Reevaluating the overhead of data preparation for asymmetric multicore system on graphics processing

Songwen Pei[1,2], Junge Zhang[1], Linhua Jiang[1], Myoung-Seo Kim[2] and Jean-Luc Gaudiot[2]
[1] Shanghai Key Lab of Modern Optical Systems,
University of Shanghai for Science and Technology,
Shanghai 200093, China
[e-mail: {swpei, lhjiang}@usst.edu.cn]
[2] Parallel Systems and Computer Architecture Lab,University of California
Irvine, CA 92697, USA
[e-mail: {myoungseo.kim, gaudiot}@uci.edu]
*Corresponding author: Songwen Pei, Myoung-Seo Kim

Received May 10, 2016; revised May 24, 2016; accepted June 5, 2016; published July 31, 2016

Abstract

As processor design has been transiting from homogeneous multicore processor to heterogeneous multicore processor, traditional Amdahl's law cannot meet the new challenges for asymmetric multicore system. In order to further investigate the impact factors related to the Overhead of Data Preparation (ODP) for Asymmetric multicore systems, we evaluate an asymmetric multicore system built with CPU-GPU by measuring the overheads of memory transfer, computing kernel, cache missing and synchronization. This paper demonstrates that decreasing the overhead of data preparation is a promising approach to improve the whole performance of heterogeneous system.

Keywords: Overhead of data preparation; heterogeneous multicore; Amdahl's law; power consumption

A preliminary version of this paper appeared in The 1st International Conference on Electronics, Electrical Engineering, Computer Science: Innovation and Convergence (EEECS 2016), January 20-22, Phuket, Thailand. This version includes a concrete analysis and supporting implementation results on asymmetric mulcitore system on graphics processing. This research was supported by a research grant from the Shanghai Municipal Natural Science Foundation (15ZR1428600), Shanghai Pujiang Program(16PJ1407600), the program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, USST incubation project(15HJPY-MS02), and the National Science Foundation (XPS-1439097).

1. Introduction

As we enter the multicore era, asymmetric multicore system[1] is becoming the mainstream in the field of future processor design. Recent examples include Intel's MIC[2], AMD's Kabini[3], and NVIDIA's Project Denver[4], etc. However, the memory wall and communication issues will continue increasing the gap between the performance of an ideal processor and that of a "practical" processor, because the overhead of data preparation becomes an unavoidable key problem.

Amdahl's Law was a state-of-the-art analytical model that guided software developers to evaluate the actual speedup that could be achieved by using parallel programs, or hardware designers to draw much more elaborate microarchitecture and components. Gene Amdahl[5] defined his law for the special case of using n processors (cores) in parallel when he argued for the single-processor approach's validity for achieving large-scale computing capabilities. He used a limit argument to assume that a fraction f of a program's execution time was infinitely parallelizable with no scheduling overhead and other overheads to generate ready data to be used by computing units, while the remaining fraction, $1-f$, was totally sequential. He noted that the speedup on n processors is governed by equation (1):

$$Speedup(f,n) = \frac{1}{(1-f)+\frac{f}{n}} \tag{1}$$

This equation Amdahl's law on basis of computing-centric system which never takes into account the potential cost of data preparation. It is only correct as long as three key assumptions are verified:

(1) the programs to be executed are of fixed-datasize and the fraction of the programs that is parallelizable remains constant as well;

(2) there exists an infinite memory space without extra overhead of switching memory blocks and disk blocks, etc.

(3) the overhead of preparing the data to be used by computing units, which includes memory access, communication on-chips or off-chips and synchronization among cores, can be neglected.

Currently, some CPU-GPU memory architectures are using separate memory address spaces for the CPU and the GPU, while the programmer is responsible for communication between them by using a relatively slow communication channel,such as PCIe bus. This communication bottleneck limits the peak throughput that can be obtained from these systems. Otherwise, the power consumption has become an important issue that impacts GPU applications. Therefore, traditional Amdahl's Law is not fit for heterogeneous computer system, we will take consideration of the overhead of data preparation in asymmetric multicore system.

Despite some criticisms, Amdahl's Law is still relevant as we enter a heterogeneous multi-core computing era. However, the future relevance of the law requires its extension by the inclusion of constraints and architectural trends demanded by modern multiprocessor chips. There are a lot of achievable researches in theory to extend Amdahl's Law. Woo and Lee[6] extended Amdahl's law for energy-efficient computing of many-core, who classified many-core design styles as three type: symmetric superscalar processor tagged with P*, symmetric smaller power-efficient core tagged with c*, and asymmetric many-core processor with superscalar processor and many smaller cores tagged with P + c*. The research results show that heterogeneous architecture is better than symmetric system to save power. Similarly, Marowka[7] extended Amdahl's law for heterogeneous computing, and investigated how energy efficiency and scalability are affected by the power constraints for three kind of heterogeneous computer system, i.e., symmetric, asymmetric and simultaneous asymmetric. The analysis shows clearly that greater parallelism is the most important factor affecting power consumption. Karanikolaou, et al evaluated experimental energy consumption based on both of performance/power and performance/energy ratio metrics[8]. Kim et al focused on the energy efficiency of the sequential part acceleration, and how to determine the optimal frequency boosting ratio which maximize energy efficiency[9]. Londono et al modeled the potential dynamic energy improvement, optimal frequency and voltage allocation of a multicore system in terms of extending Amdahl's law[10]. EyerMan et al [11] introduced a model that accounts for critical sections of te parallelizable fraction of a program. Ge et al proposed a power-aware speedup model to predict the scaled execution time of power-aware clusters by isolating the performance effects of changing processor frequencies and the number of nodes[12].

In this paper we analyse the extended Amdahl's law metioned in the papers[16-19] and demonstrate that data preparation is another promising approach to improve performance of heterogeneous system. The contributions in this paper are:

(1) Rethinking and analysing the general model of Amdahl's law;

(2) Proposing the overhead of data preparation in the extended Amdahl's law;

(3) Reevaluating an asymmetric multicore system built with CPU-GPU by measuring the overheads of communication, computing kernel, cache missing and synchronization.

This paper is organized as follows. Section II revisits the extended Amdahl's law of asymmetric multicore system by considering the overhead of data preparation[13]. Section III analyses the affecting factors of data preparation. Section IV proves the affecting factors of data preparation should not be neglected through Rodinia benchmarks. Section VI concludes our work and future missions.

2. Revisiting the extended Amdahl's law by considering the overhead of data preparation

The graphics processing unit (GPU) has made significant strides as an accelerator in parallel computing. However, GPU has resided out on PCIe as a discrete device, the performance of GPU applications can be bottlenecked by data transfers between the CPU and GPU over PCIe. So the overhead of preparing the data and accessing the memory, of transmitting data on-chip and off-chip, of transferring data between CPU memories and GPU memories for heterogeneous system, of synchronizing processes, etc, become significant that it cannot be ignored any longer. However, traditional Amdahl's law only considers the cost of instruction execution.

We will thus now assume that whole cost of executing a program can be split into two independent parts: (p_c) and $(1-p_c)$, as shown in the **Fig. 1**, one $(1-p_c)$ is preparing the data execution and the other one (p_c) is running instruction when the required data are ready.

Therefore, the Overhead of Data Preparation (ODP) includes the whole cost of preparing data for execution.

As proposed by [13], ODP can be considered to produce a new speedup equation will call it the "Extended Amdahl's law" as expressed in equation (2):

$$S_{EA}(f_c, c, p_c) = \frac{1}{((1-f_c)+\frac{f_c}{c})p_c + (1-p_c)} \tag{2}$$

where p_c denotes the computation portion, $1-p_c$ denotes the Data Preparation portion normalized to the computation portion: since the clock frequencies and the ISAs of CPUs, GPUs, off-chip bus and memory arbitrator would likely be different, we should normalize the performance of Data Preparation instructions to that of computing instructions. The f_c is the parallelizable computation portion, $1-f_c$ is the sequential computation portion.

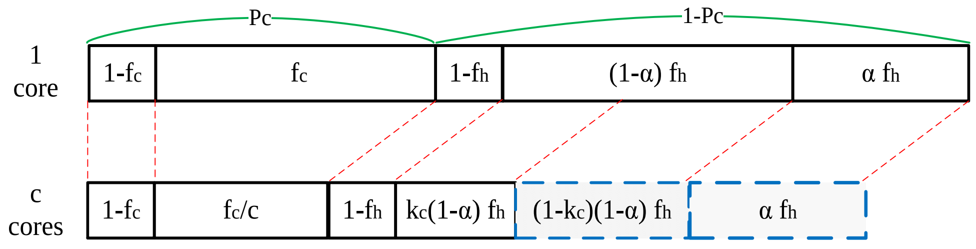

Fig. 1. Illustration of Extended Amdahl's law

As shown in **Fig. 1**, however, equation (2) does not make allowances for the introduction of techniques that would decrease or eliminate the overhead of data preparation. We further divide the portion of data preparation into three sub-parts: $1-f_h$, αf_h and $(1-\alpha)f_h$. f_h denotes the portion of the program that can be overlapped with "computing" instructions in theory, where 0< f_h <1. But, it cannot be completely overlapped in practical due to the unpredicted run-time negative factors from the limits of hardware resources. In other words, the execution of this portion is masked behind the execution of the computation instructions.

In general, memory access instructions can be executed in parallel with independent computing instructions. However, it will not be possible to execute all of the data preparation instructions simultaneously with computing instructions. For example, if a load instruction is theoretically independent of the next computing instructions, it can be theoretically issued with the next computing instructions. However, it would not be issued because the queue of issuing load instructions is full. Therefore, we introduce the parameter α to denote the percentage of data preparation instructions which are actually executed simultaneously with computing instructions before using advanced technologies, where 0< α <1. Thus, αf_h denotes the portion of real parallelized instructions for data preparation and $(1-\alpha)f_h$ denotes the portion of data preparation instructions which cannot be overlapped with computing instructions before adopting advanced technologies.

Furthermore, with the help of advanced technologies such as data prefetching, speculative execution, universal memory, no-copy data transfer, 3D NoC, etc., the fraction of data preparation instructions which cannot be overlapped on one processor will sharply decrease as the number of cores increases. In other words, $(1-\alpha)f_h$ could be decreased significantly. Furthermore, we introduce a variable k_c to model how much percentage of data preparation that cannot be overlapped on a *c* cores system even adopting advanced technologies, where $0<k_c<1$. After normalization to the computation, the fraction of data preparation instructions which cannot be overlapped on a *c* core system becomes $f_{ud}=(1-f_h)+k_c\cdot(1-\alpha)f_h$. On the opposite, the fraction of data preparation instructions which can be overlapped on a *c* core system becomes $f_{pd}=\alpha f_h+(1-k_c)\cdot(1-\alpha)f_h$, where $f_{ud}+f_{pd}=1$. While, the $1-f_h$ denotes the fraction of data preparation which is closely dependent on computation. Therefore, we can extend Amdahl's law to be a new equation called "Enhanced Amdahl's law". The performance speedup is governed by:

$$S'_{EA}(f_c, c, p_c, f_{ud}) = \frac{1}{((1-f_c)+\frac{f_c}{c})p_c + f_{ud}\cdot(1-p_c)} \tag{3}$$

3. Analysis the impact factors of data preparation

Some multicore performance models tend to ignore the impact of data synchronization, CPU-GPU communication, cache missing rate and so on. In this section, we take account of some possible impact factors such as kernel overhead, memory overhead, data transfer overhead and cache missing rate which contribute to the overhead of data preparation.Amdal's Law on basis never takes account of those overhead.

3.1 Memory Transfer Overhead

In CPU-GPU heterogeneous computing, one overhead is caused by the disjoint address spaces of the CPU and GPU and the need to explicitly transfer data between the two memory spaces. Before executing a kernel on the GPU, all of the data used by kernel needs to be transferred from the CPU memory to the GPU memory. After execution, the data produced by the kernel mostly needs to be transferred back to the CPU memory. The function to accomplish this memory transferring is usually *cudaMemcpy*.

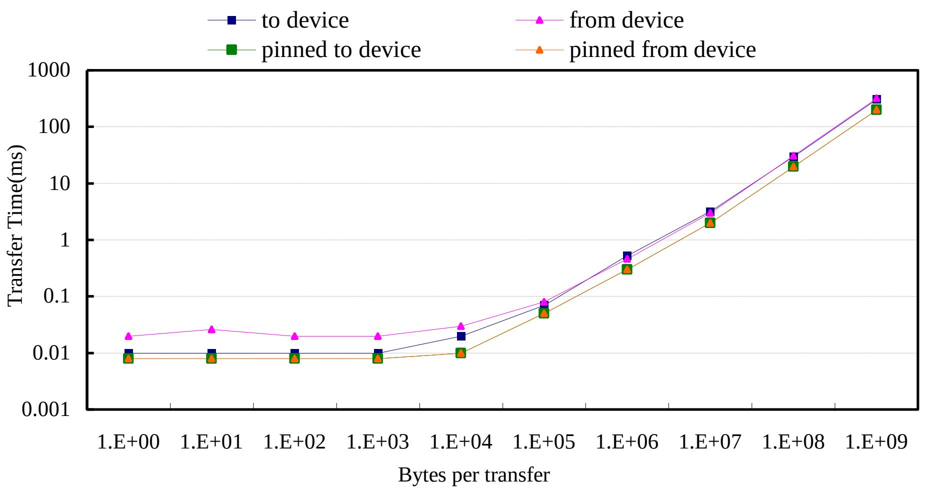

Fig. 2. Time per memory transfer for different size of transferred data

Fig. 2 shows the average transfer time by using *cudaMemcpy* strides a wide range as the size of transferring data increase. Four different configurations are measured, and their results are all different whether the data are transferred to or from the GPU memory, and whether we use pinned or non-pinned memory. Pinned memory is the memory which is allocated by using the *cudaMallocHost* function, which prevents the memory from being swapped out and provides improved transfer speeds. However, the non-pinned memory is the memory which is allocated by using the *malloc* function.

For the small size of transfer data, there is a constant overhead of transferring data which is around 10 µs. As the size of transferring data increasing about 8 KB, the transfer time starts to increase linearly. Besides, the overhead of transferring data in pinned memory is smaller than that in non-pinned memory.

3.2 Kernel Overhead

Kernel overhead includes the overhead of synchronization and launching a kernel. Synchronization overhead can be a barrier to achieving good performance for applications utilizing fine-grained synchronization. The overhead of launching a kernel can severely impact the performance of a CUDA application if the kernel is invoked many times. To evaluate the overhead of kernel execution, we measured the average time required to launch an empty kernel over a large number of kernel invocations. **Fig. 3** shows the time of calling each kernel on four different GPUs. The asynchronization bar represents the case where the kernel is invoked repeatedly without synchronization between calls. The synchronization bar represents the case where the *cudaThreadSynchronize* function is called after calling each kernel, and forces the CPU to wait for the execution of current kernel to be finished before calling the next kernel. The y-axis is the time for calling a kernel, which mostly results from the overhead of accessing the GPU driver API and Runtime API. As shown in the **Fig. 3**, the average time of calling a kernel with synchronization is about 3.5 times than that of asynchronous mode. Therefore, the synchronization overhead is much greater than that of asynchronous.

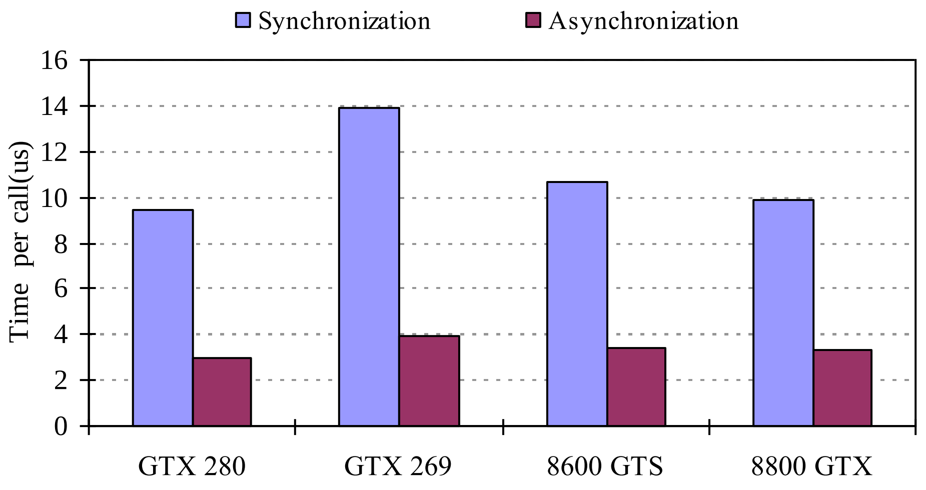

Fig. 3. Time of calling an empty kernel

3.3 Cache Missing Rate

Another overhead of data preparation is caused by the cache missing. The application with large problem sizes and irregular accessing memory would cause serious Cache missing in the run-time because of unconventional memory access patterns are poorly handled by pre-fetching.

CPUs are designed to reduce the effective memory access latency through extensive multi cache level and prefetching technologies. Thus, a slightly irregular memory access pattern can be successfully captured by the CPU caches. However, the same access pattern may be irregular enough to prevent efficient utilization of the GPU memory bandwidth, due to the restrictions on access patterns that must be met in order to achieve good memory performance. The restrictions of access pattern can be partially relaxed by taking advantage of the GPU special-purpose address spaces. Both constant and texture memory provide small on-chip caches that allow threads to take advantage of fine-grained spatial and temporal locality.

In addition, texture memory relaxes the alignment requirements that must be met in order for multiple memory accesses from within the same warp to be coalesced. Zero copy improves performance by eliminating these redundant data copies. Another effective approach is to use the software-controlled shared memory as an explicitly managed cache, which can significantly improve performance when data elements are frequently reused.

4. Experimental Evaluation

In this section, to evaluate the overhead of the data preparation, we performed different applications from *Rodinia* benchmark. In order to implement GPU programs, the *Rodinia* suite uses *CUDA* [14, 20], an extension to C for GPUs. CUDA is a parallel computing platform and programming model invented by *NVIDIA*. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU). *Rodinia* is a benchmark suite for heterogeneous computing. To help architects study emerging platforms such as GPUs (Graphics Processing Units), Rodinia includes applications and kernels which target on multi-core CPU and GPU platforms.

4.1 Platform and Applications

The benchmarks used for experiment are mentioned in **Table 1**. The *Rodinia*[15] benchmark suite is designed to provide parallel programs for the study of heterogeneous systems. We use seven benchmarks from *Rodinia*, which are k-means (KM), back propagation (BP), breadth-first search(BFS), hotspot(HS), Needleman-Wunsch(NW), speckle reducing anisotropic diffusion and stream cluster(SRAD).

Table 1. Selected applications from Rodinia Benchmark

Application	Description	Input dataset
BFS	Breadth First Search	Graph1MW_6.txt
NW	Needleman-Wunsch (Dynamic Programming)	2048*2048
HS	Hot spot (Structured Grid)	500*500
BP	Back propagation	65536
KM	K-means Clustering	819200
SC	Stream cluster	65535
SRAD	Speckle reducing anisotropic diffusion	2048*2048

The KM is a clustering algorithm used extensively in the field of data mining. This identifies related points by associating each data point with its nearest cluster, computing new cluster centroids, and iterating until convergence. BP is a machine-learning algorithm for layered neural network. This application is comprised of two phases: the forward phase and backward phase. HS is a thermal simulation tool used for estimating processor temperature based on an architectural floor plan and simulated power measurements. NW is an optimization method for DNA sequence alignments, and BFS traverses all the connected components in a graph. SRAD is a diffusion algorithm based on partial differential equations and used for removing the speckles in an image without sacrificing important image features. SRAD is widely used in ultrasonic and radar imaging applications. The inputs to the program are ultrasound images and the value of each point in the computation domain depends on its four neighbors.

Those seven benchmarks can be divided into two categories: compute-intensive and memory-intensive applications. SRAD and HotSpot are relatively compute-intensive, while Needleman-Wunsch, Breadth-First Search, Kmeans, and Stream Cluster are memory-intensive application.

4.2 Experimental Results

The GPU in our experimental platform for heterogeneous system is NVIDIA GTX 750. It has 512 streaming multi-processors (SMs), and 2 GB device memory. We investigate the overhead of data preparation by comparing against the total execution time of the applications from *Rodinia* benchmark. As mentioned in section III, We mainly focus on overhead of memory transfer, synchronization and cache missing rate. The computing performance of applications is mostly affected by those overheads, which we will discuss in the following sections.

As shown in **Fig. 4**, the overhead of different factors are analyzed, such as kernel execution, CPU-GPU communication, CPU execution and synchronization. And to different applications, the influence of those factors is not the same. For example, the CPU-GPU commucation overhead of SRAD and BP are relatively high because of they have a large dataset. In addition, as for NW, HS and BP, synchronization overhead makes up a high proportion of data preparation for the data dependent.

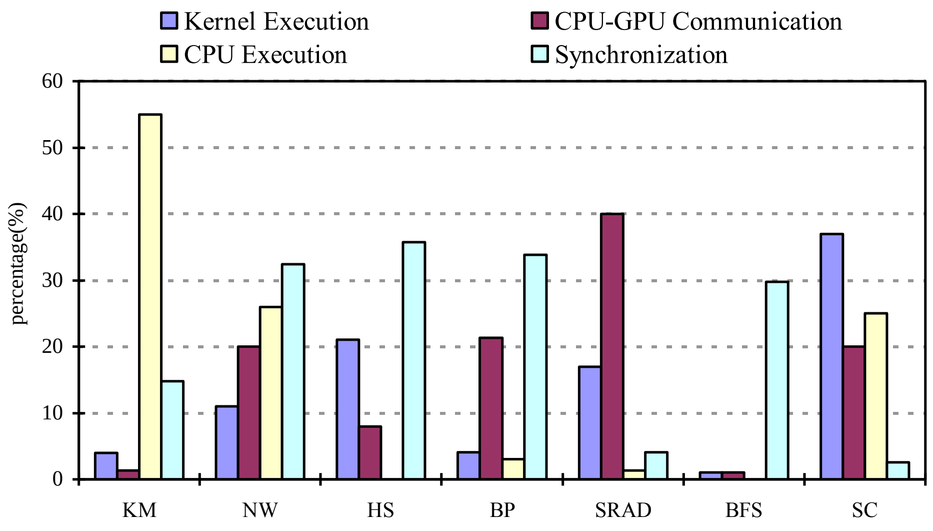

Fig. 4. Percentage of total execution time

4.2.1 Communication Overhead

The overhead of communication between GPUs and CPUs often becomes a major factor to computing performance. For large dataset, the overhead of data preparation is mainly determined by the bandwidth of PCIe bus. Slowing data transfer between CPU and GPU exacerbates the overhead of data preparation.

As shown in the **Fig. 5**, SRAD and BP require significant CPU-GPU communication time because they need a lot of data to be inputted. The average of communication time for SRAD and BP is almost one-third of the whole runtime. As for NW and HS, all of the computation is done on the GPU and the results are explicitly transferred back only after all GPU work has been completed. In these applications, the memory transfer overhead is hard to be further reduced. In conclusion, communication overhead should be considered in the Extended Amdahl's Law.

Understanding the application's memory access patterns on the GPU is crucial to achieve good performance. If the neighboring threads access neighboring rows in an array, allocating the array in column-major order will allow threads within the same warp to access contiguous elements. So warp scheduling can hide the memory latency and decrease the communication overhead to some extent.

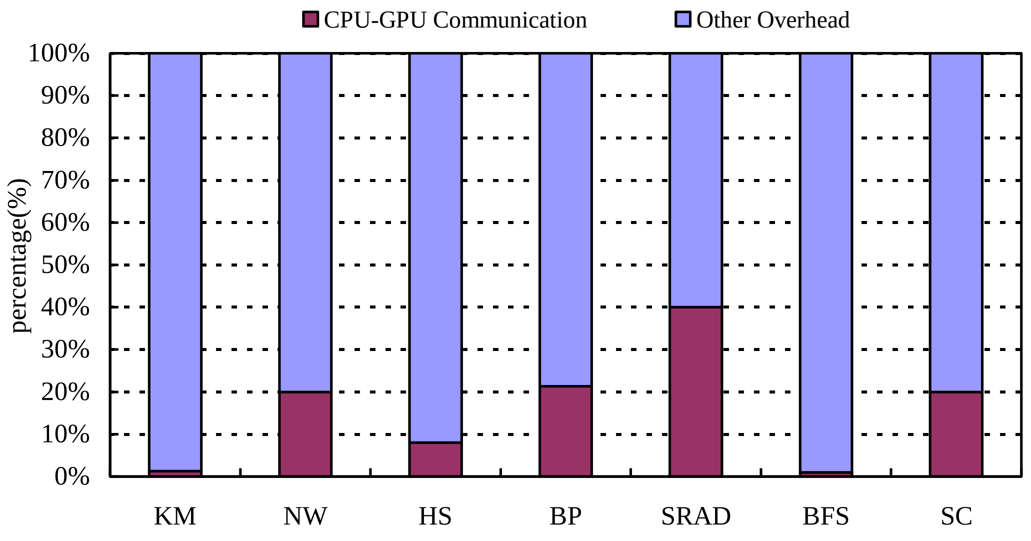

Fig. 5. CPU-GPU communication overhead

4.2.2 Synchronization Overhead

As shown in the **Fig. 6**, the overhead of synchronization is a barrier to achieve high performance of applications by utilizing fine-grained synchronization. SRAD and SC show relatively low synchronization overhead because the majority of their computations are independent. However, NW, HS, BP and BFS have great overhead of synchronization. The synchronization percentage of those applications is between 30% and 40%, because the majority of their computations are interdependent. As show in equation (3), if parallelism is low, ignore the effects of data synchronization overhead is acceptable. But as parallelism growing, this effect becomes more predominant. Che et al. [18] also showed the importance of using the cached, read-only constant memory and texture memory spaces for frequently reused data within a thread block. These methods are especially helpful in reducing the overhead of GPU's data synchronization.

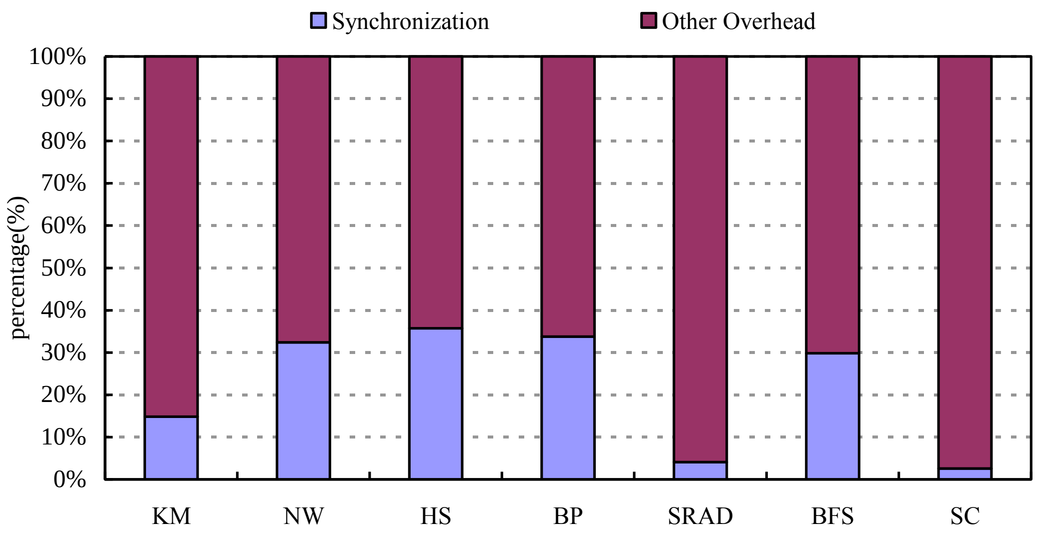

Fig. 6. Synchronization overhead

At all, the overhead of CPU-GPU communication and synchronization has great effect on data preparation . As show in **Fig. 7**, we can calculate the total percentage of communication and synchronization during the running time. The rate of NW and BP are already above the half of the total time (NW is 52.4% and BP is 55.1%).The average rate of those benchmarks is 37.83%.According to **Fig. 7**, we find that data synchronization and communication affect Amdahl's Law in the hetergeneous architecture. It is apparent that the overhead of data preparation (CPU-GPU communication and data syncronization) takes up a great proportion of the total execution time.

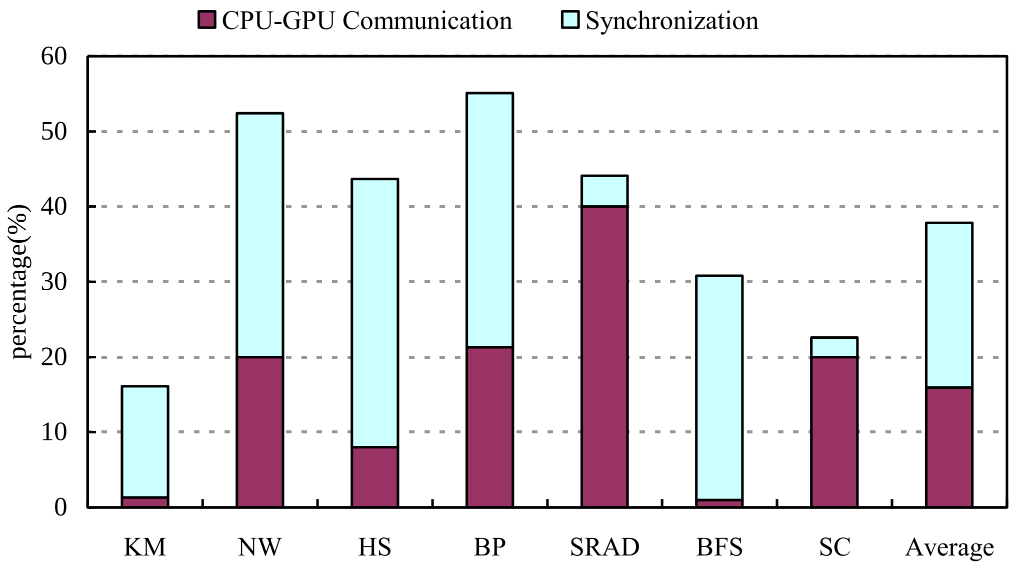

Fig. 7. Overhead of Communication and Synchronization

4.2.3 The Overhead of Cache Missing

Cache missing rate is another factor which contributes to the overhead of data preparation. As shown in the **Fig. 8**, irregular data access is poorly handled by pre-fetching. NW exhibits a high L2 cache missing rate of 41.0% because of its irregular memory access patterns (diagonal strips). BFS exhibits L2 cache missing rate of 20.8% and KM exhibits 27.1%, which exhibit regular behavior, present cache miss rates that are lower but still high enough to be of interest. The L2 cache miss rates of other applications range from 2.0% to 12.0%. In terms of those experimental data, we conclude that the applications with memory-intensive and irregular data accessing would incur high cache missing rate.

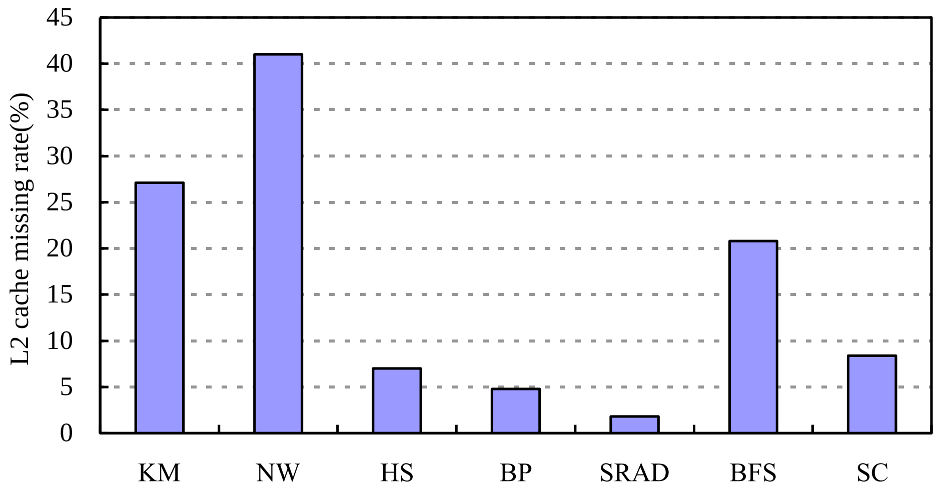

Fig. 8. L2 cache missing rate

5. Conclusion

According to the experiment results, calling a kernel would incurs high overhead of performance and programming to deal with the CPU-GPU communication (as in BP and SRAD). High synchronization overheads are caused by computations interdependent such as HS and BP. Therefore, the overhead of data preparation is a major factor which we should not be neglected in heterogeneous system. Traditional Amdahl’s Law ignores those overheads, so for tasks of high synchronization, parallel execution may result in significantly lower speedup than as predicted by traditional Amdahl’s Law.

When we take into account the overhead of data preparation, the speedup will not grow linearly as the number of cores increases. Therefore, the speedup obtained is always less than the ideal in terms of traditional Amdahl's law. Therefore, we consider the overhead of data preparation is an important factor by extending traditional Amdahl’s Law.

The effects of communication, synchronization and cache missing rate can be overcome by proper architectural and software design. If a multicore architecture has no private memory, then there would be no need in synchronization and communication: the only one memory is shared by all of the cores and data access at the same space. Therefore such memory architecture would be unaffected by those overheads of data preparation. But this may incur long memory latency.

In this paper, we have presented an Extended Amdahl’s Law taking into account the effects of data preparation. By experiment, the heterogeneous multicore performance is fundamentally limited by data synchronization, CPU-GPU communication and cache missing rate. The maximum performance speedup achieved by heterogeneous multicore can be lower than predicted by Amdahl's Law. In the future work, we will keep on investigating the performance gain within limited energy (or power) budget while considering the overhead of data preparation.

References

[1] P.Rogers, "Heterogeneous System Architecture Overview," in *Proc. of Hot Chips Sym.*, Aug. 2013. Article (CrossRef Link)

[2] A.Duran and M.Klemm, "The Intel Many Integrated Core Architecture," in *Proc. of the International Conference on High Performance Computing and Simulation(HPCS)*, pp.365-366, July 2012. Article (CrossRef Link)

[3] D.Bouvier, B.Cohen, W.Fry, S.Godey and M.Mantor, "Kabini: An AMD Accelerated Processing Unit System on A Chip," *MICRO*, vol.34, no.2, pp.22-33, March 2014. Article (CrossRef Link)

[4] Nvidia Corporation. Nvidia Project Denver, http://www.nvidia.com (available in Feb. 2016).

[5] Amdahl G M., "Validity of the single processor approach to achieving large scale computing capabilities [J]," in *Proc. of the American Federation of Information Processing Societies (AFIPS)*, 483-485, April 1967. Article (CrossRef Link)

[6] D.H.Woo and H.S.Lee, "Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era," *Computer*, vol.41, no.12, pp.24-31, Dec.2008. Article (CrossRef Link)

[7] A. Marowka, "Extending Amdahl's Law for Heterogeneous Computing," in *Proc. of the 2012 International Symposium on Parallel and Distributed Processing with Applications(ISPDPA)*, pp.309-316, July 2012. Article (CrossRef Link)

[8] E.M.Karanikolaou, E.I.Milovanovic ́, I.Z ̌.Milovanovic ́,M.P.Bekakos, "Peformance Scalability and Energy Consumption on Distributed and Many-core Platforms," *Journal of Supercomputing*, vol.70, no.1, pp.349-364,Oct. 2014. Article (CrossRef Link)

[9] S.H.Kim, D.Kim, C.Lee, W.S.Jeong, W.W.Ro, and J.L.Gaudiot, "A Performance-Energy Model to EvaluateSingle Thread Execution Acceleration," *Computer Architecture Letters*, vol.14, no.2, pp.99-102, Nov. 2014. Article (CrossRef Link)

[10] S.M.London ̃o, and J.P.Gyvez, "Extending Amdahl's Law for Energy-Efficiency," in *Proc. of International Conference on Energy Aware Computing(ICEAC)*, pp.1-4, Dec. 2010. Article (CrossRef Link)

[11] Eyerman S, Eeckhout L., "Modeling critical sections in Amdahl's law and its implications for multicore design [J].," in *Proc. of the 37th Annual International Symposium on Computer Architecture, ACM*, New York, US, 38(3):362-370, 2010. Article (CrossRef Link)

[12] R.Ge and K.W.Cameron, "Power-Aware Speedup", in *Proc. of International Parallel and Distributed Processing Symposium (IPDPS)*, March 2007, pp.1-10. Article (CrossRef Link)

[13] S.W.Pei, M.S.Kim, J.L.Gaudiot, "Extending Amdahl's Law for Heterogeneous Multicore Processor with Consideration of the Overhead of Data Preparation," *IEEE Embedded Systems Letters*, vol. 8, no.1, pp.26-29, March. 2016. Article (CrossRef Link)

[14] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *ACM Queue*, vol.6, no.2, pp.40–53, 2008. Article (CrossRef Link)

[15] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer,S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proc. of IEEE International Symposium on Workload Characterization*, pp.44–54. 2009. Article (CrossRef Link)

[16] Hill, Mark D, Marty, Michael R., "Amdahl's Law in the Multicore Era [J]," *Computer*, 2008, 41(7): 33-38. Article (CrossRef Link)

[17] Erlin Yao, Yungang Bao, Guangming Tan, et al., "Extending Amdahl's Law in the Multicore Era[J].," *ACM SIGMETRICS Performance Evaluation Review*, 37(2):24-26, 2009. Article (CrossRef Link)

[18] Xian-He Sun, Yong Chen, "Reevaluating Amdahl's law in the multicore era [J]," *Journal of Parallel and distributed Computing*, 70(2):183-188, 2010. Article (CrossRef Link)
[19] Andreou A G. Beyond Amdahl's Law: An Objective Function That Links Multi-processor Performance Gains to Delay and Energy [J]. IEEE Transactions on Computers, 61(8):1110-1126 , 2012. Article (CrossRef Link)
[20] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general purpose applications on graphics processors using CUDA," *Journal of Parallel and Distributed Computing*, 68(10):1370–1380, 2008. Article (CrossRef Link)

Songwen Pei received the B.S. in the School of Computer from National University of Defence and Technology, Changsha, China in 2003, the M.S. in the School of Information Engineering from Guizhou University, Guiyang, China in 2006, and the Ph.D. in the School of Computer Science and Technology from Fudan University, Shanghai, China in 2009. He is currently an Associate Professor of the Computer Science and Engineering Department at the University of Shanghai for Science and Technology, and he currently works as a Guest Researcher at the Institute of Computing Technology, Chinese Academy of Sciences (2011-), and a Research Scientist at the Electrical Engineering and Computer Science, University of California(2013-2015). His research interests include heterogeneous multi-core processors, parallel and distributed computing, cloud computing, big data, and fault-tolerant computation, etc. He is a member of the IEEE, ACM and CCF in China, and he is also a board member of CCF-TCCET, CCF-TCARCH and CCF-YOCSEF Shanghai respectively. He is a talent award-winner of Shanghai Pujiang Program 2016.

Junge Zhang received the B.S. in the computer science from Luoyang Normal University in 2012. She is currently a graduate student in computer architecture at the University of Shanghai for Science and Technology. Her current research interests include computer architecture, Multi-core heterogeneous systems, data prefetching and GPU power model.

Linhua Jiang received M.S. degree and Ph.D. degree from Catholic University of Leuven and Leiden University respectively. He is currently a Professor in the School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, and is the director of the Optical-Computer Science Research Group. He was a visiting Faculty of Stanford University. His current research fields include Optical-Information and Image Processing, Computer Network and Cyber-Physical System, Parallel Computing, etc. Prof. JIANG published more than one hundred scientific papers and conference articles.

Myoung-Seo Kim received both the B.S. degree in Computer Science and the B.S. degree in Electrical and Electronics Engineering and the M.S. degree in Computer Science from Yonsei University, Seoul, Korea in 2003 and 2005, respectively. He has worked for 5 years performing design and verification of portable multimedia system-on-a-chip (SoC) at Samsung Electronics, Yongin, Korea (2005-2008) and at Apple Inc., Cupertino, California (2008-2009). He received the Ph.D. degree in Computer Science from University of California, Irvine in 2016. He is currently a principal researcher in the Parallel Systems and Computer Architecture Lab at University of California, Irvine. He was the recipient of the 2011 Yonsei International Foundation Scholarship and the 2012 SK Hynix Study Abroad Scholarship. In addition, he has published more than 15 papers in journals and international conferences. His research interests include multi/many-core system-on-a-chip, computer architecture and design, low-power architecture, embedded systems, VLSI circuit design and analysis, and design automation. He is a member of the IEEE and the ACM.

Jean-Luc Gaudiot received the Diplôme d’Ingénieur from ESIEE, Paris, France in 1976 and the M.S. and Ph.D. degrees in Computer Science from the University of California, Los Angeles in 1977 and 1982, respectively. He is currently a Professor and Chair of the Electrical and Computer Engineering Department at the University of California, Irvine. Prior to joining UCI in January 2002, he was a Professor of Electrical Engineering at the University of Southern California since 1982, where he served as and Director of the Computer Engineering Division for three years. He has also done microprocessor systems design at Teledyne Controls, Santa Monica, California (1979-1980) and research in innovative architectures at the TRW Technology Research Center, El Segundo, California (1980-1982). He consults for a number of companies involved in the design of high-performance computer architectures. His research interests include multithreaded architectures, fault-tolerant multi-processors, and implementation of reconfigurable architectures. He has published over 200 journal and conference papers. His research has been sponsored by NSF, DoE, and DARPA, as well as a number of industrial organizations. In January 2006, he became the first Editor-in-Chief of IEEE Computer Architecture Letters, a new publication of the IEEE Computer Society, which he helped found to the end of facilitating short, fast turnaround of fundamental ideas in the Computer Architecture domain. From 1999 to 2002, he was the Editor-in-Chief of the IEEE Transactions on Computers. He has served on the IEEE Computer Society Board of Governors for two terms (2010-1015), was VP of Educational Activities (2013), VP of Publications (2014-2015) and is the 2017 IEEE Computer Society President. In 1999, he became a Fellow of the IEEE. He was elevated to the rank of AAAS Fellow in 2007.