SbQA: A Self-Adaptable Query Allocation Process

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

S_bQA: A Self-Adaptable Query Allocation Process

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

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

***Abstract*— We present a flexible query allocation framework, called *Satisfaction-based Query Allocation* (S_bQA for short), for distributed information systems where both consumers and providers (the participants) have special interests towards queries. A particularity of S_bQA is that it allocates queries while considering both query load and participants' interests. To be fair, it dynamically trades consumers' interests for providers' interests based on their satisfaction. In this demo we illustrate the flexibility and efficiency of S_bQA to allocate queries on the *Berkeley Open Infrastructure for Network Computing* (BOINC). We also demonstrate that S_bQA is self-adaptable to the participants' expectations. Finally, we demonstrate that S_bQA can be adapted to different kinds of applications by varying its parameters.**

I. Introduction

Efficient query allocation that ensures good system performance (typically throughput and response time) is crucial in very large distributed information systems where consumers and providers (which we refer to participants for clarity) are autonomous. Autonomy in this context means that a participant may join and leave the system at will, but also that it has special interests for some queries. For clarity, we refer to this kind of environments as *autonomous environments*. Several e-commerce sites [2], [4], volunteer computing projects [6], [3], [5], Web services applications [8], and multi-agent systems [14] are only some examples of autonomous environments.

Participants' interests reflect their *intentions* to allocate and perform queries. On the one hand, a consumer's intention may represent, for instance, its preferences towards providers (e.g. based on reputation) or the quality of service it expects. On the other hand, a provider's intention may denote, for instance, its preferences (e.g. their topics of interests or relationships), strategies, or load. Google AdWords [4] clearly illustrates such participants' interests. When clients (the consumers) query Google for some information, Google replies, in part, with commercial sites (the providers) relevant to their queries and proposes to providers potential consumers. However, participants' interests are only based on some predefined topics (keywords) while their interests may be dynamic. For instance, a provider could represent a pharmaceutical company, which wants to promote a new insect repellent. Thus, during the promotion, it is more interested in treating the queries related to mosquitoes or insect bites than general queries. Once the advertising campaign is over, its intentions may change.

Most current query allocation techniques for distributed information systems focus on distributing the query load among providers in a way that maximizes overall performance [9], [10]. This is obviously important for the efficiency of the system. Nevertheless, in autonomous environments, participants usually have certain expectations, which are not only performance-related, and hence may become *dissatisfied* with the queries they perform. Hence, they may leave the system by dissatisfaction, which causes a loss of processing capacity in the system. As a result, one may have a system with poor performance (low throughput and high response times). This motivates the development of a query allocation technique that satisfies participants so as to preserve the total system capacity, i.e. the aggregate capacity of all providers. In this context, a participant's *satisfaction* means that the query allocation technique meets its intentions in the long-run.

To capture this intuition, we proposed a general model to characterize, in the long-run, participants' *intentions* [12]. This model allows analyzing query allocation techniques implemented by a mediator from a *satisfaction* point of view. In [12], we also proposed a query allocation process, called $SQLB$, to solve the query allocation problem in autonomous environments. $SQLB$ allows trading consumers' *intentions* by providers' *intentions* based on their *satisfaction*. Furthermore, it affords consumers the flexibility to trade their preferences for the providers' reputation and providers the flexibility to trade their preferences for their utilization. In [11], we proposed a strategy, called K_nBest, to adapt the query allocation process to the kind of applications.

In this demo, we present the *Satisfaction-based Query Allocation* framework (S_bQA for short) and demonstrate its flexibility and efficiency to allocate queries using the *Berkeley Open Infrastructure for Network Computing* (BOINC) [1] as an example of highly autonomous environment. However, even if we only consider BOINC as example application in this demo, S_bQA is suitable for many more applications such as e-commerce and Web services. S_bQA uses K_nBest and $SQLB$ as the basis to perform query allocation. We demonstrate that: *(i)* the proposed satisfaction model allows analyzing different query allocation techniques no matter their query allocation principle, and *(ii)* S_bQA performs well in autonomous environments by satisfying participants and ensuring low response times as well. In particular, we demonstrate that: *(iii)* thanks to $SQLB$ the query allocation process is self-adaptable to the participants' expectations, and *(iv)* thanks to K_nBest we can adapt the query allocation process to the kind of applications by varying its parameters.

The rest of this demo is structured as follows. In Section II, we describe the way in which participants obtain their *satisfac-*

tion. We present the query allocation framework in Section III. Finally, in Section III, we present the demo overview.

II. Satisfaction Model

We discuss in this section how a participant computes its *satisfaction*. In [12], we proposed a complete model where we also define the *adequation* and *allocation satisfaction* notions in addition to *satisfaction* one. However, for this demo, we only present the *satisfaction* notion. The *satisfaction* notion may have a deep impact on the system, because participants may decide whether to stay or to leave the system based on it. The *satisfaction* notions are based on the k last interactions that a participant had with the system. The k value may be different for each participant depending on its memory capacity. For simplicity, we assume that they all use the same value of k. The *satisfaction* notion can be expressed with respect to participant's *intentions* (context dependent and hence dynamic data) or with respect to its *preferences* (context independent and hence static data). For simplicity, we only present the *satisfaction* definitions for the participants' *intentions*.

The consumer's *satisfaction* notion allows to evaluate whether a mediator is allocating the queries of a consumer to the providers which it wants to deal with. The *intentions* of a consumer, whose values are in the interval $[-1..1]$, to allocate its query q to providers in set P_q are stored in vector $\overrightarrow{CI}_q$. We define the *satisfaction* of a consumer $c \in C$ concerning its query q as follows,

$$\delta_s(c,q) = \frac{1}{n}\Big(\sum_{p \in \widehat{P_q}} \big(\overrightarrow{CI}_q[p] \ + 1\big) \Big/ 2 \Big) \qquad (1)$$

where n stands for the number of results required by the consumer and $\widehat{P_q}$ denotes the set of providers that performed q. Values of function $\delta_s(c,q)$ are in the interval $[0..1]$. Given the above equation, we define the *satisfaction*, in the long-run, of a consumer $c \in C$ as the average of its obtained *satisfactions* concerning its k last queries recorded in set IQ_c^k (see Definition 1). Its values are between 0 and 1. The closer the *satisfaction* to 1, the more a consumer is satisfied.

Definition 1: Computing Consumer's Satisfaction

$$\delta_s(c) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \delta_s(c,q)$$

The provider's *satisfaction* notion evaluates whether the mediator is giving queries to a provider according to its expectations (those of the provider) so that it fulfills its objectives. Thus, a provider is simply not satisfied when it does not get interesting queries for it. To evaluate this, a provider tracks its expressed *intentions*, whose values are in the interval $[-1..1]$, to perform the k last proposed queries in vector $\overrightarrow{PPI}_p$. We define the *satisfaction* of a provider $p \in P$ in Definition 2, where set SQ_p^k denotes the set of queries that provider p performed among the k last proposed queries.

Definition 2: Computing Provider's Satisfaction

$$\delta_s(p) = \begin{cases} \frac{1}{||SQ_p^k||}\Big(\sum\limits_{q \in SQ_p^k} \big(\overrightarrow{PPI}_p[q] + 1\big) \Big/ 2 \Big) & \\ 0 & if\ SQ_p^k = \emptyset \end{cases}$$

Its values are in the interval $[0..1]$. The closer the value to 1, the greater the satisfaction of a provider.

III. Query Allocation Framework

We now briefly describe how S_bQA works. The general system architecture is shown by Figure 1. Given an incoming query q and the set of providers that are able to perform q (denoted by set P_q), a mediator, based on the K_nBest strategy [11], first selects a set K of k providers at random among set P_q. Then, it selects the k_n less utilized providers, denoted by set K_n, from set K. After this, running $SQLB$ [12], it asks for $q.c$'s *intention* for allocating q to each provider $p \in K_n$. Also, it asks for K_n's *intention* for performing q. Once it obtains this information, the mediator computes the score of each provider $p \in K_n$ by making a balance between $q.c$'s and p's *intentions* and computes the ranking of providers in K_n. Finally, the mediator allocates q to the $q.n$ best *scored* providers in set K_n and sends the mediation result to the consumer and all providers in set K_n. We discuss further how the mediator selects providers below. Details about K_nBest and how a participant computes its *intention* can be found in [11] and [12].

The mediator allocates a query q to the $min(n, k_n)$ "best" providers, which are given by vector of ranking $\overrightarrow{R}$. Intuitively, $\overrightarrow{R}[1] = p$ if and only if p is the best ranked, $\overrightarrow{R}[2]$ stands for the second best ranked and so on. In $SQLB$, the mediator computes this vector $\overrightarrow{R}$ from that provider with the highest *score* to that having the lowest *score*. The mediator scores a provider p by considering its *intention* for performing q and the *intention* of consumer $q.c$ for allocating q to p. Formally, the score of a provider $p \in P_q$ regarding a given query q is defined as the balance between the $q.c$'s and p's *intentions* as in Definition 3.

Definition 3: Computing Provider's Score

$$scr_q(p) = \begin{cases} \big(\overrightarrow{PI}_q[p]\big)^{\omega} \big(\overrightarrow{CI}_q[p]\big)^{1-\omega} & if\ \overrightarrow{PI}_q[p] > 0 \wedge \\ & \wedge\, \overrightarrow{CI}_q[p] > 0 \\ -\Big(\big(1 - \overrightarrow{PI}_q[p] + \epsilon\big)^{\omega} \big(1 - \overrightarrow{CI}_q[p] + \epsilon\big)^{1-\omega}\Big) & else \end{cases}$$

Vector $\overrightarrow{PI}_q[p]$ denotes P_q's *intentions* to perform q. Parameter $\epsilon > 0$, usually set to 1, prevents the provider's *score* from taking 0 values when the consumer's or provider's *intention* is equal to 1. Parameter $\omega \in [0..1]$ reflects the importance that the query allocation method gives to the consumers' and providers' *intentions*. To guarantee equity at all levels, the mediator ensures such a balance (ω) in accordance to the consumers' and providers' *satisfaction*. Formally, the mediator computes ω as in Equation 2.

$$\omega = \Big(\big(\delta_s(c) - \delta_s(p)\big) + 1\Big) \Big/ 2 \qquad (2)$$

The idea is that if the consumer is more satisfied than the provider, then the mediator pays more attention to the

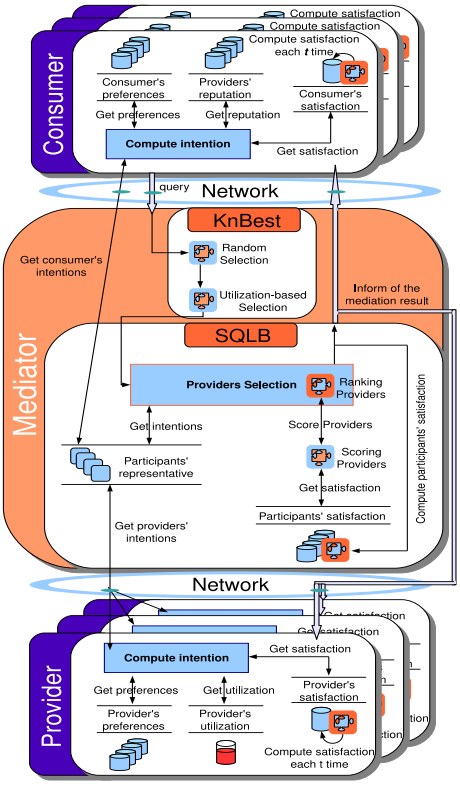


Fig. 1. System architecture.

provider's intention. One can also set the value of parameter ω in accordance to the kind of application. For instance, if providers are cooperative (i.e. not *selfish*) and the most important is to ensure the quality of results, one can set ω near or equal to 0.

IV. Demonstration Overview

We implemented S_bQA in Java and for the demo we simulate the system network using SimJava. The S_bQA prototype provides a set of GUIs that enable the user to setup the experimentations as well as to display all the relevant information (e.g. participants' satisfaction, response times, and S_bQA settings) to illustrate how S_bQA performs. Figure 2 shows some of these GUIs.

As said so far, we use BOINC as an example of highly autonomous environment to demonstrate the flexibility and efficiency of S_bQA. BOINC is a middleware system for volunteer computing. In this context, the consumers are projects, which are usually from the academia, that require computational resources to perform queries and the providers are volunteers that donate computational resources to BOINC-based projects. Participants (i.e. both consumers and providers) in BOINC are autonomous as stated in Section I. A query is an independent computational task, specified by a set of input files and an application program. Incoming queries are dispatched by a server (the mediator) to providers. As providers may be malicious, consumers may create several instances of a query so as to validate results returned by providers.

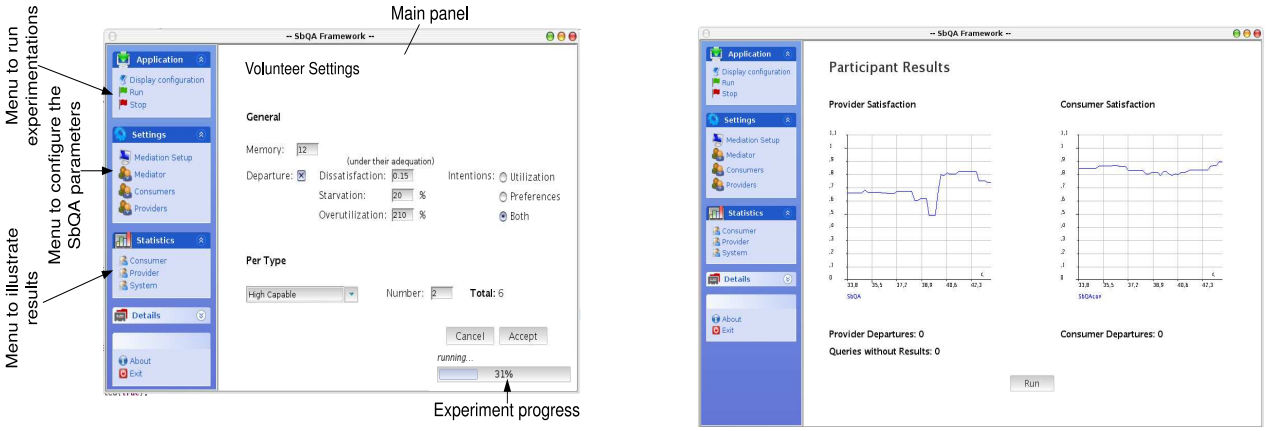
In BOINC, providers can express their intentions by specifying the fraction of computational resources devoted to each consumer. This allows providers to devote more resources to those consumers (projects) in which they are interested. However, this may waste idle computational resources of providers when their interesting consumers do not issue queries. For example, a provider may donate its computational resources to two consumers c_a and c_b in a fraction of 80% and 20%, respectively. In this case, c_b cannot use more than the assigned 20% of computational resources even if c_a is not generating queries. S_bQA could allow BOINC-providers to express their intentions in a more flexible way so that their donated computational resources be properly exploited while their intentions be also satisfied. On the other side, consumers cannot express their intentions with respect to providers in BOINC. Our framework may be used by BOINC designers to allow consumers to express intentions towards providers such as reputation-based preferences.

The example scenario we consider for the demo consists for simplicity of three consumers, i.e. three different research projects. For clarity, we assume that those projects are the SETI@home [6], proteins@home [5], and Einstein@home [3]. We create a set of volunteers devoting their computational resources to all three projects in a way that: *(i)* SETI@home is popular, i.e. the majority of providers want to collaborate in this project, *(ii)* proteins@home is normal, i.e. great number, but not most, of providers want to collaborate in this project, *(iii)* Einstein@home is unpopular, i.e. most providers desire to collaborate, in this project, with a small fraction of computational resources.

In this demo, we mainly focus on the validation of the proposed satisfaction model, the way in which queries are allocated by S_bQA, how S_bQA adapts the query allocation process to the participants' expectations, and how S_bQA can be adapted to the kind of applications. With this in mind, we consider the seven scenarios below. People attending the demo are able to see the demonstration of all scenarios in an interactive way. That is, they are able to set new experimentation values and see on-line how S_bQA performs.

Satisfaction Model: **Scenario 1**. First of all, using the proposed satisfaction model, we compare the way in which BOINC allocates queries, which is equivalent to a $Capacity\ based$ [9] query allocation technique, with an economic technique [13] from a satisfaction point of view. In this evaluation, we assume *captive environments*, that is, participants are not allowed to quit the BOINC platform. An example of these environments is when consumers use BOINC as platform for grid computing and they put in dedicated computers at their service [7]. This scenario demonstrates that our satisfaction model allows analyzing different query allocation techniques even if the way in which they allocate queries differs.

Scenario 2. We evaluate again baseline techniques, as in Scenario 1, but this time considering that BOINC is used as platform for volunteer computing, i.e. when participants are autonomous to leave the system. On the one hand, we assume that a provider leaves the BOINC platform if its satisfaction



(a) Capturing volunteer settings

(b) Drawing results on-line

Fig. 2. Some S$_b$QA GUIs.

is smaller than 0.35. On the other hand, we assume that a consumer stops using BOINC if its satisfaction is smaller than 0.5. This scenario allows seeing that using our satisfaction model one can predict possible participant's departure by dissatisfaction.

Query Allocation Process: **Scenario 3**. We evaluate S_bQA in an environment as in scenario 1 and compare its performance results (participants' *satisfaction* and response times) with those of baseline techniques. In such a comparison, we show that S_bQA's performance is not far from those of baseline techniques. This shows that S_bQA is suitable for captive environments even if it was not designed for.

Scenario 4. We run again the evaluation of Scenario 3 but, now, in autonomous environments instead of captive ones. Our objective is to illustrate that S_bQA can significantly improve the performance of BOINC-based projects by preserving most volunteers online and hence more computational resources.

Adaptation to participants' expectations: **Scenario 5**. We consider the same evaluation of Scenario 3, but we modify the manner in which participants compute their *intentions* so that projects be interested only in response times and volunteers be interested in their load. In this case, we show that S_bQA significantly improves response times and balances better queries among volunteers, which is what participants prefer. This proves that S_bQA adapts to the participants' interests and thus can deal with heterogeneous participants (from their interests point of view), which may allow BOINC-based projects to have more volunteers.

Application Adaptability: **Scenario 6**. We consider an application whose goal is to ensure low response times to consumers and that is still composed by autonomous providers. We assume again that participants compute their *intentions* by considering their preferences. An example of this application is when the BOINC platform is used for grid computing, but the computational resources composing the grid are still donated by volunteers. In this context, besides ensuring low response times, BOINC should ensure some level of satisfaction at the providers' side so that they do not quit their resources from the grid. We demonstrate that S_bQA can be adapted to perform in such applications by varying parameter k_n of the K_nBest strategy and the manner in which the mediator scores providers, i.e. by varying parameter ω.

Playing a BOINC-participant role: **Scenario 7**. We allow people attending the demo to play the role of a consumer or provider. The goal is to enable a person to set her own preferences and intentions, and observe how the different mediations react and which ones allow her to reach her objectives. Allowing this, people attending the demo can obtain a clear picture of the performance that the different mediations may have when they are confronted to human participants having different interests. In this scenario, we aim at demonstrating that the $SQLB$ mediation used by S_bQA is the only one that allows a participant to reach its objectives in all cases.

References

[1] BOINC Platform, http://boinc.berkeley.edu.
[2] The eBay System, http://business.ebay.com.
[3] The Einstein@home Project, http://einstein.phys.uwm.edu.
[4] Google adwords, http://adwords.google.com.
[5] The proteins@home Project, http://biology.polytechnique.fr/proteinsathome.
[6] The SETI@home Project, http://setiathome.berkeley.edu.
[7] The SZTAKI Project, http://desktopgrid.hu.
[8] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architecture, and Applications*. Springer, 2004.
[9] P. Ganesan, M. Bawa, and H. Garcia-Molina. Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems. In *Proceedings of the Very Large Data Bases Confeference (VLDB)*, 2004.
[10] F. Pentaris and Y. E. Ioannidis. Autonomic query allocation based on microeconomics principles. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2007.
[11] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. K$_n$Best - A Balanced Request Allocation Method for Distributed Information Systems. In *Proceedings of the Database Systems for Advanced Applications Conference (DASFAA)*, 2007.
[12] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. SQLB: A Query Allocation Framework for Autonomous Consumers and Providers. In *Proceedings of the Very Large Data Bases Confeference (VLDB)*, 2007.
[13] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A Wide-Area Distributed Database System. *Journal on Very Large Data Bases (VLDBJ)*, 5(1):48–63, 1996.
[14] K. P. Sycara. Multiagent Systems. *AI Magazine*, 19(2):79–92, 1998.