Minimizing Cache Misses Using Minimum-Surface Bodies

*Michael Frumkin and Rob F. Van der Wijngaart**

Abstract

A number of known techniques for improving cache performance in scientific computations involve the reordering of the iteration space. Some of these reorderings can be considered as coverings of the iteration space with the sets having good surface-to-volume ratio. Use of such sets reduces the number of cache misses in computations of local operators having the iteration space as a domain. First, we derive lower bounds which any algorithm must suffer while computing a local operator on a grid. Then we explore coverings of iteration spaces represented by structured and unstructured grids which allow us to approach these lower bounds. For structured grids we introduce a covering by successive minima tiles of the interference lattice of the grid. We show that the covering has low surface-to-volume ratio and present a computer experiment showing actual reduction of the cache misses achieved by using these tiles. For planar unstructured grids we show existence of a covering which reduces the number of cache misses to the level of structured grids. On the other hand, we present a triangulation of a 3-dimensional cube such that any local operator on the corresponding grid has significantly larger number of cache misses than a similar operator on a structured grid.

*Computer Sciences Corporation; M/S T27A-2, NASA Ames Research Center, Moffett Field, CA 94035-1000; e-mail: {frumkin,wijngaar}@nas.nasa.gov NASA Advanced Supercomputing Division NASA Ames Research Center

1 Introduction

A number of known techniques for improving cache performance in scientific computations involve the reordering of the iteration space. We present two new methods for partitioning the iteration space with minimum-surface cache fitting sets. Such partitionings reduce the number of cache misses to a level that is close to the theoretical minimum. We show that the coverings reduce the number of the misses by actual measurements of cache misses in computations of a second order stencil operator on structured three-dimensional grids.

A good tiling of the iteration space for structured discretization grids can be constructed by using the interference lattice of the grid. This lattice is a set of grid indices mapped into the same word in the cache or, equivalently, a set of solutions of the Cache Miss Equation [7]. In [2] we introduced a (generally skewed) tiling of the iteration space of the explicit operators on structured grids with parallelepipeds built on a reduced basis of the interference lattice. We showed that for lattices whose second shortest vector is relatively long the tiling reduces the number of cache misses to a value close to the theoretical lower bound. Constructing the skewed tiling, however, is a nontrivial task, and involves a significant overhead in testing whether a particular point lies inside the tile. Tiling a three-dimensional grid, for example, requires the determination of 29 integer parameters to construct the loop nest of depth six, and involves a significant branching overhead.

We start the paper with deriving lower bounds which any algorithm must suffer while computing an a local operator on a grid.

Then we introduce two new, coverings of structured grids: a covering with Voronoi cells and a covering with rectilinear parallelepipeds built on the vectors of successive minima of the interference lattice. In lattices with a relatively long shortest vector the cells of both coverings have near-minimal surface-to-volume ratios. Hence, the number of cache misses in the computations tiled with these cells is close to the theoretical minimum derived in [2]. Direct measurements of the cache misses show a significant advantage of the successive minima covering relative to the computations using the canonical loop ordering (maximally optimized by a compiler).

For the computations of local explicit operators on planar unstructured grids we construct a near-minimum-perimeter covering by applying the Lipton-Tarjan planar graph separator algorithm [8]. The perimeter-to-area ratio of the sets of this covering is $O(1/\sqrt{S})$, where S is the cache size. Lastly, we construct an unstructured grid that triangulates a 3-dimensional cube and show that the grid can not be covered with sets having small surface-to-volume ratios. The last result shows that any computation of an explicit operator on such 3-dimensional grid would suffer larger number of cache misses than a computation of a similar operator on a structured grid of the same size.

2 Cache Usage in Computations of Local Operators

Local operators on the grids. We consider the problem of computing a local explicit operator $q = Ku$ on data defined at the vertices of an undirected graph $G = (V, E)$

which we call grid. Locality of the operator K means that computation of $q(x)$, $x \in V$, involves values of $u(y)$, $y \in V$, where y is at a (graph) distance [1] at most r from x. This r is called the order of K and assumed to be independent of G. K is explicit, meaning that q and u are distinct arrays and, hence, the values of q can be computed in arbitrary order.

We consider structured and unstructured grids that have an explicit or implicit embedding into an Euclidean space. Structured grids are Cartesian products of line graphs, while edges of unstructured grids are defined explicitly by an adjacency matrix. We assume that the maximum vertex degree is independent of the total number of vertices. A grid is called planar if its vertices and edges can be embedded into a plane without edge intersections. A grid is called a triangulation of a body B if it can be represented as a 1-dimensional skeleton of a simplicial partition of B.

Cache Model. We consider a single-level, virtual-address-mapped, set-associative data cache memory, see [3]. The memory, with a total capacity of S words, is organized in z sets of a (*associativity*) lines each. Each line contains w words. Hence, the cache can be characterized by the parameter triplet (a, w, S), and its size S equals $a * z * w$ words. A cache with parameters $(S/w, w, S)$ is called fully associative, and a cache with parameters $(1, w, S)$ is called direct-mapped.

The cache memory is used as a temporary fast storage of words used for processing. A word at virtual address A is fetched into cache location $(a(A), z(A), w(A))$, where $w(A) = A \bmod w$, $z(A) = (A/w) \bmod z$, and $a(A)$ is determined according to a replacement policy (usually a variation of *least recently used*). The replacement policy is not important within the scope of this paper since our lower bounds are valid for any replacement policy and upper bounds are true even for direct mapped cache.

The number of cache misses incurred in computation of K depends on the order in which elements of u are stored in the main memory. We assume that for structured grids an element $u(i_1, \ldots, i_d)$ is stored at address $A = i_1 + n_1 i_2 + n_1 n_2 i_3 + \cdots + n_1 \cdots n_{d-1} i_d$, where $n_1, \cdots, n_{d-1}$ are the grid sizes. For unstructured grids we don't assume any particular ordering of the grid points (and, hence, elements of u). Instead we choose an ordering that reduces the number of cache misses.

Replacement loads. A *cache miss* is defined as a request for a word of data that is not present in the cache at the time of the request. A *cache load* is defined as an explicit request for a word of data for which no explicit request has been made previously (a *cold load*), or whose residence in the cache has expired because of a cache load of another word of data into the exact same location in the cache (a *replacement load*). Cache load is used as a technical term for making some formulas and their proofs shorter. The definitions of cold and replacement loads are analogous to those of cold and replacement cache misses [7], respectively, and if w equals 1 they completely coincide.

Surface-to-volume ratio. One technique for minimization of the number of replacement loads is to cover the grid $G = (V, E)$ with conflict-free sets $V = \bigcup V_i$, $i = 1, \ldots, k$, $|V_i| = S$, that is, sets without vertices mapped to the same location in

[1] The graph distance is the length of a shortest path connecting two vertices. The length of the path is the sum of length of edges in the path.

cache. If we calculate q in all vertices of V_i before calculating it in vertices of V_j, $j > i$, a replacement load can occur only at vertices having neighbors in at least two sets (boundary vertices). We consider only bounded degree graphs, so if we can find a covering with sets having volumes $|V_i|$ close to S and a minimal number of boundary vertices $|\partial V_i|$ (and boundary edges) i.e., bodies with minimum surface-to-volume ratio, then the computation of K will have a number of replacement loads close to the minimum. On the other hand, the total partition boundary $\sum_{i=1}^{k} |\partial V_i|$ can be used to obtain a lower bound for the number of replacement loads, see section 3, cf. [2, 9].

3 A lower bound for cache misses for local operators

In this section we consider the following problem: for a given grid and a local operator K, how many cache misses must be incurred in order to compute $q = Ku$, where q and u are two arrays defined on the grid. We provide a lower bound for the number of cache misses in any algorithm, regardless of the order in which the grid points are visited for the computation of q. The lower bound contains the minimum surface-to-volume ratio of sets covering the grid. The ratio can be calculated for a number of grids: structured grids, planar unstructured grids, FFT-grids, expanders, and matrix multiplication grids described below. It may be shown that our lower bound is tight for all above-mentioned grids and in general for any grid which may be covered by sets having an optimal surface-to-volume ratio.

We use the following terminology to describe the operator K. Locality of K means that the value of q at the grid point x is a function of the values $u(y)$, where y is a grid point at the distance at most r from x (r is called radius of K). In this section we obtain a lower bound for an explicit operator of radius 1, which, obviously is a lower bound for operators of larger radii as well.

3.1 Pointwise Computations

Depending on the separability of the kernel of the local operator, it has to be computed in pointwise or edgewise fashion. If an operator has an unseparable kernel, then it requires values of u at all neighbor points simultaniously to compute a value of q. We call such computation pointwise. In this subsection we assume that computation of q on the grid $G = (V, E)$ is performed in a pointwise fashion, that is, at any grid point the value of q is computed before computation of the value of q at another point is started. Operators with separable kernels which can be computed edgewise are considered in the next section.

In order to compute the value of q at a grid point $\mathbf{x}$, the values of u at the neighbor points of $\mathbf{x}$ must be loaded into the cache (points $\mathbf{y}$ and $\mathbf{x}$ are neighbours if they are connected by a grid edge). If $\mathbf{x}$ is a neighbor of $\mathbf{y}$ and $u(\mathbf{y})$ has been loaded in cache to compute $q(\mathbf{z})$ but is dropped from the cache before $q(\mathbf{x})$ is computed, then $u(\mathbf{y})$ must be reloaded, resulting in a replacement miss.

For a given algorithm to estimate the number of elements, ρ, of array u that must be replaced, we partition V into a disjoint union of k sets V_i, with $V = \cup_{i=1}^{k} V_i$, in such a way that q is computed at all points of V_i before it is computed at any

point of V_{i+1}, see Figure 1. Let B_i^l and B_i^u be (possibly intersecting) subsets of V_i which have neighbors of $\cup_{j<i}V_j$ and $\cup_{j>i}V_j$ respectively. The set $B_i = B_i^l \cup B_i^u$ is the boundary of V_i.

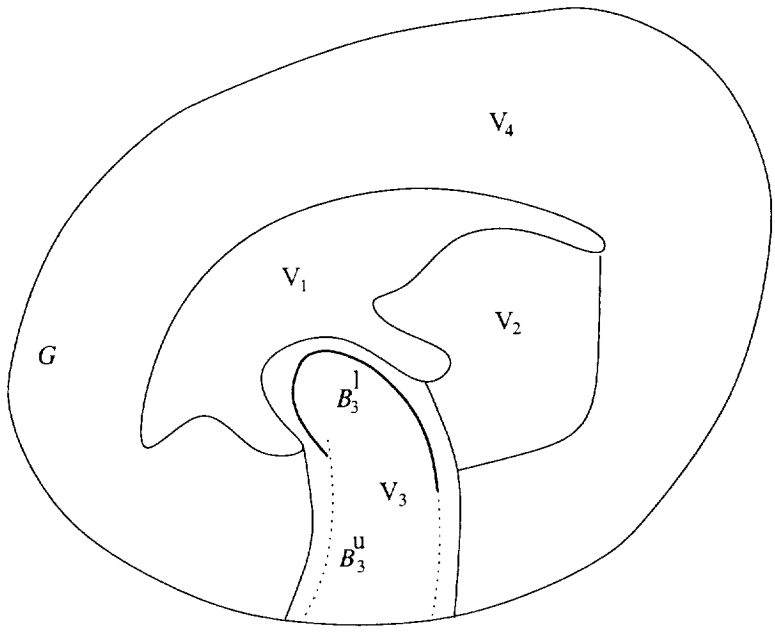

Figure 1. *The boundaries B_i^l and B_i^u of already computed values of q in a sequence of regions V_i. Reloading of some values of u on the boundary of V_3 (heavy and dotted lines) results in at least $|B_3^l| + |B_3^u| - 2S$ cache misses.*

For computation of q in V_i the values of u at points of B_i^l have to be present in cache. The values of u in B_i^l already are in cache since these values are necessary for computing q at the neigbor points in $\cup_{j<i}V_j$ and the computations in these points have been accomplished before computations in V_i have started. Since the cache can accomodate at most S of these values at least

$$\rho_i^l = |B_i^l| - S.$$

values have to be reloaded.

Symmetrically, for computation of q in V_i the values of u at points of B_i^u have to be present in cache. The values of u in B_i^u later will be required for computation of q in V_j, $j > i$. These computations started after all computations in V_i have been finished. Since the cache can accomodate at most S of these values, then at least

$$\rho_i^u = |B_i^u| - S.$$

values have to be reloaded.

Hence, in each set of the partition at least

$$\rho_i = \rho_i^l + \rho_i^u \geq |B_i^l| + |B_i^u| - 2S \geq |B_i| - 2S$$

values have to be reloaded.

The total number of reloaded values in the course of computing q on the entire grid will be at least

$$\sum_{i=1}^{k} \rho_i \geq \sum_{i=1}^{k} |B_i| - 2kS.$$

Let v be the maximum number of points in a set with $3S$ points on its boundary. Then choosing partition $V = \cup_{i=1}^{k} V_i$ in such a way that $|V_i| = v$ we get

$$\rho \geq S\frac{|V|}{v}. \tag{5}$$

Thus we have the following result.

Theorem 1. *The number of cache misses in calculation of an explicit operator on a grid $G = (V, E)$ is*

$$\mu \geq \frac{1}{w}(|V| + \rho) \geq \frac{1}{w}|V|(1 + \frac{1}{3}\alpha(G))$$

where $\alpha(G)$ is the minimum of the surface-to-volume ratio over subsets of V with $3S$ points on the boundary.

Proof. We sum the number of replacements in (5) with the number of cold loads $|V|$ and notice that $\frac{S}{v} = 1/3\alpha(G)$. Then we notice that each cache miss results in a load of w words in the cache. □

3.2 Edgewise Computations

The pointwise calculation model used in the previous subsection is too restrictive in many cases. Using separability of the kernel the number of cache misses can be reduced. For example, from Theorem 1 it follows a bound of $\mu \geq c\frac{n^4}{S}$ for a matrix multiplication algorithm (since it is easy to see that $\alpha(MM) = O(\frac{n^2}{S})$). However, it is well known [9] that the number of cache misses for this problem is $\mu = \theta(\frac{n^3}{\sqrt{S}})$ in the general case, and the upper bound is achieved by a block matrix multiplication algorithm with block size $\sqrt{S}$. In this section we present lower bounds for more general computations where values of q may be updated multiple times in arbitrary order. We call these computations edgewise.

An edgewise computation is performed at the vertices of a bipartite graph $H = (V^I, V^O, E)$ where, V^I and V^O are two copies of V, and $(x, y), x \in V^I, y \in V^O$ is an edge in H iff x and y are neighbors in G or $x = y$. The values of u are given in the points of V^I and values of q have to be computed at the points of V^O. An edgwise computation is computation of a function of two variables corresponding to an edge of H. If a value at an end of the edge is not in cache then the computation suffers a cache miss. All edges of the grid should be computed. We want to estimate the number of cache misses that each computation of $q = Ku$ must suffer.

The arguments for the obtaining the lower bound of Theorem 1 can be modified by partitioning of the edges of H into disjoint sets $E = \cup_{i=1}^{k} E_i$ in such a way that computation of any edge in E_i precedes computation of any edge in E_{i+1} and boundary of E_i is at least $3S$. Here the boundary of an edge set E_i is the set of vertices incident to an edge in E_i and to an edge which is not in E_i. The surface-to-volume ratio of an edge set is the ratio of the number of boundary vertices to the number of edges. We define $\beta(H)$ to be the minimum surface-to-volume ratio of the edge sets in H having surface $3S$. The following result can be proved by exactly same arguments as used for Theorem 1.

Theorem 2. *The number of cache misses in a separable calculation of an explicit operator on a grid $G = (V, E)$ is*

$$\mu \geq \frac{1}{w}|E|(1 + \frac{1}{3}\beta(G))$$

where $\beta(G)$ be the minimum of the surface-to-volume ratio over subsets of E with $3S$ points on the boundary.

4 Structured Grids

4.1 Interference Lattice

Interference lattice. Let u be a d-dimensional array defined at the vertices of a structured d-dimensional grid of size $n_1 \cdots n_d$. Let L be a set in the index space of u having the same image in cache as the index $(0, \ldots, 0)$. L is a lattice in the sense that there is a generating set of vectors $\{\mathbf{b}_i\}$, $i = 1, \ldots, d$, such that L is the set of grid points $\{\sum_{i=1}^{d} x_i \mathbf{b}_i \,|\, x_i \in \mathbf{Z}\}$. We call L the *interference lattice* of u. It can be defined as the set of all vectors $(i_1, \ldots, i_d)$ that satisfy the Cache Miss Equation [7]:

$$(i_1 + n_1 i_2 + n_1 n_2 i_3 + \cdots + n_1 \cdots n_{d-1} i_d) \bmod S = 0\,.$$

We will use some geometrical properties of lattices. Let B be a convex body of volume V, symmetrical about the origin. The minimal λ_i such that $\lambda_i B$ contains i linear independent vectors of L is called the i^{th} successive minimum of a lattice L relative to B. A theorem by Minkowski, see [1] (Ch. VIII, Th. V), asserts that

$$\frac{2^d}{d!V} \leq \frac{\prod_{i=1}^{d} \lambda_i}{\det L} \leq \frac{2^d}{V}. \tag{9}$$

Note that the ratios of lattice successive minima relative to the unit cube and to the unit ball can be bounded: $1/d \leq \lambda_i^{cube}/\lambda_i^{ball} \leq d$. In the section 4.2 we use successive minima relative to the unit ball and in the section 4.3 we use successive minima relative to the unit cube. In any case we call $f = \lambda_d/\lambda_1$ the *eccentricity* of the lattice (not to be confused with eccentricity of a reduced basis, defined in [2], Section 4). The eccentricity relative to a ball and cube may differ by a factor of d^2 at most.

In [2] we have introduced a tiling by parallelepipeds built on a reduced-basis of the interference lattice, which decreases the number of the cache misses to a level close to the theoretical lower bound that we also derived. Measurement shows that this tiling has significantly fewer cache loads than a compiler-optimized code. However, it has a high computational cost, since it depends on a significant number of integer parameters (29 integers for a 3D grid), and its implementation scans through a significant number of the grid points to select those suitable for cache conflict-free computations. This prompts us to consider tilings with Voronoi cells and with successive minima parallelepipeds. We show that these tilings have good surface-to-volume ratio if the lattice has a small eccentricity.

4.2 Voronoi Tiling

A Voronoi tiling is a tiling of the grid by completed cells C (Voronoi tile) of the Voronoi diagram. For each lattice point x a Voronoi cell is the set of points which are closer to x than to any other lattice point. All integer points inside each Voronoi cell are mapped into the cache without conflicts. Voronoi cells may not form a tiling since some integer points can be located on a cell boundary. There are many qualitatively equivalent ways to complete the cells to form a tiling. One way is to choose a basis in the space of the lattice and assign an integer point to the cell whose center is lexicographically closest to the point.

In order to estimate the surface-to-volume ratio of a Voronoi cell C we note that the completed Voronoi cells form a tiling of space. Hence, the volume of C equals the determinant of the lattice, which is equal to S, see [2]. On the other hand let $C_{\mathbf{o}}$ be a Voronoi cell centered at $\mathbf{o}$. Each vertex v of $C_{\mathbf{o}}$ is equidistant from d lattice points. Let r be that distance. According to the definition of the Voronoi cell, the ball of radius r, centered at v, contains no other lattice points. Hence $r \leq R$, where R is a radius of a *maximal ball* of the lattice (a lattice points free ball of the maximal radius). Hence, $C_{\mathbf{o}}$ is contained in a ball of radius R, centered at $\mathbf{o}$. Thus, the surface area of $C_{\mathbf{o}}$ is bounded by the surface of a ball of the radius R, which equals $dV_d dR^{d-1}$ where $V_d = \frac{\pi^{d/2}}{\Gamma(1+d/2)}$ is the volume of the unit d-dimensional ball (see [1] Ch. IX.7).

We estimate the radius of the maximal ball R by induction on the dimension of a sublattice. Let R_i the radius of the maximal ball inscribed into the lattice L_i built on the first i minima vectors of L. Then according Figure 2 we have

$$R_i^2 \leq (h_i/2)^2 + R_{i-1}^2 \leq (\lambda_i/2)^2 + R_{i-1}^2$$

and induction on i gives us the following assertion.

Lemma 3. *For the radius of a lattice points free ball in a d-dimensional lattice L we have the following relation:*

$$R^2 \leq 1/4 \sum_{i=1}^{d} \lambda_i^2. \tag{11}$$

where $\lambda_1 \leq \cdots \leq \lambda_d$ are successive minima of the lattice L.

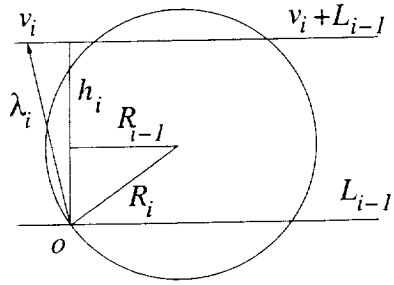

Figure 2. *The radius of maximal ball inscribed into L can be estimated through the radius R_{i-1} of the maximal ball inscribed into the lattice L_{i-1} built on the first $i-1$ minima vectors, and through the value of the last minimum $\lambda_i = |v_i|$. Here h_i is the distance between L_{d-1} and $v_i + L_{i-1}$*

Hence, for the surface area A of C we have the estimation

$$A(C) = dV_d R^{d-1} \leq d(\sqrt{d}/2)^{d-1} V_d \lambda_d d^{d-1} \leq d^{\frac{d+1}{2}} V_d^{1/d} f^{(d-1)^2/d} S^{(d-1)/d},$$

where we used the estimation $\lambda_d^d \leq \frac{2^d}{V_d} f^{d-1} S$ derived from (9), and the bound $R \leq \frac{\sqrt{d}}{2}\lambda_d$ which follows from (11). This implies the following result.

Theorem 4. *The surface-to-volume ratio of a Voronoi cell C can be estimated as:*

$$\frac{A(C)}{V(C)} \leq c_d f^{(d-1)^2/d} S^{-1/d}$$

where c_d is a constant depending on d only.

4.3 Successive Minima Tiling

The Voronoi cell tiling has cache-conflict-free tiles of maximum possible volume S, and of small surface-to-volume ratio. However, the tiles may have many faces, and it may be computationally expensive to scan through the grid points inside a tile. In this sense it is desirable to use rectilinear tiles. A *successive minima tiling* is a tiling by a Cartesian block built with use of successive minima lattice vectors of the unit cube. Such a block Q can be described by the system

$$|x_i| \leq b_i, \ i = 1, \ldots, d, \tag{14}$$

where $\lambda_1 \leq b_i \leq \lambda_d$.

The block Q can be constructed by the following "inflating" process. Take an initial cube of the form (14) with $b_i = 1$, $i = 1, \ldots, d$, and increment b_i until the face $x_i = b_i$ contains a lattice point. Continue to increment values of all b_j for which the face $x_j = b_j$ has no lattice points. At the end we obtain a block of the form (14) containing a lattice point on each of its faces and containing no lattice points inside except $\mathbf{o}$. In the best case each successive minimum vector will belong to one

of the faces of the block, meaning that $b_i = \lambda_i$ (after an appropriate reordering of the coordinates). On the other side, it is not difficult to construct a 3-dimensional lattice such that the block $b_1 = \lambda_1$, $b_2 = b_3 = \lambda_2 < \lambda_3$, so the volume of the block would be strictly less then $\lambda_1 \cdots \lambda_d$.

Any translation of the block $Q' = \frac{1}{2}Q$ obviously contains at most one lattice point and can be used for conflict free tiling. This block has a low surface-to-volume ratio if the lattice has bounded eccentricity, which can be seen from the following inequalities: $2A(Q') \leq d\lambda_d^{d-1}$ and $V(Q') \geq \lambda_1^d$. Hence, the surface-to-volume of the block can be estimated as follows:

$$\frac{A(Q')}{V(Q')} \leq 2df^{d-1}/\lambda_1 \leq d(d!V_d)^{1/d} f^{d-1} S^{-1/d}$$

since $\lambda_d = f\lambda_1$ and $\lambda_1 \geq 2(\frac{S}{d!V_d})^{1/d}$ as follows from (9).

As the representative example, the number of cache misses for tilings of 3-dimensional grids of sizes $40 \leq nx \leq 99$, $ny = 97$, $nz = 99$ with successive minima parallelepipeds is shown in Figure 3. Experiments were performed on an SGI Origin 2000 machine with a MIPS R10000 processor.

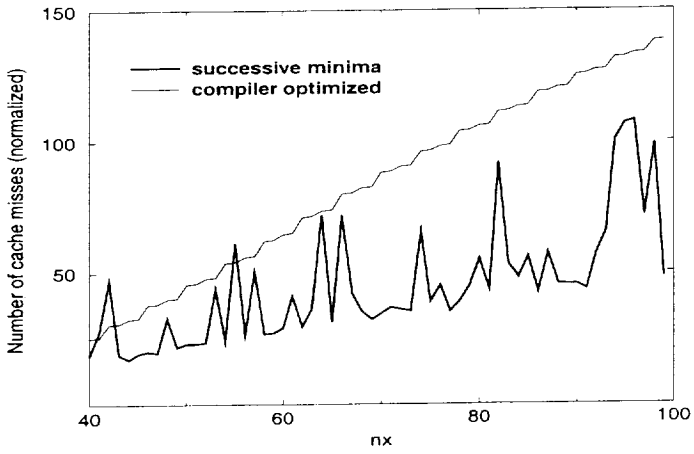

Figure 3. *Comparison of cache misses for the second order stencil operator as a function of the first dimension ($40 \leq nx \leq 99$, $ny = 97$, $nz = 99$). The top graph shows the number of cache misses for the compiler optimized nest. The bottom graph is obtained for tilings with successive minima parallelepipeds.*

5 Unstructured Grids

5.1 Lipton-Tarjan Covering

In this section we present a covering of a planar bounded degree grid with sets of size at most S with average perimeter-to-volume ratio equal $O(1/\sqrt{S})$. The tiling can be

used for computing an explicit first order operator on an n-vertex planar grid with $O(n/\sqrt{S})$ replacement loads. The tiling is based on the Lipton-Tarjan separator theorem asserting that any planar graph on n vertices has a vertex separator of the size $O(\sqrt{n})$. The separator can be constructed in $O(n)$ time [8].

We consider bounded degree unstructured grids, that is, grids having fixed maximum vertex degree d, independent of the grid size. (Calculation on unbounded vertex degree grids causes approximation problems and numerical instability. As a result, only bounded degree grids are used in numerical methods.) For bounded degree grids a node cut of size $O(n)$ has a corresponding edge cut of size $O(n)$, and vise versa. Hence, for bounded degree grids the Lipton-Tarjan separator theorem (see [8], section 2, Corollary 2) can be reformulated as follows: By removing $O(\sqrt{n})$ edges of an n-vertex planar graph it can be separated into connected disjoint subgraphs, each having at most $2n/3$ vertices.

We construct the covering by applying the Lipton-Tarjan theorem recursively. First, we choose any $C(n) \leq c_0\sqrt{n}$ cut of the original graph $G = (V, E)$, where c_0 is independent of n. According to Lipton-Tarjan theorem the cut can be chosen in such a way that it will split the graph into connected components $G_i = (V_i, E_i)$, $i = 1, \ldots, k$, $|V_i| \leq 2n/3$. Adding an extra step in this partition we can assume that $|V_i| \leq n/2$ while $C(n) \leq c_1\sqrt{n}$ for a bigger constant c_1. Then we recursively bisect each connected component $G_i = (V_i, E_i)$ while $|V_i| > S$. We will call this covering *Lipton-Tarjan covering.*

This partition process can be represented by a cut-tree T where nodes are partitioned connected components of the grid. A set is connected by edges with the nodes representing connected components obtained by removing the edges of the cut applied to the set. However, we do not include in T the connected components smaller than S in size which were not partitioned. To each node t of T we assign size $s(t)$ equal to the number of vertices in the set represented by t and weight $w(t) = \sqrt{s(t)}$. From the definition of the cut-tree it follows that size of each leaf (i.e. node having no children) exceeds S.

Lemma 5. *The total number of edges in all cuts is* $O(n/\sqrt{S})$.

Proof. In the Lipton-Tarjan covering then the total number of the edges in all cuts can be bounded by $O(\sigma(T))$, where

$$\sigma(T) = \sum_{t \text{ node of } T} w(t) \tag{16}$$

We use two properties of the weights:

$$\sum_{l \text{ leaf of } T} w(l) \leq n/\sqrt{S} \tag{17}$$

since the maximum of the $\sum \sqrt{s(l)}$ conditioned $\sum s(l) = n, s(l) \geq S + 1$ is attained at $s(l) = S + 1$ for all l. And the property

$$w(t) < 1/\sqrt{2} \sum_{\tau \text{ is son of } t} w(\tau) \tag{18}$$

which follows from Proposition 7 below.

Now $\sigma(T)$ can be estimated in two steps. First, we replace weights in each nonleaf t by the right hand side of (18) going bottom up from the leaves to the root of T. This operation will not decrease the total weight. Second, carry summation of new weights across nodes of T by noticing that each leaf l deposits into the total sum at most

$$w(l)(1 + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2^2}} + \ldots) = w(l)(2 + \sqrt{2}).$$

Hence from (17) it follows that

$$\sigma(T) \leq (2 + \sqrt{2})n/\sqrt{S} \tag{20}$$

and total number of edges in all cuts is $O(n/\sqrt{S})$. □

Now we prove the main result of this section. From a lower bound on the number of replacement loads for computations of explicit operators on structured 2-dimensional grids, see Section 3, cf. [2], it follows that the order of this bound can not be improved.

Theorem 6. *Any explicit first-order operator K on a bounded degree planar grid $G = (V, E)$ can be computed with $O(|V|/\sqrt{S})$ replacement loads.*

Proof. First, we reorder vertices of the grid in such a way that vertices of each set of the Lipton-Tarjan covering will occupy contiguous memory locations. Then we order the sets arbitrarily and compute K on each set separately. In this computation replacement loads can happen only at the vertices of the cuts of the grid. According to Lemma 5 the number of such vertices does not exceed $O(|V|/\sqrt{S})$. □

Proposition 7. *For any positive $v_1 \geq v_2 \geq \cdots \geq v_k$ such that $v_1 \leq \sum_{i=2}^{k} v_i$ the following inequality holds:*

$$\sum_{i=1}^{k} \sqrt{v_i} \geq \sqrt{2}\sqrt{\sum_{i=1}^{k} v_i}$$

Proof. The proof uses Jensen inequality, see [5], Ch 2.10, Th. 19: for $0 < r < s$

$$(\sum_{i=1}^{k} v_i^{\frac{1}{r}})^r \leq (\sum_{i=1}^{k} v_i^{\frac{1}{s}})^s$$

and in particular

$$(\sum_{i=1}^{k} v_i^2)^{1/2} \leq \sum_{i=1}^{k} v_i \leq (\sum_{i=1}^{k} v_i^{\frac{1}{2}})^2$$

Let $x_i = \sqrt{v_i}$, $i = 1, \ldots, k$, hence we have to prove

$$(\sum_{i=1}^{k} x_i)^2 \geq 2 \sum_{i=1}^{k} x_i^2$$

where $x_1 \leq (\sum_{i=2}^{k} x_i^2)^{1/2} \leq \sum_{i=2}^{k} x_i$.

Let j is the minimal index such that $x_j > \sum_{i=j+1}^{k} x_i$. Obviously, $1 < j < k$ then we have:

$$\begin{aligned}(\sum_{i=1}^{k} x_i)^2 &> \sum_{i=1}^{k} x_i^2 + 2x_1^2 + \cdots + 2x_{j-1}^2 + 2x_j(x_{j+1} + \cdots + x_k) \\ &\geq \sum_{i=1}^{k} x_i^2 + 2x_1^2 + \cdots + 2x_{j-1}^2 + 2x_{j+1}^2 + \cdots + 2x_k^2 > 2\sum_{i=1}^{k} x_i^2.\end{aligned}$$

□

5.2 Covering of Starry Grids

In this section we show that cache efficiency of starry multidimensional grids is the same as for structured grids.

Definition 8. *We call grid starry if it can be mapped to a grid in $\mathcal{R}^d$ that has the following two properties:*
$\star$ *the ratio of the length L of the longest edge of the grid to the length l of the shortest edge of the grid is limited by a constant independent of the number of grid points:*

$$L \leq c_0 l \tag{26}$$

$\star\star$ *there are no grid points at a distance shorter than l.*

A vertex of a starry grid can be adjacant only to grid points contained in a ball of radius L centered at this point. On the other hand, a ball of radius l around any vertex of a starry grid is free of other grid points. Hence, a starry grid has a bounded degree. Another simple property of starry grids is that any subgrid of a starry grid is a starry grid.

Starry grids are common in computations with particles distributed in a box and interacting via a short range potential. While there is no obvious way to verify that an abstract grid is starry , it is easy to verify that a grid in $\mathcal{R}^d$ is starry. One simple way to construct a starry grid is to delete some vertices and incident them edges from a structured grid. Also it is easy to see that any starry grid contained in a cube can be completed to a starry grid triangulating the cube.

We will show that a d-dimensional starry grid $G = (V, E)$ can be covered by sets of size S having at most $c|V|^{\frac{d-1}{d}}$ boundary points all together, see Theorem 11. This covering, as in the case of planar grids, is based on the Hyperplane Cut

Theorem , see Theorem 10, asserting that there is a hyperplane bisecting V into almost equal size parts while cutting at most $c|V|^{\frac{d-1}{d}}$ edges of the grid. From this we deduce the main result of the section.

Theorem 9. *Any explicit first-order operator K on a d-dimensional starry grid $G = (V, E)$ can be computed with $\mathcal{O}(|V|S^{-\frac{1}{d}})$ replacement loads.*

Proof. First, we reorder vertices of the grid in such a way that vertices of each set of the covering will occupy contiguous memory locations. Then we order the sets arbitrarily and compute K on each set separately. In this computation replacement loads can happen only at the vertices of the cuts of the grid. According to Theorem 11 the number of such vertices does not exceed $\mathcal{O}(|V|S^{-\frac{1}{d}})$. □

From a lower bound on the number of replacement loads for computations of explicit operators on structured d-dimensional grids, see [2], it follows that this bound can not be improved.

It is not difficult to see that any convex body has a small bisectior (for example a hyperplane orthogonal to a diametor of the body). It is not surprising that if a grid well represents the body then it has a small bisection width as well. We will construct sa et of $3d$ vectors and show that the bisecting hyperplane can be chosen to be a normal to one of these vectors. This actually gives an algorithm of complexity $\mathcal{O}(|V|^2)$ for finding such a bisector.

The following theorem is an analog of the Lipton-Tarjan cut theorem.

Theorem 10. *(Hyperplane Cut Theorem) For any starry grid $G = (V, E)$ embedded in $\mathcal{R}^d$ there is a hyperplanar cut of the size $\mathcal{O}(|V|^{\frac{d-1}{d}})$ separating vertices of the grid onto two sets of size at least $|V|/3$.*

Proof. Let us consider vectors of unit length $u_i \in \mathcal{R}^d$, $|u_i| = 1$, $i = 1, \ldots, k$ and slabs S_i bounded by hyperplanes orthogonal to u_i : $\{x \in \mathcal{R}^d | \lambda_i \leq u_i x \leq \mu_i\}$, trisecting V see Figure 4, that is

$$\#\{v \in V | u_i v \leq \lambda_i\} \geq |V|/3 \texttt{ while } \#\{v \in V | u_i v < \lambda_i\} < |V|/3, \qquad (27)$$

$$\#\{v \in V | u_i v \geq \mu_i\} \geq |V|/3 \texttt{ while } \#\{v \in V | u_i v > \mu_i\} < |V|/3.$$

Since we have the freedom to choose u_i we will choose them in such a way that all vertices have different projections on each u_i and the trisecting hyperplanes (27) exist. This choice of the slabs implies that each slab containes at least $|V|/3$ points while at most $|V|/3$ grid points can be contained strictly inside a slab. We will show that $u_1, \ldots, u_k$ can be chosen in such a way that at least one slab is wide, that is $h_i = \mu_i - \lambda_i \geq \mathcal{O}(|V|^{1/d})$. As explained at the end of the proof of this theorem, it follows from (26) that there exists a plane $H_i = \{v : u_i v = \eta_i\}$, $\lambda_i < \eta_i < \mu_i$ that intersects at most $O(|V|^{\frac{d-1}{d}})$ grid edges. This plane separates grid points into sets containing at least $|V|/3$ points.

Let V^i be a set of grid points contained in exactly i slabs and V^i_j be a subset

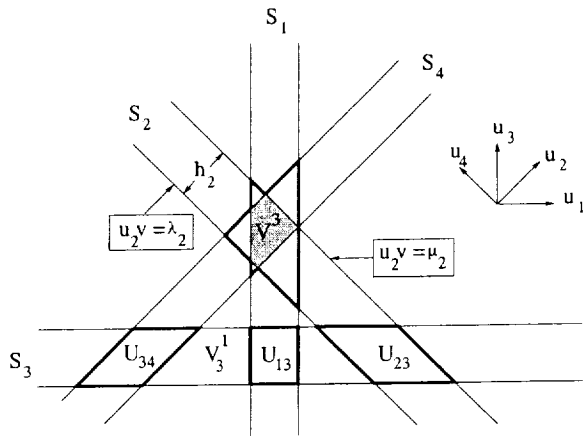

Figure 4. *Finding a wide bisecting slab. V^2 is the set of points inside the thick-lined polygons minus V^3 contained inside of the shaded polygon.*

of V^i contained in S_j. Since $\{V^i\}$ are disjoint sets and $\cup_{i\le k}|V^i| \subseteq V$ then

$$\sum |V^i| \le |V|.$$

Since each slab contains at least $|V|/3$ grid points then

$$\sum_j |V_j^i| \ge |V|/3.$$

If we sum all points in all slabs then each point in V^i will be counted exactly i times, hence

$$\sum_{1\le i\le k} i|V^i| \ge \frac{k}{3}|V|,$$

and

$$\sum_{i\ge d} i|V^i| \ge \frac{k}{3}|V| - \sum_{i<d} i|V^i| \ge (\frac{k}{3} - d + 1)|V|. \tag{32}$$

Let P_m, $m = (i_1, \ldots, i_d)$, $i_p \ne i_q$ be parallelepipeds formed by the intersection of d different slabs. Obviously, $\cup_{i\ge d} V^i \subseteq \cup_m P_m$. Let U_m be the number of grid points in P_m, then

$$\sum_m U_m = \sum_{i\ge d} \binom{d}{i} V^i \ge 1/d \sum_{i\ge d} iV^i.$$

Now if we choose $k = 3d$, then from (32) it follows that

$$\sum_m U_m \ge \frac{1}{d}|V|. \tag{34}$$

The balls B_l of radius of l centered at the grid points do not intesect then

$$\sum_m U_m \le (c_3 l^d)^{-1} \sum_m \mathtt{vol}(P_m + B_l), \tag{35}$$

where $c_3 l^d = \mathtt{vol}(B_l)$.

Now we exercise our freedom of choosing vectors u_i. We choose them to be normalized to length 1 rows of a $3d \times d$ Vandermond matrix

$$W = |(i-\alpha)^j|,\ i = 1, \ldots, 3d,\ j = 1, \ldots, d,\ 0 < \alpha < 1$$

We choose α in such a way that $u_i(v_p - v_q) \neq 0$ where v_p, v_q are different grid points. These constraints give us at most $3d|V|^2$ equations, and we choose α not to be a root any of them. Then each parallelepiped P_m may be described by a system of inequalities

$$-h_m \le D_m^{-1} W_m (x - x_m) \le h_m$$

where W_m is a square Vandermond matrix consisting of $m = (i_1, \ldots, i_d)$ rows of W, $h_m = (h_{i_1}, \ldots, h_{i_d})^t$, h_{i_j} is a halfwidth of P_m in the direction of the vector u_{i_j}, x_m is the center point of P_m and D_m is the normalizing matrix. Hence $P_m + B_l$ is contained in the ball of radius

$$D_m^{-1} |W_m^{-1}| |h_m + l| \le (3d)^d \sqrt{d} \prod_{i<j} (i-j)(h+l) < 3^{2d} d^{2d+1/2} (h+l).$$

where $h = \max_{1 \le i \le 3d} \{h_i\}$. Since $h > l$ then

$$\mathtt{vol}(P_m + B_l) \le c_4 h^d.$$

From this inequality together with (34) and (35) it follows

$$c_5 |V| \le \left(\frac{h}{l}\right)^d.$$

This results in the desired lower bound for the width of a slab

$$c_6 |V|^{1/d} l \le h.$$

Now we show that there is a hyperplane parallel to the boundaries of the slab which intersects at most $c_6 |V|^{\frac{d-1}{d}}$ edges of the grid. We slice the slab on $\lfloor \frac{h}{L} \rfloor \ge c_6 \frac{l}{L} |V|^{1/d} \ge c_7 |V|^{1/d}$ of slices of the width L. Since total number of grid points in the slab does not exceed $|V|$, then at least one slice contains less than $c_8 |V|^{\frac{d-1}{d}}$ grid points. If a grid edge intersects bisector H of the slice then at least one end of the edge is inside of the slice (since the slab is L thick and the length of any edge does not exceed L). Hence, total number of the edges intersecting H does not exceed $c_8 |V|^{\frac{d-1}{d}} \delta$, where δ is the degree of the grid, which is bounded for starry grids. Since H is inside of the slab then it separates the grid points into parts containing at least $|V|/3$ points each. □

Now we can formulate our covering result, which implies that computations of explicit operators on starry grids can be performed with the same cache efficieny as on structured grids.

Theorem 11. *Nodes of starry grid $G = (V, E)$ can be coverd by sets of size not exceeding S and with total boundary $\mathcal{O}(|V|/S^{1/d})$.*

Proof. The proof closely follows the proof of the main result of Section 5.1. As mentioned in the beginning of the section, any subgrid of a starry grid is starry, hence we construct the covering by applying the Hyperplane Cut Theorem recursively. First, we choose any bisector cutting at most $c_0|V|^{\frac{d-1}{d}}$ edges of the grid G, where c_0 is independent of G. According to the Hyperplane Cut Theorem the bisector can be chosen in such a way that it splits the grid into connected components $G_i = (V_i, E_i)$, $i = 1, \ldots, k$, $|V_i| \le 2|V|/3$. Adding an extra step in this partition we can assume that $|V_i| \le |V|/2$ while the number of edges cut by the bisector does not exceed $c_1|V|^{\frac{d-1}{d}}$ for a bigger constant c_1. Then we recursively bisect each connected component $G_i = (V_i, E_i)$ while $|V_i| > S$.

This partition process can be represented by a cut-tree T whose nodes are partitioned connected components of the grid. A set is connected by edges with the nodes representing connected components obtained by removing the edges of the cut applied to the set. However, we do not include in T not partitioned connected components smaller than S in size. To each node t of T we assign size $s(t)$ equal to the number of vertices in the set represented by t and weight $w(t) = s(t)^{\frac{d-1}{d}}$. From the definition of the cut-tree it follows that the size of each leaf exceeds S. The total number of the edges in all cuts can be bounded by $\mathcal{O}(\sigma(T))$, where

$$\sigma(T) = \sum_{t \text{ node of } T} w(t) \tag{42}$$

We use two properties of the weights:

$$\sum_{l \text{ leaf of } T} w(l) \le |V|/S^{\frac{d-1}{d}} \tag{43}$$

since the maximum of $\sum s(l)^{\frac{d-1}{d}}$ for sizes of the nodes normalized so that $\sum s(l) = |V|, s(l) \ge S + 1$ is attained at $s(l) = S + 1$ for all l. And the property

$$w(t) < c \sum_{\tau \text{ is son of } t} w(\tau) \tag{44}$$

for some $c < 1$ independent on the grid, which follows from Proposition 12.

Now $\sigma(T)$ can be estimated in two steps. First, we replace weights in each nonleaf t by the right hand side of (44) going bottom up from the leaves to the root of T. This operation will not decrease the total weight. Second, carry the summation of the new weights across nodes of T by noticing that each leaf l deposits into the total sum at most

$$w(l)(1 + c + c^2 + \ldots) = w(l)c_5.$$

Hence from (43) it follows that

$$\sigma(T) \leq c_5|V|/S^{\frac{d-1}{d}}. \tag{46}$$

meaning that the total number of edges in all cuts is $\mathcal{O}(|V|/S^{\frac{d-1}{d}})$. □

Proposition 12. *For any positive $v_1 \geq \cdots \geq v_k > 0$ such that $v_1 \leq \sum_{i=2}^k v_i$, the following inequality holds:*

$$\left(\frac{2}{d+1}\right)^{1/d} \sum_{i=1}^k v_i^{1/d} > \left(\sum_{i=1}^k v_i\right)^{1/d}.$$

Proof. The proof uses Jensen inequality, see [5], Ch 2.10, Th. 19: for $0 < r < s$

$$(\sum_{i=1}^k v_i^{1/r})^r \leq (\sum_{i=1}^k v_i^{1/s})^s$$

and in particular

$$(\sum_{i=1}^k v_i^d)^{1/d} \leq \sum_{i=1}^k v_i \leq (\sum_{i=1}^k v_i^{1/d})^d$$

Let $x_i = v_i^{1/d}$, $i = 1, \ldots, k$, hence we have to prove

$$(\sum_{i=1}^k x_i)^d \geq (d+1)/2 \sum_{i=1}^k x_i^d$$

where $x_1 \leq (\sum_{i=2}^k x_i^d)^{1/d} \leq \sum_{i=2}^k x_i$ and $x_1 \geq \cdots \geq x_k > 0$.

Let j be the minimal index such that $x_j > \sum_{i=j+1}^k x_i$. Obviously, $1 < j < k$ then we have:

$$\begin{aligned}
(\sum_{i=1}^k x_i)^d &> \sum_{i=1}^k x_i^d + dx_1^{d-1} \sum_{i=2}^k x_i + \cdots + dx_{j-1}^{d-1} \sum_{i=j}^k x_i + dx_j^{d-1} \sum_{i=j+1}^k x_i \\
&> \sum_{i=1}^k x_i^d + dx_1^d + \cdots + dx_{j-1}^d + dx_j^{d-1}(x_{j+1} + \cdots + x_k) \\
&\geq \sum_{i=1}^k x_i^d + dx_1^d + \cdots + dx_{j-1}^d + dx_{j+1}^d + \cdots + dx_k^d > (d+1)/2 \sum_{i=1}^k x_i^d.
\end{aligned}$$

□

5.3 Cache Unfriendly 3-dimensional Grid

In this section we construct a 3-dimensional grid of N vertices which has a subgrid of G the size cN that does not have small subsets with small surface-to-volume ratio. From this property, following the arguments of 3, it can be shown that for any computation of an explicit operator defined on the grid $\Omega(N/logN)$ replacement loads must occur.

Our construction is based on embedding an FFT butterfly graph into a triangulation of a 3-dimensional cube. The 2^n-point FFT graph, denoted as F_n, is a graph having $(n+1)2^n = N$ vertices arranged in $n+1$ layers of 2^n vertices each. In other words, vertices of F_n form an array (k,i), $0 \leq k \leq n, 0 \leq i \leq 2^n - 1$ and a vertex (k,i), $k < n$ is connected with vertices $(k+1,i)$ and $(k+1, i \bigoplus 2^k)$ where $i \bigoplus 2^k$ signifies taking the complement of k^{th} bit of i, see Figure 5.

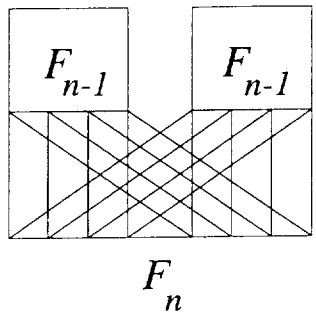

Figure 5. *A recursive construction of the FFT graph. F_n is built from two copies of F_{n-1} by adding $(n+1)^{th}$ layer of 2^n vertices and connecting them with vertices of n^{th} layer by the butterflies.*

Theorem 13. *The number of cache misses in calculation of an explicit operator on a grid F_n is*

$$\mu \geq \frac{1}{w} N(1 + \frac{c}{\log S}),$$

where c is a constant.

Proof. The theorem is a direct consequence of Theorem 1 and of an estimation of the boundary of vertex coverings of F_n given in Corollary 15. □

Lemma 14. *For any node subset $V \subset F_n$ we have*

$$|V| \leq 2|\delta V| \log |\delta V| \tag{53}$$

where δV is the right boundary of V, that is, the set of points in V either on the right boundary of F_n or having a right neighbor not in V.

Proof. Our proof of inequality (53) is based on induction and is similar to the proof of Theorem 4.1 in [9]. Let V be partitioned into three sets A, B and C, as shown in Figure 6. From the figure we can see that

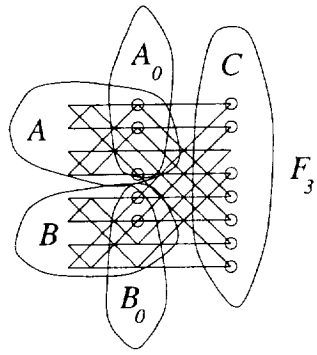

Figure 6. *Induction step for proving the surface-to-volume inequality of a subset in F_n. We can assume that $|A_0| \geq |B_0|$.*

$$|\delta V| \geq |\delta A| + |\delta B| + D + \min(0, |C| - 2|A_0|)$$

$$|V| \leq |A| + |B| + 2A_0 + \min(0, |C| - 2|A_0|)$$

where $D = |A_0| - |B_0|$. If $|C| \leq 2|A_0|$ then, by induction,

$$|V| \leq 2(|\delta A| \log |\delta A| + |\delta B| \log |\delta B| + |A_0|) \leq 2(|\delta V| \log |\delta V| - X)$$

where

$$X = |\delta A| \log(1 + \frac{|\delta B|}{|\delta A|} + \frac{D}{|\delta A|}) + |\delta B| \log(1 + \frac{|\delta A|}{|\delta B|} + \frac{D}{|\delta B|}) + D \log(|\delta A| + |\delta B| + D) - D - B_0.$$

Since $|B_0| \leq |\delta B|$ and $|B_0| \leq |A_0| \leq |\delta A|$ and either $\log(1 + \frac{|\delta A|}{|\delta B|} + \frac{D}{|\delta B|}) \geq 1$ or $\log(1 + \frac{|\delta B|}{|\delta A|} + \frac{D}{|\delta A|}) \geq 1$, or both, then $X \geq 0$. Hence $|V| \leq 2|\delta V| \log |\delta V|$.

If $y = \min(0, |C| - 2|A_0|) > 0$ then (53) follows from the fact that $v + y \leq 2(d + y) \log(d + y)$ if $v \leq 2d \log d$. □

Corollary 15. *Let $F_n = \bigcup V_i$, $i = 1, \ldots, k$, $|V_i| \leq S$ is any partition and $S \leq 2^{n/8}$. Then for the sum of boundaries of the sets of the partition the following inequality is true:*

$$\sum_{i=1}^{k} |\partial(V_i)| \geq \frac{N}{4 \log S}. \tag{58}$$

Proof. For any subset $V \subset F_n$ it follows from (58) that $|\delta V| \geq \frac{1}{2}|V|/\log|V|$, and we can estimate the sum of boundaries of the partition.

$$\sum_{i=1}^{k} |\partial(V_i)| \geq \sum_{i=1}^{k} |\delta(V_i)| - 2 \cdot 2^n \geq \frac{1}{2} \sum_{i=1}^{k} \frac{|V_i|}{\log|V_i|} - 2 \cdot 2^n$$
$$\geq \frac{1}{2 \log S} \sum_{i=1}^{k} |V_i| - 2 \cdot 2^n \geq \frac{N}{4 \log S}.$$

Where the last inequalty holds since $S \leq 2^{n/8}$. □

The FFT graph can be embedded into a triangulation of a 3-dimensional cube. A recursive construction of the FFT graph into triangulation of simplices is shown in Figure 7. The simplices can be embedded into a cube as shown in Figure 9 which then can be partitioned into parallelepipeds with further triangulation of each parallelepiped.

The butterflies connecting two last layers of F_n can be embedded into a triangulation of a simplex in such a way that the edges of the butterflies are mapped onto lines of pieces (t_0, t_3, b_7, b_4) and (t_7, t_4, b_3, b_0) of one of the ruled surfaces[2] and two skewed ruled surfaces (t_0, t_3, b_3, b_0) and (t_7, t_4, b_4, b_7). A simplex built on the appropriate vertices is separated by a ruled surface into two parts as shown in Figure 10, the top view is shown in Figure 8. The whole simplex (t_0, t_7, b_7, b_0) can be partitioned into the four simplices listed above and 5 primitive simplices: (t_3, t_4, b_3, b_4), (t_3, t_4, b_4, b_7), (t_3, t_4, b_3, b_0), (t_0, t_3, b_4, b_3) and (t_7, t_4, b_4, b_3). Each of the simplices (t_0, t_3, b_7, b_4), (t_7, t_4, b_3, b_0), (t_0, t_3, b_3, b_0) and (t_7, t_4, b_4, b_7) is separated by a ruled surface, hence it is sufficient to build a triangulation of a simplex separated by a ruled surface see Figure 10. This can be done in 3 steps: 1. adding vertices on the edges which are not parts of the ruled surface, 2. partitioning the simplex onto triangular prisms, and 3. triangulating each triangular prism into 3 primitive simplices.

It is easy to verify that the total number of vertices in the triangulation does not exceed $M = 3n2^n$ and the degree of each node does not exceed 16. Hence we have constructed a triangulation having the property declared at the beginning of this section.

6 Related Work and Conclusions

The reduction of cache misses in scientific computations is an active subject of research. One of the first lower bounds for data movement between primary and secondary storage was obtained in [9]. Recently the work has focused on developing compiler techniques to reduce the number of cache misses. In this direction we mention [7], where the notion of the cache miss equation (CME) and a tiling of

[2] A ruled surface is built by linearly parametrizing two crossing lines in 3D space and connecting corresponding points by lines. A ruled surface can be viewed as a hyperboloid containing the two crossing lines.

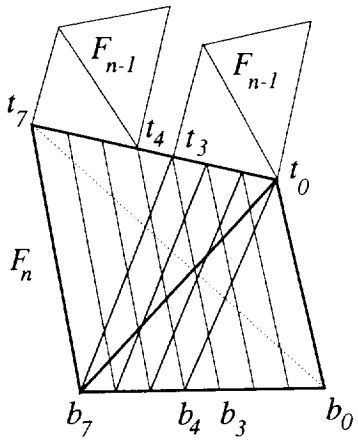

Figure 7. Recursive construction of embedding of FFT graph into a triangulations of a simplex.

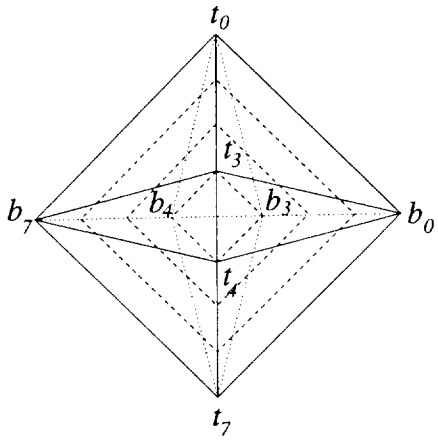

Figure 8. Embedding one layer of FFT graph into a triangulations of a simplex, top view.

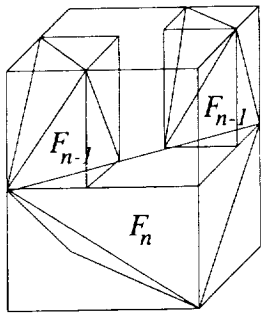

Figure 9. Recursive triangulation of a cube.

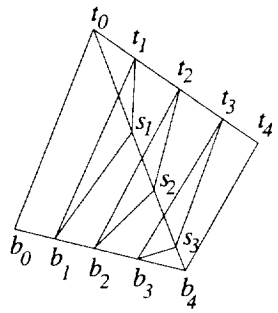

Figure 10. Triangulation of a simplex separated by a ruled surface via adding points s_1, s_2, s_3. Only partition not shadowed by the ruled surface is shown.

structured grids with conflict free rectilinear parallelepipeds were introduced. Some tight lower and upper bounds for computation of explicit operators on structured grids were obtained in [2], where a tiling with a reduced fundamental parallelepiped of the interference lattice was used for reduction of cache misses. Some practical methods for improving cache performance in computations of explicit operators are given in [10].

We showed that the reduction of cache misses for computations of explicit local operators defined on discretization grids is closely related to the problem of covering the grids with conflict free sets having low surface-to-volume ratio. We introduced two new coverings of structured grids: a covering with Voronoi cells and

a covering with rectilinear parallelepipeds built on the vectors of successive minima of the interference lattice. The cells of both coverings have near-minimal surface-to-volume ratios. Direct measurements of the cache misses show a significant advantage of the successive minima covering relative to computations using the natural loop order, maximally optimized by a compiler. We also showed that the computations of explicit operators on planar unstructured grids can be organized in such a way that the number of replacement loads is asymptotically close to one of the structured grids.

Bibliography

[1] J.W.S. Cassels. *An Introduction to the Geometry of Numbers.* Springer-Verlag, 1997, 344 P.

[2] M. Frumkin, R.F. Van der Wijngaart. *Efficient cache use for stencil operations on structured discretization grids.* NAS Technical Report NAS-00-015, November 2000, submitted to JACM.

[3] J.L. Hennessy, D.A. Patterson. *Computer Organization and Design.* Morgan Kaufmann Publishers, San Mateo, CA, 1994.

[4] H. Edelsbrunner. *Lectures in Geometry and Algorithms.* Urbana-Champaign, IL, 1994.

[5] G.H. Hardy, J.E. Littlewood, G. Polya. *Inequalities* 1934.

[6] K. Leichtweiß. *Konvexe Mengen.* Springer-Verlag, 1980.

[7] S. Gosh, M. Martonosi, S. Malik. *Cache Miss Equations: An Analytical Representation of Cache Misses.* ACM ICS 1997, pp. 317–324.

[8] R.J. Lipton, R.E. Tarjan. *A Separator Theorem for Planar Graphs.* SIAM J. Appl. Math, Vol. 36, No. 2, April 1979, pp. 177-189.

[9] J.W. Hong, H.T. Kung. *I/O Complexity: The Red-Blue Pebble Game.* IEEE Symposium on Theoretical Computer Science, 1981, pp. 326-333.

[10] G. Rivera, C.W. Tseng. *Tiling Optimizations for 3D Scientific Computations.* Proc. Supercomputing 2000, Dallas, TX, November 2000.