Algorithms for services and network resources allocation using graph embedding

Christian Cadere, Dominique Barth, Sandrine Vial

Algorithms for services and network resources allocation using graph embedding †

Christian Cadéré and Dominique Barth and Sandrine Vial

PRiSM - UMR 8144, Université de Versailles St Quentin, 45, avenue des États Unis, F-78035 Versailles, France

La virtualisation et l'allocation efficace de ressources réseaux et de services est l'une des clés des applications distribuées avec QoS. Dans cet article, nous étudions le problème du point de vue des plongements dans les graphes. Après avoir défini les principaux problèmes,nous proposons des heuristiques et évaluons leurs performances.

Keywords: distributed computing, QoS, optical network, resource mapping allocation

1 Introduction

Technologies for current distributed applications, such as Grids , already support sharing and coordinated use of various resources located in resource-end systems. Optimized services network operators would use the connectivity provided by high-speed networks. Indeed, virtualizing the set of computing and communication resources in order to place professional users into a collaborative working space will bring benefits in terms of productivity and control of expenditure. Thus the objective is to lead the network to support real-time distributed applications involving both service and communication resources. One aim of the CARRIOCAS French project [Aud07] is to develop a high bandwidth optical network to support distributed applications and, in particular, to extend to a whole network the execution of GRID-like applications. One issue of the project is to design a Virtualization Entity (VE) allowing the network to allocate *ad hoc* communication and service resources to execute distributed applications with the QoS required by its customers. This VE also has to provide an environment allowing a customer to describe a distributed application. Such a description is based here on workflow graphs [NTHE+05].

In this paper we focus on an implementation of a VE based on algorithms which map workflows graphs [NTHE+05] on a target network.

The theoretical tool we deal with is graph embedding. An embedding of a graph G in a graph H consists in an injective function, p, mapping the vertices of G to the vertices of H and in a routing function, R, associating to each edge of G a path between the embeddings of its two extremities in H (see [DPS02]). Three parameters, potentially used as constraints, are usually considered to evaluate the quality of an embedding. The first one, the dilation, is the maximal length of a path associated by R to an edge of G in H. The second one, the congestion, is the maximal number of such paths using a same edge in H. The last one, the load, is the maximal number of vertices of G mapped on the same vertex in H. Minimizing the dilation and/or the congestion of an embedding with load constraints are NP-complete problems (see [DPS02] and refs). In this paper, we embed a workflow graph, representing the target application in a graph representing the network. The three parameters given above are considered as constraints: dilation is related to communication delay constraints, congestion is related to bandwidth requirements and limitations, and load is related to service availabilities. The objective is to find an embedding respecting these constraints which minimizes the network resources allocated in the network to run the target application.

This paper is organized as follows. Section 2 presents the model and gives results concerning the complexity and inapproximability of this problem. A polynomial heuristic algorithm is presented and its performances are analyzed by simulation in Section 3.

†Partially supported by the Project CARRIOCAS of the french competitiveness cluster SYSTEM@TIC

2 Model and problem complexity

We first give the graph definitions we use to describe both network resources, distributed applications and embeddings. A network is modeled by a set S of services, each one provided by some of its nodes. In the following, each service in S is denoted by a number in $\{1,..,|S|\}$.

Definition 1 *Given a set of services S, a* **network** *is a symmetric digraph $H = (V',A')$ with three functions:* cap*: giving the capacity available on each arc.* del*: giving the delay of each vertex.* s*: giving for each node the quantity of each service of S that it provides.*

Definition 2 *Given a set of services S, an* **application** *is a digraph $G = (V,A)$ with four functions:* band*: giving the required bandwidth of each arc.* lat*: giving the maximal latency of each arc.* sv*: indicating the service needed by each vertex. q_s: indicating the quantity of the service needed by each vertex.*

The function q_s models the resource allocation (processing time, memory size, data availability) for QoS guarantees concerning services. Our objective is to map an application in a network with respect to QoS constraints related to communications and services using graph embedding.

We define a **configuration** of an application in a network by respecting some QoS constraints as follows:

Definition 3 *Given a set of services S, an application $G = (V,A)$ and a network $H = (V',A')$, a* **configuration** *of the application in the network is an embedding (p,R) of G in H that satisfies a resource constraint, a bandwidth constraint and a delay constraint.*

Note that in the following, we will also deal with the configuration of a set of applications, i.e. an union of disjoint applications. An application is satisfied if the VE (Virtualization Entity) provides for it a configuration in the network. The efficiency of the VE can be measured in terms of the overall percentage of applications it can satisfy over time. In terms of complexity, the problem of maximizing this percentage can be defined as follows: Given a set of services S, a set $\mathcal{A}$ of N applications, a network $H = (V',A')$ and an integer k, is there a subset $\mathcal{A}' \subseteq \mathcal{A}$ of size k for which there is a configuration in H? Similar problems for various domains have been shown to be NP-complete [DPS02]. We conjecture that this problem is in Σ_2 [GJ79]. We have shown in [CBV08] that even if the network topology is a tree and if every vertex provides a different service, this problem is NP-complete. Considering the NP-completeness of Problem $Max\_Application\_Satisfaction$ for trees and since in the CARRIOCAS project, application requests have to be managed in an online context, we focus on the problem of finding a configuration for one application by minimizing the amount of resources it uses, i.e., links and intermediate nodes. The main problem, denoted Min_Config_Search, we then have is to determine, if it is possible, the configuration of an application in a network in such a way to minimize the cost in terms of used links of the associated embedding. We especially focus on the case where the application is a star. We consider networks such that there is no bandwidth constraint and no delay constraint This special case is the problem of the Steiner tree problem which is NP-complete [GJ79], thus proving the NP-completeness of Min_Config_Search. Since the Steiner Tree problem is approximable within 1.55 [RZ00], we can conclude that Problem Min_Config_Search can be approximated for a very restricted set of instances. Nevertheless, in the general case, we have proved in [CBV08] that the problem is inapproximable.

3 Online heuristic approach and simulations

Since Problem Min_Config_Search is NP-complete and not approximable, we describe a heuristic algorithm to solve it and we evaluate its efficiency in an online context to solve $Max\_Application\_Satisfaction$. Let us first describe this algorithm. Given an application G and a network H, Algorithm_1 searches a configuration of G in H that minimizes the number of links used in H.

Algorithm_1(G,H)

1. For each arc of G, compute each possible pair of mappings of its extremities that fulfill the resource constraint.
2. For each pair computed this way, compute all the paths from the mapping of the origin of the arc to its destination's mapping respecting the delay constraint.

3. For each valid path from the origin to the destination, compute its cost in terms of resource occupation, depending on the percentage of free bandwidth it will use on each arc. Use the path that has the lowest cost function. The cost we use is defined by an exponential function of the available bandwidth use, in order to penalize the congestion of a link.

This algorithm may have a very long computation time with some special network topologies, but in practice, it requires a small computation times.

3.1 Simulation an performance evaluation

We have randomly generated fifty applications and a network. Each application has five vertices and between four and six arcs (according to an uniform law) and a random topology. The service needed by each vertex is generated according to an uniform law between 1 and $|S|$. The quantity of the service needed is also generated according to an uniform law between 1 and a specified value of the maximal service requirement. A random network is generated as follows: the first step consists in generating a random tree connecting all the vertices, and some arcs are added randomly after. The connectivity is then ensured. Random delays are generated for the vertices according to an uniform law between 1 and 10. Once the network and the applications are generated, we overlap all the applications on the network. That means that we create in the network just enough resources to map the fifty applications at the same time. In order to do that, each application has its resources added in the network: for each arc of the application, its extremities are randomly mapped in vertices of the network that are allocated the needed quantity of service. Communication resources are allocated on a shortest path between those vertices while the delay of the arc of the application is updated according to the delays of the vertices of the shortest path, in order to make possible to ensure the delay constraint. Once the applications and the network are created, we try to find a configuration for the fifty applications in the network taken in a random order and we compute the mean number of satisfied applications. We split the unsuccessfully configured applications in three categories: those who failed because of lack of services, those who failed because of lack of bandwidth and those who failed because of the delay constraint. In order to study the importance of each constraint, separate simulations were made where one of these constraints was relaxed while the other did not change. In figure 1, 100% of service means that the network is in its original state, i.e., that it has the exact amount of vertex resources for the applications. In figure 2, 100% capacity means also that the network is in its original state. Figure 1 shows the results of the increase of available resources in the network vertices. The line with square dots shows the percentage of unused service resources on the vertices of the network. We begin with 0% of the needed resources in the network and increase it step by step until 1000% and see that the quantity of successfully configured applications increases also (as well as the computation time). Indeed, by increasing the quantity of services provided by the nodes, we offer a wider range of possibilities for application vertices to be mapped in the network, and thus increase the paths combinations. We observe that the percentage of successful configurations stabilizes around 78% after a 800% increase of service resources. Figure 2 shows the results of the increase of the available capacity on the links of the network, beginning with 0% of the capacity to 800%. We see that while the link capacity increases, the number of successful configurations also increases, and the main cause for configuration failures is the delay constraint. The number of rejected configurations due to the resource constraints remains very low. With a 400% capacity on the network links, the percentage of successful configurations seems to reach a plateau around 77%.

Due to lack of space, we cannot explain all our experimentations, but in some of them, we observed that if the delay of the network vertices is decreasing, the percentage of satisfied applications increases. The main cause for configuration failure is then the capacity constraint. These results show that the amount of available resources in regard to the quantity that is needed is one of the main factor of efficiency for the algorithm with the drawback of higher computation time. Having a margin on all the constraints helps to achieve good results in terms of satisfied applications, but has an impact on the network cost. In order to compare these results, we compared them to a heuristic approach that was similar to Algorithm_1, but a bit simpler and faster to compute. The Algorithm_2 works as follows. For each arc of the workflow, compute each possible pair of mappings of its extremities in order to fulfill the resource constraint. For each pair calculated this way, compute the shortest path according to the delay of the crossed vertices. Use the shortest path of all the pairs as the mapping of the arc. When we compared the results of Algorithm_1

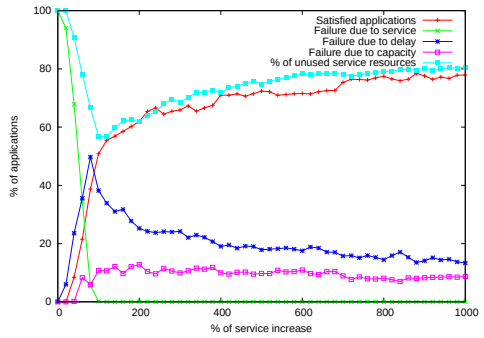

Figure 1: Evolution of satisfied applications with service increase

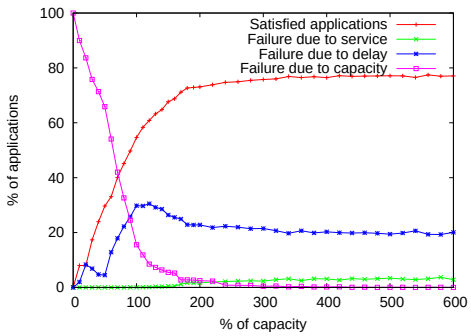

Figure 2: Evolution of satisfied applications with capacity increase

to those of this "simple" algorithm, we found that the mean of successful configurations was always about 5% better with Algorithm_1 because it explores more possibilities by computing all possible paths, but with the drawback of a computation time up to 50% higher than Algorithm_2.

4 Conclusion

We have proposed here a heuristic approach to solve Problem *Max_Application_Satisfaction* in an online context based on a strategy trying to minimize the number of resources (edges and nodes) used by each consecutive application. An important issue would be to compare the efficiency of this strategy to one based on a load balancing strategy . A work in progress is also to give some results about competitive ratio for some families of network topologies.

References

[Aud07] O. Audouin. CARRIOCAS project: an experimental high bit rate optical network tailored for computing and data intensive distributed applications. In *Proc. of SPIE*, volume 6783, 2007.

[CBV08] C. Cadéré, D. Barth, and S. Vial. Virtualization and allocation of network service resources using graph embedding. Technical Report 2008/1, PRiSM, Université de Versailles-Saint Quentin, 2008.

[DPS02] J. Diaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34:313–356, 2002.

[GJ79] M. R. Garey and D. S. Johnson. *"Computers and intractability. A guide to the theory of NP-completeness"*. W.H. Freeman and Co., 1979.

[NTHE+05] Russell N., A.H.M. Ter Hofstede, D. Edmond, , and W.M.P. Van der Aalst. Workflow resource patterns. Technical report, Eindoven University of Technology, 2005.

[RZ00] Gabriel Robins and Alexander Zelikovsky. Improved steiner tree approximation in graphs. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 770–779, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.