A Beowulf-class architecture proposal for real-time embedded vision.

P.A. Revenga †, J. Sérot ‡, J. L. Lázaro †, J.P. Derutin ‡

† Dpto de Electrónica
Universidad de Alcalá de Henares
Madrid, 28806, Spain
revenga,lazaro@depeca.uah.es,

‡ LASMEA, UMR 6602 CNRS
Université Blaise Pascal.
Campus des Cézeaux, Aubière, France
jserot,derutin@lasmea.univ-bpclermont.fr

Abstract

In this paper a new type of parallel computer architecture dedicated to real-time vision is proposed. The proposed architecture is based upon the Beowulf *concept – i.e. COTS computing nodes interconnected by a high-speed network and running standard, free software tools – but also takes into account the constraints of* embedded *vision systems, such as "on the fly" processing of video streams, small volume, and low power consumption. Its main originality lies in the presence of two separate communication media: a dedicated bus for fast video i/o and a standard, switched inter-connection network for inter-process communication. Another distinctive point is the use of a high-level parallel programming model, based upon* algorithmic skeletons. *In order to assess the validity of the proposal, a first prototype has been built and is described in the paper. It uses G4 motherboards coming from Apple Cube mass market computers, a Fast Ethernet communication network and a FireWire (IEEE-1394) video bus. Results of preliminary benchmarks are presented.*

1 Introduction

Compared to classical HPC (High Performance Computing) applications, embedded vision applications, raise two specific issues: first, they implement *reactive systems*, operating "on the fly" on digital *streams* of images. This means that they must be able to absorb input data and output results at a minimum frequency and produce responses within a maximal latency. Second, they must meet stringent operational constraints in terms of volume and power consumption. These applications may be found for instance in autonomous navigation vehicles, like space rovers, satellites, robots and cars equipped with assisted-driving systems. In most of the cases, the need to provide high performance while meeting the afore-mentioned constraints effectively rules out implementations based upon stock-hardware.

In the context of High-Performance Computing, *Beowulf*-class parallel machines [3, 18] are enjoying an ever-increasing success. A *Beowulf* is a multi-computer, scalable architecture consisting of mass market common off-the-shelf components running a freely available operating system and software packages, and inter-connected *via* a high-bandwidth network like Fast Ethernet, Giganet, SCI, Myrinet or ATM. Typical *Beowulfs* are made of industry-standard PC CPU and interface cards and run commodity software like Linux OS and the PVM or MPI message passing libraries. Compared to other parallel architectures (vector computers, MPPs, SMPs), the main advantages of *Beowulfs* are their incomparable performance/cost ratio, their scalability – a *Beowulf* can be viewed as a *cluster* of pluggable CPU+memory *compute nodes* – and their ability to take advantage of advances in processor and networking technology. But most of *Beowulf* machines built today are dedicated to number-crunching, off-line computations (simulation, data-mining, ...). For these applications, the so-called "pile-of-PC" approach to clustering is effective: machines are built by "stacking" standard PC enclosures, interconnecting them via a dedicated switch and making i/o from/to disk on one or several dedicated server nodes. For embedded real-time vision systems (such as those embarked in autonomous vehicles) this approach is clearly not possible, due to the above-mentioned constraints (volume, power consumption).

The work described in this paper therefore aims at exploring the conceptual and technological issues associated to the design of a *Beowulf* architecture dedicated to the real-time (on-the-fly) processing of digital video streams for embedded vision systems. The paper will be organized as follows. Section 2 will recall the requirements for such systems. It will then be showed that none of the existing approaches can fulfill all these requirements and an extension of the *Beowulf* concept will be proposed as a possible

0-7695-1926-1/03/$17.00 (C) 2003 IEEE
Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)

answer. The resulting architecture will be defined in Section 3 and a prototype instantiation of this architecture presented in Section 4 with results of preliminary benchmarks. Section 5 will include a short survey of related work and Section 6 will concludes this paper.

2 Architectures for real-time embedded vision

Systems for embedded real-time vision must meet several constraints:

High performance. This comes from the necessity to process "on the fly" digital *streams* of images. This means that these systems must be able to absorb input data and output results at a minimum frequency and produce responses within a maximal latency. For assisted-driving applications, for instance, the typical frequencies and latencies are in the range of 10-100 frame/s and 4-40 ms respectively. For reasonably complex applications (involving 10-100 elementary operations per pixel) and medium-sized images (512×512), this requires a computing power in the range of 0.1 to 5 Gflops.

Scalability. This refers to the ability to increase the system performances without changing its architecture, by simply adding computing nodes, to match the application requirements.

Resistance to obsolescence. Dedicated architectures can be made obsolete very quickly by the rapid evolution of micro-processor technology. It is therefore crucial for these architectures to provide an upgrade path, by which technology advances can be included without major changes in the system architecture (by simply replacing processors for instance).

Volume and power consumption. This very pragmatical concern is critical for systems that must be embarked on vehicles or on satellites. For instance, Pentium processors are known to have a high power consumption and therefore require large power supplies and big heat dissipation systems. The critical point here is the ratio of computing power to power consumption (Flops/Watt) or of computing power to volume (Flops/dm^3).

Easy to use and efficient mechanism for video data acquisition. This refers to the possibility to use mass market, cheap video sub-systems (cameras, displays, recording devices, etc.) on the one hand, and to the ability to automatically and efficiently *broadcast* video data coming from an input device to all computing nodes on the other hand. The second point was proved to be of crucial importance in multi-processor architectures: in this case, the cost of explicitly broadcasting an image from one processor (the one controlling the frame grabber for instance) to all the others can easily destroy any potential gain obtained by parallelizing processing on this image.

Cost. This issue is more likely to be solved by resorting to M^2COTS[1] components both for the i/o sub-systems and the computing devices (processors, networking, etc.).

Programmability. This refers to the availability of user-friendly programming models environments and tools, including C (as opposed to assembler) compilers, debugging and profiling tools, application-specific libraries, etc. For multi-processors, a high-level parallel programming model is required, since low-level parallel programming models – relying on explicit message-passing like in MPI [11] or on shared-memory thread coordination like in OpenMP [13] – are known to place too much burden on the application programmer and to be too error-prone. This point is of critical importance if the machine is intended to be used by people who will not be parallel programmers in the first place.

None of the various attempts that have been made to build embedded systems for real-time vision have met all of these criteria.

Solutions based upon stock-hardware, high-end PCs may provide, in certain circumstances, the required computing power but raise significant problems as regards volume and power consumption. These problems can be solved by resorting to laptop PCs but at the expense of a significant increase in the cost/performance ratio. Scalability remains, anyway, problematic.

Architectures built from specialized processors (DSP, FPGA, ASIC) generally offers the best performance/volume or performance/watt ratio but are difficult to program (often requiring skills in assembler or VHDL programming). Interfacing to COTS components for video i/o may also be far from trivial.

Dedicated multi-processor machines, such as the TRANSVISION platform [9] or the SYMPHONIE machine [6] generally offer good scalability and better programmability but have proved to suffer from quick obsolescence.

On the other hand, the Beowulf approach to high-performance computing seems to match almost all of the above-mentioned criteria. The MIMD general architecture offers good scalability. The use of standard COTS components, both for CPU nodes and network devices provides an easy and cost-effective way for upgrading and hence resistance to obsolescence. The use of standard software components for OS (Linux) and programming tools makes them easy to operate and maintain. Moreover, high-level parallel programming tools can easily be ported to these architectures, thanks to OS-level standardization. However, two issues must be addressed for a Beowulf-like architecture to be used for embedded real-time vision. The first

[1] Mass Market Commodity Off The Self

0-7695-1926-1/03/$17.00 (C) 2003 IEEE
Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)

one is the definition of an efficient hardware and software mechanism for broadcasting video data. The second concerns volume and power consumption. It is clear, indeed, that the so-called "pile-of-PC" approach used for existing Beowulf clusters – in which machines are built by simply "stacking" standard PC enclosures, interconnecting them via a dedicated switch and making i/o from/to disk on one or several dedicated server nodes – does not meet the volume and power consumption constraints of embedded vision systems. Answering these questions led us to propose a variation on the *Beowulf* concept dedicated to real-time embedded vision. This proposal is described in the next section.

3 Architecture proposal

The specific issues raised in the previous section (efficient broadcasting of images and low volume and power consumption) can be addressed by first relying on a dedicated bus for broadcasting images from video source(s) to computing nodes (and for sending video results to displaying devices) and, second, by choosing the computing nodes offering the best Gflops/watt and Gflops/dm^3 ratios in the available technology. The first point actually amounts to building a dual-network architecture, as illustrated in Fig. 1: one network (bus) dedicated to the communication of video data (preferably in a timely, synchronous manner), and another for process inter-communication. The latter can be any of the solutions used for "standard" Beowulfs (Fast Ethernet, Gigabit Ethernet, Myrinet, etc.). For the former, the best solution, in the current state of the art, seems to be the IEEE-1394 (Firewire) bus. IEEE1394 is an international, non-proprietary and inexpensive standard hardware-software digital interface for data transporting up to 400Mbps. It has a flexible topology, is hot pluggable and configurable. Digital video devices can send digital video data, can be controlled and powered by the bus. There are two types of IEEE 1394 data transfer: asynchronous and isochronous. Asynchronous transport is the traditional computer memory-mapped, load and store interface. In addition IEEE 1394 features a unique isochronous data channel interface – providing guaranteed data transport at a predetermined rate – and the possibility of controlling CMOS cameras with sub-sampling and windowing capabilities.

More latitude is given for the choice of the computing nodes. We have been experimenting with Power-PC G4 mother-boards coming from Apple Cube machines [7]. Several features of the G4 processor – and of its Cube incarnation – make it highly attractive in our context: First, it can deliver impressive performance, even at moderate clock frequencies, thanks to its built-in *Altivec* vector processing unit [1, 14]. The *Altivec* extension is specially useful for low-level iconic processing, such as found in the first stages of most of vision applications (in [8], speedups in the range of 10-15 are reported for 1D and 2D signal processing applications). Moreover, and contrary the Pentium MMX/SSE similar extensions, using the Altivec does not require assembly-level programming, thanks to the availability of an Altivec-aware version of the `gcc` compiler [2]. Second, the G4 processor has a very good Gflops/Watt ratio, with a peak power of 1 Gflops (with Altivec) and a power consumption of less than 4 Watts. Third, the Cube motherboard is very small (16x19 cm) (see Fig. 2). After removal of unnecessary devices (hard-disk, CD-drives, display controller, etc.), it should be possible to pack four G4 motherboards in the original Cube enclosure (20x20x20 cm, see Fig. 3).

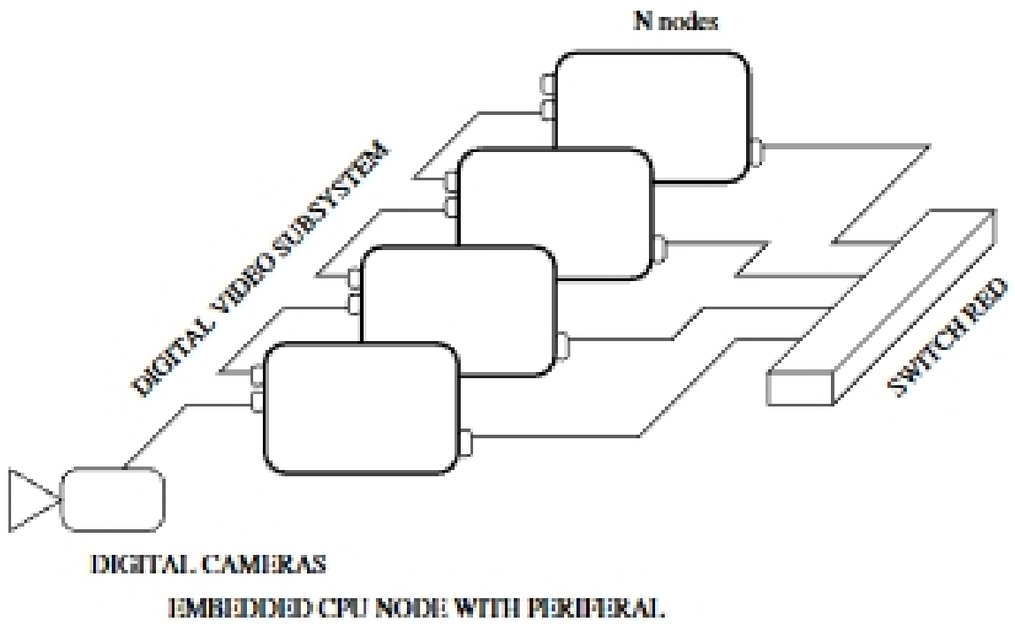

Figure 1. Ossian architecture proposal

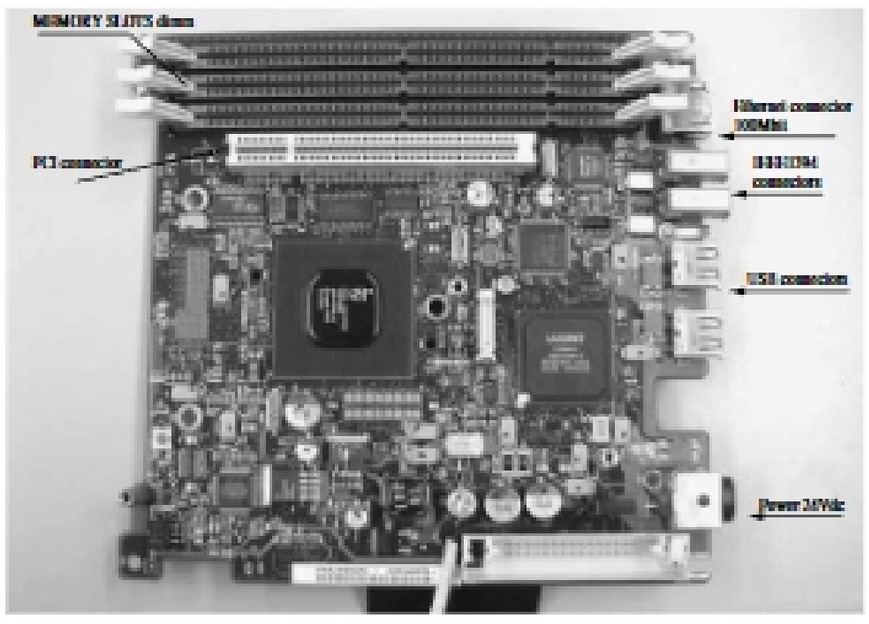

Figure 2. The G4-Cube motherboard

The software architecture of the proposed platform is a three-layered one:

- The operating system is Linux, following the *Beowulf* tradition. This is for cost and portability. The Linux distribution must of course support the target processor and chip-set (G4 in our case) and provide drivers

0-7695-1926-1/03/$17.00 (C) 2003 IEEE
Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)