Detecting Large-Scale System Problems by Mining Console Logs

Wei Xu*  Ling Huang†
Armando Fox*  David Patterson*  Michael Jordan*

*UC Berkeley  †Intel Labs Berkeley
Why console logs?

- Detecting problems in large scale Internet services often requires detailed instrumentation
- Instrumentation can be costly to insert & maintain
  - High code churn
  - Often combine open-source building blocks that are not all instrumented
- Can we use console logs in lieu of instrumentation?

  + Easy for developer, so nearly all software has them
  - Imperfect: not originally intended for instrumentation
Result preview

- 200 nodes, >24 million lines of logs
- Abnormal log segments
- A single page visualization

- Fully automatic process without any manual input
Our approach and contribution

- A general methodology for processing console logs automatically
- Validation on two real systems
Key insights for analyzing logs

• The log contains the necessary information to create features
  • Identifiers
  • State variables
  • Correlations among messages

```
receiving blk_1
received blk_1
```

```
receiving blk_2
NORMAL
ERROR
```

• Console logs are inherently structured
  • Determined by log printing statement
Step 1: Parsing

- Free text → semi-structured text
- Basic ideas

Receiving block blk_1
Log.info("Receiving block " + blockId);
Receiving block (.*\[blockId\])

Type: Receiving block
Variables: blockId(String)=blk_1

- Non-trivial in object oriented languages
  - Needs type inference on the entire source tree
- Highly accurate parsing results
Step 2: Feature creation - Message count vector

- Identifiers are widely used in logs
  - Variables that identify objects manipulated by the program
  - file names, object keys, user ids

- Grouping by identifiers
  - Similar to execution traces

- Identifiers can be discovered automatically
Feature creation – Message count vector example

- Numerical representation of these “traces”
  - Similar to bag of words model in information retrieval
Step 3: Machine learning – PCA anomaly detection

- Most of the vectors are normal
- Detecting abnormal vectors
  - Principal Component Analysis (PCA) based detection
  - PCA captures normal patterns in these vectors
- Based on *correlations* among dimensions of the vectors

0 2 2 1 2 0 0 2 0 0 0 0 0 0 0 0

0 2 2 receiving blk_1
received blk_1

0 2 2 receiving blk_2
NORMAL
ERROR
Evaluation setup

- Experiment on Amazon’s EC2 cloud
  - 203 nodes x 48 hours
  - Running standard map-reduce jobs
  - ~24 million lines of console logs
  - ~575,000 HDFS blocks
- 575,000 vectors
- ~680 distinct ones
- Manually labeled each distinct cases
  - Normal/abnormal
  - Tried to learn why it is abnormal
  - For evaluation only
### PCA detection results

<table>
<thead>
<tr>
<th>Anomaly Description</th>
<th>Actual</th>
<th>Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Forgot to update namenode for deleted block</td>
<td>4297</td>
<td>4297</td>
</tr>
<tr>
<td>2. Write block exception then client give up</td>
<td>3225</td>
<td>3225</td>
</tr>
<tr>
<td>3. Failed at beginning, no block written</td>
<td>2950</td>
<td>2950</td>
</tr>
<tr>
<td>4. Over-replicate-immediately-deleted</td>
<td>2809</td>
<td>2788</td>
</tr>
<tr>
<td>5. Received block that does not belong to any file</td>
<td>1240</td>
<td>1228</td>
</tr>
<tr>
<td>6. Redundant addStoredBlock request received</td>
<td>953</td>
<td>953</td>
</tr>
<tr>
<td>7. Trying to delete a block, but the block no longer exists on data node</td>
<td>724</td>
<td>650</td>
</tr>
<tr>
<td>8. Empty packet for block</td>
<td>476</td>
<td>476</td>
</tr>
<tr>
<td>9. Exception in receiveBlock for block</td>
<td>89</td>
<td>89</td>
</tr>
<tr>
<td>10. PendingReplicationMonitor timed out</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>11. Other anomalies</td>
<td>108</td>
<td>107</td>
</tr>
<tr>
<td><strong>Total anomalies</strong></td>
<td>16916</td>
<td>16808</td>
</tr>
</tbody>
</table>

| Normal blocks                                                                        | 558223 |

### False Positives

<table>
<thead>
<tr>
<th>Description</th>
<th>False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Normal background migration</td>
<td>1397</td>
</tr>
<tr>
<td>2. Multiple replica (for task/jobdesc files)</td>
<td>349</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1746</strong></td>
</tr>
</tbody>
</table>

How can we make the results easy for operators to understand?
Step 4: Visualizing results with decision tree

writeBlock # received exception

# Starting thread to transfer block # to #

#: Got exception while serving # to #:#

Unexpected error trying to delete block \
BlockInfo Not found in volumeMap

addStoredBlock request received for # on 
# size # But it does not belong to any file

# starting thread to transfer block # to #

#Verification succeeded for #

Receiving block # src: # dest: #

OK

ERROR

ERROR

ERROR

ERROR

OK

OK

ERROR
Future work

- Parsing
  - Extract templates from program binaries
  - Support more languages

- Feature creation and machine learning
  - Allow online detection
  - Cross application/layers logs
Summary

http://www.cs.berkeley.edu/~xuw/
Wei Xu  <xuw@cs.berkeley.edu>