Exploiting OS Support for Transparent Failure Detection and Customized Recovery

Long Wang, Zbigniew Kalbarczyk, Ravishankar K. Iyer

Motivations
- Abundant system knowledge and resource management in OS enable efficient and transparent handling of failures
- Abundant knowledge on hardware, drivers, middleware, applications.
- Rich toolkit facilitates transparent detection and customized recovery.
- System calls, interrupt handlers, signal handlers, memory access, hardware features, system call, interrupt, signal handling, memory access, hardware features.

OS Hang/Panic Detection
- When a context switch occurs:
  - The time stamps for all the active sections in the thread are saved.
  - If the sum for a section is larger than the corresponding check value, it suggests a hang.
- Examples:
  - The http request (section ID, thread ID, current count, time stamp)
  - Loop, loop body (i.e., one run of a loop), makes holding I/O protection block
  - Embedded sections: e.g., embedded loops

Application Hang Detection
- Algorithm
  - When a context switch occurs:
    - The counter records the number of instructions executed during the section.
    - The sum for a section is calculated as the corresponding check value for a hang.
    - If the sum for a section is larger than the corresponding check value, it suggests a hang.
- Examples:
  - Overlapped sections: e.g., a loop section overlapping with a mutex section

Customized Transparent Checkpoint/Recovery
- For stateless transaction-based applications:
  - Multiple important services: http, ftp, smtp, etc.
  - The application state does not depend on processing history.
- For stateful services: e.g., Apache web server
  - The application state depends on processing history.
  - Characteristics of Apache web server
  - Application state is saved to a persistent storage.

OS Hang/Panic Detection
- Potential causes:
  - Blocking operations in poorly-written drivers
  - Lack of recovery logic
  - System calls are intercepted to provide transparent checkpointing.
  - Application crash/hang detection is applied for failure detection.
  - Upon a failure the proper clean state is restored and the request is played back with the sent data without.

Application Hang Detection
- Algorithm
  - When a context switch occurs:
    - The counter records the number of instructions executed during the section.
    - The sum for a section is calculated as the corresponding check value for a hang.
    - If the sum for a section is larger than the corresponding check value, it suggests a hang.
- Examples:
  - Overlapped sections: e.g., a loop section overlapping with a mutex section

Customized Transparent Checkpoint/Recovery
- Checkpoint/Recovery of the child server
  - The clean states while waiting for a connection request or http request
  - Recorded information for request replay
  - The connection to the client
  - The http request
  - The data size the server has sent out in the reply to a request

Hardware Interface

Control

Kernel

FMC Core

Hardware

Checkpoint/Recovery

Configuration

Failure Mitigation Driver

OS Hang/Panic Detection

Application Hang Detection

Motivations

OS Hang/Panic Detection (cont.)

Application Hang Detection (cont.)

Customized Transparent Checkpoint/Recovery (cont.)

Checkpoints

Application states

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoints

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery

Checkpoint/Recovery