RICE George R. Brown School of Engineering
Computer Science

CnC-Babel
A Multi-Language Implementation of the Concurrent Collections model

Shams Imam, Vivek Sarkar
shams@rice.edu, vsarkar@rice.edu
Rice University

Adrian Prantl
adrian@llnl.gov
LLNL

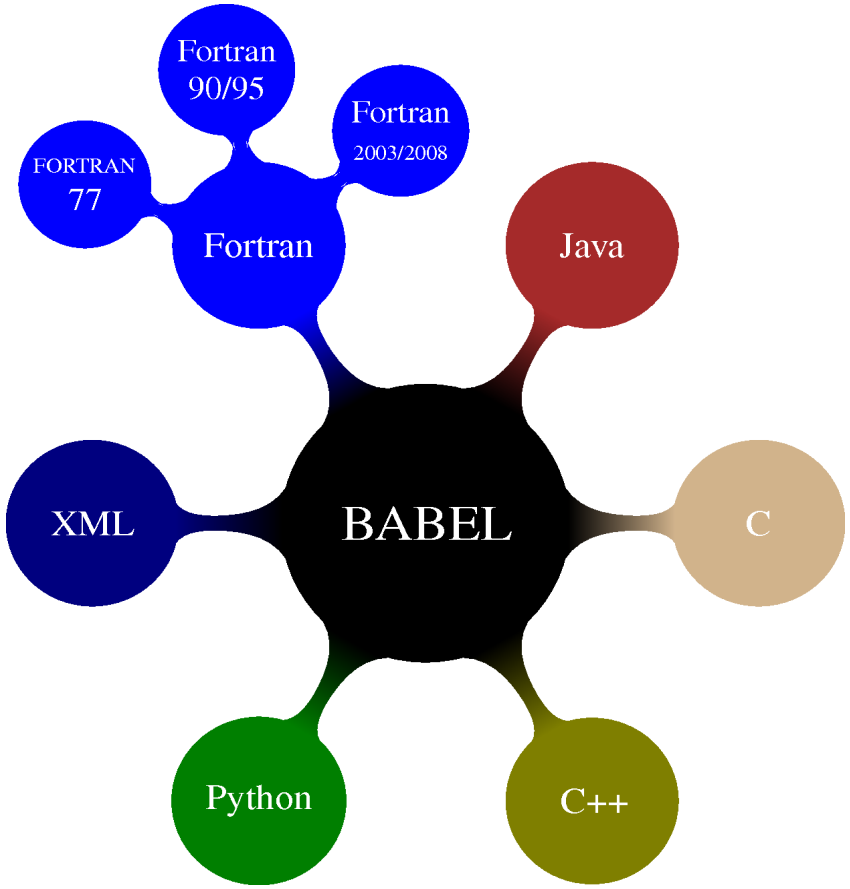
Motivation for Multi-language CnC

- Current CnC runtimes require steps be written in language chosen by the runtime
- Domain experts
  - may prefer other dynamic languages like Python
  - may prefer reusing existing code (in another language)
- Language interoperability issue!

RICE

Babel – language interoperability tool

- LLNL's language interoperability toolkit for high-performance computing
- designed for fast, in-process communication
- handles generation of all glue-code



RICE

Babel – relevant features

- programming language-neutral interface specification language – Scientific Interface Definition Language (SIDL)
- supports
  - user-defined types (classes)
  - interface inheritance
  - dynamic multi-dimensional arrays

Babel – SIDL interface definition

- First, define the interface in SIDL

```
import cncapi;
package partitionstring version 1.0 {

  interface SpansInputCollection {
    string get(in CncTag tag);
    bool containsTag(in CncTag tag);
  }
  interface SpansOutputCollection {
    void put(in CncTag tag, in string value);
  }

  class SpansCollection implements-all
      SpansInputCollection, SpansOutputCollection {
  }
}
```

RICE

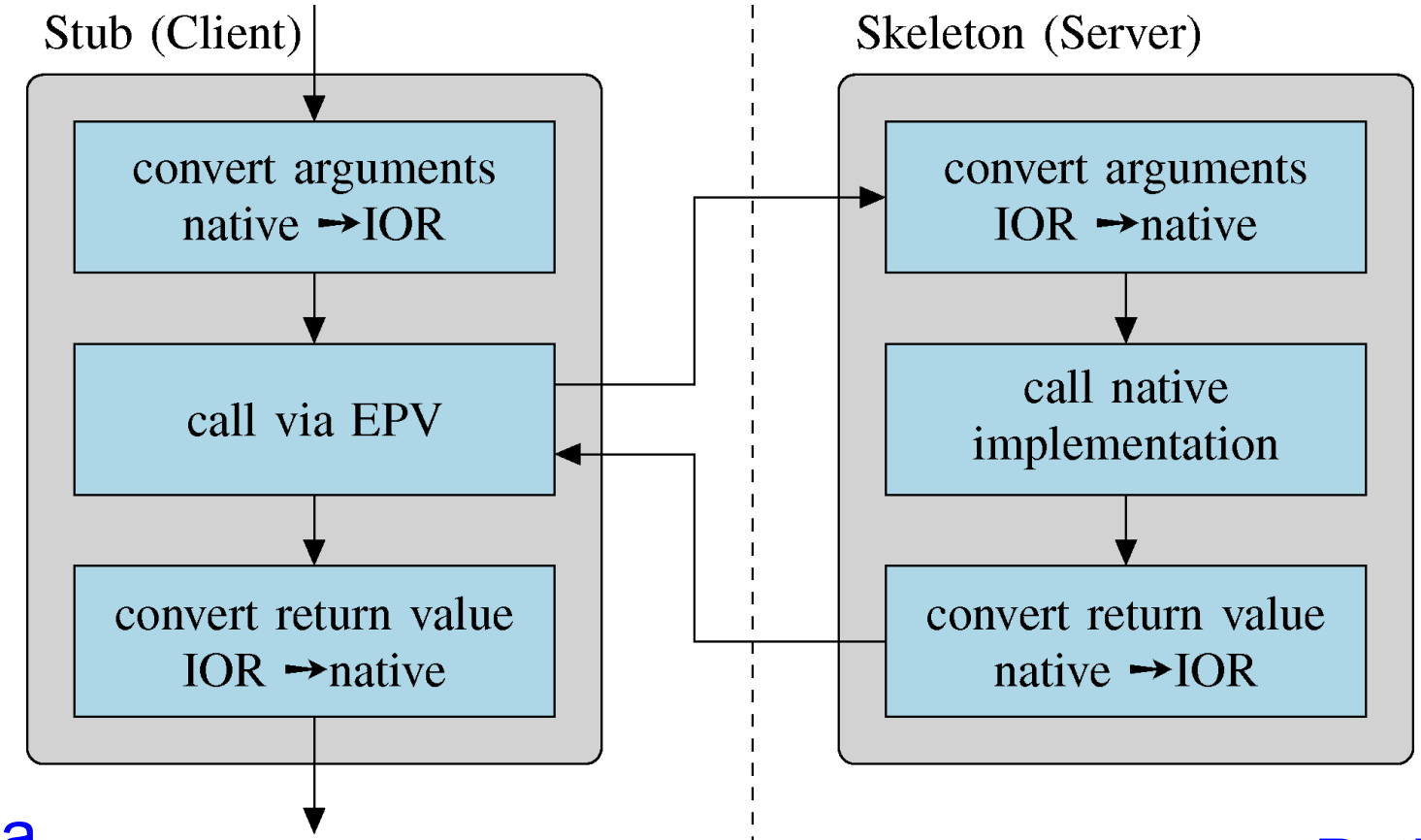
Babel – generate server and client

- Next, use the Babel compiler with the SIDL file as input to generate the server(callee) and client(caller) glue code:
- `~/ps/pyLib> babel --server=python spanc.sidl`
  - generates code for skeleton and intermediate object respresentation (IOR)
  - generates empty blocks expecting user code
- `~/ps/javaClient> babel --client=java spanc.sidl`
  - generates code for stub and IOR
  - user code uses the stub to make method calls into the external language

RICE

Babel – method invocation scheme

- Example flow while calling from Java into Python



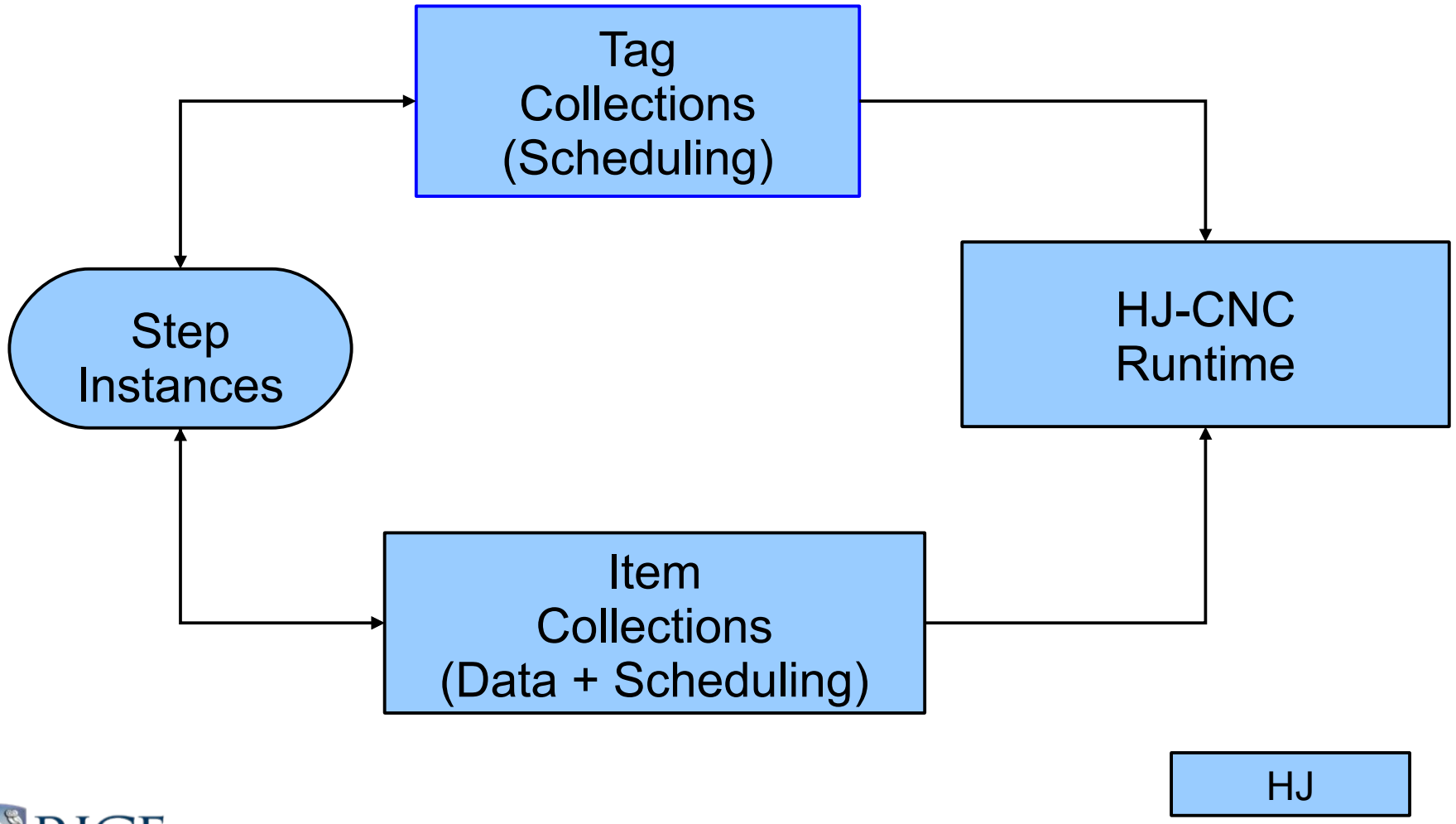
Java

Python

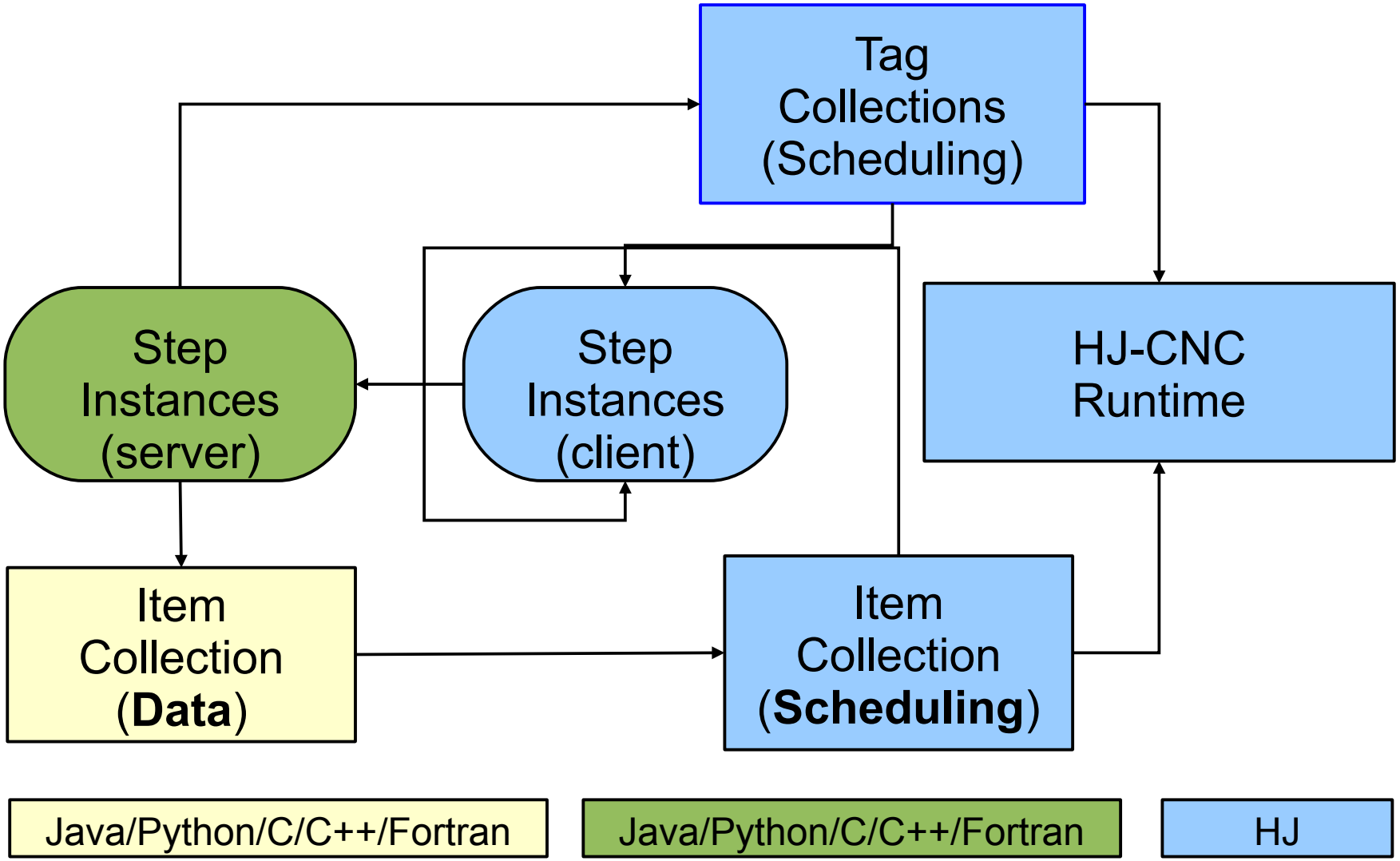
HJ-CnC and Babel?

- HabaneroJava is a fork of IBM X10
  - provides language constructs for parallel programming
  - uses these constructs for the HJ-CnC runtime
  - Steps and Item Collections are written in HJ
- Babel supports HJ!
  - HJ is effectively Java and runs on the JVM
  - HJ has bidirectional interoperability with Babel supported languages

HJ-CnC Runtime – current design



HJ-CnC-Babel – multi-language runtime



HJ-CnC-Babel – implementation details

- wrap and expose HJ Tag and Item collections
- require a well-defined type for a Tag for use in SIDL files (we use a CnCTag interface)
- require a native version of ConcurrentMap for ItemCollections
- get() and put() on native Item Collections need well-defined signatures for use in SIDL
  - `void putString(in CncTag tag, in string value)`
  - `sidl.array<int, 2> getIntArray2d(in CncTag tag)`

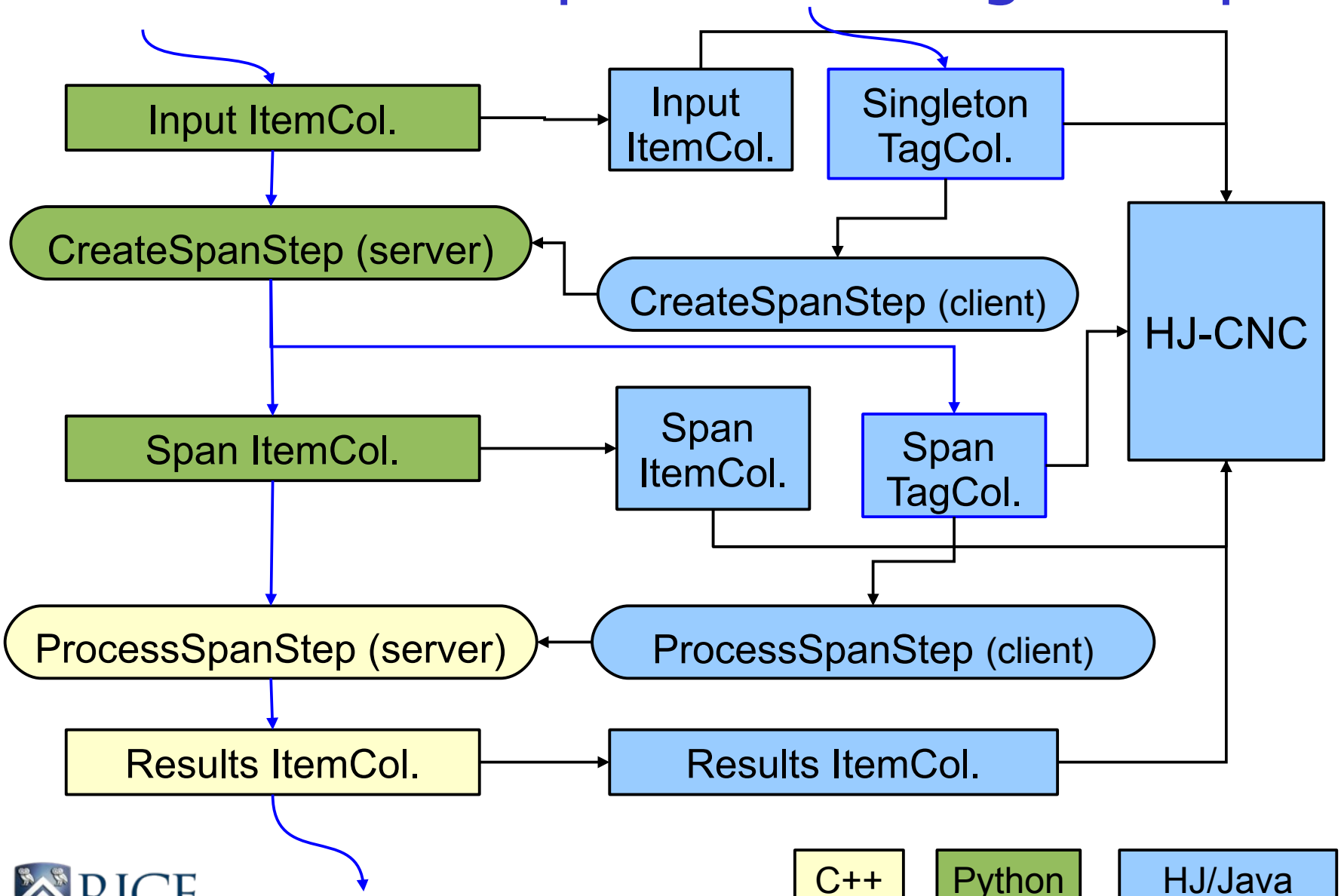
RICE

HJ-CnC-Babel – Steps to write a program

- user writes textual description of CnC graph
- user annotates graph with extra information to be used by Babel
  - impl. language for Steps and Item Collections
  - types of items stored in the Item Collections
- user runs translator to generate code
  - Babel generates Step template
- user writes Step code and initialization logic
- user runs code using HJ

RICE

HJ-CnC-Babel – partition string example



HJ-CnC-Babel – initial results

- Babel reports less than one percent overhead in common use cases
- HJ-CnC-Babel impl. of Cholesky [2000×2000]
  - run on dual core machine
  - HJ configured to use 8 threads
  - HJ-CnC version runs in about 10.8 secs
  - Step and Item Collections rewritten in C++
    - runs in about 6.4 seconds (about 40% faster)

RICE

Current Status

- HJ-CnC-Babel runtime available:
  - Steps can be written in any Babel supported language as well as HJ
  - Item Collections available for Python and C++
- code generation in progress
  - current examples implemented by manually modifying code generated by HJ-CnC

Summary and Future Work

- extended CnC-HJ runtime via Babel to allow CnC steps and item collections in multiple languages:
  - Tuning expert tunes runtime using HJ and various Item Collection implementations
  - Domain expert uses Babel-supported language of personal choice to implement steps
- plans in progress to use CnC-Babel to introduce a CnC-Python in COMP 140 at Rice

Questions