Control and Debugging of Distributed Programs Using Fiddle

J. Lourenço, J. Cunha, V. Moreira
{jml, jcc, vrm}@di.fct.unl.pt

Departamento de Informática – CITI Research Centre
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa - Portugal
Agenda

- Motivation and Goals
- FIDDLLE
- Integrated testing and debugging
- Conclusion and future work
Motivation and Goals

- Difficulties for developing parallel and distributed applications
  - Many dynamic interacting entities
  - Execution on multiple processors
  - Global non-deterministic behavior
  - Need to adapt to new hardware, OS, middleware and/or programming languages
Motivation and Goals

- Traditional debugging services
  - Based on sequential debugging
    - Breakpoints, inspect core, stack, source files

- Distributed debugging services
  - Monitoring and logging application behavior
  - Execution replay
  - Specific programming models

- Integration in development environments
  - Developers don’t use a single tool
    - High(er) level programming languages and editors
    - Simulators
    - Visualizers
    - Performance evaluators, etc
Motivation and Goals

- Requirements for flexible development tools
  - Ability to adapt to new computing platforms and/or user requirements
  - Decouple User Interface and Services
  - Separation between basic and extended functionalities
  - Tool cooperation
FIDDLE

- Distributed Debugging Engine
- Operation
- Outline of the architecture
- Functionalities
- Current implementation status
FIDDLE
Distributed Debugging Engine

- Support for interactive correctness debugging
- Separation of User Interface and Engine
- Extensibility
- Support for high-level user-defined abstractions
- Easy integration in Parallel Software Development Environments
FIDDLE
Outline of the architecture
FIDDLE

Functionalities

- Layer 0s
  - Local debugging (one machine)
  - Multiple target processes
  - Multi-threaded target processes
- Layer 0m
  - Multi-threaded client
- Layer 1m
  - Remote debugging
- Layer 2m
  - Multiple concurrent clients
- Layer 3m
  - Events and call-back routines
FIDDLE
Current implementation status

- **FIDDLE Engine**
  - Fully implemented except layer 3m, which is ongoing work

- **Client Tools**
  - Text debugging Interfaces
  - Graphical debugging Interfaces
  - Deterministic (re)execution
Integrated Testing and Debugging

- Motivations for integrated testing and debugging
- Tool composition using DEIPA
- A working example
Integrated Testing Debugging

Motivation

Static Analysis + Test phase

Dynamic Analysis + Debug phase

Source code

Interactive test tool

Generation

Scenario specification

Test phase

Debug phase

User

Distributed Execution

Controller

Distributed Debugger
Integrated Testing Debugging
Tool composition using DEIPA

(1) Technical University of Gdansk, Poland
Integrated Testing Debugging
A working example

spawn

echo_client  Ok  echo_server

Even number
Odd number
Ok
Integrated Testing Debugging
A working example
Conclusions

- FIDDLE
  - Multi layered design
    - Extensibility (to add new functionalities)
    - Flexibility (to adapt to different needs)

- Integrated Testing and Debugging
  - Controlled execution
  - Tool composition
    - Testing and debugging tools
    - DEIPA
Future work

- FIDDLE’s layer 3m
- Further experiments on tool integration
  - Debugging and visualization
  - Debugging of visual programming models and languages
- Generation of test scenarios
- Debugging large scale applications