Energy Estimation for MPI Broadcasting Algorithms in Large Scale HPC Systems

M.E.M. Diouri
INRIA Avalon - ENS Lyon
mehdi.diouri@ens-lyon.fr

O. Glück
INRIA Avalon - UCBL
olivier.gluck@ens-lyon.fr

L. Lefèvre
INRIA Avalon - ENS Lyon
laurent.lefevre@inria.fr

J.-C. Mignot
INRIA Avalon - CNRS
jean-christophe.mignot@ens-lyon.fr

ABSTRACT

Future supercomputers will gather hundreds of millions of communicating cores. The movement of data in such systems will be very energy consuming. We address in this paper the issue of energy consumption of data broadcasting in such large scale systems. To this end, we propose a framework to estimate the energy consumed by different MPI broadcasting algorithms for various execution settings. Validation results show that our estimations are highly accurate and allow to select the least consuming broadcasting algorithm.

Categories and Subject Descriptors

C.4 [**Computer-Communication Networks**]: (Performance of Systems)[measurement techniques, modeling techniques]

Keywords

Power and Energy consumption, MPI Broadcasting, Hybrid Programming, HPC Applications.

1. INTRODUCTION

Energy consumption by large-scale HPC systems is a recognized concern which will grow in importance as systems get larger. Indeed, the currently fastest supercomputer according to the TOP500[1] list published in November 2012, is able to perform 17 PFlop/s. Its energy efficiency is 2 GFlop/W while the Defense Advanced Research Projects Agency (DARPA) has set to 50 GFlop/W the energy efficiency of yet to come exascale systems [8].

Exascale applications will also involve large volumes of data: hundreds of exabytes of data are expected by 2018 [1]. Programmability of exascale systems is still under debate. MPI and its combination with OpenMP is recognized as a possible candidate for programming applications in exascale systems. Therefore, we choose to examine their broadcasting algorithms closely, especially as they are not optimized with energy-efficiency foremost in mind.

Currently, in order to evaluate the power consumption of a data broadcasting algorithm for any particular execution, the only approach is to pre-execute the application and monitor the energy consumption. This approach is not practical for algorithm selection since it does not allow to evaluate energy consumption before the execution. To address this problem, this paper proposes an accurate estimator of the energy consumption for data broadcasting algorithms. This estimator can be used to estimate the power consumption of a particular broadcasting algorithm for a large variety of execution configurations. It can also be used to compare data broadcasting algorithms from given execution configurations. Four broadcasting algorithms are considered in this paper: two pure MPI broadcasting and two hybrid (MPI + OpenMP) broadcasting.

This paper is organized as follows. Section 2 describes related works. The design of the energy estimation framework is detailed in Section 3. Section 4 presents validation results. Section 5 concludes the paper and presents some future works.

2. RELATED WORKS

Evaluating power consumption of one node or one process is not new and generates numerous researches and publications. In this section, we describe a small selected set of existing methods for measuring the energy consumption of computing systems. The PowerPac [5] framework allows a precise energy monitoring of one HPC node. Software systems like SoftPower [11] estimate the energy consumption by observing the usage of internal resources.

Some studies [3,7] have evaluated the performance of broadcast algorithms. Parallelization of applications with a hybrid approach uses OpenMP[2] to distribute the computing portion of each MPI process between threads [2]. Recent results [13] have shown that hybrid programming can improve performance on clusters of SMP. However, none of these studies have evaluated the power or the energy consumption of data broadcasting algorithms. In [10], we presented an energy estimator for fault tolerance protocols. In this paper, our goal is to apply an analogous approach to build a framework for estimating the energy consumption of data broadcasting algorithms.

[1]Titan machine: `http://www.top500.org/system/177975`

[2]`http://www.openmp.org`

EuroMPI '13 September 15 - 18 2013, Madrid, Spain

3. DESIGN OF THE ESTIMATION TOOL

In this section, we propose the a tool that estimates the energy consumption of four MPI broadcasting algorithms. In Section 3.1, we identify the high-level operations found in these four algorithms. In Section 3.2, we explain how we perform the calibration according to the hardware used. We describe our estimation methodology in Section 3.3.

3.1 Broadcasting algorithms and high-level operations

We consider the broadcast of V_{data} among N nodes with p processes per node. Our study focuses on four different broadcasting algorithms: two pure MPI broadcasting and two hybrid broadcasting: MPI + OpenMP. For the two MPI versions, we consider the Scatter & AllGather algorithm [14] provided in MPICH2[3] 1.3.3 and the Pipelining algorithm provided in OpenMPI[4] 1.4.4. For convenience, we choose to respectively call them MPI/SAG and MPI/Pipeline. In a hybrid broadcasting, the root process uses MPI to broadcast the data to one master MPI process per node. Then, each master MPI process uses OpenMP to share the broadcasted data with all the other threads within the same node: the clause used in OpenMP is CopyPrivate. We call Hybrid/SAG the hybrid broadcasting using MPI/SAG combined with OpenMP. Analogously, Hybrid/Pipeline combines MPI/Pipeline with OpenMP.

Figure 1 presents the four broadcasting algorithms that we consider and shows the sizes of the messages exchanged between the different processes. In these four algorithms, we identify the following high-level operations:

- *Scatter*: It consists of dividing a data into a number of smaller parts equal to the number of processes and sending a piece of data to each process using a binomial tree topology. It is used in MPI/SAG and Hybrid/SAG.

- *AllGather*: Given a set of elements distributed across all processes, *AllGather* will gather all of the elements to all the processes using a ring topology. It is used in MPI/SAG and Hybrid/SAG.

- *Pipeline*: It consists of splitting the source message into an arbitrary number of packets (called chunks) which are routed in a pipelined fashion. In Figure 1, the number of chunks is denoted by C. It is used in MPI/Pipeline and Hybrid/Pipeline.

- *CopyPrivate*: It consists of copying a data stored in a variable from one thread to the corresponding variables of all other threads within the same node. It is used in Hybrid/SAG and Hybrid/Pipeline.

Estimating the energy consumption of a given high-level operation *op* (*Scatter*, *AllGather*, *Pipeline*, or *CopyPrivate*) is really complex as it depends on a large set of parameters. These high-level operations are associated to parameters that depend not only on the context execution but also on the hardware used. Thus, in order to estimate accurately the energy consumption due to a specific implementation of a broadcasting algorithm, the estimation tool needs to take into consideration all the parameters of the execution context (size of data broadcasted, number of processes, source and destination storage medium of the data broadcasted) and all the hardware parameters (number of cores per node, memory architecture, type of disks, network topology, etc.).

In order to take into consideration all the parameters, the estimation tool integrates an automated calibration component.

[3] http://www.mcs.anl.gov/research/projects/mpich2/

[4] http://www.open-mpi.org/

3.2 Calibration approach

Energy consumption depends strongly on the hardware used in the execution platform. The calibration process consists of gathering energy knowledge of all the identified operations according to the server nodes and the network used in the supercomputer. To this end, we developed a set of simple benchmarks that extracts the power consumption ρ_{op} and the execution time t_{op} of each high-level operation encountered in the broadcasting algorithms. The goal of our calibration approach is to take into account the specific features of the hardware used in the supercomputer in order to provide realistic energy estimations. Although this knowledge base has a significant size, it needs to be done only occasionally.

In our calibrator component, the energy consumption of a node i performing a high-level operation *op* is:

$$\xi_{op}^{Node_i}(p,N) = \rho_{op}^{Node_i}(p) \cdot t_{op}(p,N)$$

Analogously, the energy consumption of a switch j during *op* is: $\xi_{op}^{Switch_j} = \rho_{op}^{Switch_j} \cdot t_{op}(p,N)$

$t_{op}(p,N)$ is the time required to perform *op* over the Np processes. Within the node i, p is the number of processes involved in *op*. $\rho_{op}^{Node_i}(p)$ is the power consumption of the node i during t_{op}. $\rho_{op}^{Switch_j}$ is the power consumption of the switch j during t_{op}. Thus, we need to get the power consumption $\rho_{op}^{Node_i}$ for each node i, $\rho_{op}^{Switch_j}$ for each switch j, and the execution time t_{op} of each operation. Therefore, our energy calibrator integrates a power calibrator described in 3.2.1 and an execution time calibrator described in 3.2.2.

3.2.1 Power consumption $\rho_{op}^{Node_i}$ and $\rho_{op}^{Switch_j}$

In our power calibrator, the power consumption of a node i during an operation *op* is: $\rho_{op}^{Node_i}(p) = \rho_{idle}^{Node_i} + \Delta\rho_{op}^{Node_i}(p)$. Analogously, the power consumption of a switch j during *op* is: $\rho_{op}^{Switch_j} = \rho_{idle}^{Switch_j} + \Delta\rho_{op}^{Switch_j}$.

$\rho_{idle}^{Node_i}$ (or $\rho_{idle}^{Switch_j}$) is the power consumption when the node i (or the switch j) is idle (i.e. switched on but running only the operating system) and $\Delta\rho_{op}^{Node_i}$ (or $\Delta\rho_{op}^{Switchi}$) is the mean extra power cost due to the high-level operation execution. We showed in [10] that $\rho_{idle}^{Node_i}$ may be different even for identical nodes. Thus, we calibrate $\rho_{idle}^{Node_i}$ and $\rho_{idle}^{Switch_j}$ by measuring the power consumption of each node and each switch while they are idle.

In order to measure $\Delta\rho_{op}^{Node_i}$ and $\Delta\rho_{op}^{Switch_j}$ experimentally, we isolate each high-level operation by instrumenting the implementation of each broadcasting algorithm that we consider, and we use OmegaWatt[5], an external power meter on each node [9]. This external power meter provides one measurement each second with a precision of 0.125W. Like in [10], we measured $\Delta\rho_{op}^{Node_i}$ (and $\Delta\rho_{op}^{Switch_j}$) and we found

[5] http://www.omegawatt.fr/gb/index.php

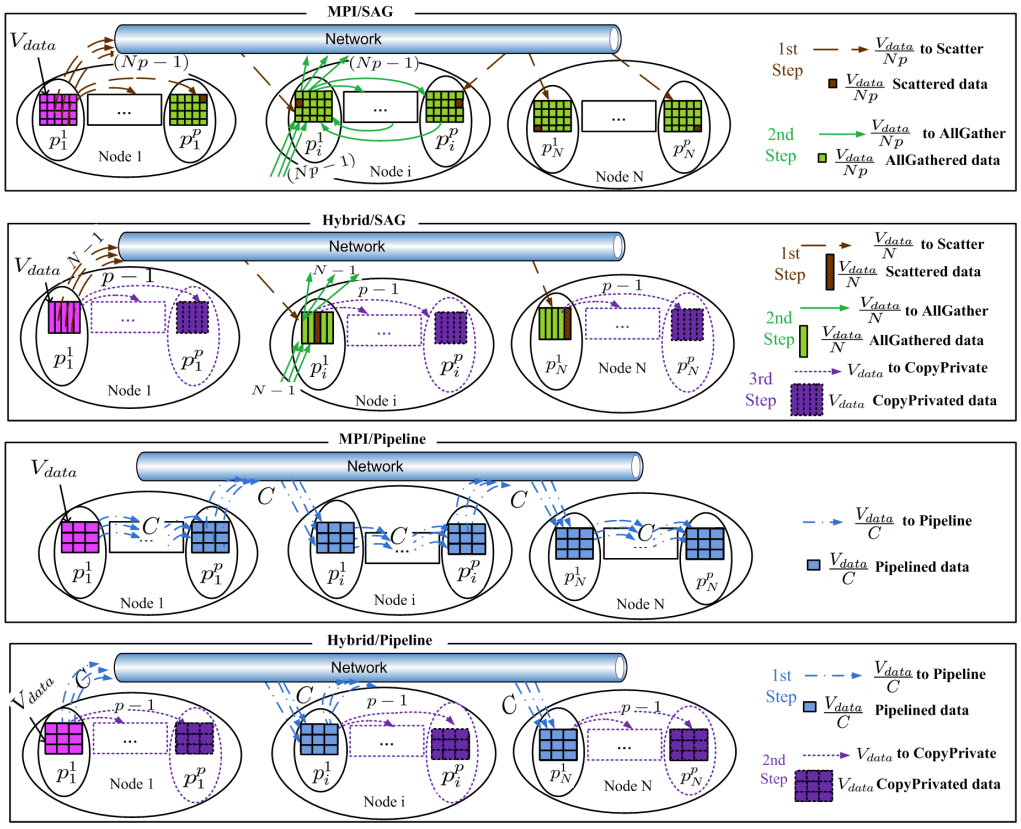

Figure 1: Broadcasting algorithms and the associated high-level operations

that for a given operation, $\Delta\rho_{op}^{Node_i}$ (and $\Delta\rho_{op}^{Switch_j}$) is the same on identical nodes (or switches). Indeed, $\Delta\rho_{op}^{Node_i}$ and $\Delta\rho_{op}^{Switch_j}$ depend only on the hardware used on the node or the switch. Thus, we measure $\Delta\rho_{op}^{Node_i}$ and $\Delta\rho_{op}^{Switch_j}$ during each high-level operation *op*, for each type of nodes/switches only.

3.2.2 *Execution time* t_{op}

In this section, we describe the execution time model that we consider for each high-level operation of each broadcasting algorithm. For each operation *op*, t_{op} depends on different parameters.

According to [3], the time required for scattering, allgathering or for pipelining a volume of data V_{data} among N nodes with p processes per node is:

$$\begin{aligned} t_{Scatter}(p,N) &= t_{AllGather}(p,N) \\ &= (T_{Snet}(p,N) + \frac{V_{data}}{R_{net}(p,N)}) \cdot \frac{Np-1}{Np} \\ t_{Pipeline}(p,N) &= (T_{Snet}(p,N) + \frac{V_{data}}{R_{net}(p,N)}) \cdot \frac{C+Np-2}{C} \end{aligned}$$

$T_{Snet}(p,N)$ is the time to start up the network link and $R_{net}(p,N)$ is the transfer rate for transmitting the volume of data V_{data}. C is the number of (equal-sized) pieces into which the data is split by the MPI/Pipeline. It is equal to the total volume of data divided by the size of each piece of data. Thus, the size of each piece of data and consequently C depend on the chosen implementation of pipelining.

For a fixed number of nodes N and a fixed number of processes per node p, $t_{Scatter}$, $t_{AllGather}$ or $t_{Pipeline}$ are linear functions of V_{data}. Therefore, to calibrate t_{op}, the estimation tool automatically runs a simple benchmark that measures t_{op} for different values of N, p and V_{data}.

The time required to copy a data to the threads of the same node is a function of the number of threads p per node and the volume of data to copy V_{data}:

$$t_{CopyPrivate}^{Node_i}(p) = T_{access}^{Node_i}(p) + \frac{V_{data}}{R_{transfer}^{Node_i}(p)}$$

$T_{access}^{Node_i}$ is the time needed to access the RAM where V_{data} has been *AllGathered* or *Pipelined*. $R_{transfer}^{Node_i}$ is the transmission rate for copying a data into the RAM.

3.3 Estimation methodology

We described previously how we perform the calibration process in the estimator. This calibration needs to be done by the administrator each time that a change occurs in the hardware used in the supercomputer. Once this calibration is completed, our energy estimator framework is able to provide estimations of the energy consumption due to a data broadcasting.

Figure 2 shows the components of the estimator and their interactions. The user provides some information related to the execution context in which the data broadcasting will occur. This information is taken as an input by the estimator component. As an output, the calibrator component provides the calibration data on which the estimation tool relies on to estimate the energy consumption of data broadcasting algorithms. In the following subsections, we detail

for each algorithm, the information collected from the user and the corresponding calibration data transmitted to our estimator component. We also detail how the energy consumption is computed thanks to this calibration output and to the information provided by the user.

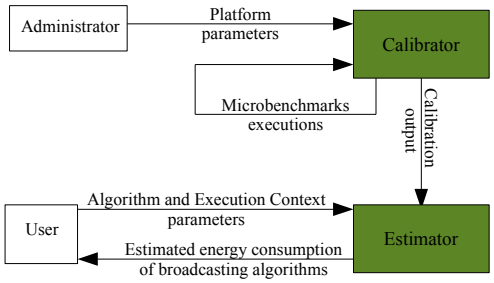

Figure 2: Design of the estimator

3.3.1 MPI/SAG and Hybrid/SAG

To estimate the energy consumption of MPI/SAG and Hybrid/SAG, the estimator component collects from the user the number of nodes N, the number of processes per node p, the number of switches M and the size of data to broadcast V_{data}. From this information, the estimator collects from the calibrator $t_{Scatter}(p,N)$, $t_{AllGather}(p,N)$ according to p, N and V_{data}. Similarly, for each node i, it collects $t_{CopyPrivate}(p)$ according to p and V_{data}.

If V_{data} is not a size recorded by the calibrator, the estimator computes $t_{op}(p,N)$ by adjusting the equation of $t_{op}(p,N)$ using the least squares method.

The estimated energy consumption of MPI/SAG is:

$$E_{MPI/SAG} = \sum_{i=1}^{N} \xi_{MPI/SAG}^{Node_i} + \sum_{j=1}^{M} \xi_{MPI/SAG}^{Switch_j}$$
$$= t_{Scatter}(p,N) \cdot (\sum_{i=1}^{N} \rho_{Scatter}^{Node_i}(p) + \sum_{j=1}^{M} \rho_{Scatter}^{Switch_j})$$
$$+ t_{AllGather}(p,N) \cdot (\sum_{i=1}^{N} \rho_{AllGather}^{Node_i}(p) + \sum_{j=1}^{M} \rho_{AllGather}^{Switch_j})$$

The estimated energy consumption of Hybrid/SAG is:

$$E_{Hybrid/SAG} = \sum_{i=1}^{N} \xi_{Hybrid/SAG}^{Node_i} + \sum_{j=1}^{M} \xi_{Hybrid/SAG}^{Switch_j}$$
$$= t_{Scatter}(1,N) \cdot (\sum_{i=1}^{N} \rho_{Scatter}^{Node_i}(1) + \sum_{j=1}^{M} \rho_{Scatter}^{Switch_j})$$
$$+ t_{AllGather}(1,N) \cdot (\sum_{i=1}^{N} \rho_{AllGather}^{Node_i}(1) + \sum_{j=1}^{M} \rho_{AllGather}^{Switch_j})$$
$$+ \sum_{i=1}^{N} (t_{CopyPrivate}^{Node_i}(p) \cdot \rho_{CopyPrivate}^{Node_i}(p))$$

where $\rho_{op}^{Node_i}(p)$ and $\rho_{op}^{Switch_j}$ are outputs of the calibrator, and $t_{op}(p,N)$ is computed by the estimator.

3.3.2 MPI/Pipeline and Hybrid/Pipeline

Compared to the estimation of MPI/SAG and Hybrid/SAG, the estimator component collects from the user an additional parameter that is the size of each data piece, in order to estimate the energy consumption of MPI/Pipeline and Hybrid/Pipeline. From the calibrator, it collects $t_{Pipeline}(p,N)$ according to p, N and V_{data}. The estimated energy consumption for MPI/Pipeline (respectively for Hybrid/Pipeline) is analogous to the estimated energy for MPI/SAG (respectively for Hybrid/SAG). The only difference is that the *Scatter* and *AllGather* operations are replaced by an unique operation: the *Pipeline* operation.

4. VALIDATION OF THE ESTIMATOR

To apply the energy estimation approach, we calibrate and run HPC applications on a homogeneous cluster of the large scale experimental platform Grid5000, a French scientific platform designed to support experiment-driven research in large scale parallel and distributed systems [4]. We run the different data broadcasting algorithms and compare the measured energy consumption to the one we estimate thanks to the energy estimating framework.

4.1 Experimental infrastructure

We use a cluster of 16 identical nodes Dell R720 (2 Intel Xeon CPU 2.3 GHz, with 6 cores each; 32 GB of memory; a 10 Gigabit Ethernet NIC; a SCSI hard disk of 598 GB). The 16 nodes are interconnected with one 10 Gigabit Ethernet switch. We monitor this cluster with an energy-sensing infrastructure of external Omegawatt wattmeters. This energy-sensing infrastructure, which was also used in [10], enables to get the instantaneous consumption in Watts, at each second for each monitored node. We ran each experiment 50 times and computed the mean value over the 50 values.

4.2 Calibration results for our platform

In this section, we calibrate our platform according to the process described in 3.2. Since each node has 12 cores, we consider different number of processes per node.

4.2.1 Power calibration

First, we measure the idle power consumption $\rho_{idle}^{Node_i}$ of each node i. Then, we calibrate the extra power consumption $\Delta\rho_{op}^{Node_i}$ of all the high-level operations found in the broadcasting algorithms (Figure 3). As each node has 12 cores each, we calibrated the extra power cost for 1,4,8, and 12 processes per node involved in a given operation.

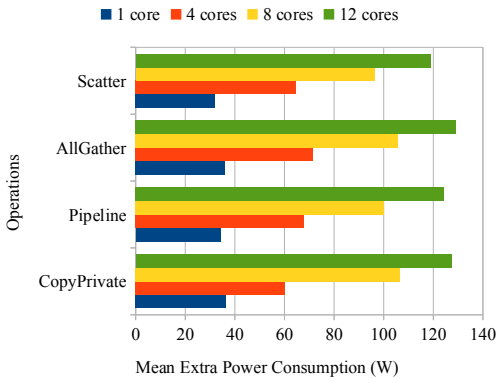

Figure 3: Extra power cost of high-level operations

We notice in Figure 3 that the mean extra power consumption varies with the number of cores per node that perform the same operation. Contrary to what one may think, the mean extra power consumption of these operations does not increase linearly with a growing number of cores per node.

Besides, we measure the idle power consumption of our 10 Gigabit Ethernet switch during 300 seconds and its power consumption during the peak network traffic during 300 seconds. The power consumption of the switch remained almost constant during the whole experiment. In other words, the

power consumption of the switch is not varying according to the network traffic. Some recent studies [6, 12] confirmed this fact by evaluating and showing the power consumption of different network equipments is not influenced by the network traffic.

4.2.2 *Execution time calibration*

To calibrate the execution time, the estimator automatically runs the corresponding benchmark in order to measure t_{op} for the different operations: $t_{Scatter}(p, N)$, $t_{AllGather}(p, N)$, $t_{Pipeline}(p, N)$ and $t_{CopyPrivate}(p)$.

For different number of nodes N and different number of processes per node p, we measure $t_{Scatter}(N, p)$, $t_{AllGather}(N, p)$ and $t_{Pipeline}(N, p)$ with a varying volume of data. We present part of the calibration results for these operations in Figure 4. for a volume of data ranging from 200 MBytes to 2000 MBytes, for 1 process per node, and for a varying number of nodes (4, 8, 12, 16). Figure 4 shows that $t_{Scatter}$, $t_{AllGather}$ and $t_{Pipeline}$ evolve in a linear fashion for a fixed N and p. This confirms the linear approach for estimating $t_{Scatter}$, $t_{AllGather}$ and $t_{Pipeline}$. In other words, $T_{Snet}(p, N)$ and $R_{net}(p, N)$ are constant when we consider a fixed number of nodes and a fixed number of processes per node. With N and p fixed, we can deduce $T_{Snet}(p, N)$ and $R_{net}(p, N)$ and then estimate $t_{Scatter}(p, N)$, $t_{AllGather}$ and $t_{Pipeline}(p, N)$.

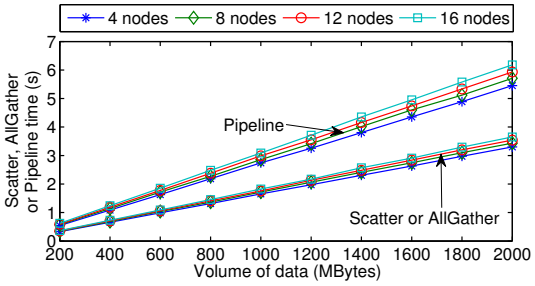

Figure 4: Calibration of $t_{Scatter}(N, 1)$, $t_{AllGather}(N, 1)$ and $t_{Pipeline}(N, 1)$

We present the calibration results for $t^{Node_i}_{CopyPrivate}(p)$ in Figure 5. As shown in [10], $t^{Node_i}_{CopyPrivate}(p)$ may vary from one node to another even if we consider identical nodes. That is why we calibrate them for each node of our cluster. For different number of processes (4, 8, 12) within a same node, we measure $t^{Node_i}_{CopyPrivate}(p)$ with a varying volume of data in a similar fashion as we already did for logging in RAM [10].

Figure 5 confirms the linear approach for estimating $t^{Node_i}_{CopyPrivate}(p)$. Thus, we need to take it into account when considering several processes per node. In addition, for a fixed volume of data, we notice that $t^{Node_i}_{CopyPrivate}(p)$ almost remains unchanged when considering a growing number of processes per node. This is due to the fact that the shared data is simultaneously logged in the memory of the different processes without contention.

4.3 Evaluation of the estimation accuracy

In this section, we want to compare the estimated energy consumption computed by the estimation tool once the calibration done (but before running the application) to the real energy consumption measured by our energy sensors

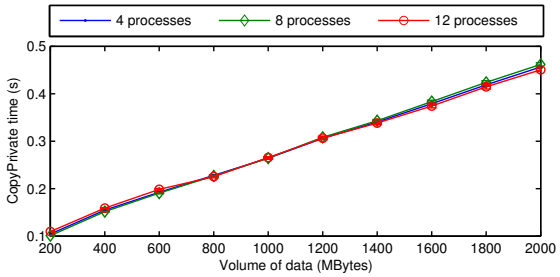

Figure 5: Calibration of $t^{Node_i}_{CopyPrivate}(p)$

during the application execution. We consider 4 classes of broadcasting applications running over different execution contexts as follows:

Name	Number of messages	Size of each message	Number of nodes	Processes per node
A	2000	1 MBytes	14	8
B	80	25 MBytes	16	1
C	4	500 MBytes	10	4
D	1	1.75 GBytes	6	12

For each application (A, B, C, D) and or each broadcasting algorithm (MPI/SAG, MPI/Pipeline, Hybrid/SAG, Hybrid/Pipeline), on the one hand, we estimate the energy consumption by summing the estimated energy of the operations involved in the given algorithm. On the other hand, we measure the total energy consumption. In our experiments, pipelining is performed with a fixed chunk size of 128 KB. Each energy measurement is done 30 times and we compute the average value.

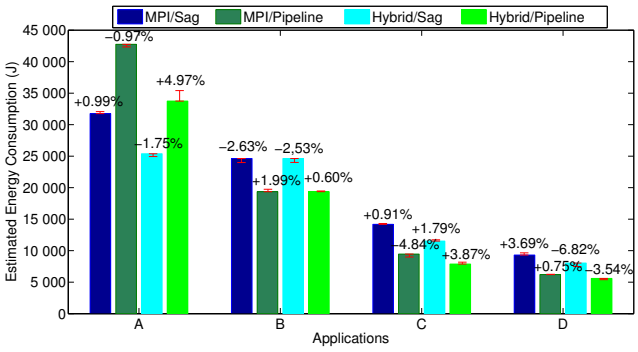

Figure 6: Estimated energy consumption (in J) of the four broadcasting algorithms compared to real energy measurements

In Figure 6, we plot the estimated energy consumption computed by our estimator for each broadcasting algorithm and for each application. We also represented the relative deviation (in percentage) between the estimated and the measured energy consumption. Figure 6 shows that the energy estimations are accurate: the relative differences between the estimated and the measured energy consumptions are low. The worst estimation is -6.82 % for Hybrid/SAG algorithm with application D.

Figure 6 shows that from one application to another the less energy consuming algorithm is not always the same. First, we notice that the hybrid algorithms are less energy consuming than the MPI algorithms, especially when considering many processes per node (applications A, C and D). In general, determining the less consuming algorithm depends on the trade-off between the volume of broadcasted data and the number of nodes involved. For A, the less energy consuming algorithm is Hybrid/SAG since the number of pipelined chunks is relatively low as the volume of data to broadcast is small. Oppositely, the less energy consuming broadcasting algorithm for B, C and D is Hybrid/Pipeline since the number of pipelined chunks starts to be high enough compared to the number of processes involved. Thus, by providing such energy estimations before executing an application, we can select the best data broadcasting algorithm in terms of energy consumption.

One may think that energy consumption of an algorithm is completely linked to its execution time. Our study shows that it is not true. Indeed, although Figure 6 shows that the energy consumption of hybrid algorithms are lower to the energy consumption of MPI algorithms in applications A, C and D, the corresponding estimated execution time of the hybrid algorithms are slightly higher to the execution time of the MPI algorithms. Indeed, the power consumption of hybrid algorithms during MPI operations (*Scatter*, *AllGather* and *Pipeline*) is much lower than the one of MPI broadcasting algorithms (see Figure 3). This is because in hybrid broadcasting, only one process is active while in pure MPI broadcasting algorithms, all the processes are active during MPI operations.

5. CONCLUSIONS AND FUTURE WORKS

This paper presents a tool that estimates the energy consumption of four broadcasting algorithms: MPI/Pipeline, MPI/SAG, Hybrid/Pipeline and Hybrid/SAG. To provide accurate estimations, this tool relies on an energy calibration of the execution platform and a description of the execution settings. Thanks to our approach based on a calibration process, this framework can be used in any energy monitored supercomputer. We showed in this paper that the energy estimations provided by the estimation tool are accurate: the relative difference between energy measurements and estimations do not exceed 6.82% for the four considered applications. By estimating the energy consumption of broadcasting algorithms, this tool allows selecting the best broadcasting algorithm in terms of energy consumption before running the application. As a future work, we will propose energy efficient improvements for broadcasting algorithms.

6. ACKNOWLEDGMENTS

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see `https://www.grid5000.fr`)

7. REFERENCES

[1] G. Aloisio and S. Fiore. Towards Exascale Distributed Data Management. *IJHPCA*, 23(4):398–400, 2009.

[2] F. Cappello and D. Etiemble. MPI versus MPI+OpenMP on IBM SP for the NAS benchmarks. In *ACM/IEEE SC'00, Dallas, Texas, USA, November 2000*.

[3] Daniel M. Wadsworth and Zizhong Chen. Performance of MPI broadcast algorithms. In *IEEE IPDPS 2008, Miami, Florida USA, April 14-18, 2008*, pages 1–7.

[4] Franck Cappello et al. Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In *IEEE/ACM GRID 2005, November 13-14, 2005, Seattle, Washington, USA*, pages 99–106, 2005.

[5] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel and Distributed Systems*, 99:658–671, 2009.

[6] H. Hlavacs, G. Da Costa, and J.-M. Pierson. Energy consumption of residential and professional switches. In *IEEE CSE*, 2009.

[7] Jelena Pjesivac-Grbovic, Thara Angskun, George Bosilca, Graham E. Fagg, Edgar Gabriel and Jack Dongarra. Performance Analysis of MPI Collective Operations. In *IEEE IPDPS 2005, 4-8 April 2005, Denver, CO, USA*, 2005.

[8] P. M. Kogge and et al. ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems. In *DARPA Information Processing Techniques Office*, page pp. 278, Washington, DC, September 28 2008.

[9] M. E. M. Diouri, M. F. Dolz, O. Glück, L. Lefèvre, P. Alonso, S. Catalán, R. Mayo, and E. S. Quintana-Ortí. Solving some mysteries in power monitoring of servers: Take care of your wattmeters! In *EE-LSDS 2013, Vienna, Austria, April, 22-24 2013*.

[10] M.E.M. Diouri, O. Glück and L. Lefèvre and F. Cappello. ECOFIT: A Framework to Estimate Energy Consumption of Fault Tolerance protocols during HPC executions. In *IEEE/ACM CCGrid 2013, Delft, Netherlands, May 13-16, 2013*.

[11] Min Yeol Lim, Allan Porterfield and Robert Fowler. SoftPower: Fine-Grain Power Estimations Using Performance Counters. In *ACM HPDC*, July 2010.

[12] Priya Mahadevan, Puneet Sharma, Sujata Banerjee and Parthasarathy Ranganathan. A power benchmarking framework for network devices. In *NETWORKING 2009 Conference, Aachen, Germany, May 11-15, 2009.*, pages 795–808, 2009.

[13] R. Rabenseifner, G. Hager, and G. Jost. Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In *17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, February 2009*, pages 427 –436.

[14] R. Thakur and W. Gropp. Improving the Performance of Collective Operations in MPICH. In *European PVM/MPI Users' Group Meeting, Venice, Italy, September 29 - October 2, 2003*, pages 257–267, 2003.