

# Deterministic Replay

Jiří Patera

---

University of West Bohemia in Pilsen  
Czech Republic



# Overview

- Introduction
- Observation
- Control
- Monitoring approaches
- Deterministic and incremental replay
- On-line vs. off-line approaches
- Further Work

# Introduction

- Distributed system is a system with components that must co-operate and are distributed over a network.
- Distributed programs are:
  - long-time running applications
  - often very complex
  - monitored during run-time
- Debugging of distributed programs is not an easy task.

# Model of Observation

- The detection of a global predicate is divided among a ***checker*** and ***non-checker processes***.
  - Non-checker processes send messages to the checker. They are situated at computation nodes and monitor local variables and messages.
  - The checker process searches for a consistent state that satisfies a predicate.

# Model of Control

- Every process has associated with it a ***supervisory process***.
  - This process observes the underlying process and controls it by delaying, disabling, or by changing the order of incoming and outgoing messages.
- Useful for testing under some desired conditions.

# Probe Effect

- It can affect any observation.
- Modifying a distributed program in any way (adding/removing a piece of code) may alter the timing in the system.
- In order to observe a system we put a probe (auxiliary code, non-checker's code) into the internals of the system.

# Monitoring Approaches

- Observation = Monitoring
- The three monitoring approaches:
  - Hardware
    - Expensive, large amount of data from the hardware data layer, no probe-effect
  - Software
    - Cheap, prone to probe-effect (can be eliminated)
  - Hybrid
    - Mixture of hardware and software monitors

# Browsing, Replay, Simulation

- The amount of information that must be recorded depends on how it will be used.
- ***Browsing***: Event history is examined through the use of specialized tools.
- ***Replay***: Debugger uses the event history to control a re-execution.
- ***Simulation***: Event history is used to simulate surroundings of any process.

# Replaying a Distributed Execution

- Two possible ways of record/replay:
  - Deterministic replay
    - useful for short computations
    - records all events from the beginning of the computation
  - Incremental replay
    - useful for long-running computations
    - records gradually last known consistent state and all events that happened after it

# Pros and Cons

- Advantage of the record/replay approach is that it is not as complex as the model checking approach (which checks all possible executions of a distributed program).
- However, important drawback is that it checks only one singular run of many possible (the recorded one).

# The Time Machine

- Debugging technique based on deterministic replay.
- It records not only events but also timing, interrupts and task-switches.
- It is based on the three elements:
  - Recorder: Collects necessary information.
  - Historian: Analyzes the recorded data and creates chronological time-line.
  - Actor: Replays the recorded history.

# On-line vs. Off-line

- On-line debugging
  - We have no information on the future of the computation, we know only the past.
  - Thus, we cannot predict e.g. deadlocks.
- Off-line debugging
  - We have all information not only on the past of the computation but also on its future.
  - So we can predict future (i.e. The things that have happened in the recorded computation).

# Further Work

- State of the art has been presented.
- In the future we will focus our research on the record/replay approach and distributed Java environment.
- We are planning to introduce a new method for observation and control of distributed Java programs. The method will be based on a modification of the Java bytecode.

# Thank You

Thank you for your attention...

Now it is time for your questions.