Decentralized Co-Allocation of Interrelated Resources in Dynamic Environments

Tino Schlegel¹, Ryszard Kowalczyk², Quoc Bao Vo³
Swinburne University of Technology
Faculty of Information and Communication Technologies
Hawthorn, 3122 Victoria, Australia

E-mail: {tschlegel¹,rkowalczyk²,bvo³}@ict.swin.edu.au

Abstract

The vision of transparent, on-demand resource utilization in distributed and open environments requires resource management techniques that are robust, scalable and able to adapt to the dynamic environment. In this paper, we propose a decentralized resource allocation algorithm for the co-allocation of interrelated resources for repeated jobs in real-time. Resource broker agents autonomously allocate resources for the execution of jobs based on information from past allocations. The coordination between agents for an overall efficient resource allocation emerges through individual feedback that agents receive from the quality of previous resource allocation decisions. The coordination between agents is achieved without any communication. We present experimental results demonstrating that the proposed algorithm is able to adapt to the dynamic environment for an efficient utilisation of the system resources.

1 Introduction

The increasing popularity of distributed computing paradigms such as grid and cloud computing has transformed the Internet into a powerful computing platform. Different software peers (e.g. agents) connected via a wide-area network can share existing computing resources across different administrative domains for the execution of jobs. In recent years, the focus of resource allocation research has shifted towards such open and dynamic environments, which pose new challenges that need to be addressed. The main challenges are the lack of full control over the resource allocation process and the uncertainty and lack of available information [9]. These are mainly due to the distribution of resources across different administrative domains.

Research in grid computing has recognised these problems and proposed a number of distributed resource allocation algorithms based on economic market models [10]. Most of these approaches are distributed but often require a central facilitator for pricing, resource discovery or dispatching jobs to resources [3]. The problem is that the degree of centralization of any resource allocation algorithm is an important aspect for open environments as this may limit the scalability of the system size as well as the availability and robustness of the system in particular when the number of resource requests increases [6].

A considerable amount of work on decentralized resource allocation algorithm such as [7, 4, 8] have tackled this problem over recent years. Mainland et al. [5] proposed a resource allocation algorithm for sensor networks with focus on the optimization of the energy consumption. All those approaches apply the principles of inductive reasoning and bounded rationality introduced by Arthur [1]. Their limitation is their ability to only allocate jobs that require one type of resource. However, most applications require the co-allocation of interrelated resources of different types at the same provider. Examples are CPU resources plus system memory to perform computations or disk storage plus network bandwidth for the exchange of data. None of the decentralised resource allocation approaches can handle such allocation requests.

In this paper, we present a decentralized resource allocation algorithm that eliminates this limitation. The presented algorithm is an extension of our previous work [8]. It adds the ability to co-allocate interrelated resources of different types at the same resource provider. The algorithm targets environments that require the allocation of resources in real-time without any provision for queuing jobs such as cloud computing [2]. The main characterisation is that users frequently request for small amounts of resources for a short period of time. This usage pattern requires an efficient resource allocation algorithm that is able to handle a large number of requests and is able to adapt to the changing environment.

The resource allocation decisions are made by intelligent, adaptive resource broker agents, deployed on each client. The agents learn from positive or negative feed-
back of previous resource allocations to improve future allocation decisions and, if necessary, adapt their behaviour to cope with the dynamic environment. This decentralised decision making process requires coordination between resource broker agents to efficiently utilise the available system resources. By efficient resource utilisation, we mean the allocation of as many jobs as possible to the available resource providers without exceeding the capacity of any single provider’s resources. The coordination arises from inductive reasoning and continuous learning from feedback of previous resource allocations. The advantage of this technique is the full decentralization of the resource allocation process assuring the robustness of the system by avoiding a single point of failure. The failure of a broker agent has only a local effect. The system performance is minimally affected if a resource provider fails or becomes unavailable. Another advantage of the decentralisation is a good scalability in the number of clients since there are no direct interactions between them that increases the complexity of the decision making.

The remainder of the paper is organized as follows: The next section presents the resource allocation model, followed by Section 3 with the description of the decentralized resource allocation algorithm. The experimental evaluation of the algorithm is presented in Section 4. Section 5 concludes the paper.

2 Resource Allocation Model

Our model consists of resource providers $P = \{p_1, \ldots, p_h\}$ offering shared resources and resource broker agents $A = \{a_1, \ldots, a_n\}$ allocating resource for the execution of jobs. Figure 1 illustrates the model. The definitions of symbols used in the description are listed in Table 1. The following subsections give a brief description of each component.

Resource Providers offer resources to serve the computational needs of resource consumers. Each provider $p$ is characterized by the resource profile $R_p$. The resource profile specifies the type and amount of provider $p$’s shared resource capacities $C(p, r)$. The offered resources can be shared by multiple jobs simultaneously. The resource utilization of resource $r$ of provider $p$ is calculated by adding the resource demand of all simultaneously executed jobs: $U(p, r) = \sum_{j \in J_t(p)} u_j(r)$. All demands and capacities are specified in discrete resource units, which correspond to real computing metrics such as CPU cycles or system memory in bytes.

Resource Consumers in the system are jobs. The user sends a job to the broker agent with the request to immediately allocate resources for its execution at one of the providers. The resource demand of a job, $j$, is characterized by its resource profile $R_j$. The resource profile specifies the types and amount of resources required for the job execution as well as the execution time and is submitted together with the job by the user.

Resource Broker Agents are located on each client platform as illustrated in Figure 1. When a broker agent receives a job from the user, it allocates resources for the job execution at one of the providers. The allocation is successful if there is a single provider who offers all resource types and the broker agent expects that enough resources for the job execution are available.

3 Resource Allocation Algorithm

The resource allocation process consists of four stages: (1) identifying suitable resource providers for the job execution, (2) selecting one of them and submitting the job for execution, (3) executing the job at the provider and monitoring resources and (4) performing clean-up tasks after the execution has successfully completed. Algorithm 1 lists the pseudo code of the algorithm.

Stage 1: The identification of suitable providers can be done by the broker agent using the local resource directory. The broker agent filters all providers offering all required resource types for the job execution (Alg. 1, line 2).

![Figure 1. Resource Allocation Model](image.png)

Table 1. Symbols and Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J$</td>
<td>Set of resource consumers or jobs.</td>
</tr>
<tr>
<td>$A$</td>
<td>Set of resource brokering agents.</td>
</tr>
<tr>
<td>$P$</td>
<td>The set of resource providers.</td>
</tr>
<tr>
<td>$J_t(p)$</td>
<td>Set of running jobs at provider $p$ at time $t$.</td>
</tr>
<tr>
<td>$R_p$</td>
<td>Resource types offered by provider $p$.</td>
</tr>
<tr>
<td>$R_j$</td>
<td>The resource profile containing all required resource types for the job execution.</td>
</tr>
<tr>
<td>$C(p, r)$</td>
<td>Capacity of provider $p$ for resource type $r$.</td>
</tr>
<tr>
<td>$u_j(r)$</td>
<td>Resource demand of job $j$ for resource type $r$.</td>
</tr>
<tr>
<td>$U(p, r)$</td>
<td>Resource utilization of the provider $p$ for resource type $r$.</td>
</tr>
<tr>
<td>$\pi_{p, r}$</td>
<td>Active predictor for estimating the resource utilization of resource type $r$ of provider $p$.</td>
</tr>
<tr>
<td>$S_x$</td>
<td>Score for predictor $x$.</td>
</tr>
</tbody>
</table>
The first step towards the selection of one of the suitable providers is the estimation of the future resource utilizations of all suitable providers. This is done with a predictor that is a function \( \pi : \mathcal{H} \times t \rightarrow \mathbb{N} \cup \{ \text{undef} \} \). The predictor forecasts the next resource utilization based on the previously collected historical information \( \mathcal{H} \). The predictor returns undef if not enough historical information for a prediction are provided. A history \( \mathcal{H} \) is a vector of \( m \) history items \( \mathcal{H}_t = [h_1, \ldots, h_t] \). The broker agent uses the active predictor \( \pi^* \) to predict the utilisation of each resource type of each suitable provider. Then, the provider’s resource utilisation \( \hat{U}_{t+1}(p,r) \) is forecasted under the assumption that the job, \( j \), is executed at the resource provider (line 5). If the broker agent estimates an over-utilisation of provider \( p \)’s resources, \( p \) is removed from the set of suitable providers (line 6). Providers that are not predictable are marked as unpredictable in line 8.

The provider with the highest margin of free expected capacity over all required resource types is selected for the job execution (ref. line 22). The margin is measured as the ratio of the provider’s residual capacity to the resource demand of the job (ref. line 12). The higher this ratio is, the larger the buffer in case the accuracy of the resource utilisation estimation is low. The broker agent allocates resources at the selected provider and submits the job for execution. This case reflects the exploitation of resource providers. In case there is no such provider, the algorithm explores one of the suitable providers from \( P_{\text{undef}} \) (line 17). The worst case is that no free resources at any of the suitable providers are expected. In this case, an allocation of resources has to be dismissed and an error is returned to the user (line 19).

Stage 3: Job execution. The resource broker agent monitors the resource provider during the job execution.

Stage 4: The clean-up includes the rating of the resource utilisation predictions, selecting a new active predictor and storing the monitored resource utilisation information locally. After completion, the resource broker is ready to process the next resource allocation request.

3.1 Resource Utilisation Predictions

Future resource utilisations are estimated with the active predictor of for each resource type. The difficulty in estimating future resource utilisation is that, in fact, they depend on the allocation decisions of other agents in the environment. Instead of relying on the prediction of only one predictor, each resource broker agent maintains a set of \( \kappa \) predictors \( \Pi(p,r) := \{ \pi_1, \ldots, \pi_\kappa \} \) for each resource type of each provider. We call this a predictor set. The individual predictor performance is measured over a range of predictions with a score \( S_\pi \). It is updated for all predictors of a predictor set in the clean-up stage. After all predictor scores are updated, the new active predictor of the predictor...
The score is based on the virtual impact of the predictor’s forecast. This is the score that the predictor would receive if it would have been the active predictor. The new predictor score is $S'_n = S_n + s$, where $s$ is the score for the last prediction, which can be: $+1$ for a correct prediction, $0$ if a prediction is not possible, $-1$ as a predictor score decay, or $-2$ for a wrong prediction.

The predictor score decay is applied if a job allocation request is dismissed, caused by the estimation that no resource for the job execution are expected (Alg. 2, line 19). As a consequence, the broker agent does not monitor any resource utilisation and no predictor feedbacks are possible. Even though this decision was a good one (which is unknown), the resource provider information is never updated, which results in the situation that the broker agent repeatedly expects resource over-utilisation. This situation is prevented by applying a predictor score decay for all predictors that forecast resource over-utilisation. Other predictors forecasting free resources get a higher chance to become the new active predictor. This leads to a balance between the failure of job allocations due to a lack of resources and the exploration of free resources at providers with the risk of resource over-utilisation.

The selection of the new active predictor from a predictor set is non-deterministic. This is to enable coordination between resource broker agents. The allocation decisions of all resource broker agents depend on each other. Hence, if all broker agents believe that provider $p$ has the biggest margin of free resources, all broker agents make the decision to allocate their job at this provider. This may cause resource over-utilisation, which invalidate the beliefs (that free resource are available) of the broker agents. The invalidation of the agents’ beliefs can be prevented if their expectations regarding free provider resources differ [1]. The non-deterministic selection of the active predictors introduces a diversity in the beliefs of the agents and prevents this situation. A roulette wheel selection based on the predictor scores is used to select the new active predictor of each set.

### 3.2 Predictors

The selection of predictors influence the coordination between broker agents. For our experiments, we employ focal forecasting models as suggested by Arthur [1]: simple predictors that are obvious and easily dealt with. Each predictor set comprised 10 randomly selected predictors from a large set of predefined predictors of the following types with different parameters of $k, \omega$:

- **$k$-cycle predictor**: $\pi_k(H) = h_k$ uses the $k^{th}$-last history value
- **$k$-mean predictor**: $\pi_k(H) = \frac{1}{k}\sum_{i=1}^{k}\omega_i h_i$ uses a weighted average of the $k$-latest history values
- **$d$-capacity predictor**: $\pi_d() = d \cdot U(p, r)$ uses a fraction of the provider’s capacity of this resource
- **$k$-mirror predictor**: $\pi_k(H) = 2 \cdot \frac{h_k}{H} - h_k$ uses the mirror image around a weighted average of the complete history of the $k^{th}$ last history value

### 4 Experimental Evaluation

This section reports on experimental results about the algorithm performance in term of the provider resource utilization from the system perspective. We have performed extensive simulations in a variety of system configurations to analyse the algorithm performance. However, due to space limitations we show only a single experiment. The results are averaged over ten experiments with identical system configuration. This is to show the typical performance of the algorithm and reduce the effect of random elements in the decision making process on the resource allocation outcomes.

#### 4.1 Dynamic Provider Capacities

**Experimental Setup:** The experiment simulates a dynamic environment containing five resource providers offering two types of resources (CPU and Memory) with changing capacities during the experiment. The experiment started with 4 initial heterogeneous resource providers offering a total of 41000 units of CPU resources and 47000 units of memory resources respectively. The resource profiles for all jobs were randomly generated with resource demands in the intervals $u_j(cpu) \in [150, 550]$ and $u_j(memory) \in [50, 775]$. A new job was created and submitted for allocation after the previous one was completed. All jobs required two types of resources for the execution with a job execution time of 1 second.

The capacities of each provider were periodically altered using different periods and amounts (refer Fig. 2). In addition to the varying capacities, new users were introduced during the experiment. The simulation started with an initial number of 70 users. After 100 seconds, one new user was added to the environment every 10 seconds until the final number of 100 users was reached 400 seconds after the start of the experiment.

**Discussion:** The performance of our decentralized algorithm in a dynamic environment is plotted in Figure 2. The good coordination of the resource brokers’ resource allocation and their ability to adapt to the changing provider capacities and the additional resource demands of new users in the open system can clearly be seen. Fig. 2a shows how the jobs’ resource demands constrain the number of simultaneous executed jobs. The resources of all providers are efficiently utilized up to their capacities. The announcement of a new resource provider after 400 seconds rebalances the
job allocations. Some broker agents allocate resources at
the newly available provider $p_4$ to explore if this can im-
prove the quality of the allocations. This reduced the re-
source utilization at other providers who operated at their
limits and led to better overall resource allocations.

5 Conclusions

This paper presented a fully decentralized resource al-
location algorithm for the co-allocation of interrelated re-
sources in real-time. The simulation results in an open and
dynamic environment are very promising and justify further
investigations with real applications in open environments.

Our solution relies on randomisation techniques to pre-
vent deadlocks between agents. This can lead to problems
in domains with many providers. Therefore, we suggest to
limit the number of suitable resource providers in each bro-
er agent’s local resource directory. This prevents broker
agents to explore a large number of providers. The detailed
investigation of that problem is a subject of further research.

References

[1] W. B. Arthur. Inductive Reasoning and Bounded Rational-
ity. American Economic Review (Papers and Proceedings),
www.ibm.com/developerworks/websphere/zone/hipods/,
October 2007. IBM.
Monash University, Melbourne, Australia, May 2002.
cation in the grid with learning agents. Journal of Grid
Adaptive Resource Allocation for Sensor Networks. In
Proceedings of the 2nd USENIX Symposium on Network Sys-
tems Design and Implementation(NSDI ’05), May 2005.
resource management: state of the art and future trends.
Balancing: A Study in Multi-Agent Learning. In Journal of
Agent-based Resource Allocation in a Multi-Server Envi-
roneent. In Proceedings of the Sixth Intl. Joint Conf. on Au-
tonomous Agents and Multiagent Systems (AAMAS 2007),
and future trends, chapter Ten actions when Grid schedul-
ing: the user as a Grid scheduler, pages 15–23. Kluwer
[10] C. S. Yeo and R. Buyya. A taxonomy of market-based re-
source management systems for utility-driven cluster com-