Allocation Problems in Scalable Distributed Data Structures [*]

ADRIANO DI PASQUALE
Università di L'Aquila, Italy

MICHELE FLAMMINI
Università di L'Aquila, Italy

ENRICO NARDELLI
Università di L'Aquila, Italy
Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", CNR, Italy

Abstract

We investigate the servers allocation problem in a network of workstations, a first step toward the analysis of more realistic scenarios for Scalable and Distributed Data Structures. Unlike the previous frameworks, we consider the cost of the messages exchanged among the sites in terms of time or length of the paths traversed in the network. The immediate consequence is that the choice of the position in the network of a new server after a split of an overflowing one is not a trivial problem. We concentrate on both off-line and on-line settings, and provide optimal and nearly optimal approximation algorithms.

1 Introduction

In this paper we consider the problem of allocating servers in a network of workstations. Informally, given a set of already allocated servers S in a network G, an overloaded server $s \in S$, in order to balance its load, may ask for the creation and placement of a new server t in a node of G not already occupied by another server. A given allocation of a set of servers S in G is evaluated in terms of the following two cost measures:

1. The maximum distance $d(s,t)$ in G between pairs of servers s, t such that t is generated by s.

[*]Research partially supported by the European Union TMR project "Chorochronos", by the EU RTN project ARACNE and by the Italian "Progetto Cofinanziato: Allocazione di risorse in reti senza filo".

2. the diameter of S in G, that is the maximum distance in G between two servers of S.

The allocation problem has several applications. A typical example is the SDDS context [1]. In such a framework whenever a server s is in overflow, that is it manages more data than its capacity, a *split* operation is performed: s instantiates a new server t and migrates to it approximatively half of its data. An algorithm solving the allocation problem (that we call allocation algorithm) with the above mentioned requirements is a basic component of the *split* operation procedure. The first requirement is important in order to minimize the startup cost of a new server, as large amount of data must be migrated spending a time proportional to the servers distance. The second one is important because the diameter of the graph is an upper bound on the cost of query and update messages in terms of time or length of the paths traversed in the network.

In all the previous works on SDDS the complexity measure of an operation is given by the number of messages exchanged between clients and servers. However, this hides many important parameters of real world distributed systems, such as the size of a message and the network topology with special concern to the time needed to deliver messages to their destinations. In some sense, the topology of the network has been always modelled as a complete graph and the allocation algorithm inside the split operation simply consists in determining any free node in the network.

However, if a more realistic scenario is considered, the allocation problem cannot be trivially solved any longer and allocation algorithms like the ones developed in this paper become of fundamental importance for the determination of efficient SDDS operations.

Another application of the allocation problem is the server mirroring in the internet. Like for SDDS, an overloaded server may decide to replicate itself to a new mirror located in another free node, performing a procedure similar to the split operation of SDDS, although here a complete copy of the data is carried out between two servers. Again, the distance between a server and its new mirror is responsible of the startup cost, while the diameter is important to bound the complexity of update operations necessary to maintain consistency between the different copies of a same initial server.

In this paper, given a bound on the diameter of the allocated servers, we concentrate on the determination of solutions that minimize the maximal distance between pairs of servers involved in a split operation. In particular, we provide optimal and nearly optimal approximation algorithms, both in the off-line setting in which all the sequence of splits is known in advance and in the on-line one in which at each step a new server must be allocated without knowledge of the future splits.

The paper is organized as follows. In Section 2 we give an example of a more realistic framework for the SDDS paradigm. In Section 3 we introduce the servers allocation problem and the related definitions. In Section 4 and 5 we

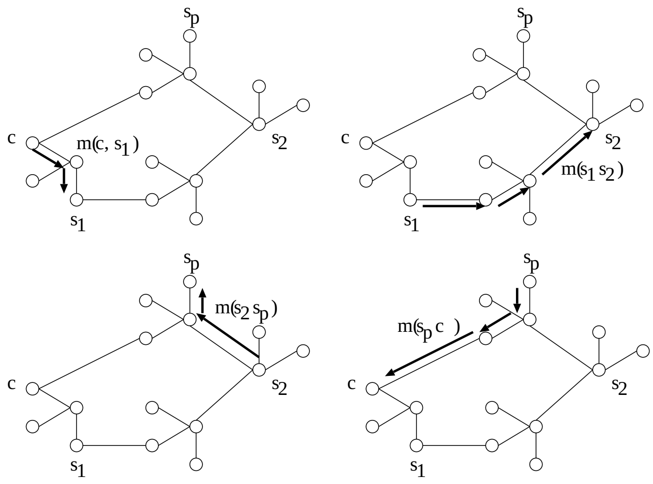

Figure 1: The client c performs a request pertinent for server s_p with address errors at servers s_1 and s_2. After performing the request, server s_p answers to the client c.

provide the above mentioned approximation algorithms respectively for the off-line and on-line cases. In Section 6 we give some conclusive remarks and discuss some open problems.

2 SDDS in a network with a given topology

In all previous works on SDDS, a standard assumption is to measure the communication complexity of an operation as the number of messages exchanged by clients and servers. This could not be adequate for a real world network with a given topology, where in general a site is not directly connected to all the other sites.

A more realistic SDDS framework consists in modelling the network as a weighted graph G where nodes correspond to sites, arcs to communication links and arc weights to the length of the links or equivalently the time needed to traverse them.

Each message has a cost proportional to its delivering time, plus a negligible elaboration overhead of yielded by the destination to handle the message. Therefore, assuming that routing is performed along shortest paths, a message $m(u,v)$ sent from a site u to a site v has a cost equal to the distance between u and v in G, or analogously the sum of the weights of the arcs traversed by the message.

Figure 1 shows an example of exchange of messages among sites in this context.

During the split of a server s_i, it is necessary to choose the position in the network of the new created server s_j. Under the above assumptions, such a choice influences both the split cost and the cost of the next requests.

The allocation policy for the allocation of s_j has to minimize:

1. The distance from s_i, because this is proportional to the cost of migrating keys during the split.
2. The diameter or maximum distance between the locations of the servers, because this is an upper bound to the cost of the exchanged messages.

Notice that the policy based only on the minimization of the distance from a splitting server could lead to a high diameter. Similarly, minimizing only the diameter could yield high distances from the splitting servers. Therefore, a nice combinatorial optimization problem arises in which one performance measure is traded for the other.

3 The allocation problem

In this section we give a formal definition of the allocation problem. For the sake of simplicity, in the following we will refer to the basic terminology used for SDDS, although all the shown results directly apply also to other contexts like the mirroring of servers in the internet.

Every sequence of splits generating a set of servers S correspond in a natural way to a rooted tree T in which the root is given by the initial server and a server associated to an internal node of T is created by a split of its father.

The *Min Server Allocation* problem (Min-SA) can then be defined as follows.

Instance:	A graph $G = (V,E)$, a set $S = \{s_0,...,s_{h-1}\}$ of h servers, a position of the initial server s_0 in G, the tree T describing the sequence of the $h-1$ splits generating S starting from s_0 and a positive number $r \geq 0$.
Solution:	Embedding of T in G such that $d_G(s_0,s) \leq r, \forall s \in S$.
Measure:	$\max_{(s_i,s_j)\in T} d_G(s_i,s_j)$

Above, “Solution” characterizes the space of the feasible solutions and “Measure” gives the function to be optimized (in this case minimized) by the returned solution. Therefore, the goal of the problem is that of determining, among all the possible embeddings in which the distance of all the servers from the initial server s_0 is bounded by r, the one that minimizes the distance between the servers s_i and s_j such that s_j is created with a split of s_i.

Notice that by definition any feasible solution for the problem has also diameter bounded by $2r$. This property is important, because we can directly apply the previous results on SDDS to obtain an estimation of the computational complexity of the operations. In fact, if we consider the basic model in which the cost of an operation is just the number of exchanged messages and $C(n)$ is the complexity cost of a given SDDS with n servers, then in the general case the cost is bounded by $2r \cdot C(n)$.

Before concluding the section, let us observe that Min-SA can be investigated in two separated realistic settings. The former is off-line and assumes that the sequence of all the splits is known in advance. The latter is on-line and consists in allocating each server generated by a split without any knowledge of the future splits. Both cases are considered in the following two sections.

4 The off-line problem

Unfortunately, Min-SA is an intractable problem.

Theorem 1 *Min-SA is NP-hard.*

Proof.

In order to prove the claim it is sufficient to show that any polynomial time algorithm for Min-SA could be exploited for deciding in polynomial time if there is a Hamiltonian path in a graph G starting at a given node v, a problem well known to be NP-complete.

Given a graph G of n nodes and the first initial node v instance of the above decision problem, we construct the following instance of Min-SA. The network coincides with G, $r = n-1$, $S = \{s_0, \ldots, s_{n-1}\}$, the position of s_0 is v and the tree T of the splits is simply a chain with s_0 as the first endpoint and an edge connecting every pair of servers s_i, s_{i+1} with $i < n-1$. Then, there exists a Hamiltonian path in G starting at v if and only if the chain T can be embedded in G with maximum distance equal to one between every pair of successive servers in T, that is if and only if there exists a solution of measure equal to 1 for Min-SA. $\square$

Notice that the above negative result holds even if the tree of the splits T is a chain. Since this case has practical applications and can be considered as a building block toward the solution of the general problem, in the following we concentrate our attention on efficient approximation algorithms for chains of splits.

Our algorithm A_{off} is based on an interesting result concerning the embedding of chains in trees (see [2]). Namely, a chain of h nodes can be embedded in any tree of $n \geq h$ nodes starting at any node v and with maximum dilation equal to 3, that is in such a way that the distance in the tree between two adjacent nodes in the chain is at most 3. Such an embedding has the additional property that, while the first node of the chain is embedded in v, if $n = h$ the last node is

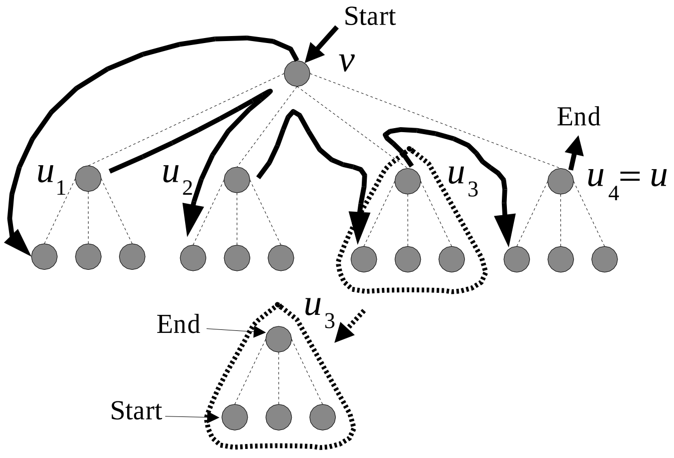

Figure 2: Embedding a chain of servers in a tree with dilation at most 3.

embedded in a neighbor of v in the tree. In the following we provide a sketch of such a procedure (see Figure 2).

Let G be any n-node tree with root v and height h, and let $u_1, u_2, ..., u_d$ be the children of v. If $h = 1$, we are done since all the nodes in G are at distance at most equal 2 the embedding is trivial. Otherwise, we proceed inductively as follows. Place the first server of the chain at v and the second server at any child of u_1 (if any). This induces an edge of dilation 2. Next use induction to place servers in each node of the subtree of G rooted at u_1, making sure to place the last node at u_1 itself. By induction, these edges can have dilation at most 3.

The next server is placed at any child of u_2 (if any). This induces an edge with dilation 3. Again we use induction to place servers in the subtree of G rooted at u_2, ending at u_2.

We continue in this fashion until only the subtree rooted at u_d is left. We first enter this subtree at child of u_d (if any) and then exit at u_d, completing the embedding of the chain of servers.

Such a procedure can be performed in polynomial time.

The algorithm A_{off}, given a weighted graph $G = (V, E)$ of n nodes, $r > 0$, a set $S = \{s_0, \ldots, s_{h-1}\}$ of $h \leq n$ servers, the position of s_0 in G and a tree T coinciding to the chain of the servers $s_0, \ldots, s_{h-1}$, performs the following steps.

1. Let $V_r \subseteq V$ be the subset of the vertices of G at distance at most r from s_0 and let G_r be the weighted complete graph defined on V_r in such a way that for every pair of nodes $u, w \in V_r$ the edge $\{u, w\}$ has weight equal to $d_G(u, w)$.

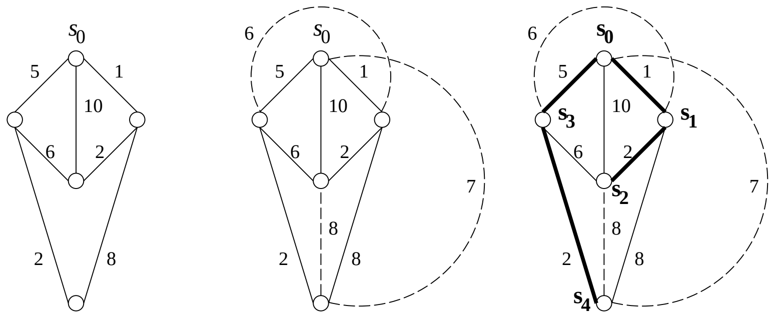

Figure 3: The steps of Algorithm Aoff. A graph G and the position of s_0 (left). The complete weighted graph G_r (center). The minimum spanning tree $T_r(h)$ for $h = 5$ and the allocation of the servers $s_1, ..., s_4$ (right).

2. Using the Prim's algorithm [3], determine the minimum spanning tree T_r of G_r and let $T_r(h)$ be the subtree of T_r induced by the first h vertices chosen by the algorithm (the position of s_0 included).

3. Determine the embedding of the chain T in T_r starting at the node v containing s_0 and ignoring the edge weights, using the above mentioned embedding procedure.

4. Return the determined embedding of T in G.

Lemma 1 *Given an instance of Min-SA, let δ be the maximum weight of an edge in the tree $T_r(h)$ determined by A_{off} and let m^* be the value of an optimal solution for the instance. Then $\delta \leq m^*$.*

Proof. Let us assume by contradiction that $m^* < \delta$. Then by hypothesis there exists a sequence $\langle v_0, ..., v_{h-1} \rangle$ of h distinct nodes in G such that v_0 is the location of s_0 and for each i, $0 \leq i < h-1$, the distance between v_i and v_{i+1} in G is strictly less than δ.

Consider the first step j of the Prim's algorithm in which an arc of weight δ is chosen, and let v_i be the first node of the sequence $v_0, ..., v_{h-1}$ not belonging to $T_r(j-1)$, that is the subtree induced by the first $j-1$ chosen nodes. Then, since v_{i-1}, v_i does not belong to $T_r(j-1)$, at step j an edge of weight δ is chosen instead of one of weight strictly less than δ: this clearly contradicts the greedy choice of the Prim's algorithm. $\square$

As a direct consequence, the following theorem holds.

Theorem 2 *A_{off} is a 3-approximation algorithm for Min-SA.*

Proof.

By the definition of A_{off}, T is embedded in T_r with dilation 3, that is in such a way that for every pair of adjacent servers s_i, s_{i+1} in T, $0 \leq i < n-1$, there is a path of at most 3 edges in T_r between the positions of s_i and s_{i+1}. By the previous lemma, it results $d_G(s_i, s_{i+1}) \leq 3 \cdot \delta \leq 3 \cdot m^*$, hence the theorem. $\square$

For what concerns lower bounds on the achievable approximation ratio, we observe that, using the same arguments of Theorem 1, it is possible to show that if $P \neq NP$ no polynomial time α-approximation algorithm for Min-SA exists for $\alpha < 2$, as it would decide the Hamiltonian Path problem with fixed initial node.

5 The on-line problem

In this section we focus on on-line algorithms for Min-SA. Namely, at each split the allocation of the created server must be performed without knowledge of the future splits. Again, we compare the value of the returned solutions with the minimum achievable one, even using off-line algorithms.

Since the number of splits is not known in advance, it makes sense to consider a more general problem in which r is not fixed, but depends on the best possible allocation at each step. Namely, given a new relaxation parameter $b \geq 1$ replacing r in the input instance of the problem, denoted as r_i the minimum radius from s_0 of an allocation of i servers (s_0 not included), a given solution is feasible at the i-th step or split if all the allocated servers are at distance at most $b \cdot r_i$ from s_0.

Our algorithm A_{on} is very simple, and consists regardless of b in placing at each step the new created server in the free node closest to s_0. Therefore, every server at step i is at distance at most r_i from s_0, and thus the solution returned by A_{on} is always feasible.

The following theorem is a direct consequence of the fact that all the servers allocated by A_{on} are at distance at most $2r_i$ from s_0 at every step i.

Theorem 3 *A_{on} is a $2r_n$-approximation algorithm for Min-SA.*

Notice that A_{on} can be simply generalized to the case in which the topology generated by the splits is a tree and not only a chain, without increasing the approximation ratio.

According to the following theorem, A_{on} is optimal.

Theorem 4 *Min-SA does not admit any α-approximation online algorithm with $\alpha < 2r_n$.*

Proof. For the sake of brevity we prove the claim only for $b = 1$.

Consider an unweighted chain with an odd number of nodes and the server s_0 initially located at node in the middle of the chain. Any on-line algorithm has to place s_1 and s_1 in the two nodes adjacent to s_0, s_2 and s_3 in the two nodes at distance 2 from s_0, and so forth. The resulting embedding has dilation $2r_n$.

$\square$

The above theorem holds even if we consider approximation algorithms having an exponential running time. In some sense, $2r_n$ can be seen as the error factor that must be paid for our missing knowledge of the future.

6 Conclusions

We studied the servers allocation problem in a network of workstations. This is has a direct application in the SDDS paradigm if we consider the cost of the messages exchanged between the sites in terms of time or length of the paths traversed in the network. All the previous SDDS proposals have always considered simplified communication complexity assumptions. Therefore, our work can be considered as a first step towards the analysis of more realistic scenarios and the determination of more effective solutions.

Two main questions are left open. First of all, we have the filling of the $2 \div 3$ gap between the lower and the upper bound on the achievable approximation ratio in the offline case. The proof of the online $\alpha \geq 2r_n$ lower bound for $b > 1$ requires edge weights. Thus, for such values of b a second open question concerns the determination of suitable lower bounds on unweighted graphs.

References

[1] W. Litwin, M.A. Neimat, D.A. Schneider: LH* - Linear hashing for distributed files, *ACM SIGMOD Int. Conf. on Management of Data*, Washington, D. C., 1993.

[2] F.T. Leighton: Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes, *Morgan Kaufmann Publishers*, San Mateo, California, 1992.

[3] R.C. Prim: Shortest Connection Networks and Some Generalizations. *Bell System Technical Journal*, 36:1389-1401, 1957

Adriano Di Pasquale is with the Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, Coppito, I-67010 L'Aquila, Italy. E-mail: dipasqua@di.univaq.it

Michele Flammini is with the Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, Coppito, I-67010 L'Aquila, Italy. E-mail: flammini@univaq.it

Enrico Nardelli is with the Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, Coppito, I-67010 L'Aquila, Italy and with the Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", Consiglio Nazionale delle Ricerche, Viale Manzoni 30, I-00185 Roma, Italy. E-mail: nardelli@univaq.it