Increasing System Availability with Local Recovery based on Fault Localization

Hasan Sozer
TRESE Technical Talks 18-02-2010

Outline
- Introduction: Local Recovery
- Fault Localization
- Spectrum-based Fault Localization (SFL)
- The Approach
- Evaluation & Results
- Conclusions

Introduction
- Fault → Error → Failure
- Fault tolerance
  - cope with errors at run-time
- Local recovery
  - recover only the erroneous parts of the system,
  - while the other parts are kept operational
- Goal: Increasing system availability

Why local recovery based on fault localization?
- Erroneous parts of the system are not always the ones at fault
  - local recovery is not always successful
  - the same error can be triggered after recovery
- Possible solution:
  - restart progressively larger subsets,
  - eventually the whole system
- Fault localization supporting recovery
  - localizes the root causes of errors
  - increases system availability by 23.4% on average

Local Recovery
- FLORA: A framework for introducing local recovery

Fault Localization (aka diagnosis)
- Locate faults that are responsible for detected errors
  - support debugging process
  - support recovery mechanisms

  - Statistical approaches
    - data collection at run-time and
    - abstraction of program traces (aka program spectra)
  - Reasoning approaches
    - utilization of a model of expected behavior,
    - combined with observations
Spectrum-based Fault Localization (SFL)

- Obtain program spectra (execution profiles) for different runs, tests, activities of a system.
- Spectra tells us which parts are active.
- Compare the activity of different parts with the occurrence of errors.

Fault Diagnosis with SFL

- Block Hit Spectrum

\[
\begin{pmatrix}
X_1 & X_2 & \ldots & X_i & \ldots & X_n \\
\end{pmatrix}
\]

- Block:
  - C statement (possibly compound)
  - e.g., cases of a switch statement

- Spectra for \( m \) test cases

\[
\begin{array}{cccc}
X_{11} & X_{12} & \ldots & X_{1n} \\
X_{21} & X_{22} & \ldots & X_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
X_{m1} & X_{m2} & \ldots & X_{mn} \\
\end{array}
\]

\( e_i = 1 \): block \( i \) executed
\( e_i = 0 \): block \( i \) not executed

- Row \( i \): the blocks that are executed in case \( i \)

- Error detection per test case

\[
\begin{array}{cccc}
X_{11} & X_{12} & \ldots & X_{1n} \\
X_{21} & X_{22} & \ldots & X_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
X_{m1} & X_{m2} & \ldots & X_{mn} \\
\end{array}
\]

\( e_i = 1 \): error in the \( i \)th test
\( e_i = 0 \): no error in the \( i \)th test

Column \( j \): the test cases in which block \( j \) was executed
Fault Diagnosis with SFL

- Compare every column vector with the error vector

\[
x_{11} x_{12} \ldots x_{1n} \quad e_j
\]

\[
x_{21} x_{22} \ldots x_{2n} \\
\vdots \quad \vdots \\
x_{m1} x_{m2} \ldots x_{mn} \quad e_j
\]

- Jaccard similarity coefficient

\[
s_j = \frac{a_{11} + a_{10} + a_{01}}{a_{11} + a_{10} + a_{01} + a_{00}}
\]

Example: Jaccard similarity coefficient

\[
s_j = \frac{a_{11}}{a_{11} + a_{00} + a_{01}}
\]
Fault Diagnosis with SFL

- For every block: similarity with the error "block"

\[
\begin{array}{cccc}
X_{11} & X_{12} & \ldots & X_{1n} \\
X_{21} & X_{22} & \ldots & X_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
X_{m1} & X_{m2} & \ldots & X_{mn} \\
\end{array}
\]

\[s_1, s_2, \ldots, s_n\]

The block with the highest \(s_i\) most likely contains the fault.

- Example:

\[
\begin{array}{cccccc}
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 \\
\end{array}
\]

\[
\frac{2}{3}, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, \frac{1}{2}, \frac{2}{3}
\]

Fault Diagnosis with SFL

- Example: Rational Sort

```python
# Python code example

# Example: Rational Sort

Example: Rational Sort
```
Fault Diagnosis with SFL

- Example: SFL applied on 6 runs

<table>
<thead>
<tr>
<th>Block 1</th>
<th>Block 2</th>
<th>Block 3</th>
<th>Block 4</th>
<th>Block 5</th>
<th>Block 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The Approach

- CASE STUDY: Mplayer
  - 5 Blocks: RM, CM, RU _GUI_, RU _Audio_, RU _Mpcore_
  - Error detection with timeouts and process health monitoring
    - Test cases:
      - Key presses
      - Processing of video frames
      - Spectra updated after every test case completion
    - Error detection

Results

- Local recovery improved the availability significantly: 83.8%
  - when successfully applied, i.e., erroneous RU ≠ faulty RU
- SFL always predicted the faulty RU correctly
- SFL is lightweight w.r.t. performance

The Approach

- Local Recovery based on Fault Localization
- FLORA + SFL (integrated in CM)
- Block: Recoverable Unit (RU)
- Spectra based on messages

Evaluation

- 3 types of faults injected
  - Illegal memory operation that causes a fatal error
  - Busy waiting or skipping a message that causes a deadlock
  - Sending of messages that causes a buffer overflow and in turn, a fatal error at the destination of the messages

- Systematically activate faults
- Observe the results and log files
- Measure mean time to recover over 100 runs

Results

- Diagnosis information enables more effective recovery strategies
  - Restart both the faulty and erroneous RUs at once, or
  - Restart the whole system directly
    - Rather than first trying (unsuccessfully) local recovery
### Results

- Improvement in availability
  - restart both the faulty and erroneous RUs at once: 32.8%
  - restart the whole system directly: 13.9%

### Conclusion

- Local recovery increases system availability significantly
- Fault localization can further improve the effectiveness of local recovery
- SFL is a lightweight approach for fault localization

### Limitations & Future Work

- Consideration of different fault/error types
- Requires extension with new error detection techniques
- Case studies with other types of applications

### Questions?