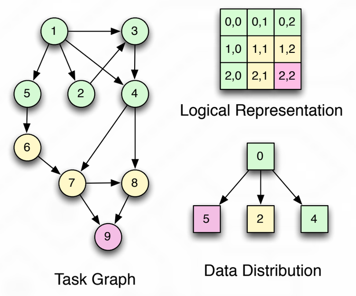
Active Global Address Space (AGAS)

Global Virtual Memory for Dynamic Asynchronous Many-Tasking (AMT) Runtimes

Abhishek Kulkarni, Andrew Lumsdaine (advisor)
Center for Research in Extreme Scale Technologies (CREST)
Indiana University

Motivation

- Traditional bulk-synchronous parallel runtimes do not excel at irregular computation patterns present in big-data and adaptive execution.
- Asynchronous many-tasking (AMT) models, characterized by lightweight tasks concurrently operating on global data, are dynamic and adaptive in nature. When work follows data, global load-balancing naturally takes the form of dynamically balancing global data.
- PGAS is less well-suited for event-driven and active-message execution where ephemeral computation is performed on global data. Static nature of PGAS restricts the system's ability to load balance computation and communication.

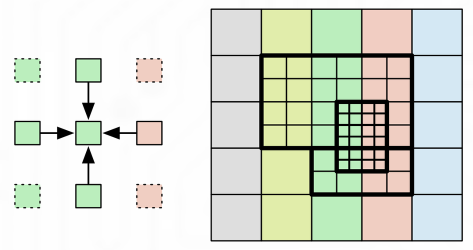


- Examples of applications demonstrating runtime imbalance: dynamic graph algorithms, adaptive mesh-refinement, etc.
- Scalability and performance depends on initial data distribution and dynamic runtime-supported load and data balancing.

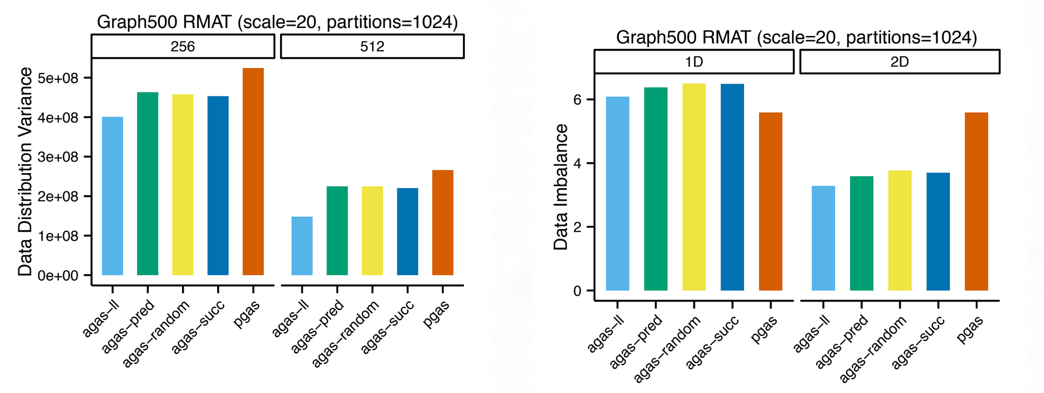
- We propose global virtual memory for AMT runtimes using an **active global address space (AGAS)**.
- The active global address space is "active" in two senses. It is virtualized and allows memory to dynamically relocate; its usage is primarily through the use of active messages.

Problem: Dynamic Load Imbalance

- Adaptive Mesh Refinement (AMR) affected by dynamic runtime load imbalance.
- Sub-optimal initial partitioning of data can create communication and computation hotspots that cause performance degradation—even after employing the common latency-hiding optimizations.



Global data dependencies of 5D-stencil operation in AMR.



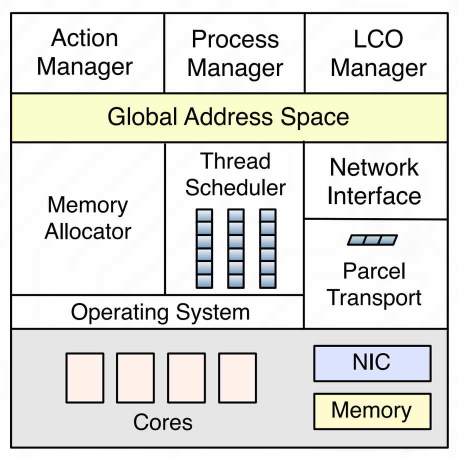
(a) Mean Variance in partition distribution for static (pgas) vs dynamic (agas) data distribution schemes.

(b) Data Imbalance for 1D vs 2D distribution of neighboring vertex list for 1K nodes.

Arbitrary, user-defined dynamic data distribution schemes in AGAS reduce load imbalance. Different load-aware data distribution policies evaluated on Graph500 graphs: least-loaded (agas-ll), predecessor (agas-pred), successor (agas-succ) or random (agas-random).

High-Performance ParalleX (HPX-5)

- HPX-5 is a library-based implementation of the ParalleX model in C.
- The HPX-5 runtime system features:
  - ✓ Cooperatively scheduler lightweight threads
  - ✓ Inter-thread synchronization (LCOs)
  - ✓ Active messages (via Parcels)
  - ✓ Global Address Space (AGAS)
- Programming interfaces for global address manipulation, translation, and allocation; parcels, lightweight threads, and LCOs.
- Parcel transport using flexible one-sided or two-sided networking interfaces.
- Parcels target global addresses, carry payload data, identify message handlers for execution, and specify continuation addresses.

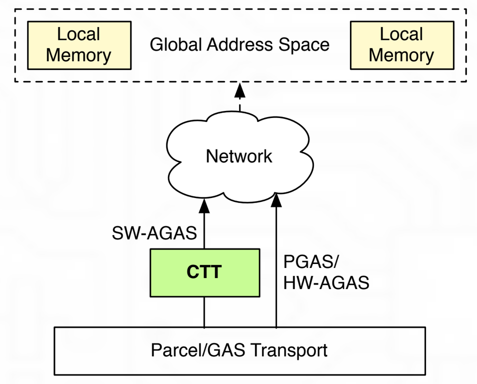


HPX-5 Architecture.

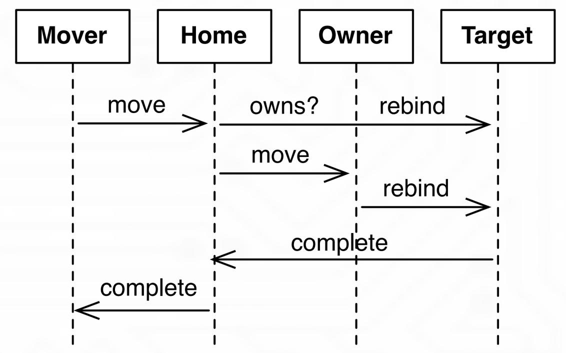
Parcel delivery spawns a cooperative lightweight thread executing the parcel's action. Lightweight threads have the ability to acquire resources, send additional parcels, and wait for asynchronous events.

Active Global Address Space (AGAS)

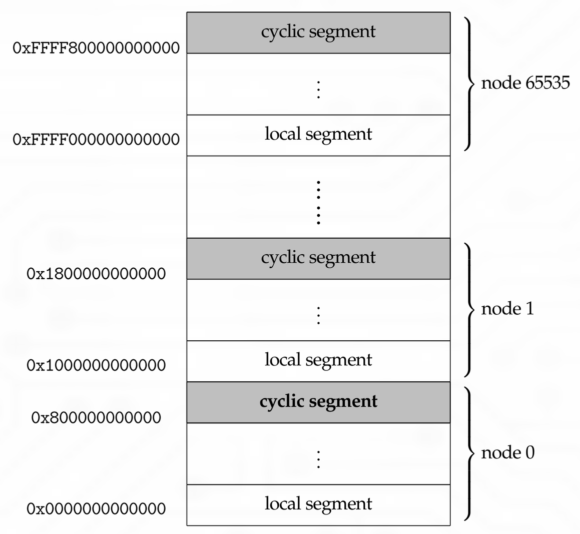
- Linear, byte-addressable, global address space organized into distinct logical blocks consisting of relocatable units (known as chunks).
- Distributed data-structures with arbitrary distribution allocated in global virtual memory.
- Dynamic remapping capability exposed through the introduction of an explicit "move" operation in the programming model.
- HPX-5 AGAS features:
  - ✓ Data migration through explicit move operations
  - ✓ Data rebalance through "data-stealing"
  - ✓ Co-data (coupling computation and data) actions
  - ✓ Global data attributes (write-once, pinned, etc.)
- The remap operation is guaranteed to be consistent and linearizable, where in-flight parcels addressed to a "moving" target are resent until they arrive at the target.
- Maintaining a scalable high-performance virtual global address space using distributed memory hardware is challenging. Software implementation of AGAS presently uses the two-sided networking interface.



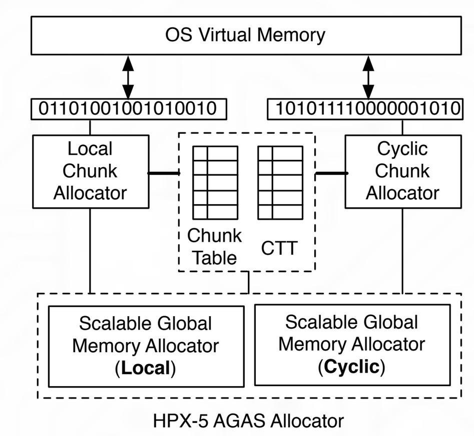
Software AGAS uses a chunk translation table (CTT) to translate GVA to GPA. In the hardware implementation, this table is maintained in the switch.



Move/remapping protocol to move a global chunk from an owner node to a target node.



(a) Global Virtual Address Space in AGAS.

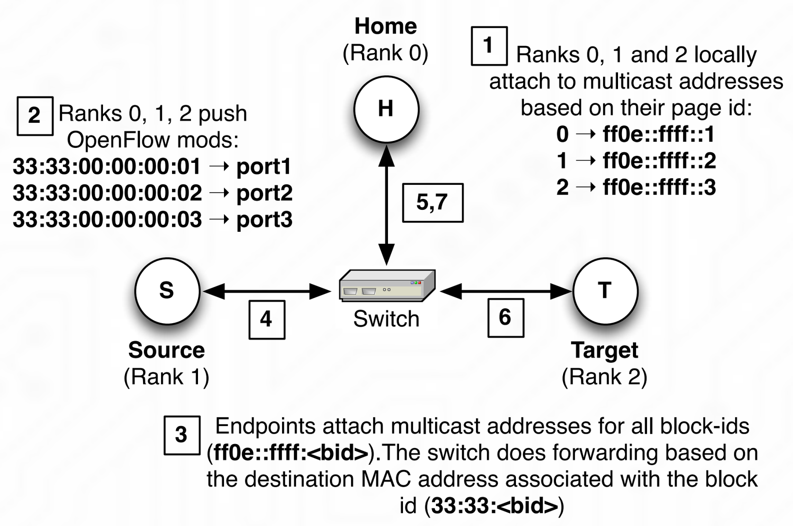


(b) Scalable AGAS allocator in HPX-5. The heaps are divided for local and cyclic memory, which itself is managed by a scalable, concurrent memory allocator.

HPX-5 AGAS architecture. Scalable, concurrent memory allocators are used for backing global memory. The per-node translation table with optional caching reduces global address translation overheads.

Network-Managed Global Virtual Memory

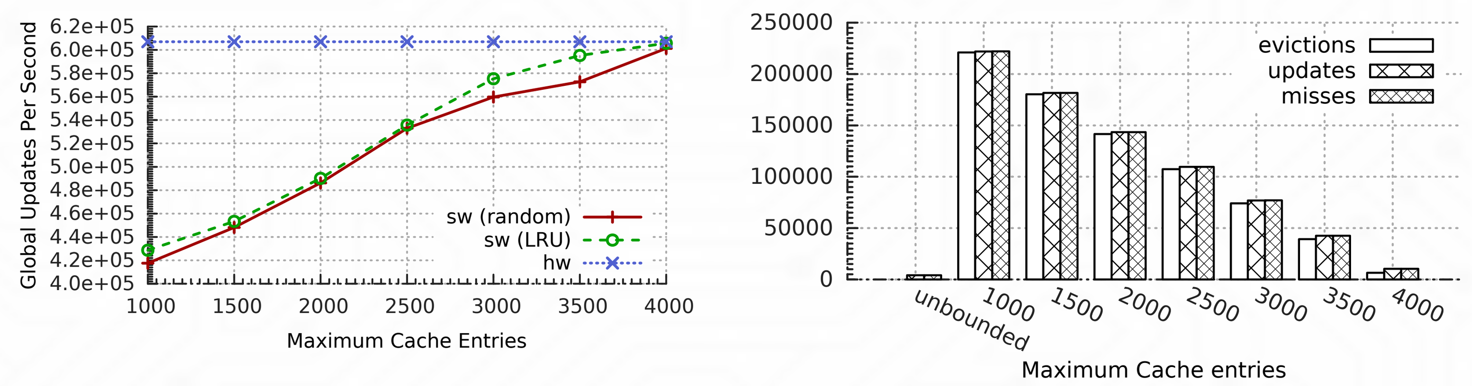
- Network-assisted AGAS leverages the capabilities of the network fabric to manage addressing rather than software at the endpoint hosts.
- Uses a GASNet conduit with IB multicast over unreliable datagram (UD) which allows receivers to accept packets from any node in the network.
- Address-to-port mapping in the switch determines the current owner of the global address.
- Storing the page table in the network switch makes loopback optimization challenging.



Use of multicast addressing with RoCE. Software-Defined Networking (SDN) used to maintain global virtual address mappings in the switch. This allows the runtime to dynamically route put and get operations to the appropriate rank based on the global address.

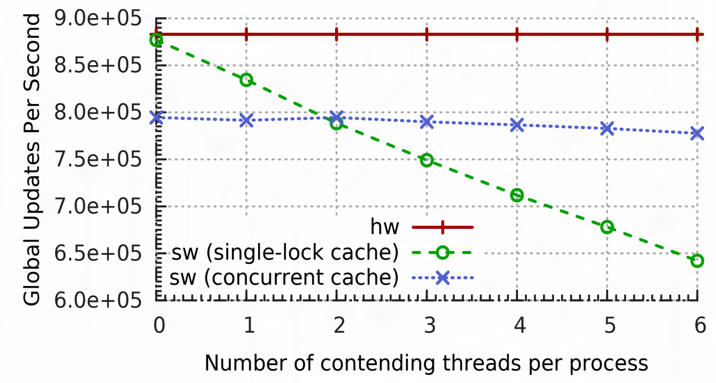
Benefits of Network-assisted AGAS

- Messaging overheads can be reduced through the use of a software translation cache
- However, software caching incurs storage and synchronization overheads
  - ✓ Page table mappings have to be evicted as pages are remapped
- Sequential direct-mapped cache extended with two cache replacement policies: random and LRU (least recently used)
- Bounding the cache size adds extra cache eviction logic and incurs capacity cache misses



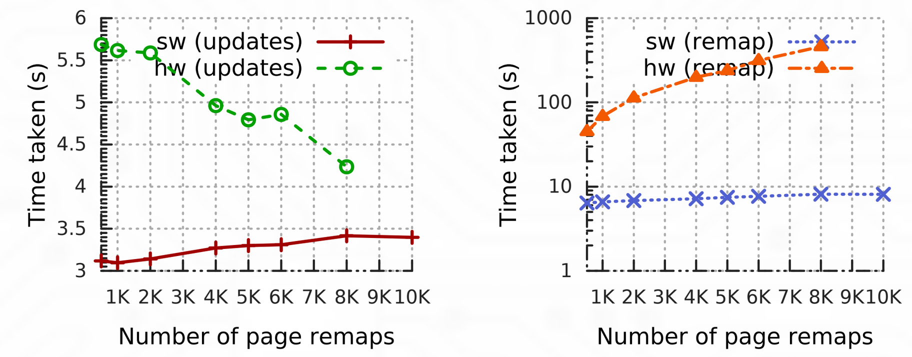
Effect of bounded storage and cache replacement policies on the GUPS random access microbenchmark at 192 cores. The figure on the right shows LRU-cache statistics at rank 0 for 2^26 random updates of a 2^21 words global table.

- Limiting the software translation cache size degrades put/get performance.
- Each cache entry was only 4 bytes resulting in a 1:1000 ratio of metadata overhead per addressable page.
- In practice, cache entries encode other page attributes page, and an overhead of mere 0.1% can potentially limit the scalability of the runtime to 1000 processors.
- More evictions lead to increased cache misses. None of the cache replacement policies scale in comparison to the network-assisted implementation.



Performance degradation of global updates due to thread contention. The concurrent cache is 25% slower than the hardware directory (cache-bypass) solution.

Impact of dynamically moving pages (i.e., "remapping" blocks)



(a) The update time measures the time it required to perform global updates ignoring the overhead of active remap operations.

(b) The remap phase measures the time to move the data associated with the *GVA*, and update its global mappings.

- The GUPS microbenchmark with a global table consisting of 4 pages distributed across 192 cores.
- As page movement frequency increases, the software approach takes increasingly longer. In contrast, the direct lookups afforded by the hardware case ensure a constant, or improved, runtime.

Conclusions

- The HPX-5 runtime system provides the dynamic, adaptive features necessary for the efficient execution of large-scale irregular applications.
- To manage both locality and load concerns for the programmer, HPX-5 provides global virtual memory using an active global address space (AGAS) where data can be dynamically migrated around in the system.
- Network-assisted AGAS incurs reduced common-case overheads, however hardware remapping remains expensive due to the cost of flow programming on present-day switches.

Further Information

Please contact *adkulkar@crest.iu.edu for more details.*

CENTER FOR RESEARCH IN EXTREME SCALE TECHNOLOGIES
INDIANA UNIVERSITY
Pervasive Technology Institute

SCHOOL OF INFORMATICS AND COMPUTING
INDIANA UNIVERSITY
Bloomington