The Data Cyclotron: Juggling data and queries for a data warehouse audience
Goncalves, R.

Citation for published version (APA):
Pereira Gonçalves, R. A. (2013). The Data Cyclotron: Juggling data and queries for a data warehouse audience

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 1

Introduction

1.1 A new Era

"The purpose of life is to obtain knowledge, use it to live with as much satisfaction as possible, and pass it on with improvements and modifications to the next generation”, Apoorva Patel in [121]. Over billions of years, biological evolution has experimented a wide range of physical systems for acquiring, processing and communicating information [121].

The optimization principles behind those systems, i.e., minimization of errors and minimization of physical resources are a source of innovation for computing technology. Their goal has inspired and lead us to a new paradigm in Science, data-intense science [77]. A paradigm beyond experimental, theoretical research, and computer simulation.

From the world of science to the competitive world of Business Intelligence (BI) data has been collected, stored, and processed in any possible way. From simulation, experimentation, or simple logging of human and machine activity, the amount of data is growing exponentially. It is clear that information is becoming more relevant for our society evolution than ever. It has become known to the public as the BigData Era.

1.2 Challenges

Collecting data is only one step in a scientific investigation, and scientific knowledge is much more than a single compilation of data points. The world is full of observations to be made, but not every observation constitutes a useful piece of knowledge.
All scientists make choices about which data is most relevant to their research and what to do with it. In a continuous cycle, they try to turn a collection of measurements into a useful data set through analysis, and how to interpret it in the context of what they already know. The thoughtful and systematic collection, analysis, and interpretation of data allows a collection of measurements to be converted into evidence that supports scientific ideas, arguments, and hypotheses.

The process of data collection, analysis, and interpretation happens on multiple scales. It occurs over the course of a day, a year, or many years, and may involve one or many scientists whose priorities change over the time.

At the primary stages, huge amounts of raw data is collected to be scanned and filtered to remove noise or irrelevant properties. The filtering stage is followed by a simple aggregation phase to detect if the data is meaningfully or not. With a single scan simple conclusions are induced from this type of analyze. However, complex analysis requires several data scans to adjust the search parameters to fine tune model parameters.

The search moves from one database subset to another, zooming in and out until the complete data space has been explored. A good example is the search of a new galaxies in the universe. Using a catalog of the space, such as SkyServer catalog [137, 67, 82], astronomers request a region of the Universe to perform the search. On each region he extracts another sub-set or directly applies a complex and fine grain investigation. Each region is designed as context of interest.

The constant shift of focus leads to a short retention period for data- and workload-allocation decisions which may cause the resource utilization to deteriorate. These applications require a generic and simple data-access as well as flexibility to change the search space, reallocate resources on demand, and high throughput with modest response time.

Hence, a grand challenge for computer scientists is to devise a self-organizing architecture which exploits all hardware resources for the current workload, achieves an accurate database subset definition, minimizes response time, and maximizes throughput without a single point for global co-ordination.

We have accepted the challenge and in this dissertation we propose a novel architecture, the Data Cyclotron (DaCy), to exploit new trends in network hardware. It addresses the challenge using a turbulent data movement through a storage ring built from distributed main memory. Computations assigned to individual nodes interact with the storage ring by picking up data fragments, which are continuously flowing around, i.e., the hot-set.

---

1A computation is a database query or a job defined by a user through the application integrated with Data Cyclotron.
The Data Cyclotron detaches itself from an old premise in distributed query processing, i.e., most distributed database systems from the past concentrate on reducing network cost. This was definitely a valid approach for the prevalent slow network connections. However, the continuous technology advancement calls for reconsideration of this design guideline.

Nevertheless, the rich research history for Database Management Systems (DBMSs) should not be ignored. On each Data Cyclotron node, a DBMS kernel is integrated to provide efficient techniques for data manipulation. With this processing efficiency and a symbiotic relationship with the network hardware trends, the Data Cyclotron stands in middle of state-of-the-art solutions as a novel approach for distributed data processing.

1.3 DataBase Management Systems history.

For more than three decades the manipulation of data stored in databases has been carried out by Database Management Systems (DBMS). The database term was introduced in mid-1960s at the same time direct-access storage for shared interactive use was made available [146].

The earliest DBMS, beyond efficiency, aimed to make the data independent from the logic of application programs, i.e., to split the logical schema from the storage schema, so the same data could be made available to different applications. An important feature to keep the user away from handling query plan optimization and resources allocation. A feature sometimes neglected by modern solutions, such as MapReduce, as we will later explain.

The DBMS had evolved from navigational systems, with Hierarchical and Codasyl model (Network model) as the most adopted ones, to relation systems. Due to the heavy demands on processing resources, relational systems only from the mid 1980s onward became widely deployed, i.e., computing hardware became powerful enough to sustain its heavy demands. Their dominance for all large-scale data processing applications became clear in the early 1990s and they still remain dominant in most areas.

A DBMS has evolved into a complex software system with the introduction of parallelism to load data, to build indexes, and to evaluate queries. Parallel DBMS improved query processing and data access latency by using multiple central processing units (CPUs) (including multi-core processors) and parallel storage. Contrary to sequential processing, with parallel processing operations are performed with overlap in time. Even in our days they still stand as one of the most efficient systems to scale vertically.

Though the parallel DBMS seek to improve performance in a single machine, i.e.,
scale up, they were not designed to scale out. Hence, distributed query processing was introduced through Distributed Database Management Systems (DDBMS) as early as the seventies.

The motivation for distributed query processing is to efficiently exploit a large resource pool: $n$ nodes can potentially handle a larger workload more efficiently than a single node. The challenge is to make the queries meet the required data somewhere in the network, taking into account various parameters crucial in a distributed setting, e.g., network latency, load balancing, etc.

The state of the art for distributed query processing has evolved from static schemes over a limited number of processing nodes to architectures with no single point of failure, high resistance to network churn, flexible replication policies, efficient routing protocols, etc. [46, 99, 150].

1.4 State of the art

In the last decade several solutions have emerged to cope with new challenges for data warehouses, e.g., data extraction and online analysis on data sets that are growing into the multi-petabyte range. Most of these challenges are being met by high-performance data warehouses, Hadoop implementations, or data integration technologies. Each of them is designed to be efficient in one of the data processing stages, i.e., data extraction, data preparation, or data presentation.

The new solutions have started to bring advanced analytic computation closer and closer to the data. Such an approach is giving rise to in-database analytics. Predictive analysis, data mining, and other compute-intensive analytic functions have been migrated from separate, standalone applications into the enterprise data warehouse taking advantage of its full parallel-processing, scalability, and optimization features.

However, the solutions designed follow the old trends in network hardware. They were created for the inflexible tree structure used in data centers to optimize cost efficiency and access to specific outside data sources. Despite the improvements to adapt them to new data-center trends, their architectural design is still tied on two orthogonal, yet intertwined issues: data allocation and workload behavior.

In most architectures a node is made a priory responsible for a specific part of the database using, e.g., a key range or hash function. The query optimizer exploits the allocation function by contracting sub queries to specific nodes or issuing selective data movements and data replication between nodes. Unfortunately, the optimizer search space increases significantly too, making optimally exploitation of resources harder. This static data allocation calls for a predictable workload to advise optimal system configurations.
Workload behavior is, however, often not predictable. Therefore, an active query monitoring system is required, e.g., a database design wizard [34, 152, 103] to advice on indexes and materialized views, followed up with scheduled database maintenance actions. The problem is magnified in the presence of a skewed query workload in combination with a poor data allocation function. Furthermore, workload characteristics are not stable over time either. The workload continuously shifts from one part of the network to another making optimal data allocation scenarios extremely hard to find due to the high cost to reshuffle data kept on disks.

The ultimate goal is to design a self-organizing architecture, which maximizes resource utilization without global coordination, even in the presence of skewed and volatile workloads [34]. There is certainly room to design new data allocation functions [69], grid-base algorithms [25], distributed optimization techniques, and associated workload scheduling policies [107].

Several companies, e.g., Greenplum, Asterdata, Infobright, are designing new solutions to exploit large clusters and compute cloud infrastructures to increase the performance for business intelligence applications using modestly changed commodity open-source database systems. Even reduced, but scalable database functionality is entering the market, e.g., SimpleDB (Amazon) and Bigtable (Google).

However, it is clear that following a research track explored for three decades gets us only up to a certain point regarding improved resource utilization. We end up in a local optimum dictated by software inertia to follow the prime hardware trends such as increased CPU performance and disk bandwidth. It obscures the view of possible better approaches, if such a solution exists. Therefore, new network hardware trends should also be considered on the design of the optimal solution.

1.4.1 Hardware evolution

From sequencing genomes to monitoring the Earth’s climate, from stock-market streams manipulation to search on social media many recent scientific and economical advances would not have been possible without data manipulation systems assisted with a parallel increase of computing power and network bandwidth.

The computing technology evolution is breaking old premises and assumptions used on the design of systems to cope with the data burst. New trends are emerging to manipulate and to explore massive data sets. Machines from one to multi-core processors, large main-memories, and high bandwidth for data-access are challenging the design of new ideas and reviving ideas once neglected.

In short-term view, network connections of 40 Gb/sec (3.7 GB/sec Ethernet) and
Remote Direct Memory Access (RDMA)\textsuperscript{2} technology enables fast transfer of complete memory blocks from one machine to another in a single (fast) step. They are making massive processing, huge main memories, and fast interconnects affordable for experimentation with novel architectures [65]. Furthermore, thanks to network hardware road-map, the data-centers topology tends to be more flexible and heterogeneous. They are now built with intuition that customers constantly require different processing power and data access speed for each application they run.

1.5 The Data Cyclotron

In the spirit of an earlier attempt in the database field [75, 15], the Data Cyclotron architecture [63] was designed. It is a different approach for distributed query processing by reconsidering de-facto guidelines of the past [92]. It provides an outlook on a research landscape barely explored.

With the outlook of RDMA and fast interconnects, the Data Cyclotron is designed around processing nodes, comprised of state-of-the-art multi-core systems with sizable main memories, RDMA facilities, running an enhanced database management system, and a Data Cyclotron service.

1.5.1 Architecture

The Data Cyclotron service organizes the processing nodes into a virtual storage ring to send the database hot-set continuously around \textsuperscript{3}. A query can be executed at any node on the ring; all it has to do is to announce its data interest and to wait for the data to pass by.

This way, continuous data movement becomes the Data Cyclotron key architectural characteristic. It is what database developers have been trying to avoid since the dawn of distributed query processing. It immediately raises numerous concerns and objections, which are better called research challenges. The query response time, query throughput, network latency, network management, routing, etc., are all areas that call for a re-evaluation.

We argue that hardware trends call for such a fresh, but unorthodox look and match it with a thorough assessment of the opportunities they create. Hence, the Data Cyclotron uses continuous data movement as the pivot for a continuous self-organizing system with load balancing.

\textsuperscript{2}http://www.rdmaconsortium.org/

\textsuperscript{3}For convenience, and without loss of generality, we assume that all data flows in the same direction, say clockwise.
1.5.2 **Hot-Set management**

Adaptation to changes in the workload is often a complex, expensive, and slow procedure. With the nodes tightly bound to a given data set, we first need to identify the workload change and the new workload pattern, then assign the new responsibilities, move the proper data to the proper nodes, and let everyone know about the new distribution.

In the Data Cyclotron a workload change affects the *hot-set* data on the ring, which is triggered by query requests for data access. Its self-organization in a distributed manner, keeping an optimal resource utilization, replaces gradually the data on the ring to accommodate to the current workload. The data, depending on its relevance for the workload, is kept in one of the three data sets, the *cold-set* (data in disk), *warm-set* (data in memory, but not in circulation), and the *hot-set* (data in circulation).

The integration of an innovative cache management reduces the number of times a data chunk commutes between the different data sets. Hence, high query throughput and low query response time is assured.

1.5.3 **Efficient computation**

For efficient computation the Data Cyclotron uses a main-memory column-store, known for its I/O efficiency. Efficient algorithms from the DBMS kernel are used for data processing. External libraries can also be integrated to explore different models of parallelization such as the MapReduce model for multi-core machines [88, 125].

Before compilation a computation ⁴ searches a lightly loaded node to execute on; the data needed will pass by upon request. This way, the load is not spread based on data assignment, but purely on the node’s resources and on the storage ring load characteristics. This innovative and simple strategy intends to avoid *hot spots* that result from errors in the data allocation and computation plan algorithms.

Computations are compiled into an acyclic graph. Without much effort, the acyclic graph is split into independent sub-graphs to consume disjoint data subsets and exploit new multi-core processors to scale up. At execution time it is assured, by the Data Cyclotron, the data requested is available in memory.

1.5.4 **Vision**

The decentralized Data Cyclotron architecture allows us to have seamless transition between a scale up solution to a scale out solution, or the best case scenario, have both

---

⁴A computation is a query, or a job defined by a user through the application integrated with Data Cyclotron.
in harmony. Based on the resources of each unit, the DaCyDB has the flexibility to explore both models in different degrees to improve throughput and give response time guarantees.

With access to all data in all nodes, computation is split into sub-computations which are distributed among the nodes to explore distributed processing. They are sent to the ring and each of them settles on a different node following the basic procedures of a normal computation (cf., Section 1.5.3).

Sub-computations are processed concurrently and the individual intermediate results are then combined to form the final result. Such freedom gives the grounds for an application to scale out without be bounded to any complex scheduling algorithm or data distribution scheme.

Multi-computation processing can be boosted by reusing sub-computations result to avoid (part of) the processing cost, i.e., they are simply treated as persistent data and pushed into the storage ring for interested computations. Like base data, intermediate results keep flowing as long there is interest.

Based on the amount of computations, data flowing, and nodes load, the Data Cyclotron adapts the ring size to seek the optimal point for throughput. It uses a pulsating ring, one that adaptively grows and shrinks to find the optimal number of nodes comprising a ring and the optimal flow of data. A pulsating ring does not rely on a central coordinator, which makes the architecture flexible and robust to different workload resource demands. All decisions are independently done at each node. It is a new concept to design fully flexible distributed query processing architectures.

Moreover, it is conceivable that multiple pulsating rings live in the same physical cluster. At any point in time, such a mesh consists of numerous overlapped pulsating rings. Each ring focus on a given subset of the workload or database. Rings appear, grow/shrink, and disappear as the workload changes.

1.6 Synopsis

The scientific contributions of this dissertation have a dedicated Chapter each and they are all supported by publications in major international refereed venues.

Preceding each individual contribution, we present Chapter 2. This Chapter gives an overview of related concepts, and the background and contextualization for an independent read of the remain Chapters. It identifies data center trends followed by a short introduction on the state-of-the-art network hardware.

In all Chapters we have anchors to previous fractions of text to revive a concept or refresh the readers memory for a better contextualization of an idea.
The anchors are distinguished from the remain text as this paragraph is from the synopsis text. An anchor is preceded by the section identification and can be ignored in case the reader recalls the context.

1.6.1 Contributions

The scientific contributions of this dissertation can be summarized as follows:

1. The Data Cyclotron architecture, Chapter 3. The Chapter introduces the logical topology used by Data Cyclotron and its symbiotic relationship with current data centers and their new trends. Consequently, the three layers that compose a Data Cyclotron node, i.e., network layer, DaCy layer and application layer, are described in detail as well as their interaction. The Chapter ends with a description on how nodes interact to achieve high bandwidth and efficient memory utilization.

2. The Data Cyclotron hot-set management, Chapter 4. The Chapter presents and provides an in depth analysis, through simulation, of algorithms and protocols for an efficient hot-set definition. With uniform and non-uniform workload scenarios, we identify each hot-set management aspect that maximizes throughput and minimizes data access latency. Supported by observations collected during simulation, we propose optimizations and improvements to make the Data Cyclotron more adaptive to a diverged set of workloads and applications. Some proposals are then used and evaluated with a full functional system.

3. DBMS integration with the Data Cyclotron: DaCyDB, Chapter 5. The Chapter describes the conceptualization of a full functional system called DaCyDB. It shows the harmonious integration of a column-store DBMS with the Data Cyclotron. The integration and efficient parallelism is studied using two different clusters. Through micro-benchmarks and a well-known decision support benchmark, TPC-H, each layer in the architecture is tested for different amounts of resources. The Chapter concludes with a description of the features which equip DaCyDB with iterative data loading and different levels of data consistency.

4. Vision for DaCyDB, Chapter 6. The Chapter lays down the DaCyDB design details which make it an outstanding solution for distributed data analysis on huge data warehouses. The description goes from multi-query parallelization to scalable processing of complex queries and from pulsating rings for dynamic

---

5 Also referred as peer.
6 www.tpc.org
ring size adjustments to the co-existence of multi-rings within a single cluster. Furthermore, it positions DaCyDB with respect to other state-of-the-art systems for big data processing such as MapReduce frameworks emphasizing the advantages and disadvantages for each data processing phase. The Chapter concludes a thoughtful analyze of on going research for energy efficiency and describing how green is the DaCyDB architecture.

The dissertation is concluded with a wrap up of the main ideas and contributions in Chapter 7.

1.6.2 Publications

The work presented in this dissertation has been the basis for several publications in major international refereed database and distributed systems venues.


5. Romulo Goncalves, Martin Kersten. An elastic system for distributed data analysis. Submitted for publication at the moment of printing this dissertation.