Asian Journal of Information Technology 9 (2): 78-84, 2010
ISSN: 1682-3915

Analyzing Scalability of Parallel Matrix Multiplication Using Dusd

Maria A. Kartawidjaja
Faculty of Engineering, Catholic University of Atma Jaya, Jakarta 12930, Indonesia

Abstract: The demand for solving large and complex problems in a relative short time has motivated scientists to improve computational performance. The performance of a parallel system can be evaluated through the scalability analysis under different constraints, such as speedup and efficiency. Although, there is no common definition for scalability in general scalability describes the ability of a parallel system to utilize an increasing number of processors. In this study, we analyze the scalability of a parallel matrix multiplication on a cluster based environment using Dimensionless Universal Scaling Diagram (DUSD) proposed by Hockney in terms of speedup and compare it with Gustafson's scaled speedup model. We conclude that Hockney's concept provides us with a sophisticated way in analyzing the scalability of parallel system that extends Gustafson's speedup concept, nevertheless Hockney's model does not consider the overlapping between computation and communication and thus does not provide the actual speedup value. Furthermore, Hockney's concept concerns only with speedup on optimum number of processors, while in reality we utilize whatever number of processors available. Gustafson's speedup model, which is merely a function of program parameters is easy to implement and provide the actual value of speedup but it does not take into account hardware parameters that actually affect the parallel performance.

Key words: DUSD, efficiency, parallel, performance, scalability, speedup

INTRODUCTION

The demand for solving large and complex problems in a relative short time has motivated scientists to improve computational performance. The performance of a parallel system can be evaluated through the scalability analysis under different constraints, such as speedup (Gustafson, 1988) and efficiency as mentioned by Alonso *et al.* (2008). Several metrics have been used to describe scalability. Wu and Li (1997) define scalability based on a fixed computation to communication ratio, whilst Jogalekar and Woodside (2000) express scalability based on the throughput of the parallel system as mentioned by Yero and Henriques (2007). Although, there is no common definition for scalability in general scalability describes the ability of a parallel system to utilize an increasing number of processors. A system that provides an improved performance proportionally with the increasing workload when more resources are available is said to be a scalable system.

Scalability analysis of a parallel system can be used to predict the performance of a parallel algorithm on a parallel architecture for a large number of processors from the known performance on that system using fewer processors. If the size of the problem is fixed this analysis can be applied to determine the optimum number of processors in order to achieve a maximum speedup. In addition, the scalability analysis can also estimate the impact of changing hardware parameters on the system performance (Kumar and Gupta, 1994). The objective of this study is to analyze the scalability of a parallel code on a parallel architecture using Dimensionless Universal Scaling Diagram (DUSD) proposed by Hockney (1996) and compare this method to the scaled speedup model of Gustafson.

MATERIALS AND METHODS

A parallel system can be viewed as multiple processing elements that communicate and cooperate to solve large problems in a more efficient fashion than using a single processing element. In essence, a parallel system is projected to offer a better performance than that provided by a sequential system.

Speedup of parallel systems: As in single processor, most performance issues in multiprocessor can be addressed by programming techniques, architectural techniques or both. Numerous techniques have been proposed to evaluate performance of a parallel system such as execution time, speedup, efficiency and price/performance. According to Wu (1999), the most general performance metric is speedup. Approximately similar idea was revealed by Karp and Flatt (1990), who stated that the useful metrics to evaluate machine performance are speedup and efficiency. Amdahl (1967), Gustafson (1988)

and Sun and Ni (1993) used speedup to quantify the performance of a parallel system, whereas Kumar and Gupta (1994) chose processor efficiency instead. In the discussion we use speedup to measure the performance of a parallel system.

Generally, speedup is defined as the execution time of the best sequential algorithm over the parallel algorithm and is expressed as:

$$Sp = \frac{T_s}{T_p} \quad (1)$$

This speedup referred to as absolute speedup. If T_s denotes the execution time of the parallel code on a single processor, the resulting speedup is called relative speedup (Sahni and Thanvantri, 2003). In the discussion, we use relative speedup as the performance metric because the aim is to evaluate the performance gain by using multiprocessors.

Gustafson's speedup model: It is often the case that one would like to solve a larger problem or to gain a better accuracy when more processors are available instead of reducing the execution time. Based on this thought Gustafson introduced a speedup model that scale up the workload with the increasing number of processors in such a way as to preserve the execution time. According to Gustafson (1988), the parallel fraction of the program (1-α) is problem dependent. Therefore, the normalized execution time on a single processor is expressed as (Gustafson, 1988):

$$T_1 = \alpha + (1-\alpha)p \quad (2)$$

and on p processors:

$$T_p = \alpha + \frac{(1-\alpha)p}{p} = 1 \quad (3)$$

Accordingly, the speedup is;

$$S_p = p - \alpha(p-1) \quad (4)$$

In this model, the scalability analysis can be used to determine how far the problem size can be scaled up with the increasing number of processors in order to preserve the execution time. In this context we assume that infinite memory is available.

Hockney's speedup model: A parallel system is defined as a combination of a parallel algorithm and parallel architecture on which the algorithm is implemented. Therefore, performance of the parallel system should consider the parallel architecture as well as the algorithm.

This is the underlying concept of DUSD model. The execution time in DUSD model is based on three hardware and three software parameters, named as 3-parameter timing model (Hockney, 1996).

The hardware parameters describe computation rate r_∞^s for floating-point operations, the asymptotic communication bandwidth r_∞^c and message latency t_0^c, respectively. The corresponding program parameters are $S^S(N;p)$, $S^c(N;p)$ and $q^c(N;p)$ that particularly denote the number of floating point operations, the number of words being communicated and the number of communication start-ups.

The three aforementioned hardware parameters depend also on the program as mentioned by Wolton but for simplicity we assume that those parameters depend entirely on the computers being used. The execution time of a parallel program on N problem size using p processors is Hockney (1996):

$$T(N;p) = \frac{s^s(N;p)}{r_\infty^s} + \frac{s^c(N;p)}{r_\infty^c} + t_0^c q^c(N;p) \quad (5)$$

Assuming that the number of operations and communications can be factorized as $s^{s,c}(N;p) = s_N^{s,c}(N)\, s_p^{s,c}(p)$ and the number of communication as $q^c(N;p) = q_N^c(N)\, q_p^c(p)$, we substitute these factorizations into Eq. 5 and divide by $t_0^c q_N^c(N)$. The resulting equation is termed as dimensionless execution time expressed as:

$$T'(N;p) = \delta_3 s_p^s(p) + \delta_2 s_p^c(p) + q_p^c(p) \quad (6)$$

with δ_2 represents the dimensionless message length and δ_3 as the dimensionless work, expressed respectively as:

$$\delta_2(N;t_0^c,r_\infty^c) = \frac{s_N^c(N)}{q_N^c(N)t_0^c r_\infty^c}$$
$$\delta_3(N;t_0^c,r_\infty^s) = \frac{s_N^s(N)}{q_N^c(N)t_0^c r_\infty^s} \quad (7)$$

Hockney defined a parameter that represents the ratio of δ_3 and δ_2 as dimensionless computational intensity δ_1 that is formulated as:

$$\delta_1(N;t_0^c,r_\infty^c) = \frac{s_N^c(N)}{s_N^c(N}\left(\frac{r_\infty^c}{r_\infty^s}\right) \quad (8)$$

Substituting δ_1 into Eq. 6 yields the following expression:

$$T'(N;p) = \delta_3 s_p^s(p) + \frac{\delta_3}{\delta_1} s_p^c(p) + q_p^c(p) \quad (9)$$

The optimum number of processors $\tilde{p}$ in order to achieve a minimum execution time is computed by differentiating the dimensionless execution time in Eq. 9 with regards to the optimum number of processors. Speedup of the parallel program is quantified by comparing execution time on a single processor to execution time on optimum number of processors. Hence, in terms of dimensionless time the speedup is defined as:

$$S_p(N;p) = \frac{T'(N;1)}{T'(N;p)} \tag{10}$$

with $T'(N;1)$ and $T'(N;\tilde{p})$ denote the execution time on one and on $\tilde{p}$ processors, respectively written as:

$$\begin{aligned} T'(N;1) &= \delta_3 s_p^s(1) \\ T'(N;\tilde{p}) &= \delta_3 s_p^s(\tilde{p}) + \frac{\delta_3}{\delta_1} s_p^c(\tilde{p}) + q_p^c(\tilde{p}) \end{aligned} \tag{11}$$

Substituting Eq. 11 into Eq. 10 results in the following speedup formula for optimum number of processors in terms of δ_1 and δ_3 as follows (Hockney, 1996):

$$\tilde{S}_p(\delta_1,\delta_3) = \frac{\left[\frac{S_p^s(1)}{S_p^s(\tilde{p})}\right]}{\left[1 + \frac{1}{\delta_1}\frac{S_p^c(\tilde{p})}{S_p^s(\tilde{p})} + \frac{1}{\delta_3}\frac{q_p^c(\tilde{p})}{S_p^s(\tilde{p})}\right]} \tag{12}$$

The numerator of Eq. 12 represents a perfect speedup and the denominator represents the associated overhead that degrades the performance for values >1s. Using Eq. 12 enables us to draw lines of constant speedup on the optimum number of processors in the δ_1-δ_3 plane. The resulting graph is called Dimensionless Universal Scaling Diagram (DUSD).

DUSD model on parallel matrix multiplication: One of the common operations in numerical computing that requires intensive computational power is matrix multiplication, an operation that is classified as Level-3 Basic Linear Algebra Subprograms (BLAS). This operation has a time complexity of $O(n^3)$, where n denotes the dimension of the matrices assuming the matrices have a square structure. Such time-consuming process is a proper candidate for parallelization and therefore we use it as the test problem.

Suppose that the product of two square matrices A and B is defined as; C = A x B. We adopt a simple parallelization strategy where matrix A is decomposed into p blocks of approximately equal sizes and these blocks are distributed among processors. The matrix B is sent to every processor as a whole. The execution of the parallel matrix multiplication in three-parameter timing model is:

$$T(N;p) = \frac{1}{r_\infty^s}\frac{n^2(2n-1)}{p} + \frac{1}{r_\infty^c}3n^2(p-1) + 3t_0^c(p-1) \tag{13}$$

The program parameters are:

$$s^s(N;p) = \frac{n^2(2n-1)}{p} \approx \frac{2n^3}{p} \tag{14a}$$

$$s^c(N;p) = 3n^2(p-1) \tag{14b}$$

$$q^c(N;p) = 3(p-1) \tag{14c}$$

and the related factorizations are:

$$s^s(N) \approx 2n^3, \qquad s^s(p) = \frac{1}{p} \tag{15a}$$

$$s^c(N) = 3n^2, \qquad s^c(p) = p-1 \tag{15b}$$

$$q^c(N) = 3, \qquad q^c(p) = p-1 \tag{15c}$$

Hence the resulting dimensionless execution time is:

$$T'(N;p) = \delta_3\frac{1}{p} + \frac{\delta_3}{\delta_1}(p-1) + p - 1 \tag{16}$$

with the value of δ_1 and δ_3 expressed as:

$$\delta_1 = \frac{2n}{3}\frac{r_\infty^c}{r_\infty^s}, \qquad \delta_3 = \frac{2n^3}{3}\frac{1}{r_\infty^s t_0^c} \tag{17}$$

The optimum number of processors is derived by differentiating Eq. 16 with respect to p:

$$\tilde{p} = \sqrt{\frac{\delta_1\delta_3}{\delta_1 + \delta_3}} \tag{18}$$

The speedup on optimum number of processors is derived from Eq. 12 as:

$$Sp(\tilde{p}) = \frac{1}{\frac{1}{\tilde{p}} + \frac{\tilde{p}-1}{\delta_1} + \frac{\tilde{p}-1}{\delta_3}} \tag{19}$$

RESULTS AND DISCUSSION

The experiment was implemented on a group of personal computers, consisted of six processors with similar characteristics connected through a 1000 Mbps

switch. Each computer has a CPU of 1.3 GHz and 256 KB RAM and the test code was written in C language. The parallel code of matrix multiplication was executed on square matrices of dimension ranging from 300-800. The workload in this research used data parallelism, where all processors execute the same code on different set of data. The execution time on various numbers of processors is shown in Table 1.

Scalability analysis on parallel matrix multiplication: Using an analysis package we performed the fitting of the mathematical model in Eq. 13 to the execution time as a function of problem size n and number of processors p. This fitting determined the values of computation rate, asymptotic communication bandwidth and message latency as $r_\infty^s = 71{,}661985$ Mflops, $r_\infty^c = 14{,}243797$ Mwords sec^{-1}, $t_0^c = 0.028013$ sec, respectively. Substituting these hardware parameters into Eq. 17 gives the values of dimensionless parameters δ_1 and δ_3 that lie in the range of $39.75 \le \delta_1 \le 106.01$ and $8.97 \le \delta \le 170.03$, respectively. Afterwards, the optimum numbers of processors as well as the speedup on that optimum numbers of processors can be calculated using Eq. 18 and 19 and the result is shown in Table 2. Apparently, the values of dimensionless parameters vary with the dimension of the matrix and so do the optimum numbers of processors and the related speedup.

The next step is drawing the contour plot as shown in Fig. 1 using the hardware parameters shown in Table 2. From Fig. 1, one can immediately find the optimal number of processors $\tilde{p}$ required to reach the maximum performance for a specific problem size n. For example, a problem size of 400 requires roughly four processors to reach the maximum performance with speedup equals to 2.3. If seven processors are available, we need to increase the problem size up to 700 in favor of achieving maximum performance, denoted by a speedup of 3.85.

Furthermore from Fig. 1, we could increase the value of δ_3 and/or decrease the value of δ_1 to improve the performance of the parallel system. Since the value of δ_3 is an inverse function of computation rate t_∞^s and message latency t_0^c as shown in Eq. 17, an increase in performance can be achieved by decreasing the value of those before mentioned parameters. However, increasing value of δ_3 by decreasing the value of r_∞^s will raise the number of processors needed to achieve the maximum performance.

Decreasing the computation rate: If we reduce the value of computation rate by 50% from the previous rate such that $t_\infty^c = 35{,}830993$ MFlops, the values of δ_3 and δ_1 will both increase. The DUSD of this hypothetical machine is shown in Fig. 2.

We observe that for a problem size of 400, the speedup is increased from 2.3 to approximately 3 but in the expense of using more processors, six processors in this case. A similar phenomenon is also observed for matrix size of 700, where the speedup increases from 3.85 to approximately 5.3 but the optimal number of processors becomes ten instead of seven.

Decreasing the message latency: If we reduce the value of message latency by 50% from the previous latency

Table 1: Execution time of parallel matrix multiplication

Matrix dimension (n)	Execution time (sec) T_1	T_2	T_3	T_4	T_5	T_6
300	1.60	1.20	0.85	0.73	0.63	0.59
400	2.10	1.38	0.98	0.84	0.76	0.70
500	3.50	2.04	1.48	1.23	1.12	1.05
600	5.90	3.23	2.34	1.90	1.74	1.63
700	9.60	5.04	3.64	2.92	2.71	2.49
800	14.18	7.33	5.28	4.21	3.92	3.58

Table 2: Optimum number of processors and corresponding speedup

Matrix dimension (n)	Dimensionless parameters δ_1	δ_3	Optimum number of processors ($\tilde{p}$)	Speedup on $\tilde{p}$ ($S\tilde{p}$)
300	39.75	8.97	3	1.66
400	53.00	21.25	4	2.23
500	66.25	41.51	5	2.80
600	79.51	71.73	6	3.34
700	92.76	113.91	7	3.84
800	106.01	170.03	8	4.31

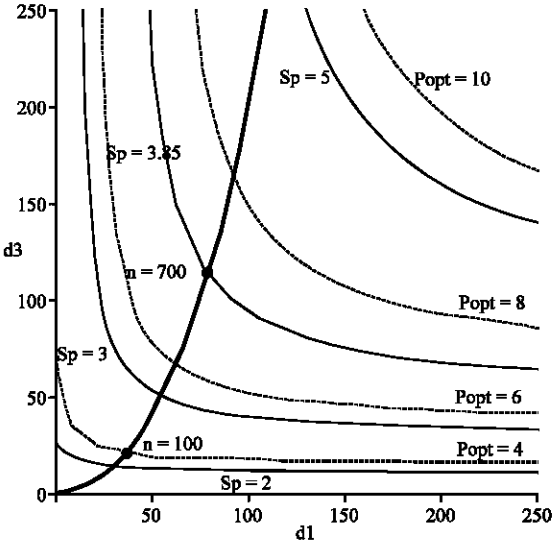

Fig. 1: DUSD for the parallel matrix multiplication

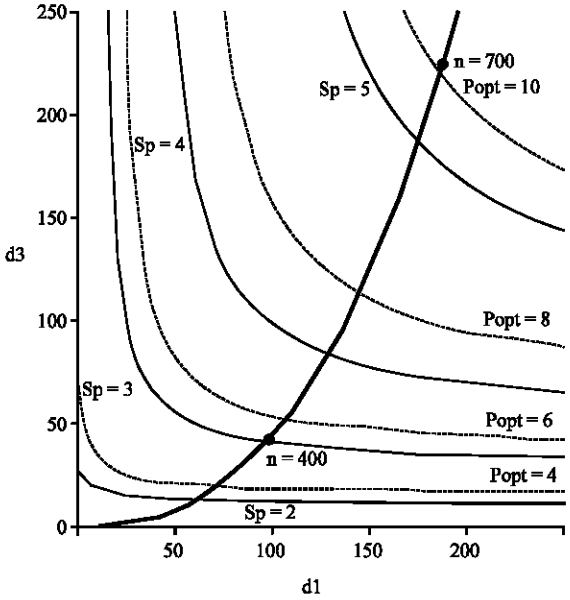

Fig. 2: DUSD on a hypothetical machine with computation rate t_∞^c = 35,830993 MFlops

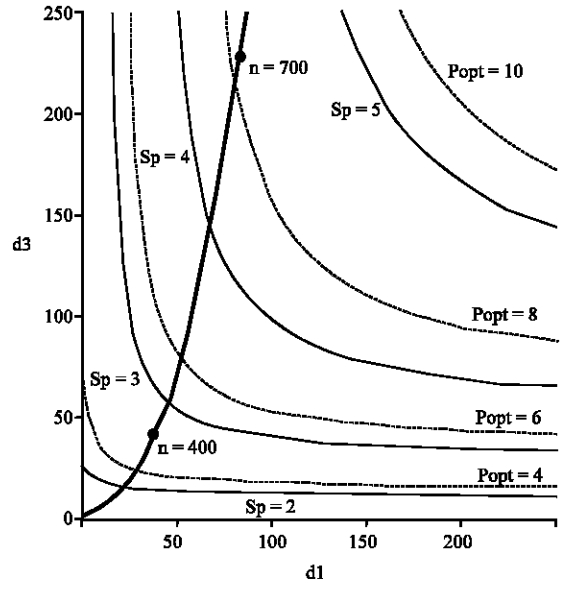

Fig. 3: DUSD on hypothetical machine with message latency t_0^c = 0.0140065 sec

such that t_0^c = 0.0140065 sec, the value of δ_3 will increase but the value of δ_1 will not be affected. The DUSD of this particular machine is shown in Fig. 3.

We observe from Fig. 3 that the speedup for a problem size of 400 becomes approximately 2.7 with the optimal number of processors equal to five. For a problem size equal to 700, the achieved speedup is approximately 4.35 on eight processors.

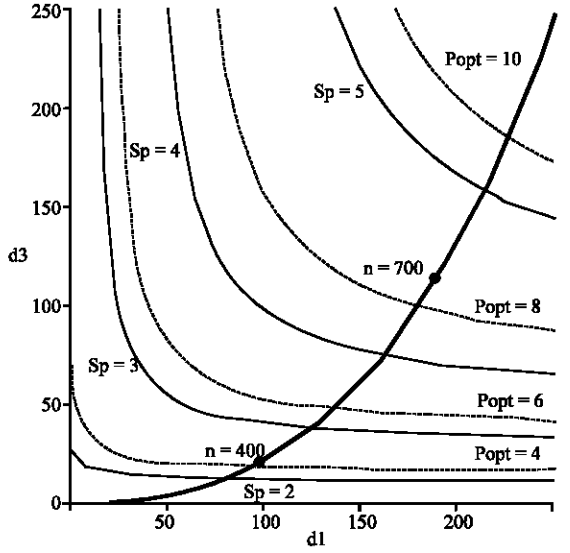

Fig. 4: DUSD on a hypothetical machine with message transfer rate t_∞^c = 28,487594 Mwords sec^{-1}

Increasing the asymptotic communication bandwidth: As previously mentioned, the performance of the system can be improved by increasing the value of δ_1. From Eq. 18, it is noted that the value of δ_1 can be increased by decreasing the value of computation rate as in previous example or by increasing the value of asymptotic communication bandwidth. Suppose the value of the latter case is increased two-fold as such that r_∞^c = 28,487594 Mwords sec^{-1}, the resulting DUSD of this machine is shown in Fig. 4. Using this particular machine, a problem size of 400 requires four processors to achieve a speedup of approximately 2.4 and problem size 700 requires eight processors to achieve a speedup of 4.5.

To acquire an insight of the impact of changing hardware parameters to the performance of the parallel system, we provide the speedup and the required number of processors to achieve that speedup in Table 3 for problem of size n = 400 and n = 700 that has previously discussed. Downward arrow in Table 3 denotes halving the associated hardware values, while upward arrow denotes doubling the corresponding hardware value.

We may notice immediately From Table 3 that reducing message latency and increasing the asymptotic communication bandwidth will result in a better performance than the performance of the working platform, however the best performance will be achieved by reducing the computation rate but in the expense of using more processors. By considering the hardware parameters, DUSD method provides us with a way to analyze what will happen to the scalability of the parallel

Table 3: Impact of changing hardware parameters on speedup and optimum number of processors

	n = 400		n = 700	
Parameter change	$S\tilde{p}$	$\tilde{p}$	$S\tilde{p}$	$\tilde{p}$
No change	2.3	4	3.85	7
$r_\infty^s \downarrow$	3.0	6	5.30	10
$t_0^c \downarrow$	2.7	5	4.35	8
$r_\infty^c \uparrow$	2.4	4	4.50	8

Table 4: Speedup of parallel matrix multiplication based on Gustafson's model

	Speedup				
Matrix dimension (n)	Sp_2	Sp_3	Sp_4	Sp_5	Sp_6
300	1.33	1.88	2.19	2.54	2.71
400	1.52	2.14	2.50	2.76	3.00
500	1.72	2.36	2.85	3.13	3.33
600	1.83	2.52	3.11	3.39	3.62
700	1.90	2.64	3.29	3.54	3.86
800	1.93	2.69	3.37	3.62	3.96

matrix multiplication if changes are made to the parallel hardware parameters. Gustafson's speedup model is easy to implement compared to Hockney's model. The speedup is directly computed from the execution time in Table 1 and the result is shown in Table 4. From Table 1, we observe that a problem size of 300 requires 1.2 sec to run on two processors. If more processors are available, four processors for instance, we could increase the problem size up to 500 and the execution time will comparatively the same, 1.23 sec to be exact. The speedup is 1.33 in the former case and 2.85 in the latter case.

Therefore, the speedup is scaling up from 1.2-1.23 if we scale up the workload from 300-500 when four instead two processors are available. Gustafson's model provide us an insight of how much can the workload be increased if more processors are available in order to preserve the execution time and hence increase the performance of the system.

If we evaluate the speedup using Hockney's model in Table 2 and the speedup using Gustafson's model in Table 4, we will notice that the speedup in Table 2 has lower values compared to that in Table 4 for any problem size on similar number of processors. For example, a problem size of 400 on four processors provides a speedup of 2.23 in Table 2 and a speedup of 2.50 in Table 4. This is due to the fact that Hockney's execution model in Eq. 5 does not take into account the overlapping between computation and communication, whilst this overlapping does occur in parallel system. The overlapping of computation and communication is central to obtaining high performance in parallel computing. From the previous discussion we may conclude that DUSD technique of Hockney is a powerful concept because a full scalability analysis is possible with only two dimensionless parameters, δ_1 and δ_3 as in the case of parallel matrix multiplication. However, Hockney's execution model does not consider the overlapping of computation and communication and thus the achieved speedup shows a lower value than the speedup computed directly from actual measurement. Furthermore, the speedup in DUSD concerns only with speedup on optimum number of processors, while in reality we utilize whatever number of processors available. In contrary, Gustafson's speedup model does not take into account hardware parameters that actually affect the parallel performance; nevertheless this method which is merely a function of the number of processors as presented in Eq. 1 is a straightforward method that capture the actual value of speedup.

CONCLUSION

From the test results and the analysis, we can draw the following conclusions: Hockney's DUSD concept provides us with a more sophisticated way in analyzing the scalability that extends Gustafson's speedup concept, unfortunately Hockney's execution model is based on the assumption that no overlapping between computation and communication and thus does not reveal the actual speedup value. In addition, Hockney's speedup model account only on optimum number of processors, while generally we utilize any number of processors.

Gustafson's speedup model, which is merely a function of the number of processors is a straightforward method that provides the actual speedup value; nevertheless this model does not consider any hardware parameters that actually affect system performance.

REFERENCES

Alonso, P., R. Cortina, I. Diaz and J. Ranilla, 2008. Scalability of neville elimination using checkerboard partitioning. Int. J. Comput. Math., 85: 309-317.

Amdahl, G.M., 1967. Validity of single-processor approach to achieving large-scale computing capability. Proceedings of the AFIPS, Spring Joint Computer Conference, April 18-20 Atlantic City, New Jersey, pp: 483-485.

Gustafson, J.L., 1988. Reevaluating Amdahl's Law. Commu. ACM, 31: 532-533.

Hockney, R.W., 1996. The Science of Computer Benchmarking. Society for Industrial and Applied Mathematics, USA., ISBN-10: 0898713633.

Jogalekar, P. and M. Woodside, 2000. Evaluating the scalability of distributed systems. IEEE Trans. Parallel Distributed Syst., 11: 589-603.

Karp, A.H. and H.P. Flatt, 1990. Measuring parallel processor performance. Commun. ACM, 33: 539-543.

Kumar, V. and A. Gupta, 1994. Analyzing scalability of parallel algorithms and architectures. J. Parallel Distributed Comput., 22: 379-391.

Sahni, S. and V. Thanvantri, 2003. Performance metrics: Keeping the focus in runtime. IEEE Parallel Distributed Technol. Syst. Appl., 4: 43-56.

Sun, X.H. and L.M. Ni, 1993. Scalable problems and memory bounded speedup. J. Parallel. Distributed Comput., 19: 27-37.

Wu, X. and W. Li, 1997. Performance models for scalable cluster computing. J. Syst. Architecture: The EUROMICRO J., 44: 189-205.

Wu, X., 1999. Performance Evaluation, Prediction and Visualization of Parallel Systems. Kluwer Academic Publishers, The Netherlands, ISBN: 0-7923-8462-8.

Yero, E.J.H. and M.A.A. Henriques, 2007. Speedup and scalability analysis of master-slave applications on large heterogeneous clusters. J. Parallel Distributed Comput., 67: 1155-1167.