# Extending Amdahl's Law for Heterogeneous Multicore Processor with Consideration of the Overhead of Data Preparation

Songwen Pei, *Member, IEEE*, Myoung-Seo Kim, *Student Member, IEEE*, and Jean-Luc Gaudiot, *Fellow, IEEE*

***Abstract*—We extend Amdahl's law by considering the overhead of data preparation (ODP) for multicore systems, and apply it to three "traditional" multicore system scenarios (homogeneous symmetric multicore, asymmetric multicore, and dynamic multicore) and two new scenarios (heterogeneous CPU-GPU multicore and dynamic CPU-GPU multicore). It demonstrates that potential innovations in heterogeneous system architecture are indispensable to decrease ODP.**

***Index Terms*—Amdahl's law, heterogeneous systems, overhead of data preparation (ODP), performance evaluation, speedup model.**

## I. Introduction

As MULTICORE processors have become mainstream, it has become crucial to identify performance bounds and performance scaling properties in exploiting the massive parallelism they offer [1]. Besides, processor design has been transiting from the homogeneous multicore model to the heterogeneous model [2]. Furthermore, energy efficiency and scalability are affected by the power constraints imposed on heterogeneous processors [3]. The memory wall [4] and communication issues will continue increasing the gap between the performance of an ideal processor and that of a "practical" processor. Therefore, the overhead of data preparation becomes an unavoidable key parameter. Assume that a fraction $f$ of a program's execution is infinitely parallelizable without scheduling and synchronizing overhead with $c$ processors (cores), while the remaining fraction $1-f$ is assumed to be sequential execution. Then, the computational speedup of the system is governed by the well-known equation (Amdahl's law [5], [6])

$$S_A(f,c) = \frac{1}{(1-f)+\frac{f}{c}}. \tag{1}$$

Manuscript received November 29, 2015; accepted January 15, 2016. Date of publication January 19, 2016; date of current version February 25, 2016. This work was supported in part by the Shanghai Municipal Natural Science Foundation (15ZR1428600) and by the National Science Foundation (XPS-1439097). This manuscript was recommended for publication by A. Gordon-Ross.

S. Pei is with Shanghai Key Lab of Modern Optical Systems, University of Shanghai for Science and Technology, Shanghai 200093, China, and also with the Parallel Systems and Computer Architecture Lab, University of California, Irvine, CA 92697 USA (e-mail: swpei@usst.edu.cn).

M.-S. Kim and J.-L. Gaudiot are with Parallel Systems and Computer Architecture Lab, University of California, Irvine, CA 92697 USA (e-mail: myoungseo.kim@uci.edu; gaudiot@uci.edu).

Digital Object Identifier 10.1109/LES.2016.2519521

The equation is correct as long as three key assumptions are verified: 1) the programs to be executed are of fixed-size 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.*; and 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. In practical multicore systems, as it turns out, the overhead of data preparation is between 30% and 90% of the total execution time depending on the kind of application considered [7]. In other words, the overhead of transferring data between CPUs and GPUs in a heterogeneous multicore system could be the bottleneck of high performance computing. As we introduce this parameter in the single-threaded multicore system, we propose a set of equations in the following sections.

## II. Revisiting Speedup Model by Considering the Overhead of Data Preparation (ODP)

As proposed by Hill & Marty [6], we first assume that a multicore chip of a given area and manufacturing technology is composed of at most $n$ base core equivalents (BCEs) (a single BCE implements a baseline core). Then, we assume that the resources of $r$ BCEs can be used to create a powerful core with sequential performance $\mathrm{perf}(r)$, while the performance of a single-BCE core is 1 [6]. $\mathrm{perf}(r)$ is an arbitrary function, where $1 < \mathrm{perf}(r) < r$.

Since an improvement in sequential performance by microarchitecture techniques alone would follow Pollack's rule [8], $\mathrm{perf}(r)$ is roughly proportional to the square root of the increase in complexity. At the same time, because of the "memory wall," the overhead of preparing the data and accessing the memory, of transmitting data on- and off- chip, of transferring data between CPU memories and GPU memories for heterogeneous system, of synchronizing processes, *etc.*, becomes so significant that it cannot be ignored any longer. However, Amdahl's law only considers the cost of instruction execution. We will thus now assume that the whole cost of executing a program can be split into two independent parts, one is preparing the data for execution and the other one is running instructions when the required data are ready. Therefore, the overhead of data preparation (ODP) includes the cost of preparing data for execution, to the exclusion of actual execution. As shown in Fig. 1, ODP can be considered to produce a

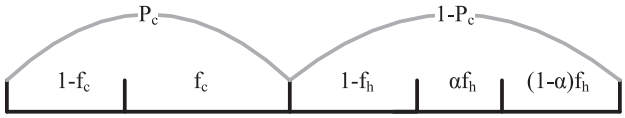


Fig. 1. Normalized task (equivalence time). Split between Computation and Data Preparation.

new speedup equation; we will call it the "Extended Amdahl's law" as expressed in equation (2)

$$S_{\mathrm{EA}}(f_c, c, p_c) = \frac{1}{\left((1-f_c) + \frac{f_c}{c}\right) \cdot p_c + (1-p_c)} \quad (2)$$

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.

As shown in Fig. 1, however, equation (2) does not make allowance for the introduction of hardware techniques that would decrease or eliminate the overhead of data preparation. We can further divide the portion of data preparation into three subparts: $1-f_h$, $\alpha f_h$ and $(1-\alpha)f_h$. $1-f_h$ denotes the portion of data preparation which is closely dependent on computing instructions. $f_h$ denotes the data preparation portion of the program that can be overlapped with "computing" instructions before introducing advanced techniques, where $0 \leq f_h \leq 1$.

Memory access instructions generally can be executed simultaneously with independent computing instructions. However, it will not be possible to execute all of the data preparation instructions simultaneously with computing instructions. Further, not all operations of data preparation would be executed with computing instructions according to our observations. For example, if a load instruction is independent of the following computing instructions, it can be issued simultaneously with them. While it would not be issued if the queue of issuing load instructions was full.

Hence, we introduce the parameter $\alpha$ to denote the percentage of data preparation instructions which are actually executed simultaneously with computing instructions before using advanced architectural techniques ($0 \leq \alpha \leq 1$). Thus, $\alpha f_h$ denotes the portion of actual parallelized instructions for data preparation and $(1-\alpha)f_h$ denotes the portion of data preparation instructions which cannot be overlapped with computing instructions without sophisticated architecture techniques.

With the help of advanced architectural techniques such as data prefetching, speculative execution, universal memory, no-copy data transfer, 3-D NoC, *etc.*, $(1-\alpha)f_h$ could be decreased significantly. Thus, we further introduce the variable $k_c$ to model what percentage of data preparation cannot be overlapped on a $c$ cores system by using new advanced techniques ($0 \leq k_c \leq 1$). After normalization to the computation part in theory, the portion of data preparation instructions which cannot be overlapped on a $c$ core system becomes $f_{\mathrm{ud}} = (1-f_h) + k_c \cdot (1-\alpha)f_h$. Assuming that the execution time of parallelizable data preparation is less than the execution time of computation and the serial data preparation time put together, the portion of parallelizable data preparation can be masked by computation and serial data preparation. Therefore, we can extend Amdahl's law to the following Enhanced Amdahl's law

$$S'_{\mathrm{EA}}(f_c, c, p_c, f_{\mathrm{ud}}) = \frac{1}{\left((1-f_c) + \frac{f_c}{c}\right) p_c + f_{\mathrm{ud}}(1-p_c)}. \quad (3)$$

## III. Case Studies of Our Enhanced Amdahl's Law Speedup Model

As in Hill & Marty [6], we also assume that a multicore chip of a given area and manufacturing technology is composed of $n$ base core equivalents (BCEs). Assumed that, if we consider only hardware techniques, the improved sequential performance obeys Pollack's rule [7], $\mathrm{perf}(r)$ is approximately proportional to the square root of the increasing size of a chip or number of transistors. For simplicity, we also assume that the performance (cost) of data preparation is also roughly proportional to $\mathrm{perf}(r)$.

### A. Homogeneous Symmetric Multicore

We follow Hill & Marty's [6] definition of a symmetric multicore architecture and assume that a homogeneous symmetric multicore chip uses one single-BCE core with $r$ BCEs to execute sequentially at performance $\mathrm{perf}(r)$, and that it uses all $c = n/r$ cores to execute in parallel at performance $\mathrm{perf}(r) \cdot n/r$. Note that performance is driven by the execution time of a given program on a processor which is only dependent on the technology of the microarchitecture. Therefore, we obtain the following speedup equation for a homogeneous symmetric multicore architecture

$$S_{\mathrm{EA}}^{\mathrm{hs}} = \frac{1}{\left(\frac{1-f_c}{\mathrm{perf}(r)} + \frac{f_c \cdot r}{\mathrm{perf(r)} \cdot n}\right) p_c + \frac{f_{\mathrm{ud}}}{\mathrm{perf}(r)}(1-p_c)}. \quad (4)$$

Fig. 2 assumes a symmetric multicore chip with a maximum number $\mathrm{n} = 256$ of BCEs and $\mathrm{perf}(r) = \sqrt{r}$. The x-axis presents the number of resources (BCEs) to be configured into a core/processor with a total budget of 256 BCEs. For example, if $x = 4$, it means that each single core uses 4 BCEs and that it has a total of 64 cores with 4 BCEs each, for a total budget of 256 BCEs resources. The y-axis shows the speedup compared to the baseline of a single-BCE baseline core. We further assume a parameter($p_c$) with different values for the portion of computation in a program. For instance, $p_c$ can be 0.2, 0.4, 0.6, and 0.8. In addition, it can be overlapped with data preparation ($f_h = 0.2$, 0.4, 0.6, 0.8), and also different values for the parallel fractions in the portion of the computation ($f_c = 0.5$, 0.9, 0.975, 0.99, 0.999). We choose a representative ($p_c = 0.6$, $f_h = 0.8$) from 80 similar curve trends as example in the following subsections.

We performed experiments on the Gem5-GPU simulator [9] which is a standard simulator of heterogeneous systems and tested *backprop, bfs, mum*, and *hs* in the Rodina benchmark [10]. The configuration parameters of simulator is accordant to [11]. We found that the average percentage of sequential load/store operations is about 1/3. Incidentally, we got some

similar supportive evidences from other research results. The floating-point operation(fpop) of $052.alvinn$ which belongs to the CFP92 benchmark in the SPEC suite makes up 27.6% of all the operations, memory operations 48.3%, branch operations 5.1%, while others (total-memory-fpop-branch) make up 18.9% [12]. Thus, the summary percentage of computation is only 27.6%, and the remaining percentage of operations which are relevant to data preparation is about 72.3%. Therefore, we empirically set $k_c$ and $\alpha$ to 1/3 and 2/3 respectively. It means that, in a given application, 1/3 of data preparation cannot be actually executed in parallel, while 2/3 of data preparation can take place in parallel with computation and other data preparation operations.

As shown in Fig. 2, the speedup will vary with the number of BCEs and reaches the maximum 56.71 when $r = 16$ BCEs, and $f_c = 0.999$. This means that almost 60% of the instructions in a program are parallelizable "computing" instructions, while the remaining 32% are data preparation instructions pertaining to memory access, data transfer, *etc.* which can be masked.

### B. Homogeneous Asymmetric Multicore

Just as the example proposed by Hill & Marty [5], a homogeneous asymmetric multicore chip is composed of one large core with $r$ BCEs, and the remaining of other $n - r$ BCEs with the same Instruction set architecture (ISA). Assuming that the serial computing part is executed by one large core and the parallel computing part is executed by the remaining $n - r$ one-BCEs cores and the large core simultaneously, the modified Amdahl's law with consideration of the ODP becomes

$$S_{\mathrm{EA}}^{\mathrm{ha}} = \frac{1}{\left(\frac{1-f_c}{\mathrm{perf(r)}} + \frac{f_c}{\mathrm{perf}(r)+n-r}\right) p_c + \frac{f_{\mathrm{ud}}}{\mathrm{perf}(r)}(1-p_c)}. \quad (5)$$

As shown in Fig. 3, the maximum speedup is 70.12 with $r = 128$ BCEs and $f_c = 0.999$. It is higher than the maximum speedup 56.71 shown in Fig. 2. In general, the homogeneous asymmetric architecture has a potential higher speedup performance. Furthermore, comparing to the homogeneous symmetric architecture, a more powerful core should be built with 128 BCEs to achieve the highest possible speedup in an asymmetric architecture. Further, the maximum speedup with consideration of ODP is almost twice that of a traditional asymmetric architecture. A full length paper will include details.

### C. Homogeneous Dynamic Multicore

Based on the same assumption as that for a dynamic multicore in [5], the serial part of a program can be executed by a large core dynamically combining up to $r$ single-BCEs by utilizing some advanced techniques (*e.g.,* thread-level speculation, helper threads, data prefetching). On the other hand, the parallelizable part is executed by all $n$ base cores with dynamic scheduling techniques such as simultaneous multithreads, dynamic instructions schedules, *etc.* The modified Amdahl's law with consideration for ODP thus becomes

$$S_{\mathrm{EA}}^{\mathrm{hd}} = \frac{1}{\left(\frac{1-f_c}{\mathrm{perf}(r)} + \frac{f_c}{n}\right) p_c + \frac{f_{\mathrm{ud}}}{\mathrm{perf(r)}}(1-p_c)}. \quad (6)$$

Based on equation (6), we find that the highest possible speedup is 106.57 for a homogeneous symmetric and dynamic multicore

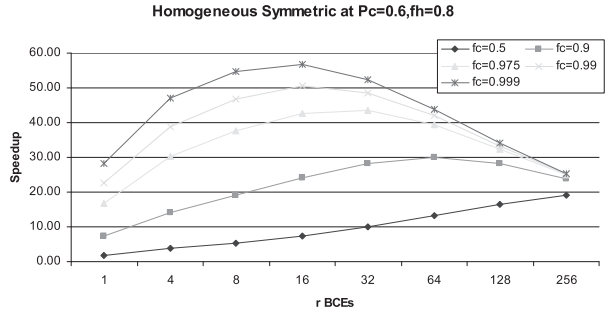


Fig. 2. Speedup distribution of homogeneous symmetric multicore where $p_c = 0.6, f_h = 0.8$.

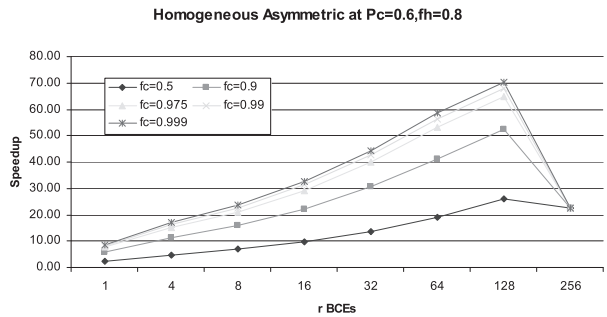


Fig. 3. Speedup distribution of homogeneous asymmetric multicore where $p_c = 0.6, f_h = 0.8$.

architecture where $r = 256$ and $f_c = 0.999$. Because it uses ideal dynamic techniques to increase processor performance, the performance will rise directly with the number of cores. Since the dynamic techniques are assumed to eliminate all the overhead from data hazards, control hazards and resource hazards, without considering the overhead caused by these dynamic techniques, the maximum performance improvements are higher than that in homogeneous symmetric architectures.

### D. Heterogeneous CPU-GPU Multicore

Just as a homogeneous asymmetric multicore,some cores may be more powerful than others. The more powerful cores are built as CPU processors, and the others are built as GPU cores. Note that the assumption of single-BCE is equivalent to a GPU core here rather than CPU core, if we consider the chip area of a single-BCE core, and a powerful CPU processor is composed of $i$ CPU cores where each one comprises $r$ BCEs. We assume that $i$ CPU processors handle all serial computing at performance $i \cdot \mathrm{perf}(r)$. The remaining $n - ir$ BCEs are equivalent to $n - ir$ GPU cores, and all the GPUs cores and CPU processors cooperatively execute the parallel portion of the program. In the same manner we calculate the computational performance, thus we assume that the serial performance of data preparation is proportional to the amount of resources (number of BCEs) on the chip by a factor $\mathrm{perf}(r)$. Consequently, we obtain the speedup equation (7) below

$$S_{\mathrm{EA}}^{\mathrm{cg}} = \frac{1}{\left(\frac{1-f_c}{i\cdot\mathrm{perf}(r)} + \frac{f_c}{i\cdot\mathrm{perf}(r)+n-i\cdot r}\right) p_c + \frac{f_{\mathrm{ud}}}{\mathrm{perf}(r)}(1-p_c)}. \quad (7)$$

As shown in Fig. 4, the highest speedup is 40.58 where $r = 32$ and $f_c = 0.999$. In other words, the highest performance het-

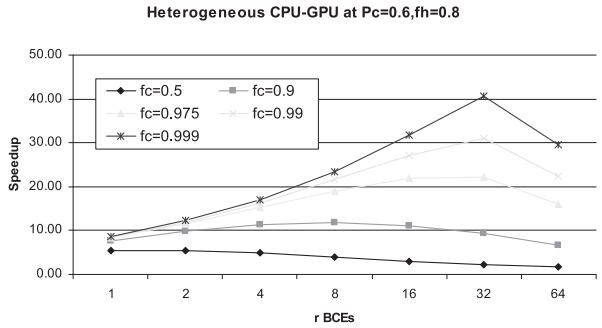


Fig. 4. Speedup distribution of heterogeneous CPU-GPU multicore where $p_c = 0.6, f_h = 0.8, i = 4$.

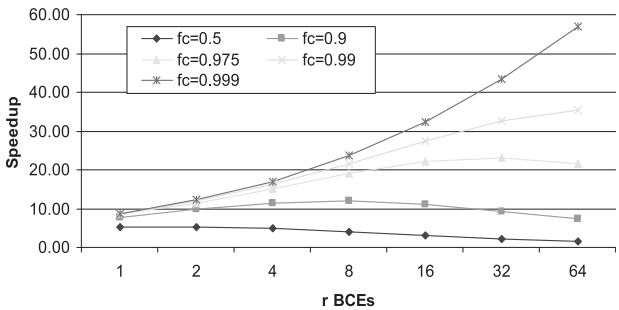


Fig. 5. Speedup distribution of heterogeneous dynamic CPU-GPU Multicore where $p_c = 0.6, f_h = 0.8, i = 4$.

erogeneous CPU-GPU architecture is composed of one CMP with 4 CPU cores, and each CPU core has 32 BCEs. The remaining 128 BCEs are built as 128 GPU cores. However, the highest speedup is less than that of homogeneous asymmetric architecture although the CPU processor also uses 128 BCEs.

### *E. Heterogeneous Dynamic CPU-GPU Multicore*

Similarly to the homogeneous dynamic multicore, we can derive an equation for a heterogeneous dynamic CPU-GPU multicore. While it can also schedule all the computation resources, just as what was assumed in [5], $n$ base cores can also execute in parallel either CPUs or GPUs cores. Further, it should be noted that we also assume the cost of data preparation to be the same as that in the general heterogeneous CPU-GPU model. Therefore, the equation becomes

$$S_{\mathrm{EA}}^{\mathrm{cgd}} = \frac{1}{\left(\frac{1-f_c}{i\cdot \mathrm{perf}(r)} + \frac{f_c}{n}\right) p_c + \frac{f_{\mathrm{ud}}}{\mathrm{perf}(r)}(1-p_c)}. \qquad (8)$$

## IV. Conclusion

When one takes into account the ODP, the speedup will not grow linearly with the number of cores. Furthermore, a heterogeneous CPU-GPU architecture would be more beneficial than a homogeneous architecture according to the comparison curves of speedup distribution.

This is a quantitative analysis strictly based on a theoretical and general model. To some extent, however, we have been able to compare the results with Hill's theoretical approaches for multicore architectures. We would also compare it with other works in a regular paper. Further research will entail investigating the energy efficiency of multicore processors and multithread processors.

## Acknowledgment

The authors would like to thank the anonymous reviewers for their invaluable comments.

## References

[1] H. Che and M. Nguyen, "Amdahl's law for multithreaded multicore processors," *Int. J. Parallel Distrib. Comput.*, vol. 74, no. 10, pp. 3056–3069, Jun. 2014.

[2] P. Rogers, "Heterogeneous system architecture overview," presented at the Hot Chips Symp., Aug. 2013.

[3] A. Marowka, "Extending amdahl's law for heterogeneous computing," in *Proc. Symp. Parallel Distrib. Process. Appl.*, 2012, pp. 309–316.

[4] X. H. Sun and Y. Chen, "Reevaluating amdahl's law in the multicore era," *J. Parallel Distrib. Comp.*, vol. 70, no. 2, pp. 183–188, 2010.

[5] G. M. Amdahl, "Validity of the single-processor approach to achieving large-sclae computing capabilities," in *Proc. Amer. Federation Inf. Process. Soc. (AFIPS)*, 1967, pp. 483–485.

[6] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 12, pp. 24–31, 2008.

[7] M. Daga, A. M. Aji, and W. Feng, "On the efficacy of a fused CPU+GPU processor (or APU) for parallel computing," in *Proc. Symp. Appl. Acc. High-Perf. Comput. (SAAHPC)*, 2011, pp. 141–149.

[8] F. Pollack, "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies," in *Proc. Int. Symp. Microarch. (MICRO)*, 1999, p. 2.

[9] J. Power, J. Hestness, and M. S. Orr *et al.*, "gem5-gpu: A heterogeneous CPU-GPU simulator," *Comput. Arch. Lett.*, vol. 13, no. 1, pp. 1–4, 2014.

[10] S. Che, M. Boyer, and J. Meng *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *Proc. Symp. Workload Characterization*, 2009, pp. 44–54.

[11] S. Pei, M. Kim, and J. Gaudiot *et al.*, "Fusion coherence: Scalable cache for heterogeneous kilo-core system," in *Proc. Annu. Conf. Adv. Comput. Arch.*, 2014, pp. 1–15.

[12] K. M. Dixit, "Overview of the SPEC Benchmarks," in *The Benchmark Handbook*. : Morgan Kaufmann Publishers, 1998, ch. 9.