PERFORMANCE MODELLING OF PARALLEL COMPUTER ARCHITECTURES

P.G. Harrison and A.J. Field

Department of Computing,
Imperial College,
London SW7 2BZ

Abstract

In this paper we describe two types of complex server aggregations which can be used to model collections of components in certain types of parallel computer systems and give a case study showing how the aggregations may be applied in practice. Analytical models of such systems are becoming increasingly important as a means of guiding the often complex design processes, particularly since recent developments in VLSI technology now make it possible to fabricate many paper-designs hitherto impractical for reasons of cost. We argue that aggregations of the type described are essential in the modelling of parallel systems; using the proposed techniques, large numbers of components can be modelled as queue-length-dependent servers within a queueing network in which the number of servers is the same as the number of distinct types of processing element in the system being modelled. Because the number of servers in the model is fixed i.e. is independent of the number of processors, very large multiprocessor systems can be modelled efficiently with no explosion in the size of the state space.

1. Introduction

The recent advances in VLSI technology have enabled the development of an entirely new generation of computer architectures hitherto impractical for reasons of cost/performance. At the same time, conventional machines are approaching upper bounds for execution speed - often dictated by natural constraints such as the speed of light. Thus the designs of new architectures often rely on executing the processes which constitute a computation in parallel in order to obtain further increased execution speed. Such an architecture is the ALICE machine, [DR81], the prototype for which is now close to completion and the subject of the case study in this paper. A large number of components are required to achieve significant improvements in performance, resulting in architectural designs which are complex and for which it is therefore hard immediately to estimate performance reliably. At the same time, experimental tests are usually precluded since there rarely exist operational machines with even remotely similar architectures.

As a result, it becomes difficult to determine the optimum hardware configurations and the operating

system algorithms (typically written in microcode) which control them. This is true equally of architectures which are to be cast entirely in VLSI or those which are to use large numbers of existing components assembled on conventional circuit boards. Ideally one would like to first construct the latter as a prototype which can then be used experimentally to determine the best design for the former, which would be rigid in its design once built and typically represent a much larger investment. However, the prototype itself is likely to be costly to build, and will also have its design constrained to some extent by the need to use off-the-shelf components with their own specifications. For these reasons, it is desirable to develop models of proposed physical architectures in order to predict their performance a priori and so to guide the detailed design and optimal paratermisation (e.g. proportion of processor to memory modules) of the system. The prototype might then be omitted entirely if a reliable modelling methodology has been established, or alternatively may be constructed partly in order to validate such models.

A common observation is that in many of the proposed parallel architectures the number of different types of logical component used is typically small, these components being replicated manyfold over the system to achieve concurrency. (By "logical component" we mean a set of of electronic components which collectively perform some logical function within the architecture, for example a microprocessor, DMA controller etc.) In view of the large number of components involved in total, simulation models tend to be hopelessly inefficient, and an analytical model is therefore preferable. There are also other advantages which analytical models have over simulation, namely their computation of precise performance predictions under a given set of assumptions, as opposed to estimates within a certain range of confidence, corresponding to specific test runs. Nevertheless, even analytical models are likely to be slow to execute if every component is represented explicitly.

In this paper we propose reduced queueing network models in which multiple occurrences of the same type of component are aggregated into a single server in the model, thereby reducing the number publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

of servers to the number of distinct types of component, which is typically small. The aggregation defines a single server with a queue length dependent service rate function which yields the same throughput as that of the original collection of components in the steady state. In Section 2, we derive analytical results for two types of aggregation of servers. One is the 'multiple server' which represents a set of identical and equally utilised servers in parallel, each with their own queue, and the second is the 'VBIS server' which represents a similar set of parallel queues in which arrivals are 'awaited' by virtual processes and so enter a reserved (empty) buffer at one of the servers.

In Section 3, we describe a case study showing how the proposed aggregations were used to model the ALICE graph reduction machine [DR81]. Here we show how the machine can be modelled using just five aggregated servers, and how performance measures such as throughputs, mean queue lengths and utilisations can be found.

In Section 4, we propose some enhancements to the ALICE model. The conclusions are laid out in Section 5.

2. Aggregation of a Set of Parallel Servers

2.1 Overall Approach

We consider a subset of servers (the subnetwork) in a product-form queueing network, [BCMP75], specifically in which there is only one class of customer, although generalisation to the multi-class case is straightforward. We use the decomposition technique first used (to the authors' knowledge) in [CHW75] - sometimes known as 'Norton's Theorem' after the result from electrical circuit theory. A subset of servers in a product form queueing network is replaced by a composite single server with service rate function which yields 'equivalent throughput'. More precisely, we consider a network of M servers and without loss of generality take the servers in the subnetwork to be numbered $1, 2, \dots, m$. Let the service rate of server i be $\mu_i(n)$ when its queue has length n ($1 \leq i \leq M$) and let the equivalent composite server have service rate $\mu(n)$ for queue length n . Now consider the closed network consisting of servers $1, 2, \dots, m$ only, with the same visitation rates, i.e. formed by 'short-circuiting' the rest of the original network in the sense described in [CHW75]. We then assign to $\mu(n)$ the value of the throughput of this closed network when its population is n , for $n = 0, 1, \dots$. It is shown in [CHW75] that the joint queue length probability distribution (qlpd) of the reduced network is equal to the corresponding marginal joint qlpd in the original network.

We will use the following additional notation: For $1 \leq i \leq M$,

e_i = visitation rate of server i

$$x_i(n) = e_i \frac{n!}{\prod_{j=1}^n \mu_j(j)}$$

$$\underline{n} = (n_1, \dots, n_M)$$

$$\underline{n}' = (n_1', n_{m+1}, \dots, n_M)$$

$$\underline{k} = (k_1, \dots, k_m)$$

$$S_n = \{ \underline{k} \mid \sum k_i = n, k_i \geq 0 (1 \leq i \leq m) \}$$

(the state space of the short-circuited subnetwork)

$$N_i = \text{queue length random variable for queue number } i$$

$$\underline{N} = (N_1, \dots, N_M)$$

$$N_1' = N_1 + \dots + N_m$$

$$\underline{N}' = (N_1', N_{m+1}, \dots, N_M)$$

$$P(\underline{n}) = \text{equilibrium qlpd for the original network}$$

$$P'(\underline{n}') = \text{equilibrium qlpd for the reduced network}$$

$$P''(\underline{k}) = \text{equilibrium qlpd for the short-circuited subnetwork}$$

By [GN67] (and by [BCMP75] for the multi-class case), product form solutions exist for the qlpd's

P, P', P'' , for example

$$P(\underline{n}) \propto \prod_{i=1}^M x_i(n_i)$$

The aggregation result states that for the given choice of μ ,

$$\Pr(\underline{N}' = \underline{n}') = \sum_{n_1 + \dots + n_m = n_1'} \Pr(\underline{N} = \underline{n})$$

in a locally balanced queueing network. We now consider two types of aggregation which yield composite servers corresponding to first asynchronous and then synchronous operation.

2.2 Aggregation of Independent Parallel Servers (Multiple Server)

We first consider the subnetwork of m parallel servers numbered $1, 2, \dots, m$, with fixed rates $\mu_1, \mu_2, \dots, \mu_m$, each with its own queue and selected with probabilities $p_1, p_2, \dots, p_m$ by arriving customers. We assume that each server has a work-conserving queueing discipline such as first-come-first-served (FCFS) or processor sharing (PS), and (initially) negative exponential service time distributions, although this is not always necessary, e.g. for PS. We consider the case where p_i is proportional to μ_i ($1 \leq i \leq m$), i.e. the servers are 'equally utilised'. If the m parallel servers are embedded in a larger network, we wish to replace them with a single server with service rate $\mu(n)$ when its queue length is n , defined as in the previous section. Thus, $\mu(n)$ is the throughput of the closed network of Figure 1 when its population is n .

This network has a product-form solution for its equilibrium qlpd given by, in the notation of the previous section,

$$P^n(\underline{k}) \propto \sum_{i=1}^m \{p_i/\mu_i\}^{k_i} \quad (\underline{k} \in S_n)$$

(The visitation rate of server i is equal to the routing probability p_i for this network, $1 \leq i \leq m$).

Figure 1: The Short-Circuited Multiple Server Subnetwork

Thus, for equally utilised servers, all of the state space probabilities are equal and so

$$P^n(\underline{k}) = 1/G_n \text{ for all } \underline{k} \in S_n$$

where G_n is the normalising constant for the network when its population is n . Hence, G_n is equal to the order of S_n , i.e.

$$G_n = (n+m-1)! / \{(m-1)!\} n!$$

which we write as ${}^{n+m-1}C_n$. This result follows from a simple combinatorial argument: it is the number of ways of putting n balls (here, customers) into m bags (here, queues), which are represented by $(m-1)$ divisions on the bi-infinite line.

Now, equilibrium throughput is equal to

$$\sum_{i=1}^m \mu_i \sum_{\substack{\underline{k} \\ k_i > 0}} P^n(\underline{k})$$

But, for each i , $1 \leq i \leq m$, again since all states have the same probability

$$\sum_{k_i > 0} P^n(\underline{k}) = {}^{n+m-2}C_{n-1} / {}^{n+m-1}C_n = n/(m+n-1)$$

(since the numerator is the number of ways of arranging $n-1$ customers in m queues - one of them is known to be at server i)

Thus, by Norton's theorem we assign

$$\mu(n) = \{n \sum \mu_i\} / (m+n-1).$$

A special case of the multiple server represents a homogeneous bank of parallel, identical processors, each with constant service rate ν , which are selected by arriving tasks with equal probability. Here $\mu_i = \nu$ and $p_i = 1/m$ for $1 \leq i \leq m$, so we have $\mu(n) = \{mn/(m+n-1)\}\nu$. (A similar result was derived by Mecklenburg in [ME78].) This is probably the most common application and is used in our model of ALICE in Section 3.

It might be noted here that the most appropriate queueing discipline for the composite server is PS - certainly if the original servers were PS. However, even if these disciplines were not PS, the total capacity of the composite server is shared, especially for large m .

2.3 Virtual Banked Infinite Server Aggregation (VBIS Server)

We now consider the case of a bank of m identical components each multiplexing a number, v , of dedicated processes (called Virtual Agents) which are synchronised with some other part of the network - for example, sending messages and awaiting corresponding replies. In this situation, there is no queueing at a virtual agent which is either idle, awaiting a reply, or else ready to execute (or executing) with no possibility of an incoming message arriving for it. Thus, we have a bank of virtual, infinite-server components as shown in Figure 2 in the short circuited network.

We assume that all of the physical components which multiplex their virtual agents have fixed service rate μ and that any idle virtual agent is equally likely to be selected by an arrival. Generalisation to 'equally utilised' components and agents is simple, as in Section 2.2.

In the short circuited subnetwork, for $0 \leq a \leq n \leq mv$ let $u_{an} = \text{Pr}(\text{a virtual agents are busy} \mid \text{total number of customers} = n)$

and consider the configurations in which the n customers can be arranged. By symmetry, each of these occurs with equal probability in the steady

Figure 2: VBIS Server with m Banks of v Virtual Agents

state. We will derive a recurrence relation for the expected number of active virtual agents given the population n , $A(n)$. This corresponds to throughput $A(n)\mu$, which is therefore the service rate function of the composite server. Now,

$$u_{an} = u_{a-1, n-1} \text{Pr}(n^{\text{th}} \text{ customer assigned to an idle processor}) + u_{a, n-1} \text{Pr}(n^{\text{th}} \text{ customer assigned to a busy processor})$$

(where the probabilities are conditional on a total of $n-1$ customers and respectively $a-1$ and a busy processors)

$$= u_{a-1, n-1} \{v(m-a+1)/(mv-n+1)\} + u_{a, n-1} \{av/(mv-n+1)\}$$

Thus,

$$(mv-n+1)u_{an} = v(m-a+1)u_{a-1, n-1} + avu_{a, n-1}$$

and $u_{00} = u_{11} = 1$, and $u_{0n} = 0$ for $n > 0$. Now define the probability generating function

conditional on n as $H_n(z) = \sum_{a=0}^{\infty} u_{an} z^a$.

It is easily seen that for $n > 1$,

$$\begin{aligned} (mv-n+1)H_n(z) - u_{0n} = \\ mvz \sum u_{a-1,n-1} z^{a-1} - vz^2 \sum (a-1)u_{a-1,n-1} z^{a-2} \\ - (n-1) \sum u_{a,n-1} z^a + vz \sum au_{a,n-1} z^{a-1} \end{aligned}$$

where the summations range over $1 \leq a < \infty$. (Note that $u_{an}=0$ for $a > n$). Thus,

$$(mv-n+1)H_n(z) = (mvz-n+1)H_{n-1}(z) + vz(1-z)H_{n-1}'(z)$$

where $H_k'(z) = dH_k(z)/dz$ ($k \geq 0$).

Now, the expectation of u_{an} is $A(n) = H_n'(1)$, so that by differentiating with respect to z , setting $z = 1$ and noting that $H_n(1) = 1$ for all $n \geq 0$,

$$A(n) = \{((m-1)v-n+1)/(mv-n+1)\}A(n-1) + mv/(mv-n+1) \quad \text{and}$$

$$A(1) = 1$$

This recurrence formula computes $A(n)$, the mean number of active physical components for $1 \leq n \leq mv$. There is no combinatorial explosion in the computations, so no numerical problems arise and we assign $\mu(n) = A(n)\mu$ to the service rate function of the composite server. As with the multiple server, the most appropriate queueing discipline for the VBIS server is PS, and here, in the individual components the virtual agents might very well be scheduled on a Round Robin basis - well approximated by PS, especially if the time quantum is small - rather than first-in-first-out.

It is worth noting here that the VBIS server has a higher throughput than the corresponding multiple server formed from the same number of identical processors. This follows because when the number of customers, n , is greater than kv for some integer $k < m$, the total service rate is at least $(k+1)\mu$ (since k banks can accommodate at most kv customers). In the multiple server, the minimum service rate is always μ for any $n > 0$. (In both cases, maximum service rate is always $\mu \cdot \min(n, m)$). This is consistent with the synchronous nature of the VBIS server as opposed to the asynchronous multiple server. The parallel server configuration with the maximum throughput is the multi-server in which m servers operate in parallel with a single shared queue, and customers are served FCFS by any available component. The service rate of the composite server for this subnetwork is clearly $n\mu$ if $n \leq m$ and $m\mu$ if $n > m$. When $v=1$, the VBIS server is clearly equivalent to the multi-server (subject to $n \leq m$), and it can be seen that $A(n) = n$ from the recurrence formula. (Consider $B(n)=A(n)/(m-n)$ so that

$$B(n) = B(n-1) + m/\{(m-n)(m-n+1)\} = n/(m-n) \}.$$

2.4 Alternative Derivation for Independent Parallel Servers

The multiple server result can also be derived via a similar recurrence relation for $A(n)$, taking

care to consider configurations rather than arrivals. A new customer can be added to a queue of length q in $q+1$ different positions, so that (for example) the total number of positions in all queues for the n^{th} customer is $m+n-1$. This leads to the recurrence relation for

$$v_{an} = \Pr(a \text{ servers busy} \mid \text{total number of customers} = n)$$

$$\begin{aligned} &= v_{a-1,n-1} \{(m-a+1)/(m+n-1)\} + \\ &v_{a,n-1} \{(a+n-1)/(m+n-1)\} \end{aligned}$$

From this, the mean number of busy servers, conditional on n customers in total is easily shown to be $mn/(m+n-1)$ as in Section 2.2 by the method of Section 2.3.

3. Performance Analysis of the ALICE machine

3.1 Outline of the Principles of Operation

ALICE (Applicative Language Idealised Computing Engine) is a parallel computer architecture designed at Imperial College, [DR81]. It is currently being prototyped by International Computers Limited prior to being cast in VLSI. The machine has the unusual property that (sic) its performance can be increased by simply adding processors - by virtue of the fact that the processors are organised as homogeneous sets operating asynchronously. The machine architecture is shown in Figure 3. Performance modelling is important in the development of this architecture to guide the design of first the prototype (which can be used for model validation once built) and then the VLSI machine itself.

Figure 3: The ALICE Machine Physical Architecture

ALICE is a graph reduction machine, i.e. it conceptually operates by manipulating expression graphs. Each node on a graph is represented in the machine by a packet which contains an identifier field, an operator (or function) field a number of argument fields and some control fields. For example, the expression $f(x+1, g(x))$ can be represented in a packet form as shown in Figure 4.

f and g here are functions defined in the user program and $+$ is a built-in operator. If x above

(packet m) has a value of 4, say, then the machine can process packet j and rewrite it to the value 5. This may trigger further computation in packet i if i requires the value of packet j to be known before its computation can proceed. The same argument applies to packet k.

Figure 4: An Example Expression

Referring back to Figure 3, the rewriting of packets is done by the Packet Rewrite Units (PRUs). The packets themselves are stored in a distributed packet pool formed by a number of Packet Management Units (PMUs). The PRUs and PMUs communicate via an interconnection network which is assumed initially to be full crossbar switch. In the machine, the PRUs are paired together and share communication controllers for the network. Information is received from the network through a shared input External Request Unit (ERU in). Information is transmitted through the network via a shared output ERU (ERU out) which contains a DMA controller. Each PMU has a direct input link from the network and has associated with it one output ERU which also contains a DMA controller.

One 'master' copy of each function definition (i.e. the expression representing the function body) is held in the distributed packet store. Further copies of these definitions can be cached within store local to any PRU. When a rewrite is to be attempted by a PRU, this cache is examined by its associated Function Definition Checker (FDC) which first inspects the arriving packet to see which function definition it will require. If the cache does not contain the necessary definition, then the 'master copy' is fetched in from the appropriate PMU in fixed-size blocks, and the rewrite is suspended until the whole definition is available. In order to prevent the PRU from becoming idle waiting for external communications to complete, each physical PRU supports a number of virtual PRUs (VPRUs or virtual agents). Whilst communication to or from one VPRU is taking place, another VPRU can be scheduled and run on the same physical PRU. Each of the components of Figure 4 is in reality an Inmos Transputer, appropriately microcoded (in occam) to perform the desired operation.

There is also a load sharing mechanism, implemented by load sharing unit (LSU) processors on each PRU and PMU board, and communicating on a ring network. This will eventually enable load sharing algorithms to be developed to optimise the usage of resources, for example, to maximise the function definition cache 'hit-rates'. For the present, load sharing is random, but we will also

consider how to predict the performance which would result from optimum load sharing - providing an upper bound.

We wish to model this system so as to determine its performance (as measured by overall system throughput, PRU and PMU utilisation etc) as we vary the physical and operational parameters of the machine (such as numbers of PMUs, PRUs, VPRUs etc). The total number of PRU boards (two PRUs on each) and PMU boards is normally fixed in any given machine and one requirement of the model is to predict the proportion of PRU to PMU boards, and the number of VPRUs in each PRU, which give maximum throughput. Other predictions might include a graph of performance versus total number of processors (or boards); in the absence of any contention arising from communication and competition for the same destination processors, this would be linear.

3.2 An Abstract Model of the Architecture

The ALICE concrete architecture is complex in that it consists of a large number of computing elements, but at the same time many of these elements are identical, both physically (i.e. they contain the same hardware components) and functionally (i.e. they perform the same tasks). We can, in fact, identify five sets of identical processing components, interconnected as shown in Figure 5.

Notes

- (i) The nodes labelled ERU correspond to both the ERU-in and FDC components associated with the PRU pairs. Their service times reflect demultiplexing and function definition cache checking operations.
- (ii) The DMA nodes correspond to ERU-out processors and are assumed to be connected to the PMUs by a full crossbar interconnection network. A PRU may thus address any one of the PMU nodes indirectly via its corresponding DMA node. The nature of the full crossbar network means that the connection cannot be made if another PRU (via its DMA) is already connected to the addressed PMU.
- (iii) Similarly the DMA' nodes are assumed to be connected to the ERU nodes via a full crossbar network.
- (iv) The multi-way connections represented by the horizontal lines marked X and X' reflect the actual location of the full crossbar networks in the original architecture, although the network is not modelled explicitly as another server type in the model (see below).

3.3 The Reduced Model

In this model, we aggregate together all nodes of the same type according to the methods described in Section 2, yielding a five server queueing network. Although such aggregation loses some

Figure 5: Explicit Abstraction of the Architecture

detail with respect to the passage of individual tasks, the method gives, by construction, exact results for aggregate measures such as throughput, resource utilisation, mean queue lengths, and even queue length probability distributions might be re-captured, see Section 4.

We therefore now consider a closed queueing network which comprises five composite servers from the two classes given in Section 2. Each PMU node has its own buffer with a capacity sufficient to accept requests from all the VPRUs in the system, i.e. the buffer can never overflow and is effectively infinite. Thus, under random load sharing, PMUs are accessed with the same probabilities and form a homogeneous subset. They are therefore represented in the reduced network by a multiple server. Similarly, their output ERUs, DMA' nodes in Figure 5, are also represented by a multiple server. Note that under an optimal load sharing strategy, an idle PMU, if one were available, would always be assigned to an arriving task, so that the reduced server would become a multi-server.

Considering now the PMU nodes, any transaction leaving a PMU is addressed to a virtual PRU, one of the set within some physical PRU, which must be waiting for it according to the specific machine design. That is, the VPRU addressed is guaranteed to be idle. We can therefore model the PRUs in the reduced model as a VBIS node with parameters $2m$ (the number of PRUs i.e. twice the number of processor boards) and v (the number of VPRUs within each PRU). Note that here too, for optimal load sharing, the composite server would again reduce to a multi-server with parameter m for all v .

Similarly, each ERU and DMA server is associated with one pair of PRUs and consists of $2v$ virtual servers. They are therefore both modelled as VBIS servers with parameters m and $2v$.

We take the queueing disciplines of the composite servers to be PS as discussed in Section 2, thus giving all the generality we need, e.g. permitting general service time distributions in each server. However, recall that for the use of composite servers to be valid at all, the unaggregated model must have the local balance property. This

certainly holds for general service time distributions if all servers have PS queueing discipline, and for the PRUs which multiplex several processes, this is not an unreasonable assumption. The PMU (and DMA') nodes, however, operate strictly FCFS, so that for local balance to hold, the service time distributions must be negative exponential. Estimating service times in terms of time spent executing microcode in the Transputers and message lengths of the transaction types indicates that the shorter service times do occur with higher probability, so that the exponential assumption is at least not outrageously bad. At the same time, one might argue that when considered together, the overall effect of the physical PMU (and DMA') nodes is one of multiplexing resources, and hence that assuming PS discipline at each node individually would make little difference. Aggregation appears to smooth out many of the characteristics of the service time distributions, and perhaps we are appealing here to the illusive 'robustness property' of queueing networks - the need for some rigorous analysis here is apparent.

We have omitted some detail in this aggregation and so introduced some approximation in that, for example, given a particular PMU, although the associated DMA' server is known, the reduced model assumes that any DMA' unit could be selected by a departing task. This is why it is difficult to predict in the reduced model distributions of time delays which require explicit routing information, whilst the resource-based predictions remain exact.

3.3.1. Incorporating the network

In reality, the ALICE interconnection network is a circuit-switched multistage network (a "delta" network) consisting of a two-dimensional array of 4×4 crossbar switches interconnected according to a well defined permutation rule. Such a network provides the point-to-point connectivity of a single full-crossbar network at reduced cost although with potential contention not present in a single full-crossbar matrix [PA79]. The representation of a delta network in the model appears difficult but if we assume the network to be a single full-crossbar switch it is relatively straightforward to model the contention for the switch outputs by adjusting the service rates for each data transfer unit attached to the network. Let us consider a circuit-switched crossbar in which a channel, once established between a source and destination node, is kept open until all data transfer between the two has completed. Let t ($=\mu^{-1}$) be the mean data transfer time once a channel has been established. We can then model the crossbar by adjusting the mean service times (t) to reflect the contention for the switch outputs, using the methods used for MPLE servers in Section 2.2.

When a of the network inputs are active, the expected number of distinct network outputs selected is $Ma/(M+a-1)$ (from Section 2.2). The average switch throughput is therefore reduced to

$$\sum_{a=1}^m w_{an} Ma\mu/(M+a-1)$$

where μ is the mean data transfer rate (e.g. the DMA service rate in the case of the DMA $\rightarrow$ PMU connections) and w_{an} is the probability that a of the servers attached to the network inputs are active given that there are n requests in total at those servers. Thus for the DMA $\rightarrow$ PMU connections w_{an} is u_{an} and for the DMA' $\rightarrow$ ERU connections it is v_{an} .

Note that for the DMA $\rightarrow$ PMU connections, as $M \rightarrow \infty$, the summation reduces to $A(n)\mu$ as in Section 2.3., giving no degradation, as required. The same is true for $n=1$, whereupon $w_{an}=1$ if $a=1$ and 0 otherwise. Denoting $\{u_{an} | 1 \leq a \leq m\}$ by $\underline{u}_n$ (and $\underline{v}_n$ similarly), it is simple to write a program to compute successively $\underline{u}_n$ in terms of $\underline{u}_{n-1}$ for $n = 2, 3, \dots, 2mv$. Thus an improved service rate can be found for the DMA servers, taking into account contention at the destination nodes (i.e. representing the full crossbar). An interesting property of this crossbar model is that as its load increases, the probabilities w_{an} increase for larger a and the factors $M/(M+a-1)$ in the summation above decrease the service rate, possibly making the DMA server into the network bottleneck.

Although there will be some loss of accuracy in assuming the network to be a full crossbar, we find that in this architecture the network is not heavily utilised and so internal contention should have little effect on the performance predictions. (In fact, if the full crossbar is replaced by full connectivity between each PRUs and all PMUs, the performance increase is of the order of only 5 to 10%. We would expect a much smaller effect when replacing the full crossbar in the model with a delta network). Ultimately, however, a saturated network will determine the overall machine performance.

3.4 Model Specification

The analytical model is defined by Figure 6.

The routing probability p is determined by the 'hit' probability, i.e. the probability that the function definition required to rewrite a packet arriving at the PRU is already in the local cache of the PRU. In the event of a 'miss', further transactions with the PMUs are necessary to load the cache before the VPRU is rescheduled.

The (adjustable) parameters of the model are m : the number of PRU boards (i.e. $2m$ physical PRUs), M : the number of PMU boards, v : the number of virtual agents in each PRU and N : the network population where $N \leq 2mv$, or else not all customers could reside in the PRUs at once. In our test cases, we always take $N = 2mv$ to predict maximum performance. The cache hit probability is another parameter, but we have so far taken this to be unity, as will be the case in the first tests on the prototype which will have large caches.

The basic unit of work in the model is the network transaction, which is a message of mean length 18 bytes. Corresponding to every request, PRU $\rightarrow$ PMU, a single response is returned (e.g. a packet), PMU $\rightarrow$ PRU, so that no new tasks are

Figure 6: Model Structure

created during a cycle, nor are any absorbed (destroyed); a fixed number of customers merely cycle round the queueing network model. The microcode used in the processors has been analysed to provide the estimates for mean service times in the (unaggregated) PRU, PMU and ERU nodes. For the mean service times of the DMA and DMA' nodes, a similar mean processing time contribution is added to an estimate for mean transmission time, based on a serial link data rate of 20 Mbit per second and mean message length of 18 bytes.

Node Type	Mean Service Time (MST Microseconds)	Service Rate (MST ⁻¹)
PRU	251	3984
DMA	71	14085
PMU	157	6369
DMA'	60	16667
ERU	72	13889

Table 1: Single Component Service Rates

The models defined by various choices of parameters m , v and M are solved for their qlpds and hence performance measures by the standard convolution algorithm for computing normalising constants, see for example [SC81]. This is simply programmed, with attention paid (we thank David Chan) to normalisation of the floating point numbers involved, and several tests have been run as discussed next.

3.5 Numerical Results and Conclusions

One prototype ALICE machine is intended to have a total of 40 boards in its rack, each of which can be either a processor (dual PRUs) or memory (PMU) module. The model was used primarily to predict, for various assignments of parameters, machine throughput measured in network transactions per second, PRU utilisation, defined by the ratio of

throughput to maximum PRU service rate (i.e. service rate of one PRU multiplied by twice the number of PRU boards) and the mean number of customers in the PMU buffers. The throughput and utilisation curves are shown in Figures 7 and 8 respectively. Figure 9 shows the effect of changing PRU service rate on throughput. Other resource-based performance measures are equally easy to compute, e.g. total queue length distributions at the PRUs or PMUs.

It can be seen from Figure 7 that maximum throughput is obtained with around 17 PRU boards. Moreover, throughput continues to increase with v , although with diminishing returns. Given sufficient PRU memory, this is quite reasonable since overhead is associated with each VPRU (i.e. increases only linearly with v) and is already included in the individual PRU node mean service time; the same for any value of v . However, throughput is limited by the service rate of the physical PRUs, the upper bound being approached asymptotically as utilisation approaches 100%. The more realistic constraint is the maximum number of processes able to be accommodated in the PRU memory, in fact giving $v \leq 8$. We conclude that v should therefore be 8. We also note that as v increases, the optimal number of PRU boards decreases slowly - again as expected since increasing v when the PRUs are not fully utilised will increase their performance, making the PMUs become the network bottleneck. This is also evidenced when the PRU service rate is increased as can be seen from the sensitivity analysis graph of Figure 8: a 10% swing in the PRU service rate changes the optimum number of PRU boards by approximately one either way. Thus, in recommending the most suitable value for m in the 40 board machine, we have proposed a value of 16 rather than 17; this is because the most easily optimised component is the PRU (or rather, its microcode) which can be fairly easily tuned by code restructuring.

It was also found that maximum throughput increased approximately in proportion to the number of boards, as expected with no network contention. By comparing the throughput of the system (and hence the network) with the network capacity, we find that the utilisation of the network is around 7.5%, so the full crossbar approximation to the delta network looks good for the time being.

As network overhead increases, the PRUs will become idle more often awaiting responses from the PMUs, and so optimum performance will be obtained for a larger value of v than in the absence of contention.

4. Model Enhancements and Conclusion

4.1 Representation of Transaction Types

A multi-class model could be used to analyse the behaviour of different transaction types with correspondingly different message lengths and service times. For example, a request for a packet to rewrite (PRU $\rightarrow$ PMU), a packet (PMU $\rightarrow$ PRU), a function definition (PMU $\rightarrow$ FDC), a reference count update (PRU $\rightarrow$ PMU).

Another immediate and simple extension is to include a better representation of the function definition cache checking. Only certain transaction types pass through the cache checking part of the FDC node (all are processed by the ERU-in processor, see Figure 1), which should therefore be modelled as a separate server. In a multi-class model, only customers of the right type would be routed this way, or in a single-class model, customers would be routed to it with probability equal to the visitation rate of customers of that type.

4.2 Modelling the delta Network

We are currently investigating an analytical model of the delta network. In the mean time it may be possible to use simulation to predict the network throughput for a given population and then use the implied service rate function at the DMA and DMA' servers.

However, if a single delta network is used (with PRUs and PMUs having input and output links to the same network) there will be interference between PRU $\rightarrow$ PMU requests and PMU $\rightarrow$ PRU requests (because of the internal contention in the delta network) leading to 'blocking' in the model. A solution to this is to use only one combined server to represent the DMA and DMA' nodes together. This would require a multi-class model to explicitly distinguish requests generated by the PRU and PMU nodes. Perhaps a more satisfactory set-up (also from the point of view of the architecture) is to use two independent networks: one for PRU $\rightarrow$ PMU communication and one for PMU $\rightarrow$ PRU communication. (This assumes, of course, that PRU $\rightarrow$ PRU and PMU $\rightarrow$ PMU communication is not required.)

4.3 Load Sharing

Load sharing algorithms are concerned with the distribution of work amongst the PRUs and packets amongst the PMUs. In the present model, the load sharing is random, giving a lower bound which any new algorithm should better. An upper bound on the performance obtainable by any algorithm can be found by replacing all composite servers by multi-servers, as noted in Section 2, and this has indicated that throughput would increase by some 25%. The smallness of this improvement is not really surprising in view of the good level of sharing achieved by a relatively large number of parallel processors, operating independently and asynchronously, but perhaps even more so because of the small grain of the workload - the rewriting of one function's defining expression. The smaller the grain, the larger the number of (small) work units spread (randomly) across the components of the machine, and the less likely these are to be idle. (At the same time the smaller the grain the higher the communication overheads, suggesting perhaps a large-grain architecture is more efficient, but this is another issue.)

4.4 Cache Size vs. Multiprocessing Level

The effect of the function cache hit rate is represented by the parameter p which we have thus far taken to be 1. It would clearly be simple to make predictions for any value of p which could be

estimated in terms of cache store size (of which it is an increasing function). However, increasing cache store size must ultimately be at the expense of the number of virtual agents which can be supported in the Transputers' memory. Thus, for a given total memory size, we have a decreasing function $V(p)$ for the optimum (i.e. maximum) multiprocessing level, v . Since throughput increases with both v and p , its graph against v (or p) should show a more marked peak.

4.5 Queue Lengths at Individual Machine Components

Finally, we look at the possibility of recapturing some of the queue length information lost in the aggregation process, i.e. the distribution of queue length at each component making up a composite server. Let the servers in an aggregated subnetwork be numbered $1, 2, \dots, m$, then for $1 \leq i \leq m$, in the notation of Section 2,

$$\Pr(N_i=k) = \sum_{n \geq 0} \Pr(N_i=k \mid N_i=n) \Pr(N_i'=n)$$

The term $\Pr(N_i'=n)$ is what is computed by the reduced, composite server model. For multiple servers, all states n with the same fixed $n_{m+1}, \dots, n_m$ and such that $n_1 + \dots + n_m = n$ have the same equilibrium probability. Thus, with obvious abbreviation of notation,

$$P(k|n) = \frac{m+n-k-2}{n-k} C_{n-k}^{m+n-1} / C_n^{m+n-1}$$

This again follows by ball-in-bag arguments, the numerator being the number of ways of placing (the other) $n-k$ balls in (the remaining) $m-1$ bags. Now, the normalising constant, G_{mn} , for the short-circuited subnetwork of m servers with population n is equal to C_n^{m+n-1} , and so $P(k|n)$ may be written as $G_{m-1, n-k} / G_{mn}$. Each normalising constant can be computed by standard methods, e.g. [BU73], giving no numerical problems. A similar analysis can be applied to VBIS servers, noting that the maximum number of customers in any bank is v , its number of virtual agents.

These results might eventually also lead to methods for computing (approximately) the distribution of time delays experienced by particular tasks, for example by considering the dependencies between successive pairs of queues in their paths, as in [HA86].

5. Concluding Remarks

To summarise, what we have done is to present a methodology for modelling the type of homogeneous parallel architectures currently being designed for new generation machines, and to illustrate its application in performance prediction with a case study. The development of the work is at an interesting stage in that there will be a real

machine against which to validate in the very near future. Moreover, some improvements can be made relatively easily to make the model more representative of the physical architecture, and so improve its accuracy, based on results already derived (see the previous section). There are also some interesting, more fundamental theoretical problems to solve, most notably the development of an analytical model of the delta network.

Acknowledgement

We would like to thank Mike Reeve for providing us with the inspiration to undertake the ALICE case study.

References

- [BCMP75] Baskett F, Chandy K M, Muntz R R, Palacios F G; 'Open, Closed, and Mixed Networks of Queues with Different Classes of Customers', JACM 22, 2
- [BU73] Buzen J P; 'Computational Algorithms for Closed Queueing Networks with Exponential Servers', CACM 16, 9
- [CHW75] Chandy K M, Herzog U, Woo L; 'Parametric Analysis of Queueing Networks', IBM J Research Development, January 1975
- [DR81] Darlington J, Reeve M; 'ALICE: A Multiprocessor Reduction Machine for the Parallel Evaluation of Applicative Languages', In Proceedings of ACM Conference on Functional Languages and Computer Architecture, Portsmouth, New Hampshire (1981)
- [GN67] Gordon W J, Newell G F; 'Closed Queueing Systems with Exponential Servers', Oper. Res. 15, 254-265
- [HA86] Harrison P G; 'An Enhanced Approximation by Pair-Wise Analysis of Servers for Time Delay Distributions in Queueing Networks', IEEE Transactions on Computers, January 1986
- [ME78] Mecklenburg A W; 'The evaluation of computer performance by means of state-dependent queueing network models', PhD thesis, University of London
- [PA79] Patel J H; 'Processor-Memory Interconnections for Multi Processors', Proc 6th International Symposium on Computer Architecture, 1979
- [SC81] Sauer C H, Chandy K M; Computer Systems Performance Modelling, Prentice-Hall, 1981

Figure 7: Graphs of Throughput vs Number of PRU boards and Number of VPRUs for 40 board Machine

Figure 8: PRU Utilisations for varying numbers of PRUs and VPRUs

Figure 9: Effect of varying PRU service rate on throughput