Hooks into runtime library to feedback

Limit threads when the memory controller gets overloaded

Kicking off batch jobs when they interfere with foreground jobs (Google Example)

If see a lot of cache misses, lower clock frequency to improve overall power/performance tradeoff

Can be a problem is all nodes do this at once (large datacenter power swings)

Varying precision, for power and/or performance

Varying the quality of the answer – don't run with more quality than the underlying data has

Turning on/off core to allow turbo boost on more cores

MPI barrier imbalance - to give more power to consistently late core, or slow consistently early ones

Using data to improve NUMA behavior

Permuting MPI ranks to align physical and logical topology (physical topo could change during execution due to failed nodes, or other jobs leaving)

Could be done per job, per time-step, or via checkpoint/restore

Develop behavior profiles for applications, and use to inform scheduling (time of day, level out variations)

Are there any synergistic opportunities – mix high memory job with high network job

All auto tuning ideas – switch algorithms based on performance observations

Resilience – map out non-functional nodes

Background Page scrubs to help with fault detection

Knowing topology holes caused faulty nodes

Can you reallocate nodes when other job finishes early, to improve a jobs topology

Using network counters to inform routing/remapping of physical & logical nodes

Do we have the right sensors in machines to measure power at the right granularity

Do we have the right ways to count data movement costs (net, CPU, memory)?

Cost of adaptation – too expensive to make the change (runtime cost of the transformation)

Using runtime data to control checkpoint intervals

Need check pointing of performance data so that restart has the context for the adaptation

- Performance data for one science run vs. one job launch
- Need to re-wire tool infrastructure when restarting on a new topology

Performance analysis on storage cluster (i.e. Lustre file system)

- Problems of shared resource (due to persistence of files)

Jitter – how to help apps deal with this?

- Jitter avoidance – get it out of the system (Blue Gene approach)
- jitter notification - inform applications about level of jitter
- Jitter tolerance – create apps that don't care about jitter
- Tools – jitter
  - measurement tools (quantify the amount of jitter and its impact)
  - help identify what parts of the code are most sensitive to jitter

work steeling schedulers is an example of one of these feeback loops

granularity of feedback loops, some are core level, node level, full machine

timescales vary too – machines cycles to scientific work flows

constraints vs. trying to maximize/minimize a value

coordination needs to span into network and I/O system too