Modeling Contention and Mapping Effects in Multi-core Clusters

Juan-Antonio Rico-Gallego[1](✉), Juan-Carlos Díaz-Martín[1], and Alexey L. Lastovetsky[2]

[1] University of Extremadura, Avd. Universidad s/n, 10003 Cáceres, Spain
{jarico,juancarl}@unex.es
[2] University College Dublin, Belfield, Dublin 4, Ireland
Alexey.lastovetsky@ucd.ie

Abstract. Modeling and formal analysis of parallel algorithms contribute to optimize their performance. Modern multi-core are complex machines composed of heterogeneous shared communication channels. Parallel Performance Models estimate the cost and capture the behavior of parallel algorithms through a set of parameters, providing valuable information about the behavior of the algorithm in these platforms. LogGP is a representative model using network related parameters to predict the cost of parallel algorithms as a sequence of point-to-point transmissions. Although extensions have been proposed for covering issues derived from modern platforms complexities as contention and channels hierarchy, such specific extensions are not enough to meaningfully and accurately model more than simple algorithms. τ–Lop is an alternative model that takes as a building block for modeling parallel algorithms the concept of concurrent transfers, that helps to capture algorithms behavior and allows to represent and accurately predict their cost in multi-core clusters. This paper shows the analysis capabilities of τ–Lop through two cases of study involving elaborated MPI collective operations.

1 Introduction

Parallel performance models estimate the cost and capture the behavior of parallel algorithms. Message passing defined in the MPI standard [8], is the most used programming model for building scientific applications. MPI is based on independent processes communicating by point-to-point messages and collective operations, whose costs have an important influence in the global performance of applications. The performance of the underlying algorithms implementing MPI collectives vary significantly with the message size, number of involved processes, and the platform characteristics. Modeling and formal analysis of such algorithms is important to gain insights into their performance.

LogGP [1] is a representative message-passing performance model. It was conceived to estimate the cost of a point-to-point message between two processes based on network-related parameters basically representing the latency

S. Hunold et al. (Eds.): Euro-Par 2015 Workshops, LNCS 9523, pp. 197–208, 2015.
DOI: 10.1007/978-3-319-27308-2_17

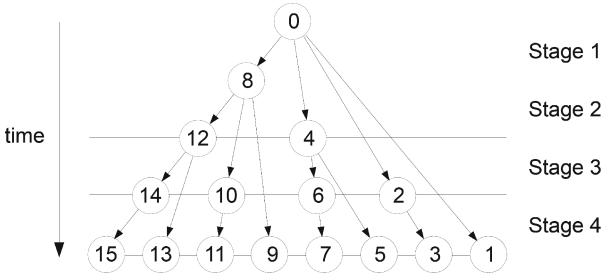

Fig. 1. A *binomial tree* used in MPI_Bcast collective operation.

and bandwidth values. LogGP assumes that collective algorithms are built upon point-to-point message transmissions deployed over a network of monoprocessors and models an algorithm as a sequence of stages, each with the cost of a single point-to-point transmission. For instance, a broadcast operation on P processes based on the *binomial tree* algorithm has a cost that equals the height of the tree ($\lceil \log P \rceil$) times the cost of a single transmission between two processes (see Fig. 1). In modern HPC platforms this is an over-simplification that leads to important errors in the cost estimation of an algorithm. In multi-core clusters, processes communicate through different physical channels with very different performance, and usually sharing the available bandwidth. To deal with some of these complexities, extensions of LogGP have appeared, but oriented to specific platforms and topics ([3,7,12]).

More recent $\log_n$P [2] introduces the concept of *transfer* for describing a point-to-point transmission as a sequence of movements of data between memory entities. Nevertheless, $\log_n$P is still simplistic and it has not demonstrated its capabilities to model complex algorithms or basic techniques as the segmentation of messages. τ–Lop [10], rooted on $\log_n$P, models algorithms as a sequence of *concurrent transfers* progressing through available communication channels, what allows to capture the effects of the contention in multi-core clusters.

The goal of this paper is to evaluate the capacities of τ–Lop to model the parallel algorithms used regularly in the implementation of MPI collective operations. We present two cases of study in comparison to LogGP. The first case of study shows the influence of the contention on the cost of an algorithm, as well as the capacity of these models to represent it. As a conduit example, two common collective algorithms are modeled in the shared memory communication channel: *binomial tree* and *recursive doubling*, implementing the *MPI_Scatter* and the *MPI_Allgather* operations respectively. The second case of study discusses the *Ring* algorithm when it is used in the *MPI_Allgather* collective. Its cost is predicted for two common mappings under τ–Lop and a LogGP offspring on a multi-core cluster.

Section 2 below briefly reviews some related works in the field of modeling parallel algorithms. Section 3 introduces the models used in this paper, LogGP

and τ–Lop. Sections 4 and 5 develop the cases of study. Section 6 gives details on the platforms used for the performance measurements, and Sect. 7 concludes.

2 Related Work

LogGP [1] is a widespread model which characterizes the communication cost according to the five parameters that compose its name: the network latency (L), the overhead or time a processor is engaged in the transmission or reception of a message (o), the time interval between consecutive message transmissions (g), and the time interval between consecutive byte transmissions (G). The cost of a transmission is formulated in terms of these parameters and the size in bytes of the message (m), as $o_S + L + (m-1)\,G + o_R$, i.e., the overhead in the sender processor, the latency in the network of the first byte, the time of sending the rest ($m-1$ bytes) of the message, and the overhead in the reception processor. For simplicity, it is usually assumed that $o = (o_S + o_R)/2$.

LogGP shows powerful analysis capabilities to guide the development of applications and algorithms. A thorough study of known MPI collective algorithms and its performance under the LogGP model can be found in [9].

Models extending LogGP contribute with specific aspects to model complex communication in modern platforms. *LogGPS* [3] provides the rendezvous protocol cost, *LoGPC* [7] adds contention in meshes with wormhole routing, and *LogGPH* [12] supports representation for hierarchical networks by splitting up the parameters in sets for modeling transmissions over each communication channel.

The *Parametrized* LogP model [4] addresses the performance modeling of parallel algorithms by slightly changing the meaning of some parameters of LogGP and making them (except latency) dependent on the message size. More recent *LMO* model [5] estimates the cost of algorithms in both homogeneous and heterogeneous systems. The per-node parameters separate the cost imputable to the processors and the network in order to gain more accurate results.

The contention modeling issue in Ethernet networks is carried out in [6]. The authors generate an specific model for this type of network based on the study of flow control mechanisms and experimentally deduce *penalty coefficients*, and categorize different types of conflicts. Paper [11] introduces the *contention factor*, represented as a parameter affecting the network performance, which is linearly incremented from the contention-free communication cost.

3 Modeling Parallel Algorithms

LogGP models a parallel algorithm as a sequence of point-to-point transmissions of m bytes between P processes. For instance, the cost of a *Binomial Tree* broadcast algorithm, represented in Fig. 1, is calculated as:

$$T_{Bin}^{LogGP} = \lceil \log P \rceil \times T_{p2p} = \lceil \log P \rceil \times (L + 2\,o + (m-1)\,G)$$

τ–Lop considers a message transmission composed of a sequence of *concurrent transfers*. The *Transfer time*, denoted $L(m, \tau)$, is the cost of τ concurrent transfers of a message of size m. The *overhead* (o) is defined as the elapsed time since a message transmission operation is invoked until data begins to be injected in the communication channel, including the communication protocol time. The cost of a transmission is the sum of the costs of its individual transfers plus the overhead. Only contiguous data messages are considered in this paper. For instance, a shared memory transmission could be divided up in two transfers in sequence, from the sender buffer to a pre-allocated intermediate buffer and then to the receiver buffer, with the cost:

$$T_{p2p}(m) = o(m) + 2\,L(m, 1)$$

The number of transfers depends on the channel, but also on the particular implementation of the algorithm and the middleware, a fact ignored by LogGP and derived models.

In respect of collective operations, they are represented in τ–Lop as a sequence of stages, formed in turn of a set of concurrent transmissions. Under a shared communication channel, if the cost of the message transmission between processes 0 and 8 in Fig. 1 is $T_{p2p}(m)$, then the cost between processes 14 and 15 is much higher due to the fact that eight concurrent transmissions share the channel bandwidth. The cost ranges from $T_{p2p}(m)$ for a perfectly parallel channel, to $8 \times T_{p2p}(m)$ for a serial one. The $\|$ operator [10] captures this concurrency effect and produces compact and expressive cost equations. For instance, τ–Lop represents the binomial tree algorithm on P processes as:

$$\Theta_{Bin}^{\tau-Lop}(m) = \sum_{i=0}^{\lceil \log_2(P) \rceil - 1} \left(2^i \,\|\, T_{p2p}(m)\right) = \sum_{i=0}^{\lceil \log_2(P) \rceil - 1} \left(2^i \parallel (o(m) + 2\,L(m, 1))\right)$$
$$= \sum_{i=0}^{\lceil \log_2(P) \rceil - 1} \left(o(m) + 2\,L(m, 2^i)\right)$$

Note how the $\|$ operator affects to the transfers concurrency (τ parameter). The overhead is attributable to the processor, and hence not affected by the operator.

4 Case Study 1: Analyzing the Effect of the Contention in Shared Memory

This section discusses the influence of the communication channel contention in the performance of algorithms through the study of two common algorithms, the *Binomial Tree* and the *Recursive Doubling*, commonly used in the *scatter* and *allgather* collectives respectively.

We use segmentation of messages, a technique commonly used in shared memory. It operates by breaking a message into k smaller chunks of a fixed size S known as *segments*, hence $m = k\,S$. The segments are sent in sequence to

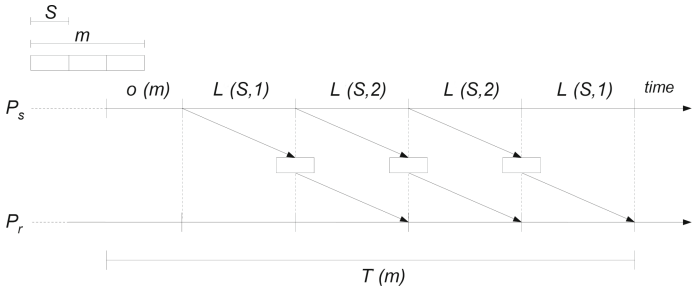

Fig. 2. A segmented point-to-point message transmission in shared memory modeled by τ–Lop. Unlike LogGP, the effect of the intermediate buffers is explicitly represented.

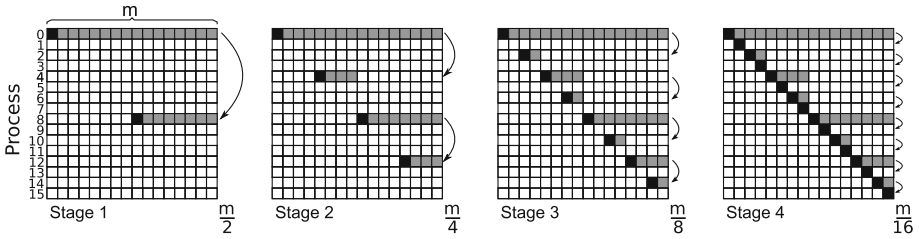

Fig. 3. The scatter binomial tree algorithm for $P = 16$ processes and $\log_2(P) = 4$ stages. An arc represents a point-to-point transmission T. The root has rank 0.

an intermediate shared buffer allowing the overlap with their reception. LogGP models the cost of a point-to-point segmented message as:

$$\begin{aligned} T_{p2p_S}^{LogGP} &= L + 2\,o + (S-1)\,G + (k-1)\,(g + (S-1)\,G) \\ &\approx L + 2\,o + S\,G + (k-1)\,(g + S\,G) \end{aligned}$$

In τ–Lop, a segmented point-to-point message transmission is composed by transfers of the first and last segments proceeding alone, while the intermediate segments are copied concurrently, for a total of $2\,k$ transfers (see Fig. 2). Hence with a cost of:

$$T_{p2p_S}(m) = 2\,L(S,1) + (k-1)\,L(S,2) \approx o(m) + k\,L(S,2)$$

The scatter binomial tree is shown in Fig. 3. In each stage, the root of each subtree receives all the data its descendants expect. The number of concurrent transmissions doubles in each stage, while the size of the message halves.

Under LogGP, the total cost for the scatter is calculated as the cost of the larger path from the root process to a leaf, shown in the Table 1.

For its part, the recursive doubling also completes in $\lceil \log_2 P \rceil$ stages, as shown in Fig. 4. In each stage k, the process with rank p performs an interchange of data with the process with rank $p \oplus 2^k$. The initial size of the message is $m' = m/P$, and it doubles at each stage.

Table 1. The *Binomial Scatter* modeled with LogGP

Stage	Message size	Cost
1	$m/2$	$L + 2o + S\,G + ((k-1)/2)\,(g + S\,G)$
2	$m/4$	$L + 2o + S\,G + ((k-1)/4)\,(g + S\,G)$
3	$m/8$	$L + 2o + S\,G + ((k-1)/8)\,(g + S\,G)$
. .		
$\log_2 P$	$m/2^{\log_2 P}$	$L + 2o + S\,G + ((k-1)/P)\,(g + S\,G)$
Scatter cost		$\log_2 P\,(L + 2o + S\,G) + ((P-1)/P)\,(k-1)\,(g + S\,G)$

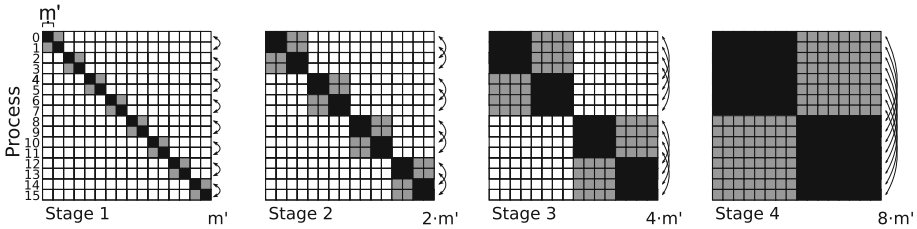

Fig. 4. The allgather recursive doubling algorithm for $P = 16$ processes. An arc represents a *sendrecv* message transmission between two processes.

Table 2. The *Recursive Doubling Allgather* modeled with LogGP

Stage	Message size	Cost
1	m/P	$L + 2o + S\,G + (1\,(k-1)/P)\,(g + S\,G)$
2	$2\,m/P$	$L + 2o + S\,G + (2\,(k-1)/P)\,(g + S\,G)$
3	$4\,m/P$	$L + 2o + S\,G + (4\,(k-1)/P)\,(g + S\,G)$
. .		
$\log_2 P$	$2^{\log P-1}\,m/P$	$L + 2o + S\,G + ((k-1)/2)\,(g + S\,G)$
Allgather cost		$\log_2 P(L + 2o + S\,G) + ((P-1)/P)\,(k-1)\,(g + S\,G)$

Table 2 shows the allgather cost modeling in LogGP. Interestingly, both operations have the same cost under LogGP, a definitely inaccurate result. The whole amount of data moved is quite different: $\log_2 P \cdot (m/2)$ bytes in *scatter* versus $(P-1) \cdot m$ bytes in *allgather*, as can be deduced from Figs. 3 and 4.

The binomial scatter algorithm is modeled by τ–Lop as[1]:

$$\begin{aligned} \Theta_{Sct_S}(m) &= \sum_{i=0}^{\log P-1} \left(2^i \parallel T_{p2p_S}\left(\frac{m}{2^{i+1}}\right)\right) = \sum_{i=0}^{\log P-1} \left(2^i \parallel \left(\frac{k}{2^{i+1}}\,L\,(S, 2)\right)\right) \\ &= \sum_{i=0}^{\log P-1} \left(\frac{k}{2^{i+1}}\,L\left(S, 2^{i+1}\right)\right) \end{aligned} \tag{1}$$

[1] The overhead (o(m)) is omitted for clarity in the rest of the paper.

With respect to the recursive doubling algorithm (RDA), Fig. 4 shows that the exchange of messages is done under a pattern of pairs of processes. Each process sends a message to and then receives a message from its partner. The send requires a k segments transfer to the intermediate buffer, and the receive requires an additional k segments transfer from the intermediate buffer. Therefore, the *exchange* operation has a per process cost of $T_{exch_S}(m) = o(m) + 2\,k\,L(S,1)$. Being P the number of processes involved, the cost of a stage is:

$$P \parallel T_{exch_S}(m) = P \parallel [o(m) + 2\,k\,L(S,1)] = o(m) + 2\,k\,L(S,P) \tag{2}$$

Note that $T_{exch_S} \geq T_{p2p_S}(m)$, because a sequence of $2\,k$ transfers perform worse that a sequence of k concurrent pairs, a fact with a key influence on the accuracy of the cost prediction. The RDA cost results in:

$$\begin{aligned} \Theta_{RDA_S}\left(\frac{m}{P}\right) &= \sum_{i=0}^{\log P-1}\left(P \parallel T_{exch_S}\left(2^i\,\frac{m}{P}\right)\right) = \sum_{i=0}^{\log P-1}\left(P \parallel \left(\frac{2^{i+1}\,k}{P}\,L\,(S,1)\right)\right) \\ &= \sum_{i=0}^{\log P-1}\left(\frac{2^{i+1}\,k}{P}\,L\,(S,P)\right) \end{aligned} \tag{3}$$

We can further expand definitions (1) and (3) as:

$$\Theta_{Sct_S}(m) = \frac{k}{2}\,L(S,2) + \frac{k}{4}\,L(S,4) + \cdots + \frac{k}{P}\,L(S,P) \tag{4}$$

$$\Theta_{RDA_S}\left(\frac{m}{P}\right) = \frac{2k}{P}\,L(S,P) + \frac{4k}{P}\,L(S,P) + \cdots + \frac{Pk}{P}\,L(S,P) = 2k\frac{P-1}{P}\,L(S,P)$$

Introducing additional contention in (4) in all the stages except in the last one, we get an upper bound for $\Theta_{Sct_S}(m)$:

$$\Theta'_{Sct_S}(m) = \frac{k}{2}\,L(S,P) + \frac{k}{4}\,L(S,P) + \cdots + \frac{k}{P}\,L(S,P) = k\,\frac{P-1}{P}\,L(S,P)$$

So how do RDA and Scatter compare? We have that $\Theta_{RDA_S}(m) = 2\,\Theta'_{Sct_S}(m) > 2\,\Theta_{Sct_S}(m)$, an expression that shows that the cost of Θ_{RDA_S} exceeds more than twofold that of scatter in Θ_{Sct_S}. Figure 5 shows both the RDA and Scatter costs measured in the *Lusitania* platform (described in Sect. 6) for different P values. The relative mean costs ($\Theta_{RDA_S}/\Theta_{Sct_S}$) for the range of message sizes shown are 2.61 for $P = 32$, 2.83 for $P = 64$ and 3.02 for $P = 128$, increasing with the number of processes because of the difference in the contention between the operations. For any m and P, RDA cost more than doubles that of scatter, what confirms the τ–Lop prediction.

5 Case Sudy 2: Modeling the Mapping Effects on Multi-core Clusters

This section shows the capabilities of the models to capture the mapping effect in the overall cost. The process mapping has a direct influence in the channels

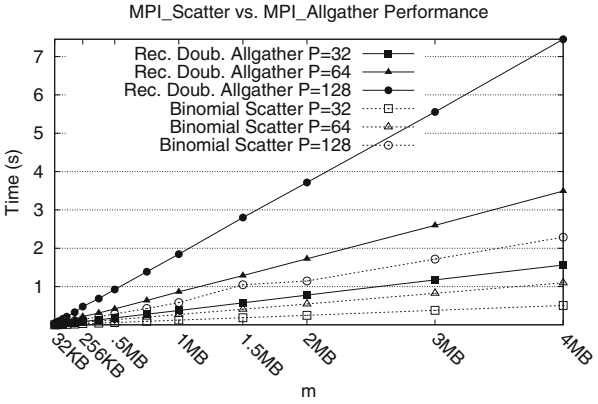

Fig. 5. The MPICH performance of the Binomial Tree Scatter and the Recursive Doubling Allgather with different number of processes P in the Lusitania machine.

used for message transmissions over the multi-core cluster, and indeed in the contention in such channels. As a conduit example, we discuss the *Ring* algorithm when it is used in the *MPI_Allgather*. We consider the *Metropolis* platform (see Sect. 6), with several multi-core nodes connected by Ethernet. Hence, the processes communicate through two channels: shared memory and network. Q is the number of cores per node and M the number of nodes, with $P = Q \times M$. Two common mappings are evaluated: *Sequential* mapping deploys processes on cores in a way that a rank p runs in the node $p \div Q$, and *Round Robin* mapping runs p in the node $p\%M$.

The Ring algorithm is executed in $P-1$ stages. In each stage, the process p invokes the *MPI_Sendrecv* operation for sending m bytes to the process with number $p+1$, and then receiving from the process with number $p-1$ a message of the same size (with wraparound). Under sequential mapping (left side of Fig. 6), the destination and the source ranks of both point-to-point transmissions could be in the same or different node, therefore, transmissions could progress through shared memory or network. LogGP models the cost of a stage as the cost of the most expensive send-receive, that is, the addition of a shared memory and a network transmissions ($Q \geq 2$). Hence, the cost will be:

$$T_{Ring}^{SEQ} = (P-1) \times [(L + 2\,o + (m-1)\,G) + (L' + 2\,o' + (m-1)\,G')] \qquad (5)$$

where L is the latency of the shared memory channel and L' that of the network, and likely for o and G. This extension to LogGP is proposed in the LogGPH model. The cost for long messages approaches to $(P-1)\,m\,(G+G')$, with the latency and the overhead costs negligible, and considering $G' \gg G$.

Under round robin mapping (see left side of the Fig. 7) all the messages travel through the network. Hence, the cost will be:

$$T_{Ring}^{RR} = (P-1) \times 2 \times (L' + 2\,o' + (m-1)\,G') \qquad (6)$$

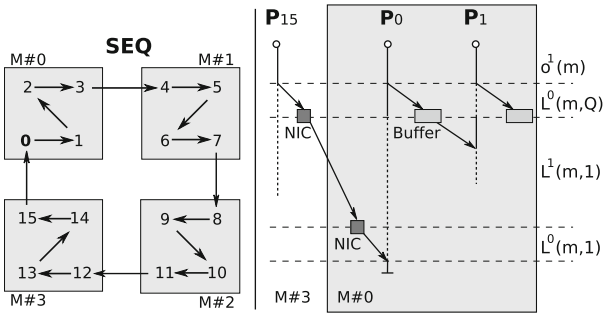

Fig. 6. The Ring algorithm with $P = 16$ processes in $M = 4$ nodes with *sequential* mapping, and a message transmission of P_0 deployed at right side.

The cost for long messages approaches to $(P-1)\,2\,m\,G'$, nearly doubling the sequential mapping cost. An important fact is that under LogGP both cost expressions (5) and (6) are independent of the number of processes per node Q, again a non plausible result, in our view.

As LogGP, τ–Lop models the send-receive operation (T_{sr}) as the addition of two point-to-point transmissions. The cost of each individual transmission depends on the channel it progresses (c) and on the contention it suffers, this derived from the process mapping and represented by C^{map}. The *Ring* cost is:

$$\Theta_{Ring}(m) = (P-1) \times [C^{map} \parallel T_{sr}(m)] = (P-1) \times \left[C^{map} \parallel \left(T^{c}_{p2p}(m) + T^{c}_{p2p}(m)\right)\right]$$

Under sequential mapping, the contention affects only to shared memory ($c = 0$), as shown at left side of Fig. 6, with Q concurrent transmissions progressing through shared memory and only one through the network ($c = 1$). The cost will be:

$$\begin{aligned}\Theta^{SEQ}_{Ring}(m) &= (P-1) \times \left[C^{SEQ} \parallel \left(T^{0}_{p2p}(m) + T^{1}_{p2p}(m)\right)\right] \\ &= (P-1) \times \left[Q \parallel T^{0}_{p2p}(m) + 1 \parallel T^{1}_{p2p}(m)\right]\end{aligned}$$

Besides, τ–Lop allows an alternative lower analysis of the behavior of the Ring operation at transfer level. We consider a transmission through the network composed of three intermediate transfers: first from the sender buffer to the NIC, second from NIC to NIC, and last from the NIC to the receiver buffer. Right side of the Fig. 6 shows the decomposition into transfers of one stage of the Ring under sequential mapping. The first transfer of all processes, either to the NIC or to the intermediate buffer, progresses concurrently through shared memory, hence with the cost $L^0(m, Q)$. Then, a network send transfer of one process per node starts while the rest of processes complete communication through shared memory. These inter- and intra-node transfers progress through different channels, and hence, they do not contend, leading to a cost of $max\{L^0(m, Q-1), L^1(m, 1)\}$. For instance, with $Q = 8$ in *Metropolis*, $L^1(m, 1) \gg L^0(m, Q-1)$, leading to the cost:

$$\Theta^{SEQ}_{Ring}(m) = (P-1) \times \left[L^0(m, Q) + L^1(m, 1) + L^0(m, 1)\right] \tag{7}$$

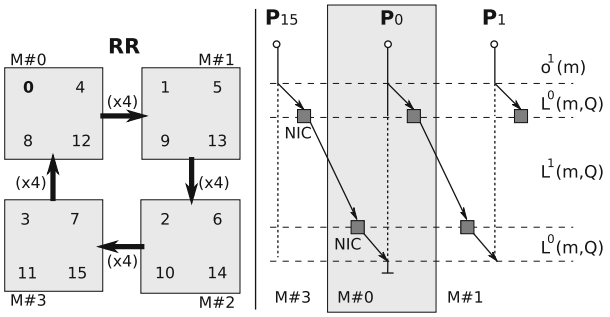

Fig. 7. The Ring algorithm with $P = 16$ processes in $M = 4$ nodes with round *robin mapping*, and a message transmission of P_0 deployed at right side.

Note that Q affects only to shared memory transfers.

The cost under round robin mapping comes determined by Q concurrent send and receive network transmissions, with the cost:

$$\Theta^{RR}_{Ring}(m) = (P-1) \times \left[C^{RR} \parallel T_{sr}(m)\right] = (P-1) \times \left[Q \parallel \left(T^1_{p2p}(m) + T^1_{p2p}(m)\right)\right]$$

Again, at transfer level, the right side of the Fig. 7 shows the decomposition of the Q concurrent transmissions through the network, leading to:

$$\Theta^{RR}_{Ring}(m) = (P-1) \times \left[L^0(m,Q) + L^1(m,Q) + L^0(m,Q)\right] \tag{8}$$

The comparison of (7) and (8) shows that the difference is the contention in the channels, determined by the number of processes per node Q, a fact not represented by LogGP. Figure 8 shows the execution cost in terms of the bandwidth of the Ring algorithm, compared to LogGP and τ–Lop former estimations. The left plot (sequential mapping) shows that the influence of the contention on the total cost is relatively small, due to it occurs in the faster shared memory channel. Note the performance increasement effect of the piggybacking mechanism of TCP in small messages, not captured in the models, which provide pessimistic predictions. The right plot shows that round robin mapping saturates links between the nodes with Q concurrent transmissions, making long messages performance to decrease sixfold with only $M = 4$ and $Q = 8$. As LogGP does not model the contention derived cost, its cost estimation is very optimistic.

6 Test Platforms

To model the contention in shared memory (first case of study), we use the *Lusitania* platform. It is a ccNUMA HP Superdome sx2000 machine with 64 Dual-Core Intel Itanium2 Montvale 9140 N processors, with a private $9MB$ L3 cache, that make up a total of 128 cores, split into 16 UMA nodes. *Lusitania* is used as an SMP test platform due to similar intra-UMA and inter-UMA communication channels performance. The operating system is Linux 2.6.16.

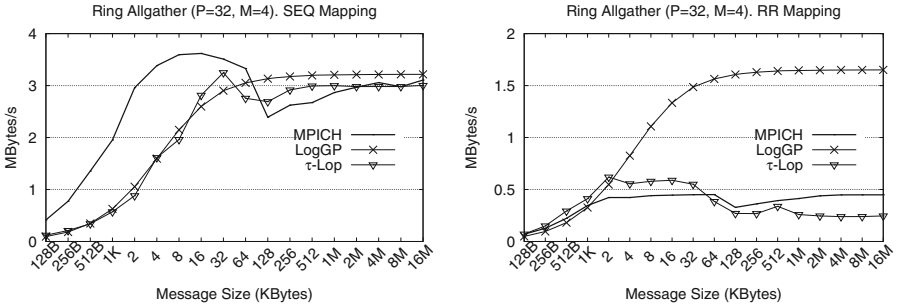

Fig. 8. Estimation of the cost of an Allgather *Ring* with LogGP and τ–Lop models, compared to the real MPICH measured bandwidth. The number of processes is $P = 32$, deployed on $M = 4$ nodes under two mappings, sequential (left) and round robin (right).

To model the mapping effect (second case of study), the platform used, named *Metropolis*, consists of four nodes connected by 10 Gigabit Ethernet through a switch. Each node has two 2.4 GHz Intel Xeon E5620 processors, making a total of eight cores per node, hyperthreading disabled, with a shared 12 MB L3 cache. The operating system is Linux 2.6.32.

IMB (Intel MPI Benchmark) running on MPICH version 1.4.1p1 is used to obtain the execution time in both platforms. The flag turning cache reuse is deactivated to avoid cache effects.

The MPI LogP Benchmark [4] is used for estimating the values of the parameters of the LogGP model. The methodology for measuring the τ–Lop parameters is out of the scope of this paper, and it is described in depth in [10].

7 Conclusions

LogGP is a representative model that estimates the cost of a collective algorithm as the cost of the longest path of the involved point-to-point transmissions. In the other hand, τ–Lop is a new model that uses the *concurrent transfer* concept.

We compare both models through two cases of study. First, we formally evaluate and analyze the influence of the contention in a shared communication channel as shared memory. The bandwidth shrinks in shared channels when several transmissions progress in parallel. τ–Lop is able to represent such concurrent transmissions through a simple but still powerful notation, with the overall result of improving the cost estimation. It has been shown that two algorithms as different as the Scatter Binomial and the Recursive Doubling Allgather are allocated the same cost by LogGP. Second, we model the influence of the process mapping on the cost of an algorithm, derived from the algorithms communication patterns. Such mapping decides the channels used for communicating in the multi-core cluster. Avoiding the representation of the mapping in a model leads to wrong cost predictions. A transfer-based model as τ–Lop offers a higher analysis capacity that provides a good accuracy of the cost prediction.

An additional conclusion of the study is that modern HPC platforms demand better parallel performance models, able of capturing their growing complexity, in particular the increasing number of cores per node and their heterogeneity.

Acknowledgments. This work has been partially supported by the computing facilities of the Extremadura Supercomputing Center (CenitS), by EU under the COST programme Action IC1305, 'Network for Sustainable Ultrascale Computing (NESUS)', and the Extremadura Government through the FEDER Funds.

References

1. Alexandrov, A., Lonescu, M.F., Schauser, K.E., Scheiman, C.: Loggp: incorporating long messages into the logp model - one step closer towards a realistic model for parallel computation. In: Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 95–105. SPAA 1995, NY, USA (1995)
2. Cameron, K.W., Ge, R., Sun, X.H.: $\log_m p$ and $\log_3 p$: accurate analytical models of point-to-point communication in distributed systems. IEEE Trans. Comput. **56**(3), 314–327 (2007)
3. Ino, F., Fujimoto, N., Hagihara, K.: Loggps: a parallel computational model for synchronization analysis. SIGPLAN Not. 36, pp. 133–142 (2001). http://doi.acm.org/10.1145/568014.379592
4. Kielmann, T., Bal, H.E., Verstoep, K.: Fast measurement of LogP parameters for message passing platforms. In: Rolim, J.D.P. (ed.) IPDPS-WS 2000. LNCS, vol. 1800, pp. 1176–1183. Springer, Heidelberg (2000)
5. Lastovetsky, A., Rychkov, V.: Accurate and efficient estimation of parameters of heterogeneous communication performance models. Int. J. High Perform. Comput. Appl. **23**(2), 123–139 (2009). http://dx.doi.org/10.1177/1094342009103947
6. Martinasso, M., Méhaut, J.-F.: Prediction of communication latency over complex network behaviors on SMP clusters. In: Bravetti, M., Kloul, L., Zavattaro, G. (eds.) EPEW/WS-EM 2005. LNCS, vol. 3670, pp. 172–186. Springer, Heidelberg (2005)
7. Moritz, C.A., Frank, M.I.: Logpc: modeling network contention in message-passing programs. IEEE Trans. Parallel Distrib. Syst. **12**(4), 404–415 (2001). http://dx.doi.org/10.1109/71.920589
8. MPI Forum: MPI: A message-passing interface standard. Version 3.0 (September 21th 2012). http://www.mpi-forum.org
9. Pješivac-Grbović, J., Angskun, T., Bosilca, G., Fagg, G.E., Gabriel, E., Dongarra, J.J.: Performance analysis of MPI collective operations. Cluster Comput. **10**, 127–143 (2007). http://dl.acm.org/citation.cfm?id=1265235.1265248
10. Rico-Gallego, J.A., Díaz-Martín, J.C.: τ-lop: modeling performance of shared memory *MPI*. Parallel Comput. **46**, 14–31 (2015)
11. Steffenel, L.: Modeling network contention effects on all-to-all operations. In: 2006 IEEE International Conference on Cluster Computing, pp. 1–10 (2006)
12. Yuan, L., Zhang, Y., Tang, Y., Rao, L., Sun, X.: Loggph: a parallel computational model with hierarchical communication awareness. In: 2010 IEEE 13th International Conference on Computational Science and Engineering (CSE), vol. 2, pp. 268–274 (2010)