Modelos de programación y cargas de trabajo para computadores escalables

Diego Federico Camacho Naranjo
Universidad Industrial de Santander
Bucaramanga, Colombia
federicocamacho97@hotmail.com

Sergio Andrés Contreras Corredor
Universidad Industrial de Santander
Bucaramanga, Colombia
sergioandrescontreras@yahoo.es

Juan Sebastian Carrillo Rodriguez
Universidad Industrial de Santander
Bucaramanga, Colombia
carrillosebas@hotmail.com

***Abstract*—Este documento electrónico, pretende describir e informar acerca de la escalabilidad en los computadores, explicando los modelos de programación y cargas de trabajo usadas en los mismos, para dar un marco teórico que permita entender ciertos modelos de reconfiguración propuestos por una empresa estadounidense que se explicarán junto con sus aplicaciones.**

Keywords—escalabilidad, procesamiento paralelo, programación lineal, computación distribuida, computadores, balanceo de carga, procesos, procesamiento, memoria, tiempo de respuesta, tiempo de ejecución, rendimiento, reconfiguración estática, reconfiguración dinámica, map reduce.

***Abstract*—This electronic document, is meant to give some information about scalability in computers, explaining some programming models and workloads used in them, to give a theoretical framework that helps to identify some reconfiguration models proposed for an American company that are explained along with their applications.**

Keywords— scalability, parallel processing, linear programming, distributed computing, computers, load balancing, processes, processing, memory, response time, runtime, performance, static reconfiguration, dynamic reconfiguration, map reduce.

I. Introducción

La tecnología se ha convertido en algo tan crucial para la humanidad, tanto que al día de hoy muy pocas personas viven sin la influencia de algún aparato tecnológico; el gran uso de la tecnología, la gran demanda de productos innovadores, y la competencia ha generado la evolución de la programación en diferentes formas;el problema es que la cantidad de datos aun sigue siendo una dificultad para la programación, por esto se buscan formas de procesar estas cantidades enormes de datos de forma efectiva y eficaz, mediante la utilización de diferentes avances tecnológicos y diferentes reconfiguraciones en estos mismos que se han planteado a lo largo de su evolución; un ejemplo de esto es VSC computers, pioneros en computación, principalmente en servicios en la nube y dirigidos a Internet; en este artículo se dará una fundamentación teórica y se explicará este tipo de arquitecturas y un ejemplo de uso de las mismas en Colombia.

II. marco teórico

A. *Escalabilidad*

La escalabilidad es la capacidad de adaptación y respuesta de un sistema con respecto al rendimiento del mismo a medida que aumentan de forma significativa el número de usuarios del mismo. Está íntimamente ligada al diseño del sistema. Influye en el rendimiento de forma significativa.

Se pueden distinguir dos tipos de escalabilidad, vertical y horizontal:
El escalar verticalmente o escalar hacia arriba, significa el añadir más recursos a un solo nodo en particular dentro de un sistema, tal como el añadir memoria o un disco duro más rápido a una computadora.
La escalabilidad horizontal, significa agregar más nodos a un sistema, tal como añadir una computadora nueva a un programa de aplicación para espejo.

B. *Modelos de programación*

Los modelos de programación indican mecanismos

disponibles al programador para expresar la estructura lógica de un programa[10]

Ciertas características identifican el modelo de programación usada, entre estas: complejidad del programa, costo de desarrollo, legibilidad, costo de mantenimiento, rendimiento, estructura de paralelización. Principalmente se distingue entre:

i. Programación lineal

Es un algoritmo a través del cual se resuelven situaciones reales en las que se pretende identificar y resolver dificultades para aumentar la productividad respecto a los recursos, aumentando así los beneficios. El objetivo primordial de la Programación Lineal es optimizar, es decir, maximizar o minimizar funciones lineales en varias variables reales con restricciones lineales (sistemas de inecuaciones lineales), optimizando una función objetivo también lineal.

ii. Programación paralelo

Muchas instrucciones se ejecutan simultáneamente, operando sobre el principio de que problemas grandes, a menudo se pueden dividir en unos más pequeños, que luego son resueltos simultáneamente. Hay varias formas diferentes de computación paralela: paralelismo a nivel de bit, paralelismo a nivel de instrucción, paralelismo de datos y paralelismo de tareas.

C. Computación distribuida

Modelo de computación en paralelo donde intervienen una colección de computadoras que pueden o no estar situadas en distintos lugares y pertenecientes a distintos dominios de administración sobre una red distribuida.
Estas utilizan estándares abiertos para llevar a cabo una tarea u objetivo común. Se caracteriza por su heterogeneidad o sea cada computadora posee sus componentes de software y hardware, los cuales el usuario percibe como un solo sistema. Para el usuario todo es transparente, accede a los demás recursos de la misma manera que accede al suyo propio. Esta colección de computadoras básicamente lo que hace es dividirse el trabajo a realizar en pequeñas tareas individuales, reciben los datos necesarios para esa tarea, la hacen y devuelven los datos para unirlos en el resultado final.

D. Balanceo de carga

El balanceo de carga es una técnica usada para dividir el trabajo a compartir entre varios procesos, ordenadores, u otros recursos. Está muy relacionado con lo sistemas multiprocesales, que trabajan o pueden trabajar con más de una unidad para llevar a cabo su funcionalidad . Para evitar los cuellos de botella, el balance de la carga de trabajo se reparte de forma equitativa a través de un algoritmo que estudia las peticiones del sistema y las redirecciona a la mejor opción

E. Reconfiguración

i. Reconfiguración estática

O configuración en tiempo de compilación; implica parar el sistema y reiniciarlo con una nueva configuración. Útil para procesos de depuración y actualización de sistemas.

ii. Reconfiguración dinámica.

Tiene el objetivo de obtener equilibrio entre velocidad de ejecución y espacio, puede ser total o parcial.

La parcial implica la capacidad de algunos dispositivos de admitir la modificación de parte de la configuración mientras la lógica restante sigue realizando la computación de forma ininterrumpida.

F. Map reduce

Este paradigma de programación permite una escalabilidad masiva a través de cientos o miles de servidores en un clúster de Hadoop (framework de software que soporta aplicaciones distribuidas bajo una licencia libre). El término MapReduce realmente se refiere a dos tareas distintas y separadas que realizan los programas de Hadoop. El primero es el trabajo de mapeo, que toma un conjunto de datos y lo convierte en otro conjunto de datos. El trabajo de reducción toma la salida de un mapa como entrada y combina las tuplas de datos en un conjunto más pequeño de tuplas. Como la secuencia del nombre MapReduce implica, el trabajo de reducción siempre se realiza después del trabajo del mapa.

G. FPGAs

Field Programmable Gate Arrays (FPGAs), son dispositivos semiconductores que se basan alrededor de una matriz de bloques lógicos configurables (CLBs) conectados a través de interconexiones programables. Los FPGAs pueden reprogramarse a los requisitos de aplicación o funcionalidad deseados después de la fabricación. Esta característica distingue a los FPGA de los Circuitos Integrados Específicos de Aplicación (ASICs), que se fabrican a medida para tareas específicas de diseño. Aunque los FPGAs programables de una sola vez (OTP) están disponibles, los tipos dominantes están basados en SRAM que pueden ser reprogramados a medida que el diseño evoluciona. [9]

III. MAP REDUCE

Es un framework utilizado para tratar con cantidades de enormes de datos(big data), este framework lo utilizan diferentes empresas como google o yahoo, permite la recuperación eficiente de información sobre grandes volúmenes de datos, permite establecer unas capacidades mínimas de un sistema de información para poder manejar big data como lo son, escalabilidad, técnicas de indexación, rendimiento.

El hombre siempre ha estado relacionado con la información y a medida que pasa el tiempo, se tiene que almacenar mas y mas información, es tal la cantidad de datos en su mayoría necesarios que se necesita una forma ordenada y precisa para almacenarlos y utilizarlos, es aquí donde entra a

relucir map reduce[12].

Este modelo nace al ver la necesidad de trabajar con grandes cantidades de datos, google necesitaba hacer multiplicaciones con matrices de enormes proporciones, de aquí emerge la popularidad de mapreduce, hablamos de terabytes de información, esto es algo enorme y difícil de utilizar. map reduce fue originalmente desarrollado para apoyar la distribución del proyecto de motor de búsqueda nutch.

Se divide el procesamiento en dos funciones que son map y reduce.

```
map(function, list) {
  foreach element in list {
    value = function(element)
    intermediateResult.add(value)
  }
}

reduce(function, list, init) {
  result = init
  foreach value in list {
    result = function(result, value)
  }
  outputResult.add(result)
}
```

Fig. 1. Implementación de mapreduce.

La figura muestra como implementar map reduce, puede utilizarse para contar la cantidad de animales repetidos en un grupoo[8], las dos funciones son:

- map(): Map funciona agrupando subgrupos de datos de cantidades enormes de datos, esto lo hace según los valores intermedios de los subgrupos para adjuntar.
- reduce(): Reduce se le aplica a cada valor intermedio que representa a los subgrupos.

H. Aplicaciones de map reduce

- Trabajar con big data
- En grep distribuido(varias aplicaciones enfocadas a una tarea que trabajan en conjunto)
- Ordenamiento distribuido
- Construcción de índices invertidos
- Sistemas de recomendación (ej:netflix)
- Machine learning[6]

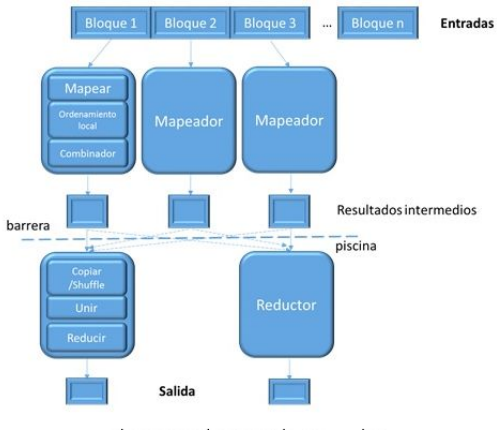

Fig. 2. Modelo de mapreduce.

I. Aplicación de map reduce en contexto colombiano

Map reduce podría aplicarse para cualquier empresa donde se traten cantidades de datos muy grandes, por ejemplo en la registraduría nacional para utilizar de manera eficiente las bases de datos que tienen o en empresas que tengan sistemas de recomendación como netflix, en los buscadores, en diferentes áreas incluso en la investigación, puesto que manejan desde datos únicos y simples hasta resultados de tecnofactos que arrojan miles de datos casi instantáneamente, para procesar todos estos datos se puede utilizar map reduce, otra aplicacion podria ser en biología o medicina puesto que cosas como los genes tienen mucha información, tanta que hasta hace muy poco era imposible sacar siquiera una parte pequeña de esa información y ponerla en una computadora, mapreduce podría trabajar con esos datos ayudando a la investigación colombiana[13].

IV. COMPUTADORES ESCALABLES WSC

J. Descripción

WSC, acrónimo de Warehouse-scale computer (WSC) es la fundación de servicios de internet que ofrece servicios de comunicación vitales actualmente, como lo son: búsqueda, redes sociales, mapas, videos, compras en línea, correo, etc..

La gran diferencia entre WSC y los demás datacenters es que, actualmente WSC funciona como una sola máquina y, disponible para todo el mundo.

Anteriormente, los supercomputadores tenían el propósito de proveer alto desempeño computacional (HPC); sin embargo, el principal objetivo de WSC es proveer tecnologías de información al mundo, por lo que probablemente juegue un rol más importante a la sociedad que los supercomputadores anteriores, y esto se demuestra porque tiene muchos más usuarios que los mismos.

Algunos de los requerimientos de los arquitectos de WSC son: desempeño-costo, eficiencia energética, fiabilidad por

medio de redundancia, amplio paralelismo.

- Relación Desempeño-costo: El trabajo hecho por dólar es crítico en parte por la escala..
- Eficiencia energética: El trabajo hecho por Joule es crítico para WSC y los servidores debido al alto costo de construcción de la infraestructura energética y mecánica.
- Fiabilidad por medio de redundancia: El software en WSC debe proveer al menos un 99,9% de disponibilidad (menos de 1 hora por año de no disponibilidad). Los arquitectos de WSC confían en servidores conectados por medio de una red de bajo costo y, redundancia gestionada por software.
- Amplio Paralelismo: Las aplicaciones por lote se benefician del gran número de conjunto de datos independientes que requieren procesamiento independiente; este procesamiento es paralelismo a nivel de datos, aplicado al almacenamiento, en lugar de a la memoria.

El enfoque principal de WSC

V. XILINX

K. Características

- Xilinx es una compañia tecnologica de los Estados Unidos que vende sus propios dispositivos lógicos programables, uno de sus mejores proyectos ha sido el FPGA. (field programmable gate array)
- Ross Freeman, Bernand Vonderschmitt y James V Barnett III fundaron Xilinx en 1984, queriendo fabricar chips que vinieran en blanco y permitieran a los usuarios programarlos ellos mismos.
- En la actualidad la compañía vende variedad de productos FPGA CPLD(Commplex programmable logic devices).
- La empresa propone una reconfiguración parcial dinámica en FPGA de la cual hablaremos más adelante.

L. Reconfiguración Parcial Dinámica En FPGA

El mercado de FPGA ha crecido rápidamente con gran cantidad de aplicaciones para diferentes industrias, con este crecimiento ha llegado un nuevo concepto que es la reconfiguración parcial dinámica o DPR.

Con esto nos referimos a la capacidad que posee el FPGA para permitir su reconfiguración lógica sin necesidad de cambiar totalmente el dispositivo, esto da pie a los diseñadores para reconfigurar según su necesidad y permite cambiar funcionalidades de las áreas o módulos seleccionados de un FPGA en la marcha mientras que las demás partes que no se quieren cambiar, se pueden dejar sin cambiar y esto no afecta el resultado.

Aunque en la actualidad encontramos diferentes arquitecturas dinámicas a disposición de los usuarios, las FPGA destacan sobre el reto puesto que es una tecnología madura sustentada con un amplio proceso de diseño durante varios años.[2]

M. Tipos de reconfiguraciones

- La primera división de los tipos de reconfiguraciones es la configuración estática y la configuración dinámica.
- La reconfiguración dinámica se subdivide en dos: total y parcial.
- Por último la reconfiguración parcial dinámica de divide en dos, diferencial y modular.[3]

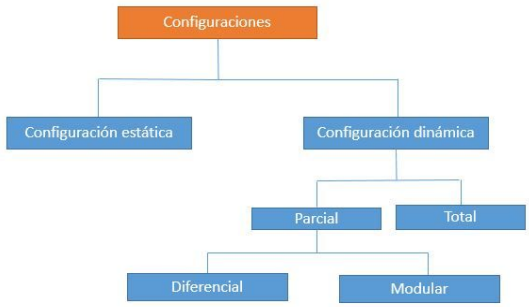

Fig. 3. Mapa conceptual de los tipos de reconfiguraciones.

N. Reconfiguración Estática y Dinámica

- La Reconfiguración Estática; también llamada Reconfiguración en tiempo de compilación, nos lleva a reiniciar el sistema con una nueva configuración, esta reconfiguración es útil para procesos de diseño y actualización de sistemas.
- La reconfiguración dinámica tiene el objetivo de obtener un equilibrio entre velocidad, tiempo de ejecución y espacio.

O. Reconfiguración parcial

Capacidad de algunos dispositivos de admitir la modificación de parte de su configuración mientras se sigue realizando el resto de las operaciones de forma interrumpida; se subdivide en reconfiguración diferencial y modular.[1]

- Reconfiguración diferencial: se refiere a realizar pequeños cambios en un diseño. Ejemplo: cambiar una compuerta, estos cambios son implementados previamente, en el FPGA y después se genera un bitstream que contiene las diferencias entre el diseño inicial y el modificado. Este flujo de diseño permite una rápida descarga del archivo de configuración porque solo se alteran sectores puntuales de la FPGA sin embargo, es un proceso que consume tiempo para generar el bitstream diferencial porque es necesario sintetizar nuevamente todo el diseño, así los cambios realizados sean pequeños.
- Reconfiguración modular o basada en módulo: cuando los cambios se realizan en módulos interrelacionados. Al modificar un comportamiento del sistema se reconfigura solo el diseño del dicho módulo que realiza estas funciones, busca disminuir el tiempo necesario para generar el bitstream parcial, en el cual se divide el área de la FPGA en diferentes secciones de acuerdo a ciertas restricciones impuestas por el fabricante. el bitstream describe la configuración de uno de estos sectores que ha sido instanciado como un módulo del sistema. La síntesis de este módulo corresponde a una parte del sistema total, entonces se gasta menos tiempo que en la reconfiguración diferencial.
- No todos los FPGA admiten reconfiguración dinámica parcial(DRP)

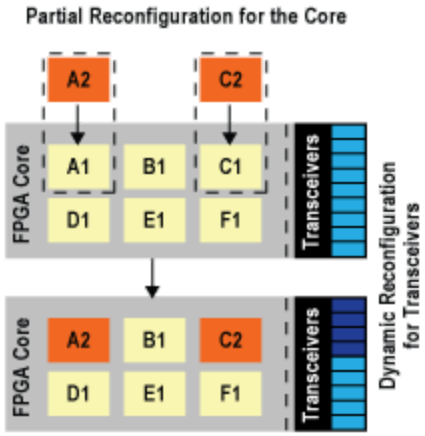

Fig. 4. Reconfiguración parcial del núcleo.

La reconfiguración dinámica parcial es soportada por cualquiera de los siguientes esquemas de configuración:

- Interfaz flash paralela xl6.
- Procesadores internos.

P. Parte estática

Esta parte permanece activa durante todo el tiempo de la ejecución de una aplicación, estática significa que no cambiará, entonces se coloca en el área del dispositivo que no cambia y se mantiene intacta todo el tiempo.

También proporciona la infraestructura para la carga y descarga de las partes dinámicas, debe incluir el controlador de configuración y la lógica necesaria para la gestión de datos e interfaz.

Esta parte gestiona todas las entradas y salidas de la aplicación y se comunica con los módulos dinámicos a través de una interfaz fija.

Q. Parte dinamica

Estas partes dinámicas son independientes del diseño de entrada y pueden estar activas durante el tiempo de ejecución de la aplicación o no estarlo.

Estas partes van dentro de un dispositivo compartiendo zonas comunes puesto que no se requiere la utilización simultánea de todas las partes.

Se cargan y descargan a un dispositivo destino conforme a lo solicitado por el programador del sistema y normalmente se almacenan como un flujo de bits en una memoria fuera de la FPGA

R. Utilidad de los FPGA

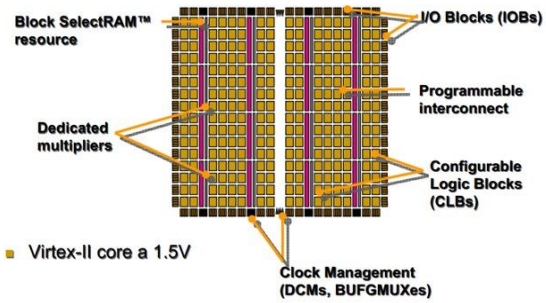

Fig. 5. FPGA.

Los FPGA son muy útiles para aplicaciones que requieren flexibilidad en el cambio del diseño sin la necesidad de modificar partes que no influyen directamente ni resetear por completo el diseño reconfigurando el dispositivo desde el principio o también pueden ser utilizados cuando se tiene el requerimiento de cargar de diferentes diseños en una misma área del dispositivo.

Se pueden utilizar en aplicaciones de alto ancho de banda que admiten interfaces multi-estándar de 150Mbps a 28bps. De esta forma se puede actualizar o modificar la funcionalidad

de la FPGA sin interrumpir los servicios de todos los clientes.

En una actividad en donde el tiempo es crítico, si se tiene que parar para resetear y reconfigurar todo, la pérdida de tiempo afectará directamente el resultado, por lo tanto, si puede hacer una reconfiguración en tiempo de ejecución se permanecerá más tiempo en actividad prestando su servicio; esta es una de las importancias de la capacidad de modificar en tiempo de ejecución[7].

Con el nacimiento de las FPGA y la posibilidad de la programación del hardware muchos investigadores empezaron a buscar unos chips que se pudieran autoreprogramarse físicamente, es decir cambiar su diseño hardware en tiempo de ejecución pero, hasta el dia de hoy esto no ha sido posible.[4]

S. Ventajas

- La reconfiguración parcial dinámica es útil en gran variedad de aplicaciones a través de muchas industrias tales como las aeroespaciales y de defensa. Los dispositivos parcialmente reconfigurables se han beneficiado del sistema de radio táctica conjunta.
- Optimización del sistema mientras que hay una parte del sistema que se encuentra en reconfiguración, otra parte del sistema se encuentra funcionando, es decir no se presenta la inactividad que vuelve ineficientes a otras aplicaciones; esto también permite múltiples aplicaciones en un único FPGA.
- Otra ventaja es que se tiene mayor flexibilidad y capacidad de cambiar el hardware: los FPGA se pueden actualizar en cualquier momento de forma local o remota permitiendo ejecutar varias aplicaciones en un solo FPGA. La reconfiguración parcial nos permite facilidad en el soporte, actualización y ahorro en el espacio necesario para almacenamiento, esto da menor consumo de energía y reduce los costos.
- Se reduce el tiempo empleado en reconfiguración, puesto que se puede modificar solo una parte seleccionada y no el dispositivo completo; se disminuye el tiempo de reconfiguración necesario para el correcto funcionamiento de la aplicación.
- Se mejora la lógica por la posibilidad de la eliminación de funciones que no actúan simultáneamente puesto que, estas funciones se pueden almacenar en memorias externas y cargarlas cuando sea necesario.

T. Inconvenientes

- La reconfiguración parcial se puede aplicar en los dispositivos, pero estos permiten hacer esto de forma diferente cada uno; es decir algunos dispositivos solo permiten la reconfiguración por columnas enteras cada vez, esto lleva al programador a tener que elegir cuidadosamente la tecnología que va a utilizar.
- El diseñador debe optimizar esto dos parámetros para satisfacer la especificación de requisitos técnicos.
- También es necesario diseñar un controlador de configuración.
- El diseñador debe analizar siempre la aplicación con mucho más detalle para tratar de estimar el rendimiento resultante. Además, este debe superar otras tecnologías en parámetros como disipación de potencia, tamaño, peso, coste, tiempo de ejecución, etc, de no conseguir alguna de estas mejoras, usar esa tecnología con DRP no tendrá sentido.

U. Modelos de programación

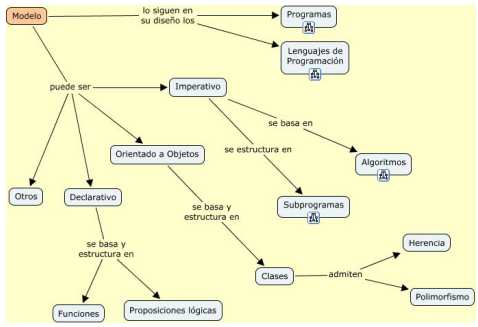

Fig. 6. Diagrama de modelos de programación.

a) Lineal

Parte de la teoría matemática que se ha consolidado en la actualidad como optimización, a groso modo trata sobre un conjunto de técnicas matemáticas usadas principalmente para obtener el mayor beneficio posible en diferentes sistemas, desde tipo tecnológico hasta de tipo social, tal funcionamiento puede describirse básicamente de un modo matemático adecuado.

La programación lineal se desarrolló como modelo matemático para planificar costos y retornos, fue usada principalmente en sus inicios dentro de la segunda guerra mundial para reducir costos militares. Varios fueron los fundadores de estas técnicas matemáticas, entre los cuales se encuentran, Dantzig, quien publicó el algoritmo Simplex, Khachiyan que diseñó el llamado algoritmo del elipsoide, a través del cual demostró que el problema de programación lineal puede ser resoluble de manera eficiente, es decir, en **tiempo polinomial.**

Uno de los ejemplos más conocidos es el de Dantzig, que cuestiona cuál debería ser la mejor asignación de 70 personas en 70 puestos de trabajo, es casi lógico percibir que la potencia de computación necesaria para examinar todas las permutaciones a fin de seleccionar la mejor asignación es muy extensa, sin embargo por medio del modelo de programación lineal y la aplicación del algoritmo simplex, esta teoría se

reduce drásticamente en cuanto a posibles soluciones.

Ejemplo práctico

Usando algoritmo Simplex:

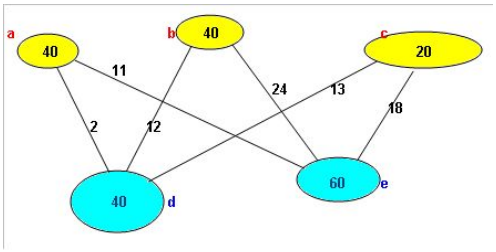

Fig. 7. Ejemplo práctico..

Existen tres minas de carbón cuya producción diaria es:

- La mina **"a"** produce 40 toneladas de carbón por día;
- La mina **"b"** produce 40 t/día; y,
- La mina **"c"** produce 20 t/día.

En la zona hay dos centrales termoeléctricas que consumen:

- La central **"d"** consume 40 t/día de carbón; y,
- La central **"e"** consume 60 t/día

Los costos de mercado, de transporte por tonelada son:

- De **"a"** a **"d"** = 2 monedas
- De **"a"** a **"e"** = 11 monedas
- De **"b"** a **"d"** = 12 monedas
- De **"b"** a **"e"** = 24 monedas
- De **"c"** a **"d"** = 13 monedas
- De **"c"** a **"e"** = 18 monedas

Si se preguntara a los pobladores de la zona cómo organizar el transporte, tal vez la mayoría opinaría que debe aprovecharse el precio ofrecido por el transportista que va de **"a"** a **"d"**, porque es más conveniente que los otros, debido a que es el de más bajo precio.

En este caso, el costo total del transporte es:

- Transporte de 40 t de **"a"** a **"d"** = 80 monedas
- Transporte de 20 t de **"c"** a **"e"** = 360 monedas
- Transporte de 40 t de **"b"** a **"e"** = 960 monedas
- Total **1.400 monedas**.

Sin embargo, formulando el problema para ser resuelto por la programación lineal se tienen las siguientes ecuaciones:

- Restricciones de la producción:

$$X_{a\to d} + X_{a\to e} \le 40\ [\mathrm{T/dia}]$$
$$X_{b\to d} + X_{b\to e} \le 40\ [\mathrm{T/dia}]$$
$$X_{c\to d} + X_{c\to e} \le 20\ [\mathrm{T/dia}]$$

- Restricciones del consumo:

$$X_{a\to d} + X_{b\to d} + X_{c\to d} \ge 40\ [\mathrm{T/dia}]$$
$$X_{a\to e} + X_{b\to e} + X_{c\to e} \ge 60\ [\mathrm{T/dia}]$$

- La función objetivo será:

$$2X_{a\to d} + 11X_{a\to e} + 12X_{b\to d} + 24X_{b\to e} + 13X_{c\to d} + 18X_{c\to e} = Min!$$

La solución de costo mínimo de transporte diario resulta ser:

- $X_{b\text{-}d}$ = 40 resultando un costo de 12 x 40 = 480 monedas
- $X_{a\text{-}e}$ = 40 resultando un costo de 11 x 40 = 440 monedas
- $X_{c\text{-}e}$ = 20 resultando un costo de 18 x 20 = 360 monedas
- Total **1.280 monedas**.

120 monedas menos que antes.

b) Paralela

Básicamente es el uso de varios procesadores que puedan trabajar en conjunto para dar solución a una tarea común, esto mediante la división del trabajo, donde cada procesador realiza una porción de problema mediante intercambio de datos por una red de interconexión, o bien a través de la memoria.

La programación paralela tiene muchas ventajas en cuanto a la optimización del trabajo, ya que, al dividirlo entre varios procesadores, no sólo puede realizarse de una manera más óptima, sino que también se puede ir más allá y resolver problemas mucho más complejos y en un tiempo más razonable.

El rendimiento de las máquinas tradicionales secuenciales está cada vez más saturado, debido a que las aplicaciones día a día requieren de más complejidad en sus trabajos, lo que las hace más ineficaces y poco óptimas, la solución está en el paralelismo, con varios procesadores en sistemas paralelos, obteniendo mejores resultados con más ganancia de eficiencia, esto teniendo en cuenta de que los algoritmos sean igualmente optimizados.[11]

2 técnicas básicas:

- Pipeline:

  Un pipeline es un conjunto de elementos procesadores de datos conectados en serie, en el cual la salida de un elemento está conectado en la entrada del que sigue. los elementos de esta técnica son ejecutados en paralelo, debe haber un tipo buffer insertando los elementos.

Las aplicaciones consisten en múltiples procesos que están ordenados de tal forma que el flujo de salida de un proceso se alimenta de la entrada del siguiente proceso.

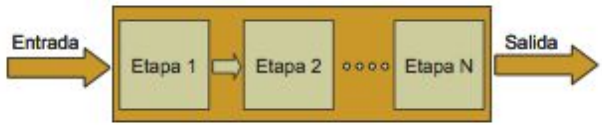

Fig. 8. Pipeline.

- Replicación Asincrónica:

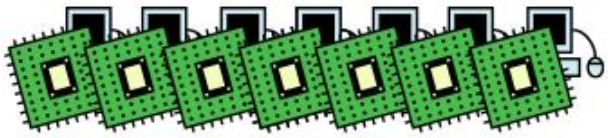

Fig. 9. Replicación asincrónica.

Tecnología en la que una vez que los datos están escritos en el sitio de almacenamiento primario, los nuevos datos a este sitio son aceptadas, sin tener que esperar que el sitio de almacenamiento secundario o remoto termine su escritura[5].

Tipos de Paralelismo:

- Paralelismo de bits:

El aumento del tamaño de la palabra reduce el número de instrucciones que el procesador debe ejecutar para realizar una operación en variables cuyos tamaños son mayores que la longitud de la palabra.

- Paralelismo de instrucciones:

Un programa de ordenador es, en esencia, una secuencia de instrucciones ejecutadas por un procesador. Estas instrucciones pueden reordenarse y combinarse en grupos que luego son ejecutadas en paralelo sin cambiar el resultado del programa. Esto se conoce como paralelismo a nivel de instrucción.

- Paralelismo de datos:

El paralelismo de datos es el paralelismo inherente en programas con ciclos, que se centra en la distribución de los datos entre los diferentes nodos computacionales que deben tratarse en paralelo. La paralelización de ciclos conduce a menudo a secuencias similares de operaciones (no necesariamente idénticas) o funciones que se realizan en los elementos de una gran estructura de datos. Muchas de las aplicaciones científicas y de ingeniería muestran paralelismo de datos.

- Paralelismo de Tareas:

El paralelismo de tareas es la característica de un programa paralelo en la que cálculos completamente diferentes se pueden realizar en cualquier conjunto igual o diferente de datos. Esto contrasta con el paralelismo de datos, donde se realiza el mismo cálculo en distintos o mismos grupos de datos. El paralelismo de tareas por lo general no escala con el tamaño de un problema.

Conclusiones

Se puede ver que hay una gran necesidad de avances tecnológicos debido a la gran cantidad de datos que se generan día a día, debido a esto se generó la existencia de frameworks como mapreduce, para solucionar problemas de big data, si se puede lograr que las máquinas trabajen esta información de manera más eficiente, se podrá conseguir cada vez un mayor entendimiento del mundo que nos rodea.

Cada día se generan más tecnologías para solucionar diferentes problemas existentes y estas tecnologías se pueden ir modificando a medida que se van encontrando fallas e inclusive, se vayan necesitando diversas mejoras para cumplir las exigencias que cada día van aumentando.

Para el caso específico de Xilinx podemos observar que la reconfiguración parcial modular es más rápida que la diferencial puesto que no exige un replanteamiento completo sino solo por módulos.

References

[1] Torrego, R., Val, I., Muxika, E., Iturbe, X., & Benkrid, K. Implicaciones del uso de la reconfiguración parcial dinámica de las FPGAs en la implementación de Radios Definidas por Software.

[2] Quero Llor, J. (2008). Aceleración por hardware de algoritmos basados en soft-computing mediante dispositivos programables y reconfiguración dinámica.

[3] Barreira, J. A. C. (2008). Planificación de grafos de tareas para sistemas multi-proceso dinámicamente reconfigurables.

[4] Clemente Barreira, J. A. (2008). Planificación de grafos de tareas para sistemas multi-proceso dinámicamente reconfigurables.

[5] Almeida, F., Giménez, D., Mantas, J. M., & Vidal, A. M. (2008). *Introducción a la programación paralela*. Thompson Paraninfo.

[6] Chu, C. T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2006, December). Map-reduce for machine learning on multicore. In *NIPS* (Vol. 6, pp. 281-288).

[7] Straka, M., Kastil, J., & Kotasek, Z. (2010, September). Fault tolerant structure for sram-based fpga via partial dynamic reconfiguration. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on* (pp. 365-372). IEEE.

[8] Hernández Dominguez, A., & Hernández Yeja, A. (2015). Acerca de la aplicación de MapReduce+ Hadoop en el tratamiento de Big Data. *Revista Cubana de Ciencias Informáticas*, *9*(3), 49-62.

[9] Patterson, D., & Hennessy, J. L. (2012). *Computer architecture: a quantitative approach*. Elsevier.

[10] Carlos, R. J. (2008). Arquitectura de Computadores.

[11] TERRENCE, W., & ZELKOWITZ, M. (1998). *Lenguajes de*

programación. Diseño e implementación. Pearson Prentice-Hall.

[12] Chu, C. T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2006, December). Map-reduce for machine learning on multicore. In *NIPS* (Vol. 6, pp. 281-288).

[13] Camargo, C., Sánchez, A., & Rosas, N. F. (2013). Reconfiguración parcial en aplicaciones de sistemas digitales utilizando Linux como herramienta para el desarrollo. *Ingenium*, *14*(27), 140-148.