Una mirada entre programación paralela y la tradicional en la práctica educativa[1].

A look from the traditional parallel programming and educational practice.

Rafael Ricardo Mantilla Güiza[2]
Universitaria de Investigación y Desarrollo, Seccional Bucaramanga, Colombia.

Artículo recibido en mes noviembre de 2014; artículo aceptado en enero de 2015

Citación del artículo: Mantilla, R. (2014). Una mirada entre programación paralela y la tradicional en la práctica educativa. *I+D Revista de Investigaciones*, 4(2), 23-33.

Resumen

En los últimos años se ha visto un gran avance en la evolución del hardware, empresas como Intel han desarrollado procesadores potentes con múltiples núcleos (core i3, i5, i7, etc.), con el fin de mejorar los tiempos de ejecución de procesos. Sin embargo el software no ha tenido el mismo crecimiento en sus lenguajes de programación, han surgido lenguajes que facilitan la programación a través de IDEs (Entornos de Desarrollo Integrados), que mejoran la interacción entre las instrucciones, sentencias, funciones, procedimientos de un programa, y en especial el entorno e interfaz del programador; pero, no en el aprovechamiento al máximo de la capacidad hardware que posee el equipo de cómputo con varios procesadores.

Así surge la pregunta de investigación: ¿Cómo mejorar el rendimiento de aplicaciones desarrolladas en visual C++ aplicando técnicas de programación en paralelo con la herramienta Intel Parallel Studio XE?. El objetivo de estudio fue determinar el impacto de aplicar las técnicas de paralelismo en algoritmos de ordenamientos e instrucciones de almacenamiento implementados en el lenguaje C++ con las herramientas de Parallel Studio XE, frente a los algoritmos de ordenamiento e instrucciones de

[1]Artículo investigativo de enfoque cuantitativo, culminado satisfactoriamente, perteneciente al programa de Sistemas, en el área de programación avanzada, desarrollado en el grupo de investigación GIDSAW, financiado por la Universitaria de Investigación y Desarrollo (UDI) de la ciudad de Bucaramanga (Colombia) y con apoyo de la Corporación Intel de Norte América. Dirección Calle 9 #23-55, PBX: 6352525. Fecha de inicio: Agosto de 2014, fecha de terminación: Noviembre de 2014.

[2] Ingeniero de Sistemas, Universidad INCCA de Colombia. Maestría en tecnología y medios innovadores para la educación, Universidad Autónoma de Bucaramanga e Instituto Tecnológico de Monterrey. Docente- investigador del grupo: GIDSAW. Universitaria de Investigación y Desarrollo (UDI) de la ciudad de Bucaramanga (Colombia): Dirección Calle 9 23-55, PBX: 6352525. Correo electrónico institucional: webmaster@udi.edu.co

almacenamiento con técnicas de programación tradicional.

El estudio es de tipo cuantitativo, con un enfoque cuasi experimental, compara dos escenarios de desarrollo. El escenario control (programación secuencial) y el escenario experimental (programación en paralelo) bajo la IDE de visual studio 2010 con su lenguaje de visual C++, incorporando librerías desarrolladas por Intel a través de su paquete Paralell Studio Xe (Padua, 2011).

La programación en paralelo se usa para acelerar la resolución de problemas de alto coste computacional, que tienen por finalidad reducir los tiempos de ejecución partiendo del tiempo secuencial t(n), usando p procesadores para reducir su tiempo de ejecución así: t(n)/p. Este tiempo de ejecución depende de diversas variables: tamaño de la entrada, compilador, máquina, y el programador entre otras.

***Palabras clave*:** programación paralela, paralelismo, multiprocesos.

Abstract

In recent years there has been a breakthrough in the evolution of hardware, companies like Intel have developed powerful multi-core processors (Core i3, i5, i7, etc.), in order to improve the execution times of processes. However, the software has not had the same growth in programming languages have emerged that make programming languages through IDEs (Integrated Development Environments) that improve the interaction between instructions, statements, functions, procedures of a program and especially the environment and interface programmer; but not the maximization of the hardware capacity that has the computer equipment with multiple processors.

So the research question arises: How to improve performance of applications developed in Visual C ++ using parallel programming techniques with Intel Parallel Studio XE tool?. The aim of the study was to determine the impact of applying the techniques of parallelism in algorithms systems and storage instructions implemented in the C ++ language tools Parallel Studio XE, compared to sorting algorithms and storage instructions with programming techniques traditional.

The study is quantitative, with a quasi-experimental approach compares two development scenarios. The stage controller (sequential programming) and the experimental stage (parallel programming) under visual studio 2010 IDE with visual C ++ language, incorporating libraries developed by Intel through its package Paralell Studio Xe (Padua, 2011).

Parallel programming is used to accelerate the resolution of problems of high computational cost, which aim to reduce the execution time of the sequential starting time t (n) using p processors to reduce runtime as follows: t (n) / p. This runtime depends on several variables: input size, compiler, machine, and the programmer among others.

Keywords: parallel programming, parallel, multi-threaded.

Introducción

El estudio permite demostrar que la programación tradicional o estructurada no aprovecha al máximo los recursos que dispone el equipo hardware computacional, que aplicando nuevas técnicas de programación en paralelo se aproveche mejor la tendencia tecnológica en referencia a los procesadores que incorporan los dispositivos móviles, computadores personales, tabletas, portátiles y la creciente gama de dispositivos inteligentes que disponen de procesadores con varios núcleos y no son utilizados eficientemente ahorrando tiempos en operaciones que demandan tiempo en los procesos(Akhter & Roberts, 2006).

La industria de los procesadores ha evolucionado y solucionando problemas presentados para disipar el calor de procesadores cada vez más veloces con la introducción de procesadores que permiten tener en ejecución varios núcleos en un solo chip, reduciendo espacios, aumentando rendimiento y mejorando aspectos relacionados con el consumo de energía y sistema de enfriamiento (Intel, 2014). Su incursión exitosa en diferentes campos como los video juegos, servidores empresariales, equipos de alto rendimiento, data center, simuladores, diseño gráfico, cine, etc. Disponible desde un celular de gama alta, lo que demanda por parte del software y como requerimiento para los programadores de aplicaciones informáticas, desarrollar algoritmos que gestionen eficientemente los recursos que ahora disponen y que con las técnicas de programación estructurada no centraban su potencial en estos aspectos, son ahora una realidad que lo demanda para que los productos desarrollados sean más competitivos en velocidad, consumo de energía, espacio físico, sencillez y funcionalidad (Foster, 1995).

Son innumerables las aplicaciones ya desarrolladas de proyectos o estudios anteriores, las que están en proceso y las que se van a desarrollar, que se verían beneficiadas con aplicar estas modificaciones en el código con las técnicas de programación paralela para mejorar el consumo de recursos y que se verá beneficiado en tiempos de respuesta y consumo de energía. Para demostrar el alcance e impacto del presente estudio se retomó un algoritmo tradicional como son los algoritmos de ordenamiento de registros que se encuentran con facilidad en cualquier libro de programación, indiferente del lenguaje que se esté trabajando. Para el caso del estudio se retomaron 5 algoritmos de ordenamiento: MergeSort, QuickSort, Shell, Improved BubbleSort and SelectionSort (Deitel & Merrel, 2004), para gestionar archivos de textos con 120000 registros y números desordenados de 18 cifras, generados con otro algoritmo que disminuía la oportunidad de repetir números.

Una vez implementados los algoritmos, se realiza un análisis minuciosos desde las pequeñas instrucciones de los cinco métodos de ordenamiento y demuestra la disminución en los tiempos de respuesta, y aunque no se puede generalizar una formula; se puede demostrar una disminución en tiempos del 50%, y en consumo de la CPU del 80%.

Lo anterior permite afirmar que la programación paralela aprovecha eficientemente los recursos hardware disponibles de un equipo de cómputo que los programas desarrollados con

técnicas tradicionales. Respondiendo al objetivo del estudio en determinar el impacto de aplicar las técnicas de paralelismo en algoritmos de ordenamiento, frente a los mismos métodos pero con técnicas de programación tradicional. El estudio permite concluir en la importancia que los estudiantes que cursan carreras o programas que se vinculan con el desarrollo de aplicaciones informáticas conozcan y generen competencias en su incorporación a los diferentes proyectos con el fin de formar profesionales competitivos con sus productos software frente al mercado que demanda menos tiempos en los procesos y mayor funcionalidad de las aplicaciones.

Método

Tipo de estudio

Investigación de tipo cuantitativo con enfoque cuasi experimental que ejecuta un algoritmo base llamado escenario control (con programación tradicional con codificación nativa), y otro con la manipulación del investigador en algunas sentencias de programación para medir su efecto en otras variables llamado escenario experimental (con paralelización en el código fuente). El diseño es correlacional por su propósito de mostrar y determinar la relación entre los resultados en las variables del desempeño y utilización de los recursos hardware con la medición de las variables relacionadas con el consumo de recursos en relación con el tiempo, es decir, velocidad de respuesta de la CPU al gestionar los procesadores en la ejecución del algoritmo de almacenamiento y ordenamiento a través de 5 métodos con vectores.

Participantes

Para el estudio, se preparó un escenario de pruebas que parte de un archivo con 120000 registros numéricos, del cual cada registro es un número con una longitud comprendida entre 8 y 18 dígitos con un orden aleatorio, que posteriormente se guardan en un vector que llevara a cabo el ordenamiento a través de cinco (5) métodos de ordenamiento diferentes y que finalmente guarda en un nuevo archivo de texto que contiene los 120000 números ordenados de forma ascendente.

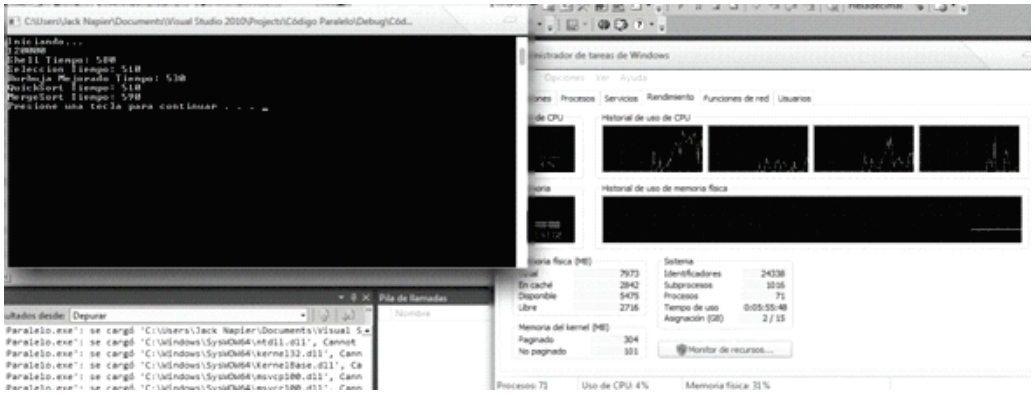
Materiales e instrumentos

Los instrumentos utilizados para recolectar los datos y poder establecer su relación corresponden a la observación natural y análisis de imágenes con los resultados presentados en unidad de tiempos de respuesta por instrucción, ciclo y sentencia, a través de las herramientas de Vtune Amplifier, Inspector y Advisor de la aplicación de Paralell Studio XE de Intel que opera articuladamente con la IDE de Visual Studio 2010 (Supalov, Semin, Dahnken, & Klemm, 2014); también, los tiempos de reloj desde sentencias de captura de tiempo dentro del código fuente y finalmente con el consumo de recursos presentado gráficamente con el administrador de tareas en recursos de Windows 7.

Observación natural.

Con la representación gráfica desde herramientas de Intel, el código de captura dentro del algoritmo y el administrador de tareas se establecen el punto de comparación para determinar el impacto del algoritmo modificado con paralelismo (ver figura 1).

Figura1. Representación gráficamente de los datos para comparar los dos algoritmos.



Fuente: propia.

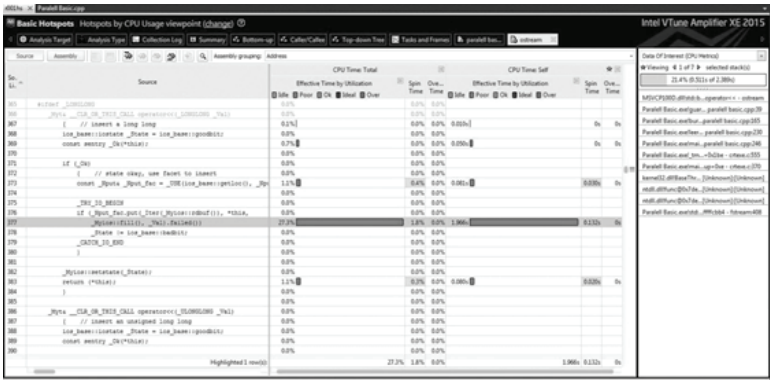
Procedimiento

El estudio genero un escenario de control para establecer los valores iniciales de comparación con las herramientas de recolección de datos (vtune, inspector, advisor, tiempos desde código y administrador de tareas) que consistía en un archivo de texto con 120000 registros aleatorios, del cual cada registro corresponde a un número con longitud entre 8 y 18 dígitos para ser ordenado a través de un algoritmo códificado en visual c++, con la IDE de Visual Studio 2010. Este código es totalmente en código nativo, sin llamar a librerías especiales, completamente con las funciones básicas del lenguaje C++.

El ordenamiento se da a través de cinco (5) métodos diferentes (MergeSort, QuickSort, Shell, Improved BubbleSort y SelectionSort) apoyándose en vectores para cargar los datos del archivo, aplicar el algoritmo de ordenamiento, y que finalmente por cada método genera un archivo plano con los 120000 valores de forma ordenada, capturando los tiempos que tarde por método la ejecución del mismo.

Seguidamente de su correcta ejecución se aplican los instrumentos de recolección de datos, con la herramienta Intel Vtune Amplifier (Reinders, 2005) se recolecto información sobre los tiempos de CPU (ver figura 2).

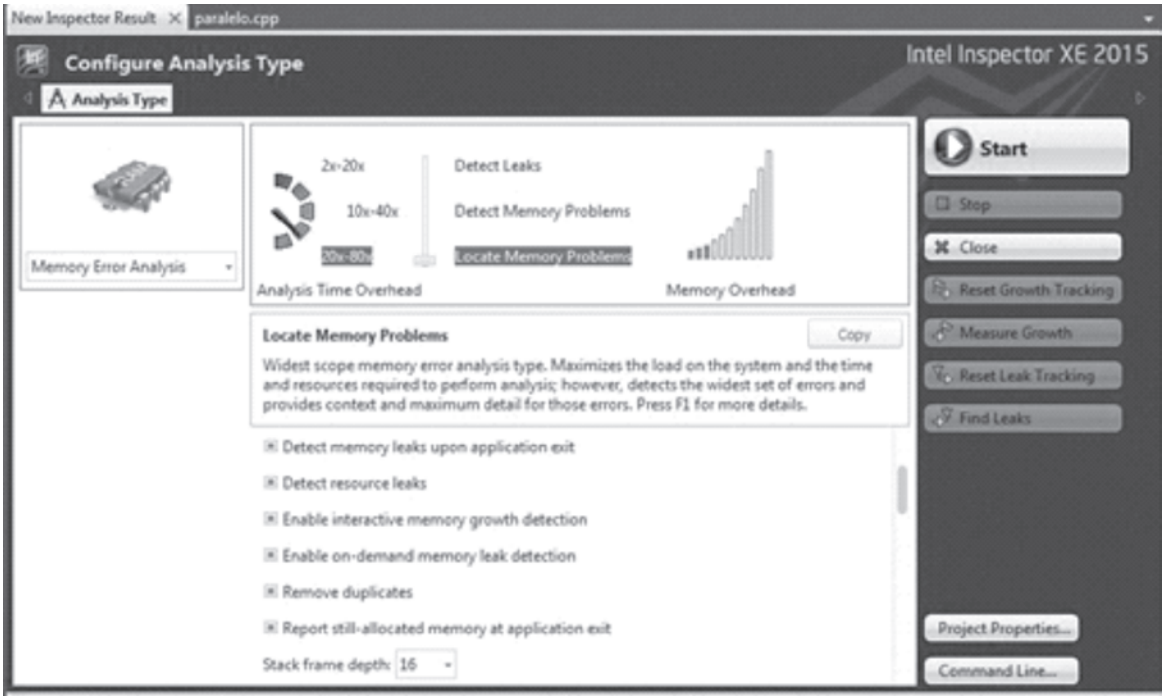
Figura 2. Resultados de Vtune Amplifier



Fuente:

Con el Intel Inspector sobre aspectos relacionados con la memoria (Blair-Chappell & Stokes, 2012) (ver figura 3).

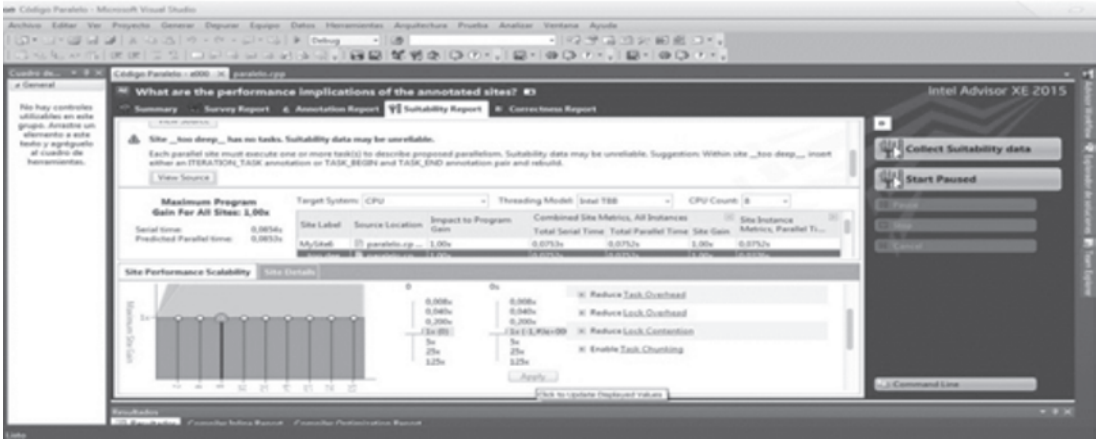
Figura 3. Resultados de Inspector



Fuente:

Con el Intel Advisor para identificar los tiempos y el código que se puede optimizar con programación en paralelo (Supalov, Semin, Dahnken, & Klemm, 2014) (ver figura 4).

Figura 4. Resultados de Advisor



Fuente:

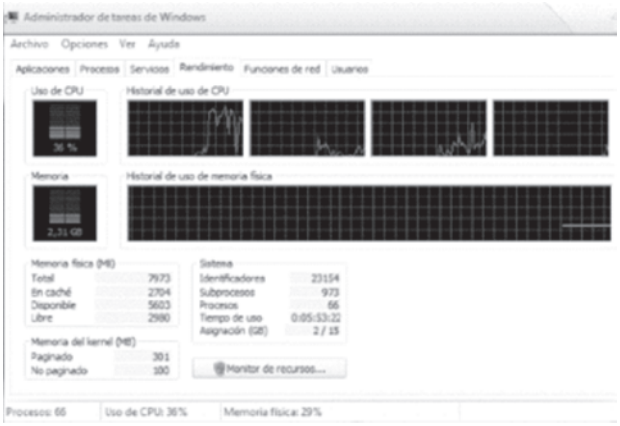
Los tiempos de respuesta capturados desde el código fuente nativo (ver figura 4).

Figura 5. Tiempos por método de ordenamiento

Fuente:

Y por último el uso de procesadores y de CPU por parte del administrador de tareas, sección de rendimiento en Windows 7 (ver figura 6).

Figura 6. Administrador de tareas - Rendimiento



Fuente:

Una vez se tienen estos datos, se ejecuta el programa de Intel Advisor y realiza las anotaciones sugeridas dentro del código fuente, a esta etapa la llamáramos paralelizar el código, que a través de estas anotaciones llama a librerías para acceder y utilizar de forma eficiente los procesadores y recursos con los que cuenta el equipo de cómputo de forma simultánea.

Y finalmente volver a recolectar los datos sobre este nuevo código paralelizado que es el algoritmo experimental para tomar los nuevos datos y comparar con los anteriores para establecer la relación y el impacto que son el objetivo del estudio para presentar los resultados y responder la pregunta de investigación.

Resultados

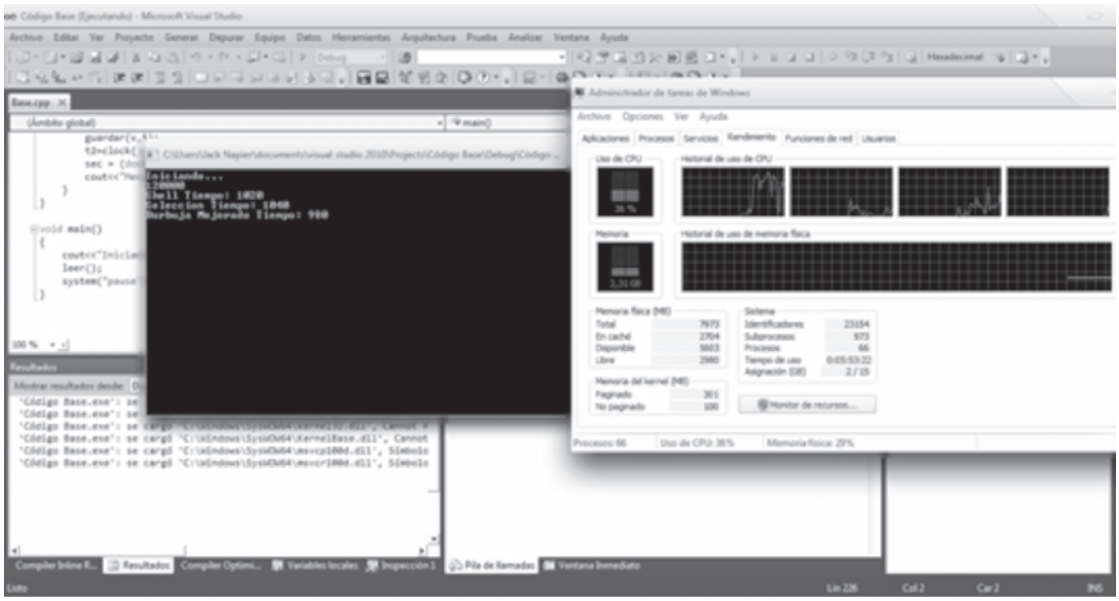
Las herramientas de Intel permiten revisar con detalle aspectos que antes no se podían cuantificar como es el tiempo y consumo de recursos por cada una de las sentencias de programación de forma detallada.

Esto no solo permite mejorar el rendimiento de una aplicación informática sino revisar posibles hábitos y falencias en los programadores, como agentes que influyen directamente con la eficiencia de un programa.

La programación tradicional ha generado la visión de revisar un problema de forma secuencial, un evento después de otro; pero, el mundo real sucede en paralelo, lo que debe reflejarse en los algoritmos con la utilización de los diferentes recursos que dispone un equipo de cómputo, y para esto Intel facilita la paralelización de un programa con su herramienta de Intel Advisor.

En el escenario de control se dio un consumo de recursos en la CPU del 32%, que permite apreciar como utiliza un procesador mientras los otros no son utilizados eficientemente para resolver el problema (ver figura 7).

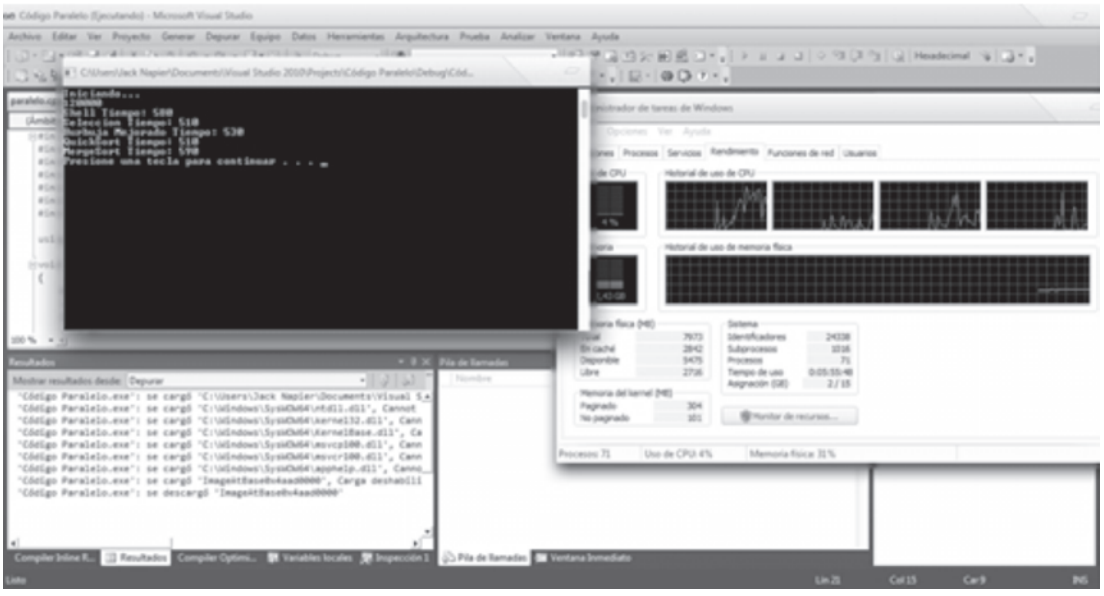
Figura 7. Escenario control, consumo de la CPU y tiempos de proceso.



Fuente: propia.

Por otra parte, el código paralelizado o con instrucciones que incorporan la programación en paralelo no solo bajó este consumo de CPU al 4%, sino que evidenciaron el uso de los diferentes procesadores para resolver el problema en el menor tiempo posible (ver figura 8).

Figura 8. Escenario experimental, uso de la CPU y tiempos de proceso.



Fuente: propia.

Cabe resaltar que el análisis demostró que el mayor consumo de tiempo estaba presente en las instrucciones relacionadas con la gestión del archivo tanto de lectura como de escritura del mismo, y por otra parte de los ciclos repetitivos o de iteracción como el "for o para" utilizados para el ordenamiento.

La herramienta supone unos nuevos recursos informáticos para los programadores, estudiantes para desarrollar aplicaciones de alto desempeño en la gestión de problemas con cálculos complejos para ser más competitivos con sus productos y mejorar sus competencias específicas.

El impacto del estudio, trae consigo la importancia de formar a estudiantes y desarrolladores en la aplicación de técnicas que generen algoritmos eficientes que gestionen los recursos hardware en su totalidad sacando el mayor provecho posible, en respuesta que la tendencia de los equipos de cómputo no está en sacar al mercado procesadores más rápidos, sino procesadores con varios núcleos que permiten ser más eficientes al manejar de forma paralela los procesos.

Finalmente, teniendo en cuenta la vida útil que tiene un equipo de cómputo, se puede afirmar que antes de desechar un computador ya sea portátil, desktop, tableta, etc., que generalmente siguen en funcionamiento y antes de adquirir un nuevo equipo con mayor prestación de servicios o por deseo de renovar; se le diera un uso óptimo de los recursos que brinda mientras presta su vida útil.

Agradecimientos

De manera muy especial agradezco a Michael Smith que en representación de la corporación Intel (empresa desarrolladores de microprocesadores a nivel mundial) apoyo con recursos software y documentación la investigación, de igual manera a la Universitaria de Investigación y Desarrollo (UDI) que me brindó los espacios físicos, tiempos y recursos hardware necesarios para desarrollar del estudio.

Referencias

Akhter, S., & Roberts, J. (2006). Multi-Core Programming. United State of América: Intel Press.

Blair-Chappell, S., & Stokes, A. (2012). Parallel Programming with Intel Parallel Studio XE. United States of American: John Wiley & Sons.

Deitel, H., & Merrel, D. D. (2004). Cómo programar en C/C ++ y Java. México: Pearson.

Foster, I. (1995). Designing and Building Parallel Programs . Chicago: Addison Wesley.

Intel. (28 de 11 de 2014). Intel® Software Academic Program. Obtenido de Intel® Software Academic Program: https://software.intel.com/en-us/courseware/hpc. Visitado el 30 de noviembre del 2014.

Padua, D. (2011). Encyclopedia of Parallel Computing. United State of America: Springer

Science & Business Media.

Reinders, J. (2005). VTune Performance Analyzer Essentials: Measurement and Tuning Techniques for Software Developers. United States of American: Intel Press.

Supalov, A., Semin, A., Dahnken, C., & Klemm, M. (2014). Optimizing HPC Applications with Intel Cluster Tools. United States of American: Apress.