Artículo de Revisión

Creación de un modelo de ejecución generalizado para sistemas computacionales paralelos de datos anidados

Building a generalized execution model for nested data parallel computing systems

Yerovi Ricaurte, Elke; Facuy Delgado, Jussen;
Molina Oleas, Wilson; Ortega Ponce, Laura

Elke Yerovi Ricaurte
eyerovi@uagraria.edu.ec
Universidad Agraria del Ecuador

Jussen Facuy Delgado
jfacuy@uagraria.edu.ec
Universidad Agraria del Ecuador

Wilson Molina Oleas
wmolina@uagraria.edu.ec
Universidad Agraria del Ecuador

Laura Ortega Ponce
lortega@uagraria.edu.ec
Universidad Agraria del Ecuador

Pro Sciences: Revista de Producción, Ciencias e Investigación
CIDEPRO, Ecuador
e-ISSN: 2588-1000
Periodicidad: Trimestral
Vol. 6, No. 45, 2022
editor@journalprosciences.com

Recepción: 30 Junio 2022
Aprobación: 4 Septiembre 2022

DOI: https://doi.org/10.29018/issn.2588-1000vol6iss45.2022pp233-249

Resumen: El presente trabajo analiza los muchos intentos de simplificar la tarea de programar las arquitecturas multiprocesador paralelas actuales, el más exitoso es el paradigma del paralelismo de datos. Gran parte del aspecto del modelo radica en su vista de alto nivel de la máquina paralela junto con su mapeo eficiente a una gran clase de arquitecturas del mundo real. Se realizará un análisis exhaustivo de la literatura científica relacionada con las variables de estudio de la investigación, en especial de los artículos referentes a la experiencia de crear modelos de ejecución generalizado para sistemas computacionales paralelos de datos anidados. Si bien el modelo de ejecución generalizado para sistemas computacionales paralelos de datos anidados ha sido muy efectivo al permitir la especificación altamente paralela de operaciones en arreglos rectangulares, su aplicabilidad a programas que usan estructuras de datos menos regulares es muy limitada.Un programa de datos paralelos está compuesto por una secuencia de operaciones orientadas a la recopilación de alto nivel a través de las cuales se ejecuta una única ruta conceptual de control. Históricamente, el éxito del modelo de datos en paralelos de computación puede atribuirse en gran medida al desarrollo de arquitecturas informáticas que incorporan directamente modelos de ejecución, los paradigmas SIMD y SPMD, útiles para soportar un modelo de datos en paralelos. Se han propuesto varios lenguajes de programación que se centran en construcciones de datos en paralelo como la fuente principal de paralelismo en los programas.

Palabras clave: multiprocesador, modelo paralelo, arquitectura, programas.

Abstract: This paper analyzes the many attempts to simplify the task of programming current parallel multiprocessor architectures, the most successful being the data parallelism paradigm. Much of the model's appeal lies in its high-level view of the parallel machine along with its efficient mapping to a large class of real-world architectures. An exhaustive analysis of the scientific literature related to the study variables of the research will be carried out, especially the articles referring to the experience of creating generalized execution models for parallel computational systems of nested data. Although the generalized execution model for nested data parallel

Cómo citar: Yerovi Ricaurte, E., Facuy Delgado, J., Molina Oleas, W., & Ortega Ponce, L. (2022). Creación de un modelo de ejecución generalizado para sistemas computacionales paralelos de datos anidados. Pro Sciences: Revista De Producción, Ciencias E Investigación, 6(45), 233-249. https://doi.org/10.29018/issn.2588-1000vol6iss45.2022pp233-249

computational systems has been very effective in allowing highly parallel specification of operations on rectangular arrays, its applicability to programs using less regular data structures is very limited. A parallel data program is composed of a sequence of high-level collection-oriented operations through which a single conceptual path of control is executed. Historically, the success of the parallel data model of computing can be largely attributed to the development of computing architectures that directly incorporate execution models, the SIMD and SPMD paradigms, useful for supporting a parallel data model. Various programming languages have been proposed that focus on parallel data constructions as the main source of parallelism in programs.

Keywords: multiprocessor, parallel model, architecture, programs.

Introducción

El propósito de esta investigación es explorar extensiones al crear un modelo paralelo de datos que lo hacen más adecuado para este tipo de problemas irregulares. En particular, se considera el paradigma del paralelismo de datos anidados propuesto por Acar (2020), que analizan técnicas para mapear de manera eficiente programas con dicho paralelismo anidado en arquitecturas multiprocesador tradicionales. A través del análisis matemático, derivando una implementación novedosa de tales características que hace uso de un modelo de computación multiproceso en cada procesador de una máquina multiprocesador. Para demostrar la validez de este enfoque, evalúan el rendimiento de varios programas compilados utilizando este sistema de lenguaje, comparando las cifras obtenidas con las de programas equivalentes compilados en los sistemas CM Fortran y NBsl.

La programación eficiente de sistemas computacionales paralelos de datos anidados, aquellas que ofrecen decenas, cientos o miles de procesadores conectados por una red de alta velocidad, es quizás uno de los problemas más difíciles que enfrenta la informática científica moderna. Es ampliamente reconocido que tales máquinas tienen un enorme potencial para proporcionar velocidades de ejecución radicalmente mejoradas para muchos programas científicos, lo que permite que los problemas más grandes sean computacionalmente factibles. Sin embargo, con demasiada frecuencia este potencial no se realiza, y los programadores que transfieren sus códigos a arquitecturas paralelas observan solo una mejora moderada en el rendimiento. (Junior, 2020)

Con mucho, el factor más importante que contribuye a este fenómeno es la complejidad inherente a los modelos de programación ofrecidos para programar máquinas paralelas. Los primeros intentos de construir tales modelos involucraron un simple injerto de características paralelas en el modelo secuencial tradicional de ejecución, tal como se expresa en lenguajes como C y Fortran. Dichos sistemas, muchos de los cuales todavía se recomiendan, tienen la ventaja de que son fáciles de aprender para un programador familiarizado con la programación secuencial. Sin embargo, como estos programadores pronto descubren, los modelos hacen poco para enmascarar la complejidad inherente que implica la programación de muchos procesadores individuales, cada uno de los cuales puede ejecutar instrucciones de manera totalmente independiente y que, en cualquier momento, pueden interactuar de manera arbitraria. (Jonasson, 2020)

En general, esta complejidad muchas veces mayor que la de programar una máquina uni procesadora debe ser manejada por el programador. Además de esto, el programador también tiene la tarea de hacer que su implementación del algoritmo deseado funcione bien al garantizar que todos los procesadores del sistema permanezcan ocupados durante toda la ejecución y que los gastos generales por ejemplo, debido a la comunicación entre nodos no sean eficientes. Considerando el esfuerzo intelectual masivo requerido para construir un programa bajo estos modelos que sea correcto en todos los casos posibles y también eficiente, no es sorprendente que existan pocos programas de este tipo. (Efros, 2020)

El modelo de programación en paralelo de datos representa una alternativa más simple a la tarea intrínsecamente compleja de programar bajo tales modelos. Su simplicidad se deriva de la provisión de una vista de nivel superior de la programación paralela que abstrae muchos de los detalles de la arquitectura en paralelo. El principio fundamental que subyace a este paradigma es que se puede lograr un alto grado de paralelismo potencialmente al dividir los grandes agregados de datos de un programa en muchos nodos de una máquina paralela, y luego enmarcar un cómputo en términos de operaciones de agregado total mediante los cálculos de elementos relativamente independientes y, por lo tanto, pueden realizarse en paralelo. (Liu, 2020)

Un programa de datos paralelos está compuesto por una secuencia de operaciones orientadas a la recopilación de alto nivel a través de las cuales se ejecuta una única ruta conceptual de control. El esfuerzo intelectual requerido para construir dicho programa es considerablemente menor que el necesario para coordinar las múltiples rutas de control concurrentes del paradigma derivado en serie. Por lo tanto, el modelo de datos paralelos es más fácil de programar.

Sin embargo, la abstracción de nivel superior que ofrece el paradigma significa que el rendimiento final de un modelo de ejecución generalizado para sistemas computacionales paralelos de datos anidados está determinado por la forma en que el compilador lo asigna al hardware paralelo. Por lo general, las operaciones de alto nivel que ofrece un lenguaje de modelo paralelo de datos anidados se eligen de manera que sean muy regulares tanto en sus patrones de cálculo como en su paralelismo. Esto simplifica la tarea de compilación, ya que tales formas tienen asignaciones claras, y al mismo tiempo eficientes, en arquitecturas paralelas.

La tecnología del compilador requerida para generar tales mapeos se entiende bien y se ha incorporado en una serie de implementaciones de modelos de ejecución generalizado para sistemas computacionales paralelos de datos anidados muy eficientes (Duncan, 2020). Estas implementaciones adoptan una estratégica paralelización que busca emitir el programa en una forma susceptible de ejecución bajo cualquiera de los programas múltiples de datos. El enfoque gira en torno a la partición de los agregados de datos dentro del programa a través de una serie de memorias disjuntas, cada una de las cuales está estrechamente asociada con un procesador.

El programa se compila en una secuencia de pasos secundarios, cada uno correspondiente a una operación en la fuente. Los diversos procesadores de la máquina paralela avanzan a través de esta secuencia en paso de bloque, sincronizándose globalmente al final de cada sub paso para eliminar cualquier posibilidad de errores en la ejecución. Donde cada operación se implementa como una operación cooperativa entre procesadores, en la que cada procesador se encarga de ejecutar las operaciones en serie por elemento para esos índices agregados almacenados dentro de su memoria asociada.

Implementaciones de hardware en la ejecución de datos en paralelos

Históricamente, el éxito del modelo de datos en paralelos de computación puede atribuirse en gran medida al desarrollo de arquitecturas informáticas que incorporan directamente modelos de ejecución, los paradigmas SIMD y SPMD, útiles para soportar un modelo de datos en paralelos. Cuando se requirieron implementaciones dirigidas a hardware paralelo menos especializado (por ejemplo, multiprocesadores de datos múltiples de instrucción múltiple para sintetizar o simular dichos modos de ejecución, los modelos de datos en paralelos que producen código para máquinas SIMD o SPMD especializadas podrían aprovechar las instalaciones optimizadas por hardware especialmente atendiendo a los estilos de interacción y sincronización que se encuentran dentro de esos modelos. Este nivel de soporte directo demostró ser un factor habilitador en el surgimiento de modelos de datos en paralelos como tecnología para la programación de alto rendimiento. (Kumbhar, 2019)

Lenguajes paralelos de datos

Se han propuesto varios lenguajes de programación que se centran en construcciones de datos en paralelo como la fuente principal de paralelismo en los programas. Dicho paralelismo se obtiene mediante la especificación de operaciones de agregado completo entre agregados que pueden distribuirse entre nodos de una máquina paralela. La mayoría de los lenguajes utilizados para manejar datos en paralelo ofrecen características similares para la introducción de este tipo de agregados y operaciones; la mayoría se pueden caracterizar por proporcionar las siguientes instalaciones comunes:

- Los agregados como objetos de datos elementales: la capacidad de pasar agregados completos como argumentos de función, y la capacidad de referirse sintácticamente a un agregado completo. Proporcionando el nombre del agregado sin ninguna deferencia de índice dentro de una expresión.
- Sub-selección masiva de objetos agregados: la capacidad de obtener los valores para una dimensión agregada completa como una sola operación, y a menudo la capacidad de poder refinar aún más esta selección especificando un sub-rango de índices.
- Expresiones agregadas y conformidad: la capacidad de especificar una operación en todos los índices de un agregado simplemente aplicando la operación al nombre de agregado no indexado, y también la capacidad de aplicar operaciones binarias simples a pares de agregados conformes. La semántica de estas últimas operaciones se define mediante índices agregados por pares.
- Coerción de los agregados para obtener conformidad: la capacidad de poder aumentar el rango de un agregado copiando valores del agregado existente a lo largo de una nueva dimensión o mediante la disminución del rango de un agregado colapsando una dimensión agregada combinando valores usando alguna función.
- Indización de expresiones: la capacidad de aplicar todos los estilos normales de indexación agregada a expresiones que representan valores agregados.
- Asignación agregada: la capacidad de poder asignar una expresión con valor agregado a un agregado conforme específico. La semántica de tal asignación es de elementos sabios; cada elemento en el agregado de destino se sobrescribe con el valor del elemento correspondiente en la expresión de valor agregado.
- Operaciones paralelas a través de agregados: un conjunto de operaciones de datos en paralelo estándar que permiten a un programador introducir paralelismo explícito a través de expresiones orientadas a la recopilación; Estos incluyen reducciones comunes de agregados, prefijos paralelos y operaciones de permutación. (Carratalá Sáez, 2016)

Estas características del lenguaje proporcionan colectivamente un entorno en el que se pueden realizar especificaciones explícitamente paralelas. La manifestación precisa de estos conceptos en los lenguajes de paralelismo de datos, incluida la elección de los tipos de agregados para permitir la expresión paralela, se rige en gran medida por el desarrollo histórico de esta clase de lenguajes y las formas computacionales para las que se diseñaron originalmente

Desarrollo histórico de lenguajes de programación de datos en paralelo

Se puede considerar que el desarrollo de lenguajes de programación de datos en paralelo ha sido impulsado por dos fuerzas distintas: la necesidad de desarrollar un medio para programar máquinas SIMD recién construidas y el deseo de construir un marco conceptual simple para expresar operaciones paralelas. La primera de estas influencias históricamente dominó la evolución temprana del paradigma y ha sido en gran parte responsable de forjar las amplias características de los lenguajes de programación de datos en paralelo que hoy son generalizados y populares. El deseo de una ejecución paralela simplificada ha sido responsable tanto del desarrollo de varios aumentos basados en la investigación del modelo de lenguajes de programación de datos en paralelo básico como, en última instancia, del refinamiento y la evolución moderna del modelo a través de la absorción de ideas de tales sistemas de investigación.

Orígenes conceptuales

Los conceptos básicos que subyacen a los datos en paralelo pueden identificarse en los primeros lenguajes orientados a la anidación de estos datos como APL y SETL. Dentro de estos lenguajes, se proporcionaron características para la expresión de operaciones de agregado total como operaciones de nivel de fuente único. Conceptualmente, un agregado era un ciudadano de primera clase en estos idiomas en lugar de simplemente un conjunto de elementos de datos y se proporcionaron características para aplicar operaciones directamente a un agregado. La semántica de dicha operación orientada a la colección se definió típicamente para implicar que se realizaría un cálculo idéntico para cada índice de la colección. También estuvieron presentes las características del lenguaje para reducir la dimensión de un agregado. Todas estas características recuerdan el estilo de programación de datos en paralelo, a pesar del hecho de que estos lenguajes fueron diseñados típicamente no por sus oportunidades de paralelismo, sino por su poder expresivo y abstracto.

Idiomas para programar máquinas SIMD tempranas

Cuando las arquitecturas SIMD comenzaron a desarrollarse en la década de 1970 como resultado de la investigación de formas rentables de construir computadoras paralelas, el desarrollo de un paradigma apropiado para programar tales máquinas se convirtió en una prioridad importante. Los primeros sistemas prototipo se codificaron directamente en lenguaje ensamblador; sin embargo, con la posibilidad de que tales máquinas pudieran convertirse en productos comerciales surgió la necesidad de desarrollar lenguajes y compiladores que pudieran explotar efectivamente el potencial de las arquitecturas. Al desarrollar estos sistemas, los proveedores se enfocan con mayor frecuencia (Globa, 2008)

Fortran como base para el lenguaje que buscaban desarrollar. Esta elección fue impulsada en gran medida por la preferencia de idioma y la familiaridad de su mercado previsto dirigido a programadores científicos. Las extensiones de los datos en paralelo del lenguaje base se derivaron de manera muy simple de una evaluación de los estilos de ejecución para los cuales se optimizó la máquina SIMD subyacente. Tales máquinas estaban claramente pensadas como procesadores de matriz rápida, por lo que se introdujeron construcciones paralelas a través de las matrices. El hecho de que siempre se requiriera que los nodos de la máquina ejecutaran la misma secuencia de

instrucciones le dio a su ejecución paralela una eficiencia regular, muy adecuado para problemas que requerían que todos los elementos de la matriz actuaran de manera idéntica. Es decir, de en forma de datos en paralelo. Por lo tanto, se agregaron características al lenguaje para permitir la expresión conveniente de este tipo de programas.

Un primer ejemplo de un lenguaje utilizado para anidar los datos en paralelo suministrado por el proveedor dirigido a una arquitectura SIMD específica es DAP Fortran un dialecto construido para el ICL DAP. El lenguaje explotó el paralelismo de las operaciones a través de matrices cuyos valores se dividieron en la máquina. Una limitación severa impuesta a tales agregados particionados fue que solo podían ser de un tamaño menor o igual al número de nodos en la máquina. Es decir, la matriz más grande que podría considerarse en las primeras máquinas DAP era 32 x 32; en máquinas posteriores se podría actuar sobre matrices de hasta 64 x 64. Esta imposición fue claramente un producto de la forma en que el cálculo se mapeó en la máquina SIMD: cada elemento de procesamiento era responsable de actuar sobre un único elemento agregado. Las operaciones de datos en paralelo en el lenguaje admitieron los siguientes tipos de operaciones en agregados particionados:

- Difundir un valor para ser almacenado en cada elemento agregado o sub agregado,
- Realizar asignaciones paralelas entre agregados de forma conformable,
- Calcular una sección de matriz o segmento,
- Expresar el cálculo de formas desplazadas de un agregado
- Realizar una operación escalar dada en todos los índices en paralelo
- Combinar agregados o dimensiones de un agregado multidimensional en paralelo de acuerdo con alguna función. (Giri, 2020)

Las características de los datos en paralelo del lenguaje se introdujeron a través de una notación de indexación especial referida, mediante una notación de elisión de subíndice, a la primera columna de la matriz A o mediante un conjunto de agregado incorporado funciones de nivel. Para permitir que se especifiquen operaciones más complejas, el lenguaje permitió que muchos de estos estilos de operación de datos en paralelo se combinaron con condicionales que definían el subconjunto de los procesadores que ejecutaría la operación. Por ejemplo, fue posible especificar dentro de una asignación de nivel agregado que solo aquellos elementos de la matriz que actualmente tienen valores inferiores a 0 deberían recibir un nuevo valor. En este caso, esto definiría una operación de datos en paralelo en la que solo participaría un subconjunto de la máquina.

Lenguajes específicos SIMD posteriores

A medida que se desarrollaron las instalaciones ofrecidas por las arquitecturas SIMD, los compiladores Fortran suministrados por el proveedor evolucionaron para agregar características que explotaran la nueva funcionalidad soportada por hardware. Las primeras mejoras sobre el DAP Fortran eliminaron la restricción sobre el tamaño del agregado distribuido sobre el que se podía actuar. Estos lenguajes aprovecharon el hardware especial presente en máquinas como el CM-2 y MasPar que implementaron un modelo de procesadores virtuales. Conceptualmente, en dichos lenguajes, una operación de datos en paralelo a través de una cuadrícula utiliza tantos procesadores virtuales como elementos en conjunto, cada uno de estos procesadores es responsable de los cálculos a través de ese elemento. El hardware dentro de la máquina mapea los procesadores virtuales en procesadores reales, permitiendo que cada procesador físico tome el papel de muchos procesadores virtuales. Este avance en los lenguajes de paralelismo de datos introdujo un grado de independencia arquitectónica: ya no se definía explícitamente el tamaño y la configuración de la máquina dentro del programa. Además, permitió que el código se transportara fácilmente entre máquinas de diferente tamaño físico. (Jonasson, 2020)

El lenguaje CM Fortran también introdujo una forma alternativa de notación de datos en paralelo basada en un bucle completo generalizado a través de los índices de una matriz. La expresividad de este concepto amplía la gama de problemas que pueden expresarse como programas de datos en paralelo; El hecho de que el lenguaje permita el anidamiento de bucles completos ofrece posibilidades interesantes para soportar códigos computacionalmente complejos o irregulares.

Otra facilidad que apareció primero en esta generación de lenguajes de paralelismo de datos como la capacidad de especificar operaciones de tipo permutación en vectores y matrices. Por lo general, aparecían en el lenguaje como funciones intrínsecas como las operaciones paralelas de SEND en CM Fortran o como operaciones especiales de indexación en las que aparece un vector entero en lugar de un escalar como índice. Las instalaciones también están comúnmente disponibles en este idioma para especificar permutaciones generalizadas en las que los valores podrían colisionar y combinarse por una función en su destino. Parece claro que este tipo de características ingresaron a los lenguajes de paralelismo de datos principalmente debido a la presencia de hardware de enrutamiento de red generalizado en las máquinas a las que fueron dirigidos.

Lenguajes de paralelismo de datos derivados de C y Lisp

Simultáneamente con estos avances en los idiomas suministrados por el proveedor, se propusieron varios idiomas de datos en paralelo que desarrollaron paradigmas en otras direcciones. Se hicieron una serie de definiciones e implementaciones para extensiones de datos en paralelo a idiomas distintos de Fortran, pero que proporcionaron estilos similares de operación basada en arreglos. El lenguaje C * fue un ejemplo temprano de tal extensión, que buscaba extender ANSI C al agregar una clase especial de operaciones paralelas especificadas usando un estilo especial de indexación a través de matrices distribuidas implícitamente de un programa. (Reiser, 2020)

Estandarización de los modelos de programación paralela mediante Fortran

Con la proliferación de muchos dialectos diferentes de modelos de programación paralela mediante Fortran, todos basados en nociones y abstracciones similares, surgió el deseo de estandarizar. El resultado final de una discusión considerable por parte del grupo ANSI X3J3 fue el lenguaje Fortran 90 (Naterop, 2020). Este estándar incorpora las características de los modelos de programación paralela introducidas con DAP Fortran y agrega la independencia de máquina de lenguajes como CM Fortran. Al igual que con la mayoría de los dialectos de los modelos de programación paralela Fortran, gran parte de la especificación de paralelismo está representada dentro de las operaciones de indexación de matriz. También se proporcionan intrínsecos paralelos especiales como multiplicación de matrices, productos de puntos, suma de todos los elementos de la matriz, desplazamientos circulares y finales. Notablemente, el estándar no incluye explícitamente operaciones de prefijos paralelos.

Direcciones recientes en de los modelos de programación paralela Fortran

Después de la codificación del estándar Fortran 2020, han surgido una serie de nuevos dialectos de modelos de programación paralela. De éstos, el más conocido es el de alto rendimiento Fortran (HPF), una extensión de la norma que trata de ofrecer al programador control directo sobre una serie de cuestiones relativas a la asignación del programa a una subyacente máquina. La ruta de implementación común para lenguajes utilizados en los modelos de programación paralela se centra en la descomposición de estructuras de matriz a través de las diversas memorias de la máquina subyacente. En HPF, hay disponibles funciones de nivel de origen que pueden usarse para guiar al compilador a elegir la descomposición de datos adecuada. Tales características son una reacción a las dificultades encontradas por los compiladores-escritores al producir un sistema que puede tomar

automáticamente decisiones de diseño de datos de alta calidad por medios analíticos. En HPF, un programador puede especificar una *descomposición* que es una entidad abstracta que se parece a una matriz pero que no implica almacenamiento. (Kumbhare, 2020)

Dichas declaraciones son útiles ya que las matrices dentro del programa pueden especificarse como alineadas con una descomposición particular. Si dos o más vectores o matrices se alinean con la misma descomposición, el compilador puede deducir que esos agregados de datos deben descomponerse de manera idéntica.

La sintaxis del lenguaje permite que se especifiquen otros tipos de información de descomposición a través de sentencias ALIGN, incluida la capacidad de alinear un agregado unidimensional con una de las dimensiones de una matriz multidimensional, y la facilidad de denotar que una matriz debería ser descompuesto de una manera que lo alinee con la transposición de otro. Si bien la adición de dicha característica al estándar Fortran 2020 puede verse como un ligero compromiso con la abstracción de alto nivel de ese lenguaje (es decir, el principio de codificación sin tener en cuenta la máquina subyacente), la asistencia que dichas directivas brindan en el proceso de compilación es importante para mejorar el rendimiento del programa.

Direcciones de investigación para modelos de programación paralela

Fortran 2020 de Intel representan lo último en lenguajes para modelos de programación paralela que se utilizan actualmente para la computación científica. La evolución de las ideas introducidas hace más de tres décadas ha dado como resultado lenguajes maduros en poder expresivo y fácilmente mapeables en una amplia gama de hardware paralelo. Sin embargo el dominio de los problemas para los cuales dichos lenguajes pueden proporcionar soluciones eficientes está limitado por la naturaleza del modelo de programación paralela que los subyace. Dado que el modelo se fundó sobre la suposición de un paralelismo regular, su expresividad y eficiencia no se extienden a casos en los que aparecen formas paralelas irregulares dentro de un programa. Ejemplos de este tipo de irregularidades surgen naturalmente en una serie de problemas científicos importantes. Para permitir la expresión de formas altamente paralelas para problemas tan significativos, se han propuesto varios idiomas en los que los conceptos generales de los modelos de programación paralela están presentes en una forma generalizada.

Límites del paralelismo de datos tradicional

El modelo de programación paralela y su implementación tradicional bajo los modelos SIMD y SPMD proporcionan una plataforma probada de alto rendimiento para la especificación de algoritmos paralelos en grandes conjuntos de datos rectangulares. Se proporcionan operaciones de programación paralela que realizan un cálculo en serie por elemento para cada índice de dicha matriz en paralelo. Otros consideran operaciones paralelas a través de sub-selecciones de estas estructuras de datos regulares, o reducciones paralelas de valores de matriz. En resumen, el paradigma ofrece una amplia gama de herramientas paralelas para modelar algoritmos que utilizan exclusivamente el rectangular siempre.

Si bien muchos cálculos científicos pueden estar razonablemente enmarcados de tal forma, existen dominios enteros de modelado científico, aquellos que consideran datos estructurados irregularmente, que no son susceptibles de expresión bajo los modelos de programación paralela. Ejemplos de tales áreas incluyen:

- Cálculo de matriz dispersa donde las cuadrículas se representan comúnmente como matrices irregulares,
- Simulaciones de química computacional donde las moléculas están representadas por una estructuración de agregados simples,
- Cálculos de elementos finitos en mallas irregulares.

Para estos problemas irregulares de computación científica, los modelos de programación paralela ofrecen solo pequeñas oportunidades para la expresión paralela de operaciones. Este es un artefacto de la naturaleza de las operaciones paralelas disponibles bajo el modelo: todos se ocupan exclusivamente de aplicar un cálculo en serie a través de los índices de una estructura de datos regular. Si se utilizan operadores programación paralela para expresar una operación a través de la matriz dispersa la multiplicación de cada elemento de la matriz por 2 debe enmarcar una paralelización como una operación de programación paralela única a través del vector externo horizontal, o una serie de operaciones de programación paralela a través de los vectores internos verticales

Paralelismo de datos anidados

Los modelos de programación paralela a pesar de sus éxitos al proporcionar una base de alto rendimiento para cálculos científicos simples y regulares al aplicar el paradigma tradicional de la ejecución paralela de datos no se aplica fácilmente a la paralelización de cálculos menos regulares. Específicamente, el modelo plano presentado por estos paradigmas no encaja bien con los agregados de datos estructurados (potencialmente de tamaño irregular o de alteración dinámica encontrados dentro de algoritmos científicos irregulares. Este desajuste hace que sea difícil para un programador que utiliza el modelo de programación paralela plano construir una forma computacional de dicho algoritmo que exponga y utilice todo el paralelismo inherentemente disponible dentro del algoritmo. Es decir, debido a las limitaciones en el modelo de expresividad, se pierden muchas oportunidades para la ejecución en paralelo.

Se ha propuesto una generalización simple del modelo de programación paralela tradicional. Donde el paralelismo de datos anidados sea un medio para aumentar la expresividad dentro de los modelos de programación paralela para eliminar muchas de estas serializaciones forzadas. Siendo necesario analizar el paradigma y describir cómo su aplicación conduce a la especificación natural y altamente paralela de algoritmos paralelos a través de datos estructurados irregularmente. Después de esto, se deben analizar las técnicas que se han adoptado para implementar el esquema generalizado en arquitecturas de multiprocesadores normales. Ninguno de los enfoques carece de defectos y ninguno cumple con precisión los requisitos de la computación científica irregular.

Conceptos de paralelismo de datos anidados

La base del paradigma del paralelismo de datos anidados radica en una generalización simple del modelo tradicional o plano) del paralelismo de datos. El modelo plano deriva el paralelismo al aplicar la misma secuencia de cálculo a cada elemento de un agregado de datos en paralelo, insistiendo en que este cálculo por elemento es puramente en serie. El paradigma del paralelismo de datos anidados, por el contrario, permite que el cálculo por elemento de un operador dentro de un modelo paralelismo de datos anidados contenga construcciones paralelas en forma de otras instancias de operador. Es decir, el paradigma permite que los operadores de los modelos de paralelismos de datos se aniden de forma análoga a la anidación disponible para otras construcciones de lenguaje por ejemplo, bucles seriales. Es importante tener en cuenta que la semántica de tal anidamiento de operaciones dentro de modelos de paralelismos de datos implica la existencia de múltiples niveles o dimensiones de paralelismo.

Lenguajes de programación paralelos de datos anidados

Varios lenguajes admiten la especificación de operaciones de modelos de paralelismos de datos a través de la composición estructurada de operadores planos. Estos se denominan lenguajes paralelos de datos anidados. Varios de estos lenguajes se describen brevemente a continuación en términos de las construcciones que proporcionan y el alcance que ofrecen para la expresión de modelos de paralelismos de datos.

Modelos de paralelismos de datos en lenguaje Fortran

Existen muchos dialectos de Fortran en los que se agregan características de modelos de paralelismos de datos ya sea mediante la provisión de instrucciones especiales de indexación o mediante construcciones de bucles paralelos entre matrices. Aquellos que ofrecen construcciones de bucle a menudo atienden su anidamiento dentro de un programa.

Esto, en esencia, define un cálculo dentro de los modelos de paralelismos de datos. Sin embargo, si bien estos lenguajes permiten la especificación de múltiples dimensiones de paralelismo dentro de un cálculo, sus compiladores rara vez son capaces de explotar dicho paralelismo. Esto normalmente se debe a las complejidades analíticas derivadas del hecho de que estos son lenguajes imperativos, a menudo es difícil para un compilador determinar cuáles de las instancias dentro del bucle paralelo anidado tienen interdependencias. Frente a los compiladores de tales dificultades, generalmente adoptan un esquema de serialización simple para tales bucles.

Modelos de paralelismos de datos en lenguaje SISAL

Los idiomas SISAL tienen su base en la expresión de alto nivel de la semántica de flujo de datos y no son explícitamente lenguajes muy utilizados en los modelos de paralelismos de datos. Sin embargo, proporcionan una serie de operaciones tipo de programación paralela en los tipos de vectores y matrices del lenguaje. Particularmente de interés es el paralelo para al programador que permite un cuerpo de bucle para crear una instancia de forma independiente para cada elemento de un rango o para cada índice de un vector. Esta construcción consta de tres partes: un generador que especifica el rango de la iteración, un cuerpo de bucle que detalla el cálculo por instanciación y una sección de valor de retorno que describe cómo se deben combinar los resultados de las instancias de cuerpo de bucle para formar un único resultado valor para el bucle en sí. Este paso de combinación representa una operación de reducción similar a la que se encuentra en la mayoría de los lenguajes utilizados en los modelos de programación paralela. SISAL ofrece una selección típica de funciones combinadas para esta reducción, incluida la suma, selección basada en algunos criterios y la construcción de una matriz a partir del conjunto de resultados individuales.

SISAL permite la anidación completa de los bucles de dos maneras. El primero es a través de la encapsulación textual completa de un bucle dentro de una expresión del cuerpo del bucle de otro bucle. La segunda forma de anidamiento utiliza una forma sintáctica especial en la que se especifica un solo cuerpo y una sección de valor de retorno, pero el generador incluye múltiples rangos separados por la palabra clave ***cross.*** Ambas formas de especificación son completamente generales y pueden usarse para especificar iteraciones de los modelos de paralelismos de datos a través de datos estructurados irregularmente.

Modelos de paralelismos de datos mediante Paralation Lisp

Paralation Lisp es un dialecto de Common Lisp que agrega un nuevo constructor de datos, el campo, que permite la especificación de una colección ordenada de elementos tipo matriz. Los elementos de campo pueden ser de cualquier tipo, incluidos los tipos de campo, lo que permite la construcción de agregados anidados irregulares. El lenguaje permite la heterogenia de tipos entre elementos del mismo campo.

Se definen dos características de Paralation Lisp en base a la postura del operador. El primero representa un iterador paralelo que acepta un conjunto de campos y una expresión corporal, calculando un campo de resultado aplicando la expresión corporal a cada elemento de cada campo de argumento. Esto es equivalente a una o más instancias del mapa de operador de aplicar a todos introducido en la pseudo sintaxis.

Una segunda característica ofrecida por el lenguaje, donde el operador, permite que una permutación generalizada de un campo se calcule en paralelo. El operador acepta un campo fuente y un mapeo. Este último argumento puede considerarse como un conjunto de flechas unidireccionales que conectan elementos del campo fuente con posiciones en el campo resultante. Durante la operación de movimiento, cada elemento de la fuente es empujado hacia abajo por las flechas que se extienden desde su ubicación. Varias flechas que salen de la misma ubicación implican una lectura concurrente del elemento fuente; una flecha múltiple que llega a una posición de resultado único implica una combinación es decir, una reducción de varios elementos fuente. La última situación es manejada por la especificación del usuario de una función de reducción para combinar elementos en colisión.

Las operaciones brindan oportunidades para que las expresiones generales se definan como cálculos por elemento por lo demás, esto constituye el cuerpo de la iteración, para la operación de movimiento deriva de la posibilidad de una función de combinación especificada por el usuario. Específicamente, estas expresiones por elemento pueden contener operadores expresando así el cálculo dentro de los modelos de paralelismos de datos en lenguaje. Dada la naturaleza del constructor de campo como un agregado unidimensional, dicho anidamiento es importante para la especificación de operaciones paralelas a través de campos anidados utilizados para representar estructuras de datos complejas posiblemente irregulares

Modelos de paralelismos de datos mediante Nesl

El lenguaje Nest de Blelloch está diseñado específicamente con la especificación dentro de modelos de paralelismos de datos en mente. El paralelismo en el lenguaje surge explícitamente a través de operaciones a través de secuencias homogéneas de valores. Los datos estructurados irregularmente pueden representarse como un anidamiento de secuencias; las operaciones paralelas a través de dichos agregados pueden especificarse mediante una anidación de los operadores disponibles para la iteración a través de secuencias simples no estructuradas.

Finalmente, el lenguaje también proporciona una serie de operaciones dentro de los modelos de paralelismos de datos importantes para calcular operaciones paralelas a través de secuencias. Estos operadores calculan en paralelo una secuencia que contiene valores que son acumulaciones parciales de los elementos del original utilizando una función de acumulación particular. Por ejemplo, el operador de exploración plus, cuando se aplica a una secuencia, genera la secuencia de todas las sumas parciales, es decir, la secuencia resultante es la suma de todos los elementos de secuencia fuente hasta la última posición.

Metodología

Se realizará un análisis exhaustivo de la literatura científica relacionada con las variables de estudio de la investigación, en especial de los artículos referentes a la experiencia de crear modelos de ejecución generalizado para sistemas computacionales paralelos de datos anidados.

Sera una investigación con enfoque cualitativo porque se realizarán observaciones y análisis de opiniones de las investigaciones referentes al tema de estudio. El tipo de estudio será exploratorio pues se basará en una investigación de campo para reconocimiento de información científica. El tipo de investigación será de campo, pues se realizará la investigación *in situ*, con datos reales tomados mediante las herramientas de recolección de datos para entender la correlación de las variables de estudio. Para la investigación se realizará una convergencia metodológica, se utilizará la observación que permitirá analizar los diferentes casos de estudio y el método empírico.

Estrategia metodológica de diseño de la investigación y recolección de datos

La investigación se dividirá en tres etapas: la primera etapa consistirá en la aplicación de un cuestionario (inicialmente exploratorio), que se prolongará en todas las demás etapas. Lo indicado es comenzar con un estudio de este tipo que permita preparar el terreno para la investigación posterior. Esta circunstancia definirá el carácter y la profundidad con que se diseñará la investigación. A los efectos de precisar el lenguaje utilizado para caracterizar los estudios, se aclara que la terminología utilizada responde a la clasificación de Dankhe (1989), adoptada por Sampieri, (1998), quien los divide en: exploratorios, explicativos, descriptivos o correlacionales. En la segunda etapa se realizará un estudio explicativo-correlacional (evaluaciones de conceptos), y paralelamente un estudio descriptivo (encuesta para medir habilidades), que se prolongará en la tercera etapa.

Para la segunda y tercera etapa, la investigación se diseñará de la siguiente forma: Se planifica realizar, por un lado, un estudio explicativo y correlacional a través de experimentos y por otro lado un estudio descriptivo a través de encuestas. En una primera instancia el estudio se considera será descriptivo, luego ambos estudios se realizarán en forma paralela, aunque temporalmente el estudio descriptivo será realizado, al finalizar las experiencias con los trabajos prácticos propuestos. Se trabajará con el 100 por ciento de la población de los estudiantes de quinto semestre de la Universidad Agraria del Ecuador, de la carrera de Ingeniería en Sistemas.

Resultados

Se procurará resaltar los resultados y las observaciones más relevantes de la investigación, describiéndose, sin hacer juicios de valor, el material y métodos empleados para el análisis. Los resultados se expondrán en figuras o/y tablas. Aparecerán en una secuencia lógica en el texto, las tablas o figuras imprescindibles, evitando la redundancia de datos.

En el primer caso, se estaría especificando un cálculo en el que 8 cálculos en serie por elemento están activos simultáneamente cada uno considerando los índices de un vector interno en serie. La segunda situación involucra una secuencia de operaciones paralelas: la primera comprende 4 cálculos por elemento activos simultáneamente, la siguiente consiste en 8, y así sucesivamente. Si bien cada una de estas formas de programación paralela es una expresión paralela del problema, ninguna expone el paralelismo completo del problema: la multiplicación de cada uno de los elementos de la matriz por una constante es completamente independiente de la multiplicación de cualquier otro elemento, por lo tanto, se podrían realizar las 40 multiplicaciones en paralelo.

Son factores como este los que hacen que el modelo de programación paralela sea de uso limitado para los programadores científicos que consideran la paralelización de sus códigos irregulares. En cambio, los desarrolladores de tales sistemas están obligados a usar ya sea construir un modelo de ejecución paralela apropiado para su programa a partir de construcciones paralelas de bajo nivel, o manipular su problema para permitir una expresión en algún otro paradigma paralelo de alto nivel por ejemplo, un modelo funcional implícitamente paralelo, o un modelo paralelo orientado a objetos. Para muchos problemas, ninguno de estos enfoques es completamente satisfactorio: una alternativa más atractiva sería el desarrollo de un modelo de programación paralela generalizado que podría soportar el paralelismo entre estructuras irregulares.

Resumen comparativo de los idiomas utilizados en los modelos de paralelismos de datos

La mayoría de los idiomas utilizados en los modelos de paralelismos de datos implementan un conjunto común de operaciones anidación de datos: una construcción aplicada a cada uno, una permutación paralela y varias operaciones de reducción y prefijo paralelo. La flexibilidad que brinda el programador en la especificación de dicha operación, sin embargo, varía entre lenguajes y entre operadores dentro del mismo idioma. Mientras que a un programador se le puede permitir especificar un cálculo arbitrario como el código por elemento para un tipo de construcción, el mismo lenguaje puede restringir las opciones de código por elemento para otra construcción a un pequeño conjunto de opciones definidas por el sistema. La Tabla 1 resume, para cada uno de los idiomas utilizados en los modelos de paralelismos de datos considerados hasta ahora, así la disponibilidad de diferentes construcciones y la flexibilidad ofrecida en su uso.

Tabla 1. Comparación de operaciones de paralelismo de datos según el idioma utilizado

Operación de paralelismo de datos	CM Fortran	SISAL	Paralation Lips	Nesl
Aplicar a cada código fuente	Sí	Si	Sí	Sí
Combinador de permutación	Sí	No	Sí	No
Combinador de reducción	Sí	Si	Sí	Sí
Escanear combinador	Sí	No	Sí	Sí

Elaborado por: los autores 2022

Soluciones de hardware SIMD

Por definición, una computadora SIMD es aquella que aplica una sola instrucción de máquina a varios elementos de datos diferentes simultáneamente. Típicamente, dicha máquina está organizada como una colección de elementos computacionales simples, cada uno con una memoria asociada que almacena el elemento de datos sobre el cual ese nodo actuará durante un cálculo. En cada ciclo de reloj, se transmite una instrucción a cada nodo computacional con lo cual se lleva a cabo en el contexto de los registros o memoria que es propiedad del nodo. Se produce un punto de sincronización de la máquina al final de cada instrucción de la máquina: una nueva instrucción solo se transmitirá después de que todos los nodos hayan completado la instrucción anterior. Esta sincronización global se proporciona directamente en hardware. (Globa, 2008)

Soluciones de hardware SPMD

La modificación de las tecnologías de procesador, junto a las observaciones generales relativas a la dificultad de obtener un buen rendimiento para los problemas del mundo real ejecutados en máquinas SIMD, tiene plomo arquitectos informáticos a considerar alternativas para apoyar el paradigma del uso de los datos paralelos en el hardware. Los candidatos obvios, fueron las máquinas MIMD de uso general actualmente. (Liu, 2020)

La experiencia ha demostrado que se puede obtener una mejor arquitectura para la ejecución de operadores de datos en paralelo especializando una máquina MIMD general al agregar hardware que implementa directamente estos protocolos de sincronización global. Se pueden realizar más optimizaciones de hardware asumiendo la situación predeterminada de cada nodo que ejecuta una imagen de código idéntica. Las caracterizaciones de las arquitecturas derivadas de tales especializaciones como máquinas SPMD ya que (como fue el caso en el modelo SIMD) todos los procesadores están ejecutando efectivamente el mismo programa en diferentes datos en paralelo.

Sin embargo, a diferencia de la ejecución que se encuentra en las máquinas SIMD, la arquitectura SPMD no tiene sincronización implícita por instrucción: los nodos proceden de forma independiente en su ejecución, excepto en los puntos del programa donde las instrucciones especiales de la máquina hacen que se sincronicen con los otros nodos. (Liu, 2020)

Las instalaciones de la red de control del permiso de la máquina Blue Gene/L posee implementaciones muy eficientes para la realización de las operaciones de datos en paralelo, casi de la misma manera la sincronización barata de la CM-200 era un factor favorable en el suministro de implementaciones de datos en paralelo según su arquitectura. Sin embargo, la red de árbol gordo ofrece una red de comunicación que, aunque está optimizada para la comunicación regular, ofrece un medio de gran ancho de banda para la comunicación punto a punto. Esto ofrece oportunidades considerables para la ejecución eficiente de los segmentos de un programa que no posee datos en paralelo, lo que hace que la Blue Gene/L sea considerablemente mejor para ejecutar problemas prácticos.

Discusión

El considerable interés de investigación en las máquinas SIMD que prevaleció en los años 1970 y 1980 representa en parte las actitudes predominantes hacia el diseño teórico de las computadoras paralelas, pero es más representativo de los problemas de ingeniería, como las compensaciones de precio-rendimiento en hardware disponibles en ese momento. En resumen, fue solo mediante la unión de una gran cantidad de elementos de procesamiento muy simples que se pudo construir una computadora paralela rentable. (Moreira, 2016)

Uno de los primeros diseños para incorporar este principio en un producto lanzado comercialmente fue el procesador de matriz distribuida ICL (Carratalá Sáez, 2016). Esta arquitectura se caracterizó por elementos computacionales muy simples (procesadores en serie de un solo bit) conectados entre sí en una cuadrícula de procesadores bidimensionales. Las primeras configuraciones del procesador de matriz distribuida comprendían cuadrículas cuadradas de 1024 (32 x 32) nodos; las versiones posteriores permitieron conectar 4096 elementos. El procesador de matriz distribuida se concibió originalmente como un procesador de matriz de propósito especial dedicado que se conectaría a un mainframe ICL (de la misma manera que se agregaron procesadores vectoriales externos a arquitecturas anteriores para proporcionar computación vectorial de alta velocidad). Efectivamente, el procesador de matriz distribuida funcionaba como una unidad especial de la memoria de la unidad central: los valores podían leerse y escribirse desde los bancos individuales de memoria asociados con cada nodo de procesamiento. Las instrucciones especiales emitidas por el mainframe causarían que algunos o todos los procesadores realicen cálculos en el contexto de su memoria local y se registren de forma SIMD. Los datos podrían moverse entre procesadores a lo largo de la red que conectaba cada uno con sus vecinos de red más cercanos al Norte, Sur, Este y Oeste. La naturaleza limitada de esta conectividad y la falta de un enrutador general significaba que solo se podían admitir patrones de comunicación simples y regulares.

Ejemplo de ello es Thinking Machines CM-2 ya que utilizo una arquitectura SIMD que incorporo muchas de las mismas nociones, pero también agrego nuevas características al diseño. Al igual que el procesador de matriz distribuida, Thinking Machines CM-2 se componía de un gran número de elementos de procesamiento de un solo bit, cada uno con una memoria asociada. Los sistemas comerciales se vendieron con entre 4096 y 65536 nodos. Para mejorar el rendimiento computacional de los nodos, Thinking Machines decidió agrupar los procesadores en vecindarios de 32 elementos y asociar un coprocesador de punto flotante basado en Weitek con cada vecindario. El diseño fue defectuoso por el hecho de que la ruta de datos entre las memorias nodales y la FPU no proporcionó suficiente ancho de banda para alimentar los datos a la unidad a una velocidad que alcanzó la calificación MFLOPS teórica del chip Weitek. Los nodos del CM-2 estaban unidos por una red de hipercubos que podría soportar tanto la comunicación regular, como la comunicación general, esto último debido a la presencia de un chip enrutador en cada nodo. Sin embargo, debe tenerse en cuenta que el costo de dicha comunicación general es varios órdenes de magnitud mayor que el uso de modos regulares. Al igual que con el procesador de matriz distribuida, el CM-2 se considera adecuadamente como el back-end para un procesador en serie convencional. (Giri, 2020)

Contemporáneo con el CM-2 fueron las diversas arquitecturas SIMD producidas por MasPar Computer Corporation, en particular los diseños MP-1/2 (Riesen, 2019). Estas fueron máquinas multiprocesador basadas en colecciones de procesadores de 4 bits dispuestos en una configuración de red única llamada X-Net. En esencia, la X-Net consiste en una red bidimensional de procesadores, cada uno de los cuales tiene conectividad directa con sus ocho vecinos más cercanos. Sin embargo, en el hardware, el nodo solo tiene cuatro rutas de datos que se extienden desde él; cada uno de estos conduce a un nodo de enrutamiento que puede cambiar la información a través de la red. Un enrutador global de alta velocidad presente dentro de X-Net permitía a las máquinas MasPar implementar comunicación general, así como los estilos de interacción regulares que se encuentran en el procesador de matriz distribuida, aunque a un costo significativamente mayor. Las configuraciones de MP-1 y MP-2 típicamente consistían en tantas como 16.384 nodos de 4 bits, cada uno de los cuales integrado una unidad de coma flotante/número entero de mejorar su potencial computacional. Para ello los nodos ejecutaron instrucciones transmitidas desde una estación DEC 5000 front end, de forma SIMD. (Giri, 2020)

Las características de los multiprocesadores SIMD como procesador de matriz distribuida, CM-2 y MasPar ofrecen considerables beneficios para la implementación eficiente de operaciones de datos paralelos. La sincronización requerida para la ejecución correcta de dicho operador puede derivarse fácilmente de la sincronización que ocurre naturalmente dentro de la máquina tras la ejecución de cada instrucción. Además, los patrones de comunicación que generalmente se encuentran en las operaciones de datos paralelos comunes muestran una regularidad significativa, lo que hace que su implementación en cada una de las tres arquitecturas sea muy eficiente. Estas dos áreas de soporte directo de hardware son responsables conjuntamente de los niveles muy altos de rendimiento de datos paralelos logrados en dichos programas de arquitectura y pueden compilarse en formas muy eficientes. Sin embargo, dicho hardware es bastante ineficiente como base para formas de cálculo menos estructuradas.

Esto lleva a que las secciones de un programa que no son de datos paralelos (por ejemplo, operaciones de indexación de vector) se vuelvan muy costosas hasta el punto de degradar seriamente la eficiencia general del programa. Dado el hecho de que muchos programas prácticos están completamente compuestos de características de datos en paralelo, muchos han visto esto como una limitación seria en la utilidad del hardware SIMD. Este factor puede identificarse como

una de las fuerzas, junto con cambios significativos en el costo/rendimiento de los procesadores seriales estándar, que motivaron una tendencia distinta a la de este estilo de arquitectura paralela especializada.

Experimentos con operaciones de prefijo paralelo

Además de desarrollar nuevas bases para los lenguajes de paralelismo de datos, los investigadores en la década de 1980 también exploraron nuevos modos de expresar computación paralela en el estilo de modelos de programación paralela. La más exitosa de las propuestas que surgieron de este trabajo fue la del operador Paralelo- Prefix o escaneo (Shazeer, 2020). La semántica de esta operación implicaba el cálculo de una serie de sumas parciales para ser calculadas de acuerdo con alguna función de combinación asociativa. Por ejemplo, una operación de exploración plus, cuando se aplica a un vector produciría el vector de sumas parciales. Aunque tal operación parece, a primera vista, inherentemente serial el resultado de cada adición debe calcularse antes de su entrada en la próxima adición, ese no es el caso: técnicas desarrolladas por Hillis y Steele (1986) permite que tales operaciones se calculen en O en una máquina SIMD con procesador. Además, la investigación sobre el poder expresivo de tales operaciones demostró que son sorprendentemente generales, con la capacidad de expresar de manera concisa operaciones complejas como la clasificación de radios paralelas, el análisis del lenguaje e incluso el recorrido de la lista.

La demostración de las operaciones de prefijos paralelos como expresiones prácticas de operaciones paralelas, que también operaron en una forma ampliamente en los modelos de programación en paralelo, llevaron a su introducción en varios idiomas, incluidos CM Fortran y LISP. También influyó en el diseño del hardware que actualmente se utiliza en la Blue Gene. (Stunkel, 2020)

Conclusiones

El modelo de programación en paralelo de datos representa una alternativa más simple a la tarea intrínsecamente compleja de programar bajo tales modelos. Su simplicidad se deriva de la provisión de una vista de nivel superior de la programación paralela que abstrae muchos de los detalles de la arquitectura en paralelo.

La abstracción de nivel superior que ofrece el paradigma significa que el rendimiento final de un modelo de ejecución generalizado para sistemas computacionales paralelos de datos anidados está determinado por la forma en que el compilador lo asigna al hardware paralelo.

La tecnología del compilador requerida para generar tales mapeos se entiende bien y se ha incorporado en una serie de implementaciones de modelos de ejecución generalizado para sistemas computacionales paralelos de datos anidados muy eficientes A través del análisis matemático, derivando una implementación novedosa de tales características que hace uso de un modelo de computación multiproceso en cada procesador de una máquina multiprocesador. El enfoque gira en torno a la partición de los agregados de datos dentro del programa a través de una serie de memorias disjuntas, cada una de las cuales está estrechamente asociada con un procesador.

Referencias Bibliográficas

- Acar, U. A. (2020). Batch-dynamic Algorithms via Parallel Change Propagation and Applications to Dynamic Trees. arXiv preprint arXiv:2002.05129.
- Carratalá Sáez, R. (2016). Aprovechamiento del paralelismo de tareas en factorizaciones de matrices jerárquicas sobre procesadores multinúcleo.
- Danhke, G. (1989). Investigación y comunicación. La comunicación humana: Ciencia social, 385-454.
- Duncan, J. L. (2020). Distinguishing between parallel and serial processing in visual attention from neurobiological data.
- Efros, V. D. (2020). Program to calculate coefficients of transformations between three–particle hyperspherical harmonics. Computer Physics Communications, 107281.
- Giri, S. K. (2020). Purifying electron spectra from noisy pulses with machine learning using synthetic Hamilton matrices. Physical Review Letters, 124(11), 113201.
- Globa, L. S. (2008). Parallel computing process algorithm. In 2008 International Conference on" Modern Problems of Radio Engineering, Telecommunications and Computer Science"(TCSET) (pp. 467-469). IEEE.
- Hillis, W. D. (1986). Data parallel algorithms. Communications of the ACM, 29(12), 1170-1183.
- Jonasson, K. S. (2020). Algorithm 1005: Fortran Subroutines for Reverse Mode Algorithmic Differentiation of BLAS Matrix Operations. ACM Transactions on Mathematical Software (TOMS), 46(1), 1-20.
- Júnior, W. &. (2020). Estudo e aplicação de visão computacional e redes neurais para localização e detecção de falhas em montagem de PCBS.
- Kumbhar, P. A. (2019). An optimizing multi-platform source-to-source compiler framework for the NEURON MODeling Language. arXiv preprint arXiv:1905.02241.
- Kumbhare, S. &. (2020). Parallel Adapative Monte Carlo Optimization, Sampling, and Integration in C/C++, Fortran, MATLAB, and Python. Bulletin of the American Physical Society.
- Liu, C. Z. (2020). An efficient iterative graph data processing framework based on bulk synchronous parallel model. Concurrency and Computation: Practice and Experience, 32(3), e4432.
- Moreira, R. E. (2016). Definição semântica de blocos everywhere para programação simd.
- Naterop, L. S. (2020). handyG—Rapid numerical evaluation of generalised polylogarithms in Fortran. Computer Physics Communications, 107165.
- Reiser, B. J. (2020). Knowledge Representation and Explanation in G| L, An Intelligent Tutor for Programming. Computer assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches, 111.
- Riesen, R. G. (2019). Overview: The Rise of Linux. In Operating Systems for Supercomputers and High Performance Computing (pp. 95-98). Springer, Singapore.
- Sampieri, R. H. (1998). Metodología de la investigación (Vol. 6). México, DF: Mcgraw-hill.
- Shazeer, N. M. (2020). U.S. Patent Application No. 16/682,611.
- Stunkel, C. G. (2020). The High-Speed Networks of the Summit and Sierra Supercomputers. IBM Journal of Research and Development.