Are we solving the core problems in stream processing?

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

Panel at BIRTE 2018
### Consistency: When it matters

<table>
<thead>
<tr>
<th>Inconsistent is Okay</th>
<th>Consistency Matters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximate</td>
<td>Exact</td>
</tr>
<tr>
<td>Transient</td>
<td>Permanent</td>
</tr>
<tr>
<td>For display only</td>
<td>Decisions and side-effects</td>
</tr>
</tbody>
</table>

*Example:* How many tweets about Donald Trump in the last hour?  
*Example:* Charge user's credit card when ad spend reaches $100.
Consistency matters whenever there is a financial implication.
My recent family vacation

Had to wait 8 hours to pick tickets after making payment
My recent family vacation

You will normally receive your digital order within 48 hours!

TOURIST PASS
My recent family vacation

Online hotel reservation committed via FAX to the hotel owner
Availability

99% projects successful
Down 21 hours/quarter
How does Serving handle datacenter outage?

New user query on google.com

Send to the closest datacenter
How does Serving handle datacenter outage?

Send to the closest datacenter

New user query on google.com
How does Serving handle datacenter outage?

New user query on google.com

Redirect to another datacenter

Primary Datacenter

Backup Datacenter
What if we did the same for Stream Processing?

Logs

Primary Datacenter

Stats

Backup Datacenter
What if we did the same for Stream Processing?

Failover: Loss of consistency!
Why is it a hard problem for Stream Processing?

Stream Processing requires global state!
Why is it a hard problem for Stream Processing?

Scale to $O(\text{millions})$ EPS

Latency few minutes
Multi-homed Stream Processing

All datacenters live -- No failovers

Automatic and Transparent Load Balancing
Multi-homed systems at Google: Photon [SIGMOD’13] / Ubiq [VLDB’16]
Service: Hard to retro-fit

- Scale by # users
  - Not only by bytes
- Isolation, prioritization
- Business logic release velocity
Stream processing - Users can pick at most 2!

- Strong consistency (exactly-once)
- Low cost [Sub-linear (#events)]
- Low latency [O(seconds)]
If users need strong consistency

Strong consistency
(exactly-once)

Low cost
[ Sub-linear (#events) ]

Low latency
O(seconds)

==> Latency / Cost tradeoff
If users cannot pay a lot of money

- Weak consistency
- Low cost [Sub-linear (#events)]
- Low latency $O(\text{seconds})$
Many organizations seem to:
- EITHER ignore consistency
- OR address it with higher latency (using Lambda design)

We need to build systems that provide both!

Core properties must hold:
- Scale
- Fault tolerance at datacenter-level
- Cost
- Expressiveness
- Service