Diagnosing Production-Run Failures Via White, Gray, Black Box Approaches

Yuanyuan (YY) Zhou

Dept. of Computer Science and Engineering
University of California, San Diego

yyzhou@cs.ucsd.edu

http://www.cs.ucsd.edu/~yyzhou

Collaboration with Ding Yuan, Weihang Jiang, Soyeon Park, Jing Zheng
& Shankar Pasupathy, Arkady Kanevsky @NetApp

My Zigzag road in Diagnosis

Yuanyuan (YY) Zhou

Dept. of Computer Science and Engineering
University of California, San Diego

yyzhou@cs.ucsd.edu

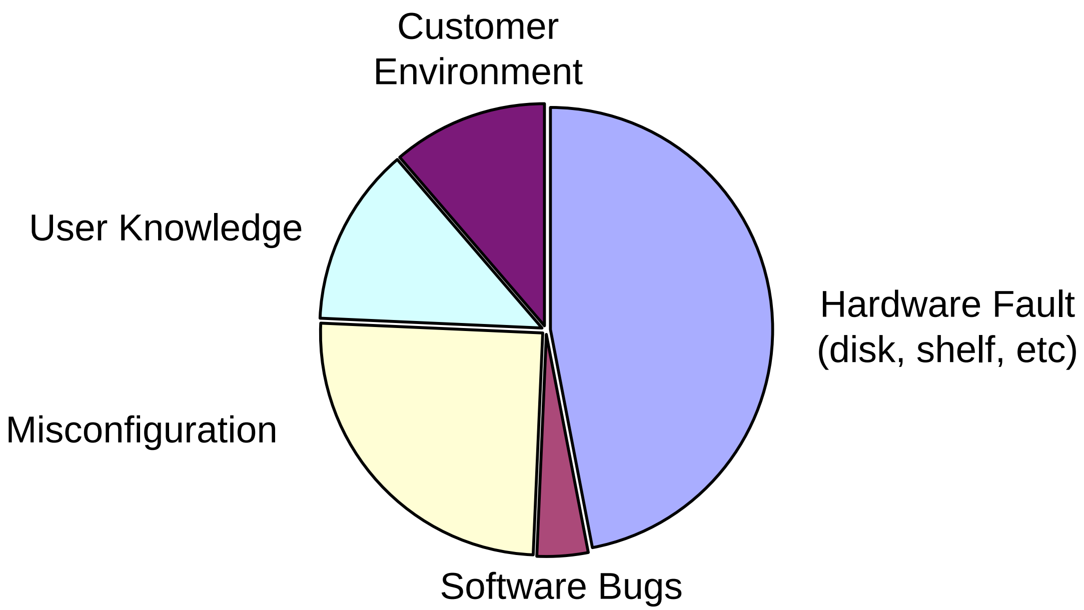
http://www.cs.ucsd.edu/~yyzhou

Collaboration with Ding Yuan, Weihang Jiang, Soyeon Park, Jing Zheng
& Shankar Pasupathy, Arkady Kanevsky @NetApp

My Assumption/Apology

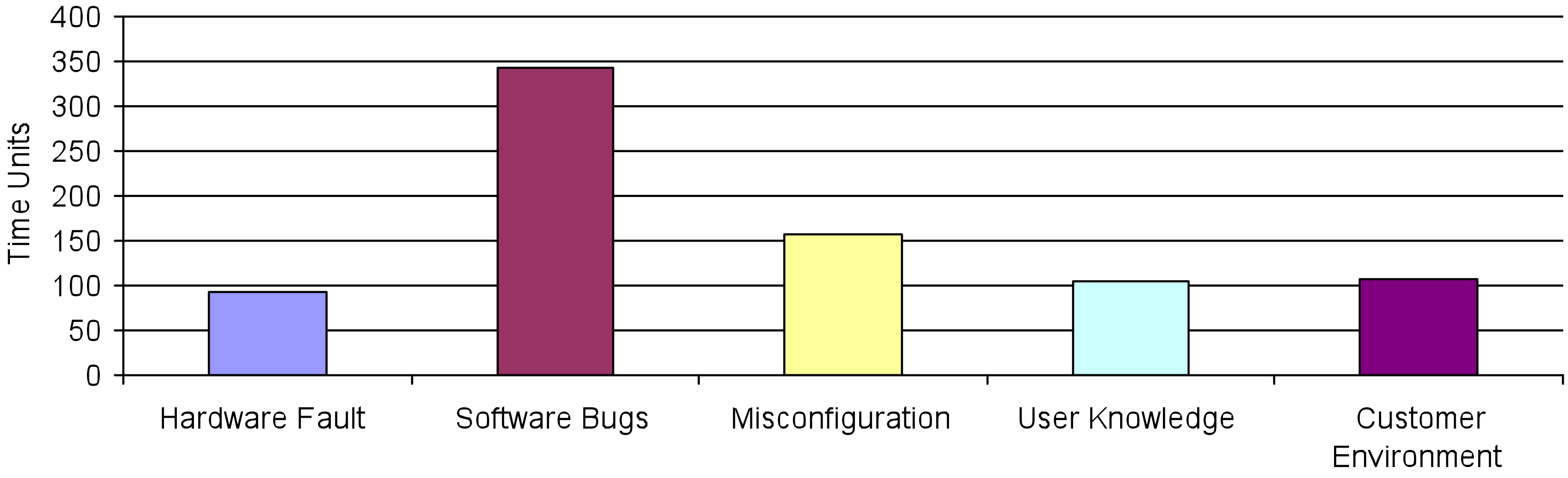
- My talk focuses on diagnosing of a single piece of software system from vendors’ customer support point of view
  - Mostly servers, not distributed systems, or clouds 🙁
  - Mostly experimental, no formula 🙁
- So pls don’t shoot me…

Once upon an opportunity…



- Data source: **NetApp 3 years of customer issues (636,108 cases)**
- Hardware fault (40%) and misconfiguration (21%) are the two most frequent categories, software bugs count for a small percentage(3%).
- User knowledge (11%) and customers' own execution environment (9%)
- More details in our joined FAST'09 paper

Problem category and troubleshooting time



- Software bugs take much longer time to troubleshoot.
- For all categories, troubleshooting is time-consuming

Troubleshooting is expensive!

- Costly downtime for customers
  - Cost a customer 18.35% of TCO [Crimson ’07]

- Expensive support cost for vendors
  - Vendors devote more than 8% of total revenue and 15% of total employee costs on customer problem support [ASP’08]

- Clouds further worse the problem

Vendor Support Costs

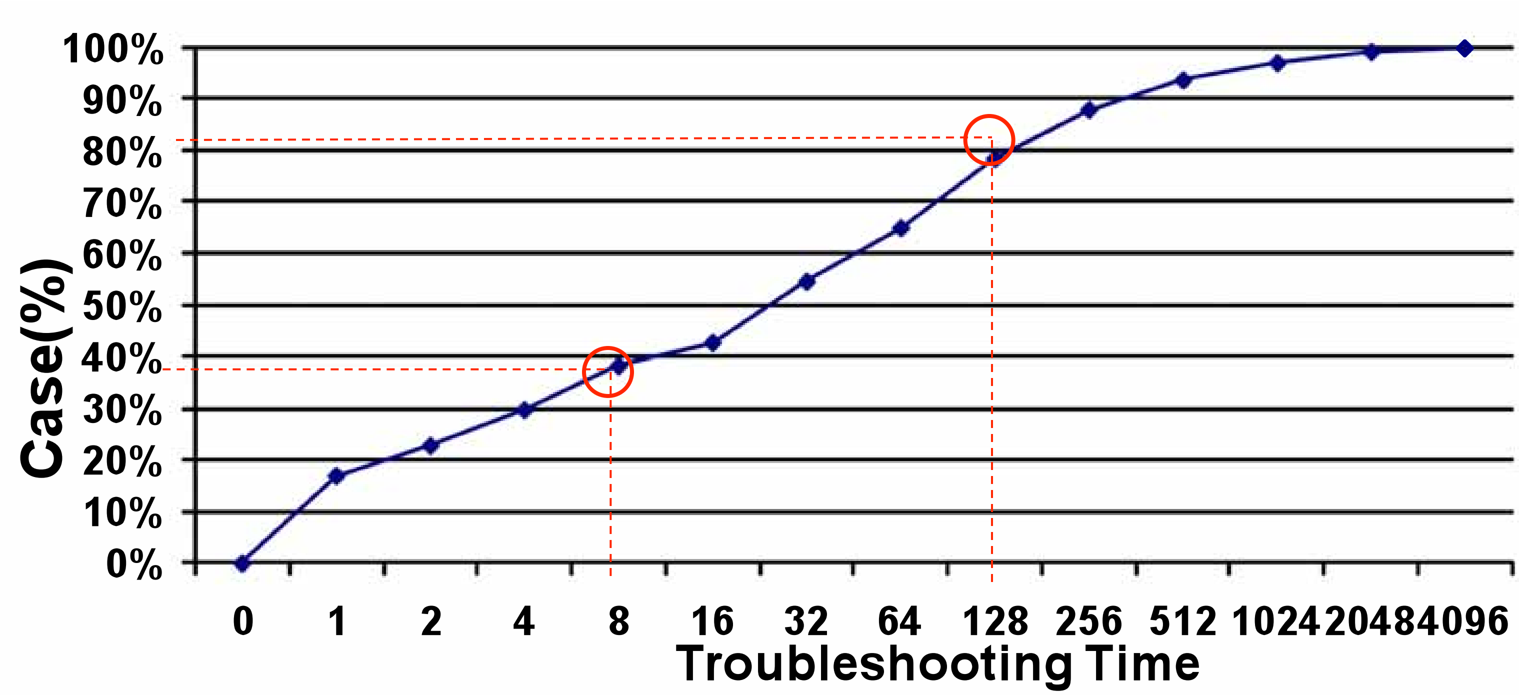
Company	Cost of Service	Revenue of Service
NetApp	$0.37B	$0.573B
EMC	$1.7B	$2.8B
Cisco	$2.6B	$6.9B
Juniper	$0.31B	$0.64B
Oracle	$3.9B	$4.6B
VmWare	$0.21B	$0.66B

35% increase/year

* numbers are from 10-K financial reports of these companies

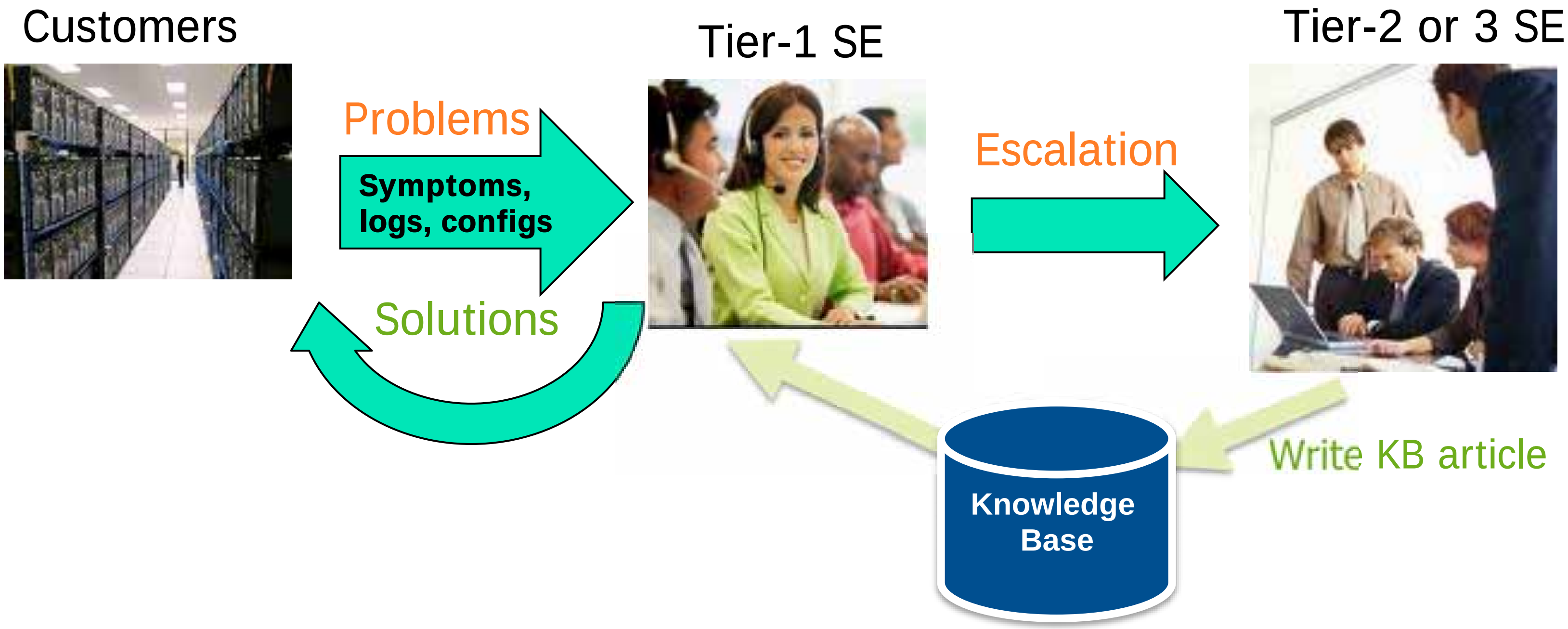
- **Other costs**
  - Customer satisfaction and competitiveness
  - Interruption to on-going effort for new product/feature development

Troubleshooting is time-consuming



Data source: NetApp and also in our FAST'09 paper

The source of the pain: Lack of automation



Customers

Problems

Symptoms, logs, configs

Solutions

Tier-1 SE

Escalation

Tier-2 or 3 SE

Write KB article

Knowledge Base

- **Labor intensive**
- **Long diagnosis time**
- **Inaccurate and expensive resolution**
- **Not scalable**

Reality: Hard to reproduce Production-run failures

Same Input

Same Configuration

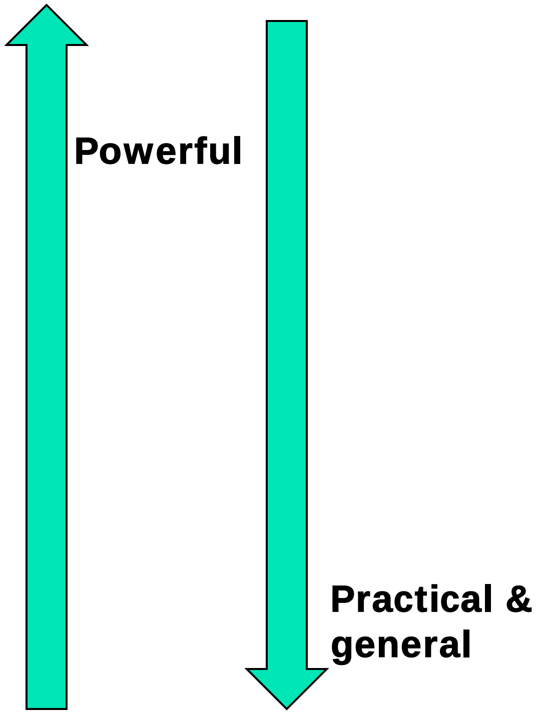
Same Environment

Can’t reproduce the failure!!!

```
bash>./execute
… …
… …
```

Our Zigzag Experience So Far

- **White** boxes
  - Actively collecting more diagnostic information
  - Ex: Triage [SOSP'07], PRES[SOSP'09]
- **Gray** boxes
  - Analyzing logs + source code
  - Example: SherLog[ASPLOS'10]
- **Black** boxes
  - Relying on logs alone
  - Example: LogMining[FAST'09]



Talk Outline

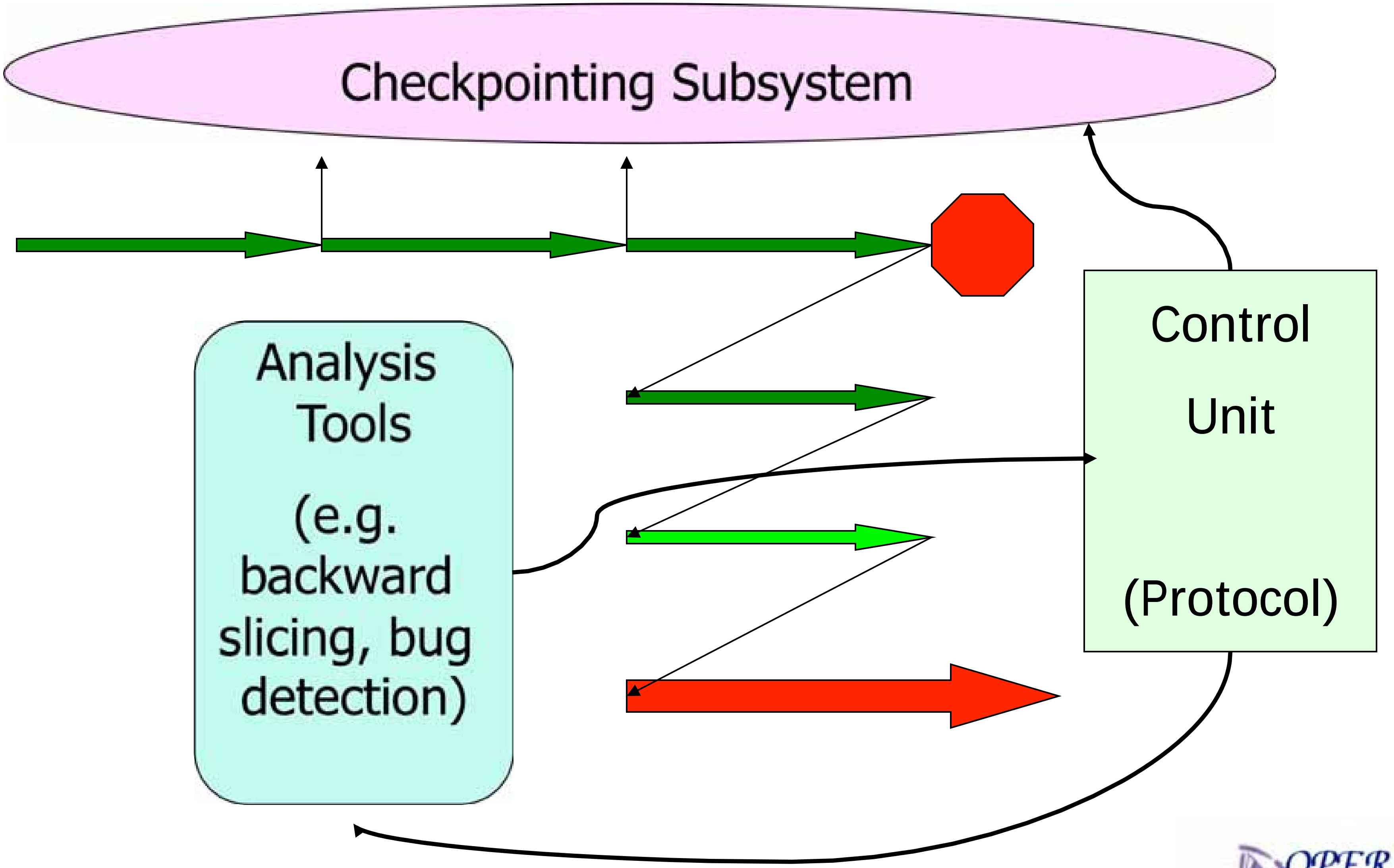
- Motivation
- Brief overview
  - ➡ White-Box: Triage [SOSP’07]
  - Black-Box: LogMining [FAST’09]

- Gray-box: SherLog [ASPLOS’10]
  - A good balance between effectiveness and practicality

Triage: Automatic, On-Site Failure Diagnosis

- Goal:
  - Collect as much diagnostic information
- Idea
  - Leverage the failure moment
  - Relive the failure multiple times via automatic rollback and re-execution
  - Each re-execution with some diagnostic techniques (slicing) enabled
  - Play what-ifs and delta-analysis to narrow down the possible root causes

Triage Process



Triage Details

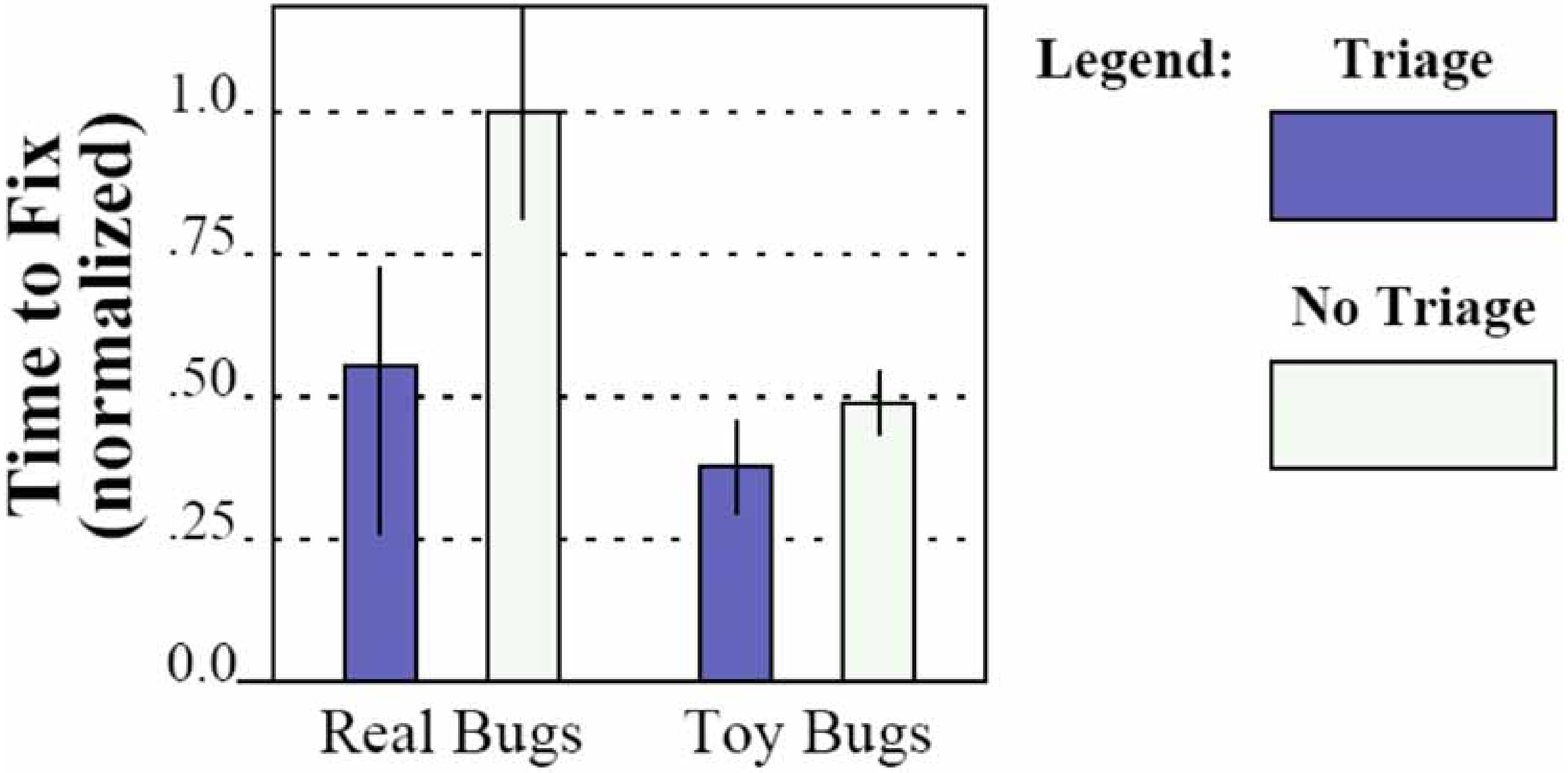
- How to get information about the failure?
  - Capture the bug with checkpoint/re-execution
  - Relive the bug on-site with various diagnostic techniques
- How to decide what to do?
  - Use a human-like protocol to select analysis
  - Incrementally increase our understanding of the bug
- How to try out “what-if” scenarios?
  - Controlled re-execution allows varied executions
  - Delta analysis points out what makes them different

Results – Human Study

- Many results in our paper
  - Effectiveness and efficiency using 10 real world failures in server applications
  - Checkpoint overhead
- Human study results
  - 15 programmers
  - Measured time to repair bugs, with/without Triage
    - Everybody got core dumps, sample inputs, instructions on how to replicate, and access to many debugging tools
      - Including Valgrind
  - 3 simple toy bugs, & 2 real bugs

Results – Human study

- For the real bugs, Triage strongly helps

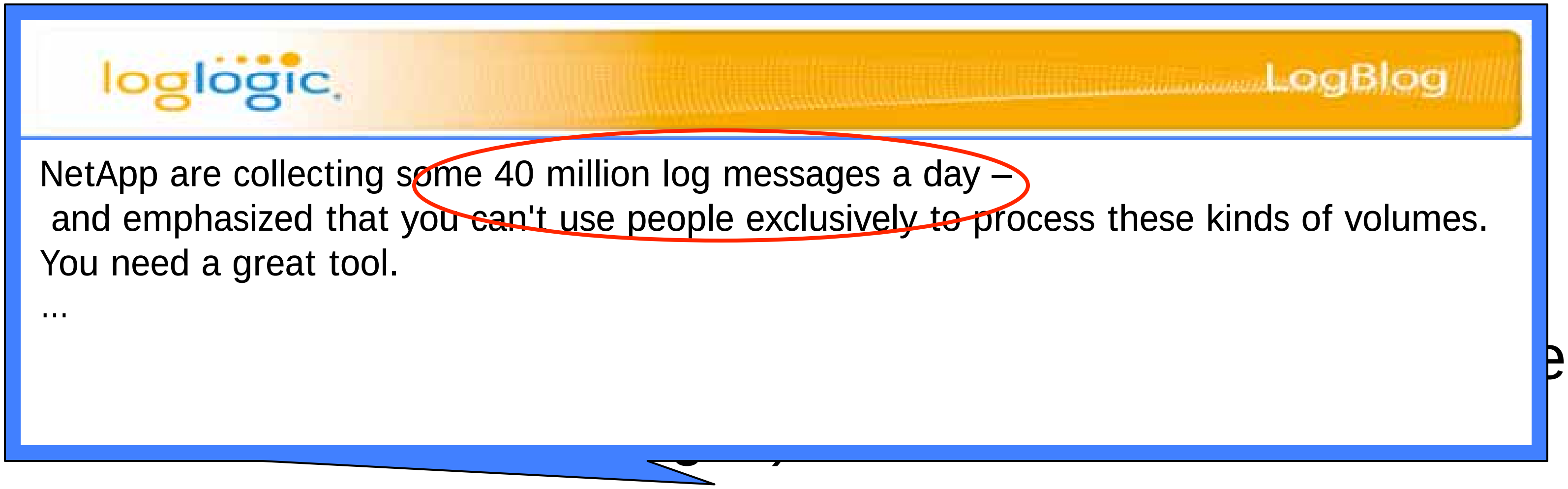


Road to Impact?

- We enthusiastically took our solution to industry

- But they said “interesting….but no” because
  - The integration complexity/cost is high
  - It require checkpoints at run-time

Common Practice in Industry:

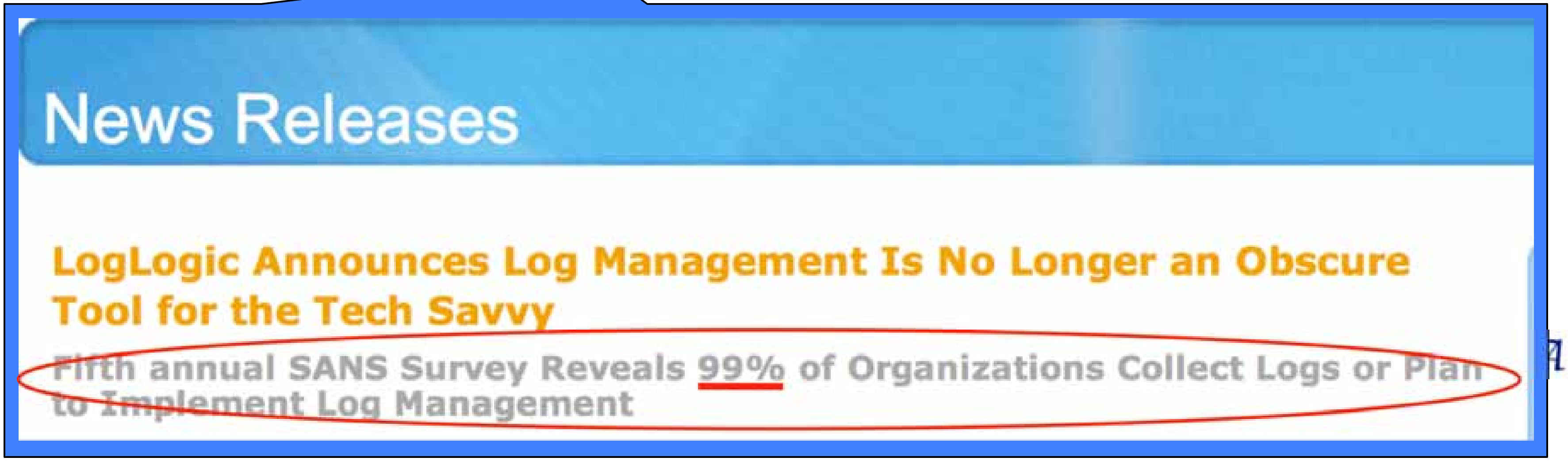


NetApp are collecting some 40 million log messages a day – and emphasized that you can't use people exclusively to process these kinds of volumes. You need a great tool.

...

es (900

- Log Collections:
  - EMC, NetApp, Cisco, Dell collect logs from >50% of their customers [SANS2009][EMC][Dell]



LogLogic Announces Log Management Is No Longer an Obscure Tool for the Tech Savvy

Fifth annual SANS Survey Reveals 99% of Organizations Collect Logs or Plan to Implement Log Management

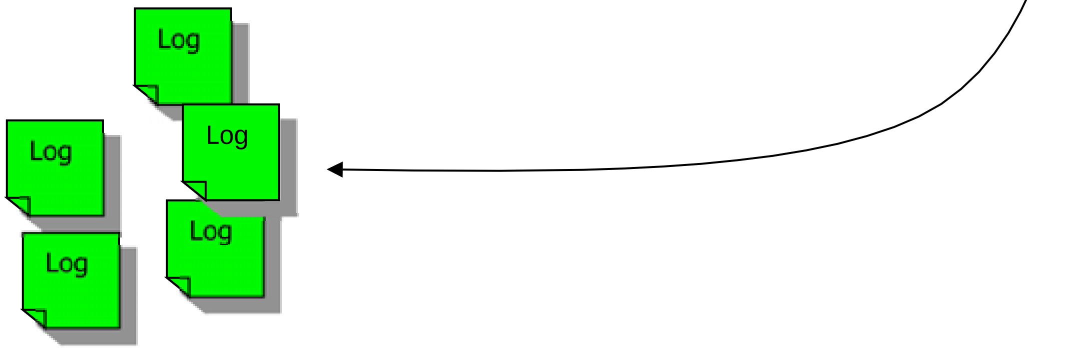
Talk Outline

- Motivation
- Brief overview
  - White-Box: Triage
  - ➡ Black-Box: LogMining [FAST'09]
- Gray-box: SherLog
  - A good balance between effectiveness and practicality

Using logs for diagnosis

Customer problem case database (636,108)

Case ID	Report Date	Resolution/ Workaround Date	Problem cause		Auto-generated	Critical Event
			High-level	Module-level		
1	5/1/06 11:21	5/1/06 13:35	Software Bugs	File System	Y	Crash
2	5/2/06 11:02	5/2/06 9:01	Hardware Fault	SCSI	N	N/A
3	5/3/06 15:40	5/8/06 14:48	Misconfiguration	Shelf	N	N/A



Storage System Log Archive (306,624 logs)

Challenges and opportunities

😐 Logs are noisy

Single Event revealing problem root cause

```
Sat Apr 15 05:58:15 EST [busError]: SCSI adapter encountered an unexpected bus phase. Issuing
      SCSI bus reset.
Sat Apr 15 05:59:10 EST [fs.warn]: volume /vol/vol1 is low on free space. 98% in use.
Sat Apr 15 06:01:10 EST [fs.warn]: volume /vol/vol10 is low on free space. 99% in use.
Sat Apr 15 06:02:14 EST [raidDiskRecovering]: Attempting to bring device 9a back into service.
Sat Apr 15 06:02:14 EST [raidDiskRecovering]: Attempting to bring device 9b back into service.
                                                  ......
Sat Apr 15 06:07:19 EST [timeoutError]: device 9a did not respond to requested I/O. I/O will be
      retried.
Sat Apr 15 06:07:19 EST [noPathsError]: No more paths to device 9a: All retries have failed.
Sat Apr 15 06:07:19 EST [timeoutError]: device 9b did not respond to requested I/O. I/O will be
      retried.
Sat Apr 15 06:07:19 EST [noPathsError]: No more paths to device 9b. All retries have failed.
Sat Apr 15 06:08:23 EST [filerUp]: Filer is up and running.
                                                   ......
Sat Apr 15 06:24:07 EST [crash:ALERT]: Crash String: File system hung in process idle_thread1
```

→ **Critical Event**

Challenges and opportunities

- 😐 Logs are noisy
- 😐 Important log events are not easy to locate

Single Event revealing problem root cause

```
Sat Apr 15 05:58:15 EST [busError]: SCSI adapter encountered an unexpected bus phase. Issuing
    SCSI bus reset.
```

Total of 106 log events

Critical Event

```
Sat Apr 15 06:24:07 EST [crash:ALERT]: Crash String: File system hung in process idle_thread1
```

Challenges and opportunities

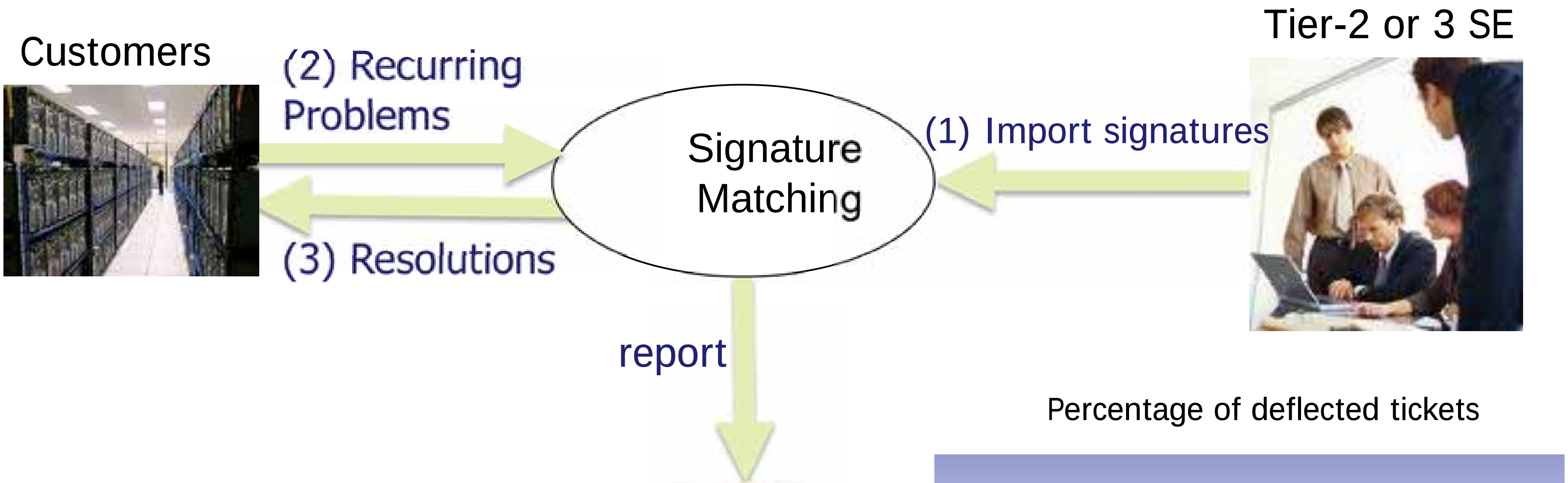
Challenges:

- 😐 Logs are noisy
- 😐 Important log events are not easy to locate

Opportunities:

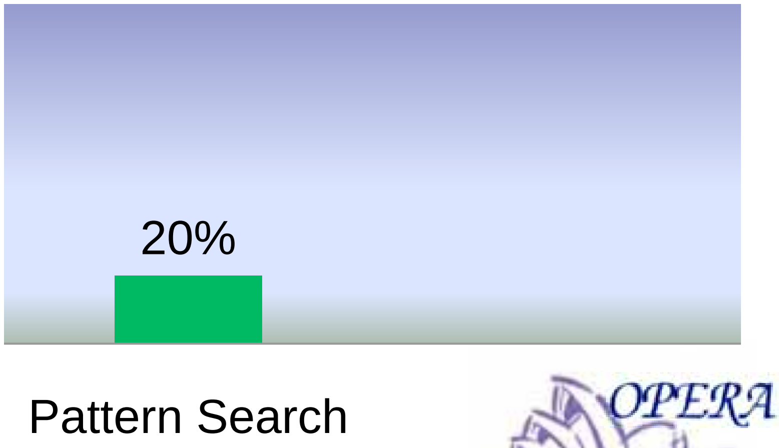
- ☺ Similar log patterns appear on systems experience the same problems

Failure Signature Matching

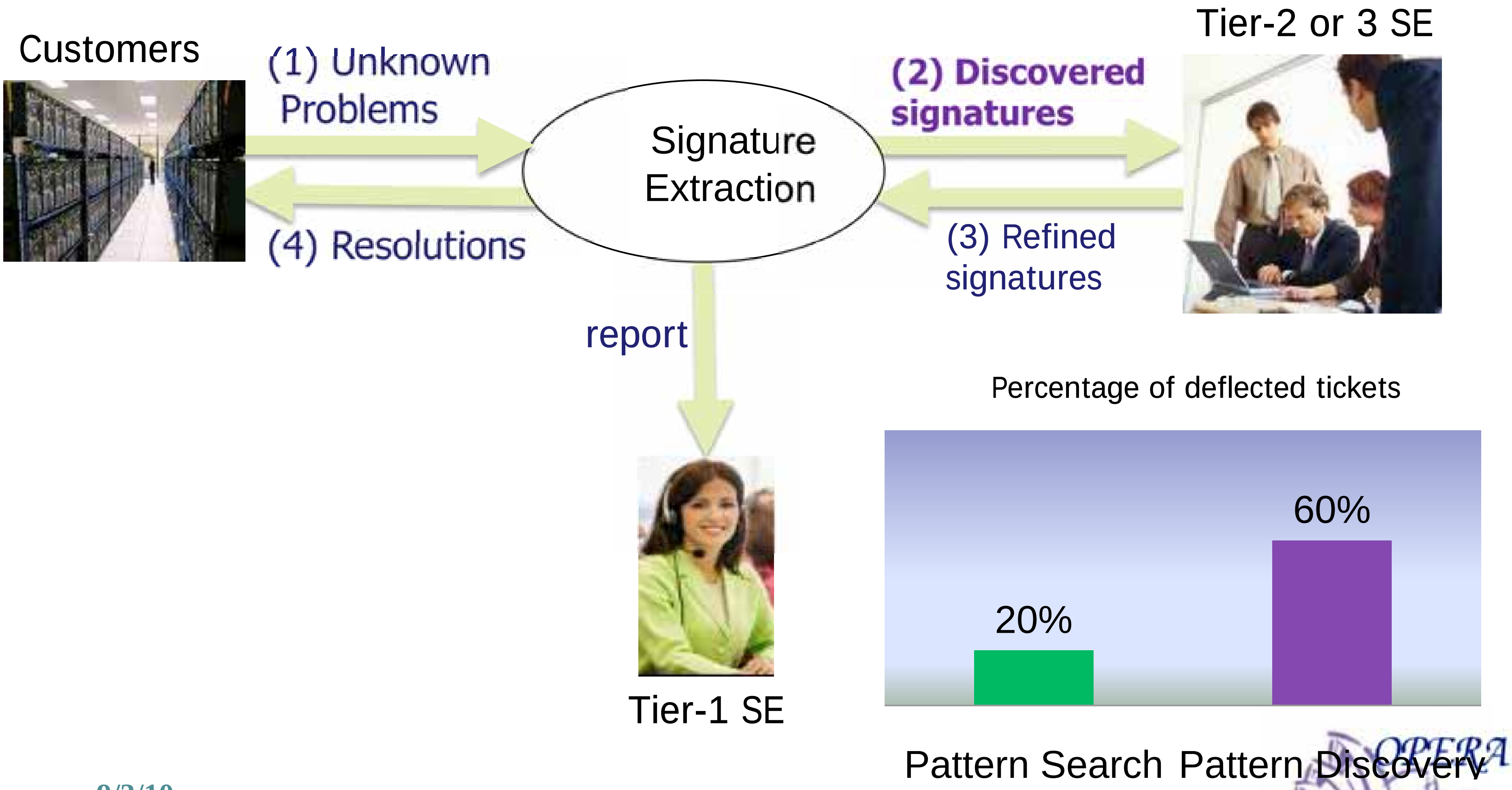


Tier-1 SE

Percentage of deflected tickets



Failure Signature Extraction



Proof of Concept Results@ a company

Signature Matching

Coverage	> 90%
Accuracy	95% – 100%
Easy to Use	50 – 100 signatures / day inserted
Speed	< 5 seconds / search query
Scalability	TB of data (1 month of logs)

Signature Extraction

	Company A(manually)	**Ours**
Cost	10 engineers, 3 months	1 PI engineer 3 days
Accuracy	--	> 80%
Signatures	13	16
Scalability	--	18000 cases

* Data from pilot at Netapp

Limitations of Log Mining

- Same limitation as other black box approaches
  - Do not take the system internal into consideration
- Limited by the quality of logs

- So what else can we take advantage without losing practicality

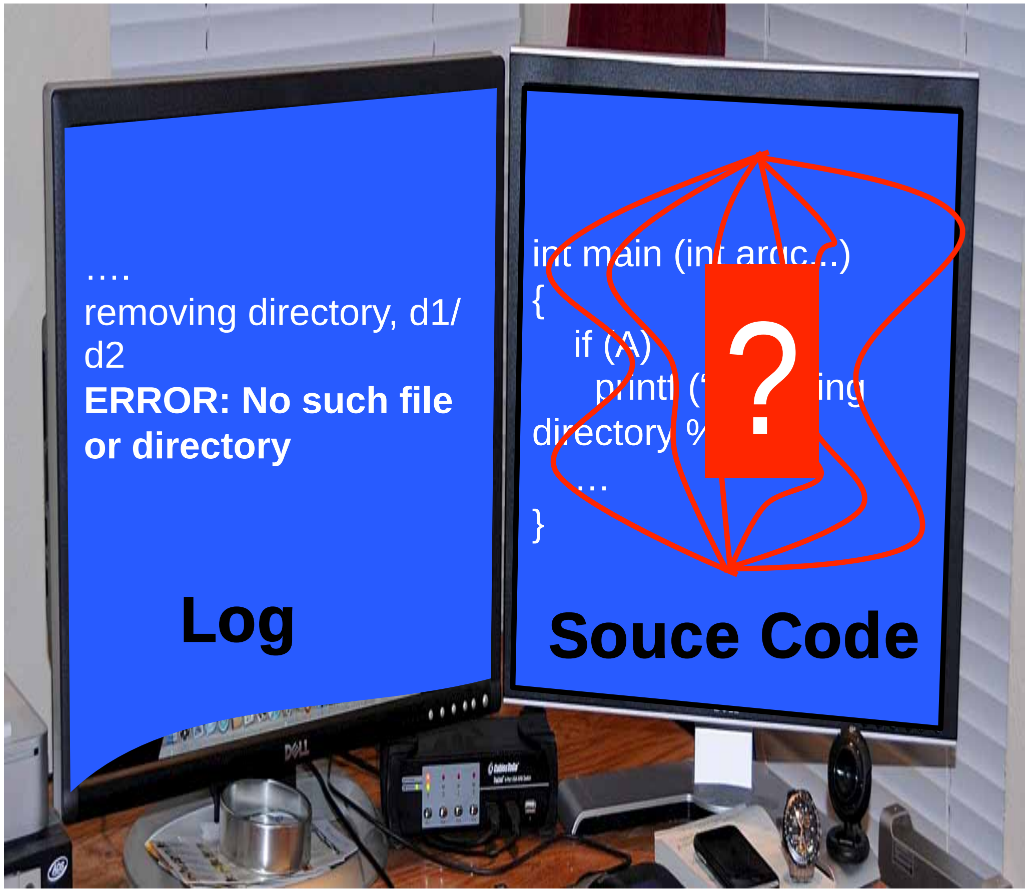
Talk Outline

- Motivation
- Brief overview
  - White-Box: Triage
  - Black-Box: LogMining [FAST'09]

➡ Gray-box: SherLog

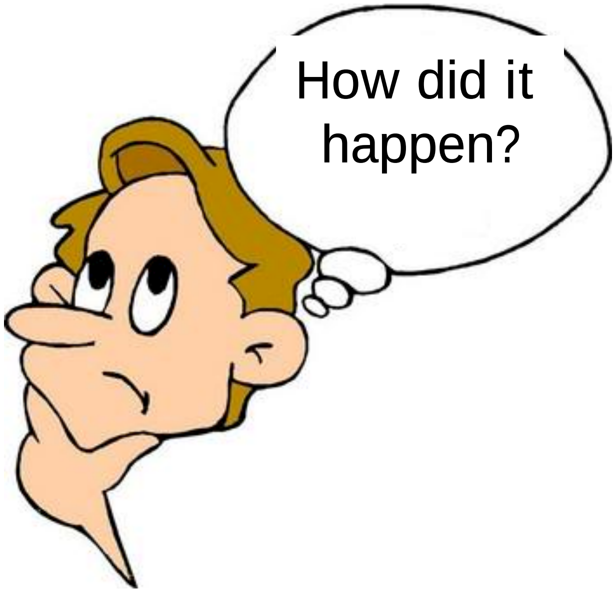
  - A good balance between effectiveness and practicality

Manual Inference with Log + Code



```
….
removing directory, d1/d2
ERROR: No such file or directory
```

```
int main (int argc...)
{
   if (A)
     printf ("   ing
directory %
   …
}
```



Follow Complex programming logic

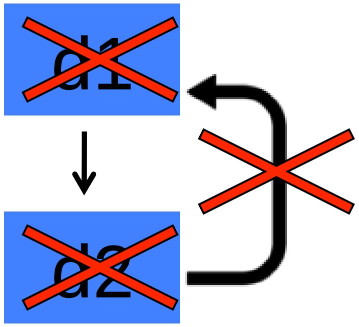
Tedious, Error prone,
Can’t be carried deep

Real Failure in rmdir@GNU Coreutils

- rmdir
  - Remove an empty directory
  - When –p specified, remove all the parent directory as well!
- Failed to remove parent directory

rmdir -pv d1/d2/

```
rmdir: removing directory, d1/d2/  [msg 1]
rmdir: removing directory, d1/d2   [msg 2]
rmdir: ‘d1/d2’: No such file or directory
[msg 3]
```



Highly Simplified from 18K LOC

Only relevant code

```
rmdir: removing directory, d1/d2/
rmdir: removing directory, d1/d2
rmdir: ‘d1/d2’: No such file or directory
```

```
remove_parents (char *path) {
  char *slash;
  while (1) {
    slash = strrchr (path, '/');
    if (slash == NULL)
                break;

    slash[0] = 0;

    if (verbose)
       error (0,0,_("removing directory, %s”), path);

     fail = rmdir (path);

     if (fail) {
        error (0, errno, "%s", quote(path)); //r2
       break;
     }
   } //end while
} //end remove_parent
```

R1

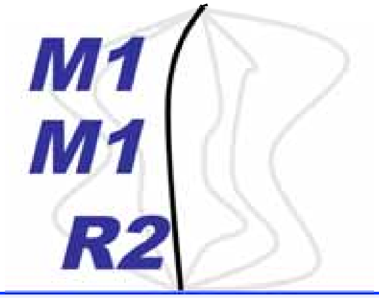
R2

```
main (argc, argv) {
  for (; optind < argc; optind++) {
    char* dir = argv[optind];
    if (verbose)
      error (0, 0, _("removing directory, %s”), dir);

      fail = rmdir (dir);

    if (fail)
      error (0, errno, “%s”, quote(dir));
    else if (empty_paths)
      remove_parents (dir);
  } //end for
} //end main
```

M1

M2

8 combinations:



```
1 remove_parents (char *path) {
2   char *slash;
3   while (1) {
4     slash = strrchr (path, '/');
5     if (slash == NULL)
6               break;
7
8     slash[0] = 0;
9
10    if (verbose)
11      error (0,0,_("removing directory, %s"),
              path);
12
13    fail = rmdir (path);
14
15    if (fail) {
16       error (0, errno, "%s",
          quote(path)); //r2
17      break;
18     }
19    } //end while
20 } //end remove_parent
```

R1

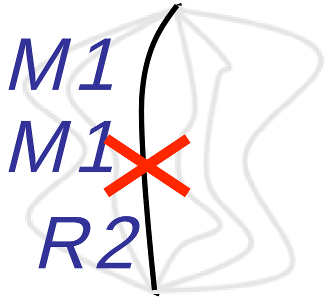
R2

```
22 main (argc, argv) {
23   for (; optind < argc; optind++) {
24     char* dir = argv[optind];
25     if (verbose)
26      error (0, 0, _("removing directory, %s")
                 dir);
27
28      fail = rmdir (dir);
29
30     if (fail)
31      error (0, errno, "%s",
                quote(dir));
32     else if (empty_paths)
33       remove_parents (dir);
34   } //end for
35 } //end main
```

M1

M2

8 combinations:



```
1 remove_parents (char *path) {
2   char *slash;
3   while (1) {
4     slash = strrchr (path, '/');
5     if (slash == NULL)
6                break;
7
8     slash[0] = 0;
9
10    if (verbose)
11      error (0,0,_("removing directory, %s"),
              path);
12
13    fail = rmdir (path);
14
15    if (fail) {
16        error (0, errno, "%s",
           quote(path)); //r2
17      break;
18    }
19  } //end while
20 } //end remove_parent
```

verbose == 0

R2

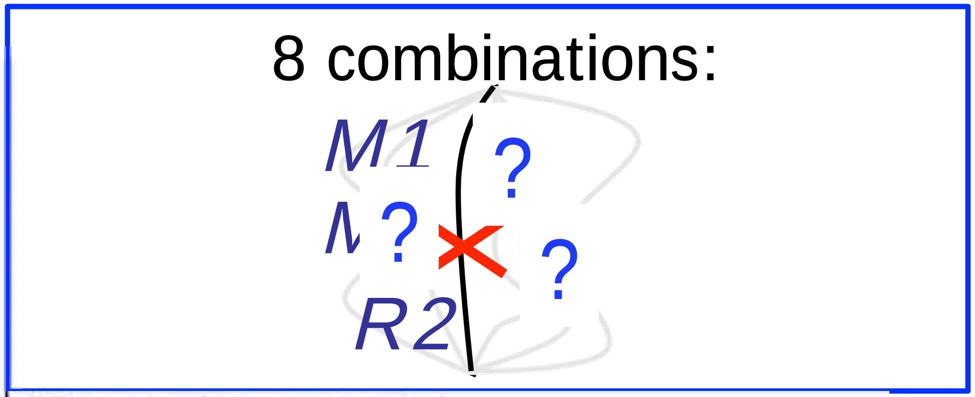
```
22 main (argc, argv) {
23   for (; optind < argc; optind++) {
24     char* dir = argv[optind];
25     if (verbose)
26      error (0, 0, _("removing directory, %s"),
                  dir);
27
28      fail = rmdir (dir);
29
30     if (fail)
31                              "%s",
32
33        remove_parents (dir);
34    } //end for
35 } //end main
```

verbose != 0

M1

Infeasible!!

8 combinations:



Complicated control- and data-flow

Other combinations?

```
ret.r                                                   &
(rmd                                                    ||
ret.errno_rmdir_non_empty == 0) && (((i<33))
(*slash) == 47) || !(slash > __arg0))
```

Non-Trivial Manual Effort!

```
while (1) {
  if (verbose)
    error (0,0,_("removing directory, %s"),
           path);

  fail = rmdir (path);

  if (fail) {
    error (0, errno, "%s",
           quote(path)); //r2
    break;
  }
} //end while
} //end remove_parent
```

~~R1~~

verbose == 0

R2

```
main (argc, argv) {
               < argc; optind++) {
               argv[optind];
          )
     , _("removing directory %s"),
    dir);

  fail = rmdir (dir);

  else if (empty_paths)
    remove_parents (dir);
  } //end for
} //end main
```

verbose != 0

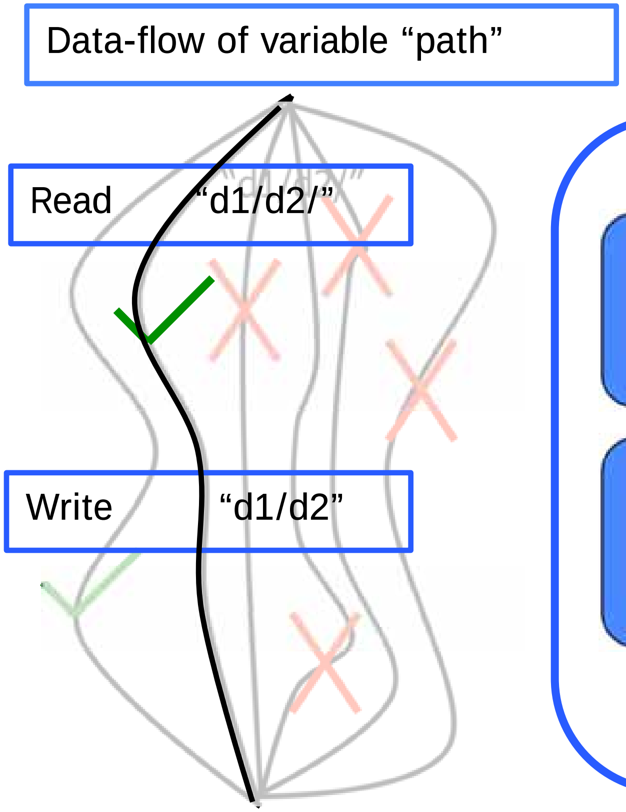
M1

```
optind < argc && ((1+optind) < argc) && !
(ret.getopt_long != -1) && rmdir.c:empty_paths !
= 0 && rmdir.c:verbose != 0 && ret.rmdir == 0
```

Infeasible!

M2

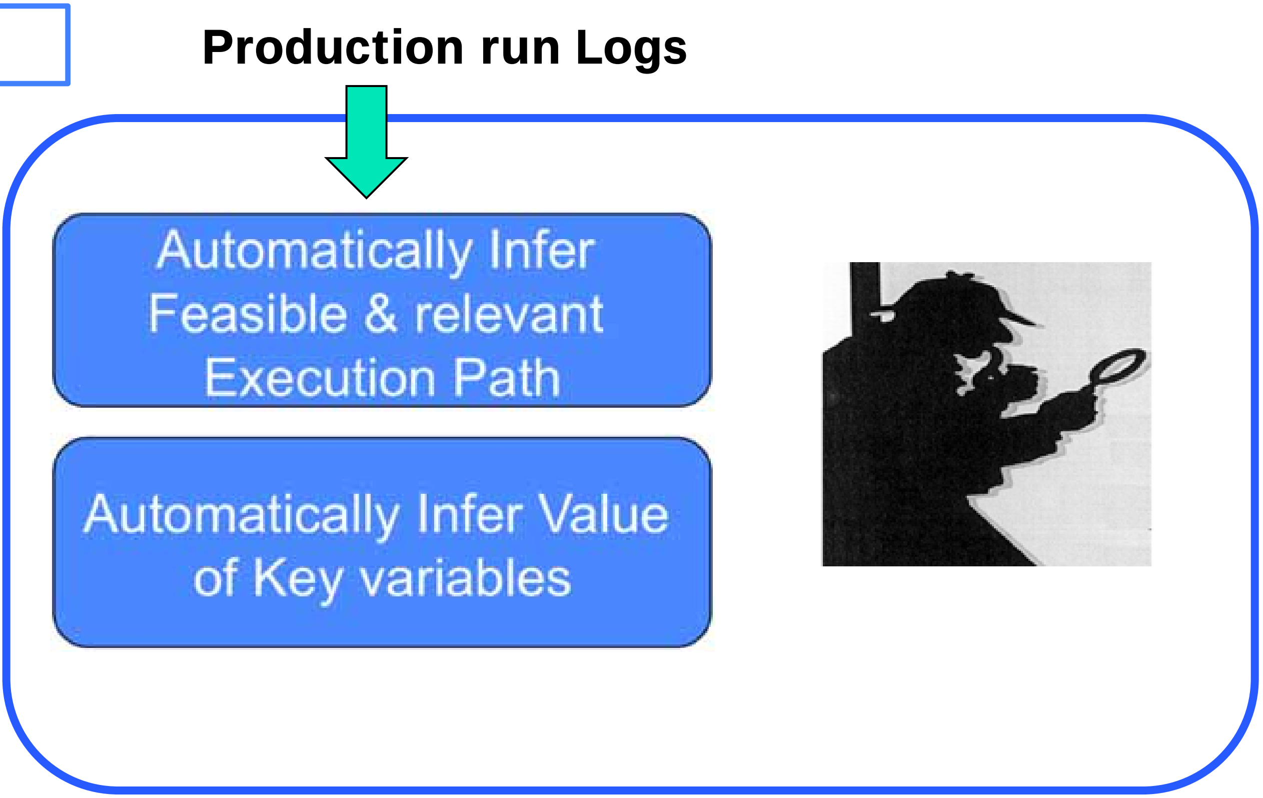
SherLog: Automated Log-Driven Inference



Data-flow of variable "path"

Read "d1/d2/"

Write "d1/d2"



Production run Logs

Automatically Infer Feasible & relevant Execution Path

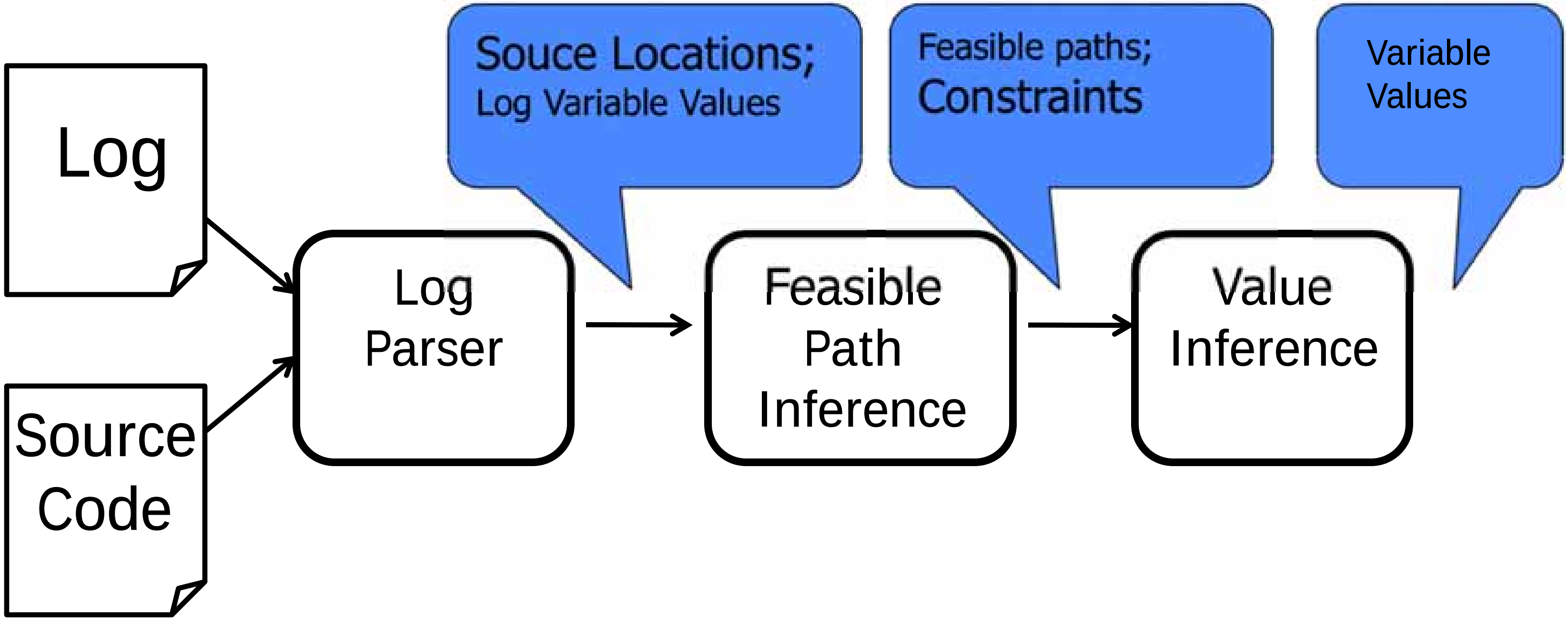
Automatically Infer Value of Key variables

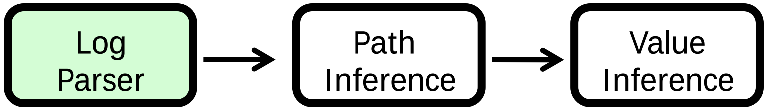
Details in our ASPLOS'10 paper

Dull material alert

- Time to take a nap!

SherLog Overview




Log Parser → Path Inference → Value Inference

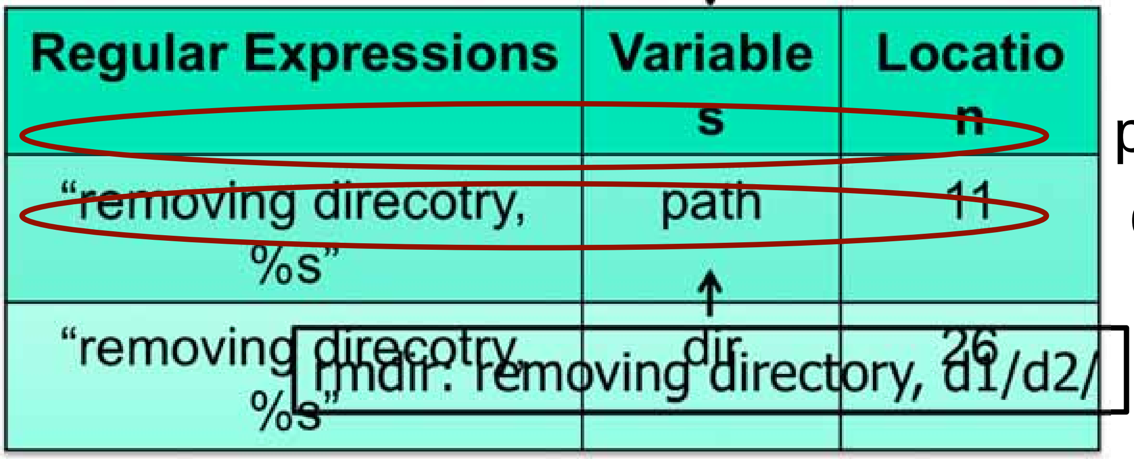
Log Parser

Provides simple annotation language for customized logging

- Goal
  - Map message to source code location
  - Map variable's value printed in log message
- Parse format string as Regular Expression

Source Code

```
error (0, 0, _("removing direcotry, %s"), path);
```



Regular Expressions	Variables	Location
"removing direcotry, %s"	path	11
"removing direcotry, %s"	dir	26

```
rmdir: removing directory, d1/d2/
```

path = "d1/d2/"

dir = "d1/d2/"

Log Parser
Path Inference
Value Inference

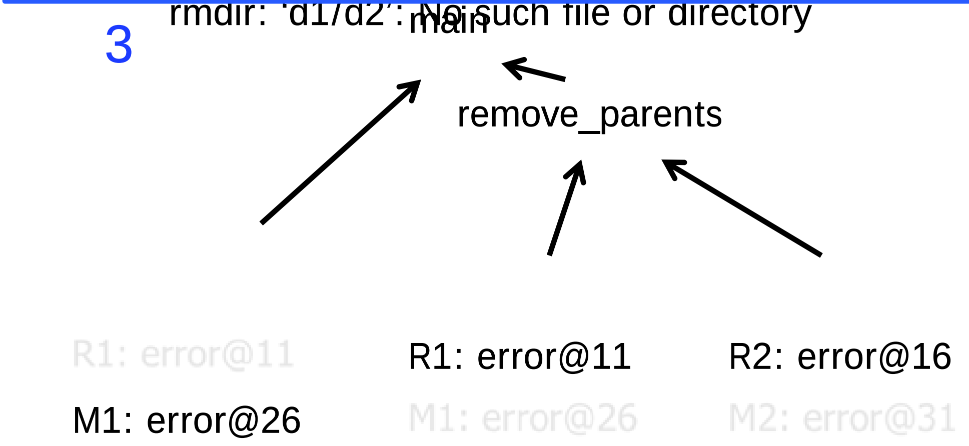
Feasible Path Inference

- Goal
  - Infer the Control Flow Paths that connects the log messages
- Problem Formalization
  - Sequence Matching problem

Feasible paths connect Longest sub-sequence of the log-site sequence

```
1  rmdir: removing directory, d1/d2/
2  rmdir: removing directory, d1/d2
3  rmdir: 'd1/d2': No such file or directory
```

Log Parser:
Maps to Logging Sites

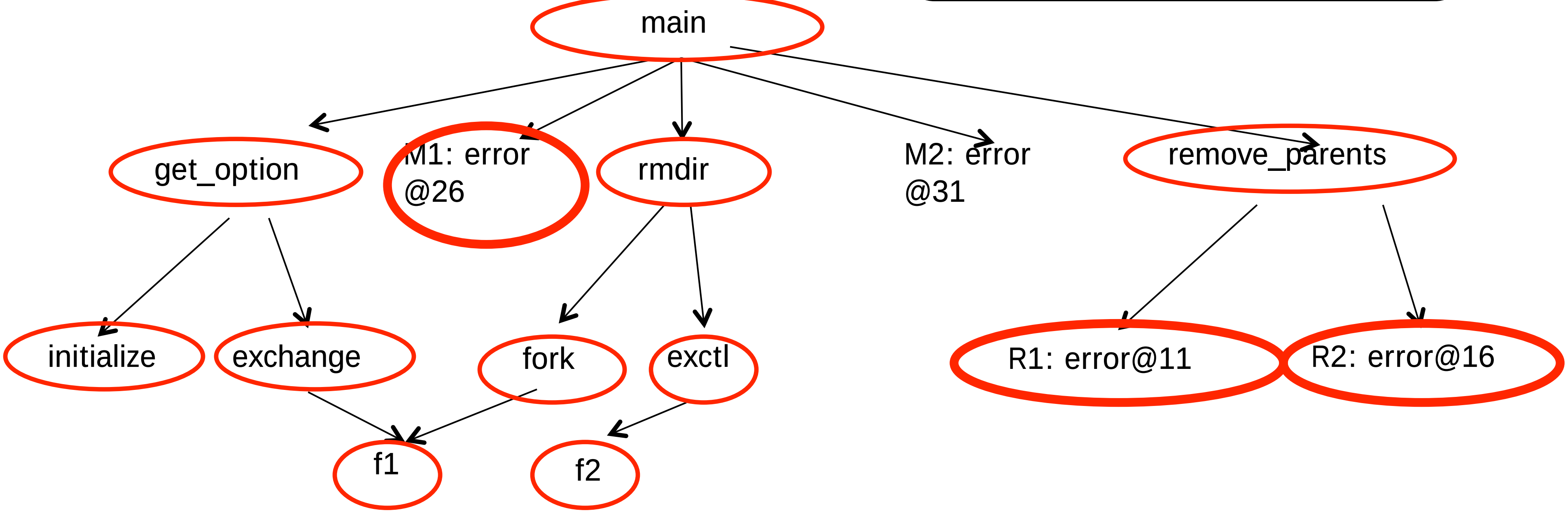


Log Parser → Path Inference → Value Inference

Challenges

- Scalability vs. Precision
  - Path Explosion
  - Most of functions irrelevant!

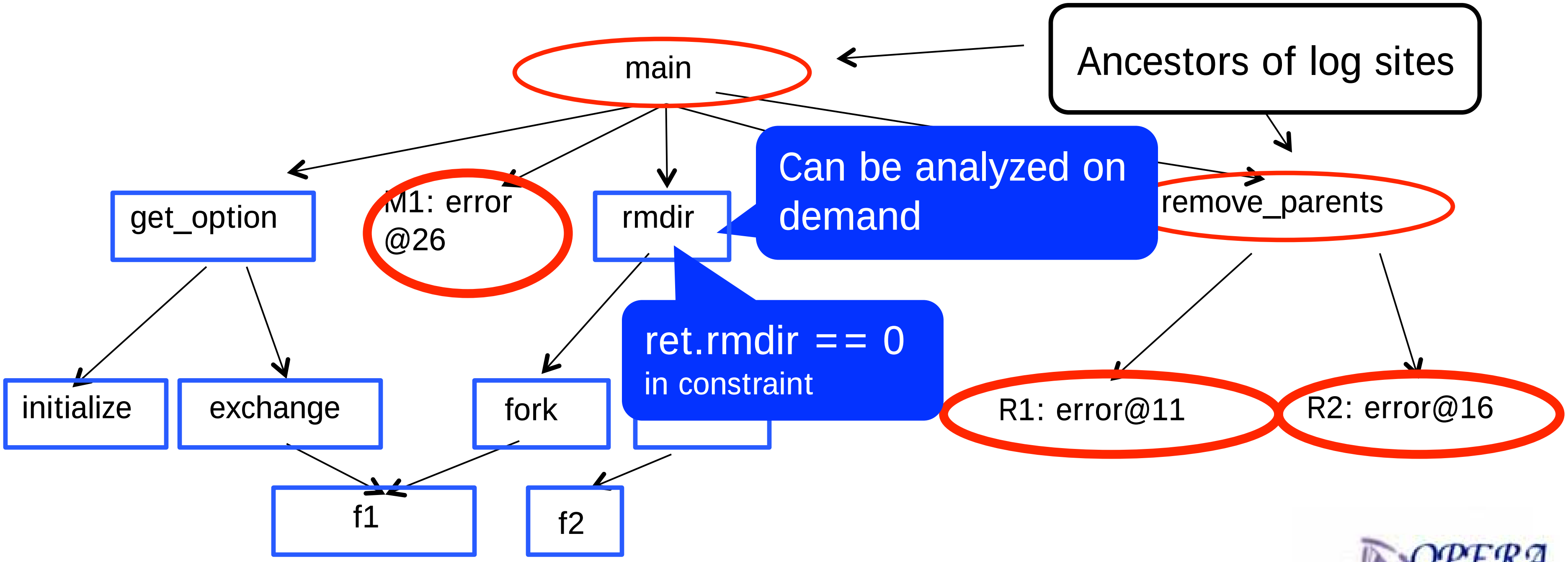
Focus only on relevant functions,
Analyze them precisely!!!



Log Parser
Path Inference
Value Inference

Solution: Log-Driven Design

- Focus on functions directly/indirectly prints log messages
  - Ancestors of log sites
  - Analyze these functions precisely down to bit level



Ancestors of log sites

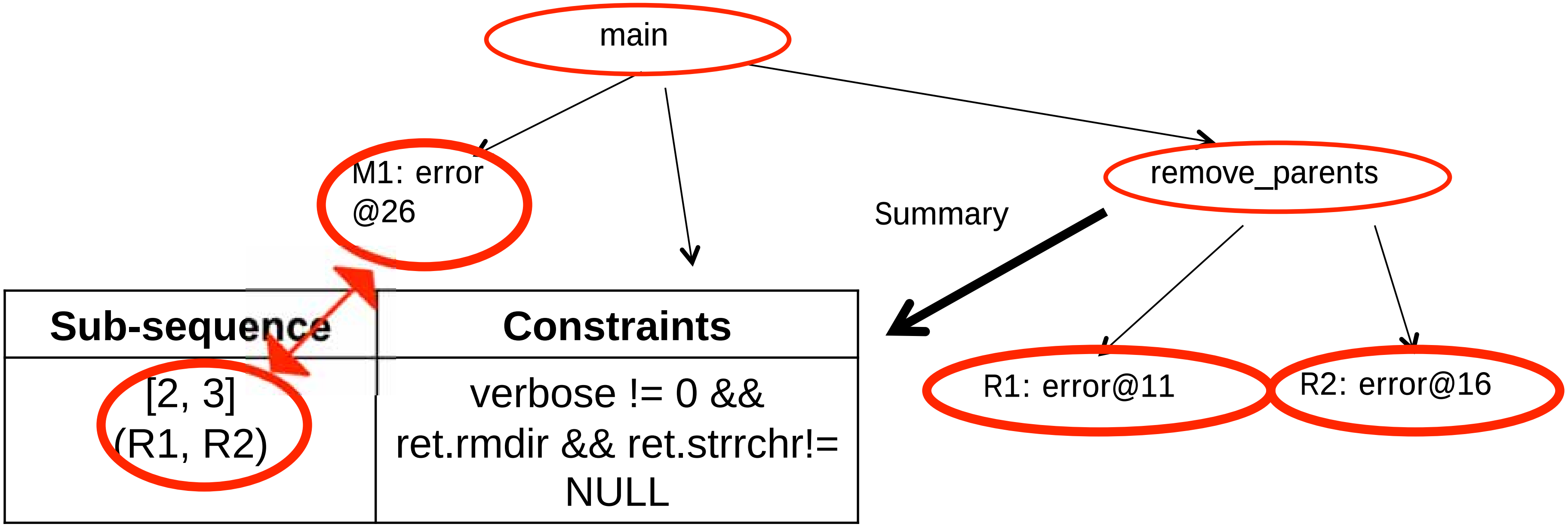
Can be analyzed on demand

ret.rmdir == 0 in constraint

Log Parser
Path Inference
Value Inference

Summary Based Analysis

- Each function is analyzed separately
  - Only one function’s representation lives in memory
  - At call-site of *f*, only *f*’s summary is used

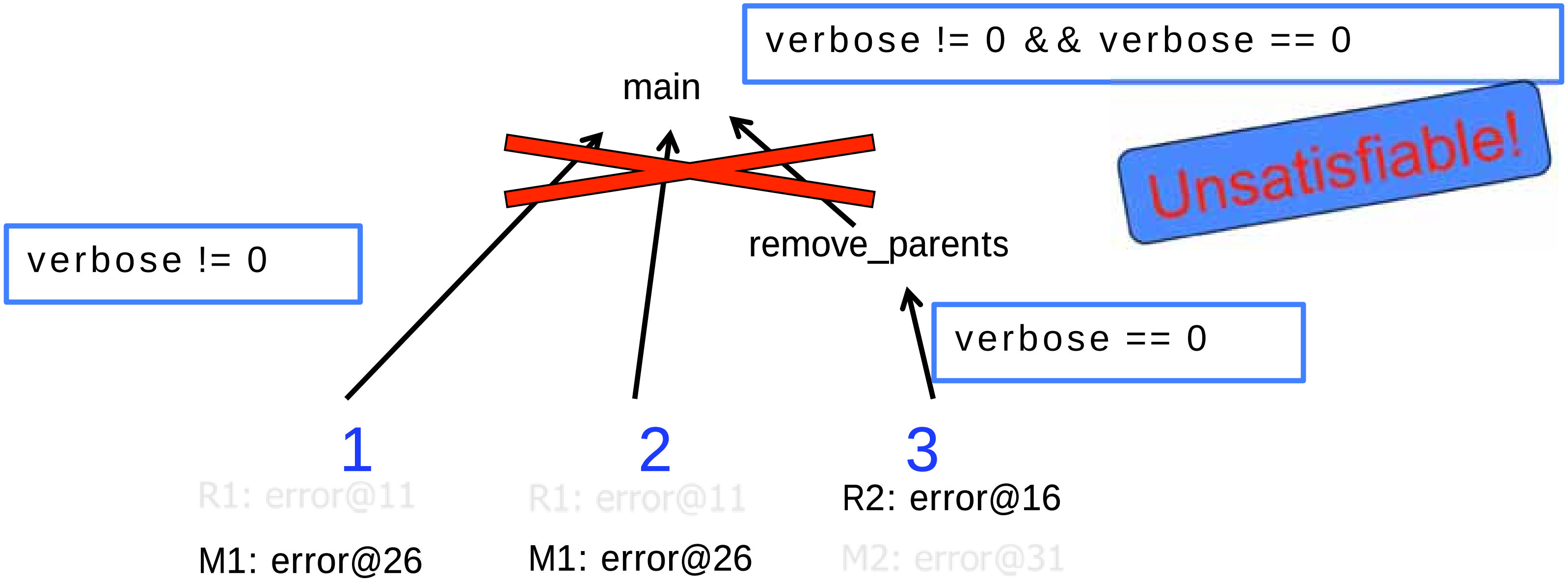


Sub-sequence	Constraints
[2, 3] (R1, R2)	verbose != 0 && ret.rmdir && ret.strrchr!= NULL

Log Parser → Path Inference → Value Inference

Constrained Sequence Matching

- Need to prune infeasible paths
  - Conditions along path as constraint formula
  - Use SAT solver to prune infeasible paths



verbose != 0 && verbose == 0

main

Unsatisfiable!

verbose != 0

remove_parents

verbose == 0

1

R1: error@11

M1: error@26

2

R1: error@11

M1: error@26

3

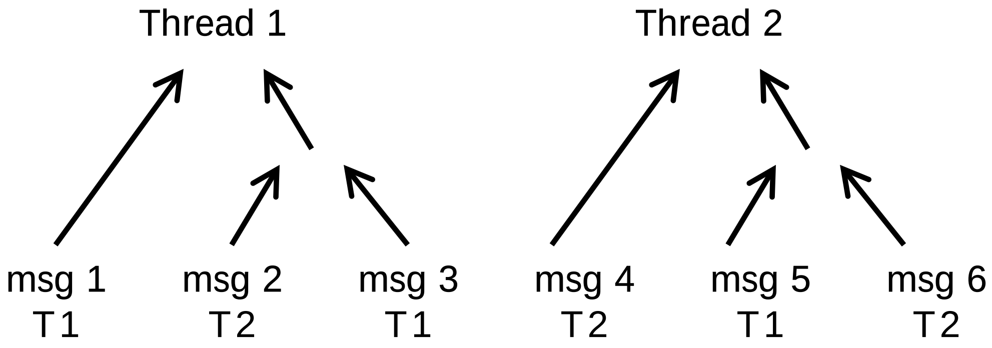
R2: error@16

M2: error@31

Log Parser
Path Inference
Value Inference

Log from multi-threaded program

- First group the messages by common thread ID
- Connects longest-continuous sub-sequence:
  - Limitation: Can’t infer across threads



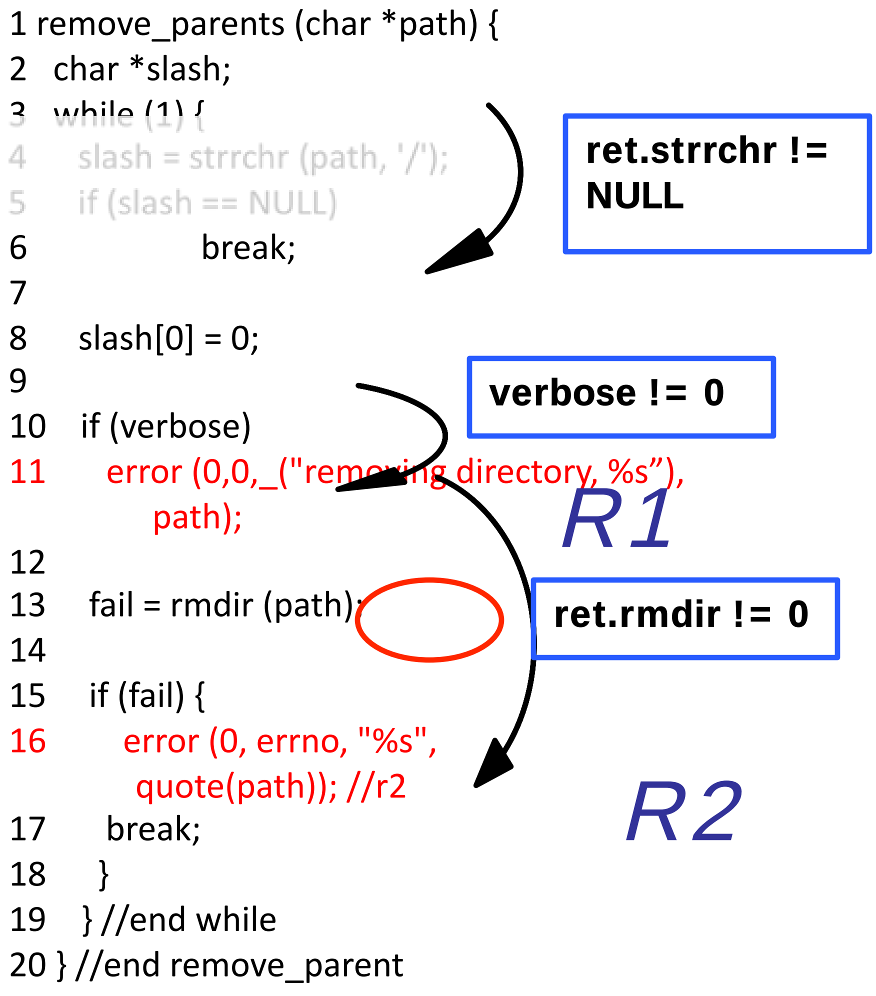
Thread ID

```
132.239.10.118 - 19248 - [28/Sep/2009:10:51:41 -0500] "CONNECT opera.ucsd.edu:443 HTTP/1.1" 405 235
132.239.10.118 - 19249 - [28/Sep/2009:10:52:00 -0500] "GET http://hq.sinajs.cn HTTP/1.1" 404 241
```

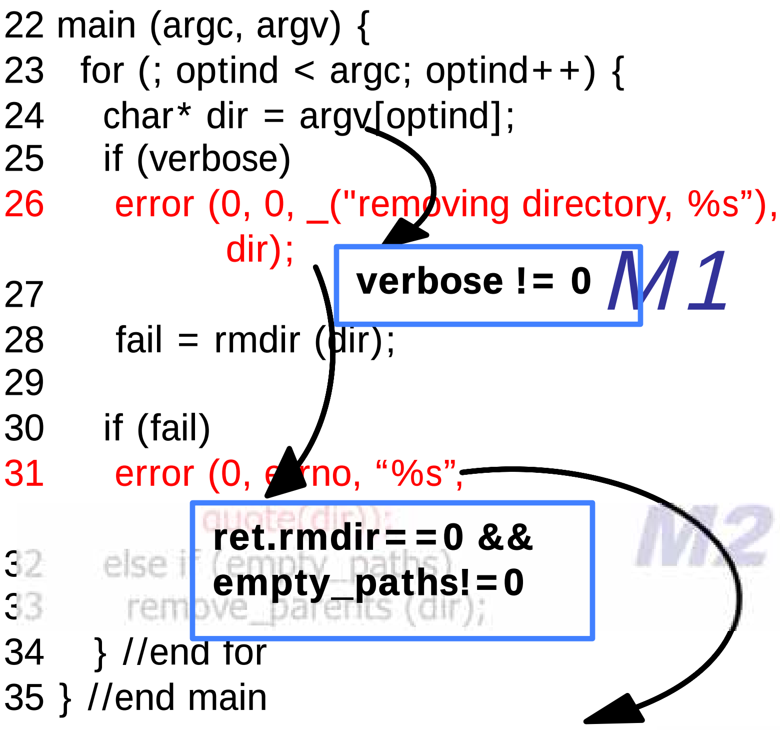
Apache HTTPD Log

rmdir: Path Inference

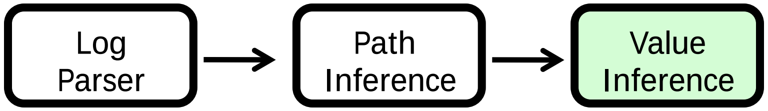
What is the value of “path”?



```
remove_parents (char *path) {
 char *slash;
 while (1) {
   slash = strrchr (path, '/');
   if (slash == NULL)
              break;

   slash[0] = 0;

   if (verbose)
     error (0,0,_("removing directory, %s”),
          path);

    fail = rmdir (path);

    if (fail) {
       error (0, errno, "%s",
        quote(path)); //r2
     break;
    }
  } //end while
} //end remove_parent
```



```
main (argc, argv) {
 for (; optind < argc; optind++) {
   char* dir = argv[optind];
   if (verbose)
    error (0, 0, _("removing directory, %s”),
          dir);

    fail = rmdir (dir);

   if (fail)
    error (0, errno, “%s”,
          quote(dir));
   else if (empty_paths)
     remove_parents (dir);
  } //end for
} //end main
```



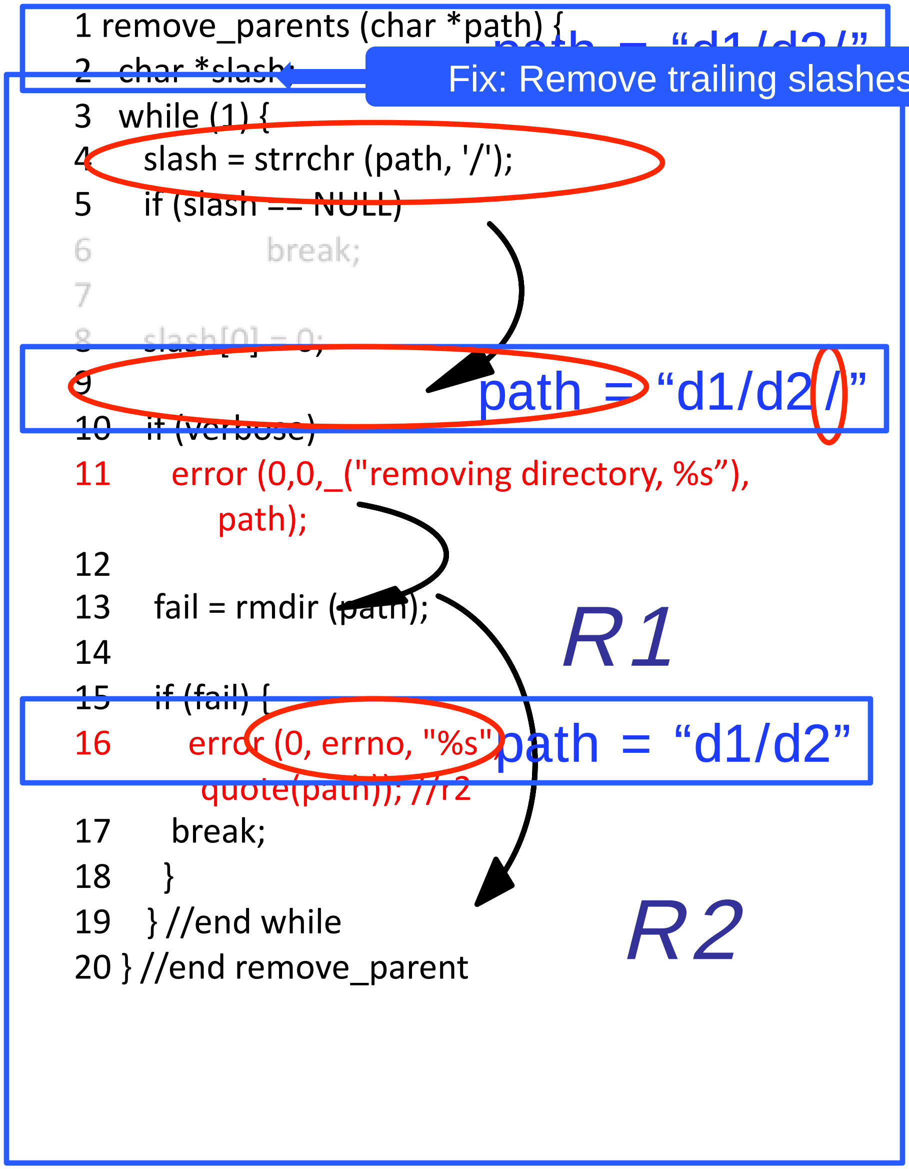
Log Parser → Path Inference → Value Inference

Value Inference

- Given a path inferred by Path Inference, use Symbolic Execution to infer the variable value
  - Scale to large application since the path is determined

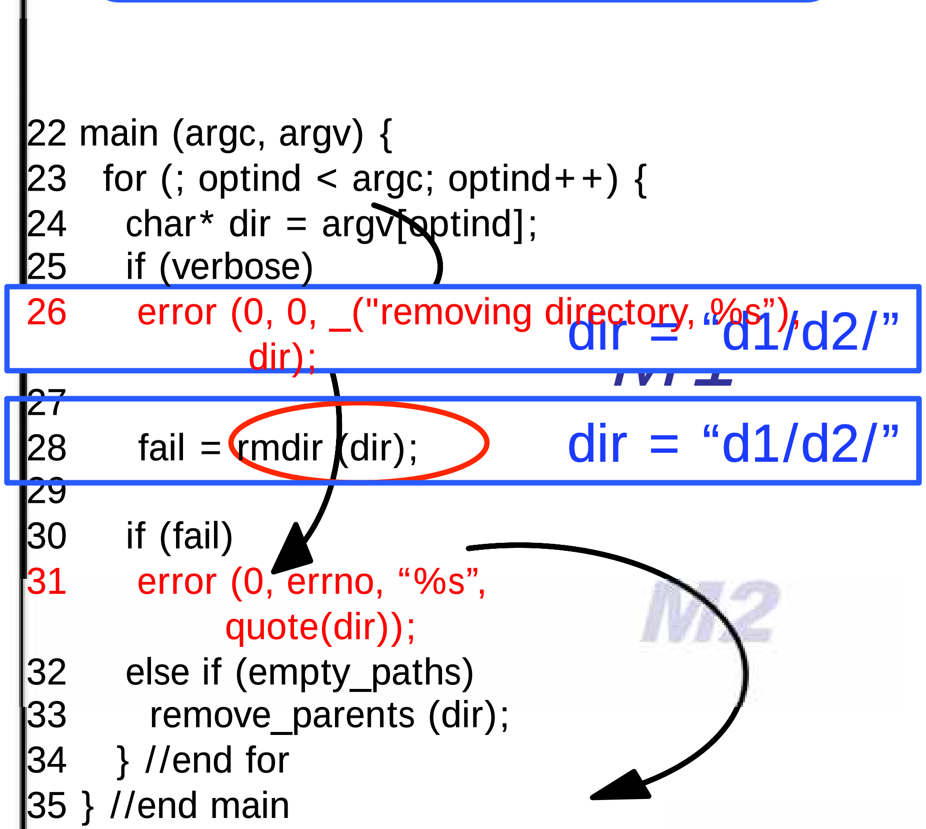
2. Infers Variable Value Information!

What is the value of “path”?



Fix: Remove trailing slashes

```
remove_parents (char *path) {
  char *slash;
  while (1) {
    slash = strrchr (path, '/');
    if (slash == NULL)
          break;

    slash[0] = 0;

    if (verbose)
      error (0,0,_("removing directory, %s”),
          path);

    fail = rmdir (path);

    if (fail) {
      error (0, errno, "%s",
          quote(path)); //r2
      break;
    }
  } //end while
} //end remove_parent
```



```
main (argc, argv) {
  for (; optind < argc; optind++) {
    char* dir = argv[optind];
    if (verbose)
      error (0, 0, _("removing directory, %s”),
          dir);

      fail = rmdir (dir);

    if (fail)
      error (0, errno, “%s”,
          quote(dir));
    else if (empty_paths)
      remove_parents (dir);
   } //end for
} //end main
```

Implementation

- Built on Saturn static analysis framework
  - Models C program semantic precisely
  - Precise intra-procedural data-flow
- Write analysis in CALYPSO Logic Programming Language

Evaluation

More case studies in paper!

8 Real World Failures Reported by Users

App.	Type	#MSG	LOC	Description
rmdir	Bug	3	18K	missing to remove trailing slashes with –p option
ln			20K	missing condition check for –target-directory option
rm			23K	missing condition check causing –i behaves like -ir
CVS 1	Config	3	148K	incorrectly setting the permission for locking directory
CVS 2	Config	2	148K	incompatibility between application and config file
HTTPD	Bug	1,309	317K	incorrectly handles EOF in response stream when set up as proxy server
Squid	Bug	197	69K	Treating certain icon files wrongly by not caching them
TAR	Bug	2	79K	Semantic bug causing tar fail to update a non-existing tarball

Server Applications

SherLog successfully inferred all diagnostic information!

Evaluation (cont.)

App.	Log Parser		Path Inference	
	Regex	Log Sites	# of paths	Msgs
rmdir	4	10	2	3 (3)
ln	17	23	1	2 (2)
rm	17	25	1	4 (4)
CVS 1	695	1,173	1	2 (3)
CVS 2	695	1,173	1	1 (2)
HTTPD	997	1,259	1	10 (1,309)
Squid	1,134	1,209	1	108 (197)
Tar	171	228	5	1 (2)

Performance

App.	Parser	Path		Value	
	Time	Time	Memory	Time	Memory
rmdir	0.02s	2.25m	174 MB	15.54s	116 MB
ln	0.02s	2.32m	194 MB	37.75s	165 MB
rm	0.01s	2.00m	511 MB	38.87s	123 MB
CVS 1	0.32s	39.56m	1,317 MB	188.53s	323 MB
CVS 2	0.19s	38.96m	1,322 MB	39.19s	232 MB
HTTPD	0.67s	28.38m	321 MB	19.23s	217 MB
Squid	0.81s	38.02m	1,520 MB	22.01s	252 MB
Tar	0.08s	6.55m	210 MB	29.14s	155 MB

Related Work

- Core-dump analyzer:
  - PSE [ManevichSIGSOFT2004], WER [GlerumSOSP09]
- Log Analysis:
  - statistic techniques: [CohenSOSP05] [XuSOSP09] [JiangTHESIS09]
  - Distributed system Causal path: [AguileraSOSP03] [BarhamOSDI04]
- Error Diagnosis without error reproduction:
  - Program slicer [HorwitzPLDI88] [ChenTACAS09]
  - Coorporative Bug Isolation [LiblitPLDI2003] [ChilimbiICSE2009]
  - Model checking/symbolic execution: [BallSIGPLAN2003] [CadarOSDI08]

Limitations

- Assume log messages are relevant to failure
- Do not infer across thread
- Do not infer across function pointer

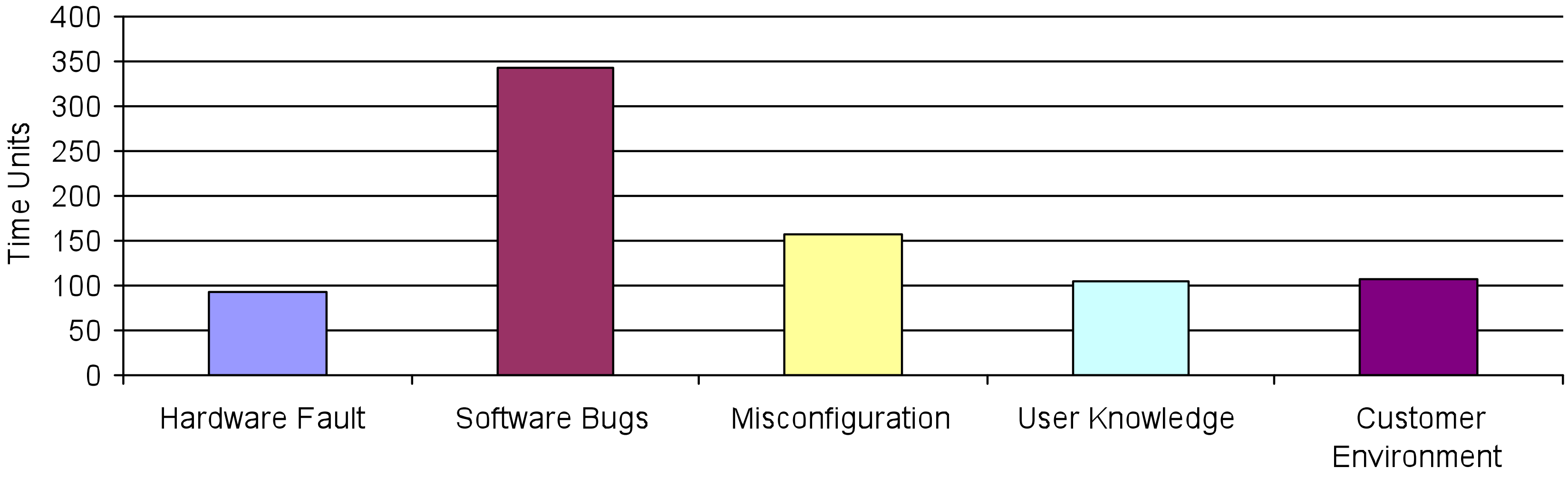
- What failures can not benefit from SherLog
  - Without log msgs
  - With long error propagation

Conclusions & Future Work

- Customer problem troubleshooting is a critical problem
  - Automation is needed and possible

- The next "zigzag"---or an ending hook to be invited again ☺
  - How to write software so it is easy to diagnose?
  - More to report next time

Troubleshooting time

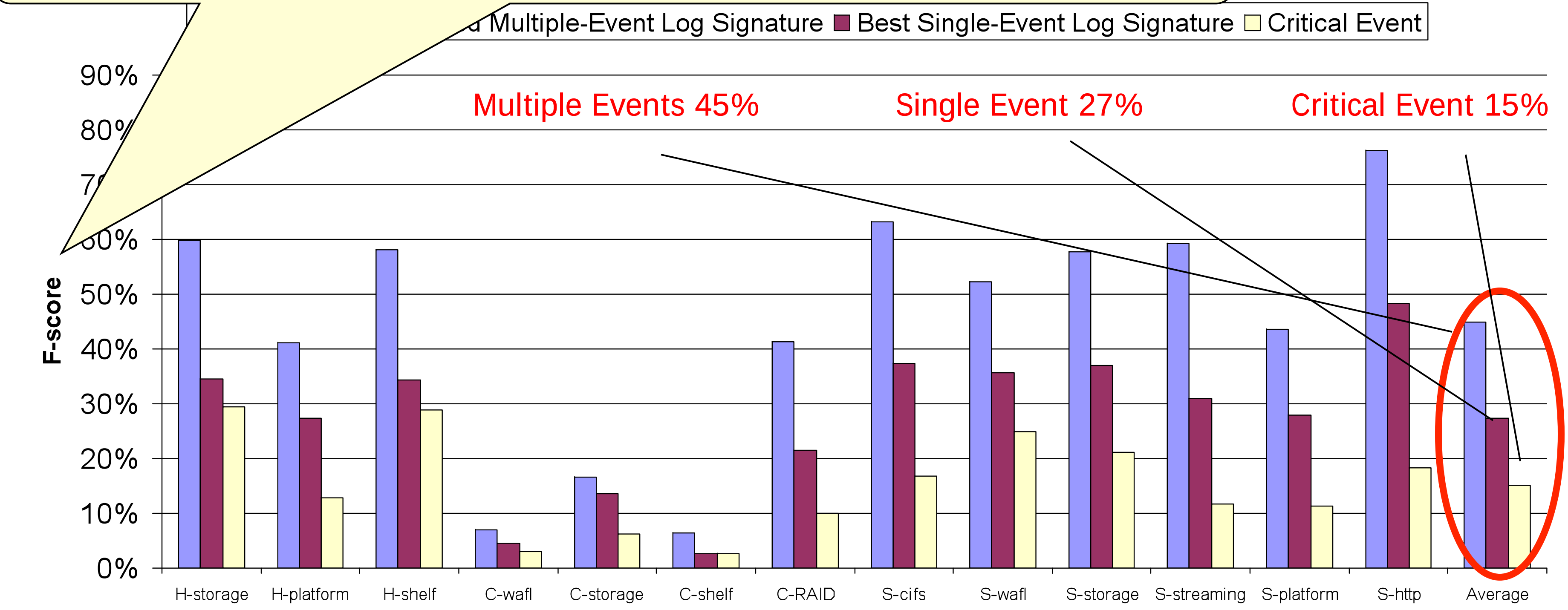


- Software bugs take longer time to troubleshoot.
- For all categories, troubleshooting is time-consuming.

More log events are more useful

How well the signature can uniquely identify cause?

F-score = 2 * Precision * Recall / (Precision + Recall)



Multiple Events 45%

Single Event 27%

Critical Event 15%

- Critical event alone is not enough.
- Using more log events can bring better accuracy.