Big Data software infrastructures

Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)

Gábor Hermann
Zoltán Zvara
András Benczúr
Informatics Laboratory
„Big Data – Momemtum" research group

MTA SZTAKI, 10/11/2016

MTA **SZTAKI**

Outline

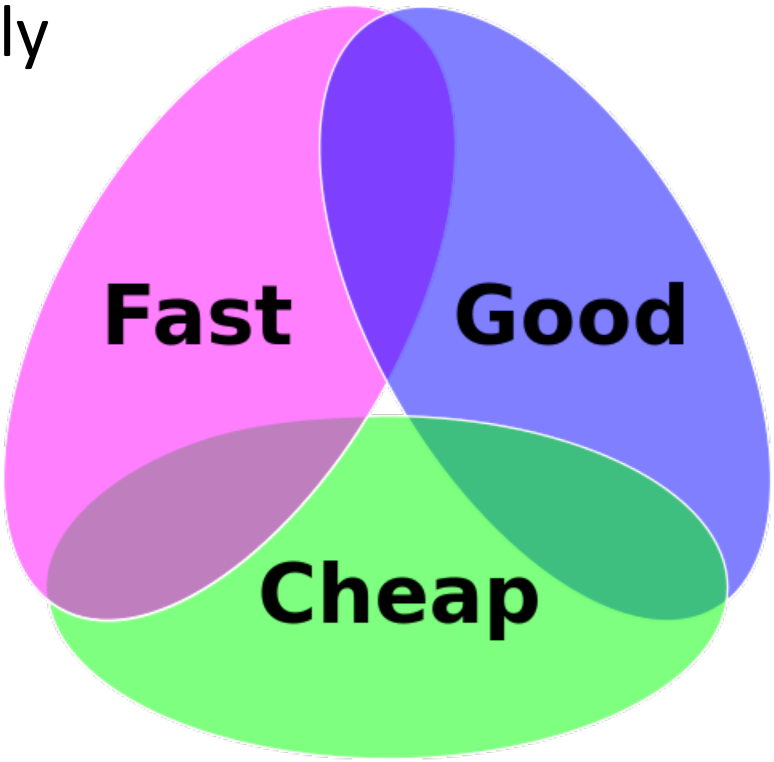
- Motivation for “Big Data”
- “Big Data” is about software infrastructure
- Batch and streaming approaches
- What we do at SZTAKI
- Solving a problem: handling dataskew

Some data about data

- Google was guesstimated to have over 1M machines in 2012
- Large Hadron Collider (LHC) collisions generated about 75 petabytes in past 3 years
- Facebook 31.25 million messages every minute (2015)

The Project Triangle

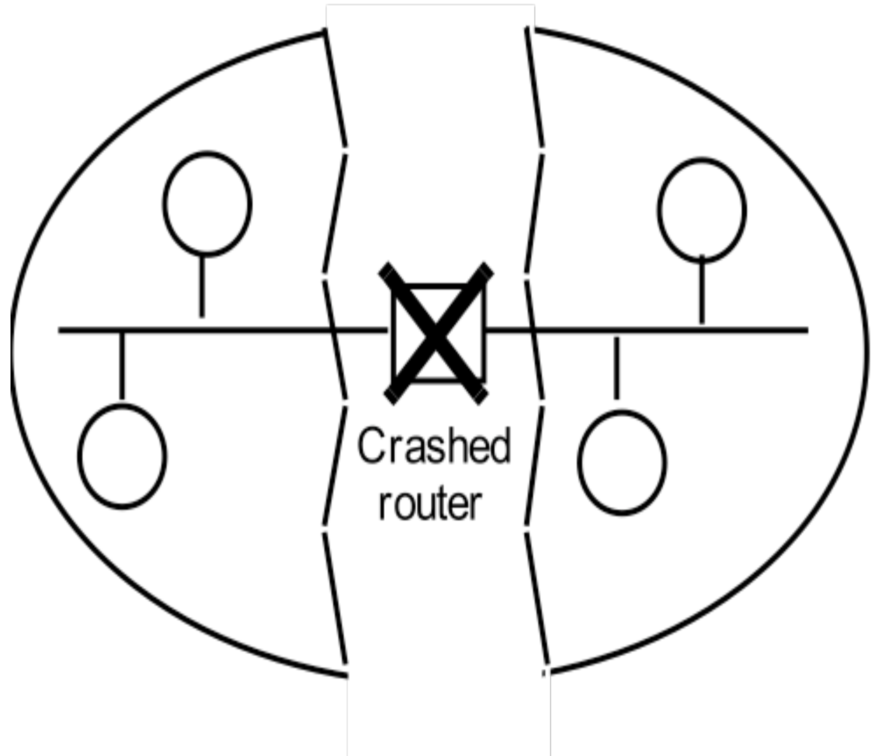
- Small machine to store and process data: slow
- Large servers are fast but costly
- Distributed computing?



MTA SZTAKI

Problem with distributed data processing

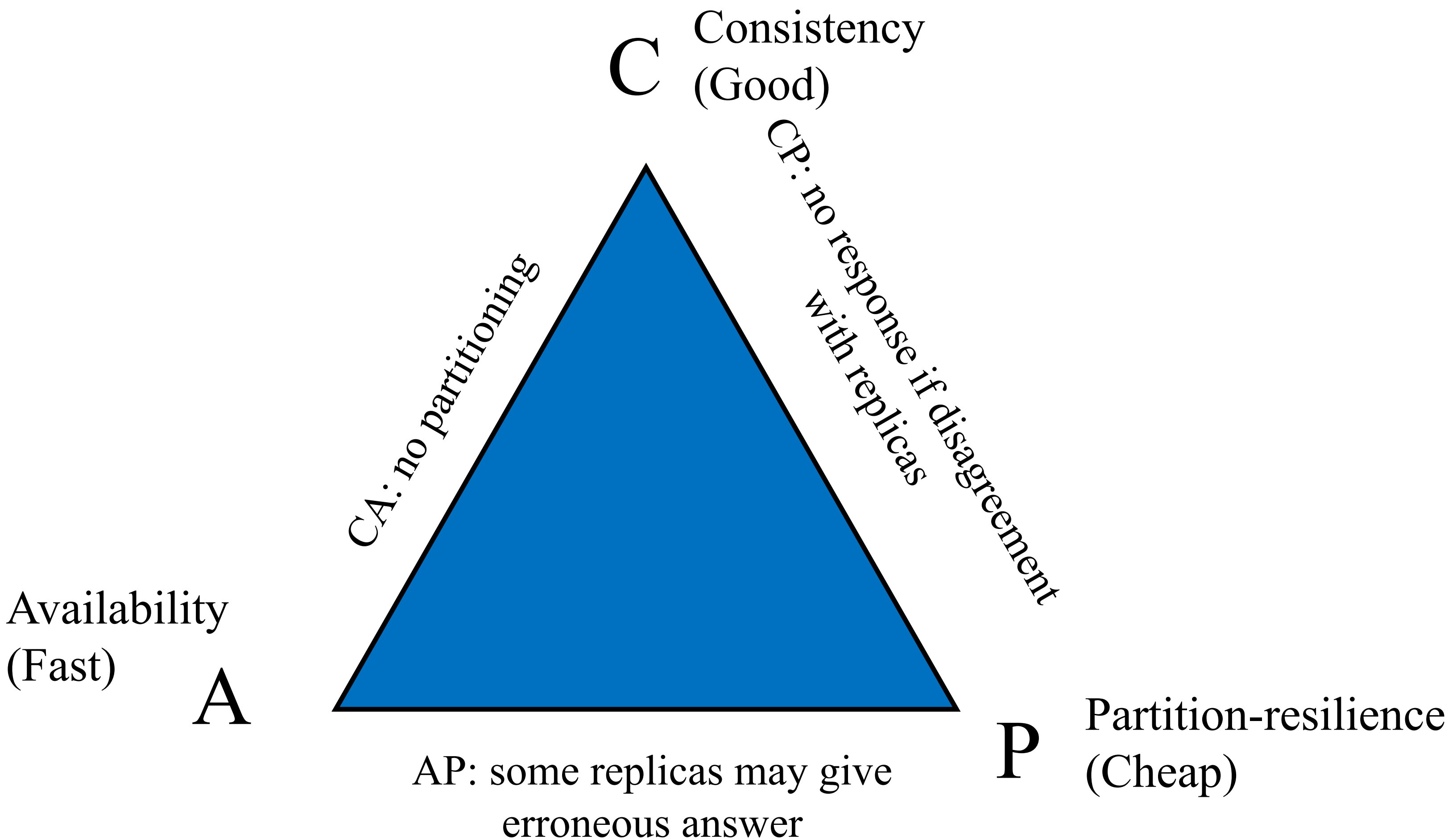
- When failures happen…
- Left side will not know about new data on right
- Immediate response from left side might give incorrect answer
- If failures can happen (**partitioned**) we can either choose correctness (**consistence**) or fast response (**availability**)



MTA **SZTAKI**

CAP (Fox&Brewer) Theorem

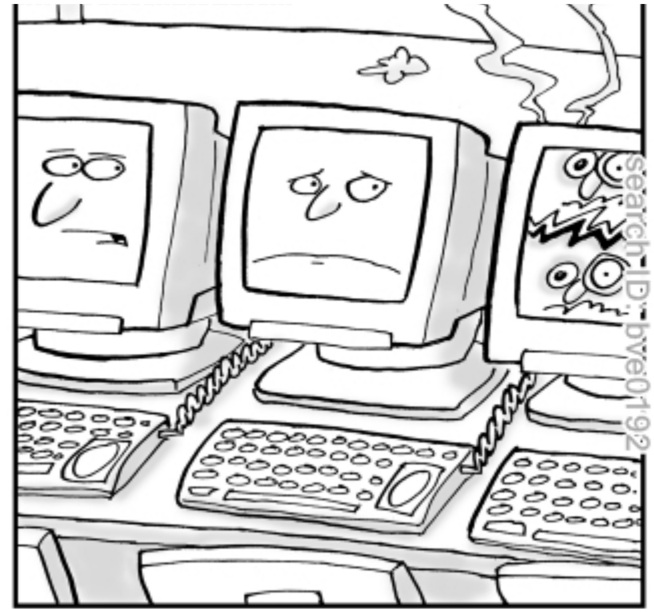
Theorem: You may choose two of C-A-P



MTA SZTAKI

Failures do happen in distributed data processing!

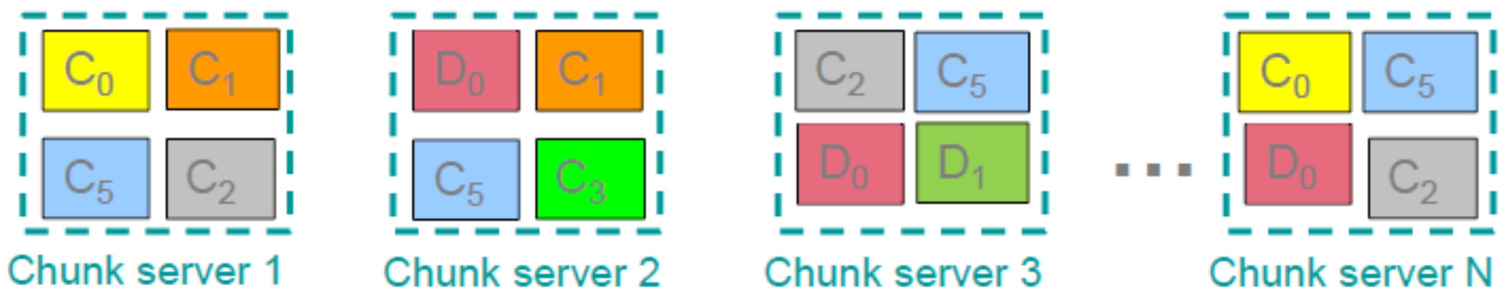
- We must have P,
  need to choose between A and C
- Fast response vs. correct results
- Most applications need fast response
  - Best we can do: eventual consistency
    (if connection resumes and data can be exchanged)
- **„Big Data" today is mostly about software infrastructure**
  - Trying to do the best



MTA **SZTAKI**

Approach 1: batch processing

- Process the whole dataset
- Consistent, but takes time (hours, days)
  - If failure happens, wait for recovery (choosing CP)
- Apache Hadoop
  - MapReduce, HDFS (distributed file system)
  - Data in chunks across many machines, replicated
  - Bring computation close to the data



Beyond Hadoop and MapReduce (batch)

- MapReduce has the first open source distributed software, Hadoop
  - Limitations
  - Join and more complex primitives
  - Graphs, machine learning
- Alternatives
  - Graph processing: Apache Giraph, Apache HAMA, …
  - In memory based: Apache Spark
  - Streaming dataflow engine: Apache Flink

Approach 2: stream processing

- Continuously process all incoming data
- Faster response time (low-latency, within 1 sec)
  - If failure: wait for recovery
  - Still choosing PC, but sophisticated recovery mechanisms give lower downtime
- Harder to implement and reason about
- Stream processing frameworks
  - Apache Flink, Apache Storm, Apache Spark

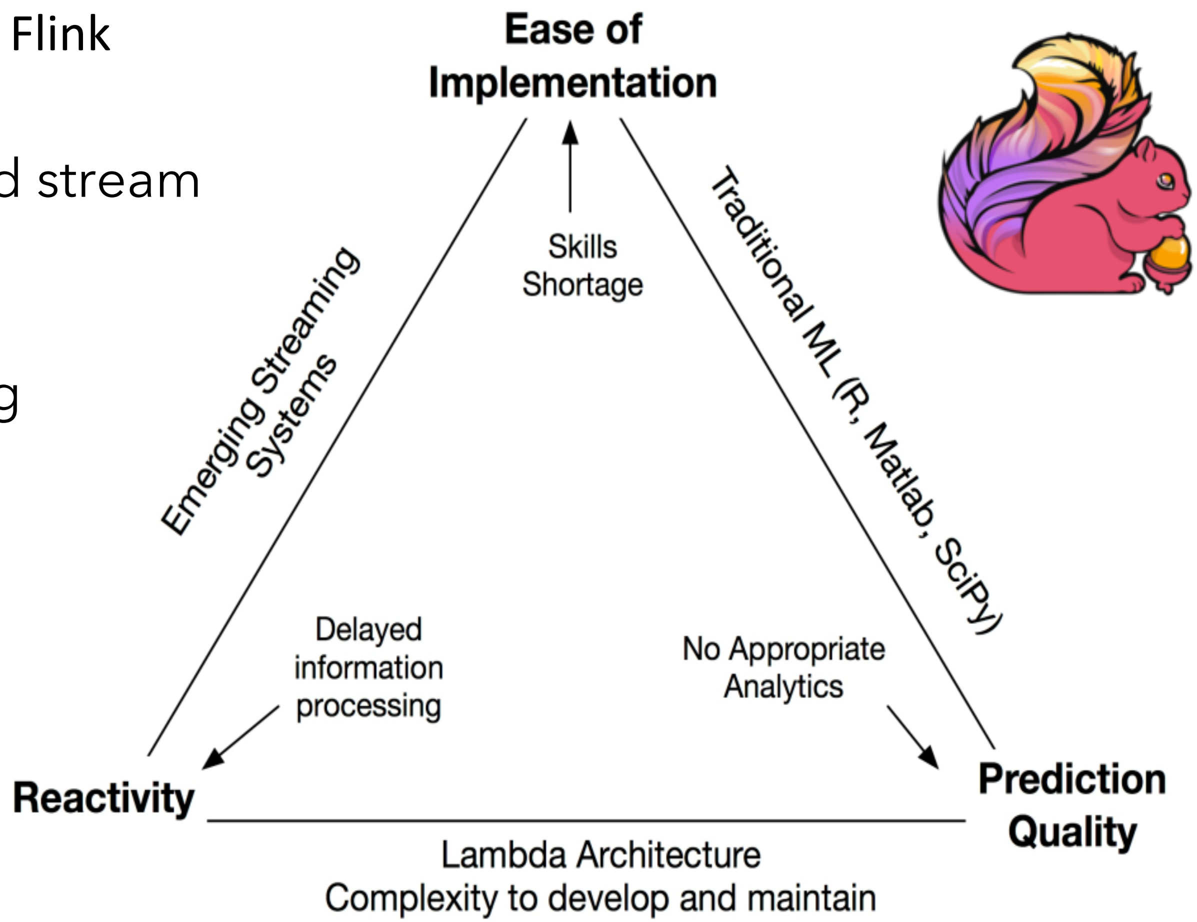
STREAMLINE H2020

New initiative on top of Apache Flink

A general data processing framework to unify batch and stream processing

At SZTAKI: Machine Learning

- DFKI (DE)
- SICS (SE)
- Portugal Telecom (PT)
- Internet Memory (FR)
- Rovio (FI)
- SZTAKI (HU)

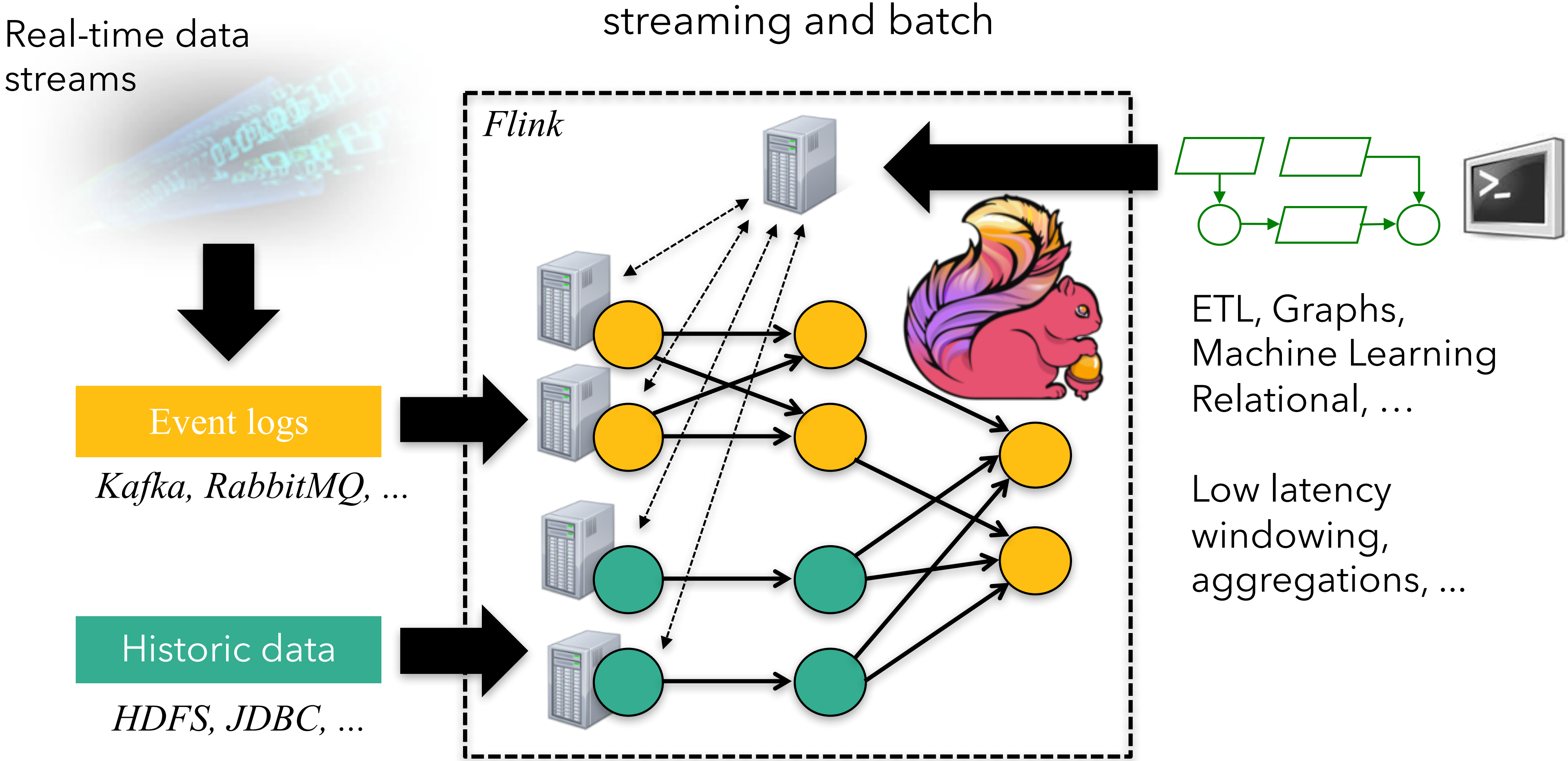


B. – Volker Markl (TU Berlin)

MTA SZTAKI

Batch and stream: same execution engine

An engine that puts equal emphasis to streaming and batch



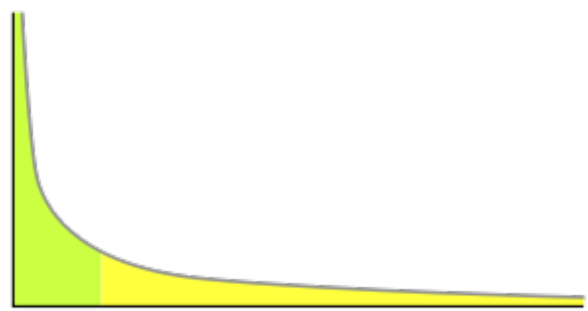
MTA SZTAKI

What we do for “Big Data” at SZTAKI

- Developing Apache Flink (STREAMLINE H2020)
  - Machine Learning algorithms, experimenting
- Projects with industrial partners
  - Using Spark, Flink, Cassandra, Hadoop etc.
- Research
  - Improvements on current systems
  - Ongoing project: handling dataskew

Solving a problem: handling dataskew

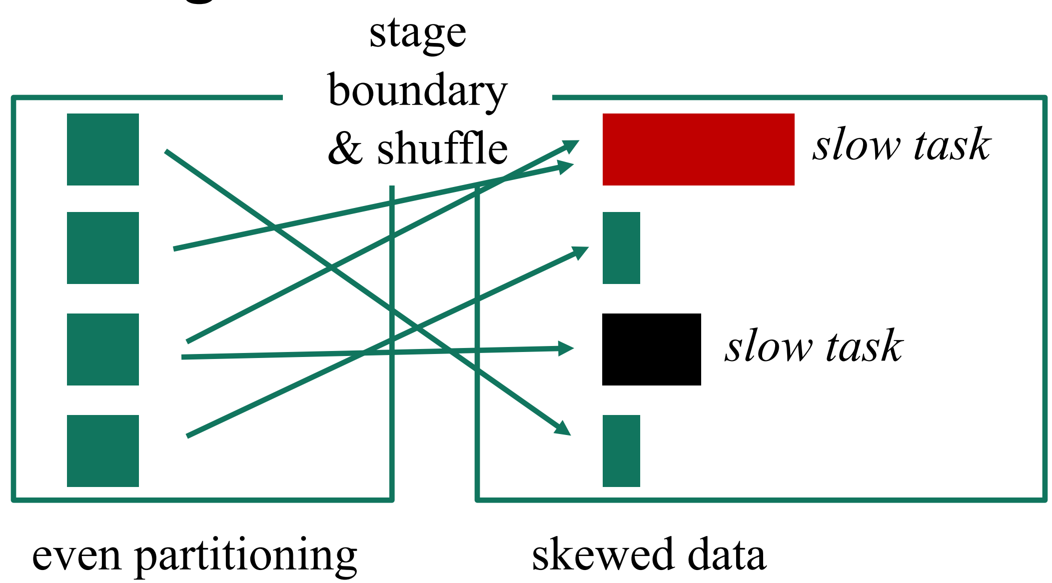
- We have developed an application aggregating telco data
- After a while, on real dataset it could become slow or even crash
- Investigated the problem: dataskew
- 80% of the traffic generated by 20% of the communication towers



MTA SZTAKI

The problem

- Default hashing is not going to distribute the data uniformly
- Data distribution is not known in advance
- The heavy keys might even change

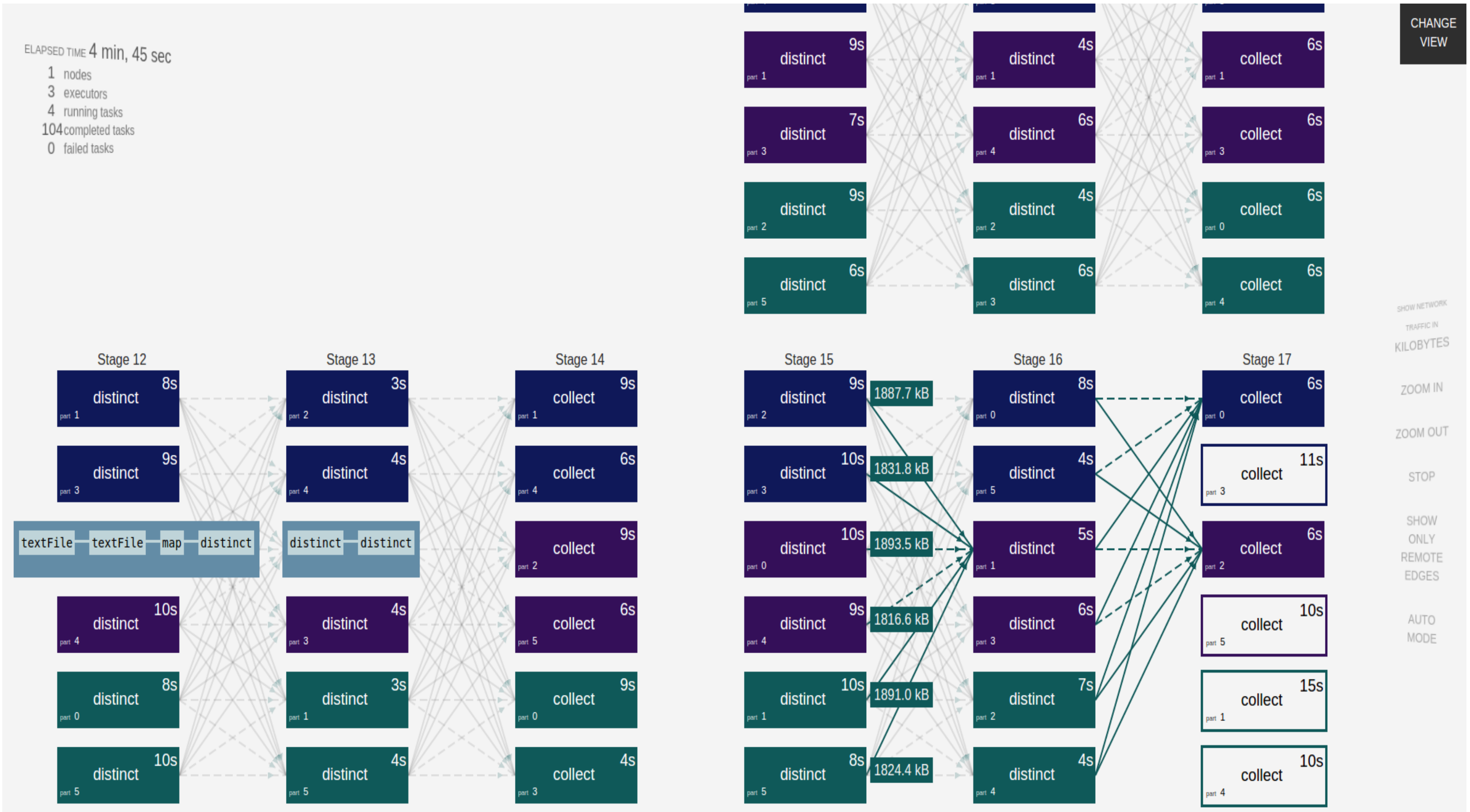


MTA SZTAKI

Our solution: Dynamic Repartitioning

- Monitoring tasks, repartitioning based on that data
- System aware
  - No significant overhead
- Can handle arbitrary data distributions
- Does everything on-the-fly
  - Works for streaming and batch
- Pluggable
  - Initially on Spark batch and streaming
  - Plugged into Flink streaming

Execution visualization of Spark jobs



MTA SZTAKI

Future in handling dataskew

- Generalizing the problem
- Balancing load **in** a processing system
- Balancing resources **between** processing systems on a cluster (YARN)

Conclusion

- We can tame “Big Data” with better software
- Lot to do…
  - Connection between batch and streaming
  - Highly scalable machine learning (batch and online both)
  - Optimizations (e.g. handling dataskew)
  - Better understanding our tools (e.g. visualization)
  - …
- Evolving fast, but we can take part in it

Questions?

- You can reach us!
  - ghermann@ilab.sztaki.hu
  - zoltan.zvara@ilab.sztaki.hu
  - benczur@ilab.sztaki.hu

References

- STREAMLINE H2020
  - https://streamline.sics.se/
- Dynamic Repartitioning
  - https://spark-summit.org/2016/events/handling-data-skew-adaptively-in-spark-using-dynamic-repartitioning/
- Visualizations
  - http://flink-forward.org/kb_sessions/advanced-visualization-of-flink-and-spark-jobs/

References (continued)

- Apache Hadoop
  - https://hadoop.apache.org/
- Apache Flink
  - https://flink.apache.org/
- Apache Spark
  - https://spark.apache.org/

References (continued)

- E. A. Brewer. Towards robust distributed systems, 2000
- J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters, 2004
- N. Marz. How to beat the CAP theorem, 2011
  - http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html

MTA SZTAKI