Statistical Debugging for Real-World Performance Problems

Linhai Song

Advisor: Prof. Shan Lu
Software Efficiency is Critical

• No one wants slow and inefficient software
  – Frustrate end users
  – Cause economic loss

• Software efficiency is increasingly important
  – Hardware is not getting faster (per-core)
  – Software is getting more complex
  – Energy saving is getting more urgent

Still Not Finished?
Performance Bugs

- Implementation mistakes causing inefficiency
- An example

```c
void ha_partition::start_bulk_insert(int rows) {
    .......
    - if (!rows)
    -     DBUG_VOID_RETURN;
    - rows = rows / m_tot_parts + 1;
    + rows = rows ? rows / m_tot_parts + 1 : 0;
    ....... // fast path using caches
}
```

MySQL Bug 26527

MySQL Bug DB

rows = 0 causing no cache allocated

MySQL Bug 26527

20x Slower
How to Diagnose Performance Bugs

• Difficult to avoid
  – Lack performance documentation for APIs
  – Workloads are quickly changing

• Diagnosis tools are needed

• The state of the art is *preliminary*

• Profilers

```c
void ha_partition::start_bulk_insert(int rows) {
    // fast path using caches
    .......
    - if (!rows)
    -     DBUG_VOID_RETURN;
    -     rows = rows / m_tot_parts + 1;
    +     rows = rows ? rows / m_tot_parts + 1 : 0;
    .......
    // Not in profiling results
}
MySQL Bug 26527
```
How to Diagnose Functional Bugs

• The state of the art is *mature*
  – Has been studied for decades
  – Many successful techniques have been proposed

• Statistical debugging

```c
int i = 0;
int j = 10;
int k = fopen(…);
if (p==NULL)
  printf("%s\n", p->str);
```

<table>
<thead>
<tr>
<th>Rank</th>
<th>Predicates</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B:p==NULL</td>
<td>......</td>
</tr>
<tr>
<td></td>
<td>B:p!=NULL</td>
<td>......</td>
</tr>
<tr>
<td></td>
<td>R:k &gt; 0</td>
<td>......</td>
</tr>
<tr>
<td></td>
<td>S:i&lt;j</td>
<td>......</td>
</tr>
</tbody>
</table>

Input: 
```
Bad
```
```
Good
```

Program: 
```
......
int i = 0;
int j = 10;
int k = fopen(...);
if (p==NULL)
  printf("%s\n", p->str);
......
```

Symptom: 
```
failure
```

Predicates:
- B: p==NULL
- R: k > 0
- S: i<j

Statistical Model
What Can We Learn?

• How about statistical debugging
  – **Q1**: How to identify failure runs?
  – **Q2**: How to obtain inputs?
  – **Q3**: How to design predicates?

```c
int i = 0;
int j = 10;
int k = fopen(...);
if (p==NULL)
    printf("%s\n", p->str);
```

**Input:** Q2?

**Program:**

```
......
int i = 0;
int j = 10;
int k = fopen(...);
if (p==NULL)
    printf("%s\n", p->str);
......
```

**Symptom:** failure

**Q1?**

**Q2?**

**Q3?**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Predicates</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B:p==NULL</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>...</td>
<td>......</td>
</tr>
</tbody>
</table>
Contributions

• Diagnosis process for performance bugs
  – Performance problems are noticed by comparison
  – Inputs are provided during reporting

• Statistical in-house performance diagnosis
  – 3 popular predicates
  – 2 widely used statistical models

• Statistical on-line performance diagnosis
  – Same diagnosis capability with <10% overhead
  – Not sacrifice diagnosis latency
Outline

• Overview
• Diagnosis process study
• In-house diagnosis study
• On-line diagnosis study
• Conclusion
Outline

• Overview
• Diagnosis process study
• In-house diagnosis study
• On-line diagnosis study
• Conclusion
Outline

• Overview
• Diagnosis process study
• In-house diagnosis study
• On-line diagnosis study
• Conclusion
Methodology

- Application and Bug Source

<table>
<thead>
<tr>
<th>App.</th>
<th>Software Type</th>
<th>Language</th>
<th>MLOC</th>
<th>Bug DB History</th>
<th>Tags</th>
<th># Bugs</th>
<th># Bug User Perceived</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>Command-line Utility + Server + Library</td>
<td>C/Java</td>
<td>0.45</td>
<td>13 y</td>
<td>N/A</td>
<td>25</td>
<td>16</td>
</tr>
<tr>
<td>Chrome</td>
<td>GUI Application</td>
<td>C/C++</td>
<td>14.0</td>
<td>4 y</td>
<td>N/A</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>GCC</td>
<td>Compiler</td>
<td>C/C++</td>
<td>5.7</td>
<td>13 y</td>
<td>Compile-time-hog</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Mozilla</td>
<td>GUI Application</td>
<td>C++/JS</td>
<td>4.7</td>
<td>14 y</td>
<td>perf</td>
<td>36</td>
<td>19</td>
</tr>
<tr>
<td>MySQL</td>
<td>Server Software</td>
<td>C/C++/C#</td>
<td>1.3</td>
<td>10 y</td>
<td>S5</td>
<td>28</td>
<td>17</td>
</tr>
</tbody>
</table>

Total: 110  65
Q1: How to identify failure runs?

- How about statistical debugging
  - Q1: How to identify failure runs?

[Diagram showing input, program, and symptom with statistical model and predicates table]
How Perf. Bugs are Observed

- Within one code base
- Cross multiple code bases
- Not using comparison

Dominating
How Perf. Bugs are Observed

- the same input with different *configuration*
- inputs with different *sizes*
- inputs with slightly different *functionality*
How Perf. Bugs are Observed

- same applications’ different versions
- different applications
How Perf. Bugs are Observed

- Within one code base
- Cross multiple code bases
- Not using comparison

- MySQL
- Mozilla
- GCC
- Chrome
- Apache
Q2: How to obtain inputs?

• How about statistical debugging
  – **Q1**: How to identify failure runs?
  – **Q2**: How to obtain inputs?

Program:

```
......
int i = 0;
int j = 10;
int k = fopen(...);
if (p==NULL)
    printf("%s\n", p->str);
......
```

Input: Q2?

Statistical Model

<table>
<thead>
<tr>
<th>Rank</th>
<th>Predicates</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B:p==NULL</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>
Bad Inputs Provided in Bug Reports

Cover all bugs

1/? vs n/?

- MySQL
- Mozilla
- GCC
- Chrome
- Apache

50
40
30
20
10
0
Good Inputs Provided in Bug Reports

Good inputs provided

- MySQL
- Mozilla
- GCC
- Chrome
- Apache
Implications

• Performance bugs are observed differently
  – Noticed through comparison

• Easy to tell successful runs from failure runs
  – Case 1: through comparison
  – Case 2: symptom is dramatic

• Statistical debugging is a natural fit
Outline

• Overview
• **Diagnosis process study**
• In-house diagnosis study
• On-line diagnosis study
• Conclusion
Outline

• Overview
• Diagnosis process study
• In-house diagnosis study
• On-line diagnosis study
• Conclusion
• In-house diagnosis
• Predicate design
  – Branch
    if (p) ...
    else ....
• In-house diagnosis
• Predicate design
  – Branch
  – Return

if (p) ...
else .....  
n=fprintf(...);
• In-house diagnosis

• Predicate design
  – Branch
  – Return
  – Scalar-pair

```c
int i, j, k;
...
i = ...;
```
• In-house diagnosis

• Predicate design
  – Branch
  – Return
  – Scalar-pair

• Statistical model design

```c
if (p) ...
else ....
```

```c
n=fprintf(...);
```

```c
int i, j, k;
... i = ...;
```
Design

• In-house diagnosis

• Predicate design
  – Branch
  – Return
  – Scalar-pair

• Statistical model design
  – Basic model
Design

- In-house diagnosis
- Predicate design
  - Branch
  - Return
  - Scalar-pair
- Statistical model design
  - Basic model
  - Delta-LDA

```c
if (p) ...
else ....
n=fprintf(...);
```

```
int i, j, k;
...
i = ...;
```
Experimental Methodology

• Benchmark selection
  – 8 C bugs, 8 C++ bugs and 4 Java bugs

• Input design and other setting
  – 10 failure and 10 successful runs

• Techniques under comparison
  – CBI for C programs
  – Pin for C++ programs
  – Compared with profiling results from OProfile
## Experimental Results

<table>
<thead>
<tr>
<th>BugID</th>
<th>Branch</th>
<th>Return</th>
<th>S-pair</th>
<th>Basic Model</th>
<th>ΔLDA</th>
<th>Profiler</th>
<th>Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla258793</td>
<td>64024</td>
<td>152724</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla299742</td>
<td>64089</td>
<td>150973</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla347306</td>
<td>6901</td>
<td>6729</td>
<td>30953</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>√1</td>
</tr>
<tr>
<td>Mozilla411722</td>
<td>8780</td>
<td>6889</td>
<td>34378</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>1198</td>
<td>886</td>
<td>/</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td>√1</td>
</tr>
<tr>
<td>MySQL26527</td>
<td>7443</td>
<td>7631</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
</tr>
<tr>
<td>MySQL27287</td>
<td>5377</td>
<td>5762</td>
<td>/</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td>√1</td>
</tr>
<tr>
<td>MySQL40337</td>
<td>7547</td>
<td>8161</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
</tr>
<tr>
<td>MySQL42649</td>
<td>15920</td>
<td>11800</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
</tr>
<tr>
<td>MySQL44723</td>
<td>10649</td>
<td>9130</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
</tr>
<tr>
<td>Apache3278</td>
<td>7</td>
<td>57</td>
<td>102</td>
<td>-</td>
<td>√1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Apache34464</td>
<td>17</td>
<td>23</td>
<td>203</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>√3</td>
</tr>
</tbody>
</table>

...
## Experimental Results

<table>
<thead>
<tr>
<th>BugID</th>
<th>Branch</th>
<th>Return</th>
<th>S-pair</th>
<th>Basic Model Branch</th>
<th>Return</th>
<th>S-pair</th>
<th>ΔLDA</th>
<th>Profiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla258793</td>
<td>64024</td>
<td>152724</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla299742</td>
<td>64089</td>
<td>150973</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla347306</td>
<td>6901</td>
<td>6729</td>
<td>30953</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>Mozilla411722</td>
<td>8780</td>
<td>6889</td>
<td>34378</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>1198</td>
<td>886</td>
<td>/</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>MySQL26527</td>
<td>7443</td>
<td>7631</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL27287</td>
<td>5377</td>
<td>5762</td>
<td>/</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>MySQL40337</td>
<td>7547</td>
<td>8161</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL42649</td>
<td>15920</td>
<td>11800</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL44723</td>
<td>10649</td>
<td>9130</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Apache3278</td>
<td>7</td>
<td>57</td>
<td>102</td>
<td>-</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Apache34464</td>
<td>17</td>
<td>23</td>
<td>203</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>√3</td>
<td>√5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>BugID</th>
<th>Branch</th>
<th>Return</th>
<th>S-pair</th>
<th>Candidate Predicates</th>
<th>Branch</th>
<th>Return</th>
<th>S-pair</th>
<th>Basic Model</th>
<th>ΔLDA</th>
<th>Profiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla258793</td>
<td>64024</td>
<td>152724</td>
<td>/</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla299742</td>
<td>64089</td>
<td>150973</td>
<td>/</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla347306</td>
<td>6901</td>
<td>6729</td>
<td>30953</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>Mozilla411722</td>
<td>8780</td>
<td>6889</td>
<td>34378</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>1198</td>
<td>886</td>
<td>/</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td></td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>MySQL26527</td>
<td>7443</td>
<td>7631</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL27287</td>
<td>5377</td>
<td>5762</td>
<td>/</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td></td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>MySQL40337</td>
<td>7547</td>
<td>8161</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL42649</td>
<td>15920</td>
<td>11800</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL44723</td>
<td>10649</td>
<td>9130</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Apache3278</td>
<td>7</td>
<td>57</td>
<td>102</td>
<td>-</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Apache34464</td>
<td>17</td>
<td>23</td>
<td>203</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>√3</td>
<td>√5</td>
</tr>
</tbody>
</table>

...
## Experimental Results

<table>
<thead>
<tr>
<th>BugID</th>
<th>Branch</th>
<th>Return</th>
<th>S-pair</th>
<th>Branch</th>
<th>Return</th>
<th>S-pair</th>
<th>ΔLDA</th>
<th>Profiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla258793</td>
<td>64024</td>
<td>152724</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla299742</td>
<td>64089</td>
<td>150973</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mozilla347306</td>
<td>6901</td>
<td>6729</td>
<td>30953</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>Mozilla411722</td>
<td>8780</td>
<td>6889</td>
<td>34378</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>1198</td>
<td>886</td>
<td>/</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>MySQL26527</td>
<td>7443</td>
<td>7631</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL27287</td>
<td>5377</td>
<td>5762</td>
<td>/</td>
<td>-</td>
<td>-</td>
<td>/</td>
<td>√1</td>
<td>√1</td>
</tr>
<tr>
<td>MySQL40337</td>
<td>7547</td>
<td>8161</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL42649</td>
<td>15920</td>
<td>11800</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MySQL44723</td>
<td>10649</td>
<td>9130</td>
<td>/</td>
<td>√1</td>
<td>-</td>
<td>/</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Apache3278</td>
<td>7</td>
<td>57</td>
<td>102</td>
<td>-</td>
<td>√1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Apache34464</td>
<td>17</td>
<td>23</td>
<td>203</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>√3</td>
<td>√5</td>
</tr>
</tbody>
</table>

...
Outline

• Overview
• Diagnosis process study
• In-house diagnosis study
• On-line diagnosis study
• Conclusion
Outline

• Overview
• Diagnosis process study
• In-house diagnosis study
• On-line diagnosis study
• Conclusion
Experimental Methodology

• Challenges in on-line diagnosis
  – Diagnosis capability
  – Low overhead

• Benchmarks and inputs

• Tool implementation
  – CBI in sampling mode for return predicates
  – LBR for branch predicates
  – Rough sampling rate is 1/100
## Experimental Results

<table>
<thead>
<tr>
<th>BugID</th>
<th>Diagnosis Capability</th>
<th>Overhead</th>
<th>Requested Failure Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla258793</td>
<td>√1</td>
<td>1.81%</td>
<td>1000</td>
</tr>
<tr>
<td>Mozilla299742</td>
<td>√1</td>
<td>7.52%</td>
<td>1000</td>
</tr>
<tr>
<td>Mozilla347306</td>
<td>√1</td>
<td>3.01%</td>
<td>10</td>
</tr>
<tr>
<td>Mozilla411722</td>
<td>√1</td>
<td>3.35%</td>
<td>1000</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>√1</td>
<td>8.58%</td>
<td>10</td>
</tr>
<tr>
<td>MySQL26527</td>
<td>√1</td>
<td>7.06%</td>
<td>1000</td>
</tr>
<tr>
<td>MySQL27287</td>
<td>√1</td>
<td>2.62%</td>
<td>10</td>
</tr>
<tr>
<td>MySQL40337</td>
<td>√1</td>
<td>3.32%</td>
<td>1000</td>
</tr>
<tr>
<td>MySQL42649</td>
<td>√1</td>
<td>4.67%</td>
<td>1000</td>
</tr>
<tr>
<td>MySQL44723</td>
<td>√1</td>
<td>0.40%</td>
<td>1000</td>
</tr>
<tr>
<td>Apache3278</td>
<td>√1</td>
<td>3.22%</td>
<td>1000</td>
</tr>
<tr>
<td>Apache34464</td>
<td>√1</td>
<td>2.13%</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>BugID</th>
<th>Diagnosis Capability</th>
<th>Overhead</th>
<th>Requested Failure Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla258793</td>
<td>v1</td>
<td>1.81%</td>
<td>1000</td>
</tr>
<tr>
<td>Mozilla299742</td>
<td>v1</td>
<td>7.52%</td>
<td>1000</td>
</tr>
<tr>
<td>Mozilla347306</td>
<td>v1</td>
<td>3.01%</td>
<td>10</td>
</tr>
<tr>
<td>Mozilla411722</td>
<td>v1</td>
<td>3.35%</td>
<td>1000</td>
</tr>
<tr>
<td>MySQL15811</td>
<td>v1</td>
<td>8.58%</td>
<td>10</td>
</tr>
<tr>
<td>MySQL26527</td>
<td>v1</td>
<td>7.06%</td>
<td>1000</td>
</tr>
<tr>
<td>MySQL27287</td>
<td>v1</td>
<td>2.62%</td>
<td>10</td>
</tr>
<tr>
<td>MySQL40337</td>
<td>v1</td>
<td>3.32%</td>
<td>1000</td>
</tr>
<tr>
<td>MySQL42649</td>
<td>v1</td>
<td>4.67%</td>
<td>1000</td>
</tr>
<tr>
<td>MySQL44723</td>
<td>v1</td>
<td>0.40%</td>
<td>1000</td>
</tr>
<tr>
<td>Apache3278</td>
<td>v1</td>
<td>3.22%</td>
<td>1000</td>
</tr>
<tr>
<td>Apache34464</td>
<td>v1</td>
<td>2.13%</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Conclusion and Future Works

• Study diagnosis process for perf. bugs
  – Noticed through comparison
  – Good and bad inputs are provided
• Study statistical debugging on perf. bugs
  – Branch predicates + two statistical models
• Future works
  – Analyze inefficient loops
  – Provide detailed fix strategies
Thanks a lot!