BDI Core trust Protocol Performance Baseline Test

basic data infrastructure

Topsector Logistiek

Colophon

BDI Core trust Protocol Performance Baseline Test

Authors
George Zachiotis
Laurens Paardekam
Huibert Alblas

Januari 2024

basic data infrastructure

Topsector Logistiek

Summary

The Basic Data Infrastructure (BDI) framework is an infrastructure framework for controlled data sharing, supporting automated advanced information logistics within next-generation data eco-systems. Departing from traditional messaging paradigms, the BDI shifts towards event-driven information collection at the source, fostering efficient and secure communication through proven publish-and-subscribe architectures.

The BDI extends the concept of Data Spaces into the physical domain, aligning with established and innovative practices in data sharing integral to operations and supply chains. A comprehensive system of (legal) agreements is imperative to automate and control mutual communication effectively and trustworthy. This system, encapsulated within the BDI, balances control (determining who can access or utilize specific data) and efficiency (enabling automated processing).
Key features include:

- Digital Trust
  - Control over data through 'collecting data at the source';
  - Certainty about digital identities;
  - High-security standards;
  - Contractual basis for autonomous system-to-system communication;
  - Agreements on Language and Concepts.

- Semantics for automatic processing
  - Mutual understanding of data formats and models for effective communication.

- Event-Based Alerts
  - Automatic signaling for timely and efficient operations;
  - Immediate awareness of relevant changes in reality.

The core digital trust protocol is based upon the iSHARE framework. In this report the term Distributed Information Access Control (DIAC) is used for the core protocol.

In order to be able to establish a baseline for performance and resources need for deployment, a proof of concept has been developed and tested in a Cloud Based server.

Based on the performed tests it seems that the components of the system included in the cost estimation can handle up to 1000 concurrent users in the tested scenario. When the rate limiter in the authorization registry is increased, it could be expected that the entire system can handle 1000 concurrent users. With the current configuration a 100 concurrent users can be served. Which would already be plenty for most of the usecases currently discussed.

The cost of the cloud server as configured is around Euro 3,08 per day, which is not prohibitive.

Contents

1 Introduction

1.1 BDI

The Basic Data Infrastructure (BDI) framework is an infrastructure framework for controlled data sharing, supporting automated advanced information logistics within next-generation data eco-systems. Departing from traditional messaging paradigms, the BDI shifts towards event-driven information collection at the source, fostering efficient and secure communication through proven publish-and-subscribe architectures.

The BDI extends the concept of Data Spaces into the physical domain, aligning with established and innovative practices in data sharing integral to operations and supply chains. A comprehensive system of (legal) agreements is imperative to automate and control mutual communication effectively and trustworthy. This system, encapsulated within the BDI, balances control (determining who can access or utilize specific data) and efficiency (enabling automated processing). Key features include:

- Digital Trust
  - Control over data through 'collecting data at the source';
  - Certainty about digital identities;
  - High-security standards;
  - Contractual basis for autonomous system-to-system communication;
  - Agreements on Language and Concepts.

- Semantics for automatic processing
  - Mutual understanding of data formats and models for effective communication.

- Event-Based Alerts
  - Automatic signaling for timely and efficient operations;
  - Immediate awareness of relevant changes in reality.

The core digital trust protocol is based upon the iSHARE framework. In this report the term Distributed Information Access Control (DIAC) is used for the core protocol.

In order to be able to establish a baseline for performance and resources need for deployment, a proof of concept has been developed and tested in a Cloud Based server.

This report aims to present and comment on the entire system's Web Load Performance, Recurring Costs and Failure Effect Assessment.

2 Web Load Performance Testing

2.1 System description

The DIAC system architecture is presented in Figure 1. It consists primarily of a Web API serving HTTPS requests to multiple Web Clients. For a user login request, the DIAC system may retrieve an authentication token from the BDI Authorization Register server. For other kind of requests, a token may be validated in the same component. The Triple Store serves as the interface to the Legacy Database. At fixed, regular time intervals data is synchronized between the two.

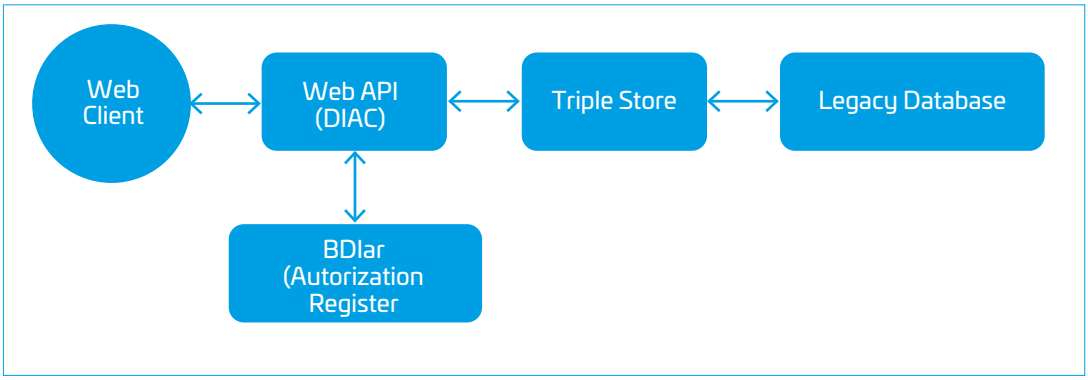


Figure 1
DIAC system architecture

2.2 Testing tools

JMeter was selected a performance testing tool, as it is a standalone, cross-platform, has a large community, support different communication protocols, has shallow learning curve, is open source and can be run in CLI mode. JMeter allows to configure powerful Web-Load tests for Web APIs and Network Interfaces. The client was a clean image of windows 11, hosted in Azure, to prevent external factors from interfering with the test.

2.3 Testing infrastructure

In a federated environment the tools and underlying infrastructure can differ, as they are organized by different parties. The underlying infrastructure is listed in this paragraph as it provides a reference to the performance measurements. SaaS products often include a rate limiter, which limits the number of incoming requests to ensure a baseline performance level for every client. Rate limiters are an artificial bottleneck, the limit can often be increased after a negotiation with the SaaS provider. Rate limiters can influence the response time of other components in a federated environment.

COMPONENT	INFRASTRUCTURE	RATE LIMITER ACTIVE DURING TEST
TRIPLE STORE	SaaS product	Yes
WEB API	Docker container on a ubuntu VM Azure standard D2s v3 - 2 vcpu's - 8 GiB memory	No
BDI AR	SaaS product	Yes
WEB CLIENT	Windows 10 virtual machine Azure standard D2s v3 - 2 vcpu's - 8 GiB memory	No

2.4 Testing strategy

In order to retrieve metrics on the system performance, the Web API's of the various components in the system have been load-stressed by generating high volumes of Web traffic towards it. This has been achieved by using a number of simulated, gradually increased number of users who concurrently executed a use-case scenario. Overall, four use-case scenarios have been examined:

i. Obtaining the iSHARE authentication token, and requesting data:
   - Fetching an iSHARE token (URL: https://diac- tsl.westeurope.cloudapp.azure.com/connect/<...>)
   - Fetching Data (URL: https://diac-tsl.westeurope.cloudapp.azure.com/Diac/GetData/<...>)

ii. Directly request data with hardcoded authentication key:
   - Fetching Data (URL: https://diac-tsl.westeurope.cloudapp.azure.com/Diac/GetData/<...>)

iii. Directly requesting data from the backend triple store database:
   - Fetching Data (URL: https://api.logistiek.triply.cc/datasets/TopsectorLogistiek/<...>)

iv. Directly obtaining authorization data from the authorization registry:
   - Fetching an iSHARE token (URL: https://api.poort8.nl/ar-preview/ishare/connect/token /<...>)
   - Fetching authorization data (URL: https://api.poort8.nl/ar-preview /<...>)

The testing process has been completed in a number of successive steps, starting with a single user executing a single use-case scenario. The concurrent use of the system was contained within a fixed-size time span of 300 seconds. During the step, a number of HTTPS requests (usually in the order of thousands) were executed. Upon completion, the collected data were processed to provide industry-standard Web-Endpoints performance metrics. In the following step, the number of users was increased and the tests repeated. The study for each use-case scenario stopped only when a large degradation of the system's performance had been observed.

The number of concurrent users using the system has been defined as the independent variable. The Average Load Time for each of the HTTPS requests has been defined as the dependent variable. The results should be able to provide an identification of the mapping between those two quantities.

2.5 Testing process pre-conditions

The system under testing had been built and deployed in the environment in which its performance was meant to be evaluated. A clean VM, acting as client, was present in the same (sub)network as the deployment environment. Before any of the tests have been executed, care had been taken to exit the DIAC system's server from a potential sleep mode status. The performance testing did not concern any aspect of the system's business logic. Thus, the received HTTPS responses have been only checked about their success status.

2.6 Testing process post-conditions

The conducted performance tests aim to identify a mapping between the independent variable i.e. the number of concurrent users concurrent executing a use-case scenario and the dependent variable i.e. the Average Load Time of the HTTPS requests included in the aforementioned scenario.

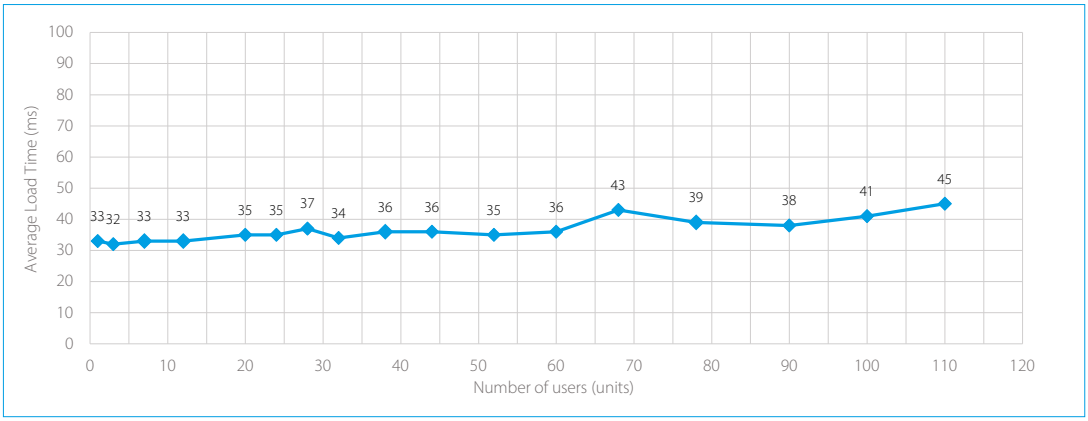
2.7 First use-case scenario

This use-case scenario tests the entire sequence of calls required for the client to retrieve data. First an iSHARE access token is being retrieved. Thereafter a second endpoint is called to retrieve the data.

Call 1: Obtaining the iSHARE authentication token

Figure 2 shows a stable load time of around 34 milliseconds up to 60 concurrent users. This is a more or less expected performance since this kind of requests (authentication token fetching) are usually lightweight in terms of resources usage on the server side. A slight increase of load time is observed in the region of over 60 concurrent users and at the max, 133 milliseconds was observed in the test with 310 users. The increase in load time, at around 110 users, of this first request is most likely caused by the increase in load time of the second request (Data Fetching).

Figure 2
First use-case scenario-Complete Login

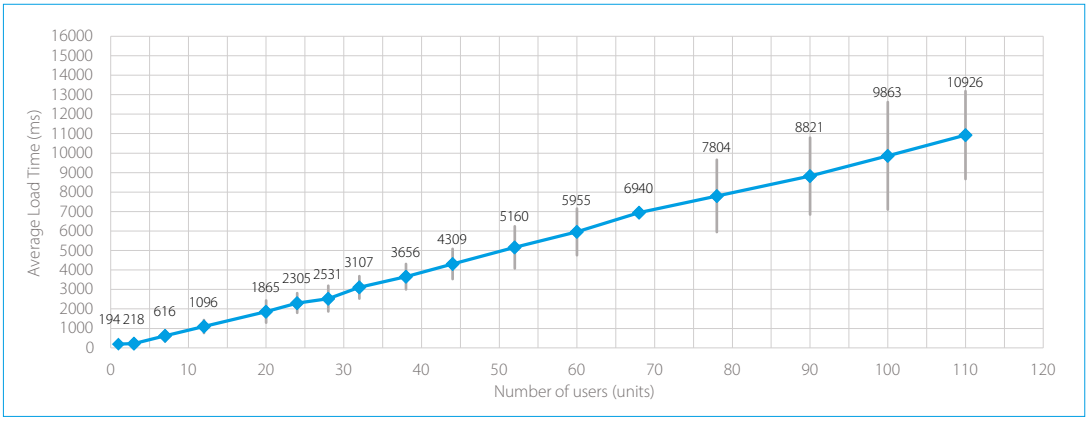


The test results are also presented in Appendix I. The markers in Figure 2 are mapping the Number of Users (x-axis) to the Average Load Time (y-axis) for the Complete Login Request. The point-to-point graph line gives an estimation about the unsampled values. From the graph, three values are missing for the upper extreme: 150 users with average load time of 54 milliseconds, 220 users with average load time of 45 milliseconds and 310 users with 133 milliseconds[1].

Call 2: Data Fetching

When testing with up to 3 users, a Load Time of around 200 milliseconds has been measured. This number constitutes a non-standardized, thought widely accepted, very good response time for a performing Web API. With the users increasing to 12, load times of around 1 seconds have been measured. At that point, performance may be considered within tolerable limits. A horizontal or vertical scaling of the system is strongly recommended when users are intended to be more than 100.

Figure 3
First use-case scenario graph-Data Fetching Request



1 The cause of these anomalies is the rate limiting on the BDI AR as used.

The point-to-point graph line intends to give an estimation about the unsampled values. The vertical lines in every marker, twice the Standard Deviation in length, is meant to give a qualitative measure of the Load Time value dispersion. For readability of the graph, three values[2] are not plotted in the upper extreme: 150 users with average load time of 16,779 seconds, 220 users with average load time of 26,884 seconds and 310 users with average load time of 52,685 seconds.

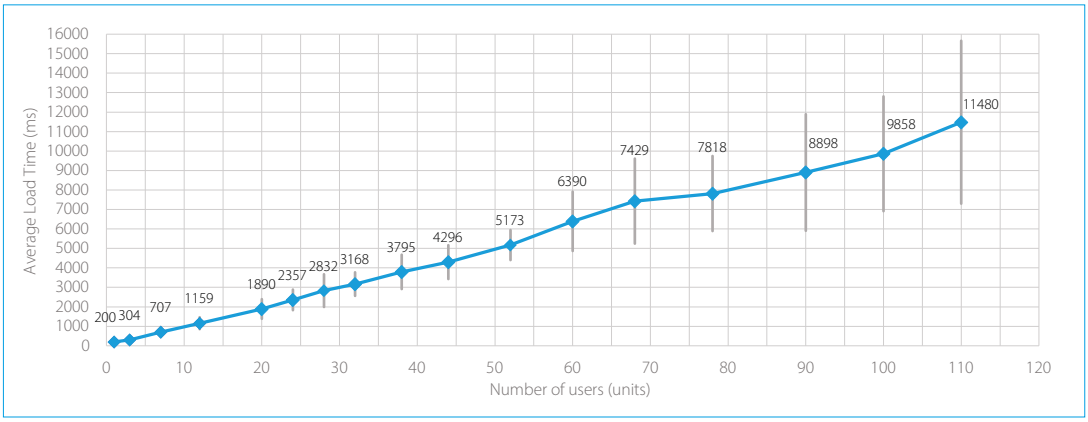
2.8 Second use-case scenario

This scenario directly retrieves data from the DIAC API with a single call, instead of two calls in the previous use-case. This is achieved by using a hardcoded authentication key. The results are comparable to the results of the second call of the first use-case.

Call 1: Data Fetching

The test results[3] are given in Appendix III and also presented in the graph of Figure 4. The markers are mapping the Number of Users (x-axis) to the Average Load Time (y-axis) for the Data Fetching Request.

Figure 4
Second use-case scenario-Data Fetching Request



The point-to-point graph line intends to give an estimation about the unsampled values. The vertical lines in every marker -twice the Standard Deviation in length- is meant to give a qualitative measure of the Load Time value dispersion. For readability of the graph, three values[4] are not plotted in the upper extreme: 150 users with average load time of 16,589 seconds, 220 users with average load time of 25,060 seconds and 310 users with average load time of 36,027 seconds.

For the starting point of the study (testing with one user) and up to the testing with 3 users, a Load Time of around 200 milliseconds has been measured. This number constitutes a non-standardized -thought widely accepted- very good response time for a performing Web API. With the users increasing to 12, load times of around 1 seconds have been measured. At that point, performance may be considered within tolerable limits. A horizontal or vertical server scaling is strongly recommended when users are intended to be more than 100.

2 The cause of these anomalies is the rate limiting on the BDI AR as used.
3 It is important to clarify that there is no evidence that the samples of this or any other HTTPS request in this study are subject to Normal Distribution. While the exact distribution is considered out of context of the study, the Deviation size as included in the graphs above may give a very good intuitive measure of the variability of the request's Load Time.
4 The cause of these anomalies is the rate limiting on the BDI AR as used.

2.9 Third use-case scenario

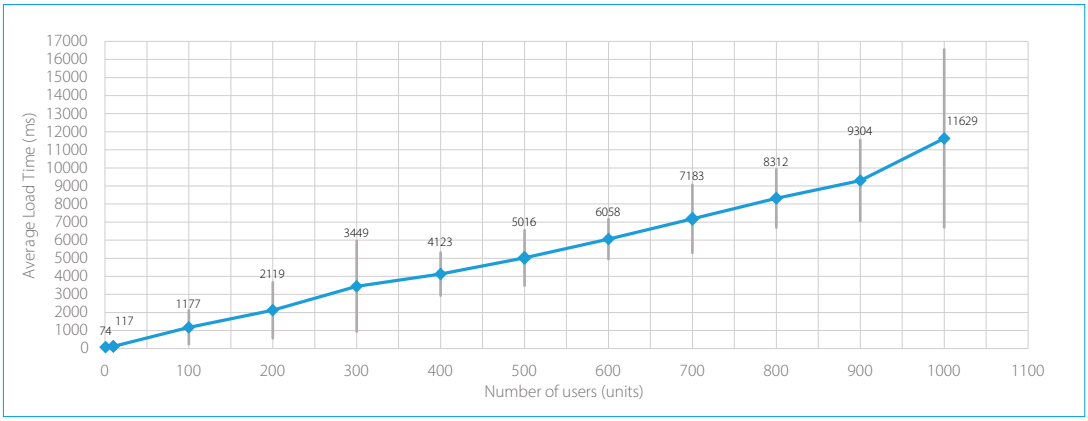
The third use-case retrieves data directly from the Triple Store, without contacting the DIAC API & Authorization registry. The response times are significantly better that when requesting the same data via the DIAC API & Authorization registry.

Call 1: Data Fetching

The test results are given in Appendix II and also presented in the graph of Figure 5. The markers are mapping the Number of Users (x-axis) to the Average Load Time (y-axis) for the Data Posting Request. The point-to-point graph line intends to give an estimation about the unsampled values. The vertical lines in every marker -twice the Standard Deviation in length- is meant to give a qualitative measure of the Load Time value dispersion.

For the starting point of the study (testing with one user) and up to the testing with 40 users, a Load Time of around 200 milliseconds has been measured. This number constitutes a non-standardized-thought widely accepted- very good response time for a performing Web API. With the users increasing to 90, load times of around 1 seconds have been measured. At that point, performance may be considered within tolerable limits. A horizontal or vertical server scaling is strongly recommended when users are intended to be more than 960.

Figure 5
Third use-case scenario-Data Posting Request



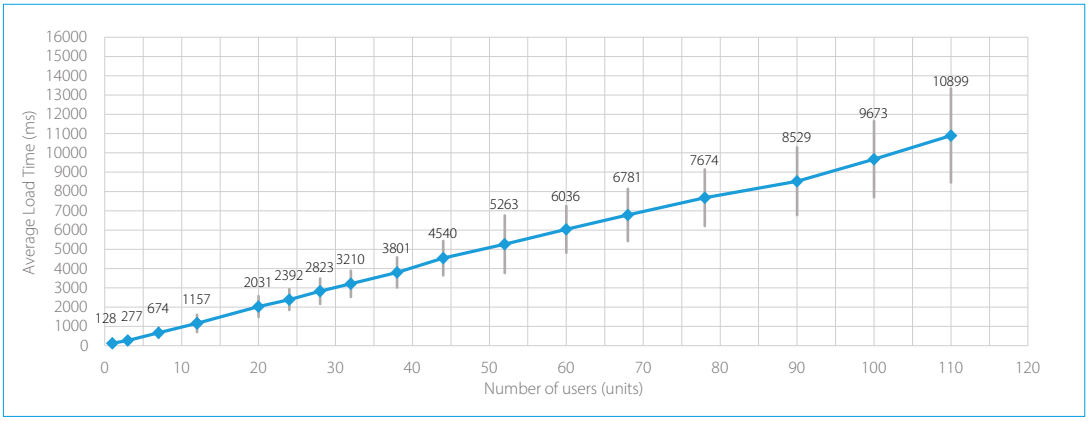
2.10 Fourth use-case scenario

The fourth and last use-case contacts the Authorization registry directly, without contacting the DIAC API. The response times are comparable to those when contacting the Authorization registry via the DIAC API. A horizontal or vertical server scaling is strongly recommended when users are intended to be more than 110.

Call 1: Obtaining the iSHARE authentication token

The test results are given in Appendix IV and also presented in the graph of Figure 6. The markers are mapping the Number of Users (x-axis) to the Average Load Time (y-axis) for the Complete Login Request.

Figure 6
Fourth use-case scenario-Complete login



The point-to-point graph line intends to give an estimation about the unsampled values. For readability of the graph, three values[5] are not plotted in the upper extreme: 150 users with average load time of 12528 milliseconds, 220 users with average load time of 136720 milliseconds and 310 users with average load time of 129896 milliseconds.

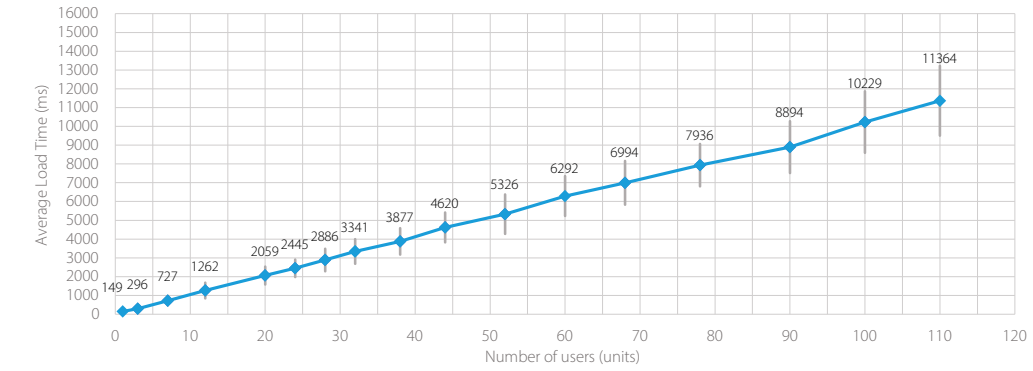
For the starting point of the study (testing with one user) and up to the testing with 2 users, a Load Time of around 200 milliseconds has been measured. This number constitutes a non-standardized -thought widely accepted- very good response time for a performing Web API. With the users increasing to 10, load times of around 1 second have been measured. At that point, performance may be considered within tolerable limits. A horizontal or vertical server scaling is strongly recommended when users are intended to be more than 110.

5 The cause of these anomalies is the rate limiting on the BDI AR as used.

Call 2: Fetching authorization data

The test results are given in Appendix IV and also presented in the graph of Figure 7. The markers are mapping the Number of Users (x-axis) to the Average Load Time (y-axis) for the Data Posting Request.

Figure 7
Fourth use-case scenario-Data Posting Request



The point-to-point graph line intends to give an estimation about the unsampled values. The vertical lines in every marker -twice the Standard Deviation in length- is meant to give a qualitative measure of the Load Time value dispersion.

For readability of the graph, three values[6] are not plotted in the upper extreme: 150 users with average load time of 12,915 seconds, 220 users with average load time of 229,982 seconds and 310 users with average load time of 231,007 seconds.

For the starting point of the study (testing with one user) and up to the testing with 2 users, a Load Time of around 200 milliseconds has been measured. This number constitutes a non-standardized -thought widely accepted- very good response time for a performing Web API. With the users increasing to 10, load times of around 1 seconds have been measured. At that point, performance may be considered within tolerable limits. A horizontal or vertical server scaling is strongly recommended when users are intended to be more than 100.

2.11 Summary

Based on the performed tests it seems that the components of the system included in the cost estimation can handle up to 1000 concurrent users in the tested scenario. When the rate limiter in the authorization registry is increased, it could be expected that the entire system can handle 1000 concurrent users. With the current configuration a 100 concurrent users can be served. Which would already be plenty for most of the usecases currently discussed within Topsector Logistics. The rate limiting on the AR used is the explanation for the data points with excessive response times.

6 The cause of these anomalies is the rate limiting on the BDI AR as used.

3 Recurring Costs Estimations

3.1 Recurring costs of operation for DIAC server

The cost analysis has been conducted in reference to the Cloud Computing Platform (Microsoft Azure) used to host the current DIAC server Virtual Machine. The pricing data for this installation can only be seen as an estimation, as specific implementations may vary. The costs are presented in Table 1. They are concerning the system's components using the DIAC server. This does not include the BDI Authorization Register.

Table 1
Repeating costs per resource per day - obtained on 01-09-2023

Service Name	Meter	Cost	Granularity
Storage	P4 LRS Disk	€ 0,17	Daily
Bandwidth	Inter Continent Data Transfer Out - NAM or EU To Any	€ 0,00	Daily
Bandwidth	Intra Continent Data Transfer Out	€ 0,03	Daily
Bandwidth	Standard Data Transfer Out - Free	€ 0,00	Daily
Virtual Machines	D2 v3/D2s v3	€ 2,62	Daily
Container Registry	Basic Registry Unit	€ 0,15	Daily
Virtual Network	Standard IPv4 Static Public IP	€ 0,11	Daily
SUM		**€ 3,08**	**Daily**

4 Failure Effects Assessment

4.1 Failure Points and effects

Possible failing points along with their consequences are presented in Table 2. Their granularity is on the system component level as shown in Figure 1. All components can be marked as 'critical' for the system to function, except for the internal legacy database.

Table 2
Failing points and their effect on the system performance

Failing point	Consequence
Web Client fails	There should be no consequence for the system and the rest of the clients. The specific client will not be able to use the DIAC system.
Web API fails	No end-user should be able to use the DIAC System. The Triple Store will still be able to provide data synchronization.
Triple Store fails	End users are unable to fetch data.
BDI Authorization Register fails	End users should only be able to use the functionality of the system that requires no authentication.
Legacy Database fails	End users should be only be able to fetch outdated data.

Appendix

Service Level Agreements (SLA) considerations

Service level agreements are common between (IT) Service providers and their customers. This should not be any different in a federated environment. However, in a federated environment it is common that multiple providers, with multiple IT systems, rely on each other to fulfill a single data request by a single end user. In the situation that an organization sources multiple external IT providers itself, the effect of the SLA's on the availability of their service should be taken in account.

The content of an SLA can be made specific for every usecase, in this chapter the main items that should be in an SLA are discussed.

5.1 Purpose

A Service Level Agreement (SLA) is a contract between a service provider and its clients, meant to state commitment on the availability and continuity of a service. An SLA can also be used between a service provider and an external maintenance party, whose job it is to solve problems that can jeopardize the availability of the service.

5.2 Description & Scope

This describes the service or application for which the SLA will be made. What parts are in scope of the SLA, and what parts are out? The responsibilities of the service provider and third party will be described as well. Lastly the service hours will be described: will there be a 24h service or only service during office hours?

5.3 Responsiveness

This describes how quickly there will be acted on incidents. The responsiveness can change during the day: in some cases, it could be acceptable that the responsiveness is lower during off-peak hours.

5.4 Service level objectives

This states goals of how well the system or service should perform. These targets should be measurable. Some relevant examples for this DIAC system:

- Uptime/availability
  - The service can be fully offline because of updates or unplanned issues. How may minutes a year is the service allowed to be offline?

- Response time/latency
  - When the end user sends a request to the service, what would be the maximum duration for the service to respond?

- Data retention/backups
  - How are data backups handled on this service? If data loss occurs, how many hours of data would be lost when restoring a backup?

- Scaling of resources
  - When the service receives an unplanned amount of extra requests, causing the service to slow down, the service could decide to scale up extra resources. How fast should this up- and downscaling happen?

- Update policies
  - If updating the service causes breaking changes to clients, this should be planned. How far ahead would clients be notified and how often a year are those kinds of updates permitted? Would alternative interfaces be provided and for how long?

5.5 Monitoring

A traditional reactive approach with submitting tickets can be sufficient, but has a downside in the speed of the mitigation. An alternative reactive approach could be an automatic monitoring system that alerts when issues are measured. Lastly a predictive system can be used to predict in advance when the service might fail.

Depending on the type of service, this can be done based on bandwidth, cpu, memory usage and predictive usage of the service. Monitoring is especially important in the scenario where multiple IT systems, with different SLA's, work together to provide a single service. Critical time can be lost when trying to find the issues in the wrong IT system.

5.6 Penalties

This describes the consequences of failing to meet the SLA. This can be a financial consequence, but depending on the use case, other types are possible as well.

Appendix I

First use-case scenario performance tests results

Table 1
Authentication Request

Users (units)	Samples (units)	Av. Load Time (ms)	St. Deviation (ms)	Throughput (units/s)
1	1047	33	4.58	3.5
3	2880	32	4.59	9.6
7	2944	33	8.39	9.9
12	3026	33	14.72	10.1
20	3046	35	23.92	10.2
24	2978	35	25.79	10
28	3178	37	15.79	58.72
32	2993	34	17.23	10
38	3024	36	29.77	10.1
44	2977	36	25.53	10
52	2966	35	29.38	9.9
60	2978	36	35.47	9.8
68	2889	43	80.64	9.7
78	2917	39	61.17	9.8
90	3037	38	59.95	9.9
100	3005	41	52.93	10.1
110	2985	45	85.02	10
150	2689	54	233.26	9
220	2571	45	44.7	8.7
310	1773	133	318.26	6.1

Table 2
Data Fetching Request

Users (units)	Samples (units)	Av. Load Time (ms)	St. Deviation (ms)	Throughput (units/s)
1	1176	194	123.47	2.5
3	2880	218	146.79	9.6
7	2943	616	280.6	9.8
12	3026	1096	334.65	10
20	3044	1865	570.77	10.2
24	2978	2305	506.09	9.9
28	3177	2531	661.05	10.6
32	2993	3107	576.96	9.9
38	3024	3656	654.21	10
44	2976	4309	776.57	9.9
52	2966	5160	1082.73	9.8
60	2977	5955	1197.72	9.8
68	2888	6940	160.11	9.5
78	2943	7804	1856.72	9.7
90	3037	8821	1973.28	9.9
100	3005	9863	2759.83	9.7
110	2985	10926	2257.13	9.8
150	2688	16779	6399.47	8.6
220	2571	26884	10981.55	7.7
310	1773	52685	11640.55	5.5

Appendix II

Second use-case scenario performance tests results

Table 1
Data Fetching Request

Users (units)	Samples (units)	Av. Load Time (ms)	St. Deviation (ms)	Throughput (units/s)
1	1489	200	102.56	5
3	2953	304	163.1	9.8
7	2962	707	251.36	9.9
12	3103	1159	307.21	10.3
20	3176	1890	503.79	10.5
24	3058	2357	529.69	10.1
28	2971	2832	840.09	9.8
32	3039	3168	605.1	10
38	3013	3795	882.48	9.9
44	3091	4296	869.35	10.95
52	3032	5173	772.15	10
60	2836	6390	1523.23	9.3
68	2811	7429	2182.64	9
78	3018	7818	1927.4	9.8
90	3066	8898	299.44	10
100	3083	9858	2945.13	10
110	5835	11480	4181.3	7.4
150	2791	16589	6430.62	8.8
220	2707	25060	9940.14	8.4
310	4592	36027	17500.14	5

Appendix III

Third use-case scenario: Data Posting

Table 1
Data Posting Request

Users (units)	Samples (units)	Av. Load Time (ms)	St. Deviation (ms)	Throughput (units/s)
1	3987	74	14.5	13.3
10	27907	117	37.96	93
100	25479	1177	936.55	84.5
200	28390	2119	1546.4	93.7
300	26190	3449	2511.16	85.9
400	29223	4123	1192.53	95.7
500	30008	5016	1535.15	98.2
600	29883	6058	1110.42	97.5
700	29462	7183	1887	95.8
800	29107	8312	1614.88	94.4
900	29274	9304	2237.99	94.5
1000	26290	11629	4921.24	83

Appendix IV

Fourth use-case scenario: Complete Login

Table 1
Authentication Request

Users (units)	Samples (units)	Av. Load Time (ms)	St. Deviation (ms)	Throughput (units/s)
1	884	128	96.11	3
3	1422	277	161.19	4.8
7	1436	674	262.48	4.8
12	1451	1157	449.09	4.8
20	1449	2031	540.56	4.8
24	1468	2392	541.51	4.8
28	1459	2823	661.34	4.8
32	1458	3210	682.78	4.8
38	1482	3801	781.05	4.9
44	1444	4540	895.79	4.7
52	1476	5263	1493.24	4.9
60	1468	6036	1210.05	4.8
68	1493	6781	1357.29	4.8
78	1490	7674	1464.82	4.9
90	1568	8529	1757.08	5.1
100	1519	9673	1975.16	4.9
110	1515	10899	2431.57	4.9
150	1770	12528	6634.2	5.8
220	413	136720	95124.24	0.8
310	468	129896	101132	2.66

Table 2
Data Posting Request

Users (units)	Samples (units)	Av. Load Time (ms)	St. Deviation (ms)	Throughput (units/s)
1	884	149	92.27	3
3	1422	296	145.5	4.8
7	1436	727	249.35	4.8
12	1451	1262	417.87	4.8
20	1449	2059	475.94	4.8
24	1468	2445	465.92	4.8
28	1459	2886	599.49	4.8
32	1458	3341	656.28	4.8
38	1482	3877	703.33	4.8
44	1423	4620	795.18	4.7
52	1455	5326	1050.28	4.8
60	1450	6292	1061.57	4.7
68	1447	6994	1163.72	4.8
78	1469	7936	1128.26	4.8
90	1536	8894	1379.5	5
100	1488	10229	1637.56	4.8
110	1450	11364	1858.14	4.7
150	1713	12915	6600.13	5.7
220	220	229982	2377.55	0.5
310	310	231007	1278.78	0.48

Topsector Logistiek
Ezelsveldlaan 59
2611 RV Delft
+31 15 251 65 65
www.topsectorlogistiek.nl

Topsector Logistiek