Parallelization of HIRLAM

Model Parallelization and Asynchronous I/O
HIRLAM Optimizations

• Two major projects:
  ① Model scalability improvement
  ⑥ Asynchronous I/O optimization

• Both projects funded by DMI and NEC
HIRLAM Parallelization

Message Passing Optimization
HIRLAM Scalability Optimization

- Methods
- Implementation
- Performance
Optimization Focus

- Data transposition
  - from 2D to FFT distribution and reverse
  - from FFT to TRI distribution and reverse
- Exchange of halo points
  - between north and south
  - between east and west
Approach

• First attempt: straight-forward conversion from SHMEM to MPI-2 put/get calls
  – It works, but:
  – Too much overhead due to fine granularity
• Original plan shattered, now what?
  – Panic (mild form)
• More to do: in-depth analysis of how things really work
• Human memory: retention at least 6 years
Approach

• Redesign of transposition based on method used in initiative by met.no called PARLAM (Dag Bjørge and Roar Skålin, 1995)
• Redesign of halo swap routines
• Benefits:
  ➤ less and larger messages
  ➤ independent message passing process groups
2D Sub Grids

- HIRLAM sub grid definition in TWOD data distribution
- Processors:

\[ nproc = nprocx \cdot nprocy \]
Original FFT Sub Grids

- HIRLAM sub grid definition in FFT data distribution
- Each processor handles slabs of full longitude lines
2D↔FFT Redistribution

Sub grid data to be distributed to all processors:

\( nproc^2 \) send-receive pairs
2D\leftrightarrow\text{FFT Redistribution}

- Sub grids in east-west direction form full longitude lines
- $nprocy$ independent sets of $nprocx^2$ send-receive pairs
- Total nr of pairs: $nprocy \cdot nprocx^2$
  or: $nproc^2 / nprocy$
- $nprocy$ times less messages
Transpositions 2D ↔ FFT ↔ TRI

2D ↔ FFT ↔ TRI
**MPI Methods**

- Three transfer methods tried:
  - Remote Memory Access: `mpi_put, mpi_get`
  - Async Point-to-Point: `mpi_isend, mpi_irecv`
  - All-to-All: `mpi_alltoallv, mpi_alltoallw`

- Buffering vs. direct
  - Explicit buffering
  - *MPI derived types*

(Method selection by environment variables)
SHMEM Support

• Integrated in new design
• Not much specific SHMEM code left: same buffer structures used for MPI and SHMEM
• Very similar to MPI-2 RMA code part
• New SHMEM code ported to SHMEM architecture by Ole Vignes, met.no
# Performance

<table>
<thead>
<tr>
<th>Test grid Details</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitude</td>
<td>602</td>
</tr>
<tr>
<td>Latitude</td>
<td>568</td>
</tr>
<tr>
<td>Vertical</td>
<td>60</td>
</tr>
<tr>
<td>NSTOP</td>
<td>40</td>
</tr>
<tr>
<td>Initialization</td>
<td>none</td>
</tr>
<tr>
<td>Time step</td>
<td>180</td>
</tr>
<tr>
<td>Time step</td>
<td></td>
</tr>
</tbody>
</table>

**Performance Test Grid Details:**

- **Longitude:** 602 points
- **Latitude:** 568 points
- **Vertical:** 60 levels
- **NSTOP:** 40 steps
- **Initialization:** None
- **Time step:** 180 seconds
Parallel Speedup on NEC SX-6

- Cluster of 8 NEC SX-6 nodes at DMI
- Up to 60 processors:
  - 7 nodes with 8 processors per node
  - 1 node with 4 processors
- Parallel efficiency 78% on 60 processors
Performance – Observations on SX-6

• New data redistribution method much more efficient (78% vs. 45% on 60 processors)

• No performance advantage with RMA (one-sided MP) or All-to-All over plain Point-to-Point method

• Elegant code with MPI derived types, but explicit buffering faster
Performance – Observations on SGI Origin

• Tests executed at met.no with grid size 468x378x40 points on 196 processors

• Time step times:
  – Old SHMEM: 1.4 s
  – New SHMEM: 1.15 s
  – MPI: 0.85 s

• Conclusions regarding SHMEM:
  – New SHMEM code faster than old SHMEM code
  – New MPI code faster than new SHMEM code
  – End of SHMEM version?
HIRLAM GRIBfile Server

Asynchronous I/O
HGS Overview

HIRLAM GRIBfile Server

• Purpose: *Take both input and output processing out of the time stepping loop*

• *Server* is a virtual concept, not a separate application or system

• Integrated in model source

• Enabled/disabled at runtime
HGS Overview

Brief history
1. HGS originally written for shared memory architecture (IPC) (Sun, Jan Boerhout, 2001)
2. Similar initiative for distributed memory architecture (MPI), output only (CSC, Jussi Heikonen and FMI, Kalle Eerola, 2001)
3. HGS rewrite for MPI, based on ideas from 1 and 2 (met.no, Ole Vignes, 2002)
4. Recently optimized for faster data exchange (NEC, Jan Boerhout, 2004; funded by DMI)
HIRLAM I/O Activity - Sequential

- 3 hours forecast
- 610 x 568 x 40 points
- 15 SX-6 processors (8GF peak)
- Hourly input and output
- Time step 360 seconds
- Graph shows function call stack on time line
HIRLAM I/O Activity - Asynchronous

Disk Input → Disk Output

Time (s)

0  20  40  60  80  100  120

HGS

Trace 2
vtrace_14_0.vfd

Model

Trace 1
vtrace_1_0.vfd

Model

Trace 0
vtrace_0_0.vfd

Model Input → Model Output
HIRLAM I/O Activity - Asynchronous

HGS:
1. Receives preread request and reads first input file
2. Reads second input file

Model:
A. Sends HGS request to pre-read first two input files and waits for HGS availability confirmation
HGS:
3. Sends data from first two input files

Model:
B. Receives data from first two input files
HGS:
4. Collects data for first output files

Model:
C. First time step
D. Sends data for first three output files
HIRLAM Model and HGS Animation

- Demonstration of output (input very similar)
  - Four model sub grids
  - One HGS process
  - Buffer block size: 4 fields (in reality: 100)

- Model running...

- Next slides: model writes output, HGS collects and stores the data
HGS Collects Output

GRIBfile Server

ECMWF Workshop 2004 - Parallelisation of HIRLAM
HGS Collects Output

GRIBfile Server

ECMWF Workshop 2004 - Parallelisation of HIRLAM
HGS Collects Output

GRIBfile Server
HIRLAM I/O Activity - Output

Output:

*Short model communication time*

Will be even shorter with DMI’s new MSLP algorithm
**HIRLAM I/O Activity - Asynchronous**

**HGS:**

5. Writes output files to disk

**Model:**

E. Sends requests to store output and pre-read next input, then continues
Model continues while HGS Writes to Disk
Model continues while HGS Writes to Disk

GRIBfile Server
Model continues while HGS Writes to Disk

GRIBfile Server
Model continues while HGS Writes to Disk
Model continues while HGS Writes to Disk
Model continues while HGS Writes to Disk

GRIBfile Server
Model continues while HGS Writes to Disk

GRIBfile Server
Model continues while HGS Writes to Disk
Model continues while HGS Writes to Disk
HIRLAM I/O Activity - Asynchronous

HGS:
6. Reads next input file
7. Sends input data

Model:
F. Sends requests to receive input data
G. Next time step
HIRLAM I/O Activity - Input

Input:

*Very short model communication time*
**HIRLAM I/O Activity - Asynchronous**

**HGS:**

8. Collects output data
   (repeat of 4)

**Model:**

H. Sends output data
   (repeat of D)
Multiple HGS Tasks Supported

Multiple HGS tasks: 3 HGS, 12 model (1 shown)
HGS Documentation

- More detailed descriptions in HGS documentation
- To be published as HIRLAM Newsletter, see: http://hirlam.knmi.nl
- Graphs generated with Vftrace, free tool for NEC SX users, see http://vftrace.jboerhout.nl
Questions?

- Thank you!