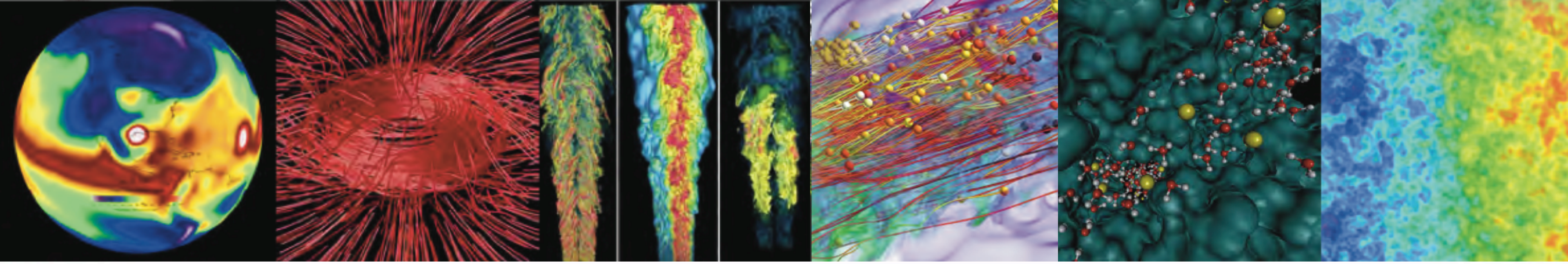
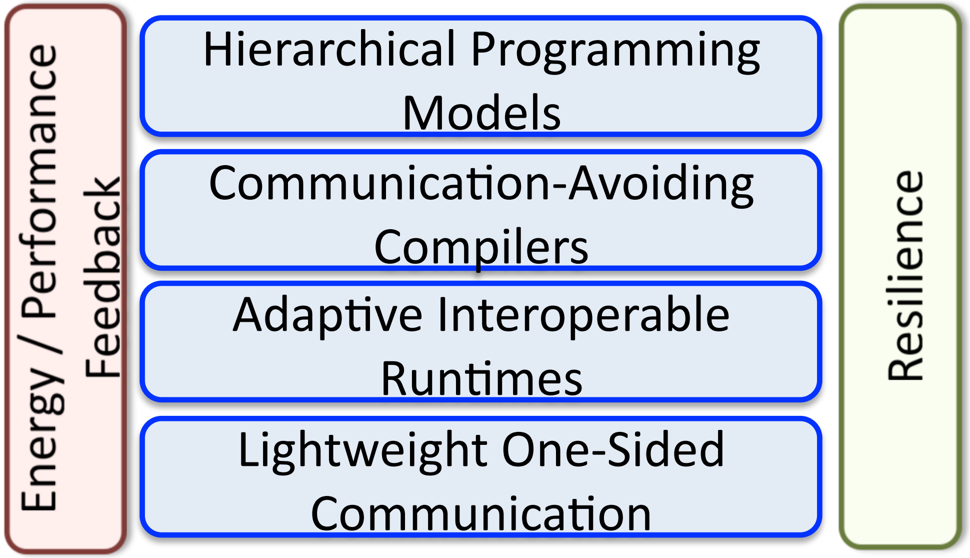


DEGAS:
Dynamic Exascale Global Address Space

Katherine Yelick, LBNL PI
Vivek Sarkar & John Mellor-Crummey, Rice
James Demmel, Krste Asanoviç & Armando Fox, UC Berkeley
Mattan Erez, UT Austin
Dan Quinlan, LLNL
Surendra Byna, Marc Day, Tony Drummond, Paul Hargrove, Steven Hofmeyr, Costin Iancu, Khaled Ibrahim, Frank Mueller (NCSU), Leonid Oliker, Eric Roman, John Shalf, David Skinner, Erich Strohmaier, Brian Van Straalen, Samuel Williams, Yili Zheng, LBNL

DEGAS Mission

Mission Statement: To ensure the broad success of Exascale systems through a unified programming model that is productive, scalable, portable, and interoperable, and meets the unique Exascale demands of energy efficiency and resilience

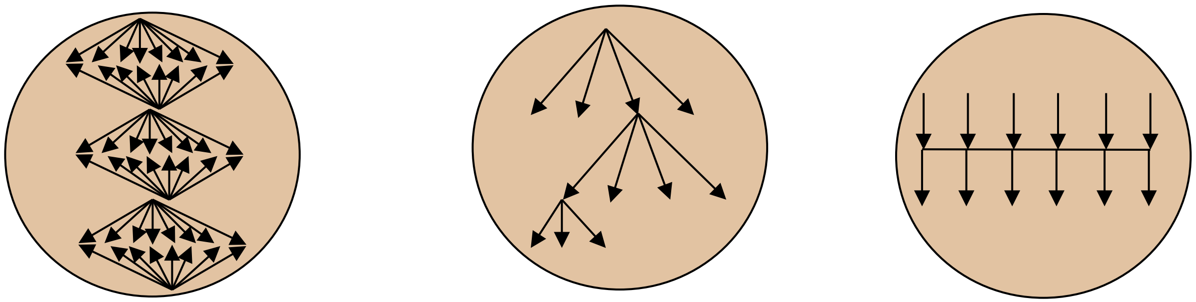


DEGAS Proposal: Goals and Objectives

- **Scalability:**
  - Billion-way concurrency, thousand-way on chip with new architectures
- **Programmability:**
  - Convenient programming through a global address space and high-level abstractions for parallelism, data movement and resilience
- **Performance Portability:**
  - Ensure applications can be moved across diverse machines using implicit (automatic) compiler optimizations and runtime adaptation
- **Resilience:**
  - Integrated language support for capturing state and recovering from faults
- **Energy Efficiency:**
  - Avoid communication, which will dominate energy costs, and adapt to performance heterogeneity due to system-level energy management
- **Interoperability:**
  - Encourage use of languages and features through incremental adoption

Two Distinct Parallel Programming Questions

- **What is the parallel control model?**

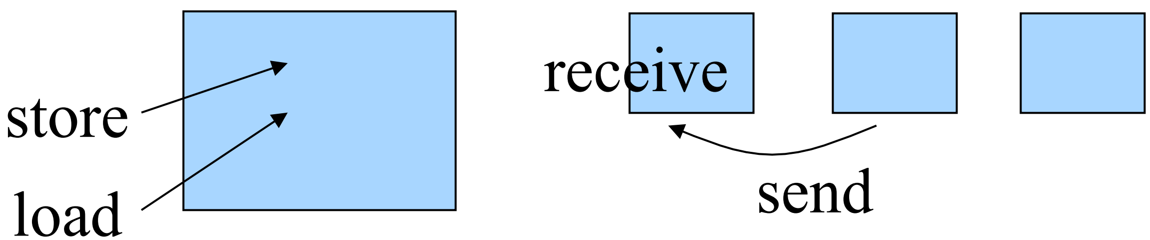


data parallel
(singe thread of control)

dynamic
threads

single program
multiple data (SPMD)

- **What is the model for sharing/communication?**

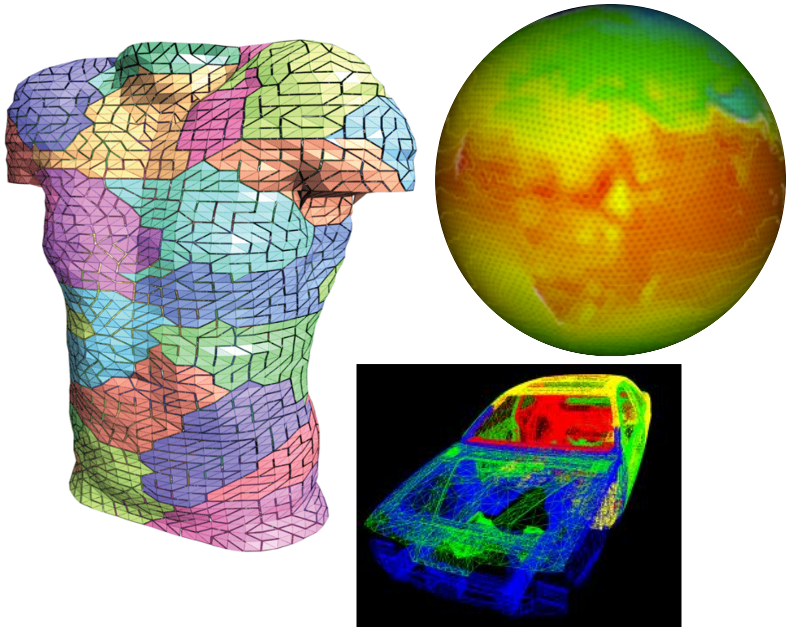


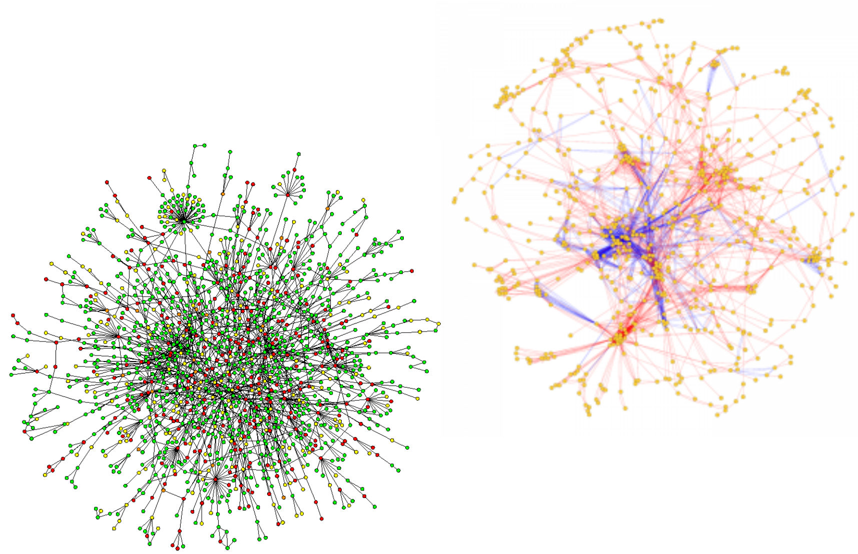
shared memory

message passing

synchronization may be coupled (implicit) or separate (explicit)

Applications Drive New Programming Models





Message Passing Programming

Divide up domain in pieces

Compute one piece and exchange

MPI, and many libraries

Global Address Space Programming

Each start computing

Grab whatever / whenever

UPC, CAF, X10, Chapel, Fortress, Titanium, GlobalArrays

Mechanisms, not Policies

PGAS + Mixins

DEGAS: Hierarchical Programming Model

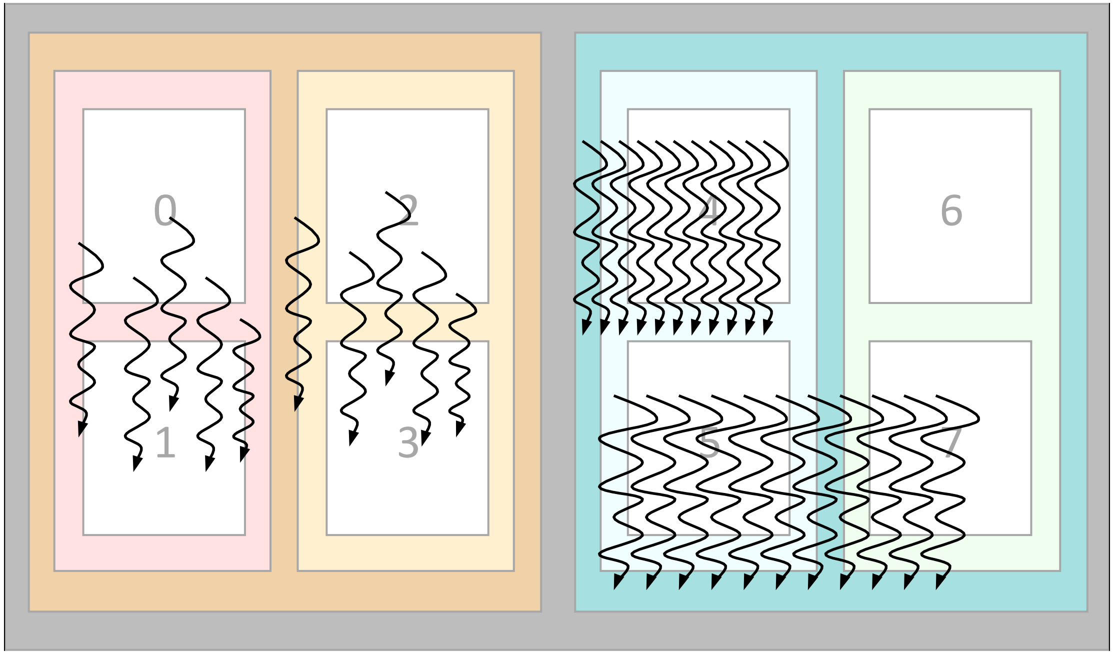
Goal: Programmability of exascale applications while providing scalability, locality, energy efficiency, resilience, and portability

- **Implicit constructs:** parallel multidimensional loops, global distributed data structures, adaptation for performance heterogeneity
- **Explicit constructs:** asynchronous tasks, phaser synchronization, locality

Built on scalability, performance, and asynchrony of PGAS models

- Language experience from UPC, Habanero-C, Co-Array Fortran, Titanium

Both intra and inter-node; focus is on node model



DEGAS: Hierarchical Programming Models

Languages demonstrate DEGAS programming model

- **Habanero-UPC:** Habanero's intra-node model with UPC's inter-node model
- **Hierarchical Co-Array Fortran (CAF):** CAF for on-chip scaling and more
- **Exploration of high level languages:** E.g., Python extended with H-PGAS

Language-independent H-PGAS Features:

- Hierarchical distributed arrays, asynchronous tasks, and compiler specialization for hybrid (task/loop) parallelism and heterogeneity
- Semantic guarantees for deadlock avoidance, determinism, etc.
- Asynchronous collectives, function shipping, and hierarchical places
- End-to-end support for asynchrony (messaging, tasking, bandwidth utilization through concurrency)
- Early concept exploration for applications and benchmarks

DEGAS: Communication-Avoiding Compilers

Goal: massive parallelism, deep memory and network hierarchies, plus functional and performance heterogeneity

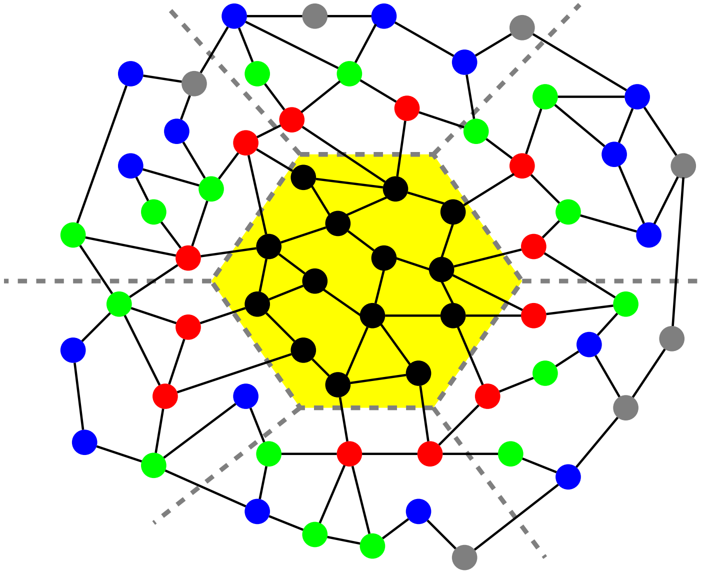
- **Fine-grained task and data parallelism:** enable performance portability
- **Heterogeneity:** guided by functional, energy and performance characteristics
- **Energy efficiency:** minimize data movement and hooks to runtime adaptation
- **Programmability:** manage details of memory, heterogeneity, and containment
- **Scalability:** communication and synchronization hiding through asynchrony

H-PGAS into the Node

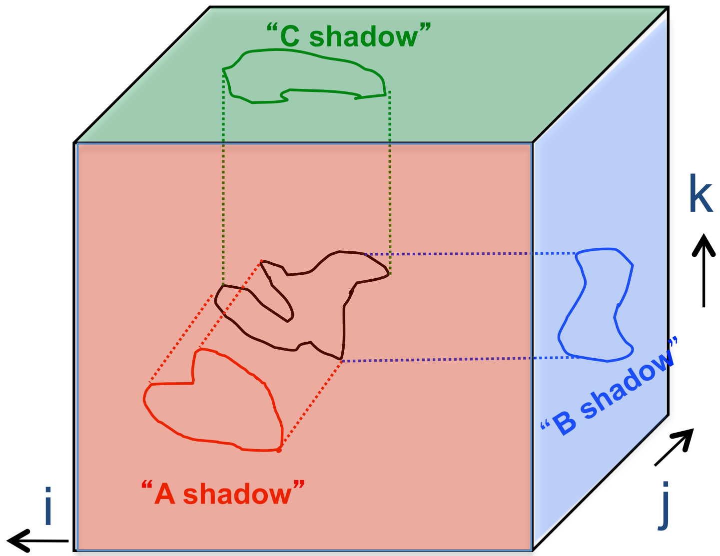
- Communication is all data movement

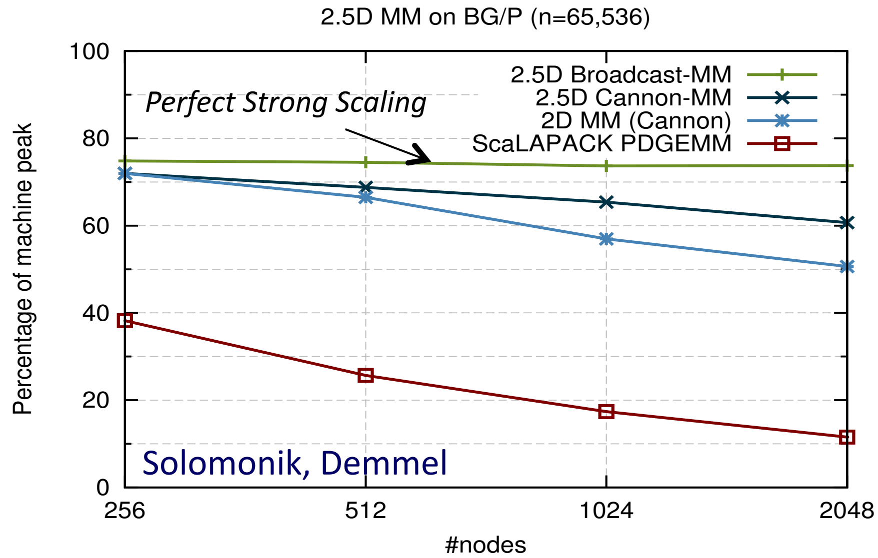
Build on code-generation infrastructure

- ROSE for H-CAF and Communication-Avoidance optimizations
- BUPC and Habanero-C; Zoltan
- Additional theory of CA code generation



Exascale Programming: Support for Future Algorithms





Approach: "Rethink" algorithms to optimize for data movement

- New class of communication-optimal algorithms
- Most codes are not bandwidth limited, but many should be

Challenges: How general are these algorithms?

- Can they be automated and for what types of loops?
- How much benefit is there in practice?

DEGAS: Adaptive Runtime Systems (ARTS)

Goal: Adaptive runtime for manycore systems that are hierarchical, heterogeneous and provide asymmetric performance

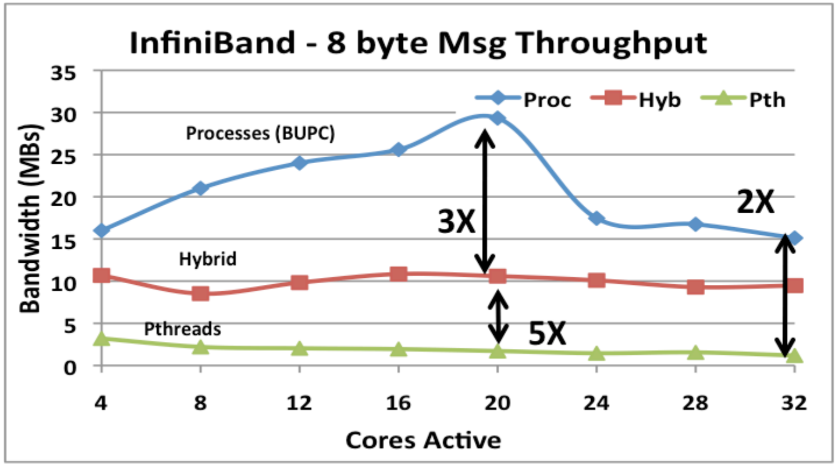
- **Reactive and proactive control** for utilization and energy efficiency
- **Integrated tasking and communication:** for hybrid programming
- **Sharing of hardware threads:** required for library interoperability

Novelty: scalable control; integrated tasking with communication

- **Adaptation:** Runtime annotated with performance history/intentions
- **Performance models:** guide runtime optimizations, specialization
- **Hierarchical:** resource / energy
- **Tunable control:** Locality / load balance

Leverages: existing runtimes

- **Lithe** scheduler composition; **Juggle**
- **BUPC and Habanero-C** runtimes



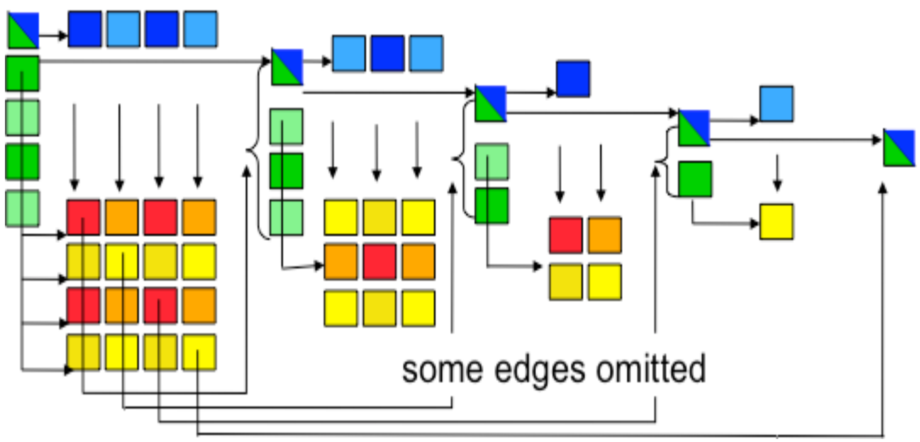
Synchronization Avoidance vs Resource Management

Management of critical resources will be more important:

- **Memory and network bandwidth limited** by cost and energy
- **Capacity limited at many levels:** network buffers at interfaces, internal network congestion are real and growing problems

Can runtimes manage these or do users need to help?

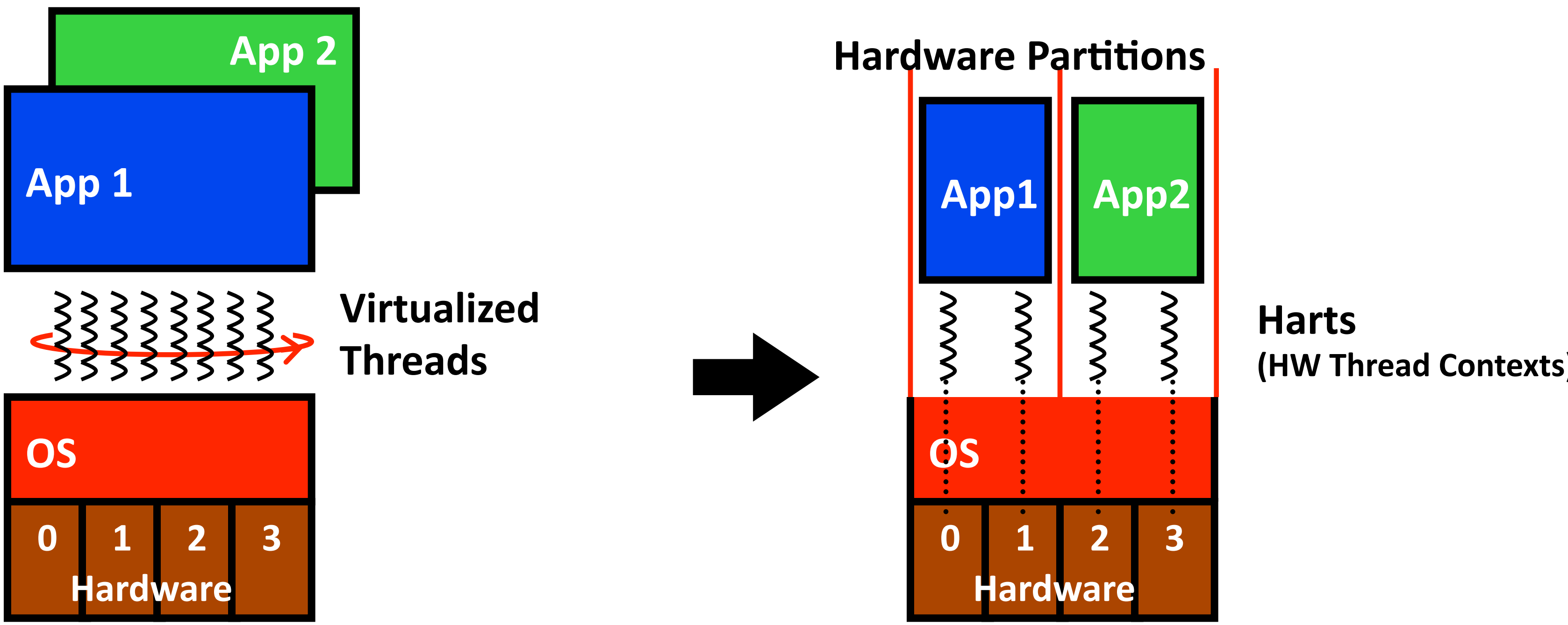
- Adaptation based on history and (user-supplied) intent?
- Where will bottlenecks be for a given architecture and application?



Resource management is complicated. Progress, deadlock, etc. are much more complex (or expensive) in distributed memory

Lithe Scheduling Abstraction: "Harts": Hardware Threads

POSIX Threads | **Harts**



- *Merged* resource and computation abstraction.

- More accurate resource abstraction.
- Let apps provide own computation abstractions

DEGAS: Lightweight Communication (GASNet-EX)

Goal: Maximize bandwidth use with lightweight communication

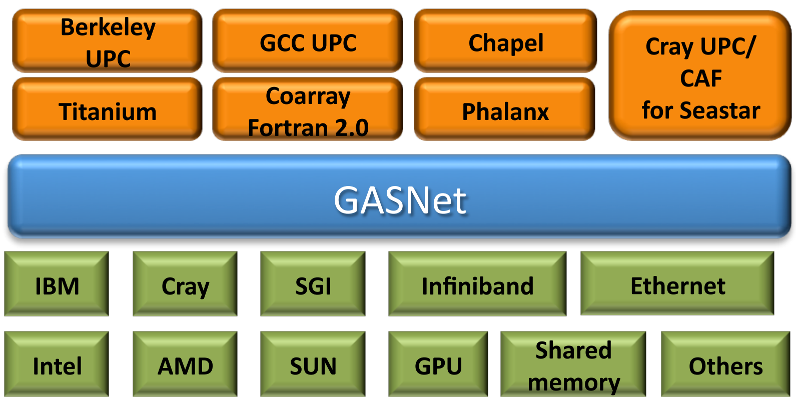
- **One-sided communication:** to avoid over-synchronization
- **Active-Messages:** for productivity and portability
- **Interoperability:** with MPI and threading layers

Novelty:

- **Congestion management:** for 1-sided communication with ARTS
- **Hierarchical:** communication management for H-PGAS
- **Resilience:** globally consist states and fine-grained fault recovery
- **Progress:** new models for scalability and interoperatbility

Leverage GASNet (redesigned)

- Major changes for on-chip interconnects
- Each network has unique opportunities



DEGAS: Resilience through Containment Domains

Goal: Provide a resilient runtime for PGAS applications

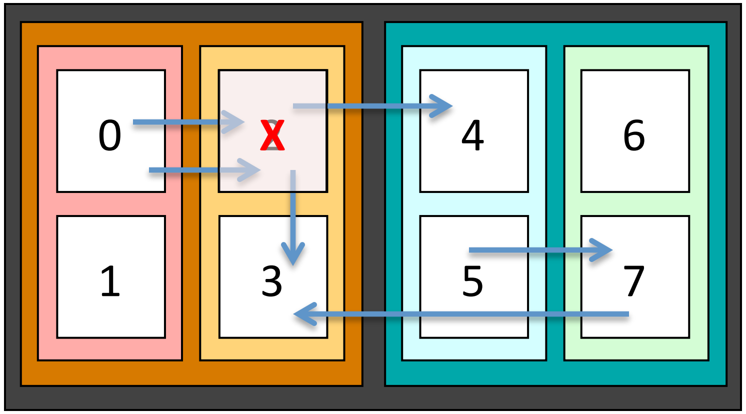
- Applications should be able to customize resilience to their needs,
- Resilient runtime that provides easy-to-use mechanisms

Novelty: Single analyzable abstraction for resilience

- PGAS Resilience consistency model
- Directed and hierarchical preservation
- Global or localized recovery
- Algorithm and system-specific detection, elision, and recovery

Leverage: Combined superset of prior approaches

- Fast checkpoints for large bulk updates
- Journal for small frequent updates
- Hierarchical checkpoint-restart
- OS-level save and restore
- Distributed recovery



DEGAS Resilience: Research Questions

1. *How to define consistent (i.e. allowable) states in the PGAS model?*

   Theory well understood for fail-stop message-passing, but not PGAS.

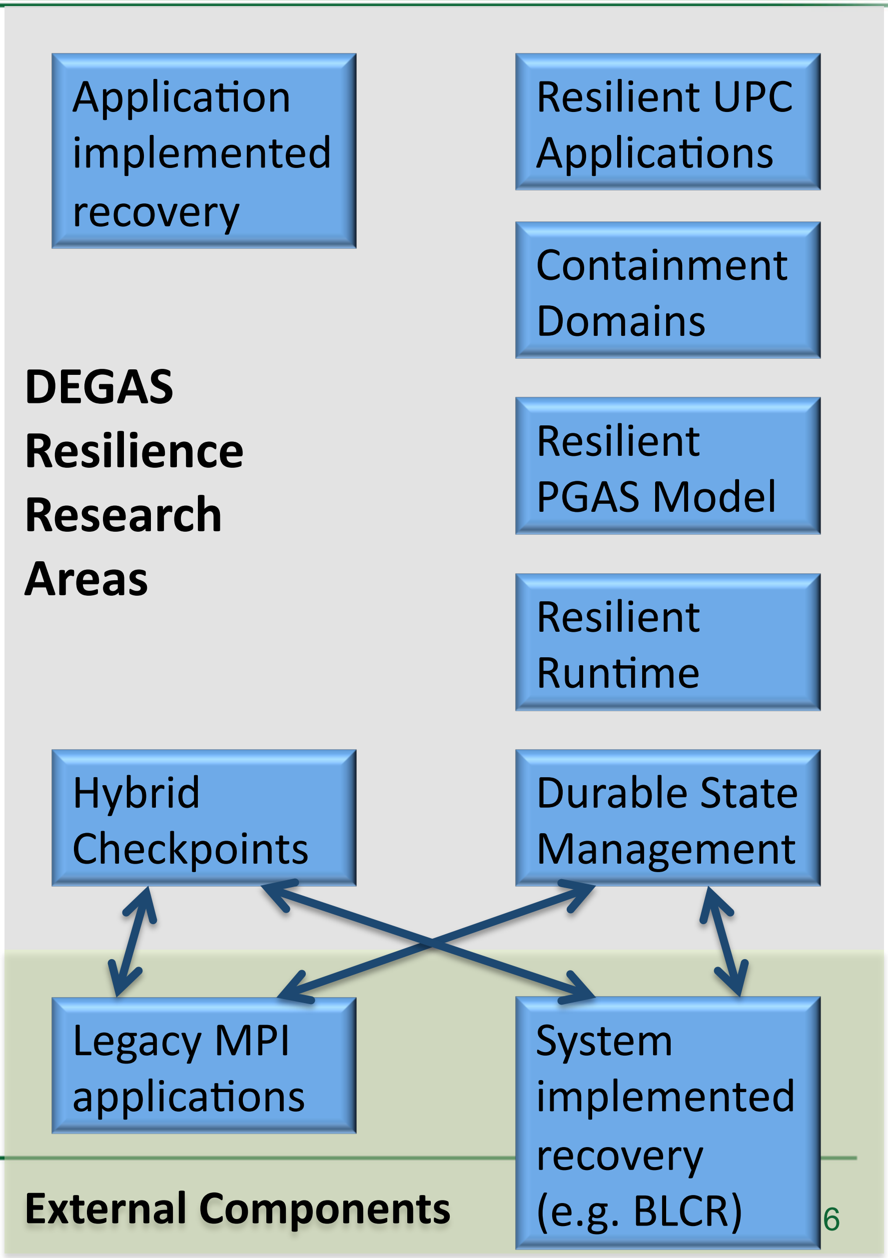
2. *How do we discover consistent states once we've defined them?*

   Containment domains offer a new approach, beyond conventional *sync-and-stop algorithms.*

3. *How do we reconstruct consistent states after a failure?*

   Explore low overhead techniques that minimize effort required by applications programmers.

   Leverage BLCR, GASnet, Berkeley UPC for development, and use Containment Domains as prototype API for requirements discovery



DEGAS: Energy and Performance Feedback

Goal: Monitoring and feedback of performance and energy for online and offline optimization

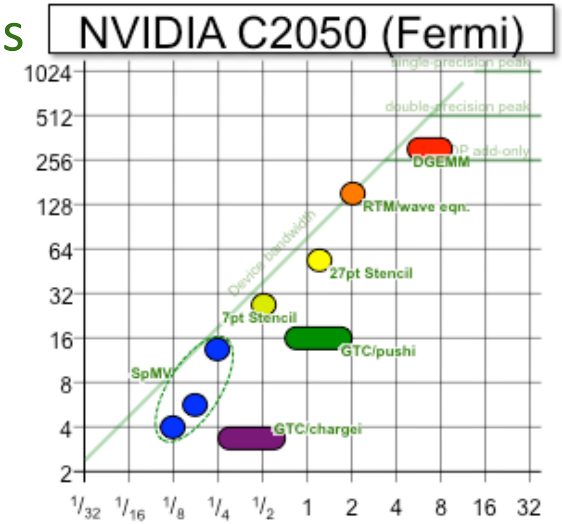
- **Collect and distill:** performance/energy/timing data
- **Identify and report bottlenecks:** through summarization/visualization
- **Provide mechanisms:** for autonomous runtime adaptation

Novelty: Automated runtime introspection

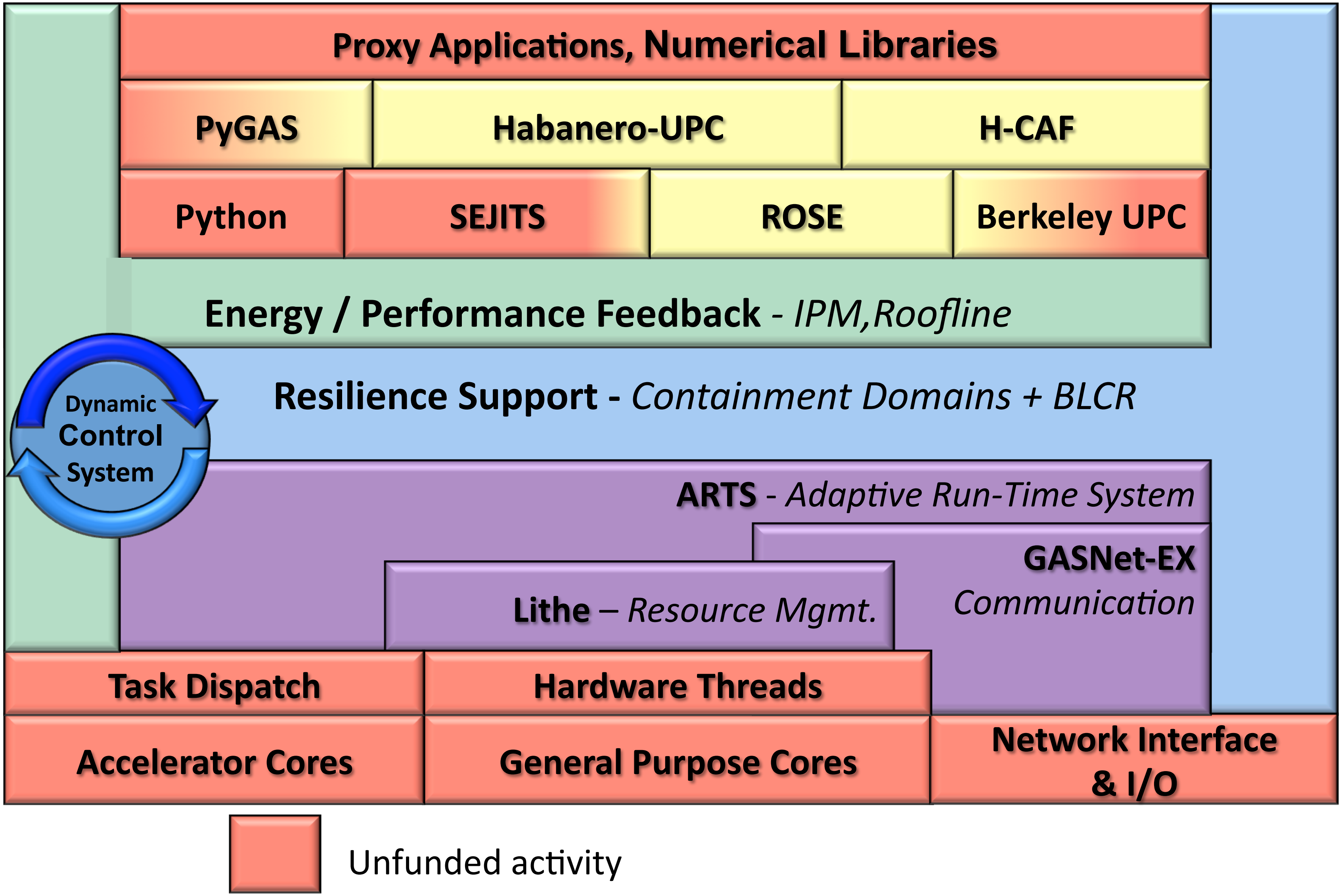
- **Provide monitoring:** power / network utilization
- **Machine Learning:** identify common characteristics
- **Resource management**: including dark silicon

Leverage: Performance / energy counters

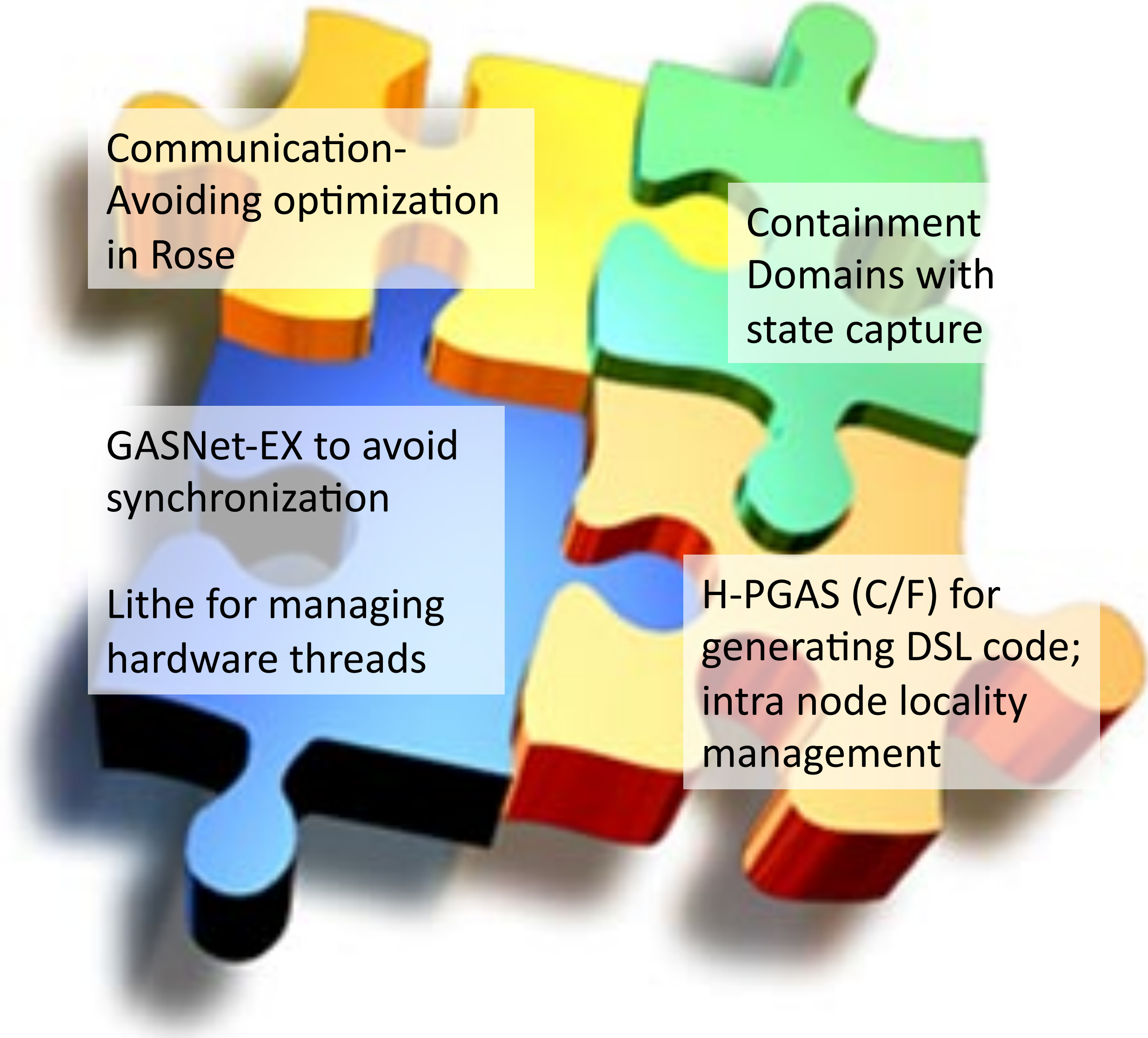
- **Integrated Performance Monitoring (IPM)**
- **Roofline formalism**
- **Performance/energy counters**



DEGAS Software Stack



DEGAS Pieces of the Puzzle



Communication-Avoiding optimization in Rose

Containment Domains with state capture

GASNet-EX to avoid synchronization

Lithe for managing hardware threads

H-PGAS (C/F) for generating DSL code; intra node locality management

Team Members


Vivek Sarkar


Kathy Yelick


John MC

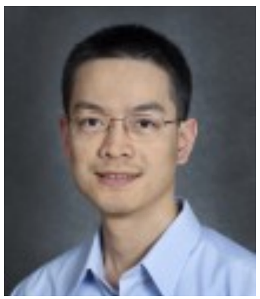

Costin Iancu


Paul Hargrove


John Shalf


Dan Quinlan


Brian VS

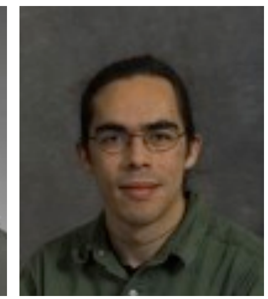

Yili Zheng

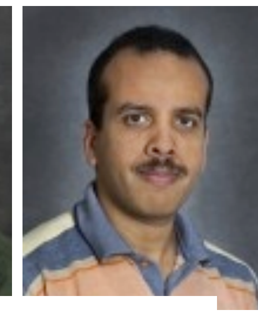

Mattan Erez


Lenny Oliker


Jim Demmel


Krste Asanovic


Eric Roman


Khaled I.


Tony Drummond


Erich Strohmaier


Armando Fox


Steve Hofmeyer


Surendra Bayna


David Skinner


Frank Mueller


Sam Williams

DEGAS Retreats Highlight and Encourage Integration

- **Semi-annual 2-day meeting of entire team, stakeholders**
  - Application and Vendor Advisory groups
- **Updates on progress, open problems, plans**
- **Demos showing integration of tools and driving applications**
- **Enforces teamwork, demos for milestones and progress metrics**
- **Feedback from team and stakeholders to refine goals and effort**
- **Long tradition of retreats at UC Berkeley**
  - Many successful large projects (from RAID to ParLab)