STEP: A Framework for the Efficient Encoding of General Trace Data

Rhodes Brown, Karel Driesen, David Eng, Laurie Hendren, John Jorgensen, Clark Verbrugge and Qin Wang

Sable Research – McGill University
http://www.sable.mcgill.ca

November 18, 2002
Outline

• The problems with current tracing methods.

• A solution: STEP
  – Framework overview
  – Example: JVMPI allocation data
  – Uses of STEP data
Problem #1 With Traces:
Formats are too event-specific

...look at the Java run-time environment:

- Virtual method dispatch
- Garbage collection
- Explicit concurrency

⇒ Need an openly extensible format
Problem #2 With Traces:
They are very large

- Generic compression methods are not good enough
- Specialized schemes can lead to better compression
- Often require tuning to specific data set

⇒ Need adjustable encoding strategies
The STEP System

STEP: a framework for encoding a variety of trace events

Main Features:

• An event definition language & compiler to generate interface elements
  ⇒ flexibility, extensibility, & documentation

• An encoding engine to implement various reduction schemes
  ⇒ encapsulation & compactness
The STEP Framework: Overview

Event Producers
- Program
  - JVM
- Modified Program
- JVMPI
- Modified Source

Data Objects
- Data Object Definitions
- Encoding Strategies
- Data Objects
- stepc

Trace Encoding
- Encoding Strategies
- trace file

Event Consumers
- STEP Decoder
- Decompress
- Data Objects
- EVolve
- JIMPLEX
- Analysis Tool
Example: Encoding JVMPI Data

Benchmark: _213_javac

- 6319542 allocation records
- naïve (verbatim) format: 150.67 MB
- compressed (.gz): 23.18 MB → 15.38%
JVMPI Event
.env

Allocation
.arena
.class
.array
.size
.newObj

Free
.objId

Method Enter
.method
.target
STEP-DL for Allocation Events

record OBJECT_ALLOCC extends JVMPI_Event {
    int arenaId;
    int classId <property: "address"><encoding: "size=4">;
    int isArray <property: "unsigned">;
    int size <property: "unsigned">;
    int newObjId <property: "address"><encoding: "size=4">;
}

STEP-DL for Allocation Events

record OBJECT_ALLOC extends JVMPI_Event {
  int arenaId;
  int classId <property:"address"><encoding:"size=4">;
  int isArray <property:"unsigned">;
  int size <property:"unsigned">;
  int newObjId <property:"address"><encoding:"size=4">;
}

- Structure
record OBJECT_ALLOC extends JVMPI_Event {
    int arenaId;
    int classId  <property:"address"><encoding:"size=4">;
    int isArray  <property:"unsigned">;
    int size     <property:"unsigned">;
    int newObjId <property:"address"><encoding:"size=4">;
}

- Structure

- Inheritance
STEP-DL for Allocation Events

record OBJECT_ALLOC extends JVMPI_Event {
    int arenaId;
    int classId <property: "address"><encoding: "size=4">;
    int isArray <property: "unsigned">;
    int size <property: "unsigned">;
    int newObjId <property: "address"><encoding: "size=4">;
}

- Structure
- Inheritance
- Attributes
STEP-DL for Allocation Events

record OBJECT_ALLOCA extends JVMPI_Event {
    int arenaId;
    int classId <property:"address"><encoding:"size=4">;
    int isArray <property:"unsigned">;
    int size <property:"unsigned">;
    int newObjId <property:"address"><encoding:"size=4">;
}

- Structure
- Inheritance
- Attributes
# Baseline Comparison

<table>
<thead>
<tr>
<th></th>
<th>raw size</th>
<th>.step size</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>150.67 MB</td>
<td>102.51 MB</td>
<td>68.04%</td>
</tr>
<tr>
<td>compressed</td>
<td>23.18 MB</td>
<td>18.21 MB</td>
<td>78.56%</td>
</tr>
<tr>
<td>ratio</td>
<td>15.38%</td>
<td>17.76%</td>
<td>12.08%</td>
</tr>
</tbody>
</table>
## Baseline Comparison

<table>
<thead>
<tr>
<th></th>
<th>raw size</th>
<th>.step size</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>150.67 MB</td>
<td>102.51 MB</td>
<td>68.04%</td>
</tr>
<tr>
<td>compressed</td>
<td>23.18 MB</td>
<td>18.21 MB</td>
<td>78.56%</td>
</tr>
<tr>
<td>ratio</td>
<td>15.38%</td>
<td>17.76%</td>
<td>12.08%</td>
</tr>
</tbody>
</table>

- Record size reduction
## Baseline Comparison

<table>
<thead>
<tr>
<th></th>
<th>raw size</th>
<th>.step size</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>150.67 MB</td>
<td>102.51 MB</td>
<td>68.04%</td>
</tr>
<tr>
<td>compressed</td>
<td>23.18 MB</td>
<td>18.21 MB</td>
<td>78.56%</td>
</tr>
<tr>
<td>ratio</td>
<td>15.38%</td>
<td>17.76%</td>
<td>12.08%</td>
</tr>
</tbody>
</table>

- Record size reduction
- Compression ratio
### Baseline Comparison

<table>
<thead>
<tr>
<th></th>
<th>raw size</th>
<th>.step size</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>150.67 MB</td>
<td>102.51 MB</td>
<td>68.04%</td>
</tr>
<tr>
<td>compressed</td>
<td>23.18 MB</td>
<td>18.21 MB</td>
<td>78.56%</td>
</tr>
<tr>
<td>ratio</td>
<td>15.38%</td>
<td>17.76%</td>
<td>12.08%</td>
</tr>
</tbody>
</table>

- Record size reduction
- Compression ratio
- Overall reduction
Remove Constant Values

record JVMPI_Event {
    int envId <property:"address"><encoding:"size=4">
}

Remove Constant Values

record JVMPI_Event {
    int envId <property:"address"><encoding:"size=4">
        <encoding:"repeat">;
    }
}
Remove Constant Values

```java
record JVMPI_Event {
    int envId <property:"address"><encoding:"size=4"><encoding:"repeat">;
}
```

<table>
<thead>
<tr>
<th></th>
<th>raw size</th>
<th>.step size</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>150.67 MB</td>
<td>78.41 MB</td>
<td>52.04%</td>
</tr>
<tr>
<td>compressed ratio</td>
<td>23.18 MB</td>
<td>17.35 MB</td>
<td>74.87%</td>
</tr>
<tr>
<td></td>
<td>15.38%</td>
<td>22.13%</td>
<td>11.52%</td>
</tr>
</tbody>
</table>
The Identifier Technique

record OBJECT_ALLOC extends JVMPI_Event {
    int arenaId;
    int classId  <encoding:"identifier">;
    int isArray;
    int size;
    int newObjId;
}

The Identifier Technique

record OBJECT_ALLOC extends JVMPI_Event {
    int arenaId;
    int classId <encoding:"identifier">;
    int isArray;
    int size;
    int newObjId;
}
The Identifier Technique

record OBJECT_ALLOC extends JVMPI_Event {
    int arenaId;
    int classId <encoding:"identifier">;
    int isArray;
    int size;
    int newObjId;
}

<table>
<thead>
<tr>
<th></th>
<th>raw size</th>
<th>.step size</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>150.67 MB</td>
<td>63.58 MB</td>
<td>42.20%</td>
</tr>
<tr>
<td>compressed</td>
<td>23.18 MB</td>
<td>17.24 MB</td>
<td>74.38%</td>
</tr>
<tr>
<td>ratio</td>
<td>15.38%</td>
<td>27.11%</td>
<td>11.44%</td>
</tr>
</tbody>
</table>
The Difference Technique

record OBJECT_ALLOC extends JVMPI_Event {
    int arenaId;
    int classId;
    int isArray;
    int size;
    int newObjId <encoding:"delta=1048576">;
}

The Difference Technique

record OBJECT_ALLOC extends JVMPI_Event {
    int arenaId;
    int classId;
    int isArray;
    int size;
    int newObjId <encoding:"delta=1048576">;
}
The Difference Technique

record OBJECT_ALLOC extends JVMPI_Event {
    int arenaId;
    int classId;
    int isArray;
    int size;
    int newObjId <encoding:"delta=1048576">;
}

<table>
<thead>
<tr>
<th></th>
<th>raw size</th>
<th>.step size</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>150.67 MB</td>
<td>46.01 MB</td>
<td>30.54%</td>
</tr>
<tr>
<td>compressed</td>
<td>23.18 MB</td>
<td>1.97 MB</td>
<td>8.49%</td>
</tr>
<tr>
<td>ratio</td>
<td>15.38%</td>
<td>4.28%</td>
<td>1.31%</td>
</tr>
</tbody>
</table>
### Example Use II - JIMPLEX

#### Methods (9)

<table>
<thead>
<tr>
<th>declaration</th>
<th>name</th>
<th>statements</th>
<th>calls (622)</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void <code>int</code>&lt;br&gt;<code>init()</code></td>
<td><code>init</code></td>
<td>3 I</td>
<td>0</td>
</tr>
<tr>
<td>public static void <code>main</code>(java.lang.String[])&lt;br&gt;<code>main</code></td>
<td><code>main</code></td>
<td>104</td>
<td>1</td>
</tr>
<tr>
<td>private static java.lang.Long <code>calcPPS</code>(long, long)&lt;br&gt;<code>calcPPS</code></td>
<td><code>calcPPS</code></td>
<td>31</td>
<td>104</td>
</tr>
<tr>
<td>private static boolean <code>isInteger</code>(double)&lt;br&gt;<code>isInteger</code></td>
<td><code>isInteger</code></td>
<td>8 I</td>
<td>405</td>
</tr>
<tr>
<td>public static boolean <code>isPrime</code>(long, int)&lt;br&gt;<code>isPrime</code></td>
<td><code>isPrime</code></td>
<td>57</td>
<td>103</td>
</tr>
<tr>
<td>public static void <code>println</code>(java.lang.String)&lt;br&gt;<code>println</code></td>
<td><code>println</code></td>
<td>4 I</td>
<td>7</td>
</tr>
<tr>
<td>public static void <code>printToFile</code>(java.lang.String)&lt;br&gt;<code>printToFile</code></td>
<td><code>printToFile</code></td>
<td>3 I</td>
<td>0</td>
</tr>
<tr>
<td>private static void <code>writeFile</code>(java.lang.String, java.lang.String)&lt;br&gt;<code>writeFile</code></td>
<td><code>writeFile</code></td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>private static void <code>fatalError</code>(java.lang.String)&lt;br&gt;<code>fatalError</code></td>
<td><code>fatalError</code></td>
<td>5 I</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Fields (4)

<table>
<thead>
<tr>
<th>modifiers</th>
<th>type</th>
<th>name</th>
<th>reads (247)</th>
<th>writes (911)</th>
</tr>
</thead>
<tbody>
<tr>
<td>private</td>
<td>double</td>
<td><code>_currentPrime</code></td>
<td>103</td>
<td>509</td>
</tr>
<tr>
<td>private</td>
<td>long</td>
<td><code>_currentPPS</code></td>
<td>107</td>
<td>209</td>
</tr>
<tr>
<td>private</td>
<td>long</td>
<td><code>_maxPPS</code></td>
<td>30</td>
<td>106</td>
</tr>
<tr>
<td>private</td>
<td>long</td>
<td><code>_minPPS</code></td>
<td>7</td>
<td>80</td>
</tr>
</tbody>
</table>
Future Directions

• Pattern analysis
  – Extraction of “hot streams”

• New reduction strategies

• Integration with other tracing tools
Summary

- The problems with current tracing methods.

- A solution: STEP
  - Framework overview
  - Example: JVMPI allocation data
  - Uses of STEP data
Links to More Information

**STEP**  http://www.sable.mcgill.ca/step/

**EVolve**  http://www.sable.mcgill.ca/evolve/

**JIL (JIMPLEX)**  http://www.sable.mcgill.ca/jil/

**Sable Research**  http://www.sable.mcgill.ca