µSystems Research Group

School of Engineering

Extending Multi-fraction Speedup Models to Normal Form Heterogeneity

A. Rafiev, M. A. N. Al-hayanni, F. Xia,
R. Shafik, A. Romanovsky, A. Yakovlev

Technical Report Series

NCL-EEE-MICRO-TR-2018-211

October 2018

Contact: ashur.rafiev@ncl.ac.uk, m.a.n.al-hayanni@ncl.ac.uk, fei.xia@ncl.ac.uk

Supported by EPSRC grants EP/K034448/1 and EP/N023641/1

NCL-EEE-MICRO-TR-2018-211

µSystems Research Group
School of Engineering
Merz Court
Newcastle University
Newcastle upon Tyne, NE1 7RU, UK

http://async.org.uk/

Extending Multi-fraction Speedup Models to Normal Form Heterogeneity

A. Rafiev, M. A. N. Al-hayanni, F. Xia, R. Shafik, A. Romanovsky, A. Yakovlev

October 2018

Abstract

Amdahl's Law is the classical model of parallelization speedup. Its simplistic assumptions have caused it to be extended on multiple occasions to cover wider workload and system realities. This report describes the newest step in the continued extension and generalization of Amdahl's Law to better cover modern multi- and many-core architectures executing realistic workloads. The key contribution is the vectorization of both parameters of Amdahl's Law, which allows the representation of wide system architecture heterogeneity and the effects of the parallelism of workloads.

Classical Amdahl's Law [1] is based on dividing the workload into the parallel f and sequential $(1-f)$ fractions. Classical Amdahl's speedup model for n-core system is

$$S(n) = \left[(1-f) + \frac{f}{n}\right]^{-1} . \tag{1}$$

One problem with this model is that most workloads do not contain infinitely parallel fractions. Workloads and parts thereof usually display finite parallelism. Parallelism p is defined as the maximum speedup for a workload given an infinite number of cores [2]:

$$p = \frac{T_1}{T_\infty} , \tag{2}$$

where T_1 is the time required to execute the workload on one core and T_∞ is the time required to execute the same workload on an infinite number of cores. Parallelism can be intuitively understood as the number of threads a workload has for mapping onto multiple cores. This means that a workload displays different f, depending on the relationship of its parallelism p and the number of cores n onto which it is mapped.

A number of research teams attempted to tackle this issue by introducing multiple parallel fractions $f_1, \ldots, f_n$ so that, for any $1 \leq j \leq n$, f_j is the fraction of the workload that is executed on j cores [3, 4, 5]. In this report, we call such models *multi-fraction speedup models*, but focus primarily on the work of Yun et al. [5].

In their paper, Yun et al. present a method of extracting f values from the application's task graph. Figure 1 shows an example of a task graph. For simplicity, the tasks are shown to be of the

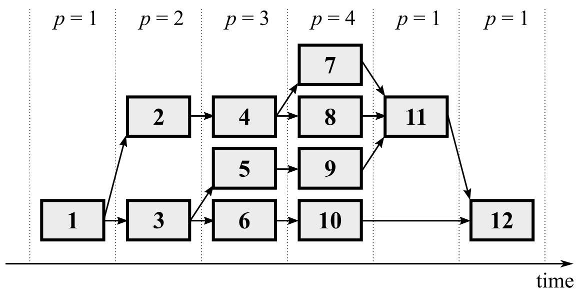

Figure 1: An example of a task graph and respective instantaneous values of parallelism.

same size, but this is not required by the method. The method groups the tasks with the same p under the respective f_p and then normalizes the value by the total number of tasks. In the shown example, the f values are: $f_1 = 3/12 = 0.25$, $f_2 = 2/12 - 0.166...$, $f_3 = 3/12 = 0.25$, and $f_4 = 4/12 = 0.333...$.

The multi-fraction homogeneous speedup model proposed by Yun et al. is

$$S(n) = \left[\sum_{j=1}^{n} \frac{f_j}{j} \right]^{-1} . \tag{3}$$

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

$$\sum_{j=1}^{n} f_j = 1 . \tag{4}$$

It is important to note that f values are dependent on n: if the task graph allows more parallelism than the number of available cores, the mapping to f values should consider the effects of scheduling by respectively "stretching" the task times. Therefore j never exceeds n regardless of the parallelism in the task graph.

An interesting consequence of this dependency that was overlooked by Yun et al. is that their method implicitly supports workload scaling models like Gustafson's [6] and Sun-Ni's [7]. The classical workload scaling models, based on Amdahl's Law, assume that f does not change with the number of cores, hence they introduce a separate parameter – the scaling function $g(n)$ – so the scaled parallel part of the workload becomes $f \cdot g(n)$. This parameter is not required for the Yun et al. model as the f values already can represent the workload that changes with n; the only required extension is to allow the task graph to change with the number of cores. However, an important detail is that, according to Gustafson and Sun-Ni, the workload scaling may break the condition (4). In fact, the total sums of their scaled workload fractions are always greater than 1 for $n > 1$. Therefore, in their models, they have to explicitly re-normalize the speedup, so (3) becomes

$$S(n) = \left[\sum_{j=1}^{n} f_j \right] \cdot \left[\sum_{j=1}^{n} \frac{f_j}{j} \right]^{-1} . \tag{5}$$

One can verify that (5) transforms into Sun-Ni's model by substituting f_1 with $(1 - f)$ and f_n with

$f \cdot g(n)$. For $g(n) = n$ the model further transforms into Gustafson's. However, the method of extracting f values from the task graph, proposed by Yun et al., produces normalized f values by design, so their models hold the condition (4) and do not actually require explicit re-normalization (5). In this report, we also assume (4) and do not re-normalize our speedup equations.

For the heterogeneous case, Yun et al. consider ARM big.LITTLE system configuration consisting of n_L low power LITTLE and n_b high performance big processors. They set the performance of LITTLE cores to be 1, and the relative performance of big cores as α_b. The authors also assume that the scheduling always prioritizes high performance cores. Their heterogeneous model is

$$S(n_b, n_L) = \left[\sum_{b=1}^{n_b} \frac{f_b}{\alpha_b \cdot b} + \sum_{L=1}^{n_L} \frac{f_{n_b+L}}{\alpha_b \cdot n_b + L} \right]^{-1} . \tag{6}$$

This heterogeneous model is limited to a single practical system.

A more general heterogeneity in multi-fraction approach is supported by the method known as Multiamdahl [8]. 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, $1 \le j \le n$. These arbitrary sections are capable 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_{exec} is:

$$T_{\text{exec}} = \sum_{j=1}^{n} f_j \cdot e(x_j) \ , \tag{7}$$

where $e(x_j)$ is the so-called *efficiency function* (although the name is somewhat misleading as the larger values of $e(x_j)$ 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:

$$\sum_{j=1}^{n} x_j \le X \ . \tag{8}$$

Multiamdahl paper does not explicitly specify the equation for the speedup and focuses directly at minimizing T_{exec} under the constraint (8), 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:

$$S(n) = \frac{T_1}{T_{\text{exec}}} = \left[\sum_{j=1}^{n} f_j \cdot e(x_j) \right]^{-1} . \tag{9}$$

Despite the generality of Multiamdahl, its assumption of sequentially executing hardware sections has been a major criticism against the practicality of the model [9].

The goal of our report is to extend (6) to a general representation of heterogeneity without restraining the model by any specific scheduling priorities or constraints.

In our previous work [10] we defined *normal form (NF) heterogeneity* based on x types of cores, such that for each type $1 \le i \le x$, the number of cores of this type is defined as n_i and the performance of each core of this type is defined as α_i in relation to some base core equivalent (BCE). These parameters can be viewed as vectors $\overline{n} = (n_1, \ldots, n_x)$ and $\overline{\alpha} = (\alpha_1, \ldots, \alpha_x)$. A simplified version of the NF does not group cores into types and works with n cores and a vector $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$ that defines performance coefficients for each core individually. In this report, we use the latter approach.

The original definition of NF-based speedup model uses the classical Amdahl's subdividing into sequential (single core) and fully parallel (all n cores) fractions. In order to apply multi-fraction approach to NF heterogeneity, we still require to add a general representation of scheduling for intermediate fractions where only some of the n cores are being used.

One way of doing this is to enumerate the cores in the order of their priority. However, this is not general enough as the priorities may shift depending on the number of parallel threads. For example, some run-time managers [11] use a single high performance core for sequential execution, when $p = 1$, and then shift to all low power cores for higher degrees of parallelism, when $p > 1$. We present two methods to generalize the scheduling model: core-based and configuration-based.

Core-based generalization In this type of generalization, we assume that the scheduling behavior is determined by the instantaneous parallelism. Thus, for any $1 \leq j \leq n$, we define a separate vector $\overline{\alpha_j} = (\alpha_{j1}, \ldots, \alpha_{jj})$ representing the BCE-relative performances of exactly those j cores that execute the fraction f_j. The combined performance of these cores is

$$A_j = \sum_{i=1}^{j} \alpha_{ji} \,, 1 \leq j \leq n. \tag{10}$$

With this consideration, the NF-extended Yun et al. model takes the following form:

$$S(n) = \left[\sum_{j=1}^{n} \frac{f_j}{A_j} \right]^{-1} . \tag{11}$$

An important note on the variable A_j is that is has the same semantic meaning as N_α in [10], which brings the question whether we should consider the effect of load balancing in the presented 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 Yun et al., the execution then switches to $f_{(j-k)}$, hence the load balancing is already captured by the multiple f values, and A_j always equals to the sum of performances, as in (10). This simplifies the model, but adds practical complexity to determining f values: for heterogeneous systems, the task graph needs to be analyzed with regard to the system's load balancing.

Configuration-based generalization The most general way of representing the scheduling is to build the entire model around the set $\{\overline{\alpha_1}, \ldots, \overline{\alpha_Q}\}$ of system configurations (or scheduling "policies"), where $Q \geq 1$ is the number of configurations. Each configuration defines the vector of n performance coefficients $\overline{\alpha_j} = (\alpha_{j1}, \ldots, \alpha_{jn})$, where $1 \leq j \leq Q$, in which the performances of inactive cores are set to 0. The combined performance of the configuration is the sum of its n performance coefficients:

$$A_j = \sum_{i=1}^{n} \alpha_{ji} \, , 1 \leq j \leq Q. \tag{12}$$

An important difference is that the workload fractions $f_1, \ldots, f_Q$ now correspond to the configurations rather than the numbers of cores. In other words, f_j represents the fraction of the workload that is executed in the configuration $\overline{\alpha_j}$. Configuration-based NF Yun et al. model takes the following form:

$$S\left(n\right) = \left[\sum_{j=1}^{Q} \frac{f_j}{A_j}\right]^{-1} \, . \tag{13}$$

The latter equation is closely connected with Multiamdahl model (9). Indeed, if we subdivide the resource X into Q parts instead of n and define the efficiency function as $e\left(x_j\right) = 1/A_j$, the equation (9) transforms into (13). The major difference, however, is that the constraint (8) 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 Multiamdahl approach can be solved by modifying (8) as:

$$x_j \leq X \, , \tag{14}$$

or in other words, the fraction f_j can use any amount of system resources as long as it does not exceed the system total. This can also be applied to (12) in the following form:

$$A_j \leq A_{\max} \, , \tag{15}$$

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

The core-based normal-form multi-fraction model of (11) is a special case of (13). 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 threads running on cores.

In this general understanding, a workload is executed on some machine, which has Q distinctive *modes* of operation. Each mode 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 mode j. The probability understanding extends the fraction assumption for deterministic systems to cover stochastic behaviour, and agrees well with the renormalization 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.

This broader understanding of the normal-form multi-fraction model returns to the broader understanding of the original form of Amdahl's Law (1), 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.

Acknowledgment

This work is supported by the PRiME (EPSRC grant EP/K034448/1) and STRATA (EPSRC grant EP/N023641/1) projects. M. A. N. Al-hayanni 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, pp. 19–20, Summer 2007.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithm.* The MIT Press, 2009.

[3] S. Tang, B. Lee, and B. He, “Speedup for multi-level parallel computing,” in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*, pp. 537–546, May 2012.

[4] S. Mercelis, *A systematic multi-layered approach for optimizing and parallelizing real-time media and audio applications.* PhD thesis, University of Antwerp, 2016.

[5] 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.

[6] J. L. Gustafson, “Reevaluating amdahl’s law,” *Communications of ACM*, vol. 31, pp. 532–533, May 1988.

[7] 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.

[8] T. Zidenberg, I. Keslassy, and U. Weiser, “Multiamdahl: How should i divide my heterogenous chip?,” *IEEE Computer Architecture Letters*, vol. 11, pp. 65–68, July 2012.

[9] 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.

[10] 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, pp. 436–449, July 2018.

[11] 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 *2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pp. 206–213, July 2018.