University of Southampton

The Adlib library

Dr Bryan Carpenter

HPC Centre
University of Southampton
Southampton SO9 5NH
United Kingdom

dbc@uk.ac.soton.ecs

History

Local precursors

- *ADLIB*—run-time library of *ADAPT* system.
- *ACL*—"Array Communication Library".

Adlib

The current *Adlib* combines ideas from both these (and from HPF).

- ACL-like framework—abstract data types used to describe distributed data and process topology in an integrated way.
- Communication functionality enhanced to a superset of that in the original ADLIB, to fully support translation of distributed Fortran.
- Data distribution formats enhanced to support complete HPF.

Adlib is implemented as a C++ class library. There is a reduced interface to Fortran.

C++ interface

Example

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

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

translation

```
Range i(10, p_1, blk), j(10, blk, p_2) ;
Array2<float> a(i, j) ;
Array1<float> b(i) ;

for(i.where() ; i ; i++) {
  for(j.where() ; j ; j++) {
    a(i, j) = b(i) + fun(j()) ;
  } j_a.erewh() ;
} i.erewh() ;
```

Adlib Communication Primitives

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

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

A couple of communication primitives are needed for IO, and some extra support is needed to implement F90 array intrinsics.

Run-time support for formatting Fortran IO will be lifted from existing sequential Fortran translators.

Communication in Adlib

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

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

translation

```
Range i(12, p_1, blk) ;
Array1<float> u(i) ;

Range i_mid(10, 1, 1, i) ;
Array1<float> n(i_mid), p(i_mid) ;

remap(n, Section1<float>(u, sect(Range(10, 0, 1, i))) ;
remap(p, Section1<float>(u, sect(Range(10, 2, 1, i))) ;
for(i_mid.where() ; i_mid ; i_mid++) {
  u(i_mid) = (n(i_mid) + p(i_mid)) / 2 ;
} i_mid.erewh() ;
```

Fortran interface

```
SUBROUTINE AD_INIT_ADLIB()
SUBROUTINE AD_FINALIZE_ADLIB()

SUBROUTINE AD_INIT_PROCS(procs_handle, rank, shape)
SUBROUTINE AD_PROC_COORD(procs_handle, coords)

SUBROUTINE AD_REMAP(dst_data, dst_attribute_vector,
                    src_data, src_attribute_vector)

...
```