Are we solving the core problems in stream processing?

Manpreet Singh
Principal Software Engineer @ Google
manpreet@google.com

Panel at BIRTE 2018
Why stream processing?

- Collection and transformation of data
- Critical for product success
Pillars of stream processing?

- Consistency
- Fault tolerance
- Latency
- Cost
- Service / SLA
- Expressiveness
### Consistency: What it means

<table>
<thead>
<tr>
<th>Databases</th>
<th>Stream Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACID</td>
<td>Exactly Once</td>
</tr>
<tr>
<td>Read-your-writes</td>
<td>Observed state only moves forward</td>
</tr>
<tr>
<td>Can't re-run events and get a different result</td>
<td></td>
</tr>
<tr>
<td>Consistent totals on all reports</td>
<td></td>
</tr>
</tbody>
</table>

**Unbounded input data makes it challenging**
Fault tolerance: What it means for streaming

- Is it alive?
- Is it caught up?
- What kind of failures can it tolerate:
  - Machine
  - ...
  - Data-center

Need to maintain global state on the list of committed events
SLA

- Latency
- Completeness

Example SLA: Max event delay < 3 minutes for 99% time
Who runs the framework code?
- Multi-tenant
- Isolation amongst users
- Business logic isolated from framework

Framework for streams is significantly more complex due to unbounded input.
Scale: How many events per second?

- 10s
- 100s
- Millions
- ...
- Billions

Auto-scale to a new workload?
Cost

- Hardware
- Design complexity
- Production support to run as a service and provide SLOs
Expressiveness: What operations can the user specify?

- Stateless ops (e.g. filter)
- Complex joins
- ...
- Stateful ops (e.g. HAVING)

Flexibility to add more sources and sinks
Can one size fit all?

- Consistency
- Fault Tolerance
- Latency
- Cost
- Scale
- Service
- Expressiveness
Is lambda architecture the right paradigm to address this?

- Use stream processing for best-effort analysis
- Use batch processing for strong consistency, high reliability, etc.

https://en.wikipedia.org/wiki/Lambda_architecture
Today’s panel

- What are the major challenges in building stream systems?
  - Which pillar matters when?
  - Can we build a system that provides all the properties?
  - What are the tradeoffs?

- Does consistency matter for streams?

- Do the traditional solutions from DBMS carryover?

- Is lambda architecture the right paradigm to address this?