Formal Analysis for Debugging and Performance Optimization of MPI

Ganesh Gopalakrishnan

Robert M. Kirby

School of Computing

University of Utah

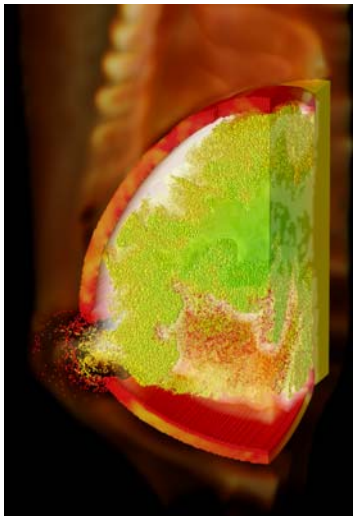
NSF CNS 0509379

THE UNIVERSITY OF UTAH

On programming machines that turn electricity into numbers...

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

→



(Image courtesy of Steve Parker, CSAFE, Utah)

Our focus: Reactive bugs in HPC Programming

An inconvenient truth: ***Bugs → wasted energy, bad numbers***

Correctness of HPC Software

- **A very opportune time to be addressing!**
  - **Multicores, Threads, Transaction memories, MPI, OpenMP, UPC, …**
  - **Msg Passing, Libraries, Memory Models, Combined Threads and Processes …**
- **Can't apply existing FV solutions directly**
  - **Formal Analysis must Consider Library Implementation**
    - » **Message Passing and Threading issues**
  - **Correctness needs to PORT and SCALE**
  - **Need domain-specific adaptations of existing FV Research**

"Correctness"

MPI Bugs do occur !

1. Incorrect Understanding of MPI

   e.g., collectives have "barrier semantics"

2. Reuse Send Buffers Prematurely
3. Deadlocks After Porting to New Platform
4. Forgotten Deallocation of Communicators

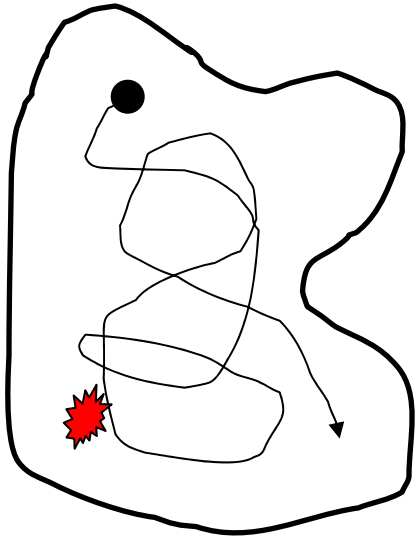
   shows up when scaled

5. MPI one-sided operations are very tricky

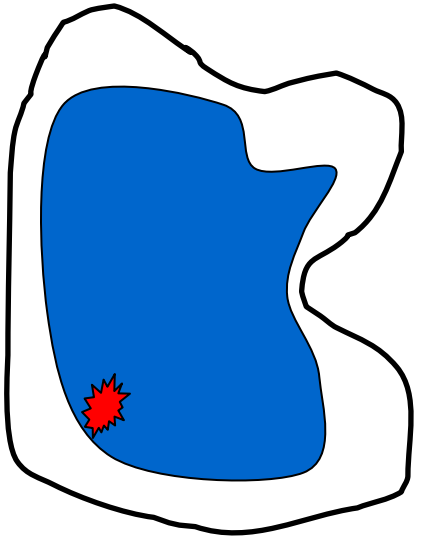
   breaks out of the msg passing comfort-zone

On the use of Model Checking in HPC Software Verification…

"Ad-hoc Testing"



"Model Checking"



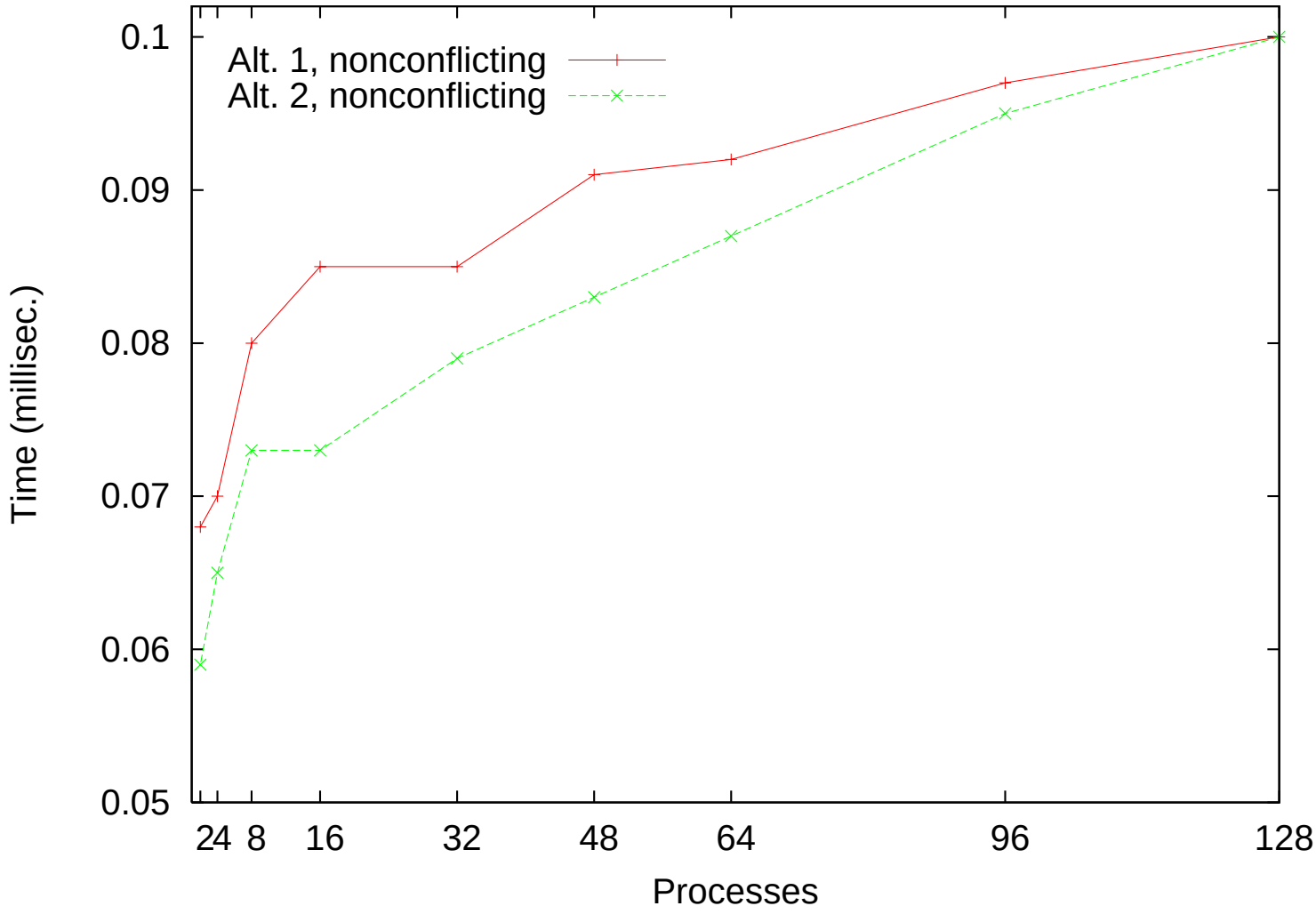
We Employ Static Analysis, Instrumentation and Direct Execution along with Model Checking…

We Emphasize Formal Specs of Libraries

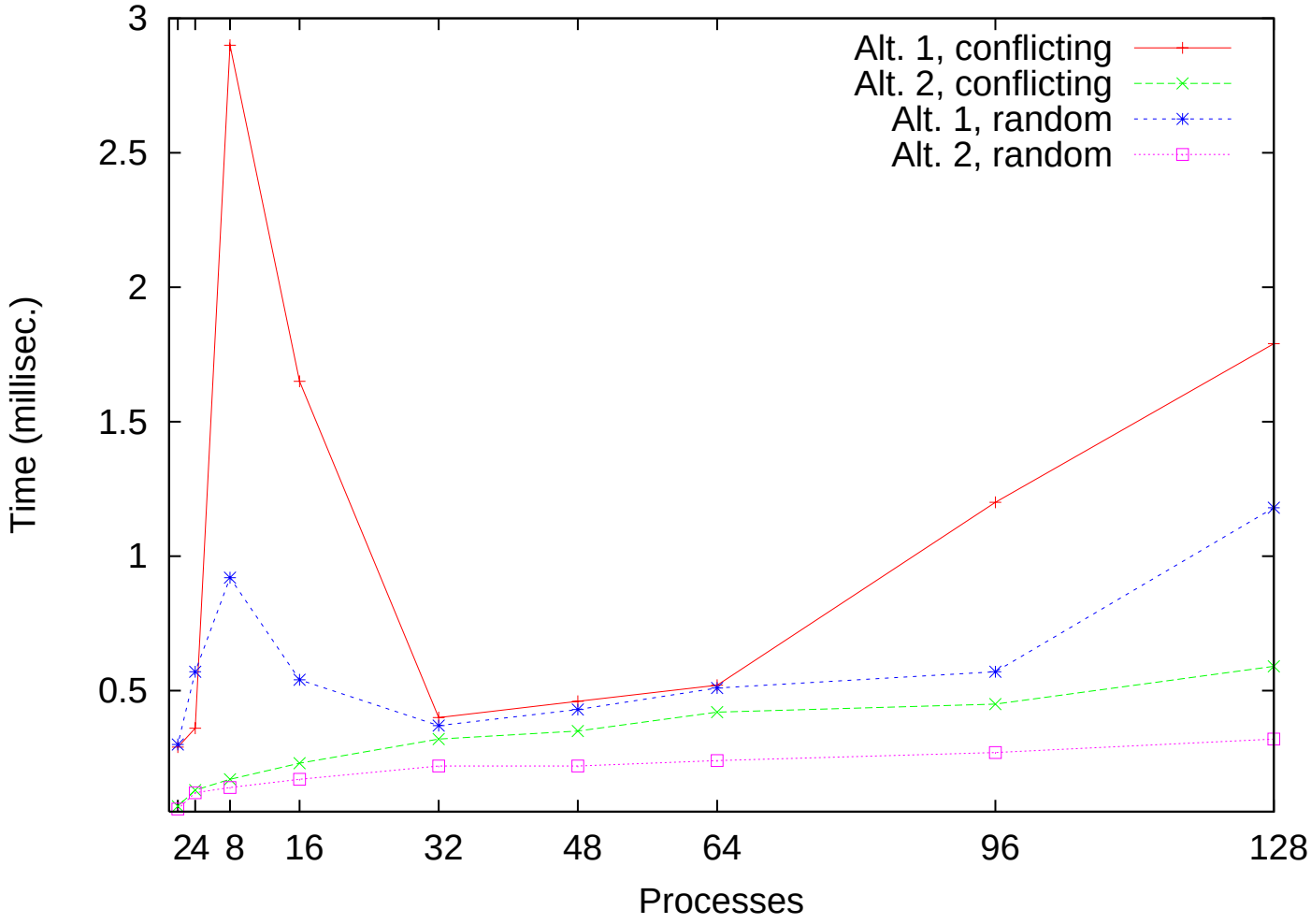
Project 1:

- **Direct Use of Existing Model Checking Technology**
  - **SPIN**
  - **Helped take first steps; now abandoned**
  - **But Gave Us our First Collaborative EuroPVM / MPI Paper (2006; One of Three Outstanding Papers)**
    with Rajeev Thakur and Bill Gropp
  - **Our Fixed Algo Performs Better, as well (see next)!**
    - » **To appear in Parallel and Distributed Computing**
  - **(Most Recent News): The SAME bug has been caught using our MPI In-Situ Model Checker that incorporates Dyn. Partial Order Reduction**

Measurement under Low Contention



Measurement under High Contention

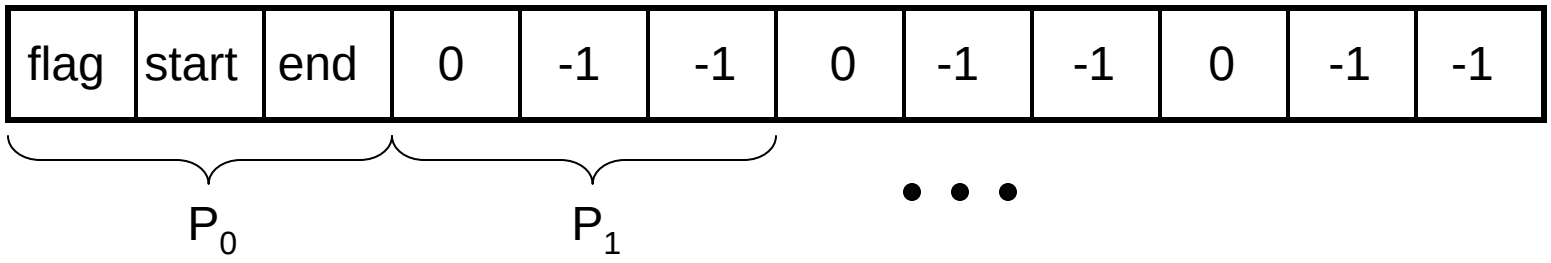


Developed NEW and FASTER Byte-Range Locking Protocol after Fixing Bug in Earlier Protocol …

```
lock_acquire (start, end) {

                Stage 1
1    val[0] = 1; /* flag */ val[1] = start; val[2] = end;
2    while(1) {
3      lock_win
4      place val in win
5      get values of other processes from win
6          unlock_win
7       for all i, if (Pi conflicts with my range)
8          conflict = 1;
```

```
                Stage 2
9      if(conflict) {
10       val[0] = 0
11           lock_win
12           place val in win
13           unlock_win
14           MPI_Recv(ANY_SOURCE)
15           }
16     else{
17           /* lock is acquired */
18         break;
19         }
20   }//end while
```



Project 2:

- **Wrote MPI + Threads Parallel and Distributed Model Checker**
  - **Helped study domain of interest closely**
  - **Software Released : Eddy-Murphi**
  - **Details already reported during last PI meeting…**

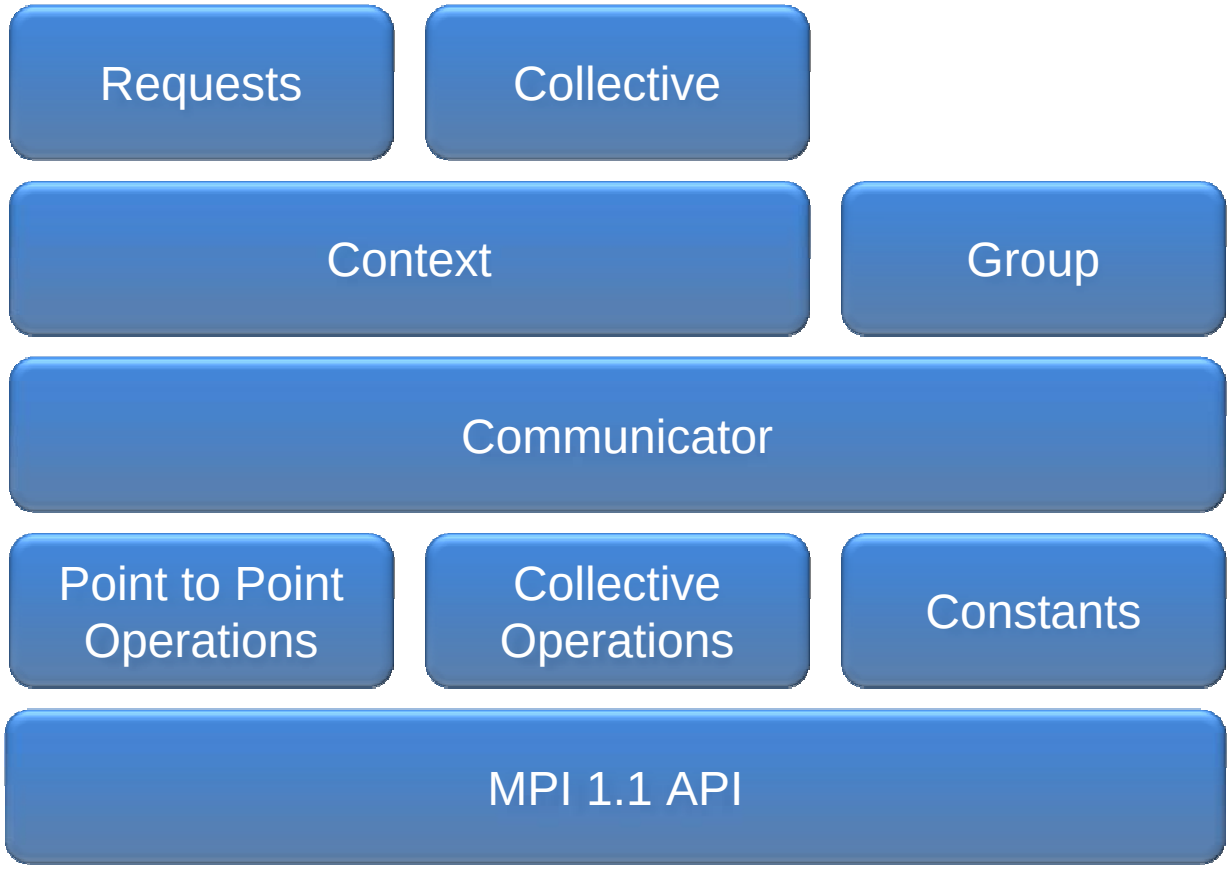
Project 3:

- **Modeled MPI 1.1 Primitives in TLA+**
  - **Helped understand MPI**
  - **Helped find corner cases (confirmed by ANL experts)**
  - **Built direct model-checker based on MPI semantics**
  - **Integrated with C-MPI → TLA+ model extractor**
  - **Integrated with VisualStudio Debugger**

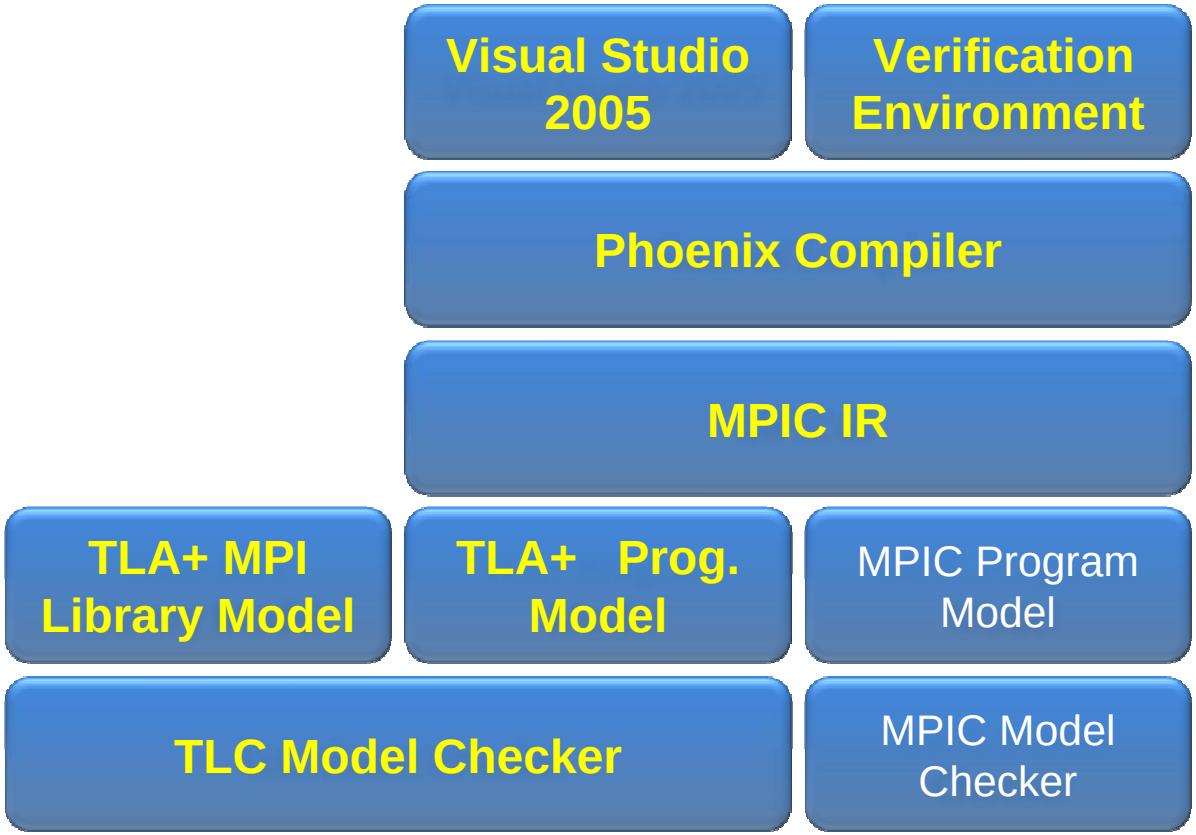
Project 3: Given This Program…

```
/* Add-up integrals calculated by each process */
    if (my_rank == 0) {
        total = integral;
        for (source = 0; source < p; source++) {
            MPI_Recv(&integral, 1, MPI_FLOAT,
                source, tag,
                MPI_COMM_WORLD, &status);
            total = total + integral;
        }
    } else {
        MPI_Send(&integral, 1, MPI_FLOAT, dest,
            tag, MPI_COMM_WORLD);
    }
```

…Our TLA+ MPI Model …



…and Integration into Phoenix/VisualStudio….

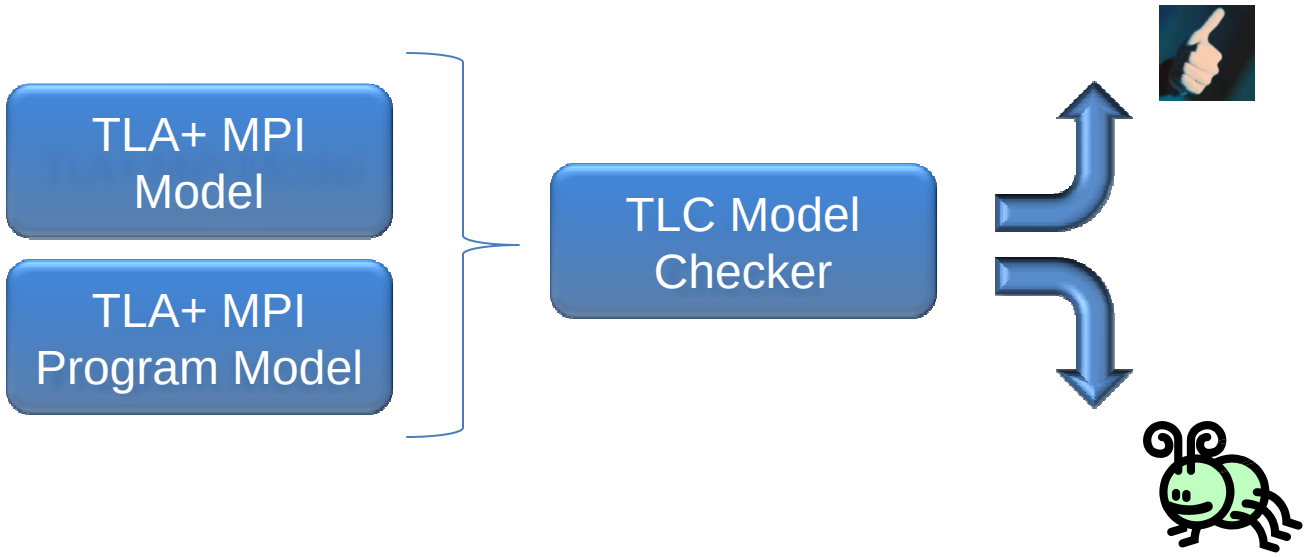


...Using The Formal Semantics of MPI At THIS Level... (MPI_Issend & MPI_Irecv Shown)

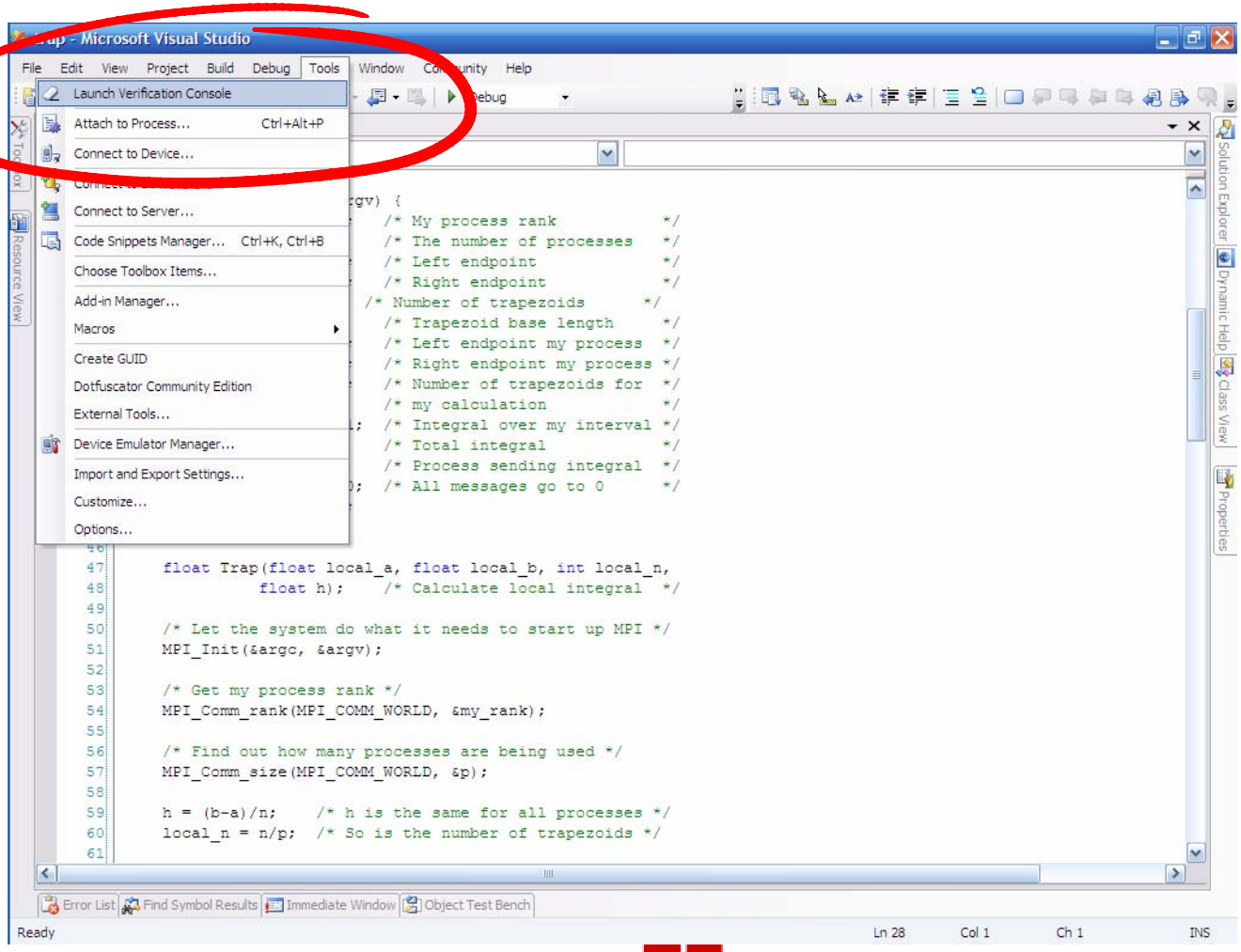
$$\frac{\begin{array}{l}\Sigma(c,p) \wedge \\ \quad p(i) = (l,g) \wedge \\ \quad proc(l(vars(\texttt{pc}))) \;=\; '\texttt{x} = \texttt{issend dest addr}'\end{array}}{\begin{array}{ll}\Sigma(c, p[i \mapsto (& l[vars(\texttt{pc}) \mapsto next(l(vars(\texttt{pc}))), \\ & vars(\texttt{x}) \mapsto (|Dom(g)|+1)], \\ & g[(|Dom(g)|+1) \mapsto (i, E[\texttt{dest}, p_i], E[\texttt{addr}, p_i], send, false)])])\end{array}}$$

$$\frac{\begin{array}{l}\Sigma(c,p) \wedge \\ \quad p(i) = (l,g) \wedge \\ \quad proc(l(vars(\texttt{pc}))) \;=\; '\texttt{x} = \texttt{irecv src addr}'\end{array}}{\begin{array}{ll}\Sigma(c, p[i \mapsto (& l[vars(\texttt{pc}) \mapsto next(l(vars(\texttt{pc}))), \\ & vars(\texttt{x}) \mapsto (|Dom(g)|+1)], \\ & g[(|Dom(g)|+1) \mapsto (E[\texttt{src}, p_i], i, E[\texttt{addr}, p_i], recv, false)])])\end{array}}$$

…helps enumerate states, finds bugs....



…And Steps VisualStudio Debugger to Show the Error Traces!



Project 4:

- **Wrote a Customized MPI Model Checker called MPIC**
  - **Our MPI Formal Semantics Helped Write MPIC Reliably**
  - **MPI Formal Semantics Gave us the Independence Theorems**
  - **This Helped Cut Down Interleavings**
  - **Seamlessly Integrated Into Same Framework Through**

    Common Intermediate Representation (MPIC IR)

Project 4: Again, Given This Program…

```
/* Add-up integrals calculated by each process */
    if (my_rank == 0) {
        total = integral;
        for (source = 0; source < p; source++) {
            MPI_Recv(&integral, 1, MPI_FLOAT,
                source, tag,
                MPI_COMM_WORLD, &status);
            total = total + integral;
        }
    } else {
        MPI_Send(&integral, 1, MPI_FLOAT, dest,
            tag, MPI_COMM_WORLD);
    }
```

…Our MPI Formal Semantics…

$$\frac{\begin{array}{l}\Sigma(c,p) \wedge \\ \quad p(i) = (l,g) \wedge \\ \quad proc(l(vars(\texttt{pc}))) \;=\; '\texttt{x = issend dest addr}'\end{array}}{\begin{array}{ll}\Sigma(c, p[i \mapsto (& l[vars(\texttt{pc}) \mapsto next(l(vars(\texttt{pc}))), \\ & vars(\texttt{x}) \mapsto (|Dom(g)|+1)], \\ & g[(|Dom(g)|+1) \mapsto (i, E[\texttt{dest}, p_i], E[\texttt{addr}, p_i], send, false)])])\end{array}}$$

$$\frac{\begin{array}{l}\Sigma(c,p) \wedge \\ \quad p(i) = (l,g) \wedge \\ \quad proc(l(vars(\texttt{pc}))) \;=\; '\texttt{x = irecv src addr}'\end{array}}{\begin{array}{ll}\Sigma(c, p[i \mapsto (& l[vars(\texttt{pc}) \mapsto next(l(vars(\texttt{pc}))), \\ & vars(\texttt{x}) \mapsto (|Dom(g)|+1)], \\ & g[(|Dom(g)|+1) \mapsto (E[\texttt{src}, p_i], i, E[\texttt{addr}, p_i], recv, false)])])\end{array}}$$

…Gives Us Independence Theorems…

1. **Local actions (Assignment, Goto, Alloc, Assert) are independent of all transitions of other processes.**

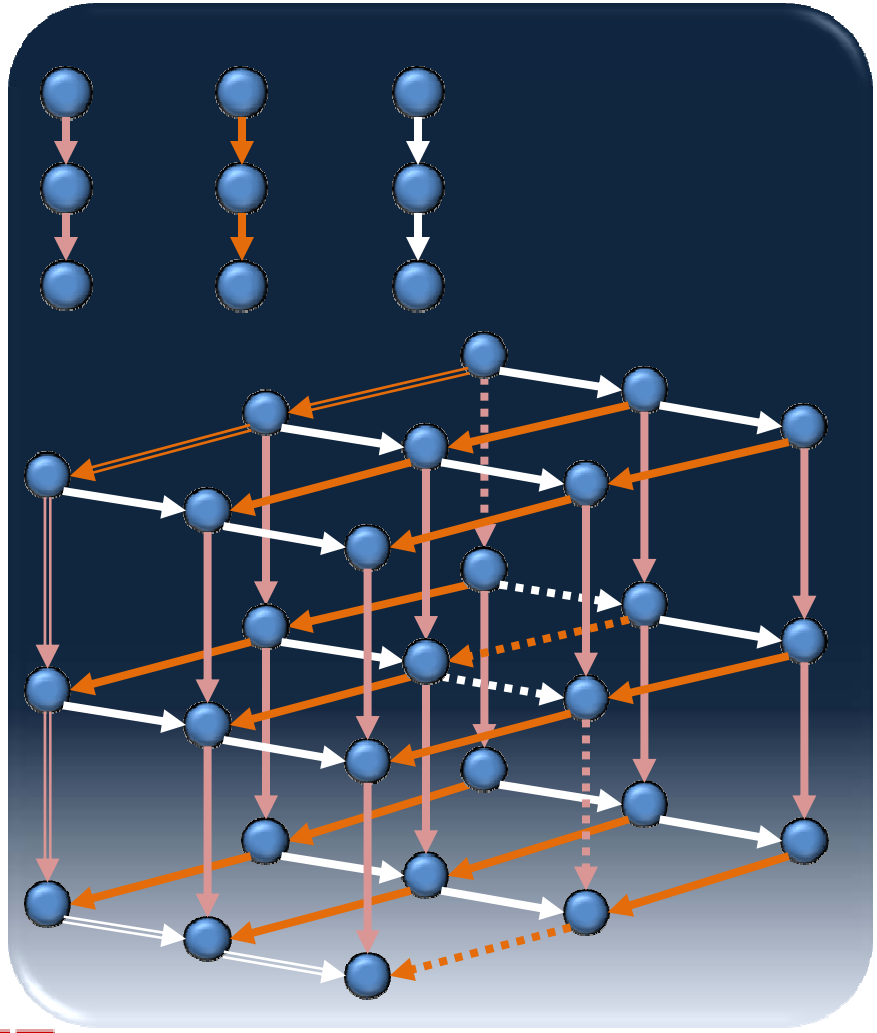
2. **Barrier actions (Barrier_init, Barrier_wait) are independent of all transitions of other processes.**

3. **Issend and Irecv are independent of all transitions of other processes except Wait and Test.**

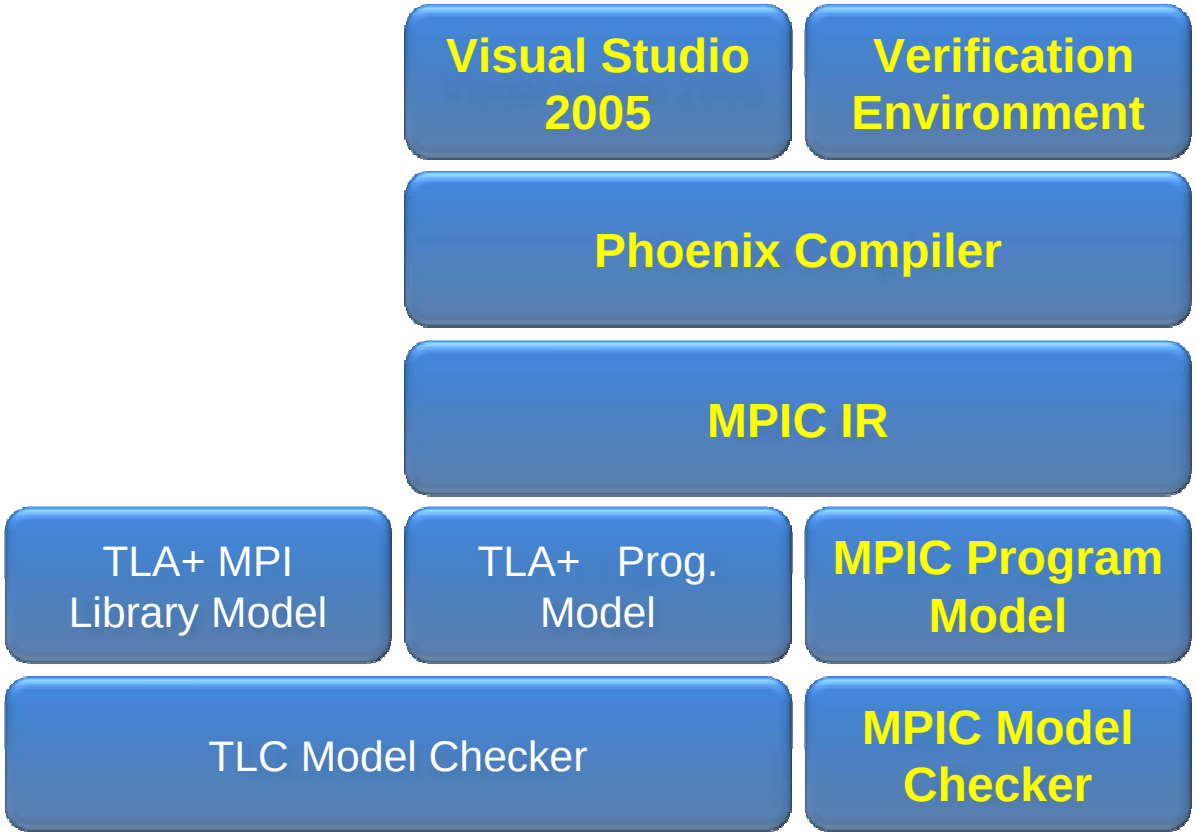
4. **Wait and Test are independent of all transitions of other processes except Issend and Irecv.**

…To Mitigate State Explosion Using Dynamic Partial Order Reduction…

- DPOR is GREAT for Handling MPI Wildcards with Associated Waits and Tests
- With 3 processes, the size of an interleaved state space is p^s=27
- Partial-order reduction explores representative sequences from each equivalence class
- Delays the execution of independent transitions



…and our Customized Model Checker MPIC, Integrated Into Framework….

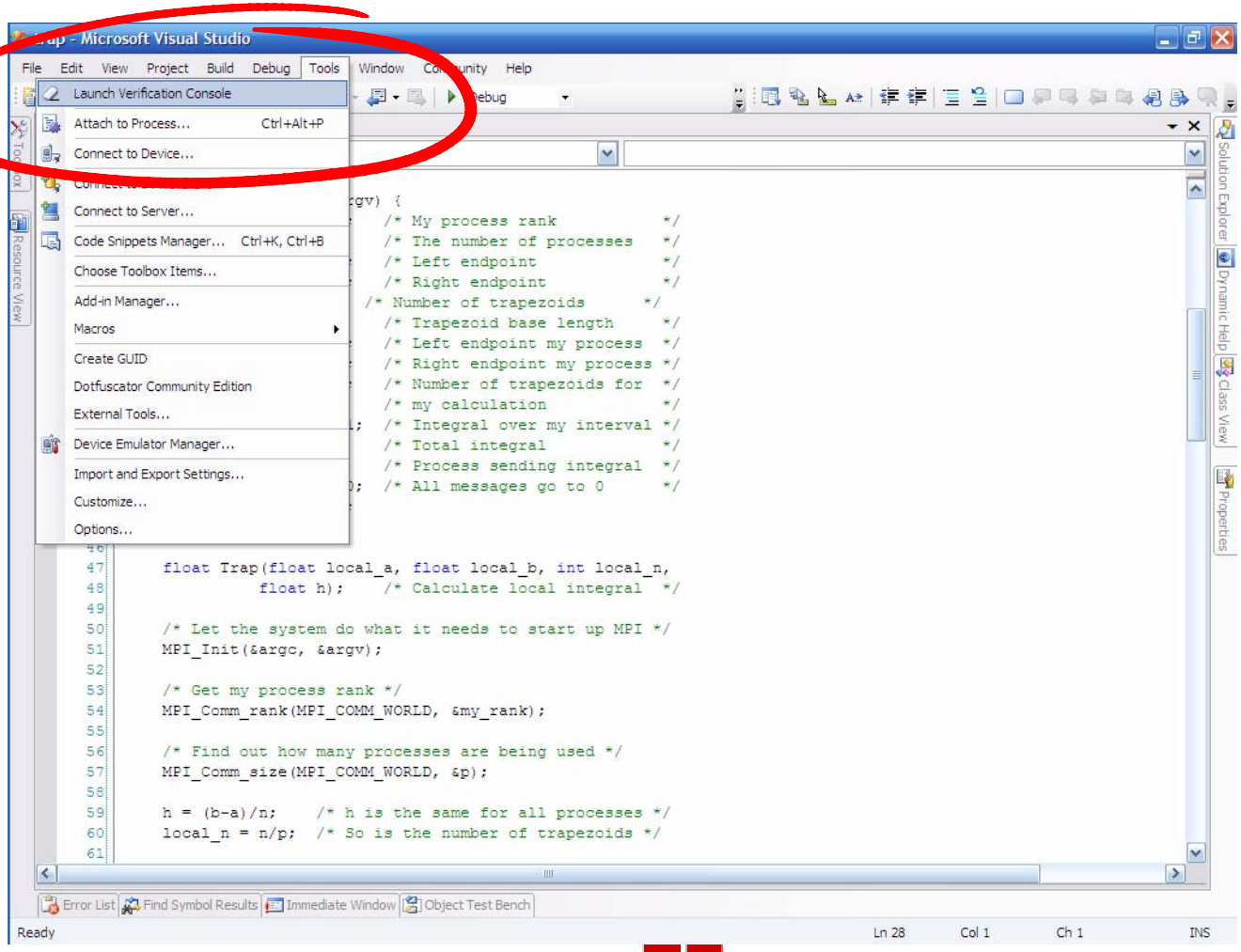


…runs much faster, finds bugs…

- **MPIC with and without DPOR**

Model	States	Time	Trans	Memory
2D Diffusion (4 processes)	**>20,000,000**	**>15 Minutes**	**>75,000,000**	**~500MB**
2D Diffusion (4 processes) (using dpor)	**7769**	**< 1 second**	**7768**	**~10MB**

…And Steps VisualStudio Debugger to Show the Error Traces!



Project 5:

- **In-Situ Model Checking of PThread Programs**
- **Avoids Model Extraction**
  - **(VERY difficult and huge up-keep chore)**
  - **Models and Verifies the "real thing"**
- **First Implementation of Dynamic Partial Order Reduction for PThreads Programs (as far as we know)**
  - **Tool is Called *Inspect***
  - **Implemented in In-House C/C++ Front-end Derived by Modifying GCC**

(We also have a DPOR-enabled In-situ Model Checker for MPI under construction)

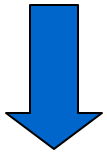
Dynamic Partial Order Reduction works really well for direct model-checking of REAL CODE…

- **Static POR relies on static analysis**
  - **Imprecise Information About Run-time**
  - **Pointers → Coarse Info → Limited POR → State Explosion**
- **Dynamic POR**
  - **More run-time information**
  - **Independence can be dynamically determined**

Given a PThreads Program, We Instrument it ……

Original code

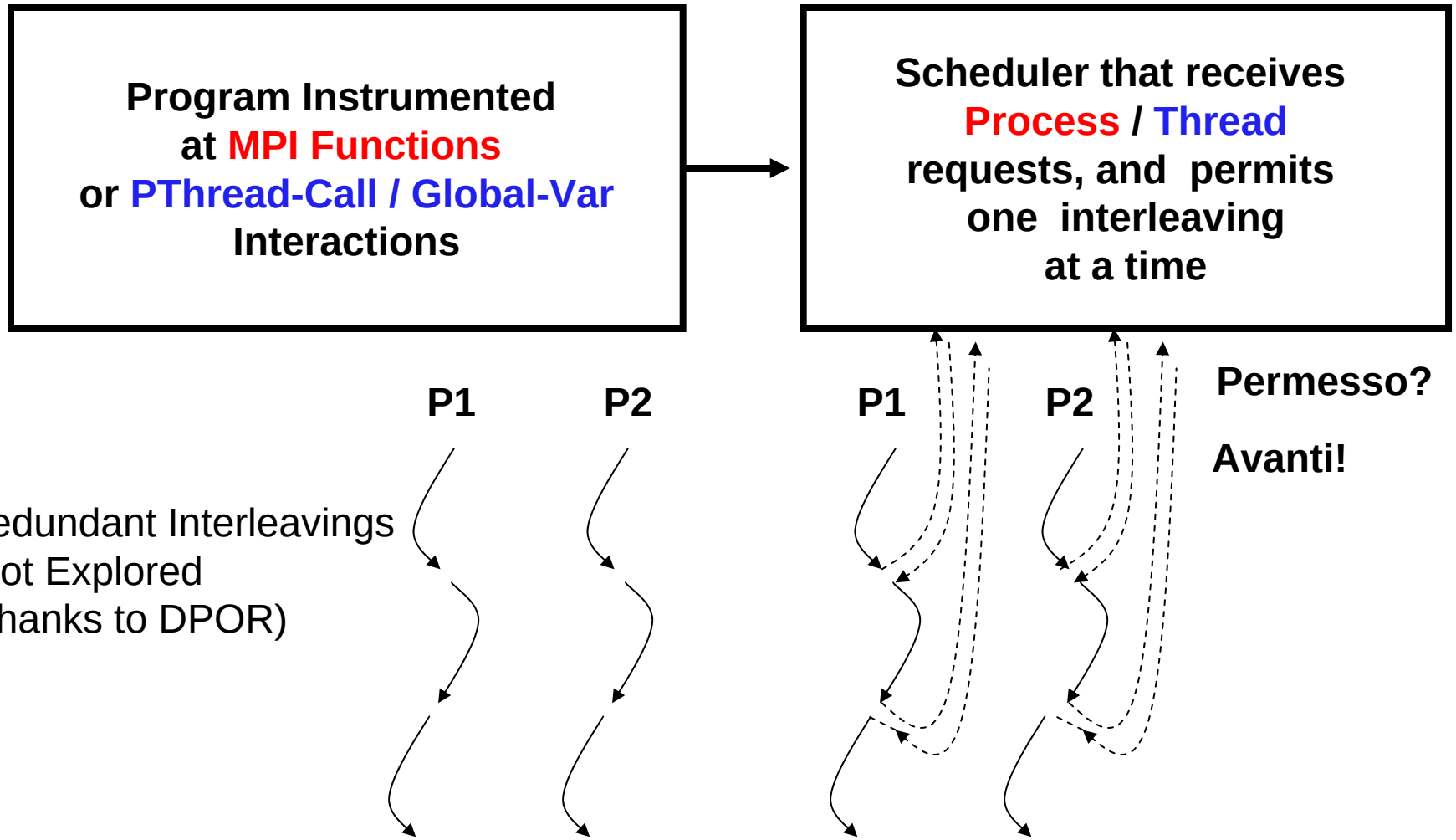
```
pthread_mutex_lock( &a ); data++;
pthread_mutex_unlock(&a);
```



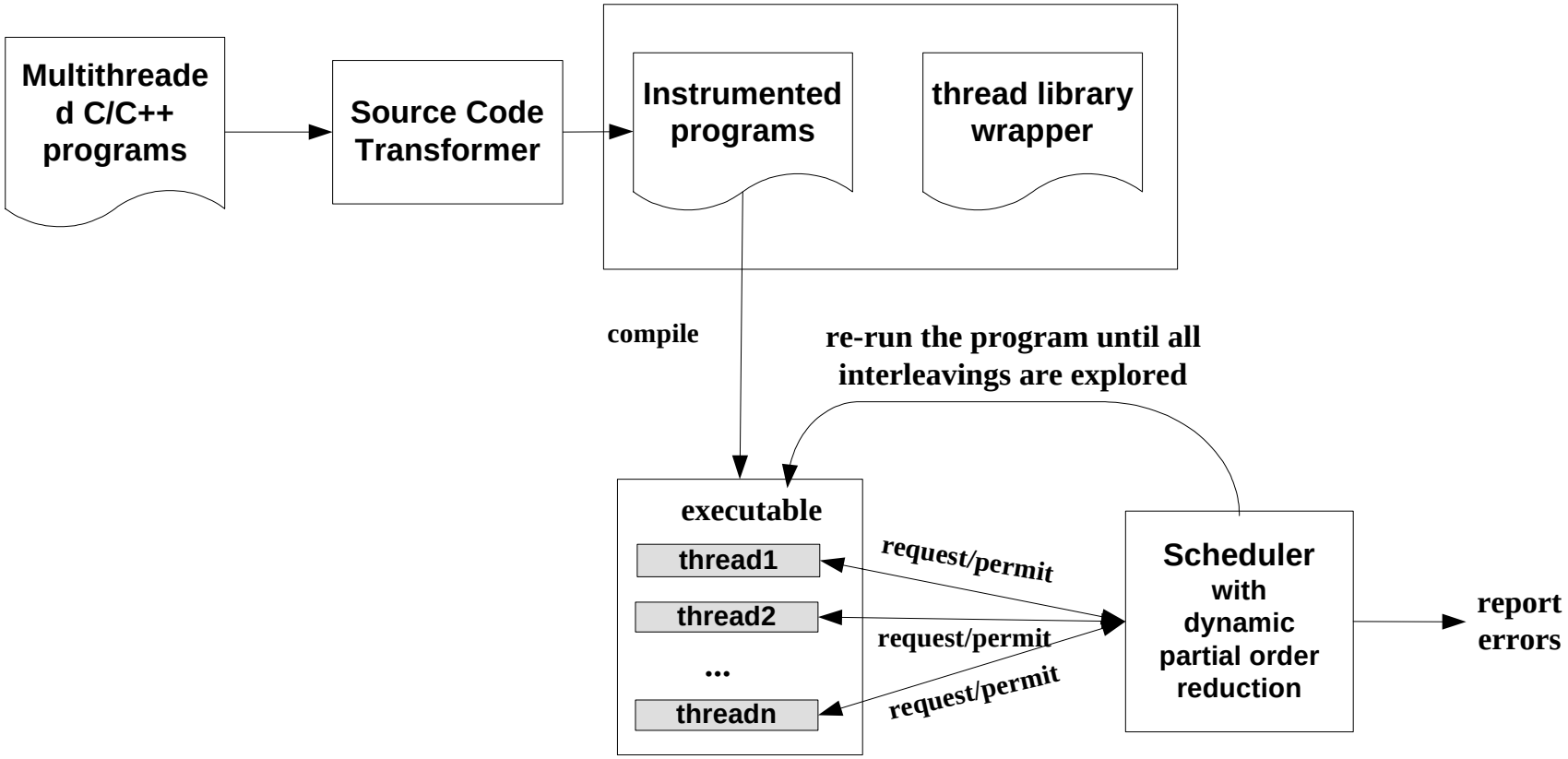
Instrumented code

```
inspect_mutex_lock( &a );
inspect_obj_write( (void*)&data)
data++;
inspect_mutex_unlock(&a);
```

…So that a DPOR-Aware Scheduler can Schedule Relevant Interleavings…



…Using Inspect's workflow…



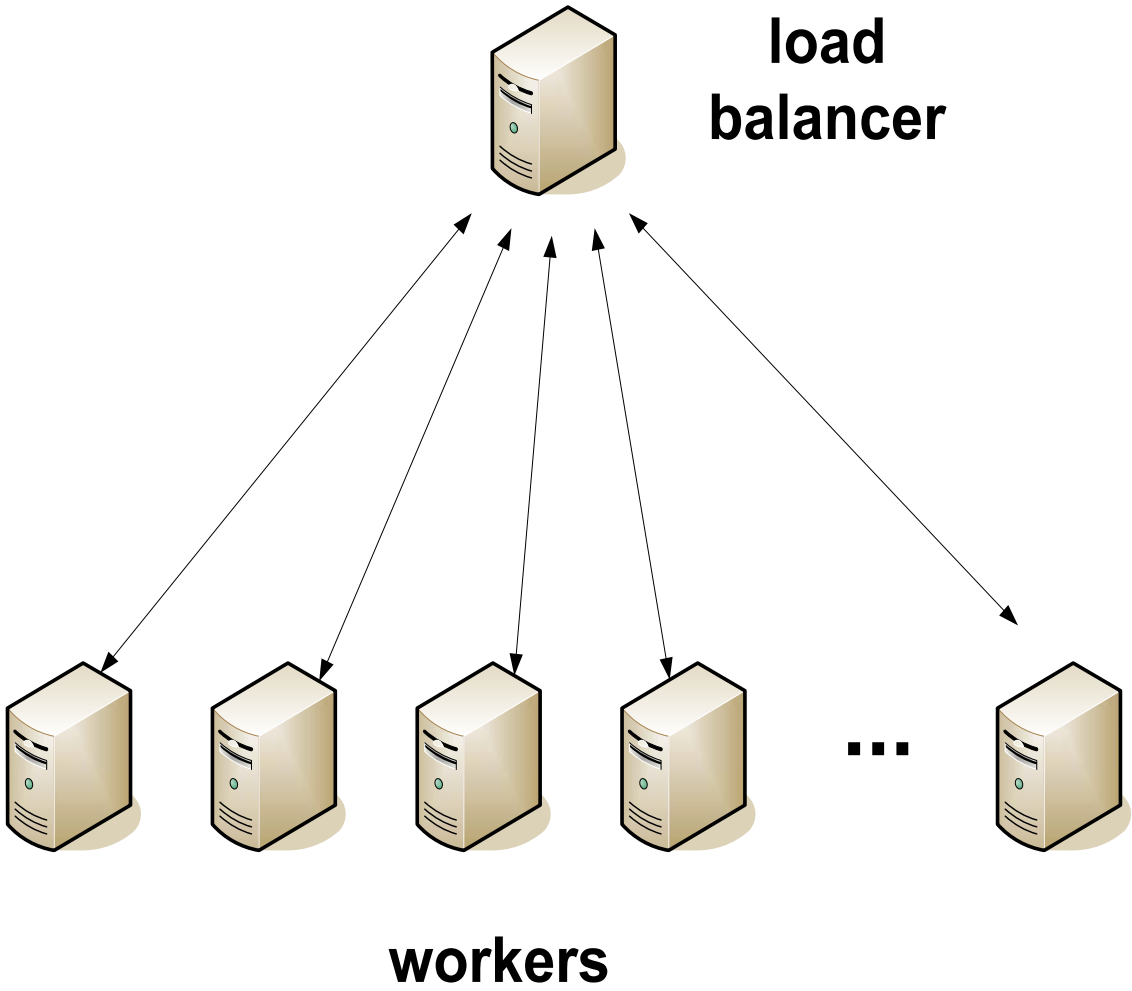
…Which Has These Plusses ….

- **Avoids Model Extraction**
  - **(VERY difficult and huge up-keep chore)**
  - **Models and Verifies the "real thing"**
- **First Implementation of Dynamic Partial Order Reduction for PThreads Programs (as far as we know)**
  - **Tool is Called Inspector**
  - **Implemented in In-House C/C++ Front-end Derived by Modifying GCC**

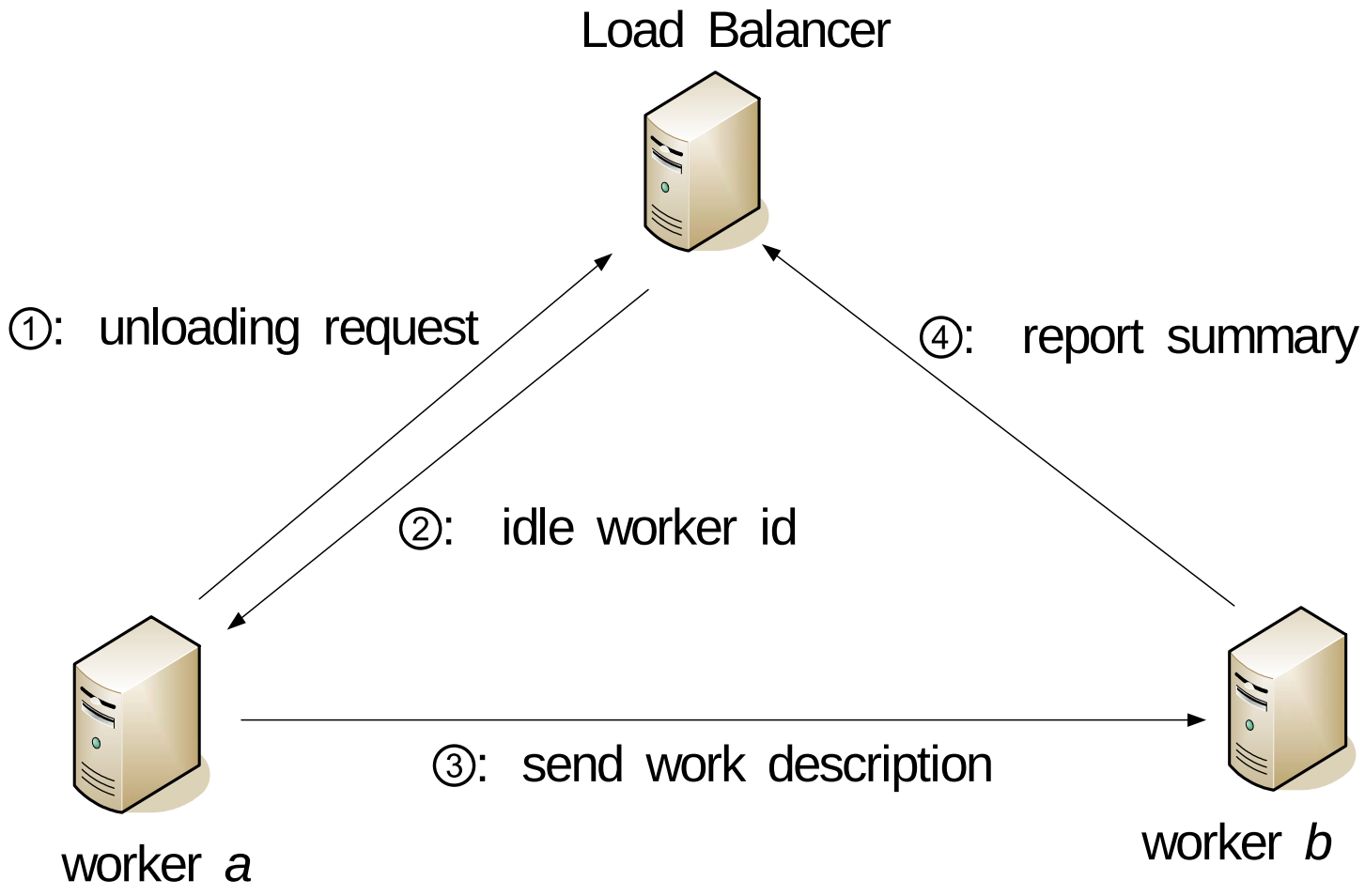
…And Gives us These Results!

Benchmark		LOC	thread	Error		
				Data race	deadlock	others
aget		**1,098**	**3**	**1**	**0**	**0**
pfscan		**1,073**	**4**	**1**	**0**	**4**
tplay		**3,074**	**2**	**0**	**0**	**0**
libcprops	**avl**	**1,432**	**1-3**	**2**	**0**	**0**
	heap	**716**	**1-3**	**0**	**0**	**0**
	hashlist	**1,953**	**1-3**	**1**	**0**	**1**
	splay-tree	**1,211**	**1-3**	**1**	**0**	**0**

Project 6: Wait! We can have a Distributed Inspect…



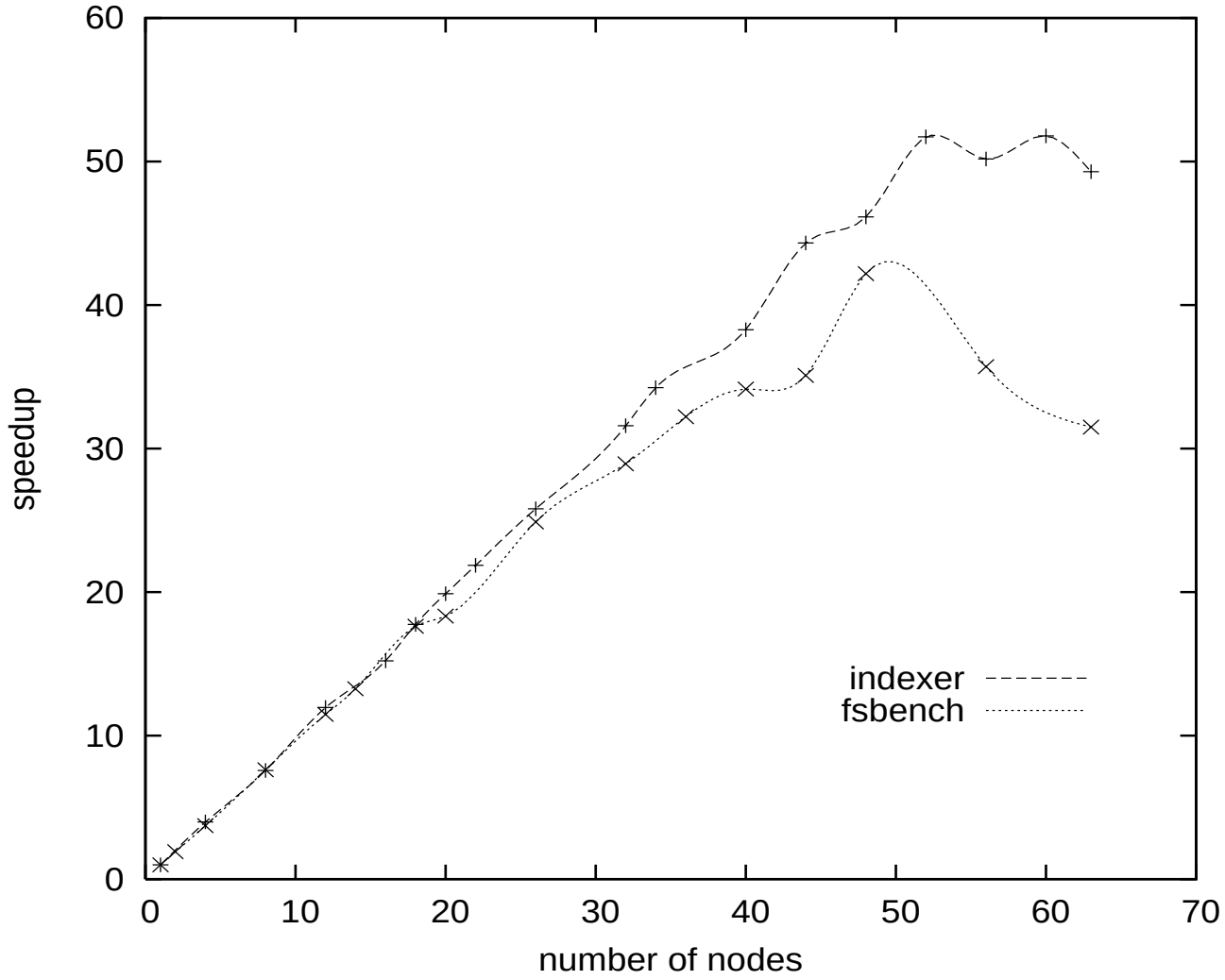
..Which has THIS Communication Flow...



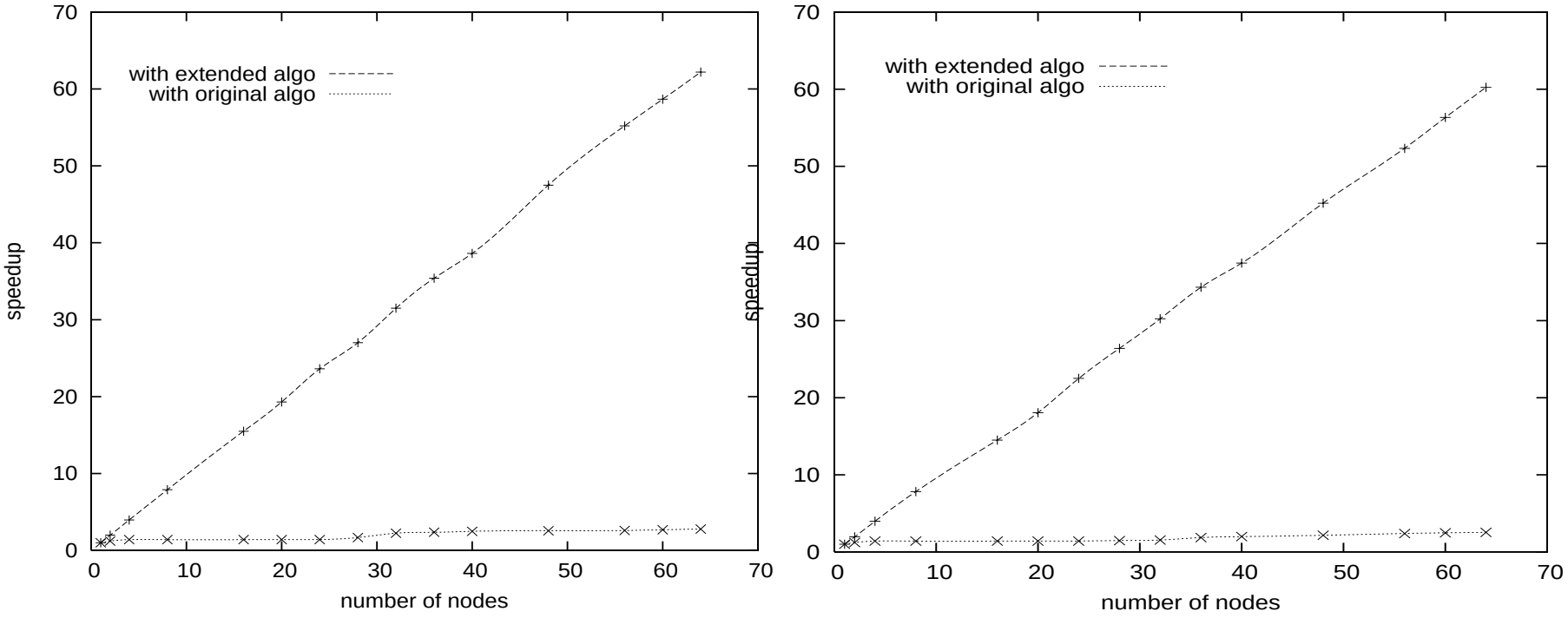
…Helping Obtain Linear Speed-up with respect to THESE Seq Inspect Runs…

benchmark	threads	runs	sequential checking **(sec)**
fsbench	**26**	**8,192**	**291.32**
indexer	**16**	**32,768**	**1188.73**
aget	**6**	**113,400**	**5662.96**
bbuf	**8**	**1,938,816**	**39710.43**

..Out of Parallelism (overheads) on two Small benchmarks…



..But Look at Speedup on **bbuf** and **aget** **(11 hours of work in 11 minutes…)**



PLANS FOR THE 3rd and 4th YEARS

- **EXTEND MPI FORMAL SEMANTICS**
- **FULL IN-SITU DPOR for MPI PROGRAMS**
- **LEARN HOW TO VERIFY MPI + THREADED CODES**
- **LEARN WHERE HPC IS HEADED AND REACT TO IT!**

Scaling MPIC

MPI Plus PThreads / OpenMP

MPI That Supports Four Thread Models

Multicore MPI

Transactions

Compilation of MPI from High-level Specs

ACKNOWLEDGEMENTS

- Robert Palmer --- Finishing PhD and Joining Intel in their new Multicore FV Group
  - Summer internships at Microsoft Research (05) and Intel (06)
- Yu Yang -- Aiming to finish PhD in 2008
  - Internships at NEC Research (06) and Cadence/ Berkeley Labs (07)
- Salman Pervez – Converted from PhD to MS
  - Aiming to finish MS in 2007 and continue at (?Purdue?)
  - Summer Internship at Argonne National Labs (06)
- Geof Sawaya – MANY Projects (e.g. MPI Optimization)
- Michael DeLisi – UG who built VisualStudio Integration
- Subodh Sharma – PhD in 2009? (Testing MPI)
- Sarvani Vakkalanka – PhD in 2009? (High Level HPC?)

PUBLICATIONS and SOFTWARE

- **Paper on Eddy-Murphi (SPIN 06) Appeared**
  - **Journal version Accepted into STTT (07)**
- **EuroPVM / MPI Paper (06)**
  - **Special Issue Parallel Computing Journal Version under Review**
- **Submission to ISSTA (07) under review**
- **Submission to SPIN (07) under review**
- **Submission Planned to**
  - **PADTAD (07)**
  - **EuroPVM / MPI (07)**
- **Organized TV (Thread Verification, Seattle, 11 / 17 / 06)**
- **Software and Benchmarks: MPI Formal Semantics; Inspect; Soon DInspect; Phoenix / Visual Studio Integration of Model Extraction and Verification; and soon In-Situ MPI Verif**

Questions ?

Answers!

1. **We plan to investigate what breaks when scaled**
   - **lesson learned from de Supinski**
2. **We plan to analyze 1-Sided within CLAM/Prema**
   - **Nikos Chrisochoides**
3. **We plan to analyze mixed MPI / Threads**
   - **Context-ID Generation Algorithm within MPI Thread-Multiple Libs**
   - **Suggested by Bill Gropp**
   - **Will involve creating environment models for**
     - **THREADS, when doing MPI In-Situ Model Checking**
     - **MPI, when doing THREADS In-Situ Model Checking**
4. **That is a very good question – let's talk!**