A Debugger for Computational Grid Applications

Robert Hood
rhoads@nasa.gov

Gabrielle Jost
gjost@nasa.gov

CSC/MRJ Technology Solutions
NASA Ames Research Center

NAS Parallel Tools Group (1)

- Parallelization support tools
  - CAPTools: Transforms serial Fortran code into MPI code with user guidance
  - CAPO: Transforms serial Fortran code into OpenMP code with user guidance
  - Charon: Library tool for data distribution and message passing on top of MPI
  - Adapt: Tool for data placement in data parallel programming models
- Current work: support of multi-level parallelization and hybrid MPI-OpenMP parallelization

NAS Parallel Tools Group (2)

- P2D2 parallel/distributed debugger
- Evaluation of various parallelization strategies:
  - Performance, type of application, type of hardware architecture, portability
- Distributed and aggregated computing:
  - Large applications running under Globus
- Jobs scheduling and resource allocation under Globus

Historical Background

- Goal in 1994: Develop a distributed debugger
  - With a user interface that scales to "many" processes
  - Portable across a large variety of machines
- Result in 1996: p2d2 (portable parallel/distributed debugger)
  - Scalable UI
  - Highly portable
  - Facilitates further research

Debugging Challenge 1998

- Need a debugger for computational grids

Rest of talk:
- Debugger architecture
- Support of heterogeneity
- Support of scalability
- Attaching to grid computations
- Quick discourse on running jobs under Globus

Debugger Dependencies

- Function of the Debugger:
  - Mapping between user view of a program at source code level onto the machine version at object level.
- Dependencies:
  - Target architecture → Breakpoint implementation
  - Operating system → Process control
  - Compiler → Symbol table information

Additional dependencies for parallel processing, e.g.:
- Thread abstraction
- Synchronization method
- Message passing format
- Process creation
Accommodating Heterogeneity

- P2P approach to heterogeneity:
  - Isolate the dependencies of the debugger from the user interface through the use of a client-server model.
- Debugger server:
  - Architecture-, OS-, and compiler-dependent code.
  - Implemented by vendor.
- User interface (UI) client:
  - Portable code

### Initial Implementation

- Use gdb from the Free Software Foundation as debugger server
  - Advantages:
    - Freely available
    - Portable
  - Disadvantages:
    - Vendor support minimal
- Replication of gdb's permits heterogeneity

### Scalability

- Main debugger operations that need to scale:
  - Process control operations
    - Setting/deleting breakpoints, continue, single step
  - State examination
    - Print, display, stack trace
- Debugging N processes:
  - Indicate on which processes control operations are performed
  - Extract state information across a set of processes
- p2p2 process navigation paradigm:
  - Process control operations to processes in control set
  - Overview of global state in process grid
  - More information about processes in focus group
  - Detailed information about focus process

### Scaling the User Interface

- Allow collective control of processes
- Provide "zooming" with 3 levels of detail for state examination.

### The Process Grid:

- Overview of all processes in the computation
- Used for "zooming in" on processes for closer examination:
  - The focus group
    - One line of text about each process in group
  - The focus process:
    - Detailed information about a single process
  - The control set:
    - Processes that receive control operations (breakpoints, continue)
      - Indicated by white frame selected by mouse click

### Brief Discourse on Globus (1)

- What are Grids?
  - Super Interests for high-performance computing
  - Worldwide collection of high-end resources:
    - Supercomputers, storage, advanced instruments, immersive environments
    - Enable the development of applications that require geographically distributed high-end resources
- What is Globus?
  - Software toolkit to facilitate the creation of Grids
  - Allows:
    - Uniform access to distributed resources
    - Information services about available resources
    - Tools for remote file management, sharing of executables and data
Heterogeneity & the UI: Customizing the Display

- Process grid view can be programmed:
  - a list of directives of the form: `<icon> if <predicate>`
  - Samples for `<predicate>`:
    - running(), eval(expr), systemMatches (string)

Heterogeneity and the UI:

Consistent Data View

- Comparing expression values across processes:
  - gdb evaluates to text
  - question: In what context should gdb do the evaluation?
- P2d2 tries to do evaluation in equivalent stack frame:

<table>
<thead>
<tr>
<th>Process 1</th>
<th>Process 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 in main</td>
<td>0 in main()</td>
</tr>
<tr>
<td>0 in m_b()</td>
<td>0 in m_b2()</td>
</tr>
<tr>
<td>0 in main()</td>
<td>0 in main()</td>
</tr>
</tbody>
</table>

In heterogeneous environment:
- function names don't match, e.g.,
  - toto vs. toto
  - convert function names to canonical form

Heterogeneity & the UI: Abstract Data View

Distributed array view

- Global Array View
- Local Array View

Status and Future Work

- Status of P2d2 debugging Globus jobs:
  - debugged a Globus job running on 3 machines
    - SGI Origin in California
    - PC/Linux in Ohio
    - Sun Sparc Workstation in Virginia
  - debugged a 128-process Globus job running on 3 Origins
  - not yet there:
    - record contact information in MG5
    - security for Globus initiated jobs
- Distribution Status:
  - plan to distribute under an "OpenSource" copyright.
- Current work:
  - relative debugging of tool-parallelized programs