Amdahl's Law in the Context of Heterogeneous Many-core Systems – A Survey

Mohammed A. Noaman Al-hayanni, *Member, IEEE*, Fei Xia, Ashur Rafiev, Alexander Romanovsky, Rishad Shafik, *Senior Member, IEEE,* Alex Yakovlev, *Fellow, IEEE*

Abstract—For over 50 years, Amdahl's Law has been the hallmark model for reasoning about performance bounds for homogeneous parallel computing resources. As heterogeneous, many-core parallel resources continue to permeate into the modern server and embedded domains, there has been growing interests in promulgating realistic extensions and assumptions in keeping with newer use cases. This paper aims to provide a comprehensive review of the purviews and insights provided by the extensive body of work related to Amdahl's Law to date, focusing on computation speedup. We show that a significant portion of these studies has looked into analyzing the scalability of the model considering both workload and system heterogeneity in real-world applications. The focus has been to improve the definition and semantic power of the two key parameters in the original model: the parallel fraction (f) and the computation capability improvement index (n). More recently, researchers have shown normal-form and multi-fraction extensions that can account for wider ranges of heterogeneity, validated on many-core systems running realistic workloads. Speedup models from Amdahl's Law onwards have seen a wide range of uses such as the optimization of system execution, and these uses are even more important with the advent of the heterogeneous many-core era.

Index Terms—

1 Introduction

Parallelization has been an essential method for improving the performance and energy efficiency of computation. On performance, it is well known that the parallelization of workloads may bring speedup [1]–[5]. On energy efficiency, with the advent of such hardware techniques as dynamic voltage and frequency scaling (DVFS), it is possible to trade an increase of the number of processing units for a reduction of energy consumption without affecting performance [6]–[9].

The scaling of CMOS electronics has persisted for decades, resulting in more and more hardware capabilities being integrated onto single chips [10], [11]. For both performance and energy efficiency reasons, systems of relatively small physical size, i.e., those integrated onto single chips, have in general moved towards multi-/many-core processors (M/MCP) from single-core structures, with the trend predicted to further develop [12]. In general, M/MCP may be configured into homogeneous MCP (HoMCP), where all of the cores are of the same type, as seen in Intel Core-i and Xeon processors [13]–[16], or heterogeneous MCP (HeMCP), where the cores are different, as seen in ARM big.LITTLE [3]–[5], [17], [18]. HeMCP may incorporate diverse architectures of processing units such as CPU, GPU, DSP and embedded FPGA, as well as complex cache memory and communication facilities [5], [19]–[21].

- *M. Al-hayanni, F. Xia, A. Rafiev, A. Romanovsky, R. Shafik, and A. Yakovlev are with Newcastle University, UK*
  E-mail: {m.a.n.al-hayanni, fei.xia, ashur.rafiev, alexander.romanovsky, rishad.shafik, alex.yakovlev}@ncl.ac.uk
- *M. Al-hayanni is also with University of Technology-Iraq*
  E-mail: {110093@uotechnology.edu.iq}
- *This work is supported by EPSRC/UK as a part of PRiME EP/K034448/1 and STRATA EP/N023641/1 projects.*

An extensive body of work has been concentrated on the modelling and analysis of the effects of computational parallelization. The property of speedup is a popular topic in these studies [1]–[5], [13], [19], [22]–[24], whilst energy and power have not been neglected [3], [5], [25]–[30]. Various problems related to mapping parallelizable (mostly software) workloads onto M/MCP hardware platforms have also been intensely scrutinized. The energy efficient load balancing, task migration and scheduling have been the target of substantial investigations [5], [31]–[34].

This paper reviews the literature on workload parallelization in the context of M/MCP speedup, particularly those related to Amdahl-type speedup models such as Amdahl [1], Gustafson [22] and Sun-Ni [2]. Non-Amdahl style speedup models relating to the concept of parallelism will also be comparatively studied [2], [13], [35], [36] and new research on extending Amdahl's Law to cover non-zero and non-infinity parallelism will be highlighted [19], [23], [24], [37], [38]. The review especially covers model extensions dealing with HeMCP with different degrees of heterogeneity [4], [5], [24], [38].

The main topic of this review is the property of speedup and its relatiohship with system improvements/enhancements in the sense of increasing the number of processing units (cores). Energy/power models for these kinds of systems related to speedup models and making use of speedup and energy/power models for system optimization will only be touched upon [15], [27], [30], [36], [39] but not treated as main topics of discussion. Other related topics such as the roofline modelling method [40], modelling speedup caused by improvement techniques not related to parallelization [30], and using Amdahl's Law to model the improvements of parameters other than speedup [41] are likewise not discussed in detail. The publications reviewed in this paper are classified in a table

which highlights the topics covered by each publication aiming to form a taxonomic view of the literature (Table 1, Section 10).

A comprehensive survey of Amdahl's Law and its extensions can be found in [42]. The authors of [42] strive to cover as many research titles as possible and discuss the contributions of each work in as much detail as possible. As a result, it is a good resource for finding related work in this area of research. However, as is the case of many survey papers, [42] skips over technical details. A reader may struggle to build a complete mathematical picture of model evolution without reading the cited publications. Developments since 2013 are also inevitably missing from that title.

Taking a different aim, in this work we focus on telling the technical story logically, by including the fundamental mathematics and their relevant explanations and derivation processes. In doing so this paper attempts to build a picture of how the models mathematically evolve and how each stage of the development relates to other stages. This should help a reader build a starting knowledge of the theories and practices of the field without leaving the paper too often. As a result, we do not have space to re-list the contributions of each paper, which can usually be found in a paper's abstract and conclusion sections. On the other hand, at the end of each section, the contributions of selected publications which are on-topic for that section are highlighted. We believe that these highlighted developments represent important points in the story of model evolution in the Amdahl's Law space.

The rest of the paper is organized as follows. Section 2 sets the scene by describing Amdahl's Law and Gustafson's model of speedup. Section 3 discusses model extensions which concentrate on the effects of non-processing costs, focusing on Li-Malek's and Sun-Ni's models. Section 4 focuses on the initial attempts at extending Amdahl's Law to cover system heterogeneity, with Hill-Marty's model in the highlight. Section 5 explores the reality of heterogeneous multi-core systems and their differences from Hill-Marty's assumptions. Section 6 presents the normal-form assumption of core heterogeneity and the resulting speedup models. Section 7 explains the relationship and differences between parallelism and parallel fraction, and the limitations of Amdahl's Law in studying workloads with non-infinity parallelism. Section 8 describes efforts on and results from extending Amdahl's Law so that heterogeneity in both workload parallelism and system core arrangements are covered, culminating in the normal-form multi-fraction speedup model. Section 9 discusses the progression of speedup model development and highlights the salient points in both the modelling and model usage. Section 10 concludes the paper and presents a classification table which taxonomizes the reviewed research.

In this paper, the mathematical forms including variable names and formulas follow a consistent presentation standard. Formulas that are derived, adopted, or adapted from literature may have been transformed to conform with this standard presentation.

2 Amdahl's Law and Gustafson's model

The classical method for modelling the *speedup* of workload processing caused by some measure of improving the computation capabilities is known as Amdahl's Law, which developed from observations presented by G. Amdahl in 1967 [1]. Amdahl did not provide a mathematical formula for this law, which was later formulated based on his verbal arguments. Given the context of this paper, which is about the parallelization of workloads on M/MCP systems, "*improvement of computation capabilities*" generally means the incorporation of multiple processing units (to be called "*cores*" in this paper) to improve the speed of workload execution, unless otherwise noted. The fundamental assumption of an Amdahl-type workload (also known as "program" or "job" [4], [35] – we use "*workload*" in this paper) is that it can be divided into a fully sequential (non-parallelizable, i.e., not affected by the improvement) part and a fully parallel (infinitely parallelizable, i.e., fully affected by the improvement) part, in the following way:

$$T_{\mathrm{w}} = T_{\mathrm{s}} + T_{\mathrm{p}}, \tag{1}$$

where T_{w} is the time consumed to execute the entire workload, T_{s} is the time taken to execute the sequential part and T_{p} is the time taken to execute the parallel part, in all three cases on a single core (before improvement). Amdahl's Law proceeds to analyze the maximum speedup that can be achieved by running such a workload on n cores (after improvement), with $n > 1$ being an integer. The most commonly seen form of Amdahl's Law focuses on the fraction of the workload execution time that is taken by the parallel part:

$$f = \frac{T_{\mathrm{p}}}{T_{\mathrm{w}}} = \frac{T_{\mathrm{p}}}{T_{\mathrm{s}} + T_{\mathrm{p}}}, \tag{2}$$

where f ($0 \leq f \leq 1$) is variously known as "parallel fraction", "parallelization factor", etc. In this paper we call it the "*p-fraction*" and use the variable name f exclusively for it. The p-fraction pertains to execution time and not numbers of instructions.

Speedup is defined as the ratio between the execution speed after improvement (expanding to n cores) and the original execution speed before improvement (running on a single core). In other words, speedup as a result of expanding to n cores is

$$S(n) = \frac{T_1}{T_n} = \frac{T_{\mathrm{s}} + T_{\mathrm{p}}}{T_{\mathrm{s}} + \frac{T_{\mathrm{p}}}{n}}, \tag{3}$$

where $T_1 = T_{\mathrm{w}}$ is the time taken to execute the entire workload on one core, and T_n is the time taken to execute the same workload on n cores. Speedup caused by an improvement is therefore the time taken by the unimproved system divided by the time taken by the improved system. Improving by expanding to n cores causes the time taken by the parallel part to shrink by n times, without affecting the time taken by the sequential part. By combining (2) and (3), Amdahl's Law in terms of the p-fraction is obtained as

$$S(n) = \frac{1}{(1-f) + \frac{f}{n}}. \quad (4)$$

An alternative form of Amdahl's Law (e.g. as found in [13]) is related to the ratio between the parallel and sequential parts of the workload:

$$\beta = \frac{T_{\mathrm{p}}}{T_{\mathrm{s}}} = \frac{f}{1-f}, \quad (5)$$

where β ($0 \leq \beta \leq \infty$) is a real number describing how large the parallel part is relative to the sequential part. Amdahl's Law, in terms of β, is then

$$S(n) = \frac{(1+\beta) \cdot n}{n+\beta}. \quad (6)$$

There are also other forms of Amdahl's Law in the literature, but they are all mathematically equivalent. This paper uses the p-fraction form of Amdahl's Law (4), which is the most commonly seen form in the literature.

A more general understanding of Amdahl's Law decouples it from parallelization and the use of multiple cores [4]. It can describe speedup from any improvement on computational capabilities, e.g. the use of an accelerator of some kind, or increasing the operating frequency of the hardware [41], [43]. A workload is divided into an infinitely improvable part and an non-improvable part, in response to the particular improvement. The variable n may be a real number [44], or even a function, describing the degree of improvement to the computation capabilities. In this context, it is best known as the "*computation capability improvement index*". The relevant equations remain unchanged and the meanings of f and β remain the same.

J. Gustafson presented a substantial modification to Amdahl's Law in 1988 [22]. It is argued that a fundamental assumption of Amdahl's Law, that a workload is a job or program of fixed size, no matter whether the computation facilities are improved or not, may be overly restrictive. Gustafson's model deals with the case where the unimproved part of the workload stays the same, whilst the improved part of the workload scales linearly with the improvement. An example of such a case can be found in modern data centres where the synchronized booting up of computers takes roughly the same amount of time regardless of the number of computers involved, but the more computers there are, the more processing jobs will be mapped onto them once they are up and running. Basically, systems with more cores tend to be used to solve larger problems.

Maximum speedup is then the ratio between the amount of workload that can be executed on an improved (n-core) system and the amount of workload that can be executed on an unimproved (single core) system, if both are given the same amount of execution time. In terms of the p-fraction, Gustafson's model is therefore

$$S(n) = (1-f) + n \cdot f, \quad (7)$$

which says that the unimproved (non-parallelizable) part is unchanged whilst the improved (fully parallelizable) part scales linearly with the rate of improvement n.

At a philosophical level, Amdahl's Law, because of its saturation of speedup if $f \neq 1$ even when $n = \infty$, was used by some, including Amdahl himself, to argue in favour of single processors [1]. In that context, Gustafson's model was intended as a counter-argument to show that parallel processing was indeed highly relevant [22]. Viewed from a more contemporary perspective, when there is no longer any question on the relevance of parallelization, both models are relevant according to the workload realities of any particular scenario [4].

Summary: Amdahl's intuitive reasoning [1] about the speedup of computation because of hardware improvements is formulated into Amdahl's Law. Gustafson [22] shows that speedup does not have to saturate if the workload consists of a serial part of constant size and a parallel part whose size scales with the hardware improvement.

3 Non-processing elements and overheads

Both Amdahl's Law and Gustafson's model consider processing only. And they are understood to estimate the maximum speedup as ideal-case models, with any effects from non-processing elements, especially overheads of all kinds, not taken into consideration. However, for real-world application, there has always been a need for representing such issues as non-zero overheads, architectural and workload diversity, and non-processing element influences. Although it may be argued that the p-fraction may be viewed as capable of including the effects of non-processing activities such as communication and memory access to some extent, the quantitative relationship is far from straightforward. A single number parameter such as f may become semantically too weak to properly represent the complex effects of communications between different parts of a workload and accessing memory which may require synchronization and waiting in shared memory architectures such as most modern off-the-shelf processors.

For instance, the "memory wall" characterizes the effects of the memory-processor performance gap on the entire system. The less than ideal situation in memory latency and bandwidth, among other factors, limits the processor's capability of accessing instructions and data. In this case, the processor will stall waiting on memory in order to continue computation. This issue becomes more complicated with M/MCP and shared memory, but networks are not immune either [45]–[48].

Communication overheads are the effects of communication on the total performance of M/MCP [49]–[51]. Synchronization overheads are the effects on performance of the joining and handshaking of multiple processes and data in M/MCP [49], [51], [52].

In general, many of these effects can be called overheads as they impede the system's capability of realizing the maximum speedup predicted by Amdahl's Law and Gustafson's model. Various attempts have been made to extend these models to cover overheads.

In 1988, the same year in which Gustafson proposed his speedup model in [22], X. Li and M. Malek

specifically incorporated communication time in their extension of Amdahl's Law [53]. The parallel part of the workload is regarded as having distinct computation and communication times. The communication time is the time required for the inter-core communication necessitated by mapping a workload onto multiple cores. This is described as

$$T_n = T_\mathrm{s} + \frac{T_\mathrm{p}}{n} + T_\mathrm{c}, \tag{8}$$

where T_c is the additional communication time which does not scale with parallelization, which is considered an additional overhead on the parallel part. Speedup is then described by

$$S(n) = \frac{T_\mathrm{s} + T_\mathrm{p}}{T_\mathrm{s} + \frac{T_\mathrm{p}}{n} + T_\mathrm{c}} = \frac{1}{(1-f) + \frac{f}{n} + \frac{T_\mathrm{c}}{T_\mathrm{w}}}, \tag{9}$$

where $T_\mathrm{c}/T_\mathrm{w}$ is known as the communication/computation ratio (CC-ratio) of a particular workload in [53]. The CC-ratio is zero when $n = 1$, as $T_\mathrm{c} = 0$ on a single core. This gives $S(1) = 1$ as all other speedup models. The paper continues to expand into the realm of statistical models. Eventually the relationship between the upper and lower bounds of speedup and the statistical distributions of all elements of inter-core communications was established analytically.

This technique, using what may be regarded as a penalty term in the denominator of Amdahl's Law (the CC-ratio in case of the Li-Malek model), has been used on multiple occasions by different researchers, to cover additive overheads of all types [54, (p.167)][23, (p.42)][48], [55], [56]. However, overheads may be caused by very complicated effects of multiple factors, which leads this line of modelling to become more and more sophisticated, expanding the penalty term from a constant to various different functions.

From 1990 [2] onwards, Y. Sun, L. Ni and colleagues produced a body of research leading to speedup models that extend those of Amdahl and Gustafson by incorporating the requirements for memory, communications and other services [2], [57]–[59] by introducing an additional function of n as well as other coefficients.

The fundamental assumption leading to Sun-Ni's model is that the speedup is memory-bounded, unlike for Gustafson's model where speedup is bounded by the number of cores. Basically, with Gustafson's model, a system with more cores is used to solve larger problems and the increase of problem size corresponds to the number of cores n. However with Sun-Ni's model, the size of these larger problems would be limited by memory and not the number of cores. In other words, as computing power increases, the corresponding increase of problem size is constrained by memory. This reasoning leads to the following speedup model:

$$S(n) = \frac{(1-f) + f \cdot g(n)}{(1-f) + \frac{f \cdot g(n)}{n}}, \tag{10}$$

where $g(n)$ is a function representing the memory bound of problem size increase. This function also takes into account the relationship between total required memory and the number of cores n, as the amount of required memory is assumed to depend on the number of cores n. A typical example problem found in the literature in the context of the memory bound function is matrix multiplication. The memory requirement of multiplying two $N \times N$ matrices is proportional to N^2, and the amount of computation is proportional to N^3, which gives $g(n) = n^{3/2}$, and following (10) the memory-bounded speedup is

$$S(n) = \frac{(1-f) + f \cdot n^{3/2}}{(1-f) + f \cdot n^{1/2}}. \tag{11}$$

The function g is more semantically powerful than a number, and can incorporate the effects of both memory and communications as well as such issues as the parallelism of the workload itself [2], [35]. This last point will be discussed further in a later section.

Adding a similar sort of penalty coefficient function to the parallel part in the denominator of Amdahl's Law was also proposed for representing the effects of synchronization [60].

With $g(n) = 1$, Sun-Ni's model reduces to Amdahl's Law. In other words, $g(n) = 1$ indicates that workload does not increase with n. With $g(n) = n$, Sun-Ni's model reduces to Gustafson's model. In other words, with $g(n) = n$, the required memory size is the same as the number of cores and the workload can also be said to be core-bounded. Both earlier models are therefore special cases of Sun-Ni's model [3].

Fig. 1 compares Amdahl's Law and Gustafson's and Sun-Ni's models. It can be observed that Amdahl's Law saturates as the number of cores n increases, Gustafson's model scales linearly with n and Sun-Ni's model has super-linear scaling, with n and $0 < f < 1$, for $g(n) = n^{3/2}$. Further development of Sun-Ni's model led to the incorporation of more parameters to better represent different scenarios [58], [59]. With certain forms of g, Sun-Ni type models may show a reduction of speedup after n goes beyond some optimal value. This represents the superlinear scaling of overheads [60].

Summary: Overheads such as communication and memory access costs are recognized as factors affecting the speedup of the parallel part of the workload. Modelling methods to represent such costs include adding a penalty term to the denominator, as exemplified by the Li-Malek model [53], and adding coefficient functions to the parallel execution time in the Sun-Ni models [57], [59].

4 Core heterogeneity, first attempts

Amdahl's Law has a very simple mathematical form, which brings with limited representation power, because it has only two parameters f and n with which differences in system characteristics may be expressed. On the side of workload characteristics, the limited semantic power of the p-fraction in the form of a single number f makes it difficult to represent the effects of non-processing factors such as memory and communication requirements in speedup models. Li-Malek [53] and Sun-Ni [2] deal with this problem by using the CC-ratio and memory bound function $g(n)$ to modify Amdahl's and Gustafson's models. On the side

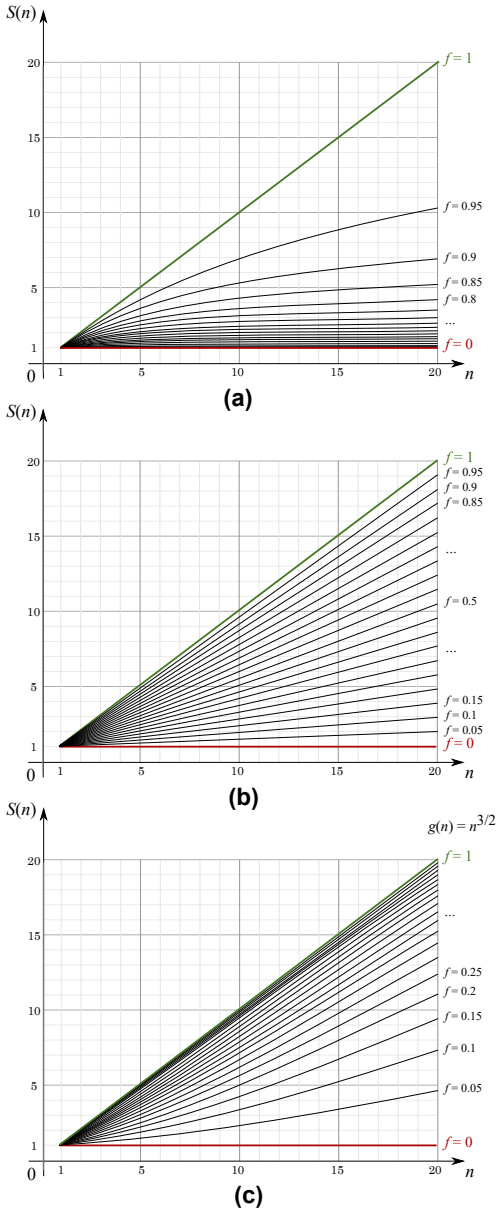

Fig. 1: Speedup vs the number of cores for (a) Amdahl's Law, (b) Gustafson's model and (c) Sun-Ni's model with $g(n) = n^{3/2}$. Figure adapted from [7].

of system improvement, the simple integer n also suffers a similar lack of semantic power when a system may consist of cores of different types, which has become increasingly common. Early attempts were made to address this issue by extending Amdahl's Law to cover simple cases of system heterogeneity in the 1990s [61].

In 2008, M. Hill and M. Marty presented a method of deriving speedup models for systems with certain types of core heterogeneity [4]. They complement Amdahl's Law, which originally can be said to be focused on software, with a corollary of a simple model of multi-core hardware chip. The hardware assumption is centred around the concept of a "base core equivalent" (BCE), which may be understood as a basic core capable of performing the workload in question with a performance of 1 (unit performance). A chip of given size and technology generation is assumed to be able to contain at most n BCEs. Multiple BCEs may also be organized together into a larger core, with a higher sequential processing performance than a single BCE. In other words, it is assumed to be possible to re-organize r ($1 \leq r \leq n$) BCEs into one more powerful core, with a performance of $perf(r)$. The case of $perf(r) \geq r$ is uninteresting because if that is the case, there is no point organizing the chip into BCEs smaller than that large core. However, for $perf(r) < r$, the Hill-Marty heterogeneous multi-core models describe the trade-offs in core organization with regard to workloads.

For instance, it was argued that doubling sequential performance requires a quadrupling of silicon [12]. In this case, $perf(r) = \sqrt{x}$, or needing to organize a large core out of four BCEs for a doubling of performance. Cases like this are covered by Hill-Marty's models.

Hill and Marty stipulate that clustering multiple BCEs into larger cores may result in two types of static organizations, called symmetric and asymmetric. Symmetric is defined as all cores on the chip being the same, i.e. of the same size r ($1 \leq r \leq n$). For full chip area utilization, n must be divisible by r. The asymmetric case has a single big core of size r with the rest of the chip organized into $n - r$ single-BCE cores. This is shown in Fig. 2.

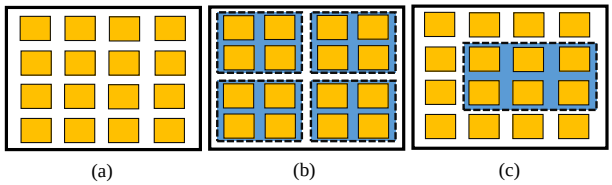

Fig. 2: Hill-Marty M/MCP diversity. (a) Symmetric Multi-Core Processor (SMCP) with 16 single-BCE cores, (b) SMCP with 4 four-BCE cores, and (c) Asymmetric Multi-Core Processor (AMCP) with one six-BCE core and 10 single-BCE cores. Figure adapted from [4].

In the context of Hill-Marty HeMCP, speedup is relative to the performance of a single BCE, which is 1. The speedup achievable by executing a workload on a symmetric chip with a core size of r follows Amdahl's Law and is

$$S_{\mathrm{s}}(n, r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f \cdot r}{perf(r) \cdot n}}. \tag{12}$$

The workload is assumed to be Amdahl's type, of fixed size, with a sequential part and a fully parallelizable part. The sequential part is executed on a single r-sized core whose performance is $perf(r)$ times that of a single BCE. The parallel part is executed on all n/r cores with a total performance of $perf(r) \cdot n/r$.

For the asymmetric case, it is natural to expect that the sequential part should be executed on the single larger core of size r, and the parallel part will be executed on all cores. This strategy is a form of "*favouring faster cores*" scheduling, and it assumes that the large core is faster than a BCE. Otherwise there is no point in assembling a large core out of multiple BCEs. The speedup is therefore

$$S_{\mathrm{a}}(n, r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{perf(r)+n-r}}. \tag{13}$$

This is similar to (12) except that the parallel part is executed at a performance of $perf(r)$ on the single larger

core and a performance of 1 on the $n - r$ BCEs. If $r = 1$, $perf(r) = 1$, these equations reduce to Amdahl's Law (4).

Hill and Marty also propose that the HeMCP may have a dynamic organization, with BCEs being combined into larger r-sized cores which can also be disbanded back to BCEs, both during run-time. This dynamic HeMCP regime, when faced with an Amdahl's type workload, works best if the entire chip is combined into a single large core, with $r = n$, to execute the sequential part, and disbanded into n single-BCE cores to execute the parallel part, given $perf(r) < r$ for all $r > 1$. This scheme is shown in Fig. 3.

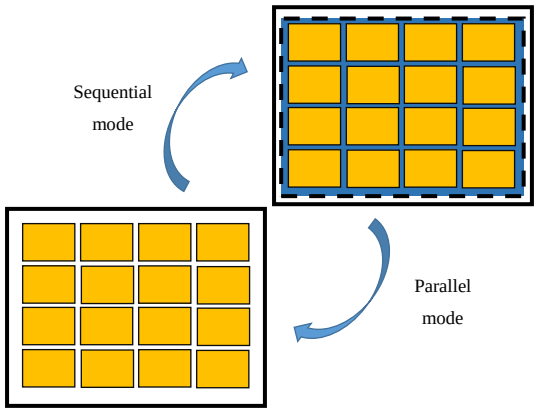

Fig. 3: Dynamic Multi-core chip with 16 one-BCE cores. Figure adapted from [4].

The speedup achievable from such a dynamic HeMCP is

$$S_{\mathrm{d}} = \frac{1}{\frac{1}{perf(r)} + \frac{f}{n}}, \tag{14}$$

with best speedup at $r = n$.

Hill-Marty's model extends Amdahl's Law, and by inference potentially Gustafson's and Sun-Ni's models, into system core heterogeneity and, similar to Sun-Ni's and Li-Malek's efforts, improves the power of direct representation of the speedup formulas by incorporating additional terms and/or functions. The difference is that whilst Li, Malek, Sun and Ni seek to improve the semantical power of the p-fraction f to include non-core factors, Hill and Marty try to enhance the semantical power of the computation capability improvement index n to include core heterogeneity.

Further research has sought to strengthen the theoretical understanding of the Hill-Marty model [42], [62], [63], and to extend it to cover Gustafson's [64] and Sun-Ni's type of workloads [60], [65]. Blem et al. conducted more sophisticated research, based on the Hill-Marty symmetric, asymmetric and dynamic taxonomy, to develop multi-core speedup models as functions of first-order single-core characteristics [66]. Overheads from such actions as memory access, on-chip communications and synchronization among cores have been a powerful motivation for extending Hill-Marty's model [55], [56], [67], [68], with such novel architectures as networks on chip being included in the consideration [69], [70]. The Hill-Marty model has also been extended to cover other properties than speedup, for example power dissipation [25].

Summary: Hill and Marty recognize the limitations posed by one of the fundamental assumptions of Amdahl's Law, as used in the context of scaling with multiple cores, that the hardware consists of multiples of processing units of the same type. The Hill-Marty models [4], developed to extend Amdahl's Law to cover a few types of core heterogeneity, inspire a large body of research in the modelling of speedup, power, energy and other non-functional properties of such heterogeneous systems.

5 Heterogeneous Multi-core Reality

Hill and Marty's speculation on HeMCP architecture has not been borne out by commercial reality, and, a decade later today, most current off-the-shelf HeMCP systems have little in common with the Hill-Marty asymmetric and dynamic structures [4]. FPGA [71] may be the best current technology for implementing the closest approximations to these architectures. However, setting up a BCE, distributing the right number of copies of this BCE across a chip, and organizing exact integer multiples of a BCE's area into a larger core are non-trivial even for FPGA. It puts unnecessary restrictions on chip configuration with possible wasting of interconnect, memory and other microarchitecture elements. Run-time, large-scale reconfiguration needed for making Hill-Marty dynamic HeMCP useful is far from realistic even after decades of focused academic research and industrial development on FPGA reconfiguration [71].

As continued technology scaling [10], [11] causes ever increasing M/MCP complexity, two types of M/MCPs have emerged: homogeneous (HoMCP) and heterogeneous (HeMCP) [4], [18], [62].

HoMCP systems incorporate multiple cores that are essentially the same as one another, organized in the symmetrical way in Hill-Marty's models [4], [17], [18]. In this type all the cores have identical performance and instruction set architecture (ISA) [72], [73].

One example HoMCP architecture is found in GPUs which are designed as special purpose processors for visual processing [74]. Modern GPUs may incorporate hundreds of cores, which are carbon copies of one another, in order to achieve parallel processing by handling thousands of threads simultaneously [75]. Classical Amdahl's Law and related models based on a simple core number n are sufficient for HoMCP systems in general, as existing HoMCP systems all have $r = 1$, i.e. their cores may be viewed as BCEs.

In contrast, an HeMCP system incorporates a number of different cores that may have different architectures. These include full-blown latency oriented cores for sequential processing, reduced-complexity cores for low-power modes, massively parallel accelerators such as VPUs or GPUs, DSPs, embedded FPGAs, media accelerators, and ASICs [19], [20].

The simplest style of HeMCP is an extension of Hill-Marty's asymmetric structure where the system includes two types of different cores, but both types may have multiple units beyond the relatively narrow scope of Hill-Marty [4], [17], [18], [76]. In this case, not all the cores have the same performance and may have a single ISA or more than one ISA [72], [77]–[79]. This type of core heterogeneity may provide an ability to manage the performance/power trade-off or some other similar trade-off. For instance, the

big.LITTLE technology from ARM is HeMCP incorporating a cluster of "big" cores for high performance and a cluster of "LITTLE" cores for low power consumption, likely with the same ISA [76], [78].

M/MCP architectures may be implemented on single chips (the Hill-Marty assumption) or form distributed structures with multiple cores connected through communications facilities such as networks [68]. In both cases, HeMCP could include different types of CPUs or CPU-GPU as shown in Fig. 4. The single-chip CPU-GPU integration offers performance improvements [28], [80]. Further advantages include reduced communication overheads and costs, and specially designed shared memory for avoiding explicit data copying [81]. They may also deliver more power and energy efficient computing [28], [80], [82].

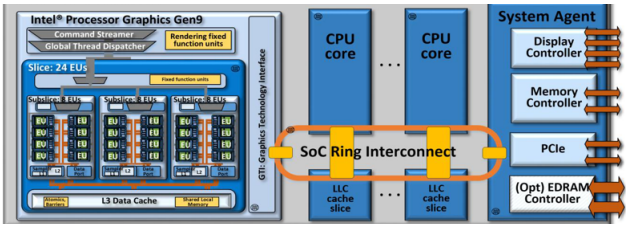

Fig. 4: Architecture of a Central Processing Unit-Graphics Processing Unit (CPU-GPU) chip. Figure from [83].

Recently, it has been claimed in the literature that optimization targeting certain applications have resulted in performance speedup from 25x to 100x or more utilizing GPUs instead of CPUs [84]. The main reason of this comes from the differences of the architecture of each. CPUs and GPUs are designed in order to execute different types of applications [85], [86]. These differences allows CPUs to achieve better performance on latency-sensitive applications which need to respond rapidly to specific events and partially parallel applications [85]–[87]. On the other hand, GPUs achieve better performance with latency-tolerant but throughput-critical applications, and the processor utilization may be high due to multi-threading [88], [89], highly parallel applications and independent applications [84]. There has been a rapid increase of using GPUs for general-purpose processing unrelated to graphics or video applications, in collaboration with the CPUs with which they are either integrated on the same chips or connected through close off-chip links [90].

In addition, other heterogeneous architectures combine unconventional "cores" such as custom logic, FPGA, and/or pipelining (including hyperthreading) in the traditional M/MCP in order to achieve superior energy efficiency and performance improvements [44], [80], [86]. This new paradigm considers the relationships between a conventional processor and a varied set of unconventional cores. It forecasts future architectures from scaling developments predicted by the International Technology Roadmap for Semiconductors (ITRS) [91]. All of this argues for speedup and other models studying the execution of the same workloads on an HeMCP with cores that differ from one another not only performance-wise, but also ISA-wise.

None of these types of architectures are directly covered by Hill-Marty's models.

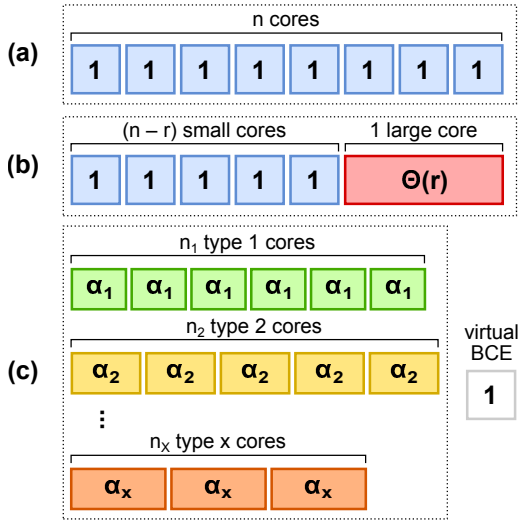

Fig. 5: Normal form of HeMCP (c) compared to HoMCP (a) and Hill-Marty's assumption [4] on heterogeneity (b). The numbers in the core boxes denote the equivalent number of BCEs. Figure adapted from [5].

Summary: The Hill-Marty core heterogeneity assumptions are inappropriate for most modern heterogeneous multi-core architectures.

6 Normal Form of Core Heterogeneity

Since 2016 [3], M. Al-hayanni et al. have engaged in extending Amdahl's Law and Gustafson's and Sun-Ni's models into HeMCPs with more general assumptions of HeMCP architectures. The aim is to cover as many current HeMCP architectures as possible directly. To that end, a "*normal form*" of HeMCP has emerged [5].

By further expanding the computation capability improvement index n by characterizing cores with a vector, the normal form extends the direct representation of M/MCP heterogeneity to include a number of different types of cores, each having a number of members. The fundamental assumption is that an HeMCP consists of x clusters (types) of cores – within each type the cores are identical (See Fig. 5(c)). The numbers of cores of the different types are then defined as a vector $\overline{n} = (n_1, n_2, ..., n_x)$, and the total number of cores is denoted as $n = \sum_{i=1}^{x} n_i$.

The performance of each core of type i is defined as α_i, relative to some BCE whose performance is regarded as 1, similar to Hill-Marty [4], and the vector $\overline{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_x)$ describes the performance of individual cores of all x types. In other words, for all $1 < i < x$, $perf(i) = \alpha_i$. The $\overline{\alpha}$ vector is therefore an extension of the $perf(r)$ method in Hill-Marty's models. This is preferable to directly making the improvement index n itself a real number, the technique used to model hyperthreading speedup in [44]. Leaving $\overline{n}$ as integers the cores remain countable.

The issue of workload distribution was not investigated in earlier models. This is partly because for HoMCP the parallel part of the workload is evenly distributed to all cores by default. Hill and Marty, however, did not explore this issue for even the asymmetric HeMCP, but effectively assumed that the workload is distributed to all cores, large and small, in such a way that they all complete

their execution at the same time [4]. This convention has been maintained without discussion by most other research following Hill and Marty.

For the normal-form model, the authors of [5] made no such assumption but investigated the quality and impact of workload distribution. It is assumed, as usual, that one of the cores is responsible for executing the sequential part of the workload, and the parallel part of the workload is distributed to all cores. The execution time for the parallel part, and therefore the speedup, depend on the distribution policy for the parallel workload. In the normal-form model, the variable N_α denotes the overall equivalent computation capability improvement index, serving the purpose of n in the HoMCP models. N_α describes the performance improvement of the parallel part of the workload, given a particular normal-form HeMCP architecture and a particular parallel workload distribution.

It is expected that legacy software (including system software), made with HoMCP in mind, by default would attempt to distribute any parallel workload equally among available cores. This causes faster cores to wait for the slowest core, as illustrated in Fig. 6(a) – a very inefficient workload distribution [5], [60]. In this case, N_α is calculated from the minimum of $\overline{\alpha}$:

$$N_\alpha = N \cdot \min_{i=1}^{x} \alpha_i, \tag{15}$$

where $N = n_1 + n_2 + ... + n_x$ is the total number of cores of all types. Equation (15) corresponds to the naïve equal-share distribution policy with no balancing. It says that with an equal-sharing of the workload across all cores, the system behaves as if it had n cores of the slowest type, in terms of speedup. In the case of Fig. 6(a), by giving all three cores an equal workload of 13, the system behaves in the same way as one with three cores, each of which having $\alpha = \alpha_2 = 3$. The faster cores 1 and 3 have to wait after they've completed their shares of workload.

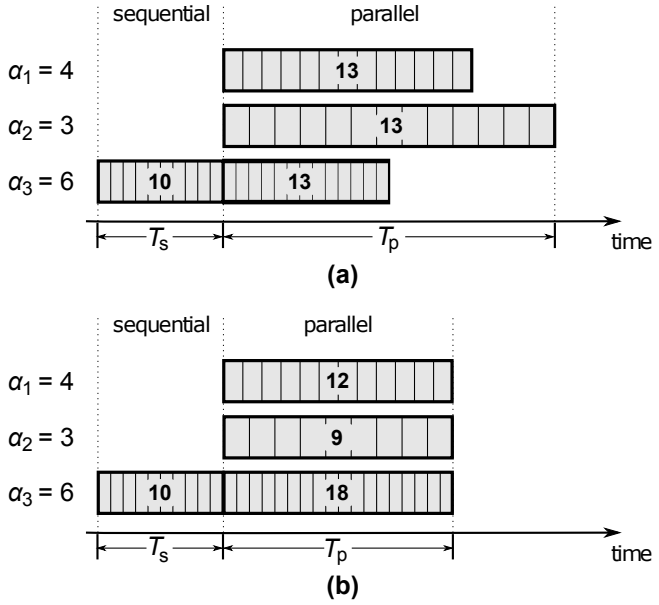

Fig. 6: Workload distribution examples following (a) equal-share model and (b) balanced model. Figure adopted from [5].

Fig. 6(b) shows the ideal case of parallel workload distribution, with workloads assigned to cores in inverse proportion to their α values [5], [60]. This method theoretically may achieve zero waiting time, with all cores finishing at the same time. N_α for this fully balanced workload distribution is

$$N_\alpha = \sum_{i=1}^{x} \alpha_i n_i. \tag{16}$$

This is what Hill-Marty's asymmetric model assumes. This ideal case almost never happens, as the α values are real numbers and workloads generally cannot be divided into infinitesimally small parts, and any dependency across threads would cause the critical path to be irreducible after some point [23, p. 42]. Some amount of waiting is therefore almost inevitable, even if such system software as load balancers may support the run-time redistribution of partially executed threads across cores, through such techniques as task (thread) migration.

Task migration is the transfer of partially executed tasks from a core or cluster of cores to another core or cluster of cores in M/MCP [31]–[34]. Task migration may be used to migrate a task from heavily loaded cores to lightly loaded or idle cores in M/MCP, in particular for HeMCP, in order to balance the load across all cores. Thus, the utilization of the cores are improved and core waiting minimized [31], [34]. On the other hand, in spite of extensive research in this area, experiments have shown that the load balancers found in off-the-shelf commercial HeMCP systems, which are designed to target the characteristics of their specific HeMCPs, never achieve ideal workload distribution in the sense of (16) and sometimes return even worse results than equal-share distribution [5].

The normal-form model does not require one of the existing core types in the system to be equal to one BCE, but speedup is still relative to one BCE which is defined as having a performance of 1, following the convention of Hill-Marty. In other words, for an Amdahl's type workload, $T_\text{w} = T_1 = 1$. Assuming one core of type s, $1 \leq s \leq x$, is used to run the sequential part, the execution time on all n cores is

$$T_{\overline{n}} = \frac{1-f}{\alpha_s} + \frac{f}{N_\alpha}, \tag{17}$$

where α_s is the performance of one core of type s. The speedup can then be calculated as

$$S(\overline{n}) = \frac{1}{T_{\overline{n}}} = \frac{1}{\frac{1-f}{\alpha_s} + \frac{f}{N_\alpha}}. \tag{18}$$

It can be seen that to maximize speedup, the sequential part of the workload should be run on a core of the fastest type, which has $\alpha_\text{fastest} = \max_{i=1}^{x} \alpha_i$.

To derive the speedup for Gustafson and Sun-Ni types of workloads, workload scaling according to the number of cores, memory and/or communications capabilities needs to be investigated. The normal-form modelling method assumes a general form of workload scaling. Specifically, the parallel part of the workload is assumed to be scaled according to a function $g(\overline{n})$, which has the same

representation power for any effects of cores, memory and/or communications as the scaling function $g(n)$ in Sun-Ni's model (10).

The general equation of normal-form HeMCP speedup models including for Amdahl, Gustafson and Sun-Ni types of workloads is

$$S(\overline{n}) = \frac{(1-f) + f \cdot g(\overline{n})}{\frac{(1-f)}{\alpha_s} + \frac{f \cdot g(\overline{n})}{N_\alpha}}. \quad (19)$$

It can be seen that the homogeneous Amdah's Law and Gustafson's and Sun-Ni's models are special cases of the normal-form model. They apply when only one core type (one-BCE cores) exists, causing $\alpha_s = 1$ and $N_\alpha = n$. The role of N_α as the parallel workload computation capability improvement index is now clear, as it may be regarded as the equivalent number of one-BCE cores in an HeMCP.

Similar to Hill-Marty's model, the normal-form model has also been extended to cover more than speedup. Power models especially have been developed for normal-form core heterogeneity. Extensive experimentation on off-the-shelf CPU-only and CPU-GPU-GPU systems covering single-ISA and multi-ISA cases validate the normal-form model's real-world applicability [3], [5], [15].

Summary: The normal-form model [3] extends Amdahl's Law and related models to cover more general types of hardware heterogeneity applicable to current platform technologies.

7 PARALLELISM *or* P-FRACTION?

In this section we deal with HoMCP, unless otherwise noted.

In parallel to researchers challenging Amdahl's Law about its assumption on hardware cores being all of the same type (parameter n), others have also questioned its assumption about the workload (parameter f). Especially, the assumed binary composition of sequential and parallel parts has also been regarded as inadequate.

In 1997, A. Downey commented that the parameter β in Equations (5) and (6) "has little semantic content" [13]. This can be understood to also apply to the p-fraction f. He proceeds to concentrate on the quantity known as parallelism, and derive speedup models based on that parameter, instead of the p-fraction.

Downey is far from the only researcher with this view, as Sun and Ni also analyzed the importance of parallelism [2], and Cassidy and Andreou commented that this binary assumption of the workload is "somewhat arbitrary" [37]. Others have been more adamant about the inadequacy of Amdahl's Law [92].

The focus of this dispute is the fundamental assumption of Amdahl's Law as described by Equation (1), which says that a workload is assumed to consist of two distinct parts, one of which absolutely cannot be parallelized and the other has full (infinite) parallelizability.

First of all, infinite parallelizability is difficult to envisage (hence semantically weak) for a workload of fixed size, which Amdahl's Law targets. The fundamental atomic element of workloads is usually agreed to be the single instruction. As a workload of fixed size does not expand into an infinite number of instructions, because of the fixed size assumption, it or any part of it should always have non-infinite parallelizability. Secondly, the idea of trying to approximate different degrees of parallelizability with a weighted sum of non- and full-parallelizability may not be regarded as attractive, and has practical limitations, as will be explained later in this section.

Even if the fundamental "a none-part plus an infinity-part" assumption is accepted, we run into the problem that the p-fraction f is regarded as difficult to determine for workloads not designed on-purpose to fit specific f values, such as synthetic benchmarks that allow the intentional tuning of f [5], [15]. Even programmers would have difficulty determining the f value of any code that they themselves have generated. As a result the practical usability of Amdahl's Law and any other models that derive from it may be negatively affected, even though beautiful mathematical forms can be readily derived.

Parallelism, on the other hand, suffers from none of these issues. The parallelism of a workload, denoted by the variable name p in this paper, is defined as follows [35, p. 780] [2], [36]:

$$p = \frac{T_1}{T_\infty}, \quad (20)$$

where T_1 is the time taken to execute the workload on one core, and T_∞ is the time taken to execute the workload on an infinite number of cores. In other words, parallelism is the maximum possible speedup of a workload through increasing the number of cores available for its execution.

A narrower, more intuitive understanding is that p is the number of concurrent threads a parallel workload has [2], [35], [93]. This works with the assumption that each core can execute one and only one thread exclusively at any time, and parallelization means mapping individual concurrent threads onto available cores, one thread per core. In this context, the maximum speedup of a parallel workload is indeed its number of concurrent threads. You can increase n and the speedup would increase, until you hit $n = p$, after which further increases of n would not improve speedup because the workload simply runs out of threads to parallelize and some cores may be starved of tasks. With this intuition, the parallelism p of a workload acquires a clear meaning as the workload's inherent parallelizability.

In this view, an Amdahl's type workload can be divided into two parts: a sequential part, whose parallelism is $p_\mathrm{s} = 1$, and a parallel part, whose parallelism is $p_\mathrm{p} = \infty$.

With Amdahl's Law, a workload can be said to have a static p-fraction, i.e. the entire workload as a whole has a constant f which describes the workload's overall parallelizability, leading to a speedup estimation based on the improvement index n. For parallelism, because the number of concurrent threads is rarely a constant throughout the entire workload, it is possible to consider a workload's average p and its variance as the static parameters. Detailed speedup models may then become more complex ending up with a family of speedup curves for different cases [13].

Another way of studying speedup with regard to parallelism is through two formulas known as the Work Law and Span Law [35], [92], which deal with bounds. In

this context, work is defined as the sum of the time taken by every one of a workload's instructions, which is the same as the total time taken by executing a workload fully sequentially on a single core. In other words, work is none other than T_{w} (T_1).

Let T_n be the shortest possible execution time for running the workload on n cores. The Work Law is then

$$T_n \geq \frac{T_1}{n}. \tag{21}$$

For this simple version of the Work Law to hold, a set of assumptions must be true. The workload is assumed to contain an integer number of instructions, which are its atomic elements. Each core executes one instruction in one unit of time. As a result, n cores at most executes n instructions per unit time. Therefore, to complete the entire workload on n cores must take at least T_1/n units of time. It has been shown that it is possible to extend the model to handle non-unit instruction times and other more complex behaviours [35].

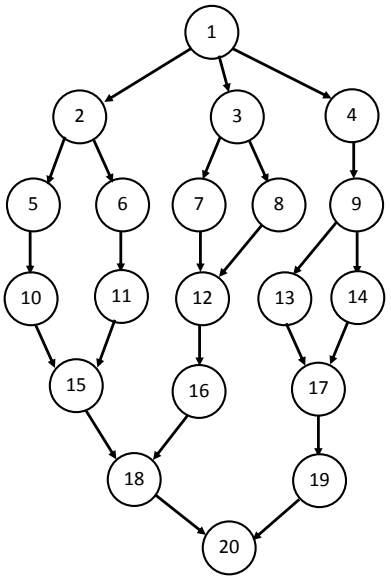

Fig. 7: Dag describing a workload consisting of 20 atomic elements.

A workload of this type may be described by a directed acyclic graph (dag) [35], sometimes known as the workload's *task graph* [24], [94]. An example dag can be seen in Fig. 7. Following the assumptions for the simple Work Law in (21), the dag in Fig. 7 describes a workload consisting of 20 instructions, with the vertices representing the instructions and the directed arcs defining the sequential order between pairs of instructions. For instance, instructions ⑦ and ⑧ start after the completion of instruction ③, and in turn complete before instruction ⑫ starts. The work T_1 of this dag is 20.

Speedup from the Work Law is

$$S(n) = \frac{T_1}{T_n} \leq n. \tag{22}$$

This simple version of the Work Law covers both Amdahl's and Gustafson's models, but does not cover the superlinearity observed in Sun-Ni's model. This is because the assumptions leading to the Work Law deals with core-bounded speedup. It may be extended to deal with complex issues, including memory and communications, that may lead to superlinear speedup [92].

"Span" is another word for critical path length. A critical path is the, or one of the, longest sequence(s) of consecutive vertices in a task graph. Given the set of simple assumptions mentioned in the preceding paragraphs, the span, in units of time, is the greatest number of instructions that must be executed sequentially in a workload. In other words, span is none other than T_∞, as the time taken to complete a workload on an infinite number of cores is indeed the workload's longest sequence of consecutive instructions. For the workload described by the task graph in Fig. 7, one of the critical paths is ①, ③, ⑧, ⑫, ⑯, ⑱, and ⑳, with seven instructions. Thus this workload has a span of 7.

The Span Law says that a parallel processing machine with n cores cannot run faster than one that has an infinite number of cores, for any value of n. Hence

$$T_n \geq T_\infty. \tag{23}$$

Taking work T_1 to span T_∞ results in a workload's parallelism $p = T_1/T_\infty$. For the example in Fig. 7 this is $20/7 \approx 2.86$. Apart from being the maximum speedup for a workload obtainable from adding cores, parallelism can also be understood as the average amount of work along each step of the critical path. For instance, the steps in the example described by Fig. 7 have parallelisms of 1, 3, 5, 5, 3, 2 and 1, which average out to about 2.86. It is also possible to use a task dag to describe a workload at the granularity of threads rather than instructions, by extending the notion of task from individual instructions to cover multi-instruction threads. The overall assumption that a task does not include any internal parallelism (i.e. $p_{\mathrm{task}} = 1$) must always be true for this extension to be valid. In this extended case, vertices represent threads, and the parallelism p is the number of threads that can be run in parallel [2]. Threads may not all take the same time to run, however, but this can easily be worked into a dag representation. Although the quantities of Work and Span will have more complex definitions which must include potentially varied thread execution times, this does not affect the qualitative validity of the Work and Span Laws.

Compared with the difficulty of obtaining correct f values for workloads, which tends to require post-design or even run-time experimentation [15], [66], the parallelism p is much more readily available to the programmer, who should be able to generate a task graph during the workload design process, which would contain information about parallelism. Program code can also be annotated or instrumented to help run-time extraction of the task graph. Independent of the programmer, there are methods of extracting the task graph or otherwise determine the parallelism during workload execution [24], [36], [94].

The average p and its variances may be useful in speedup analysis [13], but "*instantaneous parallelism*" is more useful for such endeavors as the efficient scheduling of tasks [36], [44], [95]–[97]. The concept of instantaneous parallelism is highlighted in [36]. A workload may display

a particular parallelism p at any point of its execution, as the number of parallel threads change.

The examples of task distribution shown in Fig. 6 can now be reviewed in terms of parallelism. It can be seen that the naïve workload distribution in Fig. 6(a) results in an overall smaller parallelism than the ideal distribution in Fig. 6(b). This results in a larger span (if span is understood in terms of execution time) for the former case.

Instantaneous parallelism may be used to analyze and design scheduling policies. For instance, in Fig. 6(a), p starts at 1, then becomes 3 after crossing T_s, then it reduces to 2 and eventually 1 as the faster cores stop processing. In Fig. 6(b), however, p only has two phases, 1 and 3. The greater instantaneous p in the second case is the cause of its comparatively smaller span and higher speedup. In other words, the quality of parallel workload distribution is positively related to the parallelism achieved. The points in the execution trace in Fig. 6(a) where the instantaneous parallelism reduces from 3 to 2 and from 2 to 1 may be identified as the result of non-optimal workload distribution (scheduling).

Instantaneous parallelism may also be studied during the original design of workloads. For instance, for the workload with a task graph of the shape of the dag in Fig. 7, functionally the designer may decide to move ⑨ to the fourth step, and form a new step with ⑬ and ⑭ inserted before the step of ⑮, ⑯ and ⑰. This does not affect the logical correctness of the workload, but would reduce the instantaneous parallelism of steps 4 and 5 from 5 to 4, and lead to the addition of one more step causing the span of the dag to grow from 7 to 8. The overall parallelism would reduce from $p \approx 2.86$ to $p = 2.5$. Fully sequentializing ⑤, ⑥, ⑩ and ⑪ would lead to an even bigger reduction of instantaneous parallelism and increase of span. The costs of such design changes are clearly described by changes in the instantaneous (and overall) parallelism and span, helping the designer to derive quantitative trade-offs with any benefits.

The example in Fig. 7 may be used to clarify why Amdahl's Law is regarded as semantically weak in certain cases. This dag has a maximum parallelism of 5. In no part of the workload is it infinitely parallelizable. Only in two steps out of seven is the workload non-parallelizable. The rest of the workload has $1 < p < \infty$, i.e. parallelism values of neither 1 nor infinity. It is not immediately clear what single f value can describe the entire workload satisfactorily.

Looking closer at the example by focusing on the first two steps involving tasks ①, ②, ③ and ④ in Fig. 7, it can be seen that the workload is non-parallelizable ($p = 1$) in step 1 and has a parallelism of $p = 3$ in step 2.

The case of $n = 1$ can be trivially observed to conform to Amdahl's Law. For $n = 2$, the sequential part of the workload, task ①, executes on one core, then tasks ② and ③ execute on $n = 2$ cores and ④ executes on one core, with the entire workload taking a total time of $T_2 = 3$. The speedup achieved by using two cores is therefore

$$S(2) = \frac{T_1}{T_2} = \frac{4}{3} \approx 1.33. \tag{24}$$

For Amdahl's Law, we may calculate f from (2) as $f = T_p/T_1 = 3/4 = 0.75$. From (4), the speedup can be derived as follows

$$S(2) = \frac{1}{(1-f) + \frac{f}{n}} = \frac{1}{0.25 + \frac{0.75}{2}} = 1.6. \tag{25}$$

In other words, the real speedup of the workload does not tightly observe Amdahl's Law for $n = 2$, but is smaller. In fact it follows Li-Malek's type of model with a penalty term of 1 added to the denominator (See Equation (9)), although in this case, the penalty term comes from non-ideal load balancing because of the atomicity of a task and not from inter-core communications. For $n = 3$, however, using Amdahl's Law the speedup can be calculated as $S(3) = 2$, the same as the real speedup obtained from executing ① on one core and ②, ③ and ④ on three cores because load balancing is not a problem.

When the number of available cores grows to $n > p_\infty$ (from $n = 4$ onwards in this case), the parallel part of the workload, with a parallelism of p_∞, cannot be evenly distributed to the n available cores, with some $n - p_\infty$ cores starved of workload. The speedup is a constant $S(n) = S(p_\infty) = S(3) = 2$, for all $n \geq 3$. The overall parallelism of this workload is 2, which means that the speedup achievable with an infinite number of cores is 2, which is already achieved by having 3 cores. This core starvation is not captured by Amdahl's Law, which says adding cores always improves speedup, so long as f is not zero. Testing with $n = 4$ shows that whilst the real speedup is 2, Amdahl's Law returns a higher speedup estimate of approximately 2.29. For $n = \infty$, Amdahl's Law gives a speedup of 4 for $f = 0.75$.

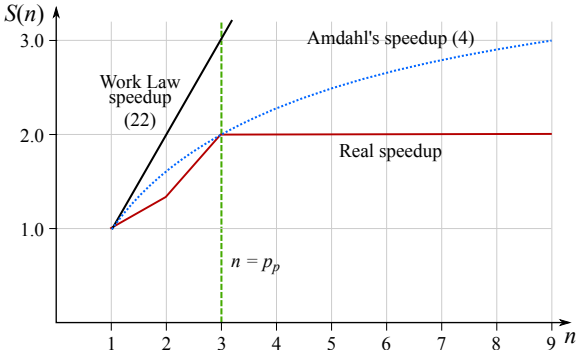

Fig. 8: Three different speedup estimations.

Fig. 8 illustrates this example discussion. Work Law speedup formula (22) gives maximum speedup $S(n) \leq n$, the black line in the figure. Amdahl's Law gives speedup $S(n) = 1/(0.25 + 0.75/x)$ according to (4), shown in blue in the figure. The real upper limit of speedup, given $p_\infty = 3$, follows the curve in red in the figure. It can be seen that the most precise speedup estimation, with the lowest speedup values obtained, is below Amdahl's Law before $n = p_\infty$ and is constant after that. This type of starvation may appear naturally suited for roofline modelling [40], however, the following sections will show that Amdah's Law may be extended to cover such behaviour without resorting to roofline models.

Compared to Amdahl's Law, because Gustafson assumes scalability of the workload in relation to n, which implies $p_\infty \geq n$ always, the problem of core starvation does not apply to Gustafson's model [22]. In other words, if you

run larger parallel problems when you get more cores, you would never run out of threads and end up with starved cores, assuming ideal load balancing [5]. Sun-Ni's model can represent the effects of parallelism in the function $g(n)$, in addition to the effects of memory, on which the model is primarily focused [2]. This will be discussed further in the following section.

Summary: Another fundamental assumption of Amdahl's Law, that the workload consists of two parts, one fully sequential and the other fully parallel, has always attracted criticism. Workload studies have progressed separately along the lines of using the concept of parallelism rather than the p-fraction when calculating speedup as a result of parallelization leading to the Span Law and Work Law [35]. More intuitive methods such as roofline modelling [40] directly represent behaviour that is not straightforward under Amdahl-style assumptions.

8 PARALLELISM *and* P-FRACTIONS!

Amdahl's Law, whose simple mathematical form results from its somewhat simplistic assumptions, has remained a popular tool for reasoning about speedup [5], [24], [42], [44], [60], [98]. A number of researchers have attempted to resolve the problems with regard to the workload parallelism p.

In 2012, S. Tang et al. presented a method for dealing with multi-level parallelization [93] and T. Zidenberg et al. presented the Multi-Amdahl method of optimizing resources on a chip [19], both hinting the possible use of multiple p-fractions. It was however Cassidy and Andreou who presented in [37] the concrete form of what we will call the *multi-fraction* model. This method has been used to make Amdahl's Law relevant in the context of workloads with different degrees of parallelism during execution [23, eq.(2.6)] [24].

The multi-fraction speedup model is based on the vectorization of the p-fraction f into $\overline{f} = (f_1, f_2, ..., f_n)$, where for any $1 \leq j \leq n$, f_j is the fraction of workload that is executed on j cores.

Using the task graph in Fig. 7 as an example, let us assume simple task distribution, i.e. cores are scheduled according to the instantaneous parallelism of the workload, with one task per core at any time, to ease the discussion. In step 2 of the dag there would be three cores executing tasks ②, ③ and ④ in parallel, and in the next step five cores execute the tasks ⑤ to ⑨ in parallel. In the dag the tasks are of the same size, this provides for better intuitive description, but is not required by the method.

The method sums the tasks with the same parallelism p under the respective f_p and then normalizes the value by the total number of tasks. In this example, two tasks have $p = j = 1$, two tasks have $p = j = 2$, six tasks have $p = j = 3$ and ten tasks have $p = j = 5$, where j is the correct number of cores used to deal with a corresponding instantaneous p value. This gives $f_1 = 2/20 = 0.1$, $f_2 = 2/20 = 0.1$, $f_3 = 6/20 = 0.3$, $f_4 = 0$, and $f_5 = 10/20 = 0.5$. Note that the sum of all f_j values is 1, just like in Amdahl's Law, because of the normalization.

Assuming that the total time taken to execute the entire workload sequentially on one core is $T_1 = 1$ (i.e. BCE, to simplify discussion without losing generality – in speedup models the actual value of T_1 does not matter as it gets cancelled out), the time taken to execute the jth fraction of the workload, $1 \leq j \leq n$, on j cores is then f_j/j, and the total time to execute the entire workload with each fraction on its corresponding number of cores (with $p = j$, according to the simple task distribution assumption) is

$$T_n = \sum_{j=1}^{n} \frac{f_j}{j}. \tag{26}$$

The multi-fraction extension of Amdahl's Law, adapted from [37, Equation (3)] is then

$$S(n) = \frac{T_1}{T_n} = \left[\sum_{j=1}^{n} \frac{f_j}{j} \right]^{-1}. \tag{27}$$

Classical Amdahl's Law (4) is a special case of (27) with $f_1 = 1 - f$ and $f_n = f$, with all other $f_j = 0$, $\forall 1 < j < n$. This is because when running an Amdahl's type workload on a system with n cores, the parallel part is fully parallelizable onto all cores hence essentially $f = f_\infty = f_n$.

The sum of f should equal to 1 in order to represent the entire workload:

$$\sum_{j=1}^{n} f_j = 1. \tag{28}$$

For the example described by Fig. 7, the speedup is therefore $S(n \geq 5) = 1/(f_1/1 + f_2/2 + f_3/3 + f_4/4 + f_5/5) = 1/(0.1 + 0.05 + 0.1 + 0 + 0.1) = 1/0.35$. This is indeed the dag's overall parallelism $p = T_w/T_\infty = 20/7 \approx 2.86$. It makes sense that for a workload whose dag shows a maximum parallelism of 5, running it on a maximum of five cores already achieves the same speedup as running it on an infinite number of cores. The issue of core starvation when there are more cores than threads is therefore naturally represented in the multi-fraction model of (27).

When the task graph is used on its own for reasoning about speedup, with no special mention of hardware, it is usually assumed that there are always cores available to execute one task per core no matter how large the workload's instantaneous p is. In other words, $p \leq n$. However, if $\max p > n$, i.e. the task graph allows higher parallelism than there are cores available, the best scheduling could only provide n cores at any time. The fraction extraction needs to represent the effects of this scheduling by appropriately "stretching" the task times. For instance, if the workload described by the task graph in Fig. 7 is executed on a system with a maximum of four available cores, instead of $f_5 = 10/20 = 0.5$, we have $f_5 = 0$, $f_4 = 8/20$, and f_2 changing from $2/20$ to $4/20$, if we push task ⑨ down a step and then execute tasks ⑬ and ⑭ in an extra two-tasks-on-two-cores step. This would add one step to the span. The speedup is reduced to 2.5 for $n = 4$ from 2.86 for $n \geq 5$.

An interesting consequence of the dependence of f_j values on n is that the multi-fraction model implicitly supports workload scaling models such as Gustafson's [22] and Sun-Ni's [2]. These classical workload scaling models assume that f does not change but describe the scaling of workload according to the improvement index (or core

number) n with a separate parameter – the scaling function $g(n)$ – so that the scaled parallel part of the workload becomes $f \cdot g(n)$. This scaling function is not required for the multi-fraction model since the f_j values already can represent workloads that change with n, as they are functions of the number of cores they correspond to, i.e. $f_j = f(j)$. The only required extension is to allow the workload task graph to change with the number of cores, i.e. the system's maximum parallelism to be no smaller than the number of cores.

However, an important detail difference is that, according to Gustafson and Sun-Ni, the workload scaling may break condition (28). In fact, for $n > 1$, the sums of their scaled workload fractions are always greater than 1. Consequently, a re-normalization is needed to derive the speedup, and (27) becomes

$$S(n) = \left[\sum_{j=1}^{n} f_j\right] \cdot \left[\sum_{j=1}^{n} \frac{f_j}{j}\right]^{-1}. \tag{29}$$

It can be verified that (29) transforms into Sun-Ni's model (10) by substituting f_1 with $1-f$ and f_n with $f \cdot g(n)$, and $f_j = 0, \forall\ 1 < j < n$. For $g(n) = n$ the model further transforms into Gustafson's (7), and for $g(n) = 1$ it becomes classical Amdahl's Law (4).

Other related models, for instance that extending models similar to Sun-Ni's over Hill-Marty asymmetric heterogeneity [60], are also covered by the multi-fraction model with similar arguments.

The method of adding a penalty term to the denominator of Amdahl's Law to represent overheads, exemplified by Li-Malek's model (9) [53] and used in other work [54, p.167][23, p.42][55], [56], is also covered by the multi-fraction model. The multi-fraction model allows the more precise representation of qualitative and quantitative overhead effects as these can usually be precisely incorporated into task graph modifications: additional tasks, lengthening existing tasks, synchronizing tasks, data overheads, instruction overheads, etc. The simplest form (9) essentially enlarges the sequential part with no change to the parallel part. This can be represented in the multi-fraction model by appropriately increasing f_1 whilst keeping the other f_j values unchanged, then re-normalizing where appropriate.

The method of extracting f_j values from the task graph, found in e.g. [24], produces normalized f_j values by design. The condition (28) holds in the resulting models without requiring re-normalization. For simplicity, in the subsequent text we use (27) and assume no need for re-normalization, without losing generality.

In 2018, Yun et al. proposed a method to further extend the multi-fraction speedup model to cover an enhanced asymmetric core heterogeneity [24]. Their method assumes that there are two types of cores in the system, each with a different processing capability, and reduces the normal-form model [5] to fit this assumption. This assumption is correct for the ARM big.LITTLE system configuration consisting of n_{L} low power LITTLE cores and n_{b} high performance big cores.

They regard the LITTLE core as BCE (with performance of 1), and the performance of a big core is represented as relative to this BCE by α_{b}. They also assume that the scheduling always prioritizes high performance cores. The resulting heterogeneous model is

$$S(n_{\mathrm{b}}, n_{\mathrm{L}}) = \left[\sum_{b=1}^{n_{\mathrm{b}}} \frac{f_{\mathrm{b}}}{\alpha_{\mathrm{b}} \cdot b} + \sum_{L=1}^{n_{\mathrm{L}}} \frac{f_{n_{\mathrm{b}}+L}}{\alpha_{\mathrm{b}} \cdot b + L}\right]^{-1}. \tag{30}$$

The degree of core heterogeneity covered by this model is limited, in the sense that there are only two types of cores and the scheduling favours the faster type.

Gupta et al. [30], [99] investigated the effects of the performance of individual cores on speedup, in a heterogeneous parallel processing context. These models focus on the performance scaling of cores, via such techniques as DVFS and performance optimization. In this context, it is legitimate to assume that neither the number of cores nor the task to core scheduling change between the unscaled baseline and the scaled execution. These considerations allow the direct use of time fractions instead of going through workload fractions. On the subject of system heterogeneity, they make two main extensions. Firstly the implicit assumption that the sequential part of a workload is executed on a single type of core is removed, and the sequential part of the task is now assumed to be executed on an arbitrary number of cores of arbitrary types, fully sequentially (i.e. potentially involving core to core handovers). Secondly the parallel part is assumed to consist of multiple phases that must be executed sequentially, phase by phase, with each phase being a parallelizable set of tasks. The model for speedup by scaling core speeds can be found in [30, Equation (5)]. In this multi-fraction model following Cassidy and Andreou [37], the execution time of each parallel phase is calculated according to a similar method to that shown in Equation (15), which is generally correct for all schemes of task to core allocation. The modelling does not make assumptions on the parallelism of each parallel phase or the number of cores available for each parallel phase, but the assumption of sequential plus parallel phases is fully within the descriptive power of dags of the type found in Fig. 7.

A more general heterogeneity in multi-fraction approach is also supported by the method known as Multi-Amdahl [19]. This model links heterogeneity with the allocation of some resource X, which can be divided into n arbitrary sections, and each section x_j is dedicated to run a fraction of the workload f_j, $1 \leq j \leq n$. These arbitrary sections are able to universally represent any type of heterogeneity; however, the authors put a very specific constraint on their model: these sections can only be executed sequentially, so that the total execution time T_n is:

$$T_n = \sum_{j=1}^{n} f_j \cdot e\left(x_j\right), \tag{31}$$

where $e\left(x_j\right)$ is the so-called *efficiency function* (although the name is somewhat misleading as larger values of $e\left(x_j\right)$ cause longer execution times; in other words, this function is reciprocal to the performance achieved by the resource

x_j). The model also explicitly states that the resources do not overlap:

$$\sum_{j=1}^{n} x_j \leq X. \tag{32}$$

The Multi-Amdahl paper [19] does not explicitly specify the equation for speedup but focuses directly on minimizing T_n under the constraint (32), but since they define their execution time in relation to a baseline $T_1 = 1$, it is straightforward to deduce that the speedup in their case is calculated as:

$$S(n) = \frac{T_1}{T_n} = \left[\sum_{j=1}^{n} f_j \cdot e(x_j) \right]^{-1}. \tag{33}$$

Despite the generality of Multi-Amdahl, its assumption of sequentially executing hardware sections has been a major criticism against the practicality of the model [42].

Rafiev et al. combined the multi-fraction model (27) with the normal-form HeMCP assumption (Fig. 5) to better cover core heterogeneity without restraining the model by any specific scheduling priorities or core or execution constraints [38].

The normal-form model of core heterogeneity makes use of two vectors $\overline{n} = (n_1, ..., n_x)$ and $\overline{\alpha} = (\alpha_1, ...\alpha_x)$ to represent the number of each of x types of cores and the core type's relative performance with regard to a BCE. In the present context, there is no need to highlight core-type differences. Without clustering cores by type, each core's performance can be defined individually, leading to a single vector $\overline{\alpha} = (\alpha_1, ...\alpha_n)$ for n cores, $n = n_1 + ... + n_x$. This individual-core view also facilitates the representation of such fine-grain control possibilities as per-core DVFS [8], [9], [100]. To derive speedup models based on this type of normal-form HeMCP assumption, models for scheduling options, which now have to deal with core heterogeneity, also need to be explored. This is because with heterogeneous core performances, some scheme of core prioritizing, exemplified by but by no means limited to "favouring faster cores", is natural in any sensible scheduling policy. And this needs to be represented properly in any reasonable model.

Similar to Gupta et al. [30], Rafiev et al. [38] generalized the representation of scheduling policy beyond "favouring faster cores". Intuitively, this may be done by enumerating the cores in the order of their scheduling priority, if these stay constant during the execution of a workload. However, this is not enough as the priorities may change depending on the number of parallel threads (instantaneous parallelism p) which may change with a workload's progress. For instance, it may be desirable to use a single high performance core for sequential execution when $p = 1$ and then in some cases change to all low power cores under higher degrees of parallelism, when $p > 1$. The scheduling priority passes from high performance to low power cores in response to parallelism changes and a constant priority per core list cannot be built. This type of scheduling is especially relevant when trying to maximize performance without exceeding some power budget [101]. Two methods for generalizing scheduling models were described in [38] to cover these kinds of characteristics.

In "core-based generalization", it is assumed that the scheduling behaviour is determined by the instantaneous parallelism p. As a result, for any $1 \leq j \leq n$, a separate vector $\overline{\alpha_j} = (\alpha_{j1}, ...\alpha_{jj})$ is defined representing the BCE-relative performances of exactly those j cores that execute the fraction f_j. The combined performance of these cores is

$$A_j = \sum_{i=1}^{j} \alpha_{ji}, 1 \leq j \leq n. \tag{34}$$

The core-based multi-fraction model extended to normal-form core heterogeneity (called "normal-form multi-fraction model" in this paper) is then

$$S(n) = \left[\sum_{j=1}^{n} \frac{f_j}{A_j} \right]^{-1}. \tag{35}$$

The variable A_j has the same semantic meaning as N_α in equations (15) to (19), i.e. the generalized computation capability improvement index. It is a function of the participating cores and their relative performances, and hence of the scheduling decision. This brings the question of whether the effect of load balancing should be considered in the models. In other words, what happens when during the f_j phase, some k cores finish early (for example, if they are faster cores)? According to the multi-fraction model (27), the execution then switches to $f_{(j-k)}$. Therefore load balancing is already captured by the multiple f_j values [23], [24], [93], and A_j always equals the sum of performances and represent the improvement index over a BCE, as in (34). This simplifies the model, but adds practical complexities to the process of determining f_j values. For an HeMCP, the task graph needs to be analyzed and potentially modified with regard to the system's load balancing.

This can be dealt with using the "configuration-based generalization" [38]. This method includes the scheduling policy details as a central part of the model, which is built around the set $\{\overline{\alpha_1}, ...\overline{\alpha_Q}\}$ of system configurations, where $Q \geq 1$ is the number of configurations. Each configuration corresponds to a scheduling policy (i.e. a mapping of the workload on up to n cores), and defines the vector of n performance coefficients $\overline{\alpha_j} = (\alpha_{j1}, ..., \alpha_{jn})$, where $1 \leq j \leq Q$. If some cores are not used in a configuration, their performances are set to 0. The combined performance of a configuration is the sum of its n performance coefficients:

$$A_j = \sum_{i=1}^{n} \alpha_{ji}, 1 \leq j \leq Q. \tag{36}$$

With this configuration-centric modelling, the workload fractions f_j now correspond to the configurations rather than cores. In other words, f_j represents the fraction of the workload that is executed in the jth configuration $\overline{\alpha_j}$, $1 \leq j \leq Q$. The configuration-based normal-form multi-fraction model then takes the following form:

$$S(n) = \left[\sum_{j=1}^{Q} \frac{f_j}{A_j} \right]^{-1}. \tag{37}$$

The equation (37) is closely connected with the Multi-Amdahl model (33). Indeed, if we subdivide the resource X into Q parts instead of n and define the efficiency function as $e(x_j) = 1/A_j$, the equation (33) transforms into (37).

The major difference, however, is that the constraint (32) is not required: different configurations are allowed to reuse the same cores or resources because their execution times do not overlap. This even applies to classical Amdahl's Law where the core executing sequential fraction is also involved in the parallel execution. The issue with Multi-Amdahl approach can be solved by modifying (32) as:

$$x_j \leq X, \tag{38}$$

or in other words, the fraction f_j can use any amount of system resources as long as it does not exceed the entire system. This can also be applied to (36) in the following form:

$$A_j \leq A_{\max}, \tag{39}$$

where $A_{\max}$ is the maximum performance that can be achieved by the system.

The core-based normal-form multi-fraction model of (35) is a special case of (37). There is also a broader understanding of what a configuration is, which extends the semantic strength of this model to exceed a world view of concurrent threads running on parallel cores.

In this general understanding, a workload is executed on some machine, which has Q distinctive *modes* of operation. Each mode M_j, $1 \leq j \leq Q$, has a relative performance A_j when executing the workload, compared to some base equivalent performance, which has $A_{\text{base}} = 1$. The variable f_j denotes the *probability* of the workload being executed in M_j. This probability understanding extends the fraction assumption for deterministic systems to cover stochastic behaviour, and agrees well with the re-normalization for Gustafson's and Sun-Ni's models. The vector of mode (configuration) performances $\overline{A} = (A_1, ...A_Q)$ is the generalized computation capability improvement index, and a vector of real numbers. Each of these real numbers is the improvement, over that of the base equivalent performance of $A_{\text{base}} = 1$, achieved by a particular mode of the machine. Special cases are all covered by this understanding, for instance $f_j = 0$ if the workload cannot be executed in, or is not scheduled to M_j.

This broader understanding of the normal-form multi-fraction model returns to the broader understanding of the original form of Amdahl's Law (4), where the computation capability improvement index n could be a real number, which may not necessarily have anything to do with parallel processing or multiple cores. Similarly, operating modes do not have to achieve their computation capability improvements through parallel processing or multiple cores. Speedup is a result of improvement, and therefore a function of the improvement index $\overline{A}$. The model is valid regardless of the specific method from which any improvement derives. It is also valid for all possible workload structures including the number of steps and the degree of parallelism of each step, and the mapping of these tasks to cores fitting arbitrary system core architectures. For instance, the speedup obtained by scaling the sequential performances of cores investigated in [30], [99] is fully covered by this model.

Summary: Cassidy and Andreou, questioning the original Amdahl binary division of workloads into sequential and parallel parts, extend Amdahl's Law into the multi-fraction model [37]. The Multi-Amdahl research results in a more general type of the multi-fraction model [19]. The normal-form model is incorporated into the multi-fraction framework, leading to the configuration-based model [5] which applies to any type of system improvement. This kind of modelling is shown to be useful for deriving speedup from non-parallelization methods [30].

9 Discussions

The attraction of Amdahl's Law has been strong ever since the original observation was made in [1]. The reason is not difficult to understand. The concept of parallel processing and computer platforms supporting it have seen rapid developments in the half century since the publication of [1], and Amdahl's Law, with its simple form and intuitive understandability, has more often than not been the first formula for researchers and engineers to study when they want to reason about speedup.

The simplistic assumptions on which Amdahl's Law depends have, naturally, been the focus of a large number of model extensions, starting from Gustafson [22] and Sun-Ni [2], [59]. The popularity of Amdahl's Law means that it formed the foundation of a large body of research with people using it as the basis of their models, which target different real-world systems. This has inevitably led to similarities and re-inventing and re-iterating very similar ideas or even the same ideas.

For instance, multiple contributions, including [23], [48], [53]–[55], have suggested adding penalty terms to the denominator of Amdahl's Law to represent diverse overheads that reduce the parallelism in many different ways. Some of the researchers appear to have arrived at this method independently of others. Others have proposed to add a coefficient function to the parallel part of the denominator of Amdahl's Law, appearing to arrive at their models independently [2], [60].

On occasion, Amdahl's Law itself has been used without it being mentioned as Amdahl's Law [13].

Core heterogeneity is another issue tackled by multiple researchers. In certain ways, Moncrieff et al. [61], published in 1996, presented a more general model of core heterogeneity than Hill-Marty's models [4], which appeared more than a decade later. And yet it was the latter which directly inspired a large body of other research, including [25], [55], [56], [62]–[70], and led towards the development of forms of extension to cover HeMCP realities [3], [5], [19].

The work on Multi-Amdahl [19], [20], [102] produced models of extensive heterogeneity both in workload and in platform hardware configurations. The speedup model that may be derived from this method is a very small distance away from the normal-form multi-fraction model in Equation (37), as explained in Section 8. However,

TABLE 1: Classification of existing research

Control techniques and management	Run-time	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
	Core affinity	No	No	Yes	Yes	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
	DVFS	No	No	Yes	Yes	Yes	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Validation methods	Real application	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	No	No	No	No	Yes	No	Yes	No	Yes
	Theoretical modelling	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Modelling characteristics	Parallelization Factor	p-fraction	p-fraction	p-fraction	p-fraction	p-fraction	Parallelism	Parallelism	p-fraction	p-fraction	p-fraction	p-fraction	p-fraction	p-fraction	Multi p-fraction and parallelism	p-fraction	No	p-fraction	p-fraction	No	Multi p-fraction and parallelism	p-fraction	No	Parallelism
	Overhead calculations	No	No	No	Load balancing and scheduling	No	No	Load balancing and scheduling	No	Synchronization and communication	No	No	No	No	No	No	Time of parallel tasks	No	No	No	No	No	Memory	Communication
Parameters modelled	Power, Energy	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No
	Performance	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Types of heterogeneity covered	CPU-GPU	No	No	No	Yes	No	No	No	No	No	No	No	Yes	No	No	No	No	No	Yes	No	Yes	No	No	No
	Distributed	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
	Dynamic	No	No	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No	No	No	No	No	No	No	No	No
	Heterogeneity	No	No	Yes	Yes	No	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No
	Homogeneity	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Model types covered	Hill-Marty	No	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No	No	No	No	No	No	No
	Sun-Ni	No	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No
	Gustafson	No	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No	Yes	No	No	No
	Amdahl	Yes	Yes	Yes	Yes	Yes	Similar	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
	Literature	[1]	[2], [57]	[3]	[5]	[7]	[13]	[14]	[15]	[17]	[18]	[4], [63]	[19], [20], [102]	[22]	[24]	[25]	[26]	[27]	[28]	[29]	[38]	[39]	[46]	[48]

Classification of existing research (continued)

Literature	Amdahl	Gustafson	Sun-Ni	Hill-Marty	Homogeneity	Heterogeneity	Dynamic	Distributed	CPU-GPU	Performance	Power, Energy	Overhead calculations	Parallelization Factor	Theoretical modelling	Real application	DVFS	Core affinity	Run-time
	Model types covered				Types of heterogeneity covered					Parameters modelled		Modelling characteristics		Validation methods		Control techniques and management		
[49]	No	No	No	No	No	No	No	No	No	Yes	Yes	Communication	No	No	Yes	No	No	No
[50]	No	No	No	No	Yes	Yes	No	No	No	No	No	Communication	No	No	Yes	No	No	No
[51]	No	No	No	No	No	No	No	No	No	No	No	Synchronization and communication	No	No	Yes	No	No	No
[53]	Yes	No	No	No	Yes	No	No	No	No	Yes	No	Communication	p-fraction	Yes	No	No	No	No
[55], [56]	Yes	No	No	No	Yes	Yes	No	No	Yes	Yes	No	Synchronization, communication and memory	p-fraction	Yes	Yes	No	No	No
[58]	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	No	Memory	p-fraction	Yes	No	No	No	No
[59]	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	No	Memory	p-fraction	Yes	No	No	No	No
[60]	Yes	No	Similar	Yes	Yes	Yes	No	No	No	Yes	No	No	Parallelism	Yes	No	No	No	No
[61]	Yes	No	No	No	Yes	Yes	No	No	No	Yes	No	No	p-fraction	Yes	No	No	No	No
[62]	Yes	No	No	Yes	Yes	Yes	Yes	No	No	Yes	No	No	p-fraction	Yes	No	No	No	No
[64]	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	No	No	p-fraction	Yes	No	No	No	No
[65]	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	No	No	p-fraction	Yes	No	No	No	No
[66]	Yes	No	No	Yes	Yes	Yes	No	No	Yes	Yes	No	Memory	p-fraction	Yes	Yes	No	No	No
[67]	Yes	Yes	No	Yes	Yes	No	No	No	No	Yes	Yes	Memory and communication	p-fraction	Yes	Yes	No	No	No
[68], [69]	Yes	Yes	No	Yes	Yes	No	No	Yes	No	Yes	No	Communication	p-fraction	Yes	Yes	No	No	No
[70]	Yes	Yes	No	Yes	Yes	No	No	No	No	Yes	No	Communication	p-fraction	Yes	Yes	No	No	No
[80]	Yes	No	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	p-fraction	Yes	Yes	No	No	No
[98]	Yes	No	No	No	Yes	Yes	Yes	No	No	Yes	No	Memory and synchronization	Parallelism	Yes	No	No	No	No
[93]	Yes	Yes	No	No	Yes	No	No	No	No	Yes	No	Communication	Parallelism	Yes	Yes	No	No	No
[103]	Yes	No	No	Yes	No	Yes	No	No	No	Yes	Yes	No	p-fraction	Yes	No	No	No	No
[104]	No	No	No	No	Yes	No	No	No	No	Yes	No	Run-time	No	No	Yes	No	No	Yes
[105]	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes	Run-time	No	No	Yes	Yes	Yes	Yes

Multi-Amdahl was tied to a Hill-Marty-like assumption of microarchitecture and chip configuration, had restrictions that are not necessary for M/MCP, and did not specifically promote a speedup model. It was not immediately picked up for HeMCP modelling [38], [42].

One of the most important questions for models is what they shall be used for. One use of a speedup model is in the optimization of system operations via design-time and run-time management [8], [15], [30], [34], [36], [96], [99]–[101], [103], [106], [107].

For practical applications, the task graph of a workload may be extracted [36], [94], [104], and the p-fraction of a workload can also be obtained using experimental methods [15], [66]. The availability of these input variables to speedup models through offline experiments, design-time modelling and run-time monitoring allow the models to be used for run-time control purposes [14], [15], [30], [36], [105].

Both generalization and specialization have happened in model design and use with regard to various extensions of Amdahl's Law. Using vectors to replace the scalars found in (4) has been one of the main ways of improving the coverage of Amdahl's Law [5], [93]. Others include adding terms in the form of constant parameters or functions [53], [54] and strengthening constants with functions [2]. Reduced forms of general models with more limited scopes have also been used in targeted cases to lessen the modelling effort and improve the presentational clarity. For instance, a reduced form of the normal-form speedup model was used in [24] to specially target the ARM big.LITTLE architecture.

Given that a form of Amdahl's Law has been derived (37) that generalizes speedup modelling to cover wide heterogeneity in workload, hardware and workload to hardware mapping, a major challenge for researchers and engineers who want to reason about speedup and performance bounds in the HeMCP era is to find the most user-friendly reduced-scope models to target their specific needs. In this context, the general model of (37) may serve as the foundation for deriving appropriate special-purpose models for practical use. Its method of use, the enumeration of all modes and all workload-mode mapping probabilities, is methodologically straightforward but may be practically challenging. This creates rich opportunities for further research and the development of innovative practical solutions. For instance, what easy-to-use functional forms can be derived to relate A_j to practical factors (e.g. effects associated with design and operational realities such as network topology, communication synchronization, and power budgeting) deemed important by system designers reasoning about mode M_j?

Table 1 summarizes the surveyed research publications highlighting the topics they cover.

10 Concluding remarks

Starting from the observation made by Amdahl in 1967 [1], researchers have come up with a series of extensions to Amdahl's Law in order to improve the semantic power of both of its parameters, the p-fraction f [2], [24], [38], [53], [93] and the computation capability improvement index n [3]–[5], [24], [38], [94]. Through the vectorization of both parameters, and by viewing them as variables and functions rather than constants, the normal-form multi-fraction models now represent wide scopes of heterogeneity in workload parallelism, processor core architectures, and scheduling decisions. The general form of the normal-form multi-fraction model (37) has now gone back to basics without specifically targeting multiple cores and parallel processing.

11 Acknowledgment

This work is part of the PRiME (EPSRC grant EP/K034448/1) and STRATA (EPSRC grant EP/N023641/1) projects. The first author also thanks his sponsor HCED–Iraq and the University of Technology–Iraq for funding and other support.

References

[1] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *IEEE Solid-State Circuits Society Newsletter*, vol. 12, no. 3, pp. 19–20, Summer 2007.

[2] X. H. Sun and L. M. Ni, "Another view on parallel speedup," in *Proceedings Super Computing '90*, Nov 1990, pp. 324–333.

[3] M. A. N. Al-hayanni, A. Rafiev, R. Shafik, and F. Xia, "Power and energy normalized speedup models for heterogeneous many core computing," in *2016 16th International Conference on Application of Concurrency to System Design ACSD*, Torun, Poland., June 2016, pp. 84–93.

[4] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 7, pp. 33–38, July 2008.

[5] A. Rafiev, M. A. N. Al-hayanni, F. Xia, R. Shafik, A. Romanovsky, and A. Yakovlev, "Speedup and power scaling models for heterogeneous many-core systems," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 436–449, July 2018.

[6] J. Rabaey and M. Pedram, *Low Power Design Methodologies*, ser. The Springer International Series in Engineering and Computer Science. Springer US, 2012. [Online]. Available: https://books.google.co.uk/books?id=9IzuBwAAQBAJ

[7] F. Xia, A. Rafiev, A. Aalsaud, M. Al-hayanni, J. Davis, J. Levine, A. Mokhov, A. Romanovsky, R. Shafik, A. Yakovlev, and S. Yang, "Voltage, throughput, power, reliability, and multicore scaling," *Computer*, vol. 50, no. 8, pp. 34–45, 2017.

[8] S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems," *International Journal of Computer Aided Engineering and Technology*, vol. 6, no. 4, pp. 440–459, 2014.

[9] S. Eyerman and L. Eeckhout, "Fine-grained DVFS using on-chip regulators," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 1, pp. 1:1–1:24, Feb. 2011. [Online]. Available: http://doi.acm.org/10.1145/1952998.1952999

[10] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff." *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 5, pp. 33–35, Sept 2006.

[11] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, no. 3, pp. 46–54, March 2011.

[12] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual Design Automation Conference*. ACM, June 2007, pp. 746–749.

[13] A. B. Downey, "A model for speedup of parallel programs," UC Berkeley, Berkeley, CA, USA, Tech. Rep., 1997.

[14] S. Sridharan, G. Gupta, and G. S. Sohi, "Adaptive, efficient, parallel execution of parallel programs," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '14. New York, NY, USA: ACM, 2014, pp. 169–180. [Online]. Available: http://doi.acm.org/10.1145/2594291.2594292

[15] M. A. N. Al-hayanni, R. Shafik, A. Rafiev, F. Xia, and A. Yakovlev, "Speedup and parallelization models for energy-efficient many-core systems using performance counters," in *2017 International Conference on High Performance Computing Simulation (HPCS)*, July 2017, pp. 410–417.

[16] Intel, *Intel 64 and IA-32 Architectures Software Developer's Manual. Volume 3B: System Programming Guide, Part 2*. Intel, September 2016. [Online]. Available: http://www.intel.co.uk/content/www/uk/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.html

[17] T. Y. Morad, U. C. Weiser, A. Kolodny, M. Valero, and E. Ayguade, "Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors," *IEEE Computer Architecture Letters*, vol. 5, no. 1, pp. 4–17, Jan. 2006. [Online]. Available: http://dx.doi.org/10.1109/L-CA.2006.6

[18] T. Sato, H. Mori, R. Yano, and T. Hayashida, "Importance of single-core performance in the multicore era," in *Proceedings of the Thirty-fifth Australasian Computer Science Conference - Volume 122*, ser. ACSC '12. Darlinghurst, Australia: Australian Computer Society, Inc., 2012, pp. 107–114. [Online]. Available: http://dl.acm.org/citation.cfm?id=2483654.2483667

[19] T. Zidenberg, I. Keslassy, and U. Weiser, "Multi–Amdahl: How should I divide my heterogenous chip?" *IEEE Computer Architecture Letters*, vol. 11, no. 2, pp. 65–68, July 2012.

[20] A. Morad, T. Y. Morad, Y. Leonid, R. Ginosar, and U. Weiser, "Generalized Multi–Amdahl: Optimization of heterogeneous multi-accelerator SoC," *IEEE Computer Architecture Letters*, vol. 13, no. 1, pp. 37–40, Jan 2014.

[21] (2017) Intel. [Online]. Available: https://www.intel.co.uk/content/www/uk/en/homepage.html

[22] J. L. Gustafson, "Reevaluating Amdahl's law," *Communications of ACM*, vol. 31, no. 5, pp. 532–533, May 1988. [Online]. Available: http://doi.acm.org/10.1145/42411.42415

[23] S. Mercelis, "A systematic multi-layered approach for optimizing and parallelizing real-time media and audio applications," Ph.D. dissertation, University of Antwerp, 2016.

[24] Y. Yun, S. Han, and Y. H. Kim, "Estimation of maximum speedup in multicore-based mobile devices," *IEEE Embedded Systems Letters*, pp. 1–1, 2018.

[25] D. H. Woo and H. 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.

[26] K. W. Cameron and R. Ge, "Generalizing Amdahl's law for power and energy," *Computer*, vol. 45, no. 3, pp. 75–77, March 2012.

[27] S. Cho and R. Melhem, "Corollaries to Amdahl's law for energy," *IEEE Computer Architecture Letters*, vol. 7, no. 1, pp. 25–28, Jan 2008.

[28] A. Marowka, "Extending Amdahl's law for heterogeneous computing," in *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, July 2012, pp. 309–316.

[29] J. Issa and S. Figueira, "Performance and power-consumption analysis of mobile internet devices," in *30th IEEE International Performance Computing and Communications Conference*, Nov 2011, pp. 1–6.

[30] U. Gupta, S. Korrapati, N. Matturu, and U. Y. Ogras, "A generic energy optimization framework for heterogeneous platforms using scaling models," *Microprocessors and Microsystems*, vol. 40, no. Supplement C, pp. 74 – 87, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933115000885

[31] L. F. Wilson and W. Shen, "Experiments in load migration and dynamic load balancing in speedes," in *1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, vol. 1, Dec 1998, pp. 483–490 vol.1.

[32] J. C. Ryou and J. S. K. Wong, "A task migration algorithm for load balancing in a distributed system," in *[1989] Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences: Software Track*, Jan 1989, pp. 1041–1048 vol.2.

[33] S. Johari and A. Kumar, "Algorithmic approach for applying load balancing during task migration in multi-core system," in *2014 International Conference on Parallel, Distributed and Grid Computing*, Dec 2014, pp. 27–32.

[34] Y. Li, J. Niu, X. Long, and M. Qiu, "Energy efficient scheduling with probability and task migration considerations for soft real-time systems," in *2014 IEEE Computers, Communications and IT Applications Conference*, Oct 2014, pp. 287–293.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithm*. The MIT Press, 2009. [Online]. Available: https://mcdtu.files.wordpress.com/2017/03/introduction-to-algorithms-3rd-edition-sep-2010.pdf

[36] K. Agrawal, Y. He, W. Hsu, and C. E. Leiserson, "Adaptive scheduling with parallelism feedback," in *2007 IEEE International Parallel and Distributed Processing Symposium*, March 2007, pp. 1–7.

[37] A. S. Cassidy and A. G. Andreou, "Beyond amdahl's law: An objective function that links multiprocessor performance gains to delay and energy," *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1110–1126, Aug 2012.

[38] A. Rafiev, M. A. N. Al-hayanni, F. Xia, R. Shafik, A. Romanovsky, and A. Yakovlev, "Extending multi-fraction speedup models to normal form heterogeneity," μSystems Research Group, School of Engineering, Newcastle University, Tech. Rep. NCL-EEE-MICRO-TR-2018-202, 2018.

[39] S. M. Londono and J. P. de Gyvez, "Extending Amdahl's law for energy-efficiency," in *2010 International Conference on Energy Aware Computing*, Dec 2010, pp. 1–4.

[40] G. Ofenbeck, R. Steinmann, V. C. Cabezas, D. G. Spampinato, and M. Püschel, "Applying the roofline model," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2014, pp. 76 – 85.

[41] U. Gupta, J. Campbell, U. Y. Ogras, R. Ayoub, M. Kishinevsky, F. Paterna, and S. Gumussoy, "Adaptive performance prediction for integrated gpus," in *Proceedings of the 35th International Conference on Computer-Aided Design*, ser. ICCAD '16. New York, NY, USA: ACM, 2016, pp. 61:1–61:8. [Online]. Available: http://doi.acm.org/10.1145/2966986.2966997

[42] B. M. Al-Babtain, F. J. Al-Kanderi, M. F. Al-Fahad, and I. Ahmad, "A survey on Amdahl's law extension in multicore architectures," *International Journal of New Computer Architectures and their Applications*, vol. 3, pp. 30–46, 01 2013.

[43] R. Ayoub, U. Ogras, E. Gorbatov, Y. Jin, T. Kam, P. Diefenbaugh, and T. Rosing, "Os-level power minimization under tight performance constraints in general purpose systems," in *IEEE/ACM International Symposium on Low Power Electronics and Design*, Aug 2011, pp. 321–326.

[44] S. D. Casey. (2011) How to determine the effectiveness of hyper-threading technology with an application. [Online]. Available: https://software.intel.com/en-us/articles/how-to-determine-the-effectiveness-of-hyper-threading-technology-with-an-application

[45] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, Mar. 1995. [Online]. Available: http://doi.acm.org/10.1145/216585.216588

[46] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin, "Scaling the bandwidth wall: Challenges in and avenues for CMP scaling," *SIGARCH Computer Architecture News*, vol. 37, no. 3, pp. 371–382, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/1555815.1555801

[47] S. A. McKee and R. W. Wisniewski, *Memory Wall*. Springer US, 2011.

[48] X. Chen, Z. Lu, A. Jantsch, and S. Chen, "Speedup analysis of data-parallel applications on multi-core NoCs," in *IEEE 8th International Conference on ASIC*, Oct 2009, pp. 105–108.

[49] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling," in *32nd International Symposium on Computer Architecture (ISCA'05)*, June 2005, pp. 408–419.

[50] E. R. Rodrigues, F. L. Madruga, P. O. A. Navaux, and J. Panetta, "Multi-core aware process mapping and its impact on communication overhead of parallel applications," in *2009 IEEE Symposium on Computers and Communications*, July 2009, pp. 811–817.

[51] T. B. Ahmad and M. Ciesielski, "An approach to multi-core functional gate-level simulation minimizing synchronization and communication overheads," in *2013 14th International Workshop on Microprocessor Test and Verification*, Dec 2013, pp. 77–82.

[52] R. Zurawski, *Embedded Systems Handbook, Second Edition: Embedded Systems Design and Virification*. Taylor and Francis, 2009.

[53] X. Li and M. Malek, "Analysis of speedup and communication/computation ratio in multiprocessor systems," in *Proceedings. Real-Time Systems Symposium*, Dec 1988, pp. 282–288.

[54] M. J. Quinn, *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill Education Group, 2003.

[55] S. Pei, M. S. Kim, and 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.

[56] S. Pei, J. Zhang, L. Jiang, M.-S. Kim, and J.-L. Gaudiot, "Reevaluating the overhead of data preparation for asymmetric multicore system on graphics processing." *KSII Transactions on Internet & Information Systems*, vol. 10, no. 7, 2016.

[57] X. Sun and L. Ni, "Scalable problems and memory-bounded speedup," *Journal of Parallel and Distributed Computing*, vol. 19, no. 1, pp. 27 – 37, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731583710877

[58] X.-H. Sun, Y. Chen, and S. Byna, "Scalable computing in the multicore era," in *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Programming*, 2008.

[59] X.-H. Sun and Y. Chen, "Reevaluating Amdahl's law in the multicore era," *Journal of Parallel Distributed Computing*, vol. 70, no. 2, pp. 183–188, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2009.05.002

[60] S. Eyerman and L. Eeckhout, "Modeling critical sections in amdahl's law and its implications for multicore design," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 362–370, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1816011

[61] D. Moncrieff, R. E. Overill, and S. Wilson, "Heterogeneous computing machines and Amdahl's law," *Parallel Computing*, vol. 22, no. 3, pp. 407–413, 1996.

[62] E. Yao, Y. Bao, G. Tan, and M. Chen, "Extending Amdahl's law in the multicore era," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 2, pp. 24–26, Oct. 2009. [Online]. Available: http://doi.acm.org/10.1145/1639562.1639571

[63] E. Yao, Y. Bao, and M. Chen, "What Hill–Marty model learn from and break through Amdahl's law?" *Information Processing Letters*, vol. 111, no. 23, pp. 1092–1095, 2011.

[64] B. Juurlink and C. H. Meenderinck, "Amdahl's law for predicting the future of multicores considered harmful," *SIGARCH Comput. Archit. News*, vol. 40, no. 2, pp. 1–9, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2234336.2234338

[65] N. Ye, Z. Hao, and X. Xie, "The speedup model for manycore processor," in *2013 International Conference on Information Science and Cloud Computing Companion*, Dec 2013, pp. 469–474.

[66] E. Blem, H. Esmaeilzadeh, R. S. Amant, K. Sankaralingam, and D. Burger, "Multicore model from abstract single core inputs," *IEEE Computer Architecture Letters*, vol. 12, no. 2, pp. 59–62, July 2013.

[67] G. H. Loh, "The cost of uncore in throughput-oriented many-core processors," in *In Proc. of Workshop on Architectures and Languages for Troughput Applications (ALTA*, 2008, pp. 1–9.

[68] N. P. Khanyile, J.-R. Tapamo, and E. Dube, "An analytic model for predicting the performance of distributed applications on multicore clusters," *International Association of Engineers (IAENG)*, 2012.

[69] ——, "Performance prediction model for distributed applications on multicore clusters," *International Association of Engineer (IAENG)*, 2012.

[70] T. Huang, Y. Zhu, M. Qiu, X. Yin, and X. Wang, "Extending Amdahl's law and Gustafson's law by evaluating interconnections on multi-core processors," *The Journal of Supercomputing*, vol. 66, no. 1, pp. 305–319, Oct 2013. [Online]. Available: http://dx.doi.org/10.1007/s11227-013-0908-9

[71] K. Vipin and S. A. Fahmy, "FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications," *ACM Computing Survey*, vol. 51, no. 4, pp. 72:1–72:39, Jul. 2018. [Online]. Available: http://doi.acm.org/10.1145/3193827

[72] A. Vajda, *Multi-core and Many-core Processor Architectures*. Springer US, 2011.

[73] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas, "Single-ISA heterogeneous multi-core architectures for multithreaded workload performance," in *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, June 2004, pp. 64–75.

[74] M. G. Arenas, A. M. Mora, G. Romero, and P. A. Castillo, *GPU Computation in Bioinspired Algorithms: A Review*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 433–440. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21501-8-54

[75] C. M. Wittenbrink, E. Kilgariff, and A. Prabhu, "Fermi GF100 GPU architecture," *IEEE Micro*, vol. 31, no. 2, pp. 50–59, March 2011.

[76] V. W. Lee, E. Grochowski, and R. Geva, "Performance benefits of heterogeneous computing in HPC workloads," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*, May 2012, pp. 16–26.

[77] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, Dec 2003, pp. 81–92.

[78] P. Greenhalgh, "White paper: big.little processing with arm Cortex-A15 and Cortex-A7 - improving energy efficiency in high-performance mobile platforms," *ARM*, 2011. [Online]. Available: https://www.cl.cam.ac.uk/ rdm34/big.LITTLE.pdf

[79] A. Venkat and D. M. Tullsen, "Harnessing ISA diversity: Design of a heterogeneous-ISA chip multiprocessor," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, June 2014, pp. 121–132.

[80] E. S. Chung, P. A. Milder, J. C. Hoe, and K. Mai, "Single-chip heterogeneous computing: Does the future include custom logic, fpgas, and GPGPUs?" in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, Dec 2010, pp. 225–236.

[81] J. Power, A. Basu, J. Gu, S. Puthoor, B. M. Beckmann, M. D. Hill, S. K. Reinhardt, and D. A. Wood, "Heterogeneous system coherence for integrated CPU-GPU systems," in *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2013, pp. 457–467.

[82] A. Pathania, Q. Jiao, A. Prakash, and T. Mitra, "Integrated CPU-GPU power management for 3D mobile games," in *Proceedings of the 51st Annual Design Automation Conference*, ser. DAC '14. New York, NY, USA: ACM, 2014, pp. 40:1–40:6. [Online]. Available: http://doi.acm.org/10.1145/2593069.2593151

[83] H. Mujtaba. (2015) Intel Skylake GPU architecture analysis. [Online]. Available: https://wccftech.com/idf15-intel-skylake-analysis-cpu-gpu-microarchitecture-ddr4-memory-impact/3/

[84] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, "Debunking the 100X GPU vs. CPU Myth: An evaluation of throughput computing on CPU and GPU," *SIGARCH Computer Architecture News*, vol. 38, no. 3, pp. 451–460, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1816021

[85] J. Palacios and J. Triska, "A comparison of modern GPU and CPU architectures: And the common convergence of both," *Oregon State University*, 2011. [Online]. Available: https://hgpu.org/?p=6610

[86] C. Cullinan, C. Wyant, T. Frattesi, and X. Huang, "Computing performance benchmarks among CPU, GPU, and FPGA," *Worcester Polytechnic Institute*, 2012. [Online]. Available: https://web.wpi.edu/Pubs/E-project/Available/E-project-030212-123508/unrestricted/Benchmarking˙Final.pdf

[87] K. J. Duda and D. R. Cheriton, "Borrowed-virtual-time BVT scheduling: Supporting latency-sensitive threads in a general-purpose scheduler," *SIGOPS Oper. Syst. Rev.*, vol. 33, no. 5, pp. 261–276, Dec. 1999. [Online]. Available: http://doi.acm.org/10.1145/319344.319169

[88] A. Agarwal, R. Bianchini, D. Chaiken, K. L. Johnson, D. Kranz, J. Kubiatowicz, B.-H. Lim, K. Mackenzie, and D. Yeung, "The MIT alewife machine: Architecture and performance," *SIGARCH Computer Architecture News*, vol. 23, no. 2, pp. 2–13, May 1995. [Online]. Available: http://doi.acm.org/10.1145/225830.223985

[89] B. Boothe and A. Ranade, "Improved multithreading techniques for hiding communication latency in multiprocessors," *SIGARCH Computer Architecture News*, vol. 20, no. 2, pp. 214–223, April 1992. [Online]. Available: http://doi.acm.org/10.1145/146628.139729

[90] S. Mittal and J. S. Vetter, "A survey of CPU-GPU heterogeneous computing techniques," *ACM Computer Survey*, vol. 47, no. 4, pp. 69:1–69:35, July 2015. [Online]. Available: http://doi.acm.org/10.1145/2788396

[91] (2017) The international technology roadmap for semiconductors ITRS. [Online]. Available: http://www.itrs2.net/

[92] C. Leiserson. (2017) What the $# ¡ is parallelism, anyhow? [Online]. Available: https://www.cprogramming.com/parallelism.html

[93] S. Tang, B. S. Lee, and B. He, "Speedup for multi-level parallel computing," in *2012 IEEE 26th International Parallel and*

Distributed Processing Symposium Workshops PhD Forum, May 2012, pp. 537–546.

[94] S. Han, Y. Yun, and Y. H. Kim, "Profiling-based task graph extraction on multiprocessor system-on-chip," in *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Oct 2016, pp. 510–513.

[95] C. McCann, R. Vaswani, and J. Zahorjan, "A dynamic processor allocation policy for multiprogrammed shared-memory multiprocessors," *ACM Trans. Comput. Syst.*, vol. 11, no. 2, pp. 146–178, May 1993. [Online]. Available: http://doi.acm.org/10.1145/151244.151246

[96] X. Deng and P. Dymond, "On multiprocessor system scheduling," in *Proceedings of the Eighth Annual ACM Symposium on Parallel Algorithms and Architectures*, ser. SPAA '96. New York, NY, USA: ACM, 1996, pp. 82–88. [Online]. Available: http://doi.acm.org/10.1145/237502.237510

[97] K. K. Yue and D. J. Lilja, "Implementing a dynamic processor allocation policy for multiprogrammed parallel applications in the solaristm," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 6, pp. 449–464, 2001.

[98] H. Che and M. Nguyen, "Amdahl's law for multithreaded multicore processors," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 3056–3069, 2014.

[99] U. Gupta and U. Y. Ogras, "Constrained energy optimizationin heterogeneous platforms usinggeneralized scaling models," *IEEE Computer Architecture Letters*, vol. 14, no. 1, pp. 21–25, Jan 2015.

[100] W. Kim, M. S. Gupta, G. Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, Feb 2008, pp. 123–134.

[101] A. Aalsaud, A. Rafiev, F. Xia, R. Shafik, and A. Yakovlev, "Model-free runtime management of concurrent workloads for energy-efficient many-core heterogeneous systems," in *28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2018, pp. 206–213.

[102] T. Zidenberg, I. Keslassy, and U. Weiser, "Optimal resource allocation with Multi-Amdahl," *Computer*, vol. 46, no. 7, pp. 70–77, July 2013.

[103] S. Lee, S. H. Kim, and W. W. Ro, "Multicore speedup models using frequency scaling with fixed power budget," in *2014 International Conference on Electronics, Information and Communications (ICEIC)*, Jan 2014, pp. 1–2.

[104] K. K. Pusukuri, R. Gupta, and L. N. Bhuyan, "Thread reinforcer: Dynamically determining number of threads via OS level monitoring," in *2011 IEEE International Symposium on Workload Characterization (IISWC)*, Nov 2011, pp. 116–125.

[105] H. Sasaki, S. Imamura, and K. Inoue, "Coordinated power-performance optimization in manycores," in *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques*, Sept 2013, pp. 51–61.

[106] K. Singh, M. Bhadauria, and S. A. McKee, "Real time power estimation and thread scheduling via performance counters," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 2, pp. 46–55, 2009.

[107] Y. Li, J. Niu, J. Zhang, M. Atiquzzaman, and X. Long, "An optimized RM algorithm by task affinity on multi-core processor," in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2016, pp. 286–293.