1. Partitioning must adapt to the workload

The distribution of data streams changes at runtime

2. Partitioning must be scalable

Partitioning can become a bottleneck

3. Partitioning: How it is currently done

- Skewed workloads lead to stragglers
- Changing the hash function comes at the cost of state migration
- Allows for balancing the load of partial aggregation
- Existing adaptive techniques:
  - incur high overhead
  - disallow scaling partitioning

4. Dalton adapts partitioning at runtime

- Rewards computed by a cost model that balances partial and final aggregation
- Continuously learn rewards
- Exploitation: 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

Note: Synchronization messages can be a bottleneck!

We propose a synchronization protocol that adjusts the sync frequency at runtime.

6. Dalton maximizes throughput

Dynamic workload

1.4-4.4x higher throughput when the data distribution is skewed

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

Dalton is the only algorithm that adapts to the data distribution and scales to multiple instances

7. Conclusion

- 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 1.4-6.3x