A Statistical Model of Skewed-Associativity

Pierre Michaud
IRISA/INRIA
Campus de Beaulieu, 35042 Rennes Cedex, France
pmichaud@irisa.fr

Abstract

This paper presents a statistical model for explaining why skewed-associativity removes conflicts better than set-associativity. We show that, with a high probability, 2-way skewed associativity emulates full associativity for working-sets up to half the cache size, and we show that 3-way skewed-associativity is almost equivalent to full associativity.

1. Introduction

Caching is a basic technique of computer architecture, used for many different purposes. Generally speaking, a cache is a dedicated memory which tries to hold a working-set of data into a limited space. The data may be actual program data, program instructions, address translations, branch predictions, memory dependence predictions, or whatever. Correspondingly, the cache may be a data cache, an instruction cache, a TLB, a BTB, etc... To be general, we speak in this study of a working-set of *objects*.

Sometimes a cache is not able to hold the whole of a working-set, but only a part of it. In this study, objects that the cache cannot hold are referred to as *missing objects*. Ideally, a cache must minimize the number of missing objects, while providing as fast an access as possible to objects lying in the cache. However these two requirements are usually incompatible. A fast access time, which implies hardware simplicity, is generally obtained at the cost of some number of missing objects. Missing objects may be due simply to the working-set being larger than the cache. They may also be due to the hashing function used to access the cache, in which case they are called *conflict* misses.

Set-associativity is the most widely used solution for overcoming the conflict miss problem. It consists of splitting the cache into several banks, each bank providing a possible location for an object and being accessed with the same hashing function. We show in this study that, while effective at solving the conflict miss problem for small working-sets, set-associativity is not very efficient when the working-set size is close to the cache size. We call this the *unit working-set* problem.

In 1993, André Seznec proposed a new solution for solving the conflict miss problem, *skewed-associativity*. The basic idea consists of indexing the banks with different hashing functions. Skewed-associativity was initially proposed for instructions and data caches [10]. It was later shown that skewed-associativity, unlike set-associativity, is rather insensitive to (good and bad) spatial locality effects [2], and is efficient for many types of objects [11].

However the reason why skewed-associativity works is not quite intuitive. The goal of this study is to provide a better understanding of the efficiency of skewed-associativity. Section 2 describes our approach to the problem and preliminary assumptions for the model. Section 3 focuses on set-associativity and emphasizes the unit working-set problem. Section 4 is the main part of the paper and focuses on skewed-associativity. We emphasize the placement problem and study three different placement algorithms. We show that, with a high probability, 2-way skewed-associativity emulates full-associativity for working-sets up to half the cache size, and we show that 3-way skewed-associativity is almost equivalent to full associativity.

2. Our Approach to the Problem, and Preliminary Assumptions

Although motivated by a microarchitecture question, this study makes no assumptions on applications properties (like spatial and temporal locality), neither on the type of objects. We adopt a statistical, "static" point of view.

We assume that an object is uniquely identified by a key, i.e., a value in $[0..A-1]$ (the key can be a block address, a page address, branch address and history bits, etc...). Our statistical model considers that, for a given working-set size n, all subsets of n keys in $[0..A-1]$ are equally likely. We assume a cache of size N and associativity degree w, i.e., the cache is split into w banks of N/w locations (if $w=1$, the cache is said to be *direct-mapped*). Each bank $i \in [1..w]$ is accessed through a hashing functions $\mathcal{H}_i$ which map keys

onto cache locations on bank i. Each object has w possible locations, one on each bank. Our goal is to determine the *average missing fraction* (**amf**), which we define as the average number of missing objects divided by the working-set size n ($amf \in [0..1]$).

The rationale behind this approach is that the amf gives information about the typical, average configuration. In particular, when the amf is very small, it means that the fraction of missing objects is very small for an overwhelming majority of working sets. On this example, the minority of working sets which have a significant number of missing objects might be termed "pathological". Whether a given working set is typical or pathological depends on $\mathcal{H}_i$. If some working sets are more likely to be encountered, one should take $\mathcal{H}_i$ so as to minimize the number of those working sets that fall in the pathological class [8, 13, 14]. But this problem is out of the scope of this study.

2.1. Balls-in-Urns Model

The statistical study of working sets can be simplified by viewing the cache problem as a balls-in-urns (aka balls-in-bins) problem. A ball is associated to a single object in the working-set, but an object may be associated to several balls. An urn generally represents a cache location, or a set of locations, and has room for an unlimited number of balls. We put into each urn the balls corresponding to the objects that **could** be placed in the cache location (or set of locations) associated to the urn. We assume all possible configurations of balls into urns are equiprobable. In order for the equiprobability of sets of keys to translate into equiprobability of balls-in-urns configurations, we need the following assumptions :

1. for every cache location $Y \in [0..N-1]$, the size of the preimage $\mathcal{H}_i^{-1}(Y) = \{X \mid \mathcal{H}_i(X) = Y\}$ equals A/N
2. $A/N \gg n$

The first assumption means that hashing functions $\mathcal{H}_i$ are balanced. The second assumption is generally true for most practical cache problems, as the key space is often several orders of magnitude larger than the working-set.

2.2. Classical Occupancy Problem

In this section, we introduce the formula that will be used throughout this study and which characterizes balls-in-urns configurations. We assume N distinguishable urns and n distinguishable balls. A ball can be in any urn, and each urn has room for an unlimited number of balls, so there are N^n distinct configurations. We assume all N^n configurations are equiprobable (Maxwell-Boltzmann statistics [3]). We search the average number μ_q of urns holding exactly q balls, with $q \geq 0$ (the case $q = 0$ is sometimes referred to in the literature as the *classical occupancy problem* [3]). We show in the appendix that, for $N \gg 1$ and $n \gg 1$, μ_q approximates a Poisson law:

$$\mu_q \simeq N \frac{(\frac{n}{N})^q}{q!} e^{-\frac{n}{N}} \tag{1}$$

Moreover, for N and n large enough, the distribution of the number of urns holding q balls is concentrated around the mean, that is, most configurations are close to the average configuration.

2.3. Monte Carlo Simulations

Formulas derived analytically in this study are mostly asymptotic approximations. We will use Monte Carlo simulations to validate experimentally these approximations, i.e., show that, under the assumptions of the model, and for realistic cache sizes, the formulas are reasonably accurate. In all our Monte Carlo simulations, we draw 10^4 random configurations, and compute average values from these configurations. The cache size for all simulations is fixed to 240 locations (we chose the highly composite number 240 for convenience, as we want the number of per-bank locations and the number of objects to be integer numbers).

3. Set-Associativity

In the case of set-associativity, all the banks are indexed with the same hashing function. Hence the set of possible locations for an object (*set of locations*, for short) is completely determined by its location on a particular bank. Let N be the total number of cache locations and n the number of objects in the working-set. The number of distinct sets of locations is $N_s = N/w$. In the remaining of this study, we define the working-set-to-cache-size ratio

$$\lambda = \frac{n}{N}$$

We can map the set-associativity problem onto a balls-in-urns problem as follows. There is one ball per object and one urn per set of locations. So we have n distinguishable balls and N_s distinguishable urns (we recall that each urn has room for an unlimited number of balls). For urns containing q balls with $q > w$, we count $q - w$ missing objects. Using Formula 1, replacing N with N_s, we obtain the number of empty cache locations

$$\sum_{q=0}^{w-1} N_s \frac{(\lambda w)^q}{q!} e^{-\lambda w} (w - q)$$

Note that this approximation is meaningful only for $N_s \gg 1$, that is, $w \ll N$. We define the *average unoccupied*

	Monte Carlo (N=240)						model					
λ	w=1	w=2	w=3	w=4	w=8	w=16	w=1	w=2	w=3	w=4	w=8	w=16
0.1	0.046	0.005	0.001	0.000	0.000	0.000	0.048	0.006	0.001	0.000	0.000	0.000
0.2	0.092	0.021	0.006	0.002	0.000	0.000	0.094	0.022	0.006	0.002	0.000	0.000
0.3	0.134	0.043	0.017	0.007	0.000	0.000	0.136	0.045	0.018	0.008	0.000	0.000
0.4	0.174	0.071	0.035	0.018	0.002	0.000	0.176	0.073	0.036	0.020	0.003	0.000
0.5	0.212	0.103	0.058	0.036	0.008	0.000	0.213	0.104	0.060	0.038	0.008	0.001
0.6	0.247	0.135	0.087	0.060	0.019	0.003	0.248	0.137	0.088	0.061	0.020	0.004
0.7	0.280	0.169	0.118	0.089	0.038	0.011	0.281	0.170	0.120	0.090	0.040	0.013
0.8	0.310	0.203	0.152	0.121	0.065	0.028	0.312	0.204	0.154	0.123	0.067	0.031
0.9	0.340	0.237	0.187	0.157	0.098	0.057	0.341	0.238	0.189	0.159	0.101	0.060
1.0	0.367	0.269	0.222	0.194	0.137	0.096	0.368	0.271	0.224	0.195	0.140	0.099
1.1	0.393	0.302	0.258	0.231	0.179	0.143	0.394	0.302	0.259	0.233	0.182	0.146
1.2	0.417	0.332	0.292	0.268	0.223	0.193	0.418	0.333	0.293	0.269	0.225	0.195
1.3	0.440	0.362	0.325	0.304	0.266	0.244	0.440	0.362	0.326	0.305	0.267	0.245

Table 1. Set-associativity : amf obtained with Monte Carlo simulations (left part) and with Formulas 2 and 3 (right part)

fraction (**auf**) as the number of empty cache locations divided by the total number of cache locations N. For a set-associative cache, the auf is

$$auf(w,\lambda) \simeq e^{-\lambda w} \times \sum_{q=0}^{w-1} \frac{(\lambda w)^q}{q!}(1-\frac{q}{w}) \qquad (2)$$

The amf can be obtained from the auf

$$amf(w,\lambda) = 1 - \frac{1-auf(w,\lambda)}{\lambda} \qquad (3)$$

Table 1 shows the amf obtained with Monte Carlo simulations and that obtained with Formulas 2 and 3. As long as $w \ll N$, the model provides a reasonably accurate approximation (the higher N, the better).

"Birthday-paradox" conflicts. For $\lambda \ll 1$, the amf can be approximated as a polynomial in λ of degree w. In particular, $amf(1,\lambda)$ has a first derivative which is positive at $\lambda = 0$. This weakness of direct-mapping is closely related to the so-called birthday paradox. These "birthday-paradox" conflicts are efficiently removed with 2-way set-associativity. However, as λ increases and gets closer to 1, it becomes more difficult for set-associativity to remove conflicts.

The unit working-set problem. Figure 1 shows the curve of the amf as a function of w for $\lambda = 1$, obtained with Formulas 2 and 3. The amf decreases slowly as w increases (assuming $w \ll N$). Even with $w = 64$, the amf is still significant, around 5%. Set-associativity is not an efficient solution for removing conflicts when the working-set size is close to the cache size. We refer to this problem as the *unit working-set* problem.

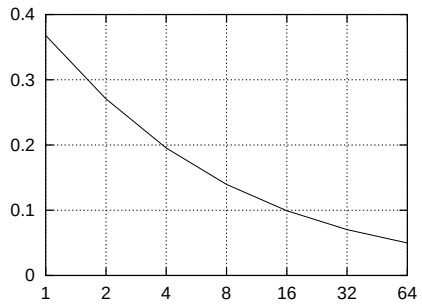

Figure 1. Set-associativity : amf as a function of w for $\lambda = 1$ (Formulas 2 and 3)

What the model does not explain lies in application characteristics. Some empirical knowledge about caches is not explained by the model. For instance, the *2:1 cache rule of thumb* [5], according to which a 2-way set-associative instruction/data cache is equivalent to a direct-mapped cache twice larger, is not confirmed by the model. This shows that such "rule" depends on applications characteristics (probably spatial locality in this case), and should not be taken as a property of set-associativity in general.

4. Skewed-Associativity

Skewed-associativity was introduced by André Seznec [10, 12, 2, 11] as an improvement over set-associativity. A w-way skewed-associative cache consists of w tables of

N/w entries each. As in the case of set-associativity, an object has w possible cache locations, one on each bank. However, unlike set-associativity, each bank $i \in [1..w]$ in a skewed-associative cache is indexed with a hashing function $\mathcal{H}_i$ different from that of the other banks. A major consequence of this skewing of the indexing functions is that the global cache occupancy depends on which of the w possible locations is chosen for an object. We refer to this as the *placement* problem.

It should be noted that the placement problem is distinct from the usual *replacement* problem. The usual replacement problem deals with the fact that some objects are accessed more frequently than others. When one among several objects must be evicted from the cache, a good non-oracle replacement policy, like LRU, tries to identify the object referenced the least frequently. With our static point of view, all the objects are "equal", and there is no replacement question.

The placement problem was irrelevant for set-associativity. For skewed-associativity, it is crucial. In this study, we will consider three placement methods: ordered-banks placement (**OBP**), quasi-optimal placement (**QOP**), and iterative random placement (**IRP**). The first placement algorithm provides an example of one-pass method. The second placement algorithm, on the other hand, provides a practical lower bound for the amf of a skewed-associative cache. The third placement algorithm corresponds to the practical case [2], and converges toward an optimal placement.

4.1. Balls-in-Urns Model

The balls-in-urns model for skewed-associativity is defined as follows. There is one urn per cache location (each urn has room for an unlimited number of balls). The set U of all urns is partitioned into w disjoint subsets of N/w urns, each subset corresponding to a different bank. A working-set of n objects is associated to w balls-in-urns configurations, one configuration per bank. Each object is represented by w balls, i.e., one ball per bank configuration. We are studying the global configuration resulting from the combination of these w per-bank configurations. The total number of global configurations is

$$\left(\frac{N}{w}\right)^{wn}$$

We assume all these global configurations are equiprobable. In addition to the conditions listed in Section 2 for each hashing function $\mathcal{H}_i$ separately, this assumption of equiprobability requires that for every w-tuple of cache locations $(Y_1, ..., Y_w)$,

	Monte Carlo (N=240)			**model**		
λ	w=2	w=3	w=4	w=2	w=3	w=4
0.1	0.001	0.000	0.000	0.001	0.000	0.000
0.2	0.006	0.000	0.000	0.006	0.000	0.000
0.3	0.017	0.001	0.000	0.018	0.001	0.000
0.4	0.035	0.004	0.000	0.036	0.005	0.000
0.5	0.059	0.013	0.002	0.060	0.014	0.002
0.6	0.088	0.029	0.008	0.089	0.030	0.008
0.7	0.121	0.053	0.021	0.122	0.054	0.023
0.8	0.155	0.084	0.046	0.156	0.085	0.048
0.9	0.191	0.121	0.081	0.192	0.122	0.082
1.0	0.227	0.161	0.123	0.228	0.162	0.124
1.1	0.263	0.203	0.169	0.264	0.204	0.170
1.2	0.298	0.244	0.216	0.298	0.245	0.217
1.3	0.331	0.285	0.262	0.331	0.285	0.262

Table 2. amf_{obp} obtained with Monte Carlo simulations (left) and with Formula 4 (right)

$$card\Big(\bigcap_{i=1}^{w} \mathcal{H}_i^{-1}(Y_i)\Big) = \frac{A}{(N/w)^w} \gg n$$

with A the size of the key space. For short, we will say that the hashing functions $\mathcal{H}_i$ are "globally orthogonal". With this assumption, the w per-bank balls-in-urns configurations have independent statistics.

It should be noted that, for $w > 2$, our definition of orthogonality is more restrictive than the pairwise orthogonality assumed in [2], namely that $card(\mathcal{H}_i^{-1}(Y_i) \cap \mathcal{H}_j^{-1}(Y_j)) = \frac{A}{(N/w)^2}$ for $i \neq j$), though global orthogonality implies pairwise orthogonality. Global orthogonality simplifies the analysis. However, simple pairwise orthogonality is simpler to implement in hardware. We tested simple pairwise orthogonality using the hashing functions defined in [12], and did not notice any impact on the amf. Although all $(N/w)^{wn}$ configurations are not possible with these functions, the configurations remaining do not seem to bias the statistics, as far as the amf is concerned. [1]

4.2. Ordered-Banks Placement (OBP)

The OBP algorithm is a one-pass algorithm. We assume that the banks are numbered from 1 to w, and the objects are numbered from 1 to n. The OBP algorithm tries to place objects $k = 1...n$ sequentially. If the location for object k on bank 1 is empty, we place object k on bank 1, else we look at bank 2. If the location for object k on bank 2 is empty, we place object k on bank 2, else we look at bank 3,

[1] We performed similar tests with the hashing functions defined in [2], which are not pairwise orthogonal, and we noticed a slight increase of the amf.

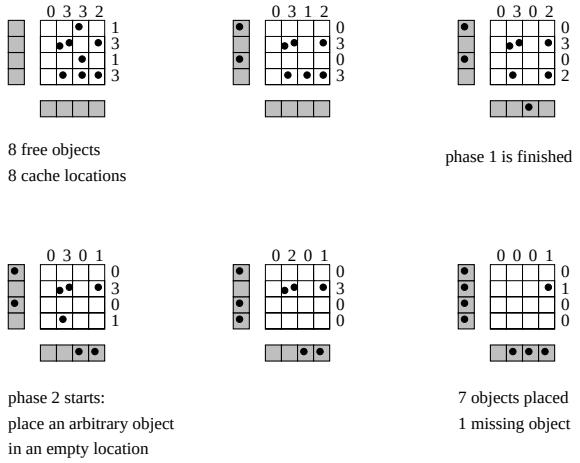

Figure 2. Example of QOP placement for w=2.

and so on ... If we were not able to place object k because all w locations were already occupied, we count object k in the missing objects, and we repeat the process for object $k+1$. Note that with this algorithm, banc i will receive, on average, more objects than bank $j > i$.

We use notation n_i for the total number of objects placed on banks 1 to i, and notation $f_i = \frac{n_i}{n}$ for the corresponding fraction. Using Formula 1 with $q = 0$ and replacing N with N/w, the average fraction of objects placed on bank 1 can be approximated as

$$f_1 \simeq \frac{1 - e^{-\lambda w}}{\lambda w}$$

i.e, we are able to fill all the locations on bank 1 that are not associated to empty urns. Assuming most configurations are close to the average configuration, we obtain f_i recursively

$$f_i \simeq f_{i-1} + \frac{1 - e^{-(1-f_{i-1})\lambda w}}{\lambda w}$$

Finally, the global amf_{obp} is

$$amf_{obp} = 1 - f_w \tag{4}$$

Table 2 shows the amf_{obp} obtained with Monte Carlo simulations and that obtained with Formula 4. We observe that Formula 4 is reasonably accurate. Comparing the numbers in Tables 1 and 2, we see that a skewed-associative cache with OBP has a lower amf than a set-associative cache with the same associativity. The behavior of a 2-way skewed-associative cache with OBP is between that of a 2-way and 4-way set-associative cache. This result shows that skewed-associativity is efficient even with a "poor" placement method. Nevertheless, OBP does not really solve the unit working-set problem.

4.3. Quasi-Optimal Placement (QOP)

The QOP algorithm places the objects one at a time, but with a global knowledge of all the objects. We call a *free* object an object that is not yet placed at a given "time" of the algorithm. Before the placement algorithm starts, all n objects are free. We denote $F(u)$ the set of **free** objects that have balls in urn u. We denote $V_t \subset U$ the subset of urns corresponding to empty locations at time t. Initially, $V_0 = U$. The QOP algorithm is defined recursively as follows:

1. If there exists any urn $u \in V_t$ such that $card(F(u)) = 1$, select one such u and place the object from $F(u)$ in the location associated to u. After that, $V_{t+1} = V_t - \{u\}$.
2. **Otherwise**, if there exists any urn $u \in V_t$ such that $card(F(u)) > 1$, select one such u, select an object from $F(u)$, and place it in the location associated to u. After that, $V_{t+1} = V_t - \{u\}$.

When we have several possibilities for choosing an urn or an object, we choose according to a predefined order. The algorithm ends when it is no longer possible to place new objects in this way. Free objects remaining at that time are missing objects. Figure 2 shows how the QOP algorithm works on an example assuming $w = 2$ (a dot represents an object, its coordinates on the grid represent its possible cache locations, each location being associated with an urn. Numbers on the top and right of the grid mean $card(F(u))$ for the associated urn).

Table 3 shows the amf_{qop} obtained with Monte Carlo simulations. If we focus on row $\lambda = 0.9$, we see that the unit working-set problem is practically solved for an associativity degree of 3.

It should be emphasized that the QOP algorithm is rather loose : not specifying how urns and objects are ordered leaves many degrees of freedom for placing objects, and leads to many possible placements.

The QOP algorithm is optimal in the case $w = 2$. It can be demonstrated that the QOP algorithm provides an optimal placement in the case $w = 2$. We distinguish two phases of the algorithm (we found the algorithm easier to analyze this way). Phase 1 of the algorithm ends the first time all urns $u \in V_t$ are such that $card(F(u)) \neq 1$. After that time, and until the algorithm ends, this is phase 2. It should be noted that the objects that are placed at the end of phase 1 do not depend on the way we selected the urns. They depend solely on the initial configuration. The placements done during phase 1 are optimal because the objects placed occupy locations that could not be occupied by any other object. So the question of the optimality of the QOP algorithm arises when entering phase 2.

	Monte Carlo (N=240)			model
λ	w=2	w=3	w=4	w=2
0.1	0.000	0.000	0.000	0.000
0.2	0.000	0.000	0.000	0.000
0.3	0.000	0.000	0.000	0.000
0.4	0.000	0.000	0.000	0.000
0.5	0.001	0.000	0.000	0.000
0.6	0.008	0.000	0.000	0.006
0.7	0.031	0.000	0.000	0.031
0.8	0.068	0.000	0.000	0.069
0.9	0.113	0.004	0.000	0.115
1.0	0.161	0.061	0.024	0.162
1.1	0.208	0.129	0.103	0.209
1.2	0.252	0.192	0.174	0.253
1.3	0.294	0.247	0.235	0.295

Table 3. amf_{qop} obtained with Monte Carlo simulations (left) and with Formula 10 (right)

We denote t_K the time when phase 2 starts. We denote K the set of free objects at time t_K, and U_K the set of urns containing any ball associated to objects in K (on the example of Figure 2, K consists of 5 objects and U_K consists of 4 urns).

We show in this section that the QOP algorithm is optimal in the case $w = 2$. At time t_K, there are $card(K)$ free objects, and $card(U_K)$ possible locations for these objects. It should be noted that $U_K \subset V_{t_K}$. Moreover, $card(F(u)) \geq 2$ for all $u \in U_K$. The maximum number of objects that can be placed during phase 2 is

$$min(card(K), card(U_K))$$

In the case $w = 2$, we have $card(K) \geq card(U_K)$. This can be seen by summing up the number of balls in U_K, and dividing by w because each object is associated to w balls

$$\begin{aligned} card(K) &= \frac{1}{w} \sum_{u \in U_K} card(F(u)) \\ &\geq \frac{2}{w} card(U_K) \\ &= card(U_K) \end{aligned}$$

Consequently, in the case $w = 2$, the maximum number of objects that can be placed during phase 2 is $card(U_K)$. Actually, it is possible to place an object in each of the $card(U_K)$ locations. We show that, at any time t,

$$\begin{cases} card(\{u \in V_t \cap U_K \mid card(F(u)) = 0\}) = 0 \\ card(\{u \in V_t \cap U_K \mid card(F(u)) = 1\}) \leq 1 \end{cases} \quad (5)$$

If this assertion holds for $t \geq t_K$, $V_t \cap U_K$ decreases as long as $V_t \cap U_K \neq \emptyset$, and we are able to fill all the locations associated to urns in U_K. We show assertion 5 inductively. It is obviously true at time t_K. We assume assertion 5 is true at time t, and we show it remains true at time $t+1$ after placing an object. There are two cases :

- For all urns $u \in V_t \cap U_K$, $card(F(u)) \geq 2$. After placing the object, $card(F(u))$ is decremented for all w urns u holding a ball associated to the object. One of these w urns is not in V_{t+1}. The $w-1$ other urns may have $card(F(u)) = 1$. However, as $w = 2$, there is at most one such urn, hence assertion 5 remains true.

- There exists an urn $v \in V_t \cap U_K$ such that $card(F(v)) = 1$. From assertion 5, $card(F(u)) \geq 2$ for $u \in V_t \cap U_K$ different from v. The object from $F(v)$ is placed in the location associated to v. After placing the object, $card(F(v)) = 0$. As $v \notin V_{t+1}$, the first part of the assertion remains true. We may have $card(F(u)) = 1$ for the $w-1$ other urns holding balls associated with the object, but as $w = 2$ there is at most one such urn, hence assertion 5 remains true.

Approximation of amf_{qop} in the case $w = 2$. To derive an approximation of amf_{qop} in the case $w = 2$, we need to derive an approximation of the average value of $card(K)$ and $card(U_K)$ respectively. As previously, we will assume that most configurations are close to the average configuration.

We recall that set K is precisely defined for a given configuration, as the precise set of objects that are placed at the end of phase 1 does not depend on how we select the urns. On the other hand, the precise set of objects that are placed on a given bank may depend on the way urns are selected. Nevertheless, for a given configuration, and for a given bank i, there exists a definite set S_i of objects that cannot be placed on bank i during phase 1, however urns are selected. Objects in K are objects that could not be placed on any bank during phase 1 :

$$K = \bigcap_{i=1}^{w} S_i$$

Let β be the probability that an object belong to S_i. The problem is symmetric (the banks have the same size), so β is the same for all the banks. We have

$$card(K) = n\beta^w \quad (6)$$

It is also possible to derive $card(K)$ by going back to the algorithm. Let us consider the set J of objects defined as

$$J = \bigcap_{i=1}^{w-1} S_i$$

i.e., J is the union of K and the set of objects than can be placed only on bank w during phase 1. The number of objects in J is

$$card(J) = n\beta^{w-1}$$

This is the number of free objects remaining when we place all the objects that can be placed during phase 1 except the objects that can be placed only on bank w. The number of objects that can be placed only on bank w equals the number of urns u on bank w such that $card(F(u)) = 1$. This number can be obtained with Formula 1 for $q = 1$, replacing n with $card(J)$ and N with N/w

$$card(J) - card(K) \simeq n\beta^{w-1}e^{-\lambda w\beta^{w-1}}$$

We get a second expression for $card(K)$:

$$card(K) \simeq n\beta^{w-1}\left(1 - e^{-\lambda w\beta^{w-1}}\right) \quad (7)$$

By equating expressions 6 and 7, we obtain an equation for β

$$\beta + e^{-\lambda w\beta^{w-1}} = 1 \quad (8)$$

One trivial solution to this equation is $\beta = 0$. For $w = 2$, this is the only solution in $[0..1]$ when $\lambda \leq 0.5$ It means that 2-way skewed-associativity with QOP emulates full associativity for $\lambda \leq 0.5$.

When there are several solutions to Equation 8 in $[0..1]$, we will take the one closest to 1. [2] Table 4 shows $card(K)/n$ obtained with Monte Carlo simulations and that obtained with Formula 6 and Equation 8 (there is a threshold effect worth noting for $w > 2$, with β going suddenly from near 0 to near 1).

Now we seek $card(U_K)$. Like K, U_K depends solely on the initial configuration. From the symmetry of the problem, urns in U_K are spread equally on the w banks on average. Let us place all the objects that are not in J, as we did previously. There remains $n\beta^{w-1}$ free objects. Urns on bank w that belong to U_K are the urns u such that $card(F(u)) > 1$. The number of such urns can be obtained with Formula 1 with $q = 0$ and $q = 1$, replacing n with $n\beta^{w-1}$ and N with N/w

$$\frac{card(U_K)}{w} \simeq \frac{N}{w} - \frac{N}{w}e^{-\lambda w\beta^{w-1}} - n\beta^{w-1}e^{-\lambda w\beta^{w-1}}$$

This simplifies to

$$card(U_K) \simeq N\beta - nw(1-\beta)\beta^{w-1} \quad (9)$$

In the case $w = 2$, we have

[2] We were not able to find a justification for this.

	Monte Carlo (N=240)			model		
λ	w=2	w=3	w=4	w=2	w=3	w=4
0.1	0.002	0.000	0.000	0.000	0.000	0.000
0.2	0.004	0.000	0.000	0.000	0.000	0.000
0.3	0.007	0.000	0.000	0.000	0.000	0.000
0.4	0.014	0.000	0.000	0.000	0.000	0.000
0.5	0.035	0.000	0.000	0.000	0.000	0.000
0.6	0.107	0.000	0.000	0.098	0.000	0.000
0.7	0.256	0.002	0.012	0.261	0.000	0.000
0.8	0.408	0.245	0.679	0.412	0.000	0.689
0.9	0.534	0.661	0.843	0.536	0.661	0.841
1.0	0.634	0.785	0.908	0.635	0.783	0.906
1.1	0.712	0.855	0.943	0.712	0.854	0.942
1.2	0.772	0.900	0.964	0.772	0.899	0.963
1.3	0.820	0.929	0.977	0.819	0.928	0.976

Table 4. $card(K)/n$ obtained with Monte Carlo simulations (left) and β^w, taking for β the greatest solution of Equation 8 in $[0..1]$ (right)

$$amf_{qop} = \frac{card(K) - card(U_K)}{n}$$

Using expressions 6 and 9, we obtain

$$amf_{qop}(2,\lambda) \simeq 2\beta - \beta^2 - \frac{\beta}{\lambda} \quad (10)$$

Table 3 shows $amf_{qop}(2,\lambda)$ obtained with Formula 10.

The QOP algorithm is not optimal for $w > 2$. During phase 2 of the QOP algorithm, we scan the set of objects according to a predefined order, and for each object, we try the banks also according to a predefined order, until we are able to place an object. In our Monte Carlo simulations, we experimented two different ways to order the objects and the banks. For $w > 2$, on some configurations, the number of missing objects was different with the two methods, although on average, as expected, there was no significant difference. This shows that the QOP algorithm is not optimal in a general way.

4.4. Iterative Random Placement (IRP)

We have seen that carefully placing the objects may significantly lower the amf. However, the QOP algorithm requires some global knowledge ($card(F(u))$) that is typically difficult to obtain in real microarchitecture situations. Nevertheless, it is possible to approximate QOP with an iterative algorithm. The IRP algorithm works as follows. The objects are numbered from 1 to n, and we scan the set of

	Monte Carlo (N=240)			model		
λ	w=2	w=3	w=4	w=2	w=3	w=4
0.1	0.000	0.000	0.000	0.000	0.000	0.000
0.2	0.000	0.000	0.000	0.000	0.000	0.000
0.3	0.000	0.000	0.000	0.000	0.000	0.000
0.4	0.000	0.000	0.000	0.000	0.000	0.000
0.5	0.001	0.000	0.000	0.000	0.000	0.000
0.6	0.008	0.000	0.000	0.006	0.000	0.000
0.7	0.030	0.000	0.000	0.031	0.000	0.000
0.8	0.068	0.000	0.000	0.069	0.000	0.000
0.9	0.113	0.002	0.000	0.115	0.000	0.000
1.0	0.161	0.060	0.020	0.162	0.061	0.021
1.1	0.207	0.129	0.103	0.209	0.130	0.103
1.2	0.252	0.192	0.174	0.253	0.192	0.174
1.3	0.294	0.247	0.235	0.295	0.248	0.235

Table 5. amf_{irp} obtained with Monte Carlo simulations after 1000 passes (left) and amf_{min} obtained with Formula 11 (right)

objects repeatedly in that order, going back to object 1 after object n. If the object is already placed, we do nothing, and simply go to the next object. Otherwise, if the object is not placed, there are two cases :

- If there exists an empty location among the set of locations, we place the object in one such location. If there are several empty locations, we choose one randomly.
- Otherwise, if there is no empty location, we choose a random bank, we evict the object already stored on that bank, and we place the new object.

The algorithm stops after an arbitrary number of passes. Missing objects are the objects that are out of the cache when the algorithm stops. The IRP algorithm was experimented in [2]. It is typically what happens in a real cache problem.

It should be noted that the number of objects placed cannot decrease. The first time we place an object, the placement may not be optimal. However, after several passes, we will find an optimal placement for the object, e.g., a location that can be occupied by no other object (in that respect, randomly choosing the bank is essential because it permits trying all the banks). Eventually, after several passes, the cache occupancy converges toward an optimal placement. This was referred to as "self data reorganization" in [2].

Table 5 shows the amf_{irp} obtained with Monte Carlo simulations after 1000 passes, which practically provides an optimal placement. Comparison with Table 3 shows that, although not optimal in the case $w > 2$, the QOP algorithm is not far from optimality. Actually, it is possible to approximate the amf of an optimal placement by assuming that

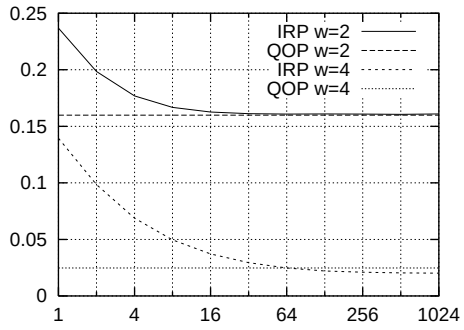

Figure 3. amf_{irp} for $\lambda = 1$ as a function of the number of passes, for $w = 2$ and $w = 4$ (Monte Carlo simulations).

an optimal second phase of the QOP algorithm could place $min(card(K), card(U_K))$ objects. Although not true for all configurations, this is true for a majority of configurations. We obtain the following formula :

$$amf_{min} \doteq max\Big(0, \frac{card(K) - card(U_K)}{n}\Big)$$

and as a function of λ and w

$$amf_{min}(w, \lambda) = max(0, \beta^w + w(1-\beta)\beta^{w-1} - \frac{\beta}{\lambda}) \quad (11)$$

Results obtained with Formula 11 are presented on Table 5, taking for β the greatest solution of Equation 8 in $[0..1]$. This matches the results of Monte Carlo simulations reasonably well. It was shown experimentally in [11] that 2-way skewed-associativity is generally better than 4-way set-associativity and 4-way skewed-associativity is generally better than 16-way set-associativity. These observations are coherent with results on Tables 1 and 5, whatever the working-set size.

Dynamic behavior of IRP. Figure 3 shows the amf_{irp} for $\lambda = 1$ as a function of the number of passes, for $w = 2$ and $w = 4$. We observe that a higher associativity degree requires more passes for IRP to converge toward an optimal placement. However, the case $\lambda = 1$ corresponds to a worst case. Figure 4 shows amf_{qop}, amf_{obp}, and amf_{irp} for $w = 3$ as a function of λ, after 1, 2, 4, and 8 passes. It can be seen that only a few passes are necessary until amf_{irp} comes close to amf_{qop}. We already mentioned the looseness of the QOP algorithm. The rather quick convergence observed on Figure 4 provides yet another indication that there is not one, but many quasi-optimal placements. We believe that this is an important reason for the observed efficiency of skewed-associativity in practical situations.

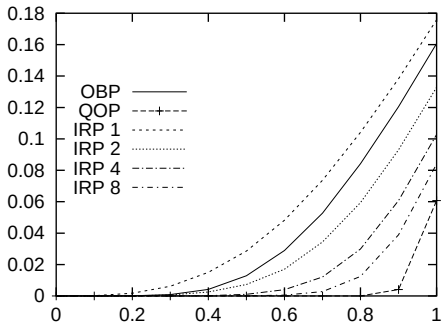

Figure 4. amf_{qop}, amf_{obp} and amf_{irp} for $w = 3$ after 1, 2, 4, and 8 passes (λ on the x-axis).

One-pass placement: OBP is better than random. On Figures 4, it is interesting to note that, after one pass, IRP is not as good as OBP: on the first pass, it is better trying the banks according to a predefined order rather than choosing a random bank. This can be understood intuitively as follows. Let us consider a 2-way skewed associative cache with N locations, and m objects already placed. We seek to place one extra object. Let us assume that, among these m objects, xm have been placed on one bank, and $(1-x)m$ have been placed on the other bank. The probability that the new object will conflict with objects already placed is

$$\frac{xm}{N/2} \times \frac{(1-x)m}{N/2} = x(1-x)\Big(\frac{2m}{N}\Big)^2$$

This probability is maximum for $x = 1/2$, i.e., when both banks hold the same number of objects. A random placement will place statistically the same number of objects on both banks. Random placement is a pessimal strategy for a one-pass placement. The OBP algorithm, on the other hand, will put more objects on one bank than on the other bank. A similar reasoning holds for associativity degrees greater than 2.

5. Related Works

Most previously published cache models focus on set-associativity (e.g., [1, 8]), as it is the most widely used technique. The goal is in general to predict the dynamic miss ratio. The main difficulty in this case consists of modeling application characteristics, like spatial and temporal locality, and their effects. In some of these works [6, 8], the fraction of static conflicts was derived with approaches similar to ours.

A few previous works have studied the performance of skewed-associativity [10, 12, 2, 11, 13]. These are mostly experimental studies, that also treat specific implementation problems, like defining adequate hashing functions, or emulating LRU replacements.

To our knowledge, the work closest to ours is [7]. It derives amf_{obp} for $w = 2$ and $\lambda = 1$. However, the essential question of the optimal placement is not addressed in this work.

6. Conclusion and Open Questions

With a combination of Monte Carlo simulations and analytical modeling, we have shown that the efficiency of skewed-associativity is inherently statistical. Our study confirmed previously published qualitative observations, clarifying them : with a high probability,

- 2-way skewed-associativity removes conflicts better than 4-way set-associativity
- 2-way skewed-associativity emulates full associativity for working-sets up to half the cache size
- 3-way skewed-associativity emulates full associativity for working-sets up to 90% the cache size, i.e., it is almost equivalent to full associativity

Although this study brings a clarification on the efficiency of skewed-associativity, several questions remain. We did not consider the impact of temporal access patterns of objects, in particular the case of some objects in the working-set being accessed more frequently than others (e.g., a working-set included in another one). For obvious reasons, the age-based replacement policy prevails over random placement [11]. This could prevent the convergence of the amf toward its theoretical minimum in some situations.

The IRP algorithm rises another question : although it takes only a few passes to come close to the maximum occupancy, the *placement misses* generated in the meantime may not be negligible under frequent working-set transitions. If some applications exhibit this kind of behavior, it might be worth trying to predict the optimal placement for an object based on past behavior.

Acknowledgment

I would like to thank André Seznec for helping clarify a few points.

Appendix

We are using in this appendix standard methods of combinatorial enumeration. We refer to [9, 4], or any equivalent book, for an introduction to these methods. We are studying here the configurations of n distinguishable balls into N distinguishable urns. We model an urn as a labeled

structure, as described in [4]. The exponential generating function (EGF) for a single urn is

$$u(z) = \sum_{n=0}^{\infty} \frac{z^n}{n!} = e^z$$

The EGF for N urns is

$$U(z) = u(z)^N = e^{zN}$$

To obtain the number of configurations of n balls into N urns, we extract the coefficient of z^n in the Taylor expansion of $U(z)$ and multiply it by $n!$

$$n![z^n]U(z) = N^n$$

which is the expected result. Until now, we have done nothing but decomposing the configurations in a way allowing to extract some information. In particular, we are interested in knowing the number of configurations with a fixed number k of urns containing exactly q balls. To this aim, we "mark" the urn containing q balls in the EGF $u(z)$ using an extra variable x. We obtain the following bivariate generating function (BGF) for a single urn

$$u(z,x) = e^z + (x-1)\frac{z^q}{q!}$$

The BGF for a sequence of N distinguishable urns is

$$U(z,x) = u(z,x)^N$$

The number of configurations with (exactly) k urns containing (exactly) q balls is obtained as

$$c_k = n![z^n x^k]U(z,x)$$

However, instead of computing explicitly the distribution, we can extract the average $\overline{k}$ directly from the ordinary generating function (OGF) of c_k

$$U_n(x) = n![z^n]U(z,x) = \sum_{k=0}^{\infty} c_k x^k$$

The average number $\mu_q = \overline{k}$ of urns with q balls can be obtained as

$$\mu_q = \frac{U_n^{'}(1)}{U_n(1)} = \frac{n![z^n]\frac{z^q}{q!}Ne^{z(N-1)}}{N^n} = N\binom{n}{q}\frac{(N-1)^{n-q}}{N^n}$$

When $N \gg 1$ and $n \gg 1$, the average number μ_q of urns with q balls can be approximated using Stirling's approximation

$$\mu_q \simeq N\frac{(\frac{n}{N})^q}{q!}e^{-\frac{n}{N}}$$

The moment of order 2 can be obtained as

$$\overline{k^2} = \frac{U_n^{''}(1) + U_n^{'}(1)}{U_n(1)}$$

It can be shown that the standard deviation $\sigma = \sqrt{\overline{k^2} - \overline{k}^2}$ becomes negligible compared to μ_q for large N and n. This concentration of the distribution around the mean is called *convergence in probability* [4]. Practically, it means that most configurations are close to the average configuration.

References

[1] A. Agarwal, M. Horowitz, and J. Hennessy. An analytical cache model. *ACM Transactions on Computer Systems*, 7(2):184–215, May 1989.

[2] F. Bodin and A. Seznec. Skewed associativity improves program performance and enhances predictability. *IEEE Transactions on Computers*, 46(5):530–544, May 1997.

[3] W. Feller. *An introduction to probability theory and its applications*, volume 1. Wiley, second edition, 1957.

[4] P. Flajolet and R. Sedgewick. Analytic combinatorics - Symbolic combinatorics. Preliminary version, May 2002. http://algo.inria.fr/flajolet/Publications.

[5] J. Hennessy and D. Patterson. *Computer architecture - A quantitative approach*. Morgan Kaufmann, 1996. Second edition.

[6] R. Kessler and M. Hill. Page placement algorithms for large real-indexed caches. *ACM Transactions on Computer Systems*, 10(4):338–359, Nov. 1992.

[7] S. Majumdar and J. Radhakrishnan. Analytical studies of strategies for utilization of cache memory in computers. arXiv:cond-mat/0001090, Jan. 2000.

[8] M. Schlansker, R. Shaw, and S. Sivaramakrishnan. Randomization and associativity in the design of placement-insensitive caches. HPL-93-41, HP Labs, June 1993.

[9] R. Sedgewick and P. Flajolet. *An introduction to the analysis of algorithms*. Addison Wesley, 1996.

[10] A. Seznec. A case for 2-way skewed associative caches. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, 1993.

[11] A. Seznec. A new case for skewed-associativity. Technical Report PI-1114, IRISA, July 1997. Also published as INRIA report RR-3208.

[12] A. Seznec and F. Bodin. Skewed-associative caches. In *Proceedings of PARLE*, 1993. LNCS 694, Springer-Verlag.

[13] N. Topham and A. González. Randomized cache placement for eliminating conflicts. *IEEE Transactions on Computers*, 48(2):185–191, Feb. 1999.

[14] H. Vandierendonck and K. D. Bosschere. Efficient profile-based evaluation of randomising set index functions for cache memories. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, 2001.