Non Intrusive Detection & Diagnosis of Failures in High Throughput Parallel Applications

Saurabh Bagchi
Dependable Computing Systems Lab (D CSL)
School of Electrical and Computer Engineering
Purdue University

Work supported by:
IBM, Purdue Research Foundation (PRF), Purdue-CRI

Black-Box Detection & Diagnosis

- Parallel applications have increased in scale
- With that, the possibility of errors in them has increased
- The errors happen due to hardware, software, or configuration problems
- It is important to detect errors quickly and efficiently in terms of resource consumption
- It is also useful to pinpoint which module is responsible for the original error
  - Due to error propagation, the detection may happen at a module distant from the originally erroneous module
- This is the role of the diagnosis system
- Representative errors
  - Application runs out of virtual memory
  - A null call to a function, i.e., returns without performing its job, such as allocating memory to a structure
Design Goals

- Parallel systems are often composed of third-party software
- Therefore, it is desirable for the detection and diagnosis framework, collectively called the fault-tolerance framework, to consider the system as a black-box
  - Source code is not always available
  - But fault tolerance framework can tap into interfaces, such as, function calls/returns
- Fault tolerance system is non-intrusive to the application
  - This is important since performance is often key to the application
  - No explicit probes during runtime
  - Two systems operate asynchronously
- Fast online detection and diagnosis
  - Online process enables possible recovery and continued operation of the application
  - Fast detection and diagnosis reduces the downtime of the application

Solution Approach: Monitor

- Monitor provides the fault tolerance services
- It overhears message exchanges between Protocol Entities (PEs), which can be software modules
- For detection, Monitor matches message interactions against an anomaly-based rule base
- For diagnosis, Monitor creates a dependency graph and runs a rulebase over the deduced state variables of the PEs
State Of Our Solution

- Monitor system applied to several applications – NASA’s Mars Rover simulation, Distributed e-learning application, Distributed e-commerce application
- Expressive rule language designed, which balances ability to express error scenarios and fast matching
  - Created based on application specification and QoS requirement
- Scalability to application comprised of many components through hierarchical Monitor infrastructure
- High throughput applications tend to stress fault tolerance framework with high rate of messages to be verified
  - Monitor has an intelligent sampling mechanism to select messages that are most likely to indicate errors

Other Solutions to this Problem

- Customized error detection and diagnosis
  - Specific to the application
  - Uses internal knowledge of the application
- Debugging support
  - Enables fine-grained traces such that a human can debug the problem more easily than baseline
- Invasive solutions
  - Send probes/tests to the application if behavior deviates from expected behavior
  - Sophisticated algorithms developed to decide on the best probe
- Non-intrusive solutions
  - Statistical clustering of failures with usage of some components
  - Determination of causality between request-response
  - Each system focused on specific type of error (such as, delay)
Conclusion

• We have the Monitor system that does low overhead detection and diagnosis in black-box parallel applications
• Applied to four real applications so far
  – Distributed e-learning
  – Distributed e-commerce
  – Virtualized server environment (IBM)
  – Mars rover simulation (NASA)
• Architecture and implementation generalizable to diverse applications
  – Requirement is access to interfaces

Publications


DCSL URL: www.ece.purdue.edu/~dcl