Last Level Cache Size Flexible Heterogeneity in Embedded Systems

Mario D. Marino*, Kuan-Ching Li† *Leeds Beckett University, m.d.marino@leedsbeckett.ac.uk
†Corresponding Author, Providence University, kuancli@gm.pu.edu.tw

Abstract

In typical multicore processors, Last Level Caches (LLC) are formed by distributed clusters of memory banks of the same size, namely homogeneous ones. By shutting down part of these clusters to save power along generations of multicore processors, clusters with non homogeneous cache sizes can be originated, named as heterogeneous ones. Given that heterogeneous clusters have typically smaller sizes than homogeneous ones, they present larger miss rates that are likely to deteriorate performance.

In this investigation, we study the impact of heterogeneous caches in embedded microprocessors, by having an arbitrary mix of homogeneous and heterogeneous clusters. That is, we propose to evaluate the architectural implications of these heterogeneous caches and a flexible algorithm that can be used to explore them. From scientific applications' experimental benchmarking, our findings show that microprocessors with heterogeneous clusters present a maximal performance degradation of about 10% and maximal performance improvement of 16%, while obtaining maximum miss hit rate of reduction and improvement up to 10%. In addition, 10% of coherence activity decrease when presenting maximum energy utilization up to 50% and maximum energy reduction of 15%.

Index Terms

cache; heterogeneity; performance; energy; embedded.

I. Introduction

In the multicore era, the growth on the number of cores increases the bandwidth demands in memory systems. The higher number of simultaneous memory requests due to the higher number of cores significantly increases the levels of memory contention. Larger core availability encourages programs' intensification of cache utilization, which further increases the bandwidth demands of the Last Level Caches (LLC).

By improving cache locality, several solutions have concentrated on sharing parts of their caches among cores [1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17]. In order to alleviate cache contention, a number of solutions are designed with distributed clusters of caches - clusters are defined as sets of cache banks or multibanks shared among processors that spread memory address accesses over these caches via embedded ring [18] or a 2D-mesh network [19], to improve memory bandwidth and cache locality, thus resulting lower latency and higher hit rates.

LLC clusters in typical PC microprocessors are represented by L3 level caches whilst L2 ones in embedded systems. These LLCs can be better exploited by multiple threads running in multiple cores. That is, data brought by one of cores can be reused by other core(s) avoiding off chip misses that improve both locality and hit rates. The idea of having different clusters allows the search for a line to be directed to other banks instead of going to memory. As result, cache locality is maximized. Typical LLC clusters use homogeneous-size caches such as those presented in [1] [20] [21] [22]. Given that clusters are not fully explored by applications, part of caches can be shut down to save power [2][15][16][17]. Taking into consideration that the shut down happens on part of LLC clusters, the resultant ones present reduced sizes when compared to the original ones, and thus, the resulting configuration presents heterogeneous-size caches.

Another interesting opportunity to tackle microprocessors with heterogeneous caches is due to fabrication process variability. Given that core families have been scaling to different cache sizes, fabrication process variability has been an important factor, and thus, future projects are likely to increase the heterogeneity aspect in their designs. Furthermore, even do not presenting the desired sizes, rather than typically discarding them, they can be explored even presenting such heterogeneous cache configurations.

Heterogeneous caches presents several interesting opportunities to be explored. In this paper, we propose to leverage the area of cache systems by exploring the opportunities offered by heterogeneous caches. The contributions of this paper include:

- A design-space exploration of heterogeneous cache sizes, considering series of configurations candidates when morphing cache sizes from a homogeneous configuration to a heterogeneous one. Additionally, analysis of chips with heterogeneous cache sizes reconfigured are done, considering different trade-offs involving performance, hit/miss rates, coherency, and energy.

- The techniques here utilized involve the design, detailed modeling, evaluation, and analysis of an embedded multicore model employing homogeneous and heterogeneous caches sizes. Analysis here proposed can assist the designer to select the best cluster size candidates to be elected for a shutdown operation, or indicate the behavior the processor with cache fabrication variability.
- Many of aspects can be used to select a heterogeneous configuration. For example, energy [2][15][16], performance [2][15][16][17], energy over performance, combination of the previous using a bio-inspired algorithm[10]. Previous works published were mostly concerned to the fundamental aspects of performance and power, which can hide the exposition of computer architectural design aspects' behavior to the designer. To approach the latter aspect, we propose a flexible hardware-based algorithm that can be utilized together with other important architectural aspects (e.g. cache hit rates, read misses, write backs, etc) that may potentially be used by the design architect to switch from homogeneous to heterogeneous configurations or among heterogeneous ones.
- The behavior of the algorithm follows the behavior of a designer when it comes to the criteria of selection of the target configurations. To perform the design behavior, proposed hardware algorithm employs a dedicated hardware (formed by performance counters and sets of registers or content address memory (CAM)) to store performance and time when sampling homogeneous and heterogeneous configurations. Furthermore, as a designer, the algorithm estimates the energy utilized using pre-calculated CACTI [23] energy estimations stored appropriately. By comparing the behavior of the homogeneous and heterogeneous ones, using the configurations which have smaller area sizes, the algorithm is able to select the target heterogeneous configuration and then performs the configuration switch.
- To the best of our knowledge, the proposed LLC architecture experimental evaluation is novel in embedded systems with high performance processors - not yet found similar research work.

The remaining of this paper is organized as follows. Section 2 describes the approach of heterogeneous cache sizes in clustered embedded multicores and related work. Section 3 discusses about the effects using heterogeneous cache sizes. Section 4 discusses the methodology and Section 5 describes the performance analysis and performance versus energy versus area trade-offs involved. Finally, section 6 presents some conclusion remarks and future directions.

II. Background and Motivation: clustered caches and different cache sizes

First, we contextualize the advantages of clustered LLC in regarding the benefits of sharing. Next, the focus is targeted to the aspect of sharing caches of different sizes.

A. Clustered shared caches

As indicated in [24], a clustered multicore is composed by a replication of a tile in 2-D mesh configuration, where each tile contains: (i) its processor, with a L1 instruction and L1 data caches, (ii) one LLC cluster or set of cache banks, and (iii) a connection to the intra-chip network. These elements are illustrated in Figure 1 with LLC represented by L2 caches, as they are the LLC in current families of embedded systems.

As an approach to address cache scalability, non-uniform cache access (NUCA) approach is proposed [25], where a cache is partitioned into several banks and the directory is distributed and held as a replicated set of LLC tags. According to [20], the number of banks and routers of a specific design is a tradeoff made between latency and energy spent for that specific configuration.

Considering (ii), the studies introduced in [20] demonstrated that there exists a compromise between the line size and the bandwidth when bringing the line from the memory to the caches and vice-versa. That is, cores can access each cluster on a private or shared manner. According to the report [24], shared caches present several advantages mainly due to likely data reuse when compared to private accesses. As depicted in Figure 1, if there is a L1 miss, the line is searched in a L2 cache bank through the network, which allows to move lines from remote L2s. Additionally, this report indicates that, if the line is found in some L2 cluster closer to the one that missed this line, that avoids congestion of the network that connects them. In case the line is not found in L2 cache bank, it can be found in one of other shared L2 cache banks. At last, if the line still cannot be found, then a memory access is required. To summarize, if the line is found by searching at local cache clusters, off-chip accesses to the main memory are avoided, thus likely to minimize latency increase [20][25].

In regards to the network that interconnects cache clusters, latency of an L2 miss depends not only on the number of hops needed, but also on the likely congestion from other L2 cache bank requests as well on the coherency messages [20]. Moreover, L2 cache bank access latency is mostly dominated by the time to access the contents and is proportional to the bank size and inversely proportional to the number of banks, the latter including the links and routers effects.

B. Heterogeneous Clusters - Cache Size and Related Aspects

In terms of cache parameters, heterogeneity aspect along the cache clusters can appear under different ways, as (i) number of lines, (ii) line sizes, and (iii) associativity.

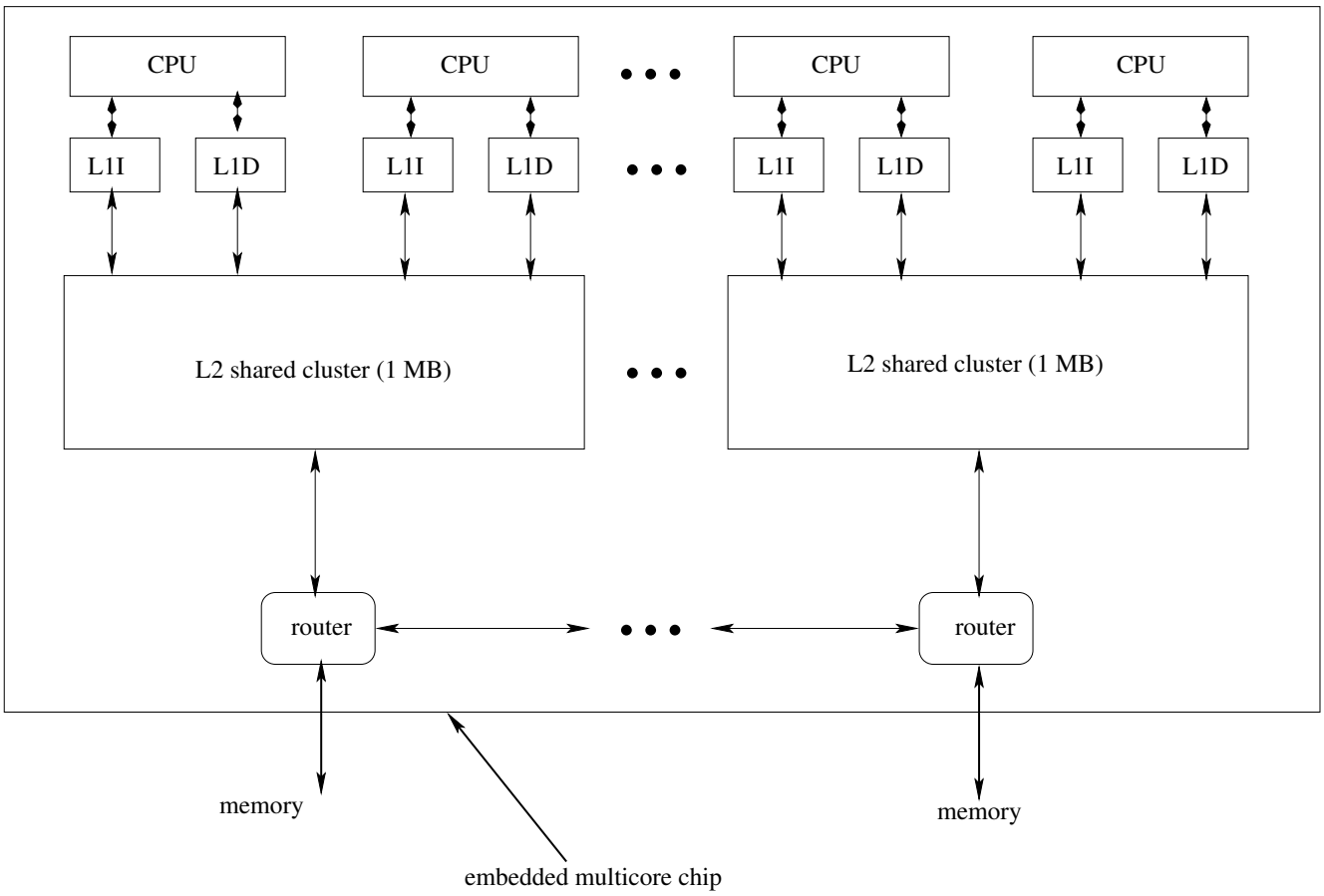

Fig. 1: example of homogeneous LLC modified from [24]

Since (iii) has been significantly explored [15][16], while in (ii) it is typically defined as multiples of 64Bytes, we restrict the focus to the aspect (i) in this research, due to (a) lower cache circuit complexity to disable cache lines, (b) predictability, and (c) likely facility of cache designing, fabrication, or obtaining them via shutdown through different number of lines [20].

To exemplify the parameter (i) number of lines, a homogeneous cluster of an embedded processor is depicted in Figure 1 , while Figure 2 illustrates a heterogeneous one.

III. Heterogeneous cache clusters - Qualitative Analysis and Algorithm

In this section it is discussed the behavior of heterogeneous clusters by assuming that they are originated from homogeneous ones. The explanation on the behavior of heterogeneous systems is through a simplistic qualitative modeling that easily illustrates the designer the implications of heterogeneity in terms of number of lines. Moreover, we also present a flexible switching mechanism in the sense it can be flexibilized to use different parameters such as general ones - energy, performance - or architectural ones - hit rates, coherency - based on the homogeneous cluster behavior.

Assuming a specific parameter is selected, the algorithm determines the respective homogeneous curves. From the closest heterogeneous configuration that present similar behavior, it selects the configuration that presents the smallest area. The algorithm behavior (i) follows the behavior of a design architect who is focused on the specific architectural parameter, as well as (ii) on energy saving (by selecting the configuration with smaller area to save power).

A. Different number of cache lines

Heterogeneous clusters using different number of cache lines are interesting alternatives to have different cache sizes [2]. Given that the total cache size is obtained through the product of cache associativity, block size, and number of lines, by having associativity and block size fixed, an asymmetry on the number of lines implies the asymmetry of cache sizes. For instance, two cluster with different sizes though similar associativity and block size, the one that presents larger number of lines is represented by Ncl while the one with smaller number of lines is represented by Ncs. It may appear as:

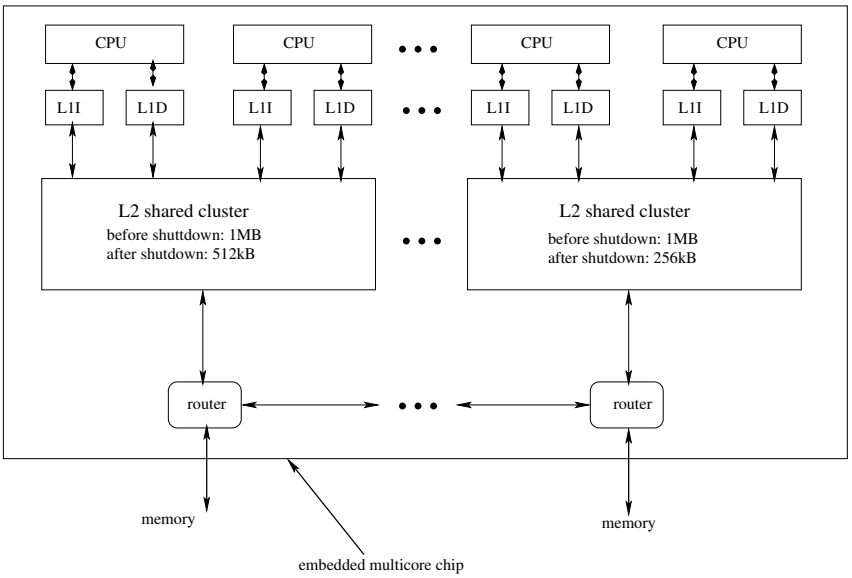

Fig. 2: example of heterogeneous LLC modified from [24]

$$Ncs \ < \ Ncl \tag{1}$$

These parameters are considered and used to qualitatively explain the behavior of several cache cluster aspects, then define smaller clusters as clusters with Ncs and larger ones, with Ncl such as $Ncs \ < \ Ncl$.

Generally speaking, the use of heterogeneous cache clusters may have different effects: (A) different L2 hit rates; (B) small L2 cache clusters closer to larger ones, likely to be served by the larger ones when a search is required; (C) decrement of L2 coherency traffic on the smaller cluster sizes; (D) L2 energy-saving and area saving; (E) performance/energy(power); (F) different L2 cache size distribution versus data distribution. We proceed next with a brief discussion about these topics.

Alternatively, we have also observed that heterogeneous hold asymmetry in terms of number of lines, given that they present opportunities to improve their efficiency in situations where some cores are often used rather than others. In this case, the use of heterogeneous clusters is likely to be better tuned to the demands of the application in terms of energy utilization. Due to replication and coherence, smaller and regular-sized caches are likely to present the most intensively-used used cache lines, i.e., the smaller ones after the shutdown operation are likely to turn down the less used lines to improve locality. As a consequence, reduced number of lines implies on the reduction of power consumption comparatively to the regular-size or larger ones.

B. Different hit rates

Due to smaller number of lines when compared to the respective homogeneous ones, heterogeneous clusters are likely to have more capacity misses. Taking the execution of the same benchmark in both configurations, we should achieve smaller hit rates as a consequence. With an L2 composed by a mix of heterogeneous and homogeneous clusters and considering $Hrhet$ as the average hit rate correspondent to heterogeneous ones, also $Hrhom$ of the homogeneous ones, we state that:

$$Hrhet < Hrhom \tag{2}$$

Furthermore, by geometrically averaging heterogeneous and homogeneous clusters, we have:

$$Hrgmean = \sum_{i=1}^{numcaches} \frac{Hrhet * Hrhom}{numcaches} \tag{3}$$

where $Hrgmean$ means the geometric mean of hit rates or the overall hit rate, while $numcaches$ corresponds to the number of cache clusters.

Considering equations 2 and 3, we can observe that $Hrgmean$ is lower than $Hrhom$ and larger than $Hrhet$ in heterogeneous clusters. Thus, the use of heterogeneous clusters lowers the overall hit-rate, which represents one aspect of the trade-off when utilizing heterogeneous cache sizes. We further discuss the energy aspect.

When comparing sizes of homogeneous/heterogeneous clusters, we can state that:

$$Hrsmaller < Hrlarger \tag{4}$$

where $Hrsmaller$ represent hit-rates for smaller cluster sizes while $Hrlarger$ for larger sizes ones.

Similarly, we calculate the time spent on misses to determine the effect on smaller clusters, likely to spend longer times with misses when compared to the balanced or larger ones:

$$\begin{aligned} Mtsmaller &= (1 - Hrsmaller) * mtsmaller \\ Mtlarger &= (1 - Hrlarger) * mtlarger \end{aligned} \tag{5}$$

where $Mtsmaller$ represents the total miss time spent on smaller clusters, $mtsmaller$ smaller cluster cache miss time, $Mtlarger$ the total miss time spent on larger clusters, $mtlarger$ the larger cluster cache miss time, and $Mtsmaller$/$Mtlarger$ obtained through the product of the miss rate by the smaller/larger cluster cache miss time ($mtsmaller$/$mtlarger$).

Since smaller clusters present smaller latencies when compared to larger ones, we can state that:

$$mtsmaller/ < mtlarger \tag{6}$$

However, since $Htsmaller$/$Htlarger$ depends on applications' behavior, considering individual equations in 5 as well as lower cache miss time of mts in 6, cores executing tasks with accesses to smaller clusters are likely to present lower performance comparatively to the homogeneous/larger ones, i.e., $Mtsmaller > Mtlarger$.

Nevertheless, due to its smaller clusters, heterogeneous versions have smaller hit rates than respective homogeneous clusters ($Hrsmaller < Hrlarger$). Larger clusters are likely to provide the requested line to the smaller ones. Assuming a heterogeneous configuration with a mix of these clusters, we can derive the average hit-rate by combining equation 2 to 4:

$$Hrgmeom < \sum_{i=1}^{numcaches} \frac{Hrlarger^2}{numcaches} \tag{7}$$

which shows that the obtained average hit-rate $Hrgeom$ is smaller than the hit-rate of larger heterogeneous or homogeneous clusters.

C. Larger caches serving smaller ones

Remote clusters are likely to back up lines of another cluster, due to their higher hit-rates which favor the chances of serving as a repository for smaller ones. Due to this, large clusters are likely to be more requested than the smaller ones, in terms of cache lines requests. As consequence, additional energy is also spent the buses connected to these caches. To measure the energy utilized, the following is proposed:

$$\begin{aligned} Ensmaller &= \sum_{i=1}^{numcaches} (Hrsmaller * Nrsmaller) \\ Enlarger &= \sum_{i=1}^{numcaches} (Hrlarger * Nrlarger) \end{aligned} \tag{8}$$

where $Ensmaller$ is the energy spent on the small clusters, $Enlarger$ on the larger ones, $Nrsmaller$ the number of smaller cluster requests, and $Nrlarger$ the number of larger cluster requests. Similarly, due to smaller hit-rates, in the case of requesting lines to smaller clusters, these are likely to spend less and have less energy dissipated on the connected buses. To formalize, through equations 8 and 4 we have:

$$Ensmaller < Enlarger \tag{9}$$

D. Decrease of L2 coherence traffic

Considering clusters with imbalanced sizes, coherence is kept according to the program behavior and cache cluster distribution. Assuming the number of replicated cache lines as a function of the number of cache lines, and combined next to equation 1 where smaller clusters are likely to have less coherence activity, we have:

$$C(Ncsmaller) \; < \; C(Nclarger) \tag{10}$$

where C means coherence, number of total lines on smaller clusters $Ncsmaller$, and on the larger ones $Nclarger$. As a result, since smaller clusters are likely to have less amount of replicated cache lines, they are likely to have lower coherence traffic when serving lines to the larger or balanced ones.

E. Energy behavior

In order to better understand how the energy behavior changes with different cache sizes, we have determined the cache energy distribution as well as the average energy-per-bit in order to find the most power saving configuration. Considering that both aspects enable to select the configuration that maximizes performance and minimizes area effectively used, yet being the most power saving one.

The dynamic hit energy per access is the dynamic part of energy which contains the tag array read and the data array read energies. Similarly, the dynamic write hit energy should consider the tag array read and data array write energies. For the cache miss part of the energy, we have considered for the read miss energy, the tag array read, data array read, tag array write and data array write energies. As for the write miss energy, we have considered the tag array read energy, tag array write energy, data array read energy and data array write energy. Roughly, for an access, we can consider that the miss energy is about twice the hit energy.

As for dynamically characterizing the energy spent in each workload for each of the configurations, we have used CACTI [23]. Considering that the number of write hits and misses is typically of low magnitude, we have separated the energy spent in hits and misses and propose the average dynamic energy per access (AvDEA) as follows:

$$\begin{aligned} AvDEA = \; & (hit\_accesses * hit\_energy \\ & + miss\_accesses * miss\_energy)/total\_number\_accesses \end{aligned} \tag{11}$$

or

$$\begin{aligned} AvDEA = \; & h \; * hit\_energy\_per\_access \\ & + (1 \; - \; h) * miss\_energy\_per\_access \end{aligned} \tag{12}$$

Since each bank has its specific configuration, we calculate the dynamic distribution of the cache energy among the banks. In this case, the distribution of AvDEA over the banks (DAvDEA) is defined as:

$$\begin{aligned} DAvDEA = \; & \sum_{\ell}(h \; * \; hit\_energy(bank)) \; + \\ & (1 \; - \; h) \; * \; miss\_energy(bank) \end{aligned} \tag{13}$$

F. Performance versus energy-saving tradeoff

In order to determine the behavior of heterogeneous clusters, we discuss the trade-off between performance and energy saving. Similar to techniques which boost clock frequency in current microprocessors, trading off performance and energy when using imbalanced caches can be beneficial [16][20].

The very immediate consequence of using heterogeneous clusters is to have a certain degree of performance degeneration whilst saving power- due to the presence of smaller clusters. Another key issue is, by comparing two imbalanced configurations, each one presenting total cluster area calculated as the summation of areas of their own respective clusters, configurations with smaller cache sizes have lower energy utilization than the larger area ones, and the larger ones have comparatively better performance. We exemplify this aspect with different configurations and energy/performance magnitudes in Section IV.

G. Cluster size versus Data distribution

Taking into consideration that no manual data distribution, no special compiler flag, either special compiler, heterogeneous configurations are likely to have more capacity misses than homogeneous ones (equation 4), due to the use of smaller caches comparatively to the correspondent homogeneous configurations.

When approaching coherence, clusters may hold program variables, whereas each one has different utilizations. In this case, some clusters hold data more intensively, whilst others hold less oftenly as an indirect reflection of data distribution.

H. Flexible Algorithm to Select a Configuration and Switch

Before bringing into description the algorithm that switches from homogeneous to heterogeneous configurations, we define some terminologies and their context. We assume that we have different homogeneous configurations (different sizes), from which we derive the heterogeneous configurations. Therefore, heterogeneous configurations can be seen as a combination of different cache sizes obtained among homogeneous configurations. Examples of these configurations are defined in the Experimental section.

The proposed switching algorithm is flexible in the sense that it can be easily modified to approach cache coherency traffic - or any other cache architectural aspect - rather than energy aspect that typically interests the design architect. It is based on a combination of reading performance counters with the implementation of the qualitative modeling equations we have developed, yet CACTI [23] estimation modeling. Within regular periods of time, the behavior of all configurations is sampled and stored. The core of the selection mechanism method is similar to what a design architect would perform based on the modeling presented in the previous subsections:

- Through running a sample - further described - of all configurations, we determine the behavior of analysed parameters (energy, performance, both combined, energy over performance, read hit rate, write back traffic, etc.) versus cache size for the homogeneous configurations. Once having such behavior data (e.g., hit rates, read misses, coherence write backs) versus cache size of different homogeneous cache sizes, they will not only allow us to determine the standard behavior, as well as to compare them to the behavior of the heterogeneous ones. Therefore, we can determine how close the heterogeneous configurations are when compared to the homogeneous ones in terms of the parameter analysed (energy, performance, etc).
- From the previous step, we can determine several different heterogeneous configurations that have a closer behavior to the homogeneous ones, as described and exemplified in the Experimental Section.
- From different heterogeneous configurations that present behavior closer to the homogeneous ones, the heterogeneous configuration is then selected using the smaller area as selection criteria.

Next, it is discussed in detail the execution of the algorithm procedures. We assume the availability of performance counters such as in [2] and/or implementation of simple hardware counters to monitor cache architectural aspects such as total number of misses, hits, and cache traffic for each of the cache clusters. Yet, we assume that a small content address memory (CAM) is available to store CACTI-based estimations regarding the energy used by the different heterogeneous memory sizes.

Besides energy estimations, the CAM also stores the all possible heterogeneous cache sizes magnitudes. Since the number of configurations is small, the CAM is also small. When using pre-computed energy magnitude estimations determined with CACTI tool, we avoid incorporating specific circuitry and respective overhead.

Similarly, in the CAM estimations for other parameters could be similarly stored, which enables the algorithm to be flexible in terms of design selection strategy. Based on the previous reports developed in [2][3][4], CAM-based structures are likely to be of reduced size, therefore their complexity overhead is neglectable.

In order to estimate the amount of energy, time is also monitored (performance counters or similar) in order to determine the energy used by the caches. Using this time information and CAM energy estimations, we derive the energy used in the system.

We assume each configuration is sampled for a reduced-magnitude time interval (e.g. execution of 200,000 instructions), yet a minimum interval between consecutive samples (e.g. 10 times the sampling time), so that the sampling time and interval do not either affect execution of the programs themselves or the switching behaviors.

The following steps describe the algorithm steps in Algorithm 1:

1) ”measure_energy_and_performance_for_a_determined_amount_of_time_using_homogeneous_configurations”: we propose to perform these operations by having a simple set of registers or small content address memory and using them to store the related statistics to homogeneous configurations, so that we can easily derive the homogeneous curve (such as the ones further showed in Section IV).
2) ”calculate_the_energy_related_homogeneous_curve”: Using the homogeneous magnitudes obtained in the previous step, we can determine the homogeneous curve by extrapolating it for all different cache sizes (stored in a small CAM) using polynomial least squares method.
3) For each of the heterogeneous L2 cache sizes previously stored, we determine the correspondent magnitude at the homogeneous curve. This way, we are able to find the heterogeneous configuration that is the closest to the homogeneous one in terms of the analysed magnitude (e.g., energy, performance, energy over performance, etc) for that specific cache size.
4) Performing similarly step 1 but only for the selected configuration.
5) We measure the difference between the energy used by the heterogeneous configuration and the respective homogeneous one (with similar size): (energy_used_by_the_configuration - energy_used_by_the_closer_homogeneous_configuration) is compared to a delta (arbitrary value). If the magnitude difference obtained between the homogeneous and respective

heterogeneous is significantly reduced (e.g., a delta can be defined), the configuration can be selected. If the difference is relevant, we remove this configuration of the set of likely-possible configurations. This step can be performed by a simple comparator circuit where delta can be potentially reconfigured if necessary.

As a final observation, the first step could also be used to compare the homogeneous and heterogeneous configurations currently being tested.

```
previous_delta¯1000;
// arbitrarily high as an initial constant, as an initial reference for delta  while set_of_selectable_configurations do
    measure_energy_and_performance_for_a_determined_amount_of_time_using_configurations_7 _8 _and_20;
    calculate_the_energy_related_homogeneous_curve;
    select_the_heterogeneous_config_with_smaller_cache_sizes_run_it;
    locate_the_selected_configuration_at_the_energy_homogeneous_curve;
    if (energy_used_by_the_configuration - energy_used_by_the_closer_homogeneous_configuration ¡ delta) then
        if delta ¡ previous_delta then
            select_this_configuration_as_candidate;
            previous_delta¯delta;
        end
    end
    increase_configuration_area_size;
    eliminate_the_configuration_from_the_set_of_selectable_configurations;
end
switch_to_the_selected_configuration;
```

Algorithm 1: determine, select, and switch to the determined heterogeneous configuration

Additional discussions on the algorithm are performed in Section IV.

IV. Experimental Section

In this section, we perform series of experiments to demonstrate the trade-offs present in heterogeneous caches. We propose the use of three different homogeneous configurations - one of them set as the baseline - and a series of seventeen (arbitrary selection of this amount) different heterogeneous configurations - considered as most representative ones - that cover most of the combination behaviors and can be derived from the three homogeneous configurations defined. These cache configurations are evaluated in terms of performance, architectural parameters (hit/miss rate, coherence) and energy. In the subsection that follow next we further exemplify these configurations.

A. Methodology

It is assumed that LLC level is represented by L2 to reflect current embedded systems. To evaluate heterogeneous cluster with different cluster sizes, we propose to evaluate significantly different L2 cluster size configurations. For such, L2 cluster of 512 KB, 1 MB and 2 MB sizes are defined, as representatives of L2 cluster cache sizes in embedded systems. Eight cache clusters are arbitrarily selected to analyze the heterogeneity, so then combine these eight clusters and seventeen representative configurations, covering most of combination behaviors.

Distribution of caches are listed in Table Ia. Each distribution corresponds to a specific configuration, owning a correspondent total L2-cluster size that corresponds to the summation of all its L2 cache clusters. For example, configuration 13 presents a L2-cache cluster size with a total size of 11 MB, corresponding to the sum of all its L2-cache clusters.

Based on the combinations of three L2-cluster configurations (512KB, 1MB, and 2MB) defined as homogeneous ones, we have created three homogeneous configurations, characterized in Table Ia and defined as numbers 7, 8 and 20.

In each figure, we have formed sets of configurations that have the same size, and have assigned them the same identification symbol for their points plotted. For example, (3), (6), (16), and (17) have 10MB, while (11), (14), and (15) total of 8.5MB, each of them has a different symbol, squares with middle dot and filled circle respectively.

Configurations can be obtained as follows. Starting from the homogeneous configurations (7, 8, and 20) and assuming that these configurations are able to shut down part of its L2-cache clusters, as we have derived seventeen heterogeneous configurations which size distributions are also listed in Table Ia. We believe that these heterogeneous distributions cover most significant heterogeneous combinations that a clustered cache should have, assuming such an operation of shut down. In order to have an area estimation of caches in each configuration, we have used Cacti tool [23] and added its estimations to the Table Ia. Interconnection area is not considered in this estimation given that we focus on the cluster cache areas.

TABLE I: a: configurations versus L2 cluster size versus area; b: summary of architectural simulation parameters

cluster	0	1	2	3	4	5	6	7	chip	area
conf	MB	MB	MB	MB	MB	MB	MB	MB	L2MB	mm^2
1	1	0.5	1	0.5	1	0.5	1	0.5	6	9.785208
2	1	2	1	2	1	2	1	2	12	15.684628
3	0.5	2	0.5	2	0.5	2	0.5	2	10	13.418140
4	0.5	1	0.5	1	0.5	1	0.5	1	6	9.785208
5	2	1	2	1	2	1	2	1	12	15.684628
6	2	0.5	2	0.5	2	0.5	2	0.5	10	13.418140
7	1	1	1	1	1	1	1	1	8	12.051695
8	2	2	2	2	2	2	2	2	16	19.317560
9	1	0.5	0.5	1	0.5	0.5	1	0.5	5.5	9.218587
10	1	2	2	1	2	2	1	2	13	16.592861
11	0.5	1	2	0.5	1	2	0.5	1	8.5	12.168296
12	0.5	1	1	0.5	1	1	0.5	1	6.5	10.351830
13	2	1	1	2	1	1	2	1	11	14.776394
14	1	0.5	2	1	0.5	2	1	0.5	8.5	12.168296
15	2	0.5	0.5	2	0.5	0.5	2	0.5	8.5	11.943286
16	2	1	0.5	2	1	0.5	2	1	10	13.643151
17	1	2	0.5	1	2	0.5	1	2	10	13.643151
18	1	2	0.5	0.5	2	0.5	0.5	1	8	11.601675
19	0.5	2	2	0.5	2	2	0.5	2	11.5	14.892996
20	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	4	7.518722

parameter	considered values
Processor	OOO,4-width,
# of cores	32 cores
cores sharing a cluster	4
number of sub-banks/cluster	8
L1 cache line size	64 B
L1 cache associativity	2-way
L1 dcache size	32kB
L1 icache size	32 kB
L1 latency	3 cycles
L1 replacement policy	least recently used (LRU)
L2 cache line size	64B
L2 cache associativity	8-way
L2 cache size	512kB, 1MB, 2 MB
L2 latency	10 cycles - 512 kB 12 cycles - 1 MB 15 cycles - 2 MB
L2 replacement policy	LRU
L2 MESI transaction	5 cycles
memory latency	100 cycles

The methodology used in this work is similar to the adopted in related cache studies [24]. We have used accurate system simulators to evaluate different configurations described in this section, which consider detailed models such as an out-of-order (OOO) processor model [26].

The SESC simulator [27] is considered since it contains models for L1, L2, network and memory in order to perform a detailed modeling. Its architectural models include latency and contention of L2 clusters, as also microprocessor out-of-order (OOO) execution.

In order to simplify reporting, all parameters are based on the latencies of the cycle. In this work, we assume 22 nm technology and each CPU is a 4.0GHz OOO processor and 4-wide issue. We employed 32 cores as a representative of future embedded systems. Most importantly, to guarantee enough pressure on the cache system. As an extra observation, although not being evaluated in this experimentation, the proposed technique is likely to present similar results in in-order processors, since it does not depend on the core features, but on the L2 ones.

Considering L1 caches, we assumed dcaches of 32 kB, 64B-line write-back, 2-way associative. Based on CACTI [23] and technology manufacturing parameters defined, we used a 2-cycle latency for L1 LRU that is adopted as the replacement mechanism of L1. The size of icaches is set as 32 kB, maintaining similar parameters to dcaches.

In order to determine L2 cluster latencies for the sizes of 512kB, 1MB, and 2 MB, we have employed CACTI [23], keeping similar other manufacturing parameters. As a result, we employed 10, 12 and 15 cycles for 512 kB, 1 MB and 2 MB respectively. LRU is adopted as the replacement mechanism for L2 clusters.

Configuration (7) with 8MB is the baseline configuration since typical embedded systems present cache clusters with 1 MB (in the selected configuration there are 8 cache clusters, with perform 8 MB). Configuration (8) with 16MB is the largest homogeneous configuration, while (20) the smallest one.

Giving that the focus is on the cache size heterogeneity aspect, we have used a simplistic model to connect L2 clusters. We assumed clusters are connected via a logical bus implemented through routers that connects L2 caches, and each router presents one port that is connected to the next router whilst the other one is connected to the cluster. The focus is given to the L2 clusters rather than to the network which connects them.

When it comes to memory, given the benchmark cache-bound features - as further described- we assumed a memory with 100-cycle-latency. Table Ib summarizes all parameters used in the experiments.

We have selected a set of applications from SPLASH-2 benchmarks [28], since these benchmarks are relevant due to their cache-bound behavior. Applications are executed with class-A input sizes in order to guarantee that benchmarks properly explore all address space of L2s.

In experiments for each benchmark, we have extrapolated the homogeneous curve, which serves as a reference for the homogeneous configurations. As evaluations of parameters (performance, power, etc), the reference represented by balanced curve (homogeneous configuration) is compared to the heterogeneous ones, typically represented by their sizes. Comparisons of heterogeneous configurations are made, based upon (i) the size-equivalent element on the homogeneous curve, or (ii) in absolute terms, i.e., specifically comparing against series of configurations.

Performance is measured via number of execution cycles (simulator statistics). Therefore, performance over energy ratio is calculated by the quotient between the number of cycles and the energy magnitudes measured using SESC infrastructure

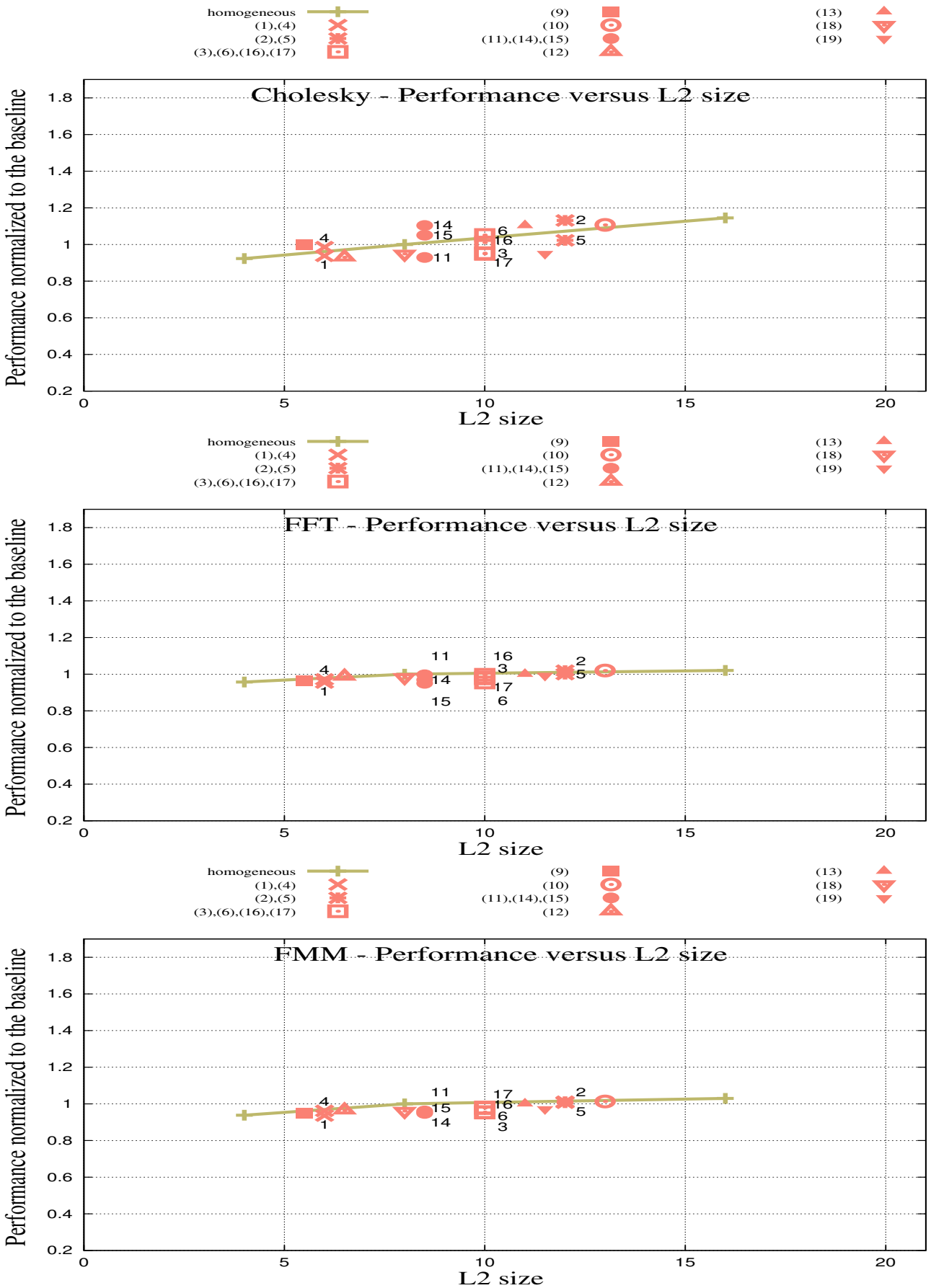

Fig. 3: top to bottom, execution times: (a) Cholesky, (b) FFT, and (c) FMM

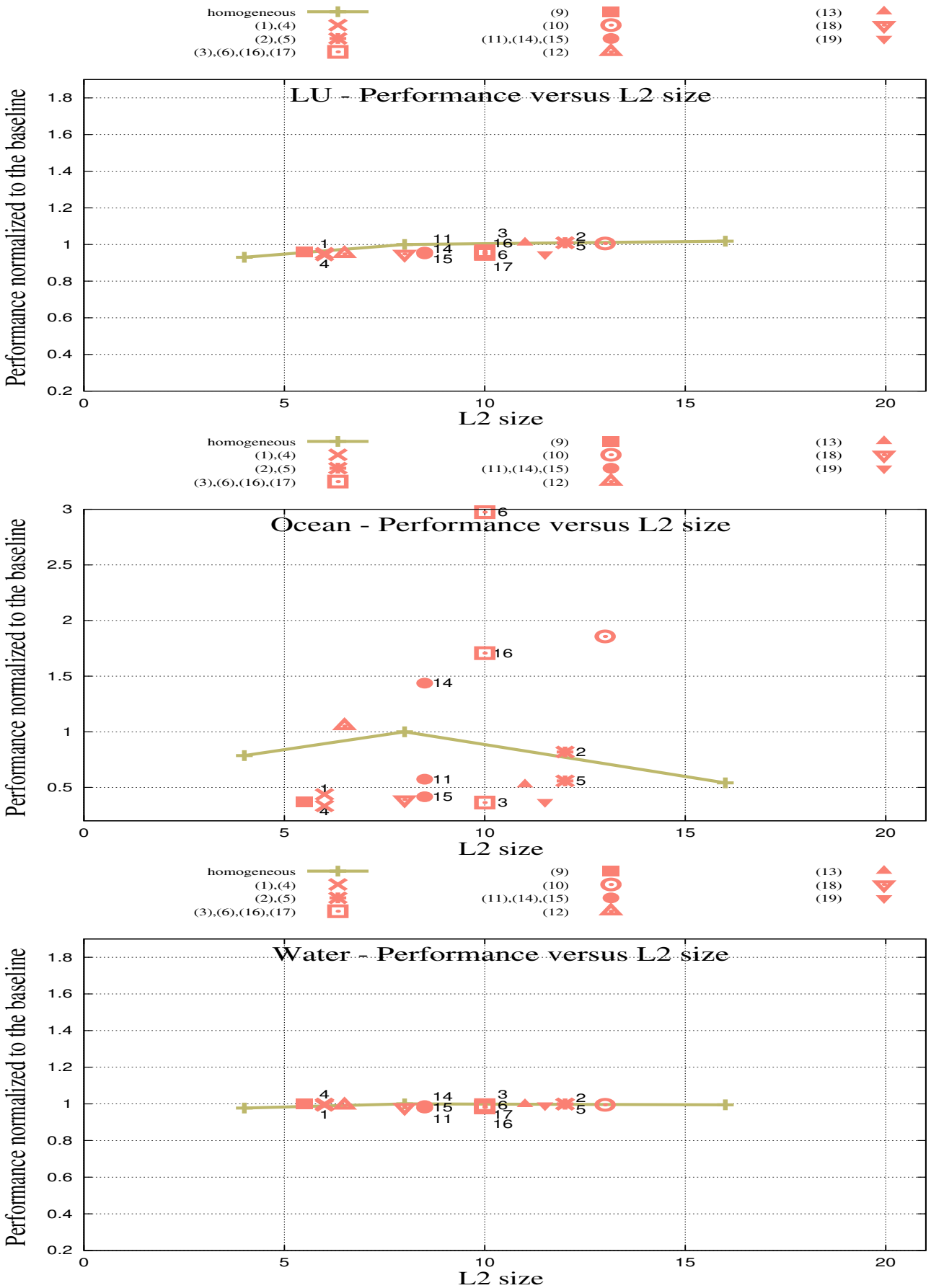

Fig. 4: top to bottom, execution times versus L2 size for (a) LU, (b) Ocean, and (c) Water

[27].

As a parameter is evaluated, when heterogeneous configurations present magnitudes above the homogeneous curve, that means that they present higher magnitudes than the correspondent homogeneous ones. Similarly, for the ones with magnitudes under the curve, these present lower magnitudes. We analyze further each experimental parameter in the following subsections.

In order to reduce and minimize errors of measurement, each simulation was executed ten times, although all results have not shown any significant variability (less than 0.01%). All results are obtained using harmonic averages.

The modeling developed in Section III can assist the cache design architect to qualitatively understand the heterogeneous behavior obtained comparatively to the respective homogeneous one.

In all experiments, we have used SPLASH-2 default data distribution and have not changed any benchmark code or compiler directives to benefit locality.

B. Results and Analysis

The presentation and analysis are done as follows. The behavior of each configuration in terms of magnitude variation of the following parameters versus L2 cache size is observed and analysed: (i) performance; (ii) energy; (iii) L2 hit rates; (iv) number of misses; (v) number of writeback messages, (vi) to be able to have an overview of each individual cluster contributions. The analysis on the speedup based on the measured parameters is performed. Next, area, energy (power) results and their distribution for all clusters and the trade-off between performance and energy-saving for the most relevant configurations are shown.

1) Performance and Energy: We restate that the baseline configuration is configuration (7) with 8 MB and the homogeneous curve represents the behavior of this configuration, configuration (20) with 4 MB, and configuration (8) with 16 MB.

Having the behavior of the homogeneous configurations as plotted yet reference curve and also assuming L2 size as the X-axis, we further observe that there are heterogeneous configurations with the same size (X-axis) and higher performance (above the reference curve), similar (closer to the curve within the 0-3%), and under (with lower performance).

Figure 3a presents the speedup obtained for Cholesky benchmark program. From experiments, we have noticed that the configurations that employ clusters of lower sizes - 512 kB and 1 MB - have lower performance. For instance, configuration 9 presents a better performance (above the balanced curve) than the equivalent 5.5MB-homogeneous one, while (6) and (16) present similar (approximately to the balanced curve) to the 10MB-homogeneous one, and (18) presents a lower one (lower than the balanced curve) than configuration (7) (8MB-homogeneous).

In summary, configurations (2), (10), (13), and (14) present the best performance magnitudes, while (1), (11), (17), and (18) present the lowest ones. Additionally, (1) and (16) present similar performance to the homogeneous size-equivalent ones. Comparing to the baseline configuration (7), the best heterogeneous performances present up to 16% higher performance when compared to the homogeneous ones.

Heterogeneous configurations are placed right to the homogeneous ones, since they are above the reference curve (homogeneous). For example, configuration (10) with 13MB can be obtained from shutting down configuration (20) - itself with 16 MB.

In respect to energy consumption as illustrated in Figure 5a, configurations (3), (6), (11), (14), (15), (16), (17), (19) are heterogeneous configurations with higher energy usage than the respective homogeneous ones. Moreover, configurations (9), (4), (12), (13), (2), (6), and (10) present comparable energy usage to the homogeneous ones. Moreover, configuration (1) presents the lowest energy usage. Compared to the baseline (7), which is the most energy-saving one, lowest energy-consumption configurations spend from 6 to 30% more energy.

Comparing performance and energy analysis previously done for Cholesky, we have noticed the following: (i) the interesting configurations are the ones that present higher performance, and lower or equivalent-to-homogeneous energy utilization, which are (13), (2), and (14); (ii) configurations (2) and (10) present similar performance and energy utilization to the equivalent homogeneous ones, and (iii) configuration (4) presents lower performance and lower energy utilization than the equivalent homogeneous one.

From the configurations mentioned and considering both performance and energy aspects, the one that has the lowest energy utilization is configuration (13), while the one that show best performance is (2). Between these two configurations, the one that has the smallest area is (13). Particularly, configuration (13) performs about 4% worser than configuration (8), which is the fastest configuration, spending about 10% more energy than configuration (7), which is the most energy-saving configuration. Therefore, by comparing performance versus energy ratio - in Figure 7a, we observe that configurations (13), (2), and (10) present the best ratios. Interestingly, configurations (1), (2), and (13) present better performance versus energy ratio than the homogeneous size-equivalent ones (shown above the curve in charts). Furthermore, in case of shut-down operation on L2 clusters in configuration (8) or cluster activation in configuration (20), configuration (13) is the best candidate for shutdown, since it presents the best performance over energy ratio.

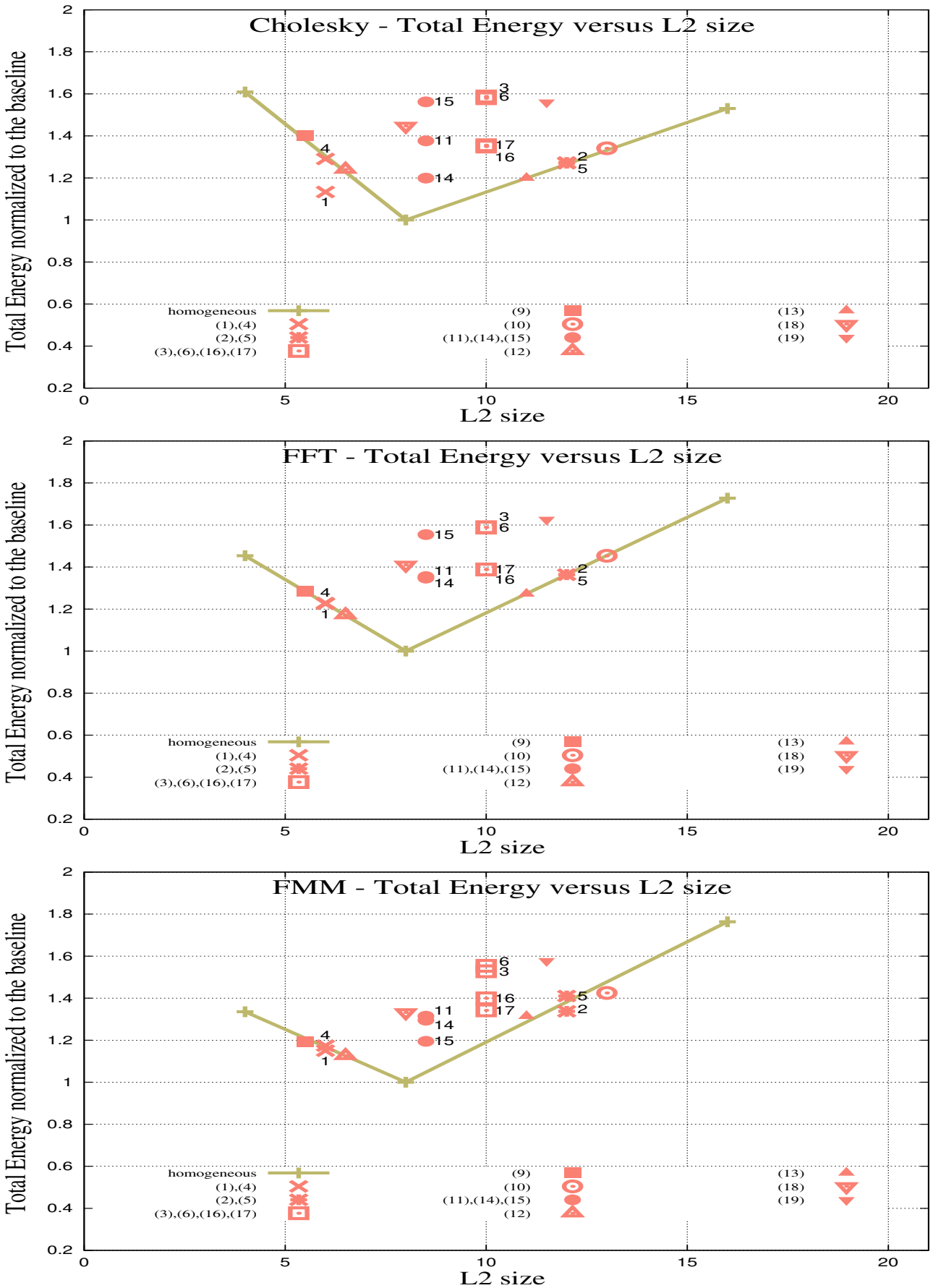

Fig. 5: top to bottom, energy versus L2 size for (a) Cholesky, (b) FFT, and (c) FMM

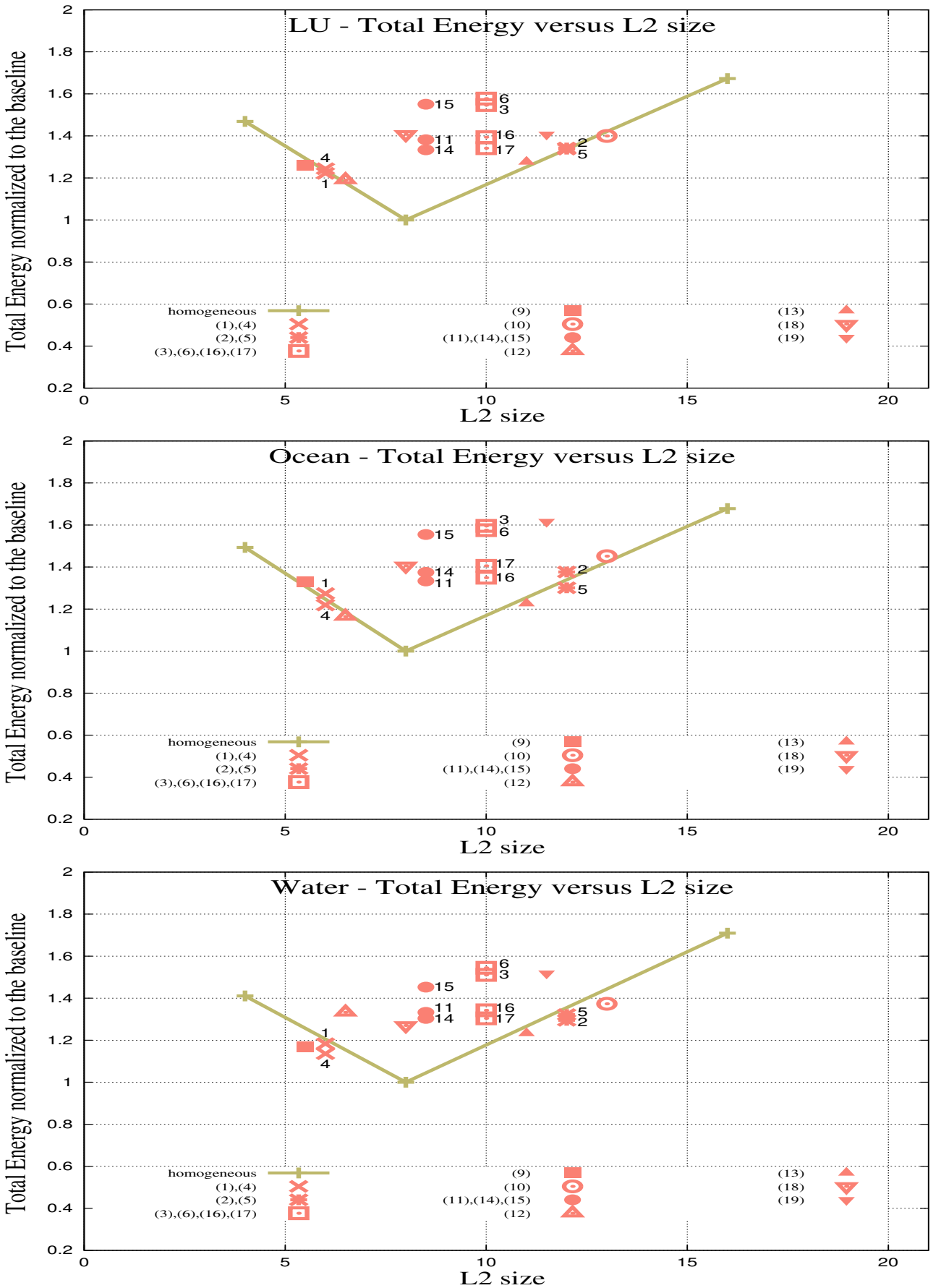

Fig. 6: top to bottom, energy versus L2 size for (a) LU, (b) Ocean, and (c) Water

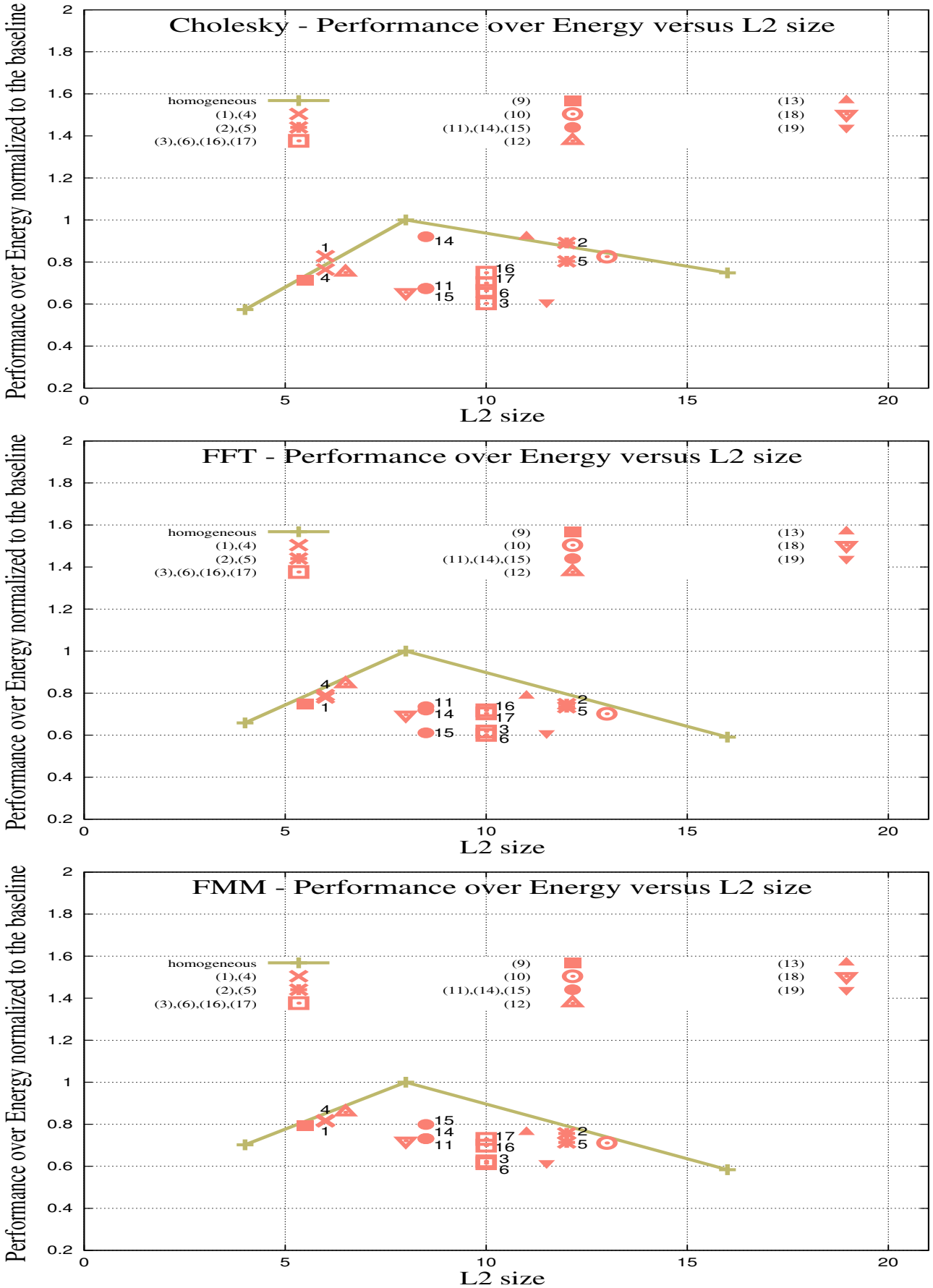

Fig. 7: top to bottom, performance versus energy versus size for (a) Cholesky, (b) FFT, and (c) FMM

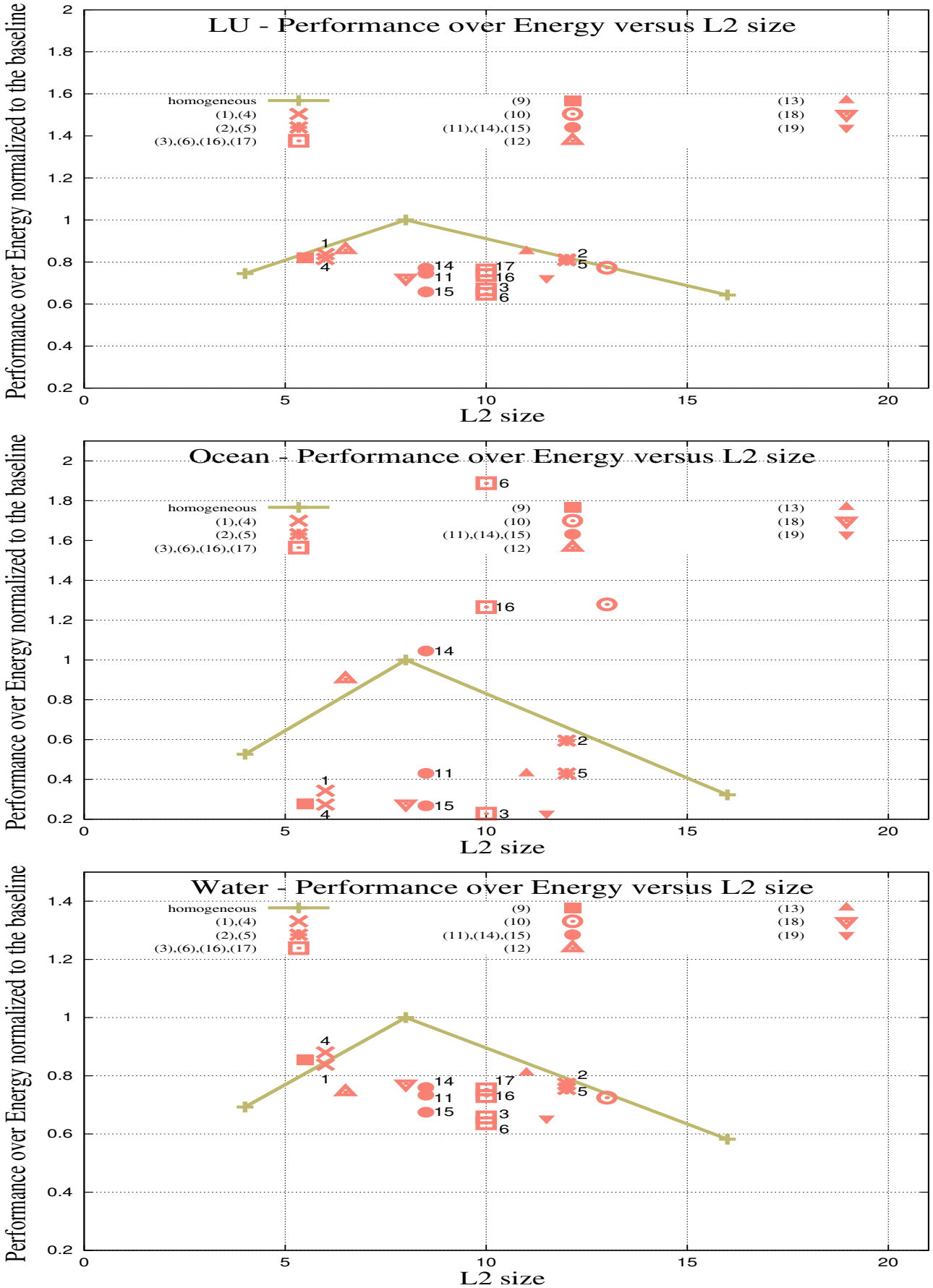

Fig. 8: top to bottom, performance over energy versus size for (a) LU, (b) Ocean, and (c) Water

Performing the same analysis on FFT (Figure 3b), we observe that configuration (10) is the one that presents the best performance, while (2) and (13) present similar performance to the homogeneous ones. Moreover, configurations (12), (1), and (9) present the lowest energy utilization among the heterogeneous configurations (Figures 5b). Compared to the baseline (7), heterogeneous ones present up to 3% better performance, while spend from 17% to 25% more energy.

We also observe that the best performance versus energy ratio behavior for FFT are present in configuration (12) as depicted in Figure 7b. In case of a shutdown of parts of L2 clusters in configuration (8) or similar activation of them in configuration (20), configuration (12) is one the candidate configurations to switch to when the target is to keep the largest ratio performance over energy.

As shown in Figure 3c, it is observed that the configuration (10) presents the best performance for FMM benchmark program, while (2), (13), and (12) present similar performance to the homogeneous ones. In addition, according to Figure 5c, configurations (12), (2), and (9) present the lowest energy utilization within the heterogeneous configurations range. Compared to the baseline (7), heterogeneous configurations present up to 2% better performance, while spending from 10% to 20% more energy. Comparing performance versus energy ratio behavior as depicted in Figure 7c, we observe that configurations (12), (1), and (9) present the best ratios, thus being the most interesting configurations when shutting down parts of L2 clusters in (8) or activating them in (20).

We observe a similar performance to FMM behavior when it comes to LU benchmark, as shown in Figure 4a. Configuration (10) presents the best performance while (2), (13), (1), and (9) present similar performance to the homogeneous ones while, (12), (1), and (4) present the lowest energy utilization among the heterogeneous configurations (Figure 6a). Compared to the baseline (7), heterogeneous configurations present up to 2% better performance, while spend from 20% to 30% more energy. Comparing performance versus energy ratio - Figure 8a, we observe that configurations (12), (13), and (1) present the ones with best performance over energy ratio. In particular, these latter configurations are the most appropriate to be switched when considering a shutting down operation of L2 cache parts on (8) or activation of them on (20).

As indicated in Figure 4b for Ocean benchmark, we observe that (6), (10), and (16) present the best performance, while configurations (12), (13), and (4) present the lowest energy utilization among the heterogeneous configurations (Figure 6b). Compared to the baseline (7), heterogeneous configurations present up to 90% better performance, while spend up to 50% more energy. Comparing performance versus energy ratio, as shown in Figure 8b, we observe that configurations (6), (10), and (16) present the largest ratios. As a consequence, these latter configurations are the most interesting when a shutdown operation of parts of L2 cache is applied on (8) or an activation of these parts on (20).

Observing Figures 4c and 6c) as results for Water benchmark program, we conclude that configurations (10),(2), and (13) are among the ones with the best performance, while configurations (4), (1), and (9) present the lowest energy usage, assuming the equivalent homogeneous as a reference. Comparing to the baseline (7), heterogeneous configurations present similar performance, while spend up 30% to more energy. Comparing performance versus energy ratio in Figure 8c, we observe that configurations (1), (4), and (9) are among the ones which present the best ratios. Due to that reason, these latter configurations are the best candidates in terms of configuration to be switched to when considering shutdown of L2 caches of configuration (8) or activation of them in configuration (20).

Analysing all previous observations about the versions which present best performance and lowest energy usage, we conclude that considering the configurations that appeared in most benchmarks, we have:

- Configurations (10), (2), and (13) are among the configurations which present best or equivalent performance to the homogeneous version.
- Configuration (12), (1), (2), and (9) are among the ones which present the lowest energy consumption.
- Combining the two above conclusions, we conclude that the interesting configurations that benefit the pair performance/power is (2).
- Comparing performance over energy results previously discussed, the most interesting configurations are (12), (1), and (2). These are the natural candidates when shutting down clusters of configuration (8) or turning them on from configuration (7) or (4).

We further combine energy, performance and size to determine the configuration that has the lowest energy utilization and best performance per area size (Figures 12 and 13, also listed in table II). In this case, we have found that the most interesting configurations are (9), (12), and (1). In addition, as a further example, we determine the configurations with the best hit rates and smaller writeback traffic.

Table II summarizes all these previous comparisons in terms of performance, energy, combination of the previous, and the remaining parameters such as hit rates, and writebacks, both further determined.

2) Number of Hits, Misses, and Write backs: Since each benchmark has three different parameters analysed on a separate curve, in order to avoid repetitions we only show the results related to Cholesky. The remaining results are located in Section VIII (Appendix, through Figures 14 to 18).

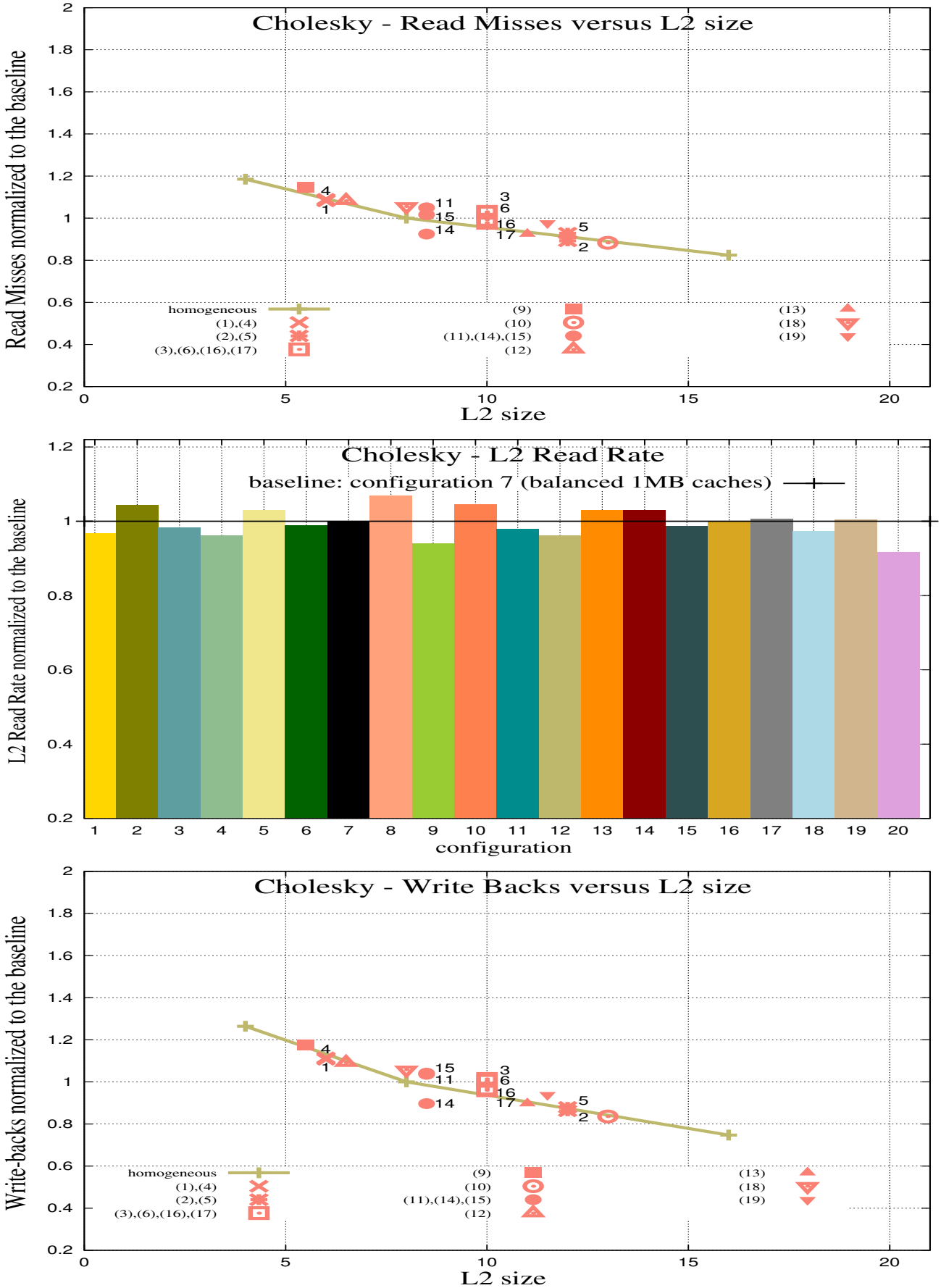

Fig. 9: top to bottom, (a) Miss rate versus size, (b) Read rate versus configuration , (c) writebacks versus L2 size for Cholesky

TABLE II: summary of the best performance, energy usage, performance over energy configurations

benchmark/	best performance	best energy usage	best performance over energy	best performance over energy over size	read hit rate	writeback
Cholesky	2,10,11	1,4	13,11,2	1,9,4	10,2,13,14	2,14,13,19,17
FFT	10,2,13	12,1	12	9,4,1,12	10,5,2,13	10,2,5,17
FMM	10,2,13	12,2,9	12,1,9	9,2,1,12	5,13,10,2	10,5,13
LU	10,2,13	2,13,1	13,12,2	9,1,4,12	2 to 17	5,10,13,2
Ocean	10,3,11	12,1,9	3,6,10,11	17,6,12,16	10,2,5,13	10,2,19,5
Water	10,2,13	1,4,2	4,9	9,4,1	10,2,5,13	10,2,5,13

In this subsection, we analyze the distribution of read hits and misses distribution on each cluster. To give a global overview of the behavior of these parameters in all configurations, we also show each individual behavior in each one.

For Cholesky, as showed in Figures 9a, along heterogeneous configurations, as L2 size is increased, the number of read misses decreases, as predicted when performing the proposed modeling (Section III) - the larger size, the large the hit rate and the lower the number of read misses. We observe similar behavior to the previous from the lower configuration (20) to the (7, baseline) one, while for the largest L2 one (8), the number of misses is increased.

Furthermore, we show a global overview of the number of misses for each different configuration in Figure 9b. This view helps to observe that in configurations where L2 size alternates among successive L2 clusters, the number of hits, misses, follow this alternated behavior. For example, in all heterogeneous configurations except (7), (8), and (20), as showed in Figure 9a, we observe that read rates follow this behavior.

The largest number of misses appear in heterogeneous configurations which have L2 sizes significantly different between successive L2 clusters, which can be easily noticed on the modeling equations developed previously (Section III). For example, configuration (3) has L2 sizes of 0.5MB and 2MB alternatively placed. Therefore, when shutting down caches, in order to avoid higher number of writebacks configurations with this feature should be avoided. Similarly to the number of misses parameter previously analysed, the number of writebacks follow this behavior. For Cholesky, the largest number of L2 hits appear at the cluster 8, due to its intrinsic data distribution.

As L2 sizes are increased, the higher the chance of a data hit, therefore the the number of memory writebacks decrease.

Since each configuration has its particular L2 size, its L2 size reflects the coherence messages presented in it. As a general rule, large L2 clusters have a higher number of coherence messages and small L2s have a lower number of coherency messages. In order to estimate the number of coherency messages, we measure the number of L2 writebacks once the experiments utilize MESI protocol.

Figure 9c shows the number of writebacks related to the heterogeneous configurations. It is interesting to note that the number of writebacks in these configurations follows closer (about 5%) the respective homogeneous ones. Therefore, the use of heterogeneous sized caches does not increase coherence activity. In addition, shutting down operations would not increase the number of coherency messages if such mechanisms were adopted. Some of the heterogeneous configurations present lower number of writebacks when compared to homogeneous ones. For example, in Figure 9c, while configuration 10 with 13MB presents the smallest number of writebacks, it has similar number of writebacks to the respective homogeneous one. In this case reduction of coherence operations implies in better data use. As a final observation, we noticed that configuration 20 (4 MB) presents in most cases the highest number of invalidates as well as configuration 8 (16 MB) presents the lowest number of them, as a direct consequence of their sizes. Finally, we observe that the obtained coherence behavior follows the modeling analysis done in Section III.

3) Design Space Exploration of the Configurations: trade-off performance, energy, size, and architectural ones: The likely best configurations in terms of trade-off performance and power are the ones which with the highest performance over energy ratio. However, several configurations can be selected in this category. In table II we illustrate different criteria mechanisms to determine the selection of the configurations. For all the subsequent results, we are assuming that all the configurations are executed. We are also assuming that:

1) we collect performance and energy results via sampling the execution of several different benchmarks;
2) we select the largest performance over energy configurations, or performance over energy over size, or energy in separate;
3) within the selected configurations, we identify the lowest area configuration ones based on the configurations listed on the table Ib or among the lowest area ones, the one which presents the best performance over energy ratio.

For example, for Cholesky, the best performance over energy results are (13), (11), and (2). Of these selected configurations, configuration (11) is the lowest area one, while configurations (13) and (2) present best performance over energy ratio.

4) Observing how the Proposed Switch Algorithm is likely to Explore Different Selection Criteria: Since the proposed algorithm goal is to follow the design architect selected criteria, it basically follows the behavior of the configurations in

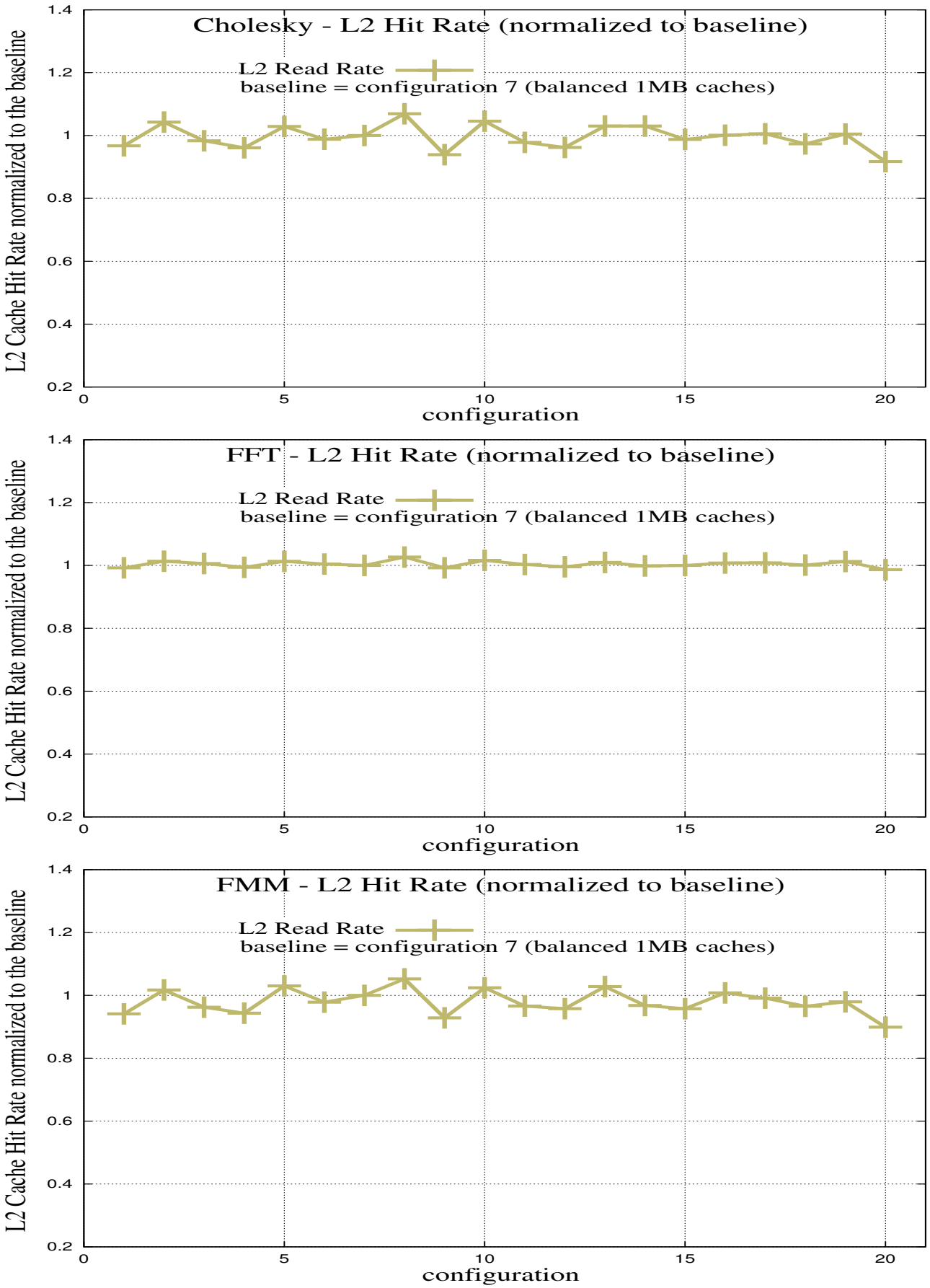

Fig. 10: top to bottom, Read Rate for (a) Cholesky, (b) FFT, and (c) FMM

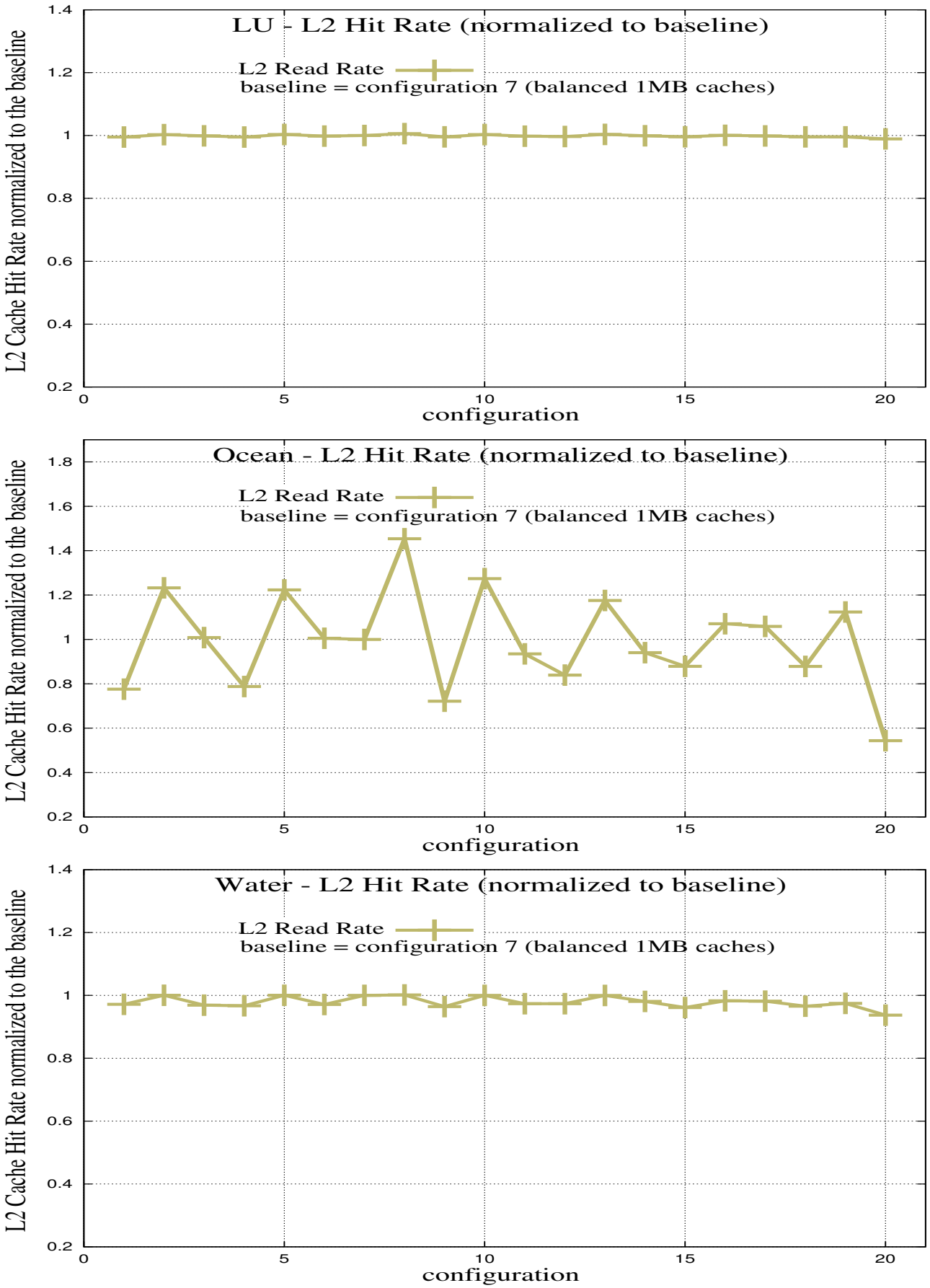

Fig. 11: top to bottom, read Hit Rate for (a) LU, (b) Ocean, and (c) Water

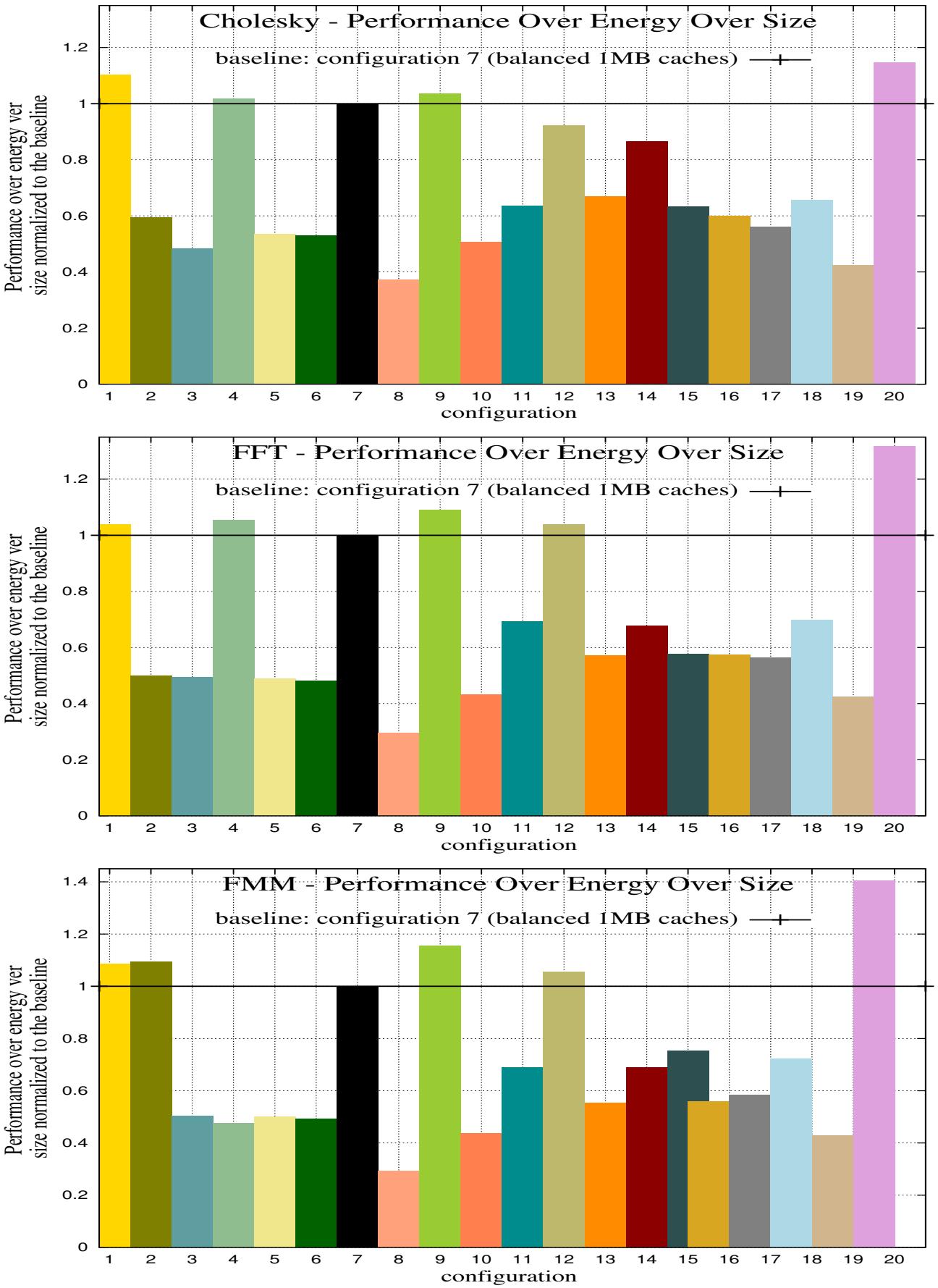

Fig. 12: top to bottom, performance over energy over size for (a) LU, (b) Ocean, and (c) Water

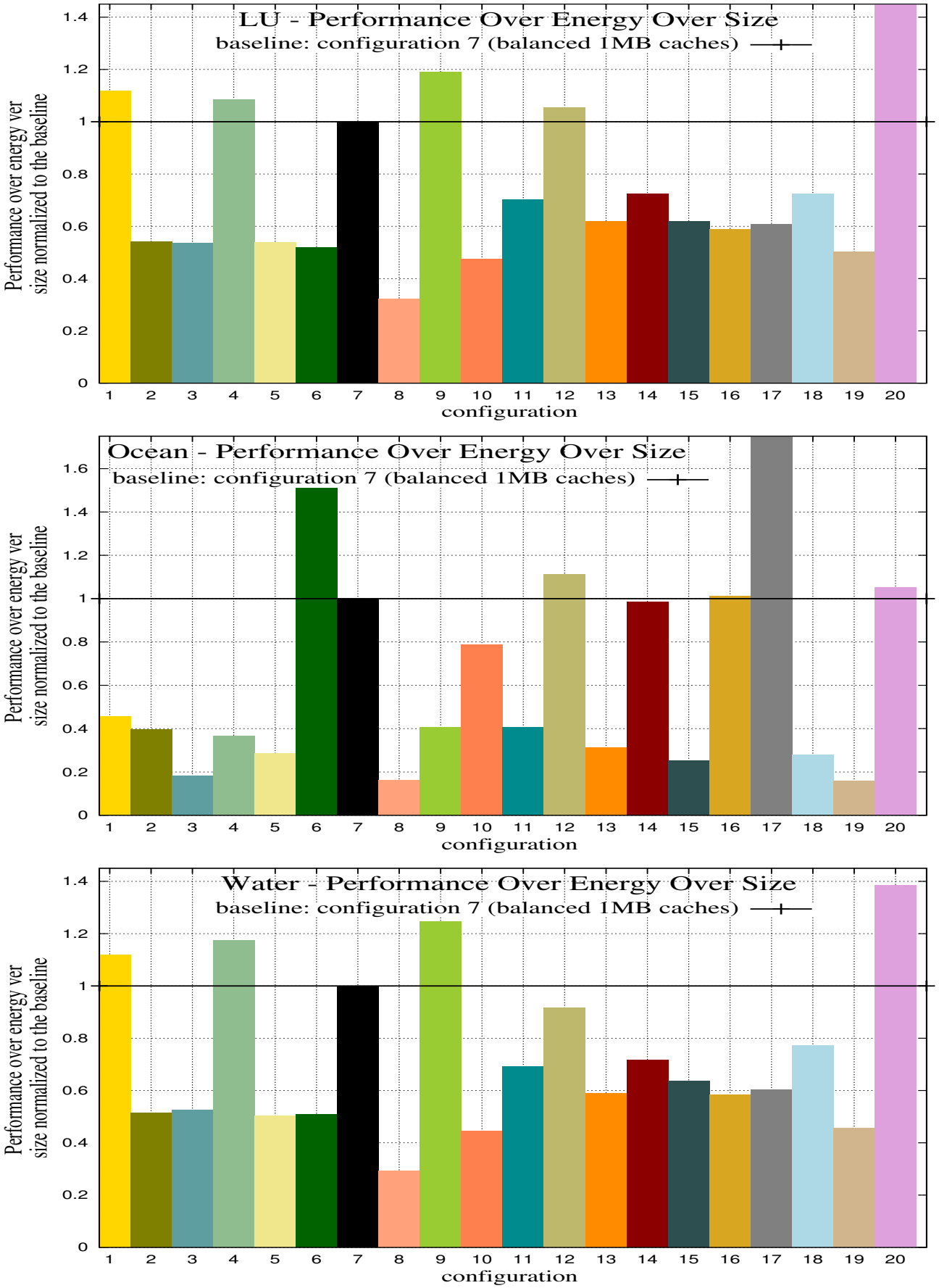

Fig. 13: top to bottom, performance over energy over size for (a) LU, (b) Ocean, and (c) Water

table II. Since the algorithm is flexible, any of the criteria listed on can be used as the selection one. Furthermore, we also assume that a previous run of the configurations was performed and the CAM already contains the values. Moreover, we arbitrarily assume delta as 5%. In addition, we assume that the sampling interval corresponds to the execution of 200,000 instructions for each of the 20 configurations, performing the total time of 4 million instructions to execute them all. The interval between two consecutive samples is corresponding 40,000,000 instructions (or 10x the sampling interval) in order not to disturb the execution behavior.

In order to avoid repetition of previous figures, we assume that the likely configurations obtained in table II are already stored on the CAM table. Rather than energy and/or performance criteria as the switch selection mechanism, to demonstrate its flexibility, we adopt read hit rate. For Cholesky, the likely configurations that present the higher read hit rate are 10, 2, 13, and 14 (table II).

According to the switch algorithm, among these configurations, by looking at the stored table, the one that has smallest area is configuration is 14. Since the read rate magnitude correspondent to it is within delta margin, 14 is selected as a candidate. Therefore, the configuration of the clusters is switched to 14 and it remains configured as 14 until the next run of configurations is performed (if the next candidate is not 14 no switch is needed).

We now turn to another example changing the selection mechanism. We select the number of writebacks as the one. For example, for Cholesky the likely configurations that present the lower writeback traffic rates are 2, 4, 13, 19, 17. According to the switch algorithm, the configuration that has the smallest area among these ones is configuration 6. However, 6 is not within delta margin, then the configuration with smallest possible area should be the likely candidate. Therefore, the next likely candidate corresponds to configuration 17. Since configuration 17 presents itself within the delta margin, it is selected as the candidate. And, in this case, the configuration of the clusters is switched to 17 and it remains configured as 17 until the next run of configurations is performed to determine the next switch.

V. Related Work

The Way Adaptable D-Nuca Cache [16] aims to reduce energy under an acceptable performance loss via bank partitioning and promotion mechanism, where each way can be dynamically turned on/off, depending on the locality of the application. In the report by Bardine et al. [29], the combination of a leakage reduction technique to architectural one targets energy reduction. The former technique is basically a state-preserving leakage reduction technique which sets the voltage of underutilized cache lines (standby) to lower power levels, i.e., to a low power drowsy mode. The latter is an architectural one derived from [16], where power gating is applied to the active ways needed to execute the workloads.

Abella and Gonzalez [15] proposed the concept of heterogeneous cache ways, from which the different cache ways may have different sizes. This technique allows cache energy savings and lower miss rates when compared to traditional caches.

Similar to Flexiway [4], Abella and Gonzalez [15], and Way Adaptable D-Nuca [16] reports, Cooperative Partitioning [6] proposes strategies to organize which ways are accessed, and then, only accessing the needed ones, whilst power-gating the unused ones.

The inter-access time per access count (IATAC) [30] is a hardware technique designed to reduce cache leakage in L2 caches, and dynamically adapts the cache size by dynamically turning off cache lines whose content is not likely to be reused.

Orthogonally to all these previous reports mentioned in the last paragraphs, where heterogeneity is obtained varying the number of cache ways, the approach of cache heterogeneity here adopted employs different cache capacities in order to obtain caches with different number of lines, and likely shutting down parts of these lines to investigate the behavior of a flexible-based shutdown mechanism.

Flexiway [4] is a cache saving technique that is similar to Abella and Gonzalez [15] one in terms of dividing caches into multiple modules and turning off different ways within these modules. In this report, for a certain amount of hits the trade-off between the dynamic energy saved by keeping a sub-way turned on versus the leakage energy saved by turning off the sub-way is the core of the cache saving technique. Furthermore, a threshold mechanism is proposed using this trade-off leveraged by a constant factor, which enables to be application-independent. Our approach employs a flexible algorithm not only considering cache hits and energy as the former, but also any other cache aspect (e.g. cache misses). Furthermore, we approach cache lines rather than ways in different modules, approached in [4].

Embedded Way Prediction [5] is composed of circuit design and architectural technique targeting lower cache energy utilization by using partial tag comparison and the inhibit bit prediction as a mechanism to predict the most used ways. Our approach does not explore either cache tag comparison or bit prediction. We instead compare the heterogeneous cache with the homogeneous and propose a flexible algorithm to switch among them.

EnCache [7] is a cache leakage energy saving scheme that employs profiling cache. Using this scheme, the application can predict the cache usage and energy efficiency in order to reduce leakage. By shutting down parts of the caches we also reduce leakage but not using a profile cache such as in EnCache.

Cashier [8] uses dynamic profiling to estimate memory energy and execution time of the program under multiple last level cache (LLC) configurations. Based on these configurations, it reconfigures LLC to an energy efficient one with a commitment to the deadline. Similar to Cashier, but applied to real-time systems, Wang [9] employs a dynamic reconfiguration and partition technique of private and shared LLC caches is combined to task deadline constraints. In this report, a static profiling determines the interesting configurations for private caches and shared ones while task deadlines are satisfied. Differently, our approach is not focused on real time deadlines. Our proposed technique is concentrated on flexibility rather than real time, in the sense that it can be used not only with performance and energy mechanisms but also architectural parameters in order to partially shutdown caches.

Similar to Cashier, Palette [11] uses a small hardware element named emulator in order to estimate performance and energy consumption of multiple cache configurations and then selects the configuration with least energy consumption.

In the report [10], Hajimiri et al. present a genetic algorithm that is able to propose an efficient dynamic cache reconfiguration and its partition aiming performance and energy efficiency. Our approach is designed to be employed by a designer to assist the

In the report by Kotera [12], a cache mechanism composed of way-allocation function and a power control function both based on the cache locality assessment is proposed.

The report by Benitez et al. [3] proposes a hardware mechanism that is able to generate cache clusters of different sizes basically via reconfiguration of address decoders. We have proposed a flexible switching mechanism that besides using performance and energy as inputs to generate heterogeneous configuration, such as the one proposed by Benitez, our algorithm can also switch configurations based on architectural parameters, thus being closer to the architectural side.

The paper by Benitez et al. [2][31] utilizes previous work [3] and the following metrics, such as cycles per instruction (CPI), static/dynamic processor energy consumption, static and dynamic processor power dissipation, and power-delay product as input elements to feed a threshold switching mechanism for reconfiguring caches and producing heterogeneous ones.

Along the same development direction, Smart Cache [13] presents similar properties to Benitez et al. [2] technique, given the use of selection of ways and controlling the number of lines.

We share with the three previous reports [2][3][13][31] the use of the number of lines to reduce the number of decoded addresses, which allows the reduction of the number of addresses or the cache size.

The reports by Benitez et al. [2][31] rely on the following aspects as part of the trigger mechanism: instruction per cycle (IPC), memory energy, and memory-energy over performance. Differently, our likely mechanism does not rely on processor power, but specifically on LLC cache architectural aspects as well as their respective energy utilization.

Yet different from the former, where hardware reconfigurable decoder has been proposed to change the number of lines, we have proposed a flexible algorithm and qualitative modeling (the latter, in order to validate the experimental results) while being a guide to the design architect. Furthermore, the previous report does not cover multicore configurations, once if there are many cores and different cache bandwidth demands - typical in multicore era - the different bandwidth demands can lead to different conclusions. We instead cover a clustered microprocessor configuration typical in the multicore era (32 cores). Moreover, our approach does not reconfigure the decoders and its addresses such as the former [2][3], however we are considering that disabling high significantly accessed address lines is required to implement it in order to produce heterogeneous ones.

Finally, our approach can employ similar parameters such as performance and energy - when compared to the reports by Benitez [2][3][31]. Differently from these reports, our algorithm can also utilize architectural parameters instead. In this sense, it employs a flexible algorithm.

In the report by Lodde et al. [17], a LLC cache reorganization is proposed based on the fact that there are many private blocks in L1, and based on tag and data entries at the LLC caches. Our strategy is to rely on the L2 level rather than on the L1 one.

In [14], a run-time adaptive i-cache partitioning mechanism is proposed in order to save leakage and dynamic power while performance conscious. Our approach is not related to cache partitioning, but relies on the behavior of the homogeneous configurations to derive the heterogeneous ones.

While being straightforward, our proposed modeling is really helpful to understand the behavior of heterogeneity. By forming heterogeneous caches as a combination of different homogeneous ones, the developed modeling facilitates understanding the expected behaviors by the design architect. For example, miss-rates obtained for heterogeneous caches are obtained as geometric mean of miss-rates of the homogeneous ones. In addition, this Mathematical modeling can save significant number of simulations if only a qualitative analysis is required. In none of the previous reports, this useful but straightforward approach was showed.

VI. Conclusions and Future Works

Heterogeneous-sized L2 clusters are obtained by partially shutting down homogeneous caches [30]. In this work, we have proposed and experimented different heterogeneous-sized L2 cache clusters configurations to verify their performance

and power implications. We have also proposed a flexible algorithm for switching from homogeneous to heterogeneous configurations based on the behavior of energy and performance - in order to approach smaller area yet closer behavior - yet on different cache parameters that potentially interest the design architect the most. We leave a further investigation of the inherent flexible aspects of the algorithm regarding different parametrizations.

We have observed that some heterogeneous configurations present performance up to 16% larger and 10% lower, while energy utilization of 15% lower and 50% higher when compared to homogeneous ones. We have found that heterogeneous configurations with sizes from 5 to 6.5 MB are interesting configurations in terms of performance, energy, and architectural aspects such as hit rates.

Another important consequence of adopting this substrate for morphing is the improvement on the sharing by an effective reduction of the coherence messages. As future works we may aim the combination of the hardware based-decoder proposed in [2] and integrate it to an optical interconnection to either reconfigure the number of lines or ways. In this future approach, designers may use this flexibility combined to optical low power utilization. In addition, we intend to derive internal distributions of cache temperature and incorporate them into the proposed algorithm.

We believe that homogeneous-to-heterogeneous morphing is very appropriate in multithreaded processors. Furthermore, we believe the proposed approach can take advantages of commuting threads on misses of smaller L2 caches.

VII. Bibliography

References

[1] *A Hardware Evaluation of Cache Partitioning to Improve Utilization and Energy-efficiency While Preserving Responsiveness*, ISCA '13, (New York, NY, USA), ACM, 2013.

[2] Benitez, Domingo and Moure, Juan C. and Rexachs, Dolores and Luque, Emilio, "A Reconfigurable Cache Memory with Heterogeneous Banks," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, (3001 Leuven, Belgium, Belgium), pp. 825–830, European Design and Automation Association, 2010.

[3] Benitez, D. and Moure, J.C. and Rexachs, D.I. and Luque, E., "A Reconfigurable Data Cache for Adaptive Processors," in *Reconfigurable Computing: Architectures and Applications*, vol. 3985 of *Lecture Notes in Computer Science*, pp. 230–242, Springer Berlin Heidelberg, 2006.

[4] S. Mittal, Z. Zhang, and J. Vetter, "FlexiWay: A cache energy saving technique using fine-grained cache reconfiguration," in *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, pp. 100–107, Oct 2013.

[5] Sleiman, F.M. and Dreslinski, R.G. and Wenisch, T.F., "Embedded way prediction for last-level caches," in *Computer Design (ICCD), 2012 IEEE 30th International Conference on*, pp. 167–174, Sept 2012.

[6] Sundararajan, K.T. and Porpodas, V. and Jones, T.M. and Topham, N.P. and Franke, B., "Cooperative partitioning: Energy-efficient cache partitioning for high-performance CMPs," in *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pp. 1–12, Feb 2012.

[7] Mittal, Sparsh, and Zhang Zhao, "Encache: Improving cache energy efficiency using a software-controlled profiling cache," in *IEEE International Conference On Electro/Information Technology*, pp. 1–12, May 2012.

[8] Mittal, S. and Zhao Zhang and Yanan Cao, "Cashier: A cache energy saving technique for qos systems," in *VLSI Design and 2013 12th International Conference on Embedded Systems (VLSID), 2013 26th International Conference on*, pp. 43–48, Jan 2013.

[9] Weixun Wang and Mishra, P. and Ranka, S., "Dynamic cache reconfiguration and partitioning for energy optimization in real-time multi-core systems," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pp. 948–953, June 2011.

[10] Hajimiri et al., "Dynamic Cache Tuning for Efficient Memory Based Computing in Multicore Architectures," in *International Conference on VLSI Design*, January 2013.

[11] Mittal, S. and Zhao Zhang, "Palette: A cache leakage energy saving technique for green computing," in *Series Advances in Parallel Computing Ebook*, pp. 46–61, Jan 2013.

[12] Kotera, Isao and Abe, Kenta and Egawa, Ryusuke and Takizawa, Hiroyuki and Kobayashi, Hiroaki, "Transactions on high-performance embedded architectures and compilers iii," pp. 135–153, 2011.

[13] Sundararajan et al., "The Smart Cache: An Energy-Efficient Cache Architecture Through Dynamic Adaptation," vol. 41, pp. 305–330, April 2013.

[14] Paul, M. and Petrov, P., "Dynamically Adaptive I-Cache Partitioning for Energy-Efficient Embedded Multitasking," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, pp. 2067–2080, Nov 2011.

[15] J. Abella and A. González, "Heterogeneous way-size cache," in *Proceedings of the 20th Annual International Conference on Supercomputing*, ICS '06, (New York, NY, USA), pp. 239–248, ACM, 2006.

[16] A. Bardine, P. Foglia, G. Gabrielli, C. A. Prete, and P. Stenstrm, "Improving power efficiency of D-NUCA caches," *ACM SIGARCH Comput. Arch. News*, vol. 35, no. 4, pp. 53–58, 2007.

[17] Lodde, Mario and Flich, Jose and Acacio, Manuel E., "Dynamic Last-level Cache Allocation to Reduce Area and Power Overhead in Directory Coherence Protocols," in *Proceedings of the 18th International Conference on Parallel Processing*, Euro-Par'12, (Berlin, Heidelberg), pp. 206–218, Springer-Verlag, 2012.

[18] Shane Bell et al., "TILE64TM Processor: A 64-Core SoC with Mesh Interconnect," in *ISSCC*, pp. 88–90, IEEE, 2008.

[19] M. Gebhart, B. A. Maher, K. E. Coons, J. Diamond, P. Gratz, M. Marino, N. Ranganathan, B. Robatmili, A. Smith, J. Burrill, S. W. Keckler, D. Burger, and K. S. McKinley, "An evaluation of the trips computer system," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, (New York, NY, USA), pp. 1–12, ACM, 2009.

[20] *Interconnect Design Considerations for Large NUCA Caches*, ISCA '07, (New York, NY, USA), ACM, 2007.

[21] *Last-level Cache Deduplication*, ICS '14, (New York, NY, USA), ACM, 2014.

[22] *Reducing Latency in an SRAM/DRAM Cache Hierarchy via a Novel Tag-Cache Architecture*, DAC '14, (New York, NY, USA), ACM, 2014.

[23] "CACTI 5.1." Accessed Date: 11/01/2014; http://www.hpl.hp.com/techreports/2008/HPL200820.html.

[24] *32-core CMP with Multi-sliced L2: 2 and 4 Cores Sharing a L2 Slice*, SBAC-PAD '06, (Washington, DC, USA), IEEE Computer Society, 2006.

[25] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," *SIGARCH Comput. Archit. News*, vol. 30, pp. 211–222, Oct. 2002.

[26] "Haswell (microarchitecture)," 2013. accessed date: 05/22/2015 - http://en.wikipedia.org/wiki/Haswell_(microarchitecture).

[27] P. M. Ortego and P. Sack, "Sesc: Superescalar simulator," tech. rep., University of Illinois, 2004.

[28] Woo, Steven Cameron and Ohara, Moriyoshi and Torrie, Evan and Singh, Jaswinder Pal and Gupta, Anoop, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *Proceedings of the 22Nd Annual International Symposium on Computer Architecture*, ISCA '95, (New York, NY, USA), pp. 24–36, ACM, 1995.

[29] Bardine, A. and Comparetti, M. and Foglia, P. and Prete, C.A., "Evaluation of Leakage Reduction Alternatives for Deep Submicron Dynamic Nonuniform Cache Architecture Caches," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 22, pp. 185–190, Jan 2014.

[30] J. Abella, A. González, X. Vera, and M. F. P. O'Boyle, "Iatac: A smart predictor to turn-off l2 cache lines," *ACM Trans. Archit. Code Optim.*, vol. 2, pp. 55–77, Mar. 2005.

[31] "Reconfigurable Caches for Adaptive High-Performance and Embedded Processors." Accessed Date: 10/15/2015; http://serdis.dis.ulpgc.es/dbenitez/index_archivos/Page318.htm.

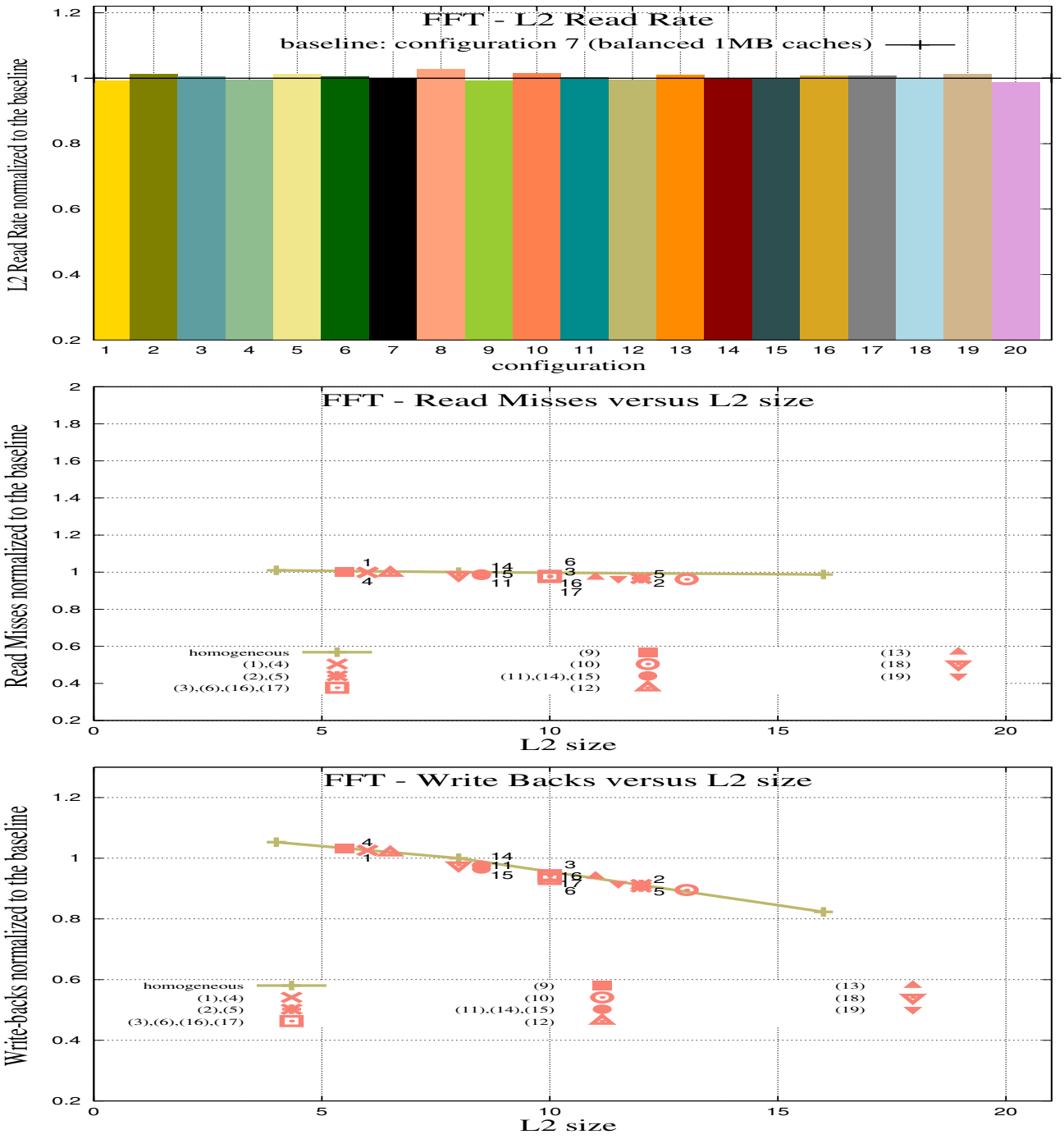

Fig. 14: top to bottom, (a) miss rate, (b) number of misses, and (c) writebacks versus L2 size for FFT

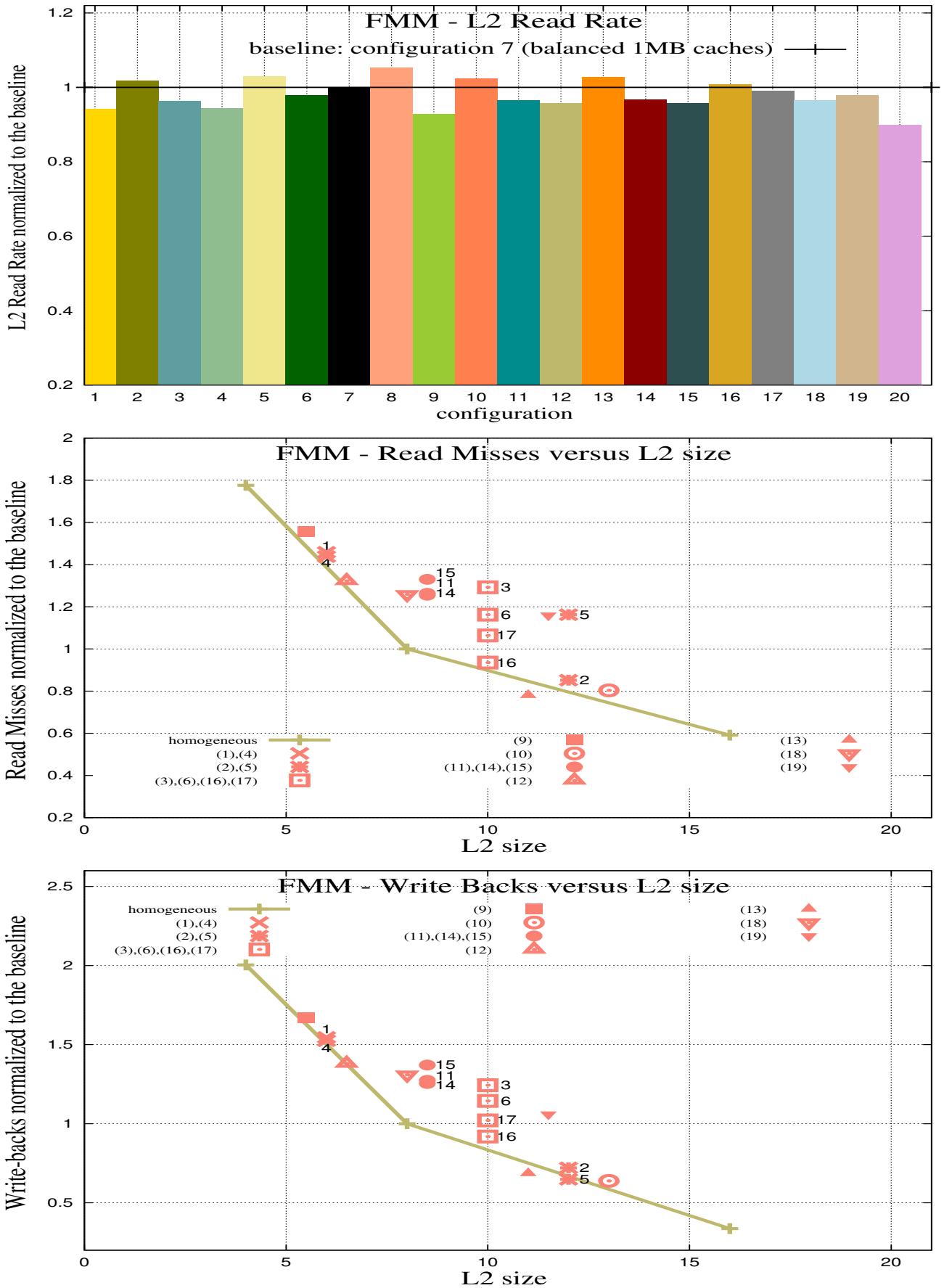

Fig. 15: top to bottom, (a) miss rate, (b) number of misses, and (c) writebacks versus L2 size for FMM

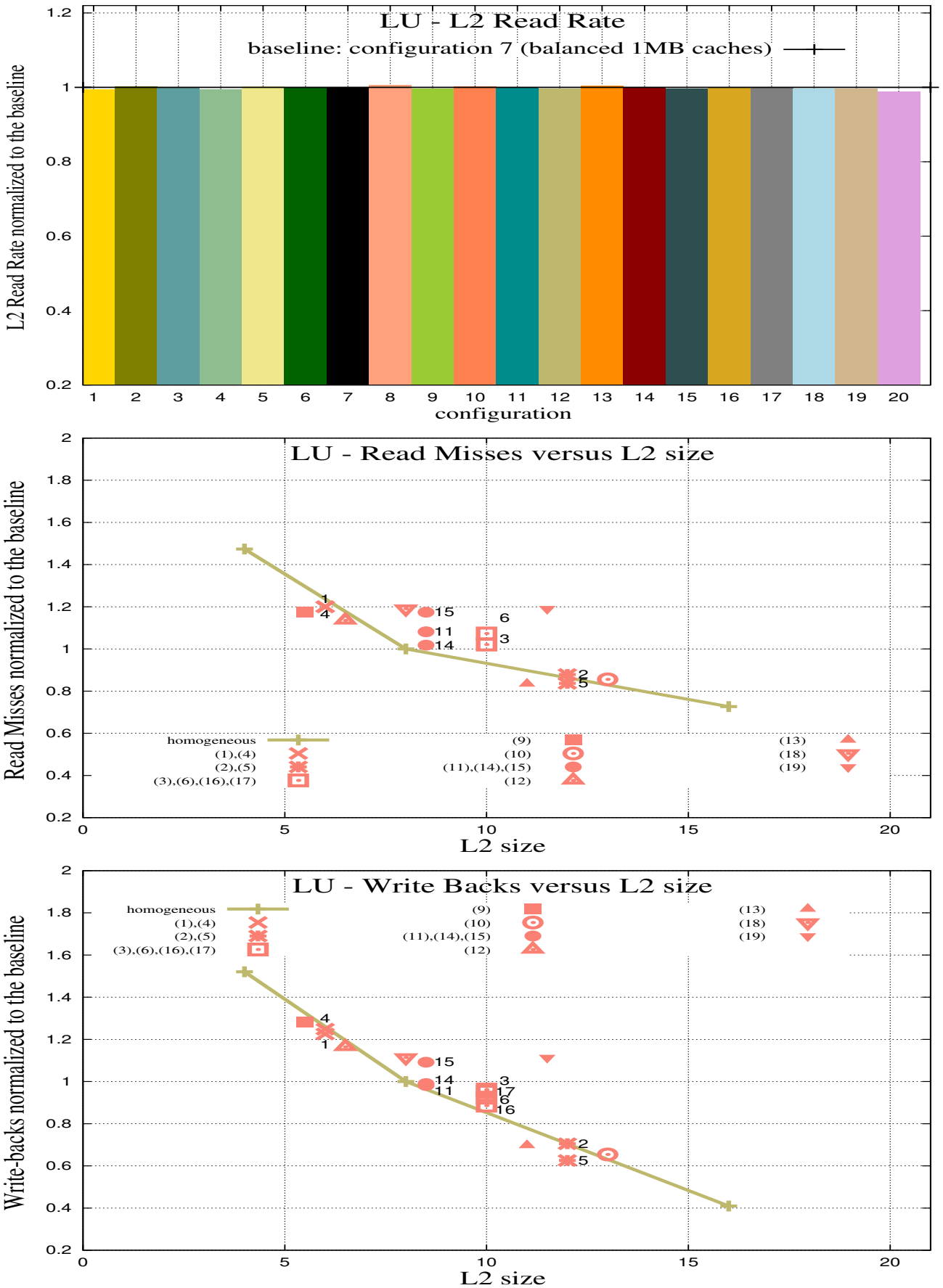

Fig. 16: top to bottom, (a) miss rate, (b) number of misses, and (c) writebacks versus L2 size for LU

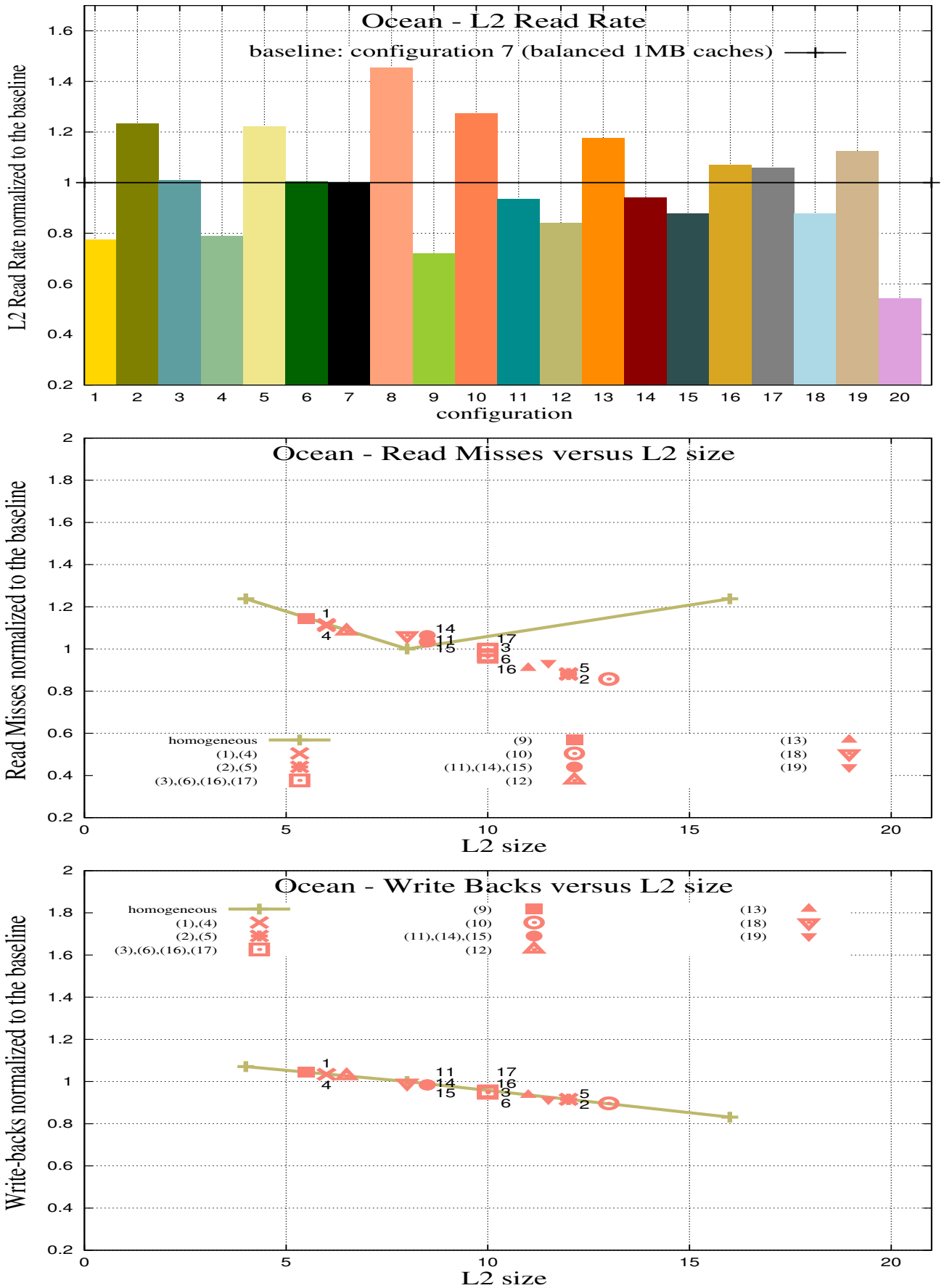

Fig. 17: top to bottom, (a) miss rate, (b) number of misses, and (c) writebacks versus L2 size for Ocean

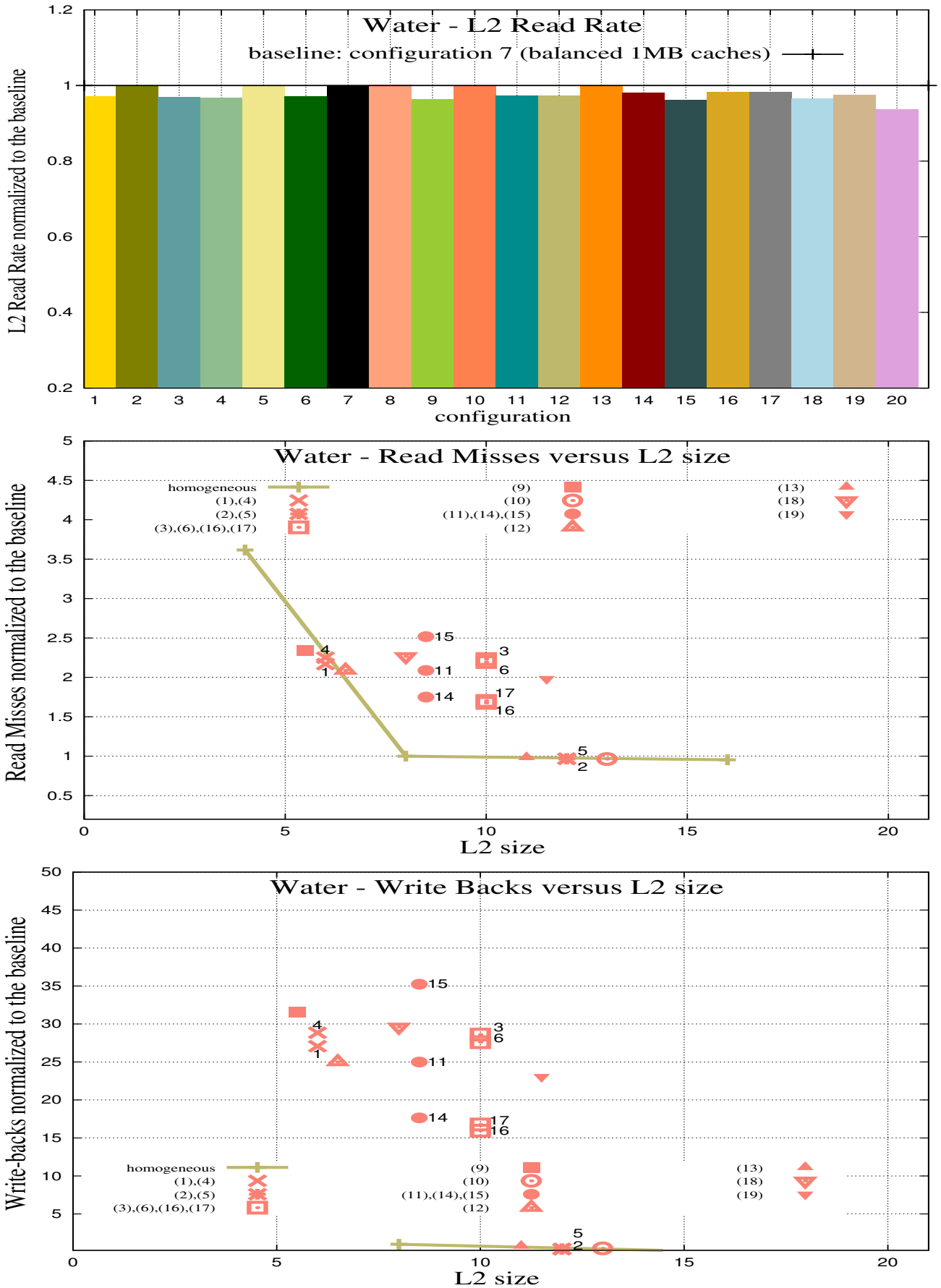

Fig. 18: top to bottom, (a) miss rate, (b) number of misses, and (c) writebacks versus L2 size for Water