	UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS ***FACULTAD DE* INGENIERIA** SYLLABUS **PROYECTO CURRICULAR:**	

NOMBRE DEL DOCENTE: ***Esteban Hernandez Barragan***	
ESPACIO ACADÉMICO (Asignatura): COMPUTACION PARALELA **Obligatorio ( ) : Básico ( ) Complementario ( )** **Electivo (X) : Intrínsecas () Extrínsecas (x)** **Área de conocimiento:**	**CÓDIGO:**
NUMERO DE ESTUDIANTES: 20	**GRUPO:** 1
NÚMERO DE CREDITOS: (Cuatro) (4)	
TIPO DE CURSO: TEÓRICO PRACTICO TEO-PRAC: X *Alternativas metodológicas:* *Clase Magistral (X), Seminario ( ), Seminario – Taller ( ), Taller ( ), Prácticas (X), Proyectos tutoriados ( ), Otro: _____*	
HORARIO:	

DIA	HORAS	SALON

I. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO (El Por Qué?)

La aparición de nuevas arquitecturas de procesadores (ARM, Gpus, FPGAs, SoC) ha cambiado la manera como entendemos el mundo de la programación, especialmente cuando estas nuevas arquitecturas se mezclan creando las así denominadas arquitecturas hetereogéneas de computación.

La programación paralela por tanto capacita al estudiante para desarrollar algoritmos eficientes en cuanto a distribución, utilización de datos y la manera como puede dividir un gran problemas en unidades pequeñas de cooperan y que pueden ser ejecutadas paralelamente para que el disminuir el tiempo total del ejecución.

Si bien los principios generales de la programación paralela pueden ser aplicados a cualquier arquitectura, las implementaciones de los mismos principios varian algunas veces de manera radical entre cada una de ellas, requiriendo conocer aspectos relevantes del hardware, del modo de transferencia y de los algoritmos y patrones que pueden ser utilizados.

Entendiendo que la programación paralela, requiere lenguajes que sean capaces de expresar los diversos niveles y formas de paralelismo, se hace necesario dominar por tanto los lenguajes y frameworks más populares al igual que los nuevos lenguajes emergentes, disponiendo asì de las capacidades y habilidades necesarias para desarrollar programas eficientes que aprovechen el poder de cómputo en el ambiente multicore/manycore y aceleradores.

***Conocimientos previos**:* Para el buen desarrollo del curso, se considera necesario e indispensable que el estudiante tenga dominio de los temas programación en C y C++, Algebra Lineal y tener conocimiento avanzado en cálculo. Xxxx

II. PROGRAMACION DEL CONTENIDO (El Qué? Enseñar)

OBJETIVO GENERAL

Conocer los elementos teóricos y prácticos básicos para que el futuro ingeniero sea capaz de desarrollar software en arquitecturas paralelas utilizando los lenguajes de programación, frameworks y librerías más populares que son utilizadas en el ambito cientifico actual.

OBJETIVOS ESPECÍFICOS

Se recomienda plantear un objetivo por cada unidad didáctica de trabajo.

1- Conocer los diferentes lenguajes utilizados en programación paralela C,C++, Fortran, Scala, Python
2- Dominar los principales framewors y librerias que son utilizadas para el desarrollo de programas paralelos (OpenMP, OpenACC, MPI, CUDA, X10 y otros)
3- Conocer los patrones de desarrollo de programas paralelos, aplicando los lenguajes, librerias y frameworks, que son utilizados en la actualidad.
4- Adquirir destrezas en la configuraciòn de los ambientes de progrogramación para el uso de Gpus y Aceleradores vectoriales, al igual que entornos multicore.

COMPETENCIAS DE FORMACIÓN:

Generales: Se pretende con la asignatura que el estudiante desarrolle competencias genéricas instrumentales, tales como la de resolución de problemas y la de capacidad de análisis y síntesis, entendidas como la capacidad de identificar, analizar, definir y sintetizar los elementos significativos que constituyen un problema para resolverlo con criterio y de forma efectiva.

Específicas: Al finalizar la asignatura el estudiante deberá ser capaz de:

1. Elegir un lenguaje de programación para el desarrollo de un programa paralelo de acuerdo a los objetivos y restricciones que se presenten.
2. Utilizar y configurar de manera adecuada un entorno de programación paralela, para el desarrollo de software en arquitecturas hetereogéneas.
3. Adquirir destreza en utilizar herramientas que permiten hacer seguimiento, profiling y debugging a software paralelo desarrollo.

PROGRAMA SINTÉTICO: (En cada unidad debe proponerse una pregunta guía y la competencia que se pretende desarrollar).

La asignatura se divide en nueve (8) unidades didácticas:

1. INTRODUCCION A LA A LA PROGRAMACIÓN EN PARALELO
2. PATRONES PARA EL DESARROLLO DE SOFTWARE PARALELO
3. PROGRAMACION MULTIHILO CON PTHREADS
4. PROGRAMACION MEMORIA COMPARTIDA USANDO OPENMP, OPENACC
5. PROGRAMACION MEMORIA DISTRIBUIDA CON MPI
6. PROGRAMACION SIMD/SIMP con CUDA, OpenCL, CilkPlus, Chapel
7. PROGRAMACION VECTORIAL
8. APLICIONES PRACTICAS

III. ESTRATEGIAS (El Cómo?)

Metodología Pedagógica y Didáctica:

El contenido de la asignatura es bastante exigente y riguroso ya que realiza una triple labor en la que se integra el diseño de tramas conceptuales evolutivas, de tal manera que permitan detectar y ayuden a recordar los conocimientos previos del estudiante; el establecimiento de nuevos conocimientos y la resolución de problemas. En consecuencia, el ritmo que se sigue en la asignatura requiere por parte del estudiante un esfuerzo importante, en particular si éste necesita afianzar conocimientos previos que no tiene suficientemente asentados.

Las clases consisten en la exposición de los conceptos teóricos, por parte del profesor, y en el desarrollo y resolución de problemas, tanto por parte del profesor como guiados por él pero realizados por los estudiantes individualmente o en grupo. En algunos ejercicios el estudiante tendrá la oportunidad de explicar al conjunto de la clase la forma en la que ha resuelto el problema.

En las clases se intenta que el estudiante capte las ideas que se presentan, aprecie su importancia y sea capaz de trasladarlas al planteamiento y resolución de problemas. En ese sentido, la clase expositiva por parte del profesor supone aproximadamente XX% del tiempo en el aula, mientras que el restante XX% se dedica a la realización de ejercicios y a la resolución de problemas.

Aunque el mayor o menor aprovechamiento del tiempo en el aula condiciona la necesidad de estudio y trabajo autónomo de parte del estudiante fuera de ella, en términos generales se puede decir que, si los conocimientos de base descritos en los conocimientos previos son consistentes y el seguimiento de la asignatura es uniforme durante el semestre, por cada hora de clase en el aula el alumno requiere otra hora de estudio fuera de la misma.

		Horas		Horas profesor/semana	Horas Estudiante/semana	Total Horas Estudiante/semestre	Créditos
Tipo de Curso	TD	TC	TA	(TD + TC)	(TD + TC +TA)	X 16 semanas	
ASIGNATURA	2	2	4	4	6	182	4

Trabajo Presencial Directo (TD): trabajo de aula con plenaria de todos los estudiantes.

Trabajo Mediado_Cooperativo (TC): Trabajo de tutoría del docente a pequeños grupos o de forma individual a los estudiantes.

Trabajo Autónomo (TA): Trabajo del estudiante sin presencia del docente, que se puede realizar en distintas instancias: en grupos de trabajo o en forma individual, en casa o en biblioteca, laboratorio, etc.)

IV. RECURSOS (Con Qué?)

Medios y Ayudas: Espacio físico (aula); Recurso docente; Recursos bibliográficos (especializados); recursos tecnológicos (equipos)

BIBLIOGRAFÍA

TEXTOS GUÍAS

ANON, *High Performance Computing*,

DONGARRA, J., FOSTER, I., FOX, G. AND GROPP, W., 2003. *Sourcebook of parallel computing*,

GROUP, F., 2008. *Handbook of Parallel Computing*,

OLUKOTUN, K., HAMMOND, L. AND LAUDON, J., 2007. *Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency*,

PASSING, M. AND FORUM, I., 2003. MPI-2 : Extensions to the Message-Passing Interface. , (21111).

RAUBER, T. AND RÜNGER, G., 2010. *Parallel programming: For multicore and cluster systems*,

YANG, L.T. AND GUO, M., 2005. *High-Performance Computing* L. T. Yang & M. Guo, eds., Hoboken, NJ, USA: John Wiley & Sons, Inc.

GREGORY, K. AND MILLER, A., 2012. C++ AMP: Accelerated Massive Parallelism with Microsoft® Visual C++®.

TEXTOS COMPLEMENTARIOS

CHANDRA, R. (2001). *Parallel programming in OpenMP*. San Francisco, Calif, Morgan Kaufmann Publishers. http://www.books24x7.com/marc.asp?bookid=7063.

GROPP, W., LUSK, E., & SKJELLUM, A. (1999). *Using MPI portable parallel programming with the message-passing interface*. Cambridge, Mass, MIT Press. http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=18119.

Wagener, Jerrold L. "High Performance Fortran." *Computer Standards and Interfaces* 18 (1996): 371–377. Web.

Saraswat, V A, V Sarkar, and C Von Praun. "X10: Concurrent Programming for Modern Architectures." *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP*. N.p., 2007. 271. Web.

Odersky, Martin, Lex Spoon, and Bill Venners. "Programming in Scala." *Theoretical Computer Science* 410 (2008): 202–220. Web.

Cook, Shane. *CUDA Programming*. N.p., 2013. Web.

Nickolls, John et al. "Scalable Parallel Programming with CUDA." *Queue* 2008: 40. Web.

Nvidia Corporation. "OpenCL Programming Guide for the CUDA Architecture." *NVIDIA* 41 (2010): 371–379. Web.

REVISTAS

http://www.journals.elsevier.com/parallel-computing/ Parallel Computing, Systems & Applications
http://www.springer.com/computer/theoretical+computer+science/journal/10766 International Journal of Parallel Programming
http://hpc.sagepub.com/ International Journal of High Performance Computing Applications

DIRECCIONES DE INTERNET

https://computing.llnl.gov/tutorials/parallel_comp/
http://groups.csail.mit.edu/cag/ps3/lectures.shtml
http://courses.engr.illinois.edu/ece408/lectures.html
https://www.cac.cornell.edu/ranger/Cintro/mathop.aspx
http://www.cs.tau.ac.il/~shanir/Workshop%20on%20Directions%20in%20Multicore%20Programming%20Education.htm

V. ORGANIZACIÓN / TIEMPOS (De Qué Forma?)

Espacios, Tiempos, Agrupamientos:

El espacio académico contempla horas de trabajo directo, trabajo colaborativo y trabajo autónomo; las temáticas se desarrollarán por unidades programadas por semana; el trabajo directo se realizará a partir de exposiciones del docente, que permitan el planteamiento de problemas y su posible solución práctica. La práctica en laboratorio (trabajo colaborativo), será abordada grupalmente y desarrollará temáticas y/o el tratamiento de problemas previamente establecidos, con el acompañamiento del docente. El estudiante desarrollará el trabajo autónomo de acuerdo con criterios previamente concertados en términos de contenidos temáticos y problemas planteados; las revisiones de avances y solución a preguntas se realizarán vía Internet.

VI. EVALUACIÓN (Qué, Cuándo, Cómo?)

Es importante tener en cuenta las diferencias entre evaluar y calificar. El primero es un proceso cualitativo y el segundo un estado terminal cuantitativo que se obtiene producto de la evaluación. Para la obtención de la información necesaria para los procesos de evaluación se requiere diseñar distintos formatos específicos de autoevaluación, coevaluación y heteroevaluación.

	TIPO DE EVALUACIÓN	FECHA	PORCENTAJE
PRIMER ANOTA			
SEGUNDA NOTA			
EXAM. FINAL			**20%**

ASPECTOS A EVALUAR DEL CURSO

1. Evaluación del desempeño docente

2. Evaluación de los aprendizajes de los estudiantes en sus dimensiones: individual/grupo, teórica/práctica, oral/escrita.

3. Autoevaluación:

4. Coevaluación del curso: de forma oral entre estudiantes y docente.

DATOS DEL DOCENTE

NOMBRE :

PREGRADO : INGENIERO EN REDES DE COMPUTACION

POSTGRADO : MASTER EN SOFTWARE LIBRE, ESPECIALIZACION EN MATEMATICA APLICADA, ESPECIALIZACION EN CONSTRUCCION DE SOFTWARE PARA REDES

E mail:

ASESORIAS: FIRMA DE ESTUDIANTES

NOMBRE	**FIRMA**	**CÓDIGO**	**FECHA**
1.			
2.			
3.			

FIRMA DEL DOCENTE

_____

FECHA DE ENTREGA: _____