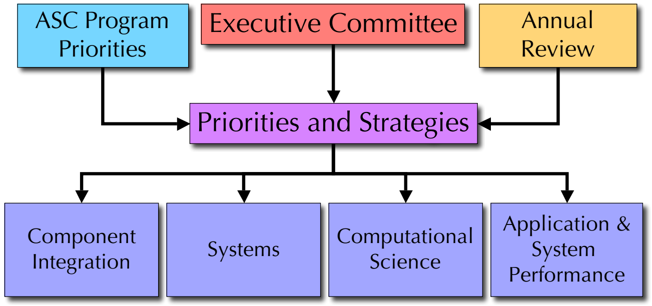
LACSI Management and ASC Impact

Review

Ken Kennedy
Rice University
LACSI Co-Director

http://lacsi.rice.edu/review/slides_2006/

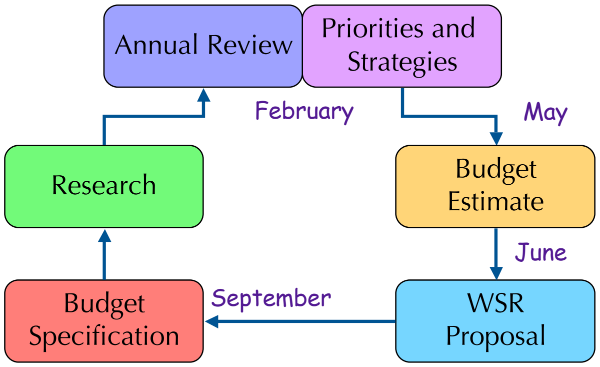
LACSI Structure and Organization



Leadership

- **Co- Directors**
  - Andy White (Los Alamos) & Ken Kennedy (Rice)
- **Executive Committee**
  - LANL: Jeff Brown, Bill Feiereisen, Adolfy Hoisie, Doug Kothe, Rod Oldehoeft, John Thorp, Andy White
  - Rice: Rob Fowler, Ken Kennedy, John Mellor-Crummey, Linda Torczon
  - Houston: Lennart Johnsson, Yuri Kuznetsov
  - New Mexico: Deepak Kapur
  - North Carolina: Dan Reed
  - Tennessee: Jack Dongarra

New Annual Planning Cycle



New Review Strategy

- Establish LACSI Review Board (LRB)
  - Include LANL and ASC stakeholders (ASC application developers, computer and computational scientists)
  - External reviewers with multi-year terms
- Combine annual review by LRB with P&S meeting
  - Schedule LRB review on the day before the P&S meeting
  - Outbrief with Executive Committee and formal report
- Use outcome of review as input to P&S planning meeting
  - P&S document becomes proposal for the next year's funding
    - Direct input into academic SOW and LANL ASC IP
- Maintain multi-year stability of projects
  - Phase out unsuccessful projects after 2 or 3 years
  - Intermediate reviews provide constructive criticism

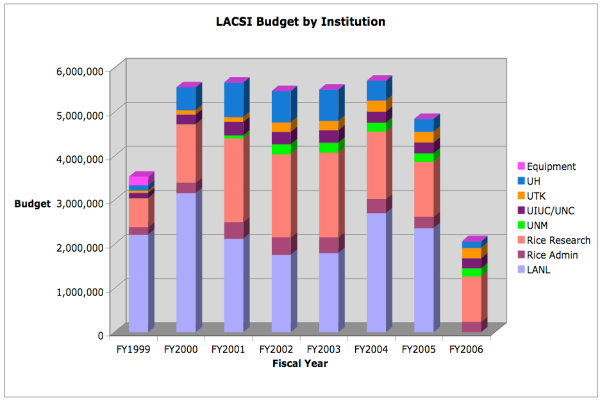
Priorities and Strategies Meeting

- FY06 Attendees (held Feb 2005):
  - Marv Alme, Jeff Brown, John Cerutti, Darren Kerbyson, Ken Koch, Stephen Lee, Craig Rasmussen, Bill Feiereisen, Rich Graham, Adolfy Hoisie, Chip Kent, Darren Kerbyson, Brett Kettering, Doug Kothe, Rod Oldehoeft, Scott Pakin, Susan Post, Mikhail Shashkov,John Thorp, Greg Watson, Andy White, Zoran Budimlic, Mike Fagan, Rob Fowler, Ken Kennedy, John Mellor-Crummey, Dan Sorensen, Linda Torczon, Deepak Kapur, Barney Maccabe, Jack Dongarra, Lennart Johnsson, Yuri Kuznetsov, Dan Reed, Scott Rixner
- Typical Agenda
  - Review of previous year's P&S plan
  - Discussion in plenary session
  - Break into discussion groups for developing plans for the next year
    - Application and System Performance, Components, Systems, Computational Science
  - Presentation of revised plans with discussion in plenary session
  - Document developed by email after the meeting

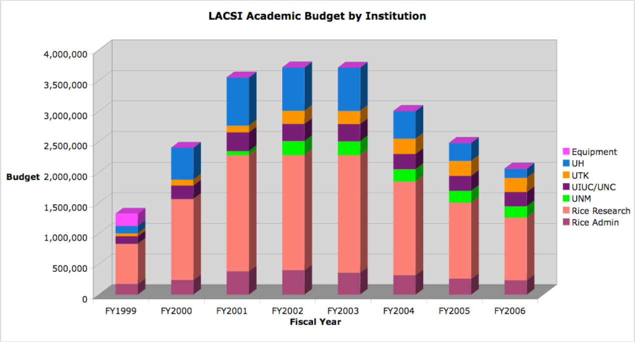
Concerns

- Budget Issues
  - Continuing trend of budget reductions
  - No funded LANL participants
  - Unclear how WSR will treat the program this year
    - More, smaller proposals
    - Will we lose integrated planning advantages?
- Collaboration Issues
  - Research should be driven by real LANL applications
    - Easier when we have more Q-cleared researchers
    - We also need to streamline methods for getting access to export-restricted codes
  - Getting attention of application developers is sometimes tricky
- Contracting difficulties
  - Working on a no-cost extension from FY05 (soon to be funded?)

LACSI Budget History by Institution



LACSI Academic Budget History



Other Management Challenges

- Enhancing collaboration
  - We understand what works: direct interaction
  - How can we foster more of this?
  - Santa Fe Information Technology lab would be Plan A
    - We are currently engaged in Plan B
    - What incentives are needed on the LANL side?
- Enhancing visibility within LANL
  - Not enough LANL staffers know about LACSI
    - No tangible point of presence
- Applicability: Bringing research (more directly) to bear on ASC problems
  - The problem of classified codes versus sanitized benchmarks
    - More sanitized codes (and data), more cleared academic researchers

Process for Deployment

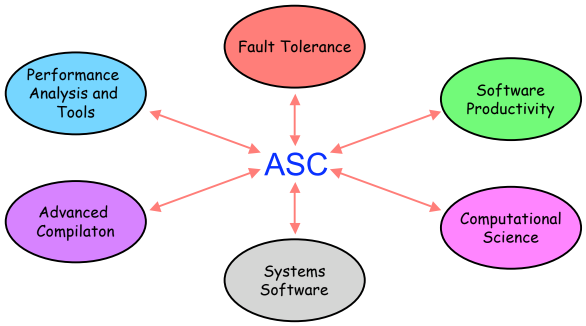
- This is difficult, systemic problem that transcends ASC and LANL
  - Not enough resources in research to support deployment
  - Not enough production resources to move all universally accepted research prototypes into production
- There have been successes, as well as frustrations
  - LACSI cannot solve this problem by itself
  - We will continue to try to bring the parties together

Impact of LACSI Research

On Advanced Simulation and Computing

Ken Kennedy
Rice University

ASC Impact



LACSI

HPCToolkit

- Long-term compiler and architecture research requires detailed performance understanding
  - identify sources of performance bottlenecks in complex applications
  - discover automatic strategies for performance improvement
- Short-term result: Programmer-accessible tools for understanding application performance
- HPCToolkit installed and used on ASC systems at LANL
  - Nirvana/Blue Mountain, Q, Lightning (Clustermatic)
- Conducted performance tuning workshops at LANL
  - Project A, Project B, Telluride, MCNP, Blanca teams
- Used to assess FLAG for ASC Burn Code review in 2003
- Applied to two X-Division codes
- Enhancements for cumulative call-graph profiling and monitoring of memory management underway

LACSI

Tuning of Codes

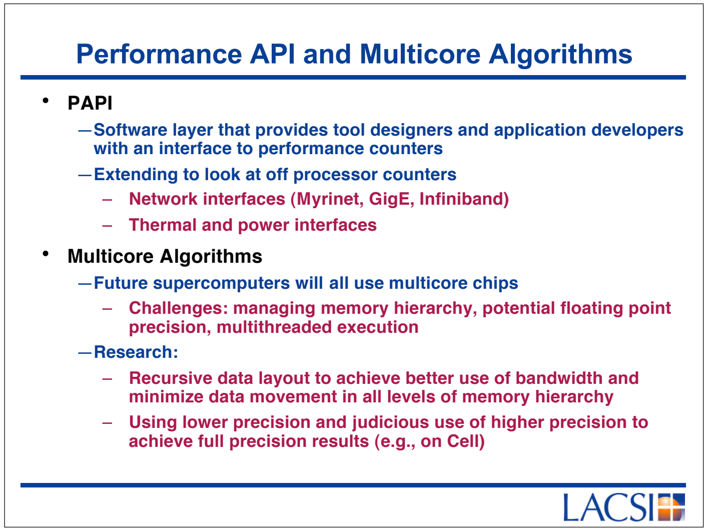
- Sage
  - Elimination of unnecessary copying: 2x on Blue Mountain, 1.5x on Q
  - New sparse matrix representation (not integrated): 2x on Itanium
- Sweep3D
  - Memory hierarchy transformations: 1.44x on Alpha, 1.9x on Origin
- Blanca
  - Algorithms for reordering vertex lists: $O(n^2)$ -> O(n log n)
    - 26x on the 5-level refinement case
- CHAD
  - Reordering of accesses to irregular mesh based on space-filling curves: 2x improvement over random ordering
  - Better scalarization of F90 array accesses: reduced memory traffic by 40%

LACSI

Performance Prediction

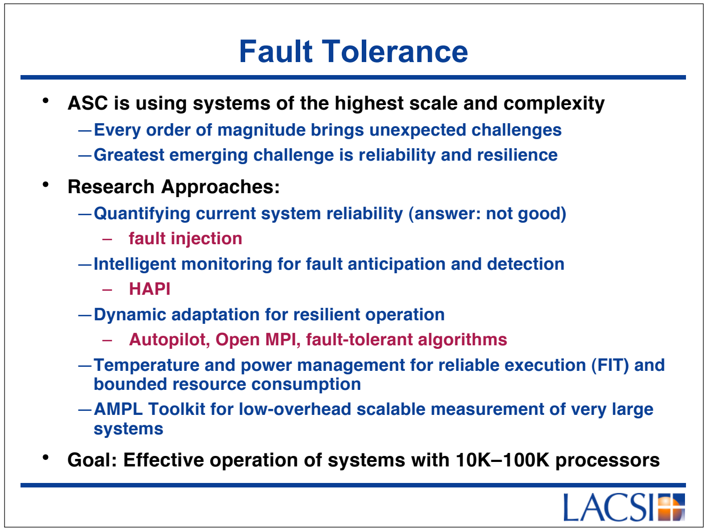
- Application Modeling
  - Run SAGE and Sweep3D on single processor ⇒
    - Performance prediction of LANL workload on parallel processor
    - Improved performance on Q by 2x
  - But what if you cannot run these applications?
    - Processor not available (being designed, …)
- Single-node Cross-architecture Performance Prediction
  - Predict the performance of known applications on an arbitrary processor
  - Extends Capability of LANL Performance Analysis Methodology
- Currently
  - Working with Olaf Lubeck to explore the impact of memory latency on the performance of ASC codes
  - Also looking at dual-core performance prediction

LACSI



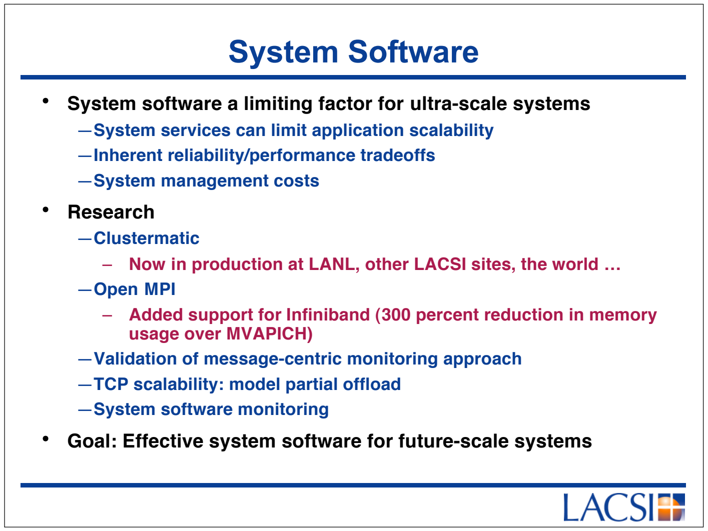
Performance API and Multicore Algorithms

- PAPI
  - Software layer that provides tool designers and application developers with an interface to performance counters
  - Extending to look at off processor counters
    - Network interfaces (Myrinet, GigE, Infiniband)
    - Thermal and power interfaces
- Multicore Algorithms
  - Future supercomputers will all use multicore chips
    - Challenges: managing memory hierarchy, potential floating point precision, multithreaded execution
  - Research:
    - Recursive data layout to achieve better use of bandwidth and minimize data movement in all levels of memory hierarchy
    - Using lower precision and judicious use of higher precision to achieve full precision results (e.g., on Cell)



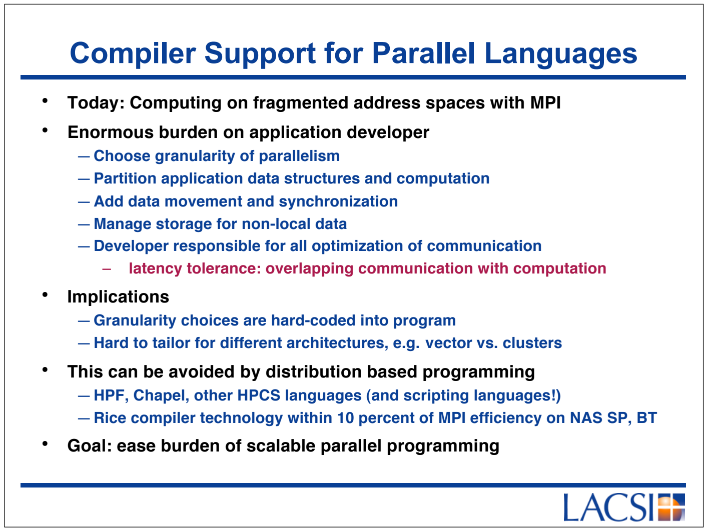
Fault Tolerance

- ASC is using systems of the highest scale and complexity
  - Every order of magnitude brings unexpected challenges
  - Greatest emerging challenge is reliability and resilience
- Research Approaches:
  - Quantifying current system reliability (answer: not good)
    - fault injection
  - Intelligent monitoring for fault anticipation and detection
    - HAPI
  - Dynamic adaptation for resilient operation
    - Autopilot, Open MPI, fault-tolerant algorithms
  - Temperature and power management for reliable execution (FIT) and bounded resource consumption
  - AMPL Toolkit for low-overhead scalable measurement of very large systems
- Goal: Effective operation of systems with 10K–100K processors



System Software

- System software a limiting factor for ultra-scale systems
  - System services can limit application scalability
  - Inherent reliability/performance tradeoffs
  - System management costs
- Research
  - Clustermatic
    - Now in production at LANL, other LACSI sites, the world …
  - Open MPI
    - Added support for Infiniband (300 percent reduction in memory usage over MVAPICH)
  - Validation of message-centric monitoring approach
  - TCP scalability: model partial offload
  - System software monitoring
- Goal: Effective system software for future-scale systems



Compiler Support for Parallel Languages

- Today: Computing on fragmented address spaces with MPI
- Enormous burden on application developer
  - Choose granularity of parallelism
  - Partition application data structures and computation
  - Add data movement and synchronization
  - Manage storage for non-local data
  - Developer responsible for all optimization of communication
    - latency tolerance: overlapping communication with computation
- Implications
  - Granularity choices are hard-coded into program
  - Hard to tailor for different architectures, e.g. vector vs. clusters
- This can be avoided by distribution based programming
  - HPF, Chapel, other HPCS languages (and scripting languages!)
  - Rice compiler technology within 10 percent of MPI efficiency on NAS SP, BT
- Goal: ease burden of scalable parallel programming

Higher-level Programming Models

- Most important barrier to progress on application development in the ASC program is limited human productivity
- Goals:
  - Simplify programming of parallel systems
  - Make applications more malleable
  - Enhance performance portability
  - Support rapid prototyping
- Research strategies:
  - Compiler and run-time technology for data parallel languages
  - Compilation of high-level scripting languages (Python,Matlab,…)
    - Augmented with domain-specific component libraries
    - Parallel Matlab implementation underway using data parallel support
  - High level prototyping frameworks

Open Source Compilers

- DOE Interest
  - Porting legacy Fortran codes to new platforms
  - Concerns: single front-end, compatibility/portability across platforms
- GCC lacks capabilities for optimizing scientific programs
- Possible Replacements
  - Open64/ORC is a strong candidate
  - LLVM is another candidate
- Mellor-Crummey leading the Open64 consolidation effort
- Cooper is involved in LLVM effort
- We are proposing an Institute to manage compiler-based open-source software for high performance computing
  - DOE Office of Science

Software Productivity

- Component-based software development is a key strategy for improving productivity of ASC application development teams
- Compiler Optimization of Object-Oriented Languages
  - Rice effort focused on aspects unique to scientific computing
    - Object inlining of arrays of objects
  - JaMake infrastructure: 1.8x improvement on Parsek
  - Usable for C++ and Python
  - Investigation of application to Ajax initiated
- Component Integration
  - Refocused goal to make component-based software more efficient
    - Precompilation of component libraries to improve performance of fine grained components
- Automatic Tuning of Components
  - Combining fusion (essential for F90) with cache tiling using models
  - Initiated community building effort

Computational Science

- Automatic Differentiation
  - Computing sensitivities of codes to various input parameters
  - Important for analysis of numerical accuracy (critical to V&V)
    - Active collaboration with several LANL users
  - Extended to Fortran 90 this year
    - Applying to Truchas and to FLAG
- Advanced Diffusion Methods
  - Yuri Kuznetsov (Houston) collaborates with LANL researchers in T-Division and CCS-Division, and one of his students is a staff member,another is a postdoctoral fellow at LANL
  - He recently conceived of a fundamentally new approach for solving the diffusion equation on general polyhedral meshes
    - This approach was refined by LANL researchers and implemented in a test code
    - The resulting method represents a genuine breakthrough
  - Implemented in FLAG (parallel version also developed)

Impact Summary

- LACSI is working on problems that are critical to the future of ASC applications
  - As observed in the 2004 review
- LACSI researchers have changed directions in response to ASC program needs
- Several intermediate results of LACSI research have had immediate impact
  - HPCToolkit, PAPI, polyhedral mesh diffusion methods, tuning of ASC codes