DEGAS: Dynamic Exascale Global Address Space

Katherine Yelick, Surendra Byna, Paul Hargrove, Steven Hofmeyr, Costin Iancu, Khaled Ibrahim, Leonid Oliker, Eric Roman, John Shalf, Erich Strohmaier, Samuel Williams, Yili Zheng (LBNL); Vivek Sarkar, John Mellor-Crummey (Rice University); James Demmel, Krste Asanoviç (UC Berkeley); Mattan Erez (UT Austin); Dan Quinlan (LLNL)

The Dynamic, Exascale Global Address Space programming environment (DEGAS) project will develop the next generation of programming models, runtime systems and tools to meet the challenges of Exascale systems. We will develop a new set of programming concepts based on a hierarchical model of parallelism and data locality, hierarchical fault containment/recovery for resilience, introspective dynamic resource management, demonstrate them using extensions to existing languages, and evaluate their utility for applications. Our solution will address the following key challenges posed by exascale systems:

- Scalability: Efficient communication (extended GASNet) and synchronization mechanisms combined with compiler (ROSE) and runtime optimizations to minimize both.
- Programmability: Rich set of programming constructs based on a dynamic, resilient Partitioned Global Address Space (PGAS) model, demonstrated in multiple language dialects (C and FORTRAN).
- Performance Portability: Non-invasive profiling (IPM), deep code analysis (ROSE) and a dynamically Adaptive RunTime System (ARTS).
- Resilience: Containment Domains and state capture mechanisms and lightweight, asynchronous recovery mechanisms.
- Energy Efficiency: Runtime energy adaptation and communication-optimal code generation.
- Interoperability: Runtime and language interoperability with MPI, OpenMP and libraries (Lithe).

The DEGAS team will work with Co-Design centers to drive the programming construct design, combined with information about hardware platforms as it emerges. We will also leverage ongoing discussions with other application and vendor stakeholders as well as mainstream language standards groups, augmented with advisory committees and semi-annual retreats involving broad representation from all three groups.

Our approach focuses on a vertically integrated programming and execution environment that incorporates the latest algorithmic approaches and application structures to effectively service ultra-scale science and energy applications. The primary focus areas of DEGAS are shown in Figure 1 along with the proposed integrated software stack.

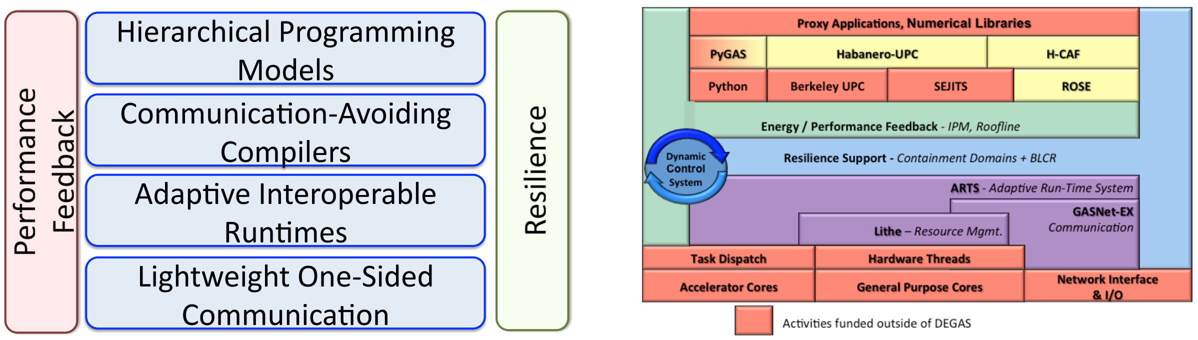
Performance Feedback
Hierarchical Programming Models
Communication-Avoiding Compilers
Adaptive Interoperable Runtimes
Lightweight One-Sided Communication
Resilience
Proxy Applications, Numerical Libraries
PyGAS
Habanero-UPC
H-CAF
Python
Berkeley UPC
SEJITS
ROSE
Energy / Performance Feedback - IPM, Roofline
Dynamic Control System
Resilience Support - Containment Domains + BLCR
ARTS - Adaptive Run-Time System
GASNet-EX Communication
Lithe – Resource Mgmt.
Task Dispatch
Hardware Threads
Accelerator Cores
General Purpose Cores
Network Interface & I/O
🟥 Activities funded outside of DEGAS

Figure 1: Organization of DEGAS research concepts (left) and software stack (right)

The proposed research addresses key technical challenges throughout the software stack and will results in an integrated stack that can be used through multiple language interfaces that interoperate. The key research thrust areas in DEGAS include:

Hierarchical Programming Constructs for Locality and Parallelism: We will develop a new programming constructs based on a hierarchical model of parallelism and data locality that extends the Habanero Hierarchical Place Tree (HPT) from a single node to a Distributed Hierarchical Place Tree (DHPT) model in a PGAS setting. The computations that can be mapped on a DHPT will be specified using multi-paradigm parallelism, including asynchronous tasks and task teams, and general multidimensional distributed parallel loops. Further, a rich set of scalable synchronization constructs will be explored to enable different forms of coordination among computations, including point-to-point synchronization with asynchronous collectives.

Language Designs: We will express the programming constructs listed above as extensions to three existing languages --- Fortran, C and possibly Python. Our goal is to demonstrate their utility for high-end applications, while providing a path to broader adoption of these advanced concepts for mainstream language standards. The hierarchical Fortran extensions (HCAF) will be based on CAF 2.0, which offers a ready migration path for existing Fortran applications. The C extensions (Habanero-UPC) will be based on UPC and Habanero-C, and will provide the quickest path for evaluating mini-applications on new hardware with specialized processors. We also plan some exploratory work based on Python (PYGAS) as an option for high level programmability.

Language Implementations (ROSE): We will use the ROSE compiler infrastructure to provide automated analysis to support dynamic asynchronous execution provided by ARTS, automated model generation and code instrumentation, and fault resilience/recovery features of containment domains. A new class of communication-avoiding (and in some cases provably communication-optimal) algorithms will be available through libraries and from advanced compiler transformations.

Communication Layer (GASNet-EX): Existing partitioned global address space (PGAS) environments presume a flat, homogeneous distributed memory model, whereas future systems are expected to be increasingly heterogeneous with disjoint memory spaces. We will enhance PGAS concepts and advanced programming constructs by extending an existing Global Address Space infrastructure (GASNet) for internode communication to hierarchical/heterogeneous memories. This work will deliver significant innovations for the execution model within a node, allowing performance portability across varying architectural solutions.

Resilience (BLCR+CD): Current bulk-synchronous approaches to checkpoint/restart for resilience are not scalable, while the likelihood of faults at exascale will increase dramatically. We will therefore deliver foundational contributions in fault-resilience using containment domains (CD), while developing advanced asynchronous rollback technology for PGAS languages. Our work will leverages and extend the low-level mechanisms from Berkeley Lab Checkpoint/Restart (BLCR), which has a history of delivering production-quality, deployed software infrastructure for fault isolation, state preservation, and recovery.

Dynamic/Adaptive Runtime (ARTS/Lithe): Existing static runtime systems fail to meet the needs of increasingly dynamic algorithms and execution environments. Our work will use a composable interface for parallel libraries called Lithe to manage and schedule resources, while enabling a smooth transition from legacy programming models. Lithe will be integrated into our novel Adaptive RunTime System (ARTS) that uses instrumentation and model-based prediction for dynamic adaption to changing runtime conditions, and automatic code optimization for data locality and energy efficiency.

Performance Feedback and Steering (IPM/Roofline): The control system for an adaptive runtime requires information to make control decisions and detect failure conditions, but current software environments fail to provide pervasive/non-invasive monitoring. We will integrate minimally-invasive integrated performance tools (IPM) to provide always-on feedback from full-scale DOE applications. Our work will also expand and automate our Roofline performance model to enable multiple levels of bottleneck analysis across diverse architectures.