

# Engineering Large-scale Distributed Auctions\*

Peter Gradwell <sup>a</sup>    Michel Oey <sup>b</sup>    Reinier Timmer <sup>b</sup>    Frances Brazier <sup>b</sup>  
Julian Padget <sup>a</sup>

<sup>a</sup> *University of Bath*  
<sup>b</sup> *Vrije Universiteit Amsterdam*

## Abstract

The functional characteristics of market-based solutions are typically best observed through the medium of simulation, data-gathering and subsequent visualization. We previously developed a simulation of multiple distributed auctions to handle resource allocation (in fact, bundles of unspecified goods) and in this paper we want to deploy an equivalent system as a distributed application. There are two notable problems with the simulation-first, application-second approach. First, the simulation cannot reasonably take account of network effects. Second, how can one recreate in a distributed application the characteristics demonstrated by the mechanism in the simulation. We describe: (i) the refactorings employed in the process of transforming a uni-processor lock-step simulation into a multi-processor asynchronous system, (ii) some preliminary performance indicators, and (iii) some reflections on our experience which may be useful in building MAS in general.

The combinatorial auction (CA) is capable of delivering the optimal solution to a resource allocation problem. Unfortunately, solving combinatorial auctions is costly, and there are circumstances that make combinatorial auctions inappropriate: (i) if resources and bidders are distributed, the centralization intrinsic to a combinatorial auction may be problematic, (ii) under soft real-time constraints, an anytime (sub-optimal) algorithm may be preferable to an optimal algorithm with an unpredictable runtime, (iii) the single-point-of-failure intrinsic to combinatorial auctions may pose an unacceptable risk for system resilience.

In an earlier paper [2], we reported on the economic characteristics and run-time performance of a market-based approach in comparison with a CA, when both are applied to common, standard data sets. In this market-based approach, resources are traded in *continuous double auctions*. There is an auction for each type of resource traded. Together they form a *Multiple Distributed Auction* (MDA). Market *traders* subscribe to one or more auctions in order to sell or buy a bundle of resources, depending on whether they are *sellors* or *buyers* of resources.

In this paper, we present the issues that have arisen in refactoring a uni-processor (centralized) simulation of an MDA into a distributed agent application (using the AgentScape platform). There are two challenges in achieving this transition:

- **Concurrency:** how to introduce *just enough* concurrency to give an advantage but not so much as might paralyze or lead to significant numbers of delicate timing bugs. We refactor the original synchronous simulation a step at a time in order to constrain the available concurrency.
- **Communication:** distributed systems usually embody significant communication overheads. Careful placement of processes and resources helps to exploit locality. In addition, we are able to determine the impact of additional messaging overhead required to facilitate a distributed architecture.

The centralized MDA simulation uses a lock-step model, in which at every step (or round) the following three sub-steps are performed sequentially:

- (i) All traders are instructed to check the status of their current bundle. If the bundle is still in progress, nothing is done. If the bundle has been completed or has failed (the trader has given up), they acquire a new sell or buy bundle.
- (ii) All auctions are instructed to perform one round. A round consists of asking all participating traders to send a shout (a bid or an ask). Any matches will be reported to the corresponding traders.

---

\*The full version [1] of this paper appeared in the Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08).

- (iii) All traders are instructed to get any trade results. At this point, the statistics are updated with completed or failed bundles.

In the centralized simulation, a single thread of execution runs each of these rounds sequentially. However, in a distributed system, processes that have no direct effect on each other can often be executed in parallel. In the case of the MDA simulation, each auction could in principle run in parallel, which could lead to performance improvements. However, in order for the results of the distributed MDA to be comparable to the centralized one, the notion of rounds as described above must still be kept. Fortunately, within each round, we can utilize parallelism to improve performance as, *within* a step, actions can be executed in parallel. In other words, in step one, all traders can check their progress simultaneously; in step two, all auctions can run one round in parallel; and in step three, all traders can process the trade results in parallel.

The distributed implementation uses AgentScape [3], a framework for heterogeneous, mobile agents, as a base. In an agent system, typically the work is divided among several agents, which all perform a part of the work. AgentScape can distribute agents over multiple hosts, and thus spread the load of an application. Porting the centralized MDA simulation to run on AgentScape was relatively straightforward. Each auction and each trader was turned into an agent. AgentScape could then run these agents on different hosts, allowing for parallel execution.

Unfortunately, the distributed simulation ran a lot slower than the centralized simulation. This performance loss was mainly due to the communication overhead between auctions and traders, and the need to synchronize the agents per round. Further analysis of the system resulted in three optimizations, which all aimed at reducing the amount of communication (e.g., method invocations) between agents:

- (i) Cache communication results, whenever possible.
- (ii) Group multiple method invocations that are often called in sequence into a single invocation.
- (iii) Localize communication by placing agents that communicate with each other a lot on the same host.

As with any distributed application, synchronization is not problematic if the cost of doing so is relatively small compared to the amount of work that can be done in parallel. In the MDA application, however, the average time for an auction to process a shout is much less than the time it takes the trader to send the shout to the auction and receive back the results. A larger grain size would reduce the impact of the communication overhead. This could be obtained by allowing traders and auctions to perform multiple steps at once.

In conclusion, MAS software development is characteristically evolutionary and a common starting point is a proof-of-concept system running on a single machine utilizing an agent platform or even a simulation framework. The software engineering challenge lies in how to scale that demonstrator up into a system comprising many more agents running over multiple machines. We have transformed a centralized simulation of a Multiple Distributed Auction into a multi-agent system, supporting large numbers of agents participating in large numbers of auctions on distributed machines. The AgentScape mobile agent platform was used to distribute the entities in the MDA over multiple hosts and to provide the necessary communication between these entities. Unfortunately, experiments have shown that the communication overhead of the distributed market is quite large compared to the benefit of gaining more computing resources.

The lesson learnt here is that synchronizing agents is very time consuming due to the amount of messaging involved. However, the barrier synchronization can be removed, at the cost of a more complex refactoring of the original code, and at the cost of producing *similar* but not identical results to the original code. Thus, the process reported here has been a tedious but necessary step to demonstrate functional *equivalence* before we move on to a situation in which we must define and demonstrate functional *similarity*.

## References

- [1] P. Gradwell, M. A. Oey, R. J. Timmer, F. M. T. Brazier, and J. Padget. Engineering large-scale distributed auctions. In *Proceedings of the Seventh Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. ACM, May 2008.
- [2] P. Gradwell and J. Padget. A comparison of distributed and centralised agent based bundling systems. In *ICEC '07: Proceedings of the ninth international conference on Electronic commerce*, pages 25–34, New York, NY, USA, 2007. ACM Press.
- [3] B. J. Overeinder and F. M. T. Brazier. Scalable middleware environment for agent-based Internet applications. In *Applied Parallel Computing*, volume 3732 of *Lecture Notes in Computer Science*, pages 675–679. Springer, Berlin, 2006.