Evaluating Fuzzy Parallel Queries in a Fuzzy Parallel Database

MILAMBU BELANGANY MICHEL [1] NTUMBA BADIBANGA SIMON[2], MBUYI MUKENDI EUGENE[3]

[1] Mathématiques et Informatique, Université de Kinshasa, Kinshasa, DRC

[2]Mathématiques et Informatique, Université de Kinshasa, Kinshasa, DRC

[3]Mathématiques et Informatique, Université de Kinshasa, Kinshasa, DRC

Abstract

In this article we present the application of fuzziness in querying imprecise databases with partitions residing on different disks of a multi-processor computer. These fuzzy queries are evaluated against conventional queries by proposed an algorithmic evaluator. For the proper evaluation, parallelism is applied to the algorithmic model in order to analyze the flexibility of fuzzy queries compared to classics that run in parallel.

Keywords: *Inaccurate database, fuzzy queries, fuzzy data, parallelism, algorithmic model.*

1. Introduction

Indeed, the mode of reasoning in fuzzy logic is more intuitive than classical logic. It allows designers to better understand natural, imprecise and difficult to model phenomena by relying on the definition of rules and membership functions of sets called "fuzzy sets".

Fuzzy sets constitute an interesting theoretical and methodological framework for the flexible interrogation of Relational Databases (RDB). In fact, associates a membership function and a linguistic label with a set or formalizes a gradual property that can then be integrated as a preference in a request addressed to a Relational DB. In this sense, fuzzy queries extend Boolean queries by taking into account tolerance and graduality in the definition of user intentions. Fuzzy logic can also brings a rich set of connectors allowing to combine these preferences in a compensatory way or not [1]. A query language called SQLf [2] has thus been proposed to exploit the expressiveness of fuzzy queries.

Thus, in a reference set E, a fuzzy subset of this reference frame E is characterized by a membership function μ of E in the interval of real numbers [0, 1] (membership degree which is the extension of the characteristic function of a classical subset). In fact, a fuzzy subset (we will say more briefly a fuzzy set) is formally defined by the application m, but to come back to the language of classical mathematics, we will speak of a fuzzy set A, and denote μA its function d 'membership [9].

2. Fuzzy Databases

Just as classical sets allow the definition of Boolean predicates, fuzzy sets (Zadeh, 1965) which describe classes of objects with vague boundaries can serve as a basis for the definition of gradual predicates.
Often, elementary fuzzy predicates correspond to natural language adjectives such as young, tall, expensive, or high. [8] A fuzzy predicate P can be modeled by a μ_P function (usually triangular or trapezoidal in shape) of one (or more) domain (s) X in the unit interval [0, 1]. The degree μ_P (x) expresses the extent to which the element x satisfies the gradual predicate P (or, equivalently, the extent to which x belongs to the fuzzy set of objects that correspond to the fuzzy concept P). An elementary fuzzy predicate can also compare two attributes using a gradual comparison operator such as plus or minus equal.

3. Fuzzy Relational Questions

The purpose of the fuzzy interrogation of precise databases is quite clear, we would like to be able to ask, for example, for the list of articles checking vague criteria such as a moderate price, good quality, etc.

It is obviously much too restrictive to ask for items lower than 100Fc, because there is a risk of excluding the one which is worth 105Fc but which would be very interesting for other criteria. We therefore see that the problems posed are once again the definition of predicates such as "expensive", "large", "good quality" ... that of vague quantifiers such as "most", "almost all", " little "... and above all the conduct to adopt for the aggregation of the criteria sought.

If the first problem is solved by the definition of trapezoidal fuzzy predicates, the second can also be solved by fuzzy sets in [0, 1].

3.1 SQLf

In SQL language, the fundamental query is:

select <object> from <relational base> where <condition> which allows to obtain the list of elements satisfying a more or less structured condition. The question can be asked according to a classification (a score) by:

select <object> from <base> where <condition 1> group by <object> having <condition 2>

The query select A from R where P in which P is a logical proposition where ¬, ∧, ∨ are interpreted as in fuzzy logic, must therefore provide a fuzzy set Q such that:

$\mu_{Q(a)}$ = sup A(x) = a min($\mu_{R(x)}$, $\mu_{P(x)}$)
Illustration: instead of an exact query such as:

select #emp from Emp where age <35 and #dep in (select #dep from Dep where budget> 100000

To obtain the young people from a set of employees whose department has a high budget, we will define the fuzzy sets 'young' and 'high' and the query:

select #emp from Emp
where age = 'young' and Emp. # dep = Dep. # dep and budget = 'high'

will give the membership function:
$\mu_{(e)}$ = min (μ_{jeune} (e.age), sup {min ($\mu_{élevé}$ (budget)), (μ= (d.#dep, e.#dep) / d ∊ Dep)

3.2 Sequential execution

This is about processing a complex query on a single processor and whose execution is too slow due to its complexity. As below:

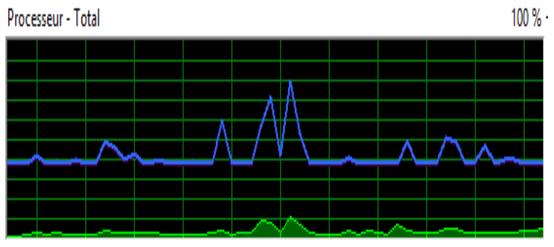


Fig 1: Sequential execution

4. Parallelism

The goal of parallel query execution is to gain performance. In addition to response time, there are several metrics to quantify the gain obtained by parallel execution. The best known are the speed-up and the scale-up. An ideal parallel system is one that achieves both linear speed-up and constant scale-up. Informally, an ideal speed-up indicates that a task can be executed twice as fast if there are twice as many hardware resources (processors, disks, memory, ...). An ideal scale-up means that twice the size of a task can be performed at the same time if you have twice the hardware resources.
The formal definition of speed-up is as follows: either a task of fixed size executed sequentially in a time Ts then executed in parallel on p processors in a time Tp, the speed-up obtained by the parallel execution is then defined by :

$$\text{speed} - \text{up}(\,) = \frac{Ts}{Tp}$$

and the speed-up is ideal if speed-up (p) = p.
We can deduce from the speed-up the efficiency Ep of the parallel algorithm, i.e. the ratio between the effective speed-up and the ideal speed-up:

$$\text{Ep} = \frac{Ts}{Tp * P}$$

4.1 Parallel execution

Thus, two types of data parallelization exist:
Parallelism of processing: The query is broken down into elementary queries which are executed in parallel on the data[13].
Data parallelism: The query is executed in parallel on subsets of the data.
Either the BDPF Relational Data Model represented in relational form and housed on several logical disks (four logical disks) of the same computer is as follows:

Store (# N ° _Mag, Name_Mag, Position_Mag, Code_Address)
Customer (#Customer_Code, Customer_Name, Category, Telephone, Address_code)

Address (#Code_Address, Town, District, Avenue, N °)
Order (# N ° _Cmd, Type_Cmd, Date_Cmd, Code_Client)
Product (#Code_Prod, Designation_Prod, Qty_Stock, Price_Prod, Qty_V, Matr_Agent, Code_Fsseur, N ° _Cmd)
Agent (#Matr_Agent, Name_Agent, Postname_Agent, Function_Agent, Department, N ° _Mag)
Supplier (#Code_Fsseur, Name_Fsseur, Category_F, Telephone_F)
Stock (N ° _Mag, Code_Prod, History_P, History_Cmd)
Consider the complex query below containing the fuzzy predicates launched on this fuzzy parallel database:

Select *
From produit, Agent, Commande, Stock, Fournisseur
---parallelism---
Where (select Designation_Prod, Qte_Stock, Prix_prod, Qté_V, Nom_Mag From Produit, Stock, Commande, Magasin Where Magasin.N°_Mag = Stock.N°_Mag AND Produit.N°_Cmd = Commande.N°_Cmd AND Produit.Code_Prod = Stock.Code_Prod AND Qte_Stock = ‘’recent’’ AND Qte_V = ‘’environ bon’’ AND Historique_Cmd = ‘’un peu ancien’’ AND Historique_P =’’ancien’’)

It will be executed on the four CPUs activated below in order to assess the execution speed of this query when it is broken down into fuzzy sub-queries r1, r2, r3 and r4 of the fuzzy query R.

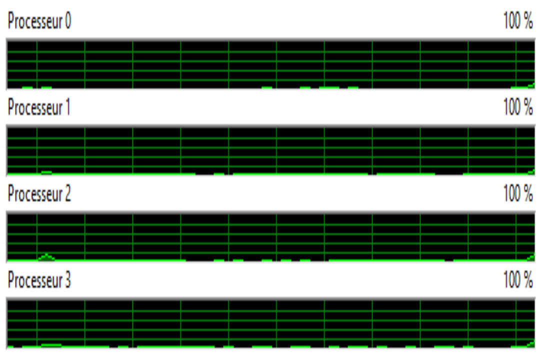
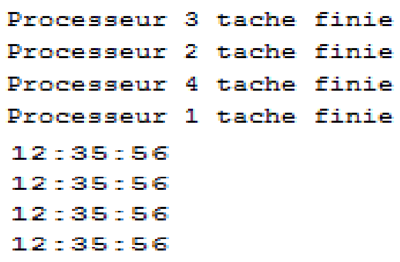


Fig 2: processors

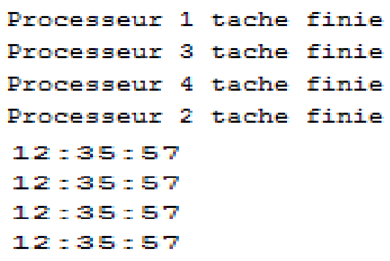
We see in the figure above that, when the fuzzy interrogations are launched on several processors (four in this case) the execution time decreases. In this case, the degree of satisfaction is reached because the execution only took 0.5 sec and the occupation of each processor was 1.75%.

4.2. Tasks executed by different processors in parallel



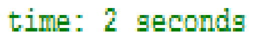
```
Processeur 3 tache finie
Processeur 2 tache finie
Processeur 4 tache finie
Processeur 1 tache finie
12:35:56
12:35:56
12:35:56
12:35:56
```

Following:



```
Processeur 1 tache finie
Processeur 3 tache finie
Processeur 4 tache finie
Processeur 2 tache finie
12:35:57
12:35:57
12:35:57
12:35:57
```

Generally :



```
time: 2 seconds
```

The parallel efficiency on the number of processors that perform the tasks after applying the evaluator of the imprecise queries launched is:

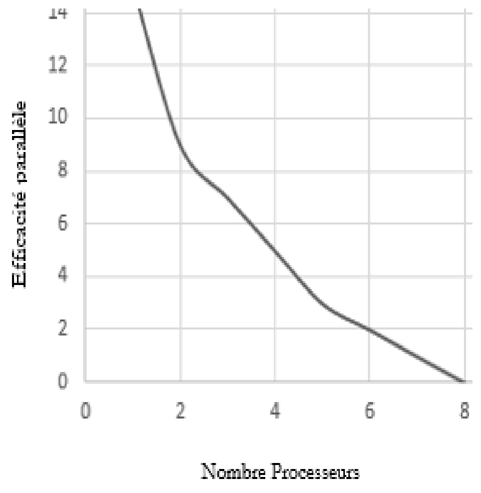


Fig 3: Parallel efficiency on the number

We see here that the more number of processors increases, the faster execution speed becomes and the result is delivered in less time.

Fuzzy sequential and parallel algorithmic evaluator model of the queries

Rs: Sequential Requests
Rp: Parallel queries
Rf: Fuzzy requests

Rpf: Fuzzy parallel queries
Is: set of sequential interrogations
If: set of fuzzy questions
Df = {d1f, d2f,…, dnf} Df: set of fuzzy data, dif: fuzzy data, i = 1 to n

```
time = 0
Repeat for i = 0 to n - 1 do
            For j = i to m - 1 do
                Selection Df [i] [j]
                    j = j + 1
                    time calculation and evaluation
                        time = time + 1
                end do
                    i = i + 1
        end for
end Repeat
        if (time = 60) then
  Writing the execution took an hour
      Otherwise if (time> 60) then
                          Write the execution over an hour: time
Otherwise Write time
```

If = {I1f, I2f, I3f, I4f, I5f,…, Inf}
I1f = {Sr11f, Sr12f, Sr13f, Sr14f, Sr15f,…, Sriinf}
I2f = {Sr21f, Sr22f, Sr23f, Sr24f, Sr25f,…, Sriinf}
I3f = {Sr31f, Sr32f, Sr33f, Sr34f, Sr35f,…, Sriinf}
...
Iif = {Sri1f, Sri2f, Sri3f, Sri4f, Sri5f,…, Srimnf}

```
                   time = 0
          For i = 0 up to n - 1 execute in parallel
                        Pi = Sriif
                  // Evaluation of the processing time
                    time = time + 1
              end for
                        $Iif = \sum_{i=0}^{n-1} Srif$
                            If (time = 60) then
                              Writing the execution took an hour
                      Otherwise If (time> 60) then
                          Write the execution over an hour: time
If not
                            Write time
End.
```

It is noted that the fuzzy interrogations launched in parallel ends quickly than those launched in sequential. As in the figure below:

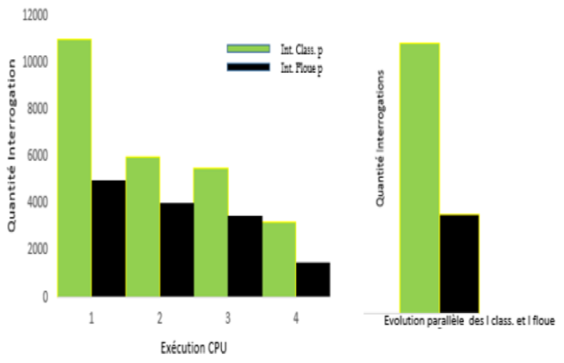


Fig 4: Fuzzy interrogations launched in parallel

5. Conclusion

This article is placed in the context of the evaluation of fuzzy queries to which parallelism has been applied. In itself, a fuzzy interrogation is more flexible than a classic interrogation but when performing a fuzzy interrogation on a computer with only one CPU, we find that this flexibility is covered just by the presentation of a good result because it goes through all borders. On the other hand, when a complex fuzzy query is parallelized, it is the execution speed of the sub queries that increases and the result is delivered in a few milliseconds. This is visible thanks to the Fuzzy sequential and parallel algorithmic evaluator model of the queries which analyzes the progress.

References

[1]. Dubois, D., Prade, H. : Using fuzzy sets in exible querying : Why and how? In : Proc. of the 1996 Workshop on Flexible Query-Answering Systems. (1996) pp. 89{103
[2]. Bosc, P., Pivert, O. : SQLf : a relational database language for fuzzy querying. IEEE Transactions on Fuzzy Systems 3 (1995) 1{17
[3]. Gregory Smits et al, PostgreSQLf : un Système d'Interrogation Floue
[4] A. Hadjali, S. Kaci, and H. Prade. Database preference queries - a possibilistic logic approach with symbolic priorities. Ann. Math. Artif. Intell., 63(3-4) :357–383, 2011.
[5] O. Pivert and P. Bosc. Fuzzy Preference Queries to Relational Databases. Imperial College Press, London, UK, 2012.
[6] O. Pivert and G. Smits. On fuzzy preference queries explicitly handling satisfaction levels. In S. Greco, B. Bouchon-Meunier, G. Coletti, M. Fedrizzi, B. Matarazzo, and R. R. Yager, editors, IPMU (1), volume 297 of Communications in Computer and Information Science, pages 341–350. Springer, 2012.
[7] Luc BOUGANIM, Exécution parallèle de requêtes relationnelles et équilibrage de charge, INRIA Rocquencourt, 2013
[8] Daniel ROCACHER et al, Relations d'ordre floues sur des quantités floues et expression des requêtes flexibles, Irisa, 2014

[9] Samuel BARTEL, Interrogation floue de bases de données : extension de iSQLf, enssat, 2006
[10] Thomas GIRAULT et al, requêtes floues et SGBD relationnels, vers un couplage renforcé, irisa, 2013
[11] Olivier PIVERT, Un Système d'interrogation floue, Irisa, Université de Rennes 1, 2013
[12] Le THANH, Système de Gestion des Bases de données parallèles et distribués : architecture et algorithmique, ESSI3, Université de nice, 1994
[13] Jean Pepe M. Buanga, Simon Ntumba Badibanga, Richard Kabamba Ilunga , Enhanced Parallel Skyline on Multi-core Architecture with Low Memory Space Cost, IJCSI International Journal of Computer Science Issues, Volume 13, Issue 5, September 2016

First Author is Assistant of computer sciences and master student at the University of Kinshasa, DRC. Research area : Data analysis. Author of many publications, such as: Data mart approach for stock management model with a calendar under budgetary constraint, IJCSI, volume 15, Issue 5, September 2018,

Second Author is professor and head of Mathematic and informatics department of the University of Kinshasa. As publications, Author of many publications, such as: "Enhanced Parallel Skyline on multi-core architecture with lax Memory space Cost", IJCSI, volume 13, Issue 5, September 2016, Data mart approach for stock management model with a calendar under budgetary constraint, IJCSI, volume 15, Issue 5, September 2018, Poster et the 2nd International conference on Big Data Analysis and Data Mining, San Antonio, USA, 30 november- 01 December 2015 "; Data Mart Approach for Stock Management Model with a calendar Uner Budgetary contraint, IJCSI, volume 15, Issue 5, September 2016,

Third author is professor at the Mathematic and informatics department of the University of Kinshasa. Director of informatics laboratory of the faculty of sciences at the university of Kinshasa. He is author of many articles in many scientific journals like in IJCSI . Poster et the 2nd International conference on Big Data Analysis and Data Mining, San Antonio, USA, 30 november- 01 December 2015.