XtremDew: a platform for cooperative tasks and data schedulers

Mohamed Labidi*

LaTICE,
University of Tunis,
Tunis, Tunisia
Email: mohamedlabidi@yahoo.fr
*Corresponding author

Oleg Lodygensky

IN2P3,
University of Paris XI, France
Email: ol@iex.ec

Gilles Fedak

INRIA,
University of Lyon, France
Email: gilles.fedak@inria.fr

Maher Khemakhem

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

Mohamed Jemni

LaTICE,
University of Tunis,
Tunis, Tunisia
Email: mohamed.jemni@alecso.org.tn

Abstract: With the emergence of big data, data scheduling is becoming an important field of research in distributed computing. Software data scheduler often relies on data management policies that can be defined by the user and provide high level features. Such advanced features become necessary nowadays to execute data intensive applications, and this implies that data and task schedulers should cooperate closely to address the large data processing issue and ensure an optimal distribution of data intensive applications. In this paper, we propose XtremDew, the data and task cooperative scheduler platform. We deal with the distribution of the optical character recognition (OCR) on large scale. We show, in particular, the benefit of the focus on data scheduling to distribute our OCR application. We build the data driven distributing platform by combining two existing middleware: BitDew, as the data scheduler, and XtremWeb-HEP, as the task scheduler. Taking advantage of both middlewares, XtremDew provides new features. To evaluate the efficiency of our approach, we compare different strategies of scheduling tasks and data and we present several scenarios that illustrate the benefits of using XtremDew to execute data-intensive applications.

Keywords: big data; data intensive application; cooperative middleware; big data processing.

Reference to this paper should be made as follows: Labidi, M., Lodygensky, O., Fedak, G., Khemakhem, M. and Jemni, M. (2020) 'XtremDew: a platform for cooperative tasks and data schedulers', *Int. J. High Performance Computing and Networking*, Vol. 16, No. 1, pp.55–66.

Biographical notes: Mohamed Labidi received his Master's degree in Information Systems and New Technologies at the Faculty of Economics and Management from the University of Sfax, Tunisia in 2007. He is currently university assistant in computer science at the Faculty of Medicine of Sfax, Tunisia and PhD student – member of Research Laboratory of Technologies of Information and Communication – Tunisia. His research interests include distributed systems, HPC, IoT in healthcare (connected medical devices).

Oleg Lodygensky is CEO of iExec: Blockchain-based Decentralised Cloud Computing since 2017. Before that, he has been an Engineer at LAL (http://www.lal.in2p3.fr), a High Energy Physics (HEP) Laboratory of University Paris South – Orsay from 2010 to 2012. He has been a partner of European projects DEGISCO and EDGI from 2008 to 2010: partner of the EDGeS European Project (http://www.edges-grid.eu/); leader of the network activity 'NA3: standardisation procedures' since 2006: leader of the XtremWeb-HEP (http://www.xwhep.org) global computing platform based on XtremWeb 1.8 by INRIA. From 2000 to 2006: member of LRI-INRIA (http://www.lri.fr) in a LAL-LRI project: study of the XtremWeb global computing platform usage for HEP applications.

Gilles Fedak is the CEO and co-founder of iExec: Blockchain-based Decentralised Cloud Computing. iExec builds a decentralised market place for computing resources using the Ethereum blockchain. The first version of the product released in November 2017. Before that, he was a permanent INRIA research scientist at ENS-Lyon, France. After receiving his PhD degree from University Paris Sud in 2003, he followed a postdoctoral fellowship at University California San Diego. He produced pioneering software and algorithms in the field of grid and cloud computing that allow people to easily harness large parallel systems consisting of thousands of machines distributed on the Internet. I co-authored more than 80 peer-reviewed scientific papers and won two best paper awards.

Maher Khemakhem received his Master of Science, his PhD and Habilitation accreditation degrees from the University of Paris Sud (Orsay), France respectively in 1984, 1987 and the University of Sfax, Tunisia in 2008. He is currently Full Professor in Computer Science at the Faculty of Computing and Information Technology, King Abdulazis University, KAS. His research interests include distributed systems, HPC, performance analysis, networks security and pattern recognition.

Mohamed Jemni is a Professor of Computer Science and Educational Technologies at the University of Tunis, Tunisia. He is the Director of ICT Department at The Arab League Educational, Cultural and Scientific Organisation (www.alecso.org) from October 2013. He has been the General Director of the Computing Center El Khawarizmi, the internet services provider for the sector of higher education and scientific research in Tunisia, from 2008 to 2013. At ALECSO, he is currently leading several projects related to the promotion of effective use of ICT in education in the Arab world. He produced two patents and published more than 300 papers in international journals, conferences, and books.

1 Introduction

With the advent of distributed computing, several research studies focused on task scheduling to improve the distribution of compute-intensive applications on different distributed computing infrastructures (Alworafi et al., 2019). Data management has become a main concern to support data-intensive applications efficiently following the generalisation of big data applications (Saidala and Devarakonda, 2019). Data management includes several types of operations, such as fault tolerance, multi-protocol file transfer, locality-aware data distribution, reliable and multi-tenant storage, data privacy and security, etc. which necessitate specific software. Furthermore, moving very large datasets can be prohibitive when considering large scale infrastructure. Consequently, obviously, numerous high-level data management environments have been developed, such as Stork (Kosar and Livny, 2004), BitDew (Fedak et al., 2008) and iRODS (Rajasekar et al., 2010), which are capable of making optimal choices regarding data distribution and data placement. This concern has pushed researchers to sometimes give priority to data distribution rather than the distribution of tasks when dealing with large-scale distributed computing. We call this approach '*tasks follow data*'. While using software dedicated for data management, the distribution of big data applications necessitates a cooperation between the data schedulers and the task. To the best of our knowledge, very few studies have addressed the intrinsic relationship between independent data scheduler and task scheduler in the context of large-scale distributed computing.

In this work, we propose an innovative approach to the topic of data-driven task scheduling, which we named *XtremDew*. It is a data-driven distributed computing platform that uses two independent middleware: BitDew, for data scheduling, and XtremWeb-HEP (He et al., 2010) for task scheduling. XtremDew enables both schedulers, XtremWeb-HEP and BitDew, to implement cooperative data and task scheduling. XtremWebHEP is an open source middleware designed for executing large bag-of-tasks applications on grids, volunteer cloud and cloud infrastructures. BitDew has been designed to manage large data and permit optimal data placement on distributed infrastructures. It implements scheduling heuristics whose robustness has already been demonstrated in our previous work.

The objective of XtremDew is to benefit from the advantages of both middleware's features, i.e., for XtremWeb-HEP: fault-tolerance, tasks scheduling, virtualisation and high security, and for BitDew: data

replication, locality-aware data placement strategy, life-cycle management and multi-protocol file transfer. During the implementation phase of our cooperative-scheduling system, we faced a number of challenges, namely:

1 The adaptation of the two middlewares in order that they can communicate together. In fact, these middlewares are not originally designed to cooperate; hence, an inter-scheduler solution had to be devised to permit cooperation between them.

2 The selection of a proper scheduling policy that meets our needs.

3 The adaptation of the task scheduler so that it assigns tasks only to the nodes where the data have already been sent by the data scheduler.

In this paper, we show that by responding to these three challenges, we succeed to develop this middleware cooperation named XtremDew. We show also that XtremDew offers several innovative features that were not possible before, such as:

1 The simplicity of deployment of multiple applications via multiple data collections.

2 The data-aware task placement to accelerate workflow execution: in the case of successive tasks, where the result of the first task is the input of the second task, it is better to avoid the redistribution of the intermediate result.

3 The opportunity for the user to select a subset of distributed computing resources (called workers) for the scheduling of tasks.

We evaluate the performance of our solution by comparing three approaches of distribution of large-scale OCR which are: task-first scheduling using XtremWeb-HEP, data-first scheduling using BitDew, and co-scheduling data and tasks using XtremDew. In addition, we present some scenarios in order to prove the validity of each feature of XtremDew. This paper is organised as follows: in Section 2, we explain the motivations of large-scale OCR and we introduce our OCR application. Section 3 provides the first approach of distributing large-scale OCR, which involves deploying Magick application on XtremWeb-HEP. Section 4 presents the second approach of distributing large-scale OCR with BitDew. Section 5 sets forward the XtremDew architecture and the communication protocol between XtremWeb-HEP and BitDew to ensure the proper functioning of XtremDew. Section 6 presents the evaluation of the performance of XtremDew by use cases, in this section, we presents also the evaluation of the performance of XtremDew by use cases. Section 6 provides related work while Section 7 concludes and presents some perspectives.

2 Large-scale OCR

2.1 Challenge and distribution motivations

In parallel with the evolution of data management, several 'big data' applications have seen an evolution in their implementation motivated by innovation in distributed infrastructures and the diversification of devices producing data flows. Optical character recognition (OCR) on a large scale is an illustrative example of this evolution of implementation. Indeed, several projects are developed during the last decade to address this need in different ways. Starting from the robotic digitisation proposed by the Kirtas project (https://www.kirtas.com/) which allows the automatic digitisation of the books of several libraries to the platform for distributed and cooperative OCR systems proposed by the OCRGrid project (http://www.ocrgrid.org/) then the online service 'Cloud OCR SDK' (https://www.ocrsdk.com/) marketed by ABBYY. All these projects and others show that large scale OCR has become more and more a challenge and consequently, the OCR distribution can be considered as a very interesting solution to this challenge.

2.2 Magick: the OCR application

In this section, we present Magick, the Arabic OCR application based on dynamic time warping (DTW) algorithm (Khemakhem and Belghith, 2005). Studies and experiments, mainly for large vocabularies, have shown and confirmed that the printed Arabic OCR – based on DTW algorithm – delivers high recognition rates (Khemakhem et al., 2007). The recognition process performed using a reference library of isolated characters with an excellent immunity against noise is considered as the main advantage of the DTW algorithm. What is more, this algorithm makes it possible to recognise either cursive or connected characters (sub words or words) without the need for a prior segmentation, which is an interesting feature.

In this work (Khemakhem and Belghith, 2009; Khemakhem et al., 2007), authors proved that DTW data distribution over a grid computing architecture provides very interesting results to speed up the DTW execution time and reach scalability.

3 First approach: task first scheduling for distributed large scale OCR

3.1 Traditional distribution: task first scheduling

In this first approach we deal with a traditional distribution of the Magick OCR application. We mean by traditional distribution, the distribution which focuses on tasks. Indeed, the distribution consists on assigning tasks firstly to different workers to participate to the computation. After that, these workers ask the server for appropriate data to execute the received tasks.

3.2 XtremWeb-HEP middleware

We used XtremWeb-HEP which is a volunteer cloud middleware to explore scientific issues and applications. Volunteer cloud (Costa et al., 2011) are intended to aggregate distributed volunteer computing resources (known as worker in XtremWeb-HEP) and distribute tasks on demand.

XtremWeb-HEP works as follows: after connecting to the server, the worker downloads a task (which consists in a binary code and its associated data). The downloaded binary code is then executed. This mode is referred to as the pull mode. XtremWeb-HEP project infrastructure could be based on a community of participants. For instance, XtremWeb-HEP makes it possible for a university, or a corporation to run a volunteer cloud for either a range of applications or a specific one.

Recently, XtremWeb-HEP has introduced an innovative feature dubbed 'volunteer sharing'. The latter will be the basis for the XtremDew system. It has now become possible, thanks to volunteer sharing paradigm, to deploy some types of particular applications like virtual machines, hypervisors, GPUs and other applications whose deployment was not possible with classic DGC. In fact, with classic DGC, a user has to deploy all required environments to be able to deploy an application, which is very difficult and even impossible for the above-mentioned applications. The latest XtremWeb-HEP's version differentiates between shared objects and deployable objects. In previous XtremWeb-HEP versions, there were no other types of objects apart from the deployable ones. This object should be downloaded by volunteer resources. However, shared objects are never downloaded since they are considered as resources. Indeed, thanks to the volunteer sharing paradigm, the worker can suggest some objects to share such as library, data and applications. Note that with volunteer sharing paradigm, workers keep the pull mode mechanism but if they propose to share some resources, these latter can be utilised by the whole platform. Like deployable objects, all shared objects should be registered on the server side. However, the objects that have been registered as 'voluntary sharing' are not downloaded by the worker because the latter must have already saved a local copy (i.e., of this 'voluntary sharing' object). For instance, if the shared object consists of data, volunteer resources which have this data could be selected to compute jobs referring to this specific data. However, volunteer resources which do not have this data – and/or do not declare it as a 'sharing' – will not be selected to run jobs referring to this data. The volunteer sharing paradigm is primordial in the XtremDew project. In fact, the data deployed by BitDew is considered as shared by XtremWeb-HEP.

3.3 Deployment of magic over XtremWeb-HEP middleware

XtremWeb-HEP middleware distributes tasks and bags of task. But the assignment of data to various tasks must be ensured by the user. In fact, the user starts by preparing the data to be processed by each task so that he can subsequently submit it to the XtremWeb-HEP server which, in turn, assigns it to the workers participating in the distributed recognition.

To deploy any application using XtremWeb-HEP, the user must follow a process which starts by the registration of the application, after that the deployment by task submission, then finally the downloading of the results. In our case, we wish to distribute the OCR of large Arabic documents by deploying Magick application. We follow the master/worker architecture. We should firstly register Magick on the server side. Thereafter, we can choose one of the two possibilities to distribute the large scale OCR. The first one is to send all data to be processed to the server. The latter will register these data and then return the uniform resource identifier (URI) of each data. This URI will be used later by the user, when submitting the tasks, in order to reference registered data. The second possibility is to create all tasks as directories; each one contains the part of the data to be processed and the program to run. After that, we have to submit these tasks without prior registration of data. We have adopted the last option because it is more suitable for data collection since the URI management (i.e., of data collection) is a tedious work for the user. Besides, this choice allows reducing the register-time of data. Once all the tasks have been submitted, the user can follow the steps of their execution by requesting the server. The latter is responsible for the distribution of the tasks among the workers. Once all tasks are executed, the user can download results by requesting them from the server.

In order to ensure load balancing, the corpus of Arabic documents to be recognised is split with the intention that all the tasks have about the same number of documents. However, this equity in distribution is confronted with the heterogeneity of the data sizes. Moreover, this strategy is useful only if the computing powers of the workers are homogeneous.

It is possible to modify the granularity of the tasks by varying the number of documents to be processed by each task. In our experiments, we chose to vary this granularity to look for the optimal distribution that gives the best response time.

We define response time by the time necessary to complete the recognition of the entire corpus (consisting of 1,000 images). We have created tasks which granularities are: 50, 10, 5 and 4 images per task, respectively. In other words, the user can submit 20, 100, 200 or 250 tasks.

Table 1 shows the correspondence between the granularities of tasks and the number of tasks.

Table 1 Correspondence between the granularities of tasks and the number of tasks

Task granularity	*Number of tasks*
50	20
10	100
5	200
4	250

3.4 Performance evaluation

It is well known that according to Amdahl law the speedup factor is given by the flowing equations:

$$S_{latency} = \frac{1}{(1-p)+p/s}$$

where

- $S_{latency}$ is the theoretical speedup of the execution of the whole task
- s is the speedup of the part of the task that benefits from improved system resources
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

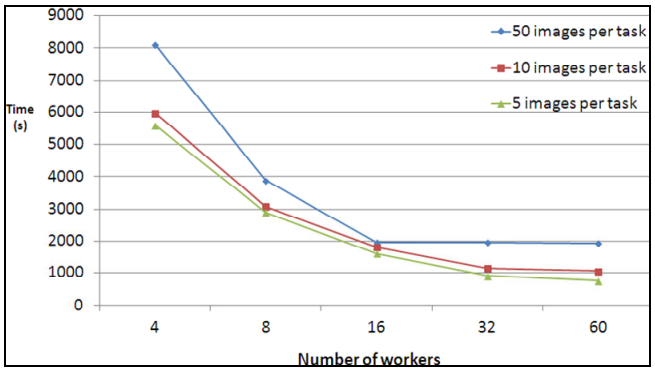
The corresponding experiments were conducted in two clusters of Lyon and Nancy Grid5000 sites. The network configuration for the experimental environment is illustrated in Table 2.

Table 2 Configurations of resources used

Cluster	*Type*	*Nodes*	*CPU*	*Mem.*	*Site*
Sagittaire	Sun Fire	74	AMD 2.4 GHz	2 GB	Lyon
Graphene	Carri System	144	Intel 2.54 Ghz	16 GB	Nancy

Figure 1 shows the variation of the response time depending on the granularity of tasks and the number of workers. The *x*-axis represents the number of workers and the *y*-axis corresponds to the response time.

Figure 1 Effect of task granularity on response time (see online version for colours)



We see a gain in response time that is growing along with the growing number of workers and decreasing with the granularity of tasks.

This gain in response time is explained by the reduction of the amount of work assigned to each worker giving the increasing of the number of workers.

On the other hand, decreasing the granularity of tasks means an increased number of tasks that will accomplish the same amount of work, allowing a more balanced distribution (i.e., of tasks).

We also find that the reduction in response times is more significant with the increase of a small number of workers. This is due to the important reduction in the number of tasks in this case. For example, between 4 and 8 workers, the number of images assigned to each worker is reduced by 125 images. On the other hand, between 32 and 64 workers, the number of images assigned to each worker is reduced by only 15 images.

This approach has the disadvantage of manual preparation of task bags. Indeed, according to the number of tasks to be distributed, the user must choose the granularity of tasks and build the compressed files containing the images to be recognised by each worker. On the other hand, this manual preparation does not take into account the heterogeneity of the images sizes.

On the other hand, with BitDew, as we will see, the assignment of data to the workers is periodic and depends on a scheduling heuristic.

4 Second approach: data first scheduling for distributed large scale OCR

4.1 The data-driven master/slave approach

The data-driven approach differs from the traditional approach in focusing on data rather than tasks. Indeed, with the data-driven approach, the master begins by assigning data, which is parameterised by attributes that dynamically control their distribution over the workers. Once the data needed to perform tasks are available on the workers, these tasks begin to be executed.

At the scheduling level, with the data-driven approach, a first scheduling step is already performed by placing the data on the workers. On the other hand, programmers are not concerned with placing tasks on workers. Instead, they should focus on the distribution of the data to be processed by the tasks when they are assigned. The major advantage of the data-driven approach is that the distribution of data to workers is implicit and dynamic using data attributes. The assignment of data before the tasks is also adopted by MapReduce (Dean and Ghemawat, 2008), the well-known programming model for data intensive application. This model is a Google product for handling massive amount of web search data. With MapReduce, large jobs are broken down into small tasks. Each task is defined by the user as Map or Reduce.

4.2 BitDew middleware

BitDew is a middleware designed for large-scale data management and distribution on desktop grid and cloud systems. BitDew is capable of governing data. In fact, the user can dynamically control data operations (such as distribution, placement, replication, etc.) onto the storage nodes using metadata called data attributes. In particular, data placement can be governed by applying the appropriate scheduling heuristic, i.e., the one suitable for a specific application.

Amongst the data attributes of BitDew, we mention:

- Fault tolerance: Designates the resilience of data in the case of machine break down.
- Replica: Defines the number of instances of a data that must be present at any given time in the system.
- Affinity: Manages data placement in accordance with dependency rules. This is a helpful parameter when a data should be assigned to a specific node with previously scheduled data.
- Lifetime: Specifies the period after which a storage host can safely delete a data. This attribute can be absolute or relative to the presence of other data.
- Transfer protocol: Defines the transfer protocol selected by the user to distribute the data. In fact, the user can choose the appropriate transfer protocol depending on the size of the data or the number of nodes required to distribute these data.
- Distrib.: Identifies the number of data that each node can have in its queue. It limits the number of data scheduled by the server and makes this assignment in accordance with the load of each node. This parameter will be used with XtremDew to define the suitable data scheduling heuristic which guarantees the load balancing.

4.3 *Adapting BitDew: added abilities*

BitDew is intended for large-scale data management. We added task management capability to enable it to deploy the Magick application and other similar applications for intensive data processing. Such applications take advantage of the advanced data management features offered by BitDew.

The task management capability is integrated in the worker and therefore depends on the application to be distributed. Whenever a worker receives a piece of data to be processed, he calls on the responsible application to process this piece of data which is already assigned to him by the server at the beginning of the work. The received data is considered completely processed only after the completion of the execution of the user's application. In this case, the worker can delete the processed data, update his queue to possibly receive another data and call the user's application again

4.4 *Deployment of Magick over BitDew middleware*

Due to the data-driven architecture followed by BitDew, data should be firstly assigned by the server to worker nodes. We run two programs on the server node in order to deploy our OCR application with a master/worker architecture. The first program will start all BitDew services, while the second will schedule data to workers after registering data and creating data collection. After that, we execute the worker program on worker nodes. The latter alert the server of their presence, ask for data, run the OCR application to process these data and finally return the result to the user. In order to optimise data distribution on the computing nodes, we implemented some scheduling heuristics in our previous work (Labidi et al., 2017). These heuristics are evaluated with both homogeneous and heterogeneous environments. We observed in particular that sorting data before their assignment to computing nodes and controlling the number of data present in the queue of each computing node optimise the data placement on computing nodes.

Here, we benefit from the data placement capacity of BitDew to improve the distribution of our OCR application. To achieve this, we deploy the 'Magick' application with different data scheduling heuristics to determine the best one and then compare it with the best result obtained by XtremWeb-HEP. We apply three scheduling heuristics implemented by BitDew, i.e.,

1 Round Robin (RR): This means that the server periodically assigns one data item to each worker without considering the number of data in its queue.

2 First come first serve with overlap 2, (FCFS-overlap-2): Means that the server sends 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, but starting by assigning the biggest data.

Figure 2 Effect of data scheduling on response time (see online version for colours)

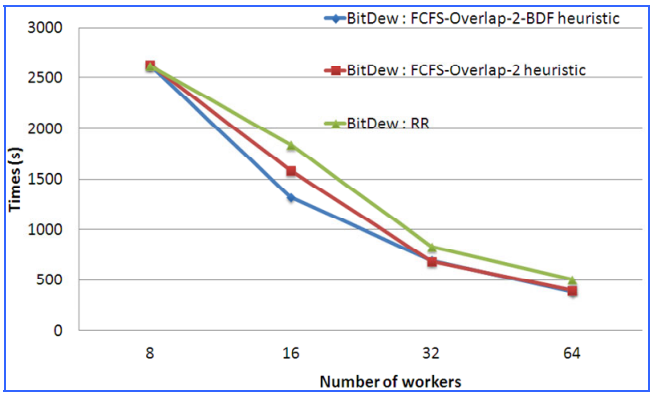
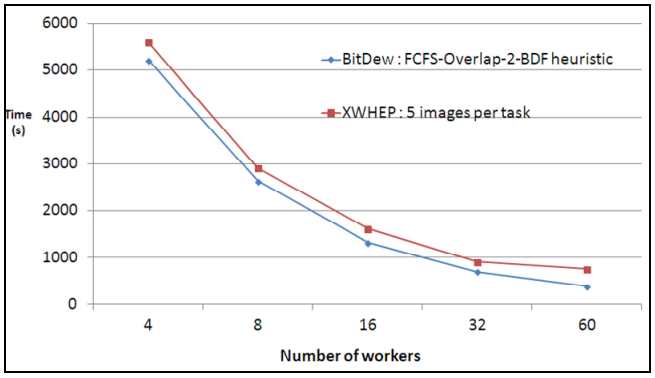


Figure 2 shows the response time that each heuristic provides. It is clear that the worst result is provided by the Round Robin heuristic, while the FCFS-overlap-2-BDF provide the best recognition time. Such result proves that the sorting of data before its assignment to computing nodes, in addition to the governance of data present in the queue of computing nodes, achieves load balancing and, therefore, improves the performance.

In Figure 3 we compare the FCFS-overlap-2-BDF heuristic with the best result obtained by XtremWeb-HEP which is the granularity of five images per tasks. We observe that the FCFS-Overlap-2-BDF scheduling heuristic outperforms the best result obtained by XtremWeb-HEP which proves the benefit of focusing on data distribution

strategies and the data driven distribution of data intensive applications.

Figure 3 BitDew vs. XtremWeb-HEP performance (see online version for colours)



5 Third approach: co-scheduling data and task for distributed large scale OCR

5.1 The motivations for cooperation

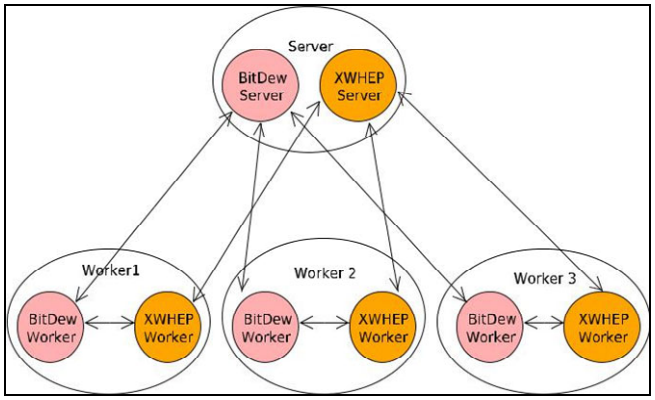
Note that with BitDew, it's impossible to deploy only applications. They must be integrated within the worker Bitdew. Indeed, the deployment over BitDew concerns data associated with a specified application and, consequently, it is impossible to switch applications on the fly. On the other hand, with XtremWeb-HEP, it is impossible to deploy data separately of tasks. By combining XtremWeb-HEP and BitDew, we obtain a middleware which is able to deploy data and applications independently.

This third approach consists on cooperate BitDew and XtremWeb-HEP schedulers in order to benefit from the strength of each one of them, namely the data scheduling of BitDew and the tasks scheduling of XtremWeb-HEP. We call the proposed cooperation system 'XtremDew'. First, we will use BitDew's ability to allocate data (Labidi et al., 2012) which optimises data scheduling. Second, we will benefit from XtremWeb-HEP's task deployment competence (He et al., 2010) and consequently get rid of data-application dependency and automate the scheduling of tasks.

5.2 XtremDew design

XtremDew follows a master/worker architecture for which the server attributes data and tasks to every computing resource (worker node). XtremDew separates data from task scheduling by exploiting at the same time both middleware: XtremWeb-HEP for the scheduling of tasks and BitDew for the scheduling of data. Figure 4 shows the global architecture and the different XtremDew components. We independently run two server programs: the BitDew server and the XtremWeb-HEP server, and execute – in each worker node – two worker programs: the XtremWeb-HEP worker and the BitDew worker. Each worker program communicates and retrieves information from its server in pull mode: it asks for tasks and data from, respectively, the XtremWeb-HEP and the BitDew servers.

Figure 4 Global Architecture of XtremDew (see online version for colours)



In order to properly implement this architecture, we have to make the most appropriate decisions concerning some alternative choices. In the following, we define these decisions by answering a number of questions:

1 Given the fact that data scheduling and task scheduling are separated, how can each data item be matched with its task?

2 How to organise communication and coordination between servers and workers?

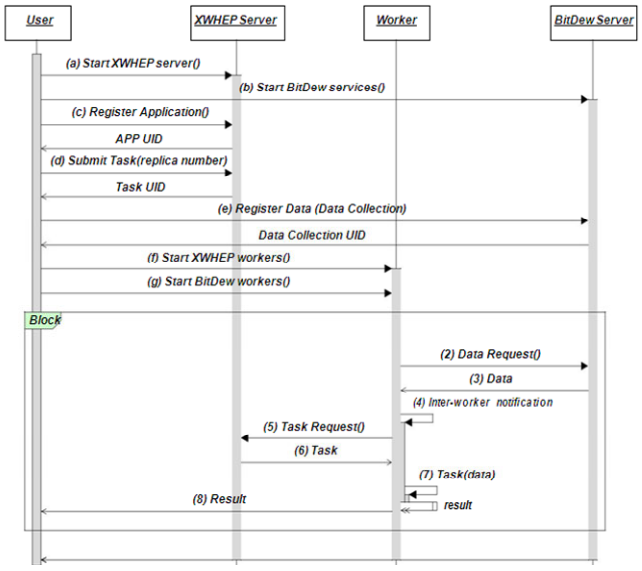
3 Once the task is achieved, which component will manage and transfer the result: XtremWeb-HEP or BitDew?

4 With BitDew, a large amount of files can be handled as a whole via the 'data collection' concept. How to make a correlation between the data collections of BitDew and the tasks scheduled by the XtremWeb-HEP server?

Let's provide solutions to these issues and then give details about the communication between XtremWeb-HEP and BitDew. For the first question, we apply a 'tasks-follow-data' strategy, thanks to the 'data-driven' policy of BitDew. For this strategy, data are firstly assigned to workers by the BitDew server. Afterward, each data item received should be matched with its corresponding task. XtremWeb-HEP server will then submit tasks to the appropriate workers. The second question was about the communication between servers and workers. As we can see in Figure 4, we choose communication to be only at the worker level and we never allow servers to communicate. Indeed, this choice means that the BitDew worker notifies the XtremWeb-HEP worker of the data received so that it can request the proper task from its server. In addition, thanks to this inter-workers communication, we can exploit the capacity of XtremWeb-HEP to define shared data and, consequently, XtremWeb-HEP middleware will be able to cooperate and communicate with BitDew without making major changes. Furthermore, by allowing the workers to communicate, we attain our goal without the need to setup communication between servers. This helps avoid the

change in server scheduling strategies. The third question concerns the choice between the XtremWeb-HEP or the BitDew approach to manage and transfer the result. With BitDew, the results are handled and transferred explicitly by the worker; whereas XtremWeb-HEP approach requires the passage of the result through the server. We adopt the BitDew approach for two reasons: we have programmed the BitDew worker in order to implement a 'worker-to-client direct communication'. Thus, it (i.e., the BitDew worker) returns the result to the user automatically without the need for its intervention. Conversely, with XtremWeb-HEP, the user should download the results by himself. Consequently, the transfer of the results requires a considerable time since they should go through the server. The second reason is that the XtremWeb-HEP server automatically removes the data once its results have been transferred. Consequently, the BitDew worker will be confused when updating its queue since the XtremWeb-HEP server interferes with the data placement and, thus, negatively cooperates with the BitDew worker. For the fourth question, we match data collection with a task by defining a task in XtremWeb-HEP which refers to a specific data collection. This task will be executed on each element of the referred data collection. For instance, if a user has a data collection in BitDew, called 'DataC1' that contains 50 files to be processed by the same task, rather than submitting 50 tasks, he can simply submit only one task which refers to the data collection 'DataC1'.

Figure 5 BitDew-XtremWeb-HEP collaboration (see online version for colours)



The sequence of interactions between the user and the different components of XtremDew to deploy an application is illustrated in Figure 5. It shows the exchange of information between the user and the XtremWeb-HEP server, which is responsible for the task scheduling, the BitDew server that is responsible for the data scheduling and one worker node (in which we run, obviously, two programs: Bitdew worker and XtremWeb-HEP worker).

The steps of deployment are as follows:

1 The user should:
   a start the XtremWeb-HEP server
   b start the BitDew server
   c register its application on the XtremWeb-HEP server
   d submit a replicated task to XTREMWEBHEP server
   e register the data collection to be processed on the BitDew server
   f start the XtremWeb-HEP worker program on workers
   g start the BitDew worker program on workers.
2 BitDew workers connect to BitDew server and ask for data.
3 The BitDew server assigns the data to be processed to the available workers, according to the scheduling heuristic chosen by the user.
4 In each worker node, the BitDew worker notifies the worker XtremWeb-HEP of the presence of the data received by the server.
5 The XtremWeb-HEP worker considers the received data as a shared data and informs its server that it is ready to run a task which corresponds to this data.
6 The XtremWeb-HEP server assigns to this worker a task that corresponds to this particular data.
7 The XtremWeb-HEP worker runs the task, produces the result and notifies its server that the task is completed.
8 The BitDew worker detects the presence of the result, transfers it to the user, deletes the processed data, updates its queue, and asks the BitDew server for a new data.
9 If the BitDew server still has data to schedule, go to step 3, else, end of work.

The worker BitDew must prepare each data received from its server to be processed by the worker XtremWeb-HEP. This preparation consists in the decompression and the copy into a new folder. This folder is used by the worker XtremWeb-HEP to retrieve the data and process it. After that it is used by the BitDew worker to retrieve the result of the task and transfer it to the user. Once the data is prepared, the worker BitDew informs the worker XtremWeb-HEP of the presence of the data. So we need an inter-worker notification in both directions:

1 BitDew worker must notify XtremWeb-HEP worker of all data received so as to be able to ask the server XtremWeb-HEP the proper task.
2 BitDew worker must be informed when XtremWeb-HEP worker completes the execution of the task so that it updates its queue and requests new data.

Given that XtremWeb-HEP worker has already a socket on which he is listening; we choose to implement the first communication by socket. Indeed, BitDew worker opens the socket in which it indicates the path of the data received as well as its data collection name. The latter will be used by the XtremWeb-HEP server as DataPackageName of a shared data to select the appropriate task. This communication serves to notify the XtremWeb-HEP worker each time a data is added to the queue of the worker BitDew or deleted from it. The second communication was not implemented by socket since the BitDew worker can know that the task is completed by detecting the presence of the result. Hence this communication was simplified by an implicit notification.

Furthermore, we implement the correlation between task and data collection by adding a new parameter to the task description of XtremWeb-HEP named DataPackageName which will have the same value of the Data Collection name. Using this DataPackageName, each task can reference a data received by the BitDew worker. Moreover, we enrich the XtremWeb-HEP task's parameter by adding a new feature facilitating the work of the user which is the possibility to submit a replicated task. Indeed, instead of submitting the same task several times, the user can use a parameter that specifies the number of times this task should be scheduled and then submit this task for only once. This feature automates job submission.

For instance this command consists of submitting one task named Magick, replicated for 120 times. This task refers to a shared data to which the attributed package name is 'text'.

xwsubmit magick 1 – xwreplica 120 – xwpackage text

5.3 Evaluation by scenario

We explain in this section the benefits of XtremDew in terms of new features provided (and which were not possible with BitDew or XtremWeb-HEP alone).

Combining the best elements of these two systems allow us to define more functionality and overcome the difficulties associated with:

1 making easy the deployment of multi-application and multi-dataset

2 speedup of workflow applications

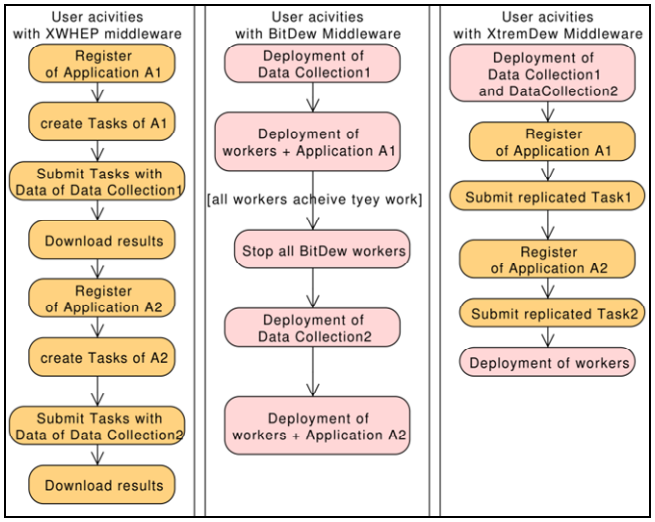
3 limiting the task scheduling to a subset of workers.

In the rest of this paper we will explain each of these features by giving some use cases.

5.3.1 Ease of deployment of multiple applications and multiple data collections

This is one of the advantages of XtremDew compared to BitDew and XtremWeb-HEP middlewares. To illustrate this first added value, we will consider the case of the deployment of two applications A1 and A2, processing respectively data collection 1 and data collection 2. Figure 6 illustrates the activities performed by the user to deploy A1 and A2 as well as their data collections over each middleware: XtremWeb-HEP, BitDew and XtremDew. The yellow colour presents activities relative to XtremWeb-HEP services and the pink colour presents activities relative to BitDew services. with XtremWeb-HEP middleware, executing application A1 on data collection 1, consist in preparing and submitting a task for each element of the data collection. Therefore, if data collection 1 has n elements, n tasks are going to be submitted and n results are going to be downloaded. The creation of tasks, their submissions as well as the download of the results are costly in terms of performance and are avoided with XtremDew middleware.

Figure 6 User activities to deploy tow applications and tow data collections with XtremWeb-HEP, BitDew, and WtremDew (see online version for colours)



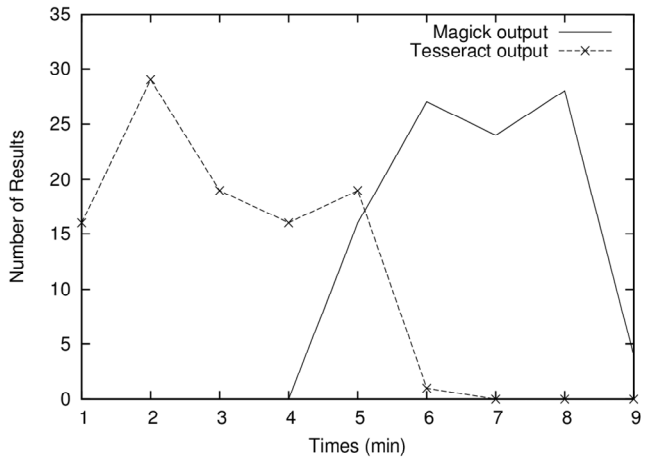
As with any data management system, BitDew middleware has no application catalogue. In other words, the worker program of BitDew is developed according to the application, and the deployment of workers (distribution, compilation and execution of BitDew worker program) depends on the application to run on these workers. Hence, to insert a new application, the user must deploy again all workers. For example, to deploy A2, the user should wait until the achievement of all the tasks of A1 and then stop all workers. It is impossible to change the application on the fly without stopping workers program. Conversely, with XtremDew, thanks to the separation of two servers: tasks and data, the deployment of workers no longer depends on the application to run by these workers. Consequently, the user does not need to make a new deployment of BitDew workers for each data collection. He needs to deploy only for one time the workers, distribute both data collections by the same deployment and submit only one replicated task that references this data collection.

The worker can, permanently, receive from its server different data from different collections to be processed by different applications also.

Use case

This feature is useful when the user has multiple data collections, each one has to be processed by a separate application. In the field of recognition, the user may have various types of scanned documents from different languages. Each type of these documents should be treated by the adequate OCR application. To test this use case, we built a second collection of English documents and we chose Tesseract application (Smith, 2007) to ensure their distributed recognition.

Figure 7 Number of results of Magic and Tesseract applications



To distribute the recognition of Arabic and English documents over XtremDew middleware, we submit both data collections to the BitDew server. Thereafter, we submit two replicated tasks to the XtremWeb-HEP server; each one corresponds to one data collection. Each BitDew worker will receive permanently data from its server and regardless of the collection to which the received data belong, it must always handle its queue in term of number of data according to the scheduling heuristic adopted. Figure 7 shows the number of results returned by workers each minute, giving that we distribute 100 documents of each data collection. We observe, in particular, that workers begin by performing tasks related to the Tesseract application. This is explained by the fact that the server BitDew has begun distributing the collection of English documents. We also note that between 4 and 7 minutes, we get results from both applications which proves that workers continue running all tasks without interruption. Indeed, some nodes start to process Arabic documents, while others still process English documents.

5.3.2 Speedup of workflow applications

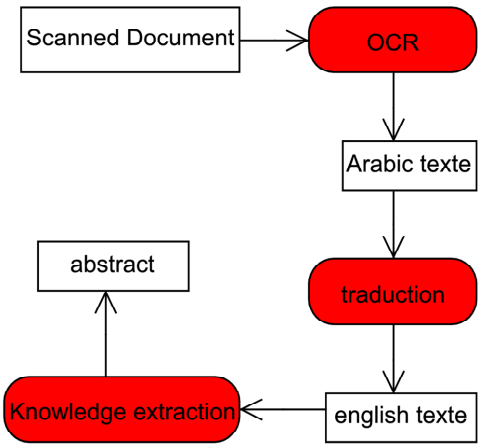
Reducing computing time by optimising resources occupation is the focus of many recent works dealing with big data (Wu et al., 2016a, 2016b; Jeba et al., 2019; others).

In this section, we aim to reduce computing time by the speedup of workflow big data applications. Workflow applications correspond to the succession of tasks. It means to treat by a second task the result of the first one on the same worker and without repeating the data distribution. By using only BitDew or XtremWeb-HEP, the deployment of a second task that uses as input the result of the first one, requires the collection of the results of the first task and the distribution of these results as data to be processed by the second task. However, with XtremDew, through communication between XtremWeb-HEP and Bitdew workers, it is possible to consider the output of a task as an input of another that runs on the same worker, without having to repeat the distribution. Indeed, the BitDew worker detects the presence of the result of a task being executed by the XtremWeb-HEP worker. If the user needs to treat this result by a second application, the BitDew worker can consider this result as data already assigned by its server and notifies the XtremWeb-HEP worker to treat it through a proper task. This possibility remarkably facilitates the work of the user when he needs to perform workflow.

Use case

In the same field of recognition, the user may need to extract knowledge from a scanned document. In this case, he should deploy, for instance, an application of automatic summary which treats as input the document recognised by the OCR application. Another need can appear, i.e., the translation of the recognised text to another language, assuming that the user ignores the language of the first document. In some cases, the user may need to run both tasks after the recognition of a document: the translation followed by the extraction of knowledge. Figure 8 illustrates the task execution order and the data flow between tasks. It is thanks to XtremDew that a worker executes this sequence of tasks by only one deployment of data. The user must submit three replicated tasks: a first one for recognition, a second one for translation and a third one for knowledge extraction. The distribution of data is done only once: before starting the execution of the first task. The worker BitDew notifies the worker XtremWeb-HEP three times to ask for three tasks: first to request a recognition task; second, when the first task is successfully completed, to request a translation task; finally, after translation, to request a task of knowledge extraction.

Figure 8 Sequence of three tasks executed by the same worker (see online version for colours)

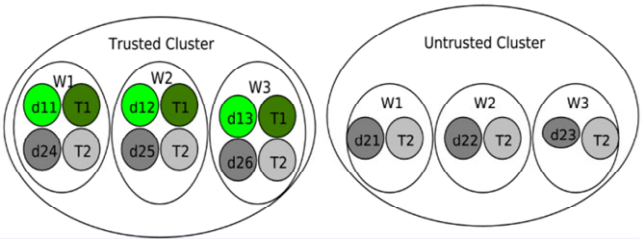


5.3.3 Limiting the task scheduling to a subset of workers

With the enormous and rapid growth of connected objects and sensors which continue to generate data (Atat et al., 2018; Wu et al., 2018), optimising the storage of this data is becoming more and more highly recommended.

With XtremDew, thanks to the 'data-driven' approach adopted by BitDew, we follow a 'tasks-follow-data' strategy for which data are initially assigned to selected workers and, thereafter, appropriate tasks are scheduled to these workers. XtremDew ensures that each task is scheduled to the appropriate worker to be properly matched to the adequate data. This strategy allows the user to limit the task scheduling to a subset of workers by limiting the number of workers selected to receive data. This capacity of selecting compute nodes in task scheduling is useful for some applications when the computing cannot be made by all the workers for confidentiality reasons. For instance, we consider two data collections: DC1, composed of d11, d12, …, d1n, and DC2, composed of d21, d22, …, d2n, to be processed respectively by two tasks: T1 and T2. If the results provided by T1 are confidential, it is possible, as it is illustrated by Figure 9, to deploy DC1 only on trusted workers. Whence T1 will be scheduled to these trusted workers and their results will be consequently protected from any malicious intruder.

Figure 9 Limiting the task scheduling to a subset of workers (see online version for colours)



6 Related work

The partnership between IT components to solve data intensive issues was the subject of several research studies. For instance, iRODS and the Middleware CiGri for the Whisper project (2019) 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 (CiGri Middleware, 2019). 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 cannot be used by itself as a middleware which provides computing ability. Whereas BitDew provides both computing and data placement solutions. Second, unlike XtremDew, 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. (2005) propose 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.

While several algorithms are compared in this work, authors only perform simulations, and performance is not verified on real systems as we have done in our work. Furthermore, our approach proves that cooperation between middleware enables new features which is not the case of this discussed work.

Deng et al. (2013) propose a data and task co-scheduling strategy that group the mostly related datasets and tasks. This approach consists on placing application datasets across distributed data centres and schedule tasks according to the data layout in order to reduce latency and makespan for workflow execution.

Makatun et al. (2015) propose a constraint programming-based planner that schedules data and computational jobs in a distributed environment with the aim of optimising resource utilisation and reducing the processing completion time. The optimisation is achieved by ensuring that none of the resources (CPUs, data storages and network links) are oversaturated and that the jobs are scheduled where the data is already present or the data is pre-placed at the site where the job runs.

Despite the similarities between our work and the two aforementioned studies, namely in the data and task co-scheduling strategy, XtremDew has the advantage of taking benefit from high-level data management environment like BitDew in data scheduling and providing new features, such the speedup of workflow applications and the easy deployment of multiple applications and multiple data collections.

7 Conclusions

In this paper, we have ascertained and shown the added values of XtremDew for executing data-intensive applications. Indeed, in the case study we have considered, we have proposed and tested three scheduling approaches to distribute large scale OCR. The first one namely the task first scheduling using XtrmWeb-HEP middleware, the data first scheduling using BitDew middleware and the data driven co-scheduling data and task using XtremDew which cooperates both: XtremWeb-HEP and BitDew. We proved in particular the importance of data-driven architecture and introduce the 'task follow data' strategy by focusing on data scheduling to improve the performance of large scale OCR.

Our goal was to benefit from the advantages of the task scheduling of XtremWeb-HEP and the data scheduling features of BitDew. We proved in particular that the user of

XtremDew can select the suitable data scheduling strategy as well as the adequate task granularity which provide the optimal data distribution. In addition, due to the independence of data and task scheduling, XtremDew provides an easy deployment of multiple applications with multiple data collections, a possibility for the user to select only a subset of resources for task execution and a data optimisation for workflow execution. 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. In addition, in the context of hybrid cloud, we can plan a workflow for which some of the processing is done locally and the rest is done in the cloud in order to execute stream applications and to follow the green revolution.

References

Alworafi, M.A., Dhari, A., El-Booz, S.A. and Mallappa, S. (2019) 'Budget-aware task scheduling technique for efficient management of cloud resources', *IJHPCN*, Vol. 14, No. 4, pp.453–465.

Atat, R., Liu, L., Wu, J., Li, G., Ye, C. and Yi, Y. (2018) '*Big Data Meet Cyber-Physical Systems: A Panoramic Survey*, DOI: CoRRabs/1810.12399.

CiGri Middleware (2019) [online] http://ciment.ujf-grenoble.fr/cigri/dokuwiki (accessed 10 April 2019).

Costa, F., Silva, L.M. and Dahlin, M. (2011) 'Volunteer cloud computing: MapReduce over the internet', in *IPDPS Workshops*, IEEE, pp.1855–1862.

Dean, J. and Ghemawat, S. (2008) 'MapReduce: simplified data processing on large clusters', *Communications of the ACM*, Vol. 51, No. 1, pp.107–113.

Deng, K., Ren, K., Song, J., Yuan, D., Xiang, Y. and Chen, J. (2013) 'A clustering based coscheduling strategy for efficient scientific workflow execution in cloud computing', *Concurrency and Computation: Practice and Experience*, Vol. 25, No. 18, pp.2523–2539.

Fedak, G., He, H. and Cappello, F. (2008) 'BitDew: a programmable environment for large-scale data management and distribution', in *SC*, IEEE/ACM, p.45.

He, H. Fedak, G., Kacsuk, P., Farkas, Z., Balaton, Z., Lodygensky, O., Urbah, E., Caillat, G., Araujo, F. and Emmen, A. (2010) 'Extending the EGEE grid with XtremWeb-HEP desktop grids', in *CCGRID*, IEEE Computer Society, pp.685–690.

iRODS and the Middleware CiGri for the Whisper project (2019) [online] https://irods.org/uploads/2014/06/BriandBzeznikIrodsUserMeeting2014.pdf (accessed 13 April 2019).

Jeba, J.A., Roy, S., Rashid, M.O., Atik, S.T. and Whaiduzzaman, M. (2019) 'Towards green cloud computing an algorithmic approach for energy minimization in cloud data centers', *IJCAC*, Vol. 9, No. 1, pp.59–81.

Khemakhem, M. and Belghith, A. (2005) 'A multipurpose multi-agent system based on a loosely coupled architecture to speedup the DTW algorithm for Arabic printed cursive OCR', in *AICCSA*, IEEE Computer Society, p.121.

Khemakhem, M. and Belghith, A. (2009) 'Towards a distributed Arabic OCR based on the DTW algorithm: performance analysis', *International Arab Journal of Information Technology*, Vol. 6, No. 2, pp.153–161.

Khemakhem, M., Belghith, A. and Labidi, M. (2007) 'The DTW data distribution over a grid computing architecture', *International Journal of Computer Sciences and Engineering Systems*, Vol. 1, No. 4, pp.241–247.

Kosar, T. and Livny, M. (2004) 'Stork: making data placement a first class citizen in the grid', *24th International Conference on Distributed Computing Systems*, *Proceedings*, pp.342–349, DOI: 10.1109/ICDCS.2004.1281599.

Labidi, M., Jemni, M. and Khemakhem, M. (2017) 'Co-scheduling data and task for a data-driven distribution of data-intensive applications', in *AICCSA*, IEEE Computer Society, pp.407–414.

Labidi, M., Tang, B., Fedak, G., Khemakhem, M. and Jemni, M. (2012) 'Scheduling data on data-driven master/worker platform', in Shen, H., Sang, Y., Li, Y., Qian, D. and Zomaya, A.Y. (Ed.): *PDCAT*, IEEE, pp.593–598.

Makatun, D., Lauret, J., Rudovà, H. and Sumbera, M. (2015) 'Planning for distributed workflows: constraint-based coscheduling of computational jobs and data placement in distributed environments', *Journal of Physics: Conference Series*, Vol. 608, p.012028, 10.1088/1742-6596/608/1/012028A.

Rajasekar, A., Moore, R., Hou, C-Y., Lee, C.A., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S-Y., Gilbert, L., Tooby, P. and Zhu, B. (2010) *iRODS Primer: Integrated Rule-Oriented Data System*, Synthesis Lectures on Information Concepts, Retrieval, and Services, Morgan & Claypool Publishers.

Romosan, A., Rotem, D., Shoshani, A. and Wright, D. (2005) 'Co-scheduling of computation and data on computer clusters', in Frew, J. (Ed.): *SSDBM*, pp.103–112.

Saidala, R.K. and Devarakonda, N. (2018) 'Chaotic tornadogenesis optimization algorithm for data clustering problems', *IJSSCI*, Vol. 10, No. 1, pp.38–64.

Smith, R. (2007) 'An overview of the Tesseract OCR engine', in *ICDAR*, IEEE Computer Society, pp.629–633.

Wu, J., Guo, S., Huang, H., Liu, W. and Xiang, Y. (2018) *Information and Communications Technologies for Sustainable Development Goals: State-of-the-Art, Needs and Perspectives*, DOI: CoRR abs/1802.09345.

Wu, J., Guo, S., Li, J. and Zeng, D. (2016a) 'Big data meet green challenges: big data toward green applications', *IEEE Systems Journal*, Vol. 10, No. 3, pp.888–900.

Wu, J., Guo, S., Li, J. and Zeng, D. (2016b) 'Big data meet green challenges: greening big data', *IEEE Systems Journal*, Vol. 10, No. 3, pp.873–887.