.01	20.75	13438124.45	4706.45	1027.43	445
HPCG	192	default	1.24	3747.41	0.02	69.03	1810479834357.26	34990358553.28	644509616.94	59758108
HPCG	96	default	1.73	5237.05	0.03	79.98	1754664151237.09	47422673196.04	724269887.07	57953424
HPCG	16	default	1.69	5199.69	0.02	59.44	1788992249148.35	47356482590.47	541396967.12	58288400
HPCG	64	default	2.03	6213.75	0.03	86.14	1781695406851.03	56438609360.72	782441470.57	58130300
HPCG	32	default	2.47	7544.22	0.03	92.28	1782049497043.73	68811648462.12	841677890.91	58375100
miniAMR	64	expanding-sphere-64p	0.39	1153.40	0.20	591.85	57800231359.83	342289340.17	175763887.23	1978375
miniFE	12	528x256x256_solveonly	1.00	2750.49	0.05	139.99	517169685641.77	8076669915.31	411071139.38	18793246
MultiGrid_C	48	results.3d_n_cell_256	0.96	1524.62	0.18	289.55	3993980757.83	59642611.54	11314979.40	250234
MultiGrid_C	48	results.3d_n_cell_128	0.91	1163.94	0.19	233.31	480096312.52	6784229.81	1365572.20	37666
MultiGrid_C	48	results.3d_n_cell_512	0.96	2272.26	0.16	383.88	35150257764.76	521923405.73	87602453.30	1491313
Nek5000	32	default	1.02	3119.57	0.32	985.76	467544340678.97	7472873800.61	2361368907.52	15331066
Nek5000	64	default	1.05	3209.83	0.33	1003.20	235332132089.98	3866995867.82	1208590596.89	7710307
Nek5000	128	default	0.94	2861.03	0.29	884.81	81474889690.39	1218100135.30	378842600.18	2667178
Nek5000	256	default	0.89	2708.87	0.27	825.33	64764651440.12	897321356.04	273391993.60	2120022
Nek5000	512	default	0.70	2145.87	0.20	599.33	40376921711.09	443158499.75	123772050.60	1321717
RSBench	1	default	0.47	1303.64	0.00	3.67	71865764399.00	526762236.00	1482737.00	2586060
Visit	1	iso_noise256	0.15	411.83	0.12	307.97	6009073268.00	14512955.00	10852876.00	225536
vodeDriver	1	drm19_0700.tab	0.02	65.01	0.00	6.86	2337016343.00	829539.00	87515.00	81667

Experiments



PADS: Performance Analytics Data Store

[Home](#)

[Apps](#)

[Instruction Mix](#)

[Communication](#)

[Clustering](#)

[Memory Bandwidth](#)

[Exp List](#)

Experiments

Exp ID	App	AppVer	Tool	ToolVer	Exp	Exp Date	Upload User	Platform
5452638c53c4282d71c341c8	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-16agg-8stride-io_30d	2014-10-30 00:00:00	sarat	Keeneland
5452547853c428017b4f856b	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-32agg-4stride-io_30d	2014-10-30 00:00:00	sarat	Keeneland
5451668c53c42855bbe76a45	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-64agg-2stride-io	2014-10-27 00:00:00	sarat	Keeneland
545164b753c4283f9fa48f7	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-2agg-64stride-io	2014-10-27 00:00:00	sarat	Keeneland
5451643e53c42838e10259c7	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-4agg-32stride-io	2014-10-27 00:00:00	sarat	Keeneland
5451626b53c4282572d74a6b	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-4agg-16stride-io	2014-10-27 00:00:00	sarat	Keeneland
545160e353c428160b5d8071	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-4agg-16stride	2014-10-27 00:00:00	sarat	Keeneland
54515fac53c428093e641b73	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-8agg-16stride-io_30d	2014-10-27 00:00:00	sarat	Keeneland
54515d6453c4286bab05406e	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-2agg-16stride	2014-10-27 00:00:00	sarat	Keeneland
545159b353c4284587a5991d	MPAS	2.1	mpip	1.0	worldOcean_QU_120km-30d:contig-default_agg_0_stride	2014-10-27 00:00:00	sarat	Keeneland

[Previous page](#) [Next page](#)

Note: Click on experiment ID to view details (JSON).

Download

Experiment Details

```
Id: 542c4ce96c0d74bf8de61aa8
Application: Nek5000
Version: 1037

Tool: mpip
Tool Version: 1.0

Experiment: vortex-64p
Exp Date: 2014-10-01 00:00:00

Platform: Keeneland

Uploaded By: sarat
Upload Date: 2014-10-01 14:50:17.364000

Runinfo:
{"totalTasks": 64}
```

[Download Raw Data](#)

Summary

- Oxbow + PADS: Collaboration platform for domain scientists, applied mathematicians, computer scientists, and hardware architects interested in Co-Design.
- Infrastructure provides ability to track changes over different software versions and problem configurations and easily identify interesting correlations across applications
- Tools can show surprising similarities and differences between application behaviors
 - Instruction mix of UMTmk and HASH are very similar, despite one being a microkernel and the other being a data-centric application
 - HPL instruction mix does not reflect most applications, even microkernels behave differently. HPCG does better.
- Tools can show surprising behavior in a given application and help in validation
 - Communication patterns deviate from expected behavior
- Tools can predict what kinds of system attributes will be selected by testing a given system with a given application
 - HPL will perform best on systems with good support for SIMD instructions
 - HPCG will perform best on systems with support for localized communication and a mix of memory, move, floating point, and integer operations
- Multiple tools can show similarities and differences for different machine subsystems
 - Instruction mix may be very similar between applications, showing similar on-node characteristics
 - Communication patterns may be very different, showing different intra-node system requirements

Acknowledgements

- Contributors and Sponsors
 - Future Technologies Group: <http://ft.ornl.gov>
 - US National Science Foundation Keeneland Project: <http://keeneland.gatech.edu>
 - US Department of Energy Office of Science
 - DOE ExMatEx Codesign Center: <http://codesign.lanl.gov>
 - DOE Cesar Codesign Center: <http://cesar.mcs.anl.gov/>
 - DOE Exascale Efforts: <http://science.energy.gov/ascr/research/computer-science/>

Project website
<http://oxbow.ornl.gov>

Sarat Sreepathi
sarat@ornl.gov

Jeffrey Vetter
vetter@ornl.gov




Backup

Platform info


Keeneland

Web: <http://www.nics.tennessee.edu/core-projects/keeneland/>

Processor

Model Name	Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz			
CPU Family	6	Model	45	
Core Speed	2593.549 MHz	Cores	8 cores	
Address Sizes	46 bits physical, 48 bits virtual			

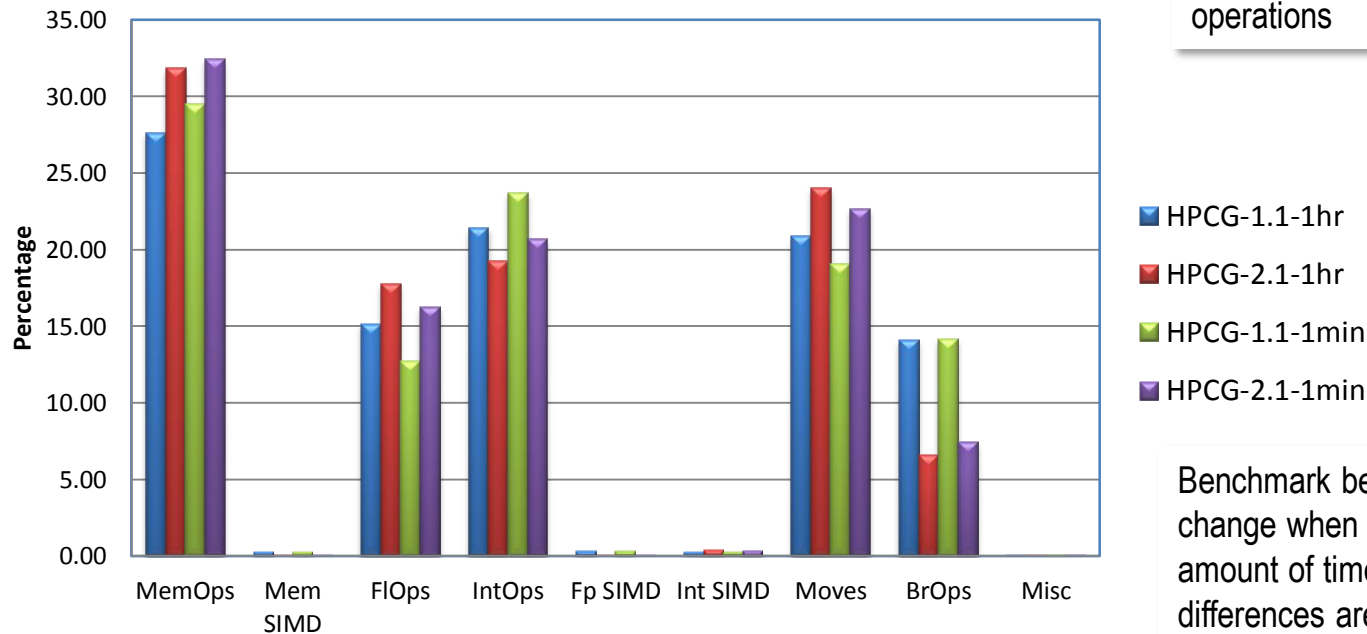
Caches

	Size	Number of Blocks	Descriptor	
L1 (data)	32 KBytes	64 blocks	8-way set associative, 64-byte line size	
L1 (instruction)	32 KBytes	64 blocks	8-way set associative, 64-byte line size	
Level 2	256 KBytes	512 blocks	8-way set associative, 64-byte line size	
Level 3	20480 KBytes	16384 blocks	20-way set associative, 64-byte line size	

Bus Information (Ishw)

Tracking changes in behavior over different versions and durations

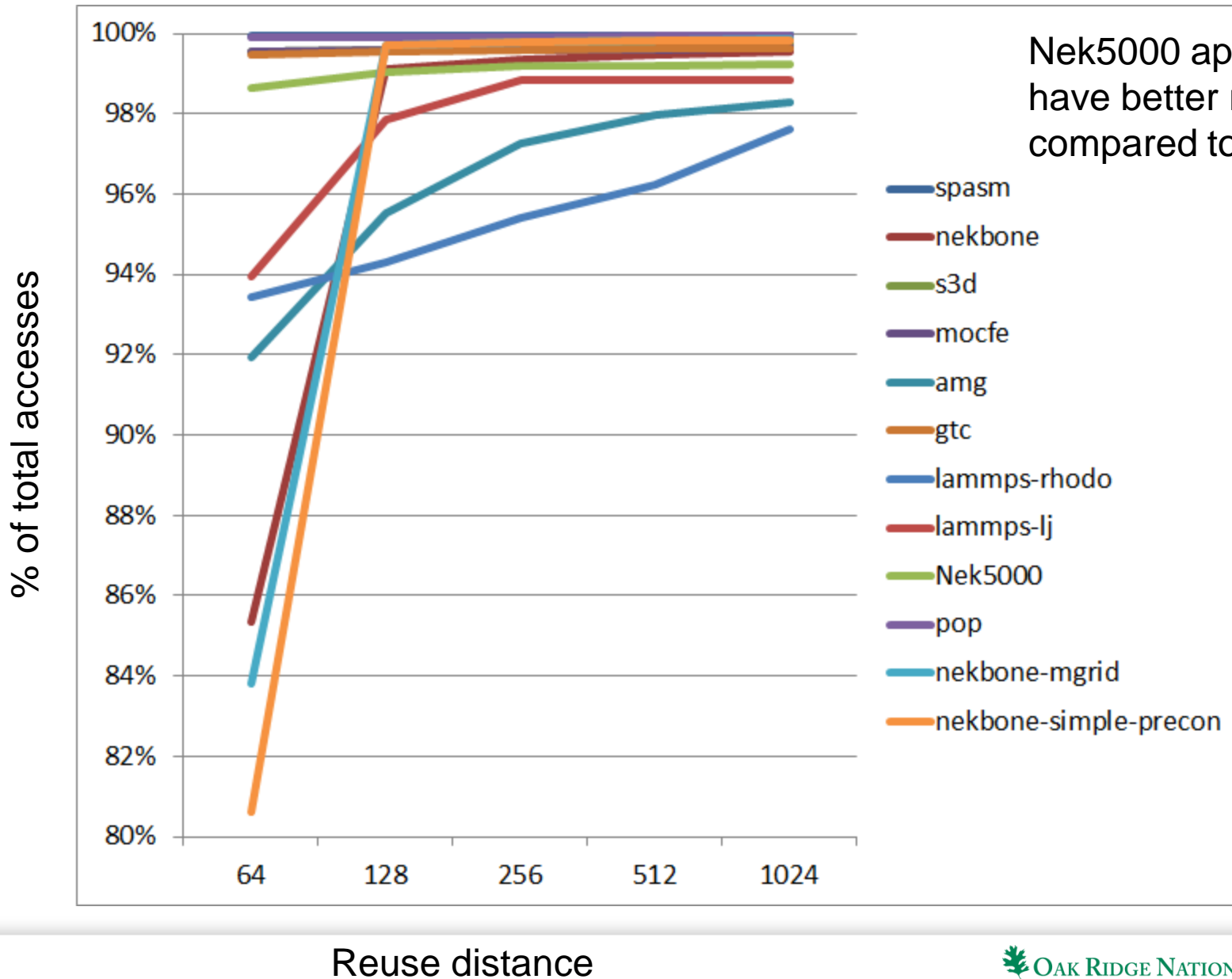
Instruction Mix Comparison
HPCG 1.1 vs. 2.1 - Duration 1hr & 1min



As applications evolve, the characteristics may change. Here, HPCG evolution from the initial alpha release to the current release shows optimizations to reduce branching instructions and integer operations

Benchmark behavior is expected to change when running for a sufficient amount of time. The instruction mix differences are not dramatic between 1 minute and 1 hour. A slight increase in the percentage of floating point operations reflects more time spent in the main CG solver loop.

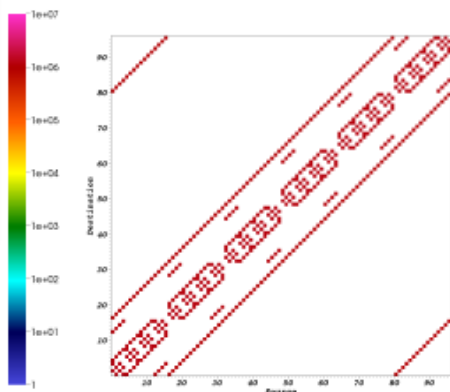
Memory Reuse Results



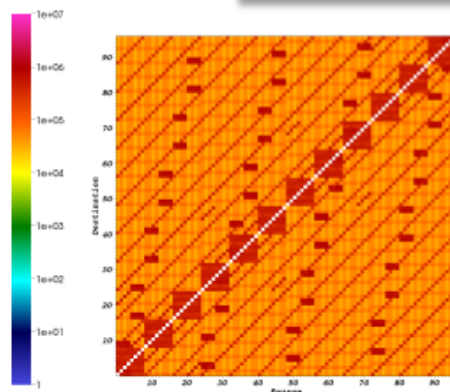
Communication Volume

LAAMPS: The same application may exhibit very different communication patterns for different problem configurations

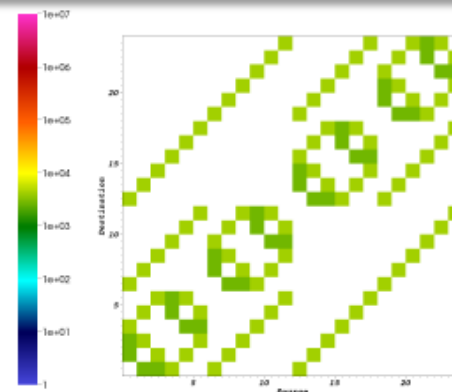
RandomAccess: Pattern is not the expected monochromatic random scatter due to in-source communication optimization



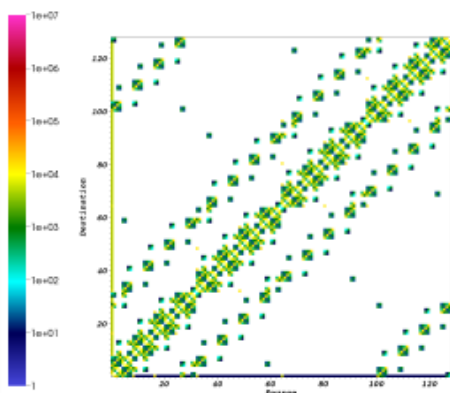
(a) LAMMPS EAM benchmark, 96 tasks.



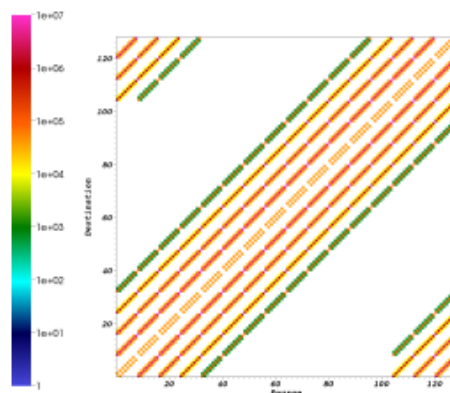
(b) LAMMPS Rhodo benchmark, 96 tasks.



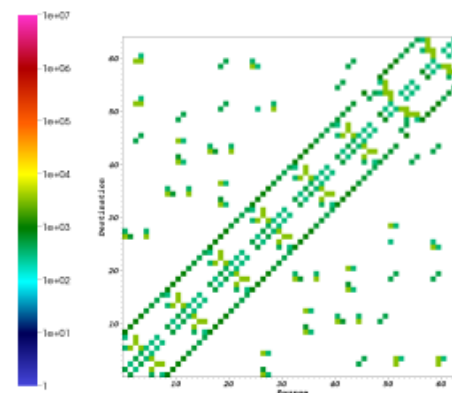
(c) HPC MPI RandomAccess, 24 tasks.



(d) Nek5000 application, 128 tasks.



(e) Nekbone proxy app, 128 tasks.



(f) POP application, 64 tasks.

Nekbone (old version - 2012) vs Nek5000: proxy application differs from modelled application in communication behavior.

Average volume of point to point communication. Color scale is consistent across all plots

Observations

- Tools can show surprising similarities and differences between application behaviors
 - HPL instruction mix does not reflect most applications, even microkernels behave differently
 - HPCG instruction mix more closely resembles other applications than does HPL, even applications implementing very different algorithms
- Tools can show surprising behavior in a given application
 - Many application perform well below system peak memory capacity
- Tools can predict what kinds of system attributes will be selected by testing a given system with a given application
 - HPL will perform best on systems with good support for SIMD instructions
 - HPCG will perform best on systems with support for localized communication and a mix of memory, move, floating point, and integer operations
 - POP and Nek5000 will benefit from systems with support for more randomized, scattered communication
- Multiple tools can show similarities and differences for different machine subsystems
 - Instruction mix may be very similar between applications, showing similar on-node characteristics
 - Communication patterns may be very different, showing different intra-node system requirements
 - E.g. HPCG instruction mix matches other applications, but communication volume resembles other benchmarks more than it resembles full applications.
- Tools can track changes over different software versions and problem configurations
 - Version optimization of HPCG has reduced branching behavior, without other significant changes.
 - LAMMPS communication patterns are very different for different benchmark problems