UPC

Unified Parallel C

January 2014
Seminar on Languages for Scientific Computing
RWTH Aachen

Reyhaneh Yazdani
Reyhaneh.yazdani@rwth-aachen.de

RWTH AACHEN UNIVERSITY AICES

Outline

- Introduction
  - *Execution Model*
  - *Programming Model*
  - *Memory Model*
- UPC Features
  - *Shared vs. Private Data*
  - *UPC Pointers*
  - *Workload Sharing*
  - *Libraries*
- Compiling and Tools
- Performance
- Summary
- References

Introduction

- Parallel Programming Language
- Project based at Berkeley Laboratory
- Extension of C
- For high performance computing on distributed shared memory architectures
- V1.3 , released in November 2013

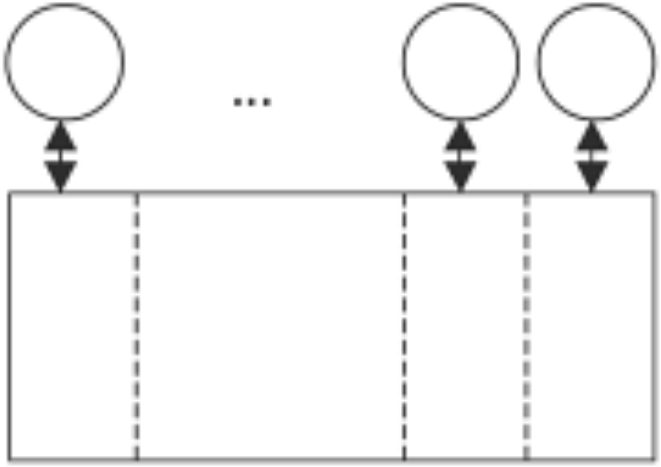
Execution Model

- Follow Single Program, Multiple Data (SPMD) model
  - Execute same program but may process different data
  - Threads working independently
  - Number of threads specified at compile time or run-time, available as program variable **THREADS**
  - **MYTHREAD** specified thread index


```
#include <upc.h>
#include <stdio.h>
int main(int argc, char** argv)
{
    printf("Thread %d of %d: hello UPC world\n",MYTHREAD, THREADS);
}
```

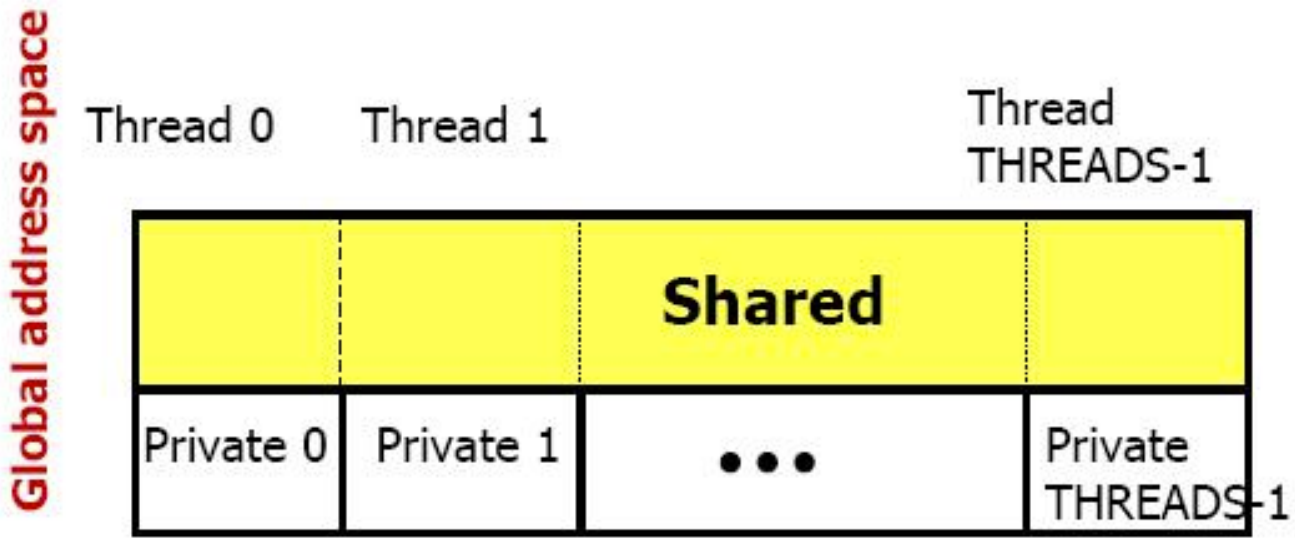
Programming Model

- Partitioned global address space (**PGAS**)
  - Shared memory and message passing at the same programming model
  - Only one address space that is partitioned among threads
  - Each partition has affinity with one thread
  - Can access everything but not with the same cost



Memory Model

- There is data-thread affinity
- A pointer-to-shared can reference all locations in the shared space
- A private pointer may reference addresses in its private space or its local portion of the shared space



http://www2.hpcl.gwu.edu/pgas09/tutorials/upc_tut.pdf

PGAS vs. other programming models/language

	UPC, X10, Chapel, CAF, Titanium	MPI	OpenMP
Memory model	PGAS (Partitioned Global Address Space)	Distributed Memory	Shared Memory
Notation	Language	Library	Annotations
Global arrays?	Yes	No	No
Global pointers/references?	Yes	No	No
Locality Exploitation	Yes	Yes, necessarily	No

http://www2.hpcl.gwu.edu/pgas09/tutorials/upc_tut.pdf

Outline

- Introduction
  - *Execution Model*
  - *Programming Model*
  - *Memory Model*
- UPC Features
  - *Shared vs. Private Data*
  - *UPC Pointers*
  - *Workload Sharing*
  - *Libraries*
- Compiling and Debugging
- Performance
- Summary
- References

Shared vs. Private data

- Private variable
  - Normal C variables
  - Allocate in private memory space
  - Used only by one thread
- Shared variable
  - Used for communication between threads
  - Allocate only once with thread 0
  - Use “shared” type qualifier
  - Can not have automatic storage duration => static
- Shared array
  - Shared arrays are spread across all thread partitions
  - Shared array can be distributed in round-robin method or by block

```
shared [block-size] type array[N]  //by block
```

Examples of shared and private data layout

```
shared int x;
shared int y[THREADS];
int z;
```

Assume **THREADS** = 3

Thread 0	Thread 1	Thread 2
x		
y[0]	y[1]	y[2]
z	z	z

Example of Shared array data layout

Assume **THREADS** = 4

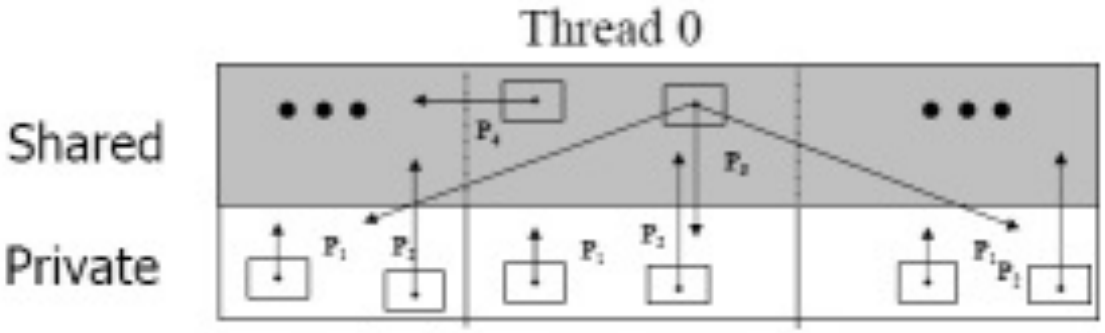
```
Shared [3] int Y[4][THREADS];
```

Thread 0	Thread 1	Thread 2	Thread 3
Y[0][0]	Y[0][3]	Y[1][2]	Y[2][1]
Y[0][1]	Y[1][0]	Y[1][3]	Y[2][2]
Y[0][2]	Y[1][1]	Y[2][0]	Y[2][3]
Y[3][0]	Y[3][3]		
Y[3][1]			
Y[3][2]			

UPC Pointers

- Private pointer: Points to its private space and its shared space
  - int *p1 : private pointer pointing locally
  - Shared int *p2 : private pointer pointing in to shared space
    - Good for speed and flexibility

- Shared pointer:Points to all locations in shared space
  - int *shared p3 : shared pointer pointing locally
    - May access violation by any thread that is not the owner of that private space
    - not recommended
  - Shared int *shared p4 : shared pointer pointing to the shared space
    - only one instance, residing on thread 0



Workload sharing

- upc_forall(init; test; loop; affinity)
  - Affinity could be an integer expression, or a
  - Reference to (address of) a shared object

  - Example

```
upc_forall( i=0; i<100 ; i++ ; i)
upc_forall( i=0; i<100 ; i++ ; &a[i])
```

Libraries

- UPC Collectives
  - All threads participate
  - Bulk data movement and computation
  - implement common communication pattern (shared-memory pace)
  - upc_barrier , upc_all_broadcast , …
- UPC I/O
  - upc_io.h
  - Fuctions are collective
  - upc_global_exit, upc_call_fclose
- UPC Numerical
  - UPCBLAS: a library for parallel matrix computations in UPC
  - UPCBLAS V1.0 released in October 2012

Outline

- Introduction
  - *Execution Model*
  - *Programming Model*
  - *Memory Model*
- UPC Features
  - *Shared vs. Private Data*
  - *UPC Pointers*
  - *Workload Sharing*
  - *Libraries*
- Compiling and Tools
- Performance
- Summary
- References

Compiling and Tools

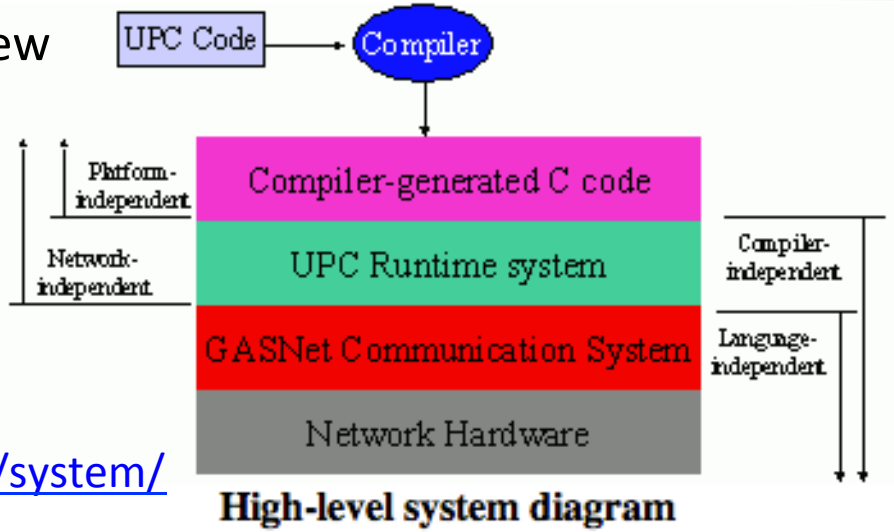
- Number of threads are defined either at **compile time** or **program startup time**

- Command for compilation :
  - `> upcc –o executablefilename –THREADS #threads filename.upc`
  - `> upcrun executablefilename`

- and if the number of threads is defined at startup:
  - `> upcc –o  executablefilename  filename.upc`
  - `> upcrun –n  #threads executablefilename`

UPC Compilers

- **HP UPC Compiler**
  - Compatible with the HP C and HP C++ products
  - UPC-specific performance optimization
  - Cache configuration at run time
  - Rich diagnostic and error-checking facilities
- **Berkeley UPC Compiler**
  - Supports a wide variety of operating systems and CPUs
    - Linux,Windows,Mac OSX, … X86,PowerPC,MIPS,…
  - Optimized pointer-to-shared implementation
  - Can be debugged by Totalview
  - Trace analysis: upc_trace



High-level system diagram

http://upc.lbl.gov/docs/system/

- **MuPC UPC Runtime system**
  - Michigan Tech UPC
  - Porable UPC, run on Linux-based
  - Reason: “spread the word” about UPC
  - Write-back software cache for remote accesses
  - Cache configurable at run-time
- **GCC-Based UPC Compiler**
  - Implemented as a C language dialect translator
- **UPC Cray T3E Compiler**

Tools

- **GASP: Global Address Space Performance tool interface**
  - Implemented in Berkeley UPC
- **Totalview : Debugger**
  - HP UPC, Berkeley UPC
  - Examined shared variable values at run-time
  - Must be GNU GCC
  - Configured with –enable-trace
- **Trace tool : Berkeley UPC**
  - Run with upcrun –trace or upcrun –tracefile TRACE_FILE_NAME
  - Retrieve statistics of runtime communication events

Outline

- Introduction
  - *Execution Model*
  - *Programming Model*
  - *Memory Model*
- UPC Features
  - *Shared vs. Private Data*
  - *UPC Pointers*
  - *Workload Sharing*
  - *Libraries*
- Compiling and Tools
- Performance
- Summary
- References

Performance

Three factors in achieving performance in UPC

- UPC Compiler
  - Avoid unnecessary communication cost => better bandwidth and latency
  - Berkeley UPC uses GASNet Communication layer
- UPC run-time system
  - UPC run-time system will map each thread and the data that has affinity to it to the same processing node
- Hand optimization
  - Threads are responsible to processing data that has affinity to that thread => use UPC features:
    - ✓ Control over data layout
    - ✓ Control over work distribution
  - Use UPC Collectives for data movements and computational operations => reduce inefficient operations
  - Use private pointers instead of shared when dealing with local shared data

Outline

- Introduction
  - *Execution Model*
  - *Programming Model*
  - *Memory Model*
- UPC Features
  - *Shared vs. Private Data*
  - *UPC Pointers*
  - *Workload Sharing*
  - *Libraries*
- Compiling and Tools
- Performance
- Summary
- References

Summary

- Designed for Parallel high performance
- PGAS language
- Follow SPMD model
- Simple to program for C writers
- Easy to read
- Exploit data locality for improved performance
- Provides rich pointer concept
- Work can be distributed conveniently

References

- UPC: Distributed Shared Memory Programming
  By Tarek El-Ghazawi, William Carlson, Thomas Sterling, Katherine Yelick
  Wiley, May, 2005 , ISBN: 0-471-22048-5

- http://upc.gwu.edu/
- http://upc.gwu.edu/tutorials.html
- http://www.cs.berkeley.edu/~bonachea/upc/
- http://upc.lbl.gov/publications/#performance
- http://upcblas.des.udc.es/

- Jorge Gonzalez-Dominguez, Maria J. Martin, Guillermo L. Taboada, Juan Tourino, Ramon Doallo, and Andres Gomez **: A Parallel Numerical Library for UPC**
- Damian A. Mallon, Guillermo L. Taboada, Carlos Teijeiro, Juan Tourino,Basilio B. Fraguela, Andres Gomez, Ramon Doallo, and J. Carlos Mourino**: Performance Evaluation of MPI, UPC and OpenMP on Multicore Architectures**
- Damian A. Mallon, J. Carlos, Mourino, Andres Gomez, Guillermo L. Taboada, Carlos Teijeiro, Juan Tourino, Basilio B. Fraguela, Ramon Doallo and Brian Wibecan: **UPC Performance Evaluation on a Multicore System**
- Hongzhang Shan, Haoqiang Jin, Karl Fuerlinger, Alice Koniges, Nicholas J. Wright**: Analyzing the Effect of Different Programming Models Upon Performance and Memory Usage on Cray XT5 Platforms**
- Anup Tapadia , Fallon Chen: **A Performance Characterization of UPC vs. MPI**

Thank you