Fingerprinting the datacenter: automated classification of performance crises

Peter Bodík[1,3], Moises Goldszmidt[3], Armando Fox[1], Dawn Woodard[4], Hans Andersen[2]

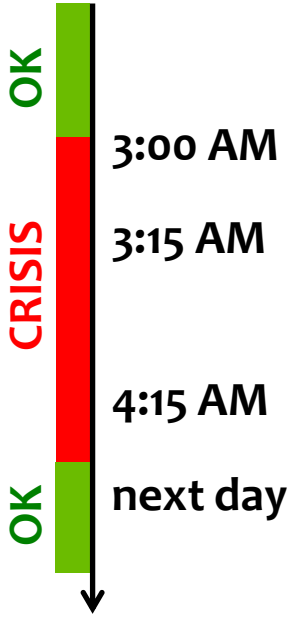
[1]RAD Lab, UC Berkeley

[2]Microsoft [3]Research

[4]Cornell University

Crisis identification is difficult, time consuming and costly

Frequent SW/HW failures cause downtime



Timeline of a typical crisis

- detection: automatic, easy
- identification: manual, difficult
  - takes minutes to hours
- resolution: depends on crisis type
- root cause diagnosis, documentation

Web apps are complex and large-scale

- app used for evaluation: 400 servers, 100 metrics

Insight: performance metrics help identify recurring crises

Performance crises recur

- incorrect root cause diagnosis
- takes time to deploy the fix
  - other priorities, test new code

System state is similar during similar crises

- but not easily captured by fixed set of metrics
- 3 operator-selected metrics not enough

Contribution: crisis identification as it happens, via classification

1. Fingerprint = compact representation of system state
   - uniquely identifies a crisis
   - robust to noise
   - intuitive visualization

2. Using fingerprints to identify crises as they happen
   - goal: operator receives email about crisis
   - "Crisis similar to DB config error from 2 weeks ago"

3. Evaluation on data from a real commercial service deployed on hundreds of servers
   - 80% identification accuracy

Outline

- Definition of performance crises
- Crisis fingerprints
- Evaluation results
- Related work
- Conclusion

Definition and examples of performance crises

Performance crisis = violation of service-level objective (SLO)

- based on business objectives
- captures performance of whole cluster
- example: >90% servers have latency < 100 ms during 15-minute epoch

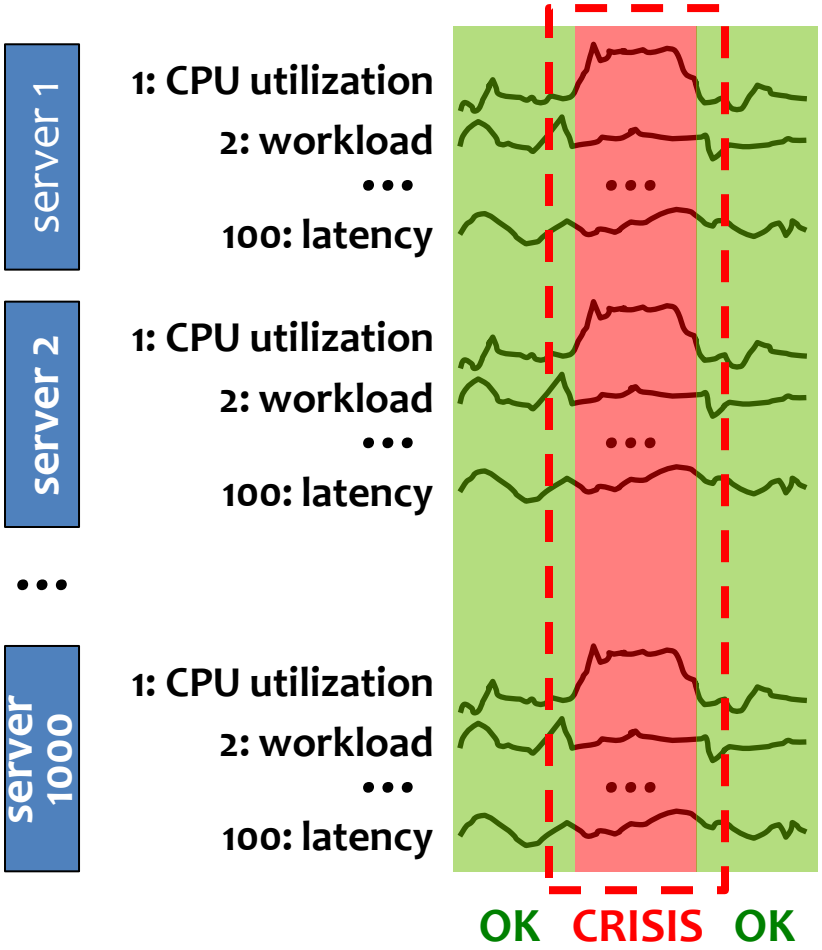
Crises we analyzed

- app config, DB config, request routing errors
- overloaded front-end, overloaded back-end

Fingerprints capture state of performance metrics during crisis

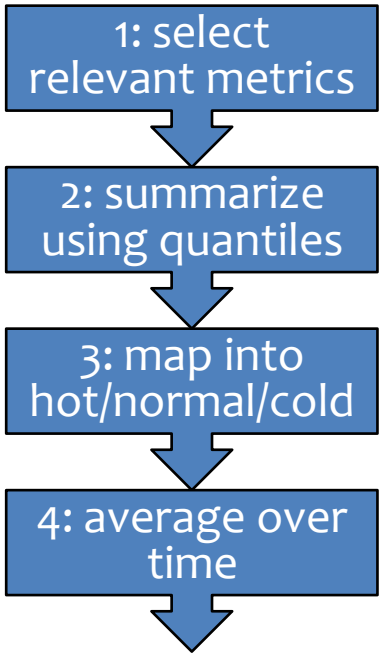
Metrics as arbitrary time series

– OS, resource utilization, workload, latency, app, …



across all the application servers (25th , 50th , and 95th percentiles)

a summary vector containing one element per quantile per tracked metric, indicating whether the value of that quantile is cold, normal, or hot during that epoch.



crisis fingerprint

Step 1: Using feature selection to pick relevant metrics

what would not work

- all 100 metrics
- 3 operator-selected metrics

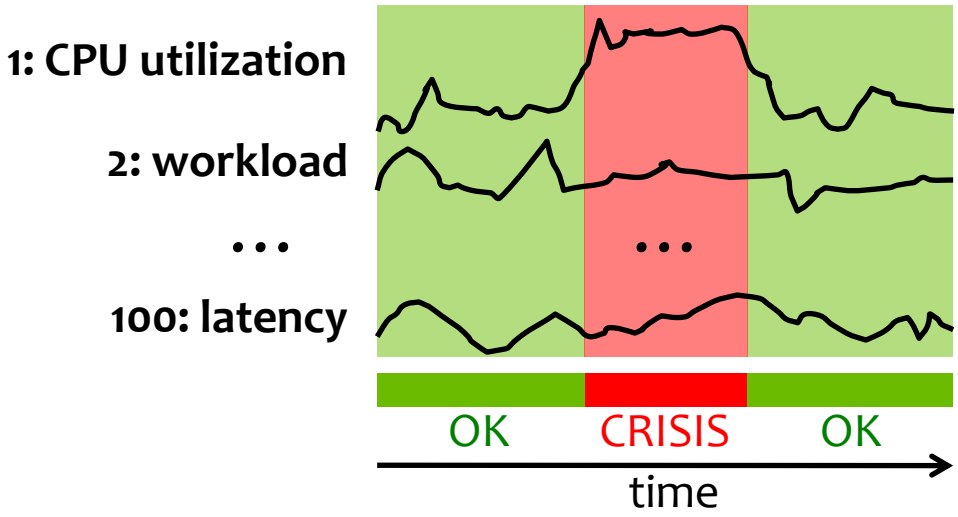
→ low identification accuracy

The idea behind regularized logistic regression is to augment the model fitting to minimize both the prediction error and the sum of the model coefficients.This in turn forces irrelevant parameters to go to zero, effectively performing feature selection.

Logistic regression with L1 constraints

- fit accurate linear more with only few metrics
- selected metrics that operators didn't consider

It has been (empirically) shown in various settings that this method is effective even in cases where the number of samples is comparable to the number of parameters in the original model

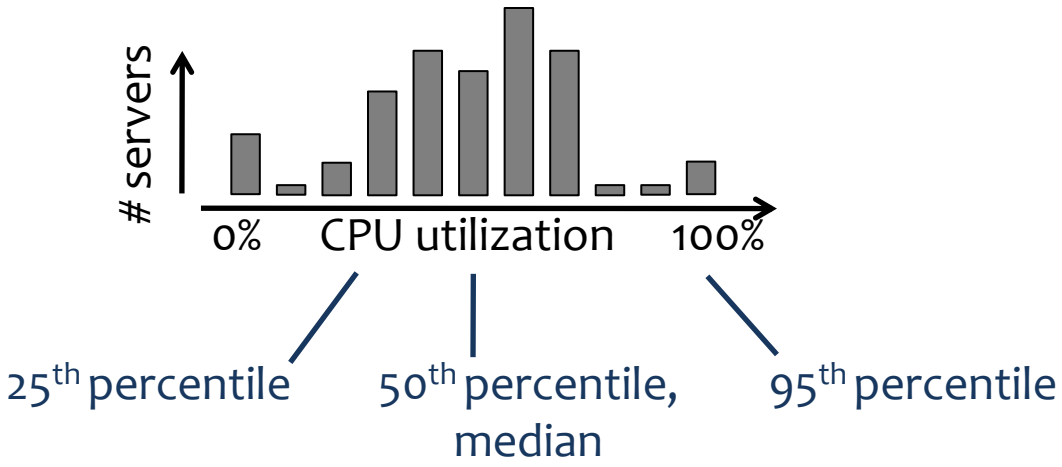


model input (all metrics)

↓

model output (binary)

Step 2: Summarize selected metrics across servers using 3 quantiles



- robust to outliers
- can efficiently compute even for datacenter-sized clusters

what would not work

- mean, variance
- only median

Step 3: Map metric quantiles into hot/normal/cold

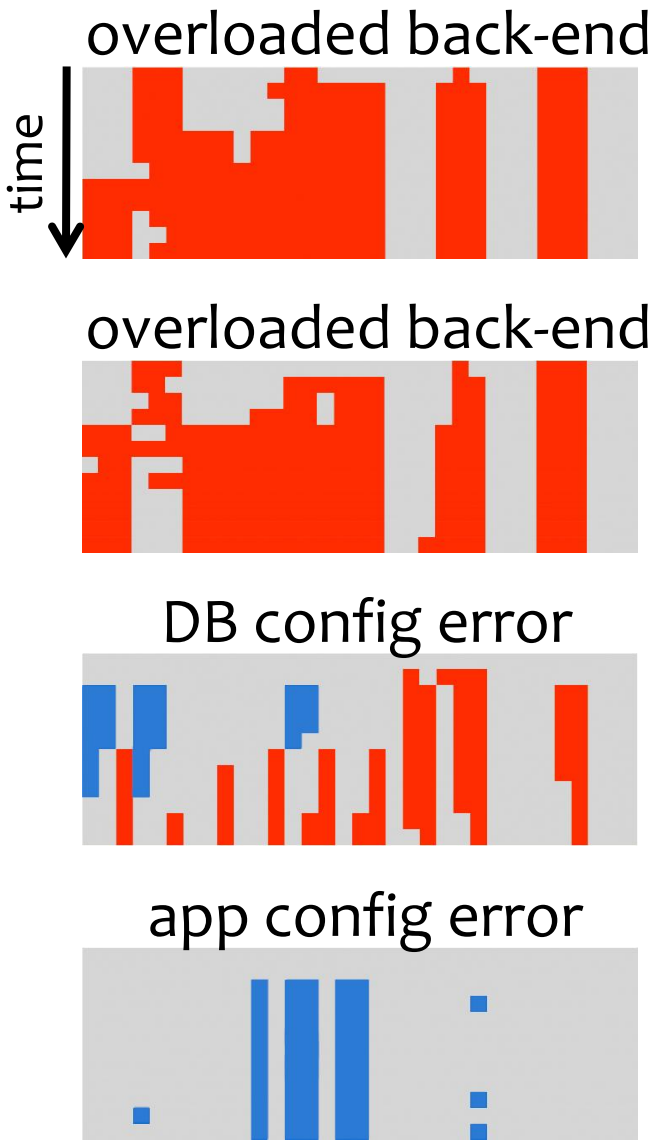
Based on historic values

Epoch fingerprints

- differentiate among crises
- compact
- intuitive

what would not work

- raw metric values
- time series model



Step 4: Averaging over time

Different crises have different durations

what would not work

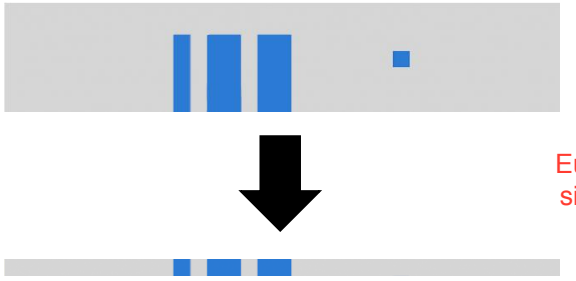
- all epoch fingerprints
- 1 epoch fingerprint

Crisis fingerprint

Each row represents an epoch, each column represents a metric quantile, and white, gray, and black represent the values –1, 0, and 1, respectively of the cold, normal, and hot state respectively. Thethree left-most columns of the third crisis from the top in the figure would be summarized as { –7 , –4 , 6 }; there are 12 12 12 12 epochs in the crisis and the column sums are –7, –4, and 6.

- average epoch fingerprints over time
- compare by computing Euclidean distance

epoch fingerprints



Euclidean distance is used as the similarity measure of two vectors

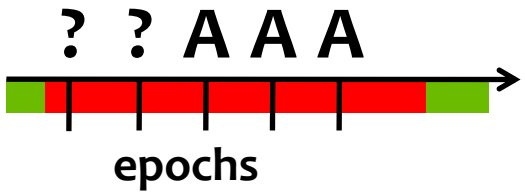
crisis fingerprint is a vector

Crisis identification in operational setting

Crisis detected automatically via SLO violation

During first hour of crisis

- update fingerprint of current crisis
- if found similar crisis P, emit label P
  else emit ? – "previously-unseen crisis"



When crisis is over

- automatically update relevant metrics, fingerprints
- ideally, operators enter supplied label into crisis DB

Outline

- Definition of performance crises
- Crisis fingerprints
- **Evaluation results**
- **Related work**
- **Conclusion**

System under study

24 x 7 enterprise-class user-facing application at Microsoft

- 400 machines
- 100 metrics per machine, 15-minute epochs
- operators: "Correct label useful during first hour"

Definition of a crisis

- operators supplied 3 latency metrics and thresholds
- 10% servers have latency > threshold during 1 epoch

19 operator-labeled crises of 10 types

- 9 of type A, 2 of type B, 1 each of 8 more types
- 4-month period

Evaluation results

Identification stability = stick to first label

- unstable: ??A??, AABBB
- stable: ?????, AAAAA, ??AAA

Previously-seen crises:

- identification accuracy: 77%
- identified when detected or one epoch later

For 77% of crises, average time to ID 10 minutes

- could potentially save up to 50 minutes
- more with shorter epochs

Accuracy for previously-unseen crises: 82%

More results in the paper

Comparison to other approaches

- using all metrics
- 3 operator-specified metrics
- failure signatures [SOSP '05]

Updating fingerprints

Sensitivity analysis

Online-clustering approach

- model evolution of fingerprint during crisis
- doesn't assume 100% correct labeling of crises

Closest related work

- **Capturing, indexing, clustering, and retrieving system history, SOSP '05**
  - **authors: Cohen, Zhang, Goldszmidt, Symons, Kelly, Fox**
- **Failure signatures**
  - **signature for individual servers**
  - **build and manage per-crisis classification models**
  - **detailed comparison in the paper**

Conclusion

Crisis fingerprint

- compact representation of system state
- scales to large clusters
- intuitive visualization

Use of Machine Learning crucial for metric selection

Correct identification for 80% crises

- on average after 10 minutes
- rigorous evaluation on production data

Selection of relevant metrics used at Microsoft

Thank you!