

Rapid development of operationally-efficient and portable large-scale image processing solutions on multiprocessor workstations

S. Chapman¹, R. M. Lea²

Dept of Electronic and Computer Engineering,
Brunel University,
Uxbridge, UB8 3PH. UK.
+44 (0) 1895 203221

1 stephen.chapman@brunel.ac.uk

2 mike.lea@brunel.ac.uk

R. White³, K. Howsen⁴

Kane Computing Ltd.
Northwich, CW9 5HB. UK.
+44 (0) 1606 351006

3 richard@kanecomputing.com

4 kevin@kanecomputing.com

ABSTRACT

An Application Development Environment (ADE) incorporating a Concurrent Processing Framework (CPF) is proposed as a step-function improvement over existing software development tools for COTS multi-processor card accelerated-workstations. The ADE and CPF aim to enable development of cost-effective application solutions through targeting rapid application development and high operational efficiency at the same time as maintaining portability. The paper describes the ADE and the abstract multi-processor architecture on which it is based, going on to briefly introduce the CPF. A real-world (Near-Earth Object detection and tracking) large-scale image processing application is implemented (on a prototype accelerated-workstation built for proof-of-principle) using the CPF. Experimental results from this implementation demonstrate a 70% decrease in lines of C/C++ code compared with an implementation using existing software tools. Additionally show minimal execution overhead introduced through the use of the CPF and near linear performance scalability for the application.

Keywords

Multi-processor accelerator, image processing, rapid application development, performance scalability.

1. INTRODUCTION

Many compute-intensive application areas exist (e.g. large-scale image processing and 3D visualisation within non-intrusive inspection, non-destructive testing and medical radiology etc.) where performance requirements are well above that achievable with general-purpose (e.g. Intel Pentium based) workstations. However, the afore-mentioned applications are well suited to parallel processing arising from their inherent natural data-level parallelism. As such it is pertinent to make use of existing COTS (Commercial-Off-The-Shelf) Multi-Processor Card (MPC)

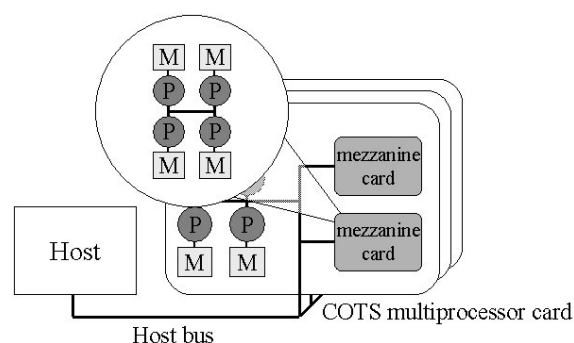


Figure 1. Multi-processor accelerated-workstation

accelerators (such as those from Mercury [7], Alacron [6] and Transtech [9] etc.) for the purpose of accelerating the general-purpose workstation, as shown in Figure 1. Performance scaling of the accelerated-workstation to match performance requirements is achieved through installation of additional MPCs.

High cost of MPC accelerators compared to general-purpose workstations means it is necessary to maximise cost-effectiveness of the overall solution through high operational-efficiency. Yet software development tools for available multi-processor accelerators (typically) do not readily lend themselves to parallel programming. Furthermore, they are targeted at use of the accelerator, independent of the workstation, and as such do not promote effective utilisation of the accelerated-workstation as a complete system. Application developers are, therefore, forced to work at a low-level of detail involving complex hardware and software development that is often outside of the domain of expertise available in-house. Long development times and high development costs ensue, derogating from overall cost-effectiveness as costs must be recovered through increased product price. Moreover, long development times not only increase the danger of missing windows-of-market-opportunity but also the inevitable danger of early hardware obsolescence. This results in poor return-on-investment and further deters application developers from accelerated solutions. Application developers instead settle for lower performance unaccelerated solutions, attracted by the benefits of well-established rapid application development environments (e.g. Windows PC based or UNIX based solutions). However, although taking this approach reduces development effort, it typically fails to meet

performance requirements.

The problem therefore, is to provide an application development environment that promotes rapid development of solutions attaining high operational-efficiency at the same time as maintaining portability. Such an environment would aid the application developer in addressing functional and performance requirements at the same time as maximising overall cost-effectiveness within a wide range of large-scale image processing applications.

This paper presents an Application Development Environment (ADE) as a solution to the problem described. An integral component of the ADE, the Concurrent Processing Framework is briefly presented. Large-scale image processing application performance requirements are then discussed and a Near-Earth Object detection and tracking application is used as a large-scale image processing exemplar. A description is given of an experimental accelerated-workstation prototype used for proof-of-principle and to derive metrics for cost-effectiveness. Experimental results from the NEO application, implemented on the prototype using the ADE and CPF, are presented showing the applicability of the ADE for rapid application development on accelerated-workstations, as well as the performance scalability of the prototype. Conclusions are drawn on the efficacy of the ADE and CPF for cost-effective application development using multi-processor accelerated workstation development.

2. APPLICATION DEVELOPMENT REQUIREMENTS

To promote rapid application of operationally efficient solutions, application developers would ideally benefit from the abstract multi-processor architecture and application development environment, discussed in the following subsections.

2.1. Abstract multi-processor architecture

The abstract multiprocessor architecture as shown in Figure 2, helps partitioning of data over the workstation and accelerator local memories (Ms) and facilitate configuration of processors (Ps), so that the abstract architecture matches the natural-parallelism of the application. Moreover, as indicated in Figure 2, performance-scalability is achieved, without loss of data bandwidth, by simply adding P and M modules to the Inter-Processor Communications Network (IPCN) of the distributed-memory multiprocessor architecture. This high-level view of the MPM (Multiple Program, Multiple Data) and SPMD (Single Programs, Multiple Data) parallelism supported by the multiprocessor accelerator (see Figure 1), enhanced by the MIMD (Multiple Instructions, Multiple Data) and SIMD (Single Instruction, Multiple Data) capability of individual processors, helps programmers to achieve much higher degrees of parallel computing efficiency.

2.2. Application Development Environment (ADE)

An ADE is provided (shown in Figure 3) to help implementation of parallel algorithms using the capabilities of the accelerated-workstation as a whole. The environment comprises three main blocks - Host, Accelerator and an optional Simulator (to enable development on unaccelerated workstations) glued together by the

Concurrent Processing Framework (CPF) described in Section 3. The three main blocks of the ADE are outlined, with reference to Figure 3, in the following subsections.

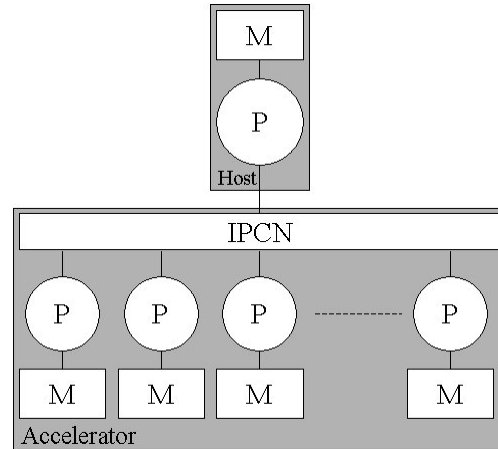


Figure 2. Abstract multiprocessor architecture

2.2.1. Host-side ADE components

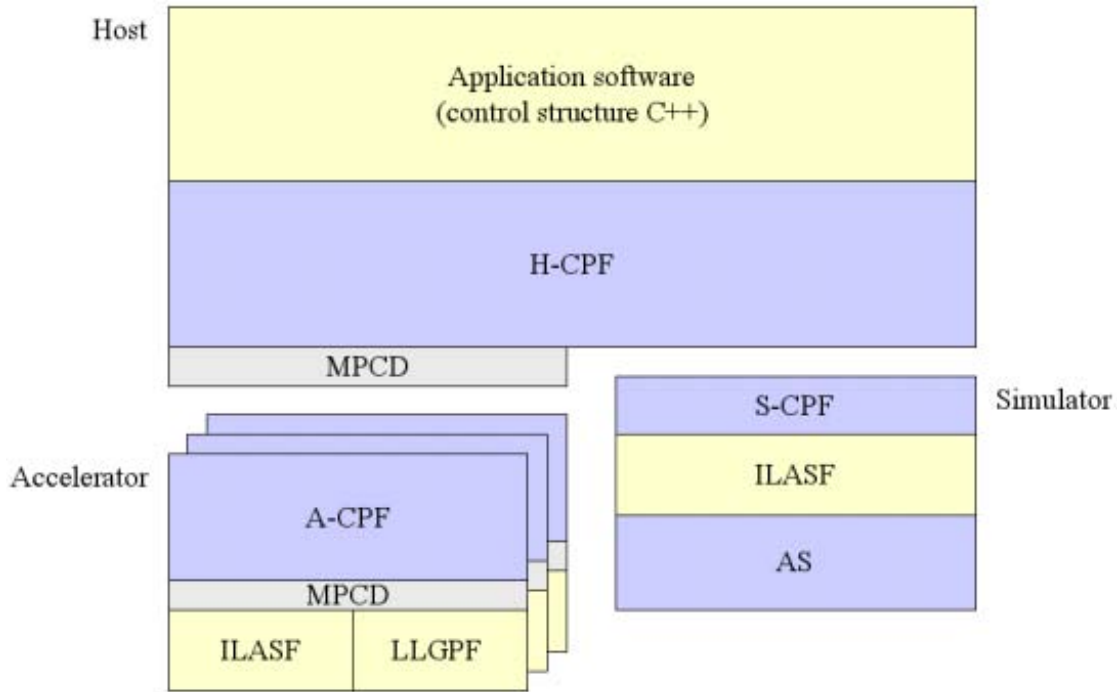
Using a GUI (Graphical User Interface) (proprietary or integrated with an existing third party tool), host-side software facilitates application development through an object-oriented view (implemented as an Application Programming Interface or API packaging a library of C++ classes) of the abstract multiprocessor architecture (see Figure 2) on which parallel algorithm design was based. A key feature of the ADE is that host-side software is easily portable across a range of accelerator hardware by simple substitution of the host-side MPC driver (MPCD) provided by the COTS multiprocessor card vendor. In addition to this, re-implementation of a software layer (internal to the Host-CPF) wrapping the functions of the MPC driver, is required.

2.2.2. Accelerator-side ADE components

To ease the application development process using the accelerated-workstation as a complete system, both accelerator processors and (conceptually) host-executed virtual processors are treated as entities on which accelerator-side components execute. Accelerator-side software simplifies development through an object-oriented view of the processor. This eases use of efficient Intermediate-Level Application-Specific Functions (ILASF) and highly-efficient Low-level General-Purpose (C/C++ and/or assembly language) Functions (LLGPF) within accelerator-side application code. Portability of the accelerator-side ADE to other accelerator hardware requires re-writing of assembly language functions and again the substitution of the accelerator-side MPC driver provided by the COTS multiprocessor card vendors (and re-implementation of the software layer wrapping calls to driver functions).

2.2.3. Simulator

This optional ADE component enables application software to be seamlessly executed on unaccelerated workstations (e.g. laptops or home computers). Off-line application development increases programming productivity where access to an accelerated workstation is either uneconomic or inconvenient.



H-CPF: Host-CPF,
MPCD: Multi-Processor Card (MPC) Driver,
ILASF: Intermediate-Level Application-Specific Functions,
LLGPF: Low-level General-Purpose Functions,
A-CPF: Accelerator-CPF,
S-CPF: Simulator-CPF,
AS: Accelerator Simulator.

Figure 3. ADE software layers

3. CONCURRENT PROCESSING FRAMEWORK (CPF)

The CPF uses a client-server (master-slave) parallel programming paradigm relating to the workstation (or host) (client/master) and accelerator (server/slave) processors. As indicated in both Figures 3 and 4, the CPF facilitates development of host-accelerator interaction by providing high-level parallel programming functionality and hiding low-level software and hardware complexities associated with data transfer and process synchronisation.

As shown in Figure 4, the client-side program runs on the host processor and implements high-level application control. Server-side processes executing on the processors of the accelerator (or within threads simulating the accelerator on an unaccelerated workstation) react to this control. Communication between client and server processes takes place via message passing.

Details of the CPF are beyond the scope of this paper and a more comprehensive description is given in [2].

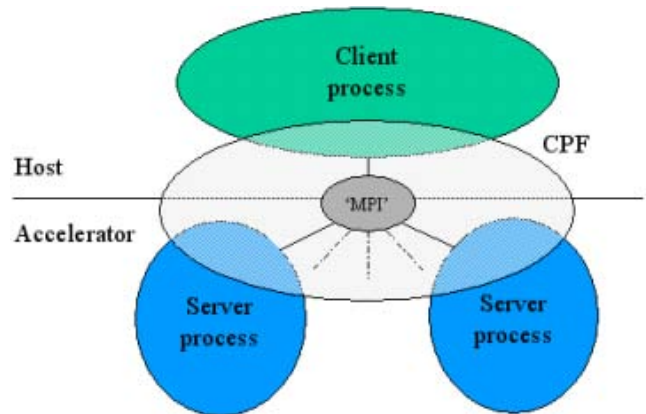


Figure 4. Concurrent Processing Framework

4. EXPERIMENTAL PROTOTYPE

A prototype accelerator incorporating a 1.6GHz Pentium IV host processor connected to 2GB of 133MHz SDRAM via a 64b local memory bus and 2 'FastImage' PCI multiprocessor daughter-boards [6] (see Figure 6) each supporting 1 or 2 'Fast4' PMC (Processor Mezzanine Card) multiprocessor daughter-boards [5] supplied by Alacron (Nashua, NH) interconnected via a 32b 33MHz PCI bus has been designed, built and tested by the

authors. Both the 'FastImage' and the 'Fast 4' daughter-boards accommodate 4 180 MHz TriMedia 1300 [8] processors each connected to 16MB of 143MHz SDRAM via a 32b local memory bus. Maximal parallelism is achieved by fully exploiting the 5-issue VLIW microarchitecture of the TriMedia processors supported by 128 registers with a 16kB data cache and a 32kB instruction cache.

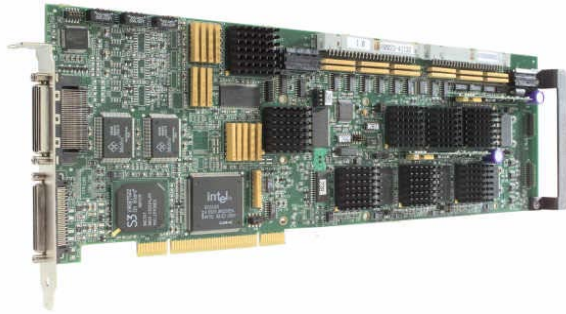


Figure 5. FastImage PCI multiprocessor card

5. LARGE-SCALE IMAGE PROCESSING

Typically an array of sensor chips (e.g. CCDs) is used for data acquisition in order to provide as high resolution as possible. Varying array arrangements are used (e.g. butted or with columns staggered by half a chip) with state-of-the-art CCD arrays capable of integrating over 8k x 8k (12b-14b) pixels (64M pixels in total).

Image processing functionality to support processing of data acquired from the sensor array includes:

- image restoration:
to compensate for limitations (e.g. ageing and faulty sensors, aberrations etc.) of the data acquisition system
- image enhancement:
to make objects more prominent
- image analysis:
to detect, discriminate and measure the dimensions of particular objects
- measurement and decision making:
which might include visualisation.

Considering such algorithm complexity it becomes evident that the number of operations per pixel could be as high as 1000.

The number of frames-per-second is dependent on sensor integration (exposure) time, which increases as light intensity decreases, and read-out time, which reduces as the number of data channels in the sensor array increases. Both of these times can range from a fraction of a second to a few minutes and there is no value to be gained from processing faster than the sum of the two. However, to prevent a backlog of data building up which might require farms of hard-disk storage, data are ideally processed on-line (i.e. in real-time) by a computer closely coupled to the data acquisition device (for high-bandwidth transfer of large volumes of data).

In summary, for large-scale image processing, performance requirements range between 100 Mega-OPS and 1Tera-OPS.

With the PMC cards mounted, the FastImage cards occupy 2 PCI slots each and, hence, the prototype could be extended up to 36 processors, supported 576MB SDRAM in a traditional PC tower using the free PCI slots on typical COTS PC motherboards. Alternatively, a standard PCI rack could accommodate up to 96 processors supported by 64MB - 1536MB SDRAM.

5.1. Application exemplar: NEO detection and tracking algorithm

To illustrate the use of accelerated-workstations, a NEO detection and tracking application was selected as good example of where (at the time of specification a decade ago) processing requirements were far in advance of general-purpose workstations. Background information regarding NEO programs are outside of the scope of this paper and further details can be found in [1] and [3].

A high-level model of a NEO detection algorithm is shown in Figure 5. The algorithm makes use of two types of processing - iconic and symbolic. Iconic processing is used to find and extract objects within an image, whereas symbolic processing of individual objects is used to calculate information such as centroids and moments etc. as well as comparison of lists of objects for classification of threatening and non-threatening NEOs.

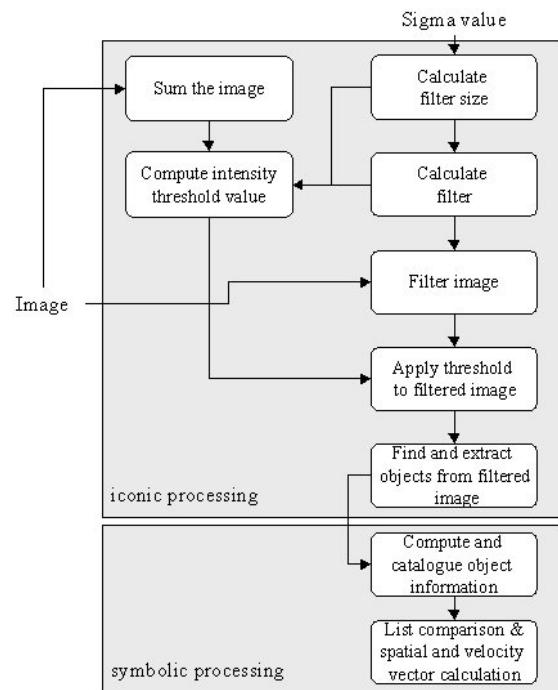


Figure 6. NEO detection algorithm

6. EXPERIMENTAL RESULTS

Taking the NEO application exemplar described in Section 5.1, the ADE and CPF are evaluated using the experimental prototype described in Section 4. Using this experimental vehicle, cost-effectiveness metrics have been derived and are described in the

following subsections.

Two implementations (one using the MPC driver from Alacron [4] and the second using the CPF) of a cut down version of the NEO detection algorithm (i.e. ignoring object list comparison) described in Section 4.1 are used for comparison.

Comparisons between the two versions highlight the efficacy of the ADE in rapid application development and in maximising operational-efficiency, as described in the following sections.

6.1. Rapid application development

Comparison of the listings for the two versions demonstrated a 70% reduction in the total lines of (C/C++) code through the use of the ADE and CPF, demonstrating reduced development effort.

6.2. Performance scalability

Results for performance scalability fall short of ideal speed-up as shown in Table 1. This is due to the algorithm requiring image partitions to be overlapped for each processor such that the overlap processing overhead increases as the number of processors grows.

Table 1. Experimental results showing performance scalability

Processors	Execution time (sec)	Speed-up	
		Ideal	Actual
1	24.51	-	-
2	12.34	2	1.99
4	6.53	4	3.75
8	3.4	8	7.21
16	1.85	16	13.25

6.3. Operational-efficiency

Minimal execution overheads (an average 0.16%) of the application implemented using the ADE and CPF compared to the version implemented with the MPC driver are shown in Table 2. Variations in overhead over increasing number of processors are due to subtleties in memory allocation and deallocation between the two versions. This is a clear indication of the low overhead introduced by the use of the ADE and CPF.

Table 2. Execution overheads of the ADE

Processors	Overhead (seconds)	% Overhead
1	-0.026	-0.11
2	-0.028	-0.23
4	0.001	0.02
8	0.025	0.74
16	0.07	0.38

7. CONCLUSIONS

Large-scale image processing application developers desiring the performance benefits of accelerating workstations with COTS Multi-Processor Cards (MPCs) have been deterred from their use by the long and often unpredictable software development times resulting from the complex and low-level detail with which they are forced to work.

This paper has proposed an Application Development Environment (ADE) and Concurrent Processing Framework (CPF) as a solution to this problem. In particular to provide a route for rapid application development of operationally-efficient solutions using accelerated-workstations.

The ADE and CPF have been successfully designed and implemented using a prototype accelerated-workstation as an experimental research vehicle. A number of exemplar applications have been developed using the ADE and CPF on the prototype accelerated-workstation, one of which (Near-Earth Object (NEO) detection and tracking) was selected as an exemplar for this paper.

Metrics derived from experimentation using this exemplar demonstrate very encouraging reduction in application development time with very low execution overheads arising from the use of the ADE and CPF. Indeed, results show evidence of rapid application development through a 70% reduction in lines of (C/C++) code, with an average execution overhead of 0.16%. Use of the simulator throughout application development enabled off-line development and is currently used to demonstrate the parallel NEO exemplar on un-accelerated workstations. Although not demonstrated in this paper, portability is achieved through re-implementation of the MPC driver abstraction layer internal to the CPF, as indicated in Figure 3 and the discussion in Section 2.1 and 2.2.

In conclusion, results have demonstrated the effectiveness of the approach described, for rapid development of large-scale image processing applications on multi-processor accelerated workstations, while attaining high operational-efficiency and maintaining portability. Therefore, the ADE and its integral CPF help application developers to maintain overall cost-effectiveness for multi-processor accelerated workstation solutions.

REFERENCES

- [1] Chapman, S., Tetnowski, P.T., and Lea, R.M. Accelerating large-scale image processing applications with performance-scalable multiprocessor workstations, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02), Las Vegas, June 24-27 2002, vol III, pp1141-1144.
- [2] Chapman, S., and Lea, R.M. A portable concurrent processing framework for cost-effective multiprocessor application development, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02), Las Vegas, June 24-27 2002, vol II, pp545-551.
- [3] Morrison, D. The Spaceguard Survey - Protecting the earth from cosmic impacts, 1992, Mercury, vol. 21, no. 3, pp. 103-106.

- [4] ALRT Runtime Software Programmer's Guide and Reference for FastSeries Processor Boards. 30002-00169, Alacron Inc., NH, 2000.
- [5] Fast4/1300 Hardware Users Guide, 30002-00180, Alacron Inc., NH, 2000, pp.3-11.
- [6] FastImage1300 Hardware Users Guide. 30002-00176, Alacron Inc., NH, 2000.
- [7] RACE++ Series VantageRT 7400. DS-4I-11, Mercury Computer Systems Inc. MA. 2000.
- [8] TriMedia32 Architecture. Ver. 1.0.2, TriMedia Technologies Inc., CA, 2000, pp.11-21.
- [9] TS-P36N Quad ADSP-TS101 DSP PCI Card with Xilinx FPGA. TSP36ND0102, Transtech DSP, NY, 2001.