Modelowanie i analiza systemów webowych

Tomasz Rak

Katedra Informatyki i Automatyki
Politechnika Rzeszowska

11 czerwca 2025

RZESZOW UNIVERSITY OF TECHNOLOGY

Spis treści

- Modelowanie systemu
- Analiza wydajności
- [Generatory zdarzeń]
- Analiza logów...

Modelowanie systemu

"All models are wrong, but some are useful." — George Box

- Introduction
- Cluster-based Web System Architecture
- Mathematical Model
- Performance Analysis

Approaches

(Chen, X., Czachórski, T., Jarabo, R., Ignacio, J., Kounev, S., Koziolek, H., Meier, P., Merseguer, J., Mironescu, I. D., Nguyen, T. B., Pant, A., Rathfelder, C., Requeno, J., Spinner, S., Zatwarnicki, K., Xiong, X., Zhou, J.)

We can not always add more and more new devices to improve performance, because the initial and maintenance cost will become too high. Power consumption depends on the load and on the number of running nodes in the cluster-based distributed Web system.

What is the performance of the system?

The main aim of the work was to develop models of cluster-based distributed Web system.

The related works can be divided into publications based on analysis of QN and PN models.

Computer system (experiments) and performance model (simulations)

Workload
⇓
Configuration parameters
⇓
Computer system
⇓
Measured performance parameters

Workload model
⇓
Configuration parameters
⇓
Performance model
⇓
Calculated performance parameters (response time)

Rak, T.: Response Time Analysis of Distributed Web Systems Using QPNs. Mathematical Problems in Engineering (2015) 1–10

Multi-Layered Laboratory Environment

Server/Parameters	Experiment
"Clients"	10.10.10.1
GlassFish AS nodes	10.10.10.4-5
Oracle DBS node	10.10.10.2
AS threads pool	30 per node
DBS connections pool	40
Number of RPS	15-60
Number of clients	500
Experiment time [s]	300

Application has all important functionalities for online stock trading system.
Transaction emulates a specific kind of client session.

Rak, T.: Performance Modeling Using Queueing Petri Nets. Communications in Computer and Information Science, vol. 718, Springer (2017) 321–335

My approach joins LT and PE

- Educated Guess
- Load Testing (LT)
- Performance Engineering (PE) models (provide some recommendations to realize the required performance level):
  - Performance model (used to predict performance of the system under study)
  - Availability model
  - Reliability model
  - Cost model

Rak, T., Werewka, J.: Performance analysis of interactive internet systems for a class of systems with dynamically changing offers. Lecture Notes in Computer Science, vol. 7054, Springer (2012) 109–123

Queueing Nets and Petri Nets

Queueing Nets have queues, scheduling disciplines and are suitable for modeling competition of equipment (QNs – quantitative analysis).

Petri Nets have tokens representing the tasks and are suitable for modeling software (PNs – qualitative analysis).

Queueing Petri Nets have the advantages of QNs (e.g., evaluation of the system performance, the network efficiency) and PNs (e.g., logical assessment of the system correctness). QPNs integrate hardware and software aspects of the system behaviour into the same model. QPNs add queueing and time aspects to the net.

QNs

- Arrival process[1] e.g. Poisson, Erlang, Hyper-exponential, General
- Service process is the time which each request spends at the station e.g. Logarithmic, Chi-square, Hyper-exponential, Exponential[2]; Service times are Independent and Identically Distributed
- Scheduling strategies (queueing disciplines) e.g.: First In First Out (FIFO), Last In First Out, Last In First Out with Preempt and Resume, Round Robin with a fixed quantum, Small Quantum $\Rightarrow$ Processor Sharing (PS), Infinite Server (IS) = fixed delay[3]
- Number of servers[4]
- Number of buffers (waiting room size[5])

[1] We analyzed closed queueing networks.

[2] We analyzed queueing systems with the exponential clients' service process.

[3] We used IS for clients station, PS for FE servers and FIFO for BE server.

[4] This model considers a single server queue.

[5] Size of the queue is infinite.

PNs

- Set of places
- Set of transitions
- Token color function[6]
- Incidence function (routing probability[7]) assigns natural numbers to arcs (weights of arcs)
- Initial marking[8] (number of tokens)

[6] It specifies the types of tokens that can reside in the place and allow transitions to fire in different modes.

[7] Routing of clients class N_1 contains all system resources in both layers. Routing of clients class N_2 contains only resources in FE layer.

[8] It specifies how many tokens are contained in each place.

Mathematical QPN model

$$QPN = (PL, TR, CF, IF, IM, QP, FW) \quad (1)$$

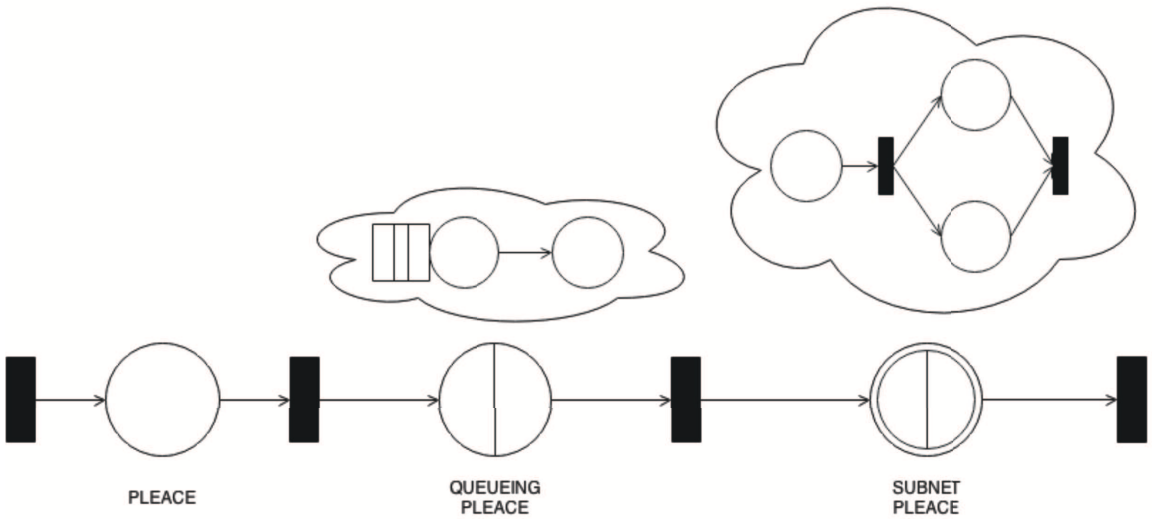
where:

- $PL = \{FE, BE, ThreadsPool, ConnectionsPool, Clients, FE\_CPU_n, BE\_I/O\}$, where $n = \{1, 2, 3\}$,
- $TR = \{t_1, t_2, t_3, t_4, t_5\}$,
- $CF(p_i)$ for $c = \{N_1, N_2, tp, cp\}$, where:
  - N_1 and N_2 - client-classes,
  - tp - threads, cp - connections,
- $IF^-(p,t)$, $IF^+(p,t)$ - if $IF^-(p,t) > 0$, an arc leads from place p to transition t and place p is called an input place of the transition, if $IF^+(p,t) > 0$, an arc leads from transition t to place p and place p is called an output place of the transition,
- $IM(p)$ for $c = \{N_1 = 250, N_2 = 250, tp_n, cp = 40\}$, where $tp_1 = 30$, $tp_2 = 60$, $tp_3 = 90$,
- $QP =$

  $(QP_1, QP_2, (-/M/1/IS_{Clients}, null, -/M/1/PS_{Sub-FE}, null, -/M/1/FIFO_{Sub-BE}, null, null))$,

  where:
  - $QP_1 = \{Clients, FE\_CPU_n, BE\_I/O\}$, where $n = \{1, 2, 3\}$,
  - $QP_2 = \varnothing$,
- $FW = (FW_1, FW_2)$, where:
  - $FW_1 = \varnothing$, $FW_2 = TR$,
  - $\forall c \in CF(t_j) : w_j(c) := 1$ (all transition firings are equally likely).

QP Net graphical notification (Queueing Petri net Modeling Environment)



- SimQPN (discrete event simulation engine)
- Queueing Petri Editor (Net Editor, Color Editor, Queues Editor)

Rak T.: Performance Analysis of Distributed Internet System Models Using QPN Simulation, Computer Science and Information Systems (FedCSIS) (2014) doi:10.15439/2014F366

Rak T.: Performance Analysis of Cluster-Based Web System Using the QPN Models, Information Sciences and Systems, Springer, pp. 239-247 (2014) doi:10.1007/978-3-319-09465-6_25

Parametry, od których zależy czas odpowiedzi

- *Service Demand, Residence Time*
- *Workload Intensivity*

Czas odpowiedzi (response time) jest równy sumie czasów obsługi w poszczególnych zasobach (residence time), gdzie: i - liczba miejsc:

$$R = \sum_{i}^{k=\mathbf{1}} R_k^{'} \tag{2}$$

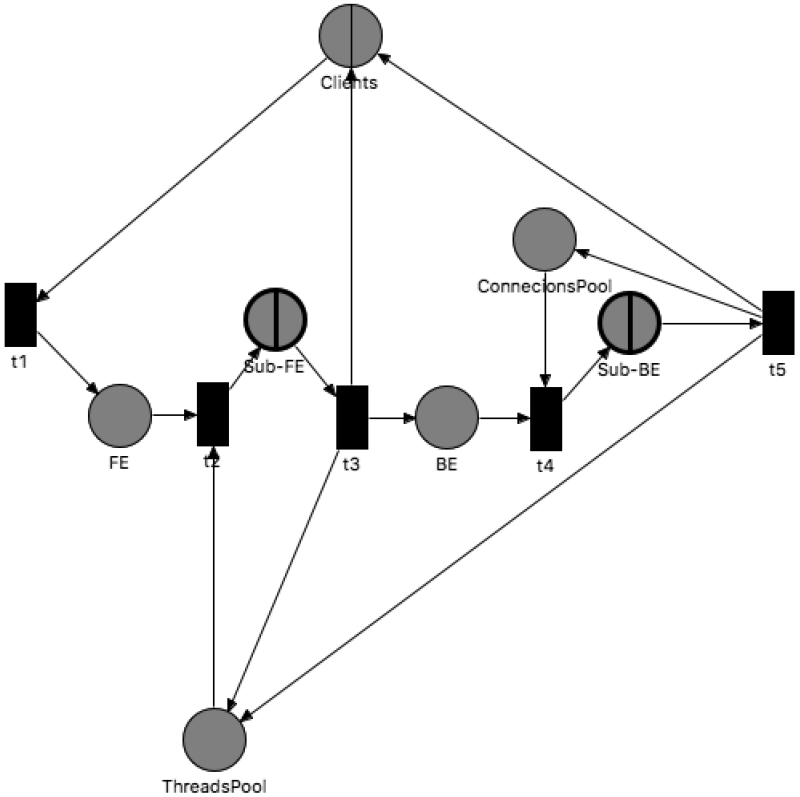
Czas obsługi w zasobie (residence time) jest sumą czasu spędzonego w kolejce (queueing time) i średniego czasu obsługi dla zasobu (service demand):

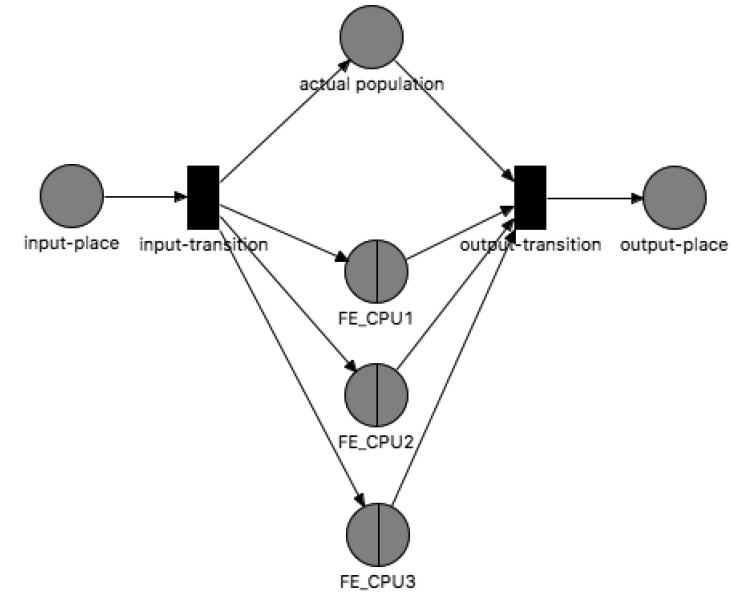
$$R_k^{'} = Q_k + D_k \tag{3}$$

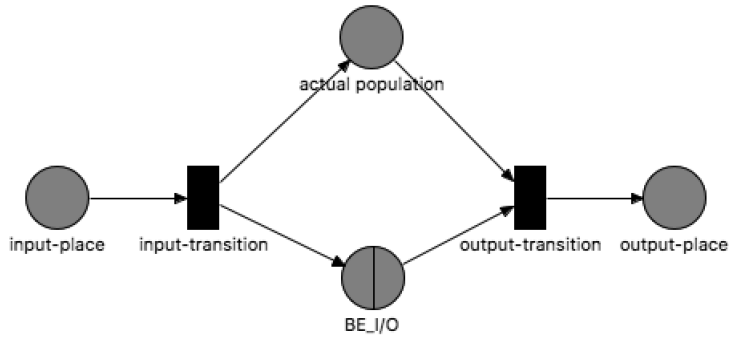
gdzie, czas spędzony w kolejce (czaso oczekiwania) na zasób to $Q_k = \sum_i^{k=\mathbf{1}} q_k$ i średni czas obsługi w określonym zasobie to $D_k = \sum_i^{k=\mathbf{1}} d_k$.

Średni czas obsługi w określonym zasobie, z wyłączeniem czasu oczekiwania na zasób. Nie zależy od obciążenia!

Model of the system







QPN net is used to predict the system response time.

Types of queues and tokens

Queues

Name	Scheduling Strategy	Number Of Servers	Description
QU_Clients	IS	1	Client queue
QU_FE1	PS	1	FE1 queue
QU_FE2	PS	1	FE2 queue
QU_FE3	PS	1	FE3 queue
QU_BE	FCFS	1	BE queue

- Clients node (*CLIENTS* queueing place) is modeled by queueing place with Infinite Server scheduling strategy (-/M/1/IS/∞ queueing system)
- Nodes of FE layer (*FE_CPU* queueing places) are modeled by queueing places with Processor Sharing scheduling strategy (-/M/1/PS/∞ queueing systems)
- Node of BE layer (*BE_I/O* queueing place) is modeled by queueing place with First In First Out scheduling strategy (-/M/1/FIFO/∞ queueing system)

Colors

Name	Real Color	Description
N1		FE-BE Requests Class
N2		FE Requests Class
tp		Threads
cp		Connections

- Client-classes (N_1, N_2)
- Application server threads *tp*
- Database server connections *cp*

Input parameters of simulations (client and system)

Parameter	One class (N_1)	Two classes (N_1 and N_2)
Clients queueing place	500	250+250[a]
$X_{Clients}$ [RPS]	15; 30; 45; 60	7.5 and 7.5; 15 and 15; 22.5 and 22.5; 30 and 30
ThreadsPool place	30; 60; 90[b]	30; 60; 90
ConnectionsPool place	40[c]	40
Simulation time [s]	300	300

[a] Each request emulates a specific type of client session with multiple round-trips over the system. Transactions are selected by the driver based on the mix (*Browse* 50%, *Purchase* 25% and *Manage* 25%). *Browse* requests are placed in FE layer and *Purchase*/*Manage* requests are placed in FE and BE layer, that why we use 50%/50% division for client-classes (N_1 and N_2).

[b] Threads for FE nodes respectively – Initial marking per node (1, 2, 3).

[c] Connections for BE node – Initial marking per node (1).

Think time (client) and service demand (system) for two client-classes

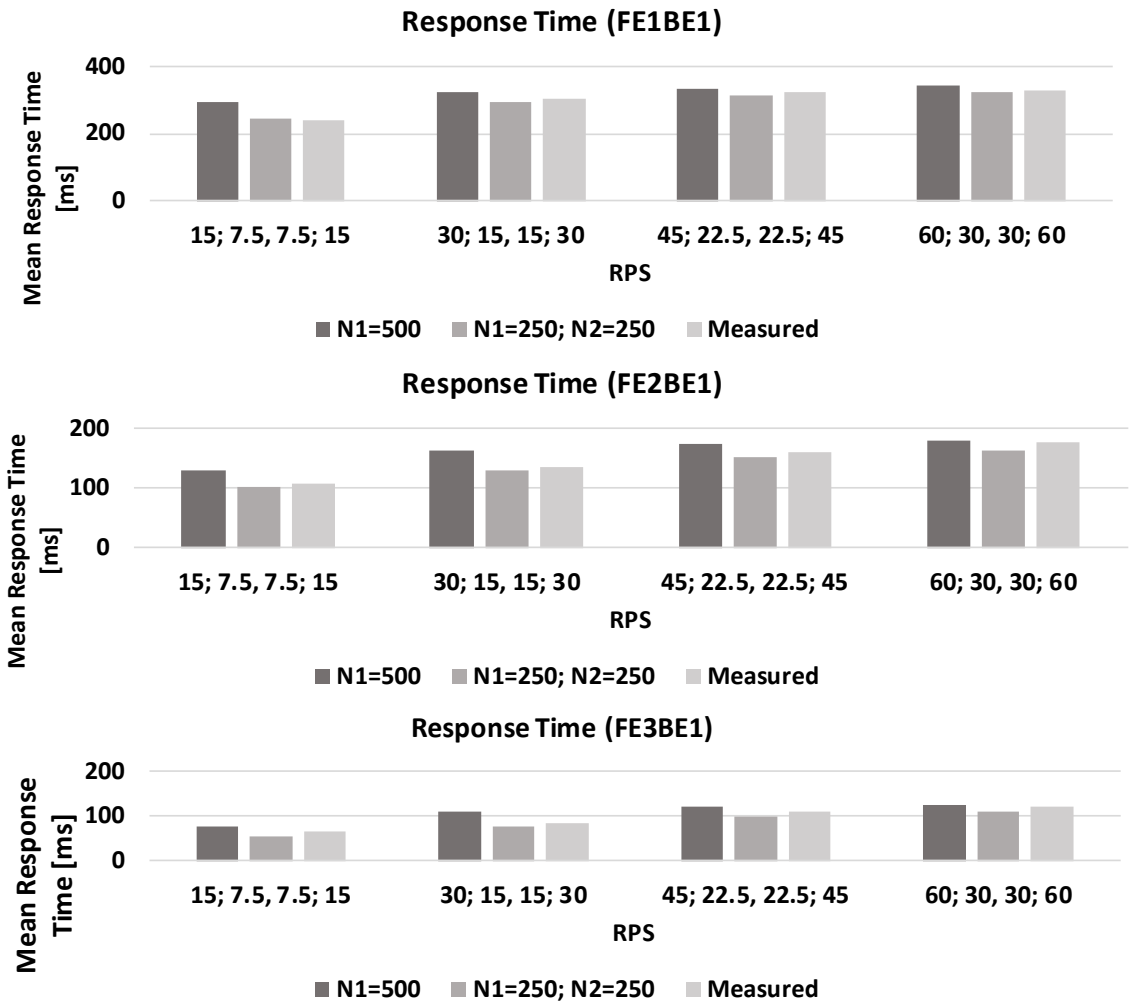
$(X_{Clients})_k[RPS]$ [a]	$(X_{Clients})_k[RPMS]$ [b]	$TT_k[ms]$ [c]
7.5; 7.5	0.0075; 0.0075	133.33; 133.33
15; 15	0.015; 0.015	66.67; 66.67
22.5; 22.5	0.0225; 0.0225	44.44; 44.44
30; 30	0.03; 0.03	33.33; 33.33

[a] *RPS* – Requests Per Second.
[b] *RPMS* – Requests Per MilliSecond.
[c] *TT* – Think Time in *Clients* queueing place.

Resource i	$(X_i)_1[RPS]$	$(X_i)_2[RPS]$	$(X_i)_1[RPMS]$	$(X_i)_2[RPMS]$	$(SD_i)_k[ms]$
FE_CPU_n	1,400	1,400	1.4	1.4	0.714
BE_I/O	7,500	7,500	7.5	7.5	0.133

Response time

One class and two classes - one, two and three nodes in FE layer (500 clients, different RPS workload)



Response time error

One class and two classes - one, two and three nodes in FE layer (500 clients, different RPS workload)

Client think time [ms]	Model with one client class [ms]	Model with two client-classes [ms]	Measured [ms]	Error for one client class [%]	Error for two client-classes [%]
133.33 and 133.33	291.14	224.58	241.00	20.63	6.94
66.67 and 66.67	323.46	291.87	303.00	6.55	3.85
44.44 and 44.44	334.45	312.62	321.00	4.16	2.64
33.33 and 33.33	340.31	324.71	330.00	2.98	1.74
133.33 and 133.33	128.58	102.11	106.66	20.55	4.27
66.67 and 66.67	162.36	128.44	135.01	20.26	4.87
44.44 and 44.44	173.49	150.63	159.42	8.83	5.51
33.33 and 33.33	178.48	162.33	175.43	1.74	7.47
133.33 and 133.33	76.46	56.23	65.12	17.41	13.65
66.67 and 66.67	110.32	76.78	85.28	29.36	9.96
44.44 and 44.44	121.23	99.38	110.83	9.38	10.34
33.33 and 33.33	126.59	109.76	120.94	4.68	9.25

Rak, T.: Cluster-Based Web System Models for Different Classes of Clients in QPN. Communications in Computer and Information Science, vol. 1039, Springer (2019) 347–365

Analiza wydajności

"If you torture the data long enough, it will confess to anything." — Ronald Coase

- Introduction
- Container-based Web System Architecture
- Experiments and Estimations
- QPN Simulations

Publications

QPN models[a]

[a] Rak T.: Modeling Web Client and System Behavior, (**2020**) doi:10.3390/info11060337
Rak T.: Cluster-Based Web System Models for Different Classes of Clients in QPN, (2019) doi:10.1007/978-3-030-21952-9_26
Rak T.: Performance Modeling Using Queueing Petri Nets (2017) doi:10.1007/978-3-319-59767-6_26
Rak T.: Response Time Analysis of Distributed Web Systems Using QPNs, (2015) doi:10.1155/2015/490835

Rak, T. Performance Evaluation of an API Stock Exchange Web System on Cloud Docker Containers. Appl. Sci. 2023, 13, 9896, https://doi.org/10.3390/app13179896

Parametry, od których zależy czas odpowiedzi

- *Service/Resource Demand, Residence Time*
- *Workload Intensivity*

Czas odpowiedzi (response time) jest równy sumie czasów obsługi w poszczególnych zasobach (residence time), gdzie: i - liczba miejsc:

$$R = \sum_{i}^{k=\mathbf{1}} R_k^{'} \tag{4}$$

Czas obsługi w zasobie (residence time) jest sumą czasu spędzonego w kolejce (queueing time) i średniego czasu obsługi dla zasobu (resource demand):

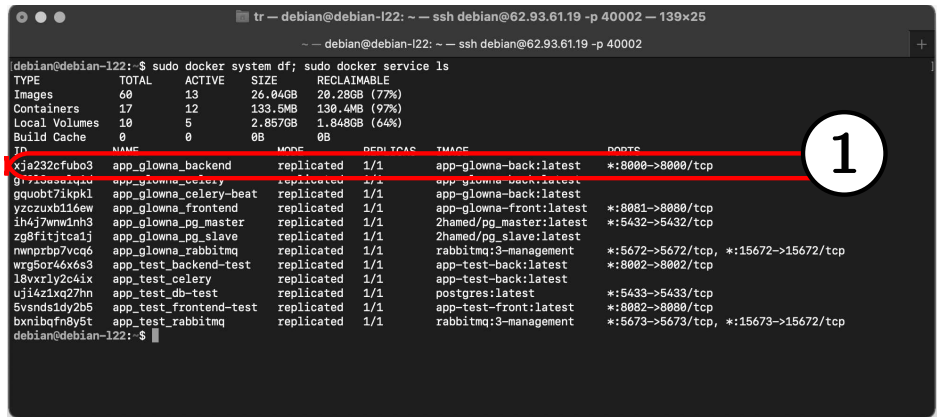
$$R_k^{'} = Q_k + D_k \tag{5}$$

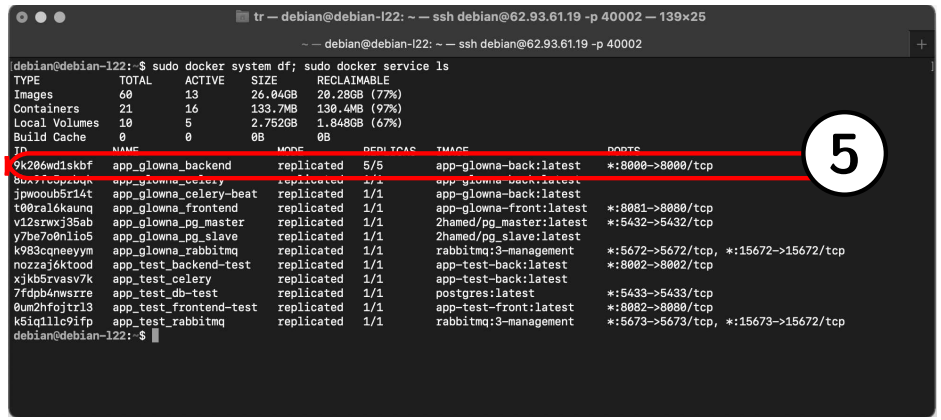
gdzie, czas spędzony w kolejce (czaso oczekiwania) na zasób to $Q_k = \sum_i^{k=\mathbf{1}} q_k$ i średni czas obsługi w określonym zasobie to $D_k = \sum_i^{k=\mathbf{1}} d_k$.

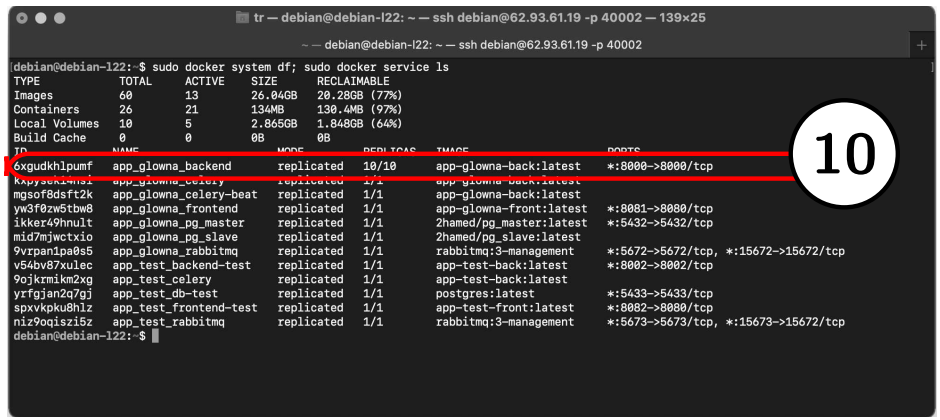
Średni czas obsługi w określonym zasobie, z wyłączeniem czasu oczekiwania na zasób.

[https://research.spec.org/tools/overview/librede.html]

Docker Engine in Swarm Mode







Debian 10 (2020.05) GL
CPU: 4 RAM: 10GB HDD: 100GB

Debian 10 (2020.05) GL
CPU: 8 RAM: 20GB HDD: 100GB

Debian 10 (2020.05) GL
CPU: 12 RAM: 30GB HDD: 100GB

Tests Scenarios

S1 - kupuj do oporu i wystawiaj oferty sprzedaży (kdoiwos)

- Rejestracja użytkownika
- Logowanie
- Lista wszystkich dostępnych zasobów
- Stan portfela obecnego użytkownika
- Kupno pojedynczego zasobu
- Lista zasobów posiadanych przez użytkownika
- Oferty sprzedaży
- Lista obecnych ofert sprzedaży/kupna danego użytkownika
- Lista zrealizowanych transakcji danego użytkownika

S2 - kupuj i sprzedawaj (kis)

- Rejestracja użytkownika
- Logowanie
- Lista wszystkich dostępnych zasobów
- Stan portfela obecnego użytkownika
- Kupno pojedynczego zasobu
- Lista zasobów posiadanych przez użytkownika
- Oferty sprzedaży
- Sprzedaż pojedynczego zasobu
- Lista obecnych ofert sprzedaży/kupna danego użytkownika
- Oferty sprzedaży

S3 - kupuj kolejne dopóki są fundusze (kkdsf)

- Rejestracja użytkownika
- Logowanie
- Lista wszystkich dostępnych zasobów
- Stan portfela obecnego użytkownika
- Kupno pojedynczego zasobu

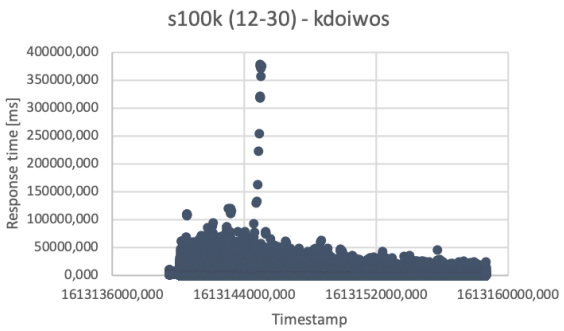
Multi-Container Laboratory Environment

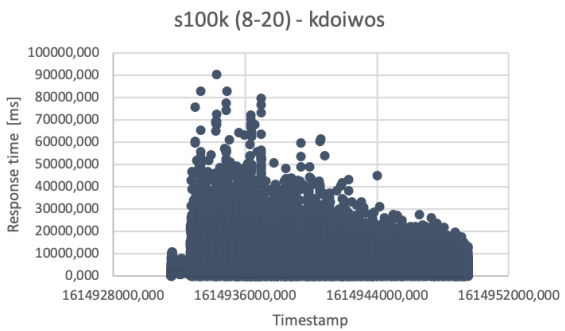
Scenario	S1	S2	S3

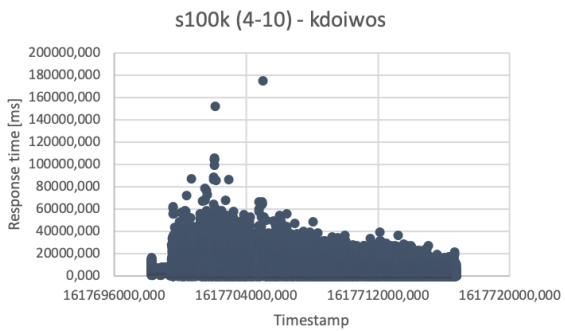
Parameter	Exp. 1			Exp. 2			Exp. 3		
Processors	12			8			4		
RAM [GB]	30			20			10		
Container [a]	1	5	10	1	5	10	1	5	10
$1/\overline{Think\_time}$ - [req/s] [b]	6,242	4,849	3,826	6,732	5,009	2,975	6,366	3,152	1,644
$\overline{Think\_time}$ - [s]	0,166	0,207	0,261	0,155	0,200	0,337	0,164	0,317	0,610

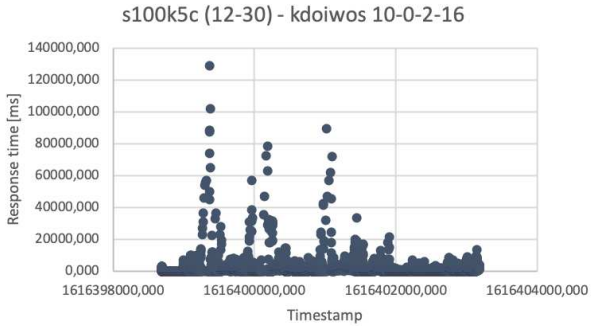
[a] Number of database connections in all cases is equal 90 per container.
[b] Number of clients (workload) in all cases is equal 90.

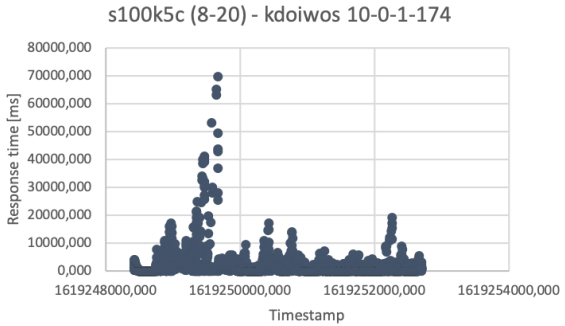
Response Time - 100000 [req]

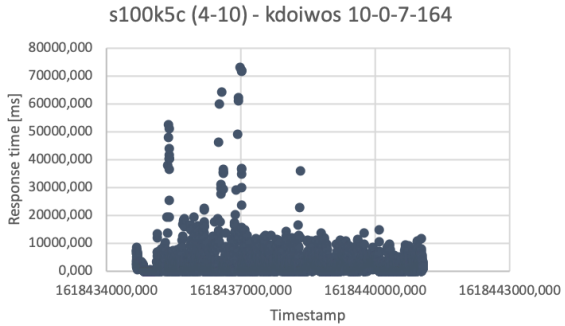


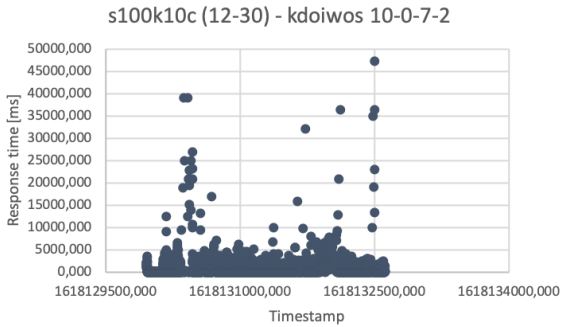


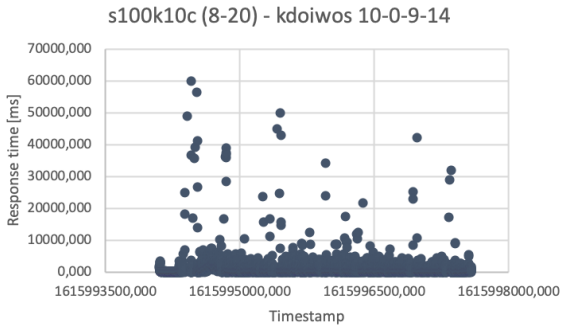


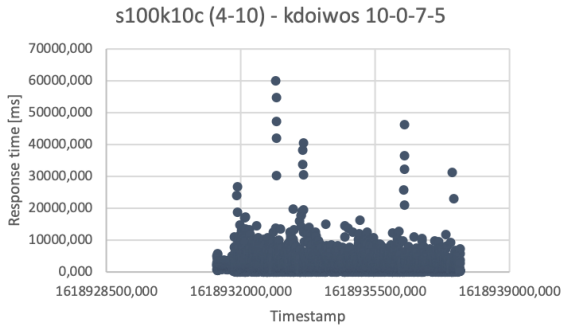




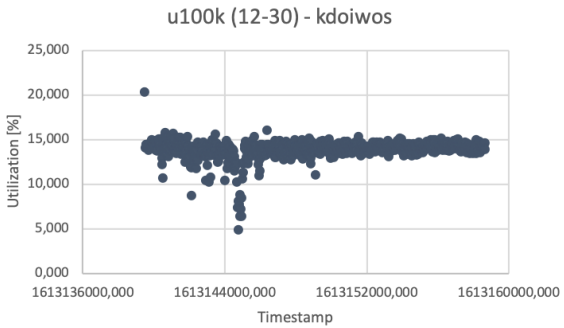


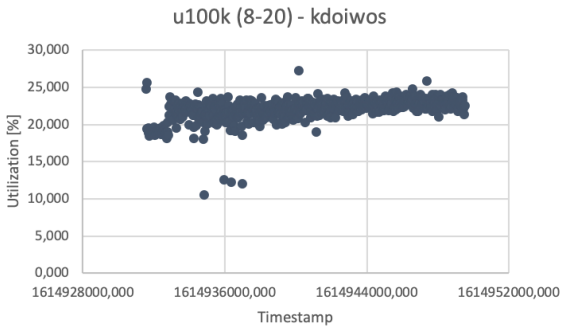


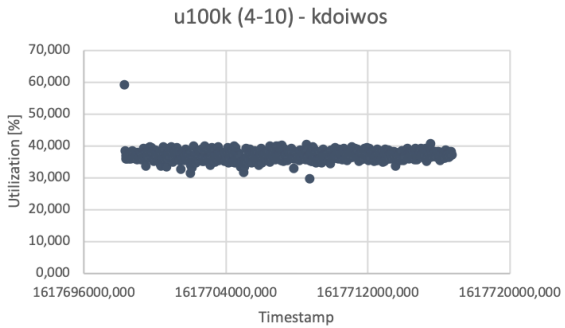


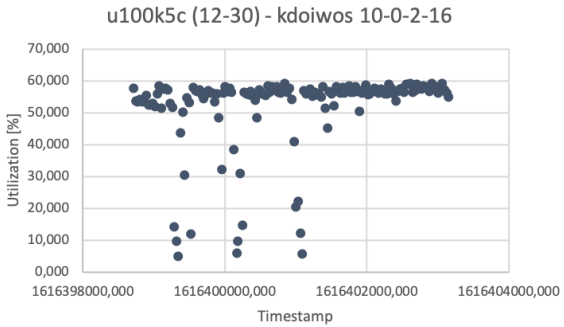


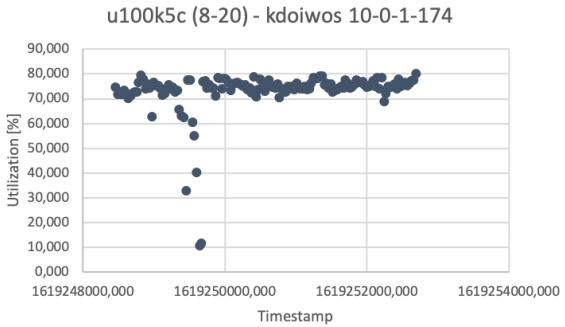
Utilization - 100000 [req]

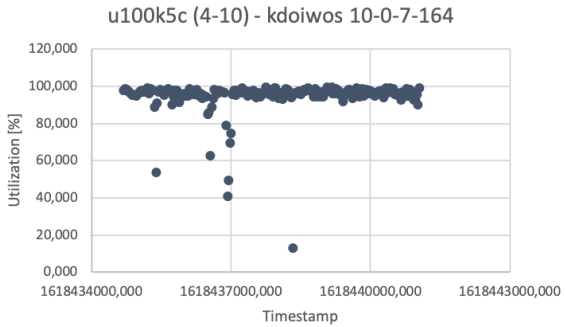


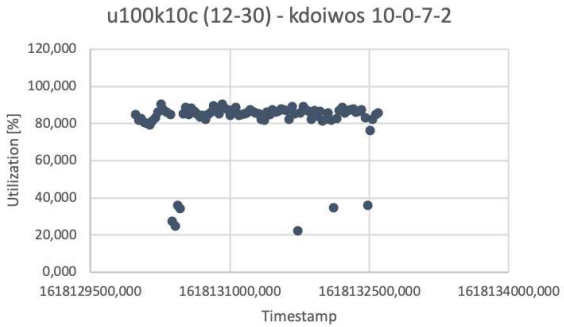


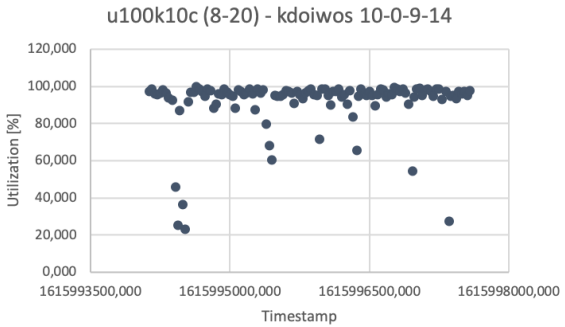


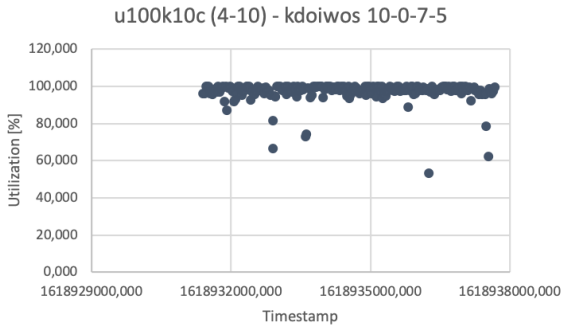






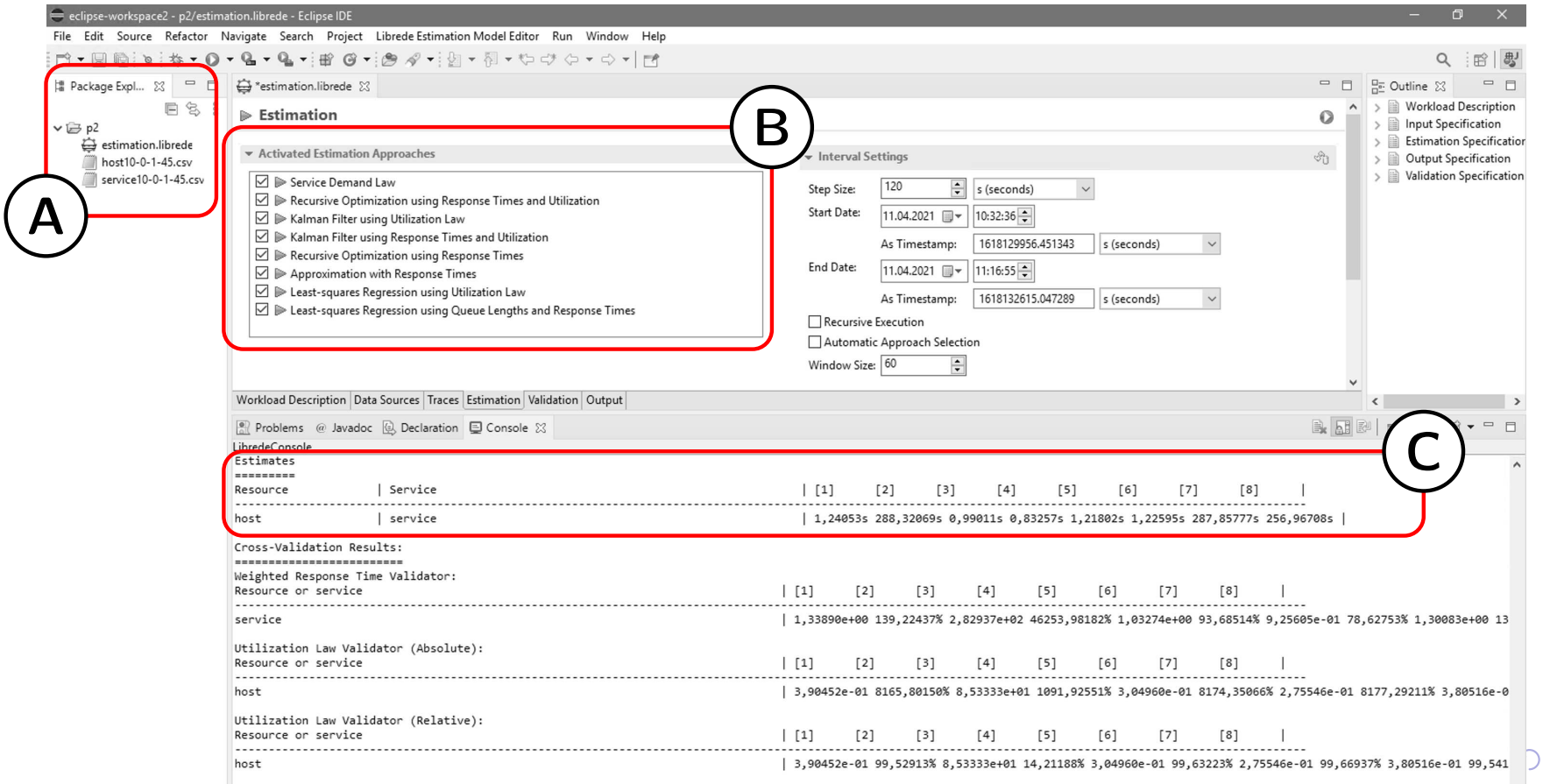






Estimation Approaches

(1)Approximation with Response Times;(2)Kalman Filter using Response Times and Utilization;(3)Kalman Filter using Utilization Law;(4)Least-squares Regression using Queue Lengths and Response Times;(5)Least-squares Regression using Utilization Law;(6)Recursive Optimization using Response Times;(7)Recursive Optimization using Response Times and Utilization;(8)Service Demand Law



Average Resource Demand

	Resource_demand [s]	1/*Resource_demand* [req/s]
12-30 $^{(a)}$	0,997588333	1,002417497
12-30_5 $^{(b)}$	0,727264667	1,37501524
12-30_10 $^{(b)}$	0,3652765	2,737652162
8-20 $^{(a)}$	0,905941667	1,10382383
8-20_5 $^{(b)}$	0,532029333	1,879595611
8-20_10 $^{(b)}$	0,457546	2,185572598
4-10 $^{(a)}$	0,849838333	1,176694391
4-10_5 $^{(b)}$	~~0,397753~~	~~2,514123086~~
4-10_10 $^{(b)}$	~~0,736487259~~	~~1,35779674~~

$^{(a)}$ One container.
$^{(b)}$ The single container.

Input Parameters of Simulations (Client and System)

Scenario	S1	S2	S3

Parameter	Sim. 1			Sim. 2			Sim. 3		
Number of servers [a]	12			8			4		
Model	QPN1	QPN5	QPN10	QPN1	QPN5	QPN10	QPN1	QPN5	QPN10
CL queueing place	90			90			90		
X_{CL} - [req/s]	6,242	4,849	3,826	6,732	5,064	2,975	6,366	3,152	1,644
CP place [b]	90			90			90		
X_{CO_i} - [req/s] [c]	1,002	1,375	2,737	1,103	1,879	2,185	0,164	0,317	0,610

[a] FCFS scheduling strategy.
[b] Connections for containers – Initial marking.
[c] i - number of containers (1, 5, 10).

Response Time

	12-30 $^{(a)}$	12-30_5$^{(b)}$	12-30_10$^{(b)}$
Simulation [s]	1,003483	1,222707	0,528284
Measured [s]	0,904	1,014	0,599
Error [%]	-11,00475664	-20,58254438	11,80567613

	8-20 $^{(a)}$	8-20_5$^{(b)}$	8-20_10$^{(b)}$
Simulation [s]	1,075831	0,832224	0,711255
Measured [s]	1,128	0,928	0,871
Error [%]	4,624911348	10,32068966	18,34041332

	4-10 $^{(a)}$	4-10_5$^{(b)}$	4-10_10$^{(b)}$
Simulation [s]	0,851	1,914936	2,031159
Measured [s]	0,804	1,455	0,846
Error [%]	-5,845771144	[illegible]	[illegible]

$^{(a)}$ One container.
$^{(b)}$ The single container.

Generatory zdarzeń

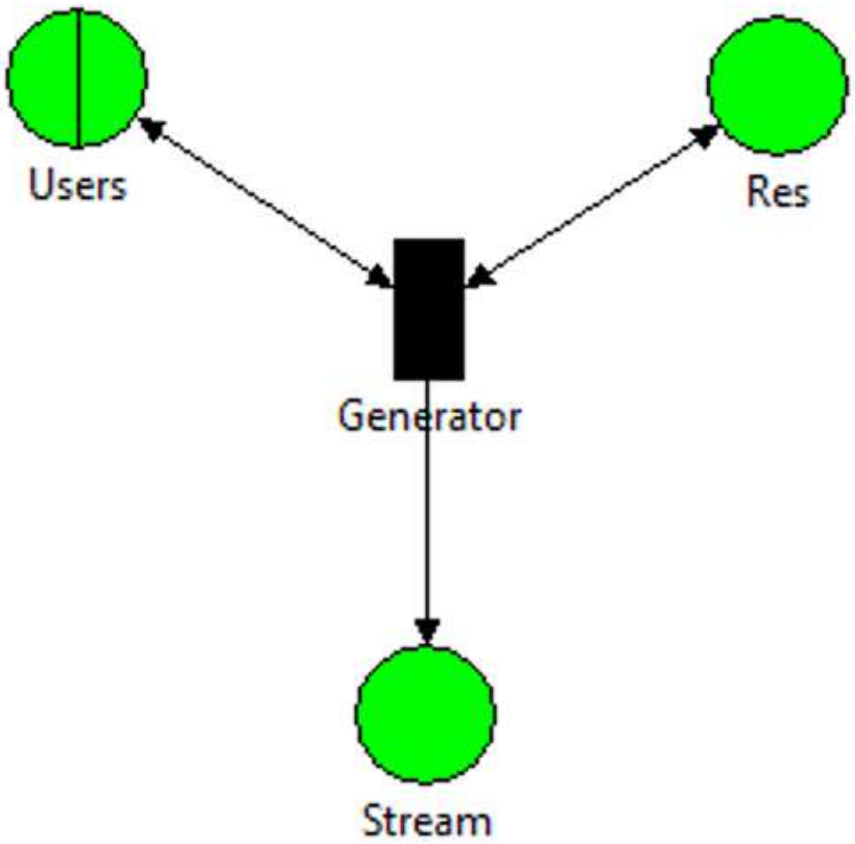
> "The best way to predict the future is to invent it."
> — Alan Kay

- Wprowadzenie
- Podstawowe modele generatorów HTCPN
- Podstawowe modele generatorów QPN
- Klasyfikacja modeli generatorów
- Złożone modele generatorów

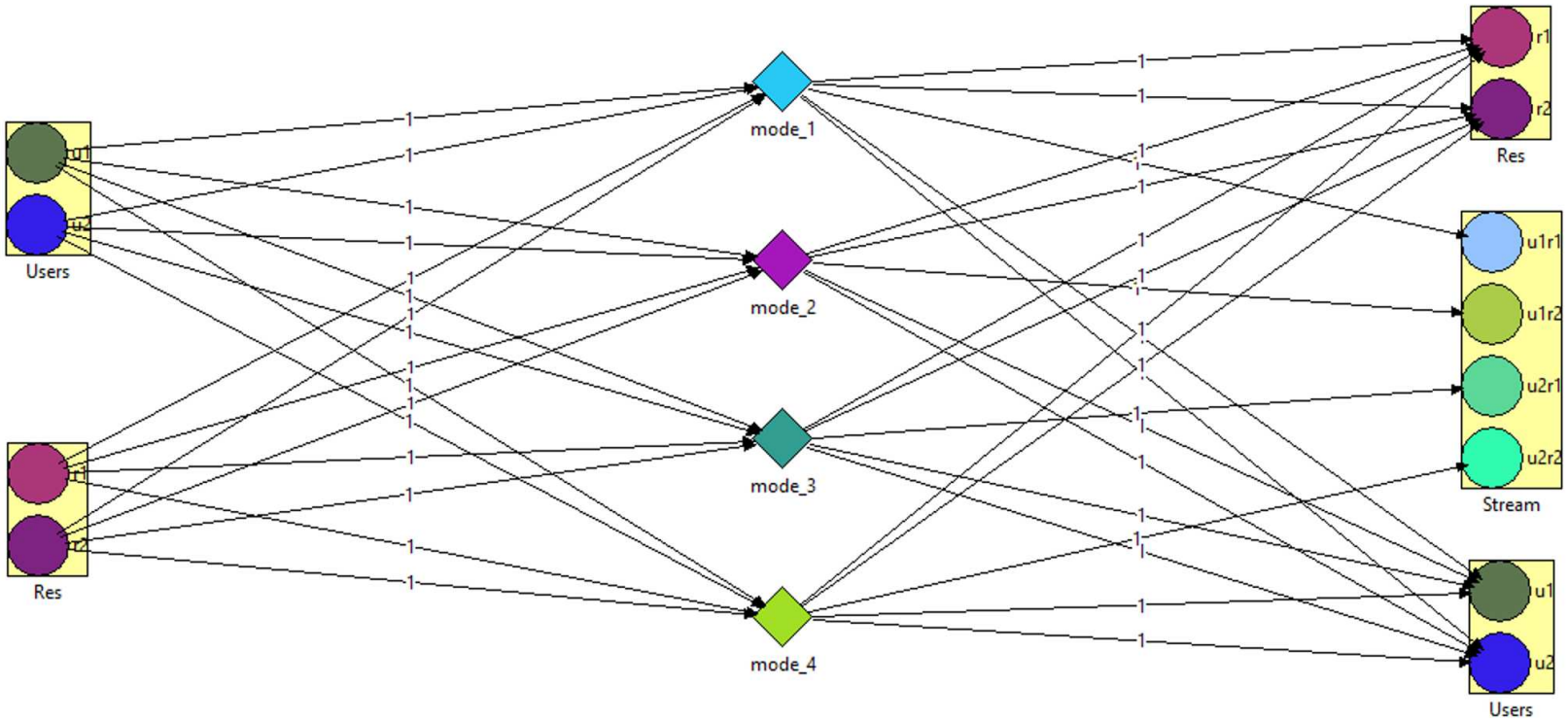
Rak, T.; Rzońca, D. Recommendations for Using QPN Formalism for Preparation of Incoming Request Stream Generator in Modeled System. Appl. Sci. 2021, 11, 11532. https://doi.org/10.3390/app112311532

Bożek, A.; Rak, T.; Rzońca, D. Timed Colored Petri Net-Based Event Generators for Web Systems Simulation. Appl. Sci. 2022, 12, 12385. https://doi.org/10.3390/app122312385

Ograniczenia w modelach QPN



Incidence Function



Modes

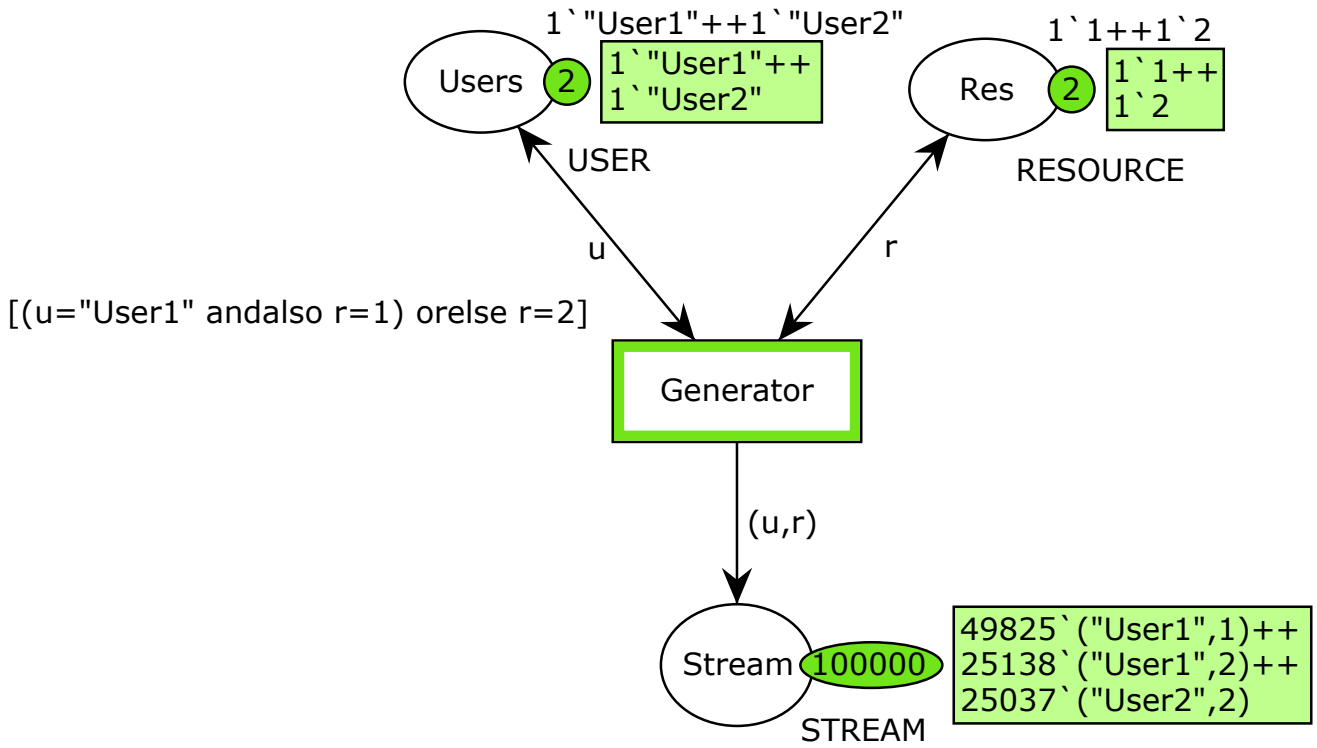
Name	Real Color	Firing Weight
mode_1		1
mode_2		2
mode_3		1
mode_4		3

Wyniki symulacji – Generator strumieniowy (Kolory tokenów)

Tryby	Wagi w	Kolory tokenów	Średnia populacja tokenów dla *Stream*
$mode_1$	1.0	$u1r1$	9,978.882
$mode_2$	2.0	$u1r2$	19,827.981
$mode_3$	3.0	$u2r1$	29,978.301
$mode_4$	4.0	$u2r2$	39,679.93
$mode_1$	1.0	$u1r1$	14,195.611
$mode_2$	2.0	$u1r2$	28,625.108
$mode_3$	1.0	$u2r1$	14,308.058
$mode_4$	3.0	$u2r2$	42,688.806
$mode_1$	1.0	$u1r1$	24,931.845
$mode_2$	1.0	$u1r2$	25,170.6
$mode_3$	0.0	$u2r1$	0.0
$mode_4$	2.0	$u2r2$	50,286.744

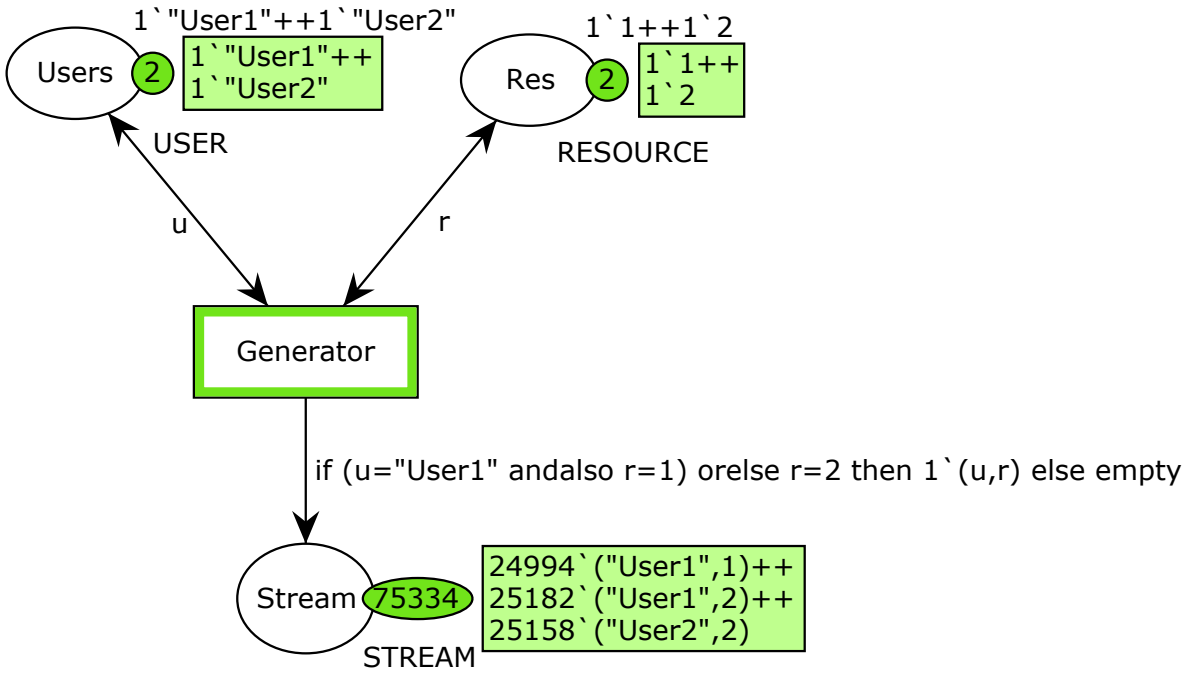
(...)

Wyniki symulacji – Generator strumieniowy 1 (Dozór)



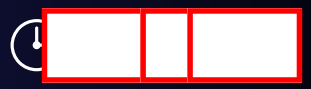
- Przejście zostało odpalone 100 tyś. razy
- Jedna z wygenerowanych par występuje częściej niż inne

Wyniki symulacji – Generator strumieniowy 2 (Łuk)



- Przejście zostało odpalone 100 tyś. razy, ale tylko ∼75 tyś. tokenów zostało wygenerowanych
- Rozkład każdej pary jest równy

Analiza logów



- Architektura systemu
- Klient automatyczny
- Metody analizy
- Wstępne wyniki analiz
- Kolejne prace

"If you don't log it, it never happened."

Rak, T.; Żyła, R. Using Data Mining Techniques for Detecting Dependencies in the Outcoming Data of a Web-Based System. Appl. Sci. 2022, 12, 6115, https://doi.org/10.3390/app12126115

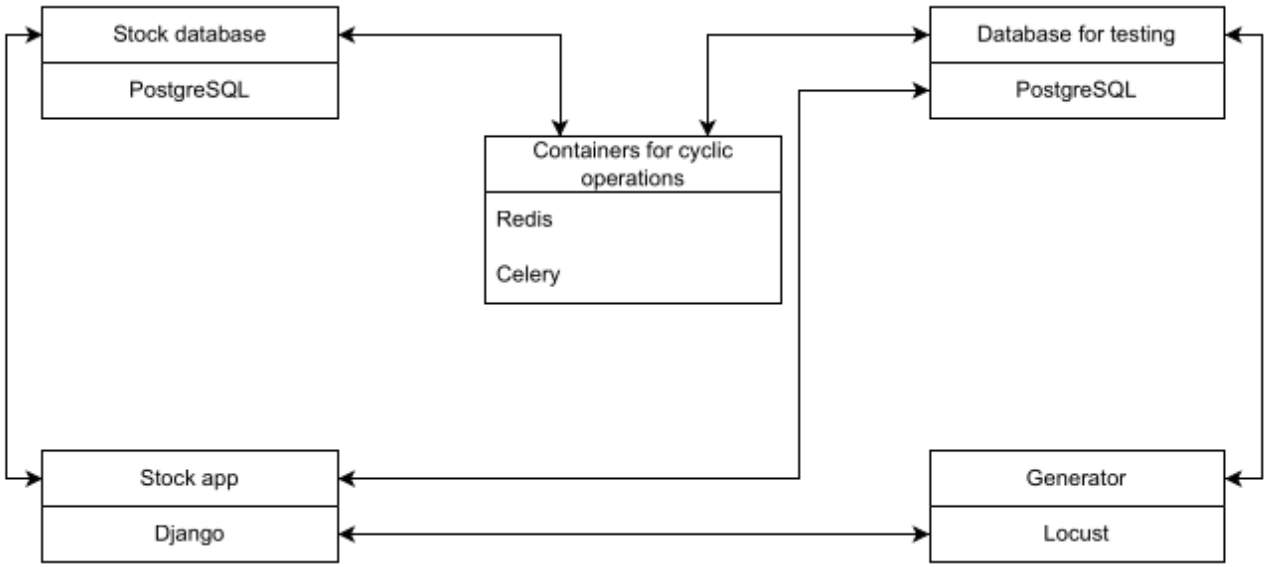
Borowiec, M.; Rak, T. Advanced Examination of User Behavior Recognition via Log Dataset Analysis of Web Applications Using Data Mining Techniques. Electronics 2023, 12, 4408, https://doi.org/10.3390/electronics12214408

Opis architektury

- Kontenerowa aplikacja składająca się z wielu usług
- Interfejs API obsługujący przychodzące żądania HTTP
- W przypadku zadań trwających dłużej (np. dopasowywanie zleceń kupna/sprzedaży lub okresowe aktualizacje) interfejs API umieszcza zadania w kolejce - Redis
- Dedykowane procesy - Celery - pobierają zadania i wykonują je asynchronicznie

Drabek, J.; Rak, T. Performance Monitoring and Analysis of a Containerized Stock Trading Application under Load (planowane)
Krogulski, P.; Rak, T. A Case Study on Virtual HPC Container Clusters and ML Application (planowane)

Architektura systemu

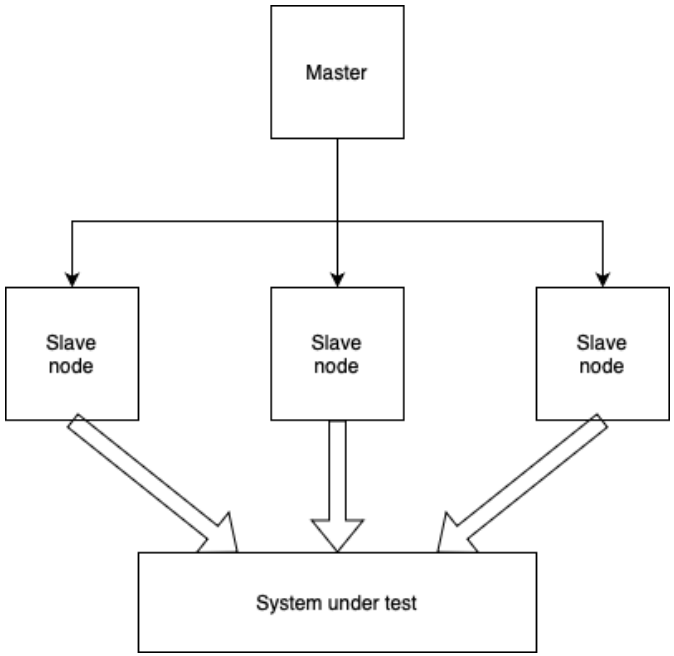


9

[9] Containerized architecture of the stock exchange system, including the Django web application (port 8000), PostgreSQL database (port 5432), Redis broker (port 6379), and Celery worker. An Nginx proxy (port 80) routes external traffic to the web application.

Klient automatyczny

- Locust symuluje równoczesnych użytkowników i generuje obciążenie systemu (wirtualni użytkownikownicy jako lekkie wątki Pythona)
- Każdy symulowany użytkownik reazlizuje zdefiniowany scenariusz działań
- Logi (zużycie procesora, zużycie pamięci oraz czasy odpowiedzi endpoint'ów, aplikacji i bazy danych)
- Czas trwania testu - 1h



Klasy klientów

- WebsiteActiveUser
- WebsiteReadOnlyUser
- WebsiteActiveUserWithMarketAnalyze

Klasyfikacja grup klientów

- Identyfikacja wzorców zachowań użytkowników na podstawie metryk wydajności
- Scenariusz testu obciążenia postrzegać się jako klasę
- Użyte algorytmy klasyfikacji: Xgb, ExtraTreesClassifier, DecisionTreeClassifier, KNeighborsClassifier, RandomForestClassifier, LinearSVC, MLPClassifier, ExtraTreeClassifier, BernoulliNB, GaussianNB, NearestCentroid, RadiusNeighborsClassifier, LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis

Key Results

Tabela: User class classification accuracy using selected algorithms.

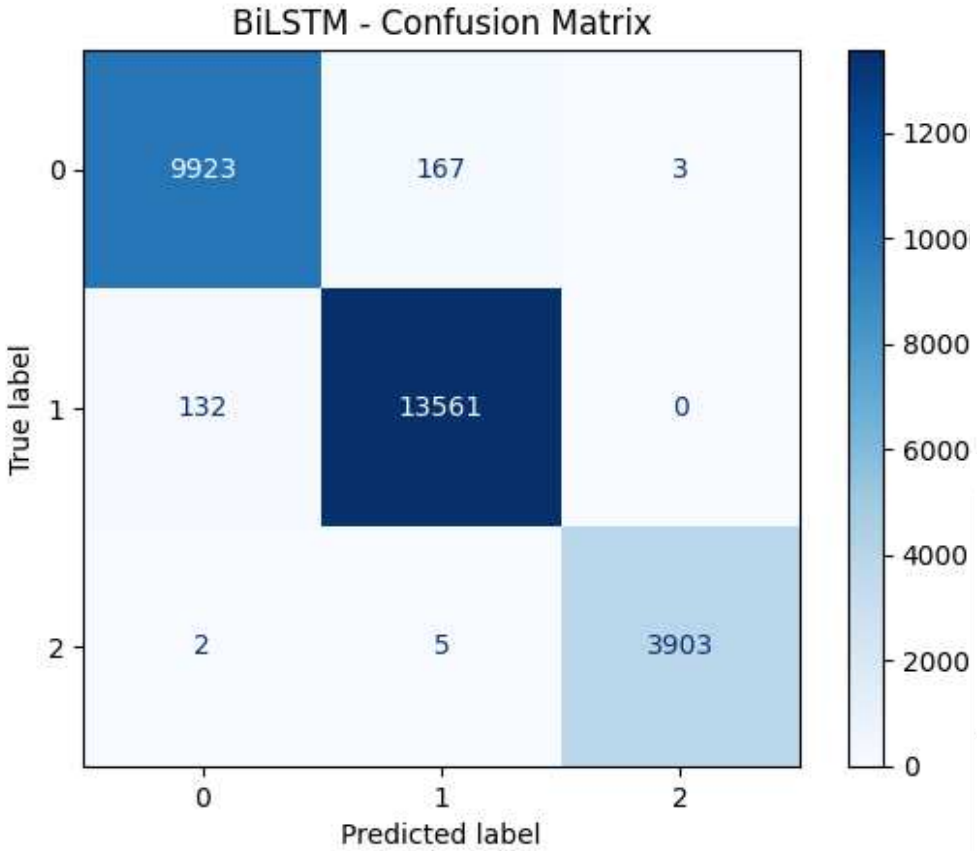
Algorithm	Test 21	Test 22	Test 23	Test 24
XGB	74%	**79%**	72%	73%
ExtraTreesClassifier	73%	**79%**	72%	74%
DecisionTreeClassifier	74%	**79%**	72%	73%
KNeighborsClassifier	72%	**76%**	68%	71%
RandomForestClassifier	73%	**79%**	72%	74%
LinearSVC	62%	**73%**	63%	68%
MLPClassifier	74%	**79%**	72%	74%

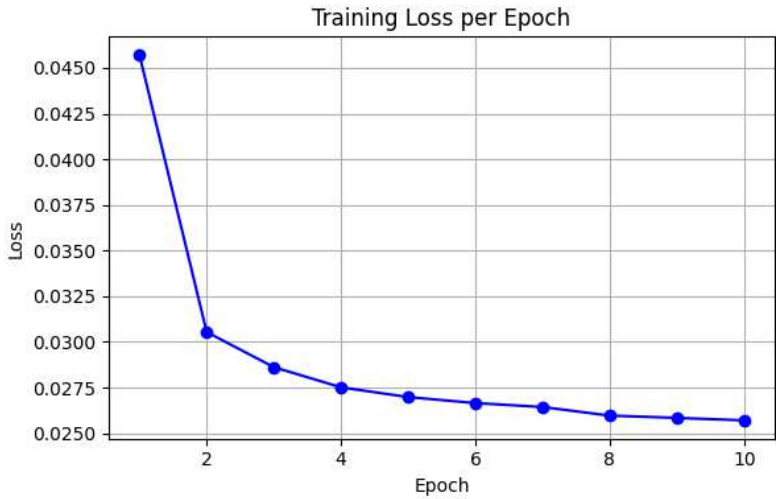
Algorithm	Test 25	Test 26	Test 27	Test 28
XGB	74%	72%	72%	73%
ExtraTreesClassifier	74%	72%	73%	73%
DecisionTreeClassifier	74%	72%	72%	73%
KNeighborsClassifier	71%	69%	69%	67%
RandomForestClassifier	74%	72%	73%	73%
LinearSVC	68%	65%	66%	67%
MLPClassifier	74%	72%	73%	73%

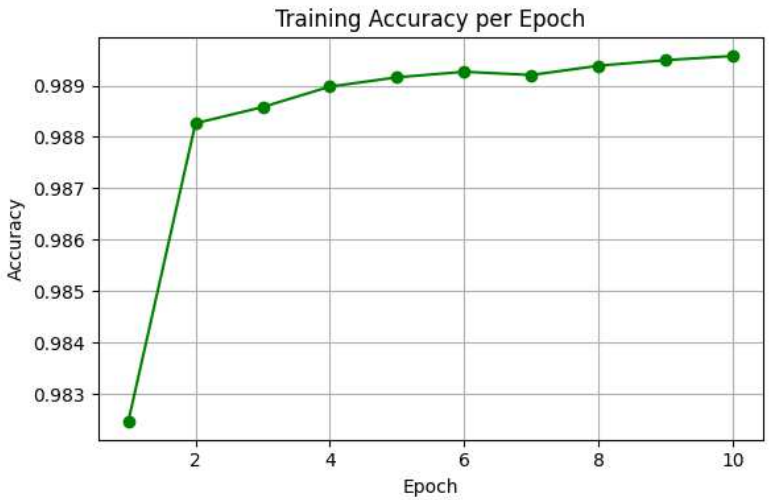
Future Work

- **Graph Databases (np. Neo4j)** — poprzez reprezentowanie zdarzeń (żądań, transakcji, użytkowników) jako węzłów, a ich interakcji lub sekwencji jako krawędzi w celu identyfikacji wzorców lub anomalii w zachowaniu systemu
- **Deep Learning (np. LSTM/GRU)** - może uczyć się „normalnych" wzorców wydajności i pomagać w oznaczaniu nietypowych zachowań systemu
- **Large Language Models** - może klasyfikować użytkowników lub przewidywać ich następne działania z wyższym poziomem zrozumienia, biorąc pod uwagę semantyczne wzorce zachowań

BiLSTM


BiLSTM - Confusion Matrix
True label
Predicted label
0
1
2
9923
167
3
132
13561
0
2
5
3903
1200
1000
8000
6000
4000
2000
0


Training Loss per Epoch
Loss
Epoch
0.0450
0.0425
0.0400
0.0375
0.0350
0.0325
0.0300
0.0275
0.0250
2
4
6
8
10


Training Accuracy per Epoch
Accuracy
Epoch
0.989
0.988
0.987
0.986
0.985
0.984
0.983
2
4
6
8
10

Modelowanie i analiza systemów webowych

Dziękuję za uwagę!

即時對每次點擊進行分類明天的高峰將成為今天的平穩運行[a]

[a] "Classify every click in real time, and tomorrow's spike becomes today's smooth ride."