Data Scheduling in Data Grids and Data Centers: A Short Taxonomy of Problems and Intelligent Resolution Techniques

Joanna Kołodziej[1] and Samee Ullah Khan[2]

[1] Institute of Computer Science
Cracow University of Technology, ul. Warszawska 24, 31-155 Cracow, Poland
jkolodziej@uck.pk.edu.pl
[2] NDSU-CIIT Green Computing and Communications Laboratory,
North Dakota State University,
ND 58108, USA
samee.khan@ndsu.edu

Abstract. Data-aware scheduling in today's large-scale heterogeneous environments has become a major research issue. Data Grids (DGs) and Data Centers arise quite naturally to support needs of scientific communities to share, access, process, and manage large data collections geographically distributed. Data scheduling, although similar in nature with grid scheduling, is given rise to the definition of a new family of optimization problems. New requirements such as data transmission, decoupling of data from processing, data replication, data access and security are to be added to the scheduling problem are the basis for the definition of a whole taxonomy of data scheduling problems. In this paper we briefly survey the state-of-the-art in the domain. We exemplify the model and methodology for the case of data-aware independent job scheduling in computational grid and present several heuristic resolution methods for the problem.

Keywords: Data Grid, Scheduling, Data Center, Expected Time to Transmit, Data replication.

1 Introduction

Traditional scheduling problems are mainly concerned with high performance parameters related to task processing (CPU related parameters) such as makespan, flowtime, resource usage, etc. These parameters usually do no take into account requirements on data needed for task completion such as data transmission time, data access rights, data availability (replication) and security issues. In most of the research on grid and cloud computing data transmission time is assured to be fast/very fast, data access rights are granted, due to the single domain of LANs and clusters, so there is no need for special data access management. Similarly, security issues are easily handled within the same administrative domain. Obviously, the situation is very different in current large scale setting, where

N.T. Nguyen (Ed.): Transactions on CCI X, LNCS 7776, pp. 103–119, 2013.

data sources needed for task completion could be located at different sites under different administrative domains.

Although data-aware scheduling has been considered in a significant volume of the research works, e.g. in parameter sweep applications [2,3], the scheduling problems in Computational Grids (CGs) and in Data Grids (DGs) is dealing with in a separated way. Much of the current efforts are focused on scheduling workloads in a data center or schedule movement of data and data placement [38] for efficient resource/storage utilization or energy-effective scheduling in large-scale data centers [37], [8], [29], [18], [44]. A recent example is that of GridBatch [40] for large scale data-intensive problems on cloud infrastructures.

Due to advent of DGs and fast development of Cloud Computing, data-aware scheduling has recently attracted considerable attention of researcher from distributed computing and optimization communities. In fact, DGs can be seen as precursors of Data Centers in Cloud Computing platforms, which serve as basis for collaboration at a large scale. In such computational infrastructures, the large amount of data to be efficiently processed is a real challenge. One of the key issues contributing to the efficiency of massive processing is the scheduling with data transmission requirements.

In this work, we consider the data-aware scheduling aiming to problem formulations that take into account new requirements such as data transmission, decoupling of data from processing [32], [47], [53],, data replication [7], [10], data access and security, [33], [16], [17]. The aim is to integrate these new requirements into a multi-objective optimization model in a similar way that it has been addressed for a classical grid scheduling. The grid schedulers must thus take into account the features of both CG and of DG to achieve desired performance of grid-enabled applications [34], [35]. We exemplify the approach for the case of data-aware independent batch task scheduling problem.

The remainder of this paper is structured as follows. We present in Section 2 a high level taxonomy for data scheduling in Data Grids. The data-aware system model for independent batch scheduling is given in Section 3. Selected heuristic-based resolution methods for solving data-aware independent batch scheduling are presented briefly in Section 4. We discuss the most important challenges in data-aware scheduling in Section 5 and conclude this paper in Section 6.

2 A Short Taxonomy of Data-Aware Scheduling Problems in Data Grids

Data Grids (GGs) are defined as computational infrastructures that provide high performance massive aggregated computing resources and distributed data storage capabilities. DGs support data intensive applications. Among several types of Data Grids elements four components seem to be fundamental, namely *Grid Organization* module, *Data Replication* mechanism, *Data Transfer* policy and infrastructure and *Scheduling* module (see also [46]) as shown in Fig. 1.

The complex hierarchy of the DG can be then organized as a collection of four sub-hierarchies, each of them dedicated to one of the DG's elements. Such

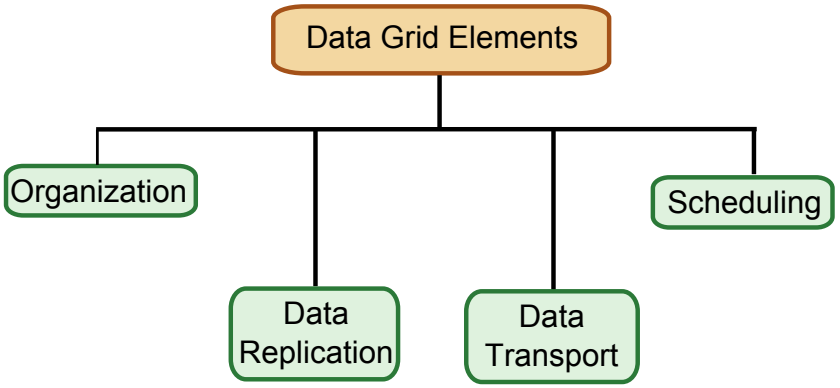

Fig. 1. Data Grid elements (based on the full taxonomy presented in [46])

complex DG characteristics are presented in [46]. In fact, each of the areas of data transport, replica management and resource management pose challenging research issues and can be analyzed as independent research areas. However, in the recent studies on grid systems there is a need to analyze the specific requirements of DG's users and environments, which in fact tends to a direct or indirect aggregation of the particular grid elements and methodologies into wider classes. In this paper we focus on the *Scheduling* sub-hierarchy.

The requirements for large data files and the presence of multiple replicas of these data files located at geographically-distributed data hosts makes scheduling of data-intensive tasks different from that of simply computational tasks. Schedulers have to take into account the network bandwidth availability and the latency of data transfer between a computational node to which a task is going to be submitted, and the storage resource(s) from which the data required is to be retrieved [19], [11]. Therefore, the scheduler needs to be aware of any replicas "close" to the computation node and if the replication is coupled to the scheduling, then create a new copy of the data.

A basic taxonomy for scheduling of data-intensive applications is shown in Fig. 2.

There are five main categories in the taxonomy, which can be characterized in the following paragraphs.

Application Model. Scheduling strategies in DGs can be classified by the application models, which are mainly determined by the manner in which the grid task is composed or distributed for scheduling. The grid tasks may be categorized into the following classes:

- *Process-oriented applications*– in this applications the data is manipulated at the process level (Message Passing Interface (MPI) programs [5]).
- *Independent tasks* – can have different objectives or may be defined as a meta-task or a bag-of-tasks. They are scheduled individually and it is ensured that each of them get the required share of resources for their completion [41].

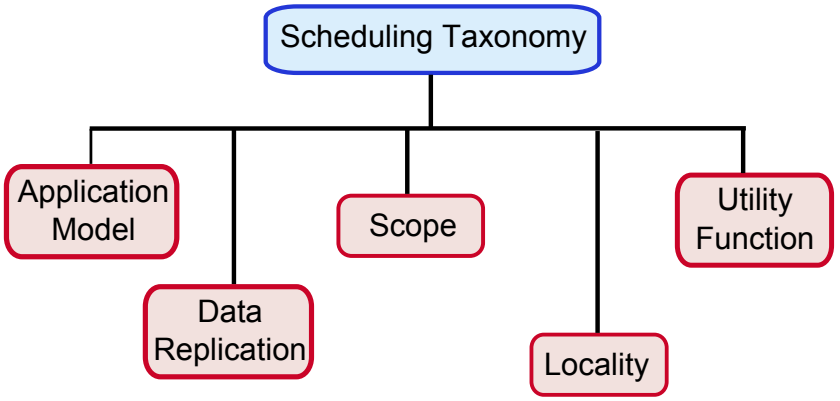

Fig. 2. Data Grid scheduling taxonomy

– *A workflow* ([45]) is a sequence of tasks in which each task is dependent on the results of its predecessor(-s) tasks. The outputs of the preceding tasks may be large data files themselves.

Scope. Scope relates to the modification of application of the scheduling strategy within the DG. If the scope is *individual*, then the scheduling strategy is concerned only with meeting the objectives from a user's perspective. In a multi-user environment, each scheduler would have its own independent view of the resources that it wants to utilize. A scheduler can be for example aware of fluctuations in resource availability, special security requirements and other policies set at the Virtual Organization level and enforced at the resource level ([48]).

Data Replication. This category relates to whether task scheduling is coupled to data replication or not. A comparison analysis of decoupled *vs.* coupled strategies performed in [42] has shown that decoupled strategies promise increased performance and reduce the complexity of designing algorithms for DG environments. Additionally, one can consider replicating full data sets or chunks of data sets. Related to data replication, there are usually a set of multiple user *QoS*. For instance, access time to data, data availability, etc. can be seen as QoS requirements. Certainly, such requirements should be taken into account by the grid scheduler [28], [9], [12], [13].

Utility Function. The utility function can vary depending on the requirements of the users and architecture of the distributed system that the algorithm is targeted to. Traditionally, scheduling algorithms have aimed at reducing in average the total time required for computing all the tasks in a given batch or set. Load balancing algorithms try to distribute load among the machines so that maximum work can be obtained out of the systems. Scheduling algorithms with economic objectives try to maximize the users' economic utility usually

expressed as some profit function that takes into account economic costs of executing the jobs on the DG. Recently, one of the key objective in green grid, cloud and high performance computing centers is to optimize the energy utilization by the system.

Locality. Exploiting the locality of data has been a common technique for scheduling and load-balancing in parallel programs [6] and in query processing in databases [43]. Similarly, data grid scheduling algorithms can be categorized as whether they exploit the spatial or temporal locality of the data requests. Spatial locality is locating a task in such a way that all the data required for the task is available on data hosts that are located "close" to the node of computation. Temporal locality exploits the fact that if data required for a task is close to a compute node, subsequent tasks which require the same data are scheduled to the same node to benefit from the data proximity.

3 Data-Aware System Model for Independent Job Scheduling

Let us consider now the problem of batch scheduling of independent data-intensive applications onto computational grid's resources. The applications can be considered as meta-tasks, which require multiple data sets from different heterogeneous data hosts. These data sets may be replicated at various locations and can be transferred to the computational grid through the networks of various capabilities. A possible variant of this scenario is presented in Fig. 3.

Formally, the main components of the data-aware grid system can be characterized as follows:

- a meta-task $N_{batch} = \{t_1, \ldots, t_n\}$ defined as a batch of independent tasks, each of which can be executed just at one or more grid resources (n - is a total number of tasks in the batch);
- a set of computing grid nodes $M_{batch} = \{m_1, \ldots, m_k\}$, ($k$ - is a total number of machines available in the system for a given batch;
- a set of data-files $F_{batch} = \{f_1, \ldots, f_r\}$ needed for the batch execution;
- a set of data-hosts $DH = \{dh_1, \ldots, dh_s\}$ dedicated for the data storage purposes, having the necessary data services capabilities.

3.1 Task Workload and Computing Capacities

The computational load of the batch can be defined as a *tasks workload vector* $WL_{batch} = [wl_1, \ldots, wl_n]$, where wl_j denotes an estimation of the computational load of a task t_j (in Millions of Instructions –MI). Each task t_j requires a set of files $F_j = \{f_{(1,j)}, \ldots, f_{(r,j)}\}$ ($F_j \subseteq F_{batch}$) that are distributed on a subset DH_j of DH. Specifically, for each file $f_{(p,j)} \in F_j$ ($p \in \{1, \ldots, r\}$), DH_j is the set of data hosts, on which $f_{(p,j)}$ is replicated, and from which it is available. We assume that each data host can serve multiple data files at a time and data

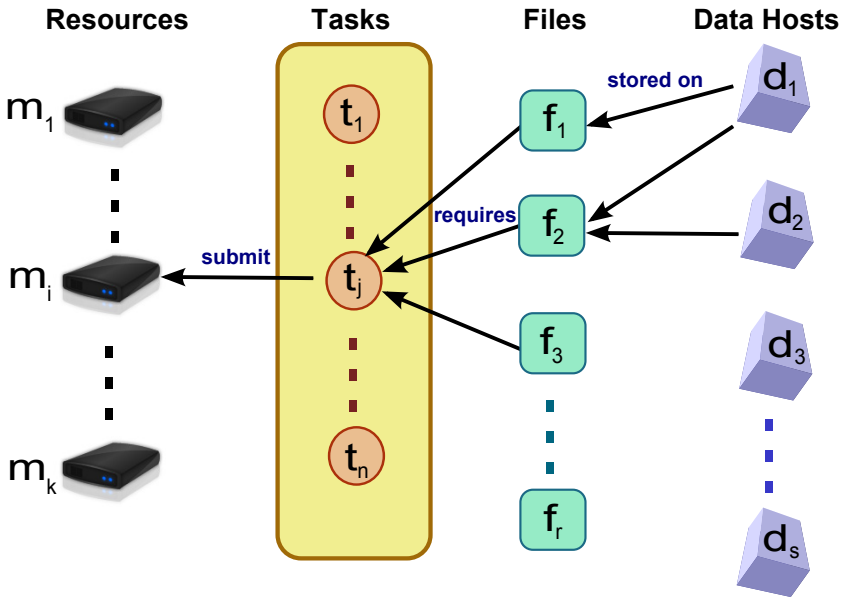

Fig. 3. Data-aware meta-task grid scheduling problem

replication is *a priori* defined as a separate replication process that may take into consideration various factors such as locality of access, load on the data-host and available storage space.

The computational power in the grid system can be characterized by its processing speed expressed by a clock frequency or by its computing capacity specified in MIPS (Million Instructions Per Second). The computing capacity of the resources available for processing a given batch is defined by a *computing capacity vector* $CC_{batch} = [cc_1, \ldots, cc_m]$, in which cc_i denotes the computing capacity of machine i. The estimation of the prior load of each machine from M_{batch} can be represented by a *ready times vector* $ready\_times_{(batch)} = [ready_1, \ldots, ready_m]$.

3.2 Data-Aware Task Execution Time Model

The data-aware task execution time model presented here follows an *Expected Time to Compute (ETC)* matrix model [1], in which an ETC matrix is defined, $ETC = [ETC[j][i]]_{n \times m}$ where $ETC[j][i]$ is an expected (estimated) time needed for the computing the task t_j on machine m_i. The workload and computing capacity parameters for tasks and machines are generated by using the Gamma probability distribution for the expression of tasks and machines heterogeneities in the grid system. The elements of the ETC matrix can be computed as the ratio of the coordinates of WL and CC vectors, i.e.:

$$ETC[j][i] = \frac{wl_j}{cc_i}. \tag{1}$$

The expected execution time of the task t_j in Eq. (1) depends on the processing speed of the machine m_i. However, for the successful task execution we need to include in the model the time needed for data transfer. For each data file $f_{(p,j)} \in F_j$ $(p \in \{1, \ldots, r\})$, the time required to transfer $f_{(p,j)}$ from the data host $dh_{(p,j)} \in D_j$ to i, denoted TT, is defined by the following formulae:

$$TT[i][j][f_{(p,j)}] = response\_time(dh_{(p,j)}) + \frac{Size\left[f_{(p,j)}\right]}{B(dh_{(p,j)}, i)} \tag{2}$$

where $response\_time(dh_{(p,j)})$ is the difference between the time when the request was made to $dh_{(p,j)}$ and the time when the first byte of the data file $f_{(p,j)}$ is received at machine m_i. This is an increasing function of the load on the data host. We denote by $B(dh_{(p,j)}, i)$ in Eq. (2) the bandwidth of the logical link between $dh_{(p,j)}$ and m_i[1] The estimated completion time for the task t_j on machine m_i, $completion[j][i]$, is the wall-clock time taken for the task from its submission till completion and is a function of computing and transmission times specified in Eq. (1) and (2). The impact of the data transfer time on the task completion time depends on the mode, in which the data files are processed by the task. The are two main such scenarios presented briefly in Fig. 4 (see also [45]).

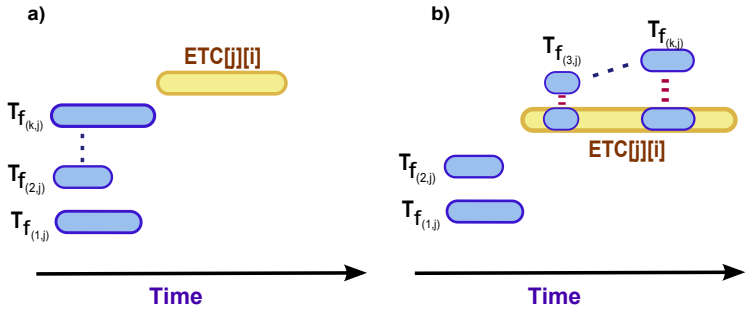

Fig. 4. Two variants of estimation of completion time of task t_j on machine m_i with the assumption of k data files needed for the task execution

In Fig. 4, for convenience, we denote the times for transferring the files $f_{(1,j)}, f_{(2,j)}, \ldots, f_{(k,j)}$ by $T_{f_{(1,j)}}, T_{f_{(2,j)}}, \ldots, T_{f_{(k,j)}}$, respectively. In the first scenario presented in Fig. 4(a) the data files needed for the task execution are transferred in parallel *before* the task execution. The number of simultaneous

[1] The physical network between the data hosts and resources consists of several entities such as routers, switches, links and hubs. However, the model presented here abstracts the physical network to consider just a logical network topology where each machine is connected to every data host by a distinct network link. Thus the bandwidth of the logical link between data host and machine is the bottleneck bandwidth of the actual physical network between them.

data transfers determines the bandwidth available for each transfer. Thus, the time of completion of the task t_j on machine m_i can be calculated by using the following formulae:

$$completion[j][i] = \max_{f_{(p,j)} \in F_j} TT[i][j][f_{(p,j)}] + ETC[j][i]. \tag{3}$$

On the other hand, Fig. 4(b) represents the idea of the second scenario, in which some of the data files are transferred completely prior to the task execution and the rest are accessed as streams during the execution. In this case, the transfer times of the streamed data files are masked by the computation time of the task, and, in the result, increase this computation time. The completion time of the task t_j on machine m_i can be calculated in the following way:

$$\begin{aligned} &completion[j][i] = \max_{f_{(p,j)} \in \widehat{F_j}} TT[i][j][f_{(p,j)}] + \\ &+ \sum_{f_{(l,j)} \in [F_j \setminus \widehat{F_j}]} TT[i][j][f_{(l,j)}] ETC[j][i]. \end{aligned} \tag{4}$$

where $\widehat{F_j}$ denotes a set of data files which are transferred prior the task execution.

We consider the data hosts as the data storage centers separated from the computing resources in order to make the system adaptable to various scheduling scenarios. Of course, in particular cases we can assume that each computing resource has its own data storage module [14], [20]. In such a case the internal data transfer times should be rather low and can be omitted in the analysis. However, for a fair estimation of the data transfers from the other computing sources there is a need in fact to decouple the data storage module from the computing module in the resource architecture [15], [22], [24]. The scalability and effectiveness of the whole system depends strongly on the replication mechanism and the resource data storage and computation capacities, [30], [23], [25], [21], which in some cases can be the main barrier in the schedulers' performance improvement [26], [27].

3.3 Scheduling Phases and Objectives

Scheduling phases in the data-ware scheduling are similar to Grid scheduling without data sets, but now it is assumed that Grid information services include also services for replicas such as replica management, discovery besides file transfer capabilities. These phases can be resumed as follows:

1. Get the information on available resources ;
2. Get the information on pending tasks ;
3. Get the information on data hosts where data files for tasks completion are required;
4. Prepare a batch of tasks and compute a schedule for that batch on available machines and data hosts;
5. Allocate tasks;
6. Monitor (failed tasks are re-scheduled).

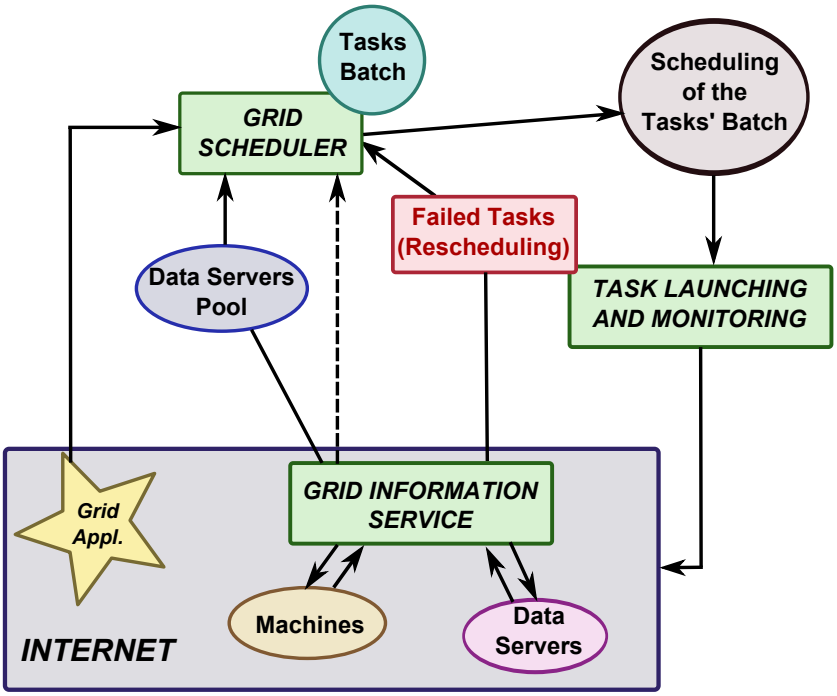

Fig. 5. Phases of the data-aware batch scheduler

These steps can be graphically represented as in Fig. 5.

The main objectives in data-aware scheduling are similar to the objectives formulated for grid scheduling without data files and include minimization of completion time, makespan, flowtime, etc.:

- Minimizing completion time of the task batch:

$$\sum_{t_j \in N_{batch}; m_i \in M_{batch}} completion[t_j][m_i]$$

  where $completion[t_j][m_i]$ is defined as in Eq. 4).
- Minimizing makespan:

$$\min_{Sched} \max_{m_i \in M_{batch}} completion[m_i]$$

  where $completion[m_i]$ is computed as the sum of completion times of tasks assigned to machine m_i.

Additionally, there are objectives related to the data-aware nature of the scheduling, such as access cost, response time, optimized *QoS*, etc. For example access cost an be computed as a weighted sum of reading cost and writing cost. Minimizing access cost affects directly the task turnaround time.

3.4 Strategies for Enhancing Data-Aware Schedulers

Several techniques can be used to reduce the transmission and access time in data-aware scheduling. As mentioned earlier, replication is a primary technique in this regard, which increases data availability, and therefore, increases scheduler's reliability. Another useful technique is that of parallel downloading of replicated data. Due to the dynamics of Grid systems, instead of replicating full data files, chunks of data files are replicated, which can further downloaded in parallel from different data hosts (see e.g. [52]).

In a similar vein, techniques used in P2P networks for downloading files can be used within the data-aware scheduling framework. The idea is that we could defined a virtual overlay on top of the Grid system by defining neighboring relations among computing sites and data hosts if computing sites contain replicas of data fragments for execution of a task assigned to the computing site. Then, we can formulate an optimization problem consisting in finding a subset of peer neighbors of the computing site from where to download/receive the data fragments [39]. The problem can be formally defined as follows.

Definition 1 (Neighbor-selection problem). *A neighbor-selection problem in P2P networks problem can be defined as* $\prod = (N, C, M, F, s)$*, in which* N *is the number of peers,* C *is the entire collection of content fragments,* M *is the maximum number of the available online peers,* F *is a single objective to optimize the number of swap fragments, or multi-objective to optimize the number of swap fragments, and to minimize the downloading cost;* s *denotes the environment constraints. The key components are operations, machines and data-hosts.*

The near-optimal resolution of this problem [39] can be used at the scheduling phase of selecting data hosts from where to get the data need for completion of the tasks in the batch.

4 Resolution Methods

4.1 Ad Hoc Methods

Ad hoc heuristics are simple procedures that need not to find even near-optimal solutions but are very fast and easy to implement. We briefly mention here some ad hoc heuristics for data-aware scheduling. An exhaustive list presented for Grid scheduling without data requirements can be found in [50] and [51].

MinMin Heuristic. In [45] the authors propose an extension of MinMin and Sufferage heuristics. In this extension they take into account the distributed data requirements of the target application model. The basic idea of the modified MinMin heuristic is to match in the beginning the meta-task to a resource set that guarantees the minimal completion time for some task in a batch. This is produced through special matching heuristics. They define Set Covering Problem (SCP) Tree Search (see also [4]), Greedy Selection, Compute-First or Exhaustive Search heuristics, which allow to select an appropriate combination of data

hosts and a compute resource that the total completion time for a given component of meta-task is minimized[2]. Then, the task with the optimal (minimal) completion time in the present allocation is assigned to the compute resource. This task is then removed from the batch structure. As task assignment changes the availability of the compute resource with respect to the number of available slots/processors, the resource information is updated and the process is repeated until all the components of the meta-task have been allocated to some resource set. When a task is scheduled for execution on a compute resource, all required data files which are not available local to the resource, are transferred to the resource prior to execution. These data files become replicas that can be used by following meta-task components.

Sufferage Heuristic. The motivation behind the modified Sufferage heuristic is to allocate a resource set to a meta-task that would be disadvantaged the most (or "suffer" the most) if that resource set were not allocated to it. This is determined through a sufferage value computed as the difference between the second best and the best value of the completion time for the meta-task components. For each task, the resource that offers the least value of the completion time is determined through the same mechanisms as that in MinMin. Then another resource with the second minimal completion time is selected to establish the 'sufferage' value for a given task. The selection of the compute resource determines both the task execution time and the data transfer times. After determining the sufferage value for each task, the task with the largest sufferage value is then selected and assigned to its chosen resource. The rest of the heuristic including dispatching and updating of compute resource and data host information proceeds in the same manner as MinMin.

Other interesting ad hoc methods are Shortest Turnaround Time (STT), Least Relative Load (LRL) and Data Present (DP) (see e.g. [52]).

4.2 Meta-heuristic Methods

Dealing with the many constraints and optimization criteria in a dynamic environment scheduling of data-intensive applications in Computational Grid remains very complex and computationally hard problem [31], [36]. The significance of meta-heuristic approaches for designing efficient grid schedulers can be explained as follows (see also [49]:

Meta-heuristics Are Well Understood. Meta-heuristics have been studied for a large number of optimization problems, from theoretical, practical and experimental perspectives.

Computing Near-Optimal Solutions. In the dynamic grid environment, it is usually impossible to generate the optimal schedules. This is so due to the fact that grid schedulers run as long as the grid system exists and thus the

[2] Referred to as the Minimum Resource Set (MRS) problem.

performance is measured not only for particular applications but also in the long run. Therefore, in such situation meta-heuristics are among best candidates to cope with grid scheduling.

Dealing with Multi-objective Nature. Meta-heuristics have proven to efficiently solve the complex multi-objective optimization problems.

Well Designed for Periodic and Batch Scheduling. In the case of periodic scheduling the resource provisioning can be done with no strong time restrictions. This means that we can run meta-heuristic-based schedulers for longer execution times and significantly increase the quality of generated schedules. In batch scheduling, we could run the meta-heuristic-based scheduler for the time interval comprised within two successive batches activations.

Hybridization with other Approaches. Meta-heuristics can be easily hybridized with other approaches, which is useful to make grid schedulers better adapted to various grid scenarios, grid types, specific types of applications, etc.

Designing Robust Grid Schedulers. The dynamics of the grid environment directly impacts on the performance of the grid scheduler. A robustness in grid scheduling is a key issue in high-quality resource allocation in the case of frequent changes in the system's states.

The heuristic scheduling methods are usually classified into three main groups, namely calculus-based (greedy algorithms and ad-Hoc methods), stochastic (guided and non-guided methods) and enumerative methods (dynamic programming and branch-and-bound algorithm). The most popular and efficient methods in grid scheduling are ad-hoc, local search-based and population-based methods. A simple taxonomy of the heuristic schedulers is presented in Fig. 6.

Each of this scheduler can be adapted to the data grid scheduling by adding some extra tasks-data files matching procedures.

5 Scheduling Challenges

The data-aware scheduling in grid systems becomes even more challenging when we consider the following key challenges:

- **Scheduling Policy:** According to the selection of data-hosts and mapping of resources, the optimization criteria such makespan and flowtime may change significantly.
- **Storage Constraints:** Only limited storage capacity is available at resources. As the tasks get executed, the data produced should be either deleted or moved. Storage aware resource scheduling problem is a major area of research. Data providence should be associated with scheduling policy.
- **Replication Policy:** The availability of replicas of data and their locality heavily depends on the replication policy. For expressing the system dynamics a dynamic replication policy that can balance the replicas among data hosts should be formulated.

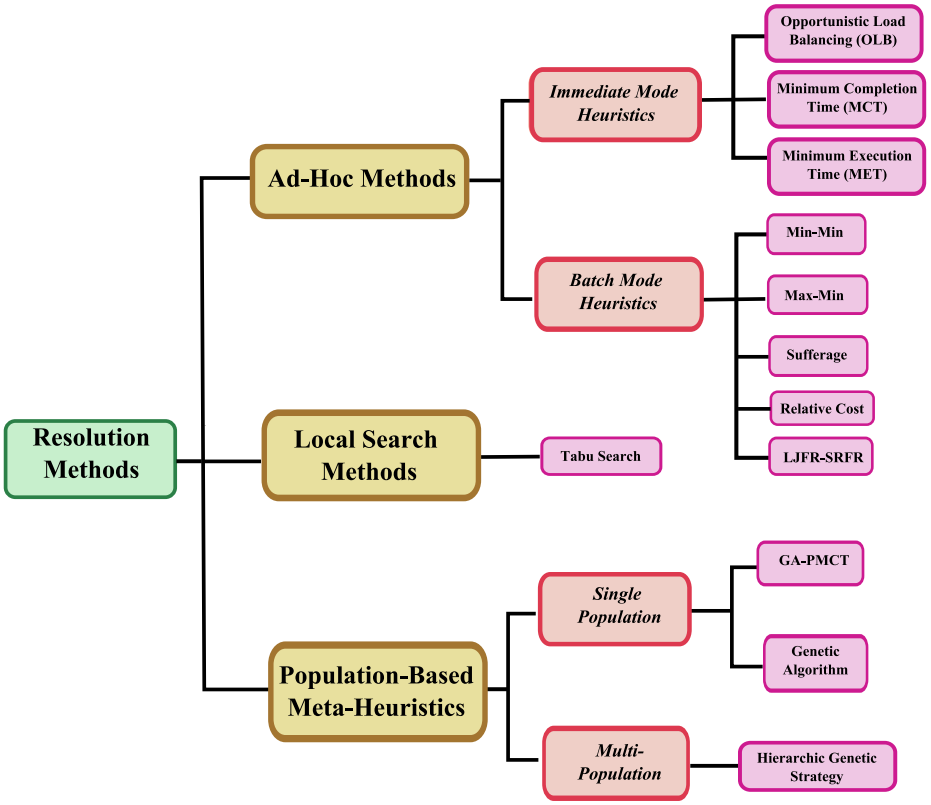

Fig. 6. Heuristic resolution methods taxonomy

- **Resource Provisioning:** In real life approaches there is a need to establish a Service-Level-Agreement (SLA) based advance reservation to circumvent the sudden scarcity of resources, which can guarantee the resource access at the scheduled time.
- **User QoS.** Users Quality of Service (QoS) criteria such as budget and deadline constraints may also be taken into account.
- **Security.** All operations in a Data Grid should be mediated by a security layer that handles authentication of entities and ensures conduct of only authorized operations. Additionally the grid users can specify their own criteria for secure task allocation at grid resources.

6 Conclusions and Future Work

In this paper we have presented a simple taxonomy of data-aware scheduling based on a set of requirements such data transmission, decoupling of data from processing, data replication and data access. By considering these requirements

a family of data-aware scheduling problems can be defined, whose resolution can be very useful to design efficient data-aware schedulers. We have focused on the Data-aware Independent Batch Scheduling for which we have formalized the transmission time, in a way that it can be easily integrated into classical optimization objectives of grid scheduling. This is particularly useful as known optimization formulation and resolution methods can be applied to the data-aware scheduling with transmission times. We have also briefly discussed the different resolution methods (including ad hoc and meta-heuristics methods) to cope in practice with the complexities of the problem.

References

1. Ali, S., Siegel, H.J., Maheswaran, M., Hensgen, D.: Task execution time modeling for heterogeneous computing systems. In: Proceedings of Heterogeneous Computing Workshop, pp. 185–199 (2000)
2. Buyya, R., Murshed, M., Abramson, D., Venugopal, S.: Scheduling parameter sweep applications on global Grids: a deadline and budget constrained cost-time optimization algorithm. Softw. Pract. Exper. 35(5), 491–512 (2005)
3. Casanova, H., Obertelli, G., Berman, F., Wolski, R.: The AppLeS parameter sweep template: user-level middleware for the grid. In: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing (CDROM) (Supercomputing 2000). IEEE Computer Society, Washington, DC (2000)
4. Christofides, N.: Independent and Dominating Sets–The Set Covering Problem. In: Graph Theory: An Algorithmic Approach, pp. 30–57 (1975) ISBN: 012 1743350 0
5. Foster, I., Karonis, N.: A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems. In: Proceedings of the IEEE/ACM SuperComputing Conference 1998 (SC 1998), San Jose, CA, USA, IEEE CS Press, Los Alamitos (1998)
6. Hockauf, R., Karl, W., Leberecht, M., Oberhuber, M., Wagner, M.: Exploiting Spatial and Temporal Locality of Accesses: A New Hardware-Based Monitoring Approach for DSM Systems. In: Pritchard, D., Reeve, J.S. (eds.) Euro-Par 1998. LNCS, vol. 1470, pp. 206–215. Springer, Heidelberg (1998)
7. Kliazovich, D., Bouvry, P., Khan, S.U.: DENS: Data Center Energy-Efficient Network-Aware Scheduling. In: ACM/IEEE International Conference on Green Computing and Communications (GreenCom), Hangzhou, China, pp. 69–75 (December 2010)
8. Kliazovich, D., Bouvry, P., Audzevich, Y., Khan, S.U.: GreenCloud: A Packet-level Simulator of Energy-aware Cloud Computing Data Centers. In: Proc. of the 53rd IEEE Global Communications Conference (Globecom), Miami, FL, USA (December 2010)
9. Khan, S.U., Ahmad, I.: A Pure Nash Equilibrium based Game Theoretical Method for Data Replication across Multiple Servers. IEEE Transactions on Knowledge and Data Engineering 21(4), 537–553 (2009)
10. Khan, S.U., Ardil, C.: A Weighted Sum Technique for the Joint Optimization of Performance and Power Consumption in Data Centers. International Journal of Electrical, Computer, and Systems Engineering 3(1), 35–40 (2009)

11. Khan, S.U.: A Multi-Objective Programming Approach for Resource Allocation in Data Centers. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, NV, USA, pp. 152–158 (July 2009)
12. Khan, S.U.: On a Game Theoretical Methodology for Data Replication in Ad Hoc Networks. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, NV, USA, pp. 232–238 (July 2009)
13. Khan, S.U.: A Frugal Auction Technique for Data Replication in Large Distributed Computing Systems. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, NV, USA, pp. 17–23 (July 2009)
14. Khan, S.U., Ardil, C.: A Fast Replica Placement Methodology for Large-scale Distributed Computing Systems. In: International Conference on Parallel and Distributed Computing Systems (ICPDCS), Oslo, Norway, pp. 121–127 (July 2009)
15. Khan, S.U., Ardil, C.: A Competitive Replica Placement Methodology for Ad Hoc Networks. In: International Conference on Parallel and Distributed Computing Systems (ICPDCS), Oslo, Norway, pp. 128–133 (July 2009)
16. Khan, S.U., Ardil, C.: On the Joint Optimization of Performance and Power Consumption in Data Centers. In: International Conference on Distributed, High-Performance and Grid Computing (DHPGC), Singapore, pp. 660–666 (August 2009)
17. Khan, S.U.: A Self-adaptive Weighted Sum Technique for the Joint Optimization of Performance and Power Consumption in Data Centers. In: 22nd International Conference on Parallel and Distributed Computing and Communication Systems (PDCCS), Louisville, KY, USA, pp. 13–18 (September 2009)
18. Khan, S.U.: A Goal Programming Approach for the Joint Optimization of Energy Consumption and Response Time in Computational Grids. In: Proc. of the 28th IEEE International Performance Computing and Communications Conference (IPCCC), Phoenix, AZ, USA, pp. 410–417 (December 2009)
19. Khan, S.U., Ahmad, I.: Non-cooperative, Semi-cooperative, and Cooperative Games-based Grid Resource Allocation. In: Proc. of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rhodes Island, Greece (April 2006)
20. Khan, S.U., Ahmad, I.: Comparison and Analysis of Ten Static Heuristics-based Internet Data Replication Techniques. Journal of Parallel and Distributed Computing 68(2), 113–136 (2008)
21. Khan, S.U., Ahmad, I.: Discriminatory Algorithmic Mechanism Design Based WWW Content Replication. Informatica 31(1), 105–119 (2007)
22. Khan, S.U., Ahmad, I.: Game Theoretical Solutions for Data Replication in Distributed Computing Systems. In: Rajasekaran, S., Reif, J. (eds.) Handbook of Parallel Computing: Models, Algorithms, and Applications, vol. ch. 45. Chapman & Hall/CRC Press, Boca Raton (2007) ISBN: 1-584-88623-4
23. Khan, S.U., Ahmad, I.: A Semi-Distributed Axiomatic Game Theoretical Mechanism for Replicating Data Objects in Large Distributed Computing Systems. In: 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS), Long Beach, CA, USA (March 2007)
24. Khan, S.U., Ahmad, I.: Replicating Data Objects in Large-scale Distributed Computing Systems using Extended Vickery Auction. International Journal of Computational Intelligence 3(1), 14–22 (2006)

25. Khan, S.U., Ahmad, I.: Data Replication in Large Distributed Computing Systems using Supergames. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, NV, USA, pp. 38–44 (June 2006)
26. Khan, S.U., Ahmad, I.: A Pure Nash Equilibrium Guaranteeing Game Theoretical Replica Allocation Method for Reducing Web Access Time. In: 12th International Conference on Parallel and Distributed Systems (ICPADS), Minneapolis, MN, USA, pp. 169–176 (July 2006)
27. Khan, S.U., Ahmad, I.: A Powerful Direct Mechanism for Optimal WWW Content Replication. In: 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Denver, CO, USA (April 2005)
28. Khan, S.U., Ahmad, I.: Replicating Data Objects in Large Distributed Database Systems: An Axiomatic Game Theoretical Mechanism Design Approach. Distributed and Parallel Databases 28(2-3), 187–218 (2010)
29. Khan, S.U., Ahmad, I.: A Cooperative Game Theoretical Technique for Joint Optimization of Energy Consumption and Response Time in Computational Grids. IEEE Transactions on Parallel and Distributed Systems 20(3), 346–360 (2009)
30. Khan, S.U., Maciejewski, A.A., Siegel, H.J., Ahmad, I.: A Game Theoretical Data Replication Technique for Mobile Ad Hoc Networks. In: 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS), Miami, FL, USA (April 2008)
31. Kołodziej, J., Xhafa, F., Kolanko, Ł.: Hierarchic Genetic Scheduler of Independent Jobs in Computational Grid Environment. In: Otamendi, J., Bargieła, A., Montes, J.L., Doncel Pedrera, L.M. (eds.) Proc. of 23rd ECMS, Madrid, pp. 108–115. IEEE Press, Dudweiler (2009)
32. Kołodziej, J., Xhafa, F.: A Game-Theoretic and Hybrid Genetic meta-heuristic Model for Security-Assured Scheduling of Independent Jobs in Computational Grids. In: Proc. of CISIS 2010, pp. 93–100. IEEE Press, USA (2010)
33. Kołodziej, J., Xhafa, F., Bogdański, M.: Secure and task abortion aware GA-based hybrid metaheuristics for grid scheduling. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 526–535. Springer, Heidelberg (2010)
34. Kołodziej, J., Xhafa, F.: Meeting Security and User Behaviour Requirements in Grid Scheduling. Simulation Modelling Practice and Theory 19(1), 213–226 (2011), doi:10.1016/j.simpat.2010.06.007
35. Kołodziej, J., Xhafa, F.: Integration of Task Abortion and Security Requirements in GA-based Meta-Heuristics for Independent Batch Grid Scheduling. Computers and Mathematics with Applications (2011), doi: 10.1016/j.camwa.2011.07.038
36. Kołodziej, J., Xhafa, F.: Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population. Future Generation Computer Systems 27, 1035–1046 (2011), doi:10.1016/j.future.2011.04.011
37. Kołodziej, J., Khan, S.U., Xhafa, F.: Genetic Algorithms for Energy-aware Scheduling in Computational Grids. In: Proc. of the 6th IEEE International Conference on P2P, Parallel, Grid, Cloud, and Internet Computing (3PGCIC), Barcelona, Spain (October 2011)
38. Kosar, T., Balman, M.: A new paradigm: Data-aware scheduling in grid computing. Future Gener. Comput. Syst. 25(4), 406–413 (2009)
39. Liu, H., Abraham, A., Xhafa, F.: Peer-to-Peer Neighbor Selection Using Single and Multi-objective Population-Based Meta-heuristics. In: Xhafa, F., Abraham, A. (eds.) Metaheuristics for Scheduling in Distributed Computing Environments. SCI, vol. 146, pp. 323–340. Springer, Heidelberg (2008)

40. Liu, H., Orban, D.: GridBatch: Cloud Computing for Large-Scale Data-Intensive Batch Applications. In: 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID), pp. 295–305 (2008)
41. Pinel, F., Pecero, J.E., Bouvry, P., Khan, S.U.: A Two-Phase Heuristic for the Scheduling of Independent Tasks on Computational Grids. In: Proc. of ACM/IEEE/IFIP International Conference on High Performance Computing and Simulation (HPCS), Istanbul, Turkey (July 2011)
42. Ranganathan, K., Foster, I.: Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In: Proceedings of the 11th IEEE Symposium on High Performance Distributed Computing (HPDC), Edinburgh, Scotland. IEEE CS Press, Los Alamitos (2002)
43. Shatdal, A., Kant, C., Naughton, J.F.: Cache conscious algorithms for relational query processing. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB 1994), Santiago, Chile, pp. 510–521. Morgan Kaufmann Publishers, Inc., San Francisco (1994)
44. Valentini, G.L., Lassonde, W., Khan, S.U., Min-Allah, N., Madani, S.A., Li, J., Zhang, L., Wang, L., Ghani, N., Kołodziej, J., Li, H., Zomaya, A.Y., Xu, C.-Z., Balaji, P., Vishnu, A., Pinel, F., Pecero, J.P., Kliazovich, D., Bouvry, P.: An Overview of Energy Efficiency Techniques in Cluster Computing Systems. Cluster Computing (2011), doi:10.1007/s10586-011-0171-x
45. Venugopal, S., Buyya, R.: An SCP-based heuristic approach for scheduling distributed data-intensive applications on global grids. J. Parallel Distrib. Comput. 68, 471–487 (2008)
46. Venugopal, S., Buyya, R., Kotagiri, R.: A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing (2009)
47. Wang, L., Khan, S.U.: Review of Performance Metrics for Green Data Centers: A Taxonomy Study. Journal of Supercomputing, 1–18 (2011), doi:10.1007/s11227-011-0704-3
48. Wasson, G., Humprey, M.: Policy and enforcement in virtual organizations. In: Proceedings of the 4th International Workshop on Grid Computing, Phoenix, Arizona, IEEE CS Press, Los Alamitos (2003)
49. Xhafa, F., Abraham, A.: Computational models and heuristic methods for grid scheduling problems. Future Generation Computer Systems 26, 608–621 (2010)
50. Xhafa, F., Carretero, J., Barolli, L., Durresi, A.: Immediate Mode Scheduling in Grid Systems. International Journal of Web and Grid Services 3(2), 219–236 (2007)
51. Xhafa, F., Barolli, L., Durresi, A.: Batch Mode Schedulers for Grid Systems. International Journal of Web and Grid Services 3(1), 19–37 (2007)
52. Zhang, J., Lee, B., Tang, X., Yeo, C.: Impact of Parallel Download on Job Scheduling in Data Grid Environment. In: Proc. of the Seventh International Conference on Grid and Cooperative Computing, pp. 102–109 (2008)
53. Zeadally, S., Khan, S.U., Chilamkurti, N.: Energy-Efficient Networking: Past, Present, and Future. Journal of Supercomputing, 1–26 (2011), doi:10.1007/s11227-011-0632-2