SPL progress report

http://d3s.mff.cuni.cz

Vojtěch Horký
horky@d3s.mff.cuni.cz
Previously on SPL

- Capture performance assumptions...
  - ... together with the code
  - ... in platform independent way
  - ... using sound statistics testing
- Prototype introduced
- Discussion on future development
Step sideways – Java & reflection
Calling methods in Java

- Direct call
  - Compiled class
  - Compilation on-the-fly
- Javassist
- Reflection

![Graph showing performance comparison between different methods in Java 1.7 using OpenJDK, 64bit. The x-axis represents various method calls: Compiled, On-the-fly, Javassist, Reflection. The y-axis represents time in microseconds (μs). The bar heights indicate the performance differences.]
Prototype innovations
Miscellaneous

• Bug fixes etc.
• Getting rid of reflection
• Linux & Windows
DiSL-powered features

- Measuring methods at run-time
- Unit tests as input generators
Software project

“Developer tools for SPL testing”
Goals

- SPL framework
  - Find, what has to be measured
  - Get it
    - Fetch from repository
    - Check-out different revision
  - Compile it
  - Measure it
  - Evaluate the results
Goals (cont.)

- Eclipse integration
- Hudson integration
- Demonstration on someone else's project
Status

- Started a month ago
- [http://sourceforge.net/p/spl-tools/](http://sourceforge.net/p/spl-tools/)
SPL-driven self-adaptation
Vision paper for CORCS

- Authors LB, TB, VH, JK, PT
- SPL to express adaptation rules
  - Performance-awareness framework
- Use cases
  - Select faster component
  - Check component contract
  - Observing trends
  - Future performance prediction

...was accepted
The proposed framework

- User friendly
  - Automatic instrumentation
  - SPL formulas

- Above components...
  - Control their creation and binding
  - Rules part of architecture description

- ...or inside code
  - Rules directly part of business code
Developer's point of view

- **Probe**
  - Collects data
  - Automatic instrumentation / manually created

- **Data source**
  - View on data collected by the probe

- **Formula**
The Big Picture
Select faster component

Use content compression component A or B?

\[ \text{Perf}^A < \text{Perf}^B \]

Formula f = Formula.create("A < B");
  f.bind("A",
      new InstrumentingSource("org.pkg.A.compress"));
  f.bind("B",
      new InstrumentingSource("org.pkg.B.compress"));
  if (f.evaluate()) {
      ...
  }
Check component contract

- *Time to respond to a request is not more than 6s with 99% probability.*
- Extend SPL with absolute times

\[
Perf^{response} < [99\%] \ 6 \text{s}
\]

Formula \( f = \text{Formula.create("R <[0.99] 6")}; \)
\( f.\text{bind("R", ...}); \)
Observing trends

- **The performance degraded twice in last 10 minutes.**
- Extend SPL with references to data obtained at particular time slot

\[ L_{\text{now last 10 min}}(T^A) > 2 \cdot L_{\text{10 min ago last 10 min}}(T^A) \]

```java
Formula f = Formula.create("A > B");
f.bind("A", new SlidingTimeSlot("...", 0, 10));
f.bind("B", new SlidingTimeSlot("...", 10, 10));
```
Future performance prediction

- *Is the performance likely to be twice as worse within 15 minutes?*
- Linear interpolation of existing data

\[
L_{\text{now } + 5 \text{ min}} \left( T^A \right) > 2 \cdot L_{\text{10 min ago}} \left( T^A \right)
\]

```java
Formula f = Formula.create("A > B");
f.bind("A",
    new InterpolatingSource(5,
        new SlidingTimeSlot("...", 0, 10)
    ));
f.bind("B", new SlidingTimeSlot("...", 10, 10));
```
Prototype demo

- Simulation of a web server
  - Access log of studuj-matfyz.cz
- IPOJO components
- Replication of the server component
Rules for starting new instance

- Detect burst traffic
- Detect gradual traffic increase
- Use yesterday as a template
Burst traffic

No server replication
With server replication
Cost (number of instances)
Gradual traffic increase
Yesterday as a template
Future plans

- “Performance prediction” use case
- Automatic instrumentation
- Truly run “above” components
- Better demo application
That's all
SPL status board

- Accepted vision paper
- In progress
  - Software project
  - Performance-aware adaptation prototype

Thank you for your attention