Statistical Debugging for Real-World Performance Problems

Linhai Song

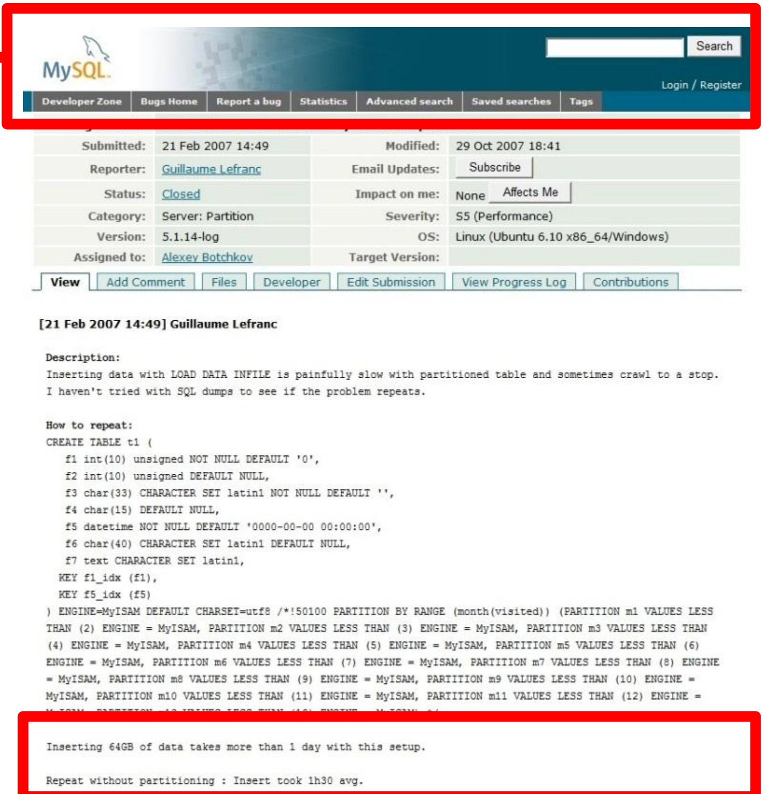
Advisor: Prof. Shan Lu

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

Performance Bugs

- Implementation mistakes causing inefficiency
- An example



MySQL **Bug DB**

rows=***0*** causing
no cache allocated

```
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

20 ***X*** 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

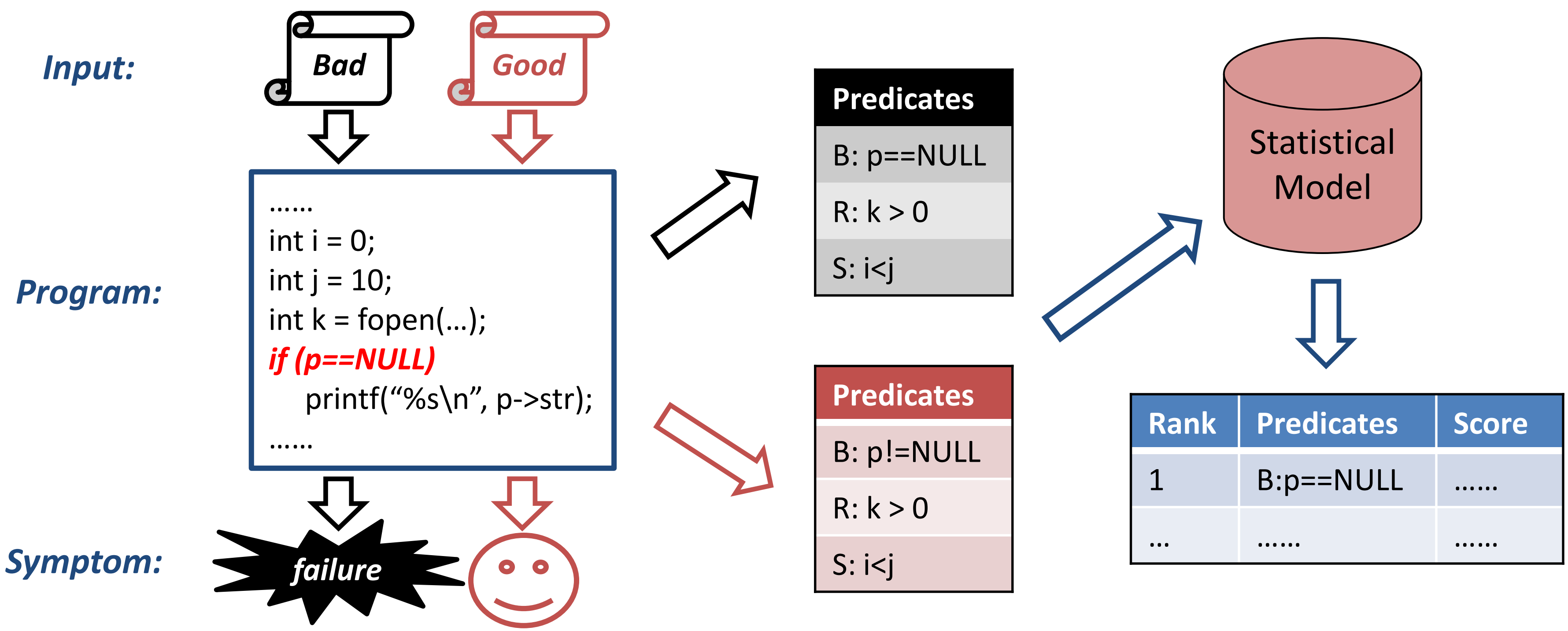
```
void ha_partition::start_bulk_insert(int rows) {
       .......
-    if (!rows)
-       DBUG_VOID_RETURN;
-    rows= rows/m_tot_parts + 1;
+    rows= rows ? rows/m_tot_parts + 1 : 0;
      ....... // fast path using caches
}
```

Not in profiling results

MySQL Bug 26527

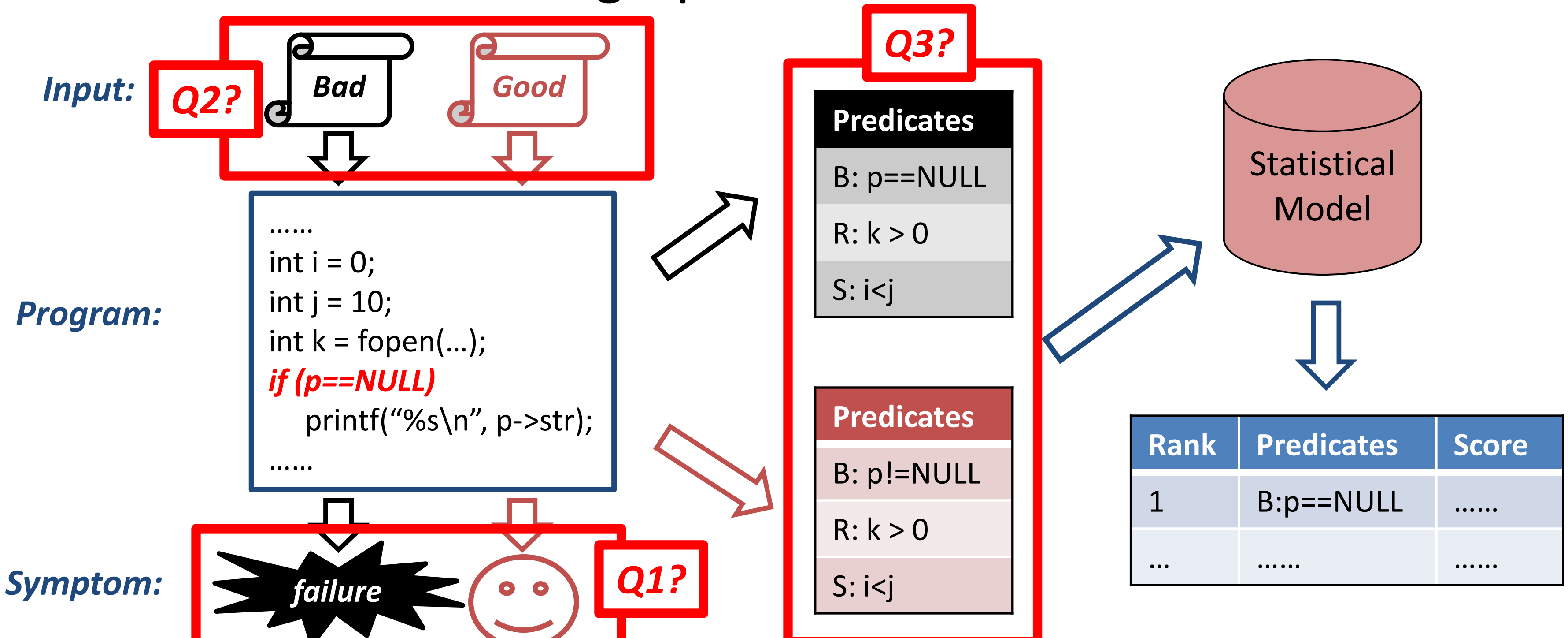
How to Diagnose Functional Bugs

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



What Can We Learn?

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



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

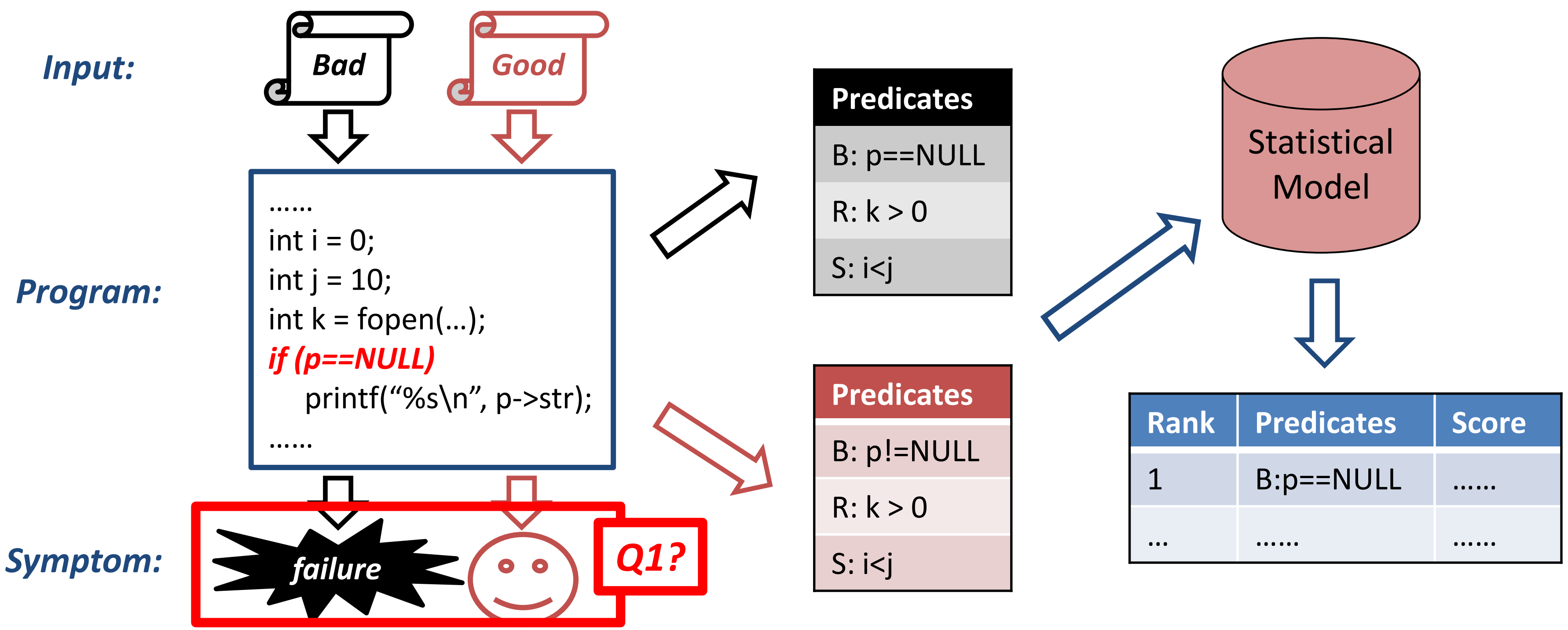
- Application and Bug Source

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

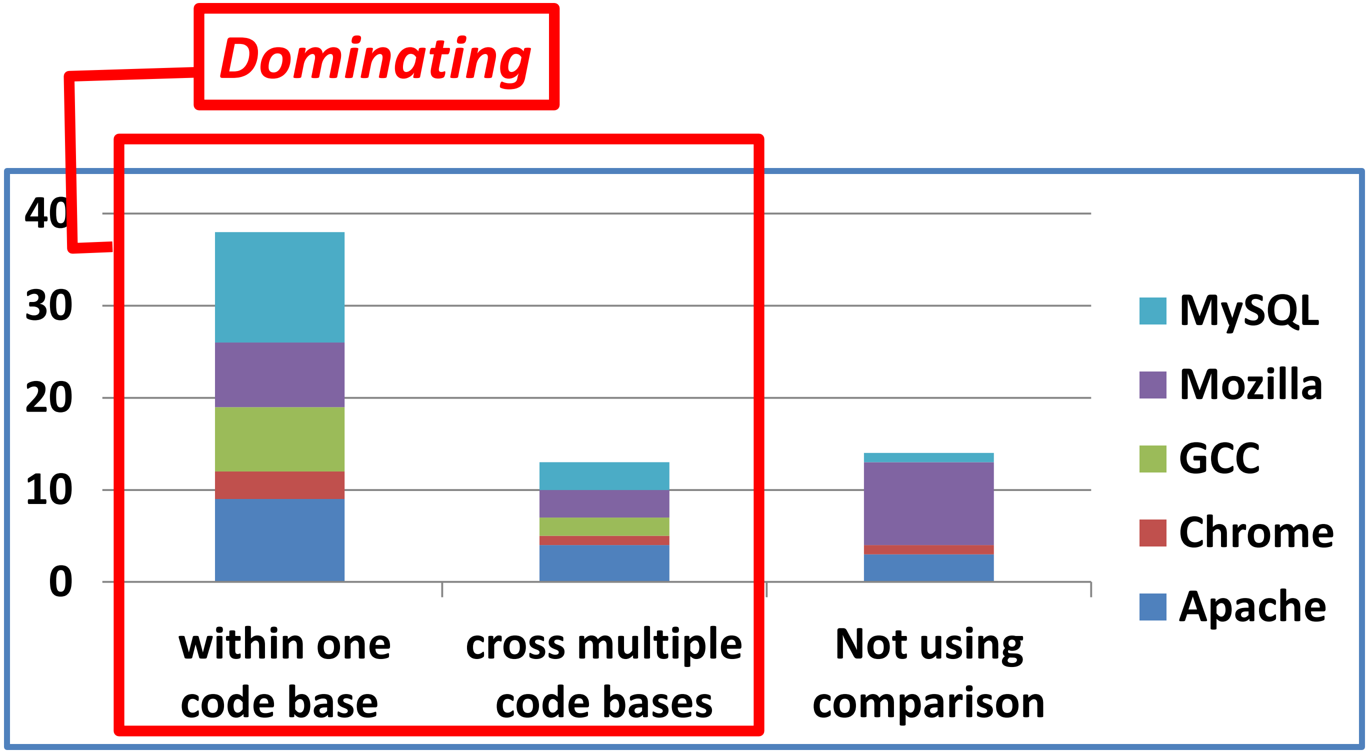
Total: 110 65

Q1: How to identify failure runs?

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

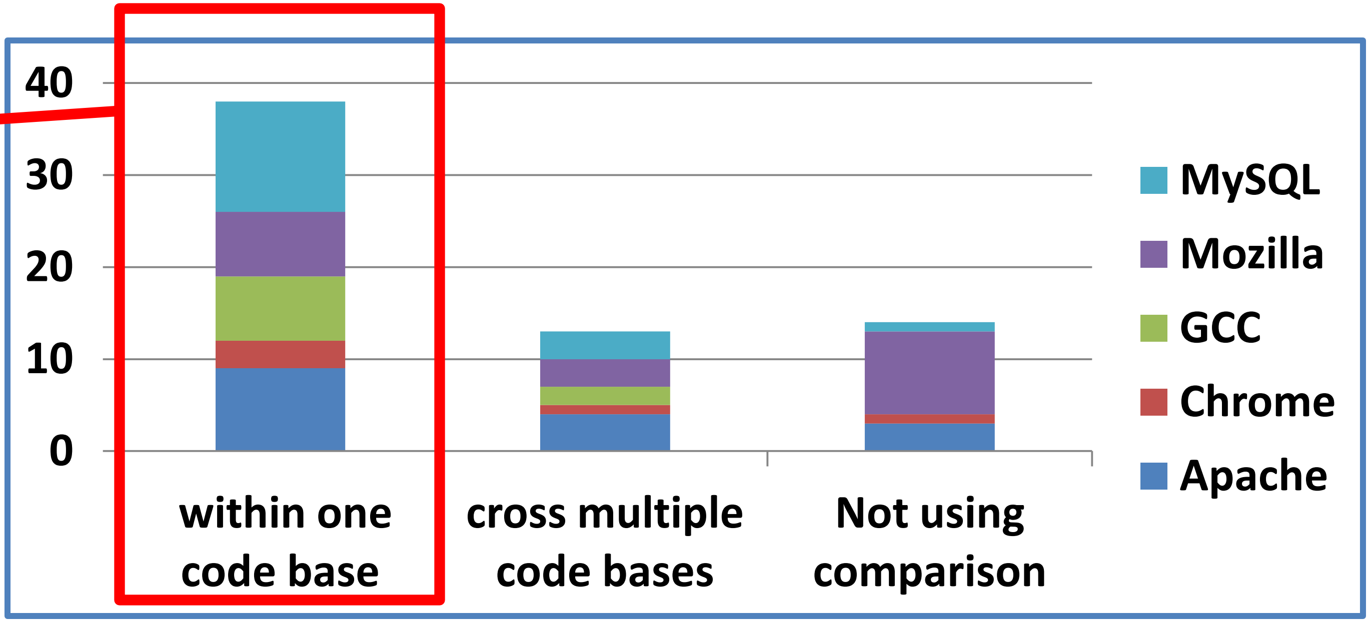


How Perf. Bugs are Observed



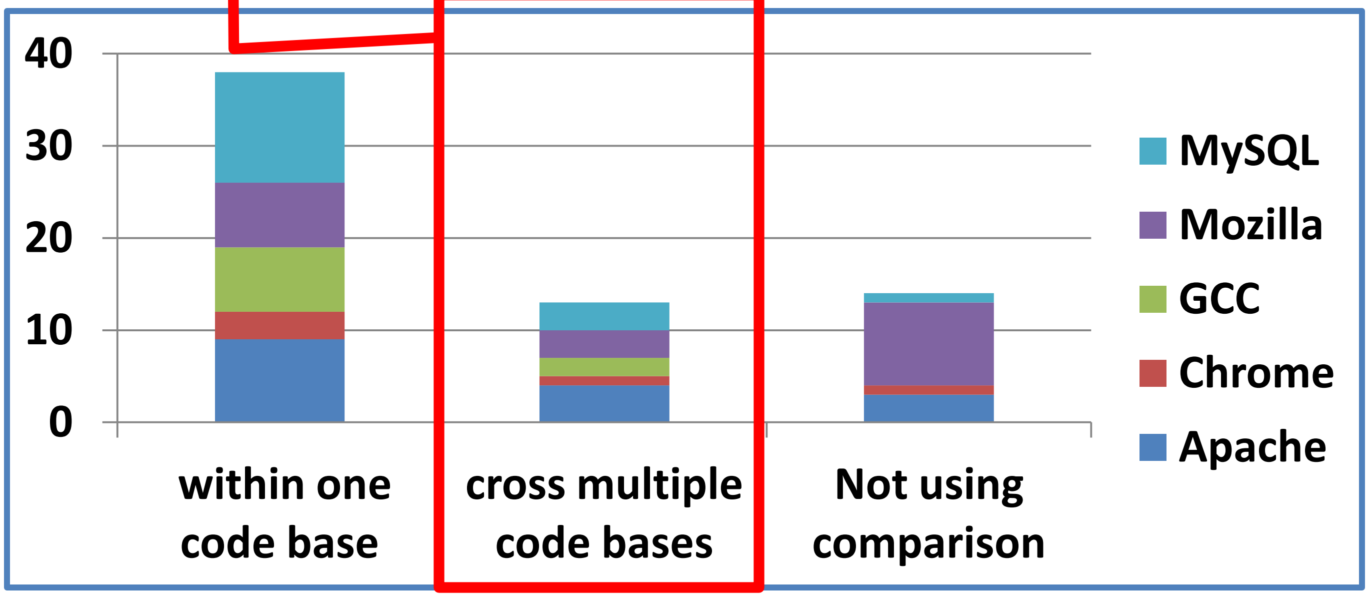
How Perf. Bugs are Observed

- the same input with different ***configuration***
- inputs with different ***sizes***
- inputs with slightly different ***functionality***

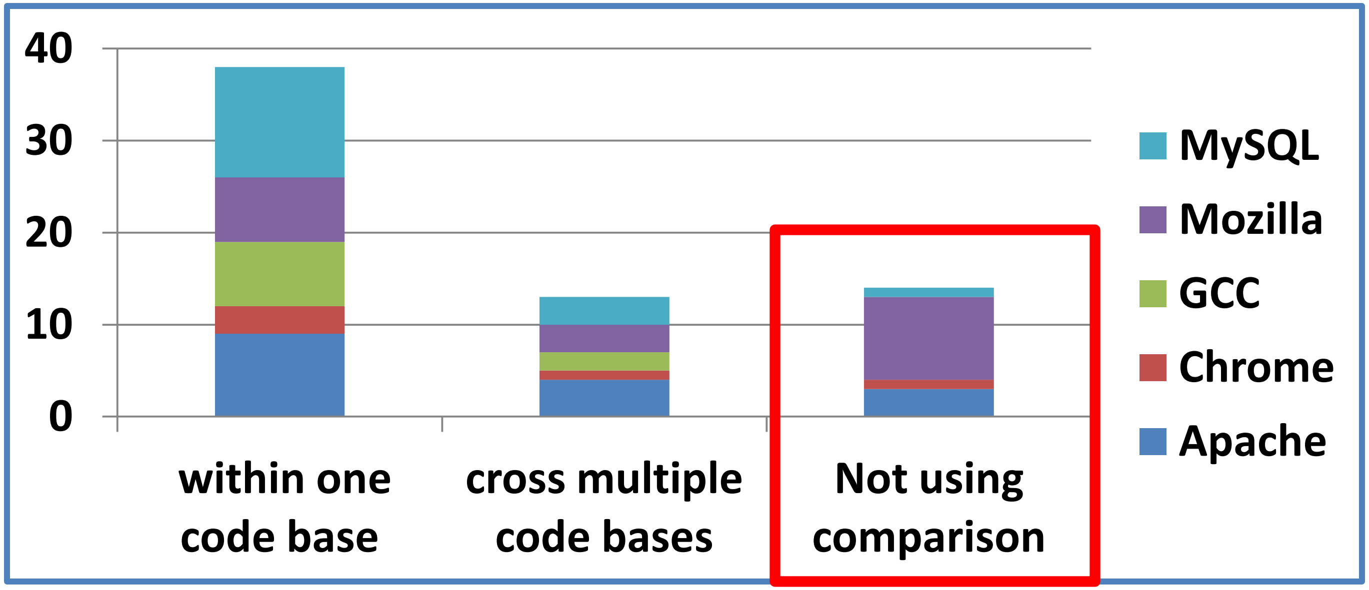


How Perf. Bugs are Observed

- same applications' different ***versions***
- different ***applications***

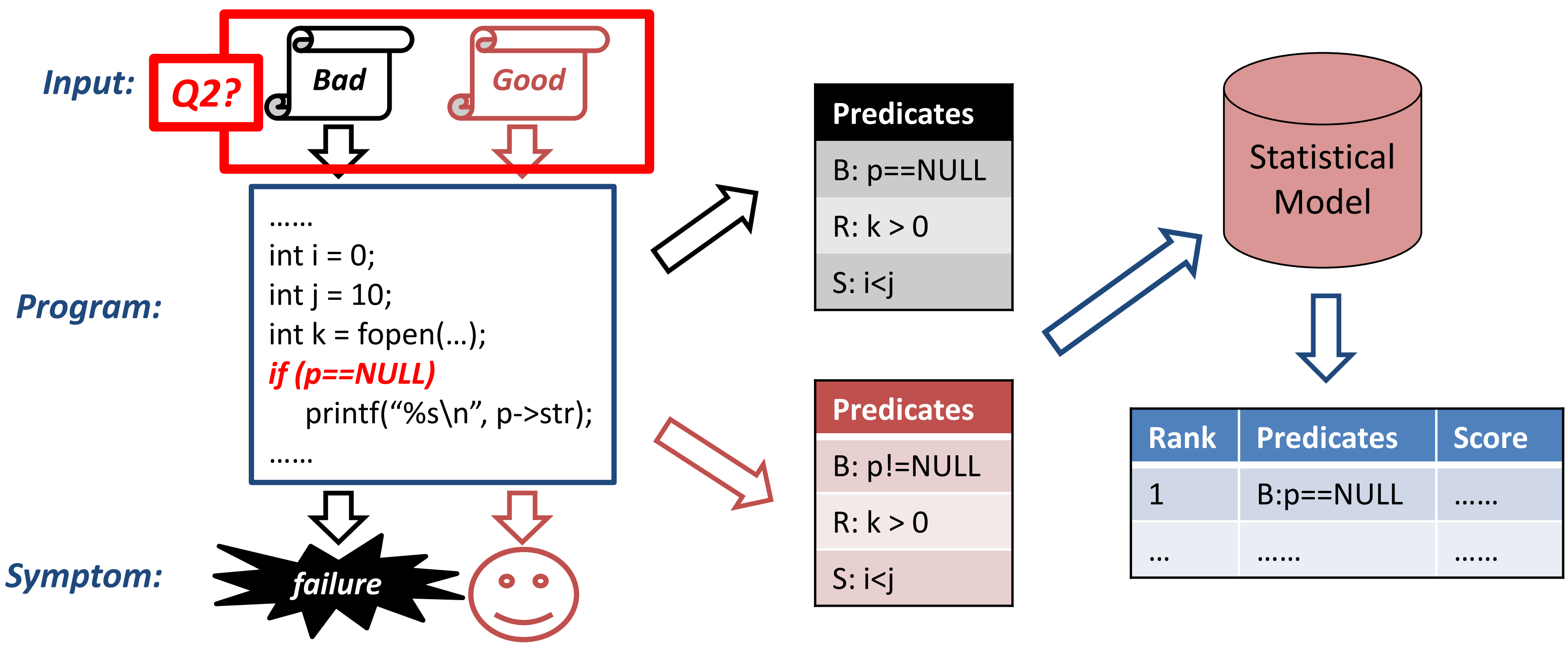


How Perf. Bugs are Observed

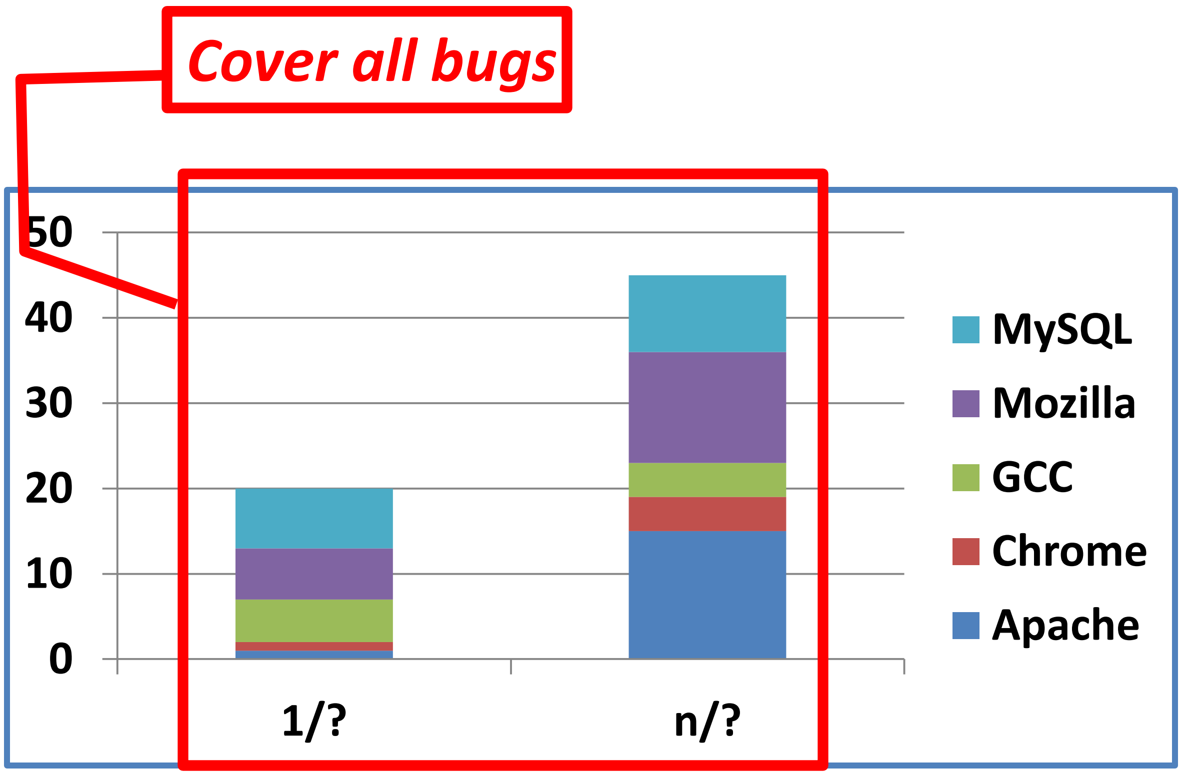


Q2: How to obtain inputs?

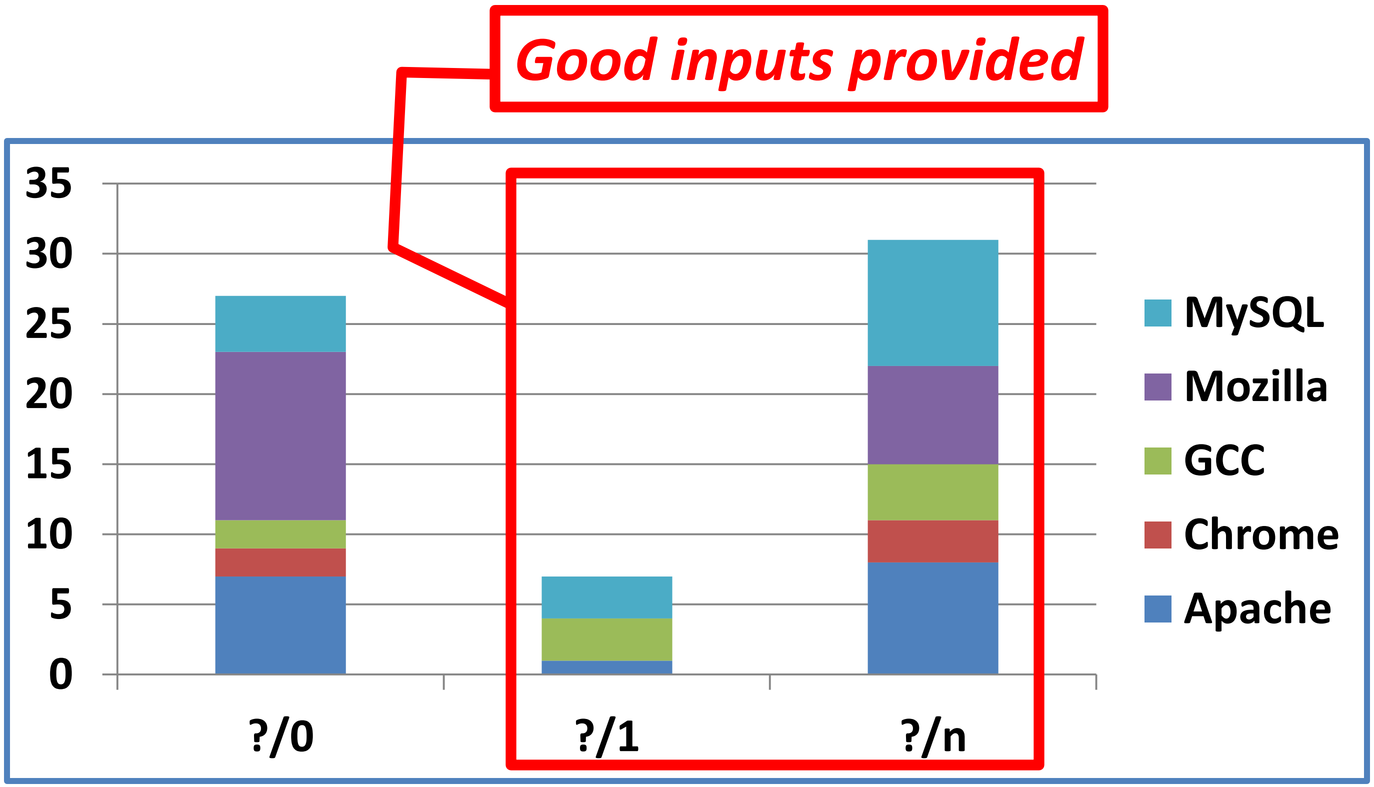
- How about statistical debugging
  - ***Q1:*** How to identify failure runs?
  - ***Q2:*** How to obtain inputs?



Bad Inputs Provided in Bug Reports



Good Inputs Provided in Bug Reports



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

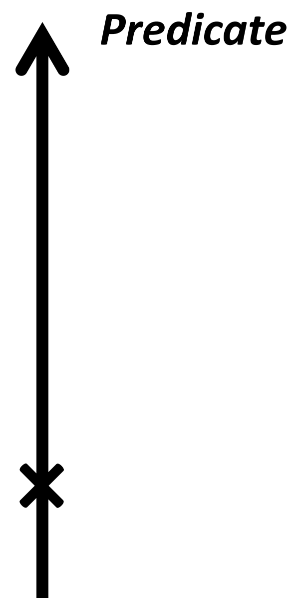
Outline

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

Design

- In-house diagnosis
- Predicate design
  - Branch

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

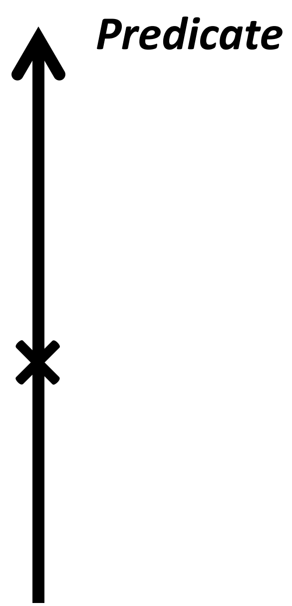


Design

- In-house diagnosis
- Predicate design
  - Branch
  - Return

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

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



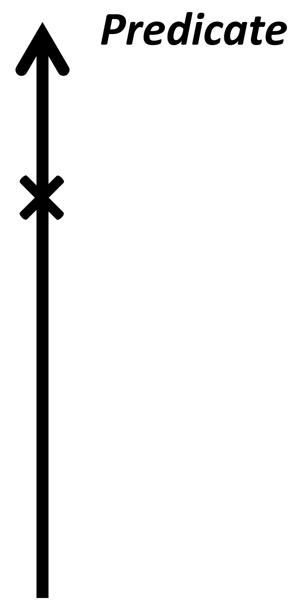
Design

- In-house diagnosis
- Predicate design
  - Branch
  - Return
  - Scalar-pair

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

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

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



Design

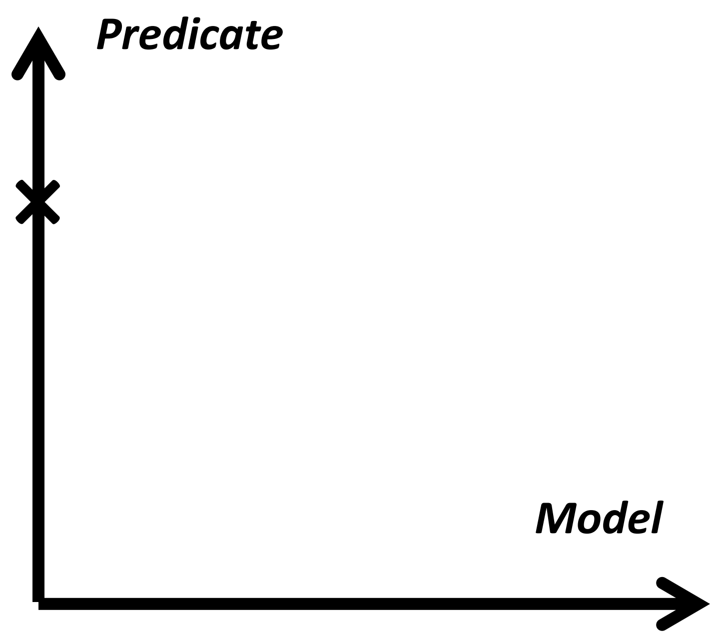
- In-house diagnosis
- Predicate design
  - Branch
  - Return
  - Scalar-pair

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

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

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

- Statistical model design



Design

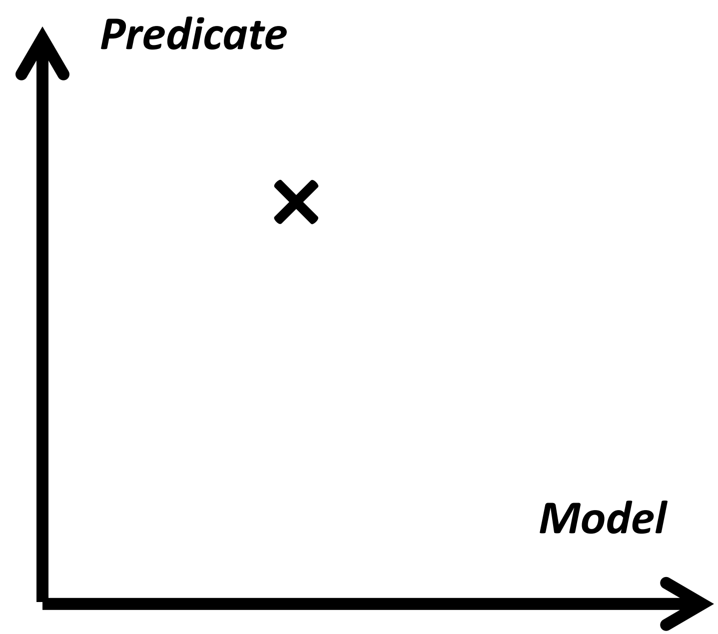
- In-house diagnosis
- Predicate design
  - Branch
  - Return
  - Scalar-pair

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

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

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

- Statistical model design
  - Basic model



Design

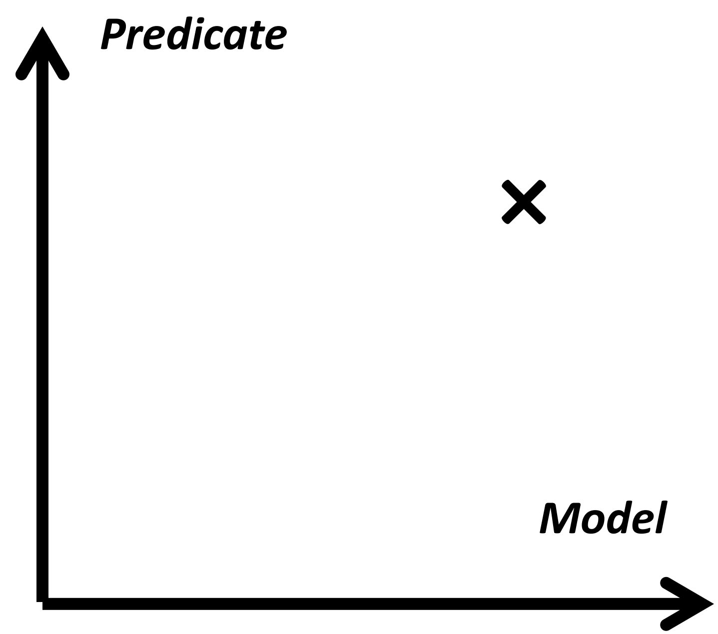
- In-house diagnosis
- Predicate design
  - Branch
  - Return
  - Scalar-pair

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

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

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

- Statistical model design
  - Basic model
  - Delta-LDA



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

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

Experimental Results

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

Experimental Results

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

Experimental Results

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

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

BugID	Diagnosis Capability	Overhead	Requested Failure Runs
Mozilla258793	√1	1.81%	1000
Mozilla299742	√1	7.52%	1000
Mozilla347306	√1	3.01%	**10**
Mozilla411722	√1	3.35%	1000
MySQL15811	√1	8.58%	**10**
MySQL26527	√1	7.06%	1000
MySQL27287	√1	2.62%	**10**
MySQL40337	√1	3.32%	1000
MySQL42649	√1	4.67%	1000
MySQL44723	√1	0.40%	1000
Apache3278	√1	3.22%	1000
Apache34464	√1	2.13%	**10**
…	…	…	…

Experimental Results

BugID	Diagnosis Capability	Overhead	Requested Failure Runs
Mozilla258793	√1	1.81%	1000
Mozilla299742	√1	7.52%	1000
Mozilla347306	√1	3.01%	**10**
Mozilla411722	√1	3.35%	1000
MySQL15811	√1	8.58%	**10**
MySQL26527	√1	7.06%	1000
MySQL27287	√1	2.62%	**10**
MySQL40337	√1	3.32%	1000
MySQL42649	√1	4.67%	1000
MySQL44723	√1	0.40%	1000
Apache3278	√1	3.22%	1000
Apache34464	√1	2.13%	**10**
…	…	…	…

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!