A Debugger for Computational Grid Applications

Robert Hood
rhoads@nas.nasa.gov
CSCMC 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
Accomodating Heterogeneity

- P2D2 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:
    - Patriots 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
- p2d2 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.

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 (breakpoint, continue)
    - indicated by white frame, selected by mouse click

Scaling the User Interface

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

Brief Discourse on Globus (1)

- What are Grids?
  - Super Interests for high-performance computing
  - Worldwide collection of high-end resources:
    - supercomputer, 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 m()</td>
<td>0 in m2()</td>
</tr>
<tr>
<td>01 in t()</td>
<td>01 in m3()</td>
</tr>
<tr>
<td>03 in m()</td>
<td>03 in t()</td>
</tr>
<tr>
<td>03 in m()</td>
<td>03 in m()</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 MDS
      - security for Globus initiated jobs
- Distribution Status:
  - plan to distribute under an "OpenSource" copyright.
- Current work:
  - relative debugging of tool-parallelized programs