netEmbed: A Service for Embedding Distributed Applications[†]

[Extended Abstract]

Jorge Londoño[‡]
CS Department
Boston University
Boston, MA
jmlon@cs.bu.edu

Azer Bestavros
CS Department
Boston University
Boston, MA
best@cs.bu.edu

1. INTRODUCTION

An increased number of applications, such as computational grids, testbeds, peer-to-peer networks, and sensor networks (among many others) rely on finding a set of resources that meet certain criteria for their operation. In particular, in many of these cases their requirements may be described as a labeled graph where nodes represent computational resources and links represent connectivity/communication requirements. Similarly, the infrastructure where the service will be deployed is also described by a labeled graph, where the attributes of nodes and links represent their capabilities. The problem of finding a feasible set of links and nodes on which to deploy the service is what we call the *embedding problem*.

As an illustrative example, consider the problem of mapping a sensor network application where nodes represent either sensing or computation operations and the application needs to find a set of resources subject to some constraints, *e.g.* the sensed variable, the location the sensor; the computation nodes need to be within some delay from the sensors to process real time data, and must have at least certain bandwidth to meet the sensor data transfer rate. The queries need to be processed at the nodes imposing a constraint in their computational power. Finally, the results have to be delivered to the clients, which gives some additional communication requirements. This simple scenario illustrates the sort of constraints and requirements that an embedding service must provide.

[†] This work is supported in part by a number of NSF awards, including CISE/CSR Award #0720604, ENG/EFRI Award #0735974, CISE/CNS Award #0524477, CNS/NeTS Award #0520166, CNS/ITR Award #0205294, and CISE/EIA RI Award #0202067.

[‡] Supported in part by the Universidad Pontificia Bolivariana and COLCIENCIAS–Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología "Francisco José de Caldas".

MC'07, November 26-30, 2007, Newport Beach, CA
Copyright 2007 ACM ISBN 978-1-59593-935-7/07/11...$5.00

The problem of finding the set of resources that match the requirements is clearly a combinatorial search/optimization problem. In particular, we distinguish the search as finding *feasible embeddings* and the optimization as finding the *best embedding* with respect to an optimality metric. Typical solutions in the bibliography have considered various heuristics to solve this problem. That is the case of Emulab/NetBed [1, 8] where both, simulated annealing and genetic algorithms have been used. Also, SWORD [6] considers some pruning heuristics, tailored for the particular case of PlanetLab, in order to significantly reduce the search space. Both approaches sacrifice soundness, though. They may return false negatives: a no-solution answer when in fact there is a solution.

On the other hand, work in other areas has proposed the usage of constraint satisfaction techniques. For example, *gang-matching* for Condor [7], and Redline [4], use constraint satisfaction techniques to match the jobs' requirements with the computational capabilities of the nodes, the available software at the nodes, etc. However, none of these approaches take topology into consideration. This issue becomes particularly important as distributed applications are being deployed in WAN environments, where link capacities and delays have a significant influence in the overall performance of the application.

In the context of overlay networks, the work in [3] showed how constraint satisfaction techniques could be efficiently applied to the solution of embedding problems. Our work follows the same lines by providing techniques that improve the performance, while preserving the soundness (*i.e.* only return true positives) by never pruning feasible parts of the search space.

2. NETEMBED FRAMEWORK

Our framework is designed to take three inputs: 1) a description of the *infrastructure* where the overlays are going to be deployed. This description is given in the form of a labeled graph, where labels describe capabilities of links and nodes; 2) a description of the *overlay*, also as a labeled graph, but this time the labels represent the overlay's requirements; and 3) a *constraint expression* that establishes the conditions on how to match the overlay's requirements and with the infrastructure's capabilities. We use the GraphML [2] standard to represent both the *infrastructure* and the *overlay*, and

the *constraint expression* is given as a boolean formula relating the attributes of both. A mapping is feasible if this expression is true for all links and nodes.

For solving the mapping problem, we developed three techniques:

Exhaustive Search with Constraint Filtering – ECF: This algorithm follows the standard technique of exploring the space using a depth-first search, with various optimizations: 1) Pruning the current branch as soon as it becomes infeasible, 2) Reordering the per-node candidate sets by increased cardinality, which significantly reduces the size of the search space, and 3) Precomputing sets feasible mappings for links, so that given the current mappings, the next node can be found using the intersection of these sets. The combination of the three techniques significantly reduces the search space, making it possible to obtain feasible mappings very quickly (in some cases all feasible mappings) [1] .

Random Walk with Backtracking – RWB: This technique is basically a variation of the previous one. Instead of doing an ordered exploration of the search space, it does a random walk pruning the search space as soon as infeasible regions are discovered, pre-computing candidate sets and reordering by the cardinality of the candidate sets. Experimental results showed that there is not significant difference in performance as compared to the ECF technique. Further analysis revealed that when constraints are tight, both ECF and RWB spend most of the time in the precomputation stage and are very fast finding the actual solution or determining that there is none.

Lazy Neighborhood Search – LNS: This technique uses the concept of arc-consistency to construct a feasible mapping. It begins by selecting a feasible mapping for a random node. It then chooses one of its neighbors and finds a feasible mapping for it. It continues growing the set of mapped nodes by choosing always a neighbor of the current set and looking for a feasible mapping for it. If found, this mapping is accepted into the current feasible set. By growing the feasible set in this fashion, at the end, when all the nodes of the overlay have been mapped, the mapping is complete. On the other hand, if at some point it turns out that there is not a valid mapping for a neighbor node, the current set is infeasible and the algorithm has to backtrack. By keeping lists of neighbors for both, the overlay nodes in the feasible set, and their corresponding mappings in the infrastructure network, this technique effectively prunes infeasible regions of the search space. However, because of the need of backtracking, the worst case complexity is still exponential.

3. EVALUATION AND CONCLUSIONS

Experimental evaluation using as hosting infrastructure both PlanetLab (which is of fixed size), and synthetic network topologies (much larger than PlanetLab), showed that these techniques can handle problems of a few thousand nodes size with running times ranging from a few seconds to a few minutes. This improves both the size and response time with respect to previously used techniques, although a fair comparison is not yet ready, as there is not yet a set standard benchmarks.

The ECF and RWB techniques proved to be quite effective for tightly constraint problems, *i.e.* problems where the constraints significantly limit the number of candidate matchings. However, when the constraints are loose (many candidate mappings per node/link) the filtering technique is expensive in space (the space requirement is $O(n^5)$), limiting their scalability. On the other hand, the as the LNS approach does not do filtering and the amount of state information kept is small, it is very effective in handling problems with loose constraints, whereas for tightly constraint problems it has to do a lot more backtracking and it is not as effective as ECF and RWB. The complementary nature of these techniques makes them very appropriate for implementing a hybrid solution, first running a pre-computation step that would determine if the input constraints produce very few candidates per node, in which case it would proceed with either the ECF or RWB technique, or otherwise using the LNS technique.

Additional information, as well as a demonstration of this project is available at: `http://csr.bu.edu/netembed/`

[1] We conducted benchmarks on a 2.4Ghz Pentium Xeon system, using PlanetLab as a hosting infrastructure (296 sites) and configurations of various sizes up to 200 nodes with running times below 10sec in all cases

4. REFERENCES

[1] C. Alfeld, J. Lepreau, and R. Ricci. A solver for the network testbed mapping problem. *SIGCOMM Comput. Commun. Rev.*, 33(2):65–81, 2003.

[2] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. Marshall. GraphML progress report: Structural layer proposal. In Springer-Verlag, editor, *Proc. 9th Intl. Symp. Graph Drawing (GD '01), LNCS 2265*, pages 501–512, 2001.

[3] J. Considine, J. W. Byers, and K. Mayer-patel. A constraint satisfaction approach to testbed embedding services. In *Proceedings of HotNets-II*, November 2003.

[4] C. Liu and I. Foster. A constraint language approach to matchmaking. *14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04)*, 00:7–14, 2004.

[5] J. Londoño and A. Bestavros. netembed: A network resource mapping service for distributed applications. Technical Report 2006-32, Boston University, Boston, MA, December 2006.

[6] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Design and implementation tradeoffs for wide-area resource discovery. In *14th IEEE Symposium on High Performance Distributed Computing (HPDC-14)*, July 2005.

[7] R. Raman, M. Livny, and M. Solomon. Policy driven heterogeneous resource co-allocation with gangmatching. In *International Symposium on High Performance Distributed Computing (HPDC03)*, pages 80–89, June 2003.

[8] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.