Integrated Performance Monitoring: Understanding Applications and Workloads

David Skinner

Snowbird, July 2008

Outline

- Overview of IPM
- The needs of production HPC centers wrt. tools
- IPM design and implementation
- Recent R&D work

Feel free to ask questions!

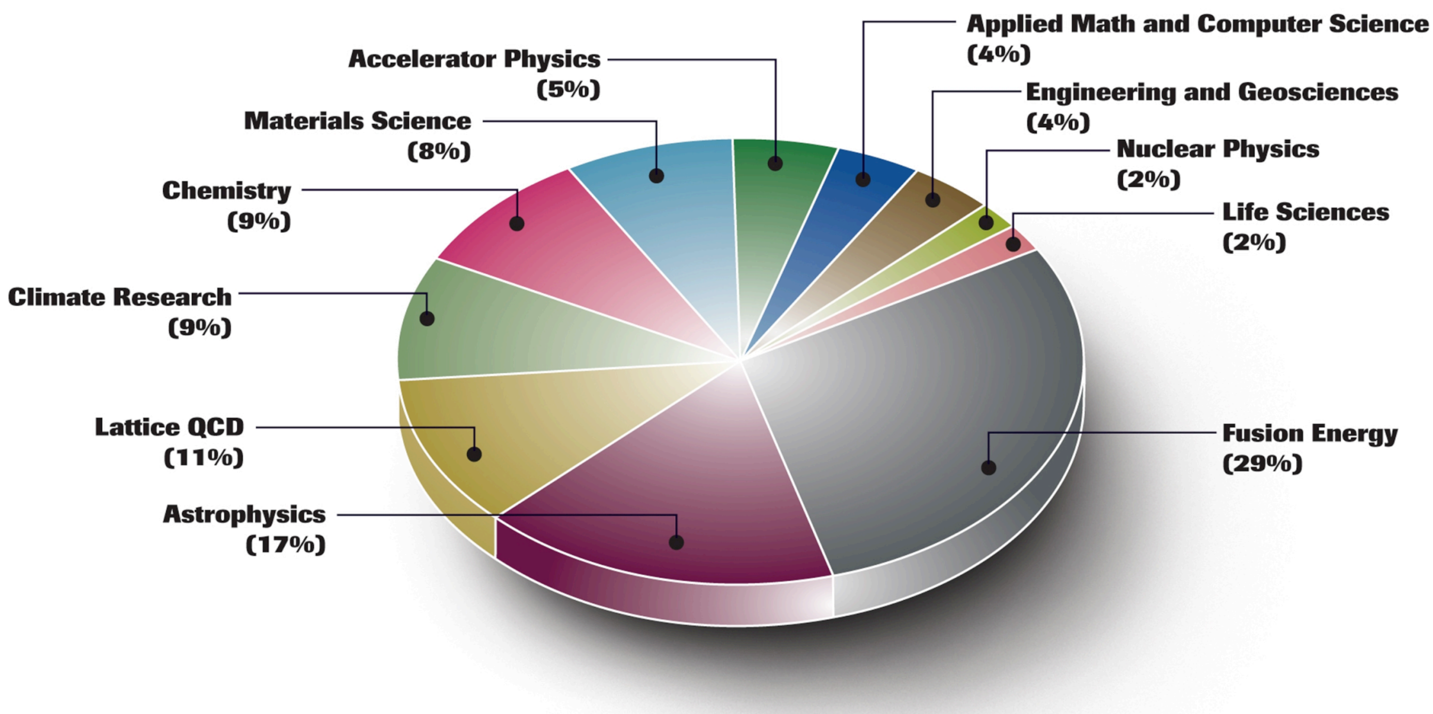
IPM Overview

- Integrating Performance Monitoring (IPM) is an easy to use, scalable HPC application profiling which serves both users and center managers. Its implemention is portable open source.

- IPM is a profiling layer more than a tool. There is no GUI, no API to learn, etc.

- IPM is funded by DOE and NSF. It's also used by at DOD MSRCs.

Realities at a production HPC center



- Large scale parallelism and data needs of science teams
- Large number of projects, users, and codes
- (10^5 tasks)(10^4 users)*(10^2 codes) performance threads
- Service oriented systems, ease of use in tools and all things
- Centerwide performance assessment for allocations

Luckily many HPC problems are simple, boring, unresearch-worthy in terms of computer science.

ERCAP Question 19.1

Each application for time at NERSC includes both algorithmic and performance assessments

19.1 Code and Application Descriptions

Code Name	Description	Mathematics	Numerical Techniques	Machines	Planned Processors	Num Procs Reason
GCP	A library to reconstruct dense detector-specific HEALpixel pointing from sparse and/or general focal plane Euler angle or quaternion pointing through interpolation and/or rotation and HEALpixelization.	Pointwise interpolation and rotation.	Polynomial interpolation and rotation matrix multiplication	Jacquard -5% Bassi - 5% Franklin - 5%	1 - 10, 000	Computational Requirements
M3	A CMB data management library, abstracting I/O for complex CMB datasets.	N/A	N/A	Jacquard - 5% Bassi - 5% Franklin - 5%	1 - 10000	Computational Requirements
MADAM	Make maps of the CMB temperature and polarization by destriping of ring-set time-ordered data.	Two phase solution, individually destriping rings and collectively solving for offsets.	Fourier transforms and dense linear algebra	Jacquard - 5% Bassi - 5% Franklin - 5%		Computational Requirements, Memory Required
	Make maximum-likelihood					

ERCAP Question 19.2

19.2 Code and Application Performance
Provide code performance data for typical processor counts used in production this past year. For machines with more than one processor per node enter # of procs as the number of nodes used times the number of processors per node.

You can use IPM to collect Gflops and Total Memory. Total Memory is the aggregate high water memory used on that number of processors.

Enter only numbers in the # of Procs, Gflops, and Total Memory columns. If you need more rows, click Save Code Description and 2 more rows will be added to the table.

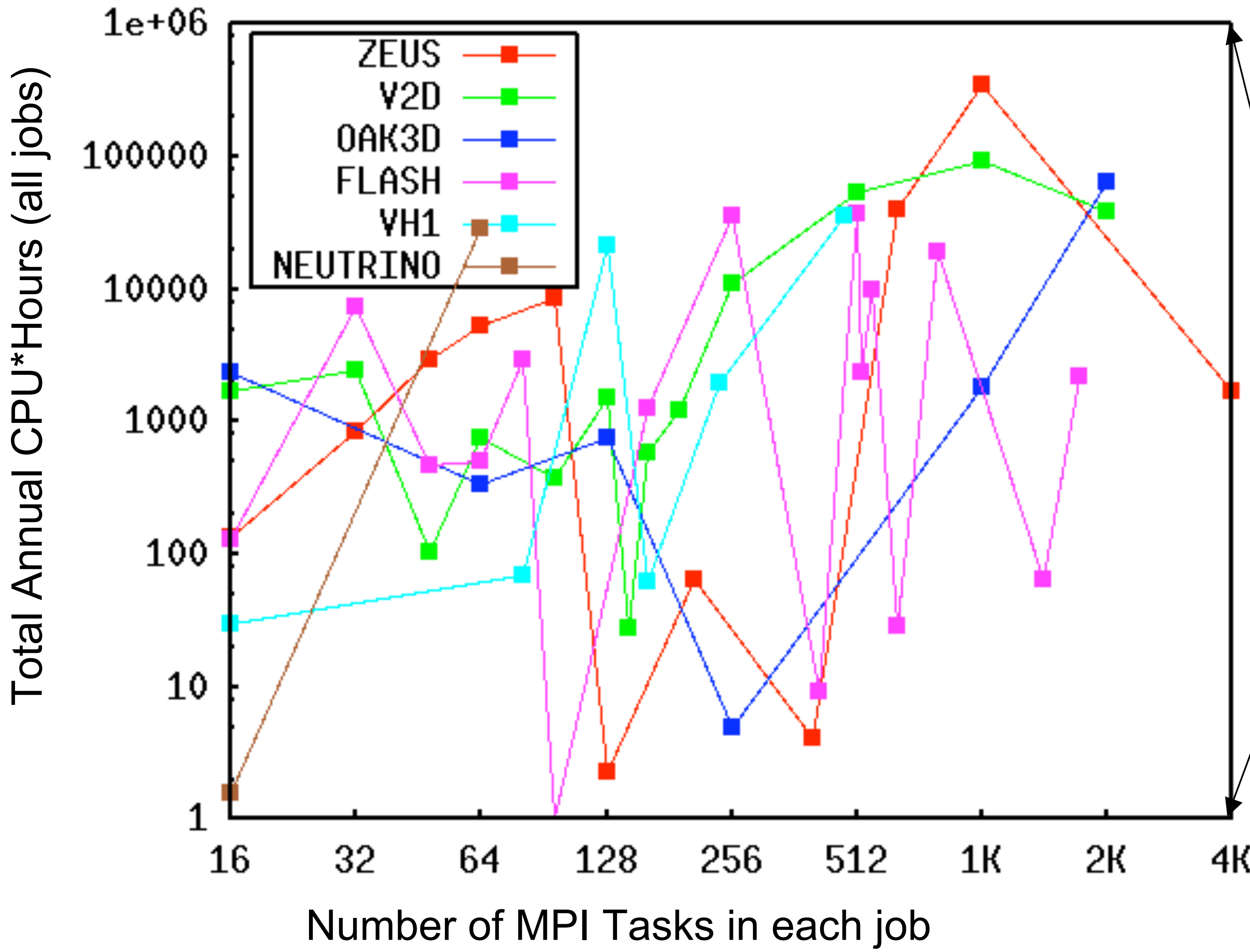
Machine	# of Procs	GFlop/sec	Aggregate Memory (GBytes)	How info was collected/comments
Jacquard	512	380	400	IPM
Jaguar	10,368	7,900	10,000	IPM Results thanks to L. Oliker

<u>Two core needs of NERSC, SDSC, TACC, etc.</u>

•How are ~400 projects going to generate this information without distraction from their research goals?

•When there is performance problem or need to tune, what's the first step?

Motivation: NERSC has many Customers and an Extremely Diverse Workload



One of ~ 300 Research Projects

A workload, qualitatively described

Science areas	Dense linear algebra	Sparse linear algebra	Spectral Methods (FFT)s	Particle Methods	Structured Grids	Unstructured or AMR Grids
Accelerator Science		X	X IMPACT-T	X IMPACT-T	X IMPACT-T	X
Astrophysics	X	X MAESTRO	X	X	X MAESTRO	X MAESTRO
Chemistry	X GAMESS	X	X	X		
Climate			X CAM		X CAM	X
Combustion					X MAESTRO	X AMR Elliptic
Fusion	X	X		X GTC	X GTC	X
Lattice Gauge		X MILC	X MILC	X MILC	X MILC	
Material Science	X PARATEC		X PARATEC	X	X PARATEC	

How do we study such a workload quantitatively?

How can we spot application performance issues?

Can we just use the vendor performance tools?

First, what is a performance tool?

1. An application that users can run to debug the performance of their code (is this what the center wants?)
2. A runtime layer implemented by the center staff that reports on application performance (is this what the user wants?)

Can we have both at the same time?

1. Must allow users flexibility in how they debug performance
2. The carrot works. The stick does not.

Performance Analysis using a Tool

HPC apps expert

Running a parallel app
w/ performance tool

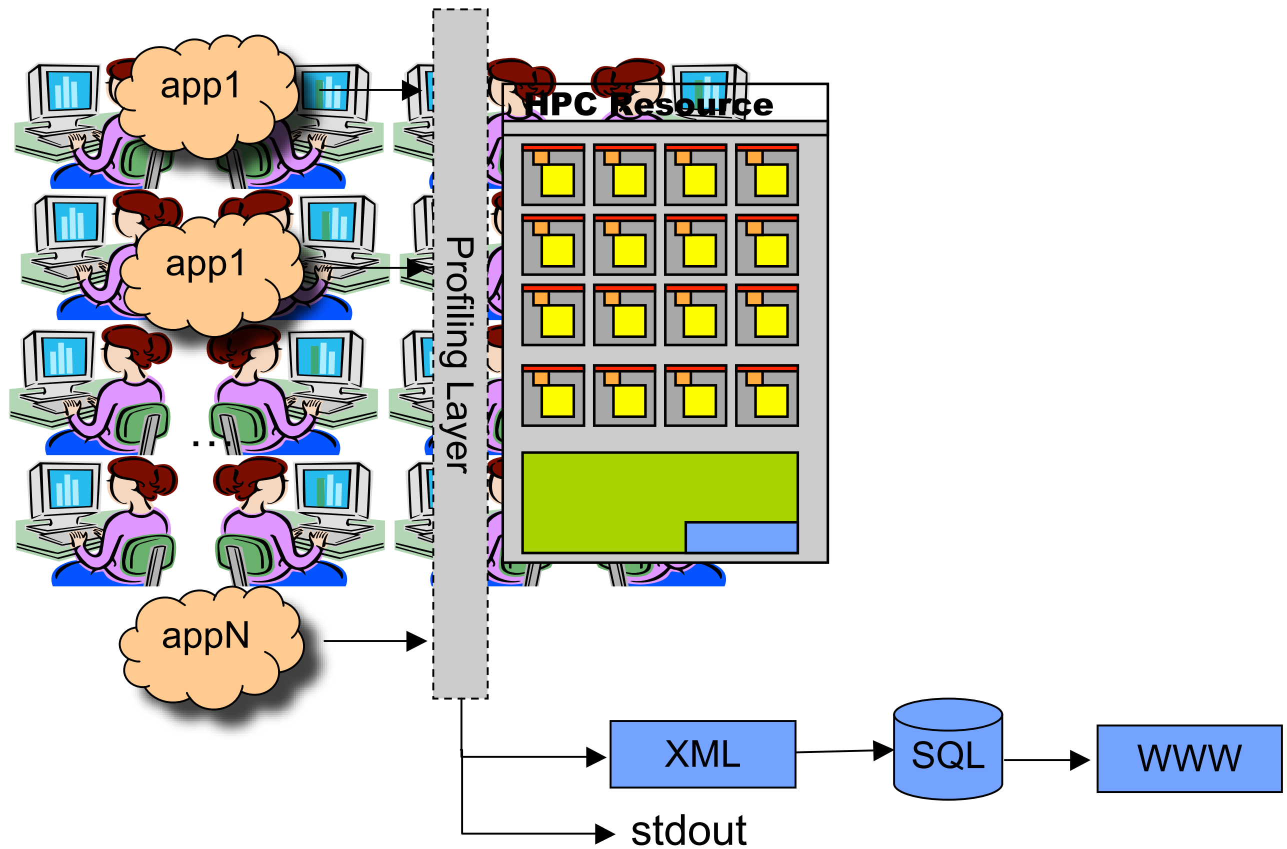
1. Get to know the algorithm and source.
2. Instrument, Run, Analyze, Summarize
3. Iterate on #2

Workload Analysis

Whole workload? No problem, this process is embarrassingly parallel.

1) What is the parallel scaling (in person hours)?
2) Are the analyses comparable?

NERSC has ~300 Projects



Layer:
1) Transparent, does not interrupt production computing
2) Simple & Systematic, comparable profiles
3) Focus on data flow, not GUI presentation

Tools:
1) Good for getting to know a code
2) Debugging or improving a code
3) Not for workload snapshot or inventory

Don't collect perf data through a GUI.
Well defined performance records.

IPM Motivation

Whether we call it a tool or a profiling layer, we want to :

- Make it easy for both users and the center to generate comparable workload performance analyses.
- Make it easy to identify the causes performance losses.
- Make it easy to state clearly which HPC resources are most critical to the center's workload.
- Make it easy to access performance profiles.

Profiling Tools

- **Many tools exist, roughly they vary by**

Type of Information
Level of Detail
Runtime Impact on Code
Scalability
Ease of Use

What tool should I use?
There is no "right" tool.

Which tool helps you
answer Question 19?

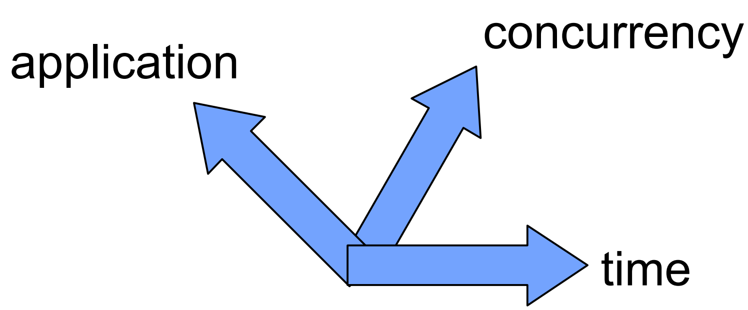
- **HPC centers with complex & dynamic workloads need an easy to use, almost transparent, low impact profiling layer that provides high level summaries about job performance.**
- **More in-depth & detailed tools can be used subsequently. Use the right tool for the job.**

Profiling Tools (contd)

- Many performance analysis tools are not scalable. The volume of data or number of files may preclude their use. They may write a file per task.
- Does the tool profile the libraries you're using or just your own code?
- A code many run differently (or not at all) when profiled by some tools.
- Getting a lot of people to use the same tool in the same way is hard, little comparable performance data between projects or machines.
- Your tool may give you an information headache

Profiling is Projection

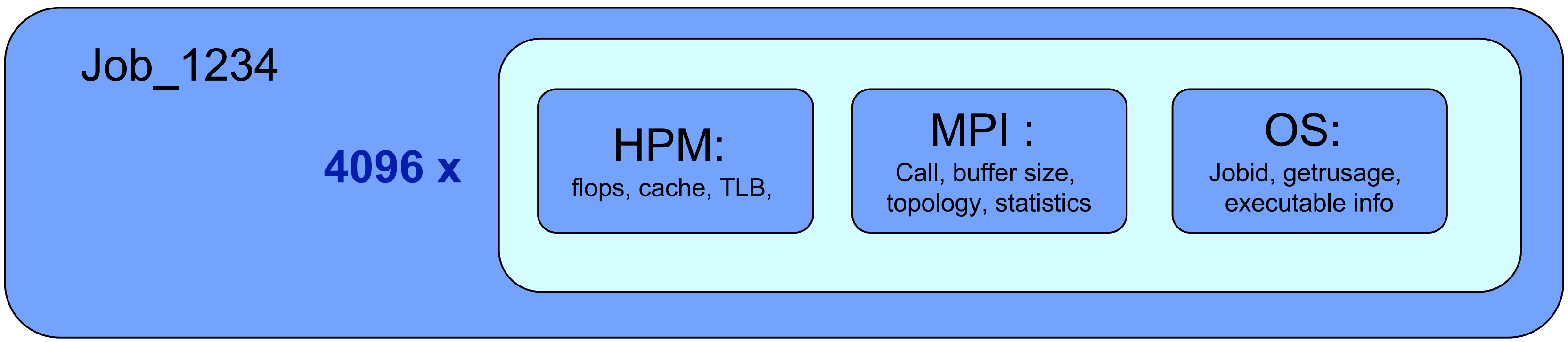
- At a high level performance events occur in a three dimensional space



- Where is the performance event?
  - In time
  - In the computer (rank space)
  - In the code (source line)
- Profiling requires projections on this space, flattening some or all of its dimensions

What do we want from a profile?

- Informative summary of an application
  - A batch job is the outermost context



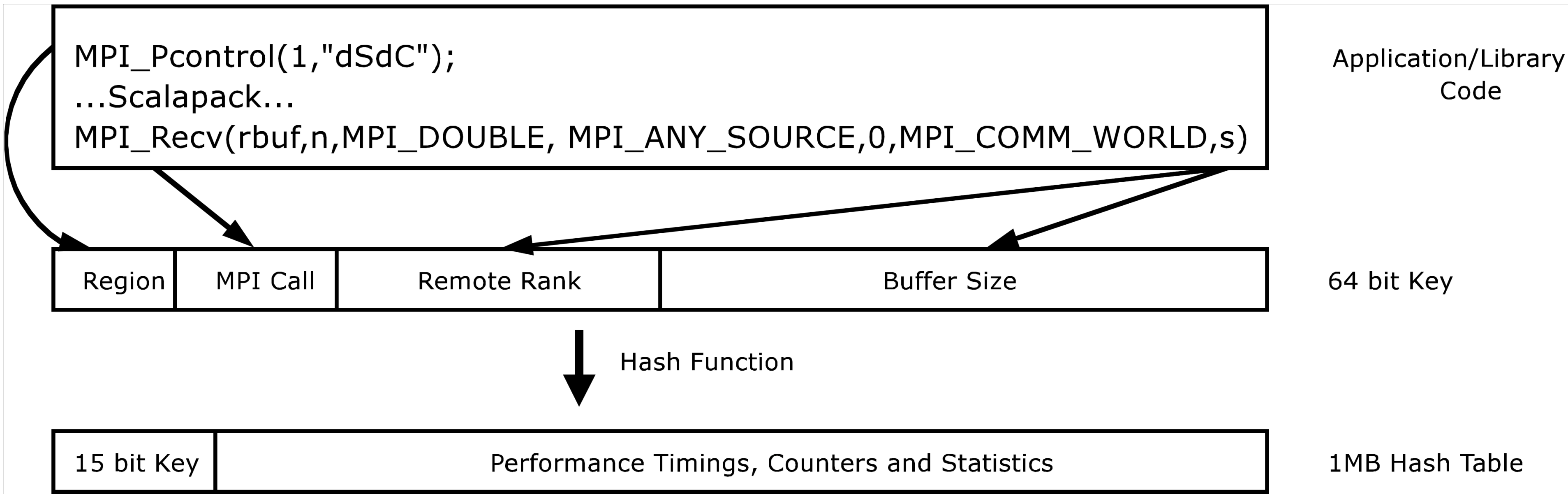
- Profiles should be comparable across applications, architectures, and concurrencies
- There is such a thing as too much information
  - Tracing should be used when needed, but it’s often not the first tool to reach for when performance is low.

IPM: Design Goals

- **Provide high level performance profile**
- **Fixed memory footprint** 1-2 MB
- **Minimal CPU overhead** 1-2%
- **Parallel aware** Use MPI, switch, and other resources at hand
- **Easy to use** Flip of a switch, no recompilation, no instrumentation
- **Portable**

IPM: Information Flow

An example w/ user controlled context tagging



How to use IPM : basics

1) Do “module load ipm”, then run normally

2) Upon completion you get

```
##IPMv0.85#############################################################
#
# command : ../exe/pmemd -O -c inpcrd -o res (completed)
# host    : s05405                          mpi_tasks : 64 on 4 nodes
# start   : 02/22/05/10:03:55               wallclock : 24.278400 sec
# stop    : 02/22/05/10:04:17               %comm     : 32.43
# gbytes  : 2.57604e+00 total               gflop/sec : 2.04615e+00 total
#
########################################################################
```

Maybe that’s enough. If so you’re done.

Have a nice day.

Q: How did you do that? A: MP_EUILIBPATH, LD_PRELOAD, XCOFF/ELF

Want more detail? IPM_REPORT=full

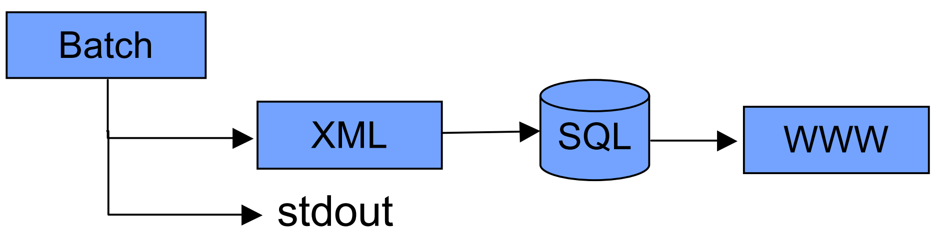
```
##IPMv0.85#####################################################################
#
# command : ../exe/pmemd -O -c inpcrd -o res (completed)
# host    : s05405                        mpi_tasks : 64 on 4 nodes
# start   : 02/22/05/10:03:55             wallclock : 24.278400 sec
# stop    : 02/22/05/10:04:17             %comm     : 32.43
# gbytes  : 2.57604e+00 total             gflop/sec : 2.04615e+00 total
#
#                         [total]       <avg>          min          max
# wallclock               1373.67      21.4636      21.1087      24.2784
# user                     936.95      14.6398        12.68         20.3
# system                    227.7      3.55781         1.51            5
# mpi                     503.853       7.8727       4.2293      9.13725
# %comm                                32.4268        17.42       41.407
# gflop/sec               2.04614    0.0319709      0.02724      0.04041
# gbytes                  2.57604    0.0402507    0.0399284    0.0408173
# gbytes_tx              0.665125    0.0103926  1.09673e-05    0.0368981
# gbyte_rx               0.659763    0.0103088  9.83477e-07    0.0417372
#
```

Want more detail? IPM_REPORT=full

```
# PM_CYC                  3.00519e+11   4.69561e+09   4.50223e+09   5.83342e+09
# PM_FPU0_CMPL            2.45263e+10   3.83223e+08    3.3396e+08   5.12702e+08
# PM_FPU1_CMPL            1.48426e+10   2.31916e+08   1.90704e+08    2.8053e+08
# PM_FPU_FMA              1.03083e+10   1.61067e+08   1.36815e+08   1.96841e+08
# PM_INST_CMPL            3.33597e+11   5.21245e+09   4.33725e+09   6.44214e+09
# PM_LD_CMPL              1.03239e+11   1.61311e+09   1.29033e+09   1.84128e+09
# PM_ST_CMPL              7.19365e+10   1.12401e+09   8.77684e+08   1.29017e+09
# PM_TLB_MISS             1.67892e+08   2.62332e+06   1.16104e+06   2.36664e+07
#
#                             [time]       [calls]        <%mpi>       <%wall>
# MPI_Bcast                  352.365          2816         69.93         22.68
# MPI_Waitany                81.0002        185729         16.08          5.21
# MPI_Allreduce              38.6718          5184          7.68          2.49
# MPI_Allgatherv             14.7468           448          2.93          0.95
# MPI_Isend                  12.9071        185729          2.56          0.83
# MPI_Gatherv                2.06443           128          0.41          0.13
# MPI_Irecv                    1.349        185729          0.27          0.09
# MPI_Waitall               0.606749          8064          0.12          0.04
# MPI_Gather               0.0942596           192          0.02          0.01
###############################################################################
```

Need a more detailed application profile?

You'll need a web browser.



IPM: XML log files

- There's a lot more information in the logfile than you get to stdout. A logfile is written that has the hash table, switch traffic, memory usage, executable information, ...

- Parallelism in writing of the log (when possible)

- The IPM logs are durable performance profiles serving

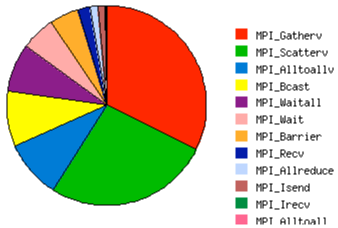
  – HPC center production needs:
  https://www.nersc.gov/nusers/status/llsum/
  http://www.sdsc.edu/user_services/top/ipm/

  – HPC research: ipm_parse renders txt and html
  http://www.nersc.gov/projects/ipm/ex3/

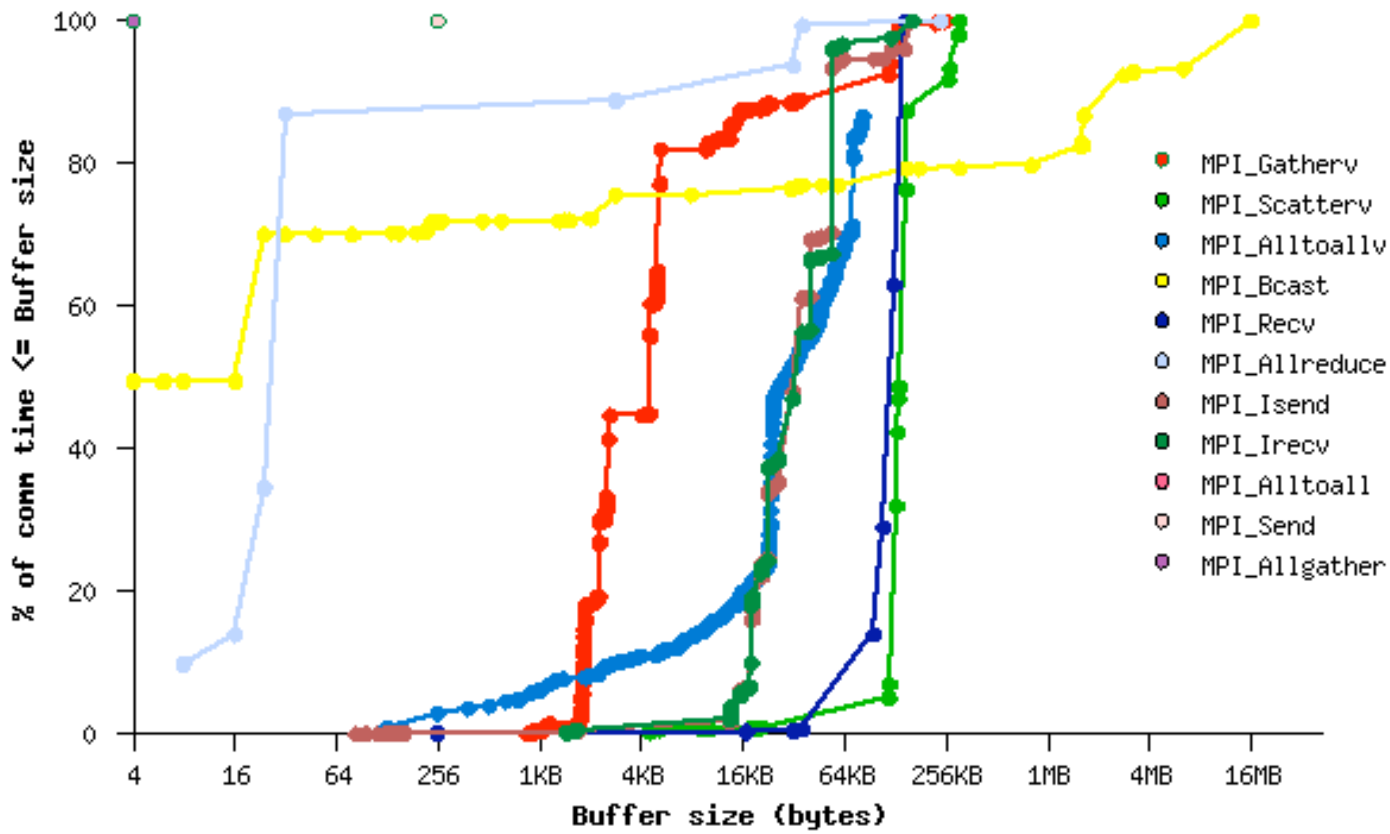
  – your own XML consuming entity, feed, or process

Message Sizes : CAM 336 way

per MPI call

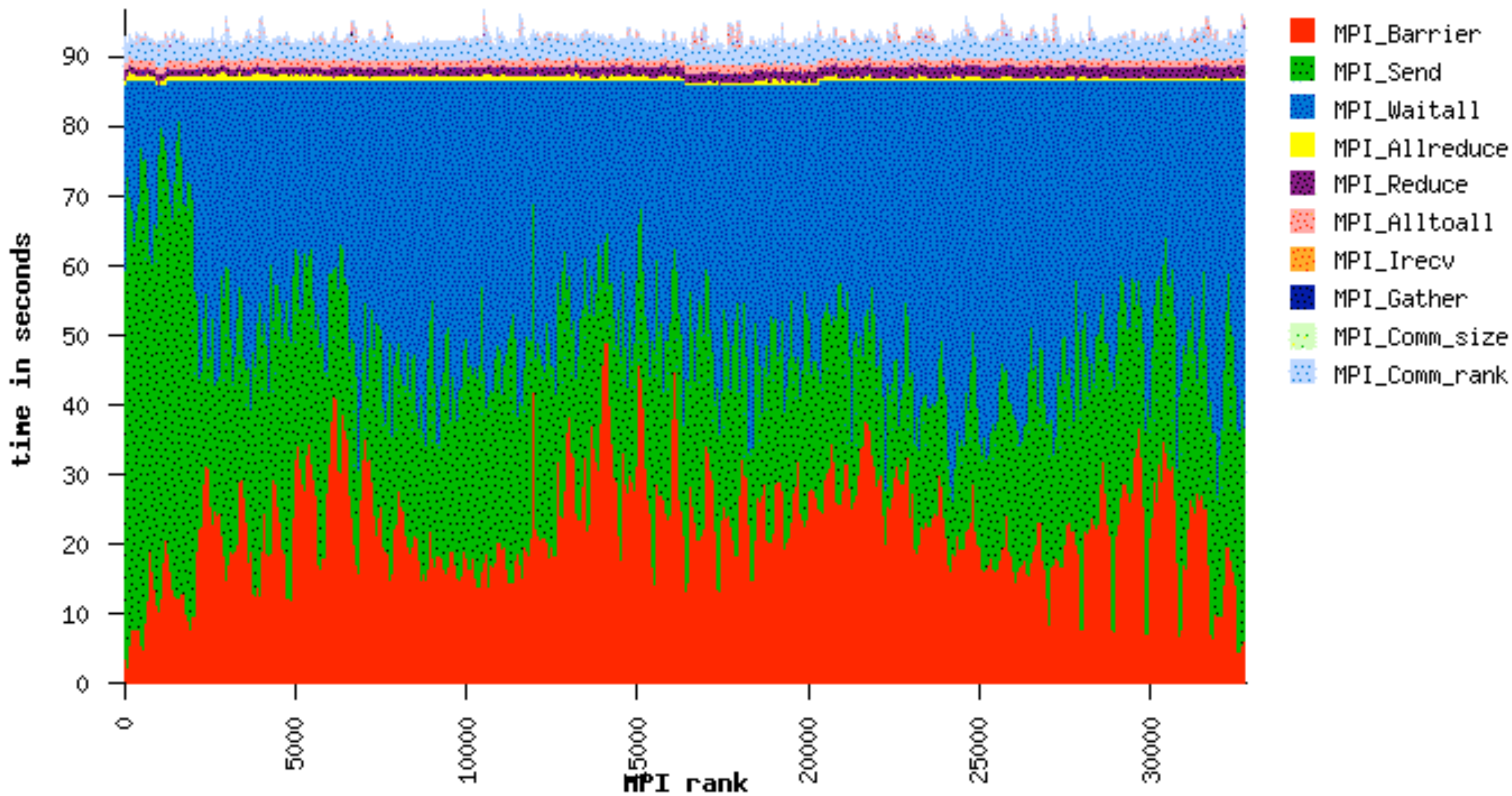


per MPI call & buffer size



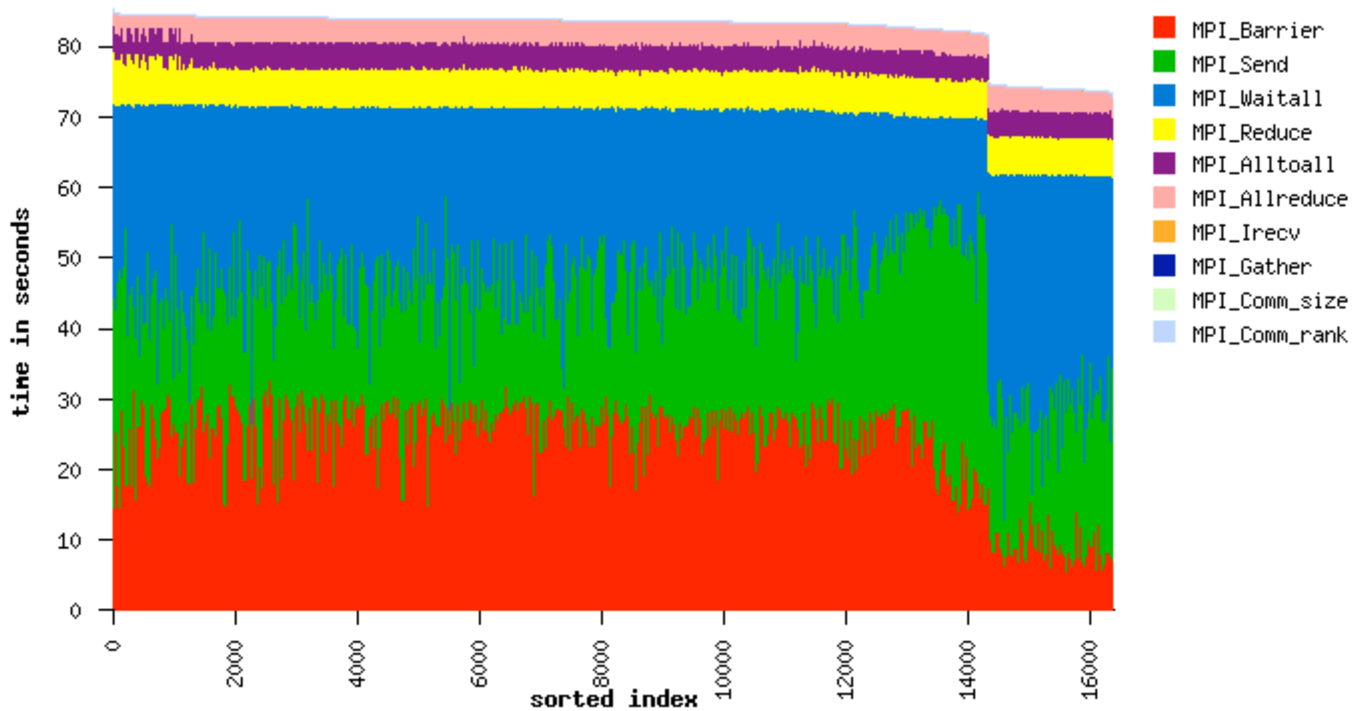
Scalability: Required

32K tasks AMR code



What does this mean?

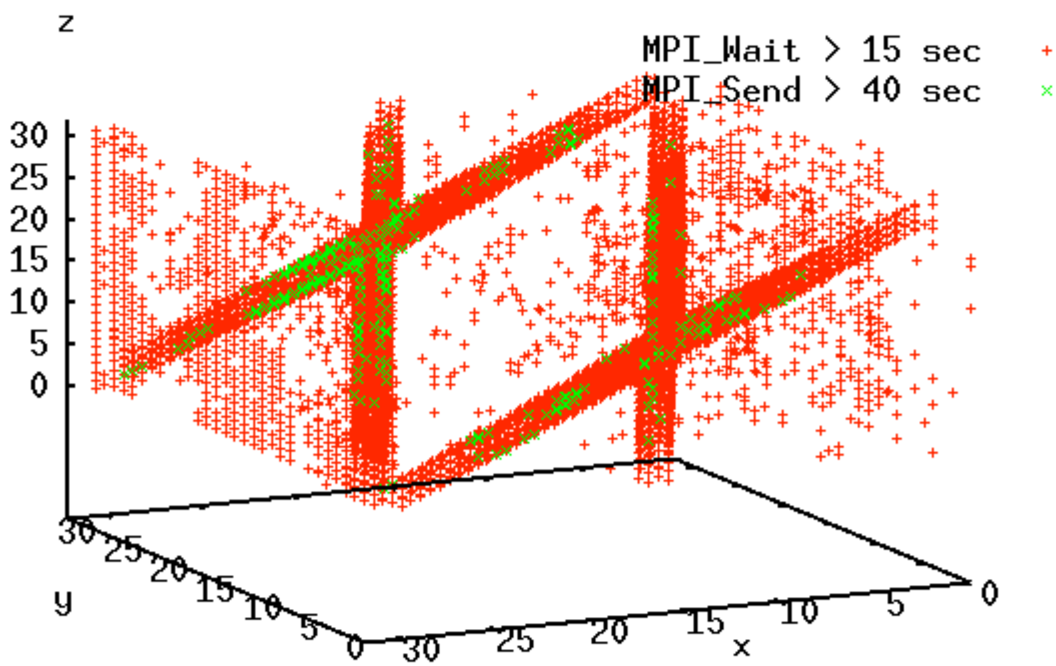
More than a pretty picture



Discontinuities in performance are often key to 1st order improvements

But still, what does this really mean? How the !@#!& do I fix it?

Scalability: Insight

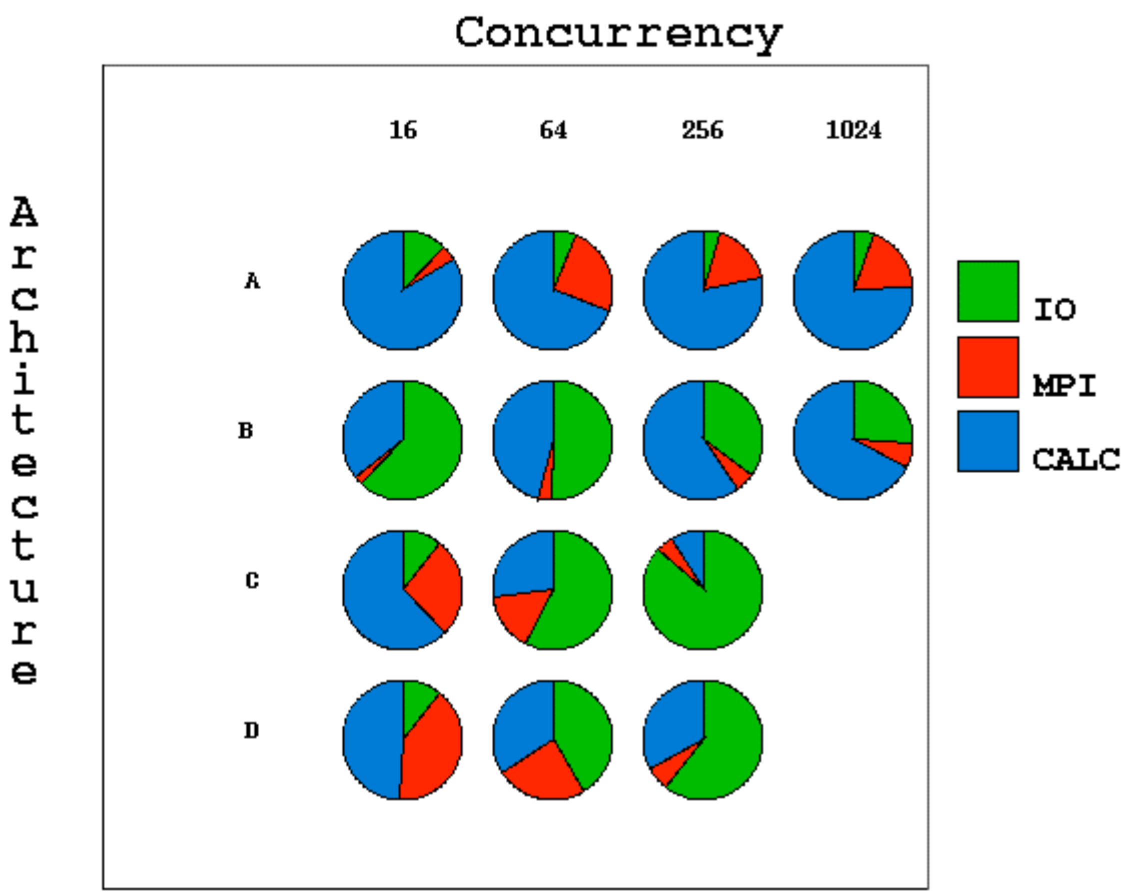


- Domain decomp
- Task placement
- Switch topology

Aha.

Portability: Profoundly Interesting

A high level description of the
performance of a well known cosmology code on four well known architectures.



IPM: Design Goals

- Provide high level performance profile ✓
- Fixed memory footprint ✓
- Minimal CPU overhead ✓
- Parallel aware ✓
- Easy to use ✓
- Portable ✓

Now at version 0.947

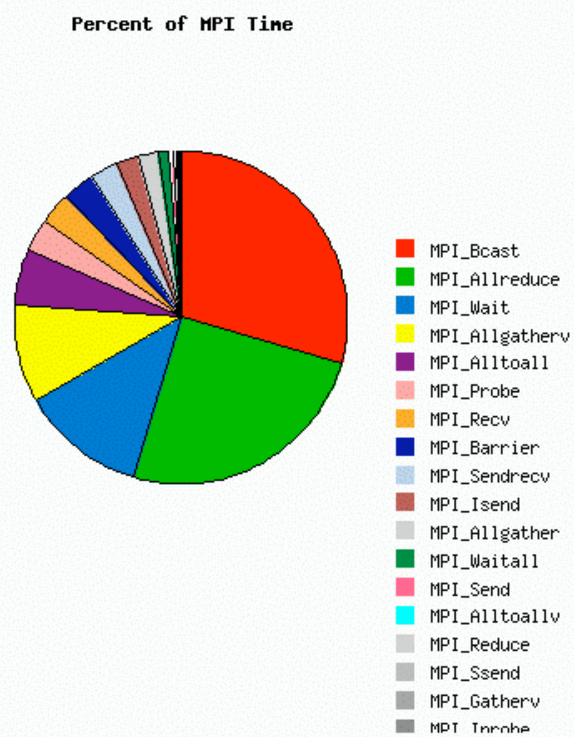
What about the workload?

Show jobs that ran after Nov 5 2003 @ 0 : 0
and ran on or before Aug 11 2004 @ 23 : 59
Current data ends at 11:30 p.m. yesterday.

Submit Query | Reset to Defaults

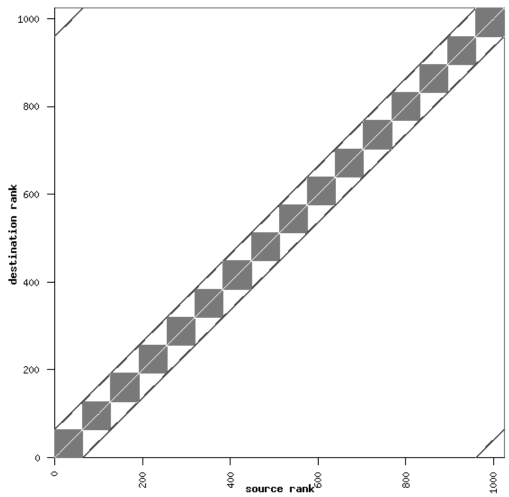
Summary for 1163 non-interactive jobs, avg. size: 34.73 tasks, avg. MPI pct: 31.1 %

Function	Total calls	Total time (sec)		Total buffer size (MB)	Avg. Buffer Size/call (Bytes)
MPI_Bcast	4.06e+11	2.85e+07	29.41%	1.10e+08	284
MPI_Allreduce	1.27e+10	2.44e+07	25.20%	3.24e+07	2,673
MPI_Wait	5.05e+10	1.17e+07	12.11%	2.99e+05	6
MPI_Allgatherv	1.61e+10	9.22e+06	9.51%	4.54e+06	296
MPI_Alltoall	3.68e+07	5.17e+06	5.33%	1.98e+04	564
MPI_Probe	6.17e+05	3.14e+06	3.24%	0	0
MPI_Recv	8.82e+09	3.08e+06	3.17%	2.61e+07	3,108
MPI_Barrier	1.79e+08	3.03e+06	3.13%	0	0
MPI_Sendrecv	5.29e+09	2.60e+06	2.68%	1.43e+07	2,839
MPI_Isend	3.45e+10	1.98e+06	2.05%	6.03e+08	18,343
MPI_Allgather	1.01e+10	1.89e+06	1.95%	1.07e+05	11
MPI_Waitall	2.32e+08	1.07e+06	1.11%	1.45e+05	657

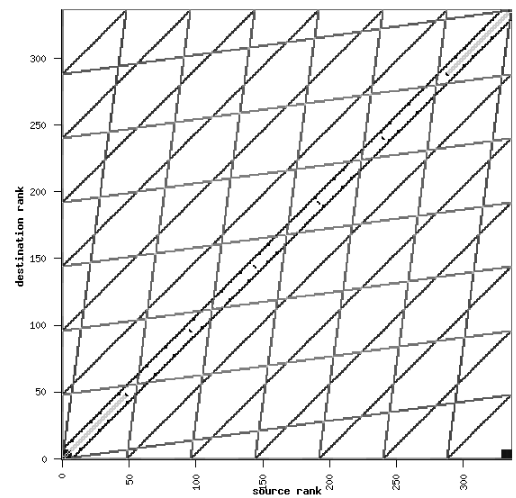


What sort of interconnect does your workload need?

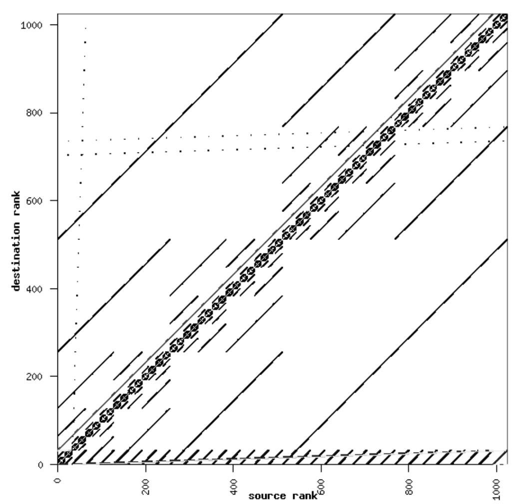
1024 way MILC



336 way CAM

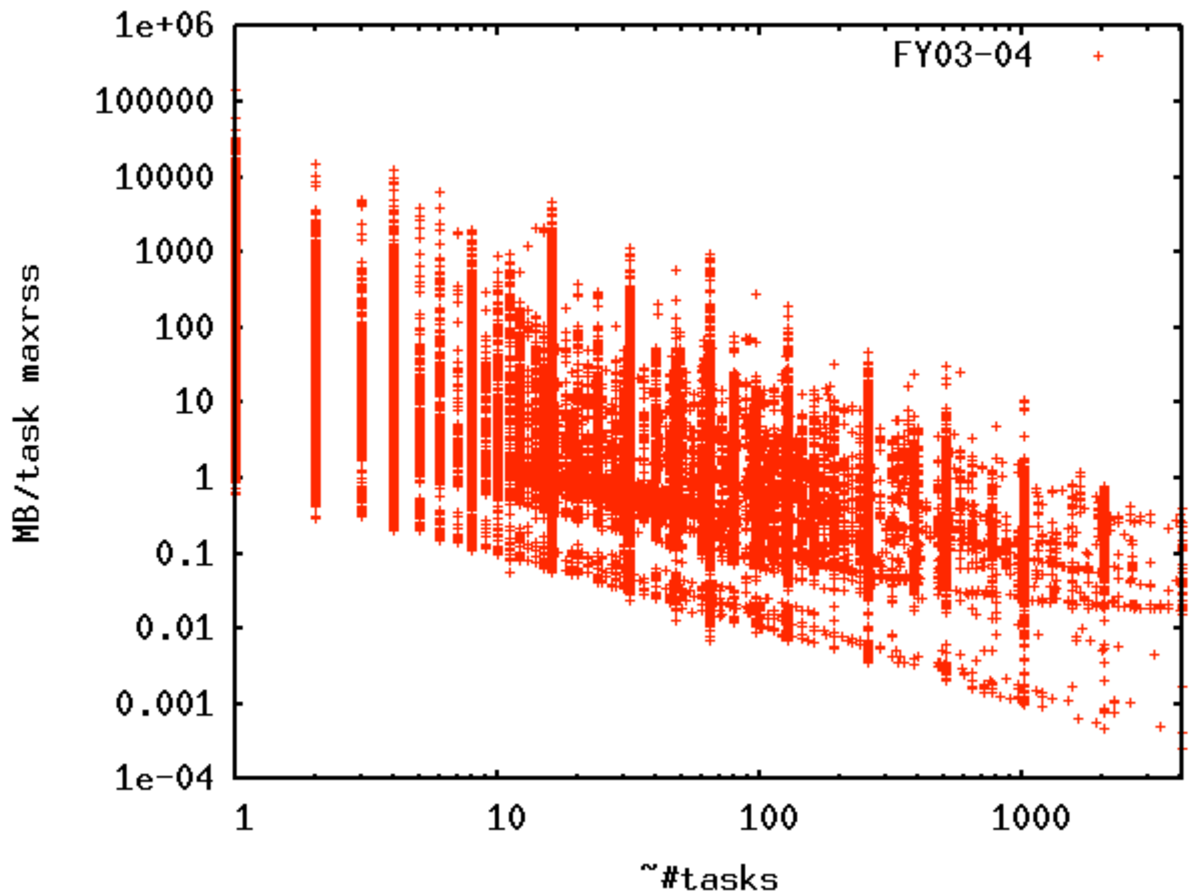


1024 way MADCAP



This is an active HPC research topic which goes on largely outside the space of center user concerns

Workload: How much memory does your workload need?



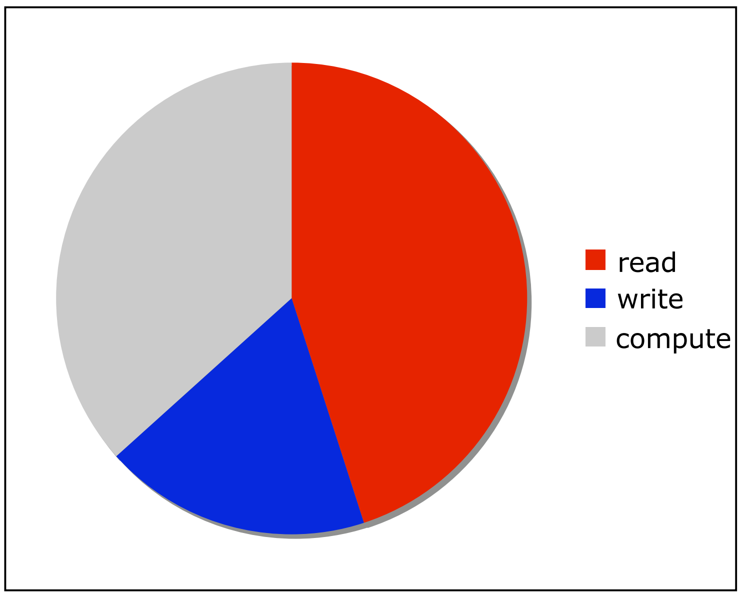
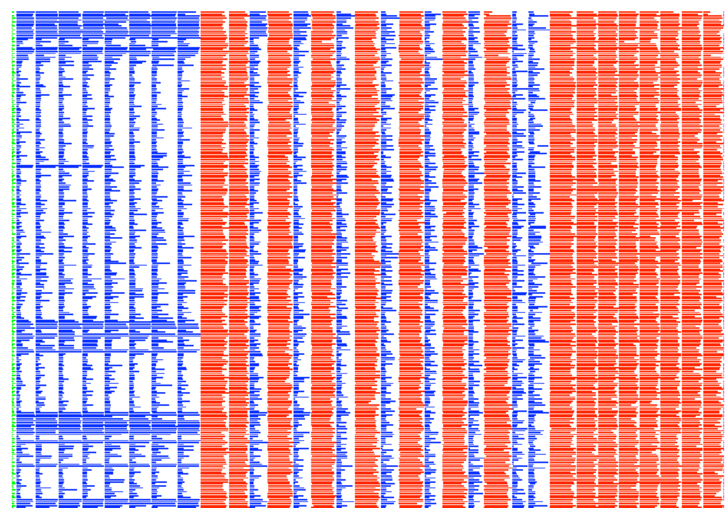
Weak vs. Strong Scaling

Recent work on I/O extensions

A POSIX I/O call looks a lot like an MPI call

Write(fd,buff, buffsize) ~ MPI_Send(buff, rank, size)

We extended IPM to include I/O profiling

Simple Profile : 63% time in I/O	Detailed Trace: An inventory of each I/O
	

Current Status & Futures

- IPM is in production use at NERSC & SDSC (bassi, datastar, franklin). In research use on BG, X1, ES, and other architectures.
- We interoperate with the PERI Perf Schema
- Now have thousands of cross-architecture application profiles. These have provided users with performance perspective and centers & vendors with architectural resource assessments and projected requirements.
- If there are things that you think should be in IPM but are not, let us know. If you want to help the development of IPM, that's even better,

http://ipm-hpc.sf.net

IPM: Upcoming Activities

Here is what we are working on.

- XT4 shared library support in CNL OS version 2.1 (release in Q2 2008)
- Finish LDRD funded I/O profiling work (Jan 09)
  - What should an I/O profile look like?
- Continuing NSF funded SDCI work
  - Deploying IPM on all major NSF machines (TACC's Ranger completed in Feb 08)
  - Exporting NERSC's web/database workload infrastructure to NSF
  - Extending IPM to PGAS languages (UPC)
- HPC research directions…

From Profiles to Models

Q: What is an application model?

A: A method of calculating wall time for an application given problem input, concurrency and a detailed computer system description. Preferably the method is in a functional form and free from heuristics.

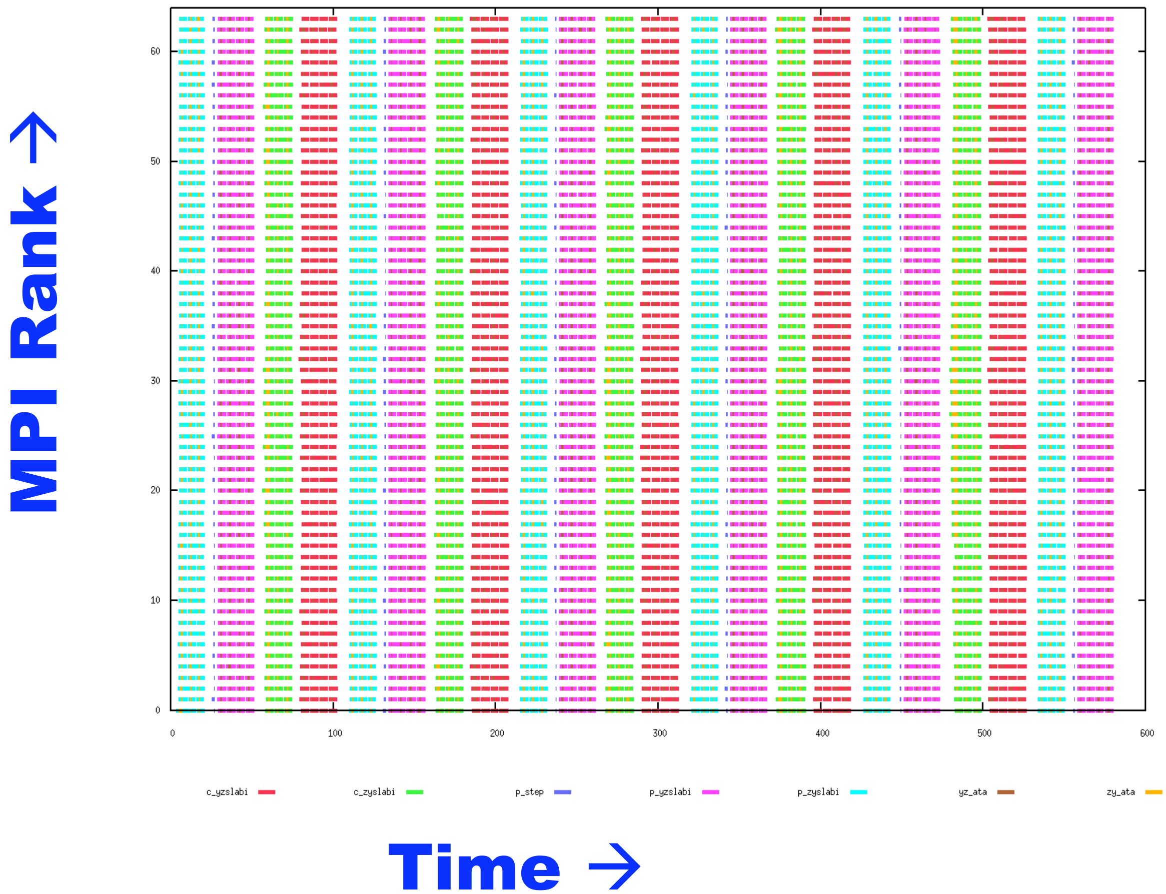
wall seconds = model (input, NP, arch)

= comm_model (I,N,A) + compute_model(I,N,A)

= Sum_i (model_i(I,N,A))

What does an HPC app look like?

Reduce → DGEMM → Exchange1 → Exchange2

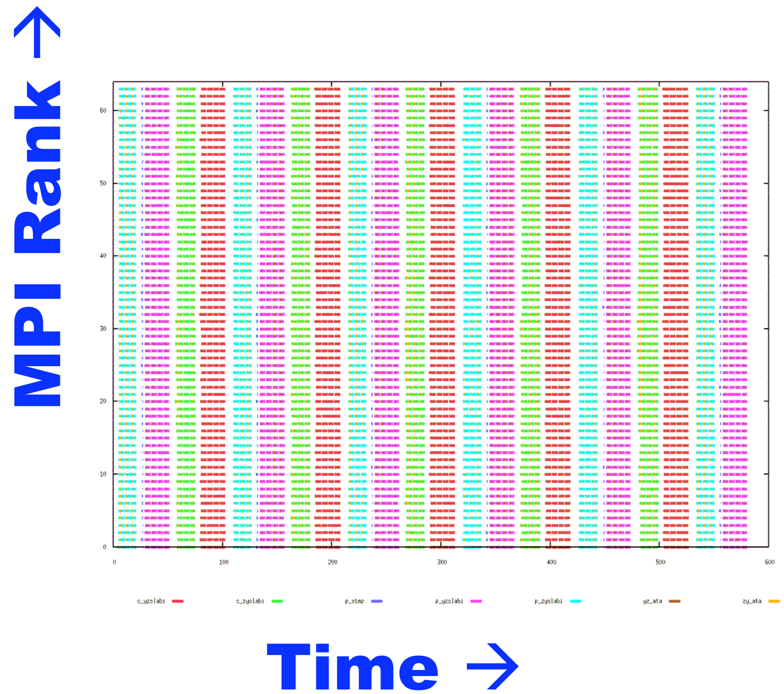


If you increase the resolution or extent of the data do you get a better picture?

Modeling: Application complexity

Q: What is the complexity of the application graph for this parallel code?

A: Not large



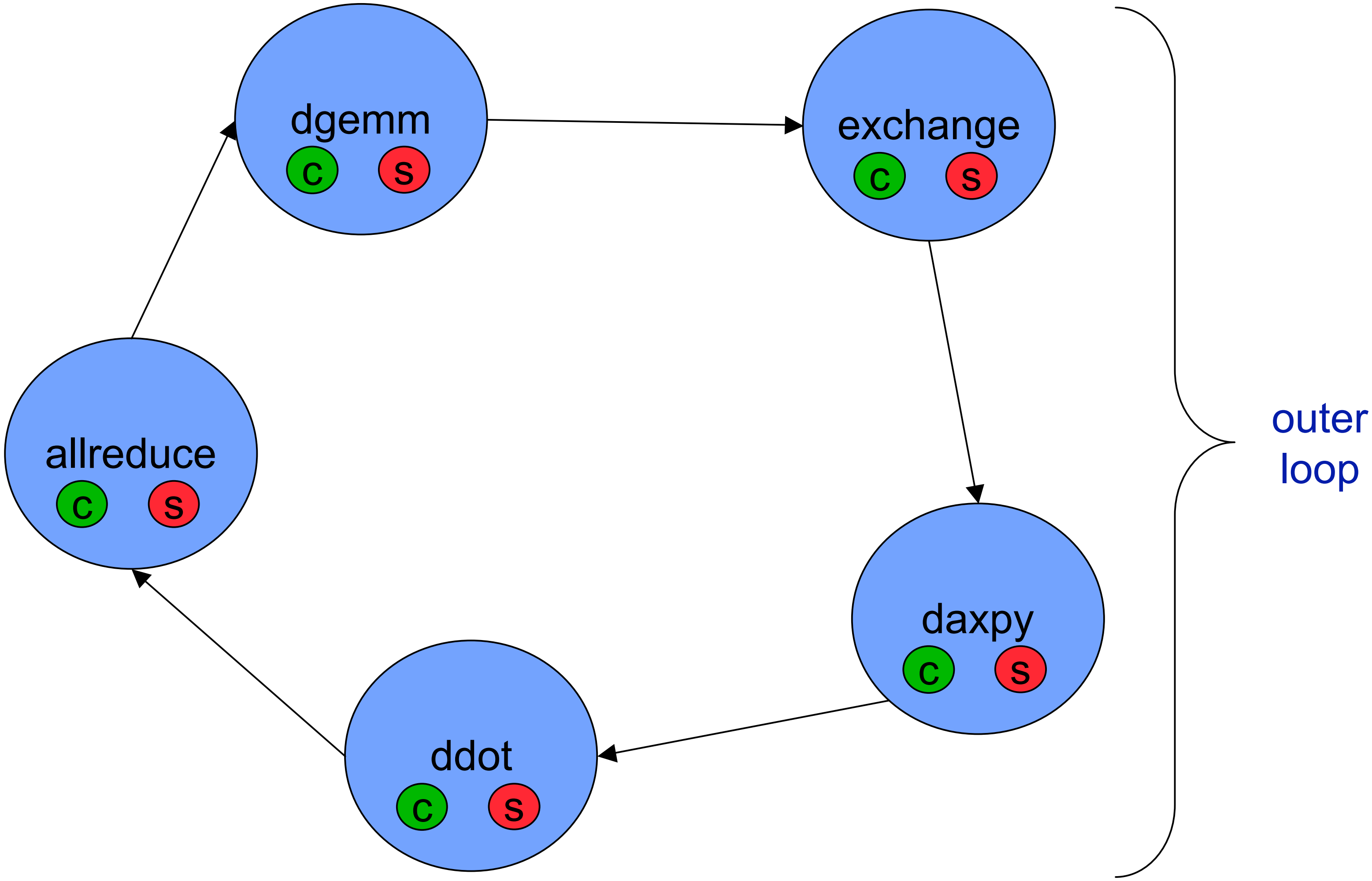
Q: How modelable are the nodes?

A: Communication is easier than compute

Application Sketching

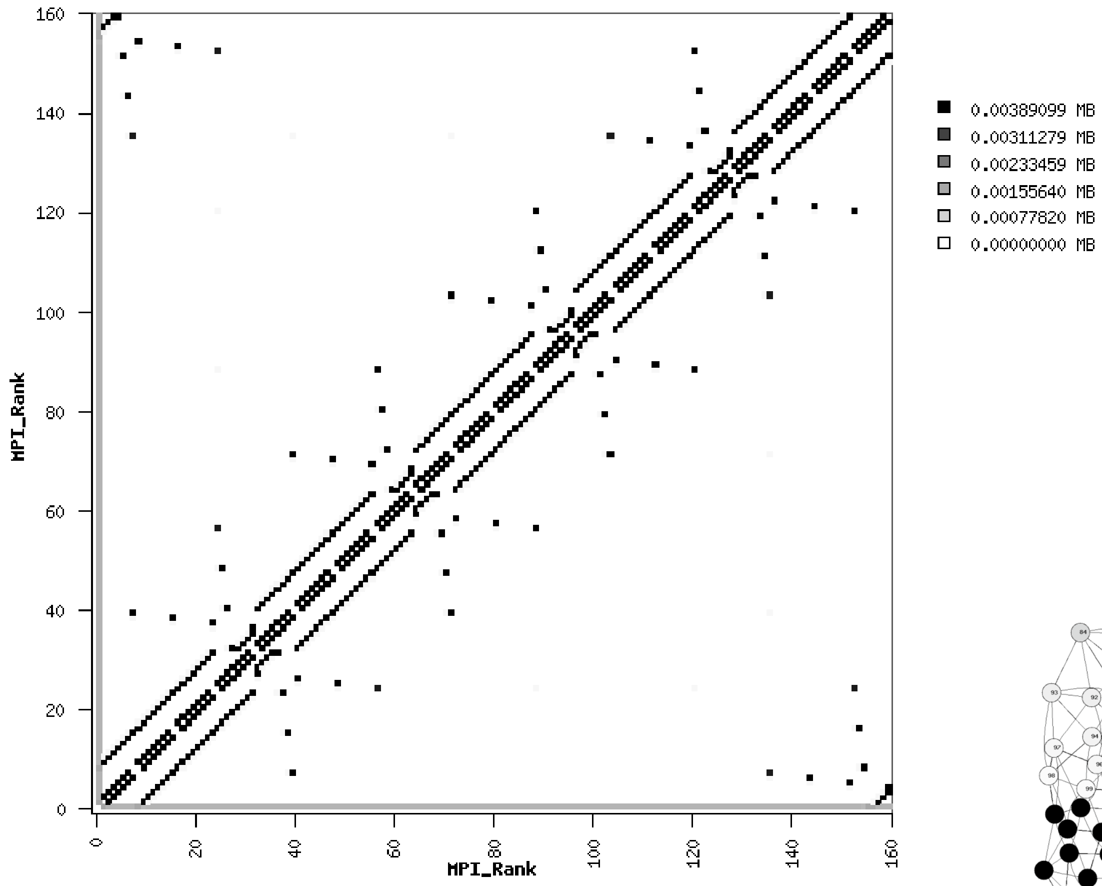
- Application sketching sometimes means the creation of applications from high level ideas about what the code should do.
- We're overloading the term to include the reverse process. Based on a profile can we make a cartoon model of the application?
- Models are the best profiles
- Hopefully this will be useful to
  - understand a code you're unfamiliar with
  - provide a concurrency-free picture of the code
  - provide a canvas to paint perf data on

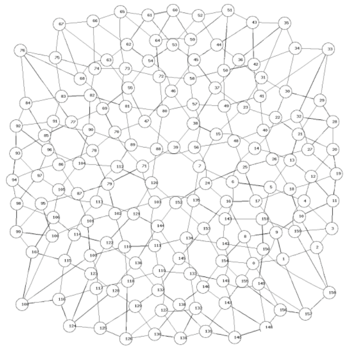
Directed Graph of Application Performance

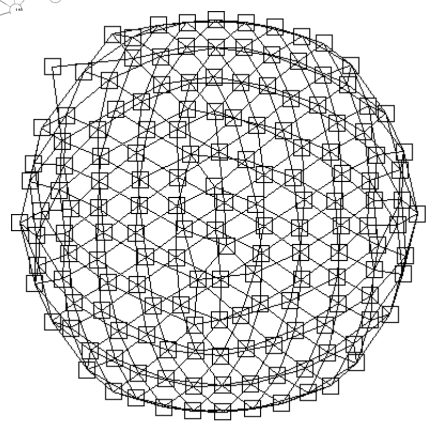


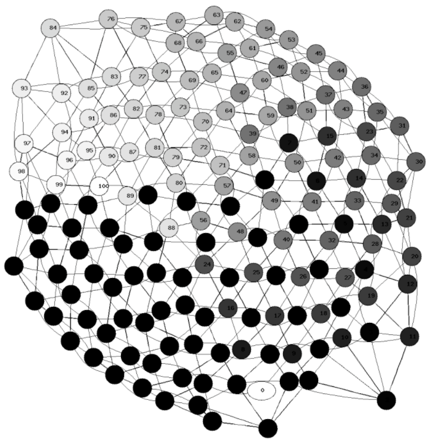
Application graph + Modelable nodes = Application Model

Application Shape and Symmetry : ex1 from the IPM website

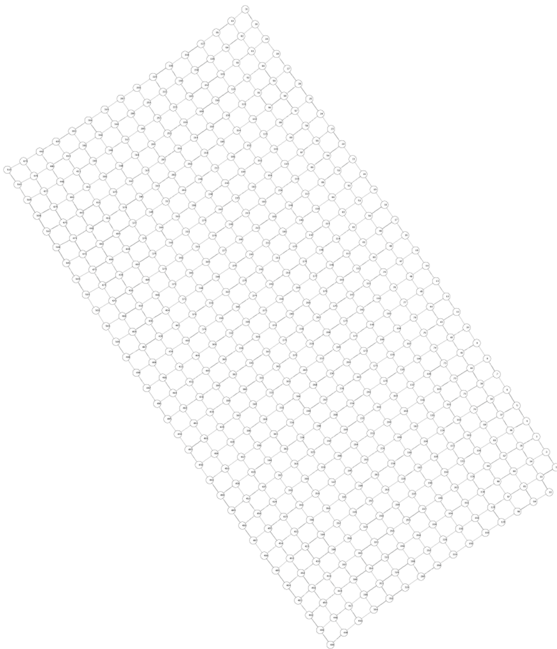


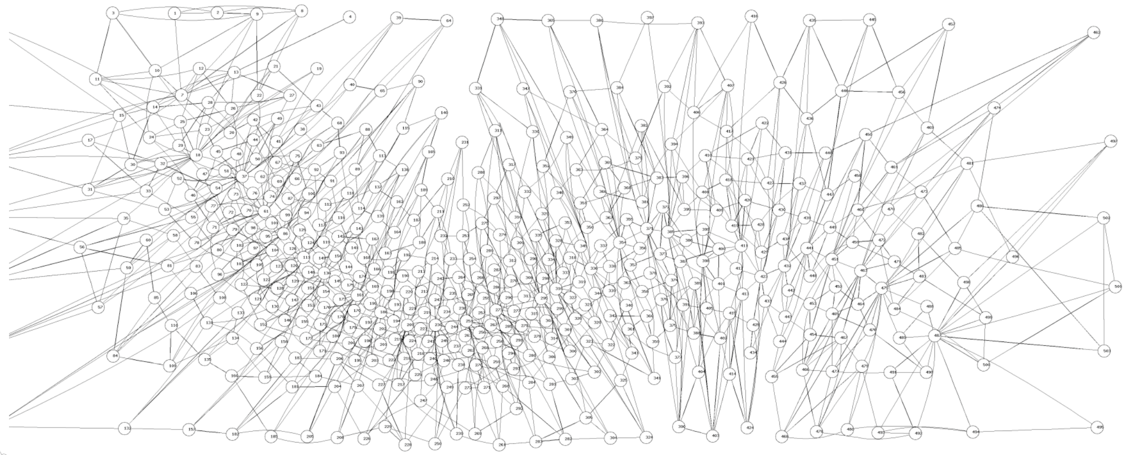


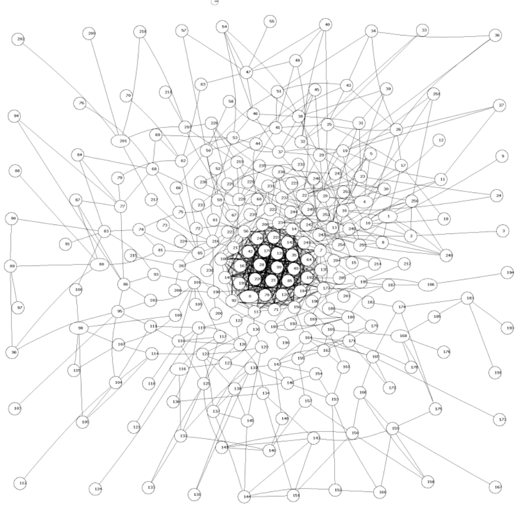


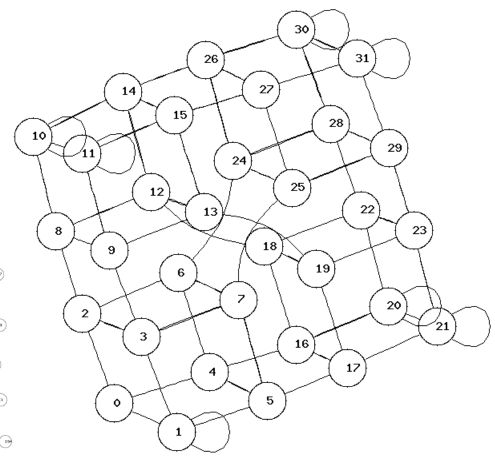


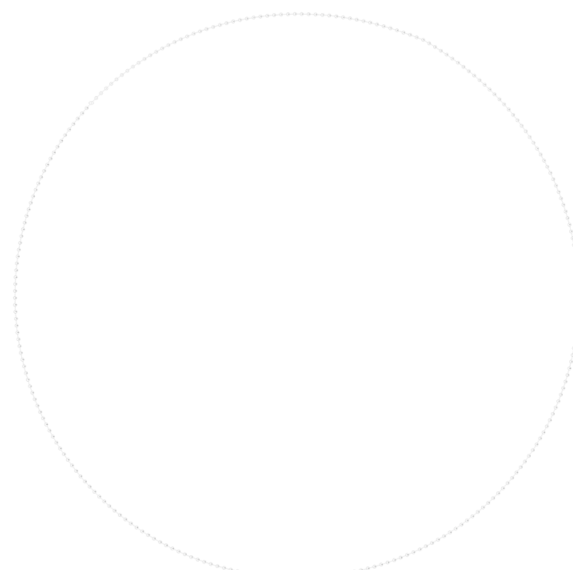
Application Shape and Symmetry











Key Points

- Building ease-of-use into performance analysis
- Building a quantitative basis for workload and procurement understanding
- Encouraging development of clear, portable, easy-to-use, failover tolerant, unintrusive, production quality APIs
- Implementing those APIs and layers in HPC resources and HPC frameworks
- The tools space is big. IPM is not a swiss army knife, we need a hierarchy of interoperable tools.