Computer platforms for hard-real time and high quality ergotic multisensory systems

Nicolas Castagné, Jean-Loup Florens, Annie Luciani

Computer Platforms for Hard-Real Time and High Quality Ergotic Multisensory Systems

- requirements, theoretical overview, benches -

Nicolas Castagne Jean-Loup Florens Annie Luciani
ACROE-ICA laboratory, INPG, Grenoble, France
E-mail: castagne@imag.fr, luciani@imag.fr, luciani@imag.fr

Abstract

The article summarizes the results of a study performed in 2004-2005 in order to choose the next generation of computer platforms (hardware, operating system) for implementing high quality enactive systems capable of supporting multisensory tasks, with ergotic gesture interaction. This category of tasks are amongst the most demanding as for the technology of man-computer interaction systems and point to several bottlenecks.

The requirements as for the performance of the computer platform are summarized. Theoretical views on appropriate hardware are given. The features that the Operating System should provide, especially regarding real time capacities, are discussed, and various existing OS are reviews. Exhaustive benches of 10 platforms (hardware and OS) are reported.

1. Introduction

Some human tasks feature a physical gesture interaction between humans and physical objects, and a multisensory context. These tasks are usually called "manipulation tasks", as compared to other categories of tasks such as navigation, localization, identification, etc. However, when willing to consider the technology to be implemented for supporting such tasks, the term "manipulation" is too general. One may additionally distinguish the tasks in which the haptic modality is involved. Unfortunately, the term "haptic" covers several meanings. It is often used as "an umbrella term covering all aspects of manual exploration and manipulations by humans and machines" [1].

For such reason, focusing on the discriminating property of energy exchanges, Claude Cadoz introduced [3, 4] the term "Ergotic" to distinguish the category of manipulation tasks in which there is a significant "mechanical energy exchange", not only between the human and the mechanical manipulated object but also between the parts of the object that produce the other distant sensible phenomena (the acoustical and the visible deformations). In Ergotic tasks, the energetic coupling modifies the physical states of both objects and humans. New mechanical properties emerge, that are significant not only for the control of the performance on the side of the human, but also for the result of the task.

Grasping, pushing, cutting, throwing, carrying, moulding, hitting, rubbing, breaking, writing, playing the violin, manipulating a marionette, etc. are ergotic tasks. Conversely, pointing an object, moving around an object (walking, etc), cutaneous touch, being pushed by an infinitely heavy object, pulling an infinitely light object, are situations that do not feature a noticeable ergotic interaction.

Only a few Virtual Reality systems focus on ergotic multisensory tasks, and major improvement in the quality of these systems are needed. Indeed, various publications discuss why implementing ergotic tasks in virtual environments is particularly difficult [5, 10]. It requires obtaining energetically valid hand-to-eye and hand-to-ear chains, and an accurate rendering of all the energy exchanges. A high quality correlation between the dual input and output variables on the gesture device, and more generally a high quality coupling between all the parts of the virtual simulated object are needed. The corresponding technological requirements are numerous and critical, especially in terms of reactivity and time-determinism for high frequency hard real-time synchronous computations [10]. They concern all the components of the system: quality of the force feedback devices, input and output means of the computer (buses, AD/DA converters, …), computer hardware reactivity and power, features offered by the OS, etc.

The study reported in this paper focuses on the computer platforms: hardware + operating system. It does not pretend to be an exhaustive and rough evaluation of all the 'real time' platforms, nor to evaluate the I/O means, but aims at evaluating various foreseen platforms in the specific context of ergotic multisensory tasks. The paper details the performances expected, reviews the possible hardware and operating systems, and presents the results of a series of benchmarks of 10 promising platforms.

Proceedings of ENACTIVE05
2nd International Conference on Enactive Interfaces
Genoa, Italy, November 17th-18th, 2005

2. Requirements for The Platform

The requirements as for the computer platform (computer hardware+OS) for supporting ergotic multisensory tasks concern various critical technical performances, but also other aspects related to the usability of the platform in correspondence with the context in which one can foresees its usages.

3.1. Latencies, Reactivity, Synchronism

In order to ensure coherency over the various sensory modalities, the algorithm must rely on the principle of synchronous hard real time. All the various processes and data flows are fully synchronized on a single clock [8].

3.1.1. Input / Computation / Output sequentiality
The sample available on step N for the output devices is computed directly from the data acquired from the gesture device on step (N-1), by executing a single computing loop, without any buffering of the input/output data. Various phases must be computed in a time window of duration T, which corresponds to the simulation frequency:

- acquisition of the data from the gesture device (A/D conversion)
- transport of the data from the ADC device to central memory along the buses
- *computation* performed by various processes
- transport of the data from the central memory to the DAC device along the buses
- D/A conversion (gesture, sound, image)

Though some pipeline optimization are possible, these phases cannot be processed in parallel.

3.1.2. Inter-process *rendez-vous*. In the case of multi-processor configurations, in which computations are split amongst various processes and processors, data flows must be exchanged on the frontier between processes in the whole computed model. Various bidirectional *rendez-vous* (namely : 2 *rendez-vous*, to exchange respectively the dual force and position variables) are necessary on each step. This splits the "computation step" discussed in §311 (Figure 1).

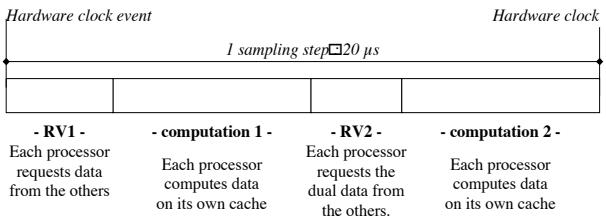

Figure 1: Fully-synchronized algorithmic scheme. The inter-process rendez vous, the per-processes computations, and the input/output phases for the gesture interaction must be executed sequentially [8]

3.1.3. Summary and quantification. The algorithmic scheme drawn out implies high frequency *rendez-vous* between processes to ensure synchronism of the computations, and many data flows (between processor caches, central memory and processor caches, memory and DAC/ADC chipsets along buses, etc).

Data exchanges and *rendez-vous* correspond with incompressible periods of time that penalize the fixed time window defined by the simulation frequency. They limit drastically the time available for performing the computations itself without breaking synchronism.

This fully synchronous real-time processing scheme is the basis of the most important technical requirement for the platform: latency must be minimal, in all its possible meanings. This concerns A/D-D/A conversions, data transfer along the buses, data exchanges between processor, memory, and processor caches in the case of multiprocessor platforms.

As a far end goal, one may aim at computing sounding models in the range of sound frequency, that is +/-50000 Hz. In case of a multi-process algorithm, this implies a time window of 20 µs to carry out, in a sequential manner, the inter-process synchronizations, the data exchanges between processors, and the computation of the model (Table 1a). As for the data exchange with the gesture DAC/ADC, a frequency of 10 kHz should be obtainaible for the more demanding research implementations (Table 1b).

Table 1: Objectives for inter-processors communications and input/output buses performance

a. Inter-processor communication
The time for a single rendez-vous (waiting time + data exchange) between processes (ie: between processors, since a single process run on each processor, see §4) should be less that 1 µs – in order to keep at least 18 µs available for computation, for a sound simulation ran at 50 000 Hz.
The size of the data exchanged on each rendez-vous between processes, in each direction, is close to 64 double precision floats (ie 512 bytes).
b. Latency for gesture data Input/Output.
Though the article is not focused on I/O, one can state, as an example, that a piano keyboard-like interaction would require 64 input and output conversions on 16 bits, and the transfer of 64*4 bytes along the buses, in less than 5 µs. In that case, (100-5=95µs) are left for computing the gesture data within the model, in the 100 µs time window that correspond with a 10 kHz gesture flow.

3.2. Regularity, Time Determinism

The algorithms must be computed within fixed-time windows in a predictable time. Any time irregularity (which may be due, for example, to a peculiar memory configuration, to memory swapping, to the global workload, to the handling of interrupts, etc.) may result in a synchronism error, and could do nothing but stop the synchronized loop.

3.3. Computation Power

Computing power is another technical requirement for the platform, though far more common than determinism and reactivity. It impacts the maximum complexity for the models.

3.4. Usability, Modularity

The ergotic multisensory system at hand relates with the so-called 'vis-à-vis' approach to virtual realities, in which one aims at implementing high quality 'table-sized systems', as opposed to simulators based on dedicated spaces (cave, show rooms, etc). The far-end-view idea is to empower the *usual* workspace of the user. Ideally, the ergotic multisensory means should extend the platforms that are used daily by the foreseen users, in their every day tasks – that is: desktop "General-Purpose" computers – by adding "natively" high quality interactive simulation means.

However, one must also consider the need of highest quality platforms for laboratory research, conceived as measurement instruments to focus psychophysics and cognitive issues such as "degree of realism", "believability", "robustness of the percept", "multisensory consistency", etc.

Consequently, a customable platform is needed, allowing a range of final products of various sizes, prices and performances: low level quality obtained by upgrading every-day General-Purpose computers; mid level platforms on specific, but low cost, computers; high level platforms with top-of-the-range computers for the most demanding experiments. Additionally, one needs a true continuity (in hardware, OS, and applicative tools...) over the range of platforms.

Given this overview of the usages, we endeavoured to study proprietarily the common 'General-Purpose' platforms. For each one, we studied whether it is possible or not to implement the functionalities needed for the lowest quality systems. Then, for supporting the highest needs, we evaluated the top of range for these General-Purpose platforms: most powerful hardware available, and, eventually, real-time oriented upgraded versions of the 'General-Purpose' OS.

4. Discussion on Hardware

The theoretical analysis of hardware architectures focused on mono-processor architectures *vs*. multi-processor architectures, 32 and 64 bits architectures, PC and professional platforms, and PC clusters.

Today's cluster architectures do not fulfill the latency and reactivity requirement, we found. None of the network technology we studied (Ethernet, Myrinet, Quadrix, SGI NUMAflex, etc.) offers the needed performances. Though some hardware latencies are said to be particularly low, they are degraded at the software levels necessary to support them.

Mono-processor architectures are the most promising regarding the latency of memory access. However, running on a single processor the real time processe(s) in parallel with the system processes, and eventually "General-Purpose" processes, would hardly allow satisfying the time determinism requirement.

Consequently, multi-processors architectures, and particularly bi-processor, are the most promising architectures in our context. We concluded on the following principle for implementing these configurations: a single processor is dedicated to the system and 'General-Purpose' user processes, whereas each of the other processors is shielded against the OS and user processes and runs a unique process, dedicated to a part of the real-time computation.

A priori, 64 bit architectures are of interest, since they are particularly powerful when dealing with 64 bit floating point arithmetic, which is probably the case of the computations at hand. Though, in our context, the advantage of 64 bits architectures concerning the size of addressable memory space is not interesting. Moreover, 64 bit memory addressing may penalize the reactivity of the machine, since any pointer computation and pointer transfer would require 64 bit. Hence, choosing amongst 32 and 64 bits architectures is not simple and calls for further benchmarks.

Given the requirements for the simulator, and the fact that the performances of a given platform, especially in terms of reactivity, are hardly dependant on the cohabitation of its components, it was difficult (probably impossible) to extend these simple observations in a deeper operational analysis. Tests and benchmark were required.

5. Discussion on Operating Systems

5.1. "Hard" *vs.* "Soft" Real Time

The algorithmic scheme discussed in §2 is related to the principle of 'Real Time' (RT) computing. Nowadays, the terms 'real time' applied to operating system (OS) covers various meanings, depending on the needs at hand. Attempts are made, however, to clarify the field. According to IEEE [9], a real time OS is a system whose correctness includes response time as well as functional correctness. Behind this definition, one today commonly distinguish upon "hard real time OS" and "soft real time OS" (see for example [7]).

5.1.1. Soft real time. A soft real time OS is a system in which time-reactivity was paid a particular attention, for supporting interactive (though non necessarily synchronized) tasks. In soft real time OS, "real time" corresponds with the idea that the machine will be "powerful" enough to accomplish a given task in a satisfying time. However, this time is not guarantied: soft real time OS are non time-deterministic.

To fulfil these requirements, a soft real time OS should may feature asynchronous handling of events, optimized latencies when swapping the contexts of two tasks, and when managing interrupts, and a management of priorities for processes and interrupts.

5.1.2. Hard real-time. A hard real time OS is a system guarantying that critical tasks complete on time, allowing computations to be fully synchronized on some real world clock. Hence, fundamentally speaking, hard-real time does not focus on power at all, and does not put the emphasis on reactivity and latency, but rather on time determinism. However, usually, hard real time OSs also feature low latency means (i.e.: meet the soft real time requirements).

In hard real time OSs, one sometimes considers that secondary storage must be limited or unused while runing real time tasks, and that data should only be handled in short term memory, or even read-only memory. More generally, a hard real time OS should at least promise, whatever the state of the machine is (workload, memory load, etc.) a time-constrained handling of interrupts for real time tasks, a time constrained execution for any basic computation, a time-constrained management of buses, memory, etc.

The requirements discussed in §2 correspond as evidence more with hard- than with soft- real time. However, though they are operational at first glance, the notions of "interactive" (soft-) and "synchronous" (hard-) real time OS are not fully clear and distinct. For example, an OS oriented toward soft real-time may be sufficient to support certain hard (synchronous) tasks – such as playing a sound file, for example. A system, conversely, can hardly be defined to be "hard real time" in itself. Indeed, it should rather be said to be well suited for supporting *certain* hard-real time (synchronized) algorithmic schemes, with given time constraints. In the context of ergotic multisensory tasks, a more precise analysis of the features that the OS should provide is needed.

5.2. Specification for an "ideal" OS

First, the OS should offer a couple of common (e.g.: "non-real time oriented") features. It must be compatible with multi-processors architectures, which we found to be promising. The necessity of a range of platforms imply that it should not be restricted to certain categories of hardware, and should be compatible with the most recent processors. To allow multisensory implementation, it must support various peripheral cards and co-processors, and support OpenGL for managing the visual outputs. Finally, given the foreseen usages, the OS should be 'General-Purpose' compliant, and allow running, for example, IDE for development, office tools, multimedia common applications, modellers, etc.

As for the real-time oriented features of the OS, Table 2 summarizes those that we consider to be necessary to support the ergotic multisensory systems.

Table 2: Real-time oriented required features

Management of access rights	Features allowing granting exclusive right accesses to various resources (memory, processor, file system…) to the real time processes are necessary.
Processor Shielding against process execution	OS usually balance the workload between processors by dynamically affecting the processes on the various processors, under the responsibility of the scheduler. This does not satisfy the need for time-determinism. The following is needed: - the system processes and the non-real time applications are ran on a single processor (e.g.: the 'system processor') - all the other processors are dedicated to the real time computation. To that aim, the OS must allow shielding processors against process execution..
Processor affinity. Assigning processes to processors	The system must allow the user to assign any process on one of the shielded processors. The process is then migrated on the given processor, until an authorized user explicitly removes it.
Deactivation of the scheduler	For maximizing reactivity, a single process should run on each shielded processor. The OS should allow deactivating the scheduler, so that the active process cannot be pre-empted anymore.
Process priority	The OS should provide a way for managing processes priorities to be used by the scheduler when deciding which process should be activated or pre-empted. Priority management will be used to make more reactive those of the non – hard real time processes that must communicate with the hard-real time loop.
Shielding against interrupts, daemons, etc.	Processing system interrupts are a cause for non time-determinism. A selective interrupt shielding for processor/processes should be offered, so that the protected process is still reactive to the events on which it is supposed to react, though being not disturbed by others.
Protection against memory swap	Memory swapping (i.e. buffering of data from the central memory to a hard drive) is not compatible with time-determinism, and should be avoidable.
High quality inter-process synchro. means. **Spin locks / active waiting.**	Hard-real time programming requires particularly high quality and low latency synchronization mechanisms, for ensuring a very high reactivity when an event occurs. Given the use of numerous shielded processors running a single process, spin locking (while waiting, a process loops actively on a synchronization buffer) is probably the most efficient synchronisation mean. It should be offered by the OS.

5.3 An Operational Taxonomy of OS

The following offers an operational review of OS – though non-exhaustive, given the vivacity of the field.

5.3.1. General-Purpose time-shared systems. A time-shared OS uses CPU scheduling and multiprogramming to provide each process with a small portion of CPU time. Nowadays, all the common General-Purpose systems are time-shared systems. One can note that they have commonality of evolving toward soft real-time – for example by providing various levels of priority for the processes. Thus, they support weak real-time applications that require some reactivity and low latency (video streaming, gaming, multimedia, common virtual reality, servers, etc.) - but without guaranty for the task to be completed "in time".

Significantly, the most recent version of Linux (kernel 2.6) does incorporate more real time oriented features than other systems such as Windows and Mac OS. One can also note that a number of so-called « real time patches » for Linux 2.4 and 2.6 are available freely. We have tried them, with our Linux experts partner, but installation was not successful, or they did not run properly. Obtaining and maintaining a 'real time' Linux patch suite, we conclude, is today still a hard job – though potentially promising.

5.3.2. Embedded "real-time" systems. Historically, embedded (or dedicated) systems are 'minimal' OS dedicated to specific hardware (mobile phones, co-processors, etc.) and tasks. They are eventually real time oriented – indeed, one can observe that historically the topic of hard real time OS originated in the context of Embedded OS.

Nowadays, the so-called embedded OS are progressively over passing the strict field of embedded application and dedicated hardware. Some of them can run architectures amongst the most powerful (including multi-processor computers), meet the most common norms for programming (POSIX, etc.). By providing progressively high quality user-interface mechanisms (like sound and graphic cards, 3D means, GUIs, etc), they today tend to apply for General-Purpose usages.

Well-known examples of such growing embedded systems are BlueCat RT / LynxOS (LinuxWorks), VxWorks (WindRiver) and Neutrino (QNX). The three of them may be used nowadays in applications ranging from embedded systems to high performance computing on clusters, and support a large variety of Micro controllers, CPU microprocessors (Intel, PowerPC, MIPS…) and motherboards. They are, eventually, fully customizable, so that the OS features can be selected depending on the needs for a given RT application.

5.3.3. Micro-Kernel based "Real-Time" systems. These systems rely on a low-level real time micro kernel that runs under a more common time-shared General-Purpose OS, and that manages the resources (Figure 2). The time-shared system is the lowest priority, non real-time, task within the micro-kernel.

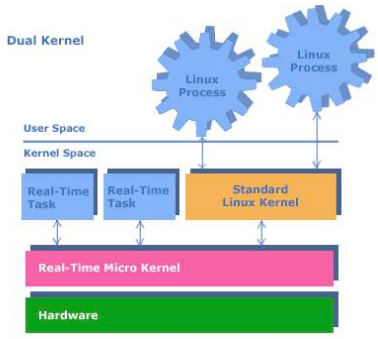

Figure 2. RTLinux software architecture. RTAI and INReal time follow equivalent principles (from RTLinuxFree technical sheet www.rtlinuxfree.com)**

RTLinux and RTAI are micro-kernel Linux-based real time systems. Within RTLinux and RTAI, real time programs must be designed as 'modules' of the micro-kernel. They do not have a direct access to the Linux features, especially to graphics. Various inter-process communication means are provided to allow a kernel task to communicate with Linux: FIFO, shared memory, interrupts, etc. When the use of a peripheral is required for RT tasks, (very) special drivers must be written for the micro-kernel.

INReal time Extension to Windows, or 'INtime', is also based on a micro-kernel. It is neither freeware nor open source. Real-time application development is necessarily made within "Microsoft Visual Studio" with "real time C++" libraries.

5.3.4. "Real-time" systems obtained by specialization. These systems are time-shared OS, specially tuned, and to which specialized feature have been added in order to make them satisfy real time requirements.

IRIX 6.5 is such a specialized time-shared system. It is available on SGI MIPS platform. Though IRIX is firstly a UNIX time-shared system, tools and system calls are provided for shielding processors against the scheduler and against interrupts. The processes ran on a shielded processor benefit from the whole CPU power. Though, determinism for using other resources (hard drive, buses, memory, etc.) is not fully ensured.

RedHAWK is the real time system by Concurrent Computer Corporation, fully based on the Linux RedHat distribution. It provides all the features offered by Linux RedHat, and can run any software compiled for Linux RedHat. Though, the Linux kernel has been deeply patched for supporting many soft- and hard- real

time features (including IRIX's, and more). These are available within a real time library. The OS is Open Source – but non-freeware. It is certified only on some platforms, but these are amongst the most powerful, and are constantly evolving. A number of peripheral cards are certified for a real time use. Linux drivers are then modified to meet real time requirements.

5.3.5. Conclusions regarding OS. The Table 3 summarizes the features of the OSs considered to be possible candidates for hard real time systems able to support the category of enactive interactions at hand.

Table 3: Summary of the features for the most promising OS, according to the needs. N = not available. O= available. Empty = unknown. R = Root only

	Multi-processor	Access rights management	Shielding against processes	Processor affinity management	Deactivation of the scheduler	Shielding against interruptions	Process Priority management	Protection against memory swap	Synchronization means/spinning	Accepts of Recent processors	Peripheral cards support	OpenGL availability
Windows	**O**	N	N	N	N	N	N	**O**	N	**O**	**O**	**O**
MAC OS X	**O**	**O**	N	N	N	N	N	**O**	N	N	-	**O**
Linux 2.4	**O**	**O**	N	N	N	N	N	**O**	N	**O**	**O**	**O**
Linux 2.6	**O**	**O**	N	**O**	N	**O**	**O**	**O**	**O**	**O**	**O**	**O**
IRIX	**O**	**O**	**O**	**O**	N	**O**	**O**	**O**	N	N	N	**O**
RedHawk 1.3	**O**	**O**	**O**	**O**	**O**	**O**	**O**	**O**	**O**	**O**	**O**	**O**
RTLinux	**O**	**O**	N	N		**O**	**O**			**O**	-	**O**
RTAI	**O**	**R**	N	N		**O**	**O**			**O**	-	N
INTime to Windows	**O**	N	N	N		N	**O**			**O**	-	N

None of the most common « General-Purpose » OS offers today the needed features natively – though this may evolve in the future, particularly within Linux that tend to incorporate some of these features.

Real time Embedded systems may meet the technical needs regarding the hard-real time oriented features. However, despite their contemporary extension toward more user interface mechanisms, they are probably not the most adapted to meet the usages foreseen for the system, since one is seeking a range of platforms that would start from the every-day computers. Consequently, we focused firstly on other potential solutions. Further discussion of this category of OS in the context of enactive systems would be interesting – but it is not provided in this paper (see [7] as a starting point).

In the free RT OS based on a 'micro-kernel', in which the traditional OS is ran as a special low-priority task, the real time tasks do not have full access to all the resources. Additionally, the communities supporting most of these systems are often too small to guarantee a fast adaptation to the newest hardware (64 bits machines, multiprocessors machines, etc). They will not be considered furthermore in this paper.

The augmented OSs, obtained by extending General-Purpose OS, satisfy the needs, both on the real time features and on the usage side. The SGI IRIX OS is not supported on the most recent and powerful hardware. According to this overview, OS such as RedHAWK appear to be promising solutions.

6. Benchmarks

Following the theoretical overview, a series of test-bed programs were designed, covering the real-time capacities of the machine, its reactivity and regularity (determinism), and its power.

6.1. The Tested Platforms

The 10 platforms tested (table 4) were chosen amongst the most recent available. They were lent by manufacturers and assemblers: SGI, MB2I (French reseller for Intel-based solutions), Concurrent Computer Corporation (CCC) and APPLE.

Table 4. The tested platforms. The technical sheets for these platforms (size of the caches, memory buses FSB, architecture…) are available at the laboratory.

	Hardware Specifications	***OS***	
PowerChallenge reference	*SGI PowerChallenge 4 proc. MIPS R8000 275 MHz*	*IRIX 6.4*	*SGI*
FUEL	SGI Fuell. 1 proc MIPS R14000 700 MHz	IRIX 6.5	SGI
O300	SGI o300. 4 proc MIPS R14000 600 MHz,	IRIX 6.5	SGI
Apple G5	bi proc IBM G5 2GHz, Front Size Bus 1 GHz; 128 bytes	MAC OS 10.3	Apple
MBII PIV 3,2	1 proc. Intel PIV, 3,2 GHz	Linux 2.6	MB2I
MBII Bi Xeon 2,66	2 proc. Intel Xeon 2,66 GHz Intel motherboard	Linux 2.4-20 NPTL	MB2I
MBII QuadriXeon 1,8	4 proc. Intel Xeon 1,8 GHz Intel motherboard	Linux 2.6-test05	MB2I
MBII Itanium 1,1	4 proc. Intel Itanium 1,1 GHz	Linux 2.4 Redhat	MB2I
CCC Bi Xeon 1,5	2 proc. Intel Xeon 2,4 GHz DELL motherboard	Redhawk 1.4	CCC
CCC Bi Xeon 3,06	2 proc. Intel Xeon 3,06 GHz DELL motherboard	Redhawk 1.4	CCC
CCC Quadri Xeon 700	4 proc. Intel Xeon 700 MHz DELL motherboard	Redhawk 1.4	CCC

In this list, the SGI quadri-processor Power Challenge (IRIX OS), which is the last-generation platform available in our laboratory, was used as a reference for evaluating the results of the benchmark.

One can note that the platforms are not equivalent regarding the real-time oriented features offered by the OS. Most of these features were not available on the G5 under MAC OS X nor on all the platforms lent by MB2I under Linux 2.4 or 2.6. These platforms were benched to evaluate the true necessity of the RT features.

6.2. The 10 Benchmarks Designed

10 programs have been designed and run for testing the critical properties of the platform: inter-processor and memory-processor communication latency and reactivity, power of the machine, features of the system as for real-time, regularity and time-determinism, performance for realizing spinning *rendez-vous*, cache coherency management, and pure computing power.

Four of the benchmarks reflect the specificities of the mass-interaction physical modeling algorithms that are implemented in the laboratory [2]. Though they are somewhat context-dependent, they still reflect the general needs for ergotic multisensory systems. The 6 other benches offer a rougher evaluation.

Three benchmarks are mono-process. The others are multi-process. For the latest, the processes collaborate strongly with each others, exchanging various size buffers on each computation step.

For each benchmark, and for each platform, we evaluated the average power for computing the algorithm, and the time determinism of the computation over various periods of times (short-time and long-time regularity). The benchmarks were run firstly without compilation optimisation, then with the optimal optimisation we found for the compilation and, eventually, after a modification of the code itself to adapt it to the machine (size of the caches, use of the RT API offered by the OS, etc.). When no detail is given, the discussed results refer to the optimized versions, for each platform.

Details on the programs, sources, optimization details and full results are available in [6] and at the laboratory. The following analysis is a brief summary of the obtained results. We review successively (1) reactivity / latency, (2) regularity/determinism, and finally (3) power performance.

6.3. Results as for Reactivity and Latency

6.3.1. Reactivity / latency of memory access. The table 5 shows a part of the results of a bench we found to be a good representative of all the results concerning the latency of memory access.

The SGI platforms, which design is older, feature access time considerably higher than others. The positive influence of the large cache is clearly visible when dealing with large segments of memory, in the tests in which no external process requires the central memory to be updated. However, the cache benefits are not sufficient for attenuating the overall poor results.

The results for the Intel-based platforms vary quite regularly with the frequency of their memory bus. The OS used (CCC real time RedHAWK, or Linux) does not impact the results in a visible manner. On Intel Xeon / Itanium architectures, 3 stages are visible in performance measurements, according to the size for the array containing the data. Each stage corresponds with the size of the cache L1, L2 or L3.

Table 5. Average access time to the memory or read/write operations of 64 bits data. Mono-process bench, no extra load on the machine. The space is the distance in memory, in number of bytes, between two successive data accessed.

	Space 4 byte.	Space 44 byte.
SGI Power Challenge, compilation 32 bit (reference)	816,6 ns	1283,8 ns
SGI Fuel, compilation 32 bit	81,8	268,1
SGI o300, compilation 32 bit	84,9	212,3
SGI o300, compilation 64 bit	85,5	212,0
Apple G5 2 GHz	30,0	59,7
MB2I P IV Extreme ed 3.2 GHz	48,6	52,0
MB2I bi Xeon 2.66 MHz	59,8	79,7
MB2I quadri Xeon 1.8 GHz	61,9	106,1
MB2I Itanium 1.1 GHz	87,5	212,9
CCC bi Xeon 2.4 GHz	65,6	76,4
CCC quadri Xeon 700 MHz	99,4	159,0

With mono-process tests, the G5 exhibits particularly low access time, probably thanks to its two 'Load and Store' units. Excellent performances are obtained when the data are close to each other in memory. This is probably due to the large cache words of 128 bytes, which may reduce the penalty when searching for an element close to the previous.

Conversely to these promising results, the G5 performances become weaker when multiple processes make concurrent accesses to the memory. The response time largely exceed the sum of the response times obtained with mono-process tests. One can suppose that the management of cache coherency in the G5 requests the system to deal with many penalizing interrupts. Anyhow, the second worse set of results for the G5 implies that inter-process & processor communication would be difficult to achieve with a satisfactory latency (especially with spin-lock mechanisms).

To sum up, the G5 offers impressive performances for memory access, though performances decrease when dealing with multiple threads with concurrent accesses

to memory. The o300 does not present any of the qualities observed in its competitors. The Intel-based platforms have generally speaking good performances. The BiXeon 3,06 GHz proved to offer good performances in most of the benches.

6.3.2. Inter-process and processor *rendez-vous* and communication. The performances measured as for inter-process communication, and especially *rendez-vous* computed with spin-lock means, are very variable from one platform to another. They are correlated with the age of the technology, and, strongly, with the quality of real-time means offered by the system, when any. Spinning is very penalized when the task is not correctly shielded against the scheduler. For spin-lock algorithms, the results obtained are also very dependent on the distance of the various spinning buffers in memory. When making a 2 process *rendez-vous* with a spin-lock mechanism, it is usually much more efficient to use spinning buffers that are far from each other. This is probably a side effect of the mechanism ensuring cache-coherency with the memory.

As an example of the performances, the Table 6 gives the maximum frequency obtained when realizing *rendez-vous* by spinning, without any other processing. When possible, each process was running on a shielded and non-scheduled processor.

Table 6. Average frequency (in KHz) of inter-process *rendez-vous* by spinning, without extra load, according to the number of processes and the relative place of the spin buffers in memory (1 byte, or 5000 bytes).

Nb threads	Position of the buffers	Bi-Xeon 2,4	Quadri-Xeon 700	Itanium 1,1	Quadri Xeon 1,8	PIV 3,2 EE	O300	Telluris (reference)
2	0 1	1296	1363	458	2436	8324	473	217
2	0 5000	1134	1222	894	1025	5444	473	207
3	0 1 2	735	798	257	732	-	671	168
3	0 5000 9999	874	939	679	797	-	232	149

The o300 performances appear to be quite poor. The two series of tests where made on the o300, with both 32 bit and 64 bit code, which modifies the size of the memory addresses and of the whole addressable space. We found that 64 bit addressing lowers performances. Incidentally, a similar phenomenon appears with the MB2I Itanium, which also uses a 64bit memory space.

With the Pentium IV, we obtained impressive results when dealing with a two-process *rendez-vous*. This is no doubt a consequence of the hyper threading technology, which was activated on the PIV: two processes can be computed (more or less) simultaneously on a single processor, sharing the data in the caches.

The Intel-based multiprocessor Xeon or Itanium behave in a satisfying way. Though, surprisingly, the performances are not always correlated with the age of the technology – even if this age impacts the buses and memory performances as well as the processor frequency. For example, all the various Intel Xeon we tested allowed approximately 850 kHz *rendez-vous* loops when dealing with 3 processes – the quickest, in that case, appears to be the oldest machine!

The results for the *rendez-vous* on the G5 and the MB2I Xeon shows that the absence of 'real time feature' that allow forcing a process to run on a specific processor dramatically damage the performances. Indeed, real-time features are necessary in this context.

As a conclusion, the benchmarks for inter-process and inter-processor communication and *rendez-vous* appeared to be discriminating. Intel 32 bits platforms offer the maximum frequencies when making inter-process *rendez-vous*, both with and without exchanging data. One can expect improving the performances of the SGI Power Challenge reference by 5 to 9 by using these platforms. On the contrary, various platforms do not meet the needs. In particular, SGI platform would clearly not allow splitting a sound computation (~50 kHz) on various processors, as we expect to do. The 64 bits Intel-based platform we tested was also quite slow when making *rendez-vous*.

6.4. Regularity and Time Determinism

Time-determinism varies a lot from a platform to another, we found (Figure 3).

The RT systems and APIs proved to be of a major interest as for regularity and determinism. When the OS does not feature « real-time » capacity, none of the configurations was satisfying: any computation is non-regular, non time-deterministic, and, moreover, the performances rapidly decrease when the workload and/or the number of involved threads in the computation increase.

In the absence of real time feature (no shielding of processes nor processors, no process priority, etc.), the o300 and the quadri-Xeon platforms were found to be the most regular, though still non-deterministic. Conversely, The time for computing one step with G5 and the BiXeons varied considerably from step to step – and from hundred of steps to hundreds of steps: two time-scales were found in the variations.

Fortunately, the real-time features within the OS, if any, proved to be really efficient for correcting the machine instability. The most stable platforms in terms of time–determinism when using the RT API are both the SGI o300, and the CCC platforms. The CCC platforms running RedHAWK, indeed, present the best time-stability, even when the overall workload of the machine was increased.

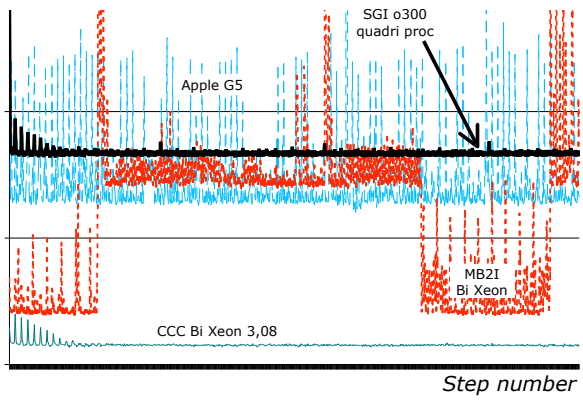

**Figure 3. Time-determinism.
Variation of processing time for various platforms, while computing a representative arithmetic algorithm on large data sets, over two synchronized threads.
No extra load (comparable results in case of extra load).**

6.5. Computation Power

Basically, one could observe a high sensitivity of the computation power to the options used for the compiler. Computing power can be increased up to twice when using a high optimisation level. In the following, we refer only to the optimised version of the benches. As a second result, most of the platforms meet, or are close to meet, the power we require: they all exhibit more than 10 times, or close to 10 times, the power of the reference SGI Power Challenge.

There exist, however, variations among the platforms (Table 7).

For the mono-process benches based on the computation of representative physical models [2], we found that, in each category of hardware, the power increases quasi-linearly with the frequency of the processor – so that we can suppose that the platform architectures do evolve in a satisfying way with the chipsets, as for the overall computing power of the machine. Conversely, the power increase for each platform seems not to depend a lot on the categories of models at hand: it is comparable whatever the basic arithmetic operation involved are (+, -, *,/ , sqrt), and whatever the size of the model is.

The G5 appears to be the most 'powerful' platform. It exhibits an average power increase of between 15 and 20, compared to the reference Power Challenge (table 7). Except the quadri-processor Xéon 700 that was lent by CCC, which could be considered as obsolete, all the Intel-based platforms also offer good performances. The more powerful Intel-based platform in the context of physical simulation with a single process was the PIV 3.2 (factor: 15). The impact of the OS real-time feature on power was found to be null (comparison of the platforms with respectively Linux and RedHAWK). The SGI configurations, as for them, were from 7 to 8 times more powerful than the reference Power Challenge. This is still close to the value of 10 we are expecting. Though, it is close to 3 times less than the G5.

Table 7. Average time reduction for computing various representative physical models, as compared to our reference. Mono-processor, mono process. Optimizationis maximized for each platform.

	Time reduction
SGI Power Challenge (reference)	0%
SGI Fuel compil. 32 bits	83%
SGI Fuel compil. 64 bits	80%
SGI o300 Compil. 32 bits	84%
Apple G5	94%
MB2I P IV 3.2 GHz	93%
MB2I bi Xeon 2.66 MHz	89%
MB2I quadri Xeon 1.8 GHz	88%
MB2I Itanium 1.1 GHz	89%
CCC bi Xeon 2.4 GHz	90%
CC quadri Xeon 700 MHz	69%

We also evaluated the dependency of computing power with the size of the data to be processed. As a result, we observe that the time for computing one step of representative basic mass-interaction physical algorithms varies per-stage with the number of instances to process in memory (ie, the size of the data), and then linearly within each stage. These stages correspond quite precisely with the size of the processor caches. Thus, the platforms that benefit from the larger caches (Itanium, SGI Fuell and SGI O300) do present a linear behaviour until approximately 20 000 basic elements (masses or interactions). For the other machines, the limit is close to 2000 modules. On these machines, for the models that feature more than 2000 modules, an increase in the size of the model is rapidly penalizing.

Other benches allowed evaluating the power of the multi-processor platforms when dealing with multiple processes (distributed computing). Basically, the global power does increase when the number of involved processes corresponds with (ie: is equal to) the number of processors. With the quadri processor platforms, distributing the computation can increase the overall power noticeably. With the MB2I quadri-Xeon (resp. with the SGI o300 quadri processor), the computation time were decreased by 19% when using 2 processes, and 42 % when using 3 processes (resp. 25% and 39%). Conversely, and not surprisingly, the number of processes involved in distributing a computation should not be more than the number of processors that are available. In that case, an important, sometimes catastrophic, loss of power is observed.

This latest result, however, does not fit with those we obtained with the mono-processors Pentium IV and the bi-processor MB2I Xeon 2,66 GHz. Both these platforms featured the Intel "hyper threading" facility. On these platform, we found, hyper-threading could

allow distributing computations in a relevant way – though, one may note that this result relates to power, not reactivity nor regularity.

7. Conclusions

The theoretical overview of hardware and OS and the bench campaign reported in this article do not pretend to be exhaustive, given the diversity of the covered fields. However, the whole study allows drawing out various conclusions on computer hardware and OS for implementing ergotic multisensory tasks.

On the hardware side, it appears that, a minima, a bi-processor platform is a promising solution: one processor for the system and non-synchronous-real-time tasks, one for the physical computations. Multi-processor (>2 processors) platforms may also be useful, though the benefit as for computing power may be lower than expected because of the necessary synchronization and exchanges over processes. More precisely, the performances obtained within the benchmark framework show that multi-processor configurations may soon allow running a single computation of a « sound physical model » (50 KHz) in a fully synchronic parallel manner. Though, such an aim is still a challenge. Finally, architectures as Intel XEON-based architectures seem to satisfying as for the equilibrium between power and reactivity.

On the OS side, it appears that none of the 'General-Purpose' OS (Windows, Mac OS X, Linux…) is today satisfactory for implementing the ergotic multisensory system at hand. Linux does propose a couple of real-time oriented features, and may become a valid solution in the future – but it is still far from being sufficient, and patching Linux is not well feasible, even for Linux experts.

Hopefully, the study shows off that, today, there are real-time oriented solutions (IRIX, RedHawk…) extending such 'General-Purpose' OS, fully compatible with them, that are really satisfactory. Thus, it is possible to plan on designing a range of platforms for ergotic multisensory tasks, in which the lowest quality corresponds with the "every-day" computer, and the highest with multi-processor platforms powered with hard real-time features – with a pragmatic continuity over the range of platforms, and without the need of a fall-back to 'embedded' RT systems.

More generally, the today's state of the art makes it possible to say that high quality enactive systems correspond with an "extreme case". On the one hand, the usages would require 'light' implementation in our every-day computer. On the other, high quality ergotic multisensory systems require a cohabitation of high density computations and high level performances as for latency, determinism, regularity, etc. It appears that today neither the hardware nor the common OS suit perfectly the requirements for implementing a high quality system. The computing power of the machines did increase clearly by a factor of more than 10 over the last 10 years. However, the same factor does not apply to reactivity and real-time capacities. This study proved that hardware architectures evolved less regarding latency and reactivity than regarding computing power. It thus confirms the analysis in [11]. Reactivity and determinism are still today bottlenecks for obtaining high quality enactive system supporting ergotic multisensory tasks; the choice of both hardware and OS are still critical.

Aknowledgments

This work was supported by the FP6 Network of Excellence IST-2002-002114 - Enactive Interfaces, and the French RNTL project 01K0719 – Cheops.

We thank the industrial partners of the study (SGI, Concurrent Computer Corporation, MB2I, IBM, Apple), and French laboratories (ID-INRIA; IRIT, IRISA…).

References

[1] Biggs, S. J., Srinivasan, M. A.,. (2001). Haptic Interfaces.

[2] Cadoz C., Luciani A. and Florens J. L.: "CORDIS-ANIMA: A Modelling and Simulation System for Sound and Image Synthesis - the General Formalism". Computer Music Journal 17(4), 1993.

[3] Cadoz C. (1994). Le geste, canal de communication instrumental. techniques et sciences informatiques. Vol 13 - n01/1994, pages 31 à 61.

[4] Claude Cadoz, Marcello M. Wanderley (2000). Gesture-Music, in Trends in Gestural Control of Music, M. M. Wanderley and M. Battier, eds, ©2000, Ircam – Centre Pompidou, pp. 71-94

[5] Castagne N, Cadoz C, Florens JL, Luciani A: "Haptics in Computer Music: a Paradigm Shift" Proc. of Eurohaptics Munich, 2004, pp422-425.

[6] Castagne N, Florens JL, Kergadallan G: "Computer hardware and OS for high quality interactive and multi-sensory simulation of physical objects" – ACROE-ICA internal report – available on http://acroe.imag.fr/opendocs/HardwareOS_2005.htm

[7] Dedicated System WebSite - Evaluations of various Real Time Operating Systems. On the Internet: http://www.omimo.be/encyc

[8] Florens JL, Cadoz C, and Luciani A, ``A real-time workstation for physical model of multi-sensorial and gesturally controlled instrument,'' in Proc. Int. Computer Music Conf. (ICMC'98), pp. 518-525

[9] IEEE Technical Committee on Real-Time Systems - Available on the Internet: http://cs-www.bu.edu/pub/ieee-rts/Home.html

[10] Luciani A, Florens JLF, Castagne N : "From Action to Sound: a Challenging Perspective for Haptics" – proceedings of the Eurohaptics/WorldHaptics conference, Pisa, 2005.

[11] Patterson, D. A. (2004). Latency lags bandwidth. Communications of the ACM, 47(10):71–75.