Stop and Restart Style Query Execution

Surajit Chaudhuri
Raghav Kaushik
Ravi Ramamurthy

Microsoft Research

Abhijit Pol

University of Florida

Motivation

- Long running decision support queries can be resource intensive
  - Resource contention can be reduced by terminating queries
- Terminated queries have to be re-rerun completely
  - Irrespective of how close to completion they were
- Can we do better?
  - Assume database is not updated

Framework

INITIAL RUN

QUERY PLAN 

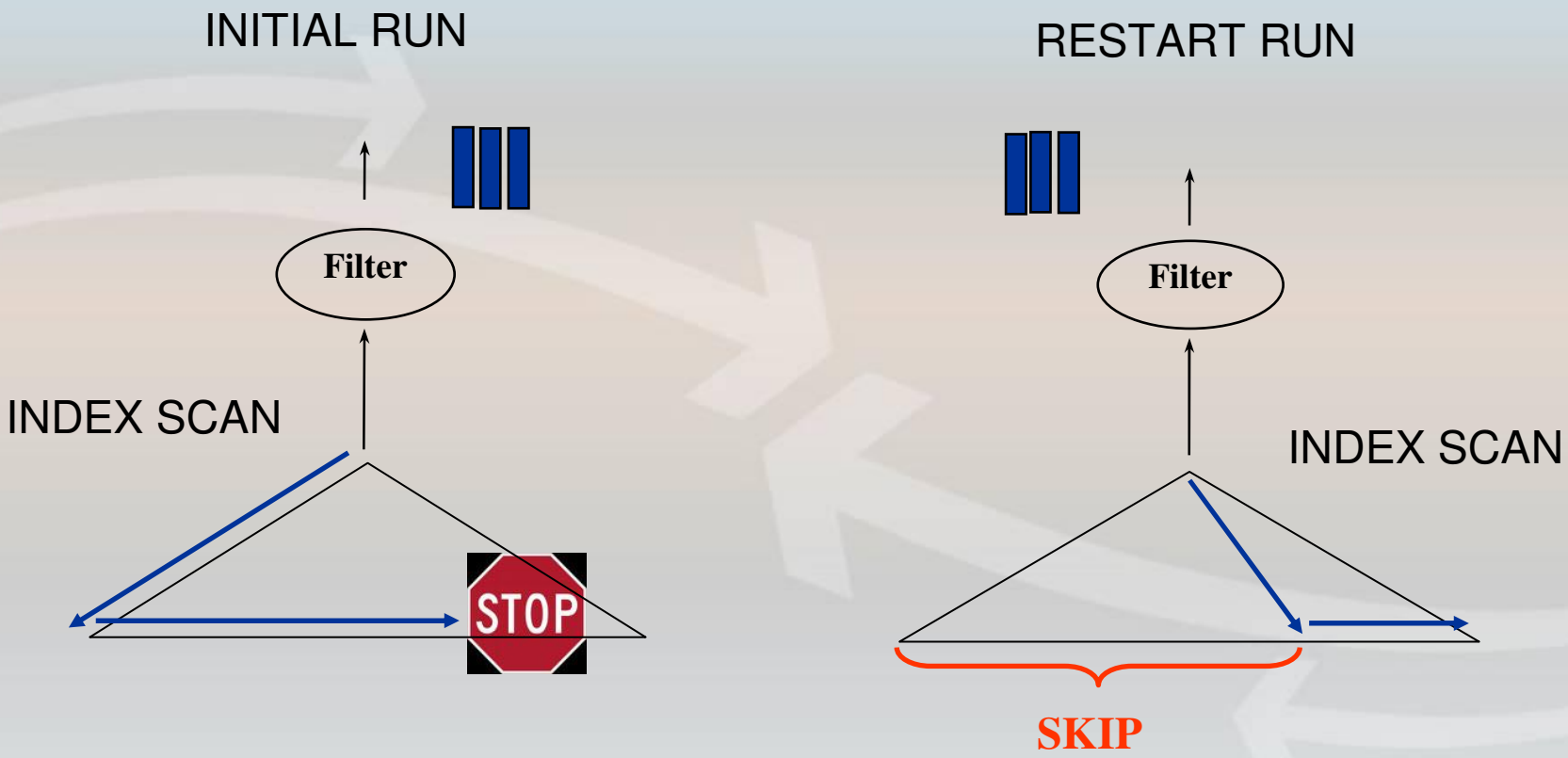
1. SAVE “STATE”
2. SAVE RESTART PLAN

RESTART RUN

EXECUTE RESTART PLAN

- **Can exploit existing interfaces for termination**
  - **Cancel Query, ODBC Timeout**
  - **Transparent to applications**
- **Return ALL the results in the restart run**
  - **Natural fit for stateless applications**

Candidate Approach



Problems with this Approach

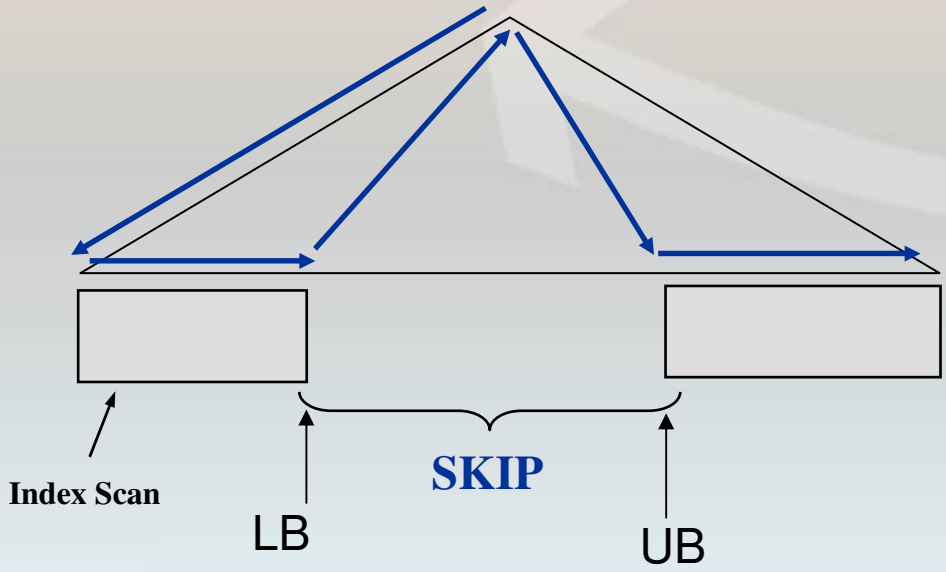
- Overheads can be unbounded
  - e.g. Filter predicate selects most of the records
  - Need to cache results in memory or periodically flush them to disk
  - The query may not be terminated!
- Bounded overhead
  - Save and Reuse “Best” K records
  - Flush results to disk when the query is terminated

Outline

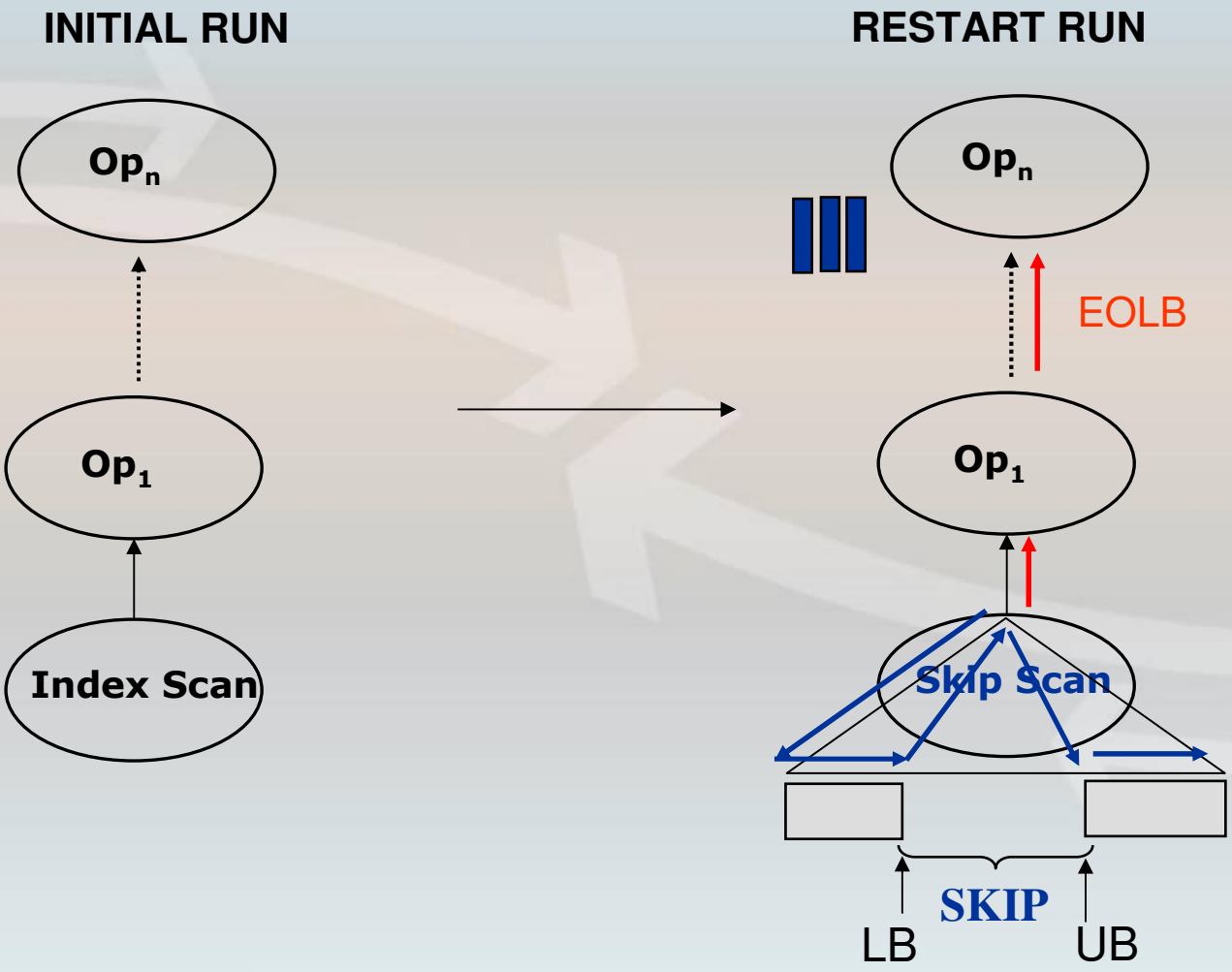
- Primitives
- Bounded Query Checkpointing
- Opt-Skip Algorithm
- Complex Query Plans
- Experiments

Skip-Scan Operator

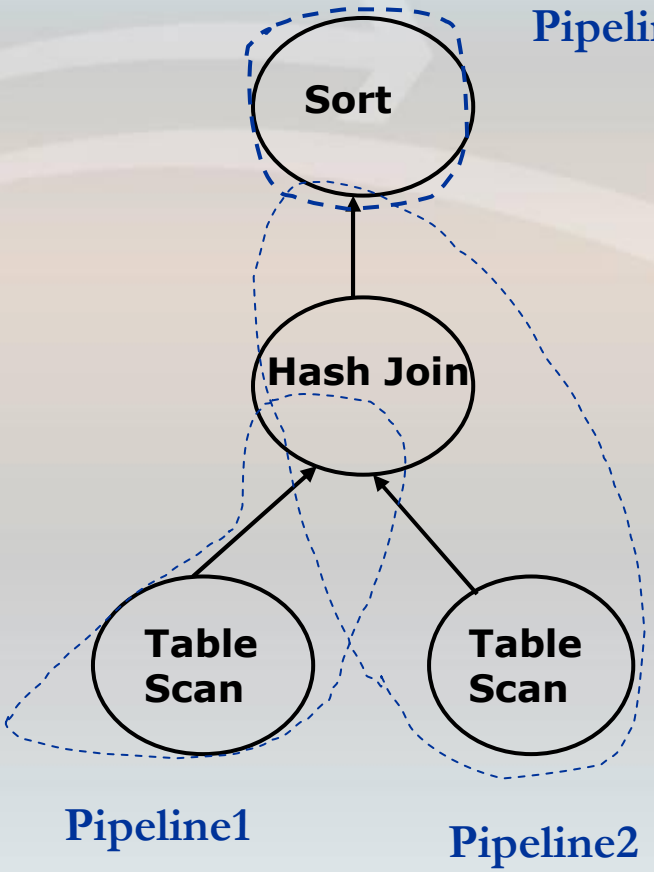
- Generalization of Scan Operator
- Takes 2 parameters (LB, UB)
- Skips all records between LB and UB
- Sends a EOLB message after processing LB



Restart Plans for Single Pipelines

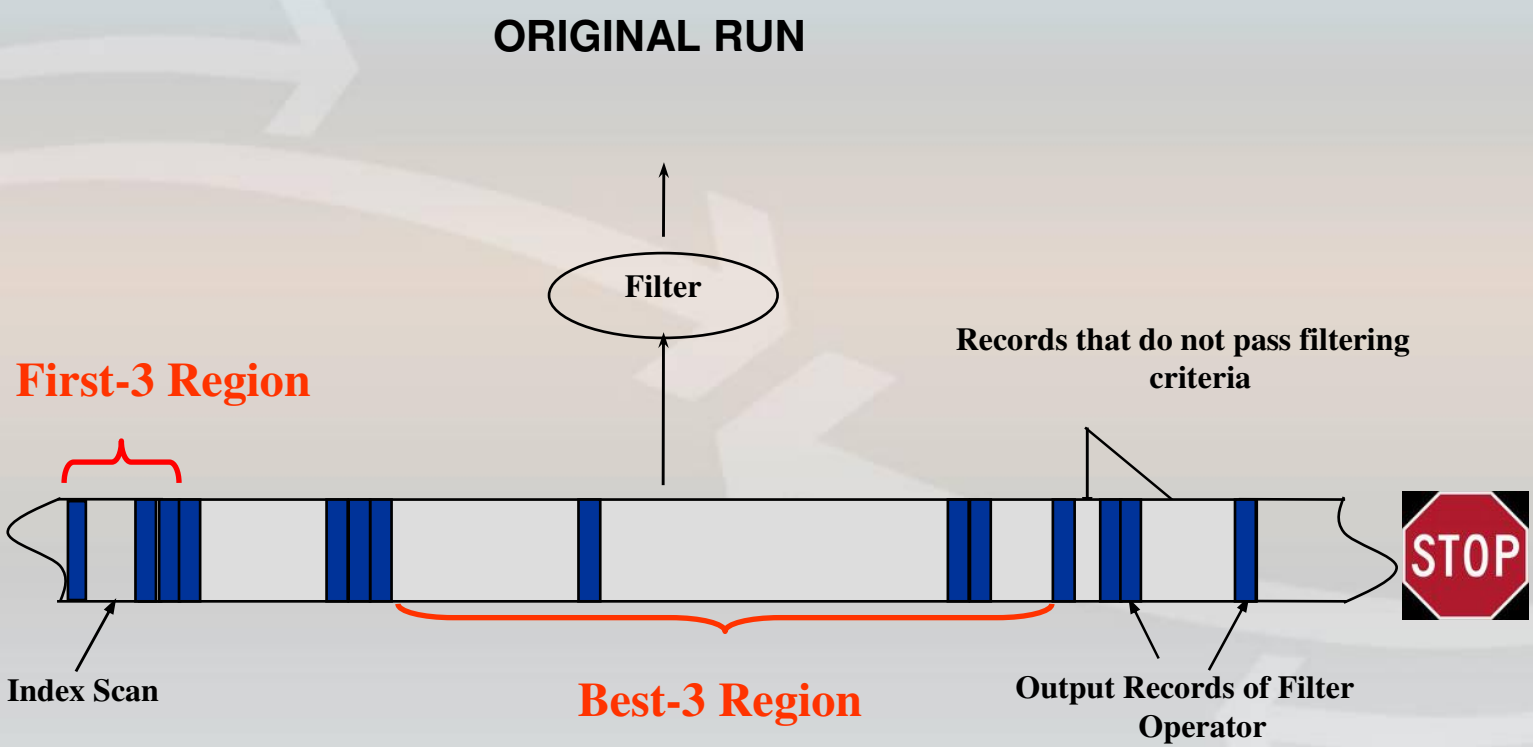


Cost Model



- Work done by query = Total number of getNext Calls issued across all operators

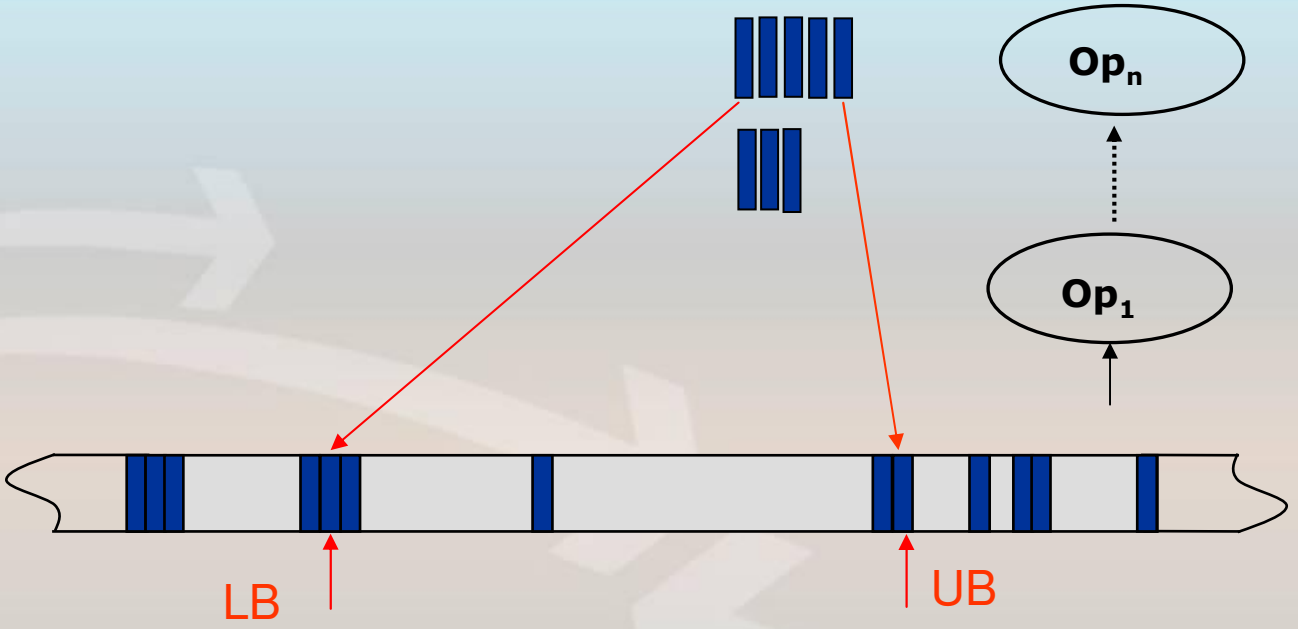
Example (K = 3)


ORIGINAL RUN
Filter
Records that do not pass filtering criteria
First-3 Region
STOP
Index Scan
Best-3 Region
Output Records of Filter Operator

Bounded Query Checkpointing

- **Parameters**
  - **Budget k**
  - **Let id be the current ID of the record in the leaf node**
- **Among all bounded RPlan(LB,UB) where LB < UB <= id**
  - **Maintain the restart plan with maximum benefit**

Candidate Windows and Restart Plans (k=3)



- Track candidate windows (r_{i-1},r_{i+k}) of size k+2
  - Derive LB and UB values from the source RIDS corresponding to r_{i-1} and r_{i+k}
- Let GN(ri) denote total GetNext calls issued in the pipeline until r_i was generated at the pipeline root
  - Benefit of Restart Plan = GN(r_{i+k}) – GN(r_{i-1})

Opt-Skip Algorithm

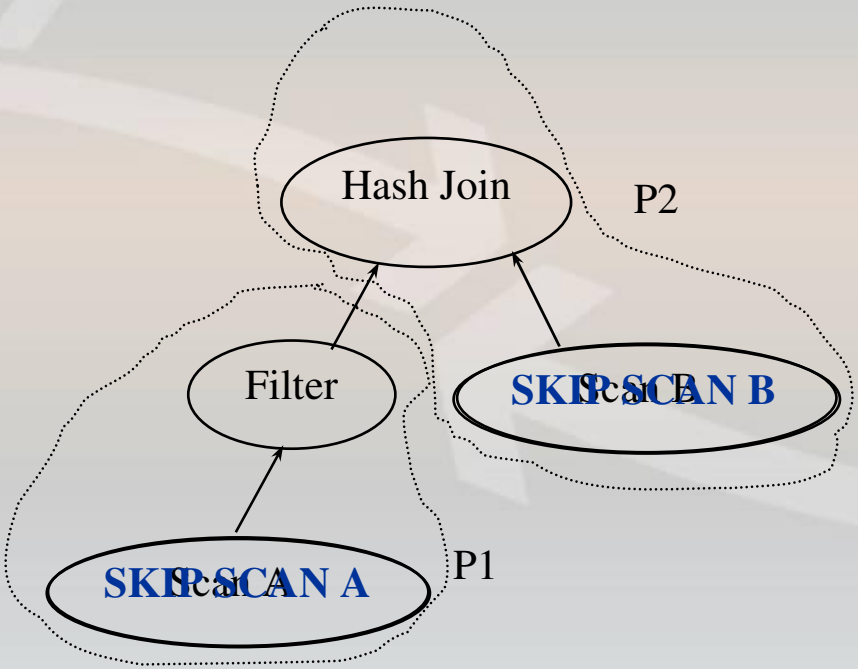
```
/* window, k= total budget */
/* BestW = best window */

Algorithm Opt-Skip
    BestW = empty set
    W = empty set
    For Each intermediate record r_i do:
        Append r_i to W
        If W.Size() > k+2 then
            W = last k+2 records in W

        SkippableW = FindSkippable(W)

        If Benefit(SkippableW) > Benefit(BestW) then
            BestW = SkippableW
```

Restart plans for Multiple Pipelines



Algorithms for Multiple Pipelines

- Current Pipeline
  - Use Opt-Skip on the current pipeline
- Max-Pipeline
  - Choose single “best” pipeline from all completed pipelines
- Merge-Pipeline
  - Heuristic to distribute the budget k among multiple pipelines
- Subtree Caching

Experimental Evaluation

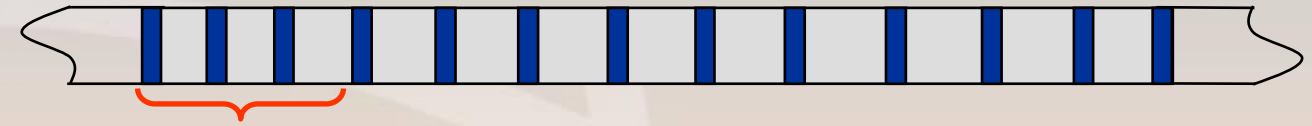
- Prototyped in Microsoft SQL Server 2005
  - Built skip-scan operator on top of the clustered index scan operator
- Evaluation Metric
  - T1 = Total number of GetNext calls in Initial Run before termination
  - T2 = GetNext calls issued in the restart to reach the same point in execution
  - Percentage Work Saved (PWS) = (T1-T2)/T1*100

Goals

- Effect of clustering
- Utility of bounded query checkpointing for complex queries
- Overheads in the initial run
- Algorithms for multi pipeline plans

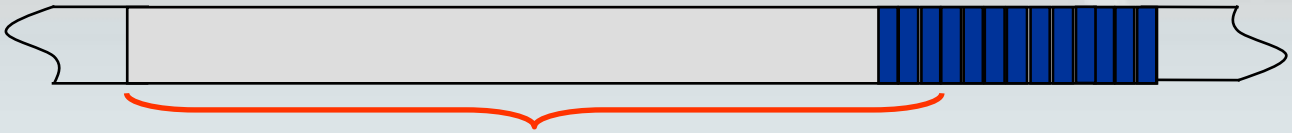
Effect of Clustering/Correlation

SELECT * FROM LINEITEM WHERE L_RECEIPTDATE > $v1



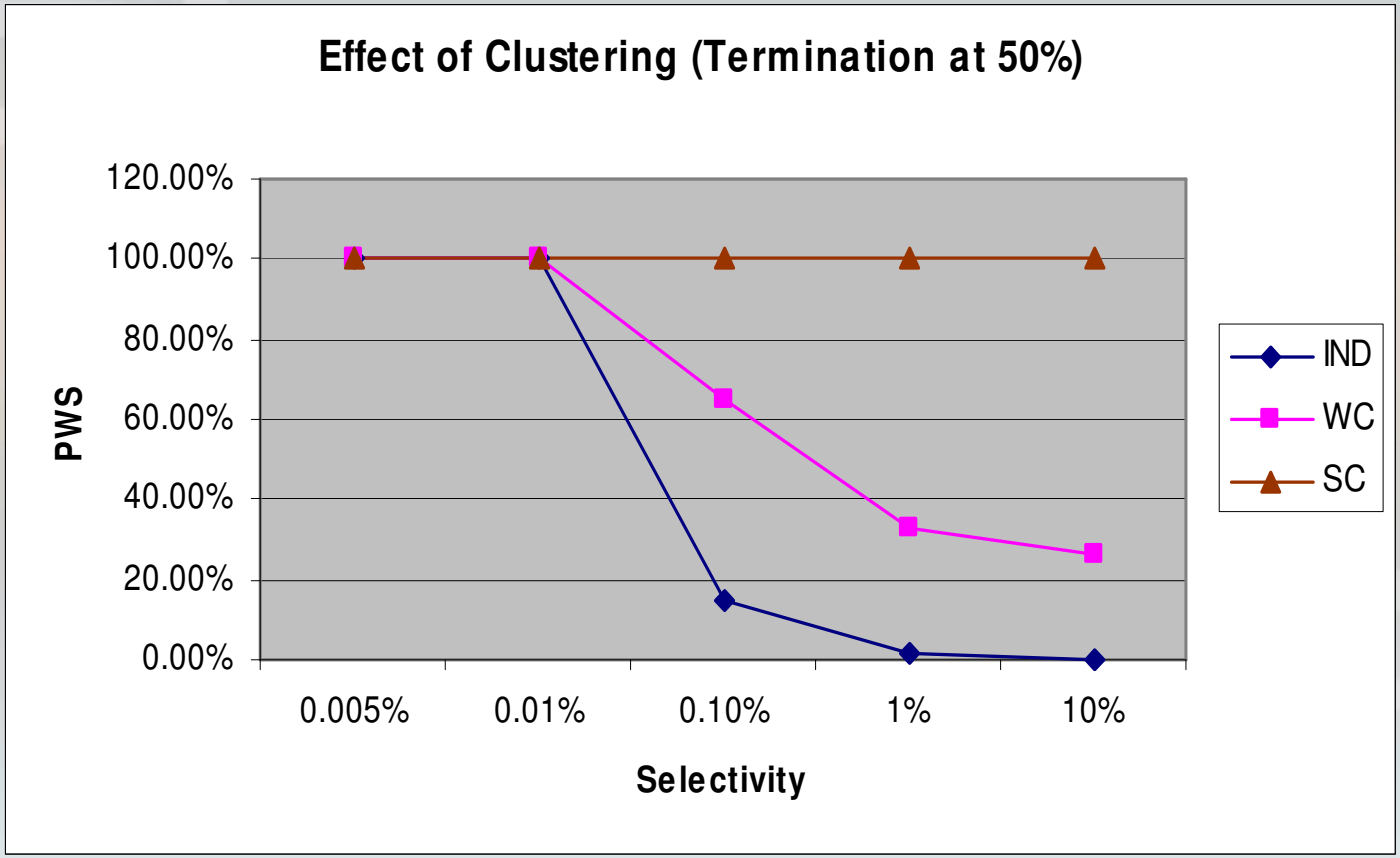
L_RETURNFLAG

L_ORDERKEY

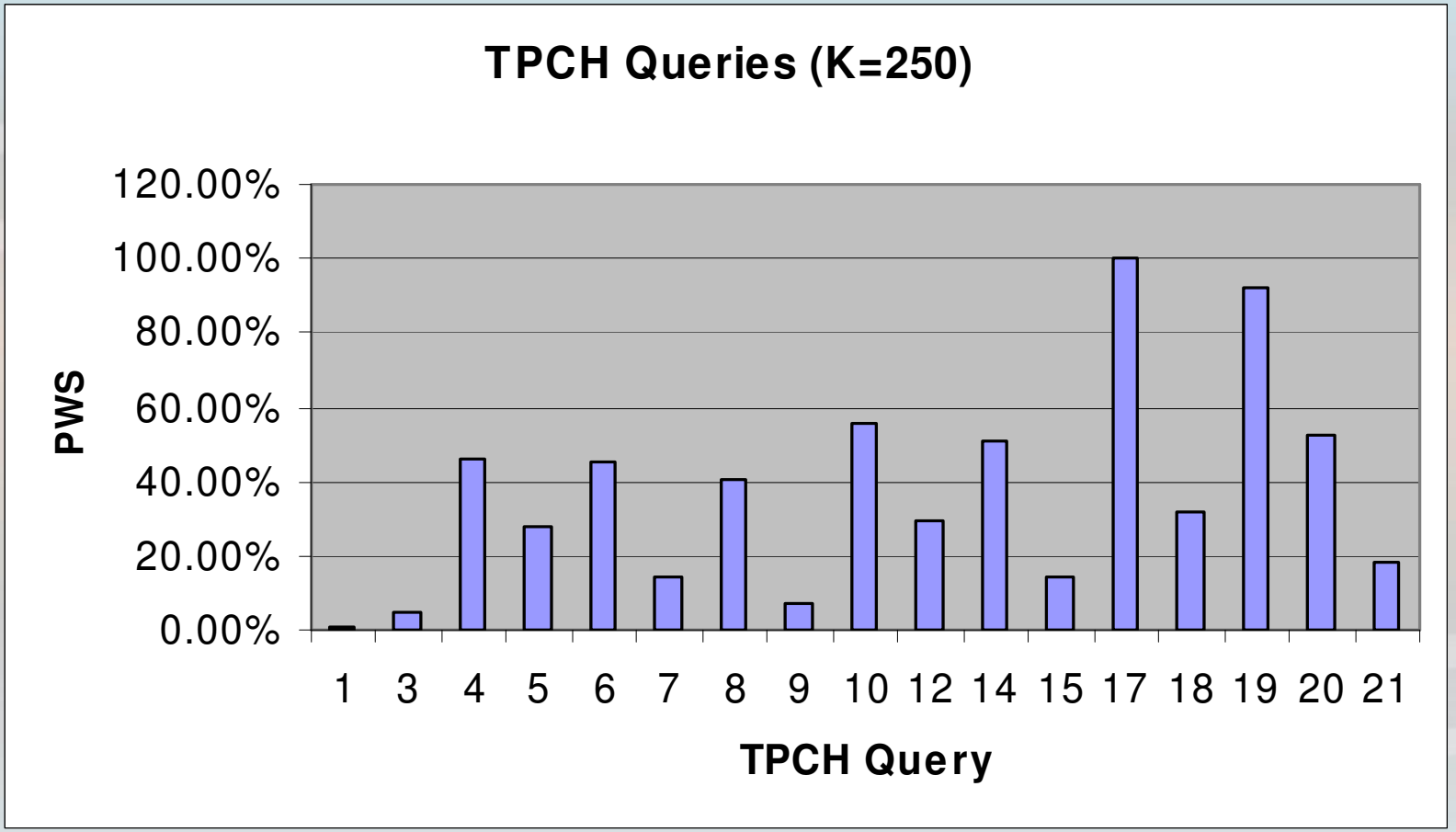


L_SHIPDATE

Effect of Clustering

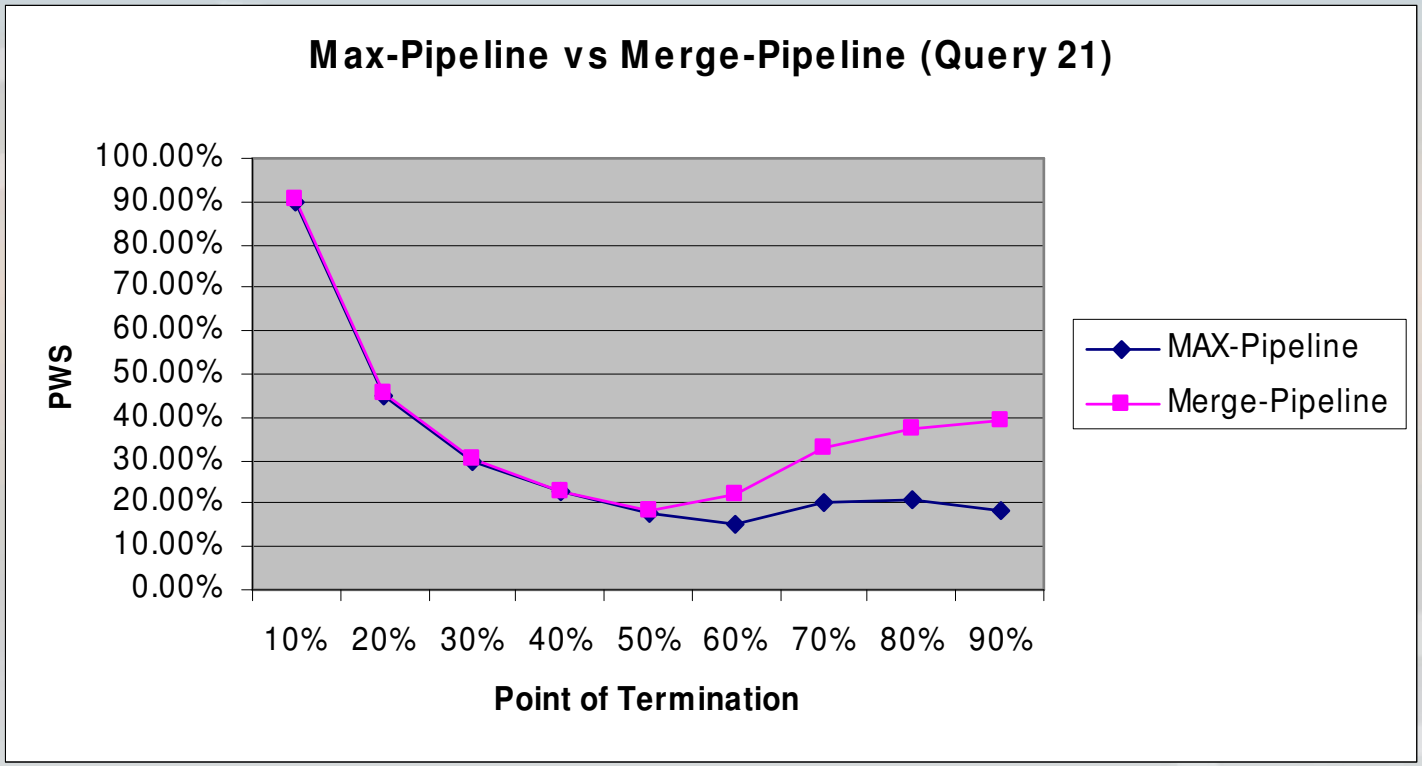


TPC-H Queries (Termination at 50%)

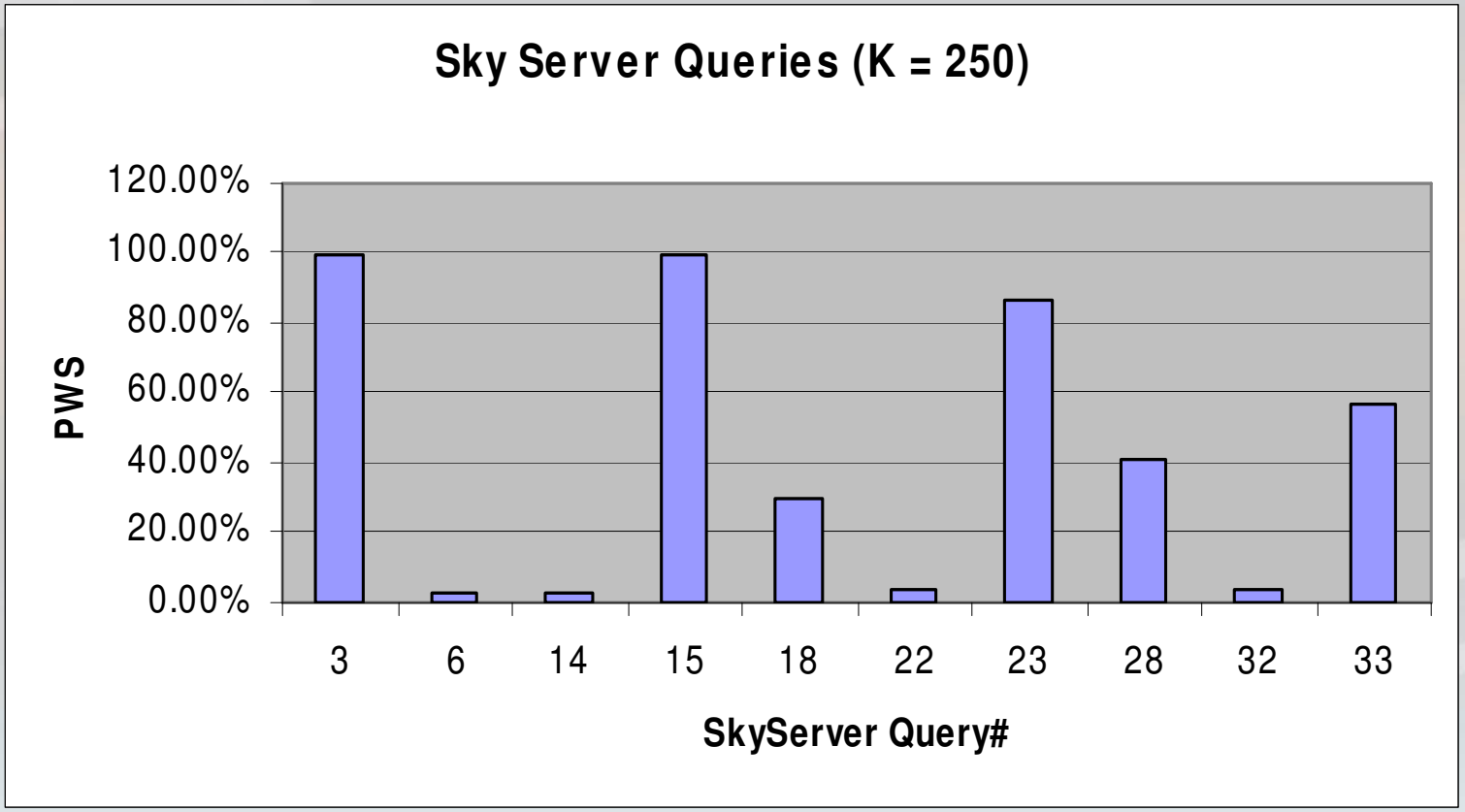


Overheads in Initial Run were < 3% for most Queries

Multi-Pipeline Algorithms



SkyServer Database



Summary of Results

- Many cases where bounded checkpointing is useful
  - selective predicates, correlation with the clustering column, subtree caching
- Saving a small number of records can result in substantial savings
- Keeping track of previous pipelines is important

Future Work

- Aggregations
  - Saving partial sums
- Hash Spills
  - Maintaining the skipping invariant
- Handling Updates
  - Validate Restart Plan

Related Work

- SQL Cursors
  - Does not release any state
- Query Resumption
  - [Labio et al. SIGMOD 2000]
  - [Chandramouli et al. SIGMOD 2007]
  - Delta results vs. Full Results
  - Skip-Scan can be utilized for resuming the build phase of hybrid hash join

Conclusions

- Resource contention can lead to termination of queries
  - Queries have to be rerun completely
- Bounded Query Checkpointing
  - Skip-Scan Operator
- Saving a small amount of records can lead to substantial savings