Enhancing Video Gaming Experience using Big Data Analytics

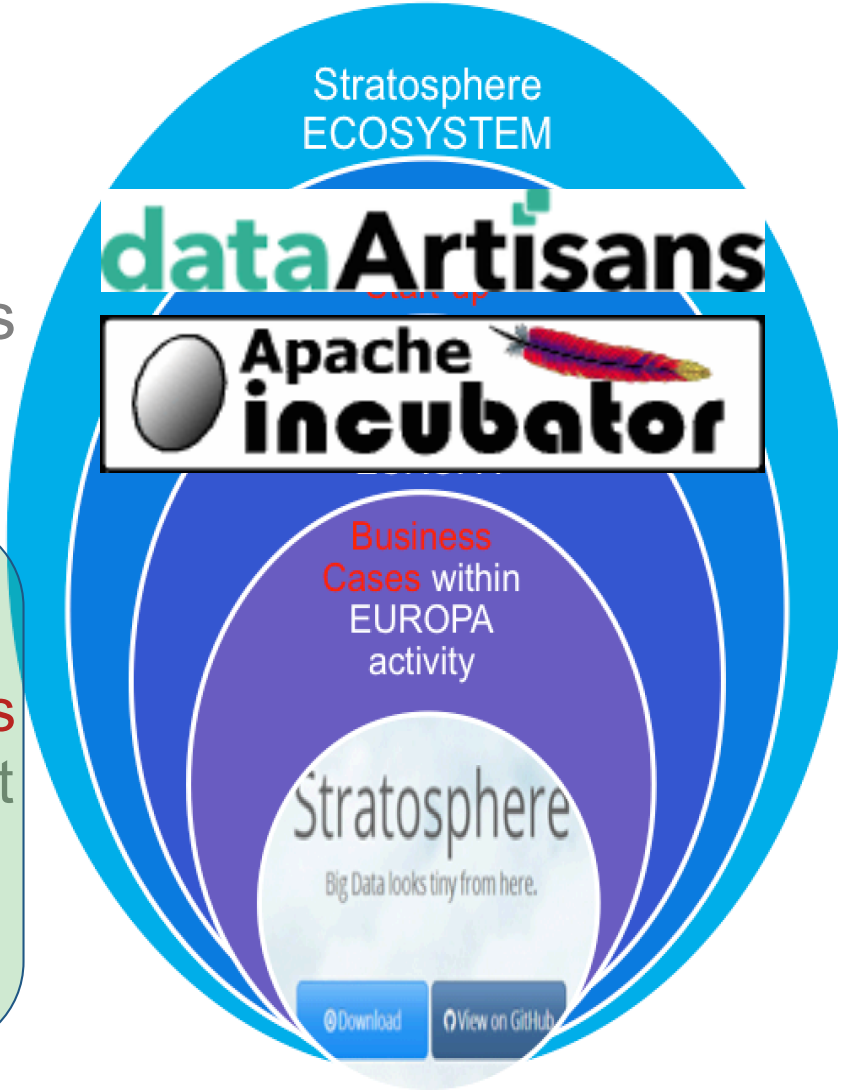
Camelia Ciolac, Alexandru Costan, Gabriel Antoniu
KerData Team, IRISA / Inria Rennes – Bretagne Atlantique

EIT Future Cloud Sympsium
Rennes, 19 October 2015

The context: building an ecosystem around Flink (formerly Stratosphere)

- Future Cloud Action Line, Europa Activity
- Stratosphere start-up operational
- rts

- nts
  ent
  few of them
  ✓ F-Secure, Thales, Transmetrics, **Tribeflame**



This talk

- The Big Data Analytics Platform: Apache Flink
- The Use Case: a Mobile Gaming Company
- Data Analytics on Video Games Logs
- The Findings

The Big Data Analytics Platform: Apache Flink

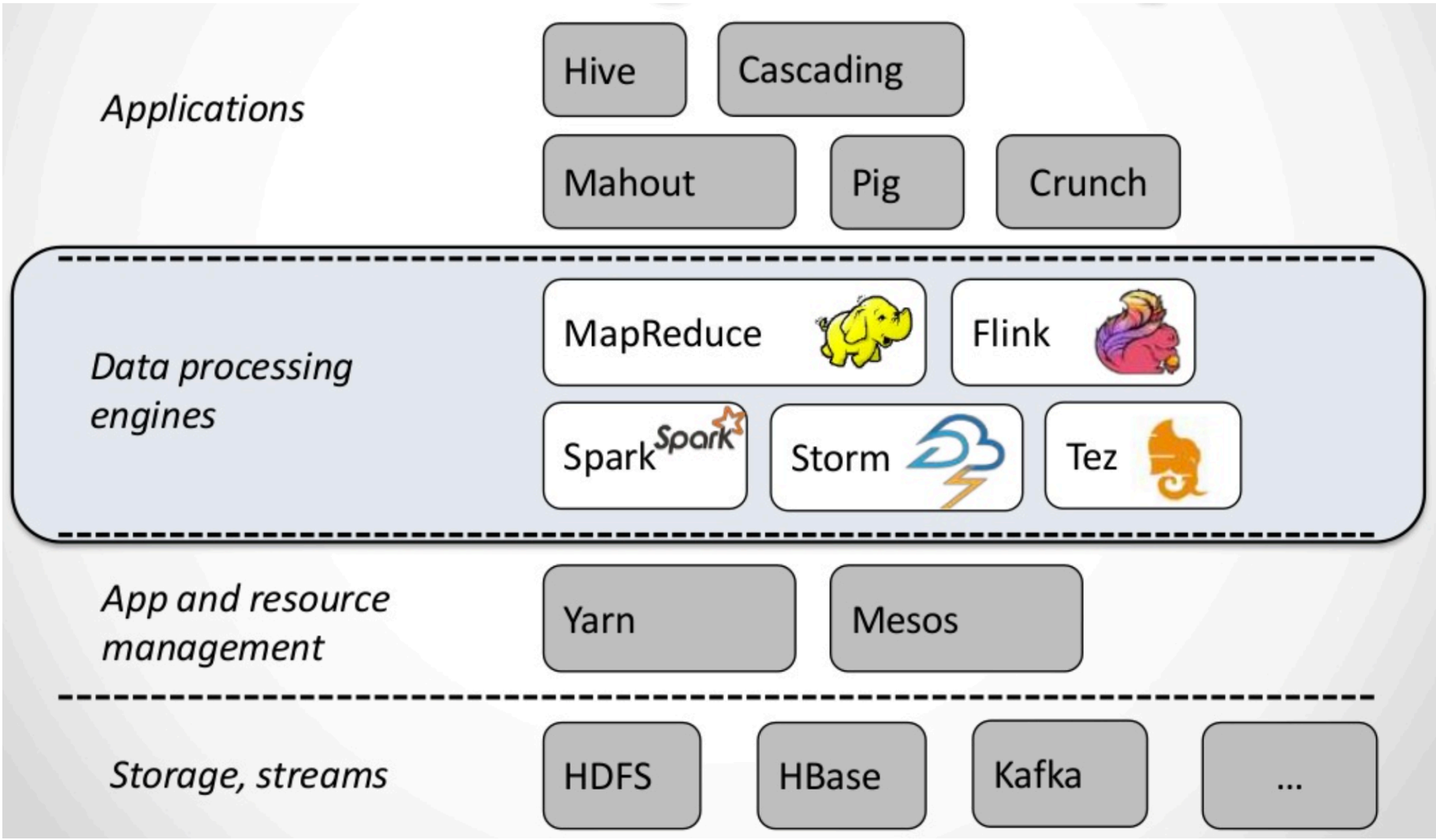
Inria

What is Apache Flink?

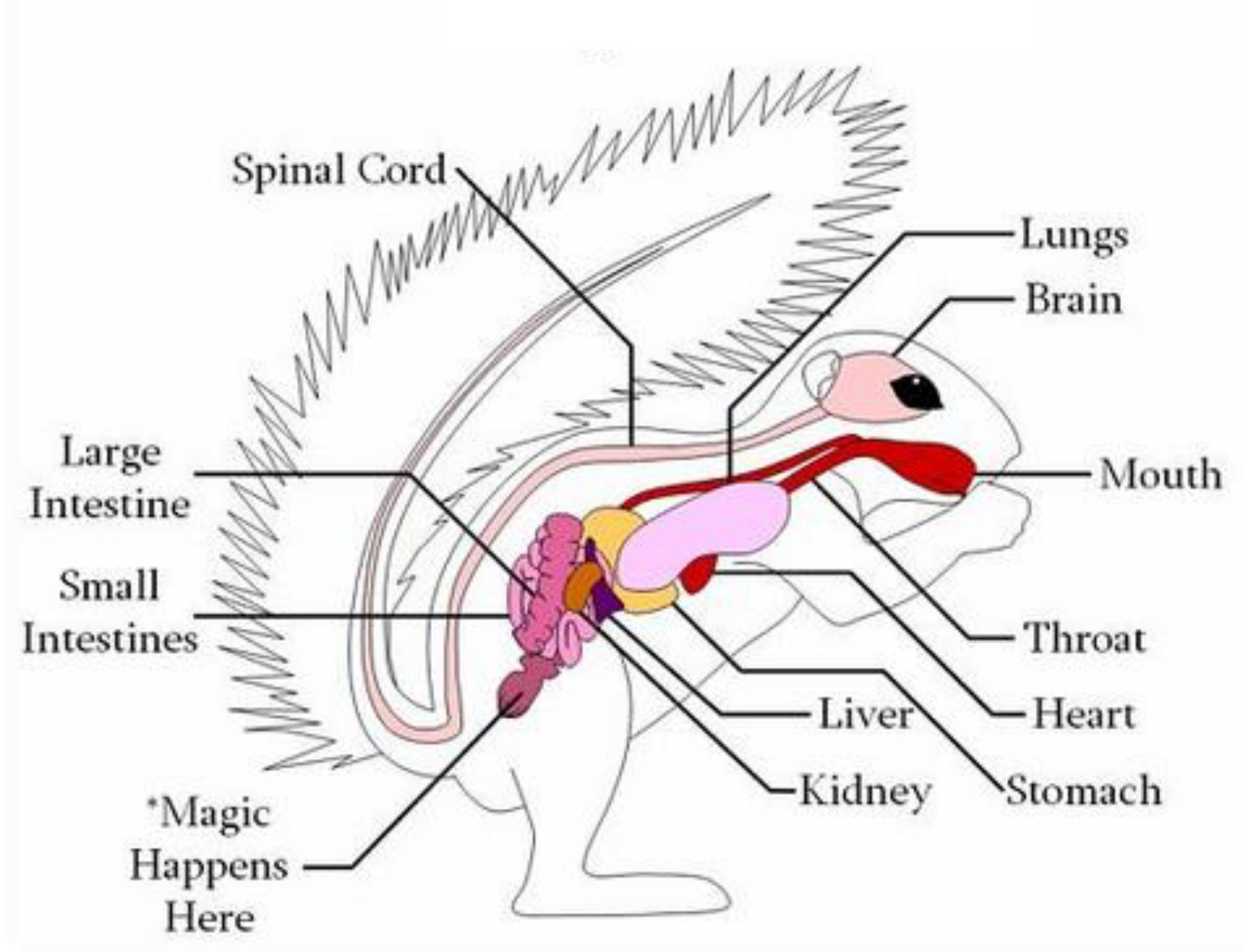
- Came out of the Stratosphere research project, started at TU Berlin in 2009
- Open source platform for **in-memory,** distributed **stream** and **batch** data processing
- Growing community
  - >100 contributors
  - Industry users:

- Rich set of operators:
  - Join, Union, Cross, Iterate, Filter, Map, FlatMap, Reduce, ReduceGroup etc.

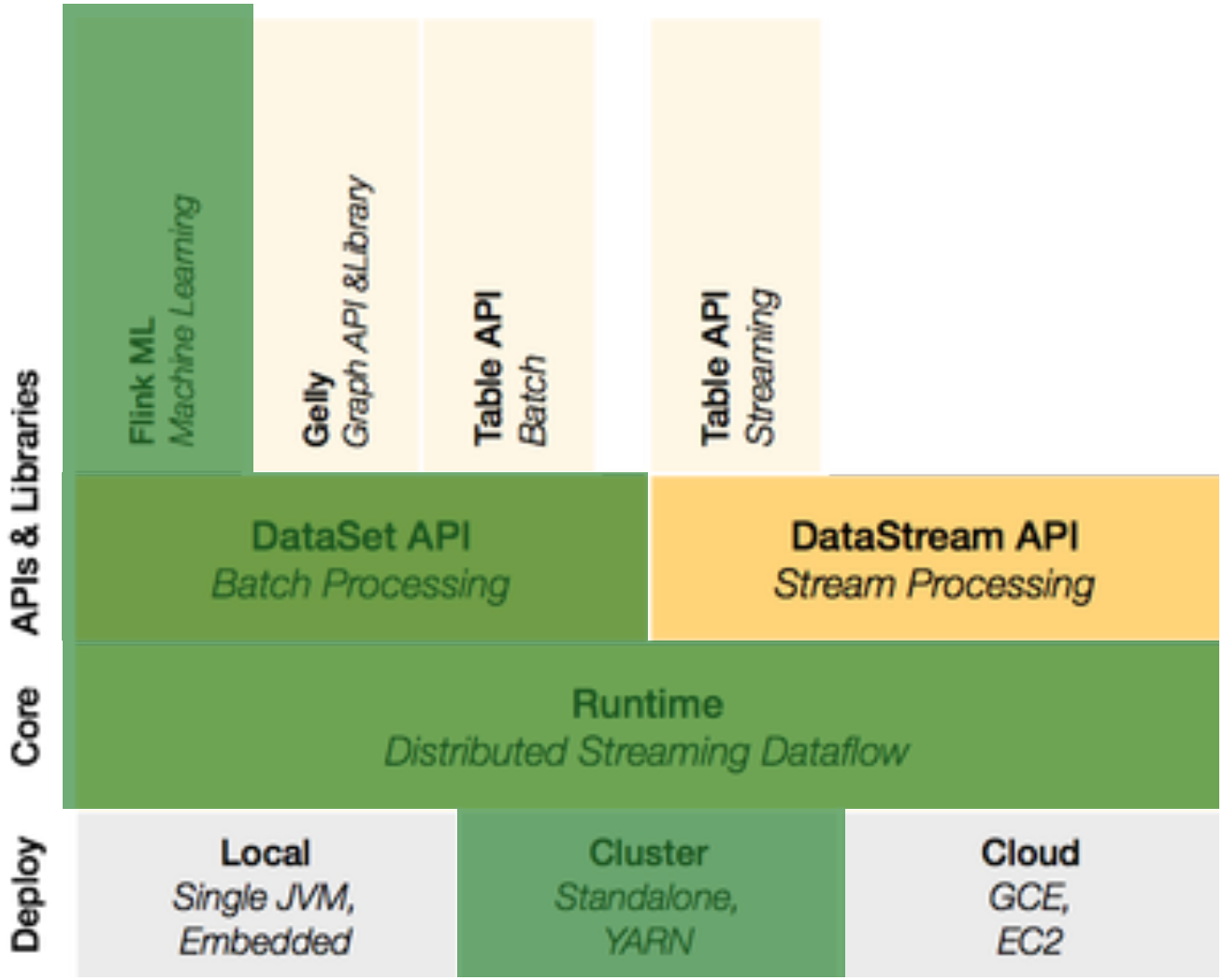
Open Source data processing landscape



Dissecting Flink



Apache Flink stack



Apache Flink features

- **Automatic Optimization:** selects an execution plan for a Common API program
  - Like an AI system manipulating your program for you ☺
- **Native Iterations:** embedded in the API
  - Delta iterations speed up many programs by orders of magnitude
- **Streaming dataflow engine:**
  - True real-time processing
  - Data distribution, communication, and fault tolerance for distributed computations over data streams.

The Use Case

The Company: 

- Finnish start-up launched in 2009
- One of the first tablet games companies in the world
- > 12 games
- Small team: 9 people
  - **No Data Scientist!**

The Game: Benji Bananas

- Fun physics based gameplay
- 100 levels
- 35M users
- 500K players daily
- Free to play, but has optional extras available for purchase.

Zoom on the collected data

- Logs store information on:
  - How users behave in the game
  - How often they play
  - Money spending
- Data format
  - JSON, per hour, compressed - structured key / value
- Size:
  - Average logs collected: 2 GB / day
  - Total logs size: 1.3 TB
- No private data (anonymized)
- Data analytics tools used so far:
  - Omniata, R, simple scripts (Python, bash)

Data Analytics challenges

- **Feedback** analysis
  - Was the game too easy? too boring?
  - How many people play level X and then how many play level X+1?
- **Monetization** strategies
  - At which point in time they are ready to buy?
- **Gaming experience** enhancement
  - Experienced bad performance (e.g. small frame rate, bugs)?

Our focus

- **Retention**
  - Differences between users playing and not playing the next day
  - How many come back in 1 day (now 30%), 7 days, 30 days?

Extracting Meaning from Video Game Logs

Inria

Current state of Machine Learning in Flink: FlinkML

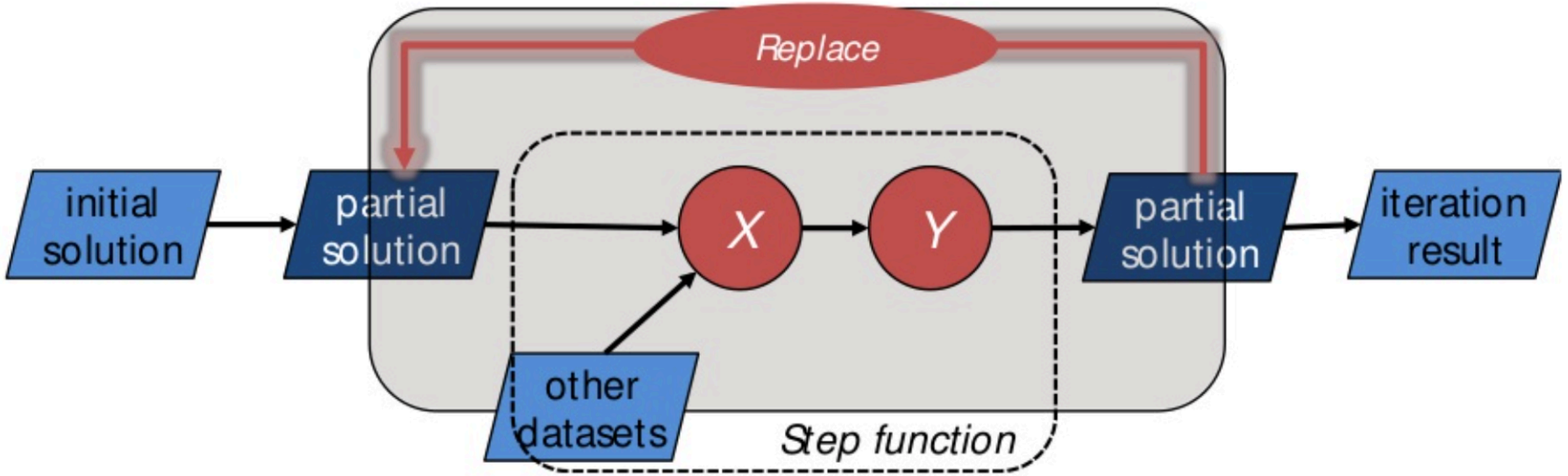
- Implemented algorithms:
  - **Supervised Learning:** SVM using communication efficient distributed dual coordinate ascent (CoCoA), multiple linear regression
  - **Data Preprocessing:** polynomial features, standard scaler, MinMax scaler
  - **Recommendation:** alternating least squares
  - **Utilities:** distance metrics

- Pipelining support
  - inspired by *scikit-learn*
  - quickly build complex data analysis pipelines

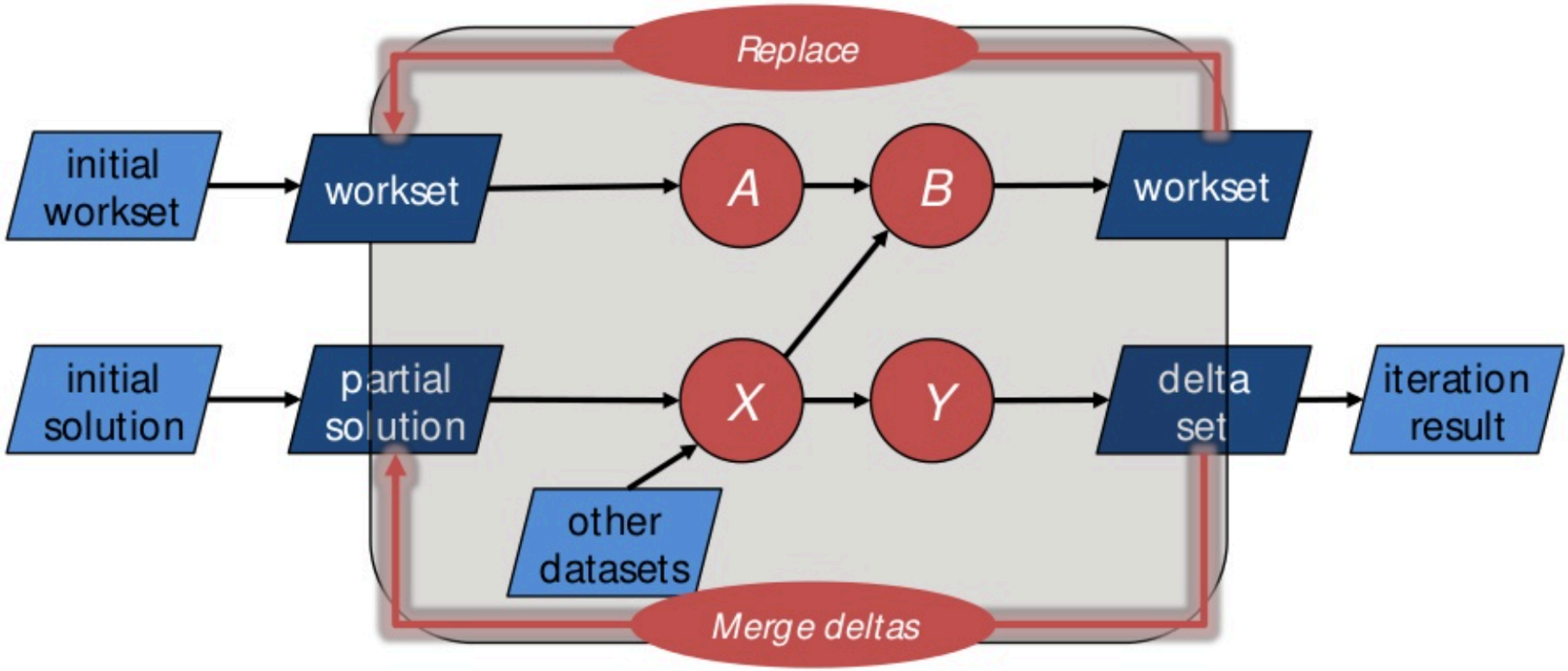
Why is Flink a good fit for ML ?

- **Streaming ML**
  - Log analysis, spam detection, credit card fraud detection, recommendation – in real-time
  - Patterns might change over time
  - Retraining of model necessary
  - Best solution: online models
- **Pipelining**
  - Avoid materialization of large intermediate state
- **Stateful iterations**
  - Keep state across iterations
- **Delta iterations**
  - Limit computation to elements which matter

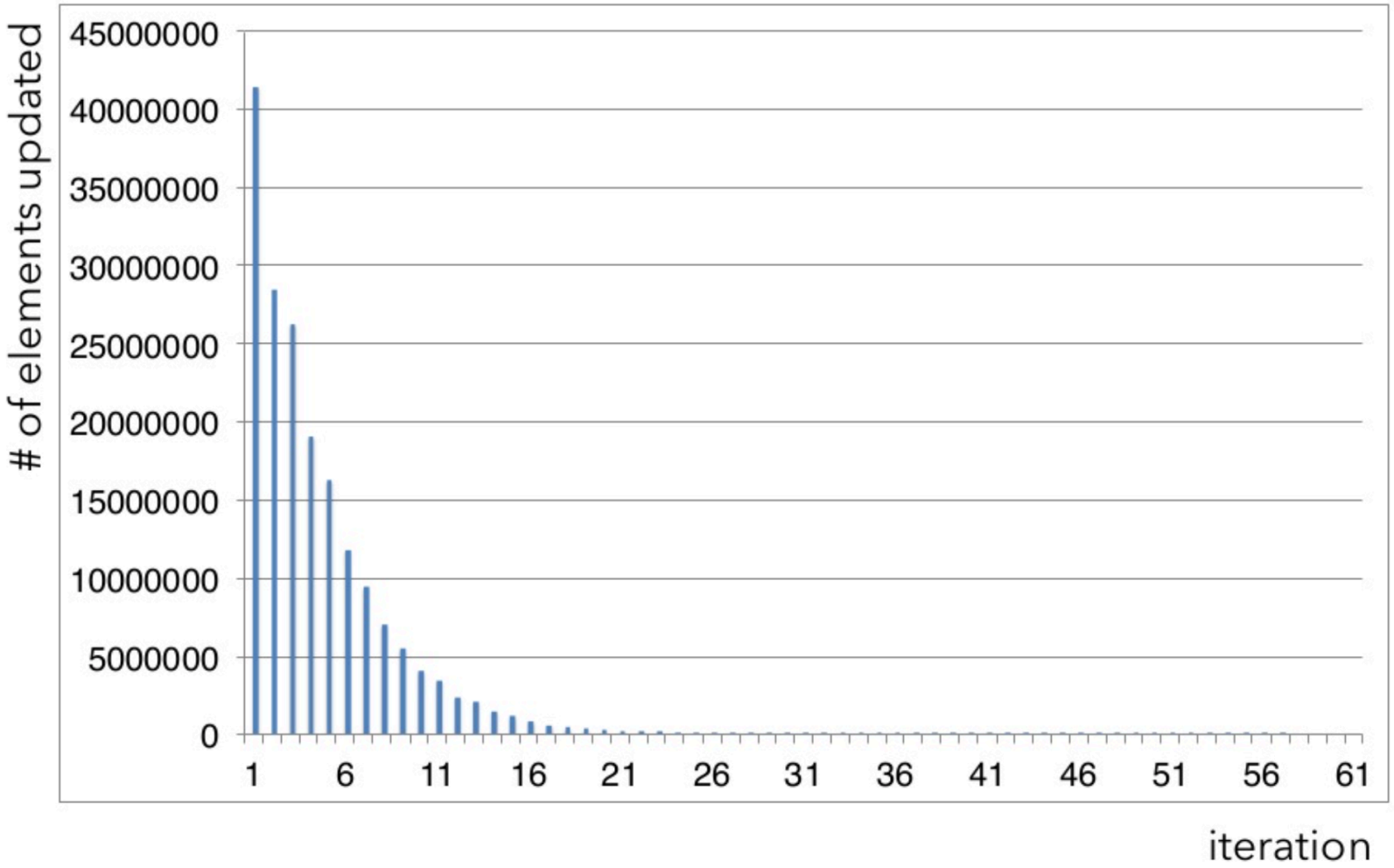
Bulk iterations



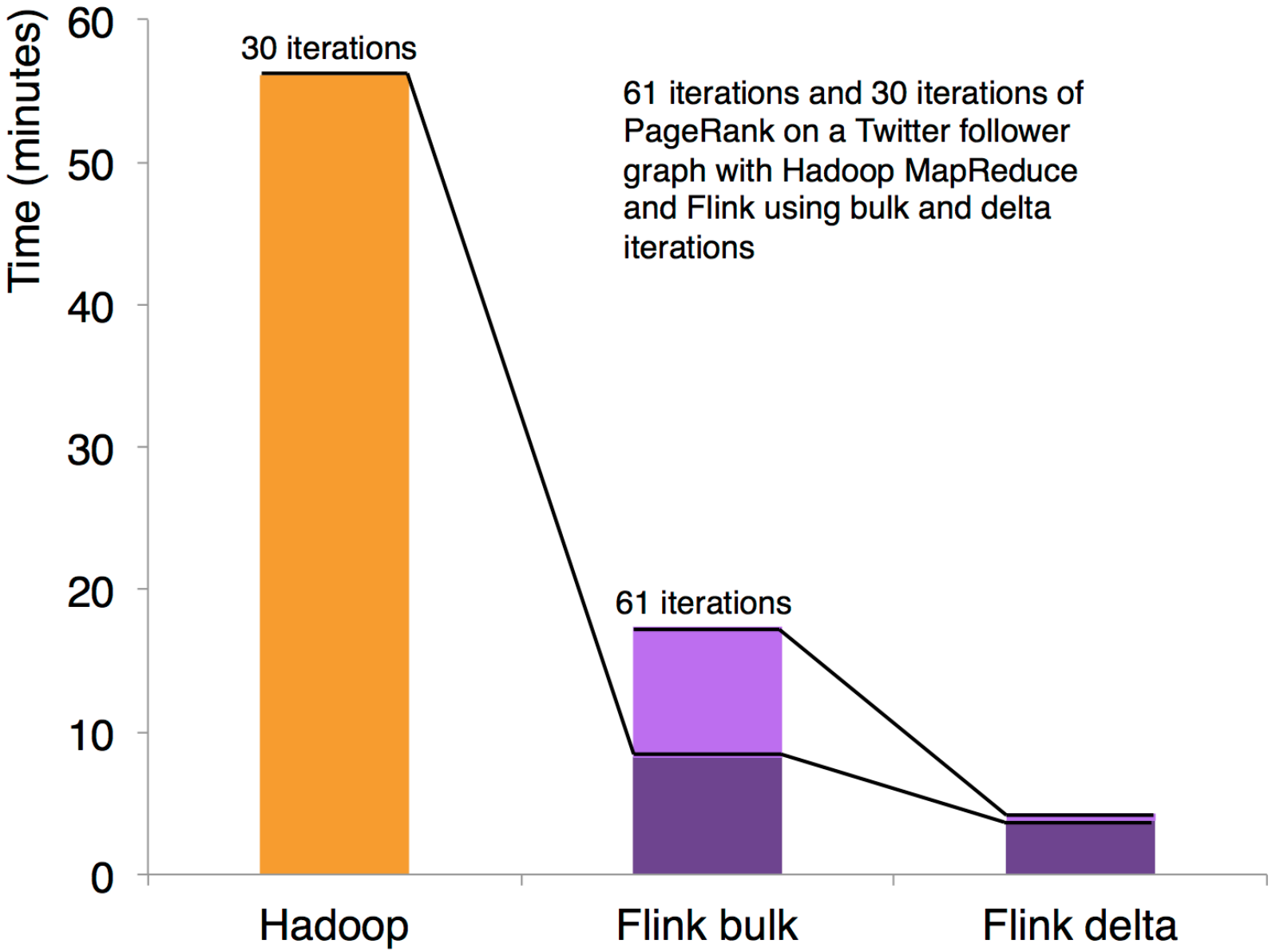
Delta iterations



Effect of delta iterations



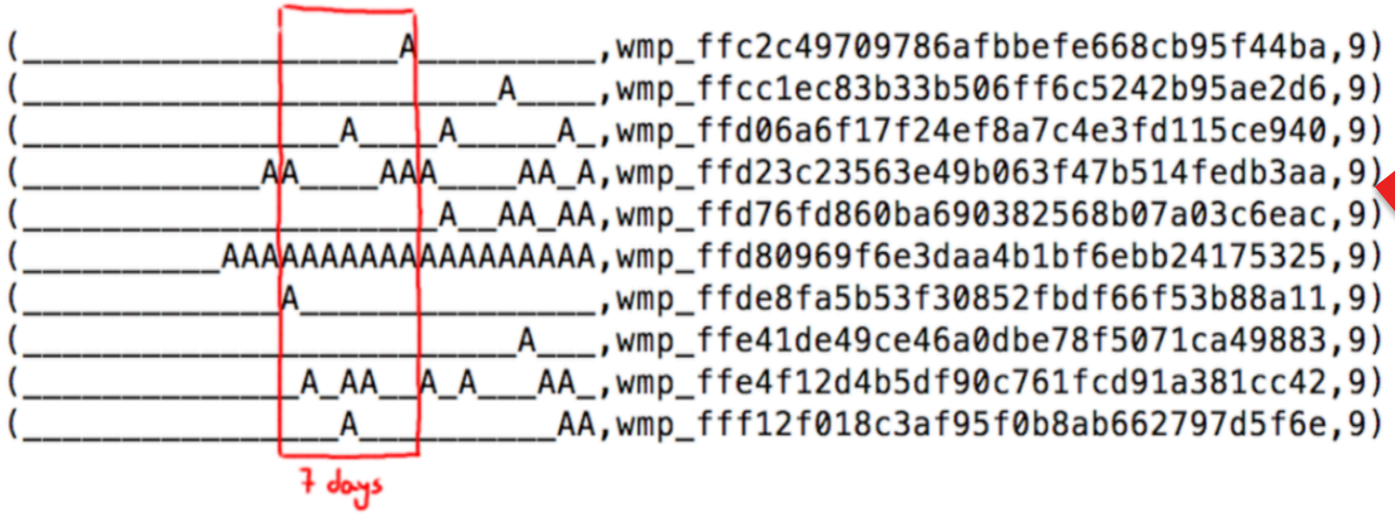
Iteration performance



Flink at work!

- Focus on **retention:**
  - ① *How likely is that a player is eager to play again within T days, after he registered a specific failure ratio on one day?*
  - ② *How probable is it for a player to return to play within T days, given his number of re-attempts of the level where he got stuck and the level's position in the game's saga?*
- Approach:
  - Logistic Regression
  - Flink features used: SQL operators, delta and bulk iterations, graph processing API

From event logs to a gameplay calendar

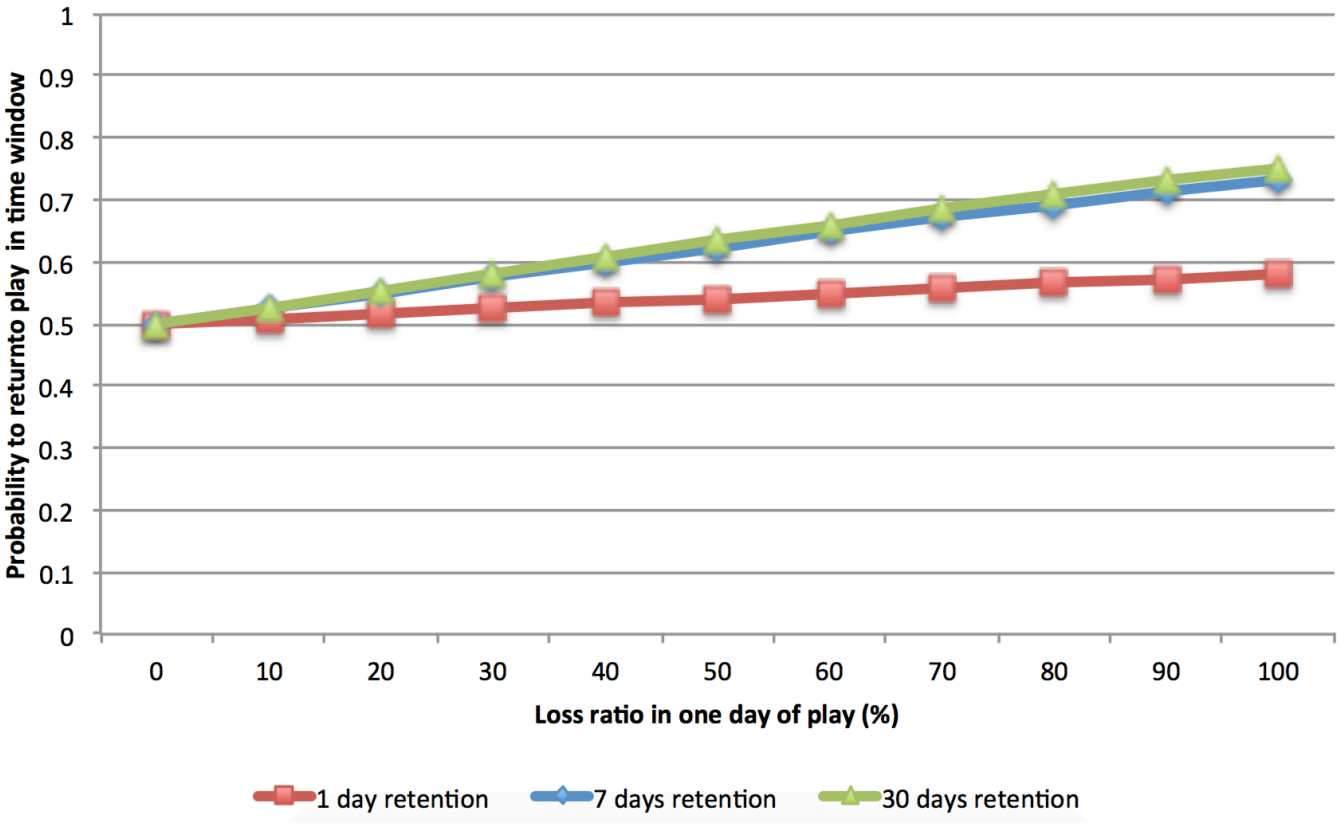


- Logistic regression in brief
  - a type of probabilistic statistical classification

(supervised machine learning = learning where a training set of correctly identified observations is available)

Our Findings

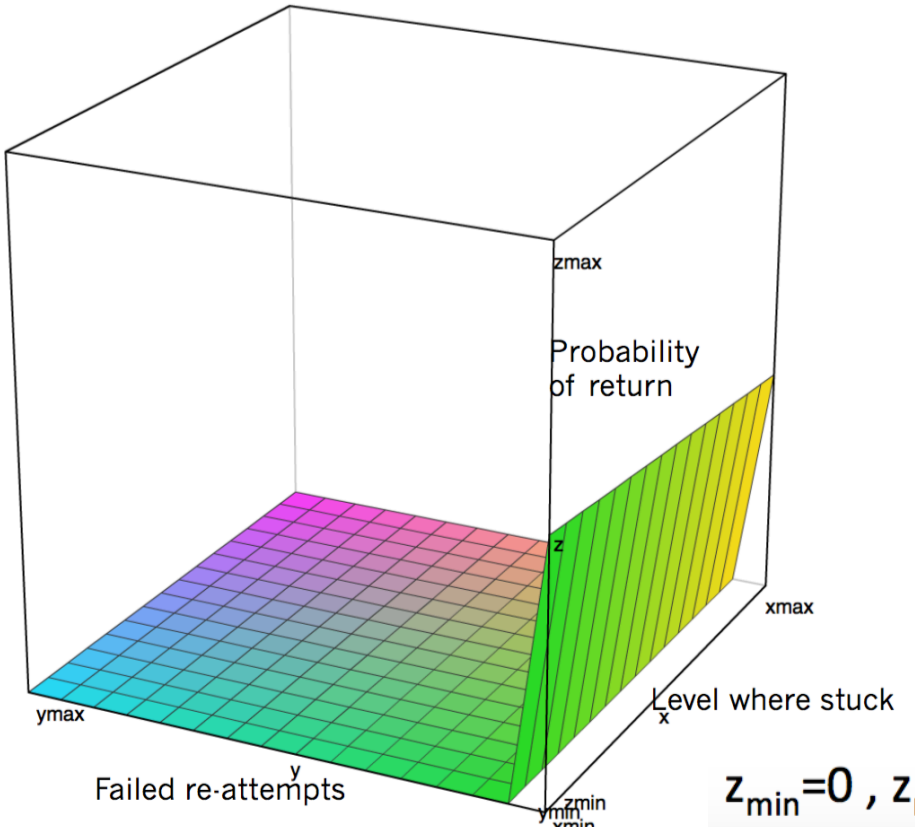
Retention (based on failure ratio)



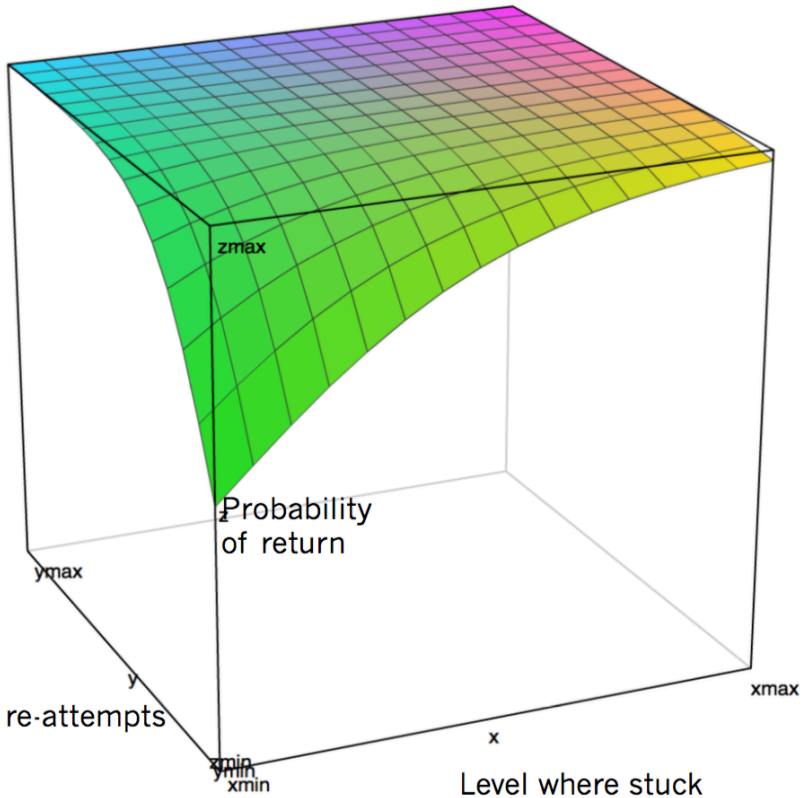
Loss ratio	Prob. to return in 1 day	Prob. to return in 7 days	Prob. to return in 30 days
20%	0.51	0.55	0.55
80%	0.56	0.69	0.70

Retention (based on re-attempts and level)

1 day



7 days

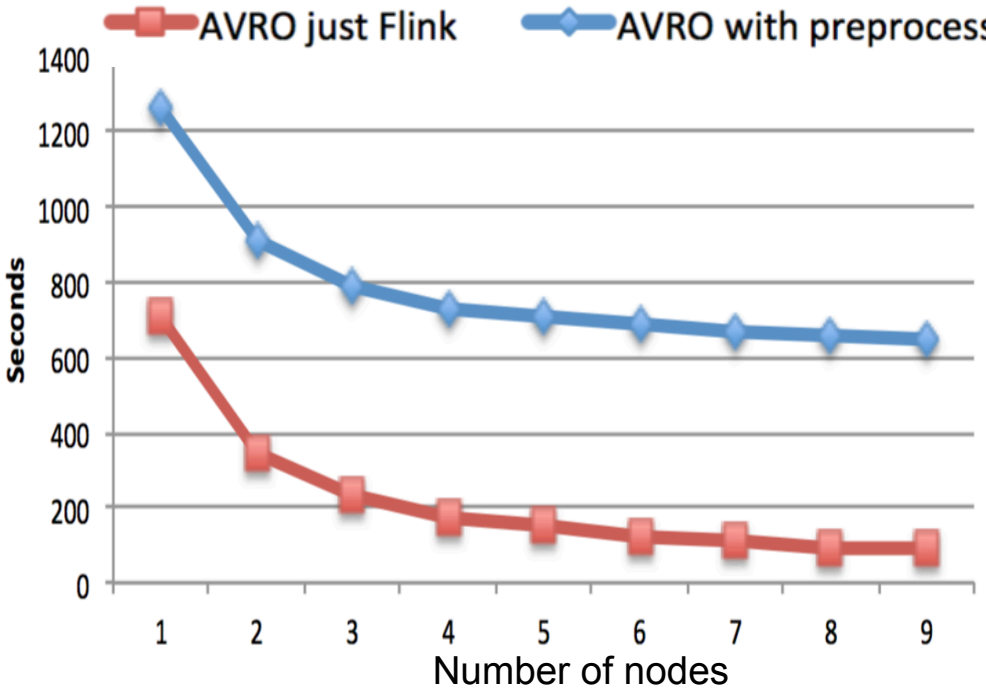


$z_{min}=0$, $z_{max}=1$

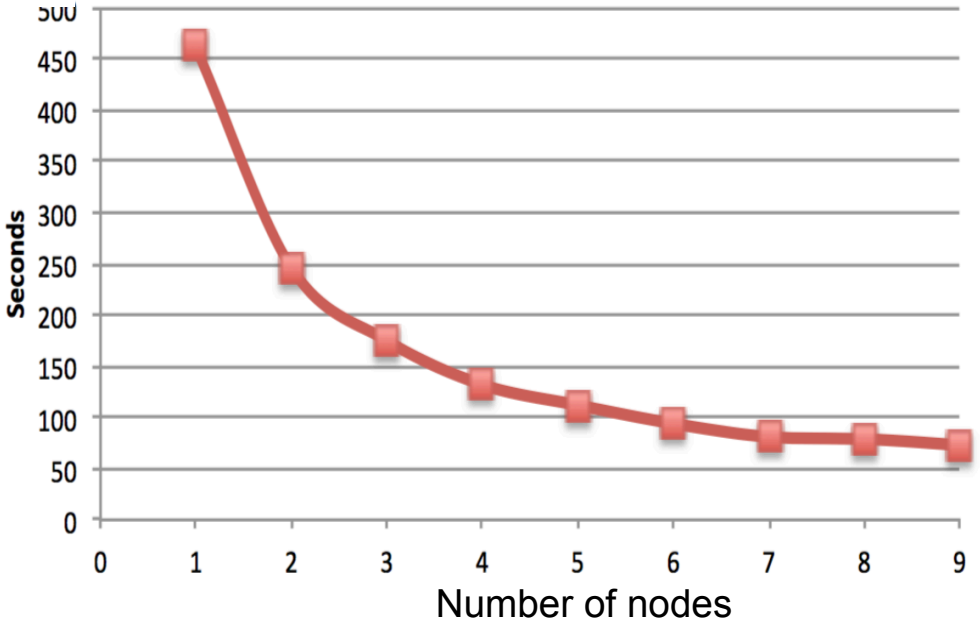
Level where stuck	Failed Re-attempts of the level	Probab. to return in 1 day	Probab. to return in 7 day
5	3	0.0335692	0.6049181
5	10	0.0000142	0.7241218
50	3	0.0294548	0.8985303
50	10	0.0000124	0.9381965

Performance analysis

- Flink cluster with 10 nodes deployed on Grid'5000
- Average pre-processing time per day of data: 10 minutes



Building Game Play Calendar from AVRO



Logistic Regression (7 day window, 1000 iterations)

To take away

- Tribeflame adopted Flink for their processing needs
- Flink allows easy taming of large datasets
- Perfectly suited for the needs of small and very small organizations:
  - runtimes for single machine, clusters or clouds
- Flink offers the only hybrid (streaming + batch) data processing
- More maturity and more adoption still needed:
  - maybe from use-cases with real-time stream processing and fast iterative processing
- Support for more ML algorithms needed

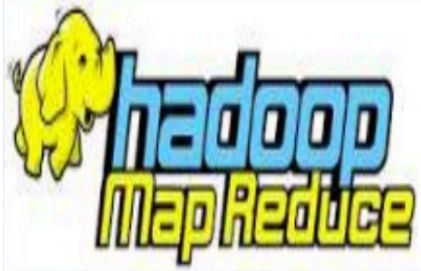
Thank you!

alexandru.costan@irisa.fr
http://team.inria.fr/kerdata/

Flink vs. Storm

- Higher level and easier to use API
- More light-weight fault tolerance strategy
- Exactly-once processing guarantees
- Stateful operators
- Flink does also support batch processing
- “Twitter Heron: Stream Processing at Scale” by Twitter or “Why Storm Sucks by Twitter themselves”!! [SIGMOD 2015]

Why Flink is the Next-Gen?

			
• Batch	• Batch • Interactive	• Batch • Interactive • Near-Real time Streaming • Iterative processing	• Batch • Interactive • **Real-Time Streaming** • **Native Iterative** processing
MapReduce	**D**irect **A**cyclic **G**raphs (DAG) Dataflows	RDD: **R**esilient **D**istributed **D**atasets	**Cyclic Dataflows**
1st Generation (1G)	2ndGeneration (2G)	3rd Generation (3G)	4th Generation (**4G**)

Flink vs. Spark

- Spark
  - Based on RDDs: allow to leverage functional programming
  - Streaming engine wraps data into mini-batches: **near real-time**

- Flink
  - True low-latency streaming engine (not micro- batches!) that unifies batch and streaming in a single Big Data processing framework
    - *"I would consider stream data analysis to be a major unique selling proposition for Flink. Due to its pipelined architecture Flink is a perfect match for big data stream processing in the Apache stack."* – Volker Markl (Flink creator)
    - Flink uses streams for all workloads: streaming, SQL, batch.

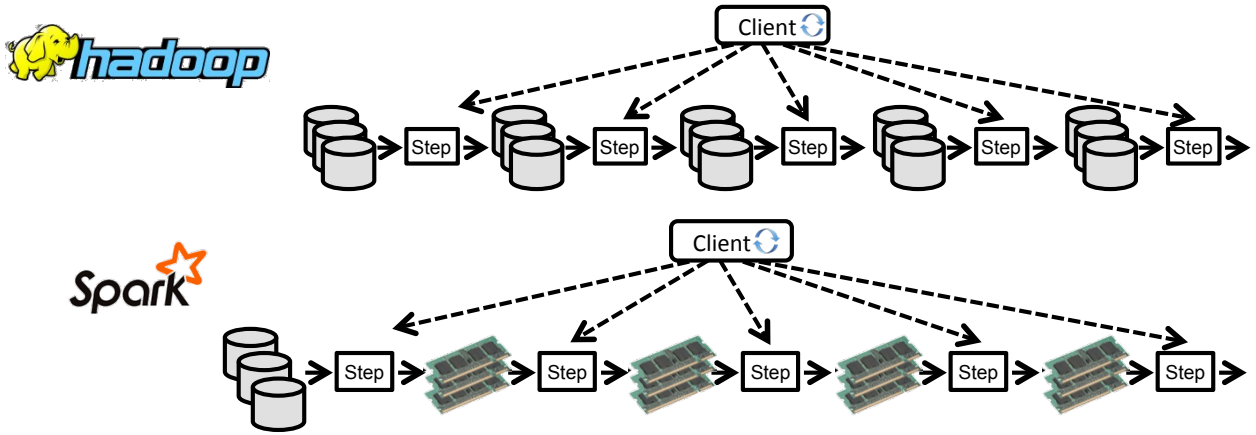
Flink vs. Spark

- Flink
  - Native closed-loop iteration operators
  - Custom memory manager
    - No more frequent Out Of Memory errors!
    - 332 Posts on OOM errors on Apache Spark Users List as vs. 7 Apache Flink Users List as of June 2015
  - Automatic cost based optimizer
    - little re configuration and little maintenance when the cluster characteristics change and the data evolves over time)
    - optimization of join algorithms, operator chaining and reusing of partitioning and sorting

Flink vs. Spark

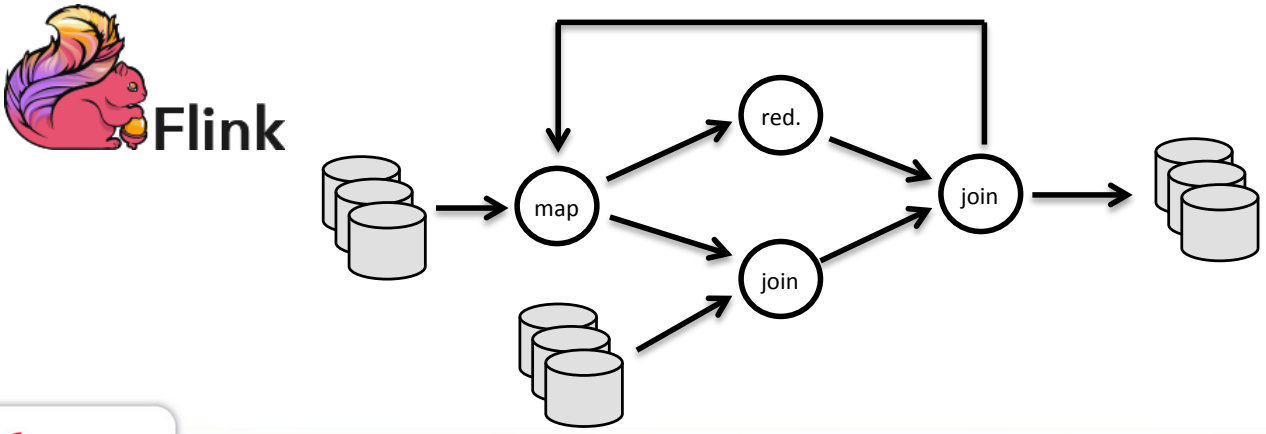
- Benchmark between Spark and Flink:
  - **Flink outperforms Spark** in the processing of machine learning & graph algorithms and also relational queries
  - **Spark outperforms Flink** in batch processing
  - `http://goo.gl/WocQci`

Built-in vs. driver-based looping



Loop outside the system, in driver program

Iterative program looks like many independent jobs



Dataflows with feedback edges

System is iteration-aware, can optimize the job

Does Machine Learning works like that?

More realistic scenario!