Interfacing Chapel with traditional HPC programming languages

Adrian Prantl

Center for Applied Scientific Computing (CASC)
Lawrence Livermore National Laboratory

November 16, 2011
Chapel Lightning Talks BoF at SC 2011

This work performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

1 This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.
Related work

Babel

- LLNL’s language interoperability toolkit for high-performance computing
- Designed for fast in-process communication
- Handles generation of all glue-code
- Features multi-dim. arrays, OOP, RMI, …
BRAID connects Babel with PGAS languages

How does it work

Programming-language-neutral **interface specification**

Scientific Interface Definition Language (SIDL)

SIDL supporting

- fundamental data types
- object-oriented programming (user-defined types)
- interface inheritance
- exception handling
- dynamic multi-dimensional arrays
Using Chapel with BRAID — I

first, define the interface in SIDL

Example

```chapel
import hplsupport;
package hpcc version 1.0 {
    class ParallelTranspose {
        // C[i,j] = A[j,i] + beta * C[i,j]
        static void ptransCompute(
            in hplsupport.Array2dDouble a,
            in hplsupport.Array2dDouble c,
            in double beta,
            in int i,
            in int j);
    }
}
```

Adrian Prantl <adrian@llnl.gov> Interfacing Chapel w. traditional HPC programming languages
Using Chapel with BRAID — II

- use Babel compiler for server/callee glue code:
  ```bash
  ~/cxxLib> babel --server=cxx hpcc.sidl
  ```
  - generates code for **skeleton** and Intermediate Object Representation (IOR)
  - generates **splicer blocks** for **user code**

- use BRAID compiler for client/caller glue code:
  ```bash
  ~/chplClient> braid --client=chapel hpcc.sidl
  ```
  - generates a Chapel **stub** that implements our interface
  - **link to server code and SIDL runtime library during compilation and run the executable!**

- Babel/BRAID bindings take care of interoperability!
BRAID provides two options for distributed arrays:

**Transparent**

Distr. arrays are automatically converted to/from a local array
- re-use your existing Fortran lib without modification

**Via DistributedArray Interface**

Exposes the Chapel runtime to legacy programming languages
- use get() and set() methods for element access
- copy only what you need
Achieved interoperability between Chapel and
1. C
2. C++
3. FORTRAN 77, Fortran 90/95
5. Java
6. Python

including support for distributed arrays

Future work

- Chapel as a server language
- Parallel-Parallel interoperability (Chapel ↔ MPI/UPC/...
Try it!

- BRAID preview is included in Babel 2.0 release: http://www.llnl.gov/CASC/components/

- Development snapshots at: http://compose-hpc.sourceforge.net (BSD licensed)

For more details

- “Connecting PGAS and traditional HPC languages”
  ➡ at SC’11 poster session!

- “Interfacing Chapel with Traditional HPC Programming Languages”
  Adrian Prantl, Thomas Epperly, Shams Imam, Vivek Sarkar
  Fifth Conference on Partitioned Global Address Space Programming Models (PGAS 2011)