SISTEMAS PARALELOS

Clase 1 - Introducción

Metodología

- Clases
  - Teorías: Jueves de 15 a 17 (aula 1-2)
  - Prácticas: Jueves de 17 a 19 (sala de PC Postgrado). Inicio: 22/03.

- Aprobación de la cursada: a partir de la entrega y aprobación de trabajos prácticos y sus respectivos coloquios (obligatorios).

- Aprobación del final:
  - Rendir examen final escrito en alguna mesa de examen.
  - Aprobar un parcial teórico al final de la cursada + desarrollo de un trabajo dado por la cátedra y su posterior coloquio (al inscribirse en la mesa de final, hasta mesa de Marzo de 2019).

Metodología

- Material y comunicación: Plataforma Ideas (ideas.info.unlp.edu.ar)
  - Buscar curso «Sistemas Paralelos» y solicitar inscripción

- Correo de la cátedra: sparalelos@lidi.info.unlp.edu.ar

Objetivos

- Plantear los fundamentos del procesamiento paralelo
- Caracterizar las arquitecturas de hardware
- Describir los modelos de programación
- Estudiar métricas de rendimiento
- Analizar y trabajar casos concretos de procesamiento paralelo, resolubles sobre distintas arquitecturas multiprocesador.

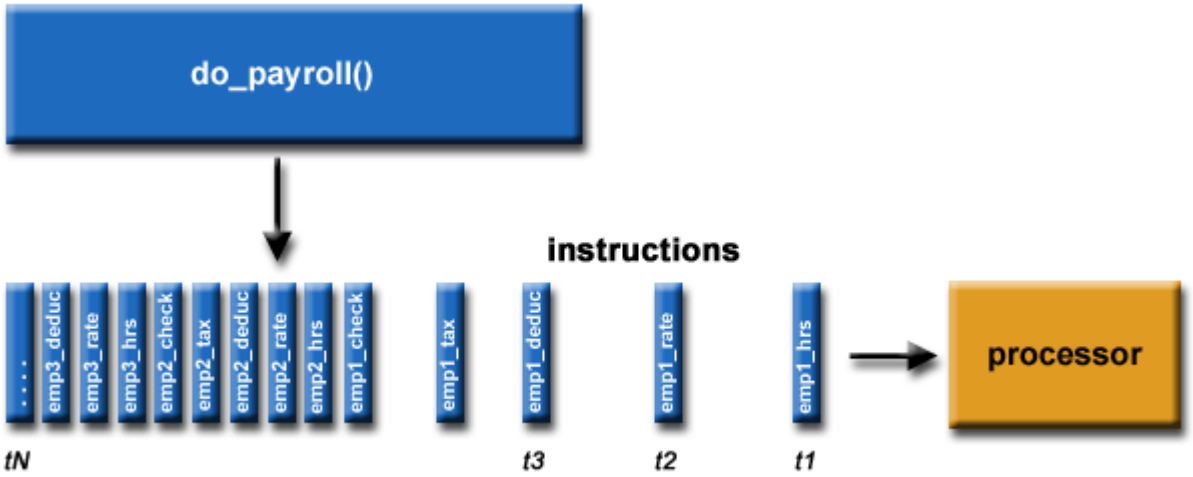
Bibliografía básica

- “Introduction to Parallel Computing”. Grama, Gupta, Karypis, Kumar. Addison Wesley (2003).
- “Introduction to High Performance Computing for Scientists and Engineers”. Georg Hager, Gerard Wellein. CRC Press (2011).
- “Parallel Programmng for Multicore and Cluster Systems”. Thomas Rauber, Gudulla Runger. Springer (2010).
- “An introduction to parallel programming”. Peter Pacheco. Elsevier (2011).
- “Parallel Programming”. Wilkinson, Allen. Prentice Hall 2005.

MOTIVACIÓN

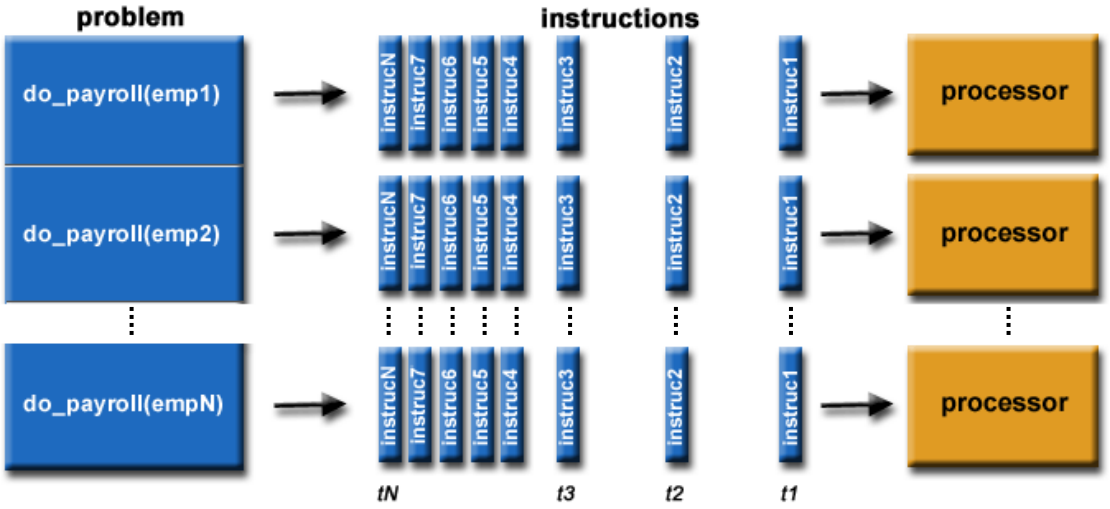
¿Qué es el procesamiento paralelo?

- Históricamente, el software ha sido desarrollado para ejecución secuencial
  - El problema se divide en conjuntos separados de instrucciones
  - Las instrucciones se ejecutan secuencialmente, una después de la otra.
  - La ejecución se realiza en un procesador único
  - Sólo una instrucción es ejecutada en un instante determinado.
- Ejemplo:



¿Qué es el procesamiento paralelo?

- En forma sencilla, el *procesamiento paralelo* es el uso de múltiples unidades de procesamiento para resolver un problema computacional.
  - El problema se divide en partes separadas que pueden ser resultas en forma concurrente.
  - Cada parte es luego dividida en una serie de instrucciones.
  - Las instrucciones de cada parte se ejecutan simultáneamente sobre diferentes procesadores.
  - Un mecanismo de control/coordinación global es necesario
- Ejemplo:



¿Por qué es importante el procesamiento paralelo?

- Resolver problemas más grandes o más complejos
  - Algunos problemas son tan grandes y/o complejos que resulta poco práctico (o imposible) de resolver empleando una única computadora
- Proveer concurrencia:
  - Una sola unidad de procesamiento puede hacer una única tarea en un instante dado. Múltiples unidades de procesamiento pueden hacer múltiples tareas al mismo tiempo.

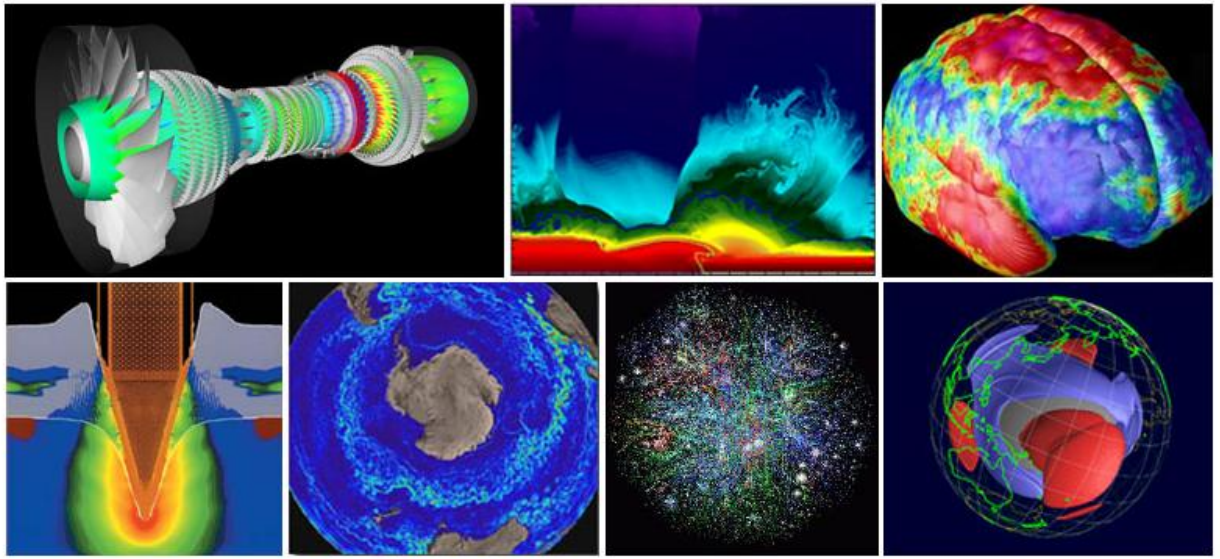
¿Por qué es importante el procesamiento paralelo?

- Ahorrar tiempo y/o dinero
  - En general, destinar más recursos a una tarea conlleva a completarla en menor tiempo, lo que a su vez puede ahorrar dinero.
  - Las computadoras paralelas pueden ser construidas a partir de componentes básicos (*commodities*).
- Hacer mejor uso de los recursos de hardware
  - Hoy todas las computadoras son paralelas con múltiples unidades de procesamiento.
  - El software paralelo se diseña específicamente para aprovechar el hardware subyacente.
  - En la mayoría de los casos, los programas secuenciales *desperdician* el poder de cómputo que ofrecen las computadoras actuales.

¿Qué problemas requieren procesamiento paralelo?

- Ciencia e Ingeniería
  - Históricamente, el procesamiento paralelo ha sido considerado como un recurso de lujo, y ha sido empleado para modelar problemas complejos de la ciencia y de la ingeniería.

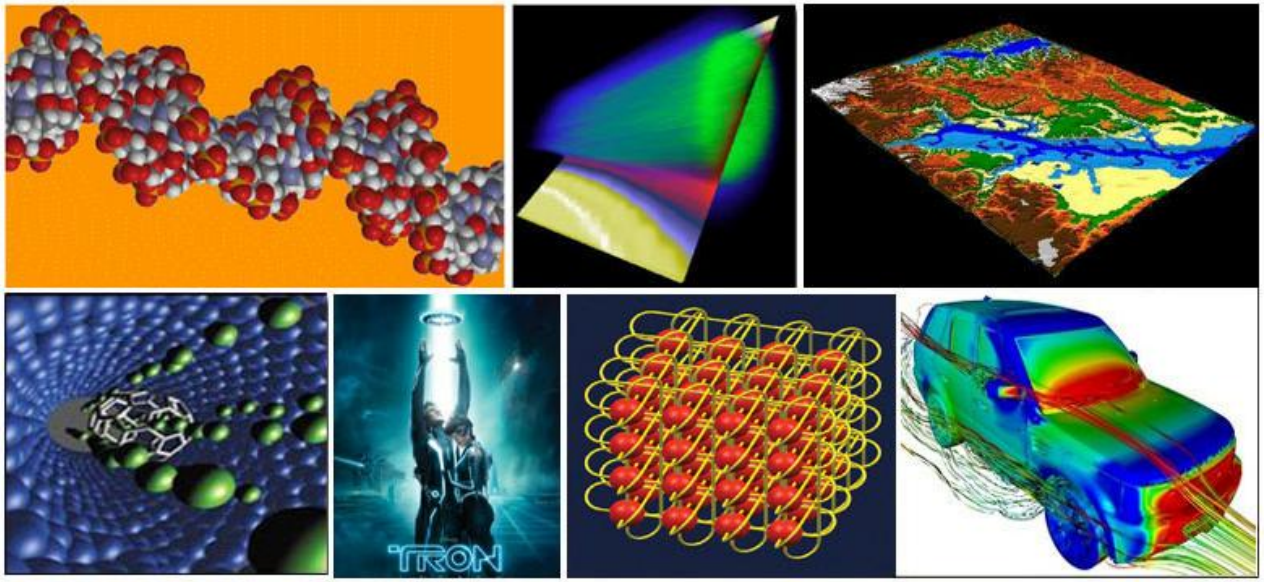
- Física – aplicada, nuclear, partículas, materia condensada
- Biociencia, bioingeniería, genómica
- Química, ciencias moleculares
- Geología, sismología
- Ingeniería mecánica, ingeniería electrónica
- Defensa
- Entre otras



¿Qué problemas requieren procesamiento paralelo?

- Industria y comercio
  - En la actualidad, diferentes área de la industria y el comercio requieren procesar grandes conjuntos de datos de forma sofisticadas.

- *Big data*, minería de datos
- Exploración de hidrocarburos
- Motores de búsqueda web
- Diagnóstico médico por imágenes
- Diseño de fármacos
- Modelización económica y financiera
- Computación gráfica, realidad virtual
- Muchas más



PROCESAMIENTO CONCURRENTE, PARALELO Y DISTRIBUIDO

Procesamiento concurrente, paralelo y distribuido

- No hay un completo acuerdo sobre la diferencia entre procesamiento concurrente, paralelo y distribuido.
- Algunos autores hacen las siguientes distinciones:
  - Un programa *concurrente* es aquel en el que múltiples tareas puede estar avanzando en cualquier instante de tiempo.
  - Un programa *paralelo* es aquel en el que múltiples tareas que se ejecutan simultáneamente cooperan para resolver un problema.
  - Un programa *distribuido* es aquel en el que múltiples tareas que se ejecutan físicamente en diferentes lugares cooperan para resolver uno o más problemas.

Procesamiento concurrente, paralelo y distribuido

- De las definiciones anteriores:
  - Los programas paralelos y distribuidos son entonces programas concurrentes
  - La concurrencia es un concepto de software y especificar la concurrencia implica *especificar los procesos* concurrentes, su *comunicación* y *sincronización*.

Procesamiento paralelo y distribuido: similitudes y diferencias

- El procesamiento paralelo busca reducir el tiempo de ejecución de un programa empleando múltiples procesadores al mismo tiempo.
  - El hardware brinda respuesta pero lo fundamental sigue siendo el software → Cómo desarrollar software que sea capaz de aprovechar el hardware subyacente
  - Una consecuencia inevitable e indeseada es la gran dependencia entre el software y el hardware subyacente para obtener alto rendimiento
  - Al mismo tiempo, el hardware evoluciona: de los multiprocesadores a los clusters, multiclusters, multicores, aceleradores…

Procesamiento paralelo y distribuido: similitudes y diferencias

- Un sistema distribuido es un conjunto de computadoras autónomas interconectadas que cooperan compartiendo recursos (físicos y datos).
  - Pueden ejecutar múltiples aplicaciones de diferentes usuarios.
  - La granularidad de los nodos y el grado de acoplamiento de los procesadores determina las características y aplicaciones de un sistema distribuido
  - En ocasiones, los sistemas distribuidos no se fabrican como tales, son que evolucionan a partir de redes LAN y WAN → Heterogeneidad
  - Problemas: no hay reloj único, se requiere planificación, la escalabilidad depende de las comunicaciones, heterogeneidad, seguridad de los datos, entre otros…

Procesamiento paralelo y distribuido: similitudes y diferencias - Resumen

- Características comunes:
  - Se usan múltiples procesadores
  - Los procesadores se encuentra interconectados por algún tipo de red
  - Múltiples tareas evolucionan al mismo tiempo y cooperan/compiten.

- Diferencias básicas:
  - Los programas paralelos se descomponen en tareas que se ejecutan al mismo tiempo
  - Los programas distribuidos se descomponen en tareas que se ejecutan físicamente en lugares diferentes con diferentes recursos locales

A veces se usan ambos términos con el mismo significado e incluso hay autores que consideran al paralelismo una sub-área del procesamiento distribuido.

SISTEMA PARALELO

Concepto de sistema paralelo

- Los algoritmos secuenciales son usualmente evaluados según su tiempo de ejecución, el cual varía de acuerdo al tamaño de la entrada.
- ¿Qué sucede con los algoritmos paralelos?
- Un algoritmo paralelo no puede ser evaluado en forma aislada de la máquina donde se ejecuta

Un **sistema paralelo** es la combinación de un algoritmo y la arquitectura paralela en la que se ejecuta

Portabilidad en sistemas paralelos

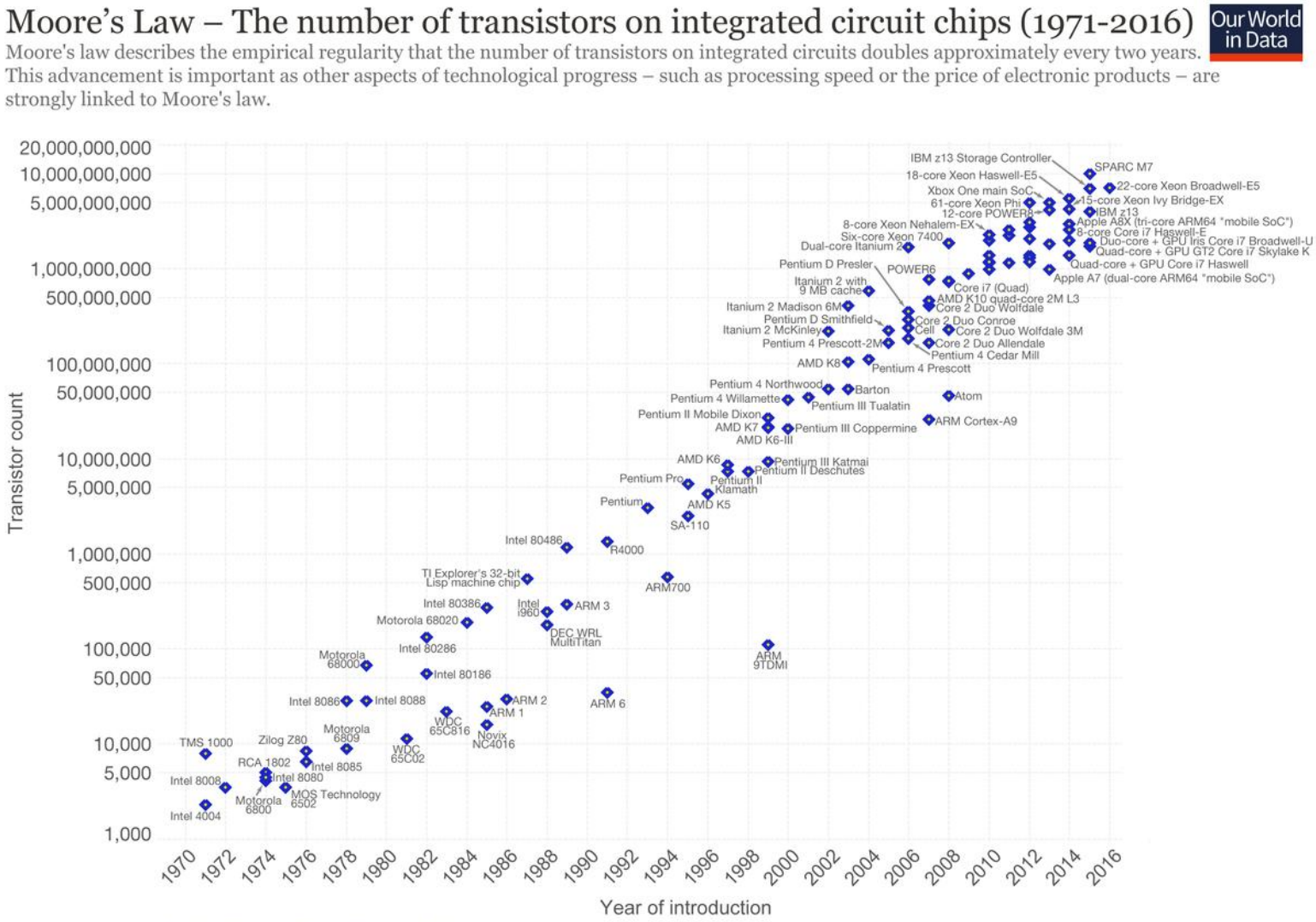
- La *portabilidad de código* se obtiene a través del uso de librerías que respetan estándares desarrollados por la comunidad académica y empresarial
  - Esto permite que un mismo programa pueda ejecutarse sobre diferentes arquitecturas
- Un programa tiene *portabilidad de rendimiento* cuando logra un rendimiento/eficiencia similar sobre diferentes arquitecturas.
  - ¿Se consigue al mismo tiempo que la portabilidad de código? ¿Por qué?

Modelos para sistemas paralelos

- En la capa de hardware tenemos múltiples alternativas: multicores, aceleradores, clusters, supercomputadoras, entre otros.
- En la capa de aplicación tenemos múltiples herramientas:
  - C, C++, Fortran, Java, Python, ADA…
  - Pthreads, OpenMP, MPI, CUDA, OpenCL, UPC…
- En la capa intermedia tenemos dos modelos:
  - Memoria compartida y memoria distribuida
  - También existen soluciones híbridas que combinan aspectos de las dos anteriores

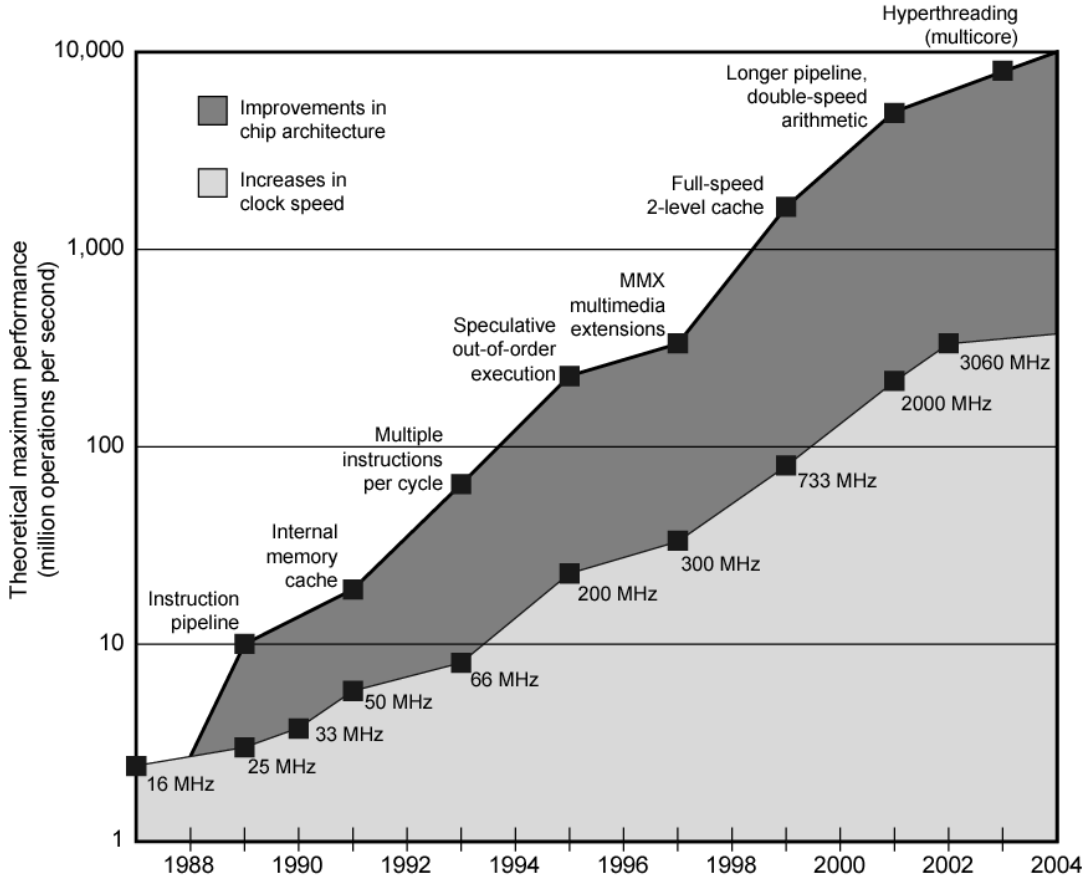
EVOLUCIÓN DE LOS PROCESADORES

Ley de Moore



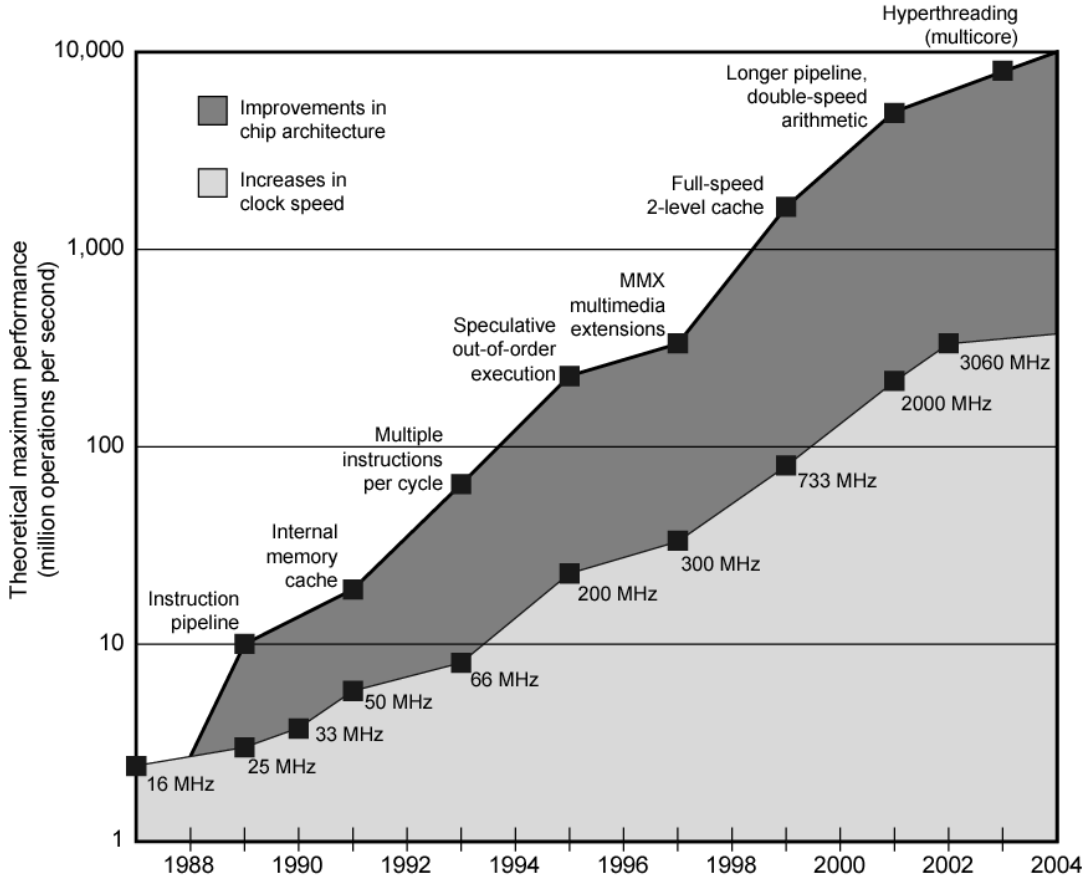
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Modelo tradicional de mejora de los procesadores



- Guiado por:
  - Aumento en el número de transistores en el chip
  - Aumento de la frecuencia del reloj
- Permitió implementar técnicas avanzadas de procesamiento que mejoran el rendimiento de las aplicaciones (ILP)

Modelo tradicional de mejora de los procesadores



- A principios de la década del 2000 llegó a su límite:
  - Imposibilidad de extraer más ILP de los programas secuenciales (*ILP wall*)
  - Imposibilidad de aumentar la frecuencia del reloj ante límites insostenibles de consumo de energía y disipación de calor (*power wall*)

Modelo tradicional de mejora de los procesadores

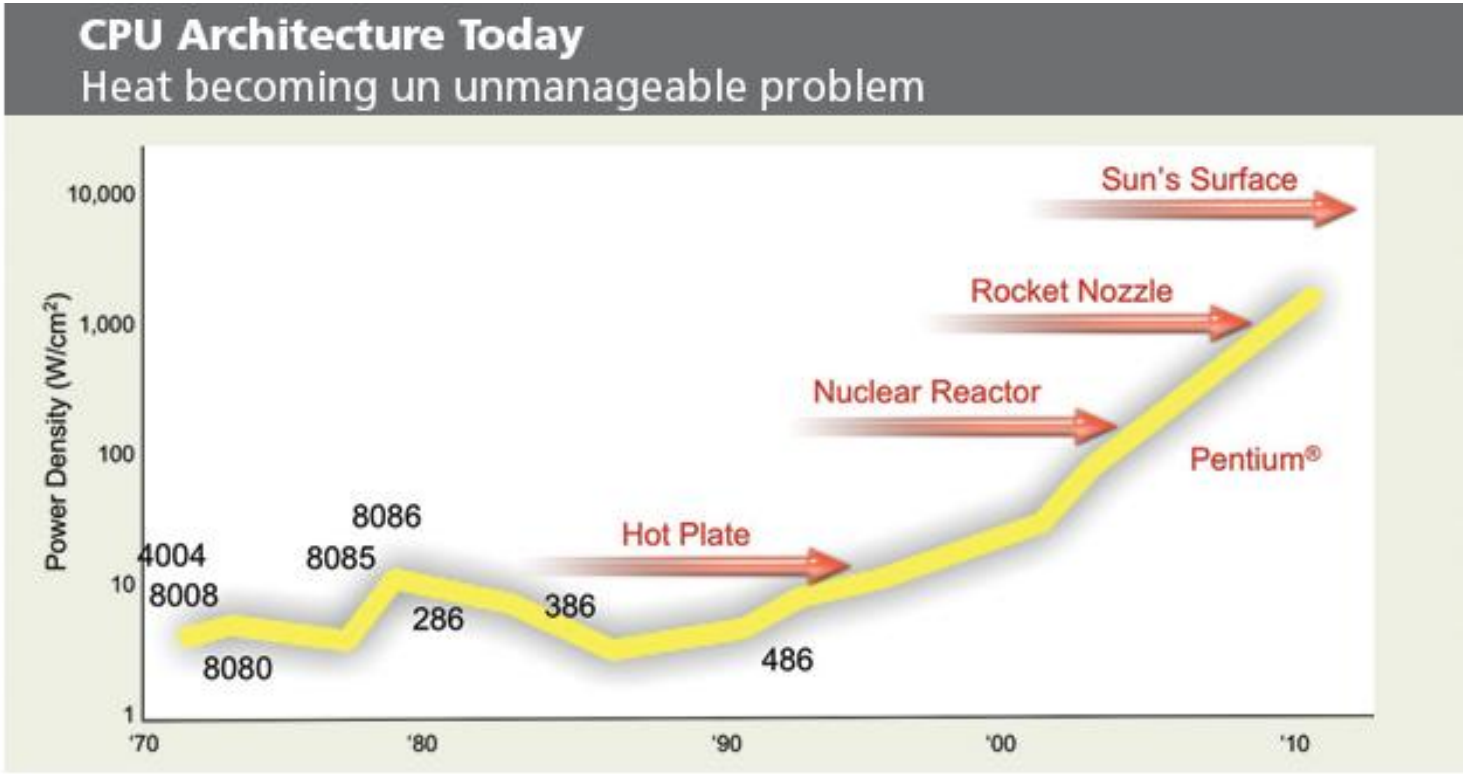


Figure 1. In CPU architecture today, heat is becoming an unmanageable problem. (Courtesy of Pat Gelsinger, Intel Developer Forum, Spring 2004)

Cambio de paradigma en la mejora de los procesadores

- Fue necesario buscar otras alternativas en el diseño de los procesadores que permitieran continuar incrementando su rendimiento.

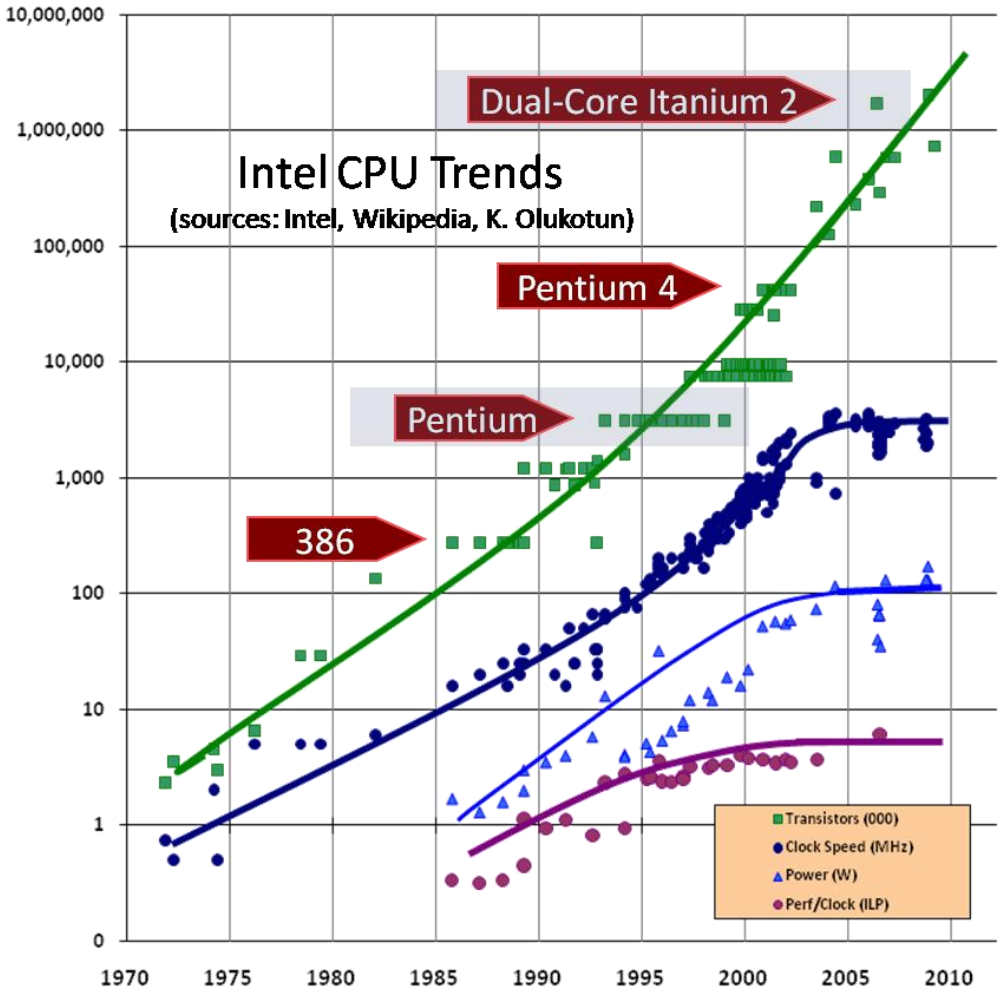
- En lugar de seguir aumentando la compleja organización interna del chip, las empresas fabricantes optaron por integrar dos o más núcleos computacionales (también llamados *cores*) más simples en un sólo chip (*multicores*).

- Si bien estos núcleos son más limitados y menos veloces, al combinarlos permiten mejorar el rendimiento global del procesador y al mismo tiempo hacerlo más eficiente energéticamente.

¿Qué consecuencias tuvo la introducción de los procesadores *multicore* para los programadores?

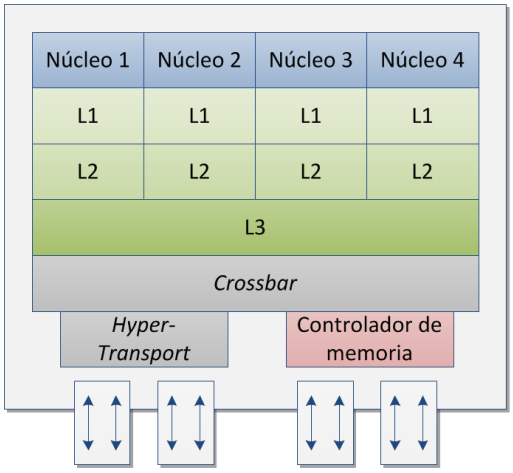
Necesidad de explotar paralelismo explícito para aprovechar potencia del procesador

Evolución de los procesadores

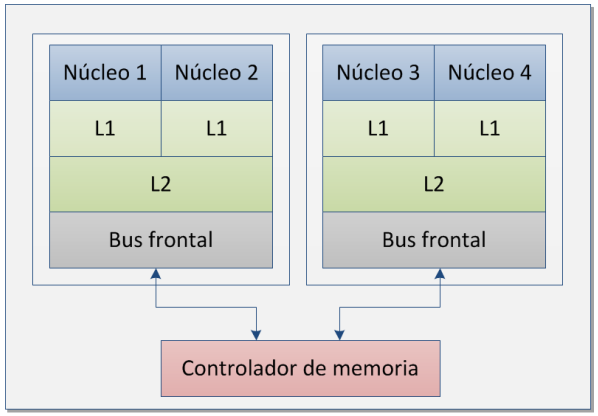


Procesadores multicore

- Desde su desarrollo, los procesadores multicore han sofisticado su diseño en las sucesivas familias.
- Los primeros procesadores de este tipo eran prácticamente dos procesadores mononúcleo en la misma oblea.
- Las siguientes generaciones han incrementado el número de núcleos e incorporado niveles de caché L2 y L3 (con o sin compartición).



AMD Opteron



Intel Xeon

Bibliografía usada para esta clase

- Capítulo 1, “Introduction to High Performance Computing for Scientists and Engineers”. Georg Hager, Gerard Wellein. CRC Press (2011).
- Capítulo 1, “An introduction to parallel programming”. Peter Pacheco. Elsevier (2011).
- “Introduction to parallel computing” Blaise Barney, Lawrence Livermore National Laboratory. https://computing.llnl.gov/tutorials/parallel_comp
- Capítulo 5, “Introduction to Parallel Computing”. Grama, Gupta, Karypis, Kumar. Addison Wesley (2003).
- Capítulo 1, “Computer Organization and Architecture. Designing for Performance”. W. Stallings (2010)