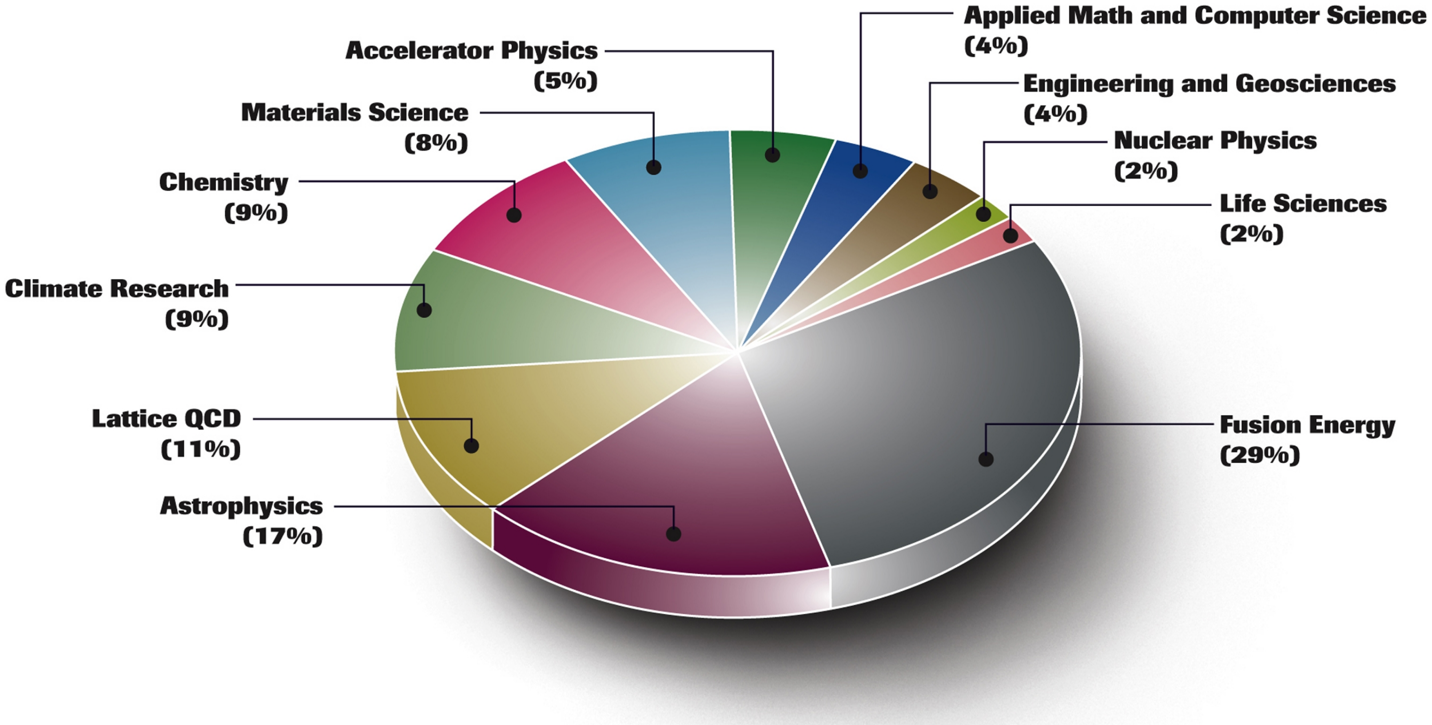
Application Performance Tools @ NERSC

David Skinner, Richard Gerber,
Nick Wright, Karl Fuerlinger
and 4000 others

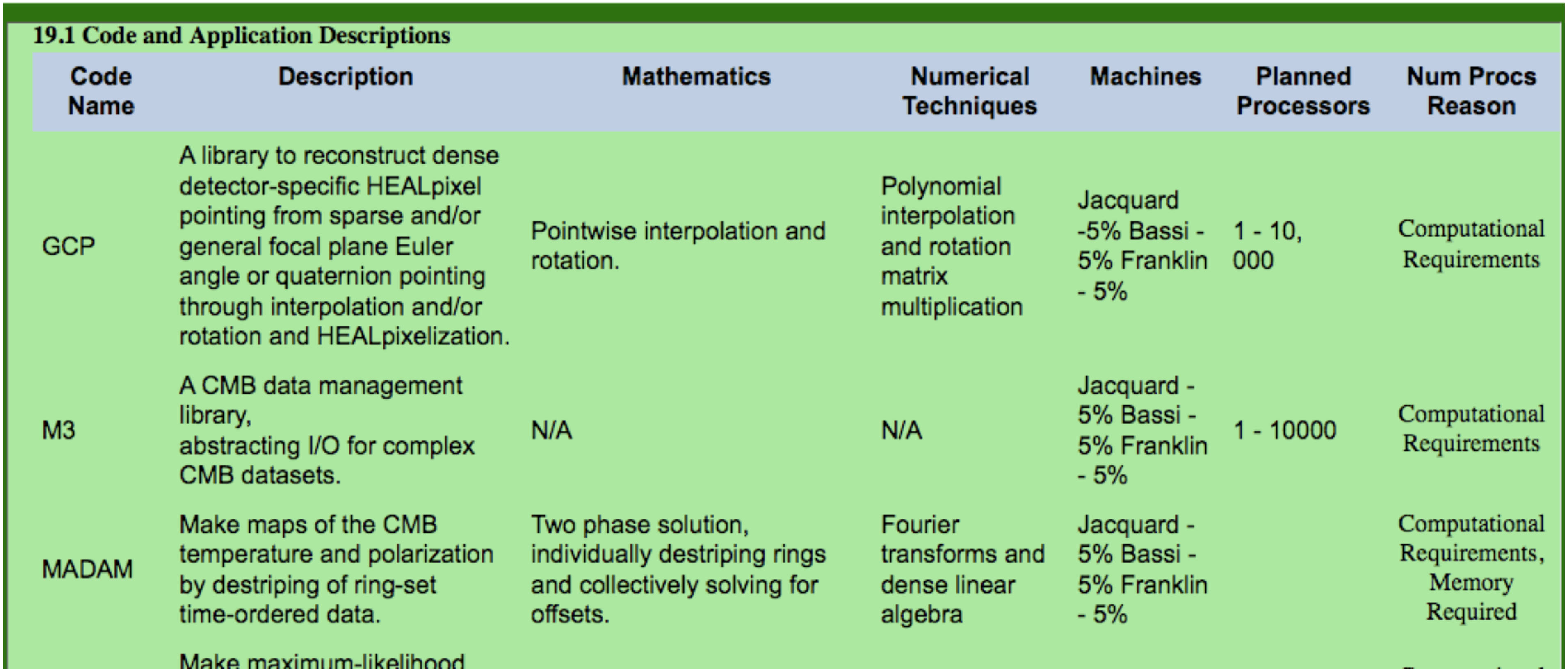
User demographics at NERSC



- •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

ERCAP Question 19.1

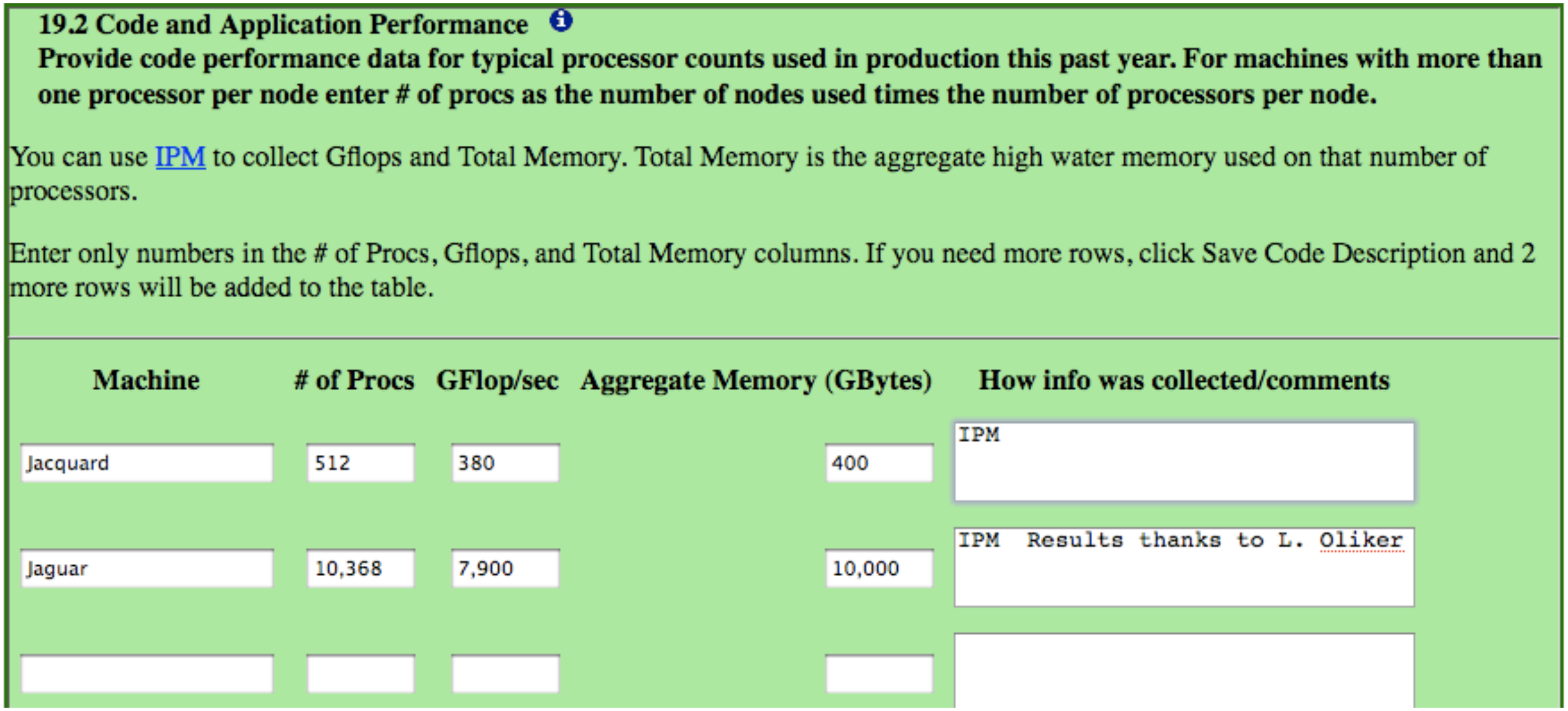
Each application for time at NERSC includes both algorithmic and performance assessments



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.

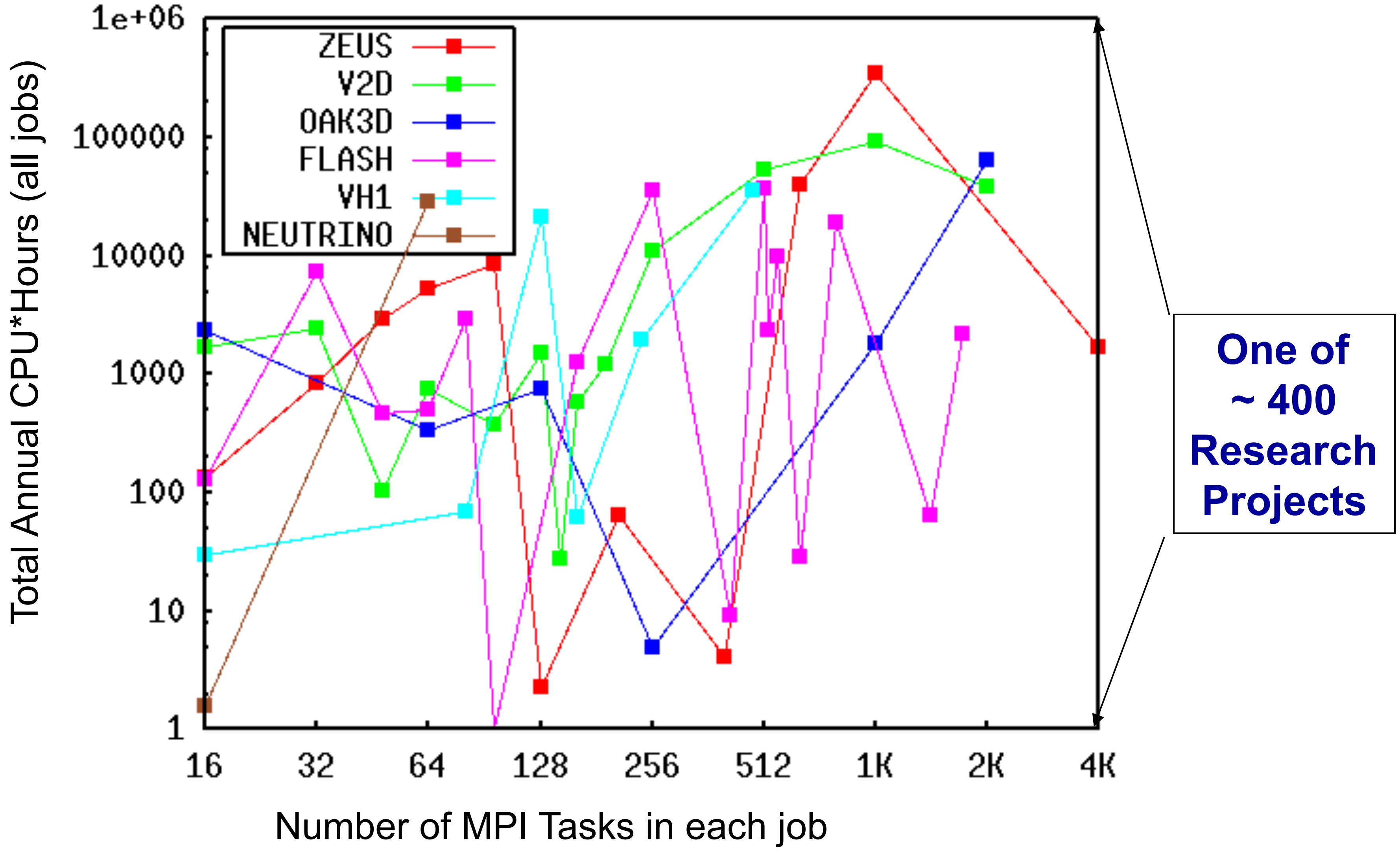
Enter only numbers in the # of Procs, Gflops, and Total Memory columns. If you need more rows, click Save Code Description and 2 more rows will be added to the table.

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

Core needs in Production HPC Tools

- 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?
- How do you even know when to tune?

NERSC has many Customers and an Extremely Diverse Workload



One of ~ 400 Research Projects

Back up, what is a performance tool?

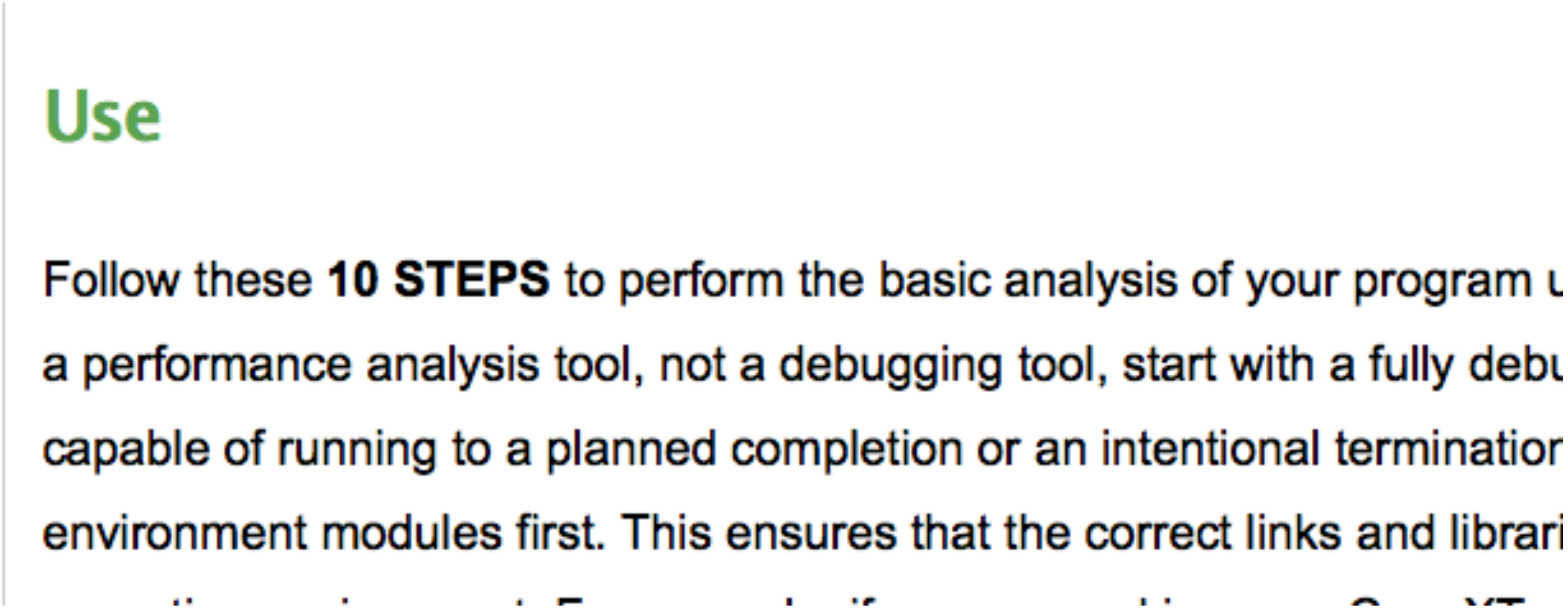
1. An application that users can run to debug the performance of their code (is this what the center wants?)
2. A runtime layer implemented by the center staff that reports on application performance (is this what the user wants?)

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.

Ease of use == It gets used

Example from NERSC web docs

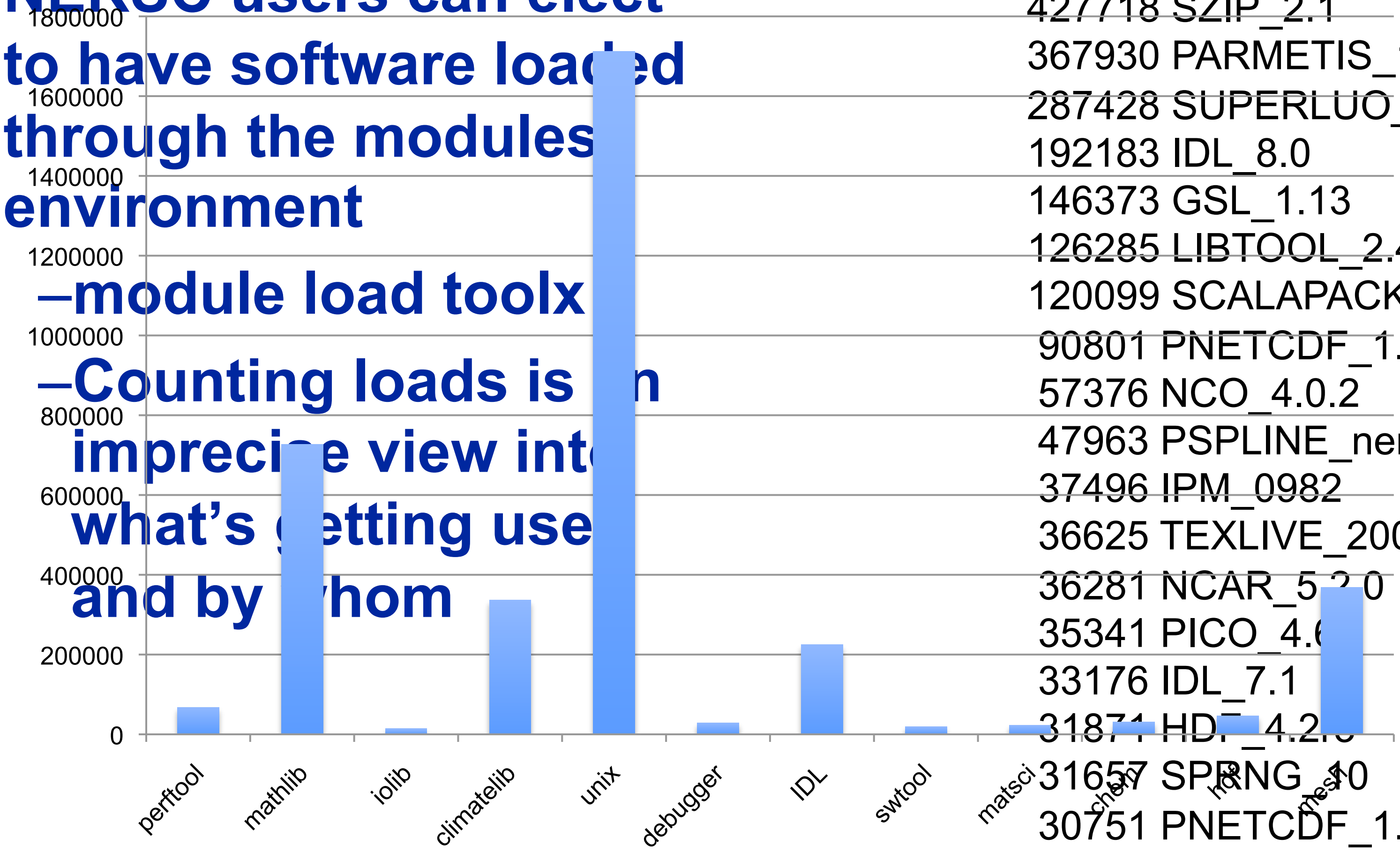


Use

Follow these **10 STEPS** to perform the basic analysis of your program u a performance analysis tool, not a debugging tool, start with a fully debu capable of running to a planned completion or an intentional terminatior environment modules first. This ensures that the correct links and librari

Are users reaching for tools?

- NERSC users can elect to have software loaded through the modules environment
  - module load toolx
  - Counting loads is an imprecise view into what's getting used and by whom



1035550	PYTHON_2.6.2
427718	SZIP_2.1
367930	PARMETIS_12
287428	SUPERLUO_DIST_20
192183	IDL_8.0
146373	GSL_1.13
126285	LIBTOOL_2.4
120099	SCALAPACK_180
90801	PNETCDF_1.0.3
57376	NCO_4.0.2
47963	PSPLINE_nersc1.0
37496	IPM_0982
36625	TEXLIVE_2008
36281	NCAR_5.2.0
35341	PICO_4.6
33176	IDL_7.1
31871	HDF_4.2.6
31657	SPRNG_40
30751	PNETCDF_1.1.0
30385	TAU_2.20.2
29473	DFFTPACK
28962	DDT_2.6
28299	PETSC_233-opkgs_O

What NERSC users say

- “We are involved in multiple studies to assess performance limitations, and often benefit from NERSC performance tools especially IPM and IPM-I/O profiling”
- “We have been using a number of performance analysis tools available at NERSC (IPM, CrayPat, PAPI) to improve the performance of the code.”
- “…gets ~12-15% nodal performance on Cray XT5 based on profiling with Tau, CrayPAT, and other performance monitoring tools.”
- “Our primary profiling tools are timing routines which are internal…”
- “Memory scalability can benefit from NERSC parallel profiling tools.”

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?

Which tool helps to answer Question 19?

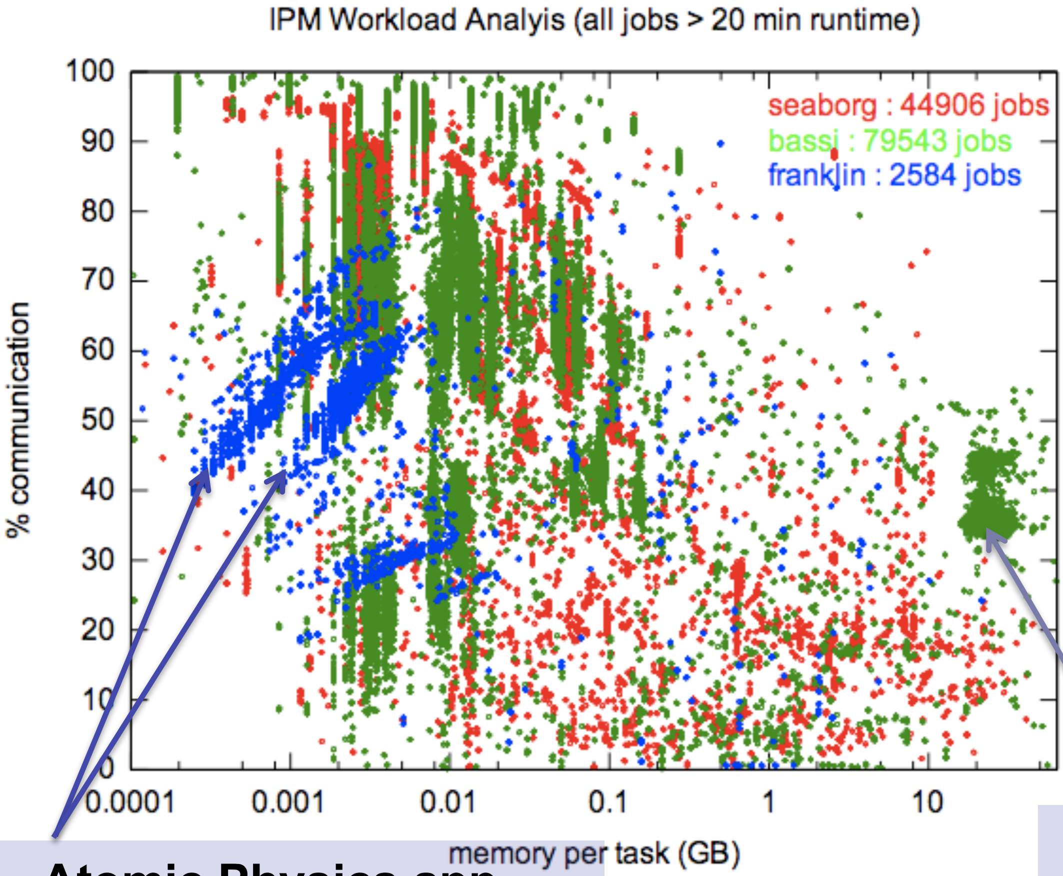
- HPC centers with complex & dynamic workloads need an easy to use, almost transparent, low impact profiling layer that provides high level summaries about job performance.
- More in-depth & detailed tools can be used subsequently. Use the right tool for the job.

Profiling Tools Gotchas (what not to do)

- 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

State of the practice at NERSC in performance analysis

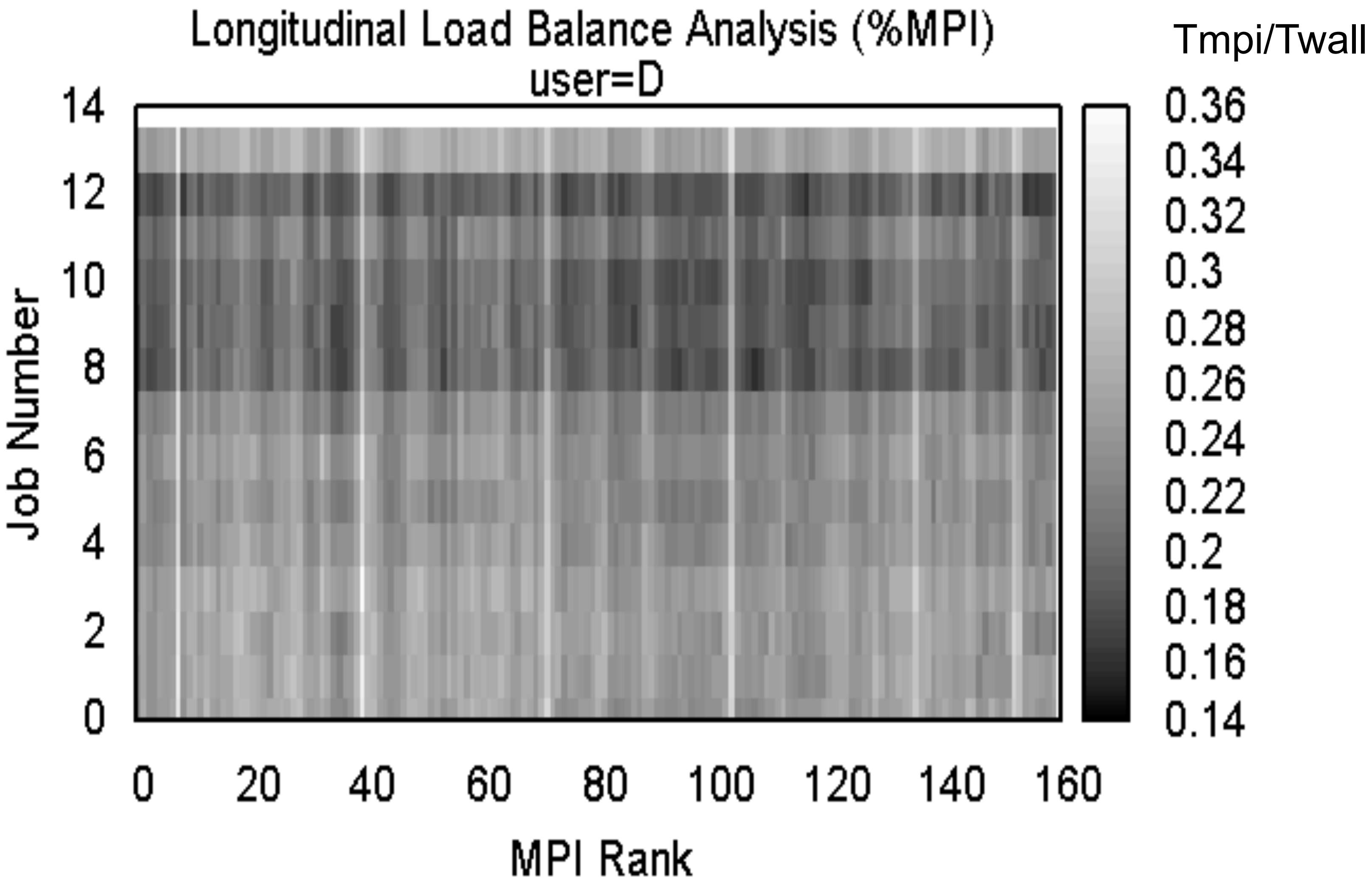
300K IPM Application Profiles



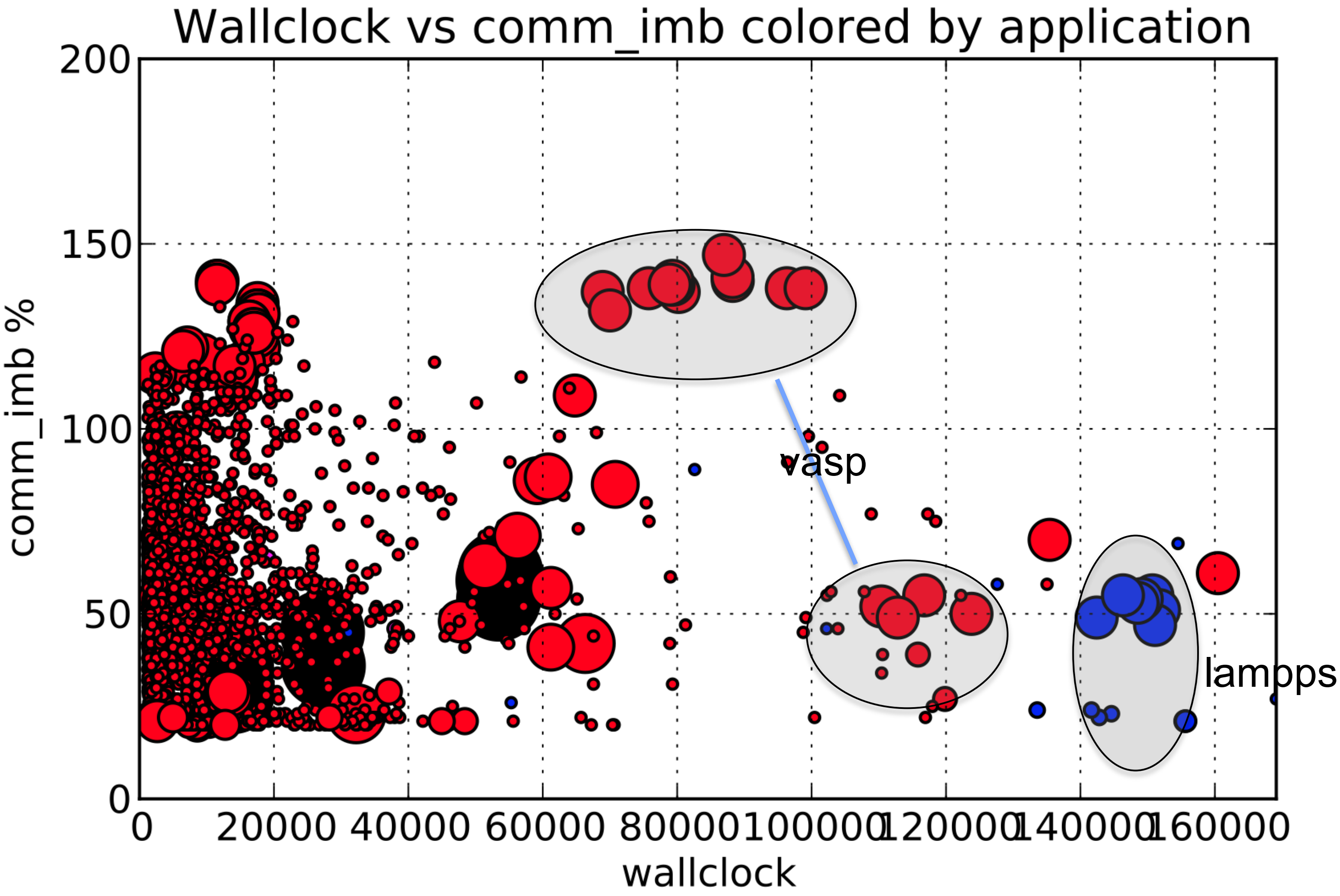
Atomic Physics app

QCD app

Performance trending in workloads

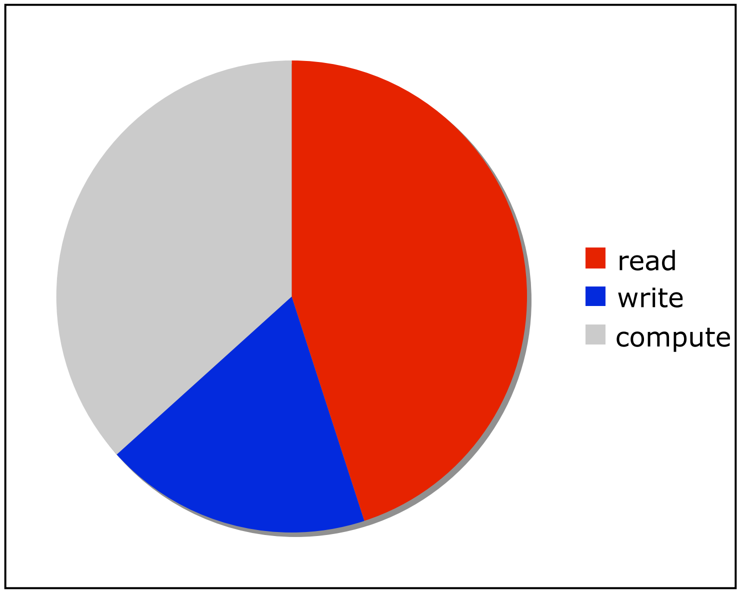


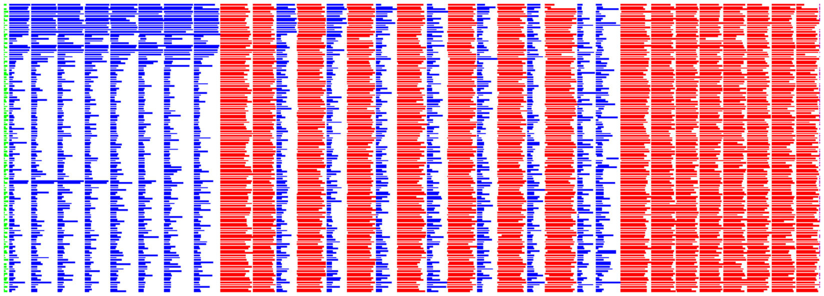
Imbalanced apps vs walltime



Rising interest in figuring out IO

Based on trends in trouble tickets and discussions with users
IO is now officially a big deal





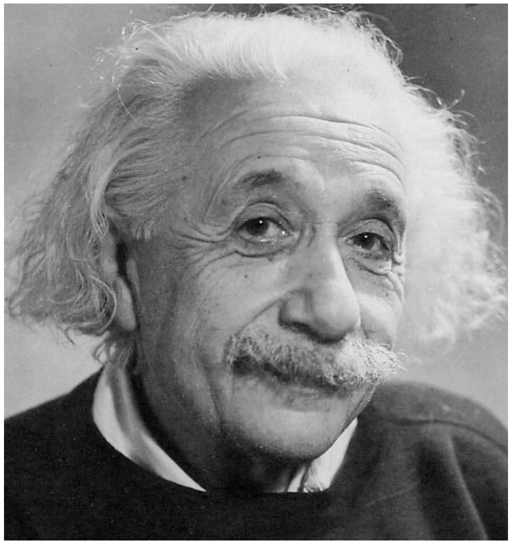
Performance Tools at Exascale

- **The general state of performance "awareness" has declined markedly overthe last ten years**
  - Exploding concurrencies
  - Multicore contention
  - Multicore counters < Pentium counters
  - Deeper memory hierachies
  - Memory touch policies
- **At Exascale how will we at least tread water?**
  - Something will be broken in a performance sense 100% of the time
  - Monitor at multiple levels (often) to corroborate
  - Need foundational software to inform tools (PAPI for everything)
- **Keep focused on users**
  - Performance in principle < performance in practice

Performance is Relative

- **To your goals**
  - Time to solution, $T_{queue}+T_{run}$
  - Efficient use of allocation
  - Do FLOPs even matter?

- **To the**
  - application code
  - input deck
  - machine type/state



No Nobel Prize in FLOPS