On Suspending and Resuming Dataflows

Badrish Chandramouli, Chris Bond, Shivnath Babu, and Jun Yang

badrish@cs.duke.edu chrisbond@google.com
shivnath@cs.duke.edu junyang@cs.duke.edu

Duke Database Group

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

Simple Solutions

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

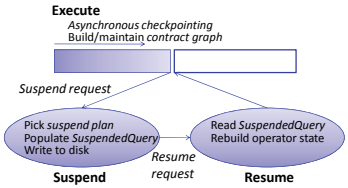
Use renice *Command*
- Ineffective → controls CPU only
- Long time to release resources

Limit Allocated Resources
- Unnecessary restriction, may be infeasible

Dump to Disk
- High suspend-time overhead
- High-priority task cannot wait

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

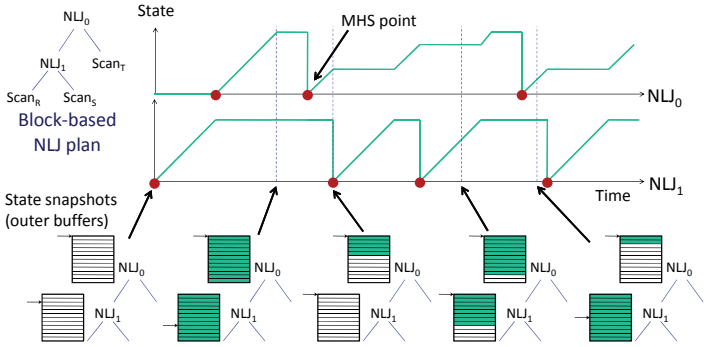
Our Solution: A New Query Lifecycle



Applications of Suspend/Resume
- Queries with different priorities
- Utility and Grid settings
- Software rejuvenation
- DBMS maintenance

Asynchronous Checkpointing

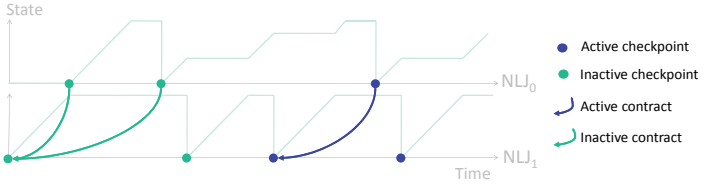
- *Minimal Heap State (MHS) points* of different operators usually do not coincide in time → synchronous checkpoints are expensive!



- 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*

Contracting Mechanism

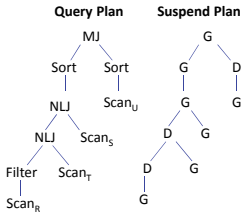
- Asynchronous checkpointing itself is insufficient
  - Child needs to be able to regenerate operator state at resume
- *Contract*: agreement by child to regenerate tuples from an old point
  - Signed between parent and child, at parent's MHS point



- Remembering latest checkpoint for each operator is insufficient
  - But, each operator has to retain only O(*h*) *active* checkpoints
- *Contract graph* stores active checkpoints/contracts: O(*nh*) *space* (*n*: number of operators in plan, *h*: height of query plan tree)
  - Kept in memory: size is a few MB, even for hundreds of operators

Choosing a Suspend Plan

- Constrained optimization problem
  - 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

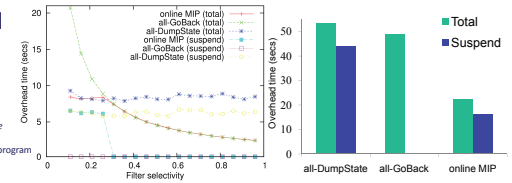


Other Improvements

- 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



all-DumpState: All operators choose *DumpState*
all-GoBack: All operators choose *GoBack*
online MIP: Suspend plan using mixed-integer program

- 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

For more details: Badrish Chandramouli, Christopher N. Bond, Shivnath Babu, and Jun Yang. Query Suspend and Resume. *To appear in* SIGMOD 2007.