Visualization of Performance Anomalies with Kieker

Results of Bachelor's Thesis

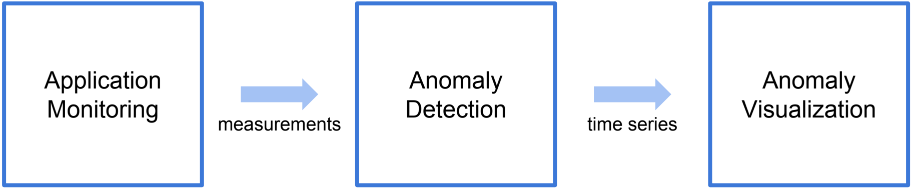
Sören Henning

November 6, 2016

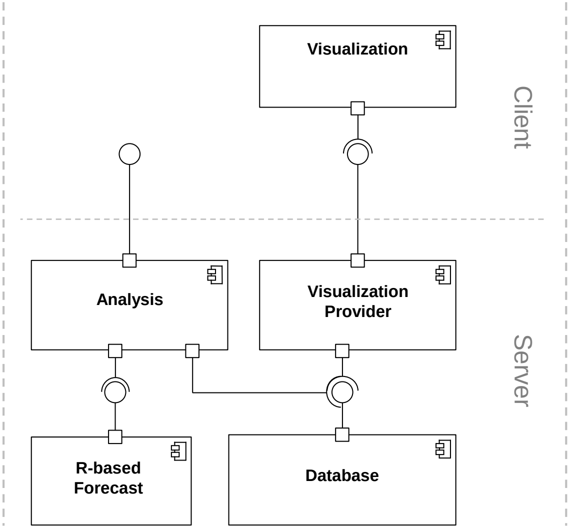
ΘPAD and ΘPADx

- Provide anomaly detection
- Part of Kieker
- Only R algorithms
- Problematic anomaly score
- No visualization
- More on this later

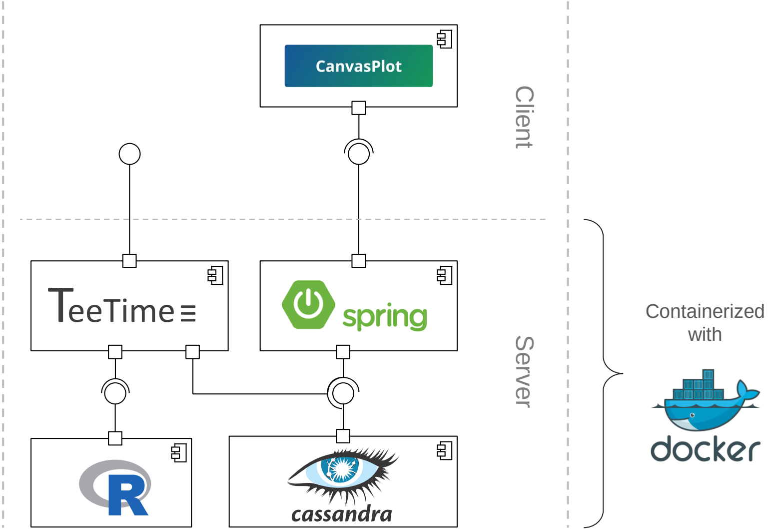
Graphical Overview of our Approach

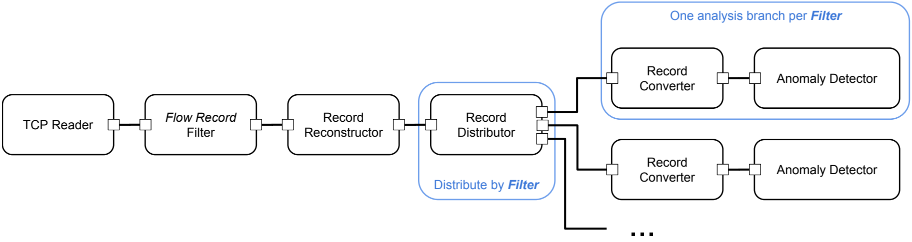


Architecture of our Approach



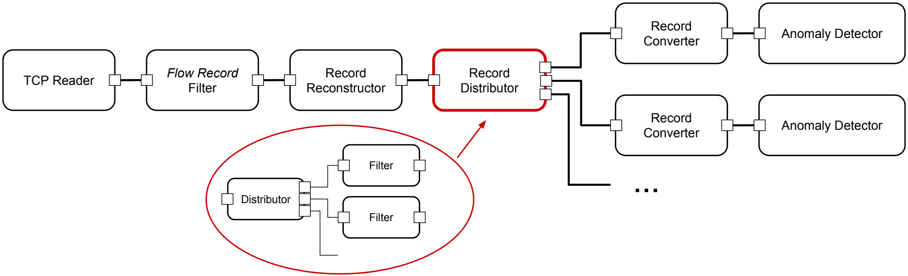
Architecture of our Implementation

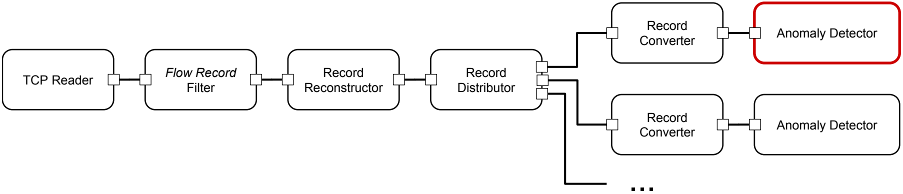




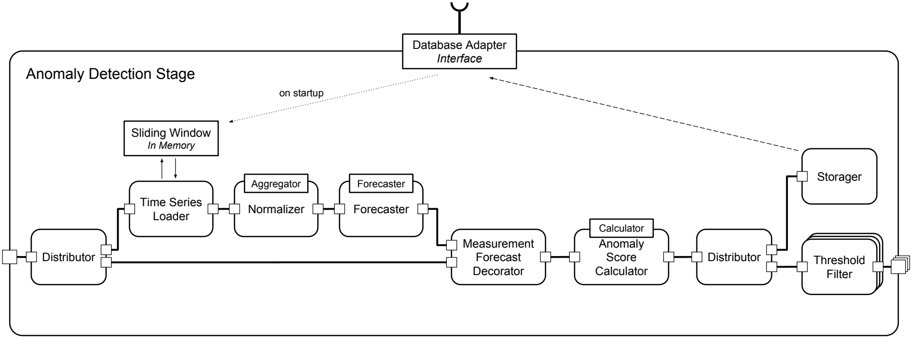
Filter:

- Operation signature
- Class signature
- Host name
- ...


TCP Reader
Flow Record Filter
Record Reconstructor
Record Distributor
Record Converter
Anomaly Detector
Record Converter
Anomaly Detector
...
Distributor
Filter
Filter


TCP Reader
Flow Record Filter
Record Reconstructor
Record Distributor
Record Converter
Anomaly Detector
Record Converter
Anomaly Detector
...

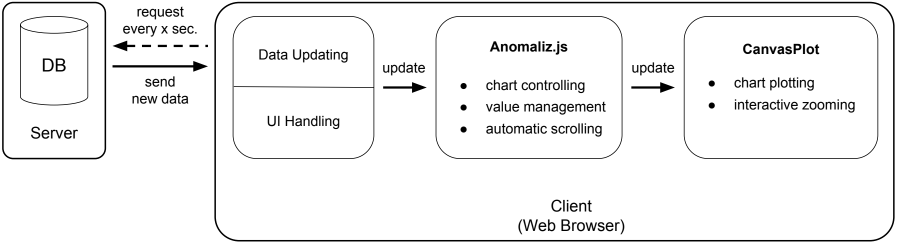
Christian-Albrechts-Universität zu Kiel
Technische Fakultät



Demo of Visualization

Architecture of Visualization

Our Approach



- Usage of Arne Johanson's CanvasPlot (Johanson 2016)

	ΘPAD	Our Approach
Architecture	monolithic	microservices
Anomaly scores calculator	bounded to $[0, 1]$, no proportional scaling	unbounded range, proportional scaling
Monitoring record processing	after filled interval	immediately
Multiple time series	separation in every single stage	separation to single *branches*
Pipe-and-Filter framework	Kieker's internal one	TeeTime
Number of implemented forecasters	1 Java-based, 8 R-based[1]	5 Java-based, 1 R-based
Database	MongoDB	Cassandra

[1] most of them introduced by Herbst et al. (2014)

Some examples:

freq.	sld. window	norm. intvl.	forecaster	∅ exec. time
5	10,000	200	Regression	1.64
5	400,000	20	Regression	4.35
50	50,000	200	ARIMA	69.98
100	100,000	500	ARIMA	78.24
150	200,000	2,000	ARIMA	187.21

...

all values in ms

Complete table: `https://build.se.informatik.uni-kiel.de/stu114708/bsc-evaluation-results`

- Further development of the ΘPAD Approach
  - Proving infrastructure via Docker containers
  - Immediately record processing

- Further development of the ΘPAD Approach
  - Proving infrastructure via Docker containers
  - Immediately record processing
- Native Java algorithms for anomaly detection

- Further development of the ΘPAD Approach
  - Proving infrastructure via Docker containers
  - Immediately record processing
- Native Java algorithms for anomaly detection
- Providing an interactive, real time visualization

- Further development of the ΘPAD Approach
  - Proving infrastructure via Docker containers
  - Immediately record processing
- Native Java algorithms for anomaly detection
- Providing an interactive, real time visualization
- All implementations available as open source: github.com/SoerenHenning

Future Work

- Handling of fast incoming measurements
  - Aggregate before analysis (ΘPAD)
  - Suggestion: Cache time series operations

Future Work

- Handling of fast incoming measurements
  - Aggregate before analysis (ΘPAD)
  - Suggestion: Cache time series operations
- Parallelized and distributed analysis
  - Is or will be supported by TeeTime

Future Work

- Handling of fast incoming measurements
  - Aggregate before analysis (ΘPAD)
  - Suggestion: Cache time series operations
- Parallelized and distributed analysis
  - Is or will be supported by TeeTime
- Take advantage of Cassandra's features for data storage

Future Work

- Handling of fast incoming measurements
  - Aggregate before analysis (ΘPAD)
  - Suggestion: Cache time series operations
- Parallelized and distributed analysis
  - Is or will be supported by TeeTime
- Take advantage of Cassandra's features for data storage
- Configuration via Rest/GUI

References

Herbst, Nikolas Roman et al. (2014). "Self-adaptive workload classification and forecasting for proactive resource provisioning". In: *Concurrency and Computation: Practice and Experience* 26.12, pp. 2053–2078. ISSN: 1532-0634.

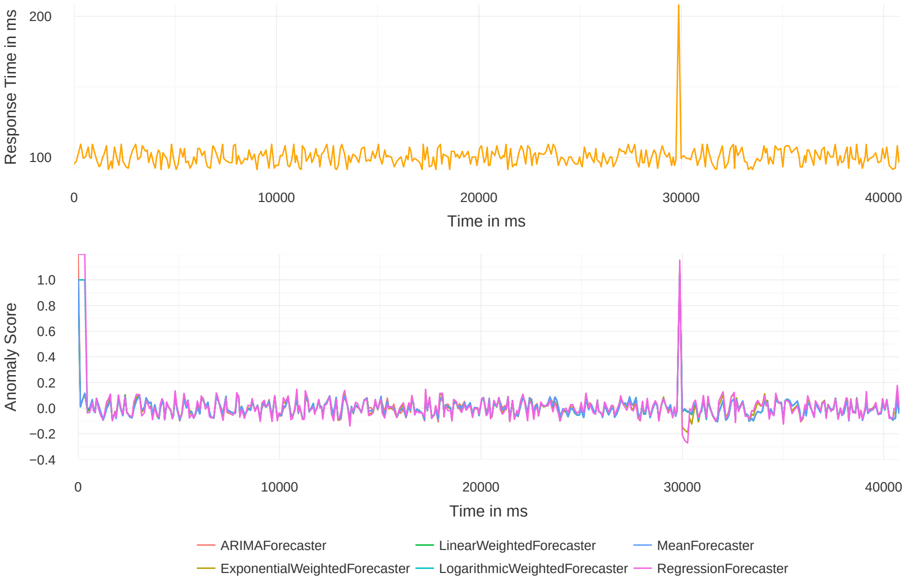
Johanson, Arne (2016). *CanvasPlot*. Accessed: 2016-09-08. URL: `https://a-johanson.github.io/canvas-plot/`.

- Take time for record processing in analysis
- Evaluate: Execution time $\leq$ measurement frequency ?
- For all parameter combinations:

Call Distance	Sliding Window	Normalization Interval	Forecaster	Aggregator
2 ms	10,000 ms	10 ms	ARIMA	Mean
5 ms	50,000 ms	20 ms	Regression	
10 ms	100,000 ms	100 ms		
50 ms	150,000 ms	200 ms		
100 ms	200,000 ms	500 ms		
150 ms	400,000 ms	1,000 ms		
200 ms		2,000 ms		

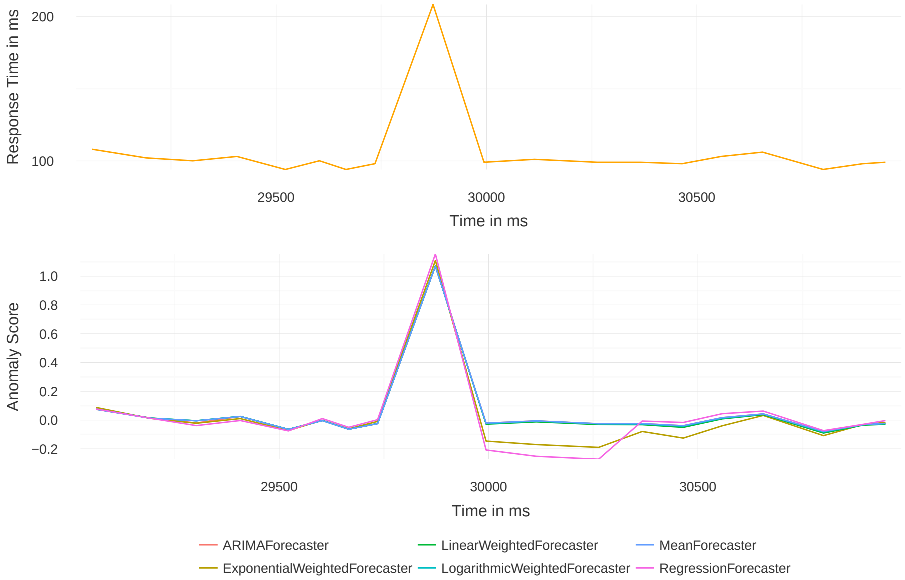
Feasibility Evaluation

Scenario: Constant with Anomaly



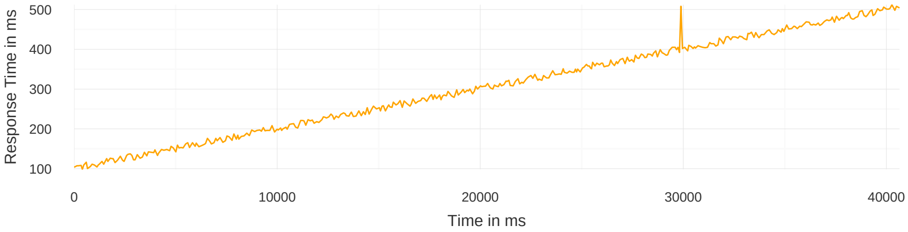
Feasibility Evaluation

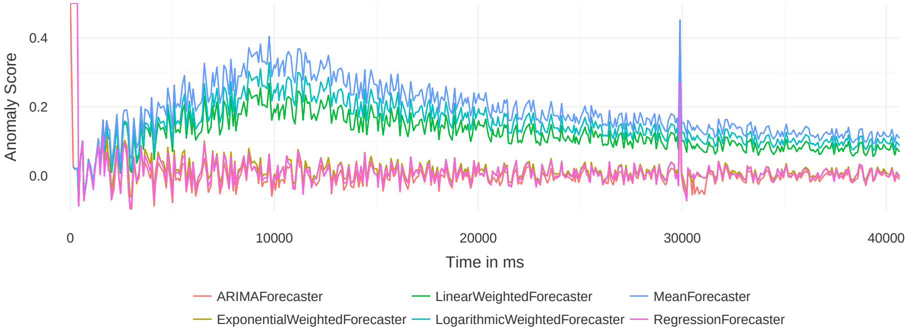
Scenario: Constant with Anomaly - Detail



Feasibility Evaluation

Scenario: Linearly Increasing with Anomaly

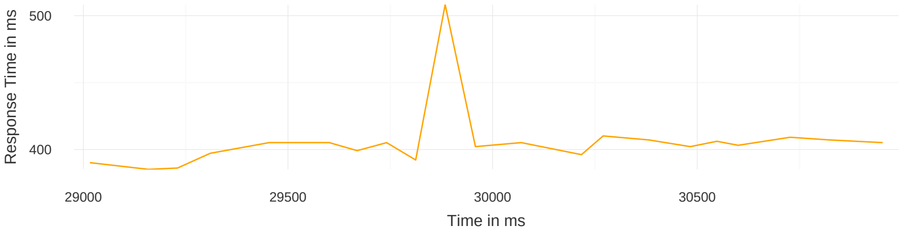


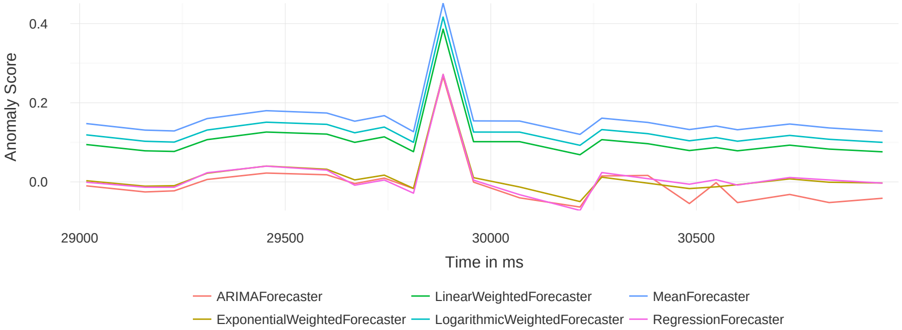


Feasibility Evaluation

Scenario: Linearly Increasing with Anomaly - Detail

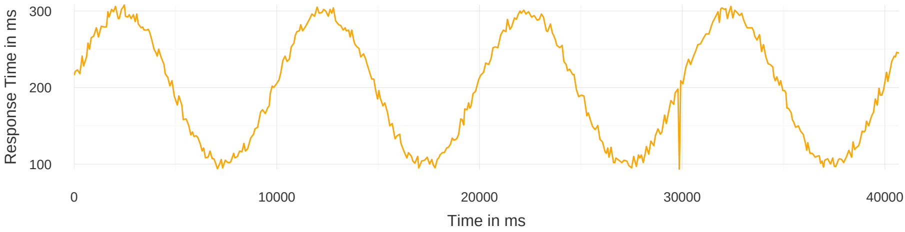
Christian-Albrechts-Universität zu Kiel
Technische Fakultät

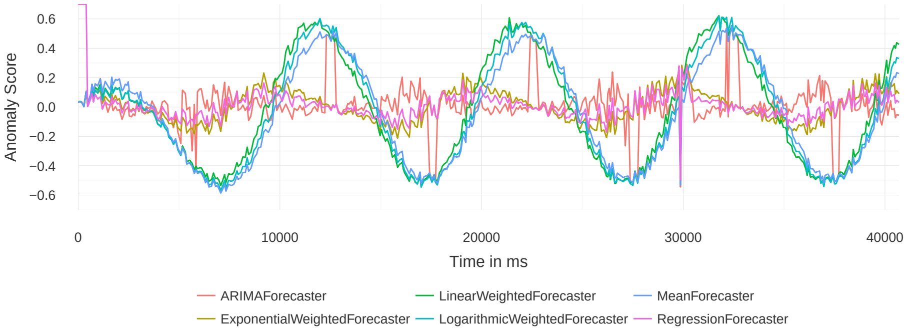




Feasibility Evaluation

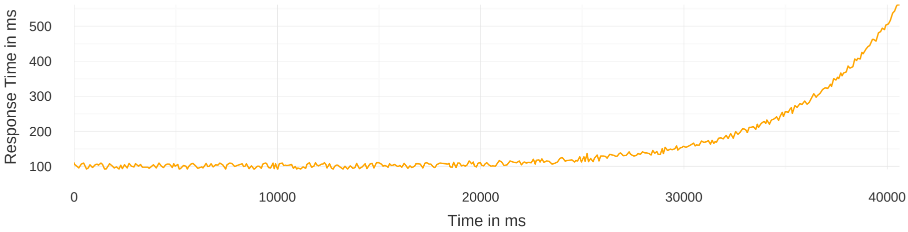
Scenario: Seasonal with Anomaly

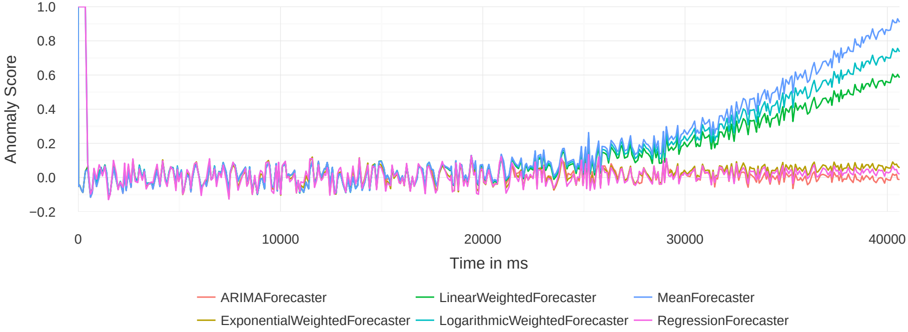




Feasibility Evaluation

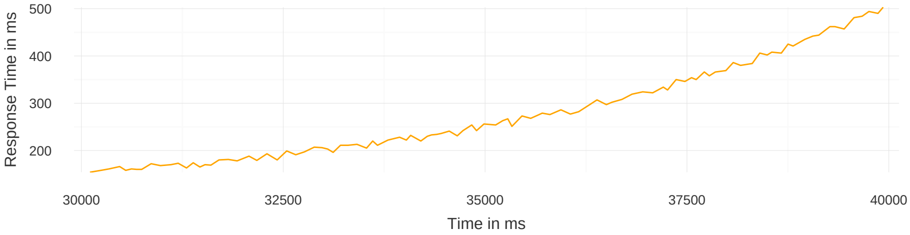
Scenario: Seasonal with Anomaly - Detail

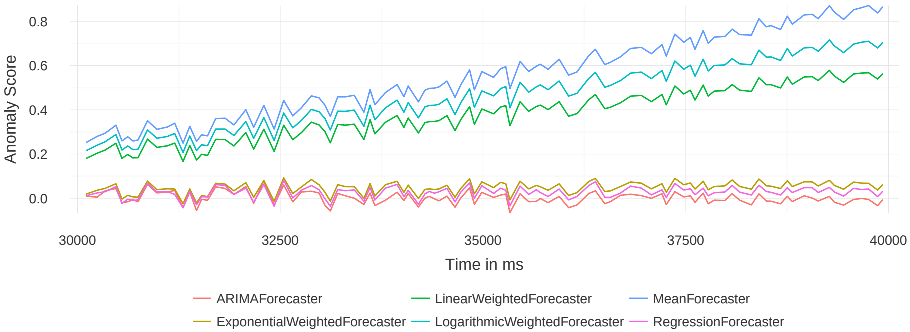

Response Time in ms
500
400
300
200
100
0
10000
20000
30000
40000
Time in ms


Anomaly Score
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
10000
20000
30000
40000
Time in ms
ARIMAForecaster
LinearWeightedForecaster
MeanForecaster
ExponentialWeightedForecaster
LogarithmicWeightedForecaster
RegressionForecaster

Feasibility Evaluation

Scenario: Exponential Increasing - Detail





Screenshots of Visualization

Screenshots of Visualization