Magellan workloads

Use of IPM to measure, predict and fix performance maladies

Profiling Magellan workloads

- Can use IPM to monitor individual apps
- But can we use it to monitor machine wide runs?
- Load imbalance amenable to analysis at a broad level using IPM profiles (significant performance issue in many codes)
- Detecting/fixing performance could save users time and put our computing dollars to better use
- All magellan jobs instrumented with IPM; ongoing efforts to do same for carver

Profiling Magellan workloads

- Broad approach
  - Profile workload (superset)
  - Identify 'interesting' areas (need to find ways of representing data)
  - Zoom into regions of interest (subsets)
    - Are a particular user's runs load imbalanced (or any other interesting questions)
    - Do we see patterns for any particular application
    - What is the effect of concurrency
    - Are there any culpable MPI calls
  - Alert user to do more detailed performance fixes
  - Note: only identifying plausibilities. We do not make claim that broad analysis gives truth; should motivate to dig deeper.

Putting a cost dimension to load imbalance

- How it might all matter in real terms
  - Hours lost to ‘useless’ computing
  - Pecuniary considerations ($$)
  - Power issues (how much power is wasted)

Procedure

- Analyze logfiles in magellan produced by IPM
- Infrastructure: Parse with ruby (mostly), python (matplotlib – plotting software), perl (misc), black magic awk scripting by David
- Store parsed data into mongodb (a non-relational database, meaning which, the database can have variable shape)
- Retrieve data from database (using parser scripts)
- Do analysis

Sample IPM output to be parsed

- #IPMv0.921##
- #
- # command : gvasp (unknown)
- # host : c1060 mpi_tasks : 24 on 3 nodes
- # start : 08/18/11/18:59:05 wallclock : 46.996800 sec
- # stop : 08/18/11/18:59:52 %comm : 42.70
- # gbytes : 7.69464e+00 total gflop/sec : 3.10956e-01 total

- # [time] [calls] <%mpi> <%wall>
- x # MPI_Alltoall 404.706 278898 6.60 10.68
- x # MPI_Allreduce 339.132 293750 5.58 8.15
- x # MPI_Alltoallv 197.134 144008 3.37 5.01
- x # write 57.6631 747624 0.49 2.79
- x # MPI_Bcast 15.9053 90005 0.19 0.36
- x # MPI_Barrier 6.47102 3807 0.11 0.16

Load imbalances

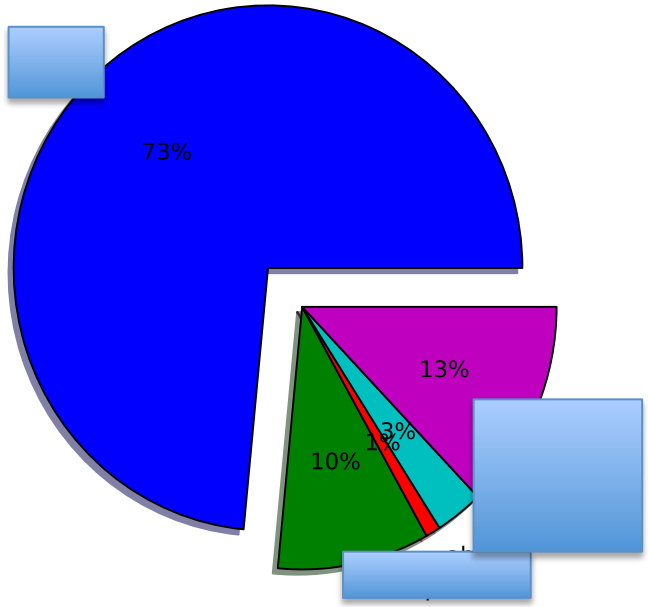
- Define metric as
  - % Imb = (max(metric) – avg (metric))/avg(metric)

$$\sigma = \frac{w_{max} - \bar{w}}{\bar{w}}$$

- Here metric can be
  - Wallclock time (turns out to be irrelevant because tasks sync after MPI_Finalize)
  - Communication time (‘interesting’)
  - IO time (irrelevant)

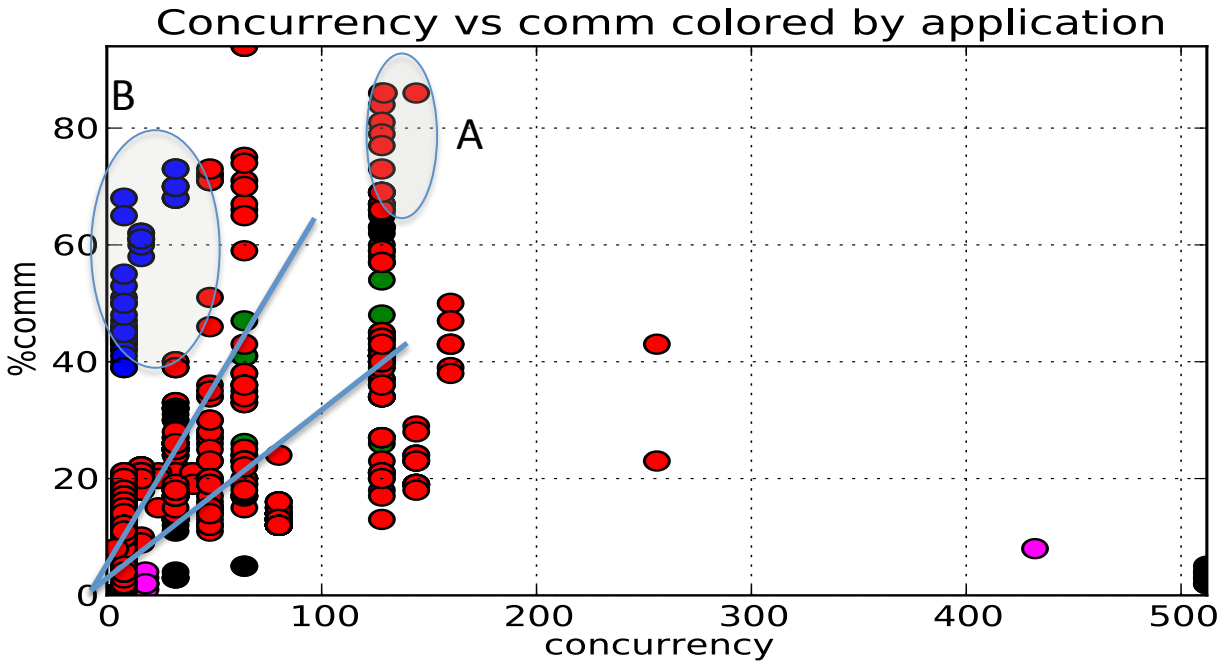
Composition of jobs

Month of June - ~50 k runs



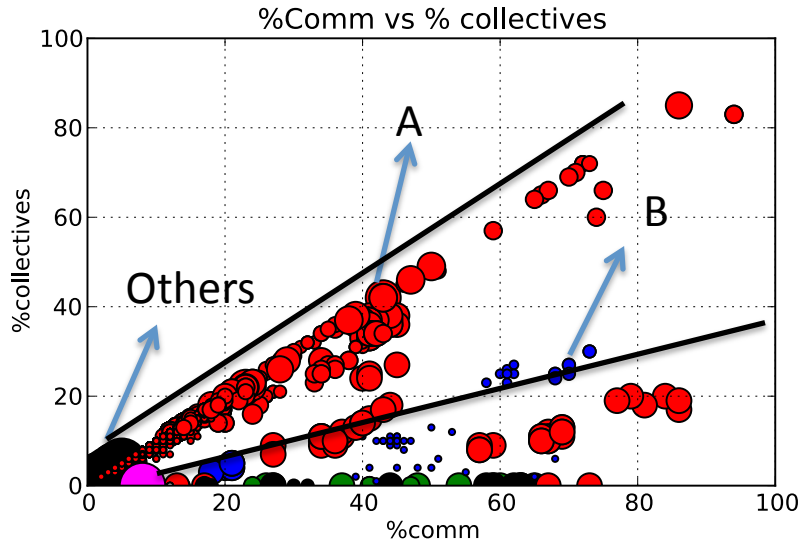
- A: 75 % of workload by wall
- Count: 45,000 files
- For a particular month – June 2011

Concurrency vs comm



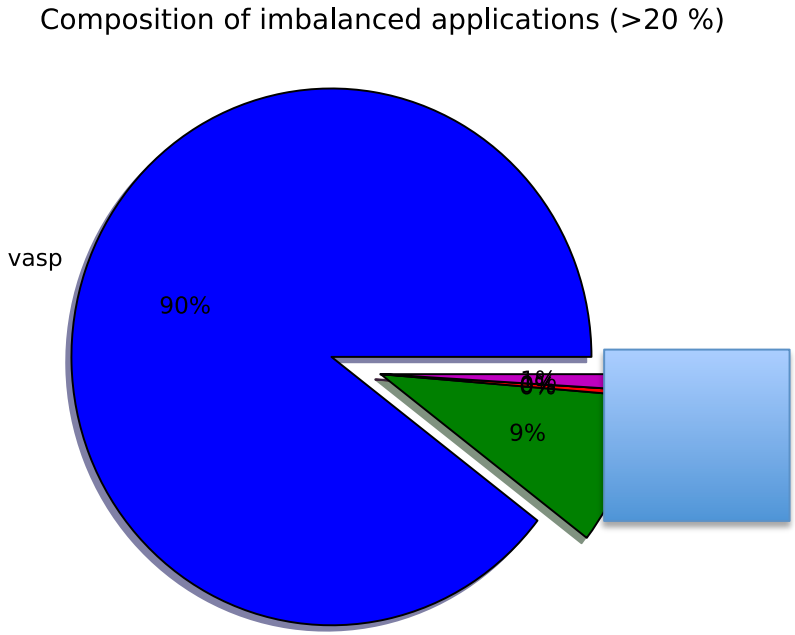
Collectives

- Allreduce+Alltoall summed up
- Suggests that
  - Imb(comm)~imb (collectives)



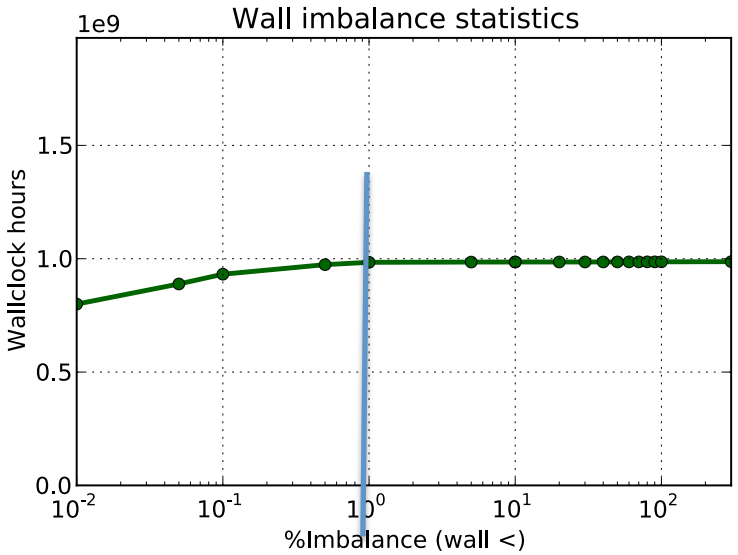
Imbalanced apps

- Filter with imbalances>20 %
- Vasp becomes even greater (why?)

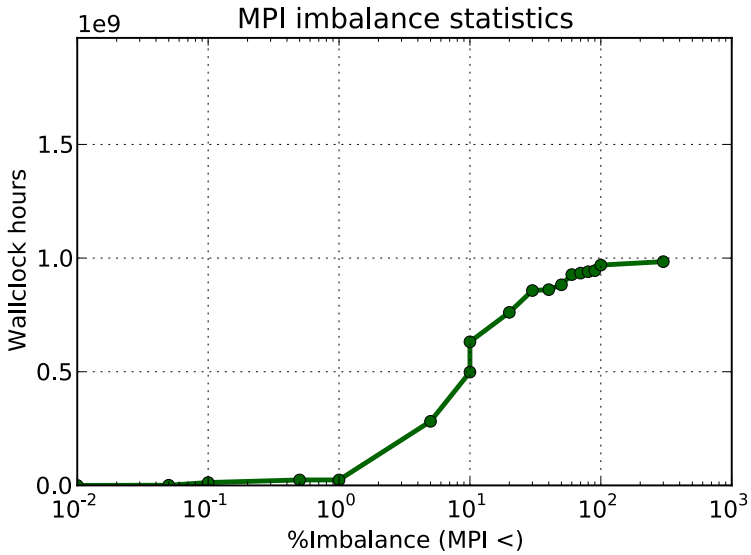


Why we wall imbalances are useless (from Aug data)

Wall imbalance (cummulative)

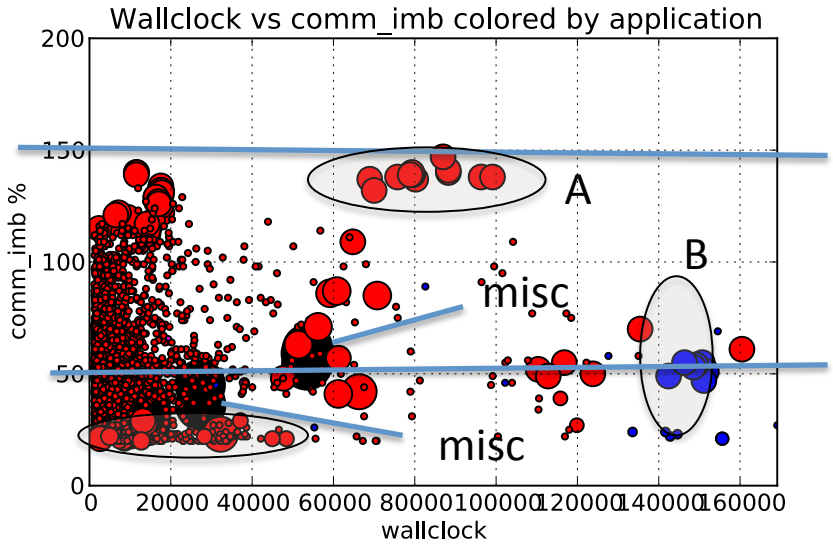


Comm imbalance (cummulative)



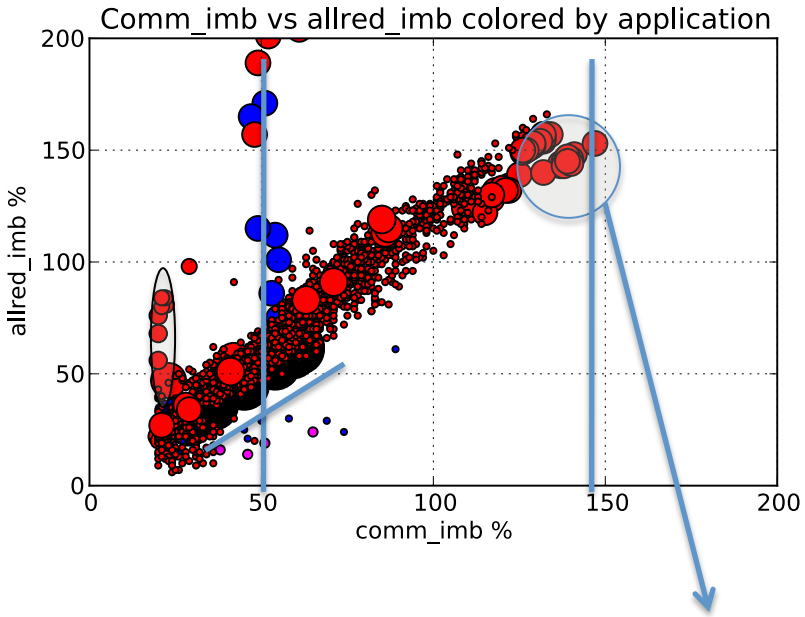
Imbalanced apps vs walltime

- Isolate groups for imbalances, based on application and concurrency
- Colored by application
- Size by concurrency



Correlating with allreduces

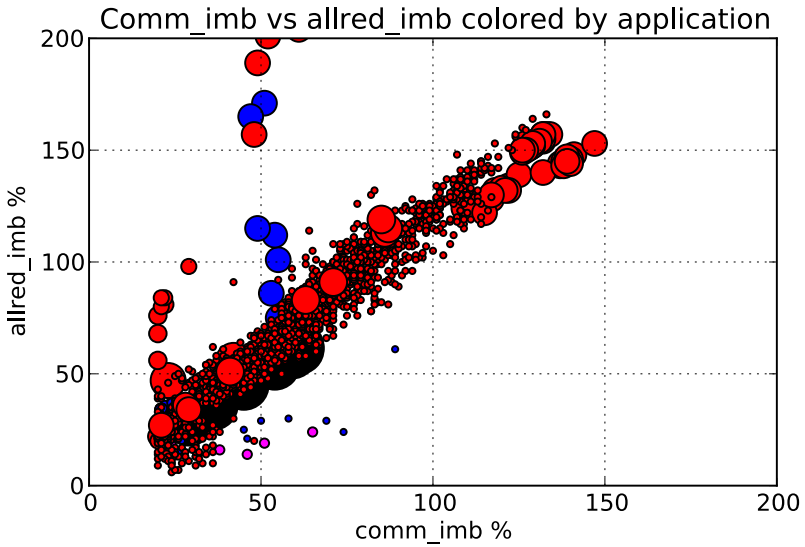
- Can correlate imbalances observed with allreduces imbalance for four of the obvious clusters
- Scatter plot shows a fairly proper y=x trend
  - Communication imbalances mostly arise from imbalances in corresponding allreduce
  - Important note: vasp is dominated by 3D FFTs, which are known to be influenced by Allreduce and Alltoallv

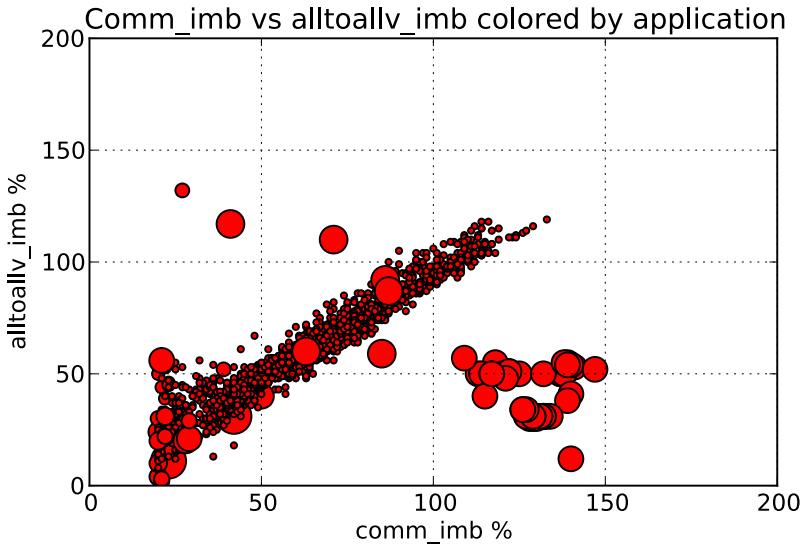


Isolate and contact users

Correlating with alltoallv

Can we come up with some machine learning so that we know, for example that we have an attribute A for an application X, we know that it is because of B?



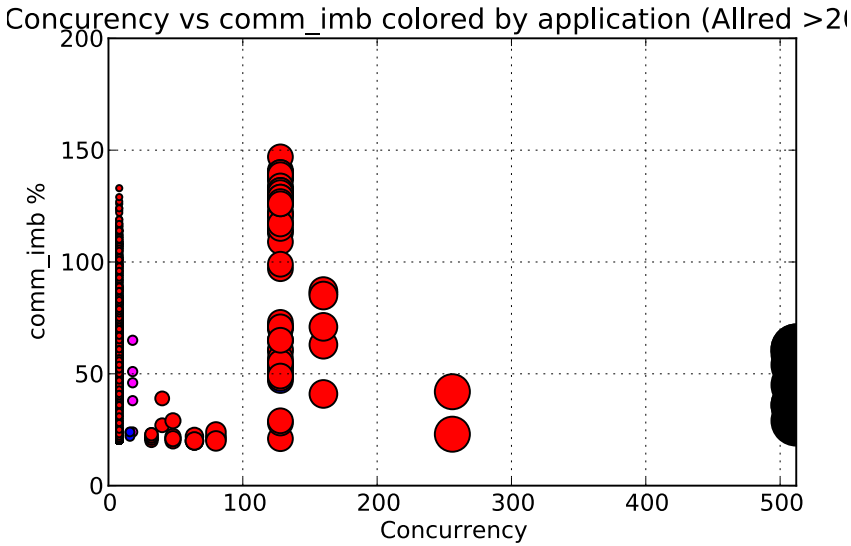


Alltoallv correlations picked up for vasp alone (expected)

Some points not correlated by allreduce Picked up by alltoallv

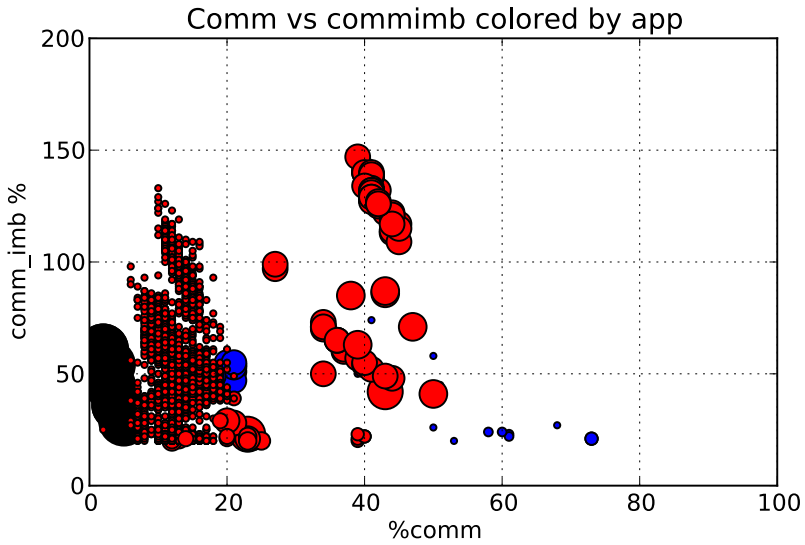
What does it look like in concurrency space

- (for imbalanced apps)
- This shows user and application behaviour
  - User: user runs at the same concurrency (say, 128)
  - But the type of run (metals or non-metal chemistry, etc.) determines communication (and imbalance) pattern.



How do comm imbalances correlate with % comm?

- Again, a separation emerges
- Shows that for a given level of %comm, we may have large variations in imbalance depending on run
- Also notice that concurrency seems to be ‘discrete’
  - Some large blobs
  - Numerous small blobs
  - Shows user behavior



In terms of cost metrics

- Hours wasted

$$\sigma = \frac{w_{max} - \bar{w}}{\bar{w}}$$

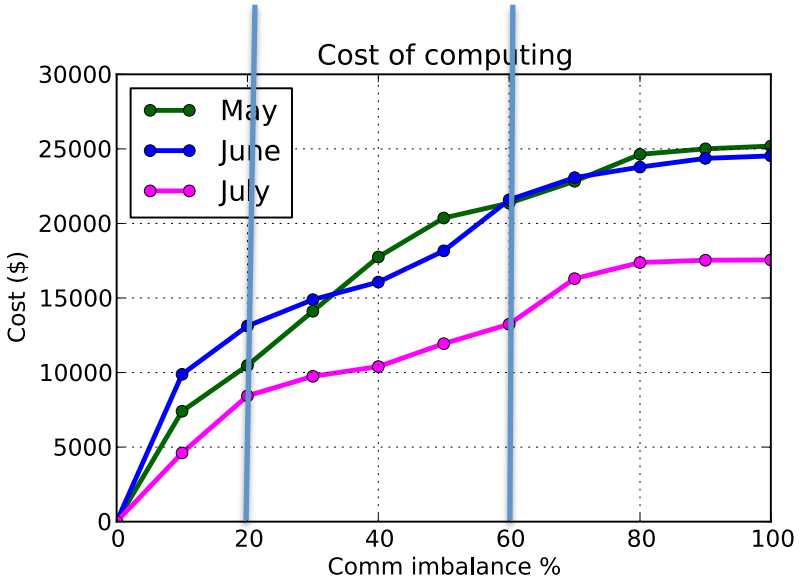
$$H = \frac{w_{max} - \bar{w}}{\bar{w}} * (N * \bar{w})$$

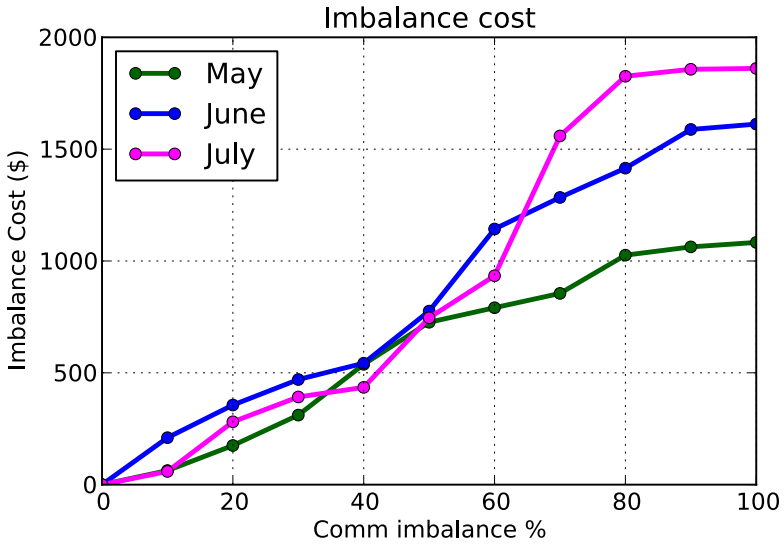
Dollar value of cost

- NERSC budget: 50 M per year
- Computing hours: 10^9 /yrs
- Cost per hr of computing ~ 5 cents/hr
- How many $$ are being lost to imbalanced computation?

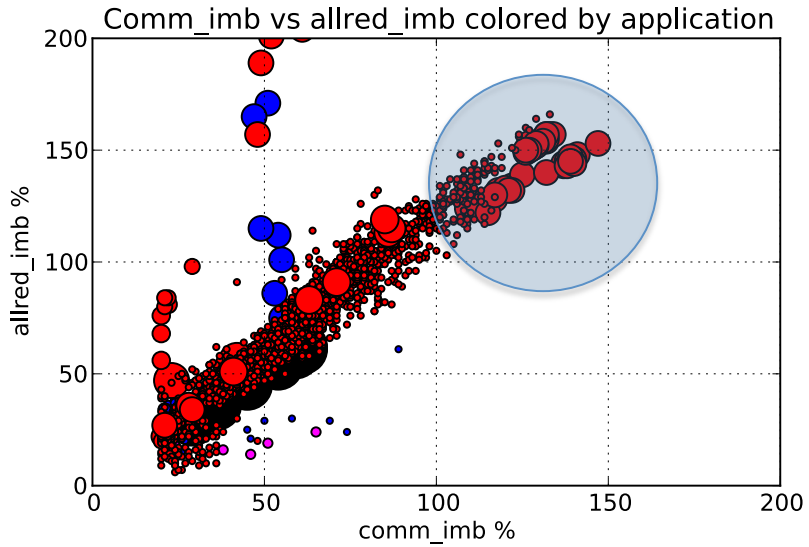
$$\$ = \frac{w_{max} - \bar{w}}{\bar{w}} * (N * \bar{w}) * C_w$$
$$= \sigma * (N * \bar{w}) * C_w$$

Cost metrics

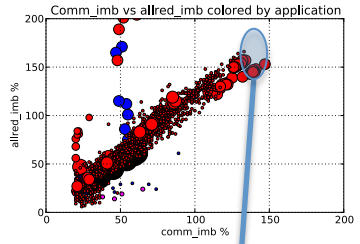


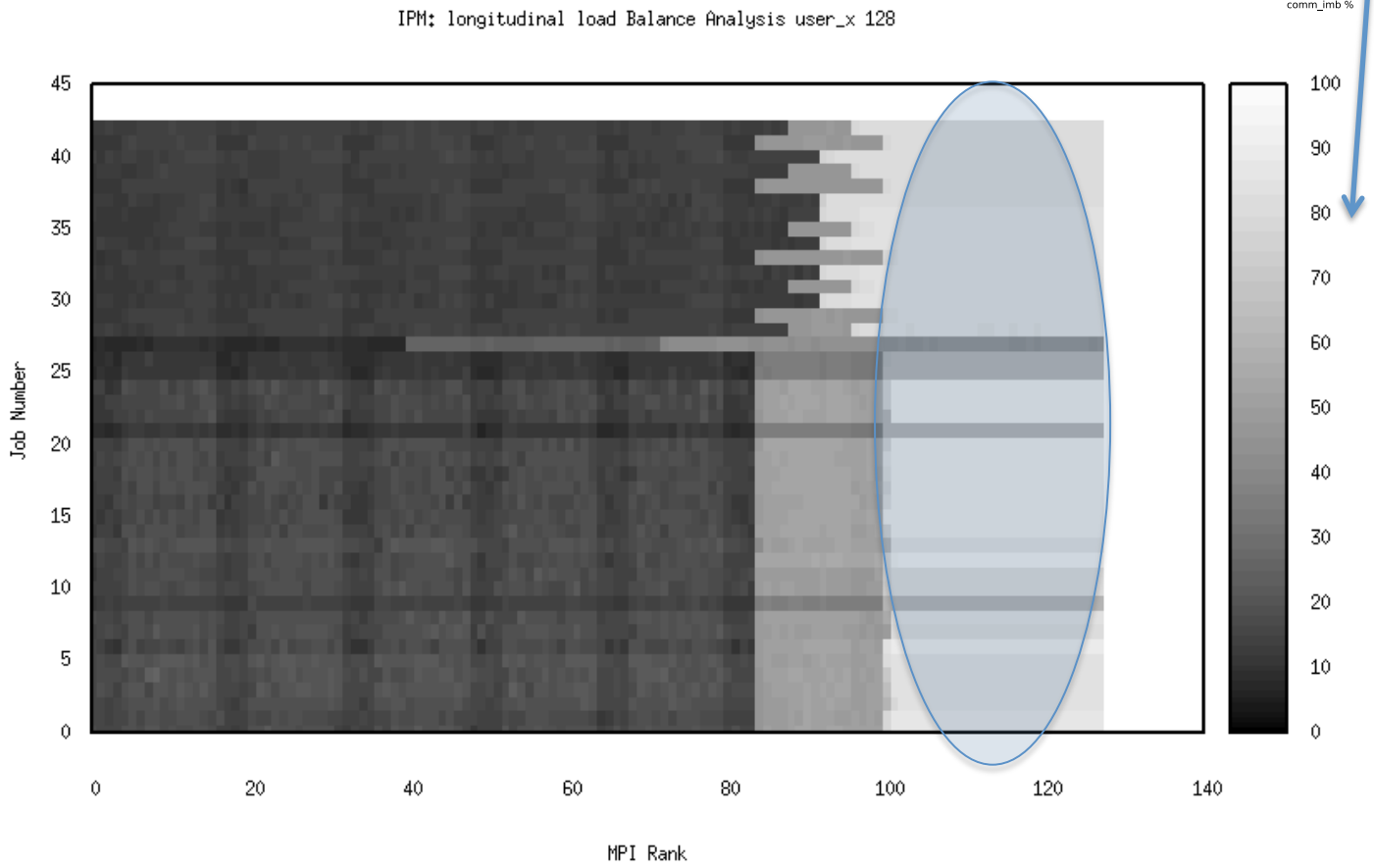


Individual appruns



User x –idle cores





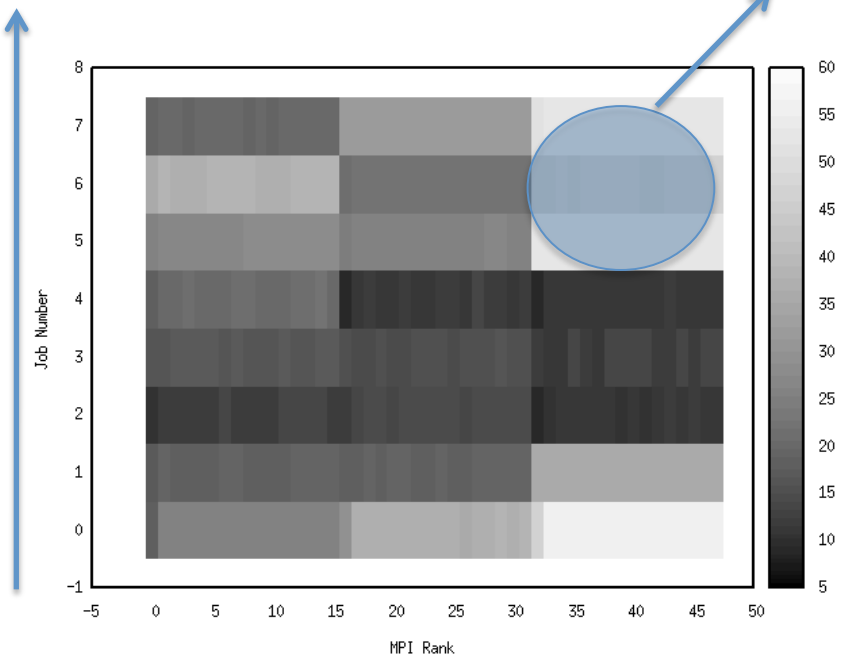
Same user, two concurrencies

Need to check: is this the same job, same parameters?

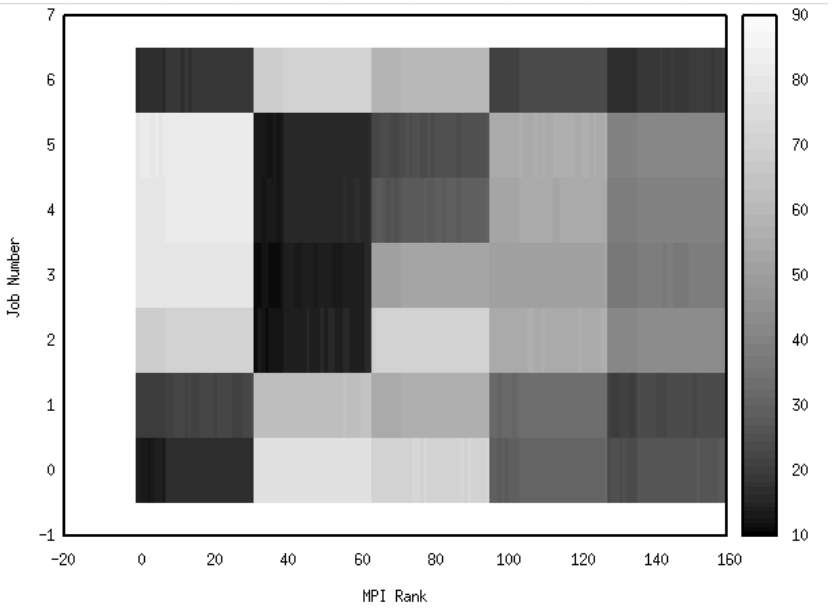
Users can run different set of conditions … (eg, different elements in A)

Machine weather

Processes not doing much computing!



48



160

Summary

- By virtue of being purveyor of supercomputing services, NERSC has access to data that we would never have from individual runs (and IPM gives us such info)
- Tentative load imbalance analysis carried out
  - Gives basic info such as user, concurrency
  - Attempts to correlate communication imbalance with underlying MPI imbalance
- Individual user runs can have considerable variability, depending on machine conditions and user input
- Possibly starker implications in more weighty runs (carver, hopper, franklin, etc.)
- Next steps: to contact individual users to initiate process