Turkish Journal of Electrical Engineering and Computer Sciences

Volume 25 | Number 1 Article 39

1-1-2017

A data-aware cognitive engine for scheduling data intensive applications in a grid

VIJAYA NAGARAJAN

MALUK MOHAMED MULK ABDUL

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 497 – 507

doi:10.3906/elk-1508-87

A data-aware cognitive engine for scheduling data intensive applications in a grid

Vijaya NAGARAJAN[1,*], **Maluk MOHAMED MULK ABDUL**[2]

[1]Department of Information and Communication Engineering, Anna University, Chennai, Tamilnadu, India
[2]Department of Computer Science and Engineering, M.A.M. College of Engineering, Trichy, Tamilnadu, India

Received: 11.08.2015 • **Accepted/Published Online:** 10.01.2016 • **Final Version:** 24.01.2017

Abstract: Data-intensive applications produce huge amounts of data that need to be stored, analyzed, and interpreted. A data grid serves as a cost-effective infrastructure for solving these data-intensive applications. Existing scheduling strategies are best suited for handling compute-intensive applications, although they lack in performance while handling data-intensive applications. In this work, a novel mechanism of incorporating cognitive science in a data grid is proposed for scheduling data-intensive workflows. A unique model is derived in which a cognitive engine (CE) is built into the middleware of the data grid. The intelligent agents present in the CE handle the request for data sets and use the LTP algorithm (learning, thinking, and perception) to effectively schedule the tasks using three phases. The CE also finds a unique solution for placing data sets dynamically nearer to the execution site based on network resource considerations by reducing the waiting time and data availability time for I/O-intensive jobs. The performance of the CE is validated by simulation and compared with that of existing scheduling strategies. The results of the simulation show that CE optimizes the data availability time, waiting time, data transfer time, and makespan.

Key words: Scheduling, cognitive science, cognitive engine, data intensive workflows, intelligent agents

1. Introduction

A grid serves as a promising infrastructure for solving large scientific applications such as high-energy physics, structural biology, neuroscience, and bioinformatics by integrating globally distributed heterogeneous resources [1]. Scientific applications are usually called workflows in a grid. A workflow is defined as a set of sequentially ordered tasks linked by logic or data dependencies and a grid workflow management system is designed exclusively to define, manage, and execute these workflow applications [2]. Workflows are classified as compute intensive and data intensive. When an individual task in workflows demands more usage of CPU then it is compute intensive. When huge amounts of data sets are processed by tasks then it is data intensive.

The following applications are examples for data intensive workflows: global weather prediction, which uses the observations made by a remote satellite; digital sky project, which exclusively makes sky surveys; brain imaging analysis, which analyzes the human brain activity; and high-energy physics, which produces huge data sets [3]. Scheduling workflows in a grid is challenging due to the following characteristics of the grid [4]: (1) the resources in the grid are heterogeneous in nature; (2) the resources are in different administrative domains; (3) resources suffer from different usage pricing and policies.

Cognitive science is an interdisciplinary field that studies the behavior of the human brain. It extracts knowledge from various fields such as philosophy, psychology, linguistics, artificial intelligence, robotics, and

*Correspondence: vijiannaunivedu@gmail.com

neuroscience [5]. The real understanding of cognitive science lies in the information processing of the human brain. Information processing includes the details of how things are perceived, processed, transformed, and presented inside the human brain. One of the basic cognitive processes of humans is problem solving. How this can be done by a human? Humans store lots of information starting from the day one of their birth to old age. When a problem is encountered, the human brain searches for the solution in the memory space and finds the way to solve the problem. The inherent mechanism in this problem-solving approach is the idea of modularity. This simulated technique of the human brain is used in computational modeling.

In this work, the human intelligence in problem solving is incorporated in every aspect of scheduling process. A unique model is proposed in which a cognitive engine (CE) is built into the middleware of the data grid where each site in the data grid will have a CE. Task executing at a site requires data sets for processing and sends the request to the CE. If the data sets are available locally, then the CE will process the request successfully within the site itself. If the data sets are not available locally, then the CE will process the request using the LTP algorithm. The LTP algorithm uses three phases: learning, thinking, and perception to effectively schedule the request across the data grid. The learning agents present in the CE will learn the knowledge base, partitioned repository, data catalogue, and prediction engine before taking dynamic decisions.

The main contributions of this paper are as follows:

- A unique model is proposed by incorporating a CE in the middleware of the data grid.
- A novel LTP algorithm using learning, thinking, and perception phases is proposed to optimize the makespan.

The rest of the paper is organized as follows: Section 2 gives an insight into workflow scheduling in the data grid, Section 3 specifies the related works carried out in scheduling, and Section 4 deals with the impact of cognitive science in grid scheduling. It also describes the proposed strategy in detail. Performance is evaluated in Section 5. Conclusions and future works are presented in Section 6.

2. Workflow scheduling in a data grid

A data grid consists of several sites, where each site has many compute and data servers. Data are distributed and stored across various data servers. Data-intensive workflows involve accessing data in the range of terabytes or petabytes and serve as a suitable environment for storing and processing the data products. Many workflow scheduling systems designed so far concentrated on computation optimization rather than data management. Hence scheduling data-intensive workflows becomes a research issue when data-transmission requirements are taken into consideration [6]. Data-intensive workflow scheduling is defined as the process of mapping and managing the execution of interdependent tasks on geographically dispersed resources [7]. The following steps are involved in processing of a data-intensive application.

- Workflow construction:

  In this step, an e-science application is converted to workflow, i.e. a dependency graph is drawn considering nodes as tasks and links as dependencies between tasks.

- Resource planning:

  In this step, the resources required for executing the tasks are selected depending on the user requirements. The tasks are allocated to the selected resources.

- Workflow execution:

  In this step, the execution of the tasks in the selected resources is monitored for data handling, resource failure, storage constraints etc.

- Storage and analysis:

  In this step, the result of the execution stage is stored, analyzed, and interpreted to the end user.

While scheduling the data intensive workflows, many issues [3,8–13] such as heterogeneity, granularity, replication, storage, security, locality, and fault tolerance are to be taken care of during the planning, execution, and storage phases.

3. Related work

Many scheduling algorithms and heuristic based solutions have been proposed for scheduling data-intensive workflows in grid environments. However, only a very few works have concentrated on data locality-aware scheduling. The work in [10] suggested a model in which the most often accessed files are replicated and transferred to the site where the data sets are required. Here the authors have proposed two sets of algorithms: external scheduler algorithms and data set scheduler algorithms. The external scheduler is responsible for scheduling the jobs submitted by users across various sites and the data set scheduler monitors the popularity of the file requested and is involved in replicating and deleting local files. In [11], the site that requires data sets for processing should subscribe the data sets. The system will take care of transferring the required files in the data sets to the subscribed site. This requires more human effort. In [12], the job components are placed on or close to the sites where the input file is located. Here a history of the execution site and the file site is maintained. Jobs are assigned to the site having minimum file transfer time. In [13], data are considered for scheduling jobs. When the jobs are queued at a particular site and request files for computation, then the penalty of transferring the required data files is calculated based on the file catalogue information. In [14], a file access log is maintained. Instead of transferring the data sets to the jobs site, here jobs will be launched to the node where the input files are located either in the local disk or in the memory. In [15] a strategy is proposed to place data replicas in the optimal location so that the maximum amount of data sets served by the replica server is minimized. A tree model is used to study and analyze the placement of replicas based on the usage frequency of the user on particular data sets. In [16], scheduling is based on network resource considerations and data locations of input files. Even though the gridway metascheduler is not suitable for data intensive jobs, the author tried to improve it.

In [17], a metascheduling approach, DIANA, is proposed, in which the jobs are allocated to the resources across multiple sites by considering the data, processing power, and network characteristics. The data location service provides the best replica of the data sets for computation. In [18], the authors proposed two algorithms: 1. CSS (combined scheduling strategy), a job scheduling algorithm that uses a hierarchical grid structure to search for the best site from a large number of distributed sites. Here the best site has the most accessed files. 2. MDHRA (modified dynamic hierarchical replication algorithm), which improves file access time by selecting the best replica location. In [19] a dynamic data structure (DDT – data distance table) is proposed. Based on this information, a data-aware scheduling heuristic is proposed that minimizes the data access delay. This heuristic technique reduces the scheduling holes created when the data replica does not exist. It also incorporates data movement in the existing min–min scheduling heuristics.

Even though extensive research has focused on makespan optimization, this work has two differentiators: 1. Introduction of cognitive science in data-aware scheduling to reduce the data availability time and queuing delay, since these are important metrics that greatly influence the makespan. 2. With the help of learning, thinking, and prediction phases the data can be transferred well in advance before the computation begins. This will reduce the data availability time and waiting time.

4. Cognitive engine

In this work, each site in the data grid will have a CE. Figure 1 shows the architecture of the CE. It performs the following functions:

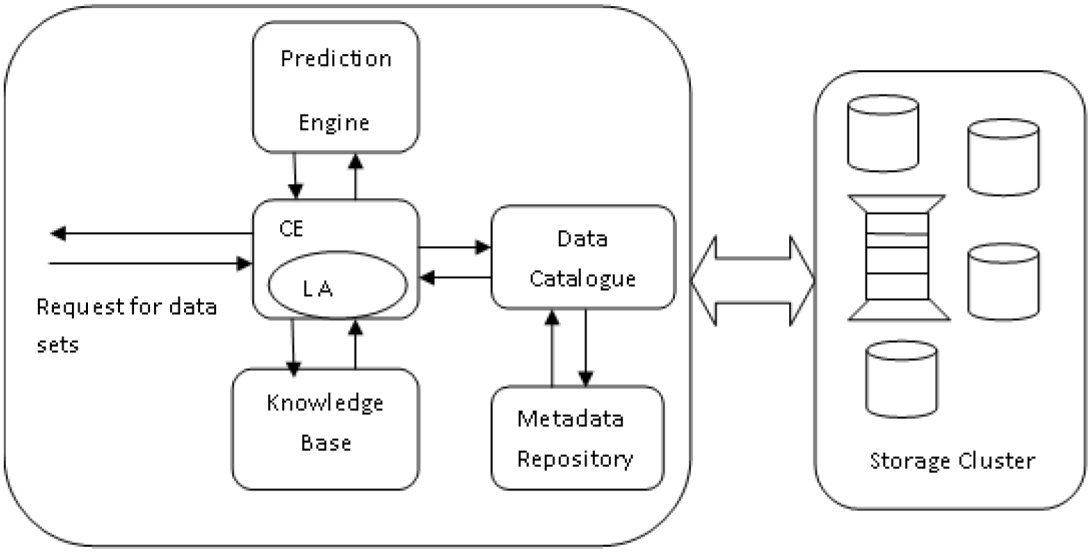

Figure 1. Cognitive engine at site 1.

- Construction of data catalogue (DC): This catalogue holds the location of the datasets (i.e. site id) and the details of the data sets.

- Storage matrix: It shows the presence of data in the particular site. When the data set passes through a neighboring site, the storage matrix of that site is checked for the presence of particular data. If the data are not found then the data get stored at that site.

- Construction of knowledge base (KB): The requests for data sets from various sites are stored in the KB. Each request in the KB contains the query, which involves the site id and the data sets.

- Learning agents (LAs): The LAs will take dynamic decisions based on the request made by several sites.

- Prediction engine (PE): The dynamic PE predicts the data sets needed by the site based on the KB. It calculates the access frequency of the stored data sets. Let us assume that sites A and B require data sets d1 and d2 for processing and another site, C, requires data sets d2. After satisfying the request of A and B, the prediction engine decides to send d1 along with the requested data set d2 to the site B.

- The access frequency of the data is calculated by counting the number of times the particular data are accessed upon the total number of accesses made.

$$\text{Access frequency } = \frac{\text{Number of times dsi accessed}}{\text{Total number of accesses made}}$$

  Since the requests are predicted in advance, the makespan and delay are considerably reduced.

- Partitioned meta-data repository (PMR): This repository holds the details of the partitioned data sets and the location of the partitioned data sets. For partitioning of large data sets, range has to be identified by dividing the size of the data sets by the bandwidth of the link. Depending upon the range values, the data sets are partitioned into equal-sized partitions. The partitioned details are kept in a repository that acts as a lookup table with partition id, range, and site id.

4.1. Intelligent agents and prediction

In Figure 1, let us assume a task executing at site 1 sends the request for data sets to the CE at site 1. If the requested data sets are available locally, then the task can be executed without any further delay. If the requested data sets are not available at the same site then the LAs will search the DC (Table 1) to find the information about the data sets. The DC holds the details of the data sets and their location. Based on the location information about the data sets, the request is forwarded to the CE at the remote site. In the remote site CE, the request is first stored in the KB and then processed. The required data sets are sent to site 1. The intelligent agents in the remote CE will search the KB for the presence of the same request. If the same request is already available, then the subsequent request made by the corresponding site is searched by the LAs. These agents will fetch the required data sets in advance and send them to site 1. This is done by the prediction engine. Therefore, the requested data sets and the predicted data sets for the next request are sent to site 1. This will reduce the placing of the next request by site 1 since the data sets needed by site 1 are predicted and transferred in advance to the requested site S1. This will minimize the data set requests and the waiting time of the tasks. The queuing time and data availability time are reduced, which in turn will create a major impact on overall workflow scheduling.

Table 1. Data catalogue at initial stage.

Siteid (node id)	Data sets
S1	d1, d2
S2	d3,d7
S3	d4
S4	d1, d5
S5	d1, d5, d4
S6	d8
S7	d2
S8	d6

4.2. Intelligent agents and replication

The intelligent agents present in the prediction engine increment the access count, which is the access frequency of the data sets calculated whenever the data sets are accessed. When the access count is greater than the threshold value, the corresponding data sets get replicated to all sites. Replicating a data set to all grid sites

that is accessed only once during the workflow execution occupies the storage space of a site unnecessarily, since the data sets are huge. If a task requires a data set that is already accessed by some other site then the data set access count is incremented. A threshold value is set depending upon the application. If the threshold value is set to 5, then for 5 times the data set may be accessed from the remote site. If the next request is also for the same data set then the data set gets replicated to all the sites in the grid. Hence the CE will replicate the data set which is most frequently accessed.

4.3. Link and transmission

The CE will also look into the transmission capacity of the link and the amount of data sets requested by the site. The information about the link capacity or bandwidth can be obtained from the Network Weather Service [20]. With the help of this information the least cost table is constructed (Table 2). Figure 2 shows the graphical representation of the interaction between different sites. A graph G = {V, E} , where V is the set of n sites $\{S_1, S_2 \ldots S_n\}$ and E is the set of weighted edges between the sites. The weight is the cost assigned based on the bandwidth of the link. If the link bandwidth is high, least cost is assigned. Else the cost assigned is high. Table 2 shows the least cost table with eight sites. Each entry in the table gives the source site id, destination site id, bandwidth, and the cost assigned for the link. Normally, data-intensive applications process huge data sets; if the bandwidth is too low to send the data sets, then the CE will partition the data sets into smaller chunks capable for transmission using the current network bandwidth.

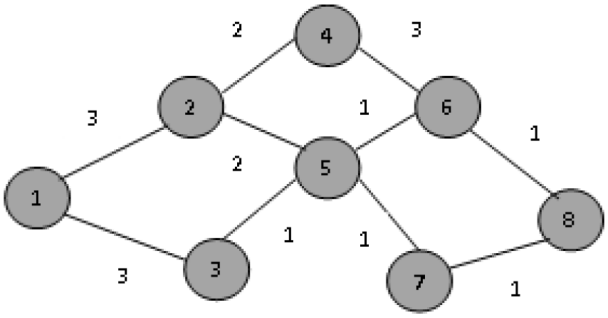

Figure 2. Graphical representation of sites linked by network links.

Table 2. Least cost table.

Source	Destination	Bandwidth (Mbps)	Cost
S1	S2	8	3
S1	S3	8	3
S2	S5	10	2
S2	S4	10	2
S3	S5	16	1
S4	S6	8	3
S5	S7	16	1
S6	S8	16	1
S7	S8	16	1

4.4. Working of the CE

Let us assume a task executed at site S2 requires datasets d1 for further processing. Data sets d1 are available at sites S1, S4, and S5 (Table 1). The CE at S2 searches the least cost table to find a nearby site for accessing

the data set d1. S5 has the least cost with more bandwidth. Hence the CE sends a remote request to site S5. The request contains the originating site id, required data sets, and destination site id. When the request reaches the CE at site S5, the request is first stored in the KB and the requested data set d1 is sent to S2. Now the storage matrix is searched for the associated data sets and it is noted that S1, S4, and S5 accessed d1 for processing. Then the associated data sets of these three sites are found using the storage matrix. The access count of the associated data sets is calculated. The data sets that have the highest access count are sent to the site S2. In this example, the data set d4 is sent to S2.

By this approach, if the next request for S2 is d4, then S2 need not place the request to the remote CE, since the data sets are already available locally. Hence the request time, queuing time, and data availability time are considerably reduced. Even though extensive research has focused on makespan optimization, this work has two differentiators: 1. introduction of a CE, to reduce the data availability time and queuing delay since they are the important metrics that greatly influence the makespan; 2. with the help of the PE, LA, and KB the data can be transferred well in advance before the computation begins. This will reduce the data availability time and waiting time.

Table 3 shows the initial storage matrix and Table 4 shows the updated DC after replication.

Table 3. Storage matrix at initial stage.

	d1	**d2**	**d3**	**d4**	**d5**	**d6**	**d7**	**d8**
S1	1	1	0	0	0	0	0	0
S2	0	0	1	0	0	0	0	0
S3	0	0	0	1	0	0	0	0
S4	1	0	0	0	1	0	0	0
S5	1	0	0	1	1	0	0	0
S6	0	0	0	0	0	0	0	1
S7	0	1	0	0	0	0	0	0
S8	0	0	0	0	0	1	0	0

Table 4. Data catalogue after transferring.

Siteid (node id)	Data sets
S1	d1, d2
S2	d3,d7
S3	d4
S4	d1, d5
S5	d1, d5, d4
S6	d8
S7	d2
S8	d6

4.5. LTP algorithm

The proposed LTP algorithm will have three phases: learning, thinking, and perception.

4.5.1. Learning phase

The intelligent agents present in the CE will learn the request and build the KB with the requesting site id and the data. The KB is studied to see what data sets are associated with the requested data sets. To find the

associated data sets, the LAs will search the entire KB for the presence of the same request. If the same request is available, then the subsequent request made by the same site is analyzed. Now the associated data set is also transferred to the requesting site. This will eliminate the site to request the associated data sets again. Hence the data availability time is reduced. Here the LAs process the request by sending the requested data sets to the requesting site.

4.5.2. Thinking phase

The intelligent agents will take the decision whether the data sets can be replicated or not based on the access frequency of the particular data. If the access frequency is beyond the threshold limit then the data will get replicated across several sites. This will enhance the data availability and reduce the access time since the requested data sets may be accessed from the neighboring site itself. The agents also look up the least cost table to find out the site that has high bandwidth to send the data. This will enable effective network bandwidth usage. The partitioned metadata repository is updated, which holds the partition id, range, and the site id where the partition resides.

4.5.3. Perception phase

The intelligent agents will delete the replicas at a particular site for two reasons: 1. if the storage space is not sufficient to store the data sets, then the data sets with least access frequency will be deleted; 2. every time, when a new data set gets stored, the old data set with least access frequency is deleted. This will avoid the overhead of checking the storage space before storing the data sets. Once the data get deleted, the intelligent agents in the CE will update the DC, least cost matrix, and storage matrix

5. Performance evaluation

In this section, we analyze the performance of the proposed approach with the existing approaches discussed in the related works. The performance metrics used for analysis are data availability time, data transfer time, waiting time, execution time, and makespan. A simulation environment is set up using gridsim [21], the java-based simulation tool kit to measure the performance. We have considered a network composed of three sites: each site consists of a cognitive engine CE, a set of processors, and storage servers. Here, the basic assumption we made is all the processors are operating at the same speed. Jobs are submitted to the simulated grid environment and we measured the results with respect to the following parameters.

5.1. Impact of CE on waiting time

In this section, the proposed approach is compared to the existing benchmark min–min and HEFT heuristics. Figure 3 shows the waiting time of jobs compared with other strategies. It is the time required by the task to collect the required data sets for execution. It consists of two components: data transfer time and data availability time.

$$WT(t_k) = DTT(t_k) + DAT(t_k) \tag{1}$$

It is shown that the waiting time is considerably reduced when a CE is incorporated in the data grid environment. While executing a job, if the future need is predicted in advance and if the data needed for the execution are sent to the execution site in advance, then the job will get executed without any further delay. Hence the waiting time is reduced. Since the associated data sets are available at the computing site in advance, there is no need to wait for the data sets.

5.2. Impact of CE on data availability time

Figure 4 shows that the data availability time is reduced when a CE is incorporated in the data grid environment. It is calculated based on the time required for the task to fetch the input data sets for execution (i.e. data staging time for computation). When the number of requests for data set increases, the intelligent agents will analyze the situation every time and the decision is taken dynamically. The PE helps in predicting the required data set in advance, which in turn will eliminate unnecessary queuing delay since the predicted data sets are available in the computing site in advance. This will reduce the data availability time.

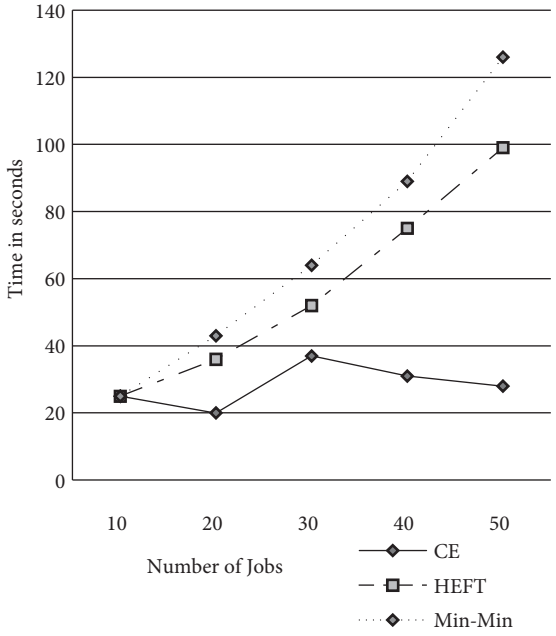

Figure 3. Analysis of waiting time.

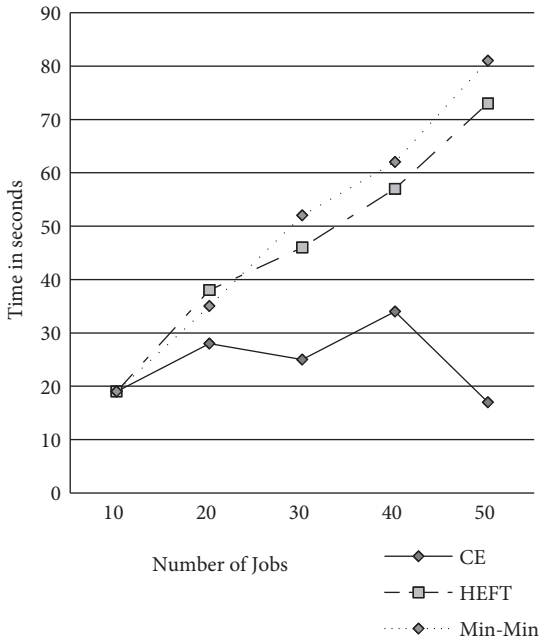

Figure 4. Analysis of data availability time.

5.3. Impact of CE on data transfer time

In this section, the data transfer time from the data site to the requesting site is compared to min–min and HEFT heuristics [4]. It is calculated by finding the time required for transferring the data sets from one site to another site. Figure 5 shows that the data transfer time is also considerably reduced since the data sets are sent through the high bandwidth link by referring to the least cost table. When the data sets reside in more than one site, the CE refers to the least cost table for high bandwidth. The least cost link is chosen to transfer the data sets. This will ultimately eliminate data being transferred in the low bandwidth line and reduce the data transfer time and minimize the delay incurred during data transfer.

5.4. Impact of CE on makespan

The primary goal of this scheduling approach is to minimize the makespan. Makespan M is the time elapsed between the submission of the first task in the workflow and the time of the result after the execution of the last task in the workflow.

$$M = \sum_{k=1}^{n} (\text{Waiting time } (t_k) + \text{Execution time } (t_k)) \tag{2}$$

Eq. (2) states that waiting time and the execution time will have major impact in the makespan. In order to optimize the makespan any one of the parameters has to be optimized. The parameter execution time depends on the resource performance, which can be optimized by selecting the best resource during the scheduling, and waiting time depends on the data availability time and data transfer time as stated in Eq. (1). Generally, in grid scheduling, we have three phases [9]: resource discovery, where the available resources are listed; system selection, where the best resource is selected from the available resources; and job execution, where the data staging, execution, and clean-up operations are carried out.

This work is focused only on the third phase of the scheduling process. Here a data-aware cognitive engine is designed based on the theory of cognitive science and incorporated in the middleware of the data grid. The main objective is to optimize the makespan by minimizing waiting time. In order to minimize the waiting time, as given in Eq. (2) the DAT and DTT should be minimized. Figure 6 shows that by reducing the data availability time, queuing delay, and data transfer time the overall makespan is reduced.

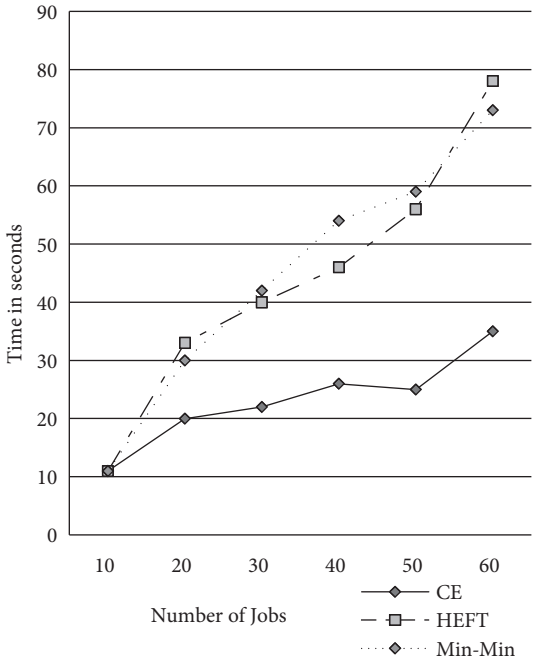

Figure 5. Analysis of data transfer time.

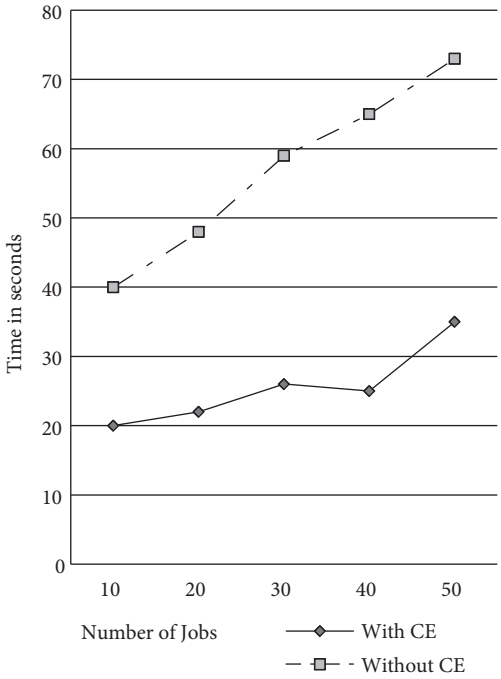

Figure 6. Analysis of makespan.

6. Conclusion

In this work, we have introduced a novel strategy of incorporating cognitive science for scheduling data-intensive workflows. By placing a CE in the middleware of the data grid, the scheduling performance is optimized. The features of cognitive science like learning from experience, inference, and prediction help to make intelligent decisions at run time. This proposed strategy optimizes the waiting time of jobs in queue, data availability time, data transfer time, and makespan. Still some issues like job failure and resource failure are not addressed in this paper. In future, this work may be extended to resolve the failure and may be applied to scheduling problems in the cloud environment.

References

[1] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: enabling scalable virtual organizations. Int J High Perform C 2001; 15: 200-222.

[2] Pandey S, Buyya R. Scheduling of scientific workflows on data grids. In: IEEE International Symposium on Cluster Computing and the Grid (CCGRID); 2008; pp. 548-553.

[3] Foster I, Kesselman C. The Grid: Blueprint for a New Computing infrastructure. San Fransisco, CA, USA: Morgan Kaufmann, 1999.

[4] Yu J, Buyya R, Ramamohanarao K. Workflow scheduling algorithms for grid computing. Metaheuristics for Scheduling in Distributed Computing Environ ments, Berlin, Germany: Springer 2008, pp. 173-214.

[5] Friedenberg J, Silverman G. Cognitive Science: An Introduction to the Study of Mind. Thousand Oaks, CA, USA: Sage, 2012.

[6] Kołodziej J, Xhafa F, Barolli L, Kolici V. A taxonomy of data scheduling in data grids and data centers: problems and intelligent resolution techniques. In: IEEE International Conference on Emerging Intelligent Data and Web Technologies (EIDWT); 2011; pp. 63-71.

[7] Yu J, Buyya R. A taxonomy of workflow management systems for grid computing. J Grid Comput 2005; 3: 171-200.

[8] Kwok YK, Ahmad I. Benchmarking and comparison of the task graph scheduling algorithms. J Parallel Distr Com 1999; 59: 381-422.

[9] Schopf JM. Ten actions when grid scheduling. In: Grid Resource Management: State of the Art and Future Trends. Norwell, MA, USA: Academic, 2003. pp. 15-23.

[10] Ranganathan K, Foster I. Decoupling computation and data scheduling in distributed data-intensive applications. In: IEEE International Symposium on High Performance Distributed Computing (HPDC); 2002; pp. 352-358.

[11] Rehn J, Barrass, Bonacorsi D, Hernandez J, Semeniouk I, Tuura L, Wu Y. PhEDEx high-throughput data transfer management system. In: International Conference on Computing in High Energy and Nuclear Physics (CHEP); 2006; pp. 173-177.

[12] Mohamed, Hashim H, Epema DH. An evaluation of the close-to-files processor and data co-allocation policy in multiclusters. In: IEEE International Conference on Cluster Computing; 2004; pp. 287-298.

[13] Cameron DG, Carvajal-Schiaffino R, Paul Millar A, Nicholson C, Stockinger K, Zini F. Evaluating scheduling and replica optimisation strategies in OptorSim. In: IEEE International Workshop on Grid Computing; 2003; pp. 52-59.

[14] Shibata T, Choi S, Taura K. File-access characteristics of data-intensive workflow applications. In: IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid) ; 2010; pp. 746-755.

[15] Lin YF, Liu P, Wu JJ. Optimal placement of replicas in data grid environments with locality assurance. In: IEEE International Conference on Parallel and Distributed Systems (ICPADS); 2006; pp. 736-744.

[16] Kumar S, Kumar N. Network and data location aware job scheduling in grid: improvement to GridWay meta scheduler. Int J Grid Distr Comput 2012; 5: pp. 87-100.

[17] McClatchey R, Anjum A, Stockinger H, Ali A, Willers I, Thomas M. Data intensive and network aware (DIANA) grid scheduling. J Grid Comput 2007; 5: pp. 43-64.

[18] Mansouri N, Dastghaibyfard GH, Mansouri E. Combination of data replication and scheduling algorithm for improving data availability in data grids. J Netw Comput Appl 2013; 36: 711-722.

[19] He X, Sun XH. Incorporating data movement into grid task scheduling. In: International Conference on Grid and Cooperative Computing (GCC); Berlin, Germany: Springer, 2005. pp. 394-405.

[20] Wolski R, Spring NT, Hayes J. The network weather service: a distributed resource performance forecasting service for metacomputing. Future Gener Comp Sy 1999; 15: 757-768.

[21] Buyya R, Murshed M. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurr Comp Pract E 2002; 14: 1175-1220.