Asignatura: Computación a Gran Escala (COMP)
Código: 32497
Institución: Escuela Politécnica Superior
Grado: Master en Ingeniería Informática
Nivel: Master
Tipo: Troncal
ECTS: 6

GUIA DOCENTE: Computación a Gran Escala

Año académico: 2017-2018

Programa: Master en Ingeniería Informática
Centro: Escuela Politécnica Superior
Universidad: Universidad Autónoma de Madrid

Última modificación: 2017/06/13
Status: Aprobado 2013/05/29

1. ASIGNATURA (ID)

Cálculo intensivo y manejo de datos a gran escala

1.1. Programa

Master en Ingeniería Informática

1.2. Código de curso

32416

1.3. Áreas del curso

Ciencias de la Computación e Inteligencia Artificial

1.4. Tipo de asignatura

Obligatoria [itinerario: todos los itinerarios]

1.5. Semestral

Primer trimestre

1.6. Créditos

6 ETCS

1.7. Lenguaje

Las clases serán en español, aunque algunos seminarios podrían ser en inglés.

1.8. Recomendaciones / Asignaturas relacionadas

Conocimientos de C, Bases de Datos, probabilidad y estadística a nivel básico son necesarios para seguir el curso

Asignaturas relacionadas:

- Procesamiento de información temporal
- Aprendizaje Automático: teoría y aplicaciones
- Métodos bayesianos aplicados

- Aceleración de algoritmos en sistemas heterogéneos
- Procesamiento de señales biomédicas y sus aplicaciones
- Procesamiento de audio y voz para biometría y seguridad
- Técnicas de análisis de secuencias vídeo para videovigilancia

1.9. Profesores

Añadir @uam.es a todas las direcciones de correo posteriores

Profesores:

Dr. Carlos Santa Cruz Fernández (Coordinador)
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Office: B-343
Tel.: +34 914972337
e-mail: carlos.santacruz
Web: http://www.eps.uam.es/~santacru

Dr. Miguel Ángel García García
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Office: C-242
Tel.: +34 914976208
e-mail: miguelangel.garcia
Web: http://www.eps.uam.es/~mgarcia/

Dr. Estrella Pulido Cañabate
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Office: B-413
Tel.: +34 914972289
e-mail: estrella.pulido
Web: http://www.eps.uam.es/~epulido/

Dr. Gonzalo Martínez Muñoz
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Office: B-422
Tel.: +34 914977528
e-mail: gonzalo.martinez
Web: http://www.eps.uam.es/~gonzalo/

1.10. Objetivos de la asignatura

Esta asignatura está divida en tres partes. La primera parte se centra en una introducción a las técnicas de programación eficiente de las arquitecturas paralelas de memoria compartida, incluyendo los procesadores multi-núcleo y los multiprocesadores fuertemente acoplados, seguida de un resumen de los entornos de programación eficiente de las arquitecturas paralelas de memoria distribuida, incluyendo la computación en clúster, grid y en la nube. En concreto, se describen herramientas de análisis de rendimiento (*profilers*), se identifican los principales factores que afectan a la eficiencia de las arquitecturas paralelas, y se estudian técnicas de paralelización de bucles en arquitecturas de memoria compartida mediante OpenMP. Finalmente, se realiza una introducción y análisis comparativo de los entornos de programación distribuida mediante OpenMPI, Hadoop y Spark.

En esta asignatura se presenta además una introducción al sistema Spark de procesamiento distribuido sobre un cluster de nodos. Este sistema además de permitir la alta disponibilidad permite el procesamiento de grandes volúmenes de datos en un cluster de máquinas

Las competencias básicas que el estudiante adquiere en esta asignatura son:

- C1. Capacidad para resolver problemas utilizando el paradigma de computación en paralelo de Apache Spark.
- C2. Capacidad para crear soluciones en Apache Spark eficientes.
- C3. Capacidad para crear soluciones en Apache Spark que utilicen: datos estructurados, métodos de aprendizaje automático y/o fuentes de datos de streaming.

La competencia de tecnología específica que el estudiante adquiere en esta asignatura es:

- TI7 - Capacidad para comprender y poder aplicar conocimientos avanzados de computación de altas prestaciones y métodos numéricos o computacionales a problemas de ingeniería.

Las cualificaciones ubicadas en el nivel de competencias transversales que el estudiante adquirirá en esta asignatura son:

- TR1 Capacidad para actualizar conocimientos habilidades y destrezas de forma autónoma, realizando un análisis crítico, análisis y síntesis de ideas nuevas y complejas abarcando niveles más integradores y pluridisciplinares.

- TR4 Capacidad para transmitir de un modo claro y sin ambigüedades a un público especializado o no, resultados procedentes de la investigación científica y tecnológica o del ámbito de la innovación mas avanzada, así como los fundamentos mas relevantes sobre los que se sustentan. Capacidad para argumentar y justificar lógicamente dichas decisiones de un modo

claro y sin ambigüedades, sin dejar de considerar puntos de vista alternativos o complementarios.

Al final de cada unidad el estudiante debe ser capaz de:

OBJETIVOS ESPECÍFICOS DE CADA UNIDAD	
PARTE I	
UNIDAD 1.- Introducción	
1.1.	Conocer las diferentes familias de arquitecturas paralelas y escoger las que mejor se ajusten a un ámbito de aplicación específico.
1.2.	Medir el rendimiento de algoritmos paralelos en términos de *speedup* y eficiencia.
1.3.	Entender los conceptos básicos de programación paralela, incluyendo tareas, procesos y mecanismos de sincronización.
1.4.	Utilizar herramientas de análisis de rendimiento (*profilers*) para analizar la eficiencia de algoritmos secuenciales e identificar porciones susceptibles de ser aceleradas mediante paralelización.
1.5.	Implementar algoritmos paralelos simples sobre arquitecturas paralelas de memoria compartida usando C y OpenMP.
UNIDAD 2.- Paralelización de bucles en arquitecturas paralelas de memoria compartida	
2.1.	Entender los diferentes tipos de bucles paralelos y escoger los que mejor se ajustan a un problema específico.
2.2.	Describir las dependencias entre iteraciones de un conjunto de bucles secuenciales mediante un grafo de dependencias.
2.3.	Paralelizar un conjunto de bucles secuenciales a partir de su correspondiente grafo de dependencias.
2.4.	Aplicar transformaciones de código de cara a optimizar la paralelización de un conjunto de bucles secuenciales.
UNIDAD 3.- Paralelización en arquitecturas paralelas de memoria distribuida	
3.1.	Conocer las diferentes familias de arquitecturas paralelas de memoria distribuida.
3.2.	Conocer los principales entornos de programación de las arquitecturas paralelas de memoria distribuida y saber escoger los que mejor se ajustan a un ámbito de aplicación específico.
PARTE II: Fundamentos de Spark	
4.1.	Conceptos fundamentales de arquitectura y organización
4.2.	Entender el flujo de un programa
4.3.	Definición de entradas y salidas
PARTE III: Spark SQL	
5.1.	Entender los conceptos de DataFrames
5.2.	Aprender a manejar la API de DataFrames en SparkSQL
PARTE IV	
UNIDAD 6.- Fundamentos de computación	
6.1.	Entender la notación en como flotante de los procesadores actuales
6.2.	Entender que el error de redondeo se debe a que la aritmética entre números no es

	exacta.
6.3.	Entender que el Erro debido a los algoritmos son independientes del hardware
6.4.	Entender que los errores pueden ser sucesivamente magnificados debido a los algoritmos
UNIDAD 7.- Solución de sistemas de ecuaciones lineales	
7.1.	Escribir un problema de regresión lineal múltiple en notación matricial
7.2.	Resolver algunos ejemplos prácticos

1.11. Contenido del curso

PARTE 1

1. **Introducción**
   - 1.1. Arquitecturas paralelas: motivación.
   - 1.2. Arquitecturas paralelas de memoria compartida.
     - 1.2.1. Procesadores multi-core y multiprocesadores fuertemente acoplados.
     - 1.2.2. OpenMP.
   - 1.3. Arquitecturas paralelas de memoria distribuida: multiprocesadores débilmente acoplados.
   - 1.4. Computación en malla (*grid computing*) y en la nube (*cloud computing*).
   - 1.5. Conceptos básicos.
     - 1.5.1. Tareas y procesos.
     - 1.5.2. Semáforos y barreras.
     - 1.5.3. Herramientas de análisis de rendimiento (*profilers*).

2. **Paralelización de bucles en arquitecturas paralelas de memoria compartida.**
   - 2.1. Bucles paralelos.
   - 2.2. Planificación (*scheduling*) de bucles paralelos.
   - 2.3. Análisis de dependencias entre iteraciones.
     - 2.3.1. Dependencias verdaderas.
     - 2.3.2. Antidependencias.
     - 2.3.3. Dependencias de salida.
     - 2.3.4. Dependencias en bucles imbricados.
     - 2.3.5. Grafos de dependencias por niveles.
   - 2.4. Generación de código paralelo.
     - 2.4.1. Componentes fuertemente conexas.
     - 2.4.2. Condensación acíclica.
     - 2.4.3. Arcos libres de barreras.
     - 2.4.4. Generación de *clusters* y segmentos.
     - 2.4.5. Generación de código.
       - 2.4.5.1. Generación de código para segmentos serie.
       - 2.4.5.2. Generación de código para segmentos paralelos.
     - 2.4.6. Problemas de ejemplo.
   - 2.5. Transformaciones para soportar paralelización.
     - 2.5.1. Normalización de bucles.
     - 2.5.2. Substitución escalar.
     - 2.5.3. Expansión escalar.
     - 2.5.4. Copiado de variables.
     - 2.5.5. Intercambio de bucles.

3. **Paralelización en arquitecturas paralelas de memoria distribuida**
   - 3.1. Fundamentos de computación distribuida

PARTE II

4. **Fundamentos de Spark**
   - 4.1. Introducción: arquitectura y organización
   - 4.2. Datos en Spark: Resilient Distributed Datasets (RDDs)
   - 4.3. Flujo de un programa Spark
   - 4.4. Entrada y salida de datos
   - 4.5. Transformaciones
   - 4.6. Persistencia
   - 4.7. Acciones
   - 4.8. Variables compartidas: broadcast y acumuladores

PARTE III

5. **Spark SQL**
   - 5.1. Introducción a DataFrame
   - 5.2. Fuentes de datos: Json, JDBC/ODBC, Parquet, etc.
   - 5.3. API de DataFrames
   - 5.4. UDFs y Window Functions

PARTE IV

6. **Introducción**
   - 6.1. Representación en coma flotante
   - 6.2. Error de redondeo
   - 6.3. Error de truncado
   - 6.4. Estabilidad

7. Solución de sistemas de ecuaciones lineales
   - 7.1. Descomposición LU
   - 7.2. Descomposición Cholesky
   - 7.3. Regresión lineal multiple
   - 7.4. Aplicaciones prácticas

1.12. Bibliografía

1. "Parallel Computer Architecture: A Hardware/Software Approach" D. Culler, J.P. Singh, A. Gupta. *Ed. Morgan Kaufmann, 1998.*
2. "Supercompilers for parallel and vector computers" H. Zima, B. Chapman *Ed. ACM Press, 1991.*
3. "Optimizing Compilers for Modern Architectures: A Dependence-based Approach". Allen, K. Kennedy *Ed. Morgan Kaufmann, 2001.*
4. "Computer Architecture: A Quantitative Approach" (5a. ed.) J.L. Hennessy, D.A. Patterson *Ed. Morgan Kaufmann, 2011.*
5. "Advanced Computer Architecture: Parallelism, Scalability, Programmability" K. Hwang. *Ed. McGraw-Hill, 1992.*
6. "Numerical Recipes in C: The Art of Scientific Computing", W. H. Teulosky, A. A. Vetterling, W. T. Flannery, B. P., Cambridge University Press, 1992
7. "Numerical Mathematics. Theory and Computer Applications" C. E. Froberg.. Addison-Wesley, Reading, Massachusetts, 1985.
8. "Scientific Computing: An Introductory Survey" M. T. Heath., 2nd. ed. McGraw-Hill, New York, 2001.
9. "Análisis numérico con aplicaciones" C. F. Gerald and P.O. Wheatley., 6a ed.Prentice Hall, México, 2000.
10. Spark Cookbook. Rishi Yadav. Packt Publishing. 2015.
11. Spark in action. Peter Zecevic, Marko Bonaci. Manning Publications. 2017.

1.13. Trabajos y evaluación

El curso consta de clases presenciales, tareas semanales, tareas de laboratorio y un examen final.

En el periodo ordinario de examen es necesario aprobar (>=5) el examen para pasar el curso. En el periodo extraordinario es solo necesario aprobar (>=5) el trabajo de investigación para aprobar el curso.

- En el periodo ordinario la evaluación constará de las siguientes partes:
  - 50 % Trabajos de laboratorio
  - 50 % Examen

  Las notas de las partes se guardarán para el periodo extraordinario

- En caso de suspender en el periodo ordinario, en el periodo extraordinario el alumno tiene la oportunidad de:
  - Presentar todas las practicas después de haberlas corregido de nuevo
  - Presentar un trabajo de investigación sobre un tema acordado con el profesor de la asignatura.

  La nota final será determinada por
  - 50 % Prácticas de laboratorio [solo si han sido presentadas de Nuevo]
  - 50 % Trabajo de investigación [solo si el trabajo se presenta]