Performance Estimation for a Dynamically Reconfigurable Multiprocessor System Applied to Low-Level Image Processing

M. Fleury, L. Hayat and A. F. Clark
Dept. E.S.E., Essex University, Wivenhoe Park, Colchester, CO4 4SQ
fleum@uk.ac.essex

Abstract:

This paper describes the performance testing of a parallel, fixed-valency processor system, which is primarily intended for low-level image processing, though the techniques developed have wide applicability. By tweaking the hardware of a Parsys Supernode it has been found possible to improve upon static networks, provided the system is correctly tuned. Specifically, in addition to compile-time reconfigurability, run-time reconfigurability is employed in an add-on fashion. The technique can upgrade the performance of store-and-forward message passing processors in a cost-effective way. Demand-based data-farming is used, which can be generalized to a number of algorithms, forming a template. A number of bounds to performance are assessed and a statistical approach to the data-farming programming paradigm is suggested. Some details of the way parameters such as buffer-size and buffer-usage determine performance are given. A number of formulae, which may succinctly capture the correct mode of operating the system, are also shown.

1 Introduction

It is possible to use reconfigurability as a low-cost way of avoiding the restrictions of a fixed-valency processor. Software has been developed [FH93] to allow runtime reconfiguration of a transputer [Inm89] network running under the ægis of the Parsys Supernode [Par89]. This multi-user machine running a Unix-like operating system, Idris, can also support several reconfigurable networks. The aim was to enhance the performance of a variety of low-level or pixel-based image processing algorithms, typically but not exclusively involving local-windowing. A convenient programming paradigm would be demand-based data farming if it were possible to ameliorate the bottleneck caused by access to the farmer processor. Apart from automatic load-balancing, data-farming has the advantage of allowing the overlap of communication with computation. A generalized and practical system has been developed using reconfigurable access, the ancestor of such a scheme being [HT90]. The merits of a number of methods whereby performance can be bounded on such a system are adumbrated.

2 The System Set-up

The farmer processor on the present system uses two independently rotating links to connect to a number of small worker networks, figure 1. In the absence of the link used to receive processed work-packages, a buffer is provided at the head of each worker network. The farmer effects a reconfiguration by using one of its links to signal to the Switch Control Unit (SCU), which in turn is connected to a programmable multi-crossbar switch, the P1085 switch [Har87]. A link can be swapped in 0.04ms, while transmitting 1K bytes over 1 hop in simplex mode takes about 3 times this. Data packets of at least 3K bytes size are commonly used in this application.

Apart from the master-buffer (MB) at the head of the chain, local buffering (LB) was used to control in and out flows to the application process resident on each processor. A router buffer (RB) was also found to be helpful. At least one packet was kept in transit so that demand for work could in part be satisfied without returning to the farmer. The worker network topology varied between a linear chain and a tree structure.

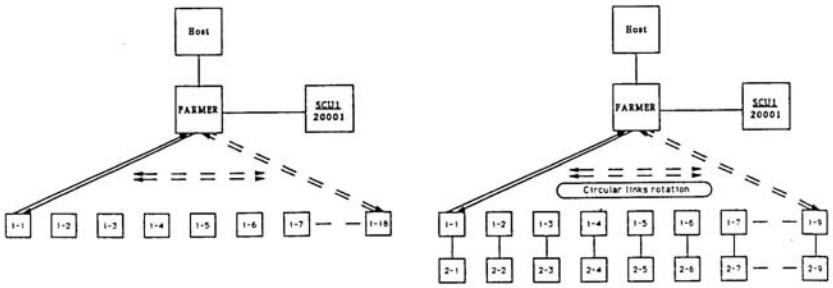

Figure 1: *Reconfigurable multi-chain*

3 Testing the System

The image was divided into strips which made addressing convenient. As the size of the strip is increased, the ratio of border lines to processed strip decreases allowing an economy of size to be made. Table 1 (EP vs. Performance) shows that, with other factors being equal, beyond 12 lines the advantage from this tactic reaches an asymptote. Note that all timings are for ease of representation in ticks, where a tick is at the resolution of the transputer low-priority clock of 15625 ticks/sec. If LB has only one slot then latency will effect performance since the return journey to request and replenish work can exceed the time to process the remaining packet. The next table (LB vs. Performance) shows that providing extra slots will also have limited effect once the buffer size balances the journey time, particularly if semaphores are not used to restrict the number of internal data movements. A distributed algorithm was used to arbitrate the allocation of work-packets, whereby priority was given to requests for work from processors at a greater number of hops from the farmer. While this can balance bandwidth against work-load it can also lead to processors repeatedly pre-empting work destined for other processors nearer the farmer. Use of a small routing buffer, RB, to provide uncommitted packets can ease this problem, but once again the value of doing this rapidly tails off, as in the table of RB vs. Performance. Increasing the size of MB, if it means that the farmer is committed to removing packets from the buffer, is also of limited effect since a larger buffer will delay the farmer from servicing other networks. Unlike the previous tables, MB vs. Performance includes the case of three chains as opposed to one single chain of nine processors. Table 2 shows how varying the number of chains but keeping the total number of processors constant reaches an optimum which is a balance between extending in the vertical and horizontal directions. Changing the topology to a tree arrangement was advantageous mainly because the use of links on a transputer is more efficient in duplex mode than it is in simplex mode, due to careful multiplexing on the single channel.

A single application instance was run on each processor, with a separate routing task co-resident on the processor (figure 2). The router was run at high-priority to avoid blocking. It was not found advantageous to decouple routing decisions from reception and transmission because it involved extra internal data movements. If threads or light-weight processes are used, by means of 3L C [3L 91], then it is possible to control access to data by means of semaphores, provided the processes are not at different priority. A further experiment revealed that removing the application from processors higher up the tree so as to increase the bandwidth was too drastic a solution.

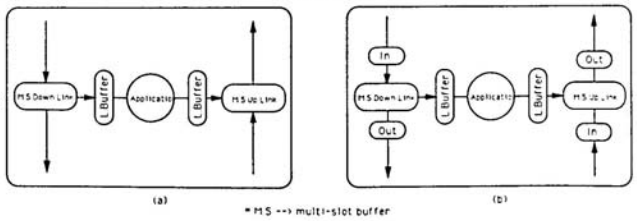

Figure 2: *Task Structure -various options.*

Table 1: *Performance vs. physical parameters*

EP vs. Performance						
MB	RB	LB	EP	SPEs	M.ELP.W	ELP.F
5	1	1	0	9	190428	192035
5	1	1	1	9	104077	106323
5	1	1	2	9	92152	94511
5	1	1	3	9	86325	88965
5	1	1	4	9	83869	86559
5	1	1	5	9	81430	84648
LB vs. Performance						
MB	RB	LB	EP	SPEs	M.ELP.W	ELP.F
5	1	2	0	9	190768	192163
5	1	2	1	9	109434	111153
5	1	2	2	9	82800	85479
5	1	2	3	9	71009	73375
5	1	2	4	9	62977	65667
5	1	2	5	9	52559	56632
5	1	3	0	9	191404	192800
5	1	3	1	9	110463	112523
5	1	3	2	9	84336	86594
5	1	3	3	9	71615	74795
5	1	3	4	9	65496	68451
5	1	3	5	9	58334	62859
RB vs. Performance						
5	2	1	0	9	108349	109745
5	2	1	1	9	62986	64706
5	2	1	2	9	50594	52637
5	2	1	3	9	50731	53098
5	2	1	4	9	50388	53078
5	2	1	5	9	51844	54857
5	2	2	0	9	109352	110749
5	2	2	1	9	64018	65738
5	2	2	2	9	50591	52635
5	2	2	3	9	50731	53098
5	2	2	4	9	50389	53079
5	2	2	5	9	51843	54855

RB vs. Performance (continued ...)						
MB	RB	LB	EP	SPEs	M.ELP.W	ELP.F
5	2	3	0	9	109852	111248
5	2	3	1	9	65048	66767
5	2	3	2	9	52123	54167
5	2	3	3	9	50729	53096
5	2	3	4	9	52908	55598
5	2	3	5	9	51841	54854
5	2	4	0	9	110879	112276
5	2	4	1	9	66079	67799
5	2	4	2	9	52124	54167
5	2	4	3	9	52758	55125
5	2	4	4	9	52909	55598
5	2	4	5	9	54891	57905
5	3	1	0	9	109857	111253
5	3	1	1	9	65050	66770
5	3	1	2	9	53659	55703
5	3	1	3	9	54791	57158
5	3	1	4	9	55429	58120
5	3	1	5	9	57946	60960
MB vs. Performance						
5	2	2	2	1	47573	52272
10	2	2	2	1	47008	58045
15	2	2	2	1	46651	65148
20	2	2	2	1	62034	70560
25	2	2	2	1	77553	78256
5	2	2	2	3	49058	50073
10	2	2	2	3	49058	50196
15	2	2	2	3	49059	52842
20	2	2	2	3	52126	52888
25	2	2	2	3	53660	54791

** Timing measured in "ticks"
** Image size = 1024 x 768

MB = Multi-slot master buffer
RB = Multi-slot router buffer
LB = Multi-slot local buffer
EP = No. of extra image lines above minimum 7 lines

9 Worker processors used throughout
SPEs = Worker processors in a single column
M.ELP.W = Maximum elapsed time at any worker
ELP.F = Maximum elapsed time at farmer processor

Table 2: *The Effect of Topology*

Chain: Static vs. Reconfiguration					
WPEs	SPEs	WIN=9	WIN=7	WIN=5	WIN=3
18	1	113931	57660	45565	34898
18	2	94881	52234	41279	31124
18	3	93922	51085	40250	30339
18	6	94779	49890	40009	30064
18	9	96942	49941	40812	31347
18	18	102522	53025	40879	31241
Tree Performance					
18	3	88656	49787	39460	29682
18	9	90011	48481	38862	29381

4 Predicting Performance

While testing the system gives useful heuristics and experience, it would be preferable to have some method of predicting performance in advance. A commonly-used method [FY91] of estimating the processor requirement is based on finding the granularity, G, of the algorithm. The granularity is the ratio of number complexity to data requirements. This can be applied to

$$p \leq G\frac{F}{C.ch} \tag{1}$$

to find p the maximum number of processors, where ch is the number of channels available for the output of results, C is the communication time per word per second, and F is FLOPS for a particular processor. However, using this method fails to account for set-up time on a store-and-forward transputer network, resulting in an over-estimate of the number of processors which can be used before performance speed-up declines. A more refined method [Pri87] is to use linear programming to predict the throughput achievable based on the physical parameters of the processor. In particular, throughput is bounded by the capacity of the final link to the farmer. Where each work packet is the same then the solution for a linear chain has a closed form:

$$p = \frac{\log\left(\frac{t_{comm}-2t_{setup}}{t_{comm}+2t_{setup}}\right)}{\log\left(\frac{t_{calc}-2t_{setup}}{t_{calc}+2t_{setup}}\right)}. \tag{2}$$

While it is not possible to achieve more efficient performance with more than p processors, a more realistic upper bound should take into account software costs. For instance, using the distributed protocol (section 3) can result in a stream of work to the lower processors in a linear chain being set up whilst the chain is initially being loaded. The effect is reinforced if the packet size returning is less than that arriving, because the communication time is less. In such circumstances a formula for the number of processors which can become fully active might be:

$$\frac{t_{calc} + (p-1)t_{back}}{t_{arrive}} = bp - 1 \tag{3}$$

with b being the local buffer size.

The performance indicators mentioned so far do not provide a method of estimating the likely effects of reconfiguration. With varying job size it is also possible for link congestion to occur, giving the system more of a stochastic nature. In these circumstances a promising method may be to apply classical queuing theory, particularly as there is a convenient formula [Kin90] which can be adapted to the case of rotating links. This formula is an extension of the Pollaczek-Khintchine formula to account for a server taking vacations or absences. The relevant result is

$$N = \rho + \frac{\lambda^2 E[S^2]}{2(1-\rho)} + \frac{\lambda E[\hat{S}^2]}{2E[\hat{S}]}, \tag{4}$$

where $\hat{S}$ is the length of vacation by the link, and 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. (ρ is the availability and λ is the arrival rate.) The server in this instance is not the farmer processor but the circulating links. A suitable extension is to use a cyclic polling model for multiple queues [Tak88]. Delay experienced at one worker network is recursively dependent on the delay at previously-visited networks. For convergence to occur it is found that

$$W = \frac{nt_{switch}}{1-(c-1)r}, \tag{5}$$

where W is the waiting time on a network, n is the number of networks, and $r = T(t_{comm} + t_{setup})$ with T, the throughput, previously established using a linear programming method.

In one experiment it was found that 12 processors out of 18 arranged in a linear chain performed useful work on packet sizes timed at 1512 ticks for 3 lines processed out of 9 transmitted in each data packet. Applying the granularity method with $G = 18$, $ch = 1$, with a net data communication rate of $\sim$ 8.5MHz and machine cycle time of 20MHz gave $p = 42$. Note that cross talk on the backplane may restrict the transputer's link speed to 10MHz. Using the linear

programming method with a minimum set-up time of 0.5 and 9K bytes of data per packet, taking 18 ticks to traverse a link, suggested that the final link should saturate with the output provided by 9 processors. Using the estimate of the number of processors that could become active indicated that, with a one-slot local buffer size, 6 processors could become active without packet pre-empting taking place. In practice the number of packets processed by each processor did drop off after 7 processors to half that of the peak value. The maximum packet round trip was 3900 ticks, while each processor had work for only 3000 ticks, indicating that latency was also slurring performance. Changing the task-routing structure failed to improve on this result. When the throughput on each chain is calculated for the examples in table 2 for a window size of 3 with a link switching time of 6 ticks, the convergence test indicated that, with 3 processors per chain and 6 chains, the waiting time fails to converge, while with 6 processors per chain it just fails. In practice, using 6 processors per chain was the optimum solution, indicating a somewhat higher throughput than estimated. The present experiments did not provide evidence of link traffic congestion, for which the waiting times provided by queuing theory would have been useful. It is expected that further work with algorithms that are not completely deterministic or cyclic will utilize this result.

5 Conclusion

order to tune a reconfigurable system of the type described here it is necessary to refine ..thods which give a bound to performance based on hardware limitations and include the effect of software organization to gain a more accurate estimate of the actual potential performance. In practice achieving a balanced load whilst the links rotate involves balancing switching time and throughput on worker networks.

Acknowledgements

This work was carried out as part of project IED3/1/2171 ("Parallel Reconfigurable Image Processing Systems").

References

[3L 91] 3L Ltd, Peel House, Livingston, Scotland. *Parallel C Version 2.0.1*. 1991.

[FH93] M. Fleury and L. Hayat. Performance on a Reconfigurable Message Passing Parallel Processor using the Data Farming Paradigm. Technical report. Essex Univ., 1993.

[FY91] A. G. Filin and I. E. Yukov. A Block-Pipelined FFT Algorithm on a Ring of Transputers. In *Applications of Transputers 3*, pages 760–761. IOS, 1991.

[Har87] J. G. Harp. Phase 2 of The Reconfigurable Transputer Project-P1085. In *ESPRIT '87 Results and Achievements*, pages 583–591. Elsevier Science, 1987.

[HT90] G. Hall and T. J. Terrell. Implementation of the Radon Transform using a Dynamically Switched Transputer Network. In *Proceedings of the Second International Conference on Applications of Transputers Southampton 1990*, pages 156–163. IOS, 1990.

[Inm89] Inmos Ltd, 1000 Aztec West, Bristol. *The Transputer Databook*. 1989.

[Kin90] P. J. B. King. *Computer Communication Systems Performance Modelling*. Prentice Hall, 1990.

[Par89] Parsys Ltd, Boston Road, London. *Hardware Reference for the Parsys SN1000 Series*. 1989.

[Pri87] D. J. Pritchard. Mathematical Models of Distributed Computation. In *7th Occam User Group Technical Meeting*. IOS Amsterdam, 1987.

[Tak88] H. Takagi. Queueing analysis of polling models. *ACM Computing Surveys*, 20:5–28, 1988.