International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August 2019

Evaluating Resource Centric Behavior of Workloads and Performance Analysis in CMPs due to Shared Resources

Preeti Jain, Sunil K Surve

***Abstract*—Multicore systems are achieving new dimensions to meet the pace between processor technology and high processing demands. These chip multiprocessors (CMPs) share some on- chip and off-chip resources to achieve higher performance. With the increase in core integration, the performance of workloads is dependent on the allocation of resources. Though the CMPs elevate performance, but the challenge imposed due to subtle interactions of several applications contending for resources leads to performance degradation by several magnitudes.**

In this direction the work performs a two-fold evaluation of application programs. The applications running on CMP cores depict distinct behavior towards consumption of shared resources. It characterizes the resource centric nature of SPEC CPU2006 benchmarks based on their resource consumption behavior. Secondly the work aims to evaluate the effect of inter-core interference on the performance of application programs based on the characterization obtained and the potential contention caused on performance due to corunners. Lastly we place significant remarks on the impact on performance due to resource sharing and its implication on resource contention.

***Index Terms*—multicore, shared resource, contention, characterization, inter-core interference.**

I. INTRODUCTION

The multicore architecture takes a leap to bridge the gap between uniprocessors and high computing demands. CMP's offer high performance with reduced power consumption. These multicore chips share certain resources like last level caches (LLC), main memory, bus bandwidth, prefetchers, memory controllers etc. Resource sharing eliminates under-utilization of resources facilitating their optimal utilization. The proliferation of cores on these systems has introduced new challenges in managing the shared resources. Also, parallel access to these resources cause resource contention and leads to variability in performance showing negative impact due to degradation [1] [2] [3] [4] [5] [6]. Thus it is analyzed that performance of multicore systems highly depends on how the resources are utilized among application programs.

Revised Manuscript Received on August 30, 2019.

* Correspondence Author

Preeti Jain*, Fr Conceicao Rodrigues College of Engineering Department of Electronics Engineering, Bandra, Mumbai, India.

Dr. Sunil K Surve, Fr Conceicao Rodrigues College of Engineering Department of Computer Engineering, Bandra, Mumbai, India.

Application programs running on these cores may be dependent on different resources for their growth in performance. Knowledge on the resource centric nature of these workloads would assist in deciding optimal corunners and thereby aim to mitigate contention introduced due to similar resource hungry corunners.

A significant amount of work has been done in the past on characterizing the application programs. Existing research, characterize workloads based on several aspects such as animal heuristic based [7] characterization, page coloring schemes [8][9], utility based classification [10], based on impact on corunner [11] etc.

The present work performs a two-fold evaluation wherein we first characterize SPEC CPU2006 benchmarks [12] based on their resource consumption characteristics. Extensive simulation is carried out using the sniper simulator [13] over various resource sizes to investigate the dependency of workloads on various resources. The work considers LLC and main memory bandwidth consumption behavior and based on it, applications are classified as herbivores, carnivores, omnivores and amphibians [14]. The outcomes present a clear insight to a workload's sensitivity towards various resource allocations.

In the second phase of work, we investigate solo and corun performance of these workloads to analyze the impact of corunners. A mix of several application programs from each class mentioned above is corun and its impact on performance is evaluated (in terms of Instructions per cycle (IPC)). The results reveal degradation in performance compared to isolated executions. We illustrate that, degree to what an application is hurt due to its corunner, depends upon type of corunners.

The paper contributes in the following manner,

i. Based on the resource consumption behavior of application programs they are classified into 4 major classes as herbivores, carnivores, omnivores or amphibians.
ii. Characterization of workloads assists in analyzing sensitivity of application programs towards constrained resources.
iii. The solo and paired performance evaluation helps in understanding the impact on performance due to resource sharing.
iv. Inter-core interference can be investigated from the outcomes of the work.

The paper is structured as follows. In Section 2 we discuss related literature.

In section 3 we evaluate workload characterization along with simulation results. Section 4 includes performance analysis of solo and paired runs of workloads in CMPs due to constrained resources. Section 5 concludes the paper.

II. LITERATURE REVIEW

Prior studies have presented various characterization schemes. In [7] the authors propose an animal behavior oriented heuristics for characterization. Applications are segregated into various animal behaviors like categories. Turtles are the workloads which do not make use of last level shared caches, as they being small programs with less need to access memory. Sheep indicate the programs which exhibit high L2 accesses even with low cache ways allotted to them. A rabbit being the programs which show gain in performance with more number of ways allotted to them with devils being memory hungry applications, cleaning previous cache code, which is beneficial to other programs to bring in data of their use.

Some works include classification of benchmarks based on a software *color scheme*[8].The authors propose OS level page coloring to map cache resources between corunners. Statistics of solo run of program with a configuration of 1MB of L2 cache are gathered, compared to 4MB baseline configuration. The slowdowns due to reduced caches are recorded and based on it, applications are classified as *Red, Yellow, Green and Black.* Red category indicates slowdown greater than 20%, slowdown between 5% and 20% as yellow, less than 5% slowdown as green and any program with slowdown besides these slowdowns as black. The experimentation requires running the program simultaneously on 2 cores with 4Mb cache size, collecting the parameters and re-running the program with reduced cache size of 1MB and recording the slowdown. In [9] the cache mapping colors are applied on frequently accessed pages called hot pages for each process. Spatial locality of the page is maintained at a page table, which is scanned for identifying hot pages. The scheme reduces memory page allocation and also reduces overhead of page copying since only hot pages are colored.

Further work on utility based partitioning of cache is carried out in [10] where the authors characterize workloads into High, Low and Saturating Utility programs. High utility are the benchmarks which get advantage due to increase in cache ways and on the contrary low utility are those workloads which are not benefitted due to increased ways. The third category being saturating utility which include the workloads which exhibit enhancement with increase in cache ways upto a certain extent, after which no performance gain is observed.

In [11] the authors characterize applications in two main categories namely heavy sharing (affect performance of corunners) and light sharing (no significant impact on corunners), utilizing very less resources. The work involves estimation of performance due to solo and corun executions.

Another classification is proposed in [15] based on two metrics of which one metric is the IPC gained with full usage of all n ways of L2 cache allotted completely to a program $w_{k\%}$ and wlru the number of ways used by each thread when corun.

In [16] the authors present classification of applications in terms of their cache fitting capacity. They show that cache resource centric applications suffer worst performance degradation due to reduced cache quota as they are cache friendly applications and hence suffer loss in performance as cache allocated reduces which leads to high miss rate.

Using machine learning techniques for characterizing work-loads is also suggested in [17] but they analyze the behavior based on shared cache only.

In a recent study in [18] the authors characterize workloads into different classes depending upon their activity along the memory domain. In its simplest form class N belongs to applications whose working set is small. While cache friendly applications are included in class C. These applications show performance enhancement with increase in cache size. While memory intensive applications are included in class S. An analysis of activities in the memory domain in a hierarchical fashion is performed based on a decision tree to characterize workloads.

In [19] the authors group workloads as contentious and contention sensitive. The term contentious includes applications which impact the performance of its corunner to a large extent, while workloads which are impacted due to high interfering programs are called contention sensitive.

In [20] the authors adopt a *software code* based cache partitioning scheme. It first involves profile generation which gathers code profile using binary instrumentation, next profile analysis which classifies code based on its characteristics and code based page coloring to indicate different code based regions namely hot, pollute and normal. The scheme eliminates the need of special hardware to keep runtime information.

Thus it can be seen that many researchers have proposed various schemes to characterize and classify benchmark programs. In the following section we demonstrate resource centric performance of workloads.

III. METHODOLOGY FOR EVALUATING RESOURCE CENTRIC PERFORMANCE OF WORKLOADS

A. Experimental setup

For the purpose of this study we perform simulation using multicore sniper simulator which functions on interval core model. The baseline processor is Nehalem micro architecture based processor which operates on 2.67 Ghz, 128 entry ROB. Memory organization is structure into a two level hierarchy with 32Kb L1 cache which is organized as private cache and the last level cache (here L2) is shared between cores. The penalty of 175 cycles is imposed towards main memory access upon every cache miss. The cache replacement policy employed is LRU.

B. Methodology of Workload Characterization

For each workload we estimate its behavior towards re- source consumption for different resource size allocations Table 1. Later, subject to the outcomes of the simulation results the applications are characterized into one of the 4different classes.

Cache Size (Kb)	512	1024	2048	4196	8192
DRAM Bandwidth (Gbps)	7.6	15.2	30.4	60.8	121.6

Table 1: Resource allocation for simulation

Depending upon the impact on performance (in terms of IPC) in utilizing limited shared resources viz LLC (in this case L2) and main memory bandwidth, following are the classes into which we characterize the benchmarks.

1) Herbivores:- These workloads respond positively to increasing cache size and show no significant growth in IPC with increasing DRAM (main memory) bandwidth.
2) Carnivores:- In contrast to herbivores, these application programs show enhancement in IPC as DRAM band-width allocation increases rather than raising the amount of cache allocated. This category includes memory bound workloads.
3) Omnivores:- These workloads exhibit growth in performance due to increase in both resources.
4) Amphibians:- The amphibian type of application pro-grams have small working set and hence their performance does not reveal significant variation due to increase in any resource size.

C. Performance Evaluation on Benchmark Programs

Extensive simulation is carried over 20 benchmarks from SPECCPU2006 benchmark suite. We use pin based simulation to conduct memory studies. A binary instrumentation based tool (sim-point) using pin-points [21] is employed to gather traces of representative regions of the workload [22]. Each benchmark program is subdivided into 4 trace files. A single trace file executes 250 million instructions. Weights corresponding to these traces are estimated. Weighted IPC is then calculated using weights and Cycles per instructions (CPI) values. Next we estimate average weighted IPC for individual application. In the present work IPC values are gathered by varying L2 cache and main memory bandwidth. Cache size is varied from 256Kb to 8Mb with a variation in DRAM bus bandwidth from 7.6 Gbps to 121.6 Gbps. Based on the trend in IPC plots benchmark programs are characterized into one of the above mentioned 4 categories.

D. Simulation Results and Discussion

The studies on shared cache and main memory bus band-width utilization reveal the sensitivity of these benchmark programs towards these resources. It can be noted from the figures that extensive simulation has been carried out over various resource allocations. Figure 1 to figure 20 present the impact of resource allocation on the workloads.

As it can be observed from figure 1, 2, 3, 4, and 5 these applications reveal herbivore characteristics. The performance in terms of IPC shows significant increase as cache size increases from 256Kb to 8Mb thus exhibiting a cache-favour behavior. While the IPC values shows a very small increment over a bandwidth rages of 7.6Gbps to 121.6Gbps.

The next set of benchmarks which include figure 6, 7, 8, 9, 10, 11 and 12 align in characteristics. As it is evident from these plots, application programs show insignificant variation in performance due to any increase in cache size, while their plots reveal distinct variation in IPC with varying DRAM bus bandwidth. These workloads show weak temporal locality and are memory bound (intensive). Applications like Libquantum, Lbm, Milc, Leslie, Bwaves, Tonto, and GemsFDTD are characterized as carnivores.

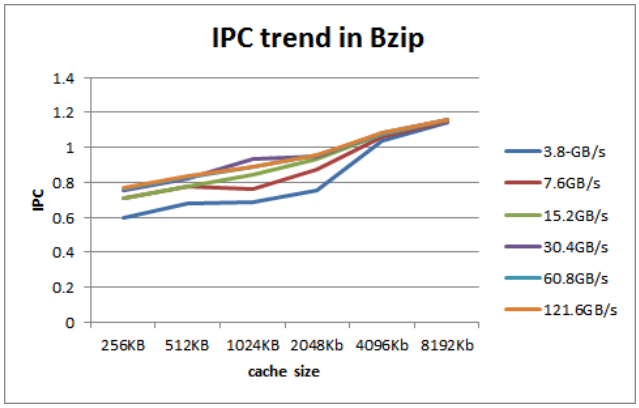

Fig. 1: IPC characteristics of Bzip

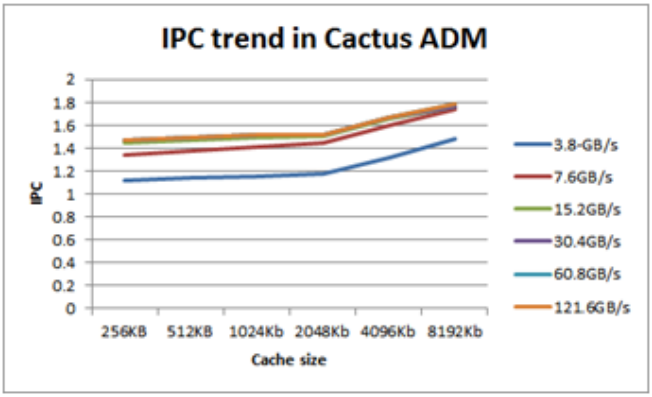

Fig. 2: IPC characteristics of cactusADM

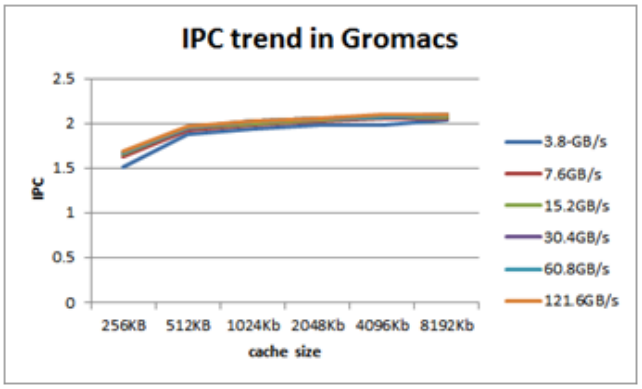

Fig. 3: IPC characteristics of Gromacs

From plots of figure 13, 14, 15, 16, 17 it is observed that the IPC grows incrementally with growing cache size and DRAM bandwidth. This shows the dependency of workloads on both resources. Application programs like Gcc, Soplex, Perlbenchmark, Astar, Zeusmp are characterized as omnivores. Lastly in the plots of figure 18, 19, 20, 21, 22 we observe that variation in IPC is around 0.01 to 0.1 % only. The y-axis shows IPC increment, which clearly shows very little deviation which indicates very less dependency on shared resources. This class includes workloads like Hmmer, Namd, Sjeng, Gobmk and Wrf.

The above stated results clearly give an insight to workloads and their behavior towards varying resource allocation. This analysis facilitates in investigating resource dependency of workloads. In the next section we evaluate the performance of these workloads due to solo and corun executions.

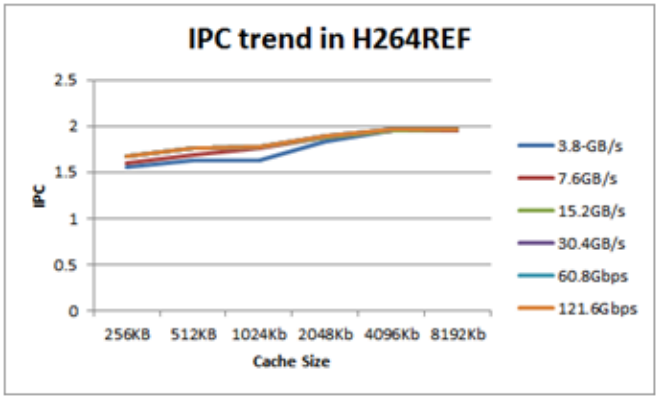

Fig. 4: IPC characteristics of H264ref

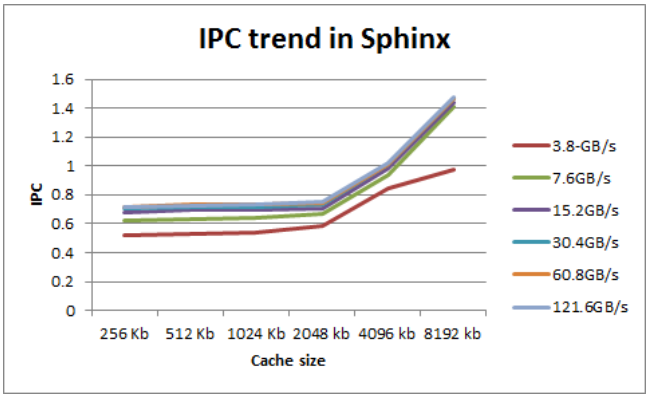

Fig. 5: IPC characteristics of Sphinx

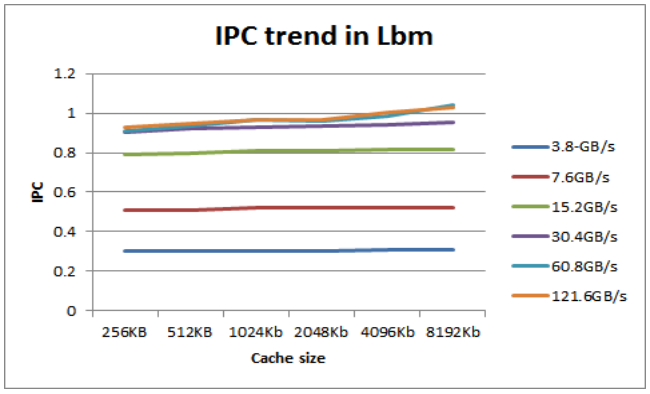

Fig. 6: IPC characteristics of Lbm

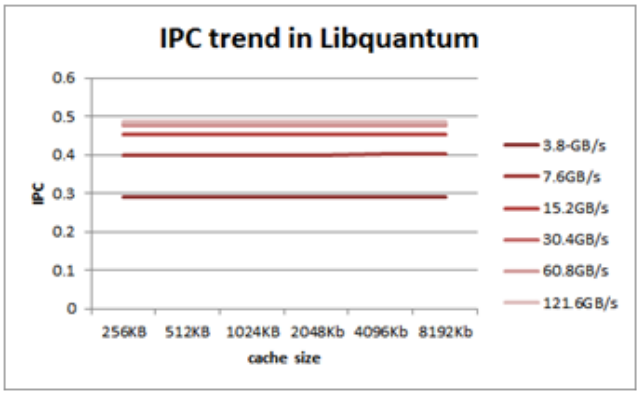

Fig. 7: IPC characteristics of Libquantum

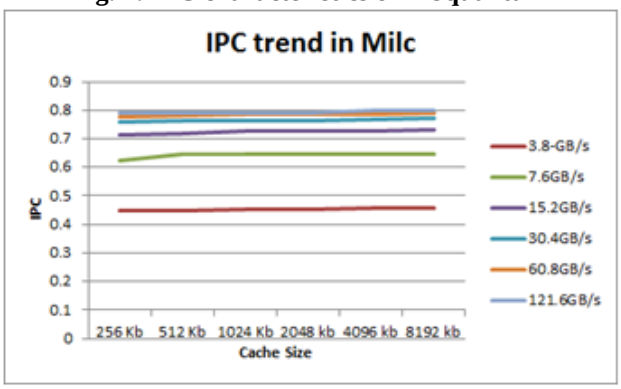

Fig. 8: IPC characteristics of Milc

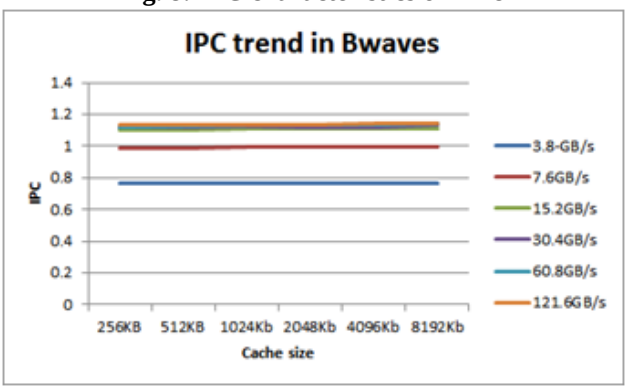

Fig. 9: IPC characteristics of Bwaves

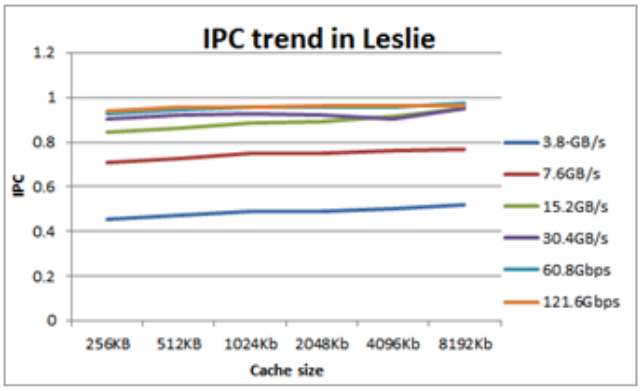

Fig.10:IPC characteristics of Leslie

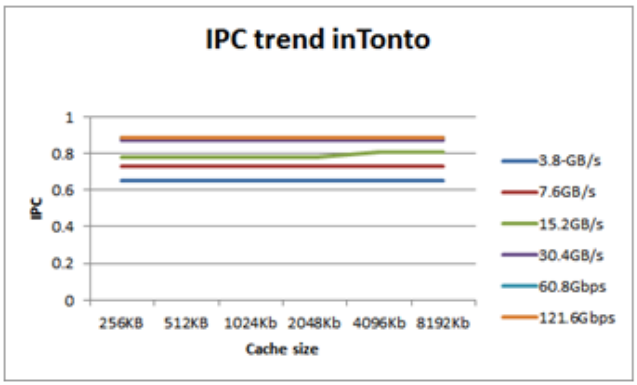

Fig.11:IPC characteristics of Tonto

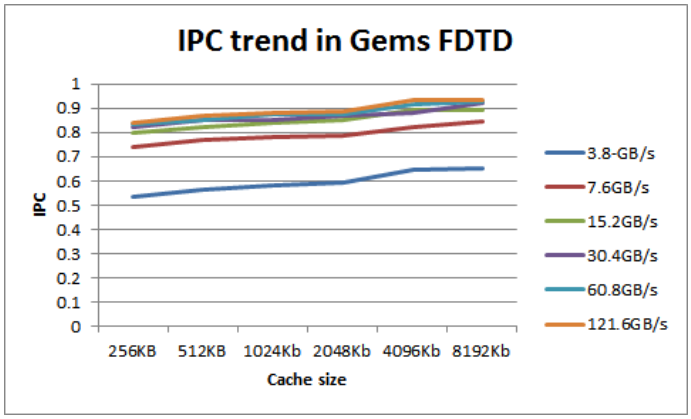

Fig. 12: IPC characteristics of GemsFDTD

IV. ON THE PERFORMANCE EVALUATION IN CMPS

Previous section analyzed the characteristics of SPEC CPU2006 benchmarks for their resource centric behavior. In this section we evaluate the performance of these workloads

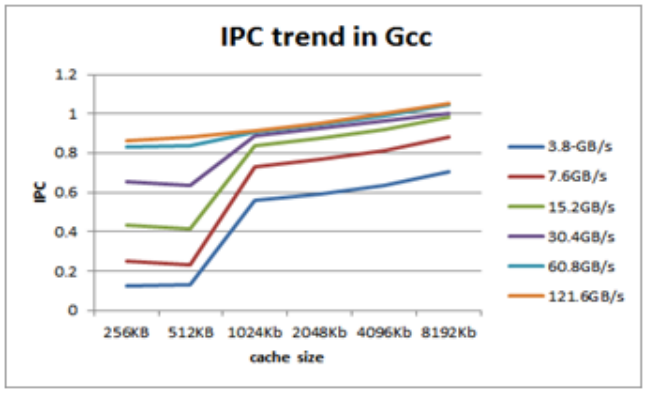

Fig. 13: IPC characteristics of Gcc

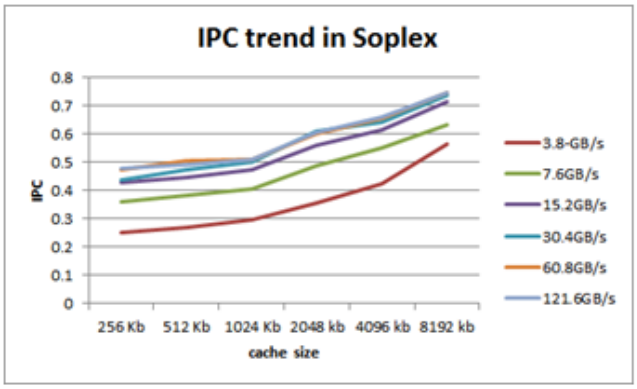

Fig. 14: IPC characteristics of Soplex

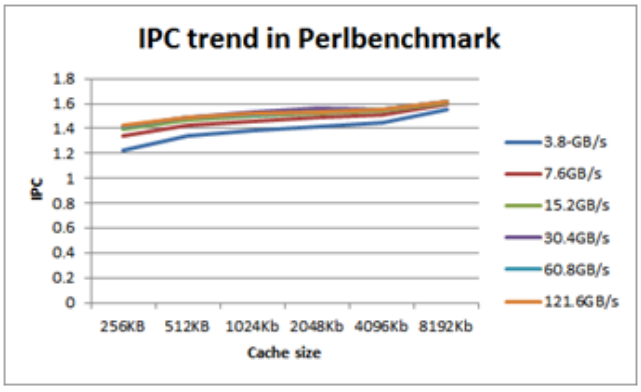

Fig. 15: IPC characteristics of Perlbenchmark

during solo and corun execution. The analysis considers a mix of application workloads from each class stated above to investigate the impact on performance due to corunners. The workloads are initially run isolated on individual cores and IPC values are noted using hardware performance counters. The experimentation is again carried out by executing two workloads on individual cores to estimate inter-core interference.

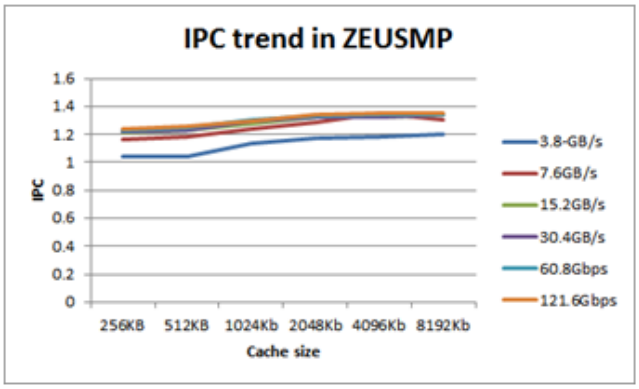

Fig. 16: IPC characteristics of Zeusmp

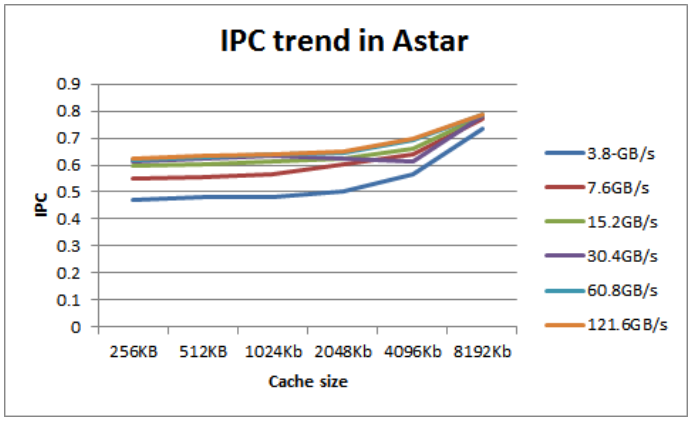

Fig. 17: IPC characteristics of Astar

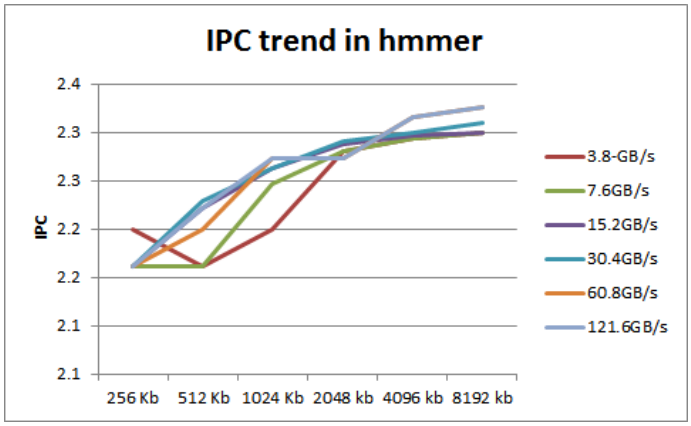

Fig. 18: IPC characteristics of Hmmer

The plots shown represent the comparison in performance due to solo and corun executions. The results shown are at resource size of 512Kb of cache and 7.6Gbps of DRAM bus bandwidth. Plots from figures 23 to figure 26 reveal the results for solo and corun performance. The workload mixes for corunners in each plot consist of one benchmark from each category i.e herbivore, carnivore, omnivore and amphibian.

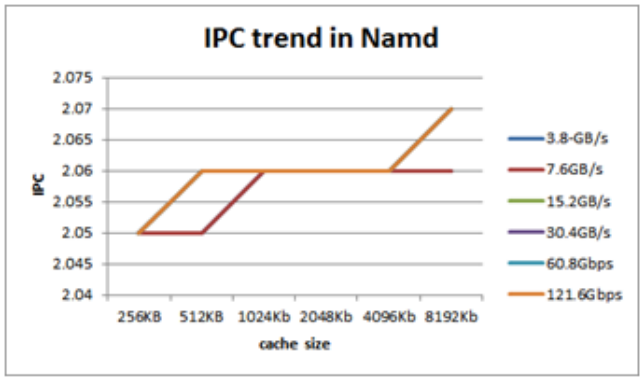

Fig.19:IPC characteristics of Namd

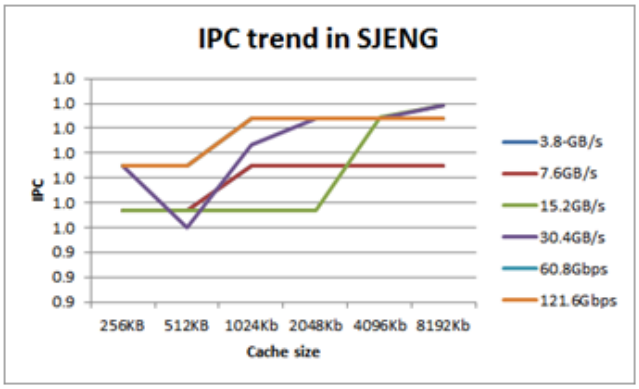

Fig.20:IPC characteristics of Sjeng

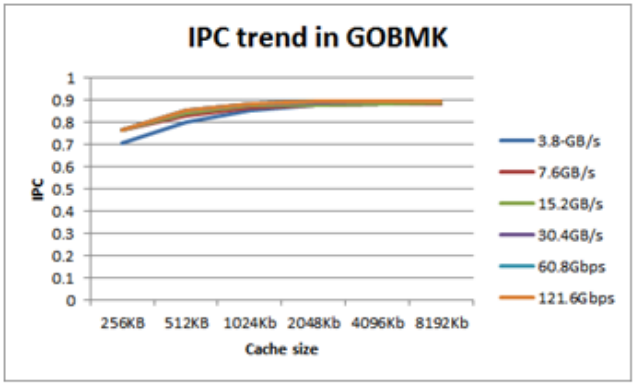

Fig.21:IPC characteristics of Gobmk

From figure 23 it can be observed that there is reduction in IPC values for corun execution as compared to solo execution. Thus we can state that performance degrades due to corunner running on neighboring core as compared to isolated performance. It can be observed that Sphinx (herbivore) suffers maximum degradation when corun with bzip (herbivore) with

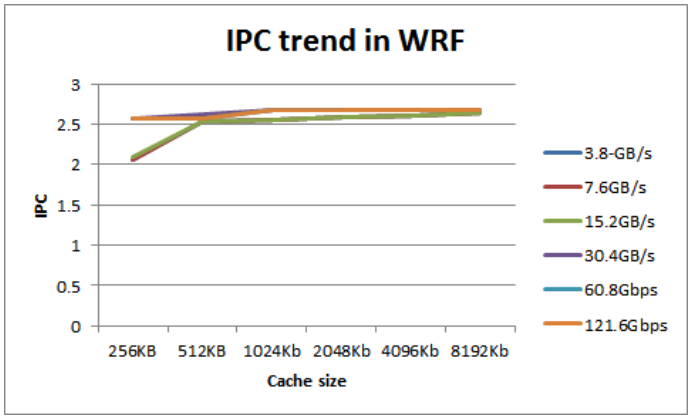

Fig. 22: IPC characteristics of Wrf

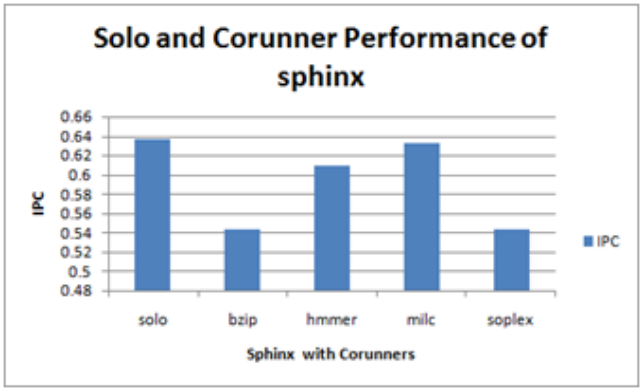

Fig. 23: Normalized IPC of Sphinx w.r.t corunners

nearly 15% degradation in IPC value as they both contend for same resource, as compared to other corunners. Plot shows an IPC degradation of 2% with hmmer (amphibian), 5% with milc(carnivore)and10%withsoplex(omnivore). Next, the

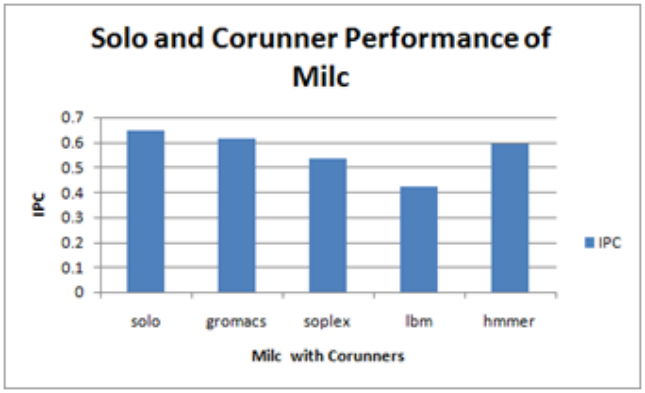

Fig. 24: Normalized IPC of milc w.r.t corunners

performance impact on Milc (carnivore type) application is analyzed in figure24. Here it can be noted that milc (carnivore) faces maximum degradation of nearly 35% in IPC value due to Lbm (carnivore) as they both contend for the similar resources. The performance shows moderate impact on IPC due to omnivore type (16%) with minimum degradation (5%) due to hmmer (amphibian) and Gromacs (herbivore type) (6%).

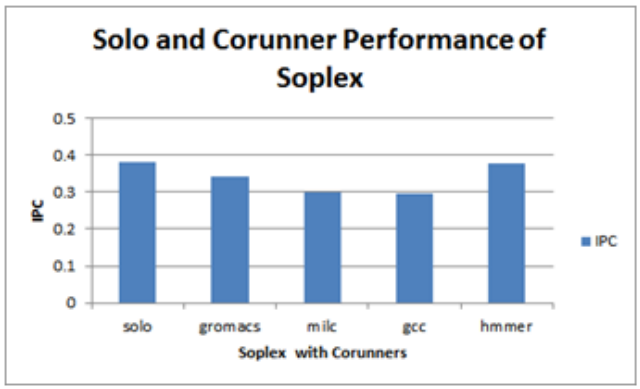

Fig. 25: Normalized IPC of soplex w.r.t corunners

From figure 25 we analyze solo and corun performance of Soplex (omnivore) benchmark. It reveals a maximum performance degradation of 23% with Gcc (omnivore), 10% with Gromacs, 18% with Milc and 9% with Hmmer respectively.

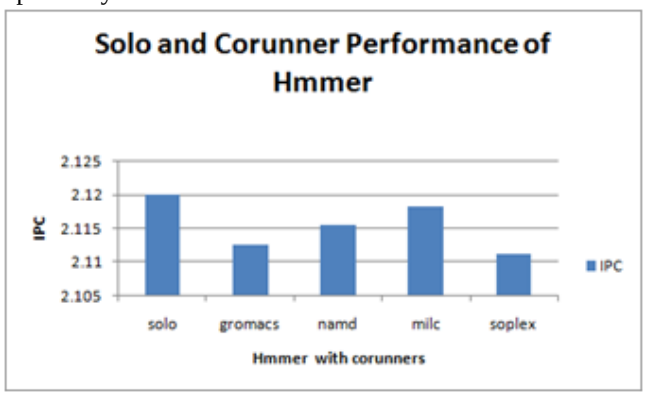

Fig. 26: Normalized IPC of hmmer w.r.t corunners

Figure26 depicts the solo and corun performance for hmmer (Amphibian type) workload. It can be observed that since these workloads have small working sets they scarcely use the shared resources and hence have insignificant impact on performance due to corunners. These plots reveal insignificant variation in IPC values with all classes of corunners.

Thus, from simulation results it can be stated that performance of application program is impacted due to corunner. The performance gain shows reduction when solo and corun performance is compared. Hence it can be stated that shared resource contention leads to performance degradation. On the other hand the degree of degradation depends on resource centric nature of corunner.

V. CONCLUSION

The multicore architecture is tailored to meet the high computing demands. Towards this end to improve resource utilization, there arises a need to share resources. The concurrent access of application programs to these shared resources at times leads to contention and degrades performance.
Characterization facilitates to understand the resource requirements of workloads. Pertaining to this the work firstly aims to characterize the application programs based on their resource consumption characteristics. To analyze this, benchmarks are run solo on several sets of resource sizes to estimate resource consumption. Based on the outcomes application programs are broadly classified into one of the four categories as Herbivores, Carnivores, Omnivores and Amphibians. Next, the study analyzes the performance of application programs in solo and corun executions. The methodology involves estimating the deviation in performance (IPC) due to solo and corun executions. From the results obtained it can be concluded that the performance is impacted negatively due to corunning process as compared to their solo executions. The outcomes of the experimentation facilitate a preliminary measure for coupling processes with diverse resource requirements. Thus, our findings state that the resource contention problem could be tackled by judiciously selecting corunners. Future work directs, implementation of optimal scheduling and resource partitioning schemes to address the problem.

REFERENCES

1. E.Ebrahimi, L.Chang, O.Mutlu, and Y. N.Patt."Techniques for bandwidth efficient prefetching of Linked Data Structures in hybrid prefetching systems".In *Proceedings. International Symposium on High Performance Computer Architecture*, pp.7-17. 2009.
2. D. Xu, C.Wu, and P. Yew. "On mitigating memory bandwidth contentionthroughbandwidthawarescheduling".In*Proceedings.ofthe19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*10, pp. 237.2010.
3. S. Zhuravlev, J. C. Saez, S. Blagodurov, M. Prieto."Survey Of Scheduling Techniques For Addressing Shared Resources In Multi-Core Processors". In Proceedings of The ACM Computing Surveys, Vol. 45, pp 1-31. 2011.
4. J. Feliu, J. Sahuqiullo, S.Petit, J. Duato "Understanding cache hierarchy contention in CMPs to improve job scheduling". In *Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium IPDPS*. 2012.
5. J.Zhao, H. Cui, J. Xue, and X. Feng. "Predicting Cross-Cores Performance Interference on Multicore Processors with Regression Analysis". In *Proceedings.IEEE Transactions on Parallel and Distributed Systems*27 pp. 114. 2015.
6. A.Garcia, J. C. Saez, M Prieto-Matias."Contention-Aware Fair Scheduling for Asymmetric Single-ISA Multicore Systems."In *Proceedings. IEEE Transactions on Computers* ,Volume: 67 , Issue: 12. pg 1703-1709. 2018.
7. Y. Xie and G. H. Loh, "Dynamic Classification of Program Memory Behaviors in CMPs," In *Proceedings of the 2nd Work. Chip Multi Processor Mem. Syst. Interconnects*, pp. 1–9, 2008.
8. J. Lin, Q. Lu, X. Ding, Z. Zhang, X. Zhang, and P. Sadayappan,"Gaining Insights into Multicore Cache Partitioning," no. 1, pp. 367–378, 2008.
9. X. Zhang, S Dwarkadas, K Shen ". Towards Practical Page Coloring based multi-core cache management". In *Proceedings of the EuroSys ACM* pp.89-102,2009.
10. M. K. Qureshi and Y. N. Patt, "Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches," In *Proceedings of the Annu. Int. Symp. Microarchitecture, MICRO*, pp. 423–432, 2006.
11. J.Feliu ,J Sahuqiullo,S Petit J Duato "Addressing Fairness in SMT Multicores with a Progress-Aware Scheduler ", In *Proceedings of the IEEE 29th International Parallel and Distributed Processing Symposium.* 2015.
12. SPEC CPU2006.http://www.spec.org/spec. 2006.
13. T. E. Carlson, "The Sniper User Manual," pp. 91–94, 2013.
14. P.N.Jain and S.K. Surve "Modeling Resource Constrained Solo Applica- tions Using Logistic Growth Model," In *Proceedings. IEEE Conference on Advances in Computing, Communication and Control (ICAC3 17)*. pp 1-9. 2017.
15. M. Moreto, F. J. Cazorla, A. Ramirez, and M. Valero, "Explaining dynamic Cache Partitioning speed ups," In *Proceedings of the IEEE Comput. Archit. Lett.*, vol. 6, no. 1, pp. 4–7, 2007.
16. L. Liu, Y. Li ,C. Ding, H. Yang, and C.Wu. "Rethinking Memory Management in Modern Operating System : Horizontal , Vertical or Random ?", 65, pp. 1921–1935. 2016.
17. R.Bitirgen, E.Ipek, and J.F. Martinez, "Coordinated management of multiple interacting resources in chip multi processors: A machine learning approach". In *Proceedings of the 41st IEEE/ACM International Symposium on Microarchitecture*, pp. 318-329. 2008.
18. K. Nikas, G. Goumas, and N. Koziris, "A Resource-Centric Application Classification Approach," COSH June 2016.
19. L. Tang, J. Mars, and M. Lou Soffa, "Contentiousness vs. sensitivity: Im- proving contention aware runtime systems on multicore architectures," In *Proceedings of the ACM Int. Conf.* , pp. 12–21, 2011.
20. J.Kim, J., Kim, I. and Y.I Eom. "Code based cache partitioning for improving hardware cache performance". In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication ICUIMC* p.1. 2012.
21. M. E. Thomadakis, The architecture of the Nehalem Processor and Nehalem-epsmp platforms, *Technical report*, 2010.
22. H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi. "Pinpointing representative portions of large intel Itanium programs with dynamic instrumentation". In *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, 2004.

AUTHORS PROFILE

Preeti Jain is currently working as an assistant professor at Datta Meghe college of engineering. She pursued her BE and ME in Electronics and is currently pursuing PhD from Fr.CRCE Bandra. Her area of interest is computer architecture and embedded systems. She is currently working on high performance computing systems and has presented papers in the international journal of high performance computing.

Dr. Sunil K Surve is professor and head department of Computer Engineering at Fr CRCE Bandra. He pursued his ME and Ph.D from VJTI, Mumbai. His research interest includes Robotics, Computer architecture, Machine learning and Software Engineering. He has published many papers in reputed journals and also serves as a reviewer for reputed journals.