1. Streaming challenges

- Highly volatile workloads
- Continuous queries
- Performance constraints: Low latency, Exact answers
- High input rate
- Must adapt to the workload seamlessly

2. More resources ≠ Better performance

- Uneven assignments
- Stragglers and resource underutilization
- High volatility

3. Partitioning: How it is currently done

- Hash partitioning
- Key splitting
- Skewed workloads lead to stragglers
- Changing the hash function comes at the cost of state migration
- Allows for balancing the load of partial aggregation
- Final aggregation can become the bottleneck

4. Dalton adapts partitioning at runtime

- Rewards computed by a cost model that balances partial and final aggregation
- Continuously learn rewards
- Exploration: leverage acquired experience
- Exploration: is more splitting beneficial?

5. Dalton scales to many partitioners

- Q-tables: Maintain information about the local hot keys
- Optimal policy for local distribution
- Dynamic workload: 1.3-6.3x higher throughput when the data distribution is skewed
- Note: Synchronization messages can be a bottleneck
- We propose a synchronization protocol that adjusts the sync frequency at runtime.

6. Dalton maximizes throughput

- Two partitioners 1.4-4.4x higher throughput with two partitioning instances

7. Conclusion

- Dalton: learns partitioning policies at runtime with minimal overhead
- Quickly adapts to the distribution and is able to scale not only the processing workers but also the partitioners
- Outperforms the state-of-the-art by a factor of 1.3-6.3x

8. More streaming challenges

- Unbounded data can lead to an unbounded state
- Multi-query optimization is crucial since queries run forever
- The query plan must be adapted upon addition of a new query