A Planning Based Approach to Failure Recovery in Distributed Systems

Naveed Arshad
Dennis Hiembigner, Alexander L. Wolf
University of Colorado at Boulder

Workshop on Self Managed Systems (WOSS’04)
Oct 31st, 2004
Introduction

• Automated failure recovery in systems using dynamic reconfiguration and AI planning
  – Recover in minimum time (but not real-time)
• Target: component based heterogeneous distributed systems
  – Application level reconfiguration
  – Not OS or network level (yet)
Goals for Failure Recovery

• Automated process
• Minimize downtime
• Handle complex failures
  – Ripple effects of failures
  – Hard to anticipate the failed state
    • Large number of possible failed states
  – Large number of recovered states
Approach (Sense-Plan-Act)

• Sensing
  – Determining if a failure has occurred

• Planning
  – Calculating the ripple effects
  – Devising a plan for failure recovery

• Acting
  – Executing the plan on the actual system
Planning

- **Domain (Static)**
  - Semantics of the System
- **Initial State**
  - Configuration of the system at the start (i.e. the failed state)
- **Goal State**
  - Configuration of the system at the end (i.e. the recovered state)
- **Plan**
  - Set of actions to get from the initial state to the goal state
An Example
A Failure Scenario

Clients

Machine 1

1 2 3 4 5 6

Machine 2

Servlet Engine 1

Machine 3

Application Server 1

Machine 4

Database

Machine 5

Servlet Engine 2

Application Server 2

Failed

Affected

Normal
Calculating Ripple Effects

• Dependency model is used to dynamically calculate effects of component failure on other components

• Components are classified into three different kinds
  – Failed Components
  – Affected Components
  – Normal Components
Styles for Recovered States

• Explicit Recovered State
  – Stating a recovered state for the planner
    • servletEngineWorking(servletengine1 machine2)

• Implicit Recovered State
  – Asking the planner to find a recovered state
    • servletEngineWorking(servletengine1)

• All goal state specifications have significant amounts of implicit specification
  • If not, then planner is not needed
Domain Specification

**Objects**
applicationserver machine webserver servletengine

**Predicates**
ServletEngineInstalled (servletEngine, machinename)
ServletEngineStarted (servletEngine)
ServletEngineWorking (servletEngine)
machineFailed (machinename)
ApplicationServerWorking (applicationServer)
WebServerWorking (webserver)
...

**Functions**
MachineRAM (machinename)
MachineStartTime (machinename)
ServletEngineInstallTime (servletEngine)
ServletEngineConnectTimeWithWS (servletEngine)
...
Domain Specification (cont.)

Actions

Start-Machine (machinename)
  Duration (= (MachineStartTime (machinename)))
Preconditions
  (not (machineFailed machinename))
effects
  machineStarted (machinename)

Install-Servlet-Engine (servletEngine machinename)
Connect-ServletEngine-AS (servletEngine, applicationserver)...
Connect-ServletEngine-WS (servletEngine, webServer)...
...

Initial State

**Objects**
- applicationserver1 – applicationserver...
- servletengine2 – servletengine...
- webserver – webserver
- database – database
- machine1 – machine...

**Initial State**
- machineStarted (machine1)
- machineFailed (machine2)
- machineStarted (machine3)
- ...

= (machineRAM (machine1) 512)
= (machineRAM (machine3) 1024)
= (machineRAM (machine4) 1024)
- ..
Initial State (cont’d)

= (machineJDK (machine1) 1.4.2)
= machineJDK (machine3) 1.3)
.. 
= (machinePlatform (machine1) Unix)
= (machinePlatform (machine3) win2k)
..
servletEngineWorking (servletengine2, machine5)
applicationServerWorking (applicationserver2)
databaseWorking (database)
..
Goal State

Goal State

servletEngineWorking (servletengine1)
applicationServerWorking (applicationserver1, machine3)

Metric
Minimize Total-time
Plan

1. Install-Servlet-Engine (servletEngine1, machine1)
2. Connect-ServletEngine-AS (servletEngine1, applicationserver1)
3. Connect-ServletEngine-WS (servletEngine1, webServer))
4. Connect-Client ...
5. ...

![Diagram with machines and services connected]
Present Work

• Prototype (Planit) is Under Development
  – Sensing
    • Java based sensing framework using Siena
  – Planning using planner named LPG-TD (Università degli Studi di Brescia)
  – Currently, using applications developed on Prism middleware (USC/UCI) as our target applications
Open Questions

• Dependency Modeling
  – How and when the dependencies should be updated?
    • Static vs. Dynamic?
  – Which dependency model to be used?

• System Learning
  – How the system learns over time?
    • Case Based Reasoning?
Summary

• Our initial results show promising prospects for using planning in failure recovery

• The next step is to use this technique in highly distributed systems and in other areas like
  – Performance Improvement
  – Distributed System Management
  – Fault Tolerance
Data Flow Diagram

Dependency Events Database

Update the Dependency Model by Dependency Modeler

Dependency Model

State Events Database

Update the State Model by State Modeler

State Model

Check the present configuration of the system

Current Configuration of the system

Execute the Reconfiguration

Script

Configuration Database

Synthesize the two models

System Model

Check if a reconfiguration is required

Model for Comparison

Find a new configuration

Target Configuration

Find a Plan for reconfiguration

Plan Library

Plan

Script Library

Find a new configuration

Find a Plan for reconfiguration

Script

Legend

Information or Model

Process

External Database or Library

Information or Model used as an Input to a process

Information required on a need basis
## Experimental Setup

<table>
<thead>
<tr>
<th>Experiment No</th>
<th>Components</th>
<th>Connectors</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
## Explicit Configurations

<table>
<thead>
<tr>
<th>Experiment</th>
<th>No of Plans Found (in 30 sec)</th>
<th>Time to Find the Best Plan (in sec)</th>
<th>Duration of the Best Plan (in sec)</th>
<th>Duration of the worst Plan (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>12.39</td>
<td>67</td>
<td>83</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>18.64</td>
<td>66</td>
<td>137</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>27.95</td>
<td>100</td>
<td>144</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>23.00</td>
<td>76</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>17.93</td>
<td>138</td>
<td>N/A</td>
</tr>
</tbody>
</table>
## Implicit Configurations

<table>
<thead>
<tr>
<th>Experiment</th>
<th>No of Plans Found (in 60 sec)</th>
<th>Time to Find the Best Plan (in sec)</th>
<th>Duration of the Best Plan (in sec)</th>
<th>Duration of the Worst Plan (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4.92</td>
<td>62</td>
<td>70</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>56.71</td>
<td>65</td>
<td>81</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>36.99</td>
<td>108</td>
<td>124</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Sensing

• Getting the information
  – Inserting sensors in the components and machines to detect failures using heartbeats and explicit pinging
  – A monitor receives the raw information and makes decision about a failure
  – Monitors can also be stacked in subsystems to form a hierarchy
  – Monitors can change various parameters to reduce the impact on the network
Other Potential Areas

- **Fault Tolerance**
  - To prevent faults from developing that lead to a failure

- **System Management**
  - Automated management of the systems

- **Performance Improvement**
  - Improve the performance of the system using planning

- **May need some modifications in our approach to accommodate these areas**
Acting

- The plan is converted into an executable script
- The script is executed on the system for recovery
- A feedback loop is established to find if the recovery process is carried out successfully