A Statistical Model of Skewed-Associativity

Pierre Michaud

March 2003

It’s about microarchitected “caches”

Type of cache	Type of object
Data/instructions cache	Data/instructions block
Translation buffer	Page translations
Branch target buffer	Branch predictions
…	…

An analysis of skewed-associativity

- Cache implementation for removing conflict misses
  - introduced by André Seznec in the early 1990's
  - experimental evidences of efficacy
- Goal of this study
  - try to understand the reason of the efficacy of skewed-associativity
    - requires understanding set-associativity under randomized hashing

The conflict-miss problem

- The access to objects in the cache should be as fast as possible
  - ==> cache size limit
  - ==> access through hashing function
- Missing objects (= not in cache) ==> performance penalty
  - working-set larger than the cache ==> capacity misses
  - collisions ==> conflict misses

Set-associativity

- Split the cache into *w* banks (*w*-way set-associative)
  - an object has *w* possible locations, one on each bank
- Index all *w* banks simultaneously with the same hashing function
- Trade-off: hardware complexity vs. conflict misses
  - higher associativity *w* ==> less conflict misses
    - if *w* equals number of cache locations ==> full associativity
  - higher associativity w ==> hardware complexity
    - *w* comparators and *w*-input multiplexor
    - access time, energy consumption per access, and cache area increase with degree of associativity *w*

Skewed-associativity

- Like set-associativity but ...
- Different hashing functions

Properties of skewed-associativity

- With a high probability,
  - 2-way skewed-associativity removes conflicts better than 4-way set-associativity under randomized hashing
  - 2-way skewed-associativity emulates full associativity for working-sets up to 50 % the cache size
  - 3-way skewed-associativity emulates full associativity for working-sets up to 90 % the cache size

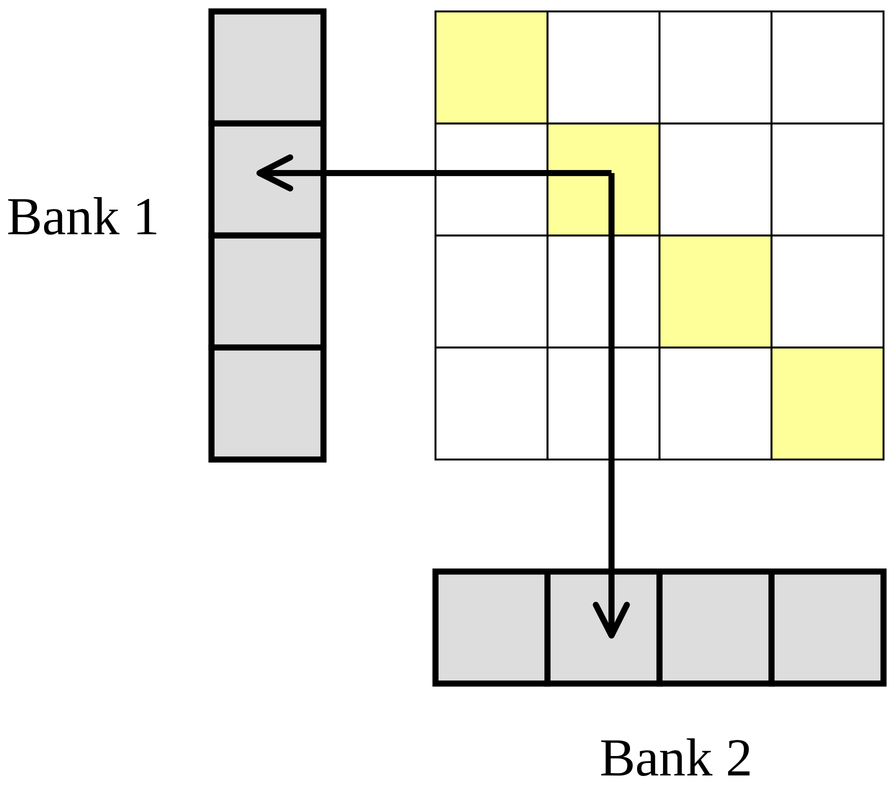
INRIA

Do you find it intuitive ?

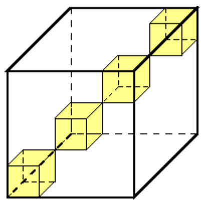
- Usual explanation
  - if several objects conflict for the same location on one bank, they are unlikely to conflict on the other banks …
- Objection: we should think globally
  - if the working-set size is close to the cache size, we should not expect to find a lot of free locations on the other banks
- Intuition fails in this kind of problem
  - optimal placement ?
  - not always better than set-associativity, statistically better

2-way set-associativity

Cache size: N = 8 locations

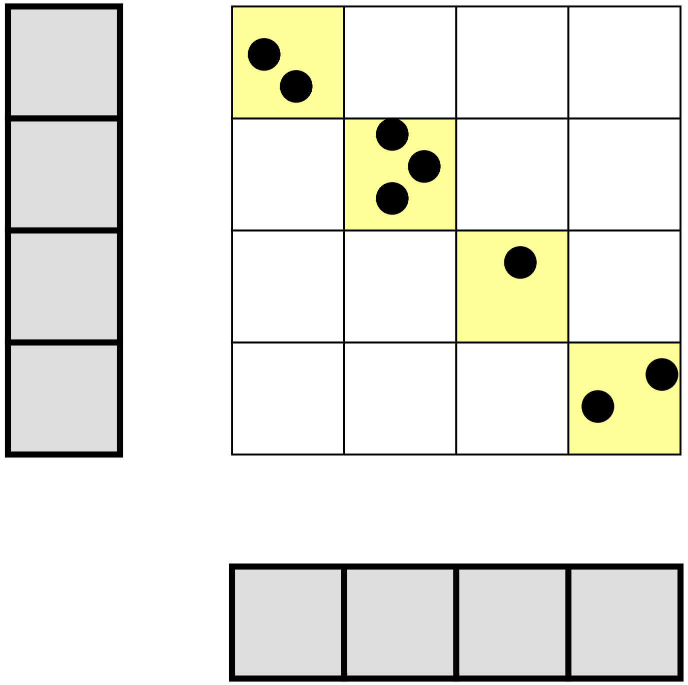


3-way associativity



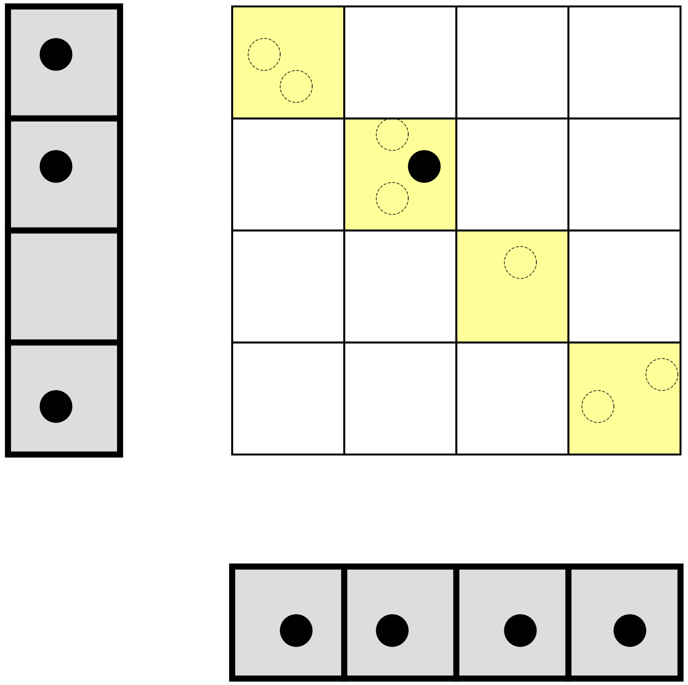
2-way set-associativity

Take $n = 8$ random objects



2-way set-associativity

Place the objects

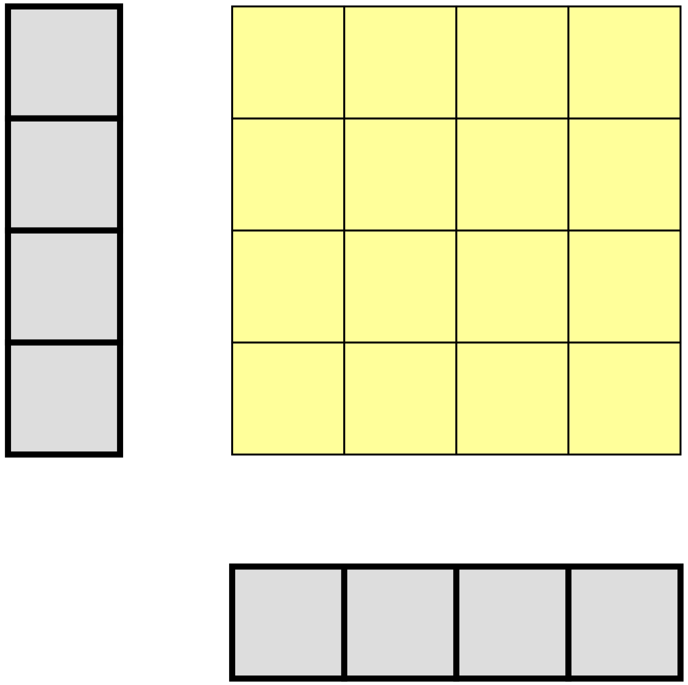


7 objects placed

1 missing object

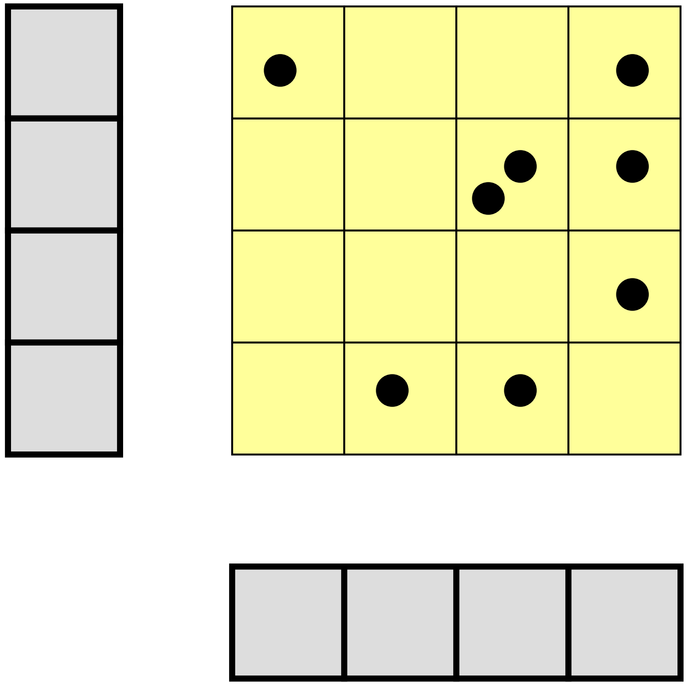
2-way skewed-associativity

"Orthogonal" hashing functions



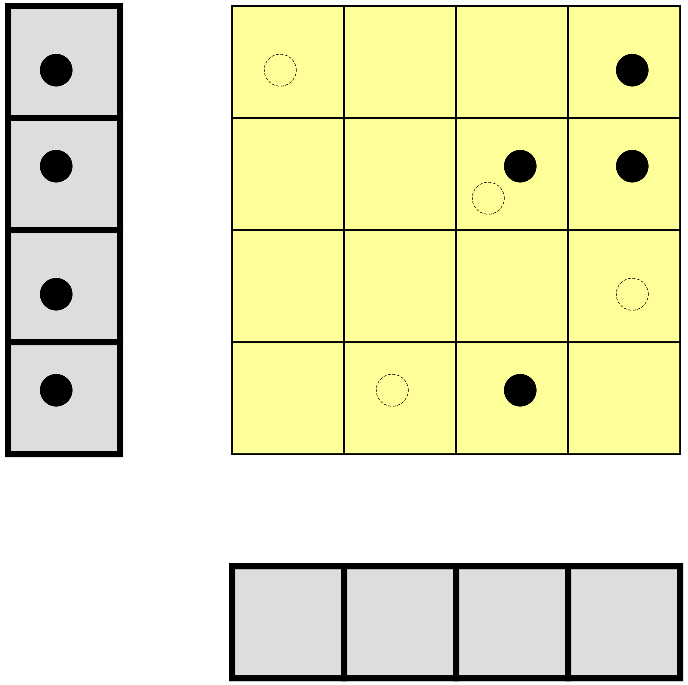
2-way skewed-associativity

Take $n = 8$ random objects



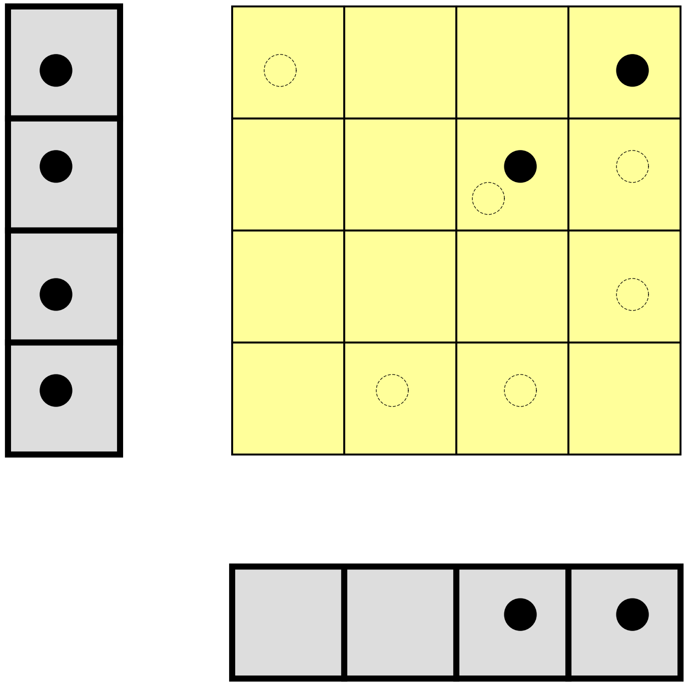
2-way skewed-associativity

Place objects on one bank



2-way skewed-associativity

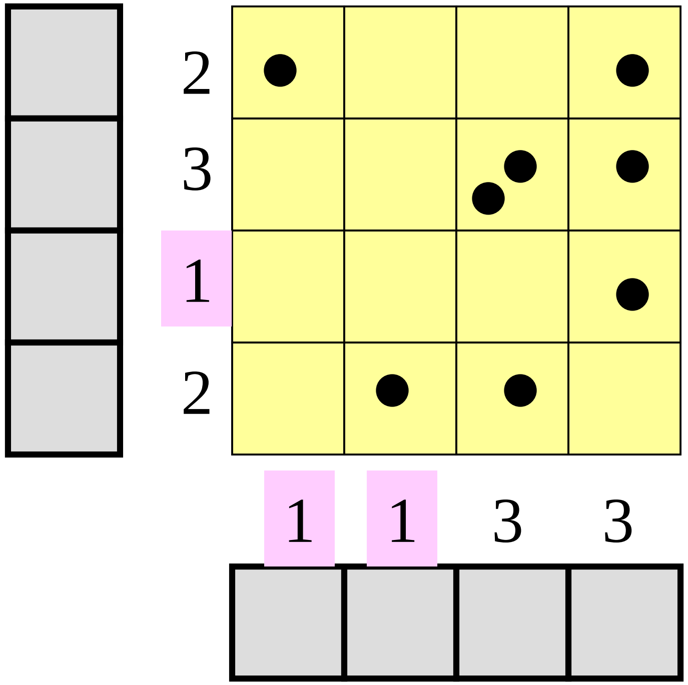
Place remaining objects on the other bank

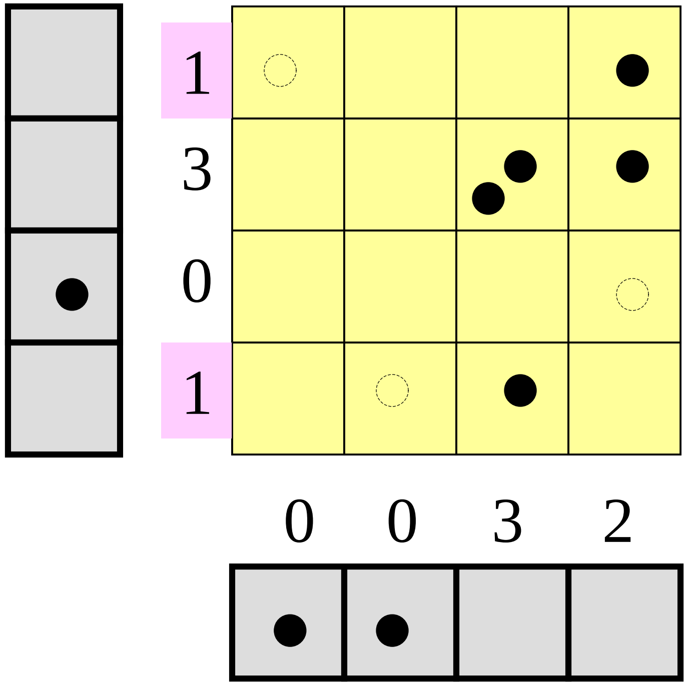


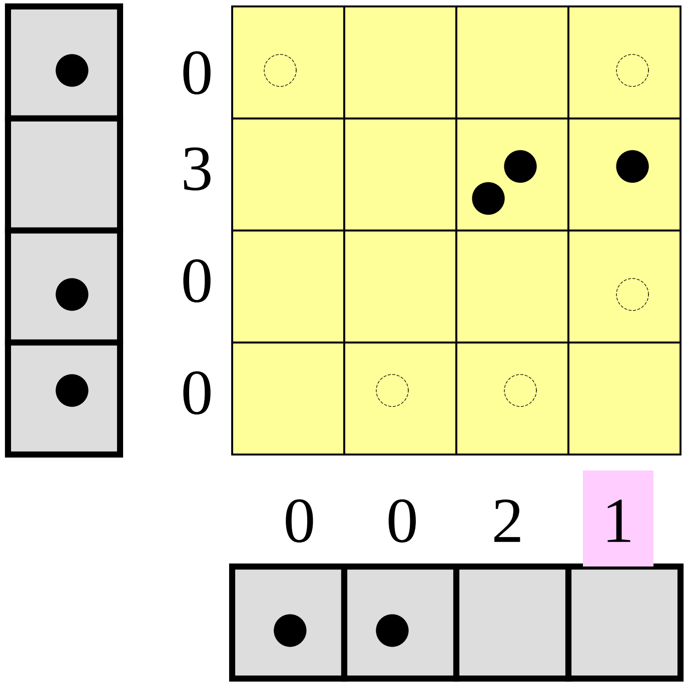
6 objects placed

2 missing objects

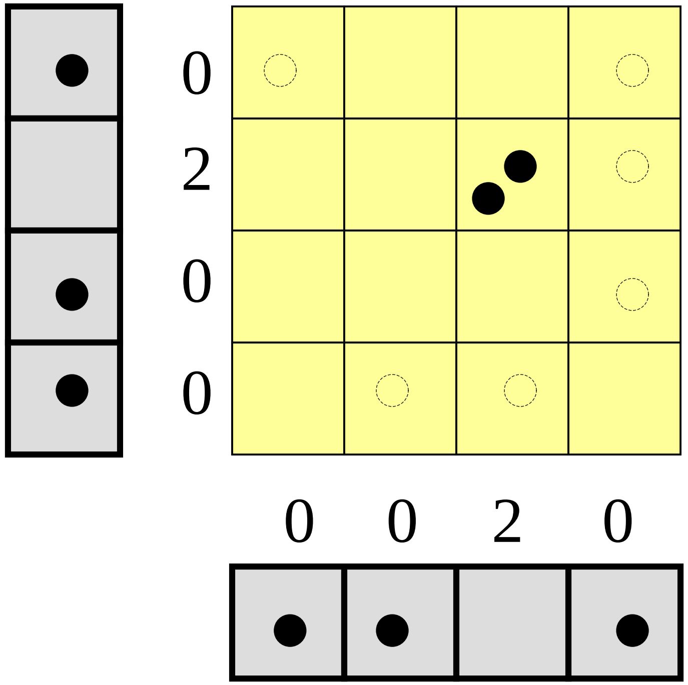
There exists a better placement




1
3
0
1
0 0 3 2


0
3
0
0
0 0 2 1

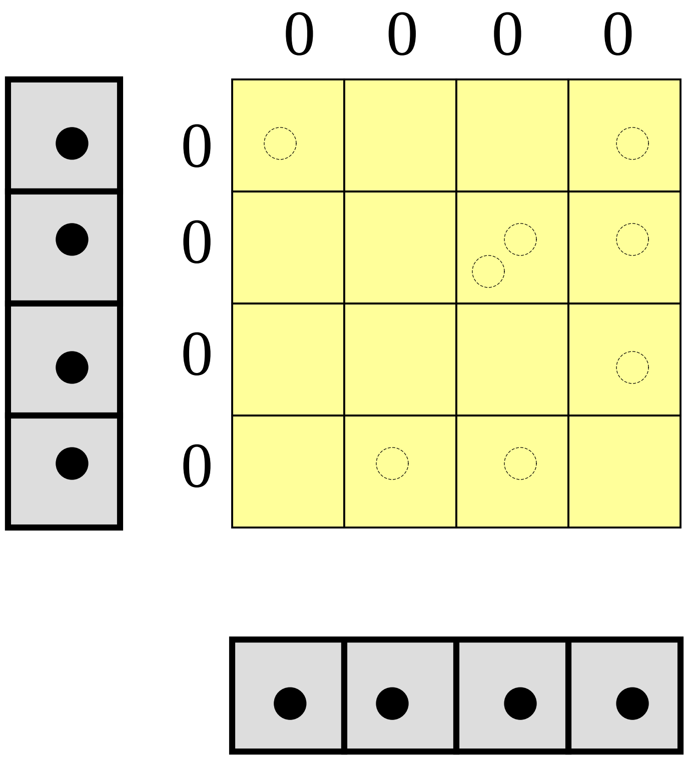
Phase 1 of the algorithm is finished, now phase 2 starts



To continue, make an arbitrary placement

This was the QOP algorithm

Quasi-Optimal Placement



Optimal for $w = 2$

Close to optimal for $w > 2$

Iterative placement

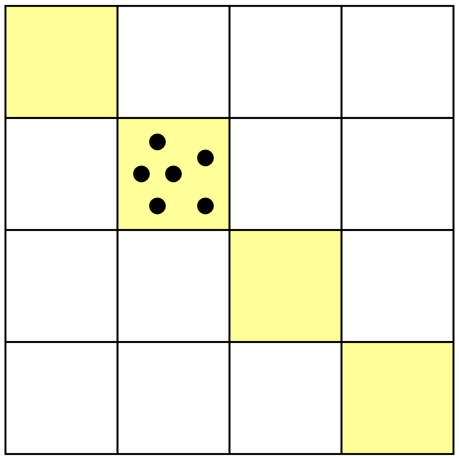
- QOP useful for analysis, not a practical algorithm
  - in a real microarchitecture situation, better to place objects as soon as encountered, even if placement not optimal
- Iterative Placement
  - place object in an empty location
    - in practice, “empty” means “cold”
  - if all locations occupied, evict object already placed
  - several passes ==> converges toward an optimal placement
    - “self data reorganization”
- How many missing objects with an optimal placement ?

Hint: the worst case

2-way set-associativity

N

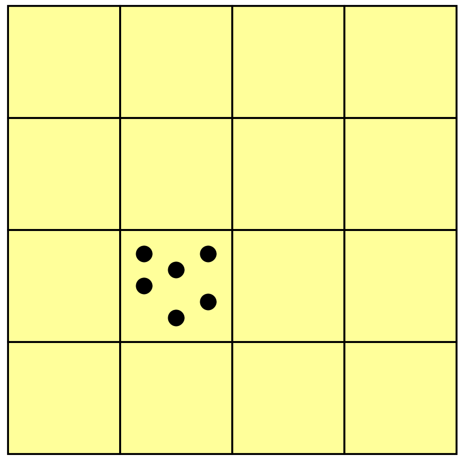
N



2N locations

n objects

2-way skewed-associativity



Total configurations N^{n}

Probability worst case $N^{-(n-1)}$

Total configurations N^{2n}

Probability worst case $N^{-2(n-1)}$

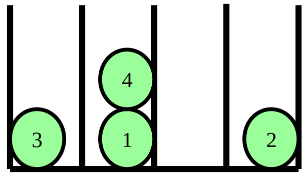
The probability is squared

The average case

- Consider all the possible configurations
  – assuming fixed cache size and working-set size
- Compute the *average missing fraction* (*amf*)
  – average number of missing objects divided by total number of objects
  – *amf* in [0..1]
- The *amf* gives information about the typical configuration
  – *amf* very small ==> few missing objects for most configurations
  – what is likely to be observed with randomized hashing or without spatial locality

The classical occupancy problem

n balls into N bins: N^n configurations

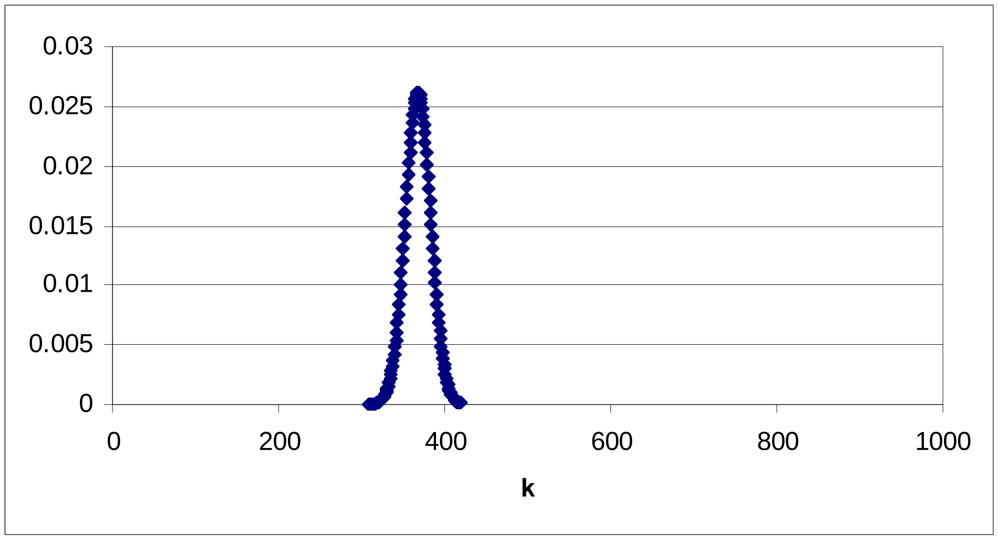


How many configurations with (exactly) k bins containing (exactly) q balls ?

$$c_q(k) = \binom{N}{k} \sum_{j=k}^{N-1} (-1)^{j-k} \binom{N-k}{N-j} \binom{n}{jq} \frac{(jq)!}{(q!)^j} (N-j)^{n-jq}$$

Example: $n = N = 1000$, q=1

$$\frac{c_1(k)}{N^n}$$



Distribution concentrated around the mean

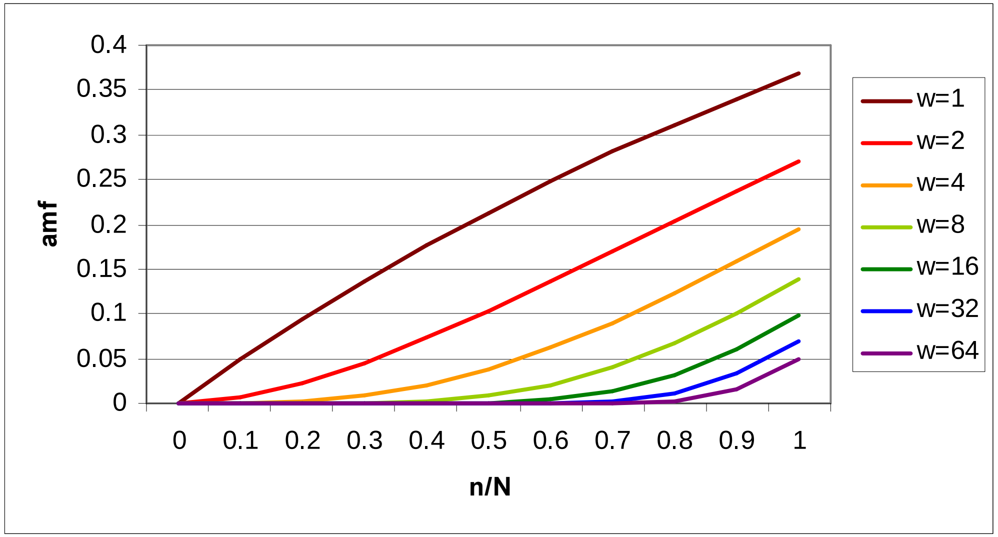
Average: Poisson law

$$\boxed{\bar{k} \approx N \frac{\left(\frac{n}{N}\right)^q}{q!} e^{-\frac{n}{N}}}$$

Set-associativity: average case

n objects
N locations

$w << N$



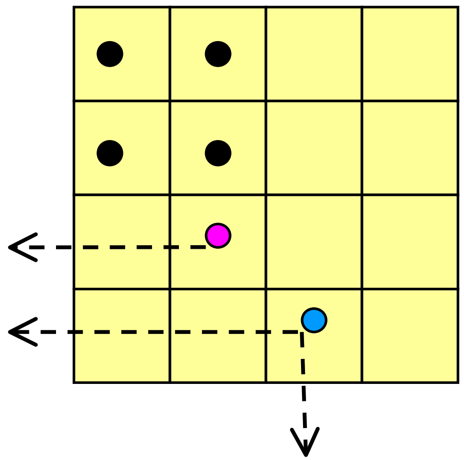
- $n/N < 1/4$: 4-way set-associativity sufficient
- $n/N > 1/2$: set-associativity rather inefficient
- Spatial locality ?
  - observed behavior often better than statistical average
  - sometimes much worse

Skewed-associativity: QOP algorithm

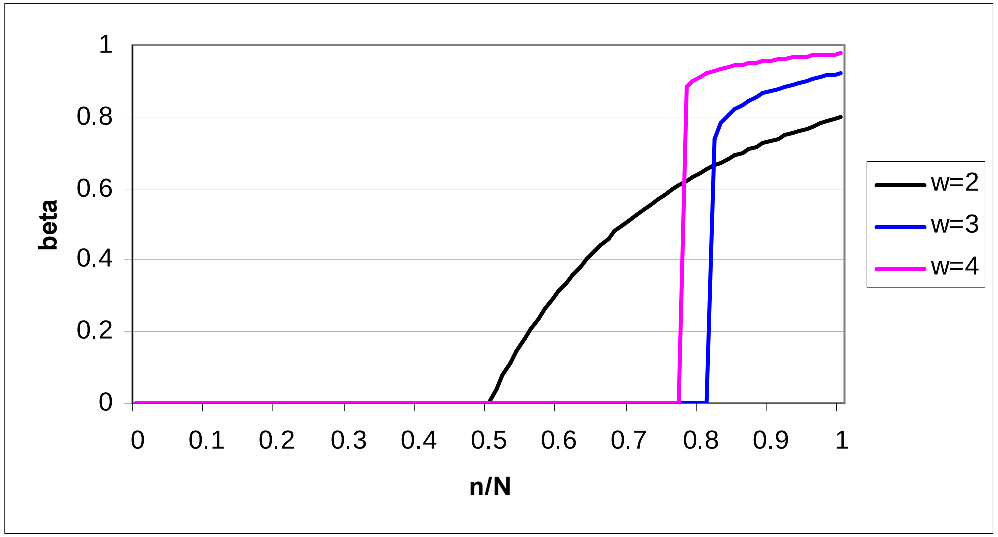
General idea: count bins containing a single ball

Intricate problem ==> heuristic reasoning

- probability β that an object cannot be placed on a given bank during phase 1
  - β=0 means *all the objects can be placed during phase 1*
  - β=1 means *start with an arbitrary placement*

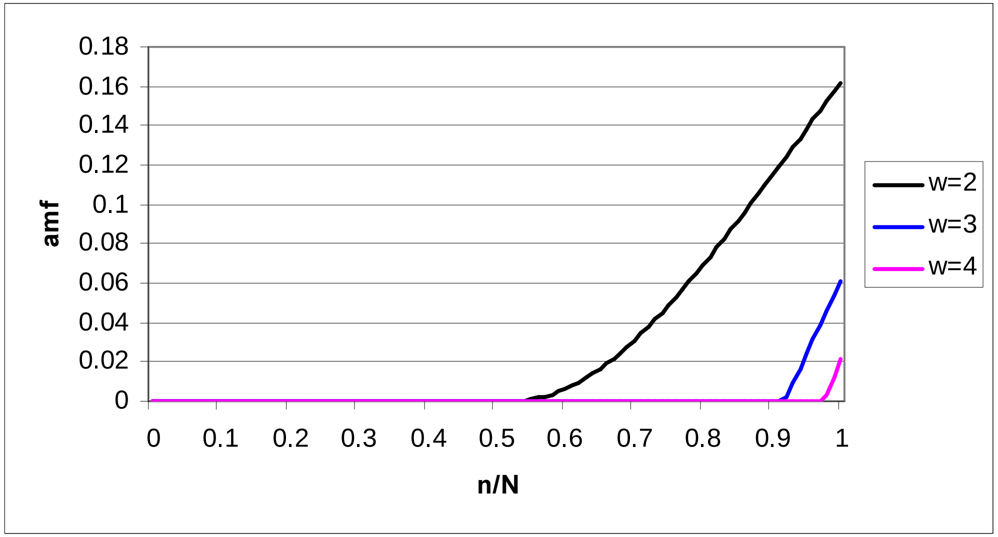


$$\boxed{\beta + e^{-\frac{n}{N} w \beta^{w-1}} = 1}$$



Average missing fraction

$$amf \approx \max(0, \beta^{w} + w(1-\beta)\beta^{w-1} - \frac{\beta}{n/N})$$

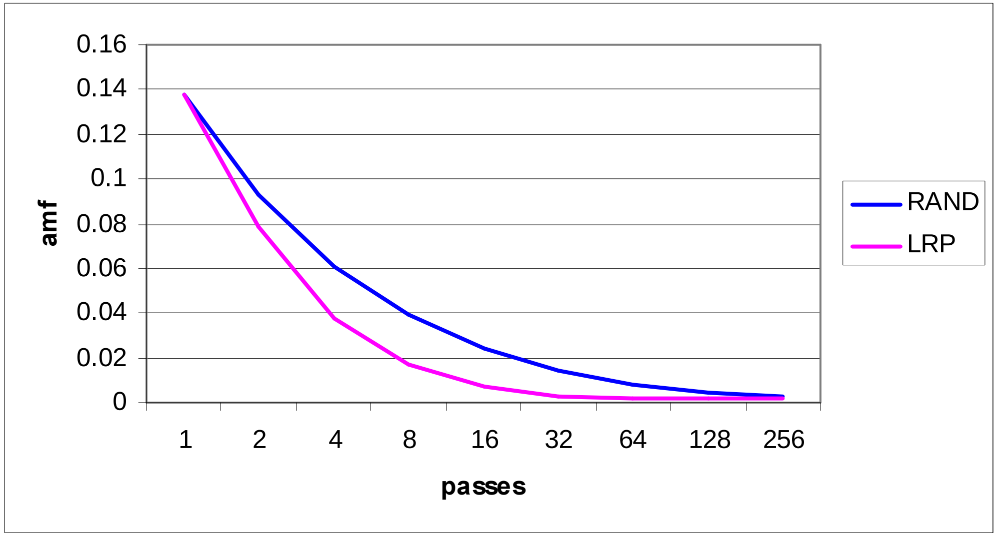


What is observed for a typical configuration

Iterative Placement

- Number the objects from 1 to n
- Iterate on the objects: $1,2,\ldots,n$, $1,2,\ldots,n$, $1,2,\ldots n$, ...
- If object no yet placed, place it in a (random) empty location
- If no empty location, choose a victim
  - RAND: random victim
  - LRP: least recently placed

$$w = 3, \quad \frac{n}{N} = 0.9$$

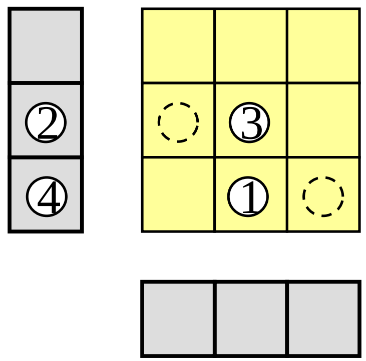


Learnings

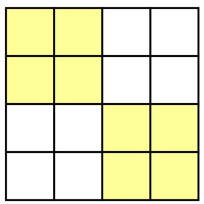
- The efficacy of skewed-associativity is intrinsically statistical
  - spatial locality not necessary
    - just make sure that we don't make spatial locality the worst cases
- 2-way skewed-associativity emulates full associativity for working-sets up to 50% the cache size
- 3-way skewed-associativity is almost equivalent to full associativity
  - iterative placement: ~10 passes are enough
  - little gain to expect with associativity greater than 3
    - greater associativity just requires less passes

Open questions

- Frequent working-set transition ?
  - *placement* misses
- LRU may prevent convergence toward optimal placement
  - but hard to beat on real workloads ...



- Implementation tradeoffs



Conclusion

- Skewed-associativity works
  - more than just the effect of randomized hashing
  - 3-way skewed-associativity almost equivalent to full-associativity with degraded LRU

- Model useful for debugging hashing functions
  - sets of random addresses
  - if measured *amf* $\neq$ theory ==> problem