An Overview of the Gridline Project

Youssef Hamadi
Microsoft Research Ltd.
7 J J Thomson Avenue
Cambridge CB3 0FB, UK
youssefh@microsoft.com

Alan M. Frisch and **Ian Miguel**
Artificial Intelligence Group
Department of Computer Science
University of York
York YO10 5DD, UK
{frisch, ianm}@cs.york.ac.uk

Abstract

Grid computing leverages and generalises distributed computing by focusing on large scale resource sharing for high performance and innovative applications. The Gridline project aims to develop a set of advanced optimisation services, based on constraint technology, which add efficiency to this generalised sharing. This position paper details three demonstrators and presents important modelling points.

Introduction

Grid computing involves sharing resources distributed across multiple administrative domains. The majority of grid research has focused on the problems raised by accessing multiple domains (authentication, performance, etc) (Foster, Kesselman, & Tuecke 2001). This has resulted in standard definitions[1] and will eventually meet the world of web services (Foster *et al.* 2002). However, resource sharing raises the problem of efficient selection and aggregation.

The goal of the Gridline project is to study the applicability of Constraint Programming (Dechter 2003) in grid resource optimisation. This technology has been very successful in industrial applications (Wallace 1996). This success comes from its high level of expressiveness along with an easy integration with imperative programming. Domains of application include scheduling and allocating both human resources (e.g., crew rostering and nurse scheduling) and material resources (e.g., airport gates and transport fleets). Gridline assumes that grid resource sharing could greatly benefit from the application of constraint programming.

The project employs constraint-based optimisation to produce three demonstrators in the grid context. The first considers advanced resources reservation, taking the rationale of a grid resource broker that maximises its utility by choosing the optimal set of customers orders. Advanced reservation will play a major role in grid systems.[2] Research has hitherto focused on defining the reservation formats. Gridline addresses the next step: efficiency in reservation selection.

[1] See the Global Grid Forum at www.gridforum.org.

[2] The Grid Resource Allocation Agreement Protocol Working Group is looking at the standardisation side of this problem.

The second demonstrator provides a distributed application mapping service. It uses a resource model of the application and searches for an optimal placement according to criteria such as price, locality, and security. This demonstrator uses information on the location and definition of resources as provided by the UDDI standard.[3]

The last demonstrator defines a web service composition engine. The composition is presented as a workflow. The engine maps each component of the flow to a particular web service. The mapping must ensure that connected components use compatible protocols. Optimality criteria consider characteristics such as locality, security, protocols and price.

The following sections detail each demonstrator.

Demo 1: Advanced Resource Reservation

In this grid usage scenario, a broker manages some bounded resource, such as network bandwidth or a computational farm. Since the resources are limited per time unit, the provider may be unable to meet all demands, so the broker must choose which requests are to be accepted. The idea here is to maximise the utility of the broker by selecting an optimal subset of customers' requests.

Figure 1 presents the architecture of this demonstrator. Gridline receives the customers' priced requests via the broker. It also obtains the status of potential resources from the grid or directly from the broker. Gridline then computes an optimal selection of orders according to prices and penalties. This selection is forwarded to the broker who notifies customers of selection/rejection.

Gridline acts as a particular web service used by the broker to optimise its business. There is no direct connection between customers and the Gridline service. This is an important privacy consideration: Gridline can be completely blind to customers' identities and can receive fake but relatively correct (homothetic) information on pricing.

Customer orders use the following reservation pattern:

- Start/End date,
- Requested QoS, i.e., amount of resources per time unit,
- Price proposed for the service, and
- Penalty if QoS is not satisfied.

[3] See the UDDI technical white paper at www.udi.org.

This can be modelled as a knapsack problem, where the hard constraints ensure that QoS is maintained given bounded resources. The optimisation function maximises the gain (price) while minimising the potential loss. It is important to minimise potential loss since the distributed nature of the grid leads to an increased rate of resource failure.

We will begin by considering allocation of a single resource. It is significantly more complicated when the decision involves multiple resources. The modelling then changes to multidimensional knapsack with a large increase in search complexity (Vasquez & Hao 2001).

Demo 2: Distributed Application Allocation

This second demonstrator provides a fundamental service for the grid. It allows the automated mapping of a given distributed application onto publicly available resources as described by the Metacomputing Directory Service[4] and UDDI. The characteristics of the selected resources must meet application requirements.

Figure 2 presents the architecture of this demonstrator. Distributed applications are modelled via a graph formalism. Nodes represent physical resources such as CPUs or storage. They are defined with a set of capacity constraints. For instance, a given node might need at least 4 CPUs, running at no less than 800MHz on a 'Windows XP operating system. Edges express communication requirements between nodes and are given with bandwidth capacity constraints.

Figure 2 depicts several typical topologies:

- Star topology, expresses client/server applications (multi-user games, collaborative framework).
- Tree topology, expresses typical multi-cast streaming.
- Round topology.
- Grid topology, representative of large parallel computation (simulation, image processing).
- Random topology.

Computed mappings are evaluated by several criteria, such as efficiency, locality and security. In the experiments, the above topologies will be implemented with varying sizes. It will be particularly useful to take advantage of the regularities of some of them. Indeed, symmetries could be used efficiently in the modelling and/or in the search (Gent & Smith 2000). In the validation stage we will assess performance against a UDDI-like "simple" allocation: successive pair-wise matching of resource nodes.

[4] See www.globus.org.

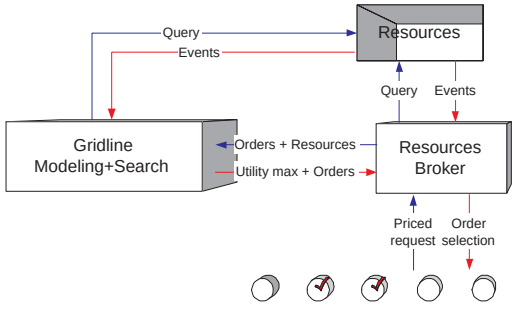

Figure 1: Advanced Resource Reservation

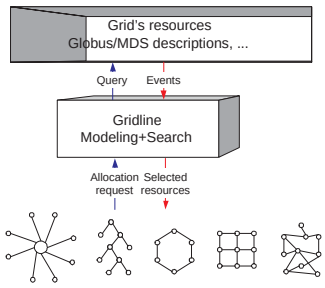

Figure 2: Distributed Application Allocation

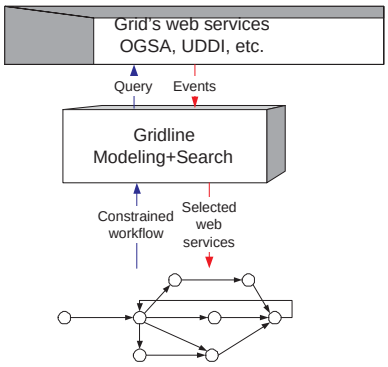

Figure 3: Web service composition

Demo 3: Web Service Composition

Web service composition addresses the goal of ubiquitous e-commerce by combining services owned by different entities to form new web services. Even though this demonstrator is e-commerce oriented, the OGSA convergence makes it relevant to the grid infrastructure. The demonstrator starts from a workflow defining a composition of simple web services (Thatte 2001). It computes an allocation of each workflow's component on compatible web services. Hard constraints represent port/interface compatibility. Bandwidth considerations can be added for some specific compositions (Deelman *et al.* 2003).

Figure 3 presents the architecture of this demonstrator. A web service composition is submitted to Gridline, which can then compute an optimal matching over single web services. The quality of an allocation can be represented with criteria such as locality, price, security level and protocol.

General Requirements

A given problem might be *modelled* in several different ways as a constraint program, which is then solved by search. Some of the resulting models will be more effective than others. To obtain a good model, the structure of a problem

must be examined and exploited. Grid resources have the following salient features:

1. Physical states can change slowly (e.g., RAM size, number of CPUs, network topology), while dynamic states can move rapidly (e.g., access queue size, load).
2. The distributed nature of the grid increases the probability of resource failures.
3. The size of the infrastructure is very important, thousands of resources loosely connected.
4. Grid users will be mostly human (e.g., e-science).

The problems arising in this environment are instances of dynamic flexible constraint satisfaction problems (Miguel 2004). Hence, the following points are key:

Solution robustness (1) and (2) require robust solutions. A solution is robust if it can integrate environment's perturbations. Robustness is a classical track in constraint programming research. It is possible to raise robustness in the search process by adding dedicated value selection heuristics. For instance in the second demonstrator, given two computers of similar price, both of which are compatible with the node's constraints, priority should be given to the more powerful one. It is also possible to integrate dynamic information like reputation to promote robustness.

Sub-optimality The combination of (1) and (2) raises the following question: do we have to pay the cost of an optimal solution search if the environment is very large and still changing? In other words, any optimal solution could rapidly become sub-optimal. If we also consider point (3), that a complete (optimal) search could be quite expensive, the conclusion is that "good" solutions could be enough. We can already expect that those good solutions will be far better than manually computed ones when manual computation is even feasible. Since optimal solutions are not required, local search (Michel & Hentenryck 2000) is a possible approach.

Solution stability (4) introduces the problem of reusing previous solutions. In a constraint-programming framework we call that solution stability. It is possible to encourage stability by minimising a constrained distance between a previous and a currently computed solution (Hamadi 2004).

Predictability The last point also raises the issue of complexity prediction. Indeed, one can imagine that end-users could take advantage of a priori information about the time complexity of the search process. For each demonstrator this knowledge can be partially grasped from a full phase transition study (Slaney & Thiebaux 1998). Those studies provide a full picture of the complexity landscape according to some instance characteristics. The study in this case would be on-line, characterising the difficulty of the distribution of problems input to the system. This could result in information like, "according to the size of that workflow, according to the number of potential matching services, a price-optimal allocation should take ... seconds."

Conclusion

We have presented an overview of the Gridline project. This running project extends the application range of constraint based optimisation to the grid infrastructure. It is built on two observations. First, grid computing involves the sharing, aggregation, selection of distributed resources. Second, constraint based optimisation has proven successful in optimising resources in an industrial context. The successes of the latter should then greatly benefit the former. The project is articulated in three demonstrators. Each demo is presented as a specific client/server service. The first demo implements an advanced reservation optimisation service. The second and third demo respectively compute optimal mappings of distributed applications and workflows. Besides each demonstrator, our work involves the identification of some salient grid features. Those features: robustness, sub-optimality, stability and predictability, are crucial considerations when attempting to achieve efficient and scalable modelling. Hence, a possible extension to Gridline will consider the use of distributed constraint solving technology (Hamadi 2002). This will allow permanent local reallocations in reaction to a changing infrastructure.

References

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Deelman, E.; Blythe, J.; Gil, Y.; Kesselman, C.; Mehta, G.; Vahi, K.; Blackburn, K.; Lazzarini, A.; Arbree, A.; Cavanaugh, R.; and Koranda, S. 2003. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing* 1(1):25–39.

Foster, I.; Kesselman, C.; Nick, J.; and Tuecke, S. 2002. The physiology of the grid: An open grid services architecture for distributed systems integration.

Foster, I.; Kesselman, C.; and Tuecke, S. 2001. The anatomy of the Grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications* 15(3):200–222.

Gent, I. P., and Smith, B. M. 2000. Symmetry Breaking in Constraint Programming. In Horn, W., ed., *Proceedings ECAI 2000*, 599–603. IOS Press.

Hamadi, Y. 2002. Interleaved backtracking in distributed constraint networks. *International Journal on Artificial Intelligence Tools* 11(4):167–188.

Hamadi, Y. 2004. Online Optimization in Internet Data Centers. Technical Report 2004-10, Microsoft Research.

Michel, L., and Hentenryck, P. V. 2000. Localizer. *Constraints* 5(1/2):43–84.

Miguel, I. 2004. *Dynamic Flexible Constraint Satisfaction and Its Application to AI Planning*. Springer.

Slaney, J. K., and Thiebaux, S. 1998. On the hardness of decision and optimisation problems. In *European Conference on Artificial Intelligence*, 244–248.

Thatte, S. 2001. XLANG web services for business process design. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/.

Vasquez, M., and Hao, J. K. 2001. A hybrid approach for the 01 multidimensional knapsack problem. In *IJCAI*, 328–333.

Wallace, M. 1996. Practical applications of constraint programming. *Constraints* 1:139–168.