MEMORY MANAGEMENT AND ITS INFLUENCE ON THE COMMUNICATION OF HOMOGENEOUS AND HETEROGENEOUS PARALLEL SYSTEMS

Ventura P. * **Ruano M. Graça** * **Cardoso e Cunha J.** **

* *GPSBio/ADEEC, Unidade de Ciências Exactas e Humanas, Universidade do Algarve, Campus de Gambelas, 8000 Faro, Portugal*
** *Departamento de Informática, FCT, Universidade Nova de Lisboa, Quinta da Torre, 2825-114 Caparica, Portugal*

Abstract: This paper reports a study held on the inter-processor communication performance of homogeneous and heterogeneous parallel systems when different data structure allocation schemes are implemented, regarding internal and/or external processor memory. To evaluate the performance parameters some case-study algorithms were implemented on homogeneous and heterogeneous architectures. Due to algorithm-machine dependency, hardware and software features have to be considered. Where the access to external memory is not efficient, some internal memory buffering methods are analysed. A comparison of the results obtained is presented, enabling establishment of memory management references, regarding the parallel architectures employed.

Keywords: architectures, digital signal processors, parallel processing, memory banks, communication channels, performance analysis

1. INTRODUCTION

Recent technological evolution allows computational resolution of increasingly complex problems, involving larger volumes of data. The viability of real-time implementation of this type of problems frequently requires the use of parallel and/or distributed systems. One of the factors affecting the performance of the global system is the hardware architecture selected and its ability to perform efficiently the problem. On a parallel processing environment, distribution of the computational load across the processors is a task involving features such as the network topology, the relative computation and communication capabilities of the processors, as well as algorithm task partitioning and consequent scheduling of these tasks to processors (Tokhi and Hossain, 1995*b*). It is important to consider all these features on the implementation of an algorithm on either a homogeneous or a heterogeneous parallel platform.

Previous works on this field considered performance evaluation on distinct parallel architectures of complex and highly demanding signal-processing and control algorithms (Tokhi and Hossain, 1995*b*; Tokhi *et al.*, 1995; Tokhi and Hossain, 1995*a*). The present study evaluates the influence of different type data structure allocation schemes on internal and/or external memory. The memory of a C program is divided into four sections, code, static storage, stack and heap. Mapping of these sections to physical memory (internal or external) can have a significant impact on the parallel system performance. Knowledge about the characteristics of a parallel system influences algorithm design. So, the major goal of the study hereby reported is concerned with the evaluation of the tradeoffs between specific

parallel systems and certain types of algorithms, when memory management and communications are considered.

2. HARDWARE

The parallel architectures in study concerned homogeneous and heterogeneous architectures. Three different type of processors were employed: T805 Transputer (T8), Texas Instruments TMS320C40 (C40) and ADSP-21060 Sharc (Sharc). In general, the behaviour of a parallel system reflects the nature of its processors, being essentially a description of the processors relevant features and of the parallel architecture itself.

2.1 *Processing elements*

The T8 (INMOS Limited, 1990) is a 32-bit processor with 25 MHz clock speed, 4 Kbytes on-chip RAM and capable of 4.3 million floating-point operations per second (MFLOPS). The T8 includes a 64-bit floating-point unit and four serial communication links. The links operate at speeds of 20 Mbits/sec, achieving data rates of up to 1.7 Mbytes/sec unidirectionally or 2.3 Mbytes/sec bi-directionally.

The C40 (Texas Instruments, 1991) is a 32-bit DSP processor with 50 MHz clock speed, 2 blocks of on-chip RAM of 4 Kbytes each and is capable of 50 MFLOPS. For inter-processor communication includes six parallel communication links with 28 Mbytes/sec asynchronous transfer rate. It has separate internal program, data and DMA coprocessor buses for support of massive concurrent I/O of program and data throughput.

The Sharc (Analog Devices, 1995) is a 32-bit DSP processor with 40 MHz clock speed, a central processing unit (CPU) peak performance of 120 MFLOPS and includes 2 blocks on-chip SRAM of 256 Kbytes each. This processor possesses six links for point-to-point communication with other processors, operating concurrently and bi-directionally. The links have a maximum communication rate of 40 Mbytes/sec. This processor has three internal buses connected to its dual-ported memory, the program memory (PM) bus, data memory (DM) bus and I/O bus. The I/O bus allows concurrent data transfers between both memory blocks and the links ports. With this dual-ported structure, accesses to internal memory by the processor and the I/O processor are independent and transparent to one another.

2.2 *Homogeneous and heterogeneous architectures*

Three homogeneous parallel architectures were considered, employing T8´s, C40´s and Sharc´s. The configuration of the homogeneous systems consisted of three processors, the root processor acting as a router of input/output operations, with the other two processors being actually responsible for the algorithm calculus.

Each processor has local memory and the links were used for communication with other processors in the network. The chosen network topology was based on the algorithmic structure.

The heterogeneous system included T8 and C40, where a T8 was employed as the root processor, and the two other processing nodes were T8 and C40 processors (Figure 1). Again, the network topology was chosen considering the algorithmic structure.

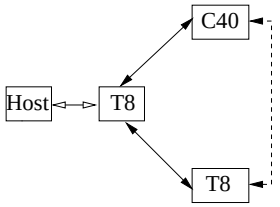

Fig. 1. Block diagram of the heterogeneous parallel system

3. SOFTWARE

Software support is essential for the development of efficient programs. Characteristics such as disk and memory accesses, input and output activities, compilation time, CPU time, and operating system overhead must be considered (Kumar, 1994). A good memory management, an efficient partitioning and mapping of tasks and the selection of a suitable compiler can affect the performance of a system. The selection of a compiler is a critical factor, since the compiler is responsible for the transformation of a high level language into a hardware dependent implementation, which differs from compiler to compiler, and presents different performance on different processors. To minimise these effects, the same compiler family the 3L Parallel C was used. To be noted that different compilers of 3L Parallel C will be used according to the targeted processor.

The ideal performance of a parallel architecture demands a perfect match between the capability of the architecture and the program behaviour. The hardware features can be optimised with new technologies and efficient resource management. However, program behaviour is difficult to predict, since it suffers the influence of factors such as data

structures and communication between processors (Tokhi and Hossain, 1995*b*).

Communication among processors is one of the main features influencing the performance of a parallel system. The amount of data, the frequency at which they are transmitted, the speed of their transmission and the route they take, are relevant factors on the communication among processors of a parallel system (Tokhi and Hossain, 1995*b*). The last two factors depend on the hardware, being the transmission speed an important discussion point since the memory zone where the data structure is mapped can influence the communication.

4. ALGORITHMS

The algorithms considered on this study were the cross-correlation (Proakis and Manolakis, 1988), the least mean square (LMS) adaptative filter (Haykin, 1998) and matrix back-propagation (MBP) (Anguita *et al.*, 1994), algorithms typically used in signal processing and control applications.

4.1 *Cross-correlation Algorithm*

The cross-correlation measures the similarity between two waveforms. Consider two signals with finite energy, $x(n)$ and $y(n)$. The cross-correlation of the two signals is defined as:

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l); l = 0, \pm 1, ... \quad (1)$$

or, equivalently, as

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n+l)y(n); l = 0, \pm 1, ... \quad (2)$$

The index l is the time shift, and the subscripts xy indicate the sequences being correlated.

4.2 *LMS Algorithm*

The LMS is one of the most successful adaptative algorithms. The LMS may be formulated as follows:

$$W_{k+1} = W_k - 2e_k \mu X_k \quad (3)$$

and,

$$e_k = y_k - W_k^T X_k \quad (4)$$

where W_k and X_k are the weight and the input signal vectors at time k respectively, e_x is the error signal, y_k is the current contaminated signal sample and μ is the constant controlling the stability and rate of convergence.

4.3 *MBP Algorithm*

MBP is a variation of the Back-Propagation. The algorithm can be defined in three phases as follows: Feed-Forward

$$S(l) = b(l).1^T \quad (5)$$

$$S(l) = b(l-1)W(l) + S(l) \quad (6)$$

$$S(l) = f(S(l) \quad (7)$$

Error Back-Propagation

$$\Delta(l) = \frac{2}{PN_L}(T - S(l)) \quad (8)$$

$$\Delta(l) = \Delta(l)g[S(l)] \quad (9)$$

$$\Delta(l) = \Delta(l+1)W^T(l+1) \quad (10)$$

$$\Delta(l) = \Delta(l)g[S(l)] \quad (11)$$

Weight updating

$$W(l) = \eta(S^T(l-1)\Delta(l) + W(l) \quad (12)$$

$$b(l) = 1^T\Delta(l) + \Delta b(l) \quad (13)$$

$S(l)$ is the neuron output matrix, T represents the network output matrix, $W(l)$ corresponds to the neuron weights matrix, $\Delta(l)$ is the propagated errors matrix and $b(l)$ is the neuron bias matrix. Besides these matrixes, there are references to functions f, the neuron activation function, and to g the first-order derivative of f. Wherever 1^T appears is a $mx1^T$ or 1^Txm operation.

5. IMPLEMENTATION

Considering that parallel systems are built upon processors with on-chip memory, message passing and data parallelism were the programming models used. The parallelisation of each algorithm was done according to Foster methodology. The algorithm parallelisation considered three phases:

1.Identification of parallelism in the algorithm, choosing domain or functional partitioning,

2.Partition of the algorithm into tasks,

3.Allocation of tasks to processors.

To investigate the algorithms performance, the referred aspects have been explored in order to minimise inter-processor communications and to enable equally distributed computation load

(Foster, 1995). Experiments were carried out varying the following parameters:

1.mapping stack, heap and static storage to internal and external memory,

2.communication of data structures in internal and external memory,

3.buffering techniques for data structures in external memory and its influence in the communication between processors.

The initial algorithmic study on data structure memory allocation considered the default compiler configuration. Taking into account the optimisation possibilities of compilers for each type of processors, other hypothesis of memory attribution were then analysed. Finally, for those cases that presented different communication times for structures mapped in internal and external memory, hybrid solutions were tested, allocating in internal memory the structures with data to transmit.

6. RESULTS

To evaluate the impact upon inter-processor data transmission of internal or external data location, some experiences were performed.

Figure 2 shows the communication times of 1000 data elements of type float on homogeneous and heterogeneous architectures with two processors. The homogeneous Sharc architecture presents internal memory data transmission 10 times faster than external transmission. As the T8 the homogeneous architecture did not have enough on-chip memory to allocate all data, only external memory data transmission is presented. No significant difference on the transmission time of data in internal and external memory can be observed in the C40 homogeneous architecture.

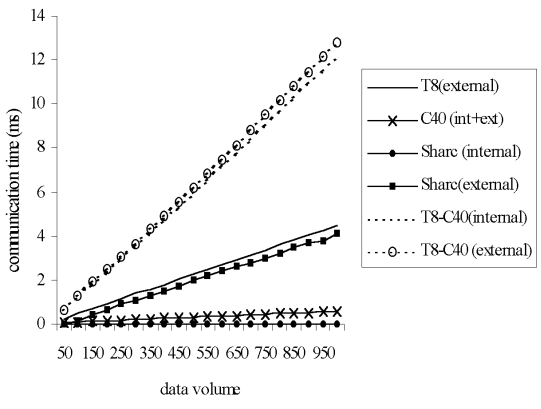

Fig. 2. Data communication times (ms) on homogeneous and heterogeneous systems

On the heterogeneous architecture composed by one T8 and a C40 (T8-C40), the worst inter-processor transmission time are observed due to the hardware links among different type of processors and the software conversion from IEEE to ITI format (and vice-versa). An overhead time is noted when, on the C40, the data is mapped to external memory.

Table 1 show a behaviour comparison of the three homogeneous architectures in the implementation of cross-correlation algorithm with its data structures allocated in internal and external memory. The implementation of cross-correlation used two waveforms of 2500 samples each. Neither the homogeneous architecture of T8's nor the C40's one have enough internal memory to hold all elements.

Table 1. Execution time (ms) of the Cross-Correlation algorithm implemented in the homogeneous architectures

Parallel architectures	Physical Memory	Static Storage	Stack Storage	Heap Storage
T8+T8	external	12504,2	12504,2	12504,2
C40+C40	local	883	884	883
	global	2260	2244	2245
Sharc+Sharc	int-b0	707	707	706
	int-b1	394	394	394
	external	416	416	416

The Sharc processor has 2 blocks of 256 Kbytes each. In this case, all data could be contained in internal memory. However, to determine how memory mapping blocks influences algorithm performance, other configurations will be considered.

The T8 homogeneous architecture presents the same execution time independently of data memory allocation. None of the stack, heap or static storage sections fit in internal memory because of their large dimensions.

On the C40 architecture, the execution time when data structures are allocated in external memory depends on the data being in local or global scope. As can be seen in table 1, the implementation of cross-correlation in local memory is about 150% faster than in global memory. When data is declared in stack or heap spaces the execution times are smaller than considering static storage. This is due to the extra cycle needed to set the data pointer (DP) to the correct page before accessing the global variable (Texas Instruments, 1995).

Cross-correlation has been implemented in the Sharc homogeneous architecture, always considering code segment in the first block of internal memory (int-B0). In cases where this block includes others segments, the algorithm presents its worst performance. Maximum performance is achieved when one block in internal memory con-

tains instructions, while the other (int-B1) contains only data. The independent PM and DM buses allow simultaneous access to instructions and data from both memory blocks. In this situation the optimisation was almost 80% compared to code and data in the same block. Good results were also achieved by storing data in external memory and the code in internal memory. In the later case, the computation time equals those observed that led to previous best solutions. Some extra is observed, caused by communication of data in external memory.

To investigate the behaviour of the architectures implementing the LMS algorithm, a finite impulse response (FIR) filter structure of 200 weights was used, the parameter μ was fixed to 0.04 and the execution of the algorithm occurred over 1000 iterations. Under these conditions, it was possible to study the internal memory access for T8 and C40 homogeneous architectures.

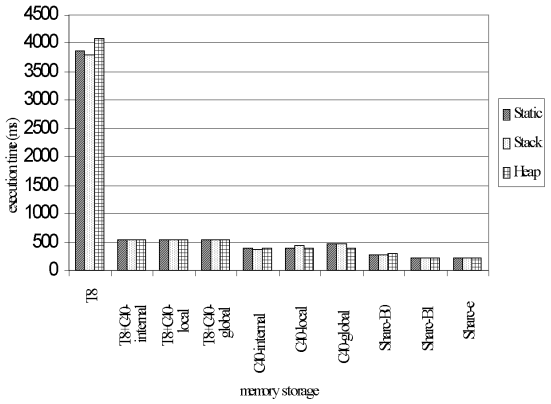

Fig. 3. Execution times (ms) obtained when implementing the LMS on the architectures

In comparison with the implementation of cross-correlation algorithm, the T8 homogeneous architecture shows some different executions times for data declared in the heap, stack and static storage (see fig. 3). The most efficient implementation is achieved with data mapped in the stack, benefiting from the stack being allocated in internal storage. The T8 processor has only 4 Kbytes of internal memory; this space is too small to include all the sections.

Considering the C40 architecture, the criteria observed with the cross-correlation were also applied to the LMS. In this case, it is possible to analyse the accessing time to internal memory. Here, the program and data memories lie in two separate spaces in internal memory. The best execution time in internal memory was obtained with data in stack storage. Working with external memory allocating data in the heap led to the best results.

In the performance analysis of LMS in Sharc homogeneous architecture, Figure 3 shows that internal and external memory accessing times are similar, except when static and heap segments exist in the same block together with the code. No difference was observed between execution time for data in second block or external memory. This is due to the small data volume involved in the experience.

The heterogeneous systems presented equal execution times independently of the memory allocation considering that the C40 is faster than the T8, the computation load distribution to T8 processor had to be less than to C40 in order to reduce the execution time. All the system performance depends on the slower T8 processor.

The study held on systems with inefficient interprocessor data transmission of external data location was done to evaluate the buffering techniques improvements in the algorithm performance. In cases where on-chip data structures have to be accessed by processes on others processors, the best option was to allocate elements in internal memory. Sharc homogeneous architecture was in those conditions, as observed in figure 2. To analyse the impact of such technique, the MPB algorithm performed the calculus of 16 bits XOR considering internal, external and hybrid storage.

Figure 4 compares execution times of the MBP algorithm implemented in Sharc homogeneous architecture, considering that the main structures were on the heap. When data was allocated to external memory, the algorithm execution time was almost the double comparing with data in second block of internal memory (int-B1). The implementation of data as global variables in internal memory was analysed to evaluate the success of hybrid storage for those structures accessed by other processes.

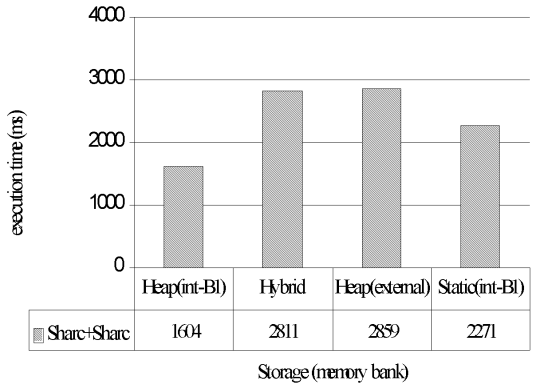

Fig. 4. Execution times (ms) obtained on the Sharc architecture when implementing the MBP algorithm

It can be observed in figure 4 that no significant improvements were obtained by the hybrid storage approach. The execution time of data allocated to static storage justifies the bad results obtained with the hybrid storage. Considering in internal memory only the structures that were accessed by other processors, this hybrid solution showed no impact in the algorithm performance.

7. CONCLUSION

This paper has explored the impact of different memory allocation schemes on the performance of parallel architectures in implementing cross-correlation, LMS and MBP algorithms, emphasising its influence on inter-processor communication performance and computational performance. It has been proved that using processors such as T8 and C40, where the internal memory capacity is low (less than 10 Kbytes (INMOS Limited, 1990; Texas Instruments, 1995)) the transmission of structures in external memory is optimised. However, using the ADSP-21060 Sharc, presenting an internal memory capacity of 512 Kbytes (Analog Devices, 1995), the transmission from external memory is not efficient. In the later case, when data structures mapped to external memory are used, the communication among processors presents a performance reduction of about 10 times regarding the case of communication of data structures in internal memory.

For processors with internal memory raised capacity, the assignment of data structure to external memory represents a loss of performance in inter-processors communication, but no significant reduction is observed in the computation. In the Sharc homogeneous architecture, the experiments performed demonstrate that code and data must be always in separated internal memory blocks.

For C40 homogeneous architectures, the best performance was observed with data mapped in the stack or heap. The access time to internal and local off-chip memory was similar, but the access to global off-chip area presented a great difference.

In the case of T8 homogeneous architectures, the processor had low internal memory capacity, allowing allocation to internal memory only to problems with small dimension. In this case, data structures stored in the stack were recommended.

The heterogeneous architecture had shown a dependency from the slowest processor, the T8. If the algorithm shows some inter-processor communication, a tendency to maintain the execution independent from the data structures storage will be observed.

This investigation had revealed that in the Sharc architecture hybrid storage implementation based on data transmission show no great improvements in performance as was expected, since the computation performance in external memory was similar to the computation in internal memory.

8. REFERENCES

Analog Devices (1995). *ADSP-2106x SHARC User's Manual*. Analog Devices.

Anguita, Davide, A. Da Canal, W. Da Canal, A. Falcone and A.M. Scapolla (1994). The distributed implementation of the back-propagation.

Foster, Ian (1995). *Designing and Building Parallel Programs - Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Publishing Company.

Haykin, Simon (1998). *Neural Networks - a comprehensive foundation*. Prentice Hall.

INMOS Limited (1990). *IMS B004 user guide and reference manual*. INMOS Limited.

Kumar, Vipin (1994). *Introduction to Parallel Computing - Design and Analysis of Algorithms*. The Benjamin/Cummings Publishing Company, Inc.. USA.

Proakis, J. G. and D. G. Manolakis (1988). *Introduction to Digital Signal Processing*. Macmillan Publishing Company. USA.

Texas Instruments (1991). *TMS320C40 User's Guide*. Texas Instruments. USA.

Texas Instruments (1995). *TMS320 Floating-Point DSP Optimising C Compiler - User's Guide*. Texas Instruments. USA.

Tokhi, M.O. and M.A. Hossain (1995*a*). Cisc, risc and dps processors in real-time signal processing and control. *Microprocessors and Microsystems*.

Tokhi, M.O. and M.A. Hossain (1995*b*). Homogeneous and heterogeneous parallel architectures in real time signal processing and control. *Control Eng. Practice* **3**(12), 1675–1686.

Tokhi, M.O., M.A. Hossain, M.J. Baxter and P.J. Fleming (1995). Homogeneous and heterogeneous parallel architectures for real-time active vibration control. *IEEE Proc. Control Theory Appl.*