Process Mapping Given by Processor and Network Dynamic Load Prediction

Jean-Marie Garcia, David Gauchard, Thierry Monteil, and Olivier Brun

LAAS-CNRS, 7 avenue du Colonel Roche 31077 Toulouse, France
tél : 05 61 33 69 13, fax : 05 61 33 69 69,
jmg@laas.fr

keywords: task allocation, observation, prediction, Round-Robin, network of workstations

1 Introduction

This article describes a process mapping algorithm for a virtual parallel host. The algorithm is a part of a mapping and virtual machine resource watcher tool: the Network-Analyser. This system collects information about the states of the hosts (CPU, memory, network traffic, etc), saves some of this information, and gives the ability to use different mapping algorithms. All of this is accessible by a graphical interface and an API [1]. The algorithm uses both the current and the average load of the hosts and the network, so that it is able to estimate the execution time of the tasks. The model of the processor and the network operation is based on a "round robin" model. Next, a differential equation describes the transient behaviour of the queues. The running time of regular parallel applications can be estimated by the integration of that equation, taking into account the stochastic load of the hosts and the networks. This algorithm is compared to other algorithms, by making event simulations.

2 Load Prediction Mapping Algorithm

2.1 Model of a Loaded Processor

The model of a loaded processor can be made using a "round robin" scheme [2]. Let's assume that Q is the time quantum, λ the poisson arrival rate, σ the leaving probability, and $\mu = (1 - \sigma)/Q$. This system can be described using Markov chain theory. Even though this model is far simpler than systems like UNIX, it gives a good description of the behaviour of the hosts. Assuming that $P_i(t)$ is the probability to have i processes in the system at time t, and $X(t)$, $\dot{X}(t)$ are the expectancy and its variation at time t, we get the equation (1).

$$\dot{X}(t) = \lambda - \mu(1 - P_0(t)) \tag{1}$$

If $t \to \infty$ in equation (2.1), it can be seen that we get the stationary case. Then we approximate $(1 - P_0(t))$ using the expression $\frac{X(t)}{1+X(t)}$. Thus we obtain an autonomous equation (2.2) in which $X_a(t)$ is the approximation of $X(t)$.

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 291–294, 1999.

$$1 - P_0(t \to \infty) = \rho = \frac{X(t \to \infty)}{1 + X(t \to \infty)}, \frac{\partial X_a(t)}{\partial t} = \lambda - \mu \frac{X_a(t)}{1 + X_a(t)} \qquad (2)$$

For a real host, coefficients λ and μ are time-dependant. We observed our laboratory's hosts for several months. So that we have been able to distinguish two different types of behaviour according to the hosts: we use the current value to predict the near future behaviour:

- special: the current load of the host is very different from its average load for this day of the week, or the host has a constant load (perhaps no load).
- cyclic: the current load is close to the average load for this particular week day and time: we use an estimator that will smoothly forget the current value and which will approach the average values of the last few weeks.

2.2 Execution on Processor Time Prediction

We wish to predict the deterministic process execution time in the stochastic model. Let's say that we have a process to run at date t_0 that needs t_c processor units, and there are $X(t_0)$ processes in the system. We would like to find out the exit time expectancy $E(t_f)$. UNIX has a time shared policy, and for a long enough process, the UNIX mechanisms (scheduler, priority, ...) have a light influence. So that we can write the relation (3.1). It can be more general for N processes that verify $t_c^1 \leq t_c^2 \leq \ldots \leq t_c^N$ (equation (3.2)[1]) ($t_f^0 = t_0$).

$$t_c = \int_{t_0}^{E(t_f)} \frac{1}{1 + X(t)} dt, t_c^j = \sum_{i=0}^{j} \int_{E(t_f^{i-1})}^{E(t_f^i)} \frac{1}{(1 + X(t))(N - i + 1)} dt \qquad (3)$$

2.3 Network Modelisation

Network packets may have a variable size, and are serialized into a wire. This allows us to model the network as in the above scheme. In this case, the time given by the server (network) to the client (emitted message) to process a packet is the time needed to transmit the packet over the wire. We noticed with the Network-Analyser that the network behaviour is nearly the same as the cpu behaviour. Thus we are able to use the same prediction method for the network.

2.4 Task Mapping over the Network

We use an application model in which there is a rendez-vous between all the tasks at each calculus step (figure 1). Let's assume that N is the number of tasks, K the number of steps, M the number of processors, $t_i(k)$ the date when the step k ends, $t_c^n(k)$ the time used by the calculus step of the task n during the step k, $t_r^n(k)$ the time needed by the communication step of the task n during the step k, and $t_s^n(k)$ the time needed to synchronize between the task n and with the others. Mapping the task in the optimal way can be done if we know

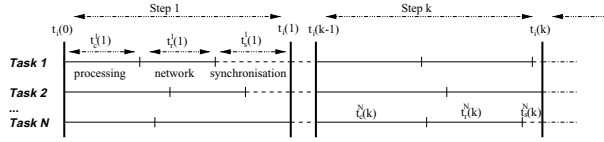

Fig. 1. Application model

the optimal mapping function $\hat{P}$, where $P : N \longrightarrow M, n \longmapsto m$ is a function which associates a task n with a host m. Let φ be the ensemble of the mapping function P. At step k, we have:

$$t_c^n(k) + t_r^n(k) + t_s^n(k) = t_i(k+1) - t_i(k) \tag{4}$$

For a given task n of the application, we look for $\hat{P}$ such that the total time of the application is a minimum:

$$\min_{P \in \varphi} \sum_{k=1}^{K} E(t_c^n(k) + t_r^n(k) + t_s^n(k)) \tag{5}$$

Using the results of the equation (3), we can evaluate $E(t_c^n(k))$ and $E(t_r^n(k))$. To find $t_s^n(k)$, one can use the following two relations ((6.1) and (6.2)).

$$t_i(k) = \max_{n \in 1 \ldots N} t_c^n(k) + t_r^n(k), t_s^n(k) = t_i(k) - (t_c^n(k) + t_r^n(k)) \tag{6}$$

For speed reasons, we use an heuristic that comes from the LPT algorithm [3] to determine an approximate value of $\hat{P}$: Tasks are mapped one-by-one. Ghost communications with still non mapped tasks are assumed to take into account the network even from the very beginning of the algorithm. These are mapped on the fastest links connected to the processor which has the current task.

3 Simulations

3.1 System Description

The network on which we did our simulations consists of two clusters of four processors each. The hosts of the first cluster are interconnected by a 10Mbits/s link, while the others connected to both the first link, and a 1Gbits/s link. Each application is a group of six tasks each having five calculus steps, and communications between them are all-to-all. The type of the algorithms we tested are random mapping, cyclic mapping, minimum run queue mapping, processor load prediction, and network and processor load prediction. We use granularity 20 for 5 times faster processors on the fast network, and granularity 2 for homogoneous processors.

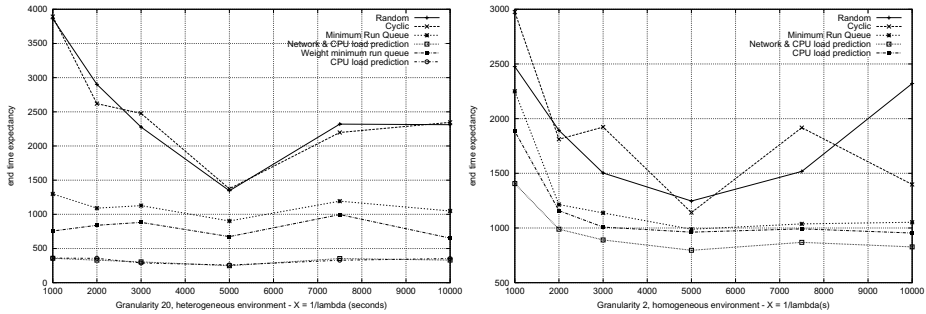

Fig. 2. Simulation results

Mapping policies become critical when the whole system is not well balanced, which is frequent. Figure 2 (granularity 20) shows that the applications with large granularity are badly mapped with the blind algorithms. Though, policies taking into account the current state of the system are much more efficient (closed loop policy). The proposed algorithm (and its simpler version not taking into account the network) has a better behaviour. Generally, one can say that the minimum run queue algorithm is a correct heuristic and very easy to implement. The network and processor load prediction algorithm is more complex to implement but offers a better guarantee for efficient use of the resources.

4 Conclusion

Sytems like the Network-Analyser are essential for an efficient parallelisation. The mapping algorithms presented in this article show their superiority compared to the classical algorithm we usually use - thanks to their precision over the execution time. The main point is the use of time in the decision model, assuming the distributed system as a stochastic system; the proposed algorithm takes into account heterogeneous communication networks.

References

[1] T. Monteil : Etude de Nouvelles Approches pour les Communications, l'Observation et le Placement de Tâches dans l'Environnement de Programmation Parallèle LANDA. Thèse, LAAS-CNRS, France, novembre 1996.

[2] E.G. Coffman, P.J Denning : Operationg Systems Theory. Prentice-Hall series in automatic computation, 1973.

[3] R.L. Graham : Bounds on Multiprocessing Timing Anomalies. SIAM J. Appl. Math.. Vol 17, No. 2, mars 1969.