Multiprocessors

- **One system --- many processors**
  - parallelism

Paralelismo

- **Expectativa**
  - más currantes
    - ✓ menos tiempo
    - ✓ más trabajo en el mismo tiempo

….. lineal

Parallelism

- **Reality**
  - Work partitioning: sometimes
    - ✓ Lack of work: **dependencies** (sequential is an extreme case)
      - – Communication / synchronization
    - ✓ Hard to equipartition: **Load imbalance**
    - ✓ **replication**
  - Overhead
    - ✓ Work generation
    - ✓ Communication: volume/surface ratio
    - ✓ synchronization

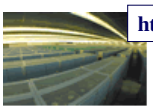
Parallelism

- **Result**
  - Team delivers more than one individual
  - **Not always linear**
  - Sometimes (few) more than the sum of the parts

Multiprocesadores (2004)

- **Desde los dinosaurios**

http://www.top500.org/



EARTH SIMULATOR
Earth Simulator Center
Kanazawa
NEC
Rmax: 35.86 TFlops

ASCI WHITE
LLNL
Livermore
IBM SP Power3
Rmax: 7.22 TFlops

TERASCALE SYSTEM
Pittsburgh
Supercomputer Ctr.
HP AlphaServer
Rmax: 4.46 TFlops

TERA
CEA
Bruyeres-le-Chatel
HP AlphaServer SC
Rmax: 3.98 TFlops

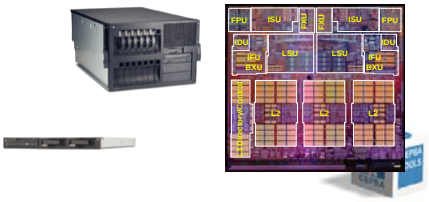
SP POWER3
NERSC/LBNL
Berkely
IBM SP Power3
Rmax: 3.05 TFlops

- **Hasta los ratones (super)**

Multiprocesadores (2005)

- **Desde los dinosaurios**
  - … todos tuvieron sus 5 minutos de gloria

- **Hasta los ratones (super)**
  - …

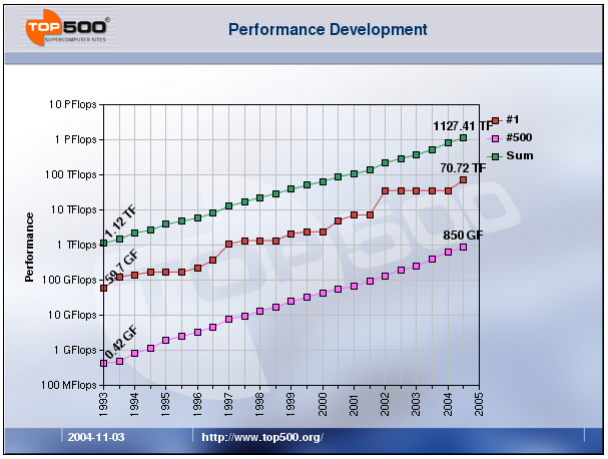


Multiprocessors (2008)

- **What changed?**
  - From elite: still there
  - To “commodity”
- **Why?**
  - Necessity: Power

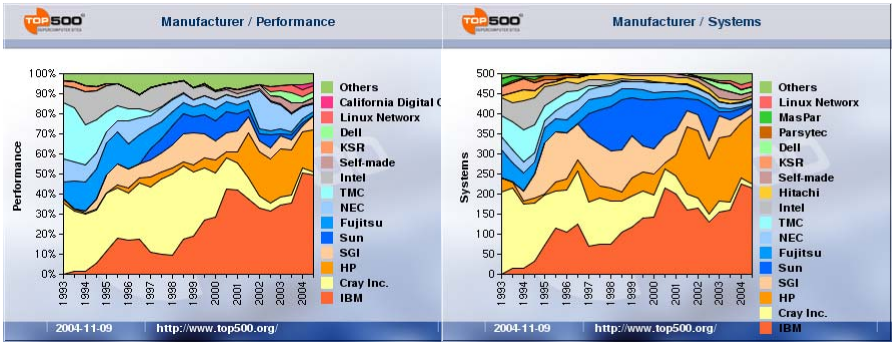
Multiprocessors

- **TOP 500**
  - http://www.top500.org/



Multiprocessors

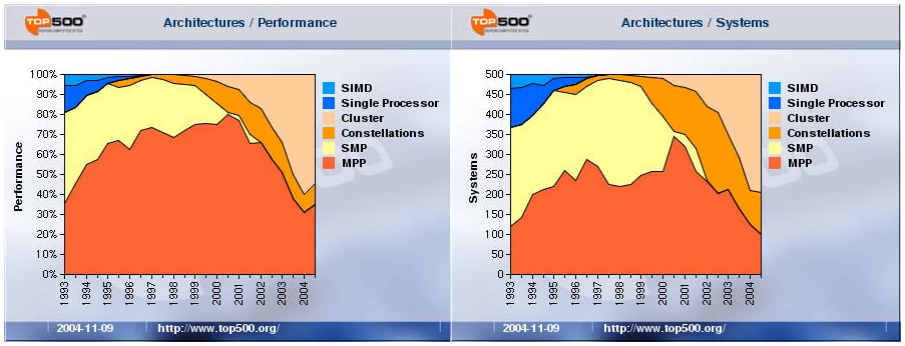
- **TOP 500**
  - http://www.top500.org/ by manufacturer



Jesús Labarta, MP, 2008

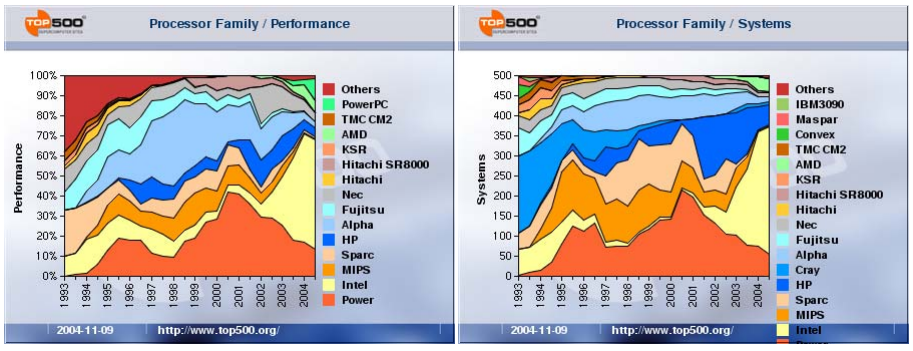
Multiprocessors

- **TOP 500**
  - http://www.top500.org/ by architecture



Multiprocessors

- **TOP 500**
  - http://www.top500.org/ by processor



Scaling models

- **Scaling:**
  - Evolution of performance indices when increasing
    - ✓ Problem size
    - ✓ Processors
  - Correlation on how to increase them?

Scaling models

- **Models:**
  - Problem Constrained scaling
    - ✓ Fixed problem size

$$Speedup_{PC}(p) = \frac{Time(1)}{Time(p)}$$

  - Time-constrained scaling
    - ✓ Keep constant total execution time

$$Speedup_{TC}(p) = \frac{Work(p)}{Work(1)}$$

  - Memory–constrained scaling
    - ✓ Keep constant memory used per processor

$$Speedup_{MC}(p) = \frac{IncreasedWork}{IncreasedTime}$$

Parallelism: useful

- **Capability computing**
  - Big problems
    - ✓ Not just time (CPUs)
    - ✓ Also Memory, I/O,...
- **Capacity computing:**
  - Throughput. Many problems

Multiprocesadores

- **Que tenemos nosotros (2004)**
  - SGI Origin2000:
    - ✓ 64 MIPS R10000
  - IBM SP:
    - ✓ : 128 Power3
    - ✓ : 32 Power4
  - HP (compaq):
    - ✓ : 12 Alpha 21264
    - ✓ : 16 Alpha 21264
  - Cluster
    - ✓ 64 Pentiums III
    - ✓ 16 Pentiums II
  - SMPs pequeño
    - ✓ 4 Pentiums
    - ✓ 4 Itaniums

…. no estamos en el mapa **…pero se puede hacer mucho**

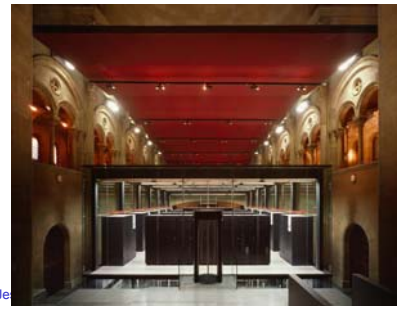
Multiprocesadores

- **Que tenemos nosotros (2005)**
  - Mare Nostrum:
    - ✓ ~5200 PPC970

…. cuartos en la foto (Nov 2004)

… octavos (Nov 2005)

…

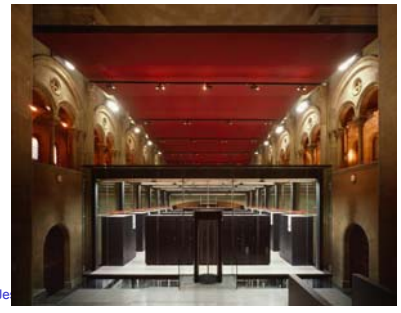


Multiprocessors

- **Que tenemos nosotros (2008)**
  - Mare Nostrum (2006)
    - ✓ ~10400 cores PPC970MX
  - Altix:
    - ✓ ~128 cores Itanium

…. 13th in the photo (Nov 2007)

…



Multiprocessors

- **Our resources (2010) ??**
  - Mare Incognito:
    - ✓ ~ 600000 cores ??
    - ✓ Cell ??
    - ✓ ????

…. ???? in the photo (2011)

…

Mare Incognito

"Homogeneous" supercomputer based on the Cell processor

We believe it is possible to build

"cheap", "efficient", "not specific" and

"widely applicable"

We know it is risky

We have "a vision" of some relevant technologies to develop/ push forward

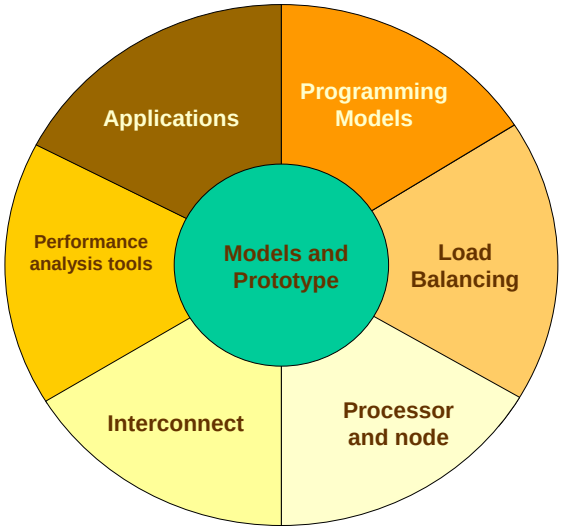
Many of them are not Cell specific and will be evaluated for other architectures

An Opportunity

Integrate many BSC research lines and increase cooperation IBM

Influence the design and use of supercomputers in the future.

Project structure



Performance models

- **Model**
  - Relationship between
    - ✓ System parameters
    - ✓ Performance metrics
  - Useful
    - ✓ Predict
      - – Expectations, Reference
    - ✓ Understand behavior

Not necessarily bad when a model does not match reality…

… indication of unaccounted factor, not understood phenomena.

Performance models

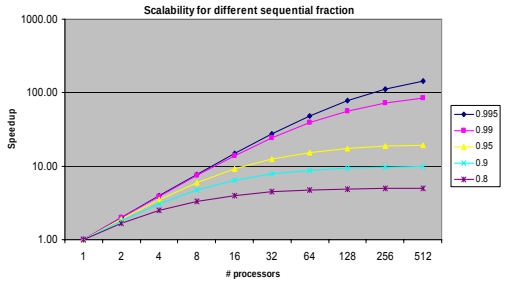
- **Performance metrics**
  - Time T(p)
    - ✓ Speed-up S(p) = T(1)/T(p)
    - ✓ Efficiency η(p) = S(p)/p
  - Throughput ??/T ??∈{jobs, iterations, problems...}
- **Other metrics**
  - Price/performance
    - ✓ MFLOP/$
    - ✓ MFLOP/Watt
    - ✓ MFLOP/m²

Performance models

- **Amdahl's law**
  - T(1) = Tseq + Tpar
  - T(P) = Tseq + Tpar/P
  - S(P) = 1/((1-f)+f/P)
    - ✓ f=Tpar/T(1) parallel fraction
  - Lo que no puede ser no puede ser y además es imposible

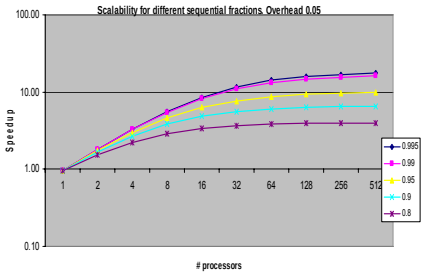
…. What is not parallelized ends up being the blottleneck

Speed-up: Amdahl's Law

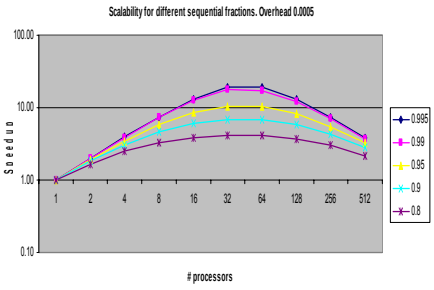


$$S(p) = 1 / (1 - f + (f / p))$$

Speed-up: Overheads



$$S(p) = 1 / (1 - f + (f / p)+o)$$



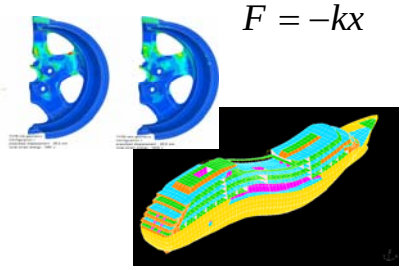
$$S(p) = 1 / (1 - f + (f / p)+o*p)$$

Laws of …

- Science

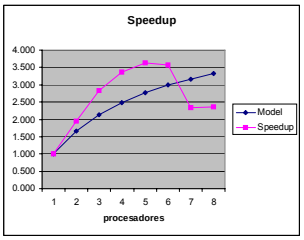
$$F = ma$$

$$F = -kx$$



- Computer Science

$$S = \frac{1}{1 - f + \frac{f}{P}}$$

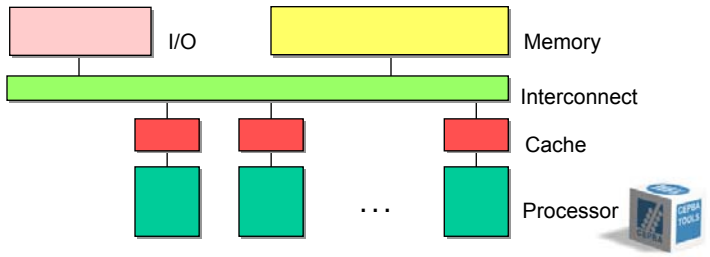


Architectures

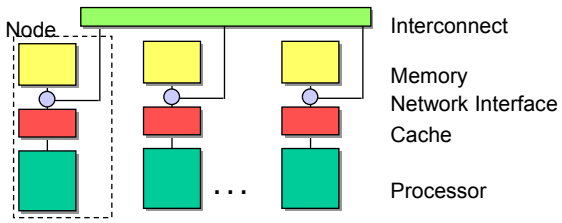
- **Shared Memory**
  - SMP
  - NUMA
- **Distributed Memory**
- **Hierarchical mix**
  - SMP nodes @ Distributed memory

Shared Memory

- **SMP (symmetric multiprocessor):**
  - UMA
  - Issues
    - ✓ Interconnection network:
      - – Aggregated bandwidth: bus, crossbar.
      - – Latency
    - ✓ Cache: coherency, consistency

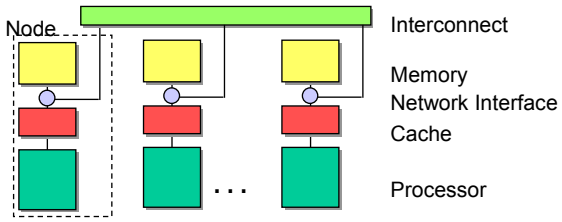


Distributed Shared Memory



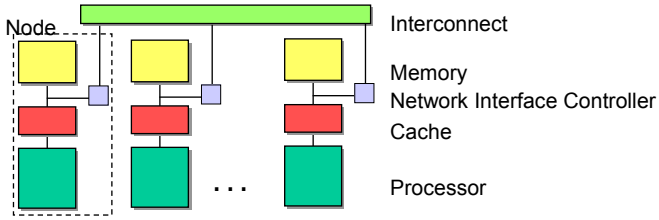
- **Load/stores: local/remote depending on physical address**
- **NUMA (non-uniform memory access): remote accesses may take 5-10 times longer**

Distributed Shared Memory



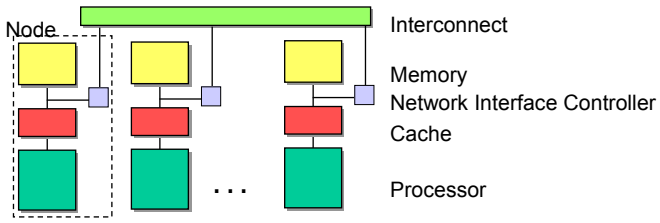
- **Issues:**
  - Interconnection network: latency, topology, bisection bandwidth
  - Cache: coherency, consistency !!
  - Locality of access

Distributed Memory



- **Load/stores: local**
- **I/O: send/receive (local address + node addressing mechanism)**
- **Private / commodity networks**

Distributed Memory



- **Issues**
  - Interconnection network:
    - ✓ latency, topology, bisection bandwidth
    - ✓ Injection mechanism
- **SMP nodes**

Hierarchical

- **Distributed memory**
- **Each node shared memory**
  - ~ SMP several chips
- **Each chip multicore/manycore**
- **Homogeneous/heterogeneous**

Modelos de programación

- **Mecanismos disponibles al programador para expresar la estructura lógica de un programa**

- **Influye**
  - Complejidad del programa
    - ✓ Costo de desarrollo
    - ✓ Legibilidad. Costo de mantenimiento
  - Rendimiento
    - ✓ Influenciado por el modelo
    - ✓ por la implementación del modelo
    - ✓ Por la estructura de paralelización

Back to Babel?

Book of Genesis

"Now the whole earth had one language and the same words" …

…"Come, let us make bricks, and burn them thoroughly. "…

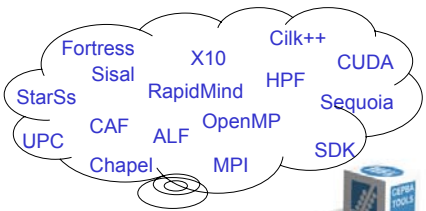
…"Come, let us build ourselves a city, and a tower with its top in the heavens, and let us make a name for ourselves"…

And the LORD said, "Look, they are one people, and they have all one language; and this is only the beginning of what they will do; nothing that they propose to do will now be impossible for them. Come, let us go down, and confuse their language there, so that they will not understand one another's speech."

The computer age

Fortran & MPI

++



Back to Babel?

- **Result?**
  - Dispersion
  - Pain
  - Thousands of years from the first confusion of tongues to the second

- **Current situation**
  - We are at a CrossRoad, Revolution, Crisis,…

THE problem:
How to program these machines!!!!

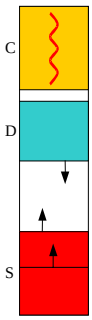
Modelos de programación

- **Componentes**
  - Datos
  - Procesos
  - Comunicación
  - Sincronización
  - Entrada/salida

Memory

- **is THE problem**
  - 16 → 32 → 64 bits:
    - ✓ A simple thing for a computer architect
    - ✓ has caused huge pain for everybody !!!!
- **Some issues**
  - Addressability
  - Association
  - Consistency

Secuencial



Variable globales: 

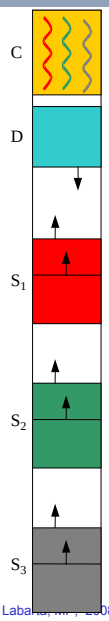
Variables dinámicas(locales): 
Visibilidad local
Se puede pasar puntero al llamar funciones
Vida limitada

Librerias:

malloc: 
Visible global

Problemas:
Desbordamiento pilas
Errores difíciles detectar (efecto variable y lejano)

Memoria compartida: espacio @



Variable globales:
Unica copia

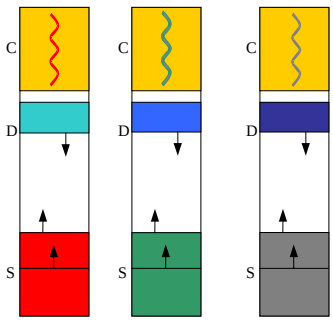
Variables dinámicas(locales):
Acceso normal: privadas
Accesibles globalmente (pasar puntero)

Librerias:
Exclusión mutua

malloc:
Visible global

Problemas:
Desbordamiento pilas
Errores difíciles detectar (efecto variable y lejano)

Memoria Distribuida: espacio @



Variables globales:
Replicadas: misma dirección lógica
mantener consistencia
Distribuidas: misma dirección, distinto objeto

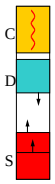
Variables dinámicas(locales):
privadas (no posibilidad compartir)
~ mismas direcciones lógicas

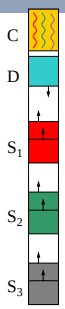
Librerías:
Locales.

malloc: local
~ mismas direcciones lógicas

Address spaces in parallel programming

Sequential



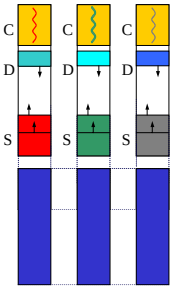


Shared Memory

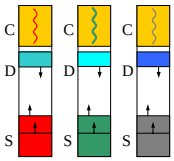
Allows for incremental Parallelization

SM architectures may better support fine grain sharing and synchronization

PGAS



Allows for separation between data transfers and synchronization



Distributed Memory

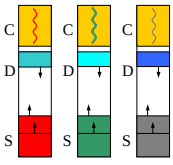
Need data transfers

Forces detailed analysis, explicit choices for everything

DM architectures can more easily support more memory bandwidth than SM architectures

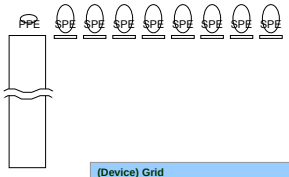
Jesús Labarta, MP, 2008

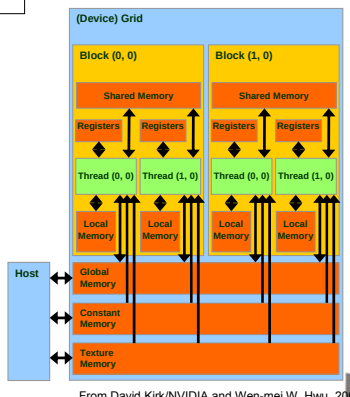
Address spaces in multicores



Homogeneous in structure

Heterogeneous





From David Kirk/NVIDIA and Wen-mei W. Hwu, 2007

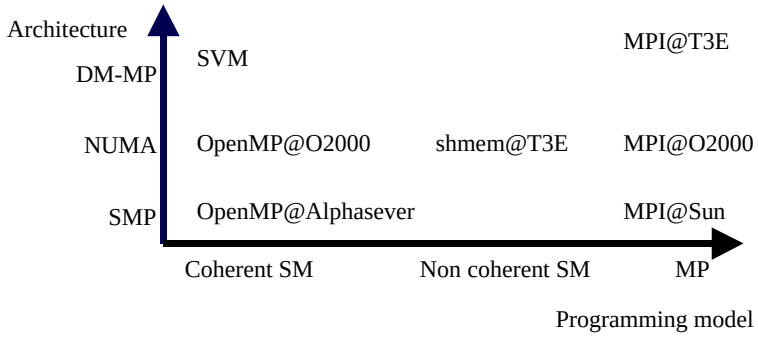
How much visible to the programmer??

Jesús Labarta, MP, 2008

Modelos

- **Memoria compartida**
  - Pthreads
  - OpenMP
    - ✓ http://www.openmp.org
    - ✓ http://www.compunity.org
- **Memoria distribuida**
  - MPI
    - ✓ http://www-unix.mcs.anl.gov/mpi/

Espectro



Paralelización

- **Fases**
  - Descomposición
  - Asignación
  - Orquestación
  - Mapeo

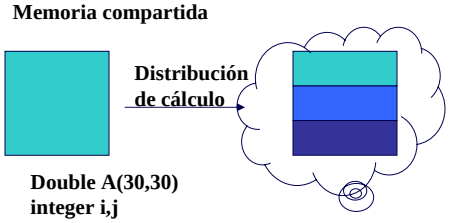
Algorítmico, ideas

Programación, sistema

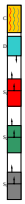
- **Aspectos básicos de programación en los distintos modelos**

Pthreads: programación

Memoria compartida



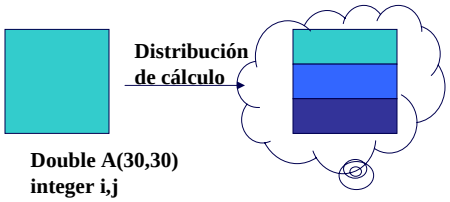

```
Double A(30,30)
integer i,j

Do j = 1,30
  Do i =1,30
    A(i,j) = 2* A(i,j)
    A(i,j) = j* A(i,j)
    A(i,j) = i* A(i,j)
```



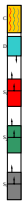
```
Double A(30,30)
interger thid, j

main() {
  Do j = 1,30
    Do thid=1,num_threads
      iinf = f(thid)
      isup = f(thid)
      create_thread(Loop_body,iinf,isup)
}

Loop_body(iinf,isup) {
  Do i =iinf,isup
    A(i,j) = 2* A(i,j)
    A(i,j) = j* A(i,j)
    A(i,j) = i* A(i,j)
}
```

OpenMP: programación

Secuencial

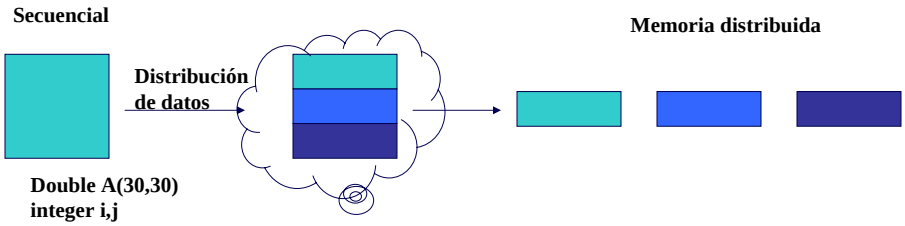



```
Double A(30,30)
integer i,j

Do j = 1,30
  Do i =1,30
    A(i,j) = 2* A(i,j)
    A(i,j) = j* A(i,j)
    A(i,j) = i* A(i,j)
```



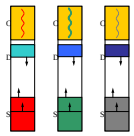
```
Double A(30,30)
interger i,j

  Do j = 1,30
C$OMP  PARALLEL DO
C$OMP  PRIVATE (i)
    Do i =1,30
      A(i,j) = 2* A(i,j)
      A(i,j) = j* A(i,j)
      A(i,j) = i* A(i,j)
```

MPI: programación




```
Double A(30,30)
integer i,j

Do j = 1,30
  Do i =1,30
    A(i,j) = 2* A(i,j)
    A(i,j) = j* A(i,j)
    A(i,j) = i* A(i,j)
```

```
Double A(10,30)

myrank = quien_soy()
Do j = 1,30
  Do li =1,10
    i=li+myrank*10
    A(li,j) = 2* A(li,j)
    A(li,j) = j* A(li,j)
    A(li,j) = i* A(li,j)
```

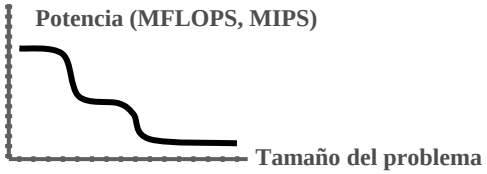


Economía

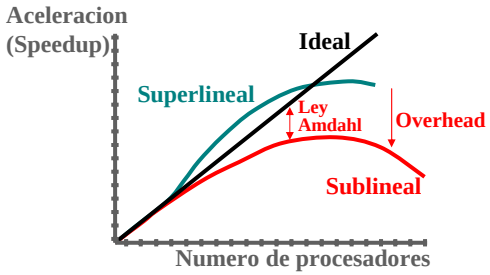
- **Precio / rendimiento**
- **Costes**
  - Adquisición
  - Mantenimiento
  - Espacio y consumo
  - Gestión
    - ✓ systems

Que se puede esperar?

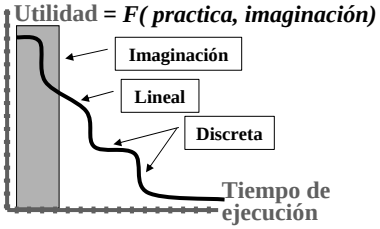
1 Procesador

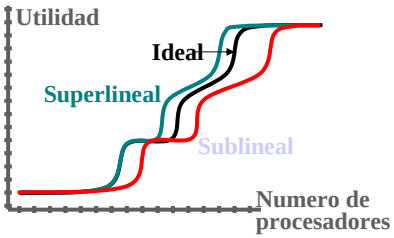


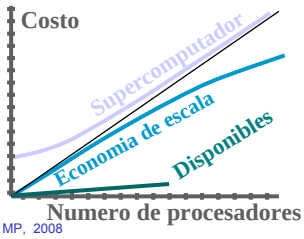
Paralelismo

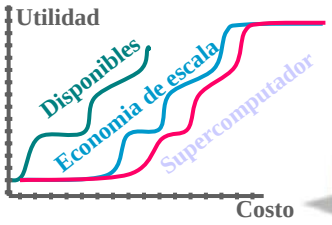


Que se puede esperar?







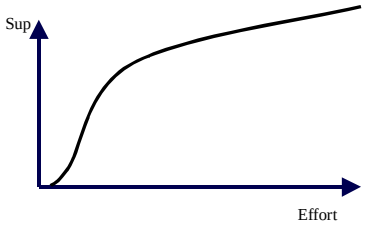


Economía

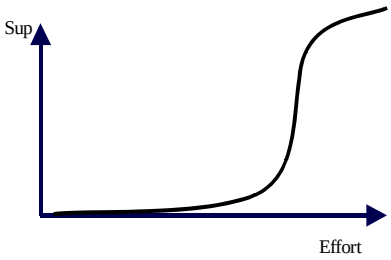
- **Necesidad:**
  - Usuario $\cong$ gas
    - ✓ PV=nRT
      - – P $\propto$ 1/ V $\Rightarrow$ Solo si esta muy comprimido empuja de verdad
  - Usuario $\neq$ gas
    - – Una vez expandido es difícil volver atrás
  - Tendencias
    - ✓ COTS: procesadores, redes, software

Economía: desarrollo

- **Memoria comparida**



- **Memoria distribuida**



Herramientas

- **Analizar una aplicación (sweep3d)**
  - Secuencial
  - Paralela
- **Profile**
  - perfex
  - ssrun
- **Traceo**
  - Paraver
  - Dimemas
  - Paradyn

Herramientas

- **Analizar rendimiento de aplicaciones**
  - Donde están sus cuellos de botella
  - Información para optimizarla
- **Fases:**
  - Adquisición datos
  - Análisis

Adquisición de datos

- **Instrumentación**
  - Qué:
    - ✓ código usuario
    - ✓ librería
    - ✓ sistema
  - Quien:
    - ✓ manual
    - ✓ compilador
    - ✓ automático
  - Método:
    - ✓ traceo / muestreo (correlado con …)
    - ✓ Post-mortem / on-line

Adquisición de datos

- **A tener en cuenta**
  - Perturbación de la ejecución:
    - ✓ Causa: tiempo instrumentación, polución cache, almacenamiento,…
    - ✓ Minimización: implementación, soporte hardware
    - ✓ Efecto: conocerlo. Vivir con el.
  - Tamaño datos:
    - ✓ Problema almacenamiento y postproceso
    - ✓ Control de la instrumentación: manual, automático
    - ✓ Procesado on-line
      - – Overhead
      - – Reducción información

Análisis

- **Objetivo**
  - Transmitir información al usuario: maximizar
- **Enfoque**
  - Percepción
  - Cuantificación
- **Presentación**
  - Qué: magnitudes => vista
    - ✓ actividad temporal, fallos de cache, rutina, …
  - Cómo: modelo de presentación
    - ✓ Textual
    - ✓ Gráfico
    - ✓ sonoro,...

Aplicaciones

- **Solver iterativo**
- **Radix sort**
- **LU**