Pertanika J. Sci. & Technol. 26 (2): 523 - 544 (2018)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Review Article

On Modelling Parallel Programmes for Static Mapping: A Comparative Study

Sina Zangbari Koohi[1], Nor Asilah Wati Abdul Hamid[1]*, Mohamed Othman[1] and Gafurjan Ibragimov[2]

[1]Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

[2]Department of Mathematics and Institute for Mathematical Research, Faculty of Science Universiti Putra Malaysia, University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia

ABSTRACT

Heterogeneous parallel architecture (HPA) are inherently more complicated than their homogeneous counterpart. HPAs allow composition of conventional processors, with specialised processors that target particular types of task. However, this makes mapping and scheduling even more complicated and difficult in parallel applications. Therefore, it is crucial to use a robust modelling approach that can capture all the critical characteristics of the application and facilitate the achieving of optimal mapping. In this study, we perform a concise theoretical analysis as well as a comparison of the existing modelling approaches of parallel applications. The theoretical perspective includes both formal concepts and mathematical definitions based on existing scholarly literature. The important characteristics, success factors and challenges of these modelling approaches have been compared and categorised. The results of the theoretical analysis and comparisons show that the existing modelling approaches still need improvement in parallel application modelling in many aspects such as covered metrics and heterogeneity of processors and networks. Moreover, the results assist us to introduce a new approach, which improves the quality of mapping by taking heterogeneity in action and covering more metrics that help to justify the results in a more accurate way.

Keywords: Heterogeneous parallel architectures, mapping, parallel application modelling, scheduling

Article history:
Received: 7 June 2017
Accepted: 5 December 2017

E-mail addresses:
zangbari@gmail.com (Sina Zangbari Koohi)
asila@upm.edu.my (Nor Asilah Wati Abdul Hamid)
mothman@upm.edu.my (Mohamed Othman)
ibragimov@upm.edu.my (Gafurjan Ibragimov)
*Corresponding Author

ISSN: 0128-7680

INTRODUCTION

In the last two decades, many distributed high-performance computers with thousands or millions of processing units have been in use. The emphasis on distributed and parallel

computers is on shorter execution time, decreasing energy consumption, and reducing idle time of resources. By means of efficient mapping and scheduling, a substantial improvement in these issues can be achieved (Lastovetsky & Manumachu, 2017; Tekinerdogan & Arkin, 2012). Mapping problems is the assignment of the processors to processes and communication links to data elements that are exchanged between the processes. Each processor in a distributed architecture usually manages more than one process in a single application. Managing the order of execution of these processes on each processor is known as the scheduling problem. These two crucial problems in parallel computing fall into NP-hard categories (Kumar, Grama, Gupta, & Karypis, 1994; Lewis & El-Rewini, 1992; Sarkar, 1989; Ullman, 1975).

Mapping and scheduling problems becomes more complex when it involves heterogeneous parallel architecture because this architecture is more complicated than homogeneous parallel architecture (Rico-Gallego, Lastovetsky, & Diaz-Martin, 2017). The first step in solving mapping and scheduling problems lies in the modelling phase. In computer science, to represent real entities, such as processing architecture, computational models are used. A computational model is a simplified version of these entities, where crucial characteristics are captured and implementation details are ignored (Leopold, 2001). The information captured from the programme to be executed is modelled by programme graphs or hyper-graphs.

There are several modelling approaches studied in the literature to capture the behaviour of parallel applications such as PRAM (Kumar et al., 1994), LogP (Culler et al., 1996), BSP (Bulk-Synchronous Parallel) (Valiant, 1990), TIG (Task Interaction Graph) (Long & Clarke, 1989), TPG (Task Precedence Graph) (Kasahara & Narita, 1985), TTIG (Task Temporal Interaction Graph) (Roig, Ripoll, & Guirado, 2007; Roig, Ripoll, Senar, Guirado, & Luque, 2000, 2002), TTIGHa (Temporal Task Interaction Graph in Heterogeneous Architecture) (De Giusti, Chichizola, Naiouf, Ripoll, & De Giusti, 2007), MPAHA (Model on Parallel Algorithms on Heterogeneous Architectures) (De Giusti, Naiouf, Chichizola, Luque Fadón, & De Giusti, 2009) and the hypergraph-based model proposed in the UMPa scheduling algorithm (Deveci, 2015; Deveci, Kaya, Ucar, & Catalyurek, 2015).

The varieties in modelling approaches motivated this research, which focused on an analysis and comparative study of frameworks, similarities, differences, characteristics and principles. The comparison will assist in determining the suitability, success factors and challenges of each modelling approach.

The remaining sections are organised as follows. In Section 2, each model is discussed in detail and its mathematical structure is shown. In Section 3, a comparative study of these models is presented and discussed. In Section 4, the new model is proposed as a future work and in Section 5, the study is concluded.

MODELLING STRUCTURES

In this section, modelling approaches are introduced. In the next section, a comparative study is presented based on these modelling approaches.

Task Interaction Graph (TIG)

The Task Interaction Graph (TIG) model isolates an application into the maximum number of sequential blocks (tasks) that are connected by edges. These edges represent the interactions between the blocks. The TIG model uses an undirected graph to abstract the application into the model. This model is an undirected graph $G = (V, E, c, w)$ (Long & Clarke, 1989), where:

- V is a set of nodes. Each node in this set represents a task T_i.
- E is a set of edges. Each edge in this set represents communication between tasks.
- $c(T_i)$ is a positive cost associated with task, T_i. This cost represents the computation time of the task, T_i.
- $w(T_i, T_j)$ is a non-negative weight associated to the edge between T_i and T_j. This weight represents the total communication volume between two edges.

To illustrate the model, the sample application exhibited in Figure 1 is considered. This sample application consists of three tasks communicating with each other. Each curve in this figure is a task. The corresponding pseudo-code of this sample application is given in Figure 2.

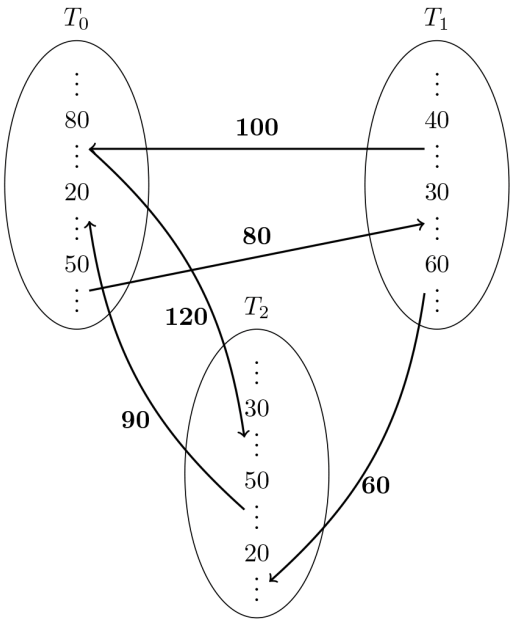

Figure 1. Temporal flow graph of sample application

Pseudo-code 1 a sample application

```
1: if Task = T0 then
2:     ... BC with CT=80 ...
3:     RECV(d1, 100, T1);
4:     SND(d2, 120, T2);
5:     ... BC with CT=20 ...
6:     RECV(d3, 90, T2);
7:     ... BC with CT=50 ...
8:     SND(d4, 80, T1);
9: end if
10: if Task = T1 then
11:     ... BC with CT=40 ...
12:     SND(d1, 100, T0);
13:     ... BC with CT=30 ...
14:     RECV(d4, 80, T0);
15:     ... BC with CT=60 ...
16:     SND(d5, 60, T2);
17: end if
18: if Task = T2 then
19:     ... BC with CT=30 ...
20:     RECV(d2, 120, T0);
21:     ... BC with CT=50 ...
22:     SND(d3, 90, T0);
23:     ... BC with CT=20 ...
24:     RECV(d5, 60, T1);
25: end if
```

Figure 2. Pseudo-code 1 sample application

In this pseudo-code, BC stands for block of computation and CT stands for computation time, while $\mathrm{SND}(d, n, T_i)$ is a communication command that sends the data element d, with volume n, to the task T_i. $RECV(d, n, T_i)$ is another communication command that receives data element d with volume n from task T_i.

Holding the above example, the TIG model for this application is formed. Figure 3 illustrates the TIG model of the sample application.

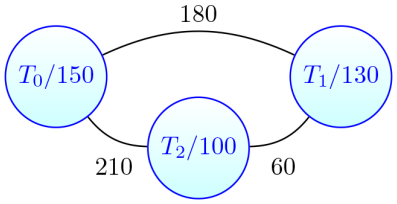

Figure 3. TIG graph for sample application

In the early years of heterogeneous parallel computing, this model was used by many researchers and in different studies, such as by El-Rewini, Lewis and Ali (1994), Hui and Chanson (1997), Kalinov and Klimov (2005), Phinjaroenphan and Bevinakoppa (2004), Sadayappan, Ercal and Ramanujam (1990), and Sanyal and Das (2005). However, with the growth of parallel computers and the appearance of more complex parallel applications, the drawbacks of this model were exposed.

Task Precedence Graph (TPG)

Liu, Shi, Lu and Mao (2007) proposed a new model named the Task Precedence Graph (TPG). The TPG is a refined version of the TIG model that attempts to capture the precedence relations between tasks. The authors used a directed graph to model the parallel applications and their communications and relations. Besides the details captured in the model, the TIG model records the predecessor and successor relations in its graph.

The TPG is a directed graph $G = (PV, PE$ (Kasahara & Narita, 1985), where:

- PV is a set of nodes. Each node pv_i represents a task T_i in the application.
- PE is a set of edges. Each edge e_{ij} represents a direct drive relation between two communicating nodes pv_i and pv_j.

$$PE = \{e_{ij} | e_{ij} = Edge(pv_i, pv_j); pv_j \in Successor(pv_i); i, j \in \{0,1, \dots, n\}\}$$

In this model, each edge shows both communication volume and the precedence relation between the tasks. Like TIG, in this model, each node has a cost representing its computation time and each edge has a weight, representing its communication volume. Considering the sample application illustrated in Figure 1, the TPG model for this application is shown in Figure 4.

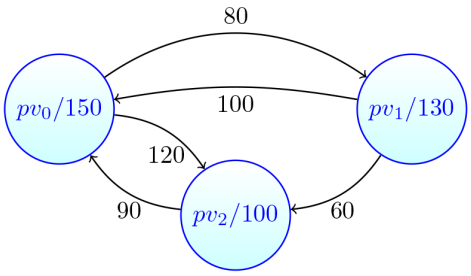

Figure 4. TPG graph for sample application

Task Temporal Interaction Graph (TTIG)

Roig et al. (2007, 2000, 2002) proposed a new model named Task Temporal Interaction Graph (TTIG), which is a refined combination of the TIG and TPG (Roig et al., 2000). In this model, a new parameter, the degree of parallelism, is proposed. This new parameter adds the ability to model recording the potential parallelism for communicating tasks with an arbitrary task interaction pattern (Roig et al., 2007).

The TTIG model for parallel applications is defined as a directed graph $G = (V, E)$ with three maps w, c, p, where:

- V is a set of nodes. Each node in this set represents a task T_i.
- E is a set of directed edges. Each edge in this set represents a temporal relation between two nodes.
- $w: V \rightarrow N$ is a function that assigns a non-negative computation time $w(T_i)$ to each task in V.
- $c: V \rightarrow N$ is a function that assigns a non-negative communication cost to each edge. $c(T_i, T_j)$ is the total volume of messages being transferred between T_i and T_j.
- $p: E \rightarrow [0,1]$ is a function that assigns a normalised index to each edge. Without considering any communication cost or any other dependencies, $p(T_i, T_j)$ is the maximum degree of parallelism that two tasks T_i and T_j can obtain during parallel execution.

To set the TTIG graph for an application, four different steps should be taken:

1. Acquiring the Temporal Flow Graph
2. Calculating the task execution and communication costs
3. Calculating the degree of parallelism
4. Forming the Temporal Task Interaction Graph (TTIG)

For more details about the degree of parallelism and the TTIG model, refer to Roig et al. (2007, 2000, 2002).

Based on the previously explained steps, the TTIG graph for the sample application introduced in Figure 1 will appear as given below (Figure 5).

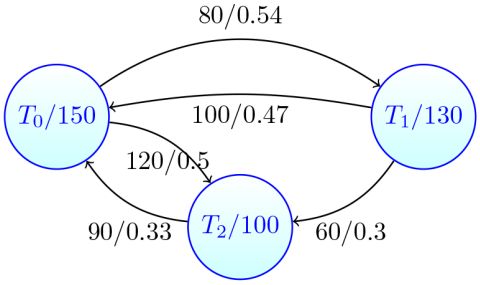

Figure 5. TTIG graph for sample application

Temporal Task Interaction Graph in Heterogeneous Architectures (TTIGHa)

Temporal Task Interaction Graph in Heterogeneous Architectures (TTIGHa) is a refined version of the TTIG model, where the heterogeneity of processors and heterogeneity of communication media in a distributed architecture are taken into account.

- Formally, TTIGHa is a directed graph $G = (V, E, T_p, T_c)$, where:
- V is a set of nodes. Each node in this set represents a task T_i.
- E is a set of directed edges. Each edges in this set represents communication between tasks.
- T_p is a set of processors. For each type of processor in the target distributed machine, there is an element in this set exhibiting the corresponding processor type.
- T_c is a set of communication media. For each type of communication medium in the target distributed machine, there is an element in this set exhibiting corresponding medium type, startup time and transfer time for one byte of the data element.

This model gathers more information about the application. First, the model determines the execution time of each task on every type of processor. $w_p(T_i)$ is the execution time of the task T_i on processor type p. A $\#m \times \#m$ dimension matrix named C is built for each edge (T_i, T_j), where $\#m$ is the number of processor types in the distributed architecture ($\#m = |T_p|$). For each edge $(T_i, T_j) \in E$, the value of $C_{sd}(T_i, T_j)$ is the time needed for transferring one byte of data from task $T_i \in V$ on processor type $s \in T_p$ to task $T_i \in V$ on processor type $d \in T_p$. A $\#m \times \#m$ dimension matrix named P is formed, where $P_{sd}(T_i, T_j)$ is the degree of parallelism between tasks $T_i \in V$ on processor type $s \in T_p$ and $T_j \in V$ on processor type $d \in T_p$. For more information about adjusting these matrices, refer to De Giusti et al. (2007).

After the formal definition of the TTIGHa model, the sample application presented in Figure 1 and Pseudo-code 1 is modelled using this approach. As this model needs information regarding the distributed architecture that the application will run on, a distributed machine with two types of processor, p_1 and p_2, is considered, where the execution speed of processor p_2 is half that of processor p_1, i.e. if task t takes e seconds to run on processor p_1, it will take $2 \times e$ seconds on processor p_2. Furthermore, it is considered that this machine utilises only one type of communication medium with a startup time of 1 second and communication speed of 0.2 seconds per byte. Considering this distributed machine, the TTIGHa model for the sample application illustrated in Figure 1 is as shown in Figure 6.

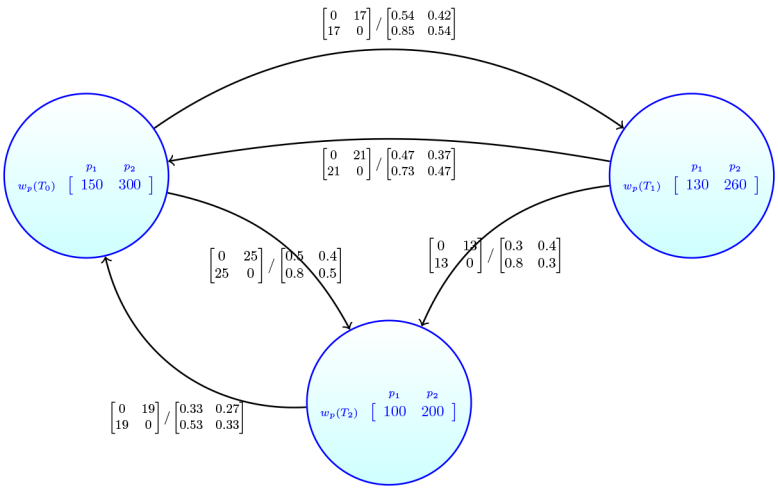

Figure 6. TTIGHa graph for sample application

Model of Parallel Algorithms on Heterogeneous Architectures (MPAHA)

The model of Parallel Algorithms in Heterogeneous Architectures (MPAHA) takes the heterogeneity of processors and networks into account, but in a clearer and simpler way. This model tries to arrange the information captured from the application in a systematic way.

According to the definition, the MPAHA is a directed graph $G = (V, E)$, where:

- V is a set of nodes. Each node in this set represents a task T_i.
- E is a set of edges. Each edge in this set represents communication between tasks.

In this model, besides the tasks and the communication between them, two other aspects of the application are recorded in the graph. The first one is the execution time of the computation phases (in this model called subtasks) for each task on different types of processor. The other one is the communication volume between computation phases (subtasks). Each task $T_i \in V$ consists of multiple subtasks I_i, where there is no communication within each subtask i.e. communication takes place between subtasks. The subtasks for each task $V_i(I_j, p)$ is defined as the computation time of the subtask I_j in task $T_i \in V$ on processor type p. For each task $T_i \in V$, the corresponding matrix $V_i(I, p)$ is formed as assigned.

The other parameter is the communication volume. $E_{ij}(I_m, I_n)$ is set as the communication volume between the subtask I_m in the task T_i ($I_m \in T_i$) to subtask I_n in task T_j ($I_n \in T_j$). Then, the matrix $E(I_m, I_n)$ is assigned to the communication E_{ij} (De Giusti et al., 2007). As in previous approaches, the sample application in Figure 1 is modelled using this modelling method. The result is shown in Figure 7.

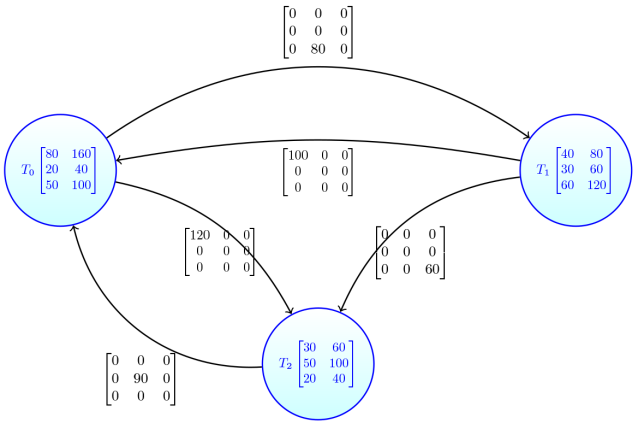

Figure 7. MPAHA graph for sample application

Hypergraph Based Modelling: UMPa

All the models discussed in the previous sections use the graph theory as their basic tool to capture the parallel application's details. However, the Hypergraph Based Modelling: UMPa proposed by Deveci (2015) and Deveci et al. (2015) uses a new structure named hypergraph. Hypergraph-based modelling approaches are more flexible in modelling parallel heterogeneous structures than graph versions.

UMPa utilises a directed hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where:

- $\mathcal{V}$ is a set of nodes. Each node in this set represents a task $t_i \in \mathcal{T}$ or data exchange in communication $d_i \in \mathcal{D}$.
- $\mathcal{N}$ is a set of nets. Each net in this set connects the task that starts the communication $t \in \mathcal{T}$ (producer), the data element being exchanged $d_i \in \mathcal{D}$ and the task receiving the data element $t_j \in \mathcal{T}$ (consumer).

A weight to each node and a cost to each net are assigned. The weight $w[v_i]$ for the node $v_i \in \mathcal{V}$ is the computation time of the task $t_i \in \mathcal{T}$ if this node denotes a task and 0 if this node denotes a data element:

$$w[v_i] = \begin{cases} exec(t_i) & v_i = t_i \in \mathcal{T} \\ 0 & v_i = d_i \in \mathcal{D} \end{cases}$$

where, $exec(t_i)$ is the computation time of the task t_i.

This weight has two functions. While it shows the computation time for each task, it also serves as a factor to distinguish between task nodes and communication nodes.

Another function, c, which assigns a cost $c[n_i]$ to each net $n_i \in \mathcal{N}$, provides the value of $c[n_i]$ for each communication between nodes equal to the communication volume of the data element being exchanged.

$$c[n_i] = size(d_i)$$

where, $size(d_i)$ is the communication volume of the data element $d_i \in \mathcal{D}$ (Deveci, 2015; Deveci et al., 2015). The model hypergraph for the sample application in Figure 1 using the UMPa approach is shown in Figure 8.

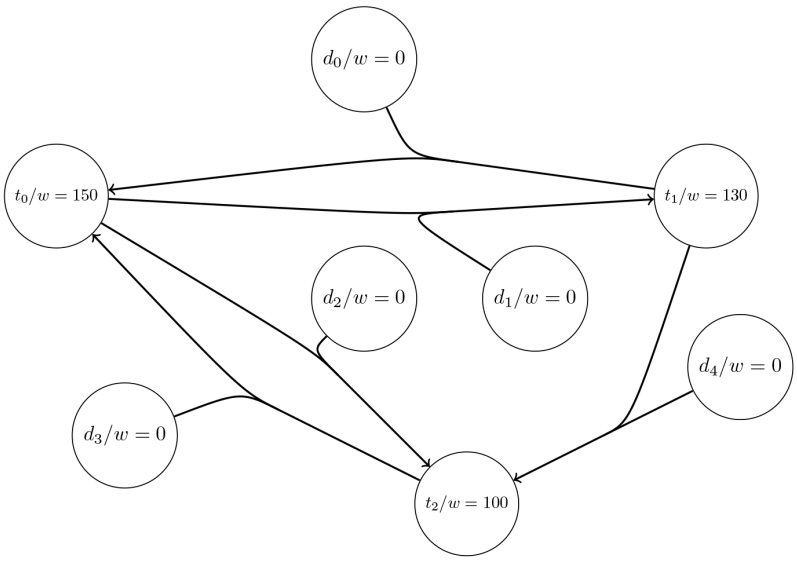

Figure 8. UMPa graph for sample application

COMPARATIVE STUDY

The modelling approaches discussed in the previous section established a set of numerous features to cover different characteristics and principles in modelling parallel applications. Due to this fact, all their possible characteristics cannot be simply listed in this study, but a group of basic characteristics and principles can be provided to distinguish between the modelling approaches. This group is divided into four dimensions: 1- temporal behaviour; 2- mathematical structure; 3- heterogeneity, and; 4- metrics. In this section these dimensions are discussed.

Temporal Behavioural

Temporal behaviour such as, "the output X within the next T time must be produced by a given system" or "the produced output must be sent in no bounded delivery time" allow for the solving of the problem of mapping and scheduling, making it more sufficient. The main objective in modelling an application is the minimisation of resource usage such as execution time (Roig et al., 2000). Inattention to temporal behaviour or task precedence relation in modelling causes tasks to be trapped in wait conditions. Waiting for a task for a message from another task to be received imposes an overhead and increases the total execution time of the application, and this reduces the performance of the mapping (Liu et al., 2007). This characteristic is not supported by all modelling approaches. Table 1 indicates the approaches that do support this feature.

Table 1
Temporal behaviour

Model	Temporal Behaviour
TIG	
TPG	✓
TTIG	✓
TTIGHa	✓
MPAHA	✓
UMPa	

Mathematical Structure

Each model employs a mathematical structure that helps to extract information from the subject application. Each structure has specific features and can capture certain types of information. In case of mapping, the common structures are graphs and hypergraphs. Moreover, each graph or hypergraph can be directed or undirected. Each modelling approach tries to capture essential details of the target application as much as possible in an unambiguous arrangement. However,

the structure being used should be able to handle the details captured. Overall, it has been shown that hypergraphs are more suitable structures in parallel application modelling and are able to capture more details than graph structures (Deveci, 2015; Deveci et al., 2015). In Table 2, the mathematical structures of the models are summed up.

Table 2
Models' mathematical structures

Models	Graph		Hyper-graph
	Undirected	Directed	
TIG	✓		
TPG		✓	
TTIG		✓	
TTIGHa		✓	
MPAHA		✓	
UMPa			✓

Heterogeneity

According to the structure being used and the weights and costs assigned to it, models can embrace some levels of heterogeneity. In modern parallel systems, two types of heterogeneity are common: processor heterogeneity and network heterogeneity. Processor heterogeneity means that a task can have a different execution time on different processors in the machine. Moreover, network heterogeneity indicates that a data block can have different transmission times on varied media that are connected to the processors. Failure to capture the heterogeneity of the target structure leads to inefficient scheduling and mapping, which reduces the overall execution performance (Asaadi, Khaldi, & Chapman, 2016; Xie, Zeng, Xiao, Li, & Li, 2017). Table 3 presents the models' ability to apprehend the heterogeneity of the machine.

Table 3
Models' ability to capture heterogeneity

Models	Heterogeneity	
	Processor	Network
TIG		
TPG		
TTIG		
TTIGHa	✓	✓
MPAHA	✓	✓
UMPa	✓	

Metrics. The final goal of modelling approaches is to map the subject application to the machine. Mapping has better quality if the produced scheduling decreases the resource usage on the target machine in comparison with a different scheduling. By defining a relevant cost function, the mapping problem transforms into an optimisation problem. The cost function for an optimisation problem specifies some constraints in the optimiser being optimised. The modelling approach should provide for the evaluation of these constraints. Metrics provided by the modelling approach correspond to the constraints of the optimisation cost function. The incompetence of the modelling approach in supporting metrics limits the optimiser's ability to produce efficient scheduling plans (Deveci, 2015). To measure the quality of the mapping, there are various metrics that have to be examined, such as total communication volume, total number of messages being sent, maximum number of messages being sent and total volume of messages being sent. The modelling scheme being used determines which metrics can be used. The modelling structure and dependent information prepare the mapper for calculating some of these metrics.

Table 4 summarises the ability of the models in calculating different metrics. In this table, the metrics shown are total communication volume (TCV), total number of messages being sent (or received) (TSM), maximum number of messages being sent (or received) (MMS) and total volume of messages being sent (or received) (TSV). The definitions and characteristics of modelling approaches are collected in the previous sections. According to this, there is no general modelling approach that can capture all required principles and characteristics in modelling parallel applications. The following is a brief discussion of our observation.

Table 4
Models' ability to provide metrics

Models	TCV	TSM	MMS	TSV
TIG	✓			
TPG	✓			
TTIG	✓			
TTIGHa	✓			
MPAHA	✓			
UMPa	✓	✓	✓	

The first and simplest modelling approach discussed is the Task Interaction Graph (TIG). Besides the broad usage of this model, the most significant flaw of this model is the lack of attention to the temporal behaviour of the applications. In this model, there is no tool or parameter to capture the temporal behaviour of the parallel programmes. Therefore, for example, if this model runs tasks t_0 and t_1 concurrently on different processors, and task t_2 needs some data produced by task t_1 at the end of its execution, then task t_2 has to wait; this waiting time is an overhead in computation resources. Furthermore, this model does not reflect the heterogeneity of the processors or networks. All these debilities make this model an inappropriate choice for big, modern distributed machines.

Nevertheless, different experiments show that this model could yield good performance in course-grain applications where a lot of computation is performed between small communication events (Ahmad, He, & Liou, 2002; Bishop, Kelliher, & Irwin, 1999; Censor, Gordon, & Gordon, 2001; Lam & Suen, 1995; Roig et al., 2007). For SPMD applications, where a single application is applied to different data sets concurrently, this model yields good performance (Karypis, Schloegel, & Kumar, 2003; Roig et al., 2002).

The other model discussed is the TPG model. This model attempts to refine the TIG model to capture the temporal behaviour of the parallel applications. It was used by many researchers such as Ali and El-Rewini (1993), Barbosa, Morais, Nobrega and Monteiro (2005), Bouvry, de Kergommeaux and Trystram (1995), Gil, Hernández, Rodriguez, Mauri and Radeva (2006), Hwang, Chow, Anger and Lee (1989), Kitajima, Tron and Plateau (1993), Kwok and Ahmad (1999, 1996), Ohtaki, Takahashi, Boku and Sato (2004), Topcuoglu, Hariri and Wu (2002), Xie and Qin (2005) and Yang and Gerasoulis (1994). These researchers showed that this model performed well for tightly coupled applications, where most of the communication events were at the beginning or at the end of the tasks. Although to model other applications it is possible to break tasks into smaller tasks where communication events take place at the beginning or end, this action causes TPG to build a very big graph of the application, which in turn, increases the complexity of the model and reduces its performance in a large scale (Roig et al., 2002). Moreover, like the TIG, this model does not seize any parameters related to heterogeneity of architecture, making it incompetent for use in model applications for modern clusters.

The TTIG model is the combination of two previously discussed models and it captures the temporal behaviour of parallel applications using a newly defined metric called degree of parallelism. This model is used in many practical algorithms as shown by Guirado, Roig and Ripoll (2013), Kang, He and Wei (2013), Upadhyaya and Rajan (2015), and Yang, Guang, Säntti and Plosila (2013). The results obtained by these authors confirmed that this model was suitable for applications for which the degree of parallelism was more than 0.5 (Roig et al., 2007, 2000, 2002). Accordingly, it is better to use previous models instead of TTIG in other applications that have a degree of parallelism more than 0.5. Additionally, this model does not weigh any parameter regarding heterogeneity of the distributed architectures; this makes it unsuitable for use in new distributed clusters.

None of the models discussed reflects the heterogeneity of the distributed architecture. The next model to be discussed, TTIGHa, is a revision of the TTIG model, which tries to capture heterogeneity within the architecture. Therefore, this model needs to be updated with the exact details of the architecture in which the parallel application is going to be executed (De Giusti et al., 2007). Some studies employ this model in their modelling phase and utilise the TTIGHa to model parallel applications (De Giusti, Chichizola, Naiouf, & De Giusti, 2008; De Giusti, Chichizola, Naiouf, & De Giusti, 2008). However, since this model is closely dependent on the machine's architecture, its structure is complex and the implementations if this model cannot be portable. In this model, the user needs to know the exact details of the architecture; however, obtaining this information is difficult or impossible in some contexts.

The MPAHA is a complete model that attempts to capture the benefits of all the previous models and propose a model that can be applied in different situations. This model captures the temporal behaviour of the applications and heterogeneity of the architecture without the

need to know the exact details of the hardware. Successful implementations of this model such as shown by De Giusti et al. (2009), and De Giusti, Chichizola, Naiouf, De Giusti and Luque (2010) reflect its good performance. However, the weakness of this model is that the application that is modelled has a lot of small volume communication in addition to some large volume communication. Since this method admits the volume of communication, in these circumstances, it induces overheads that reduce performance. In mapping that uses this model, the scheduler is able to optimise large-size communication when several such events are events are entered into the application.

The last model discussed uses the UMPa method and is called UMPa as well. Unlike the other models, this method uses the hypergraph instead of the graphs to model parallel applications. It has been shown that hypergraphs are more suitable than graphs for capturing the parallel application's characteristics (Karypis & Kumar, 2000). Although this is a new proposed method, different studies have been done on it and various implementations are done using this model (Balci & Akgüller, 2014; Schlag et al., 2015; Shahid, Raza, & Sajid, 2015). This model tries to cover many features of parallel applications such as temporal behaviour and heterogeneous architecture. The implementation of this method attests to its good performance and scalability (Balci & Akgüller, 2014; Deveci, 2015; Deveci et al., 2015; Schlag et al., 2015; Shahid et al., 2015). Nonetheless, the model has a weakness; tasks need to collect data, which are produced with more than one task. In parallel applications where the data element has more than one producer, this model cannot be used (Deveci, 2015; Deveci et al., 2015).

FUTURE WORK, A NEW MODELLING APPROACH

The main result of our observations is that the existing modelling approaches do not include all principles and characteristics needed in modelling parallel applications. In modern parallel and distributed computing, there are numerous situations, and improvements are to these models to abstract the applications.

In this section, a new modelling schema will be introduced that attempts to overcome the drawbacks of the modelling approaches. This model is an improved version of the UMPa, which was introduced in the previous section. It has been shown that hypergraphs are better structures for capturing the structure of parallel applications (Trifunovic & Knottenbelt, 2004). This model benefits from the hypergraphs along with the other measures. The detailed definition of this model is as follows. However, this is only an introduction of the model; other researchers and authors may use this as a basis for future work. In the future, the complete version of this modelling along with experiments will be provided.

Definition

In this model, for each parallel application, there are two sets:

1. Set of the tasks $\mathcal{T}$
2. Set of the volume of data elements being exchanged $\mathcal{D}$

Distributed machines on which the application is going to be executed have two sets:

1. Set of processors $\mathcal{P} = \{p_1, p_2, \dots, p_i\}$ representing the type of each processor
2. Set of communication links $\mathcal{M} = \{m_1, m_2, \dots, m_i\}$ representing the communication delay of each link in transferring 1KB of data.

Using sets $\mathcal{P}$ and $\mathcal{M}$, a symmetric matrix $\mathcal{S}$ with size $|\mathcal{P} \times \mathcal{P}|$ is defined for the distributed machine as shown below:

$$\mathcal{S} = \begin{bmatrix} ct_{11} & ct_{12} & ct_{13} & \cdots & ct_{1|\mathcal{P}|} \\ ct_{21} & ct_{22} & ct_{23} & \cdots & ct_{2|\mathcal{P}|} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ ct_{|\mathcal{P}|1} & ct_{|\mathcal{P}|2} & ct_{|\mathcal{P}|3} & \cdots & ct_{|\mathcal{P}||\mathcal{P}|} \end{bmatrix}$$

where, ct_{ij} represents the communication time of 1KB data between processor p_i and processor p_j. Clearly, $ct_{ij} = 0$ if and only if $i = j$ as there is no communication delay within any processor and for every two processors, p_i and p_j, $ct_{ij} = ct_{ji}$.

Using the sets and matrix defined above, a directed hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{A})$ is formed, where:

- $\mathcal{V}$ is the set of nodes. Each node v_i in this set represents a task $t_i \in \mathcal{T}$.
- $\mathcal{A}$ is the set of hyperarcs. Each hyperarc a_i in this set represents a data element $d_i \in \mathcal{D}$. The pins inside each hyperarc a_i ($pins[a_i]$) consist of tasks that produce the data d_i and the tasks that consume this data. Hyperarc $a_i \in \mathcal{A}$ is an ordered pair (Pr_i, Cn_i), where Pr_i and Cn_i are disjointed non-empty subsets of $\mathcal{V}$. Pr_i, which is the origin of a_i ($Pr_i = org(a_i)$), is a set of the producers of d_i. Moreover, Cn_i, which is the destination of a_i ($Cn_i = dest(a_i)$), is a set of the consumers of d_i. The hyperarc's flow is from $org(a_i)$ to $dest(a_i) = pins[a_i] \setminus org(a_i)$. (For more information on the directed hypergraph and its notations, refer to Gallo, Longo, Pallottino and Nguyen (1993))

Each node has a weight and each hyperarc has a cost. Each node v_i has a row vector $w_{V_i} = [et_{i_{p_1}} \quad et_{i_{p_2}} \quad \dots \quad st_{i_{p_{|\mathcal{P}|}}}]$ assigned to it as its weight. Each element of this vector $et_{i_{p_i}}$ represents the execution time of the task $t_i \in \mathcal{T}$ on processor $p_j \in \mathcal{P}$. Each hyperarc a_i has a cost $c[a_i]$ assigned to it that shows the volume of the data element $d_i \in \mathcal{D}$ ($c[a_i] = vol(d_i)$).

Modelling

In this part, a sample application presented in Figure 1 is modelled using the proposed modelling approach. Since this model needs information about the distributed architecture that the parallel application is going to be executed on, a four-processor machine consisting of two types of processor connecting by two types of communication medium is considered. Moreover, it is considered that the execution speed of the type two processor is half that of the type one and the communication delay of the type one medium is half that of the type two medium. The sample structure of the considered architecture is shown in Figure 9. Clearly, when there are different media connecting two processors, the maximum communication delay is assigned to the link.

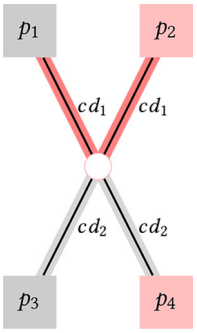

Figure 9. A schematic representation of a simple parallel machine

The preliminary sets are as below:

$$\mathcal{T} = \{t_1, t_2, t_3\}$$
$$\mathcal{D} = \{d_1, d_2, d_3, d_4, d_5\}$$
$$\mathcal{P} = \{p_{type_1}, p_{type_2}, p_{type_1}, p_{type_2}\}$$
$$\mathcal{M} = \{cd_1, cd_2\}$$

where, cd_i represents the communication delay of the type i medium for 1KB of data and $cd_2 > cd_1$.

$$S = \begin{bmatrix} 0 & cd_1 & cd_2 & cd_2 \\ cd_1 & 0 & cd_2 & cd_2 \\ cd_2 & cd_2 & 0 & cd_2 \\ cd_2 & cd_2 & cd_2 & 0 \end{bmatrix}$$

The model for this sample application is a directed hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{A})$, where:

- $\mathcal{V} = \{v_1, v_2, v_3\}$
- $\mathcal{A} = \{a_0, a_1, a_2, a_3, a_4\}$

The presentation of this model for this application is shown in Figure 10.

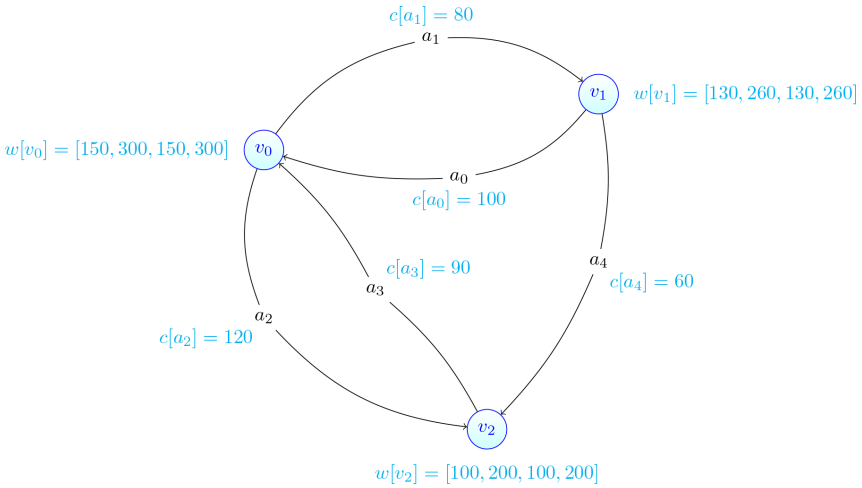

Figure 10. A graphical representation of the sample parallel application modelled using the proposed new modelling approach

CONCLUSION

This review afforded a general study of the modelling schemes proposed in the literature to enhance the understanding of unclear concepts by indicating the common principles and characteristics, similarities and differences and limitation and gap analysis of the modelling approaches. In order to help readers to make a wise selection between the models according to what they need, they are classified and compared from different aspects. The comparison framework is applied from theoretical and practical viewpoints that would recommend the most suitable model(s) according to the provisions of the machine and application. To support the conclusion drawn, four tables have been added as a summary of the evaluation. The main result of this study is that there is no one general purpose modelling approach that can capture all the principles and characteristics in modelling parallel applications. The study also identified the limitations of each modelling approach.

Finally, a new modelling approach that will solve the previous drawbacks by tolerating more types of application, allowing more metrics to measure and improving the accuracy of the mapping is briefly introduced. Extensive investigating, studying and implementing of the proposed model will be our future work.

ACKNOWLEDGEMENT

This research is fully funded by the Universiti Putra Malaysia under the Fundamental Research Grant Scheme (FRGS), FRGS No: 08-02-14-1580FR.

REFERENCES

Ahmad, I., He, Y., & Liou, M. L. (2002). Video compression with parallel processing. *Parallel Computing*, *28*(7), 1039–1078.

Ali, H., & El-Rewini, H. (1993). Task allocation in distributed systems: A split graph model. *Journal of Combinatorial Mathematics and Combinatorial Computing*, *14*(1993), 15–32.

Asaadi, H. R., Khaldi, D., & Chapman, B. (2016). A comparative survey of the HPC and big data paradigms: Analysis and experiments. In *Proceedings – IEEE International Conference on Cluster Computing, ICCC* (pp. 423–432). IEEE. https://doi.org/10.1109/CLUSTER.2016.21

Balci, M. A., & Akgüller, Ö. (2014a). Average weakly hyperedge domination number for a hypergraph and actor-network application. *International Journal of Modeling and Optimization, 4*(5), 346.

Barbosa, J., Morais, C., Nobrega, R., & Monteiro, A. P. (2005). Static scheduling of dependent parallel tasks on heterogeneous clusters. In *Cluster Computing, 2005. IEEE International* (pp. 1–8). IEEE.

Bishop, B., Kelliher, T. P., & Irwin, M. J. (1999). A detailed analysis of MediaBench. In *Signal Processing Systems, 1999. SiPS 99.* 1999 IEEE Workshop (pp. 448–455). IEEE.

Bouvry, P., de Kergommeaux, J. C., & Trystram, D. (1995). Efficient solutions for mapping parallel programs. *EURO-PAR'95 Parallel Processing* (pp. 379–390). Springer.

Censor, Y., Gordon, D., & Gordon, R. (2001). Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems. *Parallel Computing*, *27*(6), 777–808.

Culler, D. E., Karp, R. M., Patterson, D., Sahay, A., Santos, E. E., Schauser, K. E., … & von Eicken, T. (1996). LogP: A practical model of parallel computation. *Commun. ACM*, *39*(11), 78–85. https://doi.org/10.1145/240455.240477

De Giusti, L. C., Chichizola, F., Naiouf, M., & De Giusti, A. E. (2008). Robustness analysis for the method of assignment MATEHa. *Journal of Computer Science and Technology*, *8*(1), 1-7.

De Giusti, L. C., Chichizola, F., Naiouf, M., Ripoll, A., & De Giusti, A. E. (2007). A model for the automatic mapping of tasks to processors in heterogeneous multi-cluster architectures. *Journal of Computer Science and Technology*, *7*(1), 39-44.

De Giusti, L. C., Naiouf, M., Chichizola, F., Luque Fadón, E., & De Giusti, A. E. (2009). Dynamic scheduling in heterogeneous multiprocessor architectures. In *XV Congreso Argentino de Ciencias de la Computación* (pp. 221-230).

De Giusti, L., Chichizola, F., Naiouf, M., & De Giusti, A. (2008). Mapping tasks to processors in heterogeneous multiprocessor architectures: The MATEHa Algorithm. In *Chilean Computer Science Society, 2008. SCCC '08.* International Conference (pp. 85–91). https://doi.org/10.1109/SCCC.2008.11

De Giusti, L., Chichizola, F., Naiouf, M., De Giusti, A., & Luque, E. (2010). Automatic mapping tasks to cores-evaluating amtha algorithm in multicore architectures. *International Journal of Computer Science Issues, 7*(2), 1-6.

Deveci, M. (2015). *Load-Balancing and task mapping for exascale systems*. The Ohio State University.

Deveci, M., Kaya, K., Ucar, B., & Catalyurek, mit V. (2015). Hypergraph partitioning for multiple communication cost metrics: Model and methods. *Journal of Parallel and Distributed Computing*, *77*, 69–83. https://doi.org/10.1016/j.jpdc.2014.12.002

El-Rewini, H., Lewis, T. G., & Ali, H. H. (1994). *Task scheduling in parallel and distributed systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Gallo, G., Longo, G., Pallottino, S., & Nguyen, S. (1993). Directed hypergraphs and applications. *Discrete Applied Mathematics*, *42*(2–3), 177–201.

Gil, D., Hernández, A., Rodriguez, O., Mauri, J., & Radeva, P. (2006). Statistical strategy for anisotropic adventitia modelling in IVUS. *Transactions on Medical Imaging, IEEE 25*(6), 768–778.

Guirado, F., Roig, C., & Ripoll, A. (2013). Enhancing throughput for streaming applications running on cluster systems. *Journal of Parallel and Distributed Computing*, *73*(8), 1092–1105.

Hui, C. C., & Chanson, S. T. (1997). Allocating task interaction graphs to processors in heterogeneous networks. *IEEE Transactions on Parallel and Distributed Systems, 8*(9), 908–925. https://doi.org/10.1109/71.615437

Hwang, J. J., Chow, Y. C., Anger, F. D., & Lee, C. Y. (1989). Scheduling precedence graphs in systems with interprocessor communication times. *SIAM Journal on Computing*, *18*(2), 244–257.

Kalinov, A., & Klimov, S. (2005). Optimal mapping of a parallel application processes onto heterogeneous platform. In *19th IEEE International Proceedings on Parallel and Distributed Processing Symposium, 2005* (pp. 123b–123b). IEEE. https://doi.org/10.1109/IPDPS.2005.310

Kang, Q., He, H., & Wei, J. (2013). An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems. *Journal of Parallel and Distributed Computing*, *73*(8), 1106–1115.

Karypis, G., & Kumar, V. (2000). Multilevel k-way hypergraph partitioning. *VLSI Design*, *11*(3), 285–300.

Karypis, G., Schloegel, K., & Kumar, V. (2003). *ParMeTiS: Parallel graph partitioning and sparse matrix ordering library–Version 3.1*. University of Minnesota.

Kasahara, H., & Narita, S. (1985). Practical multiprocessor scheduling algorithms for efficient parallel processing. *Systems and Computers in Japan*, *16*(2), 11–19. https://doi.org/10.1002/scj.4690160202

Kitajima, J. P., Tron, C., & Plateau, B. (1993). Environments and tools for parallel scientific computing. In J. J. Dongarra & B. Tourancheau (Eds.), *Environments and tools for parallel Scientific computing* (pp. 213–228). Amsterdam, The Netherlands: Elsevier Science Publishers BV. Retrieved from http://dl.acm.org/citation.cfm?id=165125.165280

Kumar, V., Grama, A., Gupta, A., & Karypis, G. (1994). *Introduction to parallel computing: Design and analysis of algorithms*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.

Kwok, Y. K., & Ahmad, I. (1996). Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *Transactions on Parallel and Distributed Systems, IEEE7*(5), 506–521.

Kwok, Y. K., & Ahmad, I. (1999). Benchmarking and comparison of the task graph scheduling algorithms. *Journal of Parallel and Distributed Computing*, *59*(3), 381–422. https://doi.org/http://dx.doi.org/10.1006/jpdc.1999.1578

Lam, L., & Suen, C. Y. (1995). An evaluation of parallel thinning algorithms for character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (9), 914–919.

Lastovetsky, A., & Manumachu, R. R. (2017). New Model-Based Methods and Algorithms for Performance and Energy Optimization of Data Parallel Applications on Homogeneous Multicore Clusters. *IEEE Transactions on Parallel and Distributed Systems, 28*(4), 1119–1133.

Leopold, C. (2001). *Parallel and distributed computing: A survey of models, paradigms and approaches*. New York, NY, USA: John Wiley & Sons, Inc.

Lewis, T. G., & El-Rewini, H. (1992). *Introduction to parallel computing.* New York: Prentice-Hall.

Liu, X., Shi, H., Lu, Q., & Mao, Z. (2007). Visual task-driven based on task precedence graph for collaborative design. In *11th International Conference on Computer Supported Cooperative Work in Design, 2007. CSCWD 2007.* (pp. 246–251). IEEE. https://doi.org/10.1109/CSCWD.2007.4281442

Long, D. L., & Clarke, L. A. (1989). Task interaction graphs for concurrency analysis. *Proceedings of the 11th International Conference on Software Engineering* (pp. 44–52). New York, NY, USA: ACM. https://doi.org/10.1145/74587.74592

Ohtaki, Y., Takahashi, D., Boku, T., & Sato, M. (2004). Parallel implementation of Strassen's matrix multiplication algorithm for heterogeneous clusters. In *18th International Proceedings on Parallel and Distributed Processing Symposium, 2004.* (p. 112). IEEE.

Phinjaroenphan, P., & Bevinakoppa, S. (2004). A novel algorithm for mapping parallel applications in computational grid environments. In *Proceedings of Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region, 2004.* (pp. 347–350). IEEE. https://doi.org/10.1109/HPCASIA.2004.1324056

Rico-Gallego, A. J., Lastovetsky, A., & Diaz-Martin, J. C. (2017). Model-Based estimation of the communication cost of hybrid data-parallel applications on heterogeneous clusters. *IEEE Transactions on Parallel and Distributed Systems*, *9219*(99), 1–1. https://doi.org/10.1109/TPDS.2017.2715809

Roig, C., Ripoll, A., & Guirado, F. (2007). A new task graph model for mapping message passing applications. *Transactions on Parallel and Distributed Systems, IEEE 18*(12), 1740–1753. https://doi.org/10.1109/TPDS.2007.1117

Roig, C., Ripoll, A., Senar, M. A., Guirado, F., & Luque, E. (2000). Modelling message-passing programs for static mapping. In *Proceedings of 8th Euromicro Workshop Parallel and Distributed Processing, 2000.* (pp. 229–236). IEEE. https://doi.org/10.1109/EMPDP.2000.823416

Roig, C., Ripoll, A., Senar, M. A., Guirado, F., & Luque, E. (2002). A new model for static mapping of parallel applications with task and data parallelism. In *Proceedings of International Parallel and Distributed Processing Symposium., IPDPS 2002, Abstracts and CD-ROM* (p. 8). IEEE. https://doi.org/10.1109/IPDPS.2002.1015586

Sadayappan, P., Ercal, F., & Ramanujam, J. (1990). Cluster partitioning approaches to mapping parallel programs onto a hypercube. *Parallel Computing*, *13*(1), 1–16. https://doi.org/http://dx.doi.org/10.1016/0167-8191(90)90115-P

Sanyal, S., & Das, S. K. (2005). MaTCH : Mapping data-parallel tasks on a heterogeneous computing platform using the cross-entropy heuristic. In *19th IEEE International Proceedings on Parallel and Distributed Processing Symposium, 2005* (p. 64b–64b). IEEE. https://doi.org/10.1109/IPDPS.2005.274

Sarkar, V. (1989). *Partitioning and Scheduling parallel programs for multiprocessors*. Cambridge, MA, USA: MIT Press.

Schlag, S., Henne, V., Heuer, T., Meyerhenke, H., Sanders, P., & Schulz, C. (2015). k-way hypergraph partitioning via n-level recursive bisection. In *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)* (pp. 53-67). Society for Industrial and Applied Mathematics.

Shahid, M., Raza, Z., & Sajid, M. (2015). Level based batch scheduling strategy with idle slot reduction under DAG constraints for computational grid. *Journal of Systems and Software*, *108*, 110–133.

Tekinerdogan, B., & Arkin, E. (2012). *Architecture framework for modeling the deployment of parallel applications on parallel computing platforms*. In *3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD), 2015* (pp. 185-192). IEEE.

Topcuoglu, H., Hariri, S., & Wu, M. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems, 13*(3), 260–274.

Trifunovic, A., & Knottenbelt, W. J. (2004). A parallel algorithm for multilevel k-way hypergraph partitioning. In *Third International Workshop on Parallel and Distributed Computing, 2004. Third International Symposium on/Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, 2004.* (pp. 114–121). IEEE.

Ullman, J. D. (1975). NP-complete scheduling problems. *Journal of Computer and System sciences*, *10*(3), 384–393. https://doi.org/10.1016/S0022-0000(75)80008-0

Upadhyaya, G., & Rajan, H. (2015). Effectively mapping linguistic abstractions for message-passing concurrency to threads on the java virtual machine. *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications* (pp. 840–859). New York, NY, USA: ACM. https://doi.org/10.1145/2814270.2814289

Valiant, L. G. (1990). A bridging model for parallel computation. *Communications of the ACM*, *33*(8), 103–111. https://doi.org/10.1145/79173.79181

Xie, G., Zeng, G., Xiao, X., Li, R., & Li, K. (2017). Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems. *IEEE Transactions on Parallel and Distributed Systems, 28*(12), 3426-3442.

Xie, T., & Qin, X. (2005). A new allocation scheme for parallel applications with deadline and security constraints on clusters. In *IEEE International on Cluster Computing, 2005* (pp. 1-10). IEEE.

Yang, B., Guang, L., Säntti, T., & Plosila, J. (2013). Mapping multiple applications with unbounded and bounded number of cores on many-core networks-on-chip. *Microprocessors and Microsystems*, *37*(4), 460–471.

Yang, T., & Gerasoulis, A. (1994). DSC: Scheduling parallel tasks on an unbounded number of processors. *IEEE Transactions on Parallel and Distributed Systems, 5*(9), 951–967. https://doi.org/10.1109/71.308533