B/2: Working With Code Teams

(What are the models and how are the applied math and algorithm scaling addressed?)

DOE HPC Operations Review
San Francisco, November 5-6, 2013
Breakout participants

- Barb Helland (ASCR)
- Tim Williams*, ANL
- Harvey Wasserman*, Brian Austin, Jack Deslippe, John Shalf (NERSC)
- John Gyllenhaal, Mary Zosel (LLNL)
- Jennifer Green (LANL)
- Wayne Joubert, Bronson Messer (ORNL)
- Joel Stevenson, Dino Pavlakos (SNL)

* Denotes breakout session lead
Processes (scope of activity)

- What needs to be done?
  - Prepare applications for forthcoming hardware.
  - Address parallelism in current codes. Have teams ready to do it.
  - Convene application readiness teams; means different things at different labs.
  - Persistence of efforts; Q: at what point are teams successful (Include V&V? Code is running on day 1? Thru entire machine lifecycle?); difference between app readiness and system SW readiness; *not* a porting activity but helping at some labs can involve rewriting/refactoring code; require identification of staff on code team to serve as interface to AR effort.
  - Q: is there sufficient driving force for use of new architectures w/o AR teams? A: No, facilitation is needed; “catalysts” is a better characterization.
  - Problem of application transience; 3 categories: always at LCFs, new at LCFs, in between. Makes it difficult to decide which teams to work with; level of need is an important factor.
  - Important to ensure that whatever work takes place becomes part of mainstream code efforts.
  - Importance of profiling, with tool (can’t always trust conventional wisdom).
Processes (scope of activity)

- Explaining architecture choices to code teams is an important activity
- Setting user expectations for newer systems
- Question of what to do about transitioning 3rd-party apps remains; users of these codes seem to be stranded
Processes (scope of activity), cont.

- What begins first: timeline for activities (before or after hardware)?
  - ID apps and appropriate problem sets, as well as personnel in center organization – preferably ~years before HW is available
  - Vendor-provided education, philosophy and periodic updates about systems for new platforms is essential
  - Tactical (shorter) and strategic (longer) work on codes
Processes (scope of activity), cont.

- What is the role of early hardware access (either locally or remotely) and prototype systems?
  - Existential (*modulo* the risk identified later); must be in the form of complete machines with at least beta-level system software;
  - This is required in order to have codes running on the main platform by day-one of installation
  - Key lesson learned is that desktop systems probably do not suffice for this, b/c do not adequately capture parallelism characteristics. (Sometimes similar for emulators.)
Processes (scope of activity), cont.

- What is the role of vendor partnerships/contracts and role of RD&E funds, NRE funds?
  - Significant (people) resource at vendors that we can tap, and the activity is mutually beneficial
  - BUT: unless vendors are getting $, doesn’t work well; => must be part of SOW and acceptance test; involves a lot of work for center overseeing vendor efforts
  - Role of research agreements, with less-well defined goals: Important but need sharper goals; importance of key vendor personnel (typically 1-2 people)
  - Important to get as much as possible from vendors during RFP response, especially on per-node app improvement
 Processes (scope of activity), cont.

- What are the roles of research and design and engineering (NRE)?
  
  • Important to have local researchers engaged in algorithms, tools, compilers, performance evaluation methodologies
  
  • Key activity for necessary libraries such as PETSc, Trilinos, etc., although OLCF used a local center person for this via one of the apps; question of how to drive this activity at a higher level remains – may be a HQ issue
Processes (scope of activity), cont.

- What resiliency activities are executed (for example, redundancy); how do app readiness efforts deal with higher failure rates
  - Encourage increased use of generic checkpoint/restart, signal capturing with apps, message verification. We have important role in providing and promoting techniques for apps to deal with lack of HW resiliency.
  - Need for improved monitoring capabilities to determine how well apps are using the machines.
  - Diagnosing failures: Intermediary between code, system teams
Organization and management

- What is the structure of the integration and preparation teams?
  - Specifically identify AR teams.
  - Personnel may have to be pulled off of other projects and directly funded for AR efforts. Funds came from center operations funds and/or project funds (e.g. ALCF-2)
Organization and management, cont.

- What are the necessary skills for the activity team (center app readiness personnel)?
  - Reasonable up-to-date knowledge of architecture and tools; need to be carefully plugged in to next-generation activities via researchers
  - People skills! Must gain trust of code team, which comes from some knowledge of the apps in question and demonstrating interest
Experiences and lessons learned

What were the good and bad experiences and lessons learned?

- Avoid dead-end disruptive technologies
- Don’t over-invest in porting to early architectures that don’t match final platforms
- Optimizations done for more exotic technologies tend to pay off across architectures; requires care in making comparisons.
  - Restructuring for GPUs lead to 2X speedup on CPUs
Experiences and lessons learned, cont.

- What were the most productive activities?
  - Direct interaction with users (and code teams)
  - Access to reasonably-sized, earliest hardware is vital
  - Collaborations with key vendor personnel is vital
Experiences and lessons learned, cont.

What were the resiliency experiences?

- Stable hardware for app transitioning is a necessity
- Lack of info about source of faults is a major issue in new systems; app readiness personnel are expected to provide info as intermediary with systems personnel
- I/O and filesystem issues tend to dominate at early phases of lifecycle
Experiences and lessons learned, cont.

- What were the highest risks? Surprise or expected?
  - Problem where early HW that doesn’t accurately represent final platform (surprise)
  - Swimlane risk; once refactoring is done for improved parallelism on existing architectures, this risk becomes minimal
  - Can the operational entities adequately engage code teams?
  - Not enough applications ready on day 1.
Most significant observation

- Provide a summary statement for the most significant observation
  - Success of the newer systems depends critically on robust, well-funded, early and active involvement with code teams—AR facilitators ready to do “whatever it takes.”
Effort estimate

- How big of an effort was this?
  - Application readiness: 1-3 person-years per app.
    - Large fraction may be restructuring rather than specifics for new hardware (or, algorithmic changes needed)
  - ~10 codes (at each center)
Use the elements here to enhance your presentation

<table>
<thead>
<tr>
<th>Head 1</th>
<th>Head 2</th>
<th>Head 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table text</td>
<td>Table text</td>
<td>Table text</td>
</tr>
<tr>
<td>Table text</td>
<td>Table text</td>
<td>Table text</td>
</tr>
<tr>
<td>Table text</td>
<td>Table text</td>
<td>Table text</td>
</tr>
<tr>
<td>Table text</td>
<td>Table text</td>
<td>Table text</td>
</tr>
<tr>
<td>Table text</td>
<td>Table text</td>
<td>Table text</td>
</tr>
</tbody>
</table>