Scaling Data Science Solutions with Semantics and Machine Learning: Bosch Case

Baifan Zhou[1,2,*], Nikolay Nikolov[3,1,*], Zhuoxun Zheng[4,1], Xianghui Luo[5], Ognjen Savkovic[6], Dumitru Roman[3,1], Ahmet Soylu[2], and Evgeny Kharlamov[4,1]

[1] Department of Informatics, University of Oslo, Norway
[2] Department of Computer Science, Oslo Metropolitan University, Norway
[3] SINTEF AS, Norway
[4] Bosch Center for Artificial Intelligence, Germany
[5] ACM Member, Germany
[6] Department of Computer Science, Free University of Bozen-Bolzano, Italy

Abstract. Industry 4.0 and Internet of Things (IoT) technologies unlock unprecedented amount of data from factory production, posing big data challenges in volume and variety. In that context, distributed computing solutions such as cloud systems are leveraged to parallelise the data processing and reduce computation time. As the cloud systems become increasingly popular, there is increased demand that more users that were originally not cloud experts (such as data scientists, domain experts) deploy their solutions on the cloud systems. However, it is non-trivial to address both the high demand for cloud system users and the excessive time required to train them. To this end, we propose SemCloud, a semantics-enhanced cloud system, that couples cloud system with semantic technologies and machine learning. SemCloud relies on domain ontologies and mappings for data integration, and parallelises the semantic data integration and data analysis on distributed computing nodes. Furthermore, SemCloud adopts adaptive Datalog rules and machine learning for automated resource configuration, allowing non-cloud experts to use the cloud system. The system has been evaluated in industrial use case with millions of data, thousands of repeated runs, and domain users, showing promising results.

Keywords: ontology engineering · knowledge graph · semantic ETL · machine learning · cloud computing · welding · quality monitoring · Industry 4.0· rule-based reasoning · Datalog

1 Introduction

Background. Industry 4.0 [1] aims at highly automated smart factories that rely on IoT technology [2], spanning across data acquisition, communication, information processing and actuation. This has unlocked unprecedented amounts

*Baifan Zhou and Nikolay Nikolov contributed equally to this work as first authors.
baifanz@ifi.uio.no, nikolay.nikolov@sintef.no

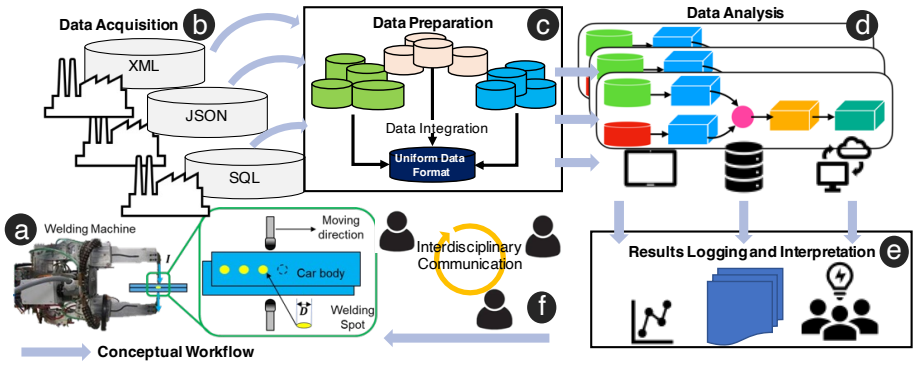

Fig. 1. Data analytics development cycle exemplified on the Bosch case of welding condition monitoring. In industrial data science projects, many users are non-cloud experts (e.g., welding experts, ML experts) and want to scale their solutions on the cloud.

of data that are generated by production systems [3] and, thus, drastically increased the demand for data-driven analytical solutions and cloud technology. We illustrate a common industrial scenario of development and deployment of data-driven solutions on cloud with a Bosch welding case[7] of quality monitoring in Fig. 1: The data from a production environment such as welding machines (a) has first to be acquired in different formats, e.g., CSV, JSON, XML (b); then they should be integrated into a uniform format (c); After that, the project team (including welding experts, data scientists, managers. etc.) wants to run data analysis on cloud infrastructures on top of the large data volumes from many factories (d); After data analysis, these users need to discuss and log the results (e); The whole process involves iterative and cross-domain communication between the stakeholders (f).

Challenges. From the scenario, we see that scaling data science solutions poses challenges related to dealing with the high data *volume*, *variety*, and more *users*, namely enabling non-cloud experts to leverage cloud systems. Indeed, industries equipped with IoT technologies produce huge volumes of production data. In the Bosch case, one factory alone produces more than 1.9 million welding records per month. The data generated by different software versions, locations, customers have a variety of data formats, feature names, available features, etc. Meanwhile, many users that are not cloud experts, such as domain experts and data scientists, want to deploy the data science solutions on the cloud. In a standard implementation of the workflow in Fig. 1, the project team requires extensive assistance from cloud experts, whenever they want to deploy solutions or make small changes to their solutions deployed on the cloud. To facilitate the adoption of cloud systems for more projects and users, one can equip all projects

[7] Automated welding is an impactful manufacturing process that is involved in the production of millions of cars annually, deployed world-wide at many factories. Data-driven analytics solutions for welding can greatly help in reducing the cost and waste in production quality. Errors in production can only be resolved by destroying newly produced cars in samples.

with some cloud experts, or launch training programs about cloud technology for all users. Both require careful planing to balance time, cost, and benefits.

Our Approach. To address these scalability challenges in terms of data volume, data variety, and democratising cloud systems, we propose SemCloud: a semantics-enhanced cloud system, that scales semantic data integration and data analysis on the cloud with distributed computing, and allows non-cloud experts to deploy their solutions. Our system is motivated by a use case at Bosch aiming at scaling data science solutions in welding condition monitoring (Sect. 2). SemCloud consists of semantic artefacts such as domain ontologies, mappings, adaptive Datalog rules (Sect. 3) and *machine learning* (ML) that learns the parameters in the adaptive Datalog rules.

In particular, the semantic data integration (extract-transform-load, ETL) (Sect. 3.2) maps diverse data sources to a unified data model and transforms them to uniform data formats. To allow distributed ETL, SemCloud slices the integrated data according to domain-specific data semantics (machine equipment identifiers in the Bosch case), separating the data into computationally independent subsets. SemCloud then parallelises the ETL and analysis of the data slices on distributed computing nodes (Sect. 3.3). Furthermore, SemCloud adopts a semantic abstraction and a graphical user interface (GUI) to democratise cloud deployment, improving transparency and usability for non-cloud experts. These include a cloud ontology that allows to encode ETL pipelines in knowledge graphs (Sect. 3.4), and a set of adaptive Datalog rules (Sect. 3.5) for automatically finding optimal resource configurations. These rules are adaptive because some of their predicates are functions learnt with machine learning (ML)(Sect. 3.6).

We note that the existing work on this topic addressed the cloud deployment issues only to a limited extent [4, 5], whereby they either only focus on the formal description of cloud, or on the limited adaptability of cloud systems. SemCloud exploits and significantly extends our previous works on ML in the context of Industry 4.0 [6, 7], and container-based big data pipelines [8] (Fig. 4) by enhancing the framework with semantic artefacts and modules for specifying container-based pipelines, including pipeline step templates for containerisation and management of inter-step communication and data transmission (Sect. 3.3).

We evaluated (Sect. 4) SemCloud extensively: the cloud deployment report to verify SemCloud performance on reducing computational time, with an industrial datasets of about 3.1 million welding spots; the performance of rule parameter learning and inference based on 3562 times of repeated runs of the system.

2 Motivating Use Case: Welding Quality Monitoring

In this section we discuss our motivating use case in more details, explain why scaling data science solution to large data sets and more users is critical and discuss requirements for the cloud system.

Condition monitoring for automated welding. Condition monitoring refers to a group of technologies for monitoring condition parameters in production machinery to identify potential developing faults [9]. The use case addresses

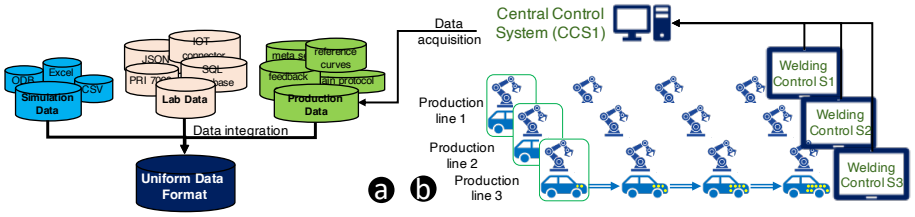

Fig. 2. (a) The data variety issue. (b) The data volume issue exemplified with the production data.

one type of condition monitoring, quality monitoring (another type is machine health monitoring), for resistance spot welding at Bosch, which is a fully automated welding process that is essential for producing high-quality car bodies globally in the automotive industry. During the welding process (Figure 1a), two welding gun electrodes press two or three worksheets (car body parts) with force, an electric current flows through the electrodes and worksheets, generating a large amount of heat due to electric resistance. The materials in a small area between the two worksheets melt and then congeal after cooling, forming a weld nugget (or welding spot) connecting the worksheets. The core of quality monitoring is to measure, estimate or predict some categorical or numerical quality indicators. The diameter (Figure 1a) of welding spots is typically used as the quality indicator of a single welding act according to industrial standards [9, 10]. Conventional practice adopts destructive methods to tear the welded car bodies apart, although they can only be applied to a small sample of car bodies, and the destroyed samples are waste and cannot be reused. Bosch is developing data-driven solutions to predict the welding quality, to reduce the waste and improve the coverage of quality monitoring.

Bosch big data. Welding condition monitoring faces big data challenges of variety and volume. In terms of *data variety*, Bosch has many data sources of different locations and conditions (Figure 2a). The production data alone are collected from at least four locations and three original equipment manufacturers (OEMs). These data differ in semantics and formats because of software versioning, customer customisation, as well as sensor and equipment discrepancy based on the concrete needs in the location. For example, they may be stored in various formats such as CSV, JSON, XML, etc., and may have different names for the same variables, have some variables missing in one source but present in another, or data may be measured with different sampling rate, etc.

In terms of *data volume*, data science models need a reasonably large amount of data to make the training meaningful and representative for the given data science tasks. For simplicity, we consider a representative example, whereby we assume one month data are meaningful, which was confirmed by data scientists at Bosch. In an example automobile factory responsible for manufacturing chassis (Figure 2b), there are 3 running production lines with a total of 45 welding machines. Each welding machine is responsible for a number of types of welding spots on the car bodies, with pre-designed welding programs. These machines

perform welding operations at different speeds, ranging from one welding spot per second, to one spot per several minutes. The data related to one single welding spot consist of several protocols or databases. After integration, these data become to a set of relational tables with 263 attributes, and a simplified estimation gives that one factory produces 64.8k spots every day, and 1.944 million spots per months, which account for the production of about 432 cars. Considering an average of 125 KB data for one welding operation gives the estimation of data volume meaningful for training as 243 GB (The real amount varies and can be larger, e.g., it was estimated as 389.32 GB in one real case. Here we adopt the simplification with a similar magnitude.).

Cloud deployment requirements. Considering the challenges, the welding quality monitoring system should give quality estimation/prediction not with excessive response time, although the data volume is large. In addition, the data come from various sources with diverse formats. Moreover, industrial data science projects involve many users that are non-cloud experts (Fig. 1). They should be equipped with tools to help deploy their data science solutions without extensive cloud expertise. The cloud infrastructure has resources of computing, memory, storage, network, etc. which need to be configured for optimised performance. Based on the information, we derive the following requirements for the system:

- *R1, Scalability on Data Volume:* The system should be able to reduce the computational time significantly when processing large data volumes.
- *R2, Scalability on Data Variety*: The system should be able to handle data variety, integrating heterogeneous data to uniform data formats.
- *R3, Scalability on Users:* The system should improve the *transparency* of the cloud system, automate resource configuration, and allow good *usability* for users, especially non-cloud experts,

3 SemCloud: Semantics-Enhanced Cloud System

To address the challenges and requirements, we propose our SemCloud system. We first give an architectural overview (Fig. 3) and then elaborate on the components.

3.1 Architectural Overview

The architecture of SemCloud is shown in Fig. 3. The *Data Analysis Workflow Layer* adopts a common workflow: data acquisition, data preparation, data analysis, results logging and interpretation; the raw data are first acquired, then prepared for data analysis, and, finally, the analysis results are generated, including models, predictions and human interpretation. In the data preparation stage, we employ *Semantic Data Integration* (Fig. 3.1) that relies on domain ontologies and semantic mappings to transform diverse data sources into uniform data formats. SemCloud scales the data analysis workflow to the cloud by with the *Distributed Computing* (Fig. 3.2), which includes the distributed ETL, distributed data analysis, and deployment orchestration that allocates cloud resources to the previous

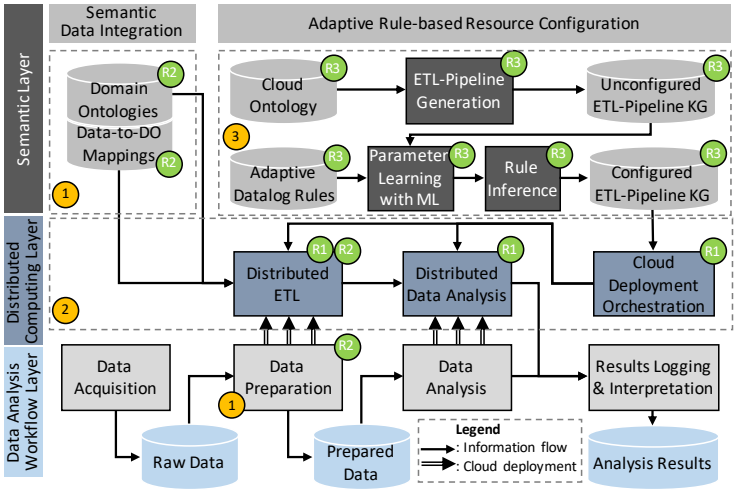

Fig. 3. Architectural overview of SemCloud including (1) semantic data integration; (2) distributed computing; (3) adaptive rule-based reasoning; each of which consists of a set of semantic artefacts (barrels) or modules (boxes). Ⓡ indicates the requirements the artefacts or modules intended to address.

two modules. *Adaptive Rule-based Resource Configuration* (Fig. 3.3) provides a cloud ontology and GUI for the users to encode ETL pipelines in KGs, which contain resource configuration information that is automatically reasoned by a set of adaptive Datalog rules. These rules consist of aggregation operations and parameterised functions, where the parameters in the functions are learned via ML.

3.2 Semantic Data Integration

To accommodate the diverse data sources/formats and convert all data to uniform data formats [6, 7], we employ domain ontologies as the data models and the semantic mappings (Data-to-DO, data to domain ontology) that map diverse data sources to the data models. In particular, the domain ontologies capture the knowledge of manufacturing processes, data, and assets. In the case of welding ontology, it is in OWL 2 QL, with 1542 axioms, which define 84 classes, 123 object properties and 246 datatype properties. The classes capture concepts such as welding operations, welding machines, welding products (spots), welding programs, sensor measurements, monitoring status, control parameters, etc. The diverse data sources have discrepancies in data formats (e.g., CSV, JSON, XML), feature names, feature composition (some features exist in some sources but not in others), etc. All features in the different data sources of the same welding process are mapped (one-to-one mapping for each data source) to object properties (for foreign keys) or datatype properties (for attributes) in the same domain ontology. All features are renamed, and data formats are unified in one of the selected formats, usually CSV (or relational database).

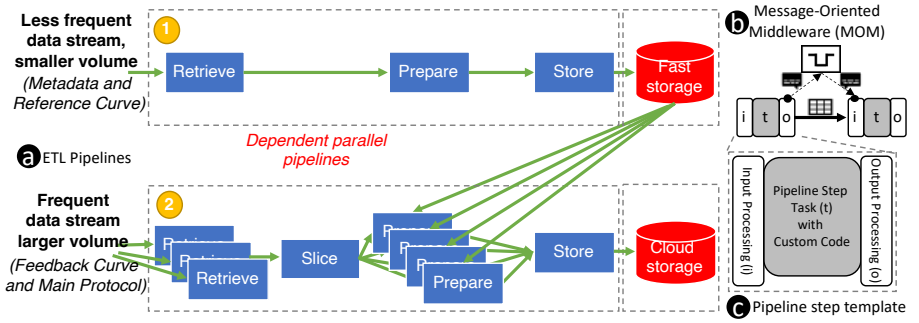

Fig. 4. (a) Dependent parallel ETL pipelines that break down ETL into four steps: *Retrieve*, *Slice*, *Prepare*, *Store*; (b) The cloud deployment of one step in the ETL pipeline as a container, where the MOM is responsible for the communication between steps of the ETL pipeline. (c) Zoom in one step, we see the three parts: *Input Processing*, *Output Processing* and *Pipeline Step Task*.

3.3 Distributed ETL and Data Analysis

Distributed ETL. To enable distributed ETL, we need to find a strategy that makes the ETL parallelisable, treat data streams with different updating frequencies, and handle data dependencies. SemCloud achieves this by breaking down the ETL into pipelines of four steps: *retrieve*, *slice*, *prepare*, and *store* (Fig. 4). Data retrieval constitutes the process of retrieving data from databases or online streams present in different factories and can normally be handled by a single computing node. These data are then split into subsets by the step *slice* to achieve parallel processing according to data semantics that make the splitting meaningful and each subset independently processable. In the welding use case, each subset only belongs to one welding machine, because the data analysis of welding quality monitoring of one machine can be safely assumed to be observable or predictable with data from this particular machine, without considering other machines. In this way, the datasets are separated into subsets that are computationally independent. We then deploy the ETL stage and the subsequent two stages on the cloud system that has resources for computing, storage and networking, to reduce the overall computational time.

An important strategy here is the hierarchically dependent parallel pipelines. We consider two types of data streams: the less frequently updated one with usually smaller volume, and the (more) frequently updated one with usually larger volume. (1) The former one has only three ETL steps: retrieve, prepare and store, because it requires resource (computing, storage, network) of one single cloud node and does not need slicing to parallelise. The intermediate results of this ETL pipeline are stored in a database using in-memory storage for fast query access. In the welding case, the metadata and reference curves follow this ETL pipeline. (2) The latter one has four ETL steps because it involves the application of slicing for parallelising. The results of this ETL pipeline are stored on dedicated cloud storage. In the welding case, the processing of feedback curves and main protocol requires more resources and is implemented through this pipeline. The ETL of these two data streams are dependent because the

prepare step of the frequent data stream must pull intermediate results of less frequent data stream.

Distributed data analysis. The key of distributed data analysis is to make assumptions of what computation is parallelisable, and split the computations into independent computing tasks. Here the target of data analysis is to predict the welding quality quantified by quality indicators such as spot diameters or Q-Values [11, 7]. The tasks include both classification (good or bad quality), regression (diameter values [12] or Q-Values), and forecasting [13] (predicting quality in the future). In practice, the latter two are preferred by domain experts because they provide more insights than a simple classification.

Both classic methods (feature engineering with e.g., linear regression) and deep learning (LSTM networks) are employed. We developed and tested various ML models [11, 13]. We used model performance for tuning the hyper-parameters and considered both model performance and adoption difficulty for selecting the best models [7]. These models take input features such as sensor measurements, monitoring status and control parameters and predict the quality indicators. The training was done with various regimes [14]: the ground truth training data included simulation data, lab data, historical production data; the validation data were subsets of the training data for selecting hyper-parameters; test data were both of the same welding machines or different machines (testing transferability). According to domain knowledge, we assume that the interplay between welding machines to be only marginally significant and that it is safe to predict the welding quality of one welding machine only by using information of this welding machine. This assumption has been verified and obtained a prediction error of about 2% [11]. Thus, the data analysis on data of each welding machine can be performed independently if each subset contains all data of one machine.

Cloud Deployment Orchestration. To orchestrate the distributed computation, SemCloud encapsulates ETL steps or data analysis as containers and runs the containers independently and in parallel, allowing for deploying multiple instances of the same ETL step or data analysis [8]. Each instance is implemented by a template composed of three main parts: *Input Processing*, *Pipeline Step Task*, and *Output Processing* (Figure 4c). The *Input Processing* fetches data from remote sources and moves the data to the step workspace. The *Pipeline Step Task* wraps custom code to process the fetched data. The *Output Processing* component delivers the processed data to a specific destination, notifies that they are available for the next steps, and clears up temporary and input data from the step workspace. Configuration and attributes of a pipeline step can be expressed as parameters and injected at deployment time. The communication between the steps is handled by Message-oriented Middleware (MOM) [15] (Figure 4b), so that the consecutive steps do not need to run simultaneously for interaction, ensuring temporal decoupling. None of the sequential steps needs to know about the existence of other steps or their scaling configuration, thus achieving space decoupling. Therefore, it is possible to assign more instances to bottlenecked pipeline steps that are more computationally heavy and reduce the overall processing time.

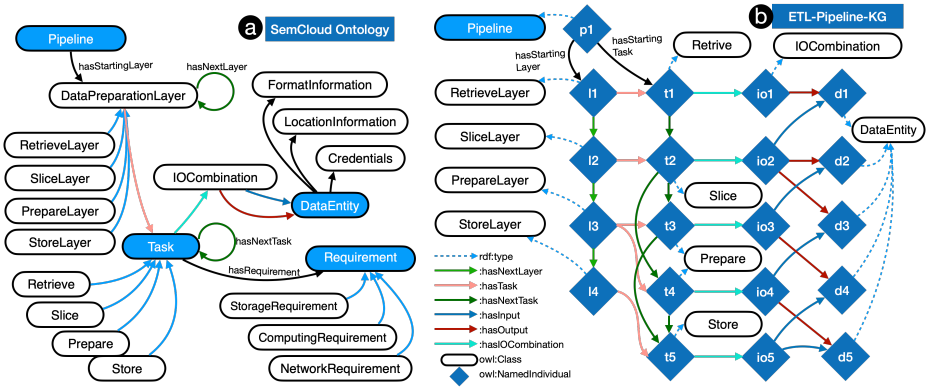

Fig. 5. (a) Schematic illustration of the SemCloud ontology. (b) Partial illustration of a KG for `ETL-Pipeline`.

3.4 ETL Pipeline Generation

Cloud Ontology. SemCloud provides the users GUI to construct ETL pipelines and encode them into knowledge graphs, based on a SemCloud ontology (Figure 5a). The ontology *SemCloud* is written in OWL 2, and consists of 20 classes and 165 axioms. It has three main classes: *DataEntity*, *Task*, *Requirement*. *DataEntity* refers to any dataset to be processed; *Task* has sub-classes that represents the four types of tasks in the data preparation: *retrieve*, *slice*, *prepare*, and *store*; and *Requirement* that describes the requirements for computing, storage and networking resources.

ETL Pipeline Generation in KGs. We now illustrate the generation of ETL pipelines in knowledge graphs with the example in Figure 5b. The data for welding condition monitoring have multiple levels of updating frequencies, which should be accommodated by the ETL pipelines. For example, data that are generated for each welding operation are updated after each welding operation, and thus are updated very frequently (about one second for one operation). For these data, the users construct an ETL pipeline `p1` with four layers (via GUI). Firstly, data are "retrieved" from the welding factories. Thus, the layer `l1` is of type *RetrieveLayer*, and has the task `t1` of type *Retrieve*. The task `t1` has an IO handler `io`, which has an output `d1` of type *DataEntity*. Then the data are read in by a task `t2` of type *Slice*, and "sliced" into smaller pieces `d2`, `d3`. These slices are input to different computing nodes to do tasks `t3` and `t4` of type *Prepare*. Finally, all prepared data entities are stored by `t5` of type *Store*.

3.5 Adaptive Datalog Rules Inference for Resource Configuration

Obtaining an optimised cloud configuration is not a trivial task. Cloud experts typically try different configurations by testing the system with various settings and use the system performance under these test settings as heuristics to manually decide the cloud configurations. In SemCloud, we design a set of declarative

adaptive rules written using logical programming to make the cloud configurations explicit, automated, less error-prone where the system optimisation is done with help of external functions learned with ML, such that the rules can be also used by non-cloud experts.

To this end, SemCloud uses adaptive rules in Datalog with aggregation and calls to external predicates learned by ML (they are adaptive because the function parameters are learned, see Sect. 3.6). In particular, we consider non-recursive rules of the form $A \leftarrow B_1, \ldots, B_n$ where A is a head of rule (the consequence of the rule application) and $B_1, \ldots, B_n$ are either predicates that apply join or aggregate function that filters out the results. For the theouploary of Datalog we refer to [16–19]. In the following we provide some example rules and explain their logic.

We have six different Datalog programs (set of rules) that run independently and that are divided into three steps: (i) graph extraction rules that populate rule predicates by extracting information from the ETL-pipeline KGs (e.g., $rule_0$) (ii) resource estimation rules that estimate the resource consumption for the given pipeline if there is only one computing node (assuming infinitely large nodes, e.g., $rule_2$) (iii) resource configuration rules that find the optimal resource allocation in distributed computing the given pipeline (e.g., $rule_3$).

Graph extraction rules. These rules populate the predicates so that these predicates will be used for the resource estimation and configuration. The $rule_0$ exemplifies populating the predicate `subgraph1` that is related to the ETL pipeline `p`.

```
subgraph1(p,n,v,ms,mp,ssl,spr,sst)  ←  ETLPipeline(p),
   hasInputData(p,d), hasVolume(d,v), hasNoRecords(d,n)
   hasEstSliceMemory(p,ms), hasEstPrepareMemory(p,mp)
   hasEstSliceStorage(p,ssl), hasEstPrepareStorage(p,spr)
   hasEstStoreStorage(p,sst)                                (rule0)
```

Similarly, we have rule $rule_1$ that creates `subgraph2(p,n,v,ms,mp,ts,tp, nc,ns,mrs,mrp,mode)`.

Resource estimation rules. These rules are used to estimate required resources assuming one computing node. For example, $rule_2$ estimates the required slice memory (`ms`), prepare memory (`mp`), slice storage (`ssl`), prepare storage (`spr`), and the store storage (`sst`). The rule then stores these estimations in the predicate `estimated_resource`.

```
estimated_resource(p,ms,mp,ssl,spr,sst) ←
    subgraph1(p,n,v,ms,mp,ssl,spr,sst),
    ms=@func_ms(n,v), mp=#avg{@func_mp(n,v,ms,i):range(i)},
    spr=#avg{@func_spr(n,v,ssl,i):range(i)},
    ssl=@func_ssl(n,v), sst=@func_sst(n,v,ssl,spr)        (rule2)
```

where `@func_ms`, `@func_ssl`, `@func_sst`, etc. are parameterised ML functions whose parameters are learnt in the *rule parameter learning* (Sect. 3.6). In the implementation, those are defined as external built-in functions that are called in the grounding phase of the program and then are replaced by concrete values [16,

17]. We also have other resource estimation rules that estimate other resources, such as CPU consumption.

Resource configuration rules. These rules find the optimal cloud configurations based on the estimated cloud resource. $rule_3$ is an example for deciding the slicing strategy and the storage strategy, and finding the optimal resource configuration such as the chuck size (`nc`), slice size (`ns`), memory reservation for *slice* (`mrs`) and for *prepare* (`mrp`). In essence, $rule_3$ stipulates that if the maximum of estimated slice memory (`ms`) and prepare memory (`mp`) is greater than a given threshold (`c1*nm`), and the maximum of estimated slice storage (`ssl`), prepare storage (`spr`), and store storage (`sst`) is smaller than (or equal to) another threshold (`c2*ns`), then the chosen strategy for the given pipeline is *slicing* (thus `nc` and `ns` are computed), and *fast storage* (`fs`, where the thresholds are calculated from cloud attributes.

```
configured_resource(p,nc,ns,fs,mrs,mrp) ←
    subgraph2(p,n,v,ms,mp,ts,tp,nc,ns,mrs,mrp,mode),
    estimated_resource(p,ms,mp,ssl,spr,sst),
    CloudAttributes(c,c1,c2,c3,nm,ns,fs,cs),
    #max{ms,mp} > (c1 * nm), #max{ssl,spr,sst} <= (c2 * ns),
    nc = @func_fs_1(n,v,ts,tp), ns = @func_fs_2(n,v,ts,tp),
    mrs = #min{ms, #max{@func_ss(n,v,nc,ns), c3*ms}},
    mrp = #min{mp, #max{@func_pn(n,v,nc,ns), c3*mp}}    (rule3)
```

3.6 Rule Parameter Learning with Machine Learning

The functions in the adaptive rules are in the form of ML models. The *resource estimation rules* are selected from the best model resulting from training three ML methods and the pilot running statistics. These three ML methods are *Polynomial Regression (PolyR)* (Eq. 1), *Multilayer Percetron (MLP)* (Eq. 2), and *K-Nearest Neighbours (KNN).* (Eq. 3). We selected these three methods because they are representative classic ML methods suitable for the scale of the pilot running statistics. PolyR transfers the input features ($x_i, i \in \{1, 2, ..., n\}$, n is the number of input features) to a series of polynomial vectors ($[1, x_i, x_i^2, ...x_i^m]$, m is the highest degree), and then constructs a predictor by multiplying a weight matrix ($\mathbf{W} \in \mathbb{R}^{m \times n}$). MLP consists of multiple layers of perceptrons, where each perceptron applies the *ReLu* function to the weighted sum of all neuron outputs of the previous layer plus the bias terms $\mathbf{W}^{(l-1)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l-1)}$. For a given data i whose output feature y_i is to be predicted, KNN finds its k samples (s, consisting of pairs of input $\mathbf{x}_s$ and output y_s) that are most similar to i (the k nearest neighbours $\mathcal{N}_k$) in the training data, and uses a weighted sum (the reciprocal of distance $d(s, i)$) of the output features y_s in $\mathcal{N}_k$ as the estimation.

The *resource configuration rules* are trained with the same three ML methods and with optimisation techniques such as Bayesian optimisation or grid search. For example, the functions `@func_fs_1` and `@func_fs_2` that find the optimal chuck size (`nc`) and slice size (`ns`) are trained by finding the arguments of (`nc`, `ns`) for the minimal total computing time ($t_{\texttt{total}}$)

$$\mathtt{nc}, \mathtt{ns} = \arg\min_{\mathtt{nc},\mathtt{ns}} t_{\mathtt{total}} = \arg\min_{\mathtt{nc},\mathtt{ns}} f(\mathtt{v}, \mathtt{n}, \mathtt{nc}, \mathtt{ns}, t_{\mathtt{slice}}, t_{\mathtt{prepare}})$$

$$\mathbf{x}_i = [1, x_i, x_i^2, ...x_i^m]^{\mathbf{T}} \quad \hat{y}_i = \sum_i \mathbf{W}\mathbf{x}_i \quad err = ||\hat{\mathbf{y}} - \mathbf{y}||^2 \tag{1}$$

$$\mathbf{h}^{(0)} = \mathbf{x} = [x_1, x_2, ..., x_n]^{\mathbf{T}} \quad \mathbf{h}^{(l)} = ReLu(\mathbf{W}^{(l-1)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l-1)}) \quad \hat{y} = ReLu(\mathbf{W}^{(L-1)}\mathbf{h}^{(L-1)} + b^{(L-1)}) \tag{2}$$

$$s = (\mathbf{x}_s, y_s) \quad \mathcal{N}_k = \{s | d(s,i) \leq d_k\} \quad d(s,i) = ||\mathbf{x}_s - \mathbf{x}_i|| \quad \hat{y}_i = \mathbf{w}\mathbf{y}_s, s \in \mathcal{N}_k \tag{3}$$

4 Implementation and Evaluation

We implemented our system with a front-end GUI based on Angular, HTML/CSS, and a back-end system based on ASP.NET Core, JavaScript, Python and DLV system [20, 21]. The GUI adopts the common design pattern of Model-View-Controller and has a RESTful API that handles the requests and responses between the front-end and back-end.

The evaluation consists of (4.1) cloud deployment report, verifying to what extent SemCloud reduces computational time for semantic ETL (R1, R2); and (4.2) rule parameter learning and inference, validating whether the rule parameter learning and inference is scalable (R1) and accurate, so that the non-cloud experts can use SemCloud with confidence (R3).

4.1 Cloud Deployment Report

Data Description. To determine whether SemCloud reduces computational time, we use a dataset of 3 production lines for one month. The dataset is anonymised and simulated based on a welding factory in Germany. We simulated the dataset because it allows the freedom of evaluating settings and the information of real data is subject to a non-disclosure agreement. One production line has 10 - 20 machines, amounting to 45 machines in total. Each machine performs welding operations at a different speed, ranging from 1 spot/second, to 1 spot per several minutes (due to maintenance time, delay time, and various situations). The total amount of data are 389.42 GB, which represent 3.1 million spots, estimated to be related to 692.3 cars

Deployment Setting. We deployed the SemCloud system on an infrastructure of 7 computing instances connected by a network that were managed by a Rancher container orchestrator [22]. We adopt the automatic setting, whereby resource configurations are provided via adaptive Datalog rules and the Rancher system automatically assigns containers to resources according to the configuration.

Performance Comparison. We demonstrate the performance comparison between the ETL processing with the legacy system (without SemCloud) and with our SemCloud system (Figure 6). The legacy system is comprised of a integrated software that performs both the preparation of the metadata/reference curves and the processing of the feedback curve and main protocol data. The legacy

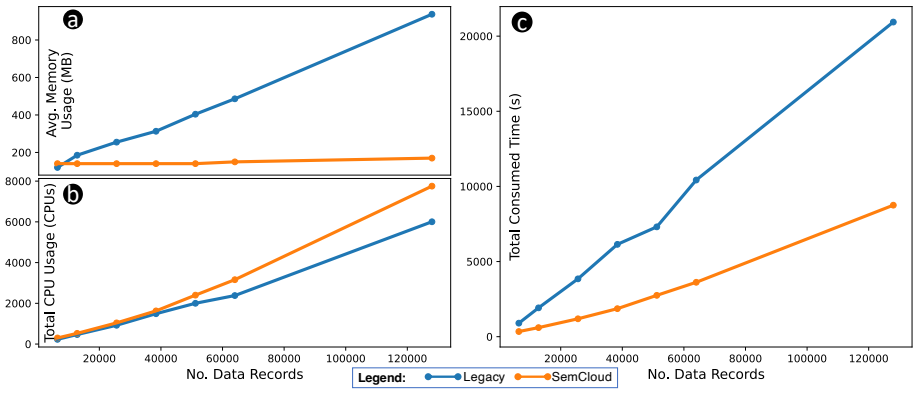

Fig. 6. (Performance comparison by (a) memory usage, (b) total CPU usage (integrated over time), (c) consumed time: SemCloud significantly reduces the memory usage and consumed time (by about 50%), and uses slightly more total CPU, compared to the *legacy* solution (Without SemCloud). X-axis: processed input data volume.

system was deployed in a node that meets the total requirements for the experimental data to monitor the resource usage.

It can be seen that the the memory usage of the computing instance for the legacy solution increases monotonously along the processed input data volume, while SemCloud requires almost zero increase of memory allocation, which means SemCloud can deploy the ETL process on many computing instances with no extra memory demand. Figure 6b shows SemCloud requires slightly more CPU power. This is expected and understandable, because the distributed deployment consumes more computing power per unit of time, but decreases the overall computing time. The latter is confirmed by Figure 6c: as the input data volume increases the reduced computing time brought by SemCloud becomes increasingly significant (R1).

4.2 Evaluation of Rule Parameter Learning and Inference

Pilot Running Statistics. To verify the scalability and accuracy of the rule parameter learning and inference, we gather pilot running statistics, train and test the ML functions in SemCloud. We run SemCloud repeatedly 3562 times with different sizes of subsets of the welding dataset in Sect. 4.1 and gather pilot running statistics. These statistics include data information, e.g., input data size, different configurations, e.g., slice size, and recorded resource consumption e.g., memory consumption, mCPU consumption.

Experiment Setting. We split the pilot running statistics so that 80% are for rule parameter training and 20% for rule testing and inference. Three ML models are trained and tested: *PolyR*, *MLP*, and *KNN*. We adopt a grid search strategy for hyper-parameter tuning. The final hyper-parameter are, PolyR: 4 degree; MLP: 2 hidden layers with neurons 10 and 9; KNN: 2 neighbours.

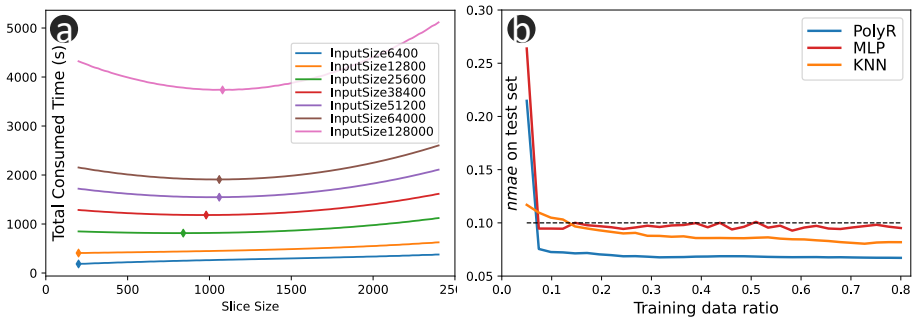

Fig. 7. (a) Optimisation to find the best slice size for the least time, when chunk size is fixed. (b) Comparing ML methods to find the minimal training data

Performance Metrics. We use several performance metrics: *normalised mean absolute error* (*nmae*) [23] to measure prediction accuracy, minimal training data amount (Min. $|\mathcal{D}_{train}|$ for yielding satisfactory results, optimisation time (Opt. time), learning time and inference time. Intuitively, *nmae* reflects the scale-independent average prediction error. It is computed as *mean absolute error* normalised by the mean value of the configuration $\bar{c}$: $nmae = mae/\bar{c}$. *nmae* reflects the mean absolute error between the ground truth configuration c and the predicted configuration $\hat{c}$: $mae = \frac{1}{|\mathcal{D}_{test}|}\sum_{\mathcal{D}_{test}} |c - \hat{c}|$, and $\bar{c} = \sum_{\mathcal{D}_{test}} c/|\mathcal{D}_{test}|$. We normalise *mae* because its scale is dependent on the variable for which we calculate *mae*. If it is divided by the mean value of the variable, it becomes *nmae* which is scale-independent.

Learning and Inference Results. The optimisation results of *slice size* shows we can configure it to find a "sweet spot" to minimise the total consumed time (Figure 7a). The performance of rule learning and inference is shown in Table 1 (R3). The learning time is the time it takes to train the models. The inference time includes the model inference time and the rule reasoning time. It can be seen that the PolyR has the best prediction accuracy, requires the least training data, and consumes the least time. Therefore, PolyR generates the best results and is selected for the use case. We presume the reason is that PolyR works better with small amounts of and not very complex data (3562 repeated running statistics). We can see MLP is not very stable (Figure 7b). This is due to the random initialisation effect of MLP.

Table 1. Parameter learning and reasoning results, recorded on Intel Core i7-10710U.

Metric	PolyR	MLP	KNN
nmae	0.0671	0.0947	0.0818
Min. $\lvert\mathcal{D}_{train}\rvert$	7.42%	50.97%	10.00%
Opt. time	1.12s	174.32s	7.25s
Learning time	20.82ms	120.31ms	27.52ms
Inference time	<1.00ms	<1.00ms	<5.00ms

5 Discussion on General Impact and Related Works

Uptake for the cloud community. SemCloud is an attempt for democratising cloud systems for non-cloud experts. We hope to inspire research and a broad range of users who are pursuing scaling data science solutions on the cloud, but

are impeded by the long training time for acquiring cloud expertise. Providing a dynamically scalable (on a step and pipeline level), general-purpose solution for big data pipelines on heterogeneous resources that a broad audience can use is an open research topic [24, 25]. The currently available multitude of tools for big data pipeline management only partially addresses these issues as shown in [26]. Data pipeline management approaches in literature such as [27–29] also partially address the issue but either specifically support a knowledge domain (scientific workflows, ad-hoc languages), fail to address issues related to individual step scalability, or are not well-suited for dynamic long-running jobs. Other works that touch that topic [30–33] also do not address the automatic resource allocation issue and are not designed for non-cloud experts. Our approach tackles these issues and the presented principles should be easy to reproduce.

Uptake in terms of semantic technology. We open-source our cloud ontology [34]. We hope this ontology can facilitate research of semantic technology in the scalability challenges, that the tenets of explicit, transparent, and shared knowledge can advance in the practice in academia and industry. We developed it as we did not find a suitable ontology for our challenges. Past works about cloud ontologies focus more on describing the different layers and components [4, 5], services [35], functional or non-functional features and the interaction between the layers [36]. They cover the cloud tasks and resource allocation, but to a limited extent. There exist other works about the resource management topic [37–39], but they do not provide mechanism or reasoning for adaptive and automatic resource configuration. Works about cloud reasoning are focused on other aspects like security attacks [40], minimising sources like computing nodes [41], computational requirements [42], service placement [43], verifying policy properties [44], deploying semantic reasoning on the cloud [45]. There is insufficient discussion on helping users to automate the cloud resource configuration.

Uptake by stakeholders and benefits. Semantic technologies play an increasingly important role in modern industrial practice. Ontologies, as a good way for formal description of knowledge, offer unambiguous "lingua franca" for cross-domain communication. They can help users to perform tasks of a remote domain that otherwise would be error-prone, time-consuming and cognitively demanding. We incorporated rule-based reasoning, falling in the category of symbolic reasoning, with machine learning, which is a type of sub-symbolic reasoning. The combination takes benefits from both: SemCloud becomes more agnostic of cloud infrastructure and adapts to the resource conditions, thus exploiting explicit domain knowledge via semantics and learning implicit relationship via ML.

In addition, we tested SemCloud with users of various backgrounds (welding experts, data scientists, semantic experts). SemCloud could improve their working efficiency. Before using SemCloud, users that are the non-cloud experts have very limited understanding of the cloud system, and did not use the cloud system. Through SemCloud, these users could obtain better understanding of the cloud system, start using the cloud system, and rely on the SemCloud to automatically configure the resource allocation. We tested the GUI with the users to collected feedback for improving the usability and expanding the functionalities.

Lessons Learnt on costs and risks. The main costs for development of such systems comprise the early development time for the semantic infrastructure that mediates between the cloud resources, data analysis solutions and users. Naturally, these costs vary depending on the specific project. It was manageable in our case, but should be carefully evaluated for each project individually. The key lessons learnt for reducing costs is that a good cross-domain communication framework is essential, where experts of different backgrounds can speak a common language and reduce misunderstanding and communication time. A possible and important risk is that the assumption could be wrong as to whether and to what extent the ETL and data analysis can be parallelised. It is recommended to verify the assumption early to avoid further costs.

6 Conclusion, and Outlook

Conclusion. This work presents our SemCloud system motivated by a Bosch use case of welding monitoring, for addressing the scalability challenges in terms of *data volume*, *variety*, and more *users*. SemCloud provides semantic abstraction that mediates between the users, ETL and data analysis, as well as cloud infrastructure. The scalability in terms of data variety is addressed by semantic data integration, data volume by distributed ETL and data analysis, and scalability to more users by adaptive Datalog rule-based resource configuration. These Datalog rules are adaptive because they have parameterised functions learnt from pilot running statistics via ML, a combination of symbolic and sub-symbolic approaches. We evaluated SemCloud extensively through cloud deployment with large volume of industrial data, and rule learning from thousands of pilot runs of the system, showing very promising results.

Outlook. SemCloud is under the umbrella of Neuro-Symbolic AI for Industry 4.0 at Bosch [46] that aims at enhancing manufacturing with both symbolic AI (such as semantic technologies [47]) for improving transparency [48], and ML for prediction power [49]. Bosch is developing user-friendly cloud technology in the framework of EU project DataCloud with many EU partners [50]. SemCloud is partially developed with production end-users and current deployed in a Bosch evaluation environment. We plan to push it into the production to test with more users and collect feedback, and work together with EU partners for transferring the knowledge and experience to other manufacturing domains to increase the wide adoption. We also plan to develop formal theories for knowledge representation and reasoning for cloud technology and automatic resource configuration, e.g., better modelling framework, advanced reasoning rules, and deeper integration of symbolic and sub-symbolic reasoning.

Acknowledgements. The work was partially supported by the European Commission funded projects DataCloud (101016835), enRichMyData (101070284), Graph-Massivizer (101093202), Dome 4.0 (953163), OntoCommons (958371), and the Norwegian Research Council funded projects (237898, 323325, 309691, 309834, and 308817).

References

1. H. Kagermann, Change through digitization – value creation in the age of Industry 4.0, in: Management of Permanent Change, 2015.
2. ITU, Recommendation ITU – T Y.2060: Overview of the internet of things, Tech. rep., International Telecommunication Union (2012).
3. S. Chand, J. Davis, What is smart manufacturing, Time Magazine Wrapper 7 (2010) 28–33.
4. L. Youseff, M. Butrico, D. Da Silva, Toward a unified ontology of cloud computing, in: 2008 Grid Computing Environments Workshop, IEEE, 2008, pp. 1–10.
5. Z. S. Ageed, R. K. Ibrahim, M. Sadeeq, Unified ontology implementation of cloud computing for distributed systems, Current Journal of Applied Science and Technology (2020) 82–97.
6. Y. Svetashova, B. Zhou, T. Pychynski, S. Schmidt, Y. Sure-Vetter, R. Mikut, E. Kharlamov, Ontology-enhanced machine learning: A Bosch use case of welding quality monitoring, in: The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II, Vol. 12507, Springer, 2020, pp. 531–550.
7. B. Zhou, Y. Svetashova, A. Gusmao, A. Soylu, G. Cheng, R. Mikut, A. Waaler, E. Kharlamov, SemML: Facilitating development of ML models for condition monitoring with semantics, J. Web Semant. 71 (2021) 100664.
8. N. Nikolov, Y. D. Dessalk, A. Q. Khan, A. Soylu, M. Matskin, A. H. Payberah, D. Roman, Conceptualization and scalable execution of big data workflows using domain-specific languages and software containers, Internet of Things 16 (2021) 100440.
9. DIN, Maintenance-maintenance terminology, Trilingual Version EN 13306:2017 13306 (2018) 2017.
10. ISO, Resistance welding – procedures for determining the weldability lobe for resistance spot, projection and seam welding, Standard, International Organization for Standardization, Geneva, CH (2004).
11. B. Zhou, Y. Svetashova, S. Byeon, T. Pychynski, R. Mikut, E. Kharlamov, Predicting quality of automated welding with machine learning and semantics: A bosch case study, in: CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020, ACM, 2020, pp. 2933–2940.
12. B. Zhou, T. Pychynski, M. Reischl, R. Mikut, Comparison of machine learning approaches for time-series-based quality monitoring of resistance spot welding (rsw), Archives of Data Science, Series A (Online First) 5 (1) (2018) 13.
13. B. Zhou, T. Pychynski, M. Reischl, E. Kharlamov, R. Mikut, Machine learning with domain knowledge for predictive quality monitoring in resistance spot welding, Journal of Intelligent Manufacturing 33 (4) (2022) 1139–1163.
14. B. Zhou, Machine learning methods for product quality monitoring in electric resistance welding, Ph.D. thesis, Karlsruhe Institute of Technology, Germany (2021).
15. M. Albano, L. L. Ferreira, L. M. Pinho, A. R. Alkhawaja, Message-oriented middleware for smart grids, Computer Standards & Interfaces 38 (2015) 133–143.
16. N. Leone, G. Pfeifer, W. Faber, F. Calimeri, T. Dell'Armi, T. Eiter, G. Gottlob, G. Ianni, G. Ielpa, C. Koch, et al., The dlv system, in: Logics in Artificial Intelligence: 8th European Conference, JELIA 2002 Cosenza, Italy, September 23–26, 2002 Proceedings 8, Springer, 2002, pp. 537–540.

17. G. Ianni, F. Calimeri, A. Pietramala, M. C. Santoro, Parametric external predicates for the DLV system, CoRR cs.AI/0404011 (2004).
URL http://arxiv.org/abs/cs/0404011
18. S. Abiteboul, R. Hull, V. Vianu, Foundations of databases, Vol. 8, Addison-Wesley Reading, 1995.
19. S. Paramonov, N. Werner, S. Ognjen, An asp approach to query completeness reasoning, Theory and Practice of Logic Programming 13 (4) (2013) 1–10.
20. DLVHEX, DLVHEX source documentation, accessed 31 July 2023.
URL http://www.kr.tuwien.ac.at/research/systems/dlvhex/doc2x/index.html
21. T. Eiter, S. Germano, G. Ianni, T. Kaminski, C. Redl, P. Schüller, A. Weinzierl, The DLVHEX system, KI-Künstliche Intelligenz 32 (2018) 187–189.
22. Rancher, Rancher kubernetes clusters, https://rancher.com/products/rancher, accessed 14 March 2022 (2022).
23. T. Chai, R. R. Draxler, Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature, Geoscientific model development 7 (3) (2014) 1247–1250.
24. M. Barika, S. Garg, A. Y. Zomaya, L. Wang, A. V. Moorsel, R. Ranjan, Orchestrating big data analysis workflows in the cloud: Research challenges, survey, and future directions, ACM Computing Surveys 52 (5) (2019) 95:1–95:41.
25. R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, et al., A manifesto for future generation cloud computing: Research directions for the next decade, ACM Computing Surveys 51 (5) (2018) 105:1–105:38.
26. M. Matskin, S. Tahmasebi, A. Layegh, A. H. Payberah, A. Thomas, N. Nikolov, D. Roman, A survey of big data pipeline orchestration tools from the perspective of the datacloud project, in: Supplementary Proceedings of the XXIII International Conference on Data Analytics and Management in Data Intensive Domains, Vol. 3036, Moscow, Russia, 2021.
27. W. Gerlach, W. Tang, K. Keegan, T. Harrison, A. Wilke, J. Bischof, M. D'Souza, S. Devoid, D. Murphy-Olson, N. Desai, Skyport-container-based execution environment management for multi-cloud scientific workflows, in: 2014 5th International Workshop on Data-Intensive Computing in the Clouds, IEEE, 2014, pp. 25–32.
28. R. Qasha, J. Cala, P. Watson, Dynamic deployment of scientific workflows in the cloud using container virtualization, in: 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2016, pp. 269–276.
29. A. Alaasam, G. Radchenko, A. Tchernykh, K. Borodulin, A. Podkorytov, Scientific micro-workflows: where event-driven approach meets workflows to support digital twins, in: Russian Supercomputing Days, 2018, pp. 489–495.
30. Q. W. Tan, W. Goh, M. Mutwil, Lstrap-cloud: a user-friendly cloud computing pipeline to infer coexpression networks, Genes 11 (4) (2020) 428.
31. M. Zhao, Z. Li, W. Liu, J. Chen, X. Li, Ufc2: User-friendly collaborative cloud, IEEE Transactions on Parallel and Distributed Systems (2021).
32. P. S. Kumar, A. Kumar, P. S. Rathore, J. M. Chatterjee, An on-demand and user-friendly framework for cloud data centre networks with performance guarantee, Cyber Security in Parallel and Distributed Computing: Concepts, Techniques, Applications and Case Studies (2019) 149–159.
33. D. Mulfari, A. Celesti, M. Villari, A computer system architecture providing a user-friendly man machine interface for accessing assistive technology in cloud computing, Journal of Systems and Software 100 (2015) 129–138.
34. B. Zhou, Z. Zheng, E. Kharlamov, The SemCloud ontology, open source under: https://github.com/nsai-uio/SemCloud (2023).

35. A. Tahamtan, S. A. Beheshti, A. Anjomshoaa, A. M. Tjoa, A cloud repository and discovery framework based on a unified business and cloud service ontology, in: 2012 IEEE Eighth World Congress on Services, IEEE, 2012, pp. 203–210.
36. M. M. Al-Sayed, H. A. Hassan, F. A. Omara, Cloudfnf: An ontology structure for functional and non-functional features of cloud services, Journal of Parallel and Distributed Computing 141 (2020) 143–173.
37. G. G. Castañé, H. Xiong, D. Dong, J. P. Morrison, An ontology for heterogeneous resources management interoperability and hpc in the cloud, Future Generation Computer Systems 88 (2018) 373–384.
38. Y. B. Ma, S. H. Jang, J. S. Lee, Ontology-based resource management for cloud computing, in: Asian Conference on Intelligent Information and Database Systems, Springer, 2011, pp. 343–352.
39. C. Zhang, Y. Yang, Z. Du, C. Ma, Particle swarm optimization algorithm based on ontology model to support cloud computing applications, Journal of Ambient Intelligence and Humanized Computing 7 (5) (2016) 633–638.
40. C. Choi, J. Choi, Ontology-based security context reasoning for power iot-cloud security service, IEEE Access 7 (2019) 110510–110517.
41. M. Ghetas, C. H. Yong, Resource management framework for multi-tier service using case-based reasoning and optimization algorithm, Arabian Journal for Science and Engineering 43 (2) (2018) 707–721.
42. A. Rakib, I. Uddin, An efficient rule-based distributed reasoning framework for resource-bounded systems, Mobile Networks and Applications 24 (1) (2019) 82–99.
43. S. Forti, G. Bisicchia, A. Brogi, Declarative continuous reasoning in the cloud-iot continuum, Journal of Logic and Computation 32 (2) (2022) 206–232.
44. J. Backes, P. Bolignano, B. Cook, C. Dodge, A. Gacek, K. Luckow, N. Rungta, O. Tkachuk, C. Varming, Semantic-based automated reasoning for AWS access policies using SMT, in: 2018 Formal Methods in Computer Aided Design (FMCAD), IEEE, 2018, pp. 1–9.
45. X. Su, P. Li, J. Riekki, X. Liu, J. Kiljander, J.-P. Soininen, C. Prehofer, H. Flores, Y. Li, Distribution of semantic reasoning on the edge of internet of things, in: 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE, 2018, pp. 1–9.
46. B. Zhou, Z. Tan, Z. Zheng, D. Zhou, Y. He, Y. Zhu, M. Yahya, T. Tran, D. Stepanova, M. H. Gad-Elrab, E. Kharlamov, Neuro-symbolic AI at Bosch: Data foundation, insights, and deployment, in: Proceedings of the ISWC 2022 Posters, Demos and Industry Tracks, Vol. 3254 of CEUR Workshop Proceedings, 2022.
47. M. Yahya, B. Zhou, J. G. Breslin, M. I. Ali, E. Kharlamov, Semantic modeling, development and evaluation for the resistance spot welding industry, IEEE Access (2023).
48. Z. Zheng, B. Zhou, D. Zhou, A. Soylu, E. Kharlamov, Executable knowledge graph for transparent machine learning in welding monitoring at bosch, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 5102–5103.
49. Z. Tan, B. Zhou, Z. Zheng, O. Savkovic, Z. Huang, I. G. Gonzalez, A. Soylu, E. Kharlamov, Literal-aware knowledge graph embedding for welding quality monitoring: A Bosch case, in: ISWC, Springer, 2023.
50. DataCloud, Enabling the big data pipeline lifecycle on the computing continuum, https://datacloudproject.eu/, accessed 14 March 2022 (2022).