“Computación de Altas Prestaciones: Paradigmas y Bibliotecas de Programación basadas en Software Libre”

JORNADAS SOBRE COMPUTACIÓN DE ALTAS PRESTACIONES Y SOFTWARE LIBRE

Ourense, 5 de Octubre de 2011

JUAN TOURIÑO DOMÍNGUEZ

Dpto. Electrónica y Sistemas

Universidad de A Coruña

http://www.des.udc.es/~juan

Contenidos

- Introducción: Motivación y Aplicaciones HPC
- Arquitectura de Sistemas HPC
- Programación de Sistemas HPC
  - Paradigma de Memoria Compartida
  - Paradigma de Memoria Distribuida
  - Programación “Híbrida”
  - Bibliotecas de Software Libre HPC
  - Lenguajes PGAS
  - Lenguajes HPCS
- Sistemas HPC Emergentes: GPUs
- HPC en Galicia

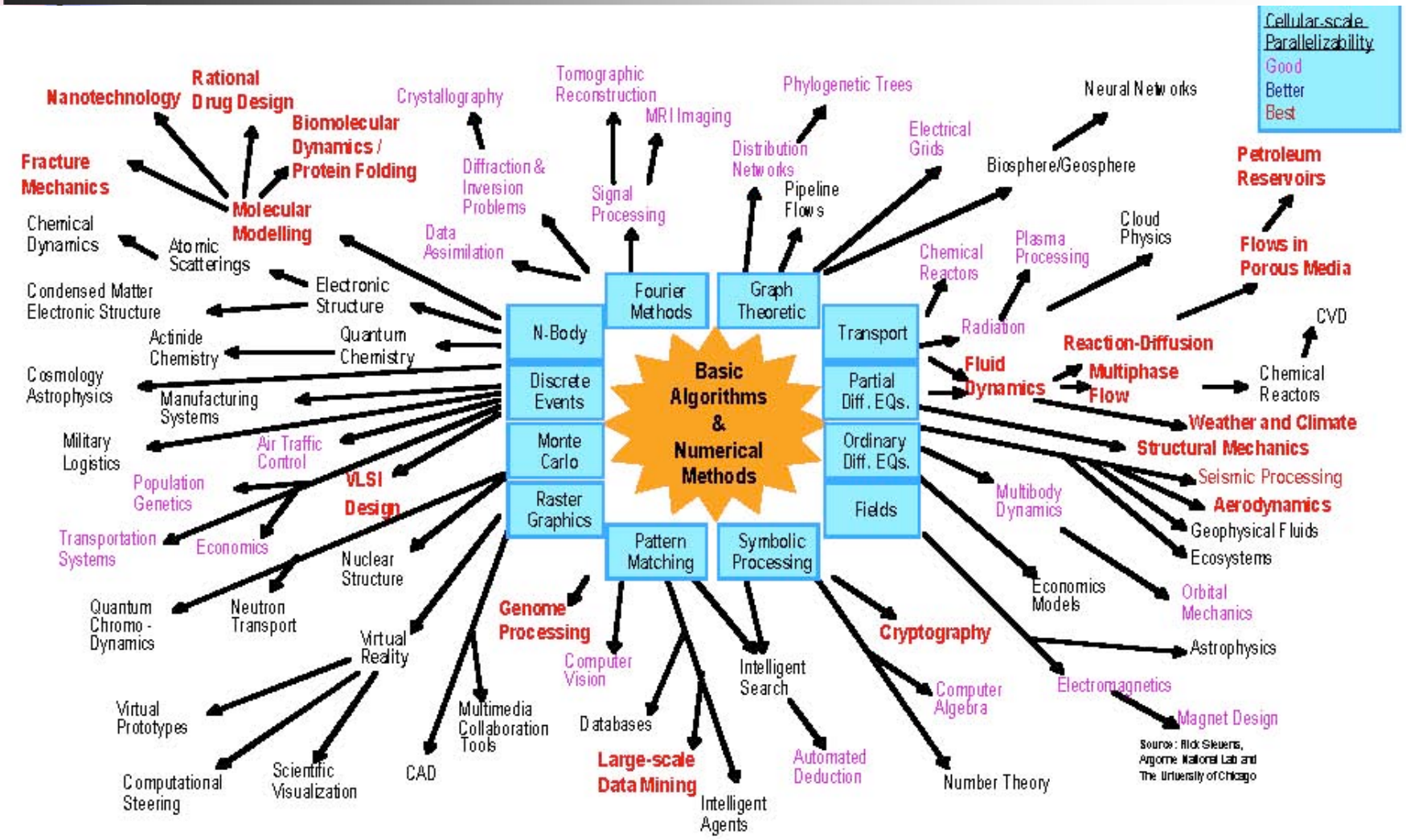
Introducción: Motivación

- **Sistemas HPC (*High Performance Computing*):** Supercomputadores, clusters … Ubicados en Centros de Supercomputación, Empresas, Universidades, Centros de I+D
- Objetivo: obtener los mismos resultados utilizando múltiples procesadores
- ¿Cuándo recurrir a sistemas HPC?:
  - Requerimientos computacionales superiores a los proporcionados por sistemas monoprocesador convencionales
  - Necesidad de resolver problemas “más reales” y de forma más precisa -> modelos matemáticos más complejos -> grandes requerimientos de computación, de memoria y/o disco
  - Obtención de resultados en un tiempo aceptable
- De hecho, ciertas aplicaciones, sólo es posible ejecutarlas en sistemas HPC (ej. *Grand Challenge applications*)
- Principales obstáculos: acceso a sistemas HPC, “dificultad” de programación (aprendizaje, nueva “filosofía” de programación …)

Introducción: Aplicaciones (I)

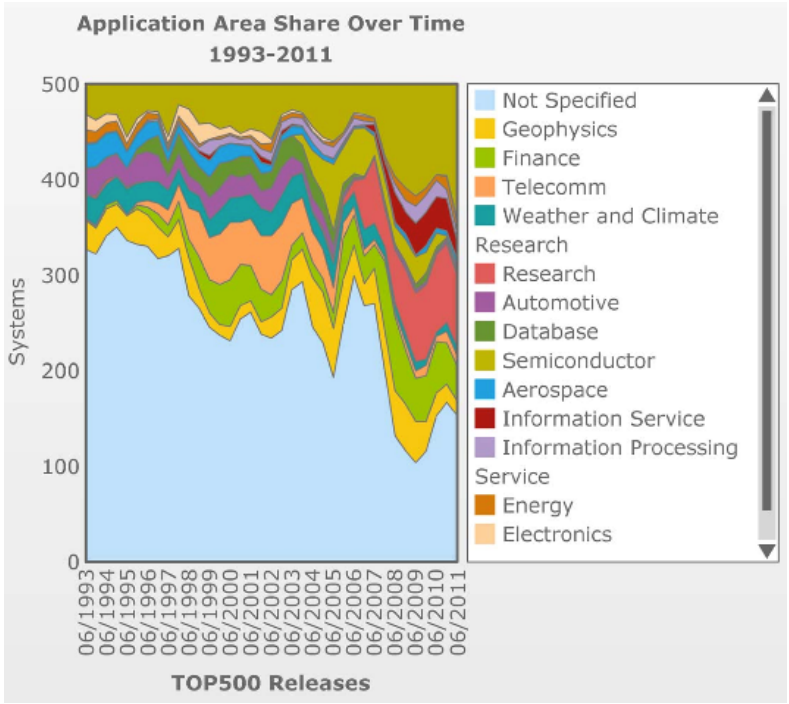
- Aplicaciones/campos de ciencia e ingeniería frecuentes en sistemas HPC (principalmente modelado y simulación):
  - Industria de automoción, naval y aeroespacial
  - Ingeniería estructural
  - Energía (ej. simulación de combustiones)
  - Mecánica de fluidos (CFD)
  - Simulación multifísica
  - Otros: química computacional (ej. dinámica molecular), bioinformática, predicción metereológica, modelado de seísmos y océanos, modelado de sistemas y circuitos electrónicos
- Ejemplos de métodos/algoritmos núcleo de numerosas aplicaciones y muy adecuados para sistemas HPC:
  - Simulaciones de Montecarlo
  - N-body
  - FFT
  - Particionamiento de grafos
  - Optimización combinatoria: algoritmos genéticos
  - Algoritmos numéricos: álgebra matricial densa y dispersa, resolución de sistemas de ecuaciones (métodos directos e iterativos), ecuaciones diferenciales-FEM …

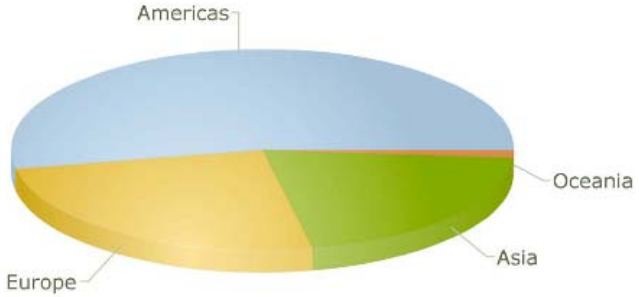
Introducción: Aplicaciones (II)

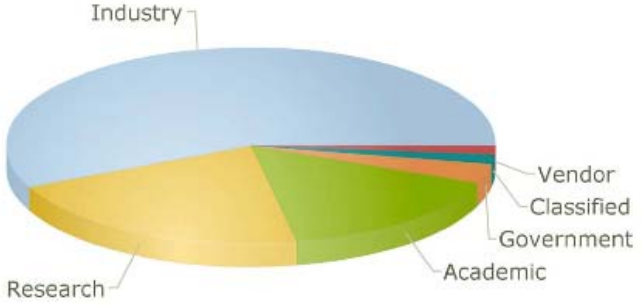


Introducción: Aplicaciones (y III)

- Lista TOP500 (Junio 2011). http://www.top500.org





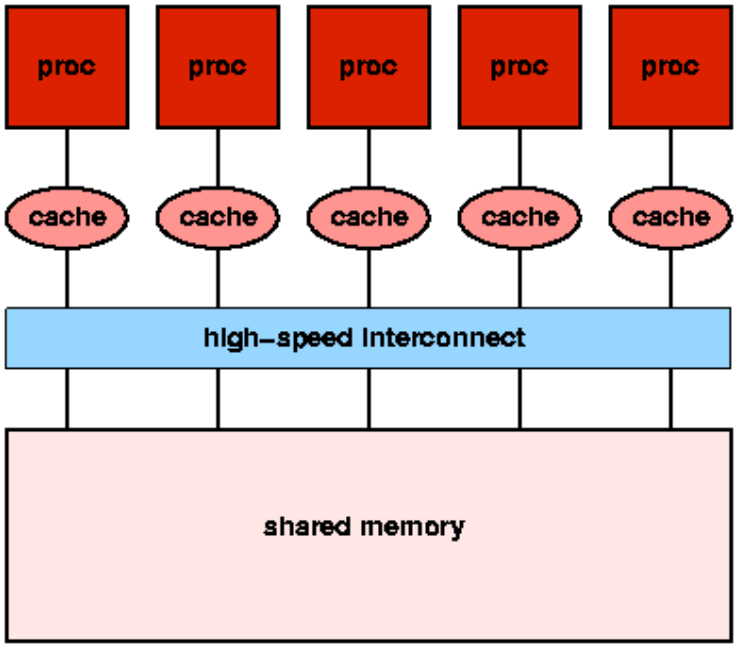


Arquitectura de Sistemas HPC (I)

- En la mayoría de los problemas computacionales no es necesario recurrir a supercomputadores del TOP500 -> clusters de mediana/pequeña escala (multicores)

Arquitectura de Sistemas HPC (II)

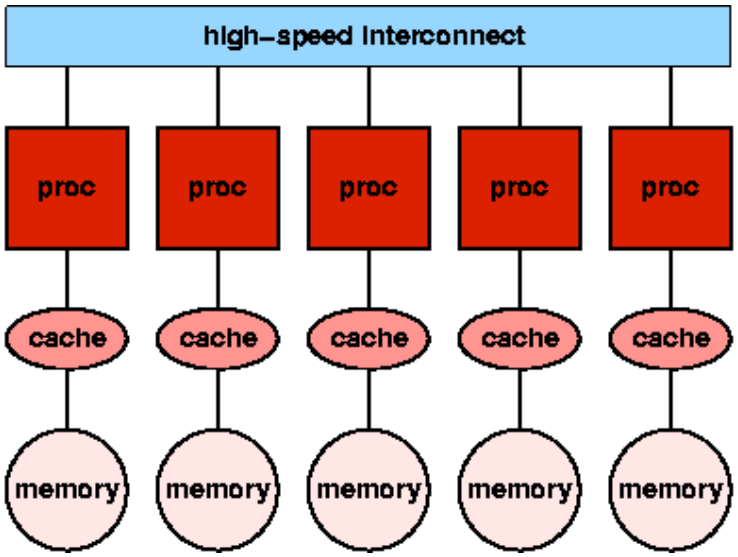
- Memoria compartida (Multiprocesadores)



• Todos los procesadores comparten el mismo espacio de direcciones (no necesario distribuir datos!)
• Comunicación implícita a través del acceso a variables compartidas
• Programación más sencilla (en general)
• Necesarios mecanismos de sincronización (semáforos, secciones críticas, barreras …)
• Actualmente en auge por las **arquitecturas multicore**

Arquitectura de Sistemas HPC (y III)

- Memoria distribuida (Multicomputadores)

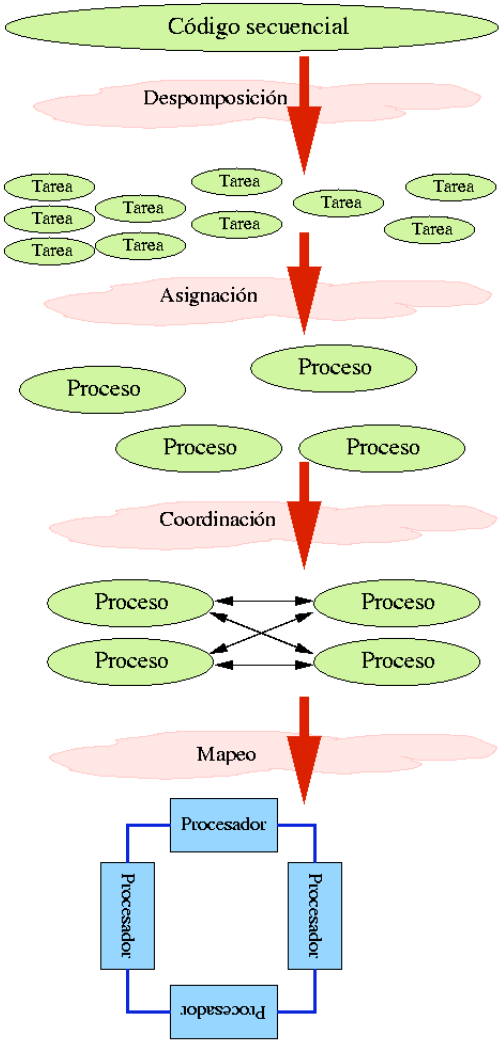


• Cada procesador tiene su propio espacio de direcciones privado (necesario distribuir datos entre memorias locales; el programador necesita conocer dónde están los datos ⇒ explotación de localidad)
• Comunicación explícita a través de mensajes
• Programación más compleja
• Sincronización más sencilla (normalmente implícita en los mensajes)
• Actualmente en auge por las **arquitecturas cluster** (escalabilidad)

Programación de Sistemas HPC

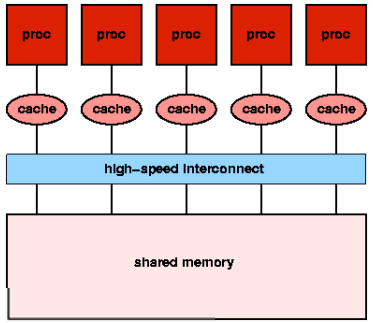
- Fases programación paralela:
  - Descomposición: dividir computaciones en tareas para ser distribuidas entre los procesadores (balanceo de carga)
  - Asignación: división de tareas entre los procesos/threads (estática vs dinámica)
  - Coordinación: comunicaciones/sincroni-zaciones necesarias (minimizar)
  - Mapeo: decidir qué proceso/thread se ejecutará en cada procesador (normalmente mapeo 1:1)

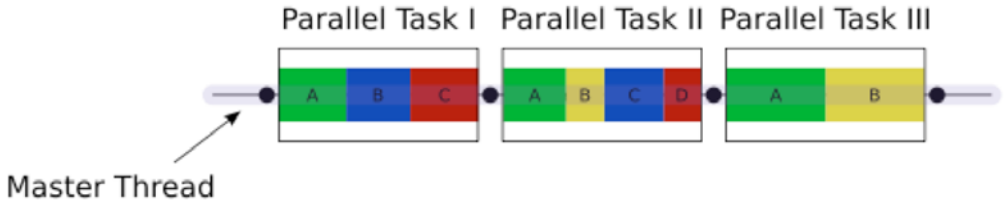
- Centrar el esfuerzo en las zonas del programa computacionalmente más costosas (*hot spots*)

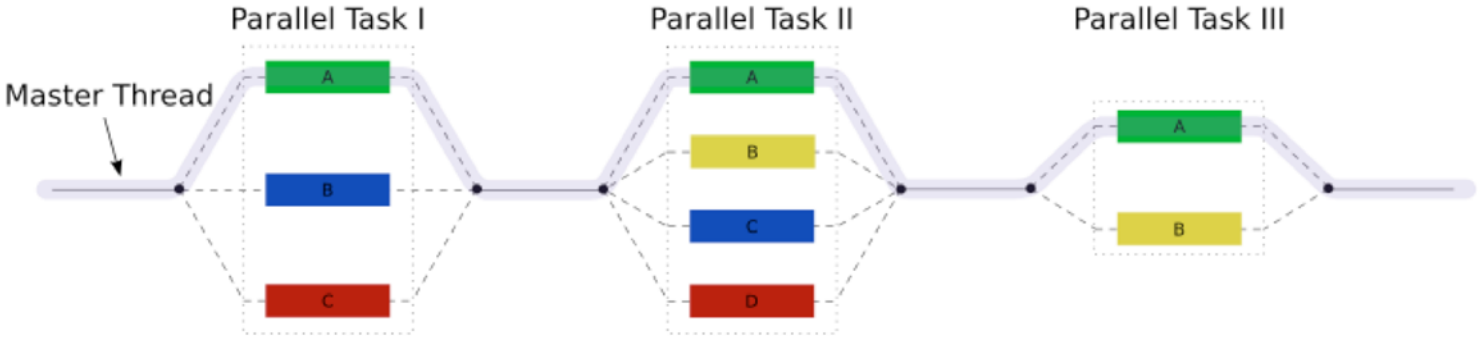


Programación de Sistemas HPC de Memoria Compartida (I)

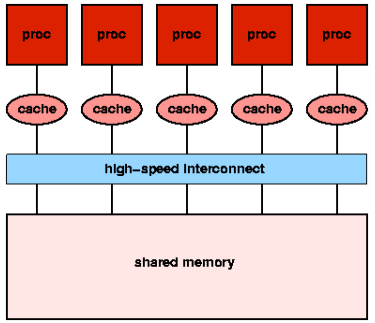
- Memoria compartida
- Programación concurrente (basada en threads, modelo *fork-join*):



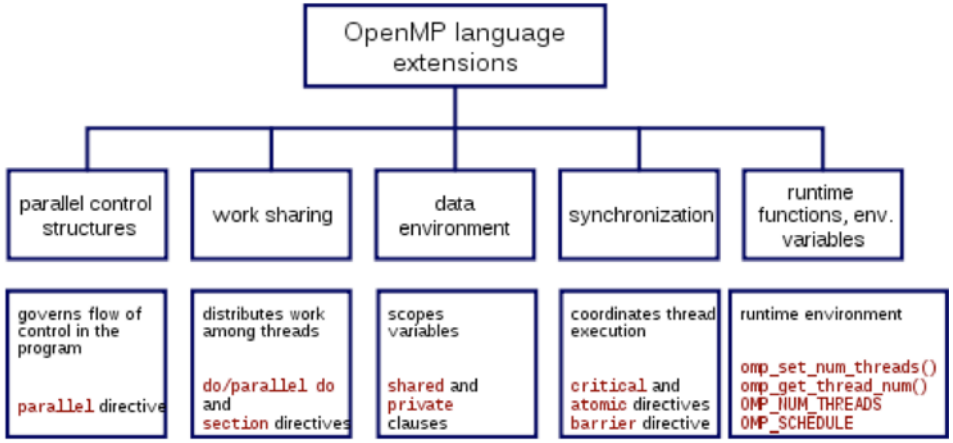




Programación de Sistemas HPC de Memoria Compartida (II)



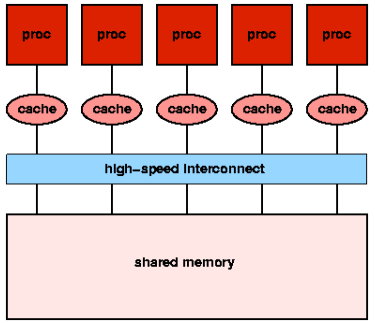
- Threads nativos (ej. Pthreads) -> programación de bajo nivel (aunque portable), basada en llamadas a bibliotecas
- **OpenMP**: API estándar para programación en sistemas de memoria compartida (Fortran, C/C++) -> programación de alto nivel (directivas en el código secuencial) -> Portabilidad



http://www.openmp.org

Programación de Sistemas HPC de Memoria Compartida (III)

- Diversos niveles de paralelismo con OpenMP
  - A nivel de tarea

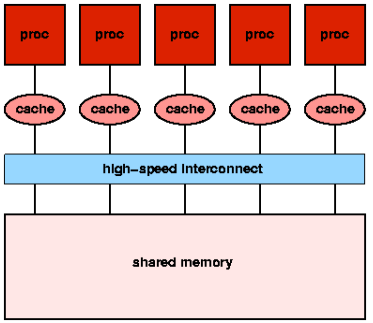



```
            integer n
            real a(n)

!$OMP PARALLEL
!$OMP SECTIONS
!$OMP SECTION
            call calcular_media(a,n)
!$OMP SECTION
            call calcular_maximo(a,n)
!$OMP SECTION
            call calcular_desviacion(a,n)
!$OMP END  SECTIONS
!$OMP END PARALLEL
```

Programación de Sistemas HPC de Memoria Compartida (IV)

- Diversos niveles de paralelismo con OpenMP
  - A nivel de lazo




```
!$OMP PARALLEL DO
      DO i=1,1000
         A(i)=B(i)
      END DO
```

(en C #pragma parallel for)

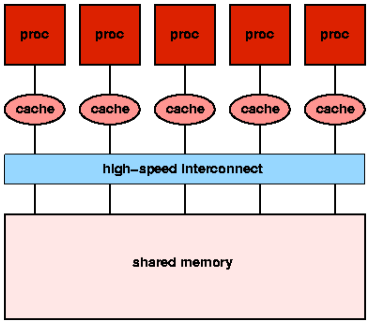
Thread 1

```
DO i=1,500
   A(i)=B(i)
END DO
```

```
DO i=501,1000
   A(i)=B(i)
END DO
```

Thread 2

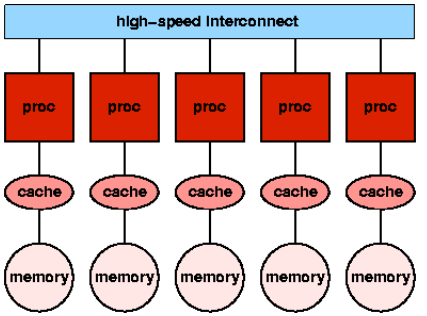
Programación de Sistemas HPC de Memoria Compartida (y V)

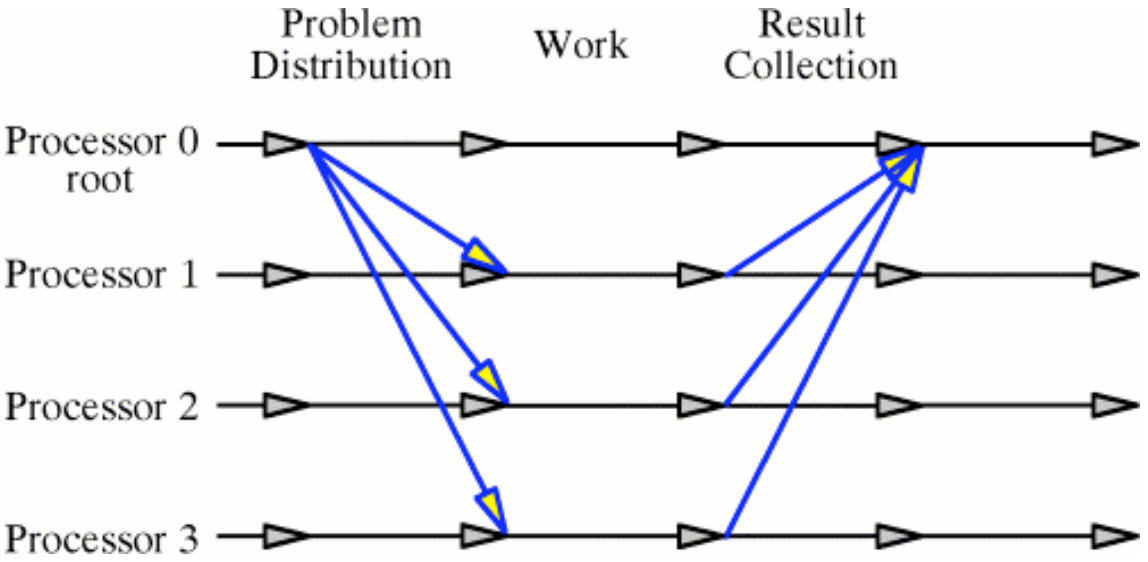


- Los grandes fabricantes tienen sus compiladores de OpenMP (ej. Intel, IBM, HP, Sun)
- Software libre: los compiladores de GNU (*gcc4* y *gfortran*) incluyen soporte para OpenMP a partir de la versión 4.2, compilando con la opción: ***-fopenmp***
- Bibliotecas numéricas/científicas de sw. libre de soporte para programación en OpenMP son escasas (existen bibliotecas propietarias -e.g. MKL de Intel con soporte OpenMP -).
- Más información sobre la implementación GNU de OpenMP en http://gcc.gnu.org/projects/gomp
- Otros mecanismos de programación en memoria compartida (basados en threads):
  - **Cilk**: http://supertech.csail.mit.edu/cilk/ (lenguaje basado en C)
  - **Intel TBB**: http://www.threadingbuildingblocks.org/ (biblioteca para expresar paralelismo en C++)

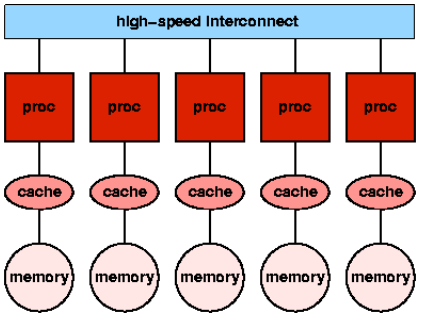
Programación de Sistemas HPC de Memoria Distribuida (I)

- Memoria distribuida
- Programación paralela (basada en procesos que se comunican a través de mensajes):





Programación de Sistemas HPC de Memoria Distribuida (II)

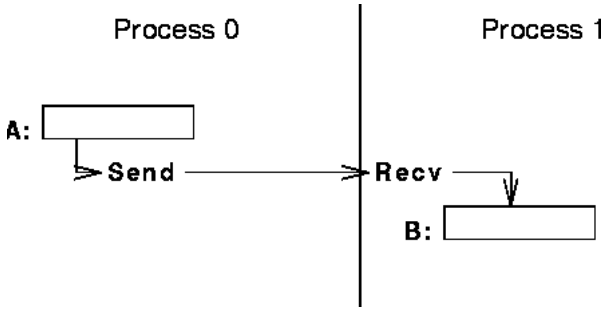


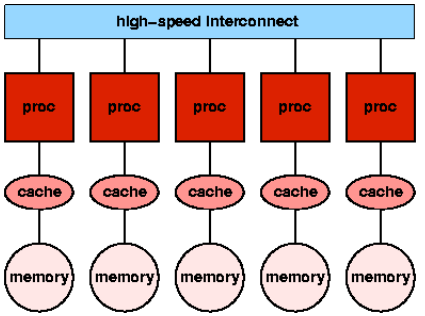
- **MPI (Message-Passing Interface)**: API estándar para programación de sistemas de memoria distribuida -> programación de bajo nivel (biblioteca de paso de mensajes para Fortran y C/C++) -> Portabilidad
- Programación de más bajo nivel que OpenMP, más "difícil", menos productiva (programas paralelos más diferentes de la versión secuencial), pero más flexible
- Rutinas de la biblioteca:
  - Comunicaciones entre procesos:
    - Punto a punto: *MPI_Send(…), MPI_Recv(…)*
    - Colectivas: *MPI_Bcast(…), MPI_Scatter(…), MPI_Gather(…), MPI_Reduce(…)*
  - Otras (más avanzadas): tipos de datos derivados, topologías virtuales, comunicadores

http://www.mpi-forum.org
http://www-unix.mcs.anl.gov/mpi/

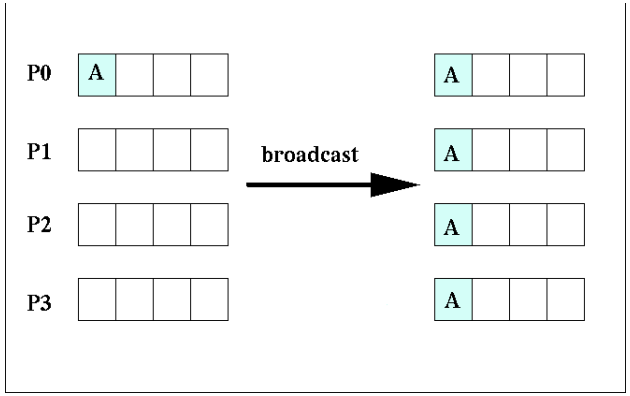
Programación de Sistemas HPC de Memoria Distribuida (III)

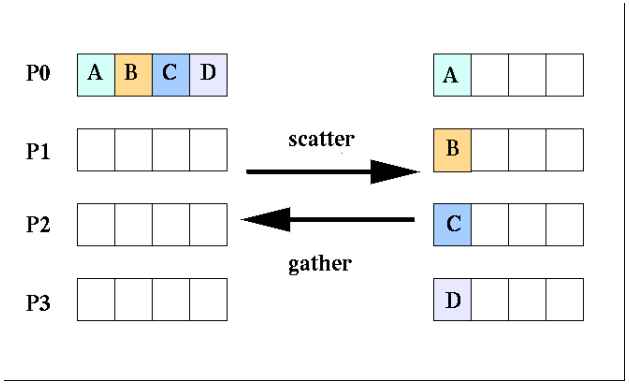
- Ejemplos de rutinas de comunicación:





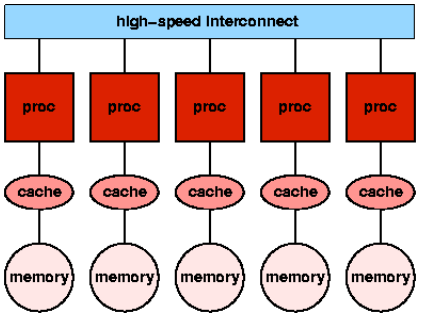
Punto a punto: *MPI_Send(…)* *MPI_Recv(…)*





Colectivas: *MPI_Bcast(…)* *MPI_Scatter(…)* *MPI_Gather(…)*

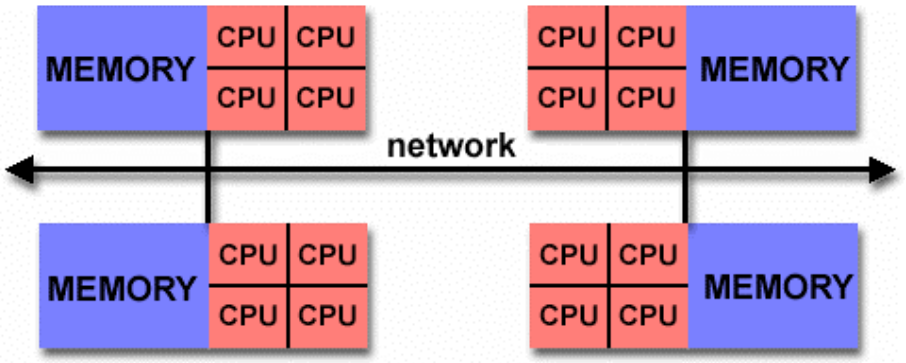
Programación de Sistemas HPC de Memoria Distribuida (y IV)



- Los grandes fabricantes tienen versiones de MPI optimizadas para sus sistemas (ej. Intel MPI, HP MPI…)
- Bibliotecas MPI de software libre:
  - **MPICH** (MPICH2): http://www.mcs.anl.gov/research/projects/mpich2/
  - **OpenMPI**: http://www.open-mpi.org
- Incluyen soporte para redes de interconexión de altas prestaciones: Infiniband, Myrinet …
- Programación en MPI (en general con paso de mensajes) tediosa -> Bibliotecas matemáticas/científicas paralelas para facilitar la programación de aplicaciones científicas y de ingeniería en MPI
- Otras bibliotecas de paso de mensajes: **PVM** (Parallel Virtual Machine), prácticamente en desuso

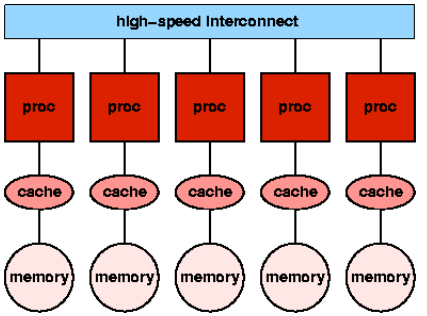
Programación “Híbrida” de Sistemas HPC

- Motivación: arquitecturas distribuidas constituidas por nodos de memoria compartida. Ej.: clusters de multicores



- Posibilidades de programación:
  - Mem. distribuida: 1 proceso MPI por CPU (core)
  - Híbrida: 1 proceso MPI por nodo que lanza 1 thread OpenMP en cada CPU (core) -> OpenMP para comunicación intra-nodo y MPI para comunicación inter-nodo
- Necesidad de versiones de MPI *thread-safe* (MPICH2 y OpenMPI incluyen este soporte en sus últimas versiones)
- Tendencia actual: programación con MPI (ventaja: uso de un único paradigma). Versiones actuales de MPI están mejorando las comunicaciones intra-nodo

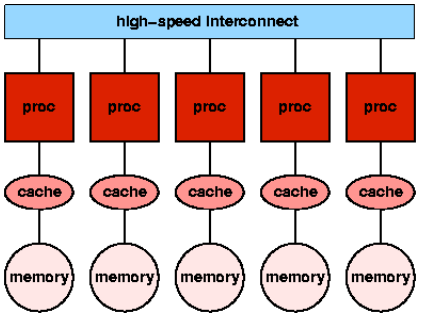
Bibliotecas Software Libre HPC (I)



BIBLIOTECAS MATEMÁTICAS (soporte MPI para numerosas aplicaciones HPC en ciencia e ingeniería)

- **ScaLAPACK:** http://www.netlib.org/scalapack/
  - Rutinas paralelas de álgebra lineal (básicamente usan como núcleo versiones paralelas de BLAS)
  - Ejemplo de funcionalidades (matrices densas):
    - Resolución de gran variedad de sistemas de ecuaciones
    - Factorizaciones (LU, QR, Cholesky …)
    - Cálculo de autovalores y autovectores
    - …
- **PSPASES:** http://www-users.cs.umn.edu/~mjoshi/pspases/
  - Conjunto de rutinas paralelas para la resolución de sistemas de ecuaciones lineales *sparse* (matrices dispersas)
  - Fases: reordenamiento para reducir el *fill-in* (llenado), factorización simbólica, factorización numérica, resolución de los sistemas de ecuaciones triangulares

Bibliotecas Software Libre HPC (II)

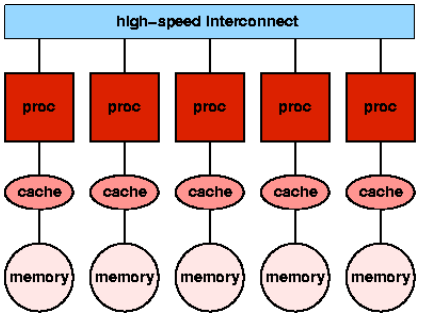


- **ParMETIS:**
  http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview
  - Biblioteca MPI con diversos algoritmos de particionamiento de grafos, mallas, reordenamientos para reducir el *fill-in* de matrices dispersas.
  - Dominio de aplicación: simulaciones numéricas gran escala y computación paralela para AMR (refinamiento de mallas adaptativo)

- **FFTW:** http://www.fftw.org/
  - Rutinas para el cálculo de la DFT en 1 o más dimensiones (y diversas variantes)
  - Dispone de una versión MPI

- **SPRNG:** http://sprng.cs.fsu.edu/
  - ***S**calable **P**arallel **R**andom **N**umber **G**enerators Library*
  - Biblioteca para la generación paralela de números pseudoaleatorios, de aplicación en simulaciones paralelas de Monte Carlo a gran escala

Bibliotecas Software Libre HPC (III)



- **PETSc:** http://www.mcs.anl.gov/petsc/petsc-as
  - ***P**ortable, **E**xtensible **T**oolkit for **S**cientific **C**omputation*
  - Rutinas y estructuras de datos para la solución paralela de aplicaciones científicas modeladas por PDEs:
    - Estructuras de datos y rutinas para la manipulación de matrices *sparse* paralelas
    - Implementación paralela de métodos iterativos de Krylov (ej. GMRES, CG, CGS, Bi-CG-Stab …)
    - Precondicionadores paralelos (ej. ILU(k), LU, block Jacobi…)
    - Resolutores no lineales paralelos basados en métodos de Newton
    - *…*

- **HPC-Netlib**: http://www.nhse.org/hpc-netlib/
  - Repositorio de software matemático para HPC (en su mayoría libre y no sólo para MPI)
  - Es parte de la NHSE (National HPCC Software Exchange): http://www.nhse.org, colección de material (software, noticias, documentos …) de la comunidad HPC

Bibliotecas Software Libre HPC (y IV)

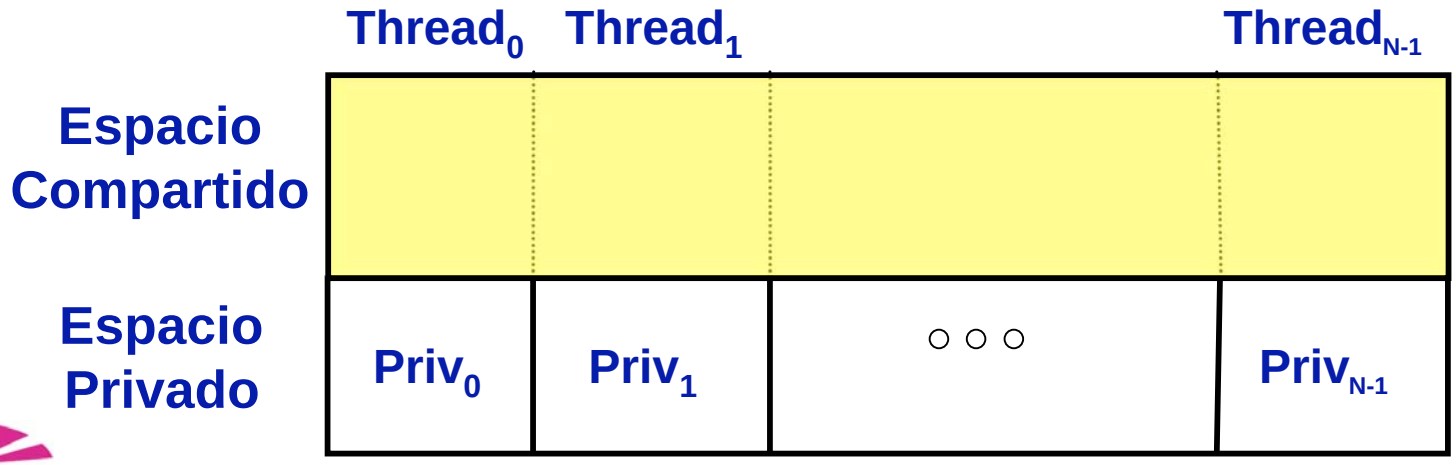
- Existen versiones/soporte/interoperabilidad HPC (mayoritariamente MPI) para gran variedad de software libre en Ingeniería y Ciencia Aplicadas:
  - Python: **MPI Python (pyMPI)** (MPI para programas Python; permite comunicación de objetos Python con MPI): http://sourceforge.net/projects/pympi/
  - Octave: **MPITB** (*MPI Toolbox for Octave*) (posibilita llamar a rutinas MPI dentro del entorno Octave): http://atc.ugr.es/javier-bin/mpitb
  - R (estadística): **RMPI** (interfaz MPI para R): http://www.stats.uwo.ca/faculty/yu/Rmpi/
  - Salome (desarrollo asistido por ordenador): En desarrollo soporte MPI ("*Ongoing developments: MPI container for parallel computing*")
  - Code Aster (simulación termomecánica): *"Parallel version of Code Aster"* (OpenMP y MPI). Consultar:

    http://www.codeaster.org/wiki/doku.php?id=en:p02_install:p100_aster_mpi
  - Elmer (simulación en multifísica): Versión paralela en MPI ("*The parallel version of ElmerSolver is based on the Message-Passing Interface -MPI- library*"). Consultar:

    http://www.csc.fi/english/pages/elmer
  - OpenFOAM (simulación en fluidos): Con soporte MPI. Consultar: http://www.openfoam.com/features/parallel-computing.php

Lenguajes PGAS (I)

- Modelo de programación **PGAS** = *Partitioned Global Address Space* (Espacio de direcciones global particionado). También llamado DSM (*Distributed-Shared Memory*, Memoria Compartida-Distribuida)
- El programa paralelo se compone de un conjunto de threads que comparten un espacio de direcciones global (mem. compartida) que está lógicamente particionado entre ellos (mem. distribuida)
- Cada porción de mem. compartida tiene "afinidad" con un thread concreto ⇒ modelo de programación "productivo" (mem. compartida) permitiendo también explotar la localidad (mem. distribuida)



Lenguajes PGAS (II)

- Extensiones del estándar de lenguajes existentes (C, Fortran, Java) para favorecer su adopción:
  - **UPC**: Unified Parallel C (extensión de C). http://upc.gwu.edu
    - Compiladores open-source: *Berkeley UPC* (LBNL): http://upc.lbl.gov, GCC UPC
    - Compiladores propietarios: HP, IBM, Cray …
  - **CAF**: Co-array Fortran (extensión de Fortran 95/2003, los coarrays han sido incluidos en el estándar Fortran 2008). http://www.co-array.org
    - Compiladores open-source: incluyéndose funcionalidades CAF en *g95* y en *GNU gfortran*, en progreso compilador *Rice CAF* (no sigue el estándar)
    - Compiladores propietarios: Cray, Intel
  - **Titanium** (extensión de Java). Presencia prácticamente nula, implementación libre de Berkeley: http://titanium.cs.berkeley.edu
- Bibliotecas de soporte (libres o propietarias) prácticamente inexistentes. Necesidad de adopción de los lenguajes y maduración de los compiladores

Lenguajes PGAS (y III)

- Ejemplos UPC. Declaración de variables (privadas y compartidas):

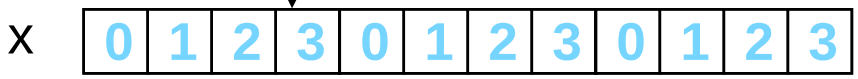
```
int x;
shared int y;
```

Privada. Hay una *x* (diferente) en el espacio privado de cada thread

Compartida. Hay una única *y* en el espacio compartido, que tiene afinidad al thread 0

```
shared int x[12];
```

Por defecto los elementos de arrays se distribuyen **cíclicamente** entre los threads



Para 4 threads

- Construcción para reparto de trabajo entre threads (paralelismo a nivel de lazo):

```
upc_forall(i=0; i<N; i++; i)
    …
```

- Mecanismos de **sincronización**: barreras (*upc_barrier*), locks para secciones críticas (*upc_lock)*

- **Operaciones colectivas** (con la cabecera *upc_collective.h*): broadcast, gather, scatter, reduce, etc.

Lenguajes HPCS

- **HPCS**: *High Productivity Computer Systems*
- Programa del DARPA (*Defense Advanced Research Projects Agency*, EE.UU.) para la creación de una nueva generación de lgjes. (tb. denominados APGAS) para la prog. concurrente/paralela de sistemas HPC. Objetivo: aunar programabilidad, productividad, rendimiento, portabilidad y robustez:
  - **X10** (IBM): Lenguaje orientado a objetos, basado en Java (pero no es una extensión). http://x10-lang.org (open source, licencia EPL)
  - **Chapel** (Cray). Lenguaje imperativo, estructurado en bloques (soporta OO, pero su uso es opcional). Nuevo, no es ninguna extensión ni está basado en un lgje. existente (coge las mejores características de varios lenguajes: Java, C, C++, C#, HPF, ML, etc.). http://chapel.cray.com (proyecto open source, licencia BSD)
  - **Fortress** (Sun, sin financiación del programa HPCS desde Nov. 2006). Nuevo lenguaje (incluso + de cero que Chapel) inspirado en lenguajes OO (Java) y funcionales. Uso de una notación muy próxima a la matemática para favorecer su adopción. http://projectfortress.java.net (proyecto open source, licencia BSD)
- Ventajas: lenguajes desde cero, con expresión de paralelismo de forma más “natural” que partiendo de lenguajes secuenciales ya existentes
- Inconvenientes: difícil adopción de nuevos lenguajes, compiladores poco maduros, inexistencia de bibliotecas de apoyo para la programación

Sistemas HPC emergentes: GPUs (I)

- **GPU** (Graphic Processing Unit): Dispositivo de propósito específico para procesamiento gráfico (ej. gráficos 3D interactivos, renderizado 3D …).
- GPUs actuales constituidas por procesadores many-core compuestos por cientos de núcleos de procesamiento (menos potentes que los de procesadores multicore convencionales)
- Tipos de paralelismo en la GPU:
  - A nivel de tarea: procesamiento de un elevado número de tareas en paralelo
  - A nivel de datos (data parallel): el mismo programa se ejecuta sobre datos ≠ en paralelo (aplicaciones con poco control de flujo y alta intensidad aritmética)
- Solución de alto rendimiento no solo para para el procesamiento gráfico para el que fueron diseñadas, sino también para el no gráfico: **GPGPU** (General-Purpose Computation on GPUs). Aplicaciones:
  - Simulaciones en ciencia e ingeniería
  - Procesamiento de imágenes
  - Codificiación y decodificación de video
  - Reconocimiento de formas
  - Computación financiera

Sistemas HPC emergentes: GPUs (y II)

- GPU: excelente relación coste/rendimiento en comparación con arquitecturas multicores basada en CPUs
- Ej.: tarjeta gráfica de NVIDIA con arquitectura FERMI (2010), orientada no solo a gráficos sino también al mercado de computación GPGPU
- Programación (+ destacables):
  - **CUDA**: entorno de programación (threads) basado en C para GPUs de NVIDIA (propietario) http://developer.nvidia.com/category/zone/cuda-zone
  - **OpenCL**: estándar abierto para programación portable de plataformas heterogéneas (CPU+GPU+Cell+DSP), soporta paralelismo de datos y de tareas. http://www.khronos.org/opencl/
    - Componentes: APIs para una librería de control de plataformas y un lenguaje de programación basado en C
    - Numerosas implementaciones de fabricantes (Apple, AMD, NVIDIA, Intel, IBM, etc.)
    - Necesidad de maduración (actualmente menor rendimiento que CUDA para GPUs de NVIDIA)
  - En curso: directivas para programación de alto nivel de GPUs ⇒ Objetivo: facilitar la prog. Ej: **directivas OpenHMPP** (Hybrid Multicore Parallel Programming), estándar abierto, de momento con implementaciones propietarias**.** http://www.openhmpp.org
  - Bibliotecas de software libre para GPUs prácticamente inexistentes (hay bibliotecas matemáticas de CUDA propietarias) ⇒ Campo con grandes posibilidades de desarrollo

HPC en Galicia

- **CESGA** (Centro de Supercomputación de Galicia): http://www.cesga.es
  - Acceso a sistemas HPC
  - Asesoramiento técnico para ejecutar aplicaciones en sistemas HPC
  - Formación en HPC (ver enlace "*Formación*" en su web)

- **Red GHPC** (Red Gallega de Computación de Altas Prestaciones): http://ghpc.udc.es
  - Agrupación de expertos en tecnologías HPC y usuarios de las mismas (desde 2007). Objetivo: potenciar colaboraciones interdisciplinares entre ellos
  - Grupos de investigación de las 3 universidades gallegas + CESGA: informática, telecomunicaciones, física, matemáticas, ingeniería …
  - Objetivo: fomentar la colaboración interdisciplinar

- Formación HPC:
  - **CESGA Computational Science Summer School** (140 horas, 3ª edición en 2011): http://www.cesga.es/gl/soporte_usuarios/formacion/SummerSchool2011
  - **Máster Interuniversitario en Computación de Altas Prestaciones** (Máster oficial, 60 ECTS, 2ª edición en 2011/12): http://gac.udc.es/master

“Computación de Altas Prestaciones: Paradigmas y Bibliotecas de Programación basadas en Software Libre”

JORNADAS SOBRE COMPUTACIÓN DE ALTAS PRESTACIONES Y SOFTWARE LIBRE

Ourense, 5 de Octubre de 2011

JUAN TOURIÑO DOMÍNGUEZ

Dpto. Electrónica y Sistemas

Universidad de A Coruña

http://www.des.udc.es/~juan