Economic Co-allocation and Advance Reservation of Network and Computational Resources in Grids

Wim Depoorter, Kurt Vanmechelen, and Jan Broeckhove

University of Antwerp, Middelheimlaan 1, BE-2020 Antwerp, Belgium
{wim.depoorter,kurt.vanmechelen}@ua.ac.be

Abstract. The introduction of economic principles allows Resource Management Systems (RMS) to better deal with conflicting user requirements by incorporating user valuations and externalities such as the usage cost of resources into the planning and scheduling logic. This allows economic RMSs to create more value for the participants than traditional system centric RMSs. It is important for an RMS to take the data requirements of an application into account during the planning phase. Traditional RMSs have been presented supporting co-allocation and advance reservation of both network and computational resources. However, to the best of our knowledge no economic RMSs proposed in the literature possesses these capabilities. In this paper we present ENARA, an economic RMS with advance reservation and co-allocation support for both network and computational resources. We will demonstrate that ENARA can significantly increase the user value compared to an online approach.

Keywords: Resource Management, Co-allocation, Advance Reservation, Grids, Grid Economics, Network Aware, Futures Markets.

1 Introduction

In shared computing environments such as grid systems, Resource Management Systems (RMSs) have to deal with conflicting requirements due to the fact that users of such infrastructures only care about their own self-interest when formulating their requests. We believe that, contrary to traditional RMSs and scheduling approaches in grid systems, the use of economic principles enables the creation of more open and sustainable grid markets oriented towards value maximization. These grid markets charge the users of the system according to their requirements and resulting allocations while taking into account those of other users as well. In this article, we propose and evaluate ENARA, an economic network and resource aware RMS that employs a futures market to trade usage rights on co-allocated and reserved computational resources and network paths. It is important to take data transfers into account when planning applications on a grid system because these transfers can take a considerable amount of time, especially when the data set required by an application is large.

We use the GESNET network model to simulate the delays associated with network communication and the transfer of data over the network. We evaluate our RMS in the context of bag-of-task applications with CPU bound jobs and input requirements. A good example is the analysis of data coming from one of the experiments at CERN. All the experiments at CERN produce roughly $15\,PiB$ of data per year. It is clear that the processing even a small part of that data still involves massive input files. We extend our original system model presented by Vanmechelen et al. [1] with data dependencies and network resources.

The system model we use and the broker we developed do not require users to specify a fixed parallelization degree for their applications in contrast to several other existing approaches [2,3,4]. Instead, the ENARA RMS is given the freedom to schedule data transfers and computational workload, as long as it can guarantee that the input files are transferred from storage to the execution site before the computation starts and that the application finishes by a given deadline. In our simulations we explicitly take into account the atomicity of jobs and the limited parallelization degree of an application. We do not require job preemption and migration in the construction of job schedules.

This article is organized as follows. First we take a look at related work in the next section. In section 3 we give an overview of the ENARA system model. We discuss the network (pre)pricing of individual links in section 4. We conclude with an evaluation of our approach. We compare our approach in terms of generated user value to an online network aware scheduling policy.

2 Related Work

There is to the best of our knowledge no other work that combines both economic aware network *and* cpu resource co-allocation and advance reservation. A recurring technique for advance reservation and co-allocation of network resources is the discretization of time to make the scheduling problem more tractable. Depending on the level of granularity, this discretization can induce sizeable internal fragmentation on resources as computational jobs and network transfers typically cannot occupy discretized time slots fully.

Takefusa et al. have proposed an advance reservation-based co-allocation algorithm for distributed computing and network bandwidth [5]. Their online planning approach incorporates both co-allocation and advance reservation, but does not integrate economic principles. The co-allocation problem is solved by discretizing time in laddered time frames and modelling it as a simplified Integer Programming (IP) problem. The MC-T scheme proposed by Stevens et al. also discretizes the time [6] but is less suited for systems to large planning windows due to a limited look-ahead. Work by Dramitinos also proposes a discretized economic advance reservation system [7] for network resources only.

The first two approaches are not economic while the last does not incorporate compute resources. In contrast with our work, all of them make use of discretized time slots.

3 System Model

In this section we give an overview of the elements of our ENARA system. First we introduce the GESNET network layer which is modeled after Lambda Grids. It is this component that provides the necessary features for transferring data and reserving the network paths needed for transporting the input files from their storage locations to the location in the network where the application will be executed. Then we describe how we use *Planning Windows*, how we model *Jobs, Workflows, Data and Requests*, *Consumers* and *Providers*. We conclude this section with a description of the most important entity in our system model, the *ENARA broker*.

3.1 Network

To accurately take into account the transfer times of data and the cost of these transfers on the network in an advance reservation setting, we have developed the GESNET network model with support for advance reservations and network pricing [8]. GESNET is modeled after Lambda Grids [9] and is built on the *Jung2* library which provides a number of standard graph algorithms [10]. Lambda Grids are fiber optic networks that allow the creation of light paths for setting up dynamically allocated point to point links between two sites in the network. There are multiple wavelengths used in each fiber optic link, dividing it in multiple logical channels. A light path is a path reserved on a set of subsequent links in the network and bound to a specific wavelength. In fact, it is a list of subsequent *channels*. A channel cannot be allocated to more than one light path at a time which contrasts with traditional packet switched networks. As such, Lambda Grids can be seen as frequency divided networks with non-intersecting network planes.

Entities participating in the simulation are placed at specific locations in the network. As such, the communication delay between them will be modelled and incorporated in the simulation. Multiple sites may be located on a single location. We denote the set of storage sites in our simulation with $S = \{s_1, \ldots, s_m\}$ and the set of compute sites with $C = \{c_1, \ldots, c_n\}$. The communication delay between two entities is calculated based on the speed of light in a fiber optic cable over the physical link-distances between the two communicating sites.

GESNET also enables the creation of network paths for file transfer, both ad hoc and as reservations for future use. Future network transfers will be planned either as soon or as late as possible, depending on the properties of the reservation request. In order to do this, GESNET will look through all the network planes for the earliest or latest possible reservation for a transfer of a specific data set from node s to c [8].

3.2 Planning Windows

ENARA uses periodic planning phases with a sliding Planning window PW. When the size of PW is 24 hours for example, the broker can plan one day in the future. The Planning Period PP is defined as the time delta between two consecutive planning phases. Each planning phase, the PW will shift forward by the same amount as the PP. The planning phase itself starts before the beginning of the PW. The specific lead time LT to the planning phase is chosen to be big enough to ensure that the planning phase will have ended before PW starts. During the lead time, we explicitly model both communication delay as provided by our network model as well as the algorithmic overhead induced by our own resource management system and given by the wall clock time difference between the beginning and the end of the planning phase. As such, we can guarantee that ENARA not only establishes appropriate allocations, but also that it does so within a realistic time frame as set by LT. We believe that this is a necessary validation of the practicality and feasibility of scheduling algorithms that is often not incorporated in the simulation model directly but evaluated separately. The Hot Window HW is defined as the time interval that will become unavailable for planning after the current planning phase. As such, when nothing is planned in or can be moved to the HW after the planning phase, the capacity in this HW is effectively lost.

Since PW's can be big in comparison to the time delta between two consecutive planning phases, it is not straightforward to determine whether the system is congested. To assess whether the system is in a state of congestion, we define a Congestion Window CW. A Congestion window can be seen as a kind of "leaky bucket" for planning and as such allows the system to accommodate occasional bursts of high activity.

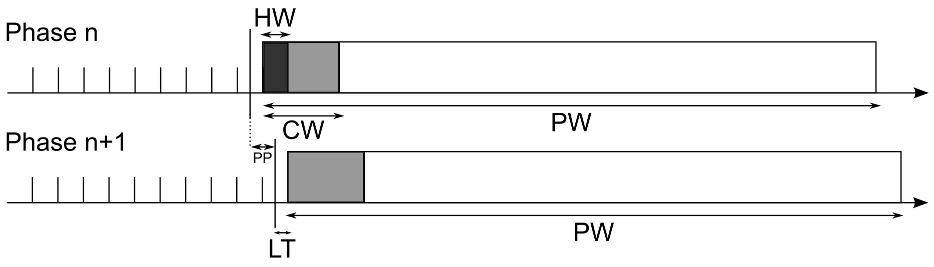


Fig. 1. Planning Windows

All these concepts are illustrated in Figure 1. As can be seen from this figure, the lead time is very small in comparison to the planning window. If we take the scale on the time axis to be 1 hour, PW is $24\,h$ and LT is $30\,min$. We have marked the $1\,h$ long hot window in dark grey. This is the window that the broker should absolutely try to fill as this capacity cannot be sold anymore in a next planning phase. The congestion window is indicated with light grey and also includes the hot window. It is $3\,h$ long. As can be seen, the planning window is shifted forward when starting the next planning phase. The size of this shift is equal to the planning period.

3.3 Jobs, Workflows, Data and Requests

Jobs are modelled as having a certain processing requirement expressed as a normalized processing time *npt* (in hours) when executed on a reference architecture. Applications can be modelled as workflows with a number of individual jobs and precedence relations between them. In this work we model bag-of-task applications and we focus on the effects and benefits of network and CPU resource co-allocation and reservation. Typical examples of such applications are parameter sweeps where individual jobs all process the same input data with different parameter values. Applications may potentially require (parts of) big datasets to process. These are modelled by data dependencies in the application model. When an application has a data dependency, the system needs to make sure that the data is transferred to the location of execution before the computational workflow starts.

To execute an application, a consumer submits an `Application Processing Request` (APR) to the RMS. This APR contains the processing workflow, possible input data requirements and the maximum budget $budget(j)$ the consumer is willing to spend for the execution of its application. A workflow consists of n jobs that have to be executed for the application to finish successfully. The normalized processing requirements of the entire workflow of request j is $npt(j) = \sum_{i=1}^{n} npt(i)$ with $npt(i)$ defined as the processing requirements of job i. When a request requires input data, this will be indicated by the presence of a data dependency in the APR. The location and size of this specific file can be found by querying the `File Catalog`. We will denote the size of the input file of request j with $f(j)$, its size with $ds(j) = ds(f(j))\,GiB$ and the storage location(s) of the data as $S(j) = \{s : f(j) \in s\}$. Requests that require input data will be called Data Dependent APRs (DDAPR) from here on. When a request does not require any input data it is called a Data Free APR (DFAPR). In our experiments we assume that data is not cached at the compute locations after an APR is finished due to insufficient storage capacity at the computational resource providers. This means that data needs to be transferred for every DDAPR.

3.4 Consumers

Consumers are modeled as entities that will submit their requests to the broker for planning. For each request, n jobs of randomized length are generated. If the request is a DDAPR, the file $f(j)$ used will be the same for all subsequent requests of a specific consumer. The budget is calculated based on $npt(j)$, $ds(j)$ and the valuations $nv_{cpu}(j)$ for computational resources and $nv_{net}(j)$ for network transfers of the specific consumer. In addition we randomize the budgets with a variance factor var_{budget}. For each subsequent request of a consumer we multiply the base budget with the random factor $RF = (1{+}rand(-var_{budget}, +var_{budget}))$ with $rand(x, y)$ a uniform random generated value between x and y. The exact formula for the calculation of the request budget is given in Equation 1. We note that $nv_{cpu}(j)$ and $nv_{net}(j)$ are a priori normalized valuations for the application execution and network transfer. These two valuations make it easier to select budget levels for consumers in our experiments and are exclusively used for determining the total request budget and then forgotten. We assume that a real user of the system would only provide an aggregate budget for the execution of its application.

$$budget(j) = (nv_{cpu}(j) * npt(j) + nv_{net}(j) * ds(j)) * RF \tag{1}$$

Before each planning phase starts, all consumers submit their request to the broker. When a request is successfully planned by the broker, the consumer has to pay both network link and compute resource providers for the usage of their resources. When a request of consumer x is successfully finished, it is added to the set of finished APRs F_x. Consumers will have an implicit value $V(j)$ attached to the execution of their application. The total value planned for a consumer is then defined as $V(x) = \sum_{r \in F_x} V(r)$.

3.5 Providers

Each compute provider manages a number of CPUs. We consider CPUs to be uniform parallel machines. This means that all computational jobs can be executed on all CPUs and that their execution time depends linearly on the relative power of the specific CPU compared with a standard CPU.

When the broker wants to schedule the jobs of the workflow of an APR, it will contact a provider, and ask it to schedule either all jobs or as many jobs as possible by means of a `Request Bid`. The provider will search for free periods in the PW of all its CPUs and select the best one for each job by means of a selection policy. In this article, the provider uses a closest to the deadline policy. This ensures that the provider keeps as much free capacity as possible in the beginning of the PW, allowing it to schedule in subsequent applications with shorter deadlines. Since the provider will schedule in jobs as close to deadline as possible, it is very likely that after the planning phase no jobs are planned in the Hot Window (HW) which is defined as the time interval that will become unavailable for planning after the end of the planning phase. As such the provider will defragment all its CPU schedules and attempt to move or shift reservations forward to fill all compute capacity still available in the HW at each specific CPU resource.

3.6 Broker

The ENARA-broker we have developed is capable of scheduling both DDAPRs and DFAPRs. It operates using sliding Planning Windows described in subsection 3.2. In every planning phase, the ENARA-broker will try to plan as many APRs as possible by going through its list of submitted APRs, or *SAPRL*. This list is sorted according to the normalized budget *nb* available to each request. The normalized budget *nb* is a measure for the budget normalized over the expected costs to execute the request's workload and transfer its data. As such it can be used to prioritize APRs from the most to the least valuable. The broker uses a greedy heuristic to try and plan in as much APRs as possible. We use this heuristic approach because our problem domain is NP complete. As such it is not possible to find the optimal allocations in a reasonable time span. Note that in our evaluation, we take both communication delays and overhead of our calculations into account when planning. Therefore, we cannot implement a strategy that may be theoretically correct but not tractable to compute.

Additionally, the ENARA-broker has approximate information of the free capacity of both the network and the compute resources. This allows it to quickly check whether the selected provider and/or the network have enough capacity available just before the planning of an individual APR. As such we have incorporated a fast-fail mechanism in the broker that can reduce both network traffic and computational overhead caused by planning attempts that are bound to fail. We have demonstrated a similar mechanism for APRs without input data [11].

Budget Normalization. In order to normalize APR budgets, the broker uses a `Pre Pricing Algorithm`. For DFAPRs, we can easily calculate their normalized budget as given by Equation 2.

$$nb(j) = budget(j)/npt(j) \tag{2}$$

For DDAPRs however, this is not so straightforward. Their normalized budget depends on the normalized prices that need to be payed for both compute and network resources and on the relative value of these normalized prices. The actual prices are the result of the demand for both network and computational resources and the budgets of individual consumers. However, the actual prices of both computational and network resources are not known until after the planning phase has ended. This means that we have to estimate the expected prices of both computational (np_{cpu}^{E}) and network resources (np_{net}^{E}) before planning the requests.

This is in fact a catch-22 situation since the expected prices in turn will influence the ordering of the APRs, the actual allocations and ultimately the actual prices themselves, which is exactly what we are trying to estimate in the first place. It is therefore important that, when calculating expected prices, we do not create discriminatory prices for individual network paths and computational resources as this would ultimately lead to unbalanced allocations. Note that in each planning phase, we only use the subset of storage sites S that store data requested by the DDAPRs in the $SAPRL$. We do take all compute locations C into account as they have new capacity available in each planning round.

The method of calculating the expected normalized network price np_{net}^{E} and the expected normalized cpu price np_{cpu}^{E} is based on the anticipated congestion of the system for both computational and network resources [8]. It is not discussed here due to space considerations. The distribution of np_{cpu}^{E} over individual links is discussed in section 4. The Gigabyte-to-Workload factor $G2W$ is defined as $np_{net}^{E}/np_{cpu}^{E}$ and is a normalization factor that is used to split the budget of DDAPRs in a separate compute and network budget. The normalized DDAPR budget of request j is defined in Equation 3.

$$nb(j) = \frac{budget(j)}{npt(j) + G2W * ds(j)} \tag{3}$$

The compute budget of a DDAPR is then given by $bc(j) = nb(j) * npt(j)$ and its data budget by $bd(j) = nb(j) * G2W * ds(j)$. This enables the broker to order both DDAPRs and DFAPRs in the submitted APR list ($SAPRL$). The ordering is based on normalized compute budget $nb(j)$ of each request. Since both DDAPRs and DFAPRs are sorted in a single list, we can use a greedy approach for planning.

Planning. The ENARA-broker will iterate over the sorted list of submitted APRs and attempt to schedule in each individual request. For all APRs we define a preferred compute providers list (PPL) which is a sorted list of the providers based on the cost of executing the entire application, including necessary data transfers. All DFAPRs share the same PPL since the location of a compute provider is irrelevant when an application has no input data requirements. For each request we try to plan the individual jobs at the first provider p in the PPL. If p cannot plan all jobs and the request is a DFAPR, the broker tries to plan the remaining jobs with the next provider in the PPL and so on (untill we can plan all jobs). If not all jobs could be planned, we cancel the reservations and remove the DFAPR from the $SAPRL$. If p can plan all jobs of a DDAPR, the broker will attempt to make the necessary network reservations. If it is not successful, all reservations are cancelled and the DDAPR is re-inserted in the $SAPRL$ with an adjusted $nb(j)$. This adjustment is due to the fact that the path price to different network locations *may* be different when the reserve price of a link is higher than the expected price. After planning is finished, the broker will have 2 lists of APRs, the list with planned APRs, $PAPRL$, and the list with unplanned APRs, $UPAPRL$. These may be used in the final pricing phase of the planning process as described in the next subsection.

Pricing. After the broker has planned in all necessary requests, it still needs to price them. For the ENARA broker, we price both compute and network resources with a `Next Highest Losing Bid` strategy (NHLB) as follows. First, we iterate over all requests in the sorted $SAPRL$ and, as long as the current request was planned, we add it to the current set cS. When a request has not been planned, we price all APRs in cS with the normalized budget of the unplanned request and empty cS. We continue iterating over the $SAPRL$ until there are no APRs left. Then we price the remaining request in cS with the reserve price of the providers. The resulting compute price is distributed uniformly over all compute providers that participate in the APR. The distribution for network providers is based on the Extended Minimal Cut List as explained in section 4.2. In this way we become uniform path prices while taking the relative importance of the individual links into account. Note that currently both network and compute resource providers are able to set minimum prices. While it would be possible to dynamically change these during the simulation, we have not experimented with this option.

We use the NHLB pricing strategy because the pricing rule used is closely related to Vickrey auctions. This kind of auction is incentive compatible and the best strategy for a participant is to reveal its true value for the item. For complex systems such as ours, the winner determination is often NP hard. As such it becomes impossible to design a system that on the one hand is incentive compatible, and on the other hand tractable to compute [12]. As such, no hard proof can be given that we have in fact created a system that is incentive compatible. Indeed, there may be situations where participants of the system can in fact gain an advantage by strategic bidding. This should not stop us from applying the most important principles of incentive compatibility to our pricing strategy, namely second price like systems are generally a good choice for participants to reveal their true valuations and are less susceptible to strategic bidding than other pricing systems.

4 Network Pricing

In this section we describe the general approach to pricing individual network links based on their importance. The first subsection deals with the search for links that are responsible for the limitations in transfer capacity between a set of sources S and a set of sinks C in a network. We then explain how the estimated prices translate to the pre-prices of individual resources. More details about the actual algorithms can be found in [8].

4.1 Limiting Links

In this subsection we define the limiting links $LL(S,C)\,\{l_1,\ldots,l_n\}$ from a set of sources S to a set of sinks C with $S \cap C = \emptyset$ as the set of *all* links where the network capacity $cap_{net}(S,C)$ from S to C decreases when *any* of the links in $LL(S,C)$ is removed. We define $LL(S,C)$ more formally in Equation 4 where the extra parameter g denotes the graph representation of the network.

$$\begin{gathered} LL(S,C,g) = \{l_1,\ldots,l_n\} \, \text{with} \, \forall l_i \in LL(S,C,g) : \\ cap_{net}(S,C,g) > cap_{net}(S,C,g \setminus l_i) \end{gathered} \tag{4}$$

Clearly, the links in $LL(S,C,g)$ are important and should be (pre)priced accordingly. We have designed an algorithm that calculates the Extended Minimal Cut List $EMCL(S,C,g)$ between a set of sources S and a set of sinks C on network graph g [8]. The algorithm first calculates the list of minimal cuts between S and C and then extends the individual minimal cuts with the additional trailing limiting links. The links in the $EMCL(S,C,g)$ are equal to the limiting links defined in Equation 4 and its ordering and composition helps in pricing the links between S and C uniformly.

To further clarify these concepts, we provide an example of an extended minimal cut list based on the network depicted in Figure 2. We take $S = \{1,2,3\}$ and $C = \{7,13,16\}$. All limiting links are marked with a diamond. Links marked with open diamonds are extending links of a minimal cut. The resulting extended minimal cut list is given in (5).

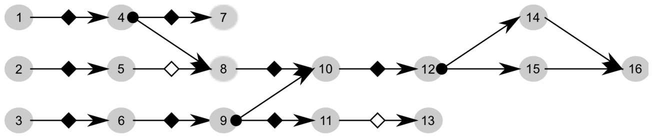


Fig. 2. Extended Minimal Cut List

$$EMCL = \left[\left\{ \begin{matrix} (1,4) \\ (2,5),(5,8) \\ (3,6) \end{matrix} \right\}, \left\{ \left[\left\{ \begin{matrix} (8,10) \\ (6,9) \end{matrix} \right\}, \left\{ \begin{matrix} (10,12) \\ (9,11),(11,13) \end{matrix} \right\} \right] \begin{matrix} (4,7) \\ \\ \\ \end{matrix} \right\} \right] \tag{5}$$

Extended minimal cut lists are encapsulated in square brackets while extended minimal cuts use curly braces. Extended links are not marked but visible by the fact that the links in an extended link are horizontally separated by commas, for example $(2,5),(5,8)$. As can be seen, the second main extended minimal cut contains an extended minimal cut list all by itself. We can also check that when we remove *any* link in $EMCL$ from the network, its capacity will decrease. The non-limiting links from $\{(4,8),(9,10),(12,14),(12,15),(14,16),(15,16)\}$ are not included in the extended minimal cut list.

4.2 Network Resource Pre-pricing

We estimate both the normalized network (np^E_{net}) and computational resource np^E_{cpu} prices based on the expected congestion on network links and computational resources respectively. All compute resources are uniformly pre-priced based on np^E_{cpu}. For network resources this is slightly more complicated as we have estimated the uniform path price. However, we have to distribute this estimation over the individual links. The distribution of the estimated price is based on the Extended Minimal Cut list.

Due to space considerations, we provide the resulting network link price estimations based on Figure 2 and an estimated network price of $np^E_{net} = 0.8\,EUR/GiB$. We represent the resulting link pre-prices on the graph in Figure 3. It is readily verifiable that any path from a source node to a sink node will be pre-priced at the desired $0.8\,EUR/GiB$. The price distribution matches the importance of the individual links which becomes clear when they are matched with the EMCL in (6). We re-use the relative importance of links when distributing the actual path price over the individual links. Note that we have not taken into account the reserve price for the non-limiting links in order to present the distribution of path prices over the limiting links in the extended minimal cut list more clearly.

$$\left[\left\{ \begin{matrix} (1,4)=.4 \\ (2,5),(5,8)=.2 \\ (3,6)=.4 \end{matrix} \right\}_{=.4}, \left\{ \begin{matrix} (4,7)=.4 \\ \left[\left\{ \begin{matrix} (8,10)=.2 \\ (6,9)=.2 \end{matrix} \right\}_{=.2}, \left\{ \begin{matrix} (10,12)=.2 \\ (9,11),(11,13)=.1 \end{matrix} \right\}_{=.2} \right]_{=.4} \end{matrix} \right\}_{=.4} \right] = .8 \tag{6}$$

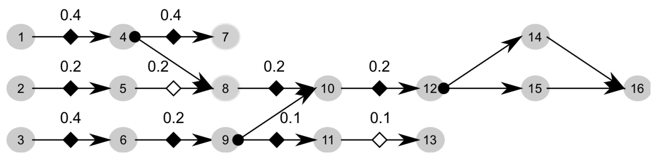


Fig. 3. Extended Minimal Cut List Pricing

5 Results

We now demonstrate with some specific experiments that the different elements of our global approach deliver correct and expected results. We note that variance var of a value v in our experiments is always given as a number in $[0, 1[$ and that for each member of a group, a uniformly random value is chosen in the interval $[v * (1 - var), v * (1 + var)[$ by means of the java `Random nextDouble` method.

For all our experiments we have used a network based on the EGEE topology. We have depicted this network in Figure 4. This figure also contains an example of network prices as calculated by our algorithms. The locations of storage providers are indicated in light grey and the locations of computational resource providers in dark grey. It is readily verifiable that the resulting network path prices are around 0.684 EUR/GiB. Note that we have omitted the reserve prices in order to avoid clutter.

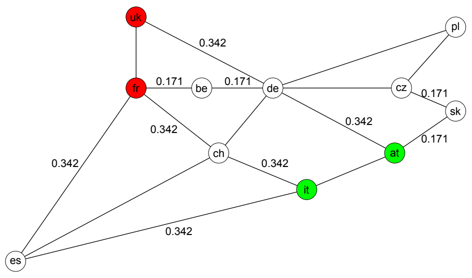


Fig. 4. Simulated Network

5.1 Online vs Offline Network Aware Scheduling

The ENARA broker is an offline scheduling system that periodically plans in all submitted requests in order to generate as much user value and utility as possible. There are however settings where the extra delay caused by this periodic planning is undesirable. As such we have imitated the behaviour of an online FIFO based scheduler that is network aware by randomizing the order in which requests are processed in the planning phase for the ENARA broker. We have named this broker FIFOna. One notable difference with a traditional online system is the fact that applications that cannot be planned immediately will (have to) be resubmitted to the broker. This means that when there is a sudden spike in high-value applications, these applications actually have a higher chance of being scheduled than in a true online FIFO based system.

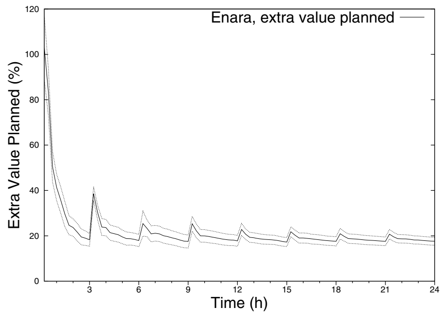
While FIFOna is quite capable of planning in applications with their data dependencies, it significantly changes the assumptions on which the pricing strategy is based. Since we are not selecting applications based on their normalized budget but rather on their arrival time, we are not assigning the allocations to the highest bidders and as such we cannot use a second pricing approach. Another aspect is that it is much more difficult to estimate the gigabyte to workload factor in an online system, especially when load peaks occur. Because of these two reasons, FIFOna uses a reserve pricing strategy. Therefore we focus in the tests that follow on the value that the system has realized for its users, not on the utility because of the unresolved issues surrounding pricing in an online setting.

The network we use is based on EGEE and is depicted in Figure 4. We have distributed compute and storage sites randomly. The random distribution of compute sites S and storage sites C creates situations with potentially different sizes for the minimal cut between S and C and thus with different network capacities. That is why we have grouped the scenarios with identical network capacities. For these tests, we have chosen 25 scenarios where the minimal cut is 3 links. We set LT to $5\,min$, PP to $15\,min$, PW to $24\,hour$ and CW to $30\,min$ and use $NHLB$ pricing for both computational and network resources. We use 6 providers with 100 cpu nodes each and a reserve price of $0.1\,EUR/hour$ for computational resources and $0.1\,EUR/GiB$ for network link capacity. The number of storage nodes in the network is 2. The simulation bound is $24\,h$. The parameters for the 4 different consumer groups can be found in Table 1. The first group, $Group_N$ are users for which the applications do not have any data dependencies. $Group_D$ are users with applications with data dependencies. The last two groups are groups of a limited number of short deadline consumers that will enter the system in $3\,hour$ intervals as indicated by the reload time rt. They also have a tighter deadline as indicated by the deadline factor df. The number of consumers $|Cons|$ of the short deadline consumer groups is limited to 15. The standard job length is 10 or $20\,min$. We have chosen an input data size $ds = 100\,GiB$ as the time it takes to transfer this amount of data is approximately equal to $15\,min$, which simplifies the parameter choice for our experiments.

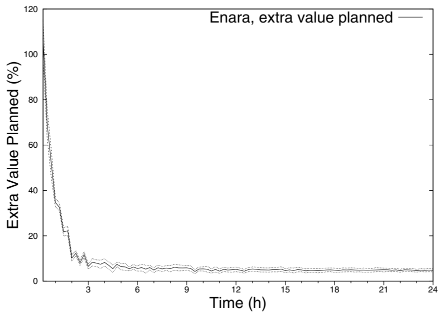
Table 1. Experiment Parameters

Param	$Group_N$	$Group_D$	$Group_{HN}$	$Group_{HD}$
$\|Cons\|$	120	120	15	15
rt	0	0	$3\,hour$	$3\,hour$
$\|jobs\|$	25	10	25	10
jl	20	10	20	10
df	25	12	3	3
np^E_{cpu}	$0.2EUR/hour$	$0.3EUR/hour$	$0.2EUR/hour * \{1, 10\}$	$0.3EUR/hour * \{1, 10\}$
np^E_{net}	n.a.	$0.3EUR/GiB$	n.a.	$0.3EUR/GiB * \{1, 10\}$

The variation in data size is 5% for these tests. However, we have also performed tests with $var_{ds} = 50\%$ without any negative impact on the resulting allocations. Note that we have chosen var_{jl}, var_{ds}, var_{budget} to be 5% and that we have varied the budget of the short deadline consumers by multiplying the budgets of the normal value consumers with $\{1, 10\}$. This results in two experiments; one where the short deadline consumers have similar valuations than the normal consumers, and one where their valuations are 10 times as high. Consumers keep generating new APRs as long as the simulation runs. For all the experiments executed the ENARA broker could achieve a utilization on both network and computational resources over 97.5%.



(a) high value consumers

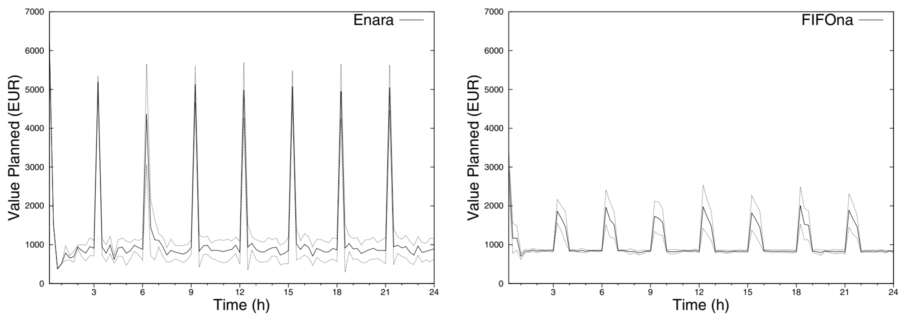


(b) no high value consumers

Fig. 5. Value Planned ENARA vs FIFOna

The relative increase in total value planned for the experiment with high value low deadline groups, is plotted in Figure 5a, together with the deviation. We can observe that offline planning is able to plan just over 17.5% of additional value compared with an online planning approach. In absolute values, the average final consumer value planned for the ENARA system is 118 172 with a deviation of 376 over 25 runs. For the FIFOna system, we obtain an average of 100 477 with a deviation of 1 704 over 25 runs. Note that it would be possible to increase this difference significantly by increasing the number of consumers in groups N and D. Such an enlarged user population would decrease the chances of high-value consumers being planned for the FIFOna broker.

When the low deadline groups have similar valuations than the normal groups, we are in fact testing whether the ENARA system by itself can plan in more value than an online approach. In that case, the difference can only be made by selecting the highest value requests. The results can be seen in Figure 5b. Towards the end of the 24 hour period, the increase in value planned by the ENARA system compared to the online FIFOna planner is approximately 4.7%. This value is very close to the budget variance of 5% and as such a validation of the capacities of the ENARA broker. The average of the final value planned for the ENARA system is 83 800 with a deviation of 455 over 25 runs. For the FIFOna system, we reach an average of 80 012 with a deviation of 233 over 25 runs.



(a) ENARA (b) FIFOna

Fig. 6. Value planned

In Figure 6 we have plotted the value planned in each planning phase and the deviation over 25 runs of both the offline ENARA system and online FIFOna system. Part (a) and (b) show the results of the experiment with high value low deadline consumer groups. We can make the observation that the ENARA broker plans in all high-value applications immediately as shown by the high peaks every 3 hours while for FIFOna the peaks are much less pronounced. Note that the peaks in the FIFOna system are also wider, which means that it can catch up somewhat with the ENARA broker while the deadlines of the remaining high-value requests have not expired.

We can deduce from the previous graphs that ENARA actively selects the highest value applications when scheduling. This is confirmed in our experiments by the sharp decline in the number of requests planned for the normal value consumers every 3 hours and the simultaneous peak in numbers for the high-value consumers. Since the online FIFOna system cannot perform this active selection, it is not capable of extracting all the additional value from the high-value consumers.

6 Conclusion

In this article we have tackled the advance reservation and co-allocation problem of computational and network resources. By estimating np^E_{cpu} and np^E_{net} and defining the $G2W$ factor we are able to flatten an inherently two-dimensional problem and use a greedy heuristic for planning both data dependent and data free APRs. We have clearly demonstrated that the ENARA broker is capable of generating higher value for the users of the system compared to an online approach. This is achieved by actively selecting and planning the highest value requests. We have mentioned that the ENARA broker is capable of planning in both network and computational resources at close to 100% utilization, which is a clear indication of the efficiency of the allocation mechanism.

Acknowledgments. This work is partially supported by the Flemish Fund for Scientific Research (FWO) under research grant 1154609N.

References

1. Vanmechelen, K., Depoorter, W., Broeckhove, J.: Market-based grid resource co-allocation and reservation for applications with hard deadlines. Concurrency and Computation: Practice and Experience 21, 2270–2297 (2009)
2. Chun, B.N., Buonadonna, P., AuYoung, A., Chaki, N., Parkes, D., Shneidman, J., Snoeren, A., Vahdat, A.: Mirage: A microeconomic resource allocation system for sensornet testbeds. In: Proceedings of the Second IEEE Workshop on Embedded Networked Sensors, pp. 19–28. IEEE Computer Society (May 2005)
3. Schnizler, B.: Resource Allocation in the Grid – A Market Engineering Approach. PhD thesis, University of Karlsruhe (2007)
4. Stößer, J., Neumann, D.: GreedEx – A scalable clearing mechanism for utility computing. In: Proceedings of the Networking and Electronic Commerce Research Conference, NAEC 2007 (2007)
5. Takefusa, A., Nakada, H., Kudoh, T., Tanaka, Y.: An Advance Reservation-Based Co-allocation Algorithm for Distributed Computers and Network Bandwidth on QoS-Guaranteed Grids. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2010. LNCS, vol. 6253, pp. 16–34. Springer, Heidelberg (2010)
6. Stevens, T., Leenheer, M.D., Develder, C., Dhoedt, B., Christodoulopoulos, K., Kokkinos, P., Varvarigos, E.: Multi-cost job routing and scheduling in grid networks. Future Gener. Comput. Syst. 25(8), 912–925 (2009)
7. Dramitinos, M., Stamoulis, G., Courcoubetis, C.: An auction mechanism for allocating the bandwidth of networks to their users. Computer Networks: The International Journal of Computer and Telecommunications Networking 51(18), 4979–4996 (2007)
8. Depoorter, W.: Economic and Network Aware Grid Resource Management. PhD thesis, UA (2012)
9. GLIF: Global lambda integrated facility (2012), `http://www.glif.is/` (accessed October 8, 2012)
10. Madadhain, J., Fisher, D., Smyth, P., White, S., Boey, Y.: Analysis and visualization of network data using jung. Journal of Statistical Software 10, 1–35 (2005)
11. Depoorter, W., den Bossche, R.V., Vanmechelen, K., Broeckhove, J.: Evaluating the divisible load assumption in the context of economic grid scheduling with deadline-based qos guarantees. In: IEEE International Symposium on Cluster Computing and the Grid, Los Alamitos, CA, USA, pp. 452–459. IEEE Computer Society (2009)
12. Rothkopf, M.: Thirteen reasons why the Vickrey-Clarke-Groves process is not practical. Operations Research 55(2), 191–197 (2007)