Speedup and Power Scaling Models for Heterogeneous Many-Core Systems

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

Abstract—Traditional speedup models, such as Amdahl's law, Gustafson's, and Sun and Ni's, have helped the research community and industry better understand system performance capabilities and application parallelizability. As they mostly target homogeneous hardware platforms or limited forms of processor heterogeneity, these models do not cover newly emerging multi-core heterogeneous architectures. This paper reports on novel speedup and energy consumption models based on a more general representation of heterogeneity, referred to as the normal form heterogeneity, that supports a wide range of heterogeneous many-core architectures. The modelling method aims to predict system power efficiency and performance ranges, and facilitates research and development at the hardware and system software levels. The models were validated through extensive experimentation on the off-the-shelf big.LITTLE heterogeneous platform and a dual-GPU laptop, with an average error of 1% for speedup and of less than 6.5% for power dissipation. A quantitative efficiency analysis targeting the system load balancer on the Odroid XU3 platform was used to demonstrate the practical use of the method.

Index Terms—Heterogeneous systems, speedup modelling, energy-aware systems, load balancing, Amdahl's law, multi-core processors

1 Introduction

From the early days of computing systems, persistent engineering efforts have been made to improve computation speed by distributing work across multiple devices. A major focus in this area of research has been on predicting a gain in system performance, called *speedup*. Amdahl's law, in use since 1967 [1], assumes that a fixed workload is executed in n processors and compares their performance with that of a single processor executing the same workload. The model shows that if the workload requires synchronization the speedup will be quickly saturated with an increase in n. In 1988, Li and Malek explicitly considered inter-processor communications in this model [2]. In the same year, Gustafson introduced the principle of workload scaling pertaining to the fixed time model [3]. His model extends the workload proportionally to system scalability with the result of having a linear increase in the speedup. In 1990, Sun and Ni put forward a new model, which calculated the workload extension by considering memory capability [4].

Over the years, increased operating frequency and smaller device geometries have led to a significant performance improvement at reduced power consumption [5]. The number of transistors per unit of area has increased substantially, conforming to Moore's [6] and Koomey's laws [7], and Pollack's rule suggests that the increase in performance is approximately proportional to the square root of the increase in complexity [8].

As a result, almost every modern consumer device or embedded system uses the computational power of multi-core processing. The number of cores in a device is constantly growing, hence the speedup scaling models remain highly important. The convenience of using Amdahl's law and the derived models is that they do not require complex modelling of inter-process communications. These models are based on the average platform and application characteristics and provide simple analytical solutions that project system capabilities in a clear and understandable way. They provide a valuable insight into system scalability and have become pivotal in multi-scale systems research. This is why it is important to keep them up to date, to make sure they remain relevant and correctly represent novel aspects of platform design.

With increasing system complexity and integration, the concept of heterogeneous computation has emerged. Initially, the heterogeneity appeared in a form of specialized accelerators, like GPU and DSP. In recent years, multiple types of CPU cores in a single device have also been made popular. For instance, the ARM big.LITTLE processor has found a wide use in mobile devices [9]. Heterogeneous systems pose additional engineering and research challenges. In the

- *A. Rafiev, M. Al-hayanni, F. Xia, R. Shafik, A. Romanovsky, and A. Yakovlev are with Newcastle University, UK*
  E-mail: {ashur.rafiev, m.a.n.al-hayanni, fei.xia, rishad.shafik, alexander.romanovsky, alex.yakovlev}@ncl.ac.uk
- *M. Al-hayanni is also with University of Technology and HCED, Iraq*
- *This work is supported by EPSRC/UK as a part of PRiME project EP/K034448/1.*
- *Data supporting this publication is openly available under an 'Open Data Commons Open Database License'. Additional metadata are available at: http://dx.doi.org/10.17634/123238-4. Please contact Newcastle Research Data Service at rdm@ncl.ac.uk for access instructions.*
- *Full version (inc. the Appendix) is published in IEEE Transactions on Multi-Scale Computing Systems, http://dx.doi.org/10.1109/TMSCS.2018.2791531*

TABLE 1
Existing speedup models and the proposed model

	homogen.	heterogen.	power	memory	intercon.	Amdahl	Gustafson	Sun-Ni
[1]	yes	no	no	no	no	yes	no	no
[2]	yes	no	no	no	yes	yes	no	no
[3]	yes	no	no	no	no	yes	yes	no
[4]	yes	no	no	yes	no	yes	yes	yes
[12]	yes	simple	no	no	no	yes	no	no
[13], [14]	yes	simple	no	yes	no	yes	yes	yes
[15], [16]	yes	simple	yes	no	no	yes	no	no
proposed models	yes	normal form	yes	no	no	yes	yes	yes

area of scheduling and load balancing, the aim is to improve core utilization for more efficient use of the available performance. Operating systems traditionally implement symmetric multi-processor (SMP) scheduling algorithms designed for homogeneous systems, and ARM have done dedicated work on modifying the Linux kernel to make load balancing suitable for their big.LITTLE processor [10].

In addition to performance concerns, power dissipation management is also a significant issue in scalable systems: according to Dennard's CMOS scaling law [11] despite smaller geometries the power density of devices remains constant.

Hill and Marty extended Amdahl's speedup model to cover simple heterogeneous configurations consisting of a single big core and many smaller ones of exactly the same type [12], which relates to the CPU-GPU type of heterogeneity. The studies in [13], [14] extended Hill-Marty analysis to all three major speedup models. The problem of energy efficiency has been addressed in [15], [16] for the homogeneous and simple heterogeneous Amdahl's model. This overview covers only the most relevant publications, an extensive survey can be found in [17].

1.1 Research Contributions

This paper extends the classical speedup models to a *normal form* representation of heterogeneity, which describes core performances as a vector. This representation can fit a wider range of systems, including the big.LITTLE processor, homogeneous processors with multiple dynamic voltage-frequency scaling (DVFS) islands, and multi-GPU heterogeneous systems. The initial work on this topic was published in [18], which includes the derivation of the speedup models and a set of power models for this extended representation of heterogeneity. In the current publication we expand the work by addressing the effects of workload distribution and load balancing, and also explore additional modes of workload scaling relevant only to heterogeneous systems. We discover that the presented models inherit certain limitations from Amdahl's law. The paper provides a concise discussion on the matter. In addition, an extensive set of experiments has been carried out to validate the proposed models on both heterogeneous CPU and CPU-GPU platforms, and to explore their practical use.

This paper makes the following contributions:

- extending the classical speedup models to normal form heterogeneity in order to represent modern examples of heterogeneous systems;
- extending models to include power estimation;
- clarifying the limitations of the Amdahl-like heterogeneous models and outlining further challenges of heterogeneous speedup and power modelling;
- validating the models on commercial heterogeneous platforms;
- practically using the models to evaluate the efficiency of the Linux scheduler's load balancing while running realistic workloads in a heterogeneous system.

This work lays the foundation for a new type of heterogeneous models. The effects of memory and interconnects are planned as future model extensions. Table 1 compares this paper's contributions to the range of related research publications.

The experimental work presented in this paper has been carried out on the Odroid-XU3 [19] development platform centred around ARM big.LITTLE and on a dual-GPU laptop (Dell XPS).

The paper is organized as follows. Section 2 gives an overview of the existing homogeneous and heterogeneous speedup models. Section 3 discusses the model assumptions and formally defines the structure of a normal form heterogeneous system. Sections 4 and 5 present the new heterogeneous speedup and power models respectively. Sections 6 and 7 experimentally validate the models. Section 8 shows the experiments with real life benchmarks. Section 9 outlines the future work, and Section 10 concludes the paper.

2 Existing Speedup Models

In homogeneous systems all cores are identical in terms of performance, power, and workload execution.

For a homogeneous system we consider a system consisting of n cores, each core having a performance of $\theta = \frac{I}{t(1)}$, where I is the given workload and $t(1)$ is the time needed to execute the workload on the core. This section describes various existing models for determining the system's speedup $S(n)$ in relation to a single core, which can be used to find the performance $\Theta(n)$ of the system:

$$\Theta(n) = \theta S(n). \tag{1}$$

Amdahl-like speedup models are built around the parallelizability factor p, $0 \leq p \leq 1$. Given a total workload of I, the parallel part of a workload is pI and the sequential part is $(1-p)I$.

2.1 Amdahl's Law (Fixed Workload)

This model compares the execution time for some fixed workload I on a single core with the execution time for the same workload on the entire n-core system [1].

Time to execute workload I on a single core is $t(1)$, whereas $t(n)$ adds up the sequential execution time on one core at the performance θ and the parallel execution time on all n cores at the performance $n\theta$:

$$t(1) = \frac{I}{\theta}, \quad t(n) = \frac{(1-p)I}{\theta} + \frac{pI}{n\theta}, \tag{2}$$

thus the speed up can be found as follows:

$$S(n) = \frac{t(1)}{t(n)} = \frac{1}{(1-p) + \frac{p}{n}}. \quad (3)$$

2.2 Gustafson's Model (Fixed Time)

Gustafson re-evaluated the fixed workload speedup model to derive a new fixed time model [3]. In this model, the workload increases with the number of cores, while the execution time is fixed. An important note is that the workload scales asymmetrically: the parallel part is scaled to the number of cores, whilst the sequential part is not increased.

Let's denote the initial workload as I and extended workload as I'. The time to execute initial workload and extended workload are $t(n)$ and $t'(n)$ respectively. The workload scaling ratio can be found from:

$$t(1) = \frac{I}{\theta}, \quad t(n) = \frac{(1-p)I}{\theta} + \frac{pI'}{n\theta}, \quad (4)$$

and, since $t(1) = t(n)$, the extended workload can be found as $I' = nI$. The time that would take to execute I' on a single core is:

$$t'(1) = \frac{(1-p)I}{\theta} + \frac{pnI}{\theta}, \quad (5)$$

which means that the achieved speedup equals to:

$$S(n) = \frac{t'(1)}{t(1)} = (1-p) + pn. \quad (6)$$

The main contribution of Gustafson's model is to show that it is possible to build applications that scale to multiple cores without suffering saturation.

2.3 Sun and Ni's Model (Memory Bounded)

Sun and Ni took into account the previous two speedup models by considering the memory bounded constraints [4], [20]. In this model the parameter $g(n)$ reflects the scaling of the workload in relation to scaling the memory with the number of cores:

$$I' = g(n)I. \quad (7)$$

The model calculates the speedup as follows:

$$S(n) = \frac{t'(1)}{t'(n)} = \frac{(1-p) + pg(n)}{(1-p) + \frac{pg(n)}{n}}. \quad (8)$$

One of the important properties of this model is that for $g(n) = 1$ Sun and Ni's model (8) transforms into Amdahl's law (3), and for $g(n) = n$ it becomes Gustafson's law (6). Further in this paper, we do not specifically relate $g(n)$ to the memory access or any other property of the system but consider it as a given or determined parameter pertaining to a general case of workload scaling.

2.4 Hill and Marty's Heterogeneous Models

Hill and Marty extended speedup laws to heterogeneous systems, and mainly focused on a single high performance core and many smaller cores of the same type [12].

Core performances are related to some base-core equivalent (BCE), which is considered to have $\theta = 1$. Given a characterization parameter r, this model studies a system with one big core having $\Theta(r)$ relative performance and $(n-r)$ little cores with BCE performances, as shown in Figure 1(b). The sequential workload is executed on the faster core, while the parallel part exercises all cores simultaneously. This transforms Amdahl's law (3) as follows:

$$S(n,r) = \frac{1}{\frac{(1-p)}{\Theta(r)} + \frac{p}{\Theta(r)+(n-r)}}. \quad (9)$$

A reconfigurable version of this model can be extended to cover multi-GPU systems, but still implies only one active accelerator at a time [21], [16]. In this work we aim to cover more diverse cases of heterogeneity pertaining to such modern architectures as ARM big.LITTLE [19] and multi-GPU platforms with simultaneous heterogeneous execution, which are not directly covered by Hill and Marty's models.

3 Heterogeneous Systems

Homogeneous models are used to compare the speedup between different numbers of cores. Similarly, heterogeneous models should compare the speedup between core configurations, where each configuration defines the number of cores in each available core type. This section discusses the problems of modelling consistency across different core types and provides the foundation for all heterogeneous models presented later in this paper.

3.1 The Challenges of Heterogeneous Modelling

Heterogeneous models must capture the performance and other characteristics across different types of cores in a comparable way. Such a comparison is not always straightforward, and in many ways similar to cross-platform comparison. This section discusses the assumptions behind Amdahl's law and similar models under the scope of heterogeneous modelling and outlines the limitations they may cause.

3.1.1 Hardware-dependent parallelizability

In the models presented in Section 2, there is a time-separation of the sequential and parallel executions of the entire workload. These models do not explore complex interactions between the processes, hence they do not provide exact timing predictions and should not be used for time-critical analyses like real-time systems research. Solving for process interactions is possible with Petri Net simulations [22] or process algebra [23]. Amdahl-like models, in contrast, focus on generic analytical solutions that give approximate envelopes for platform capabilities.

The parameter p is a workload property, assuming that the workload is running on ideal parallel hardware. Realistic hardware affects the p value of any workload running on it. From the standpoint of heterogeneous modelling, the potential differences in parallelizability between core types

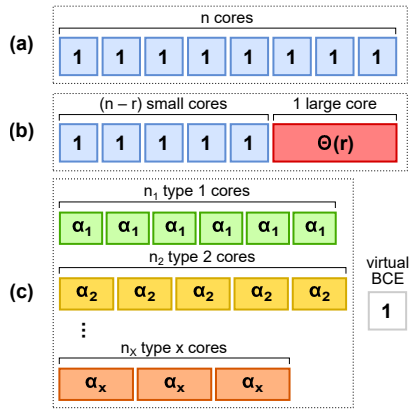

Fig. 1. The proposed extended structure of a heterogeneous system (c) compared to a homogeneous system (a) and the previous assumption [12] on heterogeneity (b). The numbers in the core boxes denote the equivalent number of BCEs.

or cache islands will cause the overall p to change between core configurations. In this paper, we do not attempt to solve this challenge. As demonstrated further in this paper, it is still possible to build heterogeneous models around a constant p and use a range of possible values to determine the system's minimum and maximum speedup capabilities.

3.1.2 Workload equivalence and performance comparison

Workload is a model parameter that links performance with the execution time. In many cases, a popular metric for performance is instructions per second (IPS), where a workload is characterized by its number of instructions. IPS is convenient as it is an application-independent property of the platform.

In heterogeneous models, it is important to have a consistent metric across all core types. For devices of different architecture types, the same computation may be compiled into different numbers of instructions. In this case, the total number of instructions may no longer meaningfully represent the same workload, and IPS cannot be universally used for cross-platform performance comparison. This is particularly clear when comparing CPU and GPU devices.

In order to build a valid cross-platform performance comparison model, we need to reason about the workload as a meaningful computation, and two workloads are considered equivalent as long as they perform the same task. In this paper we measure workload in so-called "workload items", which can be defined on a case by case basis depending on the practical application. Respectively, instead of energy per instruction, we use energy per workload item.

Hill and Marty's model, presented in Section 2.4, describes the performance difference between the core types as a property of the platform. In real life, this relation is application dependent, as will be demonstrated in Sections 6 and 7. Differences in hardware, such as pipeline depth and cache sizes, cause performance differences on a per-instruction basis even within the same instruction set [24].

3.2 Platform Assumptions

We build our models under the assumptions listed below. These assumptions put limitations on the models as discussed earlier in this section. The same assumptions are used in the classical Amdahl's law and similar models, hence there is no further reduction in generality.

- The models and model parameters are both application and hardware specific.
- The relation between performances of cores of different types can be approximated to a constant ratio, and the ratio can be determined.
- The parallelizability factor p can be approximated by a constant and is known or can be determined (exactly or within a range).
- Environmental factors, such as temperature, are not considered.

Memory and communication-related effects are not explicitly considered in this paper and are the subject of future work outlined in Section 9.

TABLE 2
List of performance-related variables

variable	description	Section
x	number of core types	3.3
n_i	number of cores of type i	3.3
$\overline{n}$	vector of core numbers	3.3
n, N	total number of cores (homo, hetero)	2, 3.3
θ	BCE performance	2, 3.3
α_i	performance factor for the core type i	3.3
$\overline{\alpha}$	vector of core performance factors	3.3
I	unscaled workload size	2
I'	scaled workload size	2.3, 4.3
$g(\overline{n})$	parallel workload scaling factor	2.3, 4.3
$h(\overline{n})$	proportional workload scaling factor	4.3
$t(\overline{n})$	unscaled workload exec. time	2
$t'(\overline{n})$	scaled workload exec. time	2.3
$t'_p(\overline{n}), t'_s(\overline{n})$	parallel and sequential exec. time	4.3
$S(\overline{n})$	speedup	2
$\Theta(\overline{n})$	system performance on n cores	2
p	parallelization factor	2
N_α	performance-equiv. number of BCEs	4.1
s	type of core executing seq. workload	4.2
α_s	performance factor of seq. exec.	4.2

3.3 Normal Form Representation of Heterogeneity

Performance-wise, the models presented in subsequent sections describe heterogeneity using the following normal form representation.

The normal form of heterogeneous system configuration considered in this paper consists of x clusters (types) of homogeneous cores with the numbers of cores defined as a vector $\overline{n} = (n_1, \ldots, n_x)$. The total number of cores in the system is denoted as $N = \sum_{i=1}^{x} n_i$. Vector $\overline{\alpha} = (\alpha_1, \ldots, \alpha_x)$ defines the performance of each core by cluster (type) in relation to some base core equivalent (BCE), such that for all $1 \leq i \leq x$ we have $\theta_i = \alpha_i \theta$. As discussed earlier, the parameter $\overline{\alpha}$ is application- and platform-dependent. The structure is shown in Figure 1(c).

The full list of performance-related variables used in the paper can be found in Table 2.

4 Proposed Speedup Models

This section extends homogeneous speedup models for determining the speedup $S(\overline{n})$ of a heterogeneous system in relation to a single BCE, which can then be used to find the performance of the system, by applying (1).

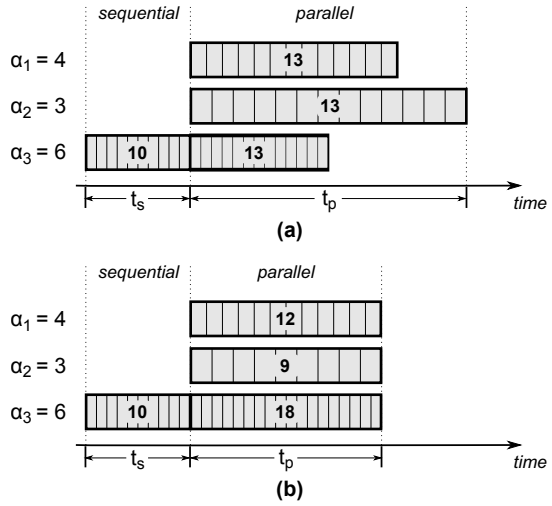

Fig. 2. Workload distribution examples following (a) equal-share model and (b) balanced model.

4.1 Workload Distribution

Homogeneous models distinguish two states of performance: the parallel execution exercises all cores, and the sequential execution exercises only one core while others are idle and do not contribute to overall system performance. The cores in such systems are considered identical, hence they all execute equal shares of the parallelizable part of the workload and finish at the same time. As the result, the combined performance of the cores working in parallel is θn. In heterogeneous systems this is not as straightforward: each type of cores works at a different performance rate, hence the execution time depends greatly on the workload distribution between the cores. Imperfect distribution causes some cores to finish early and become idle, even when the parallelizable part of the workload has not been completed.

In real systems, the scheduler is assisted by a load balancer, whose task is to redistribute the workload during run-time from busy cores to idle cores, however its efficiency is not guaranteed to be optimal [25]. The actual algorithm behind the load balancer may vary between different operating systems, and the load balancer typically has access to run-time only information like CPU time of individual processes and the sizes of waiting queues. Hence it is virtually impossible to accurately describe the behaviour of the load balancer as an analytical formula. In this section we address the problem by studying two boundary cases, which may provide a range of minimum and maximum parallel performances.

By definition, the *total execution time* for the workload I' is a sum of sequential and parallel execution times, $t'_s(\overline{n})$ and $t'_p(\overline{n})$, and it represents the time interval between the first instruction in I' starting and the last instruction in I' finishing. During a parallel execution, only the longest running core has an effect on the total execution time.

To be analogous to the homogeneous models and to simplify our equations, we also define the system's parallel performance via the *performance-equivalent number* of BCEs denoted as N_α.

4.1.1 Equal-share workload distribution

In homogeneous systems, the parallelizable workload is equally split between all cores. As a result, many legacy applications, developed with the homogeneous system architecture in mind, would also equally split the workload by the total number of cores (threads), which leads to a very inefficient execution in heterogeneous systems, where everyone has to wait for the slowest core (thread), as illustrated in Figure 2(a). In this case, N_α is calculated from the minimum of $\overline{\alpha}$:

$$N_\alpha = N \cdot \min_{i=1}^{x} \alpha_i. \tag{10}$$

The above equation implies that the workload cannot be moved between the cores. If the system load balancer is allowed to re-distribute the work, then the real N_α may be greater than (10). This equation can be used to define a lower performance bound corresponding to naïve scheduling policy with no balancing.

4.1.2 Balanced workload distribution

Figure 2(b) shows the ideal case of workload balancing, which implies zero waiting time, hence all cores should theoretically finish at the same time. N_α for optimal workload distribution is as follows:

$$N_\alpha = \sum_{i=1}^{x} \alpha_i n_i. \tag{11}$$

$N_\alpha \theta$ represents the system's performance during the parallel execution, hence N_α values from (10) and (11) define the range for heterogeneous system parallel performances. A load balancer that violates the lower bound (10) is deemed to be worse than naïve. The upper bound (11) represents the theoretical maximum, which cannot be exceeded.

4.2 Heterogeneous Amdahl's Law

We assume that the sequential part is executed on a single core in the cluster s, hence the system's performance during sequential execution is $\alpha_s \theta$. In Section 4.1 we defined parallel performance as $N_\alpha \theta$. Hence, the time to execute the fixed workload I on the given heterogeneous system is:

$$t(\overline{n}) = \frac{(1-p)\,I}{\alpha_s \theta} + \frac{pI}{N_\alpha \theta}. \tag{12}$$

The speedup in relation to a single BCE is:

$$S(\overline{n}) = \frac{t(1)}{t(\overline{n})} = \frac{1}{\frac{(1-p)}{\alpha_s} + \frac{p}{N_\alpha}}. \tag{13}$$

Hill-Marty's model (9) is a special case of (13), in which case $\overline{n} = (n-r, 1)$, $\overline{\alpha} = (1, \Theta(r))$, $\alpha_s = \Theta(r)$, and N_α is calculated for the balanced workload distribution (11).

4.3 Workload Scaling

Like in the homogeneous case, Amdahl's law works with a fixed workload, while Gustafson and Sun-Ni allow changing the workload with respect to the system's capabilities.

In this section we consider a general assumption on workload scaling, which defines the extended workload using characteristic functions $g(\overline{n})$ and $h(\overline{n})$ as follows:

$$I' = h(\overline{n}) \cdot ((1-p) I + pg(\overline{n}) I), \tag{14}$$

where $h(\overline{n})$ represents the symmetric scaling of the entire workload, and $g(\overline{n})$ represents the scaling of the parallelizable part only.

The sequential and parallel execution times are respectively:

$$t'_s(\overline{n}) = \frac{(1-p) I}{\alpha_s \theta}, \quad t'_p(\overline{n}) = \frac{pg(\overline{n}) I}{N_\alpha \theta}. \tag{15}$$

Hence, in the general case, for given workload scaling functions $g(\overline{n})$ and $h(\overline{n})$, the speedup is calculated as follows:

$$S(\overline{n}) = \frac{(1-p) + pg(\overline{n})}{\frac{(1-p)}{\alpha_s} + \frac{pg(\overline{n})}{N_\alpha}}. \tag{16}$$

The speedup does not depend on the symmetric scaling $h(\overline{n})$. Indeed, the execution time proportionally increases with the workload, and the performance ratio (i.e. the speedup) remains constant. However, changing the execution time is important for the fixed-time Gustafson's model.

4.4 Heterogeneous Gustafson's Model

In the Gustafson model, the workload is extended to achieve equal time execution: $t'(\overline{n}) = t(1)$. For homogeneous Gustafson's model: $g(\overline{n}) = n$ and $h(\overline{n}) = 1$. For a heterogeneous system, there is more than one way to achieve equal time execution.

4.4.1 Purely parallel scaling mode

The maximum speedup for equal time execution is achieved by scaling only the parallel part, i.e. $h(\overline{n}) = 1$. We know that Gustafson's model requires equal execution time, and we can find that:

$$g(\overline{n}) = \left(1 - \frac{(1-p)}{\alpha_s}\right) \frac{N_\alpha}{p}, \tag{17}$$

however, this equation puts a number of restrictions on the system. Firstly, it doesn't work for $p = 0$, because it is not possible to achieve equal time execution for a purely sequential program if $\alpha_s \neq 1$ and only parallel workload scaling is allowed. Secondly, a negative $g(\overline{n})$ does not make sense, hence the relation $\alpha_s > (1-p)$ must hold true. This means that the sequential core performance must be high enough to overcome the lack of parallelization. Another drawback of this mode is that it requires the knowledge of p in order to properly scale the workload.

In this scenario, the speedup is calculated from (16) and (17) as:

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

TABLE 3
List of power-related variables

variable	description	Section
w_0	idle power of a core	5
w	effective power of BCE	5
W_0	total background power	5
$W(\overline{n})$	total effective power	5
W_{total}	total power of the system	5
β_i	power factors of core type i	5.1
$\overline{\beta}$	vector of core power factors	5.1
w_s	sequential execution power	5.1
w_p	parallel execution power	5.1
N_β	power-equivalent number of BCEs	5.1
$D_w(\overline{n})$	power distribution function	5.2

4.4.2 Classical scaling mode

In order to remove the restrictions of the purely parallel scaling mode, and to provide a model generalizable to $p = 0$, we need to allow scaling of the sequential execution. However, since this mode potentially increases the sequential execution time, it exercises the cores less efficiently than the previous mode and leads to lower speedup.

$$g(\overline{n}) = \frac{N_\alpha}{\alpha_s}, \quad h(\overline{n}) = \alpha_s. \tag{19}$$

This scaling mode relates to the classical homogeneous Gustafson's model, which requires $g(n)$ to be proportional to the ratio between the system performances of the parallel and sequential executions. In the homogeneous case, if the sequential performance is θ, the parallel performance would be $n\theta$, leading to $g(n) = n$.

For the heterogeneous Gustafson's model in classical scaling mode, the speedup is calculated from (16) and (19) as:

$$S(\overline{n}) = \alpha_s (1-p) + pN_\alpha. \tag{20}$$

5 Proposed Power Models

We base our power models on the concept of power state modelling, in which a device has a number of distinct power states, and the average power over an execution is calculated from the time the system spend in each state.

For each core in the system we consider two power states: active and idle. Lower power states like sleeping and shutting down the cores may be catered to in straightforward extensions of these models. Active power of a core can also be expressed as a sum of idle power w_0 and effective power w that is spent on workload computation. In this view, the idle component is no longer dependent on the system's activity and can be expressed as a system-wide constant term W_0, called *background power.* The total power dissipation of the system is:

$$W_{total} = W_0 + W(\overline{n}), \tag{21}$$

$W(\overline{n})$ is the total effective power of active cores – this is the focus of our models. The constant term of background power W_0 can be studied separately.

5.1 Power Modelling Basics

In the normal form representation of a heterogeneous system (Section 3), the power dissipations of core types is expressed with the coefficient vector $\overline{\beta} = (\beta_1, \ldots, \beta_x)$, which defines the effective power in relation to a BCE's effective power, w, such that for all $1 \leq i \leq x$ we have effective power $w_i = \beta_i w$. This system characteristic is both hardware and application dependent.

The effective power model can be found as a time-weighted average of the sequential effective power w_s and parallel effective power w_p of the system:

$$W(\overline{n}) = \frac{w_s t'_s(n) + w_p t'_p(n)}{t'_s(n) + t'_p(n)}, \tag{22}$$

where $t'_s(n)$ and $t'_p(n)$ are the speedup-dependent times required to execute sequential and parallel parts of the extended workload respectively.

In a homogeneous system: $w_s = w$, $w_p = nw$. In a heterogeneous system with the core type s executing the sequential code: $w_s = \beta_s w$ and $w_p = N_\beta w$, which gives for the balanced case of parallel execution (11):

$$N_\beta = \sum_{i=1}^{x} \beta_i n_i \tag{23}$$

For equal-share execution (10), N_β is calculated as follows:

$$N_\beta = \min \overline{\alpha} \cdot \sum_{i=1}^{x} \frac{\beta_i n_i}{\alpha_i}. \tag{24}$$

N_β is the *power-equivalent number* of BCEs. Heterogeneous power models will transform into homogeneous if $\alpha_s = \beta_s = 1$ and $N_\alpha = N_\beta = n$.

The full list of power-related variables used in the paper can be found in Table 3.

5.2 Power Distribution and Scaling Models

We express the scaling of effective power in the system via the speedup and the *power distribution* characteristic function $D_w(\overline{n})$:

$$W(\overline{n}) = w D_w(\overline{n}) S(\overline{n}). \tag{25}$$

$D_w(\overline{n})$ represents the relation between the power and performance in a heterogeneous configuration. Since the speedup models are known from Section 4, this section focuses on finding the matching power distribution functions.

From (22), we can find that in the general case:

$$D_w(\overline{n}) = \left(\beta_s t'_s(n) + N_\beta t'_p(n)\right) \cdot \frac{\theta}{I'}, \tag{26}$$

thus substituting the workload scaling definition (14) and execution times (15) will give us:

$$D_w(\overline{n}) = \frac{\frac{\beta_s}{\alpha_s}(1-p) + p g(\overline{n}) \frac{N_\beta}{N_\alpha}}{(1-p) + p g(\overline{n})}. \tag{27}$$

It is worth noting that for homogeneous systems, $D_w(\overline{n}) = 1$ in all cases, and the effective power equation will transform into:

$$W(\overline{n}) = w S(\overline{n}), \tag{28}$$

i.e. in homogeneous systems the power scales in proportion to the speedup.

Power distribution for Amdahl's workload: For Amdahl's workload, $g(\overline{n}) = 1$, hence the power distribution function becomes:

$$D_w(\overline{n}) = \frac{\beta_s}{\alpha_s}(1-p) + p \cdot \frac{N_\beta}{N_\alpha} \tag{29}$$

Power distribution for Gustafson's workload: Following the same general form (27) for the effective power equation, we can find power distribution functions $D_w(\overline{n})$ for two cases of workload scaling described in Section 4.4.

For the classical scaling mode:

$$D_w(\overline{n}) = \frac{\beta_s(1-p) + pN_\beta}{\alpha_s(1-p) + pN_\alpha}. \tag{30}$$

For the purely parallel scaling mode:

$$D_w(\overline{n}) = \frac{\beta_s(1-p) + (\alpha_s - (1-p))N_\beta}{\alpha_s(1-p) + (\alpha_s - (1-p))N_\alpha}. \tag{31}$$

6 CPU-only Experimental Validations

This section validates the models presented in Sections 4 and 5 against a set of experiments on a real heterogeneous platform. In these experiments, the goal is to determine the accuracy of the models when all model parameters, such as parallelization factor p, are under control.

6.1 Platform Description

This study is based on a multi-core mobile platform, the Odroid-XU3 board [19]. The board centres around the 28nm application processor Exynos 5422. It is an SoC hosting an ARM big.LITTLE heterogeneous processor consisting of four low power Cortex A7 cores (C0 to C3) with the maximum frequency of 1.4GHz and four high performance Cortex A15 cores (C4 to C7) with the maximum frequency of 2.0GHz. There are compatible Linux and Android distributions available for Odroid-XU3; in our experiments we used Ubuntu 14.04. This SoC also has four power domains: A7, A15, GPU, and memory power domains. The Odroid-XU3 board allows per-domain DVFS using predefined voltage-frequency pairs.

The previous assumption by Hill and Marty for heterogeneous architectures, shown in Figure 1(b), cannot describe systems such as big.LITTLE. Our models do not suffer from these restrictions and can be applied to big.LITTLE and similar structures.

6.2 Benchmark and Model Characterization

The models operate on application- and platform-dependent parameters, which are typically unknown and imply high efforts in characterization. However, in order to prove that the proposed models work, it is sufficient to show that, if $\overline{\alpha}$, $\overline{\beta}$ and p are defined, the performance and power behaviour of the system follows the models' prediction. These parameters can be fixed by a synthetic benchmark. This benchmark does not represent realistic application behaviour and was designed only for validation purposes. Experiments with realistic examples are presented in Section 8.

The model characterization is derived from single core experiments. These characterized models are used to predict

TABLE 4
Characterization experiments: single core execution

benchmark	sqrt		int		log	
base workload	40000		40000		40000	
core type i	A7	A15	A7	A15	A7	A15
measured execution time, ms	49969	53206	52844	42665	41820	23506
measured active power, W	0.2655	0.8361	0.2760	0.8305	0.3036	0.9496
power measurement std dev	0.82%	0.18%	0.96%	0.87%	0.93%	0.42%
calculated effective power, W	0.1158	0.4887	0.1264	0.4830	0.1540	0.6022
α_i	1	0.9392	1	1.2386	1	1.7791
β_i	1	4.2183	1	3.8221	1	3.9094

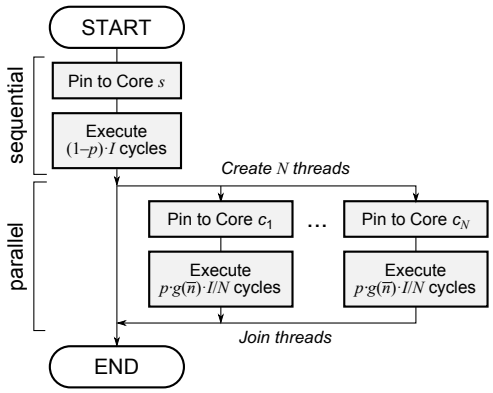

Fig. 3. Synthetic application with controllable parallelization factor and equal-share workload distribution. Parameter p, workload size I and scaling $g(\overline{n})$, the number of threads (cores) N, and the core allocation $s, \overline{c} = (c_1, \ldots, c_N)$ are specified as the program arguments.

multi-core execution in different core configurations. The predictions are then cross-validated against experimental results.

6.2.1 Controlled parameters

The benchmark has been developed specifically for these experiments in order to provide control over the parallelization parameter p. Hence, p is not a measured parameter, but a control parameter that tells the application the ratio between the parallel (multi-threaded) and sequential (single thread) execution.

The application is based on POSIX threads, and its flow is shown in Figure 3. Core configurations, including homogeneous and heterogeneous, can be specified per application run as the sequential execution core s and the set of core allocations $\overline{c} = (c_1, \ldots, c_N)$, where N is the number of parallel threads; $s, c_j \in \{\mathrm{C1}, \ldots, \mathrm{C7}\}$ for $1 \leq j \leq N$. These variables define $\overline{n}$ used in the models. We do not shut down the cores and use per-thread core pinning via `pthread_attr_setaffinity_np` to avoid unexpected task migration. To improve experimental setup and reduce the interference, we reserve one A7 core (C0) for OS and power monitors, hence it is not used to run the experimental workloads, and the following results show up to 3 A7 cores. The source code for the benchmark is available online [26].

The workload size I and the workload scaling $g(\overline{n})$ are also given parameters, which are used to test Gustafson's models against the Amdahl's law. The application implements three workload functions: square root calculation (sqrt), integer arithmetic (int), and logarithm calculation (log) repeated in a loop. These computation-heavy tasks use minimal memory access to reduce the impact of hardware on the controlled p. A fixed number of loop iterations represents one workload item. The functions are expected to give different performance characteristics, hence the characterization and cross-validation experiments are done separately for each function.

Figure 3 shows equal-share workload distribution, where each parallel thread receives equal number of $pg(\overline{n})\frac{I}{N}$ workload items. This execution gives N_α and N_β that correspond to naïve load balancing according to (10) and (24). Additionally, after collecting the characterization data for $\overline{\alpha}$, we implemented a version that uses $\overline{\alpha}$ to do optimal (balanced) workload distribution by giving each core $c_j \in \overline{c}$ a performance-adjusted workload of $pg(\overline{n})\frac{I}{N} \cdot \frac{\alpha_j}{A}$, where $A = \sum_{j=1}^{N} \alpha_j$. This execution follows different N_α and N_β, which can be calculated from (11) and (23).

6.2.2 Relative performances of cores

All experiments in this section are run with both A7 and A15 cores at 1.4GHz. Running both cores at the same frequency exposes the effects of architectural differences on the performance. In this study, we set BCE to A7, hence $\alpha_{\mathrm{A7}} = 1$; and α_{A15} can be found as a ratio of single core execution times $\alpha_{\mathrm{A15}} = t_{\mathrm{A7}}(1)/t_{\mathrm{A15}}(1)$, as shown in Table 4. The three different functions provide different α_{A15} values.

It can be seen that A15 is unsurprisingly faster than A7 for integer arithmetic and logarithm calculation, however the square root calculation is faster on A7. This is confirmed multiple times in many experiments. We did not fully investigate the reason of this behaviour since the board's production and support have been discontinued, and this is in any case outside the scope of this paper. A newer version of the board, Odroid-XU4, which is also built around Exynos 5422, does not have this issue. It is important to note that we compiled all our benchmarks using the same `gcc` settings. We include this case of non-standard behaviour in our experiments to explore possible negative impacts on the performance modelling and optimization.

6.2.3 Core idle and active powers

The Odroid-XU3 board provides power readings per power domain, i.e. one combined reading per core type, from which it is possible to derive single core characteristic values w_0 and w.

Idle powers are determined by averaging over 1min of measurements while the platform is running only the operating system and the power logging software. The idle

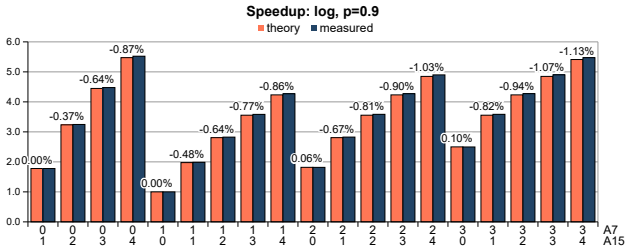

Fig. 4. Speedup validation results for the heterogeneous Amdahl's law showing percentage error of the theoretical model in relation to the measured speedup.

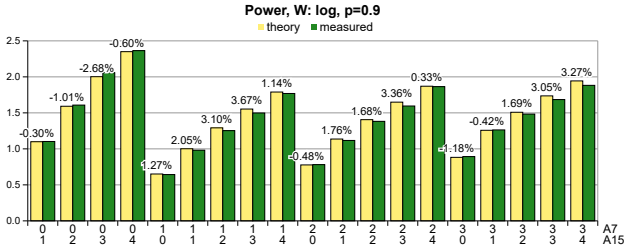

Fig. 5. Total power dissipation results for the heterogeneous Amdahl's law showing percentage error of the theoretical model in relation to the measured power.

power values are $w_{0,\mathrm{A7}} = 0.1496\mathrm{W}$ and $w_{0,\mathrm{A15}} = 0.3474\mathrm{W}$, which are used across all benchmarks. The standard deviation of the idle power measurements is 1.22% of the mean value.

Effective powers $w_{\mathrm{A7}}, w_{\mathrm{A15}}$ are calculated from the measured active powers by subtracting idle power according to (21). The power ratios are then found as $\beta_{\mathrm{A7}} = 1$ and $\beta_{\mathrm{A15}} = w_{\mathrm{A15}}/w_{\mathrm{A7}}$; the values are presented in Table 4.

6.3 Amdahl's Workload Outcomes

A large number of experiments have been carried out covering all functions (sqrt, int, log) in all core configurations, and repeated for $p = 0.3$ and $p = 0.9$. This set of runs use a fixed workload of 40000 items with equal-share workload distribution between threads. Model predictions and experimental measurements for a single example are shown in Figures 4 and 5; the full data set can be found in the Appendix (Figures 13–16). The measured speedup is calculated as the measured time for a single A7 core execution $t_{\mathrm{A7}}(1)$, shown in Table 4, over the benchmark's measured execution time $t(\overline{n})$.

The observations validate the model (13) by showing that the differences between the model predictions and the experimental measurements are very small. The speedup error is 0.2% on average across all core combinations with the worst case of 1.13%, and the power error never exceeds 5.6%, which is comparable to the standard deviation of the characterization measurements. A possible explanation for the low error values may be that our synthetic benchmark produces very stable $\overline{\alpha}$ and $\overline{\beta}$, and accurately emulates p. However, these small errors also prove that the model can be used with high confidence if it is possible to track these parameters. The model can also be confidently used in reverse to derive parallelization and performance properties of the system from the speedup measurements, as demonstrated in Section 8.

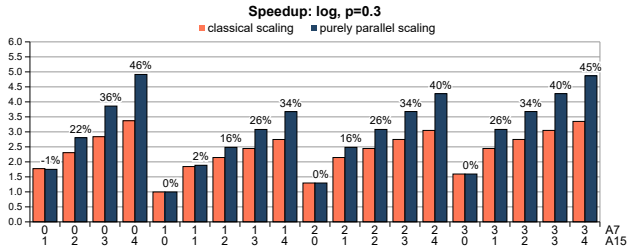

Fig. 6. Gustafson's model outcomes showing the measured speedup gain from using the purely parallel workload scaling compared to the classical scaling.

The counter intuitive result for 7-core (three A7 cores and four A15 cores) execution having lower power dissipation than four A15 cores and no A7 cores can be explained by the equal-share workload distribution. Because the parallel workload is equally split between these cores, the A15 cores finish early and wait for A7 cores. This idling reduces the average total power dissipation, however it implies that intelligent workload distribution can improve core utilization by scheduling more tasks to A15 cores than to A7 ones so that they finish at the same time. This is investigated in Section 6.5.

6.4 Gustafson's Workload Outcomes

Two sets of experiments have been carried out to validate heterogeneous Gustafson's models in both purely parallel and classical workload scaling modes described in Section 4.4. The initial workload I is set to 40000, and the scaled workload I' is defined by (14). These experiments also use equal-share workload distribution and s is fixed to A15.

The measured speedup is calculated as the ratio of performances according to (1), or as the time ratio multiplied by the workload size ratio: $S(\overline{n}) = (t_{\mathrm{A7}}(1)/t(\overline{n})) \cdot (I'/I)$. The observed errors are similar to the Amdahl's model with the speedup estimated within 3.21% error (0.54% average) and the power dissipation estimated within 6.23% difference between the theory and the measurements. A complete set of data can be found in the Appendix (Figures 17–20).

Figure 6 compares the speedup between two workload scaling modes for $p = 0.3$. The purely parallel scaling has more effect for less parallelizable applications as it focuses on reducing the sequential part of the execution, hence the experiments with $p = 0.9$ show insignificant gain in the speedup and are not presented here. Even though the purely parallel scaling is harder to achieve in practice as it requires the knowledge of p, it provides a highly significant speedup gain, especially if the difference between the core performances is high, which, in the case of log, gives almost 50% better speedup.

6.5 Balanced Execution

Previously described experiments use equal-share workload distribution, which is simpler to implement, but results in faster cores being idle while waiting for slower cores. The balanced distribution, defined in (11), gives the optimal speedup for a given workload. This section implements balanced distribution of a fixed workload and compares it to the equal-share distribution outcomes of Amdahl's law.

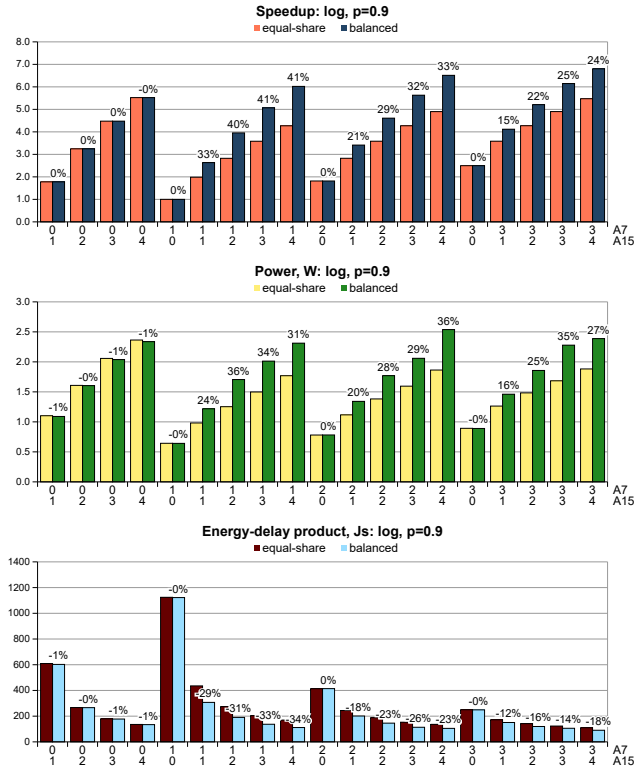

Fig. 7. Comparison of the measured speedup, power, and energy between equal-share and balanced execution.

The results are presented for $p = 0.9$, as it provides larger differences for this scenario.

In terms of model validation, the results are also very accurate, giving up to 4.63% error in power estimation and within 1.3% error for the speedup. Figure 7 explores the differences between the equal-share and optimal (balanced) cases of workload distribution in terms of performance and energy properties of the system. The balanced distribution gives up to 41% increase in the speedup. The average power dissipation is also increased up to 36% as the cores are exercised with as little idling as possible. Energy-delay product (EDP) is an optimization metric that improves energy and performance at the same time and is calculated as $W_{total} \cdot (t'(\overline{n}))^2$. The results are showing up to 34% improvement in EDP for balanced execution.

7 CPU-GPU Experimental Validations

The previous section explores the heterogeneity within the devices having the same instruction set. However, many modern platforms also include specialized accelerators such as general purpose GPUs.

OpenCL programming model [27] enables cross-platform development for parallelization by introducing the notion of a *kernel*. A kernel is a small task written in a cross-platform language that can be compiled and executed in parallel on any OpenCL device. It also provides a hardware abstraction level. GPU devices often have a complex hierarchy of parallel computation units: a few general purpose units can have access to a multitude of shader ALUs, which in turn implement vector instructions that may also be used to parallelize scalar computation. As the result, behind the OpenCL abstraction, we consider n_i not as the number of device cores but as a *degree of parallelism* – the number of kernels that can be executed in parallel.

TABLE 5
OpenCL device capabilities

core type i	CPU	IntGPU	Nvidia
device name	Intel Core i7-3520M	Intel HD Graphics 4000	GeForce GT 640M
max core freq	2.9GHz	350MHz	708MHz
compute units	4 (2+hyper)	16	2 (384 shaders)
max workgroup	1024	512	1024
max n_i	1	256	1024 (log: 64)

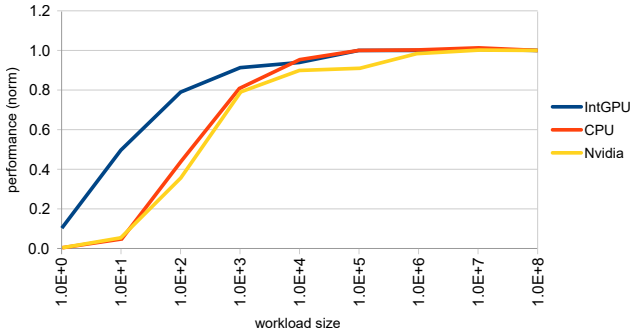

Fig. 8. The effect of OpenCL overheads on performance, can be ignored for sufficiently large workload sizes.

This section presents the experimental validation using the synthetic benchmark shown in Figure 3, but reimplemented in OpenCL with kernels replacing POSIX threads. The kernels implement the same looped computation (sqrt, int, and log). The source code for OpenCL version is also available [26].

7.1 Platform Description and Characterization

The experiments presented in this section have been carried out on 2012 Dell XPS 15 laptop with Intel Core-i7 CPU (denoted as CPU further in this section) and two GPUs: integrated GPU (IntGPU) and a dedicated Nvidia card (Nvidia). Table 5 shows device specifications as reported by OpenCL. The platform runs Windows 7 SP1 and uses OpenCL v1.2 as a part of Nvidia CUDA framework. The platform has no facility to measure power to the granularity required for the power model validation, hence this section is focused only on the speedup. Time measurement is done using the combination of the system time (for long intervals) and OpenCL profiling (for short intervals).

An important feature of the platform is that both GPU devices can execute the workload at the same time. This is done by individually calling `clEnqueueNDRangeKernel` on separate OpenCL device contexts. This paper's models cover this type of heterogeneity, while the reconfigurable Hill-Marty model [12] can model only one active type of parallel cores at a time. This has been the primary criterion for selecting a CPU-GPU platform for this section.

OpenCL adds overheads when scheduling the kernel code and copying data. However, we use computation-heavy benchmarks that do not scale the memory requirement with the workload, hence the overhead is constant, and becomes negligible if the primary computation is large enough. A series of experiments has been carried out to

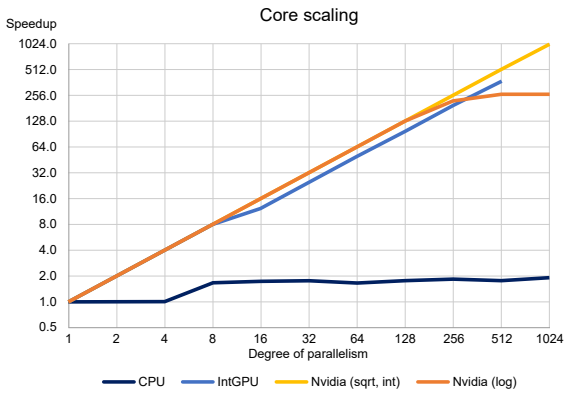

Fig. 9. Investigating the scalability potential for the requested $p = 1$.

TABLE 6
OpenCL characterization experiments

bench	core type i	workload	exec time, ms	α_i
sqrt	CPU	$8.0 \cdot 10^7$	3335	24.351
	IntGPU	$4.0 \cdot 10^6$	4060	1
	IntGPU16+		5281	0.769
	Nvidia	$8.0 \cdot 10^7$	5421	14.980
int	CPU	$8.0 \cdot 10^7$	953	44.819
	IntGPU	$8.0 \cdot 10^6$	4273	1
	IntGPU16+		5553	0.769
	Nvidia	$8.0 \cdot 10^7$	5421	7.881
log	CPU	$8.0 \cdot 10^7$	2158	42.194
	IntGPU	$8.0 \cdot 10^6$	4554	1
	IntGPU16+		5318	0.856
	Nvidia	$1.0 \cdot 10^6$	4613	0.247

find out the smallest required computation for OpenCL overheads to be negligible: 10^6 work items, as demonstrated in Figure 8.

Table 5 reports the max workgroup size, which represents the maximum number of "parallel" kernels, although OpenCL does automatic sequentialization if there are not enough real computation units. We experimentally find the real maximal degree of parallelism n_i for each benchmark by attempting to execute $p = 1$ workload and increasing the number of cores until the scaling is no longer linear. Figure 9 shows the outcome. The first observation is that CPU does not scale well in OpenCL, hence it has been decided to limit CPU to a single core used only for sequential execution. Nvidia scales perfectly for sqrt and int to its maximum allowed workgroup, but with log its performance starts to drop after 64 and completely flattens at 256. An interesting behaviour is observed with IntGPU: starting from 16 cores its performance drops by 25% (15% with log), but otherwise the scaling continues to be perfectly linear up to 256 and slightly dips at 512. We model this by representing IntGPU as two devices, as shown in Table 6. IntGPU is used as BCE, and the performance ratios $\overline{\alpha}$ are calculated as the ratio of execution times from single core experiments (except for IntGPU16+, which uses 64-core execution time).

7.2 Speedup Validation Outcomes

Due to the sheer amount of work, it is not practical to test all possible core combinations. Instead, we select configurations evenly distributed across the range. The following models have been experimentally validated: equal-share Amdahl's law and balanced Amdahl's law. Every core type

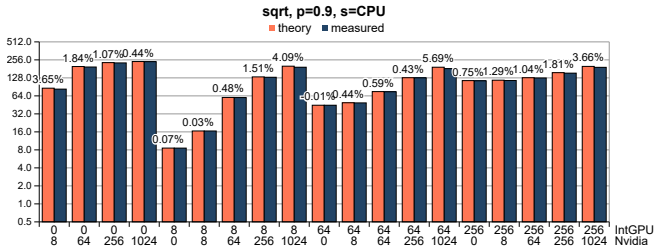

Fig. 10. Speedup validation results for the heterogeneous Amdahl's law in the OpenCL platform.

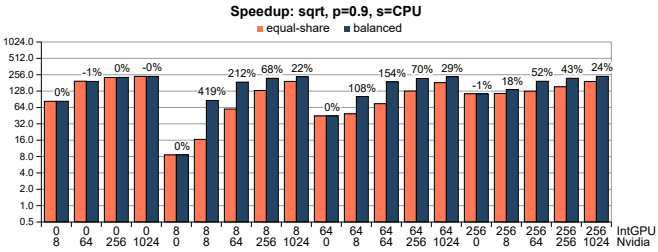

Fig. 11. Comparison of the measured speedups between equal-share and balanced execution in the OpenCL platform.

is tested in the role of sequential executor s in the case of $p = 0.9$. For $p = 0.3$, CPU is always used for s as the fastest of the devices. The speedup is calculated against the single core IntGPU experiment. Since the other devices are generally much faster, and Nvidia is capable of executing 1024 parallel kernels, the observed maximum speedup is 395.7 for Amdahl's workload, $p = 0.9$.

Figure 10 shows a typical result for Amdahl's law; the full set of results can be found in the Appendix (Figures 23–25). On average, the experiments with Amdahl's law show 1% error across the tested core combinations, but going up to 6-8% in a few points. The performance difference between the balanced and equal-share workload distributions is presented in Figure 11. Given the core performances of IntGPU and Nvidia are very different, load balancing plays a crucial role, and can provide over 400% performance boost in some cases.

8 Realistic Application Workloads

This section is focused on experiments with realistic workloads based on the Parsec benchmark suite [28]. Parsec benchmarks are designed for parallel multi-threaded computation and include diverse workloads that are not exclusively focused on high performance computing. Each application is supplied with a set of pre-defined input data, ranging from small sizes (test) to large (simlarge) and very large (native) sizes. Each input is assumed to generate a fixed workload on a given system. To our knowledge, Parsec benchmarks do not implement workload scaling to Gustafson's or Sun-Ni's models, hence this section is focused on Amdahl's law only.

In our experiments we run a subset of Parsec benchmarks (ferret, fluidanimate, and bodytrack) and use simlarge input. The selected benchmarks are representative of CPU intensive, memory intensive, and combined CPU-memory applications respectively, hence cover a wide range of workload types.The number of threads in each run is

TABLE 7
Characterization of Parsec benchmark parallelizability from homogeneous system setup

	A7					A15					
app	$S(2)$	$S(3)$	$S(4)$	p_{A7}		$S(2)$	$S(3)$	$S(4)$	p_{A15}		α_{15}
bodytrack	1.8787	2.6484	3.3211	0.9336	±0.0018	1.7980	2.4447	3.0090	0.8881	±0.0021	1.9946
ferret	1.8833	2.6716	3.3706	0.9381	±0.0004	1.9111	2.7576	3.4400	0.9518	±0.0060	1.8830
fluidanimate	1.5749	—	2.2288	0.7326	±0.0025	1.4531	—	1.9443	0.6356	±0.0120	1.8186

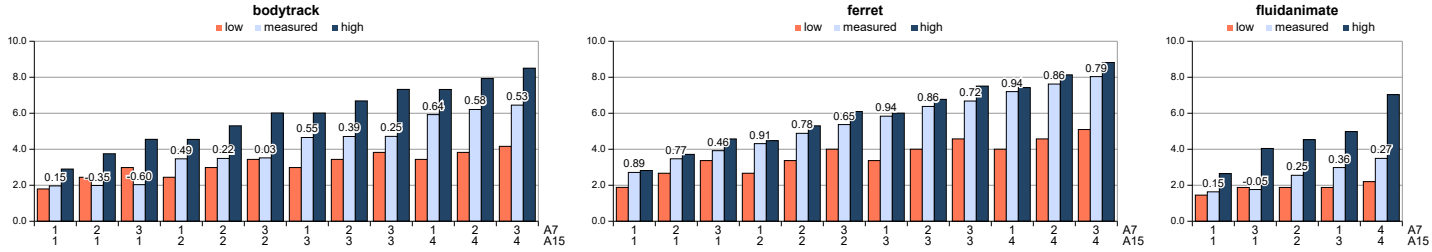

Fig. 12. Parsec speedup range results from heterogeneous system setup determining q – the quality of the system load balancer

set to match the number of active cores; fluidanimate can only run a power-of-two number of threads. Core pinning is done at the application level using the `taskset` command in Linux. The command takes a set of cores as an argument and ensures that every thread of the application is scheduled onto one of these cores. However, the threads are still allowed to move between the cores within this set due to the influence of the system load balancer [25]. This is different from the synthetic benchmark described in Section 6, which performed pinning of individual threads, one thread per core.

In this work, we do not study the actual algorithm of the load balancing or the internal structure of Parsec benchmarks, hence the workload distribution between the cores is considered a black box function: N_α is unknown. Section 4.1 addressed this issue by providing the range of values for N_α. The minimum value corresponds to equal-share workload distribution and gives the lower speedup limit $S_{\mathrm{low}}(\overline{n})$; the maximum value is defined by the balanced workload and gives the higher speedup limit $S_{\mathrm{high}}(\overline{n})$.

The goal of the following experiments is to calculate these limits and to find how the real measured speedup fits in the range. The relation provides a quality metric q for the load balancing algorithm, where $q = 1$ corresponds to the theoretically optimal load balancer, and $q = 0$ is equivalent to a naïve approach (equal-share). Negative values may also be possible and show that the balancing algorithm is not working properly and creates an obstacle to the workload execution. The metric q is calculated as follows:

$$q = \frac{S(\overline{n}) - S_{\mathrm{low}}(\overline{n})}{S_{\mathrm{high}}(\overline{n}) - S_{\mathrm{low}}(\overline{n})}. \tag{32}$$

The motivation for load balancing is to improve speedup by approaching the balanced workload behaviour. Hill-Marty [12] and related existing work [13], [14] covering core heterogeneity all assume that the workload is already balanced in their models, implying $q = 1$. This work makes no such assumption and studies real load balancer behaviours for different benchmarks, using novel models facilitating quantitative comparisons.

8.1 Model Characterization

The Parsec experiments are executed on the Odroid XU3 platform described in Section 6. The model characterization is obtained from the homogeneous configuration experiments, and then the models are used to predict system behaviour in heterogeneous configurations. Each benchmark is studied independently. Table 7 shows the obtained parameter values.

A7 is once again used as BCE, $\alpha_{\mathrm{A7}} = 1$; α_{A15} values are derived from single core executions as the time ratio $t_{\mathrm{A7}}(1)/t_{\mathrm{A15}}(1)$. Core frequencies of both A7 and A15 are set to 1.4GHz.

Parameter α_s is not known because it is not guaranteed that the sequential part of the workload will be executed on the fastest core, and it is also possible for the sequential execution to be re-scheduled to different core types, however α_s must stay within the range of $[\alpha_{\mathrm{A7}}, \alpha_{\mathrm{A15}}]$.

Parallelization factor p is determined from the measured speedup $S(n)$ for $n > 1$ solving (3) for p. For different values of n, the equation gives different p, however the differences are insignificant within the same type of core. On the other hand, the differences in p for different core types are substantial and cannot be ignored.

The lowest values of the model parameters p and α_s are used to calculate the lower limit of the heterogeneous speedup $S_{\mathrm{low}}(\overline{n})$, and the highest values are used to calculate $S_{\mathrm{high}}(\overline{n})$.

8.2 Load Balancer Quality

Figure 12 presents the outcomes of the experiments for the selected heterogeneous core configurations; full data set can be found in Appendix (Figure 26). Time measurements have been collected from 4 runs in each configuration to avoid any random flukes, however the results were consistent within 0.2% variability. This indicates that the system scheduler and load balancer behave deterministically in given conditions.

The graphs display the calculated speedup ranges $[S_{\mathrm{low}}(\overline{n}), S_{\mathrm{high}}(\overline{n})]$ and the measured speedup $S(\overline{n})$. The numbers represent the load balancer quality q, calculated from (32).

The first interesting observation is that the ferret benchmark is executed with very high scheduling efficiency despite the system's heterogeneity. The average value of q is 0.71 and the maximum goes to 0.94. According to the benchmark's description, its data parallelism employs a pipeline, i.e. the application implements a producer-consumer paradigm. In this case, the workload distribution is managed by the application. Consequently, the cores are always given work items to execute and the longest possible idling time is less than the execution of one item.

The observed q values never exceed 1, which validates the hypothesis that (11) refers to the optimal workload distribution and can be used to predict the system's performance capacity. The lower bound of $q = 0$ is also mostly respected. This is not a hard limit, but a guideline that separates appropriate workload distributions. This boundary is significantly violated only in one case, as described below.

Bodytrack and fluidanimate show much less efficient workload distribution, compared to ferret, and their efficiency seems to decrease when the core configuration includes more little than big cores. This effect is exceptionally impactful in the case of multiple A7 cores and one A15 core executing bodytrack, where the value of q lies far in the negative range and can serve as an evidence of load balancer malfunction. Indeed, the speedup of this four-thread execution is only slightly higher than two-threaded runs on one A7 and one A15. The execution time is close to a single thread executed on one A15 core, showing almost zero benefit from bringing in three more cores, and the result is consistent across multiple runs of the experiment. This issue requires a substantial investigation and lies beyond the scope of this paper, however it demonstrates how the presented method may help analyse the system behaviour and detect problems in the scheduler and load balancer.

9 Future work

This work lays the foundation for extending speedup, power and energy models to cover architectural heterogeneity. In order to achieve the level of model sophistication of the existing homogeneous models, more work needs to be completed. This should include the effects of memory, and inter-core dependencies. Modelling of the background power W_0 can be extended to include leakage power, which may indirectly represent the effects of temperature variations.

On-going research indicates that $\overline{\alpha}$ and $\overline{\beta}$ may not be constant across different phases of individual applications. Within this paper, these coefficients pertain to entire applications or algorithms and are hence average values. This is being investigated in more detail. How such more complex relations between the relative performances and power characteristics of different types of cores can be best incorporated in future models is being explored.

One important motivation for such more precise modelling into parts of applications is using the models in run-time management towards the optimization of goals related to performance and/or power dissipation. It is more typical for run-time control decisions to be made on regular intervals of time, unrelated to the start and completion of whole applications, hence the importance of phases within each application. On-going research shows promising directions with parallelizability-aware RTMs on homogeneous systems [29]. We will work towards generalizing this line of work into heterogeneous systems.

10 Conclusions

The models presented in the paper enhance our understanding of scalability in heterogeneous many-core systems and will be useful for platform designers and electronic engineers, as well as for system level software developers.

This paper extends three classical speedup models – Amdahl's law, Gustafson's model and Sun Ni's model – to the range of heterogeneous system configurations that can be described as a normal form heterogeneity. The provided discussion shows that the proposed models are not reducing applicability in comparison to the original models and may serve as a foundation for multiple research directions in the future. Important aspects, such as workload distribution between heterogeneous cores and various modes of workload scaling, are included in the model derivation. In addition to performance, this paper addresses the issue of power modelling by by calculating power dissipation for the respective heterogeneous speedup models.

The practical part of this work includes experiments on multi-type CPU and CPU-GPU systems pertaining to model validation and real-life application. The models have been validated against a synthetic benchmark in a controlled environment. The experiments confirm the accuracy of the models and show that the models provide deeper insights and clearly demonstrate the effects of various system parameters on performance and energy scaling in different heterogeneous configurations.

The modelling method enables the study of the quality of load balancing, used for improving speedup. A quantitative metric for load balancing quality is proposed and a series of experiments involving Parsec benchmarks are conducted. The modelling method provides quantitative guidelines of load balancing quality against which experimental results can be compared. The Linux load balancer is shown to not always provide high quality results. In certain situations, it may even produce worse results than the naïve equal-share approach. The study also showed that application-specific load balancing using pipelines can produce results of much higher quality, approaching the theoretical optimum obtained from the models.

Acknowledgement

This work is supported by EPSRC/UK as a part of PRiME project EP/K034448/1. M. A. N. Al-hayanni thanks the Iraqi Government for PhD studentship funding. The authors thank Ali M. Aalsaud for useful discussions.

References

[1] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). ACM, 1967, pp. 483–485.

[2] 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.

[3] J. L. Gustafson, "Reevaluating amdahl's law," *Communications of the ACM*, vol. 31, no. 5, pp. 532–533, 1988.
[4] X.-H. Sun and L. M. Ni, "Another view on parallel speedup," in *Supercomputing'90., Proceedings of*. IEEE, 1990, pp. 324–333.
[5] S. Borkar, "Thousand core chips: A technology perspective," in *Proceedings of the 44th Annual Design Automation Conference*, ser. DAC '07. New York, NY, USA: ACM, 2007, pp. 746–749.
[6] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.
[7] J. G. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *Annals of the History of Computing, IEEE*, vol. 33, no. 3, pp. 46–54, 2011.
[8] F. J. Pollack, "New microarchitecture challenges in the coming generations of cmos process technologies (keynote address)," in *Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society, 1999, p. 2.
[9] P. Greenhalgh, *big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7 – Improving Energy Efficiency in High-Performance Mobile Platforms*, ARM, 2011, white Paper.
[10] "Juno ARM development platform SoC technical overview," ARM, Tech. Rep., 2014.
[11] R. H. Dennard, V. Rideout, E. Bassous, and A. Leblanc, "Design of ion-implanted mosfet's with very small physical dimensions," *Solid-State Circuits, IEEE Journal of*, vol. 9, no. 5, pp. 256–268, 1974.
[12] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, no. 7, pp. 33–38, 2008.
[13] X.-H. Sun and Y. Chen, "Reevaluating amdahl's law in the multicore era," *Journal of Parallel and Distributed Computing*, vol. 70, no. 2, pp. 183–188, 2010.
[14] N. Ye, Z. Hao, and X. Xie, "The speedup model for manycore processor," in *Information Science and Cloud Computing Companion (ISCC-C), 2013 International Conference on*. IEEE, 2013, pp. 469–474.
[15] D. H. Woo and H.-H. S. Lee, "Extending amdahl's law for energy-efficient computing in the many-core era," *Computer*, no. 12, pp. 24–31, 2008.
[16] U. Gupta, S. Korrapati, N. Matturu, and U. Y. Ogras, "A generic energy optimization framework for heterogeneous platforms using scaling models," *Microprocess. Microsyst.*, vol. 40, no. C, pp. 74–87, Feb. 2016.
[17] 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," vol. 3, pp. 30–46, 01 2013.
[18] M. A. N. Al-Hayanni, A. Rafiev, R. Shafik, and F. Xia, "Power and energy normalized speedup models for heterogeneous many core computing," in *16th International Conference on Application of Concurrency to System Design (ACSD)*, June 2016, pp. 84–93.
[19] "Odroid XU3," http://www.hardkernel.com/main/products.
[20] X.-H. Sun and L. M. Ni, "Scalable problems and memory-bounded speedup," *Journal of Parallel and Distributed Computing*, vol. 19, no. 1, pp. 27–37, 1993.
[21] A. Morad, T. Y. Morad, Y. Leonid, R. Ginosar, and U. Weiser, "Generalized multiamdahl: Optimization of heterogeneous multi-accelerator soc," *IEEE Computer Architecture Letters*, vol. 13, no. 1, pp. 37–40, Jan 2014.
[22] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981.
[23] J. Baeten, C. A. Middelburg, and E. T. Netherlands, "Process algebra with timing: Real time and discrete time," in *Handbook of Process Algebra*. Elsevier, 2000, pp. 627–684.
[24] K. Georgiou, S. Kerrison, Z. Chamski, and K. Eder, "Energy transparency for deeply embedded programs," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 14, no. 1, pp. 1–26, 4 2017.
[25] J.-P. Lozi, B. Lepers, J. Funston, F. Gaud, V. Quéma, and A. Fedorova, "The linux scheduler: A decade of wasted cores," in *Proceedings of the Eleventh European Conference on Computer Systems*, ser. EuroSys '16. New York, NY, USA: ACM, 2016, pp. 1:1–1:16.
[26] "PThreads benchmark," https://github.com/ashurrafiev/PThreads.
[27] "OpenCL overview," https://www.khronos.org/opencl.
[28] C. Bienia and K. Li, "Parsec 2.0: A new benchmark suite for chip-multiprocessors," in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, June 2009.
[29] 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.

Ashur Rafiev has received his PhD in 2011 in the School of Electrical, Electronic and Computer Engineering, Newcaste University. At the moment, he works in the School of Computing Science, Newcastle University, as a Research Associate. His research interest is focused on power modelling and hardware-software co-simulation of many-core systems.

Mohammed Al-hayanni (Student Member, IEEE and IET) is an experienced electronics, computer and software engineer. He is currently studying for his PhD with the School of Electrical and Electronic Engineering, Newcastle University. His research interests include developing practically validated robust performance adaptation models for energy-efficient many-core computing systems.

Fei Xia is a Senior Research Associate with the School of Electrical and Electronic Engineering, Newcastle University. His research interests are in asynchronous and concurrent systems with an emphasis on power and energy. He holds a PhD from King's College, London, an MSc from the University of Alberta, Edmonton, and a BEng from Tsinghua University, Beijing.

Rishad Shafik (MIEE'06-to-date) is a Lecturer in Electronic Systems at Newcastle University. His research interests include design of intelligent and energy-efficient embedded systems. He holds a PhD and an MSc from Southampton Uni., and a BEng from IUT, Bangladesh. He has authored 80+ research articles published by IEEE/ACM, and is the co-editor of "Energy-efficient Fault-Tolerant Systems". He is chairing DFT'17 (http://www.dfts.org), to be held in Cambridge, UK.

Alexander Romanovsky is a Professor at Newcastle University, UK and the leader of the Secure and Resilient Systems group at the School of Computing. His main research interests are system dependability, fault tolerance, software architectures, system verification for safety, system structuring and verification of fault tolerance. He is a member of the editorial boards of Computer Journal, IEEE Transactions on Reliability, Journal of System Architecture and International Journal of Critical Computer-Based Systems.

Alex Yakovlev is a professor in the School of Electrical and Electronic Engineering, Newcastle University. His research interests include asynchronous circuits and systems, concurrency models, energy-modulated computing. Yakovlev received a DSc in Engineering at Newcastle University. He is a Fellow of IEEE, Fellow of IET, and Fellow of RAEng. In 2011-2013 he was a Dream Fellow of the UK Engineering and Physical Sciences Research Council (EPSRC).