Parallel query processing in PostgreSQL

Daniel Vojtek

12.2.2009

Content

- Motivation
- Query processing in PostgreSQL
- Introduction to parallelization
- Parallel processing of subquery
- Sorting
- Our approach and work
- Problems with parallelization

Motivation

- Databases are larger and larger
- More effective usage of resources
- More and more CPUs on one machine
- Speed up in query execution (linear)
- Scale up (linear)

Query processing

- For each session PostgreSQL creates one backend process
- Processing query then involves:
  - Parsing
  - Apllying rewrite rules
  - Creation of optimized execution plan
  - Executing the plan
  - Utility Processing (for DDL)

Parallelism in DB

- Usage of multiple CPUs to perform parts of a single task
- Interquery parallelism – parallelism among queries – already in PostgreSQL
- Intraquery parallelism – operations within query are executed parallely
  - Intraoperation - parallel subqueries
  - Interoperation – parallel sort

Intraquery - interoperation

- Pipelining – output records of operation A are consumed by a second operation B, even before the first operation has produced the entire set of records
  - Saves space by not storing complete intermediate results.
- Independent – operations do not depend on each other – multiple joins (4 = 2 + 2)
- Mixed – more practical solution

Intraquery – interoperation cont`d

- Planner produces tree of plan Nodes
- No support of parallelism in planner
  - Executor decides which branches of plan tree to execute in separate thread
- Smart planner
  - Adds new Parallel Nodes to plan
    - Distribute – single input, multiple output
    - Gather – multiple output, single input
  - Rejects to use parallelization for simple queries
  - Optimizes parallelization

Intraquery - intraoperation

- Parallel sorting – in memory quicksort
- Divide and conquer strategy – divides list into two sublists
- Sublists can then be processed by separate threads
- After sublists are sorted there is no need for synchronization – sort is finished
- Without preprocessing there is a linear speedup

Other tasks

- Parallel plan scoring
  - Planner can search more of the plan space
  - Search for optimal plan is NPC problem
- Index rebuilding
  - When they spawned many levels or have many deleted leaf rows
  - Importing large warehouse tables
- Partitioned tables
  - Parallel processing of partitions

Our approach

- Implement intraquery parallelization with threads
- Create global pool of threads for each backend, so different phases of query processing can use it

Problems

- Technical:
  - PostgreSQL code is not thread safe
  - Signal handling
- Logical: Structures like Locks are per process based. Deadlock management. Decision about parallelism in planner or in executor
- Support of threads differs on OS
  - POSIX threads
  - WinThreads

Competition

- Oracle
  - Large support of parallelism
  - Parallel hint for queries, parallel index, partitions
- MS SQL
  - Index rebuilding, parallel query support for partitions
- DB2
  - Parallel query, partitions.

Summary

- Speed up and scale up for processor-intensive queries
- Intraquery paralllelism
- Implemented with threads
- Work in progress

Sources

- PostgreSQL source code
- High Performance Parallel Database Processing and Grid Databases - David Taniar

Q&A