Getting Multicore Performance with UPC

Yili Zheng

Lawrence Berkeley National Lab
Berkeley UPC Group

• PI: Katherine Yelick
• Group members: Filip Blagojevic, Dan Bonachea, Paul Hargrove, Costin Iancu, Seung-Jai Min, Yili Zheng
• Former members: Christian Bell, Wei Chen, Jason Duell, Parry Husbands, Rajesh Nishtala, Mike Welcome
• A joint project of LBNL and UC Berkeley
Outline

• Introduction of PGAS and UPC
• UPC examples
• UPC on shared-memory machines
• Auto-tuned multi-threaded Collectives
• Scheduling and load balancing
• Performance tuning
Features in Computer Architectures

• Many cores on a chip, multi-sockets in a node
  – Global address space
  – *May not be cache coherent*

• Non-Uniform Memory Access
  – Multi-level memory hierarchies
  – Private vs. shared
  – Local vs. remote

• Hybrid systems
  – Heterogeneous processors
  – Separate memory systems
Partitioned Global Address Space

- Global data view abstraction for productivity
- Vertical partitions among threads for locality control
- Horizontal partitions between shared and private segments for data placement optimizations
- Friendly to non-coherent cache architecture
UPC Programming Models

SPMD

Fork-Join

Synchronizations
int *p1; /* private pointer to local memory */
shared int *p2; /* private pointer to shared space */
int *shared p3; /* shared pointer to local memory */
shared int *shared p4; /* shared pointer to shared space */
Multi-Dimensional Arrays

**Static 2-D array**: shared [*] double A[M][N];

**Dynamic 2-D array**: shared [] double **A;**

A and pointers are private and replicated on all threads.
UPC Example of Jacobi

- Good spatial locality
- Mostly local memory accesses
- No explicit communication ghost-zone management

```upc
shared [ngrid*ngrid/THREADS] double u[ngrid][ngrid];
shared [ngrid*ngrid/THREADS] double unew[ngrid][ngrid];
shared [ngrid*ngrid/THREADS] double f[ngrid][ngrid];

upc_forall( int i=1; i<n; i++ ) {
    for(int j=1; j<n; j++) {
        utmp = 0.25 * (u[i+1][j] + u[i-1][j] + u[i][j+1] + u[i][j-1] - h*h*f[i][j]); /* 5-point stencil */
        unew[i][j] = omega * utmp + (1.0-omega)*u[i][j];
    }
}
```
UPC Example of Random Access

```c
shared uint64 Table[TableSize]; /* cyclic distribution */
uint64 i, ran;

/* owner computes, iteration matches data distribution */
upc_forall (i = 0; i < TableSize; i++; i)    Table[i] = i;

upc_barrier; /* synchronization */

ran = starts(NUPDATE / THREADS * MYTHREAD); /* ran. seed */

for (i = MYTHREAD; i < NUPDATE; i+=THREADS) /* SPMD */
{
    ran = (ran << 1) ^ (((int64_1) ran < 0) ? POLY : 0);
    Table[ran & (TableSize-1)] = Table[ran & (TableSize-1)] ^ ran;
}
upc_barrier; /* synchronization */
```
UPC Parallel DGEMM

- Transfer data in large blocks (use upc_memcpy)
- Use optimized BLAS dgemm (e.g., Intel MKL)
- Use non-blocking collective communication if available (e.g., row and column broadcasts)
Matrix-Multiplication on Cray XT4

Matrix size: (8K X 8K doubles) per node
Berkeley UPC Software Stack

UPC Applications

UPC-to-C Translator

Translated C code with Runtime Calls

UPC Runtime

GASNet Communication Library

Network Driver and OS Libraries

Hardware Dependant

Language Dependant
Berkeley UPC Features

• Data transfer for complex data types (vector, indexed, stride)
• Non-blocking memory copy
• Point-to-point synchronization
• Remote atomic operations
• Active Messages
• Extension to UPC collectives
• Portable timers
Process vs. Threads

Map UPC threads to Processes vs. Map UPC threads to Pthreads

Physical Shared-memory vs. Virtual Address Space
Processes with Shared Memory

Figure from Filip Blagojevic

Aggregate Bandwidth

Bandwidth (MB/s)

Data Chunk Size (in Bytes)

(#UPC Threads, #Pthreads)

(32,1)  (32,2)  (32,4)

(32,8)  (32,16)

Bandwidth (MB/s)

0  500  1000  1500  2000  2500  3000

8  16  32  64  128  256  512  1K  2K  4K  8K  16K  32K  64K  128K

2/26/2010

SIAM PP 10 -- Getting Multicore Performance with UPC
Process vs. Threads

GUPS Speedup over (32,1)

PSEARCH Speedup over (32,1)

Figures from Filip Blagojevic
Multi-threaded Collective Communication

- Enhance both productivity and performance
- Performance Auto-tuning
  - Offline tuning for platform common characteristics
  - Online tuning: Optimize for application runtime characteristics
- Multi-threaded implementation
  - Lower context switching overhead
  - Faster shared data access
Barrier on AMD Opteron (32 cores)

Figure from Rajesh Nishtala

2/26/2010
SIAM PP 10 -- Getting Multicore Performance with UPC
Broadcast on Sun Niagara2 (128 threads)

Figure from Rajesh Nishtala
Scheduling and Load Balancing

• Over-subscription
  – Run more logical threads than physical cores
  – Moderate performance improvement if synchronization intervals are not too small

• Speed balancing
  – User-level thread scheduling based on thread progress
  – Better system throughput in shared environments

• Cooperative thread scheduling
  – Good for event-driven type of applications
  – Accelerators, e.g., CELL processor
NPB UPC on Intel (16 cores)

NPB UPC on AMD (16 cores)
Tips for UPC Programming

• Coarsen synchronization intervals
• Hierarchically map UPC threads to OS processes and Pthreads
• Pin processes and threads to cores to minimize migration cost
• Take advantage of data locality in the application level
• Overlapping communication and computation
• Use tuned math libraries, e.g., AMD ACML, IBM ESSL, Intel MKL
Tools for Debugging and Tuning UPC Applications on Multi-core Systems

• Same as other multi-process and multi-thread programs
  – Open Source tools: PAPI, TAU, Valgrind
  – Commercial tools, e.g. Intel VTUNE, TotalView

• BUPC tracing tool for analyzing the communication behavior of UPC programs

• Parallel Performance Wizard (PPW)
  – http://ppw.hcs.ufl.edu/
Summary

• Global address space improves productivity
• Data partitioning enables performance optimizations
• Interoperable with other programming models and languages including MPI, FORTRAN, C++
• Growing UPC community with actively developed and maintained software implementations
  – Berkeley UPC and GASNet: http://upc.lbl.gov
  – Other UPC compilers: Cray UPC, GCC UPC, HP UPC, IBM UPC, MTU UPC