

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 its 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 the 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.

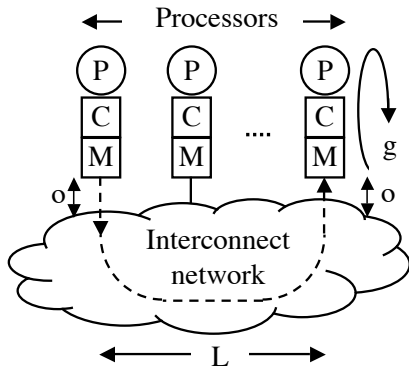


Figure 1. Generic multicomputer with interconnection network comprising P processors with their respective caches C and local memories M . The parametric model is discussed in Sect. 7.

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

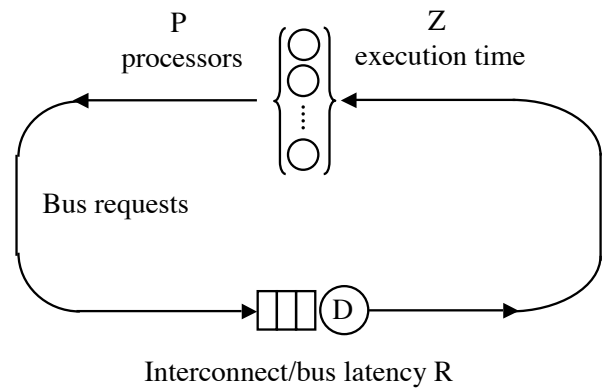


Figure 2. Representation of the multicomputer in Fig. 1 as a queue-theoretic Machine Repairman Model (MRM). Machines correspond to processors (P) each with mean execution or “up” time Z . Machines that are “down” are queued at a repairman who takes a mean service time D . The queue length determines 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

$$D_{max} = \text{Max}(D_1, D_2, \dots, D_k) \quad (1)$$

and the minimum network latency is

$$\mathcal{D} = \sum_k D_k \quad (2)$$

The mean system throughput in Fig. 2 is defined by

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

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

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

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

