The SPECIAL-K Personal Data Processing Transparency and Compliance Platform

Sabrina Kirrane[1], Javier D. Fernández[1], Piero Bonatti[2], Uros Milosevic[3], Axel Polleres[1], and Rigo Wenning[4]

[1] *Vienna University of Economics and Business, Austria*
[2] *Piero Bonatti*
[3] *Tenforce, Belgium*
[4] *W3C, Sophia-Antipolis, France*

Dated: January 26, 2020

Abstract

The European General Data Protection Regulation (GDPR) brings new challenges for companies, who must provide transparency with respect to personal data processing and sharing within and between organisations. Additionally companies need to demonstrate that their systems and business processes comply with usage constraints specified by data subjects. This paper first presents the Linked Data ontologies and vocabularies developed within the SPECIAL EU H2020 project, which can be used to represent data usage policies and data processing and sharing events, including the consent provided by the data subject and subsequent changes to or revocation of said consent. Following on from this, we propose a concrete transparency and compliance architecture, referred to as SPECIAL-K, that can automatically verify that data processing and sharing complies with the relevant usage control policies. Our evaluation, based on a new transparency and compliance benchmark, shows the efficiency and scalability of the system with increasing number of events and users, covering a wide range of real-world streaming and batch processing scenarios.

1 Introduction

The European General Data Protection Regulation (GDPR) defines a set of obligations for controllers and processors of personal data. Primary obligations include obtaining explicit consent from the data subject for the processing of personal data and providing full transparency with respect to processing and sharing.

With the coming into effect of the GDPR in May 2018, several tools [11, 16, 19] have recently been developed that can be used to assist companies to assess the compliance of their systems and processes with respect to obligations set forth in the GDPR. However, such tools are targeted at self assessment (i.e. companies complete standard questionnaires in the form of a privacy impact assessment) and cannot be used to automatically check compliance with usage constraints.

Such, automated transparency and compliance mechanisms would require not only machine-readable representations of the users consent, but also machine-readable representations of data processing and sharing. SPECIAL[1] is an EU H2020 research and innovation action, which addresses these challenges by demonstrating how Semantic Web technologies can be used for both consent and personal data processing representation and compliance checking.

In particular we devise a suite of ontologies and vocabularies that can be used to: (i) model data usage policies, conforming the SPECIAL's Usage Policy Language, (ii) represent data processing and sharing events in a semantic log. Both of which have been developed in close collaboration with legal experts, thus ensuring that our automated compliance checking is tightly coupled with the legal assessment process.

[1] `https://www.specialprivacy.eu/`

The core data points required in order to automatically assess the legality of personal data processing and sharing include: (i) the type of personal *data* collected from the data subject; (ii) the *processing* performed on the personal data; (iii) for what *purpose* the data is processed; (iv) where the data are *stored* and for what *duration*; and (v) if the data is shared who are the *recipients*. In addition, we propose the SPECIAL-K architecture, a scalable solution that can be used to log personal data usage policies and events in a manner that support automated compliance testing. In order to ensure a thorough evaluation of our platform, and to support future comparative analysis, we propose the SPECIAL transparency and compliance benchmark. Our evaluation on synthetic policies and events shows that SPECIAL-K scales with increasing number of users both in a streaming and a batch scenario.

Summarising our contributions:

- we present the novel SPECIAL policy language to represent data usage policies, which can be used to represent data subject's consent in the context of GDPR. We provide an initial taxonomy for the components of the policy (data categories, processing, etc.) for use in a variety of use cases across multiple domains (e.g., finance, media, insurance, to name but a few), which can be further extended for concrete use cases;
- we define the SPECIAL event log vocabulary, which is derived from the policy language. We show how data processing and sharing events can be used to automatically check compliance with regard to the data subjects policies;
- we present the SPECIAL-K Apache Kafka[2] based big data platform, which is able to store logs at large scale and to perform scalable compliance checking; and
- we propose a synthetic benchmark for transparency and compliance, referred to as STC-bench, which is designed on the basis of well-identified choke points (challenges) that could affect the performance of SPECIAL-K and similar systems. We make use of STC-bench to provide an evaluation of SPECIAL-K on compliance tasks.

The remainder of the paper is structured as follows. Section 2 discusses alternative policy languages, logging mechanisms and vocabularies, together with GDPR compliance tools. Section 3 motivates the problem, presenting a use case scenario and the basic components of the SPECIAL architecture. We present the SPECIAL policy language in Section 4 and the log vocabulary in Section 5. We present the main components of the practical SPECIAL-K architecture for GDPR transparency and compliance in Section 6. We define the STC-bench benchmark in Section 7 and subsequently present the results of the performance evaluation of SPECIAL-K in Section 8. Finally, Section 9 concludes and discusses potential avenues for future work.

2 State of the Art

When it comes to the representation of usage policies there are several potential candidates including semantic policy languages [27, 12, 5, 13] and standard based policy languages [6, 10]. KAoS [27] is a general policy language which adopts a pure ontological approach, whereas Rei [12] and Protune [5] use ontologies to represent concepts, the relationships between these concepts and the evidence needed to prove their truth, and rules to represent policies. Kolovski et al. [13] demonstrate how together description logic and defeasible logic rules can be used to understand the effect and the consequence of sets of access control policies. While, the Platform for Privacy Preferences (P3P)[3], is a W3C recommendation, which enables websites to express their privacy preferences in a machine readable format. An more recent W3C recommendation known as the Open Digital Rights Language (ODRL)[4], which was released in February 2018, is a general rights language, which can be used to define rights to or to limit access to digital resources. In principle any of these languages could be used to encode SPECIAL's usage policies, after the necessary auxiliary ontologies have been integrated. In SPECIAL we developed our usage policy language using OWL2, and select language constructs carefully in order to achieve an optimal trade-off between expressiveness and computational complexity.

[2] `https://kafka.apache.org/`

[3] P3P,`http://www.w3.org/TR/P3P/`

[4] ODRL,`https://www.w3.org/TR/odrl-model/`

As for transparency with respect to data processing, relevant work primarily relates to the re-purposing of existing logging mechanisms as the basis for personal data processing transparency and compliance [4]. Many of the works use a secret key signing scheme based on Message Authentication Codes (MACs) together with a hashing algorithm to generate chains of log records that are in turn used to ensure log confidentiality and integrity [2] (cf. [4] for a summary of existing approaches). MACs are themselves symmetric keys that are generated and verified using collision-resistant secure cryptographic hash functions. However, only a few works [22, 24] focused on personal data processing. Sackmann et al [24] discussed how a secure logging system can be used for privacy-aware logging. Additionally, they introduces the "privacy evidence" concept and discusses how such a log could be used to compare data processing to the user's privacy policy. [25] propose an ontology that can be used to model personal data processing and demonstrate how SPARQL with limited RDFS reasoning can be used for query-based privacy auditing. A distributed architecture to manage access to personal data based on blockchain technology has been proposed by Zyskind et al. [28]. The authors discuss how the blockchain data model and Application Programming Interfaces (APIs) can be extended to keep track of both data and access transactions. More recently, Sutton and Samavi [26] propose an extension of blockchain technology with *Linked Data* to create tamper-proof audit logs and non-repudiation. Nonetheless, very little research has been conducted into the suitability of such blockchain-based solutions in an industry context.

Additionally, just focusing on the representation, there exists a number of general event vocabularies such as the *Event*[5] ontology and the *LODE*[6] ontology [23] that could potentially be used to model privacy-aware data processing *events*. However, these ontologies do not consider the particularities and requirements (such as facilitating GDPR compliance) of the data processing and sharing events considering herein. The management of events for business process compliance monitoring and process mining [15] can be seen as orthogonal work.

As for GDPR compliance, recently the Information Commissioner's Office (ICO) in the UK [11], Microsoft [16], and Nymity [19] have developed compliance tools that enable companies to assess the compliance of their applications and business processes by completing a predefined questionnaire. In addition there has been a body of work looking at modelling GDPR concepts and obligations [20, 7], in a manner that enables compliance checking beyond consent and transparency.

In this paper, we propose vocabularies that can be used to record both usage policies and data processing and sharing events in a manner that supports automatic compliance checking. One of the primary differentiators being that both our policy language and our event log have been derived from the legal inquiry process used to assess if personal data processing and sharing complies with the GDPR.

3 Personal Data Processing

In order to set the basis of our approach, we first present a general use case scenario that exemplifies the requirements derived from the SPECIAL pilots. Following on from this we provide a high level overview of the SPECIAL consent, transparency, and compliance framework.

3.1 Motivating Use Case Scenario

Sue buys a wearable device for fitness tracking from a company called *BeFit*. During the set up, Sue is presented with an *informed consent request* associated to a data usage policy. The policy says that, in order to provide the tracking service, the device will record biomedical and location data, i.e. the heart rate, duration and location of the fitness activities. These data will be stored in BeFit's servers in EU. BeFit additionally asks if these data can be used to create an activity profile that can be shared with other companies for marketing purposes (e.g. ads related to fitness including some discount for BeFit users). Sue accepts this option and starts using the device. The signed usage policy is stored in a *transparency ledger*, together with all processing and sharing events generated from the use of the device by Sue. Two years later, Sue is not using the device anymore and she starts receiving emails from a local gym that advertises its activities. Sue can connect to the ledger and discover that (i) BeFit built her profile by mining the data collected by the device, (ii) the profile, stating that she was not doing exercise lately, was shared to the local gym, and (iii) all this was compliant with Sue's data usage policy. At this

[5]Events, `http://motools.sourceforge.net/event/event.html`
[6]LODE, `http://linkedevents.org/ontology/`

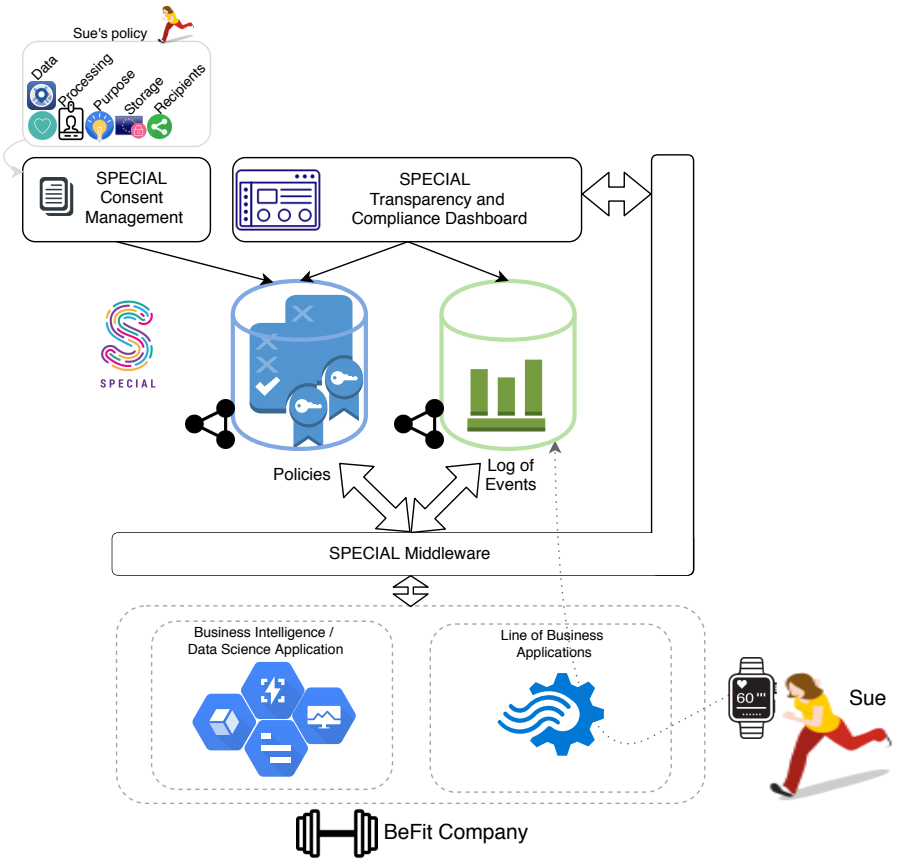

Figure 1: The SPECIAL Consent, Transparency and Compliance framework.

point, Sue can now decide to revoke the given consent and ask both BeFit and the gym to delete all of her data. The information stored in the ledger points to the data she is referring to, hence all traces can be automatically deleted.

3.2 Consent, Transparency and Compliance

The SPECIAL consent, transparency and compliance framework (shown in Figure 1) consists of two primary components the: (i) *SPECIAL Consent Management* component, which is responsible for obtaining consent from the data subject and representing it in the form of a usage policy; and (ii) *SPECIAL Transparency and Compliance Component*, which is responsible for presenting data processing and sharing events in an easily digestible manner and demonstrating that existing data processing and sharing complies with usage control policies.

SPECIAL Middleware includes sub-components that: connect the SPECIAL primary components with existing Line of Business access control mechanisms and business logic; and middleware that enables companies to perform policy aware business intelligence and data science.

In addition to existing data sources that support business operations (i.e. *Line of Business Applications*), and strategic decision making (i.e. *Business Intelligence / Data Science Applications*), we propose two additional data sources, one which is used to store the consent, regulatory and business *Policies* and another to store the data processing or sharing *Events*.

4 SPECIAL's Usage Policy Language

SPECIAL usage policies are encoded in OWL 2 [17]. In the following we provide a highlevel overview of the policy language. In the examples[7] that follow, the `spl` prefix represents `http://www.specialprivacy.eu/langs/usage-policy#`. Additional details, including the full grammar of policy expressions in Backus normal form (BNF), can be found in the SPECIAL documentation [3].

4.1 Data Usage Policy Model

Conceptually, a *usage policy* is meant to specify a *set of authorized operations*. According to the GDPR, these policies shall specify clearly which data are collected, what is the purpose of the collection, what processing will be performed, and whether or not the data will be shared with others. Usage policies can consist then of the following five elements:

- "Data" describes the personal data collected from the data subject. In order to describe which categories of data are collected, an ontology of *personal data* is needed to cover the most common data categories. It is envisaged that the ontology will be extended with suitable profiles and/or integrated with further use case specific ontologies.
- "Processing" describes the operations that are performed on the personal data. Data processing should be described through a suitable ontology of data operations.
- "Purpose" specifies the objective that is associated with data processing. Objectives such as marketing, service optimisation and personalisation, scientific research, are pervasive across a variety of contexts. Purpose descriptions are part of most usage policy languages developed so far (e.g. P3P [6] and ODRL [10]).
- "Storage" specifies where data are stored and for how long. Note that the GDPR requires that storage is strictly bound to the service needs. This implies storage minimisation, hence the need to express *upper bounds* to storage duration, that may be expressed either in terms of the duration of the service that the data have been collected for, or in absolute terms.
- "Recipients" specifies who is going to receive the results of data processing and, as a special case, whom data are shared with. The GDPR does not clearly state to which level of detail this information has to be specified, and there are potentially conflicting needs, such as the companies' desire to keep some of their business relations confidential, and the data subjects' right to trace the flow of their personal information.

Table 1 provides a high level overview of the initial vocabularies that are necessary to represent the elements of the MCM. All namespaces share the `S` which represents `http://www.specialprivacy.eu/`. Note that these vocabularies have been developed to support the initial SPECIAL use cases. Further terms will be added to accommodate additional use cases as needed.

For this purpose, SPECIAL setup the W3C Data Privacy Vocabularies and Controls Community Group (DPVCG) in 2018. The group launched on the 25th of May 2018, the official start date of the GDPR. The mission of the DPVCG is to develop a taxonomy of privacy terms, which include in particular terms from the new European General Data Protection Regulation (GDPR), such as a taxonomy of personal data as well as a classification of purposes (i.e., purposes for data collection), and events of disclosures, consent, and processing such personal data. In 2019, the group published their first versions of the Data Privacy Vocabulary[8] and the DPVCG GDPR Legal Basis Vocabulary[9]. Additional details can be found in [21].

[7] For the policy language examples we use the functional syntax which is less verbose.
[8] https://www.w3.org/ns/dpv
[9] https://www.w3.org/ns/dpv-gdpr

Table 1: SPECIAL auxiliary vocabularies for usage policies.

Category	Namespace	#Classes	Examples	Superclass
Data	svd:=(S)/vocabs/data	27	svd:Activity, svd:Anonymized, svd:Financial, svd:Health, svd:Location, svd:Navigation, svd:Preference, svd:Profile, etc.	spl:AnyData
Processing	svpr:=(S)/vocabs/processing	9	svpr:Aggregate, svpr:Analyze, svpr:Anonymize, svpr:Collect, svpr:Copy, svpr:Derive, svpr:Move, svpr:Query, svpr:Transfer	spl:AnyProcessing
Purpose	svpu:=(S)/vocabs/purposes	31	svpu:Account, svpu:Arts, svpu:Delivery, svpu:Education, svpu:Feedback, svpu:Gaming, svpu:Health,svpu:Marketing, svpu:Payment, svpu:Search, etc.	spl:AnyPurpose
Recipient	svr:=(S)/vocabs/recipients	6	svr:Delivery, svr:OtherRecipient, svr:Ours, svr:Public, svr:Same, svr:Unrelated	spl:AnyRecipient
Storage location	svl:=(S)/vocabs/locations	7	svl:ControllerServer, svl:EU, svl:EULike, svl:ThirdCountries, svl:OurServers, svl:ProcessorServers, svl:ThirdParty	spl:AnyLocation
Storage duration	svdu:=(S)/vocabs/duration	4	svdu:BusinessPractices, svdu:Indefinitely, svdu:LegalRequirement, svdu:StatedPurpose	spl:AnyDuration

4.2 Basic Usage Policies

A usage policy is composed of one or more *basic usage policies*, each of which is an OWL 2 expression of the form:

```
ObjectIntersectionOf(                                                    (1)
   ObjectSomeValuesFrom(spl:hasData SomeDataCategory)
   ObjectSomeValuesFrom(spl:hasProcessing SomeProcessing)
   ObjectSomeValuesFrom(spl:hasPurpose SomePurpose)
   ObjectSomeValuesFrom(spl:hasRecipient SomeRecipient)
   ObjectSomeValuesFrom(spl:hasStorage SomeStorage)
)
```

The important parts in this expression are the policy's attributes highlighted in bold. The policy author needs to decide for each of them a suitable range, that in the above text is highlighted in italics. The example presented authorizes all operations that:

1. fall within the specified *SomeProcessing* category,
2. operate only on data that belong to *SomeDataCategory*,
3. have any purpose covered by the *SomePurpose* category,
4. disclose the results to any member(s) of the *SomeRecipient* category, and
5. store the results in any place belonging to the *SomeStorage* category.

Therefore, policy (1) encodes the set of all authorizations that have (at least) the specified attributes, which match the minimum core model (MCM), introduced in the previous section. Although SPECIAL defines auxiliary vocabularies providing a set of classes for *SomeDataCategory, SomeProcessing, SomePurpose, SomeRecipient*, it should be noted that it is not possible to enumerate over all possible classes and as such the policy language and by extension the vocabularies were designed to be extensible.

4.3 General Usage Policies

A general usage policy may contain a union of any number of basic policies, each of them of the form (1). The resulting policy is conceptually the *union* of all the authorizations supported by the basic policies,

that is, an operation is authorized by the general policy if and only if the operation is authorized by *at least one* of its basic policies.

For instance, the following *general usage policy* states that personal data can only be used for non-commercial purposes and shall neither be stored nor disclosed to third parties, while pseudonymised data can be used freely (where auxiliary vocabularies define the terms *PersonalData, NonCommercial, PseudonymizedData*):

```
ObjectUnionOf(                                                    (2)
  ObjectIntersectionOf(
    ObjectSomeValuesFrom(spl:hasData PersonalData)
    ObjectSomeValuesFrom(spl:hasProcessing spl:AnyProcessing)
    ObjectSomeValuesFrom(spl:hasPurpose NonCommercial)
    ObjectSomeValuesFrom(spl:hasRecipient spl:Null)
    ObjectSomeValuesFrom(spl:hasStorage spl:Null)
  )
  ObjectIntersectionOf(
    ObjectSomeValuesFrom(spl:hasData PseudonymizedData)
    ObjectSomeValuesFrom(spl:hasProcessing spl:AnyProcessing)
    ObjectSomeValuesFrom(spl:hasPurpose spl:AnyPurpose)
    ObjectSomeValuesFrom(spl:hasRecipient spl:AnyRecipient)
    ObjectSomeValuesFrom(spl:hasStorage spl:AnyStorage)
  )
)
```

4.4 Use Case Specific Usage Policies

Taking the usecase scenario presented in Section 3, in Example 1 we demonstrate what Sue's policy would look like if it were represented in the SPECIAL policy language. In this example, the auxiliary vocabularies need to be extended with three new classes: the class `ex:HeartRate` (as a subclass of `svd:Health`), `ex:Profiling` (a subclass of `svpr:Analyze`) and `ex:Recommendation` (a subclass of `svpu:Marketing`).

Example 1. The following policy: *"Heart rate and location data are collected and analysed to create a user profile for the purpose of issuing recommendations. Profiles are stored indefinitely in the EU by the data controller and released to third parties."* can be formalised as follows with a factorised general policy:

```
ObjectIntersectionOf(
  ObjectSomeValueFrom( spl:hasData
    ObjectUnionOf(
      ex:HeartRate svd:Location ))
  ObjectSomeValueFrom( spl:hasProcessing ex:Profiling )
  ObjectSomeValueFrom( spl:hasPurpose ex:Recommendation )
  ObjectSomeValueFrom( spl:hasStorage
    ObjectIntersectionOf(
      ObjectSomeValuesFrom( spl:hasLocation
        ObjectIntersectionOf( svl:OurServers svl:EU ))
      DataSomeValuesFrom(   spl:durationInDays
        DatatypeRestriction( xsd:integer
          xsd:mininclusive "0"^^xsd:integer ))))
  ObjectSomeValueFrom( spl:hasRecipient spl:AnyRecipient )
)
```

□

5 The SPECIAL Log Vocabulary

Hereinafter, we focus on providing a concrete model to represent logs of data processing and sharing events, including the consent provided by the data subject and subsequent changes to or revocation of said consent. To do so, we provide the SPECIAL *SPLog* vocabulary[10] that builds upon the *SPECIAL policy language* ontology presented in Section 4 and reuses well-known vocabularies such as PROV [14] to provide provenance metadata of the log. The namespace of the vocabulary, splog, is `http://www.specialprivacy.eu/langs/splog#`.

[10]The full description of the SPLog vocabulary can be found at `http://purl.org/specialprivacy/splog`.

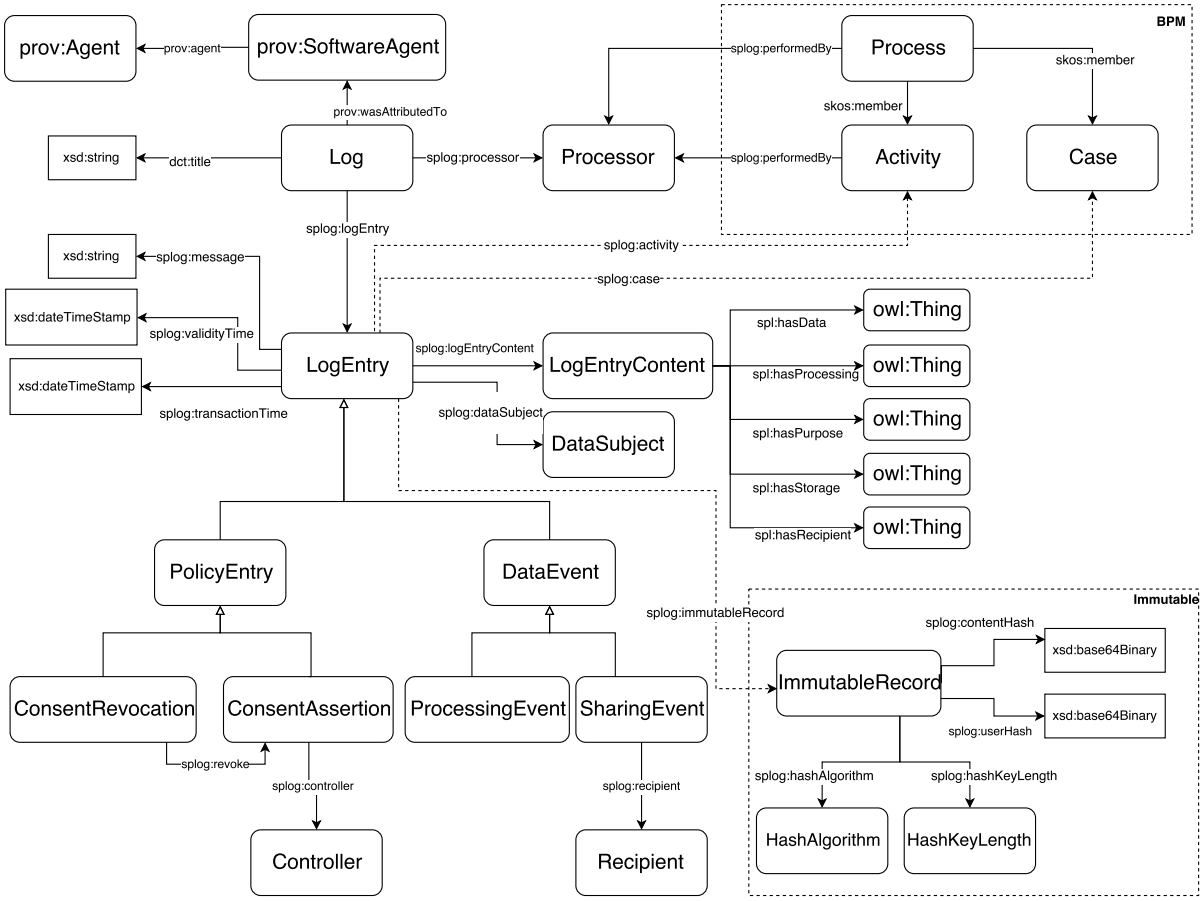

Figure 2: Outline of the SPLog main terms and their relationships

5.1 Outline of the SPLog Vocabulary

We followed the large body of work in the Business Process Management (BPM) community that focuses on using process execution events for business process compliance monitoring [15]. From this context, we borrow the following: (i) we assume a log entry contains data related to a *single process* and events are instantaneous, thus they can be associated to a single timestamp, (ii) we integrated an optional *BPM* module in our model, in order to represent BPM information (i.e. cases, processes and activities) that might be present in the company and can complement the logging information, and (iii) we integrated an optional *Immutable* module to represent that a log entry can be additionally linked to its representation as an immutable record, potentially stored in a different ledger or knowledge base (e.g. in blockchain).

Figure 2 depicts an overview of the vocabulary. Several concepts and properties have been defined to cover the log and its entries, detailed below. The description of a policy log can be complemented with the aforementioned optional conceptual modules (dashed), *BPM* and *Immutable*.

5.1.1 Log

A log (represented as `splog:Log`) is a collection of data that records data processing and sharing events as well as consent-related activities (assertion and revocation). The log can contain (i) *general log metadata* that describe the log as a whole, such as the data processor whose service is logged, modelled with the `splog:processor` property (a subproperty of `prov:agent`), and (ii) *log entries* (`splog:LogEntry`), linked via the `splog:logEntry` property (a `prov:wasGeneratedBy` subproperty).

5.1.2 Log entry

They contain information about processing and sharing events associated to data subjects, as well as actions related to the consent provided (or revoked) by data subjects. These different types of entries are represented with a hierarchy of classes, shown in Figure 2. Thus, a `splog:LogEntry` has two main types (subclasses), `splog:PolicyEntry` and `splog:DataEvent`, described as follows:

- `PolicyEntry`: This class reflects log entries related to policies and consent. We currently consider two subclasses, `splog:ConsentAssertion` specifying a consent provided by a data subject to a `splog:Controller` (which in turn can be reachable via the `splog:controller` property), and `splog:ConsentRevocation`, denoting the revocation of a given consent. Note that we assume that a consent provided by a data subject replaces any previous consent, which can be optionally linked via the `splog:revoke` property in our model.
- `DataEvent`: This class considers log entries that are actually events on the data, i.e., the aforementioned data processing and sharing events. In the case of the latter, the concrete `splog:Recipient` instances can be specified, via `splog:recipient`.

Besides general metadata and a human-friendly message (`splog:message`), the data in a log entry can be described as belonging to one of the following kinds:

- `Data subjects`: The log entry *SHOULD* reference the data subject(s) involved in the entry using the `splog:dataSubject` property (a `prov:wasAssociatedWith` subproperty). Note that in case of anonymised logs, no subject can be specified.
- `Content`: The log entry *MUST* reference the actual data of the log. This is specified with the `splog:logEntryContent` property, which points to the appropriate instance of `splog:LogEntryContent`, described below.
- `Timestamps`: The log entry *MUST* reference the (instant) time at which the event occurred using the `splog:validityTime` property (subproperty of `prov:atTime`). The log entry *SHOULD* also reflect the time in which the log was recorded, using `splog:transactionTime` (a `dct:issued` subproperty).

Optionally, the entry *MAY* reference a `splog:InmutableRecord` of its contents and the concrete BPM `splog:Activity` and `splog:Case` involved in the process, if the company maintains this information.

5.1.3 Log entry content

The content, represented by the `splog:LogEntryContent` class, describes the actual data usage using the minimum core model (i.e. data, processing, purpose, storage and recipients) defined in Section 4. This way, event content and data policy authorisations are described with the same class formalization, which facilitates compliance checking. Thus, the `splog:LogEntryContent` class definition *MUST* include the MCM elements using the properties `spl:hasData`, `spl:hasProcessing`, `spl:hasPurpose`, `spl:hasStorage`, `spl:hasRecipient` defined in the SPECIAL policy language.

Example 2. The following example provides a quick overview of how the SPECIAL Policy Log vocabulary might be used to represent a log. We make use of our BeFit scenario: we assume (i) Sue is using a wearable appliance for fitness tracking from BeFit, (ii) the application is tracking the location of Sue for *health* purposes, (iii) a new location is stored in a particular database (called *BeFitDatabaseEurope*) and reflected in the log (called *BeFitLog*). Let us also assume that the namespace for the BeFit company is `befit:` (pointing to the appropriate IRI), being `befit:Us` the main reference of the company. We first show the general log description in Listing 1.

Listing 1: Log description for BeFit devices

```
befit:BeFitLog a splog:Log;
    dct:title              "Log of BeFitDatabaseEurope"@en;
    dct:description        "This contains events on BeFitDatabaseEurope
                            tracking devices geo-located in Europe"@en;
    dct:issued             "2018-02-14"^^xsd:date;
    prov:wasAttributedTo   befit:BeFitDatabaseEurope ;
    splog:processor        befit:Us .
```

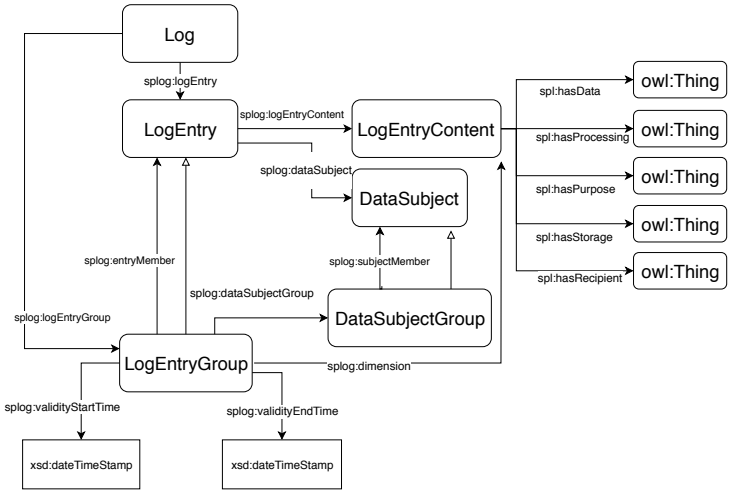

Figure 3: Pictorial summary of log entry grouping

Then, we include a new entry in the log, which is a processing event (uniquely identified as `befit:entry3918`) referencing a new tracking position of Sue, shown in Listing 2. We assume Sue's unique identifier is `befit:Sue`. The collection of the new position took place on the 3rd of January, 2018, at 13:20 (i.e. validity time) and the event was recorded few seconds later (i.e. transaction time).

Listing 2: A new event for Sue's BeFit device

```
befit:BeFitLog splog:logEntry befit:entry3918 .

befit:entry3918 a splog:ProcessingEvent;
    dct:title                "Collection of new device position"@en;
    splog:dataSubject        befit:Sue ;
    dct:description          "We collected a new position of your BeFit
                              device in our database in Europe"@en;
    splog:transactionTime    "2018-01-10T13:20:50Z"^^xsd:dateTimeStamp;
    splog:validityTime       "2018-01-10T13:20:00Z"^^xsd:dateTimeStamp;
    splog:message            "Tracking position by GPS... collected!" ;
    splog:eventContent       befit:content3918 ;
    splog:inmutableRecord    befit:iRec3918 .
```

Note that the log entry `befit:entry3918` is an instance of a `ProcessingEvent`, `befit:iRec3918` links the immutable version of the event, and `befit:content3918` points to the actual content of the event, defined in the following Listing 3.

Listing 3: The content of a new event for Sue's BeFit device

```
befit:content3918 a splog:LogEntryContent;
    dct:description      "Location data are collected by a BeFit device
                          only for the health purpose of the service"@en;
    spl:hasData          svd:Location;
    spl:hasProcessing    befit:SensorGathering;
    spl:hasPurpose       befit:HealthTracking;
    spl:hasStorage       [has:location svl:OurServers];
    spl:hasRecipient     [a svr:Ours].

befit:SensorGathering rdfs:subClassOf svpr:Collect .
befit:HealthTracking rdfs:subClassOf svpu:Health .
```

□

5.2 Grouping Log Entries

Log entries can be grouped to facilitate scalability in those scenarios where there exists a continuous flow of information, such as the envisioned big data applications. For instance, in our BeFit use case, a log

group could be used to represent (as a single entry) the collection of data during a running activity of a data subject in BeFit.

SPECIAL provides such a grouping model, outlined in Figure 3. The core class is `splog:LogEntryGroup` (a subclass of `splog:LogEntry`), which has a validity time interval denoted by the `splog:validityStartTime` and `splog:validityEndTime` properties (subproperties of `prov:startedAtTime` and `prov:endedAtTime`, respectively). The group *MUST* reference the content (data, purpose, processing, etc.) it groups via the `splog:dimension` property (a `splog:logEntryContent` subproperty), which points to a particular `splog:LogEntryContent`. The group *MAY* reference the data subject(s) in the group (all sharing the same log entry content), using the property `splog:dataSubjectGroup` (`prov:wasAssociatedWith` subproperty). This property points to a `splog:DataSubjectGroup` instance that groups all the data subject members in the group via `splog:subjectMember` (a `skos:member` subproperty). Finally, the group *MAY* point to the particular entries included in the group through the `splog:entryMember` property (a `skos:member` subproperty). This option can facilitate a fine-grained traceability at the cost of storing additional information (i.e. all log entries of the group), hence it is an optional feature.

Example 3. The following example in Listing 4 shows a log grouping the category of recommendations given to Sue, John and Rose during a month.

Listing 4: A grouping example merging all recommendations given in a month

```
befit:BeFitLog a splog:Log ;
    splog:logEntryGroup befit:recommendationsJanuary2018 .

befit:recommendationsJanuary2018 a splog:logEntryGroup
    splog:transactionTime    "2018-02-01T00:05:00Z"^^xsd:dateTimeStamp;
    splog:validityTime       "2018-01-31T23:59:59Z"^^xsd:dateTimeStamp;
    splog:dataSubjectGroup   befit:basicSubjectGroup;
    splog:dimension          befit:templateOfferRecommendation .

befit:basicSubjectGroup splog:member befit:Sue, befit:John, befit:Rose.

befit:templateOfferRecommendation a splog:LogEntryContent ;
    spl:hasData              befit:OfferRecommendation;
    spl:hasProcessing        befit:MonthlyDataAnalysis;
    spl:hasPurpose           befit:MonthlyOffersRecommendation;
    spl:hasStorage           [has:location svl:OurServers];
    spl:hasRecipient         [a svr:Ours].

befit:OfferRecommendation rdfs:subClassOf svd:Location;
    rdfs:comment  "We recommended you an offer at the end of the month
                   based on the location of your device" .

befit:MonthlyDataAnalysis rdfs:subClassOf svpr:Analyze .
befit:MonthlyOffersRecommendation rdfs:subClassOf befit:RecommendationActivity .
befit:RecommendationActivity rdfs:subClassOf svpu:Marketing .
```

6 SPECIAL Transparency and Compliance

First we provide an overview of SPECIAL compliance checking. Following on from this we provide a high level overview of the SPECIAL system architecture.

6.1 Using SPECIAL Resources for Compliance Checking

Policies and log events are described in semantically unambiguous terms aligned to the same taxonomies that are used to define usage policies, hence facilitating transparency and automatic compliance checking. Regarding this latter, the usage policy enforced by a data controller contains the operations that are permitted within the data controller's organization. Therefore, the usage U_c attached to a SPECIAL log entry *complies* with the usage policy P_s in the data subject's consent if and only if all the authorizations in U_c are also authorized by P_s, that is, U_c complies with P_s if and only if $U_c \subseteq P_s$. Thus, in OWL 2 terminology, this amounts to checking whether the following axiom is *entailed* (implied) by the combined ontology $\emptyset$ containing the SPECIAL policy language ontology plus the aforementioned auxiliary vocabularies: `SubClassOf(`U_c P_s`)`. This is inherently supported by general inference engines for OWL 2 (e.g. HermiT and FaCT++).

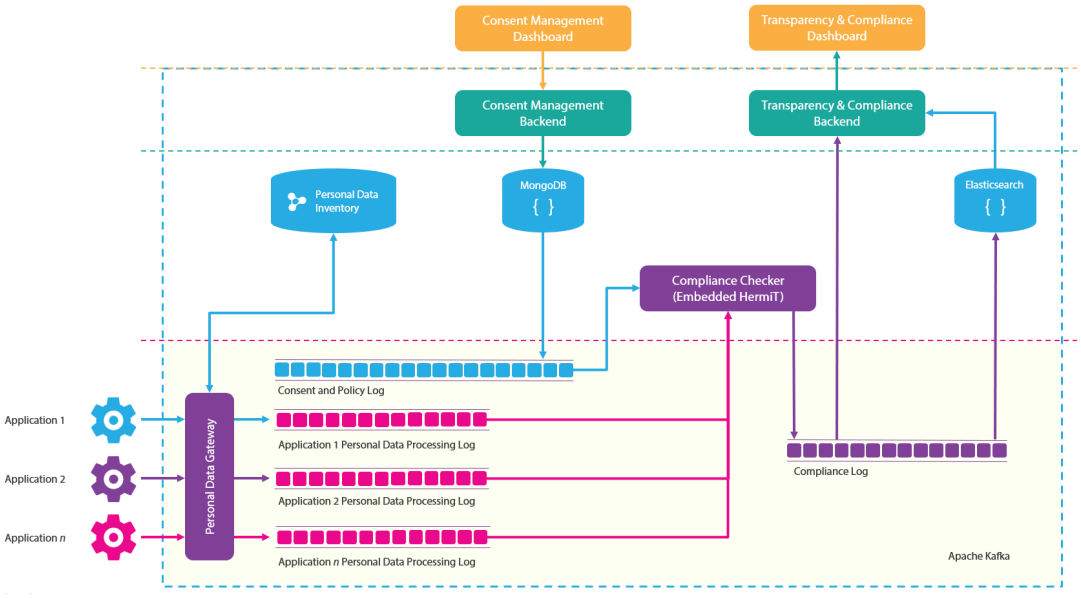

Figure 4: SPECIAL-K architecture setup for ex post compliance checking

For instance, the log entry in Example 2 specifies that there was a process of type `befit:SensorGathering` on location data. This entry is compliant with a potential usage policy stating that the controller can collect (`svpr:Collect`) location data *iff* `befit:SensorGathering` is a sublass of `svpr:Collect`.

In addition to the ex-ante compliance checking (based on event logs) described above, the SPECIAL platform also caters for ex-post compliance checking (based on business rules) of existing Line of Business and Business Intelligence / Data Science applications. In turn, the content of the SPECIAL events could potentially be described at different granularities, from categorising the content in a simple taxonomy stating the type of data, processing, etc., involved in the event, to the most fine-grained description of the actual data associated to the event (e.g. concrete location of a data subject). We assumed the log entries store categories (classes) such that compliance checking is based on the aforementioned class subsumption. Thus, we consider that actual data can be stored, linked and retrieved from an alternative data source.

6.2 The SPECIAL-K Architecture

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). 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* (*D2.5 Policy Language V2*);

(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 (D2.7 Transparency Framework V2); and

(iii) *The SPECIAL Compliance Component* focuses on demonstrating that data processing and sharing complies with usage control policies (*D2.8 Transparency and Compliance Algorithms V2*).

The SPECIAL system architecture is depicted in Figure 4. This paper specifically focuses on evaluating the scalability and robustness of the SPECIAL transparency and compliance components.

Table 2: Transparency and compliance services.

Component	Functionalities	Current support in SPECIAL platform (third release)
Transparency component	List the data processing and sharing events that 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-post and ex-ante compliance checking	Total
	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)

SPECIAL Transparency Component. Data processing and sharing event logs are stored in the Kafka[11] distributed streaming platform. A Kafka topic is used to store application logs, while a separate compliance topic (called *Compliance Log*) is used to store the enriched log after compliance checks have been completed. As logs can be serialised using JSON-LD, it is possible to benefit from the faceting browsing capabilities of Elasticsearch[12], and the out of the box visualisation capabilities provided by Kibana.

Compliance Checker. The compliance checker, which currently includes an embedded HermiT[13] reasoner uses the consent saved in MongoDB, 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.

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, optimise the performance and guide future directions.

7 The SPECIAL Benchmark

In this section we present the choke points used to identify technical difficulties that the benchmark should consider in order to challenge the system under test (our SPECIAL platform); provide details on the benchmark data generation; and outline relevant key performance indicators (KPIs); and introduce the STC benchmark tasks.

7.1 Choke Point-based Benchmark Design

We design `STC-bench` following the same methodology as most of the benchmarks under the H2020 HOBBIT project [18]. Thus, the development of the benchmark is driven by so-called "choke-points", a notion introduced by the Linked Data Benchmark Council (LDBC) [8, 1]. A choke-point analysis aims to identify important technical challenges to be evaluated in terms of query workload. 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 following transparency and compliance choke points:

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

[11] https://kafka.apache.org/
[12] https://www.elastic.co/products/elasticsearch
[13] http://www.hermit-reasoner.com/

scalability, *performance*, and *responsiveness*. 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.

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*, *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).

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 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.

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.

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.

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.

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.

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

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

7.2 Data Generation

In the following we present the `STC-bench` data generator to test the compliance and transparency performance of the SPECIAL platform. The data generation considers two related concepts: the usage policies and the data sharing and processing events that are potentially compliant with 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.

Table 3: 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

In this benchmark, we focus on the second alternative, providing a synthetic data generator following the BeFit scenario. 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. The `STC-bench` data generator supports the following configuration parameters:

- *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 serialisation 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.

7.3 Benchmark Tasks

In the following we present the set of concrete benchmark tasks for the SPECIAL compliance and transparency components. We establish here a set of simple tasks to be performed by the SPECIAL transparency component. The transparency tasks are illustrated in Table 4. 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 3.

In turn, Table 5 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 20 minutes, which delimits the number of events generated given the number of users and event generation rate in each case.

7.4 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):

- *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.

Table 4: 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 5: 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.	20 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.	20 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.	20 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.	20 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

- *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 nodes in the system.
- *Memory Usage by Node:* The average memory usage by nodes in the system.
- *Disk Space:* The total disk space used in the system.

8 Evaluation

The evaluation described in this section focuses 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. We start by describing a first analysis of scaling

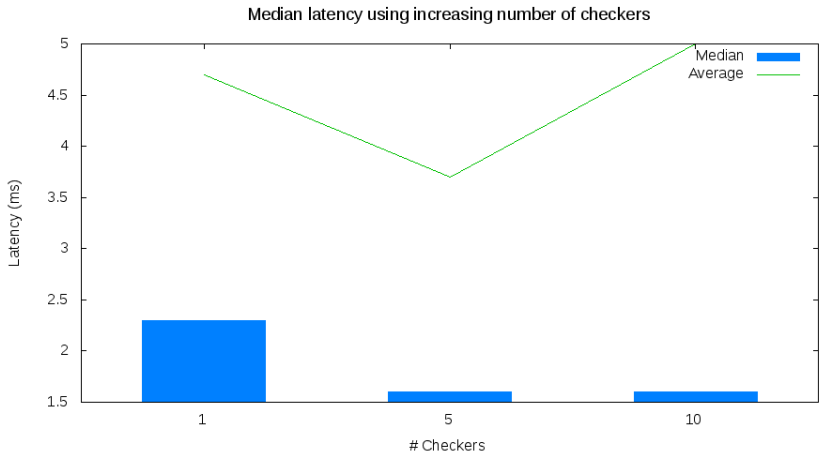

Figure 5: Median and average latencies with increasing number of compliance checkers

the number of compliance checking processes. Following on from this we present the results on the aforementioned `STC-bench` compliance tasks.

We report the averaged results of 3 independent executions. All experiments were executed on a cluster consisting of 10 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:* 10.
- *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 2023.5.0 (Rhyolite).
- *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.

8.1 Scaling the Compliance Checking Process

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

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 and batch processing scenario.

8.2 Streaming

For this scenario, we evaluate the streaming task *C-T4-4* from `STC-bench`, shown in Table 5. Note that the task considers a stream of 1,000 users, where each user generates 1 event every second. That is, we evaluate an event stream that, on average, generates *1 event every 1ms*, producing a total of 1,200,000

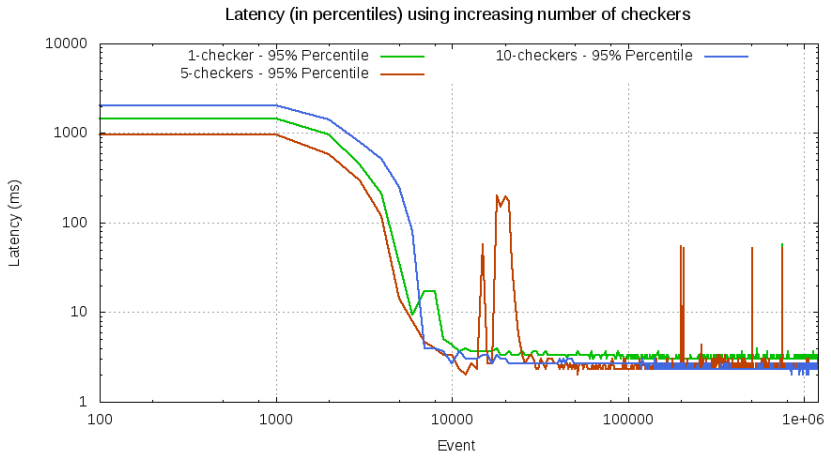

Figure 6: Latencies (in 95% percentile) with increasing number of compliance checkers (1, 5, 10 checkers)

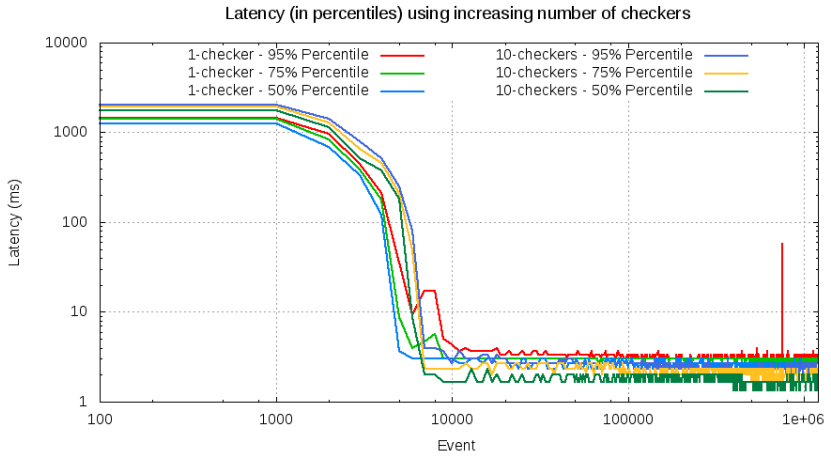

Figure 7: Latencies (in 95%, 75% and 50% percentile) with increasing number of compliance checkers (1, 10 checkers)

events. Given that we expect a performance on the level of ms per check, the streaming flow is close to the limit of one compliance checker.

Figure 5 shows the median and average latency (in milliseconds, with logarithm scale) with different number of compliance checkers in parallel, ranging from 1 to 10 (with 10 being 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 show that the (median) latency is always at the level of milliseconds (in particular, less than 2.5 ms), with a noticeable improvement when more compliance checkers are running in parallel, providing a *stable latency of 1.5 ms*. As expected, the slightly higher average figures denote 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 6 represents (in milliseconds and logarithm scale) the latency at 95% percentile, using 1, 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 an initial *warm-up* effect, with higher latencies until the first 10,000 events. Then, the latencies are stable with 1-2 ms in all cases, even at the high streaming rate of 1 event every ms. That is, in general, only 5% of the events can experience latencies over 1-2 ms. As expected, latencies are slightly greater if only 1 checker is used. It is worth noting that the evaluation uncovered a recurrent peak with 5 checkers around 20,000 events, which is subject of future inspection.

Figure 7 completes this analysis, depicting 50, 75 and 95% percentiles for the extreme cases of having

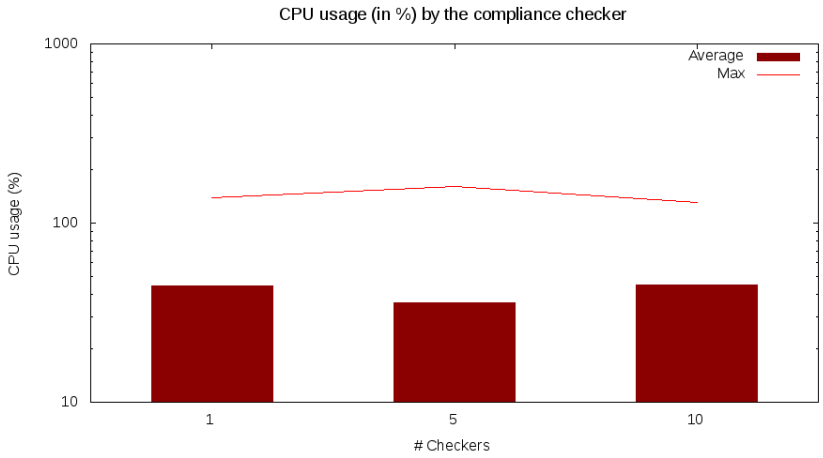

Figure 8: CPU usage (in %) with increasing number of compliance checkers

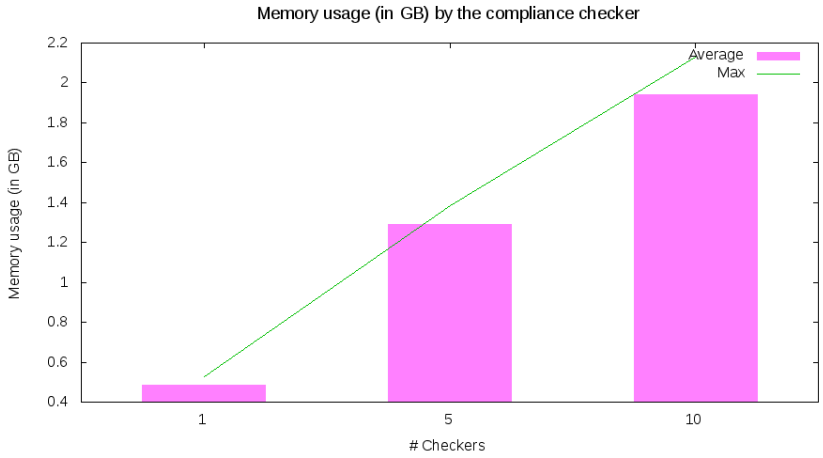

Figure 9: Memory usage (in GB) with increasing number of compliance checkers

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, 5 and 10), shown in Figures 8 and 9 respectively. We report the average and the maximum number. Results show that, thanks to latest improvements in the third release of the platform, (i) memory usage increases sublinearly (and remains under 2 GBs) as more parallel compliance checkers are running in parallel, and (ii) CPU consumption remains stable around 50%, with no major influence of the number of checkers. Both results show that Kafka is able to optimise the use of resources and to adapt to the number of parallel checkers. In addition, it is worth mentioning that Kafka is able to add compliance checkers dynamically.

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 1-2ms with an event stream of 1 event every 1ms, 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.

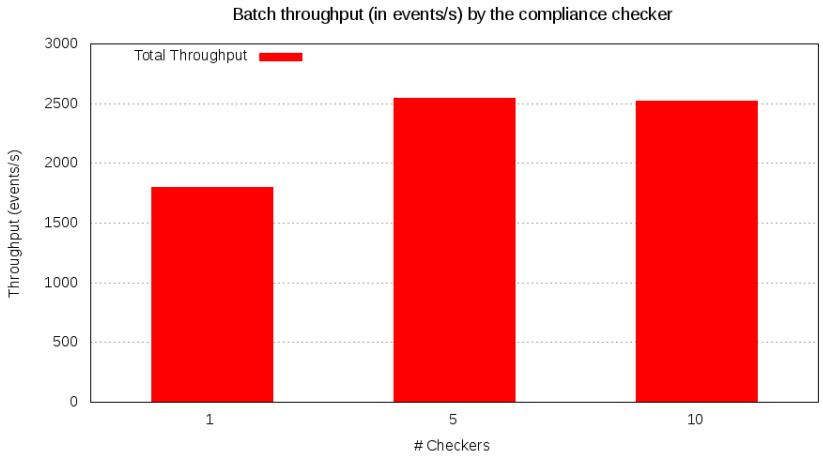

Figure 10: Total batch throughput (in events/s) by the compliance checker with increasing number of compliance checkers

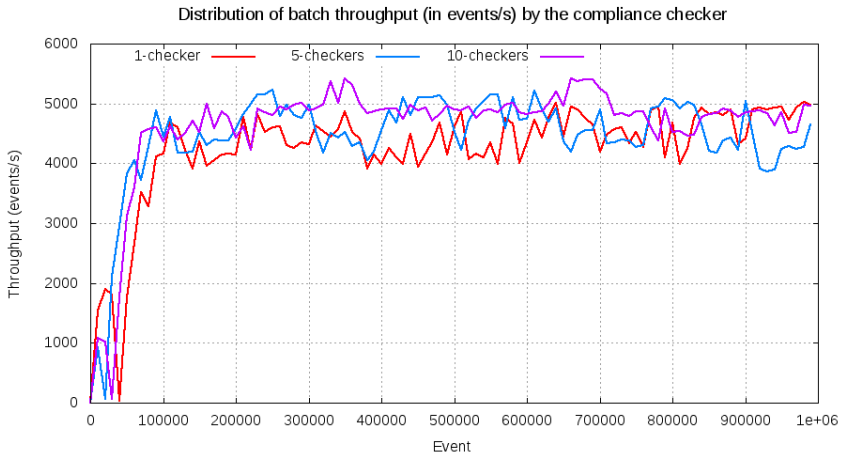

Figure 11: Distribution of batch throughput (in events/s) by the compliance checker with increasing number of compliance checkers

8.3 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-T5-2* from `STC-bench`, shown in Table 5. This task considers 1,000,0000 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 10 shows the total batch throughput (in events/s) for 1, 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 behaviour, where the throughput ranges from 1796 events/s with 1 checker to 2523 events/s with 10. The difference between 5 and 10 checkers is negligible.

Figure 11 shows the distribution of batch throughput (in events/s) across time, for 1, 5 and 10 compliance checkers. Results are consistent with the throughput reported above, showing a general constant behaviour and a better performance with 5 and 10 checkers running in parallel.

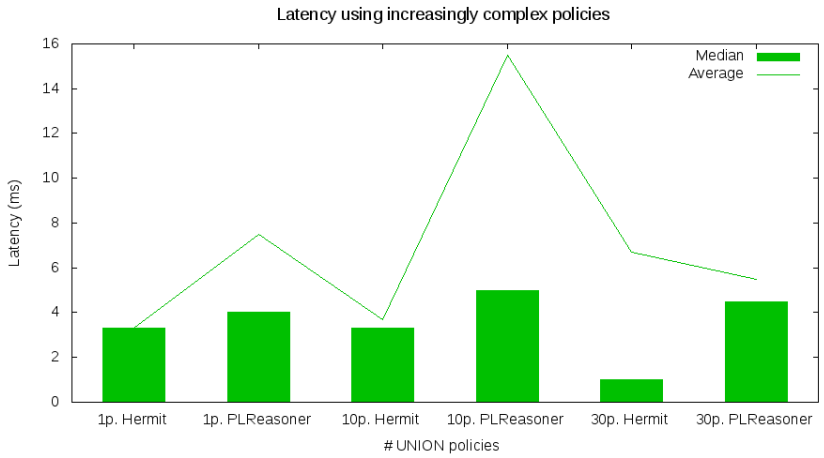

Figure 12: Median and average latencies with increasing complex policies

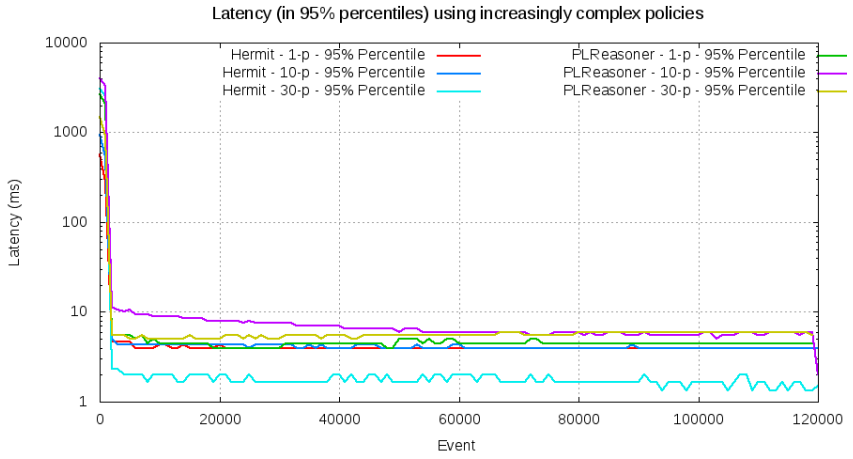

Figure 13: Latencies (in 95% percentile) with increasing complex policies

Table 6: Space requirements (MB) with increasing generation rate.

# Users	Event Rate (per user)	# Events	Disk Space (MB)
1,000	1 ev./60s	20,000	819
1,000	1 ev./30s	40,000	1,563
1,000	1 ev./10s	120,000	1,954
1,000	1 ev./1s	1,200,000	5,355
1,000	1 ev./100ms	12,000,000	59,664

8.4 Results on STC-bench Compliance Tasks

This section provides results on the `STC-bench` tasks, shown in Section 7. Rather than showing a complete evaluation on an optimised 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 functionality provided by the current third release of the SPECIAL platform and the scaling capabilities of the infrastructure. In the following we present the results for all the compliance tasks (*C-T1* to *C-T5* from Table 5). We disregard *C-T3* as no significant differences were found in our tests and we opt for a more realistic random generation of policies.

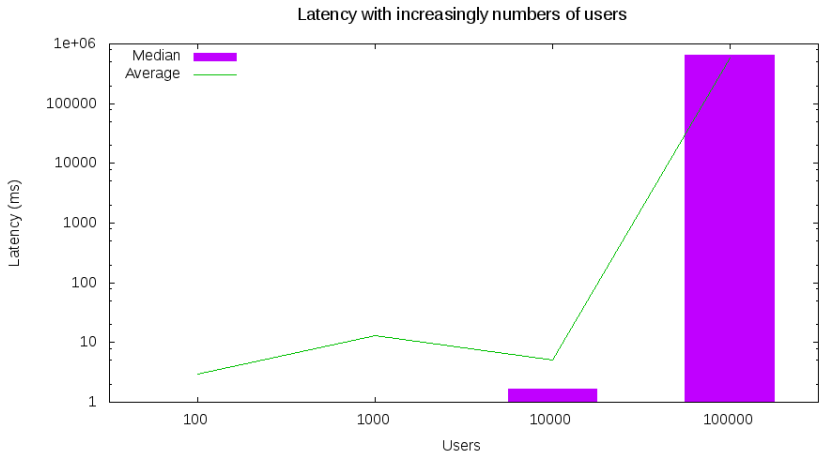

Figure 14: Median and average latencies with increasing number of users

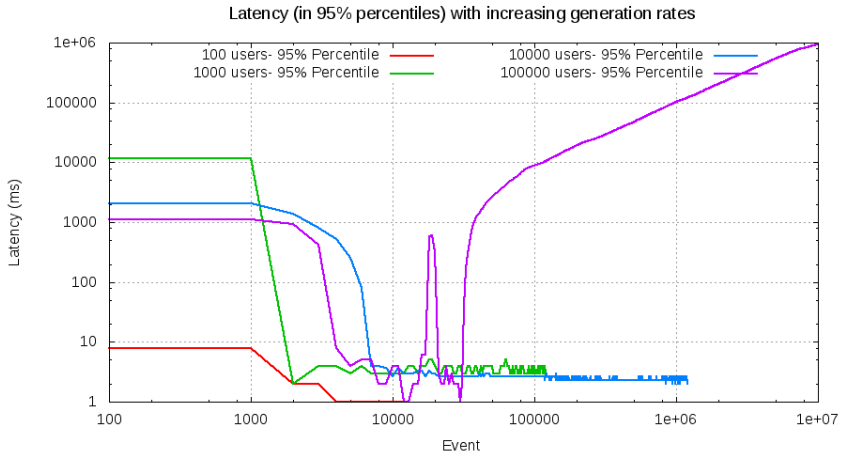

Figure 15: Latencies (in 95% percentile) with increasing number of users

8.4.1 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 1 compliance checker in order to isolate the performance of one instance. We also compare the implementation of the Hermit reasoner with our engine PLReasoner.

Figure 12 shows the median and average latencies (in milliseconds) with 1, 10 and 30 union policies. Results show that the median *latency ranges between 1.5-5 ms*, with relatively small differences as the number of union policies grows, except for the union of 30 policies. In this case, the higher number of union policies allows Hermit to quickly find a match (1.5 ms). As for the comparison of reasoners, Hermit seems to slightly outperform PLReasoner in the scenario under test. Nonetheless, when both reasoners are run in isolation, the tailored PLReasoner engine is several times faster than Hermit [9]. A first analysis shows that different parsing and deserialisation of policies can affect the times of PLReasoner in the SPECIAL platform. In addition, our isolated study generally considers richer and more complex policies than STC-bench, also including different time intervals (for the duration of the storage).

Finally, the higher figures for the average latency again denote a skewed distribution. Thus, we inspect the latency at 95% percentile (the value at which 95% of the data is included), depicted in Figure 13 for 1, 10 and 30 policies. The distribution shows that, in all scenarios, the latency at 95% percentile is stable after the warm-up, with small differences with more union policies. Results also show

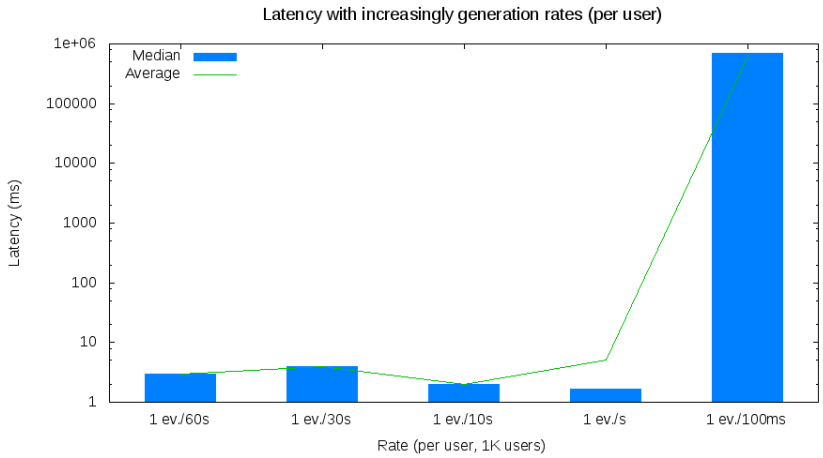

Figure 16: Median and average latencies with increasing generation rates. The rate refers to events per user, with 1K users

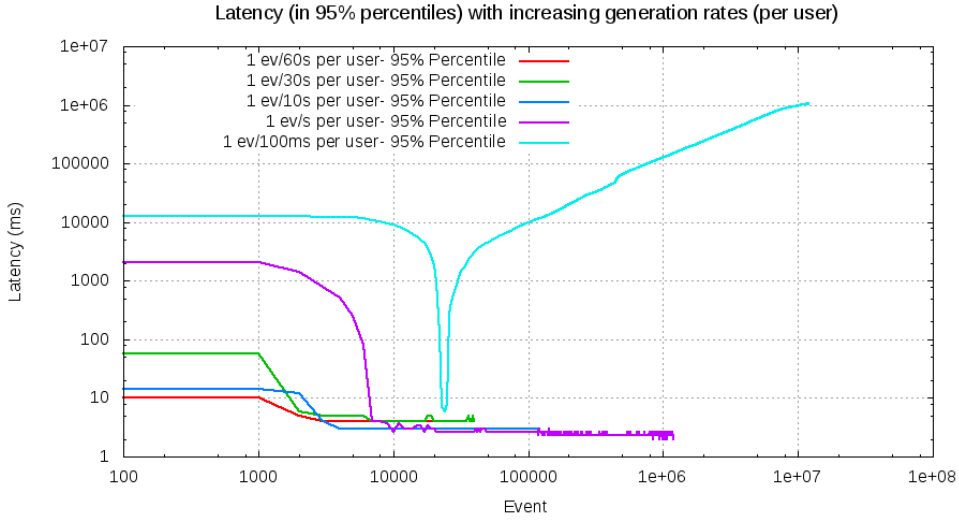

Figure 17: Latencies (in 95% percentile) with increasing generation rates. The rate refers to events per user, with 1K users

that only 5% of the events can experience latencies over 5 ms.

8.4.2 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 study the first four subtasks, covering up to 100,000 users given the characteristics of the experimental infrastructure. Note that serving 100,000 users at the aforementioned rate already implies to manage a stream of 10,000 events every second. In this scenario, we consider 10 compliance checkers 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 14 shows the median and average latencies for 100-100,000 users. Results show that the system is able to provide a median latency of less than 1ms with 1,000 users (each user with 1 event every 10 seconds, hence overall the system receives 1 event every 10 ms simultaneously), and 1.6ms with 10,000 users (overall, 1 event very ms). However, with 100,000 users, the current infrastructure needs to manage

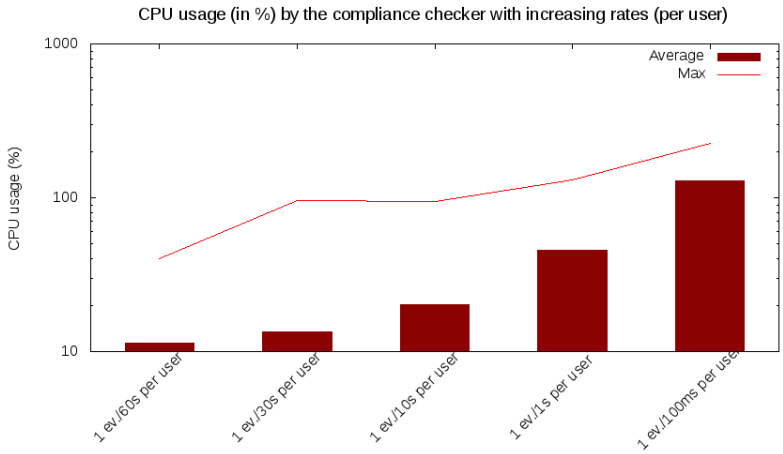

Figure 18: CPU usage (in %) for compliance checking with increasing generation rate (1K users)

1 event every 0.1ms (less than the checking time of 1ms), which causes delays of several seconds.

In order to highlight potential worst-case scenarios, we represent the latency at 95% percentile in Figure 15. Note that an increasing number of users results in more events, hence the different number of events in each scenario. As expected, results show two different scenarios. On the one hand, a number of users between 100-10,000 results in a *95% percentile around 1 ms*, with an initial warm-up step that produces higher latencies. On the other hand, a higher number of users (100,000) 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 10,000 events every second. Given that Kafka is able to optimise and adapt to the number of parallel checkers, which is solely limited by the number of partitions in the cluster, hence a more powerful infrastructure could cope with a greater number of users.

8.4.3 C-T4: Increasing Data Generation Rates

This task evaluates the performance of the system with increasing streaming rates. We consider 10 compliance checkers running in parallel in order to try to cope with the biggest rates in the defined tasks.

Figure 16 represents the median and average latencies (in milliseconds and logarithm scale), while the latency at 95% percentile is shown in Figure 17 (in logarithm scale). Several comments are in order. First, note that the median values in Figure 16 are consistent with our previous latency measures, obtaining values between 1-2ms for rates up to 1 ev/s (per user). Then, as expected, the median latency increases up to several seconds at the highest rate of 1 ev/100ms per user, that is, the system receives a total of 1 ev/0.1ms.

The huge skewed distribution for the highest rate is revealed by the 95% percentile shown in Figure 17. Note that we fix the benchmark time at 20 minutes, so more events are generated with increasing generation rates. Results shows that, the latency reaches a stable stage for rates up to 1 ev/1s per user, i.e. a total of 1 ev/1ms. In contrast, the latency at 95% percentile grows steadily for streams at 1 ev/100ms per user. This fact shows that the current installation cannot cope with such high rates and new events have to queue until they can be processed. The maximum latency reaches 17 minutes for 12 million events.

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 18. As expected, the results show that the CPU usage increases (but sublinearly) with the generation ratio. The disk space requirements are given in Table 6. 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 show the log compaction feature of Kafka as, on average, less bytes are required to represent each of the events with increasing

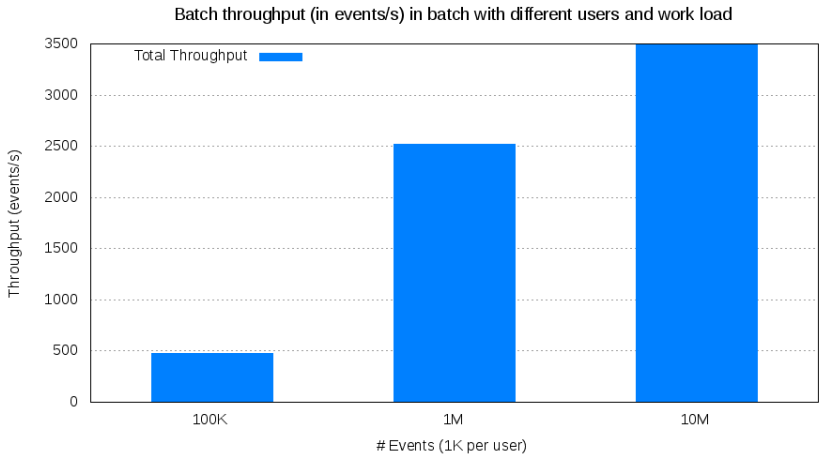

Figure 19: Total batch compliance checking throughput (in events/s) with increasing number of compliance checkers

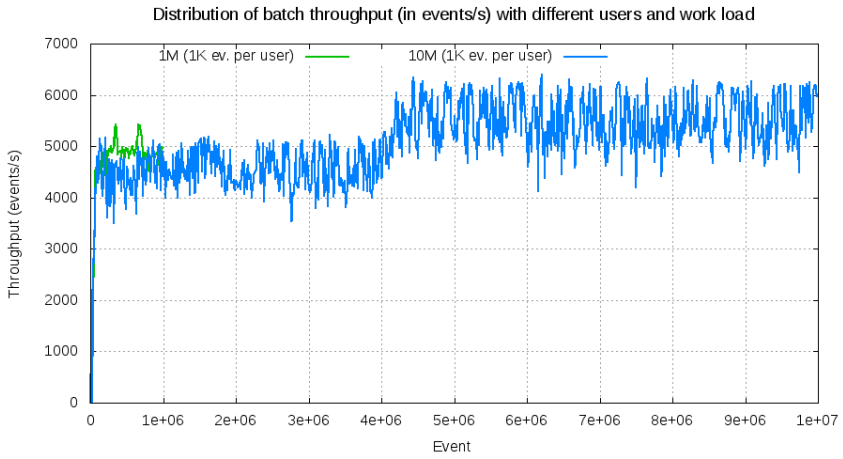

Figure 20: Distribution of batch compliance checking throughput (in events/s) with different users and work load. We consider 1000 events per user

event rates.

8.4.4 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 the first three subtasks, testing up to 10 million events[14] (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 19 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 474 processed events/s in all cases, with a maximum of 3,489 events/s in the case of 10M events.

Finally, Figure 20 looks at the distribution of the throughput for the case of 1M and 10M events. Both cases shows similar initial figures, with increased performance around 4M events.

[14]Note that the system is able to generate and process an arbitrary number of events in batch. Further results can be found in our companion website.

9 Conclusions

In this paper we introduced the initial suite of ontologies and vocabularies, developed within the SPECIAL project, that can be used by companies to record usage policies and data processing and sharing events in the manner that enables the compliance of existing Line of Business and Business Intelligence applications to be checked automatically. In addition to providing an overview of the resources and how they fit into the SPECIAL consent, transparency and compliance architecture, we also described how our the SPECIAL-K Apache Kafka based big data platform can be used for automatic compliance checking. We also proposed a synthetic benchmark for transparency and compliance, referred to as STC-bench, which is designed on the basis of well-identified choke points (challenges) that could affect the performance of SPECIAL-K and similar systems. Finally, we used STC-bench to evaluate the SPECIAL-K compliance checking, using synthesised data.

Future work includes refining the vocabularies based on additional use cases and demonstrating their effectiveness in various business settings.

Acknowledgments

Supported by the European Union's Horizon 2020 research and innovation programme under grant 731601 and the Austrian Research Promotion Agency (FFG): grant 861213 (CitySpin). The authors are grateful to all of SPECIAL's partners; without their contribution this project and its results would not have been possible.

References

[1] Renzo Angles, Peter Boncz, Josep Larriba-Pey, Irini Fundulaki, Thomas Neumann, Orri Erling, Peter Neubauer, Norbert Martinez-Bazan, Venelin Kotsev, and Ioan Toma. The linked data benchmark council: a graph and rdf industry benchmarking effort. *ACM SIGMOD Record*, 43(1):27–31, 2014.

[2] Mihir Bellare and Bennet Yee. Forward integrity for secure audit logs. Technical report, Computer Science and Engineering Department, University of California at San Diego, 1997.

[3] P.A. Bonatti, S. Kirrane, I. Petrova, L. Sauro, and E. Schlehahn. Special deliverable 2.5: Policy language v2, 2018.

[4] Piero Bonatti, Sabrina Kirrane, Axel Polleres, and Rigo Wenning. Transparent personal data processing: The road ahead. In *International Conference on Computer Safety, Reliability, and Security*, pages 337–349. Springer, 2017.

[5] Piero A. Bonatti, Juri Luca De Coi, Daniel Olmedilla, and Luigi Sauro. A rule-based trust negotiation system. *IEEE Trans. Knowl. Data Eng.*, 22(11):1507–1520, 2010.

[6] Lorrie Faith Cranor. *Web privacy with P3P - the platform for privacy preferences*. O'Reilly, 2002.

[7] Marina De Vos, Sabrina Kirrane, Julian Padget, and Ken Satoh. Odrl policy modelling and compliance checking. In *International Joint Conference on Rules and Reasoning*, pages 36–51. Springer, 2019.

[8] Orri Erling, Alex Averbuch, Josep Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat, Minh-Duc Pham, and Peter 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.

[9] Javier D. Fernández, P.A. Bonatti, U. Milosevic, and Jonathan Langens. Special deliverable 3.5: Scalability and robustness testing report v2, 2018.

[10] Renato Iannella and Serena Villata. Odrl information model 2.2. W3C Recommendation, 2018.

[11] Information Commissioner's Office (ICO) UK. Getting ready for the GDPR, 2017.

[12] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 63–74. IEEE, 2003.

[13] Vladimir Kolovski, James Hendler, and Bijan Parsia. Analyzing web access control policies. In *Proceedings of the 16th international conference on World Wide Web*, pages 677–686, 2007.

[14] T. Lebo, S. Sahoo, and D. McGuinness. Prov-o: The prov ontology. *W3C Recommendation, April*, 2013.

[15] Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil MP van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information systems*, 54:209–234, 2015.

[16] Microsoft Trust Center. Detailed GDPR Assessment, 2017.

[17] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language – Structural Specification and Functional-Style Syntax (Second Edition). W3C Recommendation, 2012.

[18] Axel-Cyrille Ngonga Ngomo and Michael Röder. Hobbit: Holistic benchmarking for big linked data. *ERCIM News*, 2016(105), 2016.

[19] Nymity. GDPR Compliance Toolkit.

[20] Harshvardhan J Pandit, Kaniz Fatema, Declan O'Sullivan, and Dave Lewis. Gdprtext-gdpr as a linked data resource. In *European Semantic Web Conference*, pages 481–495. Springer, 2018.

[21] Harshvardhan J Pandit, Axel Polleres, Bert Bos, Rob Brennan, Bud Bruegger, Fajar J Ekaputra, Ramisa Gachpaz Hamed, Elmar Kiesling, Mark Lizar, Eva Schlehan, et al. Creating a vocabulary for data privacy: the first-year report of data privacy vocabularies and controls community group (dpvcg). 2019.

[22] Tobias Pulls, Roel Peeters, and Karel Wouters. Distributed privacy-preserving transparency logging. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 83–94, 2013.

[23] Mikko Rinne, Eva Blomqvist, Robin Keskisärkkä, and Esko Nuutila. Event processing in rdf. In *Proc. of WOP-Volume 1188*, 2013.

[24] Stefan Sackmann, Jens Strüker, and Rafael Accorsi. Personalization in privacy-aware highly dynamic systems. *Communications of the ACM*, 49(9), 2006.

[25] Reza Samavi and Mariano P Consens. Publishing privacy logs to facilitate transparency and accountability. *Journal of Web Semantics*, 50:1–20, 2018.

[26] Andrew Sutton and Reza Samavi. Blockchain enabled privacy audit logs. In *International Semantic Web Conference*, pages 645–660. Springer, 2017.

[27] Andrzej Uszok, Jeffrey M. Bradshaw, Renia Jeffers, Niranjan Suri, Patrick J. Hayes, Maggie R. Breedy, Larry Bunch, Matt Johnson, Shriniwas Kulkarni, and James Lott. KAoS policy and domain services: Towards a description-logic approach to policy representation, deconfliction, and enforcement. In *Proc. of POLICY*, pages 93–96, 2003.

[28] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184. IEEE, 2015.