Representation learning to improve trace analysis

Quentin FOURNIER

Monday 6 May 2019

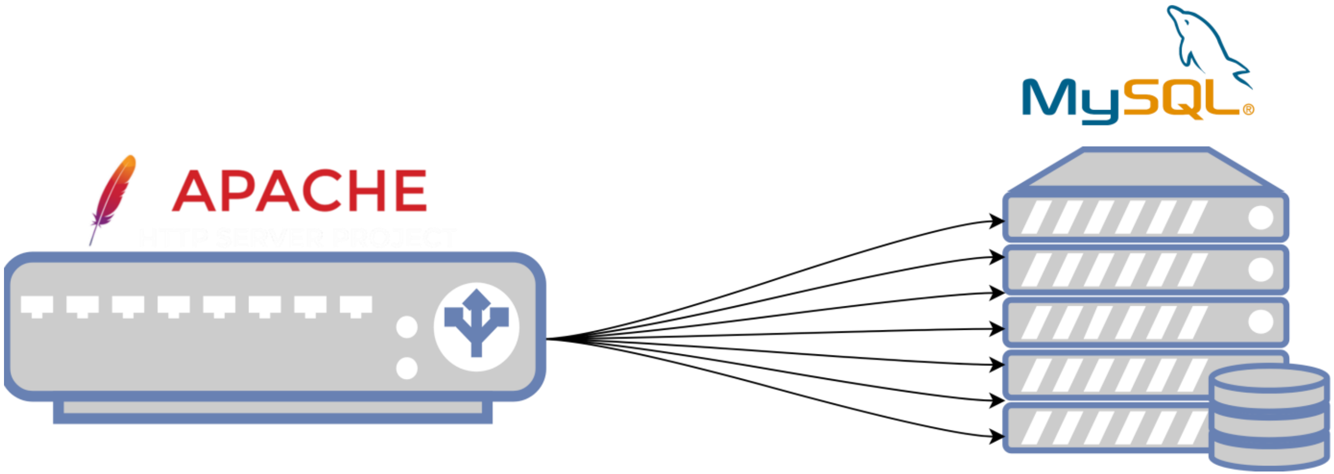
Current work

Jointly with Naser Ezzati

Goal

- Can we cluster executions/requests based on their behavior?
- Can we extract high level information (response time, ressource usage, etc.) from the clusters?
  - Can we find similar requests wihtin a cluster?
  - Can we find different requests between clusters?

Experimental setup



Request critical path

Extracting critical path states

Critical path states sequences

ap_R → ap_R → my_U → my_R → ap_P → ap_R → ap_R →...
time: 10ms

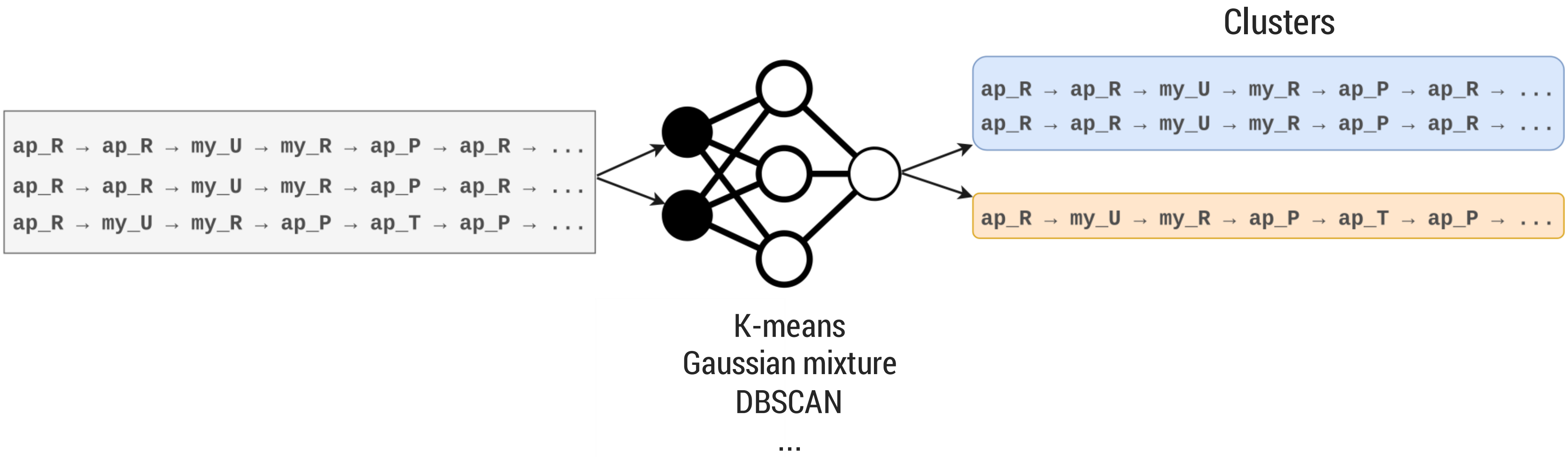
ap_R → ap_R → my_U → my_R → ap_P → ap_R → my-R →...
time: 14ms

ap_R → my_U → my_R → ap_P → ap_T → ap_P → ap-R →...
time: 36ms

...

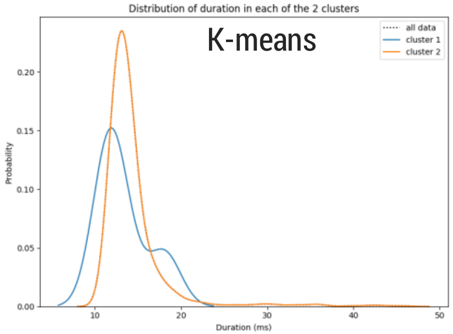
xx_R	xx running
xx_T	xx timer
xx_P	xx preemption
xx_N	xx network
xx_D	xx disk
xx_U	xx unknown

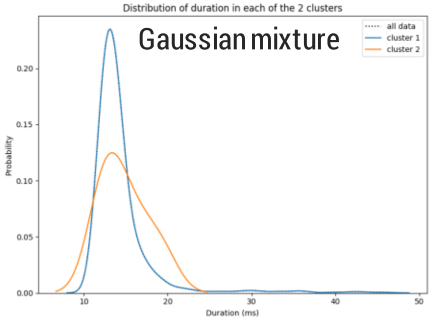
Clustering

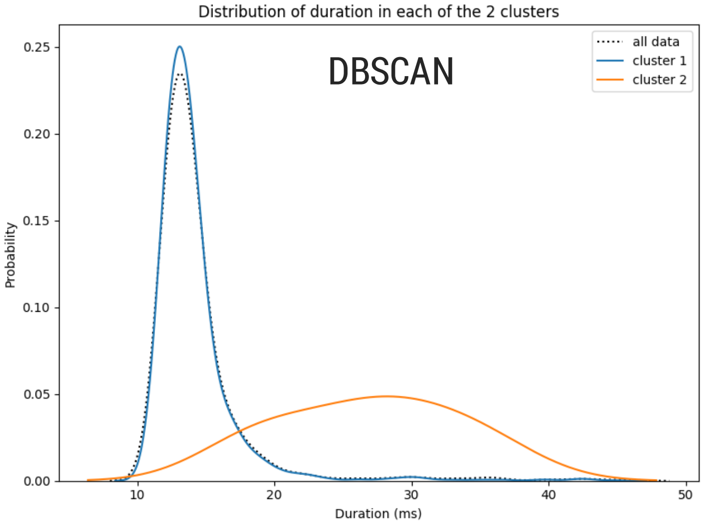


Neural Network by sachin modgekar from the Noun Project

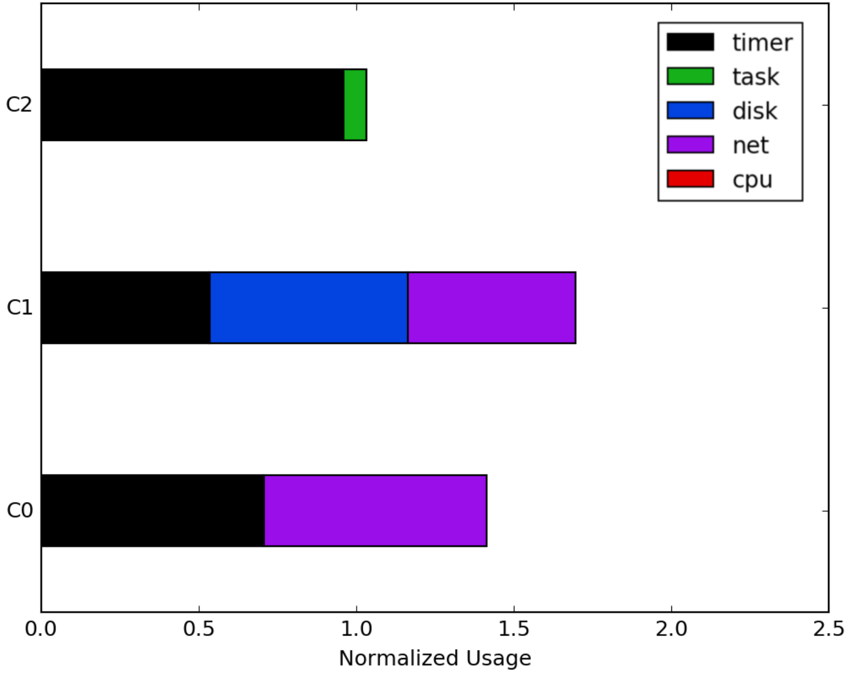
Analysis: time







Analysis: ressources



Credit to Hani Nemati & Seyed Vahid Azhari

Ngrams

Manual clustering based on duration

Top-10 ngram that differ most between each class

ngram average count	slow	fast	diff
ap_R ap_R ap_P	2.41	0.0525	2.36
ap_R ap_R ap_P ap_R	2.41	0.0525	2.36
ap_R ap_P ap_R ap_R	2.87	0.07	2.8
ap_R ap_P	3.88	0.0825	3.79
ap_R ap_P ap_R	3.88	0.0825	3.79
my_R ap_P ap_R ap_R my_R	17.3	20.8	-3.45
ap_R my_R ap_P ap_R ap_R	18.7	21.7	-2.96
ap_R my_R ap_P	20.1	22.8	-2.67
ap_R my_R ap_P ap_R	20.1	22.8	-2.67
ap_R ap_R my_R ap_P	19.9	22.6	-2.66

Ngrams

Manual clustering based on duration

Ngram that appear only in one class

ngram average count	slow	fast
ap_R ap_R my_R my_R ap_P	0	0.0075
ap_R my_R my_R ap_P	0	0.0075
ap_R my_R my_R ap_P ap_R	0	0.0075
my_R my_R ap_P	0	0.0075
my_R my_R ap_P ap_R	0	0.0075
...		
ap_R ap_R ap_P ap_R ap_P	0.255	0
ap_P ap_R ap_P ap_R ap_P	0.5275	0
ap_R ap_P ap_R ap_P	0.795	0
ap_R ap_P ap_R ap_P ap_R	0.795	0
ap_R ap_P ap_R ap_R ap_P	1.2225	0

Current work

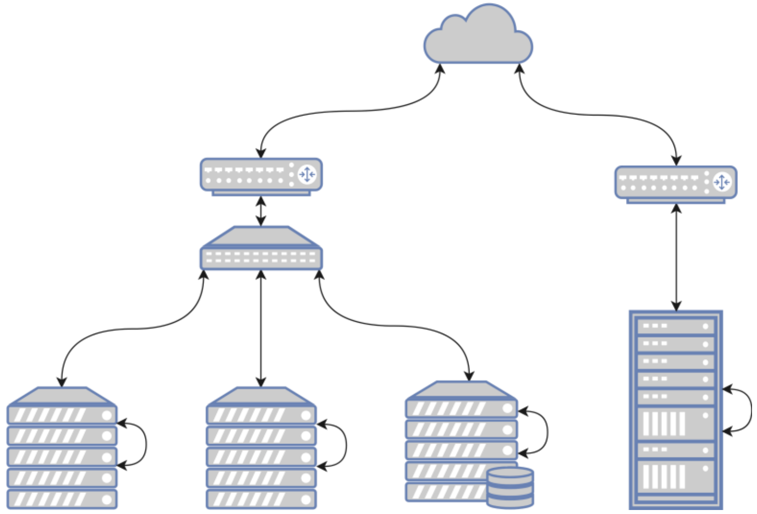
What's next?

- Try different clustering methods
- Extends clusters analysis
- Add call stack data

Future work

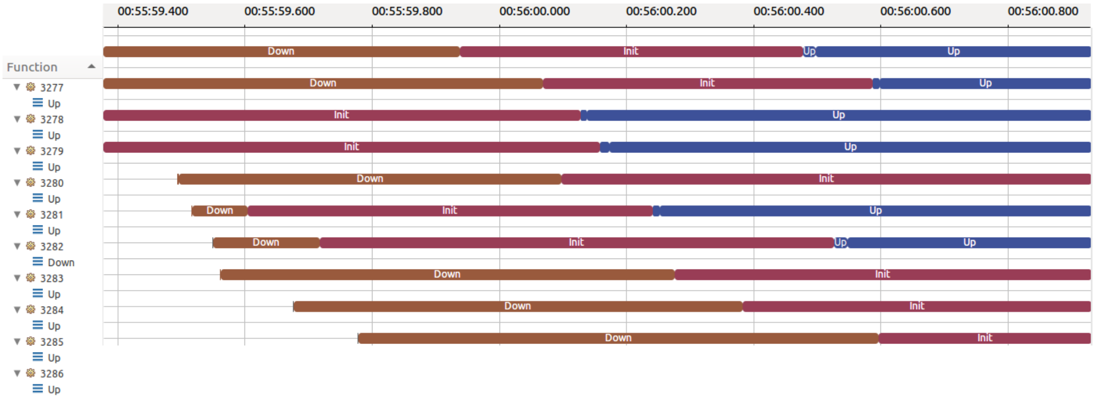
Data source

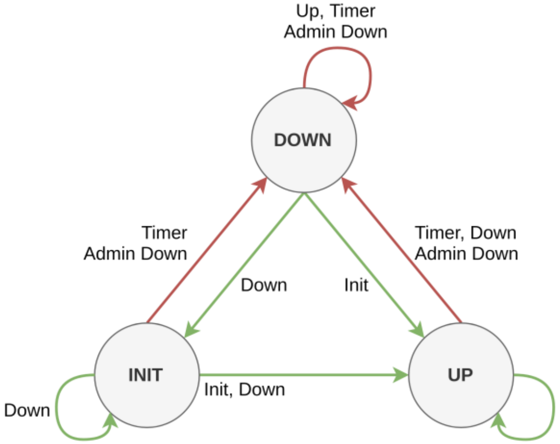
- Collaboration with Ciena
- Many protocols between multiple nodes
- Trace the state transition of each protocol
- Easy to simulate executions
- Easy to inject faults



Bidirectional Forwarding Detection (BFD)

BFD is a network protocol used to detect faults between two systems connected by a link.





Long term objectives

- Generate traces with different types of injected faults as labeld data
- Predict if a network of nodes is likely to end up faulty
- When a problem is detected, predict the fault type
- Identify the early clues of fault (≈root cause analysis)

(RFC5880) A diagnostic code specifying the local system's reason for the last change in session state. Values are:

```
   0 -- No Diagnostic
   1 -- Control Detection Time Expired
   2 -- Echo Function Failed
   3 -- Neighbor Signaled Session Down
   4 -- Forwarding Plane Reset
   5 -- Path Down
   6 -- Concatenated Path Down
   7 -- Administratively Down
   8 -- Reverse Concatenated Path Down
9-31 -- Reserved for future use
```

“ - *Thank you for your attention*