Processing Big Data: towards a novel, efficient and smart partitioning approach

Jorge GALICIA
jorge.galicia@ensma.fr

Supervised by:
Ladjel BELLATRECHE
Amin MESMOUDI

Sept 13, 2018
Outline

1 Introduction
   - Big Data
   - Cluster Computing

2 Related Work
   - Relational Model
   - RDF Model

3 Our Approach
   - Presentation
   - System overview
   - Implementation

4 Experiments

5 Perspectives

6 References
Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

1880
U.S. Census
1944
Growth of American university libraries
1961
Scientific knowledge expansion
1966
Centralized Computing Systems enter the scene
1970
Relational Database by Edgar F. Codd
1974
IBM System R system built as a research project
1978
Oracle launches the first commercially available RDBMS
1983
Information Growth

Introduction

**Big Data**

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

1880 | U.S. Census

8 years to tabulate!

**Introduction**

**Big Data**
Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

1880  •  U.S. Census
1944  •  Growth of American university libraries

"Yale Library in 2040 would have approximately 200,000,000 volumes, which will occupy over 6,000 miles of shelves"

Introduction

**Big Data**
Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

1880  U.S. Census
1944  Growth of American university libraries
1961  Scientific knowledge expansion
1966  Centralized Computing Systems enter the scene

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

1. Introduction

**Related Work**

- U.S. Census (1880)
- Growth of American university libraries (1944)
- Scientific knowledge expansion (1961)
- Centralized Computing Systems enter the scene (1966)
- Relational Database by Edgar F. Codd (1970)

1.1. Databases and Normal Forms

1.1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large bodies of formatted data. Except for a paper by Codd [1], the principal application of relations to data systems has been in deductive question-answering systems. Lavoie and Marx [2] provide maximum reference to work in this area.

In contrast, the problem that we face is that of data independence—the independence of application programs and terminal activities from changes to data types and changes in data representation—and certain kinds of data anomalies which are expected to become troublesome even in traditional systems.

1.1.2. Observing Dependence

Elements of data in a data bank may be shared in a variety of ways, some tending to concern ordering, some permitting each element to participate in one ordering only, where permitting each element to participate in several orderings. Let us consider those ordering systems which allow query or report data, elements to be stored in an least on total ordering which is locally associated with the function determined ordering of subjects. For example, the results of a life insurance part may be stored in ascending order by part serial number, data systems normally permit application programs to access the order of presentation of results from such a file to be identified by (or be a subordering of) the

Introduction

Big Data

Datasets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze [1].

- U.S. Census
- Growth of American university libraries
- Scientific knowledge expansion
- Centralized Computing Systems enter the scene
- Relational Database by Edgar F. Codd
- IBM System R system built as a research project
- Oracle launches the first commercially available RDBMS
- Information Growth

Introduction

1988 ● Data warehouse foundations

An architecture for a business and information system

by Barry Devlin and Paul Murphy
Introduction

1988
- Data warehouse foundations

1989
- Business Intelligence 1.0

1993

Providing OLAP to User-Analysts: An IT Mandate

Introduction

Overview

Recently, there has been a great deal of discussion in the trade press and elsewhere regarding the coherence of so-called transaction databases with decision support systems. These discussions usually revolve around the argument that the physical design required for acceptable performance of each is incompatible and that therefore, data should be stored redundantly in multiple enterprise databases: one for transaction processing, and the other for decision support type activities. Also, these arguments usually confuse physical schema with logical and conceptual schemas.

These arguments are fuzzy and imprecise. These arguments ignore the fundamental requirements of the type of analytical data models required for efficient information synthesis and also ignore the fact that the majority of enterprises have numerous, diverse data stores from which information needs to be synthesized. This paper defines a category of database processing: On-Line Analytical Processing (abbreviated OLAP). This paper defines the OLAP category, describes an enabling architecture for OLAP and identifies the fundamental components and criteria for evaluating a given product’s efficacy in support of the OLAP category. Finally, a commercially available product is evaluated according to the rules for OLAP. In this paper the symbol DBMS denotes a database management system.

E.F. Codd, S.B. Codd and C.T. Salley

E.F. Codd Associates

by Codd, E.F., S.B. Codd and C.T. Salley
Introduction

- 1988: Data warehouse foundations
- 1989: Business Intelligence 1.0
- 1993: World Wide Web Revolution
Introduction

1988
- Data warehouse foundations

1989
- Business Intelligence 1.0

1993
- World Wide Web Revolution

1995
- Introduction of Internet of Things

by Kevin Ashton

Web 2.0
Introduction

- Data warehouse foundations
- Business Intelligence 1.0
- World Wide Web Revolution
- Introduction of Internet of Things
- Volume, Velocity, Variety
- Open source solution for big data explosion

Laney Doug, "3D Data Management: Controlling Data Volume, Velocity, and Variety"
Introduction

1988
- Data warehouse foundations
- Business Intelligence 1.0

1993
- World Wide Web Revolution

1999
- Introduction of Internet of Things

2001
- Volume, Velocity, Variety

2006
- Open source solution for big data explosion

2009
- Linked Data
Introduction

- 1988: Data warehouse foundations
- 1989: Business Intelligence 1.0
- 1993: World Wide Web Revolution
- 1995: Introduction of Internet of Things
- 1999: Volume, Velocity, Variety
- 2001: Open source solution for big data explosion
- 2006: Linked Data
- 2009: Apache Spark
- 2010: Spark

---

Jorge GALICIA  jorge.galicia@ensma.fr  LIAS - ISAE ENSMA
Introduction

- Data warehouse foundations
- Business Intelligence 1.0
- World Wide Web Revolution
- Introduction of Internet of Things
- Volume, Velocity, Variety
- Open source solution for big data explosion
- Linked Data
- Apache Spark
- New BI Trends
- SmartCities
- ...

Jorge GALICIA  jorge.galicia@ensma.fr  LIAS - ISAE ENSMA

Sept 13, 2018  4 / 40
Introduction

What about the hardware?
Introduction

What about the hardware?

Ideal database machine [16]:

- Single infinitely fast processor with an infinite memory with infinite bandwidth.
- And would be infinitely **cheap** (free)

Technology provides relatively cheap high performance processors, fast high-capacity disks, and high-capacity RAM memories.
What about the hardware?

Ideal database machine [16]:

- Single infinitely fast processor with an infinite memory with infinite bandwidth.
- And would be infinitely cheap (free)

Technology provides relatively cheap high performance processors, fast high-capacity disks, and high-capacity RAM memories.

As stated by DeWitt D. and Jim Gray in [16] the challenge is to build an infinitely fast processor out of infinitely many processors of finite speed, and to build an infinitely large memory with infinite memory bandwidth from infinitely many storage units of finite speed.
What about the hardware?

Ideal database machine [16]:

- Single infinitely fast processor with an infinite memory with infinite bandwidth.
- And would be infinitely cheap (free)

Technology provides relatively cheap high performance processors, fast high-capacity disks, and high-capacity RAM memories.

Divide a big problem into many smaller ones to be solved in parallel.
Introduction

Cluster Computing Revolution

- A cluster is a set of independent server nodes interconnected to share resources and form a single system.
- Machines close to each other and are connected with dedicated high speed LANs and switches.

Cluster Computing Revolution Diagram:

- Ideal parallel system
- Linear speedup
- Linear scaleup

Diagram showing different cluster models:

- Shared-Everything (e.g. SMP Server)
- Shared-Disk (e.g. Oracle RAC)
- Shared-Nothing (e.g. HDFS)
Cluster Computing Revolution

- A cluster is a set of independent server nodes interconnected to share resources and form a single system.
- Machines close to each other and are connected with dedicated high speed LANs and switches.

+ Simplicity, Load Balancing
- Limited extensibility, Interference → low scalability
Cluster Computing Revolution

- A cluster is a set of independent server nodes interconnected to share resources and form a single system.
- Machines close to each other and are connected with dedicated high speed LANs and switches.

+ Lower cost, high extensibility, high availability
- More complex, load balance → relies on database partitioning, new nodes → data reorganization.
Introduction

Shared-nothing architecture

- It is the most common architecture nowadays.
- In such systems the data are *partitioned* across disk storage units attached directly to each processor allowing to scan/write large amounts of data in parallel.
Introduction

Shared-nothing architecture

- It is the most common architecture nowadays.
- In such systems the data are partitioned across disk storage units attached directly to each processor allowing to scan/write large amounts of data in parallel.

Data Partitioning

- It has its origins in centralized systems that had to partition large files.
- What distribution unit should be used? Entire files/relations/graphs? → subsets of them.
- Relational model → distribute tuples.
- RDF model → distribute triples.
Partitioning is a simple concept that is easy to implement, but it raises several new physical database design issues [16]:

- Load balancing
- Data skew
- Execution skew
- Replication
- Fault tolerance
- High availability

Trade-offs:
- Parallelism can increase throughput - inter-query parallelism, decrease transaction response time - intra-query parallelism.
Partitioning is a simple concept that is easy to implement, but it raises several new physical database design issues [16]:

- Load balancing
  - Data skew
  - Execution skew
- Replication
  - Fault tolerance
  - High availability
Partitioning is a simple concept that is easy to implement, but it raises several new physical database design issues [16]:

- **Load balancing**
  - Data skew
  - Execution skew
- **Replication**
  - Fault tolerance
  - High availability

**Trade-offs**

Parallelism can increase throughput → inter query parallelism, decrease transaction response time → intra query parallelism. However:

- In complex queries: ↑ inter-query parallelism → increase total time due to synchronization overhead.
- In a system with high number of concurrent transactions: ↑ intra-query parallelism → decrease in throughput.
What about the software?
Introduction

What about the software?

Parallel database systems in shared-nothing architectures

- Pioneered by Teradata in the late 70s.
- Support standard relational tables and SQL
- The fact that the data is stored on multiple machines is transparent to the end-user.
- Most (or even all) tables are partitioned over the nodes in a cluster.
- The system uses an optimizer that translates SQL commands into a query plan whose execution is divided amongst multiple nodes.
Hadoop Map Reduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17], Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects
Introducing Hadoop MapReduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17], Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects

- Hive
- Pig
- GraphX
- Spark
- SQL
- MLib
- Other
- HBase
- TEZ
- Spark
- Spark without YARN
- YARN
  Distributed Processing Framework
- HDFS
  Hadoop Distributed File System

Partitioning in HDFS
**Introduction**

**Hadoop Map Reduce**

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17], Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

**Apache Hadoop Ecosystem and Open Source Big Data Projects**

**Partitioning in HDFS**

[Diagram showing Hadoop ecosystem and open source big data projects]
Introduction

Hadoop Map Reduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17],Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects

Partitioning in HDFS
Introduction

Hadoop Map Reduce

- OpenSource framework introduced in 2006 integrating ideas from the papers published by Google (The Google File System[17], Google MapReduce[15]).
- It provides a simple model through which users can express relatively sophisticated distributed programs.
- The input data set is stored in a collection of partitions in a distributed file system deployed on each node in the cluster.

Apache Hadoop Ecosystem and Open Source Big Data Projects

Partitioning in HDFS
# Introduction

<table>
<thead>
<tr>
<th>Parallel DBMS</th>
<th>Hadoop and others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available for almost three decades (e.g. Teradata, Aster Data, Oracle via Exadata, DB2)</td>
<td>Hadoop was introduced in 2006</td>
</tr>
<tr>
<td><strong>Schema Support</strong>: Data should be conform to a well defined relational schema.</td>
<td><strong>Schema Support</strong>: Absence of a rigid schema defined using the relational model. <strong>Parsers are needed.</strong></td>
</tr>
</tbody>
</table>
### Introduction

<table>
<thead>
<tr>
<th>Parallel DBMS</th>
<th>Hadoop and others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available for almost three decades (e.g. Teradata, Aster Data, Oracle via Exa-data, DB2)</td>
<td>Hadoop was introduced in 2006</td>
</tr>
<tr>
<td><strong>Schema Support</strong>: Data should be conform to a well defined relational schema.</td>
<td><strong>Schema Support</strong>: Absence of a rigid schema defined using the relational model. <strong>Parsers are needed.</strong></td>
</tr>
</tbody>
</table>

Is Hadoop better then?

Pavlo et al., "A comparison of approaches to large-scale data analysis", SIGMOD, 2009
### Introduction

#### Parallel DBMS

- Available for almost three decades (e.g. Teradata, Aster Data, Oracle via Exadata, DB2)

#### Hadoop and others

- Hadoop was introduced in 2006

<table>
<thead>
<tr>
<th>Schema Support</th>
<th>Parallel DBMS</th>
<th>Hadoop and others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data should be conform to a well defined relational schema.</td>
<td></td>
<td>Absence of a rigid schema defined using the relational model. <strong>Parsers are needed.</strong></td>
</tr>
</tbody>
</table>

### Is Hadoop better then?

- Pavlo et al., "A comparison of approaches to large-scale data analysis", SIGMOD, 2009

- Compared a commercial DBMS and MR across five tasks: Data Loading, Selection, Aggregation, User Defined Aggregation.

- Compared architectural differences:
  - MR: *schema later* paradigm → parsing at runtime.
  - Parallel DBMSs: *strong schema* → parsing at loading.

- The Parallel DBMS was 3.2 times **faster** than MR.

- Hadoop easier to set up, more flexible and offering a better fault tolerance.
Introduction

Parallel DBMS | Hadoop and others
---|---
Available for almost three decades (e.g. Teradata, Aster Data, Oracle via Exadata, DB2) | Hadoop was introduced in 2006

**Schema Support**: Data should be conform to a well defined relational schema. **Schema Support**: Absence of a rigid schema defined using the relational model. **Parsers are needed.**

Is Hadoop better then?

Pavlo et al., "A comparison of approaches to large-scale data analysis", SIGMOD, 2009

- Compared a commercial DBMS and MR across five tasks: Data Loading, Selection, Aggregation, User Defined Aggregation.
- Compared architectural differences:
  - MR: *schema later* paradigm → parsing at runtime.
  - Parallel DBMSs: *strong schema* → parsing at loading.
- The Parallel DBMS was **3.2 times faster** than MR. Why? Indexes, Storage Mechanisms and Distribution, Compression Techniques, Parallel Algorithms.
- Hadoop easier to set up, more flexible and offering a better fault tolerance.

We focused our research on Data Partitioning
To **enable parallelism** in shared-nothing parallel systems, data is **partitioned** and **distributed** across the nodes of the system. We explored the partitioning strategies in these systems:

- Parallel Database Systems → *Relational model*
- Distributed Triple Store Systems → *RDF model*
- Hadoop Distributed File System
To **enable parallelism** in shared-nothing parallel systems, data is **partitioned** and **distributed** across the nodes of the system. We explored the partitioning strategies in these systems:

- Parallel Database Systems → *Relational model*
- Distributed Triple Store Systems → *RDF model*
- Hadoop Distributed File System
Related Work

Relational Model

Horizontal Fragmentation

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
<td>367</td>
</tr>
<tr>
<td>4</td>
<td>A380</td>
<td>446</td>
</tr>
</tbody>
</table>
Relational Model

Horizontal Fragmentation: *hashing*

**Airbus Planes**

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
<td>367</td>
</tr>
<tr>
<td>4</td>
<td>A380</td>
<td>446</td>
</tr>
</tbody>
</table>

**Partition 1**

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>4</td>
<td>A380</td>
<td>446</td>
</tr>
</tbody>
</table>

**Partition 2**

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
<td>367</td>
</tr>
</tbody>
</table>
## Related Work

### Relational Model

**Horizontal Fragmentation:** range

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
<td>367</td>
</tr>
<tr>
<td>4</td>
<td>A380</td>
<td>446</td>
</tr>
</tbody>
</table>

Partition 1

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
<td>367</td>
</tr>
</tbody>
</table>

Partition 2

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>A380</td>
<td>446</td>
</tr>
</tbody>
</table>
Related Work

Relational Model

Horizontal Fragmentation: derived

Airbus Planes

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
<td>367</td>
</tr>
<tr>
<td>4</td>
<td>A380</td>
<td>446</td>
</tr>
</tbody>
</table>

Company A_ID Qty

<table>
<thead>
<tr>
<th>Company</th>
<th>A_ID</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air France</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Iberia</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Delta</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Partition 1

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
<td>367</td>
</tr>
</tbody>
</table>

Company A_ID Qty

<table>
<thead>
<tr>
<th>Company</th>
<th>A_ID</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iberia</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Delta</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Partition 2

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>A380</td>
<td>446</td>
</tr>
</tbody>
</table>

Company A_ID Qty

<table>
<thead>
<tr>
<th>Company</th>
<th>A_ID</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air France</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
### Related Work

**Relational Model**

**Vertical Fragmentation**

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
<td>367</td>
</tr>
<tr>
<td>4</td>
<td>A380</td>
<td>446</td>
</tr>
</tbody>
</table>

**partition 1**

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A320neo</td>
</tr>
<tr>
<td>2</td>
<td>A330-200</td>
</tr>
<tr>
<td>3</td>
<td>A350-900</td>
</tr>
<tr>
<td>4</td>
<td>A380</td>
</tr>
</tbody>
</table>

**Partition 2**

<table>
<thead>
<tr>
<th>A_ID</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>242</td>
</tr>
<tr>
<td>3</td>
<td>367</td>
</tr>
<tr>
<td>4</td>
<td>446</td>
</tr>
</tbody>
</table>
## Related Work

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Paper</th>
<th>Year</th>
<th>DB Type</th>
<th>W-A</th>
<th>R</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Based on tuning advisors using the query optimizer</td>
<td>[35]</td>
<td>2002</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>IBM - DB2 optimizer Microsoft</td>
</tr>
<tr>
<td>Custom cost function</td>
<td>[32]</td>
<td>2012</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Coord+SkF PREF</td>
</tr>
<tr>
<td></td>
<td>[44]</td>
<td>2015</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Graph-based technique</td>
<td>[14]</td>
<td>2010</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Schism System HOPE</td>
</tr>
<tr>
<td></td>
<td>[34]</td>
<td>2002</td>
<td>✓ ✓</td>
<td>✓</td>
<td>✓</td>
<td>Fractured Mirrors PAX</td>
</tr>
<tr>
<td></td>
<td>[27]</td>
<td>2010</td>
<td>✓ ✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[24]</td>
<td>2011</td>
<td>✓ ✓</td>
<td>✓</td>
<td>✓</td>
<td>PAX+HDFS Divergent design</td>
</tr>
<tr>
<td>Column Store</td>
<td>[41]</td>
<td>2005</td>
<td>✓ ✓</td>
<td>✓</td>
<td>✓</td>
<td>C-Store System MonetDB</td>
</tr>
<tr>
<td></td>
<td>[8]</td>
<td>2005</td>
<td>✓ ✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Dynamic Workload</td>
<td>[40]</td>
<td>2016</td>
<td>✓ ✓</td>
<td>✓</td>
<td>✓</td>
<td>Clays DynPart</td>
</tr>
<tr>
<td>Dynamic Data</td>
<td>[28]</td>
<td>2012</td>
<td>✓ ✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
RDF Model

1. https://www.w3.org/TR/rdf11-concepts/
Related Work

**RDF Model**

**Definition**

- RDF is a standard model for data interchange in the Web\(^1\).
- It models relational facts as a set of triples \(<\text{subject}, \text{predicate}, \text{object}>\).  

\(^1\) https://www.w3.org/TR/rdf11-concepts/
Related Work

**RDF Model**

**Definition**
- RDF is a standard model for data interchange in the Web\(^1\).
- It models relational facts as a set of triples `<subject, predicate, object>`.

**SPARQL**
- In a simple definition SPARQL is the query language for RDF data\(^1\).
- The query solution is an ordered set of bindings for `(x, y, z)` → make the SPARQL query graph homomorphic to a subgraph in the data.

\(^1\) [https://www.w3.org/TR/rdf11-concepts/](https://www.w3.org/TR/rdf11-concepts/)
Related Work

RDF Processing Approaches

Partitioning is treated individually by each system.

- Centralized Approaches
  1. Relational Mapping
     1. Direct
     2. Property table
     3. Binary table
  2. Graph processing

- Distributed Approaches
  1. On top of distributed systems
  2. Specialized RDF Systems

# Related Work

## Centralized Approaches

### Relational Mapping

**Direct**

- Sesame SQL92SAIL (Broekstra et al, 2002) [10]
- Oracle (Chong et al, VLDB 2005) [12]
- RDF-3X (Neumann T et al, PVLDB 2008) [29]
- Hexastore (Weiss C et al, PVLDB 2008) [42]

**Property table**

- Jena (Wilkinson K, HP 2006) [43]
- IBM DB2RDF (Bornea M A et al, Sigmod 2013) [9]

**Binary table**

- SW-Store (Abadi et al, VLDB 2009)[2]

## Graph Processing

- gStore (Özsu et al, PVLDB 2011) [47]
- chameleondb (Gunes A, PhD 2015)[7]
<table>
<thead>
<tr>
<th>Distributed Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distributed Systems</strong></td>
</tr>
<tr>
<td>MapReduce / HDFS : SHARD (Direct) [36], HadoopRDF (PropertyTable) [23], CliqueSquare (HashPartition) [18], EAGRE [46], PigSparql [37]</td>
</tr>
<tr>
<td>Spark : S2X (GraphX) [39], S2RDF (Core) [38]</td>
</tr>
<tr>
<td>NoSQL : H2RDF[31], JenaHBase [25], TrinityRDF [45]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specialized RDF Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDF-3X-Execution : H-RDF-3X (Metis) [22], SHAPE (Semantic) [26], WARP (W-A) [21], Dream (Full R)[20]</td>
</tr>
<tr>
<td>Others : gStore-D [33], TriAd [19], Ad-Part [6]</td>
</tr>
</tbody>
</table>
Top Down Database Design
Top Down Database Design
Top Down Database Design

Our Approach:
Physical vs Logical Data Model

The data structures used to implement data in the storage module of the system are referred to as **physical data models** (e.g. rows, columns, binary table, property table). Conceptual/Logical models are at a somewhat higher level than data structures (e.g. relational, RDF).
Physical vs Logical Data Model

The data structures used to implement data in the storage module of the system are referred to as **physical data models** (e.g. rows, columns, binary table, property table). Conceptual/Logical models are at a somewhat higher level than data structures (e.g. relational, RDF).

- Fragmentation methods for relational and graph (e.g. RDF) models are strongly dependent on the physical data model of the system (e.g. Horizontal fragmentation → physical storage organized as rows, vertical fragmentation → column store).
Physical vs Logical Data Model

The data structures used to implement data in the storage module of the system are referred to as **physical data models** (e.g. rows, columns, binary table, property table). Conceptual/Logical models are at a somewhat higher level than data structures (e.g. relational, RDF).

- Fragmentation methods for relational and graph (e.g. RDF) models are strongly dependent on the physical data model of the system (e.g. Horizontal fragmentation → physical storage organized as rows, vertical fragmentation → column store).

What if we could partition the data at a **Logical** level independent of the **physical storage** module of the system?

We defined two logical **entities** and fragment the data based on them. The allocation is done based on the connections between these entities.
Approach

Logical or Conceptual Model

**Relational model**: based on tables.

**RDF model**: directed graph triple stating that a resource has a property with certain value.
Approach

Logical or Conceptual Model

**Relational model** : based on tables.

**RDF model** : directed graph triple stating that a resource has a property with certain value.

**Our entity model** : based on the model of the system SWORD [46]. **Forward Entity** : Given a subject $s$, the set of all triples having $s$ as subject. **Backward Entity** : Given an object $o$, the set of all triples having $o$ as object.
For example,

- Let us consider the table:

<table>
<thead>
<tr>
<th>Model</th>
<th>Length m</th>
<th>Price M$</th>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A320neo</td>
<td>37.57</td>
<td>101</td>
<td>Airbus</td>
</tr>
<tr>
<td>A330-200</td>
<td>58.82</td>
<td>242</td>
<td>Airbus</td>
</tr>
<tr>
<td>A350-900</td>
<td>66.8</td>
<td>367</td>
<td>Airbus</td>
</tr>
</tbody>
</table>
Entity Creation:

- **A320neo**
  - Lenght m: 37.57
  - Constructor
  - Price M$: 101

- **A330-200**
  - Lenght m: 58.82
  - Constructor
  - Price M$: 242

- **A350-900**
  - Lenght m: 66.8
  - Constructor
  - Price M$: 367

- **A320neo**
  - Constructor
  - Price M$: 101

- **A330-200**
  - Constructor
  - Price M$: 242

- **A350-900**
  - Constructor
  - Price M$: 367
Approach

Fragmentation:

- Model
  - Constructor
  - Length m
  - Price MS

- Model
  - Constructor
  - Length m
  - Price MS
Approach

Models mapping:

<table>
<thead>
<tr>
<th>Model</th>
<th>Length m</th>
<th>Price M$</th>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A320neo</td>
<td>37.57</td>
<td>101</td>
<td>Airbus</td>
</tr>
<tr>
<td>A330-200</td>
<td>58.82</td>
<td>242</td>
<td>Airbus</td>
</tr>
<tr>
<td>A350-900</td>
<td>66.8</td>
<td>367</td>
<td>Airbus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A320neo</td>
<td>Airbus</td>
</tr>
<tr>
<td>A330-200</td>
<td>Airbus</td>
</tr>
<tr>
<td>A350-900</td>
<td>Airbus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Length m</th>
</tr>
</thead>
<tbody>
<tr>
<td>A320neo</td>
<td>37.57</td>
</tr>
<tr>
<td>A330-200</td>
<td>58.82</td>
</tr>
<tr>
<td>A350-900</td>
<td>66.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Price M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A320neo</td>
<td>101</td>
</tr>
<tr>
<td>A330-200</td>
<td>242</td>
</tr>
<tr>
<td>A350-900</td>
<td>367</td>
</tr>
</tbody>
</table>
**Approach**

**System overview**

- **Data preprocessing**
  - Input
  - Dataset
  - Logical Graph
  - Perform analytical queries of the data (Select-Project-Join, GroupBy, OrderBy) in a parallel system.

- **Fragmentation**
  - Entities
  - Group Entities with similar characteristics
  - Compressed Graph

- **Allocation**
  - Graph Partitioning
  - Replication
  - Logical Entity
  - Logical Entity
  - Logical Entity
  - Logical Entity
  - Logical Entity
  - Logical Entity
  - Logical Entity
  - Logical Entity

---

*Jorge GALICIA  jorge.galicia@ensma.fr  LIAS - ISAE ENSMA*
Implementation

Example:
Pre-processing:
Implementation

Fragmentation:
Implementation

Fragmentation:
Implementation

Allocation:
Approach

Need for re-allocation

- The preprocessing and fragmentation tasks involve a huge volume of data to process.
- Accordingly, a scalable computing paradigm e.g. MapReduce, SparkRDD is needed to process.
- Therefore, the initial data is first randomly partitioned to the computing nodes (since it is stored to the HDFS). A re-allocation of the data is performed later on.
Experiments

We created a partitioning system:

- **Data pre-processing and Fragmentation**: Scala-Spark
- **Allocation**: Scala-Spark, Metis Latest stable release (5.1.0):

YARN Cluster infrastructure (1 MN + 4DN)

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>8GB</td>
</tr>
<tr>
<td>HD</td>
<td>100 GB + 1(400 GB)</td>
</tr>
</tbody>
</table>

RDF datasets

M : millions. # S,# P,# O denote number of distinct subjects, objects and predicates.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Triples (M)</th>
<th># S(M)</th>
<th># P(M)</th>
<th># O(M)</th>
<th>Size (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yago2</td>
<td>284.30</td>
<td>10.12</td>
<td>98</td>
<td>52.34</td>
<td>42</td>
</tr>
<tr>
<td>WatDiv</td>
<td>1,092.16</td>
<td>52.12</td>
<td>86</td>
<td>179.09</td>
<td>149</td>
</tr>
</tbody>
</table>
Experiments

Results: Number of entity classes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$#\mathcal{E}_C$</th>
<th>$\overline{#\mathcal{E}_C}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yago2</td>
<td>25,511</td>
<td>1,359</td>
</tr>
<tr>
<td>WatDiv</td>
<td>96,344</td>
<td>4,718</td>
</tr>
</tbody>
</table>

Data skew problem

- We found entity classes that were bigger than the maximum size allowed for a partition.
- We adopted two strategies to re-partition:
  1. Hashing on subject.
  2. Based on the class neighbors.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#80%</th>
<th>%80%</th>
<th>$\overline{#\mathcal{E}_C}$</th>
<th>$\overline{#\mathcal{E}_C}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yago2</td>
<td>51</td>
<td>0.19</td>
<td>45</td>
<td>35</td>
</tr>
<tr>
<td>WatDiv</td>
<td>381</td>
<td>0.38</td>
<td>31</td>
<td>49</td>
</tr>
</tbody>
</table>
Experiments

We have evaluated the performance at the moment based on two metrics:

**Execution time**

Time taken in minutes for the data loading+pre-processing, fragmentation and allocation tasks.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yago2</td>
<td>14.2</td>
</tr>
<tr>
<td>WatDiv</td>
<td>&gt;20</td>
</tr>
</tbody>
</table>

The loading times reported in [3] for both datasets are much more inferior, but the tests in this paper were performed on a 12 machine cluster, each machine has a 148GB RAM and two 2.1GHz AMD Opteron 6172 CPUs (12 cores each).

**Quality of partitioning**

We count how many edges were cut between two entity classes that are assigned to two different partitions.

<table>
<thead>
<tr>
<th>5</th>
<th>10</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>M</td>
<td>LP</td>
<td>M</td>
<td>LP</td>
<td>M</td>
</tr>
<tr>
<td>62</td>
<td>65</td>
<td>48</td>
<td>53</td>
<td>47</td>
<td>60</td>
</tr>
</tbody>
</table>
Future perspectives

- Perform exhaustive experiments in a much more complex cluster of computers.
- We need to test if the partitioning algorithm helps to improve the performance to solve a workload. We have thought of testing in the gStoreD system [33] and the execution engine developed by a LIAS PhD student.
- Test our approach on the relational model mapping relations to triples.
- Explore dynamic partitioning.
References I


References


References IV


References V


Katja HOSE et Ralf SCHENKEL. “WARP : Workload-aware replication and partitioning for RDF”. In : Workshops Proceedings of the 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013. 2013, p. 1-6. DOI : 10.1109/ICDEW.2013.6547414. URL : https://doi.org/10.1109/ICDEW.2013.6547414.
References VI


Alekh JINDAL et Jens DITTRICH. “Relax and Let the Database Do the Partitioning Online”. In : Enabling Real-Time Business Intelligence - 5th International Workshop, BIRTE 2011, Held at the 37th International Conference on Very Large Databases, VLDB 2011, Seattle, WA, USA, September 2, 2011, Revised Selected Papers. 2011, p. 65-80. DOI : 10.1007/978-3-642-33500-6_5. URL : https://doi.org/10.1007/978-3-642-33500-6_5.

References VII


References VIII


References

Alexander SCHÄTZLE et al. “S2X : Graph-Parallel Querying of RDF with GraphX”. In: Biomedical Data Management and Graph Online Querying - VLDB 2015 Workshops, Big-O(Q) and DMAH, Waikoloa, HI, USA, August 31 - September 4, 2015, Revised Selected Papers. 2015, p. 155-168. DOI: 10.1007/978-3-319-41576-5_12. URL: https://doi.org/10.1007/978-3-319-41576-5_12.


Kevin WILKINSON. Jena property table implementation. 2006.


Xiaofei ZHANG et al. “EAGRE: Towards scalable I/O efficient SPARQL query evaluation on the cloud”. In: 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013. 2013, p. 565-576. DOI: 10.1109/ICDE.2013.6544856. URL: https://doi.org/10.1109/ICDE.2013.6544856.