Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Real-Time Alarm Verification with Spark Streaming and Machine Learning

Ana Sima, Jan Stampfli, Kurt Stockinger
Zurich University of Applied Sciences

Swiss Big Data User Group Meeting
January 23, 2017

About Me

Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

InIT Institut für angewandte Informationstechnologie

- Prof. Dr. Kurt Stockinger
- Since summer 2013 at ZHAW
  - Databases, Data Warehousing, Big Data
- 2007-2013:
  - Data Warehouse &
    Business Intelligence Architect
- 2004-2007:
  - Computer Scientist
- 1999-2003:
  - Computer Scientist

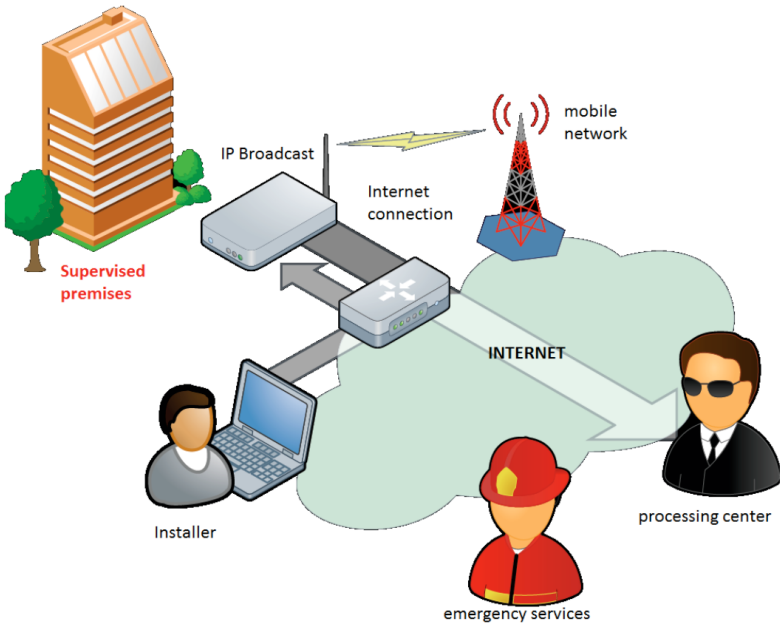
CREDIT SUISSE

BERKELEY LAB

CERN

What is the Problem with Alarm Systems?

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie



Nach Polizeiangaben sind **über 90%** der Alarmmeldungen, die bei der Polizei eintreffen, **Fehlalarme**. Fehlalarme lösen Interventionskosten von einigen hundert Franken aus, die vom Alarmanlagenbesitzer zu bezahlen sind. Dies und die Angst vor Fehlalarmauslösungen, dürften die Gründe sein, dass eine grosse Anzahl, man spricht von bis zu 60%, der Alarmanlagen auch bei Abwesenheit der Bewohner nicht eingeschaltet sind.

Quelle: http://www.quadragard.ch/wissen/einbruchschutz-1/ 18.10.2015

A Typical Example

Serious universities…

...with Serious Students

A Serious Party in a Student Home

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

… Serious Fire Fighters

Why Machine Learning?

- Around 90% of reported incidents are false alarms
- Human operators should prioritize true alarms
- Machine learning is able to
  - discover the patterns buried in the alarm data
  - separate true from false alarms with a high accuracy

DIGITAL FOOTPRINT OF A SECURED OBJECT

- Alarm panel
- Transmitter
- User behavior
- Alarm types

User behavior will improve overall process
User based business models are possible with machine learning tools
e.g. Services payments only when an alarm panel is activated

Research Project with Sitasys

ABOUT US

Welcome to Sitasys. We are a Swiss ISO 9001 certified company, specialized in secure and reliable alarm transmission, communication, and identification.

Formed as a spin-off of Ascom Security & Communication in 2012, Sitasys has over 30 years of experience in the field of alarm transmission and secure communication. Sitasys stands for **S**ecured **I**nformation **T**ransmission **A**larming **Sys**tem.

Our in-house product design and development department enables us to provide Swiss quality at its best.

Our strengths lie in a thorough understanding of your needs in the security sector, our extensive technology know-how, innovation through our highly experienced solution and development engineers, as well as our size that facilitates fast solutions and allows for a high flexibility.

Our vision

Sitasys is the preferred provider of secure and intelligent IP based connections of peripheral devices in unreliable networks.

Talk Outline

- Machine learning for alarm verification

- End-to-end performance analysis:
  - Stream processing (live analysis)
  - Batch processing (historic analysis)
  - Online machine learning (live verification)

Machine Learning with Spark

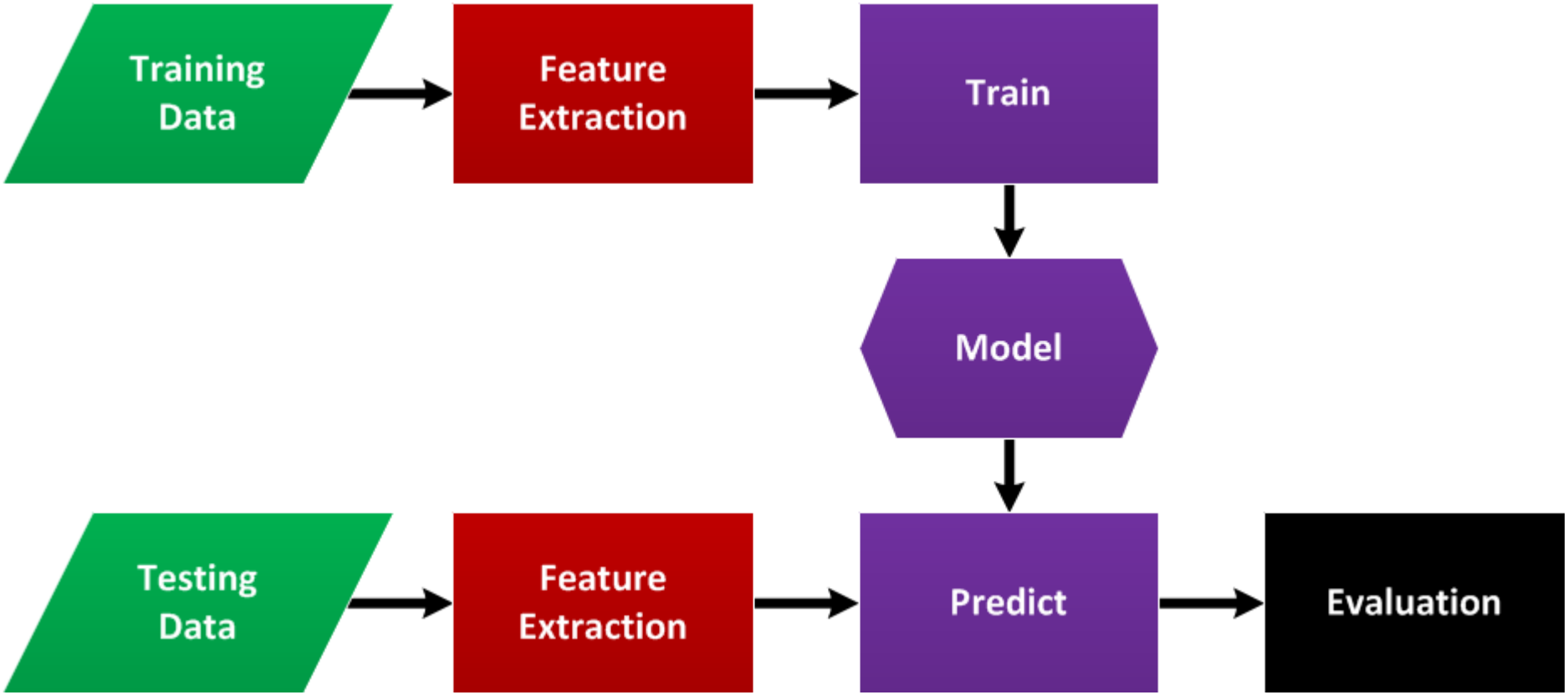
About Me

Jan Stampfli

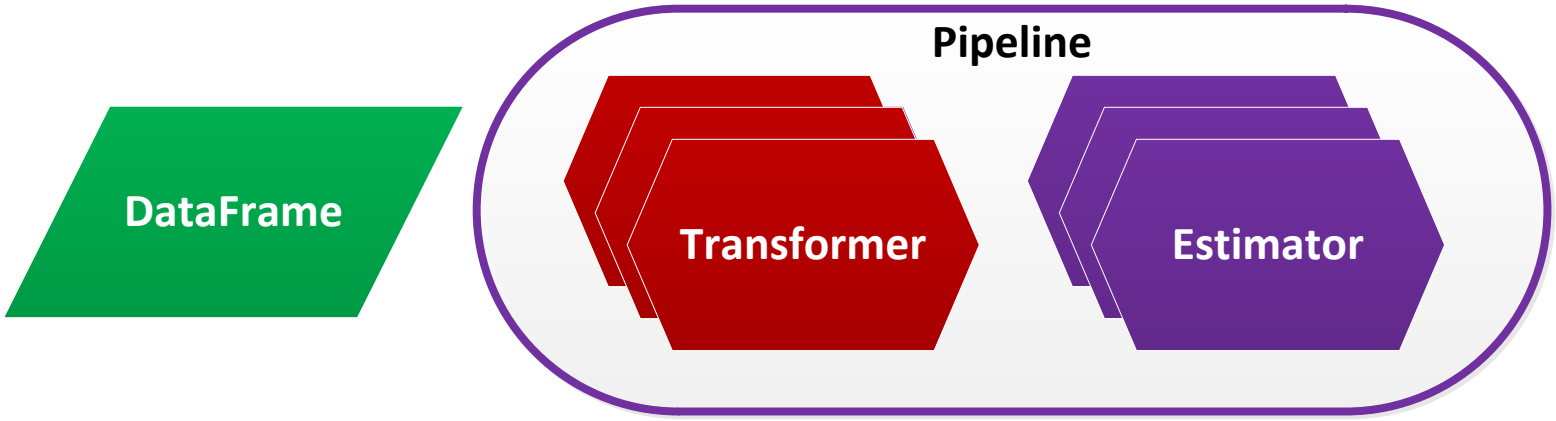
- ZHAW Bachelor in Computer Science
  - Graduated in 2014
- Research assistant at ZHAW
  - Since September 2014
- Working in research projects on the topics
  - Big Data
  - Machine Learning
  - Information Retrieval

A Typical Machine Learning Process

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie



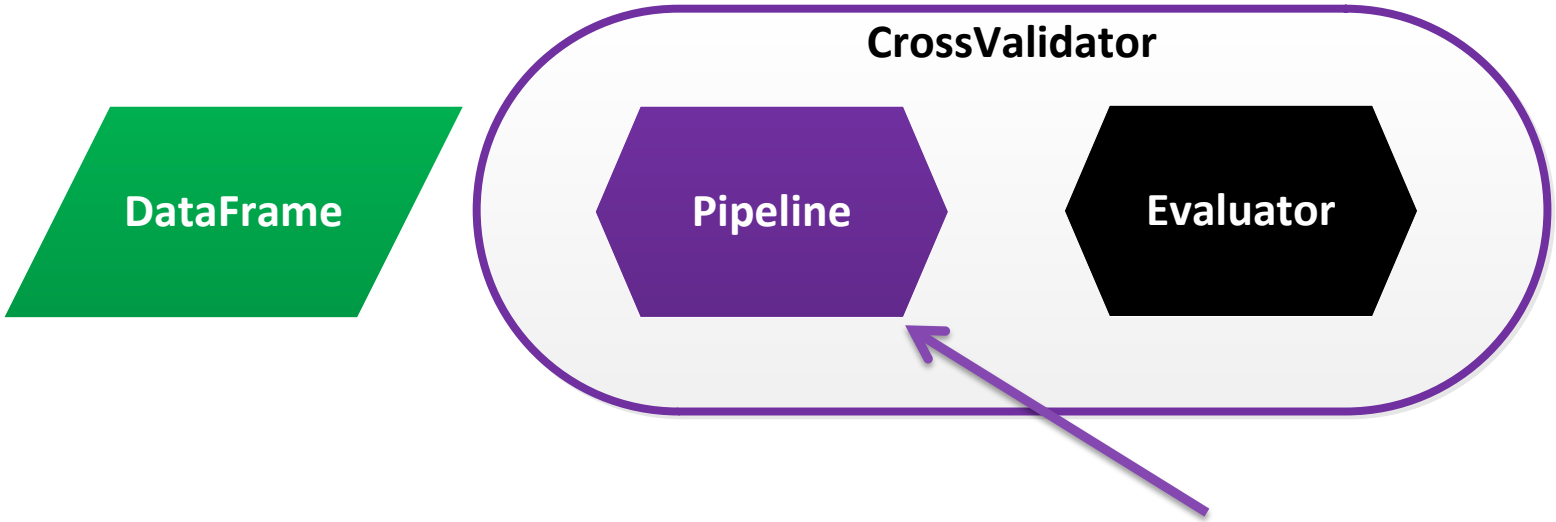
Spark ML Pipeline



location	label	hash	prediction	probabilities
Bern	1	2066911	0	[0.51, 0.49]
Chur	0	2099682	0	[0.99, 0.01]
Sion	1	2577109	1	[0.13, 0.87]

http://spark.apache.org/docs/latest/ml-pipeline.html

Spark ML Tuning

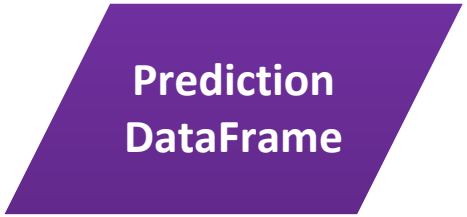


- Random Forest
  - Number of trees
  - Max depth of trees
- Parameter Map
  - [10, 20, 30]
  - [5, 25, 50]

http://spark.apache.org/docs/latest/ml-tuning.html

Spark ML Evaluation

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie



label	probabilities
1	[0.51, 0.49]
0	[0.99, 0.01]
1	[0.13, 0.87]

- Accuracy
- Precision
- Recall
- ...

http://spark.apache.org/docs/latest/mllib-evaluation-metrics.html

Applied Machine Learning

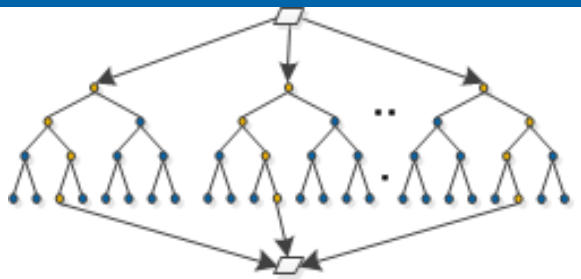
Data about Real Alarms

	Total	False alarms	True alarms
Total	339’841	165’997	173’844
Training	169’920	84’960	84’960
Test	169’921	81’037	88’884

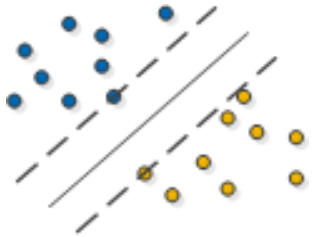
- Features
  - Location, sensor type, day of week, …
- Labels
  - 0 = false alarm
  - 1 = true alarm

Applied Algorithms

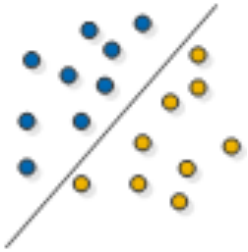
- Random Forest



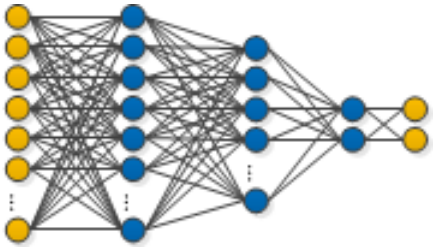
- Support Vector Machine



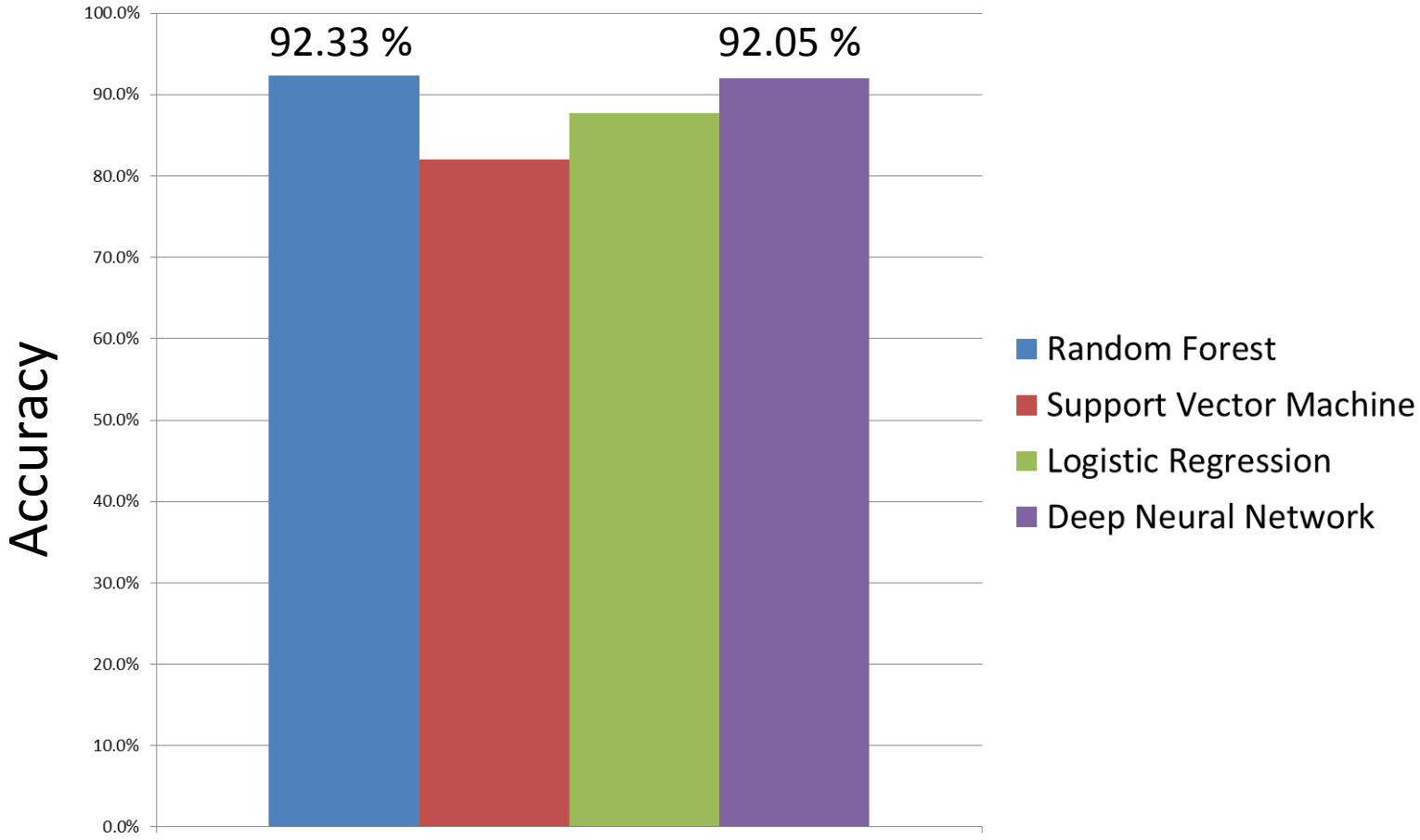
- Logistic Regression



- Deep Neural Network (DNN)



Results



Jan Stampfli, Kurt Stockinger (2016). Applied data Science:
Using Machine Learning for Alarm Verification, *ERCIM News, 107, 10*.

Evaluation – Random Forest

- Confusion matrix

	Prediction True	Prediction False
Label True	82’999	5’885
Label False	7’148	73’889

- Precision (correctly predicted alarms)

	Prediction True	Prediction False
Label True	92.07%	
Label False		92.62%

- Recall (correctly detected alarms)

	Prediction True	Prediction False
Label True	93.38%	
Label False		91.18%

Conclusion about ML

- Human operators are now able to prioritize alarms predicted as true

- False alarms must still be evaluated

Performance

About Me

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Ana Sima

- EPFL Master in SW Systems, 2015
- Previously worked on a framework for unified stream & batch processing (Cyclone)
- Joined ZHAW since November 2016

Welcome back from the holidays!

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

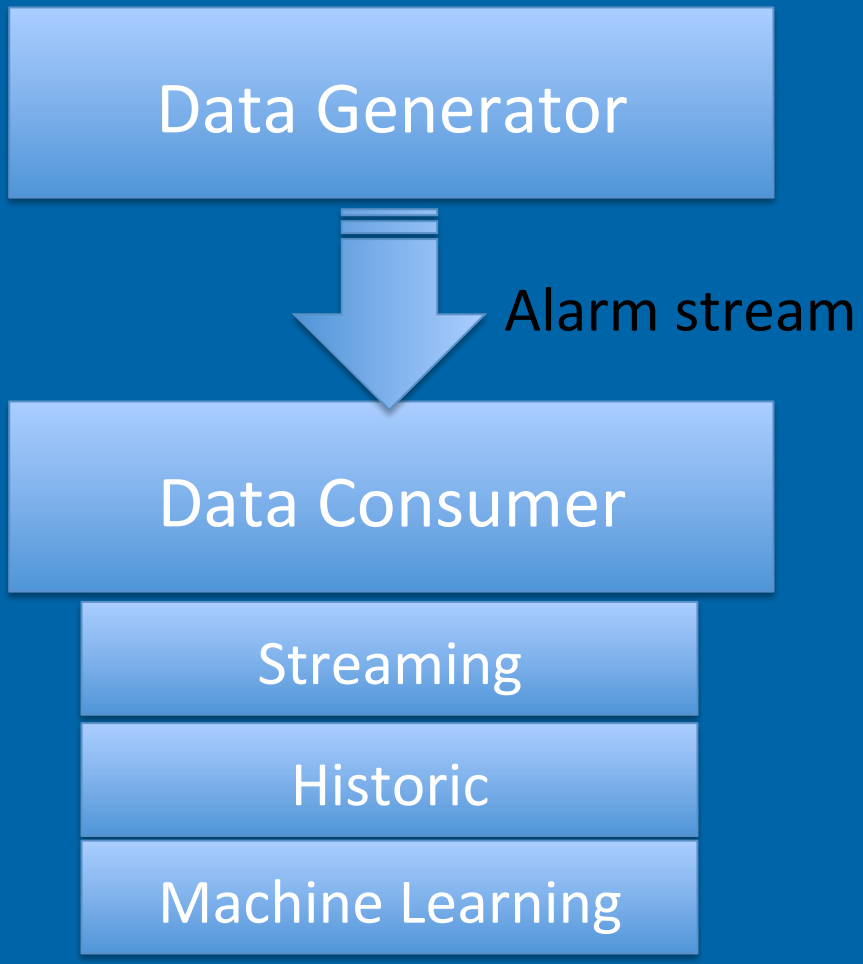
School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Why is alarm processing a time-critical mission?

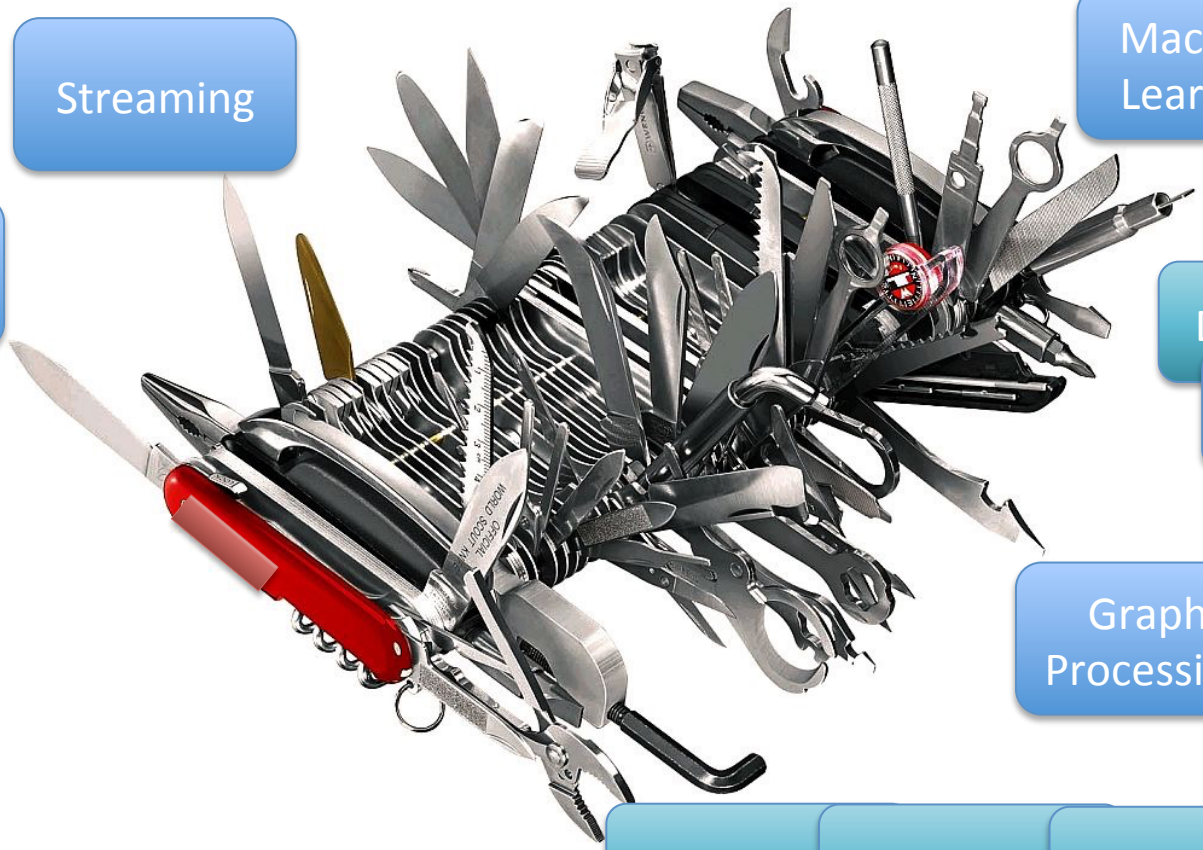
When things go wrong…

Performance testing



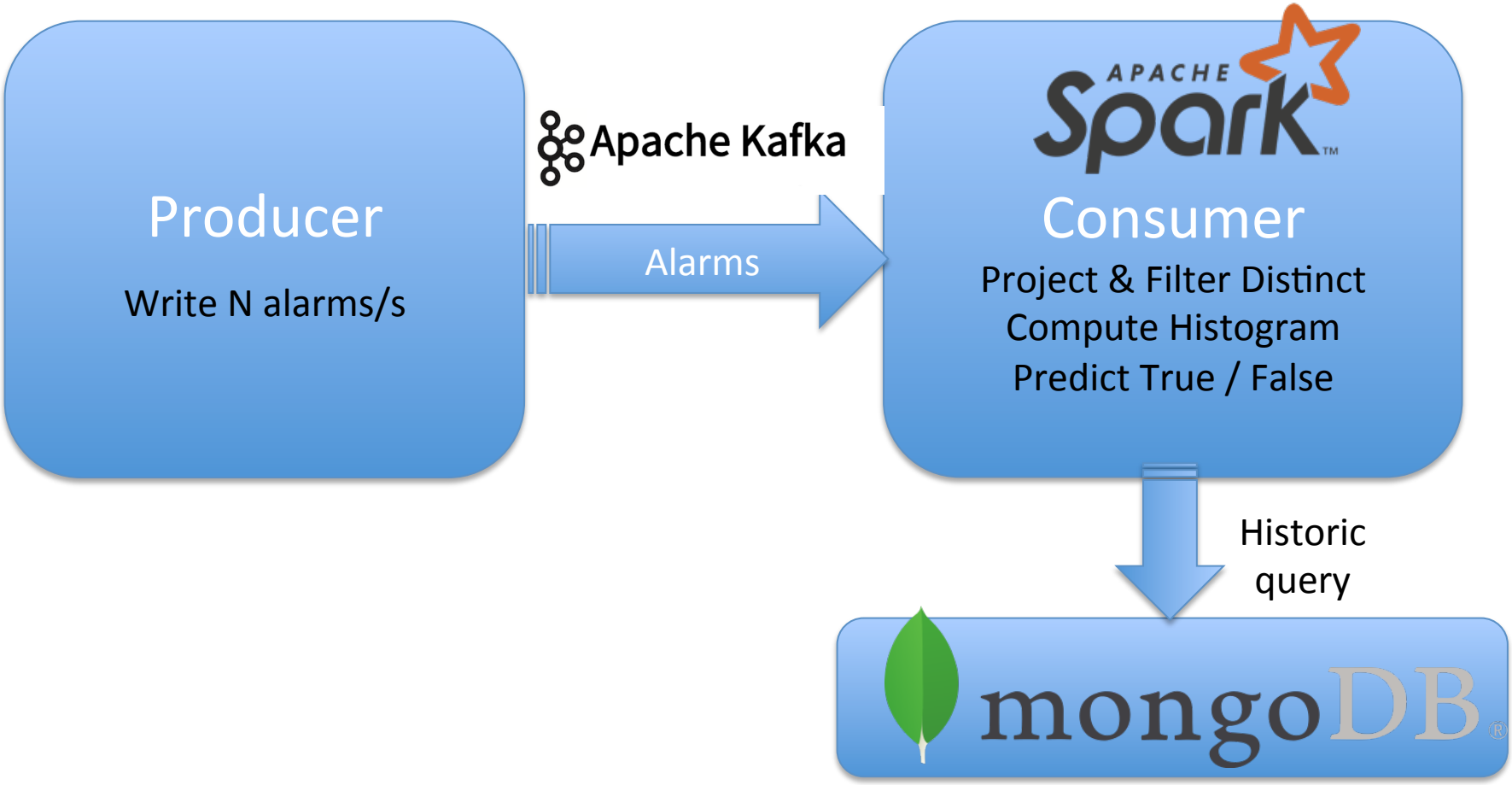
Designing the consumer

- Spark - ”A unified engine for Big Data Processing”



No one size fits all, so how do I choose???

The base case



Consumer Workflow

- Every window of 10s
  - Streaming
    - `mAddrsInWindow = alarms.map(Alarm::getMacAddress).distinct();`
  - Historic
    - `Select MacAddress, count(*) from mongoDB Where MacAddress in mAddrsInWindow Group By MacAddress`
  - Machine Learning
    - *(Covered in first half of talk)*


ACTION!
PRODUCTION SAVE Benchmark
SCENE Identifying Bottlenecks
TAKE 1
DIRECTOR ZHAW Datalab
CAMERA
DATE

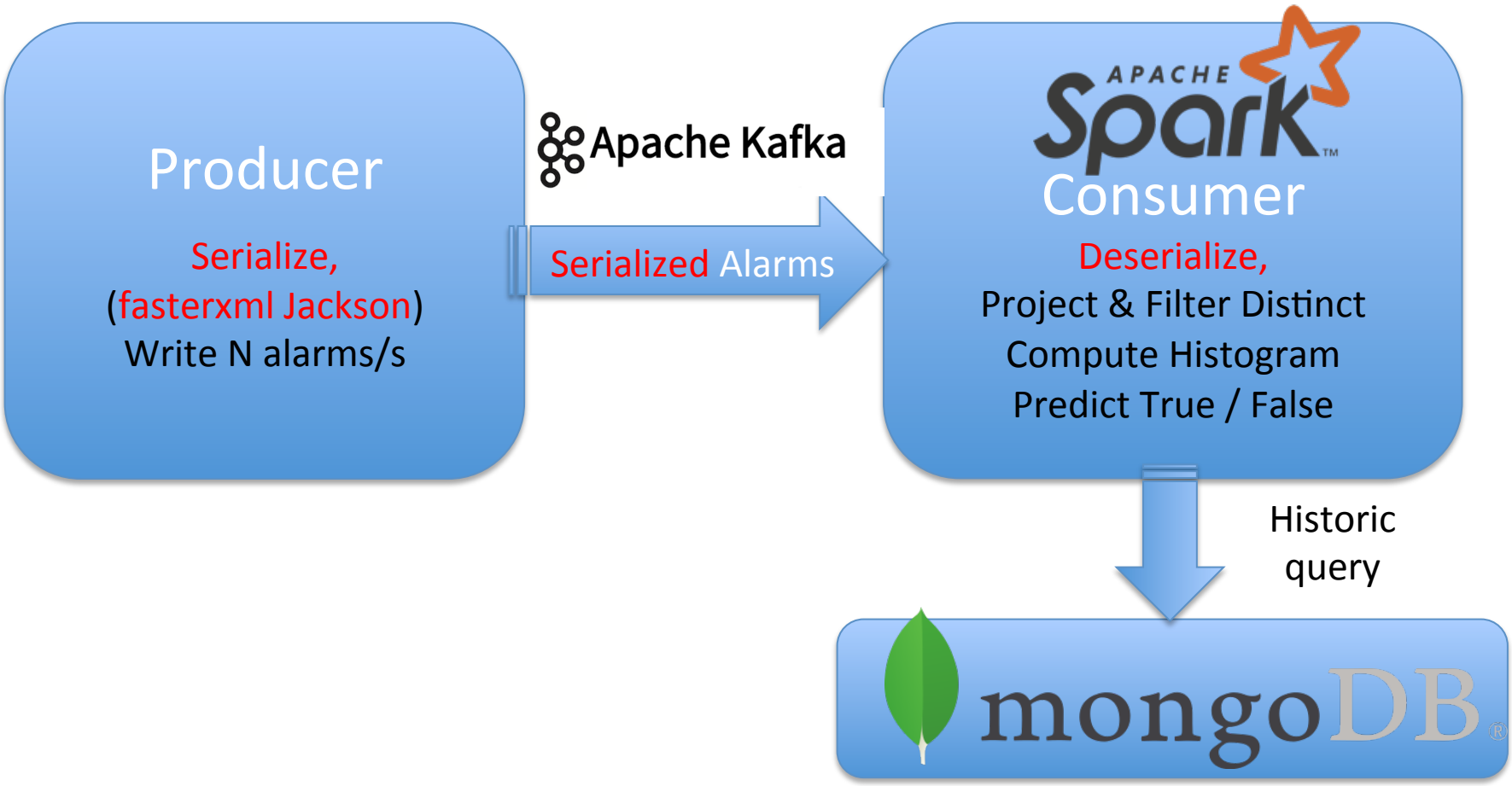
Identifying bottlenecks (1)

1. Started experiment with
   - 8-core Kafka Producer
   - 8-core Spark Consumer

2. Consumer slow even for a basic streaming task
   - *Count number of elements in window => a few sec*

3. Producer not outputting expected throughput
   - *Stumbled @ around 12K/s (record size ≃ 600B)*

Common root cause?



Does it really matter?

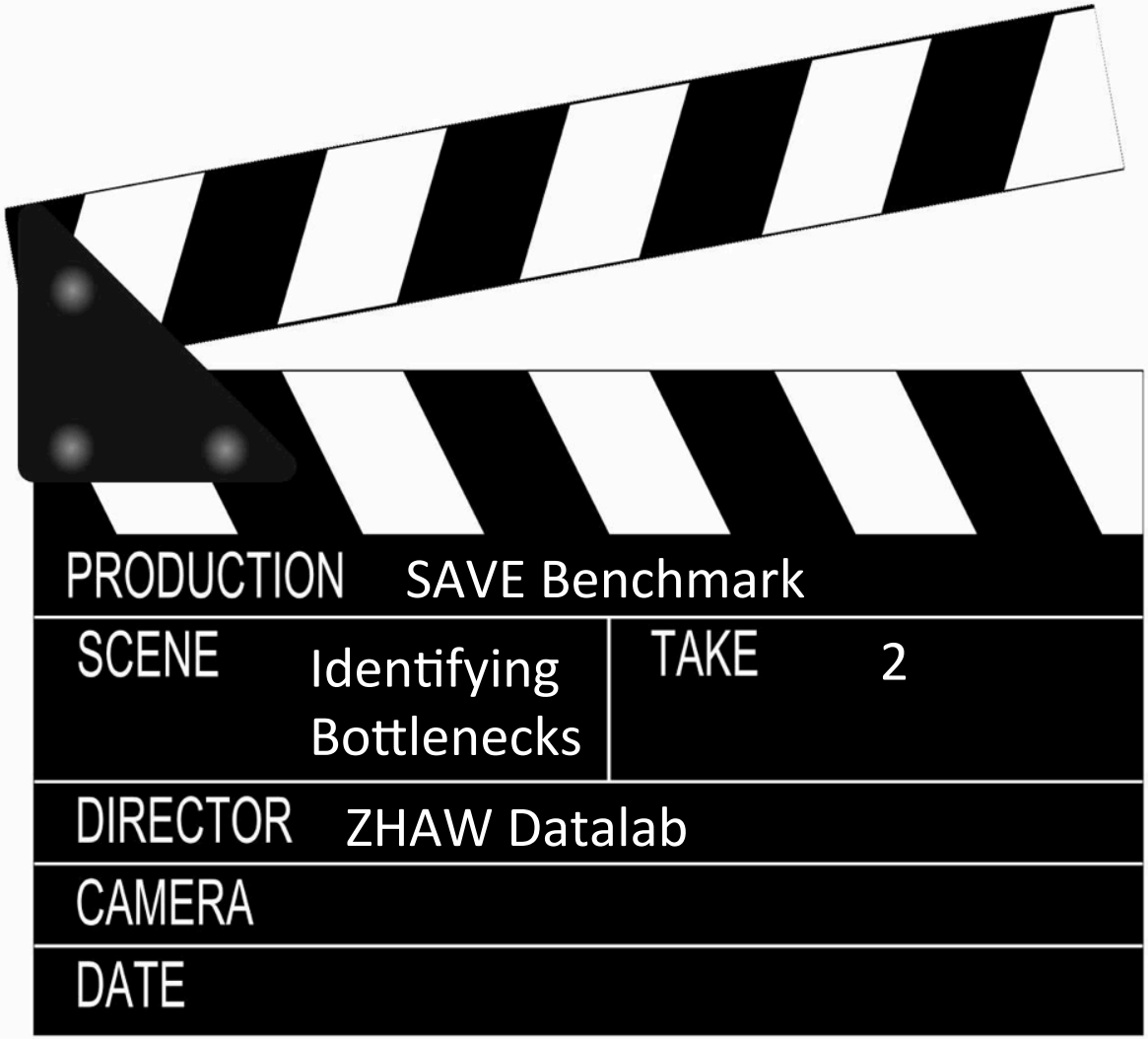
- Per-object serialization only takes a few tens of ns

- Multiply that by 12K..
- And btw, have it all ***sent*** in 1s!

Are we using the right tool?

- Answer: benchmarks*!
- ”If your environment primarily deals with lots of small JSON requests […] **then GSON is your library of interest**. Jackson struggles the most with small files.”
- Jackson: up to **3x** slower for small objects

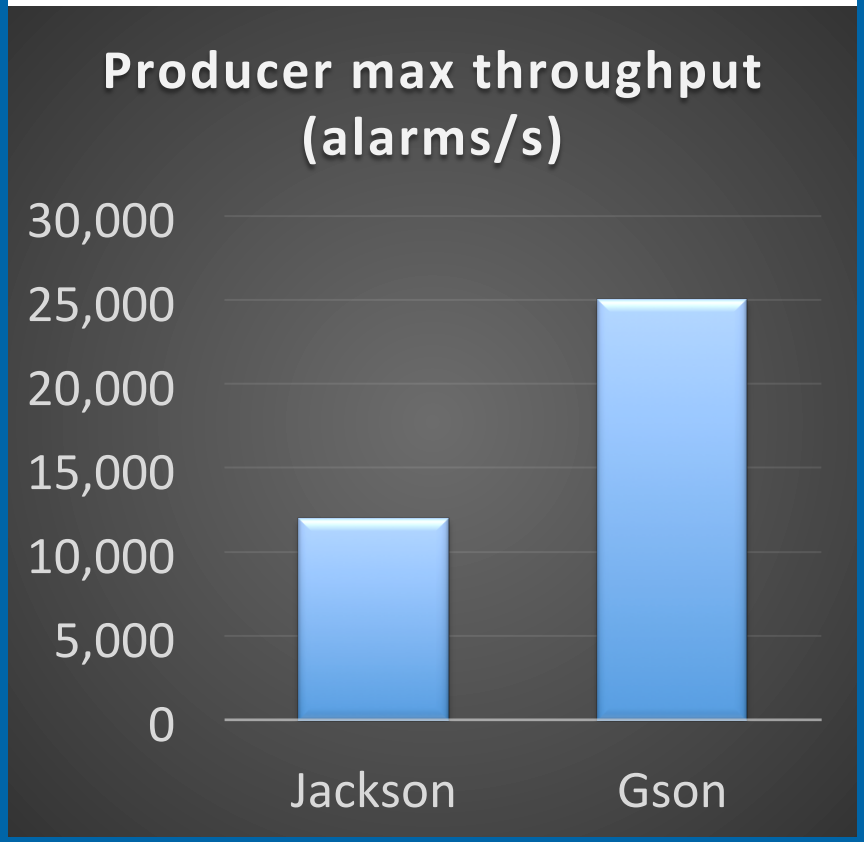
**The ultimate JSON library:*
http://blog.takipi.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/


PRODUCTION
SAVE Benchmark
SCENE
Identifying
Bottlenecks
TAKE
2
DIRECTOR
ZHAW Datalab
CAMERA
DATE

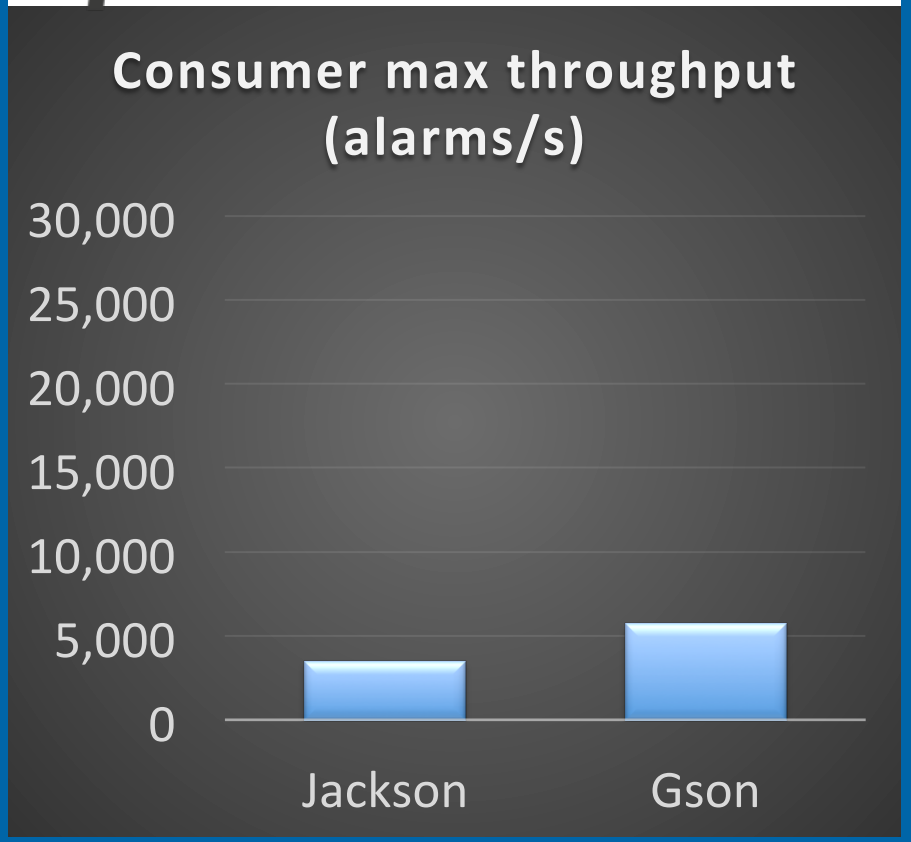
Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Results (1)

Apache Kafka

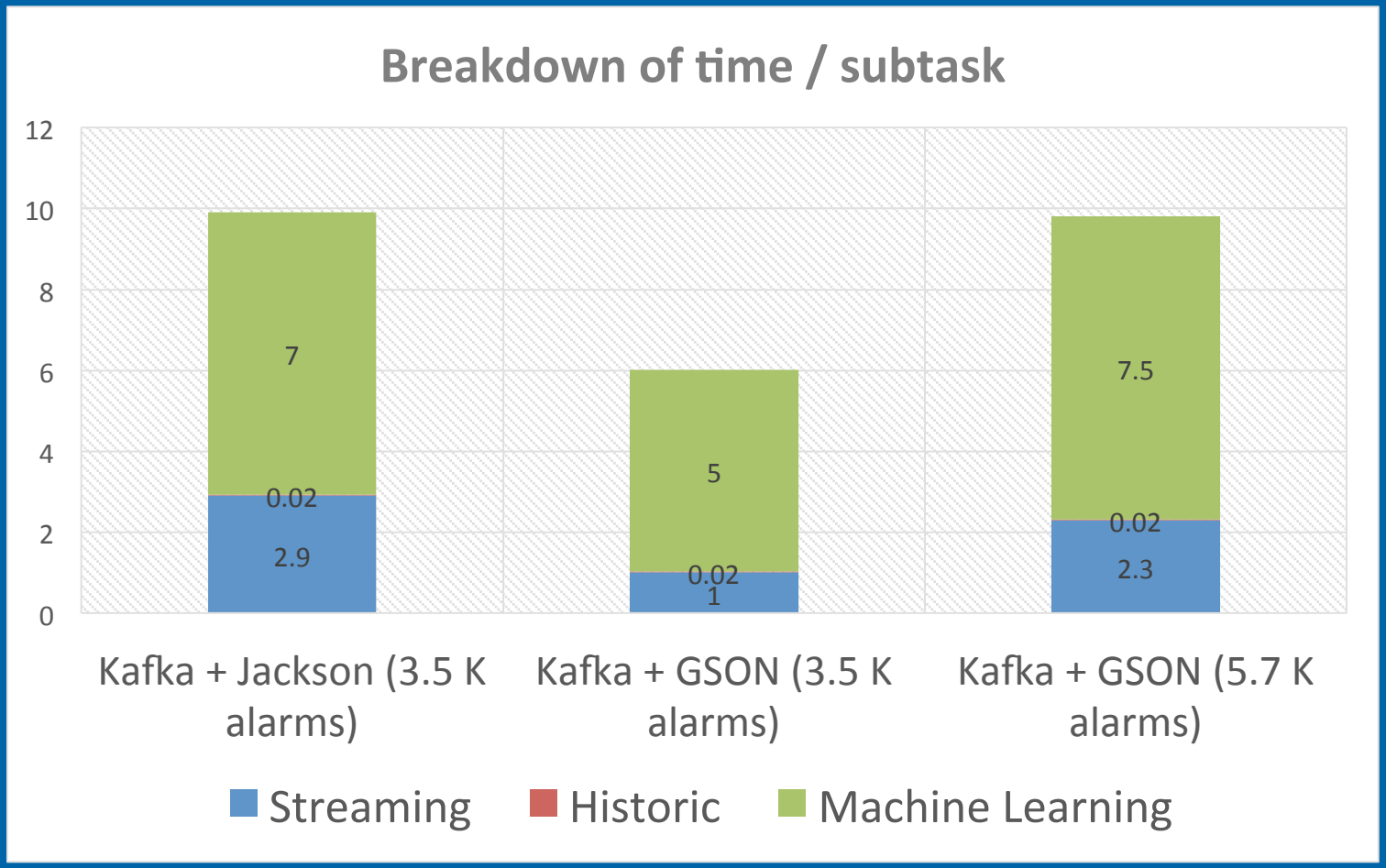


Apache Spark + mongoDB



* < 30 LOC change => ~ 2x speedup in *both* Producer & Consumer

How much will fit in 10s?



Note: the alarms RDD is used for both the streaming & the ML part => serializer change benefits both

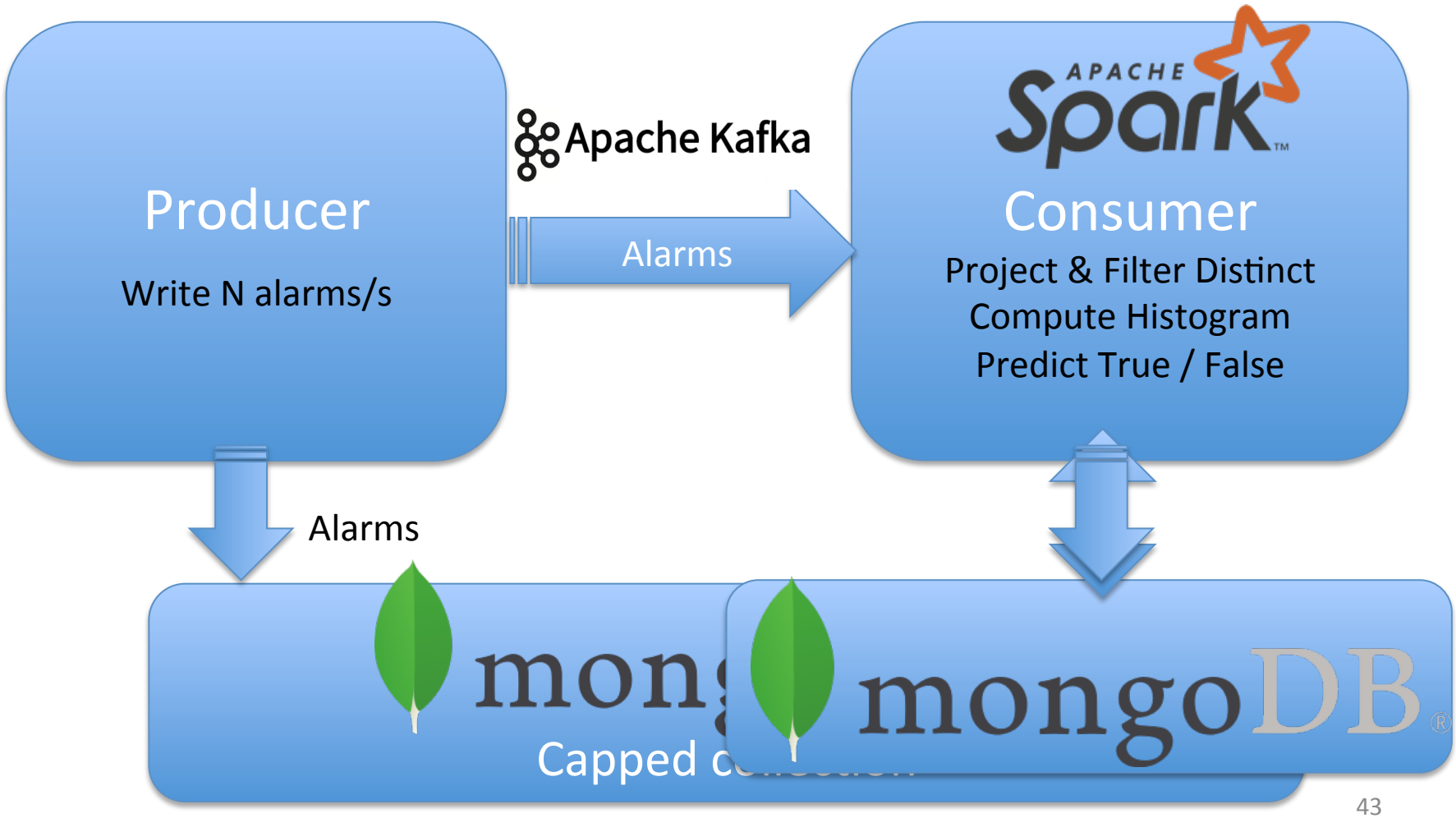
Identifying bottlenecks (2)

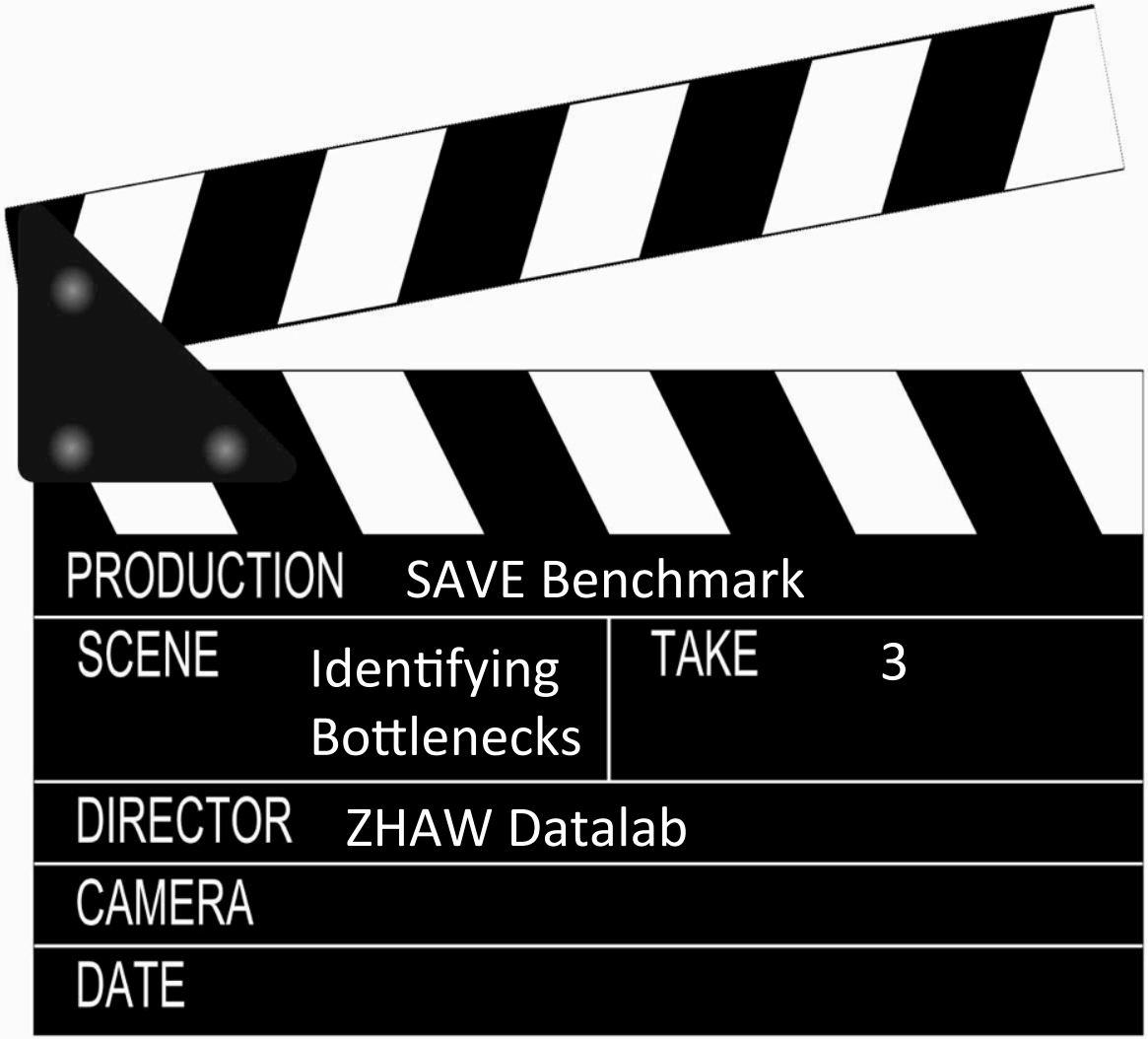
1. Caching?
   - Brought some improvement, but small (~ 5%)
   - The amount of data reused is too little (20 MB)

2. Simplifying the design?
   - The KISS* principle
     - Keep It Simple, Stupid!

Simplification

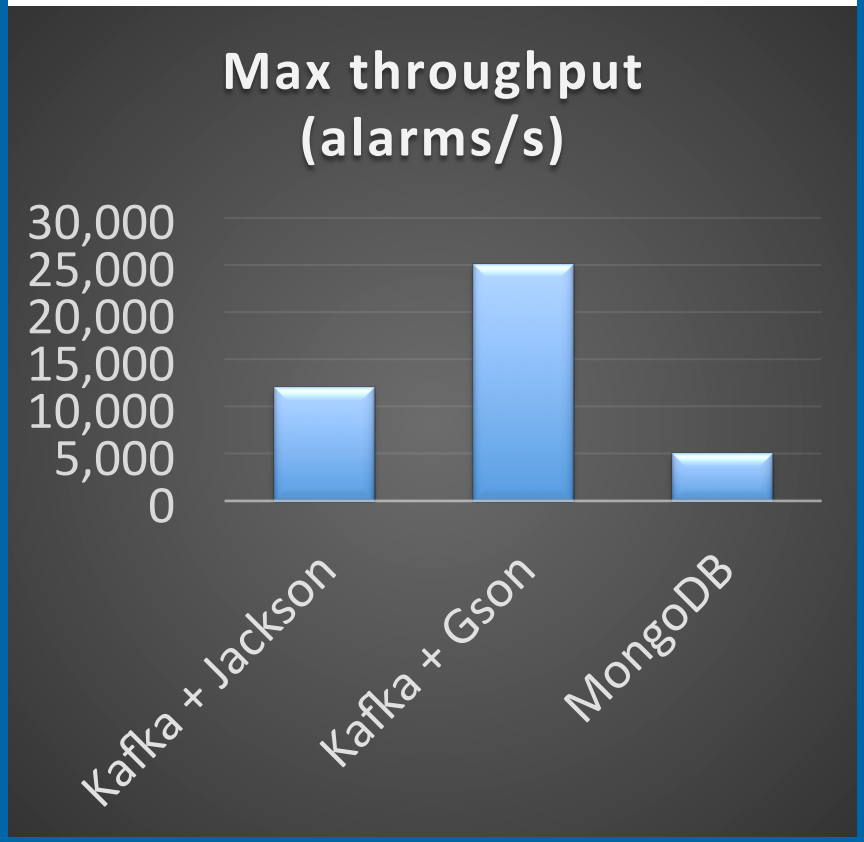
Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie



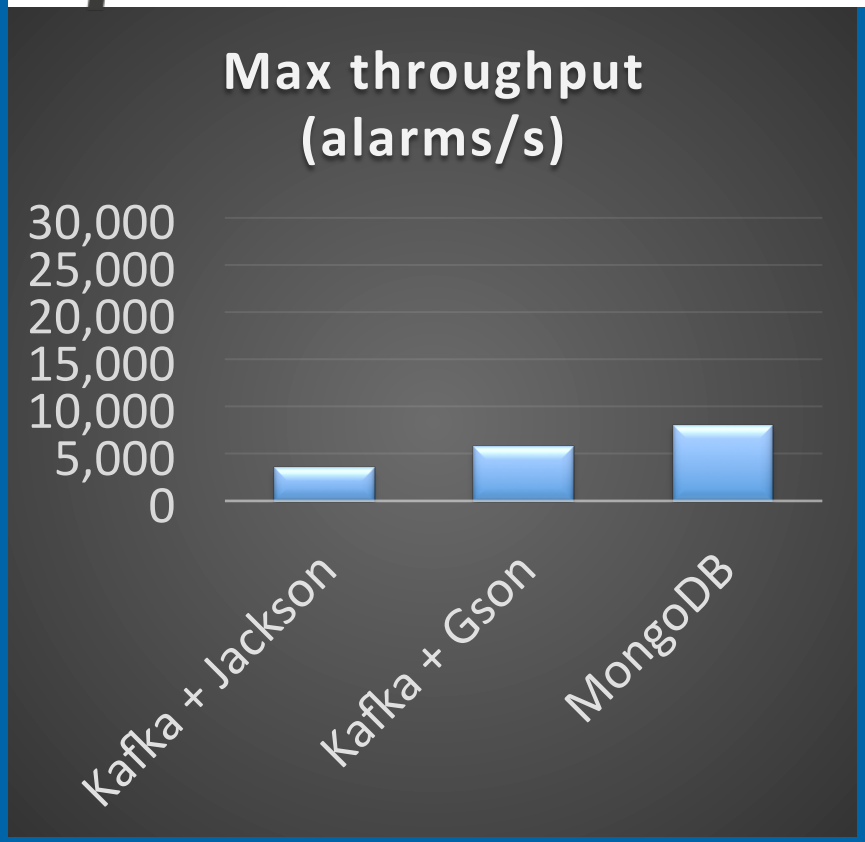

PRODUCTION
SAVE Benchmark
SCENE
Identifying
Bottlenecks
TAKE
3
DIRECTOR
ZHAW Datalab
CAMERA
DATE

Results (2)

Producer

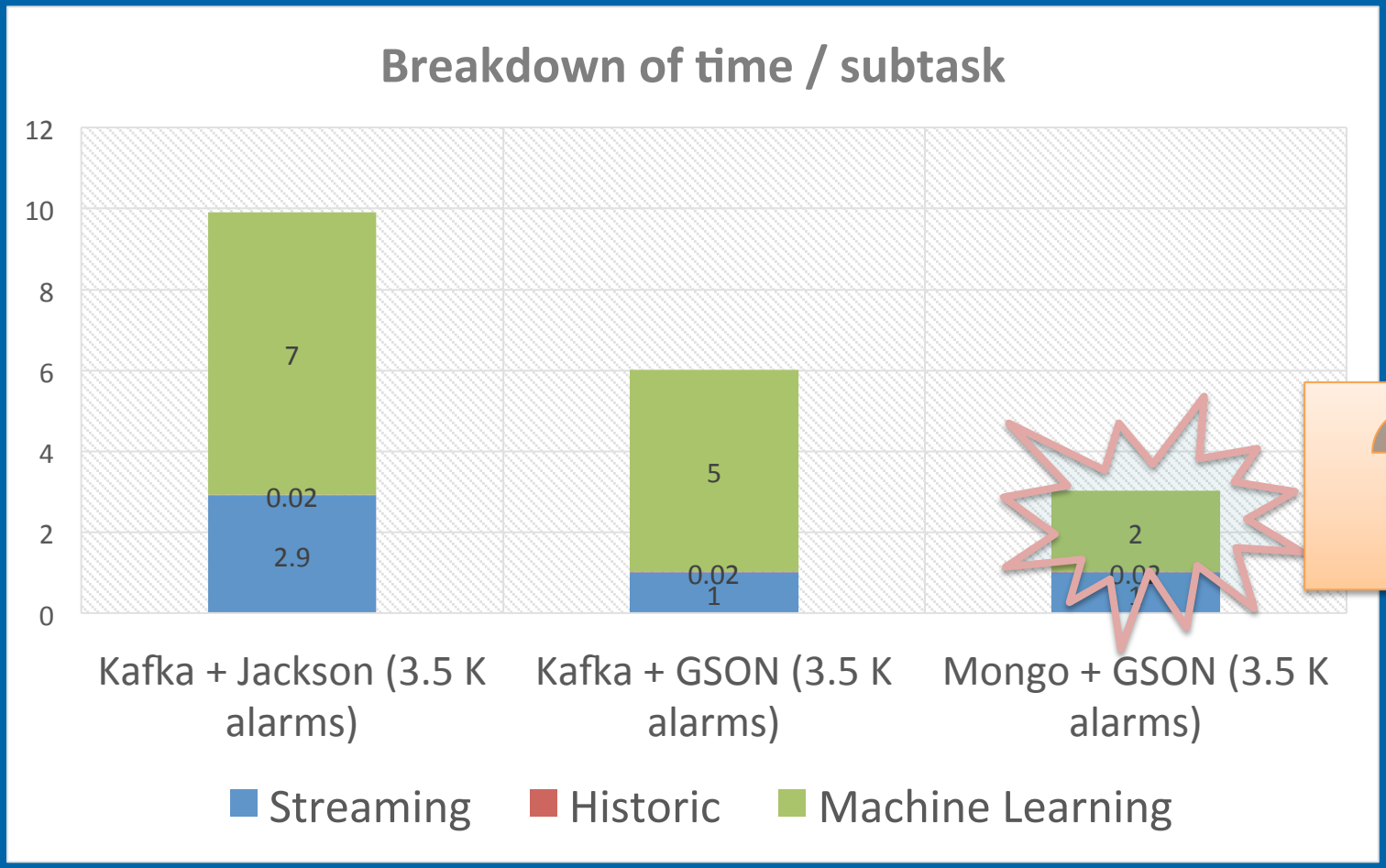


APACHE Spark + mongoDB



Note: Producer slow, but still @ previous *max consumer throughput*!
Consumer: ~ 50% speedup.

How much will fit in 10s?



Root cause?

- Using Kafka direct stream with a single partition => **poor parallelism**
  - Most operations *run on a single executor*
    - Even though Spark is configured to run on 8 cores

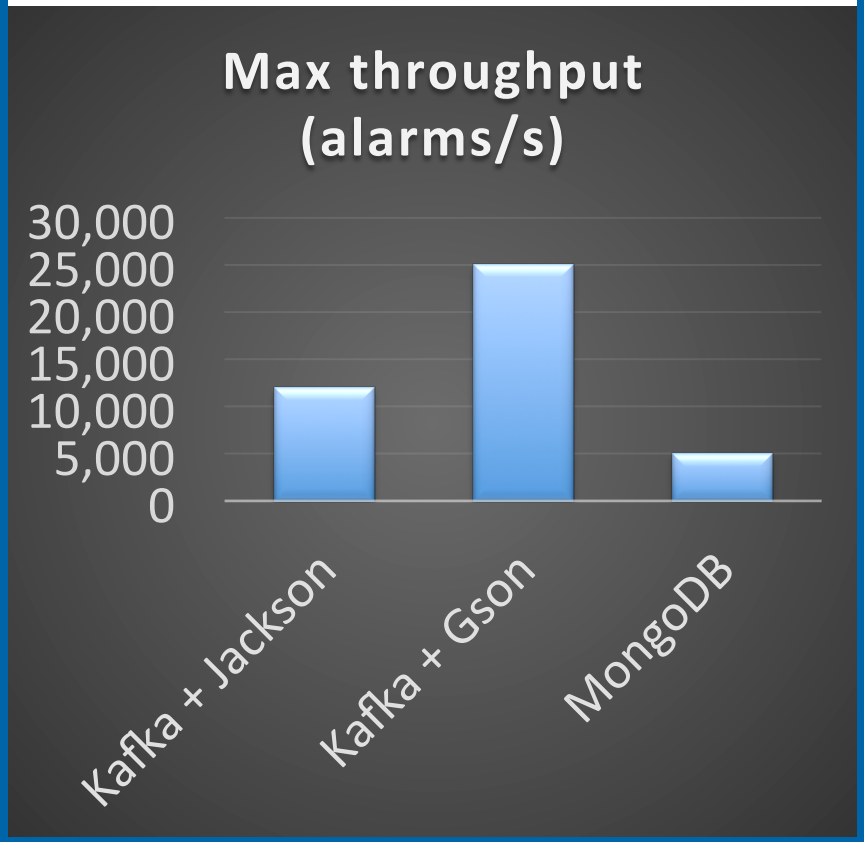
- Using MongoDB => finer grained control over parallelism level for the RDD
  - Read an array of documents from tailable cursor
  - Parallelize array into RDD & run ML on each part

FIX IT!

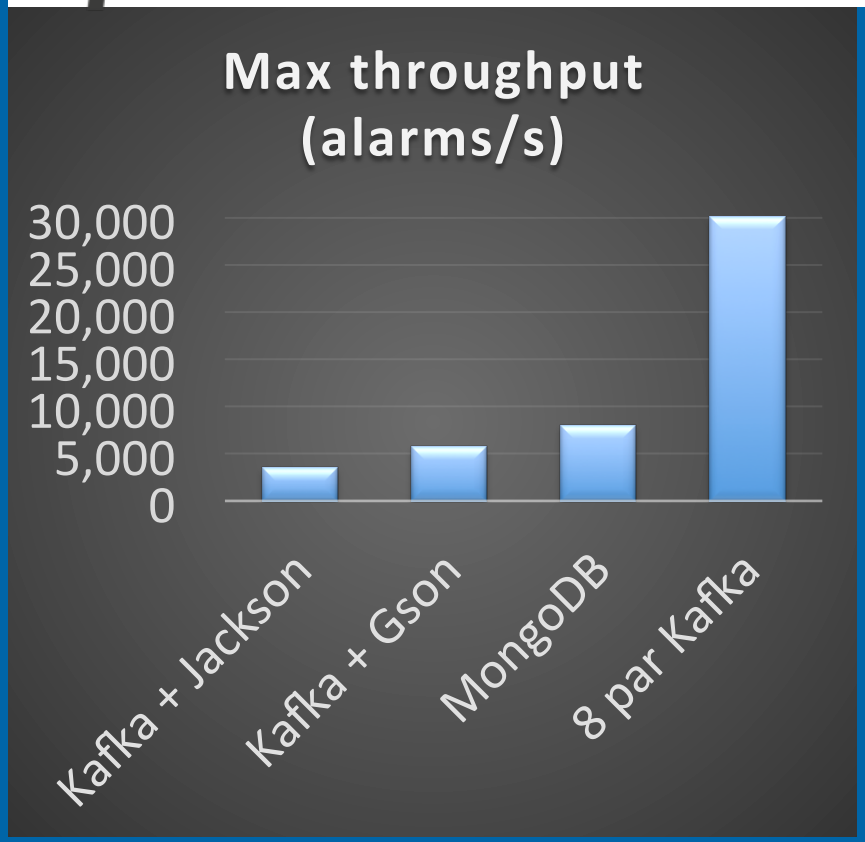
Configuring the Kafka Direct Stream in Spark with proper settings….
(num partitions = num cores)

Results (3)

Producer

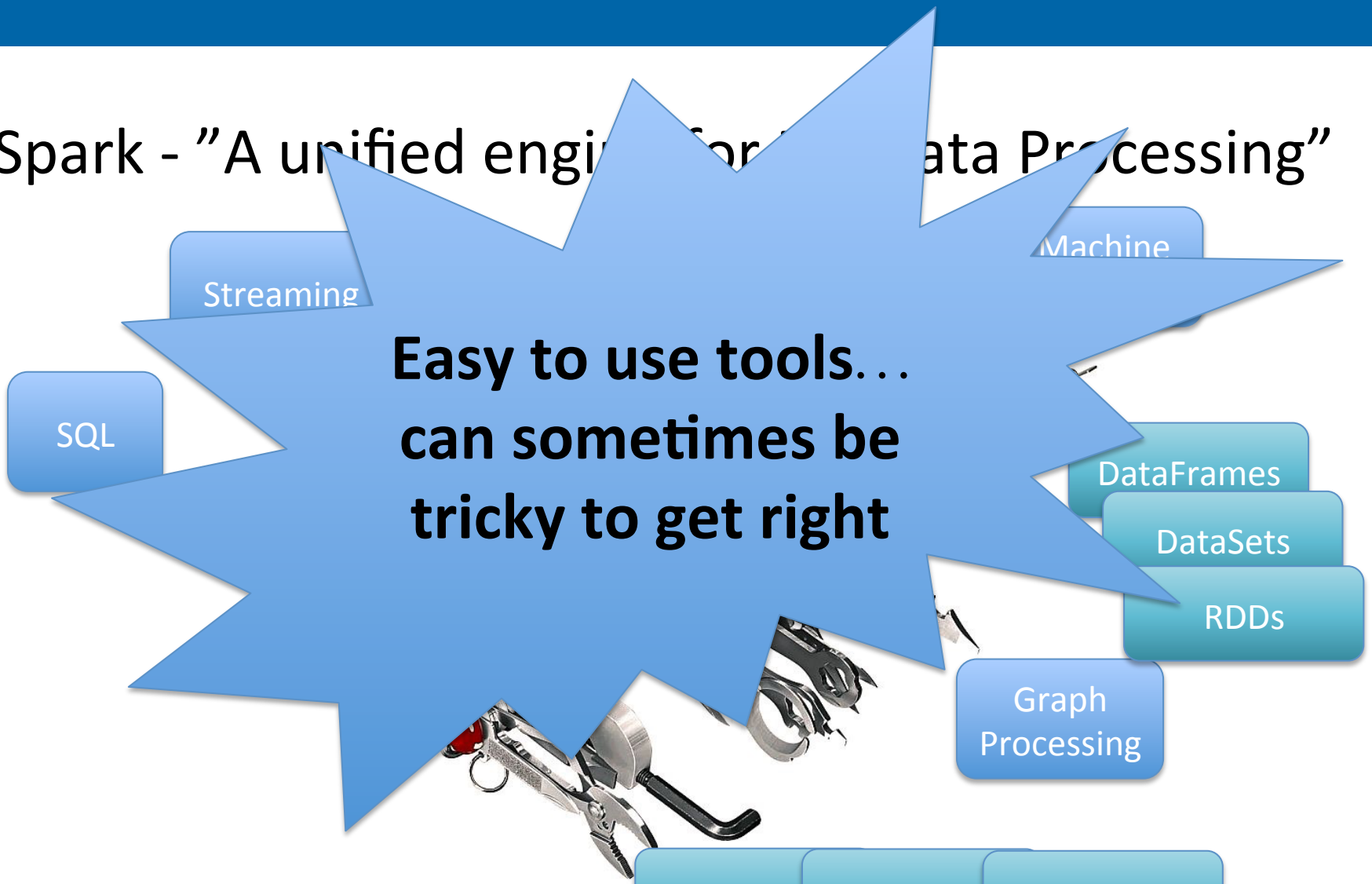


APACHE Spark™ + mongoDB®



Designing the consumer

- Spark - ”A unified engi for ata Processing”



Conclusions & Lessons Learned

- Machine learning algorithm of Spark can be directly applied
  - Works well for small and big data
  - No re-writing necessary to make algorithm scale across computing cluster

- Spark Streaming can't be used out of the box for stream and batch processing:
  - Need to use a persistency layer
  - Requires significant performance tuning

- Working with our industry partner Sitasys:
  - Great collaboration
  - Working with real-world problem is very rewarding
  - Can make real contribution and enhance existing algorithms and technology

Appendix

Pipeline
Example Code (Java)


```
// Create string indexer Transformer.
StringIndexer stringIndexer = new StringIndexer()
        .setInputCol("location").setOutputCol("indexLocation");

// Create random forest Estimator.
RandomForestClassifier randomForestClf = new RandomForestClassifier()
        .setFeaturesCol("indexLocation").setLabelCol("label");

// Create pipeline Estimator (specifying the ML workflow).
Pipeline pipeline = new Pipeline()
        .setStages(new PipelineStage[]{stringIndexer, randomForestClf});

// Create cross validation Estimator (wrapping the pipeline Estimator).
ParamMap[] paramGrid = new ParamGridBuilder()
        .addGrid(randomForestClf.numTrees()(), new int[]{10, 20, 30})
        .addGrid(randomForestClf.maxDepth(), new int[]{5, 25, 50}).build();

CrossValidator crossValidator = new CrossValidator().setEstimator(pipeline)
      .setEstimatorParamMaps(paramGrid)
      .setEvaluator(new BinaryClassificationEvaluator())
      .setNumFolds(10);
```

Training, Testing and Evaluation Example Code (Java)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

```
// Given (creation of dataset no included in example code)
Dataset<Row> trainSet = …;
Dataset<Row> testSet = …;

// Fit the cross validator to training documents.
CrossValidatorModel model = crossValidator.fit(trainSet);

// Make predictions on test documents.
Dataset<Row> predictions = model.transform(testSet);

// Create evaluator.
MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator()
      .setLabelCol("label").setPredictionCol("prediction");

// Evaluate predictions.
evaluator.setMetricName("accuracy");
double accuracy = evaluator.evaluate(predictions);
```