Towards Efficient Parallel Radiosity for DSM-based Parallel Computers Using Virtual Interfaces

Luc Renambot, Bruno Arnaldi, Thierry Priol, Xavier Pueyo

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Towards Efficient Parallel Radiosity for DSM-based Parallel Computers Using Virtual Interfaces

Luc RENAMBOT , Bruno ARNALDI, Thierry PRIOL,

Xavier PUEYO

N ° 3245

Septembre 1997

THÈMES 1 et 3

ISSN 0249-6399

Towards Efficient Parallel Radiosity for DSM-based Parallel Computers Using Virtual Interfaces

Luc RENAMBOT *, Bruno ARNALDI*, Thierry PRIOL*,
Xavier PUEYO†

Thèmes 1 et 3 — Réseaux et systèmes — Interaction homme-machine, images, données, connaissances
Projet Caps et Siames

Rapport de recherche n ° 3245 — Septembre 1997 — 19 pages

Abstract: This paper presents the performance evaluation of a new technique for radiosity computation which aims at exploiting efficiently the different levels of a memory hierarchy of both sequential and parallel computers. Such ability is essential when dealing with complex environments having several millions of polygons. The principle of our technique is to split the initial environment into several sub-environments and compute the radiosity within each sub-environment. Exchange of energy between sub-environments is performed by means of virtual interfaces and visibility masks. The size of sub-environments can be adapted in order to fit into a cache or a local memory. We performed several experiments using an SGI Origin 2000 to show the effectiveness of our solution. It improves both the sequential and parallel execution of a progressive radiosity algorithm. Our technique decreases the execution time on one processor of an SGI Origin 2000 by a factor of more than 5 and leads to a very good efficiency for complex environments (1 million of polygons) on a multiprocessor configuration.

Key-words: radiosity, complex databases, parallelism, data locality, distributed-shared memory, SGI Origin 2000.

(Résumé : tsvp)

* {renambot,arnaldi,priol,}@irisa.fr
† Dept. Informàtica i Matemàtica Aplicada, Universitat de Girona, Llus Santal, s/n E-17071 Girona, Spain, xavier@ima.udg.es

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 02 99 84 71 00 – Télécopie : (33) 02 99 84 71 71

Vers une méthode de radiosité parallèle efficace sur une machine parallèle de type DSM en utilisant le concept d'*interfaces virtuelles*

Résumé : Cet article présente l'évaluation des performances d'une nouvelle technique de calcul de radiosité dont le but est d'exploiter efficacement les différents niveaux de la hiérarchie mémoire des ordinateurs séquentiels ou parallèles. Cette propriété est essentielle pour pouvoir traiter des scènes très complexes de plusieurs millions de polygones. Les principes de notre technique sont de générer une partition de la scène initiale en sous-environnements et d'appliquer un calcul de radiosité sur chacun de ces sous-environnements. L'échange d'énergie entre les sous-environnements est exprimé par les concepts d'interface virtuelle et de masques de visibilité. La taille des sous-environnements peut être adaptée en fonction des caches ou de la mémoire locale. Nous avons mené différentes expérimentations sur une machine SGI Origin 2000 pour démontrer l'efficacité de notre solution. Elle améliore l'exécution séquentielle et parallèle de l'algorithme de radiosité progressive. Notre technique diminue le temps d'exécution sur un processeur de la machine SGI Origin 2000 d'un facteur supérieur à 5 et permet d'obtenir une très bonne efficacité pour des environnements complexes (1 million de polygones) sur une configuration multi-processeurs.

Mots-clé : radiosité, scènes complexes, parallélisme, localité des données, mémoire partagée physiquement disribuée, SGI Origin 2000.

1 Introduction

Among the different techniques to render high quality images, the radiosity method has become very popular. It computes the most important forms of illumination, the indirect ambient illumination provided by inter-reflections of lights between diffuse objects. Like ray-tracing, radiosity algorithms require large computing resources, both in terms of computing power and memory storage. Since the introduction of the radiosity method, a lot of research has dealt with the design of new techniques in order to reduce computations. However, despite the several improvements, modern workstations cannot provide the required computing resources when dealing with large complex environments (i.e. an order of several millions of polygons).

With the decay of vector supercomputers, the use of parallel computers seems to be the only way left to achieve the required level of performance for radiosity computations. After ten years of constant evolution, parallel computers have reached the maturity which is requisite to their acceptance by the computer graphics industry. Two new architectural trends stand out nowadays: cluster of workstations (IBM SP/2) and Distributed Shared Memory (DSM) parallel computers (SGI Origin). Both approaches are distributed memory parallel computers (i.e. each node has its own local memory). However, these two kinds of parallel architectures have their own programming models. While clusters of workstations have to be programmed using a message passing paradigm, DSM parallel computers provide shared variables as a communication paradigm. Even though, programming these two types of parallel computers requires roughly the same methodology when designing an efficient parallel algorithm. It is mainly a two-dimensional optimization process. The first dimension is the identification of parallelism in the original algorithm in order to split it in several independent tasks. The second dimension is data distribution in order to map data onto local memories. Nevertheless, a high degree of parallelism, in a given algorithm, does not ensure every time a good performance. Indeed, exploitation of parallelism may require a large amount of communication between processors. In that case, such parallel algorithm is said to have a poor data locality property.

Such statement can be applied to the radiosity. The computation of inter-reflections between objects involves several tasks that can be run in parallel. For progressive radiosity, several levels of parallelism have been identified [3]. However, this parallelism is obtained at the expense of a loss of data locality, because each processor may have to access any part of the environment. When dealing with large environments (typically several millions of polygons), parallel radiosity algorithms offer poor performance. Lack of data locality has also an impact on the performance of sequential radiosity algorithms due to the existing multi-level memory hierarchy available in modern workstations. Efficient exploitation of the memory hierarchy is crucial to get high performances. However, most studies [17] focus on the exploitation of the these last levels of the memory hierarchy, main memory and disks for the management of virtual memory.

Our research addresses the problem of using efficiently the different levels of a memory hierarchy for both uniprocessor and multiprocessors computers. Our approach consists in designing a new radiosity algorithm which has a good data locality property. Locality

is exploited to speedup up the computation on a single processor as well as on parallel computers. Such technique has been already successfully applied to message-passing parallel computers [1]. In this paper, we show the impact of our original technique to DSM parallel computers, in particular the SGI Origin 2000 [9].

The paper is organized as follows. Related work is discussed in section 2. Section 3 provides a brief description of the DSM concept. Section 4 presents an overview of our technique. In section 5, a detailed description of the parallel algorithm is given. In section 6, we focus on the experimental platform we used. Section 7 contains the experimental results which were obtained on an SGI Origin 2000. Finally, Section 8 draws some conclusions.

2 Related works

We mainly analyze previous works from two points of view: parallel solutions and locality enhancement.

A number of parallel solutions have been proposed using different architectures and different types of parallelizing strategies. [3, 7] present approaches for parallelizing radiosity at several levels. From the scope of the work presented here, we may distinguish between two kinds of solution. A first family of algorithms allows the access of all the processors to the entire database. This may be implemented on shared memory systems or on distributed memory systems where the database is duplicated at each local memory [10, 11] which limits the scene's size. The second family of algorithms is constituted by the techniques based on distributed memory systems where the database is distributed among the different local memories allowing the application of the algorithm to render bigger scenes [4, 6, 16, 19].

The idea of considering sub-environments (or local environments) has been used from different points of view, in order to reduce computational cost in radiosity algorithms for complex environments. The first contribution in this direction was proposed in [20]. This method divides environment into local environments and computes form-factors in each of them. Afterwards, form-factors between patches in neighboring local environments are computed from previously computed local form-factors. Similar approaches were presented in [2, 18]. Energy is accumulated at the boundaries of neighboring local environments (called virtual walls) and afterwards transfered.

Groups of surfaces are used in order to simplify energy exchange in complex environments by means of simplifying the geometry of the environment in a 3-D grouping approach in [8, 13, 14, 15]. Clustering is also used in [12] where the environment is mapped onto the walls of the cluster. This map is used only for secondary rays while primary rays do not use the clusters. The interest of parallelizing the algorithm is pointed out in [12], using the locality resulting from the mapping but not exploited during the rendering step. All these methods introduce approximations for exploiting locality but bound their effects in order to keep the required accuracy.

In [17], a technique is proposed to exploit locality in order to minimize the transfers between disk and main memory. This system is designed to compute radiosity efficiently in very large environments where the visibility relationships are locally dense but globally

sparse. It is based on a hierarchical approach. Two basic ideas are exploited: partitioning and ordering. Partitioning means identifying subsets of the database each composed by the clusters which interact with a given (receiver) cluster. Ordering implies finding an order which minimizes the difference between two successive subsets.

From this analysis, it appears that few prior works have dealt with the use of all the levels of the memory hierarchy of a modern computer. This fact led us to propose a new technique with a better data locality to fully exploit the memory hierarchy.

3 Distributed Shared Memory

Distributed Shared Memory is a way to provide a logical global address space which is distributed among the local memories of a distributed memory parallel computer. Distribution of the global address space is hidden to the user by either a software or a hardware layer. In both approaches, efficiency relies mainly on caching techniques when accessing data in a remote memory location. Research and development projects faced a serious challenge to implement a hardware cache-coherent DSM. Two main approaches have been investigated.

The first approach is the COMA (Cache-Only Memory Architecture) model which was illustrated by the Kendall Square Research's KSR-1 machine. In this approach, the local memory acts as a large cache (attraction memory) to store sub-pages that contain the data that has been requested. One benefit of such approach is that the data (i.e. sub-pages) will migrate to where it will be used. There is no fixed physical location for a particular data. Therefore, initial data placement has little impact on performance. Unfortunately, since the KSR commercial failure, there are no more COMA machines available.

The second approach is the CC-NUMA (Cache-Coherent Non Uniform Memory Access) model. This approach has been adopted by several companies such as Convex/HP (Exemplar), Data General and Sequent (SCI based architectures) and more recently by SGI (Origin 2000). In such an approach, the shared memory is distributed using information in the physical addresses of memory blocks. A virtual address space can thus be distributed using the virtual to physical address translation mechanism of the processor (i.e. a particular page of the virtual address space can be mapped onto a page frame located in a memory of a given processor using the MMU page tables). However, performance of CC-NUMA machines is very sensitive to the memory placement policy. It is usually the responsibility of the operating system to migrate or to replicate pages in order to decrease the number of remote memory accesses. However, since migration and replication policies are based on heuristics, they are sometimes not adapted to users' applications. In that case, to get efficiency, users have to design their parallel algorithms with an explicit data domain decomposition. This work is similar to the one that is performed when dealing with message-passing distributed memory parallel computers. However, since data remains in the DSM, load balancing pro-

blems can be solved easily by migrating data and computations to processors which are not overloaded. Such data migration is achieved transparently through the DSM.

4 Virtual Interface

This section summarizes briefly the virtual interface concept. A more detailed description can be found in [1].

4.1 Principles

The first idea addresses the energy transfer between local environments. Instead of accumulating the energy from different sources onto a *virtual wall* before sending it to another environment, the *virtual interface* technique allows the management of a source separately from other sources in the same local environment. As soon as a source has emitted its energy, it is sent to the next environment.

The second principle of the virtual interface describes how source transfer between local environments is taken into account. Sending the geometry and its emissivity to the next environment after a local processing implies the addition of a new structure, called the visibility mask, to the source (figure 1). The visibility mask stores in the source structure all the occlusions encountered during the processing in each local environment.

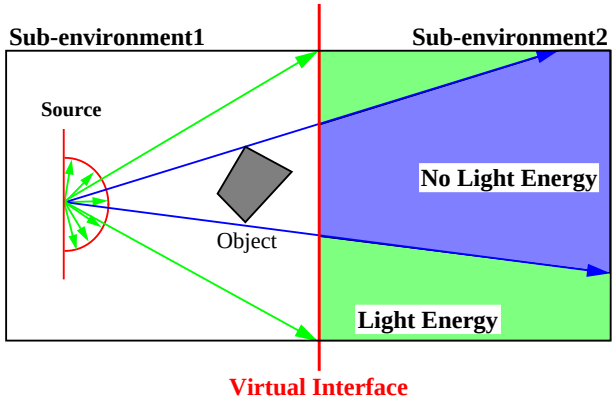


Figure 1: Virtual Interface (former Virtual Wall).

With our virtual interface concept, the energy of each selected patch, called a source, is first distributed in its local environment. Then, its energy is propagated to other local environments. However, to propagate efficiently the energy of a given patch to another local environment, it is necessary to determine the visibility of the patch according to the current local environment: an object along the given direction may hide the source (figure 2). We introduced the visibility mask which is a subsampled hemisphere identical to the one involved

in the computation of the form factors. To each pixel of the hemisphere, used for form factor computation, corresponds a boolean value in the visibility mask. The visibility mask allows the distribution of energy to local environments in a step by step basis. If the source belongs to the local environment, a visibility mask is created, otherwise the visibility mask already exists and will be updated during the processing of the source. Form factors are computed with the patches belonging to the local environment by casting rays from the center of the source through the hemisphere. If a ray hits an object in the local environment, the corresponding value in the visibility mask is set to false otherwise there is no modification. Afterwards, radiosities of local patches are updated during an iteration of the progressive radiosity algorithm. Finally, the source and its visibility mask are sent to the neighboring local environments to be processed later.

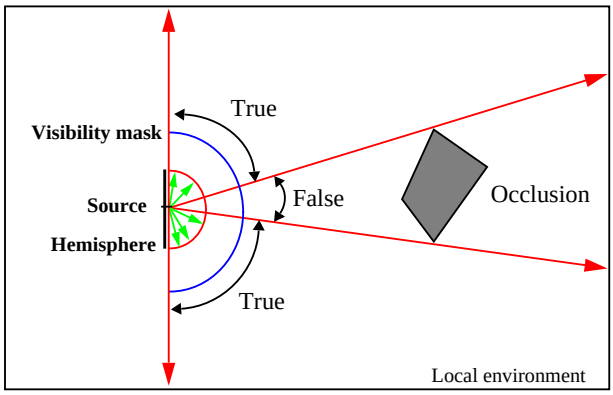


Figure 2: Initialization of the visibility mask for a local source.

4.2 Discussion

Using virtual interfaces and visibility masks, to solve the radiosity computations, may raise the question about their impact on quality degradation. We think that our method does not degrade the results since we ensure that all sources have been processed, in a correct manner, equivalent to the classical algorithm. We discuss this point in section 5. Moreover, visibility masks represents exactly the hemisphere with its resolution. Sampling and aliasing problems inherent to hemisphere form factors computation methods are not in the scope of this paper. However, our technique entails new problems such as the placement of virtual interfaces which may influence the performance. It exhibits data locality depending of the size of working sets associated to a sub-environment. Finding an appropriate placement may limit the overhead of the management of the visibility masks (for example, a virtual interface along a wall can stop light propagation).

5 Algorithms

Using virtual interface and visibility mask concepts described in the previous section, we design both a sequential and a parallel radiosity algorithm which exhibit a data locality property.

5.1 Sequential version

The main idea of this algorithm is to apply a radiosity computation on each sub-environment generated by the projection of virtual interfaces on a given scene. In our case, we implement a classical radiosity method using a progressive approach. The computation of form factors is done through a hemisphere using an accelerated ray-tracing algorithm by means of a regular 3D grid. The main difference is that we manage, for each sub-environment, a list of visibility masks which correspond to non-local sources. Therefore, the selection of the most emissive patch takes into account this list. The patch to be shot at a given iteration could be either a local source or a remote source represented as a visibility mask already processed by other sub-environments. The visibility mask of the current source is filled during the form factors computation. Once the source shoots its energy, the visibility mask is copied to neighboring sub-environment sources lists, if required. All sub-environments are processed one after another to obtain a given convergence level. The criteria is to shoot a patch if its energy is above a threshold (a fraction of the overall most emissive patch in the scene). The termination test ensures that all visibility masks are processed to keep coherent radiosities, so that lists of visibility masks of all environments are empty.

5.2 Parallel version

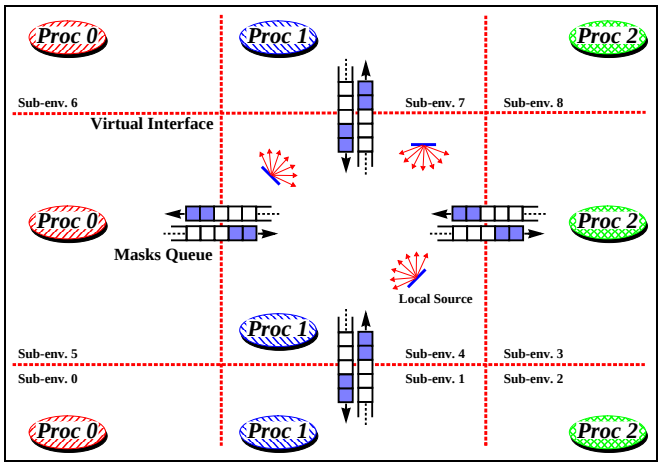


Figure 3: Environment partitioning and mapping

Our parallelization scheme uses a domain decomposition technique based on the virtual interface to generate independent computations and to distribute data. Next, we have to distribute these sub-environments onto a set of processors. The communication between distinct sub-environments is carried out by the sending of visibility masks. In a previous study [1], we show that this could be expressed trough message-passing between processors.

Processor management Each processor has to manage a set of sub-environments (one or several), and access visibility masks generated by other processors. Figure 3 shows an example of such a construction where we have nine sub-environments to distribute among three processors.

Memory management There are two ways of managing memory allocation for radiosity computations. The first one is to let the operating system allocates memory in the global shared space using default policies. The other is to organize memory, using different regions for each sub-environment, knowing that each region is accessed by only one processor. Furthermore, it can be better to allocate each region in the local memory of the processor that is responsible for it. Once this context is created, different strategies can be applied to map regions onto the set of CPUs. One may use page migration (either by explicit calls or by operating system policies) or process allocation and migration.

Termination detection The termination detection is performed by managing a shared global array holding the state of each process. It could be either *not locally converged* (currently processing a local source or a visibility mask), *locally converged* (list of visibility mask empty and no patch to shoot) or *globally converged*. This last state is reached if all sub-environments are in the state locally converged and no visibility mask is being transfered.

Synchronization There is very little synchronization in our algorithm. Each processor can perform its computations asynchronously. However, in a shared memory context, some *lock* variables are necessary to ensure safe accesses to shared structures like the visibility mask queues and global state variables.

Load-balancing In a coarse grain parallel program, it is critical to focus our attention on load-balancing to achieve good performances. In our context, we have to find a trade-off between data locality expressed in each sub-environment and the corresponding workload. In extreme cases, we could have a small environment with high energy flow and a large environment without any light source. For load balancing, our first solution is to make a cyclic distribution of all sub-environments. This could be a good heuristic with a regular scene structure and a uniform light distribution. But in most realistic cases, it would not work. Therefore a second approach is to use information given by an execution to generate a new distribution of the sub-environments. The criteria may be the number of rays handled during form factors computations.

6 Experimental platforms

The implementation of our parallel algorithm was carried out on a *Silicon Graphics Origin 2000* [9]. This is a CC-NUMA architecture as introduced in section 3. In this section, we describe briefly this machine and the implementation of our parallel algorithm.

6.1 SGI Origin 2000

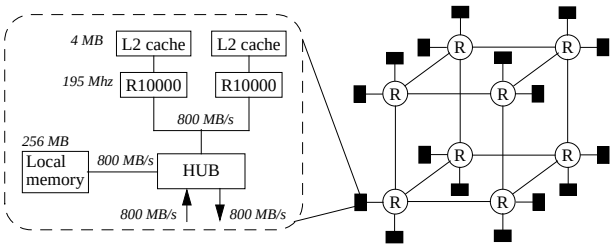


Figure 4: Origin2000 32 processors System.

Our algorithm was implemented on a 32 processors configuration organized in 16 nodes, each one of them having two *R10000* CPUs with 4Mbytes of second level cache (L2), and 256 Mbytes of local memory. The total physical memory available is 4 Gbytes. The interconnection network is a hypercube as shown in figure 4. An interesting feature of the *R10000* is its set of two performance counters. They allow performance analysis of the user's application execution with very low interferences and at program full speed [21]. Each counter can track one event among sixteen available. Some of these events are associated with the memory hierarchy such as the number of misses for the L1 and L2 caches and for the TLB. A software tool, named *perfex*, uses these counters to provide a performance analysis. It computes various information such as the estimated time per event and other statistics (Mflops, instructions per cycle, caches hit rates, ...).

6.2 Implementation issues

Two versions of our parallel algorithm were implemented. The first one is the parallelization of our sequential algorithm without data management. Data is allocated processor by processor calling *malloc* function. In a second version, we manage data placement. Each processor uses a distinct segment of shared memory for each sub-environment it own. The working set (geometries, radiosities) corresponding to one region is allocated in such a segment and thus accessed only by one processor. This can be done with the *arena* mechanism provided by the *Irix* operating system. Each arena is linked to a thread (i.e. to a processor) but a thread can manage several arenas. A file holding placement specification is used when the program is launched by the *dplace* tool, or loaded during the execution for dynamic specifications. A valid placement specification contains descriptions of memory allocation and

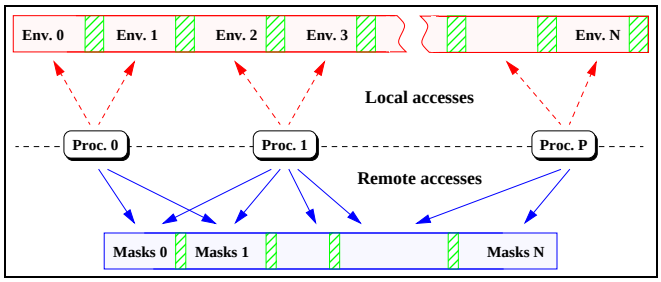


Figure 5: Data distribution and memory accesses

thread placements. Memory topologies as well as affinities to devices may also be specified. Shared-virtual-address ranges can be mapped to a specified memory. Page size and page migration thresholds can also be controlled. These features were used in the "optimized" version to perform a better placement. The visibility mask buffers are allocated in shared memory because they represent communications between sub-environments and are shared among several processors, as shown in figure 5. *Lock* variables control accesses to these buffers. Buffers and lock variables are allocated on secondary cache line boundaries (128 bytes) using *memalign* function calls and *padding* to reduce false sharing.

7 Experiments

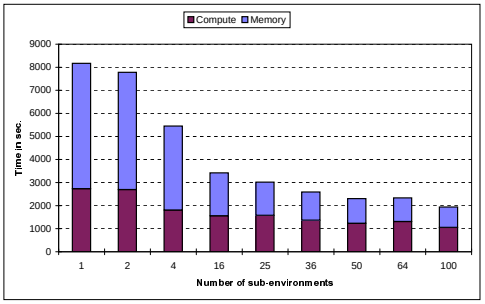
The main idea that we want to show is that radiosity computations on realistic scenes is memory bound, contrarily to studies on small scenes evoked in section 2. In this section, we conduct experiments showing that our method makes a better use of the memory hierarchy in both sequential and parallel execution.

7.1 Experimental protocol

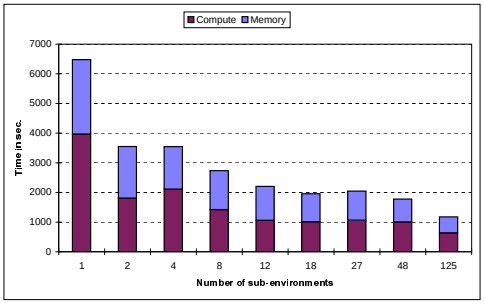
We conduct several experiments using two large databases on the Origin 2000 configuration described in section 6.1. The first database is made of around 400.000 polygons and the second has more than 1.000.000 polygons, before the meshing process. For each database, several sub-environment decompositions were generated and studied with the sequential and parallel versions of our algorithm to show the influence of the scene structure, the decomposition into sub-environments and the number of processors on performance.

7.1.1 Metrics

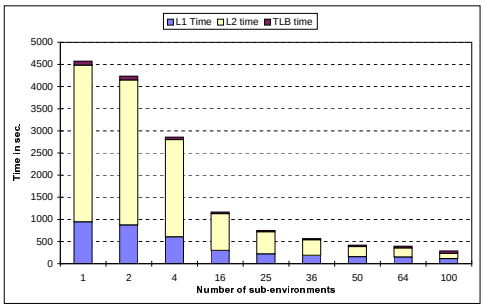
To analyze our method, we use a set of metrics that describe the behavior of each level of the memory hierarchy and the processor speed. The use of the hardware performance counters available on the *Mips R10000* processor greatly simplifies this task.



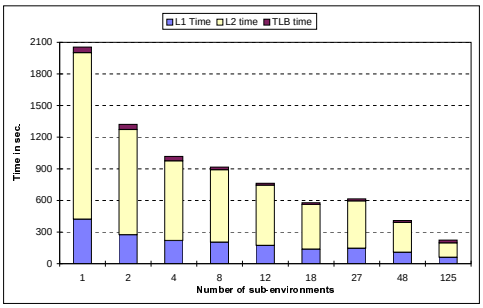
(a) Execution times, *Parking* scene



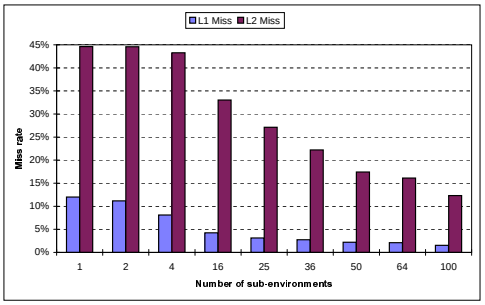
(b) Execution times, *Csb* scene



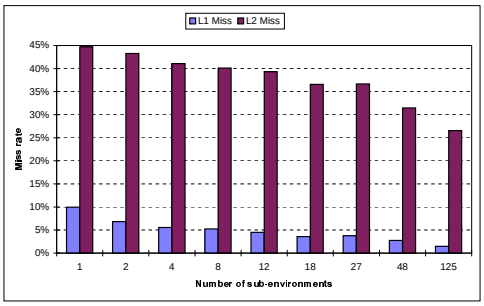
(c) Memory hierarchy, *Parking* scene



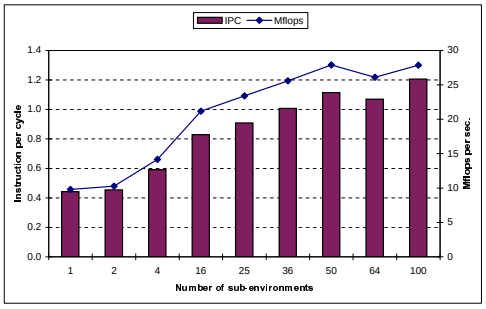
(d) Memory hierarchy, *Csb* scene



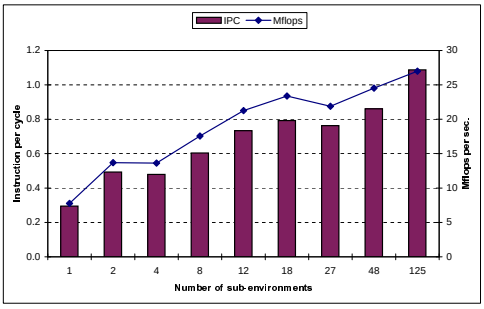
(e) Cache miss rates, *Parking* scene



(f) Cache miss rates, *Csb* scene



(g) Instruction per cycle and Mflops, *Parking* scene



(h) Instruction per cycle and Mflops, *Csb* scene

Figure 6: Sequential results

L1 cache misses The first level data cache is on chip and has a latency of 2 to 3 cycles. It is a 32 Kbytes two ways set associative cache with 32 bytes per line.

L2 cache misses The second level cache is external and holds 4 Mbytes of data and instructions. It is relatively large compared to current workstations. The miss latency is about 10 cycles compared with 61 cycles for the local memory and around 160 cycles for a non-local memory.

TLB misses The *Translation Look-aside Buffer* is a cache holding the translation of virtual addresses into physical ones of recently accessed pages. A miss in the *TLB* is a costly process, around 70 cycles latency and could lead to L2 misses. A program with an important working set and without data locality can generate a high TLB miss rate and cause a lot of operating system activity.

Number of instructions per cycle Modern superscalar out-of-order execution processors like the *Mips R10000* can overlap data requests which are non blocking during a data miss. The *R10000* can process up to four instructions per cycle. Typical values are between 0.5 and 2 because of limited instruction level parallelism and data access latency.

Memory overhead With the previous metrics and the number of load and store operations, one can compute the overall time spent in the memory hierarchy.

MFLOPS It shows the achieved performance of an application. The *R10000* is rated at 390 Mflops (Million of floating point operations per second) peak.

Speed-up The definition of the speed-up is $S = T_1/T_n$, with T_1 the *best* sequential time and T_n the time obtained in parallel using n processors.

7.1.2 Scenes

We chose two scenes. The first, named *Csb*, represents the Soda Hall Building. The five floors are made of many furnished rooms, resulting in a scene of over 400.000 polygons. It's an occluded scene. The second scene, named *Parking*, represents an underground car park with accurate cars models. The scene is over 1.000.000 polygons. It is a regular and open scene. Table 1 lists some characteristics of these databases.

We use a straightforward decomposition algorithm which places virtual interfaces evenly along each axis, producing cubic sub-environments. In section 7.2, we studied several decompositions for each scene from 1 to 100 sub-environments for the *Parking* scene (from $1 \times 1 \times 1$ to $10 \times 10 \times 1$ in a $x \times y \times z$ 2D-decomposition), and from 1 to 125 sub-environments for the *Csb* scene (from $1 \times 1 \times 1$ to $5 \times 5 \times 5$ in a $x \times y \times z$ 3D-decomposition). In section 7.3, we experimented two decompositions for each scene, giving one or three regions for each

Scene	Polygons x 1000	Patches x 1000	Sources	Shots	Memory in Mb
Csb	430.1	1202.3	342	960	495
Parking	1080.2	1288.2	140	1804	525

Table 1: Scenes

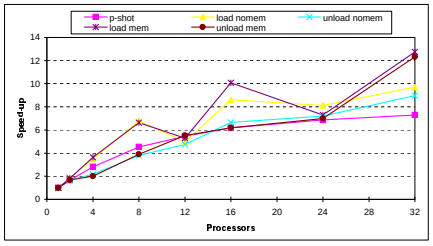
processor using a cyclic distribution ($4 \times 8 \times 1$ and $12 \times 8 \times 1$ for *Parking* scene, and $4 \times 8 \times 1$ and $6 \times 4 \times 4$ for *Csb* scene).

7.2 Sequential Results

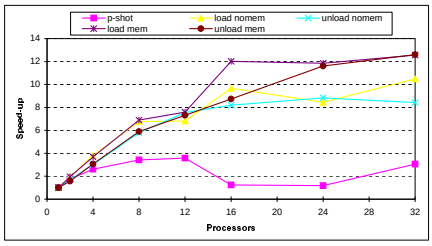
In figures 6(a) and 6(b), we show sequential execution times for each decomposition, divided into computation time and memory overhead (data misses and load/store operations). Higher decompositions do not give important gains. For example, a gain factor of 4.2 can be achieved for the *Csb* scene with 100 sub-environments, and 5.5 for the *Parking* scene. The main gain is given by a reduction of memory overhead. This is illustrated in figures 6(c) and 6(d) showing misses in the memory hierarchy (data caches, TLB). We can see a dramatic reduction of secondary data cache access time up to a factor of 30 for the *Parking* scene and a factor of 11 for the *Csb* scene. With the reduction of the working set, we enhance data locality and make a better use of the L2 cache. One can notice that, even if TLB time decreases for a small number of sub-environments, it increases with higher decomposition showing that we introduce some overhead due to sub-environments management. The outcome of better data caches management is a clear improvement of miss rates as shown in figures 6(e) and 6(f). Data locality reduces memory latency and allows the processor to issue more instructions per cycle, which is a great challenge on a superscalar processor. This is clearly stated in figures 6(g) and 6(h) where the processor can only process 0.4 instruction per cycle for the *Parking* scene with one sub-environment and achieve 1.2 instructions per cycle in a 100 sub-environments decomposition. The overall performance goes from 10 *Mflops* to 28 *Mflops*. We obtain similar results for the *Csb* scene.

7.3 Parallel Results

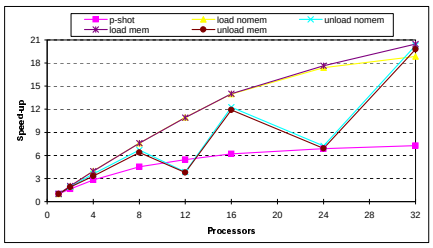
To compare our method to existing ones, we implement a classical parallelization of the radiosity algorithm where a set of processors shoot different patches in parallel [3]. The address space is unique and all the processors may access the entire database. Each processor selects a patch and shoots it onto the scene (i.e. computes form factors between the current patch and the entire database and then updates the radiosities which are stored into shared variables). Critical sections were used to ensure the coherence of radiosities values between selection and update phases. The experiments are named following these rules in figure 7 : *p-shot* for classical parallelization of radiosity, *load* or *unload* for our parallel versions with



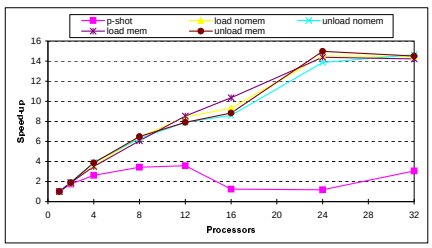
(a) Speed-up, *Parking* scene, 32 env.



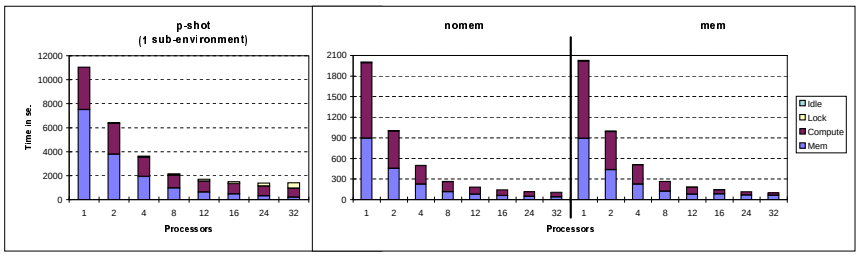
(b) Speed-up, *Csb* scene, 32 env.



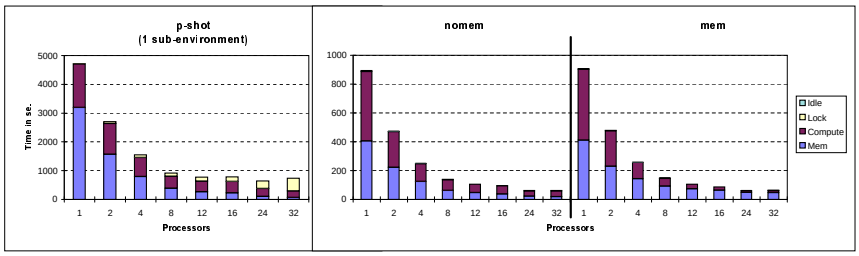
(c) Speed-up, *Parking* scene, 96 env.



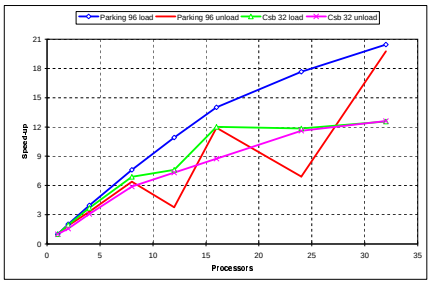
(d) Speed-up, *Csb* scene, 96 env.



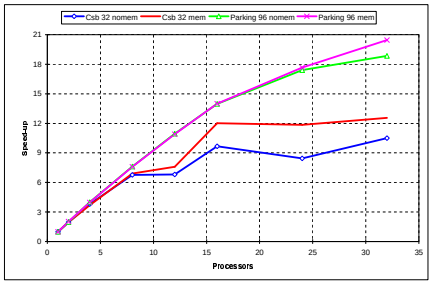
(e) Times, *Parking* scene, 96 env.



(f) Times, *Csb* scene, 96 env.



(g) Load-balancing effects, *Parking & Csb* scenes



(h) Memory managment effects, *Parking & Csb* scenes

Figure 7: Parallel results

or without static load balancing, *32* or *96* for the number of sub-environments and finally *mem* or *nomem* according to data management scheme. Figures 7(a), 7(c), 7(b), 7(d) show speed-up results for the *Parking* scene and the *Csb* scene, with 32 or 96 sub-environments.

The first comment is that the *p-shoot* method is always the least effective approach, having a very high memory overhead (figure 7(e)) and high *lock* times when the number of processors raises. It achieves a 7.3 speed-up on 32 processors on the *Parking* scene (figure 7(a)). We obtain the same behavior on the *Csb* scene up to 12 processors, then *lock* time becomes very large (figure 7(f)). The probability that processors interlock during computation on the *Csb* scene is higher than on the *Parking* scene because the *Csb* scene contains twice less polygons. With our method *lock* times are very small in all configurations. But, due to domain decomposition, our method is sensible to load-unbalance and produces less regular results. However, using a simple static load-balance strategy we obtain a speed-up of 21 on 32 processors for the *Parking* scene (figure 7(c)).

We can see comparing results for 32 and 96 sub-environments (figures 7(a),7(b),7(c),7(d)) that load-balancing is a key issue to achieve a good performance. This is clearly shown by figure 7(g) where we compare results with and without load-balancing. With load-balancing, we always obtain better results, and the more sub-environments we have the more we can distribute evenly the load.

Figure 7(h) exposes another behavior, the impact of data allocation in the shared space as the number of processors grows. The impact on speedup depends on the number of processors. The reason is due to the increasing number of non local remote memory accesses when the data management is left to the operating system.

8 Conclusion and further works

As we have shown in this paper, *Virtual Interface* and *Visibility Mask* are two efficient techniques to enhance data locality for radiosity computations. Such an enhancement is of considerable importance when rendering complex environments since a large amount of time is spent accessing the memory. Memory access times is and will remain the main bottleneck since performance of micro-processors increases at a higher rate than the memory access times decrease. By using our technique, we were able to decrease the execution times by a factor of more than five. This factor may vary with scene complexity (in term of the number of polygons). As regards to the results of the parallel version, our approach offers better performance comparing to a more traditional approach. With the two scenes, we were able to get a 50% of efficiency using 32 processors whereas a traditional approach offers no more than 25% of efficiency. Even so, further works needs to be performed before having a robust parallel solution. More experiments have to be performed to confirm these results. The lack of benchmarking scenes, which are widely accepted by the computer graphics industry, prevents us from presenting a larger amount of experimental results and from comparing our results with other already investigated approaches. Next, we would like to address one of the main problem we have encountered : the positioning of virtual interfaces and their influences on load-balancing methods. These techniques could be static and dynamic, using

new mapping of the sub-environments on processors, or even new placements for virtual interfaces. One solution to be investigated is the use of graph partitioning techniques as proposed in [5]. For both techniques, a trade-off between load-balancing and data locality would have to be found. Our study focus on the traditional and well know progressive radiosity method, which has been replaced nowadays by hierarchical methods. Theses methods seem to be even more memory consuming than previous ones. Thus, we think that the main concepts of our method (spatial decomposition and compact representation of light transfer) could be applied to hierarchical and clustering radiosity, using several levels of interaction with the visibility mask. This issue forms the subject of a future research work.

Acknowledgements

The authors thank the Centre Charles Hermite (`http://www.loria.fr/CCH`) which has provided the computing resource for the experiments provided in this paper, Dan TRUONG, Pierre MICHAUD and Mounir HAHAD for their comments.

References

[1] Bruno Arnaldi, Thierry Priol, Luc Renambot, and Xavier Pueyo. Visibility Masks for Solving Complex Radiosity Computations on Multiprocessors. In *Proc. First Eurographics Workshop on Parallel Graphics and Visualisation*, pages 219–232, Bristol, UK, September 1996.

[2] Bruno Arnaldi, Xavier Pueyo, and Josep Vilaplana. On the Division of Environments by Virtual Walls for Radiosity Computation. In *Proc. of the Second Eurographics Workshop on Rendering*, pages 198–205, Barcelona, 1991. Springer-Verlag.

[3] K. Bouatouch and T. Priol. Data Management Scheme for Parallel Radiosity. *Computer-Aided Design*, 26(12):876–882, December 1994.

[4] Martin Feda and Werner Purgathofer. Progressive Refinement Radiosity on a Transputer Network. In *Proc. of the Second Eurographics Workshop on Rendering*, pages 139–148, Barcelona, 1991. Springer-Verlag.

[5] Thomas A. Funkhouser. Coarse-Grained Parallelism for Hierarchical Radiosity Using Group Iterative Methods. In *ACM SIGGRAPH '96 Proceedings*, pages 343–352, New Orleans, August 1996.

[6] P. Guitton, J. Roman, and Christophe Schlick. Two Parallel Approaches for a Progressive Radiosity. In *Proc. of the Second Eurographics Workshop on Rendering*, pages 160–170, Barcelona, 1991. Springer-Verlag.

[7] Frederik W. Jansen and Alan Chalmers. Realism in Real Time? In *Proc. of the Fourth Eurographics Workshop on Rendering*, pages 27–46, Paris, June 1993.

[8] Arjan J. F. Kok. Grouping of Patches in Progressive Radiosity. In *Proc. of the Fourth Eurographics Workshop on Rendering*, pages 221–232, Paris, France, June 1993.

[9] James Laudon and Daniel Lenoski. The SGI Origin 2000: A CC-NUMA Highly Scalable Server. In *Proceedings of the 24th Annual International Symposium on Computer Architecure*, pages 241–251, Denver, June 1997. ACM Press.

[10] Claude Puech, Francois Sillion, and Christophe Vedel. Improving Interaction with Radiosity-based Lighting Simulation Programs. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, volume 24, pages 51–57, March 1990.

[11] Rodney J. Recker, David W. George, and Donald P. Greenberg. Acceleration Techniques for Progressive Refinement Radiosity. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, volume 24, pages 59–66, March 1990.

[12] E. Reinhard, L.U. Tijssen, and F.W. Jansen. Environment Mapping for Efficient Sampling of the Diffuse Interreflection. In *Proc. of Fifth Eurographics Workshop on Rendering*, Darmstadt, 1994.

[13] Holly E. Rushmeier, Charles Patterson, and Aravindan Veerasamy. Geometric Simplification for Indirect Illumination Calculations. In *Proc. of Graphics Interface '93*, pages 227–236, San Francisco, May 1993.

[14] Francois Sillion and George Drettakis. Feature-Based Control of Visibility Error: A Multiresolution Clustering Algorithm for Global Illumination. In *ACM SIGGRAPH '95 Proceedings*, pages 145–152, Los Angeles, 1995.

[15] Brian Smits, James Arvo, and Donald Greenberg. A Clustering Algorithm for Radiosity in Complex Environments. In *ACM SIGGRAPH '94 Proceedings*, pages 435–442, Orlando, 1994.

[16] W. Sturzlinger, G. Schaufler, and J. Volkert Johannes. Load Balancing for a Parallel Radiosity Algorithm. In *IEEE/ACM 1995 Parallel Rendering Symposium*, pages 39–45, Atlanta, October 1995.

[17] Seth Teller, Celeste Fowler, Thomas Funkhouser, and Pat Hanrahan. Partitioning and Ordering Large Radiosity Computations. In *ACM SIGGRAPH '94 Proceedings*, pages 443–450, Orlando, 1994.

[18] R. van Liere. Divide and Conquer Radiosity. In *Proc. of the Second Eurographics Workshop on Rendering*, pages 191–197, Barcelona, 1991. Springer-Verlag.

[19] Amitabh Varshney and Jan F. Prins. An Environment-Projection Approach to Radiosity for Mesh-Connected Computers. In *Proc. of the Third Eurographics Workshop on Rendering*, pages 271–281, Bristol, UK, May 1992.

[20] Hau Xu, Qun-Sheng Peng, and You-Dong Liang. Accelerated Radiosity Method for Complex Environments. In *Eurographics '89*, pages 51–61. Elsevier Science Publishers, Amsterdam, September 1989.

[21] Marco Zagha, Brond Larson, Steve Turner, and Marty Itzkowitz. Performance Analysis Using the MIPS R10000 Performance Counters. In *Supercomputing '96*, Pittsburgh, 1996.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399