POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

DORSAL

From critical path to adaptive tracing?

Masoumeh Nourollahi
Masoumeh.nourollahi@polymtl.ca

15-Jan-2021, Progress report meeting

Agenda

- Problem
- Research questions
- Related work
- Proposed method
- Ongoing work
- Conclusions and future work

Problem definition

Large scale tracing challenges

- Data collection
  - Fixed tracing events and instrumentations
- Data analysis

Trade-off

- Quality of traces
- Tracing budget

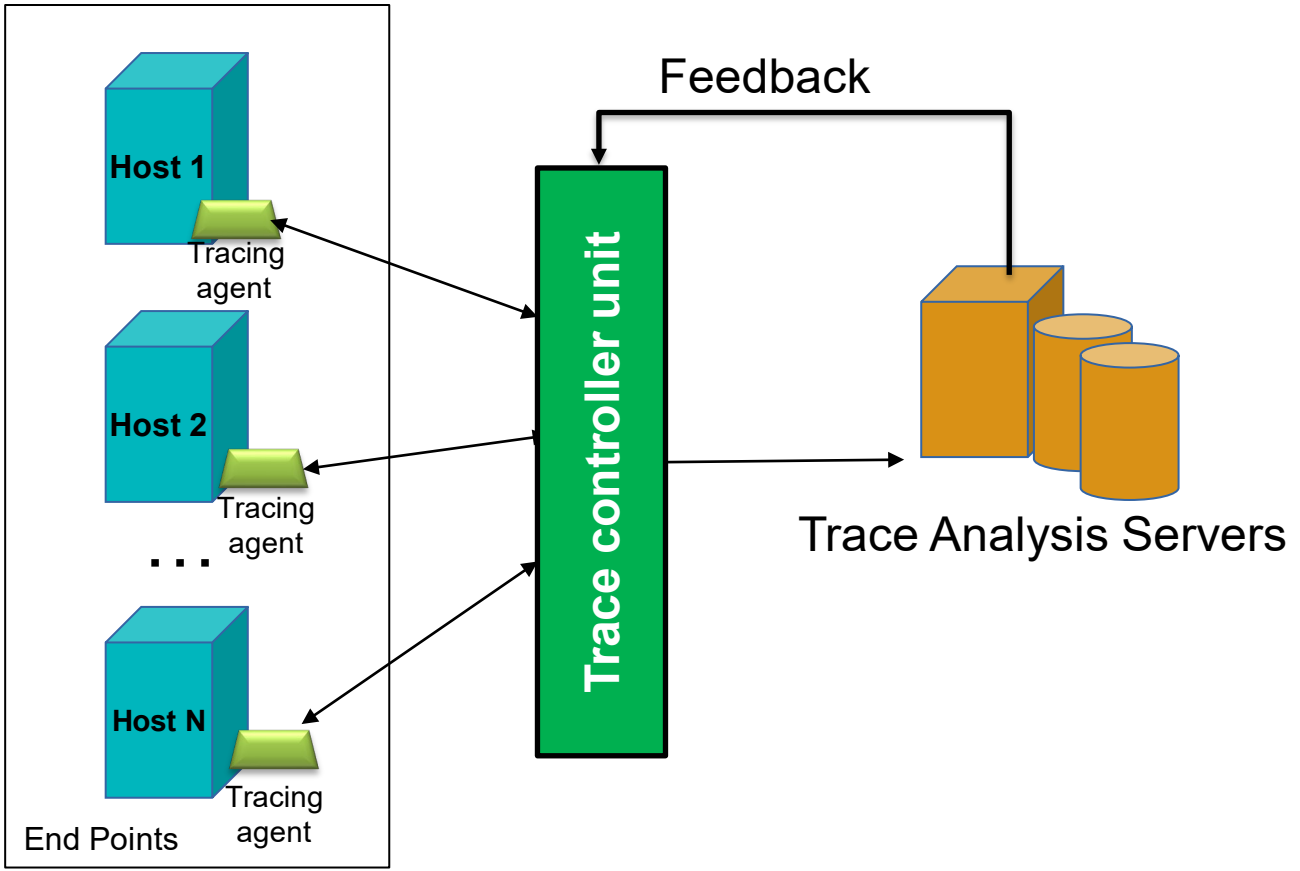
Research Questions

1. Can we use status count vectors extracted from event graphs (like critical path) to detect performance anomalies?
2. Is it possible to detect anomalies with this method in near real-time?
3. Can we use a detected anomaly in event graph status count vector to analyze root-cause?
4. Can we use anomaly root-cause to adapt tracing level of detail?

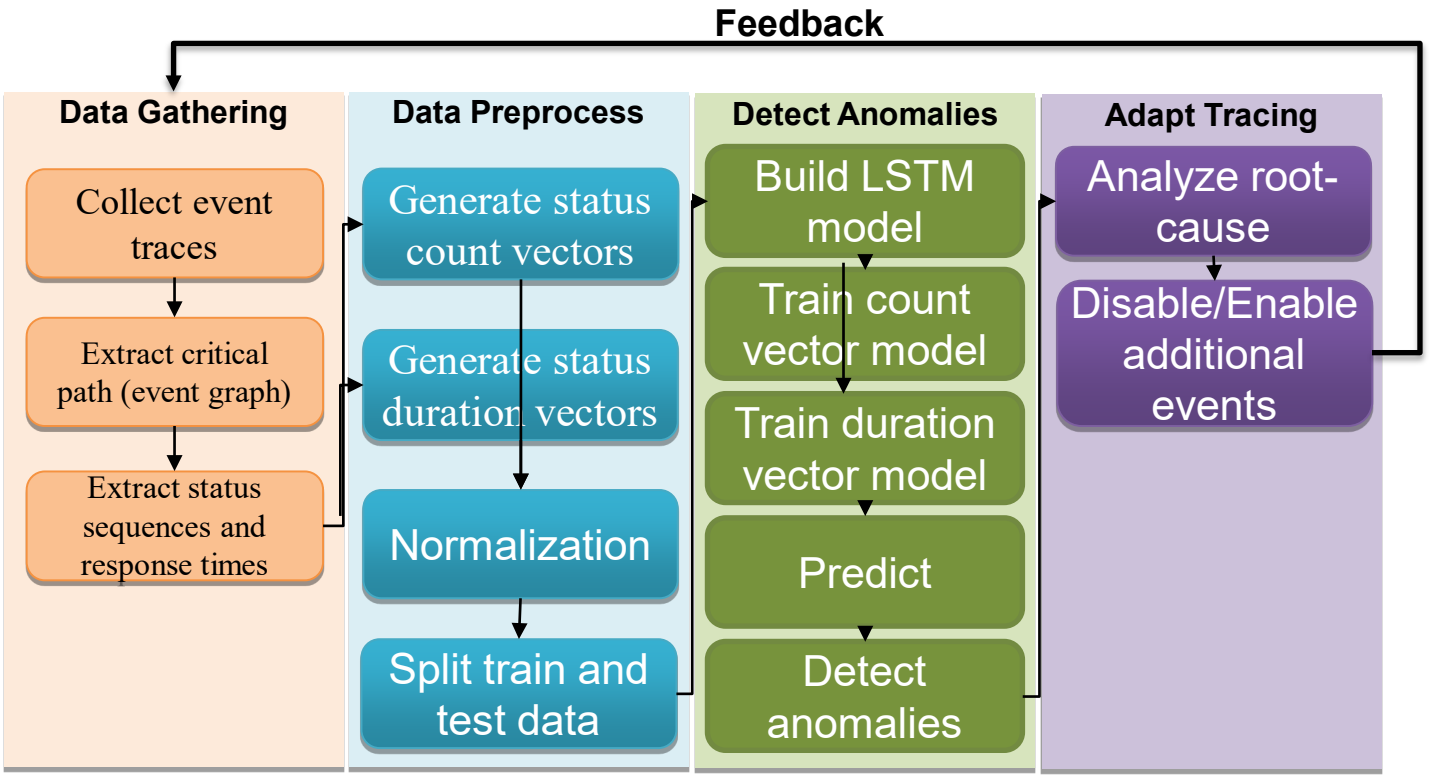
Previous work

- Joel’s demo on “Anomaly Detection with System Call Count Vectors”

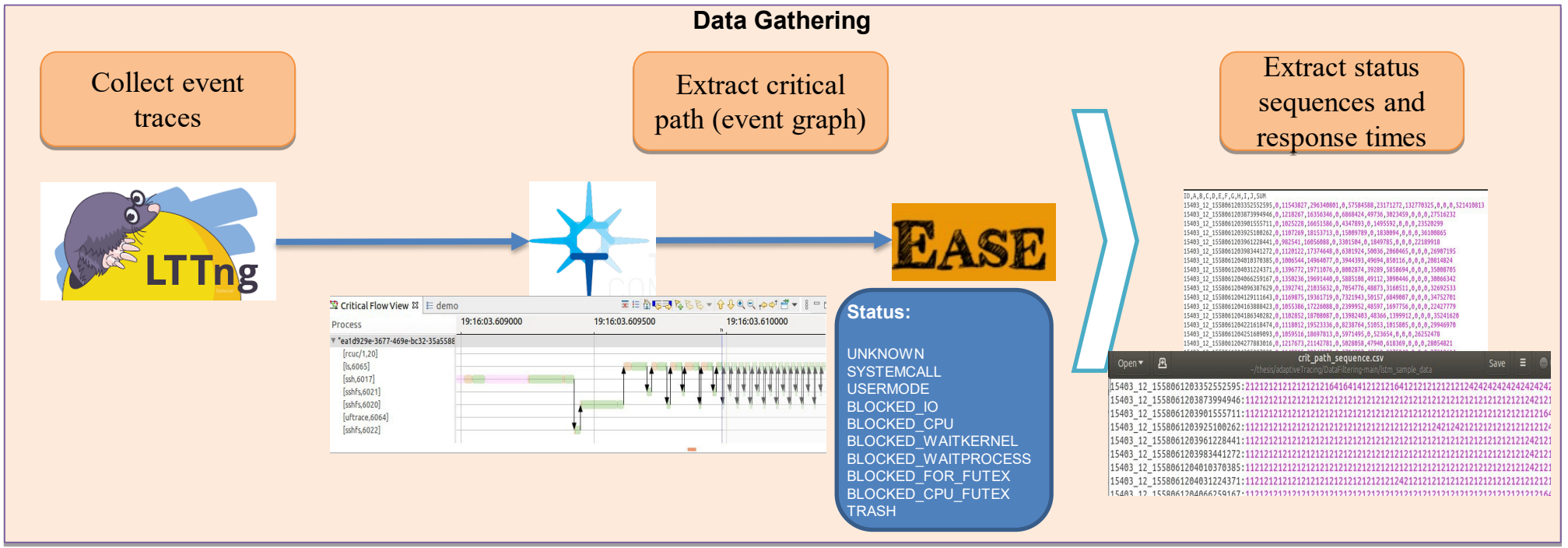
Adaptive Tracing Architecture



Adaptive tracing process



Data Gathering



Data Pre-processing

Data Preprocess

Generate state count vectors

Generate state duration vectors

Split train and test data

Normalization

```
A,B,C,D,E,F,G,H,I,J
0,2499,2626,0,402,29,221,0,0,0
0,202,209,0,38,1,14,0,0,0
0,202,225,0,56,0,15,0,0,0
0,203,205,0,37,0,16,0,0,0
0,199,207,0,35,0,13,0,0,0
0,204,213,0,46,1,16,0,0,0
0,195,213,0,37,1,8,0,0,0
0,202,203,0,34,1,15,0,0,0
0,207,205,0,41,1,20,0,0,0
```

```
ID,A,B,C,D,E,F,G,H,I,J,SUM
15403_12_1558061203352552595,0,11543827,296340081,0,57584588,23171272,132770325,0,0,0,521410813
15403_12_1558061203873994946,0,1218267,16356346,0,6868424,49736,3023459,0,0,0,27516232
15403_12_1558061203901555711,0,1025228,16651586,0,4347893,0,1495592,0,0,0,23520299
15403_12_1558061203925100262,0,1107269,18153713,0,15009789,0,1830094,0,0,0,36100865
15403_12_1558061203961228441,0,982541,16056088,0,3301504,0,1849785,0,0,0,22189918
15403_12_1558061203983441272,0,1120122,17374648,0,6301924,50036,2060465,0,0,0,26907195
15403_12_1558061204010370385,0,1006544,14964077,0,3944393,49694,850116,0,0,0,20814824
15403_12_1558061204031224371,0,1396772,19711076,0,8002874,39289,5858694,0,0,0,35008705
15403_12_1558061204066259167,0,1350236,19691440,0,5885108,49112,3090446,0,0,0,30866342
15403_12_1558061204096387629,0,1392741,21035632,0,7054776,48873,3160511,0,0,0,32692533
15403_12_1558061204129111643,0,1169875,19361719,0,7321943,50157,6849007,0,0,0,34752701
15403_12_1558061204163888423,0,1055386,17226088,0,2399952,48597,1697756,0,0,0,22427779
15403_12_1558061204186340282,0,1102852,18708087,0,13982403,48366,1399912,0,0,0,35241620
15403_12_1558061204221618474,0,1118012,19523336,0,8238764,51053,1015805,0,0,0,29946970
15403_12_1558061204251609093,0,1059516,18697813,0,5971495,0,523654,0,0,0,26252478
15403_12_1558061204277883016,0,1217673,21142781,0,5028058,47940,618369,0,0,0,28054821
```

```
train = df[: 42000]
test = df[42001:]
print("Training dataset shape:", train.shape)
print("Test dataset shape:", test.shape)
```

```
Training dataset shape: (42000, 10)
Test dataset shape: (3412, 10)
```

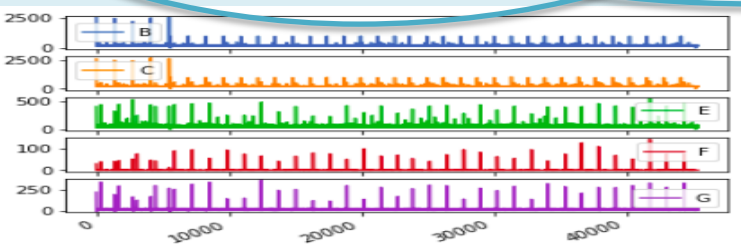
```
# normalize the data
scaler = MinMaxScaler(feature_range=(0, 1))
X_train = scaler.fit_transform(train)
X_test = scaler.transform(test)
scaler_filename = "scaler_data"
joblib.dump(scaler, scaler_filename)
```

```
['scaler_data']
```

```
# reshape inputs for LSTM [samples, timesteps, features]
X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
print("Training data shape:", X_train.shape)
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])
print("Test data shape:", X_test.shape)
```

```
Training data shape: (42000, 1, 10)
Test data shape: (3412, 1, 10)
```

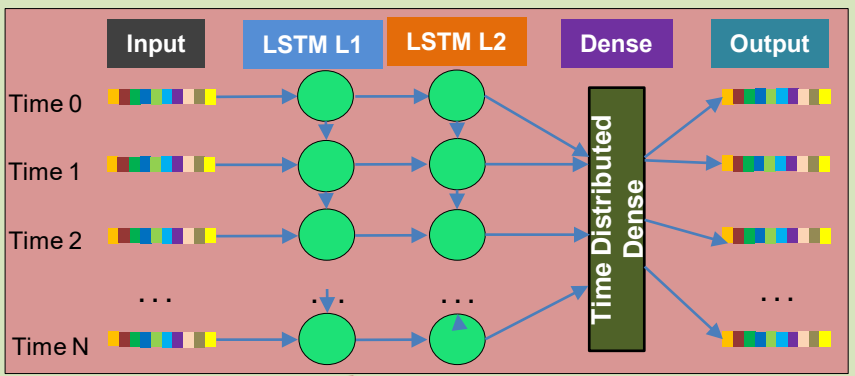
Input data overview



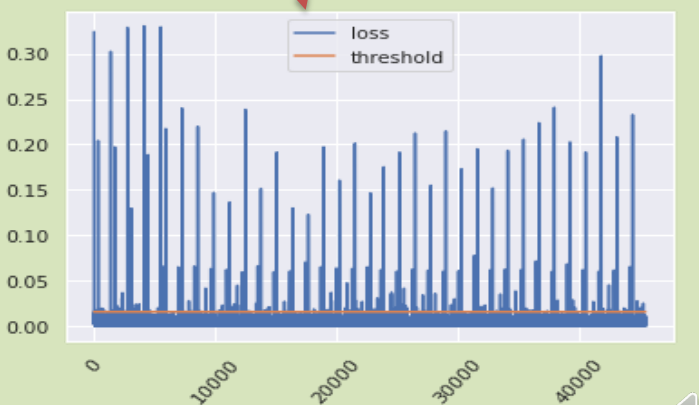
Detect Anomalies

Detect Anomalies

Build LSTM model | Train count vector model | Train duration vector model | Predict | Detect anomalies



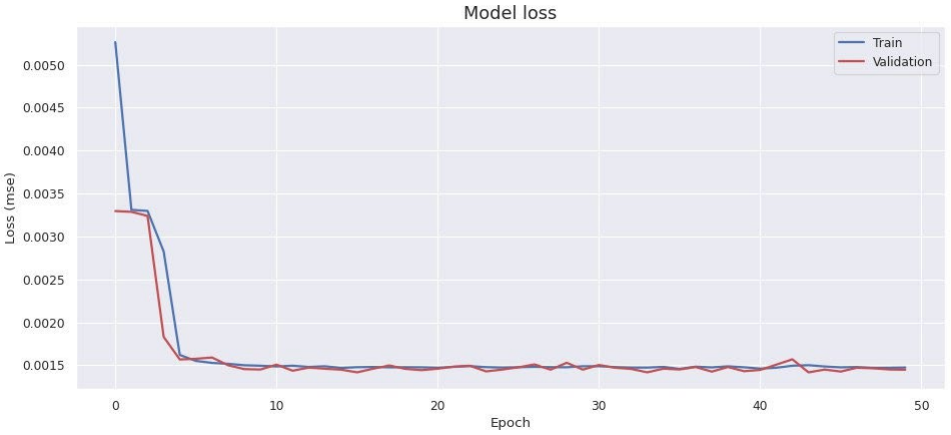

```
--- Average execution time is: 77.98811364173889 seconds ---
Accuray:  0.9999706916764362
Recall:  0.9166666666666667
Precision:  0.9999853436904587
```



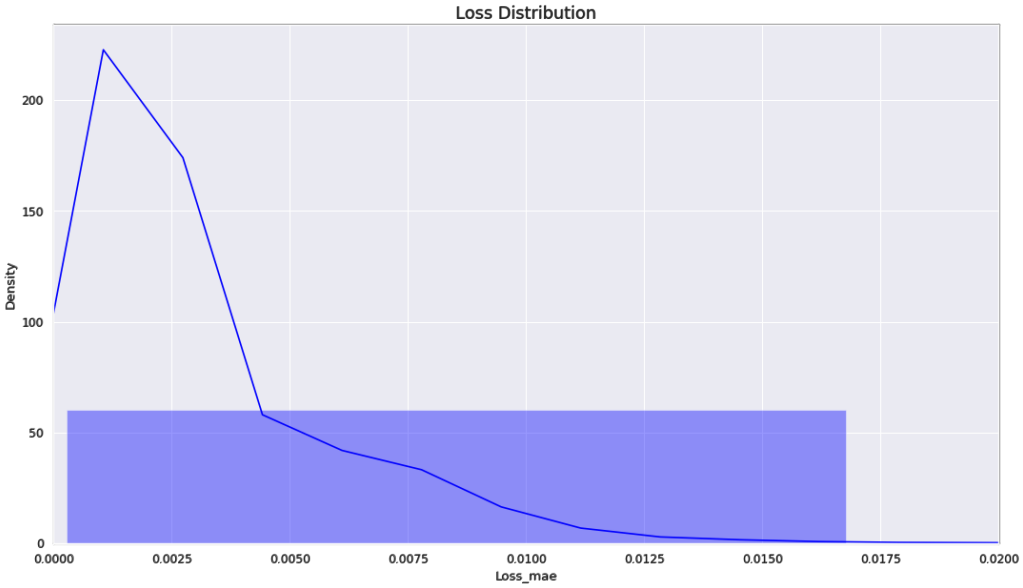

```
for i in range(10):
  # create the autoencoder model
  start_time = time.time()
  model = autoencoder_model(X_train)
  model.compile(optimizer='adam', loss='mae', metrics = [['accuracy', 'mae']])

  history = model.fit(X_train, X_train, epochs=nb_epochs, batch_size=batch_size,
                      validation_split=0.05).history
  X_pred = model.predict(X_test)
  X_pred_train = model.predict(X_train)
```

Anomaly detection

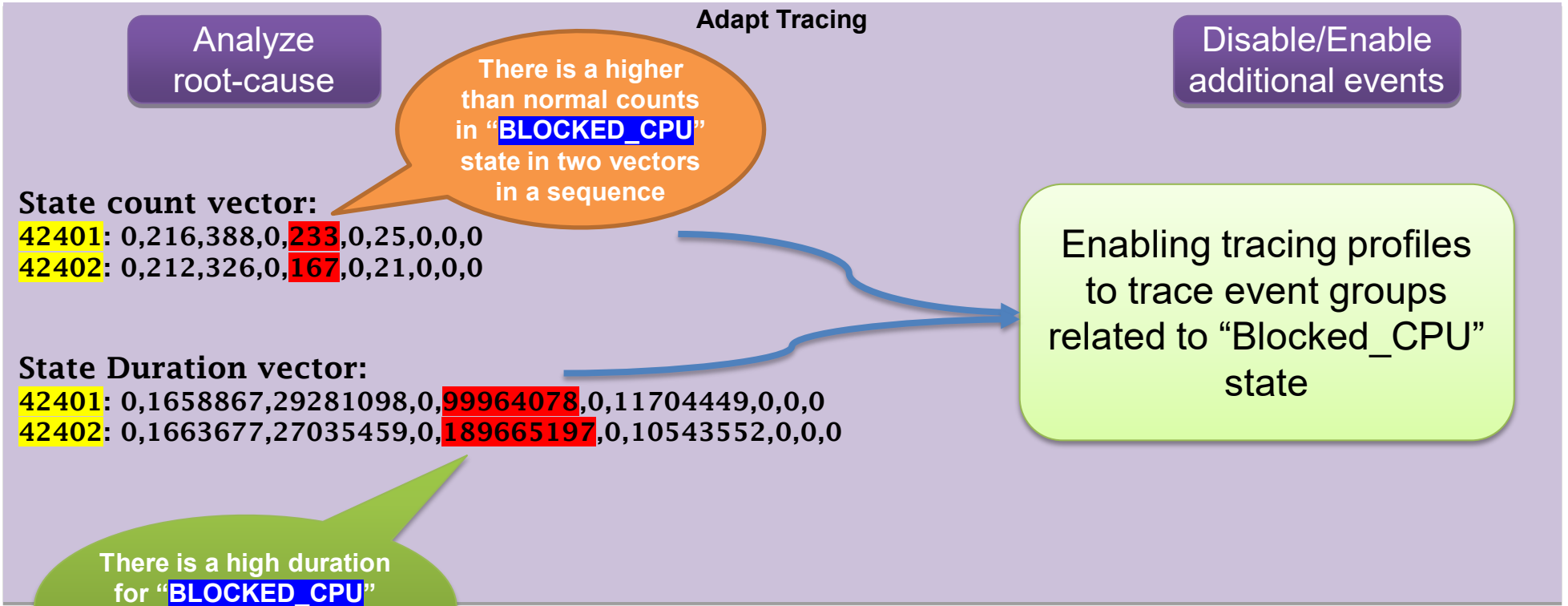



```
scored = pd.DataFrame(index=test.index)
Xtest = X_test.reshape(X_test.shape[0], X_test.shape[2])
scored['Loss_mae'] = np.mean(np.abs(X_pred-Xtest), axis = 1)
scored['Threshold'] = 0.016
scored['Anomaly'] = scored['Loss_mae'] > scored['Threshold']
scored.head()
```



By mapping the results one by one we understood that **“Status vector”** and “**Duration vector**” anomalies complement each-other

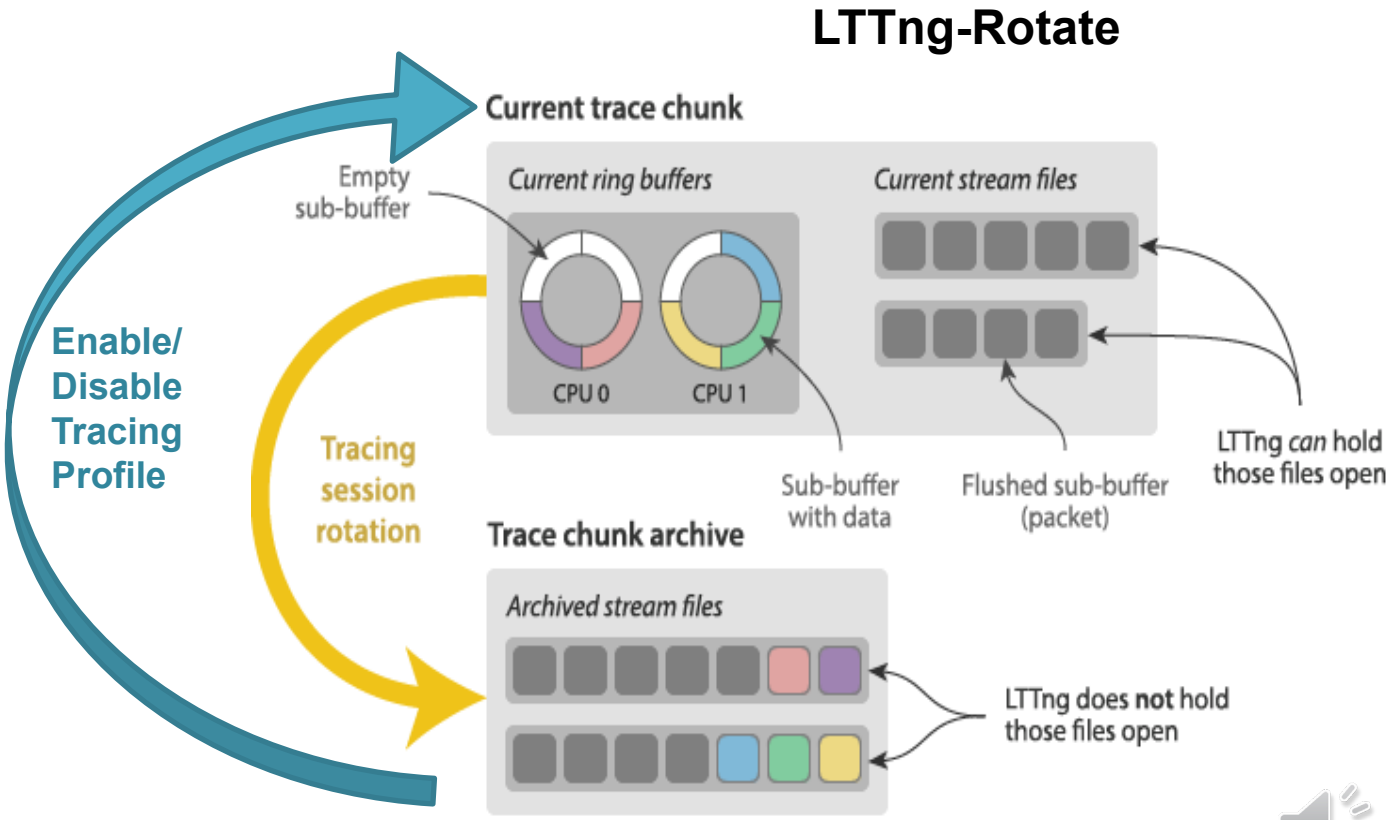
Ongoing work- Adaptive tracing



Ongoing work- Adaptive tracing

1. Event grouping to provide **tracing profiles**

2. Enhance **LTTng-Rotate** to be able to enable the required tracing profile at run-time



Ongoing work- More analysis

1. Provide an API for online anomaly detection
2. Generate traces with different types of injected faults to test effectiveness and map anomalies detected by status count vectors to system calls
3. Compare with other methods like SVM, Logistic regression, Transformer, Random Forest
4. Test the same method by considering event graph status sequence

Preliminary results

Data Set	Training time	Dataset size	Trace size	Tracing overhead	Input vector size	Accuracy	Precision	Recall
Critical path status count vector	64s	1,180,183 bytes	Smaller	Lower	10-ary	99.997	99.998	91.66
Critical path status duration vector	67s	3,906,674 bytes	Smaller	Lower	10-ary	99.993	99.995	83.33
System call count vector	66s	22,032,262 bytes	Larger	Higher	315-ary	99.993	99.949	99.919

Research Questions Revisiting

1. Can we use status count vectors extracted from event graphs to detect performance anomalies? **Yes**
2. Is it possible to detect anomalies with this method in near real-time? **Yes**
3. Can we use a detected anomaly in event graph status count vector to analyze root-cause? Ongoing research
4. Can we use anomaly root-cause to adapt tracing level of detail? Ongoing research

References

- Zhang, Shudong, et al. "Diagnostic Framework for Distributed Application Performance Anomaly Based on Adaptive Instrumentation." 2020 2nd International Conference on Computer Communication and the Internet (ICCCI). IEEE, 2020.
- Sturmann, Lilian. Using Performance Variation for Instrumentation Placement in Distributed Systems. Diss. 2020.
- Borghesi, Andrea, et al. "Anomaly detection using autoencoders in high performance computing systems." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. 2019.
- Shin, Seung Yeop, and Han-joon Kim. "Autoencoder-based One-class Classification Technique for Event Prediction." Proceedings of the 2019 4th International Conference on Cloud Computing and Internet of Things. 2019.
- Fournier, Quentin, et al. "Automatic Cause Detection of Performance Problems in Web Applications." 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2019.
- Ates, Emre, et al. "An automated, cross-layer instrumentation framework for diagnosing performance problems in distributed applications." Proceedings of the ACM Symposium on Cloud Computing. 2019.
- Kohyarnejadfard, Iman, Mahsa Shakeri, and Daniel Aloise. "System performance anomaly detection using tracing data analysis." Proceedings of the 2019 5th International Conference on Computer and Technology Applications. 2019.
- Vef, Marc-André, et al. "Challenges and solutions for tracing storage systems: A case study with spectrum scale." ACM Transactions on Storage (TOS) 14.2 (2018): 1-24.
- Dymshits, Michael, Benjamin Myara, and David Tolpin. "Process monitoring on sequences of system call count vectors." 2017 International Carnahan Conference on Security Technology (ICCST). IEEE, 2017.
- Doray, Francois, and Michel Dagenais. "Diagnosing performance variations by comparing multi-level execution traces." IEEE Transactions on Parallel and Distributed Systems 28.2 (2016): 462-474.
- J. Desfossez, M. Desnoyers, and M. R. Dagenais, “Runtime la-tency detection and analysis,”Software: Practice and Experi-ence, vol. 46, no. 10, pp. 1397–1409, 2016
- Giraldeau, Francis, and Michel Dagenais. "Wait analysis of distributed systems using kernel tracing." IEEE Transactions on Parallel and Distributed Systems 27.8 (2015): 2450-2461.

Questions?