Parallel MAGIC Development

David N. Smithe, Larry D. Ludeking, Tyler Gray

Mission Research Corporation
8560 Cinderbed Road, Suite 700
Newington, Virginia 22122

MURI teleconference
Wednesday, June 12, 2002
What’s New Since IVEC 2002?

- (Most of this talk is repeat from MUG meeting @ IVEC2002.)
- A lot of interest – not surprisingly, everybody wants simulations to run faster!
- Installation at SLAC on 3 computers was successful, and parallel run capability over local ethernet was demonstrated.
- Mostly Debugging Activity, since.
- More evaluation of MPI library options – MPI-Pro via MS a no-go.
- Discussion with SLAC about MAGIC2D on Linux Cluster.
MAGIC2D Linux version for SLAC?

- SLAC has Existing Intel Linux cluster, and has demonstrated parallel operation with ICEPIC program.
- SLAC may be able to provide programming manpower, Linux and MPI expertise.
- Lahey Compiler and Winteracter graphics both now available for Linux … so it should be possible to make Linux version without too much trouble …
- MRC / SLAC discussing details of a shared source-code development project (rare in the recent history of MAGIC).
- Cooperative effort would speed up implementation, parallelize MAGIC2D and MAGIC3D simultaneously, and improve reliability.
- Goal: 2-D Design Tool!
Overview

- Windows Parallel Processing Environment.
- Domain Decomposition Strategy.
- Distributing the Processes.
- Setting Up a Parallel Run.
- Progress Summary.
- MPI Strategy.
- Boundary Condition Strategy.
- Send / Receive Timing.
- Overall Timing.
Parallel Processing Environment

- **What’s Required?** - MPI Library DLL’s and communications software.
  - For now, download 4-month trial version of WMPI from [http://www.criticalsoftware.com/wmpi/downloads.php](http://www.criticalsoftware.com/wmpi/downloads.php)
  - MAGIC-specific documentation yet to be done …

- **Simple Case** - Dual Processor CPU - would like to double speed.
  - Looks like you’ll get about 0.65*runtime, instead of 0.50*runtime.
  - Unless you’ve got a lot of Output, or are running out of memory !!!

- **Intermediate Case** - A “cluster” of N-identical CPU’s on a fast network.
  - Sure would be nice … but this is a subset of the ...

- **Advanced Case (e.g., Reality)** - Combination of assorted CPU’s on Ethernet.
  - Some single CPU, some double CPU.
  - All CPU’s with different speeds !
Domain Decomposition Strategy

- Spatial region broken into domain-blocks, one domain per process.

- Each domain is essentially a separate MAGIC run.

- This year MAGIC will allow only 1-D decomposition (like above).

- However, the underlying code is written for 3-D domain blocking. This is expected to be available next year, after more important features are brought on-line.
Distributing the Processes

• Let’s just assume the Advanced Case - an odd assortment of PC’s all at different speeds and capabilities.
  – There will be a master “Cluster Configuration File” designating:
    » All the Computer Names and Account Names for remote processes,
    » Communication Method - TCP/IP (always) and Shared Memory (only for Dual CPU’s),
    » Number of processors,
    » Processor speed.
  – Example (my desk): Single CPU at 800 MHz, Dual CPU at 450 MHz.
    » Today, MAGIC separates processes into equal-sized blocks. So the obvious scenario for this example is 4 processes, two on the Single CPU running at 800 MHz, and two on the Dual CPU running at 450 MHz. This choice must be specified in a “Process Group File”, created before the run by the user.
    » Soon, MAGIC will utilize the full information in the Cluster Configuration File, to automatically create the “Process Group File”, and to split the problem into non-equal-sized block.

• There generally is no requirement that there be one domain-block per CPU … in the above example it made sense to have two processes on the faster CPU. There are also load-balancing reasons why one might want more than one process per CPU.
Setting up a Parallel Run

- Setting up the MPI software is the hard part!

- MAGIC automatically runs in parallel if it detects a “Process Group File” in the directory of the master input file. If the file name is “MAGICinput.M3D”, then the “Process Group File” should have the name, “MAGICinput.PG”.

- Add the “PARALLEL” option to one of the AUTOGRID commands
  
  \texttt{AUTOGRID X3 PARALLEL ;}

  If no “Process Group File” is found, then the PARALLEL option is ignored.

- Example: a 4-process “Process Group File” looks like:

  
  \begin{verbatim}
  Moose 1 "C:\Magic Tools\Magic3d_sng.exe" D:\ParallelRun\Test.m3d
  Trace 2 "C:\Magic Tools\Magic3d_sng.exe" F:\ParallelRun\Test.m3d
  \end{verbatim}

  Process #0 (master) and #1 on machine Moose, Process #2 and #3 on machine Trace. The files are on the “D:” hard-drive on Moose, and the “F:” hard-drive on Trace.
Progress Summary (the Good News)

- Last Year - MPI-Based Process Split-up, but demo-message passing via disk.
- This Year - All MPI
- Interactive Output Synchronization - Done (included on this distribution disk).
- Domain Decomposition - Done (included on this distribution disk).
- Dynamic Fields - Done (included on this distribution disk).
- Particles - Done.
- Forces - Done.
- Currents - Nearly Done.
- More Options for Domain Decomposition - Soon.
Progress Summary (the Bad News)

- Eigenmode Synchronization - Soon.
- Single Dump File Reconstruction - Not Done.
- Periodic Boundaries - Not Done, but should be easy!
- Axial Boundary in Angle-Blocked domains - Not Done.
- Cross-Domain Observes - Not Done.
- Cross-Domain Ports - Not Done.
- Cross-Domain Drivers - Not Done.
- Cross-Domain Ranges - Not Done.
- Many other Cross-Domain situations ...
This Year, The Big Four, Plus One

1) Dynamic Fields.

2) Particles.

3) Forces.

4) Currents.

Synchronization for Interactive Output.

These are the main items necessary for a simulation to actually be useful in many cases, even if all parallel features aren’t in place yet.

Also permits comparison and benchmarking with other parallel codes.

- ICEPIC
- OOPIC
Fields Example
Particle Example
Six Processes !!!
MPI Strategy

- A single MAGIC processes is essentially a full-featured independent run, except for an absolute minimum of communication at the block-boundary … and a few other things … so for now, no dynamic load balancing …

- How can we compete, then? By careful coding, designed to insure that the entire ensemble runs exactly as slow as the slowest process … **but no slower**.

- This is accomplished by insuring that All Send/Receives in the main time-step loop are non-blocking (MPI_Isend / MPI_Irecv).

- Sends - Receives always “separated” in chain of events. **CALL MPI_Barrier**

- Better than just “separated”, there is always some large compute intensive task between corresponding Send/Receive, so slowest process is always computing, while communications occur in the background and are complete by the time the compute intensive task is finished. So slowest process never waits.

- So in most cases, MAGIC is Compute-Bound by the slowest process, rather than Communication-Bound, even for common 100BaseT networking.
Implication of Non-Blocking Send / Receive

- Result is *Time-step Asynchronous* operation between processes!

- Each process may be plus/minus one time-step removed from next process in a linear blocking scheme with 1-cell overlap boundaries.

- Example illustrates a situation where the slowest process is #6, fastest processes are #0 and #3.

- Time-step Asynchronism has great advantages, for example, Process #6 may be slow because it’s doing IO this time step - it may speed up again later. In general, a short duration slowdown in one process need not slow the whole ensemble down.

- The drawback - Interactive Output - need to bring a “stampeding herd of processes” to a controlled pause while user looks at a plot of data.
Overlap Strategy

- Single Cell Overlap
- All Particles in Overlap Cell are sent to end of Particle Loop, to allow communication of Overlap Cell Fields during bulk particle push.
- Decision to pass particles to next process, or keep in same process, is made based on sign of momentum.
- Lots of fun in 3-D !!!
Send / Receive Timing

== Begin Main Loop ==

= Compute $\nabla \times B^{n-1/2}$, =
= $\Rightarrow$ Received Message $J^{n-1/2}$, $J_{\parallel}$, $J_{\perp}$, and add to $J^{n-1/2}$, =
= $E^n \leftarrow E^{n-1}$, $\nabla \times B^{n-1/2}$, $J^{n-1/2}$, =
= $\Leftarrow$ Send Message $E^n$, $E_{\parallel}$, =
= Compute $\nabla \times E^n$, =
= $B^{n+1/2} \leftarrow B^{n-1/2}$, $\nabla \times E^n$, =
= $\Rightarrow$ Received Message $E^n$, overwrite $E_{\parallel}$ field, fix $B^{n+1/2}$ in overlap cell =
= Compute all forces, $F^n$, =
= $\Leftarrow$ Send Message $F^n$, $F_{plane}^n$, =
= $J^{n+1/2} = 0$, =
= $\Rightarrow$ Received Message $P^n$, boundary-cell-particles, append to end of particle list, =
= Particle Group Loop =
= IF (group includes boundary-cell-particles) THEN =
= $\Rightarrow$ Received Message $F^n$, $F_{plane}^n$, and overwrite boundary plane forces, $F^n$. =
= ENDIF =
= $p^{n+1/2} \leftarrow p^{n-1/2}$, $F^n$, =
= $x^{n+1} \leftarrow x^n$, $p^{n+1/2}$, =
= $J^{n+1/2} \leftarrow J^{n+1/2}$, $x^{n+1}$, $x^n$, =
= End Particle Group Loop =
= Emit new particles. =
= $J^{n+1/2} \leftarrow J^{n+1/2}$, $x^{n+1}$, $x^{emit}$, =
= $\Leftarrow$ Send Message $J^{n+1/2}$, $J_{\parallel}$, $J_{\perp}$, =
= $\Leftarrow$ Send Message $P^{n+1}$, boundary-cell-particles, =
= Diagnostics, =

== End Main Loop ==
• Compare Single vs. Two Processes.

• Typically Compute-Bound above 20,000 cells.

• At high end, e.g., 2 million cells, slow down occurs because of bus-bandwidth limitations.

• Two processes on Dual-CPU machine has the bus-bandwidth of single process, while same two processes on two separate computers are faster.

• Actual Tests To-Be-Done!
Conclusion

- MAGIC3D is now parallel.

- Possibility of Parallel MAGIC2D, with SLAC collaboration.

- All MAGIC3D executables are parallel capable - no special versions.

- Present state of progress has the four major components, Dynamic Fields, Particles, Forces, and Currents, completed so that the code can be immediately useful, even though there’s still a lot more work to be done.