HPC Challenge Class 2
UPC and X10 on Multiple Platforms

IBM Team
Presenter: Calin Cascaval (cascaval@us.ibm.com)
Team

- **IBM TJ Watson Research Center**
  - George Almasi, Calin Cascaval, Vijay Saraswat, Igor Peshansky, Sayantan Sur
- **IBM Toronto SWG**
  - Kit Barton, Ettore Tiotto
- **IBM India**
  - Ganesh Bikshandi, Sreedhar B. Kodali, Krishna Nandivada Venkata, and Pradeep Varma
- **UPC Barcelona**
  - Montse Farreras
Submission Overview

- **Two languages:**
  - Unified Parallel C (UPC): [http://upc.gwu.edu](http://upc.gwu.edu)
  - X10: [http://x10-lang.org](http://x10-lang.org)

- **One common PGAS runtime to provide performance portability**
  - X10 and XLUPC compilers both target the common PGAS runtime
  - Runtime provides services such as threading, data distribution, messaging (including collective communication) thus enabling efficient execution and interoperability

- **Two platforms: Power5 SMP clusters and Blue Gene/L**
  - Power5 clusters: 32 nodes, 16-way Power5 1.9 GHz, 64 GB memory/node, AIX
  - Blue Gene/L: 8 racks (16K processors) on the BG/W machine

- **Four applications (HPL, FFT, Stream, and RandomAccess) coded in both UPC and X10**
  - Implementations from scratch using the algorithm specification
  - All applications use SPMD style parallelism in both languages.

**Two teams of programmers, different skills, similar performance**
### Lines of code

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>UPC</th>
<th>X10</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPL</td>
<td>600</td>
<td>550(*)</td>
</tr>
<tr>
<td>FFT</td>
<td>180</td>
<td>215</td>
</tr>
<tr>
<td>Random Access</td>
<td>107</td>
<td>53</td>
</tr>
<tr>
<td>STREAM</td>
<td>90</td>
<td>86</td>
</tr>
</tbody>
</table>

*: Reflects code for parallel swaprows
PGAS runtime structure

- **XL UPC front-end**
  - UPC runtime API
    - array index arith.
    - deref assign update
    - locks allocators etc
  - startup shutdown
  - shared variable directory
  - collective API
  - memory allocation, locks
  - Distributed transport API
    - BlueGene messaging
    - Myrinet GM / MX
    - TCP/IP sockets
    - LAPI/GSM
    - IB verbs
    - dummy (SMP)

- **CAF front-end**
  - CAF runtime API
    - deref assign update
    - locks allocators etc
  - Callback API
    - Asyncs clocks/finish
    - Work stealing
    - value caching

- **X10 compiler**
  - X10 API
    - Asyns clocks/finish
    - Work stealing
Productivity Considerations – HPL

- Both versions, UPC and X10, are implemented using a tiled layout and data-centric communication for scalability
  - Our previous submissions were totally asynchronous and in shared memory (X10) and a naïve global-view (UPC)

- Tiled layout
  - Supported in X10 using a fragmented representation over all places
  - Extended UPC with tiled array expressions and processor layout directives (HPF-like)

- Data-centric communication: using collectives for communication
  - X10 communicators: dynamic “unique distributions” that provide multi-place broadcast/reduce/barrier a la MPI communicators
  - Teams in UPC that allow collectives on a subset of the threads

- Common runtime support exploited by both language implementations
HPL – Optimized Broadcast

There are TX row-comms and TY col-comms
place (i,j) uses the i’th row-comm and the j’th col-comm.
Performance Results: Power5 clusters

HPL

FFT

Random Access

STREAM
Performance Results: Blue Gene/L

HPL

FFT

Random Access

STREAM
Discussion

- **HPL efficiency: Power5 (78% UPC, 55% X10), BG/L (50%)**
  - X10: Rank-1 updates used in panel, moving to Rank-K will improve performance
  - X10/BG: Memory leak in generated code affects results on Blue Gene/L
  - UPC+X10: Even higher efficiency expected with better scheduling of updates (cf X10 HCP07 submission)

- **Random Access:** designed to reach the maximum cross-section bandwidth for largest configurations
  - Blue Gene/L: At 8 racks it reaches 77% efficiency. At 64 racks it reaches 82% efficiency (2006 results)
  - Power5: Each SMP has two adapters, each capable of delivering 1 update/µs through LAPI, performance is limited by the interconnect latency

- **FFT: implementation differences**
  - X10 implementation is using blocked transposes overlapped with exchange, that pays off for the Power5 cluster, but not on Blue Gene/L
  - UPC is using a naïve non-blocking exchange that better exploits the overall Blue Gene/L cross-section bandwidth

- **Stream**
  - X10 simpler index computation allows better code generation than UPC on Power5
Why UPC and X10?

- Our submission addresses high-productivity, high-performance **programming environment** for programmers
  - An environment is more than a language!
    - Interoperability, performance portability, etc.
  - Lets programmer choose language they are comfortable with (X10 for Java programmers, UPC for C programmers)

- Common runtime helps programmers
  - Easier to write parallel, inter-operating code in both languages
  - Possible to debug programs in one language using a program in the other (e.g. we debugged X10 performance on LU using the UPC program.)
  - Exposes unified abstractions to both languages (e.g. communicators)
    - → These abstractions may serve as a basis for PGAS/MPI interoperability
Additional Information

- X10: [http://x10-lang.org](http://x10-lang.org)
Thank You!