MonetDB/X100: Hyper-Pipelining Query Execution *

Peter Boncz, Marcin Zukowski, Niels Nes
CWI, Amsterdam, The Netherlands
{P.Boncz,M.Zukowski,N.Nes}@cwi.nl

Extended abstract

Most database systems follow the traditional Volcanoe engine execution model. In this strategy, a query consists of a tree of relational operators that process data in a consumer-producer fashion. During each step, a single tuple, consisting of all relation’s attributes, is sent through the execution pipeline.

Since introduction of Volcano we have seen huge changes in computer architecture, especially in memory access model and CPUs features. First of all cache memories have been introduced to improve the memory access for the ‘hot-set’ data and instructions. As for the CPUs they have evolved from architectures with a single execution unit to now available hyper-pipelined CPUs. These consist of multiple execution units, each processing instructions in a pipeline consisting of many steps (e.g. 3-way 20-steps Pentium4 CPUs).

The Volcano approach, while simple and elegant, turns out to be inefficient on such modern computer architectures. It suffers from a number of problems, including: significant per-tuple CPU overhead, poor instruction-cache utilization, expensive lookup of attributes within a tuple, and complex code that runs slow on today CPUs. Investigating the impact of these problems has shown that modern DBMSs achieve very low IPC (instructions-per-cycle), mainly due to memory stalls and inefficient code.

MonetDB is a main-memory database system using decomposed storage model (DSM) with separate physical tables for each relation’s attribute. Queries are iterative programs consisting of simple, specialized operators. The result of one execution step is fully materialized before next step proceeds. This approach, although quite successful in dealing with issues present in Volcano model, introduces new problems, two of which are the most important. First of them is the main-memory limitation and the second is an extra cost of between-operator materialization.

X100 is a new experimental execution engine basing on the MonetDB system. It tries to combine benefits of both Volcano and MonetDB approaches to maximize performance on modern hardware. The most important design ideas of X100 include:

- a query is decomposed into a tree of relational algebra operators like in Volcano.
- data is processed in vectors - vertically partitioned fragments of a single attribute (i.e. small arrays) that are small enough to fit in the cache (ca. 1000 entries). This allows operators to read data produced below in the tree directly from the cache without materialization.
- relational operators use simple, highly-optimized primitives that expose to the compiler that work on the tuple is independent of the previous and the next tuples. This allows compilers to apply loop-pipelining, causing X100 primitives to achieve high IPC. Most (> 90%) of the execution time is spent in these primitives.

While X100 is still in an early stage of development, we have managed to run the entire 100GB TPC-H benchmark on it (using manual SQL to X100 translation). First results show that it is already 10-50 times faster than other DBMSs on most queries.

We currently work on extending the X100 execution engine with missing features. Moreover, we plan to integrate it with a dedicated storage layer to further increase performance of non-memory-based queries.

*Full version of this article will be published at the CIDR 2005 conference, 4-7 January 2005, Asilomar, CA, USA