

# 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

rows=0 causing  
no cache allocated

MySQL Bug DB

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

*MySQL Bug 26527*

The screenshot shows the MySQL Bug DB interface. At the top, there's a search bar and navigation links like 'Developer Zone', 'Bugs Home', 'Report a bug', etc. The main content area displays details for a bug submitted on 21 Feb 2007 14:49 by Guillaume Lefranc. The bug is categorized as 'Server: Partition' with a severity of 'S5 (Performance)'. The description and how to repeat the bug are visible.

[21 Feb 2007 14:49] Guillaume Lefranc

Description:  
Inserting data with LOAD DATA INFILE is painfully slow with partitioned table and sometimes crawl to a stop. I haven't tried with SQL dumps to see if the problem repeats.

How to repeat:  
CREATE TABLE t1 (  
 f1 int(10) unsigned NOT NULL DEFAULT '0',  
 f2 int(10) unsigned DEFAULT NULL,  
 f3 char(33) CHARACTER SET latin1 NOT NULL DEFAULT '',  
 f4 char(15) DEFAULT NULL,  
 f5 datetime NOT NULL DEFAULT '0000-00-00 00:00:00',  
 f6 char(40) CHARACTER SET latin1 DEFAULT NULL,  
 f7 text CHARACTER SET latin1,  
 KEY f1\_idx (f1),  
 KEY f5\_idx (f5)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 /\*150100 PARTITION BY RANGE (month(visited)) (PARTITION m1 VALUES LESS THAN (2) ENGINE = MyISAM, PARTITION m2 VALUES LESS THAN (3) ENGINE = MyISAM, PARTITION m3 VALUES LESS THAN (4) ENGINE = MyISAM, PARTITION m4 VALUES LESS THAN (5) ENGINE = MyISAM, PARTITION m5 VALUES LESS THAN (6) ENGINE = MyISAM, PARTITION m6 VALUES LESS THAN (7) ENGINE = MyISAM, PARTITION m7 VALUES LESS THAN (8) ENGINE = MyISAM, PARTITION m8 VALUES LESS THAN (9) ENGINE = MyISAM, PARTITION m9 VALUES LESS THAN (10) ENGINE = MyISAM, PARTITION m10 VALUES LESS THAN (11) ENGINE = MyISAM, PARTITION m11 VALUES LESS THAN (12) ENGINE = MyISAM, PARTITION m12 VALUES LESS THAN (13) ENGINE = MyISAM, PARTITION m13 VALUES LESS THAN (14) ENGINE = MyISAM, PARTITION m14 VALUES LESS THAN (15) ENGINE = MyISAM, PARTITION m15 VALUES LESS THAN (16) ENGINE = MyISAM, PARTITION m16 VALUES LESS THAN (17) ENGINE = MyISAM, PARTITION m17 VALUES LESS THAN (18) ENGINE = MyISAM, PARTITION m18 VALUES LESS THAN (19) ENGINE = MyISAM, PARTITION m19 VALUES LESS THAN (20) ENGINE = MyISAM, PARTITION m20 VALUES LESS THAN (21) ENGINE = MyISAM, PARTITION m21 VALUES LESS THAN (22) ENGINE = MyISAM, PARTITION m22 VALUES LESS THAN (23) ENGINE = MyISAM, PARTITION m23 VALUES LESS THAN (24) ENGINE = MyISAM, PARTITION m24 VALUES LESS THAN (25) ENGINE = MyISAM, PARTITION m25 VALUES LESS THAN (26) ENGINE = MyISAM, PARTITION m26 VALUES LESS THAN (27) ENGINE = MyISAM, PARTITION m27 VALUES LESS THAN (28) ENGINE = MyISAM, PARTITION m28 VALUES LESS THAN (29) ENGINE = MyISAM, PARTITION m29 VALUES LESS THAN (30) ENGINE = MyISAM, PARTITION m30 VALUES LESS THAN (31) ENGINE = MyISAM, PARTITION m31 VALUES LESS THAN (32) ENGINE = MyISAM, PARTITION m32 VALUES LESS THAN (33) ENGINE = MyISAM, PARTITION m33 VALUES LESS THAN (34) ENGINE = MyISAM, PARTITION m34 VALUES LESS THAN (35) ENGINE = MyISAM, PARTITION m35 VALUES LESS THAN (36) ENGINE = MyISAM, PARTITION m36 VALUES LESS THAN (37) ENGINE = MyISAM, PARTITION m37 VALUES LESS THAN (38) ENGINE = MyISAM, PARTITION m38 VALUES LESS THAN (39) ENGINE = MyISAM, PARTITION m39 VALUES LESS THAN (40) ENGINE = MyISAM, PARTITION m40 VALUES LESS THAN (41) ENGINE = MyISAM, PARTITION m41 VALUES LESS THAN (42) ENGINE = MyISAM, PARTITION m42 VALUES LESS THAN (43) ENGINE = MyISAM, PARTITION m43 VALUES LESS THAN (44) ENGINE = MyISAM, PARTITION m44 VALUES LESS THAN (45) ENGINE = MyISAM, PARTITION m45 VALUES LESS THAN (46) ENGINE = MyISAM, PARTITION m46 VALUES LESS THAN (47) ENGINE = MyISAM, PARTITION m47 VALUES LESS THAN (48) ENGINE = MyISAM, PARTITION m48 VALUES LESS THAN (49) ENGINE = MyISAM, PARTITION m49 VALUES LESS THAN (50) ENGINE = MyISAM, PARTITION m50 VALUES LESS THAN (51) ENGINE = MyISAM, PARTITION m51 VALUES LESS THAN (52) ENGINE = MyISAM, PARTITION m52 VALUES LESS THAN (53) ENGINE = MyISAM, PARTITION m53 VALUES LESS THAN (54) ENGINE = MyISAM, PARTITION m54 VALUES LESS THAN (55) ENGINE = MyISAM, PARTITION m55 VALUES LESS THAN (56) ENGINE = MyISAM, PARTITION m56 VALUES LESS THAN (57) ENGINE = MyISAM, PARTITION m57 VALUES LESS THAN (58) ENGINE = MyISAM, PARTITION m58 VALUES LESS THAN (59) ENGINE = MyISAM, PARTITION m59 VALUES LESS THAN (60) ENGINE = MyISAM, PARTITION m60 VALUES LESS THAN (61) ENGINE = MyISAM, PARTITION m61 VALUES LESS THAN (62) ENGINE = MyISAM, PARTITION m62 VALUES LESS THAN (63) ENGINE = MyISAM, PARTITION m63 VALUES LESS THAN (64) ENGINE = MyISAM, PARTITION m64 VALUES LESS THAN (65) ENGINE = MyISAM, PARTITION m65 VALUES LESS THAN (66) ENGINE = MyISAM, PARTITION m66 VALUES LESS THAN (67) ENGINE = MyISAM, PARTITION m67 VALUES LESS THAN (68) ENGINE = MyISAM, PARTITION m68 VALUES LESS THAN (69) ENGINE = MyISAM, PARTITION m69 VALUES LESS THAN (70) ENGINE = MyISAM, PARTITION m70 VALUES LESS THAN (71) ENGINE = MyISAM, PARTITION m71 VALUES LESS THAN (72) ENGINE = MyISAM, PARTITION m72 VALUES LESS THAN (73) ENGINE = MyISAM, PARTITION m73 VALUES LESS THAN (74) ENGINE = MyISAM, PARTITION m74 VALUES LESS THAN (75) ENGINE = MyISAM, PARTITION m75 VALUES LESS THAN (76) ENGINE = MyISAM, PARTITION m76 VALUES LESS THAN (77) ENGINE = MyISAM, PARTITION m77 VALUES LESS THAN (78) ENGINE = MyISAM, PARTITION m78 VALUES LESS THAN (79) ENGINE = MyISAM, PARTITION m79 VALUES LESS THAN (80) ENGINE = MyISAM, PARTITION m80 VALUES LESS THAN (81) ENGINE = MyISAM, PARTITION m81 VALUES LESS THAN (82) ENGINE = MyISAM, PARTITION m82 VALUES LESS THAN (83) ENGINE = MyISAM, PARTITION m83 VALUES LESS THAN (84) ENGINE = MyISAM, PARTITION m84 VALUES LESS THAN (85) ENGINE = MyISAM, PARTITION m85 VALUES LESS THAN (86) ENGINE = MyISAM, PARTITION m86 VALUES LESS THAN (87) ENGINE = MyISAM, PARTITION m87 VALUES LESS THAN (88) ENGINE = MyISAM, PARTITION m88 VALUES LESS THAN (89) ENGINE = MyISAM, PARTITION m89 VALUES LESS THAN (90) ENGINE = MyISAM, PARTITION m90 VALUES LESS THAN (91) ENGINE = MyISAM, PARTITION m91 VALUES LESS THAN (92) ENGINE = MyISAM, PARTITION m92 VALUES LESS THAN (93) ENGINE = MyISAM, PARTITION m93 VALUES LESS THAN (94) ENGINE = MyISAM, PARTITION m94 VALUES LESS THAN (95) ENGINE = MyISAM, PARTITION m95 VALUES LESS THAN (96) ENGINE = MyISAM, PARTITION m96 VALUES LESS THAN (97) ENGINE = MyISAM, PARTITION m97 VALUES LESS THAN (98) ENGINE = MyISAM, PARTITION m98 VALUES LESS THAN (99) ENGINE = MyISAM, PARTITION m99 VALUES LESS THAN (100) ENGINE = MyISAM);

Inserting 64GB of data takes more than 1 day with this setup.

Repeat without partitioning : Insert took 1h30 avg.

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

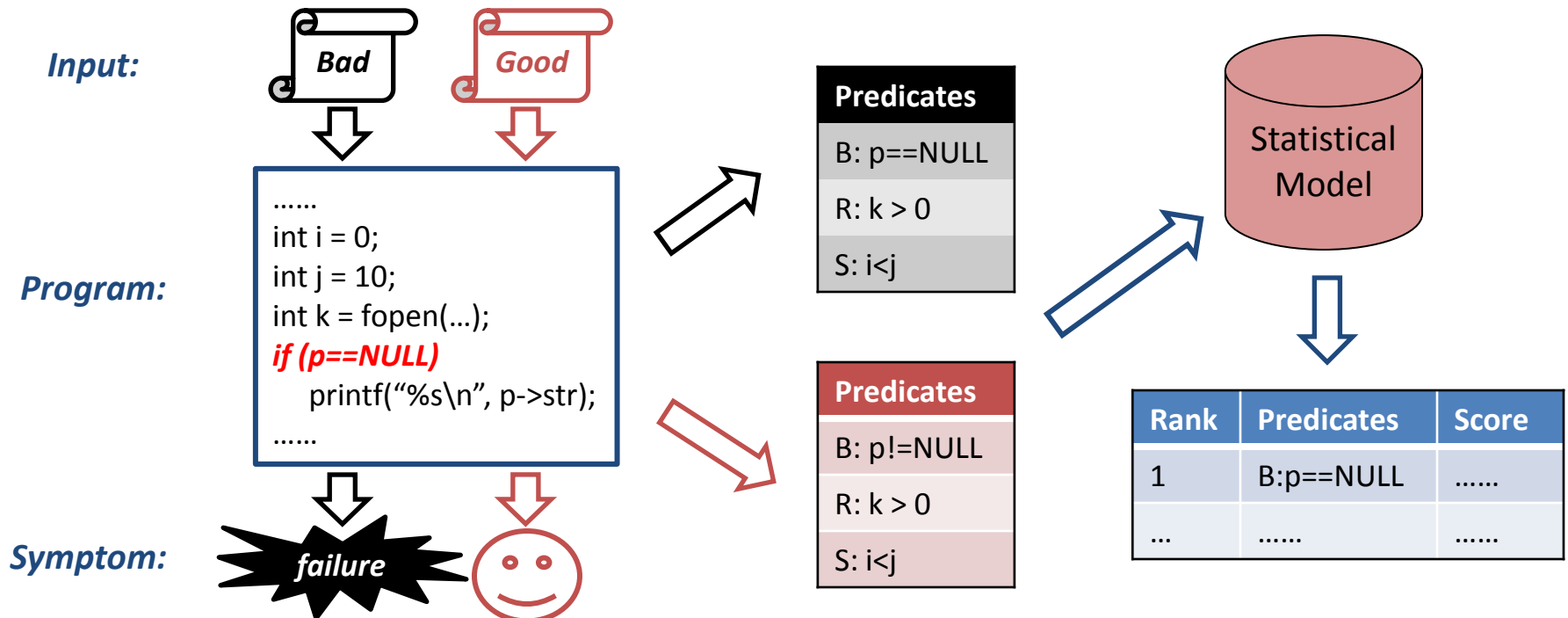
```
void ha_partition::start_bulk_insert(int rows) {  
    .....  
-   if (!rows)  
-       DEBUG_VOID_RETURN;  
-   rows= rows/m_tot_parts + 1;  
+   rows= rows ? rows/m_tot_parts + 1 : 0;  
    ..... // fast path using caches  
}
```

**Not in profiling results**

*MySQL Bug 26527*

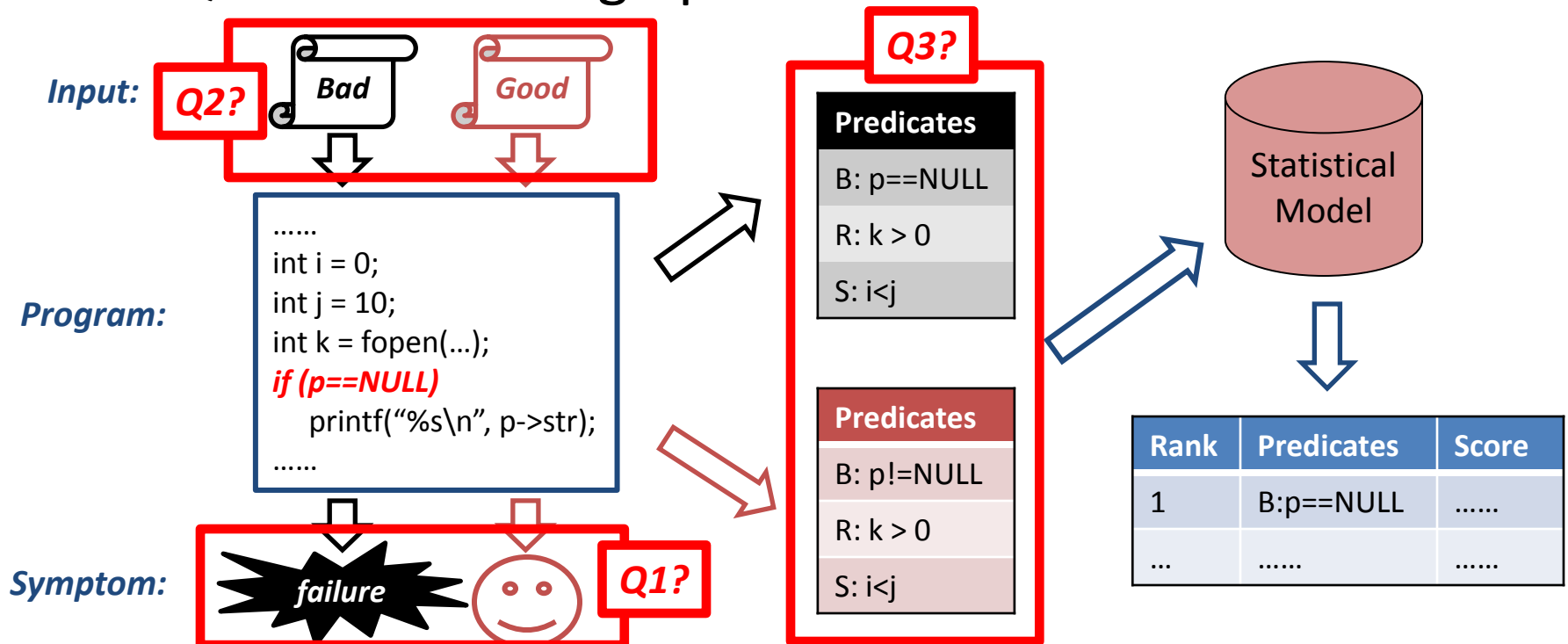
# How to Diagnose Functional Bugs

- 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

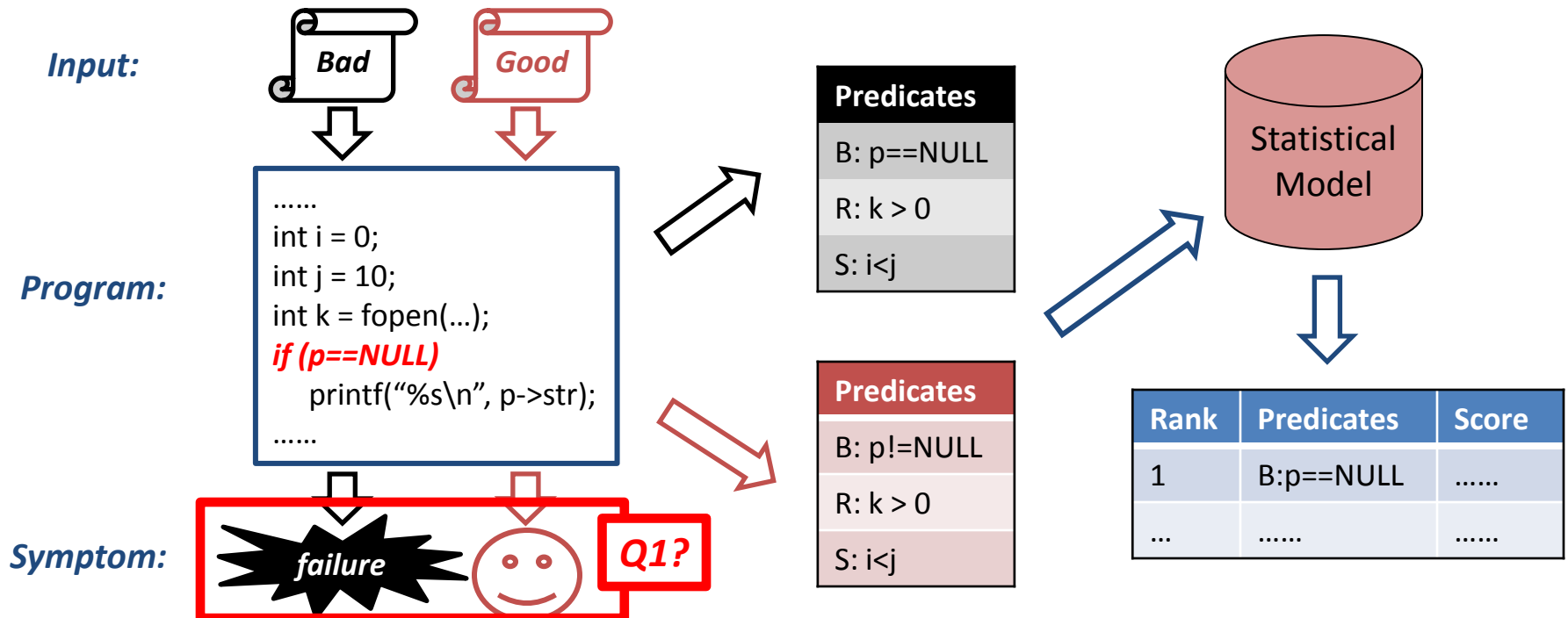
App.	Software Type	Language	MLOC	Bug DB History	Tags	# Bugs	# Bug User Perceived
<b>Apache</b>	Command-line Utility + Server + Library	C/Java	0.45	13 y	N/A	25	16
<b>Chrome</b>	GUI Application	C/C++	14.0	4 y	N/A	10	5
<b>GCC</b>	Compiler	C/C++	5.7	13 y	Compile-time-hog	11	9
<b>Mozilla</b>	GUI Application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

Total: 110

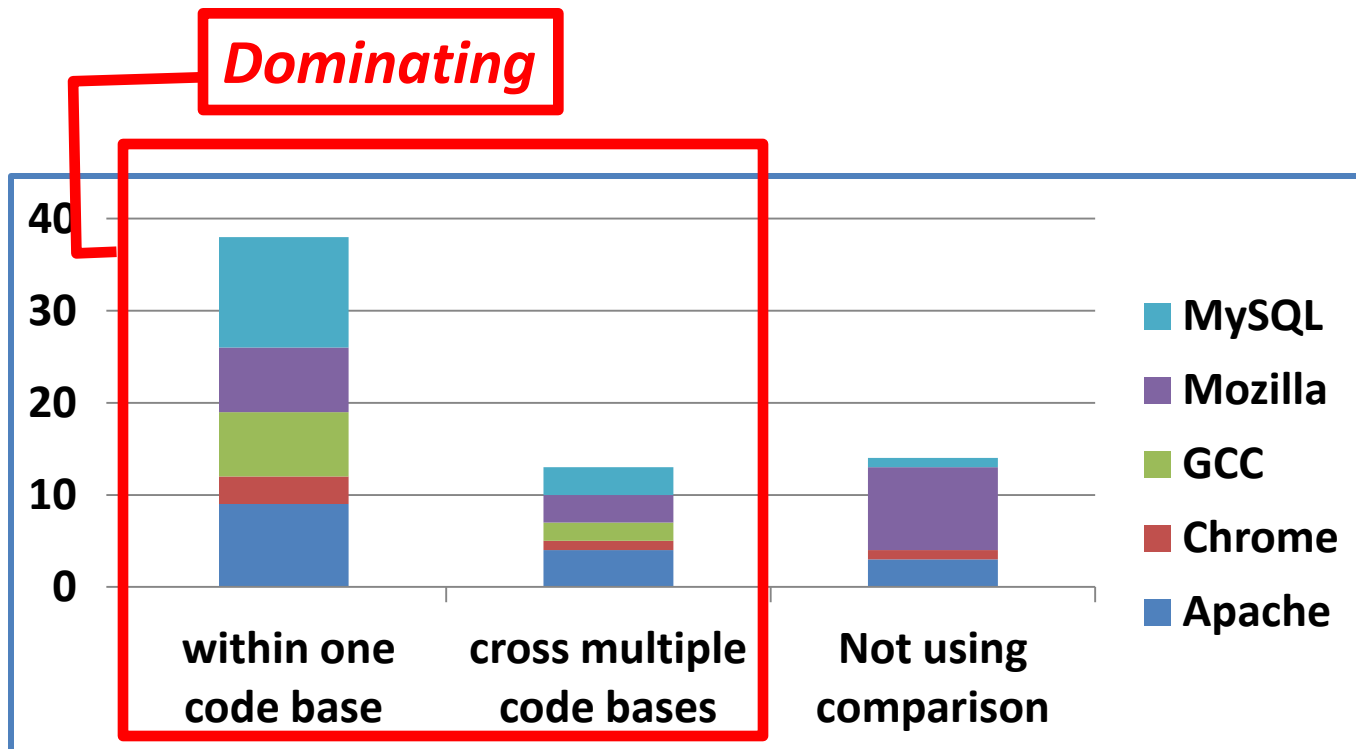
65

# Q1: How to identify failure runs?

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

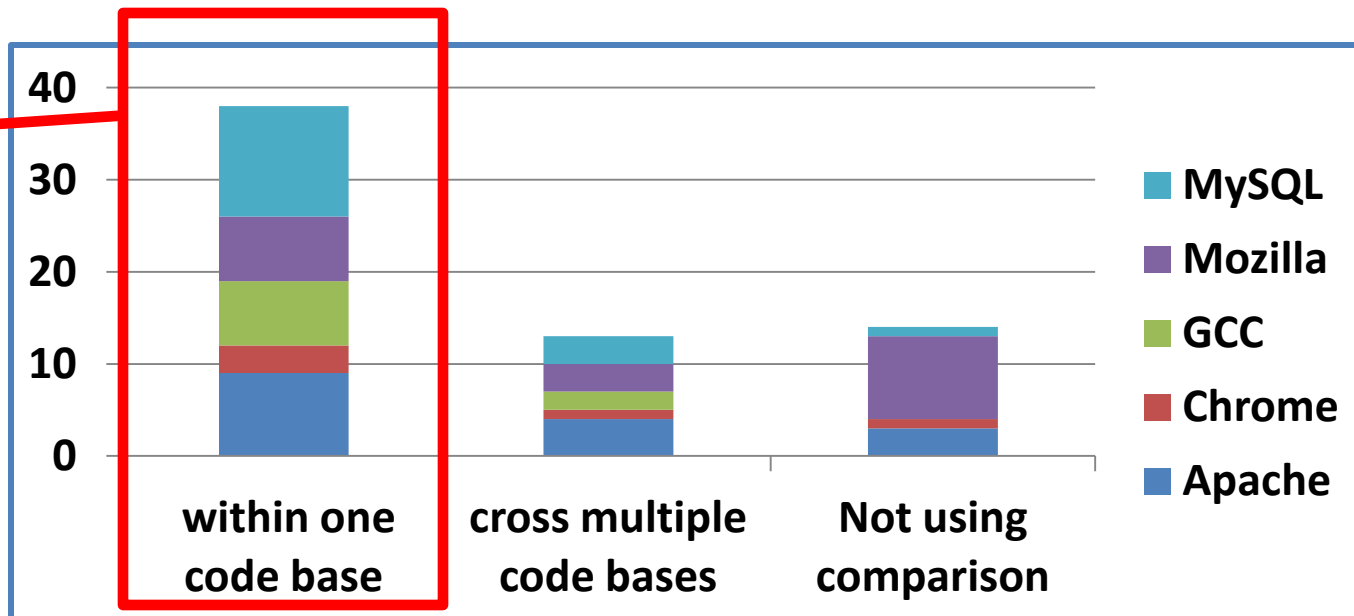


# How Perf. Bugs are Observed



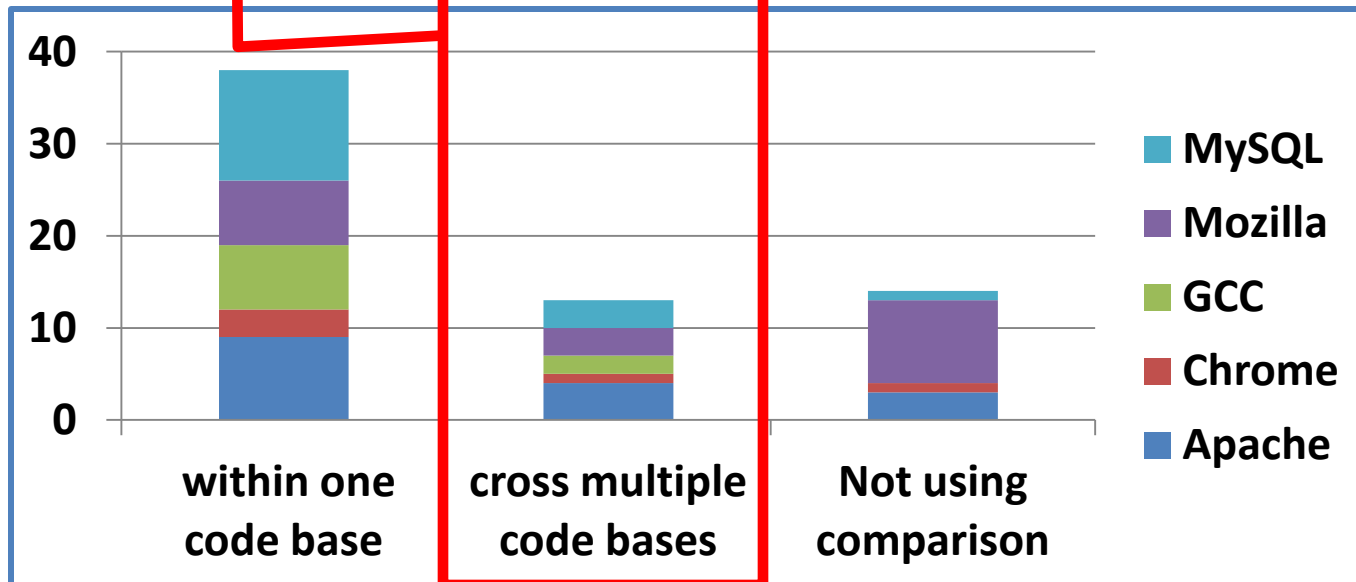
# How Perf. Bugs are Observed

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

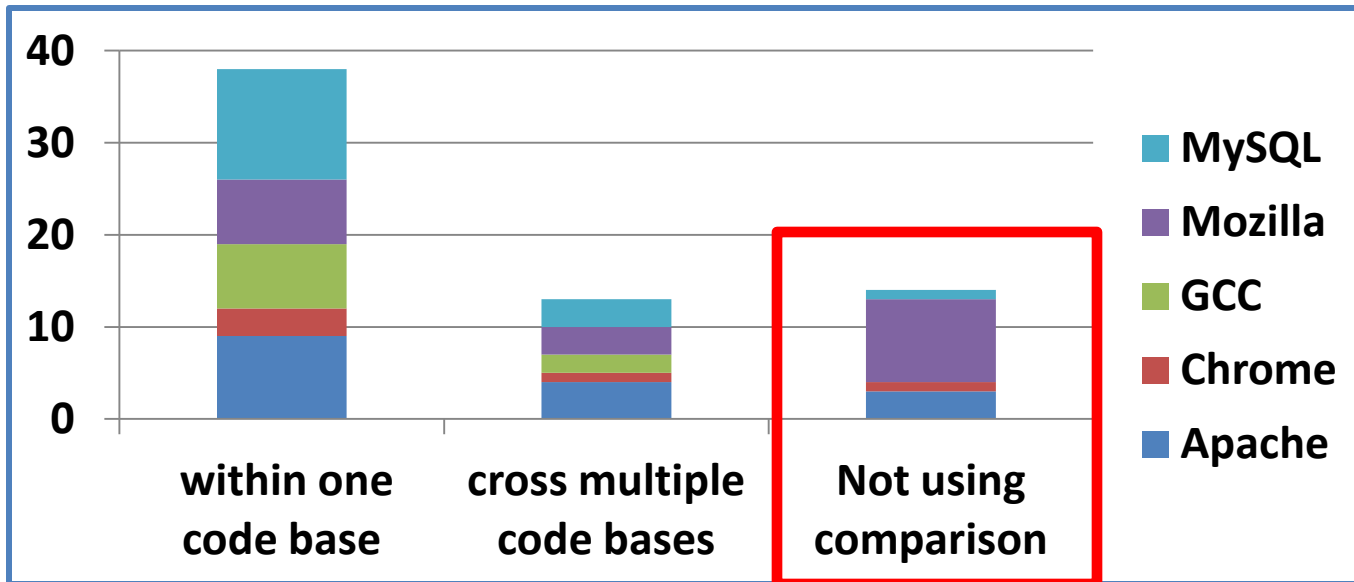


# How Perf. Bugs are Observed

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



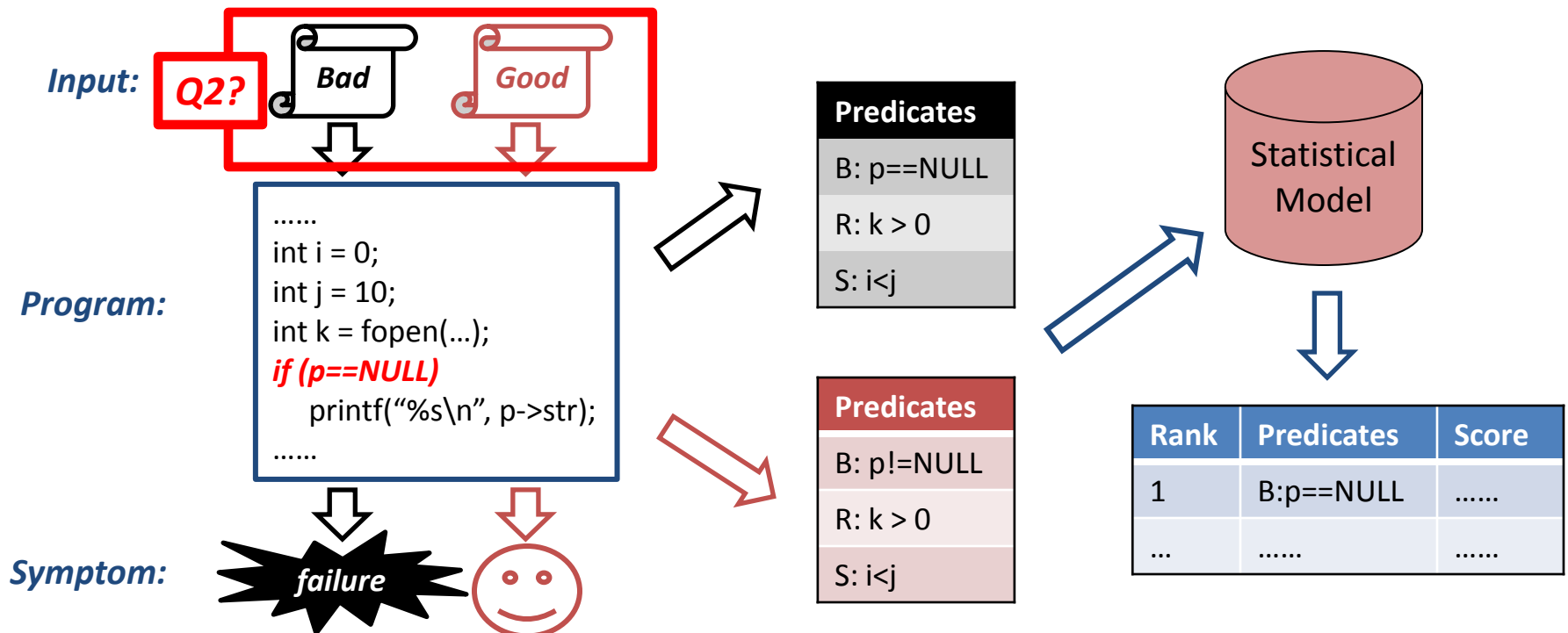
# How Perf. Bugs are Observed



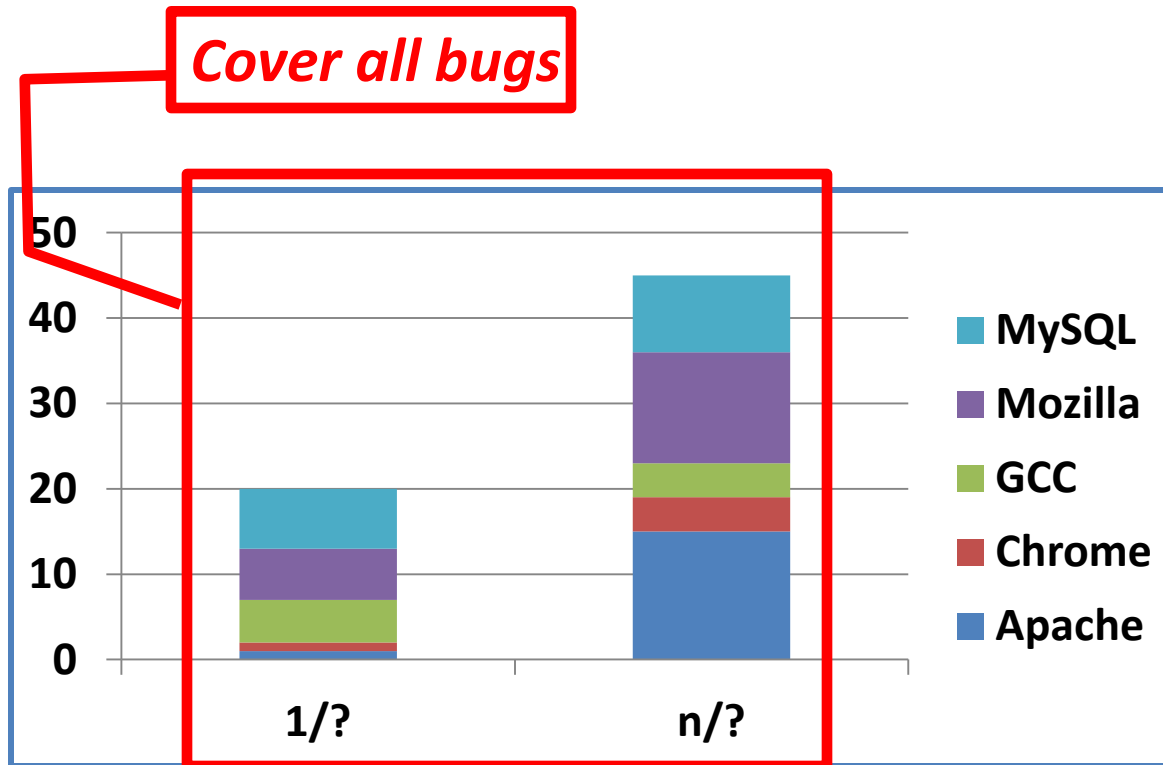


# Q2: How to obtain inputs?

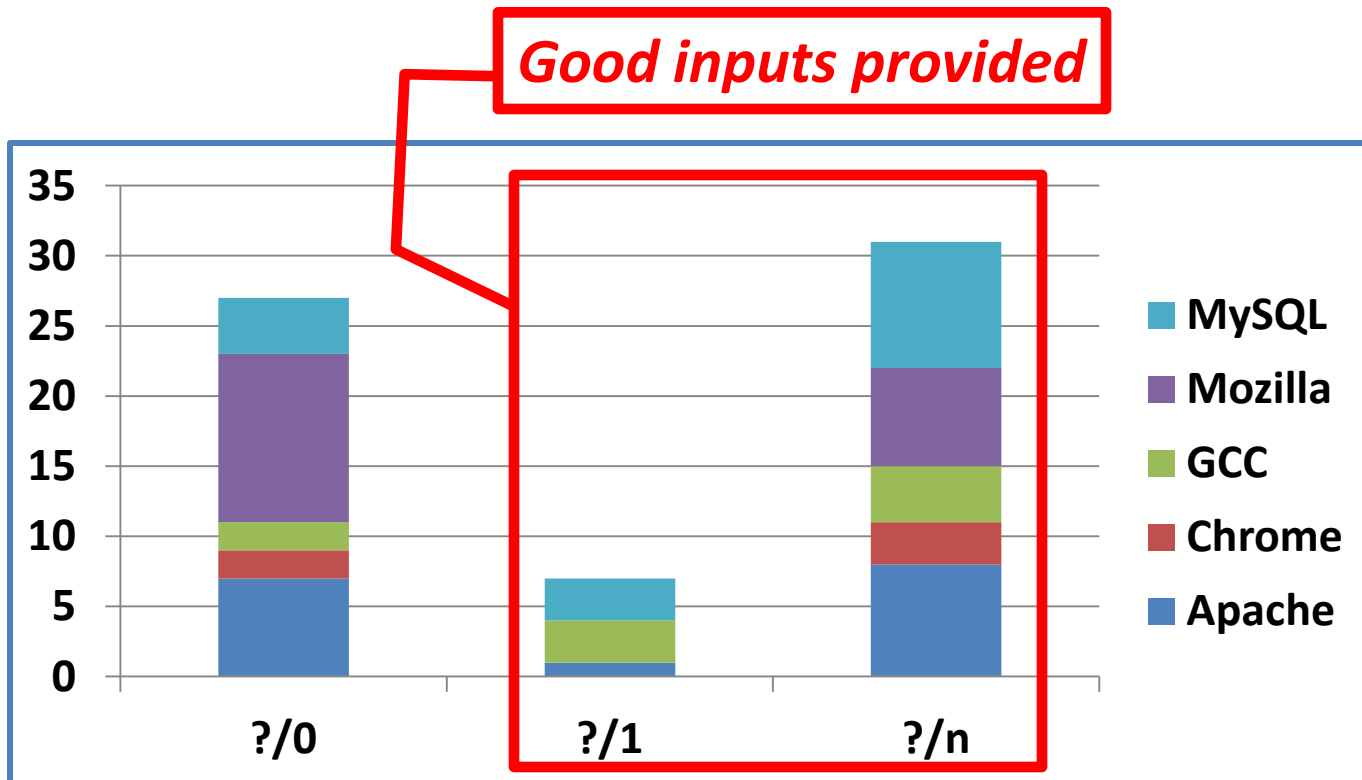
- How about statistical debugging
  - **Q1:** How to identify failure runs?
  - **Q2:** How to obtain inputs?



# 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

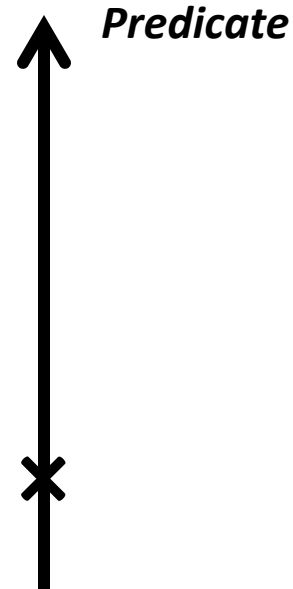
- Overview
- Diagnosis process study
- **In-house diagnosis study**
- On-line diagnosis study
- Conclusion

# Design

- In-house diagnosis
- Predicate design

– Branch

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



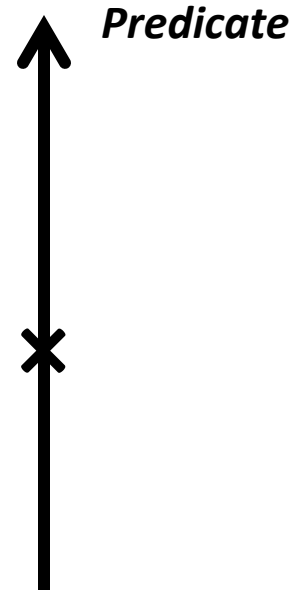
# Design

- In-house diagnosis
- Predicate design

- Branch
- Return

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

```
n=fopen(...);
```





# Design

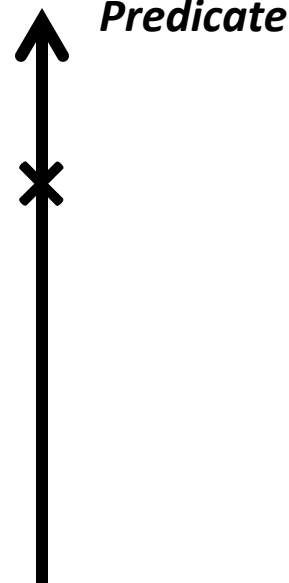
- In-house diagnosis
- Predicate design

- Branch
- Return
- Scalar-pair

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

```
n=fopen(...);
```

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



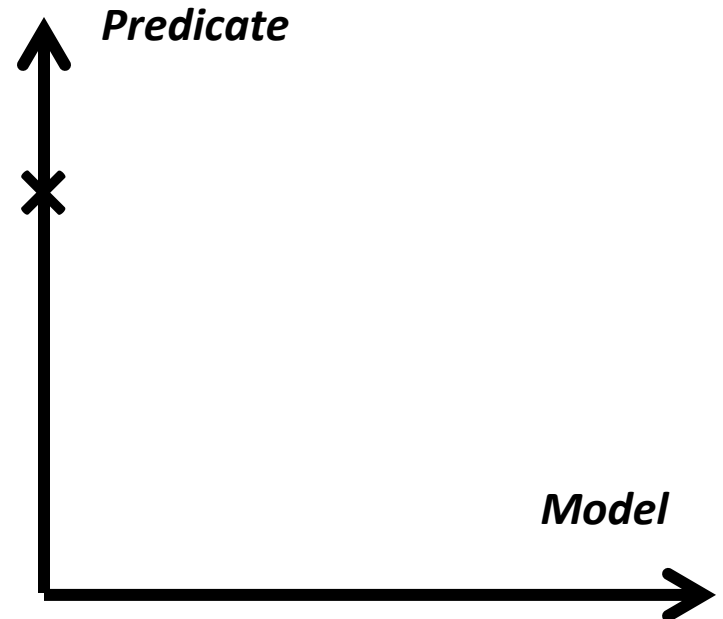
# Design

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

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

```
n=fopen(...);
```

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



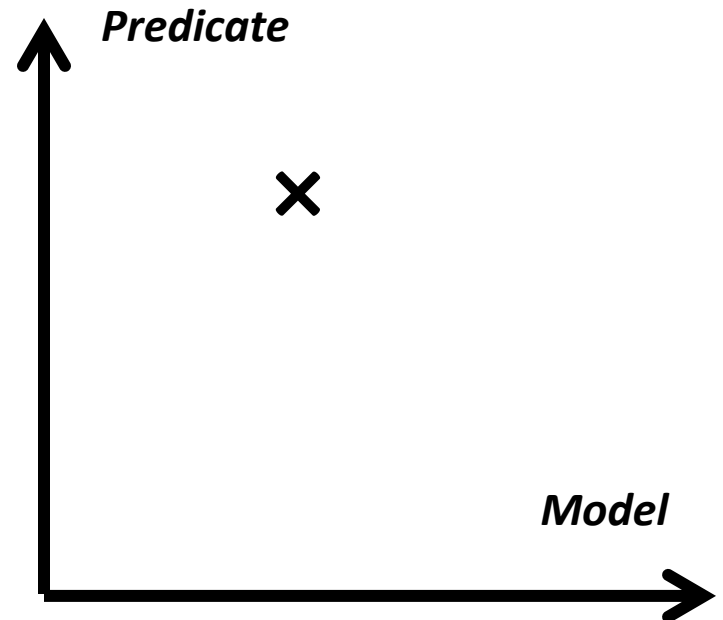
# Design

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

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

```
n=fopen(...);
```

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



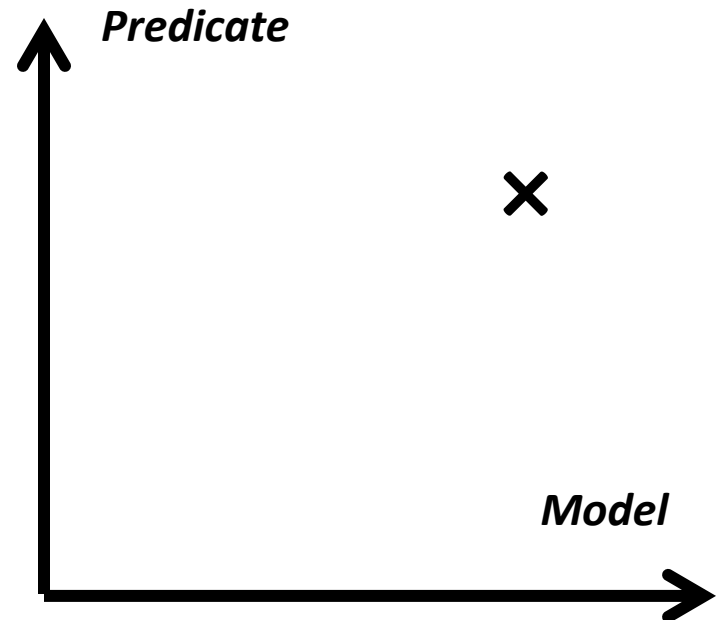
# Design

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

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

```
n=fopen(...);
```

```
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











# 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%	<b>10</b>
Mozilla411722	√1	3.35%	1000
MySQL15811	√1	8.58%	<b>10</b>
MySQL26527	√1	7.06%	1000
MySQL27287	√1	2.62%	<b>10</b>
MySQL40337	√1	3.32%	1000
MySQL42649	√1	4.67%	1000
MySQL44723	√1	0.40%	1000
Apache3278	√1	3.22%	1000
Apache34464	√1	2.13%	<b>10</b>
...	...	...	...

# Experimental Results

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

# 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!

