

Evaluating the parallel performance of a heterogeneous system

E. Post

poste@lincoln.ac.nz

Applied Computing, Mathematics and Statistics group,
Applied Management and Computing Division, Box 84, Lincoln University,
Lincoln, Canterbury, New Zealand

H. A. Goosen

formerly of

Department of Computer Science
University of Cape Town
Cape Town, South Africa

Abstract

Heterogeneous parallel systems are becoming increasingly more common, especially with the increasing use of cluster computers, such as Beowulf systems, and networks of workstations for parallel computing. Measuring and evaluating the performance of such parallel systems is not straightforward. In particular, conventional techniques such as computing speedup and efficiency are not appropriate for evaluating the performance of a heterogeneous system, and even have their limitations for homogeneous systems. This paper looks at alternative ways of measuring and evaluating the parallel performance of a heterogeneous system, such as linear speed, and extends this with the concept of linear efficiency.

Keywords

Heterogeneous, parallel, performance, Beowulf, linear speed, speedup, efficiency, linear efficiency.

Introduction

One of the primary purposes of parallelizing an application is to reduce significantly the overall elapsed time to obtain a result. Also, one would expect to reduce the total elapsed time further if more processors were used. However, this is not necessarily the case, as the addition of further processors also causes further overheads, which may impact negatively on the performance of the application. Therefore, to ensure that one is achieving the best performance possible, or to predict probable performance if further processors are added, or to identify any performance loss, it is essential to be able to measure and evaluate the performance of a parallel application.

When most parallel computers were shared-memory multiprocessors, where all the processors were homogeneous, this was relatively easy, and conventional ways of evaluating parallel performance, such as speedup and efficiency, were often used, although these do have their limitations and problems.

Parallel computers are becoming increasingly widespread, and nowadays many of these parallel computers are no longer shared-memory multiprocessors, but rather follow the distributed memory model. These systems may consist of homogeneous workstations, where all the workstations have processors with exactly the same specifications and identical memory and caches. However, increasingly systems are now composed of a number of

heterogeneous workstations clustered together, where each workstation may have CPUs with different performance capabilities and differing amounts of memory and caches, and even different architectures and operating systems. This is evident in the increasing number of Beowulf-class clusters, which may consist of hundreds of processors, which are often heterogeneous. Some Beowulf clusters are now so powerful that an increasing number of the 500 most powerful supercomputers in the world are Beowulf clusters (see ref. Top 500 Supercomputer Sites), and it is important to be able to assess the performance of such machines, even if they are heterogeneous.

In addition many institutions are becoming aware of the large amount of potential power available in the networks of increasingly more powerful workstations that they already have. Many of these institutions are now taking advantage of this power, and using the idle time of these workstations as a distributed parallel computer. Again, most usually these workstations are heterogeneous, rather than homogeneous.

Then there is another type of heterogeneity. Even if all workstations are identical, with identical CPUs, memory and caches, they may not be 100% dedicated to the task for which they are being used. Any or all of these processors may be executing other applications that reduce the performance capacity of the processor that is available for a particular application. In addition the load on these processors may vary dynamically over time, which means the capacity dedicated to a particular task may vary. Performance on a system with this type of heterogeneity needs to be compared to the baseline performance of the dedicated system, which can be evaluated as discussed in this paper, and by Crawl (1994) and Foster (1994). In this paper we are mainly concerned with measuring and evaluating the baseline performance of a dedicated system, so this type of heterogeneity due to a non-dedicated system will not be discussed further in this paper. For further information on this topic refer to Zhang and Yan (1995), Du and Zhang (1997) and Xiao, Zhang and Qu (2000) who have done much work on using non-dedicated distributed systems for parallel processing, and the complexities involved in doing this efficiently.

Thus, in the case of a parallel computer composed of a number of separate heterogeneous processors, it is important to be able to assess whether or not an application is scalable and performance improves (or at least does not get worse) when using more processors. If performance does not improve as more processors are used so that the application is not scalable the cause needs to be identified, and if possible rectified. This may even mean that the parallel application needs to be redesigned, perhaps even using different algorithms.

This paper looks at the issue of measuring and evaluating parallel performance on a heterogeneous system. It describes briefly why conventional measures such as speedup and efficiency are not appropriate for a heterogeneous system, illustrates the use of linear speed for evaluating parallel performance, and proposes the concept of "linear efficiency" as an extension of linear speed. These concepts are illustrated by using the results obtained from measuring the performance of a real scientific application running on a network of heterogeneous Unix workstations.

Although these experiments were performed several years ago, on what are now old machines with much lower performance capability than modern machines, the results and techniques for measuring and evaluating performance that are discussed in this paper are very relevant and useful for evaluating the parallel performance of modern parallel computers, such as Beowulf clusters and networks of workstations.

Background

The concepts in this paper are illustrated with results obtained from performance tests of a parallel version of a cloud radiation model that was parallelized for the Department of Meteorology and Climatology at Penn State University, USA (Post 1995). This application is a Monte Carlo simulation modelling the amount of light reflected off, transmitted through, or absorbed in a strato-cumulus cloud deck.

A real scientific application was used for these tests because when measuring performance it is important to include all aspects of running an application, including communication time, system time, input and output time, and idle time, as well as processing time. If only processing time (CPU time) is measured, or measures such as MIPS or MFLOPS, or the performance of test kernels, or manufacturer-specific benchmarks, these may not give a true reflection of everything that may affect parallel performance in a real application. Thus when a system is used for a whole application, rather than just a kernel, results may not be what was predicted by the tests (Hennessy and Patterson, 1990).

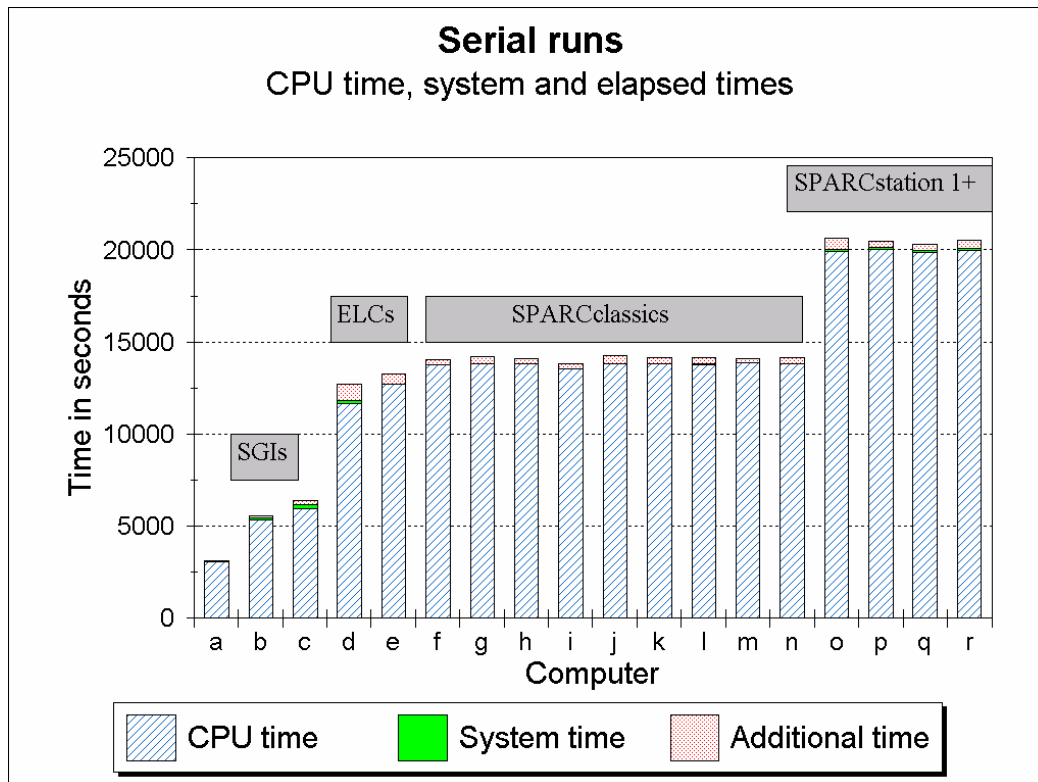


Figure 1 Serial times for processors

The parallel application was run on a network of heterogeneous Unix workstations, consisting of Silicon Graphics and Sun workstations of several different models and specifications, and also different operating systems. There were 18 workstations with widely varying performance capacity, connected by 10 Mbit Ethernet. For these tests all the workstations were dedicated to the test application only, with nothing else sharing the capacity. It was possible for the network to be used by other users while the tests were being run. However, as most of the tests were run overnight there were no other users sharing use at the network. The relative performance of these workstations is illustrated in Figure 1. This graph shows the total elapsed time taken by each machine to run the serial version of the application.

The graph illustrates the widely varying performance of the workstations used in the experiments, where the fastest processor, *a*, completes the run in about one-seventh of the time of the slowest processors, *o*, *p*, *q*, and *r*. Processors *a*, *b*, and *c* were Silicon Graphics Indigo machines with different CPU performance and different amounts of memory and caches. Machine *a* was an Indigo Extreme and the other two Indigos. All the remaining machines were Suns, with *d* and *e* being ELCs, *f* to *n* being SPARC Classics, and *o*, *p*, *q* and *r* being SPARCstation 1s. Even for those machines that had the same CPU, not all had the same amount of memory or cache, so there was considerable heterogeneity in this system.

Measuring and evaluating parallel performance

One of the most important reasons for measuring and evaluating parallel performance is to discern whether actual performance is improving if the parallel environment is changed, such as by using more processors. To do that we need a way of evaluating parallel performance and assessing whether performance is improving or not as processors are added. This section describes some techniques to do this.

In this paper we differentiate between measuring and evaluation. Measuring is recording results obtained from experiments, such as elapsed, CPU, communication and idle time. Such measurements, although of some use in understanding performance, are not enough on their own to explain the performance. Thus it is necessary to evaluate the performance by using these measurements so that they illustrate how good the performance is, and perhaps indicate where it can be improved. Common ways of evaluating parallel performance by using measurements in calculations are speedup and efficiency.

There are many aspects that must be considered when measuring and evaluating performance. It is not the purpose of this paper to cover what is already well covered elsewhere, such as in Chapter 3 of Foster (1994), and Crowl (1994), so such material will only be mentioned very briefly. The purpose of this paper is rather to explain briefly why traditional evaluation techniques such as speedup and efficiency are in particular not appropriate for heterogeneous systems, and to describe some better ways of evaluating the parallel performance of a heterogeneous system.

What to measure?

Many factors affect parallel performance, such as the performance capability of the CPU, the amount of memory available, the architecture and operating system on each component, the communication medium between processors, the algorithms used, and whether the system is dedicated or not. In particular, with heterogeneous systems the disparity between components may contribute other factors, such as further idle time while some processors wait for others. Thus it is important to measure everything that is relevant and gives information about performance, such as total elapsed time, CPU time, communication time, system time, input/output time, and idle time, for every processor. Also, knowing how much time each processor is idle is also extremely valuable when evaluating the efficiency of the system, and identifying where things can be improved. It is not useful, to measure for instance only CPU time, as this does not take into account other factors such as communication, and will give misleading results. Similarly measures such as MIPS or MFLOPS, or the performance of test kernels, or manufacturer-specific benchmarks may give misleading results as they tend to relate to CPU performance on particular architectures and do not illustrate expected performance for real applications. Hennessy and Patterson (1990) hold the position that the only consistent and reliable measure of performance is the execution time of real programs, and that all proposed alternatives to time as the metric, or to real programs as the items

measured, have eventually led to misleading claims or even mistakes in computer design. Thus many practitioners now use standard benchmarks, such as the NAS benchmarks (NAS), to measure and evaluate performance, as this benchmark suite of programs consists of real applications. Some examples of papers describing measuring and evaluation of parallel performance using the NAS benchmarks are Sukup (1994), Zhang and Yan (1995), Du and Zhang (1997). This paper uses the performance of a cloud radiation model as an illustration (Post 1995).

An important factor in the overall elapsed time of an application is the cost of memory accesses. A distributed system does have the advantage that it will frequently have much more memory available overall than is available in a serial workstation. However, there can still be problems. If the task size for a particular workstation is small enough to be totally contained within the memory of the workstation then these costs are relatively low and do not have much effect on performance. However, if the task requires more memory than is available in the workstation this will cause page-swapping which will significantly reduce the performance of the workstation (Zhang and Yan 1995). Thus these factors have to be taken into account when evaluating the performance of a heterogeneous system, as all workstations in the system will not necessarily have the same amount of memory available. One way around it is to ensure that no workstation is allocated a task that requires more memory than it has available, thus ensuring the optimum performance for that workstation. Xiao, Zhang and Qu (2000) take this into account in their load-sharing policies that consider both the CPU capability and the amount of memory available for each workstation. Differing cache sizes on different workstations can also affect the performance, but this is not so significant. (Zhang and Yan, 1995)

For the purposes of evaluating potential performance of a system it is also necessary to measure the performance of a dedicated system if at all possible, so as to establish a baseline of the performance that could be achieved. Then the performance of a system shared with other users can be compared with the overall potential performance of the same system if it was dedicated. All measurements obtained in these experiments were measured on what was essentially a dedicated system.

Measuring parallel performance

Although all these measurements as described above were recorded and used when studying the parallel performance of this application, in this paper we are concerned mainly with the measurement of total elapsed time, and using this to evaluate the overall parallel performance. Total elapsed time will include time taken for everything, whether computation, communication, system or idle time, as this is sufficient for us to evaluate performance. Other measurements of the time spent on different components, such as computation and communication, of the application can then be used to identify the factors affecting the performance, and where adjustments can be made to improve performance.

It is also important to take measurements several times, to ensure that the times are representative. In the case of the results presented in this paper, each test was run until there were at least three times in close agreement for each test, and the average time for the three results was used. In most cases only three runs for each test were needed.

Evaluating parallel performance

Once the total elapsed time has been measured it can be used to evaluate the actual performance.

Elapsed time is a good measurement, as it indicates actual performance, and is not derived from other figures. However, the conventional elapsed time graph, as in Figure 2, does not easily show whether performance is improving as larger numbers of processors are added, as the results for higher numbers of processors have little visual space. In fact, for large numbers of processors it becomes very difficult to see whether the elapsed time is continuing to decrease as processors are added, or even whether or not it may perhaps increase, indicating that the overheads of adding further processors are now outweighing the expected advantages of having more processors. (This and the remaining graphs for this paper show data points only after each group of relatively homogeneous processors has been added, since each group has difference performance capabilities, and the graphs do not show data points as each processor within the group is added. However, within each group of processors performance is homogeneous, so performance would increase linearly as more processors are added, so these graphs give a reasonably true picture, even if not all data points are shown.)

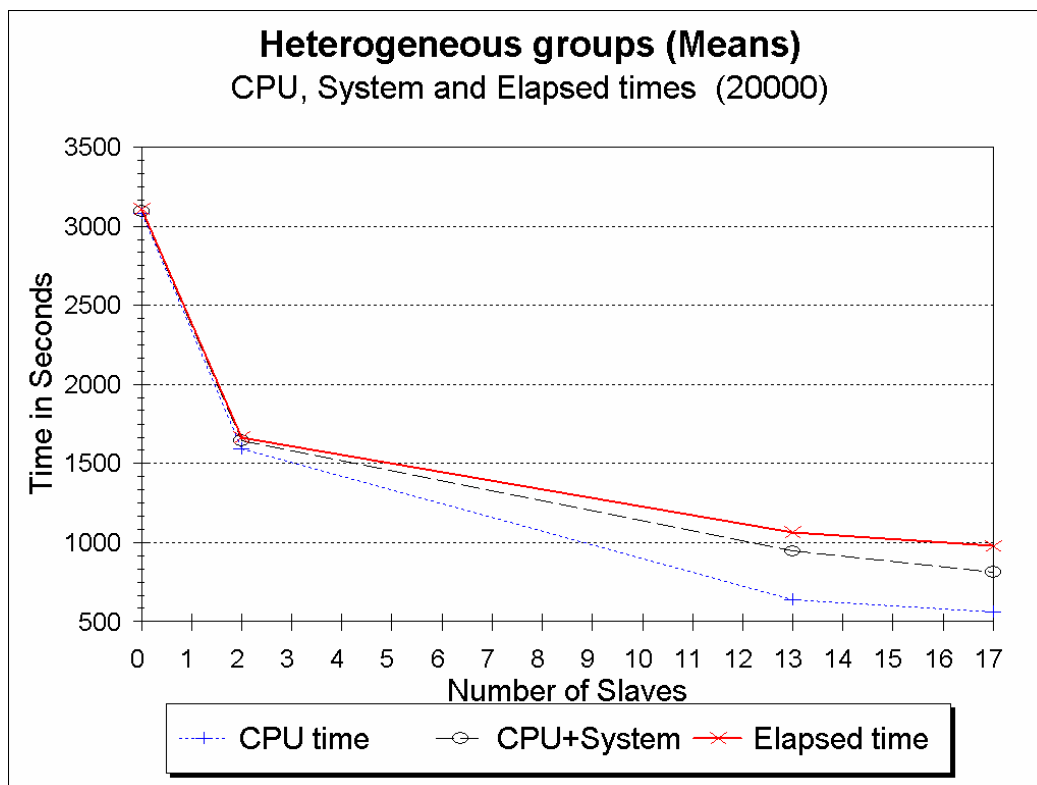


Figure 2 Elapsed time

Crowl (1994) suggests various ways of presenting measurements of parallel performance so as to give a better indication of actual performance. One of these techniques suggested by Crowl is linear speed which is illustrated later in our paper.

Speedup and Efficiency

Common techniques for evaluating parallel performance have often been speedup and efficiency.

Speedup is commonly defined as the ratio between the total elapsed time on one processor divided by the total elapsed parallel execution time on all processors. Efficiency is then calculated as the speedup divided by the number of processors and indicates how well a system is using its potential power.

However, these techniques of calculating speedup and efficiency have their drawbacks, even on homogeneous systems where speedup and efficiency do not necessarily give a true or reliable picture. For instance, Sun and Gustafson (1991) discuss some of the drawbacks of speedup and efficiency and show how speedup favours slow processors and poorly-programmed codes. They point out the necessity of having a better way of evaluating parallel performance that is machine-independent and programming-independent.

Another point to consider is what elapsed time on a single processor will be used to calculate speedup? Will it be the time of the serial program or of the parallel program running on one processor? A parallel program has some overheads, so may run slightly slower than the serial program on one processor. And surely we should be comparing parallel performance with the shortest elapsed time on a single processor, which will probably be the serial program, which may even be using a different algorithm. Foster (1994) suggests that we should use the fastest implementation running on a single processor when comparing performance with that of a parallel system, and that this will usually be the serial version.

The situation is much more difficult when evaluating the performance of heterogeneous processors. For instance, although it is easy to measure the total elapsed time for a parallel system, to what elapsed time on a single processor should this be compared to evaluate speedup? Should it be the elapsed time on the slowest processor or the fastest processor? In a widely heterogeneous system, as was used in the experiments described in this paper, each of these gives significantly different results. Although someone might want to compare the parallel elapsed time to the serial time on the slowest processor in order to show the best speedup this is not honest. Also, the real scientific user is more concerned with what happens in practice, rather than manipulating figures to give the most desired results. It seems to be generally accepted practice to compare the time taken for parallel execution with the serial time on the fastest processor in the heterogeneous system (Donaldson, Berman and Paturi, 1994), (Mechoso, Farrara and Spahr, 1994), (Zhang and Yan, 1995). This is a valid measure for evaluating speedup in a heterogeneous system. However, in itself this does not give any indication of efficiency which shows how well the system is performing.

An alternative to illustrate speedup could be to take the mean serial performance of the application on each of the processors used and use this when calculating speedup. In this way some cognisance is taken of the actual potential power added to the system as each processor is added. However, a disadvantage is that the parallel performance is compared to a different value for the new average serial performance as each processor is added, thus suggesting a more constant improvement in performance than is actually the case. The graph in Figure 3 illustrates the “speedup” calculated when using this approach. The graph in Figure 4 illustrates “efficiency” calculated from the “speedups” obtained in this way.

Both these graphs suggest good performance is achieved as fast Silicon Graphics processors are added, reasonably good performance as the SUN ELCs and SparcClassics were added, but a deterioration in performance as the slow SparcStation 1s were added. However, how can we know whether these figures are giving us a true picture of the performance?

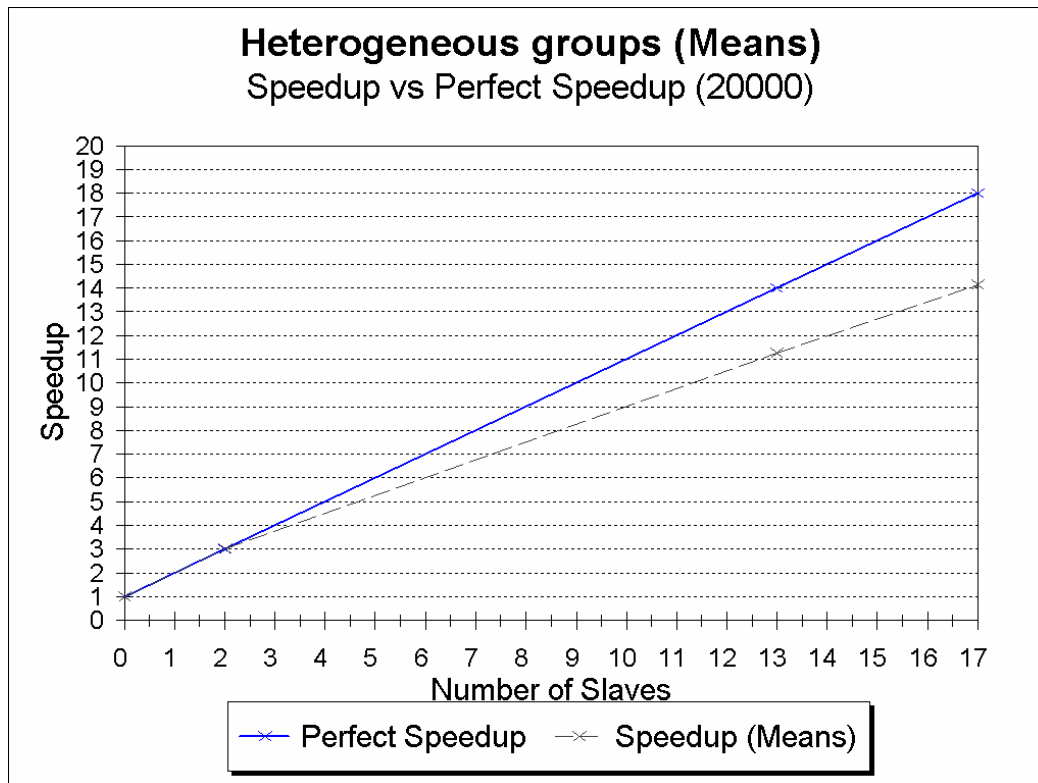


Figure 3 “Speedup” vs “Perfect Speedup”

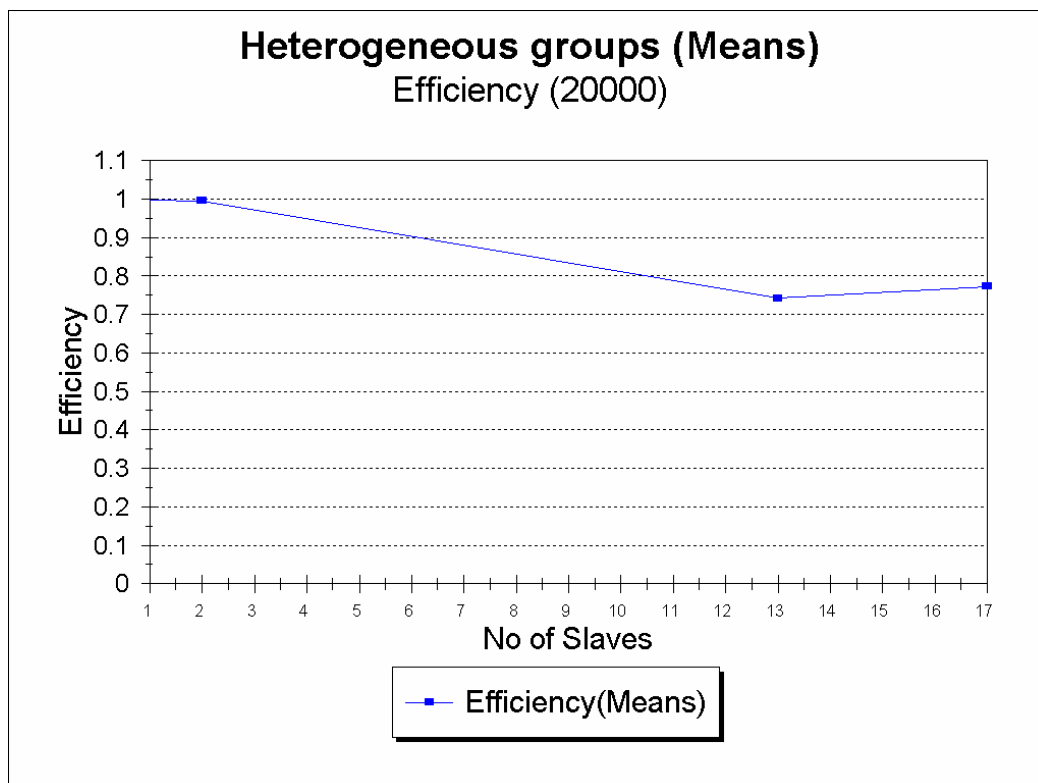


Figure 4 Efficiency

In fact, these graphs are not realistic. For instance, if one refers to Figure 1 it is clear that the serial performance of these heterogeneous processors is widely disparate. Yet the graph in Figure 3 suggests a constant improvement in performance, as though the processors were

homogeneous, which is clearly wrong, so we need to look for better ways of determining whether and how performance is improving.

Power Weight

Zhang and Yan (1995) calculate the speedup of a heterogeneous system as the ratio between the parallel performance and the performance on the fastest workstation, and propose an alternative method of evaluating the efficiency of a heterogeneous system by using the concept of power weight.

The power weight of each machine in the system is determined by comparing its performance running a real application with the performance for the same application of the fastest machine in the heterogeneous system. Thus all machines in the system will have a power weight ranging between 0 and 1. The total potential power available in the system can be calculated by summing the power weights of all machines in the system.

Zhang and Yan (1995) then use the total power weights available in the system to calculate the heterogeneous efficiency of the system as being the ratio of the total effective computing time to the total available cycle time in the system.

These are valid ways of calculating heterogeneous speedup and efficiency. A disadvantage of this method though is that the power weight of each workstation is calculated by comparing its performance with that of the performance of the fastest machine. If the fastest machine in the system is changed, either by removing it from the system, or adding another faster workstation to the system, then the power weights of all machines have to be recalculated as compared to the machine which is now the fastest one in the system. Zhang and Yan (1995) also discuss some of the difficulties of calculating heterogeneous efficiency.

Linear Speed

Crowl (1994) suggests further ways of evaluating parallel performance that can be used as alternatives. One technique he suggests is the concept of linear speed. Here, rather than directly using the elapsed time in our evaluation, we can calculate the amount of work done per unit time. This is essentially the inverse of elapsed time. Then it is possible to compare the actual amount of work done in unit time by the whole system, with the potential amount of work that could be done by the system, which is determined by summing the amount of work that could be done on each processor per unit time, as determined from measuring the serial elapsed time of the application running on each processor. This is essentially showing elapsed time graphs in another way, which gives a better visual picture of performance than does a conventional elapsed time graph.

Since it has been shown that page-swapping can significantly reduce performance (Zhang and Yan 1995), it should be ensured that the task size should fit in the memory of each workstation so that page-swapping is not necessary, and the elapsed time for such a task-size measured accordingly for the calculation of linear speed, to illustrate the best possible performance of the workstation. However, if the task size should change and become too big for memory, necessitating page-swapping, then elapsed time will have to be re-measured and linear speed recalculated, showing the reduced performance of the workstation. This would tend to indicate though that the application should rather be redesigned so that the task size will fit in memory.

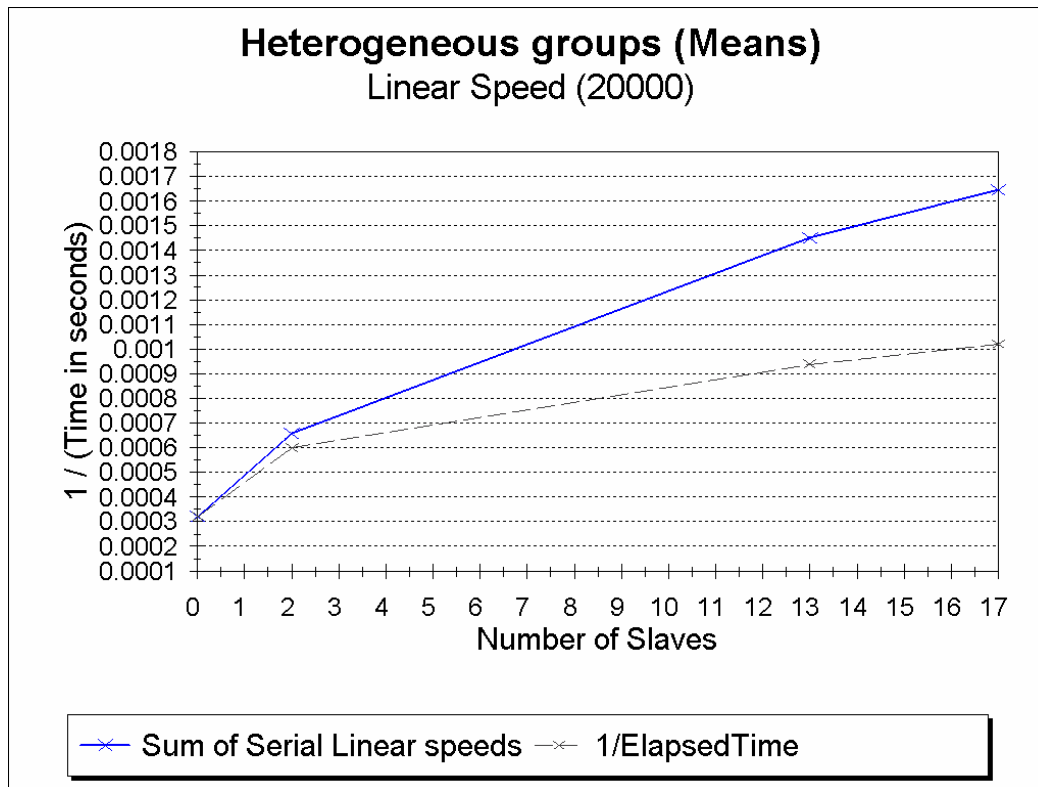


Figure 5 Linear Speed

Figure 5 shows the linear speed graph for the cloud radiation model described in this paper as compared to the potential performance denoted by a line showing the sum of the serial linear speeds for the processors used.

If this graph is compared to that in Figure 3, showing the “speedup” calculated by using the mean serial speed of all processors it can be seen that it is a similar type of graph. However, the linear speed graph gives a more realistic indication of potential performance, as it shows less potential improvement as further slower processors are added. This contrasts with the line suggesting “perfect speedup” in Figure 3, which shows a constant rate of increase in potential performance, even though the potential performance capability added by each processor is not the same for each processor.

Thus, linear speed, as proposed by Crowl (1994), gives a far more realistic way of evaluating performance of a parallel system than speedup, even ignoring all the potential complications that suggest that speedup is not a reliable indicator of performance, especially for a heterogeneous system. Linear speed is using only a measured value, the elapsed time, and is not dependent on such arbitrary factors as deciding how to compare the performance to the serial performance on which processor. This also gives an advantage in using linear speed instead of power weights as proposed by Zhang and Yan (1995), since the linear speed of any workstation is dependent on only its own performance and not on the performance of another workstation as in the case of calculating the power weights. Thus to calculate the potential capability of a system it is only necessary to sum the total serial linear speeds of the workstations involved, and it is not necessary to recalculate these if workstations in the system change as it would be necessary to recalculate the power weights if the fastest processor is changed.

Crowl also suggests other ways of presenting parallel performance, such as log-log time and log-log speed. For a discussion on these see Crowl's paper (Crowl, 1994).

Linear Efficiency

Linear speed already gives a good indication of whether or not performance is improving, especially when compared to the potential performance as indicated by the sum of serial linear speeds for all processors used. Figure 5 shows that although performance continues to improve as further processors are added, the performance does not improve as much as the slower processors are added. It would be useful to have a better indication of how efficient the system is, and whether it is being used to its full potential.

We propose an extension of linear speed as proposed by Crowl. We call this "linear efficiency" (Post, 1995)

Linear efficiency is calculated by dividing the potential linear speed of a system (sum of the serial linear speeds of processors used) by the actual linear speed achieved by the system for a parallel execution. This will usually give a value between 0 and 1, except in the cases equivalent to superlinear speedup, which would give a value above 1. (Such a situation could and does arise in cases where a larger system can actually give better performance than a small system, such as perhaps because more overall memory is available, thus leading to less swapping, as is necessary on smaller systems.)

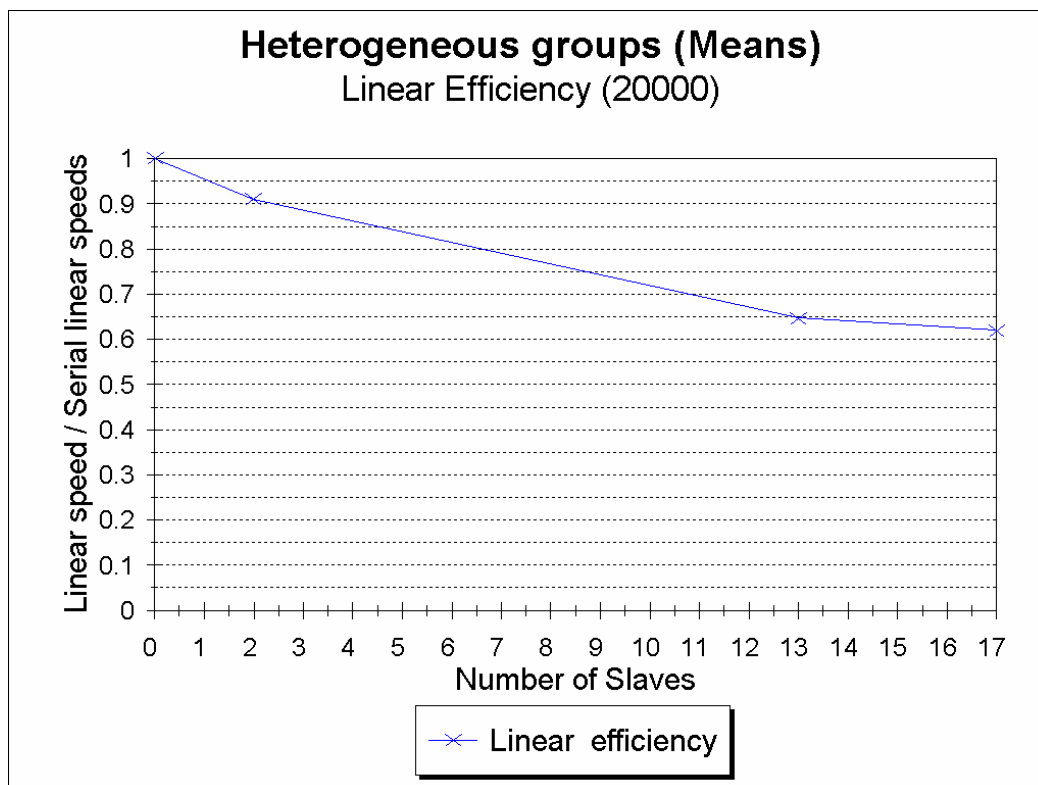


Figure 6 Linear Efficiency

The graph in Figure 6 shows the linear efficiency calculated for the experiments described in this paper. This graph essentially shows the same information as in Figure 5, but presented in a different way, so as to indicate the actual efficiency of the system.

This graph clearly shows that efficiency of the system is deteriorating as more (slower) processors are added. This suggests efficiency of between 65% and 60% as 13 or more processors are used. Since these figures are not dependent on artificially derived figures, such as speedup, which are dependent on the choice of which serial time to use, they seem to indicate more correctly that efficiency is actually less than 75% efficiency for 13 processors or more that was suggested in the graph of Figure 4.

Given this information, other measurements of performance, such as CPU, communication and idle time, could be used to determine what factors are affecting performance, and whether performance could be improved, and also whether or not it is worth adding significantly slower processors to a system composed primarily of faster processors.

Since linear efficiency uses linear speeds which are calculated for each workstation independently of the performance results for other workstations, this makes this a simpler alternative to using power weights in determining heterogeneous efficiency as proposed by Zhang and Yan (1995).

Conclusions

It is not easy to measure and evaluate parallel performance on a heterogeneous system. However, it is important to be able to do so, and to do so appropriately so as to obtain sensible results, to ensure that an application is performing as well as possible, and also that if additional processors are used, then the application is scalable and performance will continue to improve as processors are added.

This paper suggests that it is important to measure elapsed time, and not for instance just CPU time, so as to include a measure of overall performance. It has also shown that traditional measures used to evaluate parallel performance, such as speedup and efficiency, are not appropriate for evaluating parallel performance on a heterogeneous system, and has illustrated the use of linear speed as a better alternative. And finally this paper has shown how linear efficiency may be used to evaluate parallel performance of a heterogeneous system, and help assess how efficient the system is.

Also, because linear speed, and thus linear efficiency, are dependent only on the performance of the each individual workstation and not on comparisons between workstations, this makes it possible for each workstation to frequently recalculate its own linear speed without reference to any other workstations. These values will then show if the performance of the workstation is changing over time, such as in a non-dedicated system, and could then be used in a dynamic load-balancing policy.

These techniques are thus suitable for evaluating the parallel performance of modern distributed memory multiprocessors, whether clusters of computers, or networks of workstations, and even whether homogeneous or heterogeneous.

Acknowledgements

I would like to thank Dr. Tom Ackerman of Penn State University, USA, and his assistants Steve Nagle and Eugene Clothiaux, for providing the original serial application, and for their willing assistance in clarifying the meteorological aspects of the model. I also very much appreciate the constructive guidance and insight of my supervisor, Associate Professor Henk Goosen, then of the University of Cape Town, who helped me develop the concept of linear efficiency. Also I acknowledge the financial support I obtained from the Foundation for

Research and Development in South Africa which assisted in supporting my studies. I also appreciate the helpful comments of the anonymous reviewers of this paper.

References

- Top 500 Supercomputer Sites*, <http://www.top500.org/>
- CROWL, L.A. (1994): *How to Measure, Present, and Compare Parallel Performance*. IEEE Parallel and Distributed Technology, 2(1) pp9-25, Spring 1994.
- DONALDSON, V., BERMAN, F., and PATURI, R. (1994): *Program Speedup in a Heterogeneous Computing Network*. Journal of Parallel and Distributed Computing 21:pp316-322.
- DU, X. AND ZHANG, X. (1997) *Coordinating parallel processes on networks of workstations*. Journal of Parallel and Distributed Computing, Vol. 46(2) pp125-135
- FOSTER, I. (1994): *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley.
- HENNESSY, J.L. and PATTERSON, D.A. (1990): *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann
- MECHOSO, C.R., FARRARA, J.D. and SPAHR, J.A. (1994): *Achieving Superlinear Speedup on a Heterogeneous, Distributed System*. IEEE Parallel and Distributed Technology, Summer 1994.
- The NAS Parallel Benchmarks*, <http://www.nas.nasa.gov/Software/NPB/>
- POST, E. (1995): *The Efficient Parallelization of a Real Software Application*, Master's thesis, Department of Computer Science, University of Cape Town, Cape Town, South Africa.
- SUKUP, F. (1994) *Efficiency Evaluation of Some Parallelization Tools on a Workstation Cluster Using the NAS Parallel Benchmarks*, Computing Center, Vienna University of Technology, Austria, Technical Report No. ACPC/TR 94-2, January 1994.
- SUN, X-H., and GUSTAFSON, J.L. (1991) *Towards a better parallel performance metric*. Parallel Computing 17, pp1093-1109
- XIAO, L., ZHANG, X., and QU, Y. (2000): *Effective load sharing on heterogeneous networks of workstations*. Proceedings of 2000 International Parallel and Distributed Processing Symposium, (IPDPS'2000), Cancun, Mexico, May 1-5, 2000, pp. 431-438.
- ZHANG, X. AND YAN, Y. (1995): *Modeling and characterizing parallel computing performance on heterogeneous NOW*. Proceedings of the Seventh IEEE Symposium on Parallel and Distributed Processing, (SPDP'95), IEEE Computer Society Press, San Antonio, Texas, October 1995, pp. 25-34.