LogfP - A Model for small Messages in InfiniBand

Torsten Hoefler, Torsten Mehlan, Frank Mietke, and Wolfgang Rehm

Chemnitz University of Technology
Dept. of Computer Science
Chemnitz, 09107 GERMANY
{htor, tome, mief, rehm}@cs.tu-chemnitz.de

Abstract

Accurate models of parallel computation are often crucial to optimize parallel algorithms for their running time. In general the easier the model's use and the smaller the number of parameters and interdependencies among them, the more inaccuarcies are introduced by simplification. On the other hand a too complex model is unusable. We show that it is possible to derive a relatively accurate and easy model for small message performance over the InfiniBand network. This model allows the developer to gain knowledge about the inherent parallelism of a specific InfiniBand hardware and encourages him to use this parallelism efficiently. Several well known models hide this feature and some of them even penalize the use of parallelism because the model designers were not aware of new emerging architectures like InfiniBand.

1 Introduction

Communication models play an important role in designing and optimizing parallel algorithms or applications. A model assists the programmer in understanding all important aspects of the underlying architecture without knowing unnecessary details. This abstraction is useful to simplify the algorithm design and to enable the use of mathematical proves and runtime assessments of algorithms. Most models enable non-computer scientists to understand everything they need for programming and computer architects to give running time estimations for different architectures. These models have to be very accurate and should reflect all important underlying hardware properties. But an architectural or communication model must also be feasible for programmers. This means that the number of parameters and the model functions must not be too complex. The programmer has to understand the model and all its implications. It is easy to see that the accuracy and the ease of use are conflicting and the designer of a network model has to find the golden mean.

1.1 Related Work

Many different models have been developed in the past. There are models for specific network architectures [17, 3] or for the shared memory paradigm such as CICO [16, 7]. Other models like PRAM [6, 14], BSP [21], C^3 [9] or LogP [4] aim to be architecture independent and to give a general estimation of programming parallel systems. These are quite inaccurate due to their high level of abstraction. Several comparative studies [18, 8, 2, 11] are available for assessing the accuracy of subsets of these models. Our comparative study [11] and the prediction of the MPI_BARRIER latency [10] with LogP shows that the LogP model is quite accurate for small messages. Many efforts [1, 19, 13, 15] have been made to enhance the model in its accuracy for different network architectures and large messages.

2 The LogP Model

Several studies [2, 10, 11] have shown that the LogP model is very accurate for small messages. This accuracy, and the simplicity of this model drive to the decision to base further developments on it. The LogP model will be explained shortly in the following. It reflects all important aspects of the communication behavior of parallel systems which are seen as a collection of loosely coupled computers. Each computer has one or more processors with main memory, works asynchronously and is equipped with a network interconnect to reach all other computers. Basically, the model is based on four parameters:

- L - communication delay (**upper** bound to the latency for NIC-to-NIC messages from one processor to another)

- o - communication overhead (time that a processor is engaged in transmission or reception of a single message, split up into o_s for send overhead and o_r for receive overhead)

- g - gap (indirect communication bandwidth, minimum interval between consecutive messages, $bandwidth \sim \frac{1}{g}$)

- P - number of processors

The parameters of the LogP model can be divided into two layers, the CPU-Layer and the Network-Layer. The o-parameter can also be subdivided into one parameter on the receiver side (o_r) and another one on the sender side (o_s). The according visualization of the different parameters for a LogP compliant network (e.g. Ethernet) can be seen in figure 1.

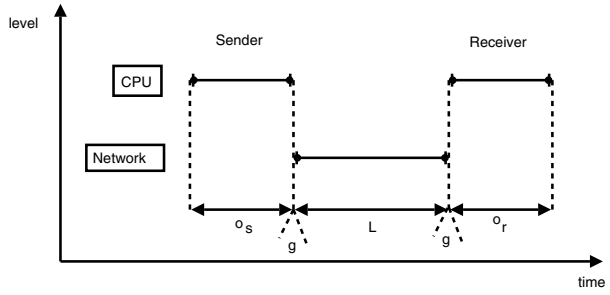

Figure 1. Visualization of the LogP parameters

The parameters have to adhere to several assumptions to make the model fully functional:

- $\left\lceil \frac{L}{g} \right\rceil$ - count of messages that can be in transmission on the network from one to any other processor in parallel (network capacity)

- L, o and g are measured as multiples of the processor cycle

An additional study [5] describes options of assessing the network parameters for real-life supercomputers. This can be very helpful to gain a deeper knowledge about the model's characteristics.

3 InfiniBand Benchmarks

This section compares different InfiniBand benchmark curves with the LogP prediction. The Benchmark determines $1 : P - P : 1$ Round Trip Times (RTT) and CPU send overheads (o) for posting the send request. We examine only RMDA-Write because it is the fastest way to transmit data over InfiniBand, and it exhibits no receive overhead o_r. Thus the only remaining overhead o_s is called o in the remainder of this paper. All curves are normalized to the number of addressed hosts. Thus, the RTT costs or o costs per message, called RTT/P and o/P are displayed over the number of addressed hosts. The LogP model predicts a constant o and the following RTT needed for transmitting a single packet to P hosts and back from all receivers to the sender. The LogP communication diagram is shown in figure 2 and the derived LogP RTT predictions are as follows:

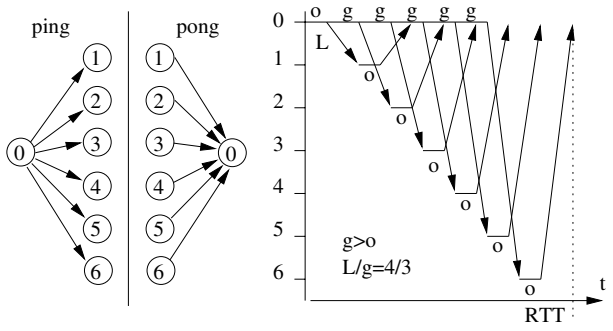

Figure 2. $1 : P - P : 1$ **Ping Pong Benchmark Scheme and LogP Communication Structure**

$$\begin{aligned} RTT &= o + (P-1) \cdot max\{o, g\} + L + o + L \\ &= 2L + 2o + (P-1) \cdot max\{o, g\} \end{aligned} \quad (1)$$

The expected graph signature of the RTT/P (cp. equation (1)) normalized by P is shown in figure 3.

The practical InfiniBand $1 : P - P : 1$ o benchmark result is shown in the lower left of figure 3 for 1 Byte and 1024KB. It can be assumed that there is no difference, and the overhead does not depend on the size of the posted message. The LogP prediction for o is constant and cannot express the benchmarked function signature properly. The RTT/P benchmark results for 1 byte messages are shown in the upper right of figure 3. The small message function has a totally different signature than the LogP prediction and the time per message has a global minimum at $P \approx 10$. This special behavior of InfiniBand^{TM} has also been investigated in

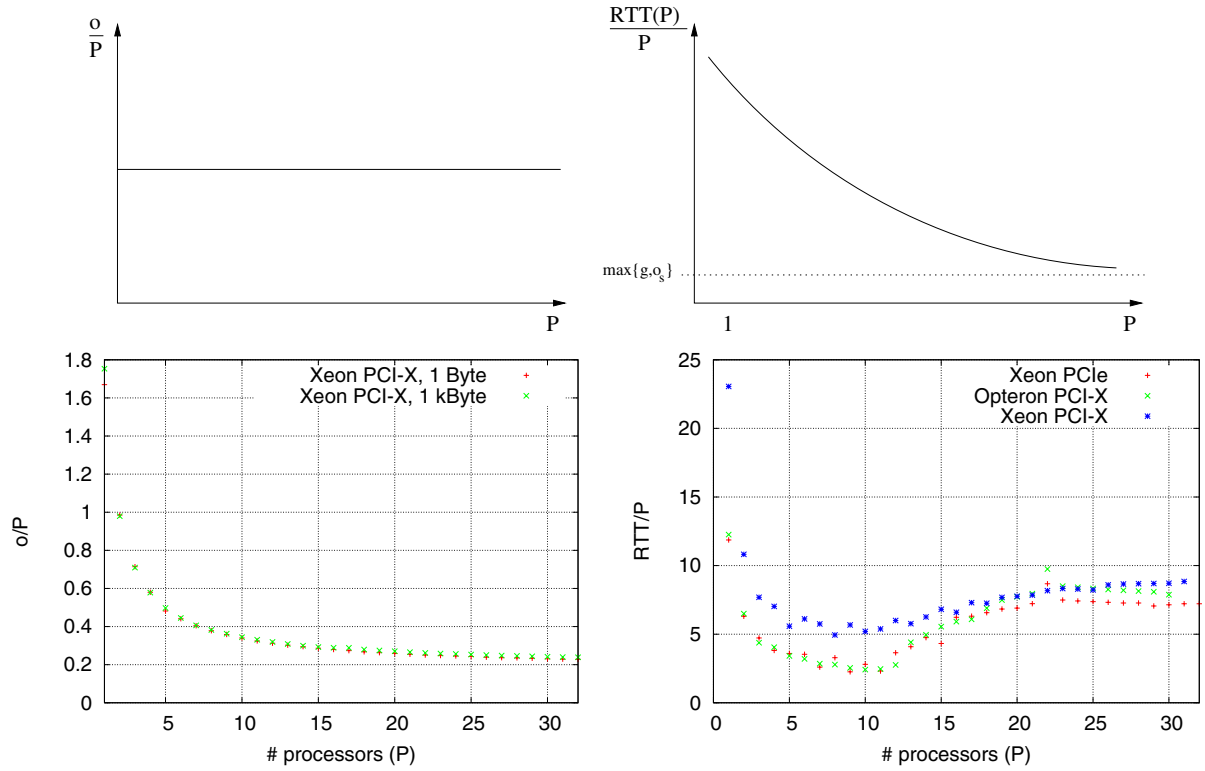

Figure 3. $o(P)$ **and** $1 : P - P : 1$ $RTT(P)$ **LogP predictions (up) and Benchmark Results (down)**

"A Communication Model for Small Messages with InfiniBand" [12] and is briefly analyzed in the following section.

4 The LoP Model

The original LoP model was presented in "A Communication Model for Small Messages with InfiniBand" [12]. The model investigates the phenomenon of the local minimum of the cost per message measured in section 3 and the varying overhead o. We derived very accurate model functions which describe the exact behavior of the InfiniBand hardware. The performance assessment function is represented by a parametrized model function. The values of the real parameters $\lambda_1 \ldots \lambda_6$ are derived by fitting the unparametrized model function to the benchmark results. Thus, not all λ_n parameters have a meaning in the "real world" and have to be seen as mathematical constructs to fit the benchmark function efficiently. P equals to the number of addressed hosts.

4.1 Overhead Model

The overhead can be modelled as a simple pipeline start up function, due to several cache effects:

$$o(P) \quad = \quad \frac{\lambda_1}{\lambda_2 + P}$$

The function signature is shown on the left side of figure 4.

4.2 RTT Model

The model for the RTT is depicted on the right side of figure 4 and is divided into three sections: The first section can be described with a typical pipeline start up function due to several cache effects and the $o(P)$ function:

$$t_{pipeline} \quad = \quad \frac{\lambda_1}{\lambda_2 + P}$$

The second section is defined by the maximum $CPU \rightarrow NIC \rightarrow NIC \rightarrow CPU$ throughput or packet processing rate of the NIC, and is thus defined as constant:

$$t_{processing} \quad = \quad \lambda_3$$

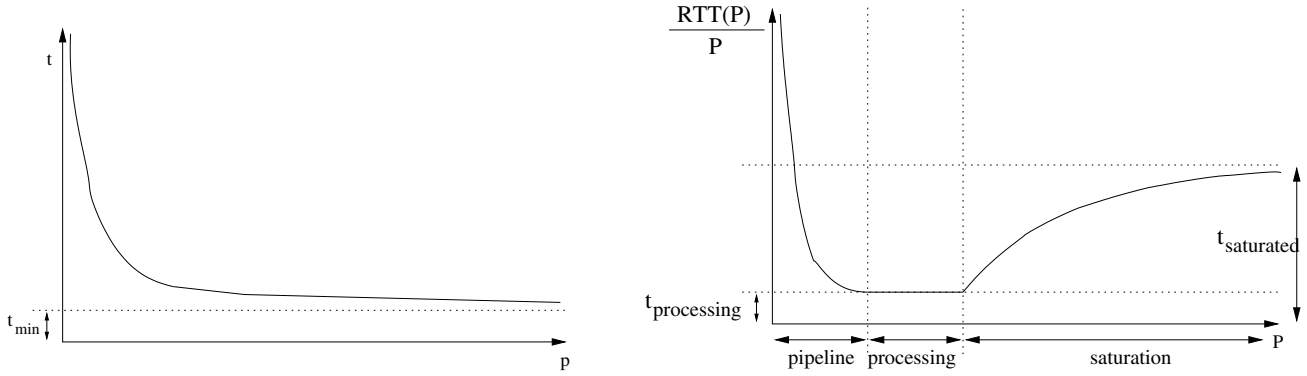

Figure 4. o (left) and RTT (right) Model

The third section reflects the network saturation which typically behaves like an exponential function as:

$$t_{saturation} = \lambda_4 \cdot (1 - e^{\lambda_5 \cdot (P - \lambda_6)})$$

λ_4 and λ_5 influence the shape of the function and λ_6 introduces a P-offset. Altogether the RTT can be described with the following abstract model, which is depicted on the right side of figure 4:

$$\begin{aligned} RTT(P)_{\lambda_{1\ldots6}} &= t_{pipeline} + t_{processing} + t_{saturation} \\ &= \frac{\lambda_1}{\lambda_2 + p} + \lambda_3 + \lambda_4 \cdot (1 - e^{\lambda_5 \cdot (p - \lambda_6)}) \end{aligned}$$

It is nearly impossible to handle six parameters as well as an exponential function to design optimal algorithms and even the mathematical proves are very hard to do. Another disadvantage is the complexity of finding the optimal parameters for a given set of benchmarks (we used a direct search in a six dimensional space with the Nelder Mead simplex algorithm [20]).

Thus, this model is practically unusable. Basing on the relatively simple LogP model and the fact that it covers most of our architectural needs, we decided to enhance it but keep its simple linear nature. The derivation of the new simplified LoP model is called LogfP because the additional parameter f defines the maximal number of send operations where no g is needed.

5 LogfP - A simplified LoP Model

The simplified LoP model, named LogfP, is derived from the original LogP model. The main characteristics and the ease of use are retained in the new design. Figure 5 shows the benchmarked o/P and RTT/P values, our proposed model function and the LogP predicted function. The f parameter indicates the number of messages where no g has to be accounted, which are essentially for $f$ree.

5.1 Overhead Model

The overhead model is simply the pipeline function stated in 4.1. The parameters are more mnemonic:

$$o(P) = o_{min} + \frac{o_{max}}{P} \quad (2)$$

Where o_{min} is the lowest achievable $o(P)$ for $P \to \infty$ which is $0.18\mu s$ in our benchmark in figure 3. The maximal value for $o(P)$, o_{max} is exactly $o(1)$ and $1.6\mu s$ in our example. It defines the shape of the prediction function. Both parameters are very easy to derive if you have just two measurement results, $o(1)$ and $o(\text{inf}) \approx o(x)$ for a sufficiently huge x (e.g. $x = 1000$). This model is still extremely accurate and easy to use. The model's prediction and the measured values are shown in the left of figure 5. The $o(P)$ parameter is very important to assess the CPU load for each send operation. It does not play a big role for the send process itself because the L parameter is usually 10 to 100 times bigger for InfiniBand. Thus, the complicated $o(P)$ could be replaced with the scalar o from the LogP model for network transmissions (this introduces a slight inaccuracy but reduces the number of parameters by one!).

5.2 RTT Model

Our model is nearly the same as the LogP model. The difference is that the g parameter is not paid for every message after the first one. We assume that multiple small messages can be processed simultaneously in the network hardware (cmp. [12]) and sent simultaneously across the network. Thus, g is only paid for

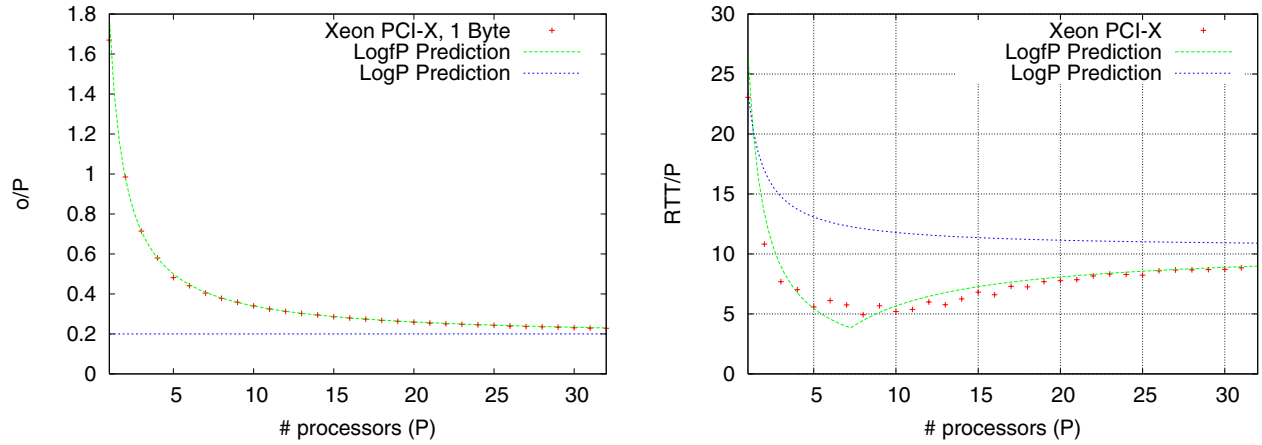

Figure 5. $1 : P - P : 1$ o **and** RTT **Benchmark Results with Predictions**

every message after f messages have been sent, which means that the first f small messages are essentially for $f$ree in our model. It is obvious that this cannot hold for large messages, due to the limited bandwidth. Thus, the Round Trip Time can be modelled as:

$$\begin{aligned} \forall(P \leq f)\ RTT(P) &= 2L + P \cdot o_s(P) + o_s(1) \\ \forall(P > f)\ RTT(P) &= 2L + o(P) + o_s(1) + \\ & \quad max\{(P-1) \cdot o(P), (P-f) \cdot g\} \end{aligned}$$

It is easy to see that our simple modification of introducing the f parameter enhances the accuracy of the model significantly. The LogfP model is quite accurate for the prediction of small messages while the LogP model overestimates all RTTs. The introduction of the $o_{min,max}$ parameters enhances the o modelling of the LogP model. LogP underestimates the needed CPU time to send a message due to its constant nature.

5.3 LogfP Parameter Assessment

All LogfP parameters can be gathered from the $1 : P - P : 1$ benchmark described above. They are explained in the following:

- o_{min} - equals to $o(\infty)/P$ of the $o(P)$ benchmark
- o_{max} - equals to $o(1)/P$ of the $o(P)$ benchmark
- L - equals $\frac{RTT(1)-2o_{min}-2o_{max}}{2P}$ of the $RTT(P)$ benchmark
- g - equals $RTT(\infty)/P$ of the $RTT(P)$ benchmark
- f - is the global minimum of the $RTT(P)/P$ curve
- P - number of processors

These parameters can easily be measured and used for modelling the running time of parallel algorithms which use small messages (e.g. the MPI_BARRIER algorithm).

6 Conclusions and Future Work

We have shown that simple modifications can enhance the accuracy of the LogP model significantly. These modifications are also applicable to a developer's daily task to optimize algorithms. The predictions of the LoP model are much more accurate but due to their complexity not usable. Our model encourages the programmer to use the inherent hardware parallelism in the transmission of small messages, because sending the first f messages is for free (despite o). Our model should be more accurate than the LogP model for other hardware offloading based networks where most packet processing is done in hardware (e.g. Quadrics, Myrinet). Future work includes enhancing the model also for large messages (cmp. LogGP [1]) and to provide several use-cases of the model to the community.

6.1 Acknowledgments

We would like to thank Tobias Klug and Carsten Trinitis from the Technical University of Munich for providing access to their InfiniBand Cluster and helpful comments. Additionally we want to thank Jens Simon from the University of Paderborn and the University of Stuttgart for providing access to their InfiniBand Cluster systems.

References

[1] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating Long Messages into the LogP Model. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1995.

[2] G. Bilardi, K. T. Herley, and A. Pietracaprina. BSP vs LogP. In *SPAA '96: Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures*, pages 25–32. ACM Press, 1996.

[3] G. Blelloch. Scans as Primitive Operations. In *Proc. of the International Conference on Parallel Processing*, pages 355–362, August 1987.

[4] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: towards a realistic model of parallel computation. In *Principles Practice of Parallel Programming*, pages 1–12, 1993.

[5] D. Culler, L. T. Liu, R. P. Martin, and C. Yoshikawa. LogP Performance Assessment of Fast Network Interfaces. *IEEE Micro*, February 1996.

[6] S. Fortune and J. Wyllie. Parallelism in random access machines. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 114–118. ACM Press, 1978.

[7] P. G. Gibbons, Y. Matias, and V. Ramachandran. Can a shared memory model serve as a bridging model for parallel computation? In *ACM Symposium on Parallel Algorithms and Architectures*, pages 72–83, 1997.

[8] S. E. Hambrusch. Models for parallel computation. In *ICPP Workshop*, pages 92–95, 1996.

[9] S. E. Hambrusch and A. A. Khokhar. An architecture-independent model for coarse grained parallel machines. In *Proceedings of the 6-th IEEE Symposium on Parallel and Distributed Processing*, 1994.

[10] T. Hoefler, L. Cerquetti, T. Mehlan, F. Mietke, and W. Rehm. A practical Approach to the Rating of Barrier Algorithms using the LogP Model and Open MPI. In *Proceedings of the 2005 International Conference on Parallel Processing Workshops*, pages 562–569, June 2005.

[11] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. A Survey of Barrier Algorithms for Coarse Grained Supercomputers. *Chemnitzer Informatik Berichte - CSR-04-03*, 2004. url: http://archiv.tu-chemnitz.de/pub/2005/0074/data/CSR-04-03.pdf.

[12] T. Hoefler, T. Mehlan, F. Mietke, and W. Rehm. A Communication Model for Small Messages with InfiniBand. *PARS Proceedings*, 2005.

[13] F. Ino, N. Fujimoto, and K. Hagihara. Loggps: a parallel computational model for synchronization analysis. In *PPoPP '01: Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming*, pages 133–142, New York, NY, USA, 2001. ACM Press.

[14] R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In J. Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume A: Algorithms and Complexity*, pages 869–941. Elsevier, Amsterdam, 1990.

[15] T. Kielmann, H. E. Bal, and K. Verstoep. Fast measurement of logp parameters for message passing platforms. In *IPDPS '00: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, pages 1176–1183, London, UK, 2000. Springer-Verlag.

[16] J. R. Larus, S. Chandra, and D. A. Wood. CICO: A Practical Shared-Memory Programming Performance Model. In Ferrante and Hey, editors, *Workshop on Portability and Performance for Parallel Processing*, Southampton University, England, July 13 – 15, 1993. John Wiley & Sons.

[17] F. T. Leighton. *Introduction to parallel algorithms and architectures: array, trees, hypercubes*. Morgan Kaufmann Publishers Inc., 1992.

[18] B. M. Maggs, L. R. Matheson, and R. E. Tarjan. Models of Parallel Computation: A Survey and Synthesis. In *Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS)*, volume 2, pages 61–70, 1995.

[19] C. A. Moritz and M. I. Frank. LoGPC: Modelling Network Contention in Message-Passing Programs. *IEEE Transactions on Parallel and Distributed Systems*, 12(4):404, 2001.

[20] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J. 7*, pages 308–313, 1965.

[21] L. G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990.