PCRC-based HPF Compilation

Guansong Zhang, Bryan Carpenter, Geoffrey Fox,
Xiaoming Li, Xinying Li, Yuhong Wen
Northeast Parallel Architectures Centre,
Syracuse University,
Syracuse, New York

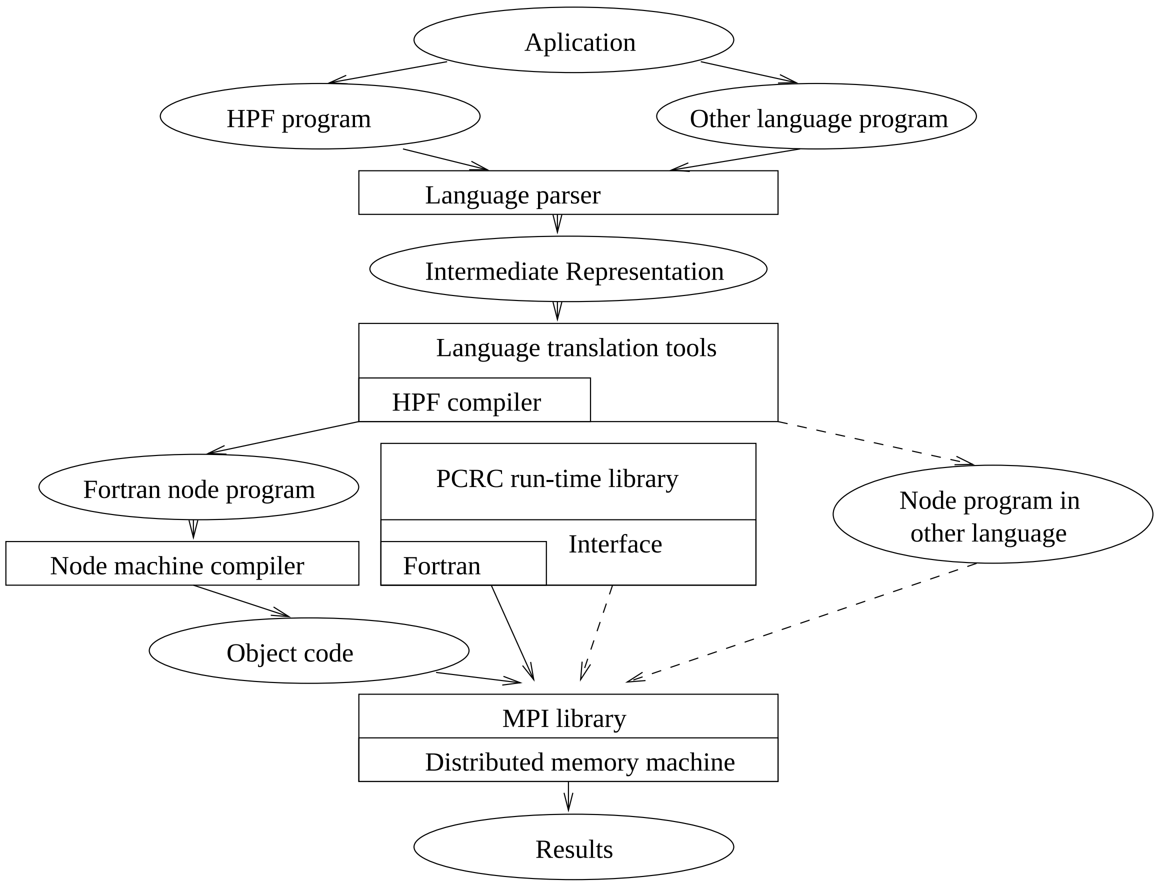
PCRC Project

- DARPA supported.
- Develop a public domain common runtime infrastructure supporting parallel compiler/interpreter development.
- Common library should be shared by different languages and programming models.
- Partners: NPAC, Rice, Maryland, Austin, Indiana, CSC, Rochester, Harvard.

Overview of NPAC HPF translation system

The translation system has three components

- **HPFfe**. A compiler front end for HPF 1.0. Complete coverage of HPF syntax and most static semantic checks. Generates IR extended from Sage++.
- **Transformation modules.** Transformations on IR and code generation.
- **PCRC runtime.** Provides functions for distribute array management, data movement, and computational functions corresponding to HPF intrinsics.


Aplication
HPF program
Other language program
Language parser
Intermediate Representation
Language translation tools
HPF compiler
Fortran node program
PCRC run-time library
Node program in
other language
Node machine compiler
Interface
Fortran
Object code
MPI library
Distributed memory machine
Results

Key Technologies

1. Distributed Array Descriptor (DAD).
2. Kernel Runtime.
3. Communication Detection and Classification.

Distributed Array Descriptor

Designing an effective DAD for HPF is non-trivial. Support many distribution formats (block-cyclic, collapsed, replicated, . . .) and alignments and embeddings, *and* describe sections constructed with arbitrary combinations of scalar and triplet subscripts.

NPAC DAD includes

- Type and rank information.
- Description of process subgroup to which array is mapped
- For each dimension, distribution format and alignment information.
- For each dimension, local memory stride.

Tabular representation of DAD

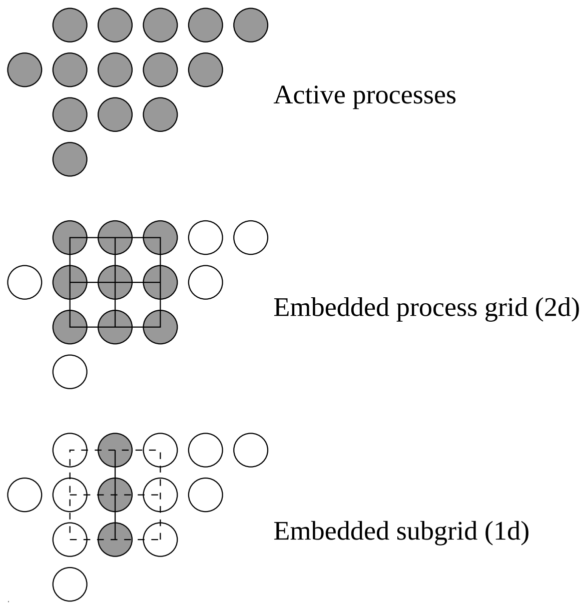
	0	1	2	3	4	5	6	
g_extent								0
t_extent								1
t_stride								2
t_offset								3
dist_code								4
on_pdim								5
l_extent								6
l_lb								7
l_ub								8
l_stride								9
ghostsize								10
p_shape								11
my_coord								12
slice_coord								13
	type	rank	p_rank	comm	major			14

bass_address

Hierarchical representation of DAD

- Array
  - Type information
    - Element type
    - Array rank
  - Global mapping
    - Process group
    - Range vector
  - Local mapping
    - Base address
    - Stride vector

Embedding groups handled by DAD

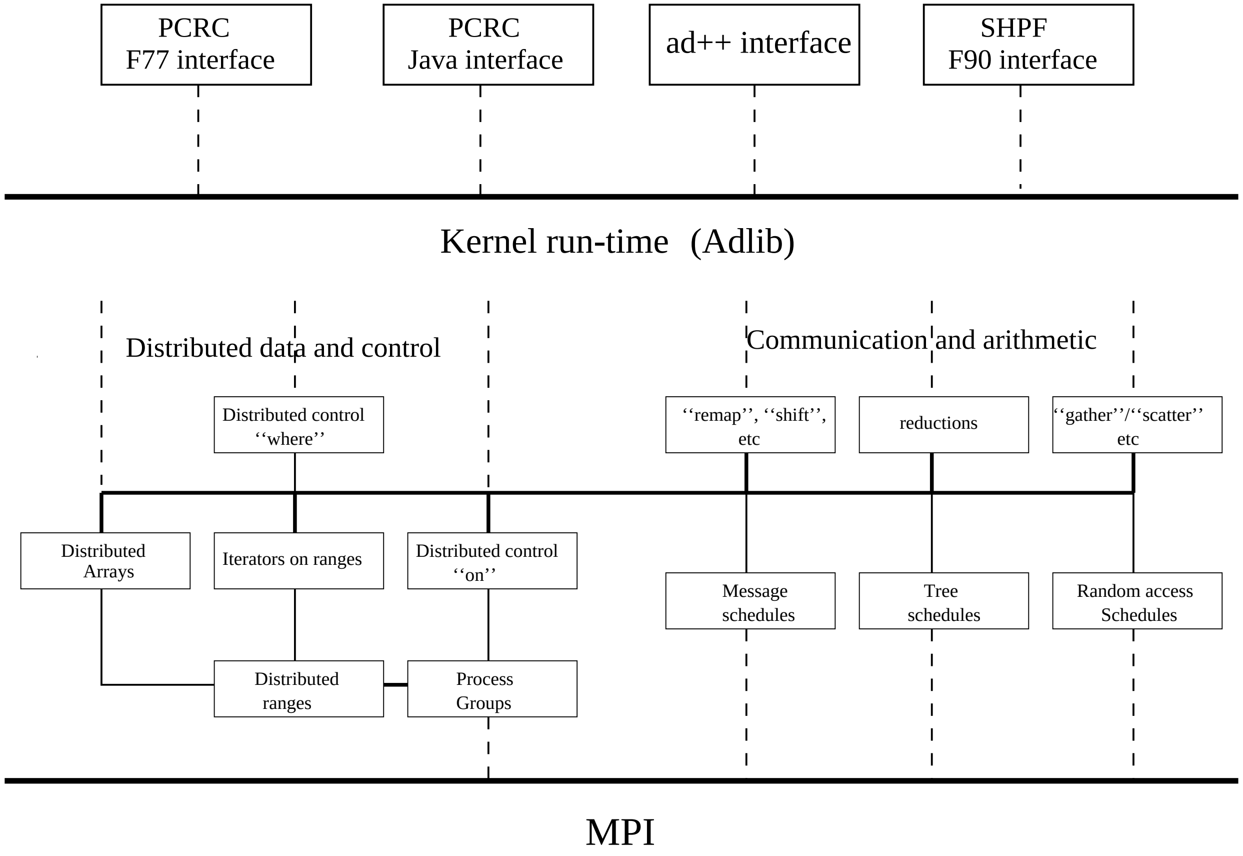


Kernel Runtime

Kernel of NPAC library is a C++ class library. Descended from runtime of *shpf* system, with influences from F90D, CHAOS/PARTI. Based on MPI.

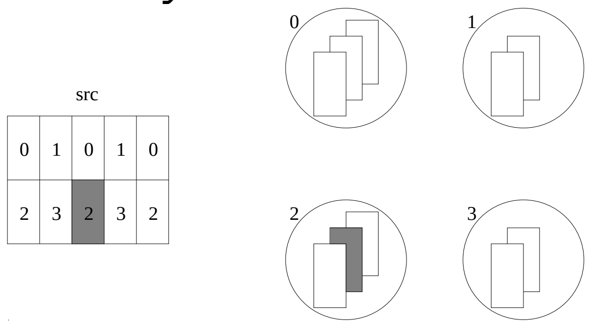
- Object oriented, *but efficient*. Eg, use inline functions extensively.
- All collective operations work with general DAD.
- Array-array copying supported by `pcrc_remap` operation.
- Shift-style communication supported by `pcrc_write_halo`.
- General gather/scatter operations.
- General reductions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

Runtime Architecture

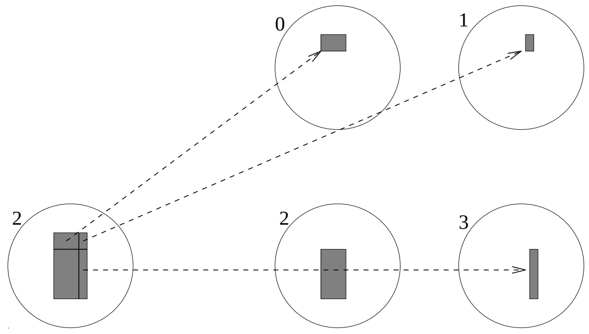


Remap Schedule

Distribution of source array:



Block send operations:



Reduction Schedules

```
template<class T>
class Sum : public Reduce_<T, T> {
public :
  Sum(const Frame* _src) : Reduce_<T, T>(_src) {}

private :
  void opCombine(void* buf) {
    *res += *((T*) buf) ;
  }

  void opZero() {
    *res = 0 ;
  }

  void opReduce(const int srcOffset) {
    blockSum(res, srcBase + srcOffset, steps, exts, rank) ;
  }
} ;
```

Communication Detection and Classification

For a regular array assignment or equivalent term in a forall, recognize three different communication strategies:

- **No communication.** Required elements are aligned with destination.
- **Shift communication.** A shift along the source array's template is sufficient.
- **Remap communication.** A general array remapping operation is required.

Shift Homomorphism

Classification based on *shift-homomorphism*.

```
!HPF$  PROCESSORS P($p$)
!HPF$  TEMPLATE TX($t_x$), TY($t_y$)
!HPF$  DISTRIBUTE TX(BLOCK) ONTO P
!HPF$  DISTRIBUTE TY(BLOCK) ONTO P

!HPF$  DIMENSION X(M), Y(N)
!HPF$  ALIGN X(I) WITH TX($a_x$* I + $b_x$)
!HPF$  ALIGN Y(I) WITH TY($a_y$* I + $b_y$)
       ...
       X($x_l$:$x_u$:$x_s$) = Y($y_l$:$y_u$:$y_s$)
```

t_x and t_y are normalized to be multiples of p. Array sections shift-homomorphic if have the same extent and

$$\frac{a_x . x_s}{a_y . y_s} = \frac{t_x}{t_y}$$

(different definition applies for cyclic distribution).

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.

Program Transformation

Transformation goes in two phases:

1. **Format transformation.** Components of source program are changed, making them suitable for being further processed. Eg, simple array assignments can be trivially converted to forall statements.
2. **Program generation.** Carries out the actual translation work and generates the node program.

DAD Generation

Consider the HPF fragment

```
      REAL X(1:205)
!HPF$ PROCESSORS P(2)
!HPF$ TEMPLATE TX(-2:205)
!HPF$ DISTRIBUTE TX(BLOCK) ONTO P
!HPF$ ALIGN X(i) WITH TX(1*i+0)

      ...
```

Translation involves

- For each processor array create a *grp* value.
- For each template dimension, create a *rng* value.
- For each partitioned array, create a *dad* value.

DAD Generation

Generated code is

```
 pcrc_shp_P(1) = 2
 pcrc_grp_P = pcrc_new_group_grid (1,pcrc_shp_P)
 pcrc_rng_TX(1) = pcrc_new_range_distribute ((-2),205,1,pcrc_grp_P,&
&1)
 pcrc_dad_X = pcrc_new_array_data (X,pcrc_real,pcrc_size_real,1,pcr&
&c_grp_P)
 call pcrc_set_array_align (pcrc_dad_X,1,1,205,1,0,0,0,pcrc_rng_TX(&
&1))

 ...

 call pcrc_delete_array (pcrc_dad_X)
 call pcrc_delete_range (pcrc_rng_TX(1))
 call pcrc_delete_group (pcrc_grp_P)
```

Assignments

Now add

```
      REAL Y(-12:161)
!HPF$ TY(-17:190)
!HPF$ DISTRIBUTE TY(BLOCK) ONTO P
!HPF$ ALIGN Y(i) WITH TY(1*i-12)
      ...
      FORALL (i=8:112:1) X(i) = Y(1*i-1)
```

Note

- A shift communication is needed. Local `Y` segment extended 2 elements to include ghost area.
- `pcrc_write_halo` communicates edge data.
- Local loop bounds calculated from forall index range and DAD of *lhs*.
- `pcrc_coef` gives address coefficients for each array dimension.

Assignments

Generated code is

```
 pcrc_irg0 = pcrc_new_range_loop (8,112,1,1,0,pcrc_range (pcrc_dad_&
&X,1))
 call pcrc_loop_bounds (pcrc_irg0,pcrc_lil_i,pcrc_liu_i,pcrc_lis_i)

 pcrc_sdd0 = pcrc_new_array_section (1,pcrc_dad_X)
 call pcrc_set_array_triplet (pcrc_sdd0,1,8,112,1,pcrc_dad_X,1)
 pcrc_gtl_Y(1) = 0
 pcrc_gtu_Y(1) = 2
 call pcrc_write_halo (pcrc_dad_Y,pcrc_gtl_Y,pcrc_gtu_Y)
 call pcrc_coef (pcrc_dad_X,1,8,112,1,1,0,0,pcrc_u00,pcrc_v00)
 call pcrc_coef (pcrc_dad_Y,1,8,112,1,1,(-1),2,pcrc_u10,pcrc_v10)
 if (pcrc_on (pcrc_group (pcrc_sdd0))) then
    do  i=0,(pcrc_liu_i-pcrc_lil_i)/pcrc_lis_i,1
       pcrc_sdx1 = pcrc_v10+pcrc_u10*i
       pcrc_sdx0 = pcrc_v00+pcrc_u00*i
       X(pcrc_sdx0) = Y(pcrc_sdx1)
    enddo
 endif
 call pcrc_delete_array (pcrc_sdd0)
 call pcrc_delete_range (pcrc_irg0)
```

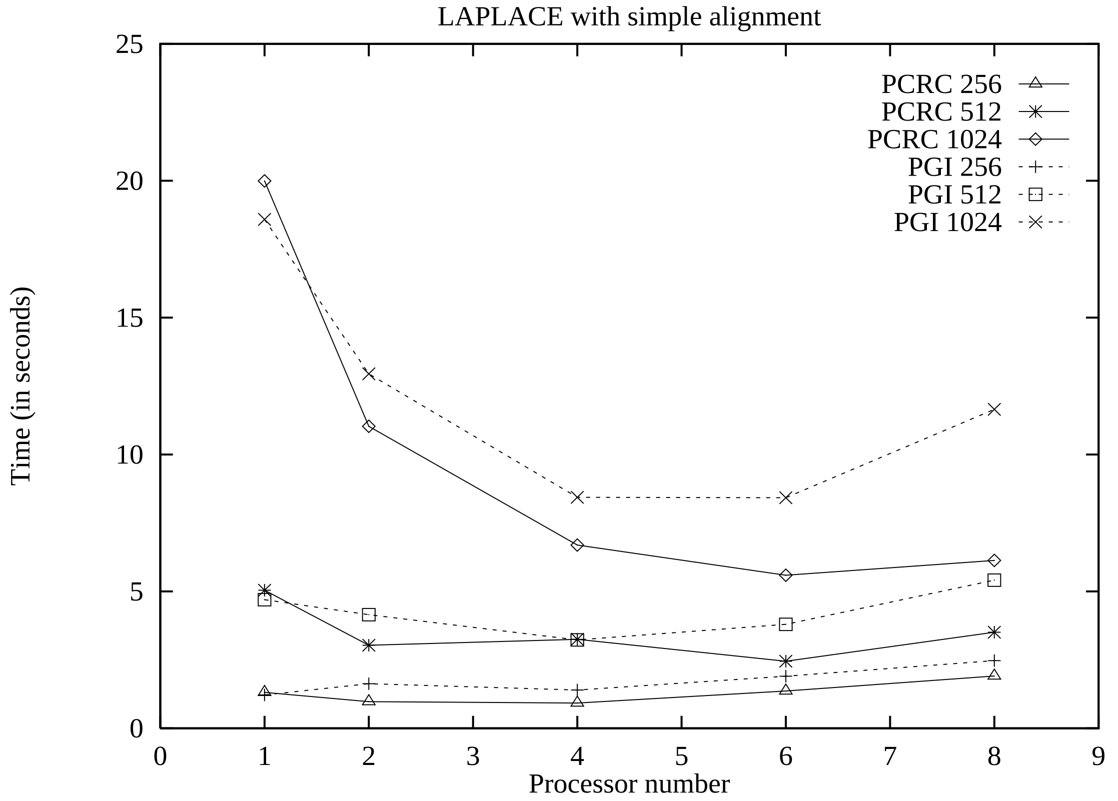
Preliminary Benchmarks

Compared the new NPAC compiler with the PGI HPF compiler, version 2.0. The programs were run on the IBM SP2 at NPAC.

The Laplace benchmark performs Jacobi relaxation on 256×256, 512×512 and 1024×1024 arrays, distributed blockwise.

Compilers achieve about the same performance on a single node, but generally our compiler exhibits better speedup on multiple processors, presumably due to more effective handling of communication.

Laplace Update



Software Availability

- HPF front end (not full translator) is available through

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

- Runtime kernel is being documented, for release later this summer.