Metodología para predecir el consumo energético de checkpoints en sistemas de HPC

Javier Balladini[1], Marina Morán[1], Dolores Rexachs[2], Emilio Luque[2]

[1]Facultad de Informática, Universidad Nacional del Comahue
Buenos Aires 1400, 8300 Neuquén, Argentina
{javier.balladini, marina.moran}@fi.uncoma.edu.ar
[2]Departamento de Arquitectura de Computadores y Sistemas Operativos, Universitat Autònoma de Barcelona
Campus UAB, Edifici Q, 08193 Bellaterra (Barcelona), España
{dolores.rexachs, emilio.luque}@uab.es

Resumen Mientras el rendimiento de los sistemas de computación de altas prestaciones continúa creciendo, las máquinas aumentan significativamente en cantidad de unidades de procesamiento. Esto hace que la tolerancia a fallos y el consumo energético se conviertan en factores cada vez más relevantes. Los métodos de tolerancia a fallos tienen fuerte incidencia en el consumo energético, y resulta de suma importancia conocer, antes de ejecutar una cierta aplicación, el impacto que pueden producir los diferentes métodos y configuraciones del mismo. En este trabajo, presentamos una metodología para predecir el consumo energético producido por el método de checkpoint coordinado remoto. La metodología se basa en una caracterización energética del sistema, una caracterízación de la aplicación, y un modelo analítico que se instancia con los parámetros caracterizados. El modelo permite predecir la energía que consumirán los checkpoints para cualquier tamaño de problema y frecuencia de CPU de ejecución de checkpoints. Los resultados de las predicciones muestran una precisión mayor al 95 %.

Palabras claves: checkpoint, consumo energético, cómputo de altas prestaciones.

1. Introducción

Mientras el rendimiento de los sistemas de computación de altas prestaciones (HPC, High Performance Computing) continúa creciendo, las máquinas aumentan significativamente la cantidad de unidades de procesamiento. Este aumento en el número de componentes hace disminuir la confiabilidad y aumentar el consumo energético de un sistema de cómputo. Así, a pocos años de arribar a la era exaescala (prevista para 2020), la tolerancia a fallos y el consumo energético se han identificado como los dos mayores desafíos a los que deberemos enfrentarnos [4,3]. La magnitud del problema es muy significativa. Por un lado, en una máquina exaescala se espera tener un fallo cada pocos minutos [3], que sumado al mayor número de unidades de procesamiento del que harán uso las aplicaciones

(a medida que aumenta el rendimiento de los sistemas de HPC, también aumenta la complejidad de las aplicaciones, ya sea por aumento del tamaño del problema, la precisión de sus cálculos, o porque nuevos problemas son viables de resolver), producirá una inevitable necesidad de utilizar mecanismos de tolerancia a fallos.

Por otro lado, el consumo energético es hoy en día un gran problema, y más aún lo será con las máquinas exaescala. Para dar una idea de la magnitud del problema, utilizaremos de ejemplo la máquina de mayor prestaciones de la actualidad, la máquina China Tianhe-2. Según el TOP500[1], esta máquina demanda 17,8 MW de potencia, lo mismo que se requiere para abastecer a los hogares de una ciudad con 200.000 habitantes (calculo realizado en base al consumo de un hogar en Argentina). Considerando un contrato energético relativamente barato de 1 millón de dolares por MW, al año se debería pagar unos 17,8 millones de dolares al proveedor de energía, causando un alto impacto económico. Pero no solo eso, la generación de tanta energía tiene un alto impacto social (por ejemplo, la mayor fuente de energía mundial se obtiene del carbón, cuya extracción es altamente peligrosa), y medioambiental (por ejemplo, represas hidroeléctricas que modifican el ecosistema).

Los métodos de tolerancia a fallos tienen incidencia en el consumo energético y la potencia demandada por el sistema al ejecutar las aplicaciones. Esto se debe a que cualquier método agrega un significativo procesamiento, y posiblemente almacenamiento en disco, adicional al de la aplicación. Actualmente, el método de tolerancia a fallos más ampliamente utilizado en grandes sistemas de cómputo es el checkpoint/restart coordinado. Esta estrategia consiste principalmente en reiniciar la aplicación reemplazando el componente que ha fallado, pero, para evitar una reejecución completa de la aplicación, se guarda periódicamente el estado de ejecución en almacenamiento secundario, local o remoto. Cuando ocurre un fallo, de software o hardware, se reinicia la aplicación a partir del último estado de ejecución almacenado. El amplio uso de este método se debe a que tiene una implementación sencilla, y a que los puntos de sincronización requeridos para almacenar el estado de los procesos están presentes naturalmente en la mayoría de las aplicaciones paralelas.

Resulta de suma importancia conocer anticipadamente el impacto energético que pueden producir diferentes configuraciones del método de checkpoint utilizado para cada aplicación. Así, un administrador del sistema podría decidir qué configuración utilizar para mantener un cierto balance entre rendimiento, tolerancia a fallos y consumo energético. En este artículo se estudia el comportamiento energético de checkpoints coordinados remotos en sistemas de tipo cluster, considerando la realización de checkpoints a diferentes frecuencias de reloj de CPU, diferentes intervalos de tiempo de checkpoints, y diferentes tamaños de datos de la aplicación. La contribución principal de este artículo se encuentra en una metodología que permite construir un modelo para predecir el consumo energético que implica el uso del citado mecanismo de tolerancia a fallos en una dada aplicación, bajo nuevos tamaños de problema y frecuencias de CPU de ejecución de checkpoints.

[1] http://www.top500.org/

El resto de este artículo se organiza de la siguiente manera. En la sección 2 se da una breve reseña de los trabajos en esta área. En la sección 3 se presenta la metodología de predicción del consumo de energía de checkpoints coordinados remotos. En la sección 4, se presentan resultados experimentales utilizados para validar la metodología. Finalmente, en la sección 5 se presentan las conclusiones y trabajos futuros.

2. Trabajos relacionados

La evaluación del consumo de energía para distintos métodos de tolerancia a fallos a sido tratado en algunos pocos artículos. En [5], se presenta una evaluación energética de protocolos de checkpoint coordinado y no coordinado, diferenciando las tres tareas principales asociadas al método de checkpoint (escritura del checkpoint, recuperación, y registro de mensajes). En [6] se realiza una comparación de tres métodos de tolerancia a fallos basados en checkpoint: checkpoint/restart, registro de mensajes y recuperación paralela. Proponen un modelo analítico para predecir el comportamiento de dichos protocolos a exaescala. En [7] se examina el consumo de energía a nivel de componente durante las operaciones de checkpoint, y exploran la reducción del consumo energético, mediante el escalado dinámico de frecuencia y tensión, durante operaciones de checkpoint de entrada y salida intensiva. En [8], se utiliza un modelo analítico para estudiar el rendimiento de técnicas basadas en checkpoint coordinado y replicación. Asimismo, proponen una variante al modelo de replicación para mejorar el rendimiento energético y el tiempo de respuesta en comparación con la técnica de checkpoint/restart y la replicación tradicional.

Nuestro trabajo intenta predecir el consumo energético de la técnica de checkpoint coordinado remoto, basándose en la caracterización del sistema de cómputo y la aplicación. Una vez realizadas las caracterizaciones, el modelo analítico permite predecir cuál será el consumo energético de los checkpoints para cualquier tamaño de problema. Solo se realiza una medición real de energía en la caracterización del sistema, por lo que una vez realizado, se puede trabajar con cualquier nueva aplicación sin necesidad de medir nuevamente la energía. Creemos que no existen trabajos previos de estas características.

3. Metodología para predicción del consumo energético

Los checkpoints son operaciones limitadas por entrada y salida, y específicamente un checkpoint remoto es intensivo en el uso de la red. En este caso, cuando se quiere guardar el estado de un cierto proceso que está ejecutando en un determinado nodo, los datos del checkpoint se almacenan en un nodo diferente. Dependiendo del software y hardware del sistema de cómputo que se está utilizando, la CPU podría tener una alta incidencia en el tiempo y consumo energético de ejecución del checkpoint. Esto puedo ocurrir en sistemas que requieren significativos recursos de CPU para realizar las transmisiones de red. Por lo tanto, es necesario estudiar el impacto de la variación de frecuencia del

reloj de la CPU involucrada en la transmisión de datos por la red. Otro factor importante a analizar es el tamaño de checkpoint, que claramente impacta en el tiempo de transmisión de los datos de checkpoint al nodo remoto.

Nuestra metodología no considera actualmente el consumo energético del nodo que realiza la escritura del checkpoint en almacenamiento secundario (disco rígido), solamente se centra en los nodos del cluster que ejecutan los procesos cuyos estados se quieren preservar. Asimismo, las mediciones de potencia se realizan a nodos completos, incluyendo la fuente de energía.

Para predecir cuánta energía consumirán las distintas aplicaciones con sus checkpoints, hemos definido una metodología que consiste en tres fases. La primera, denominada *fase de caracterización del sistema*, tiene como objetivo obtener una ecuación representativa del comportamiento energético del sistema al ejecutar checkpoints. La segunda, denominada *fase de caracterización de la aplicación*, intenta obtener una ecuación que represente el comportamiento temporal de los checkpoints para esa aplicación. En la tercera, la *fase de instanciación del modelo energético*, se construye el modelo analítico para predecir el consumo energético de la aplicación y su mecanismo de tolerancia a fallos.

3.1. Fase de caracterización del sistema

El objetivo de esta fase es obtener los atributos energéticos representativos del sistema de cómputo, considerando tanto el hardware como el software, en relación a la potencia media demandada por el mismo para ejecutar el checkpoint remoto. Se requiere conocer la potencia media para que sea posible calcular la energía consumida por el sistema al ejecutar el checkpoint. En la figura 1 se muestra la demanda de potencia a lo largo del tiempo al realizar checkpoints de una misma aplicación a dos diferentes frecuencias de CPU, el primer intervalo entre 97,63 y 150,35 es a baja frecuencia, mientras que entre 201,822 y 253,42 es a alta frecuencia. Se puede observar a simple vista cómo se incrementa la potencia cuando se incrementa la frecuencia de CPU. A su vez, la potencia media de los checkpoints no depende ni de una aplicación en particular ni del tamaño de los checkpoints. En la tabla 1 se presentan algunas mediciones de potencia media de la ejecución de checkpoints de diferentes aplicaciones (benchmarks paralelos del NAS) y tamaños de datos, a una misma frecuencia de CPU. El resultado muestra que la mayor potencia media es apenas un 1,9 % más grande que la menor. Así, como la potencia media demandada por la ejecución del checkpoint solo depende de la frecuencia de CPU, nos centraremos en encontrar una ecuación para determinar la potencia media en función de la frecuencia de CPU.

La metodología para obtener la ecuación de potencia media comprende dos etapas. La primera consiste en, para distintas frecuencias de CPU, repetir una serie de checkpoints (para alguna aplicación) y registrar las muestras de potencia. Finalmente se promedian las muestras de potencia para cada frecuencia. A continuación, se realiza un análisis de regresión con el objetivo de encontrar la ecuación que representa la potencia media en función de la frecuencia de CPU. En la figura 2 se observan muestras de potencia media para tres frecuencias de CPU, y una regresión polinómica de segundo grado (que representa la potencia

media), $PowerCheck(f) = 76{,}806 - 11{,}5988 * f + 5{,}28026 * f^2$, que se ajusta exactamente a las tres muestras. Esta ecuación se utiliza luego para instanciar el modelo analítico y poder predecir el consumo de energía de la aplicación con su mecanismo de tolerancia a fallos.

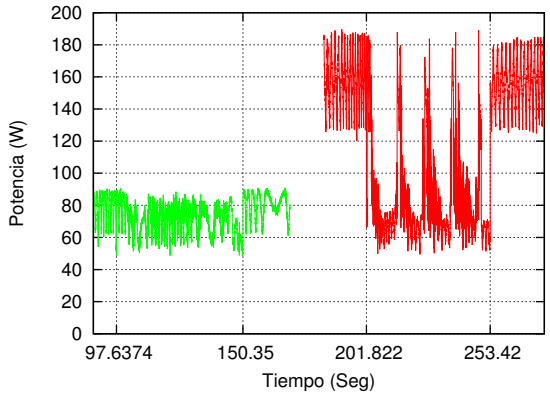

Fig. 1. Potencia que demanda un checkpoint a diferentes frecuencias de CPU.

Tabla 1. Potencia que demandan checkpoints con diferentes tamaños de datos.

Aplicación	Tamaño de checkpoint (MB)	Potencia media por nodo (W)
LU clase B	106,81	84,84
CG clase C	367,48	83,52
IS clase C	540,09	85,11

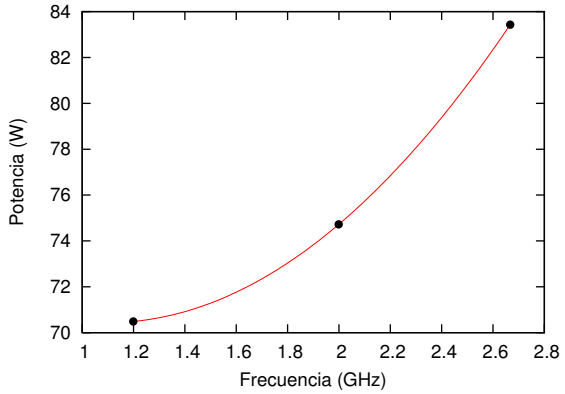

Fig. 2. Regresión de la potencia.

3.2. Fase de caracterización de la aplicación

Además de conocer la potencia media demandada por el nodo cuando ejecuta un checkpoint, para que sea posible calcular la energía consumida por el sistema, también se necesita conocer el tiempo que demora la ejecución del checkpoint.

El tiempo de ejecución del checkpoint de una aplicación depende de dos variables, la frecuencia de CPU y el tamaño del problema. En la tabla 2 (a) se muestra el tiempo de ejecución de un checkpoint correspondiente a una misma aplicación e igual tamaño de problema (CG clase C del NAS), a diferentes frecuencias de CPU. Puede observarse cómo el tiempo de ejecución disminuye a medida que se aumenta la frecuencia de CPU. Como el tiempo de ejecución varía, entonces confirmamos que depende de la frecuencia de CPU. En la tabla 2 (b) se muestra el tiempo de ejecución de un checkpoint de una aplicación (simulación de transferencia del calor), manteniendo fija la frecuencia de CPU y variando el tamaño del problema (dimensiones de una grilla). En este caso se aprecia la alta incidencia del tamaño del problema. Entonces, nos centraremos en encontrar una ecuación para determinar el tiempo de ejecución de checkpoints en función de la frecuencia de CPU y el tamaño del problema.

Tabla 2. Variables que influyen en el tiempo de ejecución de los checkpoints.

(a) Variando la frecuencia de CPU.

Frecuencia (GHz)	Tiempo de ejecución (Seg)
1,2	36,7
2	35,8
2,67	35,4

(b) Variando el tamaño del problema.

Tamaño	Tiempo de ejecución (Seg)
1000x1000	2,34
5000x5000	17,56
10000x10000	61,37

Similarmente al análisis de potencia, la metodología para obtener la ecuación del tiempo de ejecución de un checkpoint comprende dos etapas. La primera consiste en, para combinaciones de frecuencias de CPU y tamaños de problemas de una aplicación, repetir una serie de checkpoints y registran los tiempo de ejecución de ellos. Promediando los tiempos, se obtiene el tiempo de ejecución de checkpoint por cada combinación de frecuencia de CPU y tamaño de problema. Se deben seleccionar, como mínimo, tres frecuencias de CPU (alta, media y baja) y tres tamaños de problemas (grande, mediano y pequeño), obteniendo en este caso un total de nueve muestras; sin embargo, la cantidad de muestras necesarias puede variar según la aplicación. A continuación, se realiza un análisis de regresión con el objetivo de encontrar la ecuación que representa el tiempo de ejecución de checkpoint en función de la frecuencia de CPU y el tamaño de problema. En la figura 3 se observan muestras tomadas de tiempo de ejecución de checkpoints para distintas frecuencias de CPU y tamaños de datos, y una regresión polinómica de primer grado (que representa el tiempo de ejecución), con una varianza residual de 1,6, y cuya ecuación es:

$$TimeCheck(tam, f) = 1 + 6{,}05086 * 10^{-7} * tam + f - 1{,}16005 * 10^{-8} * tam * f$$

Esta ecuación luego se utiliza para instanciar el modelo analítico y predecir el consumo de energía de la aplicación con su método de tolerancia a fallos.

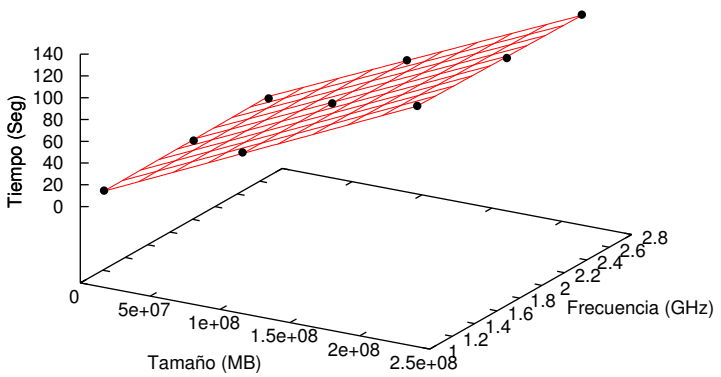

Fig. 3. Regresión del tiempo de ejecución de checkpoints.

3.3. Fase de instanciación del modelo energético

Habiendo caracterizado el sistema y la aplicación, el siguiente paso es instanciar el modelo con las ecuaciones obtenidas por medio de la caracterización. Una vez instanciado el modelo, ya se podrá predecir la energía que consumirá la aplicación con sus checkpoints, a cualquier frecuencias de CPU y tamaño de problema. A continuación se describe en detalle el modelo analítico.

El modelo analítico define la ecuación 1 que busca determinar la energía total que consumirá la ejecución de la aplicación con su mecanismo de tolerancia a fallos. La ecuación considera la energía consumida por el cómputo propio de la aplicación E_{app} y el consumo energético producido por la ejecución de los checkpoints E_{checks}.

$$E_{total} = E_{app} + E_{checks} \tag{1}$$

E_{app} es un parámetro de entrada del modelo; en particular, para el tipo de aplicación SPMD (Single Program Multiple Data), E_{app} puede obtenerse aplicando la metodología que propuesta en [2]. La ecuación 2 define E_{checks}, que depende del número de nodos $nNodes$, el tiempo de ejecución de la aplicación T_{app}, el intervalo de tiempo entre checkpoints $CheckInt$ y el consumo energético de cada checkpoint E_{check}.

$$E_{checks} = \sum_{j=1}^{nNodes} \sum_{i=0}^{T_{app}/CheckInt} E_{check_i} \tag{2}$$

$CheckInt$ y T_{app} son parámetros de entrada del modelo. En particular, para el tipo de aplicación SPMD, T_{app} puede obtenerse aplicando la metodología propuesta en [2]. E_{check_i} es el consumo energético producido por el *i-ésimo* checkpoint, que se define en la ecuación 3, y depende del tiempo que demora la ejecución del checkpoint T_{check} y la potencia media $PCheck(f)$ demandada por el sistema de cómputo durante la ejecución de un checkpoint bajo la frecuencia de CPU f.

$$E_{check} = T_{check} * PowerCheck(f) \quad (3)$$

La ecuación que define $PowerCheck(f)$ se obtiene a partir de la caracterización de potencia del sistema, realizada en la fase de caracterización del sistema. La ecuación que define T_{check} se obtiene a partir de la caracterización del tiempo de ejecución de checkpoints realizada en la fase de caracterización de la aplicación (es decir, la ecuación $TimeCheck(tam, f)$). Nótese que $PowerCheck(f)$ es dependiente del sistema e independiente de la aplicación, mientras que T_{check} es dependiente de ambos, el sistema y la aplicación.

4. Resultados experimentales

En este apartado se explica la plataforma de hardware y software utilizada, la metodología de medición de potencia, y finalmente se presenta la validación experimental de la metodología de predicción energética.

4.1. Plataforma experimental de cómputo (hardware y software)

Los experimentos se realizaron sobre un cluster de computadoras personales, con una red Ethernet de 100 Mbps, y nodos compuestos por una memoria principal de 4 GB, un disco rígido SATA de 500 GB y 7200 rpm, y un procesador Intel Core i5-750, con un rango de frecuencia de 1.2 GHz a 2.66 GHz (sin utilizar el mecanismo Intel Turbo Boost), cuatro cores (sin multithreading), y 8MB de caché. Los nodos utilizan el sistema operativo Debian GNU/Linux wheezy/sid, OpenMPI version 1.4.5 como librería de paso de mensajes MPI, y la herramienta de checkpoint DMTCP[1] (Distributed MultiThreaded Checkpointing) versión 1.2.5. Las aplicaciones seleccionadas para la caracterización del sistema son parte de la serie de benchmarks paralelos del NAS, mientras que la aplicación paralela de interés para realizar las predicciones de consumo energético es un simulador de transferencia de calor realizado en MPI. El sistema de archivos de red, utilizado para hacer la escritura remota de los datos de checkpoint, es NFS (Network File System).

4.2. Metodología de medición de potencia

Para las mediciones de potencia utilizamos el osciloscopio PicoScope 2203, la sonda diferencial activa TA041, y la pinza de corriente PP264 60 A AC/DC, todos productos de Pico Technology. Las señales eléctricas capturadas por el osciloscopio de dos canales son transmitidas en tiempo real a una notebook a través de una conexión USB. La tensión se mide utilizando la sonda TA041 que se conecta a un canal de entrada del osciloscopio. La corriente del conductor fase, de suministro energético del nodo completo (incluida la fuente de energía), se mide usando la pinza de corriente PP264 que se conecta al otro canal de entrada del osciloscopio. Entonces, la potencia se calcula como el producto de las mediciones de tensión y corriente. La tasa de muestreo utilizada para ambos canales del osciloscopio se estableció en 1000 Hz.

4.3. Validación experimental de la metodología

Para validar nuestro método, definimos un escenario de dos nodos de cómputo (del cluster) en la cual ejecuta la aplicación de simulación de transferencia de calor, y dos nodos dedicados a entrada y salida, en donde se escriben los datos de los checkpoints. Las mediciones de potencia son realizadas sobre uno de los dos nodos que ejecutan la aplicación paralela. Por simplicidad, se presenta un caso donde se realiza un único checkpoint a la aplicación.

Primero, se realiza la fase de caracterización del sistema, que comprende al análisis de la potencia al ejecutar checkpoints. En este caso, tomaremos la ecuación presentada en el apartado 3.1, ya que corresponde justamente a esta plataforma experimental de cómputo. Los benchmarks paralelos del NAS utilizados para esta fase fueron CG, IS y LU. Luego, se realiza la fase de caracterización de la aplicación, que comprende el análisis del tiempo de ejecución de los checkpoints para la aplicación de simulación de transferencia de calor. Finalmente, se lleva a cabo la fase de instanciación del modelo energético, en la cual se agregan al modelo las dos ecuaciones obtenidas en las fases anteriores.

En la tabla 3 se muestran, para uno de los nodos, los resultados reales y predichos de tiempo y potencia, para dos tamaños de problema y una determinada frecuencia de CPU (diferentes a las utilizadas para la caracterización del sistema y aplicación). En la tabla 4 se muestra, para un solo nodo, el consumo energético real, predicho y error de predicción cometido, para el checkpoint solo y la aplicación con el checkpoint (que, sin incluir el checkpoint, rondan los 54 segundos). El error de precisión para un checkpoint es de solo un 4,8 %, que naturalmente disminuye al considerar la aplicación completa.

Tabla 3. Predicción del tiempo de ejecución y potencia.

Tamaño	Frecuencia (Hz)	Tiempo real (Seg)	Tiempo estimado (Seg)	Potencia real (W)	Potencia estimada (W)
6.000x6.000	2,399	23,47	24,18	77,94	79,36
12.000x12.000	1,599	88,26	87,06	72,98	71,76

Tabla 4. Precisión de la predicción del consumo energético.

Tamaño	Checkpoint			Aplicación con checkpoint		
	E. real (Joules)	E. est. (Joules)	Error (%)	E. real (Joules)	E. est. (Joules)	Error (%)
6.000x6.000	1829,77	1918,92	4,8	7795,39	7710,816	1,08
12.000x12.000	6441,19	6247,42	3	11682,11	11477,56	1,75

5. Conclusiones y trabajos futuros

En este trabajo presentamos una metodología capaz de predecir el consumo energético de un sistema de cómputo al ejecutar una aplicación con checkpoints coordinados remotos. La metodología inicia con una con una caracterización

energética del sistema. Cada vez que se tiene una nueva aplicación, se hace una caracterización temporal (no energética) de checkpoints. Con estos datos, se instancia un modelo analítico que permite predecir para cualquier tamaño de problema y frecuencia de CPU. La metodología es relativamente simple de aplicar, y las pruebas experimentales con una aplicación de transferencia de calor muestran una precisión de predicción mayor al 95 %. Creemos que este trabajo puede resultar útil a la comunidad del cómputo de altas prestaciones en diversos sentidos. En especial, para determinar el impacto energético del checkpoint cuando una aplicación escala y ve reducido su tamaño de problema por nodo. También, podría utilizarse para realizar una predicción energética a diferentes frecuencias de CPU y luego determinar cuál de ellas es más conveniente en cuanto a tiempo, potencia y consumo energético. Los trabajos futuros se centran en complementar el modelo con el reinicio de la aplicación después de un fallo, ampliar las pruebas experimentales, abarcar más métodos de tolerancia a fallos, e integrar el modelo aquí presentado en un modelo de predicción energética de aplicaciones SPMD.

Referencias

1. Ansel, J., Arya, K., Cooperman, G.: DMTCP: Transparent Checkpointing for Cluster Computations and the Desktop. In: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing. pp. 1–12. IPDPS '09, IEEE Computer Society, Washington, DC, USA (2009)
2. Balladini, J., Muresano, R., Suppi, R., Rexachs, D., Luque, E.: Methodology for predicting the energy consumption of SPMD application on virtualized environments. Computer Science & Technology 13(1), 130–136 (Dec 2013)
3. Bergman, K., et al.: ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems (2008)
4. Dongarra, J., et al.: The international exascale software project roadmap. Int. J. High Perform. Comput. Appl. 25(1), 3–60 (Feb 2011)
5. El Mehdi Diouri, M., Gluck, O., Lefevre, L., Cappello, F.: Energy considerations in checkpointing and fault tolerance protocols. In: Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on. pp. 1–6 (June 2012)
6. Meneses, E., Sarood, O., Kale, L.V.: Assessing energy efficiency of fault tolerance protocols for hpc systems. In: Proceedings of the 2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing. pp. 35–42. SBAC-PAD '12, IEEE Computer Society, Washington, DC, USA (2012)
7. Mills, B., Grant, R.E., Ferreira, K.B., Riesen, R.: Evaluating energy savings for checkpoint/restart. In: Proceedings of the 1st International Workshop on Energy Efficient Supercomputing. pp. 6:1–6:8. E2SC '13, ACM, New York, NY, USA (2013)
8. Mills, B., Znati, T., Melhem, R., Ferreira, K.B., Grant, R.E.: Energy consumption of resilience mechanisms in large scale systems. In: Proceedings of the 2014 22Nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. pp. 528–535. PDP '14, IEEE Computer Society, Washington, DC, USA (2014)