PROBLEMAS COMPUTACIONALES ASOCIADOS A LA CONSTRUCCIÓN DE MODELOS DE SIMULACIÓN BASADOS EN AUTÓMATAS CELULARES EN PARALELO. CASO DE ESTUDIO: EVALUACIÓN DE AMENAZAS ASOCIADAS A FLUJOS DE LAVA VOLCÁNICA COMO FLUÍDO BINGHAM

SERGIO AUGUSTO GÉLVEZ CORTÉS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS
BUCARAMANGA
2015

PROBLEMAS COMPUTACIONALES ASOCIADOS A LA CONSTRUCCIÓN DE MODELOS DE SIMULACIÓN BASADOS EN AUTÓMATAS CELULARES EN PARALELO. CASO DE ESTUDIO: EVALUACIÓN DE AMENAZAS ASOCIADAS A FLUJOS DE LAVA VOLCÁNICA COMO FLUIDO BINGHAM

SERGIO AUGUSTO GÉLVEZ CORTÉS

Trabajo de grado para optar al título de
Magister en Ingeniería de Sistemas

Director
CARLOS JAIME BARRIOS HERNÁNDEZ
Doctor en Informática

Codirector
HUGO HERNANDO ANDRADE SOSA
Magíster en Informática

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS
BUCARAMANGA
2015

Dedicatoria

A mi familia, siempre una apoyo incondicional. A ellos entrego el reconocimiento de este logro.

A mis amigos, sean ellos de antes, de durante el proceso, de la academia, o de fuera de esta. Pude hacer nuevos amigos durante el proceso y no olvido tampoco los de siempre. Gracias.

Y finalmente, y como se puede aprender de muchas fuentes en la vida, dedico mi trabajo a la selección alemana de futbol campeona del mundial del 2014. En un momento de mi vida en que estaba aburrido del desorden y la improvisación reinante en mi sociedad, llegan ellos y demuestran una verdad que todos conocemos pero que hace falta que nos recuerden: Que el trabajo ordenado, disciplinado y sistemático vence todos los obstaculos en el largo plazo. Que los humanos unimos esfuerzos para superar retos mas grandes que todos nosotros, y que solo la planeación, el orden y el estudio de la técnica correcta y el uso del conocimiento adquirido permiten superar esas dificultades. Lo dedico a ellos, que en 14 años lograron construir de su más grande fracaso un grupo ganador, con la esperanza de que esto inspire de ahora en adelante mi trabajo, a ver si puedo lograr ser más disciplinado y aprovechar todo mi potencial.

Agradecimientos

A mi familia por su apoyo durante este proceso. Agradecimientos especiales a mi hermana, Adriana Lucia Gélvez Cortés, por su colaboración en la revisión del manuscrito. Pocos tienen una familia tan comprensiva y sobretodo tan fanática de la decisión de un ingeniero de encaminarse por la investigación, especialmente con la economía del pais como estuvo en este tiempo. Gracias de nuevo por el apoyo, pero muchas más por lo incondicional.

A mis directores, los profesores Carlos Jaime Barrios Hernández y Hugo Hernando Andrade Sosa, por la infinita confianza que depositaron en mi trabajo. No solo me brindaron una autonomía que me permitió aprender valiosas lecciones y ahora me permite sentirme más preparado que nunca para continuar por el sendero de la investigación, sino que apoyaron mis decisiones y tuvieron paciencia con los reveses que sufrí durante el proceso. El contacto con la academia francesa permitido por el profesor Carlos Jaime fue invaluable también en mi transformación personal como futuro investigador.

A mis compañeros de grupo de investigación, de maestría, y de la maestría en Ingeniería Química. Uno en ningún proceso está solo, y mucho menos con gente como ustedes. Sin su apoyo este viaje muy probablemente hubiera sido truncado en el primer escollo.

Y finalmente, a las comunidades en línea, llámense comunidades de software libre, comunidades académicas, repositorios de conocimientos, etc. Es este un momento en la historia en el que se puede acceder fácilmente a cantidades enormes de conocimiento. Tener dicho conocimiento disponible es una ayuda invaluable que no debe pasar incógnita.

Los resultados de los experimentos presentados en esta publicación, fueron obtenidos usando la plataforma GridUIS-2, desarrollada por el Centro de Supercomputación y Cálculo Científico de la Universidad Industrial de Santander (SC3UIS). Esta acción es soportada por la Vicerrectoría de Investigación y Extensión de la UIS (VIE-UIS) y diferentes grupos de investigación de la universidad. (http://www.sc3.uis.edu.co)

TABLA DE CONTENIDO

TABLA DE FIGURAS

LISTA DE TABLAS

Resumen

TÍTULO: PROBLEMAS COMPUTACIONALES ASOCIADOS A LA CONSTRUCCIÓN DE MODELOS DE SIMULACIÓN BASADOS EN AUTÓMATAS CELULARES EN PARALELO. CASO DE ESTUDIO: EVALUACIÓN DE AMENAZAS ASOCIADAS A FLUJOS DE LAVA VOLCÁNICA COMO FLUIDO BINGHAM[1]

AUTOR: SERGIO AUGUSTO GÉLVEZ CORTÉS[2]

PALABRAS CLAVE: SIMULACIÓN, COMPUTACIÓN DE ALTO DESEMPEÑO, GPGPU, AUTÓMATAS CELULARES, FLUJOS DE LAVA.

DESCRIPCIÓN:

Existen procesos naturales que son de interés debido a su complejidad e impacto sobre la vida humana, y las erupciones volcánicas son de los ejemplos más fáciles de identificar. Un factor importante en relación al fenómeno de los flujos de lava volcánica de especial consideración es la complejidad matemática y computacional de los fenómenos. Al hablar de esta última se involucra a la computación de alto desempeño, y en esta, la eficiencia de las simulaciones es un tema crucial; es ahí donde este trabajo tiene su motivación: El construir un modelo de simulación, y caracterizarlo en su ejecución para determinar su eficiencia.

El primer paso fue la creación del modelo partiendo del conocimiento del fenómeno, en este caso usando las teorías reológicas. Se usaron las ecuaciones de Navier-Stokes para fluídos tipo Bingham. Además, el método de autómatas celulares planteado se usó para simplificar la geometría del problema y de esa manera reducir la complejidad en la formulación matemática del mismo. Después se construyó una versión paralela del programa compatible con la infraestructura de cálculo avanzado de la universidad. De esta manera se usó como modelo de programación la GPGPU mediante CUDA©.

Finalmente, se crearon varios programas en paralelo que usaban varios mapeos de procesos sobre elementos de cómputo y sobre las diferentes jerarquías de memoria. Se realizó un experimento para determinar características del comportamiento de la ejecución de los programas. Con los datos obtenidos se determinó que el mapeo con una GPU y memoria principal más la del dispositivo es el más eficiente. Al final, se proponen caminos posibles para la continuación del trabajo, desde diversas perspectivas, tanto computacionales como del fenómeno en sí.

[1] Trabajo de Investigación.

[2] Facultad de Ingenierías Físico-mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Carlos Jaime Barrios Hernández, Doctor en Informática. Codirector: Hugo Hernando Andrade Sosa, Magister en Informática.

ABSTRACT

TITLE: COMPUTATIONAL PROBLEMS ASSOCIATED TO SIMULATION MODEL CONSTRUCTION BASED ON PARALLEL CELLULAR AUTOMATA. CASE STUDY: VOLCANIC LAVA FLOW HAZARD ASSESSMENT BASED ON LAVA AS A BINGHAM FLUID [3]

AUTHOR: SERGIO AUGUSTO GÉLVEZ CORTÉS [4]

KEYWORDS: SIMULATION, HIGH PERFORMANCE COMPUTING, GPGPU, CELLULAR AUTOMATA, LAVA FLOWS

DESCRIPTION:

There are natural processes that are of interest due to their complexity and impact on human life, and volcanic eruptions are among the easiest examples to identify. A very important factor around lava flow phenomena is their mathematical and computational complexity. When discussing the latter, high performance computing should be involved, and with it, efficiency in the simulations becomes a key topic; In that scenario this work has its motivation: The construction of a simulation model and the characterisation of its execution to determine its eficiency.

The first step was the construction of a model from the knowledge about the phenomenon, in this case, rheology and fluid dynamics. The Navier-Stokes equations for a Bingham Fluid were used, as well as a Cellular Automata Method to simplify the problem's geometry and thus reducing the complexity in its mathematical formulation. After that, a parallel implementation of the model, compatible with the university's high performance computing infrastructure, was constructed; as such, the programming model was GPGPU using CUDA©

Finally, several parallel programs were created, each one using several process mappings over the processing elements and over the different memory hierarchies. An experiment to determine characteristics of the behaviour of the programs' execution was performed. With the data obtained, It was determined that the mapping using one GPU and RAM memory plus VRAM is the most efficient one. In the end of the text, several ways to continue the work are proposed from several different perspectives: from the computing point of view as well as from the phenomenon itself.

[3] Master's degree research work.

[4] Department of Physical-mechanical Engineering. School of Systems Engineering and Computer Science. Advisor: Carlos Jaime Barrios Hernández, Ph.D. in Computer Science. Co-advisor: Hugo Hernando Andrade Sosa, M.Sc. in Computer Science.

INTRODUCCIÓN

Existen procesos naturales que son de interés debido a que conllevan riesgos para las poblaciones humanas, y las erupciones volcánicas son de los ejemplos más fáciles de identificar. Las erupciones volcánicas y los subsecuentes flujos de lava son peligrosos para la vida y la infraestructura humana que reside cerca de las áreas de influencia de los volcanes. Debido a esto, caracterizar las posibles amenazas, siendo amenaza el riesgo cuantificado en vidas, recursos, etc [2], es un problema de gran interés. Dos factores importantes en relación al fenómeno de las erupciones y los flujos de lava volcánicos son de especial consideración: La complejidad matemática y computacional de los fenómenos y los modelos que los simulan y la escala de tiempo de los procesos eruptivos.

El modelado matemático y computacional de los fenómenos geofísicos constituye uno de los problemas que más recursos computacionales usa, debido a lo complejo de la formulación y a las escalas temporales y espaciales que se estudian. Respecto a lo primero, en el caso particular de los flujos de lava volcánica, se usan las ecuaciones de Navier-Stokes para fluídos no newtonianos [3, 4, 5], las cuales son bastante complejas. El método de autómatas celulares planteado se usa para simplificar la geometría del problema [3] y de esa manera reducir la complejidad en la formulación matemática del mismo. Por otra parte, para que tengan sentido estas simulaciones, deben hacerse sobre áreas de tamaño significativo, del orden de metros cuadrados, hectáreas, acres, kilómetros cuadrados, etc, y en intervalos de tiempos que pueden ir de minutos hasta horas. Estos intervalos de tiempo y estas áreas pueden demandar gran cantidad de recursos de cómputo. Finalmente, el nivel de detalle deseado también puede hacer crecer significativamente el tamaño del problema.

La escala de tiempo de estos fenómenos, que es una escala de tiempo geológico, hace poco interesante el uso de métodos de modelado basados en estadísticas ya que no habrían datos suficientes para describir el comportamiento del fenómeno, esto por la incapacidad de registrarlos en intervalos de tiempo tan altos, que pueden preceder por mucho a la aparición del hombre en la tierra. Además estos métodos nada pueden decir de lugares donde no se hayan registrado eventos de erupciones, aún si se sabe que hay un cráter o un volcán inactivo cerca. El modelado estructural, basado en el conocimiento, permite obtener resultados de posibles sucesos volcánicos aún sin que se hayan registrado anteriormente en la zona. Esto permite apoyar el proceso de evaluación de amenazas asociadas a flujos de lava volcánica de una manera más adecuada.

Volviendo a la temática de la complejidad del modelo, la eficiencia y la posibilidad de correr estas simulaciones es crucial, y ahí es donde este trabajo tiene su motivación: El construir y caracterizar un modelo de simulación, en su ejecución, para determinar su

eficiencia. Este tipo de fenómenos no se pueden correr en la práctica sin infraestructuras de alto desempeño, entonces definir si es posible correrlos de manera eficiente en una de ellas es crucial para las expectativas de generar, a largo plazo, modelos que apoyen la evaluación de amenazas asociadas a los flujos de lava.

El primer paso es la creación del modelo partiendo del conocimiento del fenómeno, en este caso usando las teorías reológicas. El trabajo no plantea modificar nada de la formulación del modelo, no pretende aportar al estado del arte de la geofísica, de la vulcanología. Lo que si pretende es reconstruir el modelo apropiándose del conocimiento, de tal manera que se pueda explicar del fenómeno. Para la evaluación del modelo construido se usan otros modelos, ya que es difícil desde la computación llegar al nivel de precisión para compararlos con datos reales sin apoyo de expertos en el tema de la vulcanología, con los que no se cuenta. Después se deberá construir una versión paralela del programa compatible con la infraestructura de cálculo avanzado de la universidad. El modelo de programación a usar depende de esta infraestructura.

Finalmente, cuando se haya implementado la versión en paralelo, se deben implementar programas que usen varios mapeos de procesos sobre elementos de cómputo y sobre las diferentes jerarquías de memoria [1]. Se realizará un experimento, que será descrito en este texto, para determinar características del comportamiento de la ejecución de los programas. Con los datos que se obtendrán se deberá determinar cual mapeo es más eficiente y porque. Ese resultado permitirá la continuación a futuro de la creación de modelos que apoyen la evaluación de amenazas asociadas a flujos de lava volcánica. Al final se proponen caminos posibles para la continuación del trabajo, desde diversas perspectivas.

1. CONTEXTO DE INVESTIGACIÓN

1.1. DESCRIPCIÓN DE LA SITUACIÓN PROBLEMA

Los autómatas celulares han sido utilizados como herramienta de modelado y simulación en diversos campos del conocimiento debido a varias características, entre estas: la capacidad que tienen de emular el comportamiento de un sistema a nivel general partiendo del conocimiento que se tiene sobre las interacciones de sus componentes, y su capacidad de modelar fenómenos en los que la distribución espacial del fenómeno es clave.

La posibilidad de realizar un proceso de modelado estructural (basado en teorías científicas y no en datos históricos) no es la única característica de interés que presentan los autómatas celulares. Su naturaleza intrínsecamente paralela los hace muy llamativos como herramienta para la construcción de modelos de simulación, ya que esta facilita el proceso de implementación del modelo mediante computación en paralelo. Prácticamente todos los fenómenos naturales que son de interés para la simulación se benefician de la posibilidad de utilizar arquitecturas paralelas, ya que a medida que se quiere simular con mayor detalle, las representaciones crecen en escala, y así crecen los requerimientos de cómputo.

El problema físico que inspiró el estudio de los autómatas celulares en paralelo en este caso es el de caracterizar los flujos de lava volcánica producto de una erupción con el fin de evaluar amenazas en las zonas aledañas; esto resulta interesante pues un modelo estructural es más consecuente con la noción de tiempo geológico que un modelo estadístico. Para el modelado del flujo de lava se ha utilizado la teoría de fluidos newtonianos generalizados, clasificando la lava como un fluido Bingham [3, 5]. Cabe mencionar que la escala necesaria para construir una representación de una erupción adecuada para una evaluación de amenazas hace necesaria la utilización de implementaciones en paralelo.

Hay dos problemas recurrentes que se presentan en el diseño e implementación de algoritmos para computación en paralelo: El balanceo de carga y el mapeo de los procesos en las unidades de procesamiento. Estos problemas son estudiados constantemente debido a que las implementaciones en paralelo son altamente dependientes de la arquitectura a utilizar y del algoritmo mismo que se quiere implementar. Para el caso de estudio, resulta necesario estudiar esos problemas para poder construir un modelo que no solo sea adecuado sino también eficiente: de nada sirve un modelo bien concebido si no puede obtener resultados en un tiempo razonable. Al final, la experimentación con estos problemas computacionales resulta de suma importancia para la construcción del modelo de simulación para el fenómeno de interés. También cabe recordar que como el modelo está basado en una teoría reológica y de mecánica de fluidos, el investigar estos problemas pa-

ra este caso en particular aporta también a otros problemas relacionados que compartan esa teoría física como base de su descripción.

Se genera al final una serie de interrogantes de investigación claros: ¿Cuál es el mapeo adecuado para implementar un modelo de simulación para el fenómeno de interés? ¿Cómo se debe distribuir la carga de procesamiento para garantizar eficiencia? Además de ¿Cuál es una manera apropiada para construir un modelo para el fenómeno y cuáles problemas pueden surgir en ese proceso de construcción?

1.2. OBJETIVOS

1.2.1. Objetivos Generales

Caracterizar el problema del mapeo de procesos para la paralelización de un modelo basado en autómatas celulares, tomando como caso de estudio la identificación y evaluación de amenazas asociadas a flujos de lava volcánica.

1.2.2. Objetivos Específicos

- Construir un modelo, basado en autómatas celulares, de flujos de lava volcánica usando la teoría de mecánica de fluidos (fluidos tipo Bingham), identificando problemas conceptuales emergentes en el proceso de modelado mismo y generando alternativas para su posterior solución. El modelo puede estar orientado a la evaluación de amenazas asociadas a los flujos, a través de la determinación de las zonas que pueden ser afectadas.
- Establecer un esquema de mapeo para la paralelización del modelo teniendo en cuenta las características computacionales exhibidas por el fenómeno.
- Proponer una arquitectura paralela para el problema e implementar el modelo construido.

1.3. ALCANCE DEL PROYECTO

El trabajo de investigación tiene como objetivo no solo mostrar que es posible construir el modelo del fenómeno basado en autómatas celulares, lo cual ya ha sido realizado en el estado del arte, sino responder el interrogante de como hacerlo de manera más eficiente, haciendo uso de las herramientas de computación de alto desempeño, teniéndolas en cuenta desde la concepción inicial del proceso. Resulta importante anotar en este punto que el proyecto tiene tres ejes en los que se mueve su nivel de complejidad: Uno relacionado con el fenómeno modelado y sus posibles representaciones matemáticas, otro relacionado con la implementación tecnológica del mismo, y finalmente uno que contempla

las medidas de desempeño del algoritmo implementado. De esta manera el proceso de construcción del modelo se hace secundario frente a la implementación y el estudio del desempeño del algoritmo en una infraestructura real, en este caso particular la que pertenece a la universidad.

Para evaluar esta eficiencia se usaran en menor medida modelos teóricos, como esta contemplado en el proceso de diseño metodológico de algoritmos en paralelo, así como herramientas de medición de desempeño de los prototipos implementados. En particular es interesante la caracterización de las operaciones de copia entre las diferentes jerarquías de memoria contempladas en una arquitectura heterogénea como la que se tiene de referencia.

La evaluación de los resultados del modelo será proporcional al nivel de detalle utilizado en el programa; debido a que es una prueba de concepto, la exactitud a nivel predictivo del modelo no será la preocupación más urgente. Lo más importante es que el modelo describa adecuadamente el conocimiento científico detrás del fenómeno, es la capacidad de mostrar el comportamiento general del fenómeno en el modelo lo que considerará clave. El nivel de predicción quedaría para el trabajo futuro, siendo la simulación de este tipo de fenómenos de un elevadisimo nivel de complejidad. Además, el tipo de modelado patrocinado por el grupo SIMON[1], quienes son la parte interesada en este, es el basado en conocimiento. Así, resulta imprescindible no solo contrastar el modelo obtenido con la teoría disponible, si no construirlo a partir de la misma, descartando los modelos estocásticos; esto es lo que comúnmente se llama modelado estructural en los cursos que imparte el grupo en la universidad. Esto resulta compatible con la noción de tiempo geológico mencionada anteriormente.

Debido a la cantidad de factores que pretende desarrollar este trabajo de investigación resulta más importante para este conducir la evaluación de los resultados contrastándolos con los resultados de los modelos tradicionales de mecánica de fluidos, que utilizar datos reales; de esta manera se cumple con los lineamientos expresados anteriormente y al mismo tiempo se evita incurrir en un grado de complejidad de resultaría demasiado alto; al tratarse de modelos de flujo de la lava los datos reales son bastante difíciles de manejar debido a las irregularidades de las litologías [6]. Al tratarse de una prueba de concepto el llegar al nivel de precisión de las ecuaciones se considera adecuado. Por otra parte, la construcción de un modelo que permita la evaluación de amenazas con un alto nivel de resolución, así como la utilización de escenarios de simulación con topografías complejas, y la transferencia de los resultados a una plataforma de información geográfica, quedan por fuera de los alcances de este proyecto y serán abordados por trabajos posteriores.

[1] Grupo SIMON de investigación, http://simon.uis.edu.co/

1.4. METODOLOGÍA APLICADA

Hay tres etapas principales para lograr los objetivos del proyecto, junto a sus correspondientes procesos de evaluación:

Etapa 1: Modelado usando autómatas celulares y ecuaciones de dinámica de fluidos. Durante ese proceso es posible construir representaciones parciales, aumentando la complejidad a medida que se incrementa el entendimiento. Se registran los problemas que se van encontrando y los temas que así surgen se incluyen en la revisión del marco teórico. Se identifican alternativas de solución que puedan funcionar para cada problema que emerja. Se realiza una evaluación conceptual que se complementa con la del siguiente paso.

Etapa 2: Construcción de un algoritmo orientado a computación en paralelo, partiendo del modelo. Esto permitirá la simulación del fenómeno. Se realiza una evaluación del algoritmo, comparando con los resultados esperados desde la teoría. Como es una prueba de concepto se usaran escenarios de simulación controlados.

Etapa 3: Implementación del algoritmo que permite la simulación del fenómeno, en un programa prototipo para ser ejecutado en la arquitectura de computo avanzado de la universidad. Se realizan pruebas y evaluaciones de desempeño del prototipo implementado.

El proceso completo se ilustra en la figura 1.

Durante cada etapa es necesario revisar constantemente la teoría y el estado del arte. Cada una de las etapas tiene un proceso de evaluación en el que se decide si se continúa con la siguiente etapa o se refina el trabajo. En cualquier punto del proceso es posible replantear alguna de las etapas anteriores si algún hallazgo lo amerita, generándose un proceso tácito de construcción de prototipos.

Cabe anotar que en la tercera etapa del proceso, en el momento de la implementación, se debe seguir un proceso para la construcción de aplicaciones en paralelo, usando el modelo de programación soportado por la arquitectura heterogénea en la que está basada la infraestructura de cómputo avanzado con la que cuenta la universidad. Como la universidad cuenta con máquinas heterogéneas basadas en GPU de la marca NVIDIA© se debe usar CUDA©. Este proceso se basa en el que se usa para diseñar cualquier aplicación en paralelo adaptándolo para el modelo de programación particular. El método de diseño de aplicaciones en paralelo tradicional, presentado por Foster [1], se resume en la figura 2.

Esta metodología fue planteada en el documento de la propuesta de trabajo de investigación que precedió a este informe [7]. Para su aprobación se solicitó una descripción de en lo que consistía un experimento en computación, así como una descripción del expe-

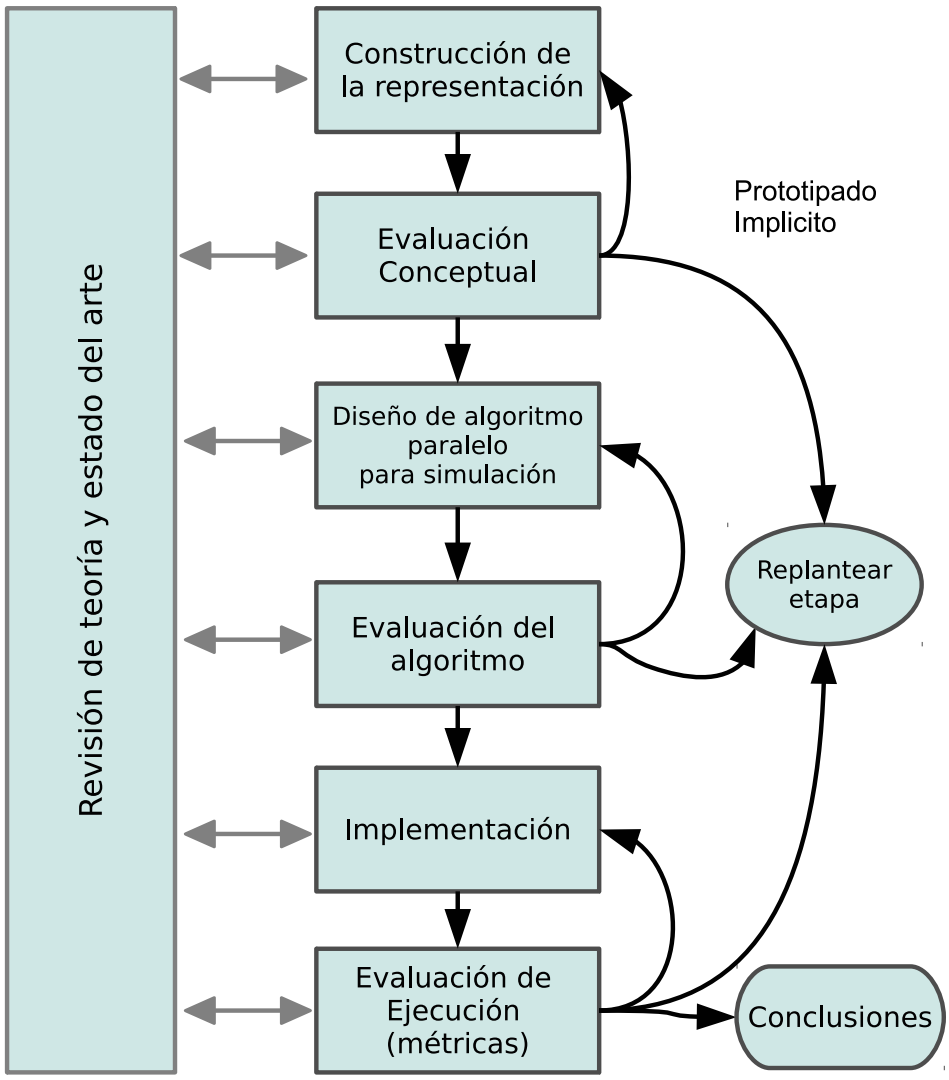


Figura 1: Descripción de la metodología

rimento a usar para el desarrollo del proyecto. Del concepto de experimento en ciencia de computadores se hablará en el siguiente capítulo; del experimento o los experimentos creados para este trabajo se hablará en el capítulo 4.

A continuación se definirán algunos conceptos básicos que se usan en la resolución del problema, incluyendo la sección sobre experimentación en ciencia de computadores mencionada anteriormente.

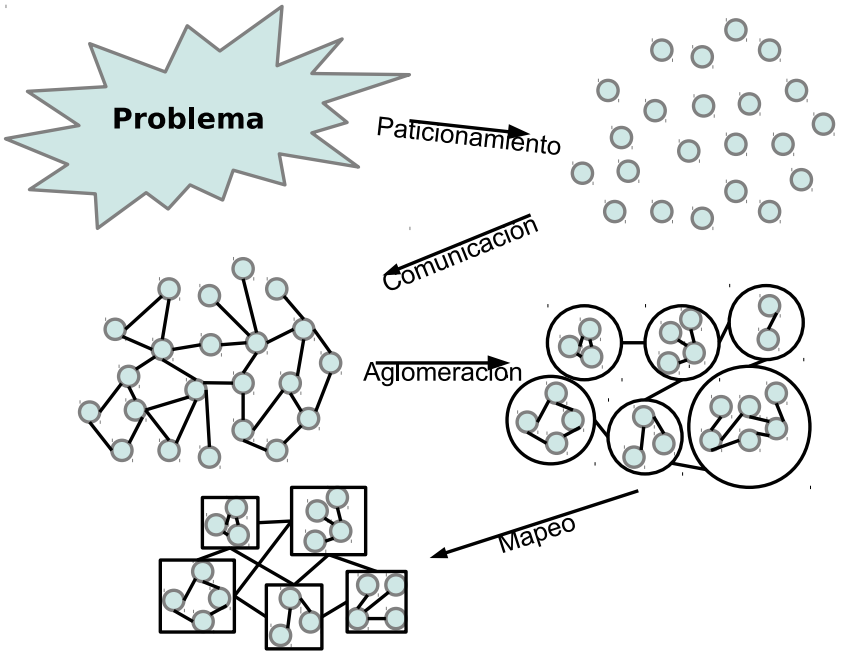


Figura 2: Proceso de diseño para aplicaciones en paralelo, según Foster [1]

2. FUNDAMENTOS TEÓRICOS

En el documento de la propuesta de trabajo de investigación[7] se definieron varios conceptos de importancia para el desarrollo de este trabajo. A continuación se complementan estas definiciones o se incluyen nuevos temas de interés para el desarrollo del trabajo.

2.1. AUTÓMATAS CELULARES

Los autómatas celulares (CA, del inglés Cellular Automata) son una clase de sistemas matemáticos temporal y espacialmente discretos, determinísticos, caracterizados por interacción local y una forma inherentemente paralela de evolución[8]. El concepto de autómatas celulares agrupa una familia de sistemas matemáticos, pero se pueden mencionar cinco características que todos los modelos de autómatas celulares tienen:

- Retículo discreto de celdas: El sustrato de sistema consiste de un retículo de una, dos o tres dimensiones.
- Homogeneidad: Todas las celdas son equivalentes.
- Estados Discretos: Cada celda toma uno de entre un conjunto finito de posibles estados discretos.
- Interacciones locales: Cada celda interactúa únicamente con celdas que están en su vecindario local.
- Dinámica discreta: En cada unidad discreta de tiempo, cada celda actualiza su estado de acuerdo a una regla de transición teniendo en cuenta los estados de las celdas en su vecindario.

Desde la formulación matemática[8, p .45], un CA es una 4-tupla $(L, \Sigma, \sigma_0, \phi)$, con sus elementos definidos de la forma:

- Espacio de estados celular discreto L: Un retículo que puede ser de celdas cuadradas, hexagonales, o incluso de forma e interconexión irregular. Lo importante es que haya una formulación para resolver la localidad, es decir, una regla para determinar los vecinos de una celda dada. Usualmente se usan las cuadradas.
- Espacio local de valores Σ: Cada celda puede tomar un valor σ de entre un conjunto de n valores posibles, siendo estos de naturaleza discreta. Esta naturaleza discreta puede incluir discretizaciones de valores reales, incluso n-tuplas de dichas discretizaciones. La homogeneidad se refiere a la posibilidad de asumir los valores, sin restricciones particulares por celda. El conjunto de los valores de todas las celdas

en L para un determinado momento de tiempo t constituye el estado global del CA para ese punto temporal.

- Conjunto de condiciones de frontera σ_0: La idealización de un modelo de CA implicaría que L debería ser infinitamente grande, pero esto no tiene un sentido practico, luego se deben definir valores de $\sigma - 1$ y σ_n para un L de longitud n (extensible a varias dimensiones) que representan los valores en los bordes del retículo. Estos valores hacen parte de Σ.

- Reglas dinámicas ϕ: $\Sigma_1 \times \Sigma_2 \times \Sigma_3 ... \Sigma_n \rightarrow \Sigma_i$ donde n representa el número de celdas que conforman el vecindario de una celda dada. De la definición se puede extraer que el valor del estado de una celda dada depende de los valores de las n celdas de su vecindario y de esta manera se puede inferir también que la dinámica de las transiciones es local. El tiempo avanza en un intervalo discreto y los estados de las celdas del instante de tiempo siguiente son calculados de acuerdo a ϕ de manera simultanea para todo el retículo L. El estado global del CA para un momento t depende de las reglas de transición, de la dinámica del CA, que a su vez solo depende del momento de tiempo $t-1$. Esta dependencia temporal particular es de suma importancia para la simulación.

Se mencionó en el documento de la propuesta que las propiedades de interacción local y dependencia especial han hecho interesantes a los CA para varias aplicaciones, y han sido mencionadas algunas en el mismo[7].

Una consideración especial es la capacidad de abstraer diversos tipos de fenómenos, lo que conlleva a los CA a ser usados en diferentes escalas espaciales para diferentes fenómenos. Para este trabajo se considera algo que en Spataro et al.[9] es llamado Autómatas Celulares Macroscópicos. En ese contexto se refiere a que las celdas representan unidades sustanciales en tamaño con relación al fenómeno simulado, en este caso en particular porciones de área respecto a un total en la escala geográfica, dándole así carácter macroscópico a los elementos representados por las celdas del autómata. Lo opuesto sería un CA que represente un flujo pequeño, donde los estados representarían porciones microscópicas del líquido, por ejemplo.

2.2. FLUJOS DE LAVA VOLCÁNICA

La lava, es una corriente de roca fundida que puede tener un amplio rango de composiciones. Aparte de su composición química, sus propiedades físicas son influenciadas por sus componentes volátiles, contenido de cristales, e historias de enfriamiento. Esto genera comportamientos muy variados incluso en lavas con composición muy similar[6, p .141]. Son fluidos reológicamente complejos, que combinan características de fluidos newtonianos (viscosos) y de sólidos deformables (hookianos), específicamente fluidos de tipo Bingham, como lo estableció Robson en 1967[3]. La caracterización matemática de los fluidos

reológicamente complejos es parte del tema de las asignaturas de fenómenos de transporte a nivel de pregrado y postgrado, y está cubierta a nivel básico en libros de texto universitarios como por ejemplo el libro escrito por Deen, Analysis of Transport Phenomena[10]. Este trabajo no hace afirmaciones novedosas respecto del tema, así que la guía directa de un texto de este tipo resulta adecuada. Finalmente, respecto al modelado matemático, la gran mayoría de los trabajos encontrados en el estado del arte se basan en una formulación propuesta por Dragoni et al[4], que construyó la aproximación de las ecuaciones de estado para flujo de fluídos Bingham sobre pequeñas superficies, permitiendo así al construcción de modelos de autómatas celulares macroscópicos de flujos de lava.

2.3. EXPERIMENTOS EN COMPUTACIÓN Y MEDICIÓN DE DESEMPEÑO

La ciencia experimental tiene tres componentes principales:

- Observación
- Prueba de hipótesis
- Reproducibilidad

La parte importante para este trabajo es dejar en claro como estos aplican en la ciencia de computadores y como se manifiestan en el diseño experimental de este trabajo. La ideas básicas se obtuvieron del texto de Feitelson [11] y de las experiencias de los equipos de investigación MOAIS[1] y MESCAL[2], del Instituto Nacional de Informática y Control de Francia, INRIA[3], quienes prestan su apoyo a los trabajos del grupo de investigación en el que se desarrollo este trabajo.

La ciencia de computadores es una disciplina en constante cambio y puede ser clasificada de tres maneras diferentes, como ciencia, como ingeniería o como parte de las matemáticas [11].

La experimentación en ciencia de computadores puede ser clasificada en varias clases, según Feitelson [11]:

La ciencia de computadores experimental como opuesto a la ciencia de computadores teórica, es decir, un área que se centra en la construcción de sistemas reales para probar su factibilidad. En ese caso se trata de desligar los sistemas abstractos de los que se pueden construir en la realidad. En ese contexto se le llamaba ciencia de computadores teórica a los productos de la investigación universitaria que no contemplaban la construcción de sistemas directamente aplicables a la realidad. Así, se puede concluir que esta definición es más cercana de la ciencia de computadores como ingeniería.

[1] http://moais.imag.fr/
[2] http://www.inria.fr/equipes/mescal
[3] http://www.inria.fr/

La ciencia de computadores experimental como el modelado matemático del comportamiento de los sistemas de computadores. De esta manera se podría decir que el estudio de los modelos abstractos que surgen en el cuerpo de conocimiento de la ciencia de computadores puede ser calificado como experimentación. El modelado y la abstracción son propuestos junto al diseño y la teoría como los paradigmas que atraviesan toda la actividad de las ciencias de la computación. Además, la experimentación puede ser considerada como un paso en el ciclo de realimentación del proceso de ingeniería: Un sistema tiene propiedades anticipadas, pero estas son probadas experimentalmente; si los resultados no corresponden con las expectativas el diseño del sistema es modificado de manera acorde.

Una tercera forma de plantear la ciencia de computadores experimental, que es la principalmente seguida en este trabajo, es la de evaluar los sistemas de computadores usando métodos empíricos análogos a los usados en la ciencias naturales. Se puede considerar esta como una forma similar a la anterior pero con un foco más reducido: Se evalúan sistemas individuales sin necesariamente llegar a tocar los conceptos fundamentales de la ciencia de computadores.

Una cosa que se puede notar es que algo similar a esto aquí planteado ya se viene realizando aunque quizás no se le categorice como tal: Hay trabajos que asumen una de estas formas para su desarrollo y también hay comunidades que tratan de caracterizar estas actividades. Lo que falta es una identificación más general, una unificación de estas comunidades.

En conclusión Feitelson apunta a mostrar que hay un espacio en las ciencias de la computación para la observación del mundo como en las ciencias naturales, así como una utilidad para las pruebas de hipótesis en un nivel directo o intermedio como parte de la evaluación y el entendimiento de sistemas creados por el hombre; finalmente, hay necesidad de reproducibilidad y repetición de los resultados como se aboga en el método científico.

2.3.1. Observación en Ciencia de Computadores

La observación en el contexto de las ciencias significa estudiar la naturaleza, el fenómeno como aparece en la realidad. Hace falta distinguir entre observación, que sería análogo a la ciencia experimental en ciencias naturales, y simulación, cuyo análogo sería la ciencia computacional. Importante resulta el aclarar que para trabajos como este en el que el artefacto computacional creado es una programa de simulación, la simulación a la que se refiere el apartado anterior sería la del funcionamiento computacional del mismo, no la que este realiza. Observar y medir son las bases para formar modelos, que llevaran de ser exitosos, a la generación de conocimiento.

Medir es una actividad más compleja de lo que parece, ya que no siempre es fácil medir de manera confiable. Varios problemas que pueden aparecer son [11]:

- Hay que aislar el fenómeno exacto que se quiere medir, lo que resulta difícil. En el contexto de la evaluación de desempeño de los procesos en un sistema operativo, por ejemplo, hay que tener en cuenta demonios, operaciones que son compuestas y encierran los procesos de interés, etc.
- La escala puede ser inadecuada, por ejemplo, unidades de medidas muy grandes o pequeñas, como registros de milisegundos para procesos que tardan fracciones de estos, lo que generaría un informe errado. A veces se pueden acumular mediante repetición, pero este proceso aumenta el riesgo de interferencia externo.
- A veces la toma de medidas puede ser tecnológicamente muy difícil de implementar en la medida en la que hay que penetrar en diversas capas de la arquitectura estudiada.
- En ocasiones el problema se produce en el diseño, cuando no se define exactamente lo que se quiere medir. Tomando nuevamente el ejemplo de los procesos, si queremos medir su duración ¿nos interesa esta en el espacio de usuario? ¿del kernel?, etc. Muchos fenómenos son complejos y difíciles de definir, y las posibles medidas que surgen pueden contar diferentes historias.
- Finalmente puede haber variaciones inexplicadas aún excluyendo la generadas en el sustrato computacional.

Un factor que queda por contemplar es lo que podría llamarse el valor social de las medidas definidas; para cada investigador habrán ciertos factores más importantes que otros, y la creación y selección de las medidas serán dictadas por esta valoración. Además se requieren metodologías que eviten perturbaciones en el sistema medido, o que incluyan mecanismos para resistirlas, y que sean aplicables a diferentes situaciones. Al diseñar las medidas se pueden magnificar ciertos efectos y el saber cuales son más problemáticos para cada situación se logra con la experiencia. Finalmente estas experiencias se deben compartir entre investigadores para de esa manera lograr mayor eficiencia en los procesos de experimentación.

Respecto a las métricas, lo que se quiere medir es la pregunta importante, y se pueden medir varias magnitudes. Las básicas serían tiempo, FLOPS, Mb/s, etc, pero también hay otras difíciles de definir como localidad. Estos conceptos son difíciles de cuantificar para compararlos; de esto se desprende que puedan emerger varias métricas distintas para un mismo factor. Un caso de interés particular para el que no existe una única métrica es el de como medir el desempeño de un computador de alto desempeño, ya que todas las métricas planteadas presentan alguna deficiencia; prueba de esto es la emergencia de listados como el graph500[4], que pretende resolver inconvenientes de la forma de medición del más tradicional top500[5]. En ocasiones la única manera de obtener una métrica ade-

[4]http://www.graph500.org/
[5]http://www.top500.org/

cuada es probarla, y con tiempo y muchos datos se puede obtener una versión efectiva de la misma.

Otro punto importante de la observación es la necesidad de encontrar sorpresas durante el proceso, ya que sin estas no hay verdadera experimentación. Es diferente calibrar o demostrar un modelo que experimentar con él, siendo lo primero un proceso en el que el resultado final debe ser predecible, ya que se buscan activamente errores el algo que tiene una finalidad definida en lugar de nuevo conocimiento. Cuando se habla de experimentación surge una problemática difícil de tratar, la inhabilidad de aceptar como correctos resultados que contradigan las ideas aceptadas: Si un resultado de experimentación contradice el cuerpo de conocimiento actual es más fácil descartarlo como un error que aceptarlo y formular nuevas teorías. Un ejemplo claro de esto es el descubrimiento de la auto-similitud en el comportamiento de la trasferencia de paquetes en una red, lo que contradecía las ideas establecidas de comportamiento de tipo Poisson[11]. El siguiente paso es el de constructor modelos a partir de estas sorpresas.

La acumulación de datos relacionadas con anomalías con respecto a las ideas establecidas, y en general relacionados a ideas nuevas permite la construcción de modelos que lleven a nuevo conocimiento. Al igual que en las ciencias naturales se va del experimento a la teoría, pasando por una serie de filtros. Los modelos deben tratar de explicar situaciones a partir de los datos obtenidos, no ajustar los datos a las teorías existentes. Además, la correcta construcción de un modelo implica un nivel de complejidad adecuado: Ni tan simple que ignore factores importantes, ni tan complejo que incluya algunos que no lo son. Otra situación interesante es la construcción sistemática de modelos sobre un fenómenos y de procesos así se obtiene que es posible modelar algo mucho o muy poco, o sea, construir demasiados modelos que intentan describir un fenómeno de manera parcial o, a su vez, construir muy pocos modelos sobre fenómenos que los requieren y que tienen datos disponibles para realizar el proceso de modelado. De las barreras para el correcto modelado destaca Feitelson como posibles la falta de imaginación y la pereza intelectual, citando nuevamente el ejemplo de los modelos de Poisson: Se tiene a modelar demasiadas cosas como si cumplieran procesos de Poisson sin probar otras alternativas, ya sea por su facilidad de tratamiento matemático, como por la simplicidad de asumir comportamiento independiente y al azar.

En este trabajo nada se dirá acerca del problema de definir si es posible alcanzar la verdad acerca de los fenómenos naturales, y en el caso particular de la ciencia de computadores, la verdad acerca de los fenómenos que acaecen en los sistemas de computadores. Aclarado esto, el problema de la verdad en las ciencias de la computación se relaciona con la relevancia de los resultados, los cuales son frágiles y muy dependientes de la situación en la que fueron obtenidos; esto se produce por lo cambiante de los sistemas de computadores. A pesar de esta fragilidad es importante continuar con los experimentos ya que se puede aprender mucho de ellos así sus resultados no sean universalmente correctos. El problema principal no es que estos resultados no lleven a verdades universales y durade-

ras, el problema es creer que lo harán. Eliminada esta presión de absoluta generalidad la utilidad de los experimentos será encontrada.

2.3.2. Prueba de Hipótesis en Ciencia de Computadores

La prueba de hipótesis es la etapa que sigue a la formulación inicial del modelo en el proceso de ciencia, es lo que lo convierte en conocimiento. Hay dos tipos de hipótesis, de nivel macro y de nivel micro. Las de nivel macro definen el campo científico de la disciplina (por ejemplo $P \neq NP$). Las hipótesis de nivel micro son de carácter atómico, intentan explicar fenómenos bien precisos y definidos. Las de nivel micro son propuestas, mientras que las de nivel macro son emergentes.

Los experimentos permiten pasar de hipótesis a conocimiento mediante las pruebas, entonces las hipótesis deben ser falsables, y el objetivo de las pruebas es reconocer un nivel de generalidad que permita diferenciar las situaciones ocurridas por azar y las que son inherentes al fenómeno. Para lograr esto se deben variar los factores del fenómeno, que corresponden con entradas en el experimento. Para probar las hipótesis en algunos casos los experimentos se plantean desde el principio y a veces se plantean nuevos experimentos como respuesta a los resultados obtenidos. Otra característica importante de los experimentos es que así como las observaciones llevan al modelo, el modelo debe ser capaz de reproducir las observaciones. Finalmente, las pruebas utilizadas en este proceso pueden mostrar que el modelo es incorrecto y que debe ser modificado o desechado.

Después de la hipótesis viene la teoría y esta debe ser refutable. Si no hay manera de intentar refutarla entonces se considera un dogma y no una teoría científica. En la ciencia nada se debe creer, siempre se debe esperar que el conocimiento cambie, y que las ideas aceptadas pueden ser reemplazadas por otras, en cuanto nuevas observaciones y modelos aparezcan, con sus correspondientes pruebas.

2.3.3. Reproducibilidad en Ciencia de Computadores

Antes de hablar de reproducibilidad hay que aclarar una cosa que a veces parece ser olvidada, especialmente en este mundo científico de incesante publicación y lucha por financiación: Los errores son una necesidad, siempre ocurrirán aún en trabajos publicados y sometidos a revisión por pares. Detectar errores es una actividad social que requiere que el trabajo pueda ser reproducido y que haya gente externa con la disposición de hacerlo. Si otras personas reproducen el trabajo y obtienen resultados similares a los del trabajo inicial se obtiene confianza sobre la validez del mismo. Así, el que la gente los repita es parte integral del proceso de experimentación.

Para que un estudio pueda ser reproducido este debe cumplir ciertas condiciones:

- Estar bien descrito, con gran detalle, a manera de una receta. Ninguna operación a realizar debe dejarse a la interpretación de quien intenta reproducir el trabajo.
- Poner a disposición el software, especialmente el código fuente, ya que de esta manera se considera más utilizable para los interesados. Si se usa software de terceros se debe especificar la version usada y la configuración completa.
- Poner a disposición los datos, particularmente el conjunto de datos de entrada.
- Permitir el acceso a la infraestructura, de ser posible, más si esta es bastante especial. Si no, entregar información detallada de esta, en los aspectos relevantes, como arquitectura, sistema base, etc.

Seguir estas indicaciones también puede resultar muy útil para aquel que hace el estudio, pues permite continuar el trabajo con facilidad. El hacer reproducible el estudio hace más eficiente la realización de estudios posteriores relacionados con el tema.

Además de los beneficios para el proceso de comprobación y validación externa de resultados la reproducibilidad sirve para mejorar el entendimiento del sistema observado. A veces el reproducir enseña cosas que no se descubrieron durante el desarrollo del trabajo original, especialmente a nivel de conclusiones; esto se genera porque a veces se busca entendimiento de una parte del fenómeno, y al ser logrado este, se ignoran observaciones interesantes en otras partes del mismo. También resulta importante tener en cuenta que la exacta reproducción numérica probablemente no es requerida; se deben reproducir las deducciones no los números. El proceso de reproducción tiene un carácter mas cualitativo que cuantitativo[11].

Otro tema que se desprende de la reproducibilidad es la estandarización. A veces se logra obtener un canon del proceso de experimentación, pero esto no implica que por esto se llegue a una mejor forma de realizar los estudios, sino que todos acceden a hacerlos de esa manera. Lo hacen porque comprenden que los beneficios de compatibilidad que se obtienen por usar un mismo enfoque superan a los obtenidos al usar uno especializado pero no aceptado por los demás. Un ejemplo notorio son las medidas de desempeño, en donde el compromiso es común y muy necesario; por eso se usan ciertos benchmarks a pesar de tener falencias documentadas. Los estudios de desempeño deberían, especialmente, incluir una sección de metodología para permitir su reproducción; estos después deberían llegar a convertirse en manuales de laboratorio y al final en conjuntos de buenas prácticas. Este proceso simplificaría el reporte de estudios similares. A pesar de las ventajas de la estandarización, siempre habrá gente interesada en hacer modificaciones a las metodologías, ya que existirá siempre el interés de mejorarlas. La cantidad de trabajos relacionados con las metodologías será menor a los trabajos generales, pero nunca se podrán desligar por completo las dos actividades, pues las nuevas metodologías solo podrán ser probadas utilizándolas en estudios generales y algunos estudios generales inspirarán nuevas metodologías.

Finalmente, es importante mencionar el progreso generado inherentemente por la reproducción de los estudios: gracias a estas reproducciones se pueden producir meta-estudios, ya que cada intento de reproducción es intrínsecamente diferente; estos meta-estudios a su vez pueden generar más conocimiento, de naturaleza más general.

En el siguiente capítulo se describirá el modelo de simulación construido desde sus fundamentos teóricos desde el punto de vista físico-matemático y computacional, hasta su proceso de implementación y evaluación. También se analizará su complejidad desde la teoría, lo que permite generar hipótesis sobre el desempeño de las ejecuciones.

3. CARACTERÍSTICAS DEL MODELO DE SIMULACIÓN DE FLUJOS DE LAVA VOLCÁNICA

En este capitulo se incluye una descripción del modelo físico-matemático básico utilizado en la simulación, explicando la abstracción realizada y los útiles teóricos y lógicos usados en su construcción. Además se explica en general el proceso de implementación y se describe de manera general las herramientas tecnológicas usadas. El capítulo también incluye una sección que trata sobre el análisis de complejidad teórica del algoritmo y también de las oportunidades de paralelismo, señalando posibles esquemas de particionamiento y comunicación, tamaños de grano de procesos y/o datos, puntos de la ejecución que requieren ser seriales, y otras características del problema. Finalmente, se incluye una sección sobre la evaluación del modelo.

3.1. CARACTERIZACIÓN FÍSICO-MATEMÁTICA DEL FLUJO DE LAVA VOLCÁNICA

El modelo se basa en las ecuaciones de estado de la mecánica de fluidos, es decir Navier-Stokes, incluyendo las ecuaciones de transferencia de calor. Como se quiere hacer correr esto sobre un sustrato computacional de autómatas celulares se requiere hacer que los flujos entre las celdas estén definidos por unas reglas, que deben describir el comportamiento del flujo a la luz de las ecuaciones. Se hace entonces una discretización de las ecuaciones, tomada de Miyamoto y Sasaki[3]. El flujo desde una celda hacia otra adyacente esta definido por:

$$\Delta V = \frac{\rho S_y h_c^2 w}{3\eta}\left[\left(\frac{h}{h_c}\right)^3 - \frac{3}{2}\left(\frac{h}{h_c}\right)^2 + \frac{1}{2}\right]\Delta t \tag{3.1}$$

Donde los parámetros están definidos de la siguiente forma:

S_y = esfuerzo umbral (parámetro del fluido Bingham)
η = viscosidad
h_c = grosor crítico
w = ancho de la celda
ρ = densidad
ΔV = Volumen transferido.

El grosor crítico es el valor de grosor en el cual la fuerza producida por la gravedad permite romper la cohesión de las moléculas del fluido y de esta manera el fluido Bingham deja de comportarse como sólido y pasa a comportarse como un líquido viscoso. Este valor está físicamente relacionado con la viscosidad y el esfuerzo umbral (también llamado fluencia)

pero de acuerdo a la aproximación realizada puede calcularse de la siguiente manera:

$$h_c = \frac{S_y\sqrt{\Delta z^2 + \Delta x^2}}{\rho g(\Delta z - \Delta h)} \tag{3.2}$$

Donde:

Δz = diferencia de grosor entre las celdas.
Δh = diferencia de altitud del terreno entre las celdas.

El modelo solo realiza transferencia de calor por radiación. La transmisión hacia el suelo no es tenida en cuenta y la convección se considera despreciable. La perdida de temperatura por radiación es:

$$\Delta T_{rad} = \epsilon A \sigma_{SB} T^4 \Delta t \tag{3.3}$$

Donde η es la emisividad, σ_{SB} es la constante de Stefan-Boltzmann, T la temperatura de la celda y A el área de la celda.

Hay que tener en cuenta la transferencia de calor por flujo, es decir, el calor que se transporta en el material, que es calor que la celda origen pierde y la celda destino gana. Se puede expresar como:

$$\Delta Q_{t,m} = (\sum_{q_i>0} q_i T_i + \sum_{q_i<0} q_i T)\rho C_v \Delta t \tag{3.4}$$

Donde T es la temperatura de la celda origen, T_i es la temperatura de las celdas destino, q_i es el flujo (calculado mediante (3.1)) y C_v es el calor específico por unidad de masa.

Finalmente hubo la necesidad de limitar la perdida de calor por radiación, esto debido a la relación grosor vs área: Si el grosor es muy pequeño resulta difícil que un volumen asociado a un grosor así pueda cubrir uniformemente el área de referencia. Como la perdida por radiación depende del área, se introdujo un valor de corte relacionado con el grosor; para grosores menores que 0,0001 la perdida de temperatura se desprecia.

Los pasos del algoritmo para el calculo de los valores de las celdas para un tiempo = t son:

- Leer los datos de altitud de un archivo de texto
- Inicializar los valores de las celdas, altitud (del archivo), grosor (0), temperatura (273K), viscosidad y fluencia (a partir de una relación, o constantes, dependiendo del escenario de simulación)
- Recorrer toda la matriz y calcular el número de salidas de lava de cada celda, teniendo en cuenta la formula para el grosor crítico (3.2).
- Obtenido el número de flujos de salida de cada celda se procede a calcular el valor del flujo de salida (3.1). Se necesita saber el número de flujos con anterioridad para dividir el flujo en partes iguales y así no extraer más material de la celda del

que puede soportar. Al calcular los flujos se calcula también el calor saliente usando (3.4). Estos cálculos se realizan para todos los elementos de la matriz.

- Se suman los flujos entrantes para cada celda. En el paso anterior se calculaban los flujos pero no se sumaban al grosor, para garantizar la dependencia temporal. Se acumulan los calores de entrada de la misma manera. Cuando se ha realizado la suma de todos los flujos se actualiza el valor del grosor para cada celda.
- Se calcula la temperatura perdida por radiación (3.3). Se calcula la perdida o ganancia de temperatura por flujo, y sumando los efectos de estas dos temperaturas, se actualiza el valor de temperatura de la celda.

Después de realizados estos pasos se puede decir que se pasó del tiempo t al tiempo $t+1$. Este proceso se debe repetir por el número de pasos de tiempo definido al inicio de la simulación. Entre cada paso se reportan los resultados, escribiéndolos en un archivo de texto plano.

3.2. IMPLEMENTACIÓN SOFTWARE DEL MODELO

3.2.1. Proceso de implementación

Teniendo una idea de los pasos de cálculo para la simulación, el siguiente paso consiste en adquirir los medios tecnológicos para implementar dicha simulación como un programa, un dispositivo software. En la concepción del proyecto se sabía que se iba a usar la tecnología NVIDIA© CUDA©, ya que el clúster de la universidad utiliza esa tecnología; de esta forma se seleccionó de antemano el modelo de programación en paralelo, correspondiente con la tecnología de la infraestructura disponible. Al usarse este modelo y estas herramientas era necesario usar el lenguaje de programación C, con las extensiones para usar CUDA (realmente constituye un lenguaje derivado del C, llamado CUDA C, con su propio compilador y herramientas de compilación).

En primer lugar se construyó un programa de ejecución serial que constituiría la base para los programas que corren en paralelo apoyados por la GPU. Después de probar el funcionamiento del mismo se implementaron las diversas versiones en paralelo, que se explican en el capitulo 4.

La parte fundamental del proceso de implementación es construir una función que calcule los flujos de una celda a otra teniendo en cuenta ciertas consideraciones:

- Es necesario implementar un mecanismo para calcular los flujos de todas las celdas para un paso de tiempo t, pero que los valores de flujo no se trasfieran de una celda a otra de manera inmediata: Se deben calcular todos, de entrada, de salida, realizar el balance de flujos, y después actualizarlos, modificando el valor de grosor de todas las celda. Esto para garantizar que a las celdas que se evalúan primero no se les

asigne todo el flujo si no que haya una repartición equitativa cuando la celda donante tenga varias celdas receptoras: no hacerlo así implicaría generar una anisotropía errónea.

- Los valores de flujo de calor funcionan de manera similar: El valor de temperatura nuevo por celda se calcula al final, después de calcular todos los flujos de calor de todas las celdas, salientes y entrantes. Así mismo, los cambios de temperatura por radiación se deben hacer o antes del proceso de cálculo de flujos, o después, pero no durante el mismo.
- Entre pasos t y $t+1$ de tiempo se deben reinicializar los flujos, pues los valores dependen estrictamente del estado al iniciar el paso de tiempo que se está evaluando. Esto significa que los flujos para paso $t+1$ no tienen memoria de los pasos anteriores, dependen únicamente del valor del grosor de la celda al iniciar la evaluación.

Además de la calcular los flujos, la función principal debe copiar en la memoria principal los datos resultantes de ese proceso de cálculo para así poder escribir los archivos de salida. Estos archivos de salida se usan para generar gráficas simples mediante GNUPlot, que permiten dar una idea general de comportamiento para un paso de tiempo; los archivos de texto quedan disponibles para realizar un análisis posterior más completo.

3.2.2. Resultados

Después de implementado el modelo en C se procedió a correr una simulación con los parámetros físicos registrados en la tabla 1: Cabe anotar que la viscosidad y la fluencia se dejaron como constantes para las simulaciones para facilitar la validación. Esto es acorde con el proceso de validación del artículo de Miyamoto y Sasaki[3].

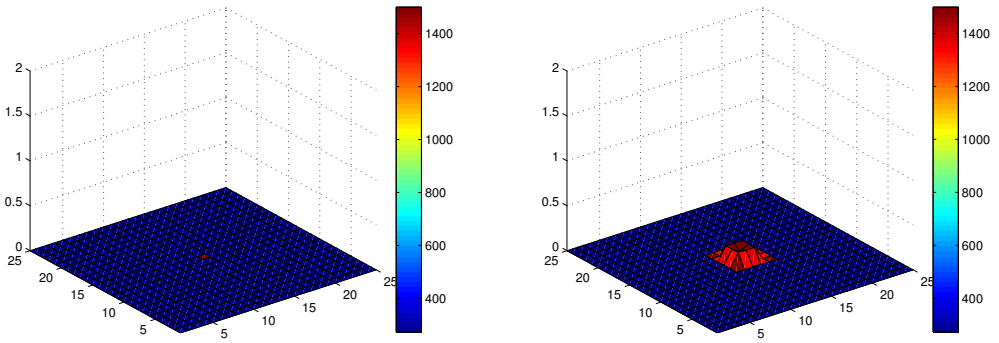
Se obtuvieron conjuntos de datos de salida con los valores en $x, y, z\, y\, T$ para cada celda, siendo x, y, z coordenadas espaciales y T la temperatura de la lava en la celda; se debe tener en cuenta que z es la altitud del modelo topográfico más el grosor de celda acumulado. Debido a la cantidad de datos de salida se decidió limitar el proceso de validación hasta $t = 360s$ tomando cada minuto, excepto por $t = 0s$ y $t = 10s$, que se tienen en cuenta para ilustrar el comportamiento temprano de la simulación. Presentaremos más adelante las gráficas en 3D de las superficies generadas por las altitudes, con un mapa de color que representa la temperatura.

En la figura 3 se muestran los imágenes de la simulación en los tiempos $t = 0, 10, 60, 120$ segundos. En la figura 4 se muestran los tiempos $t = 180, 240, 300, 360$. Se hizo un zoom de la imagen, reduciendo el área presentada para ver un poco mejor la forma. En comportamiento tiene una tendencia hacia la curva parabólica, que se puede evidenciar en las imágenes, pero se puede evidenciar de mejor manera en los perfiles para $x = 511$, que muestran la curvatura de perfil más alta ya que en esa fila se encuentra el punto de extrusión. En la figura 5 se muestran dichos perfiles para los tiempos $t = 0s$, $10s$, $60s$, $120s$, $180s$, $240s$, $300s$, $360s$. Nótese el pequeño pico que se forma en $511, 511$ debido a la extrusión

Parámetro	*Valor*	*Unidades*
Calor específico	840	$Jkg^{-1}K^{-1}$
Tasa de extrusión	100	m^3/s
Emisividad	0.9	-
Viscosidad	850	Pa/s
Fluencia	225	N/m^2
Duración de la erupción	360	s
Densidad	2500	N/m^2
Gravedad	9.8	m/s
Ancho de la celda	1	m
Temperatura base de las celdas:	273	K
Tamaño en y	1024	m
Tamaño en x	1024	m
Puntos de extrusión	1	-
Coordenadas puntos de extrusión	{511, 511}	-
Mapa de altitud	0.00 (para todas las celdas)	m

Tabla 1: Parámetros de entrada para prueba del modelo

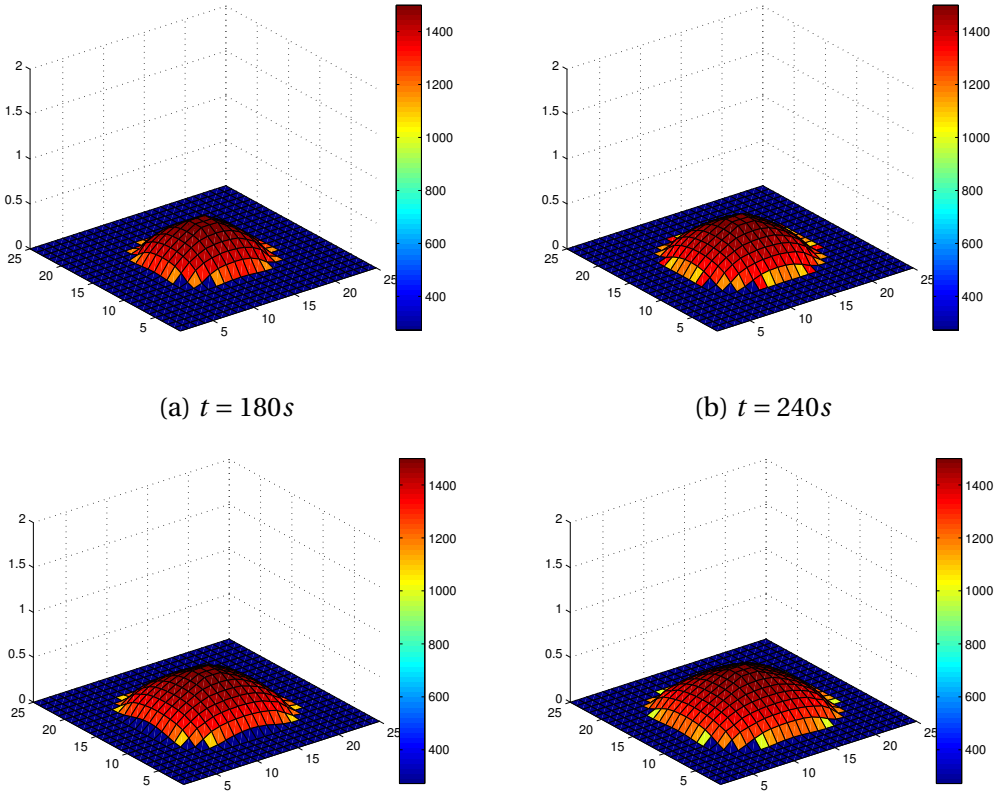
discretizada.



(a) $t = 0s$ (b) $t = 10s$

(c) $t = 60s$ (d) $t = 120s$

Figura 3: Comportamiento de la simulación para varios t



(a) $t = 180s$ (b) $t = 240s$

(c) $t = 300s$ (d) $t = 360s$

Figura 4: Comportamiento de la simulación para varios t

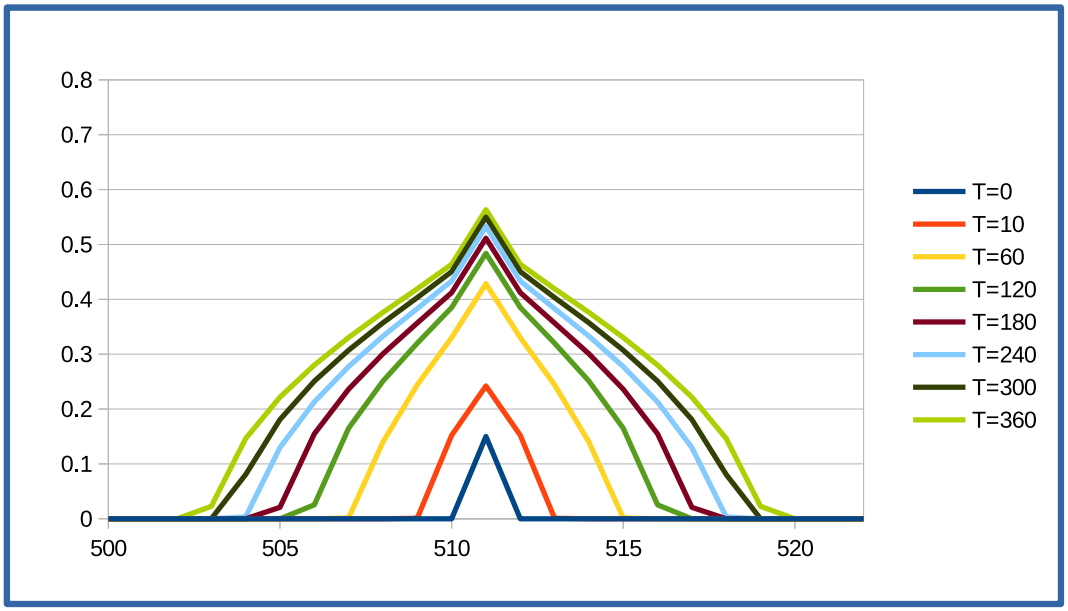


Figura 5: Perfiles de altitud, $x = 511$

3.3. ANALISIS DE COMPLEJIDAD

Teniendo en cuenta los pasos descritos en el apartado de las páginas 31 y 32 se puede inferir un aproximado del número de operaciones que ejecuta el programa. A continuación se realizará un análisis que permitirá establecer un estimado asintótico del número de operaciones requeridas.

El problema tiene tres dimensiones en las cuales puede crecer en tamaño, el tiempo t, las filas n y las columnas m. Teniendo en cuenta que se usará una matriz cuadrada para los escenarios de simulación, el numero de filas se puede asumir igual al de columnas, reduciendo las dimensiones a dos: t y n.

Para calcular los valores de estado de las celdas para un tiempo t se requieren los siguientes pasos:

- Inicializar los valores de estado auxiliares de cada celda, es decir, los que solo dependen del estado actual y sus valores no se mantienen para el paso $t+1$ (viscosidad, fluencia, número de salidas de lava, y los flujos de entrada y salida de volumen y calor). El número de salidas y los flujos se deben inicializar en 0 para cada uno de los pasos. Se producen 6 cálculos por celda.
- Calcular el número de salidas y los flujos. En este caso se debe calcular 8 veces por cada celda, debido a que puede recibir flujos de cada una de las celdas vecinas, generando un máximo de 24 operaciones de cálculo más 3 operaciones de comparación. En el peor caso se ejecutarían 27 operaciones por celda.
- Calcular para cada celda el balance de flujos, nuevo grosor de lava y temperatura. Esto genera 6 operaciones por celda, aproximadamente.

Al finalizar el proceso de cálculo para el tiempo t se han acumulado aproximadamente $6n^2+27n^2+6n^2$ operaciones. Además falta contabilizar al menos n^2 operaciones de copia de datos de memoria para generar la salida, y puede ser más si se usa una arquitectura heterogénea; en ese caso ese valor puede aumentar hasta $4n^2$. Al consolidar el número de operaciones para el núcleo del algoritmo tenemos $43n^2$ y con el tiempo en consideración se podría aproximar el número de pasos para toda la simulación a $t \times (43n^2)$. al tratarse de una notación asintótica, que es una aproximación, se puede concluir que el número de operaciones estimado es $O(tn^2)$.

Teniendo en cuenta que para el caso específico de las pruebas que se van a realizar se tiene un número finito de pasos de tiempo (que será especificado en la siguiente sección) el valor de t se puede eliminar de la función de crecimiento asintótico; La función de referencia para el problema como se trasnfoma en $O(n^2)$. Esta aproximación es válida para el análisis asintótico pero queda abierta a la discusión cuando se conozcan ciertos detalles de los escenarios de simulación; si el número de pasos de tiempo no es muy grande, el número total de operaciones se verá afectado de manera significativa. Así también, para

valores de n pequeños, los coeficientes no se pueden desechar.

3.4. ANÁLISIS DE AGLOMERACIÓN DE DATOS Y/O PROCESOS

Las características de la estructura matemática usada en el problema, los autómatas celulares, favorecen el uso de una descomposición de dominio: La homogeneidad de las celdas tanto en posibles valores de estado como en reglas de transición, así como el hecho de que los estados resultantes para cada celda en cada paso de cálculo solo dependen de los valores de sus ocho celdas vecinas, hacen del paralelismo de datos un enfoque natural para el problema. Por otra parte, la descomposición por tareas, de difícil implementación en el modelo de programación de CUDA (cabe recordar CUDA fue concebido como un modelo de programación para paralelismo de datos [12, 13]), no resultaría muy útil debido a la de cálculo simultaneo de las propiedades de las celdas para un paso de tiempo t; esta necesidad hace prácticamente imposible ejecutar tareas distintas mientras se ejecutan las relacionadas con el cálculo principal. Además, la homogeneidad de las reglas de transición obliga a que todas las tareas sean iguales, es decir no existirán tareas diferentes a ejecutar, salvo cálculos menores que están contenidos en el flujo de ejecución, anulando completamente la posibilidad de descomposición funcional a nivel del algoritmo de simulación.

En cuanto al tamaño del grano en el que se descompone el dominio, cada celda es calculada por un hilo, y este solo accede a los valores de estado de la celda a calcular y de sus ocho celdas vecinas. Cabe anotar que el conjunto completo de datos se carga en la VRAM antes de iniciar el proceso de cálculo para cada tiempo t, pero para las operaciones de cálculo cada hilo solo accede a las celdas mencionadas anteriormente; esta carga completa se hace para favorecer el desempeño, debido a la característica de alto throughput de la VRAM. De esto se concluye que el problema es de grano fino.

En cuanto a la comunicación y su influencia en la aglomeración, en la sección 3.1 en la página 31, y luego en la sección 3.3 en la página 37 se describió y caracterizó el proceso de cálculo de los valores de estado para cada celda para un paso de tiempo t. Se puede notar que esos pasos requieren congruencia de los datos, es decir, entre cada uno de los pasos se debe sincronizar la ejecución: por ejemplo, no se puede pasar en una celda del proceso de inicialización al de cálculo de flujos de entrada y salida sin que terminen de inicializarse las demás. Esto limita bastante las posibilidades de ejecución en paralelo, ya que la cantidad de instrucciones que se pueden ejecutar entre sincronizaciones es relativamente poca. Esto puede impactar el desempeño de las versiones en paralelo.

3.5. EVALUACIÓN DEL MODELO

3.5.1. Mecanismo de evaluación

El mecanismo de validación de los modelos desde el punto de vista de la simulación debe cumplir con ciertas características como tener en cuenta que las coordenadas son conjuntos ordenados, y que los atributos de interés (temperatura, grosor, etc) están relacionadas con los valores de las coordenadas. Como es necesario comparar conjuntos de resultados ordenados, se propone utilizar un análogo al coeficiente de determinación usado en la regresión lineal, ya que esta medida tiene en cuenta relaciones, mientras que un estadístico de la familia del ANOVA no tiene en cuenta el orden. No se pretende afirmar que se prueba un nivel de correlación, ya que este es un concepto estadístico bien definido. En este proceso de comparación no existe la relación variable independiente - variable dependiente ni el de variables codependientes, simplemente se busca un mecanismo para cuantificar las variaciones numéricas entre dos conjuntos de datos.

Usualmente el coeficiente de determinación (R^2) se define como:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{3.5}$$

SS_{tot} es la suma total de los cuadrados de las diferencias de un conjunto de datos con respecto a su media. Este valor es proporcional a la varianza de los datos observados, en este caso el análogo es el conjunto de los datos a comparar:

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \overline{y})^2 \tag{3.6}$$

SS_{res} es la suma total de los cuadrados de los residuos, en este caso los residuos de restar de la cantidad a comparar con la cantidad de comparación, lo que es análogo a la desviación media cuadrática y al error medio medio cuadrático. Se cálcula de la forma siguiente:

$$SS_{tot} = \sum_{i=1}^{n} (y_i - f)^2 \tag{3.7}$$

De esta manera se obtiene una medida de que tan ajustados están los datos al modelo inicial.

Finalmente, es importante aclarar que no se valida contra el fenómeno real, no se usan datos históricos de erupciones u otros datos vulcanológicos. Se valida contra un modelo que se considera válido pues ha sido presentado en la literatura, el modelo SCIARA [9]. Cabe anotar que este utiliza otro mecanismo diferente para calcular los flujos de lava, y que no es el único enfoque alternativo posible, ya que existen muchos mecanismos para simular

este fenómeno; se espera así una desviación de los datos, incluso en la forma general de los perfiles. Lo importante es que el comportamiento general sea análogo, y por eso esta medida inspirada en el coeficiente de determinación se considera adecuada.

Se comparan primero los resultados del programa serial para los pasos de tiempo $t = 0s$, $10s$, $60s$, $120s$, $180s$, $240s$, $300s$, $360s$, con los resultados arrojados por el modelo de referencia. Los resultados se muestran en la tabla 2.

N	**0**	**10**	**60**	**120**
R2 Z	1	0.99397	0.96202	0.92368
R2 Temp	1	0.9969	0.95885	0.91319
N	**180**	**240**	**300**	**360**
R2 Z	0.88534	0.847	0.80866	0.77032
R2 Temp	0.86753	0.82187	0.77621	0.73055

Tabla 2: R2 obtenido comparando contra el modelo de referencia

Se nota una tendencia a la disminución del valor de R2 a medida que se avanza en el tiempo. Este descenso se considera normal debido a que el mecanismo de simulación del modelo de referencia es diferente al usado, entonces los valores numéricos no serán los mismos. El comportamiento es similar, y los R2 siguen estando en el rango de lo aceptable, luego los resultados se aceptan como válidos.

Después, se comparan los resultados del serial con los modelos de los diversos mapeos, que serán explicados en el siguiente capítulo, para los pasos de tiempo $t = 0s$, $10s$, $60s$, $120s$, $180s$, $240s$, $300s$, $360s$. Los resultados se presentan en la tabla 3

En el siguiente capítulo una evaluación de desempeño de la simulación será presentada. Para esta se establecerán varios mapeos de datos usando un número definidos de jerarquías de memoria y se compararán los resultados de tiempos de ejecución para estos. Se extraerán conclusiones sobre cual de los mapeos resulta más adecuado para el problema de acuerdo a los resultados.

N	Comparado con Serial	Z	T	N	Comparado con Serial	Z	T
0	CUDA	1.0000	1.0000	**180**	CUDA	0.9497	0.9534
	CUDA con memoria mapeada	1.0000	1.0000		CUDA con memoria mapeada	0.9497	0.9534
	CUDA Alterno	1.0000	1.0000		CUDA Alterno	0.9497	0.9534
	CUDA Multi-GPU	1.0000	1.0000		CUDA Multi-GPU	0.9497	0.9534
	CUDA Multi-GPU (memoria mapeada)	1.0000	1.0000		CUDA Multi-GPU (memoria mapeada)	0.9497	0.9534
	GCC	1.0000	1.0000		GCC	1.0000	1.0000
10	CUDA	0.9976	1.0000	**240**	CUDA	0.9328	0.9367
	CUDA con memoria mapeada	0.9976	1.0000		CUDA con memoria mapeada	0.9328	0.9367
	CUDA Alterno	0.9976	1.0000		CUDA Alterno	0.9328	0.9367
	CUDA Multi-GPU	0.9976	1.0000		CUDA Multi-GPU	0.9328	0.9367
	CUDA Multi-GPU (memoria mapeada)	0.9976	1.0000		CUDA Multi-GPU (memoria mapeada)	0.9328	0.9367
	GCC	1.0000	1.0000		GCC	1.0000	1.0000
60	CUDA	0.9835	0.9869	**300**	CUDA	0.9159	0.9199
	CUDA con memoria mapeada	0.9835	0.9869		CUDA con memoria mapeada	0.9159	0.9199
	CUDA Alterno	0.9835	0.9869		CUDA Alterno	0.9159	0.9199
	CUDA Multi-GPU	0.9835	0.9869		CUDA Multi-GPU	0.9159	0.9199
	CUDA Multi-GPU (memoria mapeada)	0.9835	0.9869		CUDA Multi-GPU (memoria mapeada)	0.9159	0.9199
	GCC	1.0000	1.0000		GCC	1.0000	1.0000
120	CUDA	0.9666	0.9701	**360**	CUDA	0.8990	0.9032
	CUDA con memoria mapeada	0.9666	0.9701		CUDA con memoria mapeada	0.8990	0.9032
	CUDA Alterno	0.9666	0.9701		CUDA Alterno	0.8990	0.9032
	CUDA Multi-GPU	0.9666	0.9701		CUDA Multi-GPU	0.8990	0.9032
	CUDA Multi-GPU (memoria mapeada)	0.9666	0.9701		CUDA Multi-GPU (memoria mapeada)	0.8990	0.9032
	GCC	1.0000	1.0000		GCC	1.0000	1.0000

Tabla 3: R2 comparando contra el serial

4. EVALUACIÓN DE DESEMPEÑO DEL MODELO DE SIMULACIÓN

4.1. DESCRIPCIÓN DE LAS MÉTRICAS UTILIZADAS PARA CARACTERIZAR EL DESEMPEÑO DEL MODELO DE SIMULACIÓN

Después de creado el modelo de simulación, tanto su interpretación desde la matemática, como su implementación en código secuencial, se procede a construir el experimento para determinar la respuesta a la pregunta principal que motivó este trabajo: "¿Cuál es el mapeo adecuado para implementar un modelo de simulación para el fenómeno de interés?". Esto se clarifica con varias preguntas y respuestas:

El objetivo principal del experimento es: Determinar cual de entre varios esquemas de mapeo sobre GPU produce un mayor desempeño para el programa de simulación creado a partir del modelo de simulación descrito en el capítulo anterior. La decisión de reducir las opciones a los mapeos sobre GPU obedece a la disponibilidad de una infraestructura de cálculo avanzado de ese tipo en la universidad.

La medida principal de desempeño fue el tiempo de ejecución, como factor de medición inicial. Habiendo definido el tiempo hace falta aclarar que este puede ser medido de maneras diversas. En sistemas operativos se consideran tres formas de medir el tiempo de ejecución[14]:

- Real: Es el tiempo entre el inicio y el final del llamado. Esto implica que se incluye el tiempo usado por otros procesos en ejecución y el tiempo usado por el proceso mientras está bloqueado.
- User: Es la cantidad de tiempo de CPU usado en código de modo de usuario (fuera del kernel) dentro del proceso. Este tiempo no incluye lo que el proceso dura bloqueado ni el tiempo usado por otros procesos.
- System: Es el tiempo de CPU en modo kernel usado por el proceso. Solo incluye tiempo de CPU.

User+Sys dice el tiempo de CPU total. El tiempo real y el tiempo user+system, deberían ser iguales según la intuición, pero en la practica no lo son. En primer lugar User y Sys contabilizan todo el tiempo de CPU, entonces si el sistema es multicore, el tiempo reportado puede ser mayor que el tiempo reportado por Real, ya que puede incluir el tiempo de hilos que corren concurrentemente. En el caso de sistemas heterogéneos la relación es muy difícil de clarificar, ya que el tiempo usado por la GPU no será reportado directamente, pero en procesos como las copias entre memorias la CPU será usada en cierta cantidad. Además estas relaciones de uso de GPU y CPU en procesos varían bastante con relación a

la arquitectura de la GPU.

Este valor de tiempo no resulta definitivo para el análisis de desempeño, ya que hay partes no directamente relacionadas con los cálculos asociados a la simulación (pero que no se pueden eliminar) que suman tiempo: Creación de archivos de salida, exportación de gráficas, operaciones en disco en general). El problema con estas es que no dependen del mapeo. Los accesos a disco, en particular, son siempre secuenciales. De esta suerte, este análisis secundario, aunque cabe recordar que es la única manera de comparar directamente las aplicaciones, si se usa el tiempo Real. Esto se dice debido a que cualquier tipo de perfilado genera un sobrecosto computacional, así resulte mínimo, y los mecanismos de perfilados para CPU y GPU son diferentes; es difícil concluir algo de mecanismos diferentes, y además iría en contra de la independencia de factores que debe tener un experimento.

Una medida de desempeño que resulta interesante es el tiempo específico de cálculo, es decir, la suma de la duración de todos los tiempos involucrados en el cálculo de la simulación. Acá se distinguen dos experimentos, uno para CPU específicamente para el código serial y otro para GPU, debido esto a los diferentes mecanismo de perfilado.

En el caso de la CPU: Se aísla el tiempo de cálculo utilizando las funciones de la librería de C <time.h>. Se calcula el tiempo empleado para realizar los cálculo específicos de la simulación.

En el caso de la GPU: Se utiliza el perfilador de NVIDIA y se extraen los tiempos de duración de las llamadas a los Kernel, lo que permite caracterizar el desempeño, descubriendo los puntos de ejecución en los que se gasta más tiempo. Esto permite comparar el desempeño de los diferentes mapeos y también identificar posibilidades de optimización para el futuro.

4.2. DESARROLLO DE LOS EXPERIMENTOS

Primero se deben definir los mapeos a utilizar y luego los escenarios de simulación, para garantizar la independencia de los factores.

4.2.1. Descripción de los mapeos utilizados

Los mapeos escogidos se presentan en la tabla 4, en la página 44.

No fue posible usar manejo de memoria unificada debido a la arquitectura del clúster GUANE. Las GPU del clúster no soportan esa característica pues tiene compute capability

Nombre	*Descripción*
cuda	1 GPU, manejo de memoria normal para Fermi
cuda_pin	1 GPU, manejo de memoria mapeada en la RAM
cuda_alterno	1 GPU, manejo de memoria normal para Fermi, aglomeración alternativa usando bloqueos
cuda_normal_multi	Multiples GPU (2, 4, 8), manejo de memoria normal para Fermi
cuda_pin_multi	Multiples GPU (2, 4, 8), manejo de memoria mapeada para Fermi

Tabla 4: Mapeos escogidos para el estudio

2.0 al ser de arquitectura Fermi. La arquitectura Kepler si la soporta, pero actualmente no hay ninguna máquina lista para producción en el centro de supercomputación de la universidad.

Se evitó usar optimizaciones de código que no pudieran ser reproducidas a través de todos los diferentes mapeos.

4.2.2. Escenarios de Simulación utilizados

Debido a que el factor a comparar para el estudio es el mapeo, el escenario de simulación es fijo para todas las pruebas. Se utilizaron los mismo parámetros para las simulaciones de validación de los resultados, resumidos en la tabla1 (página 34). Se agregan algunos datos de parámetros de entrada: Cabe recordar que si cada celda mide un metro cuadrado, el área de estudio para el experimento equivaldría a un Kilometro cuadrado aproximadamente. Los 3600 segundos representan una hora de duración para el evento. Ambos valores se consideran apropiados para un estudio con un alcance como este.

4.2.3. Procedimiento de ejecución de las pruebas

Con el escenario definido en la sección anterior se realizan 10 corridas completas por mapeo para limitar el efecto de la variabilidad inherente a la medición. Este mecanismo resulta consistente con los lineamientos expresados en la sección 2.3, en la página 23, que trata acerca de Experimentos en Computación. Además, las técnicas de selección de muestras tradicionales en estadísticas no se transfieren de manera directa en este caso, ya que no

Descripción	*Valor*	*Unidades*
tamaño en y	1024	m
tamaño en x	1024	m
Temperatura incial de extrusión	1500	K
Tasa de extrusión	0.15	m^3
Viscosidad (η)	850	Pa/s
Fluencia (S_y)	225	N/m^2
Puntos de extrusión	1	-
Coordenadas puntos de extrusión	{511,511}	-
Mapa de altitud	0.00 (para todas las celdas)	m
Pasos de simulación	3600	s

Tabla 5: Parámetros de entrada para el escenario de simulación

existe una población per se de la cual extraer una muestra al ser este un fenómeno abstracto. Además se considera el número 10 un estándar de facto, aunque sin una base teórica sólida[15]. La selección de una valor así es lo que Feitelson considera prácticas comunes en la disciplina[11].

Corridas las simulaciones se obtienen los datos y se construyen tablas. Se extraen los promedios y desviaciones estándar, así como una aproximación a un intervalo de confianza de 95 % para los tiempos medidos, usando una distribución normal.

4.2.4. Descripción detallada de la infraestructura utilizada en las pruebas

El clúster que se utilizó para realizar el experimento consta de 16 nodos de cómputo. Las características de los nodos usados se resumen en la tabla 6. Las características de las GPU usadas aparecen en la tabla 7.

Especificación	*Valor*
CPU	2 x Intel(R) Xeon(R) CPU E5645 @ 2.40GHz
Núcleos por CPU	6
Hilos	2 x core
Total núcleos	24
Memoria total	107471960 kB (104 Gb)
GPU	8 x NVIDIA Tesla M2075

Tabla 6: Características de los nodos de prueba

Especificación	*Valor*
GPU	NVIDIA Tesla M2075
Núcleos	448
Ancho de banda de memoria	150 Gb/s
Memoria	6 Gb GDDR5
Velocidad de la memoria	1.55 GHz
Desempeño (precisión sencilla)	1.03 TFlops
Desempeño (precisión doble)	515 GFlops
Arquitectura	Fermi
Compute Capability	2.0

Tabla 7: Características de los GPU de los nodos de prueba

4.3. RESULTADOS OBTENIDOS PARA LOS EXPERIMENTOS

Todos los datos de tiempos de ejecución se encuentran disponibles por solicitud al autor o en la página del forge del Centro de Supercomputación y Cálculo Científico de la Universidad Industrial de Santander[16][1].

4.3.1. Medición del tiempo total de ejecución

Se tomaron los tiempos mediante la herramienta time de linux, y se tuvo en cuenta principalmente el tiempo real, ya que los otros tiempos no permiten una comparación directa al haber paralelismo involucrado. Los resultados de la medición del tiempo se presentan en la tablas 8 y 9, en la páginas 47 y 48. Los datos se presentan en dos gráficos, uno para los tiempos promedios (6) y otro para los valores de aceleración (7).

De los datos obtenidos se pueden extraer ciertas conclusiones:

- Los valores de speedup son mayores que uno para dos mapeos, CUDA y CUDA Alterno.
- Los valores de tiempo de ejecución para los mapeos sobre memoria mapeada (etiquetada como pin), sean multi GPU o no, son excesivamente altos.
- Los mapeos de memoria mapeada tienen un valores de tiempo total decrecientes con relacion al número de GPU. Este comportamiento es esperado, pero no con esos rendimientos tan bajos.

[1] En dicho sitio web se puede acceder al proyecto llamado SCaLaF, en el que está disponible todo lo relacionado con este trabajo

		Promedio	Speedup
Real	CUDA	***3427.3118***	**1.4825**
User		***2803.5007***	**1.6289**
System		***126.5013***	**0.8572**
User + System		***2930.002***	**1.5956**
Promedio		***3178.6569***	**1.5346**
Real	CUDA_Multi 2	***4533.6215***	**1.1208**
User		***3986.7053***	**1.1454**
System		***197.4588***	**0.5492**
User + System		***4184.1642***	**1.1173**
Promedio		***4358.8928***	**1.1191**
Real	CUDA_Multi 4	***5101.4677***	**0.996**
User		***4479.4885***	**1.0194**
System		***277.9515***	**0.3901**
User + System		***4757.44***	**0.9827**
Promedio		***4929.4538***	**0.9896**
Real	CUDA_Multi 8	***6272.8542***	**0.81**
User		***5488.7603***	**0.832**
System		***448.2342***	**0.2419**
User + System		***5936.9945***	**0.7874**
Promedio		***6104.9243***	**0.799**
Real	CUDA_pin	***33061.9328***	**0.1537**
User		***24319.5072***	**0.1878**
System		***8307.8193***	**0.0131**
User + System		***32627.3265***	**0.1433**
Promedio		***32844.6297***	**0.1485**

Tabla 8: Tiempos de ejecución para todos los mapeos, 3600 pasos, parte 1

		Promedio	Speedup
Real	CUDA_pin_Multi 2	***18622.2053***	**0.2729**
User		***13956.7108***	**0.3272**
System		***4292.0193***	**0.0253**
User + System		***18248.7302***	**0.2562**
Promedio		***18435.4678***	**0.2646**
Real	CUDA_pin_Multi 4	***11122.7717***	**0.4568**
User		***8452.4502***	**0.5403**
System		***2330.5508***	**0.0465**
User + System		***10783.001***	**0.4336**
Promedio		***10952.8863***	**0.4454**
Real	CUDA_pin_Multi 8	***6482.4672***	**0.7838**
User		***5054.85***	**0.9034**
System		***1071.9332***	**0.1012**
User + System		***6126.7832***	**0.763**
Promedio		***6304.6252***	**0.7737**
Real	CUDA_alterno	***3979.4125***	**1.2769**
User		***3001.3428***	**1.5215**
System		***122.651***	**0.8841**
User + System		***3123.9938***	**1.4965**
Promedio		***3551.7032***	**1.3734**
Real	Serial	***5081.132***	**1**
User		**4566.5627**	**1**
System		**108.435**	**1**
User + System		**4674.9977**	**1**
Promedio		***4878.0648***	**1**

Tabla 9: Tiempos de ejecución para todos los mapeos, 3600 pasos, parte 2

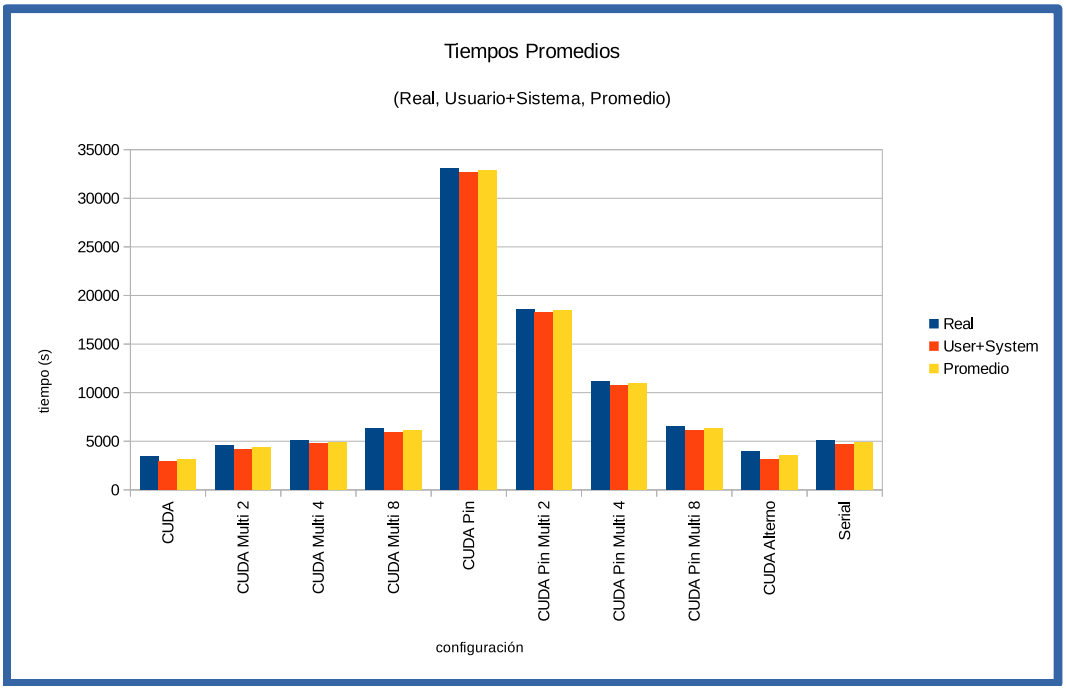


Figura 6: Valores de tiempo total promedio para cada mapeo

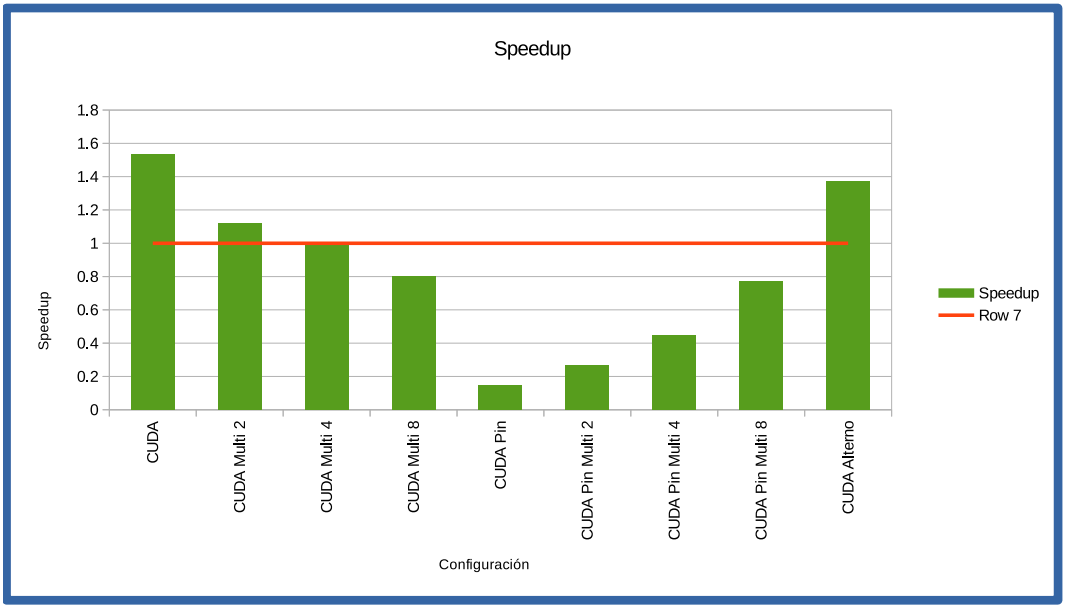


Figura 7: Valores de aceleración para cada mapeo

- Los mapeos multi GPU sin memoria mapeada tienen valores de tiempo total con una tendencia creciente, directamente proporcional al número de GPU.

Además, de las tablas 8 y 9 específicamente se pueden extraer las siguientes observaciones:

- Para los mapeos con memoria de GPU normal el tiempo User es creciente también.

Teniendo en cuenta que el tiempo user no tiene en cuenta el tiempo de GPU esto implica que efectivamente se hacen más operaciones de copia de posiciones de memoria principal al aumentar el número de GPU empleadas. Esto refuerza la hipótesis de que el uso de más tarjetas requiere más operaciones en CPU para reunir los datos calculados en las diferentes GPU.

- El tiempo User+System es siempre menos que el tiempo Real. Esto demuestra que hay posibilidad de incrementar el rendimiento usando una combinación de múltiples núcleos de CPU a la vez que se usan las GPU, pero esta situación se sale del planteamiento incial del experimento.
- El bajo valor de speedup, incluso para los casos en los resulta mayor que 1, se explica por el número de puntos de ejecución en los que hay necesidad de reunir los resultados de los hilos en ejecución. Esto es consistente con el análisis de paralelismo realizado en el capítulo 4.

Estos comportamientos generaron la necesidad de analizar más a fondo la ejecución de las simulaciones. De esta manera de generaron otros experimentos para aclarar las observaciones.

4.3.2. Medición del tiempo cálculo para el programa serial

Después de obtener los resultados de tiempo total, que resultan altamente contra intuitivos y alejados de las previsiones, se hace necesario analizar más a fondo. Se diseñó un experimento para extraer características del programa serial. Se aisló la parte correspondiente al cálculo de la simulación de las demás operaciones (ejemplo lectura de archivos, escritura en disco de resultados, etc); de esta manera se puede estimar claramente la porción que debe ser ejecutada siempre de manera serial. La parte correspondiente al cálculo se puede paralelizar, pero no se puede concluir directamente si en su totalidad o en que medida, ya que hay características del algoritmo que fuerzan la ejecución serial, referenciados en la página 3.2.1. Los resultados del experimento se muestran en la tabla 10 (p. 50). De los resultados del experimento se puede concluir que aproximadamente el

	Promedio	**Desv Est**	**Cota inferior**	**Cota superior**
Tiempo (Total C)	**3747.9820**	**3.6303**	3740.8667	3755.0973
Tiempo (simulaciones)	**2069.4200**	**4.1711**	2061.2448	2077.5952
Relación Calculo/Total	**0.5521**	**0.0008**	0.5507	0.5536
Real	**5065.8071**	**36.6329**	4994.0079	5137.6063
User	4574.1719	14.3432	4546.0598	4602.2840
Sys	99.6512	0.7988	98.0857	101.2167
User + System	4673.8231	15.0637	4644.2988	4703.3474
Promedio	**4869.8151**	**20.4582**	4829.7177	4909.9125

Tabla 10: Análisis del Serial

45 % del tiempo de ejecución del programa base se usa en operaciones que no se pueden paralelizar, al menos de manera directa. Esto explica en parte los resultados tan bajos de la aceleración, pero no totalmente. Además, hace falta una explicación para el comportamiento contra-intuitivo cuando aumenta el número de las tarjetas de vídeo y cuando se usa la memoria mapeada. En la siguiente sección se dará una descripción más detallada del comportamiento de los accesos a memoria y llamados de los kernels. Esto permitirá caracterizar el comportamiento de las simulaciones de manera más específica respondiendo a los interrogantes planteados.

4.3.3. Medición del tiempo cálculo por mapeo sobre GPU

Antes de iniciar con la presentación de los perfiles resumidos de ejecución en las GPU, que ofrecen una visión más detallada del comportamiento, es necesario analizar el caso particular de la memoria mapeada.

Para el caso de la memoria mapeada, mucho de la disparidad de los resultados obtenidos frente a los esperados se puede explicar mediante consideraciones técnicas. En primer lugar, cuando se usa la memoria mapeada se está usando en todo momento la memoria principal del computador, accesandola mediante (un mecanismo similar al) DMA; de esta suerte, se evitan operaciones de copia entre la RAM y la VRAM antes de cada kernel, pero se debe acceder posiciones de memoria RAM desde la GPU cada vez que se referencian en el código. No solo es más rápido para los multiprocesadores de la GPU accesar memoria que se encuentra en el bus de datos interno, sino que además se debe trabajar a la velocidad de la memoria principal, sin contar con la penalización por el salto de bus mencionado anteriormente. Los nodos utilizados para las corridas utilizan memoria DDR3-1333 de tipo RDIMM, de acuerdo a las especificaciones del fabricante[2]. Esto quiere decir que se usa memoria con menos ancho de banda, además de estar localizada en un espacio de memoria diferente. Las características de la memoria DDR3-1333 y la GDDR5, la que es usada en las GPU, se presentan en la tabla 11.

	Ancho de banda	**Reloj de memoria**
DDR3-1333	10.667 GB/s	1.67 MHz
GDDR5	150 GB/s	1500 Mhz

Tabla 11: Datos memoria DDR3 vs GDDR5

Así, para un número grande de accesos a memoria, el desempeño se penaliza en demasía. Este mecanismo de memoria mapeada parece resultar más útil en casos donde haya una pequeña cantidad de datos que deba ser accesada desde varios contextos de ejecución diferentes y que además deban ser actualizados durante la ejecución; como memoria principal para los cálculos en los kernel no es recomendada. La memoria asegurada frente a la paginación ("page-locked") o también llamada clavada ("pinned"), en su versión más pura, no se usa en el proyecto, pero se usa para acelerar las copias entre los espacios de memoria mediante instrucciones cudaMemcpy. No se puede combinar con la memoria

[2]http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=c04123322

mapeada pues esta es un subtipo de la primera y además no se usa copia de memoria entre los espacios de memoria sino se extrae un puntero en el espacio de la GPU mediante cudaGetDevicePointer[13].

Para continuar esclareciendo los resultados, se tomaron perfiles de cada mapeo, repitiendo las ejecuciones diez veces por mapeo. Los resultados son de bastante tamaño en el impreso, así que no se transcriben en su totalidad, pero están disponibles en la página web del Centro, en la sección de documentos del proyecto llamado "scalaf"[16].

En la página 52, en la tabla 12 se presenta un resumen de los perfiles extraídos, en este caso, sumando todos los tiempos de llamada de cada simulación, por mapeo. Los tiempos de llamada son promedios de los resultados arrojados por el perfilador. No hay que confundir que los datos presentados son el promedio de diez ejecuciones por mapeo (de ahí el AVG en la primera columna) con el promedio presentado en la sexta columna, que se refiere a la duración promedio de las llamadas a todos los kernels para todas las ejecuciones por cada mapeo; este segundo se relaciona con los valores mínimos y máximos de tiempo por llamada, presentados en las columnas siete y ocho respectivamente. Cabe recordar que para el caso de los kernels se suman los valores de tiempos de todos los que están corriendo en cada tarjeta, luego el resultado en el caso multi GPU es en realidad aproximadamente el reportado divido por el número de GPU. Esta aproximación es algo simplista, pero permite dar una idea más acertada del comportamiento.

De mayor utilidad para evaluar el desempeño son los tiempos por llamada. Estos se reportan tres veces, un valor mínimo, uno máximo, y uno promedio. En este caso se nota el tiempo reducido de los kernels al usar múltiples GPU, pero también se nota el sobrecosto de las operaciones de memoria. Para el caso de la duración de las llamadas, los datos se

	Mapeo	Time(%)	Time	Call	Avg	Min	Max
AVG	**CUDA**	99.99%	304.04730458	36001	0.0467706976	0.0454524762	0.068447978
	CUDA Multi 2	100.00%	670.30238	86401	0.04450377	0.04235257	0.06610449
	CUDA Multi 4	100.00%	1251.2787981	172801	0.037950211	0.035085612	0.06185145
	CUDA Multi 8	100.00%	2276.1634312	345601	0.036087707	0.032529862	0.058750182
	CUDA Pin	100.00%	30036.92634	18000	8.3789583	8.12753968	9.0993566
	CUDA Pin Multi 2	100.00%	40001.214013	28805	7.049325	6.7929734	7.167096
	CUDA Pin Multi 4	100.00%	70006.277905	57617	4.943389	3.810121	5.253436
	CUDA Pin Multi 8	100.00%	100023.29141	115265	3.521478	2.49279	4.541227
	CUDA Alterno	100.00%	173.25418874	10801	0.0779492679	0.057144206	0.0925852927

Tabla 12: Tiempos totales por mapeo, perfilador

presentan también mediante un gráfico, en la imagen 8. También se presenta una imagen con las duraciones de las llamadas para los mapeos sin memoria mapeada, la imagen 9

Los resultados por llamada, que incluyen kernels y transferencias de memoria se presentan en la tabla 13, que aparece en la página 54. Esta tabla, y la tabla con los datos completos a partir de los cuales se calculan los promedios que aparecen en la anterior, se pueden encontrar en el material adjunto al proyecto, disponible en la página web del forge del Centro

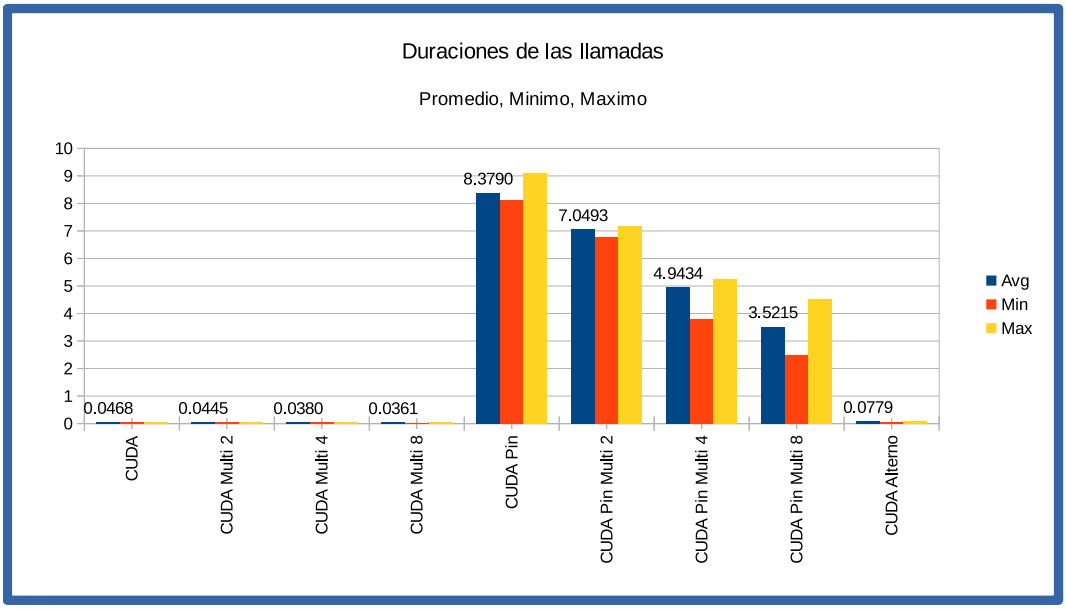


Figura 8: Duración de las llamadas, promedio

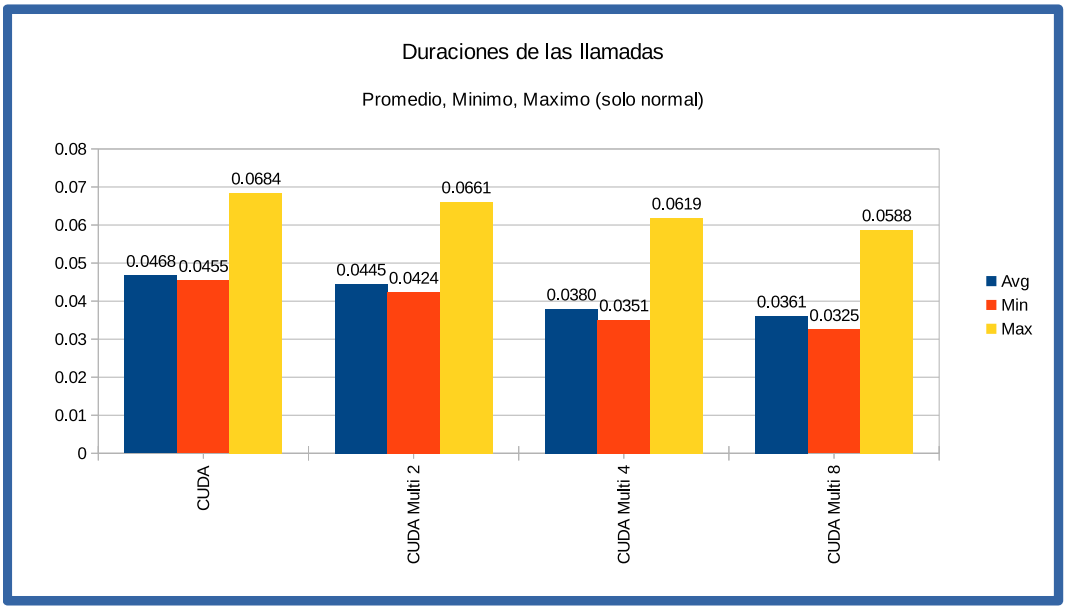


Figura 9: Duración de las llamadas, promedio, sin memoria mapeada

de Supercomputación y Cálculo Científico de la Universidad Industrial de Santander[3] [16]. Analizando los datos obtenidos se puede extraer varias conclusiones:

- En el caso de las corridas multi GPU, además de correspondiente a CUDA normal, se puede notar el alto costo de las transferencias entre los espacios de memoria, en particular del Dispositivo (La GPU) al Host. Esto se debe a la necesidad de pasar

[3] En dicho sitio web se puede acceder al proyecto llamado SCaLaF, sección documentos, archivo tiempos.ods (formato libreOffice Calc)

Mapeo	Time(%)	Time	Call	Avg	Min	Max	Name
CUDA	76.88%	233.7672	18000	0.0129871	0.0120032	0.0336037	**[CUDA memcpy DtoH]**
	8.39%	25.5012	3600	0.00708366	0.00682682	0.00795412	KernelPrincipal2
	7.01%	21.3038	3600	0.00591772	0.00585785	0.00608811	KernelPrincipal1
	5.08%	15.43691	3600	0.00428804	0.00427959	0.00429739	KernelPrincipal3
	2.64%	8.016449	3600	0.0022268	0.00221811	0.00223675	KernelInicializacion_t
	0.00%	0.0142653	1	0.0142653	0.0142653	0.0142653	[CUDA memcpy HtoD]
	0.00%	0.00748028	3600	2.08E-006	1.61E-006	0.000002608	[CUDA memcpy DtoD]
CUDA Multi GPU 2	55.08%	369.447	28800	0.012828	0.0112989	0.0330509	**[CUDA memcpy DtoH]**
	30.33%	203.1568	28800	0.00705406	0.00700009	0.00713164	KernelPrincipal2
	5.92%	39.62777	7200	0.00550388	0.00497778	0.00674495	KernelPrincipal1
	4.93%	33.00482	7200	0.004584	0.00456244	0.00462134	KernelPrincipal3
	2.54%	17.01929	7200	0.00236378	0.00235374	0.00237312	KernelInicializacion_t
	1.20%	8.035646	7200	0.00111605	0.00110562	0.00112854	[CUDA memcpy HtoD]
	0.00%	0.011054	1	0.011054	0.011054	0.011054	[CUDA memcpy DtoD]
CUDA Multi GPU 4	56.90%	712.1466	57600	0.0123638	0.010586	0.0332937	**[CUDA memcpy DtoH]**
	35.00%	437.7731	57600	0.00760023	0.00700065	0.00932787	KernelPrincipal2
	3.41%	42.68984	14400	0.00296457	0.00250961	0.0041581	KernelPrincipal1
	2.67%	33.35533	14400	0.00231635	0.00229987	0.00235131	KernelPrincipal3
	1.37%	17.14637	14400	0.00119073	0.00118043	0.0011986	KernelInicializacion_t
	0.65%	8.15661	14400	0.000566431	0.000560952	0.00057377	[CUDA memcpy HtoD]
	0.00%	0.0109481	1	0.0109481	0.0109481	0.0109481	[CUDA memcpy DtoD]
CUDA Multi GPU 8	57.57%	1200	115200	0.0126992	0.0108378	0.033536	**[CUDA memcpy DtoH]**
	38.39%	973.7124	115200	0.00845237	0.00699872	0.00933224	KernelPrincipal2
	1.69%	42.76044	28800	0.00148475	0.00126595	0.00238814	KernelPrincipal1
	1.33%	33.73163	28800	0.00117123	0.00116057	0.00119809	KernelPrincipal3
	0.69%	17.44189	28800	0.000605621	0.000598714	0.000611746	KernelInicializacion_t
	0.34%	8.505692	28800	0.000295336	0.000288908	0.000304766	[CUDA memcpy HtoD]
	0.00%	0.0113792	1	0.0113792	0.0113792	0.0113792	[CUDA memcpy DtoD]
CUDA Pin	29.52%	9000	3600	2.473081	2.263065	3.08361	**KernelPrincipal3**
	27.81%	8000	3600	2.330036	2.327958	2.331723	KernelPrincipal2
	23.37%	7000	3600	1.958107	1.920372	2.059719	KernelPrincipal1
	19.19%	6000	3600	1.607477	1.606581	1.608206	KernelInicializacion_t
	0.12%	36.92634	3600	0.0102573	0.00956368	0.0160986	[CUDA memcpy HtoH]
CUDA Pin Multi 2	28.95%	10000	7200	1.970547	1.962628	1.981717	**KernelPrincipal3**
	27.71%	10000	7200	1.885906	1.875771	1.897858	KernelPrincipal2
	23.42%	10000	7200	1.5943	1.585719	1.605359	KernelPrincipal1
	19.92%	10000	7200	1.355768	1.35202	1.362917	KernelInicializacion_t
	0.00%	1.214013	5	0.242804	0.0168354	0.319245	[CUDA memcpy HtoH]
CUDA Pin Multi 4	29.87%	20000	14400	1.366492	1.2894	1.383945	**KernelPrincipal3**
	27.50%	20000	14400	1.258189	1.022475	1.334242	KernelPrincipal2
	23.09%	20000	14400	1.056211	0.820774	1.133623	KernelPrincipal1
	19.53%	10000	14400	0.893208	0.660517	0.964902	KernelInicializacion_t
	0.01%	6.277905	17	0.369289	0.016955	0.436724	[CUDA memcpy HtoH]
CUDA Pin Multi 8	28.51%	30000	28800	0.902211	0.702894	1.173614	**KernelPrincipal3**
	28.13%	30000	28800	0.889892	0.694396	1.152132	KernelPrincipal2
	24.07%	20000	28800	0.761373	0.595615	0.981897	KernelPrincipal1
	19.27%	20000	28800	0.609673	0.483878	0.780136	KernelInicializacion_t
	0.03%	23.29141	65	0.358329	0.016007	0.453448	[CUDA memcpy HtoH]
CUDA Alterno	59.08%	102.3568	3600	0.0284326	0.0168307	0.0394248	**[CUDA memcpy DtoH]**
	40.90%	70.86011	3600	0.0196832	0.0104805	0.023325	KernelInicializacion_t
	0.02%	0.0298314	1	0.0298314	0.0298314	0.0298314	[CUDA memcpy HtoD]
	0.00%	0.00744734	3600	2.07E-006	0.000001606	4.09E-006	[CUDA memcpy DtoD]

Tabla 13: Tiempos promedio por llamada

los datos al espacio de la CPU para reconstruir la matriz debido a la imposibilidad de acceder a direcciones de memoria de otros contextos de ejecución; Ya que los apuntadores creados en el contexto de una GPU no pueden ser accedidos desde el de otra, es necesario transferir los datos a la memoria principal para que puedan ser reunidos en ese espacio, y de esta manera reconstruir la matriz completa.

- En los mapeos sin memoria mapeada multi GPU el menor tamaño de datos a procesar para cada kernel es compensado por el uso de streams. Los streams permiten sincronizar las operaciones de copia y kernel en múltiples GPU pero consumen recursos. Además, entre cada uno de los kernels es necesario sincronizar la ejecución, generando una barrera que limita el paralelismo efectivo. Esto es parte inherente del algoritmo.
- En el caso de la memoria mapeada, la ausencia de operaciones de copia entre espacios de memoria es compensada por la larga duración de la ejecución de los kernels. Interesante resulta el que no parece haber penalización directa por el número de GPU, luego probar en un sistema con espacios de memoria compartidos podría resultar interesante como trabajo futuro.
- El CUDA alterno, que se genera por una diferente aglomeración, da resultados buenos si bien no los mejores, pero teniendo en cuenta que limita las copias de memoria entre espacios, promete cierta escalabilidad con relación al tamaño del problema. El inconveniente principal es que no hay una forma trivial de extender ese mapeo a múltiples GPU.
- El principal inconveniente del CUDA alterno es la necesidad de implementar barreras con el comando __threadfence(). Esto genera un retardo fijo, pero que al crecer el problema debería ser menos significativo.

Para visualizar mejor la escalabilidad del problema se seleccionaron los mapeos que produjeron aceleraciones positivas, en este caso CUDA y CUDA Alterno, y se creo una curva de aceleración versión tamaño del problema mediante un nuevo experimento. Se escogió tamaño del problema debido a que la cantidad de elementos de procesamiento permanece constante. El nuevo experimento consiste del mismo escenario de simulación pero variando el número de pasos a calcular. Esto se realizó de esa manera para obtener resultados rápidos, ya que las corridas de 3600 pasos demoraban bastante. Se corrieron conjuntos de 512×512, 1024×1024 y 2048×2048 datos por cada mapeo. Los resultados fueron se muestran en la tabla 14.

La tabla 14 muestra los valores de speedup (columna 2) y el tiempo de ejecución (columna 3) para cada uno de los mapeos seleccionados en tres valores de tamaños de problema. Se observa gran similitud entre los resultados para los mapeos que utilizan CUDA, además de valores de aceleración significativos. Esto contrasta con los resultados obtenidos para los experimentos con 3600 pasos de tiempo. A continuación se construye una gráfica con los datos del experimento.

En la figura 10, construida con los datos de la tabla 14, se muestra la curva de aceleración.

Nombre	Speedup	Promedio
Serial 512	1	46.9744
Serial 1024	1	168.0364
Serial 2048	1	613.2723
Cuda 512	1.9982303897	23.508
Cuda 1024	5.7835493664	29.0542
Cuda 2048	12.0328509904	50.9665
Cuda Alt 512	2.0066983357	23.4088
Cuda Alt 1024	5.7897866857	29.0229
Cuda Alt 2048	12.1430200716	50.5041

Tabla 14: Tiempos de ejecución, Serial, CUDA, CUDA Alterno

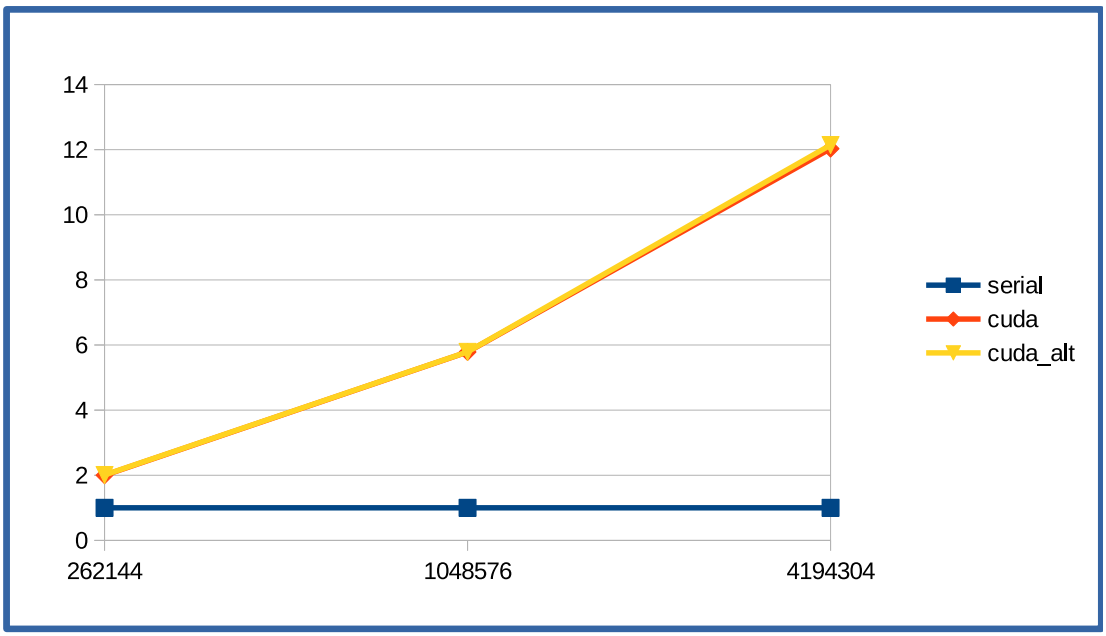


Figura 10: Curva de aceleración para CUDA y CUDA Alterno

Los resultados van en contra de las expectativas, como se expresó en el párrafo anterior. Para un número de pasos igual a 120 se obtiene mayor aceleración. Esto se puede explicar por la cantidad de ceros en la matriz, que permiten ahorrar una gran cantidad de cálculos, algo que se había mencionado en el análisis asintótico del desempeño estimado del modelo, en la sección 3.3 en la página 37. En las celdas donde no hay lava acumulada no se realizan las operaciones correspondientes al cálculo de los flujos de lava y calor, solo las comparaciones para verificar la cantidad presente. De esto se desprende que al aparecer más celdas con lava al avanzar la simulación, el tiempo empleado para calcular cada paso de tiempo t aumenta. Así, se puede concluir que la distribución de los cráteres y de las zonas cubiertas de lava en la simulación pueden afectar significativamente el rendimiento. Otro resultado para destacar es que se observa que la diferencia entre CUDA y CUDA Alterno se hace mínima a medida que se varía el tamaño del problema.

De los experimentos conducidos se puede concluir que el problema puede obtener beneficios de la paralelización mediante GPU, pero que la mejora del desempeño obtenida no es la mejor. Una aceleración de una sola cifra no se considera una gran mejora. Como trabajo futuro se recomienda usar paralelización por múltiples hilos de CPU y comparar los resultados. Finalmente, los resultados obtenidos presentados en la tabla 14 sugieren la relevancia de investigar algoritmos con balanceo de carga dinámico o en general algoritmos con paralelismo dinámico para este problema.

Finalmente, para mejorar el desempeño de los mapeos sobre solo una GPU se deben implementar varias optimizaciones a nivel de código, las que no se incluyeron en los programas aquí evaluados para igualar las condiciones de los diferentes mapeos, como por ejemplo uso de memoria fijada en la paginación (la memoria llamada Pinned, diferente de la mapeada) para acelerar las copias de RAM a memoria del dispositivo. Usar copias asíncronas en el proceso de extracción de los resultados para los pasos intermedios de simulación podría ayudar al rendimiento, pero se debería probar para identificar la penalización del bloqueo necesario para garantizar la integridad de esta copia. También se debería probar la implementación con memoria compartida (Shared Memory).

En el siguiente capítulo se presentan las conclusiones generales del trabajo y se plantean posibles caminos para la continuación del mismo.

5. CONCLUSIONES Y TRABAJO FUTURO

De la realización del presente trabajo se obtienen las siguientes conclusiones:

- Se construyó un modelo de simulación para el fenómeno del flujo de lava en dos dimensiones, usando una formulación particular de autómatas celulares. Durante el proceso de construcción se identificaron varios problemas conceptuales, el más importante de estos la existencia de varias metodologías y herramientas conceptuales que pueden servir de base al modelado de este tipo de fenómenos, y la viabilidad limitada pero suficiente de las mismas, que dificulta la comparación de los modelos construidos con estas.
- La construcción del modelo permitió el logro del avance esperado para este trabajo: La identificación de problemas tanto conceptuales como de implementación y ejecución de los modelos. Lo aprendido respecto del problema permitirá la construcción de modelos que se acerquen más al nivel de aplicación directa en la realidad.
- Se propusieron varios esquemas de mapeo de procesos sobre los elementos de procesamiento y sobre las diferentes jerarquías y espacios de memoria y se caracterizó el rendimiento de los diversos mapeos para las ejecuciones de las simulaciones sobre la arquitectura disponible, siendo de entre estos el que mejor rendimiento genera el que usa una única GPU y la memoria principal+VRAM sin mapeo directo por DMA; en segundo lugar quedó un mapeo sobre una única GPU con aglomeración diferente. Se concluye así que para este problema en particular el uso de múltiples GPU no es eficiente y resulta contraproducente.
- Al comparar los resultados de desempeño de los mapeos se obtuvo evidencia de que el problema no es apto para ser tratado en arquitecturas masivamente paralelas, esto debido a los resultados de speedup apenas superiores a 1; estos valores no se comparan a los usualmente obtenidos en la literatura para problemas que se consideran casos de éxito.

Como posibles caminos para continuar el trabajo, y resultan varios pues el trabajo tiene varias direcciones en las cuales crecer al ser de interés para varias disciplinas, se proponen:

- En el momento de disponer de una infraestructura con tecnología Kepler, es decir con compute capability 3.0, realizar pruebas sobre un programa que haga uso de la memoria unificada para así obtener una comparación de la eficiencia de las simulaciones. Sería de especial interés el probar un mapeo sobre múltiples GPU con memoria unificada.

- De disponer de una infraestructura Maxwell, es decir con compute capability 5.0, realizar pruebas generando un sistema de carga variable, haciendo uso del paralelismo dinámico. En los dispositivos Fermi el balanceo de carga es automático dentro de una GPU, entonces con los nuevos dispositivos se podría probar un sistema de carga irregular, lo que resulta interesante teniendo en cuenta los resultados de la curva de aceleración para 120 pasos de tiempo, que muestra que el número de operaciones por paso aumenta con el número de celdas con valor de grosor de lava diferente a 0.
- Para poder esclarecer aún más los resultados de los ejecuciones con memoria mapeada se podría generar un experimento sobre una arquitectura con memoria VRAM en el mismo espacio de la memoria RAM, como por ejemplo dispositivos Tegra.
- Se debería ahondar en los métodos de simulación de mecánica de fluidos basados en autómatas celulares estocásticos. Se han sugerido en la literatura, y su implementación permitiría estudiar el rendimiento de un tipo diferente de algoritmos, con sus propiedades particulares.
- Diseñar una arquitectura software basada en componentes que permita realizar las simulaciones de manera más fácil, incluyendo componentes definidos para la lectura y escritura de datos en múltiples formatos, como por ejemplo formatos estándares de los SIG, así como un componente para visualización interactiva con modificación de parámetros.
- Como centro de investigación, ahondar en el tema de la reproducibilidad de los experimentos, generando o apropiándose de herramientas que ayuden en este propósito.
- Establecer un diálogo con los encargados de la maestría en geofísica de la universidad para buscar avenidas para el trabajo interdisciplinario partiendo de los resultados obtenidos en este trabajo.

REFERENCIAS BIBLIOGRÁFICAS

[1] I. Foster. Designing and building parallel programs. [Online]. Available: http://www.mcs.anl.gov/~itf/dbpp/text/node19.html#SECTION02350000000000000000

[2] F. Bell, *Geological Hazards: Their Assessment, Avoidance and Mitigation*. Spon Press, Taylor and Francis Group, 1999.

[3] H. Miyamoto and S. Sasaki, "Simulating lava flows by an improved cellular automata method," *Computers & Geosciences*, vol. 23, pp. 283–292, 1997.

[4] M. Dragoni, M. Bonafede, and E. Boschi, "Downslope flow models of a bingham liquid: Implications for lava flows," *Journal of Volcanology and Geothermal Research*, vol. 30, p. 21.

[5] A. Vicari, H. Alexis, C. D. Negro, M. Coltelli, M. Marsella, and C. Proietti, "Modeling of the 2001 lava flow at etna volcano by a cellular automata approach," *Environmental Modelling & Software*, vol. 22, pp. 1465–1471, 2007.

[6] P. Francis, *Volcanoes: A Planetary Perspective*, 1st ed. Oxford University Press, 1993.

[7] S. A. Gélvez, "Problemas computacionales asociados a la construcción de modelos de simulación basados en autómatas celulares en paralelo. caso de estudio: Evaluación de amenazas asociadas a flujos de lava volcánica como fluído bingham." propuesta de trabajo de investigación para obtener el título de Magister en Ingeniería de Sistemas e Informática.

[8] A. Ilachinski, *Cellular Automata, a Discrete Universe*, 1st ed. World Scientific Publishing Co. Pte. Ltd., 2001.

[9] W. Spataro, M. Avolio, V. Lupiano, G. Trunfio, R. Rongo, and D. D'Ambrosio, "The latest release of the lava flows simulation model sciara: first application to mt etna (italy) and solution of the anisotropic flow direction problem on an ideal surface," *Procedia Computer Science*, vol. 1, pp. 17–26, 2010.

[10] W. Deen, *Analysis of Transport Phenomena*, 1st ed. Oxford University Press, 1998.

[11] D. Feitelson, "Experimental computer science: The need for a cultural change," School of Computer Science and Engineering The Hebrew University of Jerusalem, Whitepaper, 2006.

[12] M. Wolfe. (2012, December) Understanding the cuda data parallel threading model a primer. The Portland Group. [Online]. Available: https://www.pgroup.com/lit/articles/insider/v2n1a5.htm

[13] *CUDA C Programming Guide*, 6th ed. NVIDIA, Febrero 2014.

[14] D. MacKenzie, *Manual page time(1)*, Debian GNU/Linux.

[15] J. Loeppky, J. Sacks, and W. Welch, “Choosing the sample size of a computer experiment: A practical guide,” National Institute of Statistical Sciences, 19 T. W. Alexander Drive PO Box 14006 Research Triangle Park, NC 27709-4006, Tech. Rep. 170, 2 2008.

[16] S. A. Gélvez. Repositorio de documentos del proyecto de grado titulado: Problemas computacionales asociados a la construcción de modelos de simulación basados en autómatas celulares en paralelo. caso de estudio: evaluación de amenazas asociadas a flujos de lava volcánica como fluído bingham. http://forge.sc3.uis.edu.co/redmine/projects/scalaf/documents.

BIBLIOGRAFÍA

Abarbanel, H., et al. “Cellular Automata and Parallel Processing for Practical Fluid-Dynamics Problems”. Tech. rep., Department of Energy and DARPA, 1990.

Abidine, Zine El, Steeve Ebener, John Boos, Eman Abdel Ghaffar, and Altaf Musani. “Modelling the spatial distribution of five natural hazards in the context of the WHO/EMRO Atlas of Disaster Risk as a step towards the reduction of the health impact related to disasters”. International Journal of Health Geographics 6:8 (Marzo 2007): 28.

Andrade, Hugo Hernando, Isaac Dyner, Angela Espinosa, Hernán López Garay, and Ricardo Sotaquirá. Pensamiento Sistémico: Diversidad en Busca de Unidad. 1. División Editorial y de Publicaciones, Universidad Industrial de Santander, 2001.

Avolio, Maria, et al. “Simulation of the 1992 Tessina landslide by a cellular automata model and future hazard scenarios”. JAG 2, no. 1 (2000): 41-50.

Avolio, Maria, Gino Crisci, Salvatore Di Gregorio, Rocco Rongo, William Spataro, and Donato D’Ambrosio. “Pyroclastic flows modelling using cellular automata”. Computers & Geosciences 32 (2006): 897-911.

Bänninger, D. “Technical Note: Water flow routing on irregular meshes”. 11 (2007): 1243-1247.

Barrios, Carlos Jaime. “Parallel Programming Evironments and Parallel Programming Computation”. 2010.

Bell, Fred. Geological Hazards: Their Assessment, Avoidance and Mitigation. Spon Press, Taylor and Francis Group, 1999.

Cannataro, M., S. Gregorio, R. Rongo, W. Spataro, G. Spezzano, and D. Talia. “A parallel cellular automata environment on multicomputers for computational science”. Parallel Computing 21 (1995): 803-823.

Childs, Hank, et al. “VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data”. In High Performance Visualization–Enabling Extreme-Scale Scientific Insight, 357-372. 2012.

Consortium, Gisela. “GISELA”. GISELA. 2010.

Crisci, G.M., S. Di Gregorio, R. Rongo, M. Scarpelli, W. Spartaro, and S. Calvari. “Revisiting the 1669 Etnean eruptive crisis using a cellular automata model and implications for volcanic hazard in the Catania area”. Journal of Volcanology and Geothermal Research 123 (2003): 211-230. CUDA C Programming Guide. 6. NVIDIA, 2014.

D’Ambrosia, D., W. Spataro, and G. Iovine. “Parallel genetic algorithms for optimising cellular automata models of natural complex phenomena: An application to debris flows”.

Computers & Geosciences 32 (2006): 861-875.

Damiani, M.L., G. Groppelli, G. Norini, E. Bertino, A. Gigliuto, and A. Nucita. “A lava flow simulation model for the development of volcanic hazard maps for Mount Etna (Italy)”. Computers & Geosciences 32 (2006): 512-526.

Deen, William. Analysis of Transport Phenomena. 1. Oxford University Press, 1998.

Dragoni, Michele, Maurizio Bonafede, and Enzo Boschi. “Downslope Flow Models of a Bingham Liquid: Implications for Lava Flows”. Journal of Volcanology and Geothermal Research 30 (n.d.): 21.

Feitelson, Dror. “Experimental Computer Science: The Need for a Cultural Change”. Whitepaper, School of Computer Science and Engineering The Hebrew University of Jerusalem, 2006.

Foster, Ian. “Designing and Building Parallel Programs”. Designing and Building Parallel Programs. n.d.

Francis, Peter. Volcanoes: A Planetary Perspective. 1. Oxford University Press, 1993.

Gélvez, Sergio Augusto. “Problemas computacionales asociados a la construcción de modelos de simulación basados en autómatas celulares en paralelo. Caso de estudio: Evaluación de amenazas asociadas a flujos de lava volcánica como fluido bingham”. unpublished, n.d.

Gélvez, Sergio Augusto. “Repositorio de documentos del proyecto de grado titulado: Problemas computacionales asociados a la construcción de modelos de simulación basados en autómatas celulares en paralelo. Caso de estudio: evaluación de amenazas asociadas a flujos de lava volcánica como fluído bingham”. Repositorio de documentos del proyecto de grado titulado: Problemas computacionales asociados a la construcción de modelos de simulación basados en autómatas celulares en paralelo. Caso de estudio: evaluación de amenazas asociadas a flujos de lava volcánica como fluido bingham. http://forge.sc3.uis.edu.co/redmine/projects/scalaf/documents, n.d.

Grewe, Dominik, Zheng Wang, and Michael F. P. “A workload-aware mapping approach for data-parallel programs”. New York, NY, USA: ACM, 2011. 117-126.

Hao, Zhu, Sun Yan, Rajagopal Gunaretnam, Mondry Adrian, and Dhar Pawan. “Facilitating arrhythmia simulation: the method of quantitative cellular automata modeling and parallel running”. BioMedical Engineering OnLine 3 (2004): 29.

Herault, Alexis, Annamaria Vicari, Alessia Ciraudo, and Ciro del Negro. “Forecasting lava flow hazards during the 2006 Etna eruption: Using the MAGFLOW cellular automata model”. Computers & Geosciences 35 (2009): 1050-1060.

Hinch, John. “Stress Relaxation”. Woods Hole Oceanographic Institution. Woods Hole Oceanographic Institution, 2003. 18.

Hoefler, Torsten, and Marc Snir. “Generic topology mapping strategies for large-scale parallel architectures”. New York, NY, USA: ACM, 2011. 75-84.

Ilachinski, Andrew. Cellular Automata, a Discrete Universe. First. World Scientific Publishing Co. Pte. Ltd., 2001.

Iovine, G., S. Di Gregorio, and V. Lupiano. “Debris-flow susceptibility assessment through cellular automata modeling: an example from 15-16 December 1999 disaster at Cervinara and San Martino Valle Caudina (Campania, southern Italy)”. Natural Hazards and Earth System Sciences 3 (2003): 457-468.

Ivanov, Lubomir. “A modern course on parallel and distributed processing”. J. Comput. Small Coll. (Consortium for Computing Sciences in Colleges) 21 (June 2006): 29-38.

Loeppky, Jason, Jerome Sacks, and William Welch. “Choosing the Sample Size of a Computer Experiment: A Practical Guide”. Tech. rep., National Institute of Statistical Sciences, 19 T. W. Alexander Drive PO Box 14006 Research Triangle Park, NC 27709-4006, 2008.

Luebke, David, et al. “GPGPU: general purpose computation on graphics hardware”. New York, NY, USA: ACM, 2004.

MacKenzie, David. “Manual page time(1)”. n.d.

Mariño, Nestor Díaz. “Metodología y framework computacional para el diseño inverso de modelos de autómata celular de secuencias cortas de aminoácidos soportado en un proceso de minería de datos”. unpublished, 2011.

Mariño, Nestor Díaz. “Metodología y framework computacional para el diseño inverso de modelos de autómata celular de secuencias cortas de aminoácidos soportado en un proceso de minería de datos”. Propuesta de Tesis de Maestría, Universidad del Valle, Escuela de Ingeniería de Sistemas y Computación, 2008.

Marc Mazzariol. “Dynamic load balancing of parallel cellular automata”. Proceedings SPIE Conference, Parallel and Distributed Methods for Image Processing IV 4118 (Julio 2000): 21-29.

Miyamoto, Hideaki, and Sho Sasaki. “Simulating lava flows by an improved cellular automata method”. Computers & Geosciences 23 (1997): 283-292

Morell-Gimenez, Vicente, Antonio Jimeno-Morenilla, and José Garcia-Rodriguez. “Efficient tool path computation using multi-core GPUs”. Computers in Industry -, no. 0 (2012): - .

Nichele, Stefano. “Discrete dynamics of cellular machines: specification and interpretation”. New York, NY, USA: ACM, 2011. 767-770.

Nielsen, A. S. “Feasibility study of the parareal algorithm”. Master’s thesis, Technical University of Denmark, DTU Informatics, E-mail: reception@imm.dtu.dk, Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2012.

OpenMP, ARB. “OpenMP application program interface, v. 3.0”. no. July (2008).

PIEGARI, Ester, Rosa DI MAIO, Roberto SCANDONE, and Leopoldo MILANO. “A cellular automaton model for magma ascent: Degassing and styles of volcanic eruptions”. Journal of Volcanology and Geothermal Research 202 (Febrero 2011): 22-28.

Profiler User's Guide. NVIDIA Corporation, 2015.

Seredynski, Franciszek, and Albert Zomaya. "Sequential and Parallel Cellular Automata-Based Scheduling Algorithms". IEEE Transactions on Parallel and Distributed Systems 13 (October 2002): 1009-1023.

Smnce, Elsewer, and Greal Britain Soo. "i- 1997". 23, no. 3 (1997): 283-292.

Spataro, William, Maria Avolio, Valeria Lupiano, Giuseppe Trunfio, Rocco Rongo, and Donato D'Ambrosio. "The latest release of the lava flows simulation model SCIARA: first application to Mt Etna (Italy) and solution of the anisotropic flow direction problem on an ideal surface". Procedia Computer Science 1 (2010): 17-26.

Spezzano, G. "The CAMELot cellular automata programming environment". The CAMELot cellular automata programming environment. n.d.

Spezzano, Giandomenico, and Domenico TALIA. "Programming cellular automata algorithms on parallel computers". Future Generation Computer Systems 16 (1999): 203-216.

Talia, Domenico. "Cellular Automata + Parallel Computing = Computational Simulation". Wissenschaft & Technik Verlag. 1997. 409-414.

Talia, Domenico. "Implementing standard and nonstandard parallel cellular automata in CARPET". 2001. 243-249.

Talia, Domenico. "Parallel Cellular Algorithms and Programs". Chap. 6, edited by S. Olariu and A. Zomaya, 85-101. Chapman & Hall/CRC, 2005.

Tosic, Predrag. "A perspective on the future of massively parallel computing: fine-grain vs. coarse-grain parallel models comparison & contrast". New York, NY, USA: ACM, 2004. 488-502.

"Towards on Grid Computing Applications to Science Gateway's Platforms". CLCAR. 2012.

Valdés, Angel María. "Prototipo HOMOS 2.0: Herramienta software para el modelado y simulación basado en objetos y reglas". unpublished, 2011.

Vicari, Annamaria, Herault Alexis, Ciro Del Negro, Mauro Coltelli, Maria Marsella, and Cristina Proietti. "Modeling of the 2001 lava flow at Etna volcano by a Cellular Automata approach". Environmental Modelling & Software 22 (2007): 1465-1471.

Wolfe, Michael. "Understanding the CUDA Data Parallel Threading Model A Primer". Understanding the CUDA Data Parallel Threading Model A Primer. December 2012.