Performance and Energy Efficient Cache System Design : Simultaneous Execution of Multiple Applications on Heterogeneous Cores

Venkateswaran Nagarajan*, Kartik Lakshminarasimhan‡, Akash Sridhar‡, Prashanth Thinakaran‡, Rajagopal Hariharan‡,
Vinesh Srinivasan‡, Ram Srivatsa Kannan‡
Aswin Sridharan†
*Director, Waran Research Foundation
‡WARFT Research Trainee

***Abstract*—Future generation supercomputing clusters are endeavouring to achieve exascale performance without compromise on energy efficiency. Executing multiple applications simultaneously without space time sharing in a heterogeneous multi core environment brings out the utmost parallelism that exists within the applications. This helps to attain peak performance and also paves way for improved resource utilization. This necessitates the need for an efficient and locality aware cache replacement scheme to cater to the magnanimous data needs of underlying functional units in case of a cache miss. Reduced cache miss improves resource utilization and reduces data movement across the core which in turn contributes to a high performance to power ratio. This paper proposes a novel application aware cache replacement policy in which data blocks are assigned weights based on a set of application and data statuses. Our proposed heuristics have shown an 8-11% improvement in cache hit when compared against conventional cache replacement heuristics.**

***Keywords*-Multiple Application Execution, Cache System Design, Replacement Policy**

I. Introduction

Even though computational speed (flops/second) is the prime metric, cost due to power consumption is a major worry, demanding the users to see performance to power ratio (performance/watt). Cache system design is the prime metric deciding both the performance and power. At a multiprocessor environment, cache miss degrades the performance as the cache miss penalty scales by a very higher factor across a shared memory system when compared to general purpose processors. This necessitates the need for efficient cache replacement policies supporting the simultaneous execution of multiple applications. In order to execute applications simultaneous, major architectural changes as shown below are necessary. Current cache replacement strategies do not support such parallelism among applications.

1) Compilation Acceerator on Silicon (CAS): When the number of cores in a heterogeneous multi-core environment is increased in thousands, parallel issue of hundreds of instructions and their efficient scheduling becomes a bottleneck. This is overcome by using Compilation Accelerator on Silicon, a hierarchical hardware based compiler cum scheduler [1].

2) Algorithm Level Functional Unit (ALFU): Usage of judicious combination of ALFUs and scalars give energy and performance efficient core architectures [2].

3) Algorithm Level Instruction Set Architecture (ALISA): This is a superset of CISC and VLIW instruction. A single ALISA instruction is equivalent to several ALU, vector and VLIW instructions [2].

4) Simultaneous Execution of Multiple Applications (SMAPP): It is well established that simultaneous execution of multiple applications non space time sharing improves resource utilization [2]. The architecture of ALFUs are evolved in such a manner to support simultaneous multiple application execution [2].

While research has been carried out in the above directions with regard to cache system design, there has not been enough stress on replacement policies and the associated performance-energy relationship in the context of simultaneous execution of multiple application non space time sharing. The Least Recently Used (LRU) cache replacement policy [6] allocates resources based on current need but does not perform replacement based on cache utility. This leads to substantial reduction in performance in heterogeneous multi-core environment. While various applications compete for the cache lines in shared memory architecture, not only application based cache partitioning are needed, but also application based cache replacement policies are required for better cache performance. While framing the replacement policies, a complex design space exploration involving the application parameters is necessary in order to improve the cache hit/miss ratio. Small changes to replacement policies such as, adding tag number to cache blocks [20] to determine the miss causing application are not enough in improving performance. This paper proposes replacement polices covering the entire application characteristics. An energy efficient power model supporting the replacement policies with low cost hardware circuitry is also emphasized making the cache system design better over the existing cache architecture.

The applications can be classified depending on the cache access patterns. Some applications may frequently access the L1 cache, requiring less data while accessing each time. Some applications need huge chunk of data from cache with less frequent access. The existing replacement policies such as Not Recently Used [21], with re-reference interval prediction

978-1-4799-1331-2/13/$31.00 ©2013 IEEE

mechanisms, predicts the future interval in which the data packet is likely to be accessed, takes only the application characteristics consideration. Depending on the applications characteristics, bits are set to each data packet. However, this classification and the above replacement methodology fail in the following situations. There can be situations where a common data packet may be the inputs to various applications. A particular application requiring the data may be a LRU application but the common data packet is essentially MRU data packet as it is involved in various applications. This paper proposes replacement policies which covers a wide range of application characteristics as well as data packet characteristics.

The focus of this paper is on investigating the replacement policies and energy models to suit simultaneous execution of multiple application non space time sharing. Section 2 proposes the cache organization and its associated heuristics while section 3 describes cache controller and its associated architecture. Section 4 deals with the associated energy model which is followed by the results and analysis section

II. Cache organization and replacement strategies

All data which is being stored in the dynamic memory are in the form of data packets, and these data packets are clubbed together to be called a block of size equal to a single cache line and the group of lines are termed cache set. For mapping the data packets from DRAM to the cache employing a combination of mapping strategies based on the cache size at the respective levels as in the conventional system cache design. For mapping the data packets from the L3 cache to the L2/L1 cache, the ”weighted data packet strategy” is being proposed.

It is noticed from figure 1 that the data packets associated with multiple applications co-exists together on a single cache line. Thus the application statistics is given a serious consideration besides the instruction/data, to improve the cache performance in simultaneous execution environment, Further It is replacement policies based on the collective statistics that will influence cache performance which is the focus of the paper. On the other hand the conventional mapping strategies are readily used even in this case, as these mapping strategies are mainly based on cache sizes at respective levels and hence a combination of conventional mapping strategies are used based on the respective cache size.

Considering the block of data to be comprising of data packets, the statistics of the data are analyzed and correspondingly a weight is assigned to them namely critical, moderately critical or non-critical. When these data packets are being mapped, one parameter which is of prime importance is the hit to miss ratio.

Under each of these mapping strategies data packets from DRAM to cache are mapped in a predictive manner by analyzing the current instructions in the queue. This data packet mapping is performed by setting threshold in terms of in terms of number of time stamps (of the WIMAC simulator) by which the concerned instructions will get executed. A suitable value for this threshold is fixed to avoid the replacement of data packets mapped very recently in a predictive manner. However during the cold start, the threshold could be kept higher to avoid compulsory misses.

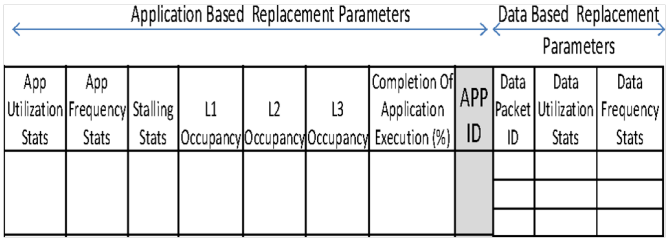

Fig. 2. Application and Data Statistics Table

There is an immediate need for application aware replacement strategy. In conventional replacement schemes are tracking only the data statistics but not application statistics. But in case of simultaneous multiple application execution both the application and data statistics must be taken in to consideration and hence replacement strategies should be based on collective decision of data and application statuses.

```
For each block within a set
    For each packet within a block
        Assigning weights to data packets with respect to following parameters
        {
        Read the number of clock cycles after which the data is required for execution
        Read the Application priority of the particular data packet
        Read the Spatial and temporal locality statistics (LRU, MFU)
        Read the Remaining amount of the data that is pending to get executed
        Read the stalling statistics of the related data packets
        Read the data dependencies field i.e. based on the number of data dependent on the
        -particular data
        Maintain the ratio of data packets of a application in L1:L2:L3 of a particular application and hence
        ensuring the even mapping of the data packets all the applications
        }
    Calculating weights of the data packet:
    Based on the weights of the particular data, If a data value goes above/below a -
    threshold value then the data packet is termed as High-weight/low-weight data

    Assigning weights to the blocks=(Number of high weight blocks)-(Number of low weight blocks)

    End
End
```

Fig. 3. Cache replacement heuristics algorithm

The statistics related to the data and application are recorded, where in a application and its associated data, where the various parametric values as listed in table shown in figure 2 are the inputs for replacement strategy. Thus the algorithm calculates the weights of the associated data packets and decides whether the packet could be dropped or not based on its associated weights.

III. Cache controller :Implementing replacement policies

Resorting to simultaneous multiple application execution across heterogeneous multi cores, replacement strategies play a vital role to achieve overall cache hit and energy efficiency. These replacement policies are discussed in detail bringing

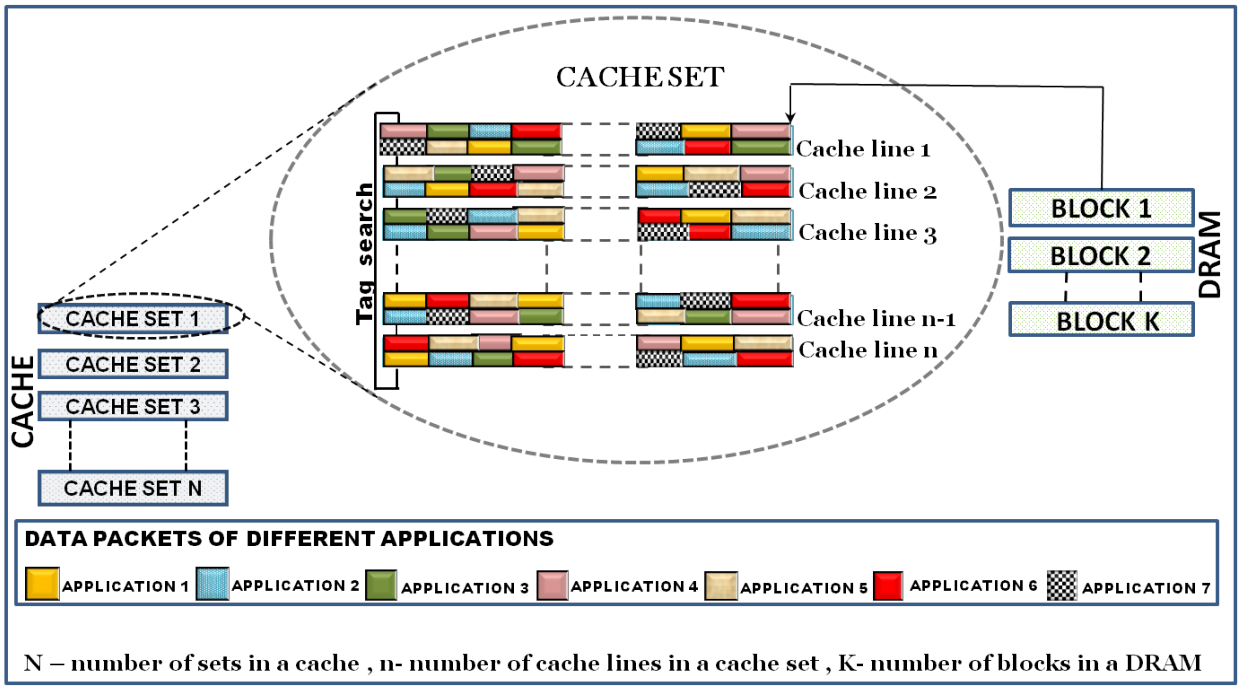

Fig. 1. Proposed Cache organisation for multiple applications

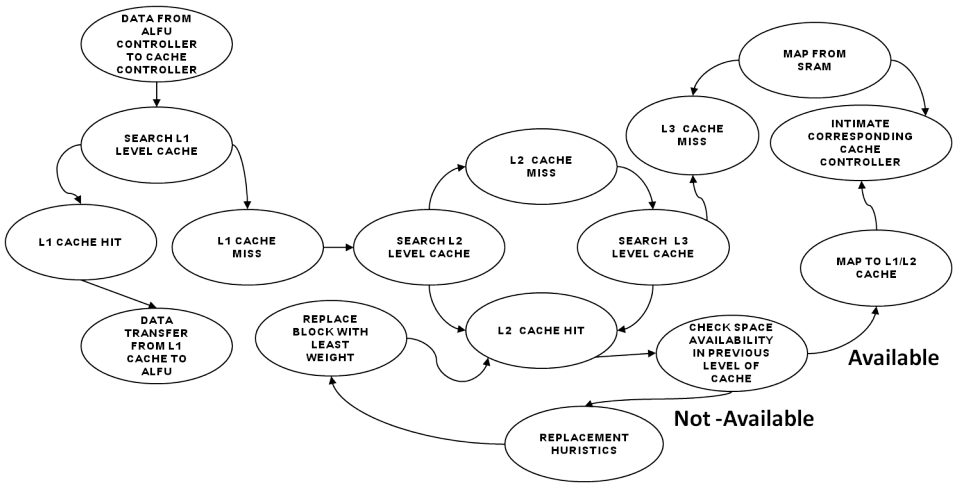

Fig. 4. Proposed Cache Controller Finite State Machine

into effect of both application characteristics and associated instruction/data sets. On the other hand, with regard to cache organization and mapping, conventional techniques are used. Cache mapping is primarily decided by cache size and application characteristics will not play a vital role. In the same way, cache mapping strategies are dependent on cache size and the levels. Either a single strategy or a combination of direct, associative or set associative strategies is used.

A. Cache Controller: Finite State Machine

The cache controllers are present at each levels of cache. The cache controller synchronizes the operations in a particular cache level. The working of the cache controller is similar in all the 3 levels of cache. When there is a miss at a lower level of cache, controller at the lower level sends a control signal to the controller at next level in order to trigger a search operation at that level. The cache controller involves different execution paths for different replacement strategies. Hence a Finite State Machine needs to be designed such that various state transitions depend on the replacement policies. The different replacement policies involve different statistics and almost similar operations. Hence, the different policies can be modeled as a single state. The design of the cache controller revolves around designing a finite state machine the hardware implementation of different replacement policies. From the

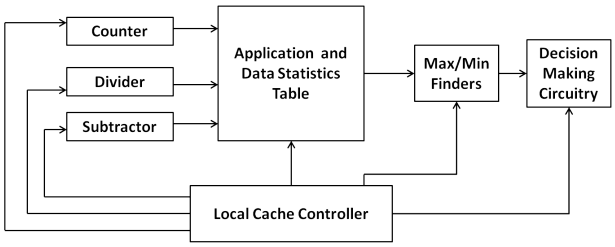

Fig. 5. Hardware implementation of cache replacement strategies

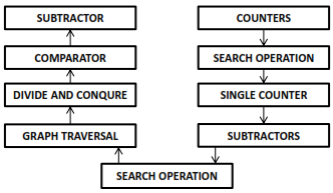

Fig. 6. Energy path taken during cache replacement

above section it is evident that most of the operations involve the replacement policies across all cache levels are almost similar leading to a simpler cache controller design. However, for parallel access of different cache levels, either independent cache controller or a single controller can be shared across all levels.

The replacement strategies for simultaneous multiple application execution are simulated using Verilog. The utilization statistics of datapackets and applications are obtained from their statistics table shown in figure 2 present in section 2.Using these statistics as input,the cache controller shown in figure 4 decides either datapackets or applications are LRU, MFU,etc.

IV. Multiple Application Execution based power model for the cache system

A unified energy model that captures the execution dynamics of the cache system is essential for the design of the cache system. Conventional cache system design methodologies make use of cache system simulation tools that are integrated with standard energy models. This method of energy estimation holds good for simple mapping and replacement policies. For the replacement policies that have been presented in the previous section for multiple application execution, there is need for a more rigorous method for estimation of energy by taking into consideration the energy consumed by the controllers that implement the working of the replacement policies. This section presents an energy model based on replacement policies that become essential when simultaneous execution of multiple applications without space-time sharing is in perspective.

The model can be viewed at two levels: the replacement policy execution level and the read/write level. This level encompasses all the operations that are associated with the working of the heuristics based replacement policy. The energy paths taken by the various states in the replacement policies are first listed as shown in figure 9. The operations involved such as search, counting, comparator operations, subtraction etc. are individually traced during execution and based on the number of times the respective units are utilized, the energy associated with the input transitions are computed. Conventional replacement policies do not involve many operations as the proposed approach and hence an execution based power model is often not given as much importance.

The read/write level of the model makes use of the conventional cache models to estimate the energy consumption of the cache. Based on technology and size, the energy associated with wordline, read/write and leakage power of every cache level. This is integrated with the execution level model and the energy associated with the entire cache system in the replacement scenario is calculated using this model.

V. Results and Analysis

To illustrate the effeciency of our proposed cache paradigm we use an in-house built simulator, Warft India MAny Core (WIMAC) [15] to capture and compare the cache dynamics of various classes of applications. The WIMAC simulator is made cycle accurate to mimic the behaviour of the underlying architecture with great precision. An integrated optimized engine helps to prune the architectural design space to select the most suitable architectural configuration to meet the power and performance requirements of the application. The architectures of Algorithm Level Functional Unit (ALFU) [2] and Compiler Acceleration on Silicon (CAS) [1] are part of this simulation framework.

Number of cores	9
L3 cache size	2MB
L2 cache size	256KB
L1 cache size	16KB
Number of L1 cache lines	448
Number of L2 cache lines	448
Number of L3 cache lines	3800
L1 associativity	4
L2 associativity	4
Number of packets	4

Fig. 7. Sample Architecture table

Figure 7 shows the architectural specifications we have taken for our simulations. We have used SPEC-INT equivalent benchmarks to analyse the effectiveness of the proposed heuristics by scaling up the number of applications in every simulation run. Executing multiple applications simultaneously without space time sharing increases the independent instruction count, thereby improving resource utilization which indirectly contributes to better cache hit rate. From the graphs,

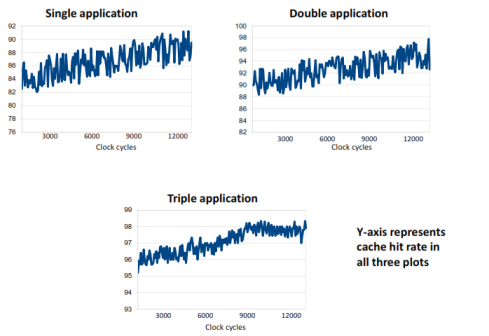

Fig. 8. Cache Hit Ratio varying across the entire simulation

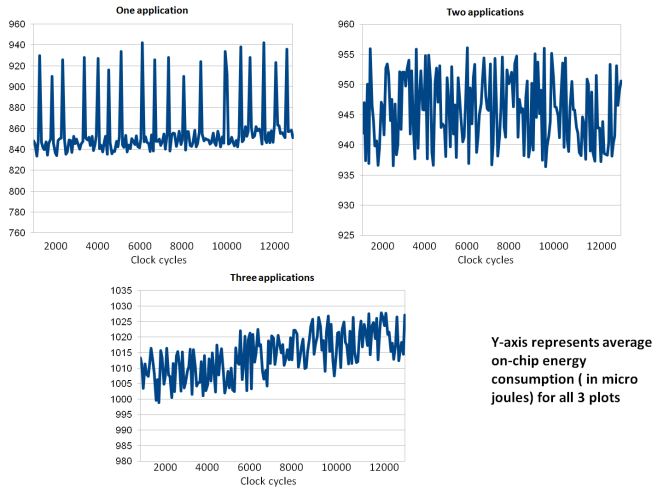

Fig. 9. Single and Multiple power model results for a specific sample of simulation after cold start

it is quite evident that our heuristics has aided in improving the cache hit rate by 8-11% for simultaneous multiple application execution.

For the purpose of energy estimation, an execution sample of about 200 clock cycles much after the cold start period is considered. As discussed earlier, the cache hit ratio for a single application is about 70%. The energy consumption is inherently lesser for this case and shoots up occasionally as can be seen in Figure 9. This shoot up in energy can be attributed to the employment of the replacement policies. In the cases where two and three applications are simultaneously executed together, the overall energy consumption increases due to better cache hit ratio which is a consequence of the proposed mapping and replacement policies. However the rise in energy consumption is not very sharp and is significantly kept constrained by the mapping and replacement policies which reduce the number of misses.

VI. Conclusion

The paper focusses on cache organization taking in to consideration simultaneous multiple application execution without space time sharing. The effectiveness of the replacement strategies related to application as well as data statistics is stressed upon. The cache controller architecture and its interaction is distinctively brought out to accurately predict the execution delay in incorporating the heuristics and the associated cache execution energy model is proposed to accurately capture the energy spent on cache execution paths. Thus, the paper comprehensively captures the cache system paradigm catering to the multiple application execution with accurate execution energy model.

References

[1] N.Venkateswaran et al *Complation Accelerator on Silicon* published in the proceedings of IEEE International Symposium on VLSI 2012

[2] N.Venkateswaran et. al *SCOC IP cores for custom built supercomputing nodes* published in the proceedings of IEEE International Symposium on VLSI 2012

[3] Keshavan Varadarajan et. al *Molecular Caches: A caching structure for dynamic creation of application-specific Heterogeneous cache regions* , In Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 39), 2006

[4] Kim, Yoongu, et al. *ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers* High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on. IEEE, 2010.

[5] Chang Joo Lee et al *Prefetch-Aware Memory Controllers* , IEEE Transactions on computers, VOL. 60, NO. 10.(October 2011)

[6] Aamer Jaleel et al *CRUISE: cache replacement and utility-aware scheduling* , In Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVII).

[7] Phadke, Sujay, and Satish Narayanasamy *MLP aware heterogeneous memory system*, Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011. IEEE, 2011.

[8] R. Kumar et al*Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance* , SIGARCH Comput. Archit. News 32, 2 (March 2004)

[9] Lee, Chang Joo, et al. *Improving memory bank-level parallelism in the presence of prefetching*, Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2009.

[10] Ipek, Engin, et al. *Self-optimizing memory controllers: A reinforcement learning approach*, Computer Architecture, 2008. ISCA'08. 35th International Symposium on. IEEE, 2008.

[11] Subramanian, Lavanya, et al. *MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems*, Proceedings of the 19th International Symposium on High Performance Computer Architecture.

[12] Ausavarungnirun, Rachata, et al. *Staged memory scheduling: Achieving high performance and scalability in heterogeneous systems*, Proceedings of the 39th International Symposium on Computer Architecture. IEEE Press, 2012.

[13] Das, Reetuparna, et al. *Application-aware prioritization mechanisms for on-chip networks*, Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on. IEEE, 2009.

[14] Zhuravlev, Sergey, Sergey Blagodurov, and Alexandra Fedorova.*Addressing shared resource contention in multicore processors via scheduling*, ACM SIGARCH Computer Architecture News. Vol. 38. No. 1. ACM, 2010.

[15] Warft India MAny Core (WIMAC) simulator *http://www.warftindia.org/joomla*

[16] Venkatraman Govindaraju et al *Dynamically Specialized Datapaths for Energy Efficient Computing*Proceedings of the 17th International Symposium on High-Performance Computer Architecture (HPCA), February 2011

[17] Sanjeev Kumar et al., *Carbon: Architectural Support for Fine-Grained Parallelism on Chip Multiprocessors*In the Proceedings of IEEE/ACM International Symposium on Computer Architecture (ISCA), San Diego, California, June 2007

[18] Xiao Zhang, Sandhya Dwarkadas, and Kai Shen *Towards practical page coloring-based multicore cache management*, In Proceedings of the 4th ACM European conference on Computer systems (EuroSys '09). ACM, New York, NY, USA, 89-102. DOI=10.1145/1519065.1519076

[19] Mohammad Abdullah Al Faruque et al., *ADAM: run-time agent-based distributed application mapping for on-chip communication*, In Proceedings of the 45th annual Design Automation Conference (DAC '08). ACM, New York, NY, USA, 760-765

[20] Qureshi, Moinuddin K., and Yale N. Patt. *Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches*, Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 2006

[21] Jaleel, Aamer, et al., *High performance cache replacement using re-reference interval prediction (RRIP)*, ACM SIGARCH Computer Architecture News. Vol. 38. No. 3. ACM, 2010