Abstract
Stream join is an essential operation in many real-time applications. On static data, the HyperCube algorithm ensures a balanced load across all processors in an optimal way. We extend this algorithm to the streaming setting, which can adapt the HyperCube configuration depending on the current data distribution.

Input tuple \((a, b) \in R:\)

- **Light Hitter:** Parallel Hash Join
  \[ (a, b) \in R, (b, c) \in S \]

- **Heavy Hitter:** Cartesian Product
  \[ (a, b, c) \in Output \]

**HyperCube**

- **Output tuple**: \((a, b, c) \in Output\)

**Challenge**
- All heavy hitter information is needed to decide the configuration of each cube.
- The heavy hitter set may change throughout the stream processing.

System Architecture

- **Source**: Stream \(R = (A, B)\)  
  Stream \(S = (B, C)\)
- **Dispatcher**
  - **Light**
    - **Heavy**
      - Heavy Hitter Tracking
        \[ b^2 = \frac{IN(b)}{IN} \]
      - For each Heavy Hitter \(b\):
        \[ |R| = |P(R)|, |S| = |P(S)| \]
        \[ OUT(b) = |P(R) \cdot P(S)| \]
        \[ OUT^2 = \sum_{b \in S} \text{OUT}(b) \]
- **Task Pool**
  - **Light Tasks**
  - **Heavy Tasks**
    - **Create**
    - **Deprecate**
    - **Resize(+)**
    - **Resize(-)**
- **Output**

Evaluation

- **Parallel Hash Join**
- **Streaming HyperCube**
- **Join-Biclique**

**Zipf Data, Varying Skewness**
- **TPC-H Data, Varying Skewness**
- **COREL Data, r defines similarity** (Higher skewness with larger r)

When the current allocation is suboptimal by a constant factor!