OnSuspendingandResumingDataflows
BadrishChandramouli,ChrisBond,ShivnathBabu,andJunYang

Scenario

**High-priority Task**
- Process as quickly as possible
- Ideally with all available resources
- E.g., real time decision queries

**Low-priority Query**
- Lots of resources, especially memory
- Extremely long runtime
- E.g., analytical (OLAP) queries

When a high-priority task arrives
- Suspend the low-priority query quickly
- Complete the high-priority task
- Resume the low-priority query

Contracting Mechanism

- Asynchronous checkpointing itself is insufficient
  - Child needs to be able to regenerate operator state at resume
  - **Contract**: agreement by child to regenerate tuples from an old point
  - Signed between parent and child, at parent’s MHS point

- Remembering latest checkpoint for each operator is insufficient
  - Wastes time and resources
  - High suspend-time overhead
  - starves low-priority task
  - High-priority task cannot wait

Choosing a Suspend Plan

- Choose strategy for each operator
  - **DumpState** (D) or **GoBack** (G)
  - Some strategy combinations are invalid
  - **Minimize total suspend/resume time**
  - **Constrained suspend budget**
  - Have all needed statistics at suspend time

Simple Solutions

**Kill and Restart**
- Wastes time and resources
- Starves low-priority task

**Use renice Command**
- Ineffective → controls CPU only
- Long time to release resources

**Limit Allocated Resources**
- Unnecessary restriction, may be infeasible

Our Solution: A New Query Lifecycle

- **Asynchronous Checkpointing**
  - **Minimal Heap State (MHS) points** of different operators usually do not coincide in time → synchronous checkpoints are expensive!
  - Idea: Checkpoint each operator independently at its MHS point → negligible runtime overhead (no disk writes)
  - At suspend, each operator has two choices: **DumpState** and **GoBack**

- **Synchronous Checkpointing**
  - Dump entire execution state periodically → high overhead during execution
  - Slow resume → should redo all the work done since last checkpoint

Applications of Suspend/Resume

- Queries with different priorities
- Utility and Grid settings
- Software rejuvenation
- DBMS maintenance

Other Improvements

- Can often skip ahead from checkpoint to target state on resume: no need to redo all the work
- Can migrate contracts to later points: more efficient resume
- The Query Optimizer can choose a suspend-aware query plan

Some Experimental Results

- Implemented and evaluated in **PREDATOR DBMS**
  - Requires minor extensions to the iterator interface
  - Currently supports NLJ, SMJ, merge sort, filter, and table scan
  - Can be extended to other operators, e.g., aggregation