### Scalable Data Race Detection

- **Active Testing**: Leverage program analysis to make testing quickly find real concurrency bugs
  - Phase 1: Use imprecise static or dynamic program analysis to find abstract states where a potential concurrency bug can happen (Race Detector)
  - Phase 2: Directed testing based on the abstract states obtained from phase 1 (Race Tester)
- **THRILLIE** – THRead Interposition Library and Lightweight Extensions: Active testing framework for UPC
- **Implementation of race detector and tester for programs written in shared memory style**
  - Supports shared memory accesses using shared pointers and bulk transfers (upc_mempy)
  - Tracks fine-grained synchronization (locks) and bulk synchronization (single- and split-phase barriers)

### UPC Collectives Library 2.0

- **Goals**
  - Improve usability and enable performance optimization over the previous UPC 1.2 collective specification
  - Chart a path towards MPI interoperability
- **Library-Only Approach**
  - Collectives operate on multi-valued shared objects
  - Ordering of non-blocking collectives
    - Execution order is defined by collective start
    - Allow users to wait for completion in arbitrary order
- **Features**
  - Equivalents to some popular MPI collectives
    - Teams for collectives (similar to MPI communicators)
    - Allreduce, Reduce and Reduce-scatter
  - “v” versions of collectives (e.g., allgather and alltoall)
  - Non-blocking collectives (with and without waiting handles)
- **Implementation**
  - Blocking collectives mapped well to MPI collectives
  - Berkeley and IBM will provide reference implementations based on GASNet and PAM, respectively
  - Applications being developed with the new collectives (e.g., communication-avoiding 2.5-D matrix multiplication and Choleskey Factorization (team broadcast and team reduce))
  - Multi-dimensional FFT (team alltoall)
  - Breath-First Search (Graph500) (allgather and alltoall)

### Optimizations for scalability

- Efficient data structures for **memory ranges**
- Interval Skiplists

### Lock Tries

- Minimize Communication
- Prune all but **weakest** shared access information
- Coalesce queries to barrier boundaries
- Sampling with Exponential Backoff

### Scalability results on large cluster (up to 1024 cores)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>LoC</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>guppie</td>
<td>227</td>
<td>2.094s</td>
</tr>
<tr>
<td>knapsack</td>
<td>191</td>
<td>2.099s</td>
</tr>
<tr>
<td>laplace</td>
<td>123</td>
<td>2.101s</td>
</tr>
<tr>
<td>javalex</td>
<td>777</td>
<td>2.982s</td>
</tr>
<tr>
<td>PT 2.3</td>
<td>2303</td>
<td>6.711s</td>
</tr>
<tr>
<td>CG 2.4</td>
<td>1939</td>
<td>3.812s</td>
</tr>
<tr>
<td>IS 2.4</td>
<td>1449</td>
<td>3.073s</td>
</tr>
<tr>
<td>MG 2.4</td>
<td>2114</td>
<td>4.895s</td>
</tr>
<tr>
<td>BT 3.3</td>
<td>9626</td>
<td>42.78s</td>
</tr>
</tbody>
</table>

### PGAS for Heterogeneous Computing

- **DARPA UHPC Echelon Project**
  - Two orders of magnitude increase in application execution energy efficiency over today’s CPU systems
  - Improve programmer productivity
  - Resilient hardware and reliable software
  - Multi-institution team led by NVIDIA
- **Phalanx Programming System for Echelon**
  - Global address space programming model with dynamic asynchronous task execution
  - Prototyped in C++ with CUDA and GASNet
  - Run on current heterogeneous GPU clusters

### Example of the new collectives interface (broadcast): upc_coll_bcast

```c
upc_coll_bcast(shared void * sendbuf, size_t sendcount, upc_coll_type sendtype,
shared void * recvbuf, int recvcount, upc_coll_type recvtype,
int root, upc_coll_team_t team, upc_flag_t flags, 
upc_handle_t * handle);
```

For the complete UPC libraries 2.0 API and detailed discussion, see "UPC Collectives Library 2.0" by George Almasi, Paul Hargrove, Gabriel Tanase, Yili Zheng in the Fifth Conference on Partitioned Global Address Space Programming Models (PGAS11).

### PGAS Research at Berkeley

- **Library 2.0** by George Almasi, Paul Hargrove, Gabriel Tanase, Yili Zheng in the Finh

### PGAS for Heterogeneous Computing

- **DARPA UHPC Echelon Project**
  - Two orders of magnitude increase in application execution energy efficiency over today’s CPU systems
  - Improve programmer productivity
  - Resilient hardware and reliable software
  - Multi-institution team led by NVIDIA
- **Phalanx Programming System for Echelon**
  - Global address space programming model with dynamic asynchronous task execution
  - Prototyped in C++ with CUDA and GASNet
  - Run on current heterogeneous GPU clusters

### Heterogeneous Phalanx Example: SAXPY

```c
main_thread(1048576) = {1048576, 1048576,
async cudaMemcpyAsync(saxpy_host, saxpy_gpu),
async cudaMemcpyAsync(saxpy_gpu, saxpy_host),
};
```

### Data Transfer Performance

- Intel Xeon 5510 with 8GB RAM and NVIDIA Fermi GPU

---

This research was supported in part by the Office of Science of the U.S. Department of Energy under contract DE-FG02-99ER25515 and by the DARPA contract HR0011-09-9-0089. This research used resources of the National Energy Research Scientific Computing Center supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231.