Overview

- A portable and high-performance UPC implementation, compliant with UPC 1.2 spec
- Features:
  - High performance UPC Collectives
  - Extensions for performance and programmability
    - Non-blocking memcpy functions
    - Semaphores and signaling put
    - Value-based collectives
    - Atomic memory operations
    - Hierarchical layout query
  - Compiler and runtime optimizations for application scalability

- Open source software (Windows/Mac/UNIX), installation CD available at UPC booth

Portatile Design

- Layered design, platform-independent code generation
- Supports wide range of SMPs, clusters and MPPs: x86, Itanium, Opteron, Athlon, Alpha, PowerPC, MIPS, PA-RISC, SPARC, T3E, X1, SX-6, XT3, Blue Gene, ...
- Linux, FreeBSD, NetBSD, Tru64, AIX, IRIX, HPUX, Solaris, MS Windows, Mac OS X, Unicos, SuperUX, ...
- Pthreads, Unix SysV, Myrinet, Quadrics Elan 3/4, Infiniband, IBM LAPI, Dolphin SCI, MPI, Ethernet, Cray X1 / SGI Altix shmem, Cray XT3 Portals

BUPC Runtime + GASNet

- Well-documented runtime interface, multiple UPC compilers (Berkeley UPC and Intrepid GCC/UPC)
- Debugging and tracing support
  - Performance Instrumentation Support (GASP)
  - Supports Parallel Performance Wizard (PPW)
  - Detailed communication tracing support
  - Ethnus TotalView debugger support
- Interoperability with other programming environments: UPC calls to/from C, C++, Fortran, MPI

Berkeley GASNet used for communication:
- Performance from inline functions, macros, and network-specific implementations
- Optimized Collective ops (booth poster)
- High-performance communication
  - Consistently matches or outperforms MPI
  - One-sided, lightweight semantics

UPC-to-C Translator

- Source-to-source translator, based on Open64
- Platform for experimenting with one-sided communication optimizations, as well as UPC specific optimizations: compiler, runtime, communication libraries
- Partial-Redundancy Elim. on shared ptrs
  - pointer add, load, and store
- Split-phase comm. – moves read initiations up, write completions down
- Coalesces fine-grained accesses to same struct/array

Fine Grained Programming Optimizations

Impact of optimization on CFD code developed at the Army High Performance Computing Research Center. (Hummingbird CFD simulation courtesy of A. Johnson)

Coarse Grained Programming Optimizations

Multithreading for Latency Hiding

When memory is distributed one needs
- Locality and Load Balance
- Remote synchronization for dependencies
- Latency Tolerance

Case Study - Linpack

- 2d block cyclic decomposition as in ScaLAPACK
- Cooperative multi-threading to mask dependences
- Non-blocking (remote get) transfers to mask latency
- Memory-constrained lookahead compared to none in ScaLAPACK, fixed parameter in MPS/HPPL
- Application-level scheduling to prioritize critical path

Linpack Performance

Other results:
- Itanium 2/Elan 4.1 – 2.25 TFlop/s, 78.5% of peak on 512p
- ITanium 2 1.5 GHz – 91.8% of peak on 512p Opteron 2.2GHz – 81.9% of peak