HAL
open science

Binary Mesh Partitioning for Cache-Efficient Processing

Marc Tchiboukdjian, Vincent Danjean, Bruno Raffin

To cite this version:

Marc Tchiboukdjian, Vincent Danjean, Bruno Raffin. Binary Mesh Partitioning for Cache-Efficient Processing. [Research Report] RR-7118, INRIA. 2009. inria-00436052

HAL Id: inria-00436052

https://inria.hal.science/inria-00436052

Submitted on 25 Nov 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Binary Mesh Partitioning for Cache-Efficient Processing

Marc Tchiboukdjian — Vincent Danjean — Bruno Raffin

N° 7118

November 2009

Thème NUM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7118--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche GRENOBLE - RHÔNE-ALPES

Binary Mesh Partitioning for Cache-Efficient Processing

Marc Tchiboukdjian* , Vincent Danjean† , Bruno Raffin‡

Thème NUM — Systèmes numériques
Équipe-Projet MOAIS

Rapport de recherche n° 7118 — November 2009 — 65 pages

Abstract: One important bottleneck when visualizing large data sets is the data transfer between processor and memory. Cache-aware (CA) and cache-oblivious (CO) algorithms take into consideration the memory hierarchy to design cache efficient algorithms. CO approaches have the advantage to adapt to unknown and varying memory hierarchies. Recent CA and CO algorithms developed for 3D mesh layouts significantly improve performance of previous approaches, but lack of theoretical performance guarantees. We present in this report a $O(N \log N)$ algorithm to compute CO layout for unstructured meshes. We prove that a coherent traversal of a N-size mesh in dimension d will induce less than $N/B + O(N/M^{1/d})$ cache-misses where B and M are the block size and the cache size. Experiments show that our layout computation is faster and significantly less memory consuming than for the best known CO algorithm. Performance is comparable to this algorithm for classical visualization algorithm access patterns, or better if the access pattern is adapted to the binary mesh partitioning produced by the algorithm. We also show that cache oblivious approaches lead to significant performance increases on recent GPU architectures.

Key-words: Cache-aware, cache-oblivious, mesh layouts, data locality, unstructured mesh, isosurface extraction.

* Marc.Tchiboukdjian@imag.fr, CNRS and CEA/DAM,DIF
† Vincent.Danjean@imag.fr, Grenoble Universités
‡ Bruno.Raffin@imag.fr, INRIA

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Partitionnement Binaire de Maillage pour un Traitement Efficace en Cache

Résumé : Les transferts de données sont le principal goulot d'étranglement lors de la visualisation de gros jeux de données. Les techniques cache-aware (CA) et cache-oblivious (CO) modélisent la hiérarchie mémoire pour concevoir des algorithms efficaces en cache. De plus, le modèle CO permet de s'adapter à la hiérarchie mémoire sans en connaître les paramètres. Les algorithmes CA et CO développés récemment pour la réorganisation de maillage 3D améliorent significativement les performances des approches précédentes mais manquent de garanties théoriques. Nous présentons dans ce rapport un algorithme de réorganisation de maillages non structurés en temps $O(N \log N)$. Nous garantissons que la traversée cohérente d'un maillage de taille N en dimension d ne cause pas plus de $N/B + O(N/M^{1/d})$ défauts de cache avec B la taille d'une ligne de cache et M la taille du cache. Les expériences montrent que notre algorithme de réorganisation de maillage est plus rapide et consomme beaucoup moins de mémoire que le meilleur algorithme CO connu. Les performances du maillage obtenu sont comparables pour les filtres de visualisation standards mais meilleures si les schémas d'accès sont adaptés à la découpe récursive produite par notre algorithme. Nous montrons également une importante amélioration des performances sur les architectures GPU récentes.

Mots-clés : localité des données, maillage non structuré, extraction d'isosurface.

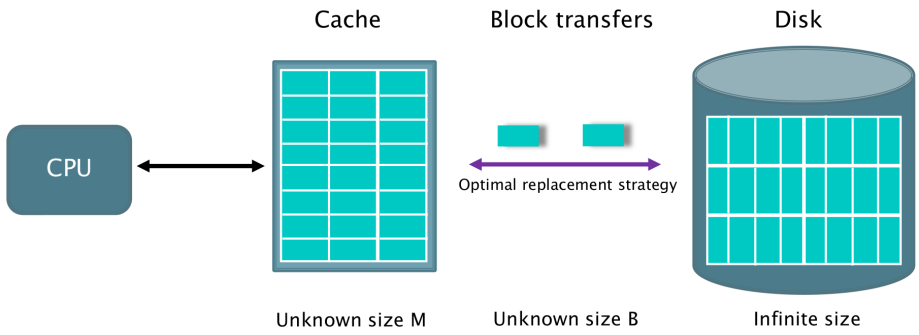


Figure 1: The cache-oblivious memory model. Data is transfered by block of B consecutive elements into a cache of size M. Both parameters are unknown to the algorithm.

1 Introduction

Combining adequately data layout and access patterns can significantly improve performance for many memory intensive applications. As classical processor architectures cache adjacent data blocks, data accessed consecutively should be stored nearby in memory to reduce cache-misses. Enforcing locality is also relevant for some SIMD or vector processing units where parallel accesses require fewer clock cycles when the target data are close in memory (nvidia GPUs [19] or Intel Larabee [21] for instance). We can expect locality of data access to be even more critical for emerging many-core architectures that show complex memory hierarchies where cores share cache and memory bandwidth.

Many visualization applications rely on read-only and memory intensive algorithms with coherent access patterns usually based on spatial locality. Space filling curves, like the Z curve, are commonly used for regular data structures [18]. They provide a cache-efficient layout for access patterns showing a strong spatial locality. For irregular data structures, computing cache-efficient layouts is significantly more difficult.

We can distinguish two classes of cache-efficient algorithms: Cache-Aware (CA) and Cache-Oblivious (CO) algorithms. CA algorithms are based on the external-memory (EM) model [1]. The memory hierarchy consists of two levels, a main memory of size M called cache and an infinite size secondary memory. Data is transferred between these two levels in blocks of B consecutive elements. CA algorithms can be very efficient but require the layouts to be recomputed for each memory architecture. It makes it difficult to efficiently share the same layout between heterogeneous processing units mixing CPUs and GPUs for instance. CO approaches [12] intend to overcome this limitation by proposing layouts that are independent from the cache and block size M and B (Fig. 1). The Z curve is an example of a CO layout (Fig. 2). For irregular data structures, the most significant and recent work is probably the CO mesh layout proposed by Yoon et Al. [26] (OpenCCL algorithm). In comparaison to other layouts, experiments show speedups ranging from 2 for in-core computations, up to 20 for out-of-core computations. However this algorithm is based on heuristics, without theoretical performance guarantee, neither on the layout computation complexity nor the quality of the resulting layout.

In this report we introduce a new CO layout algorithm for irregular meshes with a theoretical performance guarantee. It relies on a recursive mesh partitioning using a specific BSP (Binary Space Partitioning) algorithm introduced by Miller *et al.* [16].

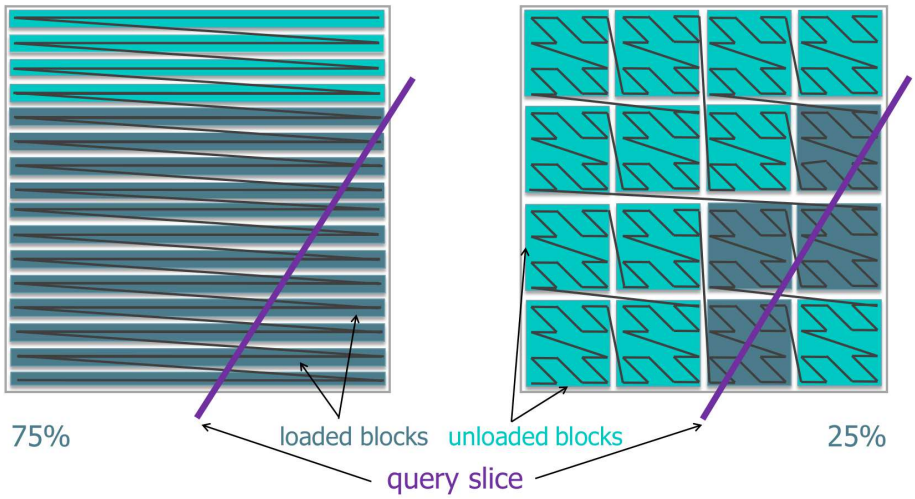


Figure 2: Good layouts can significantly reduce the number of block transfers. On the left, 75% of the data must be loaded to access the queried slice (each line corresponds to a cache line), while the CO layout used on the right (Z curve) enables to reduce this amount to only 25% of the data (each block fits into a cache line).

This algorithm cuts the mesh guaranteeing a good tradeoff between minimizing the number of cut elements and having two partitions of close size. When applied recursively it ensures that spatially close and strongly connected data tend to be partitioned deeper in the BSP tree. The CO layout is obtained by storing the data linearly in memory from the first leaf of the BSP tree to the last one. The data loaded in a cache block are thus contiguous leaves of the BSP tree. It is cache oblivious as to any block and cache size correspond a BSP tree depth level ensuring a strong locality and connectivity.

This CO layout algorithm shows several benefits. The layout computation has a $O(N \log N)$ complexity. It also guarantees that a coherent traversal of an N-size mesh in dimension d induces less than $N/B + O(N/M^{1/d})$ cache-misses where B and M are the block and cache size. Experiments show the layout computation is about two to three times faster than for the OpenCCL algorithm while requiring significantly less memory (from 5 to 35 times measured in our experiments). We are comparable with the OpenCCL algorithm for classical visualization algorithm access patterns. But experiments reveal that adapting the access pattern to the BSP tree can bring significant additional performance improvements (about 12% for in-core computations).

We also show that CO layouts can lead to significant performance improvements on recent NVIDIA GPUs (1.2 to 2.7 speedups), even if no cache mechanism is involved. As our algorithm enforces data locality, access to the mesh are faster due to coalesced data access.

Related work is discussed in section 2. The algorithm is presented and analyzed in section 3 and its implementation discussed in section 4. Then we present experimental results in section 5.

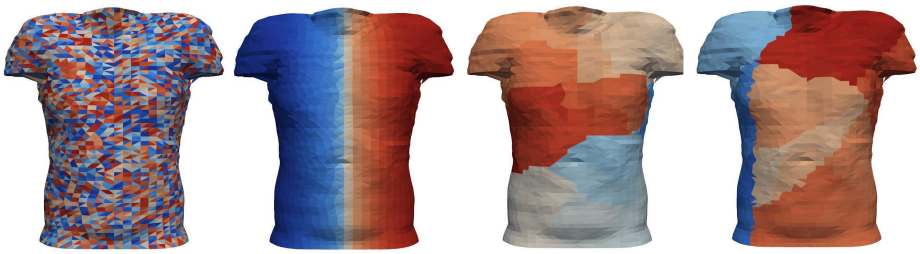


Figure 3: Visual illustration of different layouts for the torso mesh. Successive cells in memory are colored from blue to red. From left to right, the original, geometric (sorted by x,y and z coordinates), OpenCCL (cache-coherent layout from [26]) and FastCOL (our approach) layouts. For spatially close triangles, color discrepancy decreases from the left to right layouts. It denotes an improved memory locality, less likely to generate cache-misses for spatially coherent access patterns.

2 Related Work

2.1 Cache-Efficient Algorithms

Today, many algorithms have their CA or CO versions [15] where computations and data are reordered for an efficient cache use. A widely used technique is blocking or tiling: elements are mapped in memory and accessed by blocks of size B to fit in a cache line. For instance, regular search trees are made CA by grouping B keys in a single node. Such tree is called a B-tree. Another example is matrix multiplication. Instead of rows or columns, the ATLAS library [23] traverses the matrix by blocks such that all involved blocks for a partial computation fit in the cache. It is often possible to obtain the same result while being oblivious to the block size.

For example, by carefully storing a binary tree in memory with the vam Emde Boas layout [15], a search can match the I/O bound of the CA B-tree. In this layout, the tree is divided at the middle of its height into a top tree and several bottom trees. The same layout is recursively applied to each of these trees, which are then stored sequentially in memory. The same recursive blocking technique is applied within the divide and conquer matrix multiplication. Indeed, this is a CO algorithm [15] with the same theoretical complexity as its CA counterpart.

Another CO alternative to blocking is the use of space filling curves. For example, it works perfectly with matrix multiplication [2] or regular meshes [18].

CO algorithms for regular structures are not always as efficient as their CA counterpart. The access pattern to the CO layout is more complex leading to a significant overhead impairing performance. For instance, CO matrix multiplication is not competitive with the CA version [28]. We do not face this problem here as an unstructured mesh has already a complex access pattern.

2.2 Mesh Layouts

The problem of reordering mesh elements for efficient cache use was first encountered when a vertex cache has been introduced in graphic cards. In order to maximize the efficiency of the hardware vertex cache, triangles needed to be reordered before being sent to the graphic card. The algorithm developed by [13] reorders triangles to form

triangle strips. They assume the cache has a FIFO policy and the cache size is known to the algorithm. New algorithms that works for all cache sizes has been introduced in [3]. However, as they target graphic cards, they only reorder the mesh cells and not the points. Moreover, their consider the graphic card cache model (no cache line, independent vertices fetching, etc.) which is very different from the CPU cache models. To be efficient, the application must access the mesh in the exact order given by the cell layout (especially for triangle strips). Finally, work in this area mostly dealt with surface meshes.

Processing sequences [14] reorder points and cells of the mesh, but this approach focuses on streaming computations. The goal is to minimize the maximum amount of memory used during the computation. The application should again follow the mesh layout.

OpenCCL [17] presented in [26] casts the mesh layout problem as a graph optimization problem. To describe the access pattern of the application using the mesh, the user must provide a graph where vertices represent data and edges link data that are likely to be accessed in sequence at runtime. A good mesh layout is a permutation of the graph vertices that results in a more efficient layout of the mesh in memory. They developed a local metric to decide if a swap of some vertices improves the layout. They optimize this measure thanks to a multilevel optimization scheme. However their algorithm is based on a heuristic that does not give the global optimum. They do not provide a bound on the quality of the layout, i.e. the resulting number of cache-misses. Our algorithm FastCOL gives an upper bound on the number of cache-misses incurred by the application at runtime. This bound is closely related to the CA and CO metrics introduced in [25].

Aforementioned approaches mainly focused on optimizing mesh layouts when the application accesses the mesh without the help of any additional data structure. That is, the application only traverses the mesh with the help of the cells and points arrays or with the cell-to-points or point-to-cells pointers of the mesh. Another approach [27] optimizes the layout for applications accessing the mesh through bounding volume hierarchies (BVH) trees. To generate an efficient layout, they use the OpenCCL algorithm and provide two kind of links in the access graph: links representing spatial locality and links representing parent-child locality in the BVH to optimize. Our algorithm also handles these two kinds of locality. During the layout computation, we build a BSP tree that is used to reorder the mesh. This tree is tailored to efficiently use our mesh layout. In opposite to the approach in [27] where the layout algorithm takes a mesh and a BVH tree as input to produce a layout, we only take the mesh as input and produce both a layout and a BSP tree consistent with this layout. This BSP tree can be used as an acceleration structure, for isosurface extraction for instance.

2.3 Isosurface Extraction

The marching tetrahedron algorithm can be accelerated with various data structures allowing to efficiently search for the cells intersected by the isosurface. The first data structure was the min-max tree [24]. An octree where each node stores the minimum and maximum values of its subtree permits to quickly discard parts of the mesh that do not contain any intersected cell. The search is thus improved from $O(n)$ to $O(k + k \log n/k)$ where n is the number of cell and k the size of the isosurface (usually $k \ll n$). If the scalar field is spatially coherent, the performance is actually improved over the theoretical bound.

An optimal data structure for this problem is the interval tree storing for each cell c the interval whose extremes are the minimum and maximum value of the points of c [10]. The query time is improved to $O(\log n + k)$ whatever the spatial repartition of the scalar field is. The interval tree has been made I/O-efficient allowing a query with complexity $O(\log_B n + k/B)$, which is optimal [7]. However this approach is not space-efficient since vertex information is duplicated many times. The 2-level indexing scheme based on the meta-cells technique introduced in [8, 9] is both practical and space-efficient as there is no duplicated information. Spatially close cells are grouped into meta-cells, which are then used in the I/O-efficient interval tree.

The approach we developed with the consistent BSP tree is a CO alternative to the meta-cells technique but we use the min-max tree instead of the interval tree which may not be as efficient on a scalar field with high spatial variations.

3 Binary Mesh Partitioning Recursive Layout

The goal is to define CO mesh layouts for spatially coherent access patterns, i.e. when elements accessed consecutively are spatially close. Such patterns encompass a large range of visualization algorithms.

These access patterns are modeled by a class of graphs called overlap graphs. There exists an optimal separator for overlap graphs, which, when applied recursively, provides a CO mesh layout.

3.1 Overlap Graphs

We model the mesh by a graph $G = (V, E)$ where vertices represent elements of the mesh (points or cells), and edges link elements that are likely to be accessed consecutively. We also associate to each vertex v_i a coordinate p_i in $\mathbb{R}^d$, usually the coordinate of the corresponding element of the mesh. These graphs can model a large range of access patterns, even ones with a weak spatial coherency where edges connect distant elements. To be able to build efficient CO layouts with a provable quality, we restrict to the class of graphs modeling spatially coherent access patterns, i.e. where edges connect spatially close elements. Such graphs are called overlap graphs [16].

A graph is α-overlap if it is possible to associate to each vertex v_i a ball B_i centered at p_i such that each pair of balls intersect in at most one point. The radii of the balls can be freely adjusted allowing the modeling of meshes with non uniform cell sizes. Additionally edges can connect two vertices only if expanding the smaller of their two balls by a factor of α make them intersect.

Meshes are often composed of elements that are well shaped in some sense, such as having a bounded aspect ratio or angles that are not too small or too large. Such geometric features ensure that edges meet the overlap graph constraint. Every mesh with bounded aspect ratio elements in two or three dimensions is contained in some overlap graph (for a good choice of the parameter α).

Most of the spatially coherent access patterns on meshes are overlap graphs. For instance, the graph where vertices are points of the mesh and two vertices share an edge when they share an edge in a cell is an overlap graph. The one where vertices are cells of the mesh and two vertices share an edge when cells share a face is also an overlap graph.

3.2 Geometric Separator Algorithm

Given their geometrical properties, overlap graphs can be partitioned efficiently in two parts of approximately equal size, while minimizing the number of edges cut.

The following randomized algorithm introduced by Miller *et al.* [16] computes in linear time and with a high probability an optimal geometric separator. It starts by randomly sampling a constant number of points from the input mesh and project them onto the surface of the unit sphere in $\mathbb{R}^{d+1}$ (Algo. 1). Then it finds a centerpoint of this random sample in linear time relative to the sample size. A point is a centerpoint if every hyperplane passing through it divides the sample set approximately evenly, at most in the ratio $d+1:1$. With good probability, this centerpoint is a centerpoint of the original set of points [11]. Finally we randomly choose a hyperplane through the centerpoint. Repeating this process and selecting the separator cutting the smallest number of edges gives a small separator with high probability.

Algorithm 1: Geometric separator algorithm

Input: Graph $G = (\text{Vertices } V, \text{Edges } E)$
Output: A separator ϕ

1 **repeat** n_c times
2 $\quad V_s \leftarrow$ sample of $(d+3)^4$ points of V
3 $\quad V_p \leftarrow$ project V_s to the unit sphere in $\mathbb{R}^{d+1}$
4 $\quad c \leftarrow$ centerpoint of V_p
5 $\quad (\mathbf{u}, r) \leftarrow$ rotation and scaling factor to move c at the origin
6 $\quad$ **repeat** n_h times
7 $\quad\quad \mathbf{n} \leftarrow$ random normal vector
8 $\quad\quad \phi \leftarrow$ separator defined by $(\mathbf{u}, r, \mathbf{n})$
9 $\quad\quad$ compute the number of edges cut by ϕ
10 $\quad$ **end**
11 **end**
12 **return** the best ϕ

The most time consuming part of the algorithm is the quality evaluation of the separator (Alg. 1 line 8) as other operations involve only a small number of points.

The quality of the obtained separator is guaranteed by the following theorem.

Theorem 1 (Geometric separator [16]) *Let G be an n-vertex α-overlap graph in d dimensions. With high probability, the previous algorithm (Alg. 1) partitions the vertices of G into two sets A and B such that $|A|, |B| \leq \frac{d+1}{d+2}n$ and the number of edges between A and B is $O\left(\alpha n^{1-1/d}\right)$.*

Such a separator is asymptotically optimal for the class of overlap graphs. Indeed we cannot find a smaller separator for a regular d dimensional grid [16].

3.3 Mesh Layout Algorithm

The recursive application of the separator gives a BSP tree where each node is a separator. Leaves of this tree correspond to small subparts of the mesh that are stored consecutively to provide the layout (Algo. 2).

Algorithm 2: Layout algorithm

Input: Graph G
Output: Permutation of the nodes π
1 $\mathscr{T} \leftarrow$ complete BSP tree of the graph G
2 $\pi \leftarrow$ left to right order of the leaves of $\mathscr{T}$
3 **return** π

As the geometric separator has linear complexity, our layout algorithm has complexity

$$W(N) = \max_{1/2 \leq \lambda \leq \delta} [W(\lambda N) + W((1-\lambda)N)] + O(N) = O(N \log N)$$

where $\delta = \frac{d+1}{d+2}$. The only requirement to obtain the claimed complexity is to have a point sampler of linear complexity and an iterator on edges of linear complexity too.

3.4 Layout Quality

In the following theorem, we exhibit a bound on the quality of the layout generated by algorithm 2.

Theorem 2 (Layout quality) *Let G be a N-vertex α-overlap graph in d dimensions representing the access pattern of a d-dimensional N-size mesh. The layout produced by algorithm 2 ensures that a traversal of the mesh respecting the access pattern of G induces less than $N/B + O(N/M^{1/d})$ cache misses where B and M are the block and cache size.*

Let T the BSP tree obtained by recursively applying the geometric separator. Let $T_m \subset T$ a subtree of the BSP tree with the same root but whose leaves are nodes x of T verifying $size(father(x)) > m$ and $size(x) \leq m$. We first compute k_m, the number of edges cut by all the separators in T_m. We show that $k_m = O\left(\frac{N}{m^{1/d}}\right)$.

Let $K(r)$ be the maximum number of cut edges in a subtree T'_m of T_m rooted at a subgraph of size r. If $r \leq m$ then T'_m is just a leaf and $K(r) = 0$. If $r \geq m$, we count the edges cut in the left and right children and the edges cut by the separator of the root of T'_m:

$$K(r) \leq \max_{1/2 \leq \lambda \leq \delta} [K(\lambda r) + K((1-\lambda)r)] + c\alpha r^{1-1/d}$$

where c is a constant and $\delta = \frac{d+1}{d+2}$ (Th. 1). By induction, we can prove that

$$K(r) \leq c' \left(\frac{r}{m^{1/d}} - r^{1-1/d}\right)$$

as long as

$$c' \geq \frac{\alpha c}{2^{1/d} - 1}.$$

And thus

$$k_m = K(N) = O\left(\frac{N}{m^{1/d}}\right).$$

We now model an application using the mesh as a function f to be applied to all elements of the mesh. The function f is in agreement with the access graph if, to

process element i, only the neighbors of i in the graph need to be accessed. We also make the following assumptions:

- all elements in the mesh are processed exactly once; (a)
- elements are accessed one leaf of T_m after the other (with $m \leq M$). (b)

The order between leaves and the order inside each leaf can be freely chosen. As each element of the mesh is processed at least once, the number of compulsory cache misses is at least $\frac{N}{B}$. This is a lower bound on the number of cache-misses of any application that need to process each element of the mesh. Following edges that link an element in a leaf l and an element outside leaf l causes $O(1)$ extra cache-misses (bringing the corresponding block in cache can evicts a block corresponding to leaf l that might be needed later). These extra cache-misses for all leaves sum up to $O(k_m)$ misses. After the whole mesh has been processed, the number of cache-misses is thus $\frac{N}{B} + O(k_m)$. Combining this result with the previous one we obtain $\frac{N}{B} + O\left(\frac{N}{m^{1/d}}\right)$. If elements are processed in the order given by the layout, we can take $m = M$ and we obtain the claimed bound.

At this point we cannot directly compare this algorithm with OpenCCL. OpenCCL is based on a meta-heuristic and no upper bound on the quality of the resulting layout is given.

3.5 Discussion on the Access Graph Model

The access graph modeling the application access pattern is not exact. First, this graph just gives local information on which data is accessed but no global information on how the graph is traversed. Next, real life applications do not rely only on intrinsic mesh characteristics such as topology or geometry when accessing the mesh but sometimes on the layout itself. For example, the connectivity filter of VTK only uses the mesh topology, but a marching tetrahedron algorithm processes mesh cells in the order of the layout.

With our layout, an application still needs to respect some properties (assumptions (a) and (b) in the previous section) while traversing the access graph. We believe that we could obtain the same performance guarantee slackening these two assumptions to rely only on the characteristics of the mesh itself. We are however not able to prove it yet.

4 Implementation

4.1 Choosing the Access Graph

The algorithm described in section 3 can be applied to any access pattern as long as the corresponding graph is an overlap graph (actually the algorithm still works if it is not the case but the bound on cache-misses does not hold). However it is not practical to generate a new layout for each application. For instance to compute a volume rendering by ray casting of the mesh, one might want to optimize the mesh layout according to the rays direction. Both our algorithm and OpenCCL are two slow to generate a layout before each image generation. In practice it is better to compute only once a layout that will be efficient in general. We choose in the implementation to only consider the

graph where vertices are points of the mesh and edges link two vertices sharing a cell. Following on our volume rendering example, this layout will be reasonably good for any rays direction. One ray traversing c cells of the mesh induces $c/B^{1/3}$ cache-misses while a layout optimized for this specific direction may induce only c/B cache-misses (but as bad as 1 cache-miss per cell for an orthogonal direction). In this specific case, packing rays should also improve performance of the more general layout to c/B.

4.2 Layout Computation

We implemented the geometric separator algorithm in C++. We first randomly generate all the $n_h n_c$ separators. Half of them are hyperplanes and the other half are lines. We then traverse all the cells of the mesh and for each of them we check that its points are on the same side of the separator. If not, the cell is cut by the separator and we increment the cut size by 1. Using cells instead of edges to select the best separator produces a very close result and allows us not to compute the edges of the graph, which can be computationally expensive. All separators are checked against a cell before going to the next one. This allows us to dereference each cell index only once for the entire separator computation.

To keep the memory usage low, we do not project all the points before evaluating a separator but project them on the fly. This induces duplicate computation as a point is used in several cells but keep memory overhead close to zero. That way we do not need to store an entire copy of the points in memory.

Once we found the best separator, points of the mesh are reordered according to this separator. All points laying to the left of the separator are moved to the left part of the array and points laying to the right are moved to the right part of the array. The same partitioning is done on cells. When a cell is cut by the separator we choose a side according to the center of gravity of the cell. We then recurse on the left and right mesh generated. This algorithm is very similar to quicksort and could be efficiently parallelized.

We stop when a submesh has a size lower than 8. We choose $n_c = 2$ and $n_h = 30$ for the experiments (Alg. 1). As the randomized centerpoint algorithm is quite good we can keep n_c low. During our experiments we noticed that substantial changes of all these parameters does not impact very much the quality of the generated layout. The expensive part of the quality evaluation is the rotation and scaling of each center point.

4.3 Cells Layout

A mesh layout tries to optimize both points and cells ordering. As points and cells are usually accessed in a similar way, a consistent ordering for points and cells is better (for instance, in an isosurface computation points defining each cell will often be accessed right after the cell itself). Our geometric approach is naturally good as the same geometric separator can be applied for both points and cells. A separator cutting few edges for the points graph also cut few edges for the cells graph. It is not possible to do the same with OpenCCL for example as their separators are combinatorial and not geometric. It is often less efficient to compute both points and cells layouts independently. In this case, computation time is longer and runtime efficiency lower. For OpenCCL, the cell layout takes 3.2 more time than the points layout on the bucky ball mesh. An usual trick is to compute the cells layout using the order given by the minimal index of points in the cell.

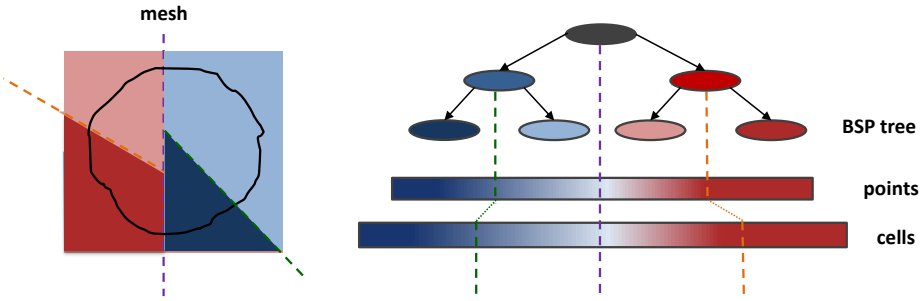


Figure 4: Illustration of the correspondance between mesh regions, BSP tree branches, and data arrays.

Consistency of points and cells layout can futher improve performance. On large meshes the consistent layout for points and cells with the BSP tree is up to 10% faster than the `min-vertex` layout applied to FastCOL on isosurface extraction. Moreover `min-vertex` applied to OpenCCL is up to 20% faster than OpenCCL applied to both points and cells.

4.4 Consistent BSP Tree

We use the BSP tree defining the partitioning of the mesh as a min-max tree. At each node of the BSP tree is stored the minimum and maximum value of the scalar field in the corresponding region of the mesh. A region that do not contain any cell intersected by the isosurface can be quickly discarded.

A nice property of this BSP-tree is that each region corresponds to a small part of the mesh stored sequentially in memory (see fig. 4). When traversing the BSP-tree in prefix order and examining the mesh cells that might contain a part of the isosurface, mesh cells are accessed sequentially. The sequence of cells can jump part of the mesh but never goes back. This leads to a traversal of the mesh inducing less cache-misses (see section 5.2.2).

5 Experiments

In our experiments, we compared the initial layout, the geometric layout, the one computed with OpenCCL, and the one computed with our FastCOL algorithm by running ten classical filters and measuring the execution time and the cache-misses. In this part, we describe the experiments performed. For sake of conciseness, we present only some representative results. Full results are provided in appendix A. Eventually, we show that these results can be correlated with the property that our algorithm optimizes (the locality of the vertices and edges in the layout).

5.1 Description of the Experiments

The experiments were conduced on three kinds of configurations: two generic processors (AMD Opteron875 @ 2.2Ghz, cache L1 8KB, cache L2 1MB and Intel Core2 E6750 @ 2.66Ghz, cache L1 32KB, cache L2 4MB) and one graphical processor (NVIDIA GTX280 30MP 1GB programmed with cuda 1.3).

We took 9 different meshes[1], processed to generate several instances of various sizes. For the original mesh, we subdivide some cells by adding a new vertex. It leads to a set of 50 meshes that can be divided by their size into 4 groups: 5 meshes of about 100 k cells, 10 meshes of about 1 M cells, 17 meshes of about 10 M cells, and 18 meshes of about 50 M cells.

Ten filters have been tested on each layout and are presented below. Six of them are algorithms implemented directly within VTK [20], two have been manually developed on CPU, and the last two ones has been written with CUDA on GPU.

Gradient The VTK gradient filter computes at each point the gradient of the scalar field of the mesh. Each point p accesses the scalar data of points connected to p by an edge of the mesh. Points are processed in the order given by the point layout.

Connect The VTK connectivity filter applies a breadth first search on the mesh to compute the connected region each cell lies in. Each cell c accesses cells sharing a point with c. Cells are processed in an order depending only on the topology of the mesh.

RC The VTK Bunyk Ray Cast filter computes a volume rendering of the mesh following [5]. Each ray traverses the mesh cell by cell.

PT The VTK Projected Tetrahedra filter computes a volume rendering of the mesh following [22]. Tetrahedra are sorted by their centroid according to the viewing direction and then sent to the GPU for projection. Each tetrahedron accesses to its points. Tetrahedra are processed in the order given by the cell layout.

HAVS The VTK HAVS filter computes a volume rendering of the mesh following [6]. Data accesses are similar to PT.

VtkIso The VTK isosurface extraction filter implements the marching tetrahedron algorithm. Each cell accesses to its points. Cells are processed in the order given by the cell layout.

CpuIso and GpuIso Home-made implementations of the marching tetrahedron algorithm. The GPU implementation is developed with CUDA.

CpuTree and GpuTree The search for cells intersected by the isosurface is accelerated with a min-max tree where the tree is simply a kd-tree. Each cell accesses to its points, like with the regular marching tetrahedron, but cells are now processed in a different order than the one of the cell layout. Also, not all cells are processed. We only measure the time spend in processing the intersected cells and not the traversal of the tree.

The bigger meshes (50 M cells) have not been tested on the volume rendering filters due to the very large execution time of these filters. Moreover, they have not been tested on GPU, due to the insufficient amount of memory available.

[1] Blunt fin, buckyball, langley fighter, liquid oxygen post, plasma64, san fernando and spx models are provided by the AIM@SHAPE Shape Repository (`http://shapes.aim-at-shape.net/`). Torso is courtesy of SCI and the last one is not published.

Table 1: Layout computation

Mesh size		OpenCCL		FastCOL	
#cells	Bytes	Time	Mem. space	Time	Mem. space
100k	3.7 MB	4.5 s	123 MB	1.2 s	22 MB
1M	43 MB	54 s	1.24 GB	17 s	81 MB
10M	370 MB	9 min 24 s	9.96 GB	3 min 50 s	0.56 GB
50M	1.8 GB	NA	$>$ 96 GB	26 min 44 s	2.72 GB

5.2 Results

In our experiments, we considered two aspects: the computation of the layouts and the effect of the layout on the filter executions.

5.2.1 Layout Algorithms

All layouts have been prepared on a Opteron875 @ 2.2Ghz with 32 GB of memory and 64 GB of swap. The geometric layout is only a coordinate sort by the x, y and z axes, so it is very fast (less than 40 s for the biggest meshes) and not memory consuming. The results presented in the table 1 include the execution time and memory consumption for computing the OpenCCL and FastCOL layouts.

Table 1 shows that our FastCOL program is about three times faster than the OpenCCL one. Our FastCOL program also requires far less memory to compute its layout. The bigger meshes with 50 M cells have not been processed with OpenCCL because their computation would have required more than 96 GB of memory.

5.2.2 Filters Performance with respect to Mesh Layouts

For each of the 50 meshes and for each filter, we measured the global time of the experiment. When using a filter on a generic processor, we also measured the number of L1 and L2 cache-misses with the PAPI software [4]. When using a filter on the GPU, we measured the number of uncoalesced parallel accesses. A parallel global memory access performed by all threads of a half-warp is coalesced into a single memory transaction as soon as the words accessed lie in the same segment of size equal to 128 bytes for 32-bit or 64-bit words.

For each experiment (architecture, graph and algorithm fixed), the measured time, the number of cache-misses and the number of uncoalesced accesses are very stable.

In tables 2 and 3, we use the initial layout as the reference and we compute the ratio with this one for the three other layouts. Then, we compute the means of these ratio for each group of 10 meshes. The "time" columns show the speed-up for the considered layout and the "L2" or "Unco." columns show the ratio of saved L2 cache-misses or uncoalesced memory accesses. In all cases, higher values are better.

Table 2 compares the initial layout versus the geometric, OpenCCL and FastCOL layouts. Table 3 compares the initial layout versus the OpenCCL layout and two variants of the FastCOL layout. The first one uses an external min-max tree while the second one reuses the BSP tree the layout is based on as described in 4.4.

[2]Uncoalesced global memory access

Table 2: Speed-up of Geometric, OpenCCL and FastCOL layouts over the original layout on various visualization algorithms on Core2 E6750 at 2.66Ghz, with 32 kB of L1 cache, and 4 MB of L2 cache.

	Mesh	Geometric		OpenCCL		FastCOL	
	size	Time	L2	Time	L2	Time	L2
Gradient	100k	1.02	1.51	1.01	1.49	1.02	1.52
	1M	1.06	3.53	1.07	4.03	1.08	3.94
	10M	1.07	2.36	1.15	8.22	1.15	7.81
	50M	1.1	1.34			1.36	10.53
Connect	100k	0.95	0.94	1.12	1.17	1.11	1.21
	1M	0.97	0.95	1.16	1.19	1.19	1.19
	10M	1.09	1.08	1.45	1.49	1.46	1.49
	50M	0.89	0.87			1.66	1.9
RC	100k	0.98	1.08	1.05	1.4	1.06	1.36
	1M	1.01	1.07	1.2	1.8	1.2	1.79
	10M	0.76	0.72	3.28	5.02	3.2	4.89
PT	100k	1.06	0.82	0.93	1.18	1.15	1.18
	1M	0.91	0.64	1.09	1.51	1.1	1.52
	10M	0.97	0.9	1.37	2.66	1.37	2.65
HAVS	100k	1.02	2.04	1.01	2	1.08	2
	1M	1.14	3.43	1.06	4.04	1.09	4.03
	10M	1.2	1.9	1.33	5.96	1.32	5.77
VtkIso	100k	0.99	1.04	1.04	1.06	1.04	1.09
	1M	1.1	1.22	1.15	1.24	1.15	1.24
	10M	1.32	1.71	1.44	1.79	1.44	1.78
CpuIso	100k	1.06	1	1.16	1.02	1.17	1.02
	1M	1.71	2.85	2.34	2.8	2.35	2.78
	10M	2.28	5.4	4.08	5.78	3.99	5.68
	50M	0.97	0.79			4.87	6.84
		Time	Unco.[2]	Time	Unco.[2]	Time	Unco.[2]
GpuIso	100k	0.96	1.18	1.56	3.08	1.52	2.97
	1M	1.26	1.04	2.2	2.59	2.11	2.38
	10M	1.83	1.25	4.09	3.85	3.8	3.39

Table 3: Speed-up isosurface extraction with a kd-tree used as a min-max tree except for FastCOL (bsp) that uses the BSP tree of the mesh layout.

	Mesh	OpenCCL		FastCOL		FastCOL (bsp)	
	size	Time	L2	Time	L2	Time	L2
CpuTree	100k	1.23	1.31	1.21	1.30	1.37	1.23
	1M	1.46	1.8	1.45	1.74	1.75	1.72
	10M	2.37	3.14	2.31	3.02	2.75	3.06
	50M			2.92	3.47	4.55	4.85
			Unco.[2]		Unco.[2]		Unco.[2]
GpuTree	100k	1.20	1.85	1.18	1.75	1.33	1.90
	1M	1.63	1.81	1.57	1.67	1.84	1.89
	10M	2.50	2.14	2.34	1.86	2.79	2.14

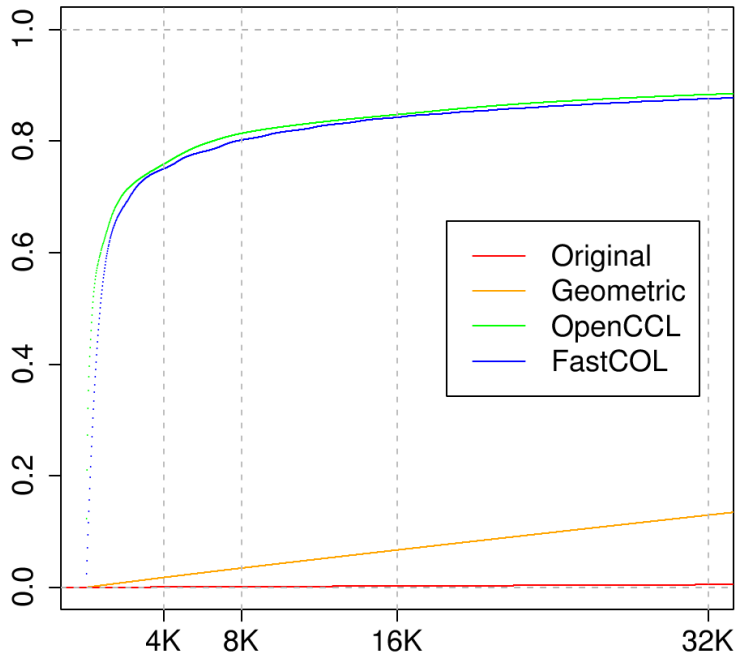
5.3 Discussion and Interpretation

Our results clearly show that mesh layouts can significantly influence filter performance. Results are consistent: more variations between layouts can be observed with bigger meshes where cache effects are predictably more important. Indeed, with smaller meshes, a bigger part of the meshes can be loaded in the cache, whatever the layout is.

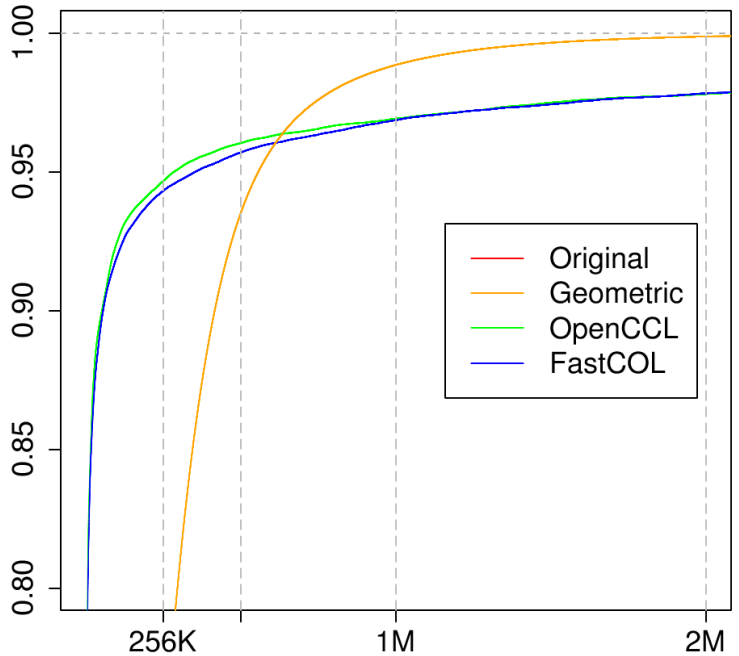
We can observe that OpenCCL and FastCOL layout nearly always lead to a speed-up greater than 1. This means that cache-oblivious layouts are an efficient approach to globally improve filter performance. Results for OpenCCL and FastCOL are similar. However, the layout computation time is about 3 times smaller for FastCOL and the memory footprint is far less important. OpenCCL is unable to compute a layout for a 50 M cells mesh with 96 GB of memory and swap, whereas this is where the biggest speed-up is observed with the FastCOL layout.

We can also observe that measured speed-ups are generally smaller with VTK filters than with home-made filters. This clearly appears for the iso-surface filter which is implemented with VTK (VtkIso) and with an home-made code (CpuIso). The VTK implementation shows a maximum speed-up of 1.44 whereas our implementation goes up to 4 (with a smaller global execution time). This is due to the VTK library that is not fully optimized and performs several other computations. For instance, in the VtkIso filter, after the extraction of the isosurface, same points are merged to give a mesh as a result.

As explained previously in this report, we think that speed-up improvements are due to better data locality in memory. We will present here some more evidence of this fact. We call "length of an edge" the memory gap between the two vertices of this edge in the vertex array loaded in memory. If a mesh has smaller edges, more of them will fit in cache and a better performance should be observed. Notice that other analysis can be conduced on similar metrics. For example, instead of considering the length of edges, we can consider the "size of a cell", which would be either the maximum memory gap between all vertices of the cell in the vertex array, or the maximum memory gap between all adjacent cells in the cell array.



(a) Zoom on L1 cache sizes



(b) Zoom on L2 cache sizes

Figure 5: Cumulative distribution function of edge lengths of the torso mesh (10M resolution).

Figure 5 shows for the 10 M cell torso mesh[3] the proportion of edges which length is smaller than the size reported in abscissa. The two graphics are zoomed around the L1 and L2 cache sizes of the tested architectures. We can clearly see on this figure how cache-oblivious layouts (OpenCCL and FastCOL) optimize smaller edge lengths. The original layout appears to be the worst with regards to the edge lengths.

The geometric layout does not particularly optimize smaller edge lengths as opposed to CO layouts, but it exhibits a size for which nearly 100% of the edges fit in cache. Once the size is big enough to contain two entire slices of the mesh in the x direction, all edges will have a size smaller than this one. This has important consequences: the geometric layout will lead to a good performance when this size is smaller than the L2 cache size. Indeed, results show clearly that the geometric layout performs well for small meshes while it is outperformed by the CO layouts for the bigger ones. The other consequence is that the geometric layout is not efficient with regard to the L1 cache. It has indeed been observed (results are detailed omitted). That is why speed-up are generally less important with the geometric layout than with OpenCCL or FastCOL layouts for small meshes.

We also compared our estimation of cache-misses and the number of cache-misses actually observed during our experiments. If we call N the size of a mesh $G = (V,E)$ (in bytes), B the size of a cache line and M the cache size, we estimate the number of cache-misses by:

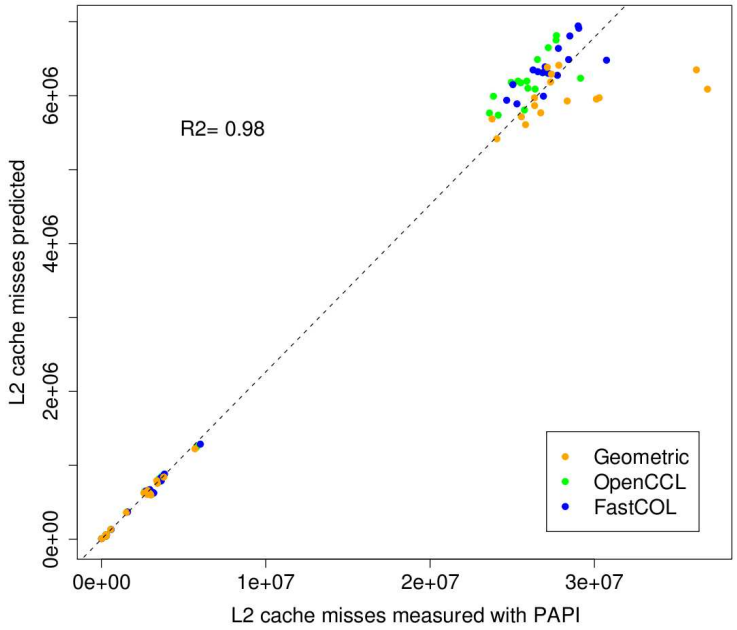
$$ExpectedCM \approx \frac{N}{B} + \sum_{e \in E} \mathbb{1}_{\lambda_e > M}$$

where λ_e is the length of the edge e. That is, we count the number of cache-misses for a linear full read of the data arrays and we add one cache-miss per edge whose length is bigger than the cache size M. The theoretical upper bound $\frac{N}{B} + O(k_M)$ we give for our FastCOL algorithm in theorem 2 is larger because in our algorithm we count one for each edge that goes from one subtree (of size smaller than M) to another one. It is immediate that all edges within a subtree of size smaller than M have a length smaller than M. But some edges between different subtrees counted as bigger than M in our theoretical bound can in fact be smaller.

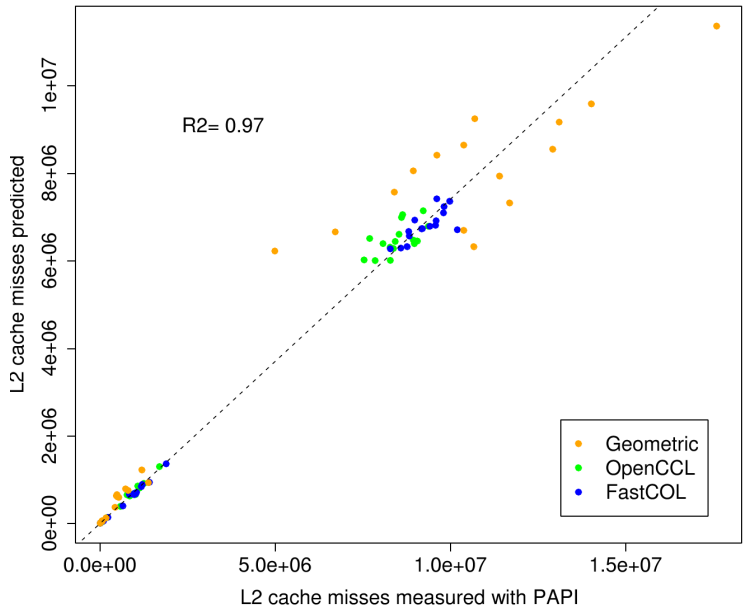
In figure 6, we display the correlation between the expected cache-misses for the considered mesh layout and the cache-misses observed with the PAPI software on the two architectures we used. Note that the factor $\frac{N}{B}$ has been subtracted for this measure. Otherwise, it would have been difficult to easily compare meshes of different N sizes. The correlation between expected cache-misses and real ones is very high with a calculated r^2 of 0.98.

We want to highlight the effect of using the structure built by our algorithm. Indeed, the BSP tree computed by our algorithm can efficiently be reused by the filters that needs such BSP tree. Table 3 illustrates this fact. In this experiment (on CPU and GPU), we compute an isosurface using a min-max tree to quickly reject some parts of the meshes. The two first experiments use a kd-tree with the OpenCCL and FastCOL layouts. The CO layouts lead to a speed-up even if the tree does not partition the meshes like the layouts. However, if in the isosurface algorithm we use the BSP tree computed by our FastCOL algorithm, the efficiency is greatly improved.

[3] The other meshes produce similar graphs.



(a) Core2



(b) Opteron

Figure 6: Correlation between edge lengths and measured L2 cache-misses on cpuiso. Each point corresponds to a mesh with geometric, OpenCCL or FastCOL layouts.

6 Conclusion

We introduced FastCOL, an algorithm relying on the geometric separator from Miller *et al.* [16] for computing CO mesh layouts. FastCOL is a $O(N \log N)$ algorithm computing layouts with a proven quality. This algorithm requires significantly less computation time and memory than OpenCCL, the best known CO mesh layout algorithm [26]. Without modifying the visualization algorithms, it can bring significant performance improvements on CPUs where it reduces the number of cache-misses, as well as on GPU architectures where it favors parallel coalesced data accesses. Performances are comparable to OpenCCL and better by more than 10% if the access pattern is adapted to FastCOL's BSP tree.

Developing efficient algorithms for multicore architectures is challenging regarding the complexity of their memory hierarchy and concurrent processing capabilities. Oblivious approaches are interesting as they bring extra performance while reducing the tuning efforts. Our future work will focus on evaluating the benefits of CO layouts on multicores with classical parallel and processor oblivious algorithms.

References

[1] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Comm. of ACM*, 31(9):1116–1127, 1988.

[2] M. Bader and C. Zenger. Cache oblivious matrix multiplication using an element ordering based on a Peano curve. *Linear Algebra and Its Applications*, 417(2–3):301—-313, 2006.

[3] A. Bogomjakov and C. Gotsman. Universal rendering sequences for transparent vertex caching of progressive meshes. In *GRIN'01*, pages 81–90, 2001.

[4] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *The International Journal of High Performance Computing Applications*, 14:189–204, 2000.

[5] P. Bunyk, A. Kaufman, and C. Silva. Simple, Fast, and Robust Ray Casting of Irregular Grids. *Scientific Visualization Conference, 1997*, pages 30–30, 1997.

[6] S. Callahan, M. Ikits, J. Comba, and C. Silva. Hardware-assisted visibility sorting for unstructured volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 11(3):285–295, May-June 2005.

[7] Y.-J. Chiang and C. Silva. I/O optimal isosurface extraction. *Visualization '97., Proceedings*, pages 293–300, Oct. 1997.

[8] Y.-J. Chiang, C. Silva, and W. Schroeder. Interactive out-of-core isosurface extraction. *Visualization '98. Proceedings*, pages 167–174, Oct. 1998.

[9] Y.-J. Chiang and C. T. Silva. External memory techniques for isosurface extraction in scientific visualization. In *External memory algorithms*, pages 247–277, Boston, MA, USA, 1999. American Mathematical Society.

[10] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Optimal isosurface extraction from irregular volume data. In *VVS '96: Proceedings of the 1996 symposium on Volume visualization*, pages 31–38, Piscataway, NJ, USA, 1996. IEEE Press.

[11] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterated radon points. In *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, pages 91–98, 1993.

[12] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-Oblivious Algorithms. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 285, 1999.

[13] H. Hoppe. Optimization of mesh locality for transparent vertex caching. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 269–276, 1999.

[14] M. Isenburg and P. Lindstrom. Streaming meshes. *Visualization, 2005. VIS 05. IEEE*, pages 231–238, 23-28 Oct. 2005.

[15] U. Meyer, P. Sanders, and J. F. Sibeyn, editors. *Algorithms for Memory Hierarchies, Advanced Lectures [Dagstuhl Research Seminar, March 10-14, 2002]*, volume 2625 of *Lecture Notes in Computer Science*. Springer, 2003.

[16] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Geometric Separators for Finite-Element Meshes. *SIAM J. Sci. Comput.*, 19(2):364–386, 1998.

[17] OpenCCL: Cache-Coherent Layouts. http://www.cs.unc.edu/ geom/COL/OpenCCL/.

[18] V. Pascucci and R. Frank. Global Static Indexing for Real-Time Exploration of Very Large Regular Grids. *Supercomputing, ACM/IEEE 2001 Conference*, pages 45–45, Nov. 2001.

[19] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W.-m. W. Hwu. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *PPoPP '08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 73–82, New York, NY, USA, 2008. ACM.

[20] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics, 3rd ed.* Kitware Inc., 2004.

[21] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: a many-core x86 architecture for visual computing. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–15, New York, NY, USA, 2008. ACM.

[22] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. *SIGGRAPH Comput. Graph.*, 24(5):63–70, 1990.

[23] R. C. Whaley and A. Petitet. Minimizing development and maintenance costs in supporting persistently optimized BLAS. *Software: Practice and Experience*, 35(2):101–121, 2005.

[24] J. Wilhelms and A. Van Gelder. Octrees for faster isosurface generation. *ACM Trans. Graph.*, 11(3):201–227, 1992.

[25] S.-E. Yoon and P. Lindstrom. Mesh Layouts for Block-Based Caches. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1213–1220, 2006.

[26] S.-E. Yoon, P. Lindstrom, V. Pascucci, and D. Manocha. Cache-oblivious mesh layouts. In *ACM SIGGRAPH*, pages 886–893, 2005.

[27] S.-E. Yoon and D. Manocha. Cache-Efficient Layouts of Bounding Volume Hierarchies. *Computer Graphics Forum*, 25(3):507–516, Sept. 2006.

[28] K. Yotov, T. Roeder, K. Pingali, J. Gunnels, and F. Gustavson. An experimental comparison of cache-oblivious and cache-conscious programs. In *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pages 93–104, New York, NY, USA, 2007. ACM.

A Additional Measures

Table 4: Mesh sizes

Meshes	#points	#cells	size (MB)	size w/ scalar (MB)
blunt	40,948	222,414	7.72	8.04
fighter	13,832	70,125	2.46	2.56
spxr1	18,845	99,593	3.47	3.61
spxu1	17,431	100,236	3.46	3.59
spx	2,896	12,936	0.46	0.48
bucky2	262,144	1,250,235	44.15	46.15
fighterr1	169,279	1,025,665	35.18	36.47
fighteru1	174,471	1,057,557	36.27	37.60
flamme2	246,668	1,429,237	49.26	51.14
plasma	274,625	1,310,720	46.29	48.38
post	108,300	616,050	21.28	22.11
sf2	378,747	2,067,739	71.77	74.66
spxr2	180,213	1,030,585	35.58	36.95
spxu2	183,981	1,130,922	38.72	40.13
torso	168,930	1,082,723	36.91	38.20
bucky2r1	1,585,333	9,951,469	339.98	352.08
bucky2u1	1,429,613	9,404,326	319.72	330.63
fighterr2	1,649,525	10,259,164	350.84	363.42
fighteru2	1,642,795	10,300,418	351.94	364.48
flamme2r1	1,564,240	10,039,758	342.19	354.13
flamme2u1	1,624,652	10,087,446	345.03	357.42
plasmar1	1,742,157	10,840,277	370.69	383.99
plasmau1	1,675,932	11,148,401	378.58	391.37
postr1	1,785,085	10,262,122	354.03	367.65
postu1	1,647,891	9,510,317	327.95	340.52
sf1	2,461,694	13,980,162	482.98	501.77
sf2r1	1,610,554	9,588,705	329.49	341.77
sf2u1	1,772,252	10,997,790	376.19	389.71
spxr3	1,671,580	10,126,742	347.30	360.06
spxu3	1,702,853	10,666,566	364.49	377.48
torsor1	1,567,838	10,072,449	343.27	355.23
torsou1	1,547,928	9,822,806	335.20	347.01
bucky2r2	7,493,319	48,070,324	1638.50	1695.67
bucky2u2	7,761,809	49,494,141	1688.10	1747.31
fighterr3	7,638,337	48,193,668	1645.58	1703.86
fighteru3	8,158,153	51,706,388	1764.68	1826.92
flamme2r2	8,009,055	51,703,067	1761.17	1822.27
flamme2u2	8,265,998	52,268,733	1784.31	1847.37
plasmar2	7,450,645	47,849,247	1630.77	1687.62
plasmau2	7,792,881	49,661,879	1693.93	1753.38
postr2	8,209,139	50,520,927	1729.67	1792.30
postu2	8,088,952	50,293,710	1719.98	1781.70
sf1r1	8,252,867	50,126,740	1718.64	1781.60

Meshes	#points	#cells	size (MB)	size w/ scalar (MB)
sf1u1	8,061,785	48,979,286	1679.25	1740.76
sf2r2	8,361,765	51,905,386	1775.41	1839.21
sf2u2	7,766,060	49,185,530	1678.77	1738.02
spxr4	8,031,393	49,897,821	1706.58	1767.86
spxu4	8,335,220	52,716,284	1799.55	1863.14
torsor2	7,831,729	50,444,782	1718.71	1778.46
torsou2	7,551,349	48,097,089	1640.64	1698.26
mean 100K	18,790	101,061	3.51	3.66
mean 1M	182,097	1,106,823	37.82	39.16
mean 10M	1,429,613	9,404,326	319.72	330.63
mean 50M	8,361,765	52,716,284	1799.55	1863.14

Table 5: Isosurfaces

Meshes	Isovalues			
blunt	0.3	1	3	4.8
bucky2	20	100	200	230
fighter	0.5	1	1.5	2.3
flamme2	300	350	425	500
plasma	1.1	1.3	1.5	1.7
post	0	1.6	3	4
sf	1.3	1.6	2	2.6
spx	0.21	0.3	0.6	0.8
torso	-50	0	100	250

Table 6: Build layouts sur Opteron

Meshes	OpenCCL	FastCOL	FastCOL vs OpenCCL
blunt	9.25	3.38	2.74
fighter	3.37	0.77	4.38
spxr1	5.17	1.01	5.12
spxu1	4.40	1.10	4.00
spx	0.49	0.11	4.45
bucky2	66.26	17.42	3.80
fighterr1	42.28	15.00	2.82
fighteru1	41.75	17.04	2.45
flamme2	65.13	18.96	3.44
plasma	68.45	20.72	3.30
post	26.53	9.24	2.87
sf2	98.27	30.70	3.20
spxr2	42.59	16.45	2.59
spxu2	41.72	15.85	2.63
torso	45.54	16.76	2.72
bucky2r1	544.49	200.84	2.71
bucky2u1	476.38	198.17	2.40
fighterr2	580.75	238.67	2.43
fighteru2	587.32	241.17	2.44
flamme2r1	537.84	223.89	2.40
flamme2u1	566.02	246.90	2.29
plasmar1	569.11	236.63	2.41
plasmau1	588.87	227.96	2.58
postr1	564.06	259.20	2.18
postu1	542.77	210.85	2.57
sf1	636.16	230.23	2.76
sf2r1	514.48	194.52	2.64
sf2u1	543.82	190.55	2.85
spxr3	563.27	221.67	2.54
spxu3	587.70	267.79	2.19
torsor1	603.92	297.79	2.03
torsou1	585.91	238.83	2.45
bucky2r2		1511.13	
bucky2u2		1519.59	
fighterr3		1757.53	
fighteru3		1918.77	
flamme2r2		1620.70	
flamme2u2		1716.28	
plasmar2		1488.51	
plasmau2		1572.15	
postr2		1983.04	
postu2		1786.25	
sf1r1		1452.39	
sf1u1		1136.74	
sf2r2		1404.00	

Meshes	OpenCCL	FastCOL	FastCOL vs OpenCCL
sf2u2		1159.84	
spxr4		1572.98	
spxu4		1681.65	
torsor2		1838.19	
torsou2		1753.46	
mean 100K	4.54	1.27	4.14
mean 1M	44.07	16.90	2.84
mean 10M	476.38	190.55	2.03
mean 50M		1983.04	

Table 7: Gradient sur Core2

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.07	1.11E+05	4.33E+03	1.02	0.41	0.93	1.03	1.38	1.02	1.02	1.28	1.14	1.01
fighter	0.36	2.20E+06	2.52E+05	1.01	1.32	1.57	1.01	4.58	1.55	1.03	4.57	1.56	0.98
spxr1	0.52	3.05E+06	2.98E+05	1.02	0.97	1.03	1.03	4.75	1.03	1.03	3.94	1.02	1.00
spxu1	0.51	2.79E+06	3.02E+05	1.00	0.89	1.06	0.98	4.16	1.04	1.01	3.59	1.04	0.96
blunt	1.14	5.17E+06	2.26E+06	1.04	1.97	2.96	1.04	3.33	2.83	1.03	3.31	2.85	1.00
post	3.05	4.35E+06	2.35E+06	1.02	0.58	1.06	0.99	1.05	1.03	0.99	0.94	1.02	1.00
bucky2	6.13	1.06E+07	5.00E+06	0.98	0.88	1.02	1.00	1.24	1.06	1.04	1.14	1.04	0.96
fighterr1	6.26	5.83E+07	3.47E+07	1.19	1.53	9.09	1.17	8.39	9.29	1.21	7.20	9.11	0.97
fighteru1	6.05	4.42E+07	2.87E+07	1.11	1.10	7.79	1.16	4.40	7.40	1.16	5.39	7.23	1.01
flamme2	7.33	1.81E+07	6.83E+06	0.97	0.27	0.53	1.01	1.88	1.28	1.01	1.60	1.26	1.00
plasma	6.04	1.04E+07	5.24E+06	1.00	0.69	1.01	0.99	1.12	1.05	1.01	1.12	1.05	0.98
spxr2	5.69	3.24E+07	1.11E+07	1.07	0.74	2.52	1.09	4.76	3.00	1.09	4.04	2.96	1.00
spxu2	6.21	3.66E+07	1.35E+07	1.08	0.74	2.89	1.08	4.78	3.30	1.09	4.16	3.24	0.99
torso	6.63	5.94E+07	4.52E+07	1.19	1.29	8.02	1.20	7.63	11.46	1.21	6.87	11.17	0.99
sf2	10.56	3.40E+07	1.04E+07	0.96	1.53	1.35	1.02	2.41	1.40	1.04	2.22	1.38	0.98
bucky2r1	52.83	3.17E+08	1.26E+08	1.00	0.93	1.65	1.05	4.55	3.39	1.05	3.95	3.24	1.00
bucky2u1	48.15	2.69E+08	8.20E+07	0.97	1.50	1.65	1.01	4.06	2.35	1.00	3.52	2.24	1.00
fighterr2	66.44	5.10E+08	5.19E+08	1.20	1.04	4.05	1.27	6.78	13.71	1.27	5.86	12.81	1.00
fighteru2	64.05	4.79E+08	4.66E+08	1.14	0.94	3.77	1.20	6.49	12.12	1.22	5.64	11.59	0.98
flamme2r1	56.78	4.18E+08	2.58E+08	0.99	0.75	0.99	1.12	5.98	6.82	1.11	5.02	6.47	1.01

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
flamme2u1	59.89	4.37E+08	3.21E+08	1.08	0.81	1.63	1.16	6.07	8.52	1.17	5.20	8.13	0.99
plasmar1	55.69	3.49E+08	1.35E+08	0.98	0.92	1.86	1.04	4.58	3.31	1.05	3.96	3.17	0.99
plasmau1	56.81	3.45E+08	1.16E+08	1.03	1.81	1.94	1.04	4.34	2.78	1.04	3.73	2.67	1.00
postr1	58.19	4.14E+08	2.75E+08	1.04	0.83	1.71	1.11	6.16	7.44	1.12	5.15	7.06	1.00
postu1	53.72	3.73E+08	2.62E+08	1.04	0.84	2.24	1.11	5.97	7.57	1.11	4.85	7.23	1.00
sf2r1	54.54	4.81E+08	2.30E+08	1.07	1.45	3.95	1.13	7.37	6.63	1.13	6.37	6.29	1.00
sf2u1	59.56	4.59E+08	1.63E+08	1.04	1.33	2.01	1.07	5.61	3.98	1.07	4.92	3.78	1.00
spxr3	59.55	4.70E+08	2.71E+08	1.06	0.89	1.42	1.15	6.73	7.34	1.16	5.51	6.91	1.00
spxu3	59.41	4.56E+08	3.04E+08	1.00	0.77	1.40	1.09	5.94	7.66	1.10	4.80	7.31	1.00
torsor1	71.48	6.81E+08	7.09E+08	1.26	1.28	3.42	1.42	9.71	18.75	1.40	7.83	17.79	1.02
torsou1	71.40	6.85E+08	7.00E+08	1.25	1.37	4.01	1.43	10.01	19.11	1.43	8.14	18.30	1.00
mean 100K	0.52	2.66E+06	6.22E+05	1.02	1.11	1.51	1.01	3.64	1.49	1.02	3.34	1.52	0.99
median 100K	0.51	2.79E+06	2.98E+05	1.02	0.97	1.06	1.03	4.16	1.04	1.03	3.59	1.14	1.00
min 100K	0.07	1.11E+05	4.33E+03	1.00	0.41	0.93	0.98	1.38	1.02	1.01	1.28	1.02	0.96
max 100K	1.14	5.17E+06	2.26E+06	1.04	1.97	2.96	1.04	4.75	2.83	1.03	4.57	2.85	1.01
mean 1M	6.40	3.08E+07	1.63E+07	1.06	0.93	3.53	1.07	3.77	4.03	1.08	3.47	3.94	0.99
median 1M	6.17	3.32E+07	1.08E+07	1.04	0.81	1.93	1.05	3.40	2.20	1.06	3.13	2.17	0.99
min 1M	3.05	4.35E+06	2.35E+06	0.96	0.27	0.53	0.99	1.05	1.03	0.99	0.94	1.02	0.96
max 1M	10.56	5.94E+07	4.52E+07	1.19	1.53	9.09	1.20	8.39	11.46	1.21	7.20	11.17	1.01
mean 10M	59.28	4.46E+08	3.09E+08	1.07	1.09	2.36	1.15	6.27	8.22	1.15	5.28	7.81	1.00
median 10M	58.80	4.47E+08	2.67E+08	1.04	0.94	1.90	1.11	6.03	7.39	1.11	5.09	6.99	1.00
min 10M	48.15	2.69E+08	8.20E+07	0.97	0.75	0.99	1.01	4.06	2.35	1.00	3.52	2.24	0.98
max 10M	71.48	6.85E+08	7.09E+08	1.26	1.81	4.05	1.43	10.01	19.11	1.43	8.14	18.30	1.02

Table 8: Gradient sur Opteron

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.10	2.75E+05	2.07E+04	1.00	1.50	0.53	1.02	2.63	0.31	1.00	2.62	1.02	1.02
fighter	0.56	1.80E+06	4.56E+05	1.07	1.51	1.22	1.03	1.32	3.14	1.09	2.29	2.78	0.94
spxr1	0.77	3.43E+06	3.53E+05	1.03	1.55	1.35	1.03	1.08	1.72	1.03	2.11	1.60	1.00
spxu1	0.77	5.39E+06	6.92E+05	0.99	2.43	2.78	1.03	2.58	1.28	1.02	5.89	3.35	1.01
blunt	1.82	7.03E+06	1.99E+06	1.08	2.04	3.10	1.15	1.67	3.90	1.10	3.47	3.64	1.04
post	4.79	1.33E+07	1.56E+06	1.02	1.71	0.79	1.05	2.36	1.10	1.03	2.33	1.09	1.02
bucky2	9.22	1.70E+07	4.04E+06	0.95	0.30	0.82	0.97	0.86	1.36	0.99	1.51	1.35	0.98
fighterr1	9.86	5.81E+07	2.43E+07	1.17	0.92	2.98	1.28	1.42	11.67	1.30	6.66	11.31	0.98
fighteru1	9.58	3.05E+07	1.90E+07	1.16	1.01	4.63	1.19	0.85	8.42	1.22	3.32	8.21	0.98
flamme2	11.62	3.37E+07	5.44E+06	0.93	0.54	0.33	1.05	1.44	1.56	1.05	1.38	1.78	1.00
plasma	9.49	4.04E+07	3.54E+06	0.97	1.09	0.81	1.02	1.57	1.19	1.02	2.24	1.28	1.00
spxr2	8.74	3.36E+07	1.03E+07	1.05	1.00	1.41	1.12	3.86	3.95	1.13	3.64	4.74	0.99
spxu2	9.38	4.14E+07	1.13E+07	1.04	1.17	1.32	1.13	4.99	5.15	1.11	5.06	4.65	1.02
torso	10.48	3.70E+07	2.78E+07	1.17	0.71	3.08	1.29	1.79	12.08	1.28	3.77	10.97	1.01
sf2	15.99	5.15E+07	7.72E+06	1.03	1.33	1.32	1.03	0.61	1.86	1.04	3.12	1.81	0.99
bucky2r1	86.59	5.49E+08	1.46E+08	1.02	1.20	1.78	1.17	7.30	7.08	1.18	6.04	6.59	0.99
bucky2u1	75.22	3.34E+08	6.64E+07	1.01	2.36	1.27	1.05	4.53	3.45	1.09	4.14	3.13	0.97
fighterr2	103.27	4.29E+08	2.33E+08	1.16	0.95	1.84	1.29	2.09	10.96	1.36	4.77	9.94	0.95
fighteru2	100.64	4.52E+08	2.06E+08	1.12	0.71	1.50	1.29	2.72	9.57	1.30	4.57	8.89	0.99
flamme2r1	90.76	5.63E+08	1.54E+08	0.96	1.14	0.77	1.17	5.71	7.17	1.20	5.69	6.54	0.97
flamme2u1	93.60	4.76E+08	1.67E+08	1.01	1.01	0.92	1.22	5.98	7.12	1.22	4.96	6.71	1.00
plasmar1	89.63	2.72E+08	1.05E+08	1.01	0.79	1.35	1.14	2.66	4.71	1.12	2.75	4.34	1.01

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	88.82	2.52E+08	9.01E+07	1.06	1.65	1.63	1.06	1.07	3.89	1.09	2.63	3.65	0.97
postr1	90.58	3.37E+08	1.64E+08	1.02	0.92	1.21	1.16	3.51	7.65	1.17	2.91	7.14	0.99
postu1	85.59	2.92E+08	1.50E+08	1.05	0.46	1.35	1.19	3.97	6.42	1.22	3.26	6.79	0.98
sf2r1	84.94	4.96E+08	1.27E+08	1.08	2.35	2.11	1.20	1.80	6.43	1.20	6.03	5.88	1.00
sf2u1	92.92	5.31E+08	1.07E+08	1.03	0.97	0.80	1.13	5.85	4.72	1.14	5.30	4.27	0.99
spxr3	90.43	5.97E+08	1.53E+08	0.99	0.98	0.92	1.17	1.62	7.39	1.20	5.35	6.60	0.98
spxu3	95.73	5.00E+08	1.67E+08	0.97	0.91	0.85	1.23	1.69	7.56	1.22	4.93	6.70	1.00
torsor1	110.36	3.65E+08	3.09E+08	1.20	0.92	1.77	1.42	4.61	14.93	1.49	3.51	13.15	0.95
torsou1	107.34	4.78E+08	3.07E+08	1.24	1.32	1.94	1.49	1.24	15.14	1.49	5.45	13.70	1.00
mean 100K	0.80	3.59E+06	7.03E+05	1.03	1.81	1.80	1.05	1.86	2.07	1.05	3.27	2.48	1.00
median 100K	0.77	3.43E+06	4.56E+05	1.03	1.55	1.35	1.03	1.67	1.72	1.03	2.62	2.78	1.01
min 100K	0.10	2.75E+05	2.07E+04	0.99	1.50	0.53	1.02	1.08	0.31	1.00	2.11	1.02	0.94
max 100K	1.82	7.03E+06	1.99E+06	1.08	2.43	3.10	1.15	2.63	3.90	1.10	5.89	3.64	1.04
mean 1M	9.92	3.57E+07	1.15E+07	1.05	0.98	1.75	1.11	1.97	4.83	1.12	3.30	4.72	1.00
median 1M	9.54	3.54E+07	8.99E+06	1.04	1.00	1.32	1.09	1.50	2.90	1.08	3.22	3.23	1.00
min 1M	4.79	1.33E+07	1.56E+06	0.93	0.30	0.33	0.97	0.61	1.10	0.99	1.38	1.09	0.98
max 1M	15.99	5.81E+07	2.78E+07	1.17	1.71	4.63	1.29	4.99	12.08	1.30	6.66	11.31	1.02
mean 10M	92.90	4.33E+08	1.66E+08	1.06	1.16	1.37	1.21	3.52	7.76	1.23	4.52	7.13	0.98
median 10M	90.67	4.64E+08	1.53E+08	1.03	0.97	1.35	1.18	3.11	7.15	1.20	4.85	6.65	0.99
min 10M	75.22	2.52E+08	6.64E+07	0.96	0.46	0.77	1.05	1.07	3.45	1.09	2.63	3.13	0.95
max 10M	110.36	5.97E+08	3.09E+08	1.24	2.36	2.11	1.49	7.30	15.14	1.49	6.04	13.70	1.01

Table 9: Connect sur Core2

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.02	5.11E+05	4.69E+04	0.95	0.83	0.93	1.00	1.05	1.04	1.01	1.06	1.23	0.99
fighter	0.12	4.52E+06	2.43E+06	0.89	1.15	0.90	1.17	1.61	1.26	1.17	1.61	1.26	1.00
spxr1	0.14	5.27E+06	3.02E+06	0.94	0.88	0.88	1.08	1.34	1.12	1.05	1.30	1.10	1.02
spxu1	0.15	5.55E+06	3.38E+06	0.93	0.90	0.92	1.17	1.35	1.25	1.15	1.33	1.27	1.02
blunt	0.36	1.24E+07	8.93E+06	1.04	1.08	1.09	1.18	1.30	1.17	1.18	1.30	1.18	1.00
post	0.76	2.28E+07	1.99E+07	0.88	0.77	0.84	0.99	0.96	1.00	0.97	0.94	0.94	1.02
bucky2	1.75	5.19E+07	4.62E+07	0.94	0.89	0.93	0.99	0.99	1.02	1.03	0.97	1.00	0.96
fighterr1	2.12	9.50E+07	5.58E+07	1.12	1.31	1.12	1.37	2.13	1.48	1.47	2.13	1.48	0.93
fighteru1	1.90	7.53E+07	5.08E+07	1.09	1.01	1.04	1.32	1.68	1.34	1.35	1.67	1.35	0.98
flamme2	2.09	6.54E+07	5.33E+07	0.80	0.57	0.76	1.08	1.10	1.05	1.07	1.08	1.05	1.01
plasma	1.83	5.42E+07	4.79E+07	0.93	0.80	0.92	1.03	0.98	1.01	1.02	0.99	1.01	1.01
spxr2	1.62	6.00E+07	4.38E+07	0.90	0.78	0.90	1.15	1.38	1.20	1.15	1.39	1.22	1.00
spxu2	1.80	6.71E+07	4.85E+07	0.92	0.77	0.91	1.18	1.41	1.22	1.20	1.41	1.24	0.98
torso	2.32	9.47E+07	5.83E+07	1.13	1.11	1.09	1.42	1.98	1.44	1.51	2.01	1.48	0.94
sf2	2.84	9.81E+07	7.23E+07	0.99	1.07	0.97	1.09	1.26	1.09	1.11	1.26	1.10	0.98
bucky2r1	16.49	6.29E+08	4.79E+08	0.95	0.86	1.00	1.21	1.49	1.32	1.25	1.49	1.34	0.97
bucky2u1	16.28	6.25E+08	4.91E+08	1.04	1.04	1.10	1.27	1.53	1.39	1.28	1.52	1.39	0.99
fighterr2	24.29	8.24E+08	6.38E+08	1.26	0.95	1.21	1.71	1.89	1.72	1.76	1.86	1.71	0.97
fighteru2	20.08	7.63E+08	5.43E+08	1.10	0.87	1.04	1.51	1.77	1.48	1.51	1.75	1.48	1.00
flamme2r1	18.43	7.34E+08	5.09E+08	0.91	0.77	0.92	1.32	1.70	1.38	1.35	1.67	1.37	0.98
flamme2u1	17.77	6.93E+08	4.97E+08	0.94	0.76	0.94	1.34	1.64	1.37	1.34	1.61	1.37	1.00
plasmar1	18.34	7.12E+08	5.44E+08	1.00	0.94	1.04	1.29	1.57	1.40	1.29	1.56	1.41	1.00

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	19.75	7.49E+08	5.96E+08	1.03	1.02	1.06	1.28	1.52	1.39	1.28	1.50	1.39	1.00
postr1	18.50	6.87E+08	4.98E+08	0.97	0.81	0.95	1.33	1.63	1.38	1.31	1.61	1.37	1.01
postu1	15.97	6.24E+08	4.48E+08	0.96	0.82	0.96	1.28	1.62	1.36	1.28	1.60	1.35	1.00
sf2r1	19.04	7.91E+08	5.16E+08	1.22	1.26	1.19	1.46	1.98	1.50	1.49	1.96	1.51	0.98
sf2u1	19.62	8.58E+08	5.57E+08	1.12	1.14	1.13	1.37	1.86	1.42	1.38	1.84	1.41	0.99
spxr3	17.10	6.57E+08	4.56E+08	0.90	0.73	0.87	1.23	1.53	1.26	1.22	1.51	1.24	1.01
spxu3	17.46	6.92E+08	4.79E+08	0.88	0.71	0.86	1.25	1.55	1.26	1.27	1.53	1.27	0.99
torsor1	29.31	1.08E+09	7.82E+08	1.54	1.17	1.47	2.10	2.47	2.09	2.14	2.44	2.09	0.98
torsou1	28.29	1.05E+09	7.63E+08	1.57	1.20	1.50	2.23	2.55	2.17	2.19	2.49	2.15	1.02
mean 100K	0.16	5.64E+06	3.56E+06	0.95	0.97	0.94	1.12	1.33	1.17	1.11	1.32	1.21	1.01
median 100K	0.14	5.27E+06	3.02E+06	0.94	0.90	0.92	1.17	1.34	1.17	1.15	1.30	1.23	1.00
min 100K	0.02	5.11E+05	4.69E+04	0.89	0.83	0.88	1.00	1.05	1.04	1.01	1.06	1.10	0.99
max 100K	0.36	1.24E+07	8.93E+06	1.04	1.15	1.09	1.18	1.61	1.26	1.18	1.61	1.27	1.02
mean 1M	1.90	6.84E+07	4.97E+07	0.97	0.91	0.95	1.16	1.39	1.19	1.19	1.39	1.19	0.98
median 1M	1.86	6.62E+07	4.96E+07	0.93	0.84	0.92	1.12	1.32	1.15	1.13	1.33	1.16	0.98
min 1M	0.76	2.28E+07	1.99E+07	0.80	0.57	0.76	0.99	0.96	1.00	0.97	0.94	0.94	0.93
max 1M	2.84	9.81E+07	7.23E+07	1.13	1.31	1.12	1.42	2.13	1.48	1.51	2.13	1.48	1.02
mean 10M	19.79	7.60E+08	5.50E+08	1.09	0.94	1.08	1.45	1.77	1.49	1.46	1.75	1.49	0.99
median 10M	18.47	7.23E+08	5.13E+08	1.01	0.90	1.04	1.32	1.64	1.39	1.32	1.61	1.39	1.00
min 10M	15.97	6.24E+08	4.48E+08	0.88	0.71	0.86	1.21	1.49	1.26	1.22	1.49	1.24	0.97
max 10M	29.31	1.08E+09	7.82E+08	1.57	1.26	1.50	2.23	2.55	2.17	2.19	2.49	2.15	1.02

Table 10: Connect sur Opteron

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.03	5.68E+05	1.73E+05	0.91	0.88	0.79	0.97	1.02	0.92	1.01	1.01	0.98	0.96
fighter	0.24	5.18E+06	1.51E+06	0.90	0.80	0.81	1.16	1.31	1.35	1.16	1.31	1.35	1.00
spxr1	0.30	5.79E+06	1.56E+06	0.96	0.93	0.88	1.08	1.17	1.21	1.05	1.09	1.15	1.03
spxu1	0.31	6.69E+06	1.67E+06	0.96	0.94	0.87	1.17	1.34	1.46	1.16	1.34	1.37	1.01
blunt	0.75	1.53E+07	3.71E+06	1.05	1.01	1.07	1.18	1.26	1.40	1.18	1.27	1.41	0.99
post	1.60	3.15E+07	6.21E+06	0.93	0.97	0.84	1.01	1.03	0.97	1.00	1.04	1.00	1.01
bucky2	3.66	7.31E+07	1.39E+07	0.94	0.93	0.83	0.99	1.03	0.95	1.03	1.14	1.02	0.96
fighterr1	4.47	8.59E+07	2.91E+07	1.19	1.01	1.31	1.42	1.25	1.95	1.53	1.54	2.31	0.93
fighteru1	3.90	7.50E+07	2.21E+07	1.14	1.04	1.27	1.34	1.34	1.85	1.37	1.41	2.04	0.98
flamme2	4.19	7.88E+07	1.99E+07	0.82	0.69	0.68	1.08	1.00	1.21	1.07	1.04	1.19	1.01
plasma	3.83	8.69E+07	1.52E+07	0.91	0.66	0.84	1.05	1.29	1.12	1.04	1.27	1.07	1.01
spxr2	3.25	6.71E+07	1.58E+07	0.93	0.88	0.84	1.14	1.19	1.41	1.14	1.22	1.41	1.00
spxu2	3.59	7.19E+07	1.77E+07	0.94	0.84	0.87	1.17	1.22	1.43	1.19	1.33	1.54	0.98
torso	4.79	9.52E+07	3.13E+07	1.18	1.01	1.31	1.48	1.36	2.11	1.54	1.63	2.42	0.96
sf2	5.75	1.00E+08	2.53E+07	0.99	0.92	0.96	1.10	1.16	1.21	1.11	1.17	1.29	0.99
bucky2r1	34.10	5.88E+08	1.65E+08	0.95	0.92	0.92	1.22	1.16	1.52	1.26	1.30	1.64	0.97
bucky2u1	34.55	6.16E+08	1.82E+08	1.04	1.03	1.08	1.27	1.28	1.63	1.28	1.28	1.73	0.99
fighterr2	45.26	7.11E+08	2.48E+08	1.14	0.90	1.20	1.55	1.35	2.12	1.59	1.50	2.31	0.97
fighteru2	38.90	6.25E+08	1.93E+08	1.02	0.87	1.02	1.43	1.42	2.00	1.42	1.43	2.02	1.00
flamme2r1	37.98	6.94E+08	2.02E+08	0.88	0.85	0.84	1.33	1.35	1.77	1.36	1.48	1.89	0.98
flamme2u1	36.10	6.08E+08	1.78E+08	0.90	0.81	0.84	1.33	1.34	1.79	1.32	1.33	1.77	1.00
plasmar1	38.06	6.58E+08	1.86E+08	0.98	0.97	0.96	1.29	1.34	1.73	1.29	1.34	1.74	1.00

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	40.57	7.04E+08	2.04E+08	1.02	1.03	0.97	1.26	1.33	1.64	1.25	1.32	1.63	1.00
postr1	36.46	6.77E+08	1.89E+08	0.92	0.87	0.91	1.29	1.31	1.63	1.28	1.30	1.63	1.01
postu1	31.77	5.58E+08	1.61E+08	0.93	0.84	0.91	1.26	1.33	1.63	1.26	1.32	1.68	1.01
sf2r1	38.48	7.18E+08	2.16E+08	1.19	1.24	1.42	1.45	1.47	2.05	1.48	1.64	2.15	0.98
sf2u1	39.64	7.07E+08	2.04E+08	1.07	1.09	1.20	1.36	1.52	1.97	1.35	1.51	1.96	1.00
spxr3	34.07	6.46E+08	1.62E+08	0.87	0.83	0.77	1.20	1.24	1.46	1.20	1.23	1.45	1.01
spxu3	35.04	6.11E+08	1.60E+08	0.84	0.76	0.73	1.23	1.28	1.60	1.24	1.36	1.70	0.99
torsor1	55.11	7.89E+08	3.26E+08	1.37	1.04	1.53	1.94	1.53	2.96	1.97	1.69	3.08	0.98
torsou1	53.29	7.16E+08	3.14E+08	1.40	0.98	1.60	2.04	1.73	3.41	2.00	1.60	3.28	1.02
mean 100K	0.33	6.70E+06	1.72E+06	0.95	0.91	0.88	1.11	1.22	1.27	1.11	1.20	1.25	1.00
median 100K	0.30	5.79E+06	1.56E+06	0.96	0.93	0.87	1.16	1.26	1.35	1.16	1.27	1.35	1.00
min 100K	0.03	5.68E+05	1.73E+05	0.90	0.80	0.79	0.97	1.02	0.92	1.01	1.01	0.98	0.96
max 100K	0.75	1.53E+07	3.71E+06	1.05	1.01	1.07	1.18	1.34	1.46	1.18	1.34	1.41	1.03
mean 1M	3.90	7.66E+07	1.96E+07	1.00	0.89	0.97	1.18	1.19	1.42	1.20	1.28	1.53	0.98
median 1M	3.87	7.69E+07	1.88E+07	0.94	0.93	0.86	1.12	1.20	1.31	1.13	1.24	1.35	0.98
min 1M	1.60	3.15E+07	6.21E+06	0.82	0.66	0.68	0.99	1.00	0.95	1.00	1.04	1.00	0.93
max 1M	5.75	1.00E+08	3.13E+07	1.19	1.04	1.31	1.48	1.36	2.11	1.54	1.63	2.42	1.01
mean 10M	39.34	6.64E+08	2.06E+08	1.03	0.94	1.06	1.40	1.37	1.93	1.41	1.41	1.98	0.99
median 10M	38.02	6.68E+08	1.91E+08	1.00	0.91	0.96	1.31	1.34	1.75	1.31	1.35	1.76	1.00
min 10M	31.77	5.58E+08	1.60E+08	0.84	0.76	0.73	1.20	1.16	1.46	1.20	1.23	1.45	0.97
max 10M	55.11	7.89E+08	3.26E+08	1.40	1.24	1.60	2.04	1.73	3.41	2.00	1.69	3.28	1.02

Table 11: RC sur Core2

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	2.14	2.95E+07	1.91E+06	0.93	0.87	0.54	0.99	1.07	0.93	1.08	1.05	0.80	0.92
fighter	2.61	6.34E+07	1.14E+07	0.94	1.20	1.03	1.06	1.40	1.65	1.03	1.38	1.69	1.03
spxr1	3.11	9.19E+07	8.60E+06	0.95	1.01	0.63	1.07	1.30	1.03	0.99	1.29	0.98	1.08
spxu1	3.70	1.09E+08	1.55E+07	1.00	1.02	0.84	1.04	1.31	1.24	1.10	1.33	1.23	0.95
blunt	5.29	1.92E+08	1.69E+07	1.07	1.41	2.34	1.08	1.21	2.15	1.09	1.33	2.35	0.99
post	6.92	2.20E+08	2.03E+07	1.00	1.04	1.03	0.95	0.85	0.89	1.00	0.99	1.04	0.96
bucky2	16.07	5.93E+08	1.55E+08	0.98	1.26	0.86	1.01	1.19	0.93	1.01	1.15	0.95	1.00
fighterr1	9.25	3.24E+08	1.78E+08	1.29	1.14	1.86	1.53	1.89	3.29	1.54	1.82	3.23	0.99
fighteru1	12.45	4.53E+08	1.97E+08	1.06	1.00	1.16	1.25	1.52	1.88	1.24	1.48	1.88	1.00
flamme2	8.74	2.58E+08	1.05E+08	0.72	0.63	0.45	1.10	1.18	1.43	1.09	1.14	1.38	1.01
plasma	15.82	5.81E+08	1.64E+08	0.95	1.19	0.88	1.00	1.16	0.95	1.01	1.19	0.97	0.99
spxr2	6.81	2.50E+08	1.00E+08	0.89	0.88	0.77	1.18	1.53	1.83	1.17	1.47	1.79	1.01
spxu2	9.10	3.22E+08	1.41E+08	0.88	0.88	0.76	1.16	1.47	1.64	1.16	1.44	1.61	1.01
torso	10.83	3.83E+08	2.15E+08	1.23	1.14	1.52	1.60	1.88	3.27	1.59	1.82	3.28	1.00
sf2	15.84	5.77E+08	2.82E+08	1.13	1.34	1.43	1.24	1.45	1.86	1.23	1.40	1.83	1.01
bucky2r1	153.56	6.36E+09	4.46E+09	0.71	0.95	0.68	2.42	1.40	3.86	2.35	1.44	3.72	1.03
bucky2u1	146.40	5.99E+09	4.22E+09	0.98	1.09	1.05	2.31	1.42	3.83	2.26	1.46	3.77	1.02
fighterr2	158.34	5.36E+09	4.88E+09	0.61	0.75	0.50	3.41	1.32	4.99	3.33	1.30	5.08	1.03
fighteru2	162.84	5.88E+09	4.93E+09	0.60	0.77	0.49	2.89	1.36	4.18	2.85	1.34	4.36	1.01
flamme2r1	162.26	5.80E+09	4.90E+09	0.56	0.81	0.47	3.38	1.48	4.98	3.32	1.45	5.19	1.02
flamme2u1	155.88	5.83E+09	4.47E+09	0.54	0.76	0.42	2.56	1.35	3.71	2.53	1.33	3.84	1.01
plasmar1	189.39	7.59E+09	5.60E+09	0.81	0.98	0.79	2.58	1.39	4.11	2.50	1.42	3.97	1.03

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	200.36	8.05E+09	5.92E+09	1.03	1.11	1.14	2.58	1.40	4.17	2.51	1.42	3.98	1.03
postr1	158.74	6.00E+09	4.66E+09	0.60	0.85	0.55	3.74	1.43	5.66	3.56	1.49	5.09	1.05
postu1	133.67	5.11E+09	3.85E+09	0.60	0.82	0.54	3.09	1.38	4.57	2.98	1.61	4.23	1.04
sf2r1	206.74	5.81E+09	6.13E+09	1.25	1.04	1.19	4.40	1.51	6.90	4.27	1.57	6.58	1.03
sf2u1	255.42	7.32E+09	7.95E+09	1.13	0.98	1.12	4.00	1.40	6.30	3.91	1.43	6.11	1.02
spxr3	123.54	5.02E+09	3.71E+09	0.46	0.71	0.44	2.77	1.23	4.11	2.66	1.26	3.88	1.04
spxu3	136.99	5.66E+09	4.06E+09	0.45	0.71	0.42	2.49	1.26	3.61	2.48	1.25	3.55	1.00
torsor1	235.95	6.35E+09	7.31E+09	0.87	0.91	0.86	5.14	1.56	7.96	5.00	1.61	7.72	1.03
torsou1	236.33	6.48E+09	7.30E+09	0.93	0.94	0.93	4.76	1.63	7.42	4.69	1.68	7.24	1.02
mean 100K	3.37	9.72E+07	1.09E+07	0.98	1.10	1.08	1.05	1.26	1.40	1.06	1.28	1.41	0.99
median 100K	3.11	9.19E+07	1.14E+07	0.95	1.02	0.84	1.06	1.30	1.24	1.08	1.33	1.23	0.99
min 100K	2.14	2.95E+07	1.91E+06	0.93	0.87	0.54	0.99	1.07	0.93	0.99	1.05	0.80	0.92
max 100K	5.29	1.92E+08	1.69E+07	1.07	1.41	2.34	1.08	1.40	2.15	1.10	1.38	2.35	1.08
mean 1M	11.18	3.96E+08	1.56E+08	1.01	1.05	1.07	1.20	1.41	1.80	1.20	1.39	1.80	1.00
median 1M	10.04	3.54E+08	1.59E+08	0.99	1.09	0.95	1.17	1.46	1.73	1.17	1.42	1.70	1.00
min 1M	6.81	2.20E+08	2.03E+07	0.72	0.63	0.45	0.95	0.85	0.89	1.00	0.99	0.95	0.96
max 1M	16.07	5.93E+08	2.82E+08	1.29	1.34	1.86	1.60	1.89	3.29	1.59	1.82	3.28	1.01
mean 10M	176.02	6.16E+09	5.27E+09	0.76	0.89	0.72	3.28	1.41	5.02	3.20	1.44	4.89	1.03
median 10M	160.50	5.94E+09	4.89E+09	0.66	0.88	0.62	2.99	1.40	4.38	2.91	1.43	4.30	1.03
min 10M	123.54	5.02E+09	3.71E+09	0.45	0.71	0.42	2.31	1.23	3.61	2.26	1.25	3.55	1.00
max 10M	255.42	8.05E+09	7.95E+09	1.25	1.11	1.19	5.14	1.63	7.96	5.00	1.68	7.72	1.05

Table 12: RC sur Opteron

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	2.55	1.31E+07	4.18E+06	1.03	0.94	1.00	1.05	1.01	0.93	1.05	1.00	1.23	1.00
fighter	3.53	2.56E+07	8.87E+06	0.98	1.06	1.16	1.10	1.24	1.34	1.07	1.17	1.19	1.02
spxr1	4.08	3.84E+07	1.06E+07	0.98	0.99	0.98	1.07	1.22	1.38	1.06	1.21	1.20	1.01
spxu1	4.89	6.14E+07	1.43E+07	0.99	1.01	0.94	1.08	1.24	1.25	1.09	1.56	1.44	0.99
blunt	7.64	1.15E+08	1.84E+07	1.10	1.57	1.41	1.10	1.20	1.35	1.11	1.82	1.65	0.99
post	10.12	8.07E+07	2.37E+07	0.96	0.85	0.90	0.98	0.74	1.02	0.99	0.90	1.08	0.99
bucky2	24.51	3.13E+08	8.93E+07	1.00	1.17	0.94	1.08	1.13	1.14	1.03	1.15	1.06	1.05
fighterr1	16.86	1.89E+08	9.38E+07	1.26	1.00	1.25	1.73	1.48	2.29	1.73	1.90	2.64	1.00
fighteru1	20.13	2.88E+08	9.40E+07	1.02	0.94	0.91	1.40	1.26	1.73	1.40	1.24	1.71	1.00
flamme2	15.11	1.50E+08	6.32E+07	0.68	0.67	0.50	1.17	1.19	1.26	1.16	1.14	1.34	1.00
plasma	23.70	4.40E+08	8.93E+07	0.97	1.17	0.92	1.02	1.19	1.00	1.03	1.18	1.02	0.98
spxr2	12.30	1.41E+08	5.48E+07	0.85	0.88	0.69	1.23	1.40	1.37	1.22	1.37	1.41	1.01
spxu2	16.19	1.85E+08	7.11E+07	0.85	0.90	0.67	1.21	1.36	1.30	1.21	1.33	1.32	1.00
torso	18.85	1.96E+08	9.51E+07	1.19	1.02	1.14	1.74	1.50	2.52	1.73	1.46	2.52	1.00
sf2	25.98	3.08E+08	1.39E+08	1.14	1.24	1.20	1.30	1.42	1.64	1.30	1.42	1.63	1.01
bucky2r1	257.59	4.02E+09	2.06E+09	0.74	0.94	0.65	2.63	1.63	3.16	2.46	1.59	2.97	1.07
bucky2u1	244.17	3.95E+09	2.16E+09	0.93	1.03	0.84	2.43	1.45	3.49	2.39	1.42	3.45	1.02
fighterr2	262.97	3.45E+09	2.10E+09	0.64	0.77	0.54	3.02	1.38	3.30	3.14	1.24	3.38	0.96
fighteru2	279.99	3.55E+09	2.27E+09	0.62	0.76	0.56	2.94	1.17	3.30	2.91	1.19	3.36	1.01
flamme2r1	281.11	3.66E+09	2.30E+09	0.61	0.84	0.61	3.71	1.73	4.33	3.51	1.72	3.94	1.06
flamme2u1	278.41	3.86E+09	2.22E+09	0.57	0.85	0.57	3.04	1.62	3.67	2.81	1.51	3.32	1.08
plasmar1	322.34	4.98E+09	2.80E+09	0.85	0.98	0.79	2.69	1.37	3.53	2.74	1.71	3.55	0.98

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	339.02	5.27E+09	3.08E+09	0.99	1.03	0.88	2.70	1.39	3.76	2.75	1.74	3.77	0.98
postr1	296.76	3.85E+09	2.53E+09	0.68	0.87	0.65	3.94	1.40	4.52	4.02	1.82	4.44	0.98
postu1	272.08	3.24E+09	2.15E+09	0.75	0.83	0.63	3.70	1.34	4.07	3.75	1.73	3.98	0.99
sf2r1	331.69	3.40E+09	2.59E+09	1.24	0.94	1.00	4.14	1.37	4.65	4.06	1.35	4.47	1.02
sf2u1	750.96	4.50E+09	3.40E+09	2.08	0.91	1.00	3.70	1.29	4.47	6.77	1.30	4.33	0.55
spxr3	237.18	3.19E+09	2.05E+09	0.55	0.73	0.53	2.97	1.17	3.49	2.88	1.16	3.33	1.03
spxu3	259.91	3.42E+09	2.11E+09	0.53	0.70	0.49	2.78	1.10	3.08	2.74	1.11	3.00	1.02
torsor1	375.32	3.78E+09	2.85E+09	0.86	0.88	0.75	4.69	1.39	4.89	4.61	1.56	4.77	1.02
torsou1	375.59	3.71E+09	2.80E+09	0.92	0.86	0.76	4.50	1.41	4.73	4.44	1.51	4.64	1.01
mean 100K	4.54	5.07E+07	1.13E+07	1.01	1.11	1.10	1.08	1.18	1.25	1.08	1.35	1.34	1.00
median 100K	4.08	3.84E+07	1.06E+07	0.99	1.01	1.00	1.08	1.22	1.34	1.07	1.21	1.23	1.00
min 100K	2.55	1.31E+07	4.18E+06	0.98	0.94	0.94	1.05	1.01	0.93	1.05	1.00	1.19	0.99
max 100K	7.64	1.15E+08	1.84E+07	1.10	1.57	1.41	1.10	1.24	1.38	1.11	1.82	1.65	1.02
mean 1M	18.38	2.29E+08	8.14E+07	0.99	0.98	0.91	1.29	1.27	1.53	1.28	1.31	1.57	1.00
median 1M	17.85	1.92E+08	8.93E+07	0.98	0.97	0.91	1.22	1.31	1.34	1.22	1.29	1.37	1.00
min 1M	10.12	8.07E+07	2.37E+07	0.68	0.67	0.50	0.98	0.74	1.00	0.99	0.90	1.02	0.98
max 1M	25.98	4.40E+08	1.39E+08	1.26	1.24	1.25	1.74	1.50	2.52	1.73	1.90	2.64	1.05
mean 10M	322.82	3.86E+09	2.47E+09	0.85	0.87	0.70	3.35	1.39	3.90	3.50	1.48	3.79	0.99
median 10M	280.55	3.74E+09	2.29E+09	0.75	0.87	0.65	3.03	1.39	3.72	3.02	1.51	3.66	1.02
min 10M	237.18	3.19E+09	2.05E+09	0.53	0.70	0.49	2.43	1.10	3.08	2.39	1.11	2.97	0.55
max 10M	750.96	5.27E+09	3.40E+09	2.08	1.03	1.00	4.69	1.73	4.89	6.77	1.82	4.77	1.08

Table 13: PT sur Core2

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.80	2.42E+05	1.11E+05	1.03	0.84	1.00	0.98	1.00	1.00	1.02	1.00	1.01	0.97
fighter	0.71	1.75E+06	4.67E+05	1.28	1.20	1.00	0.96	1.84	1.38	1.31	1.84	1.40	0.73
spxr1	0.75	1.94E+06	5.40E+05	1.00	0.86	0.72	1.01	1.60	1.17	1.39	1.55	1.17	0.72
spxu1	0.55	1.98E+06	5.47E+05	1.01	0.88	0.74	0.72	1.59	1.19	1.02	1.56	1.18	0.70
blunt	0.79	5.42E+06	1.19E+06	0.96	1.14	0.66	1.00	1.96	1.15	1.01	1.97	1.14	1.00
post	0.76	1.28E+07	3.73E+06	0.94	0.92	0.59	1.03	1.55	1.28	1.03	1.41	1.24	1.00
bucky2	1.38	1.24E+07	5.36E+06	0.83	0.42	0.34	0.96	0.71	0.88	0.96	0.78	0.92	1.00
fighterr1	1.06	4.20E+07	1.15E+07	1.02	1.36	1.01	1.26	2.39	2.33	1.24	2.31	2.25	1.01
fighteru1	1.04	3.72E+07	1.02E+07	0.97	1.12	0.84	1.18	2.01	2.07	1.20	1.95	2.10	0.98
flamme2	1.25	3.25E+07	1.06E+07	0.82	0.58	0.52	1.04	1.23	1.29	1.05	1.21	1.27	0.99
plasma	1.46	1.30E+07	5.64E+06	0.84	0.42	0.33	0.96	0.71	0.86	0.97	0.77	0.91	0.99
spxr2	0.92	2.86E+07	6.76E+06	0.89	0.80	0.61	1.09	1.72	1.47	1.10	1.66	1.47	0.99
spxu2	1.00	3.23E+07	7.62E+06	0.89	0.80	0.63	1.11	1.71	1.53	1.11	1.65	1.53	1.00
torso	1.07	4.25E+07	1.06E+07	1.01	1.12	0.91	1.24	2.47	2.26	1.25	2.37	2.25	1.00
sf2	2.56	3.46E+07	1.69E+07	0.93	0.81	0.64	1.03	1.20	1.16	1.04	1.23	1.24	0.99
bucky2r1	9.48	2.59E+08	7.62E+07	0.72	0.72	0.29	1.02	1.45	0.95	1.01	1.41	0.95	1.00
bucky2u1	10.43	2.22E+08	7.09E+07	0.75	0.79	0.27	0.99	1.32	0.93	1.00	1.30	0.95	0.99
fighterr2	15.74	4.35E+08	4.32E+08	1.18	0.99	1.40	1.66	2.10	3.67	1.66	2.05	3.49	1.00
fighteru2	14.68	4.36E+08	3.80E+08	1.04	0.94	1.16	1.57	2.08	3.33	1.55	2.02	3.30	1.01
flamme2r1	12.70	3.78E+08	2.82E+08	0.94	0.80	0.87	1.35	1.84	2.41	1.38	1.80	2.42	0.98
flamme2u1	12.89	3.84E+08	2.94E+08	0.96	0.82	0.93	1.42	1.84	2.78	1.44	1.79	2.95	0.99
plasmar1	11.11	2.84E+08	8.53E+07	0.73	0.76	0.28	1.01	1.41	0.93	1.00	1.38	0.92	1.01

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	12.75	2.83E+08	8.70E+07	0.74	0.84	0.26	0.99	1.34	0.90	1.00	1.32	0.92	0.99
postr1	12.88	3.78E+08	2.65E+08	0.98	0.88	0.96	1.38	1.98	2.61	1.36	1.88	2.53	1.02
postu1	12.57	3.49E+08	2.89E+08	0.98	0.88	0.98	1.34	1.81	2.26	1.31	1.74	2.16	1.02
sf2r1	12.77	3.56E+08	2.28E+08	1.03	1.18	1.01	1.19	2.01	1.56	1.21	1.98	1.62	0.99
sf2u1	13.91	3.88E+08	2.18E+08	0.95	1.04	0.79	1.19	1.85	1.68	1.20	1.81	1.66	1.00
spxr3	11.49	3.62E+08	2.13E+08	0.88	0.79	0.74	1.35	1.85	2.84	1.33	1.79	2.59	1.01
spxu3	12.20	3.84E+08	2.23E+08	0.88	0.77	0.75	1.33	1.80	2.45	1.37	1.74	2.93	0.97
torsor1	17.70	5.15E+08	5.40E+08	1.35	1.12	1.82	2.03	2.56	5.99	2.03	2.50	5.91	1.00
torsou1	17.27	5.10E+08	5.14E+08	1.37	1.13	1.85	2.12	2.62	7.37	2.11	2.53	7.12	1.00
mean 100K	0.72	2.27E+06	5.72E+05	1.06	0.98	0.82	0.93	1.60	1.18	1.15	1.59	1.18	0.82
median 100K	0.75	1.94E+06	5.40E+05	1.01	0.88	0.74	0.98	1.60	1.17	1.02	1.56	1.17	0.73
min 100K	0.55	2.42E+05	1.11E+05	0.96	0.84	0.66	0.72	1.00	1.00	1.01	1.00	1.01	0.70
max 100K	0.80	5.42E+06	1.19E+06	1.28	1.20	1.00	1.01	1.96	1.38	1.39	1.97	1.40	1.00
mean 1M	1.25	2.88E+07	8.88E+06	0.91	0.84	0.64	1.09	1.57	1.51	1.10	1.53	1.52	0.99
median 1M	1.07	3.24E+07	8.89E+06	0.91	0.81	0.62	1.07	1.63	1.38	1.08	1.53	1.37	0.99
min 1M	0.76	1.24E+07	3.73E+06	0.82	0.42	0.33	0.96	0.71	0.86	0.96	0.77	0.91	0.98
max 1M	2.56	4.25E+07	1.69E+07	1.02	1.36	1.01	1.26	2.47	2.33	1.25	2.37	2.25	1.01
mean 10M	13.16	3.70E+08	2.62E+08	0.97	0.90	0.90	1.37	1.87	2.66	1.37	1.82	2.65	1.00
median 10M	12.76	3.78E+08	2.46E+08	0.95	0.86	0.90	1.34	1.85	2.43	1.35	1.79	2.47	1.00
min 10M	9.48	2.22E+08	7.09E+07	0.72	0.72	0.26	0.99	1.32	0.90	1.00	1.30	0.92	0.97
max 10M	17.70	5.15E+08	5.40E+08	1.37	1.18	1.85	2.12	2.62	7.37	2.11	2.53	7.12	1.02

Table 14: PT sur Opteron

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.94	4.81E+05	1.98E+05	0.96	1.62	1.45	0.94	1.73	1.49	1.00	1.74	1.49	0.93
fighter	1.97	1.05E+06	3.41E+05	1.03	1.13	0.99	1.04	1.44	1.13	1.09	1.45	1.13	0.96
spxr1	2.24	1.16E+06	4.19E+05	0.99	0.90	0.91	1.02	1.23	1.09	1.02	1.23	1.10	1.00
spxu1	2.34	1.15E+06	4.13E+05	1.01	0.90	0.91	1.12	1.20	1.08	1.09	1.20	1.11	1.03
blunt	2.66	3.15E+06	8.18E+05	1.02	1.08	0.91	1.04	1.54	1.13	1.09	1.56	1.14	0.95
post	3.45	7.46E+06	2.40E+06	1.01	0.88	0.95	1.01	1.31	1.19	1.03	1.25	1.18	0.98
bucky2	4.84	9.67E+06	3.67E+06	0.92	0.55	0.56	0.98	0.82	0.94	0.97	0.88	0.97	1.00
fighterr1	4.05	2.30E+07	8.57E+06	1.14	1.41	1.66	1.20	1.96	2.64	1.21	1.91	2.57	0.99
fighteru1	3.93	2.05E+07	6.75E+06	1.08	1.21	1.39	1.11	1.68	2.11	1.16	1.64	2.12	0.96
flamme2	4.13	1.99E+07	6.09E+06	0.95	0.67	0.64	1.03	1.15	1.19	1.05	1.11	1.16	0.98
plasma	4.82	1.00E+07	3.84E+06	0.91	0.56	0.67	0.94	0.80	0.91	0.96	0.84	0.94	0.98
spxr2	3.61	1.59E+07	4.01E+06	1.02	0.91	0.93	1.10	1.44	1.34	1.09	1.41	1.36	1.01
spxu2	3.82	1.79E+07	4.24E+06	1.01	0.90	0.92	1.04	1.45	1.35	1.08	1.41	1.32	0.96
torso	4.16	2.30E+07	8.71E+06	1.12	1.22	1.95	1.21	2.02	2.84	1.21	1.95	2.73	1.00
sf2	7.24	2.31E+07	8.68E+06	0.96	0.84	0.76	1.00	1.17	1.11	1.02	1.19	1.13	0.98
bucky2r1	19.82	1.59E+08	4.56E+07	0.86	0.82	0.53	1.02	1.31	0.99	1.02	1.28	0.98	1.00
bucky2u1	23.23	1.44E+08	4.31E+07	0.90	0.90	0.50	1.00	1.32	0.98	1.00	1.32	1.05	1.00
fighterr2	26.13	2.11E+08	1.36E+08	1.21	0.95	1.34	1.47	1.57	2.39	1.47	1.51	2.30	1.00
fighteru2	23.74	2.04E+08	1.23E+08	1.06	0.88	1.19	1.32	1.50	2.35	1.29	1.45	2.31	1.02
flamme2r1	22.35	1.98E+08	1.05E+08	1.00	0.82	0.97	1.24	1.47	1.86	1.24	1.46	1.79	1.00
flamme2u1	22.20	1.96E+08	1.06E+08	1.02	0.82	1.03	1.25	1.44	1.97	1.24	1.40	1.91	1.00
plasmar1	23.59	1.75E+08	5.37E+07	0.87	0.85	0.54	1.01	1.33	0.99	1.00	1.33	1.05	1.00

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	28.43	1.76E+08	5.28E+07	0.89	0.93	0.47	0.99	1.27	0.91	1.00	1.25	0.90	0.99
postr1	22.66	1.93E+08	9.30E+07	1.02	0.89	1.05	1.24	1.53	1.76	1.23	1.46	1.69	1.01
postu1	21.18	1.76E+08	1.00E+08	1.00	0.87	1.05	1.16	1.40	1.58	1.14	1.37	1.52	1.01
sf2r1	24.94	1.92E+08	9.63E+07	1.10	1.16	1.17	1.17	1.65	1.58	1.19	1.63	1.61	0.99
sf2u1	27.47	2.14E+08	1.01E+08	1.04	1.04	1.03	1.16	1.58	1.53	1.16	1.53	1.48	1.00
spxr3	20.22	1.83E+08	8.39E+07	0.93	0.78	0.93	1.20	1.43	1.94	1.19	1.38	1.81	1.01
spxu3	21.47	1.94E+08	9.22E+07	0.94	0.77	0.94	1.21	1.41	2.04	1.20	1.37	1.92	1.01
torsor1	30.37	2.36E+08	1.59E+08	1.39	1.00	1.66	1.75	1.81	3.27	1.72	1.76	3.12	1.02
torsou1	28.90	2.34E+08	1.54E+08	1.37	1.03	1.72	1.75	1.83	3.74	1.75	1.78	3.56	1.00
mean 100K	2.03	1.40E+06	4.38E+05	1.00	1.13	1.03	1.03	1.43	1.18	1.06	1.44	1.19	0.97
median 100K	2.24	1.15E+06	4.13E+05	1.01	1.08	0.91	1.04	1.44	1.13	1.09	1.45	1.13	0.96
min 100K	0.94	4.81E+05	1.98E+05	0.96	0.90	0.91	0.94	1.20	1.08	1.00	1.20	1.10	0.93
max 100K	2.66	3.15E+06	8.18E+05	1.03	1.62	1.45	1.12	1.73	1.49	1.09	1.74	1.49	1.03
mean 1M	4.41	1.70E+07	5.69E+06	1.01	0.91	1.04	1.06	1.38	1.56	1.08	1.36	1.55	0.99
median 1M	4.09	1.89E+07	5.16E+06	1.01	0.89	0.93	1.04	1.37	1.26	1.07	1.33	1.25	0.98
min 1M	3.45	7.46E+06	2.40E+06	0.91	0.55	0.56	0.94	0.80	0.91	0.96	0.84	0.94	0.96
max 1M	7.24	2.31E+07	8.71E+06	1.14	1.41	1.95	1.21	2.02	2.84	1.21	1.95	2.73	1.01
mean 10M	24.17	1.93E+08	9.66E+07	1.04	0.91	1.01	1.25	1.49	1.87	1.24	1.46	1.81	1.00
median 10M	23.41	1.93E+08	9.83E+07	1.01	0.88	1.03	1.21	1.45	1.81	1.20	1.43	1.74	1.00
min 10M	19.82	1.44E+08	4.31E+07	0.86	0.77	0.47	0.99	1.27	0.91	1.00	1.25	0.90	0.99
max 10M	30.37	2.36E+08	1.59E+08	1.39	1.16	1.72	1.75	1.83	3.74	1.75	1.78	3.56	1.02

Table 15: HAVS sur Core2

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.98	1.31E+06	3.51E+05	1.01	0.66	1.01	1.02	1.18	1.00	1.02	1.15	1.00	1.00
fighter	1.14	1.35E+07	5.17E+06	0.90	1.48	2.79	0.91	3.45	2.75	1.09	3.47	2.75	0.84
spxr1	1.46	1.66E+07	3.66E+06	0.99	0.96	1.37	1.01	3.18	1.36	1.17	3.11	1.36	0.86
spxu1	1.52	1.64E+07	3.59E+06	1.04	1.16	1.36	1.01	3.46	1.35	1.04	3.18	1.34	0.97
blunt	2.46	3.35E+07	2.23E+07	1.16	1.92	3.67	1.09	3.00	3.54	1.10	3.03	3.54	0.99
post	3.38	3.26E+07	1.99E+07	1.04	0.77	1.14	0.70	1.15	1.13	0.81	1.15	1.12	0.87
bucky2	6.88	1.26E+08	4.69E+07	0.95	1.39	1.13	0.68	1.93	1.25	0.81	1.89	1.24	0.84
fighterr1	13.53	3.21E+08	3.24E+08	1.59	1.51	9.39	1.60	6.06	10.21	1.62	5.41	10.27	0.99
fighteru1	12.44	2.79E+08	2.29E+08	1.36	1.27	7.26	1.40	5.18	7.29	1.38	4.75	7.29	1.02
flamme2	9.63	1.26E+08	6.26E+07	0.72	0.34	0.57	0.82	1.78	1.47	0.87	1.62	1.48	0.95
plasma	7.03	1.13E+08	4.98E+07	1.00	1.20	1.13	0.67	1.68	1.29	0.81	1.66	1.27	0.82
spxr2	9.41	2.42E+08	9.64E+07	1.08	1.00	2.53	1.12	4.60	3.25	1.12	4.28	3.22	1.00
spxu2	10.83	2.42E+08	1.02E+08	1.10	1.03	2.57	1.13	4.54	3.17	1.13	4.25	3.13	1.00
torso	15.82	3.08E+08	2.98E+08	1.44	1.35	7.04	1.49	6.00	9.65	1.49	5.40	9.58	1.00
sf2	17.29	2.90E+08	1.08E+08	1.12	2.09	1.56	0.98	2.62	1.73	0.98	2.45	1.71	0.99
bucky2r1	94.26	1.98E+09	8.73E+08	1.06	1.19	1.03	1.05	3.54	2.60	1.06	3.50	2.54	0.99
bucky2u1	80.99	1.65E+09	6.47E+08	1.15	1.69	1.15	0.94	3.22	2.07	0.95	3.08	2.01	0.99
fighterr2	151.40	3.22E+09	3.24E+09	1.40	1.22	3.13	1.64	5.90	9.16	1.64	5.32	8.77	0.99
fighteru2	139.37	3.07E+09	2.45E+09	1.28	1.27	2.90	1.47	6.04	7.33	1.44	5.79	7.11	1.02
flamme2r1	121.56	2.65E+09	1.84E+09	0.99	0.92	1.01	1.36	5.11	5.40	1.35	4.66	5.21	1.01
flamme2u1	123.45	2.73E+09	1.80E+09	1.04	1.04	1.28	1.30	5.78	5.60	1.31	5.18	5.36	1.00
plasmar1	104.06	2.00E+09	8.88E+08	1.12	1.41	1.00	1.05	3.50	2.50	1.07	3.31	2.42	0.99

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	97.99	1.91E+09	8.44E+08	1.21	1.80	1.35	0.97	3.21	2.29	0.98	3.07	2.25	0.98
postr1	114.09	2.72E+09	1.48E+09	0.99	1.01	1.15	1.22	5.14	4.52	1.23	4.96	4.39	0.99
postu1	106.85	2.65E+09	1.36E+09	1.08	1.16	1.69	1.24	5.54	4.22	1.24	5.00	4.06	1.01
sf2r1	129.61	2.91E+09	2.32E+09	1.52	2.02	4.12	1.50	5.62	6.80	1.51	5.15	6.59	0.99
sf2u1	138.41	3.10E+09	2.19E+09	1.34	1.77	2.41	1.38	5.18	5.62	1.39	5.07	5.50	0.99
spxr3	109.04	2.60E+09	1.42E+09	0.94	0.94	0.97	1.20	4.93	4.42	1.21	4.66	4.25	0.99
spxu3	120.84	2.82E+09	1.54E+09	0.94	0.96	0.93	1.25	5.12	4.55	1.23	4.77	4.42	1.02
torsor1	197.77	3.69E+09	4.43E+09	1.48	1.33	2.81	1.86	7.55	13.86	1.87	6.83	13.34	0.99
torsou1	196.92	3.64E+09	4.37E+09	1.57	1.42	3.50	1.89	7.60	14.39	1.90	7.27	14.03	0.99
mean 100K	1.51	1.63E+07	7.02E+06	1.02	1.23	2.04	1.01	2.85	2.00	1.08	2.79	2.00	0.93
median 100K	1.46	1.64E+07	3.66E+06	1.01	1.16	1.37	1.01	3.18	1.36	1.09	3.11	1.36	0.97
min 100K	0.98	1.31E+06	3.51E+05	0.90	0.66	1.01	0.91	1.18	1.00	1.02	1.15	1.00	0.84
max 100K	2.46	3.35E+07	2.23E+07	1.16	1.92	3.67	1.09	3.46	3.54	1.17	3.47	3.54	1.00
mean 1M	10.62	2.08E+08	1.34E+08	1.14	1.20	3.43	1.06	3.55	4.04	1.10	3.29	4.03	0.95
median 1M	10.23	2.42E+08	9.91E+07	1.09	1.24	2.04	1.05	3.58	2.45	1.05	3.35	2.42	0.99
min 1M	3.38	3.26E+07	1.99E+07	0.72	0.34	0.57	0.67	1.15	1.13	0.81	1.15	1.12	0.82
max 1M	17.29	3.21E+08	3.24E+08	1.59	2.09	9.39	1.60	6.06	10.21	1.62	5.41	10.27	1.02
mean 10M	126.66	2.71E+09	1.98E+09	1.20	1.32	1.90	1.33	5.19	5.96	1.34	4.85	5.77	1.00
median 10M	121.20	2.73E+09	1.67E+09	1.14	1.25	1.31	1.28	5.16	4.97	1.27	4.98	4.81	0.99
min 10M	80.99	1.65E+09	6.47E+08	0.94	0.92	0.93	0.94	3.21	2.07	0.95	3.07	2.01	0.98
max 10M	197.77	3.69E+09	4.43E+09	1.57	2.02	4.12	1.89	7.60	14.39	1.90	7.27	14.03	1.02

Table 16: HAVS sur Opteron

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	2.35	1.20E+06	5.96E+05	1.01	0.81	1.00	1.00	1.10	0.98	1.04	1.08	1.01	0.96
fighter	3.13	7.52E+06	4.02E+06	1.07	1.52	1.92	1.04	1.91	1.99	1.08	1.98	2.03	0.96
spxr1	3.33	9.68E+06	3.83E+06	0.99	0.99	1.32	1.00	1.92	1.35	1.02	1.86	1.33	0.99
spxu1	3.39	9.05E+06	3.68E+06	1.03	0.74	1.41	1.01	1.78	1.38	1.02	1.67	1.36	0.99
blunt	5.16	2.08E+07	1.22E+07	1.21	1.26	2.46	1.10	1.96	2.05	1.13	1.73	2.15	0.98
post	7.77	3.22E+07	1.77E+07	0.97	0.76	1.23	0.80	1.02	1.08	0.92	1.03	1.14	0.87
bucky2	14.18	2.12E+08	3.85E+07	0.98	1.50	1.04	0.79	3.02	1.18	0.88	2.93	1.16	0.90
fighterr1	26.68	1.70E+08	1.18E+08	1.62	1.38	3.50	1.73	3.22	4.01	1.76	3.19	4.20	0.99
fighteru1	24.23	1.50E+08	9.29E+07	1.42	1.18	2.91	1.48	2.77	3.40	1.51	2.87	3.39	0.98
flamme2	18.76	9.17E+07	5.10E+07	0.71	0.50	0.56	0.91	1.33	1.32	0.94	1.27	1.29	0.97
plasma	14.06	2.25E+08	4.03E+07	0.93	2.27	1.01	0.76	3.02	1.19	0.83	2.75	1.18	0.92
spxr2	18.48	1.25E+08	5.54E+07	1.03	0.88	1.17	1.18	2.35	2.05	1.20	2.31	2.18	0.99
spxu2	20.67	1.37E+08	6.18E+07	1.03	0.85	1.10	1.19	2.44	2.24	1.21	2.42	2.11	0.98
torso	30.86	1.81E+08	1.33E+08	1.44	1.17	2.38	1.68	3.35	4.84	1.72	3.17	5.34	0.98
sf2	31.29	1.58E+08	7.99E+07	1.07	1.21	1.31	0.99	1.50	1.36	1.01	1.58	1.51	0.98
bucky2r1	160.82	9.07E+08	4.56E+08	0.96	0.98	0.89	1.07	1.88	1.73	1.13	1.85	1.72	0.95
bucky2u1	143.26	7.90E+08	3.62E+08	1.06	1.23	1.02	1.00	1.84	1.54	1.03	1.79	1.62	0.98
fighterr2	270.12	1.31E+09	9.33E+08	1.33	0.98	1.43	1.74	2.41	3.44	1.76	2.45	3.43	0.99
fighteru2	250.52	1.35E+09	8.07E+08	1.13	0.87	1.06	1.60	2.66	3.09	1.60	2.66	3.29	1.00
flamme2r1	216.61	1.31E+09	7.21E+08	0.88	0.80	0.72	1.43	2.39	2.71	1.44	2.60	2.74	0.99
flamme2u1	221.13	1.26E+09	6.96E+08	0.99	0.92	0.86	1.42	2.42	2.75	1.43	2.48	2.93	0.99
plasmar1	181.70	1.11E+09	4.92E+08	1.06	1.13	0.89	1.10	2.06	1.80	1.12	2.10	1.91	0.98

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	169.61	1.07E+09	4.66E+08	1.11	1.31	1.18	1.00	1.98	1.56	1.03	1.75	1.75	0.97
postr1	194.69	1.15E+09	6.19E+08	0.96	0.92	1.03	1.23	2.28	2.33	1.26	2.32	2.38	0.98
postu1	188.50	1.17E+09	5.75E+08	1.05	0.99	1.04	1.32	2.60	2.30	1.32	2.48	2.51	1.00
sf2r1	234.34	1.24E+09	7.63E+08	1.48	1.38	1.77	1.62	2.60	2.91	1.74	2.52	3.10	0.94
sf2u1	242.41	1.24E+09	7.10E+08	1.29	1.32	1.44	1.45	2.57	2.79	1.47	2.26	2.48	0.98
spxr3	196.78	1.33E+09	6.24E+08	0.89	0.91	0.79	1.29	2.74	2.62	1.29	2.59	2.43	1.00
spxu3	211.06	1.30E+09	6.18E+08	0.91	0.88	0.75	1.29	2.39	2.23	1.31	2.55	2.41	0.99
torsor1	345.73	1.49E+09	1.17E+09	1.48	1.13	1.53	1.99	3.50	5.28	2.02	3.22	5.23	0.99
torsou1	346.39	1.55E+09	1.20E+09	1.54	1.11	1.63	2.10	3.32	5.06	2.12	3.34	5.04	0.99
mean 100K	3.47	9.65E+06	4.86E+06	1.06	1.06	1.62	1.03	1.73	1.55	1.06	1.66	1.57	0.98
median 100K	3.33	9.05E+06	3.83E+06	1.03	0.99	1.41	1.01	1.91	1.38	1.04	1.73	1.36	0.98
min 100K	2.35	1.20E+06	5.96E+05	0.99	0.74	1.00	1.00	1.10	0.98	1.02	1.08	1.01	0.96
max 100K	5.16	2.08E+07	1.22E+07	1.21	1.52	2.46	1.10	1.96	2.05	1.13	1.98	2.15	0.99
mean 1M	20.70	1.48E+08	6.88E+07	1.12	1.17	1.62	1.15	2.40	2.27	1.20	2.35	2.35	0.96
median 1M	19.72	1.54E+08	5.86E+07	1.03	1.18	1.20	1.09	2.60	1.71	1.10	2.58	1.81	0.98
min 1M	7.77	3.22E+07	1.77E+07	0.71	0.50	0.56	0.76	1.02	1.08	0.83	1.03	1.14	0.87
max 1M	31.29	2.25E+08	1.33E+08	1.62	2.27	3.50	1.73	3.35	4.84	1.76	3.19	5.34	0.99
mean 10M	223.35	1.22E+09	7.01E+08	1.13	1.05	1.13	1.42	2.48	2.76	1.44	2.43	2.81	0.98
median 10M	213.83	1.25E+09	6.60E+08	1.06	0.99	1.03	1.37	2.42	2.66	1.38	2.48	2.49	0.99
min 10M	143.26	7.90E+08	3.62E+08	0.88	0.80	0.72	1.00	1.84	1.54	1.03	1.75	1.62	0.94
max 10M	346.39	1.55E+09	1.20E+09	1.54	1.38	1.77	2.10	3.50	5.28	2.12	3.34	5.23	1.00

Table 17: VtkIso sur Core2

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.01	1.85E+05	1.72E+04	0.97	0.73	1.01	1.00	1.07	1.03	1.00	1.05	1.07	0.99
fighter	0.20	3.52E+07	3.30E+05	1.02	1.42	1.13	1.07	10.27	1.15	1.07	10.34	1.17	1.00
spxr1	0.08	2.18E+06	5.45E+05	0.98	0.75	0.99	1.03	1.99	1.05	1.03	1.92	1.06	1.00
spxu1	0.05	1.61E+06	5.17E+05	0.98	0.87	1.04	1.03	1.63	1.08	1.03	1.61	1.11	1.00
blunt	0.13	1.94E+07	1.77E+06	1.01	1.13	1.04	1.04	2.45	1.01	1.04	2.52	1.04	1.00
post	0.46	6.30E+07	5.01E+06	0.99	0.86	1.00	1.00	1.07	0.99	1.00	1.12	0.99	1.00
bucky2	0.52	1.21E+07	1.06E+07	0.99	0.88	0.99	0.99	1.03	1.00	1.01	1.02	0.99	0.99
fighterr1	1.63	2.53E+08	2.26E+07	1.43	1.26	2.50	1.56	9.99	2.54	1.55	9.91	2.53	1.01
fighteru1	0.79	4.04E+07	1.41E+07	1.28	1.12	1.54	1.37	3.75	1.55	1.36	3.61	1.55	1.00
flamme2	0.64	3.59E+07	1.18E+07	0.92	0.35	0.99	1.01	1.10	1.01	1.00	1.15	1.01	1.00
plasma	0.42	1.19E+07	1.08E+07	1.00	0.93	0.99	1.00	1.04	1.00	1.01	1.03	1.00	1.00
spxr2	0.59	1.91E+07	9.42E+06	0.99	0.45	1.04	1.08	1.89	1.07	1.08	1.84	1.07	1.00
spxu2	0.41	1.62E+07	9.82E+06	1.00	0.59	1.04	1.09	1.64	1.06	1.09	1.61	1.05	1.00
torso	0.37	2.55E+07	9.76E+06	1.23	1.30	1.11	1.32	2.84	1.12	1.32	2.78	1.12	1.00
sf2	2.23	2.43E+08	1.81E+07	1.11	3.96	1.04	1.05	7.15	1.05	1.05	7.09	1.05	1.00
bucky2r1	3.96	1.63E+08	8.39E+07	1.02	0.65	1.02	1.10	1.82	1.04	1.10	1.77	1.04	1.00
bucky2u1	3.21	1.32E+08	7.81E+07	1.04	1.11	1.01	1.08	1.60	1.03	1.07	1.58	1.03	1.00
fighterr2	17.87	2.09E+09	4.40E+08	1.69	0.80	4.98	1.92	5.88	5.24	1.91	5.78	5.20	1.00
fighteru2	6.43	2.63E+08	1.74E+08	1.55	0.82	2.05	1.69	2.84	2.09	1.69	2.80	2.08	1.00
flamme2r1	4.89	4.11E+08	9.20E+07	1.01	0.47	1.08	1.17	2.42	1.14	1.17	2.39	1.14	1.00
flamme2u1	3.63	1.81E+08	9.37E+07	1.13	0.60	1.14	1.25	1.99	1.16	1.25	1.96	1.16	1.00
plasmar1	3.16	1.46E+08	8.87E+07	1.04	0.84	1.01	1.09	1.61	1.02	1.09	1.58	1.02	1.00

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	3.15	1.45E+08	9.10E+07	1.05	1.26	1.01	1.07	1.56	1.02	1.06	1.54	1.02	1.00
postr1	17.59	4.21E+09	1.22E+08	0.92	0.56	0.97	1.11	1.56	1.48	1.11	1.56	1.48	1.00
postu1	7.45	1.53E+09	8.41E+07	1.02	0.77	1.09	1.10	1.69	1.11	1.10	1.70	1.11	1.00
sf2r1	13.44	3.44E+09	1.67E+08	1.29	1.10	2.12	1.41	6.53	2.15	1.41	6.48	2.14	1.00
sf2u1	4.60	3.72E+08	1.10E+08	1.28	2.18	1.25	1.29	4.00	1.25	1.28	3.91	1.25	1.00
spxr3	4.67	2.00E+08	9.50E+07	1.04	0.40	1.12	1.17	2.10	1.17	1.17	2.03	1.16	1.00
spxu3	3.56	1.55E+08	9.70E+07	1.07	0.52	1.12	1.21	1.73	1.14	1.20	1.70	1.14	1.01
torsor1	9.35	7.64E+08	2.75E+08	2.10	0.79	3.34	2.33	2.62	3.42	2.34	2.59	3.41	0.99
torsou1	7.06	2.90E+08	2.43E+08	2.89	1.34	3.10	3.12	3.63	3.11	3.11	3.60	3.10	1.00
mean 100K	0.09	1.17E+07	6.35E+05	0.99	0.98	1.04	1.04	3.48	1.06	1.04	3.49	1.09	1.00
median 100K	0.08	2.18E+06	5.17E+05	0.98	0.87	1.04	1.03	1.99	1.05	1.03	1.92	1.07	1.00
min 100K	0.01	1.85E+05	1.72E+04	0.97	0.73	0.99	1.00	1.07	1.01	1.00	1.05	1.04	0.99
max 100K	0.20	3.52E+07	1.77E+06	1.02	1.42	1.13	1.07	10.27	1.15	1.07	10.34	1.17	1.00
mean 1M	0.80	7.20E+07	1.22E+07	1.10	1.17	1.22	1.15	3.15	1.24	1.15	3.12	1.24	1.00
median 1M	0.56	3.07E+07	1.07E+07	1.00	0.90	1.04	1.06	1.76	1.05	1.06	1.73	1.05	1.00
min 1M	0.37	1.19E+07	5.01E+06	0.92	0.35	0.99	0.99	1.03	0.99	1.00	1.02	0.99	0.99
max 1M	2.23	2.53E+08	2.26E+07	1.43	3.96	2.50	1.56	9.99	2.54	1.55	9.91	2.53	1.01
mean 10M	7.13	9.06E+08	1.46E+08	1.32	0.89	1.71	1.44	2.72	1.79	1.44	2.69	1.78	1.00
median 10M	4.78	2.77E+08	9.60E+07	1.06	0.80	1.12	1.19	2.04	1.16	1.18	2.00	1.16	1.00
min 10M	3.15	1.32E+08	7.81E+07	0.92	0.40	0.97	1.07	1.56	1.02	1.06	1.54	1.02	0.99
max 10M	17.87	4.21E+09	4.40E+08	2.89	2.18	4.98	3.12	6.53	5.24	3.11	6.48	5.20	1.01

Table 18: VtkIso sur Opteron

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.02	1.84E+05	4.58E+04	0.96	0.84	0.99	0.99	0.98	0.99	1.00	1.07	1.04	0.99
fighter	0.42	1.67E+07	7.51E+05	1.10	2.19	1.14	1.17	6.03	1.17	1.17	6.86	1.15	1.00
spxr1	0.16	1.66E+06	7.41E+05	1.00	0.89	0.99	1.05	1.47	0.99	1.05	1.46	1.12	1.00
spxu1	0.10	1.26E+06	7.09E+05	0.99	0.95	1.06	1.06	1.29	1.11	1.06	1.30	1.09	1.00
blunt	0.29	7.87E+06	1.36E+06	1.04	1.29	1.01	1.06	1.86	0.91	1.08	2.37	0.97	0.99
post	1.10	4.60E+07	3.76E+06	0.99	0.90	1.00	0.99	1.02	1.01	1.01	1.08	0.98	0.98
bucky2	1.04	1.29E+07	8.28E+06	1.00	0.89	1.02	1.01	1.06	1.04	1.01	1.05	1.15	0.99
fighterr1	5.26	1.46E+08	3.91E+07	2.17	1.46	5.41	2.50	7.75	5.54	2.49	7.52	5.96	1.00
fighteru1	2.01	3.14E+07	1.50E+07	1.67	1.22	2.13	1.81	2.71	2.12	1.81	2.66	2.37	1.00
flamme2	1.35	1.95E+07	7.31E+06	0.86	0.29	0.82	1.01	1.04	0.92	1.02	1.07	0.92	0.99
plasma	0.86	1.18E+07	7.74E+06	0.99	0.93	0.96	0.98	1.01	0.96	0.99	1.01	0.99	0.99
spxr2	1.22	1.59E+07	7.08E+06	1.00	0.61	1.00	1.11	1.44	1.11	1.11	1.39	1.22	1.00
spxu2	0.88	1.40E+07	7.65E+06	1.04	0.76	1.08	1.14	1.35	1.07	1.14	1.39	1.11	1.00
torso	1.32	2.29E+07	1.25E+07	2.03	1.72	1.91	2.16	2.57	1.85	2.18	2.55	2.03	0.99
sf2	4.49	9.23E+07	1.47E+07	1.09	2.10	1.07	1.07	2.69	1.07	1.07	2.52	1.11	1.00
bucky2r1	8.73	1.38E+08	6.73E+07	1.00	0.65	1.02	1.15	1.50	1.08	1.14	1.53	1.12	1.01
bucky2u1	7.22	1.12E+08	6.16E+07	1.06	1.04	1.01	1.14	1.35	1.05	1.13	1.37	1.08	1.01
fighterr2	41.21	9.48E+08	2.88E+08	1.53	0.58	3.08	2.14	4.49	4.36	2.13	4.39	4.47	1.00
fighteru2	13.68	1.81E+08	1.17E+08	1.49	0.70	1.69	1.79	1.94	1.80	1.78	1.96	1.82	1.01
flamme2r1	11.96	2.55E+08	8.19E+07	0.95	0.41	1.07	1.33	1.97	1.37	1.32	1.91	1.35	1.01
flamme2u1	8.64	1.47E+08	7.76E+07	1.13	0.59	1.18	1.39	1.68	1.26	1.37	1.69	1.24	1.01
plasmar1	7.22	1.26E+08	6.92E+07	1.03	0.80	1.00	1.13	1.40	1.02	1.13	1.39	1.07	1.00

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	7.44	1.24E+08	7.16E+07	1.10	1.13	1.00	1.15	1.32	1.03	1.14	1.33	1.04	1.00
postr1	52.01	3.29E+09	1.57E+08	0.46	0.54	0.16	1.27	1.50	2.44	1.27	1.48	2.62	1.00
postu1	23.36	1.27E+09	1.18E+08	0.92	0.76	0.84	1.36	1.50	1.96	1.36	1.51	2.02	1.00
sf2r1	40.35	1.86E+09	2.30E+08	1.65	1.24	3.75	2.01	8.12	3.96	2.00	8.30	3.94	1.00
sf2u1	11.34	2.75E+08	9.43E+07	1.42	1.82	1.34	1.49	2.91	1.37	1.49	2.87	1.40	1.00
spxr3	9.98	1.48E+08	7.05E+07	0.97	0.42	1.03	1.19	1.56	1.13	1.18	1.53	1.13	1.00
spxu3	8.00	1.28E+08	7.52E+07	1.05	0.51	1.10	1.24	1.42	1.14	1.24	1.42	1.16	1.00
torsor1	22.37	4.44E+08	2.01E+08	1.91	0.64	3.05	2.52	2.63	3.26	2.50	2.58	3.27	1.01
torsou1	16.68	2.16E+08	1.80E+08	2.84	1.16	2.94	3.21	2.71	2.93	3.23	2.69	2.94	1.00
mean 100K	0.20	5.52E+06	7.21E+05	1.02	1.23	1.04	1.07	2.33	1.04	1.07	2.61	1.08	1.00
median 100K	0.16	1.66E+06	7.41E+05	1.00	0.95	1.01	1.06	1.47	0.99	1.06	1.46	1.09	1.00
min 100K	0.02	1.84E+05	4.58E+04	0.96	0.84	0.99	0.99	0.98	0.91	1.00	1.07	0.97	0.99
max 100K	0.42	1.67E+07	1.36E+06	1.10	2.19	1.14	1.17	6.03	1.17	1.17	6.86	1.15	1.00
mean 1M	1.95	4.13E+07	1.23E+07	1.28	1.09	1.64	1.38	2.26	1.67	1.38	2.23	1.78	1.00
median 1M	1.27	2.12E+07	8.01E+06	1.02	0.91	1.05	1.09	1.40	1.07	1.09	1.39	1.13	1.00
min 1M	0.86	1.18E+07	3.76E+06	0.86	0.29	0.82	0.98	1.01	0.92	0.99	1.01	0.92	0.98
max 1M	5.26	1.46E+08	3.91E+07	2.17	2.10	5.41	2.50	7.75	5.54	2.49	7.52	5.96	1.00
mean 10M	18.14	6.04E+08	1.23E+08	1.28	0.81	1.58	1.59	2.38	1.95	1.59	2.37	1.98	1.00
median 10M	11.65	1.98E+08	8.81E+07	1.08	0.68	1.09	1.34	1.62	1.37	1.34	1.61	1.37	1.00
min 10M	7.22	1.12E+08	6.16E+07	0.46	0.41	0.16	1.13	1.32	1.02	1.13	1.33	1.04	1.00
max 10M	52.01	3.29E+09	2.88E+08	2.84	1.82	3.75	3.21	8.12	4.36	3.23	8.30	4.47	1.01

Table 19: CpuIso sur Core2

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.00	5.31E+04	1.11E+04	0.97	1.07	0.91	1.01	1.12	1.00	1.01	1.10	0.99	1.00
fighter	0.01	1.96E+06	2.92E+05	1.15	4.20	1.00	1.20	5.93	1.00	1.21	5.93	1.00	1.00
spxr1	0.01	1.25E+06	3.09E+05	0.99	0.85	1.02	1.16	3.65	1.01	1.17	3.47	1.01	0.99
spxu1	0.01	1.25E+06	2.76E+05	1.02	0.92	1.02	1.21	4.07	1.02	1.22	3.88	1.02	1.00
blunt	0.02	2.78E+06	6.12E+05	1.20	1.99	1.07	1.22	4.14	1.06	1.25	4.14	1.06	0.98
post	0.03	1.74E+06	1.53E+06	0.91	0.40	1.00	0.89	0.96	0.98	0.91	0.92	0.97	0.98
bucky2	0.08	4.02E+06	3.41E+06	0.97	0.67	1.00	0.88	0.98	0.97	0.92	0.93	0.96	0.96
fighterr1	0.41	5.09E+07	2.09E+07	3.67	1.95	6.94	4.86	14.57	6.82	4.82	13.63	6.76	1.01
fighteru1	0.29	3.13E+07	1.34E+07	2.55	1.04	4.53	3.56	9.06	4.43	3.54	8.43	4.39	1.00
flamme2	0.10	9.52E+06	3.92E+06	0.57	0.18	1.03	1.03	2.20	1.06	1.03	2.09	1.05	1.00
plasma	0.08	4.04E+06	3.38E+06	0.97	0.66	1.00	0.88	0.99	0.97	0.99	0.98	0.96	0.88
spxr2	0.12	1.62E+07	4.15E+06	1.06	0.49	1.49	1.63	5.03	1.47	1.64	4.74	1.46	1.00
spxu2	0.13	1.86E+07	4.38E+06	1.06	0.50	1.55	1.75	5.62	1.53	1.76	5.26	1.51	0.99
torso	0.43	4.58E+07	2.23E+07	3.99	1.34	8.62	6.67	15.05	8.49	6.56	13.83	8.40	1.02
sf2	0.20	1.97E+07	7.54E+06	1.33	1.63	1.32	1.28	2.99	1.30	1.30	2.91	1.29	0.99
bucky2r1	0.92	1.60E+08	2.87E+07	0.96	0.66	1.07	1.36	5.25	1.09	1.35	4.85	1.07	1.01
bucky2u1	0.77	1.16E+08	2.57E+07	1.18	1.34	1.07	1.27	4.17	1.06	1.25	3.85	1.05	1.01
fighterr2	5.07	3.82E+08	3.42E+08	3.84	0.99	11.29	6.52	11.45	11.73	6.35	10.52	11.49	1.03
fighteru2	3.42	3.79E+08	2.02E+08	2.82	0.98	7.65	4.97	12.15	7.45	4.86	11.22	7.34	1.02
flamme2r1	1.91	2.84E+08	7.56E+07	1.22	0.65	2.05	3.00	9.60	2.96	2.93	8.82	2.92	1.02
flamme2u1	2.03	2.58E+08	8.83E+07	1.62	0.65	3.35	3.23	8.81	3.48	3.16	8.11	3.43	1.02
plasmar1	0.93	1.64E+08	2.94E+07	1.04	0.79	1.07	1.38	5.21	1.08	1.35	4.80	1.07	1.02

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	0.93	1.57E+08	3.01E+07	1.35	2.03	1.08	1.36	4.90	1.09	1.35	4.52	1.07	1.01
postr1	1.79	2.59E+08	7.60E+07	1.28	0.66	2.10	2.80	8.88	2.94	2.74	8.15	2.88	1.02
postu1	1.52	2.43E+08	5.55E+07	1.39	0.71	2.17	2.63	9.08	2.35	2.59	8.34	2.32	1.02
sf2r1	2.93	4.17E+08	1.49E+08	3.24	1.94	5.76	4.40	14.17	5.78	4.30	13.04	5.68	1.02
sf2u1	2.20	4.03E+08	9.22E+07	2.29	1.68	3.40	3.21	12.64	3.33	3.16	11.63	3.28	1.02
spxr3	1.73	2.63E+08	7.29E+07	1.32	0.65	2.57	2.64	8.82	2.81	2.58	8.10	2.75	1.02
spxu3	1.83	2.67E+08	7.79E+07	1.38	0.63	2.85	2.79	8.68	2.93	2.72	8.00	2.88	1.03
torsor1	7.11	5.95E+08	5.17E+08	5.26	1.45	17.18	11.57	20.56	20.74	11.30	18.87	20.39	1.02
torsou1	7.03	5.94E+08	5.16E+08	6.25	1.60	21.70	12.11	21.43	21.62	11.85	19.70	21.28	1.02
mean 100K	0.01	1.46E+06	3.00E+05	1.06	1.81	1.00	1.16	3.78	1.02	1.17	3.71	1.02	0.99
median 100K	0.01	1.25E+06	2.92E+05	1.02	1.07	1.02	1.20	4.07	1.01	1.21	3.88	1.01	1.00
min 100K	0.00	5.31E+04	1.11E+04	0.97	0.85	0.91	1.01	1.12	1.00	1.01	1.10	0.99	0.98
max 100K	0.02	2.78E+06	6.12E+05	1.20	4.20	1.07	1.22	5.93	1.06	1.25	5.93	1.06	1.00
mean 1M	0.19	2.02E+07	8.49E+06	1.71	0.89	2.85	2.34	5.75	2.80	2.35	5.37	2.78	0.98
median 1M	0.12	1.74E+07	4.26E+06	1.06	0.66	1.41	1.45	4.01	1.38	1.47	3.82	1.38	0.99
min 1M	0.03	1.74E+06	1.53E+06	0.57	0.18	1.00	0.88	0.96	0.97	0.91	0.92	0.96	0.88
max 1M	0.43	5.09E+07	2.23E+07	3.99	1.95	8.62	6.67	15.05	8.49	6.56	13.83	8.40	1.02
mean 10M	2.63	3.09E+08	1.49E+08	2.28	1.09	5.40	4.08	10.36	5.78	3.99	9.53	5.68	1.02
median 10M	1.87	2.65E+08	7.70E+07	1.38	0.88	2.71	2.90	8.98	2.95	2.84	8.24	2.90	1.02
min 10M	0.77	1.16E+08	2.57E+07	0.96	0.63	1.07	1.27	4.17	1.06	1.25	3.85	1.05	1.01
max 10M	7.11	5.95E+08	5.17E+08	6.25	2.03	21.70	12.11	21.43	21.62	11.85	19.70	21.28	1.03

Table 20: CpuIso sur Opteron

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
spx	0.00	4.84E+04	1.01E+04	1.01	1.03	1.37	1.00	1.05	1.08	1.02	1.04	1.09	0.98
fighter	0.03	1.46E+06	6.83E+04	1.40	4.15	1.40	1.40	4.59	0.93	1.44	4.59	1.23	0.97
spxr1	0.02	7.58E+05	7.69E+04	1.13	1.40	1.21	1.19	2.30	1.05	1.17	2.23	0.97	1.01
spxu1	0.02	7.28E+05	9.75E+04	1.23	1.72	1.34	1.27	2.48	1.21	1.25	2.42	1.21	1.01
blunt	0.06	2.07E+06	3.09E+05	1.68	2.09	1.73	1.60	3.26	1.52	1.63	3.27	1.54	0.98
post	0.09	1.74E+06	5.18E+05	0.93	0.51	1.19	0.86	1.01	0.89	0.88	0.98	0.93	0.97
bucky2	0.20	4.84E+06	9.53E+05	1.00	0.72	1.18	0.88	1.25	0.84	0.92	1.22	0.98	0.96
fighterr1	1.42	3.37E+07	2.15E+07	6.32	2.64	39.94	7.13	10.10	25.30	7.15	9.53	26.48	1.00
fighteru1	0.87	2.07E+07	1.20E+07	3.99	1.40	23.48	4.59	6.26	15.53	4.50	5.99	15.16	1.02
flamme2	0.25	7.58E+06	1.09E+06	0.57	0.19	0.79	1.05	1.86	0.87	1.07	1.80	0.97	0.98
plasma	0.20	3.94E+06	1.03E+06	1.02	0.65	1.41	0.88	1.03	0.96	0.97	1.03	0.94	0.91
spxr2	0.31	9.80E+06	1.86E+06	1.21	0.46	3.68	1.70	3.21	2.19	1.68	3.08	2.19	1.01
spxu2	0.33	1.10E+07	1.97E+06	1.21	0.42	4.02	1.75	3.50	2.01	1.75	3.35	2.10	1.00
torso	1.43	3.06E+07	2.26E+07	5.87	1.32	48.42	8.22	10.59	22.25	8.26	9.98	26.39	1.00
sf2	0.48	1.44E+07	3.02E+06	1.41	1.35	2.52	1.32	2.29	1.78	1.32	2.26	1.79	1.00
bucky2r1	3.17	1.23E+08	1.80E+07	1.03	0.61	1.37	1.88	4.28	2.22	1.83	4.05	2.11	1.03
bucky2u1	2.54	9.73E+07	1.32E+07	1.33	1.27	1.37	1.66	3.69	1.75	1.62	3.49	1.70	1.02
fighterr2	11.49	2.01E+08	1.59E+08	3.38	0.67	15.29	6.24	6.29	20.63	6.02	5.96	18.05	1.04
fighteru2	7.45	1.72E+08	8.59E+07	2.46	0.53	17.22	4.29	5.83	10.07	4.24	5.51	9.78	1.01
flamme2r1	5.95	1.75E+08	6.30E+07	0.84	0.50	0.88	3.61	6.26	7.04	3.57	5.90	7.52	1.01
flamme2u1	5.53	1.48E+08	5.36E+07	1.21	0.43	2.06	3.36	5.35	5.92	3.33	5.05	6.11	1.01
plasmar1	3.35	1.31E+08	1.92E+07	1.02	0.73	1.09	1.95	4.40	2.24	1.89	4.16	2.06	1.03

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL vs
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	OpenCCL
plasmau1	3.47	1.29E+08	2.18E+07	1.65	1.81	2.04	1.95	4.24	2.36	1.94	4.02	2.43	1.00
postr1	5.28	1.62E+08	5.25E+07	1.27	0.56	1.92	3.17	5.83	5.85	3.10	5.49	5.95	1.02
postu1	3.97	1.24E+08	3.16E+07	1.38	0.47	2.96	2.63	4.87	4.02	2.56	4.59	3.85	1.03
sf2r1	7.95	2.45E+08	9.10E+07	3.47	1.53	13.56	4.79	8.77	10.99	4.76	8.26	11.28	1.01
sf2u1	6.24	2.53E+08	6.42E+07	2.31	1.27	7.19	3.56	8.38	7.44	3.46	7.92	6.71	1.03
spxr3	3.76	1.21E+08	2.65E+07	0.84	0.36	1.07	2.27	4.30	3.20	2.20	4.04	3.13	1.03
spxu3	4.02	1.21E+08	2.84E+07	0.79	0.33	0.95	2.35	4.15	3.03	2.31	3.92	3.13	1.01
torsor1	16.91	3.10E+08	2.81E+08	3.34	0.93	7.71	10.58	11.34	33.38	10.26	10.66	31.71	1.03
torsou1	16.83	3.10E+08	2.83E+08	4.72	0.97	21.90	10.96	11.84	33.79	10.83	11.18	33.98	1.01
mean 100K	0.03	1.01E+06	1.12E+05	1.29	2.08	1.41	1.29	2.74	1.16	1.30	2.71	1.21	0.99
median 100K	0.02	7.58E+05	7.69E+04	1.23	1.72	1.37	1.27	2.48	1.08	1.25	2.42	1.21	0.98
min 100K	0.00	4.84E+04	1.01E+04	1.01	1.03	1.21	1.00	1.05	0.93	1.02	1.04	0.97	0.97
max 100K	0.06	2.07E+06	3.09E+05	1.68	4.15	1.73	1.60	4.59	1.52	1.63	4.59	1.54	1.01
mean 1M	0.56	1.38E+07	6.66E+06	2.35	0.97	12.66	2.84	4.11	7.26	2.85	3.92	7.79	0.98
median 1M	0.32	1.04E+07	1.91E+06	1.21	0.68	3.10	1.51	2.75	1.90	1.50	2.67	1.94	1.00
min 1M	0.09	1.74E+06	5.18E+05	0.57	0.19	0.79	0.86	1.01	0.84	0.88	0.98	0.93	0.91
max 1M	1.43	3.37E+07	2.26E+07	6.32	2.64	48.42	8.22	10.59	25.30	8.26	9.98	26.48	1.02
mean 10M	6.75	1.76E+08	8.07E+07	1.94	0.81	6.16	4.08	6.24	9.62	4.00	5.89	9.34	1.02
median 10M	5.40	1.55E+08	5.31E+07	1.35	0.64	2.05	3.27	5.59	5.89	3.21	5.27	6.03	1.02
min 10M	2.54	9.73E+07	1.32E+07	0.79	0.33	0.88	1.66	3.69	1.75	1.62	3.49	1.70	1.00
max 10M	16.91	3.10E+08	2.83E+08	4.72	1.81	21.90	10.96	11.84	33.79	10.83	11.18	33.98	1.04

Table 21: CpuTree sur Core2

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL (bsp)			FastCOL (bsp)
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	vs OpenCCL
spx	0.00	2.9E+4	1.1E+4	0.91	0.62	0.82	1.01	1.06	0.98	1.01	1.03	1.01	1.06	1.05	0.99	1.06
fighter	0.01	4.8E+5	3.5E+5	1.13	0.97	1.17	1.29	1.57	1.37	1.29	1.57	1.36	1.50	1.64	1.32	1.16
spxr1	0.01	2.7E+5	2.1E+5	0.79	0.89	0.74	1.23	1.61	1.30	1.20	1.54	1.27	1.39	1.59	1.22	1.13
spxu1	0.00	1.4E+5	1.3E+5	0.82	0.90	0.77	1.33	1.65	1.47	1.29	1.57	1.41	1.48	1.61	1.34	1.11
blunt	0.00	1.5E+5	1.4E+5	1.09	1.23	1.01	1.28	1.49	1.44	1.28	1.48	1.44	1.42	1.41	1.26	1.11
post	0.01	1.8E+5	1.7E+5	0.88	0.89	0.87	0.87	0.97	0.96	0.90	0.96	0.96	0.98	0.96	0.85	1.12
bucky2	0.03	1.1E+6	1.1E+6	0.82	0.97	0.86	0.94	1.11	1.11	0.98	1.07	1.06	1.08	1.14	1.04	1.15
fighterr1	0.10	3.1E+6	3.6E+6	1.16	1.15	1.25	2.00	2.10	2.63	1.95	2.01	2.56	2.30	2.08	2.52	1.15
fighteru1	0.07	2.3E+6	2.5E+6	0.97	1.05	0.97	1.80	1.98	2.29	1.76	1.89	2.22	2.11	2.08	2.21	1.17
flamme2	0.02	6.4E+5	6.3E+5	0.56	0.57	0.45	1.14	1.18	1.21	1.12	1.12	1.16	1.32	1.21	1.14	1.16
plasma	0.01	4.6E+5	5.1E+5	0.82	0.89	0.82	0.94	1.03	1.10	0.96	0.99	1.05	1.05	1.01	1.02	1.11
spxr2	0.04	1.4E+6	1.4E+6	0.73	0.86	0.66	1.45	1.65	1.68	1.41	1.58	1.61	1.63	1.67	1.53	1.12
spxu2	0.02	7.2E+5	7.9E+5	0.74	0.85	0.68	1.53	1.71	1.83	1.49	1.63	1.75	1.70	1.69	1.66	1.11
torso	0.05	2.1E+6	2.0E+6	0.89	1.09	0.84	2.77	3.05	4.03	2.71	2.83	3.90	3.99	3.68	4.10	1.44
sf2	0.07	2.4E+6	2.2E+6	1.16	1.12	0.94	1.13	1.25	1.17	1.17	1.25	1.19	1.35	1.38	1.19	1.19
bucky2r1	0.45	1.5E+7	2.0E+7	0.92	0.94	1.05	1.99	2.03	2.80	1.96	1.93	2.67	2.19	2.01	2.63	1.1
bucky2u1	0.34	1.0E+7	1.5E+7	1.24	1.19	1.41	2.22	2.09	3.19	2.19	1.99	3.05	2.45	2.06	3.01	1.11
fighterr2	1.00	3.3E+7	3.9E+7	0.90	0.99	0.88	2.33	2.31	3.00	2.26	2.19	2.88	2.62	2.33	2.94	1.13
fighteru2	0.47	1.7E+7	1.8E+7	0.80	0.94	0.75	2.19	2.31	2.75	2.13	2.19	2.64	2.47	2.33	2.67	1.13
flamme2r1	0.25	8.3E+6	1.0E+7	0.75	0.81	0.75	2.07	2.09	2.73	2.01	1.98	2.60	2.30	2.06	2.59	1.11

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL (bsp)			FastCOL (bsp)
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	vs OpenCCL
flamme2u1	0.20	6.6E+6	8.1E+6	0.75	0.83	0.76	2.12	2.14	2.79	2.05	2.03	2.66	2.35	2.10	2.64	1.11
plasmar1	0.20	6.4E+6	9.0E+6	1.10	1.06	1.30	2.24	2.17	3.27	2.20	2.06	3.13	2.46	2.15	3.09	1.1
plasmau1	0.19	6.1E+6	8.8E+6	1.30	1.18	1.49	2.25	2.13	3.28	2.21	2.03	3.14	2.47	2.12	3.09	1.1
postr1	0.17	6.4E+6	6.6E+6	0.77	0.91	0.76	1.86	2.13	2.44	1.82	2.04	2.36	2.08	2.21	2.37	1.12
postu1	0.10	3.7E+6	4.1E+6	0.81	0.93	0.81	1.91	2.09	2.54	1.86	2.00	2.44	2.12	2.16	2.41	1.11
sf2r1	0.54	1.6E+7	2.1E+7	1.20	1.19	1.23	2.23	2.12	2.82	2.19	2.02	2.74	2.47	2.08	2.68	1.11
sf2u1	0.21	5.8E+6	7.7E+6	1.76	1.43	1.70	2.33	1.98	2.87	2.30	1.90	2.81	2.58	2.00	2.76	1.11
spxr3	0.29	1.0E+7	1.1E+7	0.63	0.77	0.60	1.74	1.93	2.18	1.70	1.84	2.08	1.92	1.91	2.04	1.1
spxu3	0.15	5.5E+6	5.9E+6	0.62	0.76	0.60	1.79	1.99	2.27	1.74	1.88	2.16	1.96	1.96	2.12	1.1
torsor1	0.40	1.4E+7	1.5E+7	0.95	1.04	0.87	3.48	3.17	4.51	3.36	2.96	4.31	4.35	3.52	4.63	1.25
torsou1	0.95	3.0E+7	3.7E+7	1.13	1.11	1.04	5.17	3.89	6.86	4.97	3.59	6.59	7.17	4.65	7.36	1.39
mean 100K	0.01	2.1E+5	1.7E+5	0.95	0.92	0.90	1.23	1.48	1.31	1.21	1.44	1.30	1.37	1.46	1.23	1.11
median 100K	0.00	1.5E+5	1.4E+5	0.91	0.90	0.82	1.28	1.57	1.37	1.28	1.54	1.36	1.42	1.59	1.26	1.11
min 100K	0.00	2.9E+4	1.1E+4	0.79	0.62	0.74	1.01	1.06	0.98	1.01	1.03	1.01	1.06	1.05	0.99	1.06
max 100K	0.01	4.8E+5	3.5E+5	1.13	1.23	1.17	1.33	1.65	1.47	1.29	1.57	1.44	1.50	1.64	1.34	1.16
mean 1M	0.04	1.4E+6	1.5E+6	0.87	0.94	0.84	1.46	1.60	1.80	1.45	1.53	1.74	1.75	1.69	1.73	1.17
median 1M	0.04	1.2E+6	1.3E+6	0.85	0.93	0.85	1.29	1.45	1.44	1.29	1.41	1.40	1.49	1.52	1.36	1.15
min 1M	0.01	1.8E+5	1.7E+5	0.56	0.57	0.45	0.87	0.97	0.96	0.90	0.96	0.96	0.98	0.96	0.85	1.11
max 1M	0.10	3.1E+6	3.6E+6	1.16	1.15	1.25	2.77	3.05	4.03	2.71	2.83	3.90	3.99	3.68	4.10	1.44
mean 10M	0.37	1.2E+7	1.5E+7	0.98	1.01	1.00	2.37	2.28	3.14	2.31	2.16	3.02	2.75	2.35	3.06	1.14
median 10M	0.27	9.3E+6	1.1E+7	0.91	0.96	0.87	2.21	2.12	2.81	2.16	2.02	2.70	2.46	2.11	2.67	1.11
min 10M	0.10	3.7E+6	4.1E+6	0.62	0.76	0.60	1.74	1.93	2.18	1.70	1.84	2.08	1.92	1.91	2.04	1.10
max 10M	1.00	3.3E+7	3.9E+7	1.76	1.43	1.70	5.17	3.89	6.86	4.97	3.59	6.59	7.17	4.65	7.36	1.39

Table 22: CpuTree sur Opteron

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL (bsp)			FastCOL (bsp)
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	vs OpenCCL
spx	0.00	2.5E+4	8.5E+3	0.81	0.77	0.66	1.07	1.03	1.07	1.05	1.01	1.04	1.46	1.01	1.41	1.37
fighter	0.03	3.7E+5	1.6E+5	1.16	0.95	1.27	1.56	1.34	2.10	1.55	1.34	2.06	2.12	1.37	3.25	1.36
spxr1	0.01	2.2E+5	8.1E+4	0.80	0.92	0.70	1.34	1.42	1.68	1.31	1.37	1.58	1.74	1.39	2.04	1.3
spxu1	0.01	1.1E+5	4.7E+4	0.83	0.93	0.74	1.42	1.43	1.84	1.37	1.38	1.71	1.84	1.41	1.92	1.3
blunt	0.01	1.2E+5	5.1E+4	1.09	1.15	1.31	1.36	1.26	1.80	1.36	1.26	1.80	1.71	1.22	1.72	1.26
post	0.01	1.5E+5	4.4E+4	0.82	0.96	0.75	0.89	0.96	0.80	0.91	0.96	0.81	1.23	0.95	0.92	1.39
bucky2	0.06	1.3E+6	2.9E+5	0.81	1.00	0.71	1.04	1.51	0.97	1.04	1.47	0.98	1.28	1.47	1.04	1.24
fighterr1	0.17	2.0E+6	1.2E+6	1.06	1.00	1.08	1.81	1.51	2.82	1.75	1.47	2.63	2.44	1.49	3.88	1.35
fighteru1	0.12	1.6E+6	8.4E+5	0.95	0.96	0.92	1.69	1.49	2.61	1.65	1.44	2.47	2.24	1.46	3.09	1.32
flamme2	0.04	5.3E+5	2.1E+5	0.64	0.69	0.47	1.19	1.09	1.34	1.15	1.05	1.26	1.53	1.07	1.56	1.29
plasma	0.03	4.2E+5	1.2E+5	0.80	0.86	0.71	0.99	1.03	0.95	1.00	1.00	0.98	1.19	0.98	1.01	1.2
spxr2	0.08	1.1E+6	5.2E+5	0.84	0.93	0.77	1.45	1.43	2.11	1.44	1.38	2.02	1.84	1.39	2.49	1.26
spxu2	0.04	5.6E+5	2.8E+5	0.85	0.94	0.79	1.54	1.46	2.24	1.49	1.39	2.10	1.96	1.41	2.30	1.27
torso	0.11	1.2E+6	8.0E+5	0.91	0.90	0.85	2.50	1.94	3.34	2.40	1.84	3.18	4.71	2.16	4.90	1.89
sf2	0.15	2.0E+6	7.4E+5	1.14	1.05	1.25	1.19	1.16	1.26	1.24	1.16	1.36	1.58	1.15	1.51	1.33
bucky2r1	0.92	1.1E+7	6.6E+6	0.95	1.03	1.05	2.07	1.64	2.89	2.01	1.59	2.76	2.61	1.60	3.30	1.26
bucky2u1	0.68	7.4E+6	4.8E+6	1.23	1.17	1.41	2.27	1.66	3.17	2.21	1.63	3.06	2.82	1.65	3.56	1.24
fighterr2	1.92	2.0E+7	1.2E+7	0.99	0.93	0.85	2.35	1.61	2.87	2.26	1.56	2.72	2.96	1.58	3.49	1.26
fighteru2	0.93	1.0E+7	5.9E+6	0.91	0.92	0.79	2.22	1.62	2.74	2.13	1.56	2.59	2.90	1.59	3.33	1.31
flamme2r1	0.49	5.7E+6	3.5E+6	0.83	0.93	0.83	2.09	1.62	2.79	2.02	1.57	2.65	2.66	1.59	3.22	1.27
flamme2u1	0.39	4.4E+6	2.7E+6	0.83	0.93	0.83	2.10	1.63	2.82	2.02	1.57	2.67	2.72	1.59	3.30	1.29
plasmar1	0.40	4.4E+6	2.9E+6	1.07	1.11	1.24	2.28	1.71	3.22	2.23	1.66	3.13	2.91	1.68	3.75	1.28

Meshes	Original			Geometric			OpenCCL			FastCOL			FastCOL (bsp)			FastCOL (bsp)
	time (s)	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	time	L1	L2	vs OpenCCL
plasmau1	0.39	4.3E+6	2.8E+6	1.26	1.16	1.48	2.32	1.70	3.28	2.25	1.65	3.15	2.93	1.67	3.80	1.26
postr1	0.33	4.3E+6	2.3E+6	0.84	0.95	0.81	1.93	1.61	2.87	1.89	1.57	2.75	2.41	1.63	3.70	1.24
postu1	0.19	2.6E+6	1.4E+6	0.86	0.97	0.85	1.93	1.64	2.88	1.88	1.59	2.73	2.32	1.65	3.52	1.2
sf2r1	0.99	1.1E+7	6.6E+6	1.15	1.10	1.18	2.21	1.59	2.95	2.13	1.55	2.81	2.68	1.55	3.43	1.21
sf2u1	0.38	3.9E+6	2.3E+6	1.67	1.22	1.77	2.33	1.50	2.83	2.29	1.45	2.80	2.75	1.48	3.56	1.18
spxr3	0.56	7.5E+6	4.1E+6	0.70	0.90	0.72	1.77	1.57	2.44	1.71	1.52	2.33	2.20	1.53	2.69	1.24
spxu3	0.30	3.9E+6	2.2E+6	0.69	0.90	0.72	1.82	1.59	2.49	1.75	1.53	2.37	2.31	1.55	2.85	1.27
torsor1	0.81	7.1E+6	4.8E+6	1.05	0.92	0.84	3.23	1.90	3.28	3.09	1.81	3.14	5.06	2.01	5.18	1.57
torsou1	1.97	1.4E+7	1.1E+7	1.30	0.95	0.94	4.47	2.20	3.99	4.26	2.08	3.84	8.57	2.45	9.35	1.92
mean 100k	0.01	1.7E+5	6.9E+4	0.94	0.94	0.93	1.35	1.29	1.70	1.33	1.27	1.64	1.78	1.28	2.07	1.32
median 100K	0.01	1.2E+5	5.1E+4	0.83	0.93	0.74	1.36	1.34	1.80	1.36	1.34	1.71	1.74	1.37	1.92	1.30
min 100k	0.00	2.5E+4	8.5E+3	0.80	0.77	0.66	1.07	1.03	1.07	1.05	1.01	1.04	1.46	1.01	1.41	1.26
max 100k	0.03	3.7E+5	1.6E+5	1.16	1.15	1.31	1.56	1.43	2.10	1.55	1.38	2.06	2.12	1.41	3.25	1.37
mean 1M	0.08	1.1E+6	5.0E+5	0.88	0.93	0.83	1.43	1.36	1.84	1.41	1.32	1.78	2.00	1.35	2.27	1.35
median 1M	0.07	1.1E+6	4.0E+5	0.84	0.95	0.78	1.32	1.44	1.73	1.34	1.39	1.69	1.71	1.40	1.93	1.31
min 1M	0.01	1.5E+5	4.4E+4	0.64	0.69	0.47	0.89	0.96	0.80	0.91	0.96	0.81	1.19	0.95	0.92	1.20
max 1M	0.17	2.0E+6	1.2E+6	1.14	1.05	1.25	2.50	1.94	3.34	2.40	1.84	3.18	4.71	2.16	4.90	1.89
mean 10M	0.73	7.6E+6	4.8E+6	1.02	1.00	1.02	2.34	1.67	2.97	2.26	1.62	2.84	3.17	1.68	3.88	1.31
median 10M	0.53	6.4E+6	3.8E+6	0.97	0.95	0.85	2.22	1.62	2.88	2.13	1.57	2.75	2.73	1.60	3.51	1.26
min 10M	0.19	2.6E+6	1.4E+6	0.69	0.90	0.72	1.77	1.50	2.44	1.71	1.45	2.33	2.20	1.48	2.69	1.18
max 10M	1.97	2.0E+7	1.2E+7	1.67	1.22	1.77	4.47	2.20	3.99	4.26	2.08	3.84	8.57	2.45	9.35	1.92

Table 23: GpuIso sur GTX280

Meshes	Original		Geometric		OpenCCL		FastCOL		FastCOL vs
	time (ms)	uncoalesced	time	uncoalesced	time	uncoalesced	time	uncoalesced	OpenCCL
spx	0.61	3.20E+04	0.78	0.65	1.10	1.58	1.06	1.53	1.04
fighter	4.15	4.14E+05	1.02	1.49	1.52	4.02	1.53	4.02	1.00
spxr1	4.40	5.13E+05	0.88	0.99	1.72	3.73	1.62	3.54	1.06
spxu1	3.87	5.32E+05	0.94	1.01	1.83	3.74	1.78	3.45	1.03
blunt	6.37	7.78E+05	1.20	1.77	1.61	2.34	1.61	2.34	1.00
post	7.42	6.55E+05	0.97	1.27	0.77	0.76	0.83	0.89	0.93
bucky2	19.93	1.48E+06	0.73	0.98	0.80	0.84	0.83	0.89	0.96
fighterr1	87.75	6.36E+06	1.96	1.08	3.82	4.34	3.61	3.81	1.06
fighteru1	74.29	5.74E+06	1.73	0.91	3.46	3.81	3.29	3.37	1.05
flamme2	32.35	3.52E+06	0.57	0.39	1.25	1.65	1.16	1.48	1.08
plasma	17.45	1.55E+06	0.76	0.99	0.80	0.81	0.84	0.87	0.95
spxr2	49.96	5.43E+06	1.16	0.86	2.54	3.83	2.39	3.56	1.06
spxu2	50.80	6.03E+06	1.17	0.86	2.55	3.78	2.41	3.42	1.06
torso	72.19	6.21E+06	2.03	0.94	4.34	3.79	4.03	3.36	1.08
sf2	76.93	7.04E+06	1.56	2.12	1.73	2.30	1.71	2.11	1.01
bucky2r1	391.98	4.87E+07	1.21	1.34	2.17	3.36	2.08	3.06	1.04
bucky2u1	345.38	4.66E+07	1.77	3.17	2.01	3.36	1.91	3.06	1.05
fighterr2	1047.67	5.66E+07	2.15	0.87	5.00	3.84	4.64	3.33	1.08
fighteru2	959.61	5.70E+07	2.35	0.87	5.40	3.81	5.06	3.37	1.07
flamme2r1	553.30	5.53E+07	1.13	0.86	3.18	3.78	2.97	3.32	1.07

Meshes	Original		Geometric		OpenCCL		FastCOL		FastCOL vs
	time (ms)	uncoalesced	time	uncoalesced	time	uncoalesced	time	uncoalesced	OpenCCL
flamme2u1	672.66	5.52E+07	1.57	0.85	3.97	3.78	3.70	3.32	1.07
plasmar1	399.81	5.35E+07	1.43	1.77	2.35	3.38	2.22	3.06	1.06
plasmau1	399.81	5.35E+07	1.43	1.77	2.35	3.38	2.22	3.06	1.06
postr1	568.08	5.84E+07	1.28	0.91	3.69	4.18	3.33	3.56	1.11
postu1	643.48	5.47E+07	1.79	0.92	4.63	4.20	4.26	3.60	1.09
sf2r1	807.14	5.84E+07	2.51	1.49	4.63	4.27	4.34	3.84	1.07
sf2u1	679.39	6.60E+07	2.14	1.53	3.94	4.13	3.71	3.70	1.06
spxr3	708.26	5.55E+07	1.58	0.86	4.23	3.89	3.87	3.34	1.09
spxu3	757.60	5.92E+07	1.73	0.86	4.29	3.83	3.99	3.37	1.08
torsor1	1059.92	6.07E+07	2.41	0.94	6.38	4.14	5.91	3.62	1.08
torsou1	1032.55	5.97E+07	2.73	0.95	7.20	4.18	6.63	3.68	1.09
mean 100K	3.88	4.54E+05	0.96	1.18	1.56	3.08	1.52	2.97	1.02
median 100K	4.15	5.13E+05	0.94	1.01	1.61	3.73	1.61	3.45	1.03
min 100K	0.61	3.20E+04	0.78	0.65	1.10	1.58	1.06	1.53	1.00
max 100K	6.37	7.78E+05	1.20	1.77	1.83	4.02	1.78	4.02	1.06
mean 1M	48.91	4.40E+06	1.26	1.04	2.20	2.59	2.11	2.38	1.02
median 1M	50.38	5.59E+06	1.16	0.96	2.13	3.04	2.05	2.74	1.05
min 1M	7.42	6.55E+05	0.57	0.39	0.77	0.76	0.83	0.87	0.93
max 1M	87.75	7.04E+06	2.03	2.12	4.34	4.34	4.03	3.81	1.08
mean 10M	689.17	5.62E+07	1.83	1.25	4.09	3.85	3.80	3.39	1.07
median 10M	676.02	5.61E+07	1.75	0.93	4.10	3.84	3.79	3.36	1.07
min 10M	345.38	4.66E+07	1.13	0.85	2.01	3.36	1.91	3.06	1.04
max 10M	1059.92	6.60E+07	2.73	3.17	7.20	4.27	6.63	3.84	1.11

Table 24: GpuTree sur GTX280

Meshes	Original		Geometric		OpenCCL		FastCOL		FastCOL (bsp)		FastCOL (bsp)
	time (ms)	uncoalesced	time	uncoalesced	time	uncoalesced	time	uncoalesced	time	uncoalesced	vs OpenCCL
spx	0.50	1.37E+04	0.94	0.58	1.01	1.28	1.01	1.18	1.06	1.42	1.04
fighter	3.43	1.33E+05	1.14	0.92	1.44	1.99	1.45	1.99	1.67	2.08	1.16
spxr1	1.77	7.20E+04	0.90	0.86	1.19	2.03	1.16	1.84	1.34	2.11	1.12
spxu1	1.10	4.61E+04	0.90	0.86	1.18	1.98	1.13	1.77	1.28	2.03	1.09
blunt	1.38	5.75E+04	0.98	1.02	1.16	1.95	1.16	1.95	1.3	1.99	1.12
post	2.05	6.90E+04	0.91	0.86	1.08	1.39	1.14	1.71	1.32	2.02	1.21
bucky2	9.91	2.85E+05	0.81	0.74	1.20	1.44	1.18	1.37	1.34	1.42	1.11
fighterr1	37.29	6.30E+05	1.98	1.00	3.08	2.20	2.96	1.96	3.58	2.30	1.16
fighteru1	20.48	5.14E+05	1.29	0.95	2.08	2.17	2.02	1.90	2.45	2.29	1.18
flamme2	5.79	1.71E+05	0.55	0.60	1.17	1.42	1.13	1.31	1.31	1.46	1.12
plasma	4.68	1.35E+05	0.76	0.81	1.16	1.50	1.14	1.42	1.26	1.49	1.08
spxr2	9.25	3.38E+05	0.64	0.90	1.33	2.08	1.27	1.87	1.5	2.13	1.12
spxu2	5.18	1.86E+05	0.58	0.92	1.33	2.13	1.28	1.89	1.5	2.18	1.13
torso	11.08	6.84E+05	1.41	0.98	2.63	2.39	2.38	2.05	2.81	2.40	1.07
sf2	20.50	5.47E+05	1.01	1.06	1.24	1.37	1.23	1.24	1.4	1.33	1.13
bucky2r1	115.87	3.70E+06	0.75	0.93	1.77	1.99	1.69	1.75	2	1.99	1.13
bucky2u1	89.89	2.55E+06	1.07	0.97	1.98	2.03	1.88	1.78	2.24	2.05	1.13
fighterr2	327.00	7.57E+06	1.17	0.93	2.74	2.18	2.59	1.89	3.13	2.18	1.14
fighteru2	162.22	4.00E+06	1.02	0.95	2.67	2.25	2.51	1.95	3.02	2.25	1.13
flamme2r1	66.26	2.06E+06	0.57	0.89	1.82	2.08	1.73	1.82	2.03	2.08	1.12

Meshes	Original		Geometric		OpenCCL		FastCOL		FastCOL (bsp)		FastCOL (bsp)
	time (ms)	uncoalesced	time	uncoalesced	time	uncoalesced	time	uncoalesced	time	uncoalesced	vs OpenCCL
flamme2u1	51.56	1.62E+06	0.57	0.89	1.80	2.09	1.71	1.82	2	2.08	1.12
plasmar1	53.87	1.51E+06	0.91	0.96	1.95	2.05	1.86	1.79	2.16	2.06	1.11
plasmau1	53.87	1.51E+06	0.91	0.96	1.95	2.05	1.86	1.79	2.16	2.06	1.11
postr1	48.66	1.54E+06	0.84	0.93	1.76	2.32	1.69	2.01	2.07	2.37	1.17
postu1	29.09	9.45E+05	0.82	0.94	1.67	2.28	1.60	1.99	1.91	2.37	1.15
sf2r1	157.99	3.74E+06	1.23	1.07	2.29	2.06	2.18	1.78	2.58	2.03	1.13
sf2u1	78.42	1.25E+06	1.87	1.26	2.77	1.80	2.63	1.50	3.07	1.69	1.11
spxr3	73.66	2.93E+06	0.53	0.93	1.55	2.25	1.47	1.94	1.71	2.22	1.1
spxu3	40.39	1.52E+06	0.48	0.93	1.54	2.20	1.46	1.92	1.69	2.20	1.1
torsor1	128.37	3.42E+06	1.53	0.97	4.21	2.30	3.89	1.99	4.59	2.32	1.09
torsou1	287.68	7.32E+06	2.59	0.99	7.49	2.33	6.63	2.03	8.39	2.37	1.12
mean 100K	1.63	6.45E+04		0.85	1.20	1.85	1.18	1.75	1.33	1.93	1.11
median 100K	1.38	5.75E+04		0.86	1.18	1.98	1.16	1.84	1.30	2.03	1.12
min 100K	0.50	1.37E+04		0.58	1.01	1.28	1.01	1.18	1.06	1.42	1.04
max 100K	3.43	1.33E+05		1.02	1.44	2.03	1.45	1.99	1.67	2.11	1.16
mean 1M	12.62	3.56E+05		0.88	1.63	1.81	1.57	1.67	1.85	1.90	1.13
median 1M	9.58	3.12E+05		0.91	1.29	1.79	1.25	1.79	1.45	2.07	1.12
min 1M	2.05	6.90E+04		0.60	1.08	1.37	1.13	1.24	1.26	1.33	1.07
max 1M	37.29	6.84E+05		1.06	3.08	2.39	2.96	2.05	3.58	2.40	1.21
mean 10M	110.30	2.95E+06		0.97	2.50	2.14	2.34	1.86	2.80	2.15	1.12
median 10M	76.04	2.30E+06		0.95	1.95	2.14	1.86	1.85	2.16	2.13	1.12
min 10M	29.09	9.45E+05		0.89	1.54	1.80	1.46	1.50	1.69	1.69	1.09
max 10M	327.00	7.57E+06		1.26	7.49	2.33	6.63	2.03	8.39	2.37	1.17

Table 25: GpuIso sur 8800GTX

Meshes	Original		Geometric		OpenCCL		FastCOL		FastCOL vs
	time (ms)	Uncoalesced	time	Uncoalesced	time	Uncoalesced	time	Uncoalesced	OpenCCL
spx	1.08	2.05E+03	1.05	1.00	1.09	1.00	1.06	1.00	1.03
fighter	5.09	9.20E+03	1.01	1.00	1.00	1.00	1.01	1.00	1.00
spxr1	5.79	1.28E+04	0.97	1.00	0.97	1.00	0.99	1.00	0.98
spxu1	5.19	1.28E+04	0.99	1.00	0.97	1.00	0.98	1.00	0.99
blunt	10.37	2.82E+04	0.92	1.00	0.96	1.00	0.96	1.00	1.00
post	26.44	7.73E+04	1.06	0.87	0.99	1.00	0.99	1.00	0.99
bucky2	59.05	1.57E+05	0.96	0.98	0.96	1.00	0.96	1.00	1.00
fighterr1	112.97	1.29E+05	2.12	1.00	2.09	1.00	2.09	1.00	1.00
fighteru1	89.80	1.33E+05	1.74	1.00	1.71	1.00	1.73	1.00	0.99
flamme2	63.50	1.79E+05	0.99	1.00	1.01	1.00	1.00	1.00	1.01
plasma	58.47	1.64E+05	0.99	0.98	1.02	1.00	1.01	1.00	1.00
spxr2	57.48	1.29E+05	1.11	1.00	1.10	1.00	1.11	1.00	1.00
spxu2	59.31	1.42E+05	1.08	1.00	1.07	1.00	1.07	1.00	1.00
torso	99.36	1.36E+05	2.39	1.00	2.15	1.00	2.21	1.00	0.97
sf2	108.18	2.59E+05	0.98	1.00	1.01	1.00	1.03	1.00	0.98
mean 100K	5.50	1.30E+04	0.99	1.00	1.00	1.00	1.00	1.00	1.00
median 100K	5.19	1.28E+04	0.99	1.00	0.97	1.00	0.99	1.00	1.00
min 100K	1.08	2.05E+03	0.92	1.00	0.96	1.00	0.96	1.00	0.98
max 100K	10.37	2.82E+04	1.05	1.00	1.09	1.00	1.06	1.00	1.03
mean 1M	73.46	1.50E+05	1.34	0.98	1.31	1.00	1.32	1.00	0.99
median 1M	61.41	1.39E+05	1.07	1.00	1.04	1.00	1.05	1.00	1.00
min 1M	26.44	7.73E+04	0.96	0.87	0.96	1.00	0.96	1.00	0.97
max 1M	112.97	2.59E+05	2.39	1.00	2.15	1.00	2.21	1.00	1.01

Table 26: GpuTree sur 8800GTX

Meshes	Original		Geometric		OpenCCL		FastCOL		FastCOL (bsp)		FastCOL (bsp)
	time (ms)	uncoalesced	time	uncoalesced	time	uncoalesced	time	uncoalesced	time	uncoalesced	vs OpenCCL
spx	0.63	1.78E+02	1.00	1.00	1.00	1.00	1.01	1.00	1	0.98	1
fighter	4.22	9.08E+02	1.24	1.00	1.31	1.00	1.32	1.00	1.4	0.96	1.06
spxr1	2.14	5.44E+02	0.92	1.00	1.05	1.00	1.05	1.00	1.1	0.99	1.05
spxu1	1.29	3.20E+02	0.91	1.00	1.04	1.00	1.05	1.00	1.09	1.00	1.05
blunt	1.65	3.84E+02	1.00	1.00	1.07	1.00	1.07	1.00	1.14	0.97	1.06
post	2.42	5.71E+02	0.86	1.00	1.00	0.99	1.02	1.00	1.08	0.99	1.08
bucky2	10.87	2.98E+03	0.86	1.00	1.01	1.00	1.02	1.00	1.07	0.99	1.06
fighterr1	56.32	4.06E+03	1.88	1.00	3.69	1.00	3.67	1.00	3.98	0.99	1.08
fighteru1	34.00	3.39E+03	1.25	1.00	2.71	1.00	2.71	1.00	2.92	0.99	1.08
flamme2	7.05	1.60E+03	0.39	1.00	1.09	1.00	1.08	1.00	1.16	1.00	1.06
plasma	6.08	1.33E+03	0.70	1.00	1.10	1.00	1.11	1.00	1.16	0.99	1.05
spxr2	10.28	2.36E+03	0.40	1.00	1.12	1.00	1.11	1.00	1.19	1.00	1.06
spxu2	5.95	1.24E+03	0.38	1.00	1.12	1.00	1.13	1.00	1.21	1.00	1.08
torso	16.79	4.16E+03	1.65	1.00	1.98	1.00	2.02	1.00	2	0.91	1.01
sf2	25.29	6.02E+03	1.01	1.00	1.15	1.00	1.14	1.00	1.2	0.97	1.04
mean 100K	1.98	4.67E+02	1.01	1.00	1.09	1.00	1.10	1.00	1.15	0.98	1.05
median 100K	1.65	3.84E+02	1.00	1.00	1.05	1.00	1.05	1.00	1.10	0.98	1.05
min 100K	0.63	1.78E+02	0.91	1.00	1.00	1.00	1.01	1.00	1.00	0.96	1.00
max 100K	4.22	9.08E+02	1.24	1.00	1.31	1.00	1.32	1.00	1.40	1.00	1.06
mean 1M	17.51	2.77E+03	0.94	1.00	1.60	1.00	1.60	1.00	1.69	0.98	1.06
median 1M	10.58	2.67E+03	0.86	1.00	1.12	1.00	1.12	1.00	1.19	0.99	1.06
min 1M	2.42	5.71E+02	0.38	1.00	1.00	0.99	1.02	1.00	1.07	0.91	1.01
max 1M	56.32	6.02E+03	1.88	1.00	3.69	1.00	3.67	1.00	3.98	1.00	1.08

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 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