Challenges of Run-Time Load Distribution in Heterogeneous Shared Clusters

Alfrânio Tavares Correia Júnior

Departamento de Informática, Universidade do Minho
4710-057 Braga, Portugal
alfranio@lsd.di.uminho.pt

Abstract. The aim of this communication is to show the challenges of run-time load distribution in heterogeneous shared clusters. The main feature to be analysed is the application level scheduler and its performance. The goals and constraints of a load management policy are presented, together with a proposed taxonomy to classify the different types of load distribution, including static ones. The communication closes with an introduction to the class of adaptive schedulers addressing the problems of uncertainty in heterogeneous shared clusters.

1. Introduction

The number of applications that requires more computing power is increasing every day. However, a sequential computer cannot provide enough computing power to accomplish the desired results in a satisfactory manner: the time spent to achieve a result is an important constraint to decide about the feasibility of an application. One way of overcoming this problem is to improve the speed of processors and other components so that they can offer the power required by computationally intensive applications. Even though, this solution is limited by the speed of light, thermodynamic laws, and the high financial costs for processor fabrication. A cost-effective alternative solution is to combine the power of multiple processors. These processors would be connected and their efforts would be coordinate to achieve the desired results in a satisfactory manner.

The different approaches to processor interconnection, their strength and weakness, are not subject of this communication (more information in [1]). The interesting of this communication is on scheduling of workloads in stand-alone or autonomous machines working together as a single integrated computing resource, and interconnected by a high-speed network. Each machine may have different types and numbers of processors. There is not a constraint about the power and the type of each machine, even in what concerns the operating system.

To achieve the best result, in this case the fastest execution time: it is important to organize the workload in the best possible way, applying load distribution approaches. This communication addresses load distribution of divisible works, stressing the application level scheduling rather than the operating system approach. In the former, the load distribution algorithm is on top of the clustered environment and in a sense embedded in the application. This brings more flexibility to the application allowing its development in the best possible way according to its characteristics.

Such systems may or may not be dedicated to a specific application. In a shared system, the total number of applications running in a cluster can be unpredictable, increasing the complexity of the load distribution. Therefore, this mechanism will need to deal with the challenge of organizing the workload in the best possible way, deciding in run-time which workload to process in each computer node. The run-time mechanism is extremely necessary, particularly in a heterogeneous shared cluster. In contrast, the use of a static load distribution may impair the application's performance.

This communication is structured as follows: section 2 characterizes scheduling concepts, presents a current taxonomy for load distribution and briefly describes the strength and weakness of competitive approaches. Section 3 describes in detail the load distribution problem in heterogeneous shared clusters. Section 4 presents an adaptive approach, a way of overcoming some limitations of deterministic or dynamic blind load distributions[1]. Section 5 concludes the communication.

2. Background

This section presents a taxonomy to the scheduling problem and relevant concepts. First, some different types of scheduler's names are analysed followed by a workload classification.[2] After that, decisions about its interaction with the environment and its functioning are considered, namely: centralization versus distribution, static versus dynamic and deterministic versus stochastic.

2.1. Scheduling

Nowadays, the term distribution is more common then scheduling, balancing or sharing. However, it is important to notice the difference between each one and to know when to use them in an interchangeable manner.

The term scheduling is used when the operating system is responsible to the workload policies and mechanisms. Nevertheless, only a distributed operating system can provide an integrated scheduling since it is necessary to transfer works to stand-alone machines in a transparent manner to the developer and user [2]. If there is not a distributed operating system, the cluster environment and the application need to play this role. However, it is common to use *"scheduling"* to represent the generic meaning of activity allocation.

The main concern of a load balancing is to equalize the average load of each node in proportion to its performance. The goal of the load sharing is not as strong as the load balancing but its responsibility is to share the units of work without incurring in overloaded nodes. The term load distribution is more generic than the others are and its goal can be declared as either an objective or a mathematical function according to the application's characteristics. Nevertheless, it usually means minimization of application's completion time. This definition is considered in the remaining text unless otherwise declared, since it is more suitable to the problem analysed in this communication [3] [4].

2.2. Workload

Another important concept is the unit of work: jobs, tasks or applications. These terms are related to the grain size of processing loads and to the independency of each one.

The term job in [1] is used to identify an agglomeration of computational tasks usually solving a complex problem. In [3][5] job is a collection of independent tasks whose loads are indivisible, and in general of different sizes. A job may be a single program or a collection of programs.

An application is a program that can be divided in small tasks from a data or functional perspective. The former is more common and simpler and can be classified as either modularly divisible or arbitrarily divisible [5].

[1] Basic knowledge in decision network (Bayesian network in [11]) is required.

[2] Scheduling and workload are concepts usually presented together in result to its correlation, but here they are split to be more understandable and clear.

Finally, the task is a generic term used to characterize a unit of work with different grain sizes and dependencies.

2.3.Centralized and Distributed Scheduling

Scheduling policies may be centralized or distributed. The choice of each one is a common issue of a distributed system [6]. A centralized approach has unique and coherent information about the environment. Therefore, there is no need to data replication and coherence mechanism incurring in additional communication's traffic. Suffering a consequence of the centralization, this approach does not scale well with the system's size. To overcoming this issue, a distributed solution is desirable. The distributed approach has information's partial view, additional communication's overhead, although, it scales with the system's size ([6] for a detailed discussion about the advantages and disadvantages of each one).

2.4.Static and Dynamic Scheduling

Paraphrasing [5]: in a broad sense, load scheduling/sharing problems can be classified in many ways. One of the possible classifications is: static (further detail in [3][7]) or dynamic (further detail in [8]).

The static solution, which is usually done at compile time, is applied when the characteristics of the parallel application are known a priori and there is a controlled and deterministic environment. Task processing times, dependencies, communication, data dependencies, and synchronization requirements must be known in advanced to take advantage of a static scheduling. Otherwise, in the absence of these informations, scheduling is done on the fly according to the state of the system. It is also important to notice that different parts of the scheduling problem can be done on-line (on-line is used as synonymous of dynamic) [9], e.g., a well known parallel application can be scheduled off-line (off-line is used as synonymous of static) but the mapping phase can be done on-line (further detail about scheduling and mapping in [3]).

2.5. Deterministic and Stochastic Scheduling

The deterministic solution is feasible when the variables that influence the scheduling are well known or can be predicted. In contrast, the stochastic solutions are based in uncertainty [9] [4] [10] and must be interpreted as a synonymous of probabilistic solutions.

According to [5], the static scheduling can be either deterministic or stochastic, e.g., the knowledge lack of the time of loads is an uncertainty to be explored by stochastic solutions in a static scheduling. However, most of the static problems are modelled in a deterministic perspective. The static scheduling problem has been proven to be NP-complete except for a few restricted cases [3]. This implies the use of some heuristics to produce the sub optimal results in a feasible time.

Similarly, dynamic scheduling can be either deterministic or stochastic, although, usually stochastic frameworks are applied.

3. The Scheduling Problem

This section presents the scheduling problem and policies that a scheduler can adopt to make decisions about the load distribution.

3.1. Generic Assumptions

The scheduling problem can be formulated as a four-step approach (a detailed and formal presentation in [5]): modelling of the system, definition of the type of processing load, formulation of an objective or cost function and specification of the constraints.

The Modelling step is responsible for system's description similar to network's topology and number of processors. As stated before, the system analysed in this communication is a heterogeneous shared cluster with a high-speed communication solution.

The load step is responsible to define the type of workload. In this communication, parallel application tasks with arbitrarily divisible data processing loads are stressed.

The objective step can be declared as either an objective or a mathematical function. It is expressed here, as an objective: reduction of the parallel application's completion time.

The constraints are related to the uncertainty of the environment. The number of available nodes, the number of running applications and the percentage of each communication path usage are examples that illustrate this step.

To accomplish its goal the scheduler needs to perceive or measure the system according to the specification of each constraint and its final objective [4]: its accuracy is responsible to its effectiveness (figure 1). However, the environment may be so complex that scheduler cannot evaluate all the relevant aspects. Some of them must be neglected or summarized, in order that the objective is accomplished in a reasonable time. Furthermore, the collected information may be either inaccurate because of the expensiveness of acquiring it with great precision or may be often out of date in consequence of system's continuous changes.

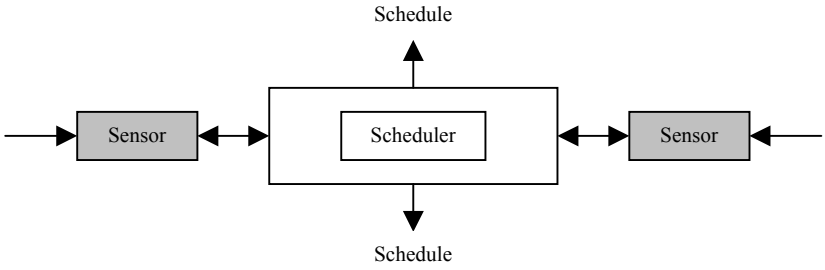

Fig. 1. An abstract representation of a scheduler with its sensors.

3.2. Policies

The interaction with the environment can be divided into a set of policies as follows [4] [2]: information policy, transfer policy, location policy and selection policy.

The information policy determines "*what"* information and *"when"* it will be collected.[3]

Observing the *"when"* perspective, it is possible to collect information either in a demand, periodic or state driven policy.

In a *"demand-driven"* policy, information is gathering when the scheduler needs it for decision-making. The scheduler probes for a suitable node to participate in the workload distribution. Each node is evaluated one by one until a suitable node is found or a probe limit is exceeded. Another solution is to send a broadcast to a group of nodes and wait for any node to manifest its possibility to join itself. If a node is not found another group is chosen.

[3] This classification is different from [4] because *"where"* perspective is placed here in the transfer policy.

In a *"periodic-driven"* a fixed overhead is imposed to the system. However, when the scheduler needs some information for a decision it is not necessary to wait for a gathering step. Since the system is always changing, this information is out-of-date. Increasing the number of measures or samples per unit time can reduce the information's age, but generates a huge overhead in the system.

In a *"state-driven"*, the information is collected whenever it changes bellow or above predefined thresholds.

Observing the "*what"* perspective, there are several load metrics defined in the literature [4] [2]: CPU queue length, CPU utilization, I/O queue length, I/O utilization and memory utilization for example. They can be observed one by one or combinations can be defined as metrics. However, it has been proved that the CPU queue length is one of the most adequate queue metrics and it is not required to be very accurate.

The transfer policy determines if a node is in a suitable state to accept a unit of work. Some thresholds must be defined for this proposal. A node is considered a *"sender"* if the measure values are above this threshold or a *"receiver"* if they are bellows it (more detailed explanation of the problems related to a sender and receiver perspective in [4] [2]).

The location policy determines the set of rules that will be used to decide which task will be transferred, and may be considered the scheduler's brain. The selection policy is consequence of the former because after node's selection, it will be necessary to define the task to be transferred. The number of location-dependent resources such as: specific data, I/O devices and communication ports must be considered. The observation of the process's life is desirable and often indispensable according to the overhead of the process's execution state.

4. Adaptive Load Distribution

As state before, the use of the term stochastic is related to probability. However, there are two schools of probability: objective and subjective. The first consider probability related to events. The former extends this and consider that there are uncertainties about the possible events dependencies and in this way, a probability is associated to each relation. Moreover, they also consider that these values can change with the knowledge about the world ([11] for more information).

In this section, the world stochastic is used to present a class of algorithms based in the second school, namely stochastic adaptive schedulers or just adaptive schedulers. Nevertheless, algorithms based in the traditional probability school are also possible, although they are not discussed.

To be able to achieve their requirements, the scheduler must have an internal execution model of the world. This execution model may be centred in deterministic or probabilistic algorithms.

The use of a deterministic dynamic scheduling may impair the results in terms of completion time as consequence of the uncertainty variables. The combination of dynamic scheduling with stochastic adaptable approaches can improve the solutions, resulting in better completion times [4].

In a stochastic scheduling, assigned to each uncertainty there is a variable with a probability of occurrence [4] [10] [11]. The scheduler's internal execution model builds probabilistic dependences between the variables. These dependences, statements about the variables, can be used to infer probabilities about any set of variables given any another one. Therefore, the stochastic approach tries to extract and encode the knowledge from data, being possible, for example, to predict the environment behaviour or the impact of an

action. For this proposal, there are many representations and techniques available for data analysis [11]. To the scheduling problem [4], a Bayesian approach is more suitable since it is possible to use incomplete data sets, to learn about causal relations, to represent the domain knowledge and data, and to avoid the over fitting of data.

The adaptive characteristic is related to the fact that this stochastic model can be trained or it can learn to reflect better actions according to the current environment's state and its knowledge about it.

The [4] identifies three places to apply the stochastic adaptive mechanism: correction of profile and load predictions, determination of diffuse factors in the execution model and control of the relationship between overhead and profit.

The first step to build such a model is the understanding of the problem. Specialists with knowledge about the subject and some background in decision theory must choose the variables and establish their dependences. Unidentified variables, usually called latent variables, with some influence in the probabilities of others (causality relation) can disrupt the model. The use of many variables can lead to more realistic models, although, it also increases the number of relations and possible models.

After that, the continuous variables (non-discrete) must be discretized. This process is responsible to transform this type of data in intervals, since with categorical data it is possible to build models that can capture the non-linear relationships between variables (more information in [11][9][4]). In other words, it is possible to gather more occurrences or information about the events since the range is smaller. The next step is to assign the probabilities of dependencies of the data.

It is important to notice that the construction of the model is highly dependent of the specialists. Their knowledge about the subject is directly related to the definition of the dependencies. Therefore, these steps must be made in a very careful manner.

To a better understanding of the main ideas of an adaptive scheduling mechanism, it is analysed a simplified example based in [4]. The scheduler's policies are as follows:

1. Centralized approach is the choice since a small cluster with a high-speed interconnection is used.
2. Every task application sends its processing rate to the scheduler after a maximum pre-defined value. If the task application never sent a value, then one is sent. If the value of the rate processing is *x percent* different from the previous then this value is sent.

To model environment's behaviour, the designer must represent its current state, the possible actions to modify it, the state transition, the next state and a gain function [4][11].

The environment's current state includes the task requirement and the resource's capacity. The task requirement is a measure that indicates the cost benefit of a migration, based in the time spent to transfer a task and its remaining processing. The resource's capacity is a measure that indicates if a node is overloaded or not, based in the processing rate of each application where a lower rate indicate an overloaded node. These tradeoffs are used to conclude if a machine is a transfer or a receptor, as a result.

The sensors (figure 1) used to measure or perceive these values must be extended since their operations can be prone to errors and imprecision.

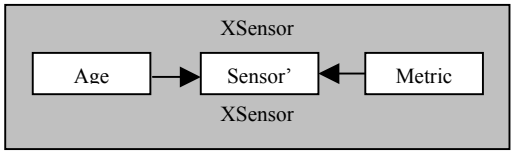

Fig. 2 An extension of the sensor presented in figure 1. XSensor stands for Extended Sensor.

Sensor' is an evidence perceived from the environment. There is no uncertainty on its values, although, they can be influenced by the information's age and the environment's current state. Therefore, a diagnostic inference is applied and after that, it is possible to known the potential current value (a probability vector represents this fact). The age probability can be discarded since it is possible to known exactly whenever a new value is read and the time elapsed since the last operation.

These sensors and their relative probabilities must be analysed to the pairs in view of the fact that the scheduler's final action is a task transfer/split from a node *a* to *b*.

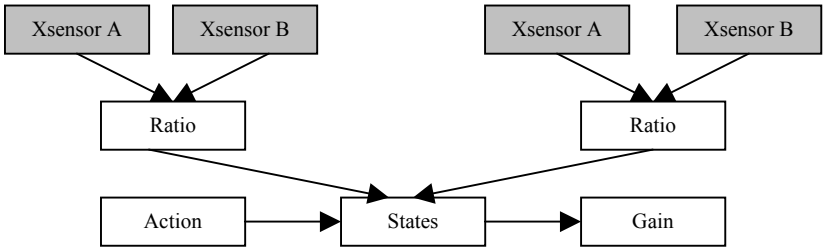

Fig. 3. The complete decision network.

Figure 3, represents the environment's model, using a decision network. With all the evidences, the scheduler obligation is to verify the gain involved in each possible action and after that decide for the best one. To accomplish this, it can follow this possible algorithm from [4]:

```
Set the evidences variables
For each possible setting of the decision node
   i)  Set the decision node to that value
   ii) Propagate the beliefs through the network to compute the new
   distribution over the relevant variables
   iii) Compute the expected gain for this action
Choose the action with the highest expected gain.
```

5. Conclusions

This communication presented the necessary background to conclude that the scheduling mechanism is highly dependent of the environment's characteristics. Therefore, in a heterogeneous shared cluster a dynamic adaptive scheduler is more suitable then others. Paraphrasing [4]: the effectiveness of the stochastic scheduler, particularly of the adaptive one, increases with the complexity of the environment being managed. For example, the newly emerging computational Grid seems to be an excellent environment to adaptive schedulers.

References

[01] Buyya, R.: High Performance Cluster Computing, Volume 1. Prentice Hall, (1999).

[02] Tanenbaum, A.: Distributed Operating Systems. Prentice Hall, (1995).

[03] Kwok, Y. and Ahmad I.: Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors. ACM Computing Surveys, 31(4), (1999), 406–471.

[04] Paulo L.: Application Level Run Time Load Management: a Bayesian Approach. PhD thesis, Departamento de Informática - Universidade do Minho, (2001).

[05] Venkataraman, M., Veeravalli, B., Debasish, G. and Robertazzi, T.: Scheduling Divisible Loads in Parallel and Distributed Systems. IEEE Computer Society Press, (1996).

[06] Dollimore J., Coulouris G. and Kindberg T.: Distributed Systems: Concepts and Design. Addison-Wesley, 3rd edition, 2001.

[07] Ahmad, I., Kwok, Y., Wu, M. and Shu, W.: Casch: A Software Tool for Automatic Parallelization and Scheduling of Programs on Multiprocessors. IEEE Concurrency, 8(4), (2000), 21–33.

[08] Sgall, J.: On-line scheduling - A Survey. In A. Fiat and G. Woeginger, editors, On-line Algorithms: The State of the Art. Lecture Notes in Computer Science, Springer Verlag, (1998), 196-231.

[09] Jain, R.: The Art of Computer Systems Performance Analysis. John Wiley, 1991.

[10] Raymond, H., Myers, R., Walpole, E. and Myers, S.: Probability and Statistics for Engineers and Scientists. Prentice Hall, 6th edition, (1998).

[11] Heckerman, D.: A Tutorial on Learning With Bayesian Networks. Microsoft Advanced Technology Division. (1995)