Multi-Query Optimization in Wide-Area Streaming Analytics

Albert Jonathan, Abhishek Chandra, Jon Weissman

University of Minnesota

University of Minnesota

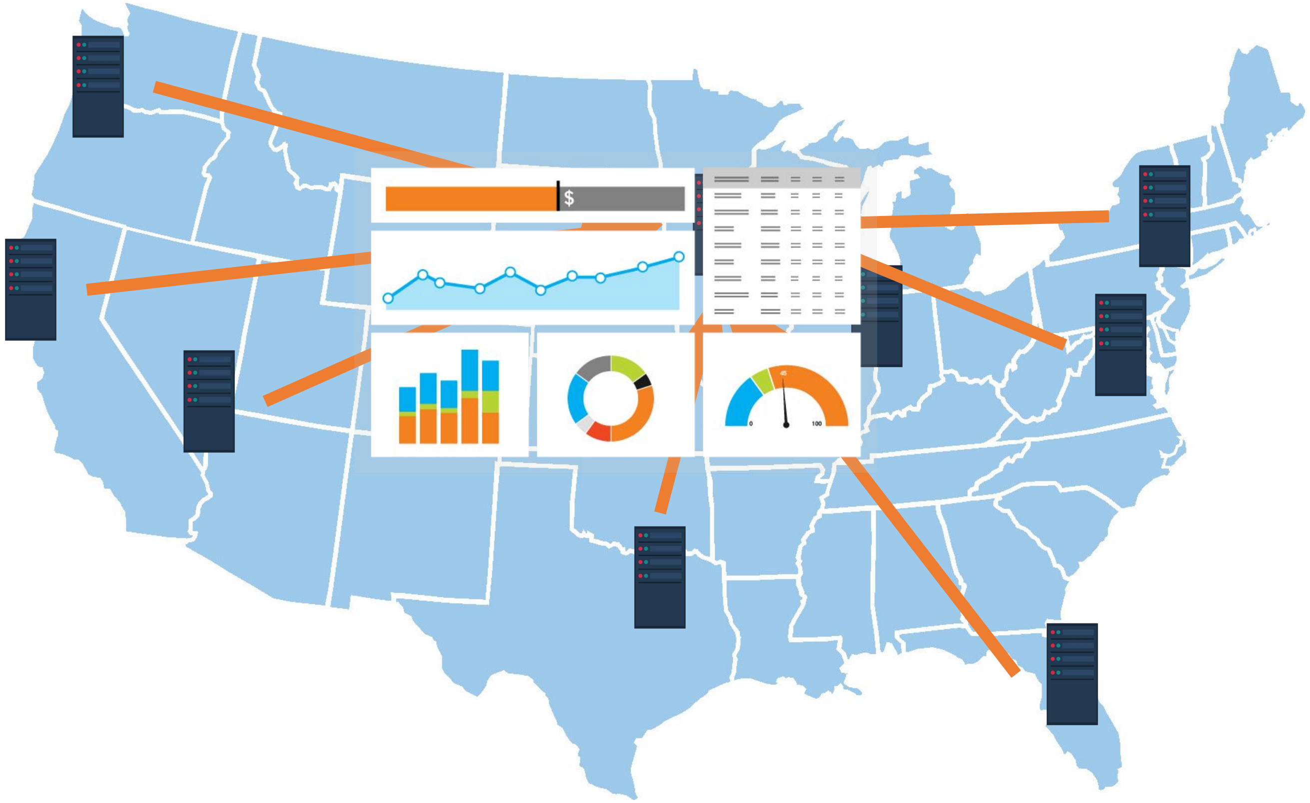
Driven to Discover[SM]

DCSG

Distributed Computing Systems Group

Wide-Area Streaming Analytics

Real-time analysis over large continuous data streams generated at the edge



Trending topic analysis

Location-based advertisement

Meeting Internet service SLAs

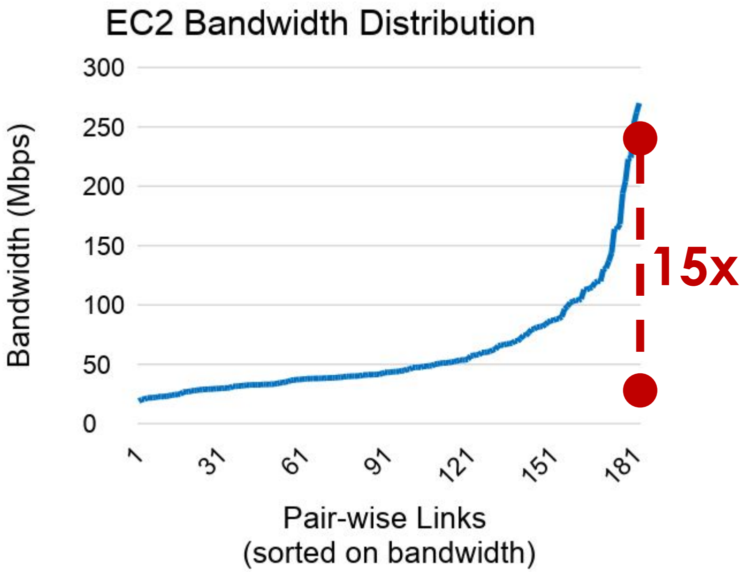
Billing dashboard

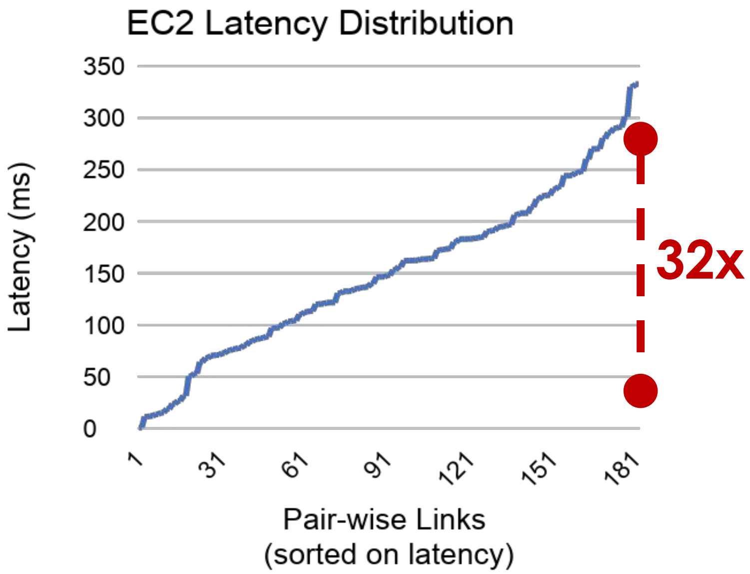
Real-time traffic control

Live video analysis

WAN Resource Demand vs. Constraints

- High resource demand:
  - Twitter, on average 6000 tweets/second (2016)
  - Facebook log updates, 25TB/day (2009)
  - Video surveillance, millions of cameras around large cities, ~3Mbps/camera (2009)
- WAN constraints:
  - Scarce bandwidth
  - High latency
  - Highly heterogeneous
  - Expensive ($$$)





Optimizing Queries Under WAN Constraints

- Existing approaches optimize each query *individually*
  - Delay ⇔ WAN Traffic [Heintz et al., HPDC’15]
  - Delay ⇔ Accuracy/Quality [JetStream-NSDI’14, Heintz et al., SoCC’16, AWStream-SIGCOMM’18]
- Multi-tenancy of streaming systems

“In production environment, the same streaming system is used by many teams.”

  - Social network: trending topic, sentiment analysis, advertisement, campaign
  - CDN Logs: monitored for performance optimization, debugging, billing
- Optimizing multiple queries to handle WAN constraints

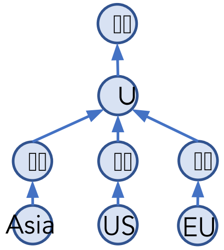
Optimizing Multiple Streaming Queries in Wide-Area Settings

- Borrow the idea of multi-query optimization (MQO) from DBMS
  - Identify commonalities (data, work) between queries → remove redundancies
- Adaptation for streaming analytics workload
  - Long-running (24x7) → incrementally optimize at runtime
  - Latency sensitive → minimal interruption to existing queries
- Adaptation to wide-area settings
  - Heterogeneous, limited bandwidth → WAN-awareness

Benefit of MQO in Wide-Area Streaming Analytics

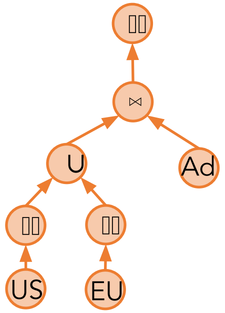
Query 1:

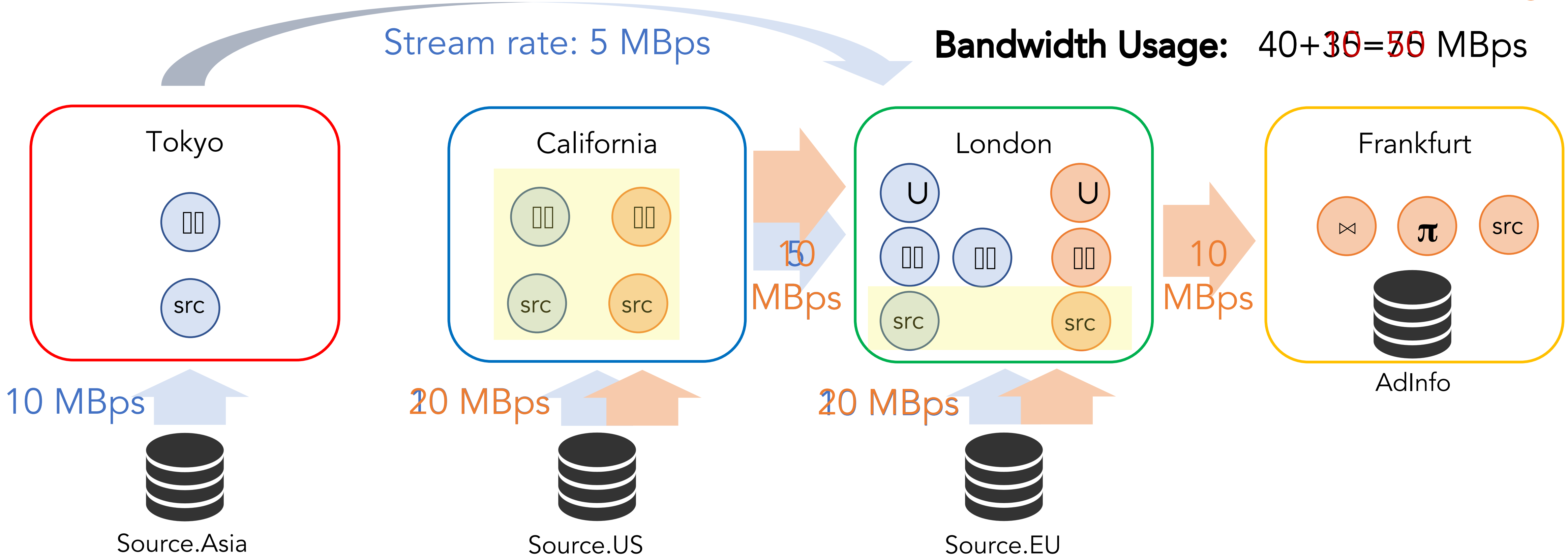
```
SELECT    Time, Topic, COUNT(*)
FROM      Src.US, Src.EU, Src.Asia
GROUP BY WINDOW(Time.Minutes(1)), Topic
HAVING COUNT(*) > 100
```



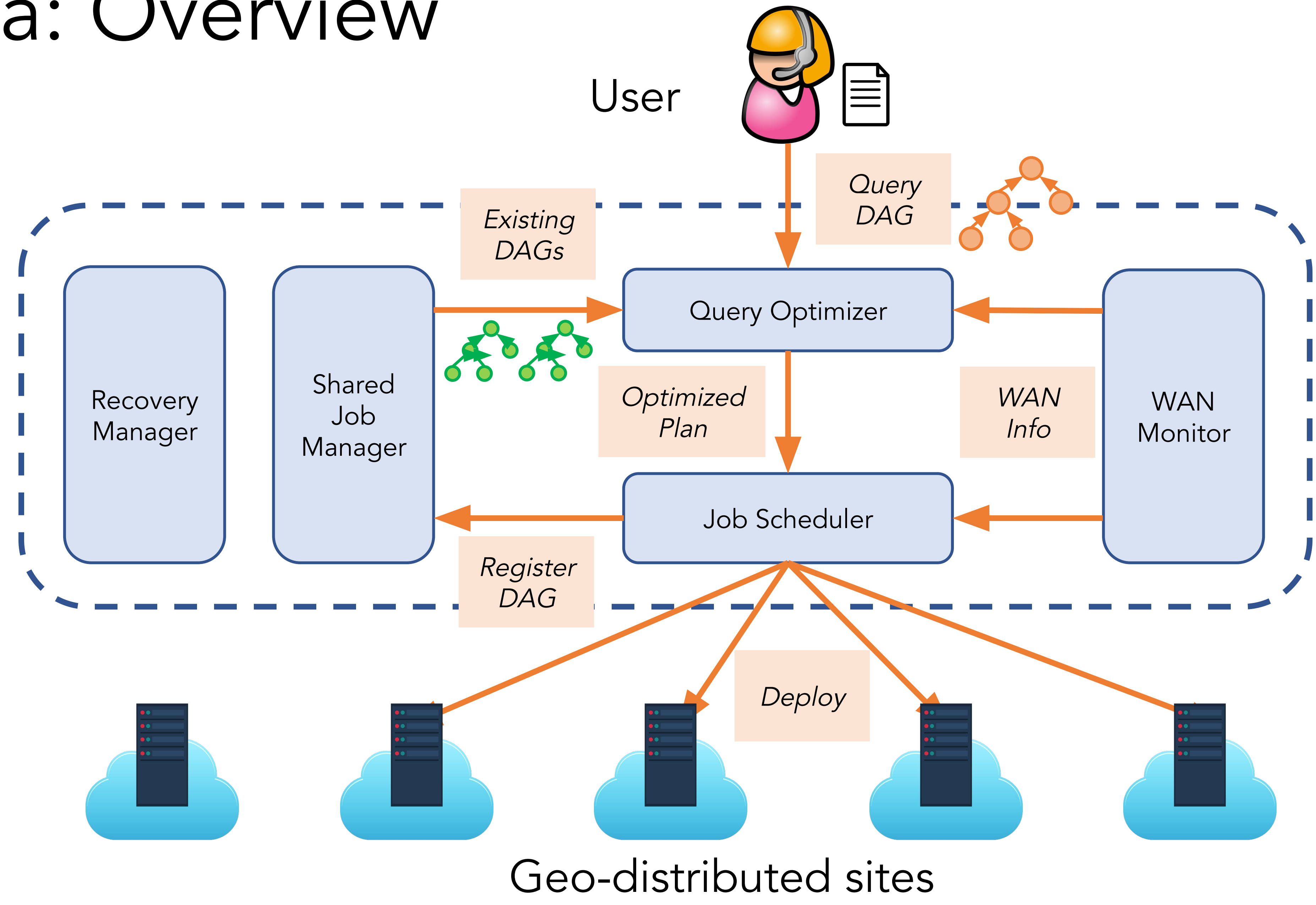
Query 2:

```
SELECT Time, AdInfo.Campaign
FROM   (SELECT  Time, Topic
        FROM    Src.US, Src.EU
        GROUP BY WINDOW(Time.Minutes(1)), Topic
        HAVING COUNT(*) > 100) AS Tweet, AdInfo
WHERE  AdInfo.Topic = Tweet.Topic
```



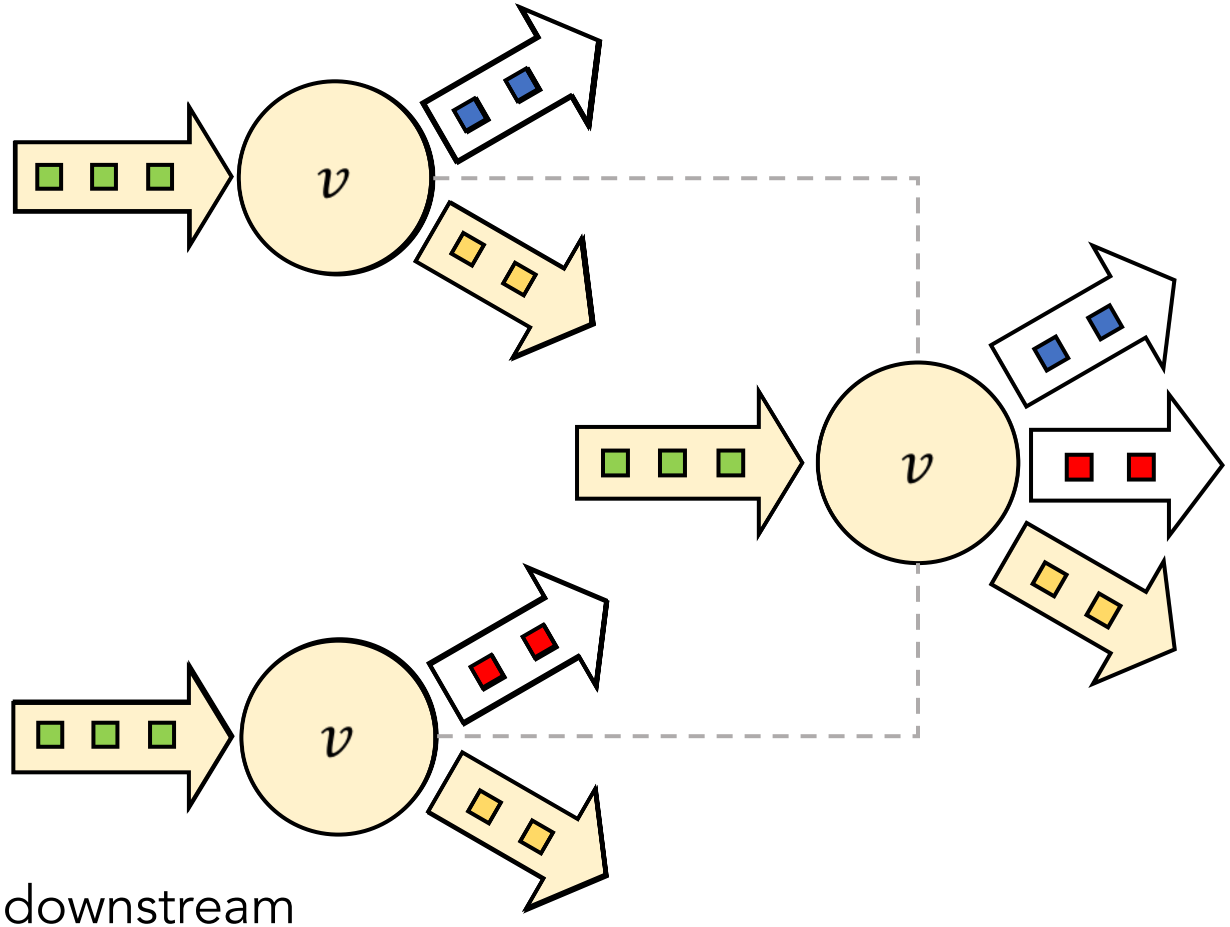


Sana: Overview



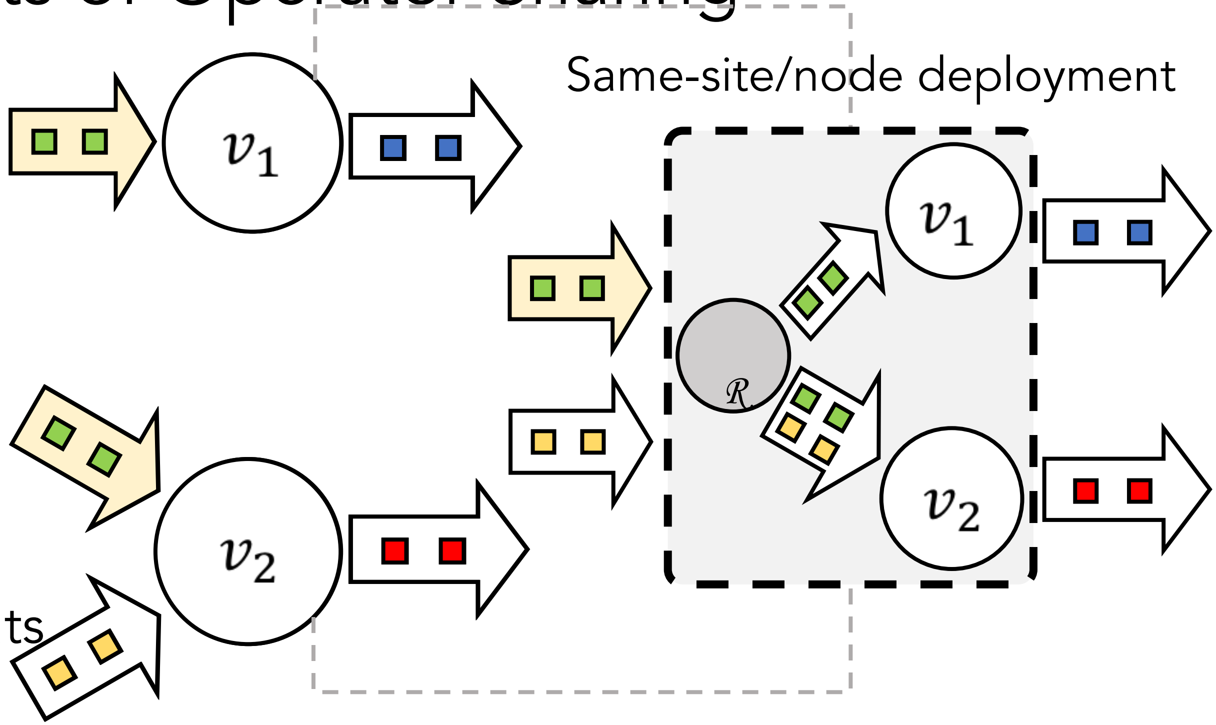
Operator Sharing

- Vertices can share operators *iff:*
  - They share the same stream operator
  - All of their inputs are the same
- Eliminate redundancies in
  - Input streams
  - Data processing
  - Output streams
- Strict sharing requirement
  - Less common for vertices that are further downstream



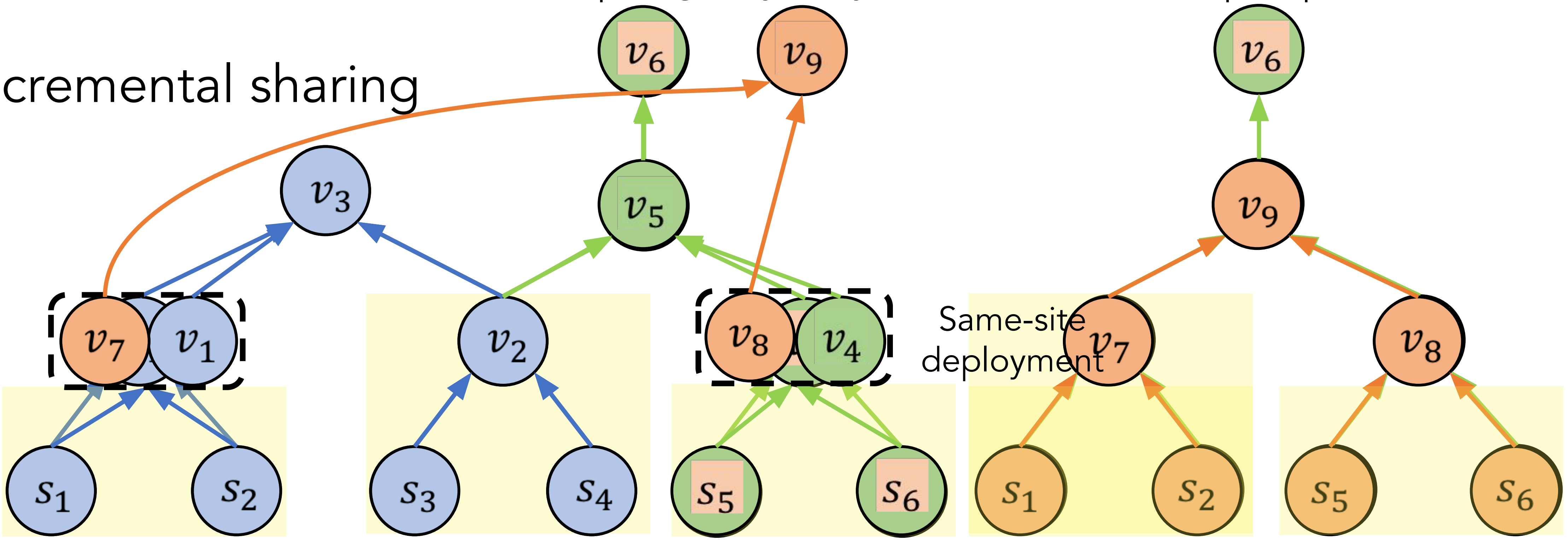
(Partial) Input-Only Sharing

- Relax the strict-equality constraints of Operator Sharing
  - Operators do not have to be the same
  - Can share partial input streams
- *Router* operator
  - Does not perform any data transformation
  - Routes input streams to multiple vertices within a site/node
  - Only added to operators with remote inputs
- Eliminate redundant input streams transmitted over the WAN



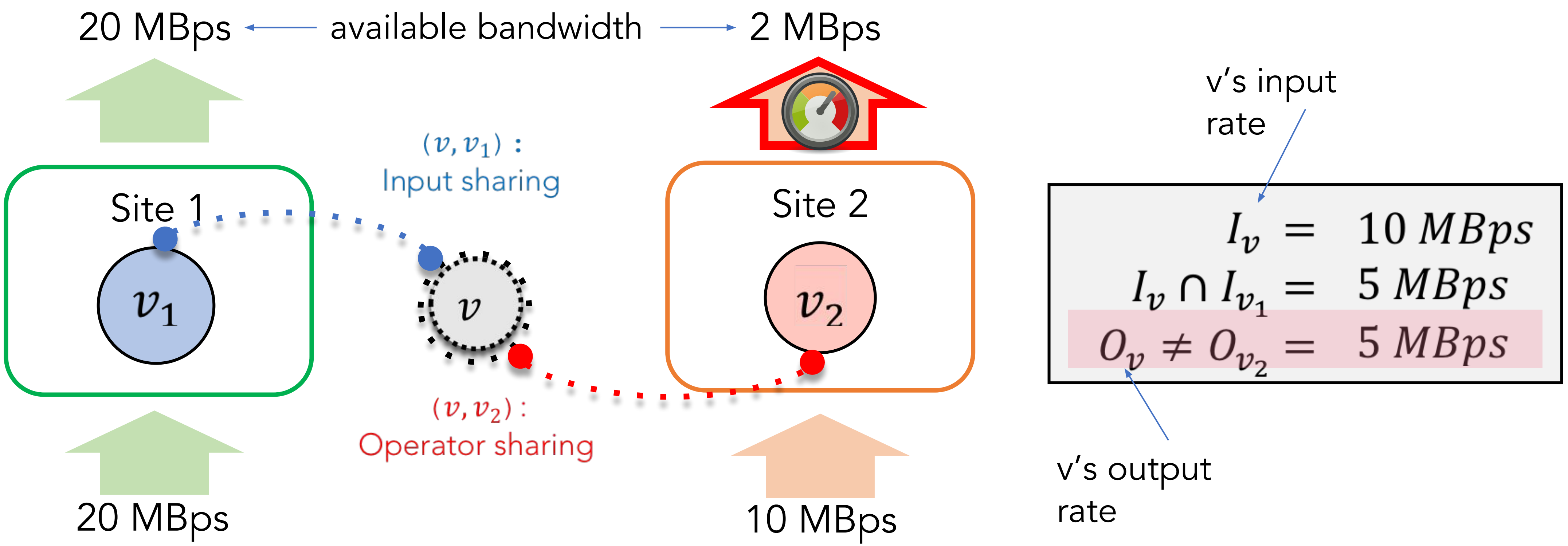
Sharing With Multiple Queries Incrementally

- Which queries to share?
  - Query-centric: maximum *similarity score* → limit to 1 query
  - Vertex-centric: traverse vertices topologically, may be shared with multiple queries
- Incremental sharing



WAN-Aware Execution Sharing

- Why MQO needs network awareness?



$$
\begin{aligned}
I_v &= 10\ MBps \\
I_v \cap I_{v_1} &= 5\ MBps \\
O_v \neq O_{v_2} &= 5\ MBps
\end{aligned}
$$

- WAN-aware MQO prevents bandwidth contention

WAN-Aware Task Deployment

- Vertices that exhibit commonalities:
  - Consider the sharing opportunities identified by the Query Optimizer
- Vertices that do not exhibit commonalities:
  - Local inputs → same site/node deployment
  - WAN-aware placement: jointly optimize latency and bandwidth

Implementation

- Sana prototype implementation on Apache Flink
  - WAN monitoring module
  - WAN-aware multi-query optimization
  - WAN-aware task placement
  - Managing execution states of shared queries
- Router operators are proactively added
  - Only added to vertices that consume remote input streams
  - Prevent suspending existing executions

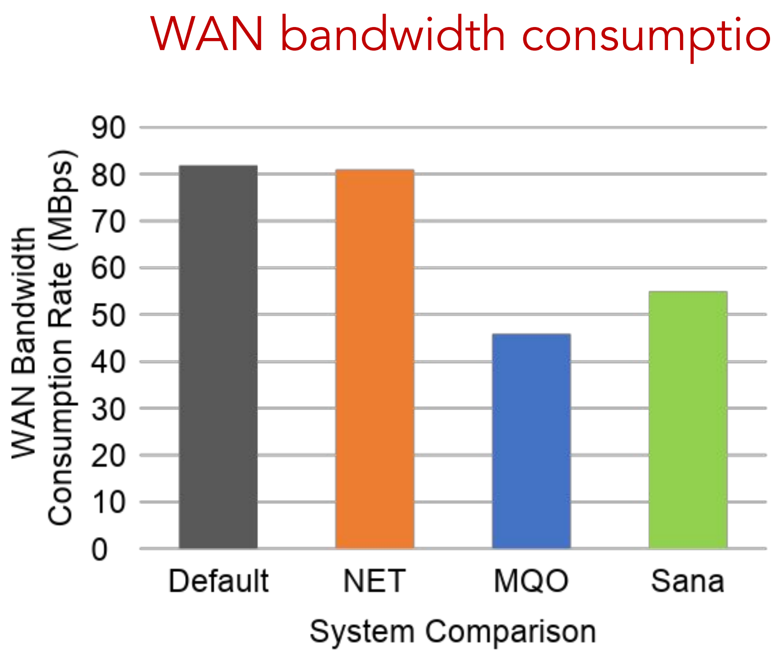
Experiment Setup

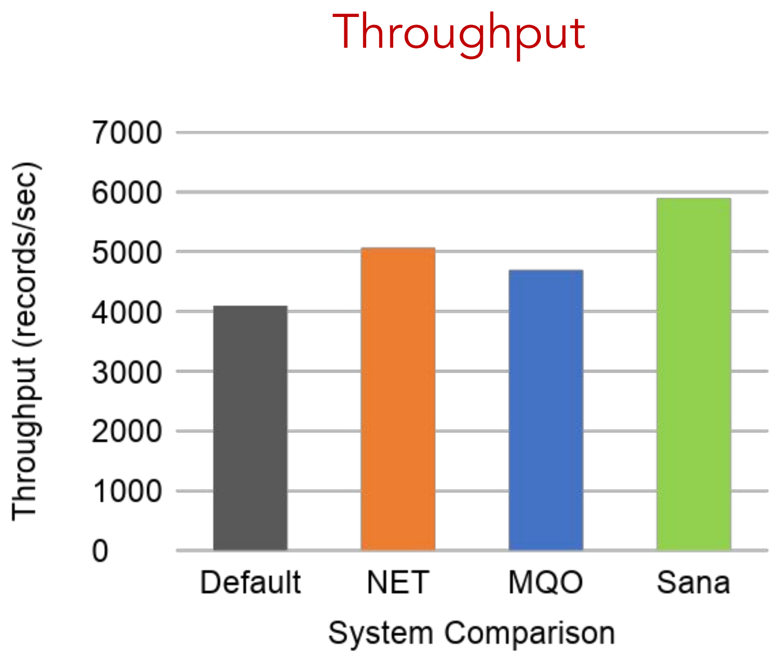
- Deployment on14 Amazon EC2 data centers

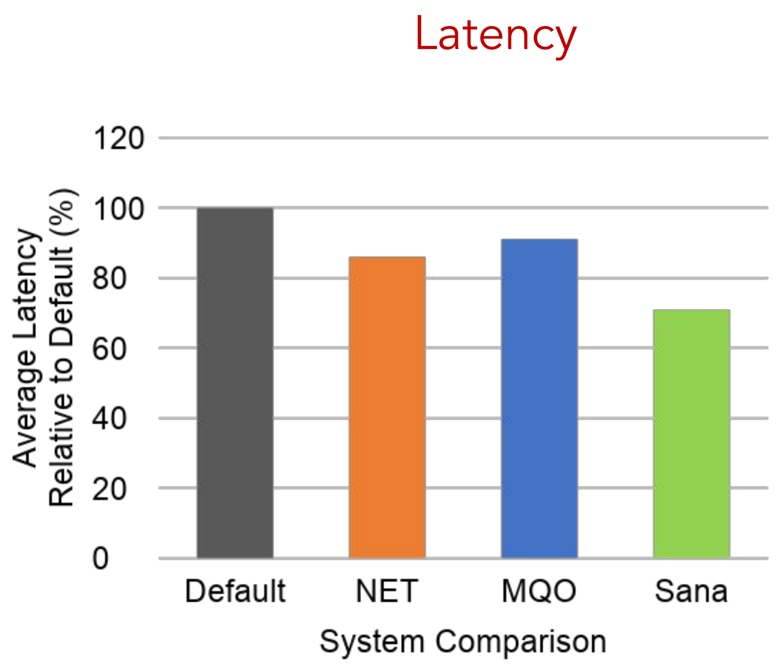
- Datasets & Queries
  - Real Twitter trace (scaled to ~6000-8000 tweets/second)
  - Distributed across 6 sites based on coordinates
  - Twitter Analytics Queries: Tweet statistics, Top-k analysis, Sentiment analysis, System metrics

- Baseline Comparison:
  - Default: WAN-agnostic, No Sharing
  - MQO: WAN-agnostic, Sharing
  - NET: WAN-aware, No Sharing
  - Sana: WAN-aware, Sharing

System Comparison





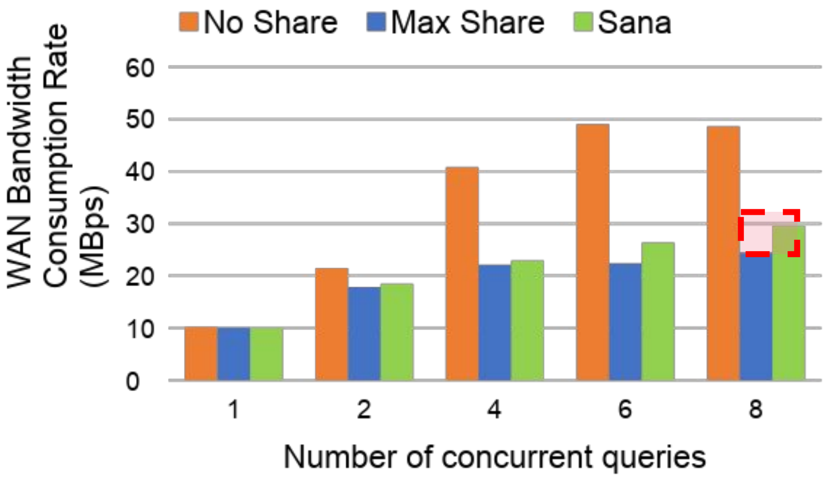


- Sana/NET: 17% higher throughput, 20% lower latency while saving 43% bandwidth
- Sana/MQO: 26% higher throughput, 23% lower latency, but consume 17% more bandwidth

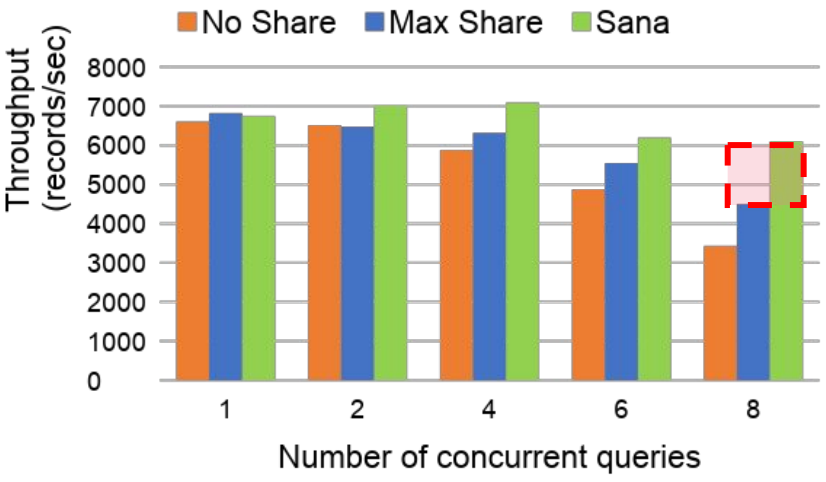
WAN-Aware Execution Sharing

- Maximizing sharing ⇏ maximizing performance
- Sana prevents bandwidth contention → higher throughput, lower latency

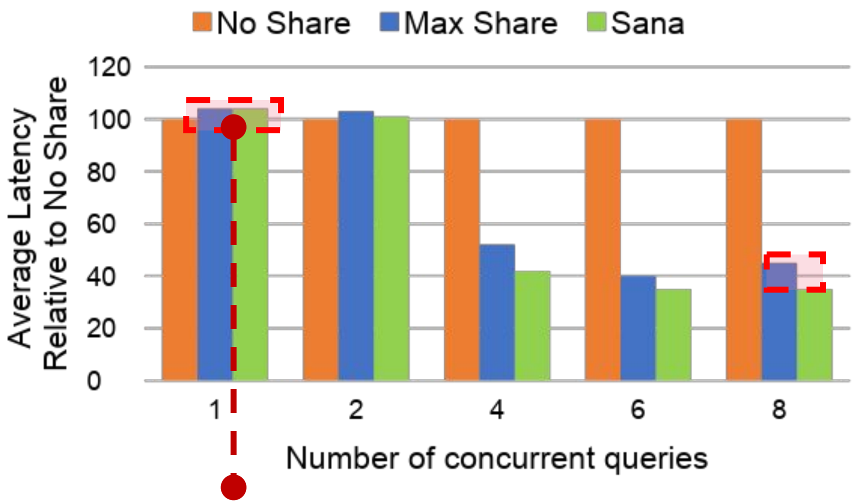
WAN bandwidth consumption



Throughput



Latency



Low overhead: 3~4% increase in latency

Conclusion

- Sana: Multi-Query Optimization for Wide-Area Streaming Analytics
    - Online incremental sharing
    - Low overhead
- WAN-aware sharing to maintain high performance executions
    - Maximizing degree of sharing $\nRightarrow$ maximizing performance
- EC2 deployment: higher performance while significantly reduce WAN bandwidth consumption

Thank You!

Questions?

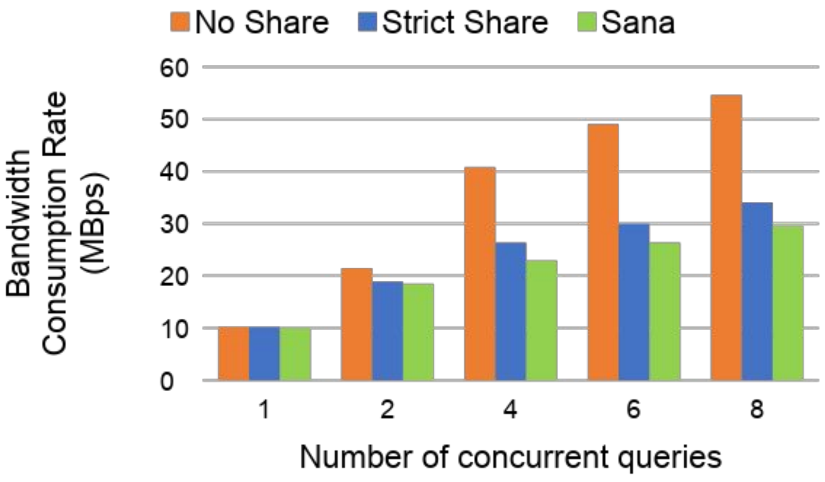
Contact:
albert@cs.umn.edu

University of Minnesota
Driven to Discover℠

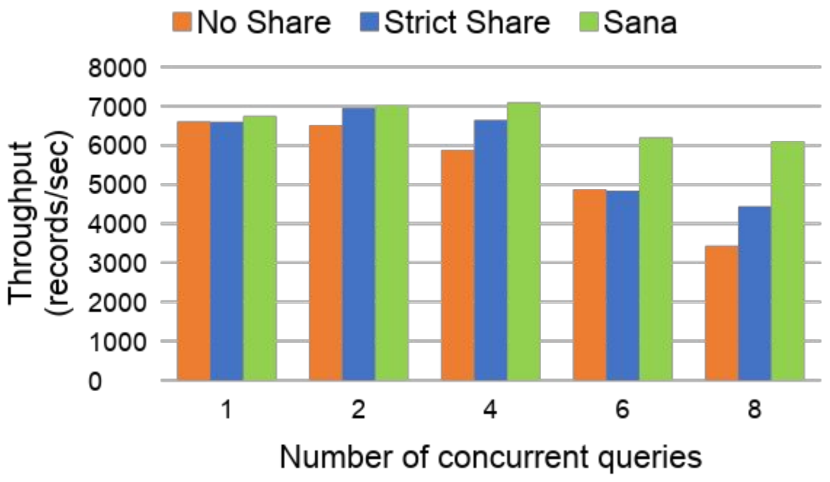
DCSG
Distributed Computing Systems Group

Benefit of Partial Input Sharing

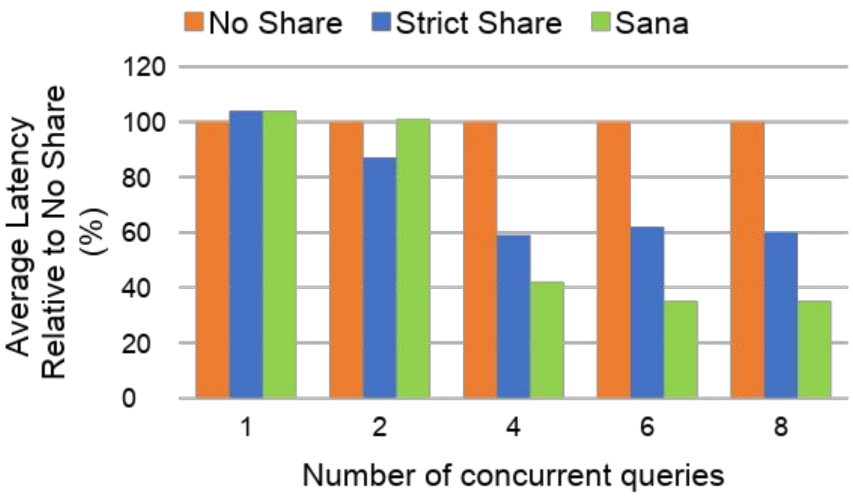
- Allowing partial sharing further improves performance (41% higher throughput) while saving bandwidth consumption rate by 45%



WAN bandwidth consumption



Throughput



Latency