Bidding support for computational resources

Xavier Vilajosana\textsuperscript{1}, Joan Manuel Marquès\textsuperscript{1}, Ruby Krishnaswamy\textsuperscript{2}, Angel A. Juan\textsuperscript{1}, Nejla Amara-Hachmi\textsuperscript{2}

Leandro Navarro\textsuperscript{3}

\textsuperscript{1}Universitat Oberta de Catalunya
\textsuperscript{2}France Telecom R&D
\textsuperscript{3}Universitat Politècnica de Catalunya
\{xvilajosana,jmarquesp,ajuamp\}@uoc.edu, \{ruby.krishnaswamy,nejla.amarahachmi\}@orange-ftgroup.com, leandro@ac.upc.edu

Abstract

The paper presents a bidding specification language and support tools for Grid-oriented open resource marketplaces. The structure of bids is based on trees that permit bidders to express their preferences for Grid resources by means of logical operators. Additionally, our bidding specification reduces the complexity of existing bidding languages since it permits imprecise description of user’s preferences. One important issue not addressed so far is that of flexible bidding and offer configuration in open resource marketplaces where multiple market mechanisms cohabit. Besides, the paper presents a support tool to adapt any bid to any type of market.\cite{6}

1 Introduction

Open resource markets are increasingly used as mediating processes for efficient resource allocation in Grids. Grids are environments characterized by the heterogeneity and diversity of their resources, applications, applications behaviours, dynamicity and scale. One important issue not addressed so far is that of flexible bidding and offer configuration in open resource marketplaces where multiple market mechanisms cohabit. In those environments buyer agents need to be able to describe their preferences for resources that possibly are being sold in different marketplaces. Furthermore, seller agents need to set their initial offers to be traded by specific market mechanisms. On the one hand a desirable property for bidding specification is that of a common bid/offer specification language that facilitates bid/offer description independently of the market mechanism used to allocate the item. On the other hand, the bidding specification may facilitate the winner determination problem resolution by providing an efficient organization of information.

0Work supported by MCYT-TSI2005-08225-C07-05 and Grid4All(IST-2006-034567).
to sell units of composite resources. For example, aggregated processing, storage capacities, bundled with a set of applications and middleware.

3. **Single Unit or Multiple Unit**: Markets may trade in single or multiple units of an item. In the case of resource markets, it is more often the case where multiple units of anonymous and indistinguishable items are traded, such as multiple units of CPUs for multiple time-slots. Bidding support needs to provide a flexible and compact way to represent the quantity preferences for both buyers and sellers.

4. **Time Factor**: Grid resources are leased. Consumers may have constraints on when the resources are needed and the duration of allocation; similarly providers may trade their resources only for specific times. Hence the bid (and offer) must be able to specify the time ranges within which resources are required and their duration. A typical bid that needs to be supported is that of a consumer that requires two CPUs satisfying attribute description \( \text{CPU} > 1\, \text{GHz}, \text{mem} > 1\, \text{GB}, \text{disk} > 20\, \text{GB} \) for 10 time slots between 10:00 and 18:00 assuming that time-slots are defined to be of 30 minutes. Within single item auctions, the time-slot may also be considered as the item.

5. **Pricing**: Linear pricing sets the same price for each item of resource; bundle pricing sets the price for the entire bundle. Bundle pricing increases the complexity since allocation must choose those bundles that maximize the objective function.

### 3 Related Work on Bidding Languages

Bidding languages have focused on the compact specifications of bidder’s preferences in mainly combinatorial auctions. This is also due to the fact that the optimization problem (to generate the matching and winning bids) is NP-Complete and hence heuristics are required to reduce the length of the problem by providing means to express compactly the preferences. "Logical Bidding Languages" rely on standard logical operators to represent user’s preferences\([4, 5]\). OR bidding languages allow bidders to define non-overlapping bids for a set of resources but not for substitute preferences. In these languages, bidders may face a budget exposure problem. XOR languages, allow bidders to express substitute preferences but all bids from a bidder are mutually exclusive. When a bidder wants any combination of items, he must explicitly bid on each combination, so the process is not scalable since it requires the expression of an exponential number of combinations for even a small number of items. ORX languages provide more expressiveness by means of "phantom variables" however such languages may not be clear to bidders on how to express their requirements [2]. [3] presents the \( L_{GB} \) language that allows the combination of goods with AND, OR and XOR operators. They also extended the language adding the clause \( k\text{-af} \) that allows bidders to express willingness for some \( k \) items within a subset. \( L_{GB} \) provides complete expressiveness to bidders but requires the construction of entire trees to bidders that may not know how to express their requirements. As far as we know the most recent work on bidding languages is TBBL, a tree based bidding language for combinatorial exchanges [1]. TBBL allows double sided exchanges, that is, not only provides semantics for the allocation of goods but also for the reallocation. TBBL has been designed to be concise and structured. It allows for specification of both bids and asks in a single structure, and allows agents to specify upper and lower bounds on their values for trades. TBBL also provides a compact way of representing bids in the tree. As our understanding, TBBL trees will not be portable within different market structures. That is, given a formulated bid, it requires a pre-process to adapt bidder’s preferences to specific market requirements.

### 4 Our approach

Our objective is to provide support for open grid resource markets. In this context, an expressive and flexible bidding specification language is of relevance. The bidding specification language needs to allow bidders to formulate their preferences for Grid resources such the description of complex combinations of items, time and quantity preferences and requirements for full or partial satisfaction.

On the other hand the support to many types of markets is required. For example, in single-item auctions only a single value, the price is communicated. Multi-unit, multi-item, and combinatorial auctions require generic preference structures.

In the next section the descriptive language is presented, section 6 presents how we deal with multiple types of markets.

### 5 A Tree Based Bidding Specification

Our envisioned bidding specification language represents bids in form of a tree where internal nodes contain logical operators (OR, AND or XOR) and leaf-nodes contain specification of bidder’s requirements. The use of logical operators is relevant for the bidding language because enables the users’ preferences elicitation. Specially, for the case of bidding specification for Grid resources since Grid applications usually have complex resource requirements. For example, considering a typical application that requires more than one type of resource (CPU and storage), and multiple quantities of each type of resource, bids should be able to convey preferences on bundles of resources. XOR operators permit the expression of substitute bids. By means of OR operators bidders indicate their willingness to accept partial satisfaction whilst AND operators indicate their requirement for complete satisfaction. Details for Leaf-nodes specification will be introduced in the following subsection.
5.1 Leaf-Node Specification

Leaf-nodes have to provide expressiveness to capture either bidders’ requirements or sellers’ offers. Two notions for resource specification have been introduced, i.e., composite resources and aggregate resources. An aggregate resource represents multiple units of the same type of resource whilst a composite resource represents a bundle of different resource types. These concepts provide flexibility to the market initiators so that they can easily decide which is the product that they will offer in the market. Furthermore, aggregated resources may be conjunctive, that is, the traded item is an atomic aggregation of multiple units of a basic resource that may be allocated entirely. In contrast disjunctive aggregated resources only require a subset to be allocated. This differentiation addresses the AND and OR notions introduced before as well. For example, a compute node is a composite resource (CPU + Storage) and four compute nodes are described as a conjunctive or disjunctive aggregated resource depending on the type of satisfaction required. In addition, leaf-nodes contain the specification of lease times (i.e. start time, end time, time slot size and number of time slots), prices, quantities and allocation constraints. In our design, leaf-nodes permit imprecise description of user’s preferences for a more compact representation of bids. For example, users are not required to build the complete tree but only specify the time range and the number of time slots that they require within the given time range. Indeed, if a bidder wants two any consecutive time slots within a time range, he is not required to build a complete tree structure with an internal XOR node and as many AND nodes as possible combinations of two consecutive time slots within that time range but only build one leaf-node with the lease imprecise description. (e.g. two time slots within the specific time range).

5.2 Implementation

The bidding language is specified as an XML encoded schema that represents the resources required/offered by buyers/sellers and the preferences. A bid can be looked upon as a tree where the non-leaf nodes represent operators such as AND, XOR, OR, and the leaf nodes specify the exact item (resource) description. An item specified at the leaf node should correspond to a resource item traded at the market or as mentioned before be an imprecise item description.

In the XML schema we refer to the non-leaf nodes as NonPrimitiveBids and the leaf nodes as PrimitiveBids. The leaf node includes attributes to indicate the desired quantity of the item, the leasing attributes (start time, end time, duration of time slot, and number of required time slots (for imprecise bids)) and the price. The item may be either a simple resource, a composite resource, or an aggregated resource. Each resource is of one type (CPU, Storage,...) that encapsulates the main attributes of that type of resource. Non-primitive nodes represent the relation between its children nodes, i.e., XOR, AND, OR. Figure 1 contains a code snipped of the item element in the XML schema. Figure 2 and 3 present an example bid for one of two configurations of resources to be used for a specified period of time (3 hours from 12:00 to 15:00). The example bid expresses that the buyer requires one of the following: one CPU of 400 FLOPS, 2 CPUs of 300 FLOPS, or one CPU of 600 FLOPS.

6 Support for multiple auction formats

Generally bidding languages have been developed to support one type of auction, that is, they organize bids in data structures that facilitate the WDP (Winner Determina-
Combinatorial Double Auction Iterative
Item Bundle and single Single Single
Units Multiple Single* Single
Time Multiple Single* Single*
2. Exact preference in quantity but not in time:

(a) **Require time to be consecutive:** A bid B4 requires one CPU of 400 FLOPS for 3 consecutive hours within 11:00 and 16:00. This case requires the formulation of one XOR bid with \((\text{number of available slots} - \text{number of required slots} + 1)\) AND (OR for the case of partial satisfaction bids) sibling bids each one with 3 precise leaf nodes. Case 1 of Figure 5a shows the bid formulated by the bidder for the case when partial satisfaction is required. Case 2, presents the same case when full satisfaction is required. Figure 5b presents the tree completely pre-processed for the case of partial satisfaction.

(b) **Do not require time to be consecutive:** A bid B5 requires one CPU of 400 FLOPS for 3 any hours within 11:00 and 16:00. This case requires to pre-process the bid into a tree with \(\kappa = C_{\text{any}}\) leaf nodes. Due to its size a figure is not provided, however the case is very similar to the one presented in figure 5. Indeed, the compact representation is the same as the figure 5a.

3. Neither exact preference in quantity nor in time:

(a) **Require time to be consecutive:**

   i. **By excess:** A bid B6 requires one CPU of 400 FLOPS at least for 2 consecutive hours within 11:00 and 16:00. In this case, the auction provides complete satisfaction (the auction only accept AND operators), otherwise does not make sense the mandatory ‘at least’. The decomposition needs to generate a bid tree of XOR(root node) and AND due to impreciseness in time (multiple possibilities). The maximum bound on number of time-slots must be set by bidder. Due to the size of the decomposition, the figure 6 only show the decomposition of non-leaf nodes whereas leaf nodes are kept in a compact representation. For the example, the bids are decomposed into precise bids for 2 time slots, 3 time slots and 4 time slots (bound set by the bidder). The complete decomposition (including leaf nodes) would be similar to the decomposition showed in figure 5b.

   ii. **By default:** A bid B7 requires one CPU of 400 FLOPS at most for 2 consecutive hours within 11:00 and 16:00. This case is simpler and similar to the case illustrated in Case 1 of figure 4. Notice that the requirement ‘at most’ indicates partial satisfaction which is expressed with an OR constraint.

(b) **Do not require time to be consecutive:**

   i. **By excess:** A bid B7 requires one CPU of 400 FLOPS at least for 2 any hours within 11:00 and 16:00. The case is similar to the
Figure 6: Exact preferences in quantity but not in time

Figure 7: Neither exact preference in quantity nor in time. Each leaf node is split in: \((\text{number of available slots} - \text{number of required slots} + 1)\) bids

There are two methods to split leaf nodes in the tree.

i. By default: A bid B8 requires one CPU of 400 FLOPS for at most 2 any hours within 11:00 and 16:00. This case needs to generate again \(\kappa = C^m_n\) leaf nodes. The tree will be represented as XOR node in the root, \(\lor\) OR nodes as siblings where each sibling has m leaf nodes. The case however, should be simplified by the bidder requiring specific time slots constrained by the OR operator.

ii. By default: A bid B8 requires one CPU of 400 FLOPS for at most 2 any hours within 11:00 and 16:00. This case needs to generate again \(\kappa = C^m_n\) leaf nodes. The tree will be represented as XOR node in the root, \(\lor\) OR nodes as siblings where each sibling has m leaf nodes. The case however, should be simplified by the bidder requiring specific time slots constrained by the OR operator.

6.2 Current status

The bid decomposition utilities have been developed as a BidManagement component developed using the Fractal component model with Java language mapping that offers a set of functionalities to decompose bids. The component approach facilitates the integration with any auction platform since the component can be bound following a straightforward approach consisting of static specification through the ADL (Architecture Description Language) that describes the system composition and binding of sub-components. Alternatively, standard interfaces like Web Services can be used by any other component to access the offered set of functionalities.

The decomposition functionality requires two parameters, the bid formulated by the bidder using our bidding language and the description what an item is for an specific auction setting. The output for the process is a bid tree semantically identical as the one formulated by the bidder but following the structure required by the auction setting.

7 Conclusions

The paper characterized Grid resources with the aim to provide a bidding language able to be supported by multiple auctions without any modification. The contributions of our presented work, are twofold: Firstly, an expressive Grid oriented bidding specification language developed as an XML schema. Secondly a set of functionalities that provide automatic bid pre-processing for the adaptation of bidders imprecise preferences to different types of auction settings. Our future directions include the integration of the bidding language components into a Grid resource marketplace that focuses on support for multiple auction formats where market rules, algorithms and activities are encapsulated as components.

References


