Using huge pages and performance counters to determine the LLC architecture

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Josué Feliu
Julio Sahuquillo
Salvador Petit
José Duato

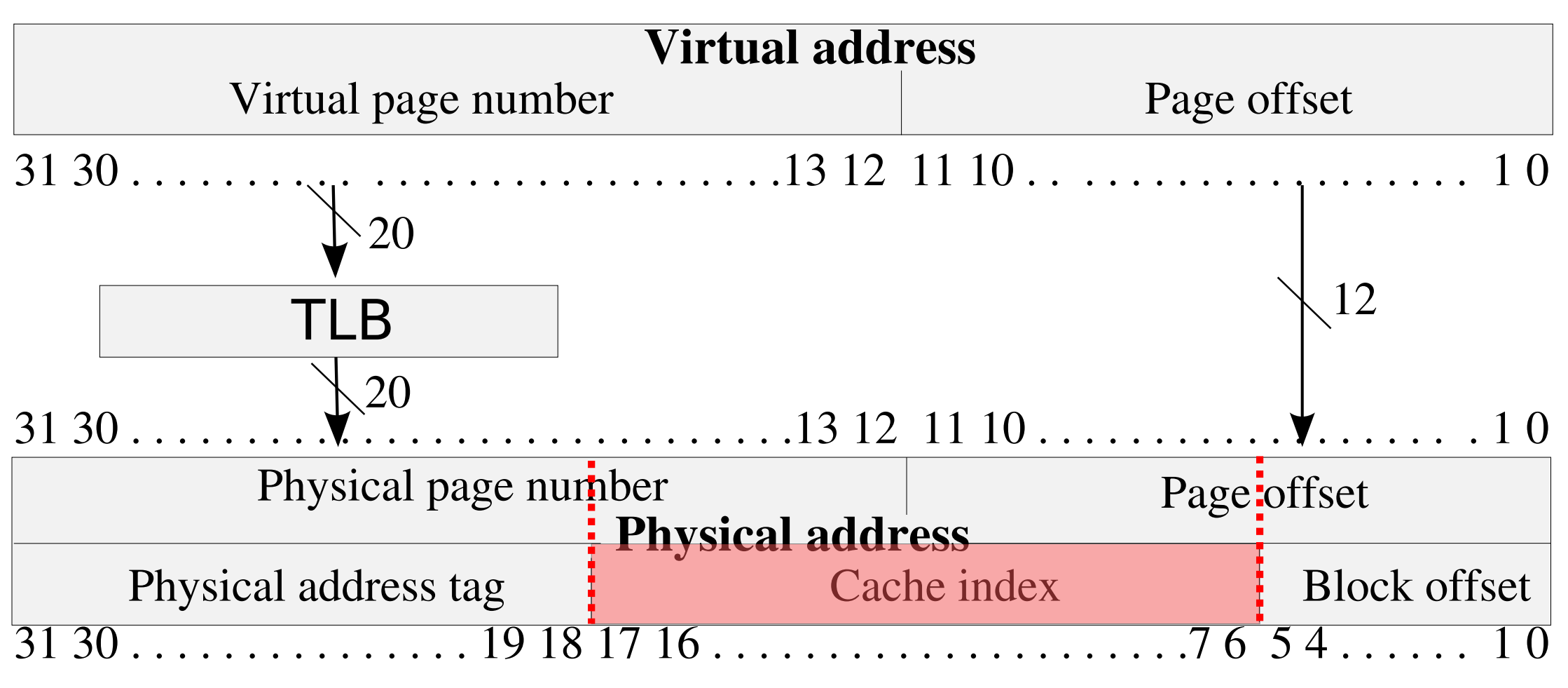
1. Introduction and motivation

Many recent research work has focused on evaluating the performance when running multiprogrammed workloads on a chip multiprocessor (CMP) system. Some authors have focused on a better management of bandwidth utilization and on reducing the interference in the last level cache (LLC) among the processes that are running concurrently. For a correct performance analysis concerning LLC caches, researchers require precise information about the LLC characteristics, mainly the cache geometry (cache size, associativity and line size) and the hierarchy organization (number of LLCs and cores sharing each LLC).
Some prior research works have also focused on determining cache parameters relevant for the system performance. Unfortunately, most of those works:
- **assume virtual indexed caches**, while current LLCs are physically indexed
- **use time-based metrics, which often require complex analysis** of the results to deduce the LLC characteristics
- **do not report complete information** of the cache characteristics

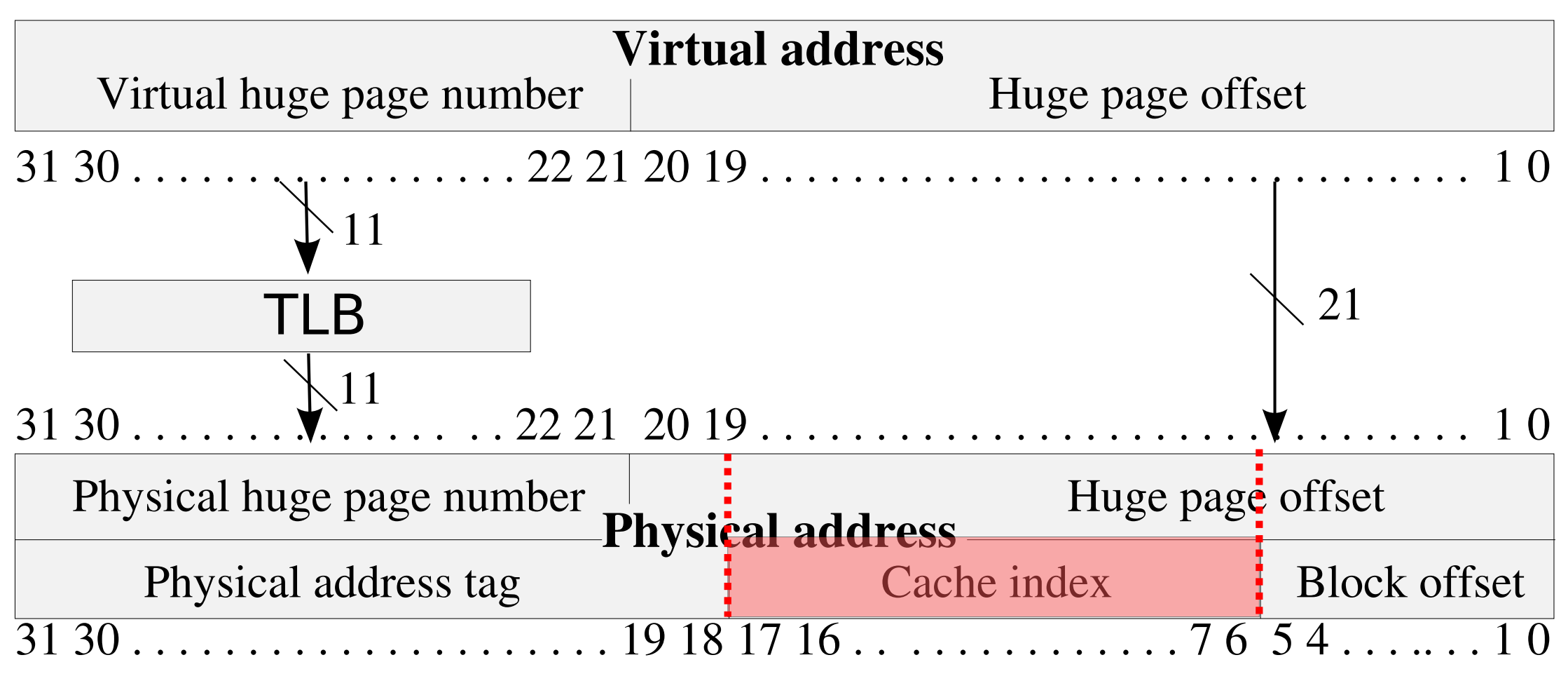
To deal with caches physically indexed and avoid complex analysis of time-based metrics, this work relies on *huge pages* and performance counters, respectively.

2. Virtual memory and address translation for standard pages and huge pages

Figure 1 presents the address translation of virtual addresses into physical addresses used to access the caches. The virtual addresses are split in two fields: virtual page number and virtual page offset. The TLB only translates the virtual page number into the physical page number so that the bits of the page offset are not affected. The physical address is broken down into physical address tag, cache index and block offset to access the cache memory.



(a) Using 4 KB pages



(b) Using 2 MB huge pages

Fig. 1: Address translation for a 4MB 16-way set-associative LLC

For standard 4 KB pages, the page offset is formed with the 12 less significant bits of the virutal address and thus, assuming a 2^6-bytes line size, the cache index field is formed with bits from the physical page number. Since it cannot be determined by user-level processes, the accessed cache sets cannot be controlled.

Using 2 MB huge pages, the huge page offset is formed with the 21 less significant bits of the virtual address and thus, all the bits that form the cache index field are included in the huge page offset. Therefore a user-level process can precisely control the cache sets it accesses.

3. Experimental tests

3.1 Determining the LLC line size

To determine the cache line size Microbenchmark 1 is used to evaluate the LLC hit ratio when accessing sequentially an array with row lengths ranging from 16 to 256 bytes. Since the array at least doubles the cache size, a given block is always evicted between consecutive accesses and hence, if the array row length is
- smaller than the cache line size, the LLC hit ratio is greater than zero (different array rows are included in a single cache line)
- equal or larger than the cache line size the LLC hit ratio approaches zero

Figure 2 presents the results of this experiment in a Xeon X3320 where it determines that the cache line size is 64 bytes. Similar results are obtained with the Core i5 750.

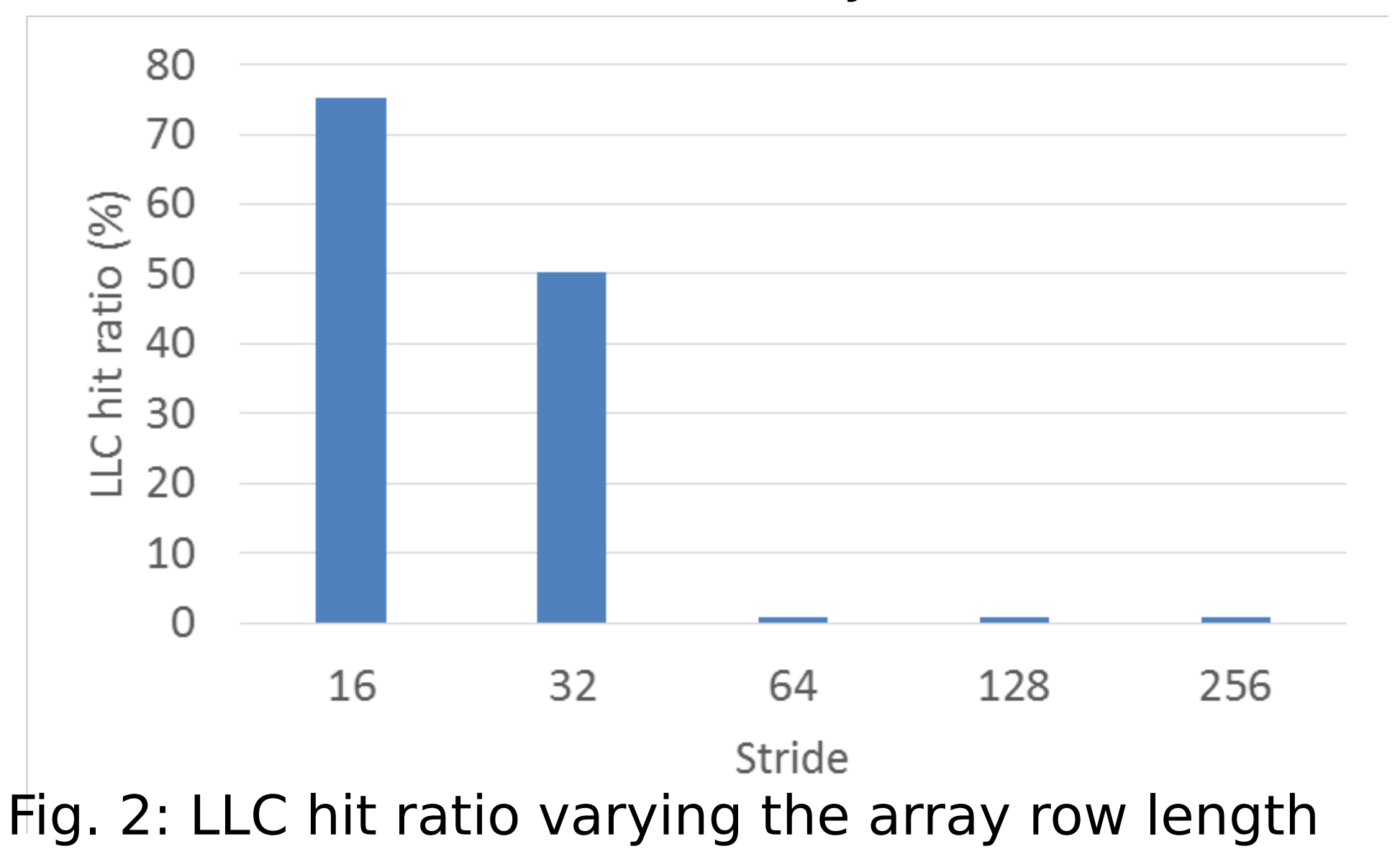


Fig. 2: LLC hit ratio varying the array row length

Microbenchmarks

```
Input: ARRAY_ROW_LENGTH
  $N = 2 * \frac{CACHE\_SIZE}{ARRAY\_ROW\_LENGTH}$
  char A[N][ARRAY_ROW_LENGTH]
  for (r=0; r<REPS; r++) do
    for (i=0; i<N; i++) do
      A[i][0] = 1
```

Microbenchmark 1

```
Input: STRIDE and ARRAY_ACCESSES,
  $N = ARRAY\_ACCESSES * STRIDE$
  char A[N][CACHE_LINE_SIZE]
  for (r=0; r<REPS; r++) do
    for (i=0; i<ARRAY_ACCESSES; i++) do
      A[i*$STRIDE$][0] = 1
```

Microbenchmark 2

3.3 Determining the number LLCs and which cores share each LLC

Microbenchmark 2 fills a target set with blocks, which are continuously accessed without causing misses. To determine if there are shared caches, two instances of it are executed concurrently in different cores. If the cores share the cache, the capacity of the target set is surpassed so resulting in conflict misses.

Figure 4 presents results for these experiments. Results for the Xeon X3320 show that it has two LLCs, one shared by cores 0 and 2, and the other shared by cores 1 and 3. Results for the Core i5 750 show that it has one single LLC shared by all the cores.

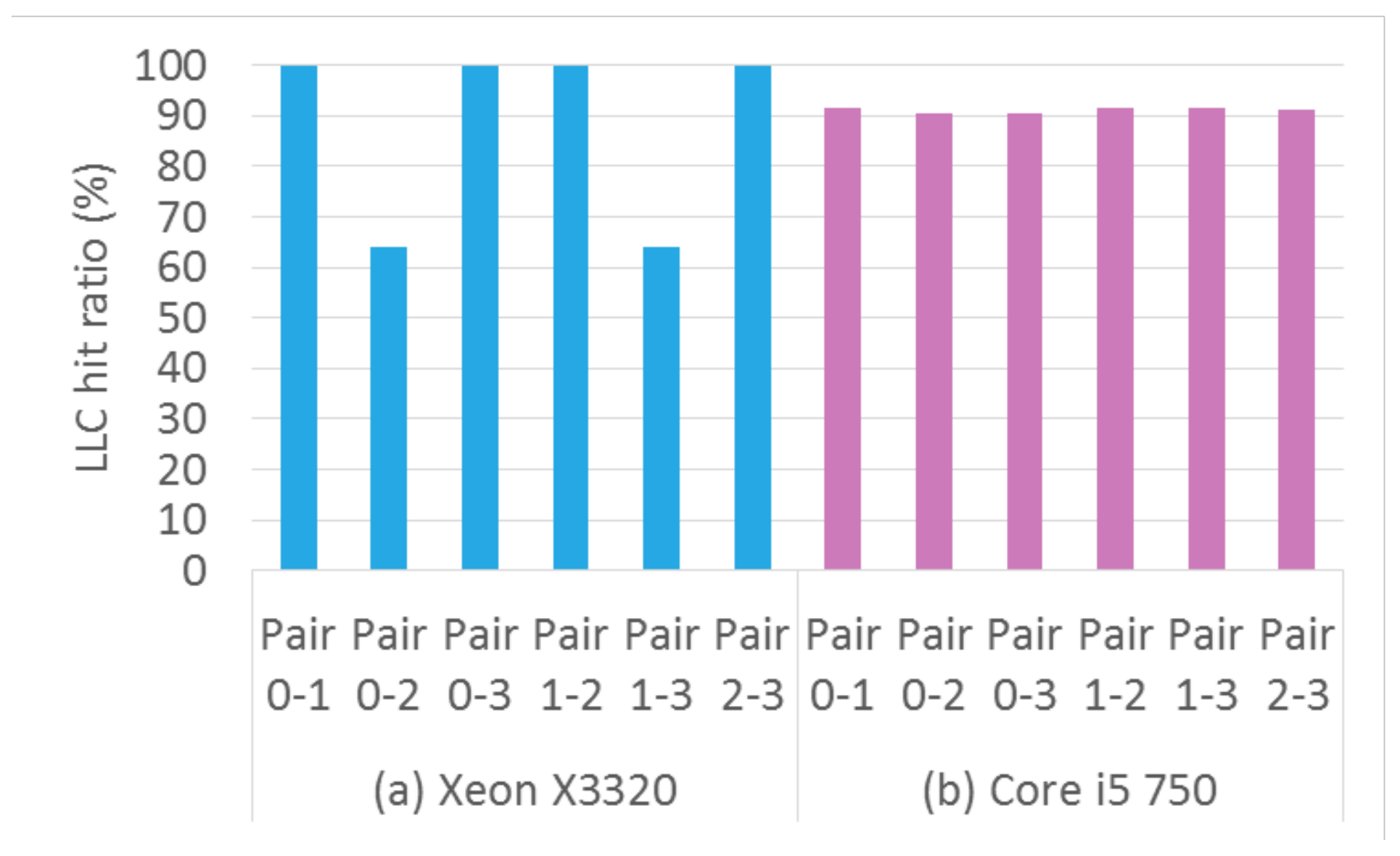


Fig. 4: Average LLC hit ratio when running two microbenchmarks in different core pairs

3.2 Determining the number of sets and number of ways of the LLC

This test has been designed to check the LLC hit ratio while varying the number of different lines that are accessed in a set. Microbenchmark 2 is executed with two input parameters: i) the stride to access the array, which is used to determine the number of sets; and ii) the number of different array accesses, which is used to determine the cache associativity.
The expected result is that the hit ratio starts decreasing when the number of different accessed lines in a cache set is higher than the number of cache ways. In this manner, the number of ways is identified. In addition, the number of sets corresponds to the lowest stride that causes the hit ratio to start decreasing.
The test is executed for tentative values of both input parameters. Results for the Xeon X3320 platform determinte:

LLC associativity: 12

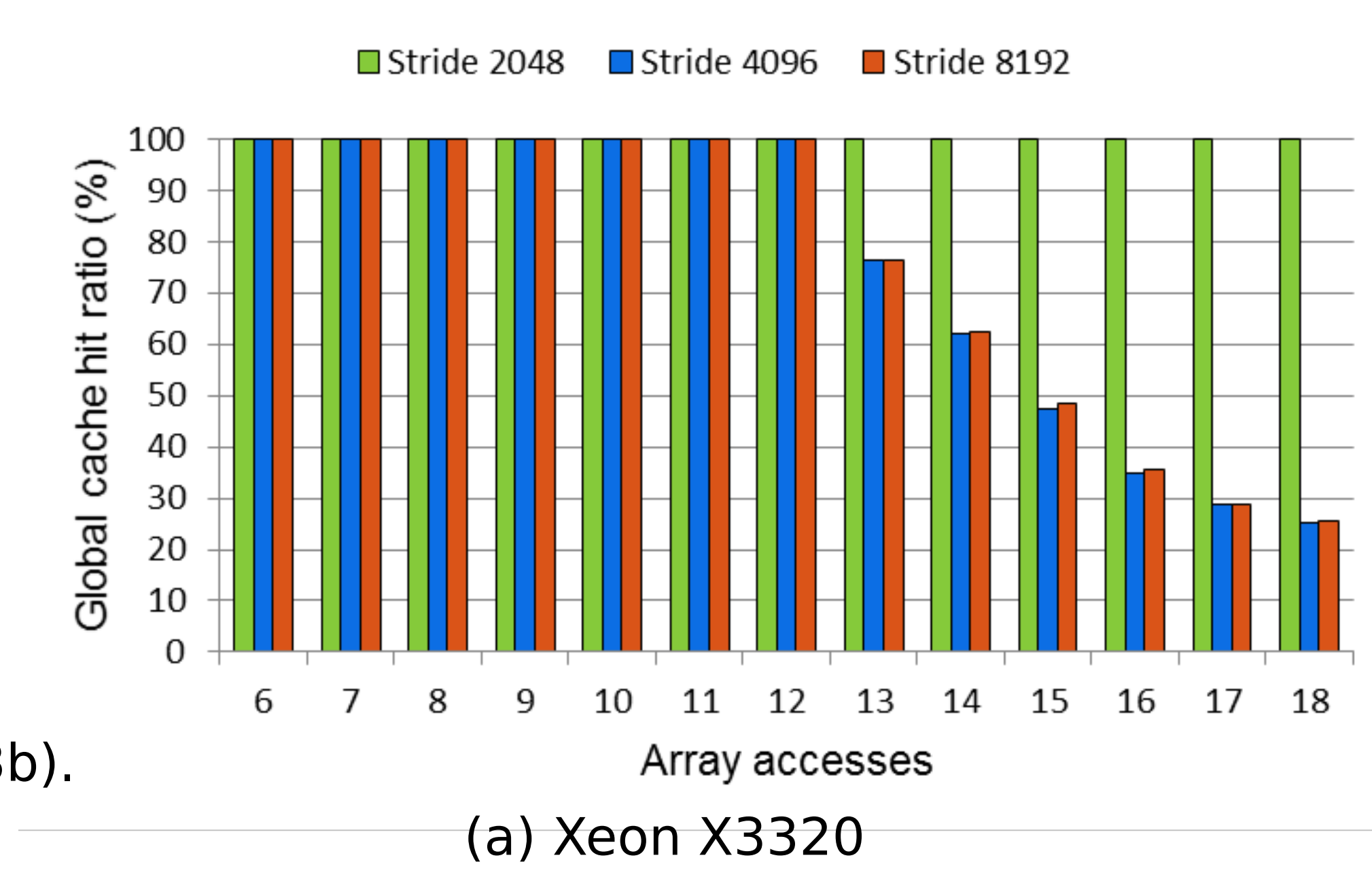
If the number of ways were higher, the LLC hit ratio would not drop with 13 different array accesses.
If it were lower, the LLC hit ratio would decrease with a lower number of different array accesses.

Number of sets: 4096

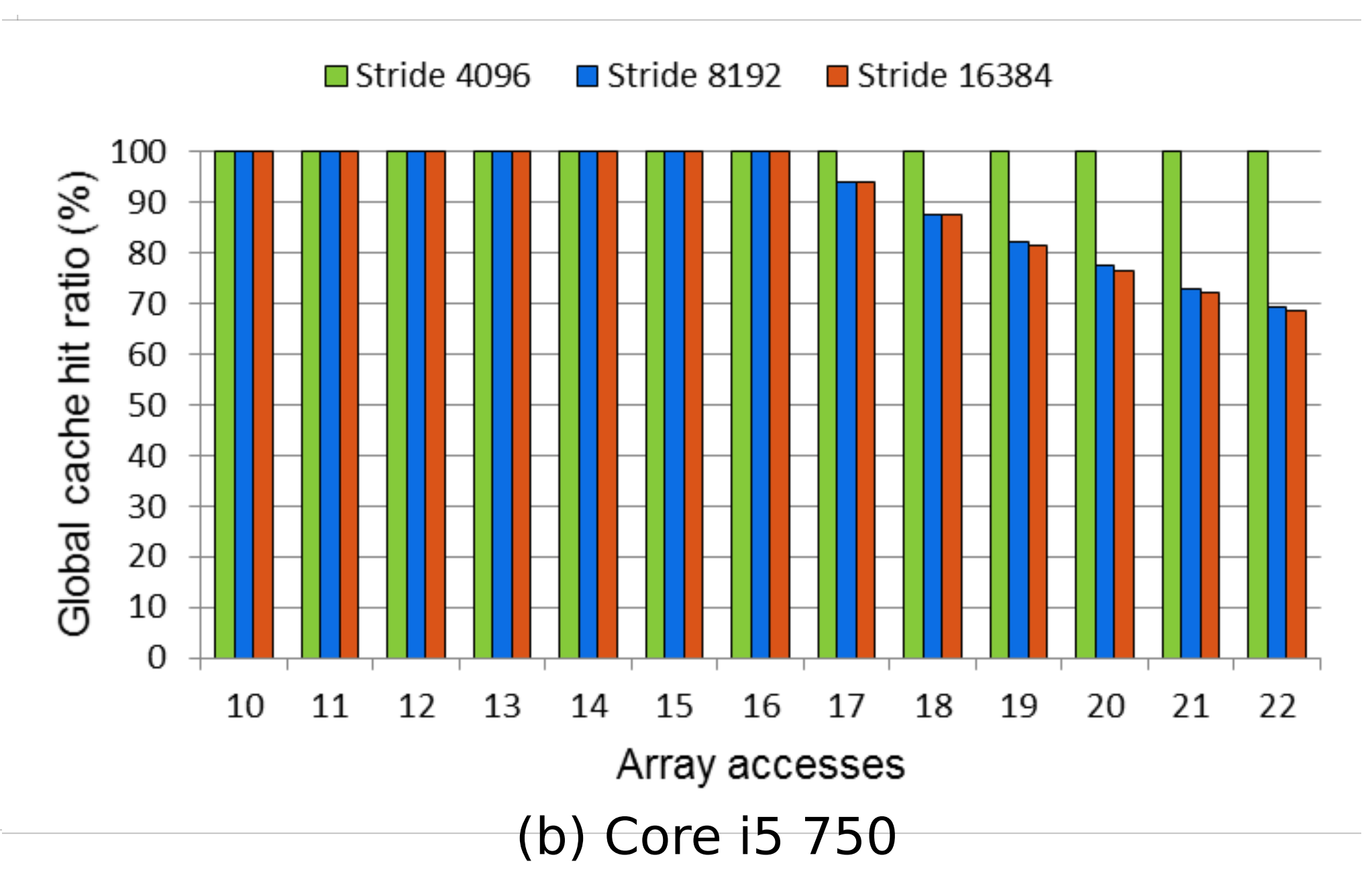
If the number of sets were lower, the LLC hit ratio would decrease with a stride of 2048.

If it were higher, the LLC hit ratio would not decrease with a stride of 4096 since the array accesses would be mapping to different cache sets.

Similar reasoning can be performed to the Core i5 750 platform (Figure 3b).
Associativity: 16, number of sets: 8192



(a) Xeon X3320



(b) Core i5 750

Fig. 3: LLC hit ratio varying the number of different array accesses and stride

International Conference on Computational Science
ICCS 2013

GAP
Parallel Architectures Group