Generalizing Gustafson's Law For Heterogeneous Processors

André A. Cesta, Geraldo M. Silva

Eldorado Research Institute
Brasilia - Campinas - Porto Alegre, Brazil

{andre.cesta,geraldo.magela}@eldorado.org.br

Abstract. *In this article we show how Gustafson's Law can be generalized and extended from a function of the number of processors 'p' to a function of intrinsic microprocessor variables such as: clock speed 'c' and hardware threads 't'. This allows that the performance or benchmark results following Gustafson's Law be modeled for heterogeneous processors. Some possible applications of this theoretical work include all previous applications from Gustafson's Law and some others: hardware upgrade or migration scenario analysis; optimal hardware selection for particular workloads; sizing or recommendations for heterogeneous machines.*

Resumo. *Neste artigo, vamos mostrar como a Lei de Gustafson pode ser generalizada e estendida a partir de uma função do número de processadores 'p' para uma função de variáveis intrínsecas do microprocessador, tais como: velocidade do clock 'c' e threads do hardware 't', permitindo que o desempenho ou resultados de benchmark seguindo a Lei de Gustafson possam ser modelados para processadores heterogêneos. Dentre as possíveis aplicações deste trabalho teórico podem ser mencionadas todas as anteriores a partir da Lei de Gustafson e algumas outras: atualização de hardware ou análise do cenário de migração; seleção de hardware ideal para cargas de trabalho específicas; dimensionamento ou recomendações para máquinas heterogêneas.*

1. Introduction

In [Cesta et al. 2011] and [Cesta et al. 2012], the authors showed how computer processing scalability laws such as Amdahl's Law [Amdahl 1967] or Gunther's Universal Scalability Law (USL) [Gunther 2010] could be generalized to the case of heterogeneous processors by transforming them from 'number of processor' functions to 'intrinsic processor characteristics' functions. This paper demonstrates how Gustafson's Law [Gustafson 1988], another well-known scalability model, can also be generalized to the heterogeneous processor scenario. In this generalization, Gustafson's Law is also transformed from a function of the number of processors '*p*', as in Eq. 1 below, to a function of processor intrinsic variables such as hardware threads '*t*' and clock speed '*c*'. Our work here is theoretical in nature.

$$S(p) = p - \alpha \cdot (p - 1) \tag{1}$$

$$\alpha \in (0,1) \rightarrow \lim_{p \to \infty} S(p) = \infty \quad (2)$$

Notice in Eq. 2 that for α in between 0 and 1 exclusively, Gustafson's Law from Eq. 1 will scale to infinity as more and more processors are added. This is in contrast to Amdahl's Law which does not scale to infinity.

Amdahl's Law, USL and Gustafson's Law are all models capable of predicting throughput or scalability when the homogeneous hardware processor counts change, given a certain workload type. Depending on the workload type, and on how the software scales, certain laws may better explain the scalability phenomena than others. Typically, for CPU-bound situations, one has to assess whether the scalability for a certain workload is better modeled by Gustafson's Law or by Amdahl's Law.

While Gustafson's Law shows that some computations involving arbitrarily large data sets can be efficiently parallelized and scaled, Amdahl's Law and USL provide a counterpoint describing limits on the speed-up that parallelization provides.

The speedup $S_G(p)$ or scalability according to Gustafson is given by:

$$S_G(p) = \frac{pt(p)}{pt(1)} = \frac{ts + p \cdot tp}{ts + tp} \quad (3)$$

That is, the ratio of program size with '*p*' processors to the program size with one processor. Notice that if '*ts*' is relatively small, doubling the number of processors (*p*) from one to two will allow close to double the number of parallel tasks (*tp*) to be fit in the same time frame, keeping the same '*ts*' time for all serial tasks.

In the Methods section, we will generalize Gustafson's Law to the heterogeneous processor case. In the Discussion section, we will submit the demonstrated equation to a sensitivity analysis involving variables such as threads and clock speed. We will also issue some cautionary notes about applicability. Finally, in the 'Conclusion and Future Work' Section, we will summarize the work results and list what we believe would fit as Future Work in this line of research.

2. Methods

We define the total program size in tasks '*pt*' as:

$$pt = ts + tp \quad (4)$$

Read '*pt*' as 'program tasks', '*ts*' as the number of serial tasks and *tp* as the number of parallelizable tasks. In the original demonstration of Gustafson's Law [Gustafson 1988], the program size function '*pt*' when scaled to more processors 'p' is:

$$pt(p) = ts + p \cdot tp \quad (5)$$

Defining α, the percent of serial tasks in a one-processor situation as: $\alpha = ts/(ts+tp)$, and β the percent of parallel tasks in a one-processor situation as $\beta = tp/(ts+tp)$, with $\alpha + \beta = 1$, we have Gustafson's Law as a function of the number of processors and the percent of serial tasks α:

$$S_G(p) = \frac{ts + p \cdot tp}{ts + tp} = \frac{ts}{ts + tp} + p \cdot \frac{tp}{ts + tp} =$$
$$\alpha + p \cdot (1 - \alpha) = \alpha + p - \alpha \cdot p = \quad (6)$$
$$p - \alpha \cdot (p - 1)$$

■

We now evolve or generalize this formulation to a heterogeneous machine scenario. Considering hardware threads '*t*' as serial processor '*p*', and considering the effect of increased clock '*c*' in scaling all processor tasks, we can reformulate Eq. 5 as:

$$pt(t) = c \cdot ts + c \cdot t \cdot tp \quad (7)$$

This means that if we double the clock speed, the number of processor tasks executed would double as well; the assumption is that the processor can perform approximately twice as many serial tasks and parallel tasks.

Here are some additional assumptions required for the model above, and for our model: 1. even parallelism, the capacity of the software to allocate the parallelizable part of a workload evenly to all processors [Sun and Ni 1993]; 2. scalable multi-processor; 3. homogeneous multi-processors in a machine; 4. nominal clock (not a turbo-boosting clock) being used in the equation.

The speed-up by Gustafson for a heterogeneous machine example generates a multivariate model based on threads '*t*' and clock speed '*c*':

$$\text{speedup(t,c)} = \text{S}_\text{G}\text{(t,c)} = \frac{\text{pt(t,c)}}{\text{pt(t = 1, c = 1)}} = \frac{c \cdot ts + c \cdot t \cdot tp}{\text{ts + tp}} \quad (8)$$

Calling*: ts/(ts+tp)* = α and *tp/(ts+tp)* = (1 - α) and continuing the development from the equation we have:

$$S_G(t,c) = \frac{c \cdot ts + c \cdot t \cdot tp}{ts + tp} = c \cdot \alpha + c \cdot t \cdot (1 - \alpha) =$$
$$\alpha \cdot c + c \cdot t - c \cdot t \cdot \alpha = c \cdot (\alpha + t - t \cdot \alpha) = c \cdot (\alpha \cdot (1 - t) + t) = \quad (9)$$
$$c \cdot (t - \alpha \cdot (t - 1))$$

The above demonstration is based on the assumption that increased clock speeds will speed up every program time component, which is more likely to be true for processor-intensive programs and benchmarks. If we take this into consideration, we can decompose the program time '*pt*' in the following algebra [Cesta et al. 2012]:

$$pt(1) = tsi + tpi + tse + tpe \quad (10)$$

The program tasks '*pt*', are broken down into: 1. '*tse*', the program parts or tasks that are serial and perform external resource access; 2. '*tsi*', the serial program tasks that perform internal work with no external resource access; 3. '*tpe*', the parallelizable tasks

related to external resource access; 4. '*tpi*', the parallelizable tasks that perform internal work with no external resource access.

The program tasks '*pt*' with variable hardware threads '*t*'; clock speed '*c*'; and external resource access speed '*es*' according to Gustafson's scaling can be expressed as:

$$pt(t,c,es) = c \cdot tsi + c \cdot t \cdot tpi + es \cdot tse + es \cdot t \cdot tpe \tag{11}$$

Unlike variables '*c*' and '*t*', which map directly to nominal values for processor clock and threads, variable '*es*' will typically be influenced by the difficulty of modeling interactions between various factors. Our '*tse*' and '*tpe*' program times component will also not map exactly to all the time the program spends relying on an external resource. Because '*tse*' and '*tpe*' is defined in terms of '*es*' in the equation, it will become just a fraction of the total time spent relying on an external resource.

Continuing the demonstration from Eq. 11, we can redefine the speed-up by Gustafson for a heterogeneous machine situation − previously presented in Eq. 9 − as a new formulation that unlike Eq. 9, does not assume clock speed will speed up every program task:

$$S_G(t,c,es) = \frac{pt(t,c,es)}{pt(1,1,1)} = \frac{(c \cdot tsi + c \cdot t \cdot tpi + es \cdot tse + es \cdot t \cdot tpe)}{tsi + tpi + tse + tpe} \tag{12}$$

One could use Eq. 12 as a model already and fit four unknowns: '*tsi*', '*tpi*', '*tse*', '*tpe*'. In order to simplify equation fitting, we will reduce the number of unknowns from four to three. The algebraic expressions will not be shown on account of the paper scope. After those simplifications we obtain our final model and generalization from Gustafson's Law to heterogeneous machines:

$$S_G(t,c,es) = i \cdot (c - es) + \pi_i \cdot (c \cdot t - es \cdot (t-1) - 1) + es \cdot (t - \alpha \cdot (t-1)) \tag{13}$$

3. Discussion

The theoretical models here presented have not yet been numerically validated, as it was done for other analogous Amdahl's Law models in [Cesta et al. 2011] and [Cesta et al. 2012]. We expect percent error results similar to those verified with our Amdahl's Law generalization to be obtained with the Gustafson's Law variant presented in this article.

Below we present a sensitivity analysis for Eq. 13 giving the general shape of the performance response surface for a system following Gustafson's Law when its thread count and clock speed vary.

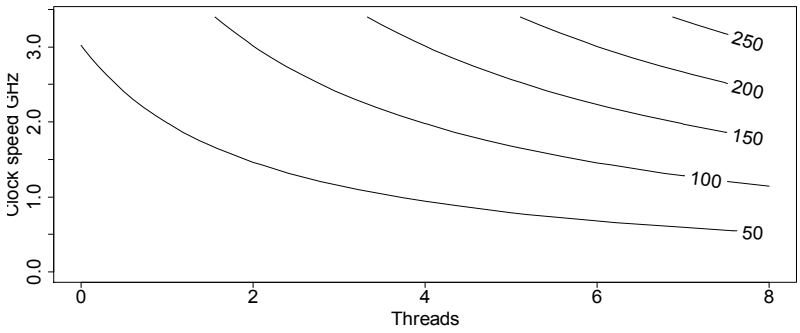

Figure 1. Sensitivity analysis for Gustafson's from threads and clock speed Eq. 13 for a configuration of: *i=0.9*; *πi=0.88*; α=0.11 and '*es*' variable set to 1.

It can be verified on the response surface from Figure 1 that doubling clock-speed is better than doubling threads for our configuration. Additionally, according to queuing theory, a system with half the service time (i.e. twice as fast) also performs better with respect to response times and throughput than a system with twice as many queue servers. This can be easily verified through simulation.

4. Conclusion and future work

In this article we have demonstrated that Gustafson's Law can be generalized from a function of processors '*p*' to a function of hardware threads '*t*' and clock speed '*c*'.

We have also performed a sensitivity analysis to better understand the response surface for Gustafson's scalability as a function of threads and clock speed.

Future work in this area should focus on: 1. validating our models empirically; 2. extending the models with more variables such as amount of processor caches.

References

Amdahl, G. M. (1967) "Validity of the Single-Processor Approach To Achieving Large Scale Computing Capabilities", in Proceedings of AFIPS, Atlantic City,NJ, AFIPS Press, pp. 483-485.

Cesta, A., Takara, A. and Moscheto, D. (2011) "Leveraging diverse regression approaches and heterogeneous machine data in the modeling of computer systems performance," in Proceedings of MSV, Las Vegas, Nevada, USA, pp. 201–207.

Cesta, A., Silva, G. and Storch, M. (2012) "Performance Prediction for Processors and External Resources," in Proceedings of CMG, Las Vegas, Nevada, USA.

Gunther, N. (2010) "Guerrilla Capacity Planning: A Tactical Approach to Planning for Highly Scalable Applications and Services". Springer.

Gustafson, J. L. (1988) "Reevaluating Amdahl's Law", in Communications of the ACM, vol. 31, no. 5, pp. 532-533.

Sun, X. and Ni, L. (1993) "Scalable problems and memory-bounded speedup", in Journal of Parallel and Distributed Computing.