Non-intrusive Performance Profiling for Entire Software Stacks based on *The Flow Reconstruction Principle*

Xu Zhao, Kirk Rodrigues, Yu Luo, Ding Yuan, Michael Stumm

UNIVERSITY OF
TORONTO

Motivations

- Debugging distributed system stacks is difficult

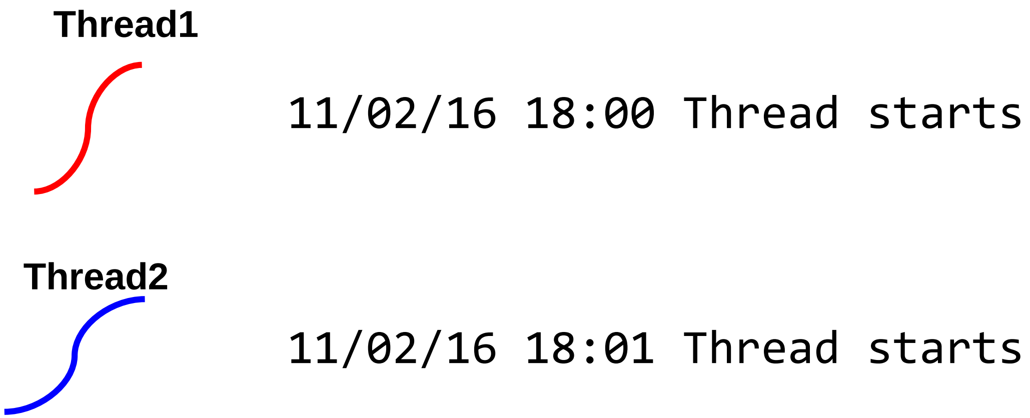
- Existing tools are limited
  - **Intrusive**: manually built-in domain knowledge
  - **Machine learning** on logs: unable to reconstruct execution flow
  - **Static analysis [Zhao OSDI’14]**: cannot cross different software components

- Use programmers’ intuition in postmortem execution analysis

The Flow Reconstruction Principle

Programmers log sufficient information so that they can **reconstruct the execution flow in distributed stacks**

- Three rules
  - Log **events** at critical points in the control path
  - Log **object identifiers** to differentiate concurrent runs
  - Log **sufficient identifiers** to unambiguously identify an object



Thread1

```
11/02/16 18:00 Thread starts
```

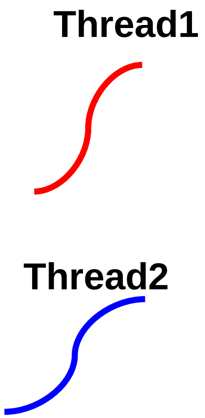
Thread2

```
11/02/16 18:01 Thread starts
```

The Flow Reconstruction Principle

Programmers log sufficient information so that they can **reconstruct the execution flow in distributed stacks**

- Three rules
  - Log **events** at critical points in the control path
  - Log **object identifiers** to differentiate concurrent runs
  - Log **sufficient identifiers** to unambiguously identify an object

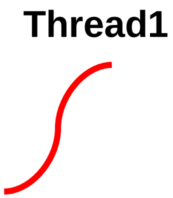


```
11/02/16 18:00 Thread 1 starts

11/02/16 18:01 Thread 2 starts
```

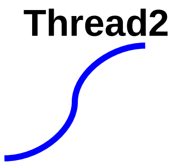
The Flow Reconstruction Principle

Programmers log sufficient information so that they can **reconstruct the execution flow in distributed stacks**

- Three rules
  - Log **events** at critical points in the control path
  - Log **object identifiers** to differentiate concurrent runs
  - Log **sufficient identifiers** to unambiguously identify an object



```
11/02/16 18:00 [Process 2355] Thread 1 starts
```

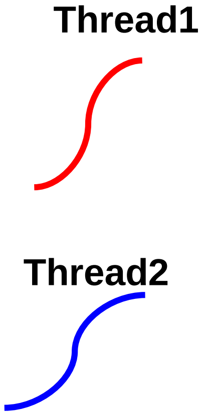


```
11/02/16 18:01 [Process 1224] Thread 2 starts
```

The Flow Reconstruction Principle

Programmers log sufficient information so that they can **reconstruct the execution flow in distributed stacks**

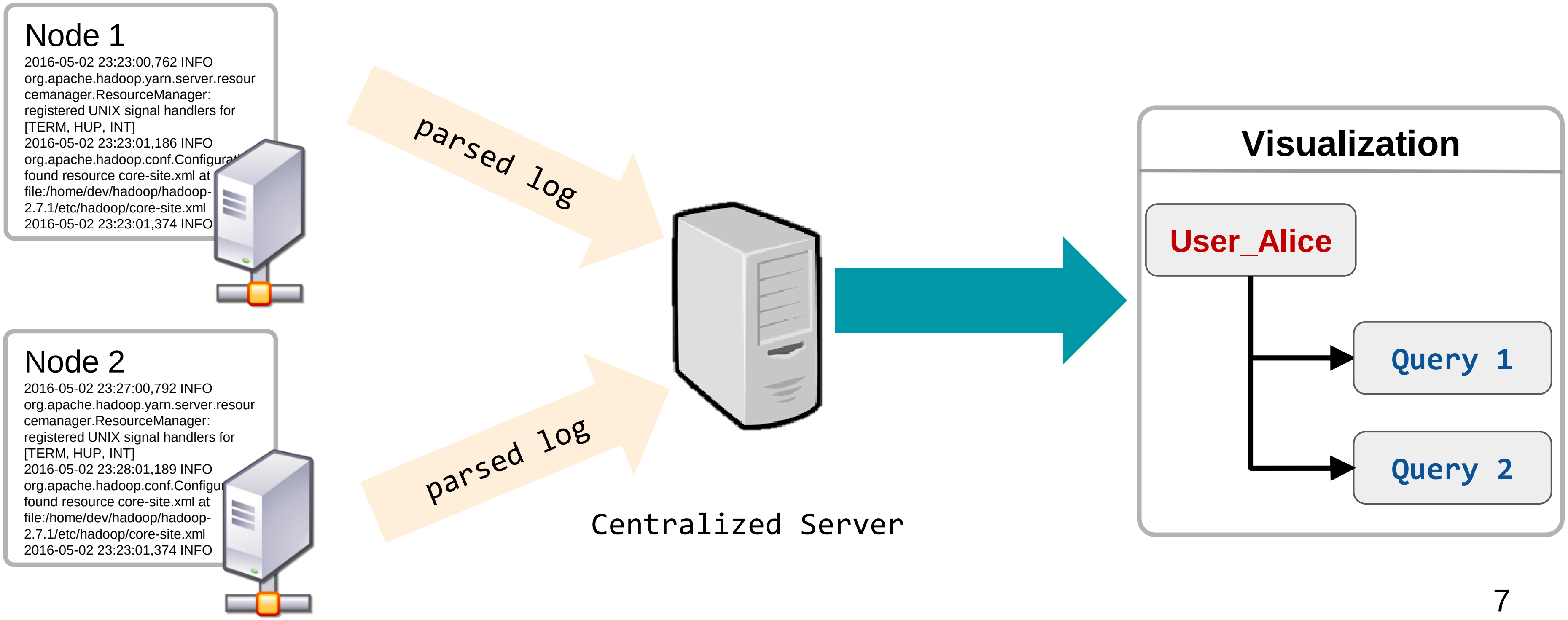
- Three rules
  - Log **events** at critical points in the control path
  - Log **object identifiers** to differentiate concurrent runs
  - Log **sufficient identifiers** to unambiguously identify an object



```
11/02/16 18:01 Process 2355 starts a Thread

11/02/16 18:01 Process 1224 starts a Thread

11/02/16 18:01 Thread 2 starts

11/02/16 18:02 Thread 1 starts
```

Stitch: A Non-intrusive Profiler for Entire Server Stacks

- Works without any built-in domain knowledge
- Purely based on IDs, throws away constant text
- Reconstructs **objects**, infers their lifetimes and hierarchies



```
2016-05-02 23:23:00,762 INFO
org.apache.hadoop.yarn.server.resour
cemanager.ResourceManager:
registered UNIX signal handlers for
[TERM, HUP, INT]
2016-05-02 23:23:01,186 INFO
org.apache.hadoop.conf.Configurati
found resource core-site.xml at
file:/home/dev/hadoop/hadoop-
2.7.1/etc/hadoop/core-site.xml
2016-05-02 23:23:01,374 INFO
```

```
2016-05-02 23:27:00,792 INFO
org.apache.hadoop.yarn.server.resour
cemanager.ResourceManager:
registered UNIX signal handlers for
[TERM, HUP, INT]
2016-05-02 23:28:01,189 INFO
org.apache.hadoop.conf.Configu
found resource core-site.xml at
file:/home/dev/hadoop/hadoop-
2.7.1/etc/hadoop/core-site.xml
2016-05-02 23:23:01,374 INFO
```

Outline

- **Real world example**
- Design of Stitch
- User study and correctness evaluation

Real World Example

- User study on 14 programmers
- We reproduced a real world performance anomaly: YARN-4610
  - Symptom: user Alice reports her Hive query takes longer than expected
- Separate programmers into 2 groups: debug using raw log or with Stitch
  - 45 mins time limit

Debugging with Raw Log

- Only 2 users using raw log can diagnose within the time limit
- Reading each log message is infeasible: over **5,000** lines of logs
- Users take two approaches: **bottom-up** and **top-down**
  - **Bottom-up** results in a wild goose chase

```
$ grep “ERROR\|WARN” -R logs/

./node0/userlogs/application_1462245782384_0024/container_1462245782384_0024_01
_000576/syslog:2016-05-02 23:46:37,010 ERROR [Thread-50]
org.apache.hadoop.mapreduce.v2.app.rm.RMContainerAllocator: Could not
deallocate container for task attemptId attempt_1462245782384_0024_r_000001_1

... [707 more]
```

Debugging with Raw Log

- Only 2 users using raw log can diagnose within the time limit
- Reading each log message is infeasible: over **5,000** lines of logs
- Users take two approaches: **bottom-up** and **top-down**
  - **Bottom-up** results in a wild goose chase
  - **Top-down** gets closer, but still cannot diagnose failure within time limit
    - Determine slowest map task

Debugging with Stitch

- All 7 Stitch users succeeded, spending 13 mins on average
- Stitch speeds-up debugging time by a factor of **3.5**
- Demo

Outline

- Real world example
- **Design of Stitch**
- User study and correctness evaluation

Client: Log Parsing

- Detect log printing processes and log files
  - Periodically read /proc
- Log parsing: separate dynamic and constant parts of the raw log
  - Extract **string constants** from binaries, match against the log
  - Filter non-identifiers using a blacklist (“memsize”, “ms”, “progress”, etc.)

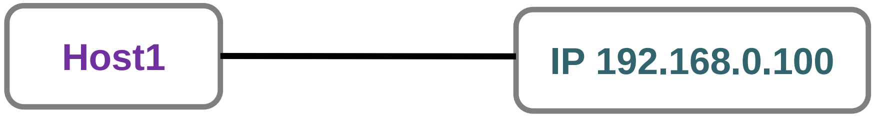
```
16/04/02 00:58 MongoDB starting: pid=22925 port=27017 dbpath=/var/lib/mongodb
```

```
// Code that prints this msg:
l << "MongoDB starting : pid=" << pid << " port=" <<
serverGlobalParams.port << " dbpath=" << storageGlobalParams.dbpath;
```

Parsed Log	Timestamp	Identifier	Type
	16/04/02 00:58	22925	pid
		27017	port
		/var/lib/mongodb	filePath

Server: Identifier Relations

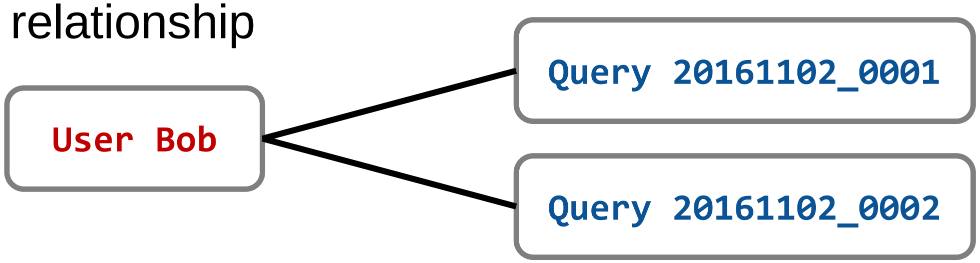
- 1:1 relation
  - Can be used interchangeably



Host1

IP 192.168.0.100

- 1:n relation
  - Hierarchical relationship

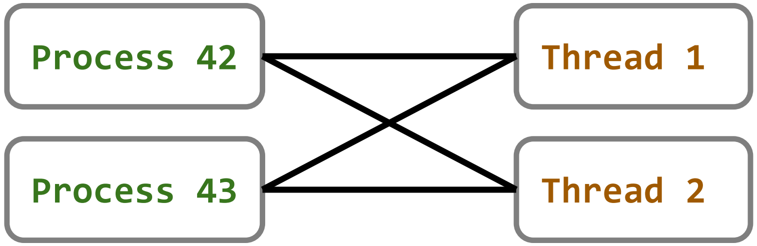


User Bob

Query 20161102_0001

Query 20161102_0002

- m:n relation
  - Combination is required to unambiguously identify an object



Process 42

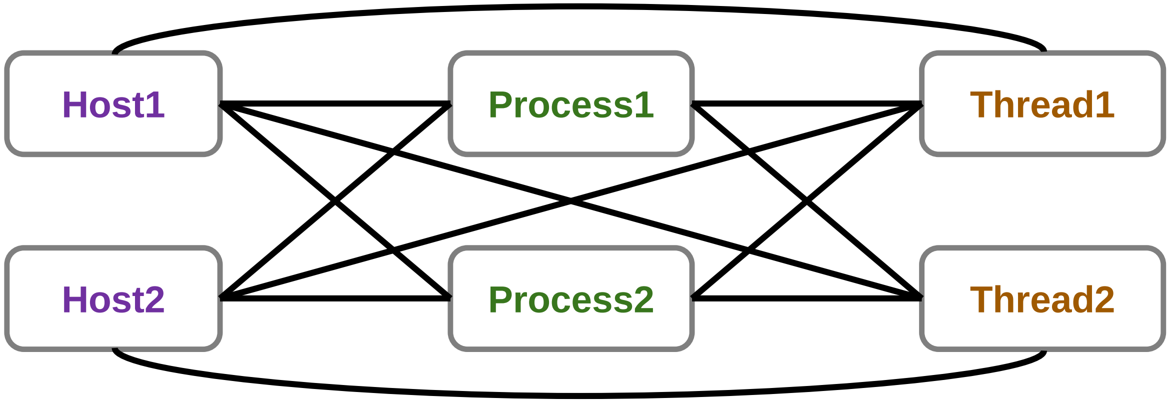
Process 43

Thread 1

Thread 2

m:n Relation

- Example: **Host**, **Process** and **Thread**

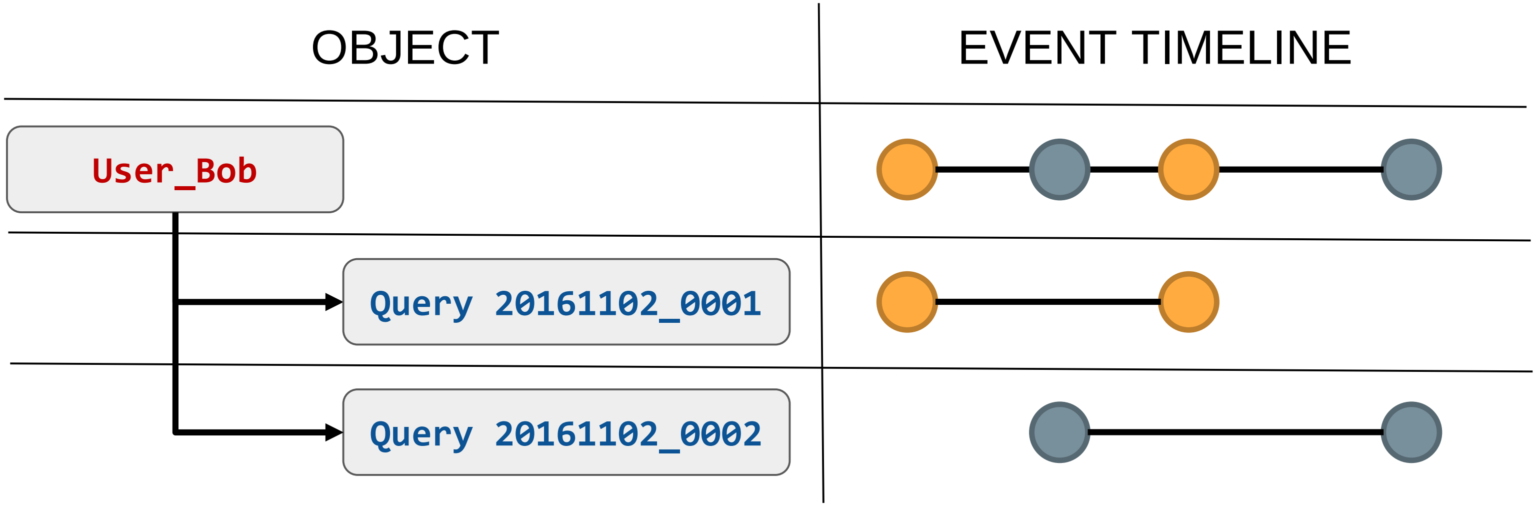


- Not every combination is meaningful
  - Meaningful ones: {**Host**}, {**Host, Process**}, {**Host, Process, Thread**}
  - Meaningless combination: {**Host, Thread**}
- Developers only log the meaningful combinations
  - Developers will never log {**Host**, **Thread**} without **Process**

Lifetime and Hierarchy

- Lifetime of objects inferred from event timestamps
- Hierarchy is inferred from 1:n relation

```
16/04/02 00:58 User Bob creates Hive query: Query 20161102_0001
16/04/02 01:03 User Bob creates Hive query: Query 20161102_0002
16/04/02 01:06 User Bob Hive query Query 20161102_0001 finishes
16/04/02 01:10 User Bob Hive query Query 20161102_0002 finishes
```



OBJECT

EVENT TIMELINE

User_Bob

Query 20161102_0001

Query 20161102_0002

Outline

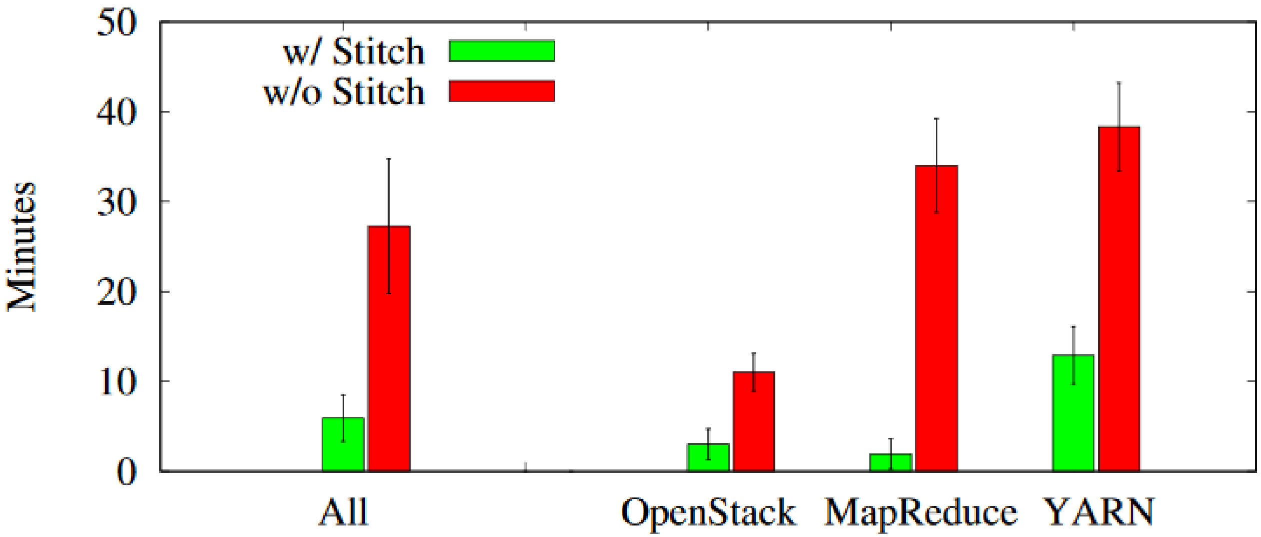
- Real world example
- Design of Stitch
- **User study and correctness evaluation**

Evaluation

- How much time does Stitch save in debugging?
- How accurate is Stitch in identifying objects?
- Do real-world systems follow the Flow Reconstruction Principle?

User Study Evaluation

- 14 users work with 3 cases
  - *Performance debugging* – An anomaly caused by a YARN scheduler bug
  - *Bottleneck identification* – Identify slow node in a MapReduce job
  - *Hierarchy identification* – OpenStack components involved in request



- At least **4.6x** faster debugging with Stitch

Correctness Evaluation

- Hive, Spark and OpenStack workload: 200 nodes run for 24 hours
- Production workload: 24-node cluster running for four months
- Together there are 19 software components

System	Objects	Object Types	Object Accuracy
Hive	295,042	32	90%
Spark	192,969	31	94%
OpenStack	214,882	14	100%
Production	8,141	24	100%
Total	711,034	101	**96%**

- Stitch correctly identifies **96%** of all objects in four software stacks

Limitations

- Stitch does not capture causal relations
- Stitch's efficacy relies on developers following the Flow Reconstruction Principle
  - Violations of the principle will mislead human debugging
- Stitch's accuracy is sensitive to the quality of identifiers and type-extraction
  - Log parsing is a solved problem
  - Industrial solutions exist, e.g., Splunk, VMWare LogInsight, LogStash

Related Work

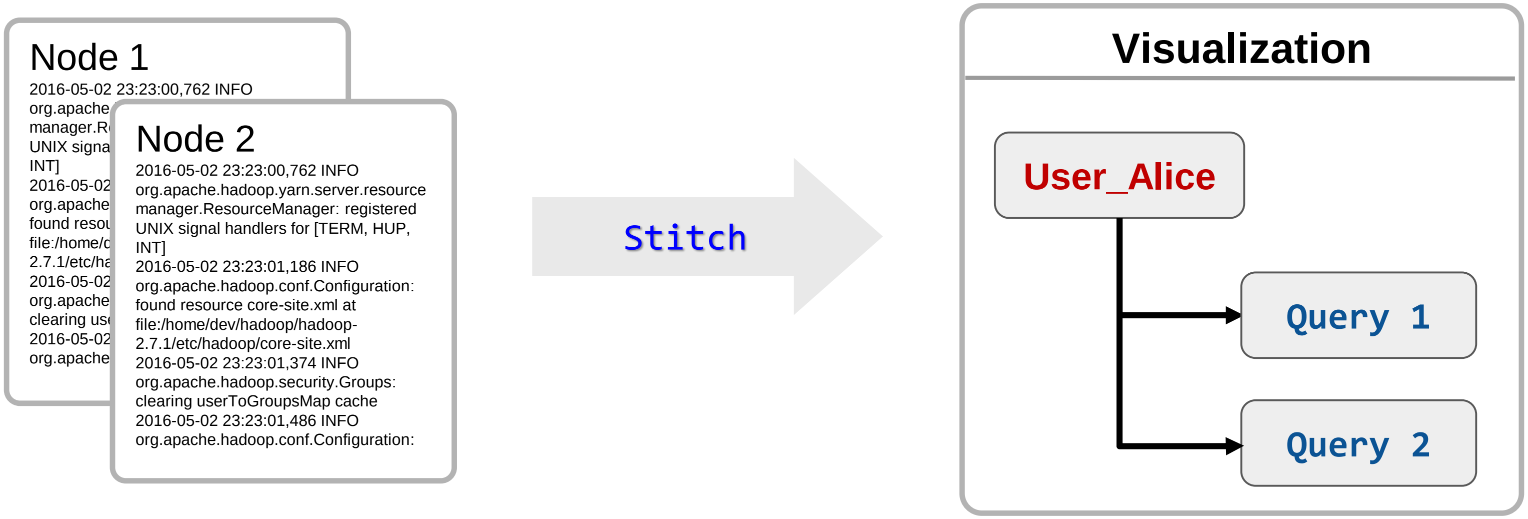
- **Intrusive tools [Aguilera SOSP'03] [Barham OSDI'04] [Chanda EuroSys'07]**
  - Captures causal relationships
  - Built-in domain knowledge
- **Static analysis tools [Zhao OSDI'14] [Yuan ASPLOS'10]**
  - Captures causal relationships
  - Cannot cross layers
- **Machine learning solutions [Xu SOSP'09] [Nagaraj NSDI'12] [Yu ASPLOS'16]**
  - Cannot reconstruct execution flow

Conclusions

The Flow Reconstruction Principle

- Log **events** at critical points in the control path
- Log **object identifiers** to differentiate concurrent runs
- Log **sufficient identifiers** to unambiguously identify an object

Stitch: non-intrusive profiler for distributed stacks



Node 1

```
2016-05-02 23:23:00,762 INFO
org.apache
manager.R
UNIX signa
INT]
2016-05-02
org.apache
found resou
file:/home/d
2.7.1/etc/ha
2016-05-02
org.apache
clearing us
2016-05-02
org.apache
```

Node 2

```
2016-05-02 23:23:00,762 INFO
org.apache.hadoop.yarn.server.resource
manager.ResourceManager: registered
UNIX signal handlers for [TERM, HUP,
INT]
2016-05-02 23:23:01,186 INFO
org.apache.hadoop.conf.Configuration:
found resource core-site.xml at
file:/home/dev/hadoop/hadoop-
2.7.1/etc/hadoop/core-site.xml
2016-05-02 23:23:01,374 INFO
org.apache.hadoop.security.Groups:
clearing userToGroupsMap cache
2016-05-02 23:23:01,486 INFO
org.apache.hadoop.conf.Configuration:
```

Q & A

Source of Inaccuracy

- Hive ID: *Stage-* : violating Flow Reconstruction Principle
  - *Stage-* and *Query ID* in separate log messages
- Spark ID: *inode* and *HostName* - small scale of workload
  - *inode* and *HostName* are 1:1 instead of m:n
  - Appear together in an error log message