Scheduling Under Conditions of Uncertainty: a Bayesian Approach*

Luis Paulo Santos and Alberto Proenca

Departamento de Informatica
Universidade do Minho
Campus de Gualtar
4710-057 Braga
Portugal
[psantos,aproenca]@di.uminho.pt

Abstract. The efficient execution of irregular parallel applications on shared distributed systems requires novel approaches to scheduling, since both the application requirements and the system resources exhibit an unpredictable behavior. This paper proposes Bayesian decision networks as the paradigm to handle the uncertainty a scheduler has about the environment's current and future states. Experiments performed with a parallel ray tracer show promising performance improvements over a deterministic approach of identical complexity. These improvements grow as the level of system sharing and the application's workload irregularity increase, suggesting that the effectiveness of decision network based schedulers grows with the complexity of the environment being managed.

1 Introduction

Shared parallel and cluster computing, coupled to the exploitation of unused cycle times on interconnected workstations, is pushing the computing paradigms towards new levels of expectations. To assure that parallel applications get a performance close to the theoretical available processing power, the workload must be effectively distributed over the available resources. The role of the scheduler is to ensure this adequate correspondence between the workload structure - both code and data - and the distributed system resources.

The scheduling problem complexity increases when applications exhibit unpredictable computing and communication requirements, and the available set of computing resources is dynamically shared among several users and applications: the scheduler must still correctly distribute the workload on the shared set of resources, but exact prediction of the environment behavior is impossible due to its dynamic nature [1].

Dynamic scheduling strategies that regularly measure the environment state seem appropriate, since they enable the scheduler to react to fluctuations on the environment behavior. However, the data gathered through the scheduler's

* This work was partially supported by grant PRAXIS 2/2.1/TIT/1557/95

sensors, about the environment's current state, quantifies only those quantities that were included on the scheduler's simplified model of the world, may suffer from noise and will age with time. Furthermore, the future state of the environment and the outcome of the scheduler selected actions can not be accurately predicted, because other processes, independent of the scheduler, are acting on the shared computing resources. The scheduler is required to decide and act under conditions of uncertainty about past, present and future system states and workload profiles.

The problem of handling uncertainty on computational models of real world problems can be solved by using decision theory. Decision theory combines probabilities and utilities to evaluate alternative actions. One of the tools proposed by decision theory for rational decision making under conditions of uncertainty are decision networks [2–4]. Decision networks allow the inclusion, on the reasoning process, of the uncertainty about the environment current state and about the consequences of the scheduling agent's selected actions, i.e., the environment next state after each scheduling event. The hypothesis put forward by this paper is that a scheduler is more effective if it uses decision making mechanisms that deal explicitly with this uncertainty, and that Bayesian decision networks are such a mechanism.

The use of decision networks to dynamically schedule a parallel application among the nodes of a distributed shared system is the main contribution of this paper; the authors have no knowledge of any related work where decision networks have been applied to solve this problem. Previous works modelled stochastically either only the environment's current state (e.g., Bayesian probabilistic networks [5], stochastic structural execution models [6, 7], handling information aging [8, 9]) or only the suitability of each possible action, given the environment's current state, such as with stochastic learning automata [10, 11]. Bayesian decision networks allow the integration of the uncertainty on both these factors on a single paradigm, and provide an automated process for computing the expected utility of each action, enabling rational decision making by selecting the action which maximizes the expected utility.

2 Handling Uncertainty

A dynamic scheduler collects information about the system state and workload profile and creates an internal image of the environment current state. Using this information and its execution model of the world, it generates estimates about the environment next state for each possible action a_i. The scheduler's decision making mechanism must select the action that leads the environment to the most desirable next state [12]. Uncertainty, however, derives from four main sources and may hinder the scheduler from meeting its performance requirements:

1. it is too expensive, or even impossible, to get exact and accurate information about the current environment state, i.e., the environment is not totally measurable;

2. the image the scheduler has about the environment state gets obsolete with time – information aging;
3. the environment complexity requires that some simplifications be included in the execution model, either by neglecting or summarizing some of the environment characteristics; therefore, the model can not provide exact and accurate predictions of the environment near future behavior;
4. the workload profile and system behavior can be unpredictable, both due to the application characteristics and to the background workload imposed upon the distributed shared system by other users and/or applications.

2.1 Bayesian Decision Networks

Due to uncertainty the scheduler's knowledge is, at best, a degree of belief on the environment's most relevant aspects. Probability theory provides a tool to process and combine these beliefs [3, 13]. A probabilistic model consists of a set of stochastic variables that represent some important aspect of the world. The joint distribution assigns probabilities to all possible states of the world. Its size grows exponentially with the number of variables and these probabilities are seldom easy to assess.

High–dimensional probabilistic models can only be computationally tractable if modularity is introduced [3, 14]. This may be achieved by introducing the concept of conditional independence among variables and by representing these local interactions. Each stochastic variable X_i is directly influenced by at most k other variables, referred to as $Parents(X_i)$. The remaining variables carry no informational relevance to X_i once the relevant ones are known. The probabilistic model may be structured in terms of direct influences among the variables, quantified by conditional probability tables (CPT) $\mathbf{P}(X_i|Parents(X_i))$. By integrating the notion of conditional independence, the required number of independent numeric values is drastically reduced and, if the model is causally constructed, the expert task of assessing each variable CPT is simplified, since the resulting model represents the expert natural way of thinking about the problem.

A Bayesian belief network is a directed acyclic graph, whose nodes are the model variables and whose links represent local causal dependencies. The network topology can be thought of as an abstract knowledge base that holds independently of the numerical assignment of conditional probabilities [2–4, 15–18]. A fully specified Bayesian network can be used as a probabilistic inference engine, which computes the posterior probability distribution for a set of query variables given the probability distribution for some evidence variables.

Decision networks [2–4], combine belief networks with two additional nodes: a decision node, which represents the choices available to the agent and has its value imposed to represent actions, and a utility node, which represents the utility function to be optimized. Evaluating an action amounts to impose the value of the decision node, which, on a non–deterministic environment, may have several different outcomes. This setting alters the probability distribution of a set of stochastic variables in the network, resulting on the probability distribution over all possible next states of the environment for that action. To be able to select

among different actions, an agent assigns utilities to all the different possible states. Utility theory states that the agent will prefer states with higher utility. The expected utility of each possible action can be computed by weighing the utility of each of the various possible outcomes of that action with the probabilities that these outcomes may occur. An agent exhibits rational behavior if it selects the action that yields the highest expected utility.

Since the numerical parameters required to quantify all the CPTs can be substantial and hard for an expert to assess, the initial assessments of probabilities for some stochastic variables U_i can be updated whenever new data is available. At each inference step n the agent observes the environment state as perceived by its sensors. The observed variables are represented by the evidence vector $\mathbf{E}$. The inference algorithm updates the belief given $\mathbf{E}$ on all stochastic variables $\mathbf{P}(U_i|\mathbf{E})$. These values can be used to update the probabilities $\mathbf{P}(U_i)$ using a process known as sequential updating [13, 19, 20]:

$$\mathbf{P}_{n+1}(U_i) = \mathbf{P}_n(U_i) + \alpha * [\mathbf{P}_n(U_i|\mathbf{E}) - \mathbf{P}_n(U_i)] \ . \tag{1}$$

This is a crucial capability, allowing the development of systems that can overcome errors in their initial model and adapt to changes in the dynamics of the environment being modelled.

3 Applying Decision Networks to a Dynamic Scheduler

On a decision network two different subsets may be identified: one related to modelling the current state, the other, referred to as the transition model, related to the outcome of actions and the environment's next state.

The Environment's Current State. The scheduling agent acquires information about the environment through its sensors and infers its belief distribution on the state of the world. Imperfect information about the stochastic variable X, representing some metric of the world, can only be obtained through perfect information about the sensor readings E and the sensor model, $\mathbf{P}(E|X)$ – which is a causal relationship: X determines the sensor readings, not the other way around. The inference process uses Bayes' theorem to compute the probability distribution over X, $\mathbf{P}(X|E)$. The possibility of noisy or incorrect sensor readings are accounted for on the sensor model. Information age, i.e.the time elapsed since the sensor was actually read, T, can be included on the model and influences the belief distribution over the sensor's readings, $\mathbf{P}(E|X,T)$. The prior probabilities associated with the metrics, $\mathbf{P}(X)$, are ideal candidates for sequential updating, since these are the quantities that actually describe the environment state. Higher level variables can be used to model abstract concepts, such as the distributed system degree of load balancing or the communication network availability (Fig. 1).

The Environment's Next State. The outcome of a scheduler's action depends not only on the current state and on the selected action, but also on the dynamics of the environment itself. These uncertainties are modelled by the CPTs related to the variables that represent the next state N_s, given the current state C_s and the selected action a, $\mathbf{P}(N_s|C_s, a)$. This set of CPTs is referred to as the state transition model.

Generic Structure. Figure 1 presents a generic structure for a scheduling decision network, whose topology complies with causal relationships: the sensors readings are an effect of the relevant quantities actual values (information age nodes have not been displayed for simplicity reasons), the next state is a function of both the current state and the selected action (factors external to the scheduler are modelled in the state transition model). Four different blocks can be identified:

- the resources capacities, the tasks requirements and the scheduling overheads model the agent's belief on the environment current state;
- the decision variable lists all the actions available to the scheduling agent;
- the next state variables model the belief on the outcome of each action;
- the utility variable computes the expected utility for each action.

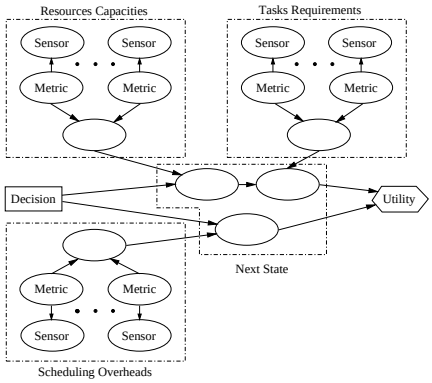

Fig. 1. A generic structure for a scheduling decision network

The variety of available sensors determines the type of information that can be included in the execution model. These can include the node computing throughput, the network latency and bandwidth, the work completed at the sampling time, the communication volume, the available memory, etc. The actions available to the scheduler, represented by the decision variable, depend on the workload and on the distributed system being managed; these may include: assign task T_1 to node n_2, migrate task T_2 from node n_3 to node n_5, transfer 35% of node n_1 workload to node n_4, etc. The next state is composed by a set of

variables that describe the most relevant aspects of the environment, after each action is executed. These should be quantities that are affected by the actions being selected, since the agent uses them to quantify the desirability of each action. These can include the tasks estimated new completion times, the resulting degree of load balancing, the actions expected overheads, etc. Finally, the utility variable represents the performance goals degree of achievement for each action.

4 Results and Concluding Remarks

Experimental Setup. To validate this approach experiments were conducted using a parallel ray tracer with image space decomposition. Four different scenes were used: balls3, balls3c, teapot9 and balls4pv[1]; these are listed by increasing order of computational and communication requirements. To simulate different levels of system sharing, four predefined synthetically generated stochastic background workload patterns were applied: dedicated (no sharing), light, medium and heavy background workloads. The evaluation results were obtained on a cluster of seven workstations, interconnected by a Myrinet network. The results obtained with the decision network based scheduler (DN) were compared with other decision making mechanisms: a static uniform work distribution, a demand driven work allocation (dd), and a deterministic sensor based strategy (det) with the same capabilities as the stochastic strategy, to enable direct comparisons between deterministic and stochastic approaches of identical complexity. The performance metric used to compare the various schedulers was the time required to render the scenes.

Results Analysis. Figure 2 shows the performance improvements for each dynamic scheduling strategy compared to the uniform work allocation for the two most complex scenes and different levels of system sharing. The performance improvement is computed as $(T_{unif} - T_{sched})/T_{unif}$, where *sched* refers to each of the scheduling strategies and $unif$ refers to the uniform work distribution.

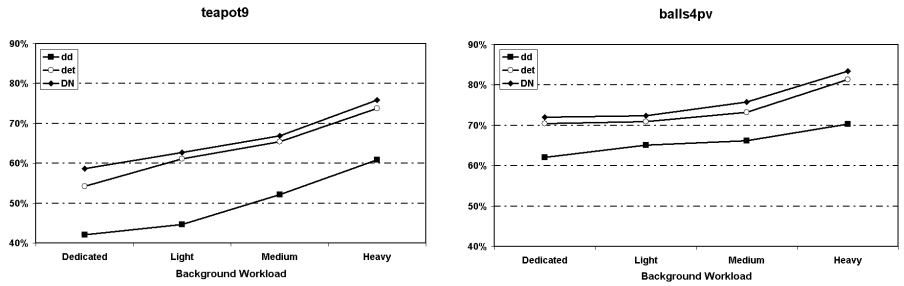

Fig. 2. Performance improvement

[1] taken from Eric Haines rendering benchmarks (ftp.princeton.edu)

Some remarks can be made from this figure:

- the improvements obtained with any of the dynamic scheduling strategies increase with the complexity of the data set and with the weigh of the background workload: the dynamic schedulers can redistribute the workload in runtime, counterbalancing changes on the nodes background workload;
- the sensor based dynamic strategies get more effective as the background workload increases when compared to the demand driven approach: the better quality of the generated schedules clearly overcomes the overheads associated with dynamically collecting data;
- the improvements obtained with DN may not seem significantly larger than those achieved with det; however, the difference between them tends to increase with the background workload weigh, the complexity of the data set and the number of nodes (Fig. 3). Additional results [12] also show that these improvements are achieved by reducing both idle times and task migrations.

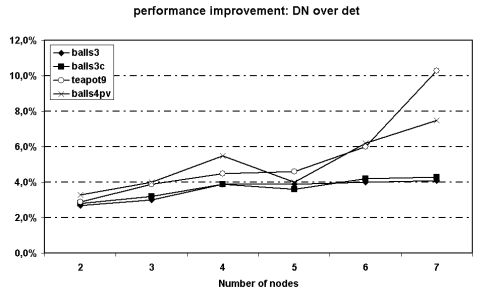

Fig. 3. Performance improvements relative to deterministic (dedicated mode)

Concluding Remarks This paper proposes Bayesian decision networks as the paradigm to handle the uncertainty that parallel application schedulers have about the application requirements, the distributed system state and the outcome of its actions. A generic structure for such a network is presented, including the entities that play a relevant role on the scheduling process. Sequential updating is used to both learn some of the model's numerical parameters and adapt to changes in the dynamics of the environment. The experimental results obtained so far show that a Bayesian decision network based scheduler gets more effective than a deterministic one with identical capabilities when the level of system sharing and the application's workload irregularity increase.

These results corroborate our initial hypothesis: the explicit representation of uncertainty on the scheduler execution model and decision making mechanism, using decision networks, increases its effectiveness. Further research is required to confirm these results with different types of applications and environments.

Additional enhancements may also be considered, namely the assessment of the state transition model. Numerical imprecisions can be corrected by updating the conditional probabilities tables whenever new data is available: either Bayesian learning techniques or reinforcement learning algorithms can be applied to improve the model accuracy.

References

1. Dinda, P., O'Hallaron, D.: Host Load Prediction Using Linear Models. Cluster Computing **3** (2000)
2. Horvitz, E., Breese, J., Henrion, M.: Decision Theory in Expert Systems and Artificial Intelligence. Technical report, Palo Alto Laboratory (1988)
3. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers (1988) ISBN 1-55860-479-0.
4. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall (1995) ISBN 0-13-103805-2.
5. Stankovic, J.: An Application of Bayesian Decision Theory to Decentralized Control of Job Scheduling. IEEE Transactions on Computers **C-34** (1985) 117–129
6. Schopf, J.: Structural Prediction Models for High–Perfomance Distributed Applications. In: Cluster Computing Conference. (1997)
7. Schopf, J., Berman, F.: Stochastic Scheduling. In: SuperComputing'99, Portland, OR, USA (1999)
8. Mitzenmacher, M.: How Useful is Old Information. IEEE Transactions on Parallel and Distributed Systems **11** (2000) 6–20
9. Dahlin, M.: Interpreting Stale Load Information. IEEE Transactions on Parallel and Distributed Systems **11** (2000) 1033–1047
10. Schaerf, A., Shoham, Y., Tennenholtz, M.: Adaptive Load Balancing: A Study in Multi-Agent Learning. Journal of Artificial Intelligence Research **2** (1995) 475–500
11. Zomaya, A., Clements, M., Olariu, S.: A Framework for Reinforcement–Based Scheduling in Parallel Processor Systems. IEEE Transactions on Parallel and Distributed Systems **9** (1998) 249–259
12. Santos, L., Proenca, A.: A Systematic Approach to Effective Scheduling in Distributed Systems. In: VECPAR'2002 – 5th Int. Meeeting on High Performance Computing for Computational Science, Porto, Portugal (2002) 813–825
13. Heckerman, D.: A Tutorial on Learning with Bayesian Networks. Technical report, Microsoft Research — Advanced Technology Division (1995) MSR–TR–95–06.
14. Cowell, R., Dawid, A., Lauritzen, S., Spiegelhalter, D.: Probabilistic Networks and Expert Systems. Springer-Verlag (1999) ISBN 0-387-98767-3.
15. Jensen, F.: An Introduction to Bayesian Networks. Springer-Verlag (1996) ISBN 0-387-91502-8.
16. Jensen, F.: Bayesian Graphical Models. Encyclopedia of Environmetrics (2000)
17. Jensen, F., Lauritzen, S.: Probabilistic Networks. Handbook of Defeasible and Uncertainty Management Systems: Algorithms for Uncertainty and Defeasible Reasoning **5** (2000) 289–320
18. Jordan, M.: Learning in Graphical Models. MIT Press (1998) ISBN 0-262-60032-3.
19. Friedman, N., Goldszmidt, M.: Sequential Update of Bayesian Network Structure. In: 13th Conference on Uncertainty in Artificial Intelligence, Morgan–Kaufmann (1997) 165–174
20. Spiegelhalter, D., Lauritzen, S.: Sequential Updating of Conditional Probabilities on Directed Graphical Structures. Networks **20** (1990) 579–605