Library-Based Approaches to Task Parallelism

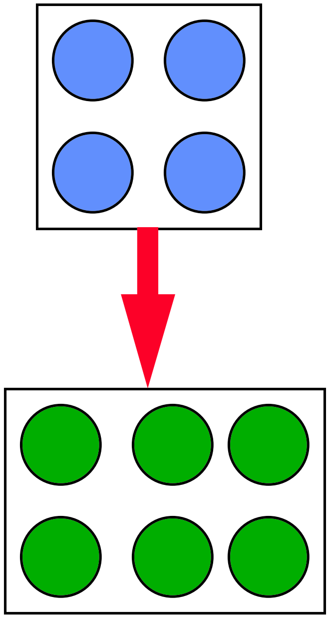
Ian Foster

Argonne National Laboratory

Joint work with David Kohr and Rakesh Krishnaiyer

Task and Data Parallelism

- Mixed task/data-parallel computations
  - Multiblock codes
  - Pipelines (e.g., image processing)
- Coupling of separately compiled data-parallel programs
  - Multidisciplinary simulations
  - Simulation + graphics, computational steering



Approaches to Integration

- Implicit, compiler-based approaches
  - Extract TP algorithms from DP specifications
  - E.g., Fx, PARADIGM
  - Difficulty: Complex compilers + execution model
- Explicit, language-based approaches
  - Explicit TP constructs
  - E.g., Fortran M, Opus, AVS
  - Difficulty: Lack of consensus on constructs

Library-Based Approaches

- HPF system can create multiple tasks
- Each task executes an HPF program
- HPF program calls coordination library functions to exchange data with other tasks
- Library handles the transfer of data from one task to another

Example: An HPF Binding for MPI

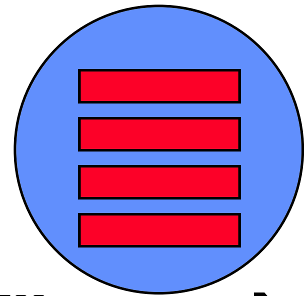
- MPI: A standard coordination library
  - Point-to-point + collective communications
  - Currently, all "bindings" assume 1 process/task
  - But no reason why we should not define a "data-parallel binding"
- HPF/MPI: An HPF binding for MPI
  - Syntactically identical to Fortran 90 binding (?)
  - Complexities lie in the implementation

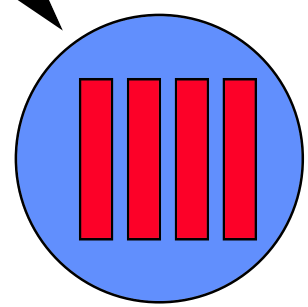
Example: 2-D FFT

```
$hpf PROCESSORS pr(8)
     complex A(8,8)
$hpf distribute A(BLOCK,*)
     do i=1,NBLOCKS
       call read(A)
       call rowfft(8,8,A)
       A = transpose(A)
       call rowfft(8,8,A)
       call write(A)
     enddo
```

Example: Pipelined 2-D FFT

```
$hpf PROCESSORS pr(4)
     complex A(8,8)
$hpf distribute A(BLOCK,*)
     do i=1,NBLOCKS
       call read(A)
       call rowfft(8,8,A)
       call MPI_Send(A,8*8,MPI_COMPLEX,
                 1,99,MPI_COMM_WORLD,ier)
     enddo
```



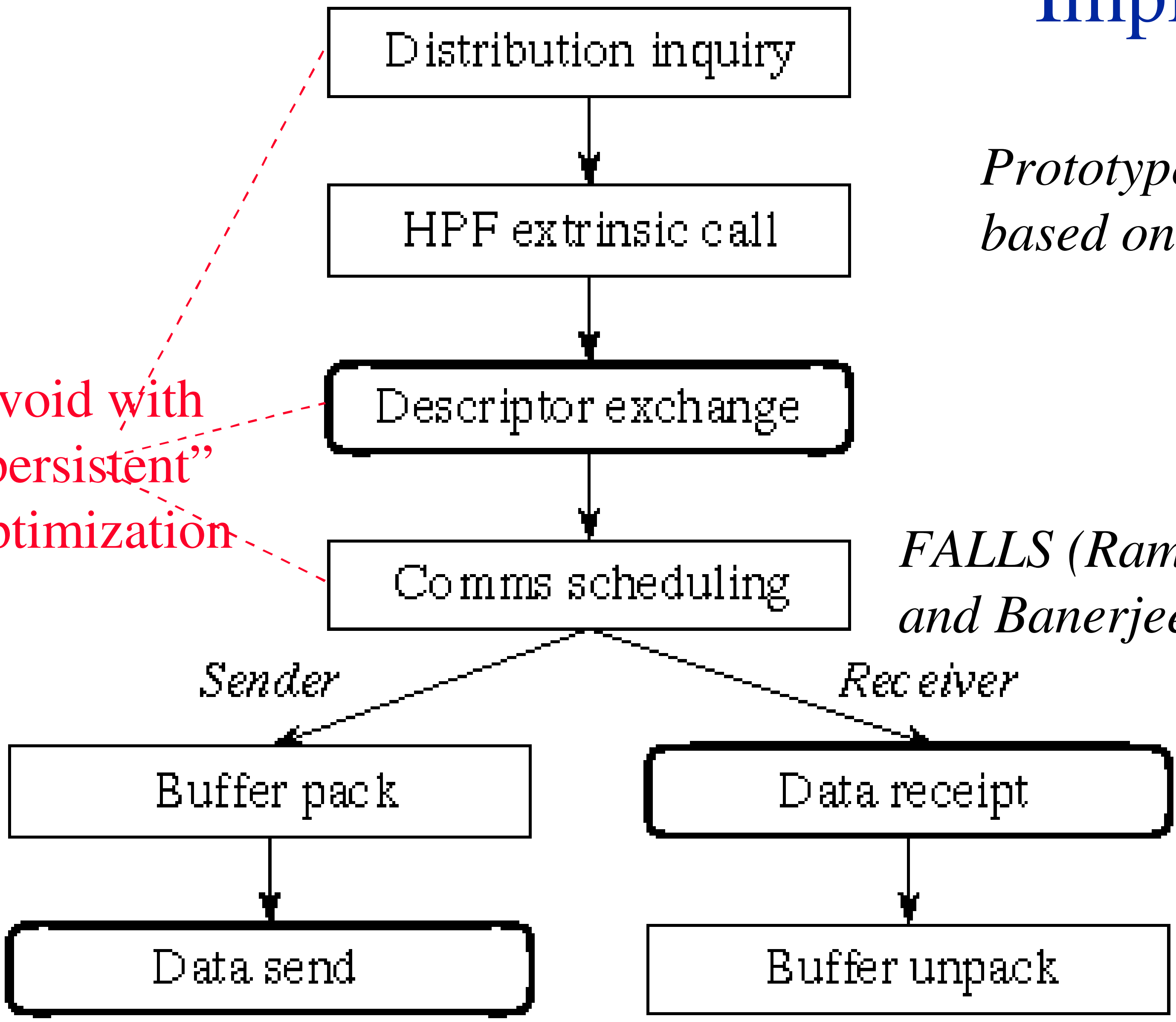



```
$hpf PROCESSORS pr(4)
     complex B(8,8)
$hpf distribute B(*,BLOCK)
     do i=1,NBLOCKS
       call MPI_Recv(B,8*8,MPI_COMPLEX,
          0,99,MPI_COMM_WORLD,stat,ier)
       call colfft(8,8,B)
       call write(B)
     enddo
```

We create two HPF tasks; each executes on four processors

Implementation

Prototype implementation based on PGI compiler



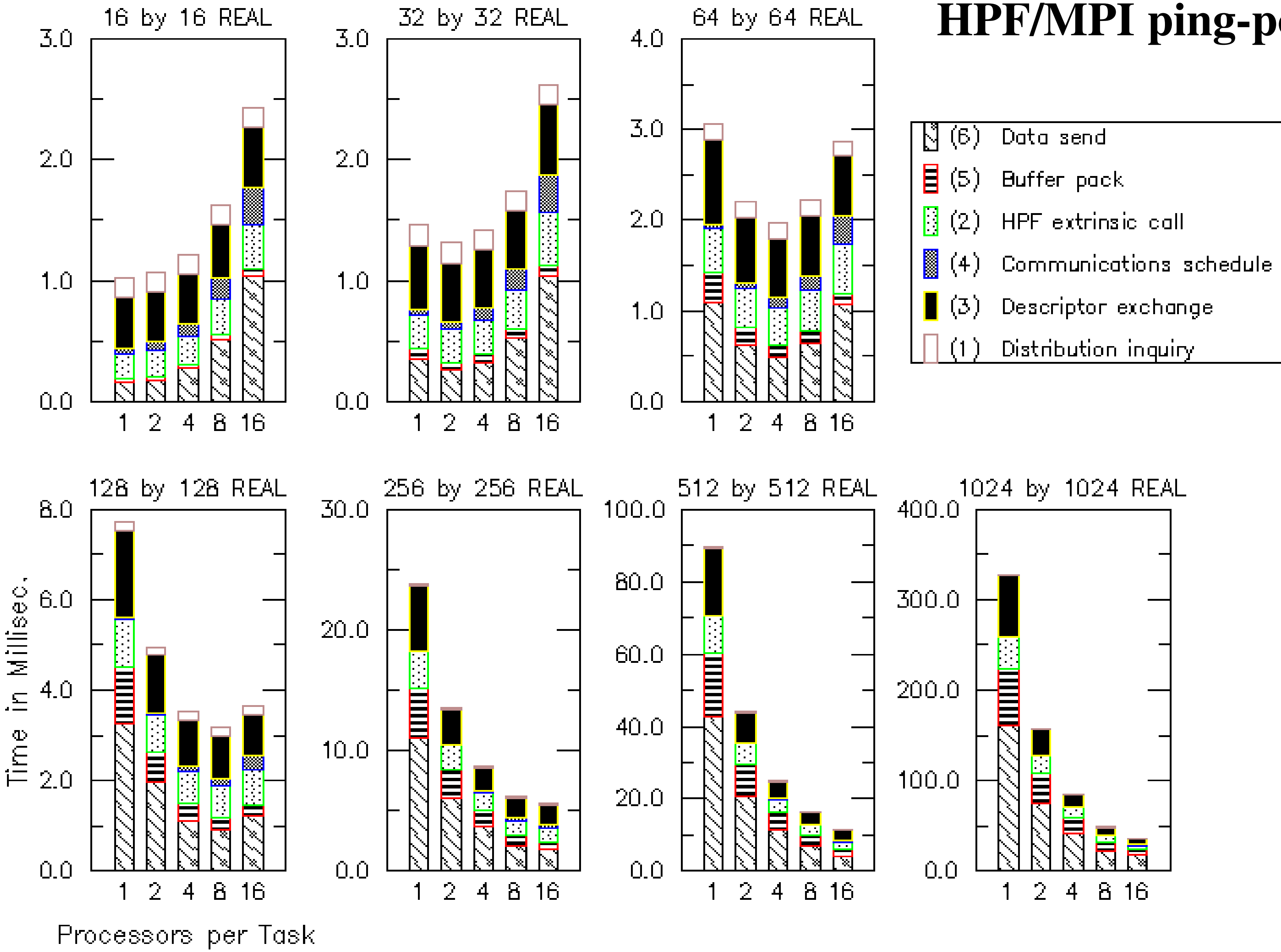
Avoid with "persistent" optimization

FALLS (Ramaswamy and Banerjee)

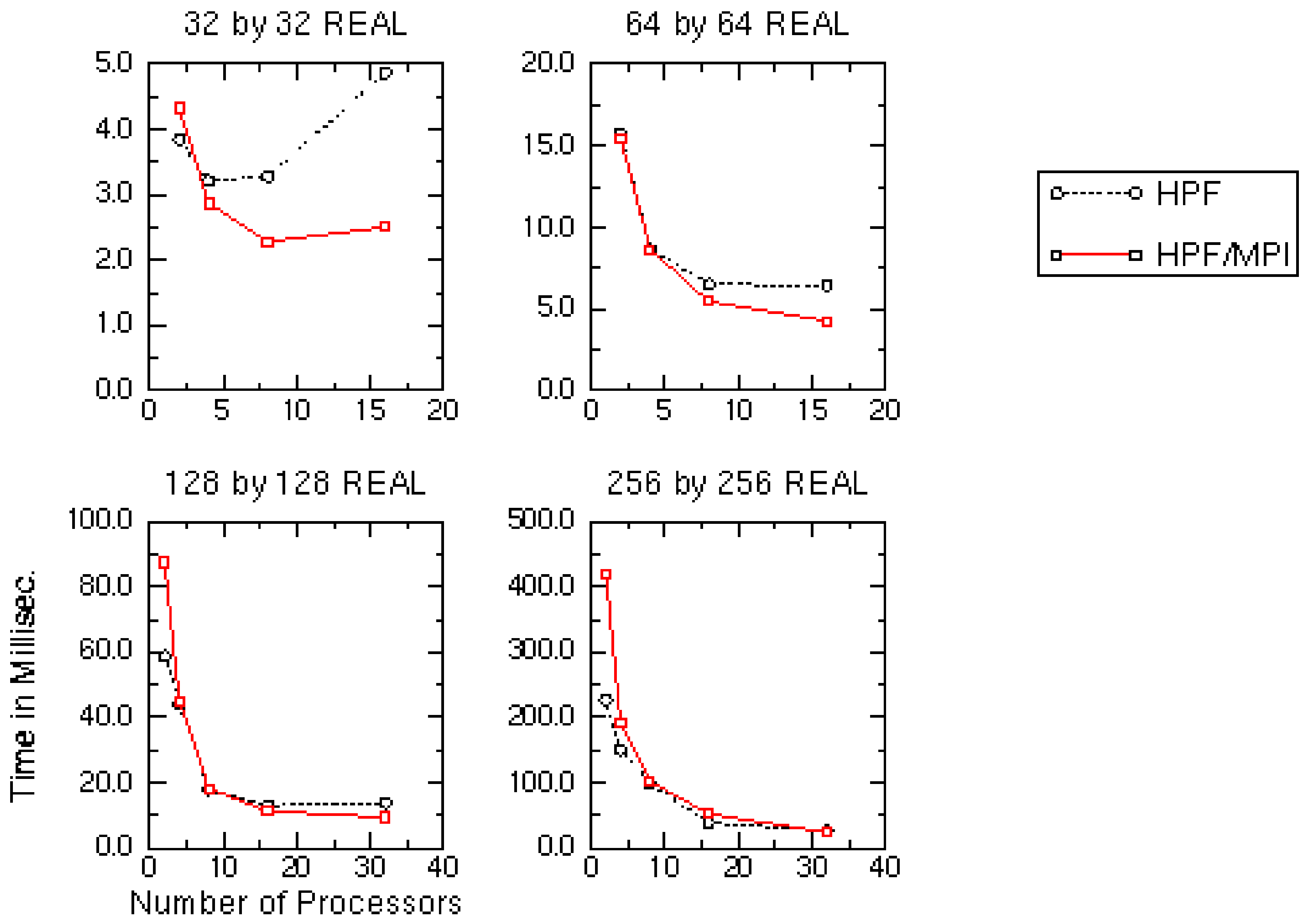
Performance Studies

- "Ping-pong" microbenchmark
  - (BLOCK, *) --> (*, BLOCK)
- Application studies
  - 2-D FFT, 2-D Convolution, Multiblock
  - All using persistent communication optimization
- Experiments on Argonne SP2 (Power 1 nodes) using modified PGI HPF compiler

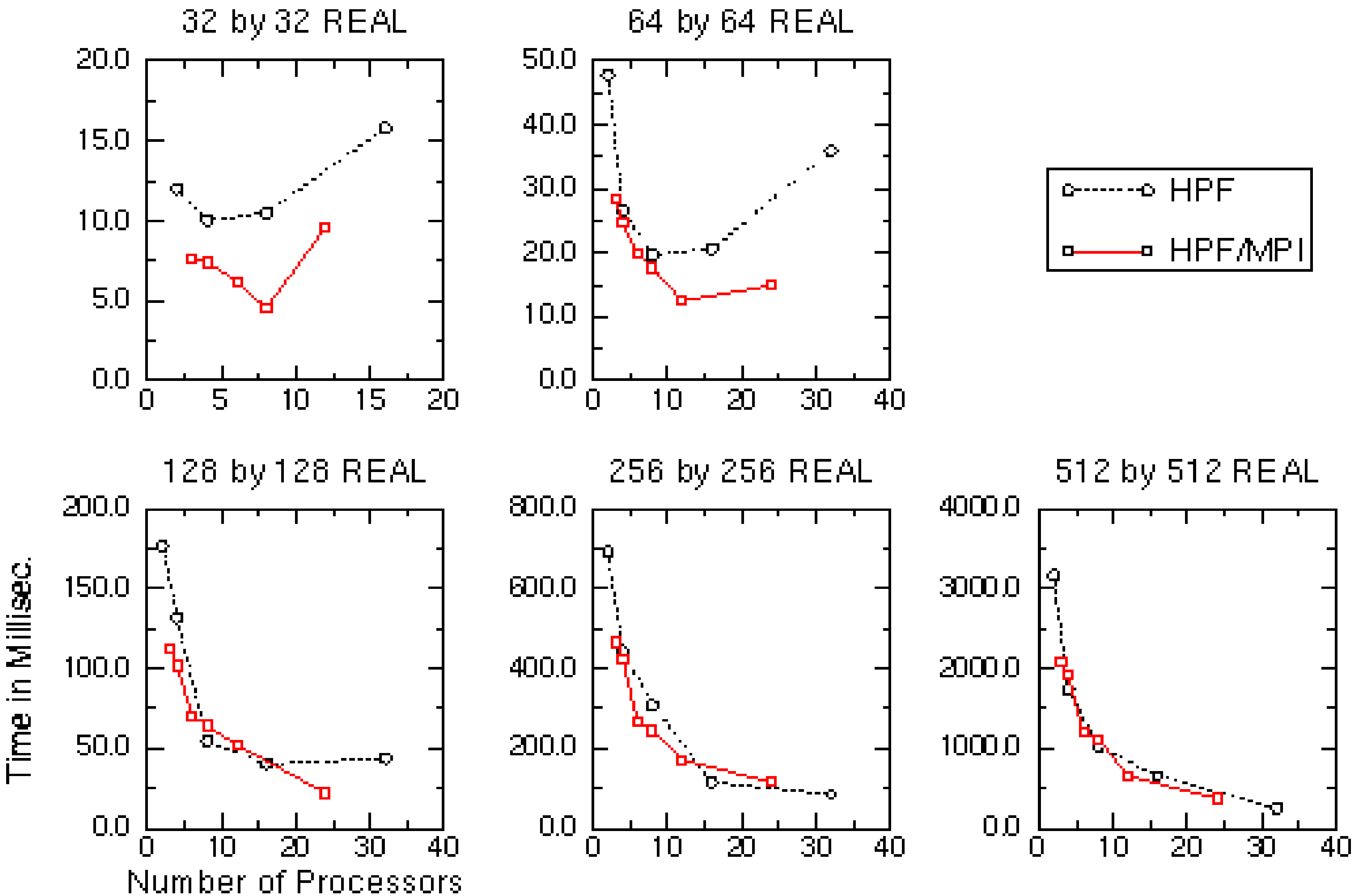
HPF/MPI ping-pong



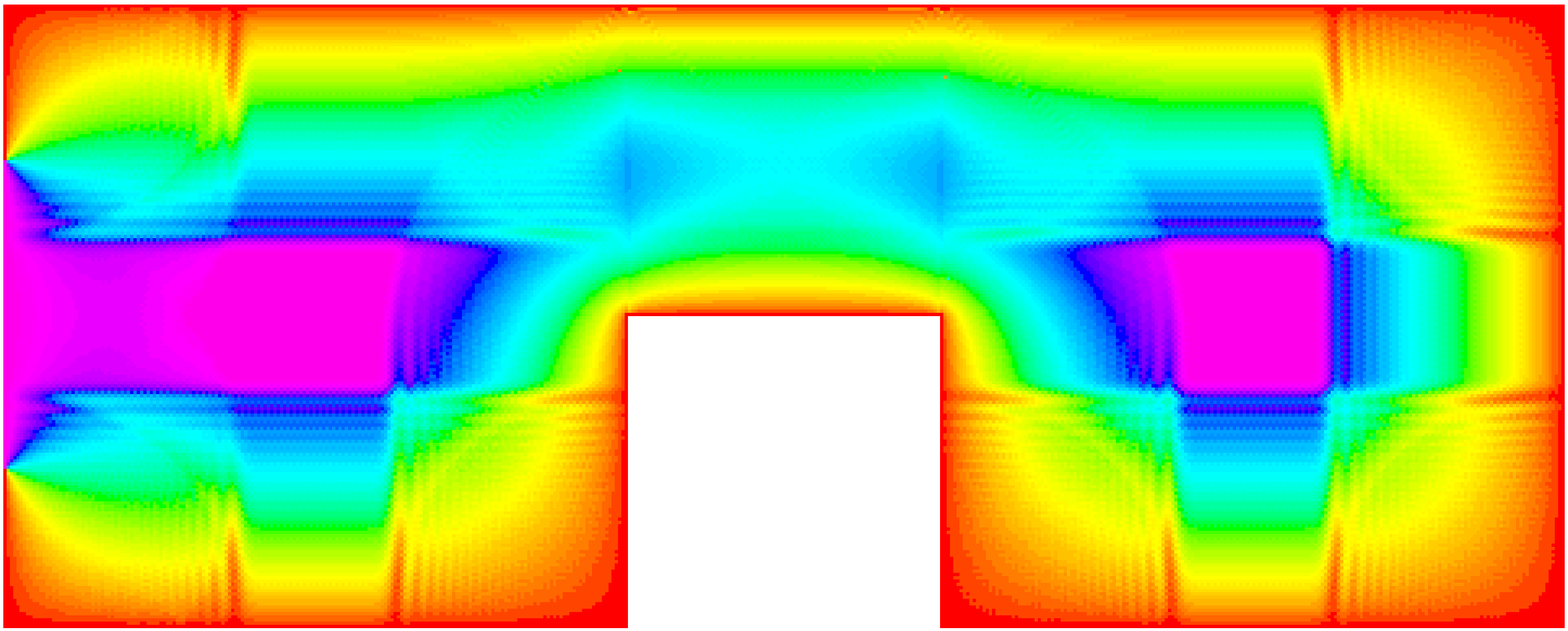
Performance of data-parallel and task/data-parallel 2-D FFT



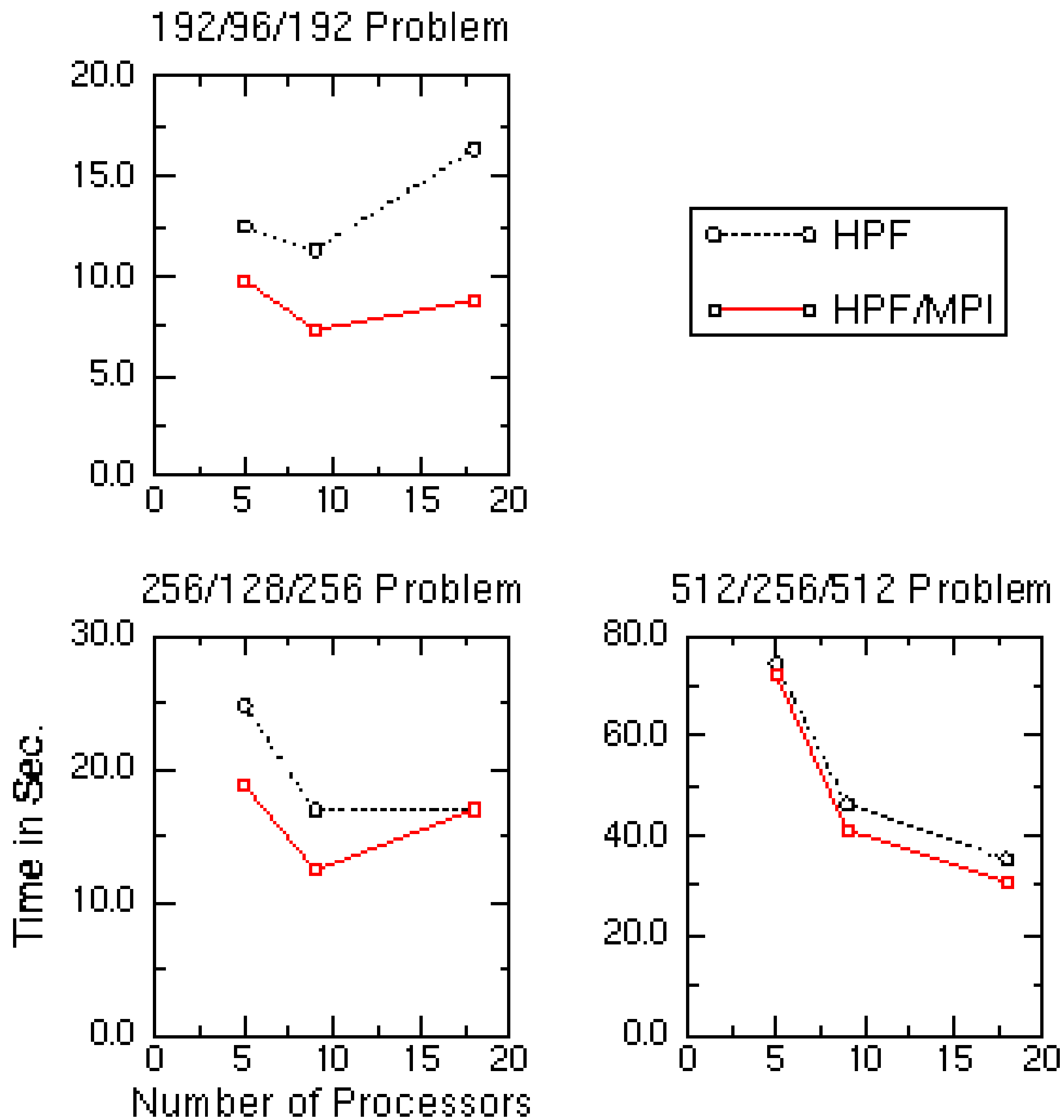
Performance of data-parallel and task/data-parallel 2-D convolution



Multiblock configuration used for experiments



Performance of data-parallel and task/data-parallel 3-block



Future Directions

- Can we improve performance by a tighter coupling of HPF and MPI?
- Can we apply the same techniques to other interaction models? (RPC, put/get, ...)
- Can we support new MPI features: dynamic processes, attachment?
- Can we apply the same techniques within HPF-2 task regions?
- Can we exploit “library-smart” compilers?

Summary

- Coordination libraries can extend scope of HPF without new compiler technology
- HPF/MPI library can be implemented without excessive complexity
- Can achieve performance superior to pure HPF in interesting applications

http://www.mcs.anl.gov/fortran-m/

Future application: monitoring and steering CHAD automotive modeling code