Satisfaction-based Query Load Balancing

Jorge-Arnulfo Quiane-Ruiz, Philippe Lamarre, Patrick Valduriez

Satisfaction-Based Query Load Balancing

Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, and Patrick Valduriez

INRIA and LINA
Université de Nantes
2 rue de la houssinière, 44322 Nantes Cedex 3, France
{Jorge.Quiane,Philippe.Lamarre}@univ-nantes.fr,Patrick.Valduriez@inria.fr

Abstract. We consider the query allocation problem in open and large distributed information systems. Provider sources are heterogeneous, autonomous, and have finite capacity to perform queries. A main objective in query allocation is to obtain good response time. Most of the work towards this objective has dealt with finding the most efficient providers. But little attention has been paid to satisfy the providers interest in performing certain queries. In this paper, we address both sides of the problem. We propose a query allocation approach which allows providers to express their intention to perform queries based on their preference and satisfaction. We compare our approach to both query load balancing and economic approaches. The experimentation results show that our approach yields high efficiency while supporting the providers' preferences in adequacy with the query load. Also, we show that our approach guarantees interesting queries to providers even under low arrival query rates. In the context of open distributed systems, our approach outperforms traditional query load balancing approaches as it encourages providers to stay in the system, thus preserving the full system capacity.

1 Introduction

We consider dynamic distributed systems, providing access to large number heterogeneous, autonomous information sources. We assume that information sources play basically two roles: consumers that generate requests[1], and providers which perform requests and generate informational answers.

Providers can be heterogeneous in terms of capacity, competence and data. Heterogeneous capacity means that some providers are more powerful than others and can treat more requests per unit time. Heterogeneous competence means that some providers may treat some query types that others cannot, and vice versa. Data heterogeneity means that requests are performed differently by different providers, i.e. the same request performed by different providers may return different results. We also consider that requests are heterogeneous, i.e., some requests consume more providers' resources than others.

Providers, on the other hand, are autonomous over their resources and data management. Thus, they can express their preferences to perform queries through

[1] We will indifferently use the terms request and query throughout this paper.

an *intention* value. Such preferences may represent, for example, their strategies, their topics of interest, the response time, or the combination of two or all of them. The providers' *intentions* might be the result of merging the preferences with others factors, such as the query load or reputation/quality of the consumers.

Providers' preferences are rather static (i.e. long term) and do not change much while their *intentions* are more dynamic (i.e. short-term). For example, a given provider may prefer some request, but at some time, for some reason (e.g. over- utilized) does not intend to perform it. In other words, preferences only depend on requests while intentions take into account the context and load.

One of the problems that has been thoroughly investigated in the area of query allocation is *query load balancing* (*QLB*). The main objective of *QLB* is to maximize overall system performance (throughput and response times) by balancing the query load among providers. However, even if performance is very good, providers may be not satisfied with the system and may leave it. Thus, the system should fulfill providers' expectations in order to preserve full system capacity.

This problem is quite important in open information systems where providers can leave the system at will. When a provider is no longer satisfied with the system, the only way to express *unsatisfaction* it is to leave. In order to achieve stability, our goal is to maximize performance while ensuring over time that providers are satisfied enough to stay in the system.

In this paper, to address this problem, we propose a *QLB* approach which allows providers to express their intention and takes care of their *satisfaction*. We also develop a model that allows providers to know whether the system is fulfilling their expectations. We provide an experimental validation which compares our approach with both query load balancing and economic approaches. The experimental results show that our approach yields high efficiency while supporting the providers' preferences in adequacy with the query load.

The rest of the paper is organized as follows. Section 2 gives a motivating scenario. Section 3 defines precisely the problem we address. Section 4 presents our *QLB* approach. Section 5 defines the metrics used for validating our approach. Section 6 presents our experimental validation. Section 7 discusses related work. Finally, Section 8 concludes.

2 Motivating Scenario

We illustrate the problem we consider in this paper by means of a pharmaceutical application example. Consider a large distributed information system gathering thousands of pharmaceutical companies with the goal of selling their products.

By promotions or simply strategies, providers[2] may have more interest in selling some specific products than others. Thus, each provider stores locally its preferences for performing requests and may change them at will. In fact,

[2] Which represent pharmaceutical companies

Table 1. Providers set that are able to deal with the request.

Providers	Utilization	Preference
p_1	0.15	No
p_2	0.43	No
p_3	0.78	Yes
p_4	0.85	No
p_5	1.1	Yes

any system with competitive companies meets this schema. Now, consider a simple scenario where a given consumer requests the system for a given product, specifying some specific parameters, such as the product's name and product's presentation. Suppose that in order to obtain a good tradeoff between product's price and quality, the consumer asks for two results.

First, the system needs to identify the providers that are capable to deal with the request. This can be done using a matchmaking technique (for example [18]). Second, the system must obtain their availability and preferences for dealing with such a request. This can be done following the architecture in [8] for example.

Assume, then, that the resulting list contains 5 providers with their utilization and preferences (see Table 1). Assume that the respective pharmaceutical companies (column 1) of this list in ascending utilization order (column 2) are: p_1, p_2, p_3, p_4, and p_5. Assume that p_1, p_2, and p_4 are not interested on serving the request (column 3) for their own reasons.

In this case, the system needs to allocate the request to the two most capable providers, such that the providers' preference is respected. Current *QLB* approaches[3] would fail in such a scenario since neither p_1 nor p_2 want to deal with the request. The only two options that satisfy the providers' preference are p_3 and p_5, but allocating the query to them may hurt response time.

This example illustrates the conflict between providers' preference and utilization in query allocation. However, considering allocation alone is not very meaningful. What is more important is that a given provider may be globally satisfied with the allocation process, even though it is sometimes overloaded or does not get queries it wants. This can be checked by making regular assessment over some k last queries. The entire treatment of this scenario encompasses different aspects. First, query planning processes may be required. This problem is addressed in different ways in the literature [13]. We do not consider it in this paper, and we can indifferently assume that it is done by the consumer or any other site. Second, the system must support matchmaking techniques in order to find the relevant providers for performing requests. Such matchmaking techniques have been proposed by several groups [9, 12, 18]. So, we simply assume there exists one in our system. In addition, we do not consider either the way in which providers obtain their preferences values since it is out of the scope of this paper and orthogonal to the query allocation problem.

[3] Whose aim is to allocate queries to the less utilized providers.

3 Satisfactory Query Load Balancing Problem

In this section, we make precise the problem we consider. The system consists of a set of consumers $\boldsymbol{C}$ and of a set of providers $\boldsymbol{P}$ which can join and leave the system at will. It is possible to have $\boldsymbol{C} \cap \boldsymbol{P} \neq \emptyset$. Consumers issue queries in a *tuple* format $q = < c, t, d, n >$ such that, $q.c \in \boldsymbol{C}$ is the identifier of the consumer that has issued the query, $q.t \in \boldsymbol{T}$ is the query type and $\boldsymbol{T}$ the set of query types that the system can support, $q.d$ is the description of the task to be realized, and $q.n \in \mathbb{N}^*$ is the number of providers to which the consumer wishes to allocate its query.

Definition 1. ***Feasible Query***
Let $T_p \subseteq \boldsymbol{T}$ be the query types that the provider $p \in \boldsymbol{P}$ can treat. Given a query q with $q.t \in \boldsymbol{T}$, issued by the consumer $q.c \in \boldsymbol{C}$, let P_q denote the set of providers which can deal with q, where $P_q = \{p : (p \in \boldsymbol{P} \backslash \{q.c\}) \wedge (q.t \in T_p)\}$. Then, a query q is said to be feasible if and only if $P_q \neq \emptyset$.

We only consider the arrival feasible queries which are defined in Definition 1. Each feasible query q has a ***cost***, $cost_p(q) > 0$, that represents the treatment units that q consumes at p. Query allocation of some feasible query q among the providers which are able to deal with q is defined as a vector (see Definition 2).

Definition 2. ***Query Allocation***
Allocation of query q amongst the providers in P_q is a vector[4] $All\overrightarrow{oc}$ of length $||P_q||$ such that: $\forall p \in P_q$,

$$All\overrightarrow{oc}[p] = \begin{cases} 1 \text{ if provider } p \text{ gets the query} \\ 0 \text{ otherwise} \end{cases}$$

with $\sum_{p \in P_q} All\overrightarrow{oc}[p] \leq q.n$

In the following, the set of provider such that $All\overrightarrow{oc}[p] = 1$ will also be noted $\widehat{P_q}$. Each provider $p \in \boldsymbol{P}$ has a finite *capacity*[5], $cap_p > 0$, for performing feasible queries. We then define the provider's utilization as in Definition 3.

Definition 3. ***Provider Utilization***
$U_t(p)$ denotes the utilization of a given provider $p \in \boldsymbol{P}$ at time t, which is the capacity portion utilized by the queries that p is treating

$$U_t(p) = \frac{\sum_{q \in Q_p} cost_p(q)}{cap_p}$$

where Q_p denotes the set of queries that have been allocated to p but have not already been treated at time t (i.e. the pending queries).

[4] $All\overrightarrow{oc}_q$ when there is an ambiguity on q.
[5] Capacity means the number of treatment units that a provider can have per time unit.

Providers are free to express their intention for performing each arrival feasible query q, denoted by $I_p(q)$, where $q.t \in T_p$, $I_p(q) \in]-\infty, 1]$. Expressing intention may be the result of merging, for example, their preferences, strategies, and utilization. If the intention value is positive, the greater it is, the greater the desire for performing queries. If the intention value is negative, the smaller it is, the greater the refusal for performing queries. Providers' refusal can go down to $-\infty$ because their utilization can grow up, theoretically, to $+\infty$.

The overall aim of the system is to allocate each feasible query q of a set of arrival feasible queries $\boldsymbol{Q}$ to providers in P_q by taking into account the providers' utilization, preferences, and *satisfaction*.

4 Satisfaction-Based Query Allocation

In this section, we present our query allocation approach based on satisfaction, called *Satisfaction-based Query Load Balancing* S_bQLB. We first present the providers characteristics on which S_bQLB is based and then present the S_bQLB approach.

4.1 Providers Characterization

In this section, we present what the providers can perceive from the system. We focus on three principal characteristics: the satisfaction with the system (*Satisfaction*), their adequation to the system (*Adequation*), and their satisfaction with the query allocation method (*Allocation Satisfaction*).

Let us first introduce some notations to describe how a provider can perceive the system. We assume that each provider maintains a vector $\overrightarrow{PP}_p^k$ of its k last preferences for performing the queries that have been proposed to it by the system[6]. This set of proposed queries is noted PQ_p^k. By convention, $\forall q \in PQ_p^k$, $\overrightarrow{PP}_p^k[q] \in [-1..1]$. Finally, SQ_p^k denotes the set of queries that have been served by p among PQ_p^k ($SQ_p^k \subset PQ_p^k$).

Satisfaction This notion denotes the satisfaction degree that the providers have from their obtained queries. That is, if they serve queries that they want in general. Providers obtain their *satisfaction* only from queries that they treat, which allows them to know if they are accomplishing their objectives in the system. We define the provider *satisfaction* as the average of the preferences that a provider $p \in P$ had to their received queries among the k last incoming feasible queries. In other words,

$$\delta_s(p) = \begin{cases} \frac{\sum_{q \in SQ_p^k} \overrightarrow{PP}_p^k[q]}{2||SQ_p^k||} + \frac{1}{2} & if\ ||SQ_p^k|| > 0 \\ 0 & otherwise \end{cases} \quad (1)$$

[6] Note that k may be different for each provider depending on it's storage capacity, or strategy. For the sake of simplicity, we have assumed here that they all use the same k.

Adequation The *Adequation* characteristic denotes the adequation degree of a provider to the system. It is based on the preferences that the provider had towards all the feasible queries proposed by the system. In other words, a provider is adequate to the system whether it receives interesting queries from the system. This notion can only be used by providers in those information systems where they can see feasible queries to pass even if they do not finally serve them. This is the case of our utilized system architecture [8]. We define the *adequation* of a given provider p as the average of its k last preferences for performing queries,

$$\delta_a(p) = \frac{\sum_{q \in PQ_p^k} \overrightarrow{PP}_p^k[q]}{2k} + \frac{1}{2} \tag{2}$$

Allocation Satisfaction The *allocation satisfaction* denotes the satisfaction degree that a provider has from the query allocation method. It allows a provider to know if the allocation method is doing a good job for it. That is, if the system tries to give them, in average, interesting feasible queries. We, thus, define the *allocation satisfaction* of a given provider p, $\delta_{as}(p)$, as the quotient of its *satisfaction* divided by its *adequation*,

$$\delta_{as}(p) = \frac{\delta_s(p)}{\delta_a(p)} \tag{3}$$

Thus, the greater the provider satisfaction value with respect to the provider adequation value is, the more satisfied the provider is from the mediation process.

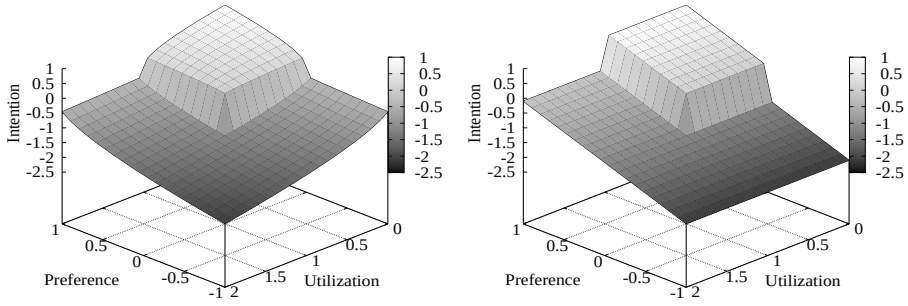
4.2 Satisfaction-Based Query Load Balancing

This section details the S_bQLB approach itself for allocating a query q to the $q.n$ providers based on the providers' satisfaction. We focus on the case where requests can be viewed as single units of work called *tasks*. Requests arrive to the system to be allocated to n providers. We assume that a previous matchmaking step has found the set of providers P_q to deal with an incoming query.

Algorithm 1 shows the main steps of the S_bQLB approach for selecting the providers that will treat the queries. These steps are detailed below.

Given an incoming feasible query q and the providers' set P_q, as a first step, the S_bQLB asks to each $p \in P_q$ for its intention[7] to deal with q and build a vector $\overrightarrow{I}^q$ containing such intentions values. The way in which providers work out their intentions is defined in Definition 4 and such details are considered private information. This way allows providers to base their intentions principally on their preferences when are not satisfied and on their utilization conversely.

[7] Notice that, considering our system architecture, this operation is realized in local which considerably reduce the network traffic



(a) for a provider's *satisfaction* of 0.5 (b) for a provider's *satisfaction* of 0

Fig. 1. Tradeoff between preference and utilization for getting intention.

Definition 4. *Providers' Intention*
Given a query q the intention of a provider $p \in P_q$ to deal with it is defined as follows,

$$I_p(q) = \begin{cases} (pref^{1-\delta_s(p)})(1-U_t(p))^{\delta_s(p)} & if(pref >= 0) \wedge (U_t(p) <= 1) \\ -(((1-pref)+\epsilon)^{1-\delta_s(p)}(U_t(p)+\epsilon)^{\delta_s(p)}) & otherwise \end{cases}$$

with $\epsilon = 0.1$ and $pref$ denotes the p's preference for dealing with q.

As second step, the S_bQLB approach computes the providers' ranking $\overrightarrow{R}^q$ based on their shown intentions. Such a ranking is introduced for an easiest use of the $\overrightarrow{I}^q$ vector and to enable the selection of the $q.n$ providers to deal with q. Intuitively, $\overrightarrow{R}^q[1]$ is the most interested provider to deal with q, $\overrightarrow{R}^q[2]$ the second, and so on up to $\overrightarrow{R}^q[N]$ which is the least.

As a result, the S_bQLB returns the $q.n$ best ranked providers in the $\overrightarrow{I}^q$ vector if $q.n <= ||P_q||$, else it returns the $||P_q||$ providers. Note that the providers' intention is the criterion by which S_bQLB chooses the providers.

As illustrated by Figures 1, providers will show positive intention for dealing with queries only when their preferences and utilization are between 0 and 1. This means, that S_bQLB strives to allocate queries to those providers that desire to perform them and that are not overutilized. This enables providers to preserve their preferences while good response times are also ensured to consumers.

On the one hand, we observe that when providers are satisfied of 0.5 (see Figure 1(a)) the providers' preferences and utilization have the same importance for providers in the obtention of their intentions. On the other hand, when providers are not satisfied at all (see Figure 1(b)) the providers' utilization has not importance for providers and are their preferences that define their intentions. Conversely, when providers are completely satisfied the utilization defines their intentions.

Algorithm 1 *providersSelection(q, k, P_q)*

1: **begin**
2: **foreach** p **in** P_q **do**
3: ask for the provider's intention $I_p(q)$
4: put the $I_p(q)$ value into $\overrightarrow{I}^q$
5: **done**
6: compute the providers' intention vector ranking $\overrightarrow{R}^q$
7: *providersSelection* $\leftarrow \overrightarrow{I}^q[\overrightarrow{R}^q[1..min(n, N)]]$
8: **end**

5 Metrics

We now propose a solution to analyze the *QLB* approaches against the *Satisfactory QLB* problem. To do so, we use two metrics, one for evaluating the ensured *satisfaction-balance* in the system and the other one for evaluating the ensured *query-balance*.

5.1 Satisfaction Balance Metrics

We define the *dissatisfaction* ratio as the ratio between the most and the least satisfied providers in the system. Since we look for providers' satisfaction, we will use the *satisfaction* term instead the *dissatisfaction* one. Obviously, the *satisfaction* ratio is defined as the *dissatisfaction* ratio's inverse.

Before going further, let us say that we have had to define how providers are satisfied by the system, $\delta_s(p)$ (see Equation 1). The following can also be developed for the *adequation* and *allocation satisfaction* functions (Equations 2 and 3), but for space reasons we just develop it for the *satisfaction* function.

A distributed information system ensures a provider *satisfaction-balance* ratio, α, if after each query allocation,

$$\min_{p \in \boldsymbol{P}} \big(\delta_s(p)\big) + c_s \geq \alpha \left(\max_{p \in \boldsymbol{P}} \big(\delta_s(p)\big) + c_s \right)$$

for some fixed constant[8] c_s and where α denotes the desired *satisfaction-balance* ratio in the system. Having said this, we measure the *satisfaction-balance* at a given time t as follows,

$$\gamma = \frac{\min\limits_{p \in P_x} \big(\delta_s(p)\big) + c_s}{\max\limits_{p \in P_x} \big(\delta_s(p)\big) + c_s}$$

γ denotes the factor under which the system is said to be *satisfaction-balanced* at a given time t.

[8] Which is set to the minimal satisfaction value that a provider could have with a query, 0.001 for example.

5.2 Query Balance Metrics

As is conventional, we define the imbalance ratio at a given time t as the ratio between the most and the least utilized provider in the system at that time. For better representation and explanation of this property, we use the term *balance* instead of *imbalance*, which is defined as the imbalance ratio's inverse,

$$\min_{p \in \boldsymbol{P}} \big(U_t(p)\big) + c_u \geq \sigma \left(\max_{p \in \boldsymbol{P}} \big(U_t(p)\big) + c_u \right)$$

for some fixed constant[9] c_u and where σ denotes the desired *query-balance* ratio in the system. In order to measure the *query-balance* λ in the system, we first must measure the providers' utilization in the system, as in Definition 3, and then, we proceed to measure λ as follows,

$$\lambda = \frac{\min\limits_{p \in P_x} \big(U_t(p)\big) + c_u}{\max\limits_{p \in P_x} \big(U_t(p)\big) + c_u}$$

We believe that it is quite important that the system also be able to balance all the incoming feasible queries among the relevant providers in average. That is, it must strive to give queries to all the providers in the system if possible, in order to avoid having providers leave the system becuase of *starvation* and *unsatisfaction*. This is why we introduce the *average query-balance*, λ', metric. We do not need to define this metric since it is symmetrical to the *query-balance* metric. We only define what an *average provider utilization* means in a discrete time interval.

Definition 5. *Provider Average Utilization*
$U_{[t_1,\, t_2]}(p)$ *is defined to be the average of the p's utilization (with* $p \in \boldsymbol{P}$*) at the time interval* $[t_1,\, t_2]$,

$$U_{[t_1,\, t_2]}(p) = \frac{\sum\limits_{t_i \in [t_1 .. t_2]} U_{t_i}(p)}{(t_2 - t_1) + 1}$$

6 Experimental Validation

In this section, we present our experimental evaluation and discuss the results. We have simulated a *mono-mediator* distributed information system with heterogeneous and autonomous provider sources and carried out a series of tests with the objective of assessing *how well the* S_bQLB *approach operates in autonomous environments*. We first describe the *Capacity Based* and *Economic* approaches against which we compare the S_bQLB algorithm. Next, we describe the simulation setup.

We have conducted three types of evaluations. In the first series of evaluations, we analyze and compare the performance of S_bQLB against *Capacity*

[9] Which is set to the minimal utilization value that a provider may have with a query, 0.001 for example.

Based and *Economic* approaches in considering the *unsatisfaction* departures of providers. The second series focuses on providers' utilization. More precisely, we measure the QLB achieved by current QLB approaches. We also study, in these experiments, if such approaches really strive to give requests to all providers in the system, i.e. we measure how well they realize the QLB in average. Finally, the third series of tests focuses on the providers' satisfactions. We study, here, how well S_bQLB and current QLB approaches satisfy the providers, and how well they do it with different query rates.

6.1 Baseline *QLB* Approaches

Capacity Based In distributed information systems, there are two well known approaches to balance requests across providers: the *load based* and *capacity based*. We discard the *load based* [1, 6] approach since, unlike *capacity based*, they inherently assume that providers and requests are homogeneous.

In the *capacity based* [11, 16, 19] approach, one common way to allocate queries is choosing the providers that have more available capacities amongst the providers' set that can deal with them. In other words, it sends queries to those relevant providers that are the less utilized. This criterion is defined below.

$$criterion : 1/(1 + U_t(p))$$

Another way is to discard those providers that are overutilized or fail a constant utilization threshold, and then balance queries among the remaining ones. However, in practice is not easy to set the utilization threshold. By this fact, we analyze in our experiments the first way for allocating queries.

Economic Economical models have been introduced in distributed systems with the goal of decreasing the data management complexity by decentralizing the resources' access and the allocation mechanisms [4, 5, 2, 17]. We implemented an not pure economical[10] QLB approach based on the *Sealed Bid Auction*, where providers pay for acquiring requests. The criterion to select providers is in considering the providers' available capacity and not just the providers' bid. That is, the highest below criterion for a provider is given access to the request.

$$criterion : bid \times (1/(1 + U_t(p)))$$

In economical approaches the *bid* is set using a bulletin board containing the preference for bidding which, conversely to our preference notion (see Section 4.1), inherently limits the provider's *preference* to the queries' type.

Moreover, let us say that for our experiment simulations, we use virtual money which is just seen as a means of regulation. In the course of the time the money is spent by providers in order to acquire requests. The process itself does not provide them anyway to earn money. Nevertheless, a source of financing is necessary to them, because otherwise, after some time, providers would not have

[10] Since not only the bids are considered to select providers

Table 2. Simulation parameters.

Parameter	Definition	Value
nbConsumers	Number of consumers	200
nbProviders	Number of providers	400
nbMediators	Number of mediators	1
qDistribution	Query arrival distribution	Poisson
iniSatisfaction	Initial providers' satisfaction	0
qTypes	Supported query types	10
nbSimulations	Number of realized simulations for each experience test	10

more credit to bid positively. Different solutions are possible. We have chosen to associate a bank with the mediator (if there were several mediators, there would be as many banks as mediators). The mediator's bank gives an specific amount of money to providers at the registration step and in the course of the time it equally redistributes the money which it gained to the providers after some given time.

6.2 Setup

In all experiments, the number of consumers and providers is 200 and 400 respectively, with only one mediator allocating all the incoming feasible queries[11]. Feasible queries arrive to the system in a Poisson distribution, which has been found in dynamic autonomous environments [10]. Consumers always ask for 1 provider to solve their requests (i.e. $q.n = 1$). Providers are initialized with a satisfaction value of 0, and a satisfaction size[12] of 500. Since our principal focus in this paper is to study the way in which requests are allocated in the system, we do not take into account the bandwidth problem and consider it as a future work.

We set the providers' capacity heterogeneity, in our experiments, in accordance to the results in [15]. This work measures the nodes capacities in the Napster and Gnutella systems which are a clear example of large distributed systems. Based on these results, we generate around 10% of low-capable, 60% of medium-capable, and 30% of high-capable providers. The high-capable providers are 3 times more capable than medium-capable providers and still 7 times more capable than low-capable ones. We generate 10 types of requests which can be of two classes that consume, respectively, 130 and 150 treatment units at the high-capable providers[13].

For the experiments, in order to simulate high autonomy, providers' preferences are randomly obtained between -1 and 1, where 1 means the greatest

[11] We assigned sufficient resources to the mediator so that it does not cause bottlenecks in the system.

[12] Which denotes the k last arrival feasible queries.

[13] Such a treatment takes almost 1.3 and 1.5 seconds, respectively.

interest and −1 the greatest refusal. More sophisticated mechanisms for obtaining such preferences can be applied ([14] for example).

6.3 Performance Results

We now investigate the impact on performance of the *Capacity Based*, *Economic*, and S_bQLB approaches against the providers' departures by *unsatisfaction*. To this end, we have to set the *unsatisfaction* threshold in which providers decide to leave the system whether they fail it. In our experiments, providers decide to leave the system if they receive more than the 50% of uninteresting queries.

We evaluate response times[14] with different request arrival rates in order to study the possible impact that the *unsatisfaction* departures could have. Also, we measured the number of *uninteresting* queries allocated by all these three approaches. Results are shown in Figure 2. We observe that S_bQLB and the *Economic* approaches allocate only interesting queries to providers, while the *Capacity Based* approach allocates in average a 50% of uninteresting queries!

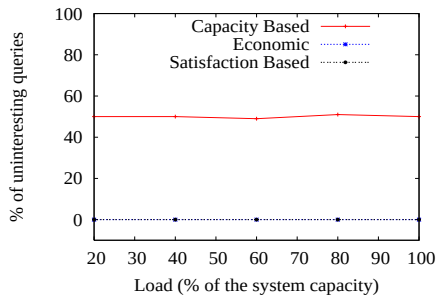


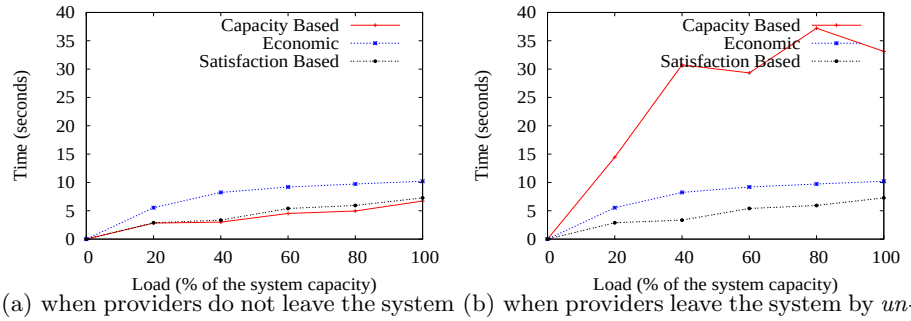
Fig. 2. Uninteresting received queries.

Now, in order to see the impact of provider departures, we show in Figure 3(a) the ensured response times when providers are not allowed to leave the system by *unsatisfaction* (i.e. with the full system capacity).

In Figure 3(b) the same parameters are drawn when providers are authorized to quite the system by unsatisfaction. As expected, the *Capacity Based* approach suffers significantly from *unsatisfaction* departures. We can observe that, in such a case, the *Capacity Based* degrades in average the response times by a factor of 4.5! This is because even if it gives requests to all providers, it usually gives them uninteresting ones. On the other hand, since the S_bQLB and *Economic* approaches strive to give interesting requests to providers, they deals better with this problem in preserving their full system capacity.

The choice of an *unsatisfaction* threshold under which the providers would leave the system is very subjective and may depend on several external factors.

[14] As is conventional, the response time is defined as the elapsed time from the time that a query q is issued to the time that $q.c$ receives the response.



(a) when providers do not leave the system by *unsatisfaction*

(b) when providers leave the system by *unsatisfaction*

Fig. 3. Response times.

As shown previously, it has a deep impact on response time, but also on adequation and satisfaction of remaining providers. By this fact, to avoid any suspicion on the choice of the *unsatisfaction* threshold, in the following experiments, we have preferred to consider captive providers: they are not allowed to leave the system whatever their degree of dissatisfaction is.

6.4 Providers' Utilization Results

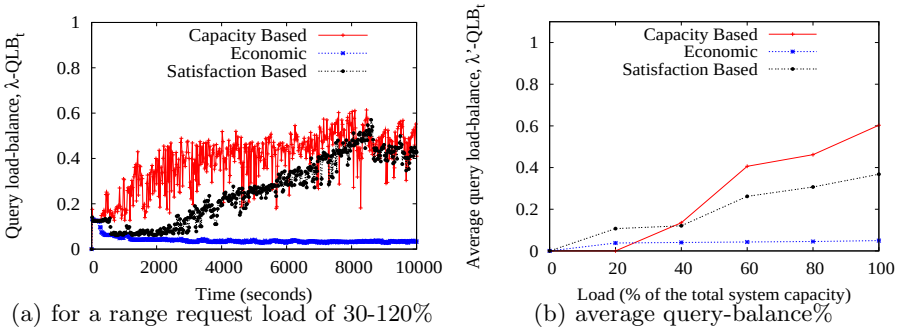
We measure the *query-balance* factor (λ-QLB_t) for different Poisson arrival rates[15]. λ-QLB_t (Y-axis) at different times (X-axis) of the experiment. Contrary to the expected, the results show that the *Economic* approach has serious problems to ensure good λ-QLB_t in the system, because for providers, preference is more important than utilization in order to express their intentions.

The results show that the *Capacity Based* and S_bQLB approaches have serious problems to ensure good λ-QLB_t with request arrival rates under 40% of the total system capacity. In contrast, when the query arrival rate increases, both approaches improve the λ-QLB_t in the system. This is due to the fact that when most providers have no queries (i.e. have all their capacity available), queries may be allocated to those providers that spend more treatment units to perform them. Hence, each time a query is allocated to the less capable providers, the distance between the more and less utilized providers increases significantly.

To prove this intuition, we studied the ensured λ-QLB_t by varying the query arrival rate from 30 (at the beginning of the simulation) to 120% (at the end of the simulation) of the total system capacity. The result is shown in Figure 4(a). We observe that both *Capacity Based* and S_bQLB approaches improve the λ-QLB_t as the query arrival rate increases.

In all cases, the *Capacity Based* is better than S_bQLB. This is because it takes into account providers' intentions. Howver, *Capacity Based* suffers from

[15] For example, given our simulation setup, the 40% and 80% of the total system capacity correspond respectively to 84 and 164 requests per second.



(a) for a range request load of 30-120% (b) average query-balance%

Fig. 4. Query-balance.

greater vairations which means that, at times, some providers are much more utilized than others (when all of them have the same chances to get queries). S_bQLB better deals with such variations by considering providers' satisfaction. Furthermore, S_bQLB is much better than the *Economic* approach.

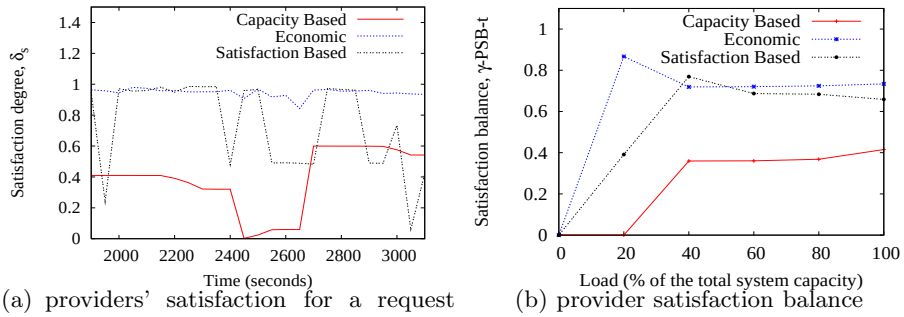
We also measured the *average query-balance* factor (λ-$QLB_{[t_1, t_2]}$) ensured by these approaches. That is, we analyzed how well these approaches avoid the *starvation* problems. To this end, we ran several tests with different request arrival rates and measure the λ-$QLB_{[t_1, t_2]}$ in such a time interval (from the beginning to the end of the simulations). The results of such experiments are shown by Figure 4(b).

Unlike the λ-QLB_t experiments, we observe that for request arrival rates under 40% of the total system capacity the S_bQLB guarantees a better λ'-$QLB_{[t_1, t_2]}$ than the *Capacity Based* and *Economic* approaches. This means that S_bQLB strives to give queries, in the course of the time, to all the providers in the system even if the arrival query rate is not sufficient to do it. In contrast, when the query arrival rate is greater than the 40% of the total system capacity, *Capacity Based* does it better. This is because the arrival query rate is sufficient for giving queries to all providers in the system. But, we observe that S_bQLB is still better than the *Economic* approach in which providers significantly suffers from *starvation* problems.

6.5 Providers' Satisfaction Results

As expected, the *adequation* of providers depends only on query arrival and their preferences for performing queries. Adequation is completely independent of the query arrival rates. In our experiments, the providers' *adequation* is 0.5 in average for all the query arrival rates. So, graphics are not presented here since they do not say anything else.

In our experiments the *allocation satisfaction* and *satisfaction* of providers are very similar but at different scale. This is because the results that providers get, depend roughly on the query allocation process. Then, because of space, we



(a) providers' satisfaction for a request load of 80%

(b) provider satisfaction balance

Fig. 5. Provider satisfactions.

just present the providers' *satisfaction* results for an arrival query rate of 80% of the total system capacity (see Figure 5(a)).

The results show that conversely to the QLB aspect (Section 6.4) the *Economic* approach better satisfies the providers than the S_bQLB and *Capacity Based* approaches. This is because, the providers' intentions in the *Economic* approach are principally based on the providers' preferences while S_bQLB takes equally into account the providers' utilization as shown in Section 4.2.

We can also observe, in our experiments, that the *satisfaction* of providers with S_bQLB are almost all the time over the providers' *adequation* (i.e. over 0.5). In contrast, the *satisfaction* of providers with *Capacity Based* are almost all the time under the providers' *adequation*. The oscillation of the providers' *satisfaction* is caused by the natural competition of providers for performing queries.

Furthermore, we observe that while the ***less satisfied*** provider with the *Economic* and S_bQLB approaches has, respectively, a *satisfaction* of 0.69 and 0.61 with an arrival query rate of 80% the total system capacity, the ***more satisfied*** provider with *Capacity Based* has a satisfaction of 0.61! This confirms that *Capacity Based* significantly hurts the providers' preferences for performing queries. The impact of this phenomenon was shown in Section 6.3.

Finally, the experiments show that the *Economic* and S_bQLB guarantee better γ-PSB$_t$ factors than the *Capacity Based* (see Figure 5(b)). In other words, both *Economic* and S_bQLB strive to satisfy equally all the providers while *Capacity Based* does not.

7 Related Work

The problem of balancing queries while respecting the providers' autonomy for performing queries has not received much attention and is still an open problem.

Much work on query load balancing has been done in distributed systems [1, 6, 11, 16, 19]. We can classify load balancing algorithms into two approaches: *load based* and *capacity based*.

Load based approach decide to allocate requests to those providers with the highest inverse probability of their reported load. Generally, load is defined as the number of request that the providers has in its arrival queries' queue. Thus, they inherently assume that providers and requests are homogeneous. *Capacity based* approach already take into account such heterogeneity by allocating requests to those providers with the greatest available capacity. The provider's capacity is defined as the maximum query rate that the provider can treat. All these works mainly model and address the problem of minimizing the providers' load or utilization for ensuring good response times in the system. However, unlike our approach, they do not consider the providers' intention for performing queries which drastically punish the providers' autonomy and the system performance (see Sections 6.3 and 6.5).

Many solutions [2, 5, 17] strive to deal with providers' autonomy by means of economical models. Providers denote their intention for performing queries via a bidding mechanism. A survey of economic models for various aspects of distributed system is presented in [4]. The motivation of economical models is to decentralize the access to the system's resources. In economical models, every provider in the system tries to maximize its own profit by selling services. Nevertheless, such models need robust market mechanisms for avoiding a degradation of the system's performance, and rationalized pricing schemes for giving *price* guarantees to consumers (see [4]).

Mariposa [17] is one of the first systems which deals with the query processing and data migration problem, in distributed information systems, based on a bidding process. In this approach, providers bid to acquire parts of a query and consumers pay for their queries' execution. In order to ensure a crude form of query load balancing, the providers' bid is multiplied by their load. Then, the mediator broker selects a set of bids that corresponds to a set of relevant queries and has an aggregate price and delay under a bid curve provided by the consumer. But, it is unclear how this approach really ensures the *QLB* in the system. Furthermore, some queries may not get processed although relevant providers exist, just because they do not intend to treat them.

In addition, our approach also differs from the above works since it also strives to balance queries in the course of the time reducing, by consequent, the request starvation in the system. In contrast, we assume that providers say the truth about their utilization and satisfaction, but also, *Capacity Based* and *Economic* approaches does for the providers' utilization and credit balance respectively. If this is not the case, works about providers' reputation can be applied to tackle this issue, for instance [3, 7].

8 Conclusion

In this paper, we addressed the problem of balancing queries in dynamic and distributed systems with autonomous providers. We considered not only providers' utilization but also providers' preference and satisfaction. This paper has several contributions.

First, we defined a model that enables providers to evaluate if they are meeting their objectives. The model relies on three definitions: *satisfaction* which defines what providers are really getting from the system; *adequation* which defines how much interesting are the proposed queries; and *allocation satisfaction* which allows providers to know whether the query allocation process is doing a good job for them.

Second, we proposed a *QLB* approach which takes care of the providers' *satisfaction*, called *Satisfaction-based QLB* (S_bQLB). S_bQLB allows providers to express their intention to perform queries by taking into account their preferences and utilization in accordance to their satisfaction.

Third, we evaluated and compared the S_bQLB approach against the *Capacity Based* and *Economic* approaches. Our results show that the *Economic* approach does not guarantee good query-balance factors for all query arrival rates. This is because it relies mainly on the bids made by the providers (i.e. indirectly on the providers' preferences). On the other hand, we found that S_bQLB and *Capacity Based* have problems to ensure good query-balance factors for low request arrival rates (rates from 10 to 40% of the total system capacity). This is due to the fact that query cost is not considered in query allocation. Thus, for low query arrival rates, almost all providers have the same chances to get queries.

The experimental results show that the *Economic* approach has serious problems of *starvation* for all request arrival rates. This is due to the fact that providers that are very interested in a given query may spend much money for getting it. Then, such providers tend to be idle by lack of money. *Capacity Based* approach suffers from *starvation* only for query arrival rates under 30% of the total system capacity. When the query arrival rate increases, it better tackles this problem. This is because the query arrival rate is sufficient for balancing queries while giving queries to all providers in the system. Our S_bQLB approach deals better with this problem than the *Capacity Based* approach for query arrival rates under 30% of the total system capacity. In all cases, it is better than the *Economic* approach.

Another important result is that the *Capacity Based* approach drastically hurts providers' autonomy in allocating them, in average, 50% of uninteresting queries. Thus, *Capacity Based* does not scale up since it suffers from providers' departures. Such departures degrade the system's response time by a factor of 4.5. On the other hand, both S_bQLB and *Economic* approaches allocate only interesting queries to providers, and then, do not suffer from providers' departures.

Conversely to the expected, economical models do not perform well the *QLB* task since they depend on the providers preferences for performing queries. Furthermore, while S_bQLB does not rely on special mechanisms, *Economic* approach does [5, 4]. This makes the S_bQLB approach more suitable for open and very large distributed systems, such as *peer-to-peer* systems.

As future work, we plan to design a *QLB* approach that also takes into account the consumers' intention for allocating requests. In order to be fair and guarantee equity at all levels, we must be able to decide which is the most

important criterion, among consumers' satisfaction, providers' satisfaction or providers' utilization, at a given time in the system.

References

1. Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced Allocations. *SIAM J. Comput.*, 1999.
2. R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. Economic Models for Management of Resources in Grid Computing. *CoRR*, 2001.
3. Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *In EC '04*, 2004.
4. D. F. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic models for allocating resources in computer systems. 1996.
5. D. F. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In *ICDCS*, 1988.
6. Z. Genova and K. J. Christensen. Challenges in URL Switching for Implementing Globally Distributed Web Sites. In *ICPP Workshops*, 2000.
7. Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, 2003.
8. P. Lamarre, S. Cazalens, S. Lemp, and P. Valduriez. A Flexible Mediation Process for Large Distributed Information Systems. In *CoopIS/DOA/ODBASE*, 2004.
9. L. Li and I. Horrocks. A Software Framework For Matchmaking Based on Semantic Web Technology. In *Proc. 12th International WWW Conference*.
10. E. P. Markatos. Tracing a Large-Scale Peer to Peer System: An Hour in the Life of Gnutella. In *CCGRID*, 2002.
11. R. Mirchandaney, D. F. Towsley, and J. A. Stankovic. Adaptive Load Sharing in Heterogeneous Distributed Systems. *J. Parallel Distrib. Comput.*, 1990.
12. M. H. Nodine, W. Bohrer, and A. H. Ngu. Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. In *Proc. 15th ICDE*, 1999.
13. M. T. Ösu and P. Valduriez. *Principles of distributed database systems (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.
14. A. Sah, J. Blow, and B. Dennis. An introduction to the Rush language. In *Proc. Tcl'94 Workshop*, 1994.
15. S. Saroiu, P. Krishna Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. of MCN*, 2002.
16. N. G. Shivaratri, P. Krueger, and M. Singhal. Load Distributing for Locally Distributed Systems. *IEEE Computer*, 1992.
17. M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A Wide-Area Distributed Database System. *VLDB J.*, 1996.
18. K. P. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic Service Matchmaking Among Agents in Open Information Environments. *SIGMOD Record*, 1999.
19. H. Zhu, T. Yang, Q. Zheng, D. Watson, O. H. Ibarra, and T. R. Smith. Adaptive Load Sharing for Clustered Digital Library Servers. In *HPDC*, 1998.