TOWARDS DATA-DRIVEN AUTONOMICS IN DATA CENTERS

ALINA SIRBU, OZALP BABAOGLU

SUMMARIZED BY ARDA GUMUSALAN
MOTIVATION
MOTIVATION

• Human-interaction-dependent data centers are not sustainable for future data centers which are expected to be bigger scale than today’s data centers.
  • Exascale-> billion of billion servers per data center.
  • Unreasonable number of employee to oversee the system.
• Data centers in the future will fully be autonomic and human technicians will be limited to setting high-level goals and policies.
  • No more low-level operations
  • Level 5 of A survey of autonomic computing-degrees, models, and applications
MOTIVATION

• Applying traditional autonomic computing techniques to large data centers are problematic.
  • Current autonomic computing technologies are reactive.
  • They lack predictive capabilities to anticipate undesired states in advance.
GOAL AND CONTRIBUTION
GOAL

• Build a predictive model for node failures in data centers.
  • Develop a new generation of autonomies that is data-driven, predictive, and proactive based on holistic models which considers computer systems as an ecosystem.
  • This is the first step towards a more comprehensive predictor.
CONTRIBUTION

• Shows that modern data centers can be scaled to extreme dimensions by eliminating its reliance on human operators.
  • Provide a prediction model.
• Combining subsampling with bagging and prediction based voting.
  • Will be explained in the upcoming slides.
• Provide an example of BigQuery usage with quantitative evaluation of running times as a function of data size.
  • Computation times will be included.
  • Data mining related parts are not directly related to autonomic computing so some parts will be omitted.
BACKGROUND
RANDOM FOREST

• A technique that is commonly applied when the feature set is large or the data is unbalanced.

• Main idea: Ensemble weak-learners to form a strong learner.
  • Weak-learner is a decision tree

1. Sample $N$ cases at random with replacement to create a subset of the data.
   - This is called bagging.

2. At each node:
   1. Randomly chose $m$ predictor variables from all predictor variables.
   2. Determine the predictors that give the best binary split according to some objective function.
   3. At the next node, choose another $m$ variables at random from all predictor variables and do the same.
At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

RANDOM FOREST

• When a new input is entered to the system:
  • Run it on all of the trees.
  • The result can be obtained as:
    • Average or weighted average the results of each tree.
    • Majority voting.
• In order to have accurate results, inter-tree correlation should be low.
  • Tradeoff: smaller $m$ leads to lower inter-tree correlation but also lowers strength of an individual tree.

BUILDING THE FEATURE SET
FEATURE SET

- The workload trace published by Google has been analyzed.
  - 29 days for a cluster of 12,453 machines.
  - Includes: amount of resources used per task for every 5 minute intervals.
- Due to the size of uncompressed data, the authors used BigQuery.
  - 17 GB for over 1 billion records.
BASIC FEATURES

• Basic features obtained for each machine for every 5 minute interval:
  • Number of tasks currently running,
  • Number of tasks that have started in the last 5 minutes,
  • Number of tasks that have finished,
    • Evicted
    • Failed
    • Completed
    • Killed
    • Lost
  • CPU load,
  • Memory,
  • Disk time,
  • Cycles per instruction,
  • Memory access per instruction.
• Total of 12 basic features.
BASIC FEATURES

• For each 12 basic feature for each time step:
  • Consider the previous 6 time windows (30 minutes)
  • 12 x 6 = 72 features in total
• A separate table per feature.
  • First 5 features took between 139 to 939 seconds.
  • The remaining took 3585 to 9096 seconds.
SECOND LEVEL AGGREGATION

- A second level of aggregation for each basic feature compute:
  - Averages,
  - Standard deviation,
  - Coefficient of variation.
- Repeat this for 6 different running window sizes:
  - 1, 12, 24, 48, 72, 96 hours.
- 3 statistics x 12 basic features x 6 window sizes = 216 additional features.
  - The sizes of these tables were ranging between 143 GB to 12.5 TB
  - However, not time consuming.
THIRD LEVEL OF AGGREGATION

- Compute the correlation between for following 7 features:
  - Number of running tasks,
  - Number of started tasks,
  - Number of failed jobs,
  - CPU
  - Memory
  - Disk time,
  - Cycles per instruction.
- A total of 21 correlation pairs.
- Calculate this for each 6 of the time windows.
  - 1, 12, 24, 48, 72, 96 hours.
- Additional 21x6=126 features.
The table shows the mean (the standard deviation) of the required times per feature.

<table>
<thead>
<tr>
<th>Aggregation</th>
<th>Average, SD, CV</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1h</td>
<td>166 (all features)</td>
<td>45(6.5)</td>
</tr>
<tr>
<td>12h</td>
<td>864 (all features)</td>
<td>258.8(89.1)</td>
</tr>
<tr>
<td>24h</td>
<td>284.6(86.6)</td>
<td>395.6(78.9)</td>
</tr>
<tr>
<td>48h</td>
<td>593.6(399.2)</td>
<td>987.2(590)</td>
</tr>
<tr>
<td>72h</td>
<td>726.6(411.5)</td>
<td>1055.47(265.23)</td>
</tr>
<tr>
<td>96h</td>
<td>739.4(319.4)</td>
<td>1489.2(805.9)</td>
</tr>
</tbody>
</table>
ADDITIONAL FEATURES

• 2 new features are added.
  • Up-time for each machine
    • The time passed since the last ADD event
  • The number of REMOVE events for the entire cluster within the last hour.
• A total of 416 features.
CLASSIFICATION APPROACH
The features explained so far are used for classification with Random Forest (RF) classifier.

Data points are separated into two parts:

- SAFE: did not fail -> negative
- FAIL: failed -> positive
- All points with time to remove less than 24 hours were assigned to FAIL class.
  - If the gap between REMOVE and ADD event for the same machine was longer than 2 hours, it counted as a failure.
  - Out of 8,957 REMOVE events, 2,298 were considered as failure.

Subset of SAFE classes were extracted.

- 0.5% of the total random sampling
- Due to imbalanced data points.
- A total of 544,985 SAFE data points and 108,365 FAIL data points.

These 653,350 data points formed the basis of their predictive study.
FEATURE SELECTION MECHANISMS

The authors explored two types of future selection mechanism:

• *Principal component analysis*: a statistical procedure that converts a set of observations of *possibly correlated variables* into a set of values of *linearly uncorrelated variables* called *principle components*.

  • Theoretically, the top principle components can be used for classification, since those should contain the most important information.
  
  • However, the performance of principle components were not better than using original features.
FEATURE SELECTION MECHANISMS

- *Filtering the original features*: Filter the original features based on their correlation to the `time_to_remove` event.
  - Only the correlation values greater than a threshold is used.
  - The best performance was obtained with null threshold indicating that once again the best results were obtained using all the features.
- *None of the feature selection mechanisms performed better than the RF with the original features.*
  - RF itself performs feature selection when building decision trees.
BUILDING THE ENSEMBLE OF RF
The authors observed that the performance of individual classifiers (a single RF) was not satisfactory.

- They combined RFs to form a forest of forests.
- A known technique especially for imbalanced with rare events.
- Each RF had a varying number of decision trees -> between 2 and 15

Each RF trained with different data -> bagging

- Every time a new classifier is trained, a new training dataset is built by considering all the points in the positive class and a random subset of the negative class.

\( \textit{f}_{\text{safe}}: \) the ratio between SAFE and FAIL classes.

- \{0.25, 0.5, 1, 2, 3, 4\}
Repeat the algorithm 5 times resulting in 420 RF in the ensemble.

- 5 reps x 6 fsafe x 14 RF size

**Require:**

```python
train_pos, train_neg, Shuffle(), Train()
f_safe ← {0.25, 0.5, 1, 2, 3, 4}
tree_count ← {2..15}
classifiers ← {}
start ← 0
for all fs ∈ f_safe do
    for all tc ∈ tree_count do
        end ← start + fs * |train_pos|
        if end ≥ |train_neg| then
            start ← 0
            end ← start + fs * |train_pos|
            Shuffle(train_neg)
        end if
        train_data ← train_pos + train_neg[start: end]
        classifier ← Train(train_data, tc)
        append(classifiers, classifier)
        start ← end
    end for
end for
```
COMBINING STRATEGY

- **Precision weighted voting**: calculate the accuracy of each classifier on training data.

- Divide the test data into two halves:
  - *Individual test*: used to evaluate the precision of individual classifiers and calculate their weights.
  - *Ensemble test*: used to calculate the final evaluation of the ensemble.

- Precision is the fraction of points labeled fail that is actually a failure.
The classification of the ensemble is computed as the sum of all individual answers multiplied by their precision.

- Each individual answer is a discrete value but combination of them is continuous.
- Higher score indicates a higher probability of failure.

For all data points, there is a score

```
Require: classifiers, individual_test, ensemble_test
Require: Precision(), Classify()

classification_scores ← {}
weights ← {}
for all c ∈ classifiers do
    w ← Precision(c, individual_test)
    weights[c] ← w
end for
for all d ∈ ensemble_test do
    score ← 0
    for all c ∈ classifiers do
        score ← score + weights[c] * Classify(c, d)
    end for
    append(classification_scores, score)
end for
max ← Max(classification_scores)
for all s ∈ classification_scores do
    s ← s/max
end for
```
CLASSIFICATION
RESULTS
CROSS VALIDATION APPROACH

- The authors separated their data as *train* and *test* data.
  - Train over 10 days and test on the 12th.
  - First two days are omitted in order to decrease the effect on aggregated features.
  - Mimics how this model should be used in real life scenarios.
  - Took up to 9 hours with a relatively low spec computer.

![Diagram of CROSS VALIDATION APPROACH](image)
CROSS VALIDATION APPROACH

• PR: Precision vs. Recall.
  • Precision: positive predictive value.
  • Recall: sensitivity

• Ex: Suppose a program for recognizing dogs in scenes from a video that contains dogs and cats. There are a total of 9 dogs and some cats. Our program identifies 7 of the animals as dogs but only 4 out of those 7 are actually dogs.
  • Precision: 4/7
  • Recall: 4/9

https://en.wikipedia.org/wiki/Precision_and_recall
Evaluation was based on receiver operating characteristics (ROC) and precision recall (PR) curves.

- ROC curves plot True Positive Rate (TPR) versus False Positive Rate (FPR).
- PR curve displays the precision vs. recall which is equal to TPR.
- For both, the higher the value, the better it is.

A threshold value, s*, is needed to classify the score of the ensemble RF.

- By decreasing s* the number of true positives grows but so do the false positive.
CLASSIFICATION RESULTS

- For all days, AUROC values are greater than 0.75 and up to 0.97
- AUPR ranges between 0.38 and 0.87
- Lower performance at the beginning can be due to the fact that some of the aggregated data were incomplete (for those over 3 and 4 days).
• Performance of individual classifiers are displayed.
• Very low FPR-> good!
• In many cases, very low TPR-> bad!
TPR increases when the $f_{safe}$ parameter decreases
- But FPR increases and precision decreases.
- The results obtained with different $f_{safe}$ values are diverse which means it is suitable to use with ensemble approaches.
- In general, the points corresponding to the individual classifiers are below the ROC and PR curves describing the performance of the ensemble.
  - This proves that the ensemble method is better than the individual classifiers.
  - Except for TPR less than 0.2.
CLASSIFICATION RESULTS

• Conclusion: Recall (TPR) rate is between 27.2% and 88.6% (lowest and highest). This means 27.2%-88.6% of the failures were identified successfully.
  • Precision: From all instances labeled as failure, between 50.2% and 72.8% are actual failures (FPR).
TIME_TO_NEXT_FAILURE ANALYSIS
TIME_TO_NEXT_FAILURE ANALYSIS

• The authors analyzed the relation between the classifier label and the exact time until the next REMOVE event.
  • 24 hours away point was considered SAFE.
  • It would be considered SAFE if it fails in 2 days. Also, no difference between failing in 10 minutes vs. in 24 hours.
• SAFE classified as FAIL counts less as a misclassification as the time to next failure decreases.
• FAIL classified as SAFE has a higher negative impact when it is close to the point of failure.
  • These will be clear in the next slide.
TIME_TO_NEXT_FAILURE ANALYSIS

• Because of the fact the authors used 24 hours as the threshold value, when the classifier gives the wrong output, we still have time to catch it before the FAIL occurs.

• A good result: if the misclassified positives (actual failures labeled as SAFE) are further in time from the point of failure compared to correctly classified failures.
  • If we do not miss the closer failures but miss the further ones then it is good. We will have some time catch it again.

• Another good result: if misclassified negatives (actual safes labeled as FAIL) are closer to the failure point compared to correctly classified negatives.
  • If we misclassify closer SAFEs (they will eventually fail after 24 hours) compared to further away misclassified SAFEs, then our prediction makes more sense.
TIME_TO_NEXT_FAILURE ANALYSIS

• The outputs are divided into:
  • TP: failures correctly identified
  • FN: failures missed

• Upper and lower limits are 0 to 24 as expected.

• TP, on average, has lower times until the next event compared to FN.
  • This is good news: there is still some time left to actual failure so classifier may detect this in the future.
  • If we take this into account, the lowest prediction goes from 27.2% to 52.5% for benchmark 4 and from 88.6% to 88.8% for benchmark 15.
TIME_TO_NEXT_FAILURE ANALYSIS

• The outputs are divided into:
  • FP: SAFE labeled as FAIL
  • TN: SAFE labeled as SAFE
    • IMPORTANT: but they will eventually fail.
• The \textit{time_to_the_next_failure}, on average, are lower for FP than TN.
  • This is good news: the classifier gives false alarms when a failure is approaching, even if it is not strictly in the next 24 hours.
ADAPTATIONS FOR REAL LIFE USAGE
PERFORMING ONLINE

- All features need to be computed online.
  - Computation needs to take less than 5 minutes.
- Data aggregation is embarrassingly parallel.
  - All independent.
- The cost of storing the data and using BigQuery for analysis is estimated to be 60 dollars per day.
- Training each RF is also embarrassingly parallel.
- It is estimated 5 minutes to update their models.
CRITIQUE
CRITIQUE

- The cost analysis of missed FAILS plus misclassified FAILS vs. not using their model?
- Performance of difference features are no presented.
- How did we sampled two halves of test data (individual, ensemble).
  - Same with training data or separate.
- The threshold value analysis of ensemble output is omitted.