Realtime Data Processing at Facebook

Abhay Venkatesh

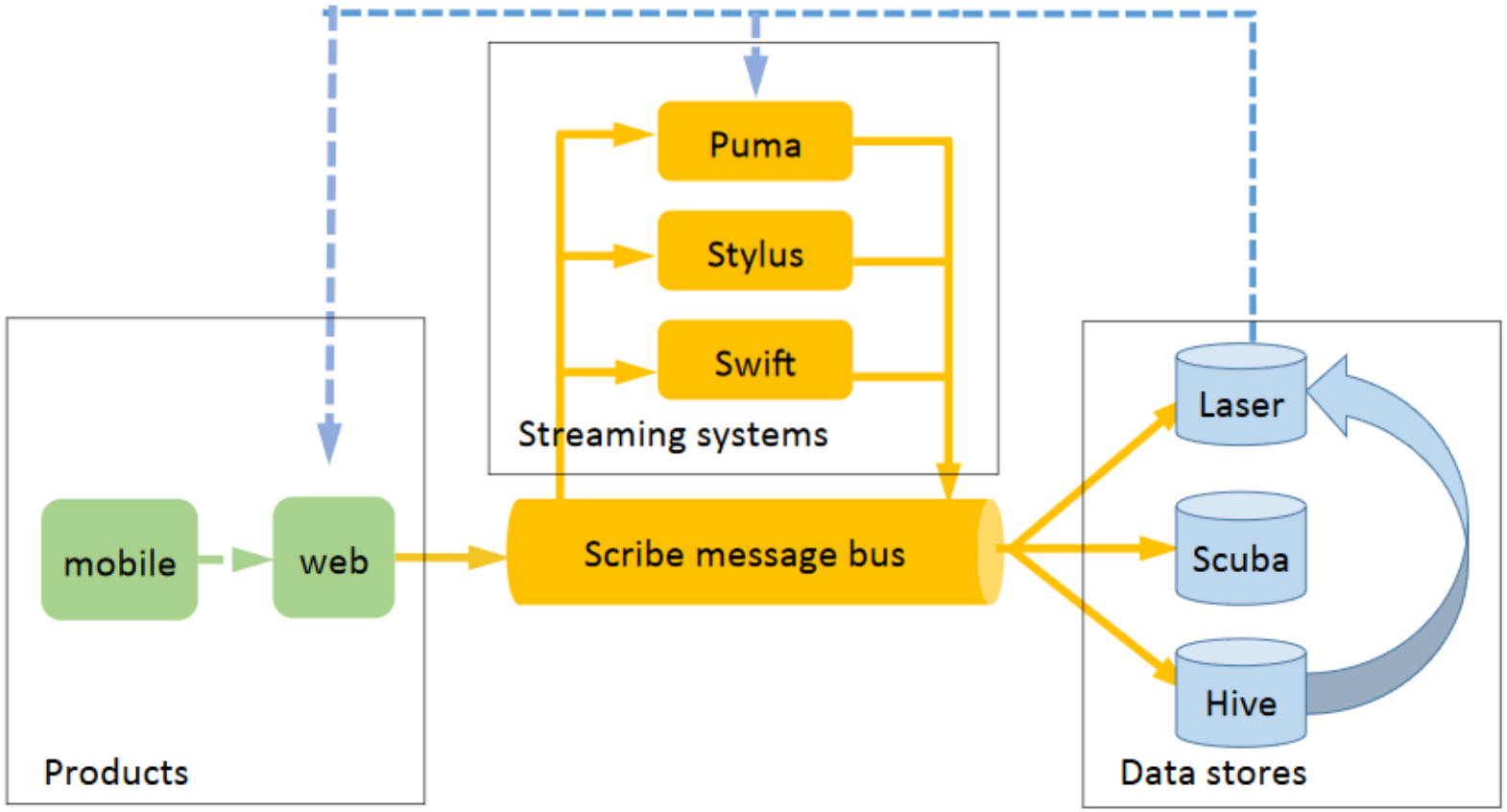
Why Streaming at Facebook?

- Actionable reports
  - e.g. Chorus: what is trending right now?
- Realtime monitoring
  - e.g. dashboard queries
- Hybrid realtime-batch pipelines
  - e.g. pre-emptive queries over data warehouse

Workload Assumptions

- s not ms, which means
  - can use persistent message bus called *Scribe*
  - which makes it easier to enable
    - Fault tolerance
    - Scalability
    - Multiple options for correctness

System Architecture



The Streaming Triad

- Puma
- Swift
- Stylus

Puma

- For apps written in a SQL-like language
- Quick to write (< 1 hour)
- But run over long periods (months to years)
- Two purposes
  - Pre-computed query results for simple aggregation queries
  - Filtering and processing of Scribe streams

A Puma App

```
CREATE APPLICATION top_events;

CREATE INPUT TABLE events_score(
  event_time,
  event,
  category,
  score
)
FROM SCRIBE("events_stream")
TIME event_time;

CREATE TABLE top_events_5min AS
  SELECT
      category,
      event,
      topk(score) AS score
  FROM
      events_score [5 minutes]
```

Swift

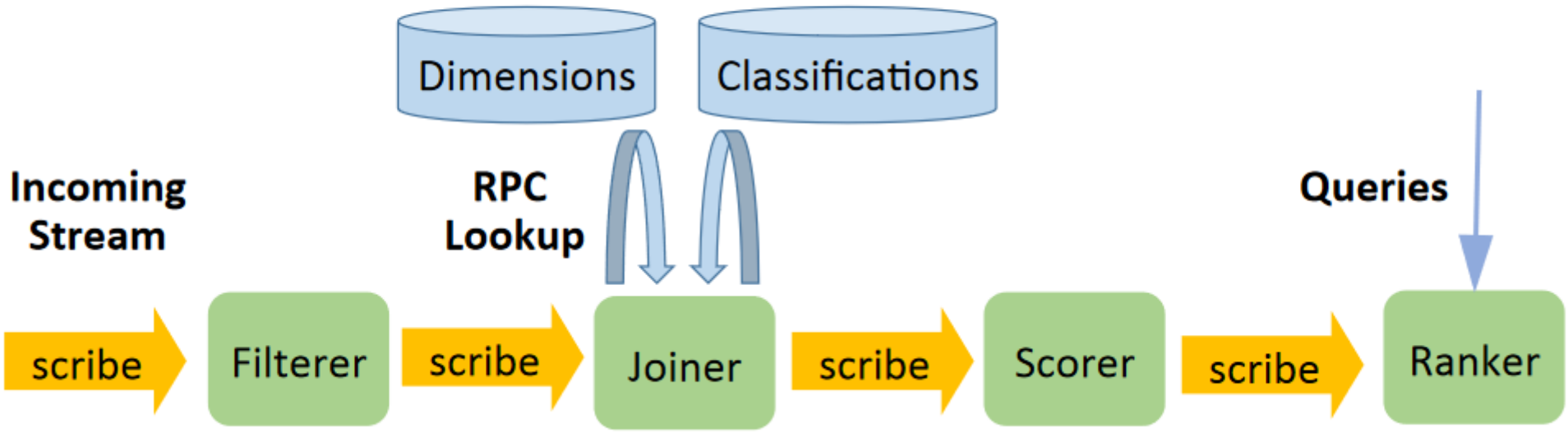
Very Basic API

- Can `read()` from a Scribe Stream
- Checkpoints every
  - N Strings, or
  - B Bytes

Stylus

- Low-Level Stream Processing in C++

Sample Application



Design Decisions

- Language Paradigm
- Data Transfer
- Processing Semantics
- State-saving mechanism
- Reprocessing

Design Decisions

- Language Paradigm
- Data Transfer

- Processing Semantics
- State-saving mechanism
- Reprocessing

Processing Semantics

- At least once, at most once or exactly once
  - State semantics (inputs)
  - Output semantics

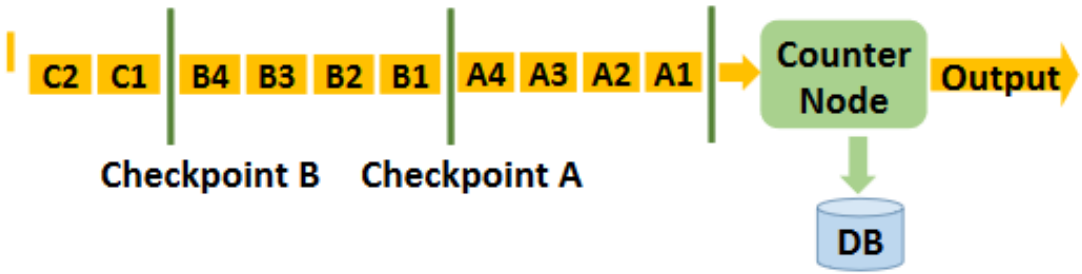
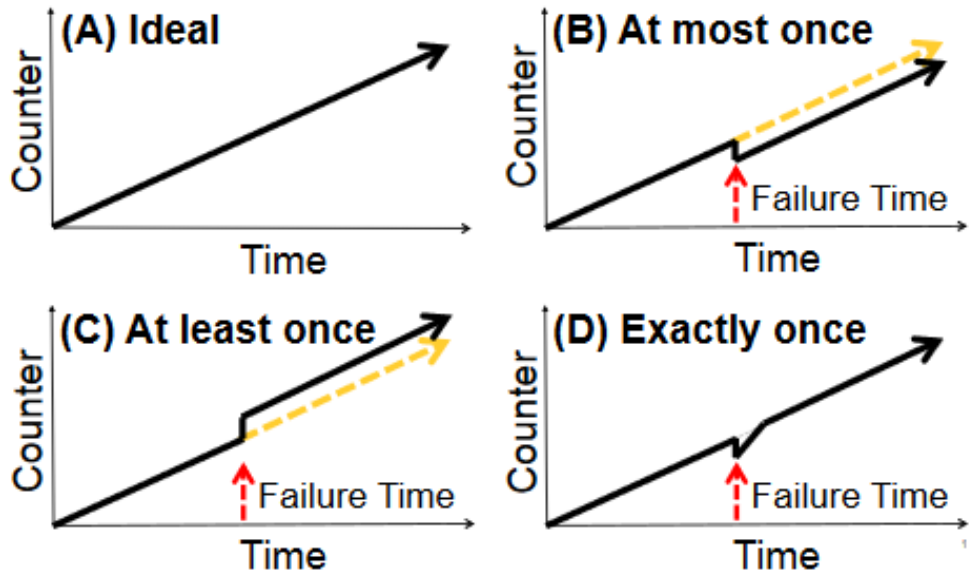
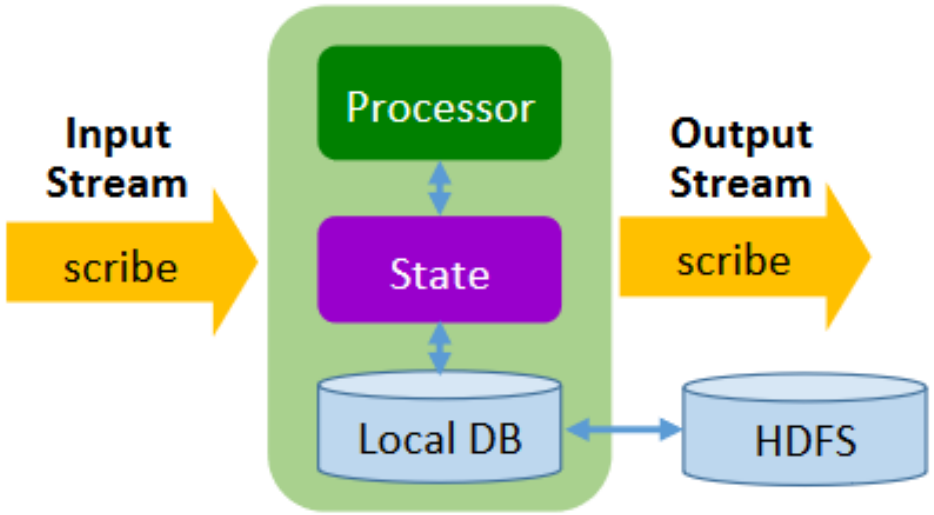
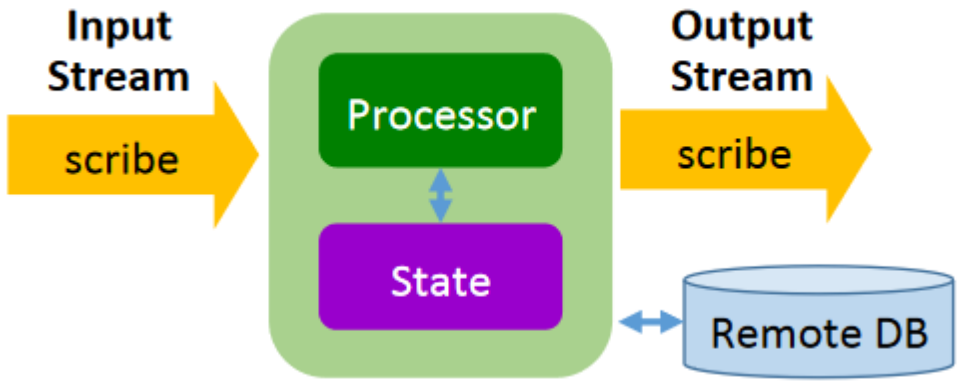


Figure 6: This Counter Node processor counts events from a $(timestamp, event)$ input stream. Every few seconds, it emits the counter value to a $(timewindow, counter)$ output stream.



State-Saving Mechanisms





Reprocessing Data

- Data warehousing with Hive
- Stream processing in batch environment
  - Puma -> Hive
  - Stylus -> stateless, stateful, and monoid

Closing Thoughts

- “Move Fast”
- Ease of debugging
- Ease of deployment
- Ease of monitoring and operation

Comparison with Naiad

Naiad	Facebook Realtime Systems

• Milliseconds, not seconds
  • Robust solutions to micro-stragglers
• Expense availability in event of failure
• Naiad consumes inputs from message queue, and writes to key-value store
	• Seconds, not milliseconds
	  • Does not handle micro-stragglers
	• Persistent message bus ensures no loss
	• Flexible, and easy to use, deploy, debug