Self-Aggregation Techniques for Load Balancing in Distributed Systems

Elisabetta Di Nitto, Daniel Dubois, Raffaela Mirandola
Politecnico di Milano
Piazza Leonardo da Vinci, 32
20133 Milano, Italy
(dinitto,dubois,mirandola@elet.polimi.it)

Fabrice Saffre, Richard Tateson
BT group
Adastral Park
Ipswich IP5 3RE, UK
(fabrice.saffre,richard.tateson@bt.com)

1 Introduction

One of the today issues in software engineering is to find new effective ways to deal intelligently with the increasing complexity of distributed computing systems. In this context a crucial role is played by the balancing of the work load among all nodes in a system composed of interconnected nodes that enter and exit the system without following any rule.

To address this issue, we are experimenting with the usage of autonomic self-aggregation techniques that rewire the system in groups of homogeneous nodes that are then able to balance the load among each others using classical techniques.

We present our approach together with some simulation experiments that show how the application of self-aggregation algorithms makes it possible to balance the load also in these extreme situations. Besides, our experiments show that the introduction of self-aggregation does not introduce a significant overhead in terms of execution time, even if it requires the exchange of a higher number of messages between nodes.

2 Dynamic Load Balancing

Assume a network of interconnected nodes. Each node can be seen as a resource that is able to process jobs. Each node corresponds to a type that defines which job(s) it is able to process. In this kind of networks the purpose of Load-Balancing is to distribute the jobs evenly to all the nodes with the aim to: (1) increase the job processing rate of the whole network; (2) increase the number of nodes involved in a computation, and reduce, at the same time, their utilization.

In literature the *Diffusive Load Balancing Algorithm* and the *Dimension Exchange algorithm* [2] are the main algorithms that, using simple local rules and knowledge, are able to balance the workload of a network. Both of them have been formally studied and their convergence has been mathematically proved in [6]. In the first one, during each iteration, a node balances its load with all its neighbors, while in the second one it balances its load with just one random neighbor. Some related approaches in literature can be seen in [4, 1]. In this study we consider the dimension exchange approach that limits its interactions to pairs of nodes. This makes this algorithm less sensitive to synchronization issues.

These algorithms do not conceive the possibility to have various nodes and jobs of different types coexisting in the same network (*heterogeneous case*). This is due to the implicit assumption that this kind of algorithms work only on homogeneous network domains, where a homogeneous domain is defined as a connected subgraph of the original network that has only nodes of a single type.

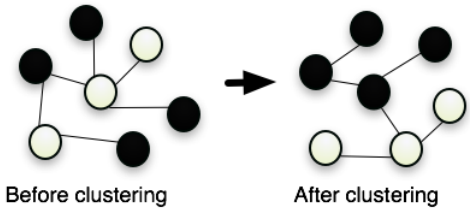
We devise the following strategies to solve the load balancing problem in heterogeneous networks:

- make the jobs traverse the incompatible nodes;
- modify the links of the network (rewire) in order to aggregate the domains of the same type.

The first solution is not applicable because the nodes are not able to forward the jobs directly to their target since they do not have enough global information about the network. In the second method we need to aggregate the nodes that can process the same job type into unique domains.

3 Self-Aggregation for LB

The final purpose of a self-aggregation (or clustering) algorithm is to rewire the network with the aim to reduce the number of links between nodes having different types and to add new links between nodes having the same type. Where a type can be any characteristic of the node. The figure below shows an example of clustering:

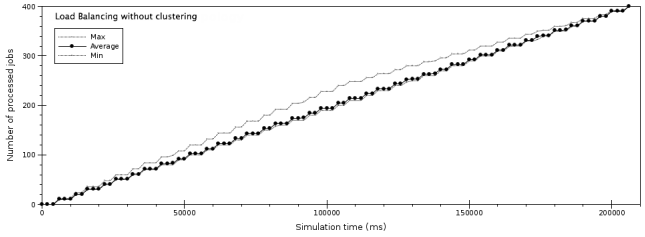


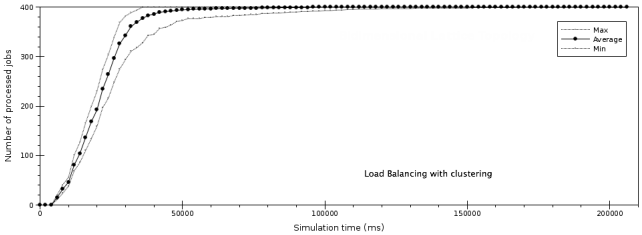
The interesting aspect of self-aggregation is that clustering is not executed by a centralized entity, external to the network. Instead, it is executed in a distributed way thanks to the ability of each node to autonomously take a simple "disconnect/maintain the link" decision on the basis of the type of each neighbor. In [3] and [5] we have presented the self-aggregation algorithms we have identified so far. To overcome the inherent limitations of classical load balancing algorithms, the algorithm we have selected for the reconfiguration of network topology is the Adaptive Clustering Algorithm presented in [3]. This algorithm runs in parallel with the Load Balancing algorithm in order to enhance its convergence rate, and therefore maximize the throughput of the system. The self-aggregation algorithms work performing continuous iterations of themselves on all the nodes of the network: they are started when the network is created and stay active forever. The only information that is used and modified is the list of neighbors of each node involved in an iteration. The Dimension Exchange Algorithm is activated when a node has in its neighbors list at least a node of the same type and its queue of jobs is not empty. It can modify only its list of queued jobs and the one of its neighbors. Since each algorithm modifies always different node properties, no conflicts are possible.

4 Experimental Results

Methodology To set up these experiments we have used a simulation framework that we have implemented for this specific purpose. The parameters we have considered are the following: number of nodes, average node degree, number of types, number of initial jobs, method of sending jobs, job processing time, initial topology. The experiments output that has been considered contains the number of completed jobs and the number of messages exchanged by each node over time.

Results The following charts show the results of a very representative experiment in which we have used the following input parameters: 100 nodes, 10 types, 4 links per node, static load of 400 initial jobs among 10 random nodes (40 jobs/node per type), bidimensional lattice topology, job processing time of 5 seconds. The first one shows the situation without self-aggregation, and the second one the same situation with self-aggregation.





As we would expect the number of processed jobs shows an improvement since the Load Balancing algorithm is now able to work in larger homogeneous domains.

Remarks The main advantages of this approach are given by its scalability, and efficiency in balancing the jobs. However the main drawbacks are inherited by the ones that have been already investigated in [3], that are the dependency of the convergence rate on the initial topology, and the message overhead that is constantly added to provide the reconfiguration that is needed to build and preserve the clustered topology.

5 Conclusions

Our experiments show that the introduction of self-aggregation improves the overall load balancing and does not introduce a significant overhead in terms of execution time, even if it requires the exchange of a higher number of messages between nodes. Future extensions to this work include an extensive simulation phase in order to validate these results in a more realistic environment.

Acknowledgments

This work has been partially supported by Project CASCADAS (IST-027807) funded by the FET Program of the European Commission.

References

[1] G. Canright, A. Deutsch, and T. Urnes. Chemotaxis-inspired load balancing. In *Proceedings of the European Conference on Complex Systems*, Nov. 2005.

[2] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *J. Parallel Distrib. Comput.*, 7(2):279–301, 1989.

[3] E. Di Nitto, D. J. Dubois, and R. Mirandola. Self-Aggregation Algorithms for Autonomic Systems. In *Proceedings of Bionetics '07*, Budapest, Hungary, December 2007.

[4] N. Nehra, R. B. Patel, and V. K. Bhat. Routing with load balancing in ad hoc network: A mobile agent approach. In *ACIS-ICIS*, pages 489–495. IEEE Computer Society, 2007.

[5] F. Saffre, R. Tateson, J. Halloy, M. Shackleton, and J. L. Deneubourg. Aggregation Dynamics in Overlay Networks and Their Implications for Self-Organized Distributed Applications. *The Computer Journal*, 2008.

[6] P. Sanders. Analysis of nearest neighbor load balancing algorithms for random loads. *Parallel Comput.*, 25(8):1013–1033, 1999.