COMPILER AND RUNTIME SUPPORT FOR OUT-OF-CORE HPF PROGRAMS

Rajeev Thakur
Rajesh Bordawekar
Alok Choudhary

Dept. of Electrical and Computer Eng. and Northeast Parallel Architectures Center Syracuse University

July 15, 1994
Outline

- Introduction
- Need for High Performance Parallel I/O
- Model for Compilation
- Compiler Design
- Runtime Support
- Performance on Intel Touchstone Delta
- Conclusions
# I/O requirements for GC apps.

## A. Environmental and Earth Sciences:

<table>
<thead>
<tr>
<th>Application</th>
<th>I/O requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environmental modeling</td>
<td>T: 10s of GB. S: 100s of MB - 1 GB per PE. A: Order of 1 TB.</td>
</tr>
<tr>
<td>Eulerian air-quality modeling</td>
<td>S: Current 1 GB/model, 100 GB/application; projected 1 TB/application. A: 10 TB at 100 model runs/application.</td>
</tr>
<tr>
<td>Earth system model</td>
<td>S: 108 MB/simulated day. 100 GB/decade-long simulation. B: 100 MB/s.</td>
</tr>
<tr>
<td>4-D data assimilation</td>
<td>S: 100 MB - 1 GB/run. A: 3 TB database. Expected to increase by orders of magnitude with the Earth Observing System (EOS) - 1 TB/day.</td>
</tr>
<tr>
<td>Ocean-Atmosphere Climate modeling</td>
<td>S: 100 GB/run (current). B: 100 MB/sec. A: 100s of TB.</td>
</tr>
</tbody>
</table>

A: Archival Storage  
T: Temporary Working Storage  
S: Secondary Storage  
B: I/O Bandwidth

(Ref: del Rosario and Choudhary: Computer, March 94)
### B. Computational Physics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Solar Activity and</td>
<td>B: 200 MB/sec</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heliospheric Dynamics</td>
<td>A: Up to 500 GB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convective turbulence in</td>
<td>S: 5-10 GB/run</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Astrophysics</td>
<td>B: 10-100 MB/s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Particle algorithms in</td>
<td>S: 1-10 GB/file; 10-100 files/run.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cosmology and Astrophysics</td>
<td>B: 20-200 MB/s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radio synthesis imaging</td>
<td>S: 1-10 GB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A: 1 TB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Compilation Issues

- Data Parallel Language like HPF.
- Out-of-core program due to large data set.
- Most MPPs do not provide virtual memory on nodes.
- Even if virtual memory is provided, performance is not good.
- True even in the case of sequential machines.
- Hence, translate to message passing node program with explicit calls to I/O routines.
Architectural Model

Processors  Network  I/O Nodes  Disks
- Local Array of each processor stored in a separate file called **Local Array File (LAF)**.

- The portion of the local array in memory is called the **In-Core Local Array (ICLA)**.

- Parts of the LAF have to be fetched into the ICLA and then stored back on disk after the computation.
Example

- Consider the following HPF program fragment

```
PARAMETER (N=1024)
REAL A(N,N), B(N,N)

...........
!HPF$ PROCESSORS P(4,4)
!HPF$ TEMPLATE T(N,N)
!HPF$ DISTRIBUT T(BLOCK,BLOCK) ONTO P
!HPF$ ALIGN (I,J) with T(I,J) :: A, B

...........
FORALL (I=1:N, J=1:N)
    A(I,J) = (B(I,J-1) + B(I,J+1) +
    B(I+1,J) + B(I-1,J))/4

...........
B = A
```
Example contd.

(A)

ARRAY IN P5

-overlap area

(B)

1. 2. 3. 4.

1. 2. 3. 4.

(out-of-core communication)

(D) in-core communication

(C) out-of-core communication

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.

1. 2. 3. 4.
Out-of-core Communication

1. Stripmine code based on memory size.
2. Schedule communication for entire out-of-core data.
3. Repeat $k$ times ($k$ is the stripmine factor).
   3.1 Read data from disk to ICLA.
   3.2 Do the computation on the data in ICLA.
   3.3 Write data from ICLA back to disk.

In-core Communication

1. Stripmine code based on memory size.
2. Repeat $k$ times ($k$ is the stripmine factor).
   2.1 Read data from disk to ICLA.
   2.2 Schedule communication for in-core data.
   2.3 Do the computation on the data in ICLA.
   2.4 Write data from ICLA back to disk.
Communication Issues

- **Explicit Communication**: Each processor accesses only its local files and sends/receives data explicitly.

- **Direct File Access**: Communication is performed by processors reading each other's files.

![Diagram of communication between processors and disks with simultaneous accesses to disk](image-url)
I/O Optimizations

• Data Reuse

• Data Prefetching
  ◦ Prefetch the next slab immediately after current slab is read.
  ◦ Overlaps computation on current slab with the reading of the next slab.
Runtime Routines

- **Array Management Routines**
  - read_vec
  - write_vec
  - read_vec_with_reuse
  - read_vec_with_prefetch
  - prefetch_wait
  - ...

- **Communication Routines**
  - out_of_core_shift
  - out_of_core_multicast
  - ...
# Out-of-Core Array Descriptor (OCAD)

<table>
<thead>
<tr>
<th>Dimension</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incore lb</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incore ub</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incore lbo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incore ubo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global sz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCLA size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OOC storage</td>
<td></td>
<td>x</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distribution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block sz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Translated Code

do k=1 to no_of_iterations
    call oc_shift(unit1, OCAD, 1, right) ! right shift
    call oc_shift(unit1, OCAD, 1, left) ! left shift
    do l=1, no_of_slabs
        call read_vec(unit1, B, OCAD, i, j, start_vec,
                       end_vec, stride)
        do j=j1, j2
            do i=i1, i2
                A(i,j) = (B(i,j-1) + B(i,j+1) +
                         B(i+1,j) + B(i-1,j))/4
            end do
        end do
    end do
    call write_vec(unit2, A, OCAD, i, j, start_vec,
                   end_vec, stride)
end do

C    exchange file unit numbers instead of explicitly
C    copying files (optimization)
    unit1 ↔ unit2
end do
Preliminary Results

Laplace Equation Solver

- time in sec. for 10 iterations
- 64 processors
- Slab Size = $1/4^{th}$ of local array size

<table>
<thead>
<tr>
<th></th>
<th>$2K \times 2K$</th>
<th>$4K \times 4K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct File Access</td>
<td>79.12</td>
<td>280.8</td>
</tr>
<tr>
<td>Explicit Communication</td>
<td>75.12</td>
<td>274.7</td>
</tr>
<tr>
<td>Explicit Communication with data reuse</td>
<td>71.71</td>
<td>269.1</td>
</tr>
</tbody>
</table>

LU Decomposition (1K $\times$ 1K array)

<table>
<thead>
<tr>
<th>Processors</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec.)</td>
<td>1256.5</td>
<td>1113.9</td>
<td>1054.5</td>
</tr>
</tbody>
</table>
Conclusions

- Important first step in techniques for translating out-of-core data parallel programs.
- Lot of open issues, for example:
  - Local array file v/s single global array file.
  - In-core communication v/s out-of-core communication.
  - Explicit communication v/s direct file access.
  - Interface between compiler and runtime library.
  - Interface between runtime library and file system.