The Berkeley UPC Project

http://upc.lbl.gov

Berkeley UPC *Compiler* Features

- Translates UPC to C w/calls to Berkeley runtime
  - Fully compliant with UPC 1.1.1 specification
  - Complete implementation of UPC collectives & I/O
  - Entirely open source, based on Open64
- Berkeley upc_mem*() library extensions for non-blocking and non-contiguous mem. copies

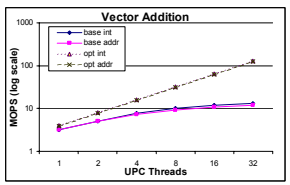
Berkeley UPC Portability

- Platform-independent generated code supports:
  - Network Hardware (supported through GASNet):
    - SMP, Myrinet, Quadrics Elan 3/4, Infiniband, IBM LAPI, Dolphin SCI, MPI, Ethernet, X1/Altix shmem
  - Operating Systems:
    - Linux, FreeBSD, NetBSD, Tru64, AIX, IRIX, HPUX, Solaris, Cygwin, Mac OSX, Unicos, SuperUX
  - CPU / System Architecture:
    - x86, Itanium, Opteron, Athlon, Alpha, PowerPC, MIPS, PA-RISC, SPARC, T3E, X1, SX-6
- Translator runs on Linux and Tru64
  - x86, Itanium, Opteron and Alpha
  - Seamless cross-compilation for other systems
    - using Berkeley translate server or your own

Compiler Optimizations

upc_forall Affinity Test Removal

- Removes the runtime branch from upc_forall loops
- Works for integer and shared address affinity expressions

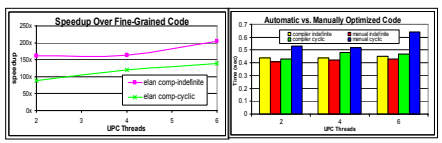


Split-Phase Access

- Transforms relaxed shared refs into split-phase comm.
- Moves reads initiations up, write completions down
- Conforms to the UPC memory consistency model

Message Coalescing

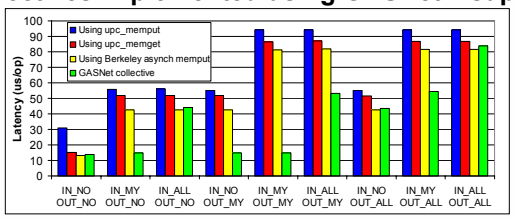
- Converts fine-grained accesses into bulk msgs
- Currently supports 1D arrays, extending to multi-D



Message Coalescing for Matrix-Vector Multiply

Berkeley UPC *Runtime*

- Fully open source, well-documented interface
- Berkeley GASNet used for communication:
  - Portability from layered design
  - Performance from inline functions, macros, and network-specific implementations
  - Collectives implemented using GASNet 2 support:



Small Broadcast Performance w/ varying sync mode
Point-to-Point UPC vs Direct GASNet collective

Interoperability

- Berkeley UPC provides app interoperability:
  - UPC calls to/from C, C++, FORTRAN, MPI
- Runtime supports multiple UPC compilers:
  - Berkeley source-to-source UPC translator
  - Intrepid GCC/UPC binary compiler

Tools and Testing

- Debugging with Etnus Totalview upcoming
- Extensive compliance test suite (>500 tests)
- *GASNet Trace* performance analysis tool
  - Reports time threads spend waiting at barriers
  - Summarizes shared memory refs, local & remote
  - Reveals load imbalance, communication "leaks", message size stats, all by source line & thread:

```
IS.c (NAS Integer Sort)
486 void upc_reduced_sum (shared
487     elem_t* buffer, size_t size) {
    ...
493   ptrs = upc_all_alloc(THREADS,
494     sizeof(elem_t *)) ;
    ...
501 upc_forall (thr_cnt=1;
502     thr_cnt<THREADS;
        thr_cnt <<= 1;  continue) {
504     if (!(MYTHREAD%(thr_cnt<<1))) {
505       upc_memget(local_array,
506           ptrs[MYTHREAD + thr_cnt],
507           size * sizeof(elem_t));
508   ...
519 }
```

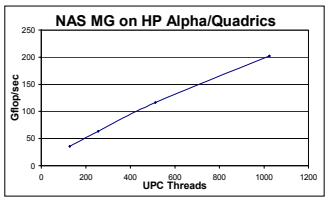
```
BARRIER REPORT:
SOURCE LINE      TYPE TIME(min     max     avg     total)   CALLS
==========================================================================
is.c : 493       WAIT      681 us   5  s   619 ms   27  s    44
    Thread 0               285 ms   5  s   819 ms    9  s    11
    Thread 1               681 us   1 ms   788 us    9 ms    11
    Thread 2               302 ms   5  s   834 ms    9  s    11
    Thread 3               287 ms   5  s   821 ms    9  s    11
...

GET REPORT:
SOURCE LINE      TYPE  MSG:(min    max     avg     total)  CALLS
==========================================================================
is.c : 505       GLOBAL     8 B   4 K    2 K     133 K     66
     Thread 0               8 B   4 K    2 K      89 K     44
     Thread 2               8 B   4 K    2 K      44 K     22
...
```

Berkeley UPC *Applications*

- Application benchmark suite
  - Used to evaluate ease-of-use, performance, and scalability
  - Motivates language/compiler extensions

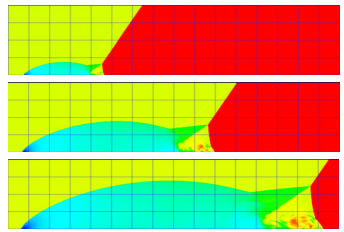
NAS MG

- Uses one-sided programming style
- Scales well to 1,000's of UPC threads



Godunov CFD

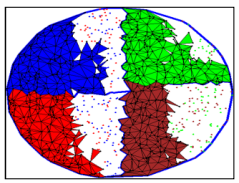
- Interfaces to FORTRAN numerics from LBNL Chombo
- Uses multi-D strided data movement
- Motivates strided memcpy extensions



Time evolution of a Mach 10 shock wave reflecting off surface at 30 degree angle, performed on SGI Altix at ORNL

Parallel Delaunay Triangulation

- Demonstrates use of irregular distributed data structures
  - *Directories* maintain replicated cached copies of remote data
  - App-level *teams* for collectives and memory allocation
- Hand optimizations motivate compiler support for message coalescing and strip-mining



Mesh Generation (Triangulation) in Progress

UPC Linpack

- Dense LU factorization (HPL)
- First step in sparse direct factorization (SuperLU)