

Overview of MPI

Overview of ISP's Handling of MPI

About 30 minutes - by Mike

The Continued and Foreseen Ubiquity of MPI

**IBM Blue Gene
(Picture Courtesy IBM)**



**LANL's Petascale machine
"Roadrunner"
(AMD Opteron CPUs and
IBM PowerX Cell)**



- The choice for large-scale parallel simulations (earthquake, weather..)
- Runs "everywhere" - esp on expensive state-of-the-art supercomputers
- Very mature codes exist in MPI - cannot easily be re-implemented
- Performs critical simulations in science and engineering
- **ISP supports dynamic verification of MPI C applications**

Overview of Message Passing Interface (MPI) API

- One of the Stunning Standardization Successes
- Lingua franca of Parallel Computing
- Runs on parallel machines of a WIDE range of sizes
- Standard is published at www.mpi-forum.org
- MPI 2.0 includes over 300 functions

Overview of ISP's Handling of MPI Calls

- MPI 2.0 includes over 300 functions
- ISP works by Hijacking the MPI Calls
- ISP currently handles over 60 Calls
 - Includes very popular ones
 - ISP has FULLY handled MANY applications (next slide)
 - We are continually extending the range
 - Calls not handled can often be directly issued (no hijack)
- Creating a tool such as ISP is a non-trivial venture
 - Ongoing research extends ISP's range

ISP's Score-Card of handling MPI / C Applications

ISP's Score-Card of handling MPI / C Applications

- Game of Life

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock
- ParMETIS - Hypergraph Partitioner

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock
- ParMETIS - Hypergraph Partitioner
 - Initial run of days reduced now to seconds on a laptop

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock
- ParMETIS - Hypergraph Partitioner
 - Initial run of days reduced now to seconds on a laptop
- MPI-BLAST - Genome sequencer using BLAST

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock
- ParMETIS - Hypergraph Partitioner
 - Initial run of days reduced now to seconds on a laptop
- MPI-BLAST - Genome sequencer using BLAST
 - Runs to completion on small instances

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock
- ParMETIS - Hypergraph Partitioner
 - Initial run of days reduced now to seconds on a laptop
- MPI-BLAST - Genome sequencer using BLAST
 - Runs to completion on small instances
- A few MPI Spec Benchmarks

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock
- ParMETIS - Hypergraph Partitioner
 - Initial run of days reduced now to seconds on a laptop
- MPI-BLAST - Genome sequencer using BLAST
 - Runs to completion on small instances
- A few MPI Spec Benchmarks
 - Some benchmarks exhibit interleaving explosion; others OK

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock
- ParMETIS - Hypergraph Partitioner
 - Initial run of days reduced now to seconds on a laptop
- MPI-BLAST - Genome sequencer using BLAST
 - Runs to completion on small instances
- A few MPI Spec Benchmarks
 - Some benchmarks exhibit interleaving explosion; others OK
- ADLB

ISP's Score-Card of handling MPI / C Applications

- Game of Life
 - EuroPVM / MPI 2007 versions of Gropp and Lusk done in seconds
- MADRE - Siegels' Mem Aware Data Redistrib. Engine
 - Found previously documented deadlock
- ParMETIS - Hypergraph Partitioner
 - Initial run of days reduced now to seconds on a laptop
- MPI-BLAST - Genome sequencer using BLAST
 - Runs to completion on small instances
- A few MPI Spec Benchmarks
 - Some benchmarks exhibit interleaving explosion; others OK
- ADLB
 - Initial experiments have been successful

Crash course on MPI

If you don't know MPI, you'll like our brevity

If you know MPI, then too you'll like the brevity



We will introduce four MPI functions

Namely **S**, **R**, **W**, and **B**

(out of the > 300)

Non-blocking send, MPI_Isend

- MPI_Isend(destination, msg_buf, req_struct)
- This is a non-blocking call
- Start copying out msg_buf
- Starts a memory-to-memory copy from issuing process to destination process
 - May benefit from runtime buffering
- MPI_Wait(req_struct) awaits completion
 - When Wait unblocks, sending proc can reuse msg_buf

We often abbreviate MPI_Isend
as Isend or simply **S**

We often abbreviate MPI_Wait
as Wait or simply **W**

Non-blocking receive, MPI_Irecv

- `MPI_Irecv(source, msg_buf, req_struct, ..)`
- This is a non-blocking receive call
- Starts a memory-to-memory copy from source process
- Source can be specified as ANY-SRC or 'wildcard' or '*'
- `MPI_Wait(req_struct)` awaits completion
 - When Wait unblocks, `msg_buf` is ready for consumption
- Source can be
 - "wildcard" or * or ANY_SOURCE
 - Receive from any eligible (matching) sender

We abbreviate `MPI_Irecv`
as `Irecv`, or simply **R**

Illustration of S and R in action
in 'lucky.c' and 'unlucky.c'

Example MPI program 'lucky.c'

Process P0

R(from:*, r1) ;

R(from:2, r2);

S(to:2, r3);

R(from:*, r4);

All the Ws...

Process P1

Sleep(3);

S(to:0, r1);

All the Ws...

Process P2

//Sleep(3);

S(to:0, r1);

R(from:0, r2);

S(to:0, r3);

All the Ws...

Example MPI program 'lucky.c'

Process P0

R(from:*, r1) ;

R(from:2, r2);

S(to:2, r3);

R(from:*, r4);

All the Ws...

Process P1

Sleep(3);

S(to:0, r1);

All the Ws...

Process P2

//Sleep(3);

S(to:0, r1);

R(from:0, r2);

S(to:0, r3);

All the Ws...



Example MPI program 'lucky.c' (lucky for tester)

Process P0

R(from:*, r1) ;

R(from:2, r2);

S(to:2, r3);

R(from:*, r4);

All the Ws...

Process P1

Sleep(3);

S(to:0, r1);

All the Ws...

Process P2

//Sleep(3);

S(to:0, r1);

R(from:0, r2);

S(to:0, r3);

All the Ws...

•-----•
deadlock



MPI program 'unlucky.c'

Process P0

R(from:*, r1);

R(from:2, r2);

S(to:2, r3);

R(from:*, r4);

All the Ws...

Process P1

// Sleep(3);

S(to:0, r1);

All the Ws...

Process P2

Sleep(3);

S(to:0, r1);

R(from:0, r2);

S(to:0, r3);

All the Ws...

'unlucky.c'

Process P0

R(from:*, r1);

R(from:2, r2);

S(to:2, r3);

R(from:*, r4);

All the Ws...

Process P1

// Sleep(3);

S(to:0, r1);

All the Ws...

No deadlock

Process P2

Sleep(3);

S(to:0, r1);

R(from:0, r2);

S(to:0, r3);

All the Ws...

More MPI Commands

- `MPI_Barrier(...)` is abbreviated as `Barrier()` or **B**
- All processes must invoke `Barrier` before any process can return from `Barrier` call
- Useful high-performance global sync. operation
- Sometimes used ‘for fear of the unforeseen’
 - ISP’s algorithm for removing the Functionally Irrelevant Bs, or F.I.B. (see EuroPVM/MPI 2008)

So you think you really understand

S, R, W, and B ?

Let us find out!

MPI Quiz involving S, R, W, and B

Will this single-process example called “Auto-send” deadlock ?

P0 : R(from:0, h1); B; S(to:0, h2); W(h1); W(h2);

Types of bugs detected by ISP

- Deadlocks
 - Quite common
 - Due to mismatched sends / receives
 - Unequal source / destination field
 - Unequal tags
 - Different communicators
 - Mismatched collectives
 - Flags many that are ‘glibly handled’ by real MPI runtimes
 - » Example : MPI_Allreduce : root must participate too
 - Or else, we must deadlock – yet MPI libraries often don’t
 - Yet easily missed in the exploding number of schedules!
 - ISP GUI allows users to identify onset of deadlock in replay trace
 - ISP finds deadlocks by
 - Generating only the relevant interleavings
 - Ensuring that ALL the above sources will be detected!

Types of bugs detected by ISP

- MPI Object Leaks
 - Quite common
 - Due to forgotten deallocations of one or more of these :
 - Communicators
 - Request Structures
 - Types
 - ...
 - Causes ‘slow death’ of important programs!
 - **Nightmarish to debug !**
 - Yet easily missed in the exploding number of schedules!
 - ISP GUI takes you to the allocation that was leaked
 - ISP finds leaks
 - Generating only the relevant interleavings
 - Instrumenting uses

Types of bugs detected by ISP

- Assertion Violations
 - Programmers **MUST** use C ‘assert’ statements
 - ISP GUI takes you to failing assert
 - **Guaranteed checking**
 - Generating only the relevant interleavings
 - Checks across ALL these - won’t miss assert checking due to “chancy” speed-dependent scheduling

Types of bugs detected by ISP

- **Default Safety Properties**
 - Many MPI usage-checking rules are yet to be built
 - Will make ISP all the more powerful

Summary of ISP

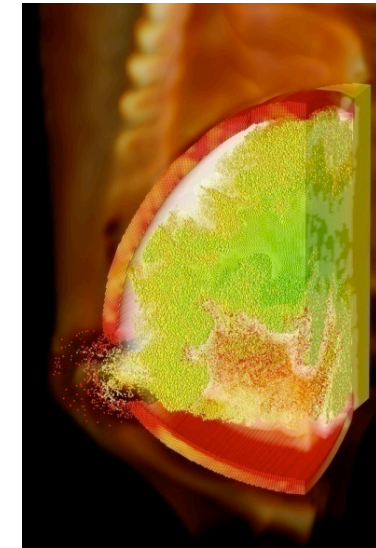
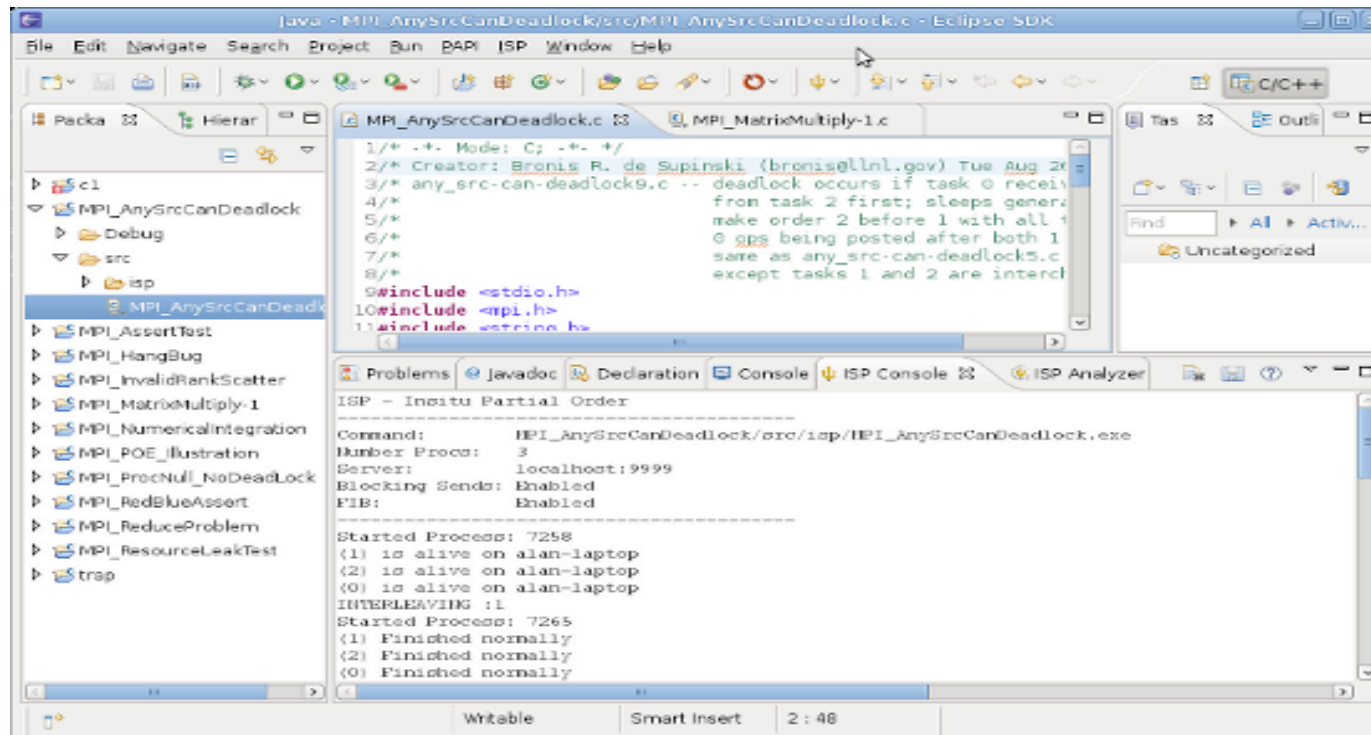
- We have built the only push-button dynamic analysis tool for MPI / C programs called ISP
 - Work on MPI / Fortran in progress
 - Runs on MAC OS/X, Windows, Linux
 - Tested against five state-of-the-art MPI libraries
 - MPICH2, OpenMPI, MSMPI, MVAPICH, IBM MPI (in progress)
 - Visual-Studio and Eclipse Parallel Tools Platform integration
 - 100s of large case studies : tool + these + LiveDVD ISO available !!
 - Guarantees to find assertion violations, deadlocks, MPI leaks (for a given test harness, all RELEVANT interleavings are replayed)
 - Efficiency is decent (getting better)
 - 15K LOC Parmetis Hypergraph Partitioner analyzed for deadlocks, resource leaks, assertion violations for a given test harness in < 5 seconds for 2 MPI processes on a laptop
 - Contribution to the Eclipse Consortium underway (lawyers looking now)
- ISP can dynamically execute and reveal the space of all standard-compliant executions of MPI even when running on **an arbitrary platform**
 - **ISP's internal scheduling decisions are taken in a fairly general way**

Features of ISP, dynamic verifier for MPI apps



(BlueGene/L - Image courtesy of IBM / LLNL)

- Verifies MPI User Applications, generating only the *Relevant Process Interleavings*
- Detects all Deadlocks, Assert Violations, MPI object leaks, and Default Safety Properties
- Works by Instrumenting MPI Calls
Computing Relevant Interleavings, Replaying



(Image courtesy of Steve Parker, U of Utah)

End of B