DEGAS: Dynamic Exascale Global Address Space

Katherine Yelick, Erich Strohmaier, Paul Hargrove, Costin Iancu, Eric Roman, John Shalf, Brian van Straalen, Sam Williams — *LBNL*
Vivek Sarkar, John Mellor-Crummey — *Rice University*
James Demmel, Krste Asanoviç — *UC Berkeley*
Mattan Erez — *UT Austin*
Dan Quinlan — *LLNL*

DEGAS

DEGAS works to enable the broad success of Exascale systems through a unified programming model that is productive, scalable, portable, interoperable, and meets the unique Exascale demands of energy efficiency and resilience.

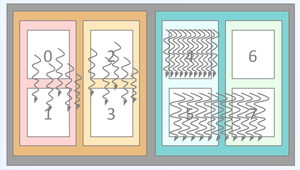
Hierarchical Programming Models

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

Novelty: High-level constructs for Hierarchical PGAS (H-PGAS)

- Hierarchical distributed data structures
- Hierarchical pointers
- Parallel multidimensional loops
- Mutual exclusion and resilience through isolation and containment
- Semantic guarantees for concurrency bugs



Two languages demonstrate DEGAS programming model

- **Hierarchical Co-Array Fortran (CAF):** CAF for on-chip scaling and more
- **Habanero-UPC:** Habanero's intra-node model with UPC's inter-node model

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

Energy/Performance Introspection for the Policy Engine

Challenge:

- **Policy engine needs real-time information on system energy and performance in order to make good scheduling decision**

Approach:

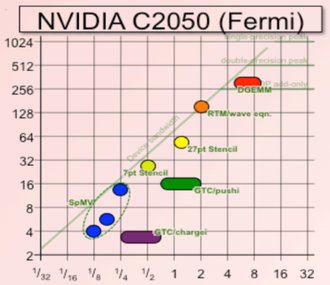
- **Collect and distill** performance/energy/timing data
- **Identify and report bottlenecks** through summarization
- **Provide mechanisms** for autonomous runtime adaptation

Novelty:

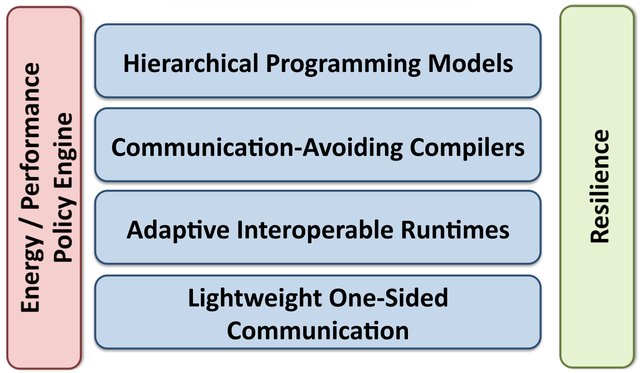
- Provides monitoring of power / network utilization
- Uses Machine Learning to identify common characteristics
- Manages Resource including dark silicon

Leverage:

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



DEGAS Components



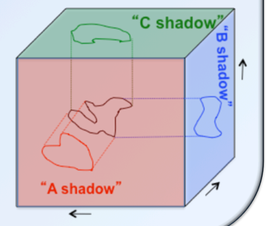
Communication Avoiding Compilers

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

- **Parallelism**: generate code that exploits massive task and data parallelism
- **Heterogeneity:** tailor compilation to core functional & performance characteristics
- **Energy efficiency:** minimize data movement and support runtime adaptation
- **Programmability:** manage details of data layout, heterogeneity, and resilience
- **Scalability:** hide communication and synchronization costs using asynchrony

Novelty: Map Hierarchical PGAS (H-PGAS) onto Exascale platforms

- Model-guided code generation for heterogeneous cores
- Integrated support for locality, resilience, heterogeneity, and adaptivity
- Communication Avoiding (CA) strategies
- Dynamic code generation for distributed data structures



Le**verage: Compiler infrastructures and code generation**

- ROSE for H-CAF and CA; BUPC and Habanero-C
- Python interpreter, SEJITS, and possibly ROSE for PyGAS
- Theory of CA-algorithms

Resilience

Goal: Provide a resilient environment for PGAS applications

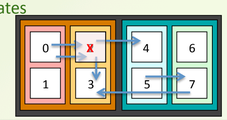
- Applications should be able to customize resilience to their needs
- Resilient runtime that provides easy-to-use mechanisms

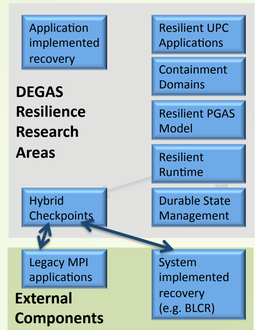
Containment domains: Prototype abstraction for resilience

- PGAS resilience consistency model
- Directed and hierarchical preservation
- Allows for global and partial/localized recovery
- Allows algorithmic and/or system-implemented detection, elision, and recovery

Leverage: prior approaches and expertise

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





ARTS: Adaptive Runtime System

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

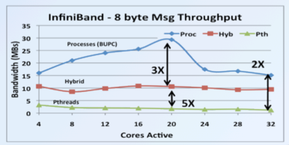
- Reactive and proactive control for utilization
- Hybrid programming and fine grained asynchrony support
- Library interoperability support by sharing of hardware threads

No**velty:**

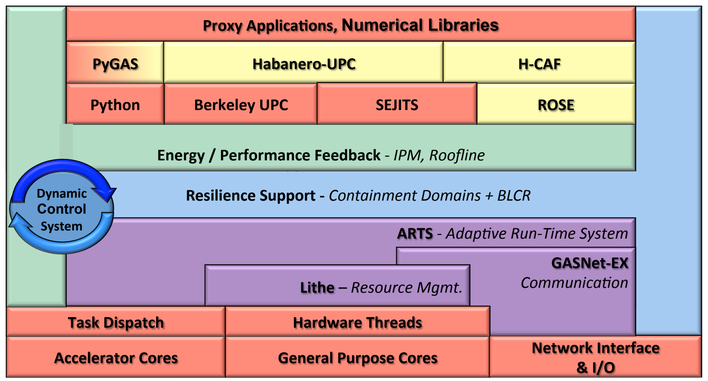
- Runtime annotated with performance history/intentions for adaptation
- Performance models guide runtime optimizations, specialization
- Hierarchical resource management
- Tunable control: Locality / load balance

Le**verages: Existing runtimes**

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



DEGAS Software Stack



Lightweight One-Sided Communication

Goal: Maximize bandwidth use with lightweight communication

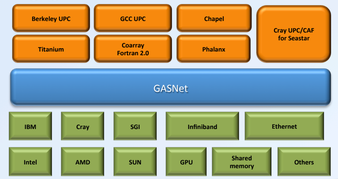
- One-sided communication: to avoid over-synchronization
- Active-Messages: for productivity and portability
- Interoperability: with MPI and threading layers

Novelty:

- Congestion management: for 1-sided communication with ARTS
- Hierarchical: communication management for H-PGAS
- Resilience: globally consist states and fine-grained fault recovery

Leverage: GASNet extended as above

- Major changes for on-chip interconnects
- Annual releases; used in production
- Enabler of several research projects
- Each network has unique opportunities



This work is supported by the X-Stack program funded by the U.S. Department of Energy Office of Advanced Scientific Computing Research