Performance Prediction for Parallel Reconfigurable Low-Level Image Processing

Martin Fleury and Adrian F. Clark
Image Processing Group
Dept. Electronic Systems Engineering
University of Essex, Colchester, UK
fleum@essex.ac.uk

Abstract

Performance models are presented for affordable parallel processing of images. A generic topology is used for pixel-based processing. A novel parallelization of an image rotation routine is given as a case study.

1 Introduction

Message-passing multiprocessors, while being more convenient to program than SIMD machines, have an unfavourable communicate to compute ratio for low-level image processing which makes satisfactory data-parallelism difficult to achieve. One solution is to manipulate the hardware to allow run-time reconfiguration under software control. The system should also be applicable to those algorithms (*e.g.*, co-ordinate transforms) where global communication is necessary. Accurate performance prediction is necessary in order to be able to optimize the number of processors, their interconnections, and the way in which reconfiguration is employed; it is also essential in order to be able to apply the same design principles to systems based on other processors or with different speeds.

The next section describes briefly the reconfigurable system employed in this work. The following section outlines an approach to performance prediction based on queueing theory. Section 5 compares the theoretical prediction with experimental timings for image rotation, a typical non-neighbourhood image processing operation.

2 A Parallel Reconfigurable System

Our experiments make use of a Parsys Supernode with twenty-four T800 transputers and associated image capture and display hardware. The Supernode achieves reconfiguration by means of a programmable crossbar switch (P1085). All software is written in 3L Parallel C, using library calls to implement the Communicating Sequential Processes (CSP) model of parallelism. Software has been developed to enable arbitrary run-time reconfigurations of the worker transputers under program control.

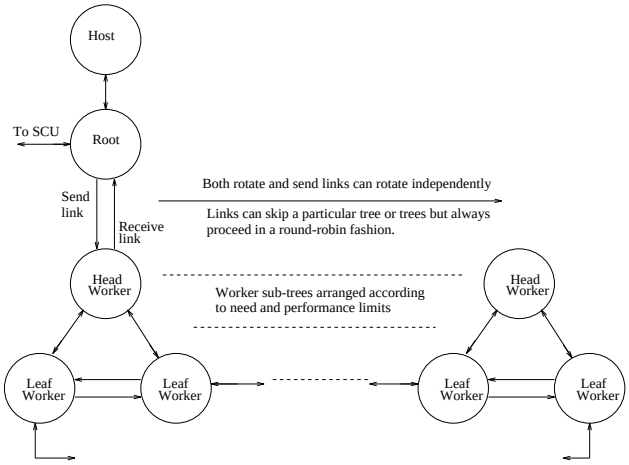

Figure 1: Transputer Configuration used in Timing Experiments

The basic approach used is similar to a process farm. The transputer configuration employs multiple worker networks, with a master that uses reconfiguration to switch links from one worker network to another in a round-robin manner— see Figure 1. Reconfiguration is achieved by sending messages over a transputer link to the Supernode's switch control unit (SCU). In order to avoid excess internal data movements, semaphores were also employed. The send and receive links rotate independently, within limits, to minimize synchronization costs. Synchronization with the head of each network was achieved by centralized polling. A large buffer at the head of each network was used to absorb arriving data in the absence of the rotating links.

3 Performance Prediction for Non-deterministic systems

Classical queueing theory commonly used for predicting the performance of telecommunications networks, can act as a starting point for estimating the potential delay suffered by returning packets of work in a computer system. In our system, it is necessary to identify the server as the rotating link servicing the head of each network and with the arrival rate being determined by the finishing times of each packet of computation. There is one requirement for this model: the arrival rate should be memory-less or Markovian, resulting from a computation rate which is variable in the Poisson sense. An attractive prospect is to use the M/G/1 model (in Kendall notation), where M requires exponentially-distributed arrival rate whilst G could be a deterministic service distribution governed, in the reconfigurable case, in part by the link transmission and switching times. This model has the advantage that there exists an extension to the Pollaczek-Khintchine formula which accounts for server vacations *i.e.* the absence of the link from a particular network. The relevant result is

$$N = \rho + \frac{\lambda^2 E[(S+\hat{S})^2]}{2(1-\lambda E[S+\hat{S}])} + \frac{\lambda E[\hat{S}^2]}{2E[\hat{S}]}$$

where $\hat{S}$ is the length of the vacation by the link S is the service length of the links. N is the mean number of packets in a queue, which is related to the mean waiting time by Little's theorem ($W = N/\lambda$), $\rho = \lambda E[S]$ is the availability, and λ the arrival rate. This result, which has been used for other multiprocessor systems [1], requires the vacation time distribution $\hat{S}$ to be known and limits service to one packet each time. It is possible to find this (see Section 4), but we have found it more convenient to use a result previously applied to token-ring networks *inter alia*. The revised model is [3]:

$$W = \frac{M\lambda E[S^2]}{2(1-M\rho)} + \frac{(1+\rho)Mr}{2(1-M\rho)}$$

Here, M is the number of networks and r is taken to be a constant switching time. This equation is for a gated service discipline but there exist a number of variants, including those for asymmetric arrival rates. We found S by measurement and then compared the prediction from the formula with actual results in order to validate the model — see Section 5. It is important to note that where several independent streams of packet arrivals contribute to one stream then the resultant distribution will be Poisson. This will be the case for packets from several processors arriving towards the head of the network.

4 Performance Prediction for Deterministic Systems

If the packet arrivals are not independent, a model based on queueing theory is not formally appropriate; this would, for example, be the case where demand-based data-farming is used. Instead, it is possible to use a linear programming approach utilizing the physical parameters of the hardware [2] for a single network. This is perhaps best illustrated by a simple example. Consider a linear chain with S_i being the throughput (in work packets per unit time) of the $i^{\underline{\text{th}}}$ processor. The upper-bound conditions on throughput are

$$\begin{aligned} (t_{comm} + t_{setup})S_n &\leq 1 \\ (t_{calc} + t_{setup})S_n &\leq (t_{calc} - t_{setup})S_{n-1} + 1 \end{aligned}$$

with output at the head of the chain being from S_n. Solving the recurrence, using the sum of a geometric series, yields

$$\frac{1}{(t_{comm} + t_{setup})} = \frac{1}{2t_{setup}} \left(1 - \left(\frac{t_{calc} - t_{setup}}{t_{calc} + t_{setup}}\right)^n\right)$$

The left-hand side of this expression represents the condition of link saturation to the head of the chain. The right-hand side must equal this to achieve maximum throughput.

The above analysis gives the throughput for one network without accounting for link absence or vacation while servicing other networks. The following analysis is one way of arriving at the steady-state delay. The service delay time experienced by each network is composed of two factors: the switching time, Mt_{switch}, where M is the number of networks and t_{switch} is the time to make one switch between two networks; and delay caused by waiting for the previous chains to be emptied of any outstanding packets. The delay depends on the number of packets arriving at the networks previously visited by the reconfigurable links since the current network was last emptied. For given throughput T we found a condition on the waiting time W at each network if convergence has occurred:

$$W = \frac{Mt_{switch}}{1 - (M-1)(t_{comm} + t_{setup})T}$$

5 Performance Results

The system described in Section 2 was used to assess the queueing theory model discussed above. The transputers operate at 20MHz and their links at 10MHz, giving a net transfer rate of 0.9Mb/s. The time taken to switch a link is 0.39ms; for comparison, the time to transfer 1Kb of data is 1.09ms as recorded on the transputer. The routine used for an exemplum is image rotation, which has a fixed

Angle (degrees)	Timing in secs: Number of Processors			
	9	12	15	18
10	5.28	4.13	3.54	3.26
20	6.12	4.77	4.45	4.53
30	6.87	5.63	5.61	5.73
40	7.21	5.80	5.69	6.49
50	7.56	6.80	6.56	6.74
60	7.77	5.94	6.34	6.92
70	7.17	6.56	6.72	6.93
80	6.37	5.98	6.35	6.94

Table 1: Timings for various angles of image rotation for a 1024×768 pixel image

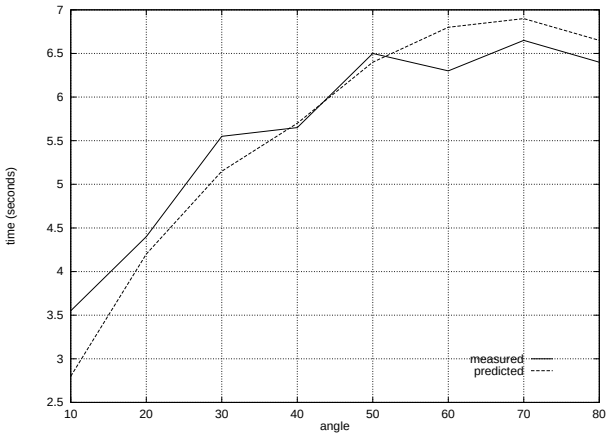

Figure 2: Comparison of measured performance with that predicted by queueing theory (15 processors)

pattern of data transfer. The algorithm in [4] was used, which partitions the rotation into three shears. The first two stages of the algorithm may be performed on the same processor but, for the transition from vertical to horizontal shear, it is necessary for each processor to make a one-to-many data transfer. In fact, the number of messages passed depends on the angle of rotation and the way in which the data are partitioned between the processors. A key problem for this type of algorithm is that each processor must know how many messages to expect before it can begin the last phase of processing. The solution found was to use an auxiliary program to pre-calculate the number of messages which any processor might expect, the totals being stored in a look-up table.

Timings for the algorithm are given in Table 1, where it will be seen that, when an image has been split too finely and the shear is large, the performance deteriorates if more processors are used. The timings include the cost of collecting the image but not of initially distributing it.

It was found that the predicted performance using the deterministic model of the system did not match these results well. However, the match with the non-deterministic model is reasonable (Figure 2). These results are promising, since image rotation is an algorithm that one might think would not parallelize well because of the quantity of messages transmitted over the rotating links, which can be in the thousands. Distribution of the messages is overlapped with calculation of the vertical shear movements, which explains the reasonable efficiencies arrived at, for instance over 80% for 9 processors at an angle of 10^o, and over 50% with 15 processors for an angle of 40^o.

6 Conclusions

In this paper, we have presented practically-tested mathematical models that allow the performance of parallel reconfigurable systems to be predicted on both deterministic and non-deterministic bases, the latter derived from classical queueing theory. Queueing theory is really intended for describing the performance of a complete system when it has reached its steady state; hence, allowance should be made for the initiation and termination phases of an algorithm. Buffering requirements are a separate issue.

Acknowledgements

This work was carried out under IED project IED3/1/2171, "Parallel Reconfigurable Image Processing Systems."

References

[1] E. Gelenbe. *Multiprocessor Performance.* John Wiley and Sons, 1989.

[2] D. J. Pritchard. Mathematical Models of Distributed Computation. In *7th Occam User Group Technical Meeting*. IOS Press Amsterdam, 1987.

[3] H. Takagi. Queueing analysis of polling models. *ACM Computing Surveys*, 20:5–28, 1988.

[4] A. Tanaka, M. Kameyama, S. Kazama, and O. Watanabe. A Rotation Method for Raster Image Using Skew Transformation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 272–277, 1986.