Co-scheduling Data and Task for a Data-driven Distribution of Data-intensive Applications

Mohamed Labidi, Mohamed Jenni
Laboratoire de Recherche en Technologies de l’Information et de la Communication & Génie Electrique (LaTICE)
ENSIT, Tunisia
mohamedlabidi@yahoo.fr
mohamed.jemni@fst.rnu.tn

Maher Khemakhem
Computer Science Department,
Faculty of Computing and Information Technology,
King Abdul-Aziz University, KSA
maher.khemakhem@fsegs.rnu.tn

Abstract— In distributed computing, data scheduling is becoming an important field of research with the emergence of Big Data. High level features provided by software data scheduler often rely on data management policies – possibly user-defined – such as fault tolerance, multi-protocol file transfer, reliable and multi-tenant storage, security and data privacy, locality-aware data distribution etc. Nowadays, to execute data-intensive applications, such advanced features become necessary, and this means that data and task schedulers are capable to cooperate closely. In this paper, we propose a data driven cooperative platform by combining two existing middleware: XtremWeb-HEP, as the task scheduler, and BitDew, as the data scheduler. Taking advantage of both middleware, our solution allows user to select the suitable data scheduling strategy as well as the adequate task granularity which provide the optimal data distribution. To evaluate the efficiency of our approach, we compare different strategies of scheduling tasks and data and prove the efficiency of the cooperation of data and task schedulers to execute data-intensive applications.

Keywords- Co-scheduling data and task · Data intensive application · Data driven.

I. INTRODUCTION

Since the appearance of distributed computing, the task scheduling problem was considered as the main angle of attack to optimize the distribution of compute-intensive applications on various distributed computing infrastructures (desktop grid, grid and cloud). With the generalization of Big Data application, data management has become a main concern to distribute efficiently data-intensive applications. Data management includes several sorts of operations on the data such as storage, distribution, processing that requires specific software. Moreover, the challenge of data-intensive application execution is that moving very large datasets can be prohibitive when considering large-scale infrastructure. Thus, unsurprisingly, several high level data management environment have been developed, such as BitDew [1],Stork [2], MetaCDN [3], that are able to take smart decisions about data placement and data distribution.

This issue has pushed researchers not only to consider data movement when scheduling tasks but also to focus sometimes on the distribution of data first rather than tasks. We call this approach tasks follow data. When using a dedicated software for data management, this approach requires the cooperation between the task and the data schedulers. At the moment very few works have investigated on the intrinsic relationship between independent task and data scheduler in the context of large scale distributed computing.

In this work, we propose a data driven distributed computing platform that uses two independent middleware: XtremWeb-HEP [6] for tasks scheduling and BitDew for data scheduling. This cooperation of middleware enables the two schedulers XtremWeb-HEP (abbreviated as XWHEP) and BitDew to implement a cooperative tasks and data scheduling, which is an innovative approach to the issue of data-driven task scheduling. BitDew is designed to manage large data and enable optimal data placement on distributed infrastructures. It implements scheduling heuristics whose robustness is already tested and proven in [5]. XWHEP is an open source middleware designed for executing large Bag-of-Tasks application on desktop grid, grids and cloud infrastructures.

The objective of this middleware cooperation is to reach a data-driven task scheduler which benefits from the advantages of both middleware, i.e for BitDew: locality-aware data placement strategy, multi-protocol file transfer, data replication and life-cycle management, and for Xtremweb-HEP: tasks scheduling, fault-tolerance, high security, virtualization and on-demand private virtual network. To provide this cooperative scheduling, we are facing several challenges, namely: i) Adaptation of the two middleware so that they can communicate together. Indeed, these middleware are not designed to work together, so inter-scheduler solution has to be designed to allow the cooperation between them. ii) The choice of a proper scheduling policy that meets our needs. iii) The modification of the task scheduler so that it assigns tasks to the nodes
where the data have already been sent by the data scheduler. In this paper, we show that our platform provides an original feature that was not possible before. Indeed, it is possible to select the suitable data scheduling strategy as well as the adequate task granularity which provide the optimal data distribution. We evaluate the performance of our solution by Benchmark: we compare different strategies of scheduling tasks and data by making some experiments which consist in distributing the recognition of printed Arabic script by “Magick”: the OCR application based on Dynamic Time Wrapping (DTW) algorithm for printed Arabic document recognition. This paper is organized as follows: Section 2 introduces BitDew and XWHEP middleware. Section 3 sets forward the architecture of the proposed platform and the communication protocol between XWHEP and BitDew to ensure the proper functioning of this platform. Section 4 presents performance evaluation. Section 5 provides related work. Section 6 concludes and presents some perspectives.

II. BACKGROUND

The basis of our proposal is enabling cooperation between XWHEP and BitDew middleware which were originally developed for desktop grid systems but are especially adapted for hybrid infrastructures: combination of cloud + grid + internet devices [4]. The proposed middleware partnership aims to benefit from the strengths of each middleware, namely the tasks scheduling of XWHEP and the data scheduling of BitDew. Each middleware will enjoy the benefits of the other. Indeed, we will exploit firstly the BitDew’s data placement capacity [5] and optimize consequently data scheduling. On the other hand, we will benefit from XWHEP’s task deployment capacity [6] and therefore get rid of application dependency to data and automate the task scheduling.

In this section we present the characteristics of each middleware and we give an example of a Master/worker application deployment over these middleware.

A. Presentation of BitDew Middleware

BitDew is a middleware designed for the large scale data management and distribution on desktop grid and cloud Systems. BitDew is characterized by its capacity to govern data. Indeed, through BitDew, the user can perfectly dynamically control data operations onto the storage nodes such as placement, distribution, replication and others using metadata called data attributes. Especially, it is possible to govern data placement and apply the appropriate scheduling heuristic which is suitable to a specific application.

Among the data attributes of BitDew, we cite:
- replica: indicates the number of instances of a data that should exist at the same time in the system.
- fault tolerance: defines the resilience of data in the case of machine crash.
- lifetime: specifies the duration after which, a storage host can safely delete a data. This attribute can be absolute or relative to the presence of other data.
- affinity: controls the placement of data according to dependency rules. This is a useful parameter when a data should be assigned to a specific node having another data previously scheduled.
- transfer protocol: specifies the transfer protocol chosen by the user to distribute the data. Indeed, the user can select the suitable transfer protocol according to the number of nodes necessary to distribute the data or the size of these data.
- distrib: defines the number of data with the same Attribute that each node can possess in its queue. It limits the number of data scheduled by the server to the nodes and makes this assignment according to the load of each node. We will use this parameter in our cooperative platform to define the appropriate data scheduling heuristic which ensures the load balancing.

In BitDew, Data distribution and management relies on BitDew services, namely Data Scheduler (DS), Data Catalog (DC), Data Repository (DR), Data Transfer (DT) and Distributed Data Catalog (DDC) (please refer to [1] for more details). It is easy with BitDew to implement a new data transfer protocol for developers due to the open and programmable architecture. For instance, the transfer protocol of amazons3 and dropbox have been recently implemented in BitDew. In particular, BitDew is highly recommended for applications composed of several sets of independent tasks sharing large data. In fact, the movements of these data through the WAN can be costly in terms of performance because the bandwidth of the internet is often limited. BitDew adopts the policy “data driven” enabling a distributed storage of data in the compute nodes with adequate scheduling algorithms to minimize data transfers and therefore improve the application performance.

Several research studies have proven the effectiveness of BitDew. For instance, in [8], BitDew is used to implement a prototype of a hybrid storage system which offers a reliable data storage service by integrating durable storage utilities and idle storage of volatile peer nodes. In this latter work, a set of strategies was evaluated to guarantee availability and durability of data, and the tradeoffs between data reliability and storage efficiency was studied.

B. Presentation of XtremWeb-HEP Middleware

XWHEP is a lightweight grid middleware written in Java language. It is a Free Software (GPL), to explore scientific issues and applications of Desktop Grid Computing (DGC). Desktop grids are intended to aggregate distributed volunteer computing resources (known as worker in XWHEP) and distribute tasks on demand. According to a local activation policy, the worker connects to the centralized server, downloads task (binary code and their associated data) and executes the downloaded binary code. This mode is known as the pull mode. XWHEP project infrastructure may be based on a community of participants. For example, XWHEP allows a High School, a University or a Company to setup and run a DGC for either a specific application or a range of applications. For detailed information about DGC, please refer to the Desktop Grid Computing book [8].
Now, we will describe an original innovative feature that has been recently introduced by XWHEP (since version 8), named “volunteer sharing” and that will be the foundation for our cooperative platform. Thanks to volunteer sharing paradigm, it has become possible to deploy some kinds of specified applications like Hypervisors, Virtual Machines, GPUs and other applications whose deployment was impossible with classic DGC. Indeed, with classic DGC, to deploy an application, user must deploy all necessary environments to run this application, something that is hard and not feasible for the above-mentioned applications. Since the version 8, XWHEP distinguishes between deployable objects and shared objects. A deployable object is an object that must be downloaded by distributed volunteer resources. This was the only object type in previous XWHEP versions. Shared objects are considered as resources and are never downloaded. By introducing the volunteer sharing paradigm, the worker is still working following the pull mode mechanism but can also propose some objects to share: data, applications and library. With volunteer sharing paradigm, a resource in a worker can be used by the whole platform. These shared objects must still be registered on server side. But objects registered as voluntary sharing are not downloaded by the worker since the former is supposed to have a local copy. For example if the shared object is an application, volunteer resources which have this application installed could be selected to compute jobs referring to this application. But volunteer resources which do not have this application installed and/or do not declare it as a sharing will never be selected to run jobs referring to this application. The volunteer sharing paradigm will serve us in our cooperative platform. Indeed, we consider the data deployed by BitDew as a data shared by XWHEP.

Figure 1 shows the architecture of the XWHEP middleware version 8, including volunteer sharing. This is still a three tiers architecture with centralized services that ensure the deployment coherency; distributed clients to use and manage the platform; and workers to aggregate volunteer resources. Figure 1 shows the volunteer sharing paradigm on worker side. This represents any object (data, application, library) the worker wishes to propose to the deployment. If the required conditions are met (mainly the security issues), this volunteer sharing may be proposed to users. It is the responsibility of the owner of any resource to propose or deny volunteer sharing from its resource.

C. Example of Application Deployment

1) Presentation of the OCR Application Magick
Magick is an Arabic OCR application based on Dynamic Time Warping (DTW) algorithm [10]. Mainly for large
vocabularies, studies and experiments have confirmed and shown that the printed Arabic optical character recognition based on DTW algorithm provides an interesting recognition rate [11]. One of the advantages of the DTW algorithm is the recognition process performed using a reference library of isolated characters with an excellent immunity against noise. Moreover, with this algorithm, it’s possible to recognize either cursive or connected characters (sub words or words) without prior segmentation, which is an interesting feature.

In [12, 13], authors proved that the DTW data distribution over a Grid Computing Architecture provides a very interesting results to speedup the DTW execution time and reach scalability.

2) Deployment of Magick over XWHEP middleware

With XWHEP, the user can register its application, deploy it by task submission, follow task states, download results, etc. For instance, to distribute the OCR of Arabic Documents over the deployment of Magick application following Master/Worker Architecture, the user should first register Magick on the server. Thereafter, he has two possibilities to deploy this application. The first one is to register all data of be processed (with the command senddata) on the server which returns the Uniform Resource Identifier (URI) of each data. The user will use this URI in order to reference registered data when submitting the tasks. The second possibility is to prepare tasks as directories, each one contains the program to run and the part of the data to be processed and then submit all tasks without registering data. We adopt the second option because it is more appropriate for a data collection for which URI management is a tedious work for the user. In addition, with this choice we save the register time of data. Once all tasks are already submitted, user can follow up their execution by requesting the server. This latter assigns the tasks to the workers. Once all tasks are executed, user can download results by requesting them from server.

3) Deployment of Magick over BitDew middleware

BitDew follows a data driven architecture. Consequently, data should be firstly placed by the server on worker nodes. To deploy Magick application over BitDew with Master/Worker architecture, user should run two programs on the server node. The first one will start all BitDew services. The second one will register data, create data collection, transfer it, and then schedule data to workers which request it. After that, the user should deploy on workers the worker program which notifies the server of its presence, requests for data, executes Magick application to process this data and return the result to the user. In [5], authors implement in BitDew some scheduling heuristics which have as objective the optimization of data distribution on the computing nodes. We adopted a data driven Master/Worker platform to carry out the deployment of Magick application. These heuristics are implemented and evaluated with both homogeneous and heterogeneous environment. We observed in particular that the optimal data placement on computing nodes is reached by two factors: the sort of data before its assignments to computing nodes and the control of data present in the queue of computing nodes.

In this research work, we benefit from the data placement capacity of BitDew to build the data scheduling part of the cooperative platform. Note that with BitDew we cannot deploy applications only. They should be integrated in the worker Bitdew. This means, with BitDew, deployment concerns data which is related to an application. It is impossible to switch applications on the fly. Also, with XWHEP, it is impossible to deploy data independently of tasks. By combining BitDew and XWHEP, we obtain a middleware which is able to deploy application and data independently.

III. PLATFORM DESIGN

The architecture of our proposed platform follows a master/worker principle for which the server assigns tasks and data to every computing resource (worker node). We separate data from task scheduling by using simultaneously both middleware: BitDew for the scheduling of data and XWHEP for the scheduling of tasks. Figure 2 illustrates the global architecture and the

![Fig. 2 Global Architecture of the Cooperative Platform](image)

We run separately two server programs: the XWHEP server and the BitDew server and we run in each worker node two worker programs: the BitDew worker and the XWHEP worker. Each worker program communicates and retrieves information from its server in pull mode: it asks for data and tasks from the BitDew and the XWHEP server respectively.

In order to implement properly this architecture, we should make correct decisions on some alternative choices. We present in the following these decisions by answering these questions:
1. Since we separated the task scheduling and the data scheduling, how is it possible to match each task to its data?
2. How to organize coordination and communication between workers and servers?
3. Once the task is completed, which component will handle and transfer the result: BitDew or XWHEP?
4. With BitDew, the concept of Data Collection is introduced to handle a large amount of files as a whole, how to make correlation between the data collection in BitDew and the task scheduled by the XWHEP server?

Let's provide answers to these questions and thereafter explain how the communication between BitDew and XWHEP was established. We answer the first question by a "tasks-follow-data" strategy, which is allowed by the "data driven" policy of BitDew. For a such strategy, data are initially submitted to workers by the BitDew server. Thereafter, by matching each data to its corresponding task, XWHEP server will schedule tasks to the appropriate workers. The second question concerns the communication between workers and servers. As we can see in Figure 2, we choose a communication at the worker level rather than at the server level. In fact, this choice allows the BitDew worker to notify the XWHEP worker of the reception of the data so that it can ask its server the proper task. Besides, this inter-workers communication allows us to exploit the capacity of XWHEP to define shared data. Therefore, we do not need to make major changes to XWHEP middleware to be able to communicate with BitDew. Moreover, the communication between workers allows us to achieve our goals without the need to establish communication between servers. This prevents the change of the servers scheduling strategies. The third question is about the choice between the BitDew or the XWHEP approach to handle and transfer the result. BitDew allows a customized transfer of the results which is performed explicitly by the worker, while XWHEP approach requires the result to pass through the server. We choose the BitDew worker to carry out this mission for two reasons: we have implemented a “worker to client direct communication”, by programming the BitDew worker so that it returns the result to the user automatically without the need of his intervention. Conversely, with XWHEP, the user should download the results by himself. Therefore, the transfer of the results requires a considerable time given that they should pass by the server. The second reason is that the XWHEP server removes automatically the data once it handles its results. Consequently, the BitDew worker will be confused when updating its queue since the XWHEP server interferes with the data placement and thus negatively interacts with the BitDew worker. For the fourth question, we relate task and data collection by defining a task in XWHEP which refers to a data collection. This task will be executed on each element of the referred data collection. For example, if a user has a data collection in BitDew, called "DC" that contains 100 files to be processed by the same task, instead of submitting 100 tasks, he simply can submit only one task which refers to the data collection "DC".

IV. PERFORMANCE EVALUATION

In this section and as mentioned previously, the proposed hybrid middleware is evaluated according to a benchmark evaluation.

The corresponding experiments were conducted in 2 clusters of Lyon and Nancy Grid’5000 sites. The network configuration for the experimental environment is illustrated in table 1.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Type</th>
<th>Nodes</th>
<th>CPU</th>
<th>Mem</th>
<th>Site</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sagittaire</td>
<td>Sun Fire</td>
<td>74</td>
<td>AMD 2.4 GHz</td>
<td>2 GB</td>
<td>Lyon</td>
</tr>
<tr>
<td>Graphene</td>
<td>Carri System</td>
<td>144</td>
<td>Intel 2.54 GHz</td>
<td>16 GB</td>
<td>Nancy</td>
</tr>
</tbody>
</table>

In this part we report and detail three experiments; the first one is intended to measure and compare the response times depending on the granularity of task in order to find the best one, we use for this reason the XWHEP task scheduler. The second is to determine the best data scheduling heuristic by using the BitDew data scheduler. The third is to compare different strategies of scheduling task and data by varying both the granularity of task and the data scheduling heuristic. All experiments in this section concern the distribution of the “Magick” OCR application in client-server mode. The binary to run and the documents to be processed are distributed by the server to the worker nodes.

A. Impact of task granularity variation

In this first experiment, we evaluate our system with a focus on task scheduling. Thus, we exploit only the XWHEP component of the cooperative platform without using the BitDew functionalities. Our goal is to determine the best granularity of tasks. For this purpose, we deploy the “Magick” application with different granularity of tasks to show the impact of this variation on the response time. We denote by granularity of task, the number of input documents for each task.

Experiments results

In our experiments, we distribute the OCR of 1000 Arabic documents; the granularity of a task is changed by increasing or decreasing the number of documents to be
processed by the task. Indeed, as has been shown in [5], the response time of the recognition with Magick is a function of the compressed size of the image. In these experiments, we deploy Magick with 4 granularities: 50, 10, 5 and 1 input documents per task. The Granularity 50 documents per task means that the server will assign to the worker 50 documents for each task and the worker will consider these 50 documents as one data to process. Figure 4 illustrates the response time obtained based on the number of workers and the granularity of the tasks. The X axis portrays the number of workers, the Y axis portrays the total recognition time. We denote by “recognition time”, the time required to achieve the recognition of the entire corpus consisting of 1000 images (each image is a document). We show that the response time reduces as the number of workers increases. On the other hand, we gain in response time when the granularity of the tasks is reduced. Such results show the role of the small tasks to reach load balancing and therefore accelerating data processing. In general, the smaller the task is, the fairer is its distribution; consequently workers complete their work nearly simultaneously. However, the adoption of a very small granularity of task results in performance degradation because of the high cost of the download time. Indeed, with the granularity of one image per task, we get 1000 results. The time required to download these results exceed the computation time, hence performance degradation.

B. Data scheduling heuristic impact

In the second experiment, we evaluate our system with a focus on data scheduling. Thus, we exploit only the BitDew component of XtremDew without using the XWHEP functionalities. Our goal is to determine the best data scheduling heuristic. For this purpose, we deploy the “Magick” application with different data scheduling heuristics to determine the best one. We compare three scheduling heuristics implemented by BitDew, namely:

1. Round Robin (RR), for which, the server assign periodically one data to each worker regardless the number of data in its queue.
2. First Come First Serve with Overlap 2 (FCFS-Overlap-2) for which the server send one data to each worker having at least 1 data in its queue.
3. First Come First Serve with Overlap 2 Biggest Data First (FCFS-Overlap-2-BDF) the same as FCFS-Overlap-2, starting by assigning the Biggest Data.

Figure 5 illustrates the response time provided by each heuristic. The X axis represents the number of workers and the Y axis represents the total recognition time. We observe that the Round Robin heuristic provides the worst result, while the best recognition time is provided by the FCFS-Overlap-2-BDF. Such result proves that the governance of data present in the queue of computing nodes as well as the sort of data before its assignment to computing nodes achieve load balancing and improve consequently the performance.

C. Data and Task scheduling

The third experiment considers the cooperative platform with both the task and data scheduler. We consider a collection of 1000 images and we want to determine the best strategies for distributing this data collection and executing the Magic application over this collection. First according to Figure 3, the collection is deployed using the data scheduler and we can choose between three data scheduling strategies as with the previous experiment, namely RR, FCFS, FCFS-Overlap-2-BDF. Second, we can control the granularity of the collection distribution. To this end, we cut the collection in N slices, where each slice contains a fixed number of images. Then, we create one replicated task in XtremWeb-HEP, where the replica number for this unique task is equal to the N, the number
of slices. Of course, this parameter will impact the task granularity execution as with experiment 1.

Table 2 shows the comparison of different strategies of scheduling tasks and data using 64 workers to distribute the OCR of the 1000 images. We vary the number of slices from 1000 to 50 and we compute, for each number of slices, the total response time for each scheduling strategy. We observe that:

- Until 64 slices, the response time decreases as the number of slices decreases. This can be explained by the fact that the decrease of the number of slices implies less communication time and improves consequently the overall response time. Indeed, with 64 slices we obtain the best response time since we use 64 workers which means no communication times.

- The FCFS strategy outperforms the FCFS-Overlap-2-BDF and the RR strategies with 64 workers.

This experience shows that varying both data scheduling strategy and collection distribution granularity allows us to reach the best response time depending on the number of workers used.

Table 2. Evaluation of the cooperative platform according to different data scheduling strategies and varying collection distribution granularity

<table>
<thead>
<tr>
<th>Scheduling Heuristics</th>
<th>#Collection Slices</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFS-Overlap-2-BDF</td>
<td>1000 250 200 100 64 50</td>
</tr>
<tr>
<td>RR</td>
<td>818 404 385 377 228 294</td>
</tr>
<tr>
<td>FCFS</td>
<td>852 444 489 554 227 298</td>
</tr>
</tbody>
</table>

V. RELATED WORK

Cyber-infrastructures imply transferring large volumes of data between different sites; some data movements are explicitly carried out by scientific workflows. In this case, raw data are transferred from the instruments to the storage infrastructure then to the computing infrastructure. Some transfers are performed implicitly by data management systems for replication management and load balancing, for example. Data intensive applications still use some classical protocols like FTP, HTTP, rsync and SCP but mainly for small files. Indeed, such transfer protocols scale poorly and tend to be unreliable when the size of the transferred files exceeds a few gigabytes [14]. Thus, new strategies are required to ensure a reliable and scalable transfer of large files, to deal with errors, to cope with unfriendly network conditions and to use multiple channels in parallel. BitTorrent [15] is a Peer-to-Peer file transfer protocol which proposes to split large files into smaller pieces, called chunks that are easier to handle, in order to address the transfer of large file problem. The integrity of moved chunks is checked using checksums. Only chunks that vary between the source and the target are retrieved. This approach (check summing after splitting) is also adopted by rsync. Given that different files frequently need different transfer protocols because of their varied size, security concerns or the transfer type (one-to-many, one-to-one, etc.), BitDew [1] provides a single interface to various protocols (FTP, HTTP, SCP, BitTorrent). BitDew permits to select, for each file, a different transfer protocol. Stork [2] also provides a single interface for various protocols focusing on data placement; it is also able to choose the appropriate protocol in a particular situation. Gator-Share [16] offers features like BitDew and provides a file system interface that facilitates application development. To raise the burden of utilizing these systems that non-computer scientists would still look at as low-level, various SaaS operated services have emerged. Globus Online [17] is the SaaS counterpart of the Globus toolkit which implements the GridFTP protocol [18], providing a simple user interface for transferring files between remote sites. GridFTP enhances the FTP protocol for secure data transfers between distant sites and parallel high throughput. It eliminates disk contention and part of network by letting files to be moved from diverse sources simultaneously. The partnership between IT components to solve data intensive issues was the subject of several research studies. For instance, in [7], the authors propose to collaborate the distributed storage system iRODS with a grid manager called CiGri. CiGri is a middleware which allows the access to a large number of cores from different clusters and launches parallel jobs on idle processors of these clusters [19]. The cooperation between iRODS and CiGri aims to provide solution to the massive data processing problem faced the seismology project called Whisper. There are two major differences between our work and this one. First, iRODS only provides storage solutions and can’t be used by itself as a middleware which provides computing ability. Whereas BitDew provides both computing and data placement solutions. Second, unlike our solution, iRODS&CiGri is designed to solve a specific issue related to the Whisper project. It is not designed to provide a generic solution which supports different kinds of applications and interests a large set of users.

Romosan et al. [20] propose an architecture for executing co-scheduled data movement and tasks by the cooperation of Condor and Storage Resource Managers (SRMs). Matching of each job to the worker that has the files needed by the job is achieved by including the information about the availability of files on the nodes provided by SRMs into the advertised information used by Condor.

VI. CONCLUSION

In this paper, we cooperate two middleware: XWHEP and BitDew in order to build a platform for co-scheduling of data and task for data intensive application. Our goal was
to benefit from the advantages of the task scheduling of XWHEP and the data scheduling features of BitDew. We proved in particular that the user of the cooperative middleware can select the suitable data scheduling strategy as well as the adequate task granularity which provide the optimal data distribution. In the future, we plan to add consideration of the iterative processing: according to the state of the result, it is possible to re-execute the task by modifying a parameter that influences this result.

REFERENCES


