Motivations, part I

- Assume Java is to become a mainstream computing language.
- Scientific parallel computing must migrate to a new language platform.
- Opportunity to learn from shortcomings of previous scientific languages, and fix some of the bugs?

Motivations, part II

- HPF is not in a good state. Time to think about more practical languages?
- Can we save good features of HPF without assuming very complex compilers?
- Java is a (relatively) simple context for experiments with language ideas.

Base Model

- Base programming model will be SPMD programming.
- Interfaces to low-level communication platform (MPI) available, directly.
- Standard Java "source" programs *unchanged* by HPJava translator.

Adding value

- Distributed arrays as language primitives.
- Access to local elements of distributed arrays facilitated by *distributed control* constructs—language primitives. Includes *overall* distributed loop.
- PCRC collective communication functions, presented as a library of HPF-like transformational functions on distributed arrays.
- Library of GA's-style functions for access to non-local patches of distributed arrays.

Application domain

- Claim proposed model is *as* convenient as HPF for simple algorithms, and more flexible for (realistically) complex algorithms. Application domain presumed to include all problems where HPF and/or GAs is likely to successful.
- Applications with very unstructured accesses to regular arrays (CHAOS-like) OK. Reusable schedules available. Room for efficiency improvements.
- Situation for highly adaptive problems (DAGH-like applications) less clear. Fall-back position: resort to low-level SPMD programming for such phases, using HPJava distributed arrays in other phases of the program...

Key Technologies

1. PCRC Kernel Runtime. Distributed array descriptors and very general collective operations.
2. Interface from Java to PCRC.
3. Source to source translator from extended Java to standard Java + calls to PCRC. Use PCRC compiler infrastructor for rapid implementation.

Kernel Runtime

Kernel of NPAC library is a C++ class library. Influences from F90D, CHAOS/PARTI. Based on MPI.

- Object oriented, *but efficient*. Eg, use inline functions extensively.
- All collective operations work with general DAD.
- Array-array copying supported by `pcrc_remap` operation.
- Shift-style communication supported by `pcrc_write_halo`.
- General gather/scatter operations.
- General reductions and other F90 array arithmetic.
- Most collective operations based on reusable *schedule objects*.

Compiler Construction Framework

The *frontEnd* system is an extensible software package for constructing programming language processing tools.

Package is divided into

- **Language parsers.** Used to convert program source code to IR. IR suitable for multiple programming languages.
- **Package tools.** Used for displaying the IR.
- **C++ class library.** Class library extended from the Sage++ class library.
- **Test suite.** Extensive test suite for parsers. Eg, package includes a "first order" exhaustive syntax test, in which each BNF syntax rule is used at least once.

Package is autoconfigured and tested on different platforms, including: IBM AIX, SUN OS/Solaris, DEC OSF, HP HP-UX, SGI IRIX and PC Linux.

```
Range x = new CyclicRange(size, id);

float a[[*,]] = new float [[size, x]];

float b[[*]] = new float [[size]]; // used as a buffer

at (x|0)
  a[[0,x]]=sqrt(a[[0,x]]);

for (int k=0; k<size-1; k++) {

  for (int s=k+1; s<size; s++)
    at (x|k)
      a[[s,x]]/=a[[x,x]];

  Adlib.remap(b[k+1:],a[k,x|k+1:]);

  over (x|k+1:)
    for (int i=x; i<size; i++)
      a[[i,x]]-=b[[i]]*b[[x]];

  at (x|k+1)
    a[[k+1,x]]=sqrt(a[[k+1,x]]);
}
```

Time scale

- PCRC kernel available in-house now.
- Java interface is being specified—implementation is considered a straightforward task.
- Translator—something in time for SC?

http://www.npac.syr.edu/projects/pcrc/HPJava/