Proceedings of the 17th IASTED International Conference
Parallel and Distributed Computing and Systems
November 14-16, 2005, Phoenix, AZ, USA

UNIFICATION OF AMDAHL'S LAW, LOGP AND OTHER PERFORMANCE MODELS FOR MESSAGE-PASSING ARCHITECTURES

Neil J. Gunther
Performance Dynamics Company
Castro Valley, CA 94552
njgunther@perfdynamics.com

ABSTRACT
We examine some well-known but disparate parametric performance models that are frequently used for the performance analysis of parallel applications running on message-passing architectures. Examples of such parametric models include: Amdahl's law, Gustafson's quasi-linearized scaleup, harmonic speedup, and LogP-type models. By invoking a paradigm shift to a more general queue-theoretic model–the Machine Repairman Model (MRM)—each of these apparently unrelated parametric models is seen to correspond to a particular choice of possible MRM parameter mappings. In this way, all of the above ad hoc parametric models are subsumed by a single unified model which, in turn, simplifies the framework for performance analysis. More significantly, the unified MRM variables offer a way to identify bottlenecks and other sources of performance degradation. Conventional parametric models cannot provide this level of performance information because it is lost within their respective parameter values. We demonstrate the advantages of MRM by applying it to the analysis of benchmark measurements on several message-passing platforms.

KEY WORDS
Amdahl's law, grid computing, LogP, MPI, performance analysis, queueing models

1 Introduction

With the advent of new interconnect technologies (e.g., infiniband, myrinet), message-passing protocols [2, 3], local cluster architectures and global GRIDs, there arises the inevitable need for performance comparisons. A variety of ad hoc performance models have cropped up in an attempt to assess each of these differing architectures. A perennial favorite is Amdahl's law [6], and it's generalizations [11] as well as the more recent LogP model [4, 14] and its extensions [1].

Because of the variety of performance models available, the question naturally arises: Which of them is best? To some extent this question can be decided on the basis of which modeling parameters can be measured. What has not been recognized before is that each of these ad hoc models represent a special case of a more general queue-theoretic model—the Machine Repairman Model (MRM). This paper presents MRM and demonstrates how it supersedes each of the above-mentioned parametric models.

There are pitfalls in using ad hoc parametric models because they can lead to anomalous performance predictions for large-scale systems [9]. Our emphasis, therefore, is on developing a physically consistent formalism that avoids possible descent into a meaningless curve fitting exercise. What is needed is a consistent understanding of the underlying *dynamics* inherent in these parametric models. By dynamics we mean an explanation that includes interaction effects between physical components of the architecture under consideration. Unfortunately, no unified dynamical interpretation of these parametric models exists in the literature. In this climate, it has even been suggested that Amdahl's *law* has no physical meaning and therefore has no "legal" standing [17]. On the contrary, we have already shown elsewhere [8] that Amdahl's law does have a definite physical interpretation in terms of a directional broadcast protocol. We take up this point further in Sect. 3 as well as identify Amdahl's law with our MRM model.

We introduce a particular queue-theoretic model as a plausible unifying framework to reveal the correct messaging dynamics. Although our interest in the messaging dynamics, we show in subsequent sections that only *steady-state*, rather than transient, solutions are required to derive the parametric performance models of interest. Steady-state implies that time-averaged queueing variables can be replaced by ensemble averages [7]. Similarly, we show that steady-state benchmark measurements, used to determine parameters for one model, can also be mapped to another model under the appropriate parameter transformations summarized in Fig. 7.

Although the performance of message-based architectures has been studied previously using analytic queueing models [See e.g., 18], as far as we are aware, there has been no focus on applying queueing models to unifying parametric bounds for parallel speedup and latency.

2 The MRM Model

A typical message-passing multicomputer architecture is depicted schematically in Fig. 1. In the context of the so-called *LogP* model, which we take up in more detail in Sect. 7, message-passing performance can be parameterized in terms of the interconnect latency (L), the message

processing overhead (o), the message generation rate (g) and the number of physical processors (P). Hence, the name.

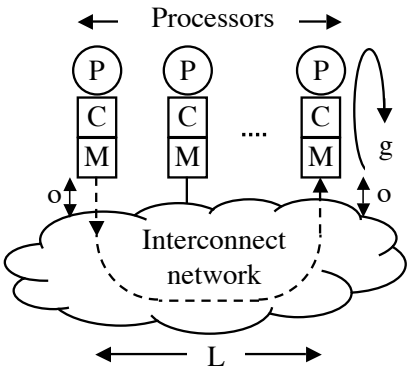


Figure 1. Generic multicomputer with interconnection network comprising P processors with their respective caches C and local memories M. The parametric model is discussed in Sect. 7.

The particular case of a queueing model with just one routing stage ($k = 1$ in our notation) in the interconnect network, is known in the operations research and computer performance literature [7, 10, 15] as the *Machine Repairman Model* or MRM. The name need not concern us here, suffice to say its historical roots lie in the performance analysis of manufacturing assembly lines. In Sect. 7 we shall generalize to interconnects with $k > 1$ stages but retain the name MRM for clarity. Multi-hop stages inevitably require the use of queueing network solvers [See e.g., 10, 15].

Table 1. MRM parameters associated with Fig. 2

P	Number of active processors
Z	Mean execution time at a processor
X	Mean system throughput
D_k	Mean interconnect latency for k routing stages
D_{max}	Bottleneck latency
$\mathcal{D}$	Minimum interconnect latency
R	Interconnect latency including waiting time

The correspondence between the two diagrams can be easily understood as follows. The processing nodes labeled PCM in Fig. 1 become the set of circles (infinite servers [15]) at the top of Fig. 2; the memory modules CM are not drawn explicitly. Similarly, the cloud labeled *interconnect network* in Fig. 1 is represented by the queueing center in the lower part of Fig. 2. An important attribute is that requests and responses circulate from the P processors to the queue and then feed back to the processors. No requests enter or leave the MRM system and the identity of which processor is sending messages and which

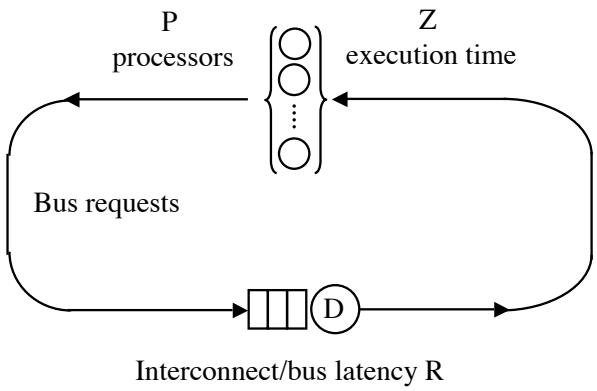


Figure 2. Representation of the multicomputer in Fig. 1 as a queue-theoretic Machine Repairman Model (MRM). Machines correspond to processors (P) each with mean execution or "up" time Z. Machines that are "down" are queued at a repairman who takes a mean service time D. The queue length determines the corresponding interconnect network latency of the message-passing architecture. Cache and memory delays are also included in the queueing time R. See Table 1.

is receiving is not enumerated since the system is calculated in steady-state. It is possible to extend MRT to distinguish between requests and response by introducing multiple classes of traffic [10], but we shall not require that level of sophistication for the subsequent discussion.

The steady-state variables that define the performance of the MRM queueing model are defined in Table 1. From these parameters it follows that

$$D_{max} = Max(D_1, D_2, \ldots, D_k) \tag{1}$$

and the minimum network latency is

$$\mathcal{D} = \sum_k D_k \tag{2}$$

The mean system throughput in Fig. 2 is defined by

$$X(P) = \frac{P}{R(P) + Z} \tag{3}$$

Both X and R are implicit functions of P, which is reflective of the feedback flow in Fig. 2, so (3) must be calculated using an analytic queueing solver [See e.g., 10, 15]. Here, however, we are mostly interested in performance bounds rather than the full performance characteristics. One such bound is the maximum achievable throughput

$$X_{max}(P) = \frac{1}{D_{max}} \tag{4}$$

which is controlled by the bottleneck latency.

Another throughput bound is due to *synchronous* queueing where all P processors suspend execution and issue a request message simultaneously. Then, the latency R

to traverse the interconnect is the product of the mean time $\mathcal{D}$ that it takes to route each message and the total number of messages P in transit i.e., $R(P) = P\,\mathcal{D}$. Substituting into (3) produces

$$Xsyn(P) = \frac{P}{P\mathcal{D} + Z} \tag{5}$$

Although this bound is known in queueing theory [15], its connection with Amdahl's law, which we derive next, seems not to have been recognized previously.

3 Fixed Size Bounds

An empirical measure of parallel performance is the *speedup* ratio

$$S(P) = \frac{T_1}{T_P} \tag{6}$$

where T_P is the elapsed time on P processors. The elapsed time T is equivalent to the execution time T_1 on a uniprocessor. The remaining time T_P can then be reduced by partitioning the application across P processors running in parallel. Symbolically,

$$T_1 = T \text{ and } T_P = \sigma T + (1 - \sigma)\frac{T}{P} \tag{7}$$

Substituting (7) into (6) and simplifying produces Amdahl's law [19]

$$S_A(P) = \frac{P}{1 + \sigma\,(P - 1)} \tag{8}$$

for a fixed size workload with the percentage of time spent in uniprocessor mode expressed in terms of a single parameter (σ), known as the *serial fraction*, having range: $0 \leq \sigma \leq 1$. In the infinite processor limit, the speedup (8) becomes

$$\lim_{P \to \infty} S_A(P) = \frac{1}{\sigma} \tag{9}$$

This limit can be interpreted as the best achievable speedup when σ^{-1} processors are running 100% busy. For example, if the serial fraction $\sigma = 0.10$ then the speedup limit corresponds to 10 saturated processors.

A more physically revealing form of Amdahl's law can be written in terms of the message-exchange diagrams shown in Fig. 3. Then, (8) can be re-expressed as the difference between ideal linear speedup and an associated inter-processor messaging overhead

$$S_A(P, k) = P - G(P, k)P(P - 1)\,. \tag{10}$$

Here, $k = 1/\sigma$, and the cost of inter-processor messaging

$$G(P, k) = \frac{1}{k + (P - 1)} \tag{11}$$

corresponds to the P-th term of a *harmonic series* (see Table 2) and is depicted by the arrows in Fig. 3. If $P = 4$

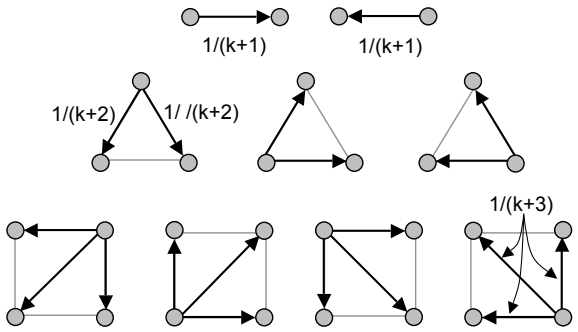


Figure 3. Pictorial representation of Amdahl's law for P = 2, 3, and 4 processors (vertices) where each arrow represents the direction of the current communication cycle with rational overhead $\sigma = 1/k$

Table 2. Amdahl cost function $G(P)$ expressed in terms of the parameter k and the serial fraction σ

P	1	2	3	4	5
$G_k(P)$	$\frac{1}{k}$	$\frac{1}{k+1}$	$\frac{1}{k+2}$	$\frac{1}{k+3}$	$\frac{1}{k+3}$
$G_\sigma(P)$	σ	$\frac{\sigma}{1+\sigma}$	$\frac{\sigma}{1+2\sigma}$	$\frac{\sigma}{1+3\sigma}$	$\frac{\sigma}{1+4\sigma}$

and the serial fraction $\sigma = 1/7$, then both (8) and (10) predict a speedup of $S_A(4) = 2.80$ where the second term in (10) corresponds to the 4-node diagrams in the last row of Fig. 3.

Equation (10) can also be interpreted as a kind of *broadcast* interaction where the requesting processor causes every other processor to halt execution and listen to the message [8]. The processors then respond cyclicly in the same way. Although the broadcast cost in Table 2 for a single processor $G(1)$ is non-zero under this interpretation, the total overhead in (10) is zero due to the $(P - 1)$ factor in the second term. The broadcast interpretation does not represent an *efficient* interaction, only a logically *correct* interaction associated with Amdahl's law. Next, we show that (8) and (10) have a related physical interpretation under the MRM model.

To see this connection, let

$$\sigma = \frac{\mathcal{D}}{\mathcal{D} + Z} \tag{12}$$

such that the range of σ values is determined by MRM variables $\mathcal{D}$ and Z

$$\sigma \to \begin{cases} 0 & \text{as } \mathcal{D} \to 0,\ Z = \text{ const.} \\ \\ 1 & \text{as } Z \to 0,\ \mathcal{D} = \text{ const.} \end{cases} \tag{13}$$

When $\sigma = 0$ the interconnect latency $\mathcal{D}$ is zero because there is maximal execution time Z with no messages exchanged between processors. Consequently, there cannot be any queueing contention in Fig. 2. Conversely, $\sigma = 1$

corresponds to zero execution time and maximal queueing latency on the communication network.

Substituting (12) into (8) produces

$$S_A(P) = \frac{P(\mathcal{D} + Z)}{P\mathcal{D} + Z} \tag{14}$$

which can be interpreted immediately in terms of the synchronous throughput (5). Amdahl speedup (14) is the ratio of the synchronous throughput with P processors to the synchronous throughput on a single ($P = 1$) processor. See Appendix A for the complete derivation. Amdahl's law therefore corresponds to worst-case queueing in our MRM model. It is the speedup bound for synchronous messaging, which can also be regarded as the queueing analog of the broadcast protocol defined by (10). Once again, we are not trying to identify the most efficient messaging protocol but rather, the correct dynamics expressed by Amdahl's law.

4 Harmonic Bounds

Although Amdahl's law constitutes the worst-case queueing bound in MRM parlance, even lower bounds on speedup do exist [5, 6]. For the sake of completeness, we briefly consider how they can be interpreted within the context of the MRM model.

If the workload is equally likely to make use of any subset of processors (equipartitioning), the speedup becomes

$$S_H(P) = \frac{P}{1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{P}} \tag{15}$$

Since the harmonic series in the denominator can be approximated by

$$\sum_{n=1}^{P} \frac{1}{n} \simeq \ln(P)\,, \tag{16}$$

an upper bound on the equipartitioned speedup is

$$S_H(P) \simeq \frac{P}{\ln(P)} \tag{17}$$

From the standpoint of MRM, S_H has to be viewed as a *worse than worst case* speedup bound ($S_H \ll S_A$) as evidenced so graphically in Fig. 4. Benchmark measurements that conform to (17) are likely to be a signal that serious performance tuning is required, although exceptions can arise e.g., a Divide-and-Conquer algorithm might produce this kind of speedup characteristic where the logarithm in (17) is expressed in base-2.

If, instead of the harmonic sum (16) of *all* possible processing subsets, we consider the *harmonic mean* μ_{H_P} of two extreme subsets of processors

$$\frac{1}{\mu_{H_2}} = \frac{1}{2}\left(\frac{1}{P_1} + \frac{1}{P_2}\right) \tag{18}$$

where $P_1 = \kappa$ is a CPU-saturated subset of the entire parallel set $P_2 = P$, then

$$\mu_{H2} = \frac{2\,\kappa P}{\kappa + P} \tag{19}$$

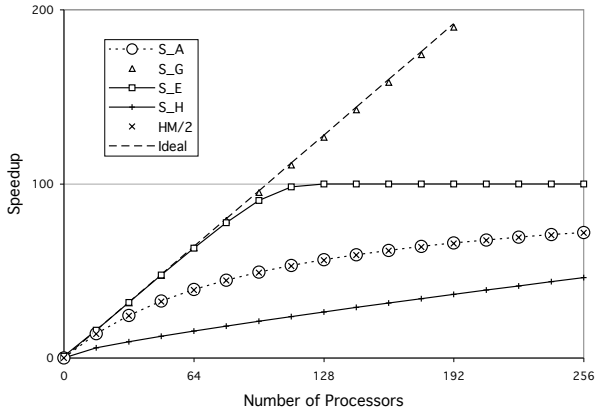


Figure 4. Speedup bounds for $\sigma = 0.01$. S_A is the synchronous speedup, the harmonic mean (20) is superimposed on the same curve, S_E is the asynchronous speedup, S_G the near-linear scaled speedup and the lowest curve S_H is the equipartitioned speedup (17)

Replacing the constant κ with the inverse of the serial fraction σ, we find that (19) is related to (8) by

$$S_A(P) \simeq \frac{1}{2}\,\mu_{H2} \tag{20}$$

In other words, the asynchronous Amdahl bound is extremely well approximated (to within 1% error in Fig. 4) by half the harmonic mean of the maximally parallel and maximally saturated processor subsets.

This connection between Amdahl's law and harmonic bounds (17) and (19) is discussed in [5, Chap. 8] and [6, Chap. 3] but otherwise seems not to be widely known. The MRM model makes it clear via the appropriate choice of parameter mappings (Fig. 7).

5 Scaled Size Bounds

It has been recognized for some time that predicting parallel speedup for real problems can differ from that predicted by Amdahl's law because the percentage of time (σ) spent in sequential sections of the program can depend on the size of the problem. Scaling up the problem size can improve speedup as follows.

If the serial fraction is assumed to vary linearly with P, the speedup (6) becomes

$$S(P) = \frac{T_1}{T_P} = \frac{\sigma + (1-\sigma)P}{\sigma + (1-\sigma)} \tag{21}$$

which simplifies to

$$S_G(P) = P + \sigma(1 - P) \tag{22}$$

By analogy with Sect. 3, (22) is sometimes referred to as the *Gustafson-Baris law* [11] which has been demonstrated for certain parallel applications [12].

The relationship to the MRM model is easily determined by substituting (12) into (22) to produce

$$S_G(P) = \frac{\mathcal{D} + PZ}{\mathcal{D} + Z} \tag{23}$$

If we invoke the change of MRM variable $Z \mapsto Z'/P$, (23) becomes

$$S_G(P) = \frac{P(\mathcal{D} + Z')}{P\mathcal{D} + Z'} \tag{24}$$

which is identical to (14). See Appendix B for the derivation. Once again, we have arrived at a well-known parametric performance model by identifying the correct set of variables in the unifying MRM model (see Fig. 7).

The MRM interpretation of Gustafson's law (22) is that rescaling offsets the impact of the synchronous queueing by inflating the mean execution time Z in direct proportion to the number of processors P. In other words, rescaling the problem size minimizes message traffic on the network.

6 Erlang Model

The near-linear speedups implied by (22) are generally extremely difficult to achieve in practice. Based on the MRM analysis of Sect. 3, a more attainable compromise would be asynchronous messaging. The asynchronous speedup can be determined from this queueing formula

$$S_E(P) = \frac{1}{\sigma}\left\{1 - E_B\left(\frac{1-\sigma}{\sigma},\, P\right)\right\} \tag{25}$$

where $E_B(A, P)$ is the Erlang B function [10]

$$E_B(A, P) = \frac{\frac{A^P}{P!}}{\sum_{k=0}^{P} \frac{A^k}{k!}} \tag{26}$$

with $A = Z/\mathcal{D}$ the *service ratio* for Fig. 2. We have used the definition of σ in (12) to rewrite the service ratio as $A = (1-\sigma)/\sigma$ in (25).

The factor $1 - E_B$ in (25) is the probability that the interconnect network is busy. It is usually more convenient to compute (26) using the following recursive algorithm:

```
erlangB = A / (1 + A);
for (k  = 2; k <= P; k++) {
    erlangB *= A / (A * erlangB + k);
}
```

which accumulates the result in the variable `erlangB`.

The idea that asynchronous speedup effects can be predicted using (25) seems to be entirely novel and is a direct consequence of realizing that $S_A(P)$ in Sect. 3 describes the synchronous MRM dynamics of Fig. 2. A comparison of S_E and S_A is provided in Fig. 4 for a serial fraction of $\sigma = 0.01$. It is noteworthy that, although S_E offers greater speedup due to asynchronous messaging, the infinite processor asymptote $S_\infty = 100$ is reached at a smaller configuration of physical processors ($P < 128$) than for S_A. The reason for this can easily be explained in terms of MRM queueing effects (Fig. 7).

From Sect. 3, S_A is associated with the lower bound on the normalized throughput due to maximal synchronous queueing in the network. Under light messaging loads ($P < 64$ in Fig. 4), the speedup is essentially linear rising because the asynchronous messaging due to each additional processor creates relatively little queueing contention in the network. However, at some point ($P \geq 128$ in Fig. 4) the network saturates (i.e., reaches 100% busy routing messages) and becomes the bottleneck.

In Sect. 2 the bottleneck was identified with D_{max} (the longest of the routing times for a k-stage interconnect) because it controls the maximum available throughput by virtue of (4). Any messages coming from faster routers upstream simply make the bottleneck queue longer, while faster downstream routers remain underutilized waiting for completions at the bottleneck. In Fig. 4, D_{max} is associated with the normalized throughput becoming bounded at $S_\infty = 100$.

7 LogP Model

LogP [4] is a latency model rather than a speedup model. Unlike the single parameter speedup models we have discussed so far, LogP has four parameters which can be measured directly [14]. Because of the correspondence between Figs. 1 and 2, however, the mapping between the original LogP parameters and the MRM variables of Sect. 2 is the most direct of all and can be summarized as follows:

$$LogP \equiv \left\{ \begin{array}{lll} L & \text{Latency} & \mapsto \mathcal{D} \text{ or } R \\ o & \text{Overhead} & \mapsto Z_{min}/2 \\ g & \text{Generate} & \mapsto Z/P \\ P & \text{Processors} & \mapsto P \end{array} \right. \tag{27}$$

Taken collectively, these parameters (which also annotate Fig. 1) give this latency model its name.

Although LogP intrinsically involves asynchronous messaging, the original formulation of the model [4] had no communication contention i.e., no queueing ($L \mapsto \mathcal{D}$) in MRM parlance. Taking as an example the measured nCUBE2 parameters reported in [4], with message size $M = 160$ bits, channel width $W = 1$ bit, mean routing hops $H = 5$, mean network latency $L = 360$ cycles and $g \equiv 2o = 6400$ cycles, the minimum round trip time is

$$RTT_{min} = L + g = 6,700 \text{ cycles} \tag{28}$$

This is equivalent to the MRM response time

$$R = \mathcal{D} + \frac{Z}{P} \tag{29}$$

with zero contention i.e., $P = 1$ where it is assumed that there can be no more than one outstanding message per processor in transit. Moreover, since we know $\mathcal{D}$ and Z from (29), we can apply (12) to estimate that the serial contention is $\sigma = 0.0274$ for the nCUBE2. Surprisingly,

even though LogP is traditionally used as a latency model we can assess its speedup characteristics using the MRM mappings in Fig. 7.

MRM can provide deeper insight into LogP contention effects but, as noted in Sect. 2, with multiple queueing stages representing a multi-hop network we need to use an analytic queueing solver [See e.g., 10] since each stage could have different service times D_k (cf. Table 1). With such tools we can predict the full latency characteristics for the nCUBE2 in Fig. 5 with $k = 5$ hops.

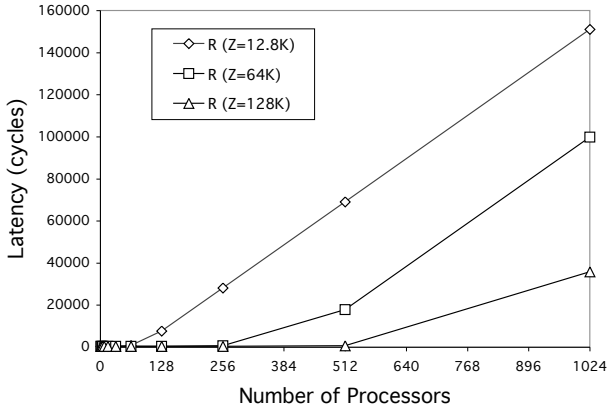


Figure 5. Latency characteristics computed using the MRM generalization of the LogP model of nCUBE2.

The lowest latency curve (bottom) corresponds to a mean execution time of $Z = 128,000$ cycles, while the highest latency curve (top) corresponds to $Z = 12,800$ cycles; the longer execution time producing fewer messages. Even for the latter case, however, the RTT grows from 360 cycles at $P = 1$ to almost 36,000 cycles under contention from $P = 1024$ processors. The ability to overlap messages could significantly reduce this contention.

In MRM, the *optimal processor configuration* is associated with the knee in the classic "hockey stick" characteristic; like those shown in Fig. 5. That knee can be predicted from

$$P_{opt} = \frac{\mathcal{D} + Z}{D_{max}} \tag{30}$$

the ratio of the minimum RTT to the bottleneck latency (see Sect. 2). Mapping the MRM variables back to the LogP model, we find (30) corresponds to

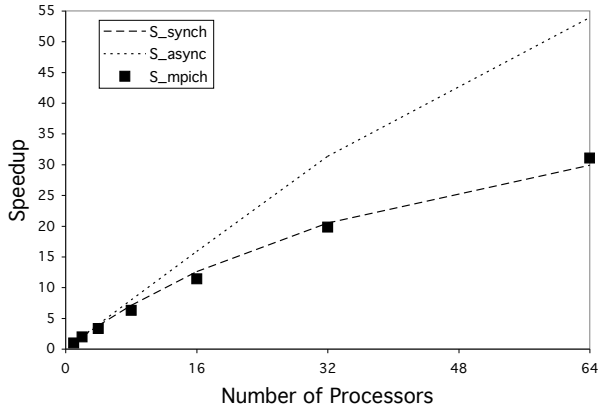
$$\frac{L + 2o}{\lceil \frac{M}{W} \rceil} \tag{31}$$

which is the ratio of the minimum RTT in (28) to the bisection bandwidth [4]. For the nCUBE2 configurations in Fig. 5, $P_{opt}(Z = 12.8K) = 81$, $P_{opt}(Z = 64K) = 401$, and $P_{opt}(Z = 128K) = 801$ processors, respectively; well below the $P = 1024$ available.

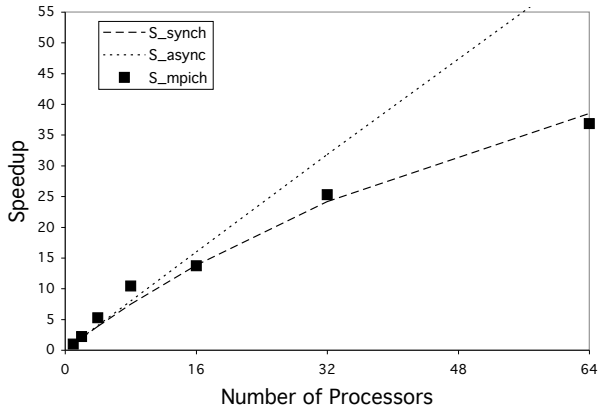
Extensions to LogP such as LogGP accounting for longer messages [1], the inclusion of memory models [3], and applications to MPI [2, 14] can also be included as extensions to MRM but we do not pursue them here.

8 Speedup Measurements

In this section, we compare the MRM model predictions with published benchmark measurements. Figure 4 shows the relationship between the various speedup models with σ chosen to be 1% producing an asymptote at $S_\infty = 100$ for clarity. Both the synchronous speedup S_A in (14) and the asynchronous speedup S_E in (25) approach this asymptote but at different rates. Asynchronous messaging provides better speedup below $P \approx 100$ but saturates more rapidly than synchronous messaging.



(a) nCUBE2 speedup measurements and predictions.



(b) Paragon speedup measurements and predictions.

Figure 6. Comparison of speedup measurements with MRM predictions for (a) the nCUBE2 and (b) the Intel Paragon.

Speedup measurements using a Jacobian solver benchmark [16] are plotted for the nCUBE2 in Fig. 6(a) and for the Intel paragon in Fig. 6(b) running the *MPICH* implementation of MPI. We used statistical regression analysis to estimate the respective σ parameters and found $\hat{\sigma} = 0.0181$ for the nCUBE2 and $\hat{\sigma} = 0.0105$ for the Intel

Paragon. The predicted speedup curves were then calculated using (8) for the synchronous case, and (25) for the asynchronous bound.

Clearly, the nCUBE2 benchmark data fall mostly in the synchronous curve. The measured speedup adheres closely to Amdahl's law which implies that there is significant contention on the interconnect network (Sect. 3). We are unable to determine the cause of that contention from the published data. Conversely, the measured speedup may be the best that can be achieved on that generation of hypercube topology running the ELLPACK benchmark workload. Those details notwithstanding, data of this type suggests that further investigation for performance tuning opportunities is likely to be worthwhile.

The Paragon benchmark data reveals a subtlety different behavior. For configurations with $P \leq 8$ processors, latencies on the mesh topology [18] are relatively small and the speedup is almost linear. At $P = 16$ processors and above, however, messaging appears to become more synchronized and, once again, the speedup falls onto the Amdahl bound.

9 Conclusion

Parametric models play an important role in analyzing the performance of message-passing architectures but, as we have tried to demonstrate in this paper, they should be grounded in a physically meaningful dynamics to avoid incorrect or inaccurate performance predictions. We have shown elsewhere [9] the dangers of constructing unphysical parametric models to assess multiprocessor scalability because they can lead to anomalous performance predictions for large-scale systems. Once properly defined and understood, however, parametric models are often more convenient to apply than complex queueing models or simulations.

That said, however, we should recognize that even a physically meaningful parametric model suffers the intrinsic limitation that what makes it convenient also makes it opaque. Benchmarks and regression analysis can be used to determine the value of its parameters. If the estimated parameter values and projected speedups are declared unsatisfactory, it is not possible to use the parametric model in reverse to determine precisely where the performance problems lie in the measured system. Such information has been lost within the modeling parameters. In that sense, the convenience of parametric models is a one-way street.

The MRM framework defined in Sect. 2 provides a unified physical interpretation for the parameters in each of the models summarized in Fig. 7. The ability to redefine a single ad hoc parameter in terms of several physical variables underlies the power of the MRM approach and it quickly narrows the search for available performance tuning opportunities in actual systems. As far as we know, this has not been achieved before. Finally, we note that MRM can provide insight into sizing buffers. Although buffering can introduce longer latencies than may be desirable for many scientific and engineering applications, it seems inevitable that buffering will play a more significant role for general-purpose applications running on newer computational architectures such as distributed clusters and GRIDs [13].

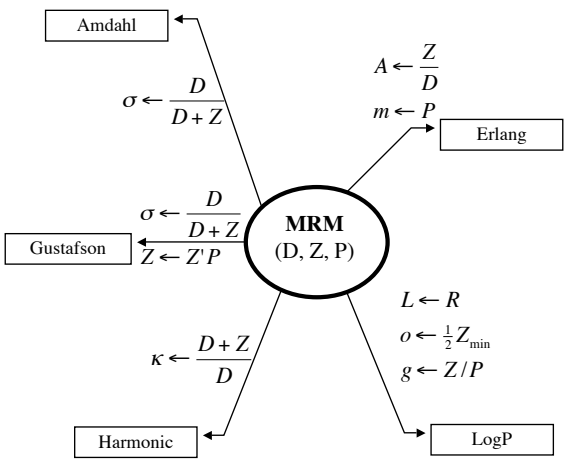


Figure 7. Mappings between MRM variables and parametric model parameters.

Acknowledgements

The author is grateful to K. Christensen, D. Lilja, C. Linn and J. Peek for commenting on earlier drafts of this paper.

Appendices

We derive the Amdahl bound $S_A(P)$ using the MRM variables defined in Section 2, and then we transform that result into the linearized form $S_G(P)$ of Section 5.

A Derivation of $S_A(P)$

Let the elapsed time on a uniprocessor be

$$T_1 = \mathcal{D} + Z \tag{A-1}$$

and on a multicomputer with P physical processors

$$T_P = \sigma T_1 + \left(\frac{1-\sigma}{P}\right) T_1 \tag{A-2}$$

From (6) the parallel speedup is

$$S_A(P) = \frac{\mathcal{D} + Z}{\sigma(\mathcal{D} + Z) + (1-\sigma)\left(\frac{\mathcal{D}+Z}{P}\right)} \tag{A-3}$$

Substituting (12) into (A-3) and collecting terms

$$\begin{aligned} &= \frac{\mathcal{D} + Z}{\mathcal{D} + \frac{\mathcal{D}+Z}{P} - \frac{\mathcal{D}}{P}} \\ &= \frac{P(\mathcal{D} + Z)}{P\mathcal{D} + Z} \end{aligned} \tag{A-4}$$

which establishes the equivalence of (14) and (A-4).

B Derivation of $S_G(P)$

Rearranging (22) to read

$$S_G(P) = \sigma + P - \sigma P \tag{B-1}$$

and substituting (12) into (B-1) we have

$$\begin{aligned} S_G(P) &= \frac{\mathcal{D}}{\mathcal{D}+Z} + P\frac{\mathcal{D}+Z}{\mathcal{D}+Z} - p\frac{\mathcal{D}}{\mathcal{D}+Z} \\ &= \frac{\mathcal{D}+PZ+P\mathcal{D}-P\mathcal{D}}{\mathcal{D}+Z} \end{aligned} \tag{B-2}$$

which, upon collecting terms, simplifies to (23). The relationship between (B-2) and (14) can be seen by rescaling the execution time $Z \mapsto Z'/P$

$$\begin{aligned} S'_G(P) &\mapsto \frac{\mathcal{D}+Z'}{\mathcal{D}+(Z'/P)} \\ &= \frac{\mathcal{D}+Z'}{\frac{1}{P}(P\mathcal{D}+Z')} \\ &= \frac{P(\mathcal{D}+Z')}{P\mathcal{D}+Z'} \end{aligned} \tag{B-3}$$

which has the same form as (A-4) and (14).

References

[1] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating long messages into the LogP model for parallel computation. *J. Parallel & Dist. Computing*, 44(1):71–79, 1997.

[2] S. Byna, W. Gropp, X.-H. Sun, and R. Thakur. Improving the performance of MPI derived datatypes by optimizing memory-access cost. SC2003 Conference Poster Session, Nov 15-21 2003.

[3] K. W. Cameron and X.-H. Sun. Quantifying locality effect in data access delay: Memory LogP. In *Proc. IEEE Intl. Parallel & Dist. Processing Symp. (IPDPS)*, Nice, France, April 2003.

[4] D. E. Culler, R. M. Karp, D. Patterson, A. Sahay, E. E. Santos, K. E. Schauser, R. Subramonian, and T. Eicken. LogP: A practical model of parallel computation. *CACM*, 39(11):79–85, Nov 1996.

[5] M. J. Flynn. *Computer Architecture: Pipelined and Parallel Processor Design*. Jones and Bartlett, 1995.

[6] E. Gelenbe. *Multiprocessor Performance*. Wiley, NY, 1989.

[7] D. Gross and C. Harris. *Fundamentals of Queueing Theory*. Wiley, 1998.

[8] N. J. Gunther. Understanding the MP effect: Multiprocessing in pictures. In *Proc. Comp. Measurement Group*, pages 957–968, San Diego, CA, Dec 1996.

[9] N. J. Gunther. A new interpretation of Amdahl's law and Geometric scalability. `xxx.lanl.gov/abs/cs.DC/0210017`, Oct 2002.

[10] N. J. Gunther. *Analyzing Computer System Performance with Perl::PDQ*. Springer-Verlag, 2005.

[11] J. L. Gustafson. Reevaluating Amdahl's law. *CACM*, 31(5):532–533, 1988.

[12] J. L. Gustafson, G. Monty, and R. Benner. Development of parallel methods for a 1024-processor hypercube. *SIAM J. Sci. Stat. Comp.*, 9(4):1–32, 1988.

[13] N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A grid-enabled implementation of the message passing interface. *J. Para. & Dist. Computing*, 2003.

[14] T. Kielmann, H. E. Bal, and K. Verstoep. Fast Measurement of LogP Parameters for Message Passing Platforms. In *4th Workshop on Runtime Sys. for Parallel Programming (RTSPP)*, number 1800 in LNCS, pages 1176–1183, Cancun, Mexico, May 2000. Springer-Verlag.

[15] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice–Hall, 1984. Available online `www.cs.washington.edu/homes/lazowska/qsp/`.

[16] S. Markus, S. B. Kim, K. Pantazapoulos, A. L. Ocken, E. N. Houstis, P. Wu, S. Weerawarana, and D. Maharry. Performance evaluation of MPI implementations using the parallel ELLPACK solvers. In *Proc. MPI Dev. Conf.*, pages 162–169. Univ. Notre Dame, 1996.

[17] F. P. Preparata. Should Amdahl's law be repealed? VI Internl. Symposium on Algorithms and Computation (ISAAC), Dec 4-6 1995. `www.cs.newcastle.edu.au/ISAAC/preparata.html`.

[18] D. A. Reed and R. M. Fujimoto. *Multicomputer Networks: Message Based Parallel Processing*. MIT Press, MA, 1987.

[19] W. Ware. The ultimate computer. *IEEE Spectrum*, 9:89–91, March 1972.