Pipelining in Multi-Query Optimization

by

Sumit Sanghai
Nilesh Dalvi
Prasan Roy
Dr. S. Sudarshan

appears in

PODS 2001

Overview of the talk

- Multi-Query Optimization(MQO).
- Pipelining Concepts.
- A model for Pipeline Schedule.
- Criterion for validity of Schedule.
- Finding least cost Pipeline Schedules.
- Greedy Heuristic
- Shared Read Optimization
- Results

Motivation for MQO

- Optimizing collection of queries.
- Common subexpressions need not be recomputed.

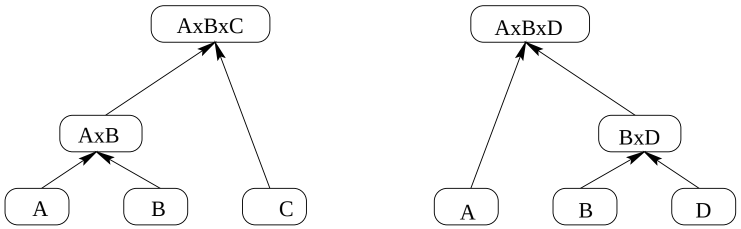


Figure 1: Two separate queries

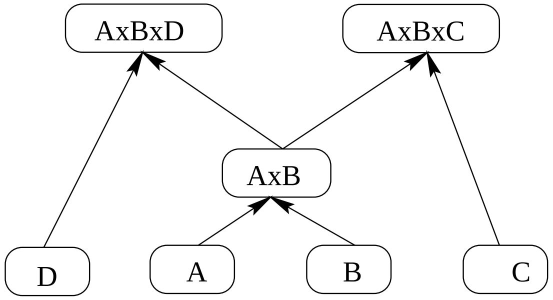


Figure 2: After MQO

Pipelining in MQO

- Pipeline intermediate shared results rather than materializing.

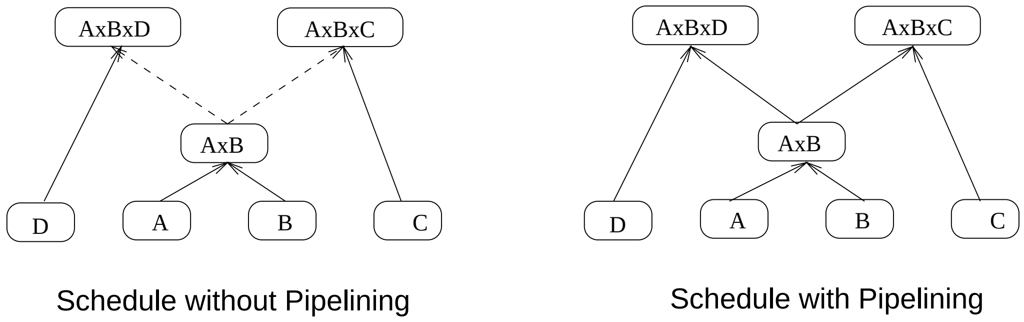


Figure 3: Motivation for pipelining

- Saves materialization cost.

Approach to Pipelining in MQO

- 1-phase optimization too expensive.
- 2-phase : incorporate pipelining in a given MQO plan(Plan-DAG).
- Annotation of Plan-DAG
  - Mark edges as pipelinable or materialized.
  - Operator at output of pipelinable edge produces tuples as it consumes input and reads it once.

Approach to Pipelining in MQO(contd)

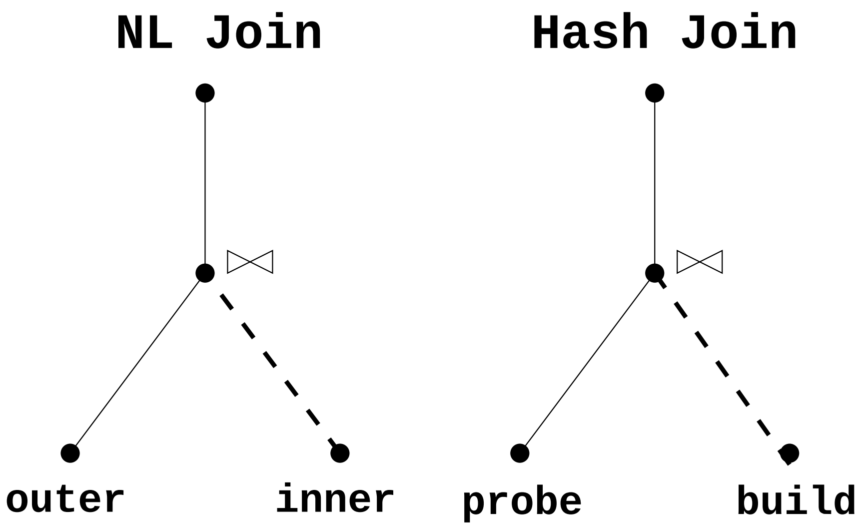


Figure 4: Examples of Pipelinable Edges

Problems in Pipelining

- Not all pipelinable edges pipelined.
- Basic assumption of limited buffer size.
- Buffer overflow, deadlock.

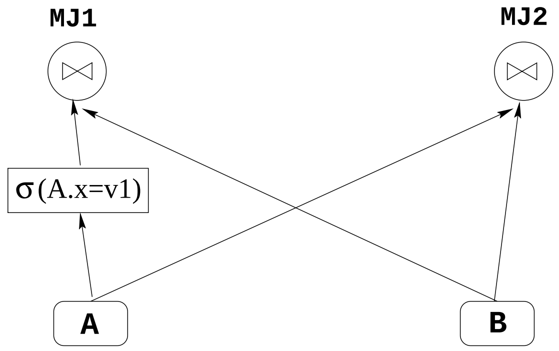


Figure 5: Unrealizable Schedule

Problems in Pipelining

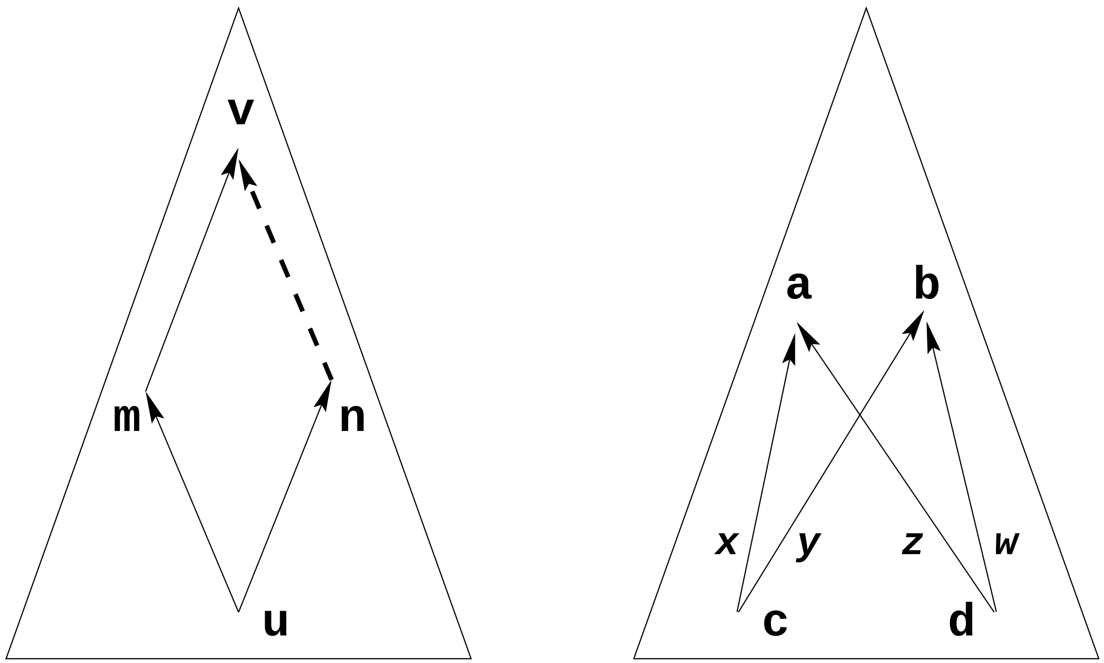


Figure 6: Unrealizable Schedules

A model for Pipelining in MQO

- *(Pipeline Schedule)* A Plan-DAG with each edge labeled either pipelined or materialized.
- *(Bufferless Pipeline Schedule)* For a pipeline schedule given $f(e, x)$, there exists $P(e, t)$, such that for every node n, with outgoing edges $o_1, o_2, \cdots o_k$, and incoming edges $e_1, e_2, \cdots e_k$, following conditions are satisfied.

  (i) $P(o_1, t) = P(o_2, t) \cdots = P(o_k, t) = P(n, t)$

  (ii) $P(e_i, t) = f(e_i, P(n, t))$, $\forall\ i$.

  (iii) $\exists\ T$ such that $\forall\ n$, $\forall t \geq T$, $P(n, t) = CARD(n)$.

  – $f(e, x)$: max tuple that n needs to produce x^{th} tuple.

  – $P(e, t)$: max tuple pipelined through e at or before t.

A model for Pipelining in MQO(Contd)

- (Valid Pipeline Schedule) Plan-DAG can be given an integer *S(n)*, referred to as the *stage number*, satisfying the following property: If n is a node, with children $a_1, a_2, \cdots a_k$, and corresponding edges $e_1, e_2, \cdots e_k$ following conditions are satisfied:

  (i) If e_i is labeled *materialized*, then $S(a_i) < S(n)$

  (ii) If e_i is labeled *pipelined*, then $S(a_i) = S(n)$

  - Operators having same stage number executed simultaneously.
  - All operators of a stage will finish before execution of higher stage starts.

A model for Pipelining in MQO(Contd)

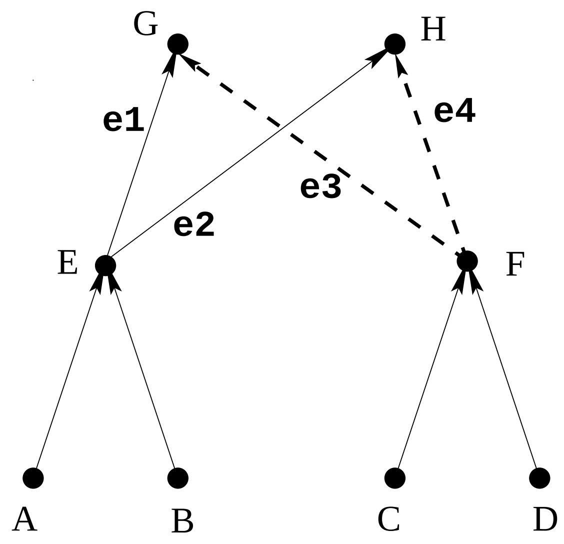


Figure 7: A plan-DAG and pipelining schedule

Criterion for Validity

- *(C-cycle)* Edges in Plan-DAG form C-cycle if they form a simple cycle when Plan-DAG considered as undirected.
- *(Opposite edges)* Two edges in a C-cycle are opposite if traversed in opposite directions.
- *(Constraint Dag)*
  - Equivalence relation $\sim$ on the vertices: $v_1 \sim v_2$ if there exists vertices $v_1 = a_1, a_2, \cdots a_n = v_2$ and there is a pipelined edge between a_i and a_{i+1} for each $1 \leq i < n$.
  - Let $E_q = C_1, C_2 \ldots C_k$ be the set of equivalence classes of $\sim$. Constraint Dag : edge from C_i to C_j if there exists v_i and v_j such that $v_i \in C_i$, $v_j \in C_j$ and there is a path from v_i to v_j.

Criterion for Validity(contd)

- (Main Theorem) Given Plan-DAG, a pipeline schedule is valid iff every C-cycle satisfies the following condition: *there exist two edges in the C-cycle both of which are labeled materialized, and are opposite.*
- Least Cost Pipeline Schedules
  - Given Plan-DAG find least cost pipeline schedules.
  - NP-complete, Greedy algorithm.

Cost Model

The cost MC(S) associated with a schedule S is,

$$MC(S) = \sum_{n \in V(S)} (WC(n) + Matdeg(n) * RC(n))$$

where,
$V(S)$ = nodes with at least one outgoing materialized edge
$Matdeg(n)$ = number of materialized edges coming out of n
$WC(n)$ = Read cost of n
$WC(n)$ = Write cost of n

Problem Definition

PROBLEM: Given a query, find a pipelinable schedule of least cost which is *valid*.

Two approaches:

- one-phase approach
- two-phase approach

NP-complete
Polynomial time reduction to EXACT-COVER problem.

Greedy Heuristic

$Merge(e)$: If $e = (n_1 n_2)$, and there is a path from n_1 to n_2 other than edge e, $Merge(e)$ fails. Otherwise, it merges edge e.

If in any Plan-DAG, there is an edge e from n_1 to n_2, then the following hold:

- e can be pipelined in a valid schedule only if the operation $Merge(n_1, n_2)$ is successful.
- A valid pipeline schedule of the Plan-DAG formed after merging, together with pipelining e, gives a valid pipeline schedule for the original Plan-DAG.
- Any valid pipeline schedule for the original Plan-DAG can be achieved through a sequence of merge operations.

$findBestGreedy(dag)$

begin

  $E \leftarrow$ set of all edges of dag

  $E_m \leftarrow \phi$

  for $e \in E$ **do**

    if $Merge(e)$ is possible **then**

      Add e to set E_m

    end if

  end for

  if $E_m = \phi$ **then**

    return

  end if

  $e \leftarrow$ edge in E_m with highest benefit

  output e as pipelined

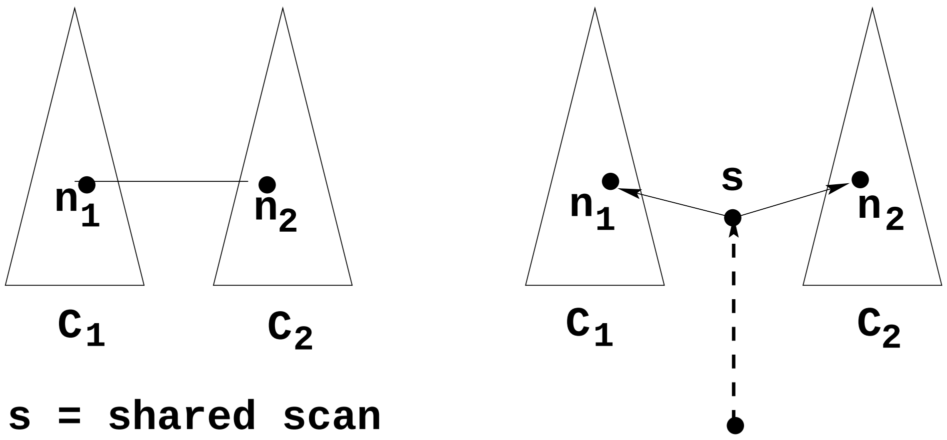
  $dag1 \leftarrow dag$ after Merge(e)

  call $findBestGreedy(dag1)$

end

Shared Read

Shared read optimization reduces the number of reads of materialized results.



s = shared scan
m = materialized node

- Add edges for shared reads to Constarint graph
- Must be acyclic for feasiblity
- Polynomial Kruskal like algorithm on Constraint graph

Performance Study

- Implemented by extending an existing Volcano-based multi-query optimizer.
- Evaluated on TPCD benchmark queries.