Berkeley UPC

http://upc.lbl.gov

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 memory functions
  - Semaphores and signaling put
  - Value-based collectives
  - Atomic memory operations
  - Hierarchical layout query
  - Localization (castability) queries
  - Collective deallocation functions
  - Compiler and runtime optimizations for application scalability

Open Source Software (Windows/Mac/UNIX), installation DVD available at PGAS booth (#124)

Dynamic Tasking in UPC

- New release UPC task library
- Task is defined as a task function + pointers to in/out
  - void task_func (void *in, void *out) { /* task body */ }
- Tasks are stored in the global task queue
- Task programming models
  - Parallel-for parallelism and Fork-Join parallelism
  - Dependent task graph with task synchronization
  - Performance improvement by dynamic load balancing
  - Load balancing exploits locality
  - Hierarchical chunk selection
  - Hierarchical victim selection

For more information – http://upc.lbl.gov/task.shtml

UPC-to-C Translator

- Source-to-source translator, based on Open64
- Enhances programmer productivity through static and dynamic optimizations: compiler, runtime, communication libraries

- Compile time message vectorization and strip-mining
- Runtime Analysis: communication instantiated at runtime based on system specific performance models
- Performance models designed to take system scale and load into account

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
- TotalView debugger support
- Interoperability with other programming env:
  - 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
- High-performance communication
  - Consistently matches or outperforms MPI
  - One-sided, lightweight semantics

Portability:

- System, Scale, Load

Hybrid “Task” Parallelism

- BUPC allows programs to use arbitrary combinations of Pthreads and Processes with shared memory
- Mixing Pthreads and Processes is required for:
  - Interoperability with MPI and non thread-safe libraries
  - Hybrid/hierarchical parallelism (for best performance)
- PSHM layer – Process Shared Memory
  - Shared memory comms through POSIX, SYSV or mmap()
  - Shared memory “network” for Active Messages support
- Hybrid processes/pthreads execution
- Hybrid always performs best

Bandwidth (Megabytes/second)

- Giga/tera/s, 8-bit/16/32-bit

- [Image of Bandwidth graph]