Debugging and Optimization Tools

Richard Gerber
NERSC User Services
David Skinner
NERSC Outreach, Software & Programming Group

UCB CS267
February 16, 2012

U.S. DEPARTMENT OF ENERGY | Office of Science
NERSC National Energy Research Scientific Computing Center
BERKELEY LAB Lawrence Berkeley National Laboratory

Outline

- Introduction
- Debugging
- Performance / Optimization

Videos, presentations, and references:

http://www.nersc.gov/users/training/courses/CS267/

Also see the DOE Advanced Computational Tools:
http://acts.nersc.gov

Introduction

- **Today's Talks**
  - Strategies for parallel performance (D. Skinner)
  - Debugging and optimization tools (R. Gerber)
- **Take Aways**
  - Common problems to look out for
  - How tools work in general
  - A few specific tools you can try
  - Where to get more information

Debugging

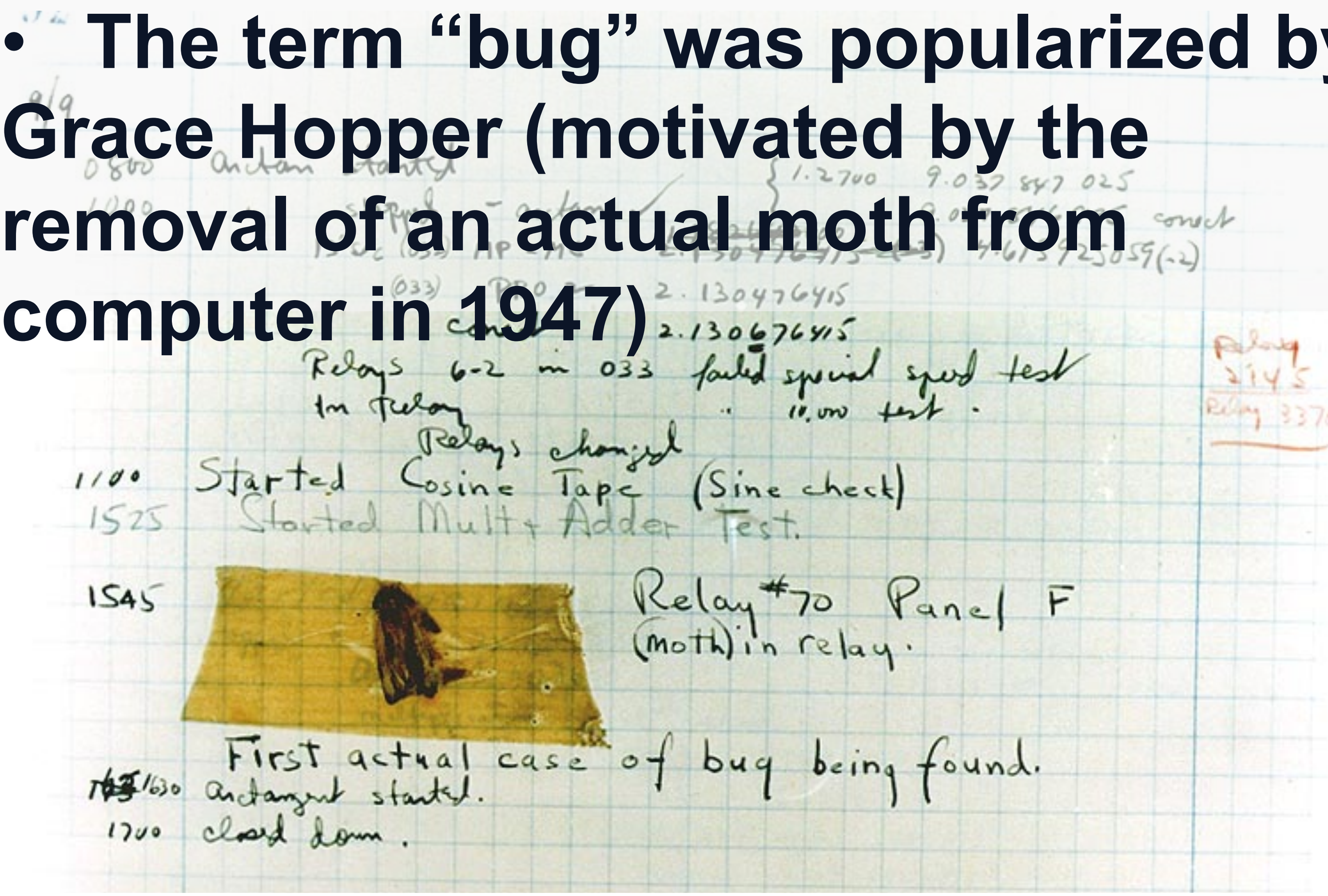
What is a Bug?

- **A bug is when your code**
  - crashes
  - hangs (doesn't finish)
  - gets inconsistent answers
  - produces wrong answers
  - behaves in any way you didn't want it to

NERSC

History

- The term “bug” was popularized by Grace Hopper (motivated by the removal of an actual moth from computer in 1947)



U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB Lawrence Berkeley National Laboratory

Common Causes of Bugs

- **"Serial"**
  - Invalid memory references
  - Array reference out of bounds
  - Divide by zero
  - Use of uninitialized variables
- **Parallel**
  - Unmatched sends/receives
  - Blocking receive before corresponding send
  - Out of order collectives
  - Race conditions

What to Do if You Have a Bug?

- **Find It**
  - You want to locate the part of your code that isn't doing what it's designed to do
- **Fix It**
  - Figure out how to solve it and implement a solution
- **Run It**
  - Check for proper behavior

Tools

printf, write

- Versatile, sometimes useful
- Doesn't scale well
- Not interactive
- Fishing expedition

Compiler / Runtime

- Bounds checking, exception handling
- Dereferencing of NULL pointers
- Function and subroutine interface checking

Serial gdb

- GNU debugger, serial, command-line interface
- See “man gdb”

Parallel debuggers Using X-Windows

- DDT
- Totalview

Compiler runtime bounds checking

Out of bounds reference in source code for program "flip"

```
…

allocate(put_seed(random_size))
…

bad_index = random_size+1
put_seed(bad_index) = 67
```

```
ftn  -c -g -Ktrap=fp –Mbounds flip.f90
ftn  -c -g -Ktrap=fp –Mbounds printit.f90
ftn  -o flip flip.o printit.o -g


% qsub –I –qdebug –lmppwidth=48
% cd $PBS_O_WORKDIR
%
% aprun –n 48 ./flip
0: Subscript out of range for array
   put_seed (flip.f90: 50)
    subscript=35, lower bound=1, upper
   bound=34, dimension=1
0: Subscript out of range for array
   put_seed (flip.f90: 50)
    subscript=35, lower bound=1, upper
   bound=34, dimension=1
```

Compiler Documentation

- **For a list of compiler options, see the man pages for the individual compilers**
  - man pgcc
  - man pgCC
  - man pgf90
  - man gcc
  - man gfortran
  - Etc.

Parallel Programming Bug

This code hangs because both Task 0 and Task N-1 are blocking on MPI_Recv

```
if(task_no==0) {

     ret = MPI_Recv(&herBuffer, 50, MPI_DOUBLE,
totTasks-1, 0, MPI_COMM_WORLD, &status);
     ret = MPI_Send(&myBuffer, 50, MPI_DOUBLE,
totTasks-1, 0, MPI_COMM_WORLD);

} else if (task_no==(totTasks-1)) {

     ret = MPI_Recv(&herBuffer, 50, MPI_DOUBLE, 0, 0,
MPI_COMM_WORLD, &status);
     ret = MPI_Send(&myBuffer, 50, MPI_DOUBLE, 0, 0,
MPI_COMM_WORLD);

}
```

Compile & Start DDT

Compile for debugging

```
hopper% make
cc -c -g hello.c
cc -o hello -g  hello.o
```

Set up the parallel run environment

```
hopper% qsub –I –V –lmppwidth=24
hopper% cd $PBS_O_WORKDIR
```

Start the DDT debugger

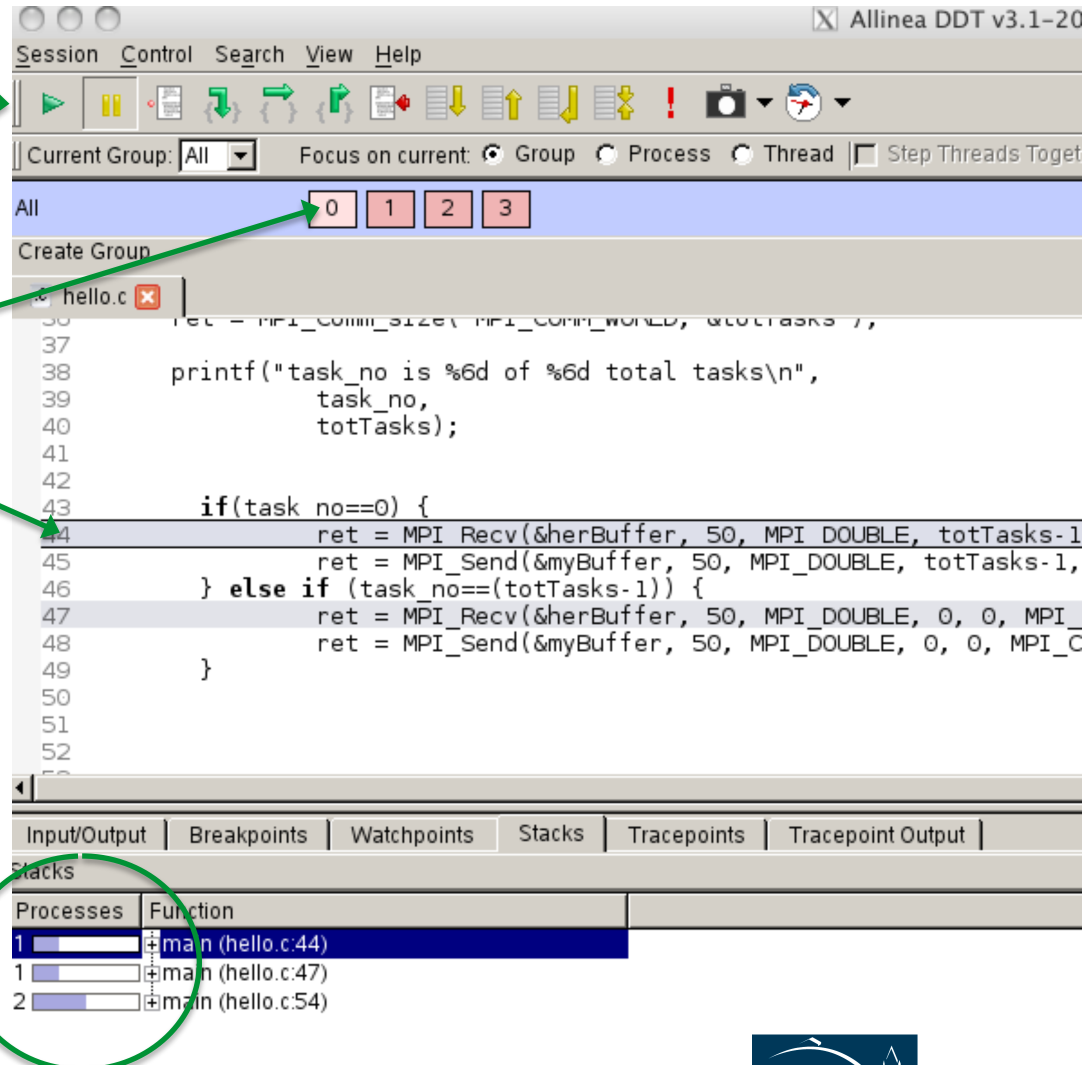
```
hopper% ddt ./hello
```

DDT Screen Shot

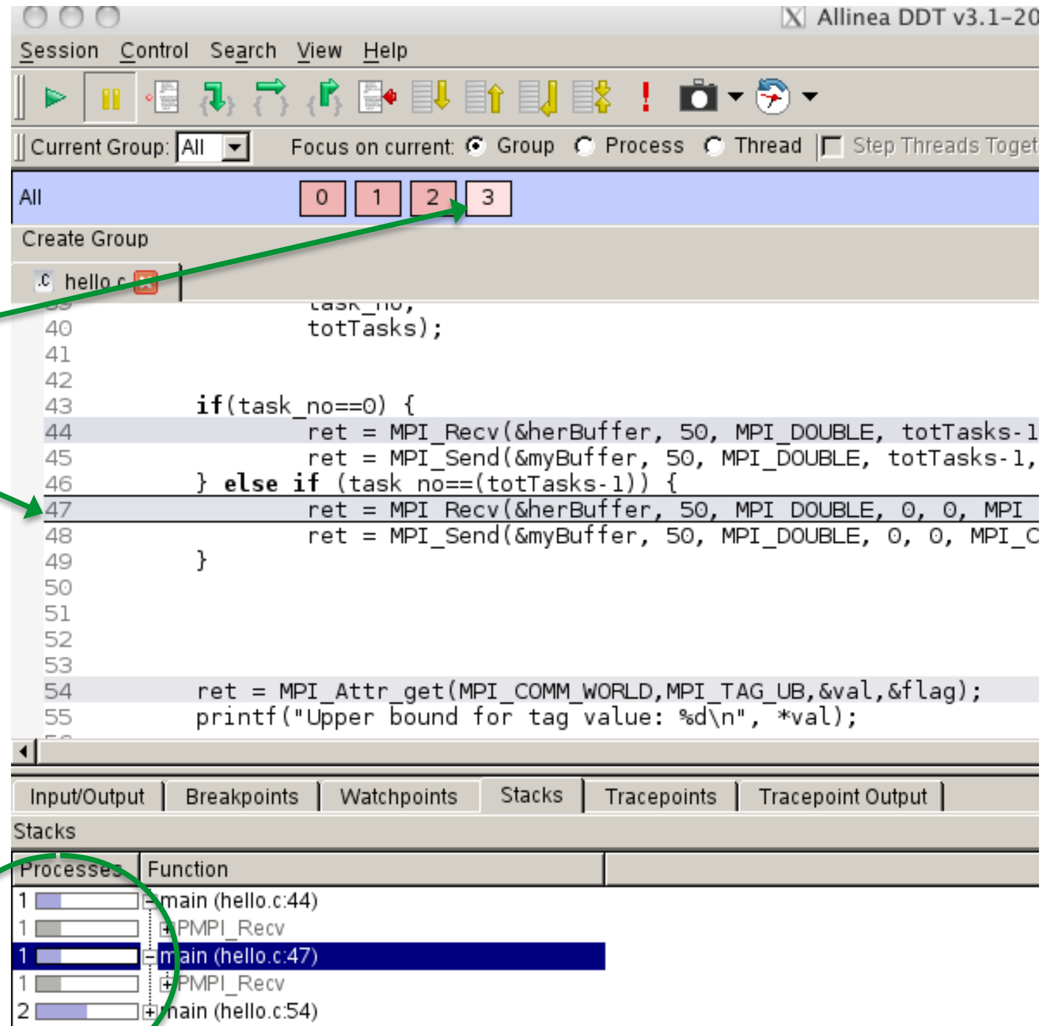
Press Go and then Pause when code appears hung.

Task 0 is at line 44

At hang, tasks are in 3 different places.



DDT Screen Shot



Task 3 is at line 47

At hang, tasks are in 3 different places.

NERSC

DDT video

- http://vimeo.com/19978486

- Or http://vimeo.com/user5729706

- This is out of date; I need to change the NX server from “Euclid” to “nx.nersc.gov and “hopp2” to “hopper”

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB Lawrence Berkeley National Laboratory

Other Debugging Tips

- **Try different compilers**
  - Diagnostic messages and language spec compliances differ
- **Look for memory corruption**
  - Bad memory reference in one place (array out of bounds) can make code crash elsewhere
  - It might appear that you're crashing on a perfectly valid line of code
- **Check the arguments to your MPI calls**
- **Call the NERSC Consultants (800-66-NERSC or 510 486-8600)**

Performance / Optimization

Performance Questions

- How can we tell if a program is performing well?
- Or isn't?
- If performance is not "good," how can we pinpoint why?
- How can we identify the causes?
- What can we do about it?

Performance Metrics

- **Primary metric: application time**
  - but gives little indication of efficiency
- **Derived measures:**
  - rates (Ex.: messages per unit time, Flops per second, clocks per instruction), cache utilization
- **Indirect measures:**
  - speedup, parallel efficiency, scalability

Optimization Strategies

- **Serial**
  - Leverage ILP on the processor
  - Feed the pipelines
  - Exploit data locality
  - Reuse data in caches
- **Parallel**
  - Minimize latency effects (aggregate messages)
  - Maximize work vs. communication
- **Both**
  - Minimize data movement (recalculate vs. send)
  - Memory locality on NUMA processors - first touch

Identifying Targets for Optimization: Profiling

- **Sampling**
  - Regularly interrupt the program and record where it is
  - Build up a statistical profile of time spent in various routines
  - Concentrate first on longest running sections or routines
- **Tracing**
  - Insert hooks into program to record and time program events (logging)
  - Reasonable for sequential programs
  - Unwieldy for large parallel programs (too much data!)

Identifying Targets for Optimization

- **Hardware Event Counters**
  - Special registers count events on processor
  - E.g. number of floating point instructions
  - Many possible events
  - Only a few can be recorded at a time (~4 counters)
  - Can give you an idea of how efficiently you are using the processor hardware

Typical Process

- (Sometimes) Modify your code with macros, API calls, timers
- Compile your code
- Transform your binary for profiling / tracing with a tool
- Run the transformed binary
  - A performance data file is produced
- Interpret the results with a tool

Performance Tools @ NERSC

- **Vendor Tools:**
  - CrayPat on Crays
- **Community Tools :**
  - TAU (U. Oregon via ACTS)
  - PAPI (Performance API)
  - gprof
- **IPM: Integrated Performance Monitoring**
  - A low overhead, low effort NERSC tool

Introduction to CrayPat

- **Suite of tools that provides a wide range of performance-related information**

- **Can be used for both sampling and tracing**
  - with or without hardware or network performance counters
  - Built on PAPI

- **Supports Fortran, C, C++, UPC, MPI, Coarray Fortran, OpenMP, Pthreads, SHMEM**

- **Man pages**
  - intro_craypat(1), intro_app2(1), intro_papi(1)

Using CrayPat

1. **Access the tools**
   - `module load perftools`
2. **Build your application; keep .o files**
   - `make clean`
   - `make`
3. **Instrument application**
   - `pat_build ... a.out`
   - Result is a new file, `a.out+pat`
4. **Run instrumented application to get top time consuming routines**
   - `aprun ... a.out+pat`
   - Result is a new file XXXXX.xf (or a directory containing .xf files)
5. **Run pat_report on that new file; view results**
   - `pat_report  XXXXX.xf  > my_profile`
   - `view my_profile`
   - Also produces a new file: XXXXX.ap2

Using Apprentice

- Optional visualization tool for Cray's perftools data
- Use it in a X Windows environment
- Uses a data file as input (`XXX.ap2`) that is prepared by `pat_report`

```
app2 [--limit_per_pe tags] XXX.ap2
```

Apprentice Basic View

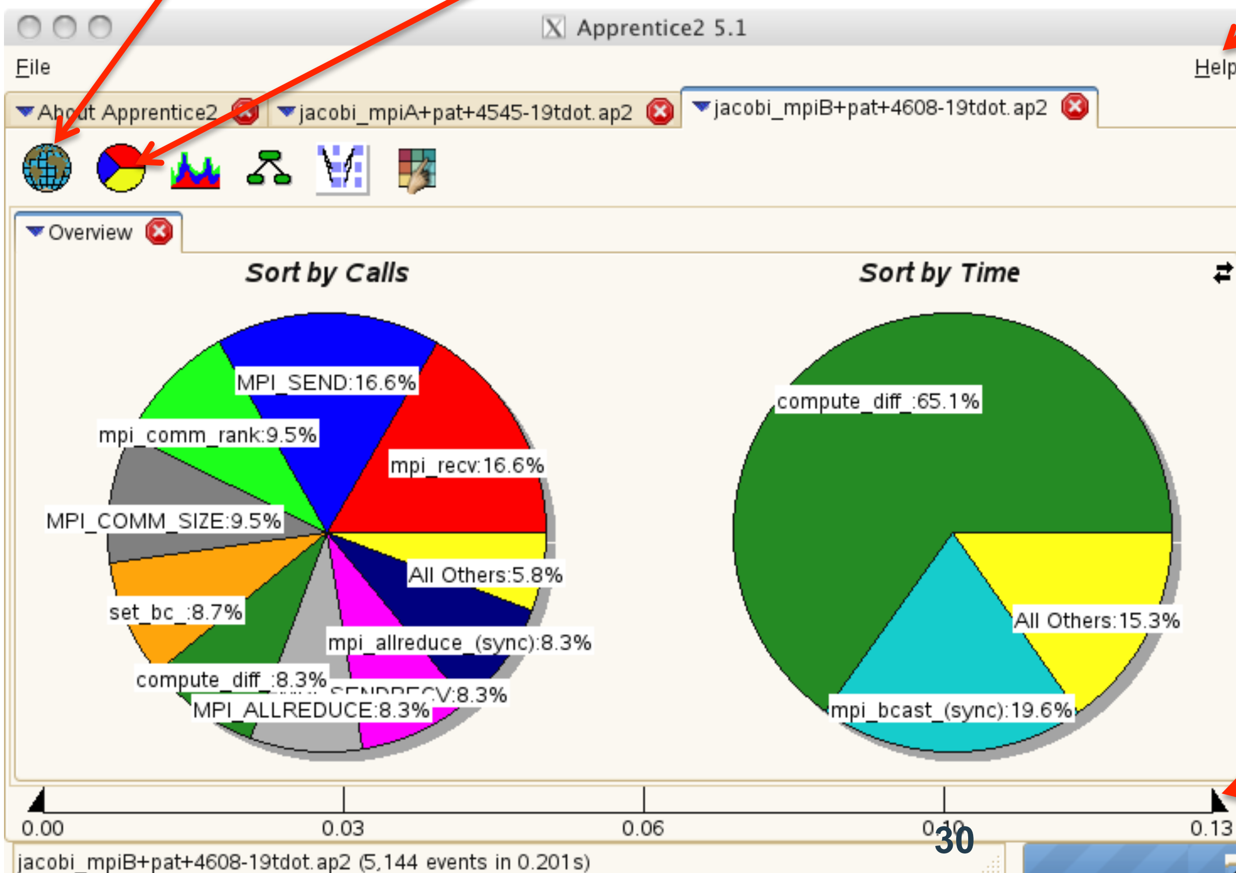
Can select new (additional) data file and do a screen dump

Worthless

Useful

Can select other views of the data

Can drag the "calipers" to focus the view on portions of the run



PAPI

- PAPI (Performance API) provides a standard interface for use of the performance counters in major microprocessors

- Predefined actual and derived counters supported on the system
  - To see the list, run 'papi_avail' on compute node via aprun:

```
qsub -I -lmppwidth=24
module load perftools
aprun -n 1 papi_avail
```

- AMD native events also provided; use 'papi_native_avail':

```
aprun –n 1 papi_native_avail
```

TAU

- **Tuning and Analysis Utilities**
- **Fortran, C, C++, Java performance tool**
- **Procedure**
  - Insert macros
  - Run the program
  - View results with pprof
- **More info than gprof**
  - E.g. per process, per thread info; supports pthreads
- **http://acts.nersc.gov/tau/index.html**

TAU Assignment

- **You will have a homework assignment using TAU**
  - `%module load tau`
  - Define paths in Makefile
  - Modify header file to define TAU macros
  - Add macro calls to the code
  - Compile and submit to batch queue
  - Use pprof to produce readable output
- **Good reference**
  - http://acts.nersc.gov/events/Workshop2011/Talks/TAU.pdf

Experience with NERSC Users

- NERSC has about 5,000 users
  - All levels of sophistication and experience
  - We're committed to supporting both the cutting edge & production HPC computing for the masses
- Users often ask for advice on which tools to use and we give them suggestions
- Our experience is that very few use programming/debugging/development tools
- A few users use a few tools a lot, but many try a tool only once

Why?

- Extremely effective?
- More likely: Too confusing, difficult, didn't work, don't know how to use, don't know which to use, tied to a platform, compiler, or language
- It's not that we don't have tools that address specific issues
  - TAU, PAPI, HPC Toolkit
  - Craypat, IBM HPC tools, OpenSpeedShop, Intel
  - Valgrind (memory debugging)
  - GPU/CUDA tools & compilers
  - Vampirtrace
- But do most users have the resources to learn how to use these tools, esp. when they don't know if there will be any benefit from any given one?

IPM

- **Integrated Performance Monitoring**
- **MPI profiling, hardware counter metrics, IO profiling (?)**
- **IPM requires no code modification & no instrumented binary**
  - Only a “module load ipm” before running your program on systems that support dynamic libraries
  - Else link with the IPM library
- **IPM uses hooks already in the MPI library to intercept your MPI calls and wrap them with timers and counters**

IPM

- **How it works (user perspective)**
  - `% module load IPM`*
  - Run program as normal
  - Look at results on the web
- **It's that easy!**
  - And extremely low overhead, so IPM is examining your production code

* (As long as your system supports dynamic load libs)

What IPM measures

- IPM "only" gives a high-level, entire-program-centric view
- Still, very valuable guidance
  - Shows whole-run info per MPI task, OpenMP thread, (CUDA under development)
  - Many pieces of data in one place
- Reveals what many users don't know about their code
  - High-water memory usage (per task)
  - Load balance
  - Call imbalance
  - MPI time
  - I/O time

IPM

```
# host    : s05601/006035314C00_AIX        mpi_tasks : 32 on 2 nodes
# start   : 11/30/04/14:35:34              wallclock : 29.975184 sec
# stop    : 11/30/04/14:36:00              %comm     : 27.72
# gbytes : 6.65863e-01 total               gflop/sec : 2.33478e+00 total
#                          [total]         <avg>          min           max
# wallclock                953.272       29.7897       29.6092       29.9752
# user                      837.25       26.1641         25.71         26.92
# system                      60.6       1.89375          1.52          2.59
# mpi                      264.267       8.25834       7.73025       8.70985
# %comm                                  27.7234       25.8873       29.3705
# gflop/sec                2.33478     0.0729619      0.072204     0.0745817
# gbytes                  0.665863     0.0208082     0.0195503     0.0237541
# PM_FPU0_CMPL         2.28827e+10   7.15084e+08   7.07373e+08   7.30171e+08
# PM_FPU1_CMPL         1.70657e+10   5.33304e+08   5.28487e+08   5.42882e+08
# PM_FPU_FMA           3.00371e+10    9.3866e+08   9.27762e+08   9.62547e+08
# PM_INST_CMPL         2.78819e+11   8.71309e+09   8.20981e+09   9.21761e+09
# PM_LD_CMPL           1.25478e+11   3.92118e+09   3.74541e+09   4.11658e+09
# PM_ST_CMPL           7.45961e+10   2.33113e+09   2.21164e+09   2.46327e+09
# PM_TLB_MISS          2.45894e+08   7.68418e+06   6.98733e+06   2.05724e+07
# PM_CYC                3.0575e+11   9.55467e+09   9.36585e+09   9.62227e+09
#                           [time]       [calls]        <%mpi>       <%wall>
# MPI_Send                 188.386        639616         71.29         19.76
# MPI_Wait                 69.5032        639616         26.30          7.29
# MPI_Irecv                6.34936        639616          2.40          0.67
# MPI_Barrier            0.0177442            32          0.01          0.00
# MPI_Reduce            0.00540609            32          0.00          0.00
# MPI_Comm_rank         0.00465156            32          0.00          0.00
# MPI_Comm_size        0.000145341            32          0.00          0.00
```

IPM Examples

Click on the metric you are want.

NERSC job details

http://www.nersc.gov/REST/jobs/job_details.php?stepid=732423.sdb&timestamp=1313679078&completion=1313679081

IPM Summary

Executable					./wrf.exe
Number of tasks	512	Aggregate GFlop/sec	0.1482	Average GFlop/sec/task	0.0003
Average wall secs	8.861e+01	Aggregate memory (GB)	32.2626	Average memory/task (GB)	0.0630
Average MPI secs/task	7.898e+01	MPI time %	89.14	Aggregate MPI calls made	7.027e+07

IPM Summary Statistics - 512 tasks

Metric	Sum over all tasks	Average (per task)	Task CV (%)	Task Minimum	Task Maxium
Aggregate Floating Point Operations (Flop x 10**9)	1.313e+01	2.565e-02	6.10	1.713e-02	2.758e-02
GFlop/sec	1.482e-01	2.895e-04	6.10	1.934e-04	3.114e-04
Maximum Memory Usage (GBytes)	3.226e+01	6.301e-02	10.12	5.701e-02	1.947e-01
Time Spent in MPI Routines (sec)	4.044e+04	7.898e+01	4.05	9.801e+00	8.359e+01
Wallclock Time (sec)	4.537e+04	8.861e+01	0.10	8.848e+01	8.895e+01

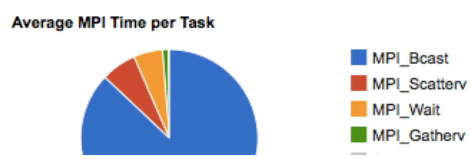
Memory in units of gigabytes; time in seconds.

Hardware counter statistics - 512 tasks

Counter Name	Sum over all tasks	Average (per task)	Task CV (%)	Task Minimum	Task Maxium
PAPI_FP_OPS	1.161799e+12	2.269139e+09	6.09	1.515023e+09	2.439529e+09

MPI Time Statistics - 512 tasks

Call	Sum over all tasks	Average (per task)	Task CV (%)	Task Minimum	Task Maxium	% of MPI	% of wall
MPI_Bcast	3.517e+04	6.869e+01	4.48	4.342e-01	7.269e+01	86.969	77.520
MPI_Scatterv	2.589e+03	5.057e+00	5.79	1.059e+00	5.540e+00	6.403	5.707
MPI_Wait	2.176e+03	4.249e+00	17.82	1.250e+00	4.968e+00	5.380	4.795
MPI_Gatherv	4.312e+02	8.422e-01	36.44	3.552e-03	2.271e+00	1.066	0.950
MPI_Isend	5.250e+01	1.025e-01	11.96	7.182e-02	1.259e-01	0.130	0.116
MPI_Irecv	1.033e+01	2.017e-02	10.21	1.217e-02	2.613e-02	0.026	0.023
MPI_Gather	1.021e+01	1.995e-02	502.07	1.391e-03	1.434e+00	0.025	0.023
MPI_Comm_rank	4.563e-01	8.913e-04	4.74	7.799e-04	1.404e-03	0.001	0.001
MPI_Comm_size	9.629e-02	1.881e-04	10.65	1.462e-04	4.859e-04	0.000	0.000
MPI_Init	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000	0.000	
MPI_Finalize	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000	0.000	



IPM Examples

NERSC job details

http://www.nersc.gov/REST/jobs/ipm_summary.php?stepid=619349.sdb&name=gflops&timestamp=1310766809

Metric	Sum	Mean	Std. Dev.	CV (%)	Minimum	Maximum
Aggregate Floating Point Operations (Flop x 10**9)	3.011e+02	1.470e-01	4.946e-03	3.36e+00	1.395e-01	2.161e-01
GFlop/sec	6.147e-01	3.002e-04	1.008e-05	3.36e+00	2.847e-04	4.411e-04
Maximum Memory Usage (GBytes)	4.101e+02	2.002e-01	9.606e-03	4.80e+00	1.781e-01	2.448e-01
Time Spent in MPI Routines (sec)	1.228e+06	5.995e+02	4.984e+01	8.31e+00	5.177e+02	6.801e+02
Wallclock Time (sec)	1.003e+06	4.898e+02	6.428e-02	1.31e-02	4.898e+02	4.927e+02

CV = Coefficient of Variance = (Standard Deviation / Mean)

Task distribution of *Aggregate Floating Point Operations (Flop x 10**9)* - as a percentage of maximum

The MPI rank is the sum of the column and row indices in the table.

Table Columns: 64

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
0	94	70	68	67	66	66	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65
64	66	65	65	66	66	65	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	67	66	66	66	66	66	66	66	67	66	66	67	67	67	67	66	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67
128	67	67	67	67	67	67	67	67	67	67	67	67	67	67	68	68	67	68	67	68	68	68	67	68	67	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68
192	68	68	68	68	68	68	68	68	68	68	69	68	68	68	69	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	69	68	68	68	68	68	68	68	68	68	69	68	68	68	68	69	68	68	69	68	68	68	68	68	68	68	68	68	68
256	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	69	68	69	68	69	68	69	68	68	68	69	68	68	69	68	68	68	68	68	68	69	68	68	68	68	68	68	68	69	69	68	68	68	69	68	68	68	69	68	69	68	68	68	69	68
320	68	68	68	68	68	68	69	68	68	69	68	68	69	68	68	68	68	68	69	68	68	68	68	68	68	69	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	67	68	67
384	68	68	68	68	68	68	68	67	67	68	67	68	67	67	67	67	67	67	67	67	68	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	66	67	67	67	67	66	67	67	66	67	66	66	66	67	66
448	66	66	67	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	67	66	66	66	67	67	67	67	67	67	67	67	67	67	67	67	67	67	68	68	68	69	69	70	71	74	99
512	100	74	72	71	70	69	69	69	69	68	68	68	68	68	68	68	68	68	68	67	68	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	68	67	68	67	68	67	68	68	68	68	67	68	68	68
576	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	69	68	69	68	69	68	68	69	69	68	69	69	69	69	68	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69
640	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	70	69	69	70	69	69	70	69	70	70	69	70	69	69	70	69	70	69	70	69	69	70	69	70	69	70	70	70	70	69	70	69	70	70	69	70	69	70	70	70	70	70	70	70	70	70	70
704	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	69	70	69	70	70	70	70	70	70	70	69	70	69	70	70	69	70	69	69	70	69	70	70	69	70	69	69	70	69	70
768	69	69	70	69	69	70	69	69	69	70	70	69	70	69	70	70	70	70	69	70	69	70	69	70	69	70	69	70	69	70	70	69	70	69	70	69	70	70	70	70	70	70	70	70	70	69	70	70	70	69	69	70	70	69	70	70	70	69	70	70	69	70	70	69
832	70	70	70	69	70	69	70	69	69	70	69	69	70	69	69	69	69	69	70	69	69	69	69	69	69	70	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	68	69	69	69	69	69	69	69	68	69	68	69	69	69	68	69	68
896	68	69	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	67	68	68	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67
960	66	67	67	66	67	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	65	66	65	66	65	66	65	65	65	65	65	66	65	66	65	66	66	66	66	66	66	66	66	67	67	68	70	95
1024	100	74	72	71	70	69	69	69	69	68	68	68	68	68	68	68	68	68	68	67	68	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	68	68	68	68	68	67	68	68	68	68	67	68	68	68
1088	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	69	68	69	68	69	68	69	68	68	69	69	68	69	69	69	69	68	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69
1152	69	69	69	69	69	69	69	69	69	69	69	69	69	69	70	70	69	70	69	69	70	69	69	70	69	70	70	69	70	69	69	70	69	70	69	70	70	69	70	69	70	69	70	70	70	70	70	70	69	70	70	69	70	69	70	70	70	70	70	70	70	70	70	70
1216	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	69	70	70	70	70	70	70	70	69	70	69	70	70	69	70	70	69	70	69	70	70	69	70	69	69	70	70	70
1280	69	69	70	69	69	70	70	69	70	70	70	69	70	69	70	70	70	70	69	70	69	70	69	70	69	70	69	70	69	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	69	70	70	70	69	70	70	70	69	70	70	70	69	70	70	69	70	70	69
1344	70	70	70	70	70	70	70	69	69	70	69	69	70	69	69	69	69	69	70	69	69	69	69	69	69	70	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	68	69	68
1408	68	69	69	68	68	68	69	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	67	68	68	68	68	67	68	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67
1472	66	67	67	66	67	67	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	65	66	65	66	65	66	65	66	65	65	65	66	65	66	65	66	66	66	66	66	66	66	66	67	67	68	70	95
1536	94	70	68	67	66	66	66	66	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65	66	65	66	65	66	65	66	66	66	66	65	66	66	66
1600	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	66	67	66	66	66	67	66	67	66	67	67	66	67	66	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	68	67	67	67	67	68
1664	68	67	67	67	68	68	68	68	68	67	68	68	68	68	68	68	67	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	69	68	68	68	68	68	68	68	68	69	69	68	69	68	68	69	68	69	68	68	68	68	68	68
1728	69	68	69	68	69	68	68	68	69	68	69	68	68	68	69	68	68	68	68	68	69	68	68	68	69	69	69	68	69	68	68	68	69	68	68	69	68	68	69	68	68	69	68	69	68	69	68	69	68	68	69	68	68	69	68	68	69	68	69	68	68	69	68	68
1792	68	68	69	68	68	68	68	68	68	68	69	68	69	68	69	69	68	69	68	69	68	69	68	69	68	69	68	69	68	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	68	69	69	69	68	69	69	69	69	69	69	69	68	69	69	68	69	69	69
1856	69	69	69	69	69	69	69	68	69	69	69	68	69	69	69	68	69	69	69	69	69	68	69	69	69	69	68	69	68	69	69	69	68	68	69	68	68	68	69	68	68	68	69	68	68	68	69	68	68	68	68	68	68	68	68	68	68	68	68	68	69	68	68	68
1920	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	68	67	68	68	68	68	67	67	67	67	67	68	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67
1984	66	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	66	67	67	67	67	67	67	66	67	67	67	66	67	67	67	67	67	67	66	67	67	67	67	67	67	67	67	67	67	67	67	67	67	67	68	68	68	68	68	69	69	69	70	71	74	99

gflops vs. MPI Rank

IPM Examples

NERSC job details

http://www.nersc.gov/REST/jobs/mpi_functions.php?stepid=619349.sdb&f_name=MPI_Allreduce&timestamp=1310766809

Time spent by each task in *MPI_Allreduce* as a percentage of the maximum value

The MPI rank represented by each cell in the table is the sum of the cell's column and row indices.

Table Columns: 48

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	100	100	100	99	99	98	98	98	98	97	97	97	97	97	97	96	94	95	94	94	94	93	93	93
48	82	82	81	81	80	81	80	80	79	79	79	79	79	80	79	79	79	78	78	77	76	77	76	76
96	76	77	76	75	75	75	75	74	74	74	75	74	74	75	74	74	75	74	74	73	73	74	73	72
144	67	66	66	65	64	65	64	64	64	64	64	63	64	64	64	63	55	54	54	53	53	54	53	53
192	50	50	50	49	49	50	49	49	49	50	49	49	50	50	50	50	49	49	49	48	48	49	48	48
240	51	51	50	50	50	50	51	50	49	50	49	49	50	51	51	51	58	58	57	57	57	57	57	57
288	58	58	58	58	57	58	58	57	57	58	58	57	58	60	59	59	56	56	56	56	55	56	57	56
336	62	61	62	62	62	63	62	62	63	63	64	63	64	65	65	66	65	65	65	65	65	65	65	65
384	57	56	56	56	57	57	57	57	58	58	59	58	59	60	60	61	60	59	60	59	59	60	60	60
432	80	80	80	79	80	80	81	80	80	81	81	81	81	82	82	83	77	76	76	76	76	76	76	76
480	75	75	75	75	75	75	76	75	76	76	76	76	77	78	78	78	79	79	79	79	79	79	79	79
528	86	87	86	86	85	85	85	85	84	84	84	84	83	83	83	83	85	84	84	83	83	83	82	82
576	73	73	73	71	71	72	71	71	71	71	71	70	71	71	71	71	68	67	67	66	66	66	66	65
624	65	65	65	64	64	64	64	63	63	63	63	62	63	63	62	62	55	55	55	54	54	53	53	53
672	44	44	44	43	43	44	43	43	43	44	43	43	43	44	43	42	43	42	42	42	41	42	41	41
720	41	40	41	40	39	40	40	40	38	39	39	39	39	40	40	39	44	44	45	43	44	44	44	44
768	54	54	53	53	53	54	54	54	53	54	54	54	54	55	55	54	51	52	52	51	51	51	52	51
816	51	52	52	52	52	52	52	52	52	54	53	53	54	55	54	55	54	54	54	54	54	54	54	54
864	61	60	61	61	61	61	61	61	61	61	61	61	62	63	63	64	59	58	59	59	59	59	60	60
912	58	58	59	59	58	59	59	59	60	60	60	60	61	62	62	62	68	68	68	68	68	68	68	68
960	75	74	75	75	75	75	75	75	76	76	76	76	77	77	78	78	77	77	77	77	77	77	78	78
1008	78	78	78	79	79	79	79	79	80	80	80	80	81	83	84	86	92	92	92	91	91	91	91	90
1056	82	82	82	81	81	81	80	80	80	80	79	79	80	80	80	80	75	75	74	74	73	74	73	73
1104	67	66	67	65	65	65	65	64	64	64	64	64	64	65	64	64	69	69	69	68	67	68	67	67
1152	55	55	55	54	53	53	53	53	52	53	53	52	53	53	53	53	51	50	51	50	49	49	49	49
1200	43	43	42	43	41	43	42	41	41	41	41	40	41	42	41	40	42	42	42	42	41	42	42	41
1248	43	43	43	42	42	43	42	42	42	43	43	42	43	44	44	43	43	43	42	42	43	42	43	42
1296	51	51	51	50	51	50	51	50	51	50	51	50	51	52	52	52	50	51	50	50	50	50	50	50
1344	54	54	55	54	54	55	55	55	54	55	55	55	55	57	57	57	53	53	53	53	53	54	53	53
1392	54	54	55	54	55	55	55	55	55	56	56	56	56	58	58	59	56	56	56	56	56	57	56	57
1440	70	70	70	70	70	70	70	71	70	70	71	71	71	73	73	73	80	81	81	80	81	81	82	81
1488	79	79	79	79	79	79	80	80	80	81	81	81	81	83	83	83	77	77	77	78	77	77	78	77
1536	98	97	97	96	96	95	95	95	95	94	94	94	94	94	94	94	88	90	89	88	87	87	87	86
1584	77	77	77	76	76	76	76	76	75	75	75	75	75	75	75	75	75	75	74	73	73	73	73	72
1632	75	76	75	74	74	74	74	74	73	73	73	73	73	73	73	73	70	70	69	69	68	69	68	68
1680	63	61	62	61	60	61	61	60	60	60	60	60	59	60	60	59	53	53	53	52	52	52	52	52
1728	51	51	50	50	49	50	50	50	49	50	50	50	50	51	50	50	50	49	49	49	48	49	49	48
1776	48	48	47	47	48	47	48	47	47	47	47	47	48	48	48	47	55	56	55	55	55	55	55	55
1824	53	54	54	54	53	53	53	53	54	55	54	54	55	56	56	56	55	55	55	55	55	55	55	55
1872	55	54	55	55	54	55	55	55	55	56	56	56	56	58	57	58	57	56	56	56	56	57	57	57
1920	58	58	58	58	58	59	59	59	59	59	60	59	60	61	61	62	64	63	64	64	64	65	65	65
1968	83	83	83	83	83	84	84	83	84	84	84	84	85	86	85	86	81	81	81	81	81	81	81	81
2016	83	83	83	83	83	83	84	83	84	84	84	84	85	86	86	85	88	89	89	89	89	89	89	89

	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
0	92	92	92	92	92	92	92	92	89	89	88	88	87	87	87	87	87	86	86	86	86	86	86	86
48	76	77	76	76	76	77	76	76	73	72	73	71	71	71	71	70	70	70	70	70	69	71	70	69
96	72	72	72	72	72	73	72	71	69	69	69	68	68	68	68	67	67	68	67	67	67	68	67	67
144	53	54	53	53	53	54	54	53	53	53	52	52	51	53	51	51	51	50	50	50	51	52	51	50
192	47	47	47	47	47	49	49	48	50	51	51	50	50	50	50	50	50	50	50	50	51	52	52	51
240	57	57	57	57	57	59	58	58	56	56	56	55	56	56	56	55	56	56	57	56	57	58	58	58
288	56	58	57	57	58	59	59	60	58	58	59	58	58	58	58	58	58	58	58	59	59	60	60	61
336	65	65	65	65	66	67	67	68	61	61	62	61	61	62	62	62	62	63	63	63	63	65	65	66
384	60	61	61	61	61	63	62	63	69	69	68	68	69	69	69	69	70	70	70	71	70	72	72	72
432	77	77	77	77	78	78	79	80	78	78	78	78	78	78	78	79	79	79	79	80	80	81	82	82
480	80	80	80	80	81	82	83	85	92	92	92	92	91	91	91	91	91	90	90	90	90	90	90	90
528	82	82	82	82	82	82	82	81	78	77	77	76	76	76	76	76	75	75	75	75	75	75	75	75
576	65	65	64	65	64	66	65	64	68	68	68	66	66	66	66	66	66	66	66	65	65	66	65	65
624	53	53	53	53	53	53	53	53	51	50	51	50	49	50	49	49	49	49	48	48	49	50	49	48
672	41	40	40	40	41	42	41	40	41	42	42	41	41	41	41	41	41	42	41	41	41	42	41	41
720	44	44	44	44	45	45	46	45	44	44	44	44	44	44	44	43	43	44	43	43	44	45	45	44
768	52	52	52	52	52	53	53	53	52	53	53	53	52	53	53	52	52	53	53	52	53	55	54	54
816	54	54	55	55	55	56	56	57	58	58	58	58	58	59	59	59	60	60	61	60	61	61	61	62
864	60	61	61	61	61	63	62	63	55	55	56	56	56	56	56	56	56	56	57	57	57	58	58	59
912	69	69	69	69	70	71	71	72	79	80	80	79	80	80	81	80	80	81	81	81	81	82	82	83
960	78	78	79	79	79	80	81	81	74	74	74	74	74	74	75	74	75	75	75	75	76	77	77	77
1008	90	90	90	89	90	90	90	89	84	85	84	83	82	82	81	81	81	81	80	80	80	80	80	80
1056	72	72	72	72	72	72	72	72	71	71	71	70	70	70	70	69	69	69	69	69	69	69	69	69
1104	67	67	67	66	66	67	66	66	64	64	64	63	63	63	62	62	62	62	62	62	62	62	62	61
1152	49	49	48	48	48	49	49	48	43	43	43	42	42	43	42	42	42	42	42	41	42	42	42	41
1200	41	42	41	42	42	43	42	42	41	41	41	41	40	41	40	40	39	39	39	40	39	41	41	40
1248	42	42	42	42	42	43	43	43	53	54	53	53	53	53	53	53	53	54	54	54	53	55	55	54
1296	50	51	51	50	51	53	52	52	51	52	52	51	51	52	52	52	51	53	52	52	54	55	54	55
1344	54	54	55	54	55	56	56	57	56	55	55	55	55	55	55	55	55	56	55	56	56	58	58	59
1392	57	57	58	57	58	59	59	60	62	61	62	62	62	63	63	63	63	63	64	64	64	65	65	66
1440	82	83	83	83	83	84	84	85	78	77	77	77	77	77	77	77	78	78	78	78	79	80	80	80
1488	78	78	79	79	79	80	80	79	81	81	81	81	82	82	82	82	83	83	83	84	84	85	86	88
1536	86	86	86	85	85	85	85	85	85	85	84	84	84	83	83	83	83	83	82	82	83	83	83	82
1584	72	72	72	72	72	72	72	72	70	70	70	69	69	69	69	68	68	68	68	68	68	69	68	68
1632	67	68	67	67	68	68	68	67	67	67	67	67	66	66	66	65	65	65	65	64	65	66	65	64
1680	51	52	51	51	51	51	51	50	53	52	52	52	51	52	52	51	50	50	50	50	51	51	51	50
1728	47	48	48	48	48	49	49	48	47	48	48	47	47	47	47	47	47	47	48	47	48	48	48	48
1776	55	56	56	56	56	57	57	56	54	54	54	53	54	54	55	53	54	54	54	53	54	55	55	55
1824	55	56	55	55	57	58	57	58	57	57	58	58	57	58	58	58	58	58	59	59	59	60	60	61
1872	57	58	57	58	58	59	60	61	56	56	56	57	57	57	57	57	57	58	57	58	58	60	59	60
1920	65	65	65	66	66	67	67	68	72	71	72	72	72	72	72	72	72	73	73	73	73	75	75	75
1968	82	82	82	82	83	83	84	84	83	82	83	82	82	83	83	83	83	84	84	84	84	85	85	85
2016	90	90	90	90	91	92	92	92																

Time vs. MPI Rank for *MPI_Allreduce*

IPM Examples

NERSC job details

http://www.nersc.gov/REST/jobs/ipm_summary.php?stepid=627129.sdb&name=memory&timestamp=1311046935

Task distribution of *Maximum Memory Usage (GBytes)* - as a percentage of maximum

The MPI rank is the sum of the column and row indices in the table.

Table Columns: 32

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	67	68	67	69	67	69	68	70	68	69	69	70	69	70	71	71	70	72	69	71	71	73	70	72	71	72	71	72	72	74	72	73
32	69	71	70	72	70	72	70	72	71	72	72	73	71	72	72	73	73	75	72	74	73	75	72	74	75	76	74	75	74	75	74	75
64	71	71	72	73	72	73	73	74	72	72	73	73	73	73	74	74	74	75	74	75	76	77	75	76	76	76	75	75	77	77	77	76
96	74	75	75	76	74	75	75	76	75	75	76	76	75	75	76	75	78	79	77	79	78	79	77	78	79	79	78	78	79	78	78	78
128	86	88	86	87	87	89	86	89	88	89	87	88	88	89	88	89	86	87	85	87	87	89	86	88	87	88	86	87	88	89	88	89
160	89	91	89	91	89	91	88	90	91	92	90	91	90	91	90	91	89	90	88	90	88	90	88	90	90	91	90	91	90	91	89	90
192	95	96	95	96	96	97	96	97	96	96	95	95	97	97	97	96	95	95	94	95	96	97	95	96	95	95	95	95	97	97	96	96
224	98	99	98	98	98	99	97	98	100	99	98	98	99	99	98	97	98	99	97	98	98	99	97	98	99	99	98	97	99	98	98	97
256	70	72	71	73	70	72	71	73	72	73	72	73	71	72	73	73	73	75	73	74	73	75	72	74	75	76	74	75	74	76	74	75
288	67	69	68	70	66	68	66	68	69	70	70	71	67	68	68	69	71	73	70	72	69	71	68	70	72	73	72	73	70	71	70	71
320	74	75	76	76	74	75	75	77	75	75	76	76	75	75	76	76	78	79	77	78	78	79	77	79	80	79	79	78	79	79	78	78
352	72	73	73	74	70	71	71	72	72	72	73	73	70	70	72	71	75	76	75	76	74	74	73	74	76	76	76	75	74	74	73	73
384	90	92	89	91	90	91	89	91	91	93	91	92	91	92	91	91	89	91	89	90	89	91	88	90	91	92	90	91	90	91	90	91
416	87	89	87	89	85	87	85	86	89	90	88	89	87	88	86	87	86	88	86	88	85	87	84	86	88	89	87	88	86	87	86	87
448	99	100	98	99	99	100	98	99	100	100	99	99	100	100	99	99	99	99	98	99	98	99	98	99	99	99	99	99	99	99	99	98
480	96	97	95	96	94	95	93	94	97	97	96	96	95	95	94	94	96	97	95	96	94	95	93	94	97	96	96	95	95	94	94	94
512	71	72	72	73	72	73	73	74	71	72	71	72	72	73	73	74	75	75	74	75	76	77	75	76	74	75	73	74	75	76	75	76
544	74	75	75	76	74	75	75	76	74	75	75	76	74	75	75	76	78	79	77	78	78	79	77	78	77	78	77	78	77	78	77	78
576	68	69	70	70	69	70	70	72	68	67	69	68	69	68	71	69	72	73	71	72	73	74	72	74	72	70	71	70	73	72	73	71
608	72	73	73	74	72	72	72	74	71	70	72	71	71	70	72	71	76	77	75	76	76	76	74	76	75	74	74	73	75	74	74	73
640	94	95	94	94	95	96	94	95	94	94	93	93	94	95	94	94	94	94	93	94	95	96	94	95	93	94	92	93	94	95	93	95
672	98	98	97	97	97	98	96	97	97	98	96	97	96	97	96	97	97	98	96	97	97	98	96	97	96	97	96	97	96	97	96	97
704	88	89	87	88	89	90	89	90	88	86	86	86	89	88	88	87	88	89	87	88	89	90	89	90	88	86	87	86	89	88	88	87
736	91	92	91	91	91	92	90	91	91	90	90	89	91	89	90	88	91	92	91	92	91	92	90	91	91	90	90	89	91	89	90	88
768	75	76	76	77	75	75	76	76	74	75	75	76	74	75	75	76	78	79	78	78	78	79	77	78	78	79	77	78	77	78	77	78
800	72	73	73	74	71	71	71	72	71	73	72	74	70	70	71	72	76	76	75	76	73	74	73	73	75	76	74	75	73	73	72	73
832	72	72	73	74	71	72	72	74	71	70	72	71	71	70	72	71	75	76	75	76	75	76	75	76	75	74	75	73	75	74	74	73
864	69	70	70	71	67	68	68	69	69	67	69	68	66	65	68	66	73	74	72	73	71	72	70	71	72	71	71	71	70	69	69	68
896	99	99	98	98	98	99	97	98	98	99	97	97	97	98	96	97	98	99	97	98	98	99	97	98	97	98	96	97	96	98	96	97
928	95	96	95	95	93	94	92	93	95	96	94	95	92	93	92	93	95	96	94	95	93	94	92	93	94	95	94	95	92	93	92	93
960	92	93	91	92	91	92	91	92	92	90	91	90	92	90	91	89	92	93	91	92	91	92	91	92	92	90	91	90	91	90	91	89
992	89	90	89	89	87	88	87	88	89	87	88	87	87	86	86	85	89	90	88	90	88	88	87	88	89	87	88	87	87	85	86	84
1024	87	88	86	87	87	90	87	89	88	89	87	88	89	90	88	89	86	88	85	87	87	89	86	89	87	88	86	87	88	90	88	89
1056	90	92	89	91	89	91	89	90	91	92	91	92	91	92	90	91	89	91	88	90	89	91	88	90	91	92	90	91	90	91	90	91
1088	95	96	95	95	96	97	96	97	96	96	95	95	97	97	97	96	95	96	94	95	96	97	96	96	95	96	95	95	97	97	97	96
1120	98	99	98	99	98	99	97	98	100	99	99	99	99	99	98	98	98	99	97	98	98	99	97	98	99	99	98	98	99	98	98	97
1152	71	73	71	73	73	74	72	74	73	74	72	74	74	75	74	74	69	71	69	70	70	72	69	71	71	72	70	71	72	73	71	72
1184	75	77	74	76	75	76	74	76	76	77	75	77	76	77	76	76	72	74	71	73	72	74	71	73	74	75	73	74	74	75	73	74
1216	76	77	76	76	78	79	76	78	77	77	76	76	78	78	78	77	74	75	73	74	75	76	75	75	75	75	74	73	76	76	75	74
1248	80	80	79	80	79	81	79	80	81	81	80	80	81	81	79	80	77	78	77	78	77	78	77	78	79	79	77	77	78	78	77	76
1280	90	92	90	92	90	92	89	91	92	93	91	92	91	92	91	91	89	92	89	91	89	92	89	91	91	92	90	92	91	92	90	91
1312	87	89	87	89	86	87	85	86	89	90	88	90	87	88	86	87	87	89	86	88	85	87	85	86	88	89	87	89	87	88	86	87
1344	99	100	98	100	99	100	98	99	100	100	99	99	100	100	99	99	99	99	98	99	98	99	98	99	100	100	99	99	99	99	99	98
1376	96	97	95	96	94	95	93	94	97	97	96	96	95	95	94	94	96	97	95	96	94	95	93	94	97	97	96	96	95	95	94	94
1408	75	77	74	77	75	77	74	76	77	78	76	77	76	77	76	77	73	75	72	74	73	74	72	74	75	76	74	75	74	75	74	74
1440	72	75	72	74	71	72	70	72	74	75	74	74	72	73	72	72	70	72	69	71	68	70	67	70	72	72	71	72	70	71	69	70
1472	80	81	80	80	80	81	79	80	81	81	79	80	80	80	80	79	78	79	77	78	77	79	77	78	79	79	77	77	78	78	77	77
1504	77	78	77	78	75	76	75	76	79	78	77	77	77	76	76	75	75	76	75	75	73	74	72	73	76	76	75	75	74	74	73	72
1536	94	95	94	94	95	96	94	95	94	94	93	94	95	96	94	95	94	95	93	94	95	96	94	95	93	94	92	93	94	95	94	95

IPM Examples

NERSC job details

http://www.nersc.gov/REST/jobs/mpi_functions.php?stepid=441560.sdb&f_name=MPI_Recv&timestamp=1306769707

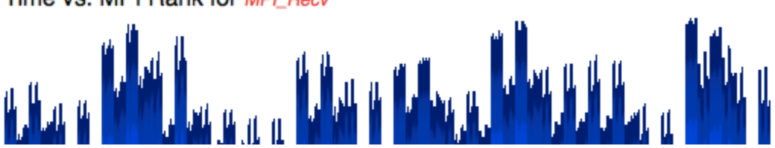
Time spent by each task in *MPI_Recv* as a percentage of the maximum value

The MPI rank represented by each cell in the table is the sum of the cell's column and row indices.

Table Columns: 32

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	74	76	69	70	77	77	70	71	60	62	60	61	64	65	65	64	78	79	72	72	78	79	73	73	63	64	63	64	66	65	66	66
32	70	71	63	65	72	72	65	66	52	54	53	55	56	57	58	58	71	73	67	68	72	73	67	69	55	55	54	55	58	58	57	57
64	91	92	88	90	91	93	90	91	76	77	74	76	79	81	79	79	95	98	92	96	98	98	96	96	78	81	77	80	84	84	83	82
96	86	87	80	80	86	89	81	82	62	63	61	61	65	67	65	65	91	94	83	86	94	94	87	86	64	66	63	65	70	69	69	68
128	69	71	64	65	70	72	65	66	54	56	53	55	60	60	59	59	69	70	64	65	69	70	65	66	56	57	55	56	60	59	58	58
160	65	67	57	59	67	68	60	61	42	43	43	44	47	46	47	47	66	67	58	60	67	67	60	60	43	44	45	46	46	46	48	48
192	86	87	80	81	88	89	82	83	67	68	66	67	71	72	70	70	85	88	81	84	89	88	84	84	68	71	66	69	74	73	72	71
224	79	79	71	71	80	80	72	72	49	49	49	50	52	52	53	52	76	79	70	73	79	79	73	74	50	52	48	52	55	55	54	54
256	83	84	81	83	85	85	84	85	69	71	69	70	74	75	74	74	85	86	85	87	86	87	86	87	73	73	71	71	76	75	74	73
288	72	73	66	68	73	74	69	70	60	62	59	60	64	65	63	63	75	76	69	71	76	76	71	71	63	63	61	61	65	65	64	64
320	94	95	93	94	93	96	94	95	77	78	75	75	80	81	78	78	99	99	96	96	99	99	98	97	79	80	77	78	83	82	81	80
352	82	83	74	75	83	84	75	76	61	62	66	66	65	65	69	69	84	85	76	77	85	85	77	78	63	64	66	67	66	66	69	69
384	84	86	76	78	86	88	79	80	64	64	63	64	68	68	67	67	85	86	77	78	85	86	78	79	65	65	66	66	68	67	69	68
416	69	70	60	62	71	72	63	64	45	46	46	47	49	50	50	51	68	69	61	62	69	70	62	62	47	47	46	47	49	49	47	49
448	98	98	99	100	99	98	100	100	85	85	78	77	89	89	81	80	94	94	95	96	94	94	97	97	87	87	78	79	91	90	82	81
480	82	82	72	73	82	84	74	74	54	54	52	52	56	57	54	55	83	83	72	73	84	84	71	73	54	55	53	54	57	57	55	56
512	88	89	89	91	90	91	92	93	78	80	78	80	83	84	84	84	93	94	93	95	94	94	95	94	80	81	82	83	85	84	86	85
544	90	91	79	81	91	93	82	83	63	65	63	64	67	69	68	68	96	97	84	84	98	97	85	85	66	66	66	67	69	68	70	69
576	76	77	74	75	76	77	76	76	65	65	66	67	67	67	70	70	79	82	77	80	83	82	81	81	67	69	69	71	72	72	74	73
608	76	77	69	69	76	78	70	71	57	57	57	58	60	59	60	60	79	81	72	75	82	81	75	75	60	62	58	61	64	63	63	62
640	92	94	85	87	96	96	90	89	75	77	75	77	81	81	81	81	93	94	87	88	94	94	88	88	77	78	76	77	81	80	80	79
672	86	87	78	78	88	88	80	80	57	58	58	59	62	62	63	63	87	87	78	79	88	87	79	79	59	59	60	60	62	61	63	63
704	78	79	74	75	80	80	76	75	63	64	63	63	68	67	67	66	78	81	74	77	81	81	77	77	64	67	63	66	70	69	69	67
736	70	71	65	65	71	71	66	67	52	52	50	51	55	54	53	53	71	73	64	67	74	73	68	68	53	55	52	54	57	57	57	56
768	94	96	87	88	96	98	90	90	71	73	70	71	75	77	75	75	95	95	87	87	96	96	87	88	74	75	73	74	79	78	77	77
800	79	80	70	71	81	83	72	73	57	58	62	63	62	62	67	67	80	79	71	71	80	80	71	72	60	60	64	64	63	63	66	67
832	82	83	77	77	82	84	78	78	63	64	64	64	66	66	67	66	86	86	79	80	87	86	81	80	65	66	64	65	69	68	68	67
864	69	70	65	65	70	70	65	66	56	56	56	56	59	59	58	58	73	73	67	67	73	73	67	67	58	59	58	59	61	59	61	59
896	92	92	85	86	93	94	88	88	71	72	69	70	76	75	74	74	92	92	87	88	94	93	88	89	73	73	70	71	76	76	74	74
928	73	74	64	65	75	76	66	67	43	44	48	49	47	48	53	54	74	73	64	65	73	74	64	66	45	45	51	52	47	48	53	54
960	74	74	70	70	75	75	71	71	63	64	61	61	66	66	64	63	76	77	72	73	77	77	73	73	63	64	62	62	66	66	66	64
992	63	63	57	57	63	64	58	58	41	41	42	42	43	42	44	43	61	61	56	57	62	62	56	57	41	41	42	42	43	42	43	43

Time vs. MPI Rank for *MPI_Recv*



Questions to You

- What tools do you use?
- What tools do you want?
- What would you like centers to support?
- Can you get to exascale without tools?

Users Want (Need?) Tools

- Users are asking for tools because HPC systems and programming models are changing
- More and more components to worry about
  - CPU (caches, FPUs, pipelining, …)
  - Data movement to main memory, GPU memory, levels of cache
  - I/O
  - Network (message passing)
  - CPU Threads (OpenMP)
  - GPU performance

What I Want in a Tool

- Let the users help themselves
- Work for everyone all (most of?) the time
- Easy to use
- Useful
- Easy to interpret the results
- Affordable ($$ or manpower support costs)
- Simple, supplement existing complex tools
  - Point the way for a "deeper dive" in problem areas

NERSC

National Energy Research
Scientific Computing Center