SPECIAL

Scalable Policy-awarE Linked Data arChitecture for prIvacy, trAnsparency and compLiance

Deliverable D3.3

Backend Scalability and Robustness testing report V1

Document version: V1.0

SPECIAL DELIVERABLE

Name, title and organisation of the scientific representative of the project's coordinator:

Ms Jessica Michel t: +33 4 92 38 50 89 f: +33 4 92 38 78 22 e: jessica.michel@ercim.eu

GEIE ERCIM, 2004, route des Lucioles, Sophia Antipolis, 06410 Biot, France

Project website address: http://www.specialprivacy.eu/

Project	
Grant Agreement number	731601
Project acronym:	SPECIAL
Project title:	Scalable Policy-awarE Linked Data arChitecture for prIvacy, trAnsparency and compLiance
Funding Scheme:	Research & Innovation Action (RIA)
Date of latest version of DoW against which the assessment will be made:	17/10/2016
Document	
Period covered:	M01-M18
Deliverable number:	D3.3
Deliverable title	Backend Scalability and Robustness testing report V1
Contractual Date of Delivery:	30-06-2018
Actual Date of Delivery:	30-06-2018
Editor (s):	J.D. Fernández (WU), Wouter Dullaert (TF)
Author (s):	J.D. Fernández (WU), Wouter Dullaert (TF)
Reviewer (s):	Sabrina Kirrane (WU), Rigo Wenning (ERCIM), Rudy Jacob (PROXIMUS)
Participant(s):	U. Milosevic (TF), Jonathan Langens (TF), P.A.~Bonatti (CeRICT)
Work package no.:	3
Work package title:	Big Data Policy Engine
Work package leader:	TF
Distribution:	PU
Version/Revision:	1.0
Draft/Final:	Final
Total number of pages (including cover):	34

Disclaimer

This document contains description of the SPECIAL project work and findings.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any responsibility for actions that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the SPECIAL consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 28 Member States of the Union. It is based on the European Communities and the Member States cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors (http://europa.eu/).

SPECIAL has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731601.

Contents

List of Figures

List of Tables

1 Summary

The aim of this deliverable is to test the scalability and robustness of the SPECIAL platform such that the results can be used to inform future releases of the platform.

What is in this deliverable

In this version of the deliverable we pay particular attention to: (i) introducing the general benchmark scenario and the non-functional desiderata, in *Chapter* 1; (ii) setting up the methodology that will guide this and future versions of this deliverable, including the preparation of the synthesised test data and the identification of key performance indicators, in *Chapter* 2; (iii) providing an initial evaluation of the SPECIAL platform both in terms of performance and scalability, in *Chapter* 3, and conclusions in *Chapter* 4.

This deliverable builds upon technical requirements from *D1.3 Policy, transparency and compliance guidelines V1*, *D1.4 Technical Requirements V1* and *D1.8 Technical Requirements V2*, the SPECIAL policy language which is described in *D2.1 Policy Language V1*, and the SPECIAL transparency and compliance framework presented in *Deliverable D2.3 Transparency Framework V1* and *D2.4 Transparency and Compliance Algorithms V1*. The System Under Test (SUT) refers to the current second release of the SPECIAL platform, presented in *D3.2 Policy & events release V1*.

What is *not* in this deliverable

Considering the iterative and agile nature of the project, this deliverable is not meant to serve as a complete evaluation of the SPECIAL platform, but rather as a summary of our current tests and results that will be updated regularly as the project advances. Thus, we do not deal here the security aspects, which are subject of the public penetration/hacking challenges in WP5 (*D5.3Public penetration/hacking challenges*). Note also that the usability testing is provided in WP4 (*D4.2 Usability testing report V1*). Instead, this document aims to describe the performance and scalability test to be performed in current and future version of the platform.

Similarly, we do not deal with any issue related to compliance checking (based on business rules) of existing Line of Business and Business Intelligence / Data Science applications (described in *Deliverable D2.3 Transparency Framework V1*). It is worth noting that the implementation and testing plans of the pilots are devoted to WP5 (*D5.1 Processing and aggregation pilot and testing plans V1*, *D5.3 Sharing Pilot and testing plans V2* and *D5.5 Final Pilot implementations and testing plans V3*). The information of this deliverable, and its future versions, will be used to guide these evaluations.

Chapter 1

Introduction

In this chapter we introduce our benchmark scenario by summarizing the current functionality and components of the SPECIAL platform, our System Under Test (SUT). Then, we collect requirements and considerations that will guide our benchmark approach, which is presented in the next chapter. Finally, we review relevant state of the art.

1 The SPECIAL platform

One of the core technical objectives of SPECIAL is to implement consent, transparency and compliance mechanisms for big data processing. The SPECIAL platform uses Semantic Web technology in order to model the information that is necessary to automatically verify that data is processed according to obligations set forth in the GDPR (i.e. usage policies, data processing and sharing events, and the regulatory obligations).

As presented in *D1.4 Technical Requirements V1* and *D1.8 Technical Requirements V2*, the SPECIAL platform consists of three primary components:

(i) *The SPECIAL Consent Management Component* is responsible for obtaining consent from the data subject and representing it using the *SPECIAL usage policy vocabulary* (*D2.1 Policy Language V1*);

(ii) *The SPECIAL Transparency Component* is responsible for presenting data processing and sharing events to the user in an easily digestible manner following the SPECIAL policy log vocabulary (*D2.3 Transparency Framework V1*); and

(iii) *The SPECIAL Compliance Component* focuses on demonstrating that data processing and sharing complies with usage control policies (*D2.4 Transparency and Compliance Algorithms V1*).

This deliverable specifically focuses on evaluating the scalability and robustness of the SPECIAL transparency and compliance components. Note that the SPECIAL consent management component is mostly related to our efforts on user interaction in WP4 (cf. see *D4.2 Usability testing report V1*).

In *D3.2 Policy & events release V1*, the SPECIAL transparency and compliance components are materialized in a practical implementation of the SPECIAL platform. Therefore, this deliverable will report on the preliminary evaluations of the developed prototype.

The system architecture of our current system is depicted in Figure 1.1.

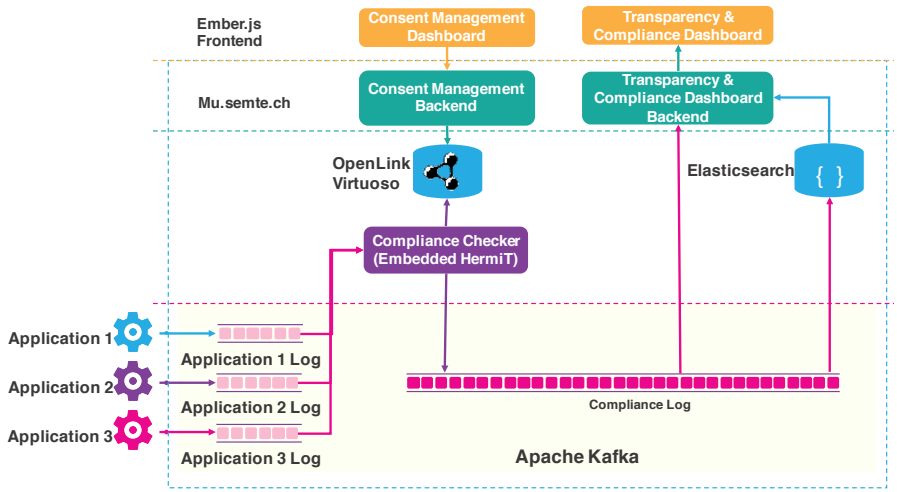

Figure 1.1: A Scalable Consent, Transparency and Compliance Architecture

SPECIAL Transparency Component. Data processing and sharing event logs are stored in the Kafka[1] distributed streaming platform, which in turn relies on Zookeeper[2] for configuration, naming, synchronization, and providing group services. A Kafka topic is used to store application logs, while a separate compliance topic is used to store the enriched log after compliance checks have been completed.

As logs can be serialized using JSON-LD, it is possible to benefit from the faceting browsing capabilities of Elasticsearch[3], and the out of the box visualization capabilities provided by Kibana.

Compliance Checker. The compliance checker, which currently includes an embedded HermiT[4] reasoner uses the consent saved in a Virtuoso triple store together with the application logs provided by Kafka to check that data processing and sharing complies with the relevant usage control policies. The results of this check are saved onto a new Kafka topic.

Interaction between the various architectural components is managed by mu.semte.ch[5] an open source micro-services framework for building RDF enabled applications.

To the best of our knowledge, no benchmark exists for the GDPR-based compliance and transparency services such as the ones provided by the SPECIAL platform. However, the existence of such systems and benchmarks is of utmost importance to identify shortcomings, optimize the performance and guide future directions. In the following, we provide additional considerations and technical requirements that are relevant in order to benchmark compliance and transparency components emerging from our efforts in SPECIAL, and we review relevant state of the art. Our benchmarking approach and evaluation is presented in the next chapters.

[1] https://kafka.apache.org/
[2] https://zookeeper.apache.org/
[3] https://www.elastic.co/products/elasticsearch
[4] http://www.hermit-reasoner.com/
[5] https://mu.semte.ch/

Table 1.1: Transparency and compliance services.

Component	Functionalities	Current support in SPECIAL platform (release - D3.2)
Transparency component	List the data processing and sharing events happened	Total
	Find data processing and sharing events by data subject, by consent, by temporal window	Partial (temporal filter is not supported)
	Add data processing and data sharing events to the transparency ledger	Total
	Export the transparency data in an interoperable format	Total
Compliance component	Coherency validation of transparency data and consent data	Total
	Can be called by an access control system for ex-ante compliance checking	Not supported
	Can process the transparency ledger for ex-post compliance checking	Total
	Get statistics for key parameters (#consents, #revocations, #data sharing events, #data processing events ...)	Partial (supported for most parameters)

1.1 Considerations and Technical Requirements

Table 1.1 recalls the services we foresee for the transparency and compliance components (see *D1.8 Technical Requirements V2*), and the current support in the SPECIAL platform (release - D3.2). As can be seen, most of the transparency services are already in place. However, our current prototype only supports basic filtering of processing and sharing events. Our current benchmark, presented in the next chapter, will consider this basic functionality, while more expressive queries are deferred to future versions of this deliverable. In turn, the compliance component implements the core functionality, but does not currently support ex-ante compliance checking. Thus, in this version of the deliverable we focus on ex-post compliance checking, which will be extended as soon as the platform implements ex-ante mechanisms.

1.1.1 Non-functional requirements

Before discussing the practical benchmark and its results, let us recall and discuss some of the non-functional desiderata presented in *D1.3 Policy, transparency and compliance guidelines V1* (also reviewed by Bonatti et al. [3]) and *D1.8 Technical Requirements V2*:

Storage: Given the volume of events and policies that will need to be handled, the scalability of event data processing is a major consideration. Parameters such as the number of data subjects, the number of consent requests and the number of data processing steps, have a multiplicative effect.

In this respect, as described in *D3.2 Policy & events release V1*, the SPECIAL platform makes use of a specific Kakfa feature, referred to as *log compaction*, which reduces storage needs. In particular, the compliance checker feeds on a compacted Kafka topic which holds the complete policies for all data subjects, where duplicates are removed. We can expect other platforms to use similar features in order to reduce the storage footprint.

It is also worth mentioning that the replication factor of the underlying distributed filesystem can increase the storage needs significantly (but improves the overall fault-tolerance of the system), hence this information is crucial for benchmarking. In our current scenario, we consider a replication factor of two, i.e., data is written to two nodes.

Finally, note that we consider instantaneous data sharing and processing events. In *Deliverable D2.3 Transparency Framework V1*, we discuss a grouping feature for the events,

which is not currently supported by the SPECIAL platform and will be consider as part of future work.

Scalability: Because of the multiplicative effect is it important that the SPECIAL architecture can adapt to larger volumes i.e. via both horizontal and vertical scaling.

As shown in Figure 1.1, the SPECIAL platform runs on proven open source software that is used at large scale by some of the largest companies in the world[6] [4, 5, 13]. In *D3.2 Policy & events release V1*, we provide details on how the system can scale to support a load beyond what a single instance can handle.

Thus, the benchmark tasks should build upon a real-word large-scale scenario, where the ability of the system to scale horizontally and vertically can be validated.

Performance & responsiveness: The total volume of data should only marginally impact the performance and responsiveness of the services. Creating a single data store will destroy the data locality for some services, impacting the responsiveness.

As discussed in *D3.2 Policy & events release V1*, Kafka is specifically dedicated for high-performance, low-latency commit log storage. Given its streaming focus (yet it efficiently supports batch oriented workload), the system can perform near real time data processing. Similarly, the SPECIAL transparency component is based on Elasticsearch, which provides efficient query times, heavily relying on the filesystem cache.

Our benchmarking scenario is designed to assure that the SPECIAL platform can cope with such requirements, assuring an overall efficient performance and low latency.

Availability & Robustness & long-term applicability: Since transparency and compliance management is bound to a legal obligation, solutions should be guaranteed to work for many years. For personal data, the GDPR calls for a long-term durable solution. If changed, the new system should be capable of importing the existing transparency and compliance data.

The SPECIAL platform makes use of the ability of Kafka to store records in a fault-tolerant durable way. For example, as described in *D3.2 Policy & events release V1*, in case of catastrophic failure where all consumers die, the system can recover the last processed event from a special *state* topic. This prevents redoing work which was already done previously and avoid data loss.

The evaluation of fault-tolerance aspects is deferred to future work.

Security: In addition to the above requirements, all components in the ecosystem must adhere to a general requirement of data security, as it is imperative that a breach of security does not hinder the operations of the systems.

D3.2 Policy & events release V1 discusses current authentication and authorization methods for the SPECIAL platform. While, *D1.8 Technical Requirements V2* identifies data privacy threats mitigations. In this deliverable we do not directly address this aspect, as security aspects will be subject of the public penetration/hacking challenges in WP5 (*D5.3Public penetration/hacking challenges*).

[6] Elasticsearch use cases:[7]

1.1.2 Considerations for Compliance Checking

SPECIAL policies are encoded using a fragment of OWL2-DL. As discussed in *Deliverable D2.3 Transparency Framework V1* and *D2.4 Transparency and Compliance Algorithms V1*, the main policy-related reasoning tasks are reduced to subsumption and concept consistency checking. That is, checking whether a data controller's policy P_0 complies with a data subject policy (i.e. consent) P_1 amounts to checking whether the inclusion $P_0 \sqsubseteq P_1$ is entailed by the ontologies for the policy language and the vocabulary.

As mentioned above, and depicted in *Figure* 1.1, our prototype performs the compliance checking on the HermiT reasoner. In practice, the subsumption algorithm is OWL API 3.4.3 1 compliant, hence HermiT should be easily swapped with any other OWL API 3.4.3 compliant reasoner, such as the provided in *D2.4 Transparency and Compliance Algorithms V1*. Thus, this deliverable focuses on evaluating the current HermiT reasoner, while the integration with the algorithm in *D2.4 Transparency and Compliance Algorithms V1*, and the evaluation of its performance is deferred to future versions of this deliverable.

1.2 State of the Art

To the best of our knowledge, no established benchmark covers the identified transparency and compliance operations, summarized in Table 1.1 nor the requirements listed in Section 1.1.1, which are the main objective of the SPECIAL platform. This motivates our proposed benchmark (presented in the next chapter), which covers most of the core operations and requirements, and it is designed to be flexible and extensible in the future.

Nevertheless, much work has been done in benchmarking OWL2 reasoners, which is a central aspects for the compliance component, as discussed above. Traditionally, the elements in OWL benchmarking are classified in *data schema*, *workload* and *performance metrics* [2, 8, 9, 10]. The former mostly refers to the structural complexity of the data schema and the usage of particular ontology constructs. The workload comprises (i) the data generation process, which often produces datasets of different sizes, and (ii) the queries or reasoning tasks to be performed by the reasoner, which should be able to evaluate the inference capability and scalability of reasoner. Finally, the performance metrics describe the quantitative measures that should be evaluated, such as: loading time, which can include different subtasks such as loading ontologies and checking ABox consistency [2], query response time, i.e. the time needed to solve the given reasoning task, completeness and soundness [7].

When it comes to well-established OWL benchmarks, the Lehigh University Benchmark (LUBM) [8] is one of the first and most popular proposals. LUBM considers an OWL Lite ontology with different ABox sizes, where different reasoning tasks of answering conjunctive queries are proposed. The University Ontology Benchmark (UOBM) [10] extends LUBM to include both OWL Lite and OWL DL ontologies and constructs. In turn, Weithöner et al [14] discuss deficiencies and challenges of OWL benchmarks, listing a set of potential requirements such as separating measurements in each step of the process, allowing for different ontology serializations, or disclosing the reasoners capabilities with respect to query caching.

Recently, the OWL reasoner evaluation (ORE) competition [12] provides different reasoning challenges. ORE is generally based on the tasks of consistency, classification, and realisation, on two OWL profiles (OWL DL and EL). Regarding the data corpus, ORE considers (i) different ontologies submitted by users and (ii) sampled ontologies from different domains.

Chapter 2

Evaluation strategy for the SPECIAL platform

In this chapter we present the benchmark for the GDPR-based transparency and consent we developed in the context of the SPECIAL project, referred to as the *SPECIAL Transparency and Consent Benchmark* (`STC-bench`) hereinafter.

The application scenario considers the SPECIAL BeFit scenario of fitness tracking presented in *D1.3 Policy, transparency and compliance guidelines V1*, which deals with a potential large volume of streaming content, namely location and heart data from BeFit devices.

As we motivated in the previous chapter, there is a lack of benchmarks to evaluate the GDPR-based compliance and transparency services such as the ones provided by the SPECIAL platform. Thus, in addition to serving our evaluation purposes, we expect `STC-bench` to become a valuable asset for similar systems implementing GDPR-based transparency and compliance.

We design `STC-bench` following the same methodology as most of the benchmarks under the H2020 HOBBIT[1] (*Holistic Benchmarking of Big Linked Data*) project [11]. Thus, the design of the benchmark considers three main aspects:

(i) First, we *identify the choke points*, that is, the identified technical difficulties that the benchmark should consider to challenge the system under test (our SPECIAL platform). We present our choke points in Section 1.

(ii) Then, the benchmark *data* is selected. In our case, and given our scenario, we propose a generator of synthetic data, described in Section 2.

(iii) Finally, we design *benchmarking tasks* to cover the identified choke points. Section 3 presents and discusses the current tasks in `STC-bench`.

The `STC-bench` data generator and the results of the evaluation (presented in the next chapter) are publicly available in our website[2], which will be continuously updated with the last results of our tests.

[1] `https://project-hobbit.eu/`
[2] `https://www.specialprivacy.eu/benchmark`

1 Choke Point-based Benchmark Design

We design `STC-bench` following the same methodology as most of the benchmarks under the H2020 HOBBIT project [11]. Thus, the development of the benchmark is driven by so-called "choke-points", a notion introduced by the Linked Data Benchmark Council (LDBC) [1, 6]. A choke-point analysis is aimed at identifying important technical challenges to be evaluated in a query workload, forcing systems onto a path of technological innovation. This methodology depends on the identification of such workload by technical experts in the architecture of the system under test.

Thus, we analysed the SPECIAL platform with the technical experts involved in the SPECIAL policy vocabulary, the transparency and the compliance components. Following this study, we identified the transparency and compliance choke points described below.

Transparency choke points:

CP1 - Concurrent access. The benchmark should test the ability of the system to efficiently handle concurrent transparency requests as the number of users grows.

This choke point mostly affects the *scalability* and the *performance and responsiveness* requirements identified in the previous chapter (see Section 1.1). On the one hand, the system must scale to cope with the increasing flow of concurrent transparency requests. Ideally, the system can dynamically scale based on the work load without interruptions, being transparent to users. On the other hand, the performance and responsiveness (in particular, the latency of the responses) should be unaffected irrespective of the number of users or, at worst, being affected marginally.

In the current version of the SPECIAL platform (release - D3.2), the transparency component fully relies on Elasticsearch, where different thread pools can be specified[3].

CP2 - Increasing data volume. The system should provide mechanisms to efficiently serve the transparency needs of the users, even when the number of events in the system (i.e. consents, data processing and sharing events) grows.

In this case, in addition to the previous consideration on *scalability* and the *performance and responsiveness*, special attention must be paid to the *storage* requirements and the indexing mechanisms of the system, such that the accessing times do not significantly depend on the existing data in the system (e.g. the number of events).

As mentioned in the previous chapter, the SPECIAL platform makes use of *log compaction* to reduce the space needs (see (*D3.2 Policy & events release V1* for further details). As for Elasticsearch, we use the default configuration, where further inspection on different compression options (e.g. using the DEFLATE algorithm[4]) is deferred to future work.

CP3 - Ingestion time in a streaming scenario. The benchmark should test that the transparency needs are efficiently served in a streaming scenario, i.e. the user should be able to access

[3] See Elasticsearch documentation: `https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-threadpool.html`

[4] See compression in Elasticsearch: `https://www.elastic.co/blog/store-compression-in-lucene-and-elasticsearch`

the information of an event (and the result of the compliance check) shortly after the event arrives to the system.

This choke point implies that no significant delays are introduced (i) by the compliance checker, and, specifically (ii) by the ingestion of the event in the transparency system.

Interestingly, engines such as Elasticsearch are mostly focused on read-intensive operations. Thus, the benchmark should consider this choke point to evaluate whether write-intensive streaming scenarios can become a bottleneck in the system.

Compliance choke points:

CP4 - Different "complexities" of policies. In general, policies can be arbitrarily complex, affecting the overall performance of any compliance checking process. Thus, the benchmark must consider different complexities of policies, reflecting a realistic scenario.

In our case, as discussed in the previous chapter, SPECIAL policies are encoded using a fragment of OWL2-DL, where the main task of the reasoner is to perform subsumption and concept consistency checking. Although this process could be very efficient, the complexity of the policy can be determined by: (i) the number on intersecting concepts in each category (*data*, *processing*, *purpose*, *storage* and *recipients*) of the SPECIAL *Minimum Core Model* (MCM), given that each of them has to be considered to perform the compliance checking, and (iii) the number of *UNION* policies that conform to the user consent, given that the compliance checker must analyse all of them before assuring that one event is not compliant with a given consent.

CP5 - Increasing number of users. The benchmark should test the ability of the system to efficiently scale and perform as increasing number of users, i.e. data processing and sharing events, are managed.

As previously discussed, the current version of the SPECIAL platform relies on Kafka to implement the compliance component. Kafka, can scale both horizontally and vertically, balancing topic partitions between cluster nodes. In this scenario, the benchmark must be able to provide a stress test to evaluate the the performance of the system when the number of users grows and starts to exceed the resource capabilities of the system.

CP6 - Expected passed/fail tests. In general, the benchmark must consider a realistic scenario where policies are updated, some consents are revoked, and others are updated. The benchmark should provide the means to validate whether the performance of the system depends on the ratio of passed/fail tests in the work load.

Note that our current version of the SPECIAL platform preserves the full history of policies and consents. However, the transparency component only considers the last consent of users to evaluate the compliance of the processing and sharing events. The implementation and evaluation of past consents is deferred to future work.

CP7 - Data generation rates. The system should cope with consents and data processing and sharing events generated with increasing rates, addressing the "velocity" requirements of most big data scenarios.

In our case, Kakfa provides the necessary toolset to deal with real-time streaming needs. However, the capacity of the system is delimited by the infrastructure (the underlying

cluster). The benchmark should be flexible enough to test the capabilities of the deployed system and its *scalability*.

Note also that this choke point is of particular interest for "online users" in ex-ante compliance checking scenarios (as shown in Table 1.1).

CP8 - Performant streaming processing. The benchmark should be able to test the system in a streaming scenario, where the compliance checking should fulfil the aforementioned requirements of *performance and responsiveness* (latency).

Note that the SPECIAL platform is specifically designed to cover such streaming needs. Nonetheless, the benchmark should help in determining the expected latency distribution for a given work load on a supporting infrastructure.

CP9 - Performant batch processing. In addition to streaming, the system must deal with performant compliance checking in batch mode.

In our case, this choke point is particularly relevant as SPECIAL is based on the streaming-based Kafka framework, which can also manage batch processing. In future work, we plan to evaluate batch-based frameworks such as our proposal SPIRIT (see *D2.4 Transparency and Compliance Algorithms V1*). SPIRIT is an architecture that leverages the SANSA[5] stack for transparency and compliance, based on Spark and Flink distributed processing tools.

2 Data Generation

In the following we present the `STC-bench` data generator to test the compliance and transparency performance of the SPECIAL platform.

First, and foremost, note that the data generation should consider two related concepts: the controllers' policies and the data sharing and processing events that are potentially compliant with the user consent.

When it comes to the policies, we distinguish three alternative strategies to generate pseudo random policies:

(a) Generating policies in the PL fragment of OWL 2, disregarding the SPECIAL minimum core model (MCM);

(b) Generating random policies that comply to the SPECIAL minimum core model (MCM);

(c) Generating not fully random (i.e. pilot oriented policies) subsets of the business policies.

In this deliverable, we focus on the second alternative, providing a synthetic data generator following the BeFit scenario. In future versions of this deliverable, we plan to investigate alternative approaches.

In addition, the classes in the policies and the log events can come from the standard SPECIAL policy vocabulary, or can be extended with new terms from an ontology. At this stage, we consider the SPECIAL policy vocabulary as the core input.

Thus, the `STC-bench` data generator can produce both policies and data sharing and processing events. The following parameters can be set:

[5] `http://sansa-stack.net`

Table 2.1: Transparency queries for the data subject and the data controller

ID	User	Query
Q1		All events of the user
Q2		Percentage of events of the user passed
Q3		Percentage of events of the user failed
Q4	Data subject	All events of the user passed
Q5		All events of the user failed
Q6		Last 100 events of the user
Q7		All events of the user from a particular application
Q8		All events
Q9		Percentage of events passed
Q10		Percentage of events failed
Q11	Data controller	All events passed
Q12		All events failed
Q13		Last 100 events
Q14		All events from a particular application

- *Generation rate:* The rate at which the generator outputs events. This parameter understands golang duration syntax eg: 1s or 10ms.
- *Number of events:* The total number of events that will be generated. When this parameters is <=0 it will create an infinite stream .
- *Format:* The serialization format used to write the events (json or ttl).
- *Type:* The type of event to be generated: *log*, which stands for generating data sharing and processing events, or *consent*, which generate new user consents.
- *Number of policies:* The maximum number of policies to be used in a single consent.
- *Number of users:* The number of UserID attribute values to generate.

3 Benchmark Tasks

In the following we present the set of concrete benchmark tasks for the SPECIAL compliance and transparency components. As for transparency tasks, note that the envisioned user stories in *D3.2 Policy & events release V1* list potential interaction with users, but they are too general to describe functionality to be considered in our current quantitative approach for benchmarking. A qualitative analysis can be deferred to pilot evaluations in WP5.

Thus, we establish here a set of simple tasks to be performed by the SPECIAL transparency component. The transparency tasks are illustrated in Table 2.2. In this case, the system is aimed at resolving *user and controller transparency queries*. Further work is needed to identify the expressivity of these queries. We consider a minimum subset of queries, described in Table 2.1.

In turn, Table 2.3 shows the tasks to be performed by the SPECIAL compliance component in order to cover all choke points identified above. Each task delimits the different parameters involved, such as the scenario (streaming or batch processing), the number of users, etc. These parameters follow the choke points, and their values are estimated based on consultation with the SPECIAL pilot partners. Note that all tests set a test time of 30 minutes, which delimits the number of events generated given the number of users and event generation rate in each case.

Table 2.2: Transparency tasks, all referring to user and controller transparency queries

Task	#Users	Event Rate	Policies	#events	Pass Ratio	Choke Point
T-T2	100 1K 10K 100K 1M	none	UNION of 5 p.	500M events	Random	CP1
T-T3	1000	none	UNION of 5 p.	1M 50M 100M 1B 10B	Random	CP2
T-T4	1000	1 ev./60s 1 ev./30s 1 ev./10s 1 ev./s 10 ev./s	UNION of 5 p.	500M events	Random	CP3

Table 2.3: Compliance tasks.

Task	Subtask	Scenario	#Users	Event Rate	Policies	Test Time	Pass Ratio	Choke Point
C-T1	C-T1-1 C-T1-2 C-T1-3 C-T1-4 C-T1-5	Streaming	1000	1 ev./10s	1 policy UNION of 5 p. UNION of 10 p. UNION of 20 p. UNION of 30 p.	30 minutes	Random	CP4,CP8
C-T2	C-T2-1 C-T2-2 C-T2-3 C-T2-4 C-T2-5	Streaming	100 1K 10K 100K 1M	1 ev./10s	UNION of 5 p.	30 minutes	Random	CP5,CP8
C-T3	C-T3-1 C-T3-2 C-T3-3 C-T3-4 C-T3-5	Streaming	1000	1 ev./10s	UNION of 5 p.	30 minutes	0% 25% 50% 75% 100%	CP6,CP8
C-T4	C-T4-1 C-T4-2 C-T4-3 C-T4-4 C-T4-5	Streaming	1000	1 ev./60s 1 ev./30s 1 ev./10s 1 ev./s 10 ev./s	UNION of 5 p.	30 minutes	Random	CP7,CP8
C-T5	C-T5-1 C-T5-2 C-T5-3 C-T5-4 C-T5-5	Batch	100 1K 10K 100K 1M	-	UNION of 5 p.	100K events 1M events 10M events 100M events 1B events	Random	CP9

4 Key Performance Indicators (KPIs)

In order to evaluate the ability of the SPECIAL platform to cope with the previously described tasks we defined the following key performance indicators (KPIs) for the first version of this benchmark:

- *Compliance Latency:* the amount of time between the point in which the compliance check of an event was performed and the time when the event was received. In our case, we consider that the compliance check is performed when the result is written to the appropriate Kafka topic storing the results of the process.
- *Compliance Throughput:* The average number of events checked per second.
- *Average transparency query execution:* The average execution time for the query.
- *CPU Usage by Node:* The average CPU usage by node in the system.
- *Memory Usage by Node:* The average memory usage by node in the system.
- *Disk Space:* The total disk space used in the system.

In addition to these indicators, when the system is deployed in a real-world scenario, the overhead with respect to the Line of Business application can be provided. This indicator can be considered in the future testing plans of the pilots, to WP5 (*D5.1 Processing and aggregation pilot and testing plans V1*, *D5.3 Sharing Pilot and testing plans V2* and *D5.5 Final Pilot implementations and testing plans V3*).

Chapter 3

Evaluation

This chapter shows the preliminary results on the evaluation of `STC-bench` on the current version of the SPECIAL platform (release - D2.4).

In this deliverable, we focus on compliance, as it is the most data and processing intensive task of the project, showing how `STC-bench` can be applied to measure the capabilities of a particular installation of the SPECIAL platform. Thus, the preliminary results are not meant to be complete or to reflect the full capacities of the SPECIAL platform, but they set up an initial baseline to guide future developments and evaluations. The large-scale evaluation of the complete transparency and compliance framework provided within the SPECIAL platform will be provided in future version of this deliverable, following the `STC-bench` methodology and guidelines presented here.

The remaining of the chapter is organized as follows. Section 1 provides details on the specification of the system running the SPECIAL platform under test. In Section 2 we perform a first analysis on the importance of scaling the number of compliance checking processes. Then, we present the preliminary results on the aforementioned `STC-bench` compliance tasks, presented in the previous chapter.

1 Experimental Framework

Our experiments run in an installation of the SPECIAL platform (release - D2.4) on a cluster consisting of 3 nodes. Although, it is expected that large-scale companies could provide more computational resources, this installation (i) can serve many data-intensive scenarios as we will show in the results, (ii) is meant to provide clear guidelines on the scalability of the platform, which can help to plan future installations and evaluations.

The characteristic of the cluster are the following:

- *Number of Nodes:* 3.
- *CPUs:* Each node consists of 4 CPUs per machine (2 cores per CPU).
- *Memory:* 16 GB per node.
- *Disk Space:* 100 GB per node.
- *Operating System:* CoreOS stable (1745.7.0).

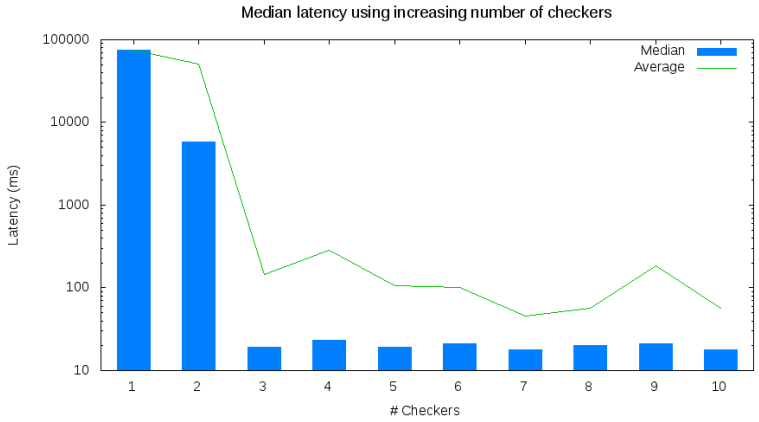

Figure 3.1: Median and average latencies with increasing number of compliance checkers

- *Replication Factor:* 2. As mentioned this implies that data is written to 2 nodes, enhancing fault-tolerance at the cost of additional space requirements and a minimum time overhead.

2 Scaling the Compliance Checking Process

Before delving into the concrete results on the `STC-bench` tasks (shown in the previous section) we present here a first study on the scalability of the system with respect to the number of processes executing compliance checking.

As stated in *D3.2 Policy & events release V1*, topics in Kafka are divided into partitions, which are the actual log structures persisted on disk. The number of partitions establishes an upper limit to how far the processing of records can be scaled out, given that a partition can only be assigned to a single consumer (in a consumer group). Thus, the total number of partitions of the application log topic will decide how many instances of the compliance checker can process the data in parallel.

Given the available resources of the cluster, we decided to set up 10 partitions, which puts an upper limit of 10 compliance checkers running in parallel.

As a first evaluation, we show how the system behaves with increasing compliance checkers running in parallel. We perform the test in a streaming (Section 2.1) and Batch processing (Section 2.2) scenario.

2.1 Streaming

For this scenario, we evaluate the streaming task *C-T1-1* from `STC-bench`, shown in Table 2.3. Note that the task considers a stream of 120,0000 events from 1,000 users, where each user generates 1 event every 10 seconds. That is, we evaluate an event stream that, on average, generates *1 event every 10ms*.

Figure 3.1 shows the median and average latencies (in milliseconds, with logarithm scale) with different number of compliance checkers in parallel, ranging from 1 to 10 (with 10 being our the upper limit defined by the number of partitions as explained above). Note that the median is usually preferred to the average given that the latency distribution can be skewed. Results

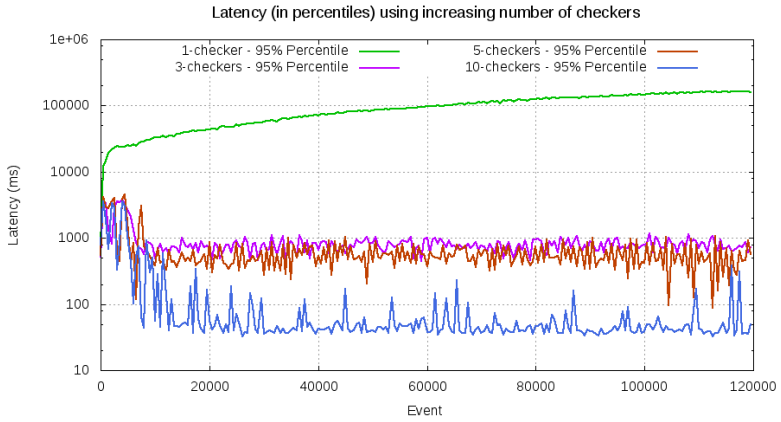

Figure 3.2: Latencies (in 95% percentile) with increasing number of compliance checkers (1, 3, 5, 10 checkers)

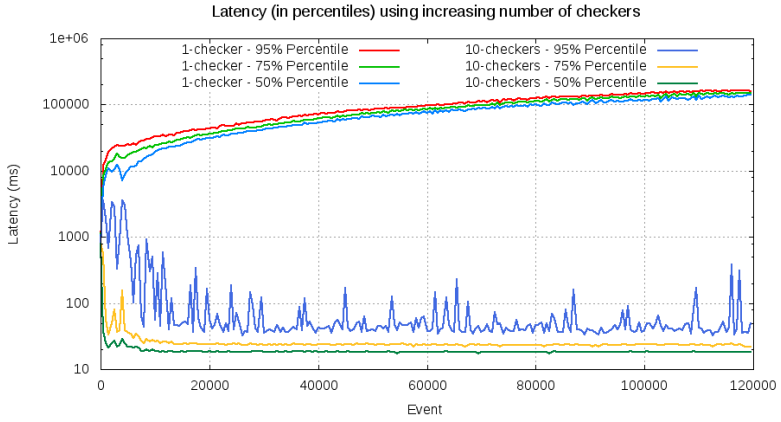

Figure 3.3: Latencies (in 95%, 75% and 50% percentile) with increasing number of compliance checkers (1, 10 checkers)

show that the (median) latency is at the level of seconds when 1 or 2 parallel checkers are considered (in particular, 75s for 1 checker, and 6s for 2 checkers), with a noticeable improvement if 3 or more compliance checkers are running in parallel, providing a *stable latency of 19-21 ms*. As expected, the slightly higher average figures denotes the expected skewed distribution.

Given this behaviour, we inspect the percentile latency, i.e, the value at which a certain percentage of the data is included. Figure 3.2 represents (in milliseconds and logarithm scale) the latency at 95% percentile, using 1, 3, 5 or 10 parallel checkers. For instance, a value of '100' ms means that 5% of the events have a latency greater than or equal to '100' ms. The distribution of 95% percentiles first shows that the latencies are stable using 3 or more checkers, but it is increasing if only 1 checker is used. This reflects that 1 checker cannot cope with the stream rate (in this case, 1 event every 10ms) and new events have to queue until they can be processed. In contrast, 3 or more checkers provides regular 95% percentile latencies. Thus, in general, only 5% of the events can experience latencies over 1 second when 3 and 5 checkers are in place, while this latency drops to 100ms when 10 checkers are used.

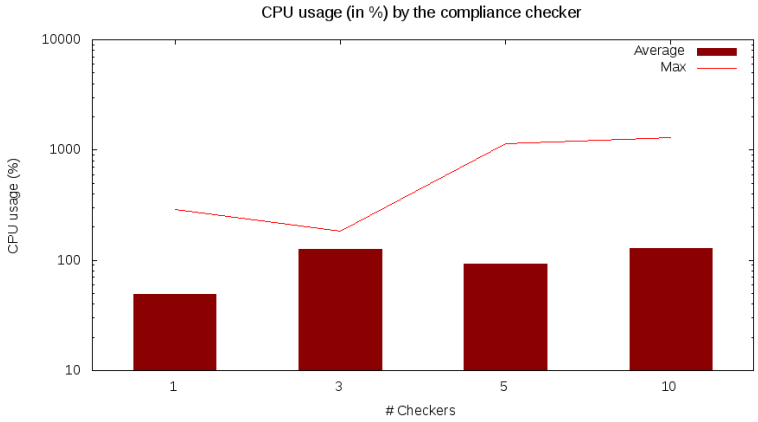

Figure 3.4: CPU usage (in %) with increasing number of compliance checkers

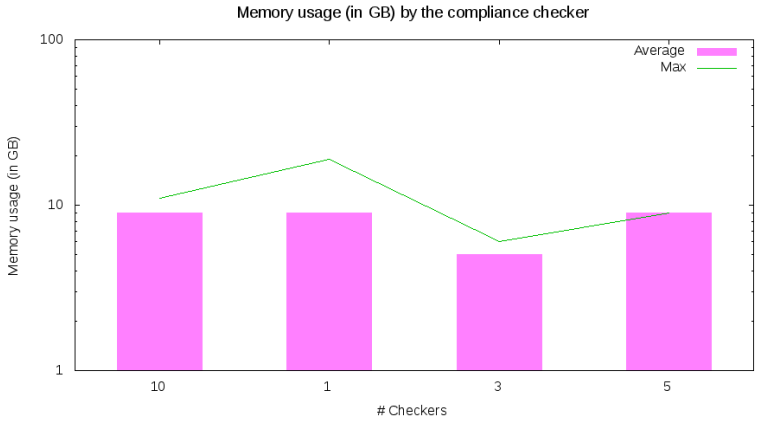

Figure 3.5: Memory usage (in GB) with increasing number of compliance checkers

Figure 3.3 completes this analysis, depicting 50, 75 and 95% percentiles for the extreme cases of having 1 or 10 checkers in place. In this case, the 50 and 75 % percentiles are close to the 95%, which reflects that most of the data is in the range of the 95% percentile.

In the following, we evaluate the CPU usage (in percentage) and memory usage (in GBs) with increasing number of parallel compliance checkers (1, 3, 5 and 10), shown in Figures 3.4 and 3.5 respectively. We report the average and the maximum number.

Results shows that (i) CPU usage increases as more parallel compliance checkers are running in parallel, and (ii) the memory consumption remains stable around 10GB, with no major influence of the number of checkers. While the first result reflects the expected behaviour when running multiple instances, the memory consumption shows that Kafka is able to optimize the use of the memory and adapt to the number of parallel checkers. In addition, it is worth mentioning that Kafka is able to add compliance checkers dynamically. Further inspection on memory management and automatic adjustment of the number of checkers based on the work load is deferred to future work.

Overall, although different application scenarios can have highly demanding real-time requirements, we expect that these figures, e.g. serving a 95% percentile latency of 100ms with

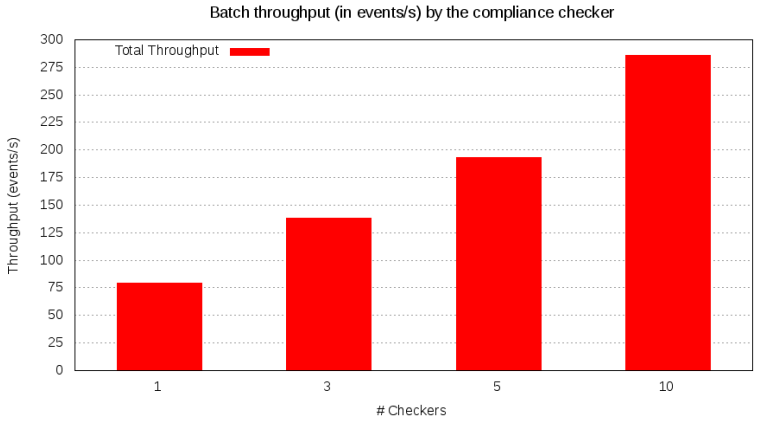

Figure 3.6: Total batch throughput (in events/s) by the compliance checker with increasing number of compliance checkers

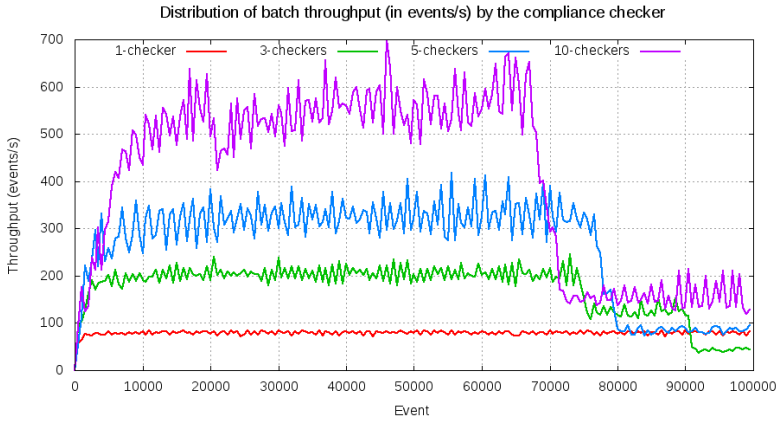

Figure 3.7: Distribution of batch throughput (in events/s) by the compliance checker with increasing number of compliance checkers

an event stream of 1 event every 10ms, can cover a wide range of real-world scenarios. Recall that the limit of 10 parallel compliance checkers is solely bounded to the number of partitions in the installation, which depends on the resources of the cluster.

2.2 Batch processing

As stated in choke point CP9, the system must also deal with performant compliance checking in batch. Thus, we repeat the previous analysis looking at different number of compliance checkers for the case of batch processing. To this aim, we evaluate the batch task *C-T5-1* from `STC-bench`, shown in Table 2.3. This task considers 100,0000 events that are already loaded in the system. Given that we process events in batch, we inspect the provided throughput (processed events per seconds) using an increasing number of compliance checkers.

Figure 3.6 shows the total batch throughput (in events/s) for 1, 3, 5 and 10 compliance checkers running in parallel. Similarly to the streaming scenario, the performance is improved significantly as more instances are running concurrently. In this case, we can observe a sublinear

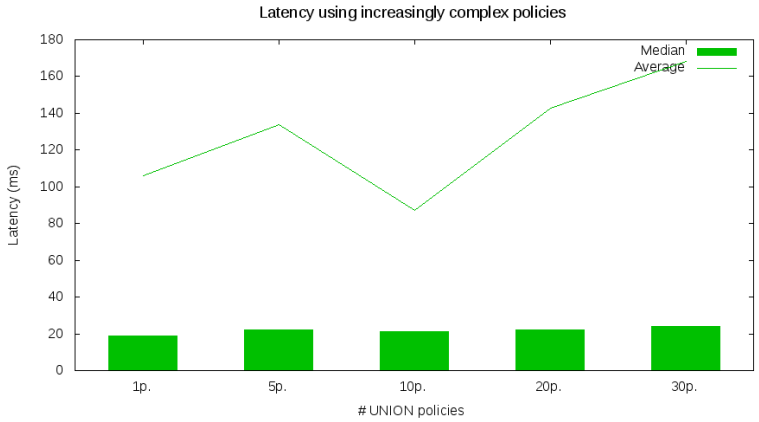

Figure 3.8: Median and average latencies with increasing complex policies

behaviour, where the throughput ranges from 79 events/s with 1 checker to 286 events/s with 10.

Figure 3.7 shows the distribution of batch throughput (in events/s) across time, for 1, 3, 5 and 10 compliance checkers. Results are consistent with the throughput reported above, showing the scalability of the system with increasing checkers running in parallel. Interestingly, the throughput is not constant, but it tends to decrease at the end of the process. This reflects the behaviour of Kafka, which assign records to a partition (and thus to a compliance checker) based on the data subject ID. As some partitions can be more loaded than others, some instances of the compliance checking may need more time to complete.

Although the results for batch processing are already promising, further work is needed to inspect and optimize the usage of the multiple checkers towards a linear scalability.

3 Preliminary Results on STC-bench Compliance Tasks

This section provides preliminary results on the `STC-bench` tasks, shown in the previous section. As mentioned above, rather than showing a complete evaluation on an optimized and performant infrastructure, we focus on testing an installation of the SPECIAL platform and pinpointing good spots for optimisation.

We limit our scope to the functionalities provided by the current SPECIAL platform and the scaling capabilities of the infrastructure (see the specifications in Section 1). In the following we present the results for all the compliance tasks (*C-T1* to *C-T5* from Table 2.3), disregarding *C-T3*, which we devote for future work. The description of each task and subtask is provided in the previous chapter (see Table 2.3).

3.1 C-T1: Different Complexities of Policies

Recall that this task regards the behaviour of the system in a streaming scenario (at 1 event/10s per user and 1K users) when different complexities of policies, measured as the number of union policies, are considered. In this scenario, we make use of 5 compliance checkers running in parallel (as detailed in Section 2.1, there is little difference between 3-10 instances).

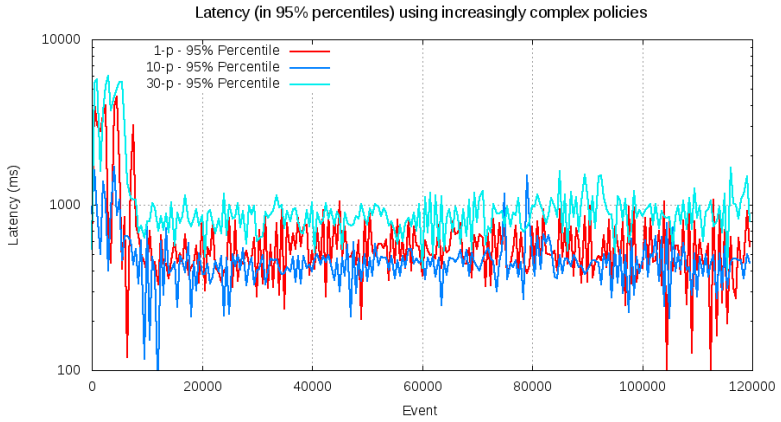

Figure 3.9: Latencies (in 95% percentile) with increasing complex policies

Figure 3.8 shows the median and average latencies (in milliseconds) with 1, 5, 10, 20 and 30 union policies. Results show that the median *latency ranges between 19-24 ms*, hence it does not increase linearly as the number of union policies grows. The higher figures for the average latency again denotes a skewed distribution.

Thus, we inspect the latency at 95% percentile (the value at which 95% of the data is included), depicted in Figure 3.9 for 1, 10 and 30 policies. The distribution shows that, in all scenarios, the latency at 95% percentile is stable, with small differences with increasing complex policies. Results also shows that, in general, only 5% of the events can experience latencies over 1 second, even when the consents consist of 30 union policies.

3.2 C-T2: Increasing Number of Users

The second task in `STC-bench` focuses on evaluating the scalability of the system with increasing number of users, from 100 to 1 million. These users are considered to be generating events in parallel, each of them at a rate of 1 event every 10 seconds. In the following evaluation, we limit our study to the first three subtask, covering up to 10,000 users given the characteristics of the experimental infrastructure (see Section 1). Note that serving 10,000 users at the aforementioned rate already implies to manage a stream of 1,000 events every second. In this scenario, we consider 10 compliance checkers (see Section 2.1) running in parallel in order to cope with such demand. As mentioned above, we expect that this evaluation can serve as a baseline to shed light on the potential of the SPECIAL platform, guiding our current efforts.

Figure 3.10 shows the median and average latencies for 100, 1000 and 10,000 users. Results show that the system is able to provide a median latency of less than 30ms with 1,000 users (generating 1 event every 10 seconds simultaneously), which is increased up to 111ms with 10,000 users (producing a total of 1,000 events per second). However, the average latency in this last case exceeds several seconds.

In order to highlight potential worst-case scenarios, we represent the latency at 95% percentile in Figure 3.11. Note that an increasing number of users results in more events, hence the different number of events in each scenario. Interestingly, results show two different scenarios. On the one hand, a low number of users (100-1,000) results in a *95% percentile around 100 ms*,

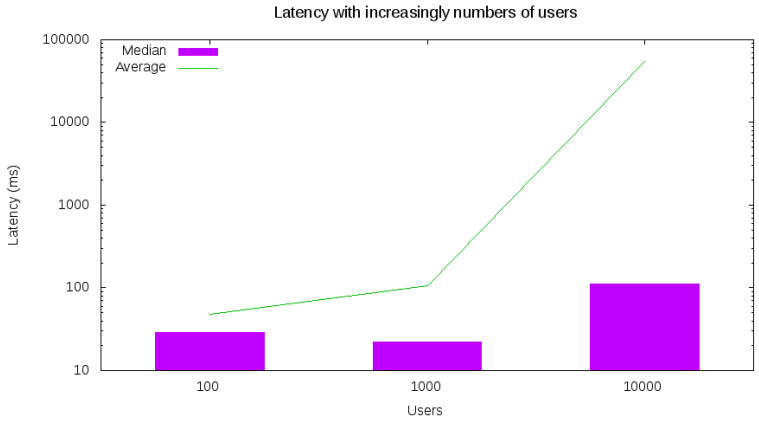

Figure 3.10: Median and average latencies with increasing number of users

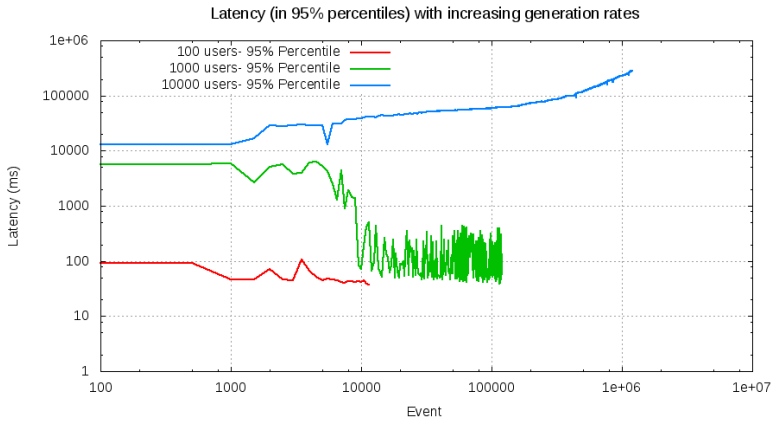

Figure 3.11: Latencies (in 95% percentile) with increasing number of users

with an initial warm-up step that produces higher latencies. On the other hand, a higher number of users (10,000) leads to increasing latencies as the number of events grows, i.e. events are queued for several seconds. The main reason is that the number of compliance checkers (10, given the amount of computational resources in the cluster) cannot cope with the overall actual ratio of 1,000 events every second. The scalability results with increasing number of compliance checkers (presented in Section 2) show that the SPECIAL platform is able to scale horizontally, hence coping with higher number of users. The evaluation of the SPECIAL platform on a different cluster configuration is deferred to future versions of this deliverable.

3.3 C-T4: Increasing Data Generation Rates

This task evaluates the performance of the system with increasing streaming rates. In this evaluation, we consider all subtasks except for *C-T4-5*, which implies a rate of 1 ev/100us that is not feasible with the experimental cluster. Given that the maximum speed is 1 ev/s (per user), we consider 10 compliance checkers running in parallel in order to try to cope with such demand.

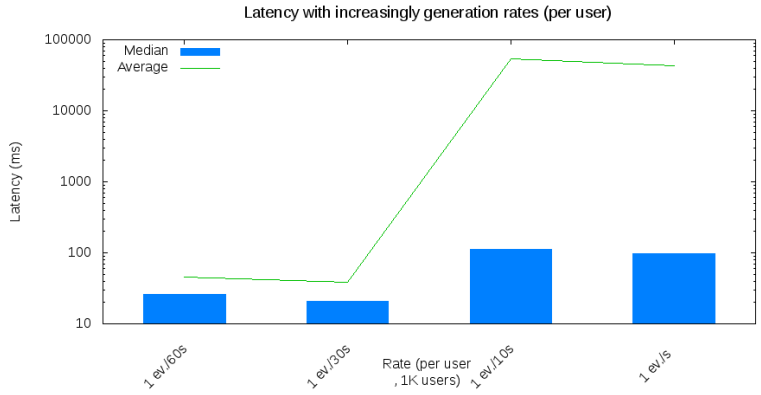

Figure 3.12: Median and average latencies with increasing generation rates. The rate refers to events per user, 1K users are evaluated)

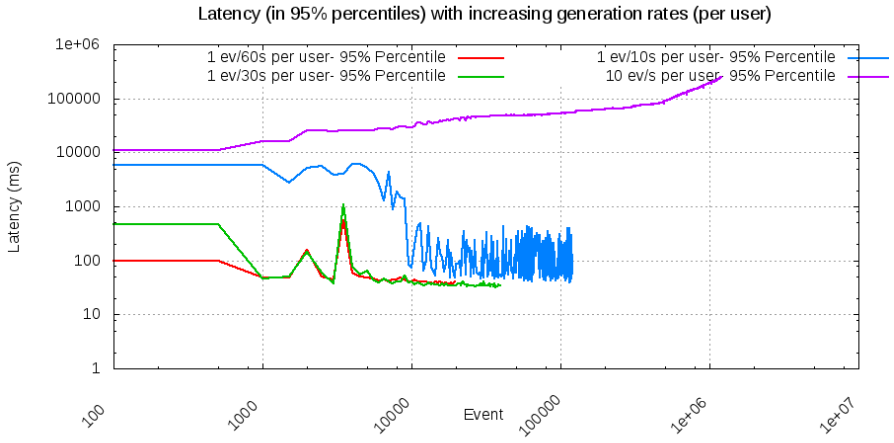

Figure 3.13: Latencies (in 95% percentile) with increasing generation rates. The rate refers to events per user, 1K users are evaluated)

Figure 3.12 represents the median and average latencies (in milliseconds and logarithm scale), while the latency at 95% percentile is shown in Figure 3.13 (in logarithm scale). Several comments are in order. First, note that the median values in Figure 3.12 are consistent with our previous latency measures (Sections 2.1 and refss:t1), obtaining values between 19-22 for rates up to 1 ev/10s (per user). Then, as expected, the median latency increases up to 98 at the highest rate of 1 ev/s.

The huge skewed distribution for the highest rates is revealed by the 95% percentile shown in Figure 3.13. Note that we fix the benchmark time in 20 minutes, so more events are generated with increasing generation rates. Results shows that, although the latency reaches a stable stage for rates up to 1 ev/10s (per user), the latency at 95% percentile grows steadily for streams at 1 ev/1s. This fact shows that the current installation cannot cope with such high rate and new events have to queue until they can be processed. The maximum latency reaches 250s for 120K events.

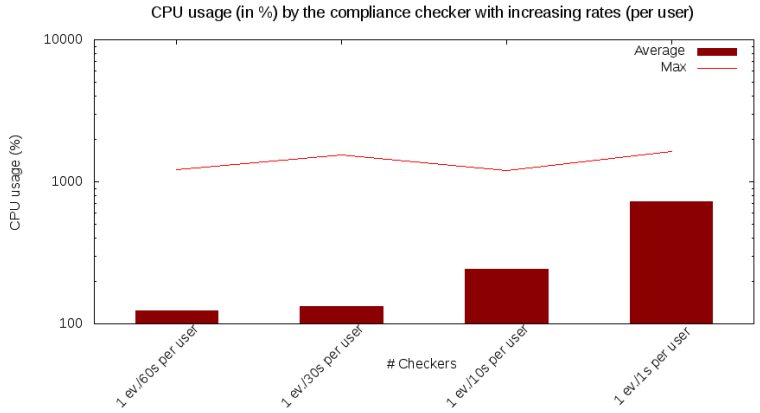

Figure 3.14: CPU usage (in %) for compliance checking with increasing generation rate (1K users)

Table 3.1: Space requirements (MB) with increasing generation rate.

# Users	Event Rate (per user)	# Events	Disk Space (MB)
1,000	1 ev./60s	20,000	733
1,000	1 ev./30s	40,000	659
1,000	1 ev./10s	120,000	1,696
1,000	1 ev./1s	1,200,000	11,068

Finally, in this case, we also inspect the CPU usage and the overall disk space of the solution. The CPU usage (in percentage) is represented in Figure 3.14. As expected, results shows that the CPU usage increases (but sublinearly) with the generation ratio. The disk space requirements are given in Table 3.1. It is worth mentioning that the disk space depends on multiple factors, such as the individual size of the randomly generated events, the aforementioned level of replication, the number of nodes and the level of logging/monitoring in the system. The reported results already shows the log compaction feature of Kafka as, on average, less bytes are required to represent each of the events with increasing event rates. In the future, we plan to study the overhead with respect to the Line of Business applications, as mentioned in the previous chapter (Section 4).

3.4 C-T5: Batch Performance

Recall that this task considers a batch processing scenario, i.e. events are already loaded in the system, with increasing number of events and users. In this evaluation, we consider all subtasks except for *C-T5-4* and *C-T5-5*. Thus, we test up to 10 million events (considering 100K events per user). We inspect the provided throughput (processed events per seconds) using an increasing number of compliance checkers. As in previous cases, we here consider 10 compliance checkers running in parallel.

Figure 3.15 shows the total batch throughput (in events/s) for 100K, 1M and 10M events. The total throughput increases with the number of events, being over 150 processed events/s in all cases, with a maximum of 608 events/s in the case of 10M events.

In turn, Figure 3.16 looks at the distribution of the throughput for the case of 1M and 10M

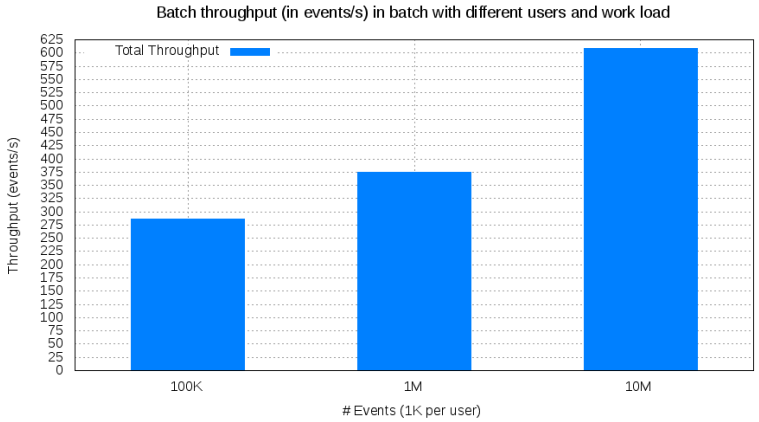

Figure 3.15: Total batch compliance checking throughput (in events/s) with increasing number of compliance checkers

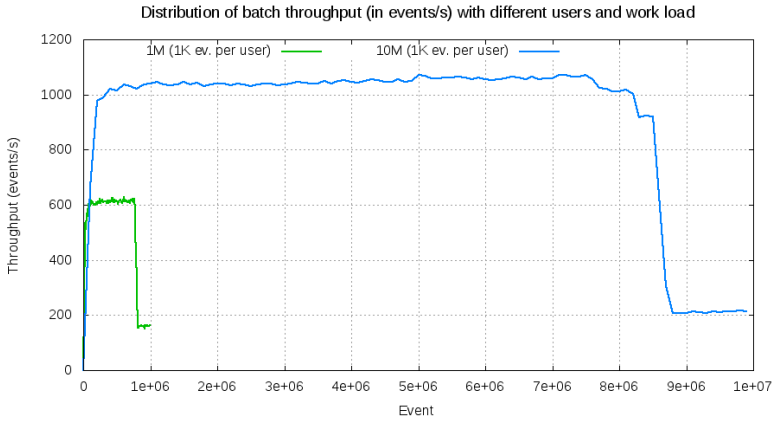

Figure 3.16: Distribution of batch compliance checking throughput (in events/s) with different users and work load. We consider 1000 events per user

events. As mentioned above, we noticed a reduction of the throughput towards the end of the process, which can point to a different work load in each partitions. Further inspection on potential optimizations in this regard is deferred to future work.

Chapter 4

Conclusions

This deliverable presents the methodology that will guide the scalability and robustness tests of the SPECIAL platform. First, we set up the scenario and discuss some of the non-functional desiderata. Then, we describe our benchmark for transparency and compliance, referred to as `STC-bench`, which (i) is designed on the basis of well-identified choke points (challenges) that would affect the performance of the SPECIAL platform and similar systems, (ii) provides a synthetic data generator that generates SPECIAL policies and data processing and sharing events, and (iii) describes key performance indicators and well-defined transparency and compliance tasks. We expect that `STC-bench` can become a valuable asset beyond SPECIAL for those tools aimed at GDPR-based transparency and compliance.

Finally, we provide a preliminary evaluation of the current version (release - D2.4) of the SPECIAL platform, limited to compliance tasks and an infrastructure consisting of a cluster of 3 nodes (each of them with 8 cores, 16GB memory and 100GB disk space per node).

Our evaluation focuses on illustrating the future use of `STC-bench` and identifying spots for optimisation. In particular, our preliminary results show that:

- The SPECIAL platform scales (sublinearly) with the number of compliance checkers running in parallel (see Section 2.1), both in a streaming and a batch scenario. Although these results are promising, further work is needed to inspect and optimize the usage of multiple checkers and to achieve a linear scalability.
- The system in place is able to serve a 95% percentile latency of 100ms with an event stream of 1 event every 10ms, which can cover a wide range of real-world scenarios.
- The system presents non-negligible delays (several seconds) when the event generation rate is faster than 1 event every 10ms. We expect to cover this scenario following two complementary strategies: (i) adding computational resources to the cluster, which will increase the number of partitions and thus compliance checkers, (ii) optimizing the compliance checking *per se*. As for this latter, we plan to compare our current built-in Hermit reasoner to a custom reasoner following the algorithm in *D2.4 Transparency and Compliance Algorithms V1*.
- The performance is marginally affected by the increasing complexity of the policies, i.e. where user consent can consist of several union policies.
- The system scales with increasing number of users, but the increased generation ratio can affect negatively the latency as mentioned above.

- The system is able to perform compliance checking in batch mode, obtaining (median) throughputs of up to 608 events per second. We plan to analyse the assignment of partitions and compliance checkers in order to optimize the process. In addition, we plan to compare to batch-oriented systems such as SANSA[1].

Overall, we expect that these insights can guide our future research and development steps of the SPECIAL platform.

[1] http://sansa-stack.net/

Bibliography

[1] R. Angles, P. Boncz, J. Larriba-Pey, I. Fundulaki, T. Neumann, O. Erling, P. Neubauer, N. Martinez-Bazan, V. Kotsev, and I. Toma. The linked data benchmark council: a graph and rdf industry benchmarking effort. *ACM SIGMOD Record*, 43(1):27–31, 2014.

[2] J. Bock, P. Haase, Q. Ji, and R. Volz. Benchmarking owl reasoners. In *ARea2008-Workshop on Advancing Reasoning on the Web: Scalability and Commonsense*. Tenerife, 2008.

[3] P. Bonatti, S. Kirrane, A. Polleres, and R. Wenning. Transparent personal data processing: The road ahead. In *International Conference on Computer Safety, Reliability, and Security*, pages 337–349. Springer, 2017.

[4] L. Engineering. Running kafka at scale, 2015. URL `https://engineering.linkedin.com/kafka/running-kafka-scale`.

[5] N. Engineering. Kafka inside keynote pipeline, 2016. URL `https://medium.com/netflix-techblog/kafka-inside-keystone-pipeline-dd5aeabaf6bb`.

[6] O. Erling, A. Averbuch, J. Larriba-Pey, H. Chafi, A. Gubichev, A. Prat, M.-D. Pham, and P. Boncz. The ldbc social network benchmark: Interactive workload. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 619–630. ACM, 2015.

[7] Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large owl datasets. In *International Semantic Web Conference*, pages 274–288. Springer, 2004.

[8] Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158–182, 2005.

[9] S. A. Khan, M. A. Qadir, M. A. Abbas, and M. T. Afzal. Owl2 benchmarking for the evaluation of knowledge based systems. *PloS one*, 12(6):e0179578, 2017.

[10] L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan, and S. Liu. Towards a complete owl ontology benchmark. In *European Semantic Web Conference*, pages 125–139. Springer, 2006.

[11] A.-C. N. Ngomo and M. Röder. Hobbit: Holistic benchmarking for big linked data. *ERCIM News*, 2016(105), 2016.

[12] B. Parsia, N. Matentzoglu, R. S. Gonçalves, B. Glimm, and A. Steigmiller. The owl reasoner evaluation (ore) 2015 competition report. *Journal of Automated Reasoning*, 59 (4):455–482, 2017.

[13] B. Svingen. Publishing with apache kafka at the new york times, 2017. URL https://www.confluent.io/blog/publishing-apache-kafka-new-york-times/.

[14] T. Weithöner, T. Liebig, M. Luther, S. Böhm, F. Von Henke, and O. Noppens. Real-world reasoning with owl. In *European Semantic Web Conference*, pages 296–310. Springer, 2007.