

Experiences from Implementation of Sorting Algorithms in UPC

Zhaofang Wen (Sandia)

With

Ron Brightwell (Sandia)

Jonathan Brown, Quentin Stout (U. of Michigan)

Sorting

- **Why sorting?**
 - Rich communication patterns
 - To test the expressiveness of UPC
 - Candidates for UPC library
- **Platform:** Cray T3E/750 at MTU (thanks to Prof. Merkey)
- **Performance comparison** (used integers for fair comparison)
 - Fastest: sample sort, radix sort (distributed styled)
 - Medium: bubble sort, merge sort (regular communication patterns)
 - Slow: quick sort (fine-grain communication)
 - Very slow: odd-even sort (very fine-grain communication)
- **Experiences**
 - UPC global shared array is convenient (compared to the lack of it in CAF)
 - It requires real efforts to redesign and to code the sorting algorithms in UPC (some of them may be impossible in MPI).
 - Need basic building blocks for expressing real algorithms
 - Need a data locality management layer (we are building it)
 - Need (more) communication collectives (already recognized)

Building Blocks for PGAS Algorithms and Programming

- **(Thread) Prefix:** A thread version of parallel prefix, that is, P threads each holding a number
- **Array Concatenation**
 - **Input:** P threads, each holding a private vector (of possibly distinct size)
 - **Output:** A global shared array whose values are a concatenation of the vectors.
- **Generalized Array Concatenate (2D version)**
- ...

Data Locality Management

- **ThreadView**: provides a view of a shared array as a distinct array on each thread, a translation (formula) from the global shared array index to the offset on a particular thread. (UPC shared --> CAF co-array)
- **ReverseThreadView**: provides a translation from an offset on a particular thread to a global shared array index. (CAF --> UPC)
- **BlockSizeMapping**: a translation formula, to allow the programmers to use shared arrays of arbitrary block size.
 - **Implication to UPC**: The block size concept at the language level is not fundamental and unnecessary
 - **Suggestion**: Hide (remove) the concept of block size from the user level. Always use a default block size (for example, N / P)
- ...

On-going and Future Work

- **On-going**
 - Building a data locality management layer for common data structures in applications (dense matrix, sparse matrix, ...)
(The ones we already have seem very useful.)
 - Experimenting with a group-barrier scheme as compared to global barrier in various algorithms/applications
- **Future**
 - UPC porting and runtime work (for example, on Red Storm)
 - More algorithms
 - Large applications
 - ...

Questions?