Adding Back-pressure to Spark Streaming

Dean Wampler, Typesafe

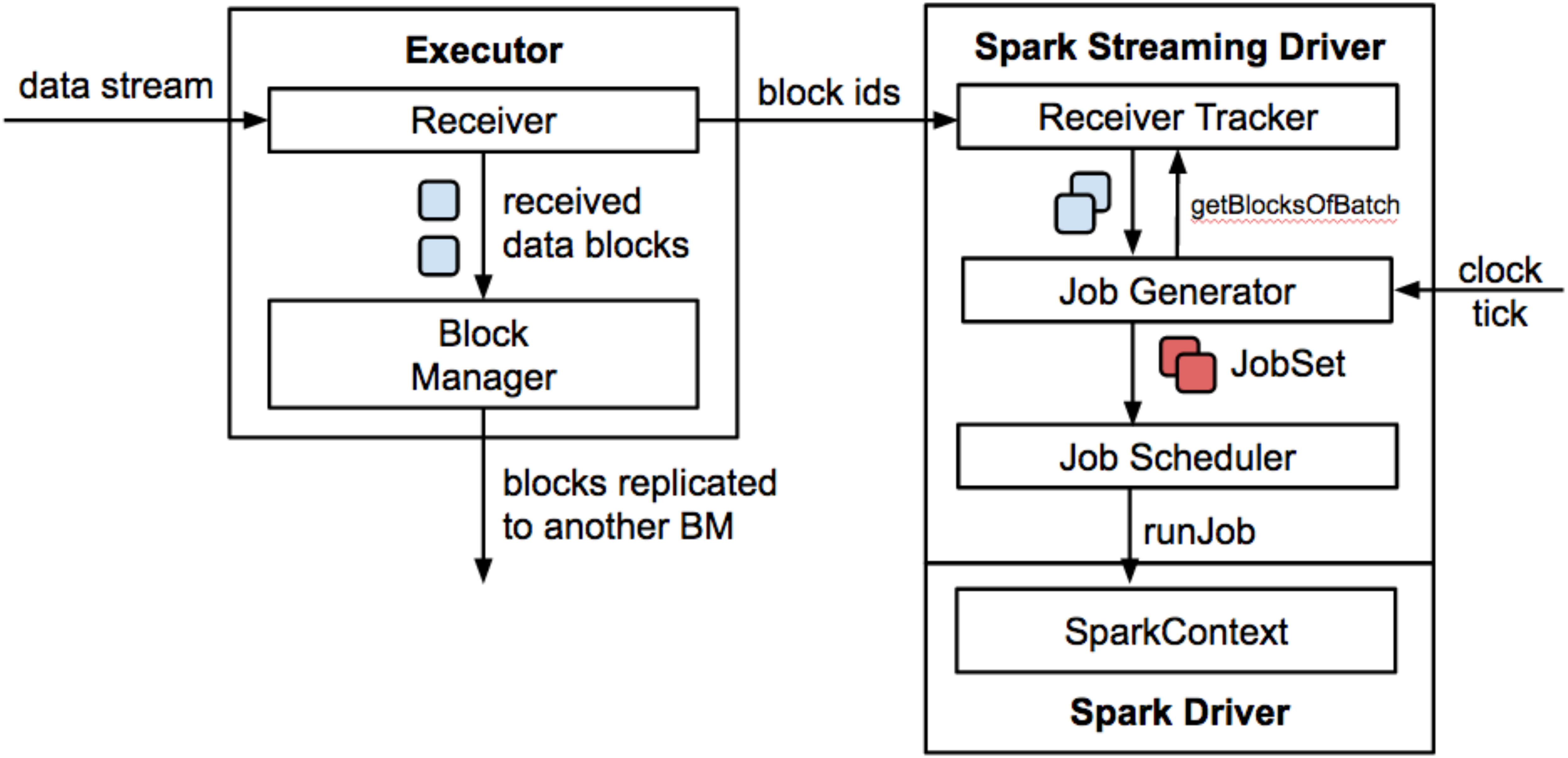
Typesafe and Spark

- Spark aligned with our passion for distributed, streaming (“fast data”) systems… and Scala!
- We’re contributing to Spark’s use of Scala, Streaming, and Mesos integration.
- Commercial support for Spark on Mesos.
    - *Mesosphere Infinity*.

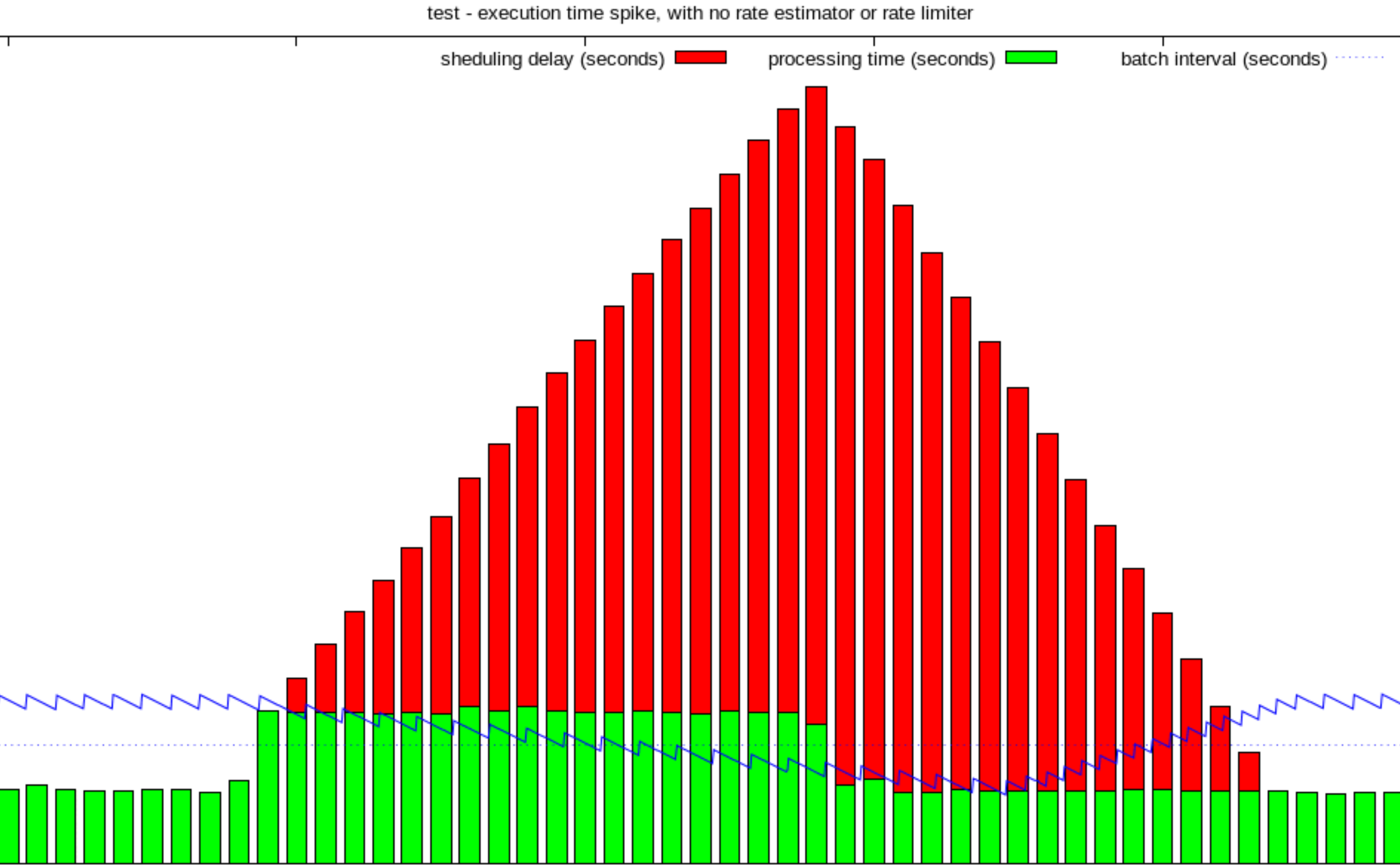
The problem

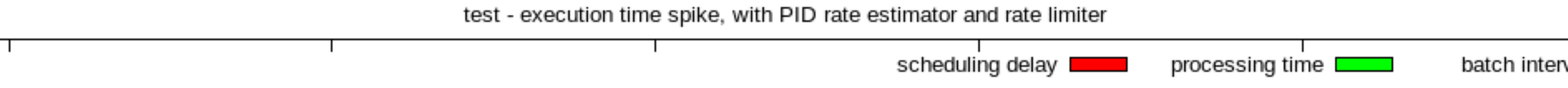
Data may arrive faster than Spark can process it, leading to an unstable system (delays, memory spills). Worst-case, you run out of memory!

Spark Streaming Architecture



If the batch interval is too short, data accumulates in the system


test - execution time spike, with no rate estimator or rate limiter
sheduling delay (seconds)
processing time (seconds)
batch interval (seconds)


test - execution time spike, with PID rate estimator and rate limiter
scheduling delay
processing time
batch interv

Rate Limiting in Spark 1.4

- You could set a **static** rate limit
  - Had to be conservative
  - difficult to find the right limit (depends on cluster size)
  - one limit to rule all streams for all time!

Back-pressure

Back-pressure

- (aka, flow control)
- a slow *consumer* should slow down the *producer*
  - Classic example, TCP
  - New example: Reactive Streams
    - reactive-streams.org

Back-pressure in Spark 1.5

- A *dynamic* rate limiter

1. Estimate the number of elements that can be safely processed by the system in the *next* batch interval

2. Communicate the limit to Receivers

3. Rely on TCP to slow down producers on the other side of the channel

Note: What’s Missing?

- What about other distributed streams?
- Do we want to rely solely on TCP back-pressure?
  - We’ll come back to this…

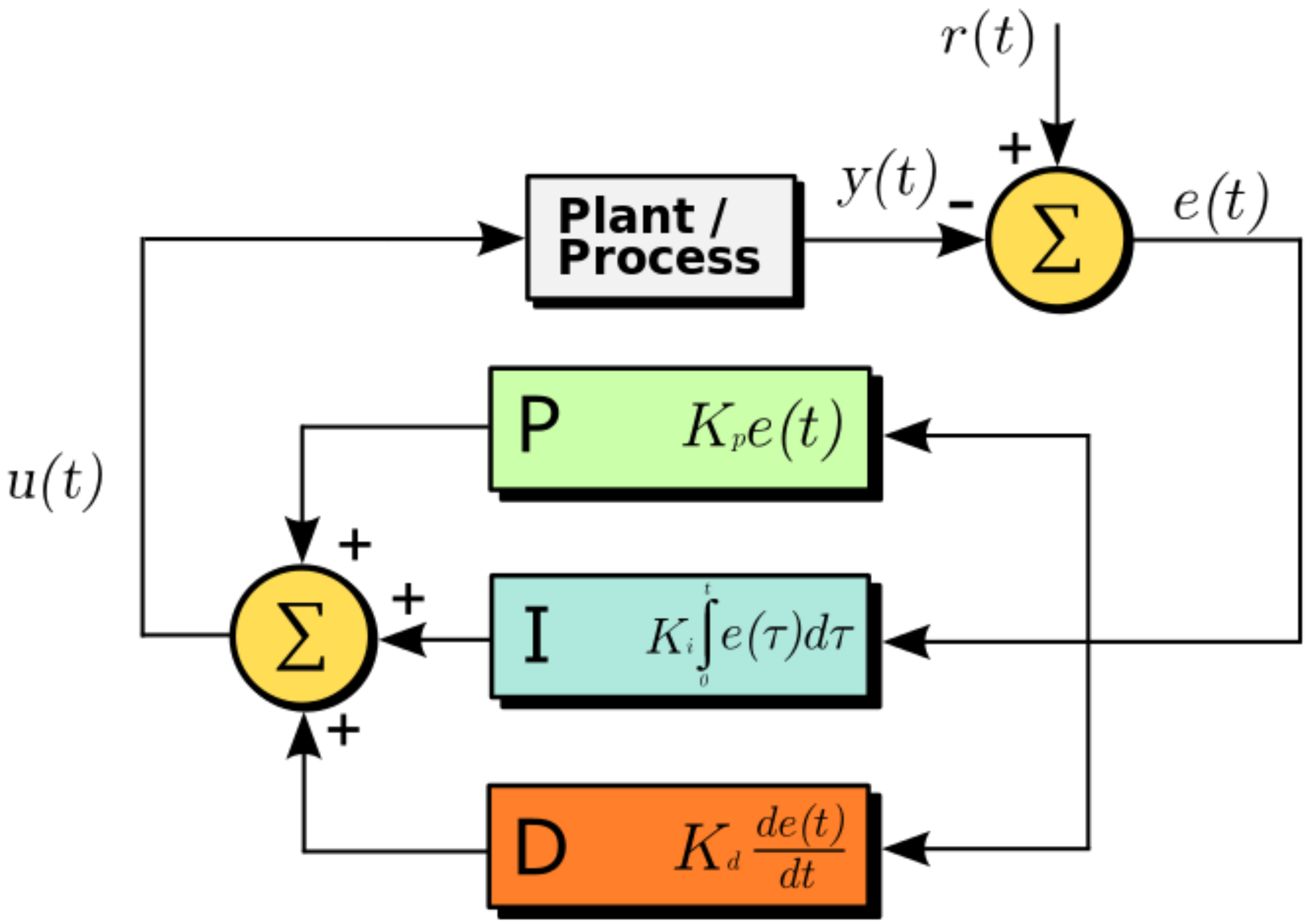
More Details…

- Each `BatchCompleted` event contains
  - processing delay, scheduling delay
  - number of elements in each `InputStream`
- the rate is `elements/processingDelay` (roughly)
  - but what about accumulated data?

PID Controller

- *Proportional, Integrative, Derivative Controller*
- https://en.wikipedia.org/wiki/PID_controller

PID Controller



https://en.wikipedia.org/wiki/PID_controller

PID Controller

- Goal: Keep the processing time close to batch interval
- Proportional term: Use the error:
  - batchInterval - processingTime/batch
- Integral component: schedulingDelay
- Derived component: error - previousError

PID Controller

- Not all 3 terms required.
- Does not require process knowledge (black box)
- P, I, D constants can change convergence, over-shooting, oscillations

To see how the constants affect convergence:

https://upload.wikimedia.org/wikipedia/commons/3/33/PID_Compensation_Animated.gif

Back pressure in Spark 1.5

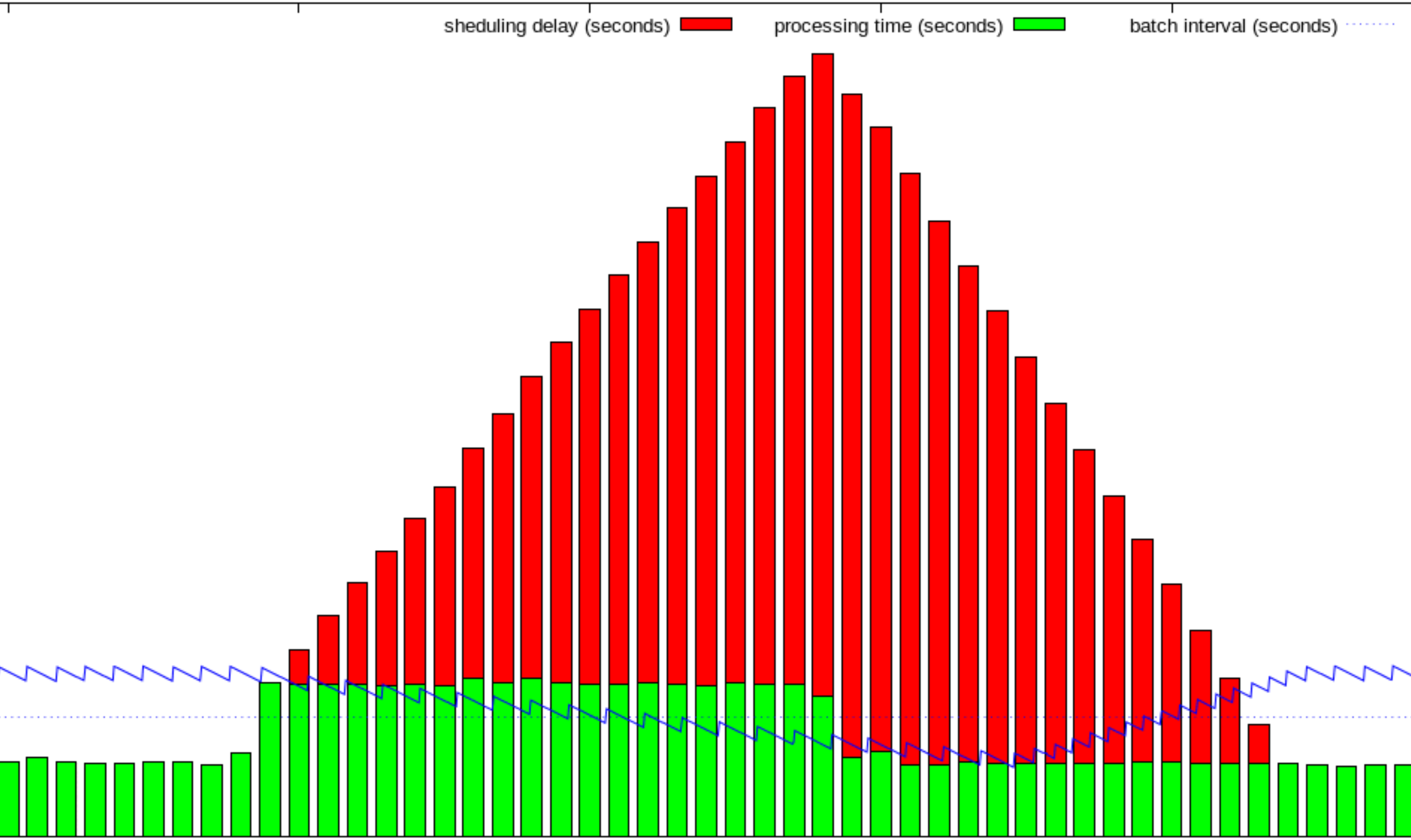
- Each input stream gets its own estimator
- Works with Receiver based inputs, and also KafkaDirectInputStream
- To enable, set the following properties:

```
spark.streaming.backpressure.enabled   true
spark.streaming.backpressure.minRate   R
```

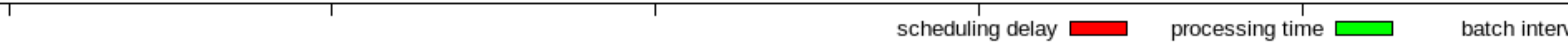
What it does in practice.

Before…

test - execution time spike, with no rate estimator or rate limiter

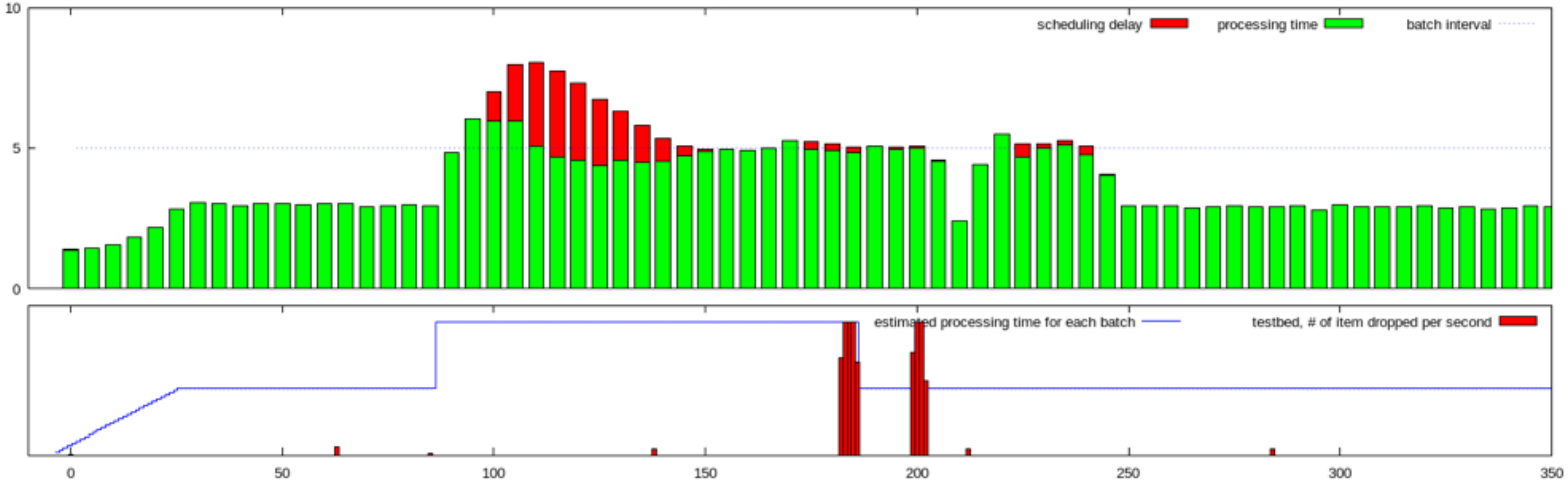


test - execution time spike, with PID rate estimator and rate limiter

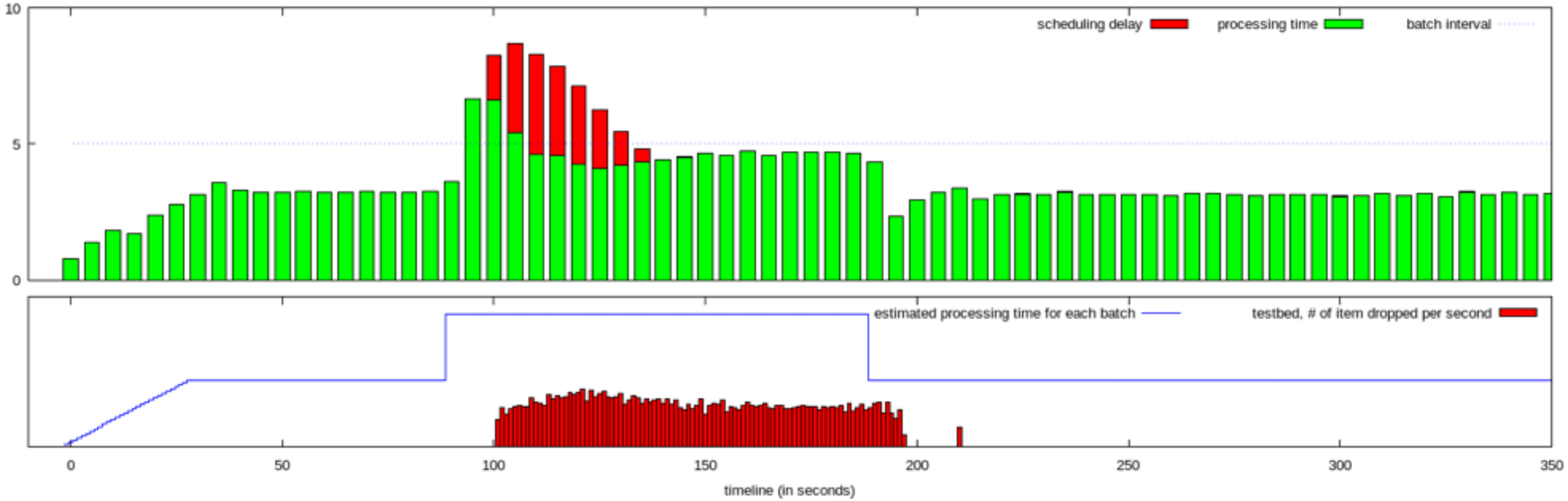


after

test - execution time spike, with PID rate estimator and rate limiter



test - execution time spike, with PID rate estimator and Reactive Stream back pressure



Limitations

- Doesn’t account for records of different sizes
- Linearity assumption not accurate
- Back pressure accumulates data in the TCP channel; what happens to the data congesting at the source end??

A Peek at Reactive Streams

Reactive Streams

- reactive-streams.org
- A standard for asynchronous stream processing
- Consumer controls rate by asking for elements from producers

Reactive Streams

- A push model is used when the consumer can keep up.
- Switches to a pull model when rate limiting required.
- Dynamically switches back and forth…

Reactive Streams… in Spark?

- Hoped to be ready for v1.6, but TBD:
- JIRA 10420 (Reactive Streams Receiver)

Credits

- Typesafe: Luc Bourlier, Iulian Dragos, Nilanjan Raychaudhuri, Dean Wampler
- (formerly Typesafe) François Garillot
- Databricks: Tathagata Das (TD)