# Fast Log Analysis by Automatically Parsing Heterogeneous Logs

Biplob Debnath and Will Dennis

NEC Laboratories America, Inc.

{biplob,wdennis}@nec-labs.com


June 25–29, 2018 | New York City, NY
www.qconnewyork.com #QConNYC

# Log Analysis

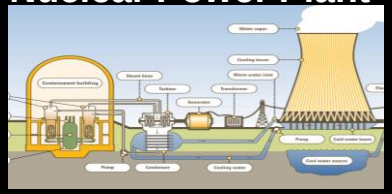
Internet of Things (IoT)

Software Systems

Computer Systems

Smart Cities

Nuclear Power Plant



Auto Production Line


- Logs ubiquitously exist in many complex systems
- Most of the logs include a massive amount of transaction or status data
- Tons of logs are generated, but difficult to investigate manually

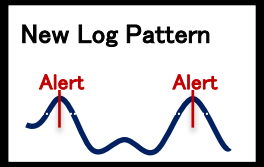
# Log Analysis: Example

**Raw Log Streams**

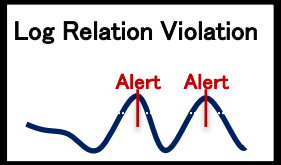
L1
L2
L3
L4
.
.
.
Ln

**Log Analyzer**

Analyzes logs using various features and reports alerts






**Sample Alert**


{"message":"\" uag412cjme107 \" , \" ZU4JNWG412PUC025 . 2013 / 09 / 24 15 : 00 : 17 \" , \" JNWSTART \" , \" - \" , \" 2013 / 09 / 25 00 : 00 : 17 . 000 ( 東京 ( 標準時 ) ) \" , \" - \" ","@version":"1","@timestamp":"2015-06-02T21:21:19.360Z","type":"No.08_Learning_file.log","host":"am14-06","seq":23}

{"message":"\" uag412cjme107 \" , \" ZU4JNWG412PUC025 . 2013 / 09 / 24 15 : 00 : 17 \" , \" JNWSTART \" , \" - \" , \" 2013 / 09 / 25 00 : 00 : 17 . 000 ( 東京 ( 標準時 ) ) \" , \" - \" ","@version":"1","@timestamp":"2015-06-02T21:21:19.364Z","type":"No.08_Learning_file.log","host":"am14-06","seq":24}

"anomaly_reason":"IP","anomaly_score":1.00,{"message":"\" IOROI - 1 \" , \" ZJd582YYx . 2015 / 03 / 31 15 : 01 : 00 \" , \" LOGIN ID = IOROI - 1 \" , \" 10 . 110 . 4 . 218 \" , \" 2015 / 04 / 01 00 : 01 : 00 . 000 ( 東京 ( 標準時 ) ) \" , \" - \" ","@version":"1","@timestamp":"2015-04-01T04:01:00.000Z","type":"No.11_Learning_file.log.Jianwu","host":"am14-06","P1F1":"1","P1NS1":"ZJd582YYx","ts1":"2015 / 03 / 31 15 : 01 : 00","P1F2":"1","IP":"10 . 110 . 4 . 218","mts":"2015 / 04 / 01 00 : 01 : 00","P1F3":"000","tags":["pattern","1"],"seq":1}

"anomaly_reason":"In between 2015 / 04 / 01 00 : 10 : 41 and 2015 / 04 / 01 00 : 20 : 41 pattern 15 exceeds previous maximum count of 1. Current count: 6",{"message":"\" NAGAMATSU - 2 \" , \" hkW351ciQ . 2015 / 03 / 31 15 : 13 : 17 \" , \" LOGIN ID = NAGAMATSU - 2 LOGIN FAILED \" , \" 10 . 110 . 4 . 107 \" , \" 2015 / 04 / 01 00 : 13 : 17 . 000 ( 東京 ( 標準時 ) ) \" , \" 2015 / 04 / 01 00 : 13 : 17 . 000 ( 東京 ( 標準時 ) ) \" ","@version":"1","@timestamp":"2015-04-01T04:13:17.000Z","type":"No.13_Learning_file.log.Jianwu","host":"am14-06","P15W1":"NAGAMATSU","P15F1":"2","P15NS1":"hkW351ciQ","ts1":"2015 / 03 / 31 15 : 13 : 17","P15W2":"NAGAMATSU","P15F2":"2","IP":"10 . 110 . 4 . 107","mts":"2015 / 04 / 01 00 : 13 : 17","P15F3":"000","ts3":"2015 / 04 / 01 00 : 13 : 17","P15F4":"000","tags":["pattern","15"],"seq":55}

"anomaly_reason":"Concurrency Violation for Pattern 1",{"message":"\" KITANO1 \" , \" cpA091acZ . 2015 / 03 / 31 15 : 01 : 42 \" , \" LOGIN ID = KITANO1 \" , \" 10 . 110 . 4 . 109 \" , \" 2015 / 04 / 01 00 : 00 : 48 . 000 ( 東京 ( 標準時 ) ) \" , \" \" ","@version":"1","@timestamp":"2015-04-01T04:00:48.000Z","type":"No.12_Learning_file.log.Jianwu","host":"am14-06","P1NS1":"cpA091acZ","ts1":"2015 / 03 / 31 15 : 01 : 42","IP":"10 . 110 . 4 . 109","mts":"2015 / 04 / 01 00 : 00 : 48","P1F1":"000","tags":["pattern","1"],"seq":2}

Log Parsing is the Core Step to any type of Log Analysis

# Log Parsing: Example

Mar 3 16:30:04 envctl APC_PDU_LEGAMPS: [INFO] PDU=pdu2z04-am-rack4f Leg=3 Amps=2.5

**Parsing Pattern ???**

%{DATETIME:timestamp} envctl APC_PDU_LEGAMPS: [INFO] PDU=%{NOTSPACE:PDU} Leg=%{NUMBER:Leg} Amps=%{NUMBER:Amps}

```
{
  “message”: “Mar 3 16:30:04 envctl APC_PDU_LEGAMPS: [INFO]  PDU=pdu-2z04-am-rack4f Leg=3 Amps=2.5”,
  "timestamp" : 2017-03-03T21:30:04.000Z,
   "PDU" : "pdu-2z04-am-rack4f",
   "Leg" : "3",
   "Amps" : "2.5“
}
```

# Heterogeneous Log Formats

Oct 23 13:53:39 am12-09 kernel: [448260.543317] sd 6:0:0:0: [sdc] tag#0 FAILED Result: hostbyte=DID_ERROR driverbyte=DRIVER_SENSE
Oct 23 13:53:39 am12-09 kernel: [448260.543324] sd 6:0:0:0: [sdc] tag#0 Sense Key : Hardware Error [current] [descriptor]
Oct 23 13:53:39 am12-09 kernel: [448260.543335] sd 6:0:0:0: [sdc] tag#0 Add. Sense: No additional sense information
Oct 23 13:54:09 am12-09 ntpd[1603]: Soliciting pool server 2001:67c:1560:8003::c8

**Syslog**

2017-10-04T12:39:44.269-0400 I NETWORK [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2017-10-04T12:39:44.269-0400 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/data/db/mongodb/diagnostic.data'
2017-10-04T12:39:44.270-0400 I NETWORK [initandlisten] waiting for connections on port 27017
2017-10-04T16:47:33.264-0400 I INDEX [conn6] build index on: testdb.testcol properties: { v: 1, key: { timestamp: -1 }, name: "timestamp_-1", ns: "testdb.testcol" }

**MongoDB**

2014-11-12 16:43:33.061 13625 INFO nova.virt.libvirt.driver [-] [instance: 9d9ffa30-b827-4330-8d8a-4bff5a782475] Instance destroyed successfully. 172.16.4.121 /var/log/nova/nova-compute.lo
2014-11-12 16:43:33.681 13625 INFO nova.compute.manager [-] [instance: 9d9ffa30-b827-4330-8d8a-4bff5a782475] During sync_power_state the instance has a pending task.
Skip. 172.16.4.121 /var/log/nova/nova-compute.log
2014-11-12 16:44:02.446 13625 AUDIT nova.compute.resource_tracker [-] Auditing locally available compute resources 172.16.4.121 /var/log/nova/nova-compute.log

**OpenStack**

(0):TimerRoutine():(w3wp.exe,0x1bd8,74):0[25 21:39:21] ---------- Process Director Timer Routine COMPLETE ---------- 457016, seconds=0.0781404
(0):GetUserBySID():2[25 21:39:21] SQL SELECT TABLE: tblUser WHERE: tblUser.guidSessionID=N'0b8b6ff1-5e87-48e4-9221-e5fe61a92cae'
(0):GetObject():2[25 21:39:21] SQL SELECT TABLE: tblContent WHERE: tblContent.oID='3b676138-0fe6-4dbd-a20b-49fe5b07725d'
(0):GetColumns():2[25 21:39:21] SQL SELECT TABLE: tblKViewColumn WHERE: tblKViewColumn.oKVID='3b676138-0fe6-4dbd-a20b-49fe5b07725d' ORDER BY tblKViewColumn.nColumnNum ASC

**Custom Application**

# Pattern Generation: Challenges

- Various log formats that can change over time
- Extremely huge amount of system logs
- Limited domain knowledge

Goal: Generate patterns with no (or minimal ) human involvement.

# Outline

- Pattern Generation Algorithm
- Demo Use-cases

# Outline

- Pattern Generation Algorithm
- Demo Use-cases

# Pattern Generation: Problem Statement

- Input
  - ❑A set of logs
- Output
  - ❑A set of GROK patterns to parse all logs

# Sample Input Logs

1. 2017/02/24 09:01:00 login 127.0.0.1 user=bear12
2. 2017/02/24 09:02:00 DB Connect 127.0.0.1 user=bear12
3. 2017/02/24 09:02:00 DB Disconnect 127.0.0.1 user=bear12
4. 2017/02/24 09:04:00 logout 127.0.0.1 user=bear12

5. 2017/02/24 09:05:00 login 127.0.0.1 user=bear34
6. 2017/02/24 09:06:00 DB Connect 127.0.0.1 user=bear34
7. 2017/02/24 09:07:00 DB Disconnect 127.0.0.1 user=bear34
8. 2017/02/24 09:08:00 logout 127.0.0.1 user=bear34

9. 2017/02/24 09:09:00 login 127.0.0.1 user=bear#1
10. 2017/02/24 09:10:00 DB Connect 127.0.0.1 user=bear#1
11. 2017/02/24 09:11:00 DB Disconnect 127.0.0.1 user=bear#1
12. 2017/02/24 09:12:00 logout 127.0.0.1 user=bear#1

## Pattern Count?

**Is it possible to give options to a user to cherry-pick his/her pattern-set of interest?**

# Pattern-Tree: Providing Cherry-Picking Options

2017/02/24 09:01:00 login 127.0.0.1 user=bear12
2017/02/24 09:02:00 DB Connect 127.0.0.1 user=bear12
2017/02/24 09:02:00 DB Disconnect 127.0.0.1 user=bear12
2017/02/24 09:04:00 logout 127.0.0.1 user=bear12

2017/02/24 09:05:00 login 127.0.0.1 user=bear34
2017/02/24 09:06:00 DB Connect 127.0.0.1 user=bear34
2017/02/24 09:07:00 DB Disconnect 127.0.0.1 user=bear34
2017/02/24 09:08:00 logout 127.0.0.1 user=bear34

2017/02/24 09:09:00 login 127.0.0.1 user=bear#1
2017/02/24 09:10:00 DB Connect 127.0.0.1 user=bear#1
2017/02/24 09:11:00 DB Disconnect 127.0.0.1 user=bear#1
2017/02/24 09:12:00 logout 127.0.0.1 user=bear#1

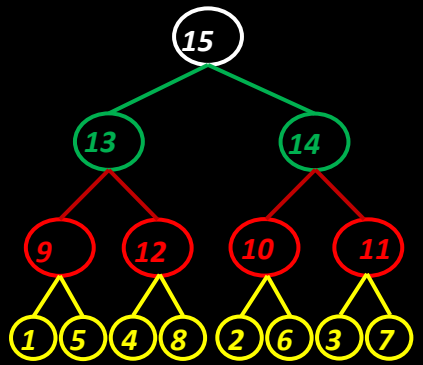
**Input Logs**

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = NOTSPACE
6. DATETIME DB Connect user = NOTSPACE
7. DATETIME DB Disconnect IPV4 user = NOTSPACE
8. DATETIME logout IPV4 user = NOTSPACE

9. DATETIME login IPV4 user = NOTSPACE
10. DATETIME DB Connect IPV4 user = NOTSPACE
11. DATETIME DB Disconnect IPV4 user = NOTSPACE
12. DATETIME logout IPV4 user = NOTSPACE

13. DATETIME WORD IPV4 user = NOTSPACE
14. DATETIME DB WORD IPV4 user = NOTSPACE

15. DATETIME * WORD IPV4 user = NOTSPACE




Pattern Tree

# Solution Sketch

- Step 1: Group similar logs into clusters
  - How to find clusters?
    - k-means: deciding k needs effort
    - OPTICS: deciding MinPts and Epsilon is problematic and space complexity $O(n^2)$
- Step 2: Generate one pattern for each cluster
  - Merge logs together
  - Challenge
    - How to decide merging order?
    - Time complexity to decide the right ordering takes $O(m^2)$, where m is the average cluster size

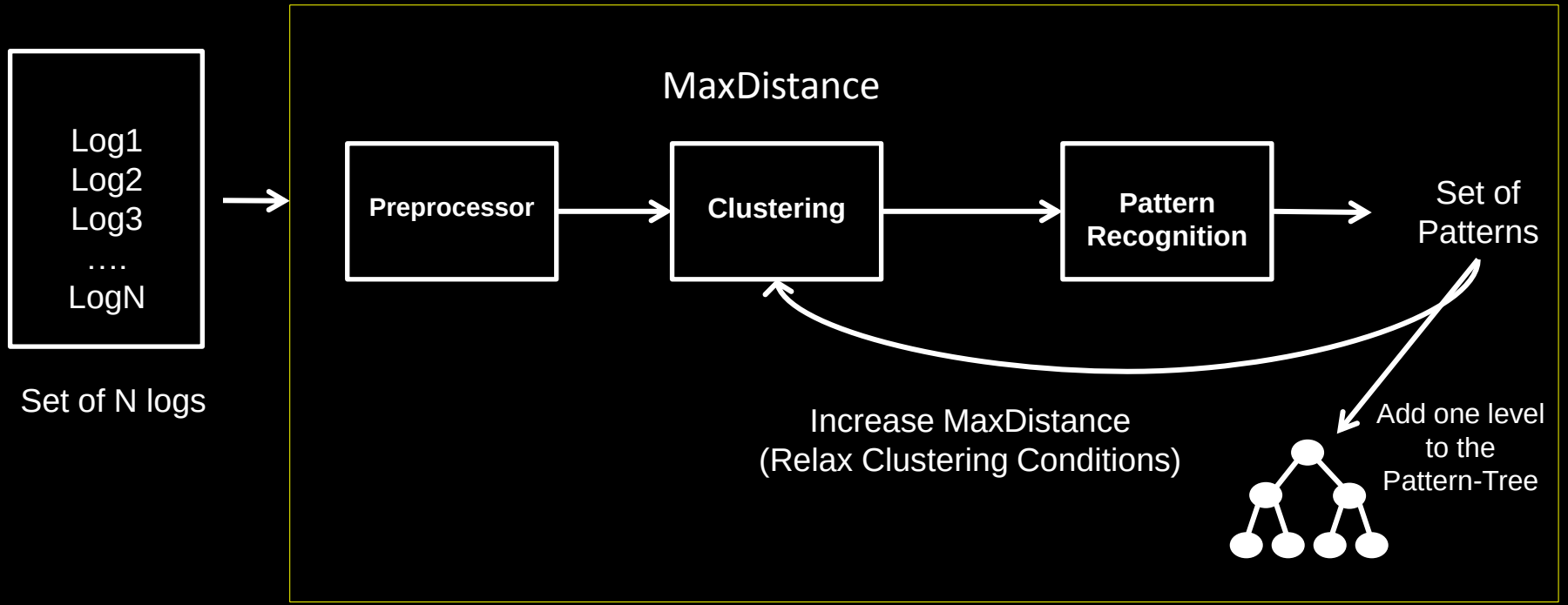
# LogMine: An Automatic Pattern Generator

- LogMine: Fast Pattern Recognition for Log Analytics [CIKM 16]

- LogMine generates a pattern-tree from a set of input log
  - Can work without any human involvement
  - Scalable
    - Scans logs only once
    - Time complexity O(# of logs)
    - Memory complexity O(# of clusters)
    - Merging time complexity is O(# logs in a cluster )

- LogMine leverages the following fact
  - Logs are not randomly generated, rather generated by computing devices
    - Thus, logs having same formats are very similar

# LogMine: Workflow




Note: MaxDistance is an internal parameter, and its initial value is set to 0.0000001

# LogMine: 3-Step Process

1. Preprocessing
   a) Tokenization
   b) Timestamp unification
   c) Datatype identification

2. Generating Pattern-Tree (Iterative Process)
   a) Starts with a very small (e.g., 0.0000001) similarity distance threshold
   b) Form clusters based similarity among logs
   c) If any cluster has multiple logs, merge them together to form one log
   d) Add clusters in the pattern-tree
   e) If more than one clusters formed
      - Increase similarity distance threshold
      - Repeat Step a, b, c, d, e

3. Selecting Final Patterns from the Pattern-Tree
   a) If user inputs a max pattern limit, then outputs the first level closer to Level-0 satisfying user's limit
   b) Otherwise, outputs the tree level having minimum patterns with no wildcards (i.e., cost = 0)

# Preprocessing: Tokenization

1. 2017/02/24 09:01:00 login 127.0.0.1 user=bear12
2. 2017/02/24 09:02:00 DB Connect 127.0.0.1 user=bear12
3. 2017/02/24 09:02:00 DB Disconnect 127.0.0.1 user=bear12
4. 2017/02/24 09:04:00 logout 127.0.0.1 user=bear12
5. 2017/02/24 09:05:00 login 127.0.0.1 user=bear34
6. 2017/02/24 09:06:00 DB Connect 127.0.0.1 user=bear34
7. 2017/02/24 09:07:00 DB Disconnect 127.0.0.1 user=bear34
8. 2017/02/24 09:08:00 logout 127.0.0.1 user=bear34
9. 2017/02/24 09:09:00 login 127.0.0.1 user=bear#1
10. 2017/02/24 09:10:00 DB Connect 127.0.0.1 user=bear#1
11. 2017/02/24 09:11:00 DB Disconnect 127.0.0.1 user=bear#1
12. 2017/02/24 09:12:00 logout 127.0.0.1 user=bear#1

1. 2017/02/24 09:01:00 login 127.0.0.1 user = bear12
2. 2017/02/24 09:02:00 DB Connect 127.0.0.1 user = bear12
3. 2017/02/24 09:02:00 DB Disconnect 127.0.0.1 user = bear12
4. 2017/02/24 09:04:00 logout 127.0.0.1 user = bear12
5. 2017/02/24 09:05:00 login 127.0.0.1 user = bear34
6. 2017/02/24 09:06:00 DB Connect 127.0.0.1 user = bear34
7. 2017/02/24 09:07:00 DB Disconnect 127.0.0.1 user = bear34
8. 2017/02/24 09:08:00 logout 127.0.0.1 user = bear34
9. 2017/02/24 09:09:00 login 127.0.0.1 user = bear#1
10. 2017/02/24 09:10:00 DB Connect 127.0.0.1 user = bear#1
11. 2017/02/24 09:11:00 DB Disconnect 127.0.0.1 user = bear#1
12. 2017/02/24 09:12:00 logout 127.0.0.1 user = bear#1

- Split a log into tokens by tokenization delimiter
  - default value: white-space characters (i.e., tab, space, etc.), comma, and "= "
- Consecutive spaces are consolidated

# Preprocessing: Timestamp Unification

1. 2017/02/24 09:01:00 login 127.0.0.1 user = bear12
2. 2017/02/24 09:02:00 DB Connect 127.0.0.1 user = bear12
3. 2017/02/24 09:02:00 DB Disconnect 127.0.0.1 user = bear12
4. 2017/02/24 09:04:00 logout 127.0.0.1 user = bear12
5. 2017/02/24 09:05:00 login 127.0.0.1 user = bear34
6. 2017/02/24 09:06:00 DB Connect 127.0.0.1 user = bear34
7. 2017/02/24 09:07:00 DB Disconnect 127.0.0.1 user = bear34
8. 2017/02/24 09:08:00 logout 127.0.0.1 user = bear34
9. 2017/02/24 09:09:00 login 127.0.0.1 user = bear#1
10. 2017/02/24 09:10:00 DB Connect 127.0.0.1 user = bear#1
11. 2017/02/24 09:11:00 DB Disconnect 127.0.0.1 user = bear#1
12. 2017/02/24 09:12:00 logout 127.0.0.1 user = bear#1

1. 2017/02/24 09:01:00.000 login 127.0.0.1 user = bear12
2. 2017/02/24 09:02:00.000 DB Connect 127.0.0.1 user = bear12
3. 2017/02/24 09:02:00.000 DB Disconnect 127.0.0.1 user = bear12
4. 2017/02/24 09:04:00.000 logout 127.0.0.1 user = bear12
5. 2017/02/24 09:05:00.000 login 127.0.0.1 user = bear34
6. 2017/02/24 09:06:00.000 DB Connect 127.0.0.1 user = bear34
7. 2017/02/24 09:07:00.000 DB Disconnect 127.0.0.1 user = bear34
8. 2017/02/24 09:08:00.000 logout 127.0.0.1 user = bear34
9. 2017/02/24 09:09:00.000 login 127.0.0.1 user = bear#1
10. 2017/02/24 09:10:00.000 DB Connect 127.0.0.1 user = bear#1
11. 2017/02/24 09:11:00.000 DB Disconnect 127.0.0.1 user = bear#1
12. 2017/02/24 09:12:00.000 logout 127.0.0.1 user = bear#1

- Identifies timestamps and unifies them into a fixed format: yyyy/MM/dd hh:mm:ss.SSS
- Very challenging problem due to the heterogeneous timestamp formats
  - For example, “2016/02/23 09:00:31” can be logged as “02-23-2016 09:00:31” or “02/23/2016 09:00:31” or “Feb 23 2016 09:00:31” or “02/23/2016 09:00:31.000”, and so on

# Preprocessing: Datatype Identification

| Datatype | Regular Expression (RegEx) Syntax |
|---|---|
| WORD | [a-zA-Z_0-9]+ |
| NUMBER | (-[0-9]*.)?[0-9]+) \| (0x[0-9A-F]+) |
| IPV4 | [0-9]{1-3}.[0-9]{0-9}.[\d]{0-9}.[0-9]{1.3} |
| NOTSPACE | ^[ \t\n\x0B\f\r]+ |
| DATETIME | [0-9]{4}.[0-9]{2}.[0-9]{2} [0-9]{2}:[0-9]{2}.[0-9]{2}.[0-9]{3} |
| ANYDATA | .* |

Table: Sample Datatype RegEx syntax

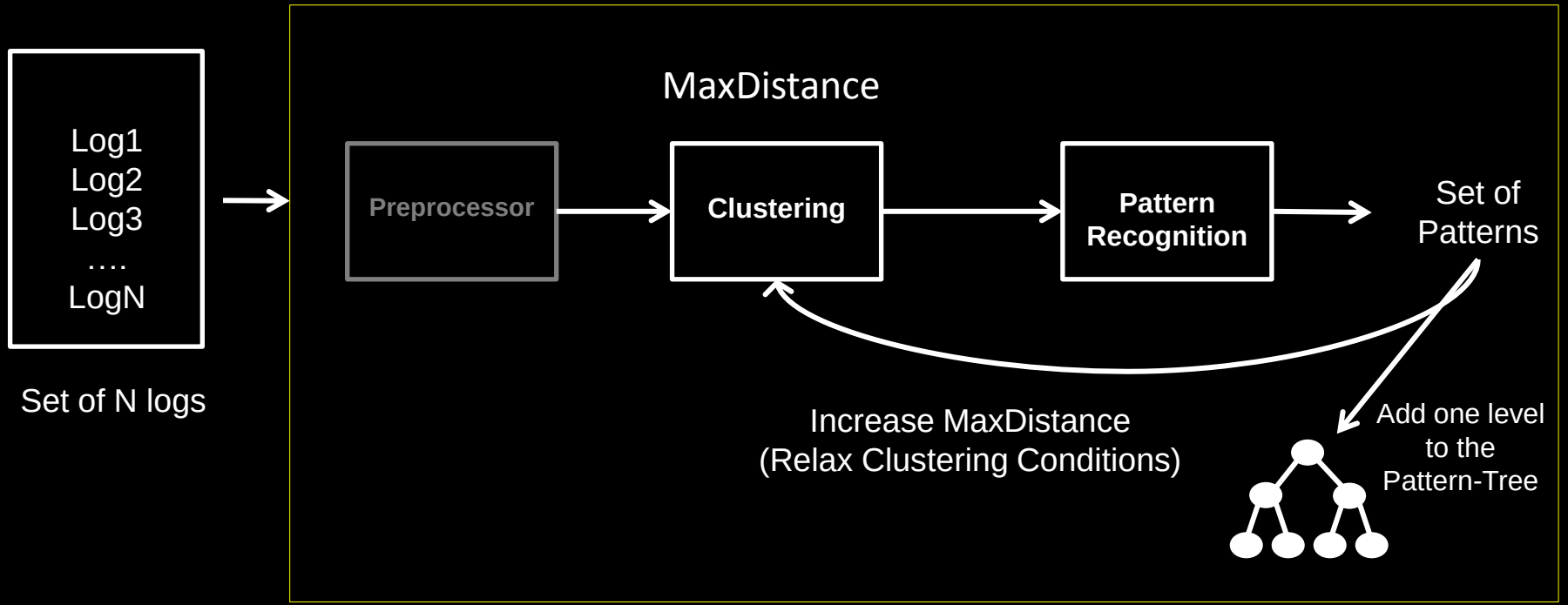
# Preprocessing: Datatype Identification

1. 2017/02/24 09:01:00.000 login 127.0.0.1 user = bear12
2. 2017/02/24 09:02:00.000 DB Connect 127.0.0.1 user = bear12
3. 2017/02/24 09:02:00.000 DB Disconnect 127.0.0.1 user = bear12
4. 2017/02/24 09:04:00.000 logout 127.0.0.1 user = bear12
5. 2017/02/24 09:05:00.000 login 127.0.0.1 user = bear34
6. 2017/02/24 09:06:00.000 DB Connect 127.0.0.1 user = bear34
7. 2017/02/24 09:07:00.000 DB Disconnect 127.0.0.1 user = bear34
8. 2017/02/24 09:08:00.000 logout 127.0.0.1 user = bear34
9. 2017/02/24 09:09:00.000 login 127.0.0.1 user = bear#1
10. 2017/02/24 09:10:00.000 DB Connect 127.0.0.1 user = bear#1
11. 2017/02/24 09:11:00.000 DB Disconnect 127.0.0.1 user = bear#1
12. 2017/02/24 09:12:00 .000logout 127.0.0.1 user = bear#1

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = WORD
6. DATETIME DB Connect IPV4 user = WORD
7. DATETIME DB Disconnect IPV4 user = WORD
8. DATETIME logout IPV4 user = WORD
9. DATETIME login IPV4 user = NOTSPACE
10. DATETIME DB Connect IPV4 user = NOTSPACE
11. DATETIME DB Disconnect IPV4 user = NOTSPACE
12. DATETIME logout IPV4 user = NOTSPACE

- Identifies a token's datatype in this order: DATETIME, IPV4, NUMBER, WORD, NOTSPACE
  - Exception: tokens having only either alphabets or single characters
    - Intuition: developers uses meaningful words to interpret log messages ☺
  - Note: this step is not needed during testing phase

# LogMine: Workflow (Recap)




Note: MaxDistance is an internal parameter, and its initial value is set to very low (e.g., 0.0000001)

# Example: Distance Calculation

Log1: abc cde efg xyz qcon

Log2: abc cde efg xyz

Similarity (Log1, Log2) = 4/5 = 0.80

Distance (Log1, Log2) = 1 – Similarity = 1 – 0.80 = 0.20

$$Dist(P,Q) = 1 - \sum_{i=1}^{Min(|P|,|Q|)} \frac{Score(Pi,Qi)}{Max(|P|,|Q|)}$$

$$Score(X,Y) = \begin{cases} 1 & If\ X = Y \\ 0 & Otherwise \end{cases}$$

# Clustering: Example

**Sample logs**

| |
|---|
| LOG1 |
| LOG2 |
| LOG3 |
| LOG4 |
| LOG5 |
| LOG6 |
| LOG7 |
| LOG8 |
| LOG9 |

MaxDistance = 0.01

# Clustering: Example

**Sample logs**

| |
|---|
| LOG1 |
| LOG2 |
| LOG3 |
| LOG4 |
| LOG5 |
| LOG6 |
| LOG7 |
| LOG8 |
| LOG9 |

MaxDistance = 0.01

LOG1

# Clustering: Example

**Sample logs**

| |
|---|
| LOG1 |
| LOG2 |
| LOG3 |
| LOG4 |
| LOG5 |
| LOG6 |
| LOG7 |
| LOG8 |
| LOG9 |

MaxDistance = 0.01

Dist (LOG1 , LOG2) = 0.2

LOG1

LOG2

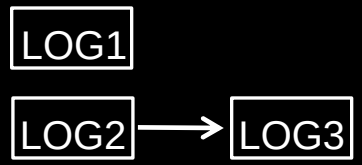
# Clustering: Example

**Sample logs**

| |
|---|
| LOG1 |
| LOG2 |
| LOG3 |
| LOG4 |
| LOG5 |
| LOG6 |
| LOG7 |
| LOG8 |
| LOG9 |

MaxDistance = 0.01

Dist (LOG3 , LOG1) = 0.8
Dist (LOG3 , LOG2) = 0.001



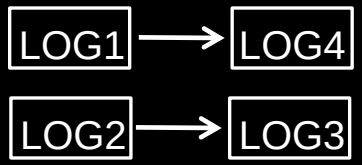
# Clustering: Example

**Sample logs**

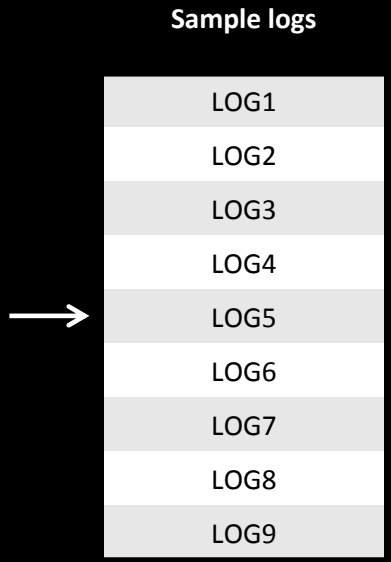
| |
|---|
| LOG1 |
| LOG2 |
| LOG3 |
| LOG4 |
| LOG5 |
| LOG6 |
| LOG7 |
| LOG8 |
| LOG9 |

MaxDistance = 0.01

Dist (LOG4 , LOG1) = 0

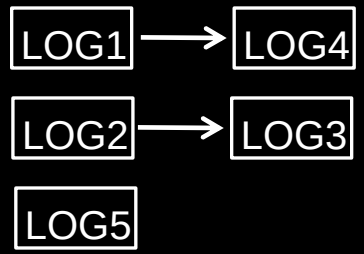


# Clustering: Example




MaxDistance = 0.01

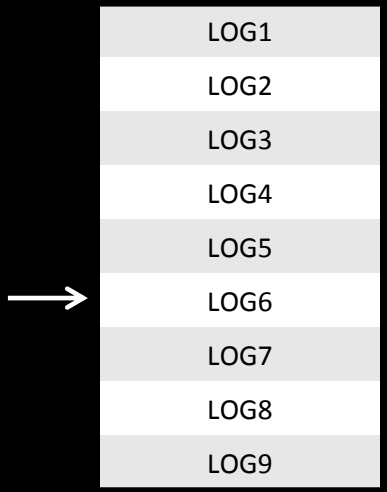
Dist (LOG5 , LOG1) = 0.2
Dist (LOG5 , LOG2) = 0.5




Optimization: Compare only with the first log in a cluster

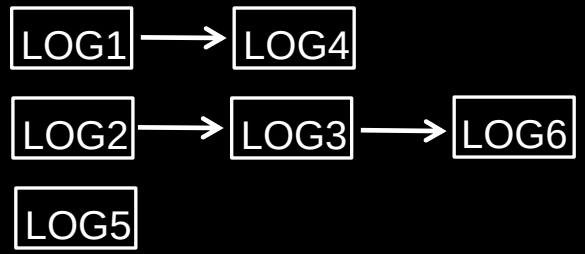
# Clustering: Example

**Sample logs**




MaxDistance = 0.01

Dist (LOG6 , LOG1) = 0.3
Dist (LOG6 , LOG2) = 0.001



# Clustering: Example

**Sample logs**

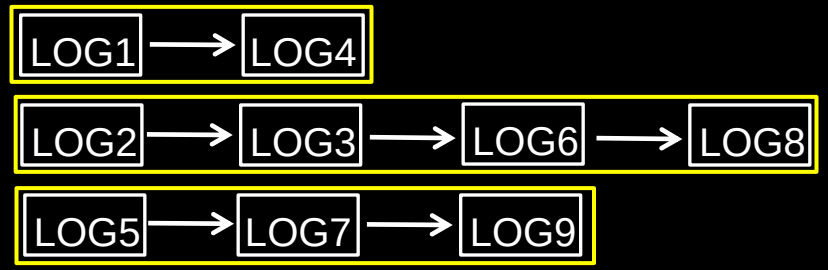
| |
|---|
| LOG1 |
| LOG2 |
| LOG3 |
| LOG4 |
| LOG5 |
| LOG6 |
| LOG7 |
| LOG8 |
| LOG9 |

Max Distance = 0.01

Cluster1

Cluster2

Cluster3



# Merging Logs: Smith–Waterman Algorithm

- Logs in a cluster are merged together to produce one final pattern per cluster.
- Optimization: No need to follow any merging order.

LOG1: DATETIME WORD IPV4 user = NOTSPACE

LOG2: DATETIME DB WORD IPV4 user = NOTSPACE

↓ **Alignment**

LOG1: DATETIME **---** WORD IPV4 user = NOTSPACE

LOG2: DATETIME DB WORD IPV4 user = NOTSPACE

↓ **Merge the Alignments**

Result: DATETIME ***** WORD IPV4 user = NOTSPACE

Reference: https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm

# Sample Merging Rules

- If token is not a datatype,  first identify its datatype

- Next, apply following rules:
  - DATETME + NOTSPACE = *
  - WORD + NOTSPACE =  NOTSPACE
  - WORD + NUMBER  =  NOTSPACE
  - IPV4 + WORD = NOTSPACE
  - IPV4 + NUMBER = NOTSPACE
  - IPV4 + DATETIME = *
  - --- + WORD = *
  - --- + NUMBER = *
  - --- + NOTSPACE = *
  - --- + DATETIME = *

# Recap: Preprocessing

1. 2017/02/24 09:01:00 login 127.0.0.1 user=bear12
2. 2017/02/24 09:02:00 DB Connect 127.0.0.1 user=bear12
3. 2017/02/24 09:02:00 DB Disconnect 127.0.0.1 user=bear12
4. 2017/02/24 09:04:00 logout 127.0.0.1 user=bear12
5. 2017/02/24 09:05:00 login 127.0.0.1 user=bear34
6. 2017/02/24 09:06:00 DB Connect 127.0.0.1 user=bear34
7. 2017/02/24 09:07:00 DB Disconnect 127.0.0.1 user=bear34
8. 2017/02/24 09:08:00 logout 127.0.0.1 user=bear34
9. 2017/02/24 09:09:00 login 127.0.0.1 user=bear#1
10. 2017/02/24 09:10:00 DB Connect 127.0.0.1 user=bear#1
11. 2017/02/24 09:11:00 DB Disconnect 127.0.0.1 user=bear#1
12. 2017/02/24 09:12:00 logout 127.0.0.1 user=bear#1

Sample logs

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = WORD
6. DATETIME DB Connect IPV4 user = WORD
7. DATETIME DB Disconnect IPV4 user = WORD
8. DATETIME logout IPV4 user = WORD
9. DATETIME login IPV4 user = NOTSPACE
10. DATETIME DB Connect IPV4 user = NOTSPACE
11. DATETIME DB Disconnect IPV4 user = NOTSPACE
12. DATETIME logout IPV4 user = NOTSPACE

Preprocessed logs

# LogMine: Pattern-Tree Formation

Level 1

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = NOTSPACE
6. DATETIME DB Connect user = NOTSPACE
7. DATETIME DB Disconnect IPV4 user = NOTSPACE
8. DATETIME logout IPV4 user = NOTSPACE

Level 2

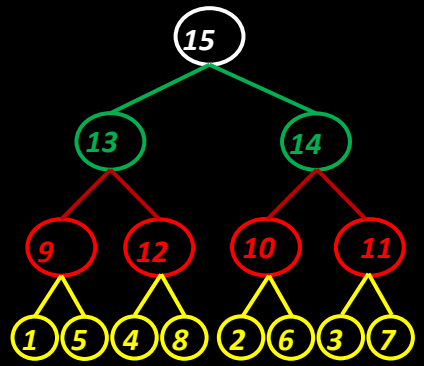
9. DATETIME login IPV4 user = NOTSPACE
10. DATETIME DB Connect IPV4 user = NOTSPACE
11. DATETIME DB Disconnect IPV4 user = NOTSPACE
12. DATETIME logout IPV4 user = NOTSPACE

Level 3

13. DATETIME WORD IPV4 user = NOTSPACE
14. DATETIME DB WORD IPV4 user = NOTSPACE

Level 4

15. DATETIME * WORD IPV4 user = NOTSPACE




Pattern Tree

Memory Usage depends on Level 1 Size

# Optimization: Fast Level 1 Formation

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = WORD
6. DATETIME DB Connect IPV4 user = WORD
7. DATETIME DB Disconnect IPV4 user = WORD
8. DATETIME logout IPV4 user = WORD
9. DATETIME login IPV4 user = NOTSPACE
10. DATETIME DB Connect IPV4 user = NOTSPACE
11. DATETIME DB Disconnect IPV4 user = NOTSPACE
12. DATETIME logout IPV4 user = NOTSPACE

Preprocessed logs

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = WORD
6. DATETIME DB Connect IPV4 user = WORD
7. DATETIME DB Disconnect IPV4 user = WORD
8. DATETIME logout IPV4 user = WORD
9. DATETIME login IPV4 user = NOTSPACE
10. DATETIME DB Connect IPV4 user = NOTSPACE
11. DATETIME DB Disconnect IPV4 user = NOTSPACE
12. DATETIME logout IPV4 user = NOTSPACE

Preprocessed logs

**Level 1 could be formed without clustering as initial MaxDistance is set to a very low value (i.e., 0.0000001)**

**Hint: Identify unique log lines by using log string as the hash-key ☺**

# Optimization: Fast Level 1 Formation

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = WORD
6. DATETIME DB Connect IPV4 user = WORD
7. DATETIME DB Disconnect IPV4 user = WORD
8. DATETIME logout IPV4 user = WORD
9. DATETIME login IPV4 user = NOTSPACE
10. DATETIME DB Connect IPV4 user = NOTSPACE
11. DATETIME DB Disconnect IPV4 user = NOTSPACE
12. DATETIME logout IPV4 user = NOTSPACE

Preprocessed logs

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = NOTSPACE
6. DATETIME DB Connect IPV4 user = NOTSPACE
7. DATETIME DB Disconnect IPV4 user = NOTSPACE
8. DATETIME logout IPV4 user = NOTSPACE

Level 1 (Unique logs)

**Note: By intelligently forming Level 1, clustering procedure can be fully avoided ☺**

# LogMine: Selecting Final Pattern-Set

Level 1

1. DATETIME login IPV4 user = WORD
2. DATETIME DB Connect IPV4 user = WORD
3. DATETIME DB Disconnect IPV4 user = WORD
4. DATETIME logout IPV4 user = WORD
5. DATETIME login IPV4 user = NOTSPACE
6. DATETIME DB Connect user = NOTSPACE
7. DATETIME DB Disconnect IPV4 user = NOTSPACE
8. DATETIME logout IPV4 user = NOTSPACE

Level 2

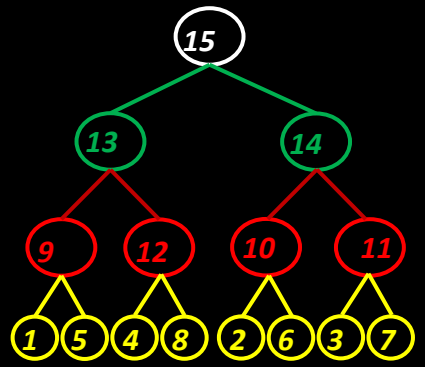
9. DATETIME login IPV4 user = NOTSPACE
10. DATETIME DB Connect IPV4 user = NOTSPACE
11. DATETIME DB Disconnect IPV4 user = NOTSPACE
12. DATETIME logout IPV4 user = NOTSPACE

Level 3

13. DATETIME WORD IPV4 user = NOTSPACE
14. DATETIME DB WORD IPV4 user = NOTSPACE

Level 4

15. DATETIME * WORD IPV4 user = NOTSPACE




Pattern Tree

Cost: 12

Cost: 0

Cost: 0

Cost: 0

Max Pattern Limit = 5, then Level 2 will be the output

Cost is Calculated based on WildCards and covered Logs Count. For example, pattern 15 contains one wildcard and it covers 12 logs – cost is estimated as 1*12= 12.

If user gives no Max Pattern limit on pattern count, then Level 3 will be the final output as it has the minimum cost with minimum number of patterns.

# Final GROK Patterns

- Selected Patterns
  - **DATETIME WORD IPV4 user = NOTSPACE**
  - **DATETIME DB WORD IPV4 user = NOTSPACE**

- **Patterns in GROK formats**
  - **%{DATETIME:P1DT1} %{WORD:P1W1} %{IPV4:P1IP1} user = %{NOTSPACE:P1NS1}**
  - **%{DATETIME:P2DT1} DB %{WORD:P2W1} %{IPV4:P2IP1} user = %{NOTSPACE:P2NS1}**

- **Renaming fields based on heuristics**
  - **%{DATETIME:timestamp} %{WORD:P1W1} %{IPV4:P1IP1} user = %{NOTSPACE:user}**
  - **%{DATETIME:timestamp} DB %{WORD:P2W1} %{IPV4:P2IP1} user = %{NOTSPACE:user}**

# Background: GROK Basics

- GROK combines RegEx into something that matches tokens
- The syntax for a GROK pattern is %{SYNTAX:SEMANTIC}
- The SYNTAX is how matching (i.e., RegEx pattern to use) is found
  - For example,  127.0.0.1 will be matched by the IPV4 pattern
- The SEMANTIC is the identifier given to the piece of text being matched
  - For example, a string 127.0.0.1 might identify the client making a request
- For parsing, “127.0.0.1” the GROK would be %{IPV4:client}

Reference: https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html

# Preprocessing Issue # 1: Timestamp Unification

1. 2017/02/24 09:01:00 login 127.0.0.1 user=bear12
2. 2017/02/24 09:02:00 DB Connect 127.0.0.1 user=bear12
3. 2017/02/24 09:02:00 DB Disconnect 127.0.0.1 user=bear12
4. 2017/02/24 09:04:00 logout 127.0.0.1 user=bear12
5. 2017/02/24 09:05:00 login 127.0.0.1 user=bear34
6. 2017/02/24 09:06:00 DB Connect 127.0.0.1 user=bear34
7. 2017/02/24 09:07:00 DB Disconnect 127.0.0.1 user=bear34
8. 2017/02/24 09:08:00 logout 127.0.0.1 user=bear34
9. 2017/02/24 09:09:00 login 127.0.0.1 user=bear#1
10. 2017/02/24 09:10:00 DB Connect 127.0.0.1 user=bear#1
11. 2017/02/24 09:11:00 DB Disconnect 127.0.0.1 user=bear#1
12. 2017/02/24 09:12:00 logout 127.0.0.1 user=bear#1

1. 2017/02/24 09:01:00.000 login 127.0.0.1 user = bear12
2. 2017/02/24 09:02:00.000 DB Connect 127.0.0.1 user = bear12
3. 2017/02/24 09:02:00.000 DB Disconnect 127.0.0.1 user = bear12
4. 2017/02/24 09:04:00.000 logout 127.0.0.1 user = bear12
5. 2017/02/24 09:05:00.000 login 127.0.0.1 user = bear34
6. 2017/02/24 09:06:00.000 DB Connect 127.0.0.1 user = bear34
7. 2017/02/24 09:07:00.000 DB Disconnect 127.0.0.1 user = bear34
8. 2017/02/24 09:08:00.000 logout 127.0.0.1 user = bear34
9. 2017/02/24 09:09:00.000 login 127.0.0.1 user = bear#1
10. 2017/02/24 09:10:00.000 DB Connect 127.0.0.1 user = bear#1
11. 2017/02/24 09:11:00.000 DB Disconnect 127.0.0.1 user = bear#1
12. 2017/02/24 09:12:00.000 logout 127.0.0.1 user = bear#1

- Identifies timestamps and unifies them into a fixed format: yyyy/MM/dd hh:mm:ss.SSS
- Very challenging problem due to the heterogeneous timestamp formats
  - For example, “2016/02/23 09:00:31” can be logged as “02-23-2016 09:00:31” or “02/23/2016 09:00:31” or “Feb 23 2016 09:00:31” or “02/23/2016 09:00:31.000”, and so on

# Preprocessing Issue # 1: Solution

- Optionally, users can specify timestamp format as input

- If no format specified, maintain timestamp format knowledgebase
  - Time Complexity: O(k) for a token, where k is the knowledgebase size
  - Overall complexity: O(k*t*n), assuming n logs and on average each has t tokens
  - Solution:
    - Reducing k: Caching the matching formats
    - Reducing t: Pruning tokens based on the commonly uses syntax to specify months (i.e., Jan,…,Dec, January,…,December, etc.), days (01, 02, …, 31, Monday, …., Sunday, etc.), years (2000, …., 2100, etc.) , hours (00,…,23, etc.) and minute (00,..59), and so on.

# Preprocessing Issue # 2: Splitting Delimiter is not Enough

- 2017/02/24 09:01:00 read file1 size 123KB in 100seconds
  - Delimiters cannot split yellow colored tokens ☹
  - Needs human engagement ☹
- Solution #1
  - User can preprocess logs before running the pattern generation algorithm
    - 2017/02/24 09:01:00 read file1 size 123 KB in 100 seconds
- Solution #2
  - User can provide RegEx rules to split tokens
    - [0-9]+KB → [0-9]+ KB
    - [0-9]+ seconds → [0-9]+ seconds
  - Increases tokenization complexity to O(k*t*n*r), where r is the number of split rules
    - It can be reduced to O(n*t). Hint: Leverage similar tricks used to optimize Level 1 (see slide # 42-43)

# GROK Pattern Issue: Field Semantics

- Selected Patterns
  - **DATETIME WORD IPV4 user = NOTSPACE**
  - **DATETIME DB WORD IPV4 user = NOTSPACE**

- **Patterns in GROK formats**
  - **%{DATETIME:P1DT1} %{WORD:P1W1} %{IPV4:P1IP1} user = %{NOTSPACE:P1NS1}**
  - **%{DATETIME:P2DT2} DB %{WORD:P2W1} %{IPV4:P2IP1} user = %{NOTSPACE:P2NS1}**

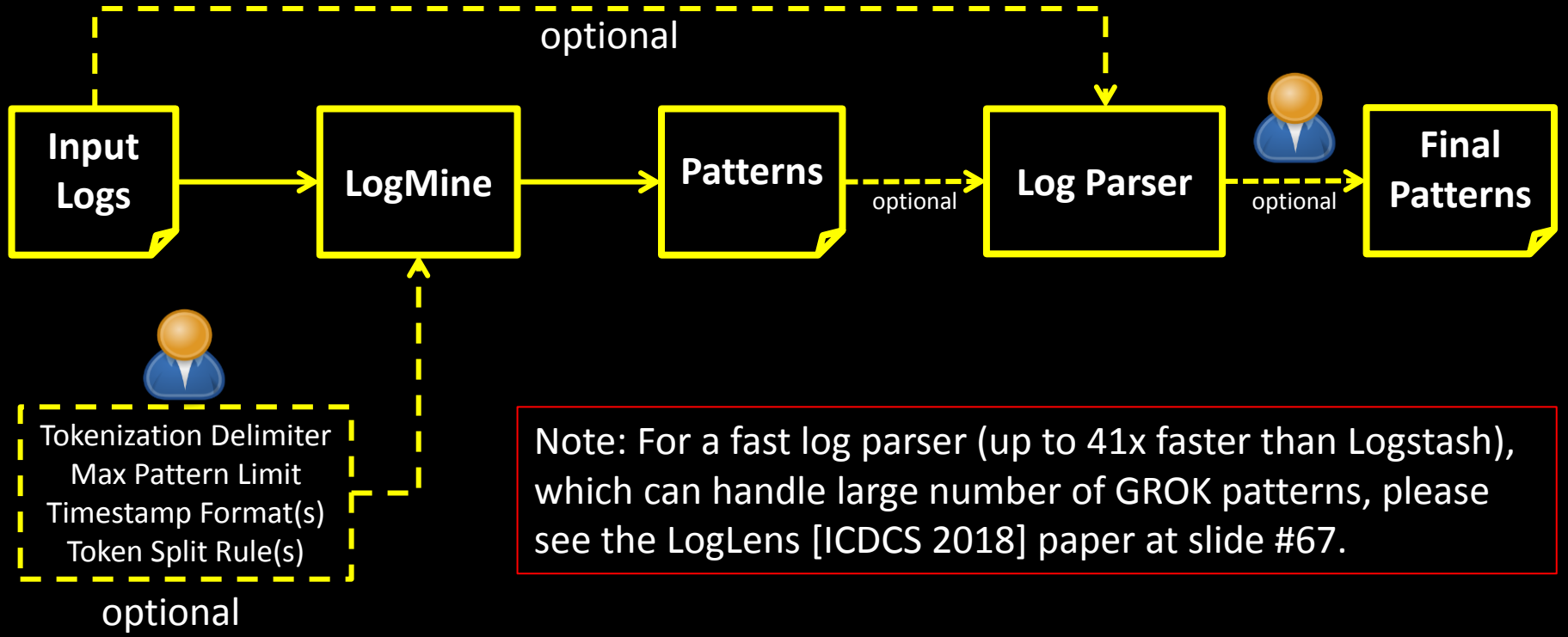
- **Renaming fields based on heuristics**
  - **%{DATETIME:timestamp} %{WORD:P1W1} %{IPV4:P1IP1} user = %{NOTSPACE:user}**
  - **%{DATETIME:timestamp} DB %{WORD:P2W1} %{IPV4:P2IP1} user = %{NOTSPACE:user}**

- **Solution: engage users to edit patterns**
  - **%{DATETIME:timestamp} %{WORD:user_action} %{IPV4:main_server} user = %{NOTSPACE:user}**
  - **%{DATETIME:timestamp} DB %{WORD:db_action} %{IPV4:db_server} user = %{NOTSPACE:user}**

# Lesson: Do not keep humans out of the loop

- Provide options to incorporate users' domain knowledge
- Instead of no human involvement, minimize human effort
  - If an algorithm has any parameters, set the default values to cover common usecases
  - Do not expose a parameter, if it can resolved internally
- Provide options to review and edit generated patterns
  - After pattern generation is done, parse input logs to generate statistics
  - For each pattern, show its coverage, sample logs and parsed outputs
  - By reviewing stats and samples, users can easily edit patterns if needed

# Workflow: Human in the Loop

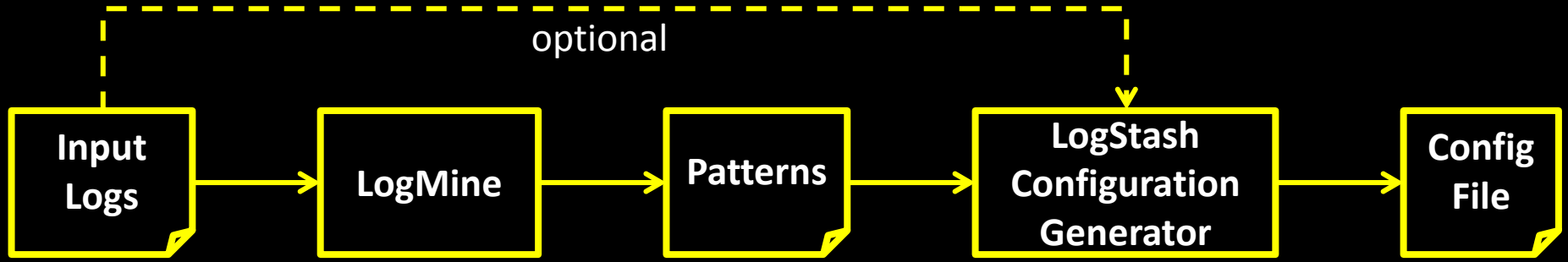



Note: For a fast log parser (up to 41x faster than Logstash), which can handle large number of GROK patterns, please see the LogLens [ICDCS 2018] paper at slide #67.

# Outline

- Pattern Generation Algorithm
- Demo Usecases

# LogMine + LogStash : Configuration Generation




**WorkFlow**

# Demo 1: Datacenter Power Usage

```
Mar  3 16:25:03 envctl APC_PDU_LEGAMPS: [INFO] PDU=pdu-2g04-apc-rack3a Leg=1 Amps=4.6
Mar  3 16:25:03 envctl APC_PDU_LEGAMPS: [INFO] PDU=pdu-2g04-rack1a Leg=1 Amps=4.1
.
.
.
Mar  3 16:30:03 envctl APC_PDU_LEGAMPS: [INFO] PDU=pdu-2g04-apc-rack1a Leg=2 Amps=1.2
Mar  3 16:30:03 envctl APC_PDU_LEGAMPS: [INFO] PDU=pdu-2g04-apc-rack1a Leg=3 Amps=0.0
```

Sample Logs

Input Logs: 14,976 lines

Patterns: 1

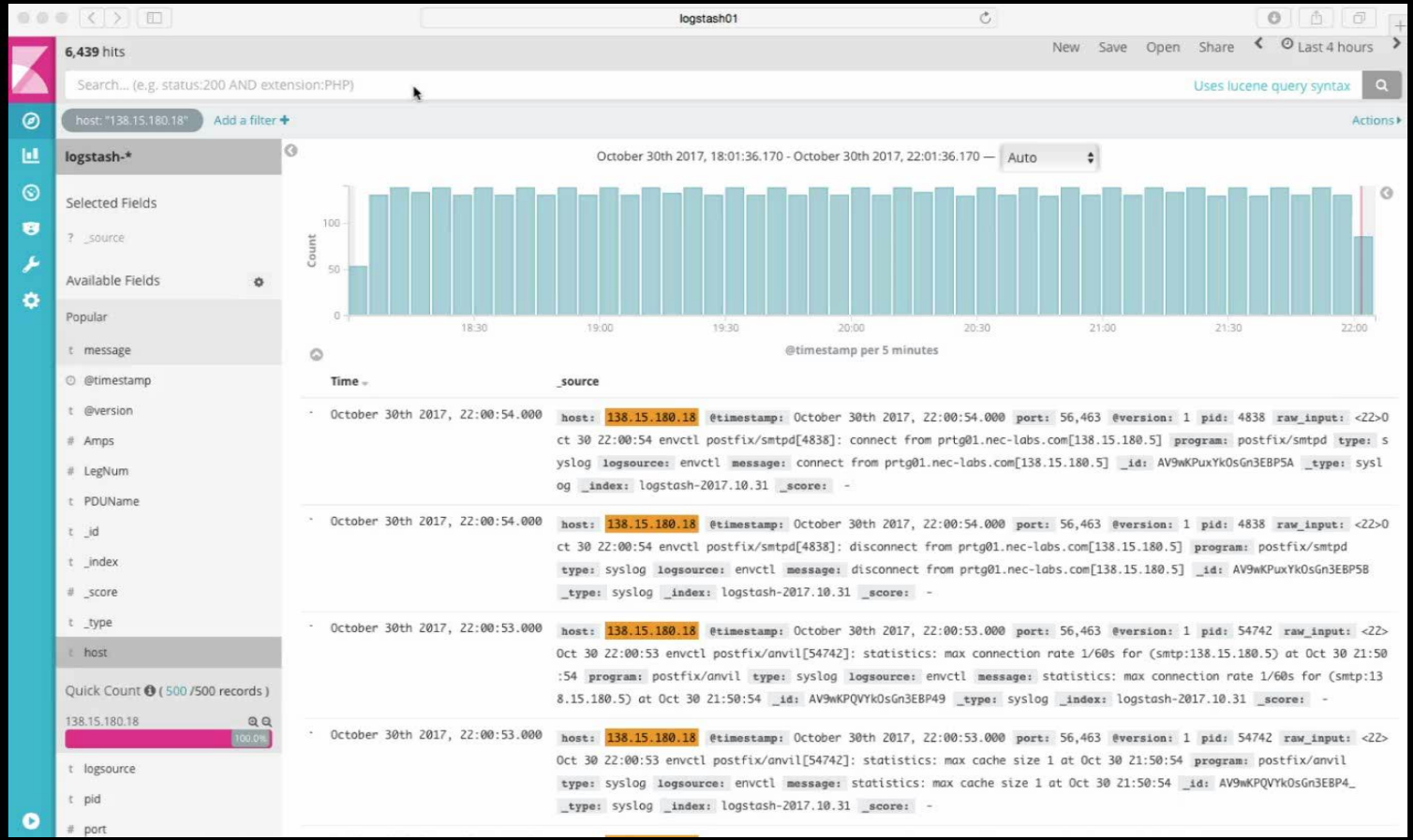
Tokenization Delimiter : “=“

Time Taken: 9 seconds (Single core)

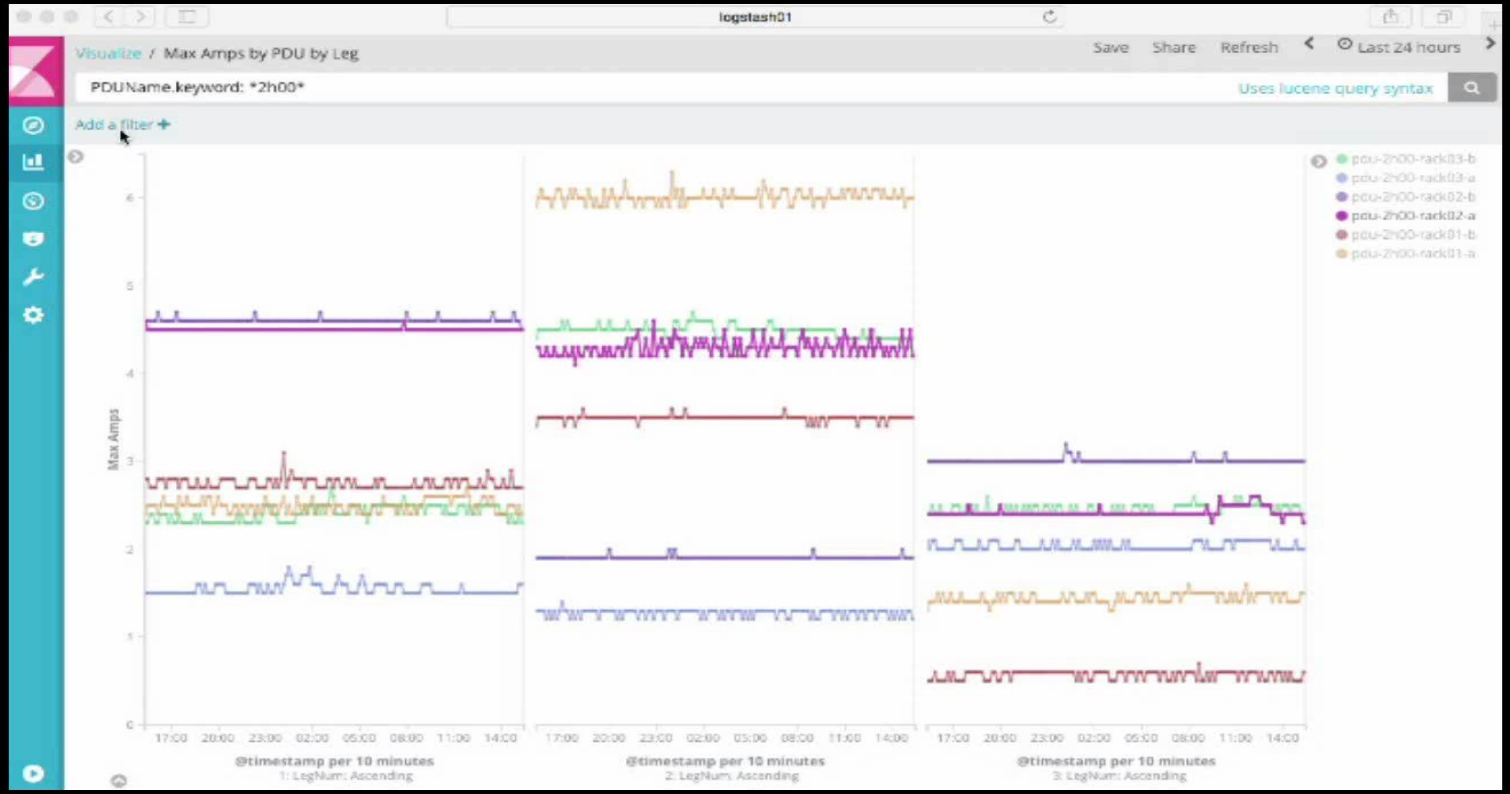
# Demo 1: LogStash Parsing Configuration

```
filter {
       mutate {
         add_field => { "raw_input" => "%{message}" }
       }

       mutate {
         gsub => [
           "message", "=", " \0 ",
           "message", "\s+", " "
         ]
       }

       grok {
         match => { "message" => "^(?<logTime>%{MONTH:month}
%{MONTHDAY:day} %{HOUR:hour}:%{MINUTE:minute}:%{SECOND:second}) envctl
APC_PDU_LEGAMPS: \[INFO\] PDU = %{NOTSPACE:PDUName} Leg =
%{NUMBER:LegNum:int} Amps = %{NUMBER:Amps:float}$"}
        }

        date {
          match => [ "logTime", "MMM dd HH:mm:ss" ]
          target => "@timestamp"
        }

        mutate {
          remove_field => ['year']
          remove_field => ['month']
          remove_field => ['day']
          remove_field => ['hour']
          remove_field => ['minute']
          remove_field => ['second']
          remove_field => ['logTime']
        }
}
```

# Demo 1: Datacenter Power Usage



# Demo 1: Datacenter Power Usage



# Demo 2: Custom Application Log

```
(0):GetFormControl():2[25 21:39:21] SQL SELECT TABLE: tblFormControl WHERE: tblFormControl.oFCID='6a602aaa-9afd-4e2c-95e9-ee900dde4b50'

(0):_GetObjects():1[25 21:39:21] KVIEW running: 'My Releases Needing Followup'

(0):_GetObjects():2[25 21:39:21] SQL SELECT TABLE: tblContent WHERE:  oPID='ad1aa290-01ae-4edd-989c-1cee2ba63707'  AND ( ( (  ( ( ( oID IN (SELECT oID
FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='6a602aaa-9afd-4e2c-95e9-ee900dde4b50' AND ((tblFormData.tValue IS NOT NULL AND
tblFormData.tValue > '1799-01-01T00:00:00.000' AND tblFormData.tValue < '2200-01-01T00:00:00.000')) ))) AND (( (nType!=15 OR oID IN (SELECT oFORMINSTID
FROM tblFormInstance WHERE tblFormInstance.oFORMID='3ebee358-2087-43d4-908b-df9ed04e74cc')) AND (nType!=14 OR tblContent.oID='3ebee358-2087-43d4-908b-
df9ed04e74cc') )) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='7e68b547-0869-4a56-a664-26b32d0b5804' AND
((tblFormData.tValue<='2017-10-26T03:59:59.000' OR tblFormData.tValue IS NULL)) ))) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE
oFORMINSTID=tblContent.oID AND oFCID='e28c6d82-532d-4618-a0a8-d62a15e00731' AND (tblFormData.sValue=N'dadf4506-2995-42c4-8616-cb43786fa382') ))) AND (
( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='2a004b8d-16ef-4973-8ec8-be7db392e436' AND ((tblFormData.sValue<>N'Y'
OR tblFormData.sValue IS NULL)) ))) ) ) )  AND (nType!=15 OR oID IN (SELECT oFORMINSTID FROM tblFormInstance WHERE oFORMID='3ebee358-2087-43d4-908b-
df9ed04e74cc') ) AND 1=1 AND (  nType=15 ) ) AND  (oID IN (SELECT oID FROM tblPerm WHERE (oGrantID='dadf4506-2995-42c4-8616-cb43786fa382' OR
oGrantID='[Authenticated]' OR oGrantID='[Anonymous]' OR oGrantID IN (SELECT oParent FROM tblMembership WHERE oChild='dadf4506-2995-42c4-8616-
cb43786fa382')) AND fRead=1) )  AND (nSubType!=2 AND nSubType!=1 AND nSubType!=4 AND nSubType!=5) AND (nType!=15 OR nVersion!=0)

(0):_GetObjects():2[25 21:39:21] SQL SELECT TABLE: tblFormData WHERE: oFORMINSTID = '418f38ce-a35e-47db-8e1c-88fc7eb09de3' AND oFCID IN ('fe53e626-
13ae-4206-8bc7-178cbc69b866', '6a602aaa-9afd-4e2c-95e9-ee900dde4b50', '1bfb5785-4f29-488b-8d09-c42faef48fee', '611e6c07-c8ba-44c5-b745-485e9faddcb4',
'3d8dfd3d-2c62-4c19-8cb5-a3bec8bf729b', '7e68b547-0869-4a56-a664-26b32d0b5804')

(0):bpPage_PreInit():(w3wp.exe,0x1bd8,17):1[25 21:40:22] Page: http://app.1nec-labs.com/kv_right.aspx?kvid=3b676138-0fe6-4dbd-a20b-
49fe5b07725d&showmax=1, request URL: http://app1.nec-labs.com/kv_right.aspx?kvid=3b676138-0fe6-4dbd-a20b-49fe5b07725d&showmax=1, remote IP:
138.15.207.74
```

Sample Logs

Input Logs: 242,232 lines

Patterns: 339

Tokenization Delimiter: “[=]“

Time Taken: 27 seconds (Single core)

# Demo 2: Custom Application Log

```
(0):GetFormControl():2[25 21:39:21] SQL SELECT TABLE: tblFormControl WHERE: tblFormControl.oFCID='6a602aaa-9afd-4e2c-95e9-ee900dde4b50'

(0):_GetObjects():1[25 21:39:21] KVIEW running: 'My Releases Needing Followup'

(0):_Get:2[25 21:39:21] SQL ontent WHERE:  oPID='ad1aa290-01ae-4edd-989c-1cee2ba63707'  AND ( ( (  ( ( ( oID IN (SELECT oID
FROM tbl FCID='6a602aaa-9afd-4e2c-95e9-ee900dde4b50' AND ((tblFormData.tValue IS NOT NULL AND
tblFormData.tValue > '1799-01-01T00:00:00.000' AND tblFormData.tValue < '2200-01-01T00:00:00.000')) ))) AND (( (nType!=15 OR oID IN (SELECT oFORMINSTID
FROM tblFormInstance WHERE tblFormInstance.oFORMID='3ebee358-2087-43d4-908b-df9ed04e74cc')) AND (nType!=14 OR tblContent.oID='3ebee358-2087-43d4-908b-
df9ed04e74cc') )) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='7e68b547-0869-4a56-a664-26b32d0b5804' AND
((tblFormData.tValue<='2017-10-26T03:59:59.000' OR tblFormData.tValue IS NULL)) ))) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE
oFORMINSTID=tblContent.oID AND oFCID='e28c6d82-532d-4618-a0a8-d62a15e00731' AND (tblFormData.sValue=N'dadf4506-2995-42c4-8616-cb43786fa382') ))) AND (
( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='2a004b8d-16ef-4973-8ec8-be7db392e436' AND ((tblFormData.sValue<>N'Y'
OR tblFormData.sValue IS NULL)) ))) ) ) )  AND (nType!=15 OR oID IN (SELECT oFORMINSTID FROM tblFormInstance WHERE oFORMID='3ebee358-2087-43d4-908b-
df9ed04e74cc') ) AND 1=1 AND (  nType=15 ) ) AND  (oID IN (SELECT oID FROM tblPerm WHERE (oGrantID='dadf4506-2995-42c4-8616-cb43786fa382' OR
oGrantID='[Authenticated]' OR oGrantID='[Anonymous]' OR oGrantID IN (SELECT oParent FROM tblMembership WHERE oChild='dadf4506-2995-42c4-8616-
cb43786fa382')) AND fRead=1) )  AND (nSubType!=2 AND nSubType!=1 AND nSubType!=4 AND nSubType!=5) AND (nType!=15 OR nVersion!=0)

(0):_GetObjects():2[25 21:39:21] SQL SELECT TABLE: tblFormData WHERE: oFORMINSTID = '418f38ce-a35e-47db-8e1c-88fc7eb09de3' AND oFCID IN ('fe53e626-
13ae-4206-8bc7-178cbc69b866', '6a602aaa-9afd-4e2c-95e9-ee900dde4b50', '1bfb5785-4f29-488b-8d09-c42faef48fee', '611e6c07-c8ba-44c5-b745-485e9faddcb4',
'3d8dfd3d-2c62-4c19-8cb5-a3bec8bf729b', '7e68b547-0869-4a56-a664-26b32d0b5804')

(0):bpPage_PreInit():(w3wp.exe,0x1bd8,17):1[25 21:40:22] Page: http://app1.nec-labs.com/kv_right.aspx?kvid=3b676138-0fe6-4dbd-a20b-
49fe5b07725d&showmax=1, request URL: http://app1.nec-labs.com/kv_right.aspx?kvid=3b676138-0fe6-4dbd-a20b-49fe5b07725d&showmax=1, remote IP:
138.15.207.74
```

Sample Logs

Input Logs: 242,232 lines

Patterns: 339

Tokenization Delimiter: “[=]“

Time Taken: 27 seconds (Single core)

# Demo 2: Pattern-Tree Stats

| Level | Patterns | Cost |
|---|---|---|
| 1 | 2582 | 0 |
| 2 | 339 | 0 |
| 3 | 256 | 297396 |
| 4 | 231 | 327373 |
| 5 | 230 | 327397 |
| 6 | 219 | 336073 |
| 7 | 217 | 336127 |
| 8 | 214 | 336153 |
| 9 | 213 | 336175 |
| 10 | 211 | 338036 |
| 11 | 210 | 338074 |
| 12 | 195 | 340653 |
| 13 | 176 | 353253 |
| 14 | 132 | 456600 |
| 15 | 121 | 641338 |
| 16 | 106 | 655485 |
| 17 | 82 | 669673 |
| 18 | 57 | 742742 |
| 19 | 52 | 745047 |
| 20 | 48 | 749203 |
| 21 | 39 | 761506 |
| 22 | 34 | 794590 |
| 23 | 30 | 795275 |
| 24 | 24 | 1171696 |
| 25 | 20 | 1333119 |
| 26 | 9 | 939438 |
| 27 | 1 | 726696 |

We put no upper limit on pattern count., therefore Level 2 is selected as the final output because it has the minimum cost with minimum number of patterns.

# Demo 2: Sample Parsing Pattern

(0):_GetObjects():2[25 21:39:21] SQL SELECT TABLE: tblContent WHERE: oPID='ad1aa290-01ae-4edd-989c-1cee2ba63707' AND ( ( ( ( ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='6a602aaa-9afd-4e2c-95e9-ee900dde4b50' AND ((tblFormData.tValue IS NOT NULL AND tblFormData.tValue > '1799-01-01T00:00:00.000' AND tblFormData.tValue < '2200-01-01T00:00:00.000')) ))) AND (( (nType!=15 OR oID IN (SELECT oFORMINSTID FROM tblFormInstance WHERE tblFormInstance.oFORMID='3ebee358-2087-43d4-908b-df9ed04e74cc')) AND (nType!=14 OR tblContent.oID='3ebee358-2087-43d4-908b-df9ed04e74cc') )) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='7e68b547-0869-4a56-a664-26b32d0b5804' AND ((tblFormData.tValue<='2017-10-26T03:59:59.000' OR tblFormData.tValue IS NULL)) ))) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='e28c6d82-532d-4618-a0a8-d62a15e00731' AND (tblFormData.sValue=N'dadf4506-2995-42c4-8616-cb43786fa382') ))) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='2a004b8d-16ef-4973-8ec8-be7db392e436' AND ((tblFormData.sValue<>N'Y' OR tblFormData.sValue IS NULL)) ))) ) ) ) AND (nType!=15 OR oID IN (SELECT oFORMINSTID FROM tblFormInstance WHERE oFORMID='3ebee358-2087-43d4-908b-df9ed04e74cc') ) AND 1=1 AND ( nType=15 ) ) AND (oID IN (SELECT oID FROM tblPerm WHERE (oGrantID='dadf4506-2995-42c4-8616-cb43786fa382' OR oGrantID='[Authenticated]' OR oGrantID='[Anonymous]' OR oGrantID IN (SELECT oParent FROM tblMembership WHERE oChild='dadf4506-2995-42c4-8616-cb43786fa382')) AND fRead=1) ) AND (nSubType!=2 AND nSubType!=1 AND nSubType!=4 AND nSubType!=5) AND (nType!=15 OR nVersion!=0)

grok {
match => { "message" => "^%{NOTSPACE:P199NS1} \[ **(?<logTime>%{MONTHDAY:day} %{HOUR:hour}:%{MINUTE:minute}:%{SECOND:second})** \] SQL SELECT TABLE: tblContent WHERE: oPID = 'ad1aa290\-01ae\-4edd\-989c\-1cee2ba63707' AND \( \( \( \( \( \( oID IN \(SELECT oID FROM tblFormData WHERE oFORMINSTID = tblContent\.oID AND oFCID = '6a602aaa\-9afd\-4e2c\-95e9\-ee900dde4b50' AND \(\(tblFormData\.tValue IS NOT NULL AND tblFormData\.tValue > '1799\-01\-01T00:00:00\.000' AND tblFormData\.tValue < '2200\-01\-01T00:00:00\.000'\)\) \)\)\)\) AND \(\( \(nType! = %{NUMBER:P199F1:int} OR oID IN \(SELECT oFORMINSTID FROM tblFormInstance WHERE tblFormInstance\.oFORMID = '3ebee358\-2087\-43d4\-908b\-df9ed04e74cc'\)\) AND \(nType! = %{NUMBER:P199F2:int} OR tblContent\.oID = '3ebee358\-2087\-43d4\-908b\-df9ed04e74cc'\) \)\) AND \( \( oID IN \(SELECT oID FROM tblFormData WHERE oFORMINSTID = tblContent\.oID AND oFCID = '7e68b547\-0869\-4a56\-a664\-26b32d0b5804' AND \(\(tblFormData\.tValue< = %{NOTSPACE:P199NS2:int} OR tblFormData\.tValue IS NULL\)\) \)\)\) AND \( \( oID IN \(SELECT oID FROM tblFormData WHERE oFORMINSTID = tblContent\.oID AND oFCID = 'e28c6d82\-532d\-4618\-a0a8\-d62a15e00731' AND \(tblFormData\.sValue = %{NOTSPACE:P199NS3} \)\)\) AND \( \( oID IN \(SELECT oID FROM tblFormData WHERE oFORMINSTID = tblContent\.oID AND oFCID = '2a004b8d\-16ef\-4973\-8ec8\-be7db392e436' AND \(\(tblFormData\.sValue<>N'Y' OR tblFormData\.sValue IS NULL\)\) \)\)\)\) \) \) \) AND \(nType! = %{NUMBER:P199F3:int} OR oID IN \(SELECT oFORMINSTID FROM tblFormInstance WHERE oFORMID = '3ebee358\-2087\-43d4\-908b\-df9ed04e74cc'\) \) AND %{NUMBER:P199F4:int} = %{NUMBER:P199F5:int} AND \( nType = %{NUMBER:P199F6:int} \) \) AND \(oID IN \(SELECT oID FROM tblPerm WHERE \(oGrantID = %{NOTSPACE:P199NS4} OR oGrantID = ' \[ Authenticated \] ' OR oGrantID = ' \[ Anonymous \] ' OR oGrantID IN \(SELECT oParent FROM tblMembership WHERE oChild = %{NOTSPACE:P199NS5} AND fRead = 1\) \) AND \(nSubType! = %{NUMBER:P199F7:int} AND nSubType! = %{NUMBER:P199F8:int} AND nSubType! = %{NUMBER:P199F9:int} AND nSubType! = 5\) AND \(nType! = %{NUMBER:P199F10:int} OR nVersion! = 0\) $"}
remove_field => [ "tags" ]
add_tag => [ "pattern199" ]
tag_on_failure => []
}

```
date {
      match => [ "logTime", "dd HH:mm:ss" ]
      target => "@timestamp"
    }
```

# Demo 2: Sample Parsed Output

(0):_GetObjects():2[25 21:39:21] SQL SELECT TABLE: tblContent WHERE: oPID='ad1aa290-01ae-4edd-989c-1cee2ba63707' AND ( ( ( ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='6a602aaa-9afd-4e2c-95e9-ee900dde4b50' AND ((tblFormData.tValue IS NOT NULL AND tblFormData.tValue > '1799-01-01T00:00:00.000' AND tblFormData.tValue < '2200-01-01T00:00:00.000')) ))) AND (( (nType!=15 OR oID IN (SELECT oFORMINSTID FROM tblFormInstance WHERE tblFormInstance.oFORMID='3ebee358-2087-43d4-908b-df9ed04e74cc')) AND (nType!=14 OR tblContent.oID='3ebee358-2087-43d4-908b-df9ed04e74cc') )) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='7e68b547-0869-4a56-a664-26b32d0b5804' AND ((tblFormData.tValue<='2017-10-26T03:59:59.000' OR tblFormData.tValue IS NULL)) ))) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='e28c6d82-532d-4618-a0a8-d62a15e00731' AND (tblFormData.sValue=N'dadf4506-2995-42c4-8616-cb43786fa382') ))) AND ( ( oID IN (SELECT oID FROM tblFormData WHERE oFORMINSTID=tblContent.oID AND oFCID='2a004b8d-16ef-4973-8ec8-be7db392e436' AND ((tblFormData.sValue<>N'Y' OR tblFormData.sValue IS NULL)) ))) ) ) ) AND (nType!=15 OR oID IN (SELECT oFORMINSTID FROM tblFormInstance WHERE oFORMID='3ebee358-2087-43d4-908b-df9ed04e74cc') ) AND 1=1 AND ( nType=15 ) ) AND (oID IN (SELECT oID FROM tblPerm WHERE (oGrantID='dadf4506-2995-42c4-8616-cb43786fa382' OR oGrantID='[Authenticated]' OR oGrantID='[Anonymous]' OR oGrantID IN (SELECT oParent FROM tblMembership WHERE oChild='dadf4506-2995-42c4-8616-cb43786fa382')) AND fRead=1) ) AND (nSubType!=2 AND nSubType!=1 AND nSubType!=4 AND nSubType!=5) AND (nType!=15 OR nVersion!=0)


```
{
  "type": "lisa-demo2",
  "tags": ["pattern199"],
  "P199NS1": "(0):_GetObjects():2",
  "@timestamp": "2017-01-25T21:39:21.000Z",
  "P199F1": "15",
  "P199F2": "14",
  "P199NS2": "'2017-10-26T03:59:59.000'",
  "P199NS3": "N'dadf4506-2995-42c4-8616-cb43786fa382')",
  "P199F3": "15",
  "P199F4": "1",
  "P199F5": "1",
  "P199F6": "15",
  "P199NS4": "'dadf4506-2995-42c4-8616-cb43786fa382'",
  "P199NS5": "'dadf4506-2995-42c4-8616-cb43786fa382'))“,
  "P199F7": "2",
  "P199F8": "1",
  "P199F9": "4",
  "P199F10": "15"
}
```

# Summary

1. LogMine is fast and scalable ☺
2. It can work with no (or minimal) human Involvement ☺
3. It is flexible because user can cherry-pick any desired pattern-set ☺
4. It is the core component of a real product ☺

*All these benefits come from the fact that*
*LogMine is customized for logs.*

# References


- LogMine [CIKM 2016]: ***LogMine: Fast Pattern Recognition for the Log Analytics***. Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Geoff Jiang, Abdullah Mueen, In CIKM Conference, 2016.
  - *Paper link: www.cs.unm.edu/~mueen/Papers/LogMine.pdf*

- LogLens [ICDCS 2018]: ***LogLens: A Real-time Log Analysis System***. Biplob Debnath, M. Solaimani, Mohammed Gulzar, Nipun Arora, Hui Zhang, Jianwu Xu, Cristian Lumezanu, Guofei Jiang, Latifur Khan, To appear in ICDCS conference, 2018.

# Contributors

- Hossein Hammoni
- Jianwu Xu
- Hui Zhang
- Guofei Jiang
- Nipun Arora

# Thanks!

biplob@nec-labs.com
wdennis@nec-labs.com