Improving the Scalability of Comparative Debugging with MRNet

Jin Chao

MeSSAGE Lab (Monash Uni.)
- David Abramson
- Minh Ngoc Dinh
- Jin Chao

Cray Inc.
- Luiz DeRose
- Robert Moench
- Andrew Gontarek
Outline

- Assertion-based comparative debugging
- The architecture of Guard
- Improving the scalability of Guard with MRNet
- Performance evaluation
- Conclusion and future work
Assertion-based Comparative Debugging
Cognitive challenge

- Large dataset on popular supercomputer:
  - Cray Blue Water:
    - > 1.5PB aggregated memory
    - > 380,000 CPU cores

- A large set of scientific data that is non-readable to human:
  - Multi-dimensional
  - Floating-point

- What is the correct state?
Challenge faced by parallel debugging (2)

- Control-flow based parallel debuggers

- Limitations of visualization tools
  - Errors in visualized presentations are hard to detect
Comparative debugging

*Data-centric debugging: focusing the data set in parallel programs*

- Comparing state between programs
  - Porting codes across different platforms
  - Re-writing codes with different languages
  - Software evolution: Modifying/improving existing codes

- Comparing state with users’ expectations
  - Invariants based on the properties of scientific modelling or mathematical theories
  - “Verifying” the correctness of computing status
Assertion for Comparative debugging

“An assertion is a statement about an intended state of a system’s component.”

- **Assertion in Guard:**
  - An ad hoc debug-time assertion
  - Simple assertion:

<table>
<thead>
<tr>
<th>Incorrect code</th>
<th>Correct code</th>
</tr>
</thead>
</table>
| \( 31: \quad \text{for}(j = 0; j < n; j++) \{ \)
| \( 32: \quad \text{for} (i = 0; i < m; i++) \{ \)
| \( 33: \quad p[j][j] = 5000; \} } \) | \( 31: \quad \text{for}(j = 0; j < n; j++) \{ \)
| \( 32: \quad \text{for} (i = 0; i < m; i++) \{ \)
| \( 33: \quad p[j][i] = 5000; \} } \) |

\( \text{assert } a::p@"source.c":33 = 5000 \)
Assertion for Comparative debugging

“An assertion is a statement about an intended state of a system’s component.”

 Assertion in Guard:
- An ad hoc debug-time assertion
- Simple assertion: \texttt{assert }\texttt{$a::var@“source.c”:34 = 1024}$

 Comparative assertions
- Observing the divergence in the key data structures as the programs execute
  \texttt{assert }\texttt{$a::p\_array@“prog1.c”:34 = $b::p\_array@“prog2.c”:37}$

 Statistical assertions
- Verifying the statistical properties of scientific modelling or mathematical theories:
  - standard deviation
  - histogram
Example of statistical assertion: histogram

- **Histogram: user-defined abstract data model**
  - Creating the model with the two phase operation
    - create `$model=randset(Gaussian, 100000, 0.05)`
    - set `reduce` histogram(1000, 0.0, 1.0)
    - `assert $a::my_array@code.c:10~$model < 0.02`
  - estimate operator: ~
    - $\chi^2$ goodness of fit test
Implementation of assertions

- An assertion is compiled into a dataflow graph
  - Dataflow machine

```
set reduce "sum"
assert $a::p_array@"prog.c":34 > 1,000,000
```

```
PROCSET($a) -> Set BKPT
  
  | Go --> BKPT Hit
  | 
  | Get VAR --> Reduce
  
  Sum(p_array) --> Compare
  1,000,000
  
  Exit
```
Dataflow graphs
The Architecture of Guard
Features of Guard (CCDB)

- A general parallel debugger
  - Supporting C, FORTRAN and MPI

- A relative debugger
  - Comparative assertions and statistical assertions

- Client/Server structure:
  - Machine independent command line client
  - Visual Studio Client for Windows
  - SUN One Studio and IBM Eclipse

- Supporting servers from different architecture:
  - Unix: SUN (Solaris), x86 (Linux), IBM RS6000 (AIX)
  - Windows: Visual Studio .NET

- Architecture Independent Format (AIF)
The architecture of Guard

Front-end:
- CLI
- Relative debugger
- Debug Client

Network:
- Socket

Back-end:
- $s_0$
- $s_1$
- $s_2$
- $s_n$
- $p_0$
- $p_1$
- $p_2$
- $p_n$
Features of MRNet

- Tree-Based Overlay Networks (TBONs)
- Scalable broadcast and gather
- Custom data aggregation
General purpose API of MRNet

- User-defined tree topology
  - Topology file: $k$-ary, $k$-nomial and tailored layout

- Communicator
  - A set of back-ends

- Stream
  - A logical data channel over a communicator
  - Multicast, gather and custom reduction

- Packet
  - Collection of data

- Filter
  - Modify data transferred across it
  - WaitForAll, TimeOut, NoWait

- Startup:
  - launching back-end processes
Improving the scalability of Guard with MRNet
The architecture of Guard with MRNet

Front-end:
- Command Line Interface
- Comparative debugger
- Debug Client
- C Wrapper
- MRNet Front-end

Back-end:
- Guard filter
- MRNet BE
- GDB
- s_0
- s_1
- s_2
- s_n
- p_0
- p_1
- p_2
- p_n
Communication with MRNet

- Communication patterns in Guard
  - General parallel debugger

- Topology
  - Balanced, $K$-nomial
  - Placement of MRNet internal nodes
    - Requiring no additional resources

- `procset`:
  - Communicator

- Channels for commands, events and I/O
  - Command stream: synchronous channel
  - Event and I/O Stream: asynchronous channel

- Aggregating redundant messages:
  - `WaitForAll` filter: synchronous channel
  - `TimerOut` filter: asynchronous channel
Creating a communication tree of MRNet

Invoking a debug session on Cray:

Front-end:
- Guard Client
  - launch helper
  - aprun

Back-end:
- MRNet FE
  - apinit
  - agent
  - $p_0$
  - $p_n$
  - $s_0$
  - $s_n$
  - GDB
Comparative assertion with MRNet (1)

assert $a::p\_array@"prog1.c":34 = $b::p\_array@"prog2.c":37$
Comparative assertion with MRNet (2)

- Centralized comparison:

Front-end:

- $s_0$, $s_1$, $s_2$, $s_3$
- $s_2$, $s_3$

Back-end:

- $p_0$, $p_1$
- $d_0$, $d_1$, $d_2$, $d_3$

- $p_0$, $p_1$, $p_2$, $p_3$
- $d_0$, $d_1$, $d_2$, $d_3$

Reconstruct and compare
Data Reconstruction

- Currently support regular decompositions
- Blockmap:
  - Assertions require the debugger to understand data decomposition

```
blockmap test(P::V)
define distribute(block, *)
define data(1024, 1024)
end
```
Comparative assertion with MRNet (3)

- Point-to-point comparison:

Front-end:

Back-end:

Comparison results
Statistical assertion with MRNet

- Standard deviation: two phase operation
  - **Parallel**: calculate a set of primary statistics
  - **Aggregation**: form a full statistical model

Front-end:

Guard client

Back-end:

\[ s_0, s_1, \ldots, s_n \]

\[ ps_0, ps_1, \ldots, ps_n \]
Performance Evaluation
Performance Evaluation

- The configuration of MRNet (3.1.0):
  - A balanced topology
  - Fan-out: 64

- Test bed: ‘Hera’
  - Cray XE6 Gemini 1.2 system
  - 752 computing nodes, each with 32GB of memory.
  - 21,760 CPU cores totally

- General Parallel Debugger
  - SP (Scalar Pentagidgonal) of NAS Parallel Benchmarks (NPB) 3.3.1

- Relative Debugger
  - Comparative assertion: data examples from WRF
  - Statistical assertion: molecular dynamics simulation
Scalability of Invoke Command

The Number of Parallel Processes

- Total-MRNet
- MRNet Instantiation
- MRNet Attachment
- MRNet Overall

Time (second)

The Number of Parallel Processes

$10^4$
Latency of Debugging Commands

The Number of Parallel Processes vs. Time (second) for All bkpt (MRNet) and All step (MRNet)
Message Aggregation

- A memory buffer of 256 bytes was added into the SP program. Its value was inspected under different degrees of aggregation (DoA).
Comparative assertion

- $320 \text{ KBytes} \times 10,000 = 3\text{GB}$. The size of data is from WRF, a production climate model.
Statistical assertion

- Molecular dynamics simulation: $209 \times 10^6$ atoms
  - The atomistic mechanism of fracture accompanying structural phase transformation in AIN ceramic under hypervelocity impact.
Conclusion and Future Work

- MRNet improves the scalability of Guard
  - >20,000 debug servers
- The overhead of creating an MRNet tree
  - Attachment:
  - Instantiation: one BE process per core
- How to take advantage of computing capability of MRNet
  - Comparison across multiple trees?
    - Building a tree across multiple aprun?
  - Programming filter for handling aggregations of statistical assertions
- The best way of maintaining C wrapper for C++ interface?
Related publications


Thanks and Questions?