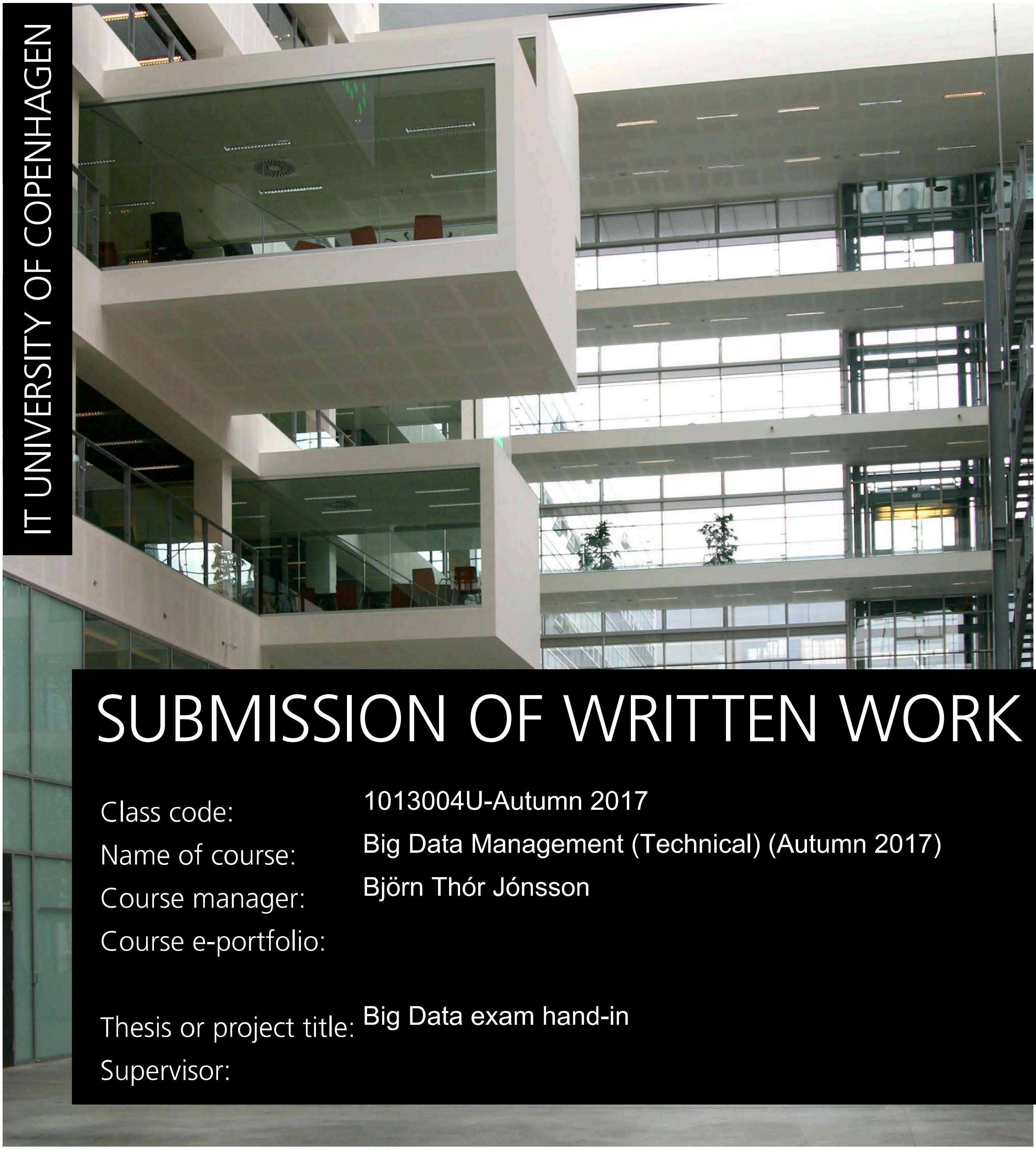
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code: 1013004U-Autumn 2017

Name of course: Big Data Management (Technical) (Autumn 2017)

Course manager: Björn Thór Jónsson

Course e-portfolio:

Thesis or project title: Big Data exam hand-in

Supervisor:

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Dennis Thinh Tan Nguyen	01/04-1993	dttn@itu.dk
2.			@itu.dk
3.			@itu.dk
4.			@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk

Big Data Technical Exam Hand-in

Dennis Thinh Tan Nguyen
dttn@itu.dk

December 19, 2017

Contents

1 Question 1 - Creepiness and Ethics

Based on the current dataset provided in project 2, the dataset does not provide or expose any explicit information about an employee and his or her workday. As it is, one may only map the network activity to a given device which does unfortunately not reveal whom this person is but only reveals where the given person was located at given timestamp and which devices is used.

However, if we assume that the dataset was to be provided to the IT-department at ITU, then one can also assume that the IT-department may possess additional data that can be unified. Since every employee and students at ITU are provided an account to access the internet and that the IT department may have additional logs of any internet sessions and activity, one may be able to map the account details to the devices that are connected.

Assume that any network traffic is monitored as well, then one may filter out traffic based on a given employee and conduct packet sniffing to extract information to reveal additional details about the given employee.

For example, if a given employee usually browses the internet during lunch, one may monitor this employee's network activity during lunch over a given timeframe. Based on the browsing activities one may use that to determine how this employee is as a person and profile him. For instance, if this employee like to browse vintage cars on the internet then one can assume that this employee may have a hobby related to that.

This might look harmless but then assume if this employee was unaware of his network being monitored and he decided to browse topics that may be regarded as private. For example, religious orientation, medical situation, or even sexual orientation this may introduce another level of profiling where any data that an employee produces may be mapped back regardless of the data's relatedness to how skilled the employee is to his profession.

Consequently, this may introduce ethical controversies regarding if it is acceptable for employers to conduct cyberstalking on their employees for profiling purposes. An employer may be biased with such data and down judge a skilled employee because some unrelated data does not look good in the perceptions of the employer whereas a less skilled worker may be regarded as better. While monitoring employees network activity might be legal in some aspects, one may still consider whether one should conduct such measure since it may introduce ethical complications and also may violate the privacy of people.

Nevertheless, an employee may use various techniques to avoid being. First of all the employee could a virtual private network (VPN) to create a secure encrypted connection between the employee and the VPN server (ExpressVPN (2017)). This would prevent the IT-department from packet sniffing the traffic due to its encrypted state and thus no information may be extracted.

However, even if the network traffic is encrypted, the employee's location can still be monitored based on when his device connects to a router. To avoid such cases, the employee may either only connect to a router when he needs to conduct work-related activities or bypass the router altogether by using a mobile hotspot that is disconnected from ITU's network.

Using the latter technique may prevent some employees from accessing resources that can only be accessed when one is connected to ITU's local network, and thus preventing them from doing some of their work. In such cases, it might be impossible or very challenging to be completely anonymous as an employee.

As discussed and experimented in the data detox in project one, it is easy to produce data while not producing data may require an immense effort to accomplish. The employee may reduce the data that he generates by utilizing a VPN and switch between a mobile hotspot network and ITU's network when accessing local resources.

2 Question 2 - Systems and Data Models

Before one can understand the purpose of Alluxio, it is critical to understand the size of the current big data ecosystem. As it stands, the big data ecosystem is enormous and complex with multiple technologies that can be employed compared to a few years ago where the size and complexity were at a much lower scale (Brust (2016)).

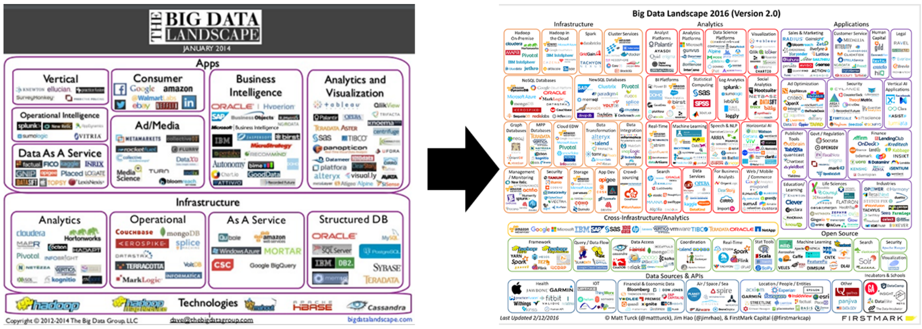


Figure 1: Technologies between 2014 and 2016

The size and complexity may very likely continue to grow over the next many years. Thus, there is a need to unify some of these big data technologies and frameworks to reduce the complexity when utilizing multiple technologies. Because of the differences between the types of applications and storage systems, it may be difficult allow existing or new application technologies to access data across multiple types of storage systems.

On the one hand, one must either integrate each application to each storage system, which does not scale well due to the increasing size of application and storage systems. On the other hand, the data may temporary be extracted to a temporary location that the applications know how to access which may result in data duplications and increased running time.

This is where Alluxio becomes very useful as it tries to address a subset of the big data landscape. Notably, the ever-evolving landscape of computation and storage technologies Alluxio (2017). Alluxio provides an interface called unified namespace that allows application technologies such as Spark, Flink, and MapReduce to communicate between multiple storage systems such as HDFS, S3, and GCS (Figure 2).

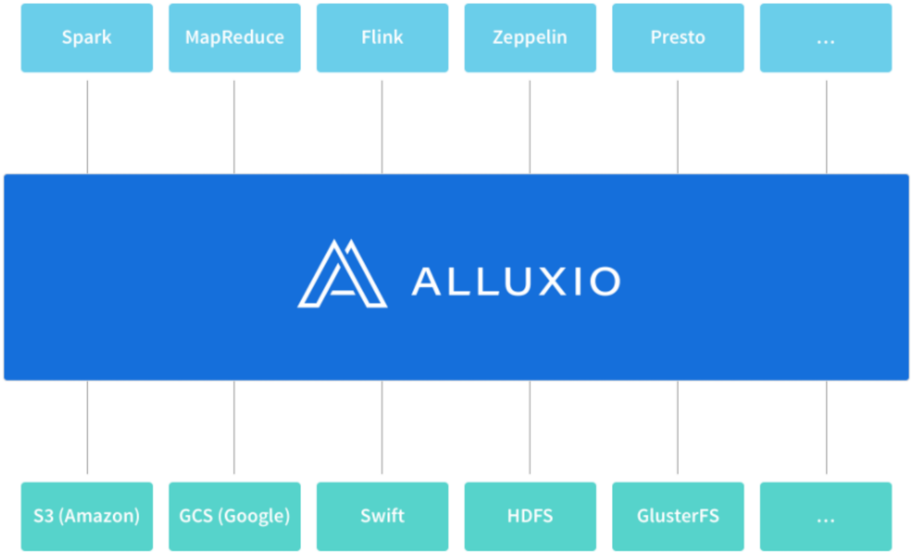


Figure 2: Alluxio as interface between storage and application systems

Alluxio takes all input from the applications and handles the communication with the underlying storage systems and outputs the results back to applications.

Thus all applications only need to know one single storage system, which is Alluxio, therefore decoupling the applications from all existing or new concrete storage systems. This level of modularity may finally promote scalability and maintainability while reducing complexity when integrating multiple systems.

Besides providing a unified namespace or interface, Alluxio also consolidates memory if one uses multiple computation systems. For instance, if one uses multiple spark jobs in the cluster and that they use the same data, Alluxio would then share a single cache of the data between the jobs instead of each job having a duplicated cache in their memory. Thus much memory may be reduced since the spark jobs may only need to read from Alluxio without caching anything.

Another benefit of storing the cache in Alluxio is during the event of a spark job crashing. The spark job can be restarted, and all the loaded data in Alluxio can be reloaded back to the spark job and thus reduce downtime of a job.

Seen from the perspective of Lambda Architecture, Alluxio would be applied between the Batch Layer and Storage layer. The batch layer and its underlying computation technologies would regard Alluxio as their storage layer while Alluxio would have a reference to all the technologies in the storage layer that contains the master dataset.

3 Question 3 - Data Cleaning and Storage

3.1 A - Data cleaning from project 2 applied to project 3

The cleaning approach used in project two was mainly focused on fixing entries with missing attributes as well as flattening the data to fit the proposed data model In project two the dataset regarding client accesses to the routers had entries which clientOS was missing.

A summary of the approach was to simply convert the JSON file to a dataset in Spark and then flatten it by exploding the readings. For each reading, the attributes were stored in a schema which spark could use. If the clientOs were missing in any readings, the string ”unknown” would be used as a placeholder. Thus all entries are consistent and can be used for further processing.

This approach may in some aspects, also be used in project three. Considering the vehicle teleportation one may assume that the data points N between point A and point B are missing. As proposed in project three one can find the distance between point A to point B and create the missing points in the timestamps that were skipped such that the teleportation is regarded as a single flow of movement.

The problem here may be if the two Points A and B are too far apart and there are multiple lanes or roads to take, then an exact path that should represent the expected path may not be generated. The generated path would then be an assumption of a path, and this may lead to wrongful data. In this case, one may disregard anomalies where the points are too far apart such that an estimated path cannot be created in a way that represents the expected path.

Taking this into consideration, the data cleaning of vehicle teleportation may be applied with a given distance threshold such that any Point A and B which distance is below the threshold, the missing data points between point A to point B can be generated to form path. If the distance is greater than the threshold the two points are disregarded and no path is created.

3.2 B - Clean before or after storing it to master data set

To understand when to clean data before storing, it is critical to reflect upon the fact that a data system exists to answer questions about information that were obtained in the past. Hence when outlining the system, one may want to be able to answer as many questions as possible from such data. To do so, the data can be stored in a form that allows the system to deduce more information from the stored data.

By way of example, given a social media platform that allows people to interact with each other, any data that are produced by the users are stored as is. The reason for this are if any processing happens between storing the data, the amount of information that can be deduced from such processed data might be limited. According to Marz, he defines such property:

> *"(...) call this property rawness. If you can, you want to store the rawest data you can get your hands on. The rawer your data, the more questions you can ask of it."* ((Marz and Warren, 2016, P. 31))

Based on Marz definition we may want to store the rawest data as more information may be deduced.

Returning to the question about when to clean data, according to Marz definition, the data should not be cleaned before storing it in the master dataset. If the data has been under the process of cleaning, then the data may not be as raw as when it was not cleaned. Consequently, any additional data that was removed or cleaned cannot be used to deduce new ideas or information that was not considered yet.

Hence storing raw data may be valuable as one does not know in advance the set of all questions in which one wish to answer. Thus by not cleaning the data. the rawest data is kept and may thus maximise the ability to deduce new insight.

Instead, the data cleaning may happen during batch processing. Hence any cleaning of the data can be customized or tweaked based on how the information should be deduced and which questions to answer. This approach may enable one to reuse the removed additional data on other views as it is still intact in the master dataset.

However, this approach of only cleaning the data during batch processing may come with some trade-offs. Since the cleaning process only happens during batch processing, one will for each batch redundantly have to clean the data again. Consequently, this may increase the computation time as well as the required system resources of the batch layer. If the underlaying system resources are limited, then some precleaning or processing of the data might be required to reduce the performance requirement of the system.

Regardless, when the system just stores the rawest form of data without any cleaning, one may also store noise that does not yield any information. By way of example if the system stores data from a social media platform, the input data might be the full HTML text. Storing the HTML text may be considered as the rawest data, but the HTML code that is included cannot be used to deduce information about the user since they only act as a "container" for the raw data. Thus, one might not regard the HTML as raw data.

Consequently, by not cleaning the data before storing it to the master dataset, may result in the storage of noise which may lead to excessive use of disk space. The space that is used to store the noise could have been used to store additional valuable raw data.

Thus considering the trade-offs about when to clean the data, one may wish to store the rawest form of data in the master dataset to maximise the number questions the system can answer. All the data cleaning and transformation may then happen during batch processing as this approach allows the raw data to be reused when deducing new information that was not considered earlier.

Additionally, when storing the data, one should also consider the level of rawness since storing noise does not contribute to anything other than wasted disk space and system resources since it needs to be cleaned as well.

Finally, some appropriate cleaning may occur before storing the data to the master dataset, if and only if it is possible to pinpoint which part of the data is noise. That is data that cannot be used to deduce any information regarding any given subjects related to the business. Only then does it make sense to clean the data. Else it may be better to just store the raw data as received to maximize the amount of information that can be deduced from it.

4 Question 4 - Views and Applications

4.1 A - Proposed view

The usefulness of such proposed view is very limited to only show the all-time total number of connections. The IT-department may use such view to estimate which routers are used the most and which routers are less used and then consider if some routers need to be upgraded, to cope with the potentially large number of connections. Also, the IT-department may query the view to get a sum of connections that has ever been connected to ITU's network.

The insight from this summation may not say much besides stating the total connections ever made. The IT department may use such view to see whether the network at ITU is used a lot or not based on some expected count. If the number is below the expected count, then one can assume the network may have some problems since the users decide to use it. For instance, if the users decide to create mobile hotspots due to bad wifi coverage or reception.

A more fine-grained view would be to add a timestamp and a location

routerId , *timestamp* , *totalConnections* , *location*

This view would allow the IT-department see the total count of connections for a given router at a specific point in time. This allows the IT-department to estimate based on a time interval which routers are mostly connected at a given location.

For instance, if the IT-department wishes to see which routers are mostly connected during night time they could query the view and see the areas at ITU which are mainly populated with users.

Thus a report could be generated and be handed to facility-maintenance-department which they could use to improve such areas; This could be installing additional coffee and snack machines in those areas to improve the study environment during late night.

4.2 B - Data as infinite stream of real data

Assumed that the that of project three was an infinite stream of real-time data. As the data flows into the system, historical data is also generated at the same time. One may be able to analyze patterns based historical data given that a set of data has already been stored from the stream.

For instance, the historical data can be used to gain insight on which areas are in general dense or sparse in population or traffic. Thus, one can improve the infrastructure to optimize traffic or improve the urbanized space to optimize the general satisfaction in those areas.

The immediate data, which is the data that has just arrived may be used for many different use cases. Assume that the data is used by the law-enforcement, the real-time data could be used to monitor and track suspects or criminals on their current location. This use case may reduce crime but may also introduce ethical and privacy controversies regarding the concept of "Big Brother" which is out of scope in this report.

Alternately, if criminals managed to get access to such data, their use case might be to estimate a real-time escape route with lowest traffic density. A proposed route would then be updated in real-time and which the criminals would use to escape the authorities. This use case might be plausible but may be very unlikely to happen in practice.

Finally, a more realistic application would be to allow the vehicles to track which roads are currently trafficked during rush-hour and then propose an alternate route with less traffic in real-time. This application may thus distribute the traffic evenly across a city and reduce the amount of traffic at areas considered as bottlenecks.

To implement such application, a streaming layer would be included in the speed-layer to enable streaming of data as input. Due to all vehicles are generating data, the system may be required to handle a massive stream of data. Also, it is required that the processed output is accurate, in order for the view to reflect the exact traffic situation. Thus the system would use a micro-batch stream processing as it provides fault-tolerant accuracy at the cost of latency.((Marz and Warren, 2016, P. 255))

The micro-batch processing comes with a trade-off as it may have higher latency due to its high throughput property. However, if each batch is estimated to last between one and five minutes before it is passed to the view, this may satisfy the functional requirement of a view that shows which areas are mostly trafficked during rush hour. The reason why this latency may not violate the functional requirement is assumed that the current traffic situation does not change much within the magnitude of seconds.

However, significant changes to a traffic pattern may occur due to accidents, but it is assumed that the probability of such cases is low enough not to disrupt the functional requirement.

The used technologies in the speed layer/streaming layer would be spark to generate the micro batches. Spark would also be applied in the batch layer to generate batch for the historical data. For the storage layer, Alluxio would be employed as an interface between the batch layer and storage layer to exploit the benefits of Alluxio's unified namespace and memory consolidation. This would promote scalability as one can mount additional storage systems. For the concrete storage layer, HDFS would be used on a cluster of machines to exploit the benefits and scalability of a distributed file system.

References

Alluxio (2017). Unified namespace: Allowing applications to access data anywhere. https://www.alluxio.com/resources/unified-namespace-allowing-applications-to-access-data-anywhere/. (Accessed on 17/12/2017).

Brust, A. (2016). The big data ecosystem is too damn big. https://www.datameer.com/company/datameer-blog/big-data-ecosystem/. (Accessed on 17/12/2017).

ExpressVPN (2017). How does vpn security work? https://www.expressvpn.com/internet-privacy/guides/vpn-security-work/. (Accessed on 17/12/2017).

Marz, N. and Warren, J. (2016). *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co.