Intermediate form

The Adhoc language is a human-readable rendering of an intermediate form for HPF. It is modelled on C++.

```
      REAL A (10, 10)
!HPF$ DISTRIBUTE A (BLOCK, BLOCK)
      REAL B (10)
!HPF$ ALIGN B WITH A ( : , *)

      FORALL (I = 1 : 10, J = 1 : 10)
        A (I, J) = B (I) + FUN (J)
      END FORALL
```

becomes

```
  range i_a(10 #blk p_1) ;
  range j_a(10 #blk p_1) ;
  array float a(i_a, j_a) ;
  array float b(i_a) ;

  where(i_a)
    where(j_a)
      a = b + fun(j_a) ;
```

assuming the obvious mapping of computation.

Communication

Simple communications performed by Adhoc primitive `remap`.

```
      REAL U (12)
!HPF$ DISTRIBUTE U (BLOCK)

      FORALL (I = 2 : 11)
        U (I) = (U (I - 1) + U (I + 1)) / 2
      END FORALL
```

could translate to

```
  range i_u(12 # t_1) ;
  array float u [i] ;

  {
    range i_internal(10 : base 1 : i_u) ;
    array float n [i_internal] ;
    array float p [i_internal] ;

    remap(n, u [10 : base 0]) ;
    remap(p, u [10 : base 2]) ;
    where(i_internal)
      u = 0.5 * (n + p) ;
  }
```

`remap` copies data between two arrays of the same shape—they can have *any* legitimate HPF mapping.

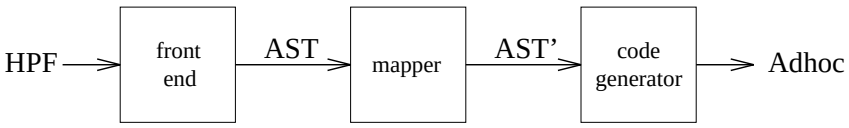
Also provides *gather* and *scatter* primitives to handle communications implied by non-linear subscripts.

Run time library

Implementation of RTL now reduced to implementing the data classes and communication primitives of Adhoc.

Currently being implemented in C++. F90 interfaces will be provided.

Translator



- *front end*—syntactic and static semantic analysis of the input HPF.
- *mapper*—"expression mapping".
- *code generator*—convert AST to Adhoc-based intermediate form with explicit communication and distributed loops, then target language.

Mapping Temporaries

A central problem in HPF translation is mapping of intermediate results, ie, subexpressions. Especially difficult in FORALL.

Are developing schemes applicable to general FORALL statements, with no reliance on "Owner-Computes" type rules—assignment not treated specially. Mapping scheme specified in Prolog.

[Now appears that initial demonstrator system will probably omit this sophistication.]

Example mapping

Input code...

```
REAL B (N)
REAL A (N, N)

FORALL (I = 1 : N, J = 1 : N)
  a (I, J) = FUN1 (I) + (b (J) + FUN2 (J))
ENDFORALL
```

translation...

```
array float b [i_b]
array float a [i_a, j_a]

array float temp1 [i_b] ;
// temp1 = b [j] + fun2(j) ;
where(i_b)
  temp1 = b + fun2(i_b) ;

array float temp2 [j_a] ;
remap(temp2, temp1) ;       // communication !

// a [i, j] = fun1(i) + temp2 [j] ;
where(i_a) {
  float temp3 = fun1(i_a) ;
  where(i_b)
    a = temp3 + temp2 ;
}
```