# Flink Notebook

*Interactive power of Flink Streaming*

MINOR THESIS
INNOVATION & ENTREPRENEURSHIP

KHUONG-DUY VU

EIT ICT LABS MASTERSCHOOL
UNIVERSITY OF TRENTO
EOTVOS LORAND UNIVERSITY

*2015*

# CONTENTS

I. Introduction 3

1. The business idea 3

2. My research background 4

3. Method to use in the Thesis 5

II. The product environment 5

1. Data Stream 5

2. Continuous Query Language 7

III. Marketability of the Idea 8

1. Market Needs 8

2. Competitors 10

3. Value proposition 12

IV. The project proposal 13

1. The project idea 13

2. The company background 15

3. Feasibility and Revenue model 15

4. Project planning 19

V. Summary 19

VI. References 20

# Flink Notebook

*Interactive power of Flink*

## I. Introduction

### 1. The business idea

In the last few years Big Data generated a lot of buzz along with the launch of several successful big data products. Thanks to contribution from open source community and several giant Internet companies, the big data ecosystem has now approached a tipping point, where the basic infrastructure capabilities of supporting big data challenges are easily available. Entering the next generation of big data, so-called Big Data 2.0, two of its concentrated areas are Velocity and Applications, besides Data Quality. The cause for the former is that data is growing at an exponential rate and the ability to analyse it faster is more important than ever. For instance, sensors can generate data on millions of events per second and store all of those data and response in real-time is non trivial. The latter is helping to overcome the technical challenges of existing frameworks by making them easy to use and understand for everyone to benefit from big data.

As a result, the demand for ***streaming processing*** is increasing a lot these days. Processing big volume of data is not sufficient in the cases that infinite streaming data is arriving at high speed and users require a system to process fast and react to any incident immediately. In addition, although hardware price has plunged year over year, it's still expensive to equip a storage which is growing terabytes every day for batch analysis. Streaming processing engines are designed to operate high volume in real time with a scalable, high available and fault tolerant architecture.

One of the disadvantages to users is that many big data frameworks provide a rich imperative API code only to process data stream. First, users must spend time learning API documentation properly since those APIs are fairly new to them. Therefore, the cycle time to develop products taking longer. Second, given that most big data applications are fairly simple application-wise, a block of API codes might be less optimal to use for most the popular queries[14]. Third, the only way to share your work is to pack it as a library or service that need to be deployed again. Fourth, the results of streaming queries keep changing upon the time. Thus, it demands a visual way to observe the streaming instead of a sequence of boring numbers. Those above drawbacks apparently provide an unfriendly UX that inhibit both productivity and system performance. It is really against what we expect from "Application" aspect of Big Data 2.0.

In this Minor Thesis, I would like to promote "Flink Notebook" - an interactive, collaborative web-based interface built on top of Apache Flink that provides a complete streaming processing solution. Flink Notebook brings to developers a friendly, effective environment without scarifying the power of Flink Streaming.

Basically, Flink Notebook allows users to write standard SQL queries on webpages. The notebook issues corresponding commands to Flink to make it operated continuously on data as they arrive. The result is updated incrementally in real-time and displayed on Notebook with visual aid. In short, users need to know nothing about Flink API, except for very common SQL basis. The result is also brought to life by a powerful visualisation web pages.

Flink Notebook facilitates the productivity by allowing several colleagues to collaborate on the same notebook on the same work in real-time despite the physically location differences. The work can also be captured and exported to a portable format that can be viewed shared, restored later at ease.

## 2. **My research background**

During my 3-month internship I am working on Designing and Implementing an SQL dialect for Apache Flink, parts of the topic is related to online and streaming data processing and user UX. I realise an eager need of a new application, which is presented in this minor thesis, to engage developers into building Big Data applications.

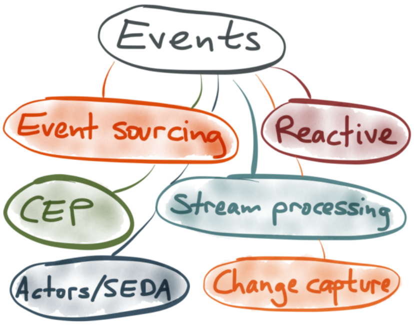


Fig 1: buzzword of Stream

Streaming Processing is not a new concept. Indeed the similar concept, Complex Event Processing (CPE) had been proposed from the 1990s by Event Simulation Research at Stanford [1]. Since that time, people have started generating a lot of different buzzwords around it [2] and often reinventing ideas borrowing from other fields, but using a different vocabularies to describe the same concepts. Basically, the idea is to analyse one or multiple data streams to identify meaningful phenomena and respond to them as quickly as possible.

According to CEP Tooling Market Survey 2014 [3], since 1996, there has existed more than 30 companies providing Streaming Processing solutions. All the major software vendors (IBM, Oracle, Microsoft, SAP) also have good to excellent offerings in the CEP space for customers.

However, since a massive amount of data is growing rapidly every second, Hadoop[1] is emerging distributed processing ecosystem today. Thanks to Hadoop, people can build a large scalable distributed system on Cloud. Even though Hadoop is designed to scale system up to thousands of machines with very high degree of fault tolerance, it is optimised to run batch

[1] Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache projects sponsored by the Apache Software Foundation

jobs with a huge load of computation. Because of time factor, Hadoop has limited value in online environment where fast processing is crucial. Therefore, existing CEP solutions are barely compatible with Hadoop ecosystem. We demand a new sort of streaming framework which is able to integrate on top of Hadoop system. Apache Flink is one of these frameworks.

In term of cooperative work, iPython Notebook is web-based interactive environment where one can combine a working session into a single document. The document can be shared and converted to other common formats such HTML, PDF for quick review. Google Docs allows multiple users to collaborate on a document in realtime. Our Flink Notebook inherits both of these technologies to provide an interactive, collaborative environment along with super power of Flink.

### 3. **Method to use in the Thesis**

Thesis proposes Flink Notebook project and its high-fidelity prototype (Appendix B). Moreover, we are doing market research to figure out why Flink Notebook is necessary. Lastly, we analyse an applied business model how to make Flink built both technically and financially.

Technically, a project developing Flink Notebook requires us to have a deep understanding on Flink Streaming, interpreter theory and web programming skills. First, we study the principle of Stream Processing and data flow in Flink. Second, we need to design SQL-like syntax which users can use to issue command to Flink. Third, a web application helps to interact with users.

The project involves many big data developers during design process. According to regular project management methods, we have to observer, to monitor their behaviour during testing; we also need to interview them for valuable feedback. Finally, to shorten the developing cycle, we take the advantage of community to integrate many open source project into ours.

# II. The product environment

## 1. Data Stream

*A **stream S** is a countably continuous and infinite set of elements $< s, t > \in S$, where s is a tuple belonging to the schema of S and $t \in T$ is the timestamp of the element.*

For example, data stream of stock transactions in security market. Each transaction contains information about <*stock symbol, quantity, price, exchange, timestamp*> which is schema for all tuples.

There are several stream definitions varying based on the execution model of systems. On the previous definition, a timestamp attribute can be a non-strict total ordered application timestamp because there is no order between elements which share the identical timestamp. Thus system may not rely on it to order the coming tuples and design which one to react before others. To overcome this drawback, an element can contain an extra physical identifier ϕ [11] such as increment tuple id to specify its order. The tuple with smaller id means that it arrives and should be processed before the tuples with bigger id. Another way to identify the order of a tuple item is to separate the concept of application and system timestamp. In SECRET model [12], each stream item is composed of a tuple for event contents, an application timestamp, a system timestamp, and a batch-id value. The idea of batch-id is critical to SECRET system we do not mention in the thesis. In short, in practice, we assume that elements of a stream are strictly totally ordered by the system timestamp and physical identifier.

In Apache Flink, we propose an extended definition of a data stream as:

> *A **stream S** is a countably infinite set of elements $s \in S$.*
>
> *Each stream element s: $<v, t_{app}, t_{sys}>$, consists of a relational tuple v conforming to a schema TP, with an optional application time value $t_{app} = f(v) \in T$, and a timestamp $t_{sys}$ generated automatically by system, due to the event arrival.*

The partial function *f* is to extract application timestamp from tuple, $f : TP \rightarrow T$. For example, for the sake of data integration, function *f* converts a timestamp in GMT+7 to one in current timezone GMT+1. This function is rather important in case of data integration with many sources.

Two advantages of our model is:

- Users are able to use either application or system timestamp to querying stream. However, the order among stream elements is reserved based on their arrival in related to the implicit system timestamp.
- Partial function *f* gives more flexibility to define the application timestamp value. Thus users have more chances to observe data in different perspective.

**Stream Windows**

From the system's point of view, it is often infeasible to maintain the entire history of the input data stream. Because data stream is running infinitely, we do not know when it ends. It nearly impossible to query over the entire stream with some operators such as sum, average. Accumulating a tuple attribute for entire stream may results to a very big value causing buffer overflow. From the user's point of view, recently data may be insight and more useful to make a data-driven decision. Those reasons motivated the use of windows to restrict to the scope of continuous queries.

*A **Window W** over a stream* S *is a finite subset of stream* S

A window over streaming data can be created, buffering a continuous sequence of individual tuples. However, the size of window is a finite number so that system must decide what and how to buffer data based on the window specification.

The specification consists of several parameters:

- An optional partitioning clause, which partitions data in window into several groups. Query on window will be taken place regard for each group, instead of the whole window. For instance, elements are grouped by name of good category in super market transactions.

- A window size, that may be expressed either as the number of tuples included in it or as the temporal interval spanning its contents. For example, show the last 10 purchases or show all the purchases in last 1 minute.

- A window slide, the distance between the upper bound of 2 consecutive windows (e.g., waiting 2 seconds or 5 data elements before starting a new window). This crucial property determines whether and in what way a window change state over time. If the window slide parameter is missing, system can assign implicitly that the slide size is equal to the window size. In this case, we have a stream of disjoint windows so called batch windows or tumbling windows.

- An optional filtering predicate, keeping only elements that satisfy the given predicate.

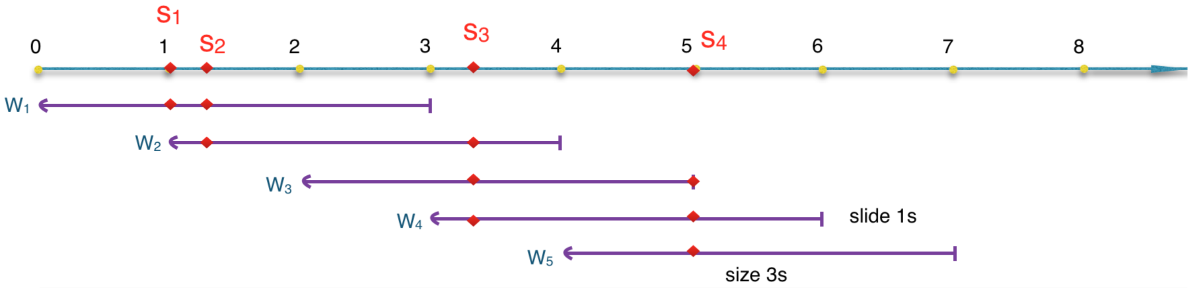


Fig 2: Windowed stream

Figure 2 depicts a windowed stream which is derived from original stream with the rule “windowing all transactions last 3 minutes once every minute”. It means that every minute has passed, system will report all transaction happened last 3 minutes. Window size of 3 minutes slide by 1 minute. Users are able to query on each window since its size is finite.

## 2. Continuous Query Language

In the data stream model, some of all the input data that are to be operated are not available for random access from disk or memory, but rather arrive as one or more continuous data streams. Data stream differs from the conventional stored relation model in several ways. Recalling two of important properties that make queries on Data stream are so different:

- They are produced continuously and, therefore, have unbounded, or at least unknown, length. We do not know when we will have seen the entire input so that we cannot determine which portion of data to query. Therefore, non-blocking query operators are more suitable for stream processing. The non-blocking query operators such as selection, projection, merge two stream and so on, do not need to wait until it has seen the entire input before producing query results. However, stream does not cope with blocking queries such as aggregation.

- Each element of data stream is accompanied to a timestamp to indicate its order on stream which is immaterial in conventional database.

One common approach [10] is that we discretised original stream into sequence of windows. The number of tuples in window is finite so that we are able to implement blocking query operators on windows such as to compute the average volumes of all transactions in last 3 hours.

Timestamp is crucial to determine the tuple order and which windows it belongs to. However, since we know which tuples belongs to a given window, tuple timestamp in not necessary anymore.

> *A **relation R** is a mapping from each time instant to a finite*
> *but unbounded bag of tuples regardless of these orders*

The concept of relation *R(t)* now is close to the concept of relational tables so that we can apply traditional database query on relation *R(t).*

In STREAM[10] engine, there are 3 classes of operators:

- *Stream-to-relation* operators that produce a relation from stream via windowing. For example, show continuously last 10 transactions every minute.
- *Relation-to-Relation* operators that produce a relation from one or more other relations. These operators are derived from traditional relational tables such as aggregation, group, order and so on. For example, continuously computes the revenue last 3 hours.
- *Relation-to-Stream* operators that produce a stream from a relation by assigning a timestamp value.

One drawback of STREAM is that when producing output by simple non-blocking query operators, STREAM maps stream to relation, operates on it and maps relation to stream again. They do not provide a directly *Stream-to-Stream* operators for simple queries. In Flink, Not only does we support both 3 above operators but also extends a set of *Stream-to-Stream* operators explicitly.

# III. Marketability of the Idea

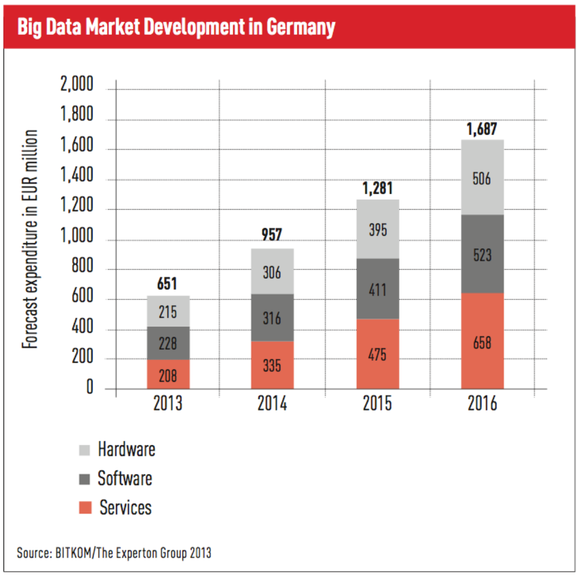
## 1. Market Needs

First and foremost, we take a glance on Big Data promising forecast in global market. According to well-known IDC forecast [4], the big data and analytics market will reach *$125 billion* worldwide in 2015. Rich media analytics such as videos, audios and images will contribute as a crucial driver to prompt a rise in big data applications. 25% of top IT vendors may announce Data-as-a-Service solutions based on expanding cloud ecosystems (expected to reach $118 billion in 2015).

One of the major business lines, which could not be adopted successfully without Stream Processing supports, is Internet of Things (IoT). IoT back end as a service (BaaS) will emerge and require a mature Stream Processing platform. Expenditure on IoT may exceed $1.7 trillion, up 14% from 2014 and go up to $3 trillion by 2020. In other words, it is growing with a five-year CAGR of 30%. Therefore, we expect IoT would bring a huge slice of the cake to Stream Processing in revenue and development.

How organisations embrace the Big Data trending? The report 'Big & Fast Data: The Rise of Insight-Driven Business' conduct by *Capgemini*, surveyed 1,000 senior decision makers in nine regions and nine industries. It reveals the fact that 65 % of organizations acknowledge they are at risk of becoming uncompetitive unless they embrace new data analytics solutions. Specifically, accessing Big Data faster is where C-suite executives see the most value – 77 percent state that decision makers increasingly require data in real-time. However, More than half (52 percent) of respondents reported that developing fast insights from data was hampered by limitations in the IT development process.

In short, the market is in strong demand of real-time Big Data Platform, along with supported tools to provide better development process.



Source: BITKOM/The Experton Group 2013

Fig 3: German Market

We are going to supply services to the German and Hungarian market at the beginning. The hint is that our teams are based on Berlin and Budapest so that we are capable of nurturing our initial market at the beginning. Even though the German Big Data market still appears to be at an early stage, it is expected to from EUR 650 million in 2013 to almost EUR 1.7 billion in 2016. Additionally, Software and Services would account for 70% of that value.

We did a research on [2]*Meetup* groups using their API to see how active Big Data and Data Analytics communities are.

In Berlin, we found about 30 Data-related groups with more than 9300 members in total (Figure 4). In Budapest, we are also able to reach about 4700 members in total (Figure 5). Even each developer can join more than 1 group but the number is till quite considerable.

[2] http://www.meetup.com

Fig 4:**Big Data Meetup Group in Berlin**

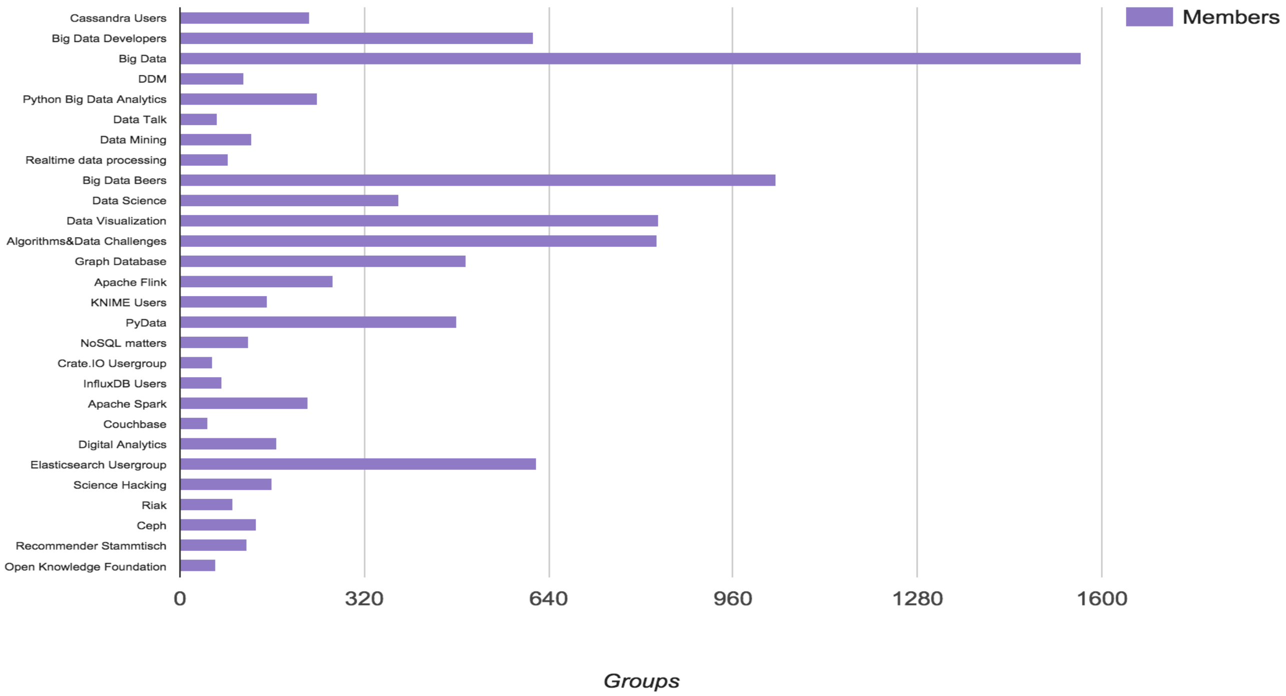
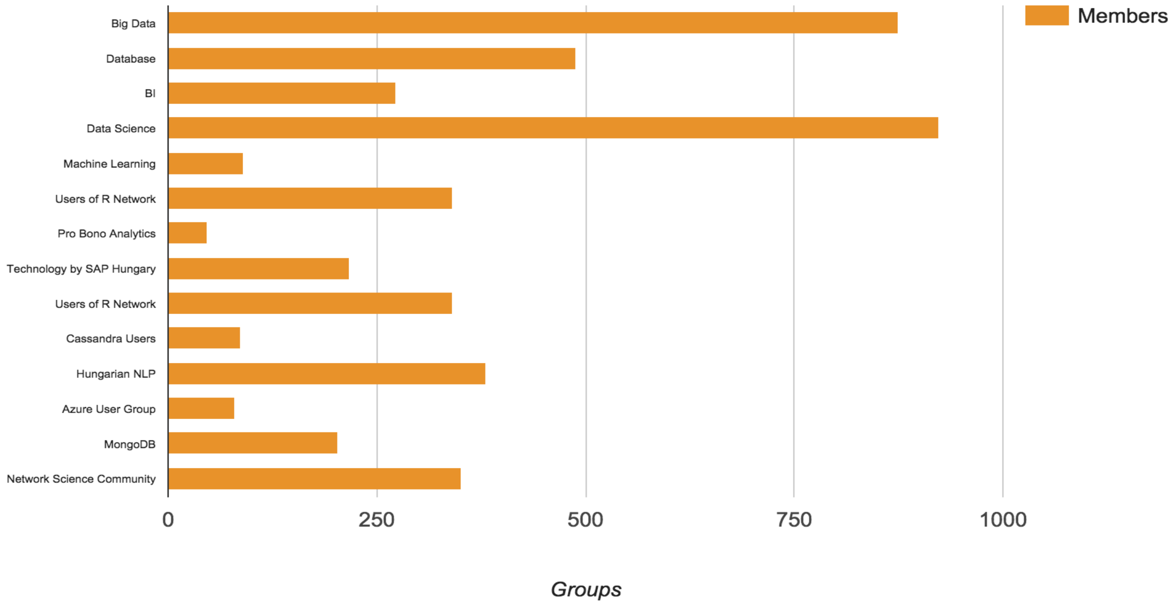


Fig 5: **Big Data Meetup Groups in Budapest**



## 2. Competitors

According to the CEP market players to end of 2014, conducted by Paul Vincent and his team, CEP tools have a long history (Fig 6) but many of them are acquired or left the market. Almost tools and platform support query languages which we also provide. Even all the major and medium software vendors offer their own solutions but we target to the open source community with some emerging top projects such as Apache Storm, Apache Spark and Apache Samza.

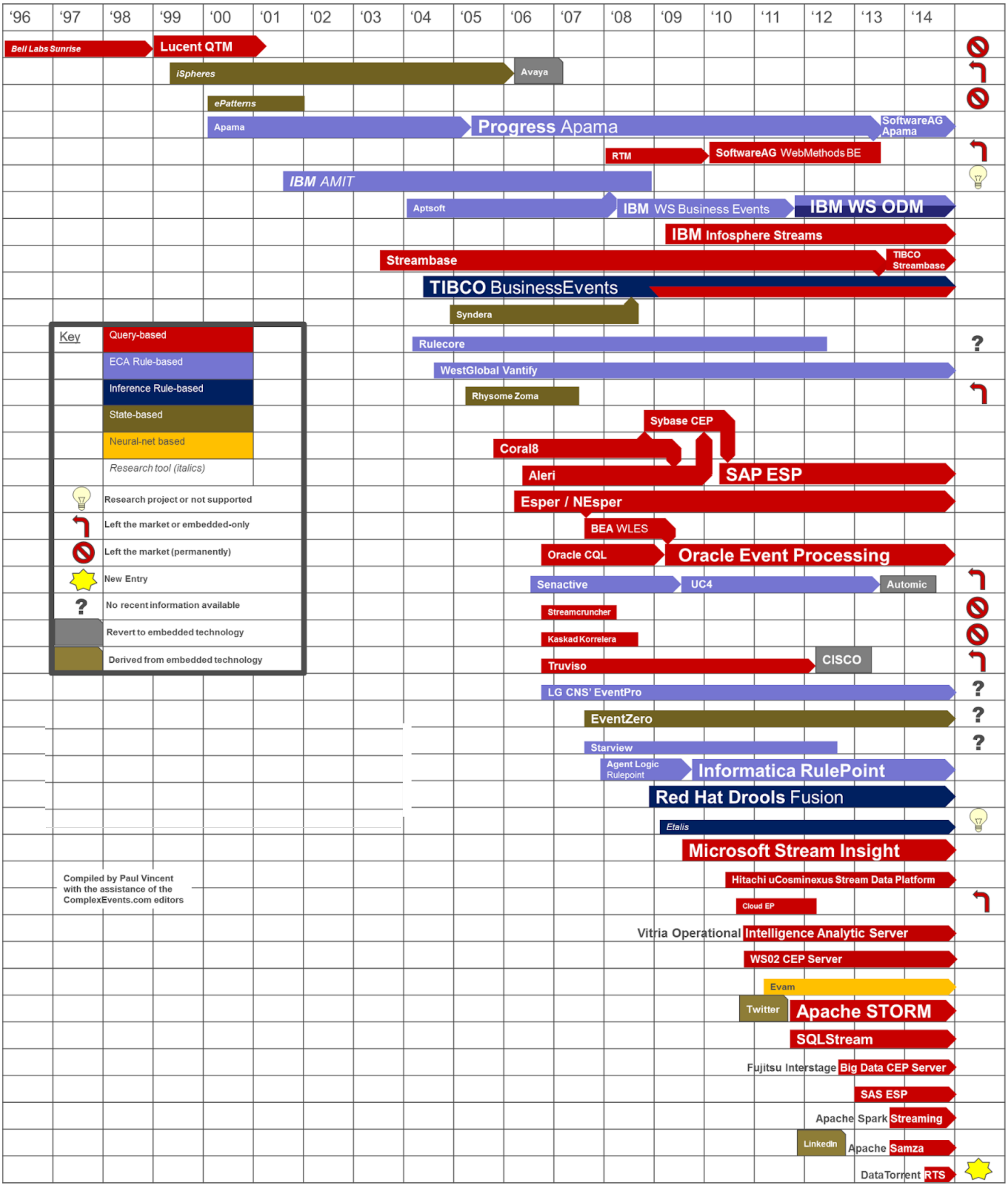


Fig 6: Complex Event Processing - market players – end of 2014

We briefly compare those tools to our solution in term of Streaming Processing in Table 1:

| | Storm | Spark | Samza | Flink | (note) |
|---|---|---|---|---|---|
| **Delivery Semantics** | >1 | 1 | >1 | 1 | smaller is better |
| **State Management** | Stateless | Stateful | Stateful | Stateful | stateful  is better |
| **Latency** | sub-second | second | sub-second | sub-second | smaller is better |
| **Language support** | Any | Scala, Java, Python | Scala, Java | Scala, Java | more is better |
| **Query Language Support** | Yes | Yes | Yes | Yes | |
| **Notebook features** | No | No | No | Yes | |

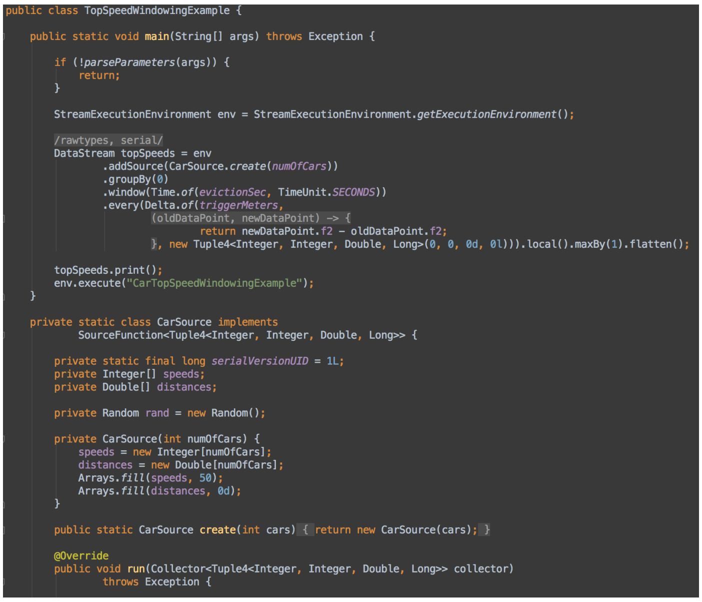
Table 1: Direct Competitors

*Notebook features: *collaborative*, *interactive interface* ; *file format to share*

## 3. Value proposition

### Customer pains:

- Working on most of the existing Big Data frameworks, developers encounter several disadvantages:
- They need to spend time learning API via documentations. The API is non-standardized and may not be well-designed, thanks to the committers' expertise. In addition, for most popular use cases, the programs are conducted from API by developers may be less optimal in term of time and space cost.
- Result from querying data stream is continuously emitted, users have hard problems to observe the change to react.
- It is not easy to share your work s with colleagues except for packed libraries or services along with additional documentations.



```
public class TopSpeedWindowingExample {

    public static void main(String[] args) throws Exception {

        if (!parseParameters(args)) {
            return;
        }

        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

        /rawtypes, serial/
        DataStream topSpeeds = env
                .addSource(CarSource.create(numOfCars))
                .groupBy(0)
                .window(Time.of(evictionSec, TimeUnit.SECONDS))
                .every(Delta.of(triggerMeters,
                        (oldDataPoint, newDataPoint) -> {
                            return newDataPoint.f2 - oldDataPoint.f2;
                        }, new Tuple4<Integer, Integer, Double, Long>(0, 0, 0d, 0l))).local().maxBy(1).flatten();

        topSpeeds.print();
        env.execute("CarTopSpeedWindowingExample");
    }

    private static class CarSource implements
            SourceFunction<Tuple4<Integer, Integer, Double, Long>> {

        private static final long serialVersionUID = 1L;
        private Integer[] speeds;
        private Double[] distances;

        private Random rand = new Random();

        private CarSource(int numOfCars) {
            speeds = new Integer[numOfCars];
            distances = new Double[numOfCars];
            Arrays.fill(speeds, 50);
            Arrays.fill(distances, 0d);
        }

        public static CarSource create(int cars) { return new CarSource(cars); }

        @Override
        public void run(Collector<Tuple4<Integer, Integer, Double, Long>> collector)
                throws Exception {
```

Fig 7: Imperative code

**Customer gains:**

- All of developers is familiar with SQL syntax which is widely adopted as a standard for data query language. Therefore, It would be much more efficient if they could issue command to Big Data application using SQL-like syntax. No much effort to learn that syntax
- A visualized output would make users easier to examine and monitor  the flow of data.
- Developers wish to have a portable file to wrap up their work, share to people and show how the output is expected. Need not to recompile the program. Share notebook to colleagues without having them install anything.

Flink Notebook drastically reduces the overhead of organising code, output and note files which allows to spend more time doing real work.

**Products and Services**

To relieve our developers' pain and accelerate their work, we would like to introduce Flink Notebook with several advanced features:

- It is a *web-based* UI environment helping users interact with Flink platform. As long as we are able to connect to Flink, we can run Flink notebook *everywhere with a browser*.
- Command the system with *SQL* syntax.
- An *interactive* UI : users are able to type their command and receive the result as real-time charts.
- Flink notebook save your work (commands and visualized outputs) to a portable**.fnb* file. We can *export* it, *hand over* to colleagues. They *could open the file at ease in any browser.* If your machine does not connect to Flink infrastructure, you can see all of issued command and its output non-interactively. Otherwise, you are empowered to re-run all of authenticated command and observe new results.
- Users can invite other people to work on the same **.fnb* format collaboratively regardless of different physical location.

# IV. The project proposal

## 1. The project idea

Stream Processing have been a very active and diverse area of research, commercial & open source development since 1990'. Along with traditional business event which is streaming stock data for financial services, streaming satellite data for the military or real-time vehicle-location data for transportation and logistics businesses, companies in multiple industries must handle large volumes of complex data in realtime.

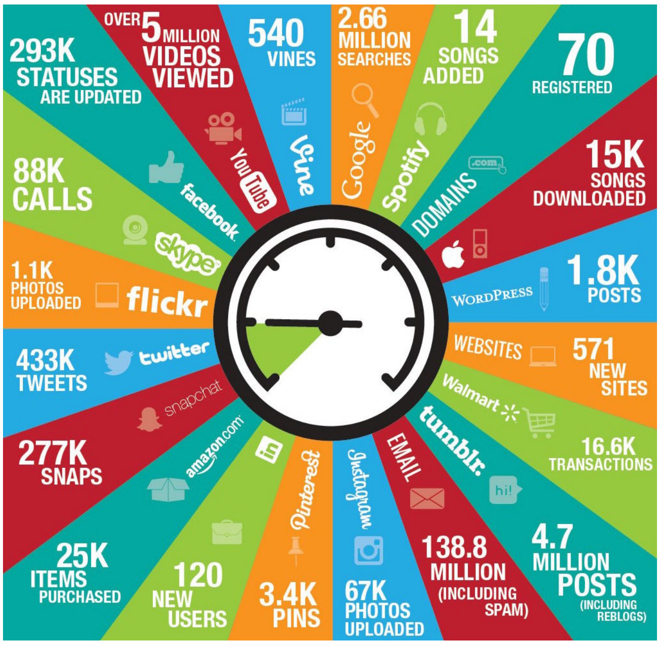


Fig 7: What happens on Internet on 60 seconds

However, Stream Processing gains more and more attention recently, thanks to the explosion of mobile devices and trendy Internet of Things. The number of smartphone users in 2014 is more than 1.64 billions which accounts for 38.4% of mobile phone users[9]. According to market research in previous section, expenditure on IoT may exceed $1.7 trillion and reach to $3 trillion in 2020. The enormous number of Internet-connected devices and the ubiquity of high-speed connectivity add to explosion of mobile data. Figure 7 depicts the number of events happens in 60 seconds on Internet. Even though the system receive an extremely high velocities of data and event throughput, users expect to receive response immediately without any delay. Those challenges can only be solved by a stream processing system, instead of traditional database technology.

At the same time, demand for business process agility and execution has only grown. Useful tools are largely available but inhibit the development process because of its poor User Experience. Most of them require users to write complicate API code, which is not trivial to many users. Scientific users tell IDC that these tools can be great for retrieving and moving through complex data, but they do not allow researchers to take the next step and pose intelligent questions.

Sophisticated tools for data integration and analysis on this scale are largely lacking today. There are opportunities to create tools and applications for Big Data. Vendors that create tools and applications for use at this scale can use them as a lever to seize market leadership positions in the Big Data market.

Main target of the project is to bring out this Flink Notebook which provides users a truly collaborative and interactive interface to help them deploy their solution as soon as possible. Furthermore, the beautiful Flink Notebook already hides the complicated but powerful analytic platform of Flink. The integrated solution brings users a powerful machine with beautiful UX.

We propose to initiate a development project to create the Flink Notebook and to assure necessary resources for the activities planned.

## 2. The company background

***Data Artisan***: a spin-off company from *Technische Universitat Berlin*, Germany. The founder team has a high level of expertise in distributed data-intensive system. They are aiming to develop the next generation technology of Big Data Analytics and Processing via Apache Flink.

***Apache Flink***: is a top-level project of the Apache Software Foundation with a community of over 100 contributors from industry and academia. Flink is an open source system for expressive, declarative, fast, and efficient Big Data analysis. It originates from EU-granted Stratosphere project since 2000s.

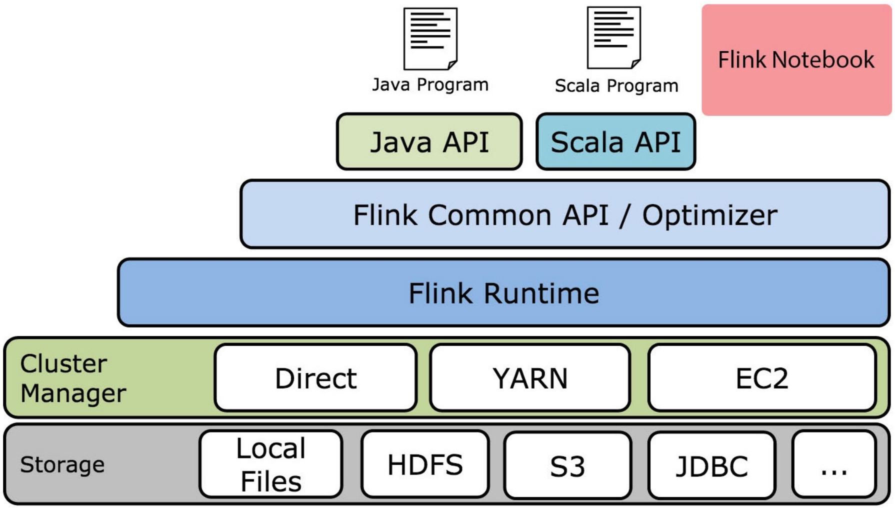


Fig 8: Apache Flink stacks

Basically, users are able to write a Java/Scala program on Apache Flink but it was painful and complicated as we mentioned before. *Flink Notebook* is considered as an application written on Scala or Java API. It receives user's commands as input and generates Flink API code automatically. Users need not to care about Java/Scala API anymore.

We are highly competent engineers on Distributed System in general and concurrency in particular so that with only two engineers we can build up the collaborative feature at ease. However, Flink notebook serves a web-based application, thus we need a small team of one web developers and one web designer whose skills are complementary to ours.

## 3. Feasibility and Revenue model

### a. Revenue Streams

Be a part of Apache Flink, we are following the Open-source business model. We warmly welcome and encourage users to use our Flink Notebook freely. Additionally, we would like to support users to make the best out of Flink Notebook. Besides public supports on forum and mailing group, customers may require private supports from our dedicated experts.

Customers get support on:

- Trouble-shooting and reporting errors.
- Refactoring data and architecture model.
- Training on collaborative development cycle with Flink Notebook.
- Rich resource and materials

Customers can choose either session supports or annual subscription.

- Session supports, we charge customer according how many hours of support. The fee is about 50$/hour
- Annual subscription plan, customer receives 24/7 supports within a year which cost s 800$ /year.

## b. Key Activities

We put most of our resources to build a product which bring the best experience to users. Besides, marketing strategy is rather important to engage users and enlarge our communities.

### *Design and development*

We strictly follow agile methods for building software to eliminate the cost during development. The process is broken into a consequence of 10-day *sprints*. The duration of a sprint depends on the difficulty of problems but no longer than 3 weeks. Each sprint consists of 5 steps [15]:

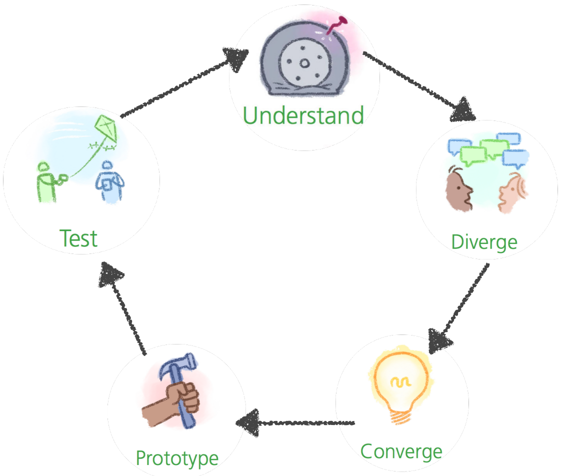


Fig 9: Design Sprint

To improve the quality of codes, we are applying Test-Driven Development method (TDD). Instead of writing code at the early stage, we first write an automated test case that defines a desired improvement or new function, then produce the minimum amount of code to pass the test, finally refactor the program to acceptable standards.

**Table 2: Sprint design in details [15]**

| | Duration | Desctiption | Tools |
|---|---|---|---|
| **1. Understand** | 1 - 3 days | Discover insights and define the problem | Facts Assumptions, Discovery interview, Task Modeling, Empathy Map, Journey Maps, Who/Do |
| **2. Diverge** | 1/2 - 2 day | Generate solutions | Challenge Maps, Scenarios, Storyboarding, Six-ups, Who/Do |
| **3. Converse** | 1/2- 2 day | Rank and select solutions | Assumption Table, 3-12-3, Storyboarding, Dot-voting |
| **4. Prototype** | 1 - 3 days | Build the solutions | Illustrator, Photoshop or Sketch, HTML/CSS, Keynote or PowerPoint, coding |
| **5. Test** | 1/2 - 1 day | Observer the customer impact | • Testing Interview • Plus/ Delta |

Passing all test cases ensures that our program satisfies all of our designed requirements and minimise the number of bugs.

To collaborate on the project, we utilize [3]G*ithub* to host the source code repository and [4]Jira to capture and organize issues, assign works, and follow team activities.

### *Marketing strategies*

Aligned with Apache Flink, we can propagate the idea of Flink Notebook through various Big Data Conferences (Strata + Hadoop, Apache Conference), invited talks & tutorials at *Meetup* group. As business model we also give 3 months of free support for the first 30 customers.

Building a screen cast channel on Vimeo, Youtube to give away basic tutorials and tips.

To stand out from other frameworks, we also establish an expert/evangelist network around the Flink to help raise many discussions/debates on the features on Flink in general and Flink notebook in particular. We manage to keep Flink Notebook as a hot topic and mentioned frequently.

[3] http://www.github.com

[4] https://www.atlassian.com/software/jira

### c. Key Resources

***Manpower***: We already have a vast support from academic and industrial community to develop Flink (20 official committers, 100 contributors, academic supervisors from TU Berlin, Sztaki, KTH,  Apache supports), but  we delicate team of 3 full-time developers and 1 designer to make Flink Notebook developed.

Flink Notebook is also backed by a robust, sustainable developer community and the governance framework and processes of Flink and Apache Software Foundation.

***Finance***: Euro 20.000 for the team operation in 6 months. Project is mostly granted by EU programs via *TU Berlin* and *Sztaki*. Other parts would come from generous donation. We have few early adopted customers testing Apache Flink  in production such as ResearchGate - a social network for scientists, music streaming service Spotify Ltd. and travel software provider Amadeus IT Group, SA and so on. We would like to ask for financial supports from them as well.

**Available assets**: in *Sztaki*, we are allowed to use a data cluster of 30 nodes (RAM 128Gb, Hard disk 400Tb). Free office for up to 10 people.

### d. Key Partnerships

We are glad to make a partner with Google to bring easy Flink deployment to the Google Cloud Platform, and enable Google Cloud Dataflow users to leverage Apache Flink as a backend. Now users even can use Flink under Google Cloud platform. And Flink Notebook would be integrated into it soon.

### e. Cost Structure

Cost during development period:

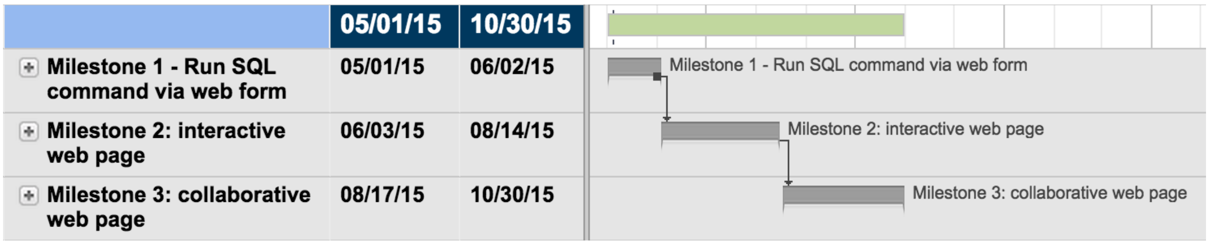
| **Fixed Cost** | **Cost** | **\|** | **Virable Cost** | **Cost** |
|---|---|---|---|---|
| (in Euro) | (monthly) | | (in Euro) | |
| **Salary** | 2800 | \| | **Marketing** | 3000 |
| **Utilities** | 100 | \| | | |
| **Jira subscription** | 20 | \| | | |
| **Server & Domain** | 50 | \| | | |
| **Total** | **2970** | | | |

Table 3: Cost structure

Since we got a lot of supports from Apache, Sztaki, TU Berlin, we spend our money mostly to pay monthly salary for 4 team members. We also plan to spend around Euro 3000 in first 6 months for marketing including organising *Meetup* talks, conference fees, influencers' writing.

## 4. **Project planning**

We use an agile methodology (SCRUM technology): it means the project is progressing through Sprint meetings. Team leader will be responsible for monitoring result at sprints, and especially achieving milestones through the process as you can see in Figure 10



| | 05/01/15 | 10/30/15 |
|---|---|---|
| Milestone 1 - Run SQL command via web form | 05/01/15 | 06/02/15 |
| Milestone 2: interactive web page | 06/03/15 | 08/14/15 |
| Milestone 3: collaborative web page | 08/17/15 | 10/30/15 |

Fig 10: Milestones

# V. Summary

In spite of many technical advantages agains our competitors, Flink still struggle to gain widespread adoption in market. We propose Flink Notebook to bring out an innovative environment where users can experience a powerful stream processing machine with superior UX design. We relieve developers from immaterial procedures, which may consume much time, to focus on main program logic ONLY. We are strongly in hope that Flink Notebook help us gain traction from big data community and continue growing.

Flink Notebook is coming. By now we have tested our entire SQL interpreter successfully. Thus users are able to query SQL commands on Flink engine. We are discussing to integrate SQL module into next release Flink version 0.9.2.

Working on Flink Notebook gives me a great chance to learn how a large scale open source project is managed smoothly. I believe that open source project is going to dominant close source in very near future.

# VI. References

[1] "The Stanford Rapide™ Project", Stanford, retrieved on May 1, 2015

[2] Kleppmann, Martin. "Stream Processing, Event Sourcing, Reactive, CEP... and Making Sense of It All". January 29, 2015. Retrived on May 1, 2015

[3] Vincent, Paul. "CEP Tooling Market Survey 2014". December 3, 2014. Retrived on May 1, 2015

[4] Pree, Gill. "6 Predictions For The $125 Billion Big Data Analytics Market in 2015" Forbes Tech. Dec 11, 2014. Retrieved on May 2 2015

[5] "New global study by Capgemini and EMC shows Big Data driving market disruption, leaving many organizations fearing irrelevance". PRNewswire. March 10, 2015. Retrieved on May 3 2015

[6] Siciliani, Tony. "Streaming Big Data: Storm, Spark and Samza". Java Dzone. Feb 2 2015

[8]Otávio M. de Carvalho, Eduardo Roloff, and Philippe O. A. Navaux. *A Survey of the State-of-the-art in Event Processing*. 11th Workshop on Parallel and DistributedProcessing (WSPPD), August 2013

[9] "2 Billion Consumers Worldwide to Get Smart(phones) by 2016". Emarketer. Dec 11, 2014. Retrieved on May 4 2015

[10] Arasu, A., Babu, S., and Widom, J. (2006). The cql continuous query language: Semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142.

[11] Petit, L., Labbé, C., and Roncancio, C. L. (2010). An algebric window model for data stream management. In *Proceedings of the Ninth ACM International Workshop on Data Engineering for Wireless and Mobile Access*, MobiDE '10, pages 17–24, New York, NY, USA. ACM.

[12] Dindar, N., Tatbul, N., Miller, R. J., Haas, L. M., and Botan, I. (2013). Modeling the execution semantics of stream processing engines with secret. *The VLDB Journal*, 22(4):421–446.

[13] Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT- SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA. ACM.

[14] Martin, K. (2014) Designing Data-Intensive Applications. O'Reilly Media

[15] Banfield R., C. Todd Lombardo., Wax T. (2015). Design Sprint. O'Reilly Media

# APPENDIX A: ONE-PAGER

## Name

Flink Notebook

## Email

note@data-artisans.com

## Website

https://note.data-artisans.com

## The current problem to solve

In most of the existing distributed data-intensive systems, especially Flink Streaming, the only provided rich imperative code makes developers or users less productive to build big data applications .

- Non-standard: Each framework has a different set of API so that developers must learn how to use its API properly to take the advantage of the framework. Probably those API is not standardised. Therefore, developer even takes long time to master it.

- Less Compacted and Optimized: given that most big data applications are fairly simple application-wise, a block of API codes might be less optimal to use for most the popular queries. In addition, the imperative code is usually more complicated and less compact than other declarative language such as SQL.

- Interaction: During the development cycle, developer usually compiles the source code and works on command line which is less interactive and definitely not friendly.

- Collaboration and Sharing: The only way to share your work is to pack it as a library or publish a service. Moreover, to collaborate with your colleagues simultaneously, you have no choice but sitting in front of a same machine.

## About the company and the product

***Data Artisan***: a spin-off company from Technische Universitat Berlin, Germany. The founder team has a high level of expertise in distributed data-intensive system. They are aiming to develop the next generation technology of Big Data Analytics and Processing via Apache Flink.

***Apache Flink***: is a top-level project of the Apache Software Foundation with a community of over 100 contributors from industry and academia. Flink is an open source system for expressive, declarative, fast, and efficient Big Data analysis. It originates from EU-granted Stratosphere project since 2000s.

***Flink Notebook***: an interactive , collaborative web-based environment inspired by iPython Notebook and Google Docs. Flink Notebook allows to build an analytics application on Flink easier and more productive in a very friendly way.

## The market need, customer segment

### *The market needs*

We are going to supply services to the German market at the beginning. Even though the German Big Data market still appears to be at an early stage, it is expected to from EUR 650 million in 2013 to almost EUR 1.7 billion in 2016. Additionally, Software and Services would account for 70% of that value.

### *Customer Segments*

We target Big Data developers in the German market. These developers are in charge of building and maintaining a data analytics platform for companies in different industrial sectors such as finance, ecommerce, Internet Of Things. They are looking for a scalable, reliable and high-available framework with excellent UX.

### Competitors, and our strategies

The idea of Complex Event Processing was declared from early 1900s. Therefore, we are challenged by many services from both all major software vendors (IBM, Microsoft, SAP, Oracle…) and new players (DataBricks, Adatao and others). However, our product owns several crucial winning features that do not exist or are still under developing at other products. We outperform them by a true streaming processing with highly optimised core.

With Flink Notebook's excellent environment, developing a big data application will never be easier than that.

### Partners

Recently, we are making a partnership agreement with Google. We now enable Google Cloud Dataflow users to leverage Apache Flink as a backend. Another agreement with Amazon web service and Microsoft Azure is on process.

### History, references, prior results

I have more than 3 years of experience on Data Mining and Analytics, especially, one year on Big Data analytics for News and Movie recommendation system. Additionally, I have jointed a collaborative research between Vietnam National University and Panasonic on "User behaviour on mobile phone".

### Explanation

***Big Data***: a popular term used to describe the exponential growth and availability of data, both structured and unstructured. Three most important features of Big Data are  Volume – Velocity (of access) - Variety.

***Data Stream***: Uninterrupted flow of a long sequence of data, such as in audio and video data files.

***Stream Processing***: analyse one or multiple data streaming to identify meaningful events and respond to them as quickly as possible.

### Alternative to grow

***Manpower***: We already have a vast support from academic and industrial community to develop Flink (20 official committers, 100 contributors, academic supervisors from TU Berlin, Sztaki, KTH, Apache supports), but  we need a delicate team of 3 developers and 1 designer to make Flink Notebook developed.

***Size of capital*** : Euro 20.000 for the team operation in 6 months.

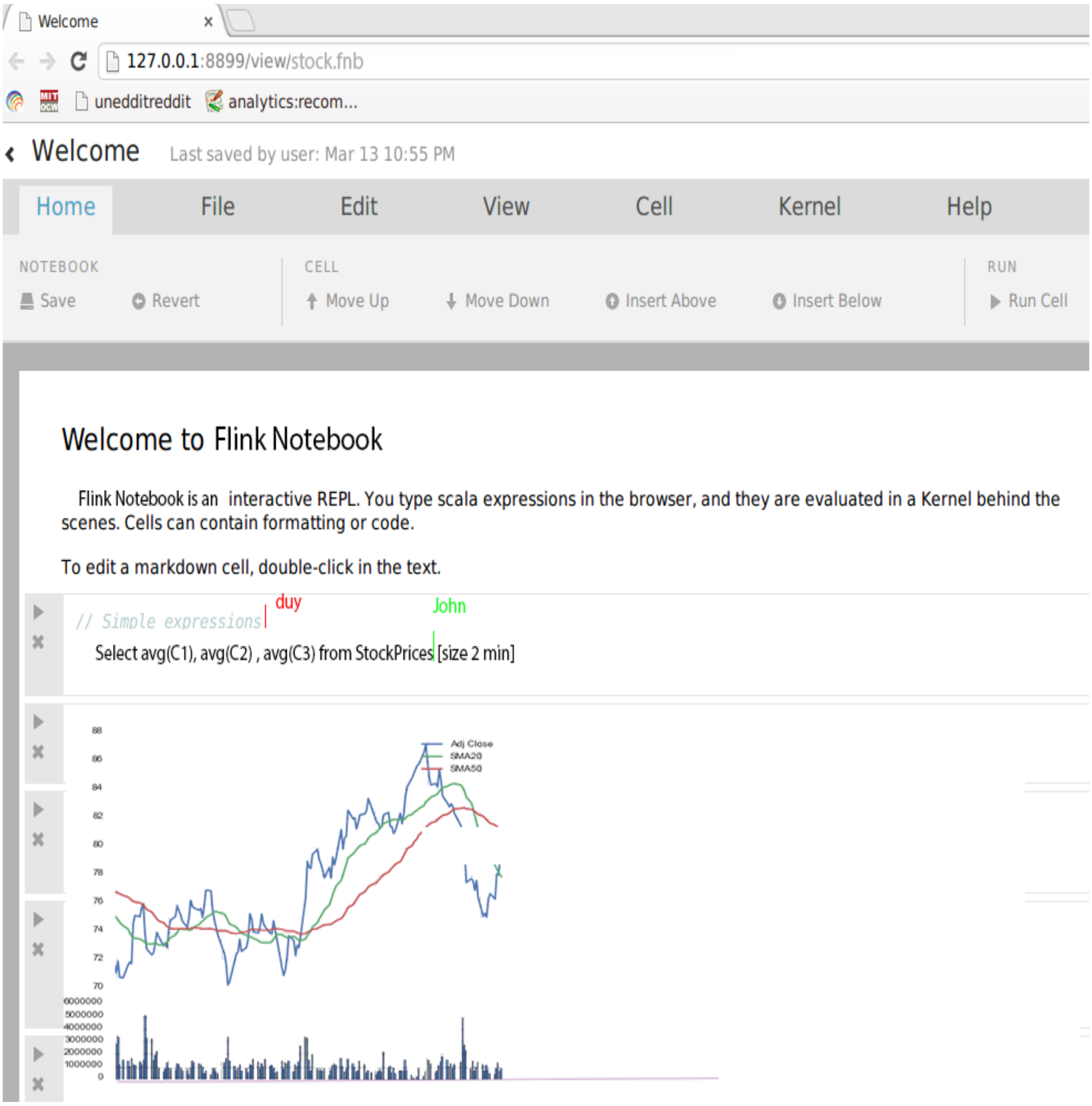
***Timing***:  First release (June 2015):  SQL syntax. Second release (August 2015): Web-based. Third release (October 2015): full collaborative features.

***Marketing***: Aligned with Apache Flink, we can propagate the idea Flink Notebook through various Big Data Conferences (Strata + Hadoop , Apache Conference), invited talks & tutorials at *Meetup* group. As a business model we also give 3 months of free support for the first 30 customers.

***HERE WE ARE !***

Flink Notebook brings Big Data developers a friendly, collaborative and interactive web-based environment to boost up your productivity without scarifying superior power from Flink Streaming platform.

# APPENDIX B: HIGH-FIDELITY PROTOTYPE



## Welcome to Flink Notebook

Flink Notebook is an interactive REPL. You type scala expressions in the browser, and they are evaluated in a Kernel behind the scenes. Cells can contain formatting or code.

To edit a markdown cell, double-click in the text.

```
// Simple expressions
  Select avg(C1), avg(C2) , avg(C3) from StockPrices [size 2 min]
```

**Features**:

- SQL syntax understanding.
- Real-time diagrams and charts for output observation.
- To save , share and load *stock.fnb* at ease even if no Flink engine is running.
- Two users *duy* and *john* collaborate on the same working session.

**Credits**:

The design is inspired by Google Docs, iPython Notebook, and Scala Notebook.