AGAS Research and Development in 2011

Bryce Adelstein-Lelbach[1], Vinay C Amatya[1,2], Maciej Brodowicz[1], Hartmut Kaiser[1,2]

[1]Center for Computation and Technology, [2]LSU Department of Computer Science

STE||AR
stellar.cct.lsu.edu

LSU CENTER FOR COMPUTATION & TECHNOLOGY

Introduction

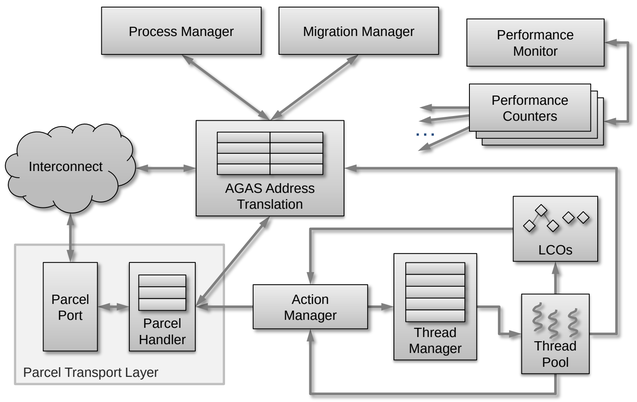
The fundamental hypothesis of our research is that traditional programming and execution models such as Communicating Sequential Processes (CSP), Message Passing Interface (MPI) and Partitioned Global Address Space (PGAS) are insufficient for combating load imbalances encountered during the runtime of certain classes of scientific applications that evolve dynamic and require adaptive management. Therefore, these applications are unable to utilize existing Petascale and planned Exascale systems. We call these scaling impaired applications **dynamic scientific applications**.

The **Systems Technologies, Emergent Parallelism and Algorithms Research (STE||AR) group** at LSU's Center for Computation and Technology (CCT) researches and develops a message-driven parallel execution model called **ParalleX**. ParalleX is an novel execution model designed to address the needs of dynamic scientific applications and Exascale systems. The key elements of ParalleX are:

- A global address space without the assumption of cache coherence, to bridge the semantic gap caused by local virtual memory boundaries.
- Fine-grained parallelism to enable latency hiding instead latency avoidance.
- Preference for moving work to data rather than moving data to work.
- Constraint-based synchronization to eliminate global barriers.
- Dynamic and heuristic resource management and task-queue based scheduling, utilizing information obtained via runtime introspection to actively manage applications.

STE||AR has developed the first implementation of the ParalleX model. **High Performance ParalleX (HPX)** is an open-source, feature complete ParalleX runtime system written in C++ (see **Figure 1**, which outlines the architecture of HPX), targeting conventional clusters and operating systems. HPX provides an extensible framework for developing parallel applications utilizing the ParalleX model.

Figure 1: HPX Architecture



My work in the STE||AR group focuses on the research and development of the **Active Global Address Space (AGAS)**. AGAS is a set of addressing services that form a hierarchical namespace that spans all resources in a particular computation. AGAS aims to ease the difficulty of programming across local virtual memory boundaries by exposing a global addressing system that can be used to address both local and remote objects. AGAS is an extension of the PGAS model used by frameworks such as X10, Chapel, UPC and Co-Array Fortran. Unlike PGAS, which statically partitions a global address space into logical blocks, AGAS supports the dynamic addition or subtraction of hardware resources and the migration of globally named objects.

This poster outlines major AGAS developments from 2011. In addition to expanding our understanding of the AGAS model, these developments have realized substantial usability and performance benefits for HPX applications.

Figure 2: Depiction of AGAS Bulk Cache Entries

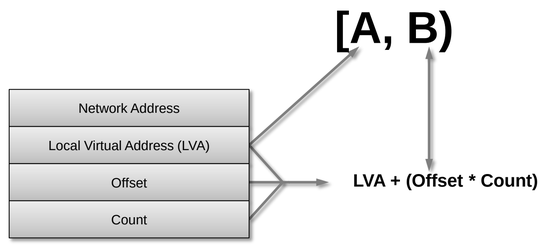


Figure 3: Effect of AGAS Caching on Walltime

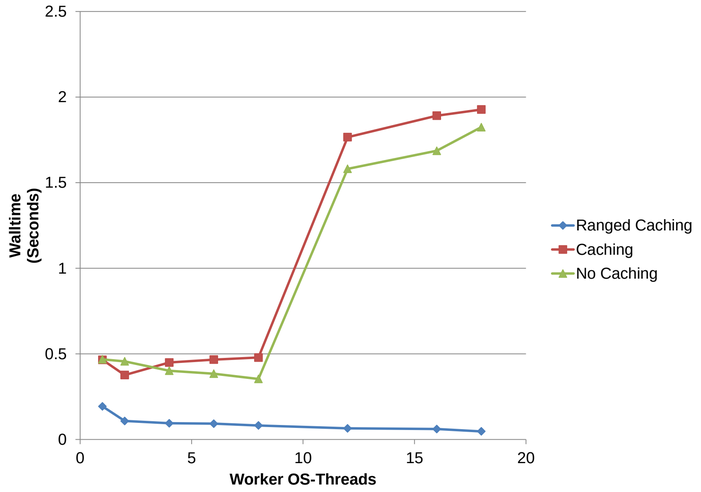
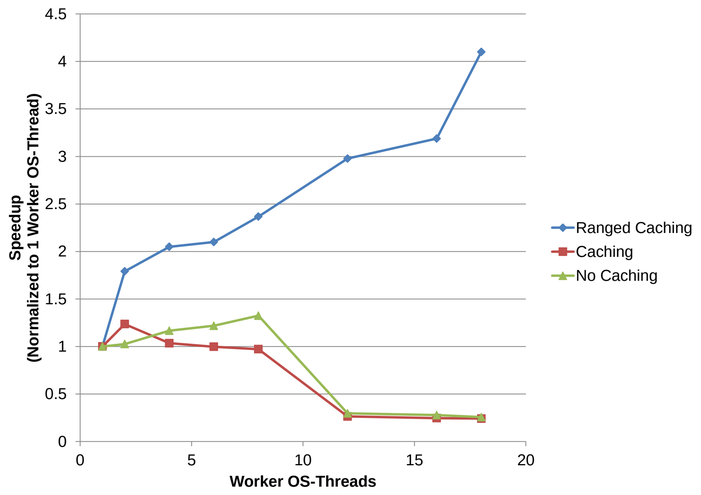


Figure 4: Effect of AGAS Caching on Speedup



AGAS Caching

AGAS's primary function is to translate **global identifiers (GIDs)** to **global addresses**. A global address is the set of information required to remotely access an object; an object with a global address is called a **global object**. GIDs are unique identifiers that reference global objects.

Global addresses managed by AGAS require the creation of address translation tables. These tables are stored on a subset of the available **localities** (in conventional clusters, a compute node is a locality). The locality or localities hosting AGAS data are called **AGAS servers**. All localities that are not AGAS servers are called **hosted localities**.

Hosted localities can resolve GIDs by querying AGAS servers. The full resolution of a GID from a hosted locality requires communication across localities, which implies multiple network turnarounds in HPX. To reduce network traffic in HPX, we have implemented software caches for AGAS services, and we are investigating hardware cache solutions via FPGAs.

Initially, HPX's AGAS cache stored single address entries. In large computations which frequently reference millions of global objects throughout their execution, this naïve method of caching leads to cache thrashing. To alleviate this issue, **range-based caching** was introduced. Range-based caching is based on HPX's pre-existing support for the bulk allocation of GIDs and bulk registration of global objects. These bulk operations use contiguous blocks of GIDs and contiguous blocks of local virtual memory. Additionally, each bulk registration operates on a particular type of object (taken as a parameter to the operation) with a fixed data size. Range-based caching optimizes the caching of global addresses that fall into these blocks by storing **bulk cache entries** which describe entire blocks. **Figure 2** depicts the information needed to resolve a block. The interval [a, b) in Figure 2 is local virtual memory spanned by the block. A similar scheme is used for storing the range of contiguous GIDs associated with block.

Range-based caching allows a relatively small cache to store very large regions of the global address space. This greatly reduces cache evictions and cache misses, reducing HPX's overhead and improving HPX's overall scalability. These performance improvements are demonstrated by results from a standard HPX benchmark, shown in **Figure 3** (overhead reduction) and **Figure 4** (improvement in scalability). The benchmark used was the HPX Eager Future Overhead (EFO) test, which is described in more detail in one of our publications, Adaptive Mesh Refinement for Astrophysics Applications with ParalleX (arXiV:1110.1131, section V, subsection A). EFO is part of the main HPX codebase, and can be obtained from our website, stellar.cct.lsu.edu.

AGAS V2

AGAS is the oldest HPX subsystem, and was originally written more than five years ago. As such, the first AGAS implementation contained a number of design flaws which ultimately required a complete rewrite of the AGAS subsystem. We call the original AGAS implementation **AGAS V1**, and the new implementation **AGAS V2**.

AGAS V1's central flaw was the use of a communication layer separate from the primary HPX message protocol, the **Parcel Transport Layer**. This out-of-band AGAS transport only supported synchronous communication, preventing the overlapping of network latencies when making AGAS queries (see **Figure 5**). AGAS V2 uses the Parcel transport layer, facilitating latency hiding and one-sided communication when making AGAS requests (see **Figure 6**). AGAS V2 has greatly improved the overall scalability of HPX.

Figure 5: AGAS V1 Communication Layers

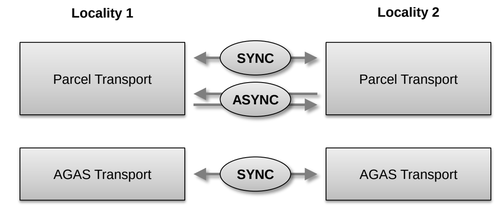
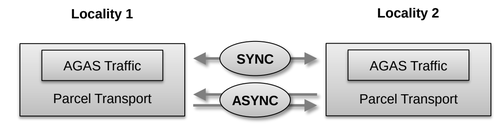


Figure 6: AGAS V2 Communication Layers



- NSF Grants 1117470, 1048019, 1029161
- DARPA UHPC Funding
- LONI Allocation loni_hpx
- CCT HR, PR, Inventory and Sysadmins
- Steven Brandt
- Thomas Heller