c-Perfect Hashing Schemes for Binary Trees, with Applications to Parallel Memories

(Extended Abstract)

Gennaro Cordasco[1], Alberto Negro[1], Vittorio Scarano[1], and Arnold L.Rosenberg[2]

[1] Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, 84081, Baronissi (SA) – Italy
{alberto,vitsca,cordasco}@dia.unisa.it
[2] Dept. of Computer Science, University of Massachusetts Amherst
Amherst, MA 01003, USA
rsnbrg@cs.umass.edu

Abstract. We study the problem of mapping tree-structured data to an ensemble of parallel memory modules. We are given a "conflict tolerance" c, and we seek the smallest ensemble that will allow us to store *any* n-vertex rooted binary tree with no more than c tree-vertices stored on the same module. Our attack on this problem abstracts it to a search for the smallest *c-perfect universal graph for complete binary trees*. We construct such a graph which witnesses that only $O\left(c^{(1-1/c)} \cdot 2^{(n+1)/(c+1)}\right)$ memory modules are needed to obtain the required bound on conflicts, and we prove that $\Omega\left(2^{(n+1)/(c+1)}\right)$ memory modules are necessary. These bounds are tight to within constant factors when c is fixed—as it is with the motivating application.

1 Introduction

Motivation. This paper studies the efficient mapping of data structures onto a parallel memory system (*PMS*, for short) which is composed of several *modules* that can be accessed simultaneously (by the processors of, say, a multiprocessor system). Mapping a data structure onto a PMS poses a challenge to an algorithm designer, because such systems typically are single-ported: they queue up simultaneous accesses to the same memory module, thereby incurring delay. The effective use of a PMS therefore demands efficient mapping strategies for the data structures that one wishes to access in parallel—strategies that minimize, for each memory access, the delay incurred by this queuing. Obviously, different mapping strategies are needed for different data structures, as well as for different ways of accessing the same data structure. As a simple example of the second point, one would map the vertices of a complete binary tree quite differently when optimizing access to levels of the tree than when optimizing access to root-to-leaf paths.

The preceding considerations give rise to the problem studied here. Given a data structure represented by a graph, and given the kinds of subgraphs one wants easy access to (called *templates*), our goal is to design a memory-mapping strategy for the items of a data structure that minimizes the number of simultaneous requests to the same memory module, over all instances of the considered templates.

Our Results. This paper presents the first strategy for mapping a binary-tree data structure onto a parallel memory in such a way that *any* rooted subtree can be accessed with a bounded number of conflicts. Our results are achieved by proving a (more general) result, of independent interest, about the sizes of graphs that are "almost" *perfect universal* for binary trees, in the sense of [5].

Related Work. Research in this field originated with strategies for mapping two-dimensional arrays into parallel memories. Several schemes have been proposed [3, 4, 8, 10] in order to offer conflict-free access to several templates, including rows, columns, diagonals and submatrices. While the strategies in these sources provide conflict-free mappings, such was not their primary goal. The strategies were actually designed to accommodate as many templates as possible.

Strategies for mapping tree structures considered conflict-free access for one *elementary* template—either complete subtrees or root-to-leaf paths or levels [6, 7] or combinations thereof [2], but the only study that even approaches the universality of our result—i.e., access to *any* subtree—is the C-template ("C" for "composite") of [1], whose instances are combinations of different numbers of distinct elementary templates. The mapping strategy presented for the C-templates' instances of size K, with M memory modules, achieves $O(K/M + c)$ conflicts.

Background. For any binary tree T: Size (T) is its number of vertices; Size (T, i) is its number of level-i vertices; Hgt (T) is its height (= number of levels).
For any positive integer c, a *c-contraction* of a binary tree T is a graph G that is obtained from T via the following steps.

1. Rename T as $G^{(0)}$. Set $k = 0$.
2. Pick a set S of $\leq c$ vertices of $G^{(k)}$ that were vertices of T. Remove these vertices from $G^{(k)}$, and replace them by a single vertex that represents the set S. Replace all of the edges of $G^{(k)}$ that were incident to the removed vertices by edges that are incident to S. The graph so obtained is $G^{(k+1)}$.
3. Iterate step 2 some number of times.

A graph $G_n = (V_n, E_n)$ is *c-perfect-universal* for the family $\mathbf{T}_n$ of n-vertex binary trees if every c-contraction of an n-vertex binary tree is a *labeled-subgraph* of G_n. By this we mean the following. Given any c-contraction $G^{(a)} = (V, E)$ of an n-vertex binary tree T, the fact that $G^{(a)}$ is a subgraph of G_n is observable via a mapping $f : V \rightarrow V_n$ for which each $v \in V$ is a subset of $f(V)$.

The simplest 1-perfect-universal graph for the family $\mathbf{T}_n$ is the height-n *complete binary tree*, $\mathcal{T}_n$, which is defined by the property of having all root-to-leaf paths of common length Hgt $(\mathcal{T}_n) = n$. The perfect-universality of $\mathcal{T}_n$ is witnessed by the identity map f of the vertices of any n-vertex binary tree to the vertices of $\mathcal{T}_n$. Of course, $\mathcal{T}_n$ is a rather inefficient perfect-universal graph for

$\mathbf{T}_n$, since it has $2^n - 1$ vertices, whereas each tree in $\mathbf{T}_n$ has only n vertices. It is natural to ponder how much smaller a 1-perfect-universal graph for $\mathbf{T}_n$ can be. The size—in number of vertices—of the smallest such graph is called the *perfection number* of $\mathbf{T}_n$ and is denoted $\mathrm{Perf}(\mathbf{T}_n)$. In [5], $\mathrm{Perf}(\mathbf{T}_n)$ is determined exactly, via coincident lower and upper bounds.

Theorem 1 ([5]) $\mathrm{Perf}(\mathbf{T}_n) = (3 - (n \bmod 2))\ 2^{\lfloor (n-1)/2 \rfloor} - 1$.

In this paper, we generalize the study of storage mappings for trees in [5] by allowing boundedly many *collisions* in storage mappings. We thus relax the "one to one" demands of perfect hashing to "boundedly many to one."

2 Close Bounds on $\mathrm{Perf}_c(\mathbf{T}_n)$

Our study generalizes Theorem 1 to c-perfect-universality, by defining $\mathrm{Perf}_c(\mathbf{T}_n)$, for any positive integer c, to be the size of the smallest c-perfect-universal graph for $\mathbf{T}_n$; in particular, $\mathrm{Perf}_1(\mathbf{T}_n) = \mathrm{Perf}(\mathbf{T}_n)$.

We first derive our upper bound on $\mathrm{Perf}_c(\mathbf{T}_n)$ by explicitly constructing a graph G_c that is c-perfect-universal for $\mathbf{T}_n$.

Theorem 2 *For all integers n and $c > 1$,*

$$\mathrm{Perf}_c(\mathbf{T}_n) \ < \ \left(2 + 2c^{1-1/c}\right) 2^{(n+1)/(c+1)} + O(n). \tag{1}$$

We construct our c-perfect-universal graph G_c via an algorithm $\mathcal{A}_c$ that colors the vertices of $\mathcal{T}_n$ in such a way that the natural vertex-label-preserving embedding of any n-vertex tree T into $\mathcal{T}_n$ (which witnesses T's being a subgraph of $\mathcal{T}_n$) never uses more than c vertices of any given color as homes for T's vertices. When we identify each like-colored set of vertices of $\mathcal{T}_n$—i.e., contract each set to a single vertex in the obvious way—we obtain the c-perfect-universal graph G_c, whose size is clearly an upper bound on $\mathrm{Perf}_c(\mathbf{T}_n)$.

Algorithm $\mathcal{A}_c$ proceeds in a left-to-right pass along each level of $\mathcal{T}_n$ in turn, assigning a unique set of colors, C_i, to the vertices of each level i, in a round-robin fashion. $\mathcal{A}_c$ thereby distributes the 2^i level-i vertices of $\mathcal{T}_n$ equally among the $|C_i|$ level-i vertices of G_c. Clearly, thus, $\mathrm{Size}\,(G_c) = \sum_i |C_i|$.

The remainder of the section is devoted to estimating how big the sets C_i must be in order for G_c to be c-perfect-universal for $\mathbf{T}_n$.

Auxiliary results. We begin by identifying some special subtrees of $\mathcal{T}_n$. Let m and i be integers such that[3] $\lceil \log m \rceil \leq i \leq n$, and let x be a binary string of length $i - \lceil \log m \rceil$. Consider the subtree $T_{(i,m)}(x)$ of $\mathcal{T}_n$ constructed as follows.

1. Generate a length-$(i - \lceil \log m \rceil)$ path from the root of $\mathcal{T}_n$ to vertex x.
2. Generate the smallest complete subtree rooted at x that has at least m leaves; easily, this subtree—call it $\mathcal{T}(x)$—has height $\lceil \log m \rceil$.

[3] All logarithms are to the base 2.

3. Finally, prune the tree so constructed, removing all vertices and edges other than those needed to incorporate the leftmost m leaves.

Easily, every tree $T_{(i,m)}$ has exactly m leaves, all of length i.

Lemma 1. *The trees $T_{(i,m)}$ are the smallest rooted subtrees of $\mathcal{T}_n$ that have m leaves at level i.*

Proof. (Sketch) For any level j of any binary tree T, we have $\text{Size}\,(T, j-1) \geq \left\lceil \frac{1}{2}\text{Size}\,(T,j)\right\rceil$. One verifies easily from our construction that the trees $T_{(i,m)}$ achieve this bound with equality. $\square$

Lemma 2. *For all i and m,*

$$2m + i - \lfloor \log m \rfloor - 1 \;\leq\; \text{Size}\left(T_{(i,m)}\right) \;\leq\; 2m + i - 1. \tag{2}$$

Proof. (Sketch) We merely sum the sizes of the complete subtree on $\mathcal{T}(x)$, plus the size of the various paths needed to "capture" them. Thus we have

$$\text{Size}\left(T_{(i,m)}\right) = (i - h_{p_{m-1}} + 1) \;+\; 2m \;-\; p_m \tag{3}$$

where p_m is the number of complete subtree on $\mathcal{T}(x)$ and $h_{p_{m-1}}$ is the height of the smallest complete subtree on $\mathcal{T}(x)$. We obtain the bound of the lemma from the exact, but nonperspicuous, expression (3), from the facts that there is at least one 1 in the binary representation of m and that $p_m \geq 1$. $\square$

Let us now number the vertices at each level of $\mathcal{T}_n$ from left to right and say that the *distance* between any two such vertices is the magnitude of the difference of their numbers.

Lemma 3. *For any integers $0 \leq i \leq n$ and $0 \leq \delta_i < i$, the size of the smallest rooted subtree of $\mathcal{T}_n$ that has m leaves at level i of $\mathcal{T}_n$, each at distance $\geq 2^{\delta_i}$ from the others, is no smaller than* $\text{Size}\left(T_{(i,m)}\right) + (m-1)\delta_i$.

Proof. If two vertices on level i are distance $\geq 2^{\delta_i}$ apart, then their least common ancestor in $\mathcal{T}_n$ must be at some level $\leq i - (\delta_i + 1)$. It follows that the smallest rooted subtree T of $\mathcal{T}_n$ that satisfies the premises of the lemma must consist of a copy of some $T_{(i-\delta_i,m)}$, with m leaves at level $i - \delta_i$, plus m vertex-disjoint paths (like "tentacles") from that level down through each of the δ_i levels $i - \delta_i + 1, i - \delta_i + 2, \ldots, i$ of $\mathcal{T}_n$. Equation (3) therefore yields:

$$\text{Size}\,(T) \;\geq\; m\delta_i + \text{Size}\left(T_{(i-\delta_i,m)}\right) \;=\; (m-1)\delta_i + \text{Size}\left(T_{(i,m)}\right). \quad \square$$

The upper bound proof. We propose, in the next two lemmas, two coloring schemes for $\mathcal{A}_c$, each inefficient on its own (in the sense that neither yields an upper bound on $\text{Perf}_c(\mathbf{T}_n)$ that comes close to matching our lower bound), but which combine to yield an efficient coloring scheme. We leave to the reader the simple proof of the following Lemma.

Lemma 4. *Let $\mathcal{T}_n$ be colored by algorithm $\mathcal{A}_c$ using 2^{δ_i} colors at each level i. If each*

$$2^{\delta_i} \;\geq\; \kappa_i \;\overset{def}{=}\; \lceil 2^i/c \rceil,$$

then any rooted n-vertex subtree of $\mathcal{T}_n$ engenders at most c collisions.

Lemma 5. *Let $\mathcal{T}_n$ be colored by algorithm $\mathcal{A}_c$ using 2^{δ_i} colors at each level i. If each*[4]

$$2^{\delta_i} \;\geq\; \lambda_i \;\overset{def}{=}\; \frac{1}{4}\,\mathrm{exp2}\left(\left\lceil \frac{n + \log(c+1) - i}{c} \right\rceil\right), \tag{4}$$

then any rooted n-vertex subtree of $\mathcal{T}_n$ engenders at most c collisions.

Proof. We consider a shallowest tree T that engenders $c+1$ collisions at level i of $\mathcal{T}_n$. Easily, the offending tree T has $c+1$ leaves from level i of $\mathcal{T}_n$. By the design of Algorithm $\mathcal{A}_c$, these leaves must be at distance 2^{δ_i} from one another. We can, therefore, combine Lemma 3, the lower bound of (2) (both with $m = c+1$), and (4) to bound from below the size of the offending tree T.

$$\begin{aligned}\mathrm{Size}\,(T) &\geq \mathrm{Size}\left(T_{(i,c+1)}\right) + c\delta_i \\ &\geq 2(c+1) + i - \lfloor \log(c+1) \rfloor - 1 + c\left\lceil \frac{n - 2c + \log(c+1) - i}{c} \right\rceil \;\geq\; n+1.\end{aligned}$$

Since we care only about ($\leq n$)-vertex subtrees of $\mathcal{T}_n$, the lemma follows. $\square$

A bit of analysis verifies that the respective strengths and weaknesses of the coloring schemes of Lemmas 4 and 5 are mutually complementary. This complementarity suggests the ploy of using the κ_i-scheme to color the "top" of $\mathcal{T}_n$ and the λ_i-scheme to color the "bottom." A natural place to divide the "top" of $\mathcal{T}_n$ from the "bottom" would be at a level i where $\kappa_i \approx \lambda_i$. Using this intuition, we choose level $i^\star \;\overset{\text{def}}{=}\; \left\lceil \frac{n-c+1}{c+1} + \log c \right\rceil$ to be the first level of the "bottom" of $\mathcal{T}_n$. (Since $c \leq n$, trivial calculations show that $n > i^\star$.) Using our hybrid coloring scheme, then, we end up with a c-perfect-universal graph G_c such that $\mathrm{Size}\,(G_c) \;=\; \sum_{j=0}^{i^\star-1} \kappa_j \;+\; \sum_{k=i^\star}^{n-1} \lambda_k$. Evaluating the two summations in turn, we find the following, under the assumption that $c > 1$ (since the case $c = 1$ is dealt with definitively in [5]; see Theorem 1).

$$\begin{aligned}\sum_{j=0}^{i^\star-1} \kappa_j = \sum_{j=0}^{i^\star-1} \lceil 2^j/c \rceil \;&\leq\; \sum_{j=0}^{i^\star-1} (2^j/c \;+\; 1) \;=\; \frac{1}{c}\,2^{i^\star} + i^\star - \frac{1}{c} \\ &\leq \frac{1}{c}\,\mathrm{exp2}\left(\left\lceil \frac{n-c+1}{c+1} + \log c \right\rceil\right) + O(n) \;<\; 2 \cdot 2^{(n+1)/(c+1)} + O(n)\end{aligned} \tag{5}$$

[4] To enhance the legibility of powers of 2 with complicated exponents, we often write $\mathrm{exp2}(X)$ for 2^X.

$$\begin{aligned}
\sum_{k=i^\star}^{n-1} \lambda_k = \sum_{k=i^\star}^{n-1} \mathrm{exp2}\left(\left\lceil \frac{n+\log(c+1)-k}{c} \right\rceil - 2\right) \;&<\; \frac{(c+1)^{1/c}}{2} \cdot \sum_{k=i^\star}^{n-1} 2^{(n-k)/c} \\
&<\; 2 \sum_{k=i^\star}^{i^\star+c-1} 2^{(n-k)/c} \;\leq\; 2c \cdot \mathrm{exp2}\left(\frac{n}{c} - \left(\frac{n-c+1}{c(c+1)} + \frac{\log c}{c}\right)\right) \\
&<\; 2\, c^{1-1/c} \cdot 2^{(n+1)/(c+1)}. \qquad (6)
\end{aligned}$$

The bounds (5, 6) yield the claimed upper bound (1) on $\mathrm{Perf}_c(\mathbf{T}_n)$. □

Because of space limitations, we defer the proof of the following lower bound to the complete version of this paper.

Theorem 3 *For all $c > 1$ and all n,*

$$\mathrm{Perf}_c(\mathbf{T}_n) \;>\; \mathrm{exp2}\left(\left\lfloor \frac{n+1}{c+1} - \frac{11}{3} \right\rfloor\right).$$

Acknowledgment. The research of A.L. Rosenberg was supported in part by US NSF Grant CCR-00-73401.

References

1. V. Auletta, S. Das, A. De Vivo, M.C. Pinotti, V. Scarano, "Optimal tree access by elementary and composite templates in parallel memory systems". *IEEE Trans. Parallel and Distr. Systs.*, 13, 2002.
2. V. Auletta, A. De Vivo, V. Scarano, "Multiple Template Access of Trees in Parallel Memory Systems". *J. Parallel and Distributed Computing* 49, 1998, 22–39.
3. P.Budnik, D.J. Kuck. "The organization and use of parallel memories". *IEEE Trans Comput.*, C-20, 1971, 1566–1569.
4. C.J.Colbourn, K.Heinrich. "Conflict-free access to parallel memories". *J. Parallel and Distributed Computing*, 14, 1992, 193–200.
5. F.R.K. Chung, A.L. Rosenberg, L. Snyder. "Perfect storage representations for families of data structures." *SIAM J. Algebr. Discr. Meth.*, 4, 1983, 548–565.
6. R. Creutzburg, L. Andrews, "Recent results on the parallel access to tree-like data structures – the isotropic approach", *Proc. Intl. Conf. on Parallel Processing*, 1, 1991, pp. 369–372.
7. S.K. Das, F. Sarkar, "Conflict-free data access of arrays and trees in parallel memory systems", *Proc. 6th IEEE Symp. on Parallel and Distributed Processing*, 1994, pp. 377–383.
8. D.H.Lawrie. "Access and alignment of data in an array processor". *IEEE Trans. on Computers*, C-24, 1975, 1145–1155.
9. R.J. Lipton, A.L. Rosenberg, A.C. Yao, "External hashing schemes for collections of data structures." *J. ACM*, 27, 1980, 81–95.
10. K.Kim, V.K.Prasanna. "Latin Squares for parallel array access". *IEEE Trans. Parallel and Distributed Systems*, 4, 1993, 361–370.
11. A.L. Rosenberg and L.J. Stockmeyer, "Hashing schemes for extendible arrays." *J. ACM*, 24, 1977, 199–221.
12. A.L. Rosenberg, "On storing ragged arrays by hashing." *Math. Syst. Th.*, 10, 1976/77, 193–210.