GASNet-EX at Lawrence Berkeley National Lab (http://gasnet.lbl.gov)

- GASNet-EX: communications middleware to support exascale clients
  - One-sided communication – Remote Memory Access (RMA)
  - Active Messages - remote procedure call
  - Implemented over the native APIs for all networks of interest to ECP
- GASNet-EX is an evolution of GASNet-1 for exascale
  - Retains GASNet-1's wide portability (laptops to production supercomputers)
  - Provides backwards compatibility with GASNet-1 clients
  - Focus remains on one-sided RMA and Active Messages
  - Reduces CPU and memory overheads
  - Improves many-core support
- GASNet-1 clients include:
  - Multiple UPC and CAF/Fortran08 compilers
  - Stanford's Legion Programming System
  - Cray Chapel Language
  - OpenSHMEM Reference Implementation
  - Omni XcableableMP Compiler
- GASNet-EX clients include:
  - ECP ST: UPC++ and Legion; and PaRSEC exploring
  - ECP AD: ExaBiome exploring
  - non-ECP: Cray Chapel exploring
- GASNet-EX augments and enhances GASNet-1
  - Enhancements address needs of modern asynchronous PGAS models
  - Interfaces adjusted for improved scalability
  - Features critical to UPC++ are being co-designed
  - Using input from Legion and Cray Chapel, who plan to adopt the new APIs
- Features delivered in FY17 and so far in FY18 include:
  - "Immediate mode" injection to avoid stalls due to back-pressure
  - Explicit handling of local-completion (source buffer lifetime)
  - New AM interfaces, for instance to reduce buffer copies between layers
  - Vector-Index-Strided for non-contiguous point-to-point RMA
  - Remote Atomics, implemented with NIC offload where available
- Features to deliver in remainder of FY18 include:
  - Teams and non-blocking capabilities
  - Dependent operations to control ordering of in-flight operations
  - Features for FY19 and beyond may include:
    - Offset-based addressing
    - Multiple endpoints/segments, for instance to enhance multithreading support
    - Support for "out-of-segment" remote addresses

 Highlights from Current Work

- Example of EX interface updates: RMA Put
  ```c
  gex_RMA_PutSB(gex_TM_t tm, gex_Rank_t rank, gex_Addr_t dest_addr,
  void *src_addr, size_t nbytes, gex_Event_t *loc_opt, gex_Flags_t *flags);
  gex_event_t return type introduces events to generalize GASNet handles.
  tm argument adds team (ordered sets of ranks), into which rank indexes.
  gex_Addr_t type will enable offset-based addressing via same interface.
  lo_opt argument introduces explicit control over local completion, generalizing
  the bulk/non-bulk interfaces of GASNet-1.
  flags argument provides extensibility. For instance:
  - To select new optional behaviors (e.g., immediate mode and offset-based addressing)
  - To provide assertions regarding the arguments (e.g., to streamline the operation)

 Vector-Indexed-Strided (VIS) Interfaces for Non-Contiguous RMA

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>NETWORK</th>
<th>INDICED</th>
<th>STRIDED</th>
<th>VECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Con-1</td>
<td>Cray X1</td>
<td>11.68</td>
<td>10.68</td>
<td>12.05</td>
</tr>
<tr>
<td>Theta</td>
<td>Cray X1</td>
<td>10.53</td>
<td>7.70</td>
<td>11.10</td>
</tr>
<tr>
<td>Titan</td>
<td>Cray X1</td>
<td>7.33</td>
<td>7.21</td>
<td>8.09</td>
</tr>
<tr>
<td>Summit-D</td>
<td>Mellanox Infiniband</td>
<td>5.45</td>
<td>5.17</td>
<td>5.67</td>
</tr>
<tr>
<td>Data</td>
<td>IBM BG/Q</td>
<td>3.98</td>
<td>3.49</td>
<td>4.01</td>
</tr>
</tbody>
</table>

- Implements the Atomic Domains concept (first introduced by UPC 1.3)
  - Domains permit use of NIC offload even when coherency is managed by CPU
  - Domains are created collectively outside the critical path
  - A Domain has an associated data type and set of allowed operations
    - Domains select the best implementation for the data type and ops
    - e.g. use offload if and only if NIC implements all the requested ops
  - Example: non-blocking atomic fetch-and-add (FADD) on unsigned 64-bit integer
    ```c
    gex_event_t ev = gex_AllocEvent();
    gex_FADD_t fadd = gex_CreateFADD(ev, addr, addend, flags);
    gex_FADD_POST_fadd, 0 / (unsigned op2 | flags);
    ...
    ...
    gex_FADD(fadd); // *result = ATOMICALLY( *target += addend )
    ...// ...fadd
    ...
    gex_FreeEvent(ev);
    ...
    ...
    ```
  - flags includes optional behaviors and assertions, such as memory fences
- GASNet-EX provides a network-independent "reference implementation"
  - Uses Active Messages to perform operations using the target CPU
  - Uses GASNet-Tools for atomicity (inline assembly for numerous CPUs)
  - Specialization for Cray Aries improves performance vs. reference implementation
  - Reduces latency of inter-node FADD from 4.9us to 2.8us
    - Greatly increases throughput under contention
  - The figure above shows throughput of 1 to 8192 processes (64 per node)

GASNet-EX Performance on Cray Aries

OLCF SummitDev (single-rail only)

- IBM S22LC
- Mellanox Infiniband EDR
- GASNet-EX ibv-conduit
- IBM Spectrum MPI 10.1.0.4
- Node configuration:
  - 2x 10core 3.5GHz IBM POWER8
  - 4x NVIDIA Tesla P100 GPU
  - 256 GB DDR4
- IBM X C++ for Linux, V13.15 (5725-C73, 3765-JOB)
- System software:
  - Linux 3.10.0-0.14.21.47.50244
  - libibverbs 1.2.1
  - ib 12.17.1016

GASNet-EX Performance on InfiniBand

UCP++ and GASNet Support for Exascale Apps and Runtimes
Scott B. Baden (PI) and Paul Hargrove (co-PI)