Latency, Damned Latency, and Streaming

Speaker: Jonathan Goldstein

Microsoft Research

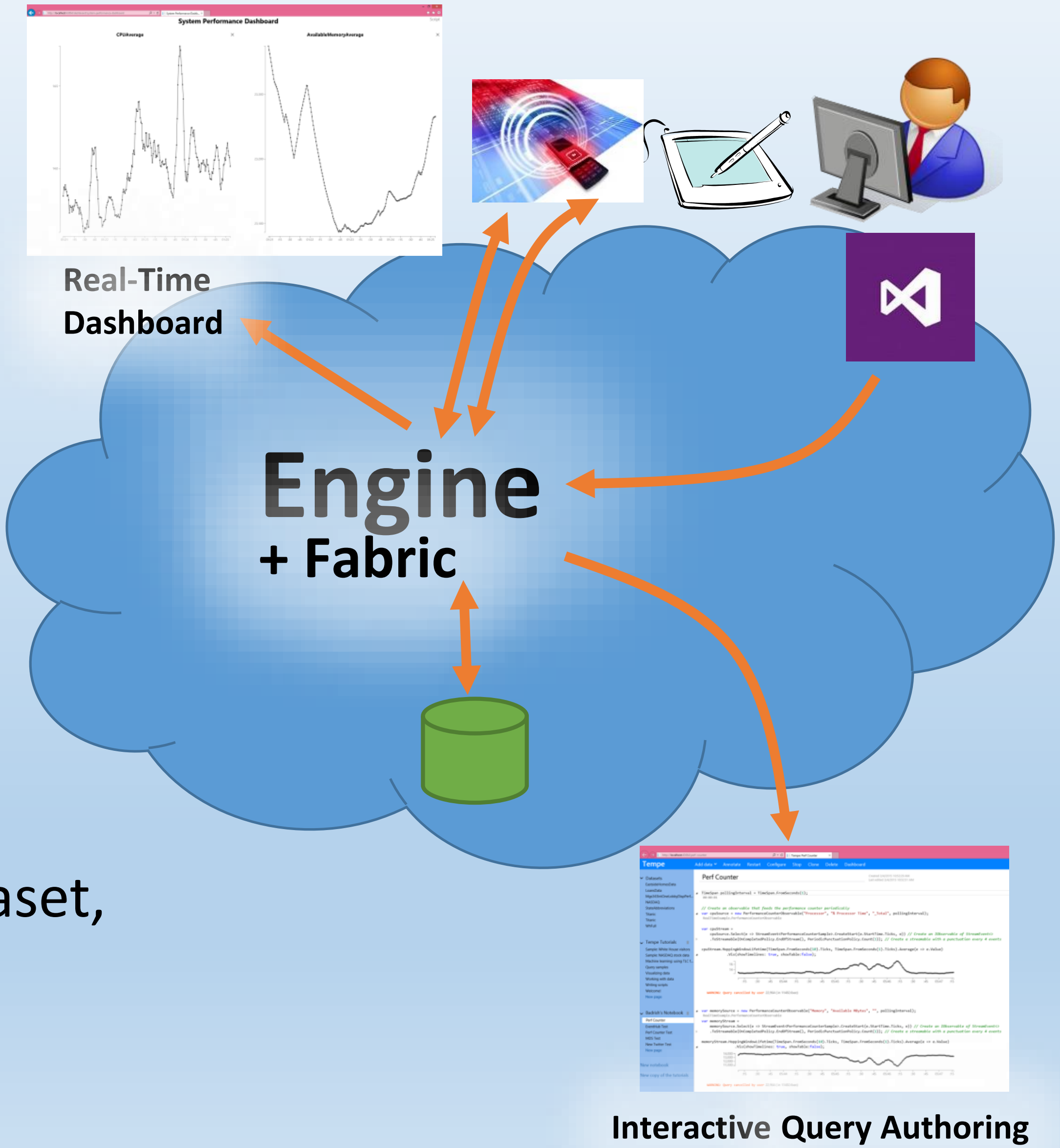
This talk incorporates insights from 8 years of research and product development, with too many valued contributers to list, but a special callout to:

Badrish Chandramouli

Who has been my colleague and research partner on this journey

Diverse Scenarios for Analytics

- Real-time
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected
- Real-time with historical
  - Correlate live data stream with historical activity (e.g., from 1 week back)
- Offline
  - Develop initial monitoring query using logs
  - Back-test monitoring query over historical logs
- Progressive
  - Non-temporal analysis (e.g., BI) over large dataset, stream data, get quick approximate results



Diverse Scenarios for Analytics

Python

SQL Hadoop/PIG

- Real-time
  - Monitor app telemetry (e.g., ad clicks) & raise alerts when problems are detected

R Splunk

- Real-time with historical
  - Correlate live data stream with historical activity (e.g., from 1 week back)

Kinesis

Matlab

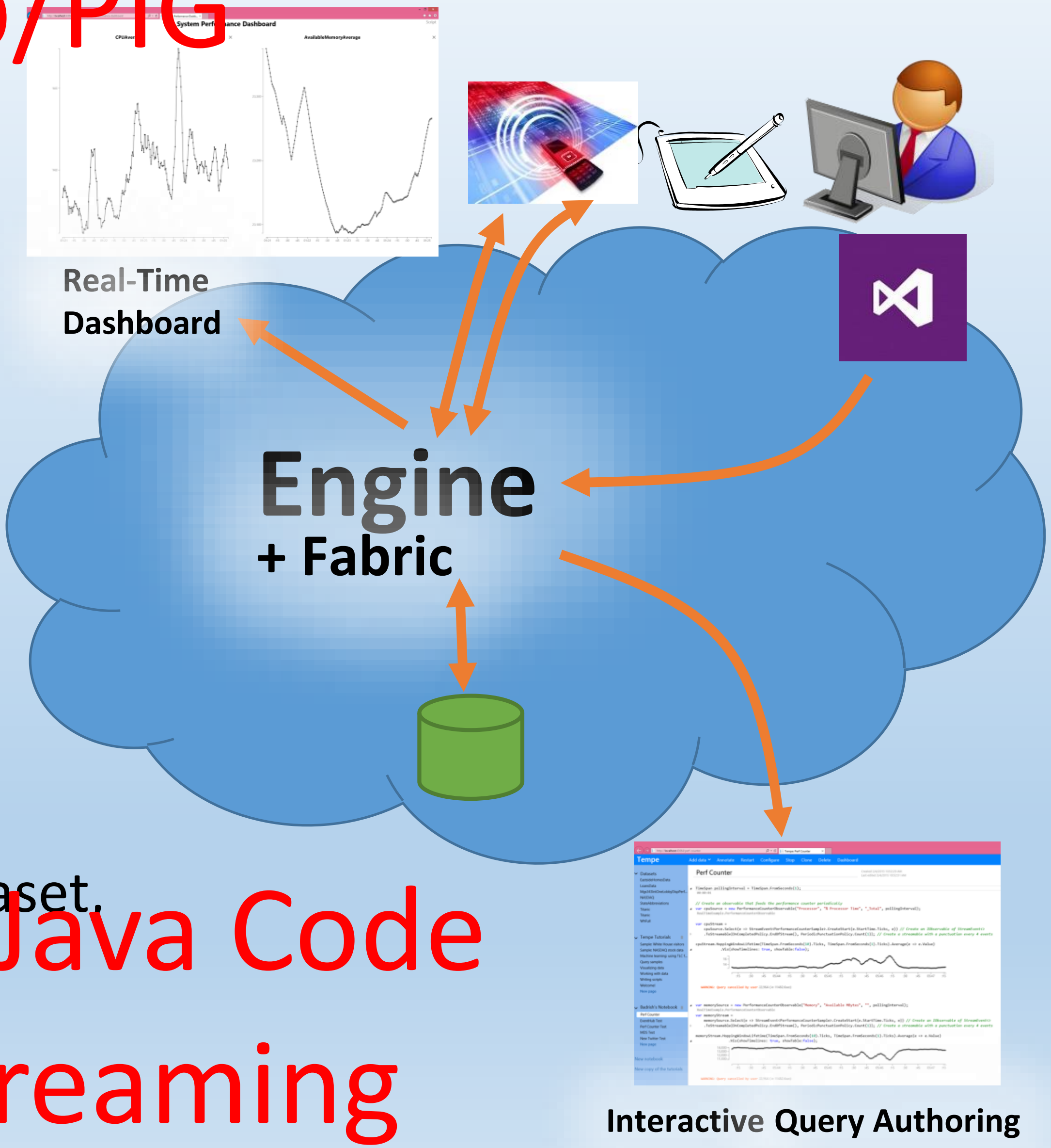
- Offline
  - Develop initial monitoring query using logs
  - Back-test monitoring query over historical logs

Storm Tableau

- Progressive
  - Non-temporal analysis (e.g., BI) over large dataset, stream data, get quick approximate results

Custom Java Code

Spark/Spark SQL/Spark Streaming



What a mess!

- Different tools made by different folks:
  - No agreement on what data is
  - No agreement on what logic/queries are
  - Real application developers pay the price

It doesn’t need to be this way

Pieces of the Data Analytics Problem

- Data Movement
  - Lots of interesting issues but I won’t say much more
- Computation

These problems aren’t as different as they seem!

Enter Trill (A Trillion Events Per Day Per Node)

- What makes Trill special?
  - Time as a first class citizen
  - Adaptability to a variety of settings
  - Performance, Performance, Performance
    - 2-4 **orders of magnitude** faster than traditional SPEs
    - Comparable to commercial column stores for offline

Trill's Use Cases

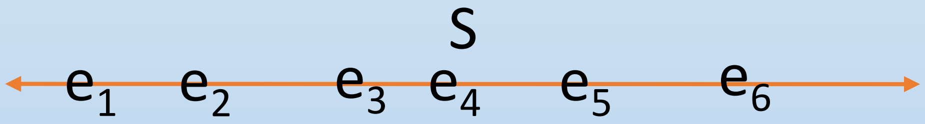
- Azure Stream Analytics Cloud service
- With Scope for Bing Ads
- With Orleans for Halo game monitoring & debugging
- …

Time as a First Class Citizen

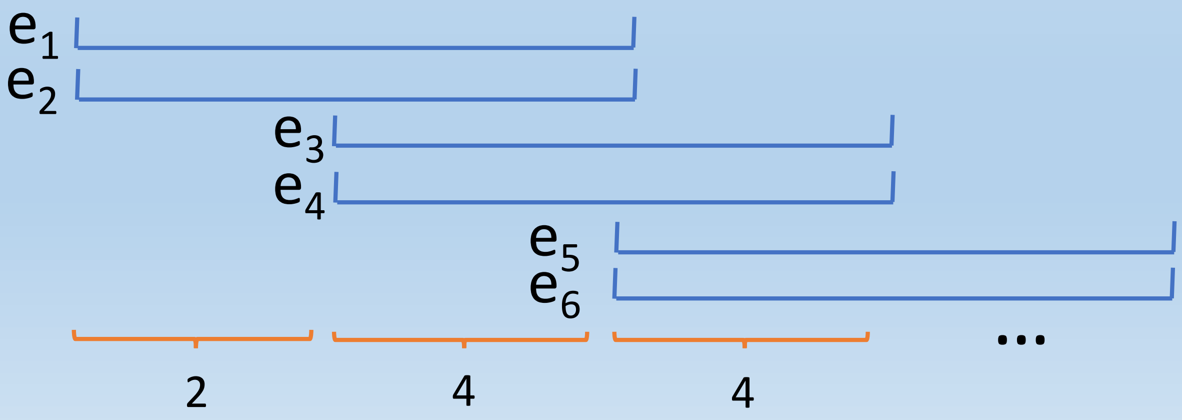
- Low latency analytics:
  - Computation that is periodically repeated over a recent subset of data
    - Business Reports, Dashboards, Model Training
  - Windows capture a critical aspect of semantics
    - Window – All data used for one reporting period
    - A timestamp in the data is needed to decide window membership
    - Very helpful for windowing to be part of the query language
    - Faking windows with groupby isn't enough
      - Hopping Windows, Session Windows
  - Nice algorithms for windowed computation when data arrives with bounded disorder
    - Incremental
    - Bounded state

Time as a First Class Citizen

- For instance, consider the CEDR algebra:
  - Rows are tagged with a contribution time interval
  - This time interval can be manipulated using new operators (e.g. HoppingWindow)

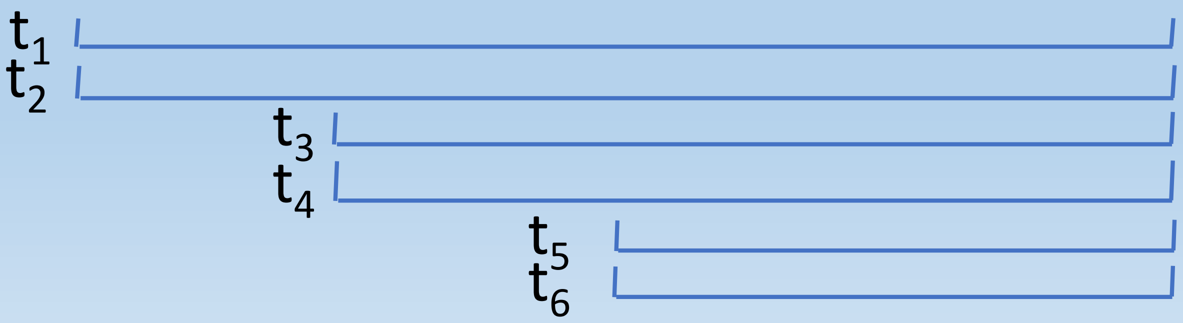


S.HoppingWindow(Hop, WindowSize).Count()



Time as a First Class Citizen

- So windows are about supporting low latency, right?
  - What about offline log analytics?
    - Pattern matching: The order of events is critical
    - Developing and debugging streaming queries
    - Be careful to log and use app time for everything, including real-time
  - What about conventional analytics with really large datasets?
    - Early answers can be useful, but what do intermediate results mean?
    - Couldn't we use something like the windowing trick and timestamps to define exactly which answers should be produced when, and based on which data?



Variety of Settings

- Customers want to use a query processor with varying infrastructure:
  - For real time (e.g. with Orleans)
  - For scaled out offline (e.g. with Map-Reduce & in-mem progeny)
  - For interactive data analysis applications (in Tempe)
- Customers find limiting data types (e.g. SQL) extremely, well, limiting
  - They want to store collections
  - Sometimes they even want to store classes with references in them!

Variety of Settings

- Solution: Trill is a passive library in a modern language (Trill uses .NET)
  - Easy to embed in any part of any .NET application
  - Payloads can contain any .NET type
  - LinQ is our query language (more later)
  - Data ingress and egress using IEnumerable and IObservable
  - Must be able to checkpoint and restore its own state

Performance, Performance, Performance

- One size fits all analytics QP requires competitive performance across a wide spectrum of analytics
- Column stores are fast!
- Really fast!
  - Orders of magnitude faster than traditional stream processors at relational queries!
- What are we going to do?

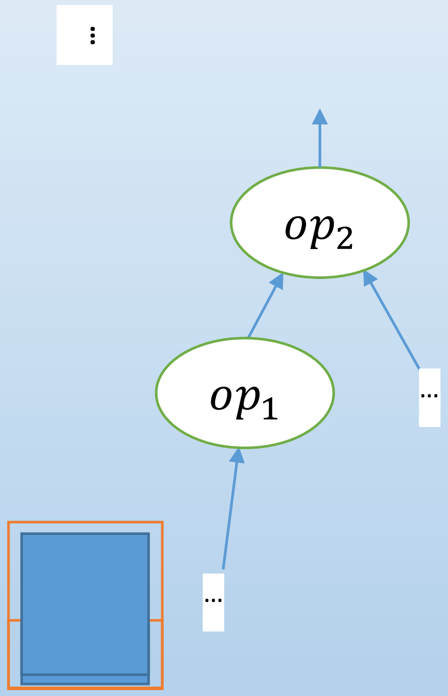
Performance, Performance, Performance

- Column stores have the answer:

Teach a streaming system how to do column store tricks!

Performance, Performance, Performance

- Data organized as stream of batches
  - Purely physical (no impact on query results)
- Users specify latency constraint (10 secs)
  - Batch up to 10 secs of data
  - Small batches → low latency
  - Large batches → high throughput
  - More load → larger batches → better throughput



+ Columnar

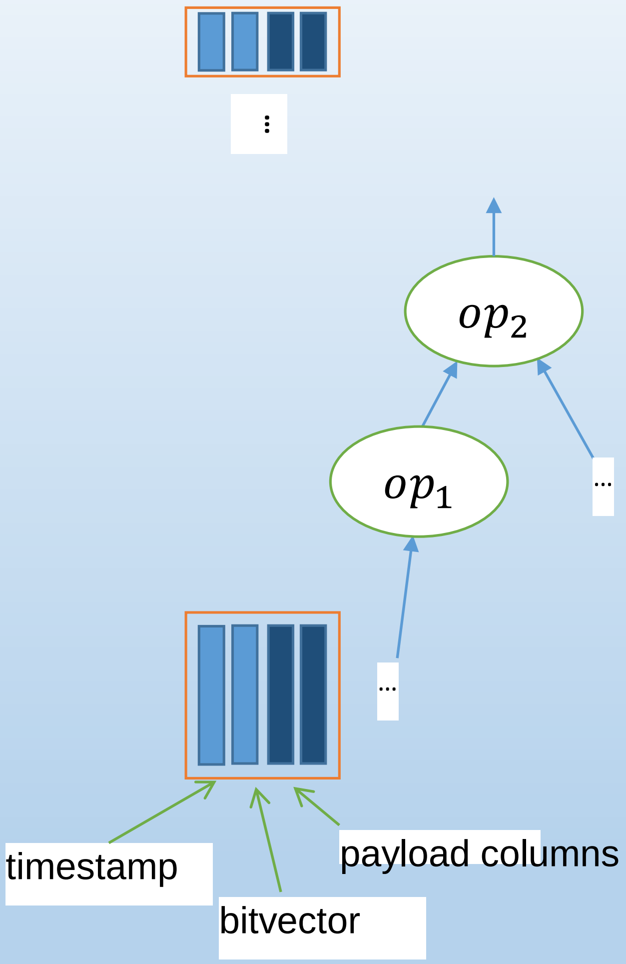
- Columnar format within each batch
  - Timestamps as arrays
  - Bitvector to indicate row absence

```
class DataBatch {
        long[] SyncTime;
        ...
        Bitvector BV;
}
```

  - One array per payload field

```
class UserData_Gen : DataBatch {
        long[] c_ClickTime;
        long[] c_User;
        long[] c_AdId;
}
```

  - Enables efficient QP & serialization

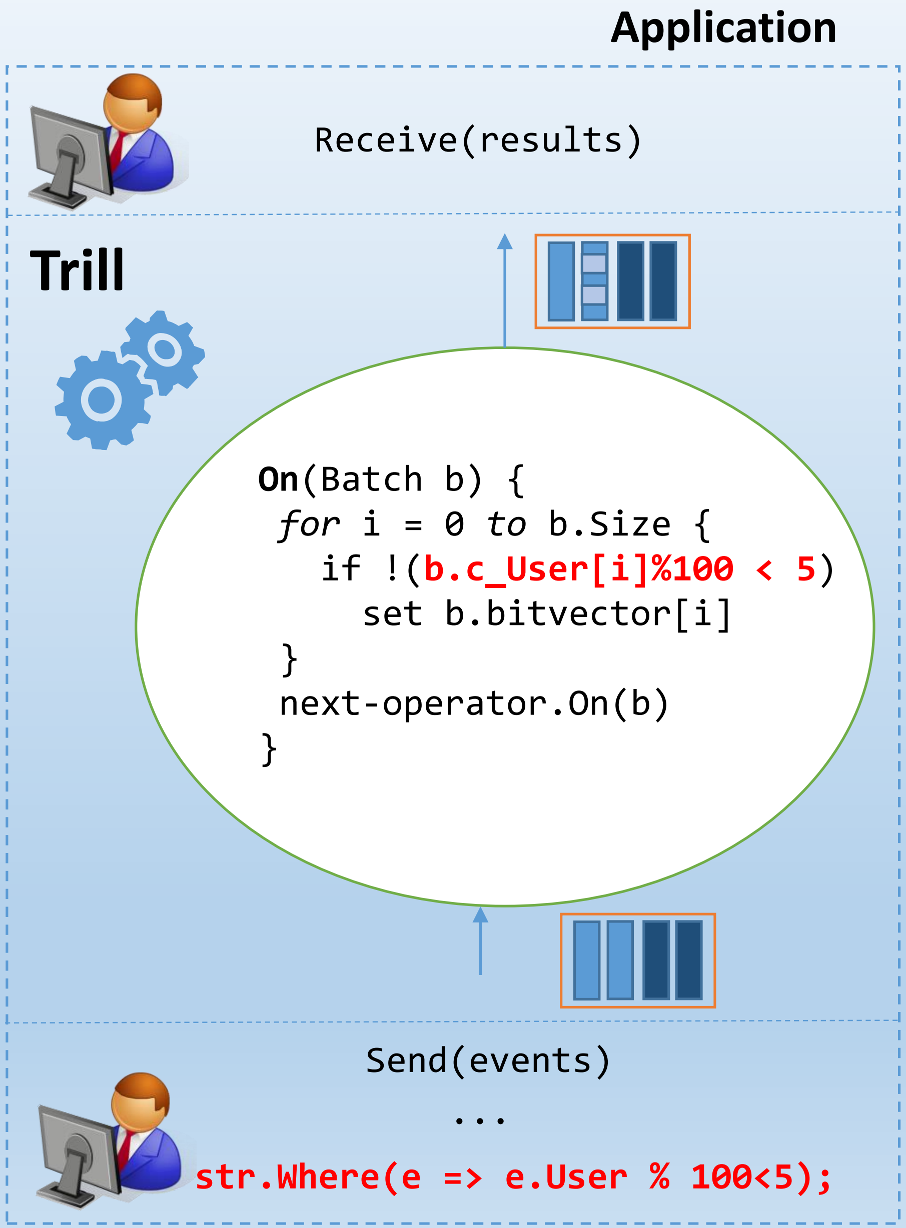


+Made Invisible and Fast By Code Generation

- User view is row-oriented
  - Dynamically generate and compile code for operators and batches
  - LinQ gives users static type checking at query composition time and intellisense
- E.g. Filter (where)

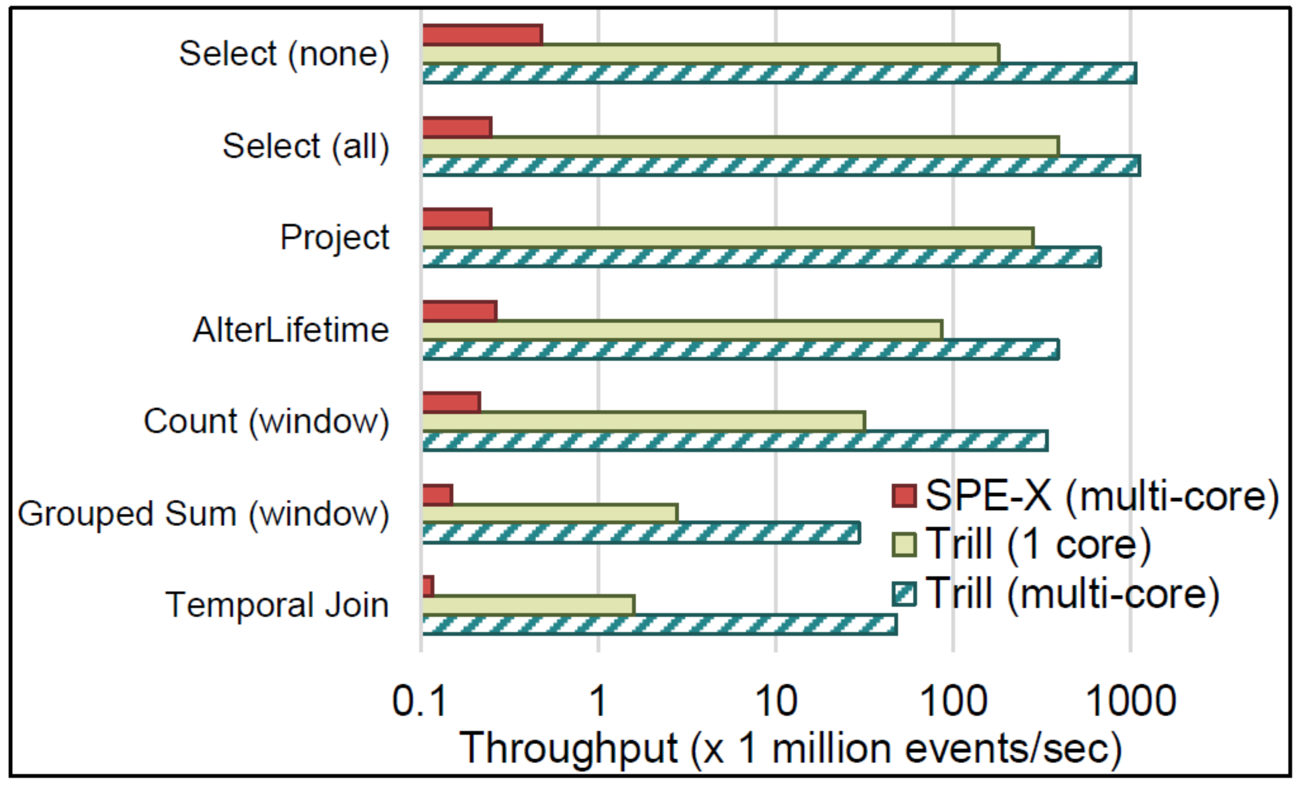
```
str.Where(e => e.User % 100 < 5)
```

- Codegen goals:
  - Tight loops over batches
  - Avoid method calls within loops
  - Columns accessed only if needed



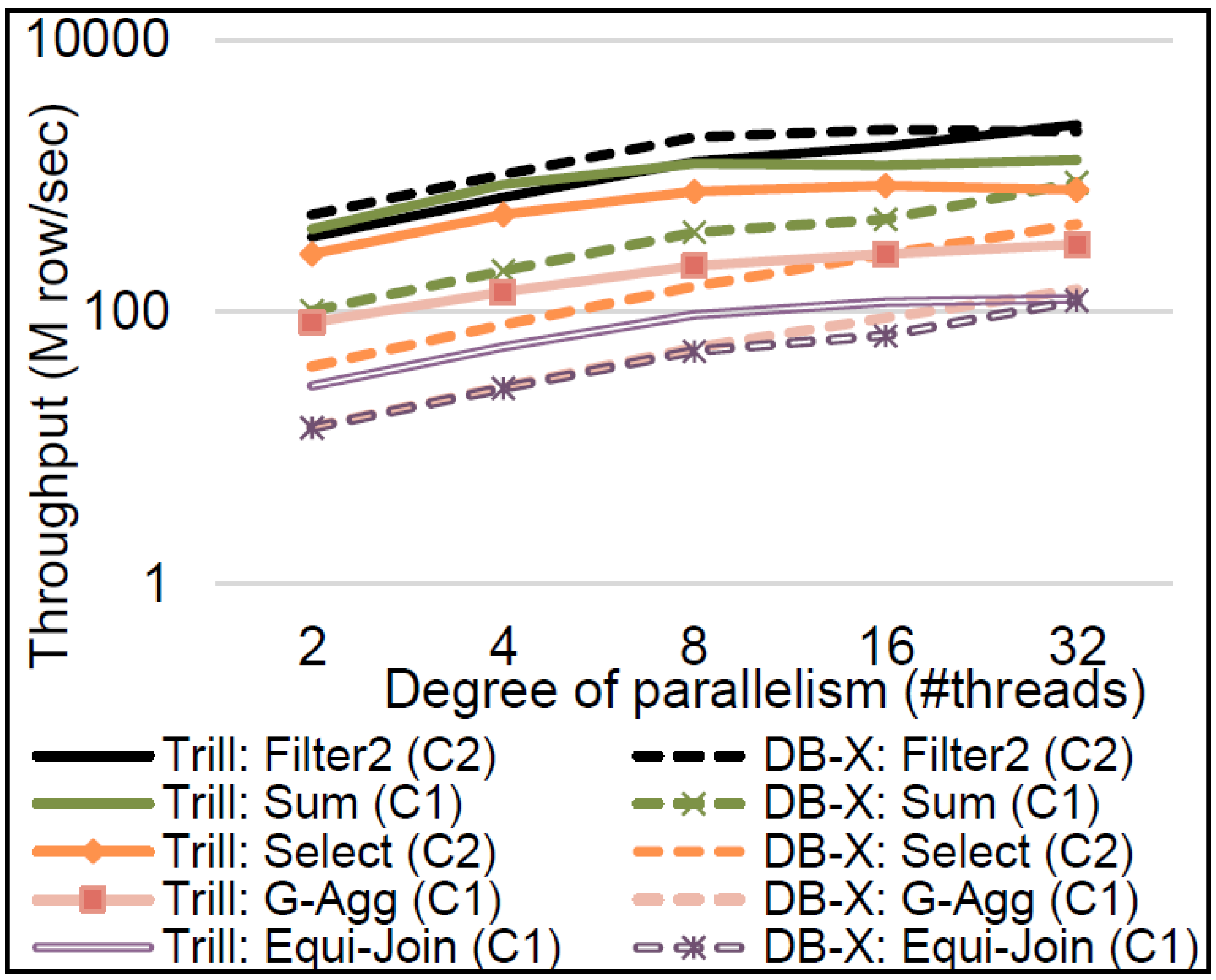
Evaluation (sample)

- Pre-loaded datasets in main memory
- 16-core machine
- Temporal queries



Evaluation (sample)

- Pre-loaded datasets in main memory
- 16-core machine
- Relational queries



Conclusions

- Low latency analytics isn't just niche streaming products
- Rather, it's an opportunity to expand the scope of analytics
- Current cloud applications are in desperate need of this unification
- Trill is a widely deployed engine used across the whole analytics spectrum.
  - Handles Time Powerfully: Covers SQL, Real-Time, Log analysis, Early answers, etc…
  - Easily deployed in any .NET app: Trill is a passive .NET library
  - Fast: Best of breed performance or better across the analytics spectrum

Where Do We Go From Here?

- How about the data movement part?
  - There's actually been a lot of work in this area (Event Hub, Kinesis, Storm, Kafka, etc…)
  - Lots of work to do, but very actionable

- How about OLTP? Can OLTP be effectively fused with modern analytics?
  - What would it mean?
  - Can it be done while matching Trill's performance and expressiveness?

Publications Describing the Presented Ideas

- The original CEDR paper (skip the 1st half):

Consistent Streaming Through Time: A Vision for Event Stream Processing. CIDR 2007

- CEDR and map-reduce system for expressing both offline and online queries:

Temporal Analytics on Big Data for Web Advertising. ICDE 2012

- CEDR and scaled out system for early answers:

Scalable Progressive Analytics on Big Data in the Cloud. PVLDB 6(14)

- Trill:

Trill: A High-Performance Incremental Query Processor for Diverse Analytics. PVLDB 8(4)