Failure Prediction of Jobs in Compute Clouds: A Google Cluster Case Study

Xin Chen, and Karthik Pattabiraman

University of British Columbia (UBC)

Charng-da Lu, Unaffiliated

Compute Clouds

- Infrastructure as a Service

Compute Clouds

Data & Storage Clouds

- Access to computational resources.
- Increasing cloud adoption in the scientific community.

Application Failures

- Application failures

```
Application application_1392853856445_0900 failed 2 times due to AM
Container for appattempt_1392853856445_0900_000002 exited with exitCode: 143 due to:
Current usage: 337.6 MB of 1 GB physical memory used; 2.2 GB of 2.1 GB virtual memory
```

- Problems
  - Higher failure rate in cloud clusters
  - Isolations of resources not guaranteed
  - Resources and power wasted in failures

Studies on Failures

- System Failures
- Application Failures

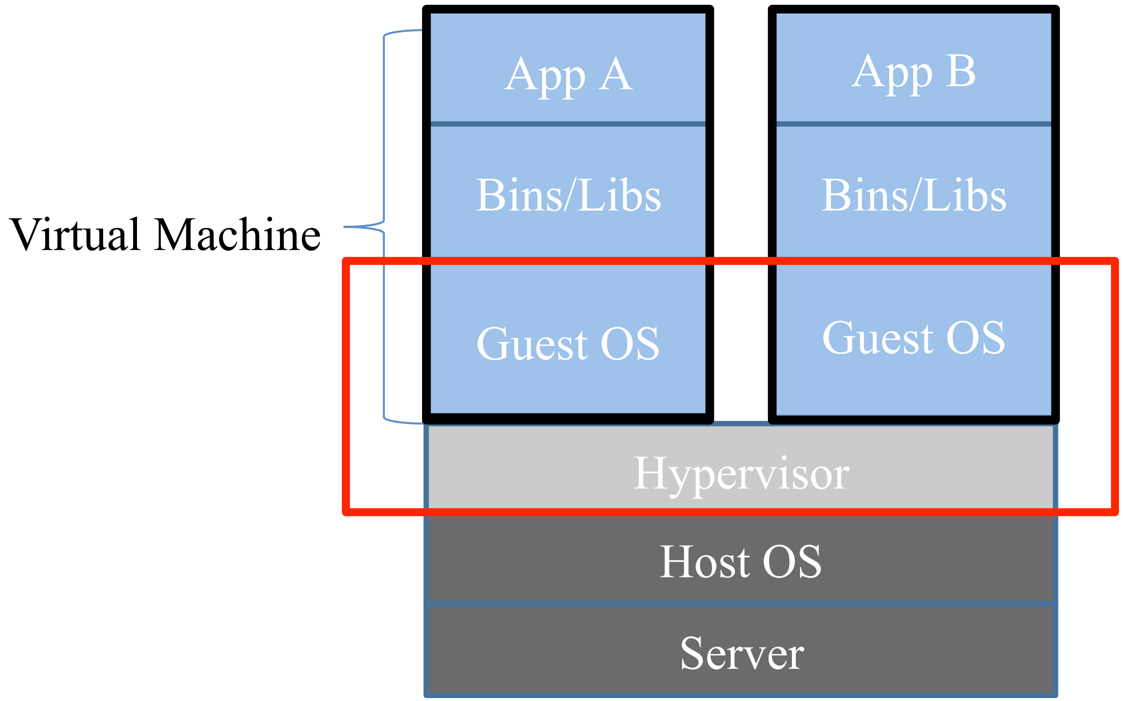
No specialized published application failure study on a generic production cloud with heterogeneous workloads.

Goal

- Enhance the reliability of running applications and managing failures in the future cloud.

- Research Question:
  - What are good predictors for application failures in a large scale compute cloud system?

Traditional Virtual Machines for Applications



- Running the entire operating system.
- Non-negligible provisioning time.
- Difficult to isolate application failures.

New Lightweight Models

- Operating system–level virtualization

Containers

LXC Linux Containers

docker

lmctfy

Support

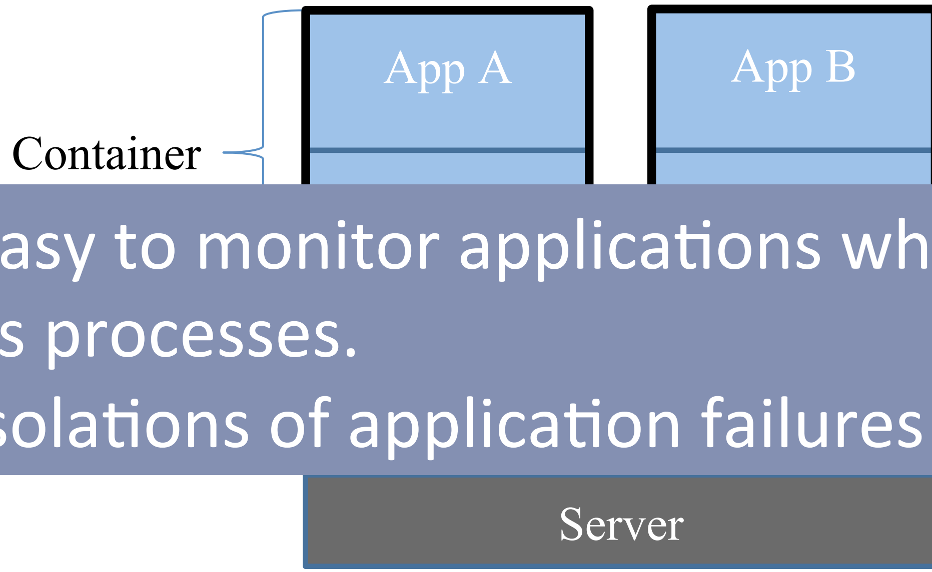
amazon web services™

Google Cloud Platform Live

Windows Azure

the rackspace cloud

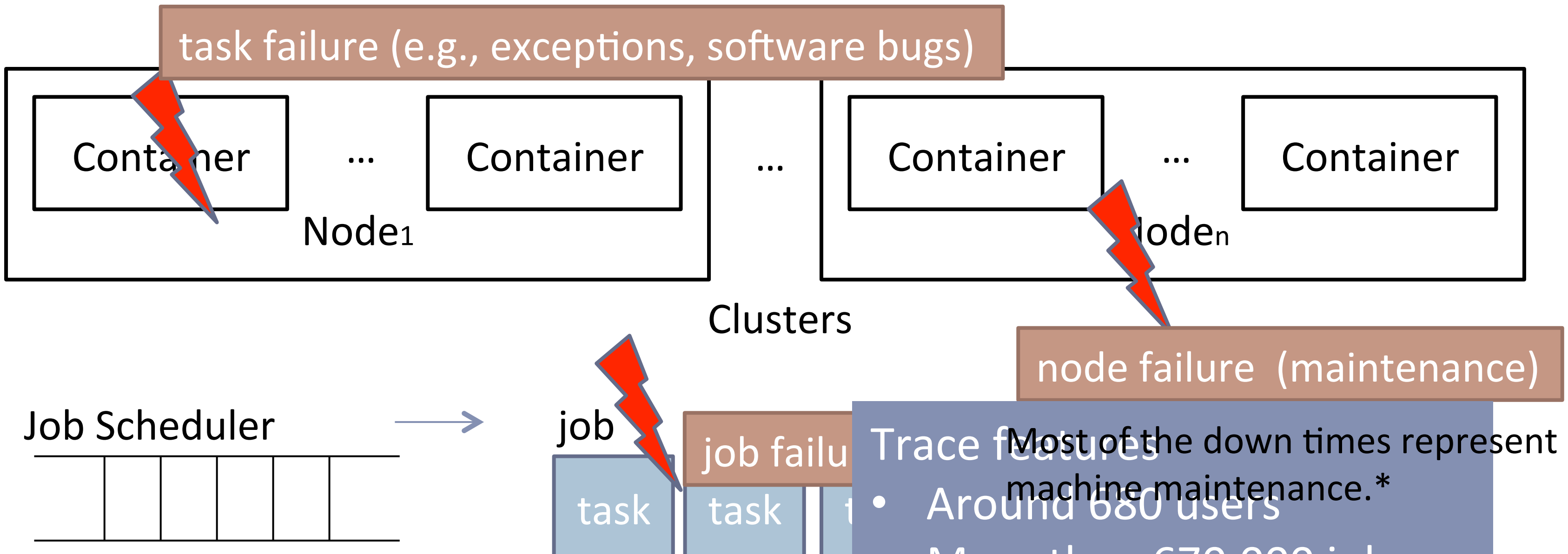
Containers



- Easy to monitor applications when regarded as processes.
- Isolations of application failures

- A few MB for extra libs.
- A few seconds for provisioning.
- Separate applications in containers.
- Extra reliability from OS failures.

Google Clusters: Failures



- Production jobs (e.g., web services)
- Batch jobs
  - MapReduce, Pregel, Percolator, etc.
  - BigTable

Trace features

- Around 680 users
- More than 670,000 jobs and 48 million tasks
- About 12,500 nodes
- 1 month period

Most of the down times represent machine maintenance.*

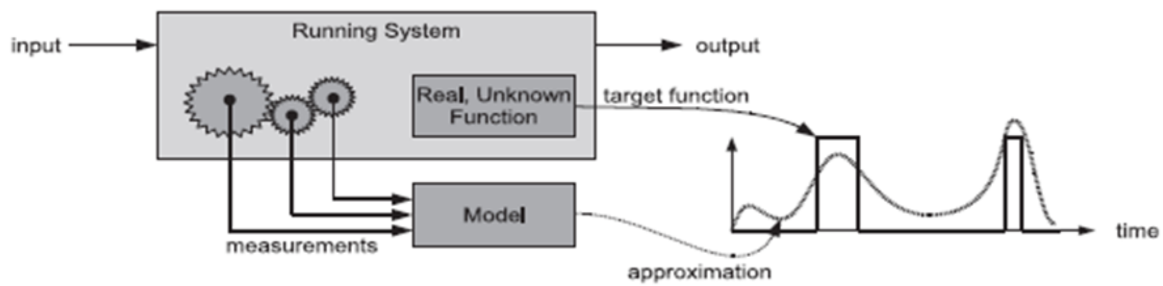
- Records we use
  - Job failures, task failures, and node failures
  - Other attributes and usage of jobs, tasks and nodes

*Reiss et al. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. SoCC 12’

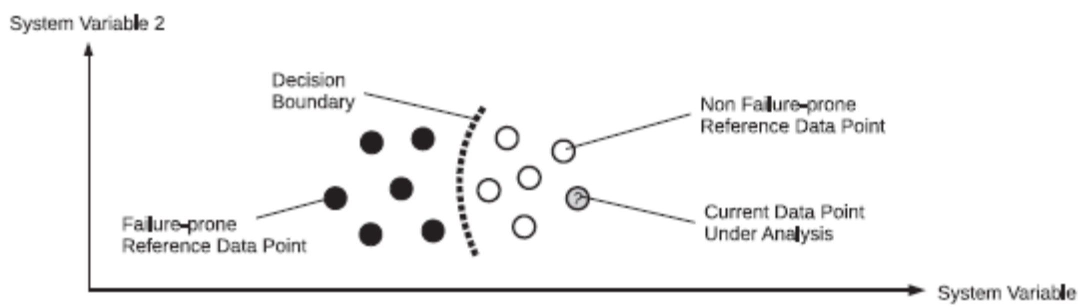
Challenges

- Features are difficult to be extracted.
- As the scale and heterogeneity increase in the cloud, simple methods have problems. *

Distributions



Classifications

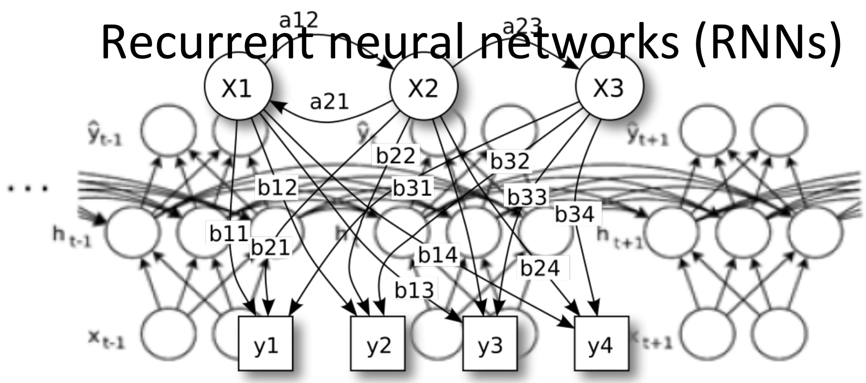


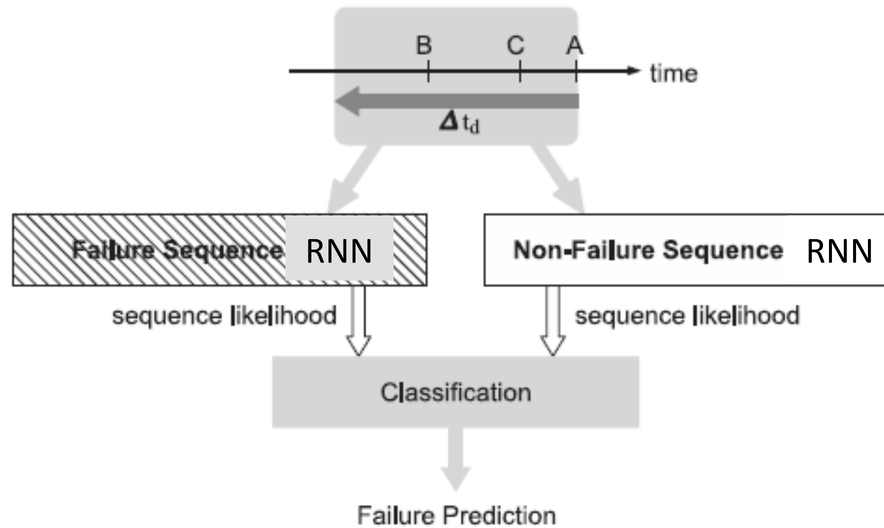
*Salfner et al. “A survey of online failure prediction methods,” ACM Comput. 2010.

Modeling Time Series

- Classifying failures and non-failures
  - Hidden (semi) Markov Models *

Recurrent neural networks (RNNs)





  - Dependencies on prior data
  - Transitions between the hidden states

*Salfner et al. “A survey of online failure prediction methods,” ACM Comput. 2010.

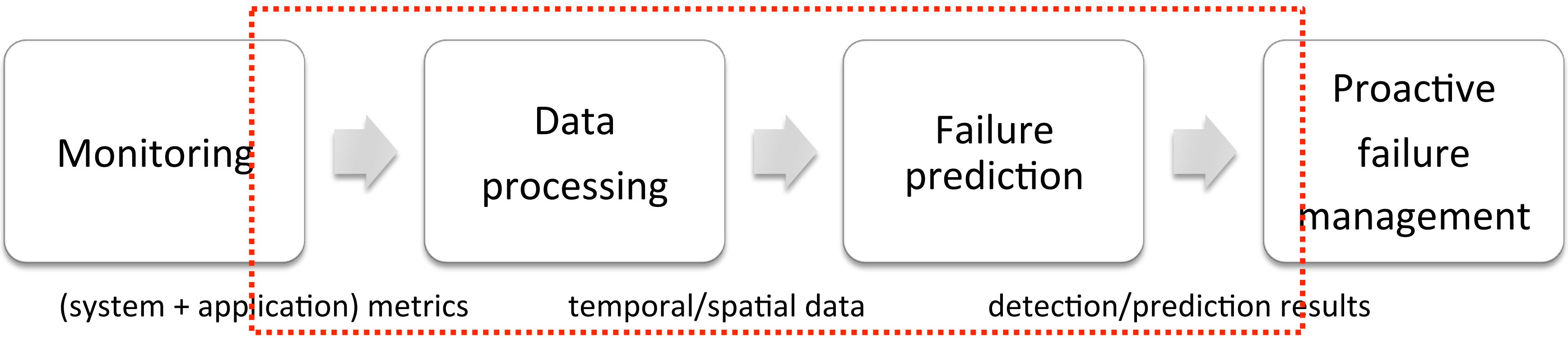
Objectives

- Build a specialized predictor for application failures.
  - Based on the characterization study (ISSRE 14')
  - Possibility to predict early

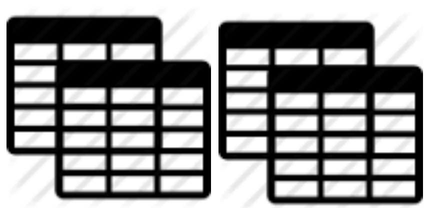
- No inferences of root causes (hidden by Google)

- Early prediction to reduce resource consumption and potentially improve scheduling.

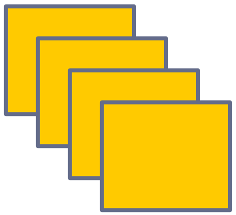
Prediction Framework for Performance Data



Node, job, and task tables

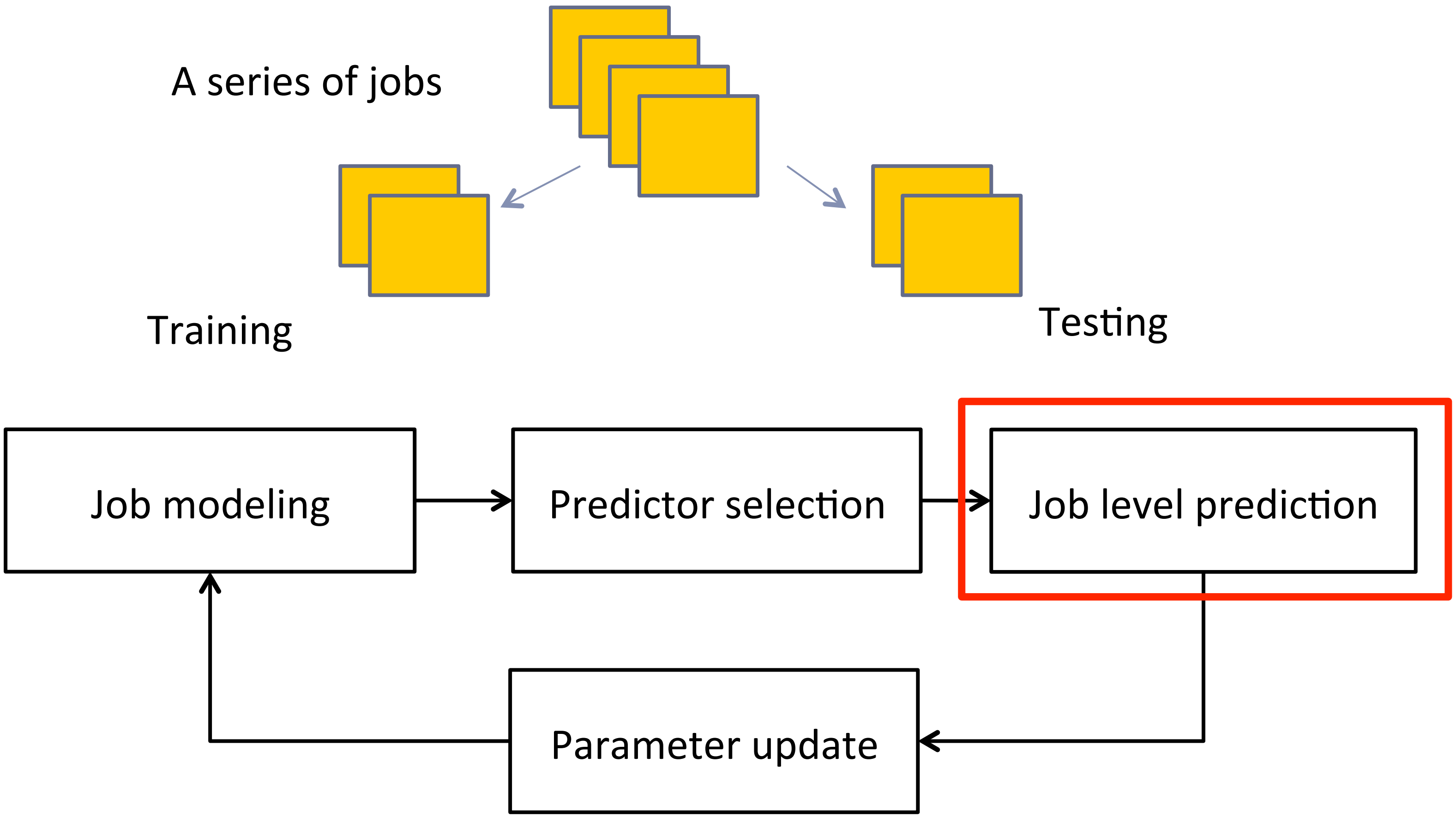


A series of jobs

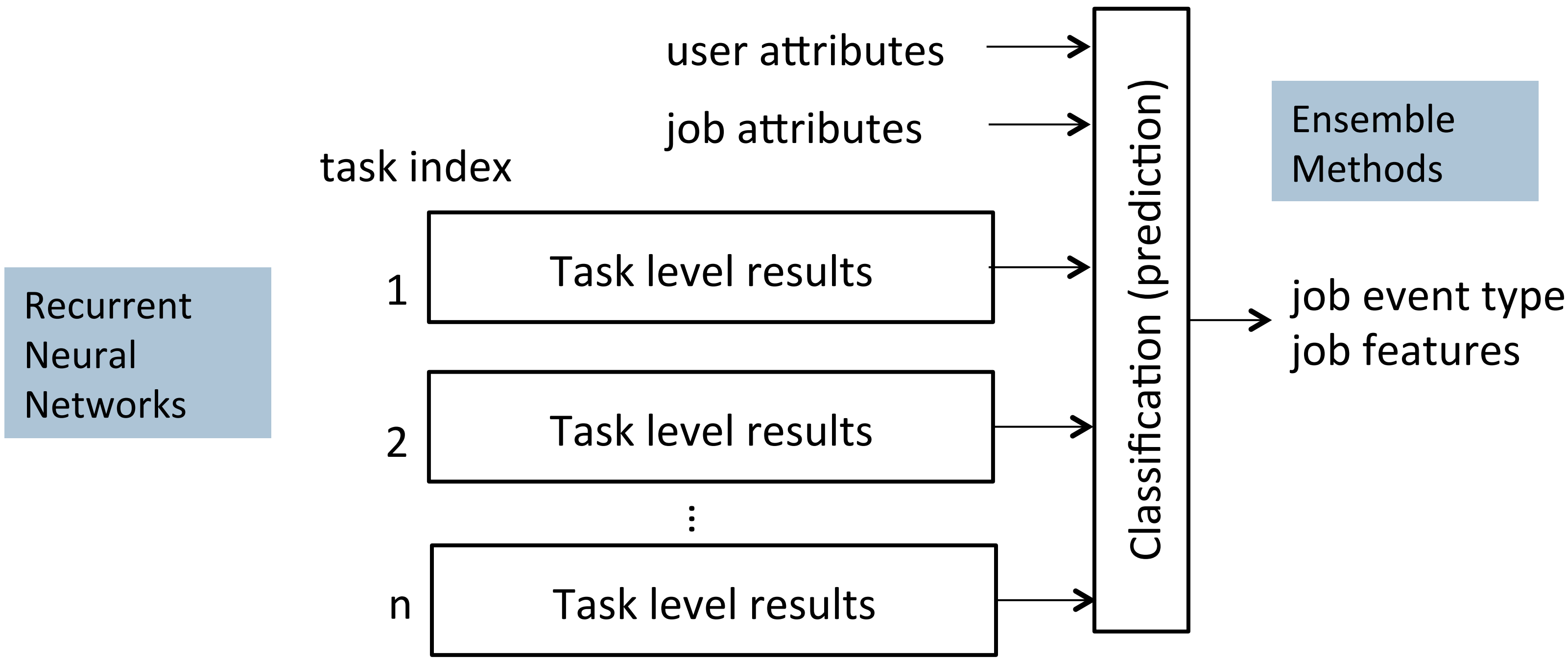


Focus of this paper

Overview of the Prediction Approach



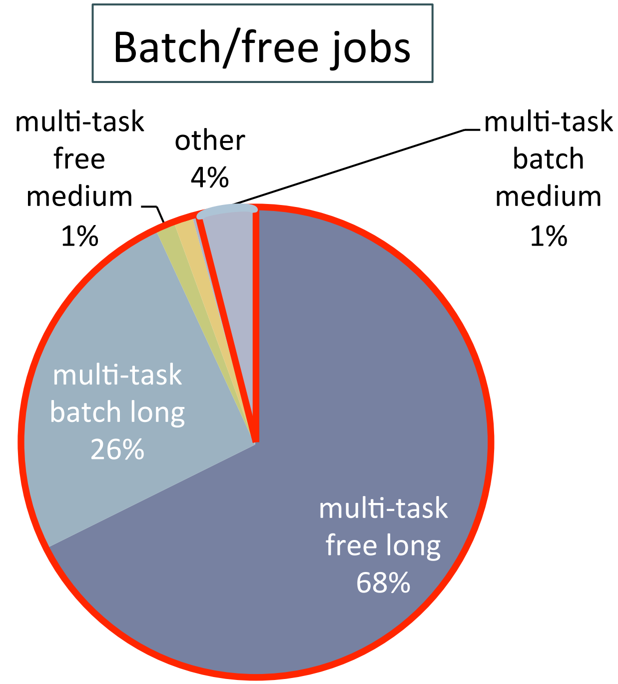
Job Level Prediction



- Usage data
  - mean CPU rate, maximum CPU rate
  - canonical memory usage, assigned memory usage, unmapped page cache, total page cache, maximum memory usage
  - disk I/O time, local disk space usage, maximum disk I/O time
  - cycles per instruction, memory accesses per instruction

Finer Categorization of Jobs

- Job length: short (< 10mins), medium (10 mins – 1h), long (> 1h)
- Job priority: Batch, free (low batch priority) and production
- Task number: single task VS multi-task



Targets the top 4 categories (96%)

Evaluation

- Data Selected
  - Failed jobs and finished job
  - The 4 categories for resource usage
  - Predict at the quarter, half and the end

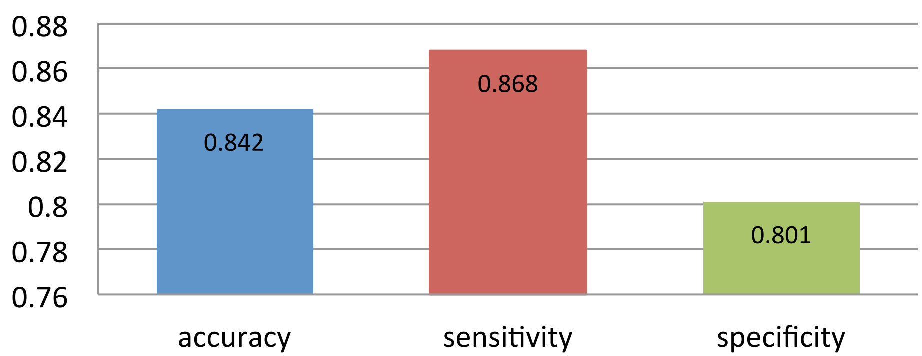
- Calculate the metrics and resource savings
- Runtime overhead
  - Training: 17.08 hours
  - Test: 9.52 minutes
  - Equivalent: 1 second to process around 37.8 minutes of the job data after high compression.

Task Level Results

- Evaluation metrics
  - Accuracy
  - Sensitivity (true positive rate or recall)
  - Specificity (1 – false positive rate)

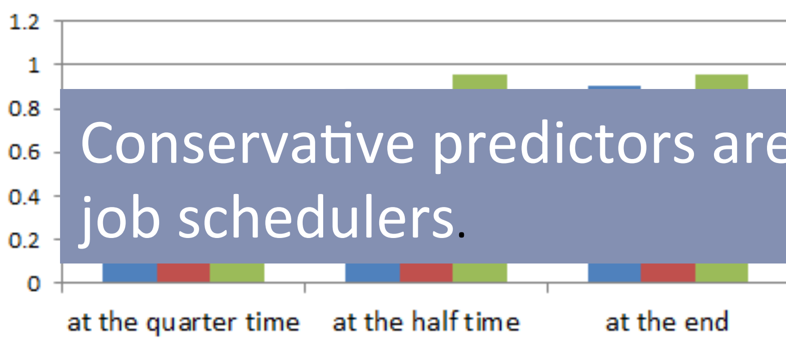
		Reality	
		Failure	Not failure
Prediction	Failure	True positive	False positive
	Not failure	False negative	True negative

- Predict if a task is successfully finished
  - Batch jobs

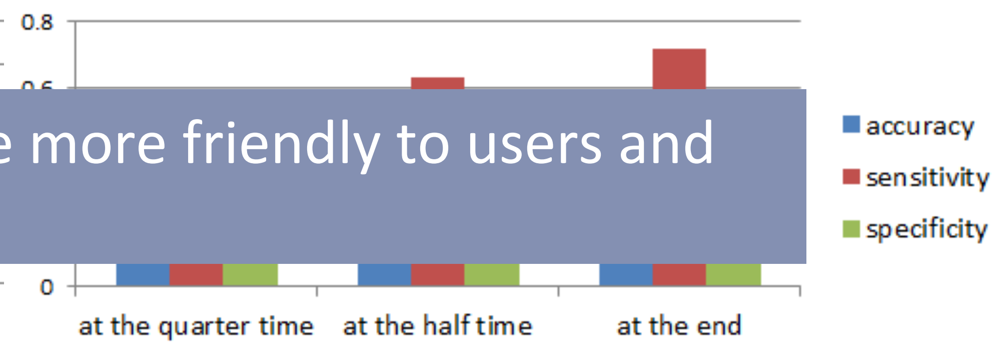


Job Level Results

- Predict a job failure
- Different classifiers (thresholds) generate
  - conservative predictor: Low TPR/FPR
  - aggressive predictor: High TPR/FPR



conservative predictor



aggressive predictor

Conservative predictors are more friendly to users and job schedulers.

conservative predictor

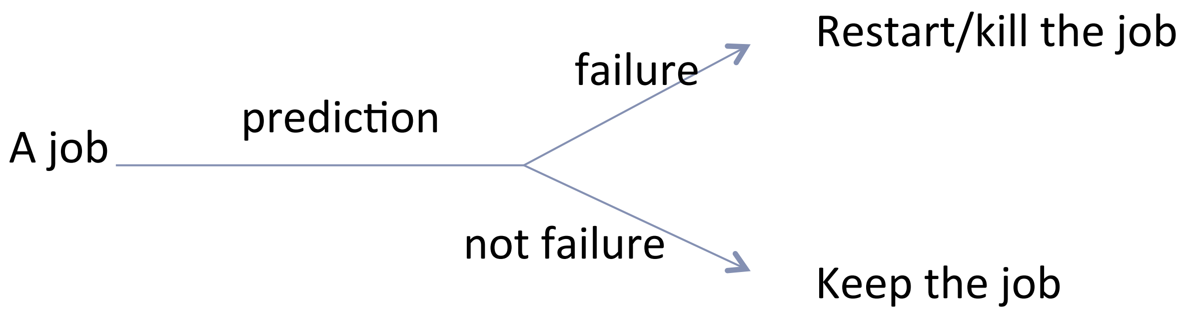
- FPR less than 10%, and TPR more than 40%.

aggressive predictor

- around 72% of TPR and 56% of FPR

An Example Method in Failure Management

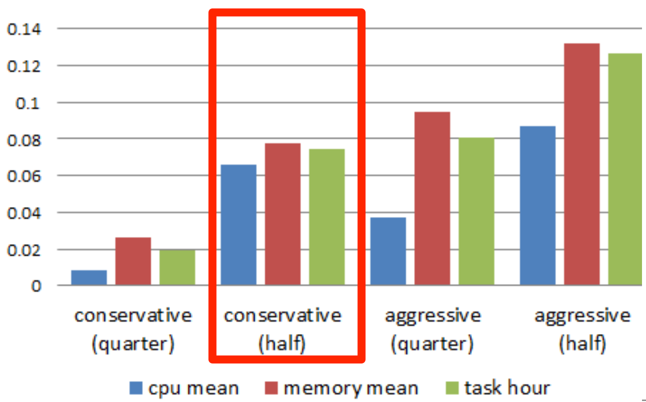
- Restart/kill the jobs that may finally fail (allowed by users)

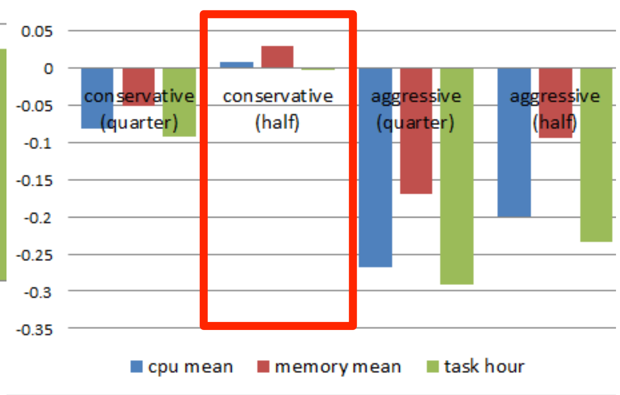


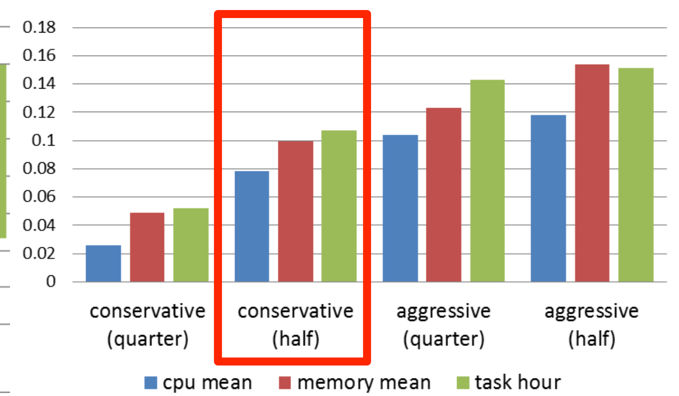
- Predict early to save resources
  - *overall improvement* = resource saved by stopping failed jobs - resource wasted by stopping finished jobs
  - $relative\ savings = overall\ improvement / resource(failed\ jobs) + resource(finished\ jobs)$

Resource Savings in Early Prediction

- Examples
  - Prediction times: quarter and half
  - Usage: CPU, memory, and task hour







multi-task batch long multi-task batch medium multi-task free long

Overall savings: 6% – 10% for the **conservative** predictors at the half time

Conclusion

- Failure prediction
  - Based on failure characterization and machine learning methods.
  - A true positive rate of more than 86% and a false positive rate around 20% at the task level.
  - 6% - 10% of resource savings for batch jobs.

- Future work
  - To improve the accuracy in the prediction algorithms.
  - To extend to a wider range of cloud systems.

Backup slides

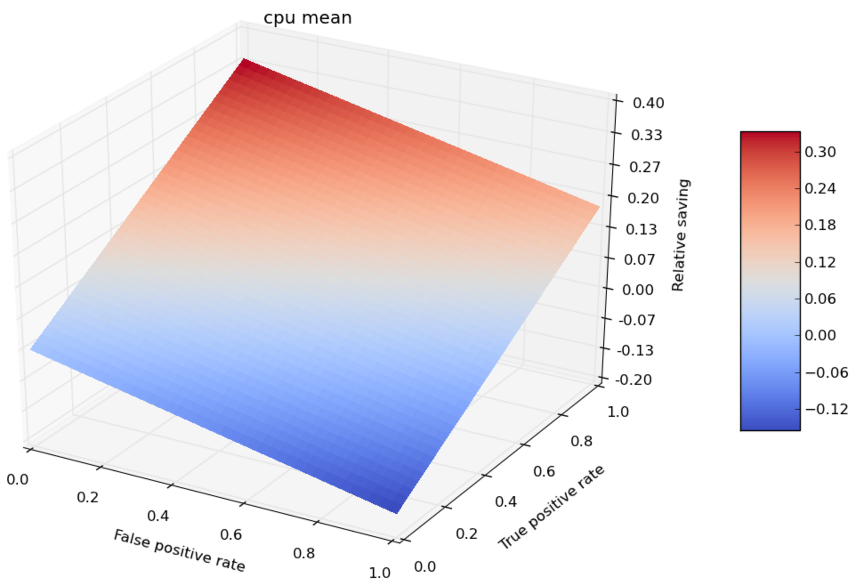
Threats to Validity

- Internal threats
  - We can not prove that the method is necessary the best.
    - Select the features that enlarge the differences between jobs.
    - Compare the results using multiple machine learning algorithms.
    - Use deep RNN to generate more and better features.
  - Failed and finished tasks may have similar properties and resource usage measures inside a job.
  - Need techniques such as predicting the job completion times.

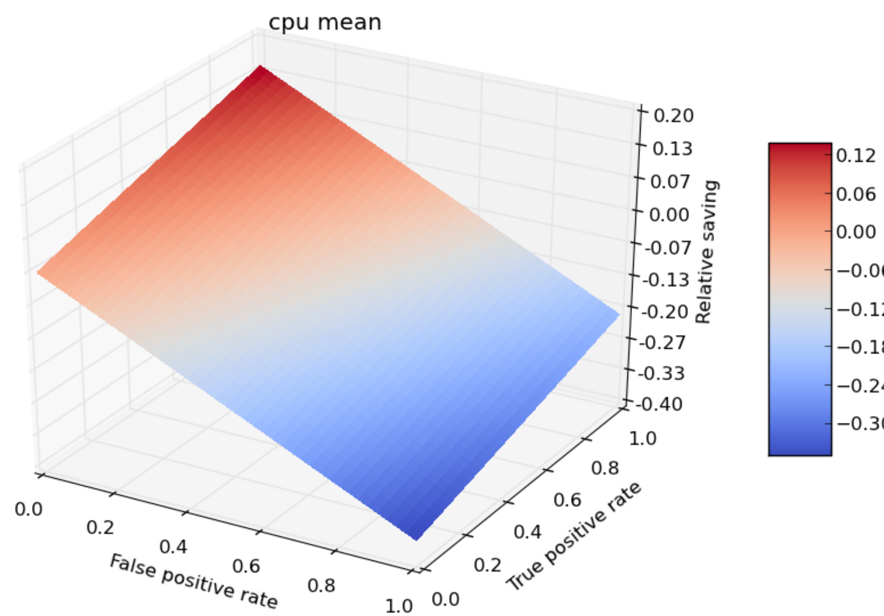
- External threats
  - The study limited to the Google clusters.

Approximations of Predictor Designs

Multi-task batch long jobs



Single-task batch long jobs



Tradeoffs

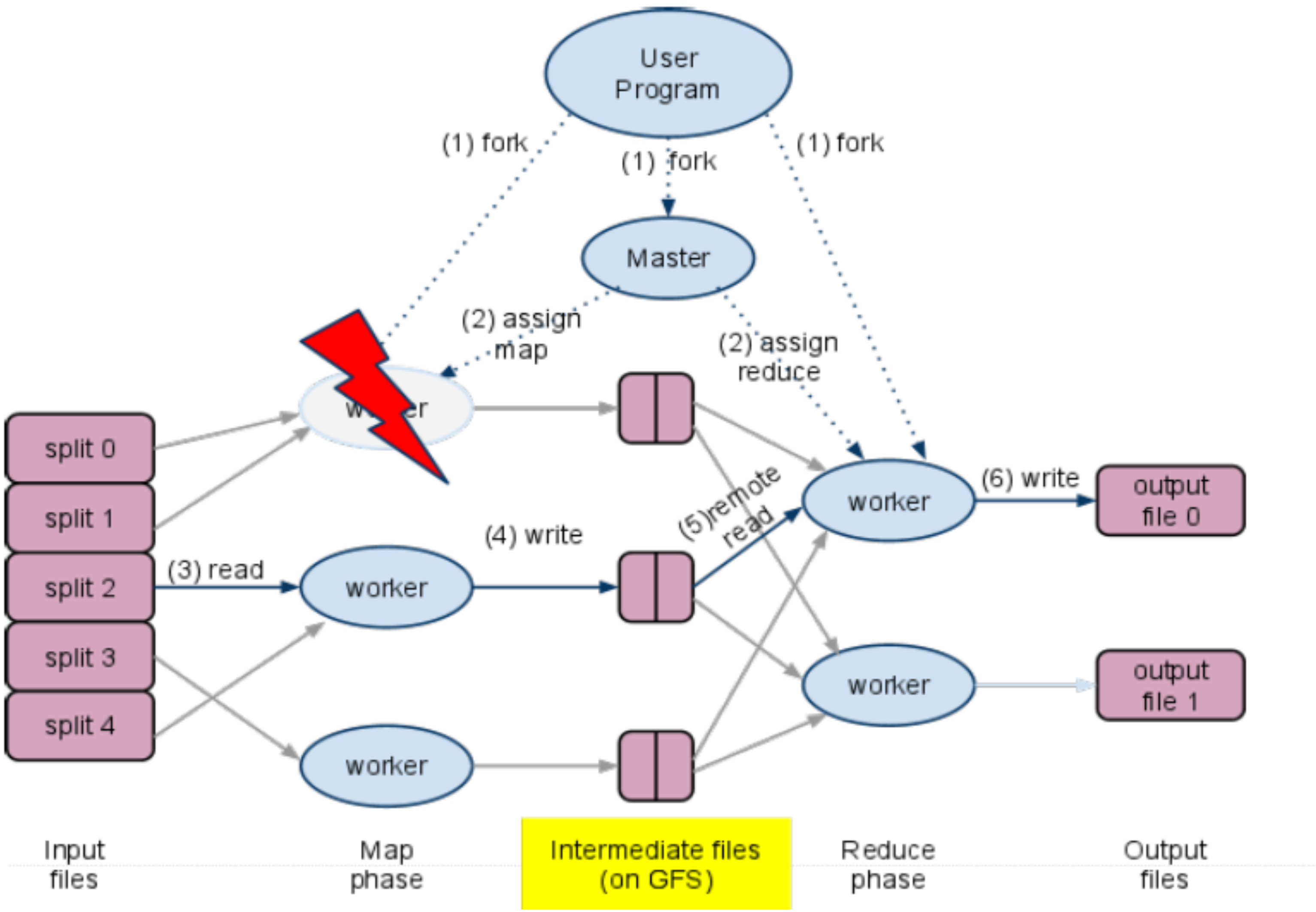
- Conservation predictor (low true positives/false positives)
- Aggressive predictor (high true positives/false positives)

User Based Optimization

- Training model:
  - Higher priority on prior jobs from the same users

- Around 7% to 10.7% for this predictor at the half times in batch jobs
- An additional 11% of increase in the true positive rate at the job level

Hadoop Case


User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
split 0
split 1
split 2
split 3
split 4
(3) read
worker
(4) write
worker
(5)remote read
worker
(6) write
output file 0
worker
output file 1
Input files
Map phase
Intermediate files (on GFS)
Reduce phase
Output files

Q & A

- Why TPR/FPR standards?
  - Our setting is more likely to be credit card fraud detection.
  - Different from failure repositories such as software bug reports (already biased to evaluate the classification).

- How to speedup the prediction?
  - select less examples in the training (not for increasing accuracy, but improving the time!)

Our Method VS Google