Native Mode-Based Optimizations of Remote Memory Accesses in OpenSHMEM for Intel Xeon Phi

Dounia Khaldi

Postdoctoral Fellow

HPC Tools Group, University of Houston

Houston, TX

What's unique about my tuning work

- **Scientific Benchmarks: STREAM, NAS IS and SP**
- **Execution mode: Native**
- **OpenSHMEM PGAS library (openshmem.org), Scalasca profiling tool**
- ***shmem_ptr:* address of a data object on a specific PE**
- **No function call → enhancing compiler optimizations**

```
shmem_int_put(target,source,B,pe);
```

↓

```
int *ptr = (int *)shmem_ptr(target,pe);
…
for (i=0; i<B; i++)
   ptr[i] = source[i];
```

Performance

- **Additional Optimizations: Vectorization and alignment**
- **Stampede Super Computer (Xeon Phi SE10P 61-cores)**
- **Improved communications:**
  - **PGAS-Microbenchmarks from University of Houston**
  - **decrease in latency by up to 60% and increase in bandwidth by up to 12x**
- **Bandwidth of STREAM Copy, Scale, Add and Triad kernels is approximately increased by 40x when we use an optimized reduction algorithm with vectorization directives (such as #pragma vector align)**
- **Improved reduction algorithms: up to 22% compared to MVAPICH and 60% compared to IMPI**
- **IS Xeon Phi performance is 3x slower than Xeon**

Insights

- On shared memory, use Load/Store instead of "fake" function calls in the compiler
- Reduction: use Recursive Doubling for small message sizes and Rabenseifner for large message sizes.

Future Work

- Extending the reduction optimizations to other collectives such as Barriers
- Automation of translating RMA calls into load/store using OpenUH for shared memory systems
- PGAS Language-based on MIC, such as Fortran Coarray