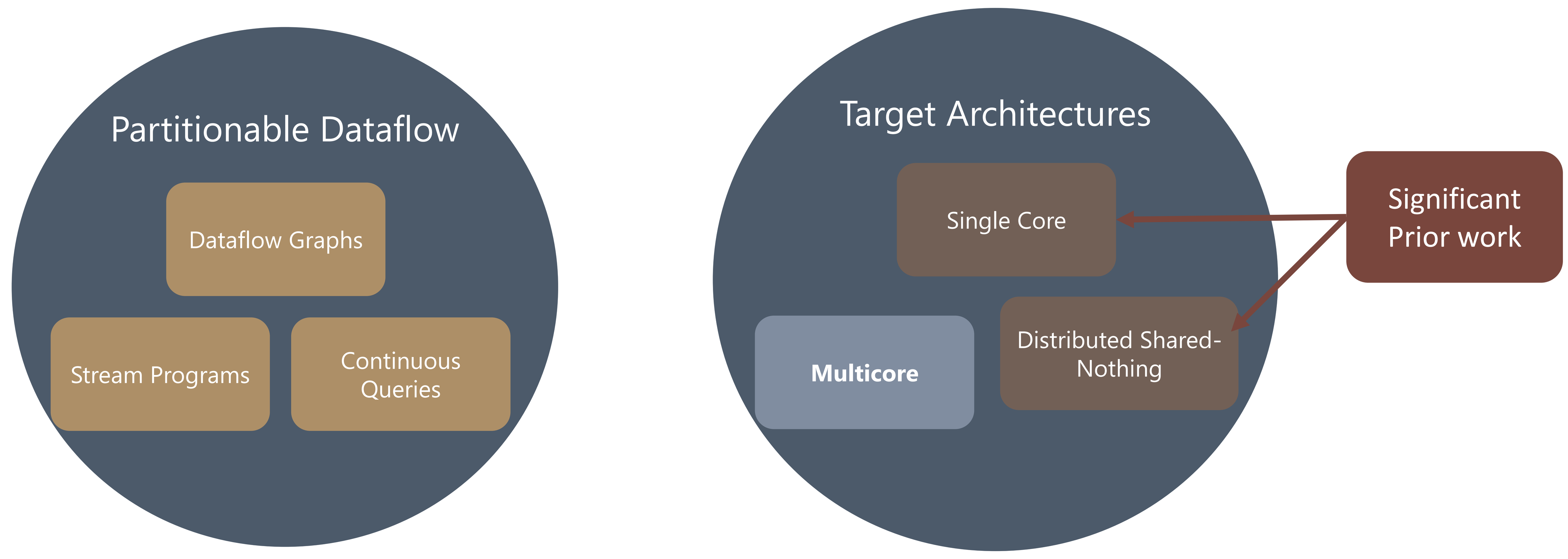
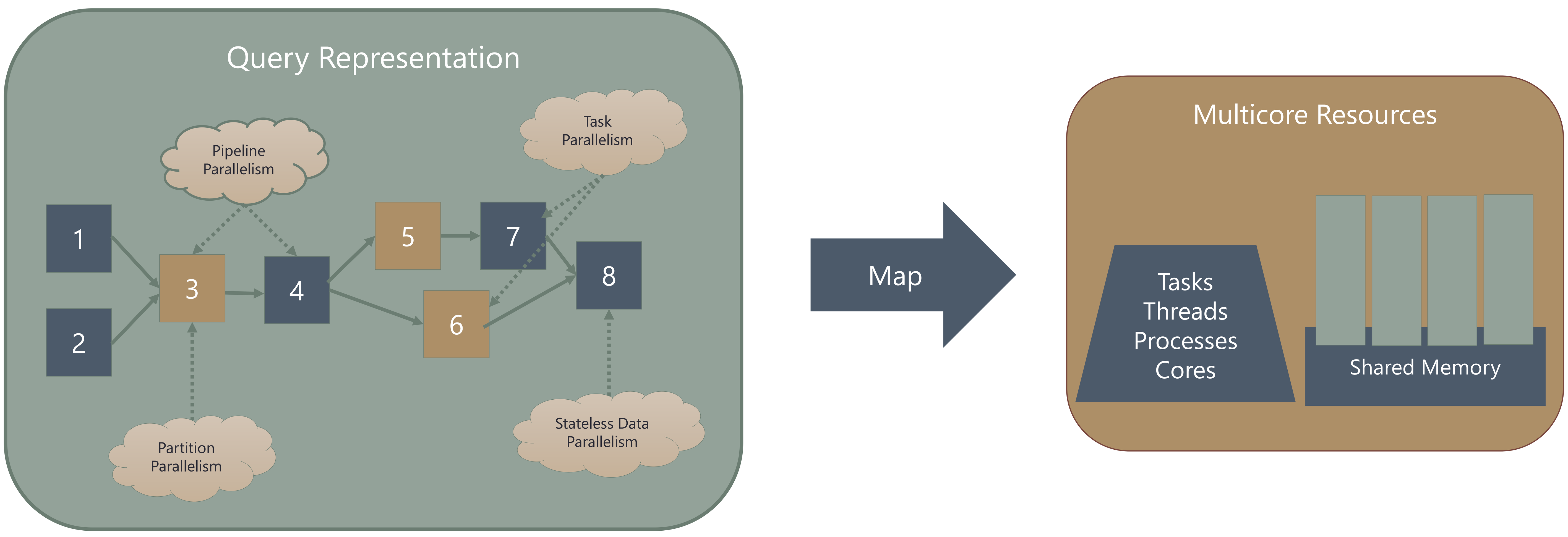
Multicore Scheduling of Partitionable Streaming Dataflow

Guna Prasaad, G. Ramalingam

Microsoft Research India





Combine smaller blocks into **super blocks**

Identify degree of parallelism for **stateless** super blocks

Identify partitioning parameters for **stateful** super blocks

Dynamically schedule the super blocks using **runtime statistics**

Optimization Techniques

- Super blocks reduce the processing cost to system overheads ratio
- Requires knowledge of processing costs and selectivity of blocks
- Graph partitioning problem – NP Complete
- Approximate solutions
  - Simulated Annealing
  - Heuristics based
- Obtain statistics using trial runs and during runtime for periodic changes

Exploiting Partition Parallelism

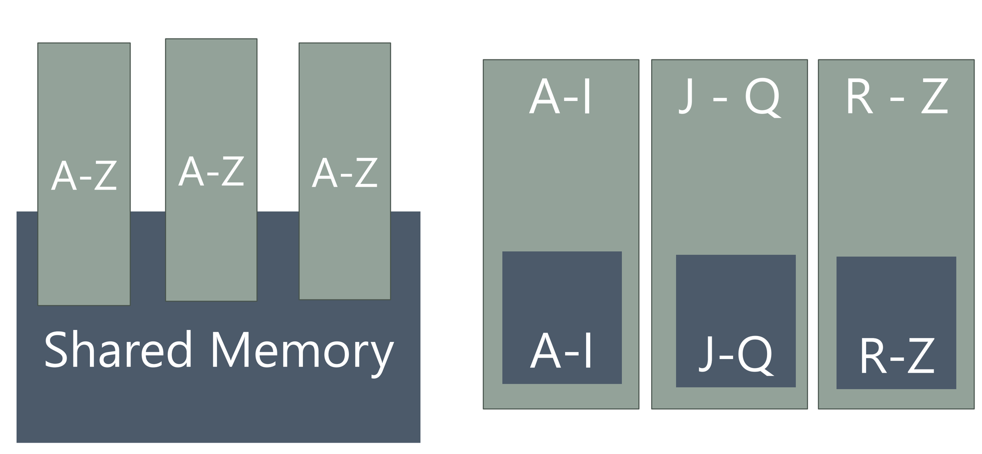


Fig: Shared state model vs. Partitioned State Model

Differences:

- Load Distribution
- Concurrency Control
- Thread Local Storage

Empirical Observations:

- Shared state model is as good as partitioned state model
- Shared memory and concurrency overheads are negligible in most practical cases

Dynamic Scheduling Strategies

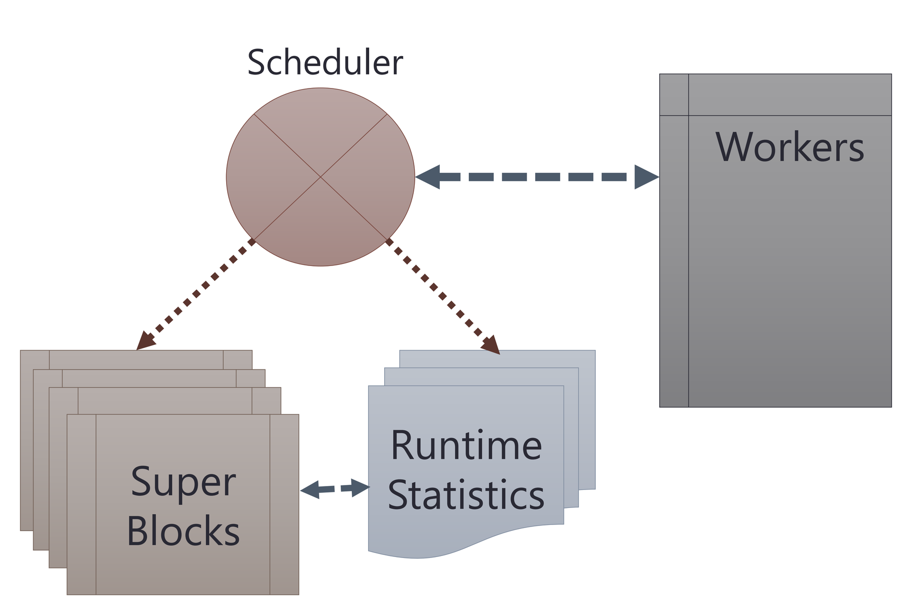


Fig: Runtime Architecture of Dynamic Scheduling

- **Objective** : to maintain a steady state flow of tuples through the pipeline
- Use runtime statistics such as average processing cost per tuple and selectivity of each block
- Identify blocks that fall behind in the pipeline and schedule them on the workers
- For eg. Estimated time to process the current queue