Performance and Status Monitoring of JavaEE Business Applications

Andreas Wälchli
AppCheck
Andreas Wälchli
Project Description

• Customer: ISC-EJPD
• Implement the ApplicationCheck WebService → Provide Status Information about the Application
• Configurable, Extendable, Generic
ApplicationCheck – WebService

Requirements:
• 1 Predefined “query” message
• Server performs checks and returns a result
• Results can be nested

@javax.ejb.Local
public interface ICNChecker {
  public CheckResponse check(String uid);
}

public class CheckResponse implements Serializable {
  private String checkName;
  private String message;
  private String description;
  private String stackTrace;
  private String errorMessage;
  private CheckResult internalResult;
  private List<CheckResponse> subChecks;
}

public enum CheckResult {
  CHECK_OK, CHECK_FAILED;
}
ApplicationCheck – Restrictions

• Responses are binary (CHECK_OK, CHECK_FAILED)
• Raw data is often fuzzy
• Check does not necessarily fail if a sub-check fails

→ Apply a decision function to determine result
Project Constraints: Minimize Everything

Data Storage

Preprocessing ↔ Postprocessing
GC Statistics Without Storing Data Points?

- Store basic metrics (min/avg/max/count) for fixed time period
- Keep a fixed number of these periods
- Discard period if too old

- Simple Preprocessing
- Simple Postprocessing
- Constant Memory Footprint
GC Statistics

Update
• $c_{n+1} = n + 1$
• $\tilde{v}_{n+1} = \min(\tilde{v}_n, v_{n+1})$
• $\bar{v}_{n+1} = \bar{v}_n \frac{n}{n+1} + \frac{v_{n+1}}{n+1}$
• $\hat{v}_{n+1} = \max(\hat{v}_n, v_{n+1})$

Merge
• $c_{[a,b]} = c_a + c_b$
• $\tilde{v}_{[a,b]} = \min(\tilde{v}_a, \tilde{v}_b)$
• $\hat{v}_{[a,b]} = \max(\hat{v}_a, \hat{v}_b)$
• $\bar{v}_{[a,b]} = \begin{cases} \text{undef} & \text{if } c_a = c_b = 0 \\ \bar{v}_a & \text{if } c_b = 0 \\ \bar{v}_b & \text{if } c_a = 0 \\ \bar{v}_a \frac{c_b}{c_{[a+b]}} + \bar{v}_b \frac{c_a}{c_{[a+b]}} & \text{else} \end{cases}$
Timer Monitoring (Existence Check)

• Can’t get list of all Timers
• Each Bean knows its timers
• Can’t access that list externally

→ Ask the Bean!

• Monitored Bean must implement an interface and method to grant access
Timer Extraction

- Class/Timer List
- Class<?> Object
- Object Instance
- Timer List
Future

• Interceptor-based Timer Monitoring
• Recognize Failure Patterns and provide Cause Hints
Release

• http://awae.ch/bachelor
• https://github.com/ksmonkey123/appcheck
• Maven Group ID: ch.awae
Demo

- Application designed with issues
  - OldGen Memory Leak
  - EdenSpace spammed with short-lived Objects

→ More and More Tests should fail (but only GC Tests!)