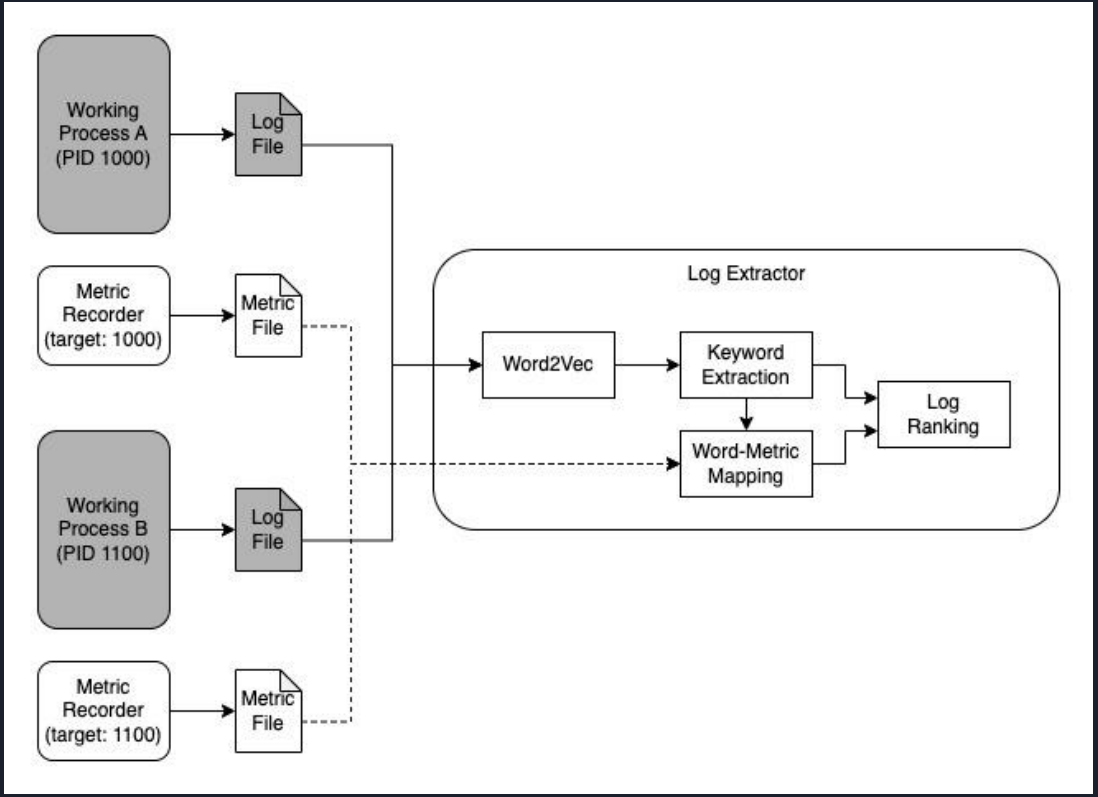
Performance Diagnosis with Logs and Metrics

Jae Jimmy Wong

Objective

- Automatic log analysis and filtering
  - Reduce the need of manual work
- Utilize system metric to identify performance issue
  - Select CPU and memory usage as target
- Not rely on source code or application specific feature
  - Applicable to more programming languages

First Attempt: Keyword Weighting


Working Process A (PID 1000)
Log File
Metric Recorder (target: 1000)
Metric File
Working Process B (PID 1100)
Log File
Metric Recorder (target: 1100)
Metric File
Log Extractor
Word2Vec
Keyword Extraction
Word-Metric Mapping
Log Ranking

Algorithm - Keyword Weighting

- Each line of log is represented by a fixed number of keyword
    - Keyword is determined by the representatives of Feature Agglomeration on the embedding vector of each word
- When the CPU usage change, all keywords appeared in that second gain additional weight
    - The log with most weighted keywords has highest score

Evaluation - Keyword Weighting

- Method
  - Signal injection to trigger hang/slowdown bug
  - Manual identify the most related line of log
- The position of most related log while sorting by score
  - HADOOP-11252: 365/6048 (7th percentile)
  - HADOOP-9106: 398/780 (52nd percentile)

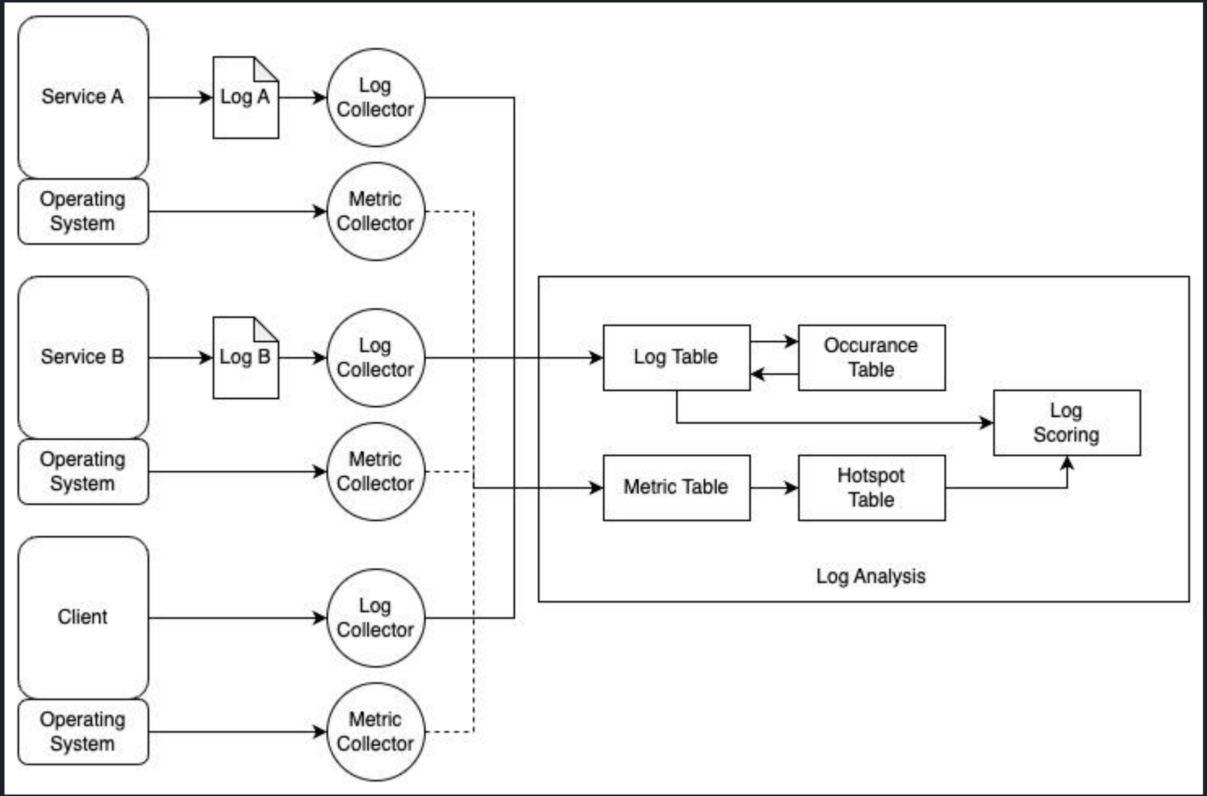
Limitation - Keyword Weighting

- The length of log lines varies a lot so a fixed number of keywords sometimes result in losing information
- Many of the words in logs are not included in pre-trained embedding vector model
- Logs with more preposition have unwanted advantage

Log Scoring

- Score = Hotness x Uniqueness
  - Logs that are far from hotspots have very low hotness
  - Logs that repeat twice or more have very low uniqueness
- Goal
  - Only logs that are performance related and non-repetitive will have higher score

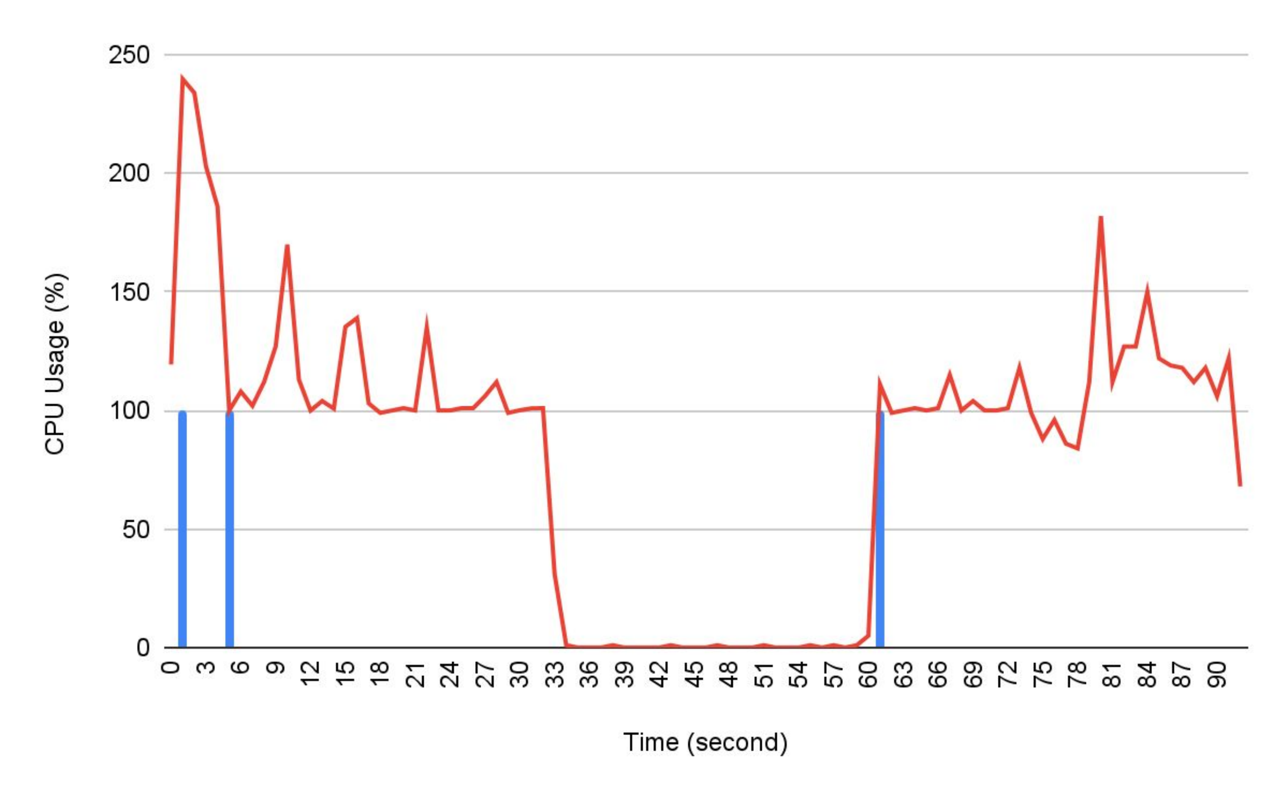
Second Attempt - Two Indicator



Resource Hotness

- Hotspot
    - Time points when CPU or memory has a big change
        - More than mean plus 3 stdev. of all changes
- Hotness
    - How close the timing is to a hotspot
        - PDF of normal distribution centered at hotspot

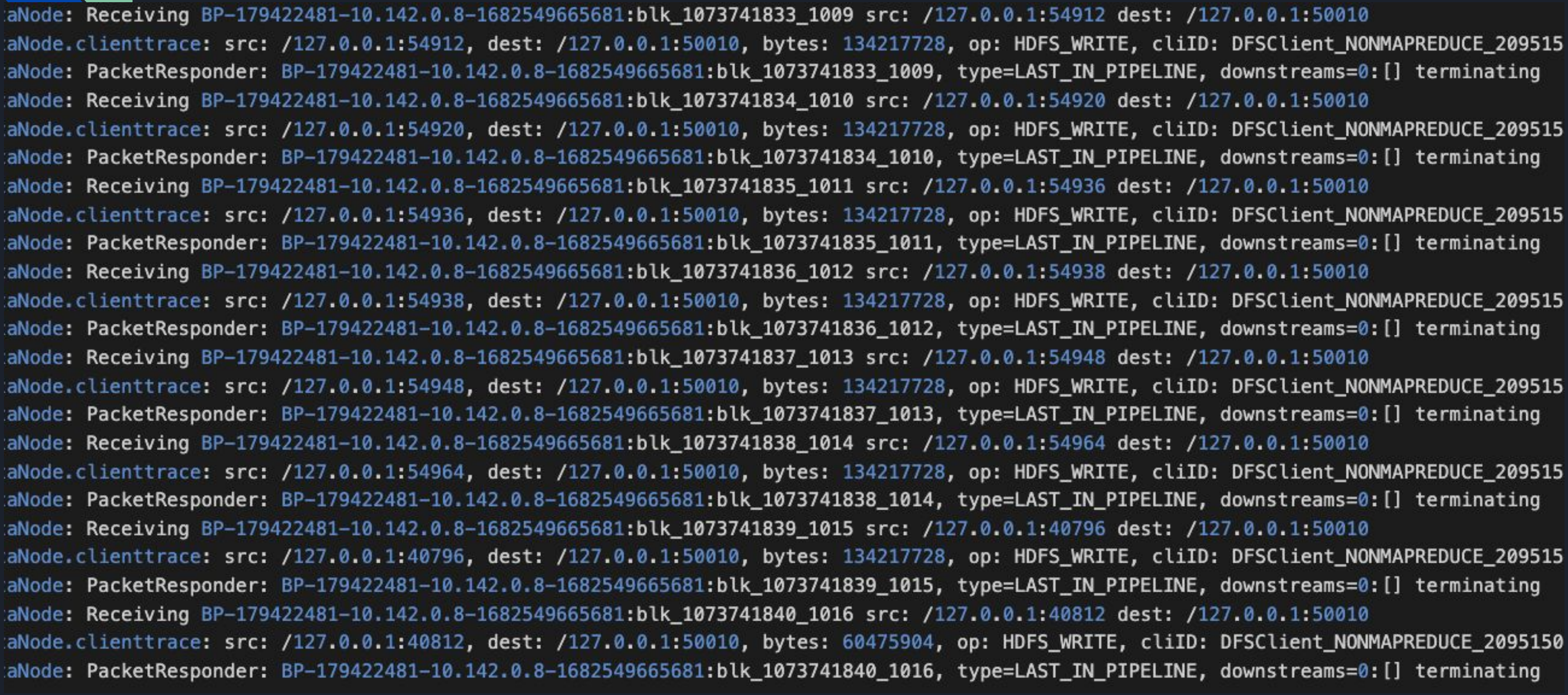
Example - Hotspot



Log Uniqueness

- Repetitive Logs are not helpful
  - Value the uniqueness of log
- Each line is split into words with only alphabet
  - Uniqueness(word) = 1 / Occurrence(word)
  - The uniqueness of each line is the mean uniqueness of its words

Example - Repetitive Log



```
aNode: Receiving BP-179422481-10.142.0.8-1682549665681:blk_1073741833_1009 src: /127.0.0.1:54912 dest: /127.0.0.1:50010
aNode.clienttrace: src: /127.0.0.1:54912, dest: /127.0.0.1:50010, bytes: 134217728, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_209515
aNode: PacketResponder: BP-179422481-10.142.0.8-1682549665681:blk_1073741833_1009, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
aNode: Receiving BP-179422481-10.142.0.8-1682549665681:blk_1073741834_1010 src: /127.0.0.1:54920 dest: /127.0.0.1:50010
aNode.clienttrace: src: /127.0.0.1:54920, dest: /127.0.0.1:50010, bytes: 134217728, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_209515
aNode: PacketResponder: BP-179422481-10.142.0.8-1682549665681:blk_1073741834_1010, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
aNode: Receiving BP-179422481-10.142.0.8-1682549665681:blk_1073741835_1011 src: /127.0.0.1:54936 dest: /127.0.0.1:50010
aNode.clienttrace: src: /127.0.0.1:54936, dest: /127.0.0.1:50010, bytes: 134217728, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_209515
aNode: PacketResponder: BP-179422481-10.142.0.8-1682549665681:blk_1073741835_1011, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
aNode: Receiving BP-179422481-10.142.0.8-1682549665681:blk_1073741836_1012 src: /127.0.0.1:54938 dest: /127.0.0.1:50010
aNode.clienttrace: src: /127.0.0.1:54938, dest: /127.0.0.1:50010, bytes: 134217728, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_209515
aNode: PacketResponder: BP-179422481-10.142.0.8-1682549665681:blk_1073741836_1012, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
aNode: Receiving BP-179422481-10.142.0.8-1682549665681:blk_1073741837_1013 src: /127.0.0.1:54948 dest: /127.0.0.1:50010
aNode.clienttrace: src: /127.0.0.1:54948, dest: /127.0.0.1:50010, bytes: 134217728, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_209515
aNode: PacketResponder: BP-179422481-10.142.0.8-1682549665681:blk_1073741837_1013, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
aNode: Receiving BP-179422481-10.142.0.8-1682549665681:blk_1073741838_1014 src: /127.0.0.1:54964 dest: /127.0.0.1:50010
aNode.clienttrace: src: /127.0.0.1:54964, dest: /127.0.0.1:50010, bytes: 134217728, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_209515
aNode: PacketResponder: BP-179422481-10.142.0.8-1682549665681:blk_1073741838_1014, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
aNode: Receiving BP-179422481-10.142.0.8-1682549665681:blk_1073741839_1015 src: /127.0.0.1:40796 dest: /127.0.0.1:50010
aNode.clienttrace: src: /127.0.0.1:40796, dest: /127.0.0.1:50010, bytes: 134217728, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_209515
aNode: PacketResponder: BP-179422481-10.142.0.8-1682549665681:blk_1073741839_1015, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
aNode: Receiving BP-179422481-10.142.0.8-1682549665681:blk_1073741840_1016 src: /127.0.0.1:40812 dest: /127.0.0.1:50010
aNode.clienttrace: src: /127.0.0.1:40812, dest: /127.0.0.1:50010, bytes: 60475904, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_2095150
aNode: PacketResponder: BP-179422481-10.142.0.8-1682549665681:blk_1073741840_1016, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
```

Evaluation - Two Indicator

Bug ID	The Rank of Most Related Logs	Total
HDFS-11252	3	404
HDFS-9106	1	389
HDFS-7005	3	30
HDFS-1490	3	75

Evaluation - Two Indicator

- Analysis Time
  - Setup: GCP E2 Machine, 4 Core, 16GB RAM
  - Zanbil.ir web server access logs
    - Size: 3.52GB
    - Result: 249.9 seconds (4.2 minutes)

Related Work

- Detecting Large-Scale System Problems by Mining Console Logs
    - Author: Xu et al.
    - Built source code analyzer to parse log
- PerfSig
    - Author: He et al.
    - Utilize function call traces in the causal analysis

Future Work

- Experiment with logs and metrics from production servers
- Compare different strategies for picking hotspot
- Add more types of metrics into consideration
    - Network throughput, packet rate, etc.

Lesson Learned

- Reproducing performance bug with a black-box test require thorough knowing a system.
- Automating bug reproduce is valuable especially for system with multiple components.
- Adding techniques into a algorithm without fully understanding their implication may not be a good idea.