UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Trabajo de Fin de Grado

Grado en Ingeniería Informática
Especialidad en Ingeniería de Computadores

Gestión de límites de energía globales para centros HPC

Autora:
Aurora Tomás Berjaga

Directora:
Julita Corbalán González, PhD

Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya - Barcelona Tech

2 de julio de 2020

Resumen

Actualmente la gestión energética se ha convertido en un tema fundamental de investigación en el ámbito HPC, pues los centros de supercomputación son grandes consumidores de energía, lo cual es un desafío tanto a nivel ecológico como económico. Sin embargo, la gestión de la energía no es algo banal ya que influyen otros factores que no solamente residen en la arquitectura en la cual se trabaja. Por este motivo es necesaria la ayuda de un software que se encargue de esta gestión.

Este proyecto se ha desarrollado dentro del contexto de Energy Aware Runtime (EAR), el cual es un software cuya función es gestionar la energía en centros HPC. EAR está formado por diversos componentes que se coordinan entre sí, entre los cuales se encuentra el Global Manager, que es el núcleo de este trabajo.

El Global Manager es el encargado de garantizar que el consumo energético durante un periodo de tiempo en un sistema no sobrepase un límite energético establecido, cuya funcionalidad recibe el nombre de *energy capping*. Este componente puede funcionar en dos modos distintos: modo manual y modo automático. El primero se limitará a controlar el consumo energético que está habiendo en el sistema, dejando que el administrador de sistemas sea quien realice las acciones que considere convenientes para el cumplimiento de los límites. El modo automático, además de monitorizar, es capaz de adaptar la configuración del sistema dinámicamente.

El modo manual se encuentra cubierto y en producción. Por contra, para el modo automático solo se encuentra un diseño inicial y carece de una evaluación, por lo que el objetivo de este trabajo es permitir que el Global Manager pueda operar automáticamente, buscando ser más flexibles.

Para ello se ha realizado una evaluación del estado inicial en que, en función de los resultados, se ha extendido la API añadiendo mejoras en funcionalidades y se han optimizado los ajustes dinámicos del Global Manager para una recuperación más rápida.

Se han diseñado un conjunto de experimentos con los que se ha evaluado extensamente el componente con todas las optimizaciones, llegando a utilizar hasta 26 nodos (1024 *cores*), aproximadamente un 10 % de los recursos de la máquina en la que se ha evaluado.

Finalmente, destacar que la nueva versión se adapta más rápidamente a las variaciones en la carga de trabajo controlando que no se exceda de los límites, según demuestran los experimentos realizados.

Palabras clave: EAR, gestión energética, límites energéticos, Global Manager, HPC

Resum

Actualment la gestió energètica s'ha convertit en un tema fonamental de recerca en l'àmbit HPC, doncs els centres de supercomputació són grans consumidors d'energia, la qual cosa és un desafiament tant a nivell ecològic com econòmic. No obstant això, la gestió de l'energia no és una cosa banal, ja que influeixen altres factors que no només resideixen a l'arquitectura en la qual es treballa. Per aquest motiu és necessària l'ajuda d'un software que s'encarregui d'aquesta gestió.

Aquest projecte s'ha desenvolupat dins del context d'Energy Aware Runtime (EAR), el qual és un software la funció del qual és gestionar l'energia en centres HPC. EAR està format per diversos components que es coordinen entre si, entre els quals es troba el Global Manager, que és el nucli d'aquest treball.

El Global Manager és l'encarregat de garantir que el consum energètic durant un període de temps en un sistema no sobrepassi un límit energètic establert, la funcionalitat del qual rep el nom d'*energy capping*. Aquest component pot funcionar en dos modes diferents: mode manual i mode automàtic. El primer es limitarà a controlar el consum energètic que està havent-hi en el sistema, deixant que l'administrador de sistemes sigui qui realitzi les accions que consideri convenients per al compliment dels límits. El mode automàtic, a més de monitoritzar, és capaç d'adaptar la configuració del sistema dinàmicament.

El mode manual es troba cobert i en producció. Per contra, per al mode automàtic només es troba un disseny inicial i manca d'una avaluació, per la qual cosa l'objectiu d'aquest treball és permetre que el Global Manager pugui operar automàticament, buscant ser més flexibles.

Per a això s'ha realitzat una avaluació de l'estat inicial en què, en funció dels resultats, s'ha estès l'API afegint millores en funcionalitats i s'han optimitzat els ajustos dinàmics del Global Manager per a una recuperació més ràpida.

S'han dissenyat un conjunt d'experiments amb els quals s'ha avaluat extensament el component amb totes les optimitzacions, arribant a utilitzar fins a 26 nodes (1024 *cores*), aproximadament un 10 % dels recursos de la màquina en la qual s'ha avaluat.

Finalment, destacar que la nova versió s'adapta més ràpidament a les variacions en la càrrega de treball controlant que no s'excedeixi dels límits, segons demostren els experiments realitzats.

Paraules clau: EAR, gestió energètica, límits energètics, Global Manager, HPC

Abstract

Currently, energy management has become a fundamental research topic in the HPC field, since supercomputing centers are large consumers of energy, which is a challenge both ecologically and economically. However, energy management is not banal since other factors influence that not only reside in the architecture in which it works. For this reason, the help of a software that is in charge of this management is necessary.

This project has been developed within the context of the Energy Aware Runtime (EAR), which is a software whose function is to manage energy in HPC centers. EAR is made up of various components that coordinate with each other, including the Global Manager, which is the core of this project.

The Global Manager is in charge of guaranteeing that the energy consumption over a period of time in a system does not exceed an established energy limit, whose functionality is called energy capping. This component can work in two different modes: manual mode and automatic mode. The first will be limited to controlling the energy consumption that is taking place in the system, leaving the system administrator to carry out the actions it deems appropriate to comply with the limits. Automatic mode, in addition to monitoring, is able to adapt the system configuration dynamically.

Manual mode is covered and in production. In contrast, for automatic mode, only an initial design is found and lacks an evaluation, so the objective of this project is to allow the Global Manager to operate automatically, seeking to be more flexible.

For this, an initial state evaluation has been carried out in which, based on the results, the API has been extended adding improvements in functionalities and the dynamic settings of the Global Manager have been optimized for a faster recovery.

A set of experiments has been designed with which the component has been extensively evaluated with all the optimizations, using up to 26 nodes (1024 cores), approximately 10 % of the resources of the machine in which has been evaluated.

Finally, it should be noted that the new version adapts more quickly to variations in workload, controlling that the limits are not exceeded, as shown by the experiments carried out.

Keywords: EAR, energy management, energy capping, Global Manager, HPC

Agradecimientos

En primer lugar quiero agradecer a Julita por haber sido una fantástica directora y por haberme dado a conocer el campo de la computación de altas prestaciones, del cual no contaba con experiencia previa y he aprendido muchísimo durante estos meses. Ha sido una gran experiencia trabajar con ella.

Gracias al equipo técnico de EAR y al grupo Lenovo por darme soporte y permitirme participar en este proyecto tan asombroso.

Quiero agradecer a los profesores que he conocido y han sido, y son, un pilar importante para mí. Quizá estéis leyendo esto, así que gracias de todo corazón.

Por otro lado, quiero dar las gracias a mi familia por creer en mí, por apoyarme siempre y darme ánimos.

Y a todos los amigos que he conocido durante estos años en la universidad, que han estado a mi lado y hemos pasado muy buenos e inolvidables momentos juntos, compartiendo inquietudes y animándonos mutuamente.

Índice general

Índice de figuras **13**

Índice de tablas **14**

Acrónimos **15**

1. Introducción **17**
1.1. Contexto 17
1.2. Formulación del problema 20
1.3. Actores implicados 21

2. Justificación **22**

3. Alcance del proyecto **24**
3.1. Objetivos 24
3.2. Requerimientos 25

4. Metodología y rigor **26**
4.1. Metodología de trabajo 26
4.2. Herramientas de seguimiento 26
4.3. Herramientas de validación 27

5. Planificación temporal **28**
5.1. Descripción de tareas 28
5.1.1. Gestión del proyecto 28
5.1.2. Familiarización con el entorno 29
5.1.3. Evaluación del estado de servicio EARGM 30
5.1.4. Modificación en la configuración 30
5.1.5. Evaluación y análisis de resultados 30
5.2. Diagrama de Gantt 31
5.3. Distribución de trabajo 31
5.4. Recursos 33
5.4.1. Recursos hardware 33

5.4.2. Recursos software 33
5.4.3. Recursos humanos 34
5.5. Gestión del riesgo 34

6. Gestión económica **37**
6.1. Costes del proyecto 37
6.1.1. Costes directos 37
6.1.2. Costes indirectos 39
6.1.3. Contingencias e imprevistos 40
6.2. Control de gestión 40
6.3. Presupuesto 41

7. Motivación **42**
7.1. Introducción 42
7.2. Gestión de la energía mediante DVFS 43

8. EAR **47**
8.1. Descripción general 47
8.2. EARL: Librería de EAR 49
8.3. Ejecución y comandos 54

9. Global Manager **55**
9.1. Descripción y estado inicial 55
9.2. Modificaciones específicas 65

10. Análisis de aplicaciones **67**
10.1. Entorno de ejecución 67
10.2. Descripción 68
10.3. Estudio de la potencia 69
10.4. Métricas básicas 73

11. Propuesta de optimizaciones **77**
11.1. Corrección de la configuración inicial 77
11.2. Ramp up 78
11.3. Ramp down 81

12. Evaluación **82**
12.1. Configuración del sistema 82
12.2. Workloads 83
12.3. Experimentos de validación 86
12.4. Experimentos de rendimiento 88
12.4.1. Experimento 20.1: Kernel intensivo en CPU 89
12.4.2. Experimento 20.5: Mix dinámico intensivo en CPU 94
12.4.3. Experimento 20.6: Mix dinámico intensivo en memoria 100

12.4.4. Experimento 20.7: Mix dinámico con pérdida de carga 105
12.5. Conclusiones 110

13. Informe de sostenibilidad **111**
13.1. Dimensión ambiental 112
13.2. Dimensión económica 112
13.3. Dimensión social 113
13.4. Contribución de EAR 113

14. Conclusiones **116**

15. Trabajo futuro **118**

A. Glosario **120**

B. Diagrama de Gantt **121**

C. Ejemplos de scripts **122**
C.1. Kernel intensivo en CPU 122
C.2. Parte variable en los mix dinámicos 123

D. Experimentos de validación **124**
D.1. Exp. 4.1: Kernel intensivo en CPU 124
D.2. Exp. 4.2: Aplicación intensiva en memoria 130
D.3. Exp. 4.3: Kernel intensivo en memoria 134
D.4. Exp. 4.4: Mix estático 138
D.5. Exp. 4.5: Mix dinámico 144

E. Experimentos de rendimiento **150**
E.1. Exp. 26.1: Aplicación intensiva en memoria 150
E.2. Exp. 20.2: Kernel intensivo en CPU de larga duración 153
E.3. Exp. 20.3: Kernel intensivo en memoria 155
E.4. Exp. 20.4: Mix estático 159
E.5. Exp. 20.5: Mix dinámico intensivo en CPU 165
E.6. Exp. 20.6: Mix dinámico intensivo en memoria 166
E.7. Exp. 20.7: Mix dinámico con pérdida de carga 167

Referencias **168**

Índice de figuras

1.1. Logo de EAR 17
1.2. Esquema de las relaciones entre los distintos componentes en EAR 19
1.3. Supercomputador SuperMUC-NG 20

7.1. Organización de los p-states en Lenox 44
7.2. Variación en la energía de aplicaciones respecto 2,4 GHz 45
7.3. Promedio del tiempo de ejecución de aplicaciones según la frecuencia a la que se ejecutan 46

8.1. Principales servicios de EAR 47
8.2. Esquema básico de la interacción de los componentes de EAR 48
8.3. Fases internas de EARL 50
8.4. Funcionamiento de la política `MIN_TIME_TO_SOLUTION` 52
8.5. Funcionamiento de la política `MIN_ENERGY_TO_SOLUTION` 53
8.6. Configuración de nodos durante la ejecución de una aplicación 53

9.1. Esquema de la interacción del Global Manager con otros componentes . . . 56
9.2. Relación del periodo T1 y periodo T2 58
9.3. Formato de la representación del grafo de estados 58
9.4. Diagrama de estados de la configuración inicial del Global Manager 61
9.5. Atributos del fichero `ear.conf` 62
9.6. Experimento intensivo en cálculo gestionado por el EARGM inicial 63
9.7. Experimento intensivo en memoria gestionado por el EARGM inicial . . . 64

10.1. Promedio de la potencia consumida de las aplicaciones según la frecuencia a la que se ejecutan 70
10.2. Variación en la potencia en las aplicaciones BT-MZ.C, DGEMM, EP.D, LU-MZ.C, LU.C y SP-MZ.C 71
10.3. Variación en la potencia en las aplicaciones STREAM, UA.C, LU-MZ.D y DUMSES 72
10.4. Variación en la potencia en las aplicaciones respecto 2,4 GHz 73
10.5. Caracterización de las aplicaciones: CPI 74
10.6. Caracterización de las aplicaciones: Ancho de banda 75

10.7. Caracterización de las aplicaciones: Ancho de banda y CPI 76

11.1. Diagrama de estados del Global Manager con optimización ramp up 80

12.1. Exp. 20.1 con optimización ramp up ajustado al nivel WARNING2 90
12.2. Exp. 20.1 con optimización ramp up ajustado al nivel PANIC 91
12.3. Exp. 20.1 con optimización ramp down ajustado al nivel WARNING2 . . . 92
12.4. Exp. 20.1 con optimización ramp down ajustado al nivel PANIC 93
12.5. Comportamiento del experimento 20.5 sin acciones del EARGM 94
12.6. Exp. 20.5 con optimización ramp up ajustado al nivel WARNING2 96
12.7. Exp. 20.5 con optimización ramp up ajustado al nivel PANIC 97
12.8. Exp. 20.5 con optimización ramp down ajustado al nivel WARNING2 . . . 98
12.9. Exp. 20.5 con optimización ramp down ajustado al nivel PANIC 99
12.10Comportamiento del experimento 20.6 sin acciones del EARGM 100
12.11Exp. 20.6 con optimización ramp up ajustado al nivel WARNING2 102
12.12Exp. 20.6 con optimización ramp up ajustado al nivel PANIC 103
12.13Exp. 20.6 con optimización ramp down ajustado al nivel PANIC 104
12.14Comportamiento del experimento 20.7 sin acciones del EARGM 105
12.15Exp. 20.7 con optimización ramp up ajustado al nivel WARNING2 106
12.16Exp. 20.7 con optimización ramp up ajustado al nivel PANIC 107
12.17Exp. 20.7 con optimización ramp down ajustado al nivel WARNING2 . . . 108
12.18Exp. 20.7 con optimización ramp down ajustado al nivel PANIC 109

13.1. Ahorro energético e incremento en la eficiencia energética para un conjunto de aplicaciones utilizando `min_time` 114
13.2. Energía consumida y variación energética para BT-MZ.C y LU-MZ.D . . . 114

D.1. Comportamiento del experimento 4.1 sin acciones del EARGM 124
D.2. Exp. 4.1 con optimización ramp up ajustado al nivel WARNING1 125
D.3. Exp. 4.1 con optimización ramp up ajustado al nivel WARNING2 126
D.4. Exp. 4.1 con optimización ramp up ajustado al nivel PANIC 127
D.5. Exp. 4.1 con optimización ramp down ajustado al nivel WARNING2 128
D.6. Exp. 4.1 con optimización ramp down ajustado al nivel PANIC 129
D.7. Comportamiento del experimento 4.2 sin acciones del EARGM 130
D.8. Exp. 4.2 con optimización ramp up ajustado al nivel WARNING1 131
D.9. Exp. 4.2 con optimización ramp up ajustado al nivel WARNING2 132
D.10.Exp. 4.2 con optimización ramp up ajustado al nivel PANIC 133
D.11.Comportamiento del experimento 4.3 sin acciones del EARGM 134
D.12.Exp. 4.3 con optimización ramp up ajustado al nivel WARNING1 135
D.13.Exp. 4.3 con optimización ramp up ajustado al nivel WARNING2 136
D.14.Exp. 4.3 con optimización ramp up ajustado al nivel PANIC 137
D.15.Comportamiento del experimento 4.4 sin acciones del EARGM 138
D.16.Exp. 4.4 con optimización ramp up ajustado al nivel WARNING1 139
D.17.Exp. 4.4 con optimización ramp up ajustado al nivel WARNING2 140

D.18.Exp. 4.4 con optimización ramp up ajustado al nivel PANIC 141
D.19.Exp. 4.4 con optimización ramp down ajustado al nivel WARNING2 142
D.20.Exp. 4.4 con optimización ramp down ajustado al nivel PANIC 143
D.21.Comportamiento del experimento 4.5 sin acciones del EARGM 144
D.22.Exp. 4.5 con optimización ramp up ajustado al nivel WARNING1 145
D.23.Exp. 4.5 con optimización ramp up ajustado al nivel WARNING2 146
D.24.Exp. 4.5 con optimización ramp up ajustado al nivel PANIC 147
D.25.Exp. 4.5 con optimización ramp down ajustado al nivel WARNING2 148
D.26.Exp. 4.5 con optimización ramp down ajustado al nivel PANIC 149

E.1. Exp. 26.1 con optimización ramp up ajustado al nivel WARNING2 151
E.2. Exp. 26.1 con optimización ramp up ajustado al nivel PANIC 152
E.3. Exp. 20.2 con optimización ramp up ajustado al nivel PANIC 153
E.4. Exp. 20.2 con optimización ramp down ajustado al nivel PANIC 154
E.5. Comportamiento del experimento 20.3 sin acciones del EARGM 155
E.6. Exp. 20.3 con optimización ramp up ajustado al nivel WARNING1 156
E.7. Exp. 20.3 con optimización ramp up ajustado al nivel WARNING2 157
E.8. Exp. 20.3 con optimización ramp up ajustado al nivel PANIC 158
E.9. Comportamiento del experimento 20.4 sin acciones del EARGM 159
E.10.Exp. 20.4 con optimización ramp up ajustado al nivel WARNING1 160
E.11.Exp. 20.4 con optimización ramp up ajustado al nivel WARNING2 161
E.12.Exp. 20.4 con optimización ramp up ajustado al nivel PANIC 162
E.13.Exp. 20.4 con optimización ramp down ajustado al nivel WARNING2 . . . 163
E.14.Exp. 20.4 con optimización ramp down ajustado al nivel PANIC 164
E.15.Exp. 20.5 con optimización ramp up ajustado al nivel WARNING1 165
E.16.Exp. 20.6 con optimización ramp up ajustado al nivel WARNING1 166
E.17.Exp. 20.7 con optimización ramp up ajustado al nivel WARNING1 167

Índice de tablas

5.1. Tabla resumen de las tareas estimadas temporalmente 32

6.1. Tabla de costes en recursos hardware 37
6.2. Tabla de costes en recursos software 38
6.3. Tabla de horas dedicadas por rol según el grupo de tareas 39
6.4. Tabla de costes en recursos humanos 39
6.5. Tabla de costes indirectos 40
6.6. Tabla de imprevistos 40
6.7. Presupuesto total del proyecto 41

10.1. Configuración de las aplicaciones 69

12.1. Descripción de los test de la etapa de experimentación funcional 84
12.2. Descripción de los test de la etapa de experimentación de rendimiento . . . 84

Acrónimos

API Application Programming Interface

BD Base de datos

BDPO Bull Dynamic Power Optimization

BEO Bull Energy Optimizer

BSC-CNS Barcelona Supercomputing Center - Centro Nacional de Supercomputación

CAPMC Cray Advanced Platform Monitoring and Control

CPU Central Processing Unit

DAC Departamento de Arquitectura de Computadores

DVFS Dynamic Voltage and Frequency Scaling

DynAIS Dynamic Application Iterative Structure detection

EAR Energy Aware Runtime

EARD EAR Daemon

EARDBD EAR Database Manager

EARGM EAR Global Manager

EARL EAR Library

EARplug EAR SLURM plugin

GEOPM Global Extensible Open Power Manager

HPC High Performance Computing

LRZ Leibniz Supercomputing Center

MPI Message Passing Interface

NASA National Aeronautics and Space Administration

SLURM Simple Linux Utility for Resource Management

TFG Trabajo de Fin de Grado

VPN Virtual Private Network

XCC Lenovo XClarity Controller

Capítulo 1

Introducción

Este proyecto se trata de un Trabajo de Fin de Grado (TFG) que se ha realizado en el Departamento de Arquitectura de Computadores (DAC), gracias a una colaboración existente con el Barcelona Supercomputing Center y Lenovo para el desarrollo del software EAR (Energy Aware Rutine). En las siguientes secciones pertenecientes a este capítulo se muestra el contexto en el cual nos encontramos, cual es el problema al que se pretende dar una solución y, finalmente, quienes son los actores que están involucrados en este trabajo.

1.1. Contexto

Energy Aware Rutine (EAR) es un software de sistema que nace de la colaboración del centro de investigación Barcelona Supercomputing Center (BSC-CNS) junto con Lenovo. Se trata de un software que ofrece una gestión energética para supercomputadores que utilizan SLURM (gestor de cargas de trabajo) y MPI (estándar de comunicación para memoria distribuida) [1]. Esto es, optimiza y monitoriza la energía tanto a nivel de nodo como de *cluster* en general.

Lenovo es una empresa multinacional tecnológica dedicada a la industria del hardware. Como podemos ver en la lista TOP500 [2], muchos de los centros de computación de altas prestaciones (*High Performance Computing*, HPC) trabajan con máquinas fabricadas por esta compañía. En la actualidad, las empresas están concienciadas en el consumo responsable de la energía ya que uno de los grandes problemas para los centros HPC es el gran gasto en potencia y refrigeración, y para ello se desarrollan nuevas tecnologías.

Figura 1.1: Logo de EAR. [3]

Un ejemplo de Lenovo sería Lenovo Neptune[1], la cual es una técnica que refrigera las máquinas mediante agua (recibe el nombre de *liquid cooling*) y es más eficiente que la clásica refrigeración por aire. Sin embargo, otras empresas no cuentan con un software que complemente todos los esfuerzos realizados a nivel hardware para ofrecer sistemas energéticamente eficientes, como es el caso de EAR. De hecho, curiosamente, muchos centros HPC no llevan una contabilidad precisa sobre el uso energético y de refrigeración que gastan.

EAR se trata de un *framework* que cuenta con una infraestructura robusta y bien definida, compuesta por varios componentes que cooperan entre sí y son los siguientes [1]:

- *Daemon* EAR (**EARD**): Un *daemon* es un proceso que se ejecuta en segundo plano sin estar directamente controlado por un usuario. En este caso, hay un *daemon* de EAR ejecutándose por cada nodo que hay en el sistema, realizando así un servicio de monitoreo energético en los nodos. Estos datos serán reportados directamente a la base de datos de EAR o a través de un *daemon* para la base de datos (en caso de estar instalado).
- Global Manager de EAR (**EARGM**): Su función es controlar la potencia y energía consumidas durante un periodo de tiempo determinado y aplicar unas políticas globales (por ejemplo decrementar la frecuencia) para garantizar un consumo máximo de la energía durante ese periodo. Sólo puede haber 1 componente EARGM en una misma máquina.
- Librería EAR (**EARL**): Es el núcleo del software EAR y se trata de una librería en *runtime* que pretende optimizar la energía de las aplicaciones, controlar la energía coordinadamente con EARGM y monitorizar la energía.
- EAR DB Manager (**EARDBD**): Este componente se encarga de almacenar los datos que recibe de EARD y EARL, y se encarga de reportarlos a la base de datos de EAR. En caso de querer instalar este servicio para la BD y tener un *cluster* suficiente grande, se recomienda 'dividir' la máquina en varias zonas para asignarle a cada una un *daemon*, reduciendo así el número de conexiones a la BD.
- Plugin SLURM de EAR (**EARplug**): Su función es cargar la librería EAR y hacer que el envío de trabajos por medio SLURM sea transparente a los usuarios y fácil para los administradores de sistemas. [4]

Sin embargo, en caso de no querer hacer uso de todas las funcionalidades que ofrece EAR no es necesario tener instalados todos sus componentes. Por ejemplo, si únicamente se quiere llevar un *accounting* energético, bastaría con instalar EARplug y EARD. En cambio, si se desea llevar a cabo una monitorización global del sistema, EARGM pasa a ser necesario. EARL se deberá instalar si se busca optimización energética. [3]

[1]Para más información: `https://www.lenovo.com/es/es/data-center/Lenovo-Neptune/p/neptune`

A continuación, vemos esquematizado el diseño de EAR con las conexiones entre sus diferentes componentes.

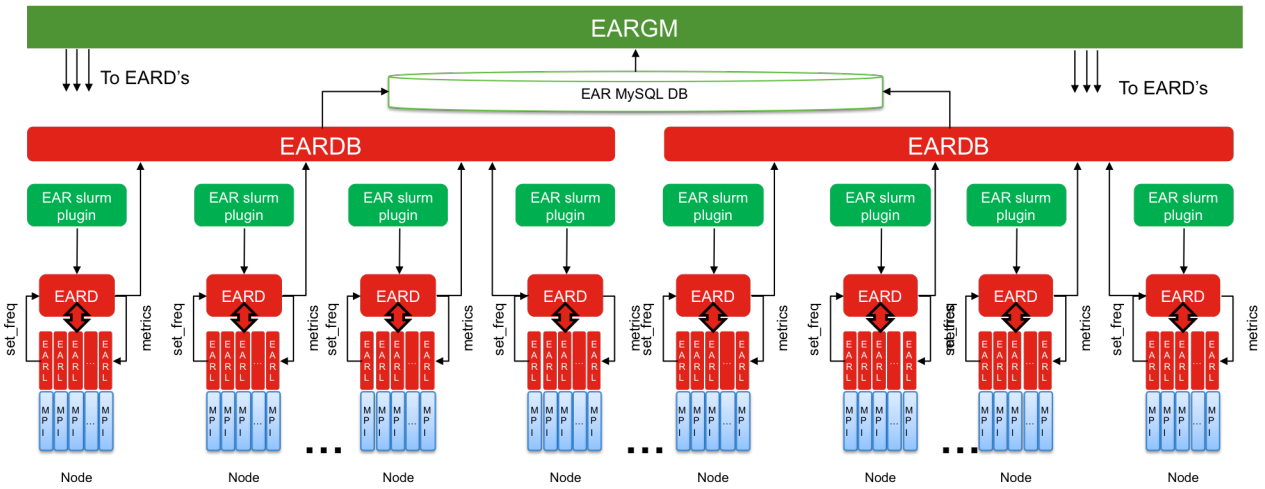


Figura 1.2: Esquema de las relaciones entre los distintos componentes en EAR. [5]

En resumen, vemos que EAR ofrece, principalmente, los siguientes servicios [6]:

- Optimización energética.
- *Accounting* de las aplicaciones.
- Análisis y optimización de las aplicaciones.
- Monitorización del sistema.
- Control energético.

Por lo tanto, EAR se trata de un software fácil de usar con soluciones eficientes para optimizar la energía, dónde todos los componentes están debidamente coordinados y es de libre distribución[2].

Por último, se puede afirmar que EAR se trata de un proyecto robusto que funciona muy bien y está dando buenos resultados, ya que se encuentra instalado y en ejecución en el supercomputador SuperMUC-NG, el cual es considerado el noveno computador más potente del mundo (según la lista TOP500 [2]). Este superordenador se encuentra en el centro de investigación Leibniz Supercomputing Centre (LRZ), situado en Garching (cerca de Munich, en Alemania).

[2]Disponible contactando con: ear-support@bsc.es

Figura 1.3: Supercomputador SuperMUC-NG. [7]

1.2. Formulación del problema

El *framework* EAR se encuentra en fase de desarrollo y cada vez abarca más y más funcionalidades. Como se ha comentado en la sección anterior, el objetivo de EARGM es garantizar que se consume un máximo energético en un periodo de tiempo determinado. Sin embargo, este componente puede operar en 2 modos distintos: modo de monitorización o modo automático.

Si se opta por el modo de monitorización, se limitará a reportar a la base de datos sobre el estado global del sistema. Por otro lado, si se escoge el segundo modo será capaz de adaptar la configuración del sistema automáticamente a los requerimientos que considere necesarios.

El modo de monitorización se encuentra cubierto y en funcionamiento, ya que en un inicio se acordó que EARGM sólo monitorizaría por el momento. No obstante, se quiere poder realizar un control dinámico de la energía (modo automático). Por lo tanto, este trabajo se va centrar en el componente EARGM en el que se buscará dotarle de capacidades para que pueda tomar decisiones sobre los ajustes en las configuraciones por sí solo. Actualmente, sobre el modo automático únicamente hay un prototipo de un diseño inicial. Por lo tanto, se llevará a cabo una evaluación inicial para detectar qué casos hay que optimizar o incluir para luego diseñar e implementar un nuevo grafo de estados y evaluarlo.

1.3. Actores implicados

Este trabajo tiene involucradas a una gran cantidad de personas y entidades que velan por lograr el mayor éxito en este nuevo software y hay una serie de actores implicados o bien por ser a quienes irá dirigido o bien serán quienes lo usen o bien se beneficiarán de él. A continuación, describo a estos agentes:

- **Barcelona Supercomputing Center:** Beneficiará al centro de investigación en ser pionero en la realización de este software. Dentro del centro, beneficiará en concreto a todos los científicos en la línea de investigación dedicada a la ejecución de EAR, dentro del grupo System software for energy management in HPC (perteneciente al departamento de Computer Sciences de la empresa) ya que implicará un avance en su desarrollo.
- **Lenovo Group:** Beneficiará a la empresa en ser los únicos en el mercado con un software de tales características para sus máquinas en los centros HPC.
- **Colectivo HPC:** El proyecto puede ser de interés para toda persona relacionada con el colectivo HPC (por ejemplo administradores de sistemas e investigadores). Esto es debido a que se conseguiría ser más eficiente a la hora de gastar recursos energéticos, que a la larga puede suponer un ahorro.

Por otro lado, la sociedad en general se verá beneficiada indirectamente ya que se llevará a cabo un consumo responsable por parte de estos centros HPC, lo cual producirá un impacto en el ahorro energético.

Para finalizar, comentar que obviamente también beneficia a la investigadora (en este caso, la estudiante) ya que puede realizar un estudio sobre un tema de su interés, y también a la directora de proyecto como parte interesada.

Capítulo 2

Justificación

EAR es un software de sistema cuyos componentes ofrecen diferentes servicios (optimización y monitorización principalmente) y a diferentes granularidades (nodo y sistema). EAR es mucho más completo que otras propuestas ya que todos los componentes están coordinados entre sí, a diferencia de otros sistemas que solo proporcionan monitorización, o solamente optimización, o no presentan una visión global del sistema.

Por lo tanto, no hay ninguna otra propuesta que cumpla con las funcionalidades del EARGM o tenga un componente similar a éste. Sin embargo, se describen éstas alternativas relacionadas que podemos encontrar.

A nivel *cluster*, algunos fabricantes como Lenovo ofrecen programas específicos que no acaban de ajustarse a lo que constituye un servicio de sistema y, además, suelen utilizar comandos complejos, redes alternativas, usuarios diferentes a los de Unix/Linux, etc. En el caso de Lenovo el programa es Lenovo XClarity Controller (XCC) y se ejecuta totalmente al margen del resto del sistema en un procesador diferente [8].

Por otro lado, encontramos otros programas que ofrecen servicios parcialmente similares a EAR pero no incluyen el control de la energía a nivel *cluster*. Estos serían:

- GEOPM es un software de optimización de energía desarrollado por Intel que ofrece optimización a nivel de aplicación pero no incluye ninguna solución a nivel de *cluster* [9]. Compite con EAR en algunos de sus componentes pero no incluye esta visión global del sistema.
- Bull es otro de los fabricantes más importantes de sistemas HPC y ofrece un par de componentes llamados BEO (Bull Energy Optimizer) y BDPO (Bull Dynamic Power Optimization) que optimizan la energía. Estos dos componentes proporcionan monitorización de energía y potencia a nivel de nodo y optimización a nivel de aplicación [10]. Sin embargo, igual que en el caso anterior, tampoco ofrecen ningún servicio a nivel *cluster*.

- Cray ofrece un programa llamado CAPMC que aporta monitorización y control de sistemas Cray. Es un programa que se suele utilizar integrado con el sistema de colas (por ejemplo SLURM) para utilizar la capacidad que ofrecen estos sistemas de agregar la información de utilización de recursos y centralizarlo. Este programa ofrece la funcionalidad básica para monitorizar y modificar la configuración y estado del sistema [11]. No obstante, la lógica para decidir sobre cómo debe modificarse esta configuración se suele apoyar en otros componentes. Por ejemplo, Cray ofrece dos *plugins* para el sistema de colas SLURM: uno para monitorización de la energía y otro para el control de límites de potencia del sistema [12]. Estos componentes junto con CAPMC garantizan que la máxima potencia del sistema no sobrepase la establecida en la configuración. Por contra, al depender totalmente del sistema de colas, no es fácilmente modificable ni extensible.

Finalmente, vemos que lo más similar a EARGM sería la opción de SLURM con el *plugin* de gestión de potencia que ofrece Cray. No obstante, solo actúa a nivel de potencia y tampoco es lo mismo ya que, a diferencia de EARGM, no tiene relación con ningún componente que se encuentre en los nodos.

Capítulo 3

Alcance del proyecto

Una vez en contexto y formulado el problema, antes de empezar a trabajar, es importante delimitar y tener claros cuáles van a ser los objetivos que se deben alcanzar, qué requerimientos e identificar los posibles riesgos y obstáculos que pueden ocurrir durante el desarrollo del proyecto.

3.1. Objetivos

Dado que el objetivo principal es ajustar el consumo energético a lo especificado por el administrador de sistemas, este TFG se ha centrado principalmente en los siguientes puntos:

- **Evaluación del estado del servicio de EARGM:** Recordemos que el componente EARGM puede funcionar en 2 modos: monitorizando o realizando un control energético dinámico. El primero de ellos está probado desde el inicio que funcionaba correctamente. Sin embargo, el último no estaba cubierto ya que solamente había un diseño inicial y era un prototipo, por lo que se ha realizado una evaluación previa sobre el estado en que se encuentra.

- **Estudio sobre el impacto que puede provocar modificar la configuración de las políticas en función del tipo de cada aplicación:** En primer lugar se realizará un estudio con el objetivo de ver la caracterización que tienen las aplicaciones. Es decir, si es intensiva en CPU, en memoria o una combinación de ambas. Una vez realizado el estudio, se llevarán a cabo unos experimentos para observar qué ocurre si realizamos una serie de modificaciones en las políticas (por ejemplo modificar la frecuencia, establecer un nuevo límite para la potencia máxima, etc.) según las características que presentan estas aplicaciones.

- **Configuración automática del EARGM:** Inicialmente, las acciones que realizaba el componente estaban *hardcoded*, en otras palabras, predeterminadas. Para poder dotar al EARGM de inteligencia y ofrecer más flexibilidad, se han seguido los siguientes subobjetivos:
  - Definición de los casos de uso en el EARGM: Se han explorado todas las situaciones en que podía encontrarse y cómo las debe gestionar. Una vez estudiado, se ha realizado una especificación del diagrama de grafos de estados.
  - Diseño e implementación del grafo de estados del EARGM.
  - Evaluación y análisis de los resultados obtenidos del EARGM con la ejecución de varios *workloads*: En este punto en que ya se tienen todos los casos cubiertos se ha realizado una extensa evaluación del componente en que, finalmente, se ha determinado que es conveniente extender la API.

3.2. Requerimientos

Este proyecto identifica un conjunto de requerimientos funcionales y no funcionales, y son los siguientes:

Requerimientos funcionales:

- Ser capaz de identificar las características de las aplicaciones que pueden influir en la gestión de la energía.
- Proporcionar los ajustes de energía en función de lo especificado por el administrador de sistemas.
- El componente EARGM debe estar dotado de inteligencia para tomar decisiones a la hora de ajustar las configuraciones de las políticas.
- Identificar dinámicamente los valores que deben aplicarse a los parámetros de las políticas en función de la evolución del consumo y la tipología de las aplicaciones.

Requerimientos no funcionales:

- Ofrecer una gestión dinámica de la energía de forma transparente al usuario.
- La gestión del EARGM debe poder ser escalable. Es decir, debe seguir funcionando tanto en máquinas con 4 nodos como con 6000.
- Los datos reportados por el EARGM deben ser fiables y disponibles. En otras palabras, no se pueden perder.
- Programar empleando un estilo leíble y entendedor.

Capítulo 4

Metodología y rigor

En este capítulo se describe el método de trabajo escogido para la realización del proyecto, cómo se ha realizado el seguimiento sobre la evolución en el estudio y cómo se han verificado los resultados obtenidos.

4.1. Metodología de trabajo

La metodología de trabajo escogida se ha basado en el método Agile [13], el cual consiste en marcar unos objetivos en periodos de tiempo cortos. De esta manera, obtenemos una flexibilidad con la que se puede hacer frente a posibles imprevistos que puedan surgir y poder así adaptar tales objetivos a las condiciones en que nos encontremos. Otro beneficio que nos aporta esta metodología es el hecho de poder rectificar rápidamente en caso de estar avanzando por una vía equivocada. De este modo, conseguimos minimizar el tiempo perdido.

4.2. Herramientas de seguimiento

Para llevar un control del seguimiento del trabajo se ha utilizado la herramienta GitHub[1], la cual nos ha permitido administrar los cambios efectuados en el código, aparte también de poder disponer de una copia de seguridad en caso de pérdida o corrupción del código, o por necesidad de volver a un cierto punto anterior. Por otro lado, el resto del equipo también podía consultar en cualquier momento el estado en que se encontraba el código del proyecto.

También se han llevado a cabo reuniones semanales con la directora del proyecto para verificar que se estaba desarrollando todo correctamente y en los plazos estimados. Finalmente,

[1] GitHub: `https://github.com/`

anoté en un diario por mi cuenta todos los cambios realizados diariamente, con el fin de no olvidar detalles importantes.

4.3. Herramientas de validación

Gracias a la metodología ágil, una de las herramientas de validación se ha llevado a cabo en las reuniones semanales con la directora del proyecto. En estas reuniones se verificó que los resultados no hubiesen sufrido ninguna interferencia que los pudiera distorsionar, identificando así posibles errores cometidos, y se planificaron las tareas a realizar para la siguiente semana. Sin embargo, inicialmente estaba previsto que las reuniones fueran presenciales pero se realizaron de forma telemática debido a la situación por alerta sanitaria, a través de Skype[2]. Este programa nos permitió realizar llamadas y compartir nuestras pantallas durante las reuniones, permitiéndonos seguir verificando los resultados y resolviendo dudas.

Por otro lado, se utilizaron una serie de *benchmarks* y aplicaciones reales en un conjunto de nodos reservados para garantizar que el consumo de potencia/energía correspondía solamente a los trabajos que nosotras empezábamos.

[2]Skype: `https://www.skype.com/es/`

Capítulo 5

Planificación temporal

5.1. Descripción de tareas

Este TFG se inició en diciembre de 2019 y se va a defender en el turno de junio de 2020, por lo que ha tenido una duración de unos seis meses de trabajo.

A continuación, cada una de las siguientes secciones muestra los diferentes grupos de tareas que han formado parte de este trabajo.

5.1.1. Gestión del proyecto

En este grupo se han llevado a cabo las tareas relacionadas con la gestión del proyecto, fundamentales para su desarrollo.

- **T1.1 - Reuniones con la directora:** Tal y como se ha comentado en el capítulo 4, se han realizado reuniones semanales con la directora del proyecto, que mayoritariamente fueron telemáticas. En estas reuniones se ha realizado un control de seguimiento sobre el trabajo efectuado durante la semana y se han resuelto las dudas surgidas. También se planificaron los objetivos para la siguiente semana. Para cada reunión se calculó una duración de, aproximadamente, 1 hora y cuarto. Por lo tanto, se estimaron unas 30 horas en total.
- **T1.2 - Elaboración de la documentación:** Elaboración tanto de la memoria final del proyecto como de los distintos informes solicitados. Se ha realizado en paralelo con el desarrollo del proyecto y se estimó una dedicación de unas 70 horas de trabajo.
- **T1.3 - Especificación del alcance del proyecto:** Es una parte fundamental del trabajo en la que se identifica el contexto en el que se sitúa el TFG, el porqué del trabajo, los objetivos que se pretendían alcanzar y cómo se iba a realizar. La duración de la tarea se estimó en unas 25 horas.

- **T1.4 - Planificación:** Una vez definido el alcance, se planificaron las tareas a realizar. Para ello se describieron y planificaron con los recursos necesarios según la duración prevista para cada una de ellas y los requisitos entre ellas. Se estimó una dedicación de 10 horas.
- **T1.5 - Elaboración del presupuesto:** En cuanto estuvo lista la planificación, se realizó una estimación del coste económico que suponía la elaboración del proyecto. Para su ejecución se estimaron unas 5 horas.
- **T1.6 - Análisis e informe sobre la sostenibilidad del proyecto:** Elaboración de un informe de sostenibilidad sobre el impacto que ha supuesto el proyecto en la huella ecológica y el compromiso social. Se comenzó una vez definido el alcance del trabajo y se ha terminado al finalizar el proyecto. Para ello se previeron unas 5 horas.
- **T1.7 - Preparación de la defensa del TFG:** Preparación previa para la lectura del trabajo. Se realizará una vez terminada la memoria y consistirá en preparar unas transparencias y practicar la presentación. Se estima una dedicación de unas 10 horas.

5.1.2. Familiarización con el entorno

Antes de realizar la gestión del proyecto ya se empezó a trabajar en él aunque la dedicación fue mucho más relajada debido, entre otros, a la carga de trabajo continua por las asignaturas de la universidad que estaba cursando en ese momento. Las tareas previas que se empezaron son las pertenecientes a este grupo.

- **T2.1 - Acceso a la VPN y a Lenox:** Para poder trabajar es necesario conseguir acceso a una red corporativa (*Virtual Private Network*, VPN) de Lenovo, ya que si no es así no es posible acceder a sus máquinas. Una vez hecho, el siguiente paso es conseguir acceso al *cluster* Lenox. Para ello esperamos a que el administrador de sistemas de Lenox creara una cuenta de usuario y después se comprobó que se pudiera acceder correctamente. El acceso a la VPN se ha hecho por medio de un programa llamado GlobalProtect y el acceso a Lenox, a través del comando para conexiones remotas *ssh*. Hasta que el administrador de sistemas diera de alta el perfil de usuario transcurrieron entre 1 y 2 semanas de espera. Una vez hecho se necesitaron unas 2 horas para configurar y acceder a la VPN y al *cluster*.
- **T2.2 - Configuración del entorno:** Configuración del perfil de usuario con el entorno de trabajo y las herramientas necesarias, como el compilador ICC. Hasta tenerlo configurado correctamente se estimaron unas 18 horas de trabajo.
- **T2.3 - Familiarización con SLURM:** Debido a la no experiencia previamente en un sistema de colas como SLURM, fue necesario dedicar un tiempo a aprender a enviar *scripts* correctamente para poder luego realizar el trabajo sin errores en la configuración de estos *scripts*. También aprendí a utilizar programas en un entorno MPI. Se estimaron unas 20 horas de trabajo.

- **T2.4 - Familiarización con EAR:** Lectura de la documentación del software EAR y búsqueda de funcionalidades por parte de la directora con el fin de aprender e identificar dónde se llevan a cabo tales peticiones en el software. Para ello se previeron 20 horas de trabajo.

5.1.3. Evaluación del estado de servicio EARGM

Ha consistido en una única tarea (T3) en la que se ha evaluado el estado en que se encontraba el componente EARGM inicialmente. Recordemos que únicamente se encontraba un prototipo de un diseño inicial. Para esta tarea se estimaron unas 20 horas.

5.1.4. Modificación en la configuración

- **T4.1 - Análisis de aplicaciones:** Se han analizado distintas aplicaciones con el fin de estudiar su variabilidad en la potencia y las características que presentan. Es decir, si se trata de aplicaciones intensivas en CPU, en memoria o una combinación de ambas. Para ello se han realizado múltiples pruebas según la aplicación, el número de nodos utilizados y el número de iteraciones (para más o menos carga). Se han anotado todos los datos obtenidos (ancho de banda, energía, potencia, número de ciclos por instrucción, tiempo total de ejecución, etc.) en una hoja de cálculo de LibreOffice y se han realizado unas gráficas. Esta tarea se puede empezar a realizar una vez familiarizados con el entorno SLURM y se prevén 40 horas de trabajo. Las aplicaciones están programadas en Fortran, por lo que ha sido necesario un compilador IFORT.
- **T4.2 - Modificación en las políticas del EARGM:** Una vez evaluado el estado en que se encuentra EARGM y se han analizado los comportamientos de distintas aplicaciones, se pueden aplicar modificaciones a las políticas para EARGM. Esto es, como reducir la frecuencia, establecer un nuevo límite para la potencia máxima, etc. Para ello se deberán realizar numerosos cambios y se estima una dedicación de unas 50 horas.

5.1.5. Evaluación y análisis de resultados

- **T5.1 - Definición de los casos de uso:** Estudio sobre todos los casos de uso posibles para integrarlos en el nuevo diseño del diagrama de estados del EARGM. Esta tarea la podremos realizar una vez hayamos realizado el estudio sobre la modificación en las políticas y está previsto unas 30 horas de duración.
- **T5.2 - Especificación del diagrama de estados:** Una vez identificados todos los casos de uso, se procederá a realizar una especificación del diagrama de estados de EARGM. Se prevén unas 30 horas de trabajo.

- **T5.3 - Diseño e implementación del diagrama de estados:** Cuando se tenga la especificación del diagrama, será el momento de implementarla. Para ello se estiman 50 horas de trabajo.
- **T5.4 - Diseño de experimentos:** Paralelamente a la implementación del diagrama de estados (una vez especificado el diagrama) se podrán empezar a diseñar experimentos con los que, posteriormente, se evaluarán los resultados obtenidos con el nuevo comportamiento implementado. Se prevén unas 25 horas de dedicación.
- **T5.5 - Evaluación de los resultados obtenidos:** Cuando estén listos los experimentos y la implementación, se evaluarán los resultados con la nueva implementación. Se estima que serán necesarias unas 50 horas de dedicación.
- **T5.6 - Propuesta de modificaciones:** En función de los resultados obtenidos se propondrán una serie de modificaciones para EARGM, por ejemplo para la API o las políticas. Se han previsto unas 40 horas de trabajo.

5.2. Diagrama de Gantt

Para consultar el diagrama de Gantt, ver el apéndice B.

5.3. Distribución de trabajo

A lo largo de este capítulo se ha visto la duración prevista por cada tarea a medida que se describía. Por lo tanto, podemos resumir (como vemos en la siguiente Tabla 5.1) las tareas según su duración prevista, sus dependencias y los recursos necesarios específicos para poder llevarlas a cabo.

Cabe comentar que, obviamente, el ordenador portátil es un recurso necesario para todas las tareas y por eso no se ha especificado explícitamente. Por otro lado, a partir del tercer grupo de tareas, cuando ya se tiene acceso al *cluster*, al especificarse Lenox como recurso también se está especificando implícitamente el uso de GlobalProtect.

Finalmente, es importante comentar que no han habido variaciones significativas, por lo que ni la planificación ni la duración de las tareas previstas se ha visto alterada.

ID	Tarea	Duración	Depend.	Recursos
1	**Gestión del proyecto**	**155 h.**		
T1.1	Reuniones con la directora	30 h.		Skype, Lenox, GlobalProtect
T1.2	Elaboración documentación	70 h.		LaTex
T1.3	Especificación del alcance	25 h.		LaTex
T1.4	Planificación	10 h.	T1.3	LaTex
T1.5	Elaboración presupuesto	5 h.	T1.4	LaTex, LibreOffice
T1.6	Análisis e informe sostenibilidad	5 h.	T1.3	LaTex
T1.7	Preparación de la defensa	10 h.	T1.2	LaTex
2	**Familiarización con el entorno**	**60 h.**		
T2.1	Acceso VPN y Lenox	2 h.		GlobalProtect, Lenox
T2.2	Configuración del entorno	18 h.	T2.1	Lenox, Vim
T2.3	Familiarización SLURM	20 h.	T2.2	Lenox, SLURM Vim
T2.4	Familiarización EAR	20 h.		
3	**Evaluación estado servicio**	**20 h.**		
T3	Evaluación estado servicio EARGM	20 h.	T2.[3,4]	Lenox, Vim, ICC
4	**Modificación configuración**	**90 h.**		
T4.1	Análisis de aplicaciones	40 h.	T2.3	LibreOffice, MPI, SLURM, IFORT, Vim, Lenox
T4.2	Modificación en las políticas	50 h.	T3, T4.1	Lenox, ICC, Vim
5	**Evaluación y análisis resultados**	**225 h.**		
T5.1	Definición casos de uso	30 h.	T4.2	
T5.2	Especificación diagrama	30 h.	T5.1	
T5.3	Diseño e implementación diagrama	50 h.	T5.2	Lenox, Vim, ICC
T5.4	Diseño de experimentos	25 h.	T5.2	Lenox, Vim, MPI, SLURM
T5.5	Evaluación resultados	50 h.	T5.[3,4]	LibreOffice, MPI, SLURM, Lenox, Vim, Paraver
T5.6	Propuesta de modificaciones	40 h.	T5.5	Lenox, Vim, ICC
Total:		**550 h.**		

Tabla 5.1: Tabla resumen de las tareas según su identificador, nombre, duración estimada, dependencias y recursos específicos.

5.4. Recursos

Para realizar el proyecto se han destinado unos gastos generales ya que es necesario un espacio para trabajar (equipado con mobiliario), luz, conexión a Internet y gastos debido al transporte hacia el lugar de trabajo. Por otro lado, a continuación se listan los recursos hardware, software y humanos necesarios.

5.4.1. Recursos hardware

- **Ordenador portátil Lenovo ThinkPad T480s:** con Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz y 8GB de RAM. Ordenador desde el cuál se ha desarrollado el proyecto y se han documentado los informes y la memoria.
- **Monitor ThinkVision, teclado y ratón:** Periféricos para una mayor comodidad para trabajar.
- ***Cluster* Lenox:** con Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz. Infraestructura donde se ha llevado a cabo el desarrollo del proyecto y la realización de las pruebas.

5.4.2. Recursos software

- **GlobalProtect:** Acceso a la VPN de las máquinas de Lenovo.
- **Ubuntu 18.04:** Sistema operativo que permite conexiones remotas *ssh*.
- **Vim:** Editor de texto para programar.
- **LaTex:** Editor de texto para redactar documentos.
- **Compilador ICC**: Compilador de Intel para el proyecto.
- **GitHub:** Sistema para control de versiones, para copias de seguridad y para compartir el código con el resto del grupo.
- **LibreOffice:** Para almacenar una gran cantidad de datos y realizar gráficas.
- **SLURM:** Sistema de colas para ejecutar las cargas de trabajo.
- **Entorno MPI:** Entorno de desarrollo y ejecución de aplicaciones MPI estándar.
- **Compilador IFORT:** Compilador para aplicaciones escritas en Fortran.
- **Paraver:** Herramienta de análisis y visualización de trazas.
- **Skype:** Para realizar las reuniones semanales de forma remota.

5.4.3. Recursos humanos

Los principales actores que han participado en este trabajo son:

- **Estudiante:** Encargada de desarrollar y documentar el proyecto.
- **Directora:** Encargada de dirigir y guiar durante el proyecto.

Por otro lado, también han colaborado indirectamente compañeros del grupo de investigación como soporte.

5.5. Gestión del riesgo

En todo proyecto siempre pueden aparecer complicaciones en cualquier momento. Para ello se debe estar preparado y tener un plan de actuación en caso de ocurrir. A continuación se exponen las situaciones de riesgo y obstáculos que se consideraron al inicio del proyecto que podrían suceder y conllevarían un impacto negativo en el resultado de este trabajo, así como su ralentización. En la prevención de estos riesgos hemos evaluado el impacto que podrían tener sobre la duración del TFG, cómo se resolverían y, en caso de ser necesario, los recursos adicionales.

- **Fallos en la planificación:** Podría darse el caso de una estimación insuficiente en cuanto a la duración de algunas tareas. En caso de que ocurriera, se elevaría el número de horas con tal de poder reajustarnos a la planificación prevista. Sin embargo, si no fuera suficiente se deberán redefinir otras tareas. Por lo tanto, tiene un impacto bajo y un riesgo medio-bajo.
- **Errores de programación:** Es humano cometer errores, y en la programación no es una excepción. Es recomendable hacer pruebas pequeñas aparte simulando el entorno de la funcionalidad que se quiere desarrollar o hacer uso de *debug*.
- **Fallos en EAR no detectados previamente:** Al ser un software nuevo en fase de desarrollo y que abarca muchas funcionalidades, cabe la posibilidad de encontrar algún error que deba ser solventado antes. Si ocurriera, serían errores muy leves que quizá alarguen un poco más de lo previsto la duración de una determinada tarea, o se deba pausar una tarea para solucionarlos. En algún caso podría ser necesaria la ayuda de la directora o de otro compañero del grupo de investigación para solucionarlo. Tendría un impacto pequeño y el riesgo de que suceda es medio-bajo.
- **Falta de experiencia en entornos HPC y software EAR:** Supone un tiempo de adaptación en este nuevo entorno y aprendizaje del funcionamiento del software.
- **Retirada del acceso a la infraestructura de realización del proyecto:** Este TFG se desarrolla en un *cluster*, llamado Lenox, que pertenece a Lenovo. En caso de retirarse el acceso, conllevaría dificultades para seguir adelante con el trabajo ya que

es imprescindible una infraestructura como Lenox para poder realizar las debidas evaluaciones y, entonces se debería buscar una máquina alternativa pero es difícil medir el tiempo que puede transcurrir hasta tener acceso a una nueva. Además, se debería adaptar el trabajo a las nuevas características *hardware* en las que nos encontráramos. No obstante, la probabilidad de que ocurra es prácticamente nula (riesgo muy bajo) ya que Lenovo garantiza el acceso a recursos.

- **Imprevistos en la infraestructura de realización del proyecto:** Por un lado, podría ocurrir que Lenox deje de estar operativo temporalmente debido a averías tales como problemas por fallos en los componentes o simplemente por mantenimiento. En caso de ser por éste último, tendría un impacto bajo ya que suele durar unas pocas horas. Por contra, si sufriera fallos en el *hardware* tendría un impacto terrible, igual que en el caso de la retirada de la cuenta en la máquina, y se resolvería de la misma forma. No obstante, la probabilidad de que suceda es muy baja ya que para ello se van realizando labores de mantenimiento. Finalmente, también podría darse el caso de hacer una larga espera hasta conseguir recursos en Lenox porque otros investigadores han ocupado antes la máquina y también deben hacer sus experimentos. En consecuencia se dedicaría más tiempo de lo previsto a una tarea, que podría conducir a fallos en la planificación. Sin embargo, la probabilidad de que ocurra es baja aunque puede suceder en el momento más inoportuno.

Por otro lado, en el informe de seguimiento se hizo mención a un nuevo riesgo detectado que también podía afectar a la planificación y que afortunadamente no sucedió:

- **Infección causada por el SARS-CoV-2:** Dado que el coronavirus se contagia muy fácilmente nos encontramos en situación por alerta sanitaria y existe el riesgo de caer enfermo tanto por parte de la estudiante como de la directora. Para minimizar el riesgo se realizaron las reuniones de forma remota. Sin embargo, al tratarse de una pandemia está fuera de nuestro control y no contábamos con un plan de contención, por lo que el riesgo es indeterminado y el efecto que pudiera tener en el trabajo es impredecible.

Finalmente, durante el desarrollo del proyecto se produjo un incidente importante:

- **Problemas de acceso a Lenox:** A pesar de haber una probabilidad insignificante de que sucediera, hubieron problemas de acceso al *cluster* debido a fallos en la alimentación, por lo que se estuvo un par de semanas sin poder acceder.

  No obstante, al volver a tener acceso a la máquina, el problema no estaba totalmente solucionado, por lo que el administrador decidió mantener apagados la mitad de los nodos. En caso de querer utilizar algún nodo particular que no estuviera activo se debería encender, por lo que las pruebas empezaron a contar con un *delay* de encendido.

Se desconocía cuándo se volvería a tener acceso. Sin embargo, quisimos ser previsoras desde el inicio e intentamos comenzar lo antes posible todos los experimentos por si hubiese un incidente. A pesar de esta circunstancia la planificación no se vio alterada ya que se aprovechó para adelantar la documentación de la memoria y dejar planteadas las ideas de las siguientes tareas una vez se volviera a tener acceso. Por otro lado, en caso de no haber podido realizar los debidos experimentos se hubiese tenido que posponer la defensa del proyecto para la convocatoria extraordinaria del mes de octubre.

Capítulo 6

Gestión económica

6.1. Costes del proyecto

6.1.1. Costes directos

Por costes directos entendemos aquellos que están claramente identificados y relacionados con una actividad o producto. Es decir, son aquellos recursos necesarios para llevar a cabo una actividad. Por lo tanto, en este proyecto se distinguen costes en hardware, software y recursos humanos.

Seguidamente, se muestra por medio de tablas los distintos elementos que podemos encontrar en cada tipo de recurso.

Recursos hardware:

A continuación se especifican los recursos hardware necesarios para llevar a cabo el proyecto con las respectivas amortizaciones por hora trabajada que suponen. Para el *cluster* Lenox no es posible cuantificar ni su coste ni su vida útil ni poder aproximar su amortización por hora ya que no se cuenta con la información necesaria.

Hardware	**Uds.**	**Coste/ud.**	**Total**	**Vida útil**	**Amortización**
ThinkPad T480s	1	1681 €	1681 €	5 años	0,18678 €/h.
Monitor ThinkVision	1	179 €	179 €	4 años	0,02486 €/h.
Teclado Logitech	1	15 €	15 €	4 años	0,00208 €/h.
Ratón Logitech	1	17 €	17 €	4 años	0,00236 €/h.
Cluster Lenox	1	-	-	-	-
Total:	-	-	**1892 €**	-	**0,21608 €/h.**

Tabla 6.1: Tabla de costes en recursos hardware.

Dado que el proyecto ha tenido una duración de 550 horas y se han utilizado todos estos elementos para todas las tareas, se ha calculado que en recursos hardware se acumula una amortización de 550 horas * 0,21608 €/hora = 118,84 €.

Recursos software:

Software	Uds.	Coste/ud.	Total	Vida útil	Amortización
Ubuntu 18.04	1	0 €	0 €	-	0 €
GlobalProtect	1	0 €	0 €	-	0 €
GitHub	1	0 €	0 €	-	0 €
Editor LaTex	1	0 €	0 €	-	0 €
Editor Vim	1	0 €	0 €	-	0 €
Compilador ICC	1	0 €	0 €	-	0 €
LibreOffice	1	0 €	0 €	-	0 €
SLURM	1	0 €	0 €	-	0 €
Entorno MPI	1	0 €	0 €	-	0 €
Compilador IFORT	1	0 €	0 €	-	0 €
Paraver	1	0 €	0 €	-	0 €
Skype	1	0 €	0 €	-	0 €
Total:	-	-	**0 €**	-	**0 €**

Tabla 6.2: Tabla de costes en recursos software.

El compilador ICC tiene un coste pero se me ha proporcionado de forma gratuita puesto que viene instalado en el *cluster* Lenox. Sin embargo, en caso de no estar instalado se hubiese usado el compilador GCC que es gratuito. Para el entorno MPI y el compilador IFORT sucede lo mismo, y en su defecto se hubiese usado el entorno OpenMPI y una oferta gratuita para estudiantes respectivamente. Por otro lado, en cuanto al resto del software utilizado cabe decir que tampoco supondrán ningún coste en programas informáticos ni ningún añadido en amortización ya que son de libre distribución.

Recursos humanos:

El coste principal para este proyecto viene dado por los recursos humanos necesarios para realizarlo, cuantificado en base a los diferentes roles identificados entre las distintas tareas que conforman este trabajo. Los principales roles que se identifican son tres:

- **Jefe de proyecto:** Sus funciones son las relacionadas con la gestión del proyecto, como por ejemplo: elaborar la documentación, planificación, presupuesto, etc.
- **Analista:** Lleva a cabo las distintas evaluaciones realizadas a lo largo del proyecto.
- **Desarrollador:** Es el responsable de desarrollar el proyecto en sí: programando, diseñando y comprobando la correcta funcionalidad.

Para poder contabilizar el coste total en recursos humanos, primero debemos contabilizar en qué tipo de rol se han empleando las horas en este proyecto. En la siguiente tabla se muestran el número de horas que ha empleado cada rol según el grupo de tareas:

Grupo de tareas	**Horas**	**Horas/rol**		
		Jefe proyecto	**Analista**	**Desarrollador**
Gestión del proyecto	155 h.	125 h.	-	30 h.
Familiarización con el entorno	60 h.	-	-	60 h.
Evaluación estado servicio	20 h.	-	20 h.	-
Modificación de la configuración	90 h.	-	40 h.	50 h.
Evaluación y análisis de los resultados	225 h.	-	50 h.	175 h.
Total:	**550 h.**	**125 h.**	**110 h.**	**315 h.**

Tabla 6.3: Tabla de horas dedicadas por rol según el grupo de tareas.

Una vez identificadas las horas que cada rol ha dedicado, hay que identificar el salario bruto por hora de cada rol y, finalmente, se podrá calcular el coste total en recursos humanos. A continuación se muestran estos cálculos resumidos en una tabla:

Rol	**Salario bruto/h.**	**Horas**	**Total**
Jefe de proyecto	26,29 €/h.	125 h.	3286,25 €
Analista	23,57 €/h.	110 h.	2592,70 €
Desarrollador	21,50 €/h.	315 h.	6772,50 €
Total:	-	550 h.	**12651,45 €**

Tabla 6.4: Tabla de costes en recursos humanos.

El salario bruto por hora se ha obtenido por medio del salario bruto promedio descrito en el informe sobre el estado del mercado laboral en España [14], presentado por ESADE e InfoJobs, en el cual se ha divido por 1800 horas (horas estimadas laborables reales en un año) y se ha multiplicado por 1.3 para añadir los costes de la Seguridad Social, que representan un 30 %.

6.1.2. Costes indirectos

Por otro lado, encontramos los costes indirectos. Éstos son costes que ya no están directamente relacionados al producto y son debidos al hecho de realizar la actividad. Principalmente contabilizaremos un espacio para trabajar, la tarifa eléctrica, la tarifa de Internet y el desplazamiento en transporte público.

En cuanto al lugar de trabajo se ha contabilizado un espacio de *coworking* durante 6 meses, ya que es la duración que ha tenido el proyecto. La luz, Internet y mobiliario ya vienen

incluidos en las mensualidades del alquiler del espacio de trabajo, por lo que no se añadirán. Por último, para el transporte se han contabilizado dos tarjetas T-Jove ya que tienen una duración trimestral, por lo cual hacen falta dos para cubrir toda la duración del trabajo.

Recurso	**Coste/ud.**	**Uds.**	**Total**
Espacio de *coworking*	292 € [1]	6	1752 €
T-Jove 1 zona	80 €	2	160 €
Total:	-	-	**1912 €**

Tabla 6.5: Tabla de costes indirectos.

6.1.3. Contingencias e imprevistos

Con el concepto de contingencias se ha añadido un coste de un 10 % sobre la suma de los costes directos e indirectos como margen de seguridad para cubrir posibles obstáculos no previstos.

Por otro lado, en cuanto a imprevistos pueden ocurrir los siguientes con sus respectivas probabilidades de suceder:

Imprevisto	**Riesgo**	**Coste**	**Total**
Inaccesibilidad en Lenox	0,5 %	-	-
Falta de tiempo en evaluaciones (20 h.)	20 %	471,40 €	94,28 €
Falta de tiempo en implementación (20 h.)	15 %	430 €	64,50 €
Total:	-	-	**158,78 €**

Tabla 6.6: Tabla de imprevistos.

6.2. Control de gestión

Durante la realización de este proyecto se ha llevado a cabo un seguimiento sobre las horas estimadas por cada tarea y las horas reales que se han empleado, con tal de procurar que la desviación fuera mínima. Para ello, anoté en un diario las tareas realizadas diariamente y se fue consultando la planificación estimada. Para medir las posibles desviaciones se usaron una serie de indicadores.

En cuanto a las estimaciones en horas previstas de trabajo se calculará el desvío en eficiencia por parte de la mano de obra y el desvío en eficiencia por amortización, ya que se deberá actualizar la amortización prevista para el hardware según las horas reales empleadas. Ambos indicadores se calcularán con la misma fórmula:

$$(\text{consumo horas estimado} - \text{consumo horas real}) \times \text{ coste estimado}$$

[1] `https://meetbcn.com/coworking-barcelona/espacio-coworking-barcelona/`

En cambio, por parte de las estimaciones por costes se calcularán el desvío en coste por tarifa de la mano de obra y el desvío en coste por tarifa del alquiler del lugar de trabajo, ya que los costes fueron aproximados y pueden sufrir variaciones. Ambos indicadores se calcularán siguiendo la siguiente fórmula:

$$(\text{coste estimado} - \text{coste real}) \times \text{ consumo horas real}$$

Por otro lado, en caso de ocurrir imprevistos se anotarán y al final del proyecto se calculará la desviación entre los costes estimados para imprevistos y los costes reales que han supuesto. El valor de éstos imprevistos se verá reflejada en la eficiencia por parte de la mano de obra.

Finalmente, se obtendrá la desviación final obtenida en el proyecto de la diferencia entre el coste total estimado y el coste total real. Entonces se verá si el margen de contingencia calculado fue suficiente.

6.3. Presupuesto

Una vez identificados todos los costes necesarios, es posible realizar el presupuesto para este proyecto. A continuación se muestra el presupuesto total final:

Concepto	**Coste**
Costes directos	**12770,29 €**
Recursos humanos	12651,45 €
Recursos hardware	118,84 €
Recursos software	0,00 €
Costes indirectos	**1912,00 €**
Total acumulado	**14682,29 €**
Contingencias	1468,23 €
Imprevistos	158,78 €
Total	**16309,30 €**

Tabla 6.7: Presupuesto total del proyecto.

A pesar de los imprevistos detectados, tanto los plazos como los gastos planteados inicialmente se han cumplido, por lo que no han ocurrido desviaciones. Por lo tanto, el presupuesto final se ajusta al presupuesto que se calculó inicialmente.

Capítulo 7

Motivación

En este capítulo se expone la motivación que ha conducido a la creación del software EAR y la importancia de la gestión energética, ya que no se trata de algo trivial en que los factores influyan de forma proporcional. Se comienza concienciado sobre la importancia de llevar a cabo esta gestión, definiendo los elementos que influyen en ella y cómo se relacionan. Finalmente se muestra cómo EAR gestiona la potencia y se presentan conceptos propios de la arquitectura con la que se trabaja.

7.1. Introducción

La gestión energética se ha convertido en un tema en el área de la investigación ya que los centros HPC son grandes consumidores de energía y eso supone un problema. Al consumir energía se emiten gases que tienen efecto en la huella ecológica, con el añadido de que se traduce a final de cada mes en un coste económico. Por lo tanto, vemos que se trata de un problema tanto ecológico como económico, y es por eso que estos centros de datos definen límites de energía a consumir para un periodo de tiempo (*energy budget*) junto con otros límites como es el caso de la potencia, que vienen definidos por la infraestructura.

El reto para estos centros consiste en lograr eficiencia energética, es decir, aprovechar al máximo posible la energía que se está consumiendo, que a la larga se acaba traduciendo en ahorro económico y respeto con el medio ambiente. Sabiendo que la energía es el área que comprende la potencia durante un periodo de tiempo (ecuación 7.1), se podría pensar en reducir la frecuencia para minimizar la potencia (ecuación 7.2). Sin embargo, como se describe en [15], esta técnica podría ser contraproducente ya que lo más probable será que aumente el tiempo de ejecución (provocando una pérdida de rendimiento), y llegando incluso a incrementar la energía a consumir.

$$\text{Energía} = \text{Potencia} \times \text{Tiempo de ejecución} \tag{7.1}$$

$$\text{Potencia} = \text{Capacitancia} \times \text{Voltaje}^2 \times \text{Frecuencia} \tag{7.2}$$

La eficiencia energética es un compromiso entre potencia y rendimiento, en que para los centros de altas prestaciones no es aceptable el ahorro energético a cualquier coste, como la pérdida de rendimiento.

7.2. Gestión de la energía mediante DVFS

Existen diferentes propuestas para mejorar la utilización de la potencia. Entre ellas, para EAR se decidió utilizar la técnica Dynamic Voltage and Frequency Scaling (DVFS) [16], la cual es un mecanismo que soportan cada vez más procesadores y que permite la gestión de la potencia mediante la modificación de la frecuencia. Esta técnica posibilita al sistema operativo seleccionar una determinada frecuencia y voltaje.

La selección de frecuencia y voltaje del procesador se hace por medio de un *driver* que permite al sistema operativo modificar la frecuencia a la que trabaja dicho procesador. Cada *driver* ofrece un conjunto de **governors**, que son módulos en que cada uno tiene asociado unas funcionalidades concretas. En Lenox se encuentra instalado el *driver* `acpi-cpufreq`, necesario para poder utilizar EAR, ya que permite tener cargado el governor `userspace`, el cual posibilita al usuario seleccionar dinámicamente la frecuencia a la que funciona la CPU. Para cambiar la frecuencia los governor utilizan lo que se conoce como p-state.

Por **p-state** (*processor state*) se entiende una combinación que asocia una frecuencia y voltaje determinados. Los procesadores tienen asociados un vector de p-states que entre diferentes arquitecturas pueden diferir sus combinaciones de frecuencia y voltaje del vector.

Este trabajo se ha realizado en el *cluster* Lenox, el cual está basado en una arquitectura Intel x86_64 en que cada nodo está formado por dos Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz. Lenox cuenta con un vector de 16 p-states en que cuanto menor es el p-state, mayor es su frecuencia y voltaje de funcionamiento. Por lo tanto, los p-states más bajos nos proporcionan mayor rendimiento con un consumo mayor de potencia. En cambio, los p-states mayores consumen menos potencia pero penalizan el rendimiento. En la figura 7.1 se muestra de forma esquematizada cómo se organiza el vector de p-states en el *cluster* Lenox.

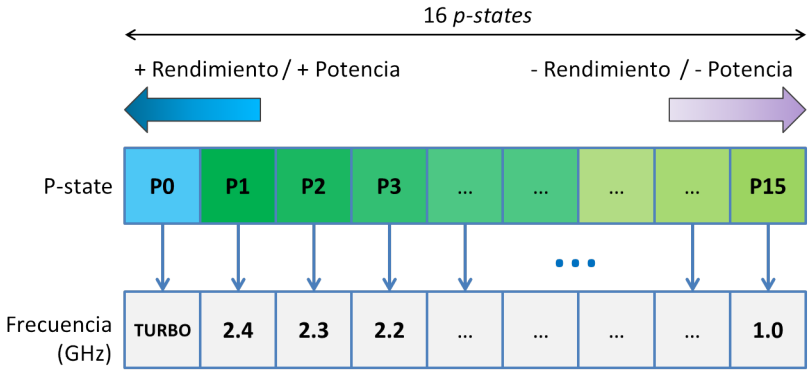


Figura 7.1: Organización de los p-states en Lenox.

El p-state de color azul (p-state 0) se utiliza para representar de forma genérica al rango de frecuencias pertenecientes al **modo turbo** (es el único caso en que no pertenece a una frecuencia y voltaje concretos). Por otro lado, en el p-state 1 se representa la **frecuencia nominal o máxima** del procesador, la cual se trata de la frecuencia máxima que establece el fabricante en que puede funcionar una CPU sin tener en cuenta el modo turbo. Como se ha visto, para el caso de los procesadores que utiliza Lenox, la frecuencia nominal es de 2,4 GHz. Seguidamente se encuentra la siguiente frecuencia más elevada permitida (2,3 GHz), y así sucesivamente hasta llenar el vector con frecuencias a las que puede ejecutar *jobs* una CPU. En Lenox los p-states se asignan de forma consecutiva y descendente, pero se trata un detalle propio de la arquitectura que puede diferir con otros modelos de procesadores.

A continuación, en la gráfica 7.2 se muestra una evaluación de forma numérica sobre la variación en la energía al reducir la frecuencia. La variación se ha estudiado comprobando la energía que consumen unas determinadas aplicaciones con una frecuencia específica respecto a la energía consumida ejecutada a la frecuencia nominal, en concreto 2,4 GHz.

Por otro lado, el capítulo 10 se dedica a analizar un conjunto de aplicaciones en que se muestran más detalles sobre las variaciones.

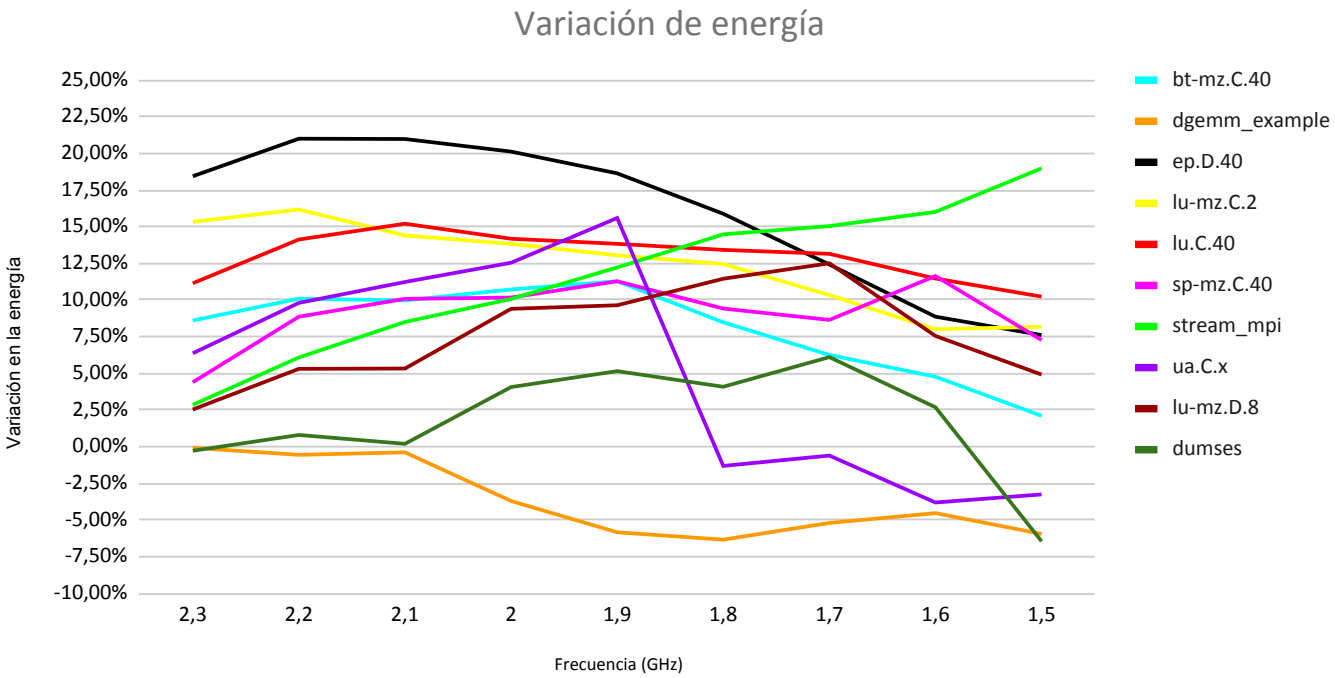


Figura 7.2: Variación en la energía de aplicaciones respecto 2,4 GHz.

Como se puede observar en la figura 7.2 no tiene un comportamiento lineal (o proporcional), ya que **depende de la frecuencia y de la propia aplicación en sí que se está ejecutando**.

A través de la gráfica vemos que con la aplicación STREAM se obtiene ahorro en energía al reducir la frecuencia. Por otro lado, hay aplicaciones que llegan a consumir incluso más, como es el caso de la aplicación DGEMM.

Podría parecer que con la mayoría de aplicaciones el reducir la frecuencia ya sería suficiente, aunque el ahorro no sea siempre el mismo. Sin embargo, no es tan simple como reducir la frecuencia, como se muestra en la figura 7.3.

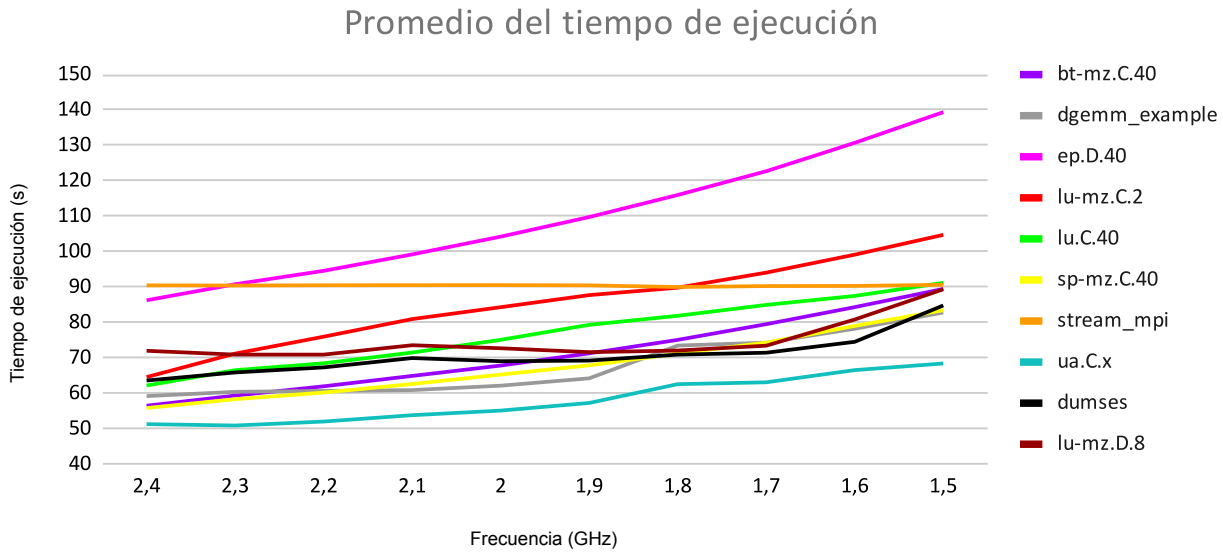


Figura 7.3: Promedio del tiempo de ejecución de aplicaciones según la frecuencia a la que se ejecutan.

La figura 7.3 muestra el promedio del tiempo de ejecución de las aplicaciones según la frecuencia a la que han sido ejecutadas. En la gráfica vemos que a medida que se reduce la frecuencia, el tiempo de ejecución aumenta para todas las aplicaciones excepto STREAM, DUMSES y LU-MZ.D. El porqué estas tres aplicaciones prácticamente no aumenta su tiempo de ejecución se revela en el capitulo 10. El hecho de incrementar el tiempo de ejecución ocasiona una pérdida del rendimiento, y como se comentó anteriormente en este capítulo, en un entorno HPC no es aceptable el ahorro energético a cambio de pérdida de rendimiento.

Por lo tanto, vemos que la eficiencia energética es una gestión crítica y no sencilla en la que no sólo se trata de reducir la frecuencia. Con esto vemos que la gestión es compleja ya que no depende exclusivamente de la arquitectura, y para ello es necesario algo más: un software que gestione la energía como lo es EAR.

Capítulo 8

EAR

En el presente capítulo se muestra una visión general para situar en contexto el *framework* EAR (Energy Aware Runtime) [4][17]. Este proyecto se basa en uno de los componentes de EAR, el Global Manager, que se encuentra descrito con más profundidad en el capítulo 9, quedando el resto de componentes fuera del alcance de este trabajo.

8.1. Descripción general

EAR [3] se trata de un software de sistema que ofrece una gestión energética (proporcionando un control eficiente de la energía) para un conjunto de nodos interconectados. En el capítulo introductorio (capítulo 1) ya se ha presentado una visión inicial de lo que significan EAR y sus componentes. Vimos que ofrece, en resumen, cuatro servicios para lograr eficiencia energética (tanto a nivel de aplicación como a nivel global del sistema), que son:

Figura 8.1: Principales servicios de EAR.

- Monitorización energética
- *Accounting* energético
- Optimización energética
- Control energético

La monitorización, el *accounting* y el control energético se llevan a cabo de manera independiente del tipo de *job*. Sin embargo, la optimización energética la ofrece el componente EARL que, por el momento, solo funciona para aplicaciones totalmente MPI o híbridas (MPI+OpenMP).

Como modo de recordatorio y para un mejor seguimiento de la lectura se muestra en la figura 8.2 un esquema con los componentes que conforman EAR y las interacciones que hay entre ellos. Los componentes de EAR son: 1) *daemon* de EAR (EARD), 2) EAR Global Manager (EARGM), 3) librería de EAR (EARL), 4) EAR Database Manager (EARDBD) y 5) *plugin* de EAR (EARplug).

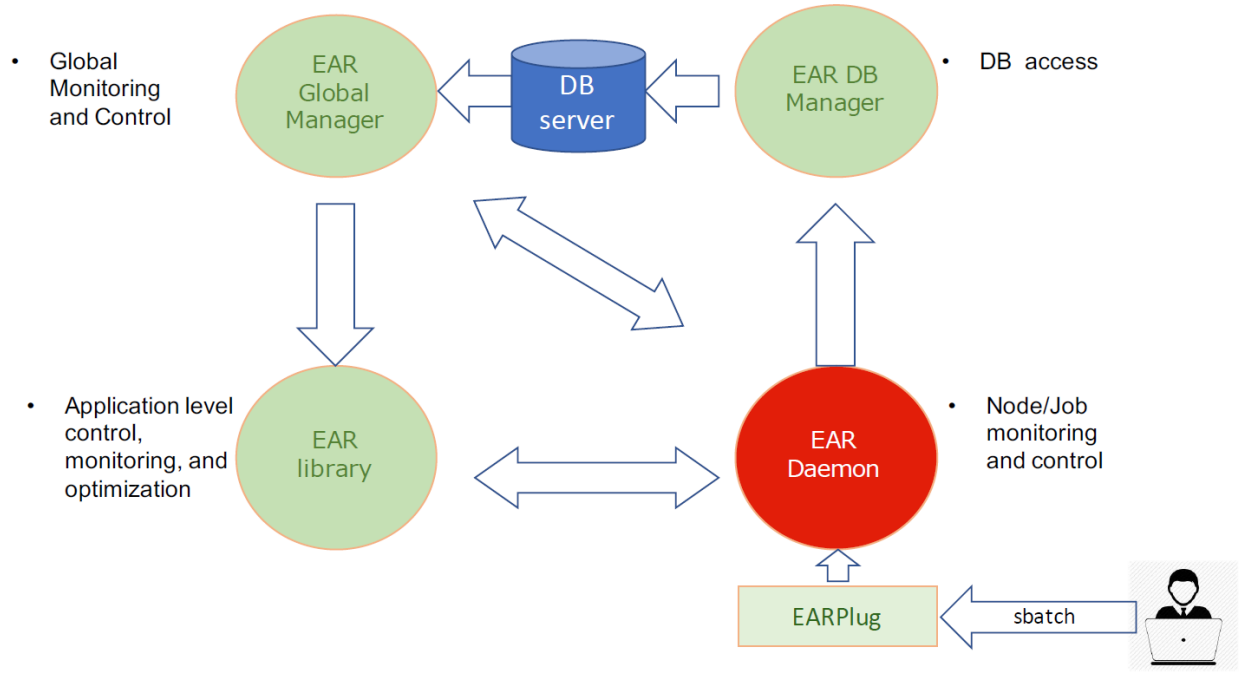


Figura 8.2: Esquema básico de la interacción de los componentes de EAR. [18]

- Un **EARD**, también conocido como Node Manager, consiste en un *daemon* por nodo que se ejecuta en los nodos de cálculo con permisos de superusuario (*root*) para ofrecer unos servicios. Por un lado, proporciona métricas privilegiadas (como la frecuencia media, temperatura, etc.) y métricas energéticas (potencia del nodo y de la DRAM) que reportará a la BD a través del EARDBD. Por otro lado, accede a funcionalidades de muy bajo nivel, como modificar la frecuencia de la CPU, necesarias para el momento en que se deban aplicar ajustes notificados por el EARGM. Finalmente, también lleva a cabo un servicio de monitorización de la potencia por nodo, que permite llevar un control de la energía consumida en el sistema, así como una revisión periódica del funcionamiento de los componentes hardware.
- La función del **EARGM** es ofrecer un control energético global. Para ello monitoriza el consumo energético del sistema y, en función de su estado, aplica unas acciones para garantizar un consumo máximo de la energía en un periodo determinado. Como este componente es el corazón de este trabajo, se dedica el capítulo 9 a exponerlo en profundidad y minuciosamente.

- **EARL** es el encargado de optimizar el consumo energético en las aplicaciones. Este componente, al estar coordinado con el Global Manager para ofrecer también control energético, se va desarrollar más detalladamente en la sección 8.2.
- **EARDBD** se trata de un *daemon* cuya función es reportar en una BD centralizada para EAR los datos que recibe de los EARDs e EARL (por lo que deberá tener privilegios de acceso a dicha BD). Recordemos que si el sistema es robusto, es recomendable tener varias instancias de este componente y un conjunto de nodos asignados en cada EARDBD que se instale para reducir la cantidad de inserciones y conexiones a la base de datos. Es de utilidad instalarlo cuando se quiere mantener un *accounting* energético para poder analizar datos posteriormente. En este TFG se utiliza la base de datos MariaDB[1], aunque EAR también puede soportar PostgreSQL.
- **EARplug** se trata de un mecanismo que permite cargar dinámicamente EARL para *jobs* que utilizan el gestor de recursos SLURM. Es capaz de identificar automáticamente cuándo empieza o termina un *job* y lo notifica al EARD para ofrecer *accounting* a nivel de aplicación. Estos datos tienen asociados el identificador del *job* para poder relacionar esta información de EAR con la reportada por el *scheduler*, pudiendo realizar un análisis posterior.

  Los autores de EAR, cuando crearon este componente, siguieron la filosofía de "la facilidad de uso" para lograr éxito dentro de la comunidad HPC. Se trata de un *plugin* SLURM SPANK[2] que permite extender opciones a través de comandos de SLURM (srun, sbatch y salloc), funcionando de forma transparente para el usuario y simple para los administradores de sistemas.

En caso de querer conocer los detalles de instalación y configuración del software EAR se puede consultar la guía para los administradores de sistemas en la referencia [17].

8.2. EARL: Librería de EAR

El componente EARL se trata de una librería en *runtime* que proporciona optimización energética por aplicación y, en colaboración con EARGM también ofrece control energético (por aplicación). EARL trabaja de forma totalmente transparente al usuario, pues éste no debe modificar su aplicación (por ejemplo, marcar regiones en el código) ni añadir parámetros de entrada especiales.

EARL garantiza un uso óptimo de la energía del sistema siguiendo los requerimientos especificados por el administrador de sistemas. En la figura 8.3 se muestra de forma esquematizada las fases que sigue el componente EARL para lograrlo.

[1] MariaDB: `https://mariadb.org/`
[2] SLURM SPANK plugin: `https://slurm.schedmd.com/spank.html`

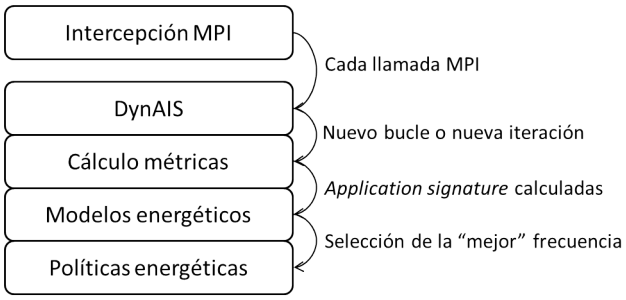


Figura 8.3: Fases internas de EARL. Adaptación de [4].

EARL utiliza la interfaz de *profiling* de MPI (PMPI) y la variable de entorno LD_PRELOAD para cargar y ejecutar la librería de modo transparente con el *job*, interceptando las llamadas MPI. Luego se ejecuta un algoritmo clave llamado DynAIS (Dynamic Application Iterative Structure detection) que detecta en *runtime* los patrones (bucles) correspondientes a las aplicaciones MPI o híbridas (MPI+OpenMP). Este algoritmo, a diferencia de otras propuestas que hay en el mercado, hace posible que prácticamente no haya *overhead* [15]. En la referencia [4] se encuentra el algoritmo DynAIS explicado en detalle y evaluado, en el que se demuestra que EARL es una librería *lightweight* en que el *overhead* es tan bajo, a veces incluso nulo, que apenas influye en el rendimiento.

Una vez detectado un bucle, se calculan una serie de métricas por cada iteración que reciben el nombre de *application signature*. Aportan información sobre las características de la aplicación, así como el CPI, tiempo por iteración, potencia media por nodo y el porcentaje de instrucciones vectoriales presentes. Es importante comentar que en cuanto a la potencia media de un nodo, EARL no sólo se fundamenta en la energía de la CPU sino que también incluye la energía de otros componentes, como la memoria. Estas métricas identifican el comportamiento de la aplicación, que depende de múltiples factores (como qué nodos se están usando, su configuración y datos de entrada), contribuyendo a un mejor uso de los recursos del sistema ya que participan en la optimización de las aplicaciones y ayudan a determinar si podrían rendir mas óptimamente. Las *application signature* junto con otras métricas llamadas *system signature* se utilizan para calcular unos modelos energéticos de EAR.

Las *system signature* son métricas que recogen datos del sistema, informando del rendimiento de los componentes principales que hay en un nodo, como la CPU y la memoria. Se utilizan para optimizar la energía del sistema, asegurando una eficiencia máxima del rendimiento, aunque también sirven para detectar cuando un componente no está funcionando correctamente. Estas métricas se calculan durante el proceso de instalación de EAR.

Los modelos energéticos de EAR se encargan de predecir el tiempo de ejecución y potencia por cada nivel de frecuencia posible para un nodo. Con los resultados de estos pronósticos, EARL seleccionará la frecuencia que considere óptima según unas políticas energéticas que a continuación se comentarán. Cabe decir que al ser EARL consciente de las características

de las aplicaciones, se va revaluando con la frecuencia seleccionada y podría deshacer ajustes realizados, incluso detectando cambios en las métricas anteriores, recalculándolas de nuevo.

EARL ofrece dos políticas energéticas: `MIN_TIME_TO_SOLUTION` y `MIN_ENERGY_TO_SOLUTION`. Con todo, encontramos una tercera 'política' bajo el nombre de `MONITORING_ONLY`. A pesar de implementarse como una política energética, no lo es realmente ya que se limita a reportar la información de *accounting* que recoge de cada aplicación, en la que las métricas de la configuración del nodo permanecerán sin modificar salvo que el componente EARGM decida cambiarlas.

- `MIN_TIME_TO_SOLUTION`: Trata de recompensar a las aplicaciones que son energéticamente eficientes. Esto lo hace aumentando la frecuencia siempre que se garantice una mínima mejora de rendimiento denotada por un límite de eficiencia bajo el nombre de **threshold**. La mejora de rendimiento se mide a través de una ratio entre el beneficio en rendimiento que se predice que se puede obtener y el incremento de la frecuencia (ver ecuaciones 8.1, 8.2 y 8.3).

  Con esta política las aplicaciones empiezan ejecutándose en una frecuencia que se denomina **frecuencia por defecto**, predefinida por el administrador de sistemas, y es inferior a la **frecuencia máxima o nominal** (frecuencia máxima establecida por el fabricante a la que puede funcionar una CPU sin tener en cuenta el modo turbo). Por lo tanto, las aplicaciones que sean más eficientes energéticamente serán las que funcionen a frecuencias más cercanas a la frecuencia nominal, dentro de este rango entre frecuencia por defecto y nominal.

$$Ganancia\ rendimiento \geq Ganancia\ frecuencia \times threshold\ \equiv$$

$$\frac{Ganancia\ rendimiento}{Ganancia\ frecuencia} \geq threshold \tag{8.1}$$

  donde *Ganancia rendimiento* y *Ganancia frecuencia* se calculan como:

$$Ganancia\ rendimiento = \frac{(T - T_{nuevo})}{T} \tag{8.2}$$

$$Ganancia\ frecuencia = \frac{(f_{nueva} - f)}{f} \tag{8.3}$$

  En la ecuación 8.2, T representa el tiempo de ejecución que se estima con la frecuencia actual y T_{nuevo} el tiempo de ejecución estimado con un nivel de frecuencia superior.

En la ecuación 8.3 se calcula la ratio para la variación de la frecuencia respecto al siguiente nivel, donde f_{nueva} es la frecuencia para la que se está estimando el rendimiento y f representa la frecuencia actual.

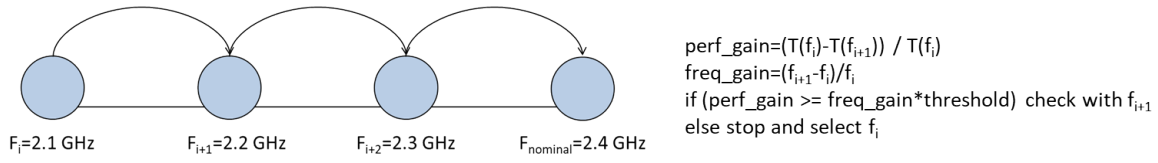


Figura 8.4: Funcionamiento de la política `MIN_TIME_TO_SOLUTION`. Adaptación de [1].

Ejemplo: Si nos encontramos en un sistema donde su administrador lo ha configurado con la política `MIN_TIME_TO_SOLUTION`, threshold al 70 %, frecuencia por defecto a 2,1 GHz y se tiene una frecuencia máxima a 2,4 GHz, la aplicación empezará a ejecutarse con la frecuencia por defecto (2,1 GHz). Cuando EARL calcule las métricas de la aplicación (*application signature*), predecirá los modelos energéticos para el siguiente nivel de frecuencia (2,2 GHz) y calculará la mejora de rendimiento que prevé que podrá obtener. Como se ve en la imagen 8.4, si la ganancia de rendimiento que se predice es como mínimo un 70 %, la política energética hará las mismas predicciones y comprobaciones para el siguiente nivel de frecuencia (2,3 GHz), y así hasta llegar a la frecuencia máxima. En el momento en que no se alcance un rendimiento mínimo del 70 %, la política energética escogerá la última frecuencia en la que se obtenía un rendimiento mínimo del 70 %. Ejemplo adaptado de [1].

- `MIN_ENERGY_TO_SOLUTION`: Trata de averiguar la frecuencia que genera una energía mínima, pero está limitada por una pérdida máxima de rendimiento que viene indicada por el threshold. Es decir, hay definido un linde que indica la máxima pérdida de rendimiento permitida, en que nunca se escogerá una frecuencia con la que se prevea un rendimiento peor al indicado por dicho límite (ver ecuaciones 8.4 y 8.5). Con esta política las aplicaciones empiezan ejecutándose con la frecuencia máxima.

$$P\acute{e}rdida\ rendimiento \leq threshold \tag{8.4}$$

donde *Pérdida rendimiento* se calcula como:

$$P\acute{e}rdida\ rendimiento = \frac{(T - T_{defecto})}{T_{defecto}} \tag{8.5}$$

En la ecuación 8.5, T indica el tiempo de ejecución estimado con la nueva frecuencia y $T_{defecto}$ el tiempo de ejecución estimado con la frecuencia máxima.

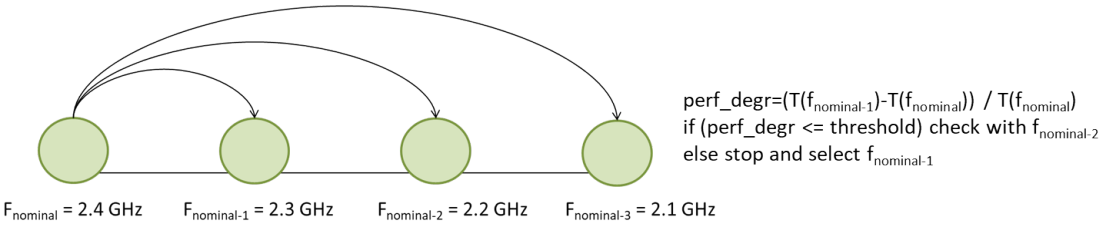


Figura 8.5: Funcionamiento de la política `MIN_ENERGY_TO_SOLUTION`.

Ejemplo: Si nos encontramos ante un sistema en el cual el administrador ha configurado un threshold del 10 % y la política `MIN_ENERGY_TO_SOLUTION`, EARL nunca escogerá una frecuencia que pueda causar más de un 10 % de degradación en el rendimiento respecto al que se obtendría con la frecuencia nominal.

Sin embargo, este TFG se ha realizado ejecutando todos los *jobs* con la política `MIN_TIME_TO SOLUTION`, ya que es la que EAR utiliza por defecto, por lo que se debe tener en mente durante todo este trabajo.

```
[xatomas@login2301 BT-MZ]$ econtrol --status
  hostname      power   temp    freq    job_id  stepid  policy  pfreq  th
   cmp2618        366    68C    2.95    127275       0      MO  2.40    0      MT  2.10   70      ME  2.40   10
   cmp2620        359    70C    2.93    127275       0      MO  2.40    0      MT  2.10   70      ME  2.40   10
   cmp2630        349    65C    2.92    127275       0      MO  2.40    0      MT  2.10   70      ME  2.40   10
   cmp2631        364    69C    2.98    127275       0      MO  2.40    0      MT  2.10   70      ME  2.40   10
```

(a) Aplicación ejecutándose sin EARL.

```
[xatomas@login2301 BT-MZ]$ econtrol --status
  hostname      power   temp    freq    job_id  stepid  policy  pfreq  th
   cmp2618        316    62C    2.38    127274       0      MO  2.40    0      MT  2.10   70      ME  2.40   10
   cmp2620        300    60C    2.38    127274       0      MO  2.40    0      MT  2.10   70      ME  2.40   10
   cmp2630        281    55C    2.38    127274       0      MO  2.40    0      MT  2.10   70      ME  2.40   10
   cmp2631        299    58C    2.38    127274       0      MO  2.40    0      MT  2.10   70      ME  2.40   10
```

(b) Aplicación ejecutándose con EARL utilizando la política `min_time`.

Figura 8.6: Configuración de nodos durante la ejecución de una aplicación.

Finalmente, en la parte superior de la figura 8.6 se muestra un ejemplo de una determinada aplicación ejecutándose en 4 nodos sin EARL y, en la parte inferior se presenta la misma aplicación ejecutándose en los mismos nodos con EARL utilizando la política `MIN_TIME_TO_SOLUTION`.

Estos resultados se han obtenido a través de uno de los comandos que EAR proporciona (en la siguiente sección se describen estos comandos y cómo activar EARL), con el que es posible ver la configuración en tiempo real de los nodos que hay en el sistema. Este comando indica por cada nodo (identificado por la columna hostname) la potencia, temperatura, la frecuencia de cada uno a la que está trabajando, el identificador del *job* para SLURM, el

stepid y, en grupos de tres, se muestra una política energética y su configuración en términos de frecuencia por defecto y threshold. En la parte superior vemos que la aplicación funciona a prácticamente 3 GHz, consumiendo de media 360W y alcanzando una temperatura de unos 70ºC por nodo. En cambio, al activar EARL se consumen 300W y una temperatura de unos 60ºC por nodo, yendo a una frecuencia de 2,4 GHz.

8.3. Ejecución y comandos

Para ejecutar una aplicación con EARL y poder disfrutar de la optimización energética que nos ofrece, se hace ya de forma totalmente transparente al usuario en el caso para aquellos sistemas en los que EARL esté activo por defecto (como sucede en el supercomputador SuperMUC-NG [19]). Por otro lado, si se trata de una opción optativa sería simplemente activar el flag de EAR (`--ear=on`) como una de las opciones al lanzar un *job* para ejecutarse. Su utilización es así de sencilla gracias al *plugin* de SLURM.

Por otro lado, un aspecto importante a comentar es que **EAR se configura a través de un único fichero de configuración (`ear.conf`)**, el cual sigue un patrón parecido al fichero de configuración de SLURM (`slurm.conf`), simplificando así el proceso de instalación a los administradores de sistemas que conozcan el gestor de colas SLURM. EAR también ofrece una serie de comandos que aportan información relevante y herramientas útiles para el administrador de sistemas [17]:

- `eacct` (energy account): Muestra información de *accounting* almacenada en la BD de EAR sobre los *jobs* una vez han finalizado (tanto por nodo como la media de todos los involucrados). Se muestra, entre otros, la frecuencia media a la que ha funcionado, tiempo de ejecución, potencia, ancho de banda, CPI, energía, etc.
- `econtrol` (energy control): Permite modificar la configuración del *cluster* (sin afectar al fichero `ear.conf`) relacionada con la configuración de la política de energía, por ejemplo, variando el threshold, que será necesario en aquellos casos en que el EARGM trabaje en modo manual. Entre otras opciones, y como hemos visto en la figura 8.6, también se puede consultar la configuración de unos determinados nodos, incluso su estado viendo si en la columna de job_id aparece un identificador o no, sabiendo así si está ocupado.
- `ereport` (energy report): Devuelve el consumo energético de los nodos durante un determinado periodo de tiempo . Estos datos son los que se encuentran en la BD, recogidos por los nodos y son muy útiles de cara al análisis del consumo energético. Se utilizará este comando para obtener los datos con los que se ha evaluado el componente (capítulo 12).

Capítulo 9

Global Manager

Este capítulo se dedica exclusivamente a exponer el EAR Global Manager (EARGM), que es el componente central del trabajo y ofrece un control global energético para sistemas de alto rendimiento. En otras palabras, gestiona los limites de energía globales, lo que constituye el motivo de este trabajo. En las siguientes secciones se describirá que es EARGM y en qué consiste, la configuración inicial de la que se parte y las modificaciones especificas que han sido necesarias para realizar el proyecto.

9.1. Descripción y estado inicial

El Global Manager se trata de uno de los componentes que conforman el software EAR y es **único** dentro de un sistema. Los supercomputadores tienen un gasto continuo, la energía, motivo que lleva a la creación de este componente ya que es necesario un gestor. EARGM es un *daemon* que ofrece continuamente monitorización de la energía que se está consumiendo y lleva a cabo un conjunto de acciones para proporcionar lo que se conoce como *energy capping*, que es la energía máxima consumida que se permite para un periodo de tiempo determinado (límite energético). El objetivo de EARGM es garantizar que los límites energéticos máximos definidos para el *cluster* no se alcancen.

Para que EARGM logre su funcionalidad (control energético global) necesita de la colaboración de otros componentes de EAR. La idea es sencilla y se muestra de forma visual en la figura 9.1[1]. En primer lugar, los EARDs reportan a la base de datos (o a un EARDBD en caso de tener uno asignado) los resultados de la continua monitorización de la potencia que realizan sobre el nodo en el que se están ejecutando cada uno. Si se encuentran EARDBDs instalados en el sistema, recordemos que únicamente se dedican a reportar los datos que reciben de los EARDs a la BD del sistema. Posteriormente, el Global Manager ejecuta una

[1]Es un esquema simplificado ya que en un sistema muy grande se encontraría el componente EARDBD replicado.

query muy simple y rápida que recoge los datos que han reportado los EARDs para monitorizar la energía que se está consumiendo en el sistema. Con estos datos, el EARGM evalúa si el estado del sistema se encuentra dentro de los límites energéticos globales especificados por el administrador de sistemas. En caso de estar correcto, no aplica ninguna acción. Sin embargo, en caso de no ser así, ordena a todos los EARDs unos ajustes que deben realizar con tal de cumplir con los límites energéticos establecidos. Asimismo existe la posibilidad de mostrar al administrador de sistemas mediante un fichero de *log* los datos registrados, el nivel energético en el sistema y las acciones realizadas, en caso de efectuarse. Una vez los EARDs reciben los ajustes ordenados por el EARGM, lo notificarán a la librería que reevaluará las condiciones y las actualizará en función de los nuevos ajustes. En caso de no haber EARL, los EARDs aplicarán los ajustes.

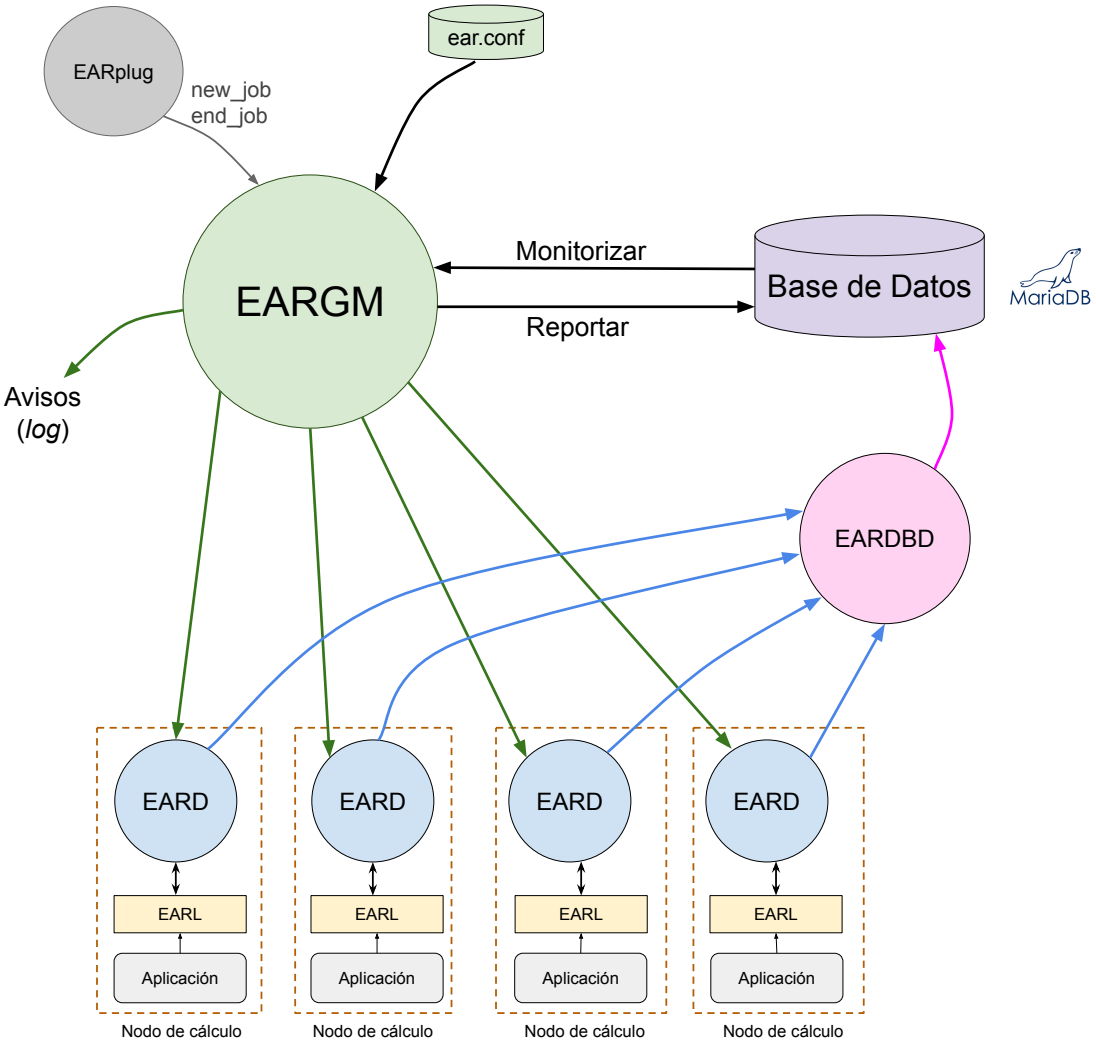


Figura 9.1: Esquema de la interacción del Global Manager con otros componentes.

Por otro lado, en el dibujo se muestra una comunicación (coloreada en gris) en que el EARplug contacta con el Global Manager al comenzar y terminar un *job*. Sin embargo, se trata de una opción que se encuentra desactivada desde los inicios. Fue diseñada como una opción para incluir acciones asociadas al comenzar o finalizar un *job*. Es una comunicación existente pero no es utilizada en este trabajo.

El Global Manager se configura a través del fichero `ear.conf`, como podemos apreciar en la figura anterior. Esto es una gran ventaja ya que no es necesaria ninguna API para reconfigurar este componente, por lo que para realizar cualquier ajuste será tan simple como modificar tal fichero y volver a iniciar el servicio de sistema, permitiendo así una reconfiguración dinámica.

Como se comentó en la introducción del trabajo, este componente se puede configurar para trabajar en dos modos:

- **Modo de monitorización** (o modo manual): Si se opta por utilizar este modo, el Global Manager se limita a informar al *sysadmin* sobre el estado del sistema, responsabilizándole para que tome las acciones correspondientes manualmente. Como vimos en el capítulo 8, EAR ofrece un conjunto de comandos que permiten ajustar la configuración de la máquina. Anteriormente se comentó que este modo ya se encuentra cubierto y en producción.
- **Modo automático**: Si se escoge este modo, por contra, el Global Manager tomará acciones por sí solo en función del estado en que se encuentre el sistema. Para ello, ajustará la configuración de los nodos con tal de cumplir con las restricciones especificadas por el administrador de sistemas. La totalidad de este proyecto trata el modo automático ya que sólo existía un prototipo de un diseño inicial y carecía de evaluación.

Por otro lado, EARGM también lee del `ear.conf` el límite energético, un periodo de tiempo corto y otro periodo de tiempo estrictamente más largo que el anterior. El periodo más corto en el Global Manager recibe el nombre de **periodo T1**, en el cual se indica cada cuanto tiempo se quiere actualizar la información en el componente. Por otro lado, el periodo más largo recibe el nombre de **periodo T2**, en el cual se indica el periodo de tiempo en el que se evalúa el cumplimiento de los límites energéticos establecidos. En otras palabras, y como vemos en la figura 9.2, tiene el funcionamiento de una ventana que se desplaza en el tiempo (*sliding window*) en que se va moviendo el periodo T1 hacia adelante para la evaluación de los limites energéticos. A continuación se muestra un ejemplo:

Ejemplo: Supongamos que nos encontramos ante un sistema en que su administrador ha configurado el EARGM con un periodo T1 = 1 hora, periodo T2 = 12 horas y un límite energético de 1300000J. Esto significa que EARGM calculará la energía que se ha consumido en las últimas 12 horas y comparará si está dentro del límite energético establecido (en este caso 1300000J), en que la energía consumida se actualizará cada hora. Posteriormente hará lo mismo para las últimas 12 horas, y así sucesivamente como vemos en la figura 9.2.

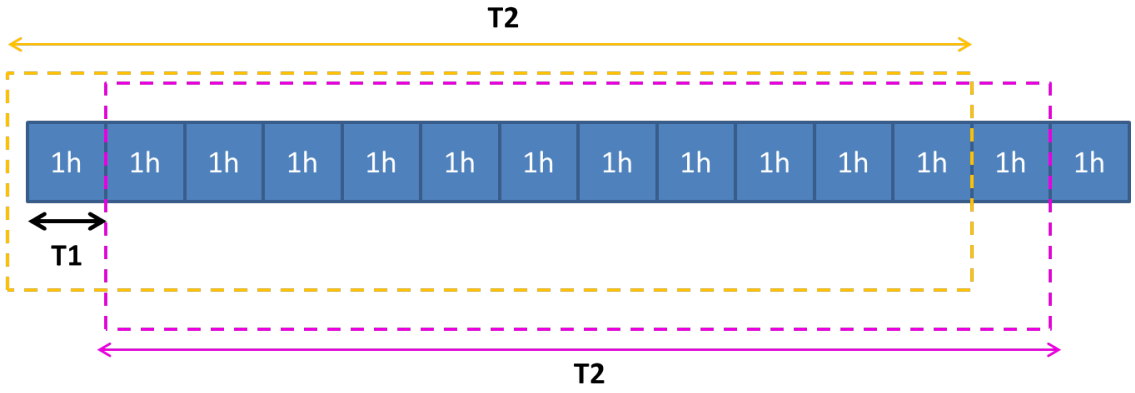


Figura 9.2: Relación del periodo T1 y periodo T2.

En caso de configurarse en modo automático, el Global Manager tomará acciones con tal de garantizar que el consumo energético se encuentre dentro del límite fijado, ajustando la configuración de los nodos del sistema. Para ello se definen cuatro niveles de criticidad, que en EAR se conocen como **niveles de *warning***, en que cada uno realiza una serie de ajustes relativos.

En caso de aplicarse ajustes en algún nivel de *warning*, el Global Manager se espera una cantidad de periodos T1 definida por el administrador en que no se evalúa el estado del sistema, que recibe el nombre de **grace period**. Una vez transcurrido el grace period ya se puede volver a evaluar el estado del sistema.

Es importante recordar que en este proyecto se asume la política `min_time`, la cual acelera (permite el funcionamiento con frecuencias más elevadas) aquellos *jobs* energéticamente eficientes en frecuencias altas, manteniendo siempre un mínimo de mejora en rendimiento.

En la siguiente figura 9.4 se muestra el diagrama de estados correspondiente al estado en que se encontraba antes de empezar a trabajar en este proyecto. En el grafo se indica el estado del sistema (nivel de *warning*) en los nodos y en cuyos arcos se indica la condición a cumplir y las acciones a llevar a cabo, como se indica en la figura 9.3.

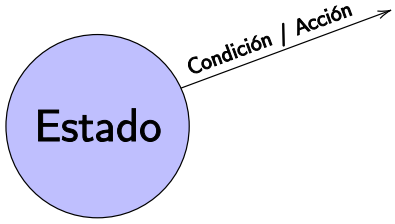


Figura 9.3: Formato de la representación del grafo de estados.

La configuración inicial del Global Manager cuenta con cuatro niveles de *warning* definidos en el `ear.conf`: **NO PROBLEM**, **WARNING1**, **WARNING2** y **PANIC**. A continuación se describen las acciones que se llevan a cabo para cada nivel de criticidad y se adjunta pseudocódigo para facilitar su comprensión y relacionar con el diagrama.

El primer paso es determinar el estado del sistema (nivel de *warning*). Por ejemplo, si tuviéramos unos niveles de *warning* definidos al 85 %, 90 % y 95 %, implicaría que un consumo energético durante el periodo T2 inferior al 85 % corresponde al estado NO PROBLEM. En cambio, en caso de alcanzar entre el 85 % y el 90 % correspondería al nivel de WARNING1. Sino, en caso de situarse entre el 90 % y el 95 % estaríamos en el estado de WARNING2. Finalmente, a partir del 95 % y superior ya nos encontraríamos ante el nivel más crítico, PANIC.

Para el caso particular del estado inicial se encuentran definidos los niveles de *warning* al 85 %, 90 % y 95 %.

```
if (% energía < 85) then NP_cond
else if (% energía <= 85 && % energía < 90) then W1_cond
else if (% energía <= 90 && % energía < 95) then W2_cond
else if (% energía >= 95) then PANIC_cond
```

Una vez determinado el nivel correspondiente al estado del sistema, se llevan a cabo unas determinadas acciones que son relativas para cada nivel.

- **NO PROBLEM**: Este nivel indica que la energía que se está consumiendo durante el periodo T2 no está cercana al límite energético establecido. Por lo tanto, no se lleva a cabo ninguna acción ya que se considera que se está teniendo un consumo energético correcto.

  ```
  if (NP_cond) then {};
  ```

- **WARNING1**: Este nivel no se considera muy crítico por lo que únicamente se limita a incrementar en un 5 % el nivel del threshold. Sin embargo, este ajuste tiene un efecto indeterminado ya que puede darse el caso de *workloads* que reaccionen ante él y otros que no, donde principalmente tendrán efecto las cargas intermedias. Vimos en el capítulo 8 que se trata de una de las métricas que la política de la librería utiliza para ajustar la frecuencia de funcionamiento, por lo que si no se utilizara EARL, los EARDs no aplicarán este ajuste ya que no tendría ningún valor.

  ```
  if (W1_cond) then {
      threshold = threshold + 5%;
  }
  ```

- **WARNING2**: Es el nivel intermedio de criticidad en que además de aumentar el threshold se reduce la frecuencia. Concretamente, aumenta el threshold en un 10 % e incrementa el p-state en dos posiciones. Como vimos en el capítulo 7, aumentar el p-state implica reducir la frecuencia ya que los p-states más bajos se corresponden con las frecuencias más elevadas. Es importante comentar que al reducir el p-state se reduce tanto para la frecuencia nominal como para la frecuencia por defecto.

```
if (W2_cond) then {
    threshold = threshold + 10%;
    pstate = pstate + 2;
}
```

- **PANIC**: Este nivel está extremadamente cercano al límite por lo que aplica las medidas más duras. Efectúa las mismas acciones que el nivel WARNING2 pero reduciendo otro p-state más (tanto para la frecuencia nominal como para la frecuencia por defecto), siendo así aún más estrictas.

```
if (PANIC_cond) then {
    threshold = threshold + 10%;
    pstate = pstate + 3;
}
```

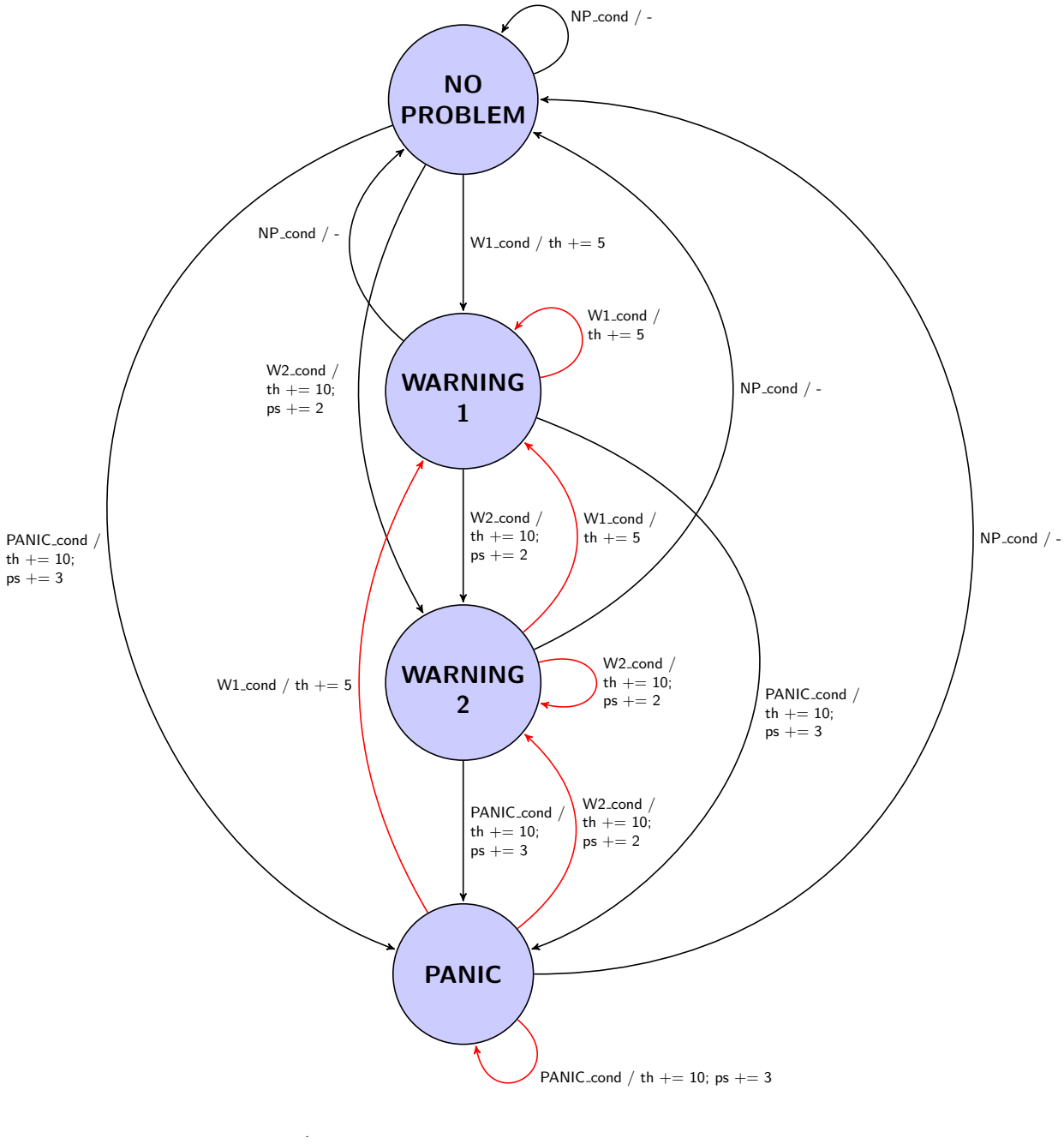


Figura 9.4: Diagrama de estados de la configuración inicial del Global Manager.

Sin embargo, con este diseño inicial se puede apreciar a simple vista determinados ajustes en los que trabajar de inmediato. En los arcos de color rojo se identifican acciones acumulativas, por lo que el rendimiento acabará perjudicado. Por ejemplo, si nos encontramos en

el estado PANIC se aumentará el threshold en un 10 % y se bajará la frecuencia 3 p-states. Si al aplicar estos ajustes reducimos el nivel de criticidad y pasamos a WARNING2, se reducirían otros 2 p-states y se aumentaría otro 10 % el threshold. Por lo que sólo con este cambio ya se habrán reducido 5 p-states de la frecuencia de funcionamiento. Además, si se siguiera en un nivel de criticidad se irían aplicando los ajustes cada vez, perdiendo así el rendimiento en el sistema. Por otra parte, otro aspecto importante es que carece de una funcionalidad de retorno a los ajustes iniciales, es decir, no hay recuperación de la configuración inicial.

En la figura 9.5 se muestran algunos de los campos del fichero `ear.conf`. Se observa que los límites que definen el nivel de *warning* son configurables. Sin embargo, para este trabajo se han utilizado los niveles que vienen preconfigurados en el *cluster* Lenox ya que son unos límites razonables (habiendo un 5 % de diferencia entre cada uno).

```
# EAR Global Manager

GlobalManagerVerbose=1
GlobalManagerPeriodT1=60                    #specified in seconds
GlobalManagerPeriodT2=300                   #specified in seconds
GlobalManagerUnits=-                        #'-' Joules; 'K' KiloJoules ; 'M' MegaJoules
GlobalManagerEnergyLimit=X
GlobalManagerHost=mgt2309
GlobalManagerPort=50000
GlobalManagerMode=1                         #0=manual;  1=automatic
GlobalManagerMail=nomail
GlobalManagerWarningsPerc=85,90,95
GlobalManagerGracePeriods=10
GlobalmanagerUseLog=1
```

Figura 9.5: Atributos del fichero `ear.conf`.

A continuación se muestran dos ejemplos extraídos de la ejecución de dos experimentos compuestos por aplicaciones de características distintas[2] (uno intensivo en cálculo y otro intensivo en memoria) para exponer el comportamiento del Global Manager con la configuración inicial en la que se encontraba.

Ambas aplicaciones se han ejecutado con la política `min_time`, configurada con una frecuencia por defecto a 2,1 GHz y un threshold al 70 %. El Global Manager se ha configurado con un periodo T1 = 1 minuto, periodo T2 = 5 minutos y el número de periodos T1 en el grace period es de 10. Para cada experimento se presentan los resultados de las ejecuciones mediante gráficas para mostrar por cada periodo T1 de la ejecución: energía consumida (tanto en % respecto el límite como en valor absoluto), nivel de *warning*, nivel de threshold y valores de los p-states para la frecuencia máxima y la frecuencia por defecto. Para las gráficas, los primeros periodos corresponden al comienzo de la carga de trabajo y los últimos a su finalización.

[2]En el capítulo 10 se presenta un estudio sobre la caracterización de las aplicaciones.

a) Caso experimento compuesto por una aplicación intensiva en cálculo:

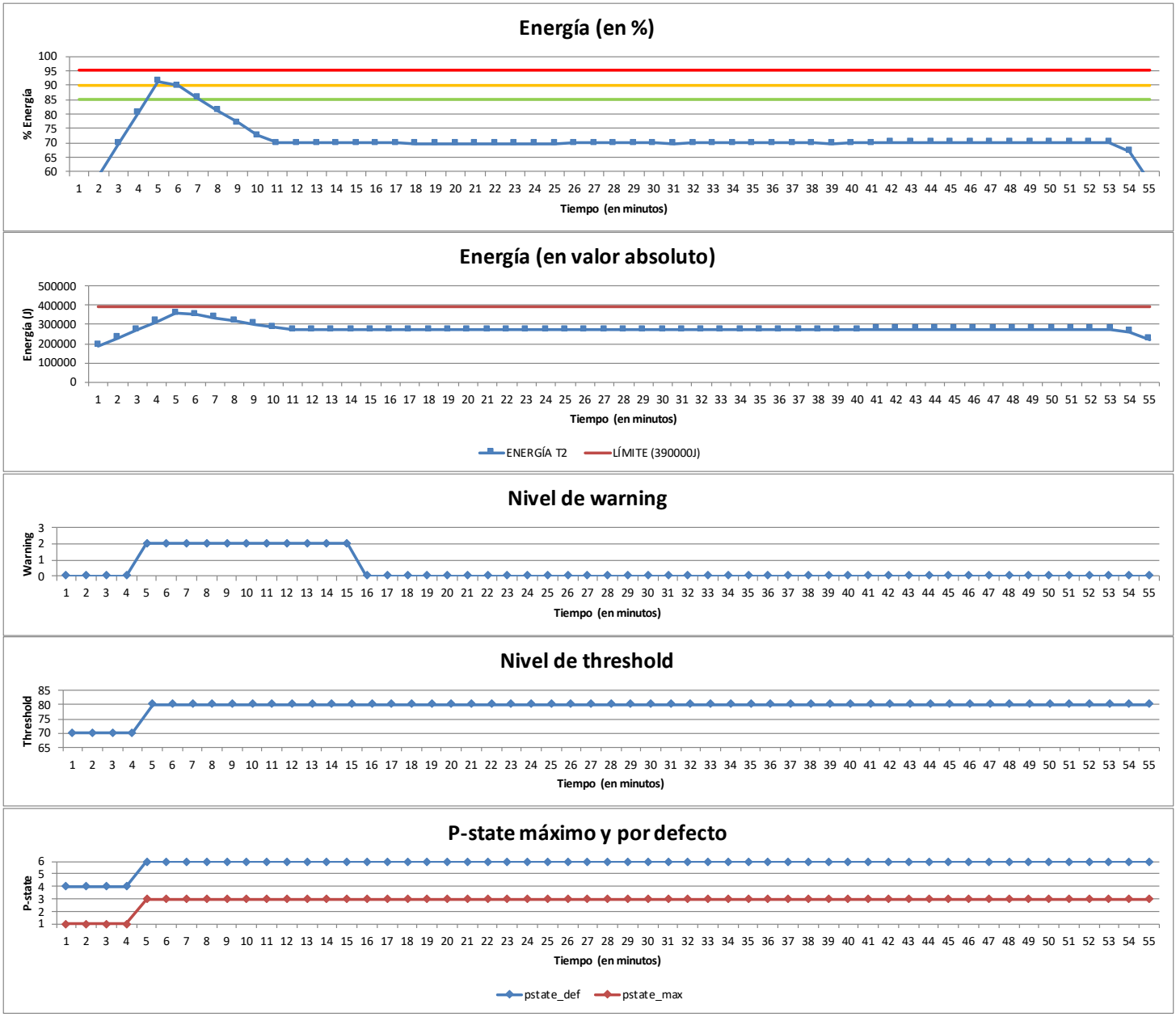


Figura 9.6: Experimento intensivo en cálculo gestionado por el EARGM inicial.

El experimento de la figura 9.6 ha llegado al grado de criticidad WARNING2 y se le han aplicado los ajustes correspondientes a su nivel de *warning*. Vemos que el consumo energético se desploma con la nueva configuración. No obstante, se observa que transcurre la mayor parte del tiempo desaprovechando el límite energético establecido que, por contra, perjudicará a la aplicación en términos de rendimiento.

b) Caso experimento compuesto por una aplicación intensiva en memoria:

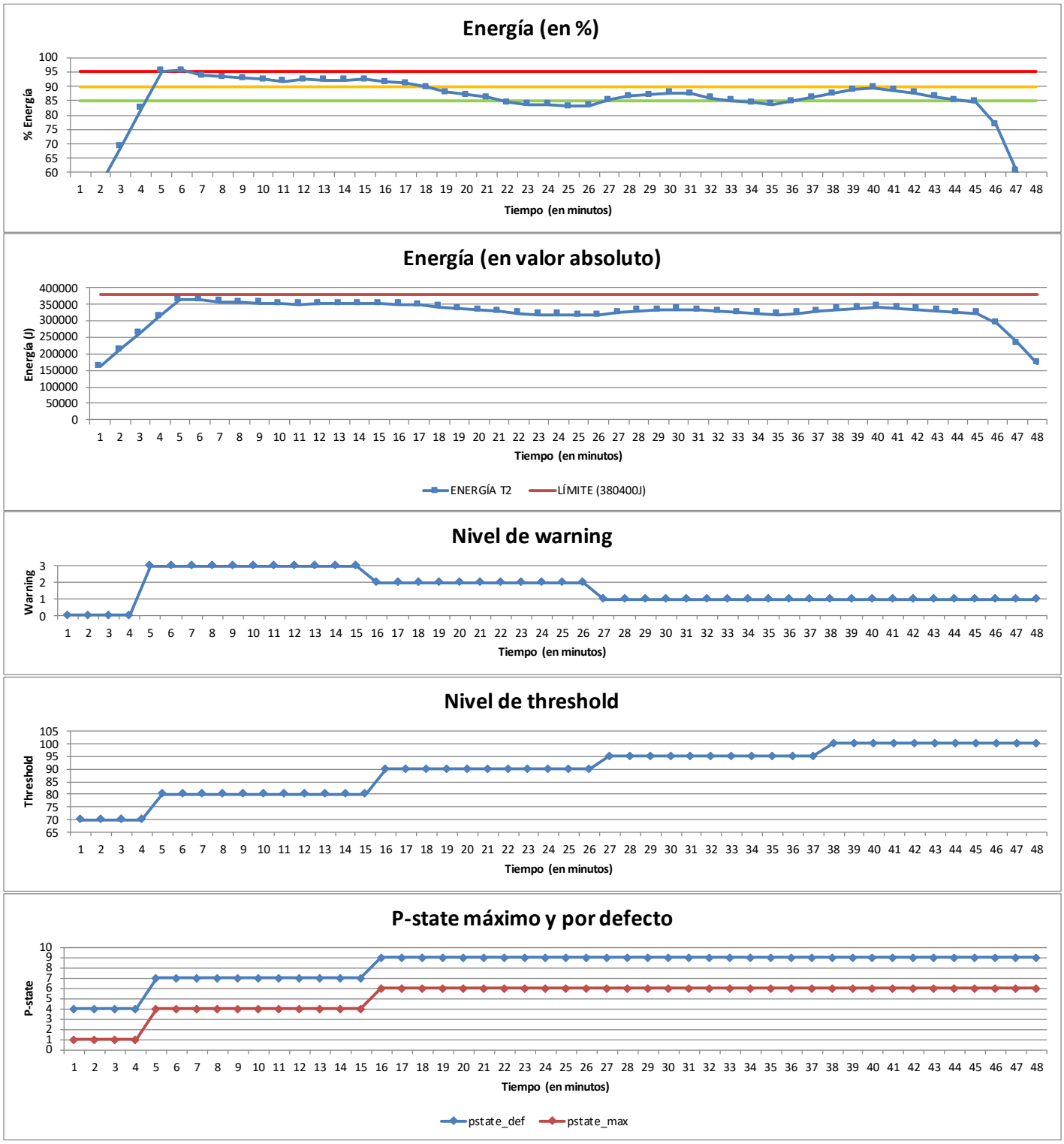


Figura 9.7: Experimento intensivo en memoria gestionado por el EARGM inicial.

El experimento de la figura 9.7 ha llegado al nivel de *warning* PANIC, por lo que se le han aplicado ajustes más agresivos que en la prueba anterior. Sin embargo, debido a la naturaleza de la carga de trabajo, no reacciona prácticamente a los ajustes (se verá más detalladamente en el capítulo 10). En consecuencia, a pesar de lograr reducir un nivel de criticidad, se van aplicando más ajustes, llegando a incrementar el threshold a un 100 % y a reducir hasta 5 p-states para las frecuencias máxima y por defecto.

En resumen, vemos que el Global Manager se trata de un componente que vela para que los *workloads* que se encuentren en ejecución en un sistema no lleguen a los límites energéticos definidos (proporcionando *energy capping*) pero sin comprometer al rendimiento. Para ello controla y adapta la configuración del sistema a tales límites establecidos. Por otro lado, hemos visto que EARGM determina un estado de forma global (de todo el sistema) en que sus órdenes son detectadas y reaccionan dinámicamente en *runtime* por el conjunto de EAR, aunque localmente pueden haber diferentes ajustes. Finalmente, hemos comprobado que es obvia la necesidad de un rediseño del comportamiento de este componente junto con una evaluación.

9.2. Modificaciones específicas

Para poder realizar este trabajo han sido necesarias un conjunto de modificaciones, que son:

- **Eliminación de privilegios para comandos EAR**: Mi cuenta de usuario en el *cluster* Lenox no tiene privilegios en EAR. Por contra, los comandos de EAR requieren de estos privilegios para poder ejecutarse. El uso de estos comandos son imprescindibles para realizar mi proyecto, por lo que se ha modificado el código fuente de los comandos, suprimiendo la parte del código que comprueba si se tienen permisos.
- **Filtraje por nodos**: Como el Global Manager aplica acciones para todos los nodos del sistema, se ha incorporado en él un sistema que filtra por nodos su actividad. Los nodos que se quieren evaluar se indican en el fichero `ear.conf` y, de esta manera, se puede comprobar su funcionalidad y realizar los experimentos sin afectar al resto de investigadores que se encuentren trabajando en el *cluster*.

  Para poder trabajar en el proyecto filtrando por nodos ha sido necesario programar un conjunto de nuevas funciones que lo soporten, como hallar con qué nodos se quiere trabajar, incrementar el threshold, reducir la frecuencia o devolver el estado para un conjunto de nodos determinado, entre otras.
- **Creación de una tabla propia dentro de la base de datos**: Se ha creado una tabla en la BD de EAR en Lenox exclusivamente para mi uso, con tal de evitar interferencias de los datos recogidos por mí y los recogidos por el equipo de EAR en el *cluster*. Al no tener privilegios de superusuario, esta tabla ha sido creada por el

equipo técnico de EAR. Para poder escribir en esta nueva tabla es tan simple como activar `#define EXP_EARGM 1` como una opción de compilación.

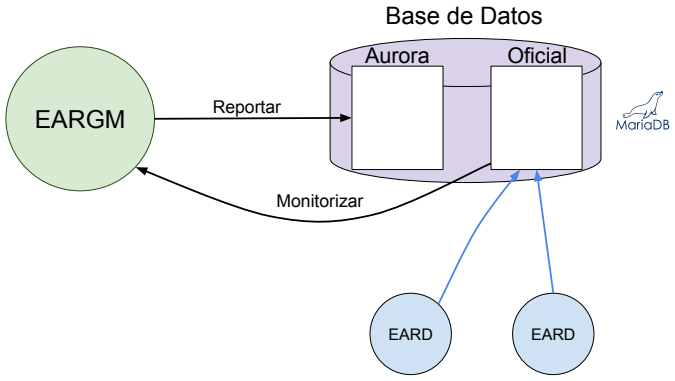


Figura 9.8: Interacción del EARGM con la base de datos de Lenox.

En la figura 9.8 se presenta la interacción del Global Manager con la base de datos de Lenox. En ella vemos que los EARDs reportan sus datos a la tabla oficial de EAR, donde después el EARGM lee de ella estos datos para llevar a cabo la monitorización. Por el contrario, para reportar utiliza esta nueva tabla propia, que en el dibujo se encuentra indicada como Aurora.

- **Definición dinámica del límite energético basada en la potencia media por nodo**: Se ha creado la variable de entorno EAR_MAX_POWER para indicar la potencia máxima permitida por nodo, y así simplificar la especificación del límite energético. Además, de este modo también evitamos modificar el fichero `ear.conf` cada vez. Con el valor del límite de la potencia (proporcionada por esta variable de entorno), el número de nodos y el periodo de tiempo indicado en el periodo T2 se obtiene el límite energético para el sistema.

$$\begin{aligned} \text{Energía} &= (\text{Núm. nodos}) \times (\text{Potencia/nodo}) \times (\text{Tiempo de ejecución}) \\ &= \text{Potencia} \times \text{Tiempo de ejecución} \end{aligned}$$

Capítulo 10

Análisis de aplicaciones

En este capítulo se va a analizar un conjunto de *benchmarks* con la finalidad de entender el impacto que tienen los distintos p-states sobre la potencia en las aplicaciones. En otras palabras, estudiar la variación en potencia que sufren. Esto nos servirá para poder entender el comportamiento de los *workloads* en sus ejecuciones. En la variación de potencia no sólo influye la frecuencia en la cual se encuentren, sino que también influyen las características propias de cada aplicación. Por lo tanto, este capítulo también incluye un estudio sobre la caracterización de las aplicaciones para poder entender su reacción sobre la potencia consumida. En la primera sección se presenta el entorno de trabajo en el cual se han ejecutado las aplicaciones y con en el que también se han realizado los diversos experimentos para la evaluación del componente EARGM, que se encuentra en el capítulo 12.

10.1. Entorno de ejecución

Tanto el análisis de aplicaciones como la evaluación del componente Global Manager se han realizado en el *cluster* Lenox, cuya propiedad es de Lenovo y está compuesto por nodos Lenovo ThinkSystem SD530. Se han realizado todos los experimentos en nodos basados en una arquitectura Intel x86_64, configurados con dos Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz con 20 *cores* cada uno, sumando un total de 40 *cores* por nodo. Estos nodos cuentan además con una memoria de 96GB o 192GB y están conectados mediante una red de interconexión Intel OmniPath (abreviado Intel OPA). Con todo, en nuestro caso no es importante el tipo de nodo porque los experimentos no utilizan la capacidad máxima de la máquina.

La máquina tiene instalado un sistema de colas SLURM, versión 18.08.8, para poder ejecutar los *jobs* en los distintos nodos. Lenox también cuenta con *environment modules* (módulos), lo cual es un sistema que permite cargar de forma dinámica paquetes software en un entorno sin tener que instalarlos de forma manual [20]. He cargado los módulos que contienen los compiladores ICC e IFORT para compilar el componente y las aplicaciones,

respectivamente. No hay que olvidar que es imprescindible contar con un entorno MPI.

Para el análisis de este capítulo, las aplicaciones se han ejecutado con la política `MONITORING ONLY` utilizando 1 nodo y, para poder llevar a cabo un buen estudio de los resultados, se ha utilizado el mismo nodo para todas las ejecuciones. También comentar que los experimentos pertenecientes a este capítulo se han realizado con el rango de frecuencias desde 2,4 GHz hasta 1,5 GHz, ambos incluidos. Aunque el procesador puede trabajar en frecuencias aún más bajas, no se incluyen en los experimentos ya que no se utilizan. Por eso se han realizado las pruebas con los rangos de trabajo habitual en un entorno HPC.

10.2. Descripción

En esta sección se describe brevemente la función de las distintas aplicaciones que se han analizado. Algunas de ellas pertenecen a un conjunto de *benchmarks* creados por la NASA, los NAS Parallel Benchmarks (NPB), los cuales están diseñados para evaluar el rendimiento en supercomputadores que explotan paralelismo [21]. Por otro lado, también encontramos otro conjunto de *benchmarks* de la NASA que se trata de versiones de los NPB con la diferencia de que trabajan con múltiples zonas (MZ). Se distinguen bajo las siglas NPB-MZ y están diseñados para explotar múltiples niveles de paralelismo en las aplicaciones y soportan modelos de programación híbrida MPI+OpenMP [21]. Es importante comentar que los conjuntos de aplicaciones NPB y NPB-MZ tienen los tamaños de problema predefinidos en lo que se denomina *clase*, que viene indicado con una letra alfabética.

- **BT-MZ** (Block Tri-diagonal Multi-Zone) [22] resuelve múltiples sistemas independientes de ecuaciones de Navier-Stokes tridiagonales de bloques. Al ser MZ lo hace con múltiples zonas.
- **EP** (Embarrassingly Parallel) [23] estima los límites superiores alcanzables para el rendimiento de coma flotante. Se trata de una aplicación *embarrassingly parallel*, como dice su nombre, que en paralelismo se designa a aquellas en que apenas hay comunicación y dependencias entre tareas.
- **LU** (Lower Upper Symmetric Guass-Seidel) [22] resuelve múltiples sistemas independientes de ecuaciones de Navier-Stokes con una variante del método de Gauss-Seidel.
- **LU-MZ** (Lower Upper Multi-Zone) hace lo mismo que la LU descrita anteriormente, con la diferencia que ésta trabaja con múltiples zonas. Además, para esta aplicación se analizan dos clases distintas.
- **SP-MZ** (Scalar Penta-diagonal Multi-Zone) [22] resuelve múltiples sistemas independientes de ecuaciones de Navier-Stokes pentadiagonales escalares. Al ser MZ lo hace con múltiples zonas.
- **UA** (Unstructured Adaptive) [24] mide el rendimiento al resolver problemas que implican accesos irregulares y dinámicos a memoria.

Por otro lado, también se han analizado otras aplicaciones que no forman parte de esos conjuntos, que son:

- **DGEMM** (Double-precision General Matrix Multiply) [25] mide la ratio de operaciones en coma flotante en multiplicaciones de matrices en doble precisión.
- **STREAM** [26] mide el rendimiento de cuatro operaciones vectoriales con el objetivo de medir el ancho de banda de memoria real.
- **DUMSES** [27] es una simulación 3D híbrida (MPI+OpenMP) magneto-hidrodinámica de Godunov de segundo orden en coordenadas cartesianas, esféricas y cilíndricas.

En la tabla 10.1 se presenta la configuración de las aplicaciones que se ha utilizado para obtener los resultados. Para ello se muestran el número de nodos utilizados, cuántos procesos (MPIs), cuántos *threads* por proceso (OpenMPs) y el número de *cores* utilizados en total.

Aplicación	**Núm. nodos**	**MPIs**	**OpenMPs**	**Total cores**
BT-MZ.C	1	40	1	40
DGEMM	1	1	40	40
EP.D	1	40	1	40
LU-MZ.C	1	2	20	40
LU.C	1	40	1	40
SP-MZ.C	1	40	1	40
STREAM	1	40	1	40
UA.C	1	1	40	40
DUMSES	1	32	1	32
LU-MZ.D	1	8	5	40

Tabla 10.1: Configuración de las aplicaciones.

10.3. Estudio de la potencia

En esta sección se muestra un estudio sobre la variabilidad en potencia consumida que presentan las aplicaciones en función de la frecuencia a la que se ejecutan. Para realizar este análisis se han efectuado múltiples ejecuciones variando la frecuencia y obteniendo unos resultados que se han representado mediante gráficas. A continuación, primero se va a observar la potencia que han consumido según la frecuencia seleccionada y la aplicación. Finalmente, se estudia la variación de la potencia entre la frecuencia seleccionada respecto 2,4 GHz y se verá esta variabilidad tanto de forma individual por aplicación como de forma global con todas ellas.

En la siguiente figura 10.1 vemos el transcurso del promedio de la potencia (*average DC power*) consumida de cada una de las aplicaciones en función de la frecuencia a la cual se han ejecutado, con el fin de tener una primera imagen global de la situación.

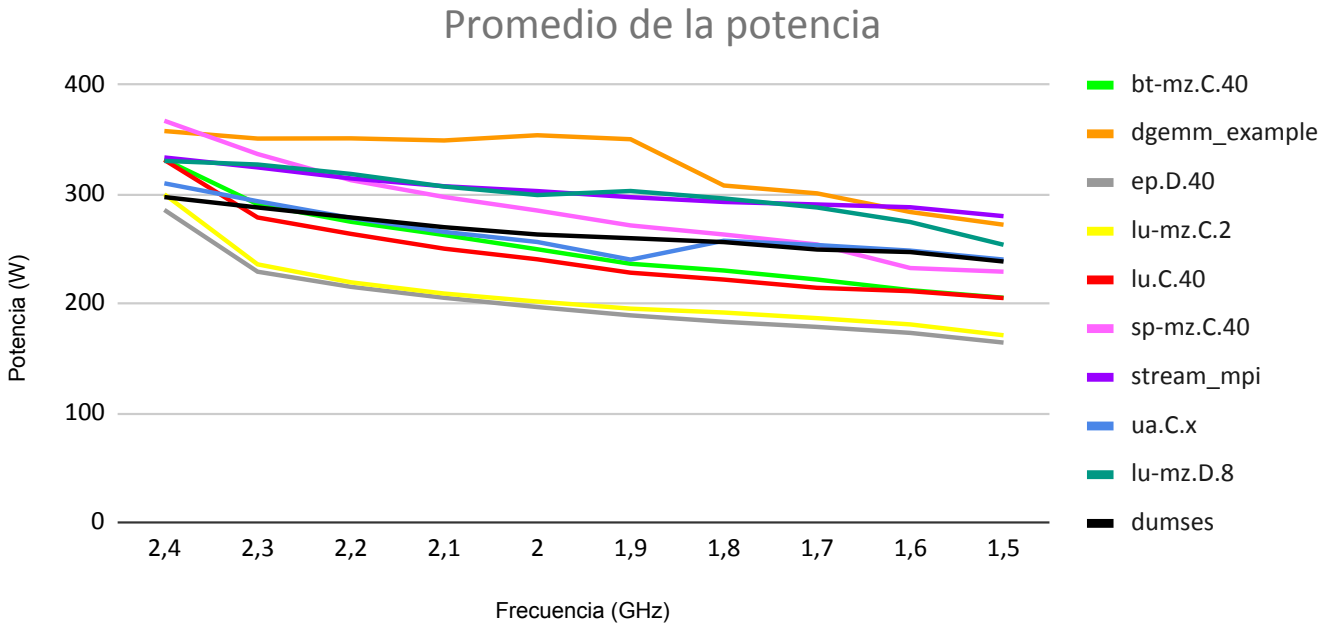
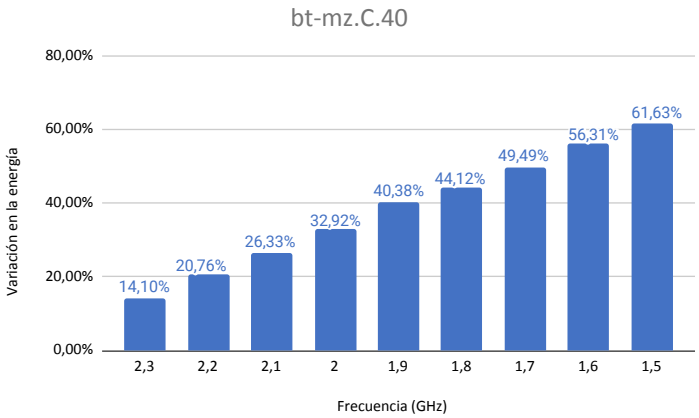


Figura 10.1: Promedio de la potencia consumida de las aplicaciones según la frecuencia a la que se ejecutan.

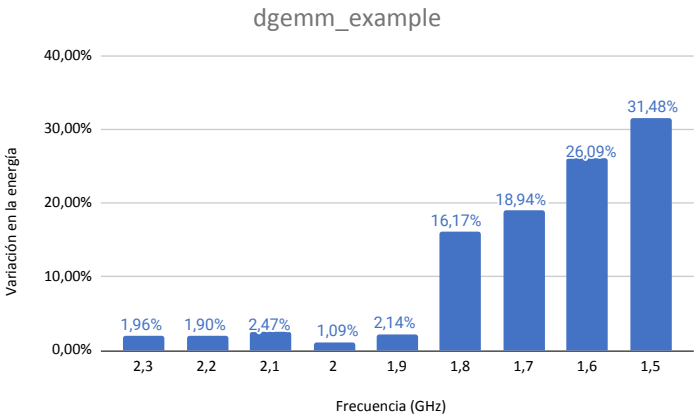
Vemos que todas las aplicaciones excepto la UA.C, que es un caso especial que presenta un punto de inflexión anómalo de 1,9 GHz a 1,8 GHz, tienen una tendencia descendente, es decir, cuánto más baja es la frecuencia menos vatios consumen. También se observa que no todas reaccionan del mismo modo, donde algunas lo hacen de una forma más agresiva que otras.

Una vez se tiene esta visión global de la situación es hora de ver de forma individual la variación en potencia que sufre cada aplicación al reducir un p-state respecto al anterior. En las figuras 10.2 y 10.3 se muestra para cada aplicación la evolución de la variación de la potencia respecto su ejecución a 2,4 GHz. La variación y representación en porcentaje se ha calculado por medio de la formula 10.1:

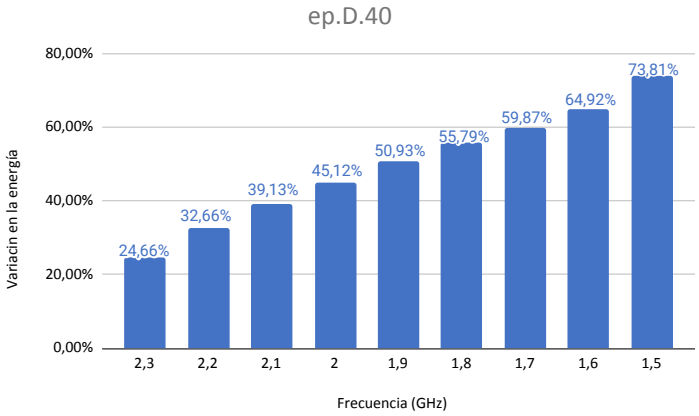
$$\text{Variación en la potencia} = \left(\frac{\text{Potencia consumida con 2,4 GHz}}{\text{Potencia consumida con nueva frecuencia}} - 1\right) \times 100 \quad (10.1)$$



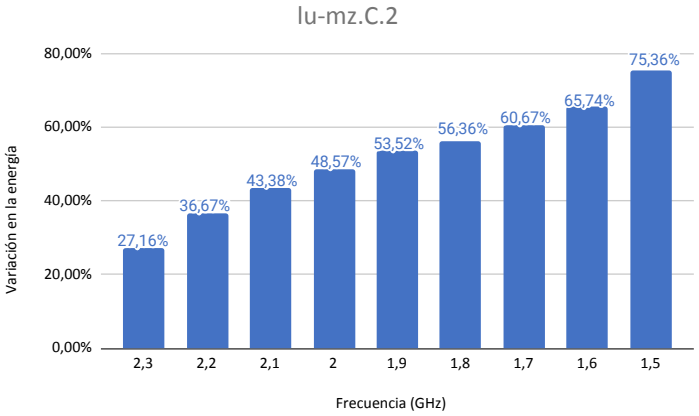
(a) BT-MZ clase C



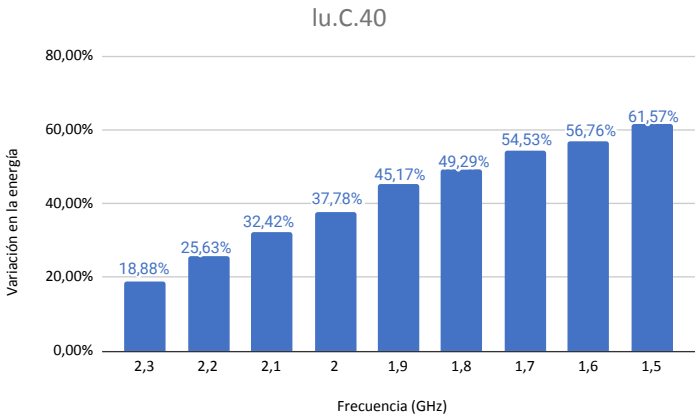
(b) DGEMM



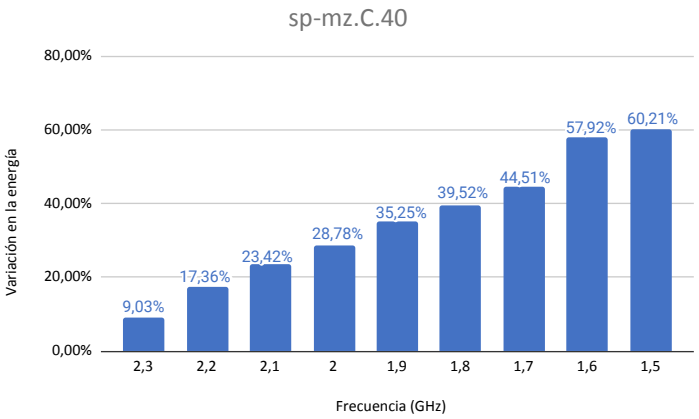
(c) EP clase D



(d) LU-MZ clase C

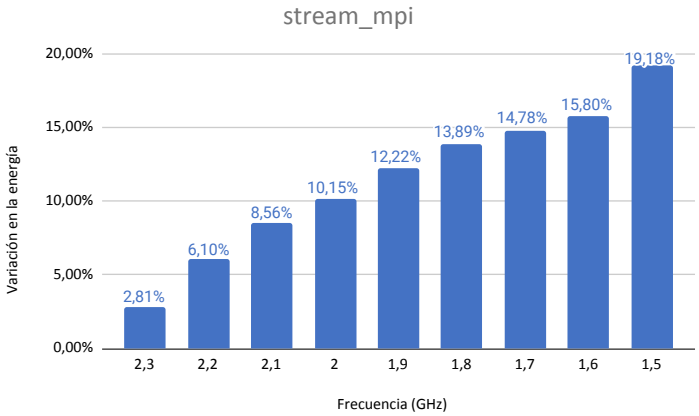


(e) LU clase C

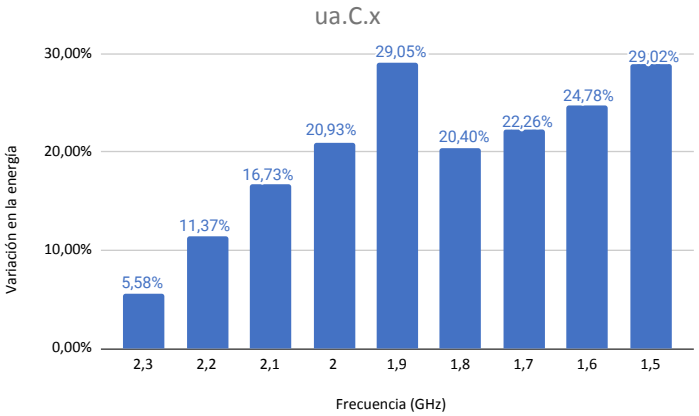


(f) SP-MZ clase C

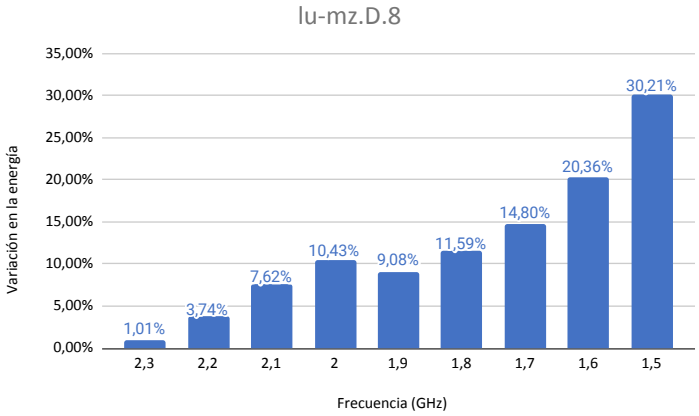
Figura 10.2: Variación en la potencia en las aplicaciones BT-MZ.C, DGEMM, EP.D, LU-MZ.C, LU.C y SP-MZ.C.



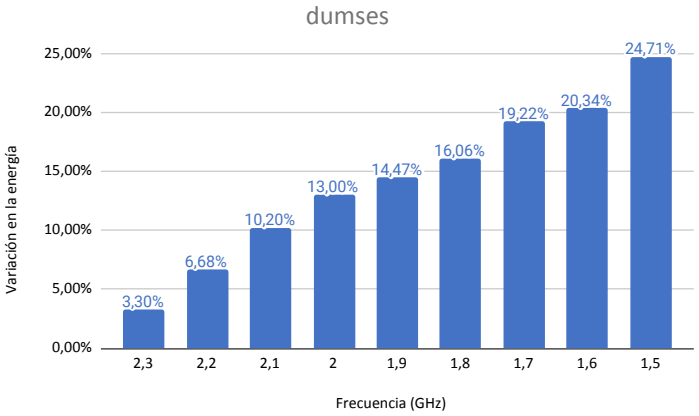
(a) STREAM



(b) UA clase C



(c) LU-MZ clase D



(d) DUMSES

Figura 10.3: Variación en la potencia en las aplicaciones STREAM, UA.C, LU-MZ.D y DUMSES.

A través de estas gráficas vemos que hay algunas aplicaciones que obtienen una variabilidad mucho mayor que otras, que en la siguiente sección entenderemos porqué es así. Sin embargo, se observa que mayoritariamente se obtiene un mayor ahorro en potencia, de una frecuencia a otra, reduciendo las primeras frecuencias.

Estas gráficas que se han analizado individualmente se pueden resumir de forma global en la figura 10.4, que muestra esta variación que hemos visto para todas las aplicaciones.

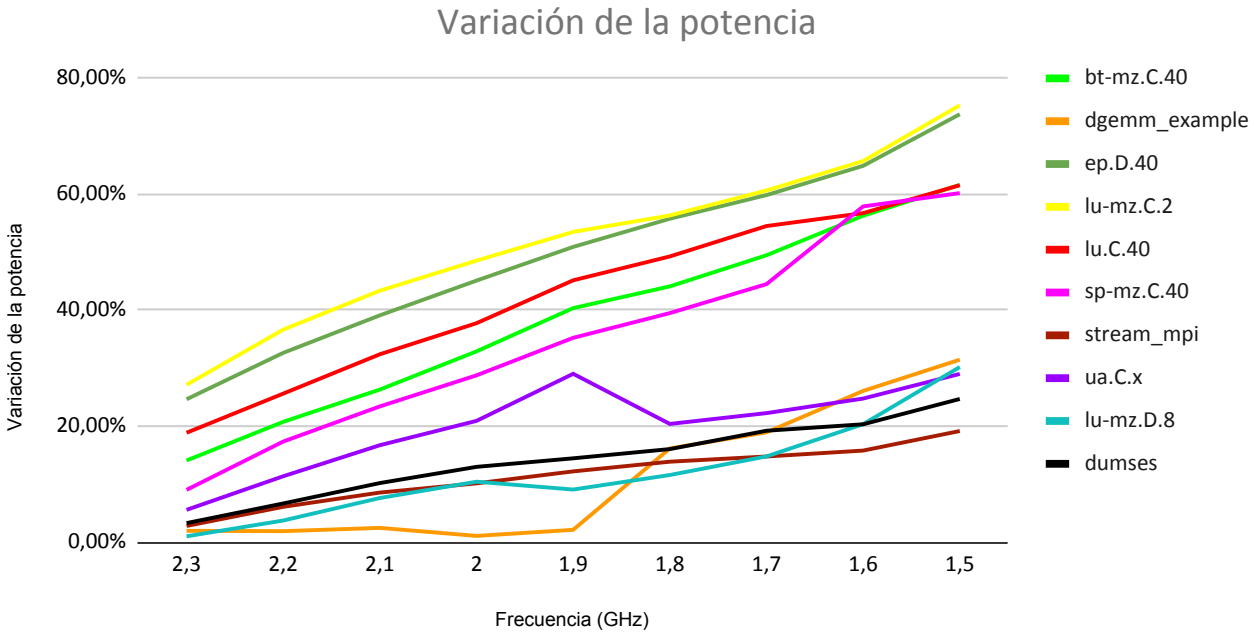


Figura 10.4: Variación en la potencia en las aplicaciones respecto 2,4 GHz.

En conclusión, vemos que la variación de la potencia no sólo depende de la frecuencia a la que se está ejecutando y de la reducción en p-state que se le aplica, sino que también influye la propia aplicación. No obstante, la mayoría tienen en común que se consigue un mayor ahorro en potencia reduciendo en las frecuencias más altas.

10.4. Métricas básicas

En esta sección se muestran las características de las aplicaciones seleccionadas, en que los datos corresponden a la misma ejecución que la sección anterior. Las principales métricas a tener en cuenta de una aplicación son los ciclos por instrucción (CPI) y el ancho de banda, ya que estos datos nos informan sobre sus características, con las que nos podemos hacer una idea de qué tipo de aplicación es cada una. Seguidamente, en las figuras 10.5 y 10.6 se observan, respectivamente, el CPI y ancho de banda para cada una de las aplicaciones estudiadas.

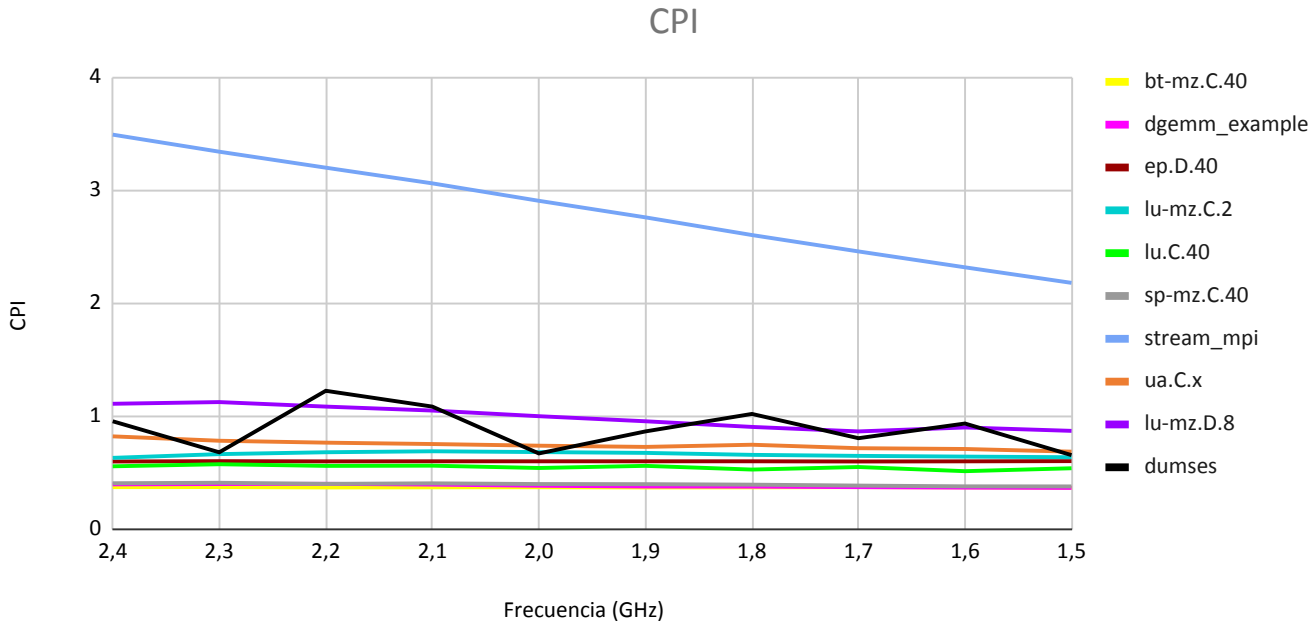


Figura 10.5: Caracterización de las aplicaciones: CPI.

El CPI son los ciclos necesarios para completar la ejecución de una instrucción, por lo que reducir la frecuencia implicará disminuir el número de ciclos por tiempo. Las aplicaciones que cuentan con un CPI elevado son aquellas que realizan una cantidad considerable de accesos a memoria. Al reducir la frecuencia únicamente hemos decrementado la frecuencia del procesador, por lo que la frecuencia de la memoria no se modifica. Por ello, al reducir la frecuencia se realizarán menos peticiones a memoria por segundo.

En la figura 10.5 se observa que hay tres aplicaciones que cuentan con el CPI más elevado: STREAM, LU-MZ.D y DUMSES. En primer lugar llama la atención el comportamiento de STREAM en que, a diferencia de que la mayoría a penas presentan cambios en su CPI, tiene un comportamiento notoriamente decreciente al reducir la frecuencia. Esto es debido a que STREAM es un *benchmark* diseñado únicamente para realizar accesos a memoria, por lo que la mayor parte del tiempo son ciclos perdidos debido al cuello de botella que representa la velocidad de la memoria. Vemos pues que STREAM muestra la proporción entre la velocidad del procesador y la memoria, en que si fueran a la misma velocidad el CPI sería de 1, o si la memoria fuera la mitad de rápida necesitaría dos ciclos para completar una instrucción. Por ello, al reducir la frecuencia del procesador, hará que la CPU sea más lenta, igualando así las velocidades entre procesador y memoria.

Por otro lado, también sorprende el comportamiento irregular que presenta la aplicación DUMSES. No obstante, es muy difícil de determinar la causa ya que es la única aplicación analizada que es una aplicación real, a diferencia del resto que son *benchmarks* o *kernels*, las cuales están pensadas para ejecutar unas funcionalidades concretas.

Finalmente, el resto de aplicaciones apenas sufren cambios ya que su CPI depende tanto de la CPU como de la memoria, a diferencia de STREAM que dependía totalmente de la memoria. En resumen, vemos que cuanta más dependencia haya sobre la memoria, más afectado se verá el CPI al reducir la frecuencia.

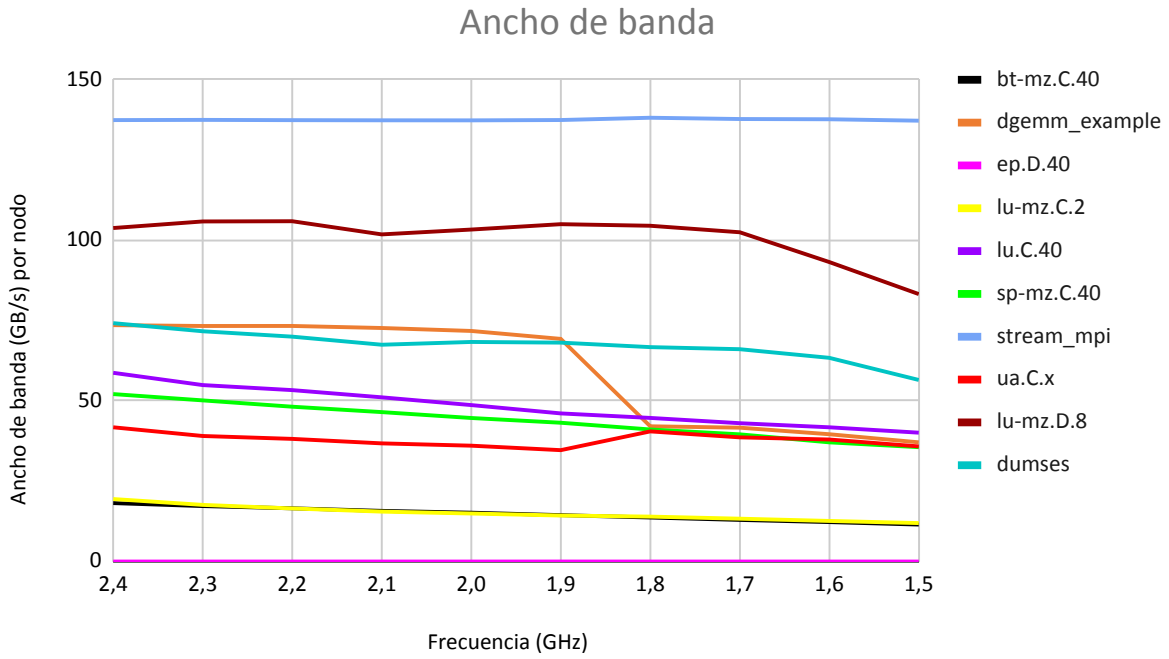


Figura 10.6: Caracterización de las aplicaciones: Ancho de banda.

El ancho de banda también es un indicador del uso de la memoria que está haciendo una determinada aplicación. En la figura 10.6 se muestra el ancho de banda obtenido por cada frecuencia a la cual se ha ejecutado cada aplicación.

En general, el ancho de banda se mantiene bastante constante a pesar de reducir la frecuencia ya que se siguen generando peticiones a memoria, sólo que se generan menos peticiones por segundo, pero no suele apreciarse un cambio evidente. Sin embargo, en la gráfica vemos que la aplicación DGEMM sí muestra un cambio notable a partir de un determinado rango, pero se trata de un caso especial ya que es una aplicación que trabaja con instrucciones vectoriales AVX-512.

Con todo esto podemos englobar la información de caracterización en una gráfica conjunta, que se muestra en la figura 10.7, en la que se puede apreciar de mejor forma la relación entre el CPI y el ancho de banda en una aplicación. Con las barras azules se representa el ancho de banda (GB/s) y con la línea de color naranja el CPI. El eje vertical a la izquierda corresponde al ancho de banda y el eje vertical en el lado derecho corresponde al CPI. Estas métricas se han obtenido ejecutando las aplicaciones a 2,4 GHz.

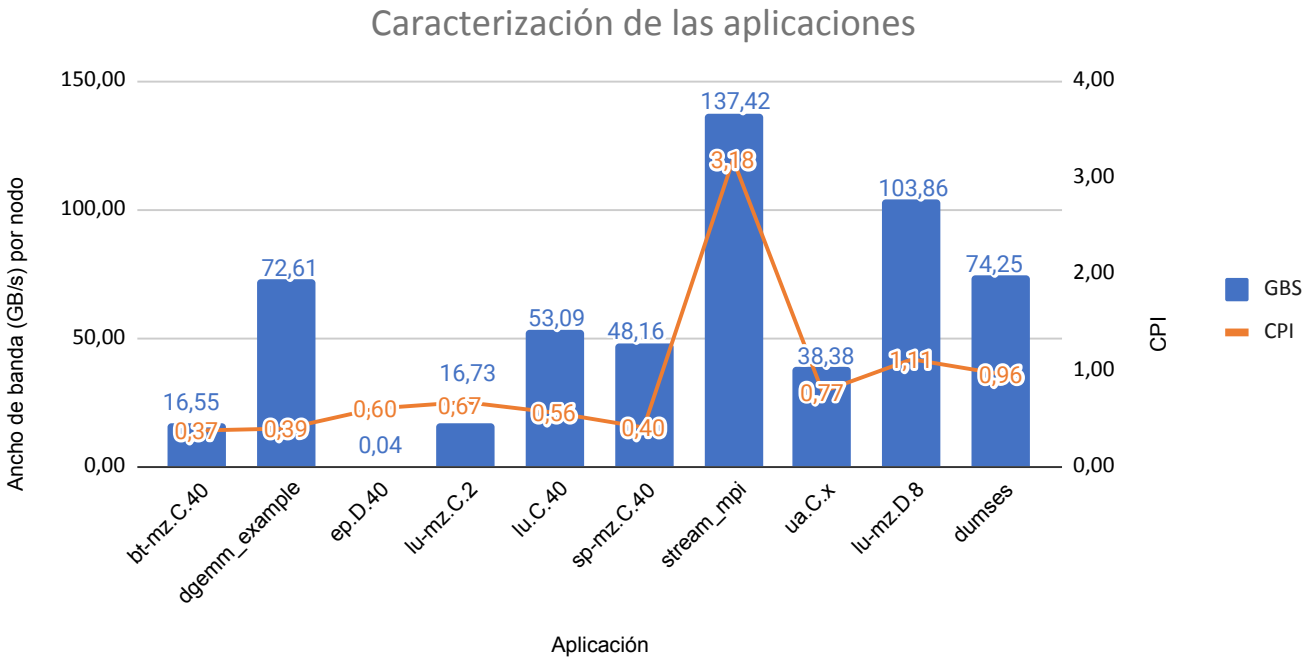


Figura 10.7: Caracterización de las aplicaciones: Ancho de banda y CPI.

Por medio de esta gráfica vemos que hay tres casos claros de aplicaciones intensivas en memoria, STREAM, LU-MZ.D y DUMSES, ya que presentan un gran ancho de banda y un CPI elevado, el cual indica que se necesitan más ciclos para ejecutar una instrucción, provocado por los accesos a memoria. Contrariamente se encuentra el *kernel* BT-MZ.C como un caso inequívoco de aplicación intensiva en cálculo, con un CPI y ancho de banda bajos. Por otro lado, vemos que el resto de aplicaciones representan un *mix*, en que su clasificación entre intensiva en memoria o en cálculo no es tan evidente.

En resumen, hemos visto que el impacto que se produce en la potencia al variar la frecuencia es mucho mayor en aplicaciones intensivas en CPU comparado con las aplicaciones intensivas en memoria. Por ejemplo, hemos visto que la aplicación DUMSES obtiene un 24 % de ahorro en potencia si se ejecuta con la frecuencia más baja. En cambio, con la aplicación BT-MZ, con solo reducir un p-state, ya obtiene un 14 % de ahorro en potencia.

Además, cuando la aplicación se está ejecutando con la librería de EAR, esta diferencia es aún mayor ya que las aplicaciones intensivas en memoria se estarán ejecutando con frecuencias bajas donde el impacto en potencia es aun menor.

Capítulo 11

Propuesta de optimizaciones

En este capítulo se presentan los diferentes cambios en el diseño e implementación que se han aplicado al Global Manager. Se comienza solucionando los problemas conocidos que se detectaron en el capítulo 9 y añadiendo algunas mejoras de funcionalidades más relevantes. Luego se muestra un conjunto de optimizaciones para controlar fases donde la energía crece muy rápidamente (fases de ramp up) y fases donde la energía se reduce muy bruscamente (fases de ramp down). Cuando la pendiente de estas fases es muy pronunciada, necesitamos una respuesta rápida del EARGM para evitar: o bien pasar el límite de energía en el ramp up, o bien tener una configuración poco eficiente al tardar mucho tiempo en recuperar una configuración adaptada al contexto (en las fases de ramp down). La evaluación del componente implementado con estas propuestas se encuentra en el capítulo 12. Las correcciones y optimizaciones, a medida que se exponen, son mejoras acumulativas.

11.1. Corrección de la configuración inicial

En el diagrama de la figura 9.4, perteneciente al capítulo 9, se detectaron en los arcos de color rojo acciones a corregir de inmediato. Carecía de sentido aplicar ajustes mientras se permaneciera en un mismo estado, ya que eso indicaba que los ajustes no estaban teniendo el impacto suficiente para las características de la carga de trabajo del sistema, por lo que se deberían aplicar otros más agresivos. Además, tampoco era correcto que al conseguir reducir el nivel de criticidad se aplicaran más medidas correctivas, pues es una señal de que la nueva configuración está teniendo un efecto sobre el sistema.

Este comportamiento se ha decidido modificarlo en función del estado del cual se proviene, por lo que se deberá recordar el último nivel de criticidad del sistema que haya habido en cada periodo de evaluación. De este modo se aplica un filtro en las acciones en que únicamente se ejecutarán nuevos ajustes si se viene de un estado con un nivel de criticidad inferior. A continuación se muestra en pseudocódigo el nuevo comportamiento entre niveles:

```
if (W1_cond) then {
    if (last_level == NO_PROBLEM) aplica_acciones();
}

if (W2_cond) then {
    if ((last_level == NO_PROBLEM) or (last_level == WARNING_1))
        aplica_acciones();
}

if (PANIC_cond) then {
    if (last_level != PANIC) aplica_acciones();
}
```

Otro de los problemas que presentaba el diseño inicial era que los ajustes se realizaban de forma acumulativa sobre el valor resultante de la última configuración. Esto se ha solucionado aplicando los distintos ajustes sobre la configuración inicial. Para ello se deberá guardar la configuración original al iniciar el servicio.

Por otro lado, otro aspecto que llamaba la atención era que los ajustes que se aplicaban eran bastante agresivos. Por ejemplo, para el nivel de WARNING 2 se reducían 2 p-states y para PANIC, 3. Disminuir 2 o 3 puntos de la frecuencia es una cantidad considerable, en que algunos *workloads* salen penalizados fuertemente en rendimiento. Como vimos en el capítulo 10, mayoritariamente se obtenía el mayor ahorro en potencia reduciendo los primeros p-states.Con todo esto se ha decidido que en su lugar el nivel de WARNING 2 reduzca 1 p-state y PANIC reduzca 2. Al reducir la frecuencia, igual que en el diseño original, se reduce tanto para la frecuencia máxima como para la frecuencia por defecto. En cuanto al threshold se siguen aplicando los mismos valores.

Finalmente, también se ha añadido la funcionalidad de restaurar la configuración inicial en caso de que el *daemon* reciba un signal[1] `SIGINT`. Esto, pues, se trata de una mejora de una funcionalidad.

De acuerdo con la directora del proyecto, para este punto no era necesario realizar una evaluación explícitamente ya que eran ajustes que se debían realizar obligatoriamente.

11.2. Ramp up

El nivel de *warning* NO PROBLEM indica que el estado en el sistema es correcto, por lo que no toma ninguna acción. Sin embargo, a partir de ahora se aprovecha este estado para ir restaurando la configuración inicial paulatinamente, vimos pues que era otro de los problemas que presentaba el diseño inicial. Se irá restaurando cuando hayan pasado el número de periodos T1 que conforman un periodo T2 y durante todos estos periodos hayan

[1]Un signal es un evento que recibe un proceso. `SIGINT` está reprogramado para que finalice su ejecución.

estado en nivel NO PROBLEM. Esta restauración gradual de la configuración comienza recuperando el threshold completamente, y luego los p-states se recuperan uno por uno.

Para que el Global Manager reaccione más rápidamente al producirse subidas prolongadas del consumo energético, se ha incorporado esta optimización, lo cual se traduce en nuevas condiciones para controlar los incrementos en energía que suceden en un sistema. Consistirá en acortar los ciclos de grace period cuando se detecte que las acciones aplicadas no están teniendo ningún efecto. Para comprobar que no están teniendo impacto deberán haber pasado un mínimo de tiempo, en este caso se ha decidido un 30 % de los periodos del grace period, y su comportamiento sea incremental, ya que eso implicará que los ajustes no están funcionando y necesita otros más agresivos. Por ejemplo, si nos encontramos en el nivel de WARNING 1, hemos incrementado el nivel de *warning* (WARNING 2 o PANIC) y ha pasado un mínimo de un 30 % de los grace periods, abandonaremos este periodo de gracia y se aplicarán los ajustes correspondientes al nivel en el que esté. Lo mismo sucederá si estamos en WARNING 2 y pasamos a PANIC.

```
if ((last_level == WARNING_1) && ((current_level == WARNING_2) ||
    (current_level == PANIC)) && grace_periods_active &&
    grace_periods_done > 30%){
        stop_grace_periods();
        aplica_acciones_estado_actual();
}

if ((last_level == WARNING_2) && (current_level == PANIC) &&
    grace_periods_active && grace_periods_done > 30%){
        stop_grace_periods();
        aplica_acciones_PANIC();
}
```

En la figura 11.1 se muestra un diagrama de estados con todos los cambios mencionados hasta el momento.

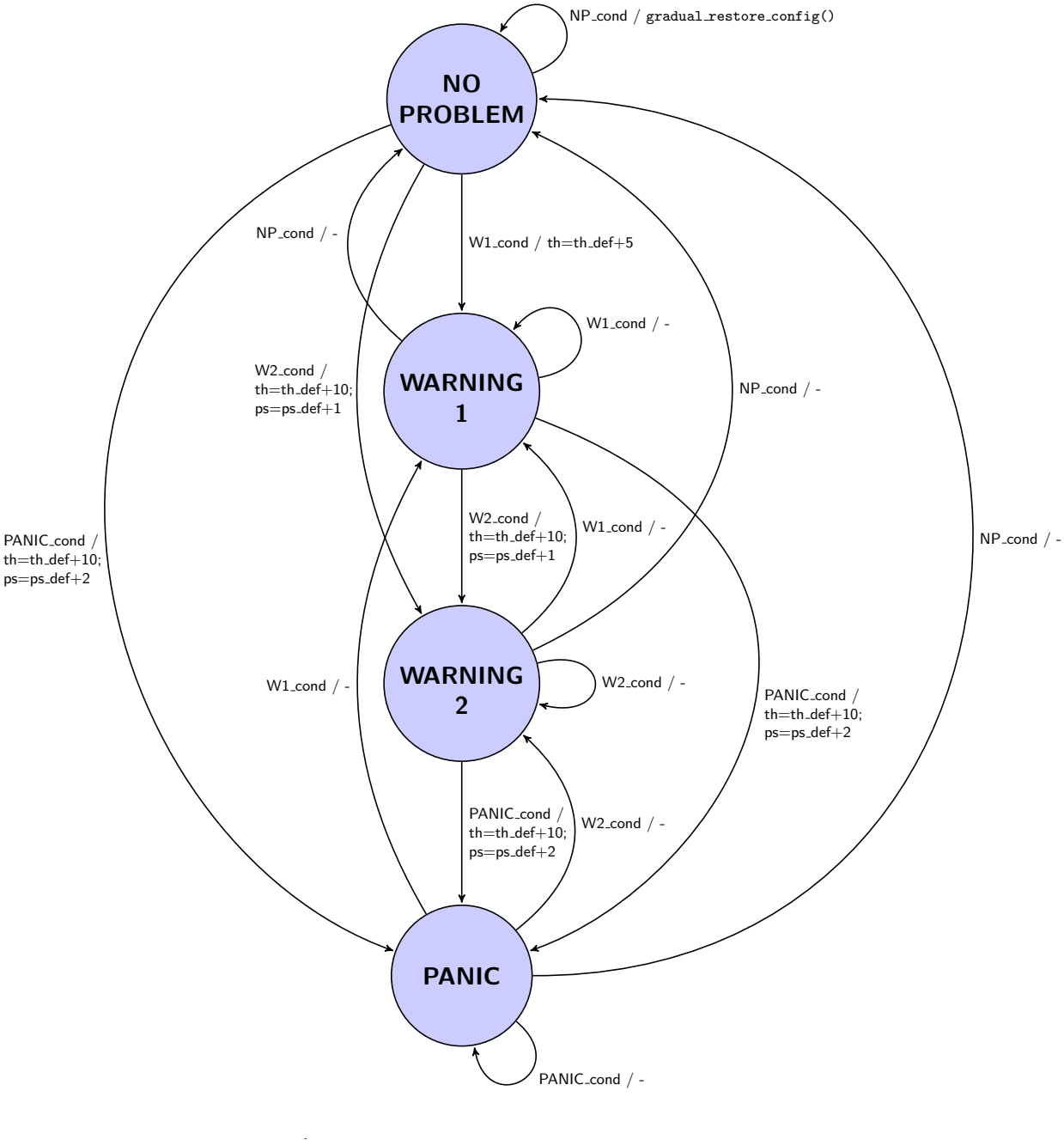


Figura 11.1: Diagrama de estados del Global Manager con optimización ramp up.

11.3. Ramp down

La optimización de ramp down consiste en detectar y controlar las fases donde la energía se reduce muy drásticamente, ya que eso implica pérdida de rendimiento. El objetivo es detectar lo antes posible situaciones en las que se permanece muy por debajo de los límites, debido a ajustes previos en la configuración, y recuperar de forma progresiva el estado por defecto del sistema. Para ello, si se reducen dos niveles de *warning* o más, se considera un descenso brusco. Por ejemplo, si partimos del estado PANIC y al aplicar la nueva configuración disminuimos a niveles de WARNING 1 o NO PROBLEM, se abandonará el periodo de grace periods para facilitar la fluidez de decisiones entre estados. Sucederá de la misma manera si aplicamos ajustes del estado de WARNING 2 y reducimos hasta NO PROBLEM.

```
if ((last_level == PANIC) && ((current_level == NO_PROBLEM) ||
    (current_level == WARNING_1)) && grace_periods_active){
        stop_grace_periods();
}

if ((last_level == WARNING_2) && (current_level == NO_PROBLEM) &&
    grace_periods_active){
        stop_grace_periods();
}
```

Por otro lado, para evitar el uso de los recursos del sistema muy por debajo de sus posibilidades, se ha incorporado un límite inferior para evitar que se apliquen ajustes demasiado agresivos que conduzcan a que transcurra una gran parte del tiempo desaprovechándolos. Este límite inferior es una extensión del estado inicial del EARGM donde no existía el concepto del desaprovechamiento de los recursos, ya que se asumía que la configuración del sistema no llegaría a esa situación. Este límite inferior se ha fijado en un 5 % adicional inferior respecto al correspondiente al nivel de WARNING1, siendo en nuestro caso este límite inferior un 80 % de la energía consumida. En caso de estar consumiendo por debajo de ese límite, significa que los ajustes han sido demasiado agresivos para la carga del sistema y, por consecuencia, se están desaprovechando los recursos, por lo que se restaurará toda la configuración por defecto.

```
if (% energía < (W1_cond-5%)) restaura_configuración_por_defecto();
```

Finalmente, no se incorpora un nuevo dibujo del diagrama de estados ya que es el mismo que para la optimización anterior (figura 11.1), a excepción que ahora las acciones en el estado NO PROBLEM ya no consistirá únicamente en restaurar gradualmente la configuración inicial, sino que además la restaurará por completo si se sobrepasa este nuevo límite inferior.

Capítulo 12

Evaluación

El presente capítulo es de una gran relevancia ya que es donde se pone a prueba el funcionamiento del Global Manager. Para ello se ha diseñado un conjunto de *workloads* que emulan diversas situaciones que pueden suceder en un sistema para así poder ver la reacción del componente ante ellas. En las primeras secciones se describen la configuración del sistema y las cargas de trabajo con las que se ha hecho el estudio. Por otro lado, las siguientes secciones están dedicadas a observar las decisiones que toma el EARGM y ver cómo afecta en el sistema para cada una de las propuestas de optimizaciones que vimos en el capítulo anterior.

12.1. Configuración del sistema

Los experimentos se han llevado a cabo en la misma máquina y entorno que se presentó en la sección 10.1. Para evaluar el Global Manager se ha diseñado una gran variedad de cargas de trabajo que se presentan en la siguiente sección, combinando múltiples características, en que todas ellas se han ejecutado utilizando la política `MIN_TIME_TO_SOLUTION`. La política utiliza la frecuencia nominal del procesador (2,4 GHz) y se ha configurado con una frecuencia por defecto a 2,1 GHz y un threshold del 70 %, siendo conservador a los ajustes definidos por la máquina de trabajo.

El EARGM se ha configurado con un periodo T1 = 1 minuto y un periodo T2 = 5 minutos. La ratio es de un 20 %, por lo que se verán los cambios rápidamente. En un sistema real estos tiempos se adaptarían a las necesidades del centro de computación. Con el número de grace periods también se ha sido conservador, equivaliendo a 10 periodos T1 aunque, como veremos a lo largo del capítulo, es un valor muy dinámico.

12.2. Workloads

La evaluación del componente Global Manager ha consistido en dos etapas. En la primera se han llevado a cabo unos test de validación, en la que se han realizado algunas pruebas con pocos nodos, en concreto 4 (160 *cores*). Después de haberme asegurado de que esta etapa funcionaba correctamente, se ha pasado a una segunda etapa donde se han realizado unos test de rendimiento, en que los experimentos se han efectuado con bastantes recursos más (20 nodos y un caso con 26, utilizando 800 y 1024 *cores* respectivamente), en que se ve de una forma más realista que el EARGM está actuando razonadamente. Los datos con los que se han realizado las debidas evaluaciones han sido extraídos con el comando `ereport` que proporciona EAR, como vimos en el capítulo 8.

Basado en los conocimientos previos a este capítulo se han definido un conjunto de cargas de trabajo para cada etapa de validación, ya que como vimos anteriormente uno de los factores que influye en el consumo de la potencia es el tipo de aplicación. Por lo tanto, el objetivo es evaluar el impacto en función la carga que hay en el sistema tanto por el tipo de aplicación (intensivas en CPU o en memoria) como por el porcentaje de carga del sistema (carga constante o variable).

A continuación se muestra mediante tablas las configuraciones de cada experimento realizado en este capítulo de evaluación. En la tabla 12.1 se especifican los experimentos que conforman los test de validación (experimentos funcionales) y en la tabla 12.2 los que se han utilizado para la etapa de validación a través de test de rendimiento. En ambas tablas se indica un identificador de *workload* en que el primer dígito corresponde al número de nodos para facilitar su identificación. En las tablas también se indican cuantas y qué aplicaciones se han utilizado, y entre paréntesis su configuración en términos de procesos (MPIs) y threads por proceso (OpenMPs). Finalmente, se añade para cada experimento un comentario que describe su característica.

Como modo aclaratorio, en las tablas se ha utilizado el formato número de *jobs* × número de nodos por *job* para representar cuántos *jobs* con cuántos nodos han sido necesarios por aplicación para crear el *workload*. Por ejemplo, para el experimento 4.5 la descripción de su *workload* (1x2xSP-MZ.C (80x1) + 1x2x[LU-MZ.D (16x5) / BT-MZ.C (80x1)]) significa que hay una parte constante para toda la ejecución formada por un único *job* constituido por la aplicación SP-MZ.C, configurada con 80 MPIs y 1 OpenMP, que se ejecuta en 2 nodos, y otra parte dinámica en el tiempo que también es un único *job* ejecutado en 2 nodos que alterna una aplicación LU-MZ.D (con 16 MPIs y 5 OpenMPs) y una BT-MZ.C (con 80 MPIs y 1 OpenMP).

Es importante comentar que los experimentos con carga dinámica ejecutan dos veces una misma aplicación, comenzando por la primera que viene indicada en la tabla. Siguiendo con el ejemplo anterior, se comenzará ejecutando la aplicación LU-MZ.D. Cuando finalice se ejecutará la aplicación BT-MZ.C y, cuando ésta última finalice volverá a ejecutarse la aplicación LU-MZ.D y, finalmente, otra vez la BT-MZ.C.

ID	Aplicaciones (MPIsxOpenMPs)	Descripción
4.1	1x4xBT-MZ.C (160x1)	**Kernel intensivo en CPU**
4.2	1x4xDUMSES (128x1)	**Aplicación intensiva en memoria**
4.3	1x4xLU-MZ.D (32x5)	**Kernel intensivo en memoria**
4.4	1x2xBT-MZ.C (80x1) + 1x1xLU-MZ.D (8x5) + 1x1xLU-MZ.C (8x5)	**Mix estático**: Mezcla 50 % intensiva en CPU, 25 % en memoria y 25 % medio.
4.5	1x2xSP-MZ.C (80x1) + 1x2x[LU-MZ.D (16x5) / BT-MZ.C (80x1)]	**Mix dinámico**: Mezcla 50 % estática intensiva en CPU y 50 % dinámica en el tiempo que alterna intensivo en memoria y cálculo.

Tabla 12.1: Descripción de los test de la etapa de experimentación funcional.

ID	Aplicaciones (MPIsxOpenMPs)	Descripción
26.1	1x26xDUMSES (1024x1)	**Aplicación intensiva en memoria** ejecutada con 26 nodos.
20.1	1x20xBT-MZ.C (160x5)	**Kernel intensivo en CPU**
20.2	1x20xBT-MZ.C (160x5)	**Kernel intensivo en CPU de larga duración**: Kernel intensivo en cálculo configurado con un T2 = 10 minutos y la duración del experimento es del doble de tiempo.
20.3	5x4xLU-MZ.D (32x5)	**Kernel intensivo en memoria**
20.4	1x10xBT-MZ.C (80x1) + 1x5xSP-MZ.C (100x2) + 1x5xLU-MZ.C (20x10)	**Mix estático**: Mezcla 50 % intensiva en CPU, 25 % en memoria y 25 % medio. Sin embargo, la aplicación media se vuelve bastante de cálculo por el paralelismo.
20.5	1x10xSP-MZ.C (100x4) + 5x2x[LU-MZ.D (16x5) / BT-MZ.C (80x1)]	**Mix dinámico intensivo en CPU**: Mezcla 50 % estática intensa en cálculo y 50 % dinámica en memoria y cálculo. De este modo, en el tiempo se obtendrá un máximo del 100 % cálculo y un mínimo del 50 %, que en media resultará 75 % intensiva en cálculo.
20.6	10x1xLU-MZ.D (8x5) + 5x2x[LU-MZ.D (16x5) / BT-MZ.C (80x1)]	**Mix dinámico intensivo en memoria**: Mezcla 50 % estática intensa en memoria y 50 % dinámica en memoria y cálculo. De este modo, en el tiempo se obtendrá un máximo del 100 % memoria y un mínimo del 50 %, que en media resultará 75 % intensiva en memoria.
20.7	1x10xSP-MZ.C (100x4) + 1x10x[BT-MZ.C (100x4) / sleep]	**Mix dinámico con pérdida de carga**: Mezcla 50 % estática intensiva en CPU y 50 % dinámica que alterna intensiva en CPU y pérdida de carga (ejecutando `sleep`).

Tabla 12.2: Descripción de los test de la etapa de experimentación de rendimiento.

En los apéndices se adjuntan un par de ejemplos de los *scripts* utilizados para ejecutar los *workloads*. En el apéndice C.1 se muestra un ejemplo para ejecutar el experimento del kernel intensivo en CPU. Por otro lado, en el apéndice C.2 se muestra la parte variable de los experimentos mix dinámicos, en que se aprecia como se alternan las distintas aplicaciones.

La metodología que se ha empleado para llevar a cabo los distintos experimentos ha consistido en realizar previamente un experimento preliminar de cada tipo, el cual se tomará su consumo como referencia. Los datos obtenidos se utilizaron para definir los límites para poder forzar los niveles de *warning* en el Global Manager. Por lo tanto, se han definido diferentes niveles para estresar cada vez más al sistema y así evaluar la capacidad de reacción del componente y ver si es lo suficientemente rápido. Por ejemplo, para el caso concreto del experimento 4.1, la carga sin recibir ajustes tiene un consumo medio de potencia de 298W por nodo. Por lo tanto, para el nivel de WARNING1 se ha ajustado el límite a 340W por nodo, para WARNING2 a 325W y para PANIC a 310W.

Por un lado, para la optimización de ramp up se han evaluado todos los niveles de *warning* para todas las cargas de trabajo descritas. Sin embargo, para la optimización de ramp down sólo se han analizado aquéllas en que se ha detectado un descenso brusco de la energía y, por lo tanto, la optimización podía introducir una mejora.

Es interesante comentar que EAR cuenta con un *plugin* que permite la generación de trazas de Paraver[1], con las que se muestra de manera visual el comportamiento que ha tenido una aplicación durante su ejecución para así poder analizarla *a posteriori*. Se ha utilizado esta herramienta para validar el comportamiento de las aplicaciones.

Los resultados de cada experimento se representan mediante cinco gráficas por cada configuración definida por un nivel de *warning* y optimización. En dichas gráficas se representa en el eje de abscisas el tiempo de ejecución en intervalos de 1 minuto, los cuales corresponden con la definición del periodo T1 para los experimentos. Estas cinco gráficas muestran:

- **Energía consumida en porcentaje respecto al límite para cada experimento según el periodo T2 actual**: En esta gráfica se observa la energía consumida respecto el límite energético en forma de porcentaje, para así identificar de manera sencilla el nivel de criticidad en el cual se encuentra el sistema en un momento determinado. Los niveles de *warning* están definidos al 85 % para WARNING1, 90 % para WARNING2 y 95 % para PANIC. Estos límites están marcados a través de unas líneas fijas coloreadas de verde, naranja y rojo para los niveles WARNING1, WARNING2 y PANIC respectivamente.
- **Energía en valor absoluto medida por el EARGM respecto al periodo T2 actual**: Esta gráfica muestra la energía consumida en valor absoluto por cada periodo T2.

[1]Paraver: `https://tools.bsc.es/paraver`

- **Nivel de *warning***: En esta gráfica se muestra el nivel de criticidad en el cual se encuentra el sistema. Se ha mostrado de forma numérica donde el 0 corresponde a NO PROBLEM, 1 a WARNING1, 2 a WARNING2 y 3 a PANIC.
- **Nivel de threshold**: Por defecto los experimentos han utilizado un threshold configurado al 70 % y en la gráfica se puede apreciar como su valor varía en el tiempo en función de los ajustes que se efectuan.
- **Valores de los p-states máximo y por defecto durante la ejecución**: La gráfica muestra los valores de p-states máximo y por defecto, que corresponden a la frecuencia máxima y por defecto respectivamente, y que también varían en el tiempo. Inicialmente el p-state máximo está definido a 1 y el p-state por defecto a 4.

Para las gráficas, los primeros periodos corresponden al comienzo de la carga de trabajo, y los últimos, a su finalización. Para el experimento dinámico se han dibujado unas líneas verticales en el tiempo que aproximan el comienzo y finalización de una aplicación (el ejemplo del *script* para el test dinámico imprime la fecha y hora para una mejor aproximación al dibujar y analizar las gráficas).

También, para cada prueba se muestra la evaluación con la optimización ramp up y, seguidamente, la evaluación con la optimización ramp down para los casos afectados. Es importante comentar que para un mismo experimento se han utilizado los mismos nodos a fin de poder tener una buena comparativa entre resultados, tanto entre los distintos niveles de *warning* como entre las diferentes optimizaciones.

12.3. Experimentos de validación

En esta sección se muestra la evaluación de los distintos experimentos correspondientes a la validación con test funcionales, cuya ejecución se ha llevado a cabo con 4 nodos (160 *cores*) para comprobar su funcionamiento.

Debido a la gran cantidad de información en gráficas que presenta este trabajo, se ha decidido que las que derivan de los experimentos pertenecientes a esta sección se agreguen en el apéndice D y seguidamente se comentan los resultados obtenidos.

- **Experimento 4.1: Kernel intensivo en CPU**: Este experimento es un caso extremo de intensivo en cálculo en el que el aumento del threshold no tiene ningún impacto sobre él ya que su ratio de eficiencia es superior a la establecida por el umbral. Por otro lado, la reducción tanto en 1 como en 2 p-states (casos WARNING2 y PANIC respectivamente) tiene un gran impacto sobre la carga de trabajo, concordando con los resultados obtenidos en el análisis del capítulo 10. No obstante, debido a la gran reacción provocada por la reducción de la frecuencia, se emplea una gran parte del tiempo desaprovechando los recursos del sistema, por lo que es un claro ejemplo que

justifica la necesidad de la optimización de ramp down. Los resultados obtenidos para este experimento se encuentran en la sección D.1.

- **Experimento 4.2: Aplicación intensiva en memoria**: Este experimento se trata de una aplicación real que vimos en el estudio previo y que es intensiva en memoria. Como es de esperar en los resultados, el aumento del threshold no tiene ningún impacto sobre él ya que al ser poco eficientes trabajan con las frecuencias más bajas. Por otro lado, la reducción de la frecuencia tiene un impacto muy flojo, coincidiendo también con los resultados del estudio previo. Por lo tanto, para este tipo de *workloads* la optimización ramp down no les produce ningún efecto. Los resultados obtenidos para este experimento se encuentran en la sección D.2.
- **Experimento 4.3: Kernel intensivo en memoria**: Este experimento está compuesto por un *kernel* con la misma naturaleza que el experimento anterior (intensivo en memoria). Por lo tanto, el impacto en threshold y p-state es prácticamente el mismo, en que la reducción en frecuencia coincide con la estudiada para esta aplicación determinada en el capítulo 10. Los resultados obtenidos para este experimento se encuentran en la sección D.3.
- **Experimento 4.4: Mix estático**: Este experimento es una mezcla de aplicaciones con características diferentes. Este caso es un ejemplo de una carga de trabajo en la que el aumento del threshold tiene impacto. Por otro lado, la reducción de los p-states tiene un efecto que no es tan agresivo como para los *workloads* intensivos en cálculo, ni tan flojo como para los intensivos en memoria. Para este experimento la optimización de ramp down le beneficia en que se comienza a restaurar la configuración inicial antes. Los resultados obtenidos para este experimento se encuentran en la sección D.4.
- **Experimento 4.5: Mix dinámico**: Este experimento es una mezcla de aplicaciones en que la mitad de la carga es dinámica en el tiempo en que el aumento del threshold tiene un ligero efecto al ejecutarse la aplicación intensiva en cálculo. Por otro lado, la reducción de la frecuencia mediante los p-states no resulta un impacto tan alto como en los casos intensivos en cálculo ya que la mitad estática de la aplicación es un intermedio. La aplicación de la optimización de ramp down les beneficia en que se comienza a restaurar la configuración inicial con anterioridad. Los resultados obtenidos para este experimento se encuentran en la sección D.5. Tanto este experimento como el anterior (experimento 4.4) son los más representativos de este grupo de pruebas, aunque no se pueden usar como referencia ya que la escala es muy pequeña (4 nodos).

Para finalizar, como apunte, la optimización ramp down no se ha ejecutado para los experimentos puramente de memoria (experimentos 4.2 y 4.3), ni para los niveles NO PROBLEM ni WARNING1 ya que *a priori* conocíamos que no habría ningún efecto respecto a no activarla, por lo que no hemos gastado más horas de CPU.

12.4. Experimentos de rendimiento

En esta sección se muestra la evaluación de los distintos test correspondientes a los experimentos de rendimiento, cuya validación es ahora con 20 nodos y hay un caso con 26 (utilizando 800 y 1024 *cores* respectivamente). La duración de los experimentos ha sido de poco menos de una hora, excepto una prueba concreta que ha durado dos horas.

Esta sección también presenta una gran cantidad de información en gráficas por lo que se ha decidido explicar en este capítulo las que pueden resultar más interesantes en las siguientes subsecciones y, el resto se han agregado al apéndice E. A continuación comentamos sus resultados:

- **Experimento 26.1: Aplicación intensiva en memoria**: Se ha realizado un experimento idéntico a este para los test validación, en el que se ha comprobado, ajustando a los niveles WARNING2 y PANIC, que el comportamiento es el mismo que con los casos pequeños, en que al reducir la frecuencia apenas se produce un impacto. No obstante, como no se esperaba ninguna diferencia respecto a su test de validación correspondiente, y así ha sido, no se han repetido los experimentos ni para NO PROBLEM ni para WARNING1. Los resultados obtenidos para este experimento se encuentran en la sección E.1.
- **Experimento 20.2: Kernel intensivo en CPU de larga duración**: Este experimento se ha realizado para contemplar un caso en que la ratio entre el periodo T1 y T2 es menor que en el resto de pruebas. Se ha analizado para el caso con el nivel de criticidad más alto (PANIC) en el que se observa lo que se esperaba: los cambios en la variación energética son mas suavizados. Con el fin de tener una imagen fiable del comportamiento del EARGM, se ha extendido la duración del experimento al doble de tiempo que el resto. Al estar compuesto por una aplicación intensiva en cálculo ocurre que la reducción de la frecuencia tiene un gran impacto sobre el *workload*, por lo que también se ha analizado con la optimización ramp down. Los resultados obtenidos para este experimento se encuentran en la sección E.2.
- **Experimento 20.3: Kernel intensivo en memoria**: Este experimento está compuesto por varios *kernels* de naturaleza intensiva en memoria ya que, en caso contrario, se volvería muy eficiente y no se obtendría esta característica que buscamos para evaluar. El aumento del threshold no produce ningún impacto ya que las aplicaciones intensivas en memoria son poco eficientes, por lo que trabajan a frecuencias bajas. Por otro lado, la reducción de la frecuencia, tanto en 1 p-state como en 2, apenas tiene impacto, coincidiendo con los resultados del análisis realizado en el capítulo 10. Los resultados obtenidos para este experimento se encuentran en la sección E.3.
- **Experimento 20.4: Mix estático**: Este experimento, a diferencia de su correspondiente en los test de validación, es más eficiente, por lo que ya no comparten exactamente la misma naturaleza. El aumento del threshold, ahora, no produce prácti-

camente ningún efecto. Por otro lado, la reducción de 1 o 2 p-states es más agresiva que en el caso de los test de validación, ya que al ser ahora más eficiente, tiene una naturaleza muy parecida a los casos intensivos en cálculo. Por lo tanto, la optimización de ramp down será necesaria para los casos en que se reduce la frecuencia. Los resultados obtenidos para este experimento se encuentran en la sección E.4.

12.4.1. Experimento 20.1: Kernel intensivo en CPU

Este experimento muestra el ejemplo de un caso extremo de un *workload* intensivo en cálculo, del cual hay un test de validación con una carga de las mismas características. Para esta prueba solo se han analizado los niveles de criticidad WARNING2 y PANIC ya que no se esperaba diferencia, y no la ha habido, respecto a los resultados obtenidos con menos nodos, por lo que no se han repetido los experimentos para el resto de niveles de *warning*. Por lo tanto, se ha tomado como referencia el nivel de NO PROBLEM correspondiente al experimento de estas características de los test de validación (experimento 4.1).

Por lo tanto, el aumento del threshold no causa ningún impacto sobre la carga de trabajo ya que de por sí es muy eficiente. Por otro lado, la reducción tanto de 1 p-state como de 2 tiene un gran efecto (ver figuras 12.1 y 12.2), tal y como se analizó en el capítulo 10. Como consecuencia de este impacto tan grande, se emplea buena parte del tiempo desaprovechando los recursos ofrecidos por el sistema por lo que conduce a implementar la optimización de ramp down, la cual agiliza la restauración de la configuración inicial y así recuperar el rendimiento.

Por ejemplo, para el caso de WARNING2, en la figura 12.1 se observa que hasta el minuto 20 no se empieza a recuperar gradualmente la configuración que, hasta el minuto 26 no comienza a tener efecto. En la gráfica también se observa que desde el minuto 10 hasta el 27 se están desaprovechando los recursos del sistema. Por otro lado, la figura 12.3 muestra el resultado de aplicar la optimización ramp down en que en el minuto 10 ha recuperado completamente su configuración inicial ya que al tener un gran efecto llega a sobrepasar el límite inferior, por lo que en el minuto 12 vuelve a incrementar su consumo energético. Al retomar la configuración inicial volverá a situarse en el nivel de WARNING2, por lo que se volverán a aplicar los ajustes, y así sucesivamente. En resumen, vemos que con solamente la optimización ramp up la configuración inicial se restaura al cabo de unos 18 minutos, mientras que con la optimización ramp down lo hace en 2 minutos.

Para el caso de PANIC sucede el mismo comportamiento, como se puede observar en las figuras 12.2 y 12.4.

Finalmente, usando como referencia el límite ajustado para PANIC obtenemos para el nivel WARNING2 con la optimización ramp up un consumo energético medio del 81 %, mientras que con ramp down es del 85 %. Para el nivel de PANIC se obtiene un consumo energético medio del 80,6 % con ramp up, mientras que con ramp down se consigue un 85,2 %.

Optimización ramp up:

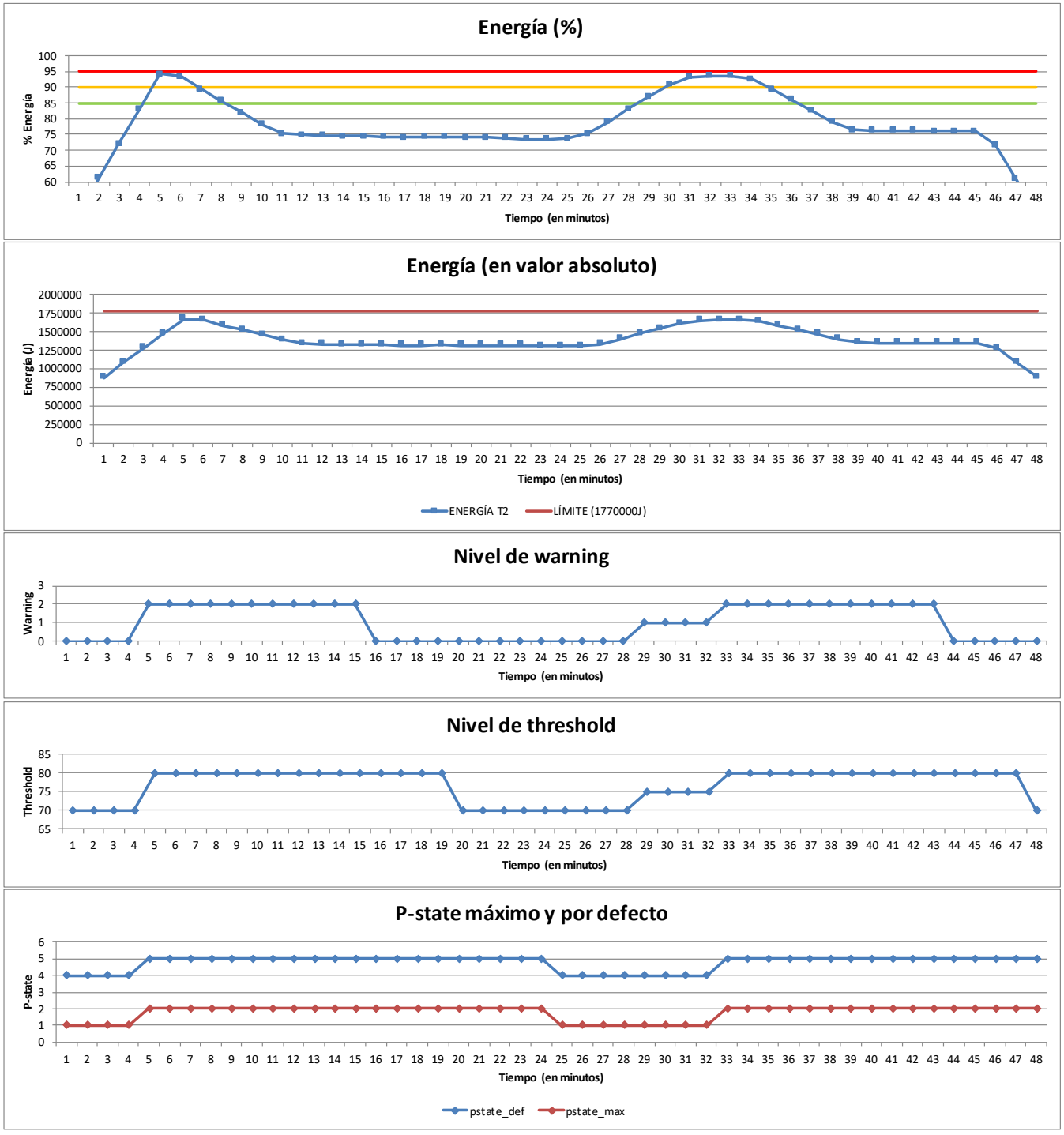


Figura 12.1: Experimento 20.1 con optimización ramp up ajustado al nivel WARNING2.

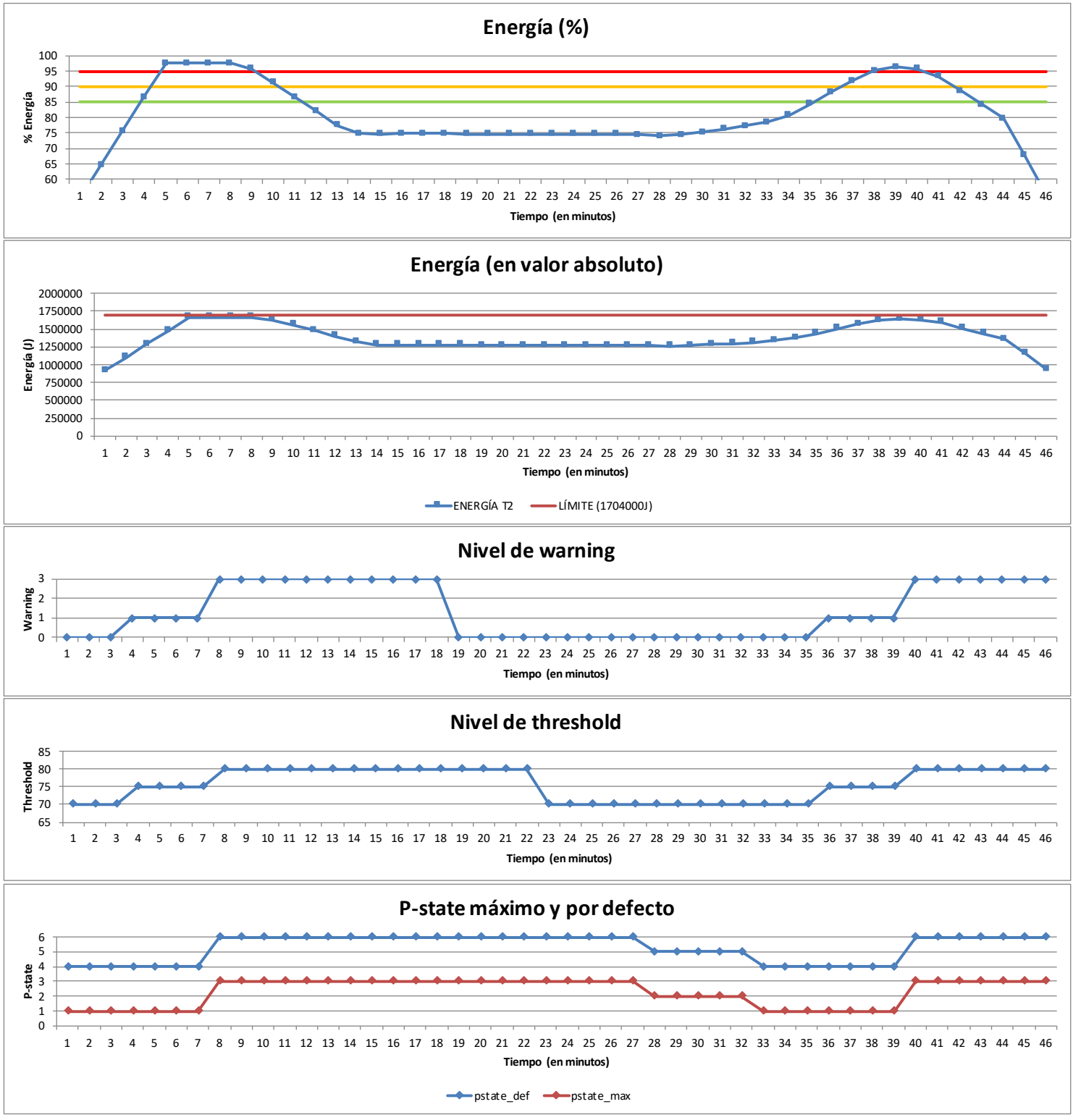


Figura 12.2: Experimento 20.1 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

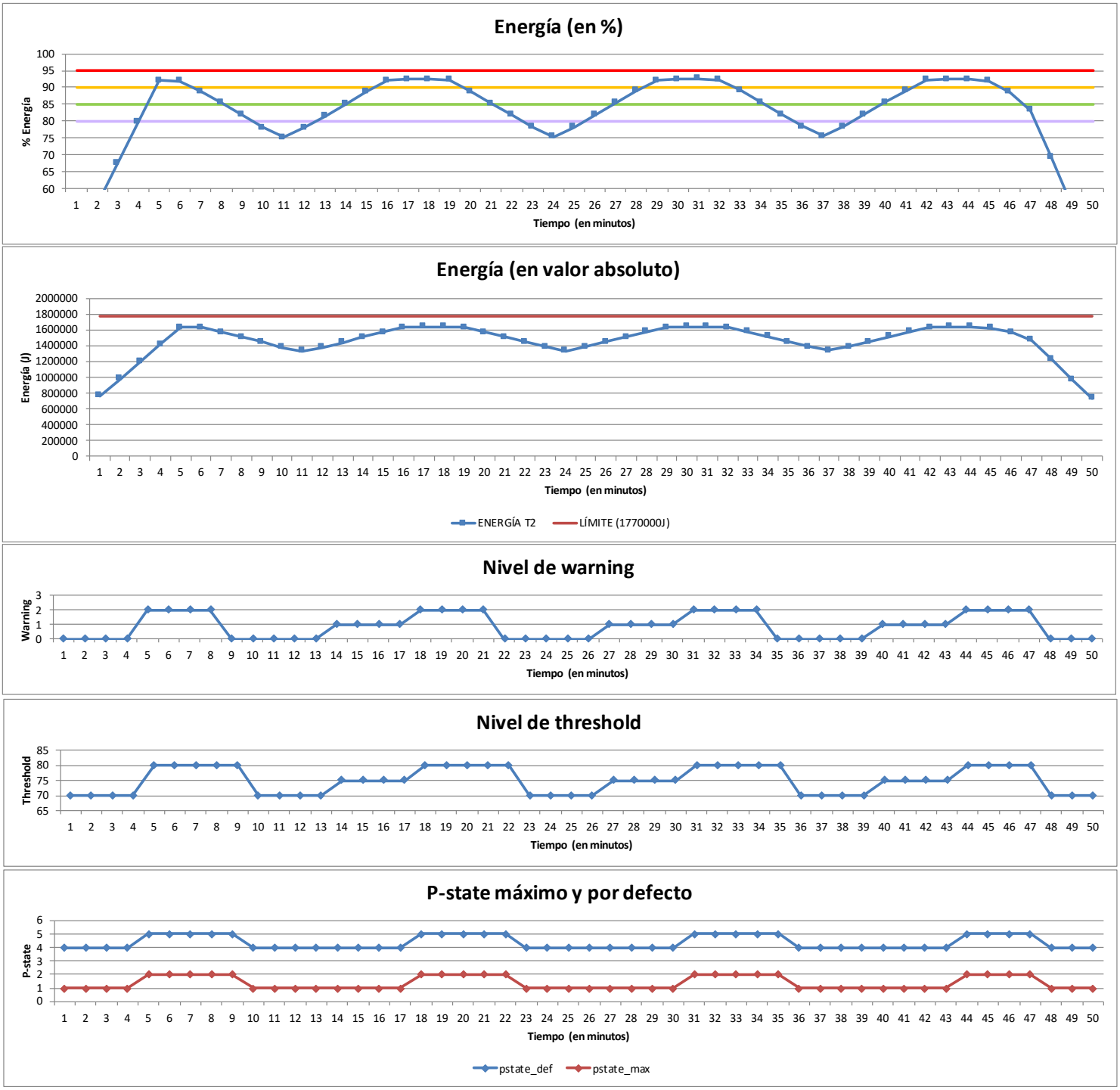


Figura 12.3: Experimento 20.1 con optimización ramp down ajustado al nivel WARNING2.

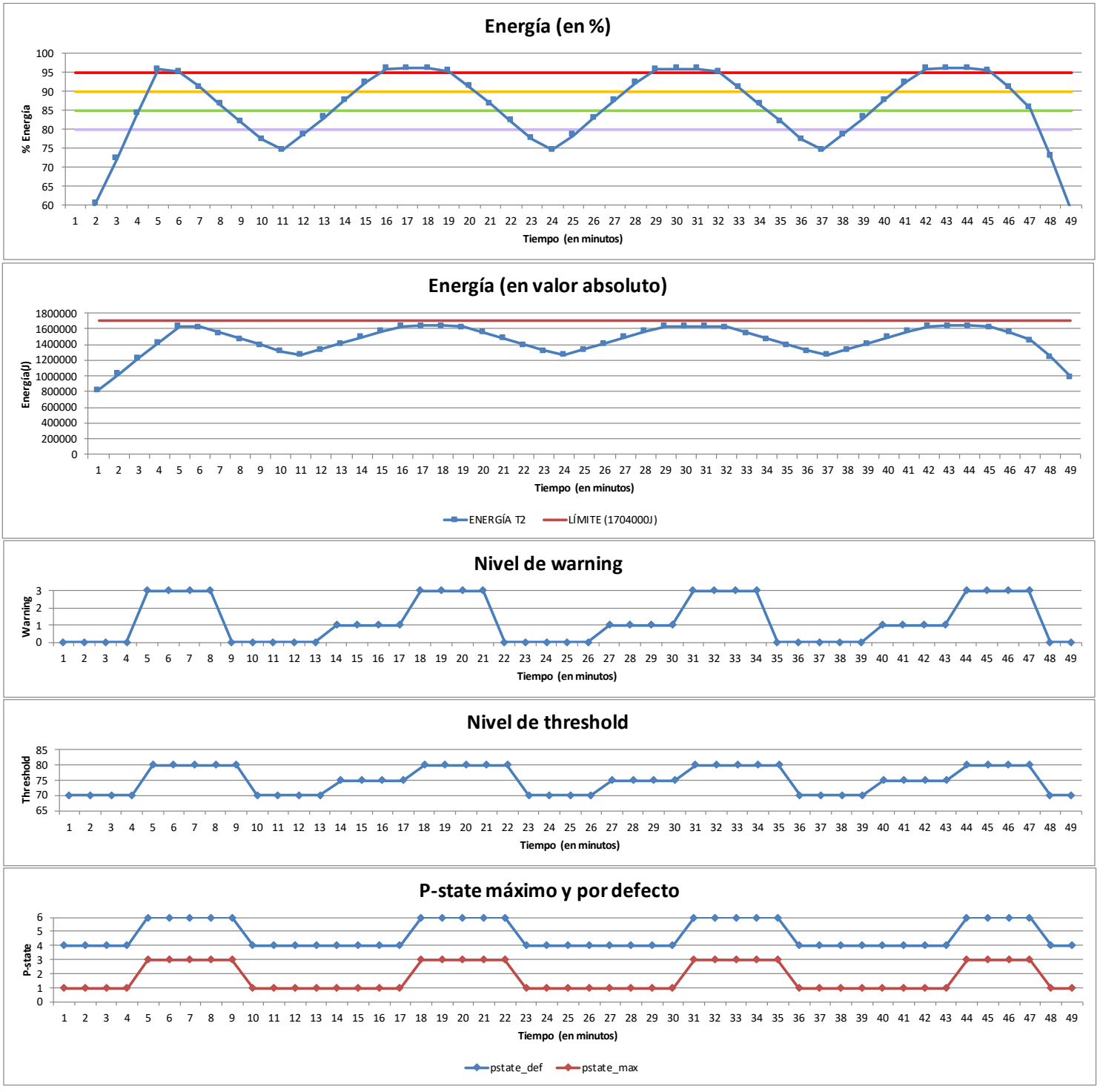


Figura 12.4: Experimento 20.1 con optimización ramp down ajustado al nivel PANIC.

12.4.2. Experimento 20.5: Mix dinámico intensivo en CPU

Este experimento muestra una carga de trabajo en que la mitad es estática e intensiva en cálculo y la otra mitad alterna en el tiempo entre intensiva en cálculo o en memoria, de modo que la carga intensiva en cálculo sea de un 75 % de media. En la figura 12.5 se muestra la ejecución de la carga ajustada a NO PROBLEM, con el fin de observar su comportamiento natural, sin modificaciones, y así observar el impacto de los ajustes.

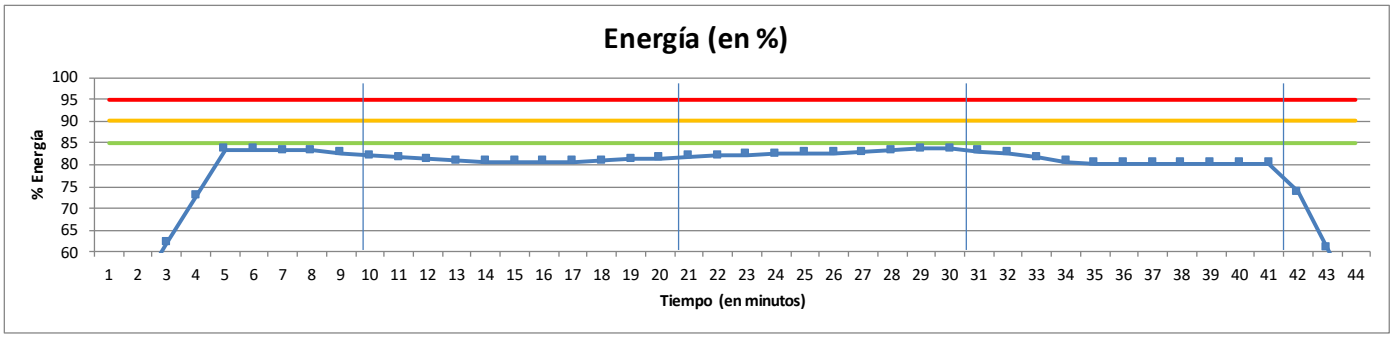


Figura 12.5: Comportamiento del experimento 20.5 sin acciones del EARGM.

Las siguientes figuras 12.6 y 12.7 muestran, respectivamente, para la optimización ramp up los niveles de criticidad WARNING2 y PANIC. En cambio, los resultados para WARNING1 se han agregado al apéndice E.5 al no considerarse un nivel crítico. Por un lado, con los casos dinámicos es más difícil determinar el impacto causado por el aumento del threshold. No obstante, al comparar los resultados obtenidos en WARNING1 con la gráfica tomada por referencia, observamos que hay un sutil efecto ya que los incrementos y decrementos energéticos son un poco más pronunciados, por lo que no vienen únicamente determinados por la variación de la carga.

Por otro lado, la reducción de 1 o 2 p-states tiene un impacto evidente aunque no es tan agresivo como sucede para los experimentos intensivos en CPU, ya que una parte del *workload* es intensiva en memoria, la cual ayuda a moderar el crecimiento y decrecimiento energético, tal y como se observa en las figuras 12.6 y 12.7. Para el caso de PANIC (figura 12.7) la optimización ramp up le favorece ya que al aplicar ajustes de niveles de *warning* inferiores (minuto 35) le causan efecto, por lo que no será necesario aplicar los más duros.

Vemos, pues, que la optimización de ramp down es conveniente también en este experimento para los niveles de WARNING2 y PANIC, y se encuentran los resultados en las figuras 12.8 y 12.9, respectivamente. En estas figuras se observa que la recuperación de la configuración inicial comienza considerablemente antes, permitiendo aprovechar más tiempo los recursos que ofrece el sistema.

Por otro lado, para el caso particular de PANIC, se restaura la configuración por defecto ya que desciende hasta atravesar el límite inferior. No obstante, acaba estabilizándose, encontrando una configuración adecuada para las distintas cargas.

Finalmente, usando como referencia el límite ajustado para PANIC obtenemos para el nivel WARNING2 con la optimización ramp up un consumo energético medio del 82 %, mientras que con ramp down es del 86 %. Para el nivel de PANIC se obtiene un consumo energético medio del 79,5 % con ramp up, mientras que con ramp down se consigue un 82,2 %.

Optimización ramp up:

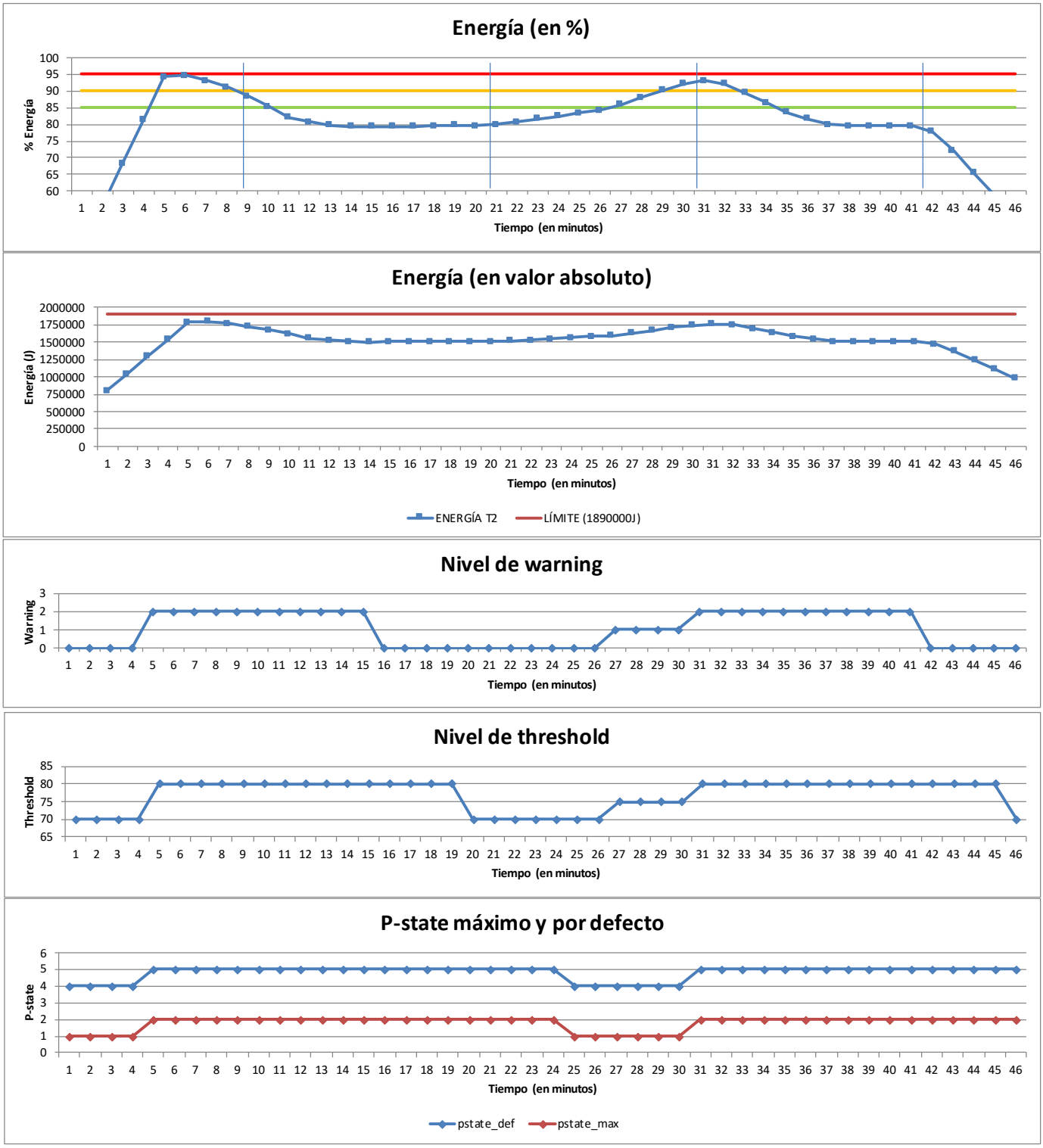


Figura 12.6: Experimento 20.5 con optimización ramp up ajustado al nivel WARNING2.

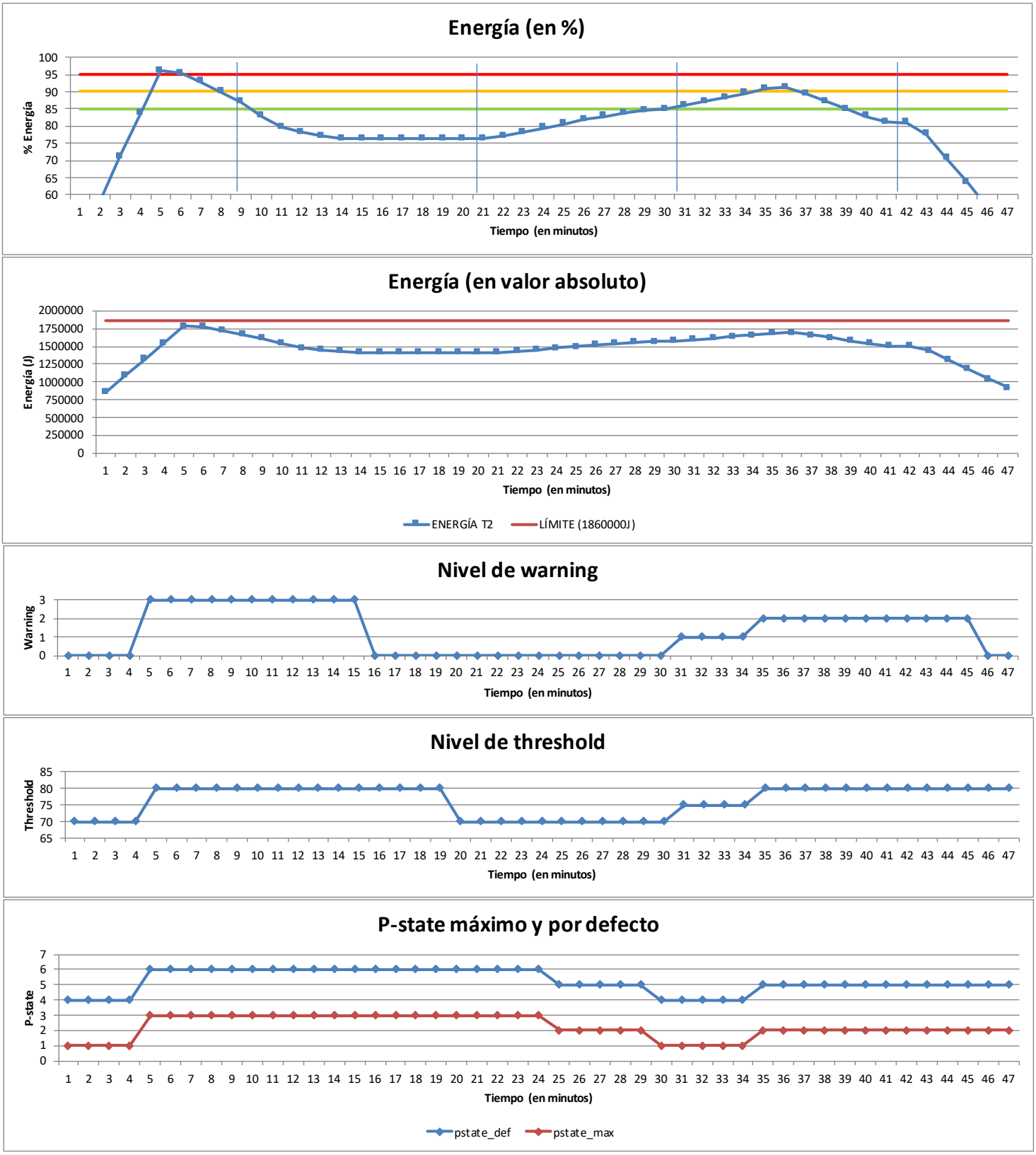


Figura 12.7: Experimento 20.5 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

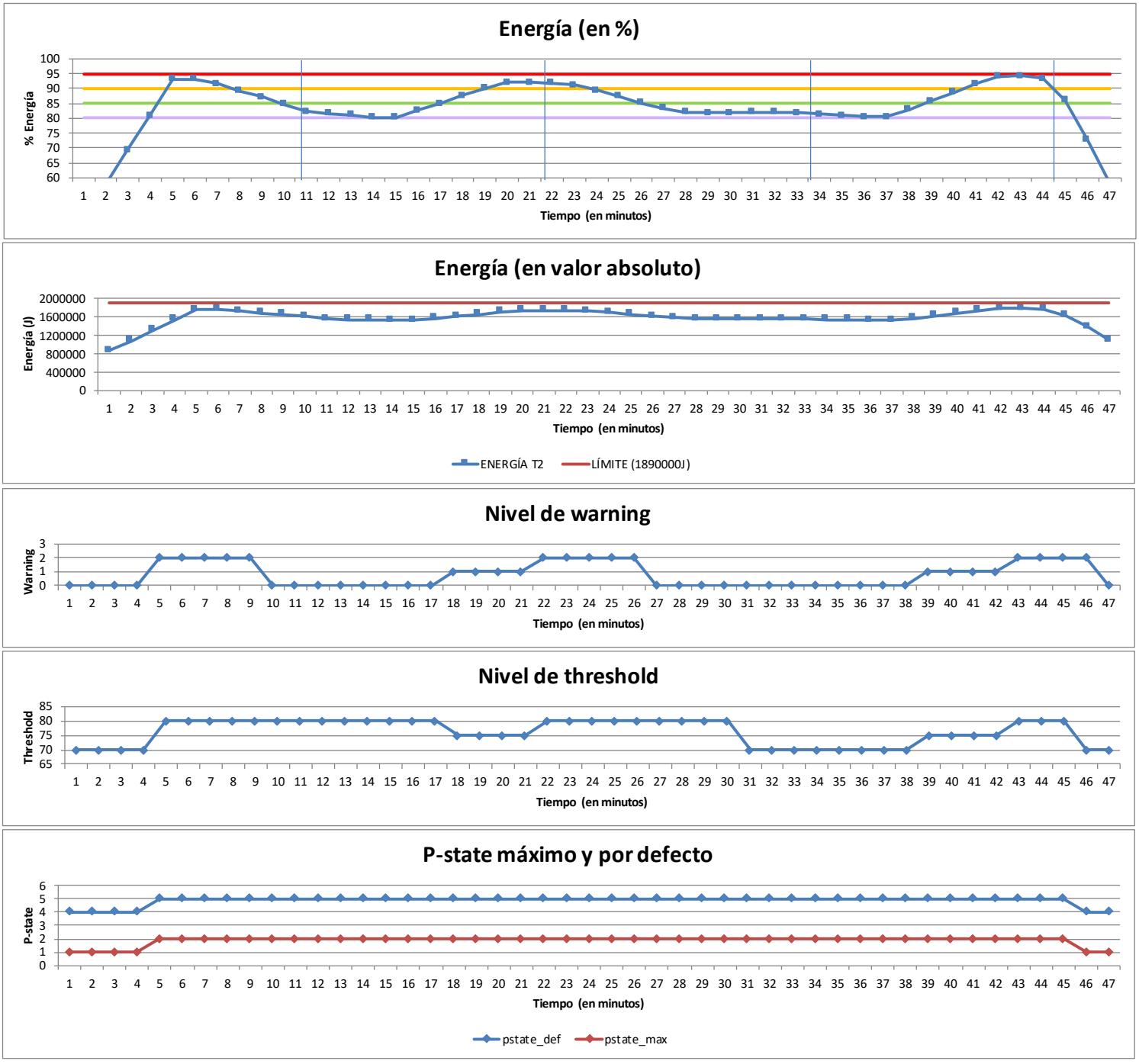


Figura 12.8: Experimento 20.5 con optimización ramp down ajustado al nivel WARNING2.

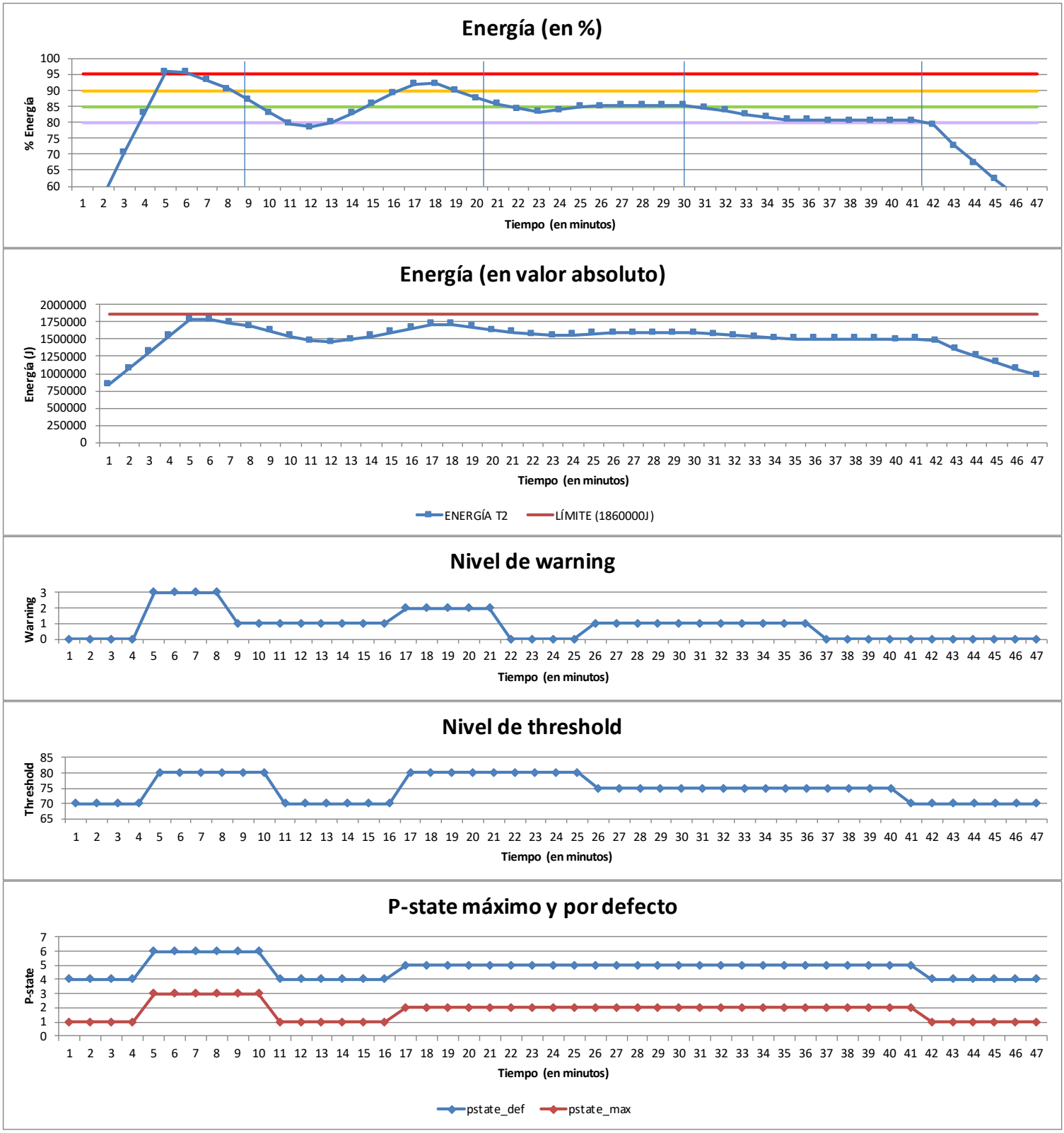


Figura 12.9: Experimento 20.5 con optimización ramp down ajustado al nivel PANIC.

12.4.3. Experimento 20.6: Mix dinámico intensivo en memoria

Este experimento es parecido al anterior pero ahora se trata de una carga de trabajo en que la mitad es estática e intensiva en memoria y la otra mitad alterna en el tiempo entre intensiva en cálculo o en memoria, de modo que la carga intensiva en memoria es ahora de un 75 % de media. La figura 12.10 muestra la ejecución de la carga ajustada a NO PROBLEM, con el fin de observar el comportamiento natural de la carga para tomarla como referencia y luego apreciar el impacto de los distintos ajustes.

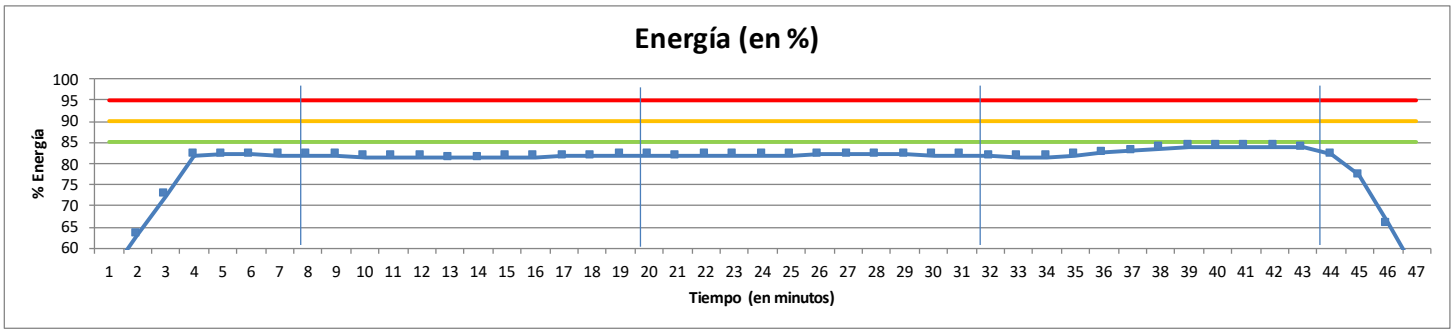


Figura 12.10: Comportamiento del experimento 20.6 sin acciones del EARGM.

Sin embargo, debido a los problemas de potencia que sufrió la máquina, este experimento se vio alterado. Posteriormente a la incidencia, los límites ajustados no concordaban con los obtenidos antes del contratiempo y se redefinieron. Antes de continuar con los experimentos comprobamos que el comportamiento no había variado, y así fue, por lo que seguimos adelante.

Las siguientes figuras 12.11 y 12.12 muestran, respectivamente, para la optimización ramp up los niveles de criticidad WARNING2 y PANIC. Igual que en el experimento anterior, los resultados obtenidos para el WARNING1 se han agregado al apéndice E.6. En primer lugar se observa que el threshold no produce ningún impacto sobre el comportamiento del *workload*. Esto era previsible ya que la gran parte de la carga de trabajo es intensiva en memoria, por lo que en general es poco eficiente y opera a las frecuencias más bajas, por lo que no le produce ningún efecto este aumento en el linde.

Por otro lado, al reducir la frecuencia se obtiene un impacto suave aunque algo más manifiesto en comparación con los experimentos puramente intensivos en memoria, ya que en esta prueba concreta hay una parte de la carga de trabajo dedicada intensivamente al cálculo, contribuyendo a que los ajustes no sean tan rígidos respecto a los cambios.

Se ha realizado el experimento con el nivel de PANIC para la optimización ramp down ya que consigue reducir un par de niveles, y se muestra en la figura 12.13. Sin embargo, no sufre prácticamente variaciones respecto a únicamente la optimización ramp up (figura 12.12) puesto que no tiene un descenso significativamente grande como para observar grandes

variaciones. Otro punto a tener en cuenta es que este experimento que cuenta con la optimización ramp down es el que vio alterado su límite de potencia.

Finalmente, usando como referencia el límite ajustado para PANIC se obtiene para este nivel un consumo energético medio del 87,4 % con ramp up, mientras que con ramp down se consigue un 88,7 %.

Optimización ramp up:

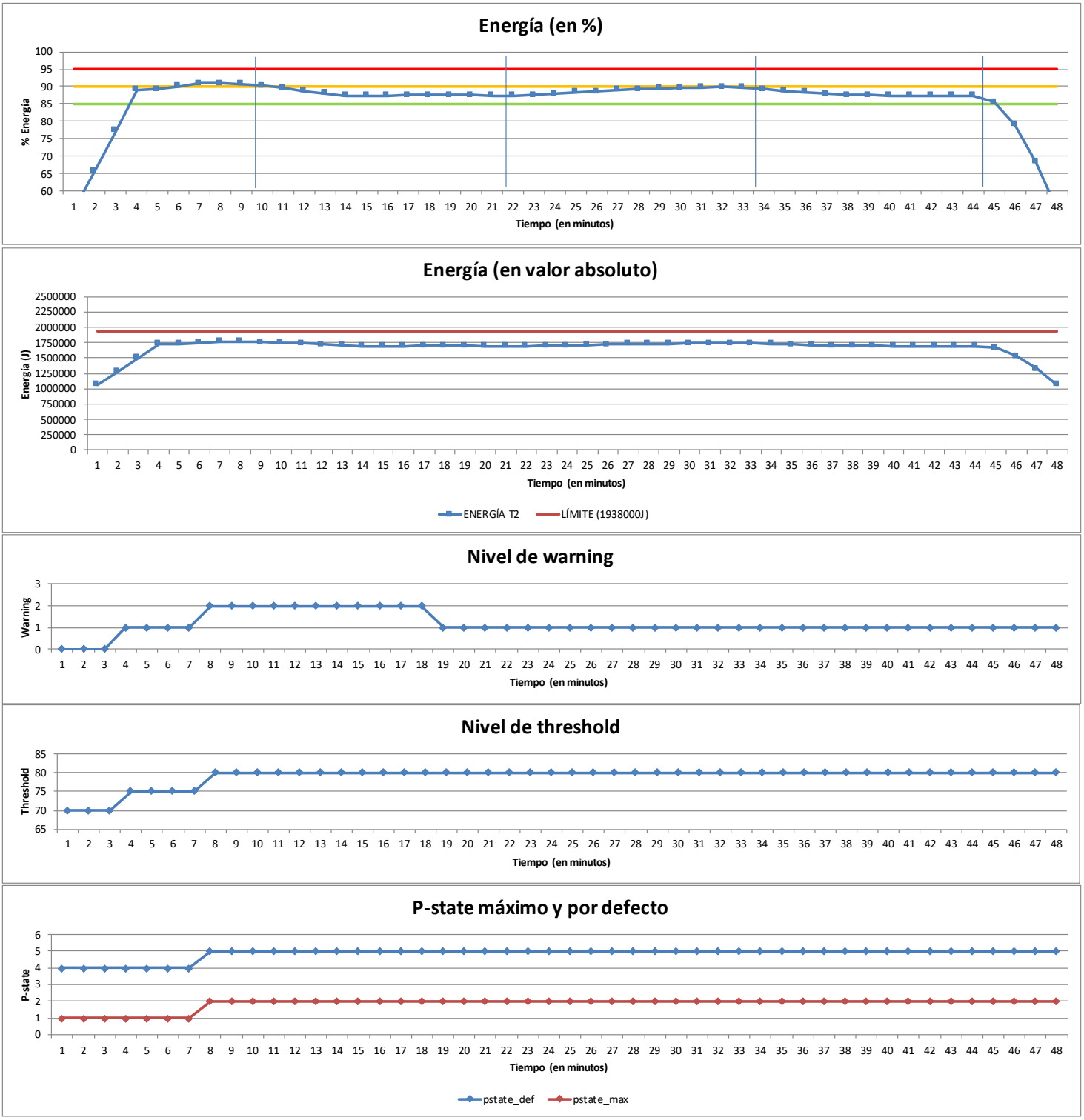


Figura 12.11: Experimento 20.6 con optimización ramp up ajustado al nivel WARNING2.

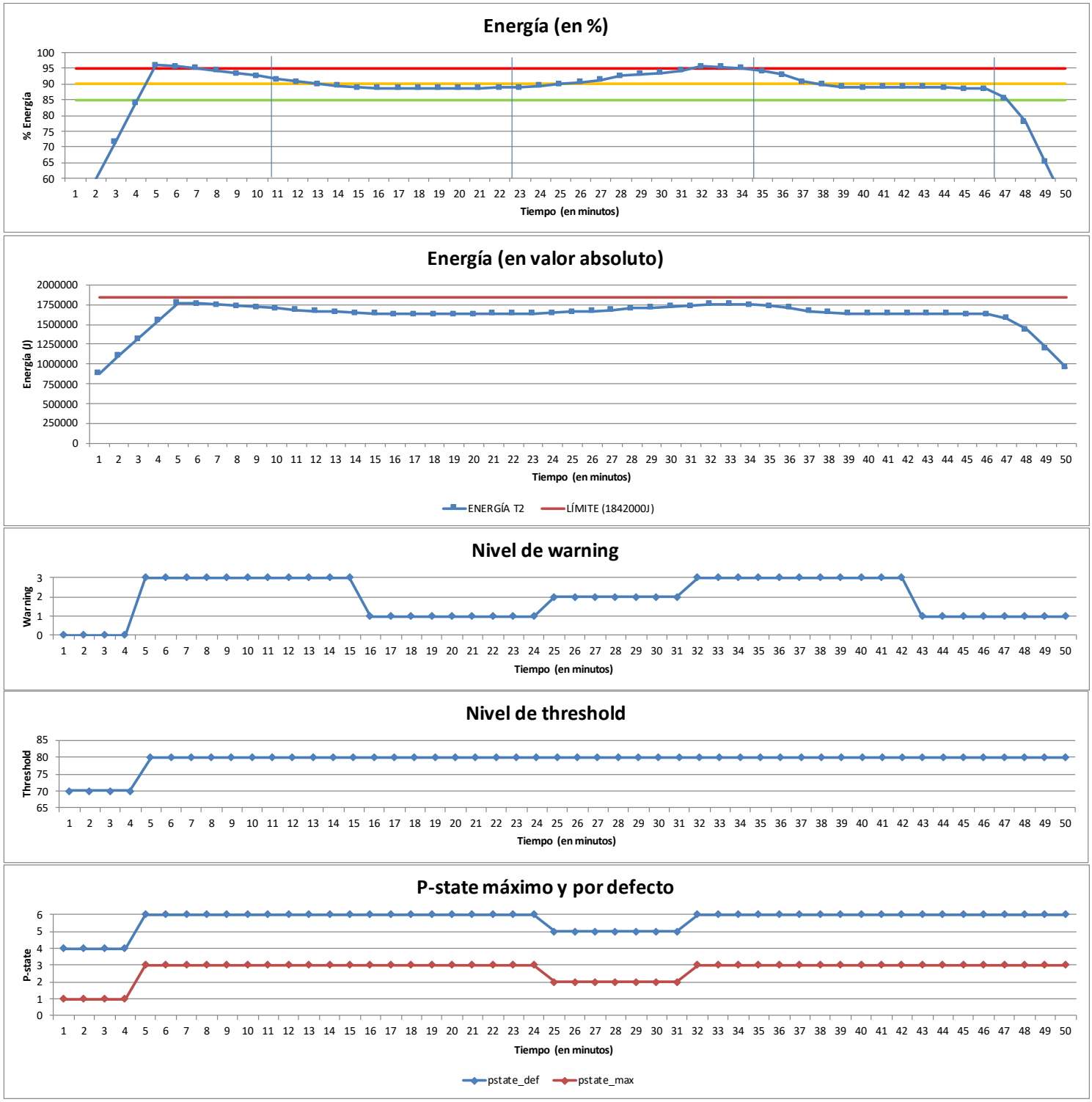


Figura 12.12: Experimento 20.6 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

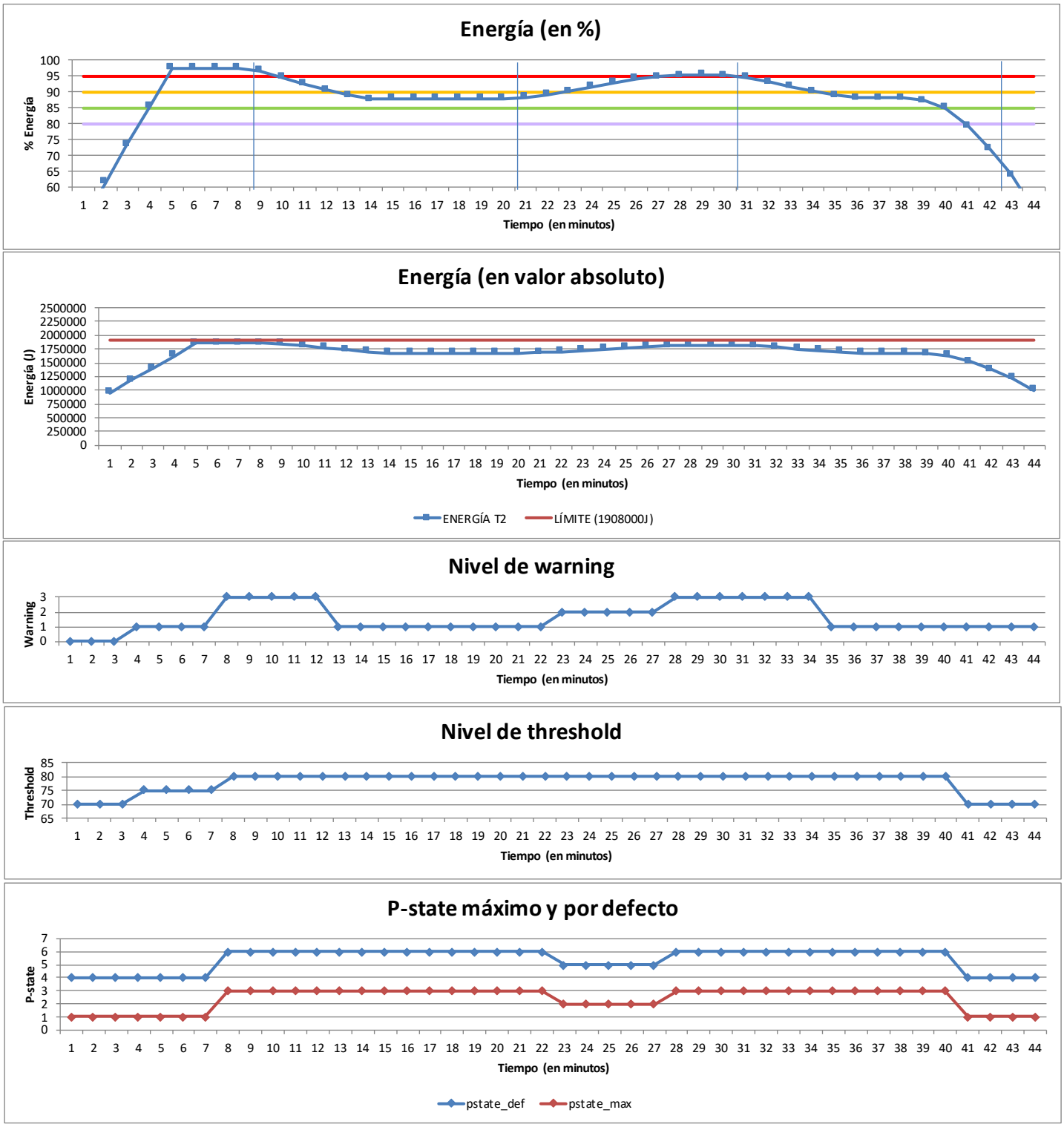


Figura 12.13: Experimento 20.6 con optimización ramp down ajustado al nivel PANIC.

12.4.4. Experimento 20.7: Mix dinámico con pérdida de carga

Por último y no menos importante, se presenta este experimento el cual es un *workload* formado por un *kernel* intensivo en cálculo que va perdiendo la mitad de la carga dinámicamente y la va recuperando. En la figura 12.14 se muestra la ejecución de la carga de trabajo ajustada a NO PROBLEM, con el fin de observar su comportamiento natural y examinar el impacto de los ajustes sobre ella.

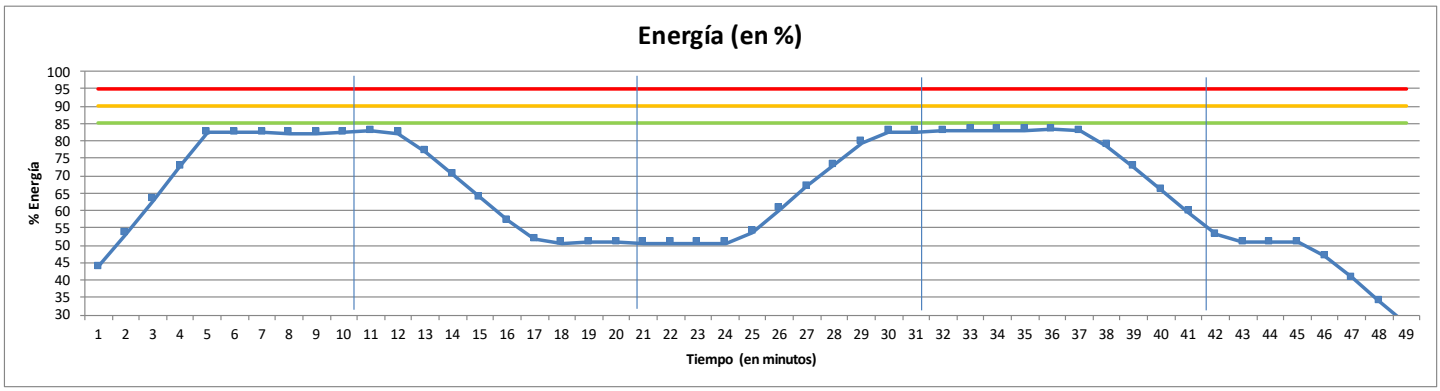


Figura 12.14: Comportamiento del experimento 20.7 sin acciones del EARGM.

Las siguientes figuras 12.15 y 12.16 muestran, respectivamente, para la optimización ramp up los niveles de criticidad WARNING2 y PANIC. También, para este experimento se han agregado los resultados obtenidos con el nivel de WARNING1 en el apéndice E.7. El aumento del threshold no tiene ningún efecto sobre este experimento, por lo que la variación entre el nivel de WARNING1 y NO PROBLEM viene determinada por la alternancia entre una carga totalmente intensiva en CPU y la mitad del sistema vacío.

Por otro lado, al comenzar con una carga completamente de cálculo, se observa que el consumo energético desciende a unos niveles que concuerdan con los experimentos intensivos en CPU. No obstante, para el caso concreto de WARNING2 (figura 12.15), aproximadamente en el minuto 11 se pierde la mitad de la carga del sistema, por lo que desciende a niveles muy bajos. Durante el tiempo que se permanece con el *workload* funcionando a la mitad, se va restaurando gradualmente la configuración inicial, de modo que al volver a operar toda la máquina vuelve a consumir como lo hizo inicialmente, aplicando de nuevo los debidos ajustes. Por otro lado, para el caso concreto de PANIC tiene el mismo comportamiento salvo que, al reducirse 2 p-states, la pendiente es más pronunciada.

Por último, se ha analizado la optimización de ramp down en este experimento para los niveles de WARNING2 y PANIC, cuyos resultados se encuentran en las figuras 12.17 y 12.18 respectivamente. En ambas se observa que al traspasar el límite inferior se restaura la configuración inicial, aunque pocos minutos después pierde la mitad de la carga. Es importante tener presente que la pérdida de la carga, al tener un impacto muy intenso, ocasiona una rápida variabilidad en las gráficas.

Optimización ramp up:

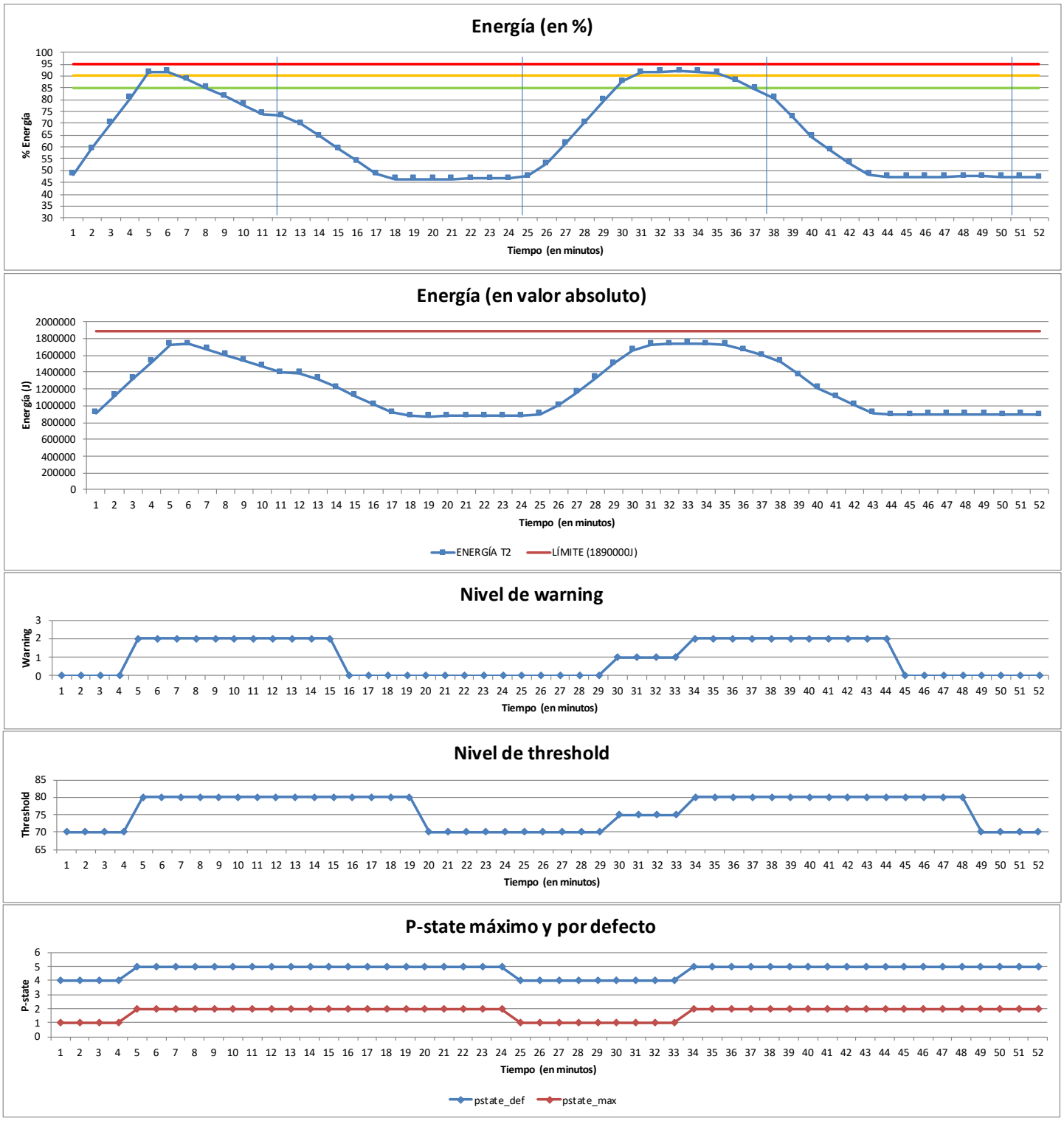


Figura 12.15: Experimento 20.7 con optimización ramp up ajustado al nivel WARNING2.

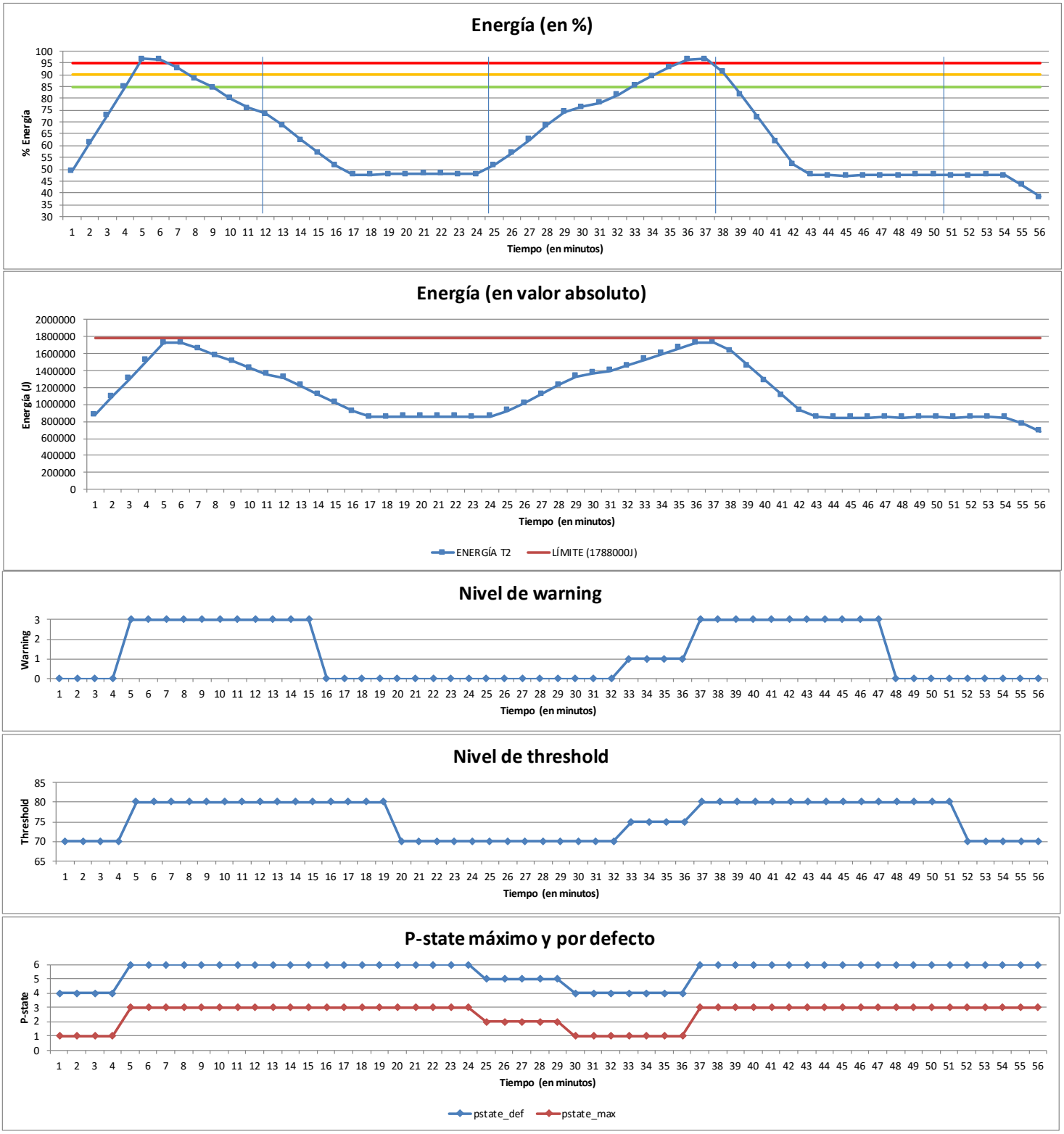


Figura 12.16: Experimento 20.7 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

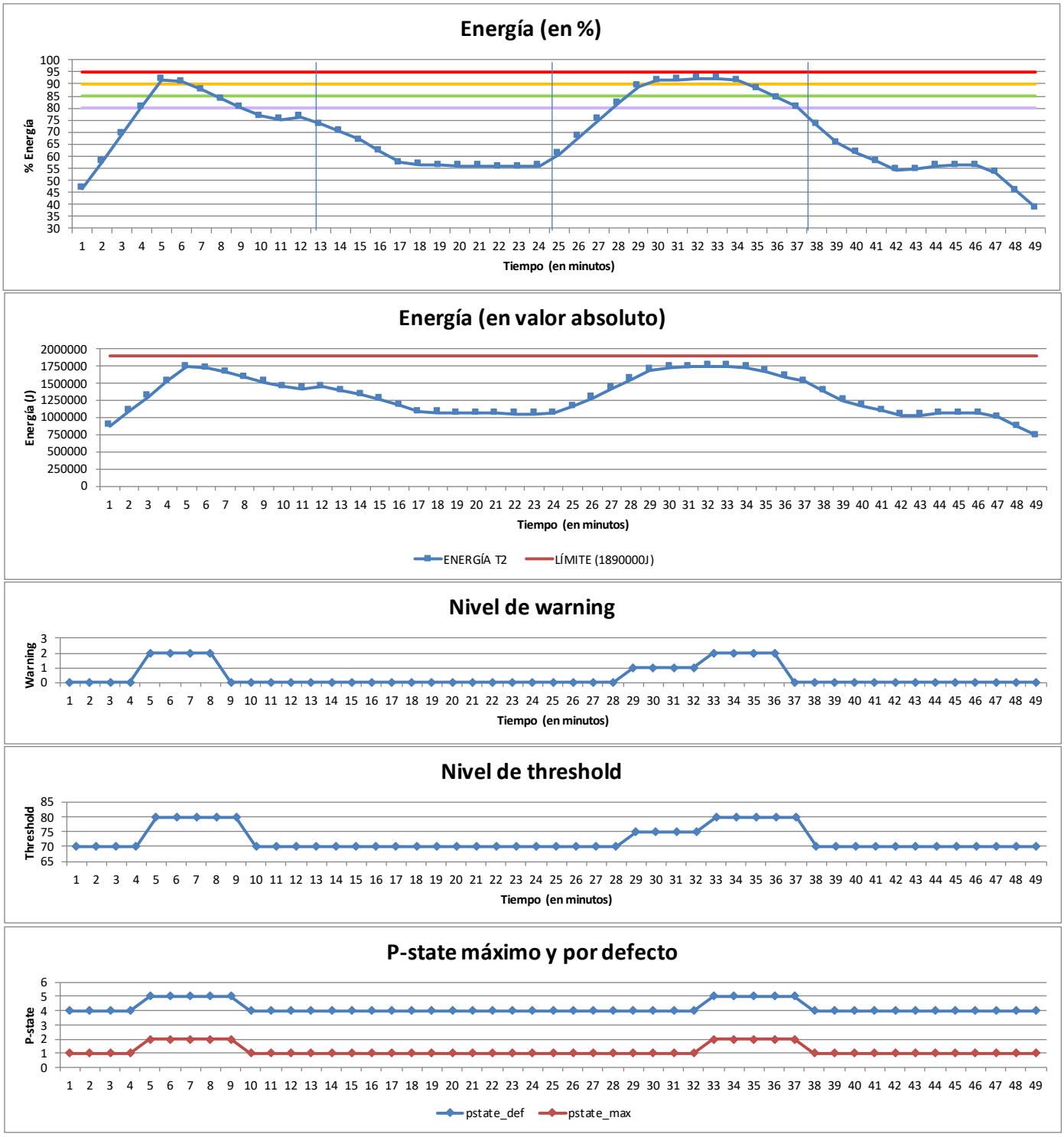


Figura 12.17: Experimento 20.7 con optimización ramp down ajustado al nivel WARNING2.

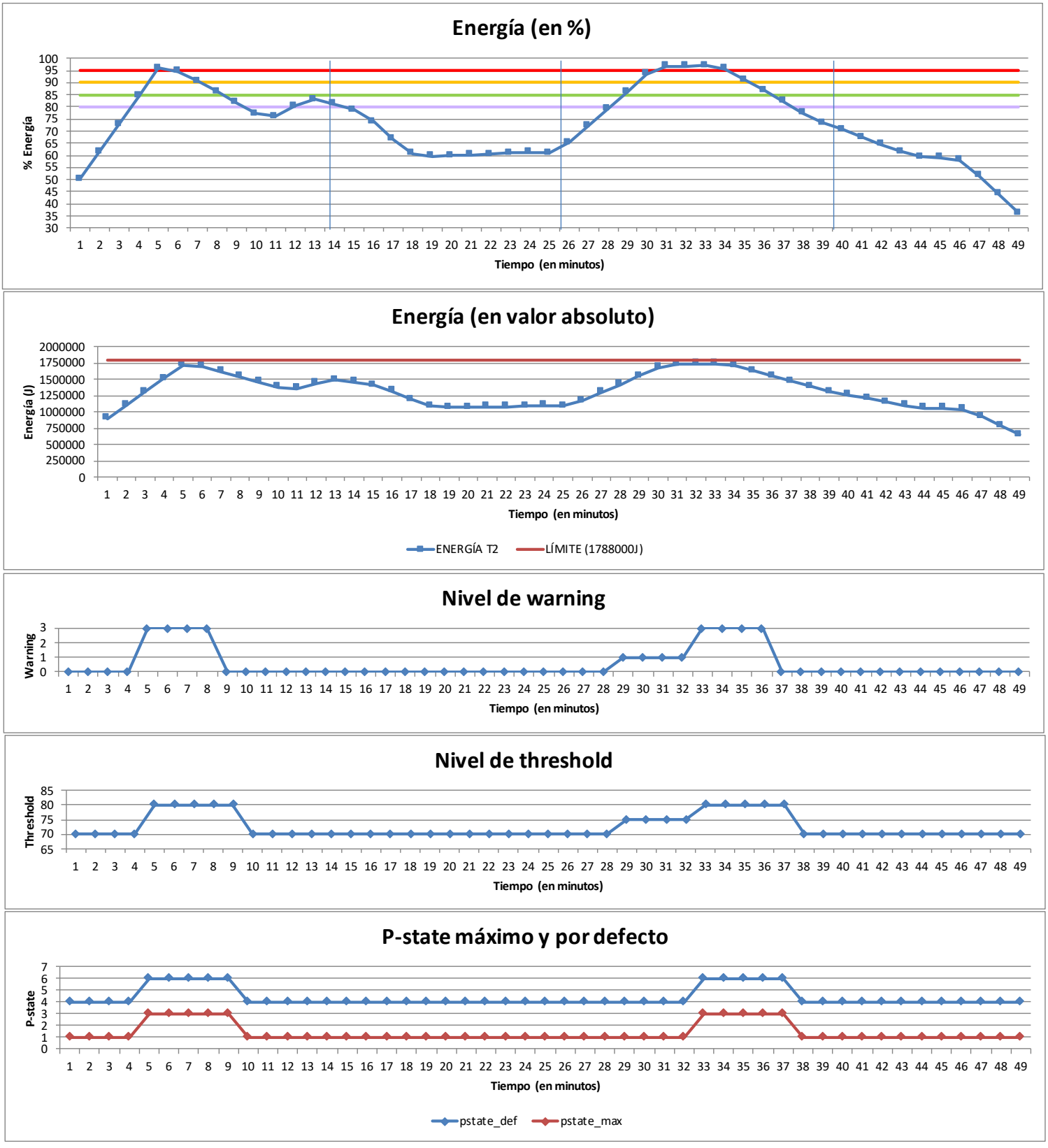


Figura 12.18: Experimento 20.7 con optimización ramp down ajustado al nivel PANIC.

La optimización de ramp down puede aparentar que al fin y al cabo no haya mucha diferencia respecto a únicamente ramp up, pero realmente se está mejorando el rendimiento, aprovechando mejor los recursos el máximo de tiempo posible, ya que al perder la carga no se puede hacer más.

Finalmente, usando como referencia el límite ajustado para PANIC obtenemos para el nivel WARNING2 con la optimización ramp up un consumo energético medio del 69 %, mientras que con ramp down es del 73 %. Para el nivel de PANIC se obtiene un consumo energético medio del 64,7 % con ramp up, mientras que con ramp down se consigue un 73,2 %.

12.5. Conclusiones

A lo largo de este capítulo se han expuesto una gran variedad de situaciones posibles en un sistema por medio de cada *workload*, y se han evaluado las decisiones del componente Global Manager y el impacto generado como consecuencia.

Hemos verificado que aquellos *workloads* con un alto porcentaje de intensidad en CPU son los que reciben un impacto mayor al aplicar los ajustes ordenados por el EARGM, a diferencia de los experimentos más intensificados en el uso de la memoria en que apenas se han percibido cambios. Sin embargo, hay que tener en consideración que no sería habitual encontrar un sistema con un *workload* intensivo en memoria que sufriera problemas energéticos.

Apoyándonos en los resultados obtenidos a lo largo del TFG, se ha evidenciado que las cargas de trabajo intensivas en memoria operan a frecuencias bajas (cercanas a la frecuencia por defecto), ya que al ser poco eficientes, la política `min_time` no les recompensa incrementando la frecuencia hacia frecuencias elevadas cercanas a la nominal. Por este motivo, al estar trabajando por debajo de la frecuencia nominal del sistema sería absurdo tener problemas, ya que indicaría que se tiene un límite energético demasiado bajo para las características de esa máquina. Por lo tanto, al utilizar la librería de EAR los *workloads* con un alto porcentaje de aplicaciones intensivas en memoria será difícil que tengan problemas de configuración.

Por otro lado, para *workloads* intensificados en el uso de la CPU sí será más usual que ocurran problemas de configuración.

Hemos visto y comprobado que el EARGM, con el uso conjunto de las optimizaciones ramp up y ramp down, controla los límites energéticos y logra una recuperación de la configuración más rápida para aquellos casos que presentan una variación considerable, con tal de no obtener una pérdida de rendimiento significativa.

Para concluir, se ha detectado que es necesario realizar una modificación de la configuración a un nivel más fino en que no se ordenen acciones a todos los nodos del *cluster* para evitar pérdida de rendimiento.

Capítulo 13

Informe de sostenibilidad

En un proyecto informático influyen tres dimensiones muy importantes: la ambiental, la social y la económica. Sin embargo, suelo tener más presente las dimensiones social y ambiental, dejando de lado la económica. No obstante, comento que la realización de la gestión económica para el proyecto ha contribuido a tomar más consciencia del impacto económico que puede llegar a suponer una determinada actividad.

En cuanto a la dimensión ambiental, soy plenamente consciente del impacto en la huella ecológica que puede producir tanto el hardware en residuos electrónicos como el software en gasto energético. Previamente he participado en el reciclaje de componentes hardware que ya no eran de mi necesidad pero sí se les podía dar una segunda vida. Sin embargo, quiero poner énfasis en que aún existen barreras para reciclar determinados componentes.

Por otro lado encontramos la dimensión social, que quizás ha sido la más estudiada durante el grado, ya que se obtienen conocimientos para poder identificar problemas existentes en la sociedad y tratar de resolverlos (finalidad principal de un ingeniero). Sin embargo, considero que no cuento con las suficientes herramientas para cuantificar el impacto social que supone.

En cuanto a la dimensión económica, este proyecto me ha servido para poner en práctica los conocimientos sobre la planificación y análisis económico, dando como resultado el presupuesto presentado en el capitulo 6.

Para concluir, observo que hay consciencia sobre el impacto que puede suponer un proyecto en todas las dimensiones. No obstante, falta formación en herramientas para poder analizar y medir los impactos, y así luego poder proporcionar soluciones más sostenibles.

13.1. Dimensión ambiental

El impacto ambiental que ha supuesto este proyecto ha sido debido al consumo eléctrico tanto por parte del portátil y la pantalla como por el *cluster* Lenox, donde este último consume infinitamente más que los otros dos. Sin embargo es muy difícil poder cuantificar el consumo eléctrico total ya que se desconoce el consumo del portátil, aunque sí es conocido el consumo energético en el *cluster* por mi parte (284400286745J) gracias al comando `ereport` que proporciona EAR. Si ahora hiciera de nuevo el proyecto, podría realizarlo logrando un consumo eléctrico menor ya que actualmente cuento con una experiencia que previamente no tenía, por lo que se podrían evitar suficientes horas de ejecución en virtud de pruebas que se tuvieron que repetir. Asimismo, también se reduciría el coste en la dimensión económica.

Durante la vida útil del proyecto se utilizarán *clusters*, siendo inevitable su uso, ya que este proyecto forma parte de un software específico para centros HPC, lugar donde se encuentran este tipo de máquinas. No obstante, como se ha comentado anteriormente, es difícil de precisar el impacto ambiental para este tipo de máquinas, pero vendrá determinado por el consumo eléctrico. Dado que EAR ofrece entre un 9 % y un 10 % de ahorro energético en media, este trabajo contribuirá a una supercomputación más eficiente, que a la larga se traducirá en reducción de consumo energético y en una mejora en la huella ecológica (siendo más ecológico y económico). También, este proyecto está libre de riesgos que puedan producir un aumento de la huella ecológica.

13.2. Dimensión económica

En el capítulo 6 se ha cuantificado detalladamente el coste económico que ha supuesto la realización de este proyecto, el cual se ha ajustado al coste previsto gracias a que se ha cumplido la planificación. Para reducir el coste se ha utilizado el *cluster* Lenox, entorno que ha hecho posible que mi versión pudiera coexistir con la versión oficial sin tener que reinstalar una versión exclusiva para mí. De otro modo hubiese sido mucho más costoso ya que se tendría que haber paralizado el funcionamiento de un Global Manager operativo para que yo pudiera realizar el proyecto, aunque tampoco es cuantificable.

Por otro lado, en la cuantía económica no se ha considerado el coste de posibles ajustes o actualizaciones durante la vida útil del proyecto. No obstante, si se produjera un cambio de arquitectura no afectaría ya que trabaja a un nivel más alto y posibles cambios en la configuración tendrían un coste 0. En caso de cambios en EAR, el coste sería el de recompilar el software. Por último, si se realizaran extensiones tendría el coste de realizar dicha extensión, el cual sería el único aspecto que podría influir en el impacto económico.

Por último, este proyecto está libre de riesgos que puedan perjudicar monetariamente ya que forma parte de un software de libre distribución cuyo objetivo es contribuir a una supercomputación más eficiente, no busca lucrarse económicamente.

13.3. Dimensión social

Este TFG me ha dado la oportunidad de poder adentrarme en el mundo de la investigación en el campo de la computación de altas prestaciones, ya que no tenía experiencia previa y siento un profundo interés en este área. Por otro lado, también me ha permitido coger experiencia en trabajar con más personas en un proyecto grande.

Este trabajo ha solucionado completamente el problema planteado inicialmente y beneficiará al personal relacionado con el campo de HPC. En concreto a investigadores y administradores de sistemas, ya que tendrán al alcance una herramienta que les ayude a reducir su huella ecológica, que junto con la librería EARL se logrará más eficiencia energética, lo que a largo plazo podrá suponer un ahorro en electricidad. Beneficia tanto en términos económicos como ecológicos. Además, ningún colectivo debería verse perjudicado, más bien al contrario, ya que se conseguirá ser más respetuoso con el medio ambiente gracias a un consumo más responsable.

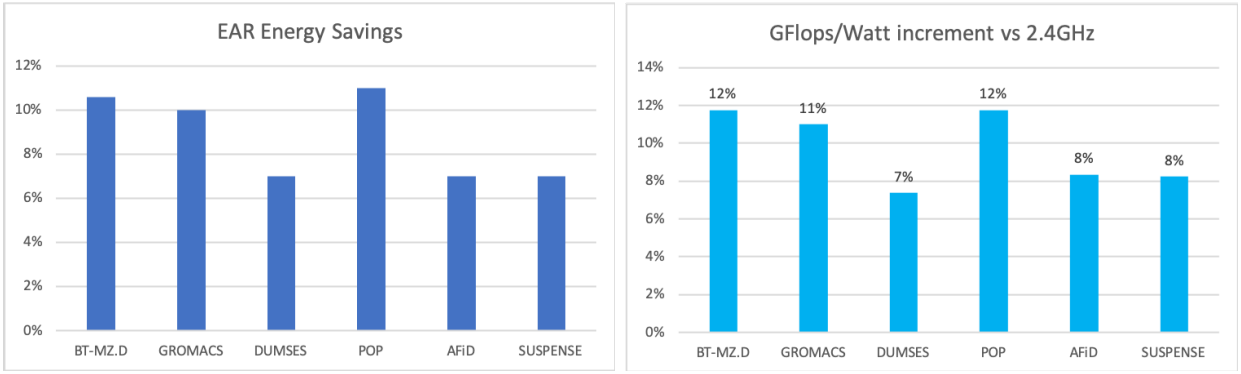
Finalmente, el proyecto está libre de riesgos ya que en ningún caso puede perjudicar a ningún segmento de la población ni debilitar la posición de algún usuario. El trabajo se trata de una herramienta creada para facilitar la faena a los administradores de sistemas para la gestión de los límites energéticos en un *data center*. Si por algún motivo no la quisieran utilizar, tienen la libertad de activar el modo manual y aplicar manualmente los ajustes.

13.4. Contribución de EAR

Este trabajo va puramente relacionado con la energía, por lo que es importante dedicarle una sección a analizar su contribución a la sostenibilidad. El componente Global Manager es un servicio que forma parte de una infraestructura que está totalmente vinculada a la sostenibilidad (software EAR).

A diferencia de otros componentes de EAR, el EARGM no proporciona eficiencia energética pero sí se encarga de controlar y aplicar un conjunto de ajustes con tal de garantizar que se cumplan unos requerimientos, en términos energéticos, especificados por el administrador de sistemas. Así pues, vemos que el Global Manager desempeña un papel fundamental para el software en general ya que es necesario un gestor energético.

Por otro lado, el indicador que nos sirve para medir el impacto en la sostenibilidad es a través de la unidad que mide la energía, los *joules* (J). Seguidamente se presentan unas gráficas extraídas de un *paper* publicado recientemente por el grupo de investigación de EAR [18], en que se muestra un conjunto de aplicaciones analizadas con la política `min_time` y se compara respecto a no usarla:

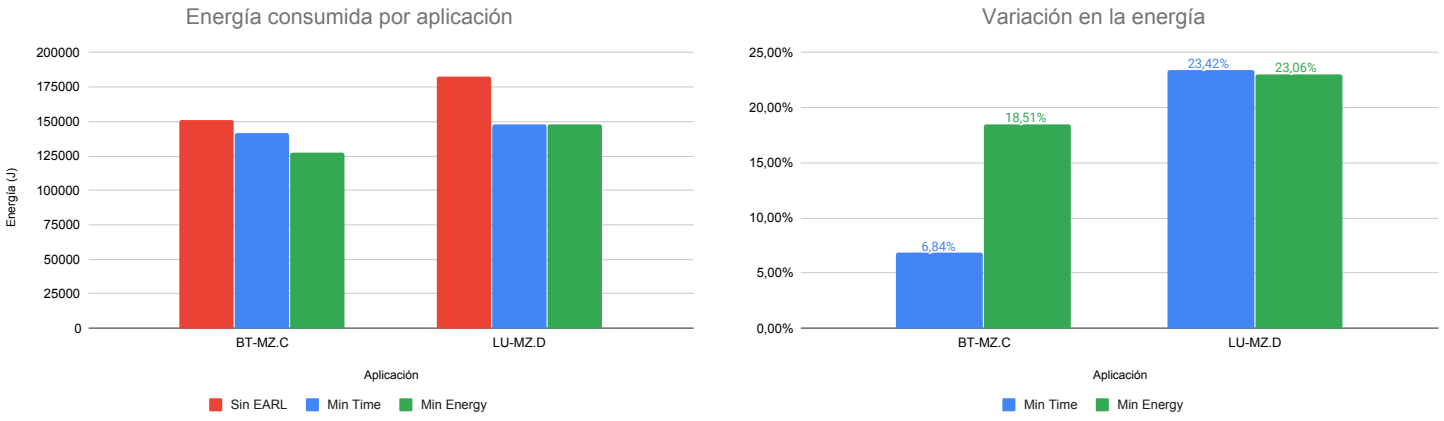


(a) Ahorro energético.

(b) Incremento en la eficiencia energética.

Figura 13.1: Ahorro energético e incremento en la eficiencia energética para un conjunto de aplicaciones utilizando `min_time`. [18]

Observamos que la ratio de eficiencia energética (GFlops/Watt) mejora notoriamente. Por ejemplo, vemos que para la aplicación POP se ha incrementado en un 12 %. A continuación, muestro como resultados de analizar una aplicación de cálculo (BT-MZ.C) y una aplicación de memoria (LU-MZ.D), los ejemplos por excelencia en este trabajo. Cada una se ha ejecutado con 4 nodos, los mismos para cada experimento y aplicación, para comprobar la variación energética: sin utilizar EARL, utilizando EARL con la política `min_time` y EARL con la política `min_energy`. La frecuencia máxima es 2,4 GHz, para `min_time` la frecuencia por defecto configurada es 2,1 GHz y un threshold del 70 %, y para `min_energy` se ha configurado el threshold al 10 %.



(a) Energía consumida en total.

(b) Variación energética.

Figura 13.2: Energía consumida y variación energética para BT-MZ.C y LU-MZ.D.

Se ha incluido la política `min_energy` ya que es otra opción en EAR a pesar de no haberse utilizado en este proyecto, y ésta trata de encontrar la frecuencia que genera la energía mínima con una pérdida de rendimiento máxima. Por las gráficas de la figura 13.2, los resultados hablan por sí solos en que se obtiene un buen ahorro energético con ambas políticas respecto a la alternativa sin gestión energética. Por ejemplo, para la aplicación LU-MZ.D la diferencia es importante (23 % con ambas políticas).

Capítulo 14

Conclusiones

La importancia de la gestión energética ha quedado plenamente reflejada en este trabajo, ya que los centros HPC se han convertido en unos grandes consumidores de energía y esto les ocasiona un reto tanto a nivel ecológico como económico. No obstante, se ha comprobado que esta gestión no es sencilla puesto que no depende únicamente de la arquitectura con la que se trabaja o de la aplicación, siendo imprescindible la ayuda de un software.

El corazón de este proyecto ha sido el Global Manager, integrante en el software de gestión energética EAR. Este componente tiene la finalidad de garantizar que el consumo energético en un sistema para un periodo de tiempo está dentro de unos límites especificados por el administrador de sistemas. Inicialmente se encontraba una implementación muy básica y en que, como se ha comprobado, las decisiones que tomaba no eran correctas.

El objetivo de este trabajo era implementar un Global Manager capaz de tomar decisiones razonables según el estado del sistema en el que se encuentra y penalizando al mínimo el rendimiento. Para ello se han arreglado las principales deficiencias que se presentaban, incorporando también mejoras en funcionalidades para perfeccionar la robustez del código. Además, también se han implementado optimizaciones para acelerar las decisiones y obtener una recuperación más pronta de la configuración inicial para lograr mejor rendimiento.

Para la evaluación se han utilizado aplicaciones intensivas en CPU y en memoria, ya que tienen una respuesta muy diferente al variar la frecuencia del procesador. En cuanto a las aplicaciones intensivas en memoria se ha comprobado que poco se puede hacer para reducir significativamente el consumo de la potencia, ya que están más afectadas por la memoria que por la CPU, pero al menos ya no se les castiga continuamente aplicando ajustes como ocurría en el estado inicial. Sin embargo, también es importante tener presente que este tipo de *workload* que trabaja a frecuencias bajas (con la frecuencia por defecto) no es habitual que padezca problemas de energía puesto que ya se está ejecutando a frecuencias por debajo de la frecuencia nominal de la máquina. Por otro lado, en cuanto a los *workloads* con carácter intensivo en CPU sí es más común y ha quedado verificado que las optimizaciones les ayudan notoriamente a mejorar el rendimiento, aprovechando más los recursos que

ofrece el sistema. De hecho, la ejecución del experimento correspondiente a la figura 9.6, al ajustarse al límite PANIC correspondiente, obtuvo un consumo energético medio de un 74,13 %, alcanzando un máximo del 95,76 %. Sin embargo, para el mismo *workload* con el mismo nivel de *warning* se ha logrado aumentar a un consumo medio del 87 %, alcanzando un máximo del 96 %. En resumen, vemos que se ha aumentado el consumo energético en un 13 %, indicio de que se están aprovechando más los recursos del sistema.

A nivel personal este proyecto me ha permitido conocer el mundo de la supercomputación, del cual no contaba con experiencia previa debido a que durante la carrera no se adquieren conocimientos de esta especialidad. De hecho, quiero remarcar que he aprendido muchísimo ya que, para comenzar, todos los conceptos eran nuevos y considero que he desarrollado más ampliamente mi capacidad analítica, a la par que he formado parte de un proyecto muy grande y de gran interés para mí.

Para acabar, a raíz de la experiencia vivida por los problemas surgidos por el fallo de alimentación en el *cluster* he aprendido una lección. Esta consiste en ejecutar uno tras otro todos los experimentos pertenecientes a la misma carga de trabajo, y así evitar problemas como que los límites de potencia se vean alterados. Sin embargo, inicialmente era mi idea pero dado que competía constantemente con otros investigadores por los recursos del sistema no me fue posible. En definitiva, he aprendido que de cara a las siguientes evaluaciones que realice, concertaré con el administrador de sistemas una reserva de unos determinados nodos para ejecutar los experimentos sin interferencias.

Capítulo 15

Trabajo futuro

A lo largo de este trabajo hemos podido comprobar que los resultados obtenidos con los debidos arreglos y optimizaciones han sido muy favorables, logrando un Global Manager mucho más robusto. No obstante, gracias a la profunda evaluación que se ha realizado, detectamos que hay casos en que se podrían suavizar las curvas, tanto para el proceso de incremento de energía como para su descenso, y ahí fue cuando abrimos la caja de Pandora.

Para conseguir aplacar aún más los incrementos y decrementos energéticos es obvio que ordenando las acciones para todos los nodos del sistema no es posible, por lo que se deberá acotar un rango de nodos considerando las características de los *workloads*. Por ello nos preguntamos cómo escoger quiénes reciben las órdenes con las acciones, y nos salen dos alternativas:

1. **Aplicar las acciones a nivel de *job*:** El Global Manager recibiría información sobre los nodos pertenecientes a un mismo *job*. En función de los resultados estimaría el objetivo energético a conseguir y ordenaría acciones a aquellos nodos, pertenecientes a un mismo *job*, que pudieran cumplir con la predicción.

2. **Aplicar las acciones a nivel de nodo:** El Global Manager contaría con información individual por nodo que describiría las características de la carga que están ejecutando, así como el CPI y el ancho de banda. Según las necesidades del sistema ordenará unas acciones a uno o más nodos.

A simple vista parece tener más sentido aplicar las acciones a nivel de nodo antes que a nivel de *job*. Sin embargo, nos encontramos ante diferentes estrategias que se podrían seguir:

- **Minimizar el número de nodos afectados:** Se aplicarían los ajustes a aquellos nodos que trabajan a frecuencias más elevadas, ya que eso es un indicio de que están usando más recursos que otros. Estos nodos, al trabajar a las frecuencias más altas, tendrían un gran impacto y por eso el número de nodos sería reducido. No obstante,

esto implica empeorar el rendimiento a unos nodos que ejecutan una carga de trabajo muy eficiente.

- **Penalizar los nodos con un CPI mayor:** En este caso se aplicarían los ajustes a aquellos nodos que cuentan con un CPI elevado, es decir, trabajan a frecuencias bajas. Estos nodos no tienen *workloads* tan eficientes, por eso trabajan a frecuencias inferiores, por lo que su impacto será menor que en el caso anterior. Sin embargo, al tener un impacto pequeño significa que hay que aplicar acciones a una cantidad mayor de nodos.

Con esto vemos que sería necesaria una evaluación para decidir qué alternativa sería la más apropiada y resolutiva. No obstante, para ello sería necesario otro estudio valorativo que equivaldría a otro trabajo de fin de grado o máster, saliéndose de los límites de mi TFG. Por ello se expone la continuación de esta investigación como trabajo futuro.

Para acabar, también se abre como futuro trabajo la implementación de un mecanismo de coordinación con SLURM para casos anómalos y poco frecuentes con restricciones muy altas. Estas situaciones implicaría que el sistema, por algún motivo, está configurado con un límite bastante bajo, por lo que la única opción posible sería el apagado de algunos nodos, que podría procederse cuando se detectaran situaciones críticas. El gestor de colas SLURM [28] ofrece el estado DRAIN, el cual permite dejar inoperativo un nodo que, en caso de haber un *job* ejecutándose, se espera a que termine para dejar de estar disponible. Se ejecutaría con el comando `scontrol` indicando el o los nodos a apagar, el estado a DRAIN y un motivo, por ejemplo:

```
scontrol update NodeName=node[42-44] State=DRAIN Reason="Limit"
```

Por otro lado, para volver a encender los nodos se ejecutaría el comando `scontrol` cambiando ahora el estado a RESUME, por ejemplo:

```
scontrol update NodeName=node[42-44] State=RESUME
```

En resumen, consistiría en diseñar los comandos que se ejecutarían automáticamente en los diferentes niveles de riesgo para modificar la configuración del *scheduler* en función del consumo energético.

A Glosario

Application signature En terminología de EAR, métricas de rendimiento que permiten conocer el comportamiento de una determinada aplicación (CPI, ancho de banda, etc.) y es uno de los *input* en los modelos energéticos.

Cluster Conjunto de nodos interconectados por una red de alta velocidad.

Frecuencia máxima o nominal Frecuencia máxima establecida por un fabricante a la que puede funcionar una CPU (sin incluir las frecuencias pertenecientes al modo turbo).

Frecuencia por defecto En terminología de EAR, frecuencia con la cual se inicia la ejecución de una aplicación, es inferior a la frecuencia máxima si se utiliza la política `min_time` y es igual a la frecuencia máxima si se utiliza la política `min_energy`.

Governor Política de gestión de la potencia de una CPU por medio de p-states.

Job *Script* que se ejecuta en los nodos de un *cluster*, solicitando recursos e indicando los comandos que se quieren ejecutar.

Nodo Elemento de un *cluster*. En este trabajo nos referimos únicamente a los dedicados al cálculo, los cuales están formados por 1 o más CPUs, memoria y aceleradores.

P-state Índice en un vector de frecuencias que tiene asociado una frecuencia y voltaje determinados.

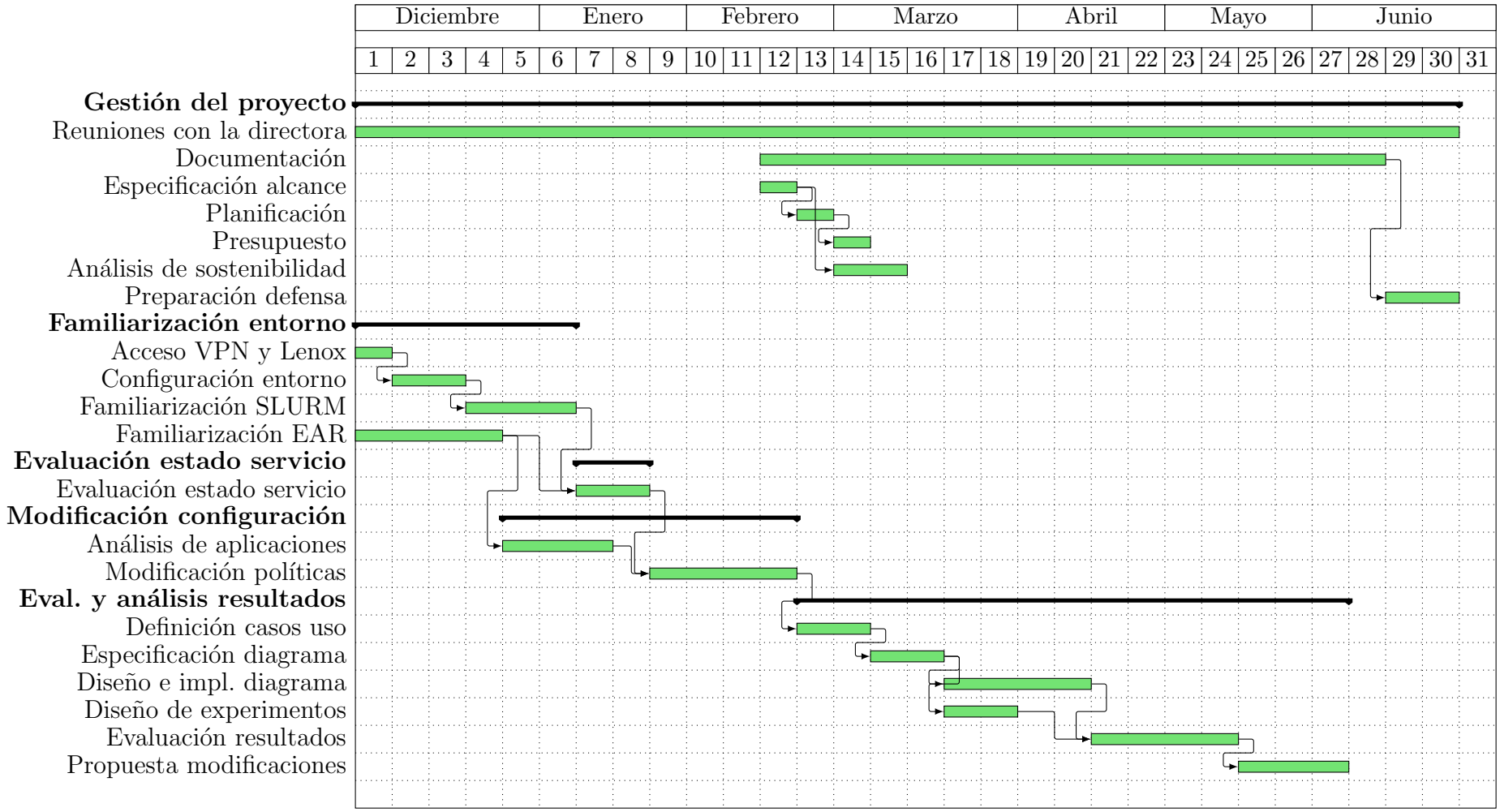
Periodo T1 En terminología EAR, periodo de tiempo que indica cada cuanto tiempo se quiere actualizar la información en el Global Manager.

Periodo T2 En terminología EAR, periodo de tiempo en el que se evalúa el cumplimiento de los límites energéticos globales establecidos.

System signature En terminología de EAR, métricas de rendimiento obtenidas de los principales componentes de cada nodo perteneciente al sistema y es uno de los *input* en los modelos energéticos.

Threshold En terminología de EAR, delimita un umbral en las políticas energéticas.

B Diagrama de Gantt



C Ejemplos de scripts

C.1. Kernel intensivo en CPU

```
#!/bin/bash
#SBATCH --job-name=bt
#SBATCH --output=OUTS/bt.%J.out
#SBATCH --error=ERR/bt.%J.err
#SBATCH --nodes=20
#SBATCH --ntasks=160
#SBATCH --partition standard
#SBATCH --time=3:00:00
#SBATCH --exclusive
#SBATCH --constraint="6148&OPA"
#SBATCH --account=COLBSC
#SBATCH --nodelist=cmp[2752,2754-2772]

module load mpi/intel/2018.4
module load compiler/intel/18.5

export I_MPI_PIN=1
export I_MPI_PIN_DOMAIN=auto

export CTT=/hpc/base/ctt
source $CTT/bin/setup_modules.sh

export SLURM_EAR_TRACE_PLUGIN=$EAR_INSTALL_PATH/lib/plugins/tracer/tracer_paraver.so
export SLURM_EAR_TRACE_PATH=RAMPDOWN/W2/

export KERNELS_PATH=/home/xatomas/benchmarks/BT-MZ/bin
export kernel=bt-mz.C.x.250k
export MPIS=160
export OMP_NUM_THREADS=5

export I_MPI_HYDRA_BOOTSTRAP=slurm
export I_MPI_HYDRA_BOOTSTRAP_EXEC_EXTRA_ARGS="--ear-policy=min_time --ear-verbose=1"
mpirun -np $MPIS -l $KERNELS_PATH/$kernel
```

C.2. Parte variable en los mix dinámicos

```
#!/bin/bash
#SBATCH --job-name=LU-D_BT
#SBATCH --output=OUTS/btlud.%J.out
#SBATCH --error=ERR/btlud.%J.err
#SBATCH --nodes=2
#SBATCH --ntasks=16
#SBATCH --partition standard
#SBATCH --time=03:00:00
#SBATCH --exclusive
#SBATCH --constraint="6148&OPA"
#SBATCH --account=COLBSC
#SBATCH --nodelist=cmp27[57-58]

module load mpi/intel/2018.4
module load compiler/intel/18.5

export I_MPI_PIN=1
export I_MPI_PIN_DOMAIN=auto

export CTT=/hpc/base/ctt
source $CTT/bin/setup_modules.sh

export KERNELS_PATH_BT=/home/xatomas/benchmarks/BT-MZ/bin
export kernel_bt=bt-mz.C.x.11500
export KERNELS_PATH_LU=/home/xatomas/benchmarks/LU-MZ/bin
export kernel_lu=lu-mz.D.x.525

export I_MPI_HYDRA_BOOTSTRAP=slurm
export I_MPI_HYDRA_BOOTSTRAP_EXEC_EXTRA_ARGS="--ear-policy=min_time --ear-verbose=1"

for i in 1 2
do
export MPIS=16
export OMP_NUM_THREADS=5
current_time=`date "+%Y-%m-%d %H:%M:%S"`
echo "Time for LU:$current_time"
mpirun -np $MPIS -l $KERNELS_PATH_LU/$kernel_lu
export MPIS=80
export OMP_NUM_THREADS=1
current_time=`date "+%Y-%m-%d %H:%M:%S"`
echo "Time for BT:$current_time"
mpirun -np $MPIS -l $KERNELS_PATH_BT/$kernel_bt
done
```

D Experimentos de validación

En este apéndice se muestran los experimentos que forman parte de la etapa de validación. Por cada una de las pruebas se muestran los resultados de ajustar el límite energético para cada nivel de *warning* con la optimización ramp up. En caso pertinente, también se presentan para los niveles más críticos con la optimización ramp down.

D.1. Exp. 4.1: Kernel intensivo en CPU

El comportamiento por defecto del experimento se muestra en la figura D.1.

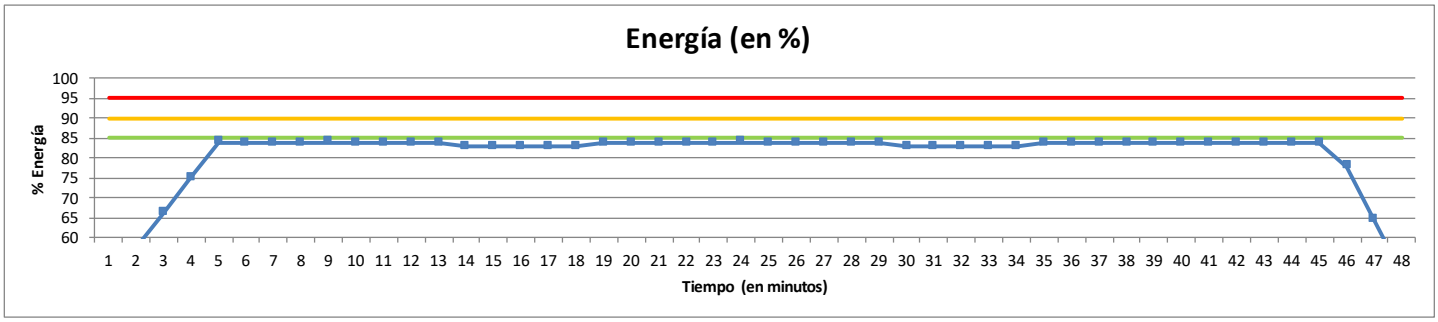


Figura D.1: Comportamiento del experimento 4.1 sin acciones del EARGM.

Optimización ramp up:

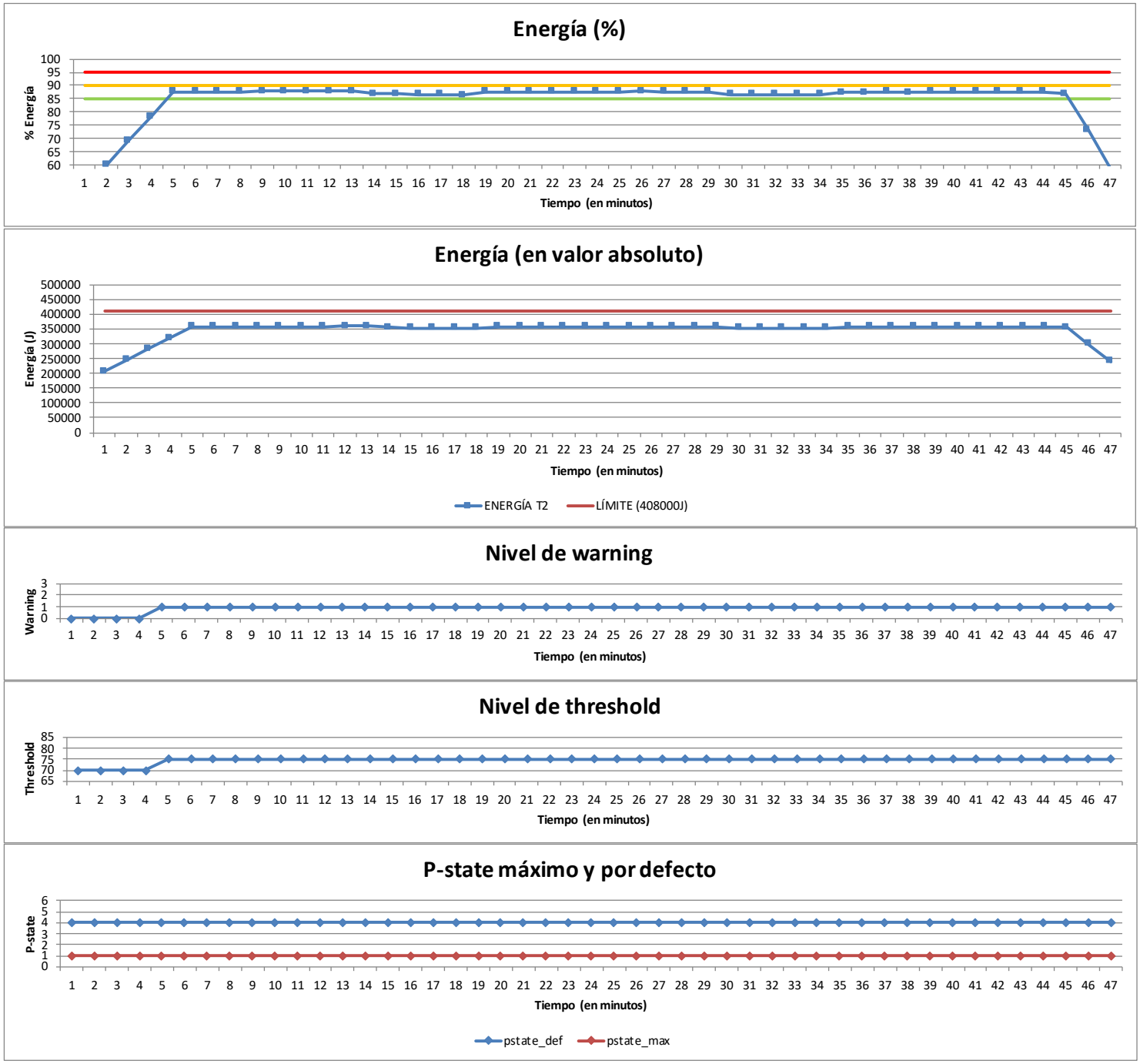


Figura D.2: Experimento 4.1 con optimización ramp up ajustado al nivel WARNING1.

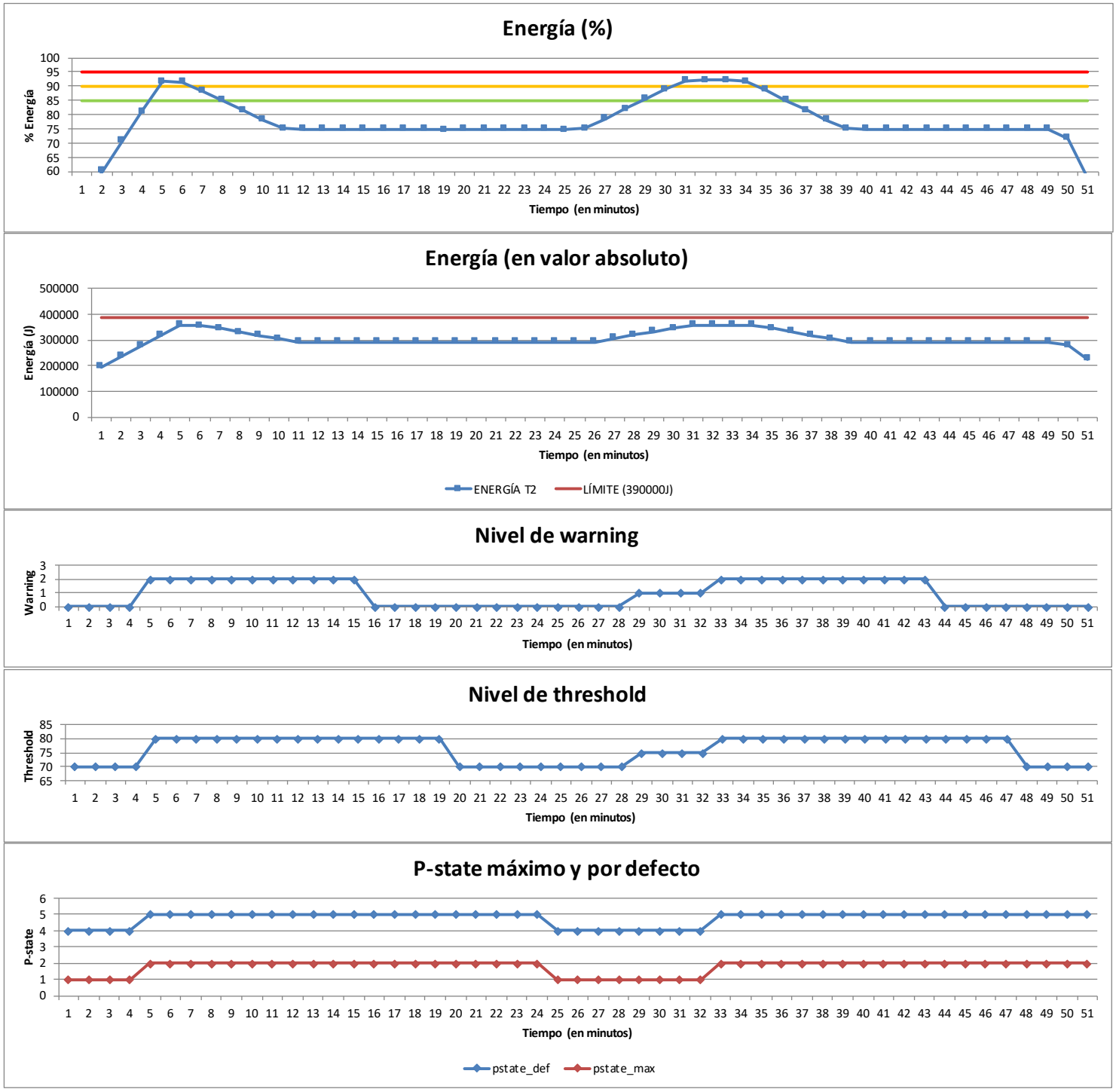


Figura D.3: Experimento 4.1 con optimización ramp up ajustado al nivel WARNING2.

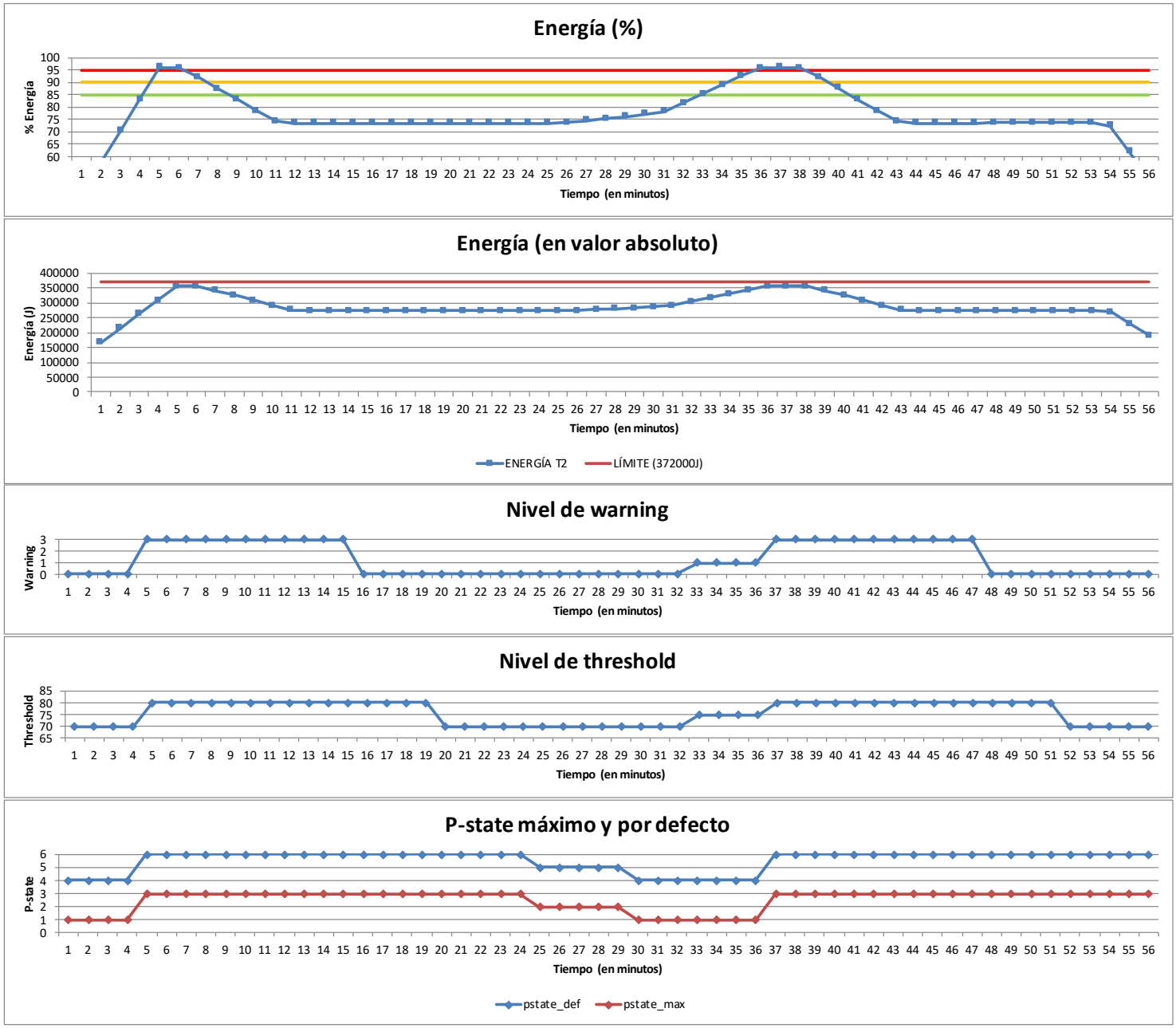


Figura D.4: Experimento 4.1 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

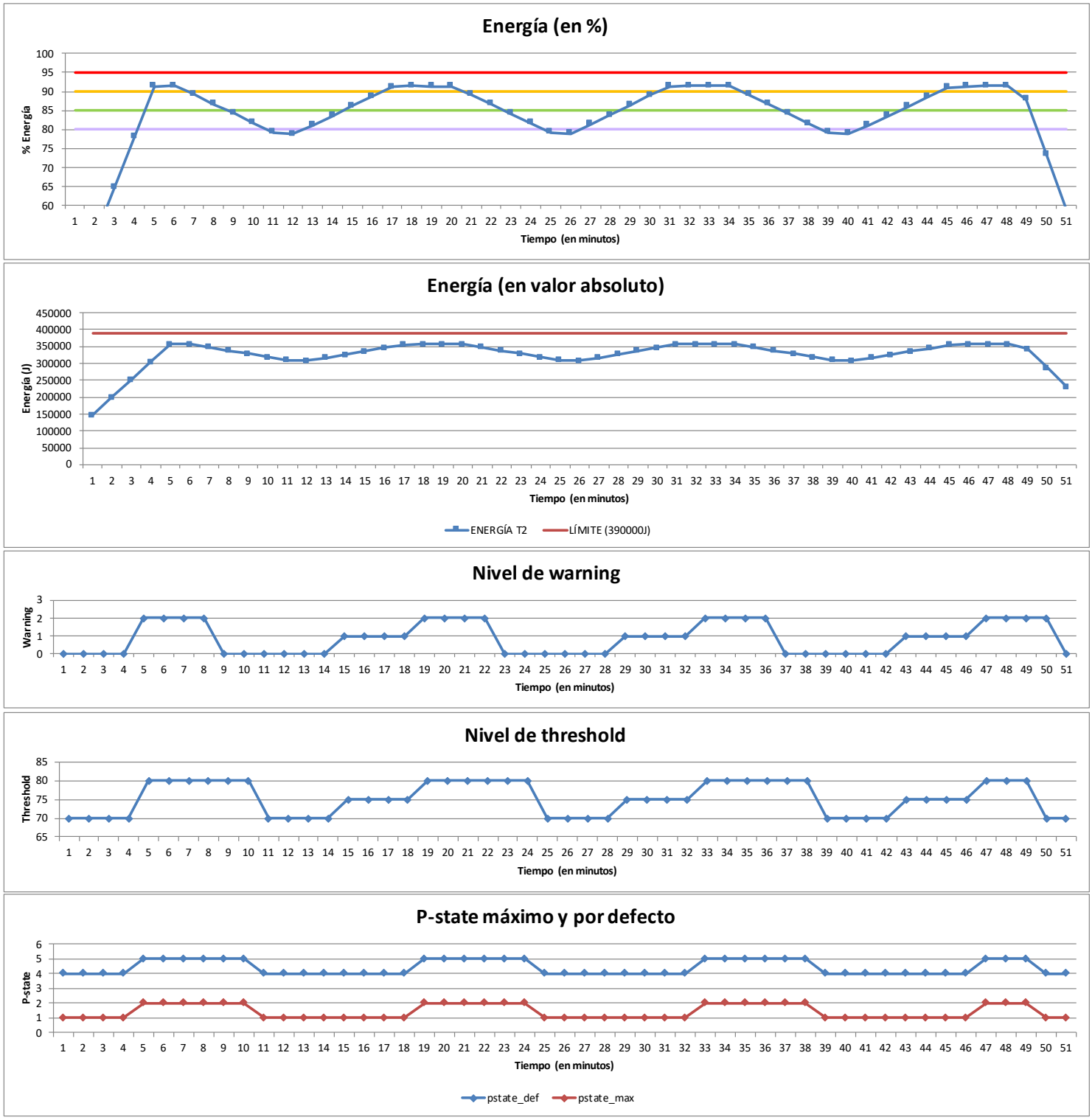


Figura D.5: Experimento 4.1 con optimización ramp down ajustado al nivel WARNING2.

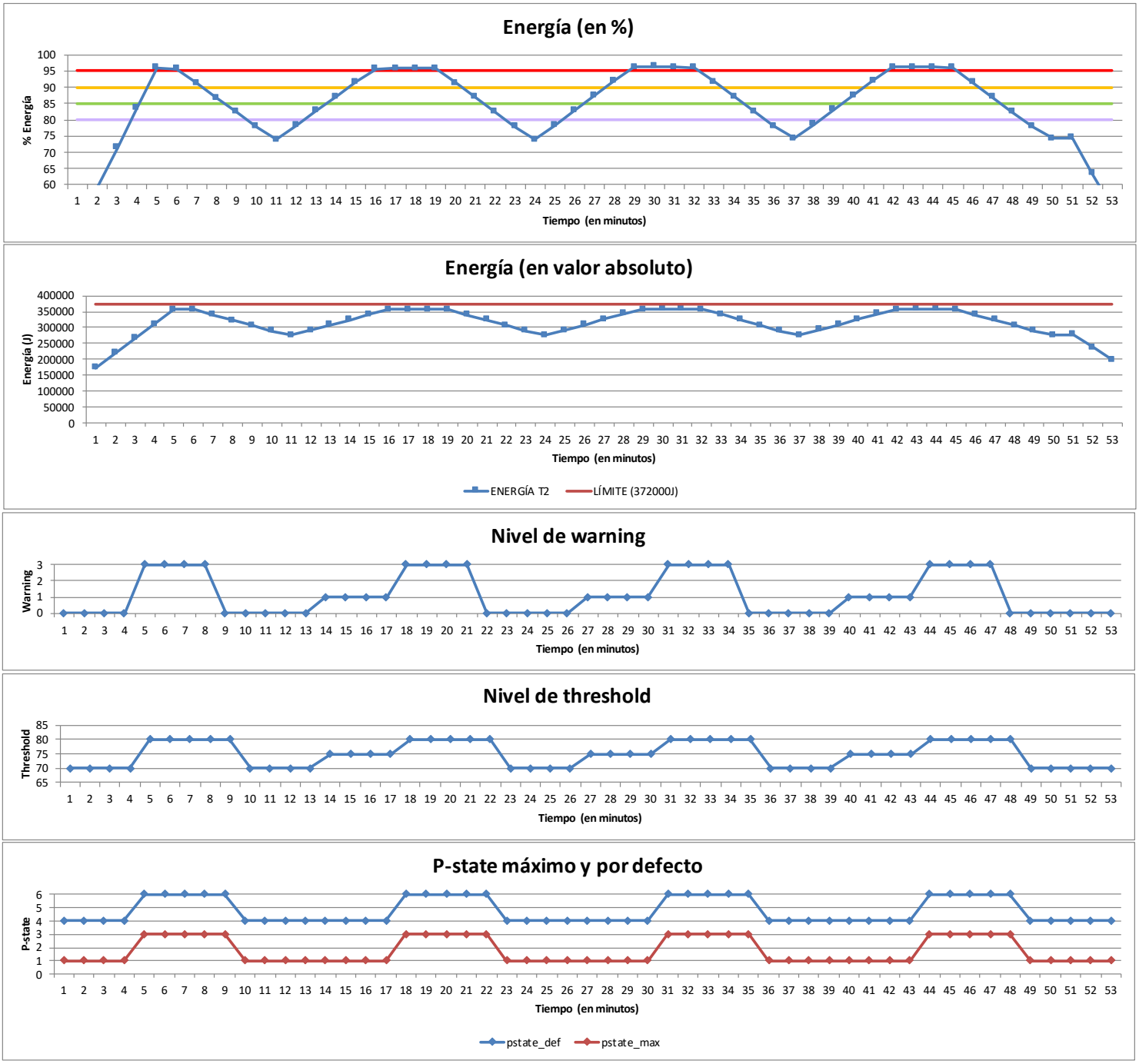


Figura D.6: Experimento 4.1 con optimización ramp down ajustado al nivel PANIC.

D.2. Exp. 4.2: Aplicación intensiva en memoria

En la figura D.7 se muestra el comportamiento del experimento ajustando al nivel NO PROBLEM.

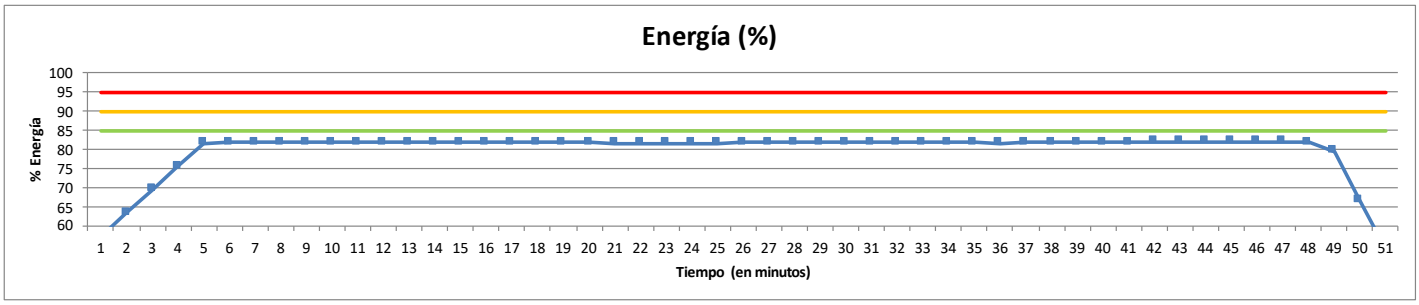


Figura D.7: Comportamiento del experimento 4.2 sin acciones del EARGM.

Optimización ramp up:

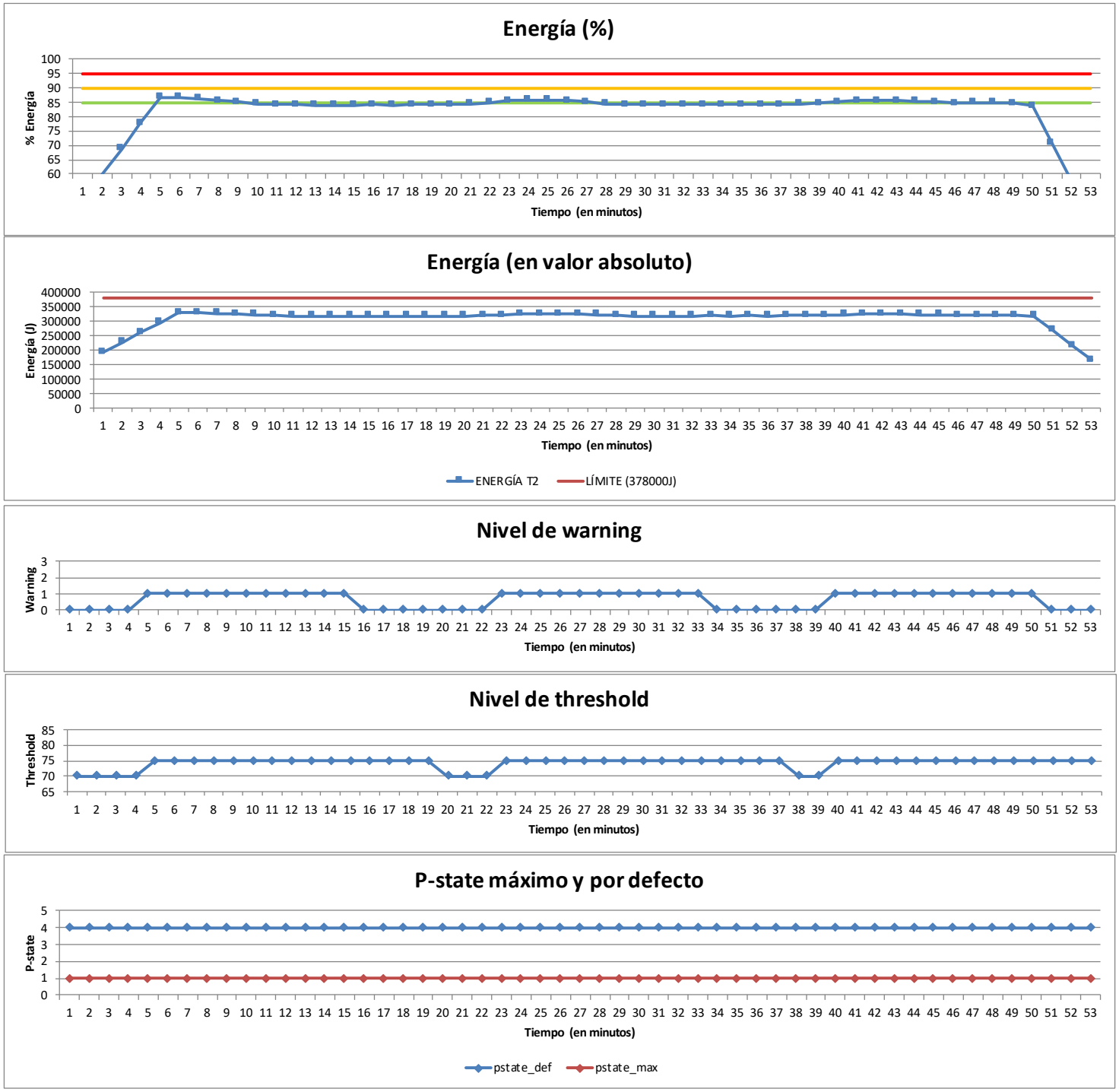


Figura D.8: Experimento 4.2 con optimización ramp up ajustado al nivel WARNING1.

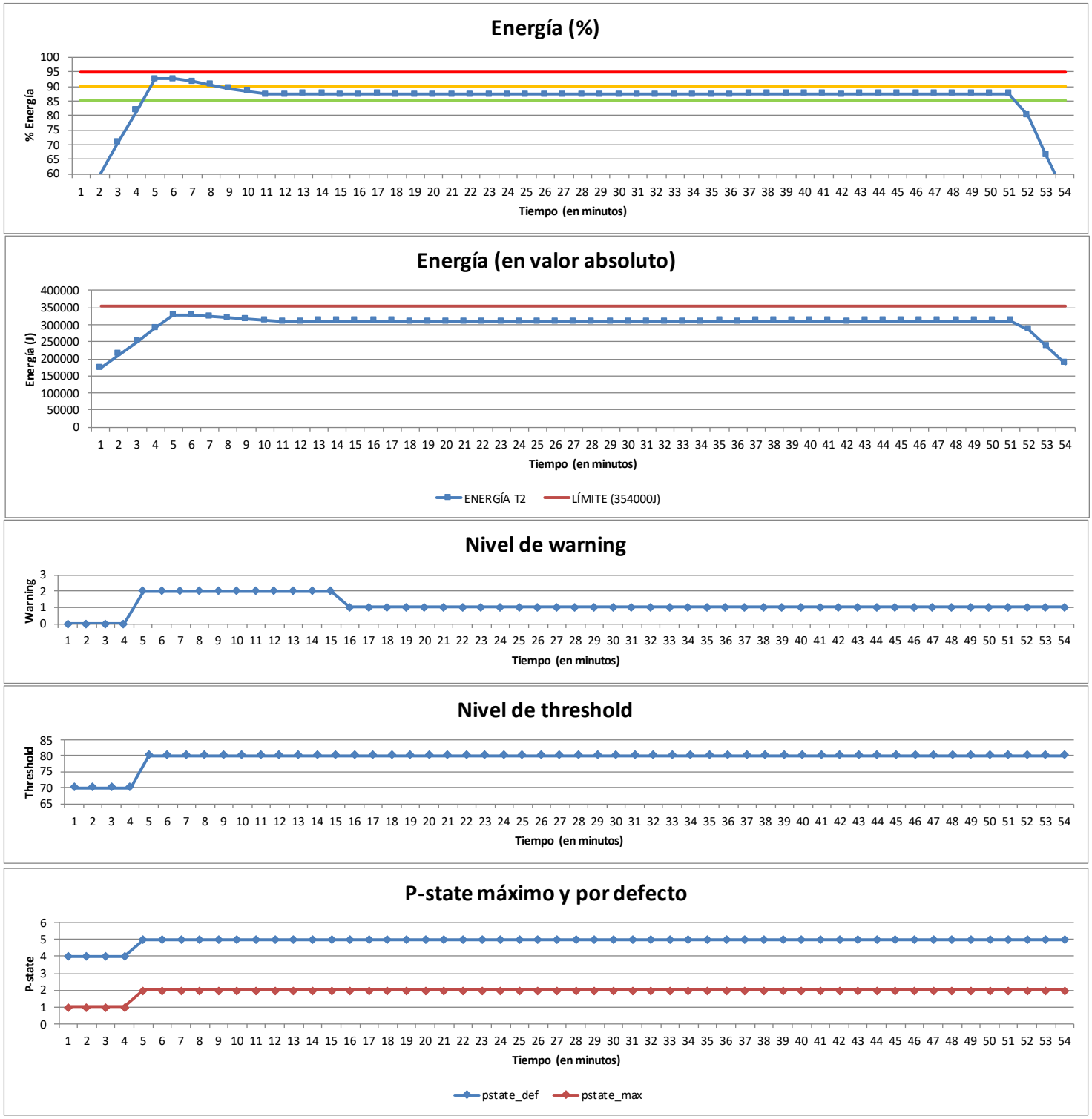


Figura D.9: Experimento 4.2 con optimización ramp up ajustado al nivel WARNING2.

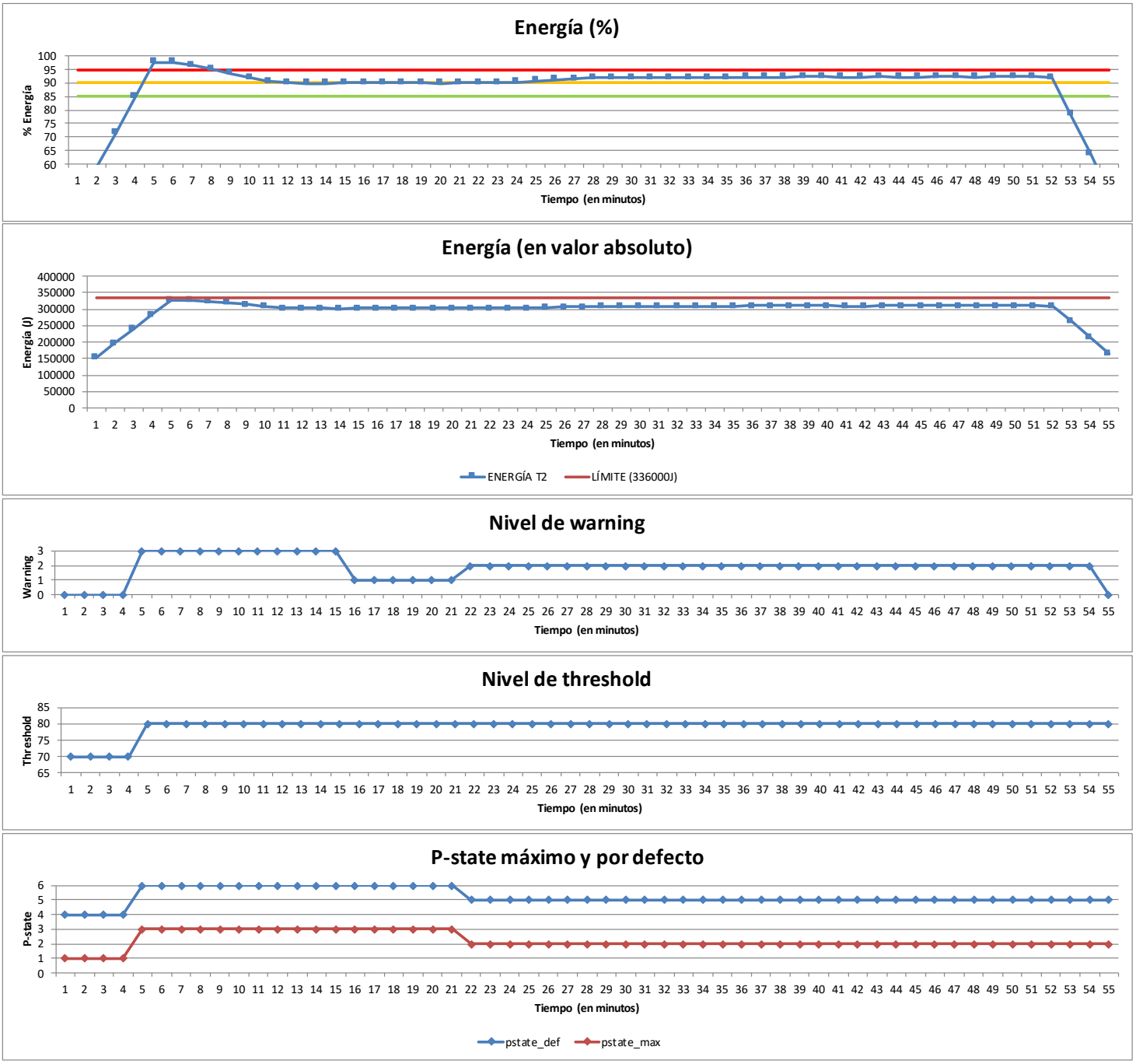


Figura D.10: Experimento 4.2 con optimización ramp up ajustado al nivel PANIC.

D.3. Exp. 4.3: Kernel intensivo en memoria

En la figura D.11 se muestra el comportamiento del experimento ajustando al nivel NO PROBLEM.

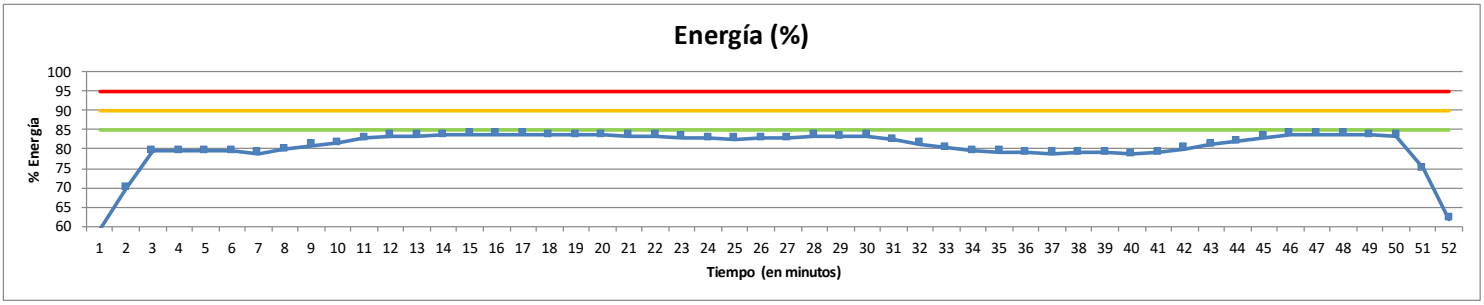


Figura D.11: Comportamiento del experimento 4.3 sin acciones del EARGM.

Optimización ramp up:

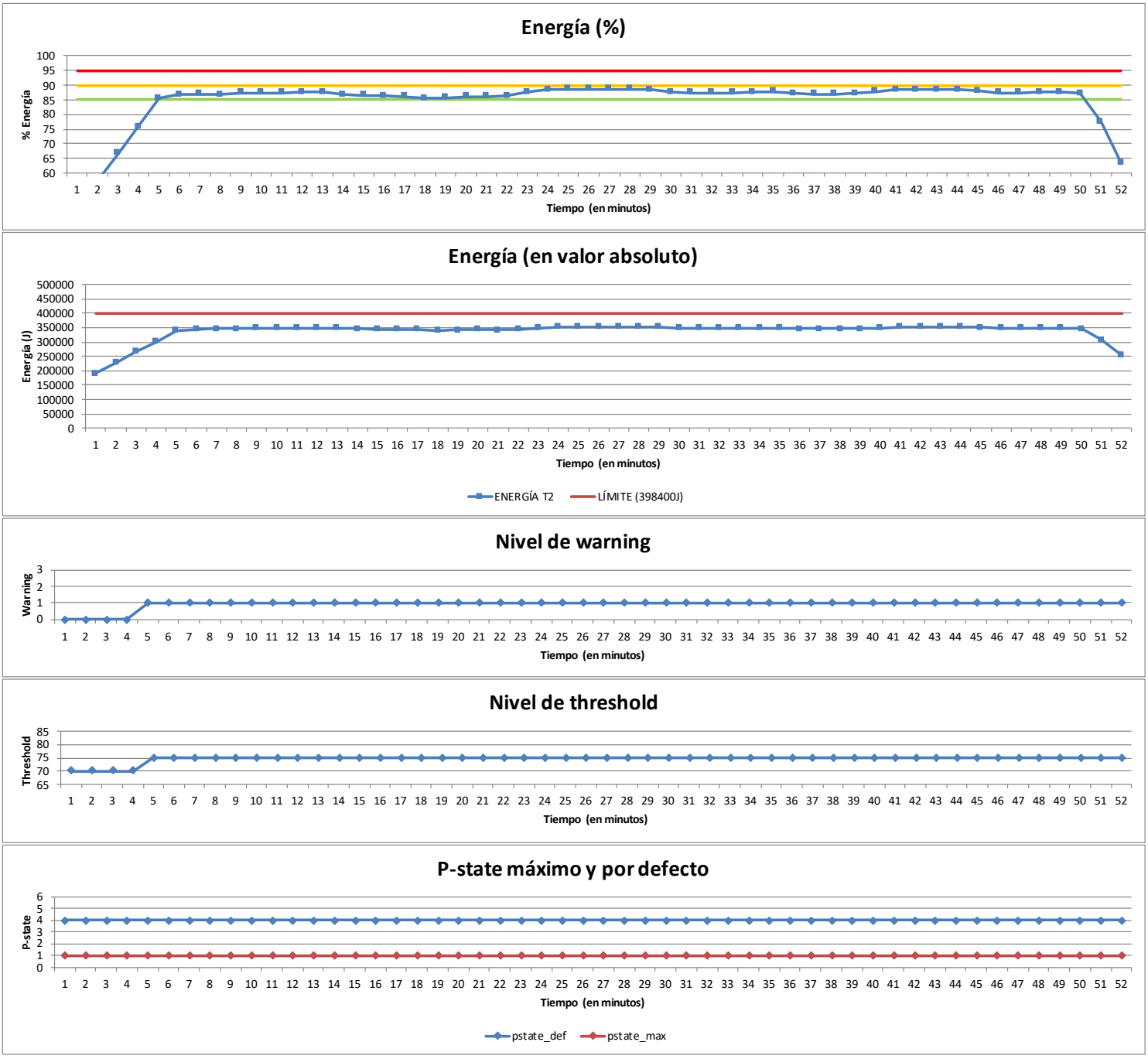


Figura D.12: Experimento 4.3 con optimización ramp up ajustado al nivel WARNING1.

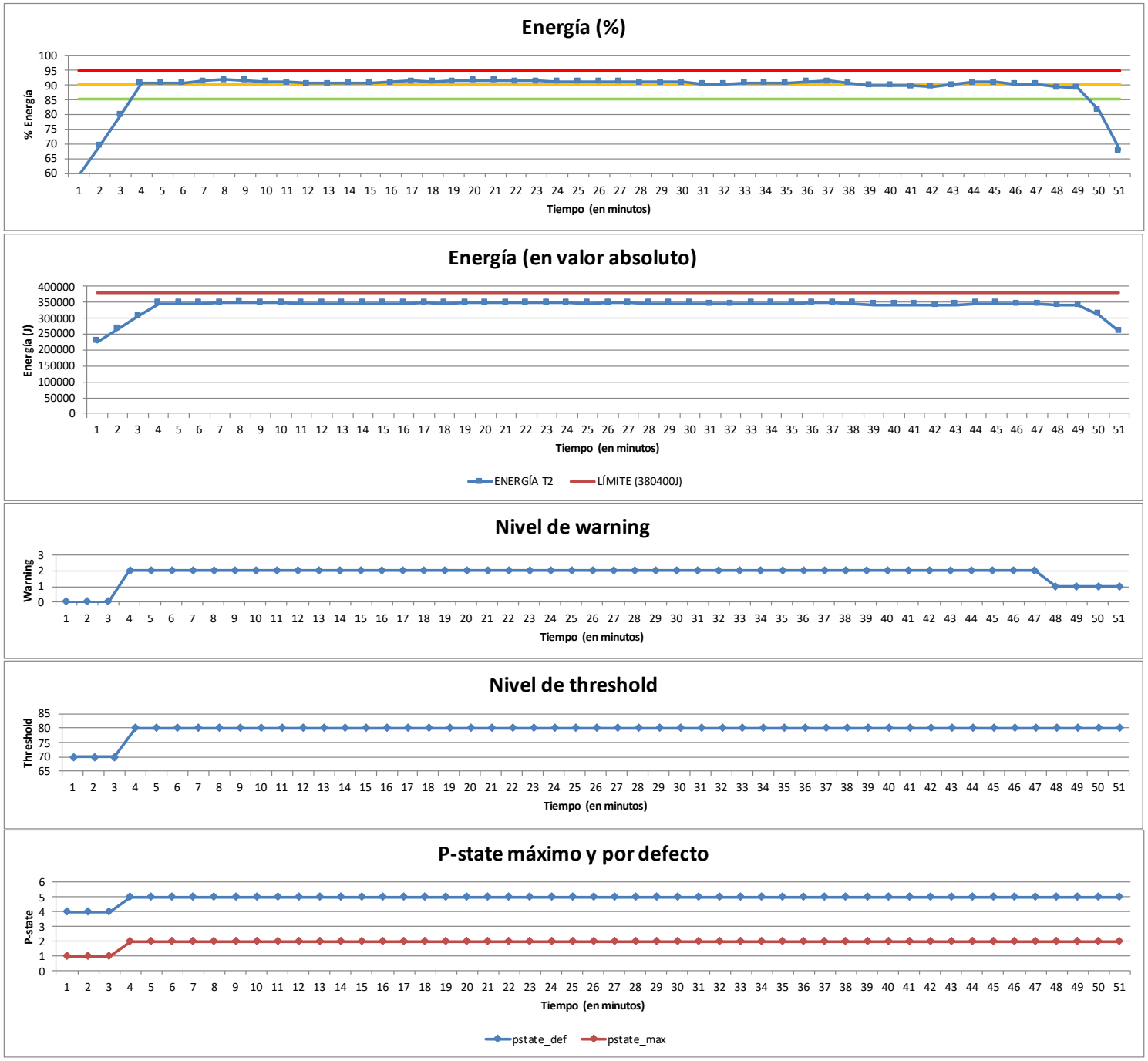


Figura D.13: Experimento 4.3 con optimización ramp up ajustado al nivel WARNING2.

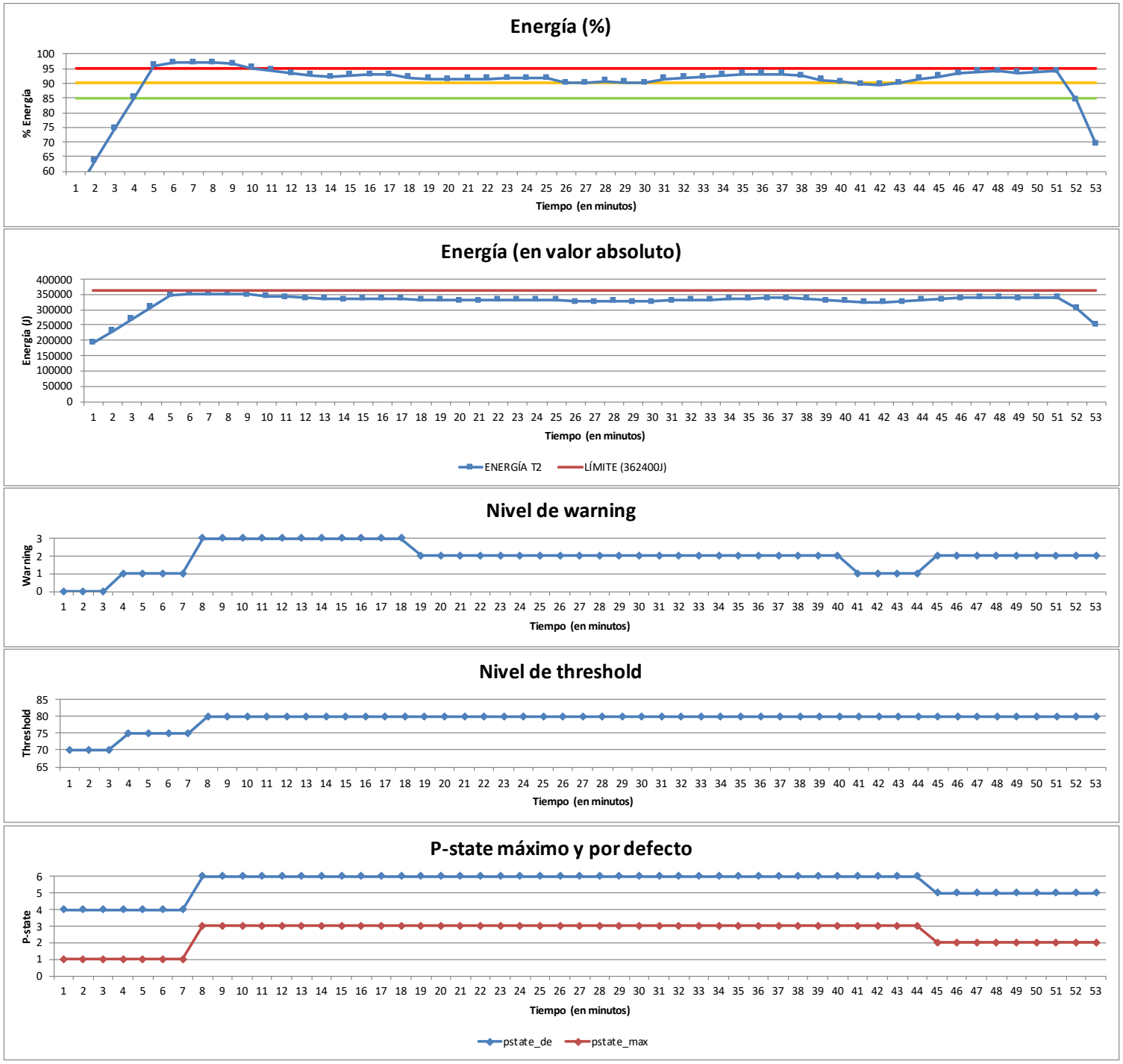


Figura D.14: Experimento 4.3 con optimización ramp up ajustado al nivel PANIC.

D.4. Exp. 4.4: Mix estático

En la figura D.15 se muestra el comportamiento del experimento ajustando al nivel NO PROBLEM.

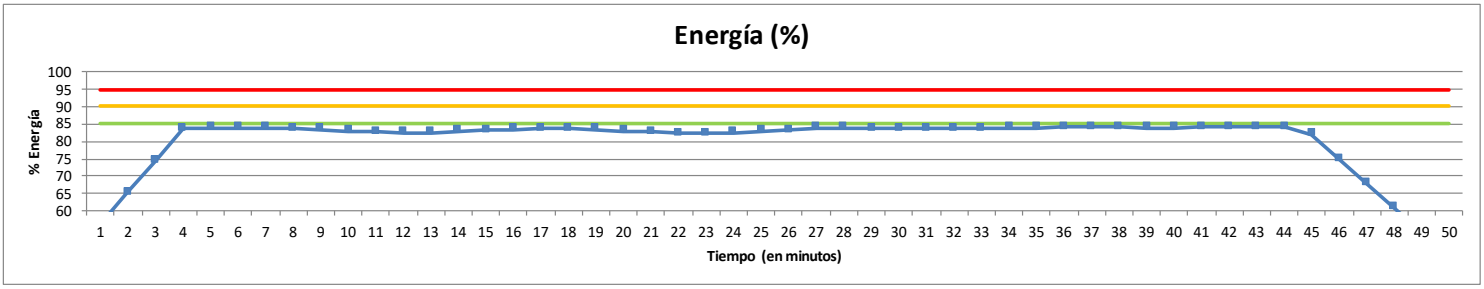


Figura D.15: Comportamiento del experimento 4.4 sin acciones del EARGM.

Optimización ramp up:

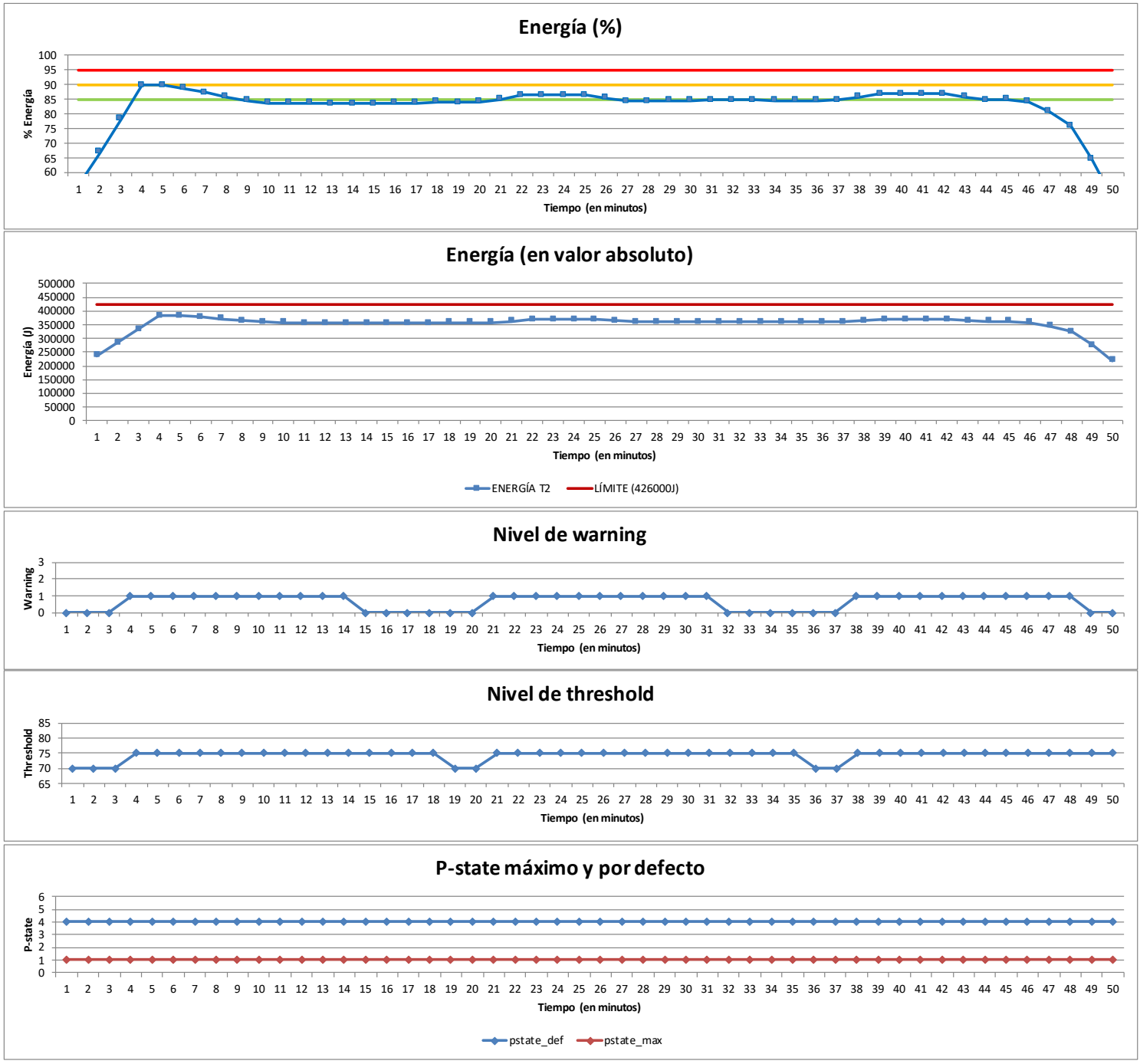


Figura D.16: Experimento 4.4 con optimización ramp up ajustado al nivel WARNING1.

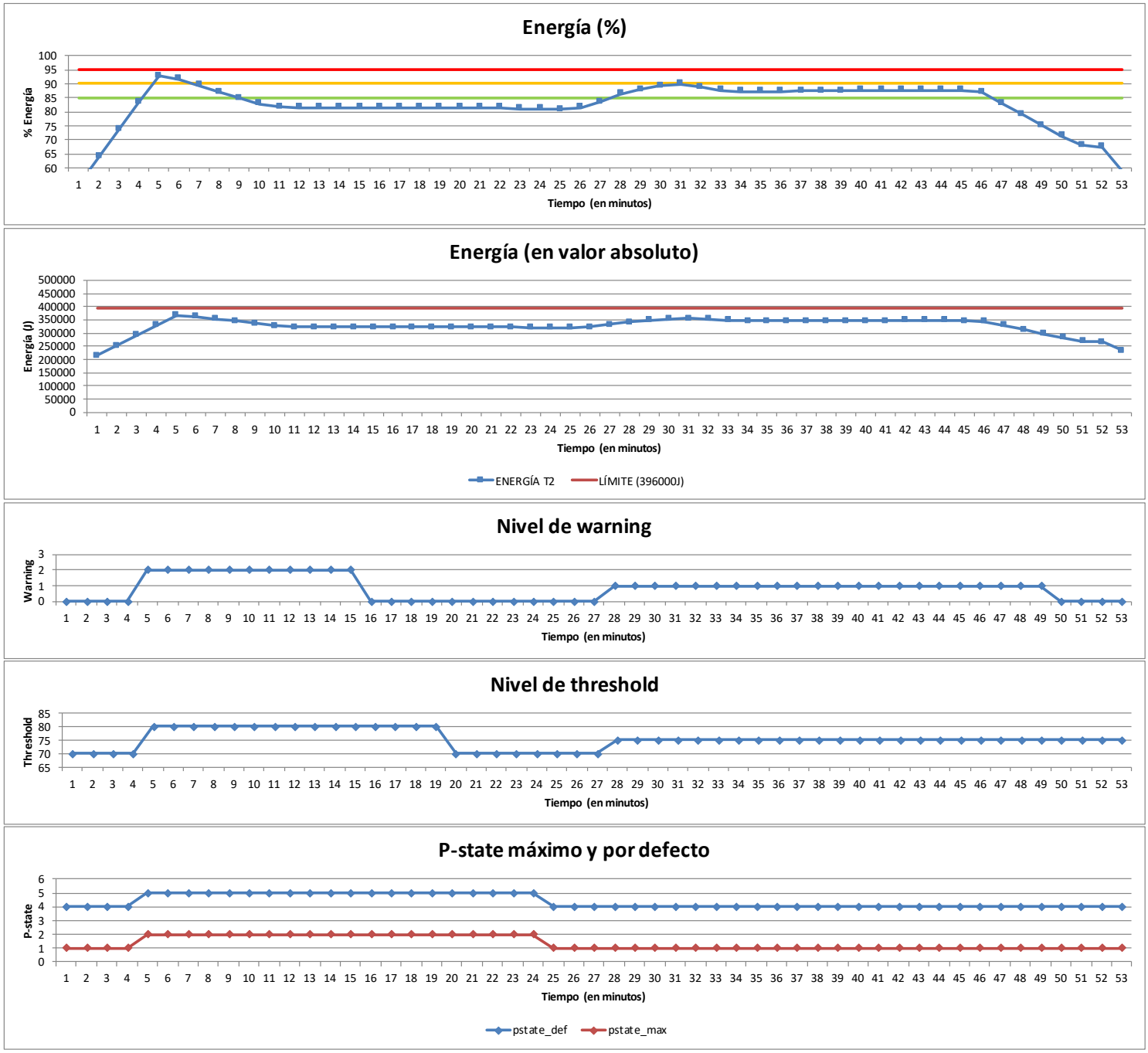


Figura D.17: Experimento 4.4 con optimización ramp up ajustado al nivel WARNING2.

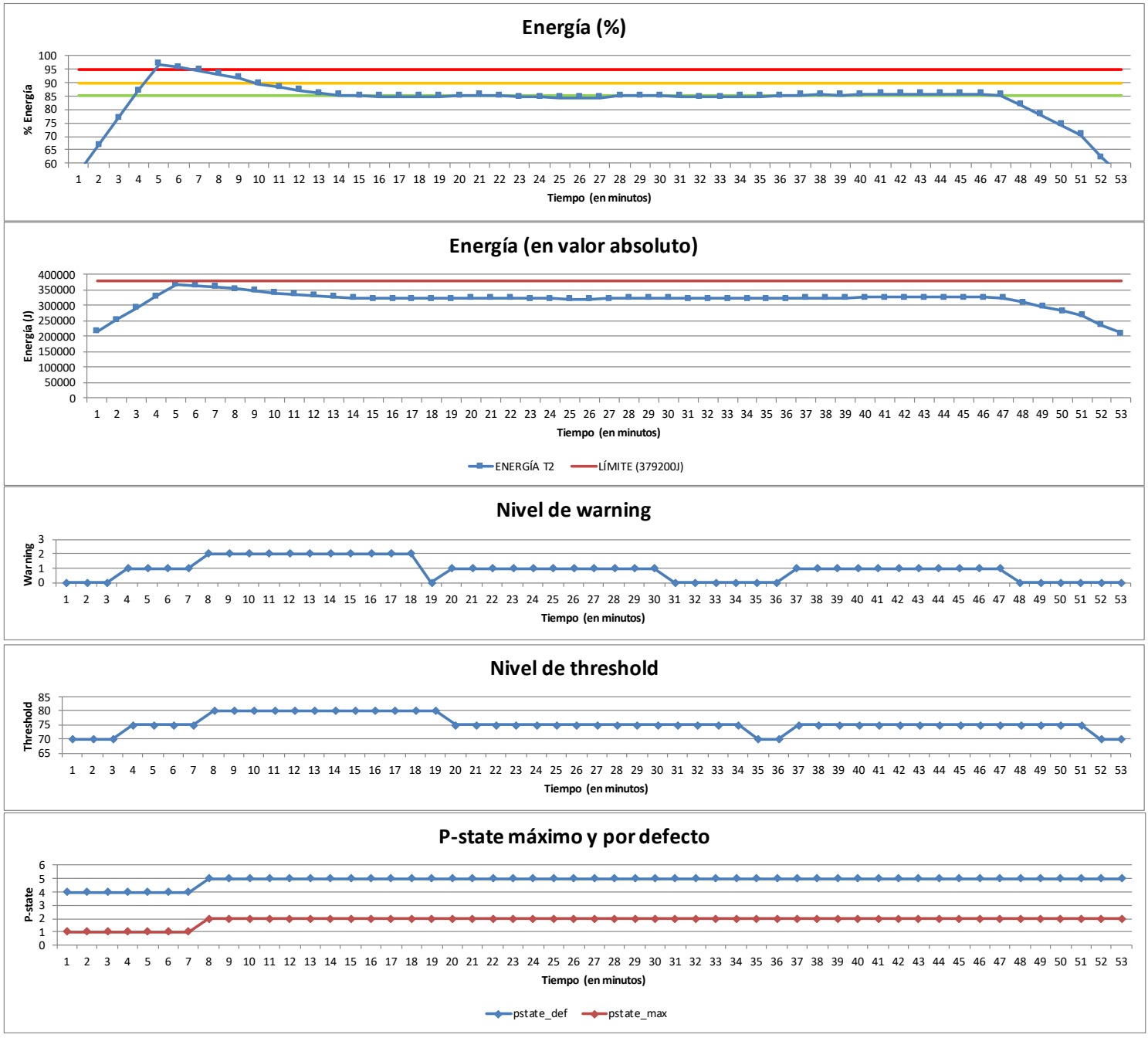


Figura D.18: Experimento 4.4 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

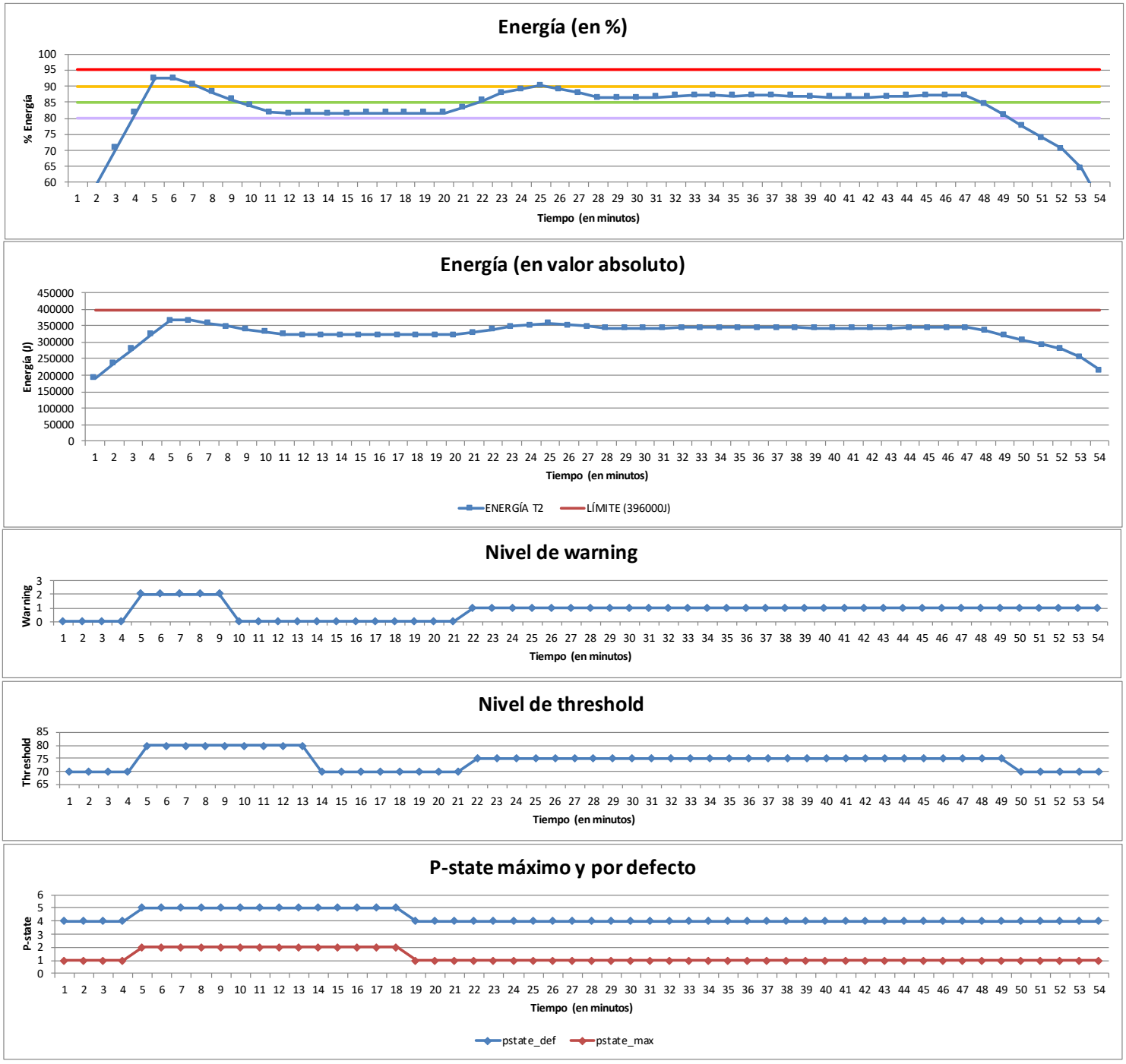


Figura D.19: Experimento 4.4 con optimización ramp down ajustado al nivel WARNING2.

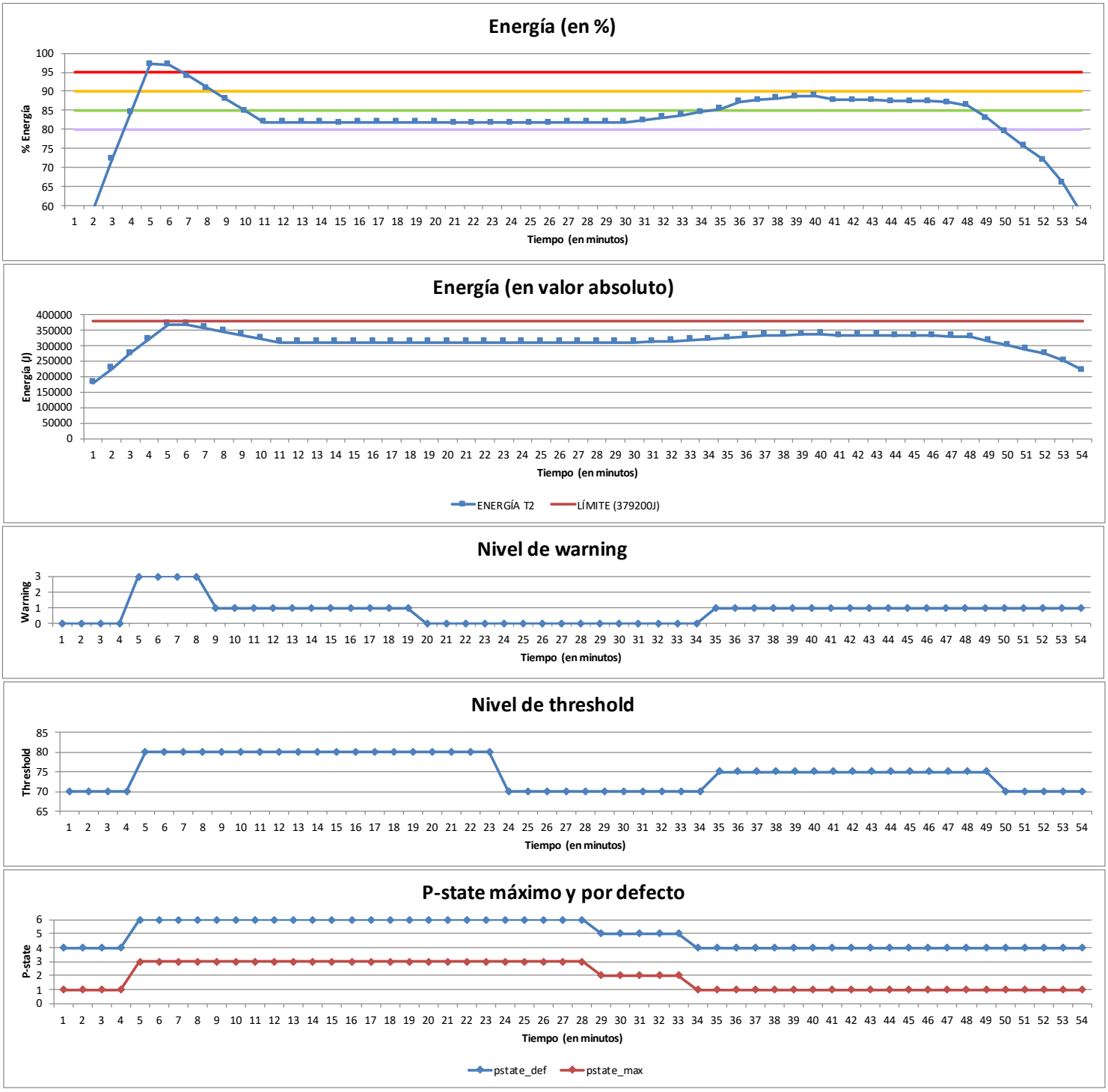


Figura D.20: Experimento 4.4 con optimización ramp down ajustado al nivel PANIC.

D.5. Exp. 4.5: Mix dinámico

En la figura D.21 se muestra el comportamiento del experimento ajustando al nivel NO PROBLEM.

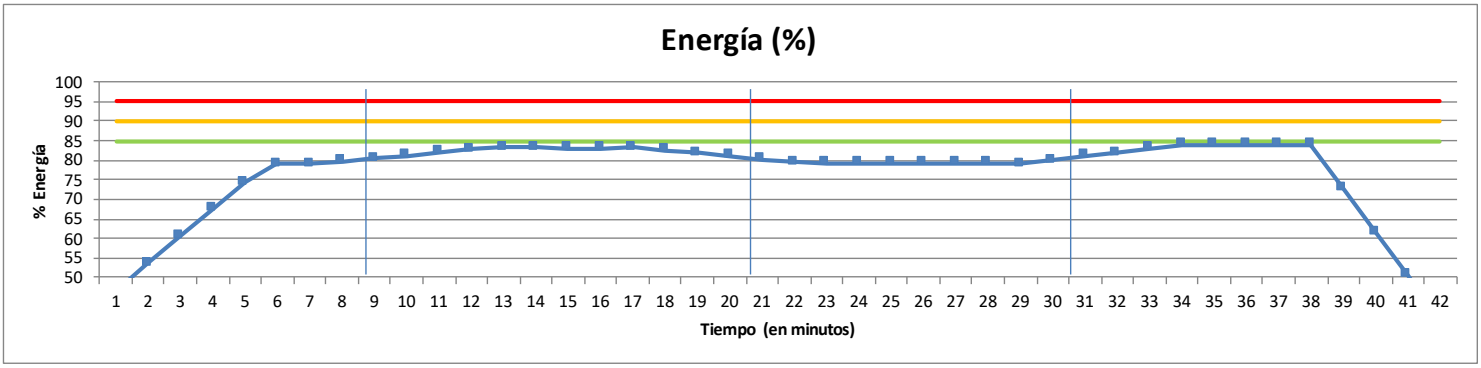


Figura D.21: Comportamiento del experimento 4.5 sin acciones del EARGM.

Optimización ramp up:

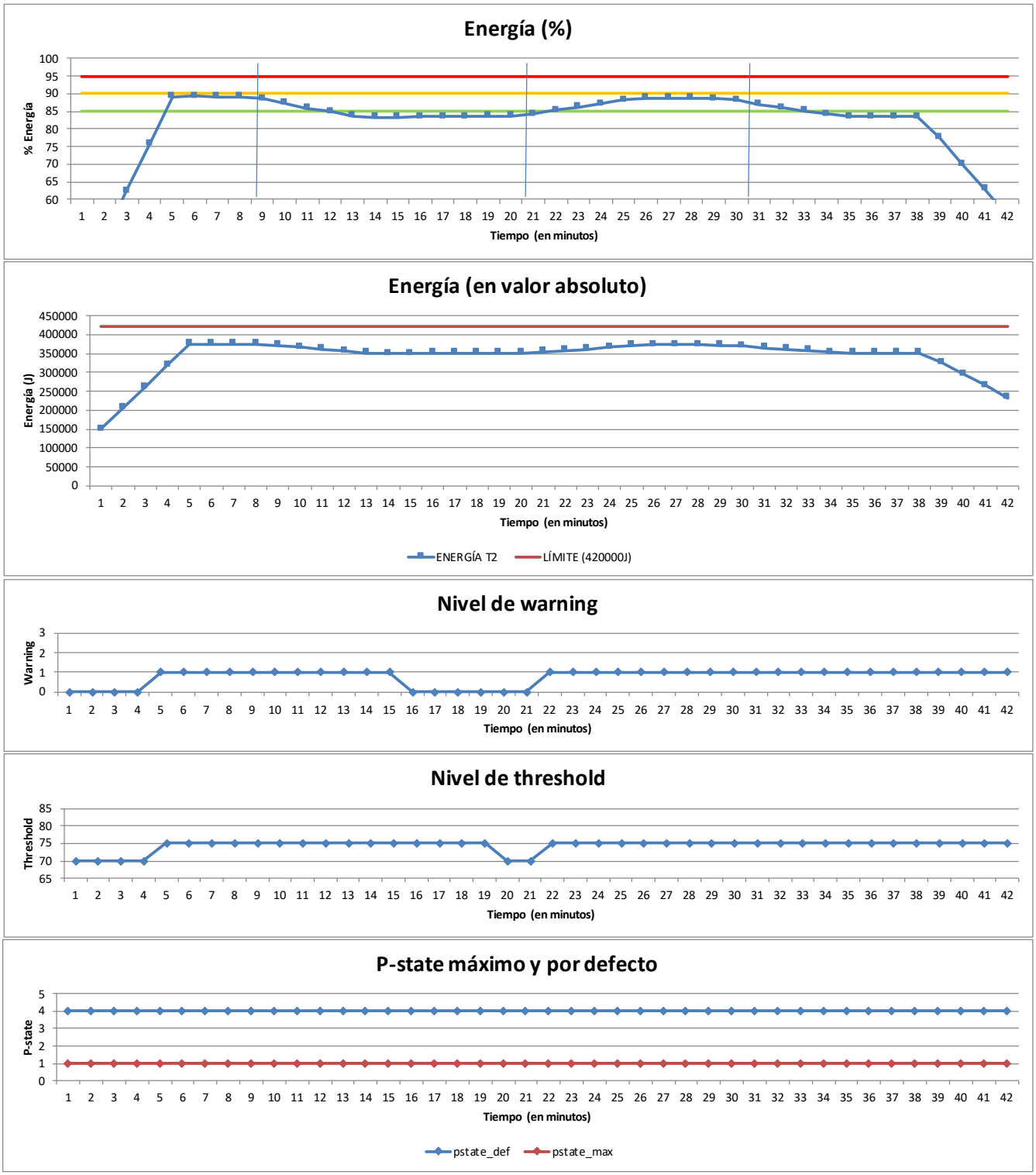


Figura D.22: Experimento 4.5 con optimización ramp up ajustado al nivel WARNING1.

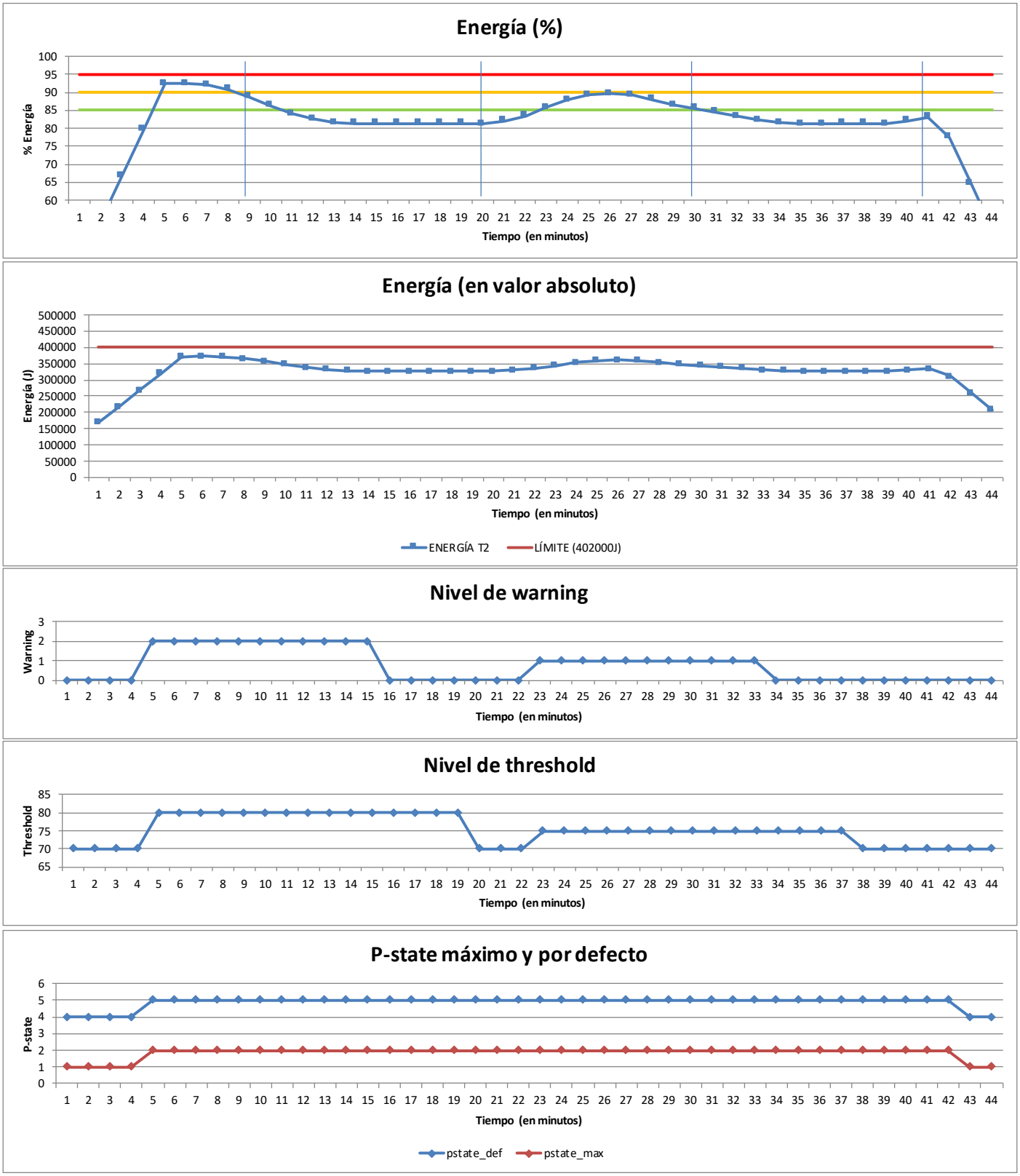


Figura D.23: Experimento 4.5 con optimización ramp up ajustado al nivel WARNING2.

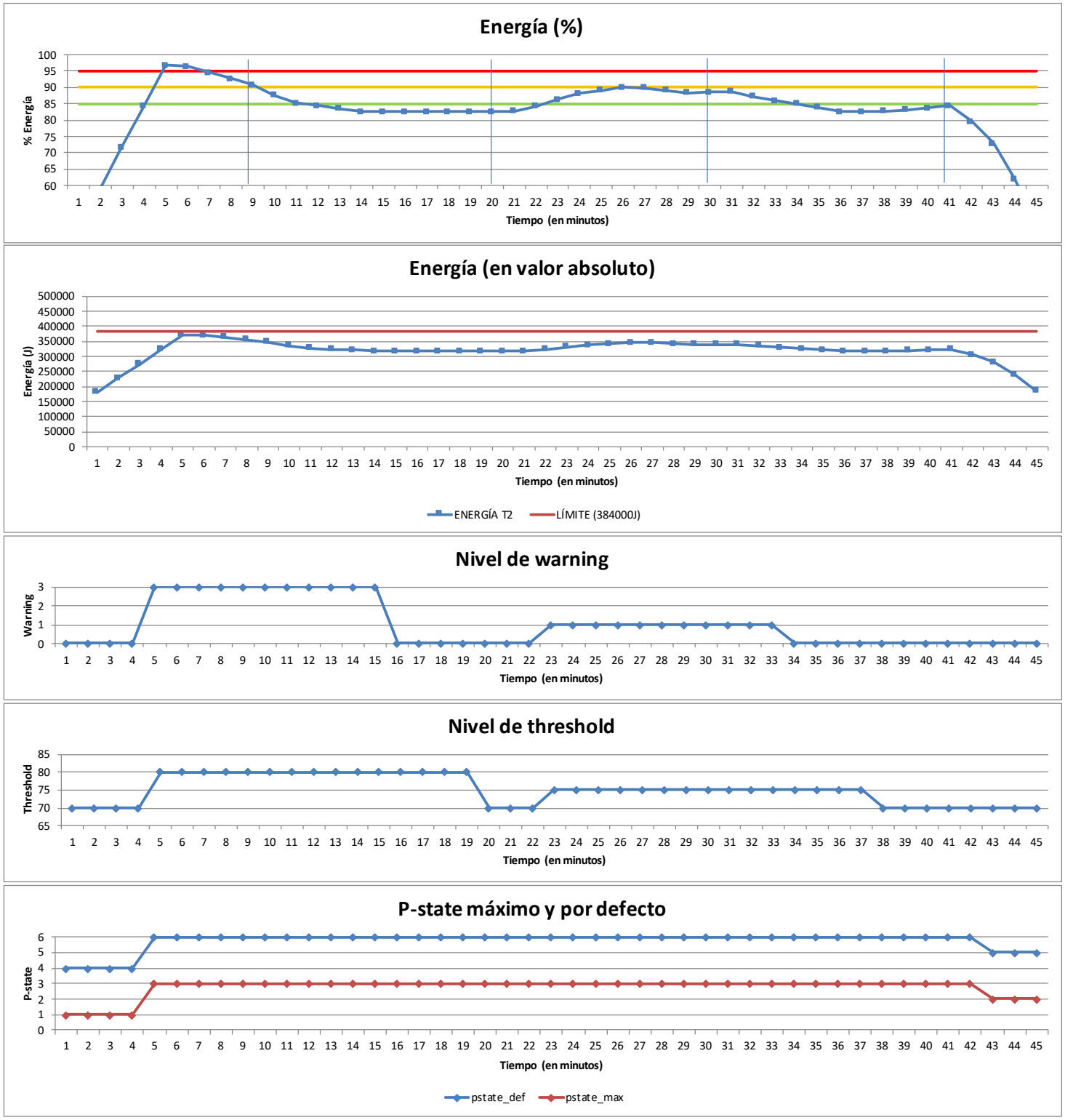


Figura D.24: Experimento 4.5 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

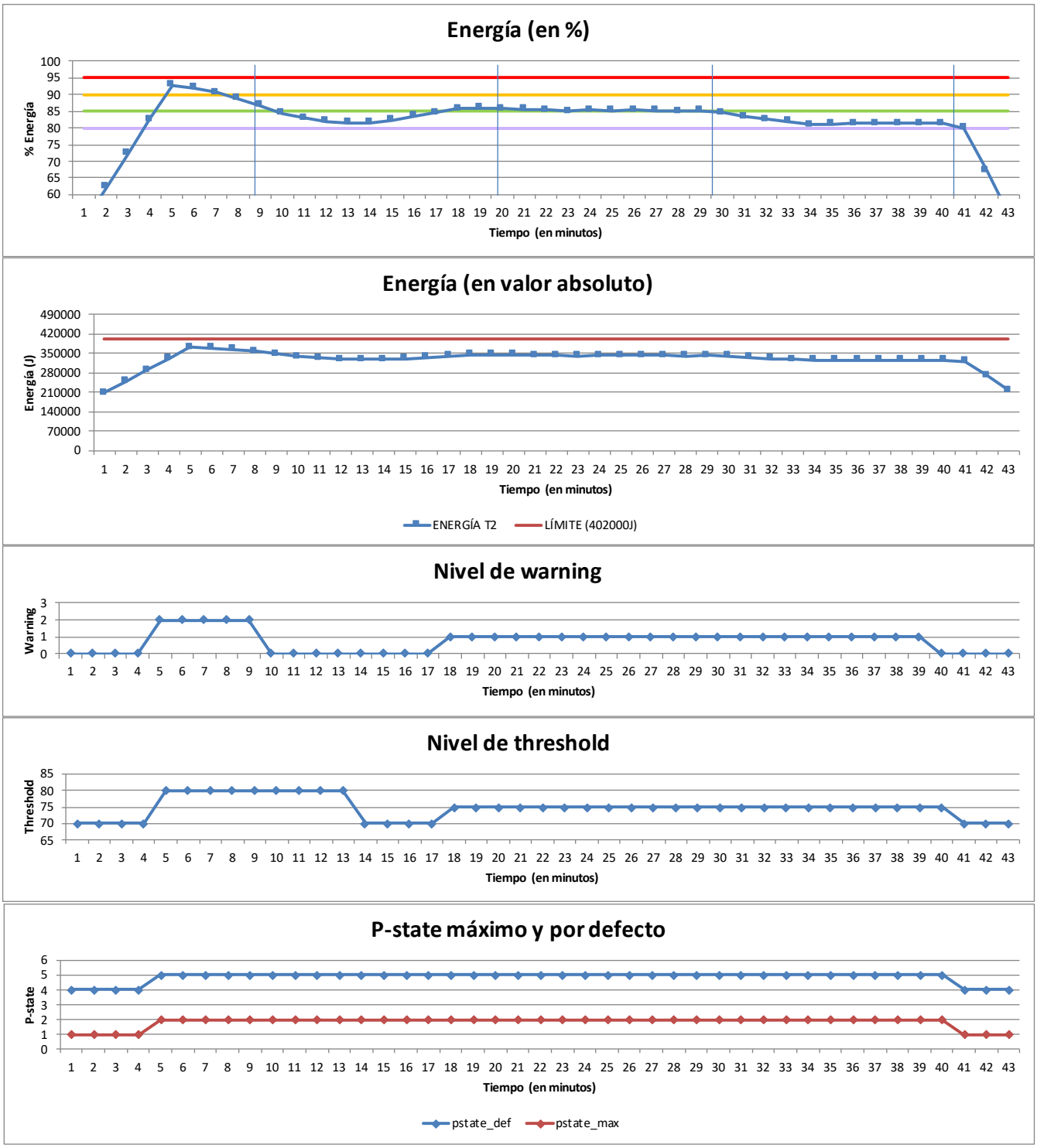


Figura D.25: Experimento 4.5 con optimización ramp down ajustado al nivel WARNING2.

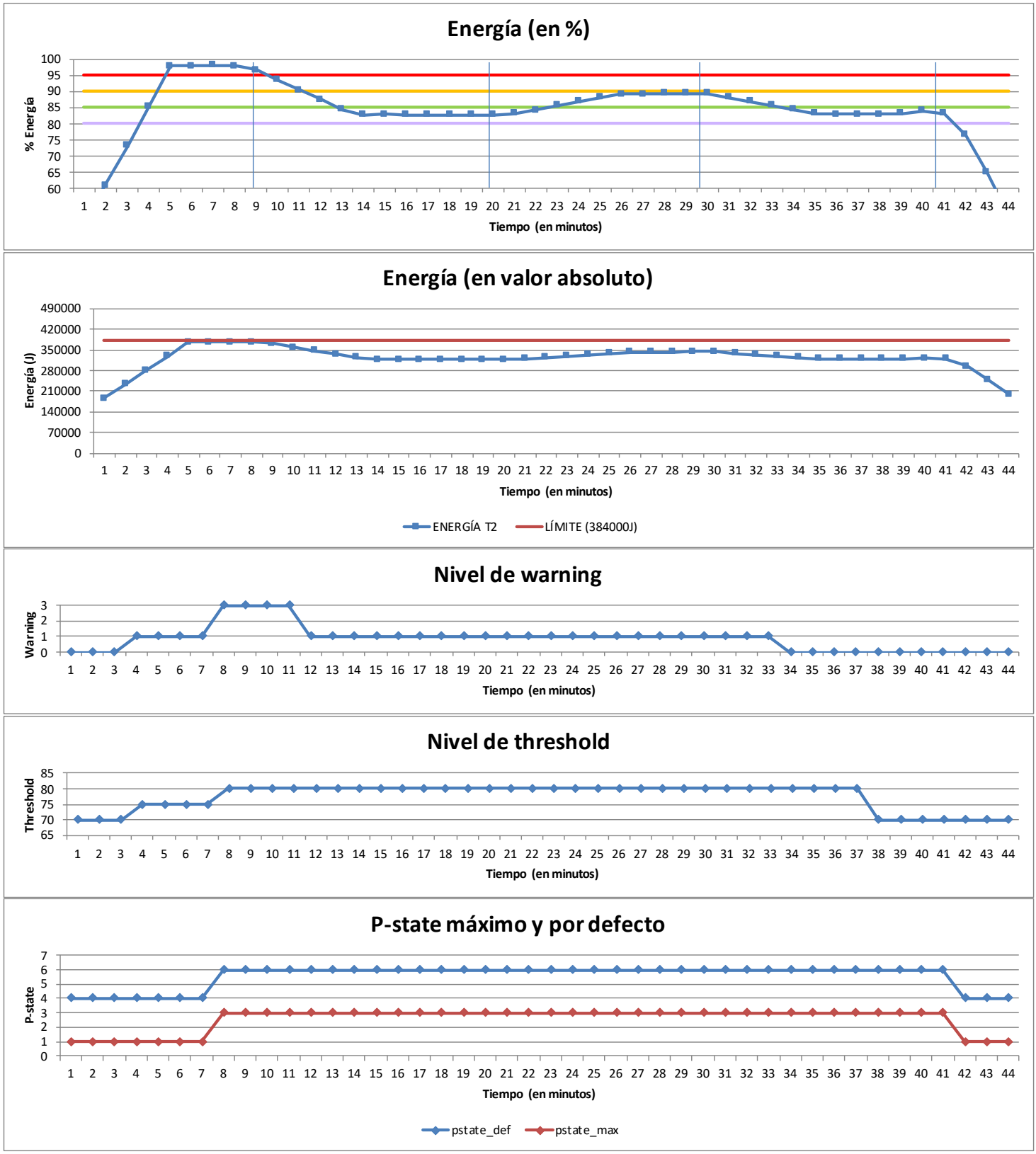


Figura D.26: Experimento 4.5 con optimización ramp down ajustado al nivel PANIC.

E Experimentos de rendimiento

En este apéndice se muestran los experimentos de rendimiento realizados: algunos están completos y otros parcialmente, ya que en el capítulo 12 se presentan algunos de ellos. De la misma manera que con los test de validación, por cada experimento se muestran los resultados de ajustar el límite energético para cada nivel de *warning* con la optimización ramp up. En caso pertinente, también se muestran para los niveles más críticos con la optimización ramp down. Estas pruebas se han realizado con 20 nodos (800 *cores*), excepto el experimento 26.1 que se ha efectuado con 26 nodos (1024 *cores*).

E.1. Exp. 26.1: Aplicación intensiva en memoria

Optimización ramp up:

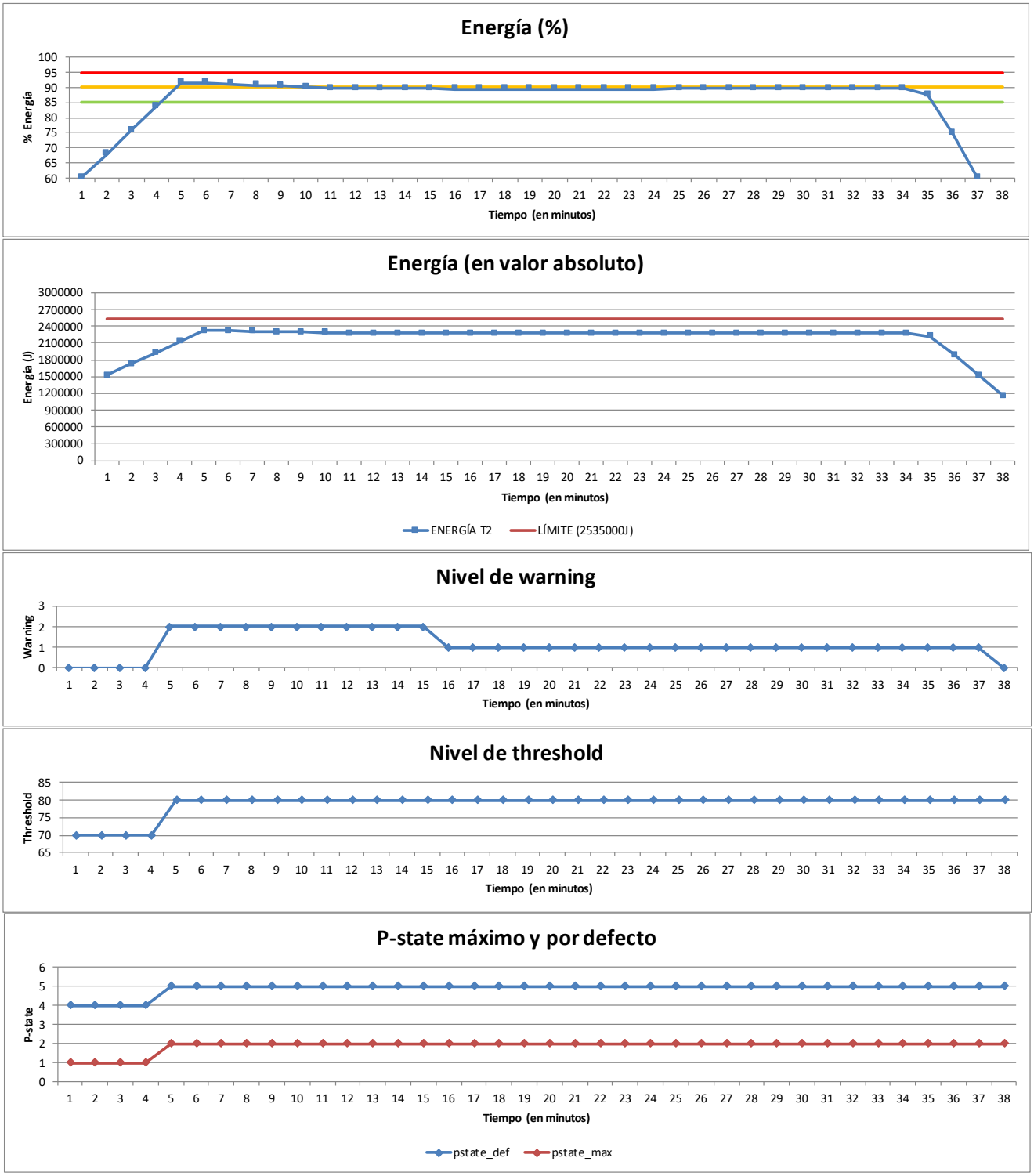


Figura E.1: Experimento 26.1 con optimización ramp up ajustado al nivel WARNING2.

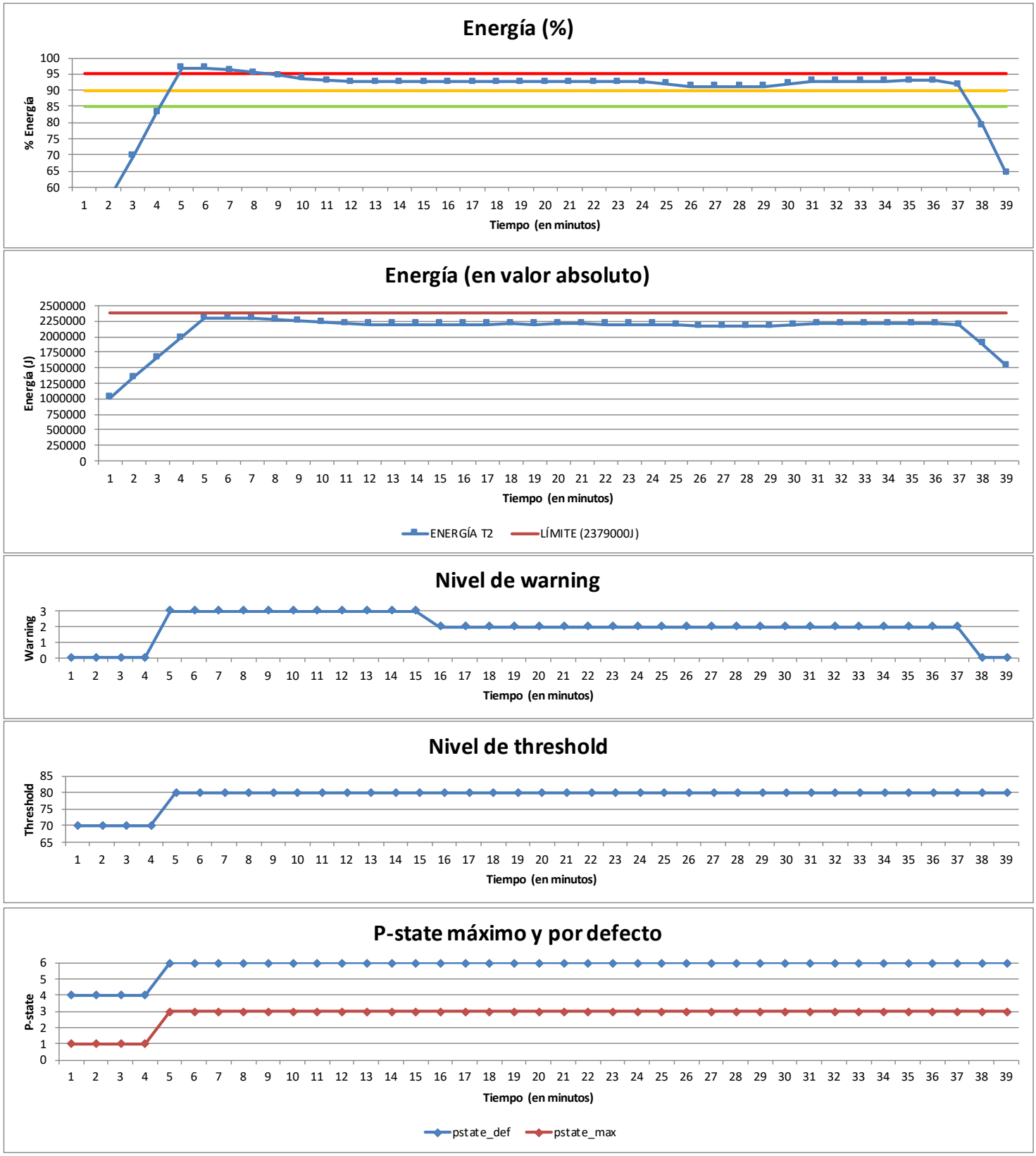


Figura E.2: Experimento 26.1 con optimización ramp up ajustado al nivel PANIC.

E.2. Exp. 20.2: Kernel intensivo en CPU de larga duración

Optimización ramp up:

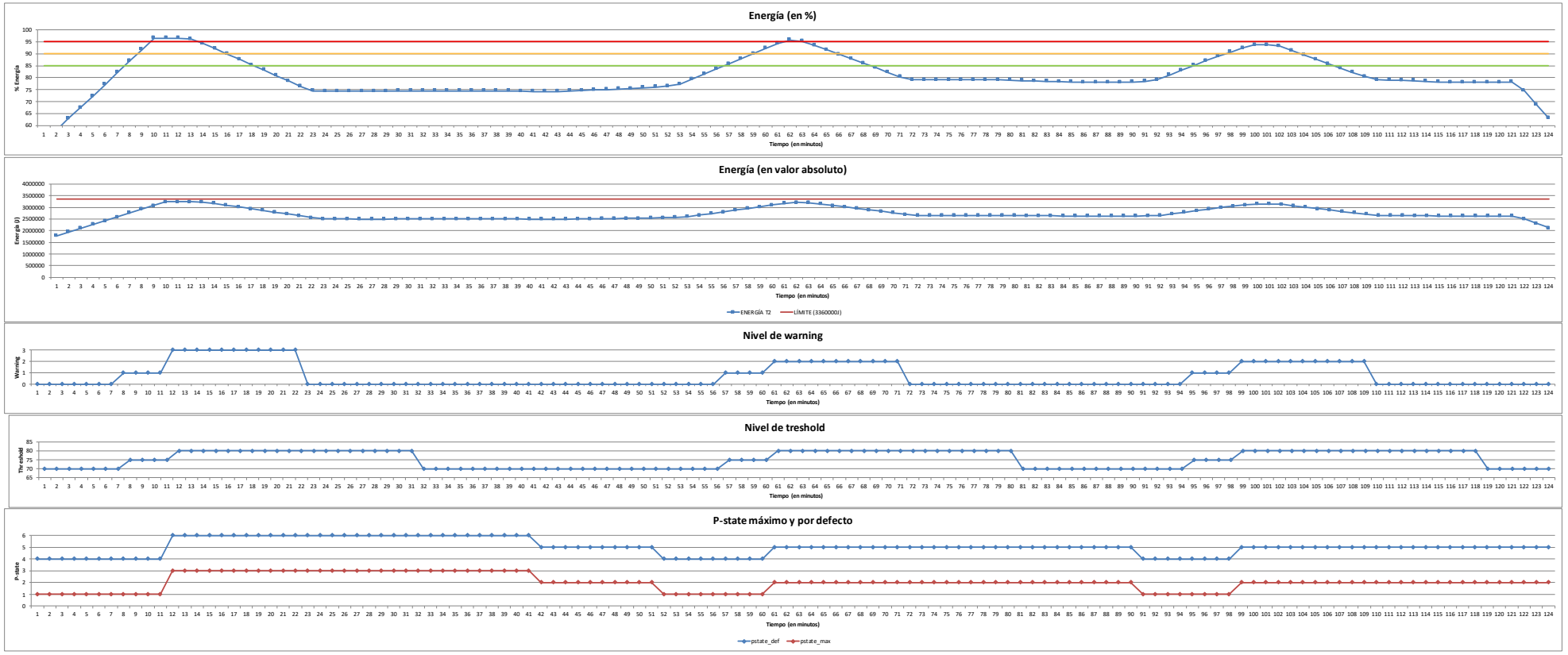


Figura E.3: Experimento 20.2 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

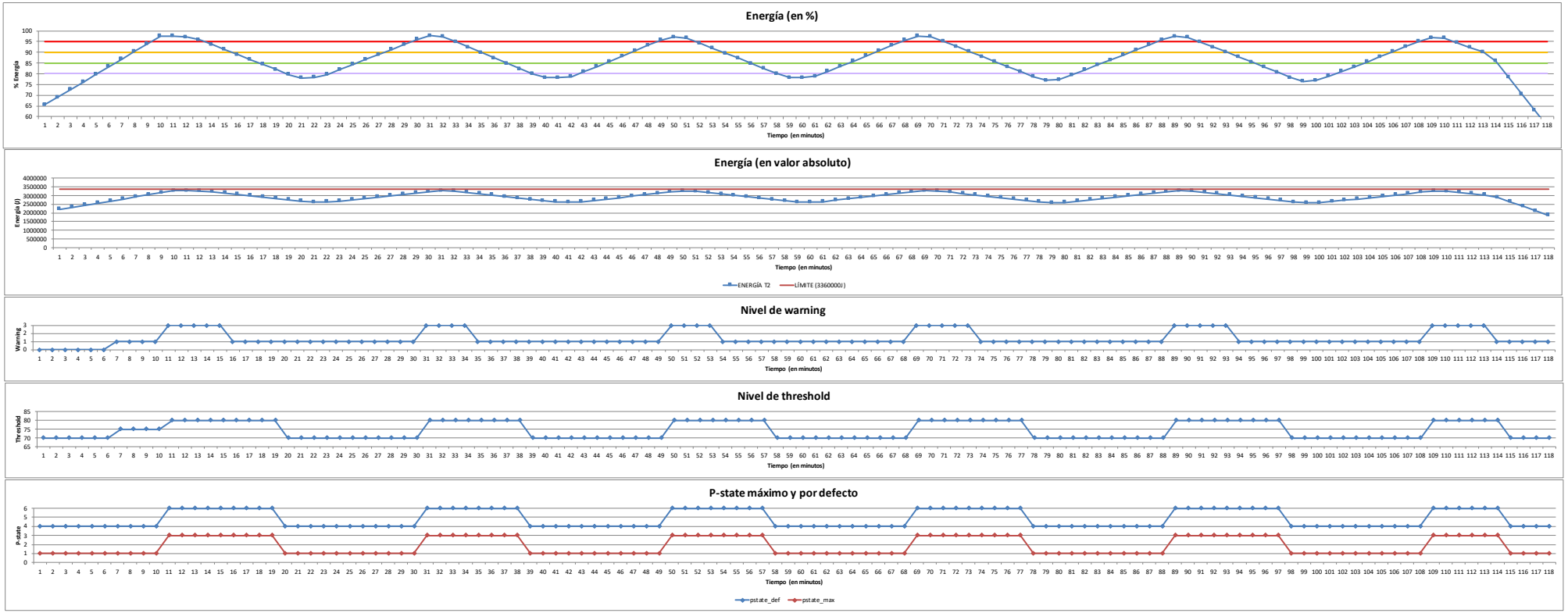


Figura E.4: Experimento 20.2 con optimización ramp down ajustado al nivel PANIC.

E.3. Exp. 20.3: Kernel intensivo en memoria

El comportamiento por defecto del experimento se muestra en la figura E.5.

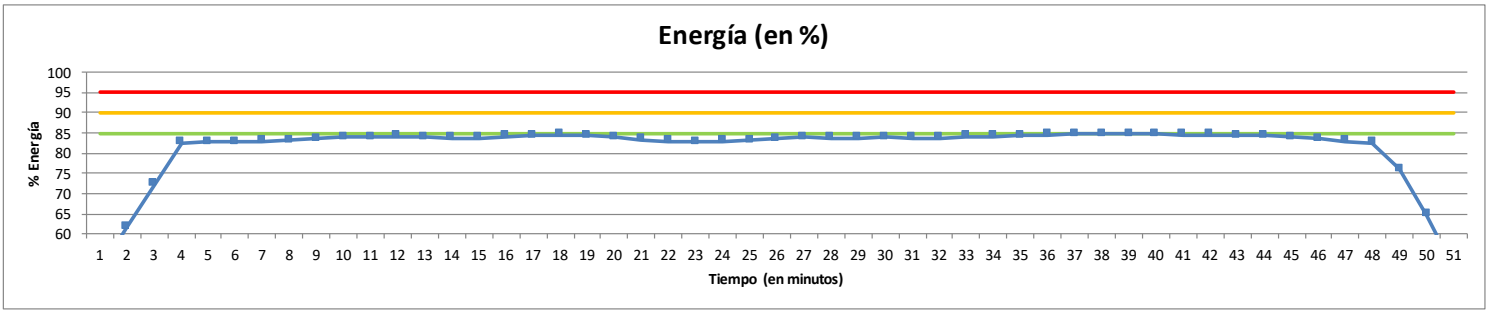


Figura E.5: Comportamiento del experimento 20.3 sin acciones del EARGM.

Optimización ramp up:

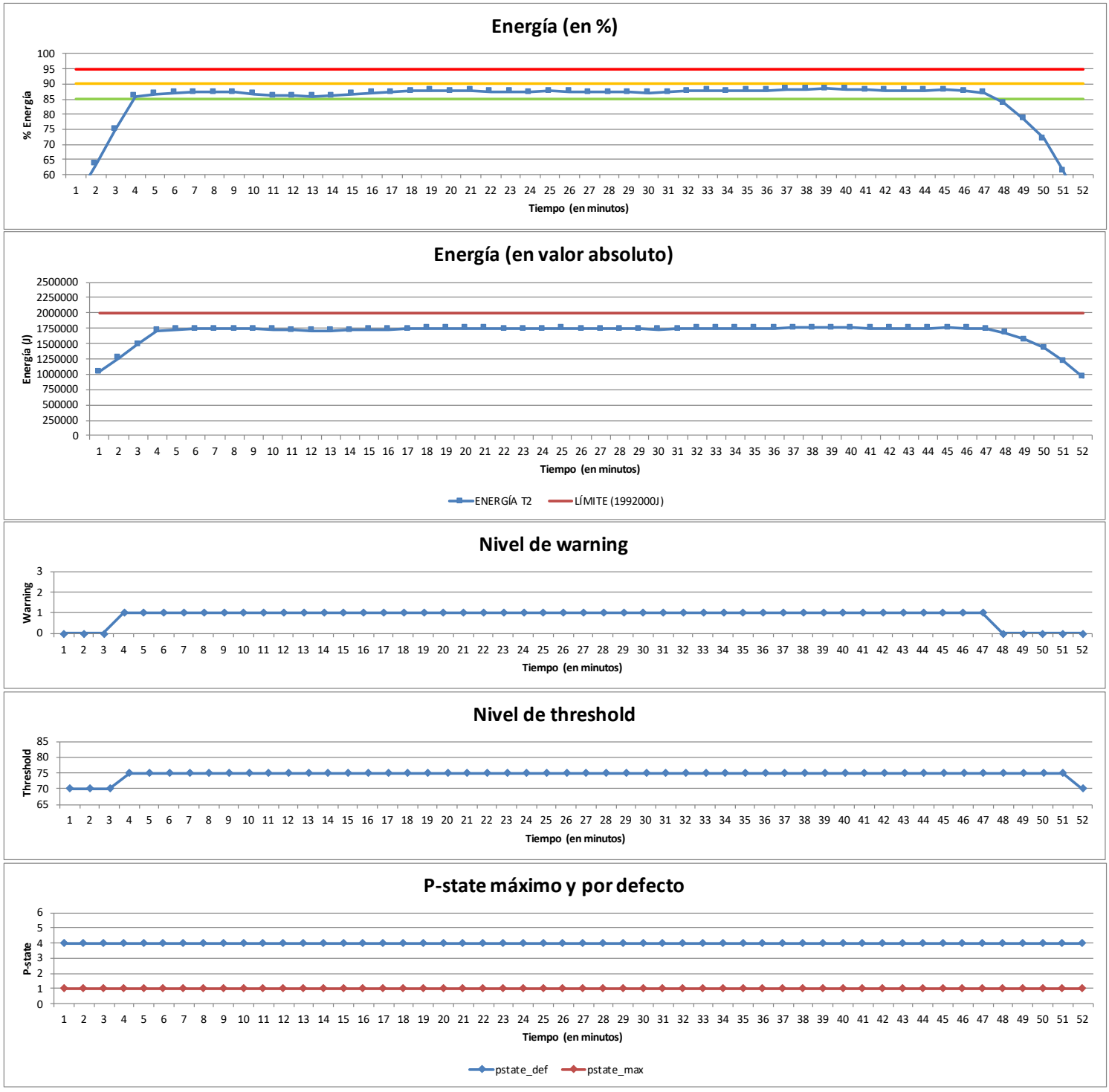


Figura E.6: Experimento 20.3 con optimización ramp up ajustado al nivel WARNING1.

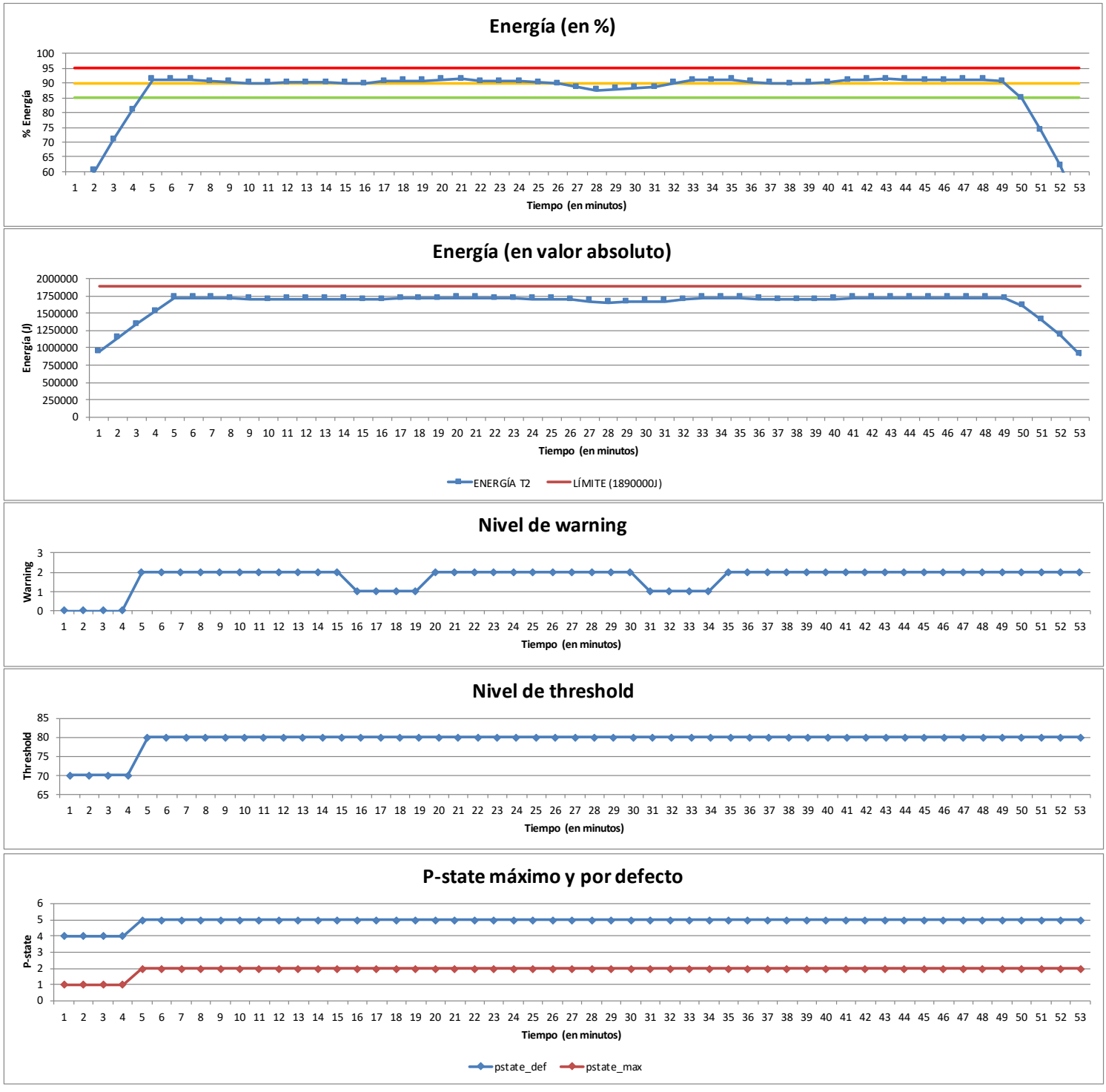


Figura E.7: Experimento 20.3 con optimización ramp up ajustado al nivel WARNING2.

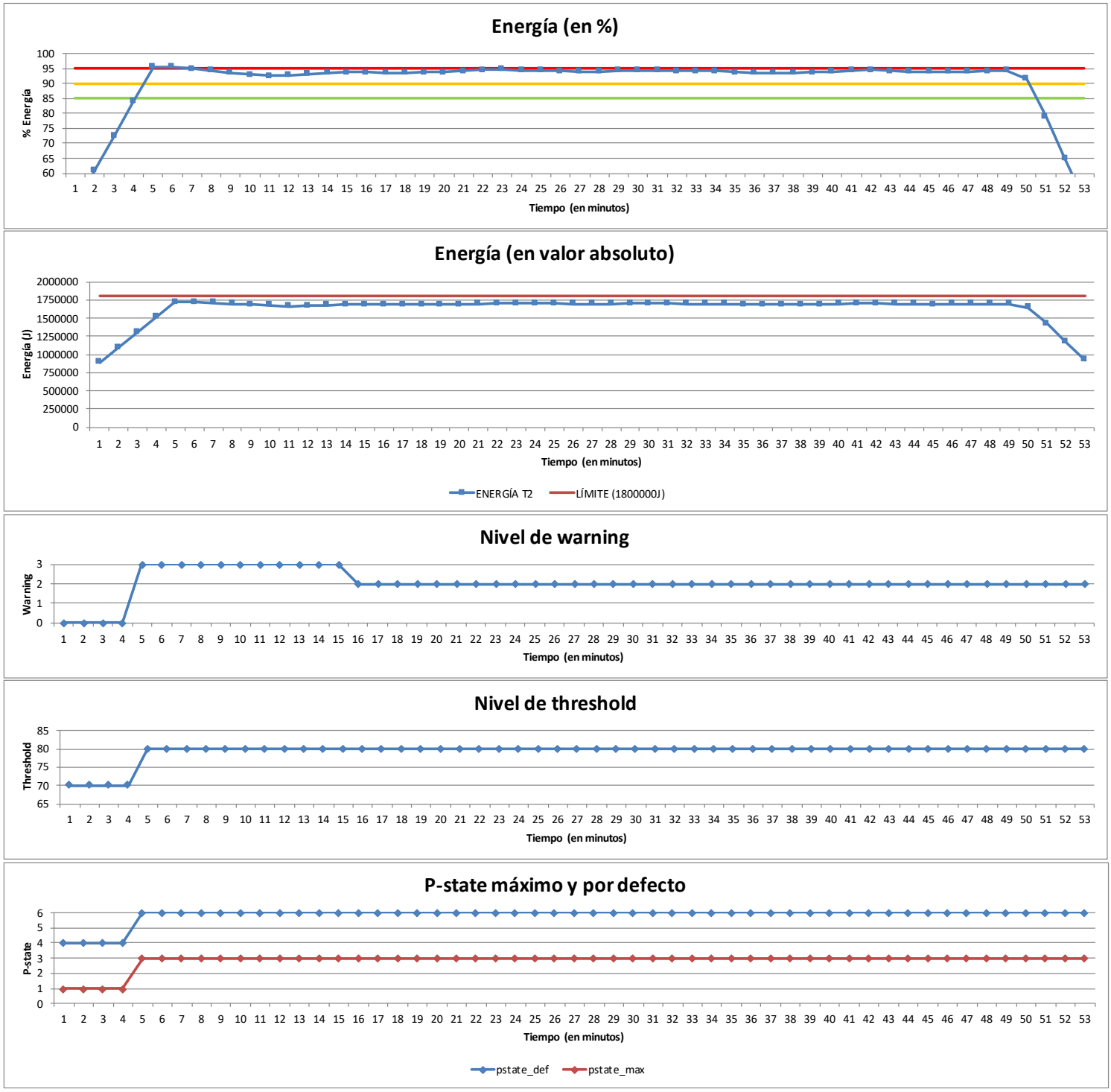


Figura E.8: Experimento 20.3 con optimización ramp up ajustado al nivel PANIC.

E.4. Exp. 20.4: Mix estático

En la figura E.9 se muestra el comportamiento del experimento ajustando al nivel NO PROBLEM.

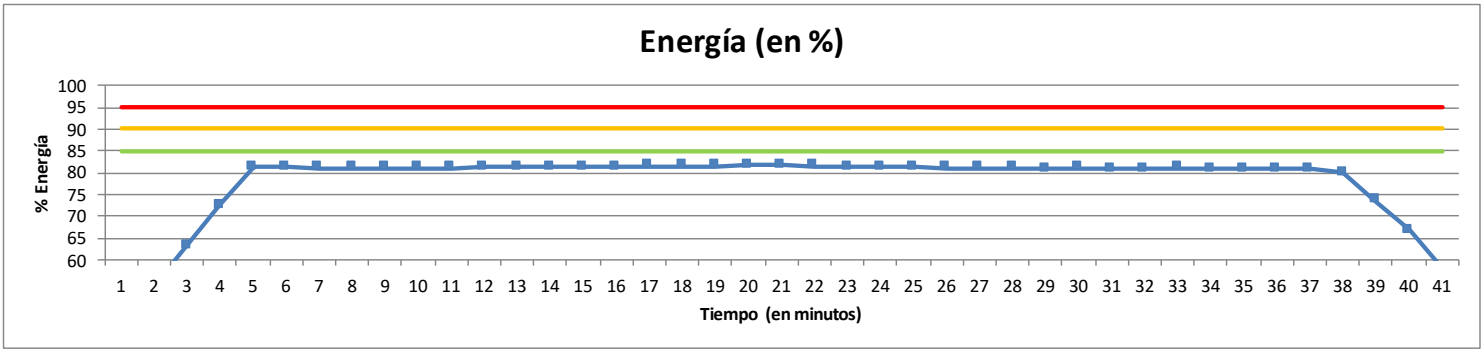


Figura E.9: Comportamiento del experimento 20.4 sin acciones del EARGM.

Optimización ramp up:

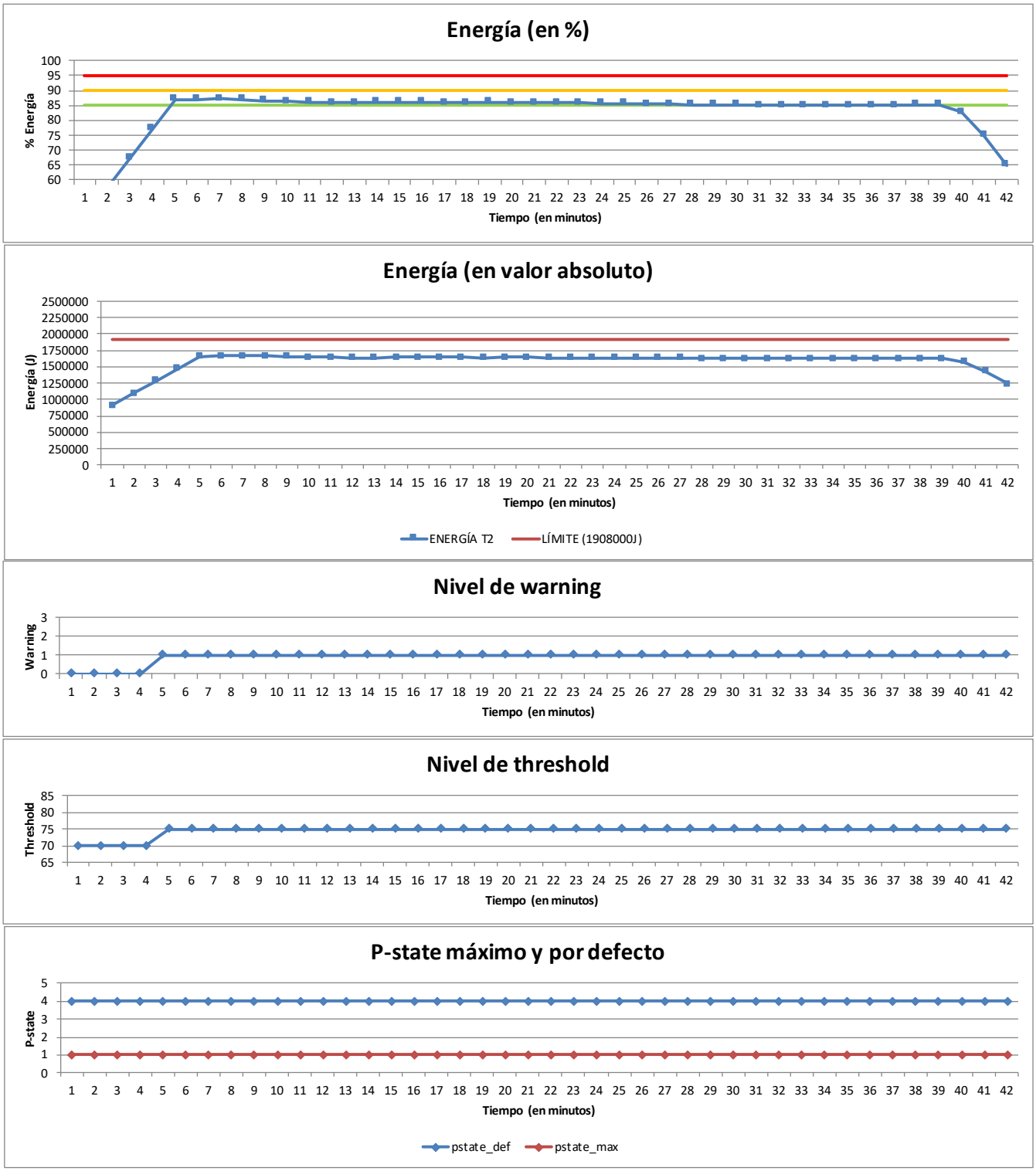


Figura E.10: Experimento 20.4 con optimización ramp up ajustado al nivel WARNING1.

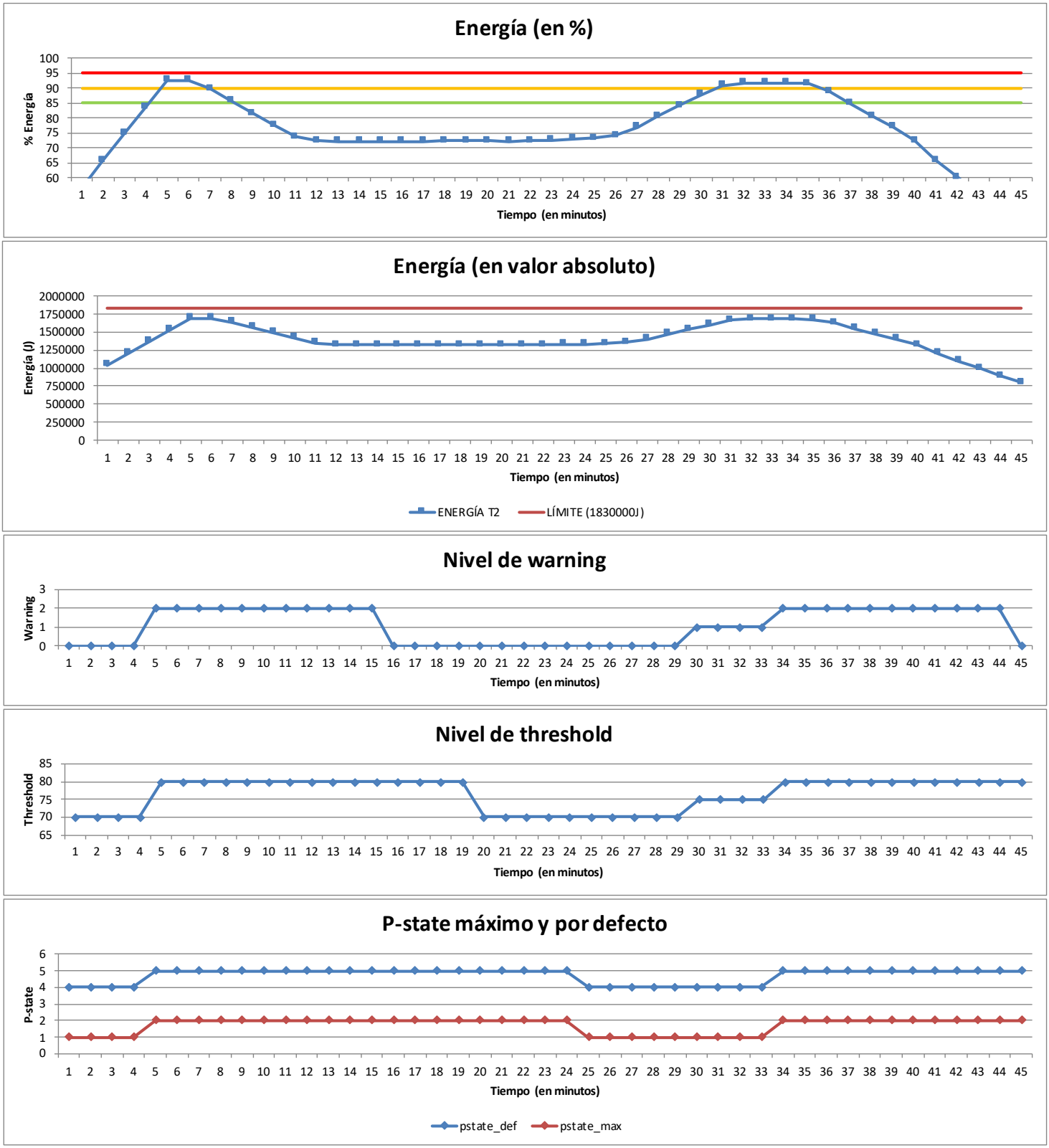


Figura E.11: Experimento 20.4 con optimización ramp up ajustado al nivel WARNING2.

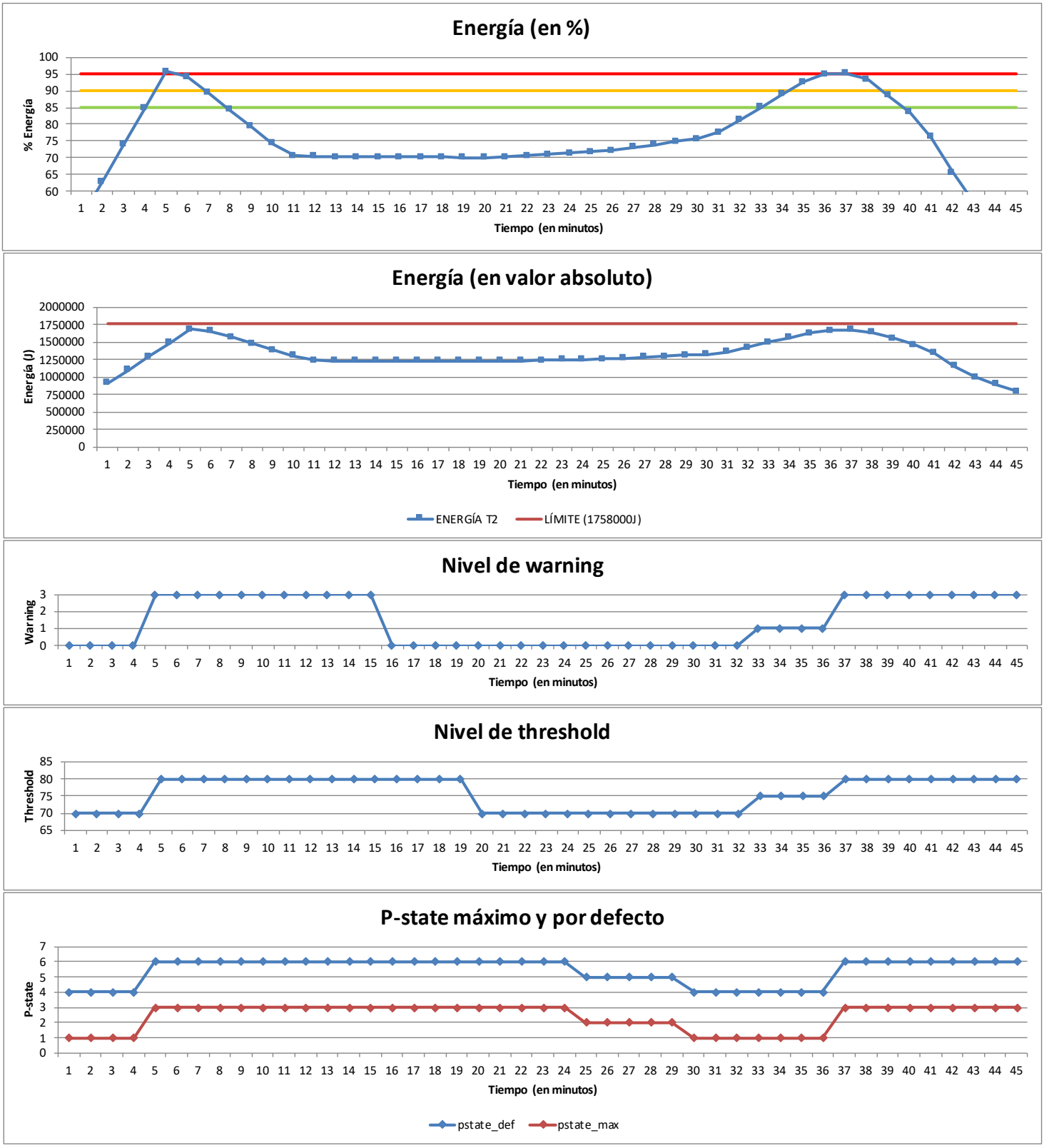


Figura E.12: Experimento 20.4 con optimización ramp up ajustado al nivel PANIC.

Optimización ramp down:

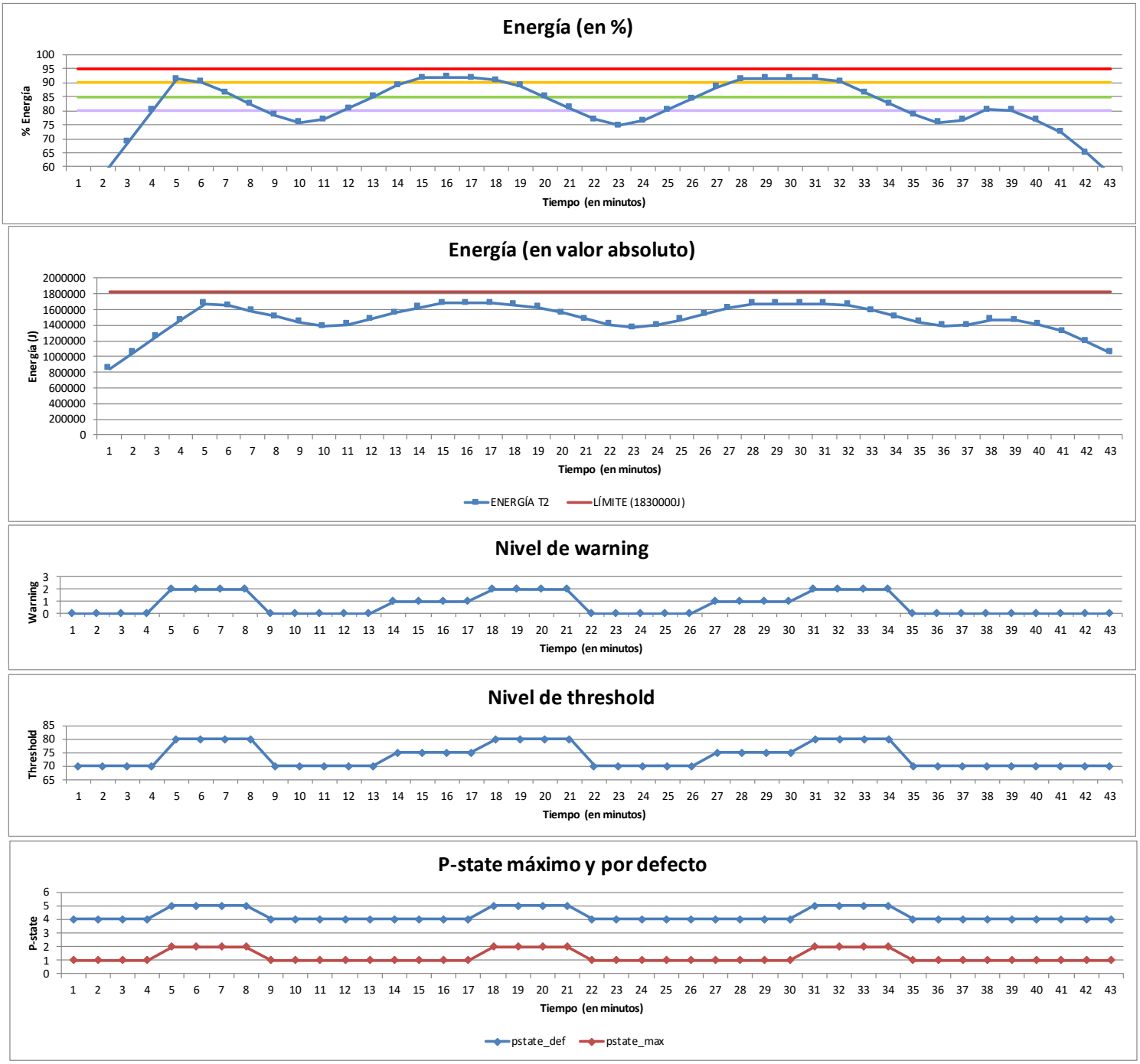


Figura E.13: Experimento 20.4 con optimización ramp down ajustado al nivel WARNING2.

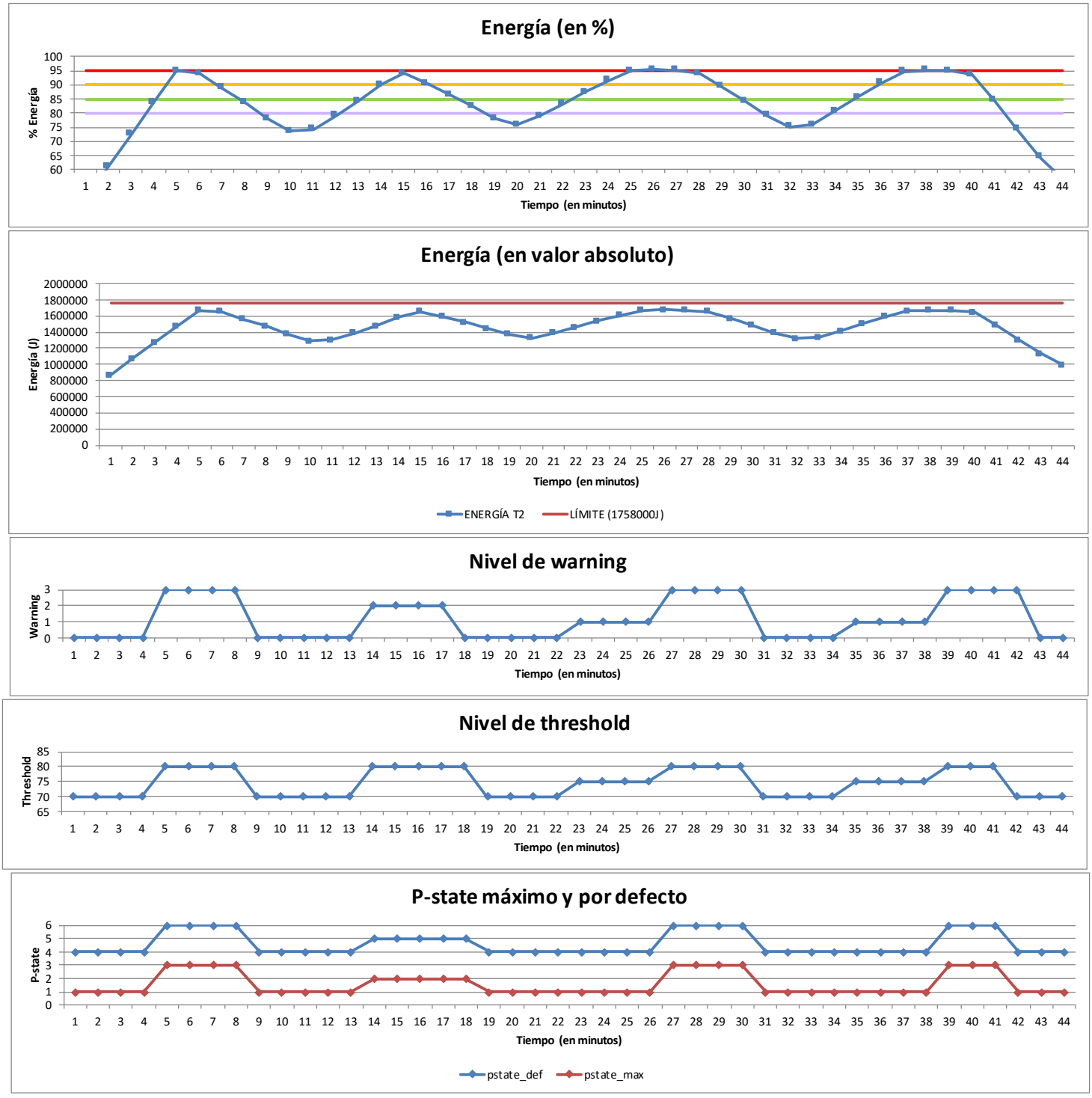


Figura E.14: Experimento 20.4 con optimización ramp down ajustado al nivel PANIC.

E.5. Exp. 20.5: Mix dinámico intensivo en CPU

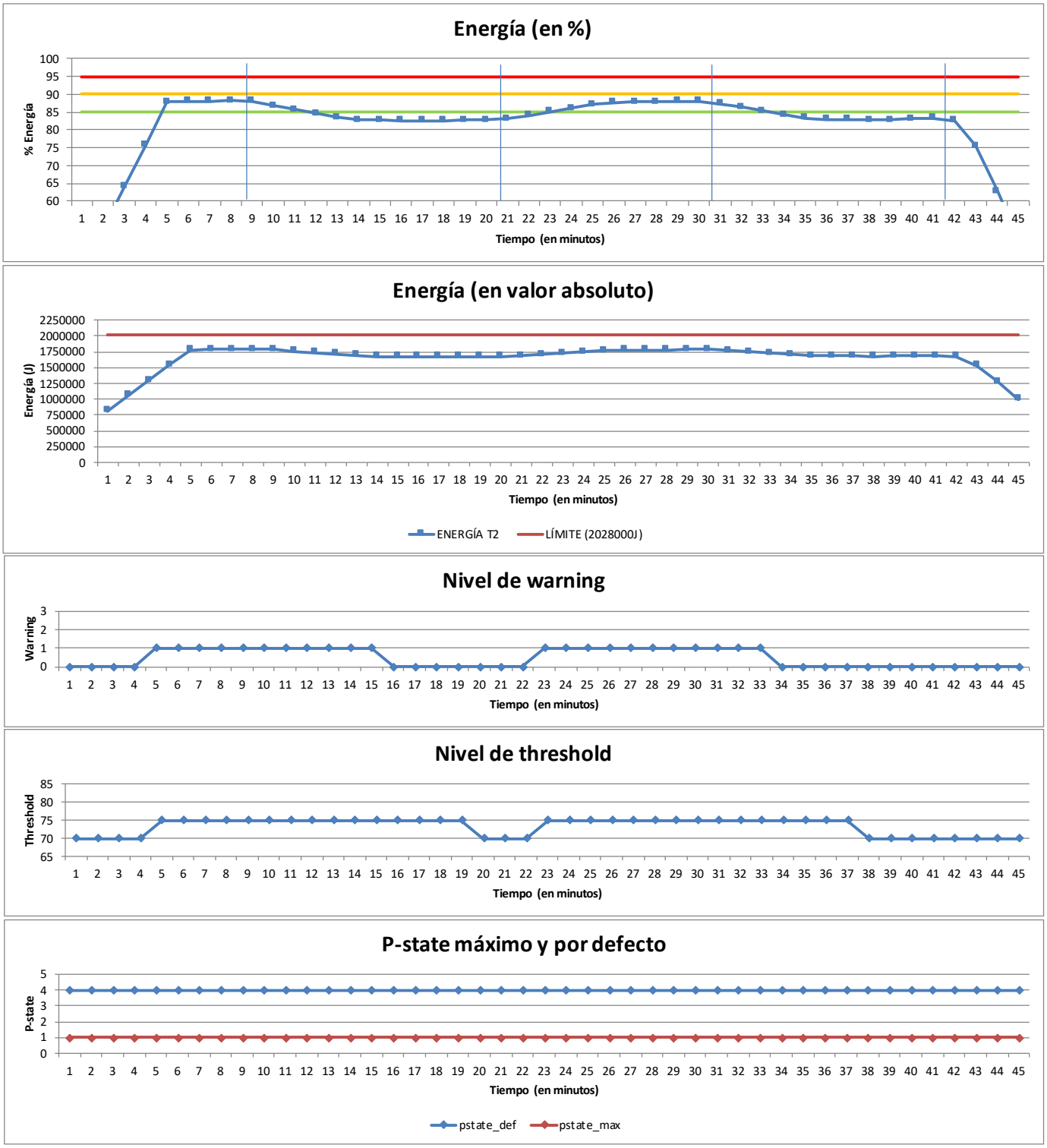


Figura E.15: Experimento 20.5 con optimización ramp up ajustado al nivel WARNING1.

E.6. Exp. 20.6: Mix dinámico intensivo en memoria

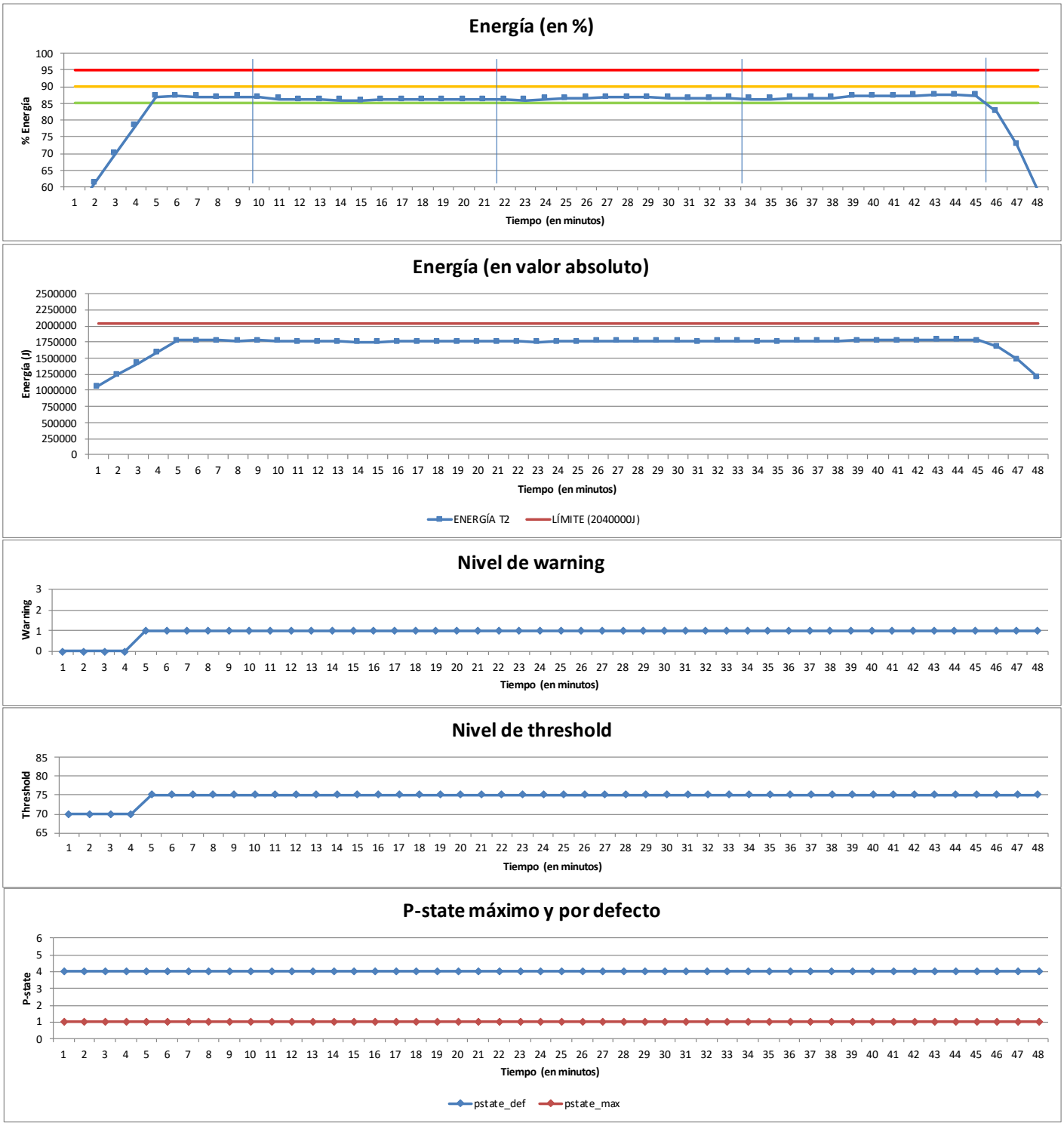


Figura E.16: Experimento 20.6 con optimización ramp up ajustado al nivel WARNING1.

E.7. Exp. 20.7: Mix dinámico con pérdida de carga

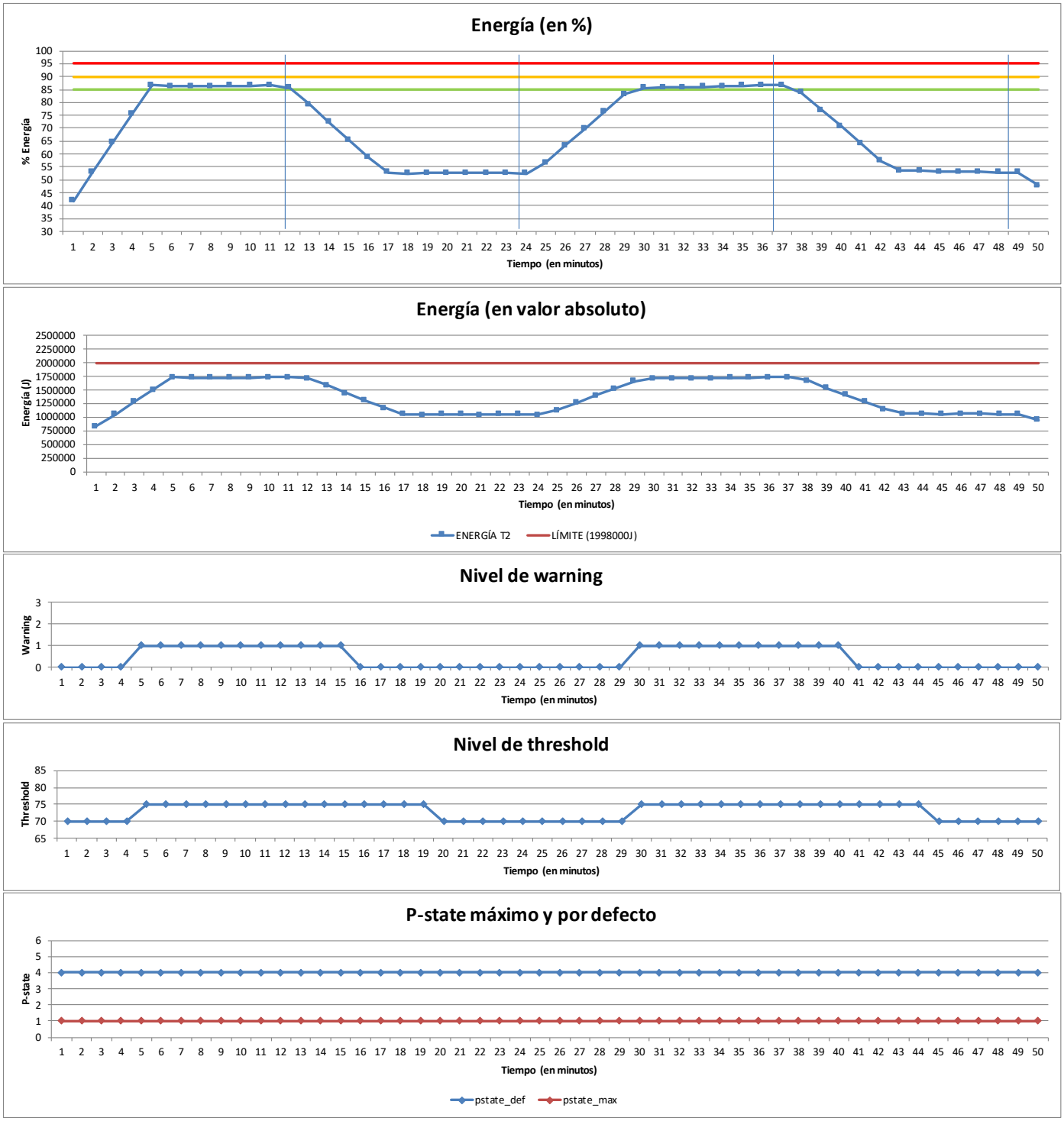


Figura E.17: Experimento 20.7 con optimización ramp up ajustado al nivel WARNING1.

Referencias

[1] Barcelona Supercomputing Center. *EAR Wiki documentation.* [Consulta: 22 Febrero 2020]. https://github.com/BarcelonaSupercomputingCenter/ear/wiki/3-%CB%97-Architecture.

[2] TOP500. [Consulta: 22 Febrero 2020]. https://www.top500.org/lists/2019/11/.

[3] Barcelona Supercomputing Center. *EAR: Energy management framework for HPC.* [Consulta: 22 Febrero 2020]. https://www.bsc.es/research-and-development/software-and-apps/software-list/ear-energy-management-framework-hpc/documentation.

[4] Corbalán, J.; Brochard, L. *EAR: Energy management framework for supercomputers.* IPDPS Conference, 2019. https://www.bsc.es/sites/default/files/public/bscw2/content/software-app/technical-documentation/ear.pdf.

[5] Corbalán, J. *EAR: Energy management for supercomputers.* The HPC PowerStack Seminar, Junio 2019. https://www.bsc.es/sites/default/files/public/bscw2/content/software-app/technical-documentation/ear_powerstack_seminar.pdf.

[6] Barcelona Supercomputing Center; Lenovo; Leibniz Supercomputing Center. *EAR is an Energy Management Framework.* SC'19: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2019. https://www.bsc.es/sites/default/files/public/bscw2/content/software-app/technical-documentation/111319_sc19_ear_flyer_1.pdf.

[7] Leibniz Supercomputing Center. *SuperMUC-NG.* [Consulta: 22 Febrero 2020]. https://doku.lrz.de/display/PUBLIC/SuperMUC-NG.

[8] Lenovo. *Lenovo XClarity Controller (XCC) Support on ThinkSystem Servers.* https://lenovopress.com/lp0880.pdf.

[9] Maiterth, M. (Intel). *Global Extensible Open Power Manager.* European HPC Summit Week, Mayo 2018. https://www.etp4hpc.eu/pujades/files/2018-05-30_EEWG_Maiterth.PDF.

[10] Simon, M. (Atos). *HPC, AI and Quantum - Exascale*. HPC-AI Advisory Council - Perth Conference, Agosto 2019. https://www.hpcadvisorycouncil.com/events/2019/australia-conference/pdf/day-two/M_Simon_Atos_Exascale_Wed_280819.pdf.

[11] Martin, S.; Rush, D.; Kappel, M. *Cray Advanced Platform Monitoring and Control (CAPMC)*. CUG 2015: Proceedings of the 2015 Cray User Group, Abril 2015. https://cug.org/proceedings/cug2015_proceedings/includes/files/pap132.pdf.

[12] SLURM Workload Manager. *SLURM Power Management Guide*. [Consulta: 24 Febrero 2020]. https://slurm.schedmd.com/power_mgmt.html.

[13] Kent Beck, et al. *Principios del Manifiesto Ágil*. 2001. [Consulta: 20 Febrero 2020]. https://agilemanifesto.org/iso/es/principles.html.

[14] InfoJobs y ESADE Business School. *Informe sobre el estado del mercado laboral en España*. Mayo 2019. https://nosotros.infojobs.net/wp-content/uploads/2019/05/Informe_Mercado_Laboral_InfoJobs_ESADE_2018.pdf.

[15] Brochard, L.; Kamath V.; Corbalán, J.; Holland, S.; Mittelbach, W.; Ott, M. *Energy-Efficient Computing and Data Centers*. Wiley-ISTE Ltd, 2019. ISBN: 9781786301857 DOI: 10.1002/9781119422037.

[16] Lim, M.Y.; Freeh, V.M.; Lowenthal, D.K. *Adaptive, Transparent Frequency and Voltage Scaling of Communication Phases in MPI Programs*. SC'06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. 2006. http://dkl.cs.arizona.edu/publications/papers/sc06.pdf.

[17] Equipo técnico de EAR (Barcelona Supercomputing Center). *EAR 3.2 documentation*. https://www.bsc.es/sites/default/files/public/bscw2/content/software-app/technical-documentation/ear3.2.pdf.

[18] Corbalán, J.; Alonso, L.; Aneas, J. *Energy optimization with EAR*. https://www.bsc.es/sites/default/files/public/bscw2/content/research-line/documentation/ear_eval_june_2020.pdf.

[19] Leibniz-Rechenzentrum (LRZ). *Documentation: Energy Aware Runtime*. [Consulta: 25 Mayo 2020]. https://doku.lrz.de/display/PUBLIC/Energy+Aware+Runtime.

[20] University of Sussex. *Environment Modules - HPC User Guide*. [Consulta: 25 Abril 2020]. https://info.hpc.sussex.ac.uk/hpc-guide/software/environment-modules.html.

[21] NASA Advanced Supercomputing Division. *NAS Parallel Benchmarks*. [Consulta: 26 Abril 2020]. https://www.nas.nasa.gov/publications/npb.html.

[22] Bailey, D.; Barszcz, E.; Dagum L; Simon, H. *NAS Parallel Benchmark Results 10-93*. RNR Technical Report RNR-93-016, Octubre 1993. https://www.nas.nasa.gov/assets/pdf/techreports/1993/rnr-93-016.pdf.

[23] Bailey, D.; Barszcz, E.; Barton, J.; Browning, D.; Carter, R.; Dagum, L.; Fatoohi, R.; Fineberg, S.; Frederickson, P.; Lasinski, T.; Schreiber, R.; Simon, H; Venkatakrishnan, V; Weeratunga, S. *The NAS Parallel Benchmarks*. RNR Technical Report RNR-94-007, Marzo 1994. `https://www.nas.nasa.gov/assets/pdf/techreports/1994/rnr-94-007.pdf`.

[24] Feng, H; Van der Wijngaart, R; Biswas, R; Mavriplis, C. *Unstructured Adaptative (UA) NAS Parallel Benchmark, Version 1.0*. NASA Technical Report NAS-04-006, Julio 2004. `https://www.nas.nasa.gov/assets/pdf/techreports/2004/nas-04-006.pdf`.

[25] Intel Developer Zone. *Intel Math Kernel Library*. [Consulta: 27 Abril 2020]. `https://software.intel.com/en-us/mkl/choose-download`.

[26] McCalpin, J.D. *STREAM: Sustainable Memory Bandwidth in High Performance Computers*. [Consulta: 2 Junio 2020]. `http://www.cs.virginia.edu/stream/ref.html`.

[27] Joos, M. *DUMSES Hybrid*. [Consulta: 2 Junio 2020]. `https://github.com/marcjoos-phd/dumses-hybrid`.

[28] SLURM Workload Manager. *Frequently Asked Questions*. [Consulta: 10 Junio 2020]. `https://slurm.schedmd.com/faq.html#comp`.