Analytic Model of a Cache Only Memory Architecture

Carlos Carreras[1], Carlos A. López[1] and Manuel Hermenegildo[2]

[1] Departamento de Ingeniería Electrónica, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain
[2] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Madrid, Spain

Abstract. An approximate analytic model of a shared memory multiprocessor with a Cache Only Memory Architecture (COMA), the bus-based Data Difussion Machine (DDM), is presented and validated. It describes the timing and interference in the system as a function of the hardware, the protocols, the topology and the workload. Model results have been compared to results from an independent simulator. The comparison shows good model accuracy specially for non-saturated systems, where the errors in response times and device utilizations are independent of the number of processors and remain below 10% in 90% of the simulations. Therefore, the model can be used as an average performance prediction tool that avoids expensive simulations in the design of systems with many processors.

1 Introduction

Performance prediction is a key issue in the first stages of the design of a shared memory multiprocessor. Approximate analytic models have been propossed to obtain average performance estimates fast [9, 14, 12]. They allow exploring the parameter design space for different workloads. Analytic models provide guidance to adjust the system parameters so that detailed simulations are only required in a verification phase. The analytic model becomes a compromise between a detailed description that captures all the relevant features of the system and the simplifying approximations that allow fast computation of performance estimates. This paper presents an analytic model of a specific shared memory multiprocessor, the bus-based Data Difussion Machine (DDM).

The DDM [13, 4, 5] belongs to a new class of architectures called Cache Only Memory Architectures (COMA). These are distributed shared memory systems with directory-based coherence where memory modules behave like large second level caches. COMA constitute a new approach to design large-scale multiprocessors. The bus-based DDM is a COMA with a coherence protocol supported by a scalable hierarchy of directories and buses. Other examples of COMA are the link-based DDM from Bristol University [10] and the comercial ring-based KSR1 from Kendall Square Research [1].

The analytic model developed here is based on similar principles to those found in [9]. The response time of a closed system, the DDM, is obtained from the response times of the open models of its subsystems: the processors, the memories and the interconnection network. At the subsystem level, probabilistic models are used. Contention is modeled using simplified queueing systems previously validated. The error introduced by these approximations is reduced with parameter adjustment from system simulations.

The model equations are solved through iteration. Convergency occurs quickly, usually after a few iterations. The solution corresponds to the steady state situation where response times and execution rates are balanced. The outputs of the model describe in detail the overall system's performance as well as internal traffic rates and delays.

Validation is performed against simulation results available from an independent simulator developed at the Swedish Institute of Computer Science (SICS). These results correspond to real applications from the SPLASH suite of benchmarks [11] executed over different topologies. The comparison shows that the error introduced by the approximate model is acceptable specially if the system is not saturated, which is the desirable condition. In this case, errors are inside the expected bounds established in [7] (30% for response times and 10% for utilizations) in more than 90% of the simulations.

The model is extended with approximate scalable models of the main application-dependent parameters as a function of the topology and the number of processors. Uniform memory references are considered. Even though this is the worst case for COMA, modeled parameters show qualitative changes similar to those observed in parameters obtained from simulations.

The paper is structured as follows. The next section introduces the main features of the DDM. Section 3 presents the modeling methodology. Section 4 develops the processor, memory and network models. The algorithm to solve the model equations is presented in Sect. 5. Section 6 describes the simulation environment and the benchmarks used to validate the model. In Sect. 7, the model is validated against existing simulator results. In Sect. 8, approximate models of the application-dependent parameters are derived and evaluated. Finally, conclusions are summarized in Sect. 9.

2 The DDM Multiprocessor

The DDM has been introduced as a COMA. The main contribution of COMA is the definition of protocols that allow data to replicate and migrate between memory modules dynamically at run time, as oppossed to systems where data is statically distributed between the memories. Memories behave like large second level caches, changing their contents according to the needs of their local processors. This results in overall traffic reduction if the allocation of tasks to processors takes advantage from the locality of memory references. Therefore, COMA provide a new strategy to design large-scale multiprocessors. The cost of the COMA strategy is the search procedure required to locate data in the

system. A hierarchical COMA performs a hierarchical search, which is fast for requests to neighbour nodes in the hierarchy, but penalizes requests to distant nodes with longer search times. The bus-based DDM is a hierarchical COMA implemented through a hierarchy of buses and directories.

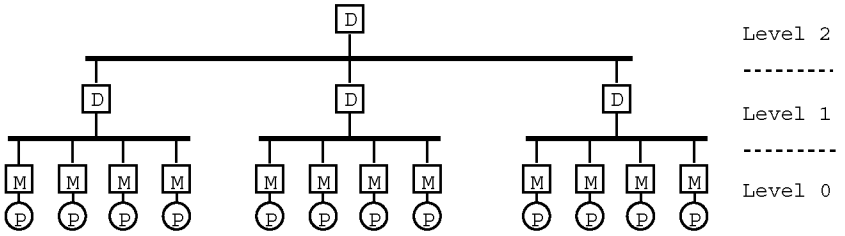

Fig. 1. A basic bus-based DDM (P: Processor/Cache, M: Attraction Memory, D: Directory)

A block diagram of a basic bus-based DDM is represented in Fig. 1. It has three main subsystems: the processor nodes which include a first level of caches, the attraction memories which are memories that support the DDM protocol, and the hierarchy of buses and directories. Directories don't store data. They only store state information used to locate data in the memories below them. Attraction memories store data and state information associated to them. In this situation, read or write requests that can't be serviced by the local attraction memory are forwarded up the hierarchy. A request reaching a DDM bus is processed in two pipelined phases: bus transaction and state lookup in the devices connected to the bus. The state lookup allows locating the requested datum. A transaction goes up the hierarchy until the search is successful, and then goes down to the attraction memories below where data is actually stored. The DDM protocol allows that the reply to a search follows the path back to the requesting memory module, while keeping consistency between the contents of memories and directories. It is a *write-invalidate* protocol.

A prototype of a clustered bus-based DDM is near completion at the Swedish Institute of Computer Science (SICS). In a clustered DDM, memories provide local service to a cluster of processors through a fast bus. A local coherence protocol is required for local consistency in the cluster. The implementation details in the prototype are based on the M-bus and the 88000 series from Motorola. In this case, the local protocol is of type *write-once*. It has been adapted to work together with the DDM protocol so that all data in caches and memories are kept consistent.

3 Modeling Methodology

The analytic model of the DDM is obtained from the open models of its subsystems. This approximation allows modularity in the model and has been used

to evaluate different types of processor and memory nodes. However, only the types used in the SICS prototype, from which simulator results are available, are presented here.

Three types of subsystems are considered in the model: the processor nodes, the memory nodes and the network of directories. A memory node model supports local and remote operation. It is assumed that all subsystems of the same type have the same probabilistic behaviour. Each subsystem model expresses its output transaction rates and response times as a function of its input transaction rates and internal parameters and queues. The average input rate to the system is F_{avg} instructions per processor cycle and its average response time is T_{avg} processor cycles per instruction. All subsystem input rates are obtained from F_{avg}, while T_{avg} depends on all subsystem response times and queue waiting times. Queues are solved locally for each subsystem using simple queueing models. Therefore, this approach results in three sets of equations: transaction rates, queue waiting times and response times. The steady-state condition says that $F_{avg} = 1/T_{avg}$, so these sets are related in a cyclic fashion and the general equation of the model can be expressed as $T_{avg} = \mathcal{F}\{\mathcal{G}[\mathcal{H}(T_{avg})]\}$.

Subsystem models have been developed under a common methodology which can be summarized in the following steps:

1. Classify the input transactions according to the requested operations and obtain their rates.
2. Define the internal parameters describing hardware and application variables.
3. Define all possible internal operations in the subsystem according to differences in operation response times or generated output transactions.
4. Obtain the probability of each operation.
5. Determine the transaction rates and utilization times involved in internal queues and obtain the expressions of the queue waiting times.
6. Obtain the subsystem's average response times and the output transaction rates per class.

Input transaction class rates are obtained from the output rates of other connected subsystems (Step 1). F_{avg} is an input class to the processor nodes, being T_{avg} their average response time. The expressions of operation response times or generated output transactions (Step 3) are determined from the input rates, the internal parameters, the subsystem's architecture, the supported protocols and other subsystems' response times. The probability of each operation (Step 4) is obtained assuming that the internal parameters are independent. In general, rates and times in the queues (Step 5) are not equal to operation rates and times and must be determined. The queues are solved using simplified queueing models previously validated against results from a queue simulator. The subsystem's average response times (Step 6) are computed as weighted averages of the internal operation response times involved. Finally, the subsystem's output class transaction rates are obtained as weighted additions of the transactions generated by internal operations. In both cases, the weights are obtained from the probabilities of the operations.

4 Subsystem Models

4.1 Processor Node Model

The model of a processor node represents a *single-threaded* processor with separate instruction and data caches. The processor remains idle on cache misses. The data cache supports a *write-back* local coherence protocol. Contention in local memory accesses is included in the memory model so the processor node model has no internal queues.

Input transaction classes are F_{avg} from the processor's own thread, and *data transactions*, *write-acknowledgements*, *invalidation requests* and *update requests* from the local memory. The internal parameters include the cache line size, the probability that an instruction is a memory reference (p_{mem}), the probability that a memory reference is a write (p_{write}), the instruction and data cache miss ratios (m_I, m_D), the probability that a data cache miss requires cache replacement, and the minimum average response time for the application obtained considering no cache misses. Operations are defined according to the previous input classes, probabilities and miss ratios. Output transaction classes are *instruction reads*, *data reads without replacement*, *data reads with repl.*, *data writes without repl.*, *data writes with repl.*, and local *memory updates*. The resulting model expresses the output transaction rates in terms of F_{avg} and provides an equation of T_{avg} in terms of the memory response times for the different output transaction classes.

4.2 Cluster Memory Node Model

This model includes an attraction memory and the bus that connects it to the processors of the cluster. The attraction memory is assumed to be large enough to satisfy all private requests from local processors, so that only shared data travel to and from the remote memories through the network of buses. It is also assumed that remote memory references are uniformly distributed among the attraction memories. The cluster memory node is modeled according to the SICS implementation around the M-bus. A block diagram is shown in Fig. 2. Accesses to the data memory are part of the read and write M-bus transactions, while the state memory sits in the M-bus like another requesting device while interfacing to the network of directories. The *interrupt-retry* mechanism of the *write-once* M-bus protocol allows the devices in the bus to interrupt the current transaction which is retried later. This is used by the state memory to maintain remote consistency and by the local caches for internal cluster coherency.

Input transaction classes from below are the processors' output classes. Input classes from the network are *read requests*, *invalidate requests* and *replace requests* from remote memories, and replies to previous read and invalidate requests from the attraction memory itself (*data transactions* from remote memories and *invalidate-acknowledgements* from upper level directories). Replacement transactions relocate a replaced memory line in a different attraction memory.

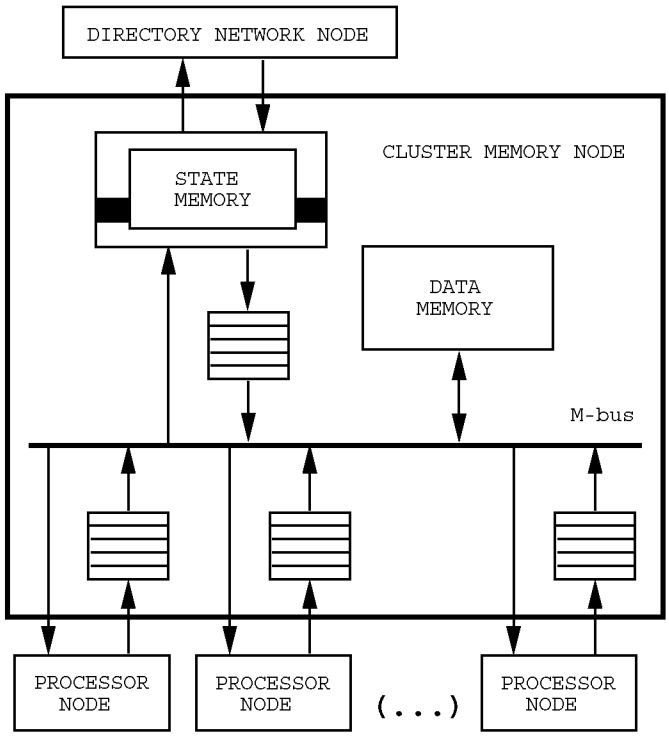

Fig. 2. Organization of a cluster memory node

Internal parameters include the number of processors per cluster (N_0), the memory line size, independent memory miss ratios for reads and writes ($m_{r,0}, m_{e,0}$), the probability that a memory miss requires replacement in the memory, the probabilities that local or remote transactions in the M-bus cause a memory update, the probability that a request requires arbitration to access the M-bus, and detailed hardware parameters that specify the internal timing of M-bus accesses. Again, operations are defined according to these probabilities and miss ratios. Output transaction classes to the processors have been listed in the previous section, while output classes to the network are *read requests*, *invalidate requests* and *replacement transactions*, and replies to remote read requests previously received (*data transactions* to remote memories).

While the dual-port state memory can be accessed without interference from above and below, there is contention to access the M-bus. In this situation, the M-bus is modeled as a queueing system with multiple classes of independent Poisson arrivals and deterministic service times. The *round-robin* M-bus scheduling policy selects a new device when the currently serviced device has no more pending requests. Arbitration only occurs when a new device is selected and the arbitration time is assigned to its first pending request. This policy has no exact

queueing model so approximate models have been used. In particular, a model that approximates such policy to an ordered sequence of $(N_0 + 1)$ fixed priority schemes, one per device in the M-bus, has been developed. This model, called RR here, expresses the average waiting time in a M-bus queue, W_q, as

$$W_q = \sum_{k=1}^{N_0+1} (\frac{\lambda_k}{\lambda}) \sum_{m=1}^{N_0+1} (\frac{\rho_m}{\rho}) W_{km} \quad (1)$$

where λ_k and λ are the total arrival rates to queue k and to the M-bus respectively, ρ_m and ρ are the M-bus utilization by requests from queue m and the total M-bus utilization, and W_{km} is the waiting time in queue k when queue m is serviced, that is, when queue m has maximum priority. Assuming that queue $(m-1)$ follows queue m and queue (N_0+1) follows queue 1 in the ordering of queues, the value of W_{km} is obtained from the adapted fixed priority equations

$$W_{km} = \frac{W_0}{(1 - \sum_{i=k}^{m} \rho_i)(1 - \sum_{i=k+1}^{m} \rho_i)} \qquad (1 \leq k \leq m \leq N_0 + 1)$$

$$W_{km} = \frac{W_0}{(1 - \rho + \sum_{i=m+1}^{k-1} \rho_i)(1 - \rho + \sum_{i=m+1}^{k} \rho_i)} \qquad (1 \leq m < k \leq N_0 + 1) \quad (2)$$

where W_0 is the mean residual life of a service time in the M-bus, which can be computed from arrival rates and service times [6]. Besides the RR model, an approximate model based on FCFS scheduling has also been used. Simulations for different classes, queues and M-bus utilizations show that the RR and FCFS results are always inside the 99% confidence interval around the mean simulated values, that is, the interval containing 99% of simulated waiting times.

4.3 Directory Network Model

This model is based on the directory node which includes a directory that stores states and a DDM bus. The internal structure of a directory node is similar to that in Fig. 2 but without a data memory. The level-1 directories interface with the memory nodes. Otherwise, directories connect to other directories above and below as in Fig. 1. Directory accesses and arbitration are pipelined with the transactions in the DDM bus.

Input and output classes to and from the network were described in the previous section. Internal parameters include the number of levels in the hierarchy (L), the branching factor at each level (N_i), the DDM bus cycle time at each level, the number of additional bus cycles required by transactions that carry data, independent directory miss ratios for read and invalidate requests $(m_{r,i}, m_{e,i})$, and the average number of memories with a copy of a datum at the time the datum is written (K). With respect to replace requests from a memory, it is assumed that they are always serviced by another memory connected to the same level-1 DDM bus, so they are not transferred up the network. It is considered that the branching factor N_i is constant at each level. Therefore, only balanced topologies are modeled. Operations in the network are defined from

the input classes, the number of levels and the read and invalidate miss ratios at each level.

Again, interference in each bus is determined considering Poisson arrivals and deterministic service times. The *round-robin* DDM bus scheduling policy selects a new device after each transaction, so it is different from the M-bus policy. However, simulation results show that the RR and FCFS approximations can also be used since they are always inside the 99% confidence interval. The traffic in each queue is computed from the protocol definition, the directory miss ratios, the branching factors, and, in the case of children invalidate transactions descendant from an invalidate request going up the hierarchy, from the parameter the average number of copies between writes, K.

5 The Algorithm

The algorithm to solve the model for steady-state conditions assumes that an interval $[a, b]$ is known to contain the model's solution for T_{avg}. In this case, iterative binary division of the interval is a fast method to find such solution if convergence conditions are met. Binary division implies introducing the mid-point value of the interval, x, in the model equations. If $\mathcal{F}\{\mathcal{G}[\mathcal{H}(x)]\} > x$ the subinterval $[a, x]$ includes the solution and is selected for the next iteration. Otherwise, the subinterval $[x, b]$ is used. The iterative process is repeated until the interval is small enough to assure that the error is less than ϵ. The algorithm is fast since only a few tens of iterations are required to get an approximate solution with $\epsilon = 0.01$ processor cycles. The DDM model meets the convergency conditions [9] since increasing response times cause decreasing execution rates, increasing transaction rates cause increasing queue waiting times and $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ are positive continuous functions, at least in a domain around the solution.

Computing $\mathcal{F}\{\mathcal{G}[\mathcal{H}(x)]\}$ in the iterative process is easy as long as the three sets of model equations maintain an ordered dependency. In the DDM, this ordering is broken by the *interrupt-retry* mechanism of the M-bus protocol: requests in the M-bus which are forwarded to remote memories are repeatedly issued and interrupted in the M-bus until the reply arrives from the network. Therefore, there is an internal loop of dependencies between transaction rates in the M-bus and network response times. This new internal cyclic set of equations is solved again through iteration. Therefore, the actual algorithm for the DDM model consists of two nested iteration loops.

Convergency requires that the value a in $[a, b]$ is a feasible value of T_{avg}. Otherwise, bus utilizations may go above 1 and the method does not converge. In this situation, a is determined as the minimum T_{avg} (maximum F_{avg}) that keeps all bus utilizations below 1 in the model, and b is the (maximum) T_{avg} derived from a, $b = \mathcal{F}\{\mathcal{G}[\mathcal{H}(a)]\}$. The value of a is obtained again by binary division of any interval $[T_1, T_2]$ where T_1 forces at least one bus utilization to be greater than 1 and T_2 causes all bus utilizations to be less than 1.

6 Simulation Environment and Benchmarks

The results from a simulator of the DDM developed at SICS [3] have been used to verify the analytic model. It is an *execution-driven* simulator [8] that provides system and device performance statistics as well as counts of internal events. Systems with one or two DDM levels and clusters with up to four processors running at 20 MHz have been simulated for each benchmark. Instruction and data caches of 16 Kbytes and cluster memories of 32 Mbytes are used. The memory line size is set equal to the cache line size which is four words. The DDM bus cycle times are set to four processor cycles in all levels of the hierarchy, and all transactions in DDM buses are processed in one bus cycle. The simulator also includes secondary memory from where the benchmarks are initially loaded. It is accessed through a M-bus like an attraction memory but from the upper level DDM bus controller (root). Accesses to this unit mostly occur during startup with rates that are negligible when to the whole simulation time is considered.

Three benchmarks from the SPLASH suite representing real scientific parallel applications, WATER, MP3D and CHOLESKY, have been simulated. They are described in detail in [11]. Besides, simulation results for a matrix multiplication program, MATRIX, are also available. WATER computes forces and potentials in a dynamic system of liquid water molecules. Input systems with 192 and 384 molecules and working sets of 320 and 640 Kbytes have been used. MP3D simulates very low density fluid flow around an object using a discrete statistical model of the medium. Most available results correspond to a $14 \times 24 \times 7$ cell space with 75,000 molecules for a working space of 4 Mbytes. Other simulations consider 40,000 and 80,000 molecules. CHOLESKY is a sparse matrix factorization parallel program. Results for matrices *bcsttk14* and *bcsttk15* (see [11]), which occupy 420 and 800 Kbytes unfactored and 1.4 and 7.7 Mbytes factored respectively, are available. In the following section, they are treated like individual benchmarks named CHOLESKY14 and CHOLESKY15. Finally, MATRIX is a program that multiplies two matrices using a blocking algorithm. The matrix size is set to (500×500) elements for a working space of 3 Mbytes.

7 Model Validation

Parameters of the analytic model which depend on the hardware or the topology are obtained from the simulator inputs. The resulting latencies in cycles for basic memory read and write operations (local and remote without contention and arbitration) are summarized in Table 1.

Application-dependent parameters of the model are extracted from the internal counts of the simulator. Then, the performance results from the model and the simulator are compared. Simulated performance is available in terms of average instruction execution time per processor and bus utilizations. The iterative nature of the algorithm, which departs from an arbitrary interval, assures that the model results are actually obtained from the model and not directly from the input parameters.

Table 1. Latencies for memory references

Reference (cache miss)	Local (M-bus)	Remote (level 1)	Remote (level i)
read	7	38	$(38 + 16i)$
write	10	32	$(32 + 8i)$

Analytic model and simulator have been developed independently following different approaches. In fact, a few input parameters of the model must be obtained through approximate computations. Besides, a secondary memory module had to be added to the model. However, results are quite acceptable. Table 2 contains, for each application and configuration for which simulator results are available, the errors obtained from the model. In some cases, different application input parameters have been used to simulate the same system.

The notation for each configuration has the form $[N_0 \times N_1 \times N_2 \times \cdots]$, where N_0 is the number of processors per cluster and N_i is the branching factor at level i in the hierarchy. The minimum system with only one processor is $[1 \times 1]$. T_{avg} describes the system's response time, $U_{cluster}$ represents the M-bus utilization, and $U_{bus,1}$ and $U_{bus,2}$ refer to the utilizations of level-1 and level-2 DDM buses. Errors are differences between simulated and modeled values expressed as percentages of the simulated values. The numbers in parenthesis relate to absolute bus utilizations below 0.01. They are not considered in the comparison since they represent negligible absolute errors. This is also the case for bus utilizations in the secondary memory module, which are not included in the table.

If total numbers from Table 2 are analyzed, the errors in system response times (T_{avg}) for most configurations (97%) are well within the 30% margin defined in [7] as acceptable. Errors in device utilizations are acceptable if they are within a 10% margin around the simulated values. This is verified by most $U_{cluster}$ and $Ubus, 2$ results (89% and 93% respectively), but only by some $U_{bus,1}$ values (35%). The low percentage obtained for $U_{bus,1}$ is due to the saturation of the M-bus ($U_{cluster} > 0.8$) for MP3D and CHOLESKY15. Small errors in traffic rates in the M-bus cause significant errors in M-bus waiting times and, therefore, in traffic rates reaching the level-1 DDM bus. Once it is concluded that the model degrades in saturation, it can be observed that errors in $U_{bus,1}$ for the non-saturating applications are below or around 10% in most cases (93%). Finally, it can be observed that errors don't show any clear dependency on the number of processors or levels in the system.

8 Application-dependent Parameters

So far, the application-dependent parameters of the model have been obtained from simulation. In order to model large systems, models of these parameters as a function of the topology are required to scale the values obtained from small system simulations.

Table 2. Difference between analytic and simulated values

Applic.	Config.	(%) T_{avg}	(%) $U_{cluster}$	(%) $U_{bus,1}$	(%) $U_{bus,2}$
WATER	$[1 \times 1]$	-0.23	0.92	(23.44)	
WATER	$[4 \times 1]$	3.26	0.28	(25.53)	
WATER	$[4 \times 8]$	2.05	3.13	-1.37	
WATER	$[4 \times 8 \times 2]$	-0.32	2.06	-8.37	5.39
WATER	$[2 \times 8 \times 2]$	2.76	0.18	-10.36	0.70
WATER	$[1 \times 8 \times 2]$	4.55	0.27	-11.63	-2.03
MP3D	$[1 \times 1]$	-0.02	-0.66	(0.00)	
MP3D	$[4 \times 1]$	5.23	-14.20	(-25.00)	
MP3D	$[4 \times 2]$	21.89	-5.23	-18.94	
MP3D	$[4 \times 4]$	23.17	-1.15	-20.01	
MP3D	$[4 \times 4 \times 2]$	27.75	6.64	-29.72	-22.03
MP3D	$[2 \times 8 \times 2]$	1.81	2.83	-10.08	-1.83
MP3D	$[2 \times 8 \times 2]$	2.87	1.85	-10.69	-3.24
MP3D	$[2 \times 8 \times 2]$	9.26	-3.40	-15.45	-8.02
MP3D	$[1 \times 8 \times 2]$	4.11	-7.17	-17.38	-9.82
CHOL-14	$[1 \times 1]$	0.44	-0.82	(40.48)	
CHOL-14	$[2 \times 4]$	6.02	13.33	-5.39	
CHOL-14	$[2 \times 8]$	5.39	9.99	-4.66	
CHOL-14	$[2 \times 8]$	2.70	8.44	-4.34	
CHOL-14	$[1 \times 32]$	-1.24	-7.06	-2.47	
CHOL-14	$[1 \times 32]$	2.11	-5.38	-2.60	
CHOL-14	$[2 \times 8 \times 2]$	-1.88	4.74	-8.98	5.64
CHOL-14	$[2 \times 8 \times 2]$	-1.01	7.47	-8.60	2.26
CHOL-14	$[1 \times 8 \times 2]$	3.40	2.07	-13.69	0.79
CHOL-15	$[1 \times 1]$	0.05	-0.85	(20.00)	
CHOL-15	$[4 \times 1]$	19.36	-11.20	(23.50)	
CHOL-15	$[4 \times 4]$	33.57	-1.73	-25.13	
CHOL-15	$[4 \times 8]$	23.31	3.24	-18.85	
CHOL-15	$[2 \times 8 \times 2]$	-0.94	0.07	-24.84	-7.08
CHOL-15	$[2 \times 8 \times 2]$	8.67	7.20	-16.39	-6.50
CHOL-15	$[2 \times 8 \times 2]$	9.40	5.24	-17.41	-4.84
MATRIX	$[1 \times 1]$	-0.21	-10.40	(-100.00)	
MATRIX	$[4 \times 1]$	4.13	-2.58	(12.50)	
MATRIX	$[4 \times 4]$	2.92	-4.65	-2.64	
MATRIX	$[4 \times 8]$	1.51	-2.53	-1.90	

The three main application parameters that directly depend on the topology are the memory miss ratios ($m_{r,0}, m_{e,0}$), the directory miss ratios ($m_{r,i}, m_{e,i}$), and the average number of copies of a shared datum between invalidates (K). Their models are based on parameters which are assumed to remain constant for all topologies, including two new parameters: the probability that a request to the local memory references a shared datum (p_{shared}) and the probability that a write request to the local memory references a shared datum ($p_{write,shared}$).

The model of K is developed first. Startup effects are not considered and read-only shared data is assumed to be loaded in the local memories. In this situation, when a shared datum is written, $(K-1)$ copies in memories are invalidated and one is updated. Therefore, $(K-1)$ can be expressed as the rate ratio between read requests and invalidate requests issued to the network. If $m_{r,shared}$ and $m_{e,shared}$ are the local memory read and write miss ratios defined when only shared data references are considered:

$$K = 1 + \frac{(m_I + p_{mem} m_D) p_{shared} m_{r,shared}}{p_{mem} p_{write} m_D p_{write,shared} m_{e,shared}} \tag{3}$$

It is known that at least $(K-1)$ processors read the shared datum between writes and, if N is the total number of cluster memories, at least $[N_0(N-K)]$ processors don't read it. This proportion can be extended to approximate the behaviour of the remaining processors for a total of P processors. In this situation, $m_{r,shared}$ can be approximated to the probability that a read request reaching the local memory is the first one in the cluster requesting the shared datum since the last time it was written, multiplied by the probability that this cluster did not issue the last write. With respect to $m_{e,shared}$, it can be approximated to the ratio between the number of processors outside one cluster and the total number of processors in the system except for the one requesting the write:

$$\begin{aligned} m_{r,shared} &= \frac{N_0(N-K) + (K-1)}{P} \left(1 - \frac{1}{N}\right) \\ m_{e,shared} &= \frac{P - N_0}{P - 1} \end{aligned} \tag{4}$$

Substituing (4) in (3) and solving for K:

$$\begin{aligned} K &= \frac{1 + \mathcal{A}(P-1)}{1 + \mathcal{A}(N_0 - 1)} \\ \mathcal{A} &= \frac{(m_I + p_{mem} m_D) p_{shared} \left(\frac{N-1}{PN}\right)}{p_{mem} p_{write} m_D p_{write,shared} \left(\frac{P-N_0}{P-1}\right)} \end{aligned} \tag{5}$$

The models of memory miss ratios depend on K through $m_{r,shared}$ and $m_{write,shared}$:

$$\begin{aligned} m_{r,0} &= p_{shared} m_{r,shared} \\ m_{e,0} &= p_{write,shared} m_{e,shared} \end{aligned} \tag{6}$$

Directory miss ratios are modeled using a combinatorial approximation based on the uniform distribution of memory references. It is considered that the average number of copies available for a read request reaching the network is $K/2$, while the average number of copies to erase by an invalidate request is $(K-1)$. Considering a generic directory at level i, $m_{r,i}$ expresses the probability that none of the $K/2$ copies are in the $N_i N_{i-1} \cdots N_1$ cluster memories below it. Similarly, $m_{e,i}$ is the probability that not all $(K-1)$ copies are in the memories below:

$$m_{r,i} = \frac{\binom{N - N_i N_{i-1} \cdots N_1}{int(K/2)}}{\binom{N-1}{int(K/2)}}$$

$$m_{e,i} = 1 - \frac{\binom{N_i N_{i-1} \cdots N_1 - 1}{int(K-1)}}{\binom{N-1}{int(K-1)}} \tag{7}$$

Since K is not an integer, the previous combinatorial expressions approximate $K/2$ and $(K-1)$ to their closest integer values. It should be mentioned that this can lead to non-convergency situations when solving the DDM model if the error bounds that stop the iterative process are very tight.

It is interesting to compare the results of these models with the values obtained from the simulator, even though the simulations don't meet the model assumptions. The results are in Table 3, where superscript (s) identifies simulated values and superscript (a) refers to results from the analytic models. They show that, in most cases, the analytic values for a given benchmark describe the simulated behaviour, at least qualitatively. The values of $m_{r,i}$ and $m_{e,i}$ are not as significant since the models are intended for larger configurations.

9 Conclusions

Cache Only Memory Architectures have appeared as a new alternative to build large-scale shared memory multiprocessors. They have a memory coherence protocol that allows migration and replication of data. In this paper, an approximate analytic model of a COMA, the bus-based Data Difussion Machine, has been presented. The DDM protocol is supported by a hierarchical network of directories. The purpose of this model is to predict the average behaviour of systems with many processors.

The DDM model is based on the open models of its subsystems. A methodology to develop the subsystem models has been presented and applied to obtain the processor, memory and directory network models. Computation time to solve the complete DDM model is low (less than 1 second in all modeled configurations). The model results have been compared to results from an independent simulator for some real benchmarks from the SPLASH suite. The comparison

Table 3. Comparison between analytic and simulated parameters

Applic.	Config.	$K^{(a)}$	$K^{(s)}$	$m_{r,0}^{(a)}$	$m_{r,0}^{(s)}$	$m_{e,0}^{(a)}$	$m_{e,0}^{(s)}$	$m_{r,1}^{(a)}$	$m_{r,1}^{(s)}$	$m_{e,1}^{(a)}$	$m_{e,1}^{(s)}$
WATER	$[4 \times 8]$	2.17	2.13	0.25	0.27	0.72	0.69				
WATER	$[4 \times 8 \times 2]$	2.49	2.22	0.30	0.31	0.76	0.73	0.53	0.36	0.53	0.49
WATER	$[2 \times 8 \times 2]$	2.36	2.19	0.32	0.35	0.77	0.79	0.53	0.35	0.53	0.46
WATER	$[1 \times 8 \times 2]$	2.30	2.14	0.34	0.43	0.80	0.95	0.53	0.34	0.53	0.43
MP3D	$[4 \times 2]$	1.43	2.01	0.07	0.16	0.34	0.35				
MP3D	$[4 \times 4]$	1.80	2.00	0.18	0.24	0.48	0.52				
MP3D	$[4 \times 4 \times 2]$	2.07	2.04	0.27	0.28	0.54	0.58	0.57	0.55	0.57	0.58
MP3D	$[2 \times 8 \times 2]$	2.24	2.05	0.34	0.31	0.58	0.62	0.53	0.50	0.53	0.54
MP3D	$[2 \times 8 \times 2]$	2.25	2.06	0.34	0.32	0.58	0.61	0.53	0.48	0.53	0.54
MP3D	$[2 \times 8 \times 2]$	2.26	2.06	0.34	0.32	0.58	0.63	0.53	0.49	0.53	0.53
MP3D	$[1 \times 8 \times 2]$	2.24	2.03	0.35	0.31	0.60	0.63	0.53	0.51	0.53	0.54
CHOL-14	$[2 \times 4]$	2.00	2.04	0.28	0.31	0.77	0.81				
CHOL-14	$[2 \times 8]$	2.17	2.07	0.42	0.44	0.84	0.93				
CHOL-14	$[2 \times 8]$	2.23	1.93	0.42	0.47	0.84	0.99				
CHOL-14	$[1 \times 32]$	2.42	1.91	0.56	0.56	0.90	0.99				
CHOL-14	$[1 \times 32]$	2.42	2.02	0.56	0.55	0.90	0.99				
CHOL-14	$[2 \times 8 \times 2]$	2.31	1.94	0.50	0.53	0.87	0.99	0.53	0.56	0.53	0.62
CHOL-14	$[2 \times 8 \times 2]$	2.31	2.05	0.50	0.52	0.87	0.99	0.53	0.48	0.53	0.53
CHOL-14	$[1 \times 8 \times 2]$	2.35	1.89	0.53	0.51	0.90	0.99	0.53	0.56	0.53	0.60
CHOL-15	$[4 \times 4]$	1.85	2.05	0.09	0.10	0.72	0.65				
CHOL-15	$[4 \times 8]$	2.10	2.07	0.13	0.12	0.81	0.79				
CHOL-15	$[2 \times 8 \times 2]$	2.91	2.24	0.16	0.13	0.87	0.99	0.53	0.47	0.53	0.49
CHOL-15	$[2 \times 8 \times 2]$	2.28	2.03	0.17	0.14	0.87	0.85	0.53	0.52	0.53	0.53
CHOL-15	$[2 \times 8 \times 2]$	2.28	2.05	0.17	0.12	0.87	0.78	0.53	0.45	0.53	0.46
MATRIX	$[4 \times 4]$	2.85	2.00	0.01	0.01	0.32	0.12				
MATRIX	$[4 \times 8]$	2.94	2.04	0.02	0.03	0.36	0.54				

shows that the model is specially accurate if the systems are not saturated. In this case, the error in response times and device utilizations is below 10% in around 90% of the simulated systems.

The application-dependent parameters of the model must be described in terms of the system's topology if machines with many processors are to be evaluated. At this respect, approximate models of the parameter K (copies of a shared datum between writes) and the read and write miss ratios of memories and directories have been presented. They describe the same qualitative behaviour of the parameters observed in the simulations.

The model of the DDM have been used to evaluate the behaviour of COMA in [2]. Future plans include the adaptation of the application-dependent models to describe specific real applications. In particular, locality in the distribution of memory references is the next aspect to be included since it is directly related to the possible advantages of COMA.

References

1. H. Burkhardt, S. Frank, B. Knobe, and J. Rothnie. Overview of the KSR1 Computer System. Technical Report KSR-TR-9202001, Kendall Square Research, 1992.
2. C. Carreras. *Modelo Analítico de la Máquina de Difusión de Datos y Efecto de la Inclusión de Procesadores Multicontexto.* PhD thesis, E.T.S.I.Telecomunicación, Universidad Politécnica de Madrid, Sep 1993.
3. E. Hagersten, P. Anderson, A. Landin, and S. Haridi. A Performance Study of the DDM - A Cache Only Memory Architecture. Technical Report R91:17, Swedish Institute of Computer Science, Nov 1991.
4. E. Hagersten, S. Haridi, and D. H. D. Warren. The Cache-Coherence Protocol of the Data Diffusion Machine. In M. Dubois and S. S. Thakkar, editors, *Cache and Interconnect Architectures in Multiprocessors.* Kluwer Academic Publisher, 1990.
5. E. Hagersten, A. Landin, and S. Haridi. DDM - A Cache-Only Memory Architecture. *IEEE Computer*, 25(9):44–54, 1991.
6. L. Kleinrock. *Queueing Systems (Volumes 1 and 2).* John Wiley and Sons, 1975.
7. E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance - Computer System Analysis Using Queueing Network Models.* Prentice-Hall, 1984.
8. M. Lofgren. *A Simulator in C++ for a Parallel Architecture.* PhD thesis, Swedish Institute of Computer Science, Nov 1990.
9. A. Norton and G. F. Pfister. A Methodology for Predicting Multiprocessor Performance. In *Proceedings 15th Annual International Symposium on Parallel Processing*, pages 772–781, 1985.
10. S. Raina and D. H. D. Warren. Traffic Patterns in a Scalable Multiprocessor through Transputer Emulation. In *Proceedings International Hawaii Conference on System Science*, 1991.
11. J. P. Singh, W.-D. Weber, and A. Gupta. SPLASH: Stanford Parallel Applications for Shared-Memory. Technical Report CSL-TR-92-526, Computer Systems Laboratory, Stanford University, Jun 1992.
12. M. K. Vernon, R. Jog, and G. S. Sohi. Performance Analysis of Hierarchical Cache-Consistent Multiprocesors. In *Proceedings International Seminar on Performance of Distributed and Parallel Systems*, pages 111–126. North-Holland, Dec 1988.
13. D. H. D. Warren and S. Haridi. Data Diffusion Machine - A Scalable Shared Virtual Memory Multiprocessor. In *International Conference on Fifth Generation Computer Systems.* ICOT, 1988.
14. A. W. Wilson. Hierarchical Cache/Bus Architecture for Shared Memory Multiprocessors. In *Proceedings 14th Annual Symposium on Computer Architecture*, pages 244–252, 1987.