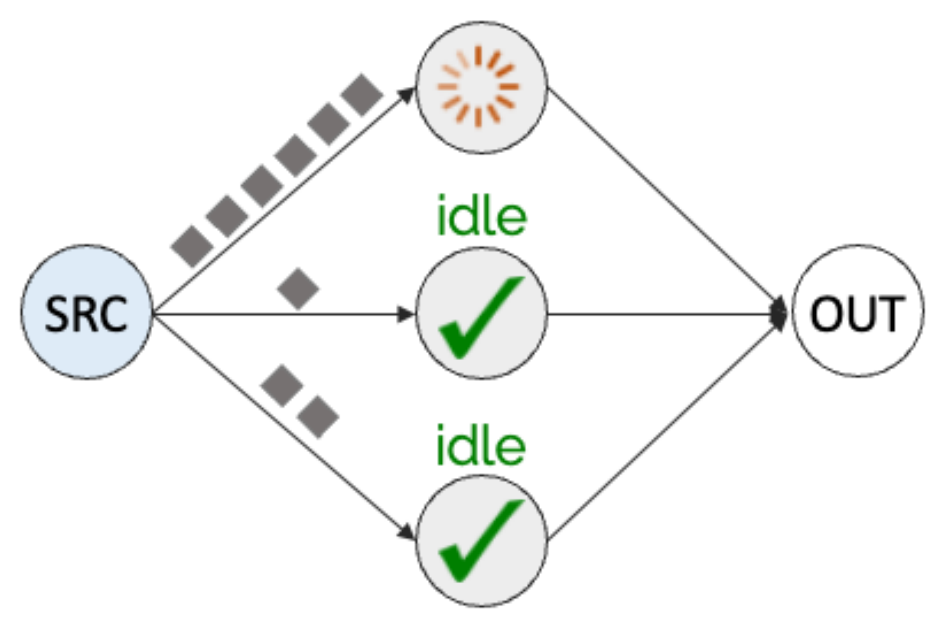
Dalton: Learned Partitioning for Distributed Data Streams

Eleni Zapridou, Ioannis Mytilinis, Anastasia Ailamaki

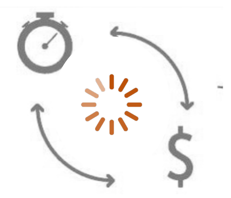
firstname.lastname@epfl.ch

1. Partitioning must adapt to the workload



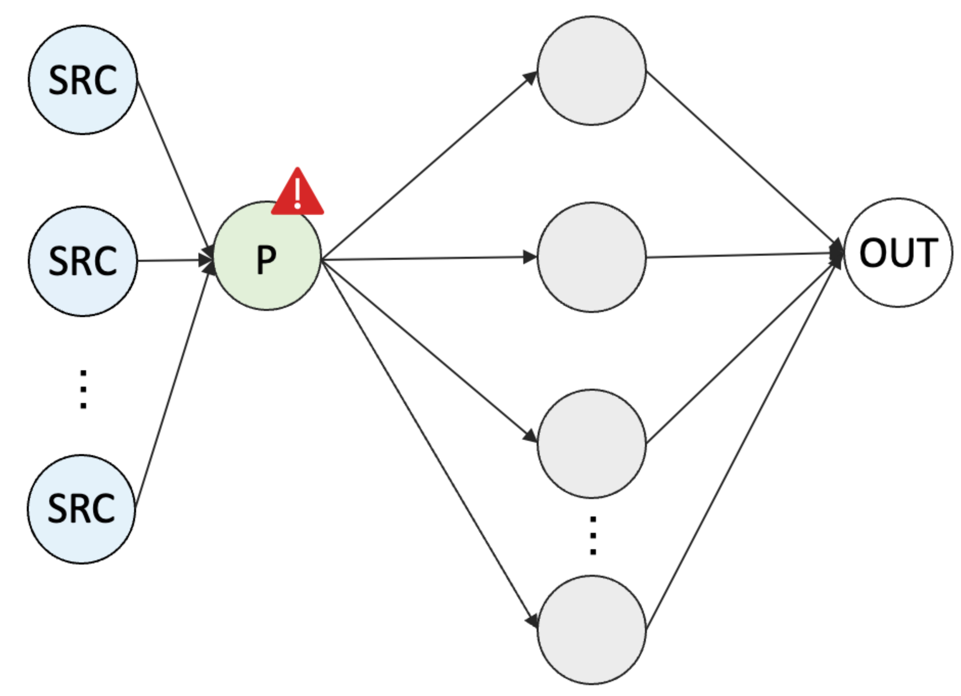
Uneven assignments

Stragglers and Resource underutilization



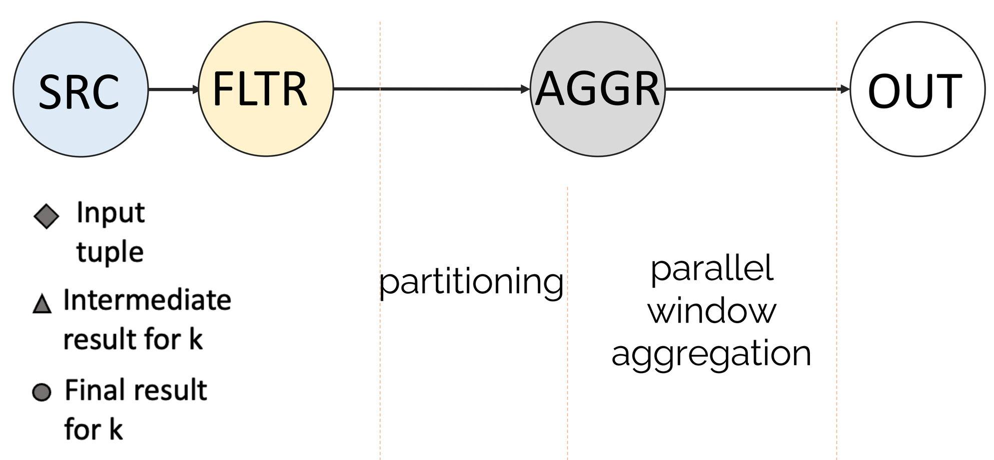
The distribution of data streams changes at runtime

2. Partitioning must be scalable



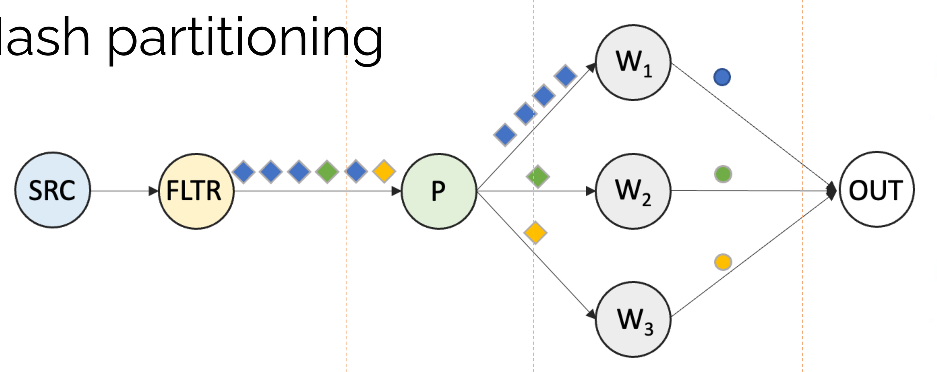
Partitioning can become a bottleneck

3. Partitioning: How it is currently done



SELECT * **FROM** Stream S
WHERE S.v > 10
GROUP BY S.k
WINDOW 60 **SLIDE** 1

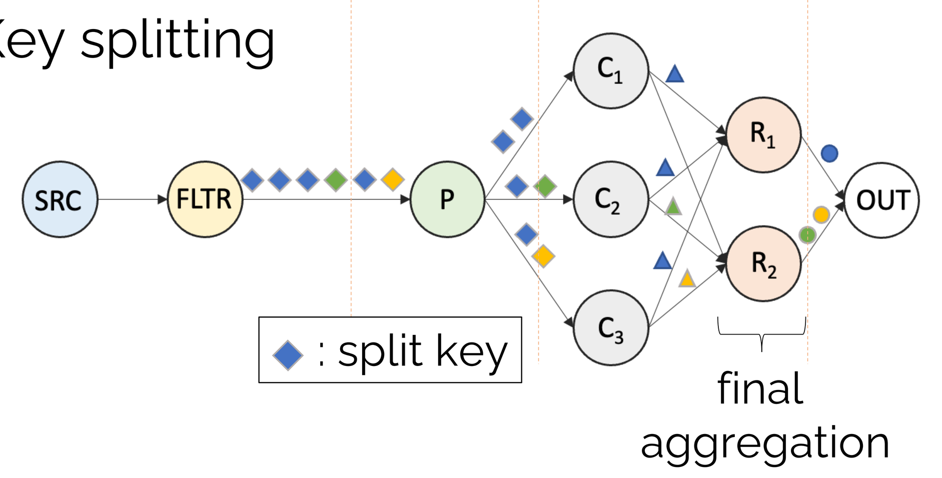
Hash partitioning



Skewed workloads lead to stragglers

Changing the hash function comes at the cost of state migration

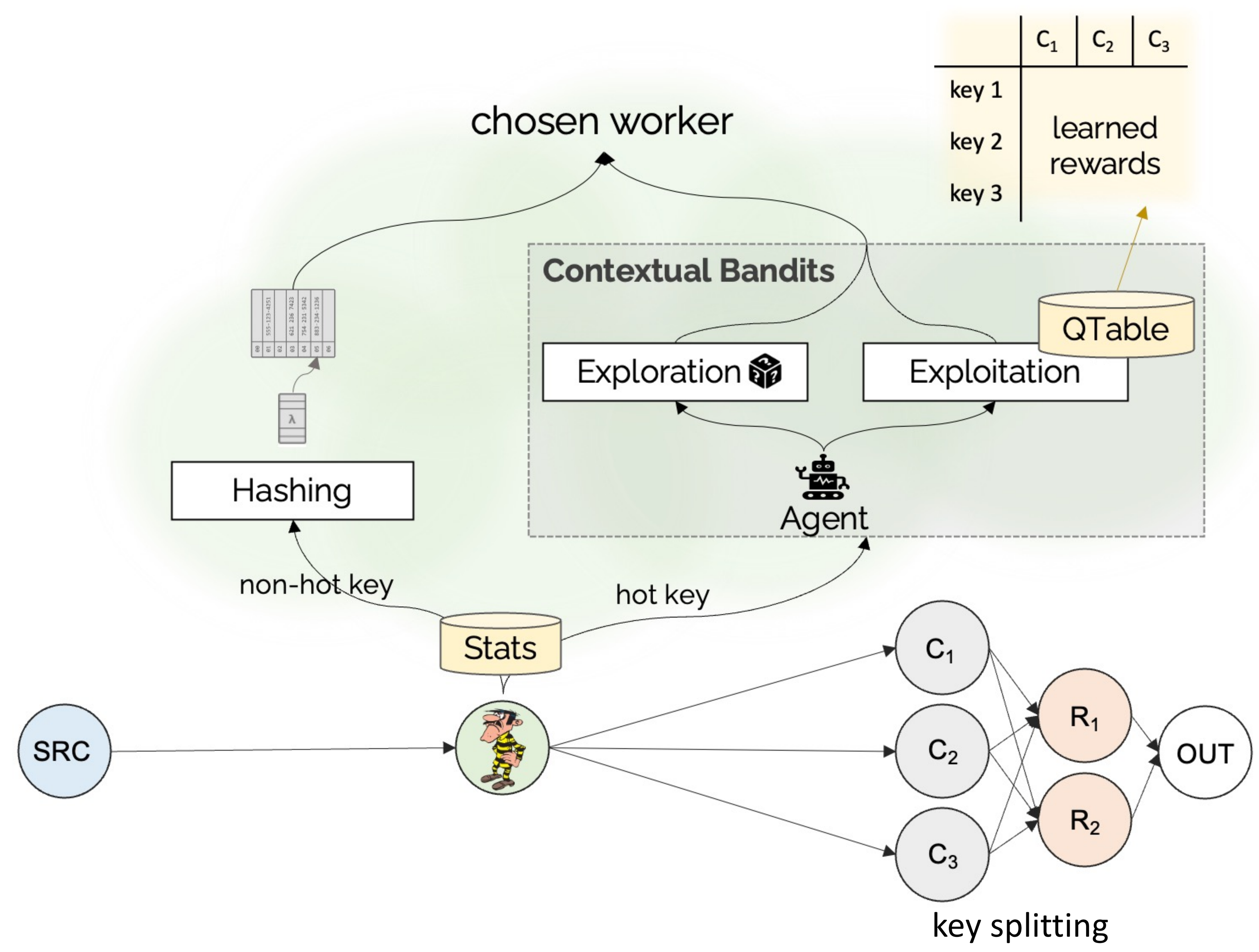
Key splitting



Allows for balancing the load of partial aggregation

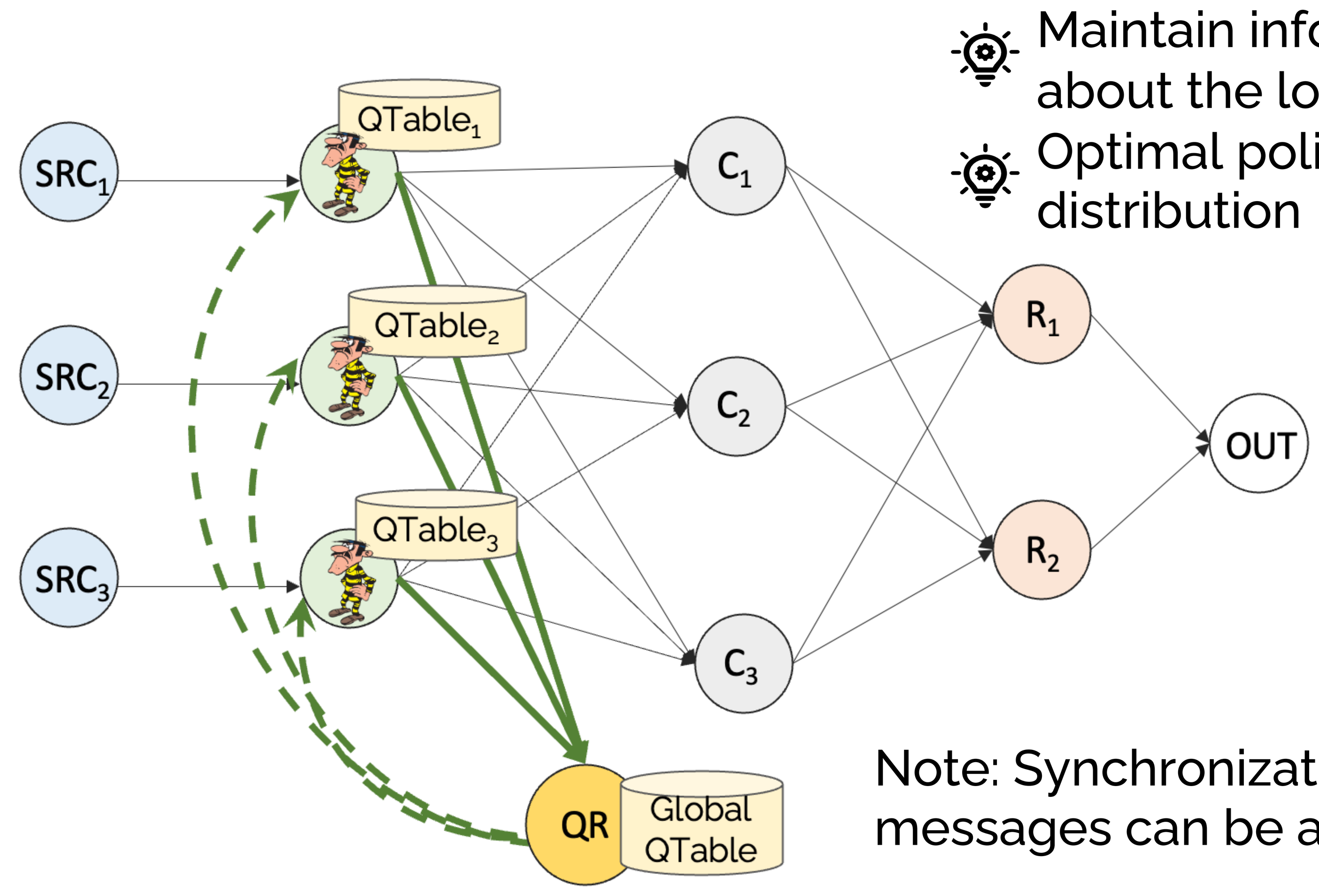
Existing adaptive techniques:
- incur high overhead
- disallow scaling partitioning

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

QR computes and sends to partitioners a global policy that maximizes aggregate throughput

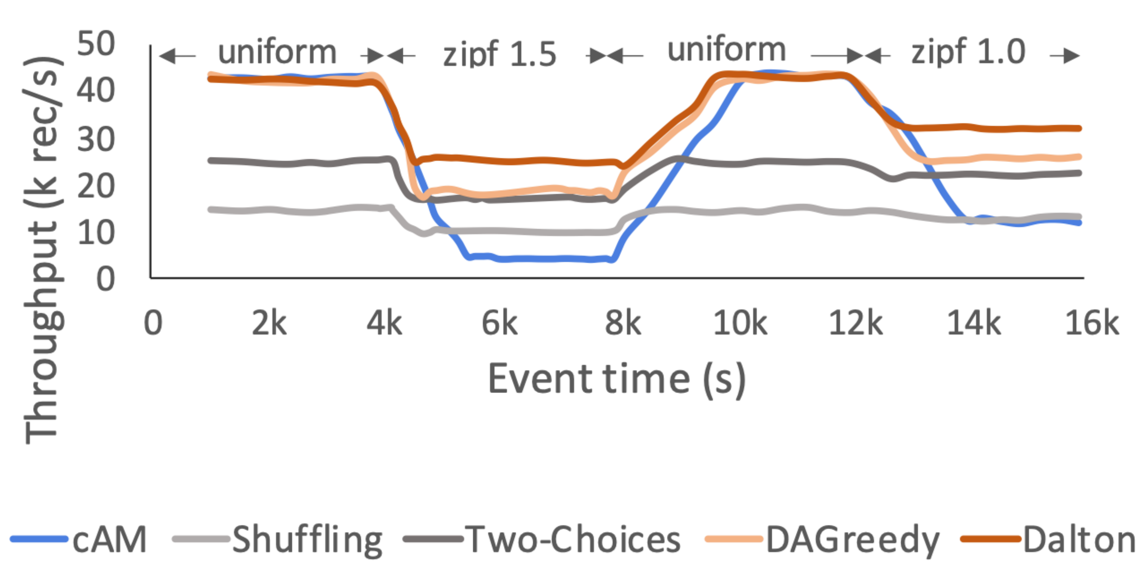
Note: Synchronization messages can be a bottleneck!

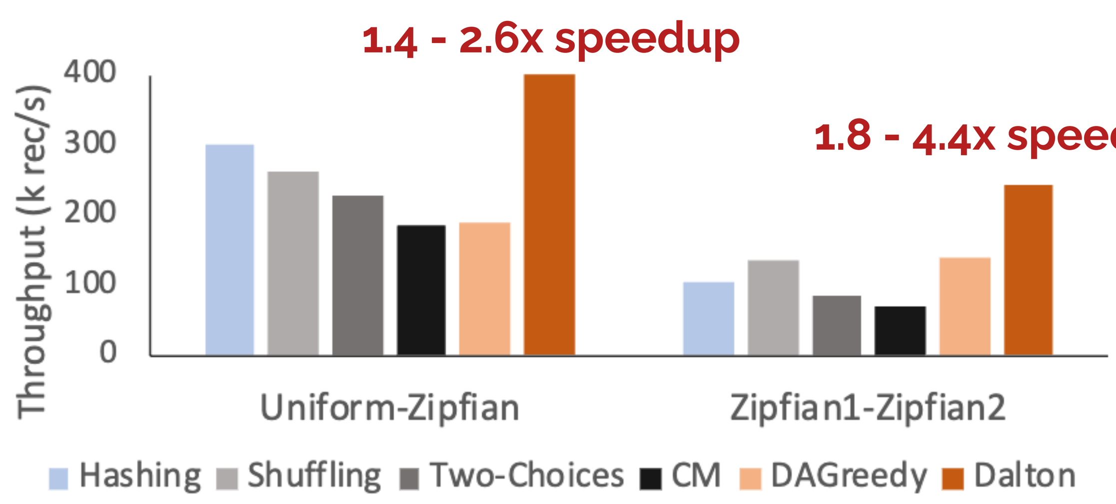
We propose a synchronization protocol that adjusts the sync frequency at runtime.

6. Dalton maximizes throughput

Dynamic workload

1.3-6.3x 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

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 1.4-6.3x

EPFL