

Título

Computación de Alta Performance: Modelos, Métodos y Medios.

Motivación

Las computación de alto rendimiento (HPC) es un campo interdisciplinario que tiene un impacto importante en muchas áreas de la ciencia, la tecnología, la medicina y el comercio. Como un campo interdisciplinario, HPC involucra talentos en las grandes áreas de arquitectura de hardware, el diseño, sistema de software, lenguajes de programación, herramientas, algoritmos paralelos y técnicas computacionales. Sólo a través de una presentación interdisciplinaria, los estudiantes pueden experimentar la interacción mutua y la necesidades de cada componente con los demás. Mediante ésto, se obtiene la determinación de los factores que influyen en el resultado global de la capacidad computacional. El objetivo del curso es producir una generación de técnicos para soporte y de expertos científicos en el desarrollo, funcionamiento y aplicación de la computación de alto desempeño.

Las áreas a ser cubiertas por el curso incluyen:

1. Introducción y visión general de HPC,
2. Aplicaciones a gran escala y métodos de algoritmos paralelos,
3. Tecnologías que hacen posible la HPC,
4. Sistema de nodo único, arquitecturas paralelas y clusters de PCs,
5. Métricas de performance, monitoreo, medida y benchmarking,
6. Métodos de programación y herramientas para la creación de computación de capacidades y habilidades.

Programa

1. Introducción

El problema. Un estudio de los métodos actuales. Una breve historia de la supercomputación. Acerca del libro del curso.

Temas:

- ¿Por qué HPC y cuáles son las aplicaciones que motivan su desarrollo?
- ¿Qué es una supercomputadora, cuáles son sus principales elementos, qué lo hace ir rápido que lo hace ir lento?
- ¿Cuál es el papel del software del sistema, su organización, y sus desafíos?
- ¿Cómo se aplican los sistemas HPC de los problemas de usuario; cómo se programan y cuáles son los algoritmos paralelos empleados?
- ¿Cuáles son las métricas claves, cómo ellas se miden en el mundo real, y cómo se utilizan los benchmarks?
- Una breve historia de la supercomputación.
- Este curso: cómo está organizado y cómo hacer el mejor uso posible de la experiencia y los recursos.

2. Aplicaciones a Gran Escala.

Grandes problemas en la ciencia y la tecnología que se benefician de la HPC y qué tipo de recursos de tiempo/espacio requieren/requerirán para alcanzar sus objetivos.

Temas:

- Principales problemas que requieren de las supercomputadoras actuales y futuras.

- Recursos necesarios y los retos implícitos en dichas aplicaciones.
 - Ejemplos de códigos de aplicación específica que se utilizan hoy en día, y los lenguajes de programación utilizados para crearlos.
3. **Tecnologías que hacen posible HPC**
Una breve historia de la tecnología de los dispositivos. Tecnologías actuales, incluyendo: lógica, almacenamiento y comunicaciones.
Temas:
- Principales clases de tipos de componentes necesarias para implementar supercomputadoras.
 - Una cronología de las tecnologías utilizadas para construir estas clases de componentes en los últimos 60 años.
 - Hoja de ruta SIA ITSR de las tecnologías de semiconductores.
 - Tecnologías de red.
 - Chasis y soportes físicos.
 - Desafíos del consumo de energía, fiabilidad, tamaño y costo.
4. **Arquitectura de Nodo Único y su Rendimiento**
Estructuras de nodo único, incluyendo multi-core y una introducción a la utilización de OpenMP como una metodología de programación. Medidas de rendimiento y las cuestiones relacionadas con el rendimiento sostenido incluyendo niveles de paralelismo y las estructuras de memoria. Entrada/Salida externa como sequela para sistemas escalables más grandes.
Temas:
- Arquitectura monoprocesador para obtener buen rendimiento.
 - Multiprocesadores simétricos.
 - Programación OpenMP.
 - Sistemas de memoria caché y como evitar su latencia.
5. **Arquitectura de Computadoras Paralelas y Rendimiento**
La interacción entre la tecnología y los avances de Arquitectura de Computadoras. Las principales formas de arquitectura HPC: Vector, SIMD, MPP, DSM, los clusters. System area networks (SAN).
Temas:
- Procesadores vectoriales.
 - Arrays SIMD.
 - Memoria compartida distribuida.
 - MPP.
 - Clusters a partir de PCs comunes.
 - Arquitectura de red y protocolo.
6. **Clusters de PCs: Un ejemplo de un sistema HPC**
Linux Clusters en detalle, exponiendo todos los aspectos de los componentes, la organización, el sistema de software, el funcionamiento, y las capacidades medidas.
Temas:
- Clusters de PCs, Beowulf y NOW.
 - Ejemplo específico.
 - Instalación y puesta en marcha de Linux en un entorno distribuido.
 - Instalación y puesta en marcha de Condor.
 - Instalación y puesta en marcha de MPI.
 - Experimentos con códigos y benchmarks enlatados.
7. **Benchmarking**
Benchmarks standard, métricas y técnicas de medición.
Temas:
- Linpack.
 - NPB.
 - SPEC.
 - Desafío HPC.
 - SPIO.
 - Métricas para cada benchmark.
8. **Throughput Computing y Condor**

Métodos para utilizar Condor a fin de ejecutar trabajos de manera simultánea.

9. **Programación MPI**

Modelo de programación básico, llamadas y herramientas.

Temas:

- Modelo de computación Communicating Sequential Processes (CSP).
- Framework básico para el funcionamiento de MPI.
- Constructores de pasaje de mensajes.
- Operadores de reducción.
- I/O Paralelo.

10. **Monitoreo del Desempeño, Métricas, y Mediciones**

Herramientas y métodos utilizados para determinar lo que está sucediendo, cuan bien está rindiendo el sistema y la aplicación, y donde están los cuellos de botella.

Temas:

- Métricas del tiempo de respuesta, rendimiento, ancho de banda, latencia, retraso.
- Instrumentos y contadores dentro del Procesador.
- PAPI.

11. **Algoritmos de Núcleo Paralelo y Diseño de Aplicaciones**

Algunos de los algoritmos de núcleo básicos utilizados para una amplia gama de aplicaciones. Cómo diseñar/crear un código paralelo incluidas las cuestiones de distribución de las estructuras de datos, enfocando particularmente su escala, y cómo estos se relacionan con los algoritmos y métodos de programación.

Temas:

- Linear solvers.
- N-body Barnes-Hut.
- Particle in Cell (PIC).
- FFT.
- Matrices Ralas.

12. **Entornos de Programación de Dominio Específico**

Paquetes y bibliotecas que nos ayudan en la construcción de aplicaciones paralelas.

13. **Visualización**

Como comprender el significado de los resultados. Paquetes principales. Que necesitan para trabajar eficazmente.

Temas:

- Modos de presentación de datos.
- Herramientas de visualización.
- Visualización interactiva, cambios de rumbo en tiempo de ejecución.

14. **Software de Sistema**

Sistema Operativos y middleware. Schedulers como Maui y PBS.

Temas:

- Linux para servicios de procesamiento paralelo.
- PBS.
- Maui.
- LSF.
- ROCKS.

15. **I/O Paralelo**

Saliendo al mundo real. Como lograr persistencia en el almacenamiento.

Desplazamiento masivo de datos entre sistemas. Interfaz de usuario interactiva en tiempo real.

Temas:

- Checkpoint & Restart.
- Sistemas de archivos paralelos.
 - PVFS.
 - Lustre.
 - Panasas.
- Acceso a la grilla para datos compartidos remotos.

- Medición de la performance de I/O.
16. **Más allá de los Fundamentos**
 Se menciona brevemente todos los temas avanzados que no están cubiertos en este curso con una bibliografía anotada y URLs a fuentes confiables. Se tratará de articular distintos puntos de vista, y mostrar donde las incógnitas están mostrando las oportunidades de investigación.
 Temas:
- Constructores MPI-2.
 - Arquitectura Avanzada de Computadoras.
 - FPGA.
 - Streaming, Merimac.
 - Modelos de programación alternativa.
 - CAF.
 - UPC.
 - Otras formas de explotar el paralelismo.
17. **Hacia el futuro**
 Los desafíos de HPC para el hardware y el software en el corto plazo y algunas estrategias de investigación para direcciones futuras. Se hablará de tecnologías alternativas de dispositivos, arquitecturas, programación y modelos de ejecución.
 Temas:
- Los desafíos que deben superarse para sostener la oportunidad de la Ley de Moore.
 - Tecnologías alternativas que se están estudiando, incluyendo nanoescala.
 - Arquitecturas de computadora alternativas.
 - Más allá de la Ley de Moore.
 - La computación cuántica y otros paradigmas alternativos.
 - Modelos de programación de muy alto nivel.
 - Sistemas auto reparables.

Correlatividades

Sistemas Operativos y Redes de Computadoras.

Bibliografía

El curso está basado en el libro de próxima aparición:

- Thomas Sterling. "**High Performance Computing: Concepts, Methods and Means**". Book in progress.

La siguiente bibliografía es solo una selección de los títulos más importantes que referencia el libro del curso:

- J. Hennessy, D. Patterson, "Computer Architecture A Quantitative Approach", 3rd Edition, Morgan Kaufmann, 2003.
- T. Sterling, "How to Build a Hypercomputer", Scientific American, 285(1), pp. 38-45, July 2001.
- T. Sterling, "The Scientific Workstation of the Future May Be a Pile of PCs", Commun. ACM, 39(9), pp. 11-12, 1996.
- T. Sterling, "Continuum computer architecture for exaflops computation", Commun. ACM, 44(3), pp. 78-80, 2001.

- T. Sterling, "An Introduction to PC Clusters for High Performance Computing", *The International Journal of High Performance Computing Applications*, 15(2), pp. 92-101, Summer 2001.
- T. Sterling and Hans P. Zima, "Gilgamesh: a multithreaded processor-in-memory architecture for petaflops computing", pp. 1-23.
- T. Sterling, "Challenges to Evaluating Petaflops Systems", *QEST*, pp. 166-167, IEEE Computer Society, 2005.
- R. Kendall, M. Sosonkina, W. Gropp, R. Numrich and T. Sterling, "Parallel Programming Models Applicable to Cluster Computing and Beyond", Chapter in *Numerical Solution of Partial Differential Equations on Parallel Computers*, Springer, Fornebu, Norway, 2006.
- T. Sterling and M. Brodowicz. "Continuum Computer Architecture for Nano-Scale and Ultr-High Clock Rate Technologies", Chapter in *Innovative Architecture for Future Generation High-Performance Processors and Systems*, The Institute of Electrical and Electronics Engineers, Inc., Danvers, MA, 2005.
- W. Hargrove, F. Hoffman and T. Sterling, "The do-it-yourself supercomputer", *Scientific American*, 285(2), pp. 72-79, August 2001.
- T. Sterling, "Beowulf Cluster Computing with Linux". MIT Press, Cambridge, MA, 2001.
- Raj Jain, "The Art of Computer Systems Performance Analysis", Wiley Professional Computing, 1991.
- J. Dongarra, P. Luszczek and A. Petitet, "The LINPACK Benchmark: past, present and future", *Concurrency Computat.: Pract. Exper.* 2003; 15:803-820.
- David Bailey, "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers", *Supercomputing Review*, 4(8), pp. 54-55, August 1991.
- D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience", *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.
- T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor - A Distributed Job Scheduler", in Thomas Sterling, editor, *Beowulf Cluster Computing with Linux*, The MIT Press, 2002.
- C.A.R. Hoare, "Communicating Sequential Processes", *Comm. ACM*, 21(8), pp. 666-677, August 1978.
- Argonne National Laboratory, Mathematics and Computer Science Division, "The Message Passing Interface (MPI) standard", <http://www-unix.mcs.anl.gov/mmpi/>.
- W. Gropp, "Tutorial on MPI: The Message-Passing Interface", Mathematics and Computer Science Division, Argonne National Laboratory.
- Lawrence Livermore National Laboratory, High Performance Computing Division, "Message Passing Interface (MPI) Tutorial", <https://computing.llnl.gov/tutorials/mppi/>.
- P. Pacheco, "A User's Guide to MPI", Department of Mathematics, University of San Francisco, 1998.
- Lawrence Livermore National Laboratory, High Performance Computing Division, "OpenMP Tutorial", <https://computing.llnl.gov/tutorials/openMP/>.
- G. Andrews, "Foundations of Multithreaded, Parallel, and Distributed Programming", Addison Wesley, 2000.
- S. Browne and J. Dongarra and N. Garner and P. Mucci, "A Scalable Cross-Platform Infrastructure for Application Performance Tuning Using Hardware Counters", August 2000.
- P. Mucci, "Performance Analysis Tools and PAPI", HP Labs, Palo Alto, CA, February, 2005.
- W. H. Press and others, "Numerical Recipes in C (Second Edition)", Cambridge University Press, 1992.
- T. Cormen, C. Leiserson and R. Rivest, "Introduction to Algorithms", MIT Press, 1990.
- A. Silberschatz, P. B. Galvin and G. Gagne, "Operating System Concepts", 6th edition, Wiley, 2004.
- W. Stallings, "Operating Systems: Internals and Design Principles" (5th Edition).

- M. Bach, "The design of the UNIX operating system", Source Prentice-Hall Software Series, 1986.
- J. Weinberg, "Job Scheduling on Parallel Systems", Ph.D. Research Examination, University of California, San Diego, June 2006.
- T. Vedhuizen, "What is a Library?", Talk given at the Dagstuhl workshop Software Libraries: Design and Evaluation, Schloss Dagstuhl, Germany, March 9-11 2005.
- R. Heinzl, "Modern Application Design using Modern Programming Paradigms and a Library Centric Software Approach", OOPSLA 2006, Workshop on Library Centric Software Library-Centric Approach Design, Portland, Oregon, October 2006.

Modalidad de Dictado y Evaluación

Dictado de las clases a cargo del Prof. Thomas Sterling, via videoconferencia o web video streaming desde Louisiana State University (LSU).

Clases de consulta, evaluación de trabajos prácticos y exámenes a cargo del Lic. Nicolás Wolovick en Fa.M.A.F. .

Las clases comienzan el **12 de Enero de 2010** y finalizan el **6 de Mayo de 2010**.

Durante el mes de Enero la Fa.M.A.F. estará cerrada por lo que los alumnos deberán seguir las 5 primeras clases de Introducción a través de web video streaming provisto por LSU. La entrega de los dos primeros trabajos prácticos evaluables se postpondrá hasta el martes 2 de Febrero de 2010 donde se entregarán juntos.

La evaluación es en base a prácticos de papel y programación para resolver y entregar **de manera quincenal** durante todo el desarrollo del curso, además de un examen intermedio y uno final. El esquema de calificación es el siguiente:

- 30% examen intermedio,
- 35% examen final,
- 35% prácticos evaluables.

Carga Horaria

En total son 27 clases de 1.5hs de duración, y se estiman 2 horas de trabajo personal (estudio, programación, búsqueda bibliográfica, instalación de software) por cada hora de clase teórica, totalizando $27 \cdot 1.5 + 27 \cdot 1.5 \cdot 2 = 121.5$ hs.

Se darán clases de consulta en Fa.M.A.F. de 2 horas, una vez por semana.