IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 13, 2019, accepted May 14, 2019, date of publication May 31, 2019, date of current version June 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919970

Prediction of Energy Consumption by Checkpoint/Restart in HPC

M. MORÁN[1], J. BALLADINI[1], D. REXACHS[2], AND E. LUQUE[2]
[1]Facultad de Informática, Universidad Nacional del Comahue, Neuquén 8300, Argentina
[2]Departamento de Arquitectura de Computadores y Sistemas Operativos, Universitat Autònoma de Barcelona, Campus UAB, 08193 Barcelona, Spain

Corresponding author: M. Morán (marina@fi.uncoma.edu.ar)

This work was supported in part by the Agencia Estatal de Investigación (AEI), Spain, in part by the Fondo Europeo de Desarrollo Regional (FEDER) UE under Contract TIN2017-84875-P, and in part by a research collaboration agreement with the Fundacion Escuelas Universitarias Gimbernat (EUG).

ABSTRACT The fault tolerance method most used today in high-performance computing (HPC) is coordinated checkpointing. This, like any other fault tolerance method, adds additional energy consumption to that of the execution of the application. Currently, knowing and minimizing this energy consumption is a challenge. The objective of this paper is to propose a model to estimate the energy consumption of checkpoint and restart operations and a method for its construction. These estimates allow the evaluation of different scenarios in order to minimize energy consumption. We focus on coordinated checkpoint/restart at the system level, in single-program multiple-data (SPMD) applications, on homogeneous clusters. We study the behavior of the power dissipated by the compute node during a checkpoint/restart operation, as well as its execution time, considering different parameters of the system and the application. The experimentation carried out on two platforms shows the validity of the proposal. We also evaluate the impact on power and energy consumption of the processor's C states, the configuration of the network file system (NFS), where the checkpoint files are stored, and the compression of the checkpoint files. This paper contributes to the objective of predicting energy consumption in the execution of applications that use checkpoint/restart. Not counting the outliers, we can estimate the energy consumed by checkpoint/restart operations with errors lower than 7.5%.

INDEX TERMS Checkpointing, energy consumption, fault tolerance, high performance computing.

I. INTRODUCTION

Today's High Performance Computing (HPC) systems use hundreds to billions of processing units and the tendency is to continue increasing their computing power. However, this growth in computing power leads to an increase in energy consumption. Given the limitations that exist to supply energy to these types of computers, it is necessary to know the behavior of their energy consumption to find ways to decrease it. Specifically, there is a limit defined as a guide for exascale of the 20 MW [1].

The fault tolerance method most used today in HPC is coordinated checkpoint. With this method, the processes involved in the application are stopped in order to safeguard the current execution status in stable storage [2]. These methods add additional energy consumption to the execution of the application [3]. We ask ourselves, what energy saving opportunities does fault tolerance present? Is it possible to reduce the energy consumption of the execution of an application by making its fault tolerance method more energy efficient?

In this article, we propose a model and a method that allows us to predict the energy consumption of checkpoint and restart (CR) operations, considering different parameters of the system and the application. We focus on coordinated CR at the system level, in Single Program Multiple Data (SPMD) applications, on homogeneous clusters. The model allows us to answer questions such as, how does the compression of checkpoint files affect energy consumption? Compression is an operation that demands CPU and therefore dissipates more power, but storing compressed files requires less transmission time through the network and less storage space. This model can be used to consider which parameters of the system and the application to use while protecting an application, in order to decrease its total energy consumption.

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaowen Chu.

The contributions of this article are:

- A method for constructing energy consumption prediction models, for coordinated CR at the system level, in SPMD applications, on homogeneous clusters.
- A prediction model of energy consumption of the CR for a heat propagation application that follows the parallel programming paradigm SPMD. The results show that the model is able to predict the energy consumption of CR with an error of 7.5% without considering outliers.
- A study of the system (hardware and software) and application factors that impact the energy consumption produced by the checkpoint and restart operations.

The rest of the article is organized as follows. Section II introduces concepts about CR and DMTCP, which is the CR tool we use, as well as some definitions about power and some ways to administer it. Section III brings together some articles on topics related to CR and energy consumption. Section IV presents the theoretical model in order to know the energy consumption. In section V we see how to obtain the formulas that implement the model. We explain the results of the experimentation in Section VI and finally in Section VII we analyze some factors of the system and the application that influence the energy consumption of CR operations. Section VIII has the conclusions.

II. BACKGROUND

In this section, we introduce some concepts about coordinated CR and DMTCP, which is the CR tool used, and we give some definitions related to the power and the ways to administer it.

A. CHECKPOINT/RESTART

Currently, the fault tolerance method most widely used in large computer systems is coordinated CR. With this method, the consistent global state is guaranteed by synchronizing all the processes at the time of the checkpoint. Faced with the failure of a process, all processes must restart from the last checkpoint. The wide use of this method is due to its simple implementation, since the synchronization points required to store the state of the processes are present in most parallel MPI applications.

The recovery of failures through checkpoints consists mainly of restarting the application replacing the component that has failed, but to avoid a complete re-execution of the application, the state of execution is periodically saved in stable, local or remote, storage. When a fail-stop failure is detected, the application is restarted from the last stored execution state [2]. In the case of parallel applications, we can define a consistent global checkpoint as the set of local checkpoints, one for each process, which form a consistent global state. Any consistent global checkpoint can be used to restart the execution of an application after a failure [4].

Checkpoints have to be taken periodically. The time interval between checkpoints depends on the duration of the checkpoint and on the expected failure rate in the system. There are proposals on how to estimate the optimal time interval between checkpoints [5]–[7]. In this work we will evaluate the energy behavior of individual checkpoints and restarts, operations that are not affected by the checkpoint interval.

B. DMTCP

With DMTCP (Distributed MultiThreaded Checkpointing) [8], it is possible to make checkpoints on an application in a transparent way. DMTCP runs at the user level, which does not require privileged access to the system to operate it. In the case of parallel applications, it is possible to restart the processes in nodes which are different from those used at the time of the checkpoint, and a different number of processes per node or a different number of nodes can be restarted. DMTCP provides a coordinator, which must be permanently running in any node of the cluster. From this coordinator it is possible to perform a checkpoint, abort the application, among other functions. In our experiments, we inject the fault by killing the processes from the coordinator. The checkpoint image of each process is written to a file with a unique name, in a directory indicated by the user.

By using the gzip program, DMTCP is able to compress the process state to require less disk storage space and reduce the amount of data transmitted over the network (between the compute node and the storage node). It is possible to indicate if we want to use compression or not by using an environment variable, or with an option of the dmtcp_launch command. The dmtcp_launch command is the command used to execute an application. The first call to dmtcp_launch will launch the coordinator, if it did not already exist. For example, to launch an MPI application with 8 processes and without using compression, we can use:

```
$dmtcp_launch -no-gzip mpirun -np 8 ./pr
```

To perform a checkpoint manually from the command line we use:

```
$dmtcp_command c
```

To restart the application after a failure, the script generated by DMTPC is used as follows:

```
$sh dmtcp_restart_script.sh
```

As we will see later, we are interested in knowing the average power dissipated and the time of each CR operation. These values depend on several factors, including the system architecture, the file system, the application, and the fault tolerance system used. In particular, from the DMTCP's point of view, the variables that the user can modify and that affect power and time are:

- Application problem size: The totality of the memory assigned to the user process must be safeguarded.
- Whether compression is used or not: A compressed file takes less time to store, and compression dissipates more power.

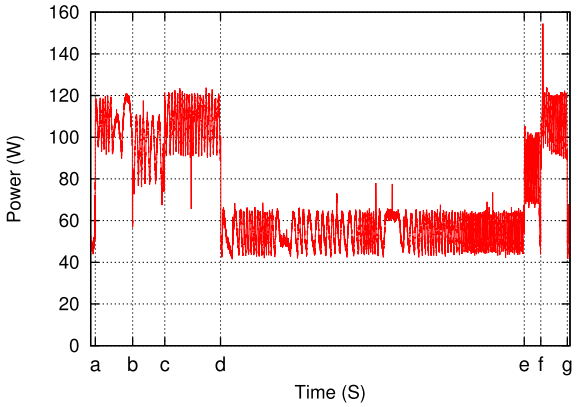

FIGURE 1. Real power dissipated during checkpoint, restart, application, and idle time.

- Whether incremental checkpoint is used or not: An incremental checkpoint has a smaller size than a full one.

C. POWER DEFINITIONS

In this subsection we define some power related terms used throughout the article.

- Instantaneous power: This is the power absorbed by an element at any instant of time, and is given by the product of instantaneous voltage (V) and instantaneous current (I), which pass through that element, as shown in 1. In this work, this power is obtained from the voltage and current measurements delivered by the oscilloscope.

$$P[Watts] = I[Amperes] \times [Volts] \tag{1}$$

- Active or real power: This is the average of the instantaneous power over an alternating current (AC) complete cycle. In AC, a sine wave represents the value of voltage and current through time. In countries where the frequency is 50Hz, the change of direction occurs 50 times per second, giving a duration of the complete cycle of 20 milliseconds.
- Average power: This is the average of the instantaneous power in a given period of time. In our case, we will average the real power for the duration of the checkpoint or restart. From now on, when we mention power or dissipated power, we mean average dissipated power.
- Base power: The power dissipated by the node when the processor is idle.
- Dynamic power: This is the power dissipated without considering the base power.

Fig. 1 shows the graph of the real power, obtained with the measurements delivered by the oscilloscope, when an application is executed and a fault is injected (to see how the fault is injected, refer to subsection II-B). The execution of the application starts at time 'a'. At time 'b' the checkpoint begins, and it finishes at time 'c'. The fault is introduced at time 'd', and a period of inactivity begins there (we can see that the base power is around 50 W). At time 'e', the restart is launched. Finally, at the end of the restart, the execution of the application continues, at time 'f'.

D. POWER ADMINISTRATION: P AND C STATES

The ACPI specification (Advanced Configuration and Power Interface) provides an open standard that allows the operating system to manage the power of the devices and the computing system.[1] It allows us to manage the energy behavior of the processor, the component that consumes the most energy in a computer system. ACPI defines Processor Power States (Cx states), where C0 is the execution state, and C1…Cx are inactive states.

A processor that is in the C0 state will also be in a Performance State (Px states). The P0 state means an execution at the maximum capacity of performance and power demand. As the number of P state increases, its performance and demanded power is reduced. The processors implement the P states using the technique of Dynamic Frequency and Voltage Scaling (DVFS) [9]. Reducing the voltage supply reduces the energy consumption. However, the delay of the logic gates increases, so it is necessary to reduce the clock frequency of the CPU in order for the circuit to work correctly. In certain multicore processors, each core is allowed to be in a different P state.

When there are no instructions to execute, the processor can be set in a state C greater than 0 to save energy. There are different levels of C states, where each of the levels could turn off certain clocks, reduce certain voltages supplied to idle components, turn off the cache memory, etc. The higher the number of the C state, the lower the power demanded, but a high latency is required to return to state C0 (execution state). In certain multicore processors, each core is allowed to be in a different C state.[2]

III. RELATED WORK

Some studies evaluate the energy behavior of the coordinated and uncoordinated CR with message logs. In [10] they compare the power dissipated by saving the message logs in RAM and in HDD. They measure the consumption of atomic operations found in the coordinated and uncoordinated protocols. In [11] they also evaluate the parallel recovery and propose an analytic model to predict the energy behavior at exascale. They perform experiments on a cluster to measure the consumption and then compare with the energy estimated with the model. They show how parallel recovery can reduce the execution time and energy consumed in a faulty scenario. Parallel recovery is based on task migration, so they use Charm++ instead of pure MPI. In this work we measure the complete checkpoint operations on MPI applications.

There are some other studies that do not take power measures. In [12] an analytic model is used to compare the

[1] http://www.acpi.info

[2] https://software.intel.com/en-us/articles/power-management-states-p-states-c-states-and-package-c-states

execution time and the consumed energy of the replication and the coordinated CR. References [13] and [14] use analytic models to estimate the optimal interval of a multilevel checkpoint in terms of energy consumption. They do not measure dissipated power but use values from other publications. An optimization problem is presented in [15], finding the clock frequency that minimizes the power consumption of an application that uses shadow processes as a fault tolerance method. Each main process has its corresponding shadow process, which executes the same program as the main one, but at a slower clock frequency. When a main process fails, the shadow replaces it, and the frequency at which it should continue executing the new main process is calculated in order to meet a stipulated maximum time limit. It is designed primarily for cloud, and it performs all the analysis in an analytic manner.

A framework to estimate the energy consumption of coordinated, uncoordinated and hierarchical checkpoints is considered in [16]. They measure the power and the execution time of the high level operations involved in the checkpoint, varying the number of cores involved. They do not use different processor frequencies, nor do they indicate whether the checkpoint is compressed or not.

References [17], [18] and [19] all present a framework for CR energy saving. In [17], many small I/O operations are replaced by a few large ones, executing in a single core, to make the checkpoint and restart more energy efficient. They use RAPL to measure and limit energy consumption. Reference [18] proposes having a core to execute a replica of all the processes of the node in order to avoid re-execution from the last checkpoint and analytically compare the energy consumption of this proposal with the traditional checkpoint. Another paper that focuses on I/O is [19]. Here, they first characterize different storage media by executing a microbenchmark that reads and writes large files, and then they design a runtime that allows modifying the clock frequency and the number of processes that perform the I/O to optimize energy consumption according to the type of storage used. In our case, we evaluate the energy behavior of a specific checkpoint library.

Another study analyzing the impact of dynamic scaling of frequency and voltage on the energy consumption of checkpoint operations is [20]. They measure the power at the component level while writing local and remote checkpoint files. They also compare two remote storage media: NFS using the kernel network stack and NFS using the IB RDMA interface. They do not include the restart in the measurements.

In [21] they evaluate the energy consumption of an application that uses compressed checkpoints. They show that using compression requires more power but saves time, so that the complete execution of the application with all its checkpoints can benefit in terms of energy. They do not modify the frequency of the processor. Like us, they measure the power dissipated by the entire checkpoint operation (and not subtasks or suboperations).

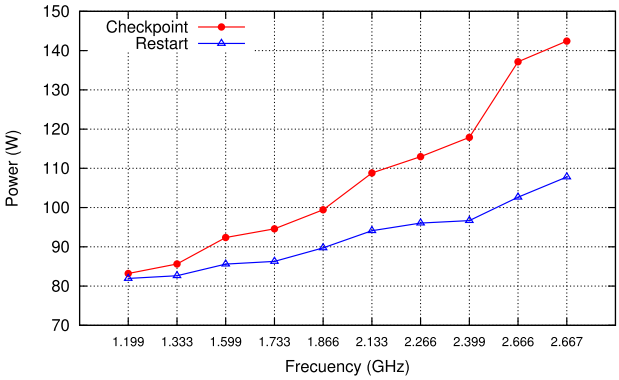

FIGURE 2. Power dissipated during checkpoint and restart for different CPU frequencies.

Our work focuses on the prediction of the energy consumption of the system level coordinated CR, for an SPMD application. The dissipated power values of the checkpoint and restart operations are measurements obtained with an external meter that measures the complete node, including the source. We evaluate the impact that the CPU clock frequency, the problem sizes, the processor C states, the compression of the checkpoint files and the NFS configuration all have on the energy consumption of the CR.

IV. ENERGY MODEL

Energy can be calculated as the product between power and time, as in 2.

$$E[Jules] = P[w] \times T[s] \tag{2}$$

We need to know the average power dissipated and the time of each operation, checkpoint or restart. The power, $P[w]$, is obtained by averaging the instantaneous power (section II-C) corresponding to the execution period of checkpoint or restart. The time, $T[s]$, is obtained from DMTCP (section II-B).

Now, what variables affect power? And what variables affect time? Next, we analyze these variables to determine which ones are significant for our model.

A. CLOCK FREQUENCY

The CPU clock frequency affects the voltage supplied, and therefore, affects the dynamic power [22]. If the checkpoint files are compressed, compression is a CPU intense operation. In Fig. 2 we observe that the higher the clock frequency, the higher the power dissipated.

With regards to time, in Fig. 3, we observe how the clock frequency especially affects the checkpoint time, whereas the restart time is much less affected.

B. PROBLEM SIZE

Problem size refers to the size of the memory assigned to the user process. As we can see in Fig. 4, problem size has little impact on the dissipated power during a CR operation. On the other hand, as the system level checkpoint has to save the total memory assigned to the user process (section II-A),

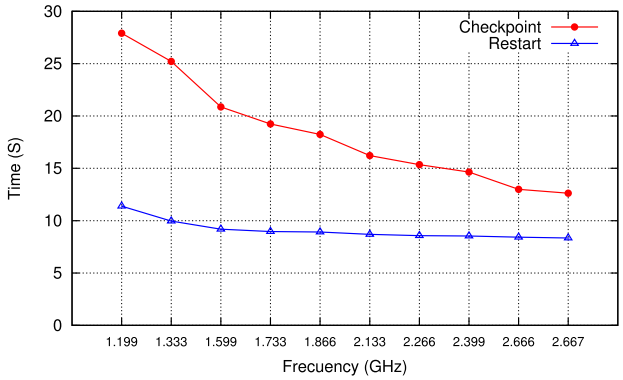

FIGURE 3. Time consumed by checkpoint and restart for different CPU frequencies.

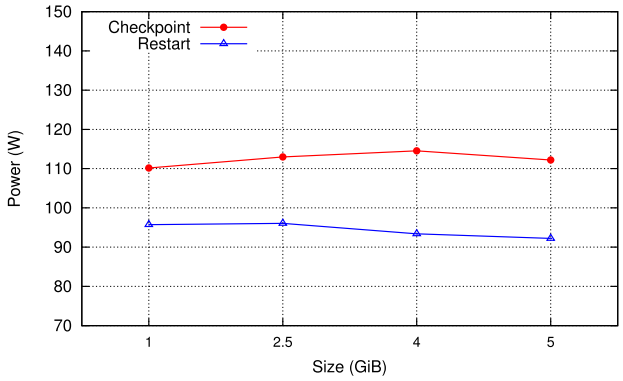

FIGURE 4. Power dissipated during checkpoint and restart for different problem sizes.

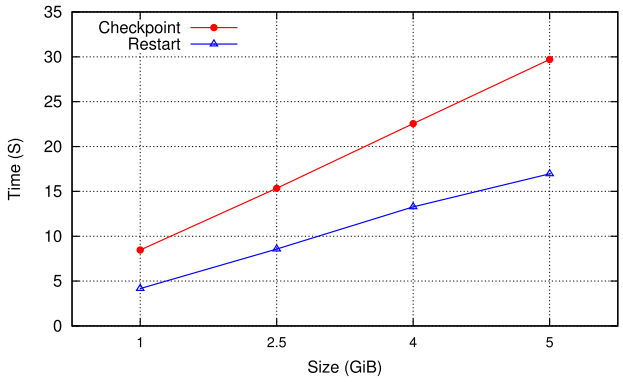

FIGURE 5. Time consumed by checkpoint and restart for different problem sizes.

the time of a CR is affected by the problem size, as shown in Fig. 5.

C. NUMBER OF CORES

The number of cores used to execute the CR affects both power and time. However, as we use the computation node as a minimum unit, we did not include this variable in this study.

In summary, we will evaluate the behavior of dissipated power and time varying on the one hand, the frequency of the processor, and on the other hand, the size of the problem.

In this way, we obtain the following general equations of the energy consumed per node, during checkpoint, E_C, and during restart, E_R:

$$E_C = P_C(f) \times T_C(f, ps) \quad (3)$$

$$E_R = P_R(f) \times T_R(f, ps) \quad (4)$$

where f is the clock frequency, ps is the problem size per node, P_C and P_R indicate the average dissipated power during a checkpoint and restart operation respectively, and T_C and T_R indicate the checkpoint and restart time respectively.

V. ENERGY CONSUMPTION ESTIMATION

In this section we show how to build the model to estimate power and time. This construction consists of two steps: collecting data through the characterization of the system, and regression analysis. When characterizing the system, we want to know how it behaves with respect to power and time while performing checkpoint and restart. Some considerations on this characterization are presented in the following subsection. The regression analysis allows us to obtain the functions to build the model and to make the estimations. Subsection V-B shows the regression analysis performed on the data obtained from the experimentation on Platform 1.

A. SYSTEM CHARACTERIZATION

We want to be able to characterize the system with the lowest possible cost, that is, with a small set of tests. The following questions arise; what to measure and how to measure it? Measuring the power of the cluster nodes requires the connection of external instruments, and accesses to the cluster that can be expensive. The number of repetitions of the experiment should be considered according to the variability of the measurement instruments.

In the following subsections, we describe some considerations about the duration of the preheating and the frequencies, problem sizes and application to be used in the characterization.

1) PREHEATING

Before starting to measure the dissipated power, the processor must be preheated, that is, the application must be left running until it reaches the maximum levels of power dissipated. This period can be defined empirically, observing when the dissipated power curve reaches the maximum values for the application.

2) CLOCK FREQUENCIES SELECTION

The CPU frequencies for the experiments have to be defined. Because the range of clock frequencies is discrete and limited, if within the selected frequencies the minimum and maximum frequencies are included, the power function obtained, P_{CR}, will show the behavior of the remaining frequencies. In the case that there is no curve that adequately represents the power, a split function can be defined. In any case, this

would result in a large number of experiments, and consequently, longer access to the cluster with the measuring instruments connected. If we have 13 different frequencies, and we do three repetitions of each experiment, we need almost 40 experiments. In this paper we will show how it is possible to obtain good predictions of the dissipated power by measuring only four frequencies, reducing the number of experiments by 70%.

3) PROBLEM SIZES SELECTION

The problem sizes for the experiments have to be defined. In applications that do not vary the amount of memory used throughout their execution, it will be enough selecting the size of the problem before each experiment. The selected problem sizes must not exceed the capacity of the main memory, to avoid the use of swap. If there are no user requirements for too-small problem sizes, it is preferable to avoid them, as they show greater variability.

4) APPLICATION SELECTION

System characterization is done for different problem sizes, therefore the selected application must allow to change this value. Problem size refers to the size of the program data that is stored in main memory. Checkpoint and restart operations save the program state, which is mainly program data, and are not affected by the application executed by the processor. Due to this, the consumption of the application does not intervene in the construction of the model and it is sufficient to characterize the system using a single application. Applications that allow to change the problem size in a simple and flexible manner are preferred.

B. REGRESSION ANALYSIS: OBTAINING FUNCTIONS AND COEFFICIENTS

The power and time measurements obtained must be analyzed to find the function that best approximates them, and then the coefficients of this function are obtained by the least squares method. In the following subsections, this analysis is shown (see section VI-A for the experimental platform used).

1) POWER

Fig. 6 shows the dissipated power during the CR for the four clock frequencies and the four per node problem sizes selected. When observing this cloud of points, we see that a linear or quadratic function could represent it. Applying the least squares method with both functions, we obtained smaller errors with the quadratic function, both for checkpoint and restart. The formulas obtained are the following:

$$P_C(f) = 19.7 \times f^2 - 37.7 \times f + 99 \quad (5)$$

$$P_R(f) = 9.55 \times f^2 - 18.62 \times f + 88.45 \quad (6)$$

where f is clock frequency and its unit is in GHz.

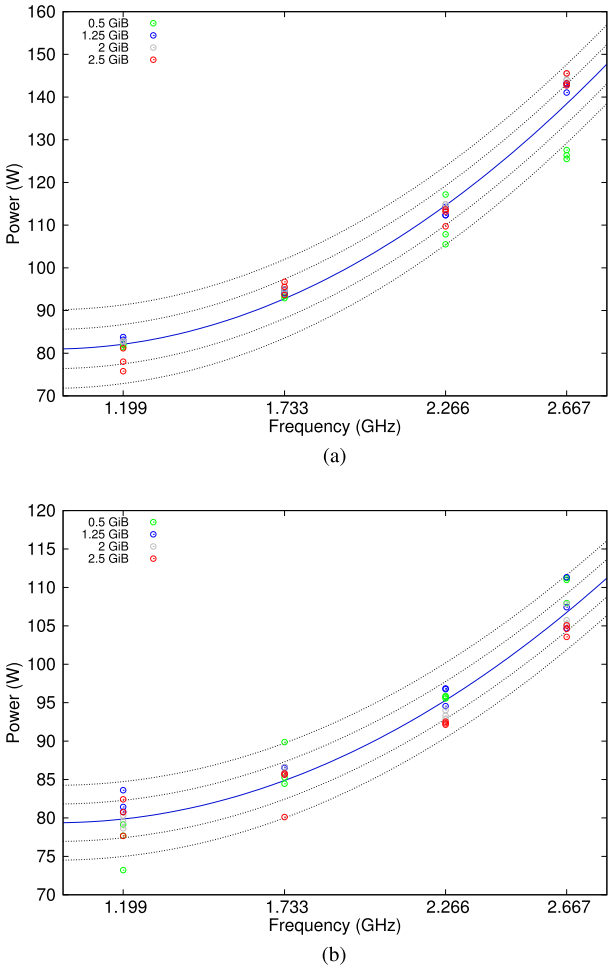

FIGURE 6. Power measurements and regression. (a) Checkpoint. (b) Restart.

Fig 6 also shows the regression obtained. As we can see, the measurements are mostly within the range of +/− two standard deviations (SD) from the curve. The exceptions are given, for the case of the checkpoint, at the maximum frequency, and in the case of the restart, at the lowest frequency. The average of the absolute errors can be considered low (3.2W for the checkpoint and 2W for the restart), as well as the standard deviations (4.6W for the checkpoint and 2.4W for the restart). It is worth mentioning that the standard deviation of the checkpoint is almost twice the standard deviation of the restart, which indicates a greater variability.

Fig. 7 shows the corresponding box graph. A few outliers can be observed. The outliers are values that are more than one and a half times away from the inter-quartile range. Another interesting observation is that checkpoint and restart higher boxes, that is, the greater dispersion of data, are given for the case of the maximum clock frequency (2.667 GHz).

2) TIME

Fig. 8 shows the execution times obtained for the four clock frequencies and the four per node problem sizes. When

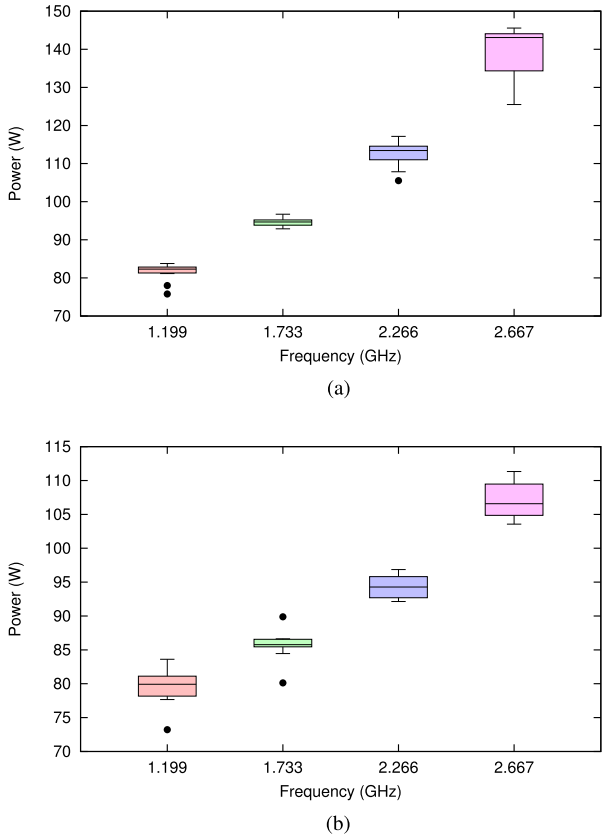

FIGURE 7. Power measurements box plots. (a) Checkpoint. (b) Restart.

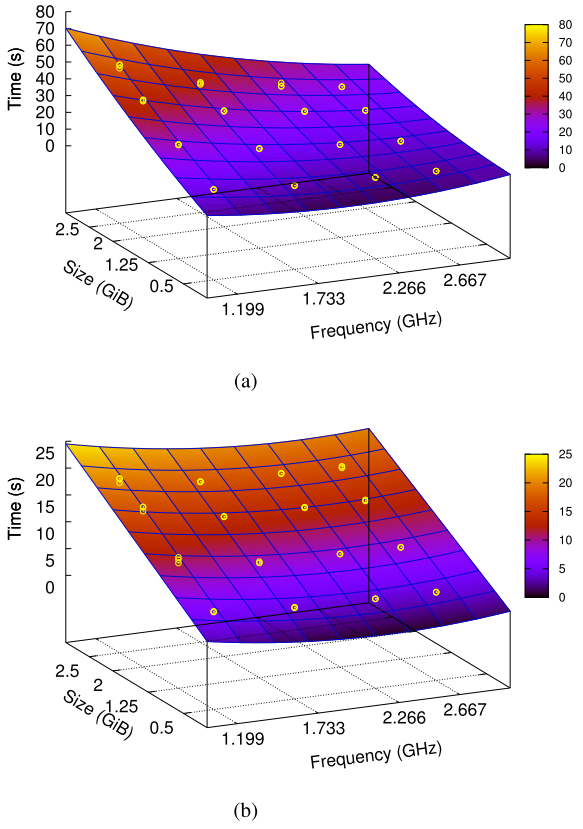

FIGURE 8. Time measurements and regression. (a) Checkpoint. (b) Restart.

observing this cloud of points, we see that a linear or quadratic function could represent it. We applied the least squares method on both functions and we obtained smaller errors with the quadratic function, both for checkpoint and restart. The formulas obtained are the following:

$$T_C(ps,f) = 1.42 \times ps^2 + 6.07 \times f^2 + 23.74 \times ps - 23.87 \times f - 7.67 \times ps \times f + 26.97 \quad (7)$$

$$T_R(ps,f) = 2.11 \times f^2 + 7.45 \times ps - 9.31 \times f - 0.46 \times ps \times f + 10.75 \quad (8)$$

where f is clock frequency in GHz and ps is the problem size per node in GiB. Fig. 8 also shows the regression, where 94% and 96% of the measurements fall within the band defined by $+/-$ two standard deviations for the case of the checkpoint and restart respectively. The correlation coefficient obtained is high (0.99 in both cases, checkpoint and restart), and the standard deviations give very acceptable values (1 s for the checkpoint and 0.42 s for the restart).

VI. RESULTS ANALYSIS

In this section, the experimental platforms and experimental design are detailed, the power and time models are validated for new clock frequencies (1.466 GHz, 1.999 GHz and 2.533 GHz) and new problem sizes (1 GiB, 1.5 GiB and 2.25 GiB), errors in the energy estimation are calculated, and the dissipated power on the two experimental platforms is compared.

A. EXPERIMENTAL PLATFORMS

The experiments were carried out on two platforms. Platform 1, on which most of the analysis of this work is carried out, and Platform 2, which is used to contrast the results obtained with a different hardware. Platform 1 is a cluster of computers, with a 1 Gbps Ethernet network. Each node, both computing and storage, has 4 GiB of main memory, a SATA hard disk of 500 GB and 7200 rpm, and an Intel Core i5-750 processor, with a frequency range of 1.2 GHz to 2.66 GHz (with the Intel Turbo Boost[3] disabled), four cores (without multithreading), 8 MiB of cache and 95 W TDP. Platform 2 is a compute node connected to a storage node with a 1 Gbps Ethernet network. The compute node has an Intel Xeon E5-2630 processor, a frequency range of 1.2 GHz to 2.801 GHz (with the Intel Turbo Boost mechanism disabled), six cores (with multithreading disabled), 16 GiB of main memory, 15 MiB of cache and TDP of 95 W. It uses a Debian 9 "Stretch" operating system. The storage computer has an Intel Core 2 Quad Q6600 processor, four cores (without

[3]https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html

multithreading), 8 GiB of main memory and 4 MiB of cache memory.

The nodes of Platform 1 and the computing node of Platform 2 use the GNU/Linux operating system Debian 8.2 Jessie (kernel version 3.16 of 64 bits), OpenMPI version 1.10.1 as an MPI message passing library, and the tool checkpoint DMTCP version 2.4.2, configured to compress the checkpoint files. The network file system used to make the remote writing of the checkpoint files is NFS v4 (Network File System).

For power measurements we use the PicoScope 2203 oscilloscope (whose accuracy is 3%), the TA041 active differential probe, and the PP264 60 A AC/DC current clamp, all Pico Technology products. The electrical signals captured by the two-channel oscilloscope are transmitted in real time to a computer through a USB connection. The voltage is measured using the TA041 probe that is connected to an input channel of the oscilloscope. The current of the phase conductor, which provides energy to the complete node (including the power source) is measured using the current clamp PP264, which is connected to the other input channel of the oscilloscope.

The selected application[4] for system characterization is a SPMD heat transfer application written in MPI that uses the float data type. This application describes, by means of an equation, the change of temperature in time, on a plane, given an initial temperature distribution and certain edge conditions.

4.2.6 Limits of the formalization. El segundo párrafo separa lo que es estrictamente ''tamaño del problema'' y ''trabajo/dificultad''. Entonces, en el artículo se podría aclarar que aquí no hacemos referencia a la ''dificultad'' de la tarea sino al tamaño de los datos que se almacenan en memoria principal.

B. EXPERIMENTAL DESIGN

Each compute node writes to a dedicated storage node through an NFS configured in asynchronous mode. The sampling rate used for both channels of the oscilloscope was set at 1000 Hz. The power measurements correspond to the power dissipated by the complete node including the source. The tests were performed with the processor C states option active. For the measurements of the checkpoint and restart time, we use the option provided by DMTCP. To change the frequency of the processor, the GNU/Linux userspace governor is used, which permits modifying the file sysfs scaling_setspeed available for each core. The same frequency is used in all cores at the same time.

Each experiment consists of launching the application with one process per core, letting it run during a preheating period (20 seconds), performing a checkpoint manually, aborting the application from the DMTCP coordinator and re-starting the application from the command line with the script generated by DMTCP. The experiment is repeated three times for each frequency and problem size due to the low variability of the measurement instruments used.

[4]https://computing.llnl.gov/tutorials/parallel_comp/#ExamplesHeat

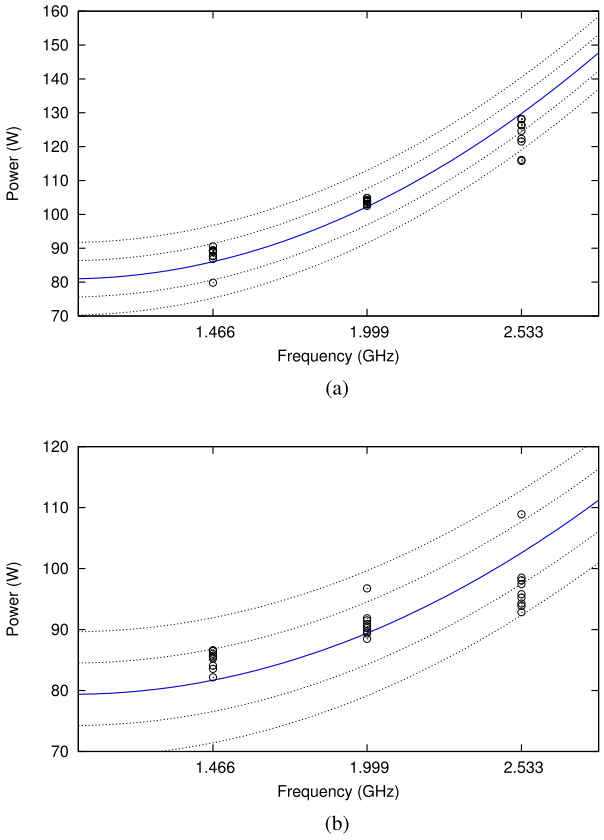

FIGURE 9. Checkpoint and restart power regression validation. (a) Checkpoint. (b) Restart.

C. MODEL VALIDATION

1) POWER VALIDATION

In order to validate our power model, we measured the dissipated power for new clock frequencies and calculate the errors obtained. Fig. 9 shows the graph with the regression and the new measurements. We can observe how most of the new values fall within the range between $+/-$ one standard deviation. The average of the errors' absolute values is low (3.7 W for the checkpoint and 4 W for the restart), and the standard deviations increase slightly when compared with the previous experiments (5.4 W for the checkpoint and 5.1 W for the restart), but they are still acceptable.

POWER ON PLATFORM 2

In order to observe the behavior of the power dissipated in a different computer, the same experiments were performed on Platform 2. Fig. 10 shows the initial observations (marked with circles), the regression function that best fits those observations, and validation measurements (marked with squares). As we can see, most measurements fall within the band

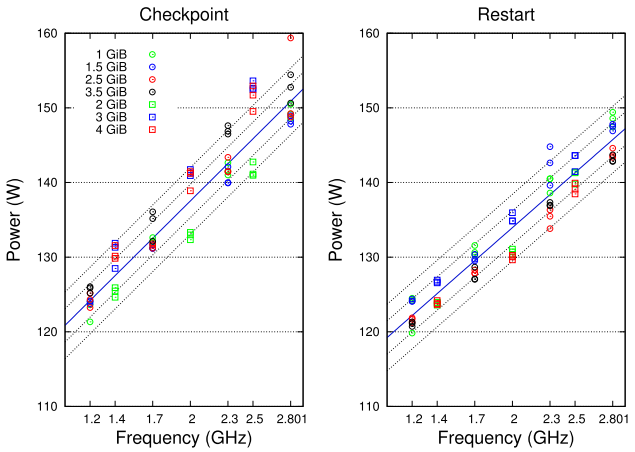

FIGURE 10. Platform 2 power linear regression and validation.

TABLE 1. Power on Platform 2: Correlation coefficient, standard deviations and average of absolute errors.

	Checkpoint	Restart
Correlation Coefficient	0.97	0.97
Standard Deviations (W)	2.34	2.26
Average of Absolute Errors (W)	1.68	1.87

defined by $+/-$ two standard deviations. Table 1 shows the indicators that validate the obtained regressions.

We will mention two differences with respect to the results obtained with Platform 1. On the one hand, in this case it is a lineal function that best fits the point cloud of the power, whereas on Platform 1 it was a curve. On the other hand, on this platform we observed that there is a relationship between power and problem size. For example, in the checkpoint case, we see large problem sizes to be mostly above the regression line, and small sizes, below. In any case, the estimation obtained is good enough and we don't consider it justified to make the model more complex.

Fig. 11 shows the real power of Platform 2. It presents a high and a low phase during the checkpoint, not present in Platform 1. The high phase indicates that the CPU is compressing, and the low phase that it is transferring and has finished compressing. In graphics not shown here, we can see how the high phase becomes shorter as the frequency increases (since it compresses faster).

2) TIME VALIDATION

In order to validate the model for the CR time, measurements were made for new frequencies and problem sizes, and the errors obtained were calculated. The errors in absolute value have an average of 0.7 seconds for the checkpoint and 0.4 seconds for the restart. The standard deviations are kept low, these being 1.03 seconds for the checkpoint and 0.6 seconds for the restart. With these new measurements, we can verify that the proposed model can predict CR time very well.

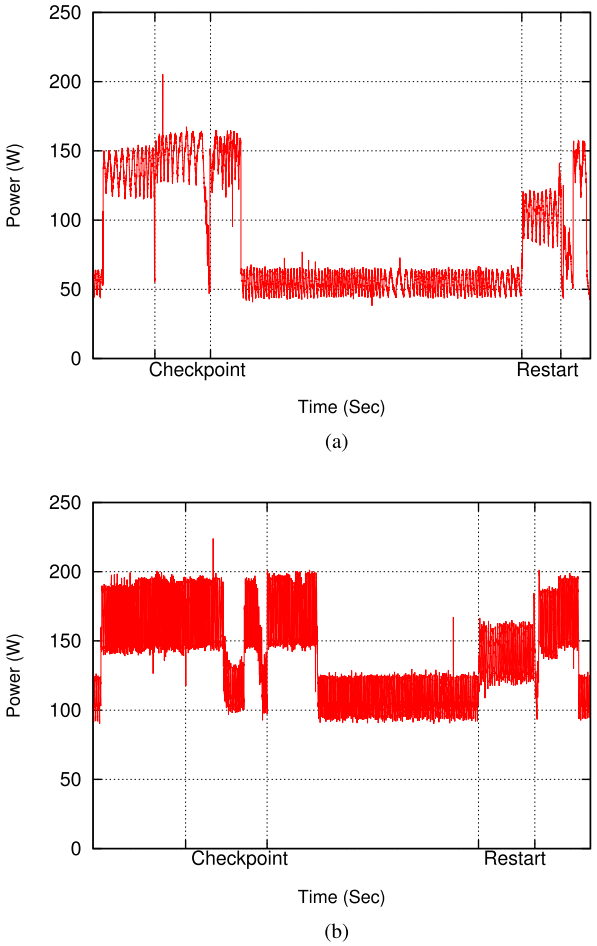

FIGURE 11. Real power dissipation during checkpoint and restart. (a) Platform 1 at 2.667 GHz. (b) Platform 2 at 2.5 GHz.

TABLE 2. Energy consumption prediction: Percentage of measurements that fall within +/- one and two standard deviation.

	+/- one SD	+/- two SD	Total
Checkpoint	79%	13%	92%
Restart	67%	29%	96%

3) ENERGY VALIDATION

We will now calculate the errors obtained in the prediction of the CR energy consumption. Fig. 12 shows the prediction function of the energy and the values observed in the experiment. More than 90% of the measurements fall within the range defined by $+/-$ two standard deviations (for both checkpoint and restart), as summarized in Table 2.

Table 3 shows the average error incurred for each size and frequency, in the prediction of the energy consumption. In the case of the checkpoint, the highest error percentages are around 6%, except for one case, corresponding to the smallest size and the maximum frequency (24.15%). In the case of the restart, the highest error percentages are around 7.5%, except for two cases, corresponding to the smallest size (10.54% and 12.16%). In this way, without taking outliers, we have been

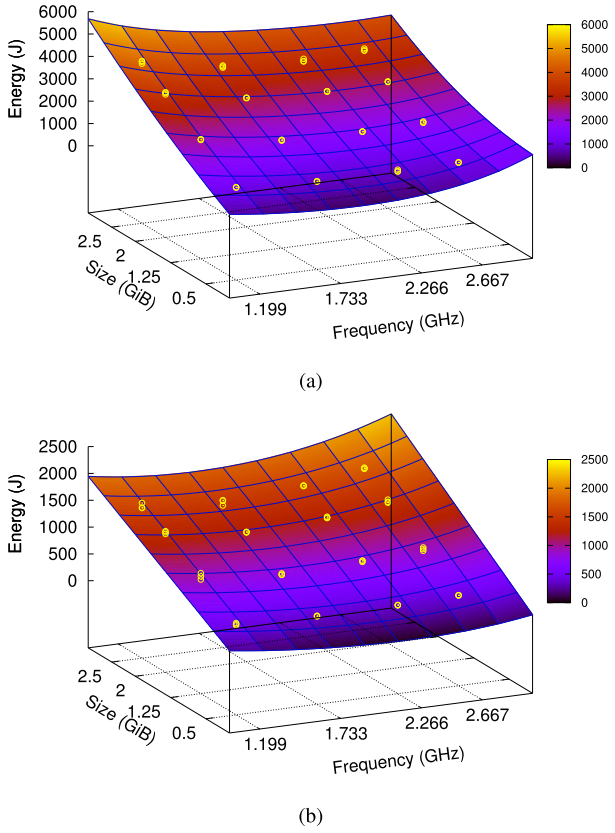

FIGURE 12. Energy measurements and prediction. (a) Checkpoint. (b) Restart.

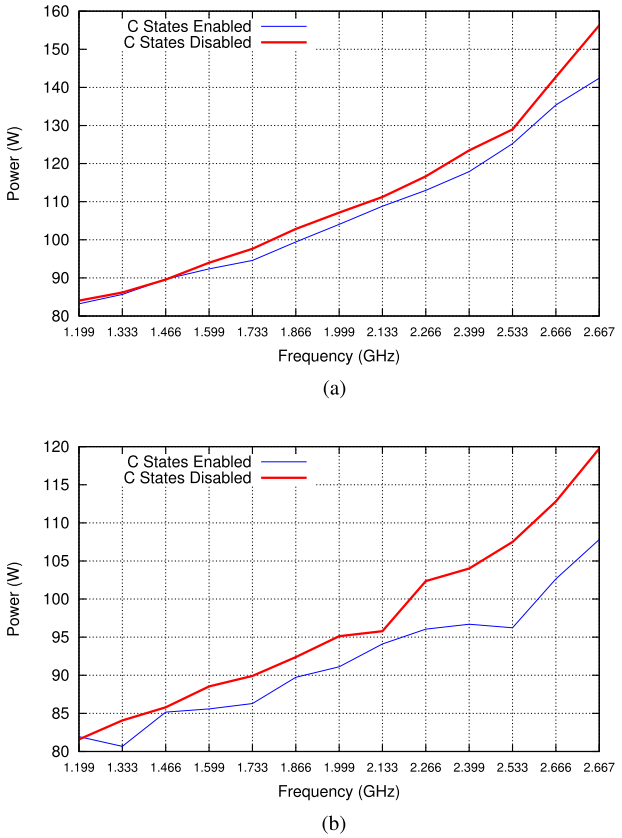

FIGURE 13. Power dissipated with C states enabled and disabled. (a) Checkpoint.(b) Restart.

TABLE 3. Average errors (in %) by frequency and problem size.

Problem Size (GiB)	Frequency (GHz)	Checkpoint	Restart
0.50	1.199	6.17%	10.54%
1.25	1.199	2.81%	5.37%
2.00	1.199	1.10%	3.29%
2.50	1.199	2.25%	2.64%
0.50	1.733	5.94%	7.46%
1.25	1.733	2.51%	1.91%
2.00	1.733	2.13%	3.81%
2.50	1.733	2.07%	2.45%
0.50	2.266	6.27%	12.16%
1.25	2.266	5.91%	1.87%
2.00	2.266	0.56%	2.10%
2.50	2.266	1.16%	0.32%
0.50	2.667	24.15%	0.92%
1.25	2.667	1.14%	3.41%
2.00	2.667	2.32%	2.45%
2.50	2.667	5.27%	0.75%

able to estimate the energy consumed by the CR with an error of less than 7.5%.

USE OF THE PREDICTION MODEL FOR THE CHECKPOINT ENERGY CONSUMPTION

Table 4 shows the prediction of the checkpoint energy consumption for a fixed problem size and ten clock frequencies. This allows us to know what clock frequency to use if we decide to prioritize the energy consumption. In this case, the frequency with the lowest energy consumption is 2.266 GHz. The table also shows the errors incurred.

TABLE 4. Predicted and measured energy (1.25 GiB problem size).

Frequency (GHz)	Measured Energy (J)	Estimated Energy (J)	Error (%)
1.199	2,321.73	2,256.52	-2.81
1.333	2,159.14	2,098.24	-2.82
1.599	1,927.73	1,860.72	-3.48
1.733	1,820.13	1,774.35	-2.51
1.866	1,813.91	1,708.66	-5.80
2.133	1,764.75	1,636.36	-7.28
2.266	1,734.54	1,631.96	-5.91
2.399	1,727.28	1,651.38	-4.39
2.666	1,781.61	1,776.15	-0.31
2.667	1,797.76	1,776.87	-1.16

VII. FACTORS THAT AFFECT THE ENERGY CONSUMPTION AND/OR THE PREDICTION QUALITY

In this section we analyze some system and application factors that influence the energy consumption of CR operations.

A. C STATES

During the writing or reading of a checkpoint file it is possible that the processor becomes idle and therefore transitions occur between the C states (section II-D) that affect the power dissipation of the processor. To study its behavior,

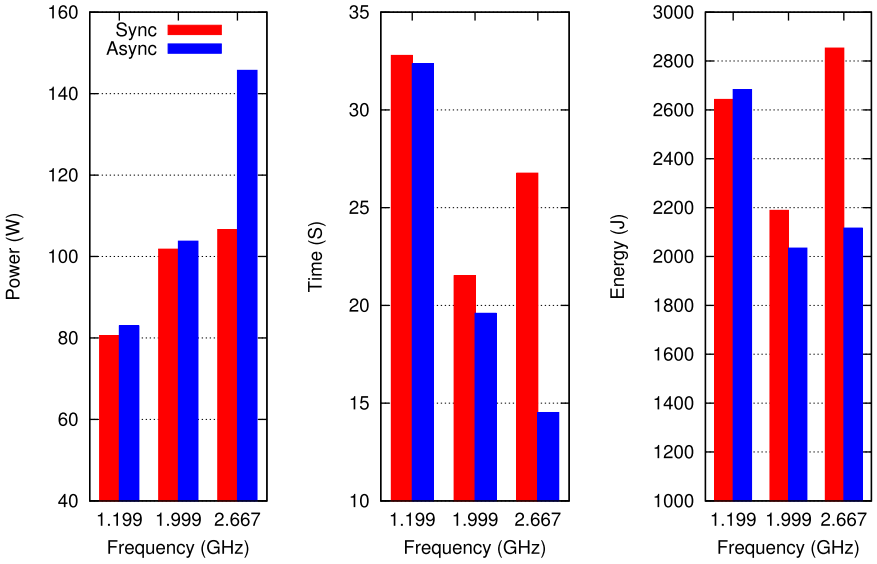

FIGURE 14. Power dissipated, time and energy consumed during checkpoint with sync and async NFS mode.

CR operations were performed with C states enabled and disabled, for all processor frequencies. In Fig. 13 it is observed that the power measurements with the C states enabled show greater variability, especially in the restart. This variability affects the prediction. In any case, the best option is to keep the C states enabled, since they reduce the energy consumption by up to 13% for the checkpoint, and up to 20% for the restart. The execution times showed no variation when enabling or disabling the C states.

B. NFS CONFIGURATION

The NFS allows folders to be mounted synchronously (sync option) or asynchronously (async option). If an NFS folder is mounted with the sync option, the writes at that mount point will cause the data to be completely downloaded to the NFS server and written to persistent storage before returning control to the client.[5] Thus, the time of a write operation is affected by varying this configuration.

Fig. 14 shows the comparison of the power dissipated, the execution time and the energy consumed by a checkpoint stored on a network file system mounted with the option sync and async, for three different clock frequencies (minimum, medium and maximum available in the processor). For all three frequencies, the dissipated power is greater and the execution time is shorter when the asynchronous configuration is used. This is because the idle time of the processor decreases when using this configuration (and therefore the average power increases).

For the small and middle frequency, the differences in time and power are small, resulting in a similar energy consumption. However, for the maximum frequency, this difference gets higher, reaching an energy saving of up to 25% when using the asynchronous mode.

[5]https://linux.die.net/man/5/nfs

It was also observed that the checkpoint times (for the same problem size and clock frequency) show greater variability in the synchronous mode, differences of up to 40% (in this case we carry out more experiments), when compared to the asynchronous mode, differences less than 1%. The low variability of the checkpoint time with the asynchronous mode benefits its prediction.

Due to its lower dissipated power, and its lower time variability, it is preferable to store the checkpoints in an NFS mounted with the async option.

C. COMPRESSION OF CHECKPOINT FILES

The compression or not of the checkpoint files impacts the time and the power demanded, therefore it is another factor that affects the energy consumption.

Fig. 15 compares the power dissipated, the time and the energy consumed by the checkpoint and restart, with and without compression of the checkpoint files, for three different clock frequencies (minimum, medium and maximum values available in the processor). The experiments were performed on a single compute node, writing to a single storage node. We can make the following observations when using compression compared with no compression:

- With regard to power: The dissipated power and its variability are greater, both for checkpoint and restart.
- With regard to time: The time variability is greater, both for checkpoint and restart. The restart is always faster. The checkpoint is faster only for the maximum frequency and slower for the minimum frequency.
- With regard to energy: The checkpoint consumes more energy (up to 55%) and the restart consumes less energy (up to 20%).

Taking into account that, in general, checkpoint is performed many more times than restart, it is advisable not to use compression to reduce energy consumption in this platform.

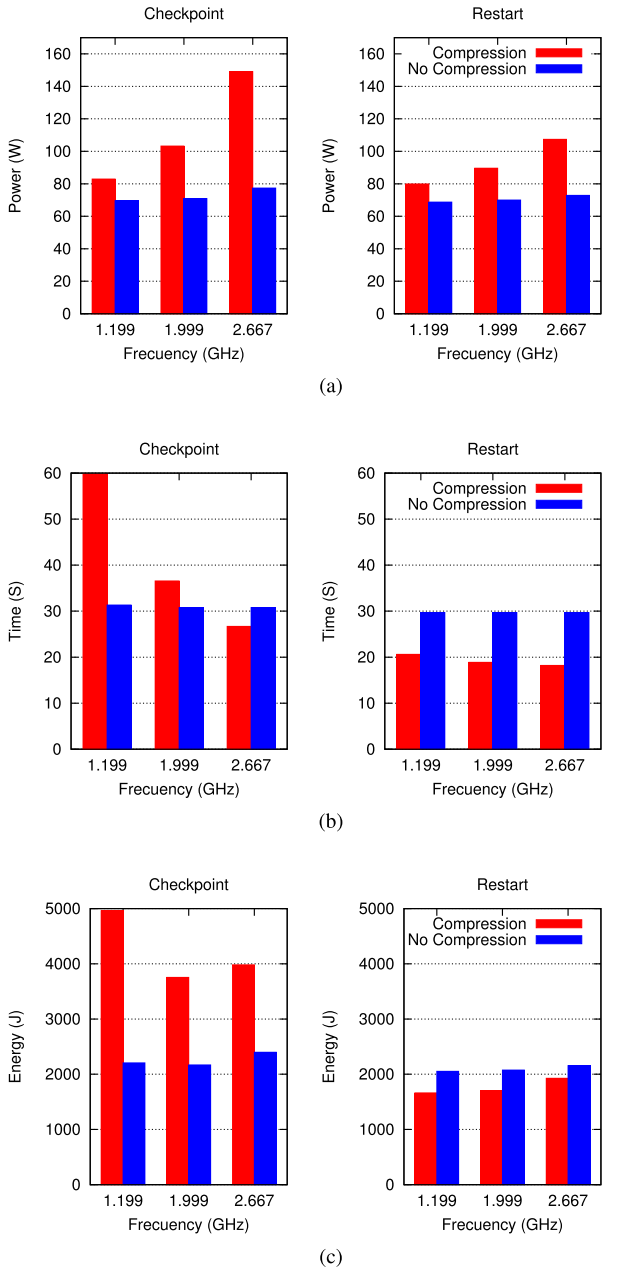

FIGURE 15. Comparison of compression/no compression of checkpoint files. (a) Power dissipated. (b) Time consumed. (c) Energy consumed.

VIII. CONCLUSIONS

This work shows how it is possible to estimate the energy consumption of system level coordinated CR operations, on an SPMD application executed in a homogeneous cluster. We proposed a method for constructing a model to predict the dissipated power and time of CR operations. The method, based on regression analysis, was applied and validated in an experimental platform. Besides, a regression analysis and validation was performed for the dissipated power of a different machine. With this model, it is possible to know the energy consumption for new CPU frequencies and problem sizes without the need to execute the checkpoint and restart of the application and to take measurements.

In addition, the impact of different system and application factors on energy consumption was evaluated in an experimental platform. In particular, the impact of the processor's C states, the synchronous and asynchronous configuration of the NFS, and the compression or not of the checkpoint files were evaluated. We saw how it is better, from an energy point of view, to use an asynchronous configuration of the NFS to enable the C states of the processor and so as not to compress the checkpoint files. These results show how it is possible to influence the energy consumption of CR.

This work contributes to the objective of predicting energy consumption in the execution of applications that use CR. Not counting the outliers, we can estimate the energy consumed by the CR with errors lower than 7.5%. Among future work it is expected to evaluate the energy behavior of other types of applications, of uncoordinated CR, as well as to evaluate the energy consumed by the storage node.

REFERENCES

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, and S. Karp, "Exascale computing study: Technology challenges in achieving exascale systems," Defense Adv. Res. Projects Agency, Arlington, VA, USA, Tech. Rep. 15, 2008.

[2] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Comput. Surv.*, vol. 34, no. 3, pp. 375–408, 2002.

[3] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *Proc. Int. Conf. High Perform. Comput. Comput. Sci.* Springer, 2010, pp. 1–25.

[4] S. Sankaran, J. M. Squyres, B. Barrett, V. Sahay, A. Lumsdaine, J. Duell, P. Hargrove, and E. Roman, "The LAM/MPI checkpoint/restart framework: System-initiated checkpointing," *Int. J. High Perform. Comput. Appl.*, vol. 19, no. 4, pp. 479–493, 2005.

[5] J. W. Young, "A first order approximation to the optimum checkpoint interval," *Commun. ACM*, vol. 17, no. 9, pp. 530–531, 1974.

[6] J. T. Daly, "A higher order estimate of the optimum checkpoint interval for restart dumps," *Future Gener. Comput. Syst.*, vol. 22, no. 3, pp. 303–312, 2006.

[7] N. El-Sayed and B. Schroeder, "To checkpoint or not to checkpoint: Understanding energy-performance-I/O tradeoffs in HPC checkpointing," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Sep. 2014, pp. 93–102.

[8] J. Ansel, K. Arya, and G. Cooperman, "DMTCP: Transparent checkpointing for cluster computations and the desktop," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, May 2009, pp. 1–12.

[9] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proc. Int. Conf. Power Aware Comput. Syst.*, 2010, pp. 1–8.

[10] M. el Mehdi Diouri, O. Glück, L. Lefevre, and F. Cappello, "Energy considerations in checkpointing and fault tolerance protocols," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN)*, Jun. 2012, pp. 1–6.

[11] E. Meneses, O. Sarood, and L. V. Kalé, "Energy profile of rollback-recovery strategies in high performance computing," *Parallel Comput.*, vol. 40, no. 9, pp. 536–547, 2014.

[12] B. Mills, T. Znati, R. Melhem, K. B. Ferreira, and R. E. Grant, "Energy consumption of resilience mechanisms in large scale systems," in *Proc. 22nd Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process.*, Feb. 2014, pp. 528–535.

[13] M. A. Amrizal and H. Takizawa, "Optimizing energy consumption on HPC systems with a multi-level checkpointing mechanism," in *Proc. Int. Conf. Netw., Archit., Storage (NAS)*, Aug. 2017, pp. 1–9.

[14] D. Dauwe, R. Jhaveri, S. Pasricha, A. A. Maciejewski, and H. J. Siegel, "Optimizing checkpoint intervals for reduced energy use in exascale systems," in *Proc. 8th Int. Green Sustain. Comput. Conf. (IGSC)*, Oct. 2017, pp. 1–8.

[15] B. Mills, T. Znati, and R. Melhem, "Shadow computing: An energy-aware fault tolerant computing model," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2014, pp. 73–77.
[16] M. el Mehdi Diouri, O. Glück, L. Lefevre, and F. Cappello, "ECOFIT: A framework to estimate energy consumption of fault tolerance protocols for HPC applications," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud, Grid Comput.*, May 2013, pp. 522–529.
[17] R. R. Chandrasekar, A. Venkatesh, K. Hamidouche, and D. K. Panda, "Power-check: An energy-efficient checkpointing framework for HPC clusters," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2015, pp. 261–270.
[18] X. Cui, T. Znati, and R. Melhem, "Adaptive and power-aware resilience for extreme-scale computing," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld)*, Jul. 2016, pp. 671–679. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0111.
[19] T. Saito, K. Sato, H. Sato, and S. Matsuoka, "Energy-aware I/O optimization for checkpoint and restart on a NAND flash memory system," in *Proc. 3rd Workshop Fault-Tolerance HPC Extreme Scale*, 2013, pp. 41–48.
[20] B. Mills, R. E. Grant, K. B. Ferreira, and R. Riesen, "Evaluating energy savings for checkpoint/restart," in *Proc. 1st Int. Workshop Energy Efficient Supercomput.*, 2013, pp. 6:1–6:8.
[21] D. Ibtesham, D. DeBonis, D. Arnold, and K. B. Ferreira, "Coarse-grained energy modeling of rollback/recovery mechanisms," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 708–713.
[22] A. Miyoshi, C. Lefurgy, E. van Hensbergen, R. Rajamony, and R. Rajkumar, "Critical power slope: Understanding the runtime effects of frequency scaling," in *Proc. 16th Int. Conf. Supercomput.*, 2002, pp. 35–44.

M. MORÁN received the degree in computer science from Universidad Nacional del Comahue, Neuquén, Argentina, in 1999. She is currently pursuing the Ph.D. degree in computer science with the Universidad Nacional de La Plata, La Plata, Argentina. Her research interests include fault tolerance and energy consumption in high-performance computing (HPC).

J. BALLADINI received the Ph.D. degree in computer science from the University Autònoma of Barcelona, Spain, in 2008. He is currently a Professor and the Director of the Computer Engineering Department, National University of Comahue, Argentina. He is also the Secretary of Science and Technology of the Network of Argentinian National Universities with degrees in Computer Science (RedUNCI). His major research interests include high-performance computing (HPC), energy consumption of HPC systems, and real-time big data processing in critical care units.

D. REXACHS received the Ph.D. degree in computer science from the University Autònoma of Barcelona (UAB), Spain, in 1988, where she is currently an Associate Professor with the Computer Architecture and Operating System Department. Her main research interests include parallel computer architecture, parallel input/output, fault tolerance in high-performance computing (HPC) systems, and tools to evaluate and predict the performance in parallel computers.

E. LUQUE received the Doctor Honoris Causa from the National University of La Plata (UNLP), Argentina, in 2018. He is currently an Emeritus Professor with the Computer Architecture and Operating System Department, University Autònoma of Barcelona (UAB), Spain, where he is leading the HPC4EAS Research Group. He is also an Invited Professor at different universities in USA, Asia, Europe, and South America, a key note speaker in computer science conferences, and the leader in several research projects funded by the European Union (EU), the Spanish Government, and different companies. He has been the supervisor of 24 Ph.D. theses and has published more than 270 papers in technical journals and international conferences. His major research interests include simulation and optimization of emergency services in hospitals (smarter health services), performance and scalability prediction in high-performance computing (HPC) systems, efficient applications, and fault tolerance in parallel computers. He is also a member of the Editorial Board of various technical journals. Recognitions for his research include the Distinguished Paper at the European Conference on Parallel Processing (Euro-Par 2006), the Outstanding Paper Award at the International Conference on Computational Science (ICCS 2011), the Best Paper Award at the International Conference on Computational Science (ICCS 2012), and the Best Paper Award at the Seventh International Conference on Advances in System Simulation (SIMUL 2015).