Facultad de informática de Barcelona
Grado en ingeniería informática
Especialidad : Computación

Trabajo final de grado

Instrumentación de OpenACC en Extrae

Autor: Ali Muhammad Shiekh

Directora: Judit Gimenez Lucas
Ponente : Jesús Labarta Mancho [AC]

30 de Junio del 2020

En colaboración con Barcelona Supercomputing Center - Centro Nacional de Supercomputación.
Performance Tools Group, Departamento de Ciencias de la Computación

Memoria del trabajo final de grado

A mis padres, por creer en mí

Agradecimientos

Al grupo de performance tools del BSC-CNS. Con especial mención a German Llort.

Abstract

BSC-CNS has developed and provides their own performance analysis tools, Extrae, Paraver and Dimemes for an HPC environment.

As of today Extrae provides no support for the parallel programing model OpenACC.

The aim of this project is to develop a new functionality for the Extrae tool that allows it to register the activity of the OpenACC model. And furthermore enable the performance analysis of OpenACC applications.

Resumen

BSC-CNS es el centro de supercomputación nacional que dispone de sus propias herramientas Extrae, Paraver y Dimemas para hacer análisis de rendimiento en entornos HPC.

Actualmente, Extrae no da soporte para el modelo de programación paralela OpenACC.

El objetivo de este proyecto es incorporar una nueva funcionalidad a la herramienta Extrae, para poder registrar la actividad de OpenACC, con la finalidad de usarse para hacer el análisis de rendimiento de aplicaciones OpenACC.

Resum

El BSC-CNS és el centre de supercomputació nacional que disposa de les seves pròpies eines Extrae, Paraver i Dimemas per fer l'anàlisis de rendiment en entorns HPC.

Actualment, l'eina Extrae no dóna suport per al model de programació paral·lela OpenACC.

L'objectiu d'aquest projecte és incorporar una nova funcionalitat a l'eina Extrae, per poder registrar l'activitat de OpenACC, amb la finalitat d'utilitzar-lo per fer l'anàlisis de rendiment d'aplicacions OpenACC.

Índice de contenidos

Listado de figuras

Listado de tablas

Listado de códigos

Acrónimos

BSC-CNS : Barcelona Supercomputing Center - Centro Nacional de Supercomputación

HPC: High Performance Computing

MN4 : MareNostrum 4

CTE-POWER : Cluster de Tecnologías Emergentes

1 Contexto del proyecto

1.1 Marco del proyecto

Este proyecto es un trabajo final de grado (TFG), del Grado de Ingeniería Informática (GEI) de la Facultad de Informática de Barcelona (FIB). Ha sido desarrollado en colaboración con el Barcelona Supercomputing Center - Centro Nacional de Supercomputación (BSC-CNS).

BSC-CNS es el centro nacional de supercomputación en España. Está especializado en informática de alto rendimiento (HPC). Gestiona el supercomputador MareNostrum, que es uno de los supercomputadores más potentes de Europa.

En el BSC trabajan más de 500 expertos y profesionales en I+D, y las líneas de investigación se centran en cuatro áreas: ciencias de la computación, ciencias de la vida, ciencias de la tierra y aplicaciones de la computación en ciencia e ingeniería en.

Uno de los clúster HPC de los que dispone el BSC es el CTE-POWER (Cluster de Tecnologías Emergentes) [11]. Para realizar este proyecto se usará CTE-POWER.

En los superordenadores se ejecutan programas informáticos. De estos programas informáticos necesitamos conocer si se hace un uso eficiente de los recursos. Es por eso que el centro tiene su propio grupo de análisis de rendimiento. Este grupo se dedica al desarrollo de aplicaciones informáticas para hacer el análisis de rendimiento en entornos HPC, y también hace investigación en este campo.

El grupo de performance tools pertenece al departamento de ciencias de la computación del BSC-CNS. El grupo tiene como herramientas principales para el análisis de rendimiento: Extrae, Paraver y Dimemas [9]

1.2 Conceptos clave y herramientas

Para poder seguir este proyecto es clave entender los siguientes conceptos y herramientas.

1.2.1 Extrae

Extrae [4] es una herramienta del BSC-CNS capaz de capturar información para hacer el análisis de rendimiento en tiempo de ejecución y de generar una traza detallada de su actividad. Es capaz de hacerlo para varios modelos de programación paralela.

1.2.2 Paraver

Paraver es una herramienta **VISUAL** que muestra la información sobre análisis de rendimiento que se obtiene habiendo usado Extrae previamente, aunque no necesariamente. Es decir, podemos obtener trazas para Paraver habiendo usado otras herramientas similares a Extrae. [8]. Paraver es una herramienta del BSC-CNS.

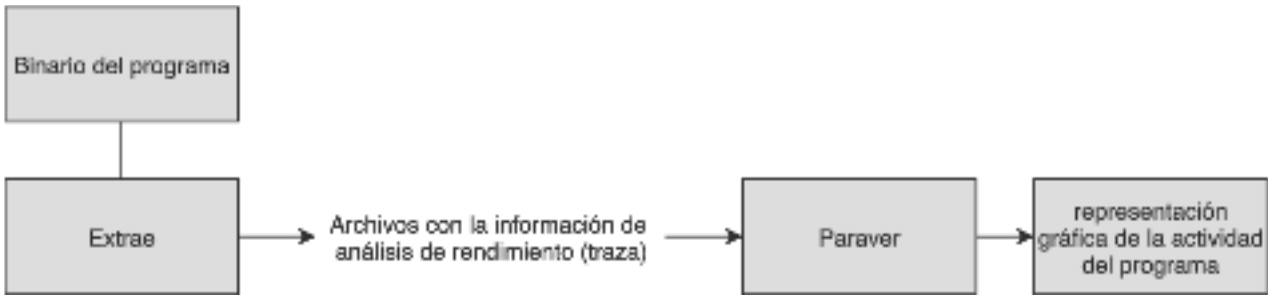

Figura 1: Relación entre Extrae y Paraver

Relación entre Extrae y Paraver

El binario del programa se ejecuta. En tiempo de ejecución, la herramienta Extrae registra la actividad del programa. Como resultado obtenemos ficheros que contienen información de análisis de rendimiento. En este caso, para formato Paraver.

Paraver es capaz de leer estos ficheros, interpretarlos y mostrar de forma visual/gráfica, la actividad que se ha registrado en tiempo de ejecución de nuestro programa.

1.2.3 OpenACC

OpenACC es un modelo de programación paralela creado en 2012 [5].Fue desarrollado por Cray, CAPS, Nvidia y PGI. Es un modelo de programación paralela basado en directivas (#pragma acc ...).

OpenACC está pensado para simplificar la programación paralela en sistema heterogéneos tipo CPU/GPU. El modelo está pensado para aprovechar la unidades de cálculo de nuestra GPU para computar de forma paralela. El compilador de OpenACC interpreta las directivas (#pragma acc …), y paraleliza el código en consecuencia.

Modelo de ejecución

En OpenACC tenemos dos tipos de dispositivos diferentes. Estos pueden ser CPU o GPU(aceleradores). En el modelo de ejecución tenemos dos tipo de roles. El rol de host y el rol de device. En el mínimo nivel de paralelización tenemos threads (hilos) que harán el papel de host o de device exclusivamente. Típicamente, un thread de CPU hará el papel de host. El thread de alguna GPU, si la hay, hará el papel de Device.

La ejecución en el modelo OpenACC es host-directed. Esto quiere decir que el thread que haga el papel de host tiene la lógica de la ejecución, y es el que controla el flujo de la ejecución. El thread que hace de device llevará a cabo las tareas que hayan sido mandadas por el host.

Puede haber más de un thread que haga de host si usamos, por ejemplo, OpenMP [1](que se puede usar junto con el modelo OpenACC).

Las CPUs pueden ser multicore. En este caso, el mismo dispositivo CPU puede tener la capacidad de tener threads que jueguen el rol de host o de device exclusivamente. En este caso, aún no habiendo ningún acelerador, las restricciones en el modelo de ejecución de OpenACC se deben cumplir, pues estas vienen determinadas según el papel (host/device) que juega cada thread durante la ejecución.

Modelo de memoria

En general, en OpenACC el host thread (típicamente un hilo de la CPU) tiene su memoria, y el device thread (típicamente de la GPU) tiene su memoria. Es decir, no comparten memoria entre los dos tipos de dispositivos, ya que cada uno tiene la suya.

[1] "OpenMP: Home." https://www.openmp.org/. : Modelo de programación paralela

La transferencia de datos entre CPU y GPU se hace a través de sus respectivas memorias. Es decir, si la GPU necesita una variable de la CPU para hacer la computación, entonces se copia el dato desde la memoria de la CPU a la memoria de la GPU.

En el caso inverso, es decir, para pasar la variable que contiene el resultado de la computación, desde la GPU hacía la CPU, se copia el dato de la memoria de la GPU a la memoria de la CPU.

En OpenACC la transferencia de datos entre CPU y GPU, suele ser crucial ya que afecta directamente al rendimiento de los programas.

1.3 Problema a resolver y actores implicados

Definidos los conceptos y la herramientas, ya estamos preparados para entender la problemática que pretende resolver este proyecto.

Problemática

La problemática está en el hecho de que no podemos usar la herramienta Extrae para hacer el análisis de rendimiento de aplicaciones que usen el modelo de programación paralela OpenACC.

Es por eso que se requiere incorporar el soporte del modelo OpenACC a la herramienta Extrae.

En definitiva, todo el proyecto trata de incorporar una nueva funcionalidad a la herramienta Extrae del BSC-CNS.

Actores implicados

El proyecto está desarrollado y testeado en el clúster CTE-POWER [11] del BSC.

Por el momento, el proyecto está enfocado para que sea usado por las personas del entorno del BSC-CNS. Como requisito estas personas deben tener acceso al clúster CTE-POWER del BSC.

En consecuencia, los requerimientos funcionales están pensados para cubrir las necesidades de las personas del entorno del BSC-CNS.

Las personas del grupo performance tools del BSC-CNS, me asesoran y guiarán para hacer este proyecto. Además, también me darán directrices sobre cómo hacer el proyecto.

1.4 Justificación de la solución y del proyecto

Hay herramientas específicas para hacer el análisis de rendimiento de aplicaciones OpenACC en el mercado. Es decir, la implementación de OpenACC en Extrae, no sería la única opción.

Soluciones actuales al problema

En primer lugar, tenemos Score-P,[2] a diferencia de Extrae, hace la instrumentacion de OpenACC. Score-P hace la instrumentación de OpenACC a nivel de compilador.

En segundo lugar, tenemos la herramienta nvprof[3] de Nvidia. Podemos usar nvprof para extraer información de análisis de rendimiento de una aplicación OpenACC. Una vez extraída la información, podemos usar el nvidia visual profiler (nvvp) para tener un representación gràfica del análisis de rendimiento de la aplicación.

Estas herramientas son suficientes para satisfacer todas nuestras necesidades. La diferencia está en que Extrae no da soporte para registrar la actividad de OpenACC. Y eso es exactamente lo que se pretende conseguir.

Justificación de mi solución

A nivel estratégico, es interesante que el personal del centro use las herramientas Extrae y Paraver, que son open source y que del mantenimiento nos encargamos nosotros (personal del propio centro, en concreto, el grupo de performance tools). Además, las funcionalidades que implementaremos irán en función de las necesidades de las personas que usan OpenACC en el entorno del BSC-CNS.

Usar herramientas que son el propio BSC (Extrae/Paraver) nos da el poder y el control de decidir de qué queremos hacer el análisis de rendimiento, cuándo lo queremos hacer y cómo lo queremos hacer. Este control no lo tenemos si usamos herramientas de terceros. Por lo tanto, se justifica la utilidad y necesidad de este proyecto.

[2] "Projects :: Score-P - VI-HPS." https://www.vi-hps.org/projects/score-p.: Herramienta similar a Extrae que hace la instrumentación de OpenACC a nivel de compilador.
[3] "nvprof" , http://docs.nvidia.com/cuda/profiler-users-guide/index.html. Herramienta para hacer profiling propiedad de la empresa Nvidia

2 Alcance del proyecto

En este capítulo definiremos el alcance del proyecto, es decir, detallaremos los objetivos y subobjetivos del proyecto, las funcionalidades que queremos implementar, comentaremos algunos riesgos que podemos tener y definiremos la metodología de trabajo y validación del proyecto.

2.1 Objetivos y subobjetivos

El objetivo principal del proyecto es hacer que la herramienta Extrae de soporte para hacer el análisis de rendimiento de aplicaciones escritas usando OpenACC (instrumentación de OpenACC en Extrae).

El primer subobjetivo del proyecto sería conseguir hacer la implementación del estándar más alta de OpenACC que seamos capaz de verificar. En este caso es el 3.0.

El segundo subobjetivo es que podamos usar cualquier compilador de OpenACC, pero usar la misma instrumentación de OpenACC en Extrae.

Para cumplir el objetivo principal usaremos la Profiling API del estándar de OpenACC en su versión 3.0, la última hasta la fecha 23 Junio [12]

2.2 Requerimientos funcionales

Lo que pretendemos conseguir es analizar el comportamiento de aplicaciones OpenACC.

Para ello, debemos implementar en Extrae el registro para la actividad del host y del device para aplicaciones OpenACC. El registro de la actividad quedará almacenado en los ficheros de salida de Extrae.

Los ficheros de salida de Extrae (trazas), se cargan usando Paraver, con la finalidad de ver la actividad del host y del device.

Así pues, las funcionalidades que debemos implementar son, por un lado, registrar la actividad del host, y por otro lado, registrar la actividad del device. Para implementar estas funcionalidades vamos a usar la profiling API del estándar 3.0 de OpenACC[12], sección 5 de la especificación.

2.3 Posibles riesgos

A continuación, detallaremos los posibles riesgos que podemos tener durante el desarrollo de este proyecto.

El primer riesgo que podemos tener es que la profiling API de OpenACC 3.0 [12]no sea realmente útil, ya que puede ocurrir el hecho de que no esté lo suficientemente desarrollada como para usarla para implementar las funcionalidades que nosotros queremos.

Otro riesgo que podemos tener es el uso de código informático sin tener en cuenta la eficiencia. El riesgo que corremos es que el eventual análisis de rendimiento quede invalidado por tener un overhead [4] no admisible durante el análisis de rendimiento.

2.4 Metodología de trabajo y validación

En esta sección detallaremos la metodología de trabajo y validación del proyecto.

2.4.1 Metodología de trabajo

Se ha elegido la metodología de trabajo Agile. El entorno de trabajo es grande y dinámico, por ese motivo, en muchas ocasiones, no da tiempo a adquirir todos los conocimiento, para poder acabar un paso complemente, y avanzar al siguiente. Lo normal es acabar un paso, avanzar al siguiente, y entonces se evalúa si volver al paso anterior para mejorarlo o no.

El proyecto está planificado en tres sprints. En cada sprint se hace una actividad que permite hacer el siguiente sprint. Después de cada sprint se valida si está bien hecho, o si hay que cambiar/revisar algún paso de dicho sprint.

En el primer sprint se creará una librería dinámica[5] para capturar el registro de la actividad de un aplicación OpenACC. Esta librería se incorporará en Extrae en el segundo sprint.

En el segundo sprint se incorporará esta librería en Extrae y se implementará todo lo relativo al registro de la actividad del host.

En el tercer sprint se hará lo mismo que el segundo sprint, pero para registrar la actividad del device.

[4] "Overhead" : En este caso, se entiende por overhead, el tiempo de más añadido al tiempo de ejecución de una aplicación, por el hecho de hacer análisis de rendimiento de la misma.
[5] "Librerías dinámicas" : Se enlazan al ejecutar, el sistema operativo debe encontrarlas al ejecutar el programa

Las tareas en concreto que se harán en cada sprint, están especificadas en la sección 7.1.4

2.4.2 Metodología de validación

Se usará la herramienta Paraver, como herramienta para verificar el desarrollo de la instrumentación de OpenACC en Extrae.

Los resultados de la validación del sprint uno, se mostraran con los resultados de la validación del sprint dos, pues la librería dinámica será incorporada en Extrae en el sprint dos.

Esencialmente, como resultado del sprint uno y dos, se pretende validar que Extrae sea capaz de registrar correctamente la actividad del host, y de obtener trazas Paraver que contengan la actividad del host.

La validación será correcta en la medida en que seamos capaces de ver toda la actividad OpenACC del host usando Paraver.

Por otro lado, como resultado del sprint tres, se pretende validar que Extrae sea capaz de registrar correctamente la actividad del device. En concreto, se pretende registrar el tiempo, de forma aproximada, que tarda el device en hacer la computación, que ha sido puesta en cola por el host.

El sprint tres estará bien hecho en la medida en que seamos capaces de ver en Paraver que la actividad del device queda registrada.

Finalmente, se validará que la instrumentación de OpenACC en Extrae, provoque un overhead acceptable. Ya que un overhead cero en el análisis de rendimiento es inevitable.

3 Librería dinámica para registrar la actividad de OpenACC

En este capítulo explicaremos para qué y cómo hemos hecho la librería dinámica para capturar la actividad de OpenACC. También explicaremos el concepto técnico que hay detrás.

El mismo concepto técnico y la librería que se han desarrollado, forman parte y se usan para hacer la implementación de OpenACC en Extrae en el capítulo 4..

3.1 Concepto técnico para registar la actividad de OpenACC

En el momento en que se compila una aplicación OpenACC y ésta se ejecuta a posteriori, en el runtime environment del sistema operativo se ejecutan unas funciones que hacen que la aplicación funcione y se pueda ejecutar.

Algunas de estas funciones son propias de nuestro código y tenemos control sobre ellas. Otras funciones son ajenas a nuestro código, aunque igual de necesarias para ejecutarlo.

La idea principal consiste en usar el mecanismo LD_PRELOAD para interponerse en las funciones propias del runtime de OpenACC, con la finalidad de registrar la actividad de OpenACC.

3.1.1 El mecanismo LD_PRELOAD

El mecanismos LD_PRELOAD [3] es un mecanismo del sistema operativo Linux que se usa para dar prioridad a las librerías.

En general, podemos escribir una librería dinámica que implemente las funciones malloc(argumentos) y free(argumentos). De esta forma, cuando carguemos nuestra librería dinámica usando el mecanismo LD_PRELOAD, lo que se va a ejecutar realmente son nuestras definiciones de malloc(argumentos) y free(argumentos), en vez de las “reales”.

Usando la misma idea podemos implementar las funciones que se ejecutan durante el runtime ,relativas a OpenACC, para registrar la actividad de OpenACC.

En concreto, lo que pretendemos conseguir es tener una librería dinámica que se cargue antes que el binario de nuestra aplicación OpenACC usando el mecanismo LD_PRELOAD. De esta forma, nos interponemos con la finalidad de poder capturar la actividad de OpenACC.

3.2 Profiling API de OpenACC

Hay dos formas de interponerse con la finalidad de registrar la actividad de OpenACC.

La primera opción es hacerlo a nivel de compilador. Es decir, implementamos las funciones del compilador que se ejecutan durante la runtime, relativas a OpenACC, y nos interponemos con LD_PRELOAD. Esta es una alternativa que no hemos escogido.

La otra forma de registrar la actividad de OpenACC es usando la Profling API de OpenACC . Nosotros lo vamos a hacer de esta forma.

En la especificación oficial de OpenACC [12] (sección 5). Esto es, un conjunto de funciones que nos dan información relativa para hacer profiling de OpenACC. La implementación de la profiling API de OpenACC la hace el compilador de OpenACC.

En el runtime se ejecuta la función void acc_register_library(acc_prof_reg, acc_prof_reg, acc_prof_lookup_func). Para registrar la actividad de OpenACC usando la profiling API, se debe implementar dicha función en nuestra librería dinámica.

En el momento de cargar la librería dinámica usando el mecanismo LD_PRELOAD, junto con el binario de la aplicación OpenACC, lo que sucederá es que se ejecutara nuestra implementación de la función acc_register_library(argumentos).

Podemos orientar la implementación de acc_register_library(argumentos) para registrar la actividad de OpenACC.

Para poder registrar la actividad de OpenACC usando la profiling API, se deben entender los elementos de los que disponemos para hacerlo. Estos son, esencialmente, las definiones del fichero OpenACC.

Cada vez que se ejecuta una aplicación OpenACC saltan algunos eventos de OpenACC. El hecho de que se produzca un evento de OpenACC, es debido que nuestra aplicación ha ejecutado alguna operación relacionada con OpenACC. Es decir, los eventos son indicadores de la actividad de OpenACC. Capturarlos implica estar registrando la actividad de OpenACC. La semántica (el motivo por el que son capturados), es dependiente del programador de la librería dinámica.

El fichero acc_prof.h del compilador debe implementar las definiciones para hacer el profiling de OpenACC. En concreto, en el fichero acc_prof.h están las definiciones (funciones y eventos) para registrar la actividad de OpenACC.

3.2.1 Estructura de callbacks de OpenACC

A continuación, hay una figura que muestra la relación entre diferentes llamadas y argumentos del acc_prof.h.

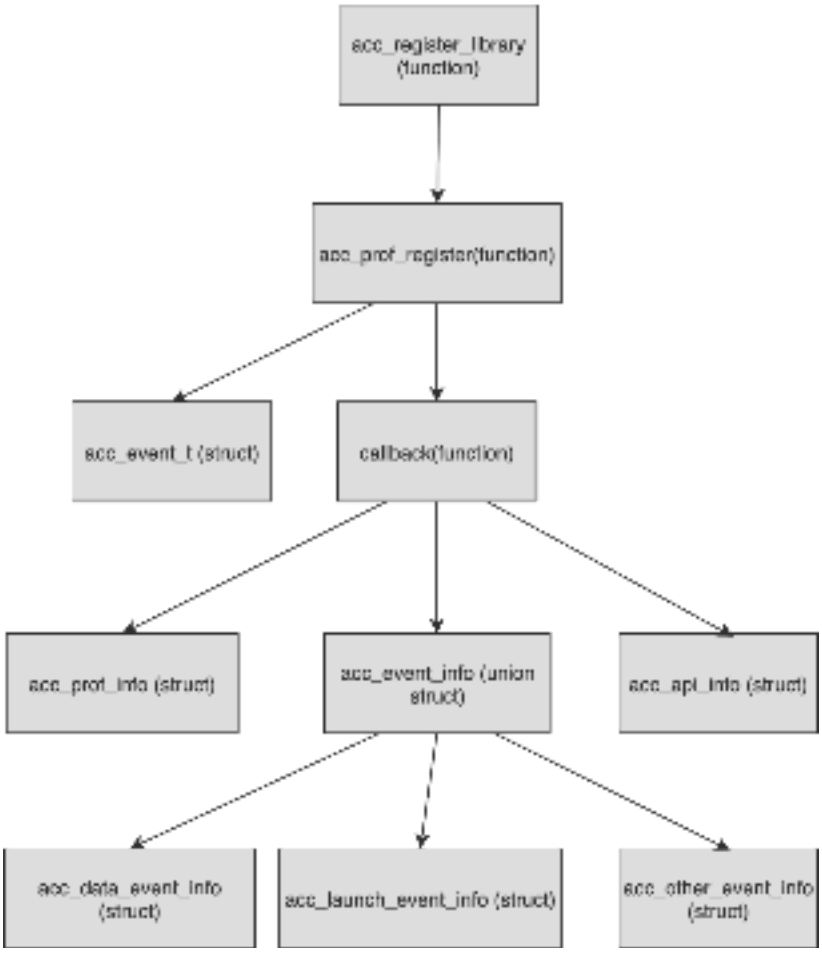

Figura 2: Esquema del fichero acc_prof.h

El esquema mostrado son esencialmente funciones y structs que contienen argumentos que son útiles, en su mayoría, para hacer el profiling de aplicaciones OpenACC. Este esquema se ha realizado usando [12].

En tiempo de ejecución se invoca la rutina acc_register_library. La rutina acc_register_library llama a la función acc_prof_register para cada evento, y este a su vez, ejecuta la función callback para dicho evento.

La función acc_register_library tiene como argumento la dirección de la función que registra cada evento con su callback. Es decir, tiene como argumento la dirección de la rutina acc_prof_register.

La función acc_prof_register tiene como argumentos el evento y la callback asociada al mismo. Puede haber tantas callback como queramos para cada evento, pero todas tienen la misma firma (argumentos). Los argumentos de las callbacks son acc_prof_info, acc_event_info y acc_api_info.

El struct acc_event_info es un “struct union” que bien puede ser acc_data_event_info (si es un evento de tipo de data), acc_launch_event_info (si es un evento de tipo de launch), o acc_other_event_info (si es un evento de otro tipo).

3.2.2 Eventos de OpenACC

Hay tres tipos de eventos en OpenACC. Eventos tipo launch, eventos tipo data y eventos que no son ni launch ni data (conocidos como: otro tipo de eventos).

Los eventos tipo launch se disparan cuando se pone en cola alguna computación que debe ser hecha por los device threads.

Los eventos tipo data se disparan cuando se crea, reserva, destruye y libera memoria. Los eventos tipo data también saltan cuando se ponen en cola las operaciones upload y download de datos entre el host y el device.

Los eventos que no son ni de launch ni de data son “el resto” (other type), saltan cuando hay una operación OpenACC que no tiene nada que ver con encolar una computación al device, o de un eventual movimiento de datos. Es decir, por ejemplo cuando se produce la inicialización o apagado de los devices y del runtime.

A continuación, tenemos una tabla que relaciona los eventos con su tipo.

Evento	Tipo
acc_ev_device_init_start	Other
acc_ev_device_init_end	Other
acc_ev_device_shutdown_start	Other
acc_ev_device_shutdown_end	Other
acc_ev_runtime_shutdown	Other
acc_ev_create	Data
acc_ev_delete	Data
acc_ev_alloc	Data
acc_ev_free	Data
acc_ev_enter_data_start	Other
acc_ev_enter_data_end	Other
acc_ev_exit_data_start	Other
acc_ev_exit_data_end	Other
acc_ev_update_start	Other
acc_ev_update_end	Other
acc_ev_compute_construct_start	Other
acc_ev_compute_construct_end	Other
acc_ev_enqueue_launch_start	Launch
acc_ev_enqueue_launch_end	Launch
acc_ev_enqueue_upload_start	Data
acc_ev_enqueue_upload_end	Data
acc_ev_enqueue_download_start	Data
acc_ev_enqueue_download_end	Data
acc_ev_wait_start	Other
acc_ev_wait_end	Other

Tabla 1: Relación entre evento OpenACC y su tipo

Podemos usar esta tabla para determinar qué tipo de operación OpenACC se está dando.La tabla ha sido hecha usando [12].

3.3 Librería dinámica externa para registrar la actividad de OpenACC

El estándar de OpenACC en la sección 5 define su profiling API [12] Si disponemos de un compilador que implemente el estándar , esencialmente también podremos usar el mismo compilador para implementar nuestra librería dinámica y personalizada, para registrar la actividad de OpenACC.

Hemos escogido el compilador PGI. Este compilador sigue fielmente el estándar de OpenACC para hacer la implementación del mismo. Es el compilador más desarrollado para OpenACC hasta la fecha (2020). Además, implementa la versión 3.0 de la profiling API [10].

En concreto, para hacer nuestra librería hemos usado el compilador PGI en su versión 20.1. La librería está hecha en lenguaje C. Usamos pgcc como compilador del lenguaje C de PGI.

En nuestro caso, lo único que nos interesaba era ser capaz de usar esta librería para detectar y capturar los eventos cuando se ejecuta un programa OpenACC.

Para hacer la librería generamos un archivo .c (p.e, myoaccprofiling.c). En el que hacemos un include del fichero acc_prof.h. El fichero acc_prof.h está en la carpeta include del compilador PGI.

```
#include <stdio.h>
#include <stdlib.h>
#include <acc_prof.h>

void prof_launch(acc_prof_info* profinfo, acc_event_info* eventinfo,acc_api_info* apiinfo) {
        acc_launch_event_info* launchinfo;
        launchinfo = (acc_launch_event_info*) eventinfo;
        switch(launchinfo->event_type){
               case acc_ev_enqueue_launch_start:
                 printf("i'am acc_ev_enqueue_launch_start\n");
                 break;
                 case acc_ev_enqueue_launch_end:
                 printf("i'am acc_ev_enqueue_launch_end\n");
                 break;
                 default:
                 break;
        }
}
void acc_register_library(acc_prof_reg acc_prof_register, acc_prof_reg acc_prof_unregister,
acc_prof_lookup lookup) {
      //launch event capture
        acc_prof_register(acc_ev_enqueue_launch_start,prof_launch,0);
        acc_prof_register(acc_ev_enqueue_launch_end,prof_launch,0);
}
```

Código 1: Ejemplo para capturar eventos tipo launch usando la librería dinámica de OpenACC

El código registra los eventos de OpenACC que se van produciendo a lo largo de la ejecución de un programa OpenACC. Este código es parte de nuestra librería y usa las definiciones del fichero acc_prof.h y el mecanismo LD_PRELOAD.

Hemos hecho tres callbacks, una para cada tipo de evento de OpenACC. En la línea `launchinfo = (acc_launch_event_info*) eventinfo;` estamos haciendo un cast, ya que sabemos que si se ha invocado este callback es porque se van a tratar eventos tipo launch. En la librería tenemos tres callbacks, una para cada tipo de evento OpenACC (launch, data y other).

Para compilar la librería y usarla con el mecanismo LD_PRELOAD hacemos lo siguiente:

```
//compilación de la librería dinámica de profiling
pgcc -fpic -c myoaccprofiling.c
pgcc -shared -o libmyoaccprofiling.so *.o
//ejecucion: a.out es el ejecutable de un programa OpenACC LD_PRELOAD=/libmyoaccprofiling.so ./a.out
```

Código 2: Compilación y ejecución de la librería dinámica

Como resultado obtendremos los eventos OpenACC de ./a.out.

En definitiva, el código de la librería lo usaremos como un wrapper en Extrae (openacc_wrapper).

4 Instrumentación de OpenACC en Extrae

En este capítulo explicaremos cómo hemos realizado la instrumentación de OpenACC en Extrae. La librería del capítulo anterior y el mecanismo LD_PRELOAD son esenciales para hacer esta implementación. La idea técnica es la misma que 3.1.

Actualmente, Extrae es capaz de capturar y almacenar información relativa al análisis de rendimiento de programas escritos usando el modelo CUDA[6], entre otros modelos de programación paralela. Hemos seguido la misma metodología que se siguen con CUDA para instrumentar OpenACC en Extrae.

Para hacer este proceso hay que hacer las modificaciones pertinentes en el código fuente de Extrae. En este capítulo explicaremos las ampliaciones y modificaciones que le hemos hecho al código fuente de Extrae para instrumentar OpenACC.

Extrae está en su versión 3.7.1 actualmente (Marzo 2020). Hemos descargado el código y lo hemos ido modificando a partir de esta versión.

Extrae es una herramienta que utiliza diferentes mecanismos de interposición para capturar información de análisis de rendimiento, uno de ellos es el LD_PRELOAD.

Nosotros usaremos el mecanismo LD_PRELOAD. Esto es, implementaremos una librería dinámica llamada **liboacctrace.so**, que usa la misma idea que 3.1, para capturar información relativa al análisis de rendimiento de OpenACC. Es decir, liboacctrace.so será una librería más dentro de Extrae.

En las siguiente secciones explicaremos, esencialmente, todos los aspecto técnicos para que Extrae pueda capturar información relativa al análisis de rendimiento de aplicaciones OpenACC.

4.1 Eventos Extrae - OpenACC

En general, en Extrae se permiten definir eventos personalizados para cualquier aspecto que queramos indicar/representar. Es decir, Extrae va más allá en este sentido, y permite tener un nivel de detalle mayor en cuanto a la semántica de la instrumentación. Es por eso que, en primer lugar, definiremos una relación entre eventos Extrae-OpenACC.

[6] "CUDA." https://arcb.csc.ncsu.edu/~mueller/cluster/nvidia/GPU+CUDA.pdf. Es un modelo de programación paralela propiedad de Nvidia

También podemos tener eventos para representar movimiento de cierto tipo de parámetros en el programa, más allá de indicar simplemente las operaciones que es lo que buscamos nosotros. Es por eso que los eventos de Extrae no deben ser necesariamente y exclusivamente los mismos que los de OpenACC.

Como conclusión, debemos definir y relacionar los eventos genuinos de OpenACC, vistos en la tabla 1, con los que queremos crear en Extrae, para poder representar y registrar la actividad de OpenACC.

```
#define OPENACC_END_VAL                          0

enum OPENACC_LAUNCH {
  OPENACC_ENQUEUE_LAUNCH_VAL=1,  //acc_ev_enqueue_launch_start
  OPENACC_DEVICE_EXECUTE_VAL,    //valor custom para indicar la ejecución en la GPU
};

enum OPENACC_DATA {
  OPENACC_CREATE_VAL=3,          //acc_ev_create
  OPENACC_DELETE_VAL,            //acc_ev_delete
  OPENACC_ALLOC_VAL,             //acc_ev_alloc
  OPENACC_FREE_VAL,              //acc_ev_free
  OPENACC_ENQUEUE_UPLOAD_VAL,    //acc_ev_enqueue_upload_start
  OPENACC_ENQUEUE_DOWNLOAD_VAL   //acc_ev_enqueue_download_start
};

enum OPENACC_OTHERS {
  OPENACC_DEVICE_INIT_VAL=9,     //acc_ev_device_init_start
  OPENACC_DEVICE_SHUTDOWN_VAL,   //acc_ev_device_shutdown_start
  OPENACC_RUNTIME_SHUTDOWN_VAL,  //acc_ev_runtime_shutdown
  OPENACC_COMPUTE_CONSTRUCT_VAL, //acc_ev_compute_construct_start
  OPENACC_ENTER_DATA_VAL,        //acc_ev_enter_data_start
  OPENACC_EXIT_DATA_VAL,         //acc_ev_exit_data_start
  OPENACC_UPDATE_VAL,            //acc_ev_update_start
  OPENACC_WAIT_VAL               //acc_ev_wait_start
};

#define OPENACC_LAUNCH_EV                        65000001
#define OPENACC_DATA_EV                          65000002
#define OPENACC_OTHERS_EV                        65000003
```

Código 3: Relación entre evento Extrae y evento OpenACC, fichero events.h

En el código podemos ver la relación entre eventos Extrae-OpenACC. Nosotros consideramos sólo tres eventos, un evento Extrae por cada tipo de evento de OpenACC (OPENACC_LAUNCH_EV, OPENACC_DATA_EV y OPENACC_OTHERS_EV). Los eventos OpenACC en Extrae empiezan por 65xxxxxx.

Por otro lado, por cada tipo de evento tenemos los diferentes valores que puede tener en función de la funcionalidad del mismo. Para evitar tener eventos redundantes por cada tipo de finalización (end), se define un único valor de finalización para todos (OPENACC_END_VAL con valor 0).

En el fichero events.h, que está en src/common/, están definidos los eventos de Extrae para todos los modelos. Este fichero lo hemos modificado para añadir los eventos de OpenACC.

4.2 Integración del modelo OpenACC en Extrae

La integración del modelo (librería dinámica liboacctrace.so) está hecha en **/src/tracer/wrappers/OPENACC.** Hay 3 ficheros clave, junto con sus cabeceras **openacc_wrapper.c, openacc_commons.c y openacc_probe.c.** En esta sección explicaremos para qué sirven estos archivos y que se implementa.

4.2.1 openacc_wrapper

El código que se hizo para la librería en 3.3 hará la función de wrapper de OpenACC en Extrae. Esencialmente, en el openacc_wrapper.c nos dedicamos a capturar los eventos OpenACC.

```
void prof_launch(acc_prof_info* profinfo, acc_event_info* eventinfo,acc_api_info* apiinfo) {
    acc_launch_event_info* launchinfo;
    launchinfo = (acc_launch_event_info*) eventinfo;
    switch(launchinfo->event_type){
        case acc_ev_enqueue_launch_start:
        Extrae_openacc_enqueue_launch_Enter ();
        break;
        case acc_ev_enqueue_launch_end:
        Extrae_openacc_enqueue_launch_Exit ();
        break;
        default:
        break;
    }
 // more code…

void acc_register_library(acc_prof_reg acc_prof_register, acc_prof_reg acc_prof_unregister,
acc_prof_lookup lookup) {
    //capture the event and call prof_launch.
    acc_prof_register(acc_ev_enqueue_launch_start,prof_launch,0);
    acc_prof_register(acc_ev_enqueue_launch_end,prof_launch,0);
   // more code...
}
```

Código 4: Capturando eventos de OpenACC en Extrae

El código es esencialmente similar al de la librería dinámica que se explica en el capítulo anterior. Esto es así ya que en el wrapper hacemos la misma funcionalidad. La diferencia está en que en este código tratamos los eventos con las funciones tipo

`Extrae_openacc_enqueue_launch_Enter ().` La implementación de este tipo de funciones está en el fichero openacc_common.

4.2.2 openacc_common

En el fichero common está la lógica de la instrumentación. Es decir, todos los aspectos relacionados con la instrumentación están en este fichero, como por ejemplo, la inicialización de los devices, tratar los eventos del host, tratar las acciones del device (sincronizar los relojes, medir el tiempo de cálculo del device etc..).

```
//lanuch
void Extrae_openacc_enqueue_launch_Enter () {
    Probe_openacc_enqueue_launch_Enter();
}

void Extrae_openacc_enqueue_launch_Exit () {
    Probe_openacc_enqueue_launch_Exit();
}
```

Código 5: Lógica simplificada de la instrumentación para los eventos launch enter y launch exit

La relación entre el wrapper y el common está en el hecho de que cada vez que se produce un evento OpenACC, el wrapper lo captura, y llama a la función pertinente del openacc_commons donde está la lógica de la instrumentación.

Normalmente, por cada evento hay dos funciones, cuando empieza el evento (start) y cuando acaba (end). Es decir, por ejemplo, cuando capturamos un “acc_ev_enqueue_launch_start” en el wrapper, se llama a la función del commons Extrae_openacc_enqueue_launch_Enter(), aquí es dónde está la lógica para la instrumentación de la entrada.

Por otro lado, cuando capturamos un “acc_ev_enqueue_launch_end”, se llama a la función del commons Extrae_openacc_enqueue_launch_End(), aquí es dónde está la lógica para la instrumentación de la salida.

En cada función de este tipo del openacc_commons, se introduce en un buffer el evento que se capturó en el wrapper de forma cronológica. Para introducir el evento en el buffer lo hacemos usando las funciones del openacc_probe.

Por ejemplo, para el evento “acc_ev_enqueue_launch_start”, usamos la función Probe_openacc_enqueue_launch_Enter(). Por otro lado, para “acc_ev_enqueue_launch_end” se llama a Probe_openacc_enqueue_launch_End().

4.2.2.1 Técnica usada para medir el tiempo de cálculo del device

La profiling API de OpenACC no dispone de una opción para medir el tiempo que la GPU tarda en hacer la computación que ha puesto en cola el host. Es por eso que aprovechando que se van a usar GPU de Nvidia, que soportan CUDA, podemos usar las funciones del runtime de CUDA [7] .

La idea es la misma que OpenACC, pero ahora usamos funciones del runtime de CUDA para medir el tiempo de cálculo del device. Es decir, estamos combinando runtimes de dos modelos de programación distintos.

```
static cudaEvent_t t1, t2;
void Extrae_openacc_enqueue_launch_Enter (const char * kernel_name) {
    Probe_openacc_enqueue_launch_Enter();
    cudaEventCreate(&t1);
    cudaEventRecord(t1,0);
}

void Extrae_openacc_enqueue_launch_Exit() {
    cudaEventCreate(&t2);
    cudaEventRecord(t2,0);
    cudaEventSynchronize(t2);
    float end_time;
    cudaEventElapsedTime (&end_time, device_reference_time, t2);
    cudaEventDestroy(t2);
}
```

Código 6: Código simplificado de la medición del tiempo que tarda la GPU en computar una tarea

Las funciones tipo cudaEventCreate, cudaEventRecord, cudaEventSynchronize y cudaEventElapsedTime son funciones de la runtime API de CUDA.

Esencialmente, estamos poniendo en la cola de ejecución del device, dos kernels falsos, uno antes (cudaEventRecord (t1,0)), y otro después (cudaEventRecord (t2,0)), del kernel que hace la computación real. Estos kernels falsos se encargan de medir el tiempo.

Lo que estamos haciendo en el código anterior se refleja en la siguiente figura de forma gráfica.

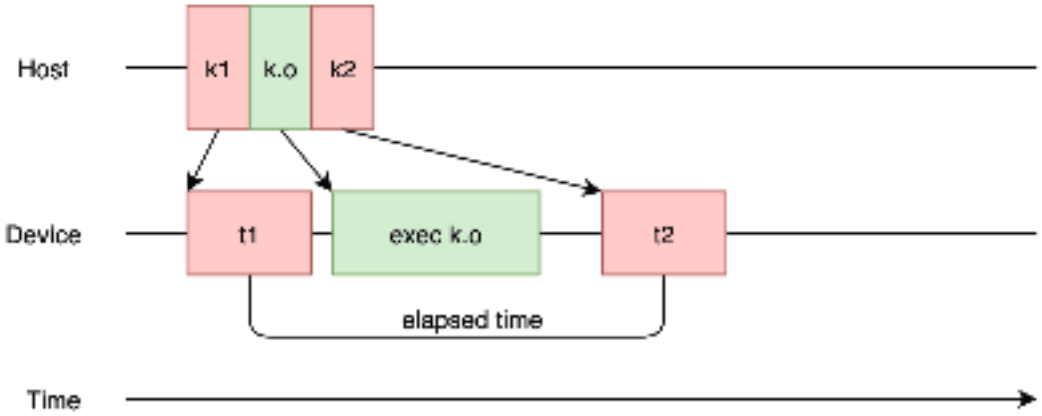

Figura 3: Midiendo el tiempo aproximado que tarda la GPU en ejecutar la tarea

La etiqueta k.o hace referencia al encolado de la tarea original. k1,k2 son las dos tareas falsas encargadas de medir el tiempo inicial y final t1,t2 respectivamente.

Como las computaciones se hacen en orden de encolado, sabemos que primero la GPU ejecuta el kernel que inicia el reloj (lo hará en algún momento en su línea temporal, y cuando a la GPU le convenga, pero será necesariamente antes que el kernel real k.o). Luego, iniciará la computación de nuestro kernel real, el que se encarga de hacer la computacion real.

Finalmente, la GPU ejecutará el segundo kernel falso que se encarga de medir el reloj. Este último kernel también será ejecutado cuando le convenga a la GPU, pero sabemos que será necesariamente después del kernel que hace la computación real.

El elapsed time entre los dos kernels falsos será, aproximadamente, lo que tarda la GPU en hacer la computación de nuestra tarea.

Nótese que el elapsed time es un valor aproximado. El encolado des del host es contiguo, por lo contrario, la ejecución de estos kernels en el device no tiene porque ser contigua. Es decir, puede darse el caso de que inmediatamente después de t1 se haga exec k.o, o que no sea así, ya que la GPU, probablemente, hace otras tareas internamente que no sean ejecutar los kernels.

Además, tampoco sabemos cuándo se va a ejecutar k1 (la tarea falsa inicial) en el device para medir el tiempo. Lo único que sabemos de forma segura es que hay una orden de ejecución de tareas por parte del device. Este orden viene dado por el orden de encolado de las tareas des del host. Aprovechando esta propiedad podemos medir el tiempo aproximado que el device tarda en ejecutar una tarea.

4.2.3 openacc_probe

En el fichero openacc_probe introducimos en el buffer de eventos los eventos del host.

```
//LAUNCH
void Probe_openacc_enqueue_launch_Enter() {
    DEBUG
    if (mpitrace_on && Extrae_get_trace_OpenACC())
        TRACE_MISCEVENTANDCOUNTERS(TIME, OPENACC_LAUNCH_EV, OPENACC_ENQUEUE_LAUNCH_VAL, EMPTY);
}

void Probe_openacc_enqueue_launch_Exit() {
    DEBUG
    if (mpitrace_on && Extrae_get_trace_OpenACC())
        TRACE_MISCEVENTANDCOUNTERS(TIME, OPENACC_LAUNCH_EV, OPENACC_END_VAL, EMPTY);
}
```

Código 7: Insertando eventos de host en el buffer

Las funciones TRACE_MISCEVENTANDCOUNTERS insertan los eventos en el buffer de eventos. Por cada evento insertado en el buffer se debe indicar: el tiempo des del punto de vista del host, el evento extrae-openacc, su valor y su parámetro.

El tiempo del host y el device, al ser ser sistemas heterogéneos e independientes entre sí, no tiene porque ser el mismo tiempo necesariamente. Es decir, los eventos del host se introducen usando el tiempo del host de forma totalmente independiente al tiempo del device.

4.3 Modificaciones en el merger

El merger es una nueva fase en la que unificamos toda la información del análisis de rendimiento que hayamos obtenido, y le damos un formato de salida. Para nuestro caso, el formato de salida es para Paraver. Por ese motivo hemos creado dos ficheros : openacc_prv_events.c y openacc_prv_semantics.c en src/merger/paraver, junto con sus cabeceras.

Cada evento significa que ocurre una acción OpenACC, a cada evento le podemos dar como mínimo una semántica, o más de una, en cuyo caso se debe implementar openacc_prv_semantics.

En el fichero openacc_prv_semantics usamos la función trace_paraver_event (cpu, ptask, task, thread, current_time, EvType,EvValue); para cada tipo de evento, ya que en nuestro caso,

esencialmente, queremos marcar el inicio y el final de las operaciones de OpenACC en paraver, así como la actividad de la GPU.

Con lo cual, nosotros solo tenemos una semántica y con añadir los eventos con trace_paraver_event por cada tipo de evento será suficiente.

Por otro lado, Extrae, una vez ejecutado, produce tres ficheros de salida (output.pcf, output.row y output.prv). Estos ficheros contienen la información relativa al análisis de rendimiento. Están orientados para ser vistos, gráficamente (en una línea temporal), usando Paraver.

En el fichero openacc_prv_events.c indicamos que eventos deben ir al fichero output.pcf. Indicamos el tipo de evento y los valores. Luego paraver, automáticamente, puede interpretarlo para asignarle el nombre y asociarle un color.

```
if (inuse[OPENACC_LAUNCH_INDEX]) {
    fprintf (fd, "EVENT_TYPE\n" "%d   %d    OPENACC LAUNCH\n", 0, OPENACC_LAUNCH_EV);
    fprintf (fd, "VALUES\n" "0 End\n");
    fprintf (fd, "%d OPENACC_ENQUEUE_LAUNCH\n", OPENACC_ENQUEUE_LAUNCH_VAL);
    fprintf (fd, "%d OPENACC_DEVICE_EXECUTE\n", OPENACC_DEVICE_EXECUTE_VAL);
}
```

Código 8: Formato de salida para Paraver de eventos launch: nombre del evento y su valor

El código indica que si se está usando un evento tipo Launch, se deben de escribir los valores para este tipo de evento (enum OPENACC_LAUNCH del events.h) en el fichero output.pcf.

La fase de merger es ajena a la librería dinámica liboacctrace.so, pues esta fase es común para todos los modelos de programación que es capaz de instrumentar Extrae.

5 Validación de los resultados de la instrumentación de OpenACC en Extrae

En este capítulo, hacemos la validación del desarrollo de la instrumentación de OpenACC en Extrae que se ha explicado en el capítulo anterior. Mostraremos y explicaremos los resultados del registro de la actividad del host y del device.

Debemos verificar que la traza que obtenemos con Extrae contiene la secuencia correcta de eventos que ejecuta el programa. Es decir, debemos validar la actividad de OpenACC que hemos capturado y almacenado en una traza de Paraver.

Usaremos la herramienta Paraver 1.2.2, para validar los resultados. Usaremos dos tipos de vistas Paraver, la vista last val y la vista last type.

La vista “last val” filtra los eventos según su valor. De esta forma consigue mostrarnos el inicio y el final de los eventos, ya que cada inicio del evento tiene un valor numérico mayor que 0, pero todos terminan con valor 0, provocando así un cambio de color. El cambio de color permite distinguir visualmente el evento.

La vista “last type” filtra los eventos por el tipo de evento al que pertenecen (launch, data y others), según la tabla 1.

Vamos a ejecutar todo los programas que usaremos para validar, con el mecanismo ld_preload de Extrae. Obtendremos trazas como resultado de la ejecución Figura 1.

5.1 Resultados del registro de la actividad del Host

En esta sección validamos, exclusivamente, la actividad del host. La actividad del host se valida ejecutando códigos de aplicaciones OpenACC. Los #pragmas y funciones de OpenACC hacen saltar determinados eventos. Estos eventos pertenecen a alguna de las tres categorías de eventos (launch, data, others).

En primer lugar, debemos validar que los eventos se capturen de forma cronológica, y que en Paraver también se muestren de forma cronológica.

En segundo lugar, debemos verificar, usando la vista de last val, que estén todos los eventos que se han capturado.

En tercer lugar, debemos verificar, usando la vista de last type, que los eventos se agrupen de forma adecuada dependiendo de si son tipo launch, data o others , tabla 1.

Finalmente, para compilar los programas que usaremos para validar la actividad del host, se ha usado el compilador PGI/20.1 con el flag -acc. El flag sirve para que se puedan interpretar los #pragmas de OpenACC.

5.1.1 Ejemplo 1: kernels.c

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[] )
{
  int KERNELS = 1; // #jobs to enqueue
  for (int i = 0; i < KERNELS; ++i)
  {
  #pragma acc kernels
    {
      //job to do in gpu
      for (int i = 0; i < 1000; ++i)
      {

      }
    }
  }
  return 0;
}
```

Código 9: Código del programa kernels.c

El programa kernels ejecuta un bucle que va encolando tareas que debe hacer el device (la GPU). La tarea que encola es un bucle fijo de 1000 iteraciones que se ejecuta de forma paralela en la GPU.

El código kernels.c es muy simple, de hecho, al no hacer ninguna tarea concreta, probablemente el compilador lo optimize. Aún así, lo elegimos porque es el ejemplo más simple con el que podemos validar la actividad OpenACC.

Para variar, podemos ir modificando el número de tareas que se que host pone en cola, si cambiamos la variable KERNELS.

```
TIME|EVENT_NAME|NUM
2700555730228069 acc_ev_device_init_start
2700555730420200 acc_ev_device_init_end
2700555730438016 acc_ev_compute_construct_start
2700555730454767 acc_ev_enqueue_launch_start
2700555730617468 acc_ev_enqueue_launch_end
2700555730627856 acc_ev_wait_start
2700555730640860 acc_ev_wait_end
2700555730648119 acc_ev_compute_construct_end
```

Figura 4: Output del programa kernels.c, con una tarea KERNELS=1

La primera columna de la figura anterior indica el tiempo del host. Podemos observar que el evento i-ésimo+1 tiene un valor del tiempo mayor que el evento i-ésimo. Lo que indica los eventos se están capturando de forma cronológica.

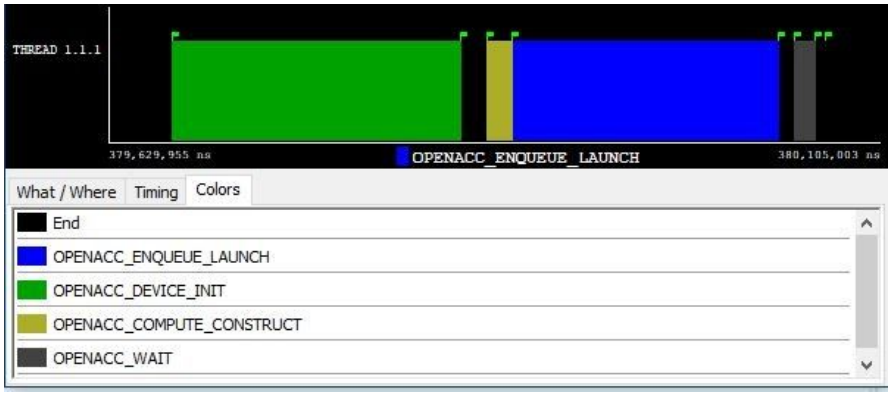

Figura 5: Vista last val de Paraver del programa kernels.c, con KERNELS=1

De la figura anterior podemos verificar que los ocho eventos (banderas de color verde), se registran y que lo hacen en orden cronológico.

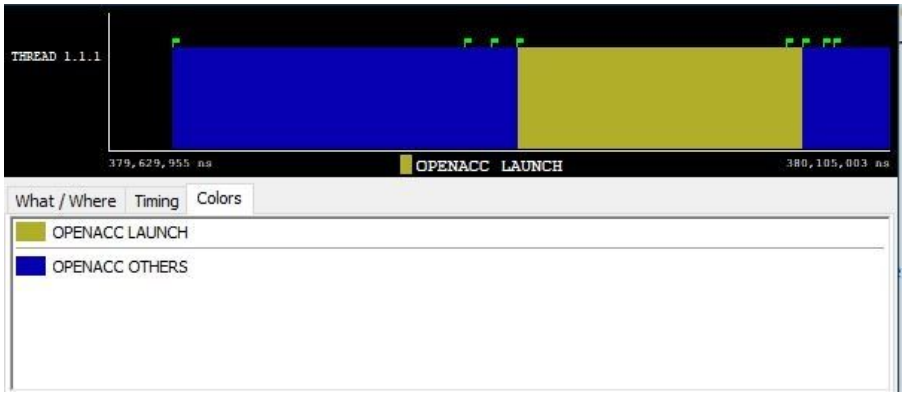

Figura 6: Vista last type de Paraver del programa kernels.c, con KERNELS=1

Podemos verificar de la figura anterior, que no hay ningún evento del tipo data. Todos los eventos son, o bien de launch, o de others. No hay ningún evento de data, pues no se hace ninguna operación que implique movimiento de data entre el host/device en el programa kernels. Lo que indica que la actividad está bien registrada.

Ahora podemos aumentar los kernels de 1 a 2 cambiando la variable KERNELS=2 del programa kernels.c. Veremos que el número de eventos se ha incrementado. La inicialización (device_init) solo se debería ver una vez, en cambio todo lo demás debería replicarse.

```
kernel number: 0
2703915688823136 acc_ev_device_init_start
2703915689010117 acc_ev_device_init_end
2703915689028060 acc_ev_compute_construct_start
2703915689045487 acc_ev_enqueue_launch_start
2703915689229595 acc_ev_enqueue_launch_end
2703915689239921 acc_ev_wait_start
2703915689252143 acc_ev_wait_end
2703915689259350 acc_ev_compute_construct_end
kernel number: 1
2703915689270115 acc_ev_compute_construct_start
2703915689276829 acc_ev_enqueue_launch_start
2703915689320142 acc_ev_enqueue_launch_end
2703915689326328 acc_ev_wait_start
2703915689336599 acc_ev_wait_end
2703915689342298 acc_ev_compute_construct_end
```

Figura 7: Output del programa kernels.c, con dos tareas.

Efectivamente, al duplicar el número de kernels, también se duplican los eventos para el kernel, tal como se muestra en la figura anterior. El evento device_init solo se hace una vez, pues el código se ha ejecutado usando una sola GPU.

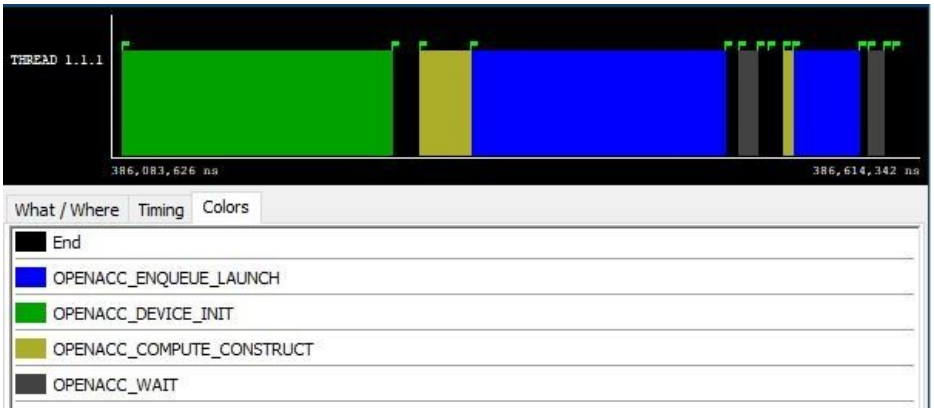

Figura 8: Vista last val de Paraver del programa kernels.c, con KERNELS=2.

Podemos ver en la figura anterior que todos los eventos se vuelven a registrar en la vista last val y de forma cronológica.

Por último, para terminar de validar kernels.c, podemos ejecutarlo con 10 kernels para ver la vista de last val. Esto validará la multiplicidad de la implementación. Es decir, si ponemos en cola 10 kernels , entonces se registra exactamente la actividad de 10. Por lo contrario, si ponemos en cola 2 kernels, debe salir la actividad de 2, tal como sucede en la figura anterior.

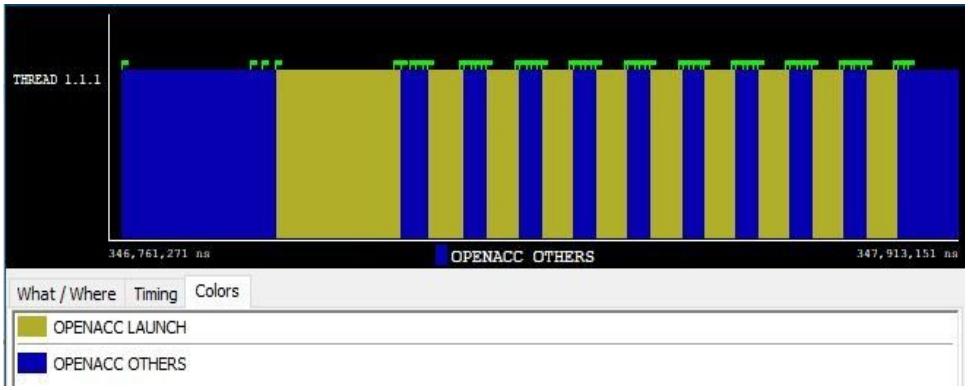

Figura 9: Vista last val de Paraver del programa kernels.c, con KERNELS=10

En la figura anterior vemos la actividad de 10 kernels. Exactamente los 10 kernels que el host pondrá en cola, para que el device los ejecute. Esto valida la multiplicidad de la implementación en cuanto al registro de la actividad del host.

5.1.2 Ejemplo 2 : vadd.c

```
#include <stdio.h>
#include <stdlib.h>

void vecaddgpu( float *restrict r, float *a, float *b, int n ){
    #pragma acc kernels loop copyin(a[0:n],b[0:n]) copyout(r[0:n])
    for( int i = 0; i < n; ++i ) r[i] = a[i] + b[i];
}

int main( int argc, char* argv[] ){
    int n; /* vector length */
    float * a; /* input vector 1 */
    float * b; /* input vector 2 */
    float * r; /* output vector */
    int i;
    if( argc > 1 ) n = atoi( argv[1] );
    else n = 1000; /* default vector length */
    if( n <= 0 ) n = 1000;
    a = (float*)malloc( n*sizeof(float) );
    b = (float*)malloc( n*sizeof(float) );
    r = (float*)malloc( n*sizeof(float) );
    for( i = 0; i < n; ++i ){
         a[i] = (float)(i+1);
         b[i] = (float)(1000*i);
    }
    /* compute on the GPU */
    vecaddgpu( r, a, b, n );
   return 0;
}
```

Código 10: Código del programa vadd para sumar dos vectores usando la GPU

El programa vadd computa la suma de dos vectores en la GPU. El host transfiere data al device, en concreto, transfiere los dos vectores que se deben sumar. Luego, des del device se vuelve a transferir el resultado de la suma (vector r, result) al host.

Se usa este ejemplo para terminar de validar la actividad del host, ya que este ejemplo tiene movimiento de datos entre el host/device , lo que hace que se capture eventos relacionados con data.

El ejemplo 1 (kernels.c) no tenía movimiento de datos entre host/device. Era normal no ver ningún evento tipo data.

```
2731315541352748 acc_ev_create
2731315541381208 acc_ev_alloc
2731315549677227 acc_ev_enqueue_upload_start
2731315549698287 acc_ev_enqueue_upload_end
2731315549764189 acc_ev_enqueue_upload_end
2731315549818701 acc_ev_create
2731315549835374 acc_ev_alloc
2731315549845881 acc_ev_enqueue_upload_start
2731315549851000 acc_ev_enqueue_upload_end
2731315549885790 acc_ev_enqueue_upload_end
2731315549934612 acc_ev_create
2731315549948393 acc_ev_alloc
2731315550204881 acc_ev_delete
2731315550216699 acc_ev_delete
2731315550225916 acc_ev_enqueue_download_start
2731315550232107 acc_ev_enqueue_download_end
2731315550278607 acc_ev_enqueue_download_end
2731315550285357 acc_ev_delete
```

Figura 10: Output del programa vadd, filtrando solo los eventos relacionados con data

De este ejemplo, en concreto, nos interesa ver que todos los eventos de la categoría data que se producen, como resultado de la ejecución del programa, están siendo capturados. Por ese motivo, en el output de la figura anterior, hemos obtenido traza, pero filtrando los eventos y solamente mostramos los que son del tipo data.

Además, tal como podemos observar en la figura anterior, los eventos están siendo capturados de forma cronológica. El número delante del evento i-ésimo es siempre menor al número delante del evento i-ésimo+1. Lo que valida que son eventos capturados de forma cronológica.

Figura 11: Vista last val de Paraver del programa vadd.c; visualizando los eventos de data

En la figura anterior verificamos que la lo eventos tipo data están siendo capturados.

```
2733971241147516 acc_ev_device_init_start
2733971241385183 acc_ev_device_init_end
2733971241396802 acc_ev_enter_data_start
2733971241467334 acc_ev_create
2733971241492900 acc_ev_alloc
2733971250761072 acc_ev_enqueue_upload_start
2733971250786099 acc_ev_enqueue_upload_end
2733971250863910 acc_ev_enqueue_upload_end
2733971250920240 acc_ev_create
2733971250940759 acc_ev_alloc
2733971250953315 acc_ev_enqueue_upload_start
2733971250958166 acc_ev_enqueue_upload_end
2733971251005823 acc_ev_enqueue_upload_end
2733971251052516 acc_ev_create
2733971251068508 acc_ev_alloc
2733971251080957 acc_ev_wait_start
2733971251097401 acc_ev_wait_end
2733971251106360 acc_ev_enter_data_end
2733971251122674 acc_ev_compute_construct_start
2733971251140097 acc_ev_enqueue_launch_start
2733971251323867 acc_ev_enqueue_launch_end
2733971251334994 acc_ev_wait_start
2733971251347550 acc_ev_wait_end
2733971251355630 acc_ev_compute_construct_end
2733971251364707 acc_ev_exit_data_start
2733971251372139 acc_ev_exit_data_end
2733971251379428 acc_ev_update_start
2733971251396302 acc_ev_delete
2733971251409573 acc_ev_delete
2733971251421004 acc_ev_enqueue_download_start
2733971251428393 acc_ev_enqueue_download_end
2733971251483275 acc_ev_enqueue_download_end
2733971251491800 acc_ev_delete
2733971251500298 acc_ev_wait_start
2733971251519911 acc_ev_wait_end
2733971251527190 acc_ev_exit_data_end
2733971251533260 acc_ev_update_start
```

Figura 12: Output del programa vadd, sin filtrar por eventos tipo data

Lo que observamos en la figura anterior son todos los eventos de OpenACC que se dan por el hecho de ejecutar el programa vadd.

Figura 13: Vista last type de Paraver del programa vadd.c

De la figura anterior podemos observar que hay un mayor número de eventos relacionados con el movimiento de data, que cualquier otro grupo. Por otro lado, solo hay una computación (en color amarillo). Esto concuerda con el código vadd, ya que la tarea solo se envia una vez para ser computada en la GPU.

Como conclusión del registro de la actividad del host, podemos verificar que los eventos se capturan cronológicamente, se muestran cronológicamente en Paraver y que hay multiplicidad. Esto es,, si ponemos en cola 10 tareas en nuestra aplicación OpenACC, la actividad que se registra es de exactamente 10 tareas, y con toda la secuencia de eventos de host debidamente capturada. En consecuencia, queda validada la implementación para el registro de la actividad del host.

5.2 Resultados del registro de la actividad del device

En esta sección validamos el registro de la actividad del device. Lo haremos ejecutando el programa kernels.c . Se ejecuta el programa usando un device.

Lo primero que deberíamos observar, a diferencia de la sección de anterior, es que ahora tenemos un objeto más junto al host. Este nuevo objeto es el device.

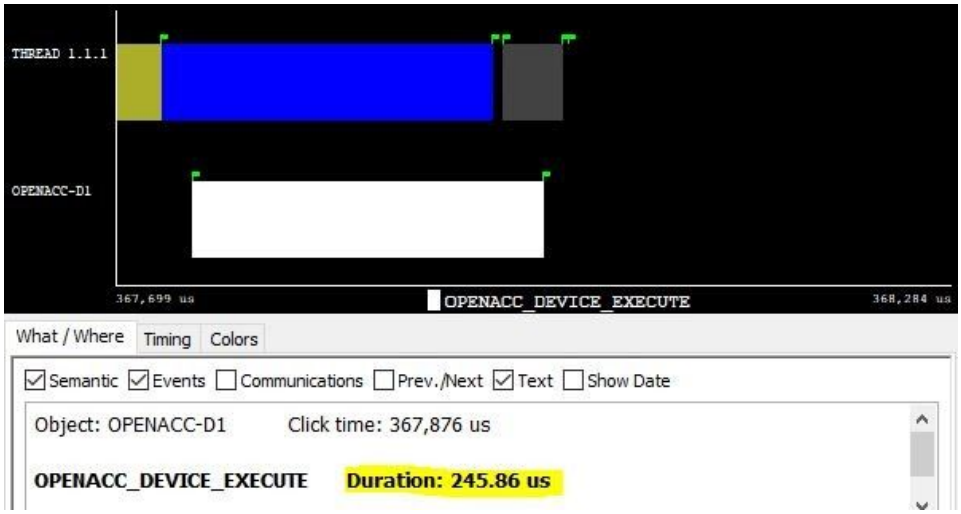

Figura 14: Actividad del device del programa kernels.c, con KERNELS=1

De la figura anterior, podemos observar y validar varios aspectos.

En primer lugar, validamos la presencia del device (OPENACC-D1).

En segundo lugar, vemos (de color blanco) la ejecución del kernel, que el host a puesto en cola, y el device ha ejecutado. Como se pone en cola exactamente una tarea (KERNELS=1), debe aparecer exactamente la ejecución de una tarea en el device, tal como podemos validar en la figura anterior.

Podemos observar que la ejecución de la tarea en el device dura **245.86 us**.

```
Accelerator Kernel Timing data
/gpfs/home/bsc41/bsc41636/extrae-funcional-ejemplos/results/device/kernels/kernels.c
  main  NVIDIA  devicenum=0
    time(us): 4
    11: compute region reached 1 time
        13: kernel launched 1 time
            grid: [8]  block: [128]
             device time(us): total=4 max=4 min=4 avg=4
            elapsed time(us): total=164 max=164 min=164 avg=164
```

Figura 15: Duración de la tarea de la ejecución de un kernel en el device usando el compilador

En la figura anterior, podemos ver el tiempo en microsegundos, que dura la computación en el device (obtenida usando el compilador PGI). El dato relevante es el “elapsed time” en (us), que en este caso es de **164 us**.

Los 245.86 us frente a los 164 us es una aproximación bastante realista dado la unidad de tiempo (us), y dado a que estamos comprando el tiempo de Extrae, frente al tiempo que obtenemos del propio compilador PGI (PGI_ACC_TIME=1). Además, también debemos tener en cuenta que las mediciones del elapsed time que hacemos en Extrae son aproximadas, tal como lo indicamos en 4.2.2.1.

Finalmente, para validar la multiplicidad, ponemos en cola 10 tareas (KERNELS=10). La ejecución de estos 10 kernels debería quedar registrada en la línea del device.

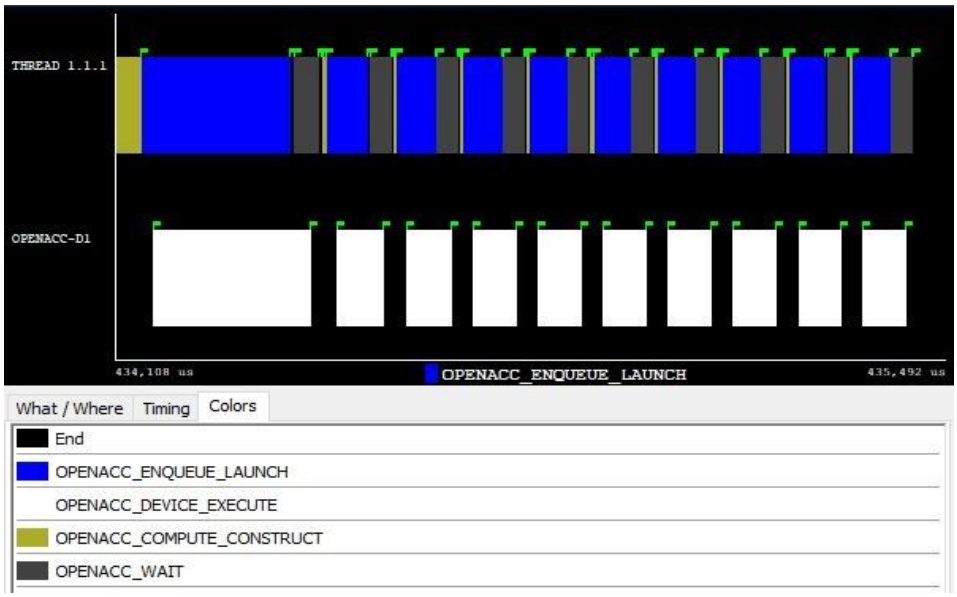

Figura 16: Actividad del device ejecutando 10 tareas, KERNELS=10

En la figura anterior queda validada la multiplicidad. Es decir, cuando se pone en cola un kernel, entonces vemos en el device, la actividad de un kernel exactamente. Por lo contrario, si

se ponen en cola 10, veremos la actividad de 10. Esta es la conclusión de obtenemos de la figura anterior.

Como conclusión del registro de la actividad del device, podemos validar que se puede registrar la actividad del device. En concreto, hemos registrado y validado que se pueda observar, en la línea del device, lo que tarda el device en hacer la computación (de forma aproximada), y la multiplicidad en cuanto al número de tareas que el host pone en cola, y que el device ejecuta.

5.3 Validación del overhead de la instrumentación de OpenACC en Extrae.

Por el hecho de hacer el análisis de rendimiento se produce un overhead.

En este caso, se entiende por overhead el tiempo de ejecución de más, que se suma al tiempo de ejecución real, por el hecho de hacer el análisis de rendimiento de un aplicación. Esto es, overhead = time_to_perform_analysis_on_execution_time.

final_time = real_execution_time + overhead

Hemos de conseguir que el “time_to_perform_analysis_on_execution_time” sea el mínimo posible durante la implementación de OpenACC en Extrae. Esto influye directamente sobre cómo debemos hacer la implementación.

Lo que pretendemos validar es que la proporción en el overhead se mantenga. Esto es, una tarea que ha durado menos (en tiempo de ejecución), debería producir un overhead igual, en proporción, que una tarea que dura más (en tiempo de ejecución).

El entorno de pruebas es el clúster CTE-POWER. Las pruebas se han realizado usando una gráfica Tesla V100-SXM2-16GB. El compilador usado es PGI/20.1. La versión de Extrae es la 3.7.1 (más la implementación de OpenACC).

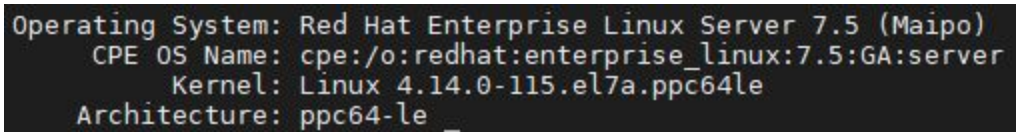

Figura 17: Información del entorno usado para hacer el análisis del overhead

El programa usado para medir el overhead es matrix_sum.c . El programa suma dos matrices de size N*N usando el modelo de programación paralela OpenACC.

```
#define SIZE 10000
float a[SIZE][SIZE];
float b[SIZE][SIZE];
float c[SIZE][SIZE];

double what_time_is_it()
{
    struct timespec now;
    clock_gettime(CLOCK_REALTIME, &now);
    return now.tv_sec + now.tv_nsec*1e-9;
}

int main() {
  double time = what_time_is_it();
  int i,j,k;
  // Initialize matrices.
  for (i = 0; i < SIZE; ++i) {
    for (j = 0; j < SIZE; ++j) {
      a[i][j] = (float)i + j;
      b[i][j] = (float)i - j;
      c[i][j] = 0.0f;
      }
    }
  // Compute matrix sum.
  #pragma acc kernels copyin(a,b) copy(c)
  for (i = 0; i < SIZE; ++i) {
    for (j = 0; j < SIZE; ++j) {
      c[i][j] = a[i][j] + b[i][j];
      }
    }
  printf("time taken %.6lf (us)\n", (what_time_is_it()*1000000) - (time*1000000));
  return 0;
}
```

Código 11: Código del programa matrix_add.c para sumar dos matrices usando OpenACC

Variando el size de las matrices a sumar, en pasos de +5000 empezando desde 1000, el tiempo de ejecución aumentará. Lo que pretendemos validar es que el overhead de hacer la instrumentación se mantenga a pesar de que aumente el tiempo de ejecución del programa.

El overhead de la instrumentación de OpenACC en Extrae será aceptable en la medida en que seamos capaces de mantener su proporción, conforme aumenta el tiempo de ejecución.

N (tamaño)	Tiempo Ejecución (us)	Tiempo Extrae liboacctrace.so (us)	Relación
1000	297118,42	305247,62	1,027360135
5000	374007,0998	376006,86	1,005346851
10000	595749,44	606572,01	1,018166312
15000	973543,67	976914,55	1,003462485
20000	1477898,68	1495384,79	1,011831738
25000	2142791,17	2163302,94	1,009572454

Tabla 2: Resultados de la proporcionalidad del overhead en relación al tamaño de la entrada

La relación se mide como Tiempo Extrae / Tiempo S.O. Observamos que esta relación se mantiene .

Relación entre tiempo de ejecución y el tiempo de Extrae

El tiempo de ejecución, es el tiempo de la ejecución convencional en (us) . El tiempo de Extrae de ejecución más el overhead que produce nuestra implementación.

El hecho de aumentar la entrada implica un aumento del tiempo de ejecución (tanto el de ejecución, como el de Extrae para registrar la actividad de OpenACC), pero no del overhead.

No se produce un overhead más allá del esperado por el hecho de emitir/capturar eventos que se muestra en la siguiente figura.

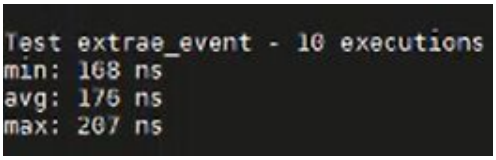

Figura 18: Tiempo en emitir un evento, fase traceo, test del propio Extrae.

Como conclusión de esta sección, podemos afirmar que la implementación es eficiente de acuerdo con lo significa ser “eficiente” en este estado del arte en concreto.

Esto es, no provocar un overhead no admisible que invalide la implementación. Donde no admisible significa ser no proporcional a medida que aumenta el tiempo de ejecución del programa.

Por otro lado, como conclusión general del capítulo, podemos afirmar que, hemos registrado la actividad del host y del device correctamente, sin introducir un overhead mayor del esperado. Esto válida la implementación en su totalidad.

6 Futuras mejoras y conclusiones

Como en cualquier proyecto que consiste en añadir nuevas funcionalidades. Siempre se puede mejorar. En este capítulo explicaremos las mejoras que podría tener este proyecto, así como las conclusiones generales.

6.1 Ampliaciones en el registro de la actividad del device

Las mejoras que podemos hacer son, esencialmente, en el registro de la actividad del device. En concreto, proponemos dos mejoras.

Representar la actividad de uploads y downloads:

El modelo OpenACC usa los device para computar tareas que son programadas por el host. El flujo normal es que el host mande una tarea y la información (data) necesaria para realizar la computación. De vuelta, como resultado de la computación, el device vuelve a transferir la información al host.

En consecuencia, una mejora que se podría hacer es representar en la línea del objeto device (OPENACC-D1) representada en la sección 5.2, el tiempo en el que se hacen este tipo de operaciones de memoria (download y upload de la información).

Para ello, se debería/podría emplear la misma técnica que se empleado para enqueue_launch en la sección 4.2.2.1. Es decir, poner en cola dos kernels falsos , con la finalidad de medir el elapsed time.

Esta vez se debería implementar las funciones Extrae_openacc_enqueue_upload y Extrae_openacc_enqueue_download en vez de Extrae_openacc_enqueue_launch. Se debe tener en cuenta la sincronización entre el host y el device, pues son sistemas heterogéneos y el tiempo del device no es el mismo que el de host.

Además, el código sería potencialmente complejo, ya que se tendría que tener en cuenta registrar la actividad para múltiples acciones que hace el device, y no solo para enqueue_launch (como se hace hasta ahora). Esto último implicaría, combinar el registro de múltiples actividades que el device hace usando estructuras (structs y arrays) junto con las sincronizaciones pertinentes entre host device, y entre las múltiples actividades a registrar.

Representar la actividad de más de un device.

Por otro lado, OpenACC también permite usar más de un device para realizar tareas. Es por eso que otra mejora que podríamos hacer es registrar la actividad para más de un device. A

continuación, mostramos un prototipo de la mejora que estamos mencionando en la siguiente figura

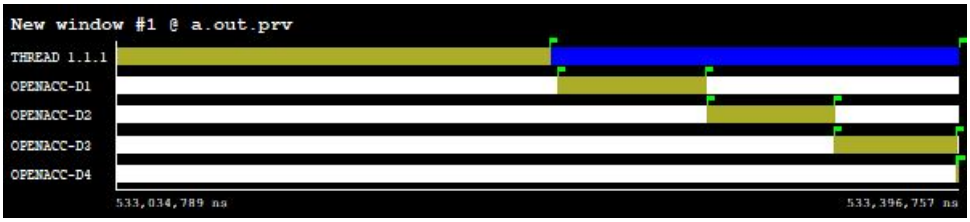

Figura 19: Prototipo para representar la actividad de más de un device

Cada device que esté usando nuestro programa, vendrá representada por una línea (OPENACC-DX), donde X es el identificador del device.

Para hacer esto, se debería de usar estructuras robustas en el código (structs, arrays de structs, etc..), para organizar todo lo relativo a poner en cola los kernels falsos para calcular los elapsed times. En concreto, esta mejora se debería implementar en openacc_commons.c Por otro lado, también haría falta modificar/ampliar lo relativo a los devices y los buffers de eventos.

6.2 Conclusiones

El objetivo y los subobjetivos técnicos se han cumplido. También hemos conseguido implementar las funcionalidades que se han especificado.

En primer lugar, he logrado que la herramienta Extrae del BSC-CNS tenga soporte para instrumentar aplicaciónes OpenACC. He realizado todas las extensiones pertinentes en el código fuente de Extrae 3.7.1 para que así sea.

Ahora somos capaces de registrar la actividad de una aplicación OpenACC a lo largo de su ejecución. La actividad del host se registra por completo. De la actividad del device, se registra el intervalo de tiempo, de forma aproximada, en el que el device ejecuta la computación que ha puesto en cola el host.

Registrar la actividad del device ha sido la mayor la dificultad técnica del proyecto. Ha implicado tener que hacer investigación, pues la profiling API de OpenACC [12] no ofrece una opción directa de hacerlo.

El registro de la actividad del device se ha resuelto usando las funciones del runtime de CUDA, junto con la técnica de poner en cola de dos kernels falsos para medir el tiempo, tal como se explica en 4.2.2.1.

En este sentido, al registro de la actividad del device hay que añadirle las mejoras mencionadas en 6.1. Aunque lo esencial, es decir, medir el tiempo de cálculo del device se ha conseguido.

En segundo lugar, nuestra implementación es multi compilador, ya que sigue el estándar oficial de OpenACC. Es decir, cualquier aplicación OpenACC compilada con algún compilador que también siga el estándar, podrá ser analizada usando Extrae.

El compilador en cuestión debe tener un archivo llamado acc_prof.h, ya que este fichero contiene las definiciones de la profiling API de OpenACC [12] que se usa en Extrae para registrar la actividad de OpenACC.

En tercer lugar, se ha conseguido registrar la actividad de OpenACC usando la profiling API de OpenACC en su versión 3.0 [12], tal como queríamos.

Finalmente, podemos concluir que los objetivos se han cumplido, las funcionalidades se han implementado y las dificultades técnicas y no técnicas (pandemia covid-19) se han superado.

7 Gestión del proyecto

7.1 Informe de planificación temporal

Este proyecto se va a desarrollar en el BSC-CNS bajo un convenio de prácticas. El contrato de prácticas (Convenio de cooperación educativa de la UPC) tiene vigencia del 1/02/2020 hasta el 15/06/2020.

Las horas de las que dispongo para hacer este proyecto es de 540 horas (30 horas /crédito x 18 créditos del proyecto final de grado), es decir, 540 horas.

NOTA:

Durante el desarrollo de este proyecto ha habido una pandemia que ha obligado al confinamiento de la población a nivel mundial, y consecuentemente, a la cancelación de muchos planes. Por ese motivo, hay algunas desviaciones en la planificación temporal y presupuestaria del proyecto.

Por otro lado, hay otras desviaciones que no tienen nada que ver con esta situación, y son las típicas desviaciones que pueden surgir durante el desarrollo de un proyecto de estas características.

7.1.1 Planificación temporal

A continuación, pasaremos a detallar las tareas y su planificación temporal. La planificación está dividida por fases, pero el nivel de detalle de planificación es a nivel de tarea.

Por cada tarea, especificaremos en qué consiste, las horas dedicadas, las dependencias respecto a otras tareas y si han habido desviaciones respecto la planificación inicial.

En el documento GEP están los detalles de la planificación estimada. Así pues, en este documento haremos la comparativa entre la planificación estimada y la real. Las tareas que se han añadido, o que se han tenido que modificar, las explicaremos de nuevo siguiendo el mismo método que en el documento GEP.

Aunque se han modificados algunas tareas, o se haya eliminado/añadido otras, siempre se ha intentando mantener el número de horas de dedicación de forma global, es decir, unas 540 horas.

7.1.2 Fase de GEP

En esta fase se explicarán los detalles de las tareas relacionadas con la gestión del proyecto. El material usado en esta fase ha sido el portátil y Google Docs.

Reuniones

En las reuniones se hablará del progreso del proyecto y de eventuales preguntas técnicas que pueda tener respecto a la evolución del proyecto. Esta tarea es independiente del resto de tareas.

Debido a la pandemia han habido modificaciones en el número de horas. Se habían planificado 16 horas, pero han sido 30 horas, ya que se han tenido que resolver algunas cuestiones telemáticamente, lo que ha supuesto un incremento de horas.

Explicación del coste temporal del informe de contextualización y alcance del proyecto

Esta tarea consiste en la elaboración del informe de contextualización y alcance del proyecto. Para ello, he tenido que hacer reuniones, buscar información de diversa índole, entenderla y redactar el informe. Todo ello me ha llevado unas 16 horas.

Esta tarea ha dependido de las reuniones que he mantenido. No han habido desviaciones en ningún sentido.

Explicación del coste temporal del informe de la planificación temporal

Esta tarea consiste en la elaboración del informe de la planificación temporal del proyecto. He tenido que redactarlo, pensar en la tareas que tendré que llevar a cabo y aprender a usar la herramienta para hacer diagramas de Gantt.

Por todo ello he tardado 15 horas. Esta tarea es dependiente del alcance del proyecto. No han habido desviaciones en ningún sentido.

Explicación del coste temporal del informe de presupuesto

Esta tarea consiste en la elaboración del informe de los presupuestos. He tenido que hacer cálculos de diversa índole (coste del personal, materiales, imprevistos etc..).

Me ha llevado 15 horas. Esta tarea es dependiente del informe del coste temporal del proyecto y del informe de contextualización/alcance. No han habido desviaciones en ningún sentido.

Explicación del coste temporal del informe de sostenibilidad

Esta tarea consiste en la elaboración del informe de sostenibilidad de este proyecto. Lo he hecho basándome en la matriz de sostenibilidad y analizando los costes en plano económico, social y ambiental.

Me ha llevado 15 horas. Esta tarea es dependiente del informe del alcance del proyecto y del informe de la planificación temporal. No han habido desviaciones en ningún sentido.

Explicación del coste temporal de la elaboración de la memoria

La tarea consiste en la elaboración de la memoria del proyecto.

Esta tarea es dependiente de todas las demás. Le he dedicado 80 horas. No han habido desviaciones en ningún sentido.

Explicación del coste temporal de la presentación oral ante el tribunal.

Para esta tarea debo preparar las diapositivas que voy a presentar ante el tribunal, ensayar la exposición y preparar la demostración.

Esta tarea es dependiente de la elaboración de la memoria. Le dedicaré 15 horas. El material que utilizaré será el portátil, la cámara y el micrófono. La pandemia ha hecho que la presentación sea de forma telemática.

Modificaciones en la fase de gestión del proyecto

En la fase de GEP se ha añadido la tarea del “informe de seguimiento”. Se ha tenido que realizar antes en la fase del sprint final para evaluar el estado del proyecto. Por lo que es una tarea dependiente de la fase de implementación del proyecto

Explicación del coste temporal del informe de seguimiento.

La tarea consiste en hacer una revisión de la planificación del proyecto a nivel global. De valorar los objetivos alcanzados y de detallar las posibles desviaciones junto con sus justificaciones.

Ha supuesto un incremento de 16 horas en la fase de gestión de proyectos. Es dependiente de la fase de implementación del proyecto.

7.1.3 Fase de trabajo previo (alcanzando el estado del arte)

A continuación, pasaremos a detallar las tareas de la fase de trabajo previo realizado con la finalidad de alcanzar el estado del arte.

Ninguna de las tareas de la fase de trabajo previa tiene dependencia con otras tareas. El material usado en esta fase es la máquina CTE-POWER, el portátil, el compilador GCC/PGI, Extrae y Paraver.

Aprender C avanzado y usar el superordenador CTE-POWER

Está tarea consiste, por una parte, aprender aspectos técnicos de punteros y gestión de memoria del lenguaje C. Por otra parte, aprender a usar a nivel de usuario el superordenador CTE-POWER.

En la planificación inicial eran dos tareas separadas. Fueron de las primeras tareas que se realizaron, se hicieron conjuntamente. Tardé menos tiempo del esperado, y por eso se han unido. En concreto 8 horas, cuando se estimaron 16 horas entre las dos. El material usado fue el portátil, el compilador GCC, y el superordenador CTE-POWER.

Aprender el paradigma de OpenACC

Esta tarea consiste en aprender aspectos del paradigma de paralelización de OpenACC y su profiling. La empresa Nvidia ofrece cursos gratuitos online para este propósito. El curso dura unas 20 horas en total.

Por lo tanto, he tardado 20 horas. No han habido desviaciones en ningún sentido.

Aprender Extrae y Paraver a nivel de usuario

Esta tarea consiste en leer manuales de Extrae y Paraver, instalar estas herramientas e ir aprendiendo a usarlas a nivel de usuario.

Esta tarea me ha llevado unas 8 horas. Se estiman 18 horas y eran 2 tareas separadas en la planificación estimada. Pero como ya tenía conocimientos por haberlas usado durante la carrera, no me llevo el tiempo estimado.

Modificaciones en la fase del trabajo previo:

En la fase de trabajo previo todas las tareas siguen siendo independientes aunque han habido reajustes en el número de horas, y el contenido de algunas de ellas. Además, también se ha añadido una tarea nueva:

Aprender sobre los archivos fuentes de Extrae 3.7.1

En la planificación estimada no se contemplaba la tarea de “Aprender sobre los archivos fuentes de Extrae”. Esto no ha sido así, ya que el hecho de poder compilarlo y probarlo con OpenACC, implicaba que se tuviera que saber sobre los archivos fuentes de Extrae. Esto ha supuesto cambios en la planificación de los sprints que explicaremos en la siguiente fase.

La “nueva” tarea “Aprender sobre los archivos fuentes de Extrae” ha sido una tarea que ha requerido mucho más tiempo ya que era el “core” para poder hacer el proyecto. Es por eso que ha acabado siendo una tarea longeva (40 horas). El material usado es el portátil, el superordenador CTE-POWER y el código fuente de Extrae versión 3.7.1.

7.1.4 Fase de la implementación

En esta fase detallaremos las tareas que hemos llevado a cabo para hacer la implementación del proyecto. También comentaremos las desviaciones respecto a la planificación estimada.

En la planificación real sigue habiendo tres sprints, tal como había en la planificación temporal estimada. Aunque han habido cambios significativos dentro de las tareas de cada sprint. Para empezar, los tres sprints ya no contienen tareas similares.

Las desviaciones se deben principalmente a que inicialmente se había planificado hacer la incorporación de la librería dinámica para capturar callbacks de OpenACC de forma gradual. Es decir, una vez añadida una funcionalidad nueva a la librería dinámica, se pasaba a implementar la misma funcionalidad, pero en Extrae. Los tres sprints estaban orientados para hacerlo de esta forma.

En la planificación real se decidió cambiar de estrategia. La justificación es que se observó que no hacía falta extender más la librería, y también porque evitamos duplicar tareas, en consecuencia, optimizar mejor el tiempo.

Por ese motivo se decidió heterogeneizar los sprints. En la planificación real, cada sprint cumple un propósito específico y total, que permite avanzar al siguiente, una vez terminado el sprint i-1. Es decir, hay dependencia entre los sprints.

Para la fase de implementación hemos usado la máquina CTE-POWER, el portátil, el compilador PGI, un editor de código (sublime), Extrae y Paraver.

Sprint 1: Librería dinámica para capturar callbacks de OpenACC

La tarea que se ha llevado a cabo en este sprint es hacer la librería dinámica para capturar callbacks de OpenACC.

Buscar y/o hacer ejemplos de código OpenACC

La tarea consistía en buscar ejemplos de códigos OpenACC que permitieran probar la librería. Si no se encontraban en la red, se debían hacer.

Esta tarea me ha llevado 8 horas, ya que ha habido algún código que he tenido que escribirlo yo, pues me servía para testear la librería de una forma más rigurosa.

Implementar la librería externa de OpenACC

La tarea consistía en implementar la librería dinámica para capturar callbacks de OpenACC . Para ello he tenido que leer y entender la especificación de la profiling API de OpenACC 3.0 [12] Para luego, implementar la librería dinámica usando la especificación. Esta tarea me ha llevado 60 horas.

Testear los ejemplos usando la librerÍa externa

La tarea consistía en testear la librería usando los códigos que habíamos escrito y/o buscado. Esta tarea me ha llevado 8 horas. Esta tarea depende de "Buscar y/o hacer ejemplos de código OpenACC" y de "Implementar la librería externa de OpenACC".

Desviaciones respecto al sprint 1 definido en la planificación inicial

He ahorrado (50+40+30) - 50 = 70 horas de la tarea "Implementar funcionalidades en la librería externa de profiling" entre todos los sprints en los que estaba en la planificación inicial. Aunque después, esas mismas horas se han tenido que invertir en tareas como en entender los archivos fuentes de Extrae (de la fase del trabajo previo/alcanzando el estado del arte), y en las tareas de desarrollo en general.

Sprint 2 : Integración de OpenACC en Extrae

El sprint 2 depende de la sprint 1. La tarea que se ha llevado a cabo en este sprint es hacer la integración de la librería dinámica para capturar callbacks de OpenACC en Extrae.

Implementar y editar los ficheros y makefiles de Extrae para incorporar OpenACC

Esta tarea ha consistido en modificar y añadir ficheros de configuración en Extrae para dar soporte para hacer la instrumentación de OpenACC.

Esta tarea me ha llevado 24 horas, ya que se ha tenido que aplicar que las modificaciones en varios ficheros de Extrae e incluso añadir nuevos ficheros.

Implementar e incorporar ficheros del modelo OpenACC en Extrae

Esta tarea ha consistido en añadir ficheros relacionados con la incoporación del modelo OpenACC en Extrae (openacc_wrapper, openacc_commons, openacc_probe y events). Es decir, este es el paso en el que se usa la librería del sprint 1.

La tarea me ha llevado 40 horas. Era de la partes más complejas a nivel técnico. Esta tarea es dependiente de "Implementar y editar los ficheros y makefiles de Extrae para incorporar OpenACC"

Implementar ficheros de eventos de OpenACC para Paraver

Esta tarea ha consistido en añadir ficheros relacionados con el merger en Extrae para que se pueda dar formato de salida tipo Paraver (openacc_prv_events, openacc_prv_semantics).

La tarea me ha llevado 24 horas. Es dependiente de "Implementar e incorporar ficheros del modelo OpenACC en Extrae".

Añadir ejemplos de OpenACC en Extrae

Esta tarea consistía en añadir un ejemplo de prueba de OpenACC. De esta forma, cuando se instala Extrae, con la finalidad de instrumentar OpenACC, podemos tener un ejemplo de código OpenACC con el que probar que la instalación está bien realizada.

Esta tarea ha durado 6 horas, pues se tenía que buscar el ejemplo adecuado, se tenían que modificar ficheros de configuración de Extrae. Esta tarea tiene dependencia respecto a la tarea de "Implementar ficheros de eventos de OpenACC para Paraver".

Testear Extrae con OpenACC usando los ejemplos

La tarea consiste en usar varios ejemplos de códigos escritos usando OpenACC y ejecutar Extrea con capacidad para instrumentar OpenACC. Se debe analizar/testear que todos los eventos OpenACC aparezcan en las vistas de Paraver.

Esta tarea me ha llevado 8 horas. Tiene dependencia respecto a la tarea de "Implementar ficheros de eventos de OpenACC para Paraver"

Desviaciones respecto al sprint 2 definido en la planificación inicial

De forma contraría al sprint 1, en el que habíamos ganado horas, aquí las hemos perdido/invertido. En concreto, hemos tenido que argumentar el sprint 2 en 4 horas (102 horas). El sprint 2 en la planificación estimada era de 98 horas.

También hemos tenido que incorporar nuevas tareas y romper con la homogeneización de los sprints, tal como hemos mencionado anteriormente.

Sprint 3 : Implementación para registrar la actividad de la GPU

El sprint 3 es dependiente del 2. La tarea que se ha llevado a cabo en este sprint es registrar la actividad de la GPU. En concreto, se debe marcar el intervalo de tiempo en el que la GPU hace la computación.

Investigar cómo registrar la actividad de la GPU

Esta tarea ha consistido en investigar cómo registrar la actividad GPU de OpenACC. Se ha tenido que buscar en los foros del compilador PGI, de Nvidia y de OpenACC como se debía/podía hacer, ya que la profiling API que queríamos usar, no permitía hacerlo de forma directa. Esta tarea me ha llevado 40 horas. Esta tarea no tiene dependencias.

Registrar la actividad de GPU : Tiempo de cómputo

Esta tarea ha consistido en modificar el fichero openacc_commons para registrar la actividad de la GPU. Se ha usado una solución ingeniosa para hacerlo, ya que no se podía hacer usando la profiling API de OpenACC. Ha sido la tarea más compleja del proyecto a nivel técnico.

La tarea Me ha llevado 40 horas. Es dependiente de la tarea de “Investigar cómo registrar la actividad de la GPU”.

Testear resultados con vistas de Paraver

La tarea consiste en usar varios ejemplos de códigos escritos usando OpenACC para analizar/testear la actividad del device usando Paraver.

Esta tarea me ha llevado 8 horas. Es dependiente de la tarea “Investigar cómo registrar la actividad de la GPU”.

Desviaciones respecto al sprint 3 definido en la planificación inicial

De forma contraría al sprint 1, en el que habíamos ganado horas, en el sprint tres las hemos perdido/invertido. En concreto, hemos tenido que argumentar el sprint tres en 16 horas. La justificación es que se ha tenido que hacer investigación, debido al contratiempo de no tener una forma directa de registrar la actividad del device usando la profiling API de OpenACC.

Testing general del proyecto:

Esta tarea ha consistido en testear el proyecto en conjunto. Es decir, probar de instalar Extrae en la CTE-POWER y probar todas las fases de implementación en conjunto, usando Extrae y Paraver. Esta tarea me ha llevado 16 horas. Es dependiente de toda la fase de implementación.

7.1.5 Comparativa entre planificación inicial/final y desviaciones

A continuación, tenemos la tabla comparativa entre la planificación estimada y la real.

Tarea estimada	Horas estimadas	Tarea	Horas	Comentario
GEP	**172**	**GEP**	**202**	
Reuniones	16	Reuniones	30	
Informe del alcance del proyecto	16	Informe del alcance del proyecto	16	
Informe de la planificación temporal	15	Informe de la planificación temporal	15	
Informe del presupuesto del proyecto	15	Informe del presupuesto del proyecto	15	
Informe de sostenibilidad del proyecto	15	Informe de sostenibilidad del proyecto	15	
Documento de la memoria del proyecto	80	Informe de seguimiento	16	Se ha añadido esta tarea nueva
Presentación del proyecto	15	Documento de la memoria del proyecto	80	
		Presentación del proyecto	15	
Trabajo previo	**54**	**Trabajo previo**	**72**	Se ha reducido el número de tareas
Aprender C avanzado	8	Aprender C avanzado y usar el superordenador CTE-POWER	8	Se ha juntado con aprender c avanzado y usar CTE-POWER
Aprender paradigma de OpenACC	20	Aprender el paradigma de OpenACC	16	
Aprender a usar el superordenador cte-power	8	Aprender Extrae y Paraver a nivel de usuario	8	
Aprender a usar Extrae	12	Aprender sobre los archivos fuentes de Extrae 3.7.1	40	Se ha añadido
Aprender paraver	6			
Implementación	**298**	**Implementación**	**266**	
Sprint 1	**128**	**Sprint 1 : Implementación de la librería externa**	**76**	Se ha reducido el número de tareas del sprint
Buscar y/o hacer ejemplos de código OpenACC	8	Buscar y/o hacer ejemplos de código OpenACC	8	
Implementar funcionalidades en la librería	50	Implementar la librería	60	

externa de profiling		externa de OpenACC		
Implementar las mismas funcionalidades en Extrae	50	Testear los ejemplos usando la librerÍa externa	8	
Hacer test visualización de OpenACC en Paraver	20			
Sprint 2	**98**	**Sprint 2 : Integración de OpenACC en Extrae**	**102**	Se han incrementado el número de tareas del sprint
Buscar y/o hacer ejemplos de código OpenACC	3	Implementar y editar los ficheros y makefiles de extrae para incorporar openacc	24	
Implementar funcionalidades en la librería externa de profiling	40	Implementar e incorporar ficheros del modelo openacc en extrae	40	
Implementar las mismas funcionalidades en Extrae	40	Implementar ficheros de eventos de OpenACC para paraver.	24	
Hacer test visualización de OpenACC en Paraver	15	Añadir ejemplos de openacc en extrae	6	
		Testear extrae con openacc usando los ejemplos	8	
Sprint 3	**72**	**Sprint 3 : Registrar la actividad de la GPU**	**88**	**Se ha reducido el número de tareas del sprint**
Buscar y/o hacer ejemplos de código OpenACC	2	Investigar cómo registrar la actividad de la GPU	40	
Implementar funcionalidades en la librería externa de profiling	30	Registrar la actividad de GPU : Tiempo de cómputo	40	
Implementar las mismas funcionalidades en Extrae	30	Testear resultados con vistas de paraver	8	
Hacer test visualización de OpenACC en Paraver	10			
Testing	**16**	**Testing**	**16**	
Revisión general de todos los sprints y memoria	16	Testing general del proyecto	16	
Total	**540**		**556**	
Desviación(%)			**2.96**	

Tabla 3: Comparativa entre planificación estimada y real

Ha habido una desviación del 2.96% respecto a las 540 horas planeadas inicialmente.

Se ha hecho un diagrama de Gantt con la planificación estimada en 10.1. Pero para ser pragmáticos y dada la situación emergencia, en la que no se ha podido cumplir con las horas y los días de trabajo. Se ha considerado hacer una tabla con la comparativa entre la planificación

estimada y real, en lugar de diagrama de Gantt, pues la comparativa entre estos diagramas no sería fidedigna.

En general, han habido dos desviaciones respecto a la planificación inicial del proyecto

La primera ha sido la pandemia del covid-19. Esto ha hecho que se tengan que invertir más horas, ya que no es lo mismo hacer un proyecto en una centro de investigación acompañado de personas que conocen el estado del arte perfectamente, y te pueden guiar de forma precisa, respecto a hacerlo desde casa, de forma remota y con la situación de alerta/excepcionalidad.

Además, el estado del arte en este caso es complejo. Luego, el rendimiento del las horas de trabajo se ve claramente afectado de forma negativa. Esto es, cuando no se consigue hacer una tarea, estando en el centro puedes resolverlo de forma rápida y eficaz. Hacerlo remotamente supone una dificultad añadida y una mayor inversión de tiempo. En definitiva, todo se vuelve más complejo de lo que realmente es y el rendimiento de las horas baja.

En segundo lugar, tal como se predijo en las gestión de riesgos de la planificación inicial. Podía darse el caso de que alguna de las funcionalidades que quisiéramos implementar, como por ejemplo, la actividad del device, no se pudiera hacer usando la profiling API. Esto supone a nivel práctico, inversión temporal en hacer investigación sobre cómo se podría/debería solventar esta problemática.

Esto implicó que para poder registrar la actividad del device en Extrae, se tuviera que usar la runtime de CUDA, en vez de la runtime OpenACC. Además, se tubo que aplicar una solución ingeniosa para computar el tiempo que tarda el device en ejecutar la tarea que le manda el host, tal como se indica en 4.3.

En definitiva, las dos desviaciones mencionadas, y el hecho de escoger una nueva estrategia para hacer el proyecto (no incorporar la librería dinámica de forma gradual). Han acabado provocando desviaciones en las tareas, sus horas y las dependencias de las mismas. En consecuencia, desviaciones en la planificación temporal.

.

7.2 Informe de presupuestos

El presupuesto de este proyecto está directamente relacionado con la planificación temporal.

NOTA:

Durante el desarrollo de este proyecto ha habido una pandemia que ha obligado al confinamiento de la población a nivel mundial, y consecuentemente, a la cancelación de muchos planes. Por ese motivo, hay algunas desviaciones en la planificación temporal y presupuestaria del proyecto.

Por otro lado, hay otras desviaciones que no tienen nada que ver con esta situación, y son las típicas desviaciones que pueden surgir durante el desarrollo de un proyecto de estas características.

7.2.1 Coste del personal por actividad (CPA)

En este proyecto habrá esencialmente tres tipos de roles. El sueldo de un director de proyecto de TI es de 42.000€/año. El sueldo de un ingeniero del software es 32.000 €/año. El sueldo de un senior quality assurance de TI es de 24.000€/año. Datos obtenidos de la red social linkedin [6].

En el BSC-CNS, cualquier persona a jornada completa, trabaja al año 1792 hora. Luego, los precios por hora de cada rol son de:

Rol	**Precio (€/h) sin S.S**	**Precio (€/h) con S.S 30%**
Director del proyecto	23.44	30.47
Ingeniero del software	17.86	23.22
Senior quality assurance	13.39	17.41

Tabla 4: Coste de los roles

Con los datos ajustados con la planificación temporal, el coste del personal por actividad (CPA) es de :

Tarea	Horas	#Director de proyecto (h)	#Programador (h)	#Tester (h)	Coste por tarea (€)
GEP					6154.94
Reuniones	30	30	0	0	914.1
Informe del alcance del proyecto	16	16	0	0	487.52
Informe de la planificación temporal	15	15	0	0	457.05
Informe del presupuesto del proyecto	15	15	0	0	457.05
Informe de sostenibilidad del proyecto	15	15	0	0	457.05
Informe de seguimiento	16	16	0	0	487.52
Documento de la memoria del proyecto	80	**80**	**0**	**0**	2437.6
Presentación del proyecto	15	15	0	0	457.05
Trabajo previo					1671.84
Aprender C avanzado y usar el superordenador CTE-POWER	8	0	8	0	185.76
Aprender el paradigma de OpenACC	16	0	16	0	371.52
Aprender Extrae y Paraver a nivel de usuario	8	0	8	0	185.76
Aprender sobre los archivos fuentes de Extrae 3.7.1	40	**0**	**40**	**0**	928.8
Implementación					**6327.08**
Sprint 1 : Implementación de la librería externa					1718.24
Buscar y/o hacer ejemplos de código OpenACC	8	0	8	0	185.76
Implementar la librería externa de OpenACC	60	0	60	0	1393.2
Testear los ejemplos usando la librerÍa externa	8	**0**	**0**	**8**	139.28
Sprint 2 : Integración de OpenACC en Extrae					2321.96
Implementar y editar los ficheros y makefiles de extrae para incorporar openacc	24	0	24	0	557.28
Implementar e incorporar ficheros del modelo openacc en extrae	40	0	40	0	928.8

Implementar ficheros de eventos de OpenACC para paraver.	24	**0**	**24**	**0**	557.28
Añadir ejemplos de openacc en extrae	6	0	6	0	139.32
Testear extrae con openacc usando los ejemplos	8	0	0	8	139.28
Sprint 3 : Registrar la actividad de la GPU					2286.88
Investigar cómo registrar la actividad de la GPU	40	40	0	0	1218.8
Registrar la actividad de GPU : Tiempo de cómputo	40	0	40	0	928.8
Testear resultados con vistas de paraver	8	0	0	8	139.28
Testing					278.56
Testing general del proyecto	16	0	0	16	278.56
Total					14432.42
Desviación (%)					3.63

Tabla 5: Coste del personal por actividad

El coste estimado para la CPA era de 13927.37€ en la planificación inicial (documento GEP). Ahora es de 14432.42€, lo que supone una desviación del 3.63% en los CPA. La justificación es que han habido modificaciones en las tareas y las horas. Estas modificaciones se han justificado en el informe de planificación temporal 7.1

7.2.2 Costes genéricos (CG)

En primer lugar, hay una tabla indicando los costes de los recursos que se pueden amortizar.

Seguidamente tenemos las explicaciones de los costes de elementos como electricidad, internet, instalaciones.

Por último, el coste genérico (CG), será la suma de los costes de los recursos, la electricidad, internet e instalaciones.

Recurso	Coste del recurso (€)	Vida útil (meses)	Amortizaciones
Hardware			
Portatil DELL 7940	2303	48	191.92
Impresora	116.47	48	9.71
Ratón	4.99	48	0.42
Teclado	9.99	48	0.83
Pantalla	127.99	48	10.67
Software			
PGI compiler	1,790.50	12	596.83
Github	28	4	28
Material de oficina			
Silla	68.87	96	2.87
Mesa	173	96	7.21
Libros	193.16	4	193.16
Material vario de oficina	30	12	10
Total	**4845.97**		**1051.62**

Tabla 6: Coste de los recursos

Amortización

La fórmula usada para calcular la amortización es Amortización = coste del recurso * (meses que dura el proyecto / vida útil del recurso). El proyecto dura 4 meses. La vida útil se mide en meses. El total de las amortizaciones es 1051.62€

Otro software:

Otro software que he usado pero que tiene coste nulo, ya que es gratis es :
Editores de código:Geany y Sublime.
Software de ofimática: Google docs y el paquete Libreoffice.
Compiladores: GCC.
Desarrollo y deployment del software: Git
Sistema operativo: Ubuntu 18.04.

Todo este software me ha supuesto un coste de 0€. El software que me ha supuesto un coste (PGI compiler y Github), ya está especificado.

Coste del internet:
El coste medio de internet son 50€/mes. En un mes hay 720 horas. El proyecto dura 540h. Luego, el coste es de internet para este proyecto es (regla de tres) 540/720 * 50 = 37.5€

Desviación: Al hacer el proyecto en casa y no en la oficina. Mi coste de internet son 30€/mes lo que modifica estos cálculos. 540/720*30 = 22.5€

Coste de la electricidad:
La tarifa del kWh es de 0.1290€, en modalidad fija tal como se menciona en [2]. El portátil y el monitor consumen 400W. Usaré el portátil 540h. Luego, 0.1290*400*540/1000=27.86€

Desviación: Usaré el portátil 556 horas, y no 540 horas, de acuerdo con la planificación temporal. Luego, 0.1290*400*556/1000=28.68€

Coste de la instalación
Estamos en un despacho grande con 9 personas trabajando en Barcelona. Tenemos servicio de limpieza y acceso las 24h del dia. El coste de coworking en Barcelona, según la web thespaces es de 275€/mes por persona.
Voy a estar 4 meses haciendo este proyecto en el despacho. Luego, el coste es 275€/mes * 4 meses = 1100€

Desviación: He estado un 1 mes en el despacho y los otros 3 meses trabajando desde casa. Por lo que se ha modificado este coste.

Pago de alquiler 500€/mes. Estoy en la habitación grande que por proporción cuesta 100€. Luego, el coste total real de la instalación es : 275€/mes *1 mes + 100€/mes*3 meses = 575€.

Coste del superordenador

En este proyecto se va a usar un superordenador para desarrollarlo, almacenarlo y testearlo. Lo que repercute en el presupuesto. No se entra en detalle sobre el coste de este recurso, ya que es compartido y repercute de forma indirecta en los presupuestos de este proyecto.

Luego, el coste genérico del proyecto es

El coste genérico es :

Concepto	**Coste (€)**
Coste de los recursos	1051.62
Internet	22.5
Electricidad	28.68
Instalación	575
Total gastos generales	**1677.8**

Tabla 7: Resumen costes genérico del proyecto

Por último, el coste genérico es la suma del coste de los recursos, internet, electricidad y las instalaciones. Es decir, costes directos más indirectos.

7.2.3 Contingencias

Se ha usado un valor del 15% como porcentaje para el cómputo de las contingencias. Como justificación de este porcentaje podemos decir que es el habitual en proyectos de este tipo.

Luego, el coste de las contingencias es : (CPA+CG)*0.15 (euros), es decir:

Contingencias:	**2416.53**

7.2.4 Imprevistos

En el informe GEP se detectaron tres imprevistos que podían surgir.

El primero era tener que hacer un cambio importante en la forma de hacer la implementación. Es decir, tener que hacer la instrumentación a nivel de compilador en vez de usar la profiling API de OpenACC, tal como lo hace Score-P. Para lo cual hubiésemos necesitado más tiempo. En consecuencia, dinero.

El segundo imprevisto que podíamos tener era tener un diseño ineficiente y tener que intervenir tiempo para hacerlo eficiente.

El tercero era cambiar el entorno en el caso de fallo del superordenador. Para lo cual nos haría falta dinero (para comprar un ordenador con gráfica dedicada), y tiempo para adaptar el entorno.

En este caso, afortunadamente, no han habido ninguno de los imprevistos que habíamos contemplado. Con lo cual, nos ahorramos el coste de los imprevistos 1325.04 (calculado en el informe GEP).

7.2.5 El total de los presupuestos

FInalmente, la suma de (CPA+CG+Contingencias+Imprevistos) es el presupuesto de este proyecto. En la siguiente tabla resumen tenemos el coste global del proyecto.

Concepto	Coste (€)
Total CPA	**14432.42**
Total CG	**1677.8**
Contingencias	**2416.53**
Imprevistos	**-1325.04**
Total sin IVA	**17201.71**
Total (IVA 21%)	**20814.07**

Tabla 8: Coste total del proyecto

El coste total del proyecto es:

Total (IVA 21%)	**20814.07**

7.2.6 Viabilidad y control de los presupuestos

Uno de los motivos para escoger la metodología Agile para realizar este proyecto fue, que el hecho de hacerlo por sprints, te permite hacer una reevaluación de las horas que has dedicado en cada iteración a cada tarea, y consecuentemente, te permite controlar el gasto en recursos y tiempo.

En primer lugar, en el CPA ha habido una desviación del +3.63%, ya que se han tenido que modificar las horas de dedicación de algunas tareas.

En segundo lugar, han habido modificación de algunos costes como por ejemplo, los costes de internet y de las instalaciones. Estos son debidos a la pandemia, ya se ha tenido que trabajar desde casa.

En tercer lugar, el plan de contingencias ha sido del 15% (típico para este tipo de proyectos). No ha habido ningún imprevisto, lo que ha supuesto un ahorro de 1325.04€.

Finalmente, el coste total sin IVA es de **17201.71**€ , frente a los **19961.45€** de la planificación inicial. Lo que supone un ahorro del 13.82%, respecto al presupuesto inicial. El ahorro se ha conseguido controlando el hecho de no tener los imprevistos mencionados en la planificación inicial, y reevaluando las horas por cada sprint de desarrollo que se hacía.

7.3 Informe de sostenibilidad

El informe de sostenibilidad del proyecto esta hecho en base a la matriz de sostenibilidad

		PPP	Vida Útil	Riesgos
Ambiental	I	¿Has estimado el impacto ambiental que tendrá la realización del proyecto? ¿Te has planteado minimizar el impacto, por ejemplo, reutilizando recursos?	¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará ambientalmente tu solución a las existentes?	
	F	¿Has cuantificado el impacto ambiental de la realización del proyecto? ¿Qué medidas has tomado para reducir el impacto? ¿Has cuantificado esta reducción?	¿Qué recursos estimas que se usarán durante la vida útil del proyecto? ¿Cuál será el impacto ambiental de estos recursos?	¿Podrían producirse escenarios que hiciesen aumentar la huella ecológica del proyecto?
		Si hicieras de nuevo el proyecto, ¿podrías realizarlo con menos recursos?	¿El proyecto permitirá reducir el uso de otros recursos? ¿Globalmente, el uso del proyecto mejorará o empeorará la huella ecológica?	
Económico	I	¿Has estimado el coste de la realización del proyecto (recursos humanos y materiales)?	¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará económicamente tu solución a las existentes?	
	F	¿Has cuantificado el coste (recursos humanos y materiales) de la realización del proyecto? ¿Qué decisiones has tomado para reducir el coste? ¿Has cuantificado este ahorro?	¿Qué coste estimas que tendrá el proyecto durante su vida útil? ¿Se podría reducir este coste para hacerlo más viable?	¿Podrían producirse escenarios que perjudicasen la viabilidad del proyecto?
		¿Se ha ajustado el coste previsto al coste final? ¿Has justificado las diferencias (lecciones aprendidas)?	¿Se ha tenido en cuenta el coste de los ajustes/actualizaciones/reparaciones durante la vida útil del proyecto?	
Social	I	¿Qué crees que te va a aportar a nivel personal la realización de este proyecto?	¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará socialmente (calidad de vida) tu solución a las existentes?	
			¿Existe una necesidad real del proyecto?	
	F	¿La realización de este proyecto ha implicado reflexiones significativas a nivel personal, profesional o ético de las personas que han intervenido?	¿Quién se beneficiará del uso del proyecto? ¿Hay algún colectivo que puede verse perjudicado por el proyecto? ¿En qué medida?	¿Podrían producirse escenarios que hiciesen que el proyecto fuese perjudicial para algún segmento particular de la población?
			¿En qué medida soluciona el proyecto el problema planteado inicialmente?	¿Podría crear el proyecto algún tipo de dependencia que dejase a los usuarios en posición de debilidad?

Figura 20: Matriz de sostenibilidad

La figura se ha obtenido de [1]. El proyecto se ha desarrollado y testeado en la máquina CTE-POWER del BSC-CNS. Además, es un proyecto para ser usado en esta misma máquina.

El informe de sostenibilidad está directamente relacionado con el consumo de esta máquina, junto en con el informe de presupuestos y gestión temporal del proyecto.

Se va a analizar la matriz de sostenibilidad por filas. Se contesta a las preguntas que se plantean en la matriz (color azul, F), pero en el contexto de este proyecto.

7.3.1 Impacto ambiental

Proyecto puesto en producción: consumo del diseño

El hecho de usar un superordenador puede afectar drásticamente al medio ambiente por su elevado consumo eléctrico, ya que el hecho de producir electricidad, potencialmente genera gases de efecto contaminante, que van a la atmósfera produciendo consecuencias fatídicas para el medio ambiente.

En concreto para este proyecto, al haber usado el clúster CTE-POWER para aprender (alcanzar el estado del arte), desarrollar y testearlo ha supuesto un coste de 354 horas. El mayor gasto de electricidad que hemos tenido es el del superordenador. El dato exacto de lo que consume el superordenador (Khw) no lo sabemos. Es por eso que no podemos determinar, en exactitud el coste energético y la generación de residuos.

Este proyecto requería de un ordenador con gráfica dedicada para poder hacerlo. Todo (trabajo previo, desarrollar y testear el proyecto) se ha tenido que hacer en el superordenador, ya que el BSC-CNS no ofrece a sus empleados ordenadores con gráficas dedicadas.

Si se hiciera el proyecto de nuevo, se recomienda el uso de ordenadores personales con gráficas dedicadas. De esta forma, solamente tendremos que usar el superordenador para la fase de testeo (16 horas en este proyecto). Lo que reduciría el consumo del diseño en un (1-(16/354))*100 = 95.48%.

Consecuentemente, en la fase de puesto en producción del proyecto, el consumo de electricidad y la generación de residuos (debido a la producción de electricidad), se reduciría un 95.48%.

Vida útil: huella ecológica

El proyecto está hecho para ser usado en la CTE-POWER. No sabemos con exactitud la vida útil del proyecto, ya que depende del uso que se le de en el entorno del BSC-CNS. Es por eso que no sabemos determinar con exactitud el impacto ambiental.

Lo que sí sabemos es que el impacto ambiental del proyecto depende directamente de lo que consuma CTE-POWER y los residuos que se generen debido a su uso.

Riesgos ambientales

El impacto ambiental ha sido negativo por el hecho de tener que usar la CTE-POWER para todo (estado del arte, desarrollo y testing). Usando CTE-POWER exclusivamente para el testing, hubiésemos ahorrado en un 95.48%, el impacto ambiental del proyecto.

7.3.2 Impacto económico

Proyecto puesto en producción: Factura

Uno de los motivos para escoger la metodología Agile para realizar este proyecto ha sido el hecho de hacerlo por sprints, te permite hacer una reevaluación de las horas que se han dedicado en cada iteración a cada tarea, y consecuentemente, te permite controlar el gasto en recursos y tiempo. Es por eso que, tal y como se especifica en la sección 7.2.6 , ha habido un ahorro del 13.82%. respecto al presupuesto inicial.

El ahorro se ha conseguido controlando el hecho de no tener los imprevistos mencionados en la planificación inicial, y reevaluando las horas por cada sprint. De esta forma hemos conseguido reducir el uso del superordenador, y consecuentemente reducimos la huella ambiental del mismo.

No tuvimos los dos primeros imprevistos indicados en 7.1.4, por lo que nos hemos ahorrado 230+50=280 horas de uso de la CTE-POWER. Por lo que hemos ahorrado el coste proporcional al uso de 280 del clúster CTE-POWER.

Vida útil: plan de viabilidad

El proyecto está pensado para ser usado en la CTE-POWER, por lo que el coste en la vida útil del proyecto vendrá determinado por lo que cueste, en recursos y dinero, el uso de la CTE-POWER (electricidad, mantenimiento etc..). No podemos determinar este dato de forma exacta.

Riesgos económicos

El mayor riesgo que tiene este proyecto para no ser viable económicamente es que en el entorno del BSC-CNS, se deje de usar la tecnología OpenACC en un futuro. Si esto llegase a pasar, entonces este proyecto perdería el sentido. Por lo que sería económicamente inviable. Supondría una pérdida de 20814.07€, ya que es el coste total del proyecto.

7.3.3 Impacto social

Proyecto puesto en producción: impacto personal

La puesta en producción del proyecto me aporta conocimientos (técnicos y no técnicos) que, potencialmente, podría usar a favor de esta sociedad.

He aprendido que durante la realización de cualquier proyecto hay que valorarlo en sus tres vertientes principales. La vertiente económica, la ambiental y la social. Todo ello, con la finalidad de ir hacia mejor en todos los aspectos, y minimizar de esta forma, las posibles consecuencias negativas que la realización de cualquier proyecto.

Vida útil: impacto social

Analizando objetivamente, a nivel social, de momento, la realización de este proyecto tiene un impacto pequeño. Pero en un futuro esto puede cambiar, todo ello dependerá del éxito de la tecnología OpenACC en el entorno del BSC-CNS. En la medida en la que esta tecnología triunfe, el proyecto tendrá una vida útil mayor o menor.

El proyecto tendrá repercusión en todas la personas del BSC-CNS que usen la herramienta Extrae, y que usen la tecnología OpenACC en sus aplicaciones. Esto es así , ya que la tecnología OpenACC se usa en muchos entornos en los que hay tareas de computación gráfica (imágenes en medicina , clima , simulaciones etc..).

Riesgos sociales

No se detectan riesgos sociales negativos derivados de la realización de es proyecto.

8 Bibliografía

[1]Climent, J. *et al.* (2018) «El informe de sostenibilidad del Trabajo de Fin de Grado del área de las ingenierías», 16(2)), p. 75.

[2]*Consulta el precio diario de la luz (€/kWh): tarifas y comparativa* (sin fecha) *tarifasgasluz.com*. Disponible en: https://tarifasgasluz.com/faq/precio-kwh (Accedido: 23 de junio de 2020).

[3]Evans, J. *LD_PRELOAD is super fun. And easy!*, *Julia Evans*. Disponible en: https://jvns.ca/blog/2014/11/27/ld-preload-is-super-fun-and-easy/ (Accedido: 23 de junio de 2020).

[4]*Extrae* | *BSC-Tools*. Disponible en: https://tools.bsc.es/extrae (Accedido: 23 de junio de 2020).

[5]*Homepage* | *OpenACC* . Disponible en: https://www.openacc.org/ (Accedido: 23 de junio de 2020).

[6]*LinkedIn Salary - Descripción general* | *Ayuda LinkedIn Ayuda LinkedIn*. Disponible en: https://www.linkedin.com/help/linkedin/answer/85616/linkedin-salary-descripcion-general?lang=es (Accedido: 23 de junio de 2020).

[7]*NVIDIA CUDA Library: Event Management* . Disponible en: http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/online/group__CUEVENT.html (Accedido: 23 de junio de 2020).

[8]*Paraver: a flexible performance analysis tool* | *BSC-Tools* . Disponible en: https://tools.bsc.es/paraver (Accedido: 23 de junio de 2020).

[9]*Performance Analysis Tools: Details and Intelligence* | *BSC-Tools* .Disponible en: https://tools.bsc.es/ (Accedido: 23 de junio de 2020).

[10]«PGI Release Notes Version 20.1 for x86 CPUs» . Disponible en: https://www.pgroup.com/resources/docs/20.1/x86/pgi-release-notes/index.htm (Accedido: 23 de junio de 2020).

[11]*Support Knowledge Center @ BSC-CNS* . Disponible en: https://www.bsc.es/user-support/power.php (Accedido: 23 de junio de 2020).

[12]*The OpenACC Application Programming Interface* (5 November, 2019). https://www.openacc.org/. Disponible en: https://www.openacc.org/sites/default/files/inline-images/Specification/OpenACC.3.0.pdf.

9 Anexo

9.1 Diagrama de Gantt de la planificación estimada

gantter™

Project Name **TFG_gantt**

		Nombre	Duración	Inicio	Fin	Predecesoras	Recursos	Custom 1	Custom 2	Custom 3	Custom 4	Custom 5
1		Reuniones	80.5días	12/02/2020	03/06/2020		Lau,German,Ali,J					
2		**⊟GEP**	**88.5días**	**17/02/2020**	**18/06/2020**							
3		Informe del alcance del proyecto	5días	17/02/2020	24/02/2020							
4		Informe de la planificación temporal del proyecto	5.5días	24/02/2020	02/03/2020	3						
5		Informe del presupuesto del proyecto	5días	03/03/2020	09/03/2020	3,4						
6		Informe de sostenibilidad del proyecto	5días	03/03/2020	09/03/2020	3,4						
7		Memoria del proyecto	77días	17/02/2020	02/06/2020							
8		Presentación del proyecto	11.5días	03/06/2020	18/06/2020	7						
9		**⊟Trabajo previo**	**22.5días**	**12/02/2020**	**13/03/2020**							
10		Aprender lenguaje C avanzado	5días	12/02/2020	19/02/2020		Ali,Dell[1]					
11		Aprender el paradigma de OpenACC	6.5días	18/02/2020	26/02/2020							
12		Aprender a usar el superordenador cte-power	5.5días	25/02/2020	03/03/2020							
13		Aprender a usar Extrae	5.5días	28/02/2020	06/03/2020							
14		Aprender a usar paraver	5.5días	06/03/2020	13/03/2020							
15		**⊟Implementación**	**69.5días**	**02/03/2020**	**05/06/2020**		**Ali,Dell[1],CTE-P**					
16		**⊟Sprint 1**	**23días**	**02/03/2020**	**01/04/2020**		**Ali,Dell[1],CTE-P**					
17		Buscar y/o hacer ejemplos de código OpenACC	4días	02/03/2020	05/03/2020							
18		Implementar funcionalidades en la librería externa de profilin	6.5días	06/03/2020	16/03/2020	17						
19		Implementar las mismas funcionalidades en Extrae	8.5días	16/03/2020	26/03/2020	18						
20		Hacer test de visualización de OpenACC en Paraver	3.5días	27/03/2020	01/04/2020	19						
21		**⊟Sprint 2**	**23.5días**	**02/04/2020**	**05/05/2020**		**Ali,Dell[1],CTE-P**					
22		Buscar y/o hacer ejemplos de código OpenACC	4.5días	02/04/2020	08/04/2020	20						
23		Implementar funcionalidades en la librería externa de profilin	5.5días	08/04/2020	15/04/2020	22						
24		Implementar las mismas funcionalidades en Extrae	10.5días	16/04/2020	30/04/2020	23						
25		Hacer test de visualización de OpenACC en Paraver	3días	30/04/2020	05/05/2020	24						
26		**⊟Sprint 3**	**19días**	**05/05/2020**	**01/06/2020**		**Ali,Dell[1],CTE-P**					
27		Buscar y/o hacer ejemplos de código OpenACC	1.5días	05/05/2020	06/05/2020	25						
28		Implementar funcionalidades en la librería externa de profilin	5días	07/05/2020	13/05/2020	27						
29		Implementar las mismas funcionalidades en Extrae	9.5días	14/05/2020	27/05/2020	28						
30		Hacer test de visualización de OpenACC en Paraver	3días	27/05/2020	01/06/2020	29						
31		Revisión general de todos los sprints y revisión de la memoria	2.5días	03/06/2020	05/06/2020	20,25,30,7	Lau,German,Ali,J					

PNG Generated On: 20/6/2020 19:49:41

Figura 21: Listado de tareas del diagrama de Gantt de la planificación estimada

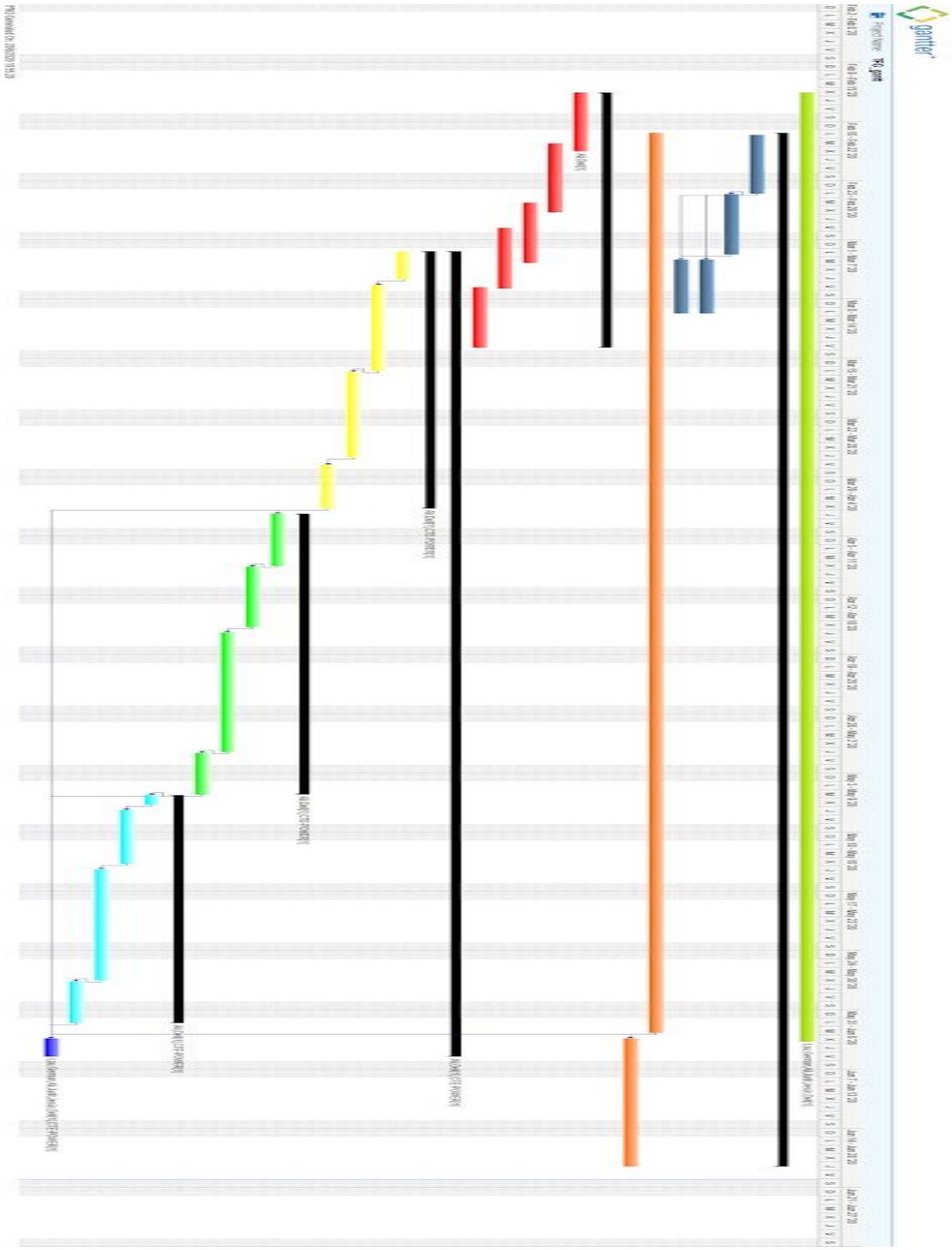

Figura 22: Timeline con las de tareas del diagrama de Gantt de la planificación estimada

9.2 Instalar Extrae para instumentar OpenACC

Como prerrequisito debemos instalar las librerías estrictamente necesarias para la instalación de Extrae.
https://github.com/bsc-performance-tools/extrae/ . En la máquina CTE-POWER ya están debidamente instaladas estas librerías.

Seguidamente debemos debemos ejecutar el ./bootstrap para obtener el fichero ./configure.

Una vez tenemos el fichero ./configure debemos indicarle el lugar de instalación de Extrae, y el lugar donde tenemos instaladas las librerías esenciales de Extrae en nuestro sistema.

Por último, debemos indicarle la ruta de instalación del compilador PGI para OpenACC. En el caso de la máquina CTE-POWER es:
/gpfs/apps/POWER9/PGI/2020-201/linuxpower/20.1/

Extrae es capaz de soportar la versión 3.0 del estándar OpenACC, es por ello que debemos utilizar el compilador PGI 20.1 o superior, ya que és el único compilador en el mundo , por ahora, que implementa el estándar 3.0 de OpenACC.

Un ejemplo de instalación en la máquina CTE-POWER del BSC-CNS https://www.bsc.es/user-support/power.php

```
>./bootstrap
>./configure --prefix=/home/bsc41/bsc41636/extrae-funcional-instalacion
--with-mpi=/gpfs/apps/POWER9/OPENMPI/4.0.1/GCC-8.3.0/ --with-unwind=/gpfs/apps/POWER/LIBUNWIND/1.2
--with-papi=/gpfs/apps/POWER9/PAPI/5.6.0 --without-dyninst
--with-openacc=/gpfs/apps/POWER9/PGI/2020-201/linuxpower/20.1/
>make -j
>make install
```

Código 12: Instalación de Extrae con capacidad para instrumentar OpenACC en el clúster CTE-POWER del centro de supercomputación BSC-CNS