YEREVAN STATE UNIVERSITY

Architecture enabling Data Driven Projects for a Modern Enterprise

by

Artyom Topchyan

Faculty of ICT Research and Development Center, YSU

October 2016

Contents

List of Figures

List of Tables

Abbreviations

CSV	**Comma-separated values**
CRISP-DM	**Cross Industry Standard Process for Data Mining**
DWH	**Data Warehouse**
IMP	**Information Market Place**
NEAT	**Neural Evolution of Augmenting Topologies**
AWS	**Amazon Web Services**
DB2	**IBM DB2**
LDA	**Latent Dirichlet Allocation**
LSH	**Locality Sensitive Hashing**
ODP	**Operational Data Platform**

Introduction

With the growing volume and demand for data a major concern for an Organization is how to use this data more effectively to generate value for the organization. To address this, more and more Organizations are aiming at implementing Data-Driven projects [2][3], which are increasing the quality, speed, and/or quantity of information gain for innovating a new methodology or the economic benefit to an organization.

These projects are about finding data, understanding data and accessing it. This has become not a simple task with the amount of data and documentation being created at organizations growing rapidly. Large organizations have thousands of employees that create dozens of systems, which produce 10's of TB's of data every month. All of this data is stored in database- and file-systems scattered throughout the organization. While there is a defined way to manage the descriptions of such technical systems themselves, the same thing is not true for all the data and meta-data which describe what the data actually contains, which is confirmed by the research carried out by Google [4]. The outlined tool, which was developed in parallel with the presented research and the first publication, which does not contain technical details was published a few months ago.

TB's of data and thousands of documentation pieces and meta-data definitions for tables, types and etc do not have a defined way of finding and using them in projects. This has become even a larger problem, with the growing requirement for more low-latency use-cases, where Organizations want to very quickly react to something a customer does. Low-Latency also means reacting in seconds as opposed to hours or days, which is commonly the case. While there are systems, that approach these issues, such as Data Lakes

and Data Warehouses [5] [6] for data access and Enterprise Data Management Systems for knowledge management, there is no single architectural approach that aims to efficiently solve these problems together from the specific view point of an Organization with years of organizational structure and system already in place and follows the CRISP-DM model [7], which defines the methodology for search and using data.

The aim of the dissertation is to create an Architecture an a number of tools which address the problem of creating Data-Driven projects at an Organization. These Architecture and tools have to address the specific requirements of organizations; they have to support the CRISP-DM model, be scalable, fault-tolerant, functional under limited resource contains and support lowlatency processing. The following components are supported:

- A flexible way for discovering data and interconnections of the data, based on meta-data, functional descriptions and documentation. The following requirement have to be fulfilled:
  - An automated, scalable, low-latency stream-processing based approach for discovering organizational data and interconnections of that data, based on organizational meta-data, functional descriptions and documentation.
  - Fully stream processing based system allowing for low-latency updates and access to information.
  - Automatically scalable systems, adapting to incoming data and consumers and requiring low maintenance.
- System for collecting and processing all of the organizational data, that allows for collaborative and shared creation of Data-Driven projects as well simplify testing and deployment of said projects. The following requirement have to be fulfilled:
  - A model for fault-tolerant raw data ingestion of all organizational data sources, which support low-latency data processing and does not adversely effect already existing projects and pipelines.
  - A model for isolated Project Development environments, ensuring the security and collaboration of multi-disciplinary teams.

Abbreviations

  - A set of primitives, which enable large scale, low-latency, autoscaling, structured data ingestion pipelines.
  - A system that provides automatically scalable, low-maintenance, strong correctness and delivery guarantees, which adapts adapting to incoming data and consumers.

Chapter 1

Data-Driven projects at large Organizations

In this chapter we will describe in more detail what Data-Driven projects are, how they are managed in Industry and the challenges faced when implementing these project from the perspective of the Business departments, Data-Scientists, IT departments. To clarify the problem, we are approaching let's define in more detail what a Data-Driven Project is and what the life cycle and goals are.

1.1 Data-Driven projects

A Data-Driven project aims at increasing the quality, speed and/or quantity of information gained from Data. Any type of data can be used varying in size, source and business/operational importance. Such projects usually involve Data Scientists, Business and IT working together to build up use cases by analysing, processing and presenting data in a meaningful way. The result of the project may be a report, dashboard, or a web service used by other systems. These are very involved projects and require a great degree of domain, statistical, modelling as well as large scale data processing knowledge. These are usually cross disciplinary project, that aim at analysing and increasing interactions with customers as well as improving internal processes in the organization.

To highlight this, lets outline two examples of Data-Driven projects. Both these examples center on use-cases at a large car manufacturing organization, that is starting to develop Data-Driven projects.

- A project that aims the Quality Assurance tickets generated by the organizations manufacturing departments. This opens up a large variety of potential use-cases, such as identifying root causes of certain failures based on previous similar tickets as well as providing a flexible way to organize tickets, spare parts and vehicles based on issues. This can be done by joining the analysed text of the tickets with vehicle and spare parts data.

- A project that aims to analyse Telematics(sensor) data from vehicles in real-time to analyse how these vehicles are being used by the customers of the company and suggest potential repair or repair windows. This is a classic example of predictive maintenance [8]. Such a project involves fairly complex model as there are a large number of useful sensory inputs that have to be considered. In some vehicles up to 500 reading parameters. This has to be done in a low-latency way to enable quick interaction with the customer.

These project are an example of a Data-Driven project as they combine multiple organizational data source and use modelling approaches to predict possible future events based on this. Both sensory data and binary text data are often not used extensively in the Industry as they present a challenge in analysis and processing. These issues are what we are aiming to solve and are going to center on throughput this work. To this end we will be referring back to these two example throughout this work to highlight some of the challenges of the logical and technical implementation.

Due to the interdisciplinary and data oriented nature of these projects, the way they are structured is quite different to classical IT and Business projects. Most Data-Driven projects follow a variation of the Cross Industry Standard Process for Data Mining project life-cycle [7]. Let’s outline it in more detail and analyse why it is more applicable to such types of project.

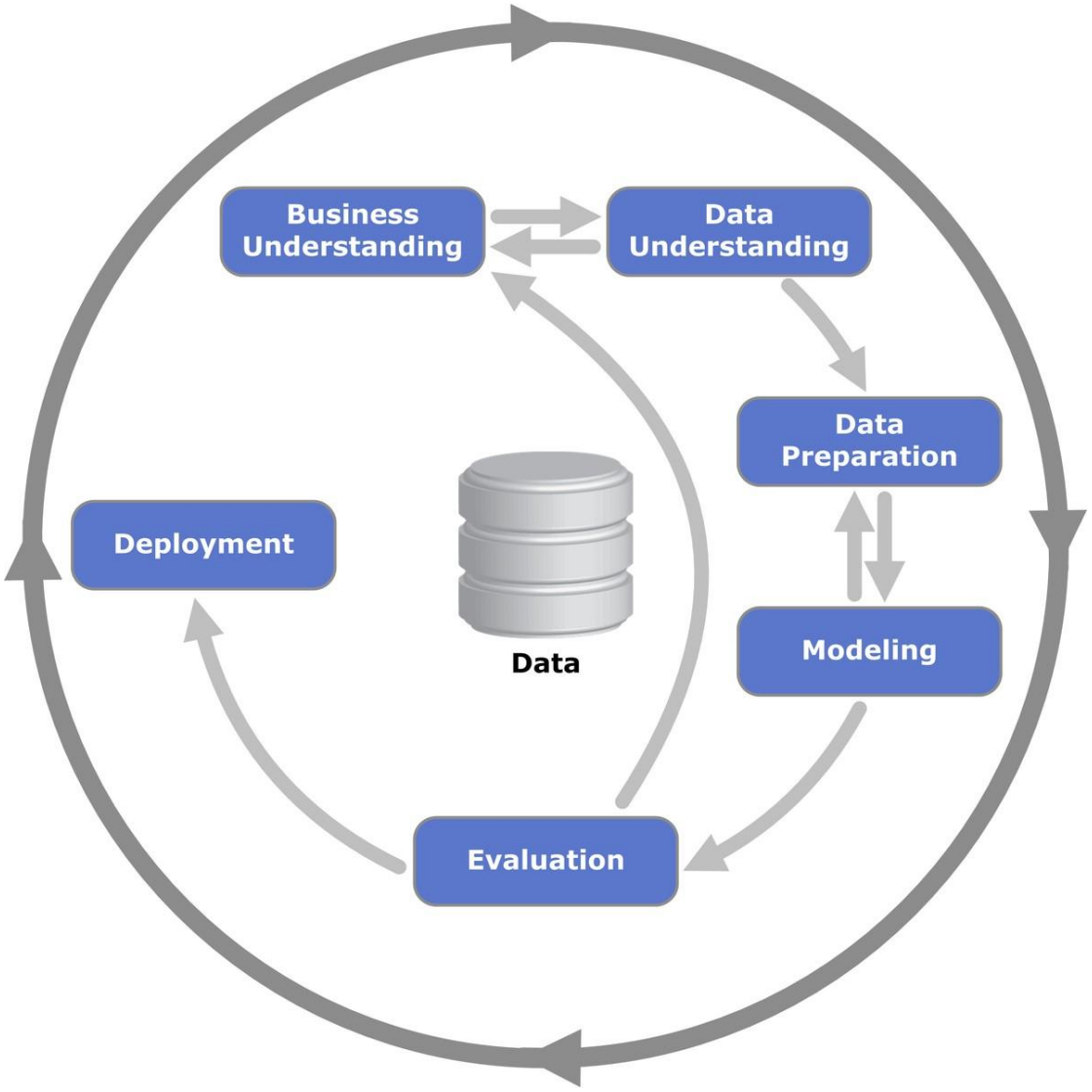


Figure 1.1: Cross Industry Standard Process for Data Mining (CRISP)

1.1.1 Cross Industry Standard Process for Data Mining

Most Data-Driven projects follow a variation of the CRISP model. This varies from organization to organization depending on the maturity of the DataDriven mindset, but CRISP is one of the most widely accepted approaches to such projects. The life-cycle usually consist of six stages of development, which can be iterated upon and repeated or completely abandoned. And each step iteration usually involves handling different source of data. This is highlighted in Fig. 1.1 The CRISP-DM model was designed by a collaboration of multiple organization aiming to standardize the process of developing Data-Mining projects. And in our opinion in clearly applies to Data-Driven projects, which are essentially and extensions of Data-Mining projects.

CRISP-DM organizes the data mining process into six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. These phases are used by organizations to understand the data-driven projects and provide a rough plan to follow while implementing such projects.

- **Business Understanding.** This initial phase focuses on understanding the project objectives and requirements from a business perspective, and then converting this knowledge into a data-driven problem definition, and a preliminary plan designed to achieve the objectives [8]. In our outlined examples the Business perspective of optimizing costs for customers as service and suggesting repair windows and determining potential similar faults with vehicle parts in order to optimize Quality Assurance processes translate to building a predictive model of the vehicles sensory data in order to predict future faults with the current usage and a clustering of the quality assurance tickets based on their content and spare parts groups.

- **Data Understanding.** Once the case is more or less clear it has to be clarified if there is any data to support it. The business department often knows what data can be used, but more substantive knowledge of the data is required, so it is a task of the business department and the Data Scientists to find out who owns and has knowledge of any data related to the case. This can take a very long time and is notoriously difficult in a large organization, because it is most often unclear who the data owner is and who is knowledgeable about this data. These are quite often different people. In our experience, this process might take upwards to two months' time and often it is discovered there is not substantial data to support the use case. This is already approximately 3 months on a case that potentially is not even possible.

- **Data acquisition and preparation.** If the data is present the next challenge is to acquire even a small sample of the data, which is usually customer data and is not shared easily between departments. This is again a costly process and can take up to two months. Luckily Data Scientist can start work on at least sample data if it can be supplied.

  But this again does not guarantee any data will arrive in the end. The data has to be transferred and transformed into a usable state. This may also take a large amount of time and is quite often the most time consuming phase that involves technical work. In our experience this process is repeated multiple times throughout

the project and each iteration may take weeks. At this stage it can be found out that the data is corrupted, with columns missing or being uninterpretable due to formatting loss or it is just very sparse.

- **Modeling.** In this phase, various modeling techniques are selected and applied, and their parameters are calibrated to optimal values. This is dependent on what type of problem is being solved and is greatly influence by the type of data available. Some techniques have specific requirements on the form of data. Therefore, stepping back to the data preparation phase is often needed [7]. Depending on the problem and budget this can take anywhere from one to two months.

- **Evaluation.** Before proceeding to final deployment of the model built by the data analyst, it is important to more thoroughly evaluate the model and review the model's construction to be certain it properly achieves the business objectives. At this stage some result can be shown and the models and approach evaluated, preliminary results discussed and it is decided if there any value in continuing the project. To evaluate the model, it has to be evaluate on the entire set of data.

- **Deployment.** Creation of the model is generally not the end of the project. Even if the purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in a way that it is usable. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as establishing a fault-tolerant and monitorable process to repeat the modelling and provide periodic or real time results to the user. This is usually done in conjunction with the IT department. This process has to often be subject to the requirements and limitation of the departments hosting the solution, which greatly limits flexibility and performance. In our experience this is actually the most complicated step based on the problem and can take many months.

It should be noted, that quite often these project involved external suppliers, which means they are inherently more expensive the longer the projects take. So in essence a project can fail on many separate stages. Which might take months and can be very expensive and deliver few to no results. Because of this organizations are interested in optimizing each stage of the process in order achieve faster success and faster failure time windows. This optimization poses certain organizational and technical challenges from the view point of

the organizations departments, which will outline in the following section. The tasks executed in each individual stage are highlighted in more detail in Fig. 1.2

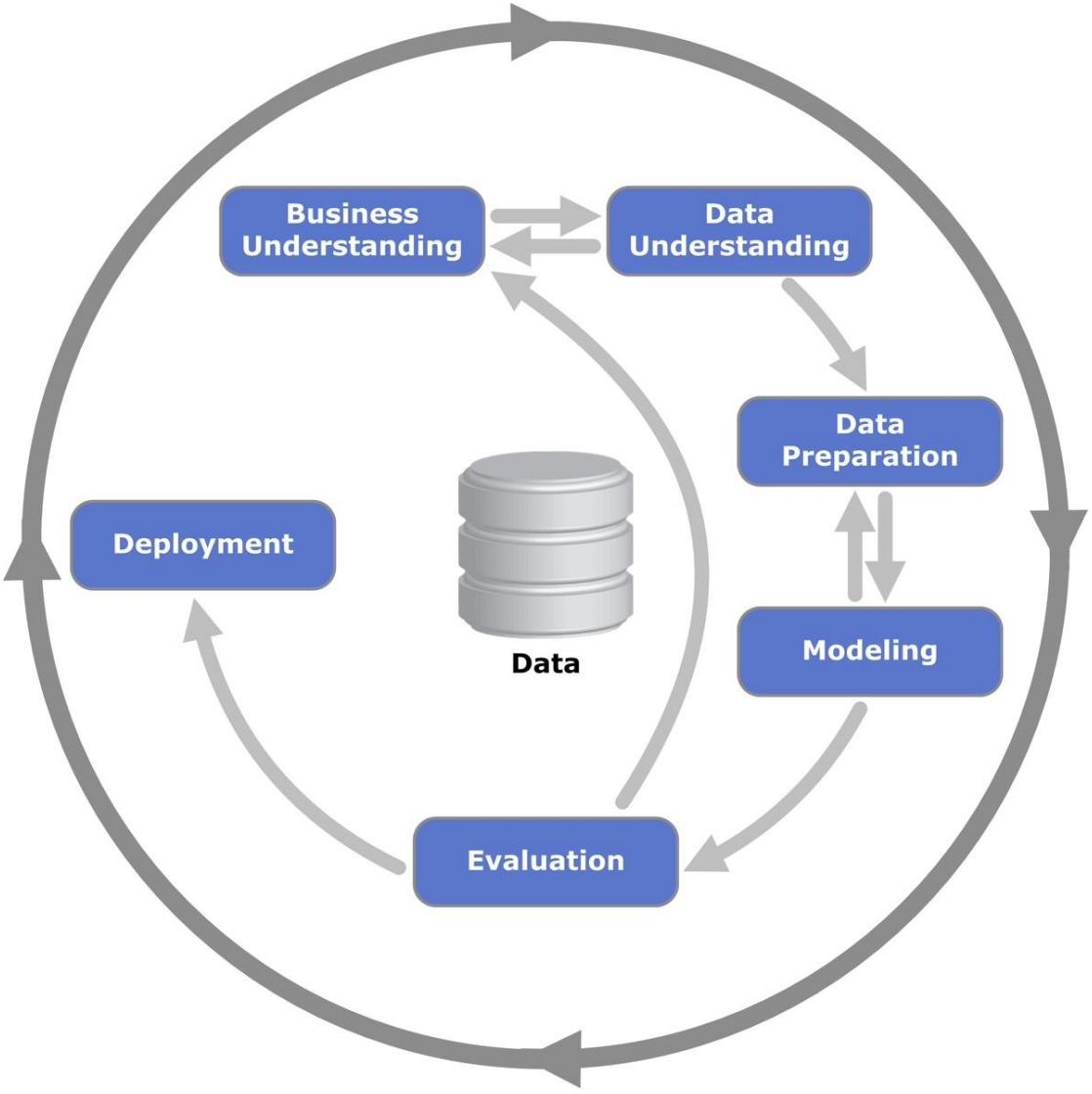


Figure 1.2: Cross Industry Standard Process for Data Mining (CRISP) tasks, Wikipedia 2016.

1.2 Stakeholder challenges

Solving these issues requires a complex approach, which introduces technical challenges, that are not easily tackled with standard industry approaches. This is even more so exacerbated, because of the growing requirement for more low-latency use-cases, where Organizations want to very quickly react to something a customer does. This means reacting to events in seconds as opposed to hours or days, which is commonly the case. There is a lack of unification on a logical and technical level between Data Scientists, IT departments and Business departments, as it is very unclear where the data comes from, what it looks

like, what it contains and how to process it in the context of existing systems and existing solutions to tackle such problems do not attempt to bridge this gap.

In this section we will outline the organizational and technical challenges that arise when developing Data-Driven projects while following the CRISP model for managing a projects life-cycle.

1.2.1 IT department viewpoint

In this section we will outline in more detail the challenges often faced by the IT department. When following the CRISP-DM methodology the IT department is usually most involved in the Data acquisition, Evaluation and Deployment stages. This is due the fact that the IT department is usually the owner of the data storage systems, processing systems and is in charge of the deployment, maintenance and operations of any newly developed project.

1.2.1.1 Data acquisition

When in the Data acquisition stage some major problems are:

1. For any organization with thousands of employees and large range of products, the number of systems, which are usually Databases that contain business data is in the dozens. It is very impractical to allow access to these sources directly to allow data acquisition. There is a large variety of systems and external systems suppliers for almost every systems and it is very uncommon for them to provide a unified interface. Vendors and Organizations usually have specific support agreements, which allow for a very specific type of interaction with the system. If an export interface is available, it is usually vendor specific. These interfaces apply some transformations to clean up the data and transform it to usable format. This formats often vary from vendor to vendor. A common approach is to export to a columnar format such as CSV, but then there is the issue of different vendors using different separators, because there is no real standard for this.

One solution to this is to use central storage systems, such as Enterprise Data Warehouses [5], which are central repositories of integrated data from one or more disparate sources.

They are used to stored current and historical data and are mostly used to create specific reports throughout the organization. This introduces another problem, specifically:

2. The inability to give developers access to raw data. DWH's usually only contain very specific processed views of the data. This serves the specific function of answering common queries on the data efficiently, but may essentially make the data useless for anything outside the scope of the specific use-case. On the other hand, the data still has to be stored in a structured and descriptive format and allow for over time changed, which is very challenging to do on a large scale. This is often the reason for the DWH containing transformed and aggregated views, as the the raw data structure has changed so much that any use-cases running on the new data would not function without a significant development or transformation of the input data. For most Data Science use cases this is not optimal, because structured transformation tends to remove some useful data as they reflect the needs of the application, which may or may not align with the goals of the data scientist.

To highlight this problem let's take the project describe in the previous section, that aims to analyze text descriptions of defects in different car models of an automotive manufacturer. These text descriptions also contain information about, which car models the issues are about. Now let's say the business department is interested in using this data to build a report of faults by car to analyze the efficiency of the Quality Assurance department, but some older models that are not produced anymore or have their names changed since the inception of the system. So the view in the data-warehouse should contain the actual names and only the specific models the business department is interested in. This works very well and provides the report that the business department is interested in. Now a Data Science project is started to analyze these descriptions and use them to predict possible future issues in newer models based on the problem description and historic data. The data in the Data Warehouse would be extremely biased towards specific problem for specific cars, it would also essentially not contain some models or contain ambiguity between the model dictionary the data scientist has and what the data contains. would lead to the Clustering and Classifications models not generalizing to the entirety of possible issues and cars.

Another problem, which is relatively new is the support for Low-Latency project:

3. In the classical approach a DWH is rebuilt periodically, most often once a day. This satisfies most of the requirement of reports that are currently common. The

problem with this is that this does not support quick interaction with customers. Currently organization are looking at engaging their customers as fast as possible, this usual means as soon as a customer does something, the organization wants to respond in a matter of seconds. This can be a product recommendation, a reaction to customer feedback and other potential use-cases. Any data contained in a DWH is then as late as the load period. There are certain technical challenges that arise with updating data in a low-latency setting, which we will outline in more detail in the following sections.

Another major problem, which is often ignored in a standard DWH setting are unstructured files:

4. Unstructured files are files, which do not have a fixed tabular or schematic representation, such as audio file, images, binary documents and other. These are most often not stored in a central system, such as a database, but are scattered around the departments and are handled in very specific ways. Most storage systems used by DWH's are not optimized for large binary file storage or processing, but rather data with fixed schema's and data types, which allow for optimizations of processing. This data is nonetheless immensely valuable when combined with user data, and the systems which can load and interpret this data, such as monitoring and operational systems are not designed to provide facilities for data export and analysis outside their specific context.

1.2.1.2 Evaluation

When in the evaluation stage the IT department has to provide all the data required to evaluate the model, sufficient resources to be able to train and analyse the model results as well as infrastructure for the Data Scientists to carry out this work in a secure environment. This leads to two main issues:

1. First the is the problem of scalability. The IT department has to provide an environment with enough processing resource to calculate often very complex models on a large amount of data. In large organizations this means potentially analysing TB's of information's daily, which when building a model on this data means significant overhead added to the just being able to read the data. While in the modelling stage Data Scientist can work with just a subset of the data, in the

Evaluation stage, the full set of data is required. As an example the Telematics project described in the previous section would mean processing approximately 30 TBs of data for just a month's worth of data. When considering, that multiple projects are being developed at the same time, this becomes a large issue. And the amount of data potentially doubles every year. This would make it impractical to allow this kind of calculations to run on any environment, that has any kind of business critical applications already running on top.

This problem could be addressed by fixed time windows when a Data Scientists can access the system and assigning priorities to the project. But this would lead to issues of multi-tenancy

2. As Data-Driven projects are highly dynamic and have to adapt to results and changes all the time allowing single access at specific times would hinder productivity and flexibility of the platform. At the same time allowing multiple users into the platform and allowing them to build and evaluate use-cases is on its own quite challenging. Not all models are the same and their resource and framework requirement are quite different, so there is a need to allow multiple users to access this platform concurrently without overlapping with other people's work.

1.2.1.3 Deployment

In the deployment stage the IT infrastructure usually faces problem of standardization and performance. As opposed to the Evaluation stage, the project is in a fixed position and the requirement for scalability is not as important. But even so it is often unclear how to run multiple projects at the same time with reproducible results. The models resource and framework requirement are not always clear as the use case has been developed by a specific team in their own environments, which usually differ vastly from the productive deployment environments. This means the IT department has the following problems:

1. It is often unclear how to go from the development environment of the Data-Driven project to testing or production. It is unclear what libraries and resources are required to actually run the application outside the development environment. Dependencies, configurations and resource specifications have to be supplied to the operations team, which has to somehow package this into a solution that conforms to the organizations standards and can be run in production.

2. Often the developed project has to be adapted to to meet the standards of the deployment and monitoring that the IT department has in place. This usually means having IT engineers collaborate with the Data-Scientist to define depends collection, resource definitions as well as Test cases.

1.2.2 Data Science team viewpoint

The teams off Data Scientists trying to build Data-Driven projects at the organization often face the same issues as well, as they are essentially the users of the storage and processing systems, which they use to develop their project, train their analytical models or generate reports. But there are some issues that are very specific to the way Data-Driven project are developed from the view point of Data Scientists. The problems faced by DataScientists that do not overlap with the IT department usually are encountered during the Business Understanding, Data Understanding and Modelling stages.

1.2.2.1 Business understanding and Data Understanding

When in the Business understanding stage, Data Scientist have to work with the business department to identify potential use-cases.

1. Large Organizations have a large number of Departments, which vary widely based on their size, project they take on and the way these projects are completed and documented. This leads a large variety of data sources, column names and documentation being created on the same subject by a large number of stakeholders from different Departments some of whom might not be part of the Organization anymore. To bridge this issue, there are large undertakings for an organization wide change management process, which pushes for standardization. On a technical level this changes translate to the centralization and standardization of project related documentation as well as rigid data views in a central database. Due to the complexity of the data and the Organization itself an Enterprise Data Warehouses or a Wiki-System cannot directly solve this issue, while satisfying all the requirements on structure and intelligence. This leads to the creation of Organization wide specialty tools, data/knowledge repositories and integration layers providing each Department

with access and management capabilities, in order to adapt to the specific requirements of the Organization.

Essentially what this entails is a full organizational restructure to create solution for data and information management. In reality, this is a vast and complex process and can cost a large amount of money and resources from the side of the Organization and in some cases might decrease productivity. Each individual Department has an approach of managing projects and in most cases such a monolithic system allows for less flexibility for individual Departments. This may lead to decreased productivity. It is often unrealistic to expect full and informed cooperation from each Department and its staff in order for each project to be documented, every data column described, every project contributor to be listed and all the interconnections between documents and data of multiple Departments being identified and documented. This is further complicated by the fact, that there is always more data and information each year, so all of the created documentation should be retroactively update periodically. The learning period for a complete change and standardization of such processes and the time required to update all of this information, can bring an entire Department to a halt for an extended period of time.

1.2.2.2 Modelling

When in the modelling stage, Data Science teams require an environment from where they can flexibly access the data they require and collaborate on using this data to build the project. Such environments have to generalize their parameters to the main functional requirements of the problem. This usually comes down to the following:

1. Modelling implies using a variety of statistical and machine learning models. There is a large amount fo such models and Data Scientists often have to make use of existing models and environment in order to quickly experiment and develop a model that solves their specific use cases. This means potentially using a large number of external 3d party package and frameworks. There has to be a defined way of fetching and using such packages.

2. Data Scientists teams have to have a well defined approach for collaborating and storing their results. Model results can be sensitive, so this has to be carried out in a secure fashion. No data-loss or breach by other teams can be allowed. Most

Organizations require for these projects to be developed in a fully on-premise environment, which greatly limits the approaches for handling multiple teams of Data Scientist working in a shared environment.

1.2.3 Business Department viewpoint

From the perspective of the business department being the most knowledgeable of business processes and what the data actually means in a business context, the issues more context and insight related. The Business department is often concerned with:

1. Making the life-cycle of developing a Data-Driven project more effective. This usually concerns making the research on the data contents, feasibility of the use-case and processing concern less time intensive for the teams.

2. Finding out if there is data that supports certain business use-cases they are interested in. This is need to make the business understanding phase of a Data-Driven project.

3. In a simple way of querying and exploring data by means of structure queries or analytical views. This is interesting for them in order to understand what effect their business decisions potentially informed by Data Scientist and Analysts, actually have. This is in most cases quite similar to what the Data Scientist and Analysts want as well. The key difference is, that the Business department is most interested in how effective all the Data-Driven projects are, be it the value they bring or in case of failed projects, how much resources were wasted.

It is very challenging to move ahead with Data-Driven projects without addressing these issues. Without access to a defined model to handle data exploration and processing, this leads to most of the project time being spent on actually finding out information about the existing data, finding people involved, data owners and where the data lies, as opposed to actual analysis. This can be quite costly time and resource and each stage of a Data-Driven project is impacted by this. This is particularly challenging for existing Enterprises with years of organizational structure and system already in place, as completely changing the way data is accumulated, handled, shared and used is not feasible.

1.3 Current Industry Approaches

In this section we will outline the current industry approaches and highlight some drawbacks these approaches have. We will specifically center on the issues described in the previous sections. We will summarize the different viewpoints into three technical categories:

- Storage and Processing
- Information retrieval and management
- Development Environments

As mentioned above we specifically concentrate on the aspects of these solutions, which hinder productivity and flexibility. Quite often these project involved external suppliers, which means they are inherently more expensive the longer the projects take. So in essence a project can fail on many different separate stages. Which might take months and can be very expensive and deliver few to no results. To this end the primary interest is in making the development of Data-Driven project more flexible and less time consuming, which would enable more Data-Driven project to be developed.

1.3.1 Storage and Processing

The answer would be to go to the actual data source and use the raw data, in its original form. This is often very complicated or even not possible due to the structure of the Enterprise and the way data ownership is handled and is very costly to implement on project by project basis. The currently accepted solution for this is to load all enterprise raw data into a single repository, a Data Lake [6]. A Data Lake is a method of storing data within a system that facilitates the collocation of data in variable schemas and structural forms, usually object blobs or files. Data Lakes are a popular way to store data in a modern enterprise. The usual architecture is fairly similar to a Data Warehouse, with the exception of almost all transformation. The main role of a Data Lake is to serve as a single point of truth, which can be used to create use cases, which join and analyse data from multiple departments. It addresses issues of scalable and affordable storage, while keeping raw data intact by loading the data unchanged into a distributed file system, like the Hadoop File

System [9] and provides and batch oriented integration layer for downstream consumers and use cases. This approach has a lot of merits, but most implementation lacks certain key aspects, which are more and more important for a modern business, such as self-describing data, tolerance to changes in the data source and support for low latency data sources.

1.3.2 Information Retrieval and Management

To bridge this issue, there are large undertakings for an organization wide change management process, which pushes for standardization. On a technical level this changes translate to the centralization and standardization of project related documentation as well as rigid data views in a central database. Due to the complexity of the data and the Organization itself an Enterprise Data Warehouses [5] or a Wiki-System [10] cannot directly solve this issue, while satisfying all the requirements on structure and intelligence. This leads to the creation of Organization wide specialty tools, data/knowledge repositories and integration layers providing each Department with access and management capabilities, in order to adapt to the specific requirements of the Organization [4].

Essentially what this entails is a full organizational restructure to create solution for data and information management. In reality, this is a vast and complex process and can cost a large amount of money and resources from the side of the Organization and in some cases might decrease productivity. Each individual Department has an approach of managing projects and in most cases such a monolithic system allows for less flexibility for individual Departments. This may lead to decreased productivity. It is often unrealistic to expect full and informed cooperation from each Department and its staff in order for each project to be documented, every data column described, every project contributor to be listed and all the interconnections between documents and data of multiple Departments being identified and documented. This is further complicated by the fact, that there is always more data and information each year, so all of the created documentation should be retroactively update periodically. The learning period for a complete change and standardization of such processes and the time required to update all of this information, can bring an entire Department to a halt for an extended period of time.

There are also a large number of technical considerations, such as file format support, performance of the search and indexing. Data Scientists and the Business Department would like to see changes to any information as fast as possible.

1.3.3 Approach for Development environments

Projects are developed in environments, which first of all usually reflects internal requirements of the project. In our case project vary greatly in their complexity and the amount of data they process. So a environment is required that can generalize their parameters to the main functional requirements of the problem. A very important point to note in this context is that most Organizations require for these project to be developed in a fully on-premise secure environment, which greatly limits what can be done towards building a scalable solution. This means we need to design the environment to fit the largest possible problem, that would need to be solved, which entails processing terabytes of data from many disparate systems.

To this end one practical solution used in the industry is building clusters of many machines, that can handle these types of workloads [11] [12]. In a modern Organization this usually means using the Hadoop ecosystem [9] to create a generalized processing environment with support for most data types and approach to analysis. This usually goes hand in hand with a Data Lake implementation [6].

Hadoop clusters are immensely powerful and flexible, but suffer from a number of oversights. Hadoop environment usually do not provide a very flexible way for dozens of users to access the systems in parallel. The usual approach for this is to provide an Edge Server [12], which is a secured server, that allows for selective access to certain resources of the cluster. There are usually a set number of such server per cluster and are not necessarily very powerful servers. The idea behind such servers is, that they should only be used to deploy job to the cluster and not carry out any calculation themselves. This is more geared towards Data-Engineers deploying applications and not an analysis environment, where Data Scientists have to explore the data and retrieve results in an ad-hoc manner. For this analysis many routine libraries and tools are often required and this leads to the Edge Servers running out of resources and contain a large number of potentially clashing software as these are shared environments. As more and more users start using these servers, this becomes an impractical approach.

To address this there are approaches to use vitalization to provide on demand environments for each individual team [13] [14]. Due to vitalization overhead multiple teams need to share these machines, which are often over-provisioned and have much more resource than it is required as reconfiguring such machines is usually a lengthy administrative task. Requiring

multiple teams to share a virtual environment is also the simplest way to enable collaboration as individual developers can access the code, results or services of other team members. But this leads to the resource of the machines not being fully utilized and decreased performance when the project requires a large number of services or processing workloads, that do not run on the cluster.

Another downside to such approaches is the complexity it brings to testing and deploying such projects, once development has finished. It is unclear what libraries and resources are required to actually run the application outside the development environment. Dependencies, configurations and resource specifications have to be supplied to the operations team, which has to somehow package this into a solution that conforms to the organizations standards and can be run in production. Such Edge Servers are not self-describing and fully rely on the development team to deliver well documented, testable, deployable code. This is often not simple to do due to the multidisciplinary and complex nature of Data-Driven projects. What is required is a self-describing environment, that is simple to ship into production. It is possible to directly deploy the vitalized edge-nodes with all the required software inside them, but this is quite expensive and slow, when working in an on premise environment as such approached are too rigid in their requirements.

Chapter 2

Data Storage Management and Processing

In this chapter we will outline how we address the problems outlined in the previous chapter, specifically the problem with the standard approach of building an organization wide raw data repository using the data lake approach as well as managing a large number of multi-disciplinary user accessing this data and creating data-driven projects.

2.1 Data Management requirements

As described in the previous chapter managing a large amount of data coming from various systems in the organization is fairy non trivial. Specifically based on the issues we described in the previous chapters a list of requirements has been gathered, that reflects most of these issues and challenges:

- **Access to full historical data for all systems.** As described in the previous chapters it is commonly impractical to keep a full historical archive of data from all system. The volume of the data and the different forms it appears in make it impractical to store this in a central database, but just saving the data to a file-system is also not enough. Data-Driven applications greatly depend on mixing data from various systems and thus access to this is of great importance.

  Access to Raw representation of data. The gathered data needs to be in its rawest possible form. Preserving the raw form of data and the relationships of data is of great importance, because of the examples described in the previous chapters.

- **Access to predefined, structured views of the data.** While raw data has to be accessible, Data Scientists need to be able to execute simple predefined queries on the data, preserving the relationships similarly to a classical Data Warehouse solution.

- **Low Latency Data ingest.** As mentioned in the previous chapter it has become very important for organizations to support low-latency use-cases. Such use-cases should support detecting almost real-time interactions between customers and the organization. This means the incoming data has to be made accessible to consumers as fast as possible.

- **Low Latency Data Views.** The incoming data has to be query-able with structured queries as fast as it comes in. This means certain views on the raw incoming data have to be made available for querying. Not all data is suitable for this, but it would be immensely valuable for some use cases.

- **Support for large scale data ingest.** Large organization have a very large number of systems and with customer interactions increasing every year, the amount of data can also increase to very large proportions. This means the data collection and view creation pipeline has to be scalable as well.

- **Homogeneous Low-Maintenance data-pipeline supporting a strict low-latency semantic.** This pipeline needs to be low maintenance, scalable and fault-tolerant as possible. With more and more systems being created all the time the pipeline should support a flexible addition of new data sources. The operations of such pipelines can become exceedingly expensive and it is a very important requirement for the system to be self-managing and require little to no intervention. This means supporting a self-scalable and fault tolerant pipeline for data ingestion.

2.2 Data-Driven project development environment requirements

As described in the previous chapters managing a large number of user accessing a large amount of data and working in teams to analyse this data is a complex problem to tackle. Specifically based on the issues we described in the previous chapters a list of requirements has been gathered, that reflects most of these issues and challenges:

- **Single-User Secure isolated environments.** It is beneficial for users to have fully isolated environments as this simplifies development and later on deployment. Because of this environment need to be fully isolated and secured in accordance with the Organizations requirements.

- **Support a large variety of demanding workloads.** Such environment should support all types of workloads, that can be submitted to the Hadoop cluster as well as ran locally inside the environment itself, which is essential for ad-hoc analysis. Such environments should also support being used as servers for tools required by other projects. For example a standalone Database deployment.

- **Managing Resources in a highly multi-tenant environment.** We need flexibility to manage many of such instances efficiently, potentially on very limited hardware. Environments should consist of services based on required resource allocation

- **Collaboration tools.** The environment should have in-built tools that enable collaboration. This is very important to enable well documented, testable projects and code.

- **Scalability and Fault-tolerance.** It should be straightforward to scale individual environments up and down on demand, in perspective automatised. As it is important to ensure uninterrupted work and no loss of work data, such environment should be Fault-Tolerant.

- **Going to Test, Production quickly after development.** The environments should be self-descriptive and flexible enough to speed up this process and not introduce more technical hurdles.

  Flexible access to all stored Data. Because this is an environment for a Data-Driven project, the most important thing is access to all required data, but this access has to be customizable as to provide fun grained access for teams and individual developers.

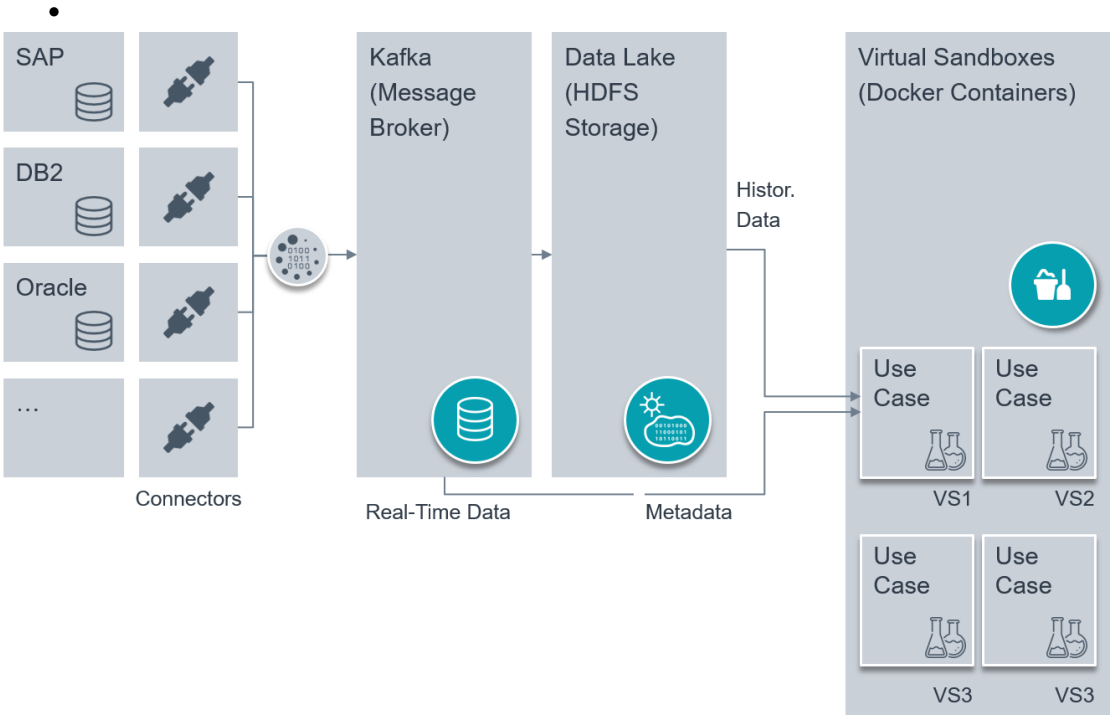


Figure 2.1: High Level Architecture for Data Sandbox Environment.

This is not an exhaustive list, but it outlines the basic requirements such a system must fulfill from the view point of teams working on Data-Driven projects. It is a list of complex functional requirements which Project, System Developers and Designers need to map to a technical problem definition and implementation. An architecture, which contains tools that satisfy these requirements would allow to create a Data Sandbox environment for data analysis, processing and the development of Data-Drive projects. The proposed architecture is outlined in 2.1 In this chapter this architecture will be outlined in detail.

2.3 Data Reservoir

In contrast to the commonly accepted practice of just ingesting all the data as is into separate parts of the file-system and then transforming it into a meaningful state, our approach to model the data in a more structured as we impose the format, structure and extraction semantics, but we still remain flexible as the data is still ingested in almost raw form and Data Vault modelling is only applied to sources for which it makes sense. In the context of the platform we want to support all kinds/structures of data (so called Poly-Structured) can be extracted and ingested in the Data Reservoir architecture. Besides structured data coming from any kind of database also, semi-structured and also unstructured data such as log files, audio-files, images, websites and social-media.

Our approach is based on 6 layers:

1. Ingest
2. Store
3. Organize
4. Analyse
5. Process
6. Decide

We will outline the specific layers in more detail

2.3.1 Ingest

Ingestion in the context of the Data Reservoir architecture means extracting the data from the data sources and loading it into the Raw Data Reservoir in the raw format. No transformation or modification will be done on these data pipeline.

The objectives of the ingestion layer are:

- To get all data with all kinds of structure
- To extract the data incrementally and in real-time if possible
- To unify data as messages

  Temporally saving it in a fault tolerant architecture
- To provide an architecture where many consumers can consume messages in parallel without performance bottlenecks

More details regarding technical implementation details will be highlighted later in this chapter.

-

2.3.2 Store

Data extracted from data-source is stored in partitioned in a file-system. This is called the Raw Data Reservoir layer contains the full history of all data which has been extracted. Using a file-system here is advantageous as arbitrary binary storage formats can be used. Rigid typed views can be created on this data at a later point. We structure the data according to the following schema:

RawDataReservoir/DATATYPE/SOURCE/TABLE/2016/06/30/*.type (2.1)

Respectively every day a new folder with new and updated data will be available.

The objectives of the Raw Data Reservoir part are

- To store all kinds of data in a same format
- Extract schema information
- To be able to get access to the full history

More technical details on the implementation will be provided in the next sections.

2.3.3 Data Hub Layer

The Data stored in low-latency in the data reservoir has to be structured in a meaningful way for actual business application to benefit from them.

Views on the stored data are created based on the business demand, existing project, some analysis or processing a Data Scientist carried or a generic view of the raw data. This might be a static periodic view or a lowlatency streaming view integrated into a serving layer as described above. In the Data Hub Layer the data will be modelled, shaped and enriched with business transformation. This will be achieved by implementing the Data Vault 2.0 [15] architecture. The Data Vault model will be implemented in a Database, optimized for analytical requests.

The objective of the Data Hub Layer is:

- To model the different data from different data sources in a
- To create a global model
- To have an easy way to extend the data model easily without impacts
- To access this data over structured queries

2.3.3.1 Data Hub

The Data Hub based on the Data vault modelling is a database modelling method that is designed to provide a modular way to incorporate multiple operational systems [15]. It is also a method of looking at historical data that, apart from the modelling aspect, deals with issues such as auditing, tracing of data, loading speed and resilience to change, which is critical for the Data acquisition and Processing steps.

Data vault contains three different types of technical entities

1. **Hubs**. Hubs contain a list of unique business keys with low propensity to change.
2. **Links**. Associations or transactions between business keys (relating for instance the hubs for customer and product with each other through the purchase transaction) are modeled using link tables.
3. **Satellite**. The hubs and links form the structure of the model, but have no temporal attributes and hold no descriptive attributes. These are stored in separate tables called satellites.

All the principal data sources, such as CRM data, Telematics data, Accounting Data, Product detail, Quality Assurance data are modelled in the Data Vault approach and exposed as views and APIs. This model is organization wide and usually involves modelling the same entities across countries or regions of business. As an example lets highlight the interconnection of the

Satellites, Links and Hubs in Fig. 2.2

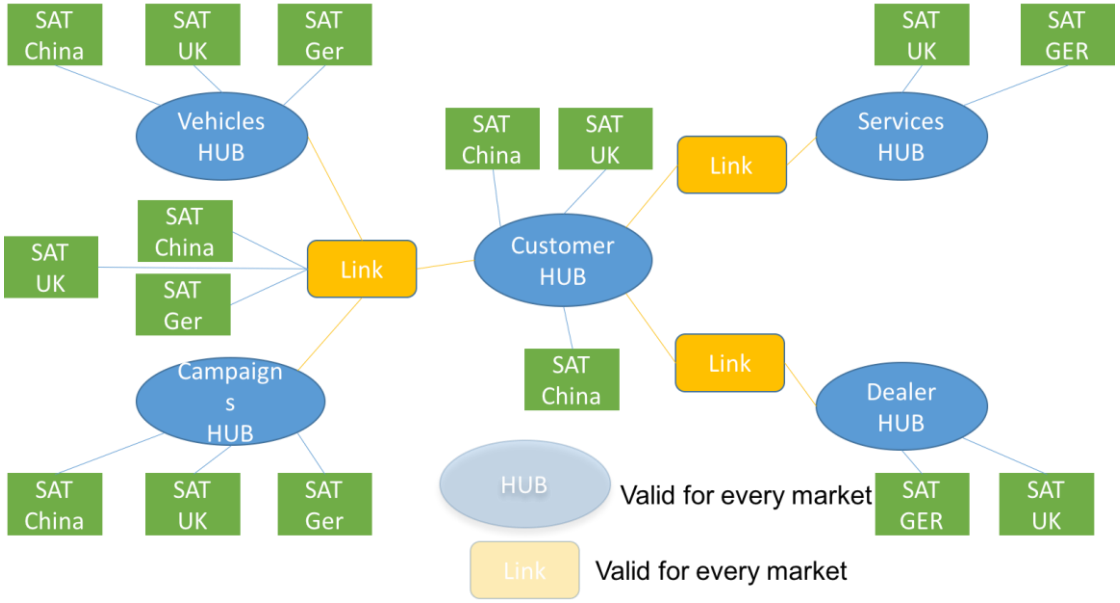


Figure 2.2: Satellites, Links and Hubs

2.3.4 Organize

The data has to be organized and transformed to reflect the business requirements. The Business transformations can be implemented in any transformation tool.

The objective of the Organize layer is:

- To transform data

The entities will be modelled similarly to the Data Hub. The relation between the two is highlighted in Fig. 2.3 More details regarding the technical implementation will be highlighted later in this section.

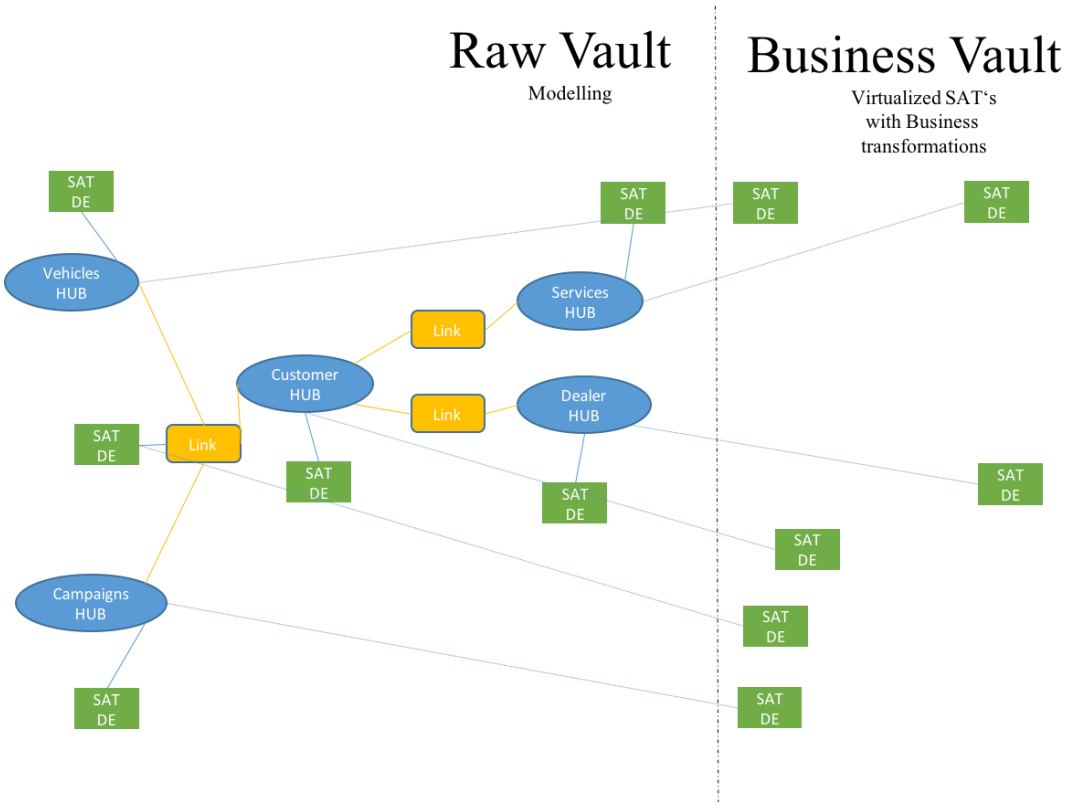


Figure 2.3: Organized Business satellites

2.3.5 Analyse

The analytics environment is an environment, which can be setup on demand. The analytics environment contains pre-installed analytical components and does have access to the Raw Data Reservoir and the Data Hub
Layer.

The Data Scientist gets a distributed environment to process huge datasets with complex and heavy algorithms. The processed results created by the Data Scientist in this environment will be made available in the Application Layer, where Decision tools can access them.

The objectives of the Analytical Environment layer are:

- To get a distributed analytical processing environment on a push of a button
- To get access to raw as well as to modelled data
- To quickly find important insights
- To use machine learning models to implement analytical analysis.

This analytical environment will be highlighted in more details in the next sections. This is an integral part of the platform and as outlined in the previous chapter is not fully addressed in commonly implemented approaches.

2.3.6 Decision Environment

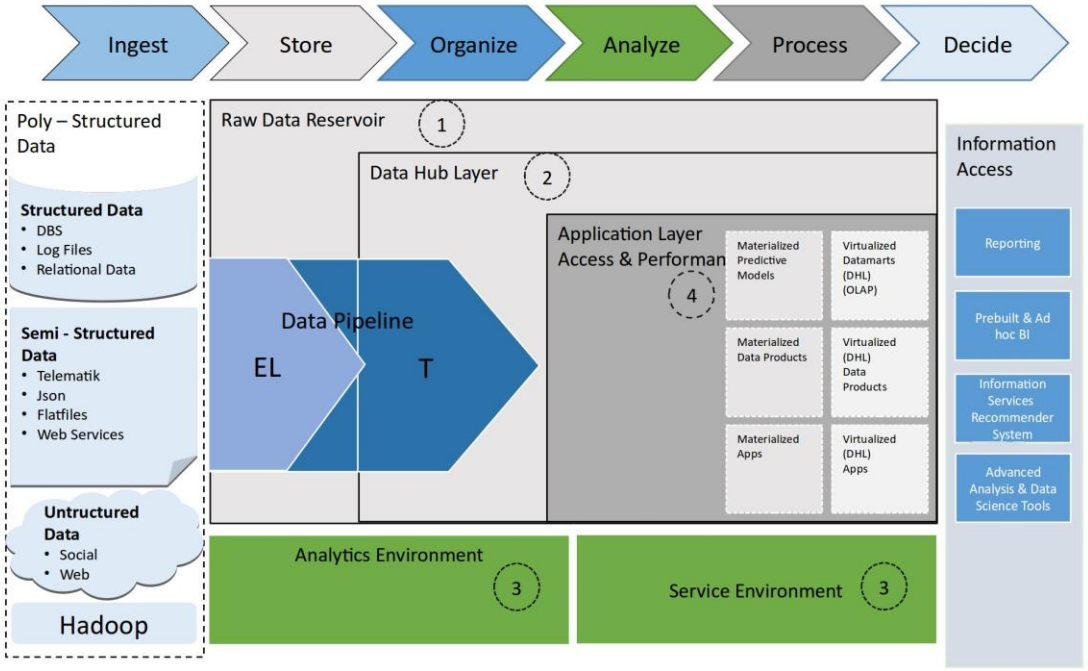


Figure 2.4: Data reservoir layers.

The Decision Environment is similar to the Analytical Environment with the difference that it won't be shut down. This environment where all the projects, that were created in the analytical environment are ran after development and testing.

The objectives of the Decision Environment layer are:

- To implement Machine Learning Models
- To stream latest data in order to derive better Models

More details will be highlighted in the next sections.

2.3.7 Application Layer

The Application Layer is the delivery layer for the Decision Layer. Results are processed and performance optimized.

The objectives of the Application Layer are:

- Provide an access layer for the Decision Tools
- Provide better Performance for the Decision Tools
- Handle the access to the data for the Decision Tools More details will be highlighted in the next sections.

2.3.8 Decide

Decision is the last step, where Tools support the user to make an decision on the data products implemented in the Data Reservoir. The respective tools in this platform depend on the organization itself.

The objectives of the Decide layer are:

- Illustrate the results over graphs
- Support the user to make decisions

More details will be highlighted in the next sections.

The overall structure is outlined in 2.4. It includes all the Layers from Ingestion to Serving (Decision) layer and based on our experience key components are the Lambda Architecture underpinning this and the Organizational Hub layer, modelled as a Data Vault. The Ingest, Store, Process and Analyse are layers, which are mostly based on a variation of a Lambda Architecture and also include the Raw Data Storage layers, which is essentially a Data Lake implementation. We will outline this architecture in more detail in the next sections.

2.4 Low-Latency Architectures for Data Ingestion and Modelling

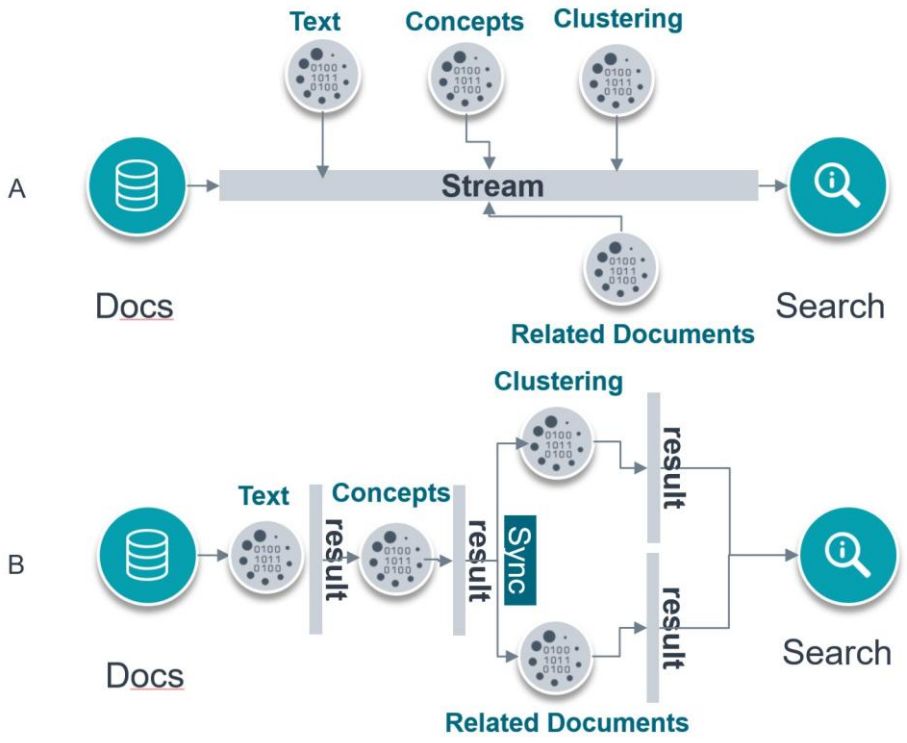


Figure 2.5: (a) Data-flow requirements in the context of stream processing. (b) Data-flow requirements in the context of batch processing.

The combination of real-time data ingestion, processing and Data Vault modelling leads to a simple and flexible Data Model, which solves the problem we posed in this section in a more robust way as compared to the Data-Lake model. Implementing it, while supporting the defined requirements present many technical challenges. There are significant architectural and algorithmic considerations when mapping the requirement set outlined in the previous chapter to a technical implementation. To achieve all the requirements for performance this has to be distributed application running on multiple machines, which introduce a large set of benefits and problems.

First let's consider the architectural side of the problem, before outlining how the required information is extracted and modelled. To outline the architecture, we first need to define, what input we have and what required output has to be produced.

Input. Our input is collections of data each coming from each system of the organization. These collections are unbounded and may increase and decrease as well as change over

time, and so can the data sources. Currently a single collection can contain hundreds of tables with thousands of columns.

Additionally, such a system should also be ready to process raw organizational data. This would be important to analyze the actual data content as opposed to definitions. This can become extremely large with terabytes of data coming every month.

Output. As the output the raw data and aggregated views are required. This information should be exposed through an interface, that supports direct access as well as structured queries.

Based on the requirements we need to find a way to get from the input to the output as fast as possible, whilst applying a non-fixed number of potentially process intensive transformation functions in a sequence on an unbounded collection of data. Additionally, the latency of processing, should not be affected by the amount of data in a collection at any point in time, meaning there should be a defined way of scaling each step independent of the others. It is an accepted approach in the Industry to frame this problem in the context of stream processing [16][17] as opposed to a periodic batch process. In stream processing each collection of events can be represented as a separate stream of data, which changes constantly over time. Each new element of the stream is processed one by one, out-of-order in parallel. The output of each transformation is represented as an another stream of data and can be directly consumed by another transformation.

This is quite different from a periodic batch approach, where each collection is considered static at the point in time and all transformations have to be applied to the entirety of collection. This is disadvantageous for our case as certain transformations can depend on others, which means a transformation has to be applied on all data before proceeding to the next step. Because This means we need either join the transformations together into more monolithic blocks or create large intermediary collections of transformed data and orchestrate the order of the batch jobs. In our experience, this is highly impractical, as this entails either very large, not flexible transformation steps or a complex scheduling problem. Such problems are very tricky to tune to scale correctly. Not to mention the fact, that this process by definition is high latency, but this of course depends on the amount of data and the complexity of transformations at any point in time. The difference between the two approaches is highlighted in Fig.2.5. The main issue comes from the necessity to sync between transformations. A transformation has to be notified that a previous step has been completed. This introduces complexity and latency.

In our approach we settled on decreasing the granularity of data slices and define a stream of data. This in our opinion allows for flexibility of adding and removing steps at essentially any point. As opposed to only passing simple events around [18] we model the data as an unbounded stream, so we can processes these elements on-by-one or in larger batches as well without intermediate storage. This greatly simplifies the amount of state that has to be tracked in the entire solution and as a side effect, in our experience, leads to more concise, well performing and testable services. But we still need to define an approach for scaling this type of solution and more importantly ensuring a certain amount of automatic fault-tolerance to make the solution as flexibly and low-maintenance as possible.

For this we adopt a variation of the Lambda Architecture, which is an Architecture appropriate in our context of data processing.

2.4.1 Lambda Architecture

The Lambda architecture is a data-processing architecture designed to handle massive quantities of data by taking advantage of both batch- and streamprocessing methods. This approach to system architecture, used in our context, attempts to balance latency, throughput, and fault-tolerance by using batch processing to provide comprehensive and accurate views of batch data, while simultaneously using real-time stream processing to provide views of incoming data. The two view outputs can be joined before presentation [16].

In a classical Lambda Architecture [19]:

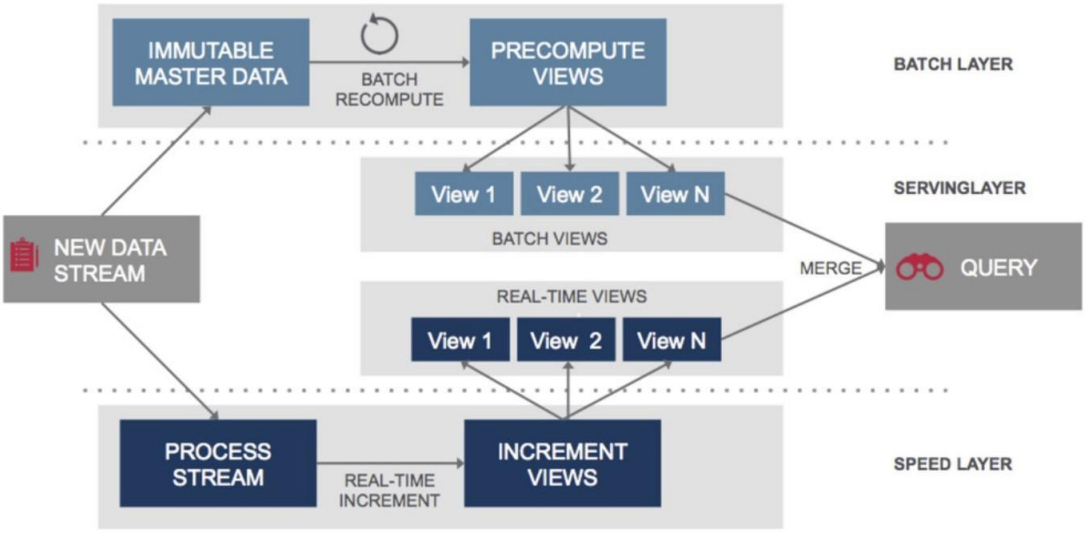


Figure 2.6: Lambda Architecture overview.

Source: https://www.mapr.com/developercentral/lambda-architecture

1. All data entering the system is dispatched to both the batch layer and the speed layer for processing.
2. The batch layer has two functions
   (a) Managing the master dataset (an immutable, append-only set of raw data)
   (b) Pre-compute the batch views.
3. The serving layer indexes the batch views so that they can be queried in low-latency, ad-hoc way.
4. The speed layer compensates for the high latency of updates to the serving layer and deals with recent data only.
5. Any incoming query can be answered by merging results from batch views and real-time views.

The main rational for a Lambda Architecture is to efficiently answer a query over the entire dataset. The challenge is that running arbitrary functions of an unbounded set of data is very expensive. Thus the Lambda Architecture decouples this two processes and offloads only efficient simple queries to the real-time layer as outlined in 2.6.

To achieve this the Lambda Architecture models the ingestion layer as a stream of data, which are ingested into a Distributed Message Queue, which allows to both create an Immutable Master dataset, which is an append only historical log of all events and transactions, and a real-time layer, which answers very specific questions on the incoming stream of data. Having an Immutable set of data not only allows to easily build up analytical usecases, as well as creating reliable point in time snapshots of data, which allows to recreate the full dataset at any point in time. Most commonly in batch view in the Lambda architecture is where the data persistent to a filesystem takes place, this is a simplification as achieving low latency exactly once persistence is quite challenging. So the persistence layer is essentially part of the processing layer.

2.4.2 Dynamic Streaming Architecture

We have certain variations from the normal Lambda Architecture. Specifically, the ingestion layer is fully decoupled from the processing layer and also supports low-latency incremental persistence to HDFS as opposed to a scheduled batch job, as in the standard Lambda Architecture. [16]. This allows the ingestion layer to be orthogonal to the actual views that have to build on the data. In practice this turns out to be most beneficial for large Organizations, where it is sometimes unclear what the default views and queries are before analysing the data. This also allows for the same degree of consistency offered by the Lambda Architecture, but with less latency.

To implement the Lambda Architecture, we require the batch layer, real-time layer, storage layer and the view layer. In our architecture we will define a durable stream processing approach, which is flexible to both real-time and batch cases. Durable in this content means scalable and fault-tolerant. There is a significant connection between Fault-Tolerance and Scalability. Scalability is usually most directly solved by trying to parallelize the costly process, most often, by launching more instances of the same service and balancing the work between each instance. But according to the CAP theorem, which states that it is impossible for a distributed computer system to simultaneously provide Consistency (all nodes see the same data at the same time), Availability (every request receives a response about whether it succeeded or failed) and Partition tolerance (the system continues to operate despite arbitrary partitioning due to network failures.

To this end the most important architectural block of a streaming processing systems is the representation of the "stream". As we described we need a proven way of storing unbounded amounts of information, which has to be tolerant to failures and scale to a large number of events and services.

2.4.2.1 Stream component

A widely accepted approach to this is using a distributed message broker, such as Apache Kafka [20], which is a high throughput, distributed and durable messaging system. Each stream can be represented as a topic in Kafka. A Topic is a partitioned log of events. Each partition is an ordered, immutable sequence of messages that is continually appended to a commit log. The messages in the partitions are each assigned a sequential id number called the offset that uniquely identifies each message within the partition [20]. This is a flexible representation as it allows us to create many topics per stage, which can be consumed by processors in a stage(consumers). Each Kafka consumers is part of a consumer group and Kafka itself is balancing the partitions in the topic over the consumer processes in each group. This means that it is simple to achieve processing scalability just by launching more consumer processes. As data is automatically distributed and consumption is balanced, Availability is straightforwardly addressed by dynamically launching more instances of the service and relying on a balancing mechanism. On the other hand, as described in [21] [22] streaming systems are most sensitive to Partitioning and Consistency problems.

The generally accepted approach to try and solve this [23] [1] is essentially based on offset tracking. Offset tracking is tracking how far along in the stream the processing service is. If each instance of the service has to make sure it processed a single or batch of event before requesting new events. It would be quite expensive to commit such offsets for every event, so offsets are usually committed is small batches. This is supported as a core functionality within Kafka itself, which offers approaches to store offsets in a separate topic as well as allows for automatic batching of events. Using Kafka, we introduce a scalable and fault tolerant representation of a stream, both on the storage and processing level.

2.4.2.2 Service Scheduling component

With the described approach each service is only responsible for deciding how many events to process and from which offsets to start and work balancing, storage and transfer are handled by the stream representation. Kafka has no control over the consumer processes as well has no idea how costly each operation or what state(offsets) have to be tracked. This means it is simple to scale processes just by launching more processes, but making sure all events were processes correctly during failures, as well as quickly is up to the consumer itself. To this end we require a defined way for dynamically scaling process as well as restarting and retrying in case of consumer outage.

To solve this issue, we can exploit the fact, that with the proposed way of handling data storage and state, each service is essentially immutable. Which essentially means a crashed service does not need to be restarted or recovered, but we can just start a new copy, which will catch up to the state the previous service was automatically. Coupled with the fact, that we have a large number of machines at our disposal, the problem of service fault tolerance and scaling becomes a process scheduling problem. Technically we have a set amount of resources for each stage, each stage takes a certain amount of time to compute and we are trying to reach a certain amount of latency. Using this information, we can approximate how many processes have to run and where in order to achieve the required latency. We require a component, that can decide whether to start, stop or restart processes.

This is an accepted approach to handle scaling large-scale distributed application and has been successfully implemented in the industry [24] [25] by means of a global resource manager. Such a resource manager is essentially a higher level version of an Operating Systems Kernel, but running on many machines. Applications decide based on their processing time and requirement, what resources they require and the resource manager tries to accommodate this. If the service crashes or the computing node fails, it can be transparently restarted somewhere else assuming the application can transparently handle something like this. Which our proposed approach can. Here we use Apache Mesos, a commonly accepted High-Availability implementation of such a system. Such a system has also proven to be very flexible to new services or any adaption [26]. The system just needs to know what resource and what process to run.

2.4.3 View component and Architecture Implementation

The last thing missing from our architecture is the "view" layer. This is where the output of the processing stages is stored in its final form, as well as any extra data. Based on the outlined requirements we need a flexible and scalable storage systems, that supports a large scale of analytical queries. In our system we need to support structured queries, raw data access and text-based search queries. To this end an analytical database or storage system can be used depending on the project requirements. For example the Elasticsearch [27] is used to provide text-based searches for incoming data.

Based on this we can define our general purpose architecture for building a scalable and Fault-Tolerant stage based Event-Driven application for the purpose of extracting information.

2.5 Technical Architecture

2.5.1 Data extraction framework

For the purposes of a scalable low-latency extraction a framework is defined based on the existing Kafka-Connect framework [1], which allows the solution to satisfy the defined requirements. The implemented connectors are based on Kafka-Connect and share the three major architectural models as defined in [1]:

- Connector model: A connector is defined by specifying a Connector class and configuration options to control what data is copied and how

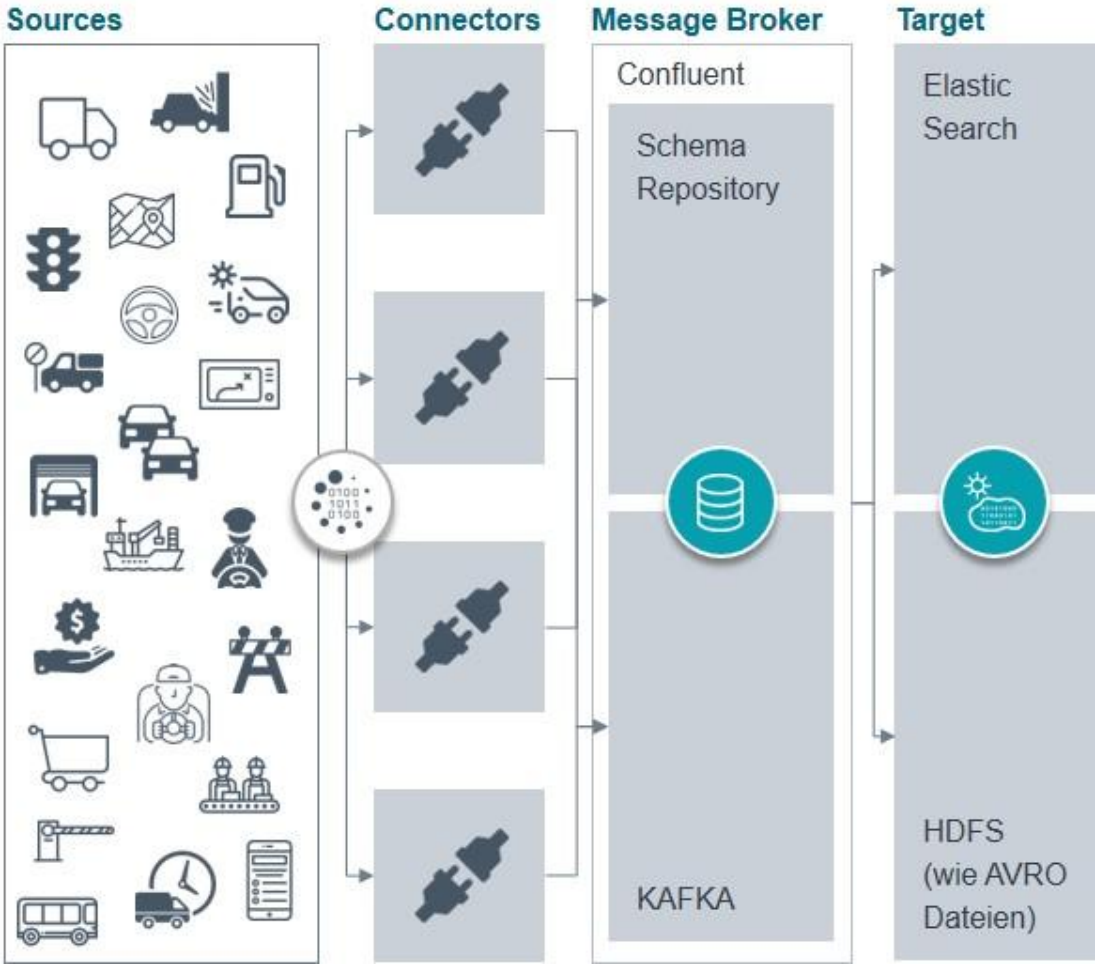


Figure 2.7: Data Ingestion Architecture.

to format it. Each Connector instance is responsible for defining and updating a set of Tasks that actually copy the data. Kafka Connect manages the Tasks; the Connector is only responsible for generating the set of Tasks and indicating to the framework when they need to be updated. Source and Sink Connectors/Tasks are distinguished in the API to ensure the simplest possible API for both.

- Worker model: A Kafka Connect cluster consists of a set of Worker processes that are containers that execute Connectors and Tasks. Workers automatically coordinate with each other to distribute work and provide scalability and fault tolerance. The Workers will distribute work among any available processes, but are not responsible for management of the processes; any process management strategy can be used for Workers (e.g. cluster management tools like YARN or Mesos, configuration management tools like Chef or Puppet, or direct management

  of process lifecycles).

- Data model: Connectors copy streams of messages from a partitioned input stream to a partitioned output stream, where at least one of the input or output is always Kafka. Each of these streams is an ordered set messages where each message has an associated offset. The format and semantics of these offsets are defined by the Connector to support integration with a wide variety of systems; however, to achieve certain delivery semantics in the face of faults requires that offsets are unique within a stream and streams can seek to arbitrary offsets. The message contents are represented by Connectors in a serialization-agnostic format, and Kafka Connect supports pluggable Converters for storing this data in a variety of serialization formats. Schemas are built-in, allowing important metadata about the format of messages to be propagated through complex data pipelines. However, schema-free data can also be use when a schema is simply unavailable.

There are many alternative approaches to loading data, batch of real time into similar systems, such as [28] [29] [30]. Such systems are often either very specific [30], depend on local state for fault tolerance [29] and due to the complex centralized pipeline, do not offer required delivery guarantees. Compared to these Kafka-connect was chosen as the standard framework to implement a range of connectors on due to the following criteria, which are necessary to satisfy the requirement:

- Streaming and batch – Support
- Low-Maintenance scaling
- Fault-tolerance based on a centralized store
- Immutability of a single running connector
- Delivery guarantees

The implemented connectors usually are of two types, Source and Sink. Sinks are used to transfer data from the Kafka stream representation to another system, such as HDFS or a Elasticsearch. And Sources, which are used to extract data from external systems, such as Databases, FileSystems and Websites.

2.5.1.1 Connector

A connector is essentially a meta-definition for extracting data from a system or writing to a system. Each connector can contain multiple definitions of jobs, which are then divided into a set of Tasks that can be distributed to Kafka Connect workers, which can be running anywhere in the datacenter. A single Task is responsible for the transfer of subset of data to or from Kafka, this is highlighted in Fig 2.8. This assignment is transparently handled by using Kafka as synchronization mechanism. A subset of data that a connector copies must be represented as a partitioned stream, similar to the model of a Kafka topic, where each partition is an ordered sequence of records with offsets. Each task is assigned a subset of the partitions to process. Sometimes this mapping is clear: each file in a set of log files can be considered a partition, each line within a file is a record, and offsets are simply the position in the file. In other cases, it may require a bit more effort to map to this model One possible mapping uses a timestamp column to generate queries to incrementally return new data, and the last queried timestamp can be used as the offset [1].

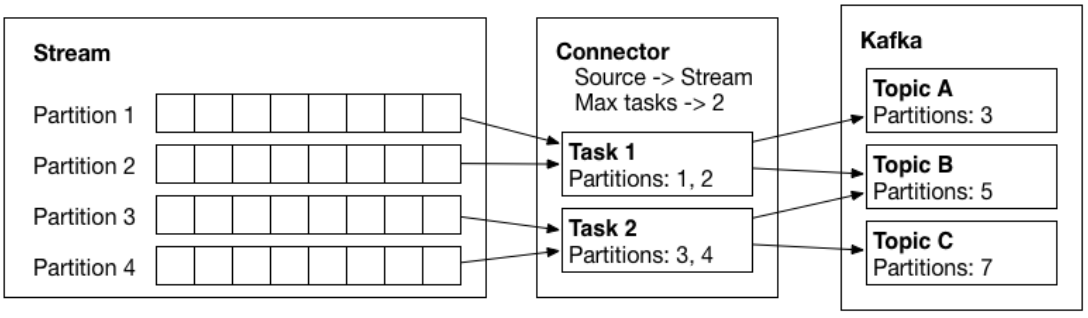


Figure 2.8: Example of a source connector which has created two tasks, which copy data from input partitions and write records to Kafka [1].

Source: http://docs.confluent.io/2.0.0/connect/devguide.html

To this end the challenge of implementing different connectors is modelling the correct data partitioning and tracking these offset. This is the most important aspect of the framework and as each datum receives a unique offset, so in most cases the connector can be implemented to guarantee exactlyonce data delivery, where each piece of data is only written once. This is important to avoid duplicates, which could pose a problem for downstream consumers, such as Data Scientists.

2.5.1.2 Partitioning

Each partition is an ordered sequence of key-value records. Both the keys and values can have complex structures. In our cases these are usually binary Avro records. In order to track the structure and compatibility of records in partitions, Schemas may be included with each record.

Each record also has an attached partition Id's and offsets. These are periodically committed to Kafka to keep track of the data that has been successfully written. In the event of a failure, processing can resume from the last committed offsets, avoiding unnecessary reprocessing and duplication of events.
This is highlighted in Fig. 2.9

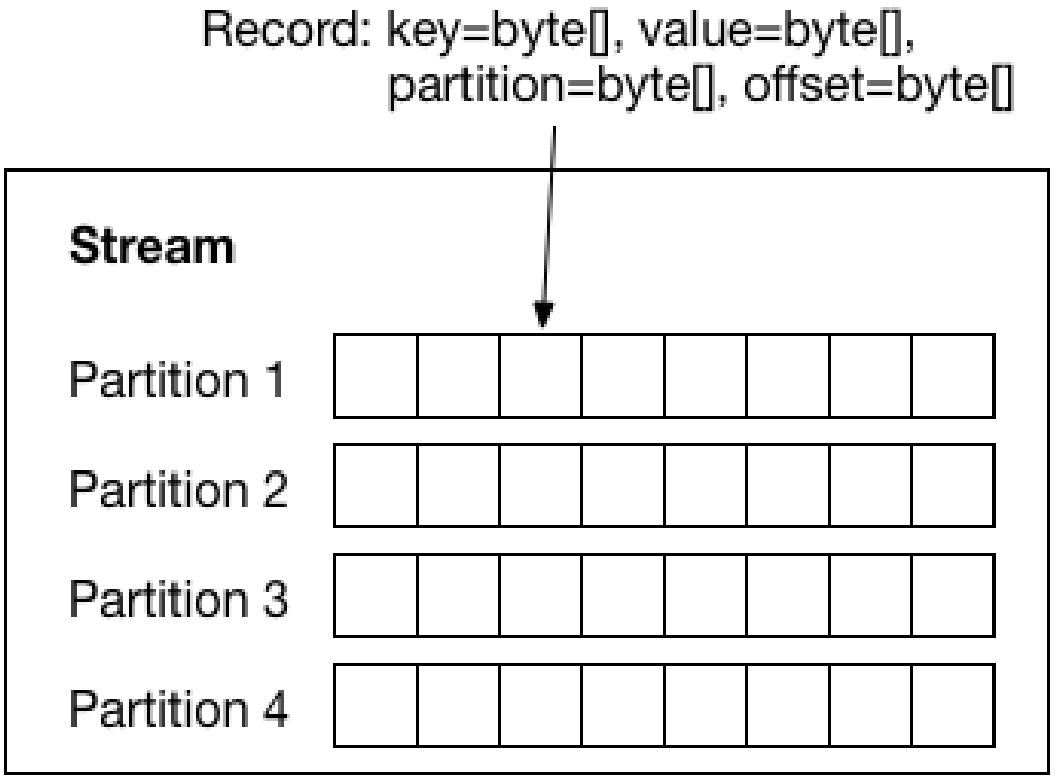


Figure 2.9: A partitioned stream: the data model that connectors must map all source and sink systems to. Each record contains keys and values (with schemas), a partition ID, and offsets within that partition [1].

Source: http://docs.confluent.io/2.0.0/connect/devguide.html

Currently a variety of connectors where specifically developed for the described platform. In the context of the architecture these represent the framework for extracting data from

almost any system a large organization might have. As each connectors deals with rather different types of systems, these connectors are also a reference implementation for most types of storage systems.

- JDBC connector
- File Stream connector
- Elasticsearch connector
- HDFS connector
- Binary File connector • SAP connector
- Couchbase connector
- FTP connector
- REST endpoint connector
- SOAP endpoint connectors

The general architecture for the data ingestion pipeline is outlined in Fig. 2.7.

2.5.1.3 Data Serialization and Storage

An important functionality required for building a useful Data Vault model is correct data partitioning and handling data schema changes. If the model stops functioning after column name changes or a full scan is always required to execute common queries, then the model has diminished value as an organization wide repository of data.

As an example let's take sensor data or documents arriving in a stream to the platform. A very common query one would need to do to analyze this data is to sort by date. If this data is stored as is in a single flat directory, this can be an extremely expensive query. For example car, sensor data may increase by terabytes each month. To address this common query pattern, we employ time based partitioning with a maximum granularity on a month, as described in the previous sections. The actual record assignment to time partition depends

on what delivery semantics we are using. Process-time (when the event was received) and event-time (when the event actually happened). The choice if either depends on data source.

Now let's also consider what happens if a column name or data type changes. This can lead to inconsistency and even break some process that are already running if this data is ingested. To solve this all data is stored in an efficient binary file format, that supports schema evolution, Apache Avro [31]. Avro relies on schemas. When Avro data is read, the schema used when writing it is always present. This permits each datum to be written with no per-value overheads, making serialization both fast and small. This also facilitates use with dynamic, scripting languages, since data, together with its schema, is fully self-describing. If the program reading the data expects a different schema this can be easily resolved, since both schemas are present. The large benefit of this is a compact storage format and a self-describing dataset.

This means that events are stored in an organized manner with the current schema always stored with the data, which makes the described example much easier to handle. While such a model has many strong points there are also a number of drawbacks, specifically the management of the schemas themselves. In this work Kafka is used as the stream representation. Using Avro would mean that each event dispatched into Kafka, would need to hold the schema as well as the data. This poses a problem as some schemas have hundreds of tables and are essentially large than the data itself. This poses the problem of unnecessary event size increases. Another problem is the operational overhead for managing the schema definitions. To this a central repository of the schemas, would be beneficial. A schema registry was implemented, which allows the connectors to register their schema against the registry and receives a unique id, which is then transferred with the event instead of the entire schema. The Schema-Registry handles the schema evolution and data integrity checks. The schema can be retrieved at any time and attached to the actual event for long-term storage.

2.5.2 Data extraction Auto Scaling

As outlined in the previous sections, all data source and sink connectors are running by reading and writing messages to Kafka, most of the job progress tracking, reporting and orchestration is done by using Kafka. This allows for the connectors to run in a distributed fashion with automatic or manual scaling done via Apache Mesos [25]. More connectors are launched based on demand and are fully immutable, which means if a connectors crashed, a

new instance will be transparently started somewhere else and it will resume work from the last offset. This is quite critical in order to always provide a consistent and up to date view of the source systems to the Data Scientist using the platform, which is a central part of almost all the CRISP stages.

As the all the data moves through Kafka as an event stream and the connectors auto-balance the extraction tasks, it is fairly straightforward to define an auto-scaling based on monitoring the Kafka consumer offsets, estimating an optimal number of consumers required to consume this and use Mesos to dynamically launch more or less consumers.

In Kafka consumer can be grouped into so called consumer groups. Each consumer thread then gets ownerships of a single partition of data inside Kafka. Like this the scaling of the consumption process is handled transparently by Kafka itself, as there is a unique mapping between a partition and consumer. So we can launch as many consumers as partition. In this sense only the sink type connectors can be transparently auto-scaled and the problem can be defined as thread allocation problem similarity to [32] as each consumer thread can be assigned a partition. Optimally each partition gets a single thread, but based on the type of processing that is being done (saving to HDFS, transformation or others) this might not be effective because the target system may be overloaded or each transformation requires to many processing resources, such as CPUs or RAM, which are not available inside the cluster.

To implement this type of auto-scaling a Mesos-Framework was developed to keep track of each consumer, its resource requirements and processing rate and automatically scale processing tasks. For this it is required to efficiently monitor a large number of consumers and accurately estimate their lag as well as resource consumption and predict how many consumers are optimal to keep the production and consumption as balanced as possible.

2.5.2.1 Consumer lag estimation strategy

The consumer group lag estimation is implemented similarly to [33] and [20]. Several rules are defined and each partition of a Kafka topic is checked against these rules to determine its status and the state of the consumer group. These are empirical rules are useful to detect if a consumer group is functioning optimally or it is viable for auto-scaling. This is more effective in contrast to setting a fixed threshold for each consumer group, which could lead to the scaling processing launching or killing more consumers all the time. Every partitions is evaluated in order to provide a good estimate of the entire consumer groups consumption

and is not limited to specific topics only. Some consumers may process multiple topics at a time and by only considering single topics the cases, where one of two topics being consumed is written to much more than the other one, which effect the consumption, would not be properly considered.

As there is potentially a large amount of message produced and consumed every second it is more effective to consider the lag check over a sliding window of time as opposed to discrete measurements. As a large number of messages are written and consumed a noticeable change in consumer lag might take some time to become noticeable and this approach seeks to optimize resource usage as well as consumer over allocation should be avoided. This sliding window value may depend on the specific consumer type and can be configured individually. As the consumers offset are the main value being tracked the windows "time" moves with each new consumer offset commit ed. For each consumer offset, the offset itself, the timestamp that the consumer committed it, and the lag are stored. The lag is calculated as difference between the HEAD offset of the broker and the consumer's offset. Because broker offsets are fetched on a fixed interval, it is possible for this to result in a negative number. If this happens, the stored lag value is zero
[33].

Based on this the following evaluations rules can be defined similarly to [33]:

- If any lag within the window is zero, the status is considered to be OK.
- If the consumer offset does not change over the window, and the lag is either fixed or increasing, the consumer is in an ERROR state, and the partition is marked as STALLED.
- If the consumer offsets are increasing over the window, but the lag either stays the same or increases between every pair of offsets, the consumer is in a WARNING state. This means that the consumer is slow, and is falling behind. At this point auto-scaling should kick in.
- If the difference between the time now and the time of the most recent offset is greater than the difference between the most recent offset and the oldest offset in the window, the consumer is in an ERROR state and the partition is marked as STOPPED. However, if the consumer offset and the current broker offset for the partition are equal, the partition is not considered to be in error.

The rules are only evaluated against groups and partitions for which we have a full windows of time have passed. As new consumers might be added or removed or crash on their own offset expiration has to be considered as well as Kafka will keep the full history of consumer offset at all times, but this might skew the lag estimate. The newest offset for each partition is checked to see if it was received longer than the specified number of seconds ago, and if so, the partition is removed for the consumer. If all partitions in a topic are removed, the topic is removed as well. If all topics for the consumer group are removed, the group is remove.

Based on this a well-defined way of estimating if a consumer group is behaving poorly can be defined. Using this the automatic-scaling problem ca be defined as consumer allocation problem and we need to estimate how many consumers are required to bring the consumers back into balance. This is based on [32], but using an evolutionary based approach to estimate the number of consumers(threads).

2.5.2.2 Evolution based Auto Scaling

Evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based meta-heuristic optimization algorithm. An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions (see also loss function). Evolution of the population then takes place after the repeated application of the above operators [34].

In this work the NeuroEvolution of Augmented Topologies (NEAT) is used to solve the posed problem. NEAT is a genetic algorithm for the generation of evolving artificial neural networks. It is used to evolve the network topology and also the corresponding connection weights [35].

The algorithm is initialized with a random population of networks grouped into species based on a K-Means clustering approach using the Manhattan distance metric. The initial population is a collection of random topologies, which ensures topological diversity from the start. NEAT starts with a minimal population, which then is gradually complexioned, this is done in order to get a minimal solution in the outcome [35].

Having initialized and already containing an initial population from the previous generation, the algorithm needs to evaluate the current genome population and assign a fitness score

which will be the indication of its innovation and will directly impact the evolution in the future generations. The Evaluation is done by allowing the individual networks to control the auto-scaler for a pre-specified duration. Evaluation will be initiated for every consumer group, that is marked as **warning** and **error** at the end of the rule evaluation window.

To measure how successful the current genome was at controlling the autoscaler, some kind of objective function must be used. In evolutionary methods that function is most generally referred to as the fitness function and it measures the fitness of the current genome, thus expressing how successful it was at carrying out the objective. A choice of a suitable fitness is a very important task and in most cases drastically influences the outcome of the evolutionary process. At the end of the evaluation window a list of all problematic consumer groups is available. For each consumer group the following quantities are available:

- ψ_i latest offset for offset for consumer group of topic i
- λ_i latest offset for topic i
- c cpu cores used in total by current consumer group

Based on this the fitness can be defined to minimize the different the "consumer lag" and resources used:

$$\frac{(\lambda_i - \psi_i)}{c} \tag{2.2}$$

After a genome population has been evaluated and fitness values have been assigned, the next step is to further complexify the networks in the population and do another evaluation round. The complexification is achieved by random mutations applied to the current population. These are basically applied transformations on the population that allow to generate new individuals that might perform better. Neat contains two mutation variations, structural and non-structural.

- Non-structural mutation are mutations applied to the weights of the connection between neurons. The changes of weight values are usually quite small and entirely new values are rarely assigned. In order to bias the weight assignment towards newer

values a mechanism, that more probably changes newer values rather then older ones in the genome, is introduced.

- Structural mutations on the other hand directly influence the network topology and can split an existing connection into a new gene node and two new connection, add a new connection between two gene nodes or remove that connection, which may result in the removal of one or both previously connected nodes.

By iterating on this process populations are created, that are able to achieve better fitness values, which in this case means they can control the autoscaler more effectively. This has shown to be perform similar to alternative approaches [36].

2.6 On-demand Sandbox environments

Based on the outline requirements and the overall conceptual model a "sandbox" development environment can be defined based on the same architectural primitives. There are significant architectural considerations when mapping the requirement set outlined in the previous chapter to a technical implementation. We will outline the technical design of these sandboxes and the necessary components we use to provide the technical functionality.

First of all we require access to all the existing cluster resources preferable via the native interfaces, but provide a remote accessible and simple to use environment. As one of the main gaols is to simplify development, this approach should also support some way to distribute source code and dependencies along with the environment. We require this environment to be stateless and immutable as possible, but still contain all the necessary tools and code when moved.

2.6.1 Immutable environments

To achieve the flexibility required we just replace it with another instance to make changes or ensure proper behavior. This would allow sandboxes to not only serve as an analytical environment, but as a template for all Data-Driven projects, that ever come to production.

To this there has been a large shift in the Industry to use Linux Containerization technology [37] to solve these types of problems. **Linux Containers**. LXC (Linux Containers) is an operating-system-level virtualization method for running multiple isolated Linux systems (containers) on a control host using a single Linux kernel [37]. The Linux kernel provides the cgroups functionality, that allows limitation and prioritization of resources (CPU, memory, block I/O, network, etc.) without the need for starting any virtual machines, and namespace isolation functionality that allows complete isolation of an applications' view of the operating environment, including process trees, networking, user IDs and mounted file systems [37]. Such environments can be created based on a defined specification, which allows the environment to inherently be self-describing [37].

This essentially would allow us to launch anything we want in completely independent environments. For example, on-demand edge-nodes. This approach is very useful for creating immutable architectures, where every component can be replaced at any point in time. It is being used more and more in the Industry and is becoming the standard for running distributed services and applications [38].

The use of this type of process management and assuming that each sandbox is fully isolated and immutable allows us to easily reason about FaultTolerance and Scalability.

2.6.2 Fault Tolerance and Scalability

In the context of Immutable containers, service fault tolerance and scaling becomes a process scheduling problem. Technically we have a set amount of resources in total. Each sandbox takes a certain amount of resources, such as CPU, RAM, Disk space and Network. Considering the that fact a large amount of resources are required, but limited amount of Hardware at our disposal, we need to effectively plan where each sandbox can run and how much resource it can actually use. This is a well-known problem in the Industry as is usually tackled by global Resource Manager or planners, such as Apache Mesos [25], Google Kubernetes [39] or Borg [24] and there are well defined patterns of tackling such problems [38]. In this work we use Apache Mesos due to its popularity in the Industry and its proven ability to scale.

Such a resource manager is essentially a higher level version of an Operating Systems Kernel running on many machines. Applications decide what resources they require based on their

processing time and requirement, and the resource manager tries to accommodate this. If the service crashes or the computing node fails, it can be transparently restarted somewhere else assuming the application can transparently handle something like this. This would allow us to efficiently run many sandboxes on very limited hardware by automatically scaling how much resource each one requires as well as making each sandbox Fault-Tolerant by launching another one during outages.

2.6.3 Collaboration

A central aspect of a solution like this is simplifying collaboration. As we propose isolated environments, this becomes more challenging. The accepted approach using central source code-based collaboration, such as Git and SVN [40], which we also adopt. But it is often challenging to share large files, such as models, test data and dependencies between team members using these systems. To this end we implement a shared network file-system layer between team members of the same project. Each project team gets a filesysytem, with individual workspaces for each members. Team members can then decide how to structure this environment in order for it to be more productive for them.

One crucial requirement for such system is Fault-Tolerance. Which is even more crucial due to our implantation of Immutable sandboxes. Any large files generated by the project team should be durable to sandbox and computing node outages. Loosing modelling results can lead to days of lost work time. To address this there are a number of high performance and durable implantation of Network Filesystems being used in the Industry [41] [42]. In this work we adopted GlusterFS as our network storage due to its proven performance, simplicity and integration capabilities with systems like Mesos [42].

2.6.4 Isolation

As we outlined above isolating the sandboxes simplifies management and development. By isolation we are specifically referring to isolation of resources, such as:

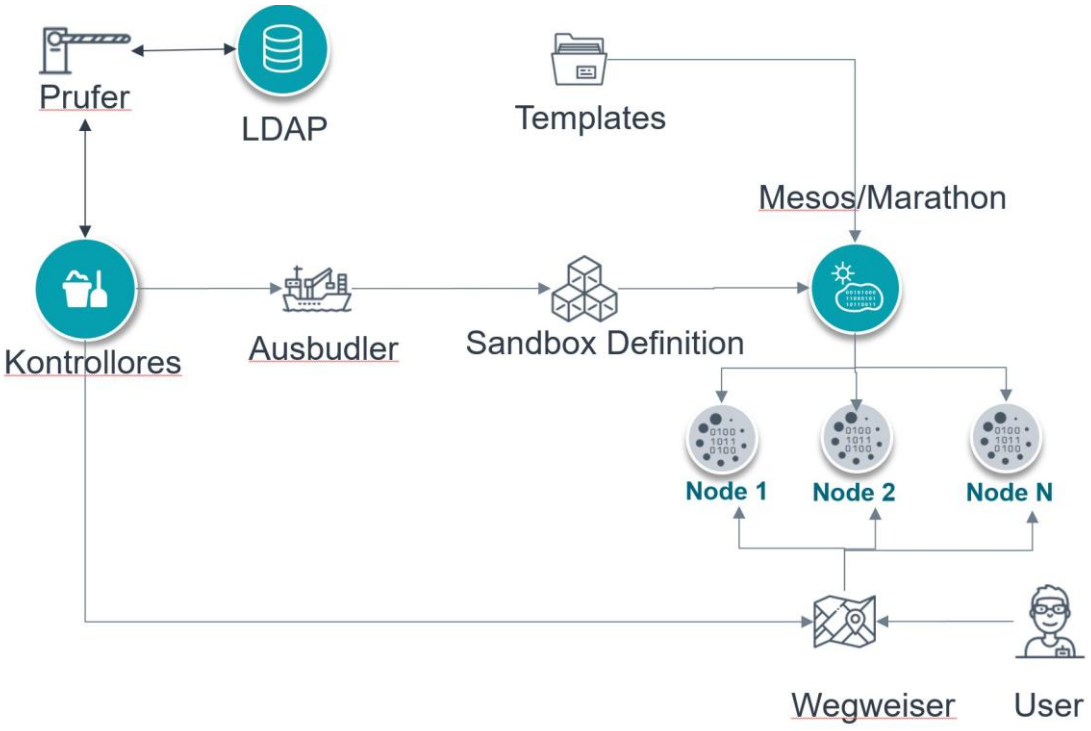


Figure 2.10: Sandbox environment orchestration

- CPU
- RAM
- Filesystem
- Libraries and System tools

Most of these are out-of-the box supported by Linux container technology and are supported and managed by Apache Mesos. On the other hand as these sandbox environments are not exclusive to Analytical Cases, which produce reports as their output, but Service based applications as well. An example of this would be a web-service that provides recommendations based on customer input data. Such applications usually run as a webservice and most commonly rely on some database system, which might not be provided as part of the Hadoop installation. Such services can be easily launched in a separate sandbox environment. We provide teams of developers with easy access to these services or databases, provided they are part of the same use-case. As there might be any number of such services running inside the cluster, this might lead to problems as defined in [43]. So in order to simplify the development process we would need full Network isolation as well.

To solve this it is now an accepted approach in the Industry to use software based overlay network solutions, which are computer networks that are built on top of another network [44] [45]. This gives us the ability to selectively isolated certain Sandboxes from each other, allowing the users of these sandboxes to launch any service they want in their sandboxes and transparently make it accessible to other team members, without leading to network clashes with other environments. In this solution we use Calico networking, which is well integrated with Mesos environments.

2.6.5 Security

Due to the level of isolation provided Security becomes an Authorization problem and can be implemented as per requirements from the Organization. A commonly accepted approach in the Industry is to integrate with a central Organization wide authority, which has all the information about user rights and permissions. In most cases we integrate directly with the Organizations central LDAP, such as Kerberos or Microsoft Active Directory [46]. As most Hadoop installation also support this integration we can transparently enable security throughout the solution without introducing a new authorization and authentication concept in the Organization.

2.7 Architecture

Having discusses the individual building blocks of the solution, we will outline how this translates to a technical architecture and implementation as a number of automation services are required to build the necessary platform. This architecture is implemented similarly to the Data-Hub layer and must fulfil the same requirements of low-maintenance, low-latency and flexible services. To this end, we adopted a micro-service architecture, which a lightweight and flexible approach to implementing Service Oriented Architectures [47] [48].

The Sandbox orchestration service is defined with the following components.

Sandbox Templates The first required component is a set of templates for sandbox environments. These vary based on the Organizational standards, but these typically contain all the necessary tools to interface with the Hadoop Cluster, run analytics and build applications. For example:

- Secure Shell access
- Hadoop client to connect to a HDFS cluster
- Integrated Development Environment
- Programming environments
- Database applications
- Access to defined views in the Data-Vault
- Distributed Computing environment that allows to submit jobs to the larger Cluster
- Collaboration tools

These are common tools and having a repository of such as propose built templates allows to formalize the tool choice and testing and deployment processes during development. New templates can be added based on demand by extending the already existing ones.

Marathon is a production-grade container orchestration platform for Apache Mesos [49]. This allows for API based orchestration of containers based on the specific templates. We use this as our central orchestration platform for all sandboxes.

Kontrollores is our central control service. It handles request for creating new teams and sandboxes for them. It provides the following functionality:

- Handle Sandbox creation requests
- Authorize request
- Create code repositories
- Create collaboration file system
- Initiate Sandbox creation
- Report status

A request to Kontrollores must contain necessary details for launching a sandbox. An exmaple request can look like:

```
{

        "sandbox": {
                "cpus": 16,
                "mem": 10240,
                "disk": 20480,
                "user": "userID",
                "group": "groupID",
                "services": ["spark", "hadoop","ipython"],
                "sources": ["telematics", "crm"]

}       }
```

Ausbudller Is an interface service to Marathon and handles the actual creation and monitoring of a single sandbox. It validates these request based on the user permissions and quotas stored in the central LDAP.

Ausbudller directly interfaces with Marathon and the Sandbox Template Repository in order to initiate the deployment of each sandbox. It infers what the required resources, data and software requests by Kontrollores translates to as a rigid definition. This includes actual usernames, group names, source paths, wegweiser route definitions and etc.

Based on the example request from Kontrollores the request to Marathon will look like:

```
{
        "sandbox": {
```

```
        "id": "human-readable-display-name-ead123mas1031asd1031123na1239",
        "uid": 1132012,
        "gid": 1239212,
        "group_id": 1239514,
        "HADOOP_USER_NAME": "userID",
        "NB_UID": "userID", "cpus": 1,
        "net": "networkID",
        "mem": 1024,
        "disk": 20480, "user": {
            "username": "userID",
            "name": "John Doe",
            "email": "j.doe@reply.de",
            "password": "12345678"
        },
        "group": "groupID",
        "display_name": "human-readable-display-name", "access": {
            "telematics":"hdfs://hadoopservice/data/raw/vault/
telamatics/*",
            "crm":"/data/crm"
        },
        "image": "templates/spark-hadoop-ipython",
        "network":"sandbox-global"
        "wegweiser": {
            "proto":"ws,http"
            "ingore":false,
            "load-balance":false,
            "authentication":"ldap:userID:groupID" }
    }


}
```

This contains all the necessary information for the sandbox to star.

Wegweiser is the central routing component. As the sandboxes are in isolated environments we require a secure gateway to dynamically route individual users through the internal network to their sandbox. This is achieved by discovering sandbox details based on their definition retrieved from Kontrollores and building a scalable reverse-proxy for HTTP, Websocket and SSH traffic and allow transparent access for the teams of Data-Scientist based on the central authentication mechanism. Wegweiser detects necessary configurations for routing to a sandbox based on its definition as defined by Ausbudller.

Networking is based [44] networking, which allows to create decoupled networks for one or multiple sandboxes, with firewall rules defining what resources can be accessed or not.

Prufer handles validation of resource and tools requests based on the users in the team. This is important as a "sandbox" depends on a variety of services, such as the Wegweiser router, the networking of the sandboxes, the internal services running inside the sandboxes and the installed tools. Prufer runs a list of predefined rules-based tests against the running sandboxes and restarts them if any failures are detected. The aggregated errors are then written as a stream to Kafka, which is required for reporting.

To this end the analytical sandbox environments are proposed, which are generated, isolated environments provided to Data Scientists, Analysts and Engineers so that they can build up their project on the data. It is a fully isolated environment, where the user can install or download any extra tools they require and is accessible via an analytical and console view. The defined micro-service architecture is outlined in Fig.2.10.

Chapter 3

Information Retrieval and Sharing

One of the biggest challenges faced by an Organization when exploring possible use cases for Data Scientists is in experience knowledge sharing and transfer. In this chapter we will outline in more detail what issues this entails, what kind of requirements this generates for a system aiming to solve them and an approach to implement a system that fulfils these requirements.

3.1 Information Management requirements

Large Organizations have a large number of Departments, which vary widely based on their size, project they take on and the way these projects are completed and documented. This leads to a large variety of data sources, column names and documentation being created on the same subject by a large number of stakeholders from different Departments some of whom might not be part of the Organization anymore. This leads to challenges for Data Scientists and the IT Department, which have to identify the relevant information and people or documents describing the data, especially when the project involves more than one data source. The most important consideration here is time spent on actually finding out if data is available and similar issues as opposed to more productive data analysis.

To this end we propose a list of requirements we accumulated based on our experience working with teams of Data Scientists at large organizations:

- **1.Index decentralized documentation repositories.** Each Department should be able to retain their knowledge repositories with little to no functional changes. Data should be acquired from these repositories with as little effort as possible from the Departments side.

- **2. Support a large number of formats.** Each Department should be able to use any binary file format for their project documentation.
- **3. Document Metadata has to be preserved.** Such properties as authors, auditors, names, comments should all be extracted and preserved and be uniquely identifiable.
- **4. Low Latency Indexing.** Changes to or new documents should be reflected as fast as possible.
- **5. Content Indexing.** As a minimum all textual content has to be searchable, there has to be a possibility to later extend this to non textual content, such as audio and image files.
- **6. Author extraction.** Each document must be tagged automatically with the authors of that document. This can be extracted depending on the metadata or by analysis of the document or related documents.
- **7. Document summarization.** A short summary of the Document would be very beneficial when exploring a large collection. Giving context knowledge about document can often be use full to understand the content at a glance. This could be implemented by extracting important sentences or keywords.
- **8. Context search.** Each document describes a specific context or project in relation to the organization or the Department. This representation of the document if very valuable specifically for finding similar projects, but which are not specifically referring to specific data sources, projects or Departments.
- **9. Cluster by Department context.** It is often useful to look at groups of documents referring to specific contexts as opposed to exploring them one by one. This is specifically interesting to see if documents of one Department refer to a context usually associated with another Department.
- **10. Datasource search.** It should be possible to search for specific data sources referred to in the documents. Meaning every document should point to specific data sources it mentions. Used with the document clustering approach, documents that have no mention of any document can also be tagged.
- **11. Datasource summarization.** It is very important to present an outline of the data, that is actually available in connection to a datasource a document refers to.

- **12. Flexible faceting.** All of this extracted information should be made available as facet views, available when using the tool to explore the organizational information.
- **13. Decoupling of functionality.** As we described above, the landscape of the organisational data and systems is always changing and there should be no hard dependencies between specific Departments or types of analysis. If a Department stops producing documentation in a specific format or at all or author information should no longer be stored, the systems should no require a substantial rewrite to remove these functions.
- **14. Flexibility for extensions.** Similar to the previous requirement the system should be easily extensible if further analysis is required, such as for example support for extracting information from image data. This should also not invalidate the previous requirement and require a substantial modification to the system. New attributes or indices should be handled transparently.
- **15. Performance.** This tool will be used by Data Scientist and the Business Department in their day to day work, this means it should allow for a responsive experience and support many hundreds of concurrent users.
- **16. Scalability.** The entire solution entails fairly complex computation as there are potentially tens of gigabytes of documentation(from many Departments) and data meta information(data schemas) as well as terabytes of data that has to be analysed. This means the solution should be scalable in order to provide the similar levels of performance independent of new documentation or data source.
- **17. Fault tolerance.** This is more of an exploratory tool and as such full end-to-end correctness is not necessarily required. By correct we mean, that no documents are processed more than once and all results are always consistent throughout the solution. Nevertheless we need to provide a certain level of fault tolerance thought the system so little to no supervision is required to ring the system to a consistent state. In this case we are looking more at the system being eventually consistent [50].

This is a list, that outlines the base requirements a system for information retrieval and sharing should fulfill. It is list of complex functional requirements which we need to map to a technical problem definition and implementation.

In the next section we will explore this definition and the architecture as well as the algorithmic implementation of this tool, which we called the Organizational Information Marketplace.

3.2 Information Marketplace

To outline the solution we first need to define, what input we have and what required output has to be produced.

Input. Our input is a collections of documents for each Department/datasource the organization has. This collections are unbounded and may increase and decrease as well as change over time and so can the data sources. Currently a single collection of documents can be larger than ten gigabytes with documents containing hundreds of words. Additionally such a system should be ready to process raw organizational data as well. This would be important to analyze the actual data content as opposed to definitions. This can become extremely large with terabytes of data coming every month.

Output. As the output we require the document with additional extracted or calculated metadata information, such as the summary of the document, the semantic/contextual representation of the document, similar documents and others. This information should be exposed through an interface, that supports a rich set of text based queries. As the entire content of the document has to be indexed with all the additional extracted information, we are looking at indexes tens of gigabytes in size per Departments.

Based on the defined requirements the Information marketplace is proposed as a solution for Information Management and Retrieval. The Information Marketplace is an intelligent search engine system, specifically designed to tackle the problem of information retrieval and sharing. It is a centralized hub for finding relevant information and people or documents describing the data, especially when the project involves many varying data sources.

Information Marketplace is created to minimize time needed for a Data Analyst to find out if data required for a specific scenario is present in company repositories, find a documentation for that data and find people and departments responsible for it and having knowledge about it. IMP has an web portal for querying documentations from all departments and for all data sources. It provides tools for finding relevant documents,

finding documents similar to a known one and finding links to people and departments within the organization with knowledge of data and data source.

IMP is designed to automatically detect document modification and new documents addition and then to automatically extracts existing metadata and enhance it based on similar documents. This data is then added to an existing searchable index of documents. It has a responsive UI for performing multifaceted queries on documents and is designed to be used by hundreds of users on a daily basis. It is also designed to works without requiring any modifications to the existing documentation management system of each department.

IMP allows each Department to retain their knowledge repositories without functional changes. Documents will be acquired from Departments' repositories automatically, with as little effort as possible from the Departments side. IMP supports large number of input document formats. It does not impose any restrictions on each Department, so they should be able to use any binary file format that fits best as their project documentation. Document metadata will be preserved in IMP. Such properties as authors, auditors, names, comments will all be extracted and preserved and be uniquely identifiable.

IMP has a low latency indexing. So changes to old documents or addition of new documents are reflected as fast as possible.

All textual content of documents is searchable. Later this may be extended to non-textual content, such as audio and image files.

IMP automatically tags each document with the authors of that document. This is done based on the metadata or by analysis of related documents if author key is not present in metadata. It is planned to automatically generate document summaries as set of most important phrases/sentences of the document.

IMP allows for search of similar documents. Each document is describing a specific context or project in relation to the organization or the Department. This representation of the document is very valuable specifically for finding similar projects, but which are not specifically referring to specific data sources, projects or Departments.

IMP supports clustering by Department context. Looking at groups of documents referring to a specific context is often much more useful for a Data Scientist or Data Analyst as opposed to exploring them one by one. This is also specifically interesting to see if documents of one Department refer to a context usually associated with another Department.

It is possible to search for specific data sources referred to in the documents. Meaning every document should point to specific data sources it mentions. Used with the document clustering approach, documents that have no mention of any datasource can also be tagged.

IMP has a datasource summarization. Which means that it presents an outline of the data, that is actually available in connection to a datasource a document refers to. This means each data-source is also profiled and useful statistics, such as null values, distinct values, max values, min values and etc. is available.

IMP supports flexible faceting. All of the aforementioned extracted information is made available as facet views and is available when using the UI to explore the organizational information.

Data collection engine is flexibility for extensions. The system is easily extensible if further analysis is required, such as support for extracting information from image data. This does not require a substantial modification to the system. New attributes or indices should be handled transparently. Also if a Department stops producing documentation in a specific format or at all or author information should no longer be stored, the systems should not require a substantial rewrite to remove these functions.

IMP is designed to be used by Data Scientist and the Business Department in their day to day work. It allows for a responsive experience and support many hundreds of concurrent users.

IMP is designed to be scalable in order to provide the similar levels of performance independent of amount of new documentation or data source.

3.2.1 Usage patterns

Figure 3.1: IMP stages overview

There are a few usage patterns of IMP that satisfy different needs of a client. We will talk about the following ones just to outline possible usage scenarios.

Find documents that are relevant to search keywords

The main use case of IMP will be to find documents that contain specified keywords, which may represent any data type, idea or any other feature or name that Data Scientist may

want to find. Relevant documents will be shown based on the frequencies of keywords and importance of keywords to the topics describing the document. There are further facet filters that IMP web UI provide for localizing desired documents based on document author, date, department and etc., but those will be discussed separately after descriptions of usage scenarios. Results will be shown based on their relevance score, but there is also an option of sorting by date.

Find columns of data sources relevant to search keywords

It may be very useful to find columns of datasource by their name. Found columns can be later filtered out for better results, or documentation of corresponding columns can be viewed to better understand data and datasource at hand and its relevance to use case.

Contextual search: find topics relevant to search query

Apart from indexing textual data IMP data retrieval stage also generates contextual topics for each document. They are very useful for finding categories of documents within a Department, and documents where similar features are discussed. Those topics are defined as sets of words describing the category, and those topics can also be queried for. When displaying the results of a query with specific keywords our search engine will also present the topics which are relevant to the query. This may speed finding relevant datasources a lot when searching using sets of keywords usually associated with desired feature.

Find documents that describe a specific datasource or column

IMP's web UI can be used even when names of required datasources or columns is already known. It can be used for finding all the documentation on given datasource or column.

Find all documents by a specific author

When the name of person responsible for a datasource or column is known, IMP can be used for finding all documents authored by him or referring to him. This can be very handy if we know that the person should have knowledge of data that is relevant within the business case.

Find documents of a specific department

Similar to the case above, one may want to find all documents from a specific department. This is also supported by IMP.

Filtering results

There are the following facet filters that can be used for drilling down results of any query:

- Author

- Date
- Department
- Datasource
- Topic
- Type

Results scoring

By default, query results in IMP are presented based on the relevance score of each document. This ordering can be changed to be by date. That may be useful for tracking changes of data.

In essence the IMP is a search engine, which we have to fill with documentation with additional metadata field, which would allow for the set of queries that were described. It is need to map the requirements described in the previous chapters to a set of stages of transformation a single document has to go through and query-able views, which are the end result of these transformations. First lets define the views, which are needed to satisfy the requirements:

- Document Full-Text Search View
- Search View based on Author, Time, File-Format, Department, Datasource
- Search View based on extracted keywords, contexts, summaries
- Related document View, based on contexts and keywords

Based on this we can define the information that has to be extracted from the documents, whic are shortyl outlined in Fig. 3.1. The detailed outline of implementation of each stage will be presented in the next sections. This included such topics as scalability and fault tolerance considerations for each stage.

3.3 Text Analysis for Context Understanding

To full-fill the requirement outlined in the previous section we need to solve the following problems:

- Create an efficient contextual representation of documents in the context of its originating department
- Create an efficient representation for a data-source definition
- Define an efficient way to compare documents with documents and documents with data-sources, based on these representations
- Extract a condensed summary of a document

In this section the issues of performance of the presented algorithms will not be tackled, this will be addressed in the next section. First lets outline how we extract the document contextual representation.

3.3.1 Data source representation and matching

As described in the previous section, what we have as input is a lot of documents with no extra information apart from their content and origin. We need to map each document received to one or more data sources. If we had some documentation for each data-source of the organization we could rephrase this issue as a context matching problem, this could be solved by one of our other stages of analysis. But it is very rarely the case, that an Organization has any significant amount of information on every data source they have, quite often the information is there, but has been lost in the thousand of documents scattered across Departmental repositories. This is the aspect of the problem we are trying to solve in this stage.

As it stands assuming we have the technical definition of each data source, schemas, column definitions and etc., we need to find instances of documents that contain references on these attributes. A straightforward approach would be to scan through each document and count how many instances of all permutations of all columns are contained inside the document.

But if we consider the complexity of such calculations for a typical organization, it becomes a very suboptimal approach. For example this stage has to function with low latency processing at an organization with dozens of different database systems, which have thousands of tables in total and each table containing potentially hundreds columns. Such an organization has tens of thousands of pieces of documentation, with a hundreds of

documents being modified or created every day. Such documents usually contain hundreds of words on average. This is an obviously suboptimal calculation, which is not practical to compute with low latency. For example lets say we have 3000 tables with 100 columns each and 10000 documents with 500 words each. This would mean for one document we would need to do $500 \times 300000 = 150000000$ not simple comparisons. Another downside of such an approach is, that we would need to keep all data related information in memory as a collection, which would lead to large resource requirements.

To this we adopt a method similar to the approach described in [51], which rephrases this issue as a comparison of sets of data. Indeed we can efficiently represent a document and each data source specification as sets, and then our problem would be to find a set most similar to each incoming document.

Of course this would mean, that we are taking the overlap of a document and a full set of data columns, which would never truly fully overlap, but as described in [51] this would give us a a good approximate match and we could then analyse, which of the matched datasource is truly mentioned in the document.

For this we build an efficient representation of each data source specification using the MinHash (min-wise independent permutations locality sensitive hashing scheme) algorithm [52].

3.3.1.1 Minhash

The Jaccard similarity coefficient is a commonly used indicator of the similarity between two sets. For sets A and B the Jaccard similarityis defined as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cup B|} \tag{3.1}$$

We can phrase this as the ratio of the size of the intersection of A and B to the size of their union. J lies in $0 <= J <= 1$. More specifically, if $J = 1$ the sets are identical, if $J = 0$ there are no shared members otherwise the value can be interpreted as “the probability that a random element from the union of two sets is also in their intersection” or “the probability that a randomly chosen element chosen from one of the sets is also in the other set.”. This is widely accepted measure of similarity, but it is still quite expensive to compute [51] especially for our case. To this end [52] describes an algorithm to estimate the Jaccard similarity (resemblance) between sets of arbitrary sizes in linear time using a small and fixed

memory space. It is also quite suitable to use in a stream processing context due to the low computation time and compact representation. First lets define a matrix representation of the comparison of multiple sets. Lets define matrix M where m_{ij} is 1 if element i is in set S_j and 0 otherwise. Lets denote N as the number of all unique elements in each set. Then for N lets take K random hash functions $\{h_1,h_2,\ldots h_k\}$, which map each element to a random unique ID from $[N]$. Then for for k counters $c_1,c_2,\ldots,c_k$ we can define the Minhash for the set S as: Then set $m_j(S) = c_j$ and we can define the Jaccard distance estimate as:

$$JS_k(S_i, S_{\hat{i}}) = \frac{1}{k}\sum_{i=1}^{k=1} I(m_j(S_i) = m_j(S_{\hat{i}})) \tag{3.2}$$

where $I(\sigma) = 1$ if $\sigma = 1$.

This also gives us a compact representation, which we can calculate and store for each data source specification, meaning we only need to calculate the Minhash for each incoming document. But this is still computationally expensive as we still need do the computation at least $O(N_{tables})$, meaning the query cost increases linearly with respect to the number of datasources.

A popular alternative is to use Locality Sensitive Hashing (LSH) index.

3.3.1.2 Minhash-LSH

One approach to implement LSH is to "hash" each element many times, so that similar items are more likely to be hashed to the same bucket. We can then just look at the pairs, which ended up in the same bucket. The main idea as described in[51] is that most of the dissimilar pairs will never hash to the same bucket, and therefore will never be checked, and the amount of false positives is low.

Algorithm 1 Min hash on Set S

1: **for** $i = 1$ to N **do** 2: **if** $S(i) = 1$ **then** 3: **for** $j = 1$ to k **do** 4: **if** $h_j(i) < c_j$ **then**
5: $c_j \leftarrow h_j(i)$
6: **end if** 7:**end for** 8: **end if** 9: **end for**

For a minhash representation computed as described above, an effective way to choose the hash functions is to take the matrix of the representation and partition it in to b partitions

with r rows each. Then we use a hash function for each partitions and calculate the mapping to many buckets. Each partitions can use the same has function and keep a separate buckets for each partition. This means columns with the same vector in different bands will not hash to the same bucket. Then we can check each band for matches in each bucket.

Based on the describes algorithms we create a Minhash representation for all datasource specifications and build and LSH index over these. This allows us to very efficiently query for the most similar data sources for an incoming document in a stream. We take the 3 top matches and find, which one actually is most referenced inside the document. This is attached to the document as extra metadata and is written to another stream. Due to the implementation this can be easily scaled by launching more instances as the MinHash representation is fairly compact.

3.3.2 Context extraction

We require a representation of each document in the context of the Department it is generated by, as well as other Departments. To do this we need a model, that captures the concepts most used and referred to by projects in the Department. This would give us a valuable contextual representation of key concepts. This could be achieved by extracting keywords or popular words similar to the methods described in the Document Summarizing stage, but what this kind of approach lacks is the document level representation. We want to consider the context both on the level of separate concepts as well as groups of these concepts(documents). This would give us a more complete view of the context of the Department as well as allow us to extract, such a contextual representation for any document. Meaning we would see how a document relates to work usually carried out in the Departments. We also want to analyse the extracted context to find similar concepts. Another requirement we have is that any model used should be flexible and require little maintenance to be updated in the future.

For similar problems Probabilistic Topic Models are a popular choice [53]. Using these models we can infer the semantic structure of a collection of document using Bayesian analysis. These models are fairly popular and have been used to solve a wide range of similar problems [54] [55]. There are a large number of available approached to modelling documents [53] [56] in a similar fashion, but in this work we decided to use Latent Dirichlet Allocation(LDA), due to its proven performance and the specifically the existence of an on-line algorithms for training. As outlined in the previous section it is imperative for the model

to be flexible to updates as well as support low-latency inference. We will outline LDA and its specific strength and the reason it was chosen in more detail below.

3.3.2.1 Latent Dirichlet Allocation

In LDA documents in a corpus are assumed to be generated from a set of K topics. Each topic k is represented by a Dirichlet distribution over the vocabulary:

$$\beta_k \sim Dir \tag{3.3}$$

linux	software	experience	business	windows
servers	based	grow	requirements	systems
install	experience	company	functional	server
plus	development	work	analyst	exchange
architect	paid	salary	technical	experience
infrastructure	leading	please	create	administrator
monitoring	small	full	systems	microsoft
tools	expertise	benefits	design	active
various	daily	start	user	directory
services	developers	time	support	administration

Figure 3.2: LDA example

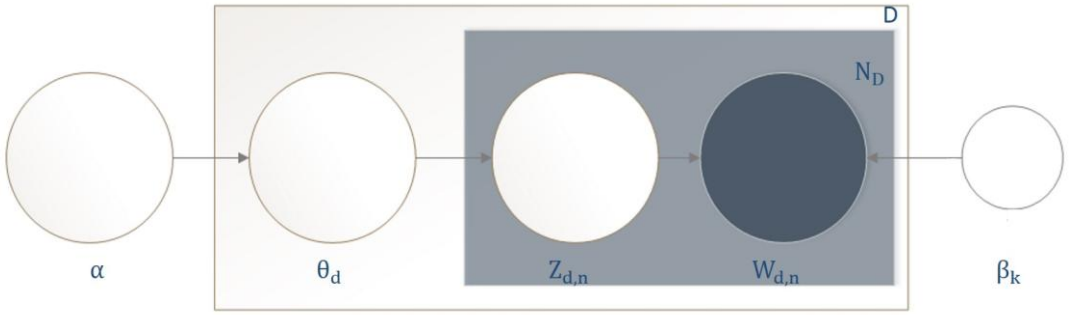


Figure 3.3: The graphical model. The shaded circles indicate observed variables, while the unshaded one represents the latent variables.

Each document d is also repented as a Dirichlet θ_d. Each word w in document d is then assumed to be generated by drawing a topic index z_{dw} from a multinomial distribution $Mult(\theta(d))$, then choose a word w_{dn} for that topic from $Mult(\varphi_{zdw})$. By assuming this generative process we can go backwards and estimate these matrices from a collection of documents. In this case a collection per Department, which will generate a model per Department. We can summarize this generative process in a graphical model depicted in Fig. 3.3.

This depicted graphical model described the generative process, which creates the topics of LDA. In more detail, it can be summarised as the following process:

1. For each topic k from K
   (a) Create a distribution β_k for each topic from the dictionary
2. For each document d from (D)
   (a) Create distribution θ_d denoting the probabilistic proportions for each topic
   (b) For each word w in d
      i. Create topic assignment $Z_{d,n}$ from θ_d, where $Z_{d,n}$ lies in $[1,k]$ ii. Create word assignment $W_{d,n}$ for $Z_{d,n}$ from $\beta_{Zd,n}$

, where β_k is a V dimensional vector of word probabilities, which we currently assume is known. Each θ_d is a K dimensional Dirichlet random variable and can take values in the $(k-1)$ simplex, so $\sum_{i=1}^{k} \theta_{k,i}$ and has the following probability density:

$$P(\theta, \alpha) = \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma\left(\alpha_i\right)} \theta_1^{\alpha_1 - 1} \ldots \theta_k^{\alpha_k - 1} \tag{3.4}$$

, where the parameter α is a k dimensional vector of positive elements and Γ is the Gamma function. Each hyper parameter α_j can be seen as a prior observation count for the numner ot times topic j is assigned to a document, before having observed any actual words from that document. It i s con v en ient to u se a symmetric Dirichlet distribution with a single hyperparameter α ,such that $\alpha_1 = \alpha_2 = \ldots = \alpha_K$. By placing a Dirichlet prior on the topic distribution θ we will get a smoothed topic dsitribution, with the amount of smoothing determined by α. We can see an example of three topics in a two dimensional simplex in Fig. 3.4 .

For α and β the joint distribution of a topic mixture θ for N topics $\mathbf{z}$ and a set of N words $\mathbf{w}$ is the following:

$$p(\theta, \mathbf{Z}, \mathbf{W} | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^{N} p(Z_n | \theta) p(W_n | Z_n, \beta) \tag{3.5}$$

,where $p(z_n|\theta)$ is θ_i for i, where$Z_n^i = 1$. We can get the marginal distribution of a document by:

$$p(\mathbf{W}|\alpha,\beta) = \int p(\theta|\alpha) \prod_{n=1}^{N} \sum_{Z_n} p(Z_n|\theta)p(W_n|Z_n,\beta)d\theta \tag{3.6}$$

And to get the probability of the entire corpus:

$$p(D|\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \prod_{n=1}^{N_d} \sum_{Z_{d,n}} p(Z_{d,n}|\theta_d)p(W_{d,n}|Z_{d,n},\beta)d\theta_d \tag{3.7}$$

The parameters α and β are are sampled once in the process of generating a corpus. The variables $theta_d$ are sampled once for every document, and $Z_{d,n}$ and $W_{d,n}$ are sampled once for each word in each document.

A highlight of the LDA model is, that this method does not simply account for contextual value for a particular word, but rather operates with word frequencies and tries to extract the contextual interconnections by placing many words in a single context(topic). In turn, this feature is also a strong point, as the model can be applied to any text data in almost any language without any changes. This also sets LDA apart from a simple document clustering approach. A classical clustering model usually restricts a document to being associated with a single topic. LDA, on the other hand, involves three levels, and notably the topic node is sampled repeatedly within the document. Under this model, documents can be associated with multiple topics [53]. This is advantageous as it is not realistic to assume, that there a single context in an entire department of an organization.

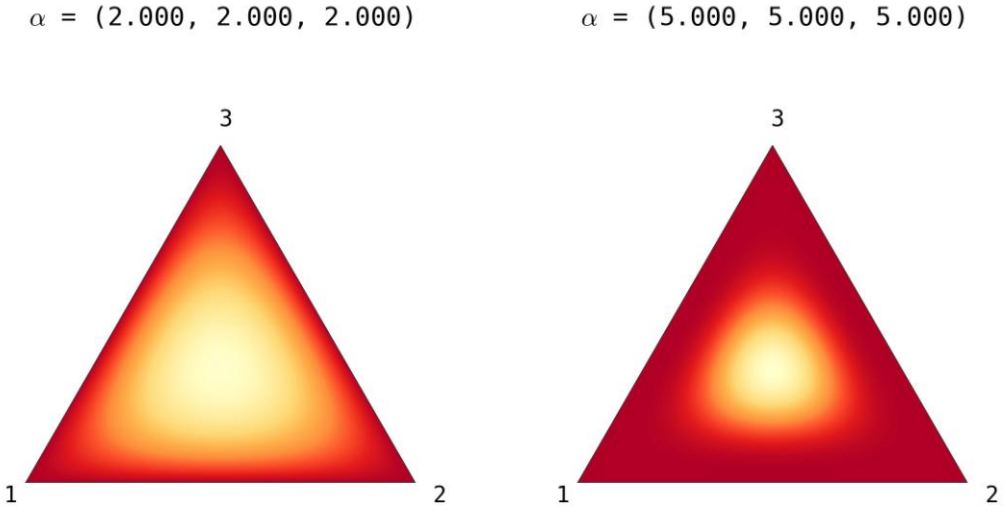


Figure 3.4: I llustrating the symmetric Dirichlet distribution for three topics on a two-dimensional simplex. Darker colours indicate lower probability.

The main goal is to analyze a corpus of documents with LDA. This can be achieved by examining the posterior distribution of the topics β, topic proportions θ, and topic assignments z . This will reveal latent structure in the collection that can be used for prediction or data exploration.

For a set of document the distribution of latent varaibles can be denoted as:

$$p(\theta, z|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, z, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)} \tag{3.8}$$

To normalize the distribution we can integrate Eq. 3.6 on θ and sum by z:

$$p(\mathbf{w}|\alpha, \beta) = \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma\left(\alpha_i\right)} \int \prod_{i=1}^{K} \theta_i^{\alpha_i - 1} \left(\prod_{n=1}^{N} \sum_{i=1}^{K} \prod_{j=1}^{V} \left(\theta_i \beta_{ij}\right)^{w_n^j} \right) d\theta \tag{3.9}$$

It is computationally intractable the calculate this integral [53], so multiple methods can be used to approximate it. Examples can be Markov Chain Monte Carlo(MCMC) methods or variational inference [53] [57]. Both classes of methods are effective, but both present significant computational challenges in the face of massive data sets. In this work as we require low latency inference as well as flexibility to do automatic model updates, we adopt the Online Bariational Bayes algorithm [57]. This method also has the quality of being static in memory as the entire stream of documents does not have to be loaded fully when required. The online variational approach is the preferred solution due to the fact it allows for realtime, incremental training on unbounded streams of data. More detail is outlined in the next section.

3.3.3 Online variational inference for LDA

As proposed in [57] the online variational inference for LDA is an alternative learning algorithm, which is used in this work due to its specific strong points.

This method also has the quality of being static in memory as the entire stream of documents does not have to be loaded fully when required. MiniBatches of documents are

processed to infer these matrices. As with most variational algorithms, the processing consists of two steps:

- The E-Step: Given the minibatch of documents, updates to the corresponding rows of the topic/word matrix are computed.
- The M-Step: The updates from the E-Step are blended with the current topic/word matrix resulting in the updated topic/word matrix.

Each document received is added to the model and after that the topics are extracted, this allows us to continuously update the model as well as extract topics from documents. It is more effective to update the model with minibatches [57]. A mini-batch is a small batch of documents. In our case this is usually set to 100. This is also efficient for throughput as well as simplified offset tracking. Offsets are committed only after a mini-batch of documents has been successfully processed. After each iteration the model is save to disk and only then the offsets are committed. This means we ensure the update model always contains all the documents and is up to date. The topics are attached to the document with their distribution, this is necessary later on for ranking text matches.

3.3.4 Text Summaries

As we only have unstructured documents and most organization do not tag or write abstracts for each document, we need to rely on an unsupervised method for text summarisation [58]. We summarize the given text, by extracting one or more important sentences from the text. This summarizer is based on the "TextRank" algorithm [59], which was chosen when considering its performance and a simpler implementation for key-sentence extraction vs keywords.

3.3.4.1 TextRank

In the TextRank algorithm [59], a text is represented by a graph. Each vertex corresponds to a word type or sentence. Vertices v_i and v_j are connected with a weight w_{ij}, which corresponds to the number of times the corresponding word types co-occur within a defined window in the text. The main objective of TextRank is to compute the most important vertices of the text. If we denote the neighbours of vertex v_i as $Adj(v_i)$, then the score $S(v_i)$ is computed iteratively using the following formulae:

$$S(v_i) = (1-d) + d \times \sum_{v_j \in Adj(v_i)} \frac{w_{ji}}{\sum_{v_k \in Adj(v_j)} S(v_j)} \tag{3.10}$$

Vertexes with many neighbours that have high scores will be ranked higher. We are of course not interested in keyword extraction, but sentences based summarization.

To this end as shown in [59] it is we can adapt the same algorithm to sentences. Each vertex will represent a sentence and, as "co-occurrence" is not a logical relation between sentences, defining another similarity measures. A similarity of two sentences is defined as a function of how much they overlap. The overlap of two sentences can be determined simply as the number of common tokens $w_1^i ... w_N^i$ between the lexical representations of the two sentences S_i and S_j. In [59] it is defined as:

$$Sim(S_i, S_j) = \frac{|\{w_k | w_k \in S_i, w_k \in S_j|\}}{log(|S_i|) + log(|S_j|)} \tag{3.11}$$

By applying the same ranking algorithm 3.10 and sorting the sentences in reversed order of their score we can find the most important sentences, that represent the document. This algorithm was implemented and applied in a stream of document to produce the output of the stage. The algorithm performance only depends on the length of the documents, and based on our average length of documents, the processing speed satisfies out latency requirements.

3.4 Architecture and Implementation

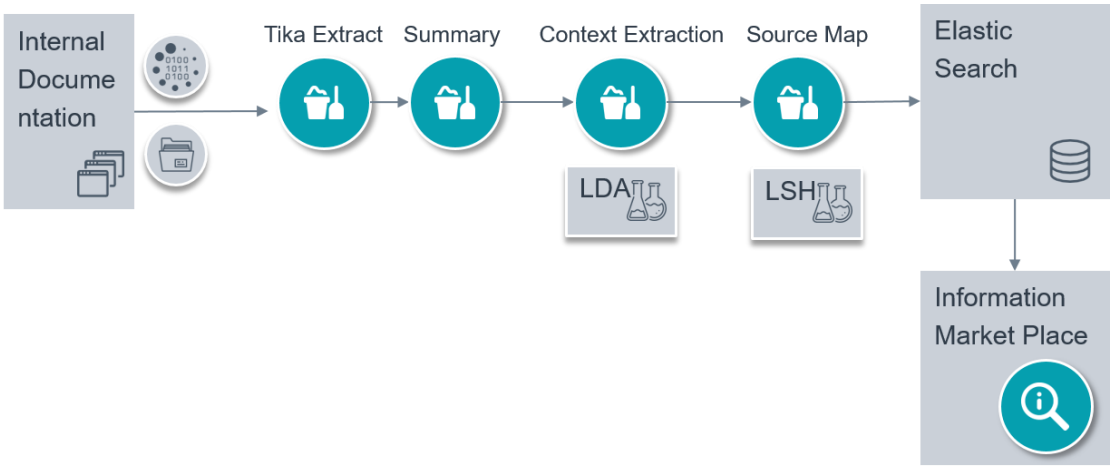


Figure 3.5: Information Marketplace Architecture

There are significant architectural and algorithmic considerations when mapping the requirement set outlined in the previous chapter to a technical implementation. To achieve

all the requirements for performance this has to be distributed application running on multiple machines, which introduce a large set of benefits and problems.

First lets consider the architectural side of the problem, before outlining how the required information is extracted and modelled. To outline the architecture we first need to define, what input we have and what required output has to be produced.

Input. Our input is a collections of documents for each Department/datasource the organization has. This collections are unbounded and may increase and decrease as well as change over time and so can the data sources. Currently a single collection of documents can be larger than ten gigabytes with documents containing hundreds of words. Additionally such a system should be ready to process raw organizational data as well. This would be important to analyze the actual data content as opposed to definitions. This can become extremely large with terabytes of data coming every month.

Output. As the output we require the document with additional extracted or calculated metadata information, such as the summary of the document, the semantic/contextual representation of the document, similar documents and others. This information should be exposed through an interface, that supports a rich set of text based queries. As the entire content of the document has to be indexed with all the additional extracted information, we are looking at indexes tens of gigabytes in size per Departments.

Based on this and the defined requirement the stream processing approach defined in Chapter 3 can be used again. It is advantageous as it fits the latency requirements as well the requirement for flexibility. The high level architecture is presented in Fig.3.5.

3.4.1 Streaming Layer implementation

Based on the described architecture we need to map the requirements described in the previous chapters to a set of transformation a single document has to go through to be ready to ingest into the view layer, which allows querying and presentation. What is required is a full representation of the document, data sources and Departments. Based on this we can define the stages of transformation:

1. Document load from source
2. Document original format to plain text conversion with metadata extraction

3. Map document to specific organizational data source
4. Extract document summary
5. Extract semantic representation of the document(contexts)
6. Find closely related/similar documents
7. Find Departments a data source might refer to
8. Index document

We will now outline in more detail what each transformation does and how it is implemented.

3.4.1.1 Load document

- Input - Directories to watch for files
- Output - Stream of document creation, update, delete

Documents may reside on a filesystem in every Department. We need to detect all the files already there and detect the creation of new files to fulfill the low-latency requirement. We implemented a distributed directory watcher, which watches for filesystem event and emits these events into a Kafka stream. Each watcher keeps a reverse index of the filesystem it is watching. This is required in order to track what files we have already processed, as no duplicates should be processes if possible. The filepath and the last modified date are used as the offset key in the reverse index. The offsets are only commited if the file was detected, and has not been modified in the last few seconds. This is done in order to avoid reading files, which are still being written to. With this approach we ensure fault-tolerance and exactly once processing of all files. Because this index is stored as a collection of offsets in a Kafka topic no duplicates or missing documents will be processed during service or full node failures.

We implemented these watcher using the Kafka Connect Framework [1], which is a framework tightly coupled with Kafka and offers some utility methods to more simply write data to Kafka. Each watcher writes an event to Kafka containing the source of the event, the nature of the event and the filepath. This stage is then defined as a collection of such watchers accessing multiple organizational sources distributed throughout the organization.

3.4.1.2 Document conversion

- Input - Stream of documents
- Output - Stream of document metadata and text representations

We need to extract information from a large variety of document formats. As most of the document ion is written we need to convert it to a simple text representation as opposed to a native binary format such as Microsoft Word or PDF. For parsing a large variety of format we use Apache Tika [60].

For extracting information from a large variety of document formats we use Apache Tika. Tika also allows us to extract a full set of file metadata from the file, such as authors, auditors, modification dates, owners, comments and etc. The following document formats are supported by Apache Tika:

- HTML files: Tika supports virtually any kind of HTML found on the web.
- XML and derived formats like XHTML, OOXML and ODF.
- Microsoft Office document formats, starting from Microsoft Office 97.
- OpenDocument format (ODF), which is most notably used as the default format of the OpenOffice.org office suite.
- Portable Document Format (PDF)
- Electronic Publication Format (EPUB) used for many digital books.
- Rich Text Format (RTF)
- Plain text files. Particularly, Tika uses encoding detection code from the ICU project to automatically detect the character encoding of a text document.
- Multiple mail formats like mbox, used by many email archives and Unixstyle mailboxes, and Microsoft Outlook PST and MSG formats.
- Compression and packaging formats like Tar, Zip, 7Zip, Gzip, RAR and etc.

Tika also allows us to extract a full set of file metadata from the file, such as authors, modified dates, owners and etc. This stage produces three different text representations.

1. HTML bases representation, preserving the formatting, usefull for display
2. Plain Text representation
3. Filtered text, based on language stop words are filtered, characters are normalized

As document arrive in a highly asynchronous manner from many data sources in a single stream we can implement a distributed stream processing application running on top of Mesos to convert this representation to Text. Essentially each streaming consumer recieves the file, extracts the text and metadata and materializes the result to another stream. Processes can be pre-allocated based on the number of documents committed and average processing time, this is implemented keeping the main ideas of [32].

```
{

        "path": /data/department/project/doc.pdf,
        "project":"project",
        "department": "department",
        "name":"doc.pdf",
        "metadata": {

                "content_type":"pdf" ,
                "created":"1476637503 " ,
                "last_modified": "1476637501 " ,
                "author": "John Smith",
                "title": "Document about things" , },
         "content":"Document about things...",
         "tokens":["document","about"]


}
```

The output of this stage would be:

1

2

3

3.4.1.3 Datasource mapping

- Input - Stream of documents
- Output - Stream of documents with Organization data sources labelled

As we outlines in previous sections understanding which data a document is referring can be very beneficial to Data Scientists, when trying to build a project. This info can both be used to find out what the data is about or the reverse, which data matches a certain topic.

The output of this stage would be:

```
{

        "path": /data/department/project/doc.pdf,
        "project":"project",
        "department": "department",
        "name":"doc.pdf",
        "metadata": {

                    "content_type":"pdf" ,
                    "created":"1476637503 " ,
                    "last_modified": "1476637501 " ,
                    "author": "John Smith",
                    "title": "Document about things" , },
         "content":"Document about things...",
         "tokens":["document","about"],
         "sources": [
      {
        "name": "iwh.AVIS.A01_LIGHT",
        "prob":    0.203125

      },
      {
        "name":    "iwh.AVIS.MVD",
        "prob":    0.1875

      }]


}
```

3.4.1.4 Document summary

- Input - Stream of documents
- Output - Stream of documents with added summary

This stages receives a stream of documents as input and outputs the same document with an additional summarization field attached to the metadata. A text summary in this context is a set of most important phrases/sentences of the document. Such kind of method are very sensitive to stopwords, words that do not impact the semantic meaning of the text, such as "and", "or" and etc. So this stage relies on the third form of the text representation, which is already filtered and cleaned.

The output of this stage would be:

```
{
    "path": /data/department/project/doc.pdf,
    "project":"project",
    "department": "department",
    "name":"doc.pdf",
    "metadata": {
                "content_type":"pdf" ,
                "created":"1476637503 " ,
                "last_modified": "1476637501 " ,
                "author": "John Smith",
                "title": "Document about things" , },
     "content":"Document about things...",
     "tokens":["document","about"],
     "sources": [
  {
    "name": "iwh.AVIS.A01_LIGHT",
    "prob": 0.203125
  },
```

```
        {
            "name": "iwh.AVIS.MVD",
            "prob": 0.1875
        }]
        "summary":"important sentence"


}
```

3.4.1.5 Context extraction

- Input - Stream of documents with extracted information
- Output - Stream of documents with context
- Output - Context Model for each Department

Each document received is added to the model and after that the topics are extracted, this allows us to continuously update the model as well as extract topics from documents. It is more effective to update the model with minibatches [57]. A mini-batch is a small batch of

documents. In our case this is usually set to 100. This is also efficient for throughput as well as simplified offset tracking. Offsets are committed only after a mini-batch of documents has been successfully processed. After each iteration the model is save to disk and only then the offsets are committed. This means we ensure the update model always contains all the documents and is up to date. The topics are attached to the document with their distribution, this is necessary later on for ranking text matches.

A model is produced per Department in order to captured the unique context of each. This means the stage contains at least one instance of each model, but the output of the stage is still a single stream.

```
{
```

The output of this stage would be:

```
1
2
3
```

```
        "path": /data/department/project/doc.pdf,
        "project":"project",
        "department": "department",
        "name":"doc.pdf",
        "metadata": {

                        "content_type":"pdf" ,
                            "created":"1476637503 " ,
                              "last_modified": "1476637501 " ,
                        "author": "John Smith",
                        "title": "Document about things" , },
          "content":"Document about things...",
          "tokens":["document","about"],
          "sources": [
    {
        "name": "iwh.AVIS.A01_LIGHT",
        "prob": 0.203125
    },
    {
        "name": "iwh.AVIS.MVD",
        "prob": 0.1875
    }]
    "summary":"important sentence",

"context": [
    {
        "word": "anzahl",
        "topic_p": 0.6300002727311531,
        "prob": 0.03350803662692222,
        "topic": 60
    },
    {
        "word": "installiert",
        "topic_p": 0.6300002727311531,
        "prob": 0.0325865118143565,
```

```
            "topic": 60
        }]


}
```

3.4.1.6 Related documents

As we are extracting the context of each document via the LDA model and attaching this information with the specific weights to the document that is indexed in the Search Engine, it is fairly trivial to find contextually related documents, just by including this field and its weights when executing text searches.

3.4.1.7 Departmental context

- Input - Stream of Documents
- Output - Stream of documents with attached "Department" label

If we take the described LDA Topic Model for each Department as a contextual representation of the concepts used, we can use this as the representation of what the Department does in the context of the organization. We essentially have a collection of sets of topics per Department and we want to find which other Departments this document relates to. We can exploit the Minhash based set matching approach we used in the previous stages and apply it to the problem of matching a document to an organization. We can define an LSH index over each set of topics for each organization. Then for any incoming document we need to find the most similar set of of topics from each organization a document relates to. Based on this we can extract a very rough estimation of "related Departments".

As we are using the same efficient representation as with the datasource mapping stage we can efficiently calculate this with low resource requirements


```
{

        "path": /data/department/project/doc.pdf,
        "project":"project",
        "department": "department",
        "name":"doc.pdf",
        "metadata": {

                    "content_type":"pdf" ,
                    "created":"1476637503 " ,
                        "last_modified": "1476637501 " ,
                    "author": "John Smith",
                    "title": "Document about things" , },
         "content":"Document about things...",
         "tokens":["document","about"],
         "sources": [
      {
        "name": "iwh.AVIS.A01_LIGHT",
          "prob":  0.203125

      },
      {
          "name":  "iwh.AVIS.MVD",
           "prob":  0.1875
```


The output of this stage would be:

```
    }]
    "summary":"important sentence",

"context": [
    {
        "word": "anzahl",
        "topic_p": 0.6300002727311531,
        "prob": 0.03350803662692222,
        "topic": 60
    },
    {
```

```
            "word": "installiert",
            "topic_p": 0.6300002727311531,
            "prob": 0.0325865118143565,
            "topic": 60
        }]
    "realted_departements":[
        {
            "departement": "departement",
            "p":1
        }
        ]

}
```


3.4.1.8 Document indexing

- Input - Stream of documents with extracted information
- Output - Elasticsearch indexes

This is the stage that actually indexes the data into a search engine, in this case Elasticsearch is used to create a reverse token index of the documents. We implemented these watcher

using the Kafka Connect Framework [1]. Each index writes an event from Kafka into Elasticsearch.

This stage is then defined as a collection of such indexers accessing multiple sources throughout the organization. The data is keyed with the filenames, event timestamp, and a unique offset is generated via a has function to ensure, that no duplicate data is written into Elasticsearch even during node faults. We can rely on Elasticsearch's idempotent write semantics to ensure exactly once delivery. By setting ids in Elasticsearch documents, the connector can ensure exactly once delivery by only overwriting the data and not creating new records. You can restart and kill the processes and they will pick up where they left off, copying only new data.

As opposed to the data loading stage we can read from and write to Elasticsearch in parallel and so we can launch more services to scale this stage based on amount of input.

3.4.2 View Layer implementation

Based on the described architecture we need to map the requirements described in the previous sections and the outlined transformation a flexible view layer is required, which is the end result of outlined transformations. These have to support the outlined usage patterns.

Based on the outlined requirements we need a flexible and scalable storage systems, that supports a large scale of analytical queries. Considering the fact, that most of our data are documents. It is more effecient to store the output in a Search Engine. In this work we use Elasticsearch [27] due to its proven scalability and performance, specifically in the context of stream processing.

Chapter 4

Operational Data Platform

We described a proposed end-to-end environment for creating and running Data-Driven projects at a large scale Enterprise. The described platform can efficiently manage the resources large number of users and services. On the data modelling side we proposed a flexible way to organize and transform stored Data Sets in order to become ready to answer analytical questions and generate value. We proposed a flexible way for discovering data and interconnections of the data, based on meta-data, functional descriptions and Documentation, in an automated and intelligent way, while not requiring a full departmental restructure. We also proposed a dynamic and scalable sandbox environment to allow collaborative and shared creation of Data Science use cases based on the data and simplify the deployment of these use cases into the production.

Based on the defined models, approaches and architecture a full implementation of the platform was created and integrated in two environments (production and mirror) for MAN GmbH. The platform containing the Data Ingestion and Modelling, Data Sandbox and Information Marketplace environments is defined as the Operational Data Platform (ODP). The overview architecture diagram is presented in 4.1. We will outline this in more detail below

4.1 ODP Architecture

The architecture is subdivided into six functional layers, which are then subdivided into three technical layers based on the technical environments where those parts of the layers are running. These are specifically the BareMetal Cloudera Hadoop environment and Apache Mesos managed scalable, streaming processing environment and MAN systems.

4.1.1 Data Sources

Contains an overview of connected Data Source from MAN, this includes:

- Raw Data - SAP Databases, DB2 Databases, Filesystems, Log Data.
- Processed Data stored in 5 distinct Data-Warehouses maintained by separate departments.
- Telematics Data stored in AWS.
- Historical Telematics Data Stored in EMC platform.

Currently the amount of extracted Data is around 30 TBs, but this is expected to grow dramatically once all trucks are connected to the Telematics System.

4.1.2 Ingestion

Contains all the implemented components used to create the ingestion layer, based on the concepts outlined in Chapter 2. It contains the scalable, distributed streaming Message Broker (Kafka) as well as all the Kafka-Connect connectors achieving the low-latency data extraction and processing. These are the backbone of the entire data ingestion pipeline and are running on Apache Mesos and are scaled automatically using the described auto-scaler based on analysing the Kafka queue and used resources via Mesos.

4.1.3 Storage

This layer contains the Data HUB: the raw data lake as well as the data vault modeled relationships. This is implemented on top the Hadoop File System (HDFS).

4.1.4 Analyse/Process

This layer contains all the tools and frameworks available for cluster based ad-hoc and repeatable analysis, such as Apache Spark, HUE, Hive and others.

4.1.4.1 Monitoring

As these tools are running on bare-metal hardware and outside the single resource based Mesos environment, monitoring is done separately.

4.1.4.2 Cluster Services

This layer contains general purpose tools, which are used by the Hadoop cluster for state and resource management

4.1.5 Application

This layer contains the bulk of the implemented systems. it includes all the streaming processing, analysis applications, micro-services and databases used to build up the platform. All of this is controlled by Apache Mesos, Docker and Marathon applications. The Information Marketplace is running as a distributed application in this layer, which includes all the staged stream processing services as well as the Elastic Search search engine database. This layer also includes the developed machine learning models used by the IMP.

This is also the layer, where the project developed in the Sandbox environment will run after development and testing.

	Without Specific tooling	ODP stable release
Access to Datasets	2 months	2 weeks
Data Discovery	3 weeks	1 week
Provisioning of Development Enviroment	1 month	1 week
Data Sicntists	3	20
Active Projects	2	10

Table 4.1: ODP impact.

4.1.6 Sandbox

The Sandbox orchestration micro-services as well as the actual sandbox development environments are running in this layer. This layers is backed the GlusterFS storage layer, which provides fault-tolerant data storage for data outside the HDFS.

4.2 Main Results

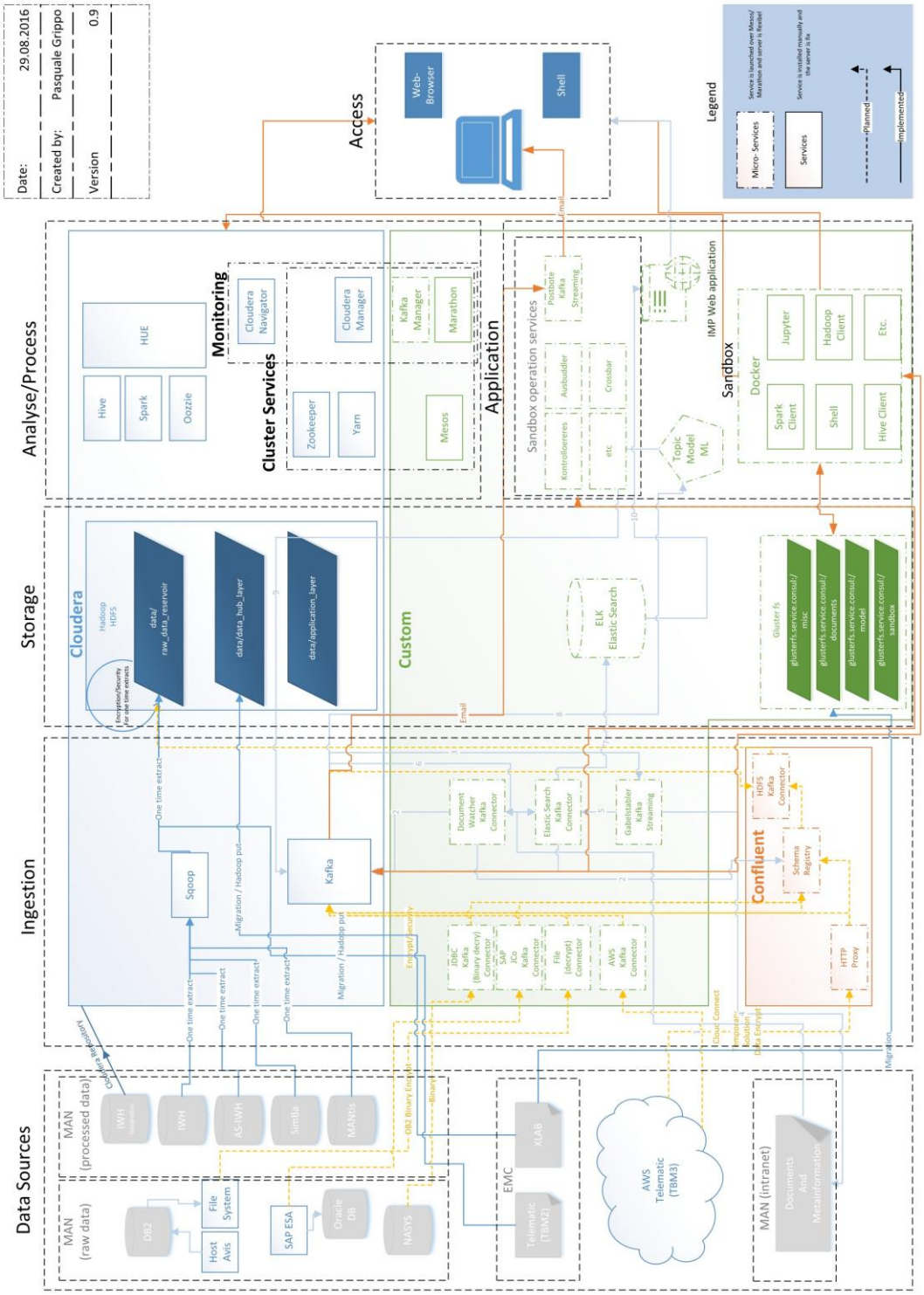
The outlined solution has been developed and deployed at large organizations, such as MAN in on-premise and cloud data-centers. It is actively used and evaluated by 20 different teams working on Data-Driven projects at MAN and a noticeable increase in productivity has been noted by many participants. This increase is highlighted in Table 4.1.

We believe the approach, proposed platform and its architecture provide a well structured environment to simplify Data-Driven projects at large organizations.

Main results of the work:

- Developed a conceptual model for creating Data-Driven projects on the basis of an Organizations data, which fulfill the project requirements, provides access to the raw data and to sources of data ingestion, storage and processing capabilities, while taking into account the existing organizational systems and restrictions.
- Proposed an Information Retrieval environment for exploring the organizations meta-data and knowledge basis contained in documentation.

  Text Analysis approaches for summarizing, context labelling and clustering documents are used to discover information about the organizational data and projects as well as uncovering relationships between them.
- Developed “Sandbox” environments, which provide multi-disciplinary teams access to the organizations data and computational resources, which are used to research new data models and evaluate them. These environments provide security to the teams and ensure the integrity of the source organizational systems by means of full resource and access isolation.
- The platform is implemented based on the concepts of stream processing, resource management and micro-services, which allows to effectively utilize limited hardware resources, customize and configure the platform to fit the existing organizational systems and processes. Based on this the platform can transparently scale to the volume of the organizations data as well as a large number of users.



turediagram.(blue)Bare-MetalCloudera Hadoop environment(green)Apache Mesos managedscalable, streaming processing environment.(gray) MAN systems

Bibliography

[1] C. Inc. (2016, Aug) Kafka-connect. [Online]. Available: http://docs.confluent.io

[2] N. Rahman and F. Aldhaban, "Assessing the effectiveness of big data initiatives," in *2015 Portland International Conference on Management of Engineering and Technology (PICMET)*. IEEE, 2015, pp. 478–484.

[3] T. H. Davenport, "Putting the enterprise into the enterprise system," *Harvard business review*, vol. 76, no. 4, 1998.

[4] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang, "Goods: Organizing google's datasets," *SIGMOD*, 2016.

[5] P. S. Patil, S. Rao, and S. B. Patil, "Optimization of data warehousing system: Simplification in reporting and analysis," in *IJCA Proceedings on International Conference and workshop on Emerging Trends in Technology (ICWET)*, vol. 9, 2011, pp. 33–37.

[6] T. H. Davenport and J. Dyché, "Big data in big companies," *International Institute for Analytics*, 2013.

[7] C. Shearer, "The crisp-dm model: the new blueprint for data mining," *Journal of data warehousing*, vol. 5, no. 4, pp. 13–22, 2000.

[8] R. K. Mobley, *An introduction to predictive maintenance*. Butterworth-Heinemann, 2002.

[9] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[10] S. Schaffert, "Ikewiki: A semantic wiki for collaborative knowledge management," in *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06)*. IEEE, 2006, pp. 388–396.

[11] A. Rowstron, D. Narayanan, A. Donnelly, G. O'Shea, and A. Douglas, "Nobody ever got fired for using hadoop on a cluster," in *Proceedings of the 1st International Workshop on Hot Topics in Cloud Data Processing*. ACM, 2012, p. 2.

[12] C.-F. Lin, M.-C. Leu, C.-W. Chang, and S.-M. Yuan, "The study and methods for cloud based cdn," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*. IEEE, 2011, pp. 469–475.

[13] P. Li, L. Toderick, and J. Noles, "Provisioning virtualized datacenters through virtual computing lab," in *2010 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2010, pp. T3C–1.

[14] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 5.

[15] D. Linstedt, "Super charge your data warehouse," *Dan Linstedt-Verlag*, 2010.

[16] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

[17] T. Dunning and E. Friedman, *Streaming Architecture: New Designs Using Apache Kafka and Mapr Streams*. " O'Reilly Media, Inc.", 2016.

[18] B. M. Michelson, "Event-driven architecture overview," *Patricia Seybold Group*, vol. 2, 2006.

[19] M. Hausenblas and N. Bijnens. (2015, Aug) Lambda architecture. [Online]. Available: http://lambda-architecture.net/

[20] T. P. Neha Narkhede, Gwen Shapira, *Kafka: The Definitive Guide Real-time data and stream processing at scale*. O'Reilly Media, 2016.

[21] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized streams: Fault-tolerant streaming computation at scale," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 2013, pp. 423–438.

[22] T. Akidau, A. Balikov, K. Bekiroglu, S. Chernyak, J. Haberman, R. Lax,˘ S. McVeety, D. Mills, P. Nordstrom, and S. Whittle, "Millwheel:

fault-tolerant stream processing at internet scale," *Proceedings of the VLDB Endowment*, vol. 6, no. 11, pp. 1033–1044, 2013.

[23] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fernández-Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt *et al.*, "The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1792–1803, 2015.

[24] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015, p. 18.

[25] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center." in *NSDI*, vol. 11, 2011, pp. 22–22.

[26] Netflix. (2016, Aug) Distributed resource scheduling with apache mesos. [Online]. Available: http://techblog.netflix.com/2016/07/ distributed-resource-scheduling-with.html

[27] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide*. " O'Reilly Media, Inc.", 2015.

[28] A. S. Foundation. (2016, Aug) Apache nifi. [Online]. Available: https://nifi.apache.org/

[29] ——. (2016, Aug) Apache flume. [Online]. Available: https://flume.apache.org/

[30] C. Inc. (2016, Aug) Kafka camus. [Online]. Available: http://docs.confluent.io/1.0/camus/docs/intro.html

[31] A. S. Foundation. (2016, Aug) Apache avro. [Online]. Available: https://avro.apache.org/

[32] A. Newell, G. Kliot, I. Menache, A. Gopalan, S. Akiyama, and M. Silberstein, "Optimizing distributed actor systems for dynamic interactive services," in *Proceedings of the Eleventh European Conference on Computer Systems*. ACM, 2016, p. 38.

[33] L. Inc. (2016, Aug) Burrow. [Online]. Available: https://github.com/linkedin/Burrow

[34] Wikipedia. (2016, Aug) Evolutionary algorithm. [Online]. Available: https://en.wikipedia.org/wiki/Evolutionary_algorithm

[35] K. O. Stanley and R. Miikkulainen, "Efficient evolution of neural network topologies," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 2. IEEE, 2002, pp. 1757–1762.

[36] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 500–507.

[37] R. Rosen, "Resource management: Linux kernel namespaces and cgroups," *Haifux, May*, vol. 186, 2013.

[38] B. Burns and D. Oppenheimer, "Design patterns for container-based distributed systems," in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[39] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.

[40] D. Spinellis, "Version control systems," *IEEE Software*, vol. 22, no. 5, pp. 108–109, 2005.

[41] J. Heichler, "An introduction to beegfs®," 2014.

[42] J. Klaver and R. Van Der Jagt, "Distributed file system on the surfnet network report," *University of Amsterdam System and Network Engineering*, 2010.

[43] W. Lin, "Isolating network traffic in multi-tenant virtualization environments," Feb. 8 2011, uS Patent 7,885,276.

[44] T. Inc. Project calico. [Online]. Available: https://www.projectcalico.org

[45] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013.

[46] V. Koutsonikola and A. Vakali, “Ldap: framework, practices, and trends,” *IEEE Internet Computing*, vol. 8, no. 5, pp. 66–72, 2004.

[47] S. Newman, *Building Microservices*. " O’Reilly Media, Inc.", 2015.

[48] ——, *Building Microservices*. " O’Reilly Media, Inc.", 2015.

[49] Marathon. [Online]. Available: https://mesosphere.github.io/marathon

[50] J. E. Boritz, “Is practitioners’ views on core concepts of information integrity,” *International Journal of Accounting Information Systems*, vol. 6, no. 4, pp. 260–279, 2005.

[51] J. Cohen, “A power primer.” *Psychological bulletin*, vol. 112, no. 1, p. 155, 1992.

[52] A. Z. Broder, “Identifying and filtering near-duplicate documents,” in *Annual Symposium on Combinatorial Pattern Matching*. Springer, 2000, pp. 1–10.

[53] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[54] R. Krestel, P. Fankhauser, and W. Nejdl, “Latent dirichlet allocation for tag recommendation,” in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 61–68.

[55] G. Maskeri, S. Sarkar, and K. Heafield, “Mining business topics in source code using latent dirichlet allocation,” in *Proceedings of the 1st India software engineering conference*. ACM, 2008, pp. 113–120.

[56] Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents.” in *ICML*, vol. 14, 2014, pp. 1188–1196.

[57] M. Hoffman, F. R. Bach, and D. M. Blei, “Online learning for latent dirichlet allocation,” in *advances in neural information processing systems*, 2010, pp. 856–864.

[58] K. S. Hasan and V. Ng, “Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art,” in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*.
Association for Computational Linguistics, 2010, pp. 365–373.

[59] R. Mihalcea and P. Tarau, “Textrank: Bringing order into texts.” Association for Computational Linguistics, 2004.

[60] A. Tika. [Online]. Available: https://tika.apache.org