Data Driven Improvement of Data Center Performance

Cecily Chase — *Brown University*
Brian Nieves — *Duke University*
Harry Xie — *Duke University*

Motivation: Straggler Detection

In data centers, unbalanced or limited resources slow down certain tasks. These tasks are called *stragglers*. The focus of our research is to investigate the causes of stragglers in a Lenovo data center. We analyzed response times through the use of the so-called Hound methodology [1]. We discussed performance anomalies with Lenovo and created a monitoring framework that flags stragglers in real time. Additionally, we sought to generalize Hound into a format that can be applied to numerous data centers.

Lenovo Dataset

- The Lenovo data center has three machines.
- Each machine contains dozens of *containers*, also known as *jobs*, each of which runs part of a *service*. Examples of services are "carts" and "front-end."
- Variable values are recorded in thirty second intervals.
- Each time stamp entry contains its container ID, response time, and usage statistics, known as *features*.

TIME	CTN ID	RES	CTN CPU	CTN MEM	MAC MEM	MAC CPU	...
21:40:00	carts-lgb84	243.8	6.055	471.162	19153	79.122	...
21:40:30	carts-lgb84	253.9	5.477	471.542	20108	78.362	...
21:41:00	carts-lgb84	277.4	6.724	486.133	20112	81.977	...
21:41:30	carts-lgb84	215.9	11.432	483.187	20239	80.688	...
21:42:00	carts-lgb84	216.4	7.231	479.258	20322	82.744	...

Figure 1: Example of Lenovo data format

The goal of running Hound on the Lenovo dataset is to determine which of the recorded features are the primary causes of higher response times for each container and overall.

Hound Framework

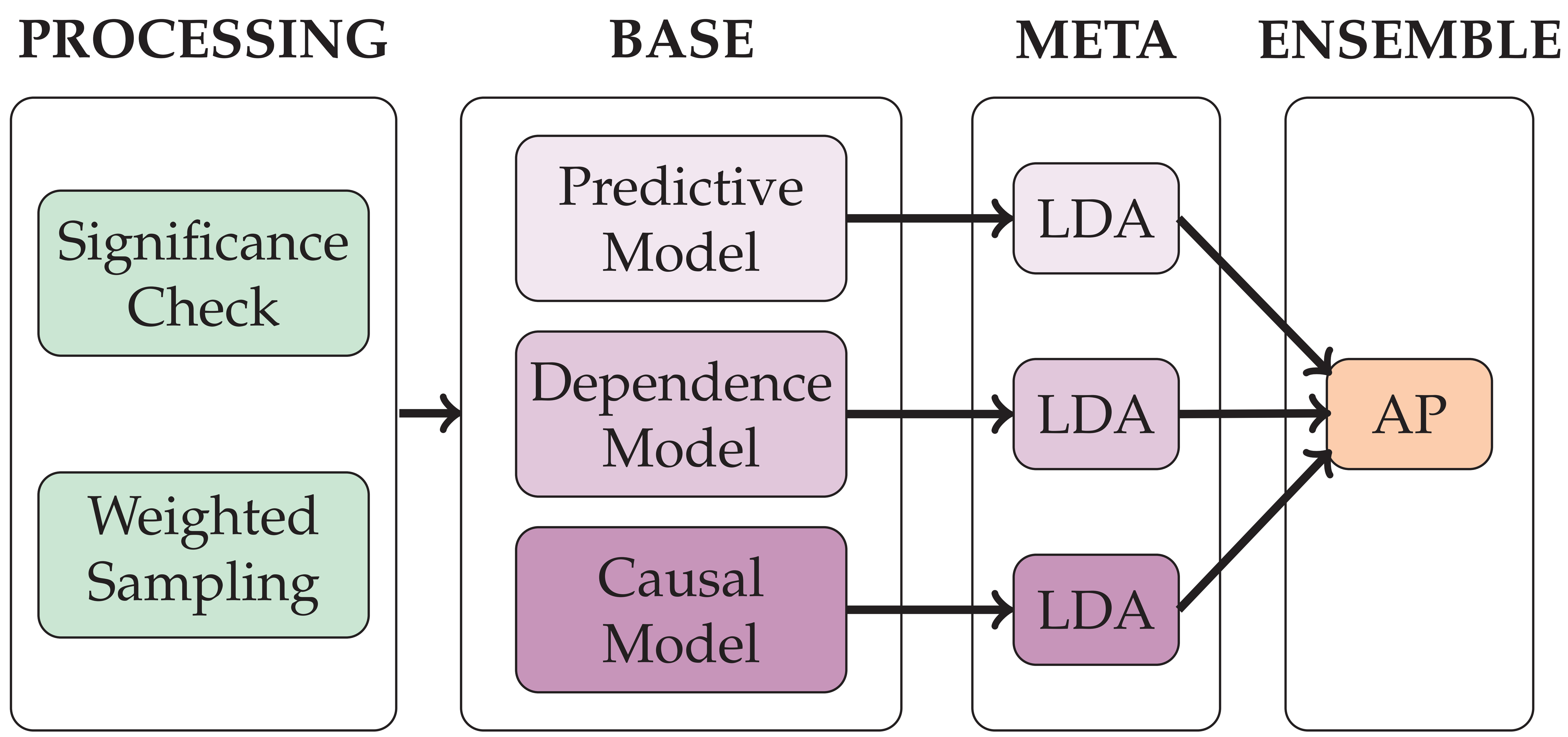
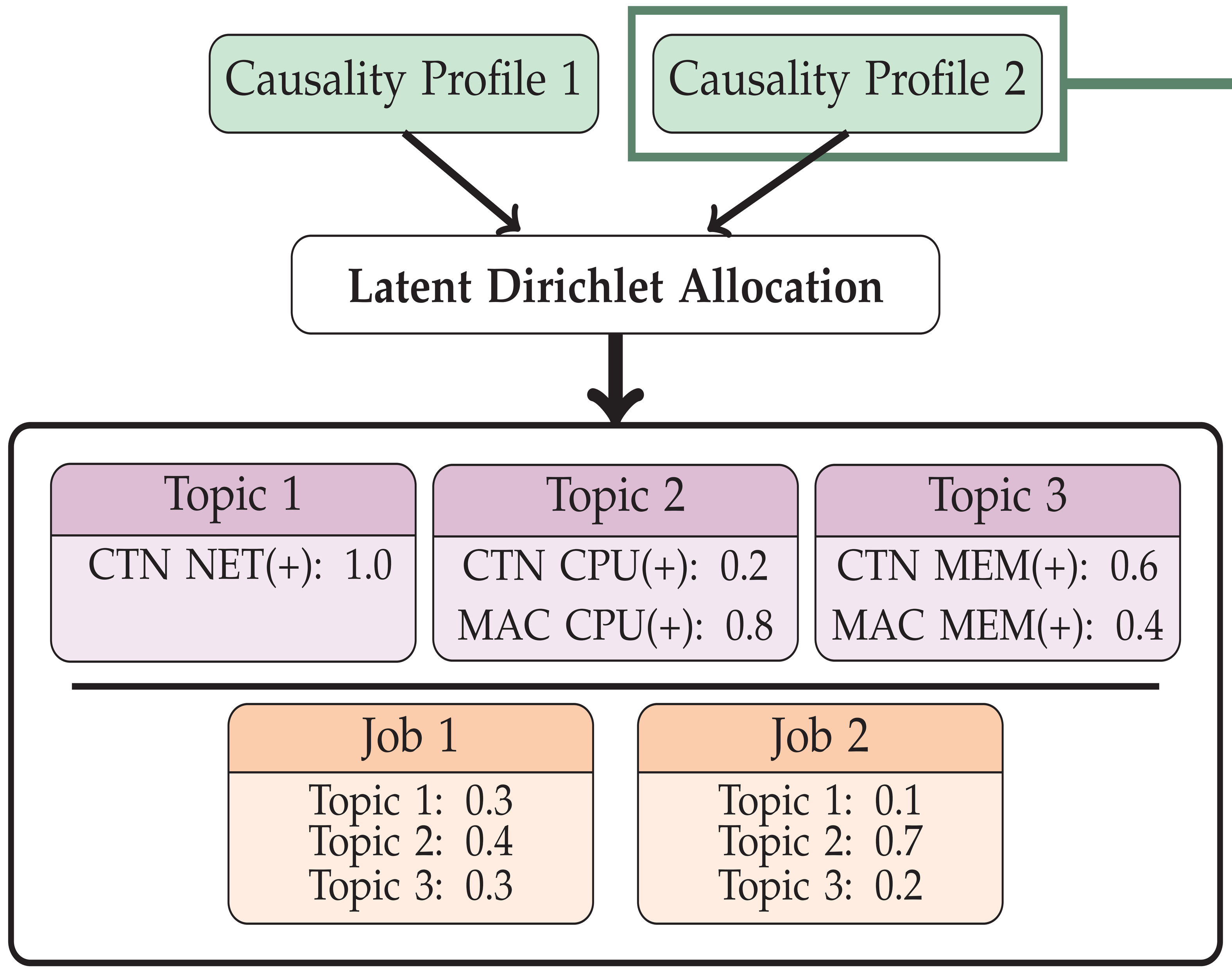


Figure 2: The four steps of the extended Hound framework

1. **Processing**: Pre-processing identifies features significantly related to response time via a paired t-test and samples the slower input tasks through parallelized weighted sampling [2].
2. **Base Learners**: Each base learner creates a causality profile that quantifies the relationship between features, including various machine statistics, and response time.
   - **Predictive:** Bagging Augmented Elastic Net
   - **Dependence:** Signed Schweizer-Wolff Dependence
   - **Causal:** Rubin Causal Inference with Adaboost
3. **Meta Learners**: Latent Dirichlet Allocation (LDA) finds patterns among causality profiles and creates causal topics. These causal topics consist of related features that commonly have a significant relationship with response time.
4. **Ensemble**: Affinity Propagation (AP) combines LDA topics to provide a detailed analysis of straggler causes [3]. These causes can then be mapped back to explain particular stragglers.

TOPIC MODELING

Each base learner creates a causality profile for every job. All of the profiles created by a single base learner makes up a corpus, from which topics are generated.



Job causality profiles are produced and analyzed for each base learner.

Causal Model: carts-lg85d

Features	Coefficient
CTN CPU	0.0
CTN MEM	0.046
MAC CPU	-0.151
MAC MEM	0.191
MAC NET R	0.0
MAC NET T	-0.026
CTN NET R	0.0
CTN NET T	0.0

Figure 3: Base Learner outputs (right) are fed to LDA, creating descriptive topics (left).

These causality profiles result in three coefficients relating response time to a given feature, each via a different base learner. Notice that coefficients tend to have the same sign across base learners.

Feature	Predictive	Dependence	Causal
CTN CPU USAGE	0.00	0.00	0.00
CTN MEM USAGE	-0.082	-0.351	-0.219
CTN NET READ	0.00	0.00	0.00
CTN NET WRITE	0.00	0.00	0.00
MAC CPU USAGE	-0.151	-0.444	-0.097
MAC MEM USAGE	0.191	0.583	0.139
...	...	...	...

Figure 4: Example of three causality profile outputs for a single job.

Once causality profiles and subsequent LDA topics have been produced for each job, these topic results can be aggregated into ensemble topics.

ENSEMBLE LEARNING

A single base learner creates 4-6 LDA topics per service with enough jobs. In total, there are 7 services, for a combined 106 topics.

Ensemble Topics By Affinity Propagation

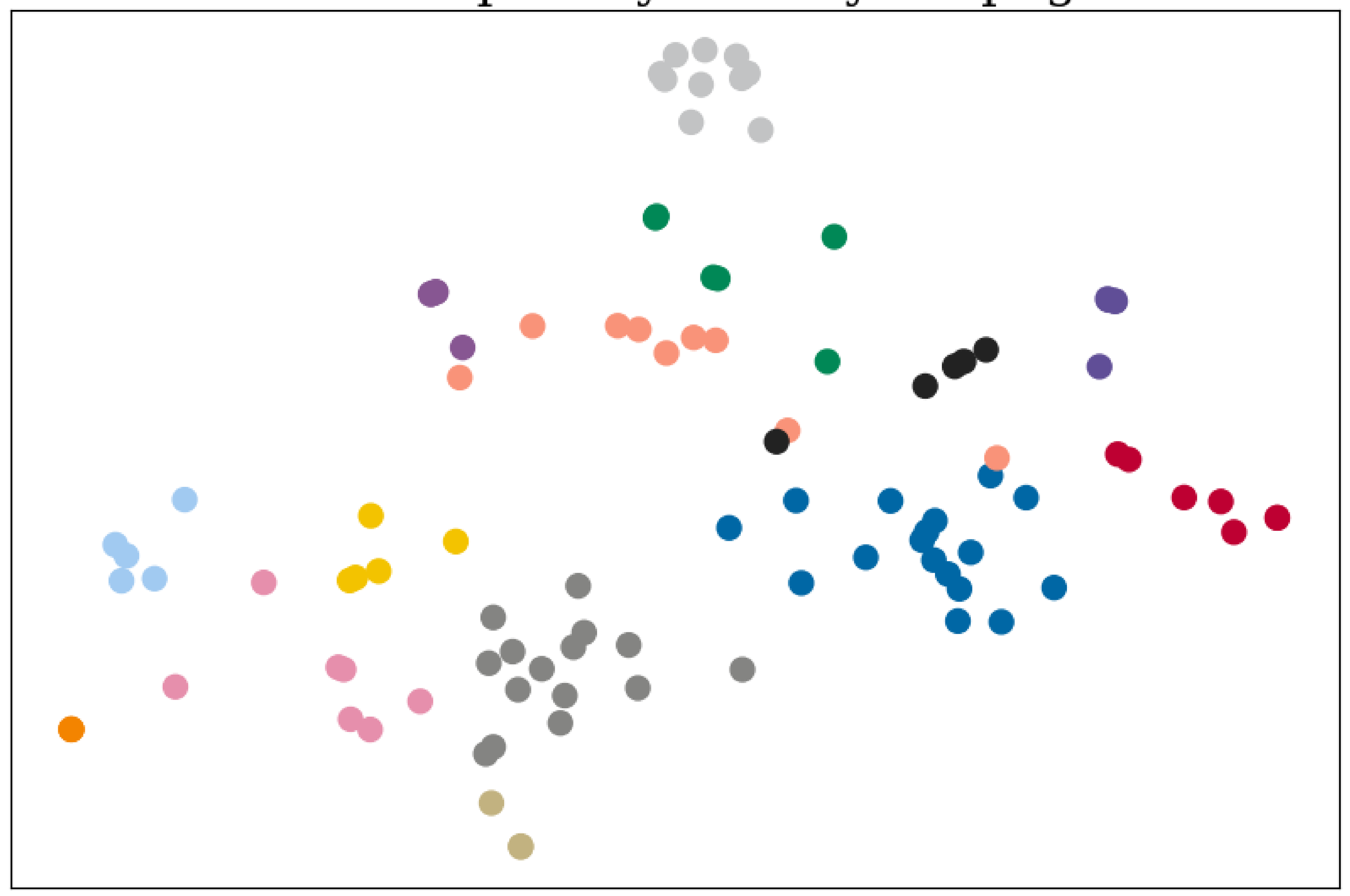


Figure 5: Clusters of all LDA Topics from all base learners.

Because many of these are similar, we create clusters via Affinity Propagation and calculate average weights for each cluster. These are the three ensemble topics with the highest coverage, where coverage is the proportion of jobs at least partially explained by the topic.

Topic	Keywords	Weights	Coverage	Interpretation
E_0	MAC. NET T. (+)	0.42	25.2%	Heavy Network Traffic
	MAC. NET R. (+)	0.41		
	MAC. CPU (+)	0.17		
E_1	CTN. MEM (+)	0.54	20.9%	Memory Bandwidth Contention
	MAC. CPU (+)	0.46		
E_2	MAC. CPU (+)	0.5	12.3%	Computation Skew
	CTN. CPU (+)	0.36		
	CTN. MEM (-)	0.14		

Figure 6: Ensemble topics and job coverage.

Once the ensemble topics for the data set have been assembled, one can map them back to particular jobs. In this way, we are able to explain the causes of particular stragglers in the data set. This is also helpful in demonstrating the differing causes of stragglers across different services in the data set.

An Example Job

For a particular job in the dataset, Hound outputs can be applied to explain stragglers. The following is an example result for the job "front-end-79f895cb65-dmwsg," in which the ensemble topics are mapped back to explain stragglers in the particular job. As shown, container memory usage (+) and machine CPU usage (+) are deemed most responsible for the response time, with the (+)'s meaning there is a positive relationship between each feature and the response time. These results are corroborated by the subsequent time series, which shows how all three of these variables behave over the course of a day's worth of data.

Topics	Keywords	Weights	Interpretation
E_1	Container Memory Usage (+)	0.326	Memory Contention
	Machine CPU Usage (+)	0.237	Computational Overload
E_6	Container Network Transmitted (+)	0.177	Insufficient Network
	Container Network Read (+)	0.177	Bandwidth

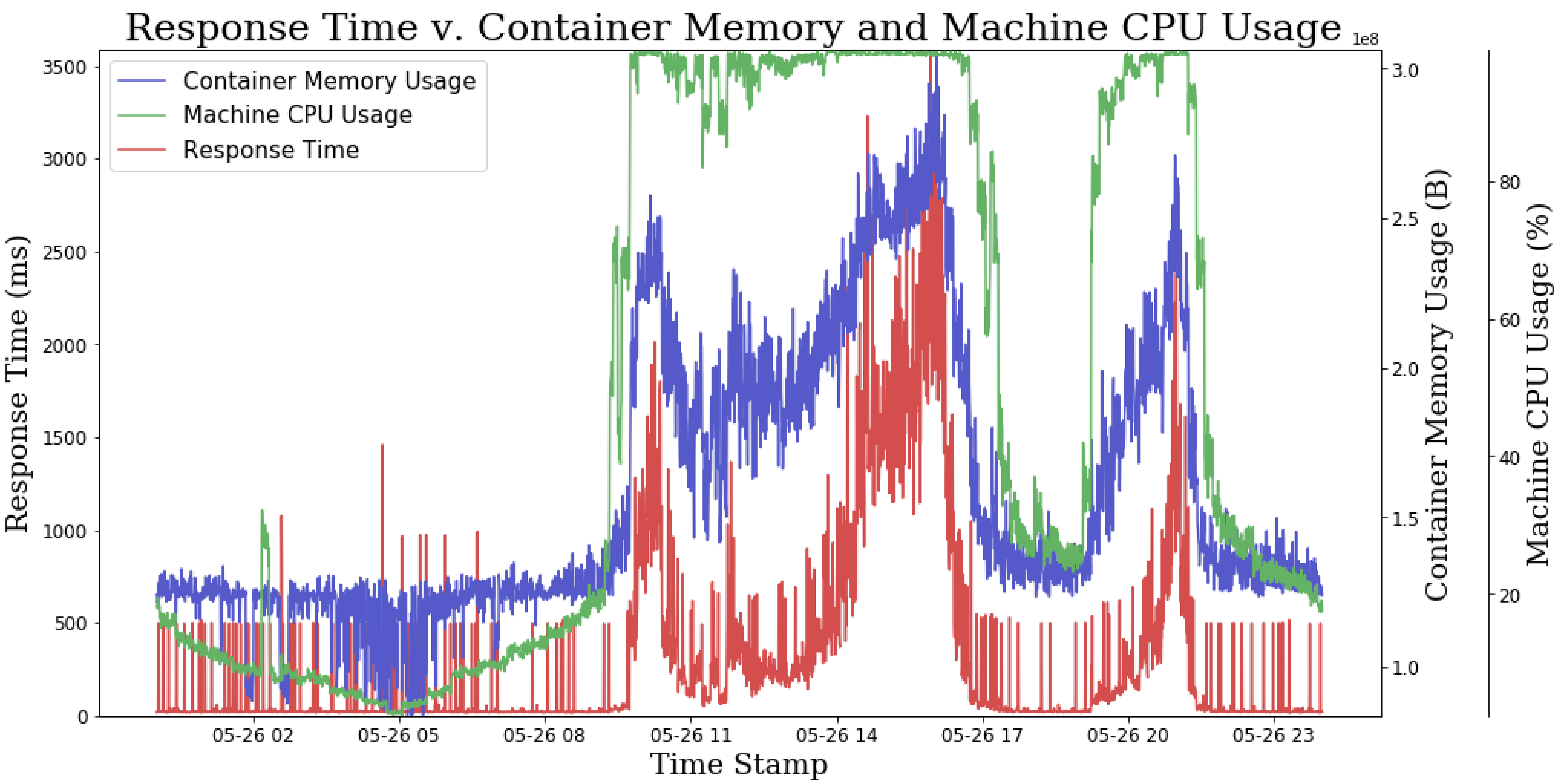


Figure 7: Hound Ensemble results mapped back to this container (top), and a time series of identified features compared to container response time (bottom).

Future Work

1. **Rescheduling**: We hope to create a rescheduling schema capable of analyzing job performance and determining appropriate countermeasures. This includes an algorithm that can flag under-performing jobs and then assess how to reschedule the job/process to optimize data center performance.
2. **Near real time detection**: We want to optimize Hound to diagnose jobs as data comes in, so that datacenters can detect and reschedule in near real time.

References

[1] **Hound: Causal Learning for Datacenter-scale Straggler Diagnosis** Pengfei Zheng and Benjamin C. Lee. 2018. *POMACS.* 2, 1, Article 17 (March 2018), 36 pages. **https://doi.org/10.1145/3179420**

[2] **Algorithms Every Data Scientist Should Know: Reservoir Sampling** Josh Wills. *Cloudera Engineering Blog* (April 2013). Retrieved from *https://blog.cloudera.com/blog/2013/04/hadoop-stratified-randosampling-algorithm* (July 2018)

[3] **Affinity Propagation** Retrieved from *http://scikit-learn.org/stable/modules/clustering.html #affinity-propagation* (July 2018).

Acknowledgments

We would like to thank Jianming Zhang at Lenovo for providing us with the dataset. We would also like to thank Professor Benjamin Lee for his oversight, Dmitry Vagner for his management help, and Calvin Ma and Pengfei Zheng for their contributions to this project.

We would also like to thank Ariel Dawn, Paul Bendich, Kathy Peterson and the Data+ Program for their support on the project.