
# A Graphical representation for identifier structure in application logs

Ari Rabkin, Wei Xu, Avani Wildani, Armando Fox, David Patterson and Randy Katz

SLAML
October 3, 2010


# Motivation & Summary

- Log analysis is fundamentally constrained by the information content of the underlying logs
- Need tools to help developers spot flaws in their loging
- We propose a compact graph-based representation for log structure
- Differs from previous work in analyzing logging behavior, not logs of particular executions


# Focus on identifers

- We focus on *identifiers* in logs
  - Variable fields that refer to entities in a system.
  - Can be operationally defined as variable fields with increasingly many possible strings [Xu 09]
- Previous work has modeled logs as sets of concurrent state machines. [Fu 09, Tan 08]
  - Identifiers tie together messages that correlate to the same state machine


# Some defects

- Imagine a transaction processing system.

```
3:45 Starting transaction t123
3:46 Transaction failed
3:50 Starting transaction t123
3:51 Finished trans that was
started at 3:50.
```


# Missing IDs

- Imagine a transaction processing system.

```
3:45 Starting transaction t123
3:46 Transaction failed
3:50 Starting transaction t123
3:51 Finished trans that was
started at 3:50.
```

No ID


# Inconsistent IDs

- Imagine a transaction processing system.

```
3:45 Starting transaction t123
3:46 Transaction failed
3:50 Starting transaction t123
3:51 Finished trans that was
started at 3:50.
```

← Inconsistent identification


# Ambiguous IDs

- Imagine a transaction processing system.

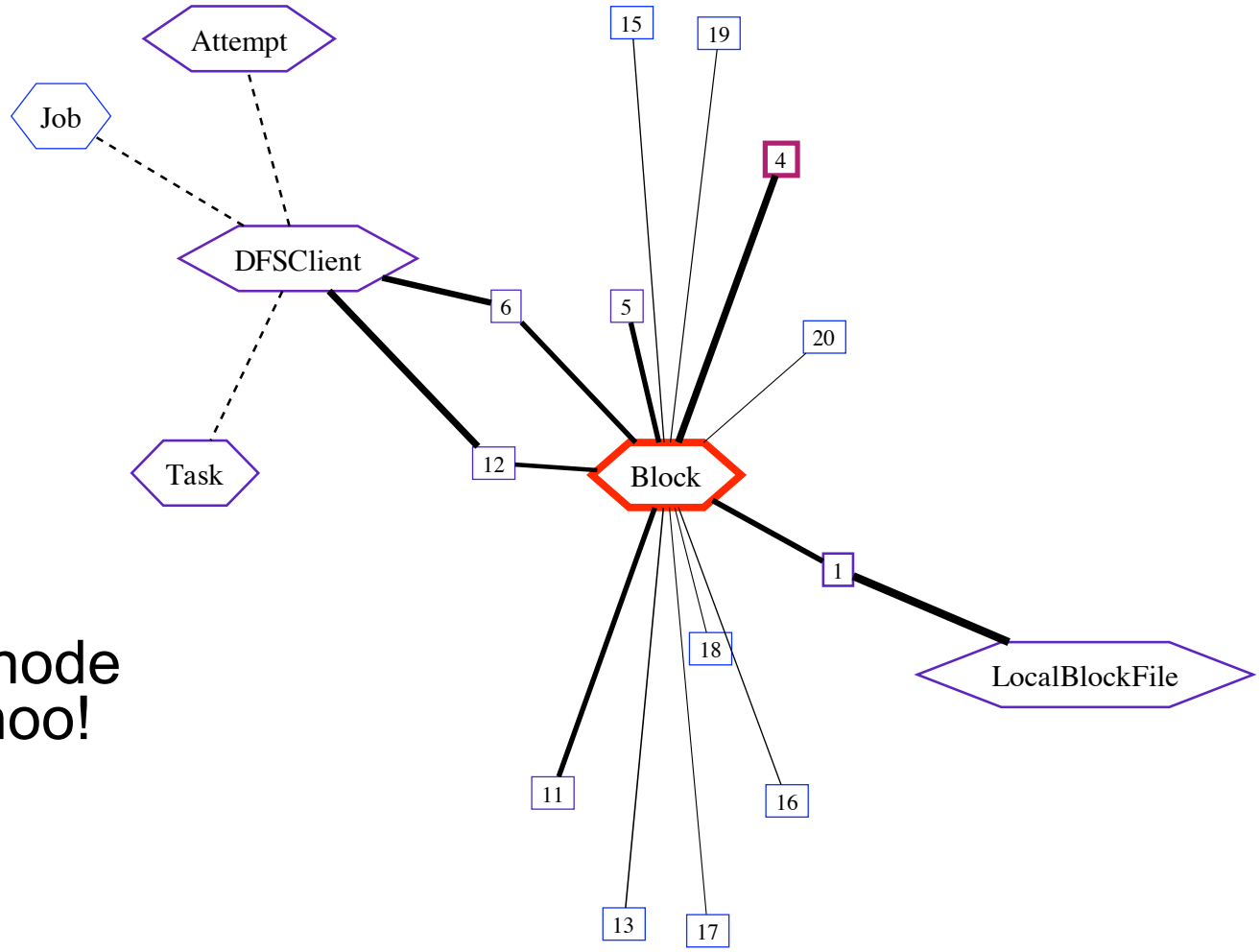
```
3:45 Starting transaction t123
3:46 Transaction failed
3:50 Starting transaction t123
3:51 Finished trans that was
started at 3:50.
```

Ambiguous identification


# Goals

- Seek a compact representation for logs
- Make common logging flaws visible
- Facilitate comparison across related logs
- Not depend on details of particular execution traces

# A real example




Hadoop datanode logs from Yahoo! M45 cluster


# Definitions

- Definitions:
  - A *log message* is a string.
  - Each log message is associated with a specific message type.
  - All messages of a type are structurally identical. (same set of identifier fields)
  - Identifiers belong to *identifier classes.*


# Assumptions

- Assumptions
  - Have representative sample of logs
  - Can find message type from message
  - Can extract identifiers from messages
  - Have identifier class for each identifier field in a message type

# Core structure

- Ex: `Starting task` *`t123`* `on node` *`n`*

Task ID  Host name


Formally: a graph with
V = { identifier classes} U {message types}
E = { (i,m) | message *m* includes an identifier of class *i*}

# Subsumption

- Sometimes, one identifier includes another.
- Model this by adding a graph edge between two identifiers if one inclues another.
- Call this *subsumption*
  - E.g., URLs subsume host names

# Frequency

- Can encode frequency information on diagram


- Scaled relative to most-frequent message or identifier
- γ-correction: scale by sqrt(freq / Max(freq))


# Ubiquity

- Can show information about joint ID-message statistics
- Want to distinguish (ab)normal messages
- Defn:
  The ubiquity of identifier class C for message type T is the fraction of identifiers belonging to class C appearing in messages of type T.
- Orthogonal to frequency of message

# Drawing ubiquity

- Line thickness proportional to ubiquity

# Diagramming defects

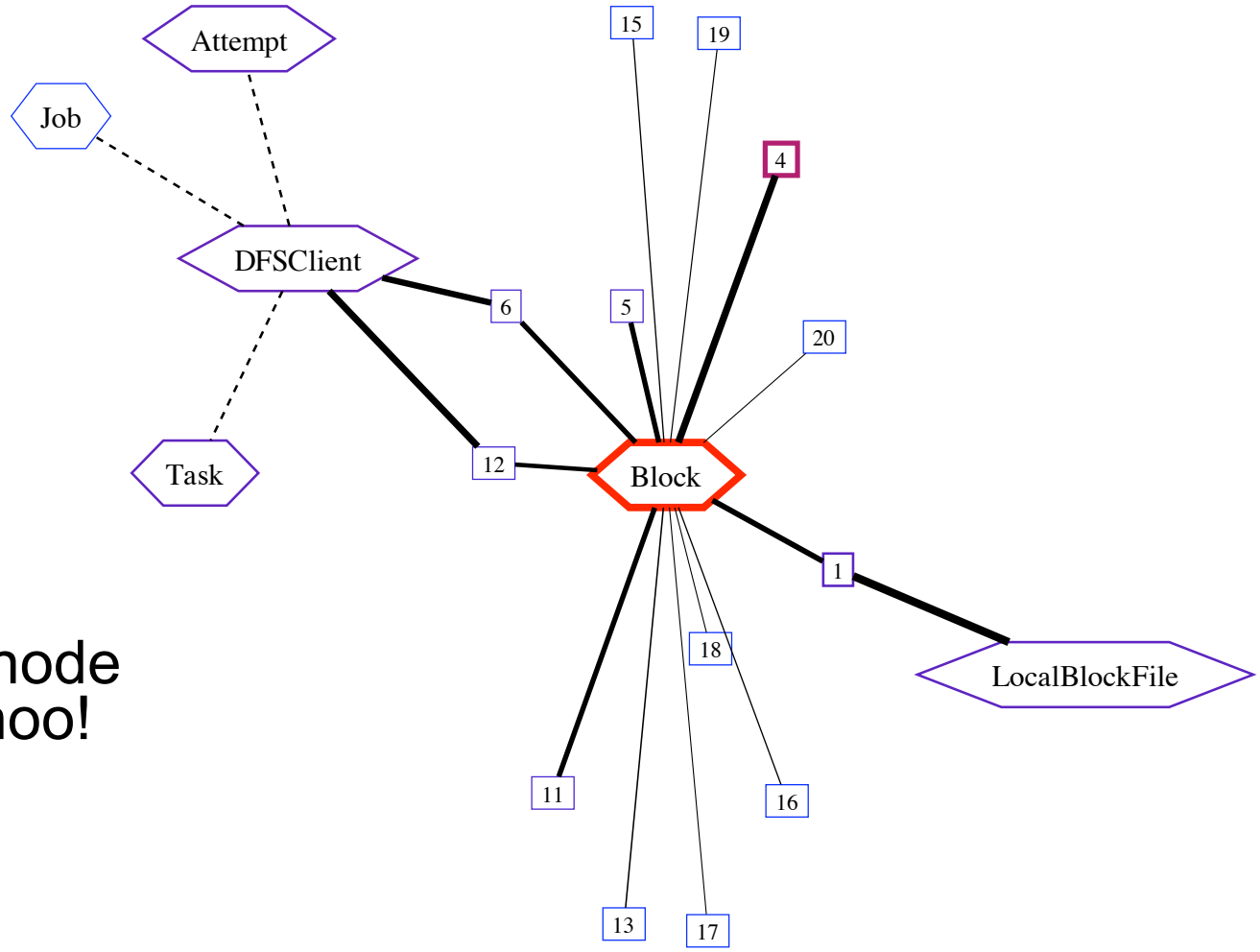
- Missing ID:


- Inconsistent IDs


# Our prototype

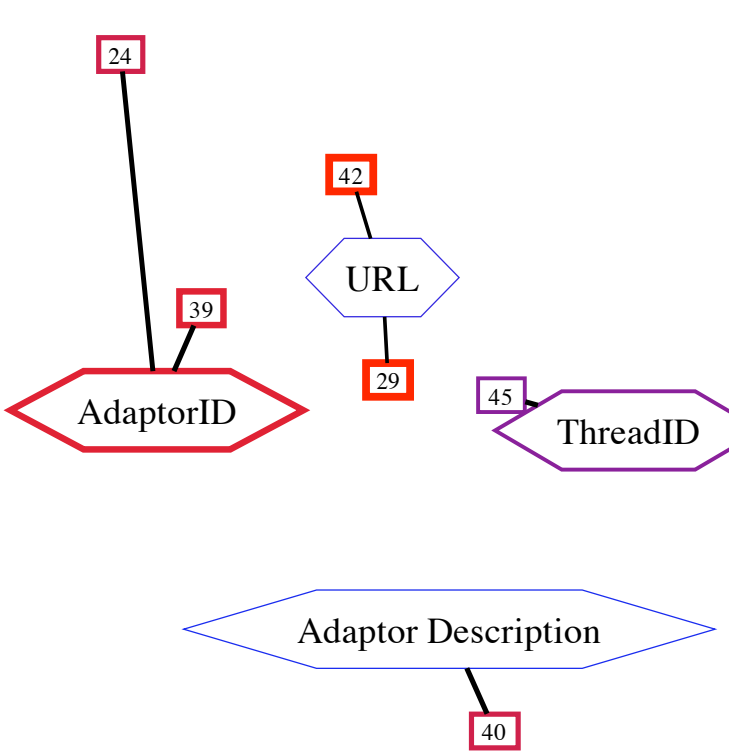
- Have a prototype that converts logs into .dot files for rendering with GraphViz
- Pluggable parsers
- Omit message strings; output alongside

# A real example, part 2

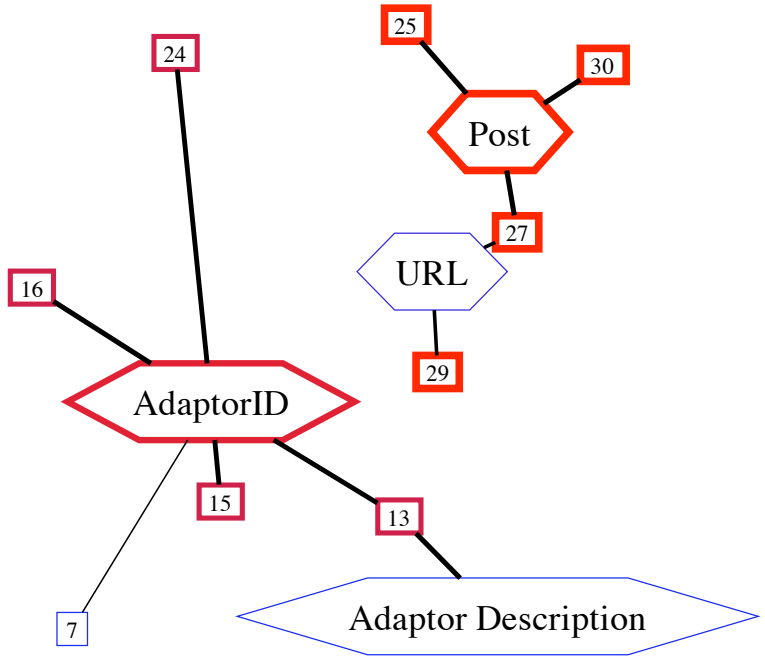



Hadoop datanode logs from Yahoo! M45 cluster

# Inconsistent identifiers



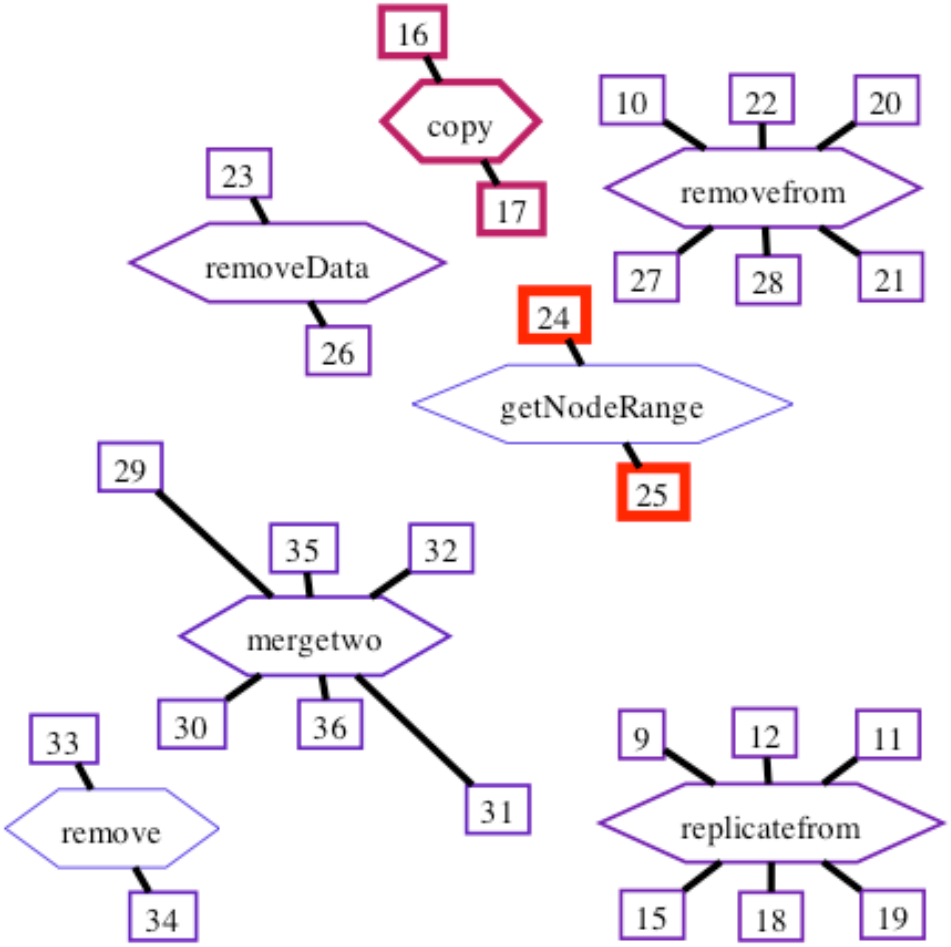

Old




New

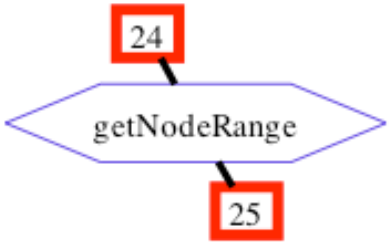
Logs from Chukwa, an open-source log collection system [Boulon 08, Rabkin 10]

# Ambiguous identifiers



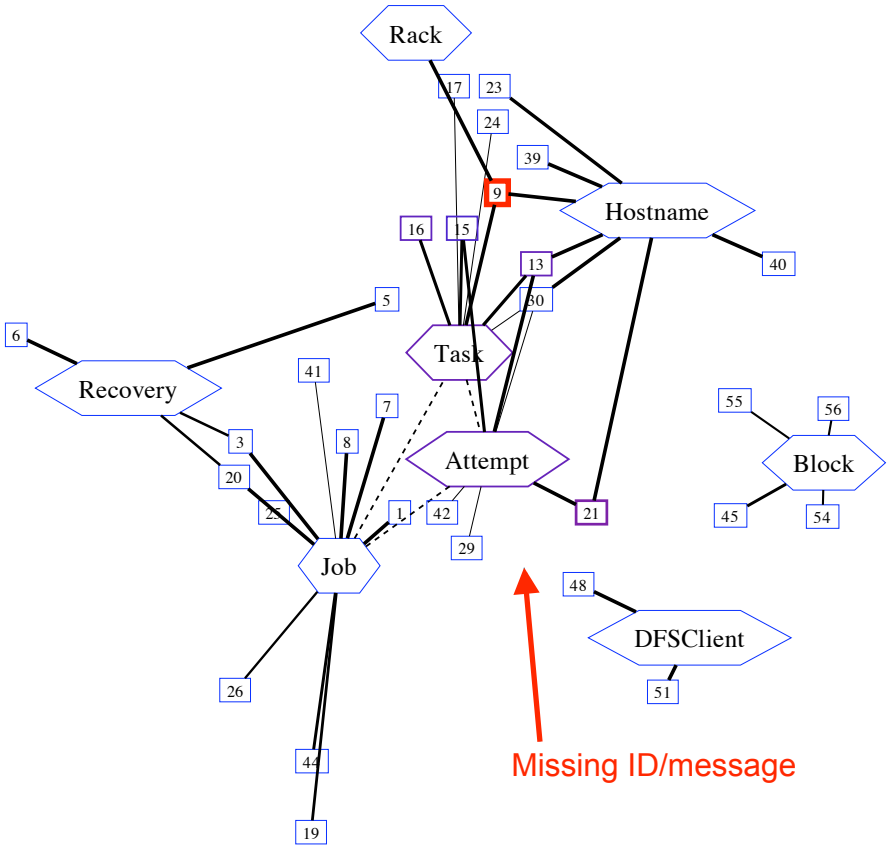

Logs from SCADS, an experimental system at Berkeley

# Ambiguous identifiers

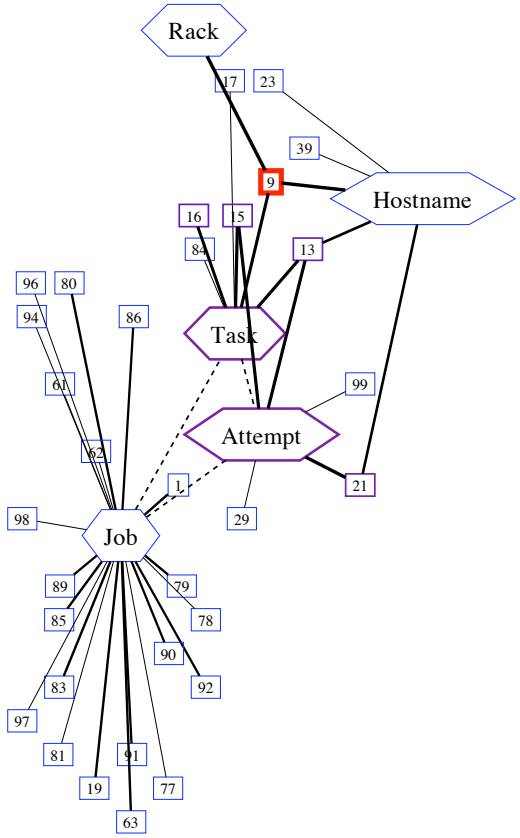



Logs from SCADS, an experimental system at Berkeley

# Comparing logs




15-node cluster at Berkeley




M45 cluster
(professional management)

Comparing Hadoop JobTracker logs


# Conclusions

- Aspects of log structure can be encoded in succinct diagrams.
- Our choice of representation captures:
  - missing identifiers, inconsistent identifiers, and ambiguous identifiers
  - How much detail about different topics
  - Ratio of routine vs peculiar messages + types
- Usable on real systems, even with limited understanding of system and logs
- No need for temporal information


# Questions?


# A note on parsing

- I used semi-hand-written parsers.
- Wrote rules to tag identifiers:
  - e.g., "job_..." is a job ID
- Tokenized lines, identified line by token sequence + constants
  - Special cases for numbers
- Explored using program analysis to extract messages
  - Came out ugly, but cleanable.
  - Need to fix names
  - Need to merge some categories


# Related work

- Xu 09
- State machines
- Entropy as metric?