

# Questions for Discussion (3)

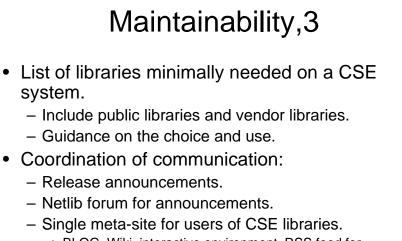
- Role of vendors / SW companies
  - What do they build, what do we build?
  - What do they support us to build?
  - Multicore as opportunity to fund building some kernels
  - Open source and/or proprietary
    - Licensing (LGPL vs mBSD)
- Tools for future
  - Scalability testbed (eg RAMP)
  - Reproducibility

### Maintainability

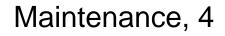
- Hong:
  - 50% time helping users.
  - Automation will not help.
  - 3 people continuous for PETSc.
  - Mike H: 3 people for Trilinos.
  - Documentation alone does not eliminate.
  - One-to-one is very important.
  - Users are testers. Provide ideas for new development.



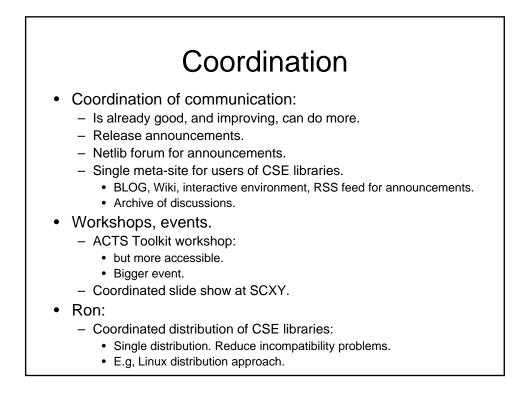
- Marc:
  - Tutorial for users starting from the class of problem they want to solve.
  - Database of what is available to solve my problem.
- Jack: Coordination of the libraries: DOE, Vendors.
  - How the libraries install, work together.
  - Common look & feel, common accessibility.



- BLOG, Wiki, interactive environment, RSS feed for announcements.
- Archive of discussions.

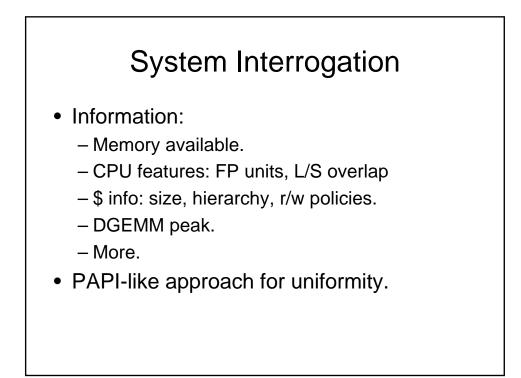


- Model of support is broken.
  - Mature, used but not actively developed, software is not well supported.
  - DOE has large collection of very valuable software.
  - Stewardship: little is done.
  - Should be an incentive to continue development of successful SW.
  - Currently penalized, since new development is given priority.



# Jack's 4 challenges

- Manycore: no contention.
- Autotuning: no contention.
  - Addressing several axes of performance:
    - Speed, memory use, accuracy, etc.
    - Saving power, reduce clock speed dynamically.
- Fault-tolerance (at algorithm level).
- Use of mixed precision:
  - For performance & accuracy.
  - For memory use & and power consumption.



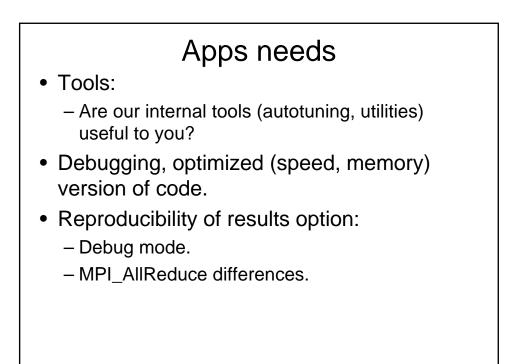
# What Apps need

• Serguei:

- Standard CSE software environment:
  - Autotools, BLAS, LAPACK, etc.
  - Fortran compiler.
  - Minimal set: RedHat package set.
  - Would enable binary distribution.
- Installability
- Windows install tool.
- Binary distribution.

# Matlab-like APIs

- Needed for Petascale?
- How seriously should we think about Matlab (Star-P, Python, Octave) as the API? YES!
- Productivity issue.
- Used natively or to generate code, or both?



# Apps needs

- Rich:
  - Global sparse triangular solve is present bottleneck.
  - Can we develop an alternative at any level:
    - Better implementation.
    - Brand new algorithmic approach.
- Marc: Standard benchmark targets for some critical functionalities:
  - Global sparse triangular solve.
  - SpMV for several app areas.
  - Bakeoffs?
- Improved feedback loop from users:
  - Usage, problems.
  - Formal observation events of usage.
- Julien:
  - Good software engineering practices need to be transmitted to apps developers.
  - From library developers to apps developers: good design, best practices, etc.

### **Transition to Manycore**

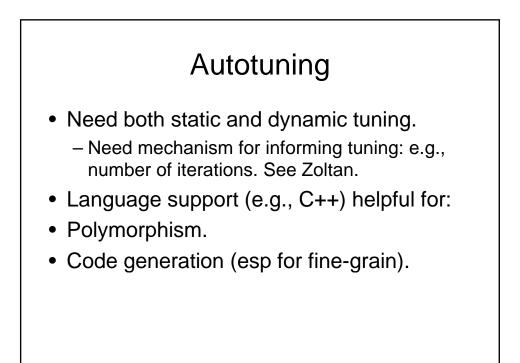
- Libraries migrate first.
  - Need a standard mechanism to go from flat MPI to MPI+shared, dynamically.
  - App will be running MPI-only.
- Translation tools for app:
  - Help migration.
  - Can it be transparent to the app?

#### Manycore concerns

- HW model is still vague:
  Shared memory, local memory, cache coherent?
- SW model not clear.
- Parallel changes ubiquitous:
  - Transition from serial to MPI: MPI forced app framework changes, but left vast majority of complex physics code unchanged.
  - Vectorization: Happened automatically.
  - Manycore parallel will not be automatic (?).
  - Transition from MPI-only to MPI+manycore: Changes will be more disruptive, pervasive.

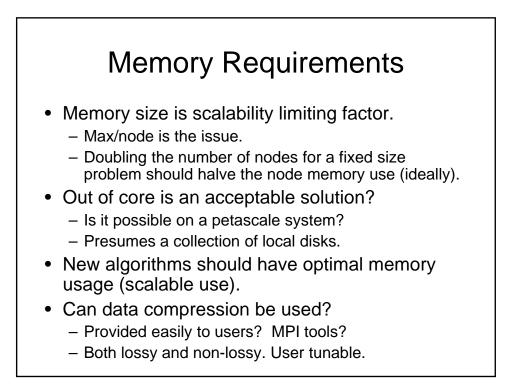
## Manycore concerns

- Large-scale regeneration of libraries is easy to justify:
  - impacts thousands of users
  - only so many libs.
  - Small relative total cost.
- Similar rewrite of apps less broad impact:
  - may impact fewer users,
  - 100s or 1000s of apps.
  - Large total cost.
  - Need tools to reduce this cost.
- Typical programmer in MPI code does not need expert knowledge of MPI.
- Can we abstract the parallelism of manycore so the average programmer does not need to think in parallel?



# MPI needs

- Better support for overlapping comm & and comp.
- Becomes more important for manycore because of bandwidth issues.
- Asynch doesn't work all the time.
- Even parallel language extensions (CAF, UPC) don't give user control over process for most efficient execution.



#### Complete app rewrite? E.g. In Chapel/Fortress

- Ron:
  - Small codes:
    - Common in some areas:
      - Dynamics (chem), 100s-1000s LOC.
    - Possible. Weeks to months to rewrite.
  - Large codes:
    - Gaussian comps.
    - O(100K-1M) LOC. (GAUSSIAN O(2M) LOC)
    - NWCHEM O(1M) LOC. Requires 50 man-years to rewrite.
- Rich:
  - Too large, too costly to verify correctness.
- Serguei:
  - Just a few small codes.
  - Most important codes too expensive.

# Debugging/profiling parallel codes

- Still really hard.
- Especially large-PE-count-only failures.
  - Runs on 10s or 100s of Pes, not on 1000s or more.
- Profiling:
  - Performance.
  - Memory use: Sampling capabilities.
    - Esp. non-virtual memory machines.