APLICACIONES CIENTÍFICAS, LIBRERÍAS Y COMPILADORES

CESGA

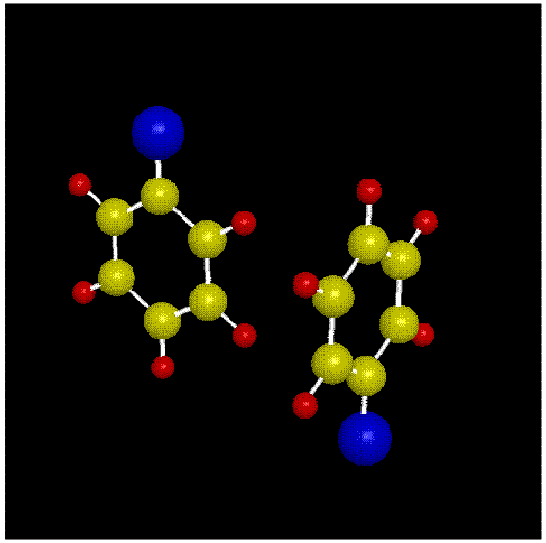
Dr. Andrés Gómez Tato
Adm. Aplicaciones y Proyectos

agomez@cesga.es

CENTRO DE SUPERCOMPUTACIÓN DE GALICIA

CONTENIDOS

- Aplicaciones en el CESGA
- Librerías de cálculo
- Mejoras del rendimiento
- Compilación
- Conclusiones



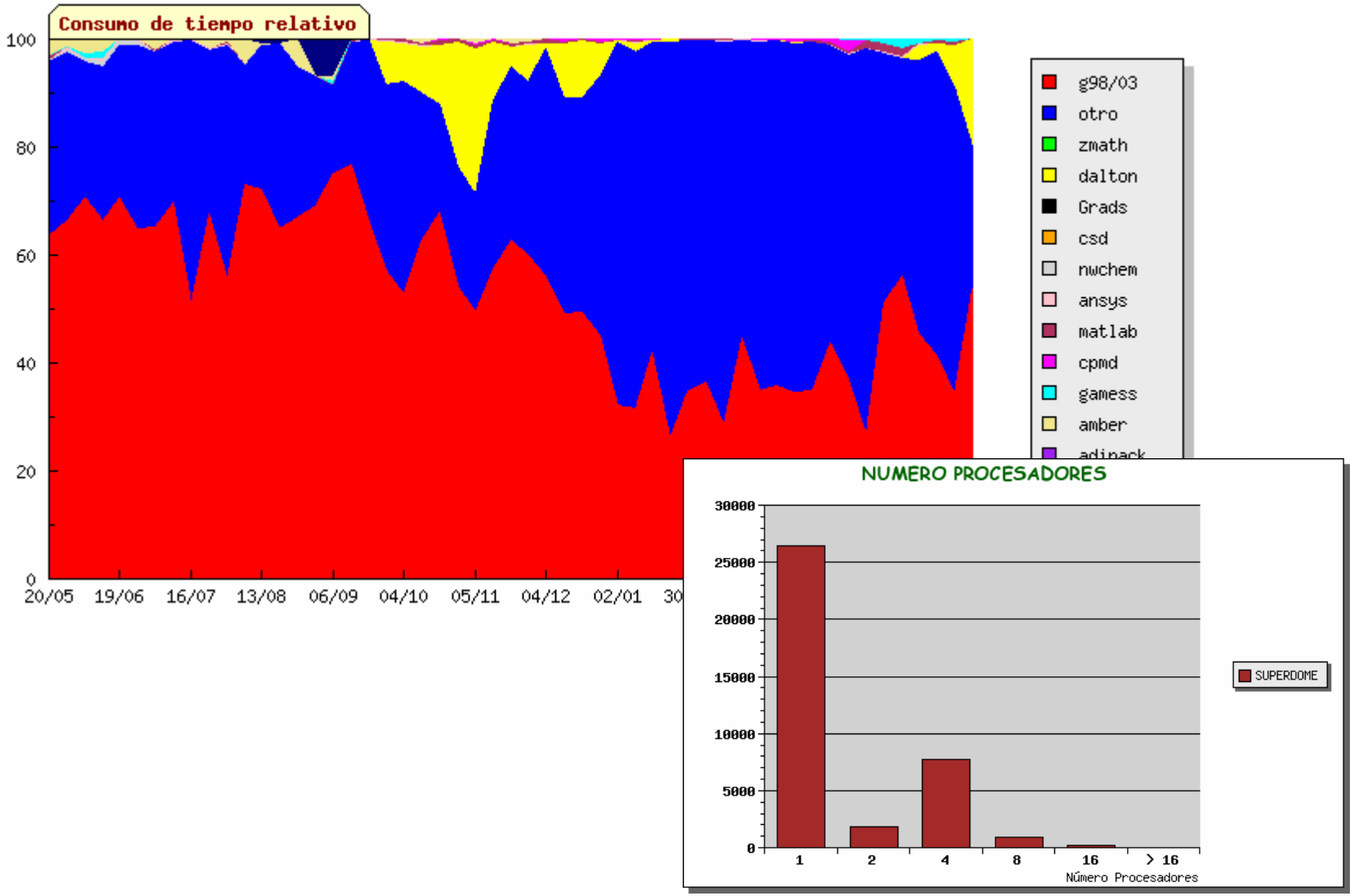
CESGA

Aplicaciones instaladas

Temas		SD	SVG	SC
Cálculo Estructural, Fluídos e Magnetismo	Ansys 5.7			X
Cálculo Molecular	Amber 8.0	X	X	X
	Gaussian 98	X	X	X
	Gaussian 03	X	X	X
	Dalton	X	X	X
	CPMD	X	X	X
	GAMESS	X	X	X
	Molden	X		X
	NWCHEM	X	X	X
	GROMACS	X	X	X
Simulación	EGSnrc		X	
	Geant		X	
	SIMULINK		X	X
	MATLAB		X	X
Análise Científica	PAW, PAW++		X	
	ROOT		X	X
Bioinformática	BLAST	X		X
	ClustalW	X		
	Combiner	X	X	
	GeneHunter	X		
	Genscan	X	X	
	GlimmerM	X	X	
	MUMer	X	X	
	Phylip	X		
Visualización científica	GRADS	X		X
	NCAR			X
	NETCDF	X		X

CESGA

Uso Aplicaciones



CESGA

Librerías de Cálculo científico

Librerías de cálculo

		SD	SC	SVG
libm	Funciones matemáticas	X	X	X
BLAS 1	Operaciones Vector-Vector	X	X	X
BLAS 2	Operaciones Matriz-Vector	X	X	X
BLAS 3	Operaciones Matriz-Matriz	X	X	X
BLAS SPARSE	Matrices sparse	X	X	
LAPACK	Ecuaciones lineales, mínimos cuadrados, autovalores y descomposición valores singulares	X	X	X
FFT	Transformadas rápidas de Fourier	X	X	E
ScaLAPACK	LAPACK para memoria distribuida	X	X	X
SuperLU	Solución de sistemas de ecuaciones lineales sparse	X		
Metis	partición grafos, elementos finitos, ordenación de matrices sparse	X		

Utilizar siempre que se pueda librerías portadas por el fabricante.

CESGA

Librerías de cálculo: mult. de matrices

- Caso 1: **2.3 Gflops (800x800)**

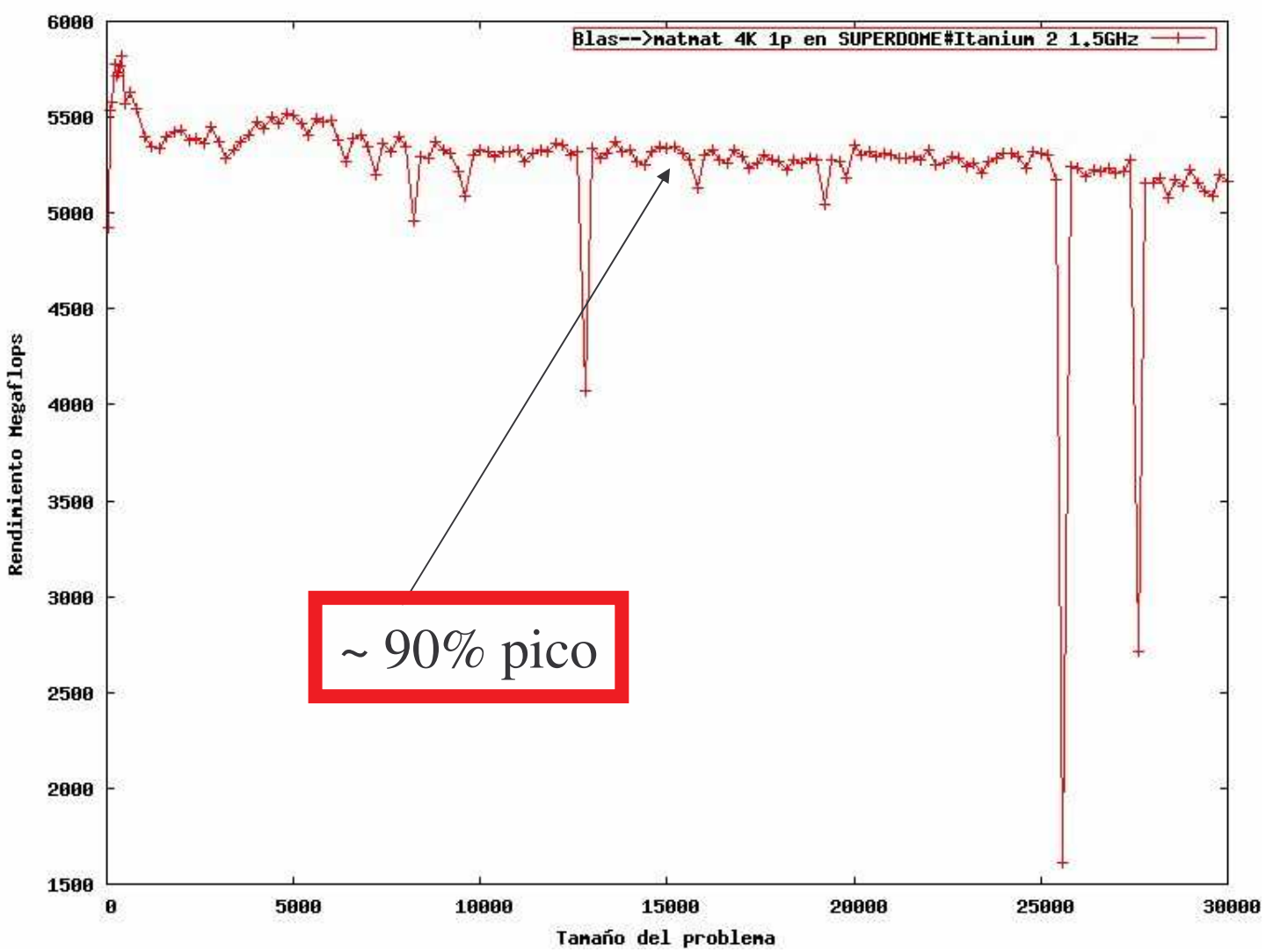
```
do j = 1, p
    do k = 1, n
          c(i,j) = c(i,j) + a(i,k) * b(k,j)
    enddo
enddo
```

- Caso 2: **5.7 Gflops (800x800)**

```
call dgemm('no transpose','no transpose',m,n,p,
     &          1.0d0, a, m, b, n, 0.0d0, c, m)
```

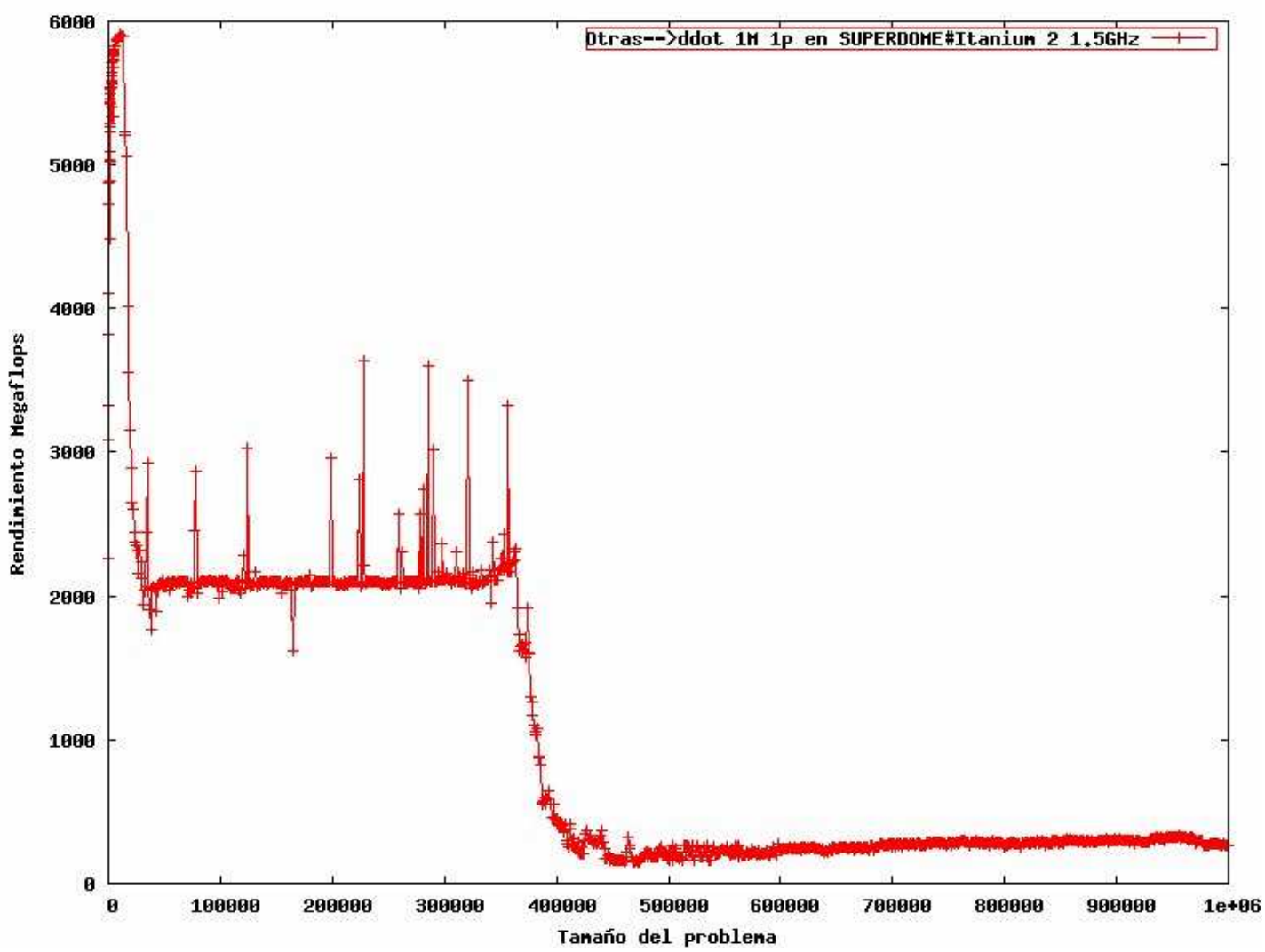
CESGA

Librerías de cálculo: mult. de matrices



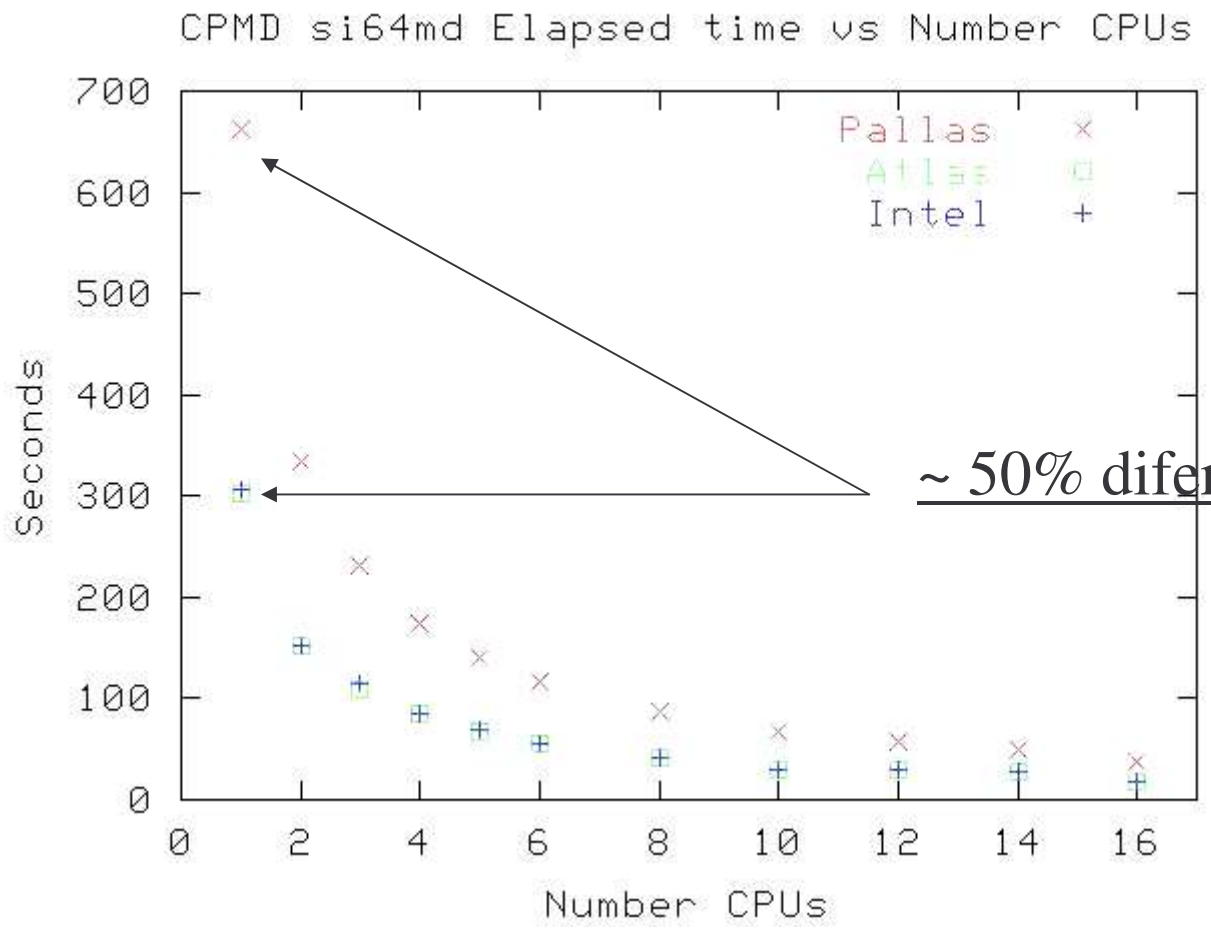
CESGA

Librerías de cálculo: DDOT 1.5GHz



CESGA

Librerías de cálculo: Ejemplo



~ 50% diferencia

CESGA


Ejecución y Rendimiento
CESGA

Ejecución y Rendimiento

A la hora de ejecutar hay que tener en cuenta:

1. Es paralela. Grado de paralelización

2. ¿Qué algoritmo es el más idóneo para mi problema?

3. ¿Qué puedo hacer para mejorar el rendimiento?

4. ¿Qué recursos necesito?

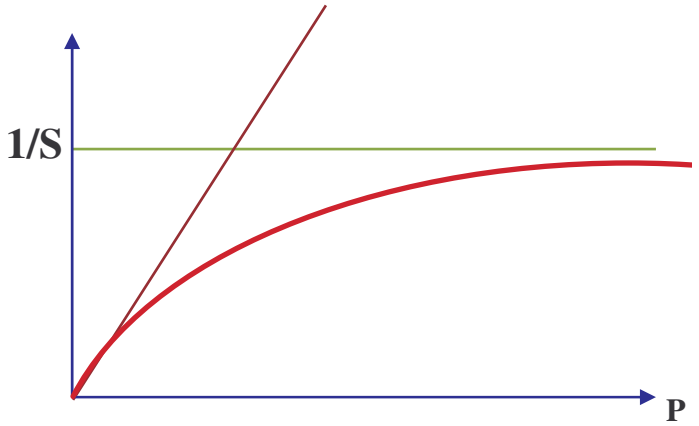
CESGA

Ejecución y Rendimiento: Paralelización

Speed-up= t_1/t_n

t_1 tiempo de ejecución 1 CPU

t_n tiempo ejecución con n CPUs



Ley de Amdahl:

s = fracción código en serie

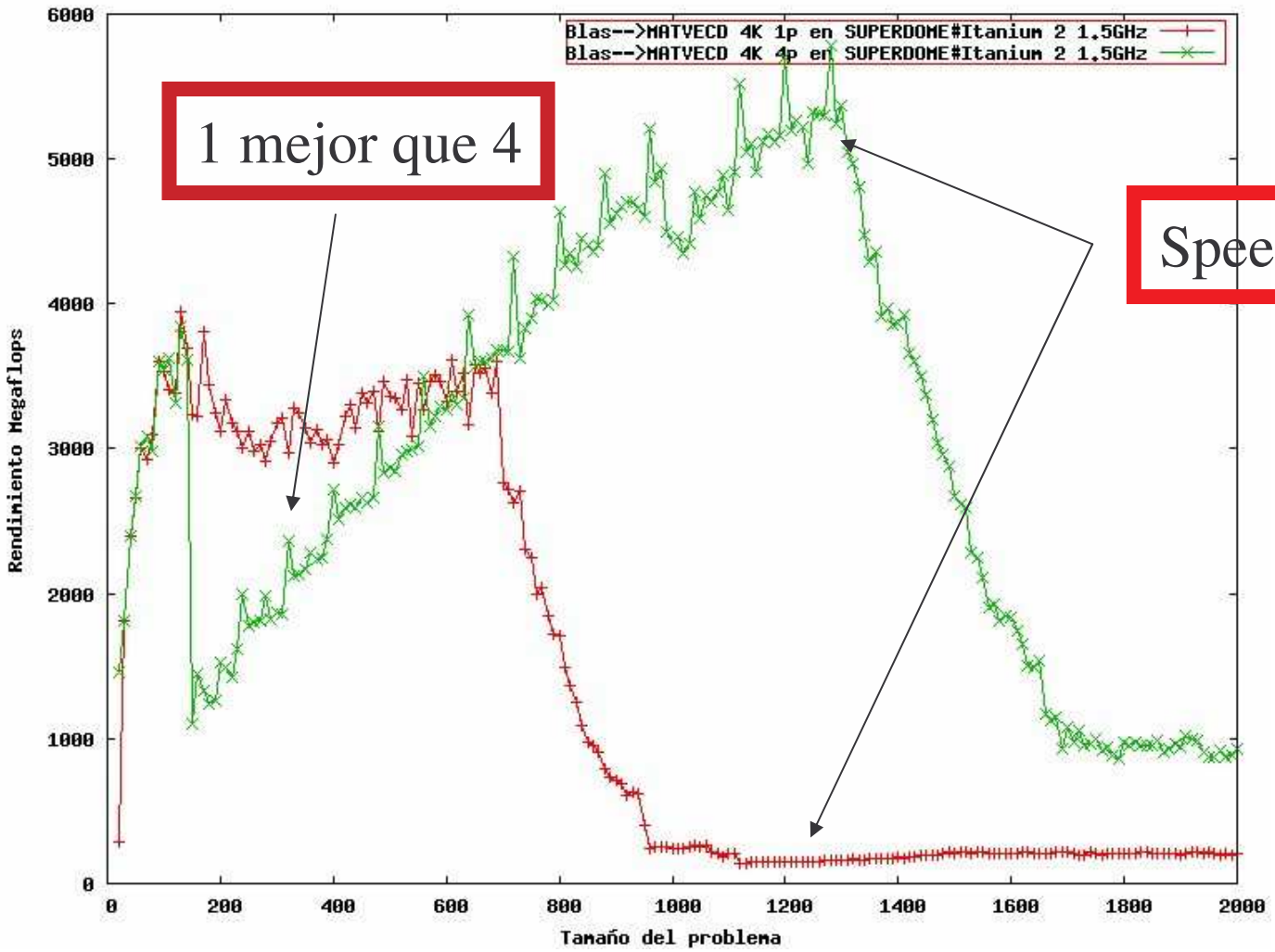
p = fracción código en paralelo

$t_1 = m$

$t_n = m * (s + p/n)$ ➡ $S = n/(s * n + p)$ ➡ $S(\infty) = 1/s$

CESGA

La ley de Amdahl no siempre es cierta



1 mejor que 4

Speed-up > 20!!

CESGA

Ejecución y Rendimiento: Paralelización

El grado de paralelización depende de:

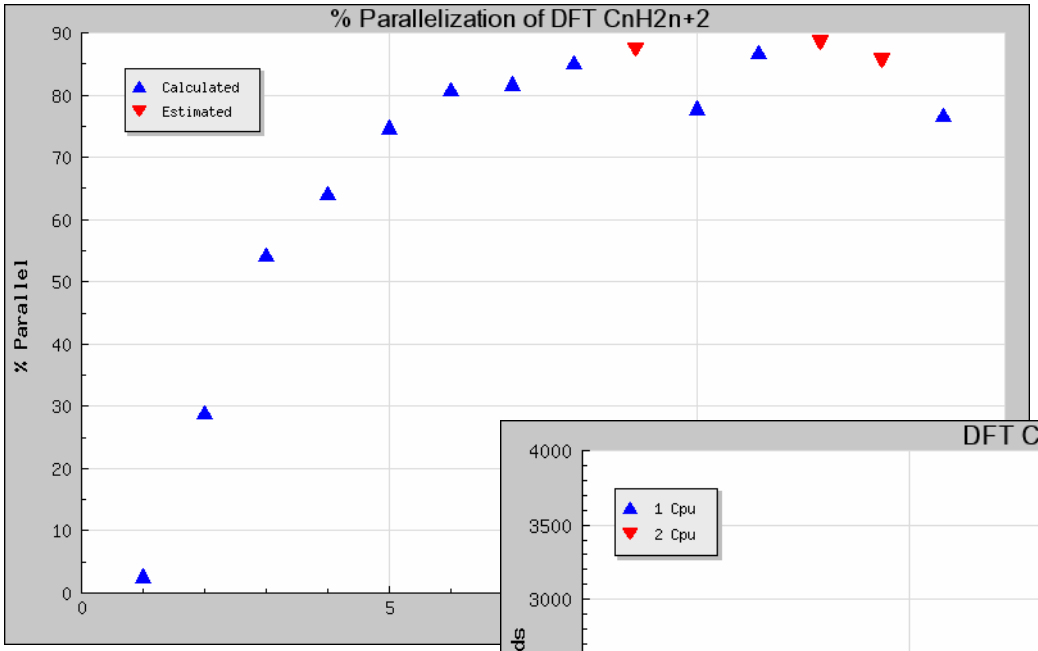
- El software
  - Fracción de código paralelo
  - Tecnología de paralelización: OpenMP, MPI, OpenMP+MPI
  - Etc.

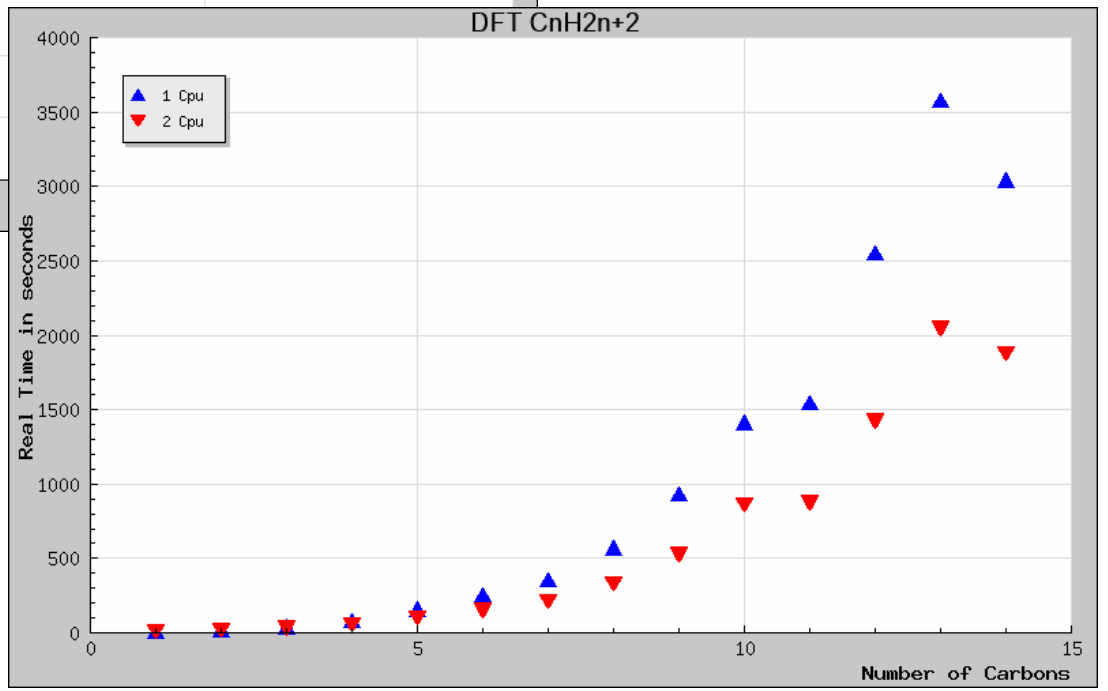
- El problema
  - Fracción de instrucciones paralelas

En general, MPI escala mejor que OpenMP

CESGA

Ejecución y Rendimiento: Paralelización





Ejemplo:

DFT C_nH_{2n+1}

CESGA

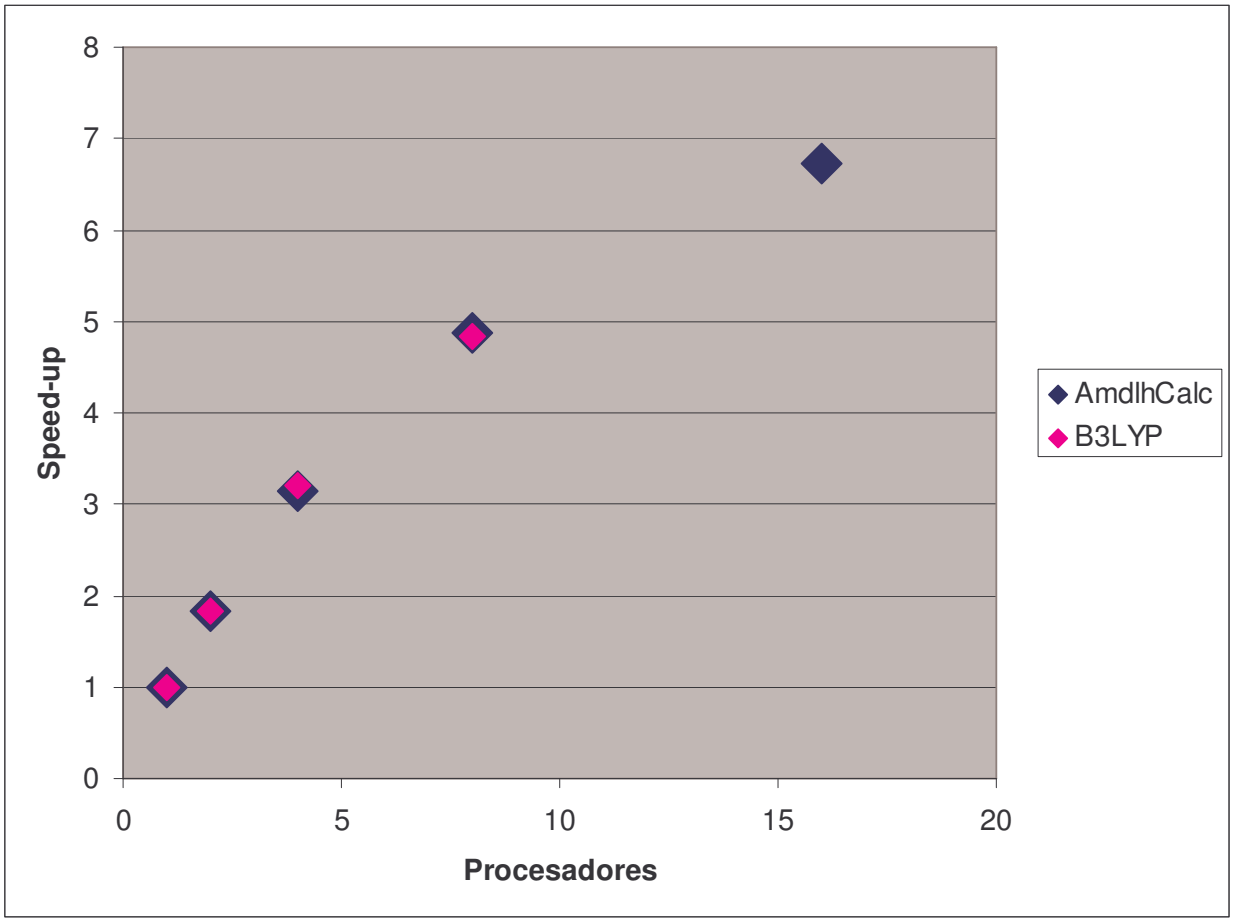
¿Cuántos procesadores utilizo?

1. Problema con la memoria: tantos como sea necesario para ejecutar el problema.

2. Problema de tiempo:
   - Calcular el fracción de código paralelo (t(1) y t(2))
   - Calcular la aceleración teórica
   - Generalmente seleccionar un número par

Ejemplo: http://perfsuite.ncsa.uiuc.edu/AmdahlCalc/

Ejecución y Rendimiento: Paralelización



Ejemplo: http://perfsuite.ncsa.uiuc.edu/AmdahlCalc/

CESGA

Ejecución y Rendimiento: Pagesize

- El HP Superdome soporta varios tamaños de página de memoria desde 4k hasta 4G
- Se puede seleccionar independientemente para datos y código
- Por defecto, al compilar con +Ofast se selecciona 4M
- La selección se realiza con el comando *chatr*
- En PIV están soportadas 4k, (2M) y 4M (en Linux, ver /usr/src/linux-2.4/Documentation/vm/hugetlbpage.txt)

CESGA

Ejecución y Rendimiento: Chatr (Solo SUPERDOME)

Permite cambiar propiedades del ejecutable. Por ejemplo:

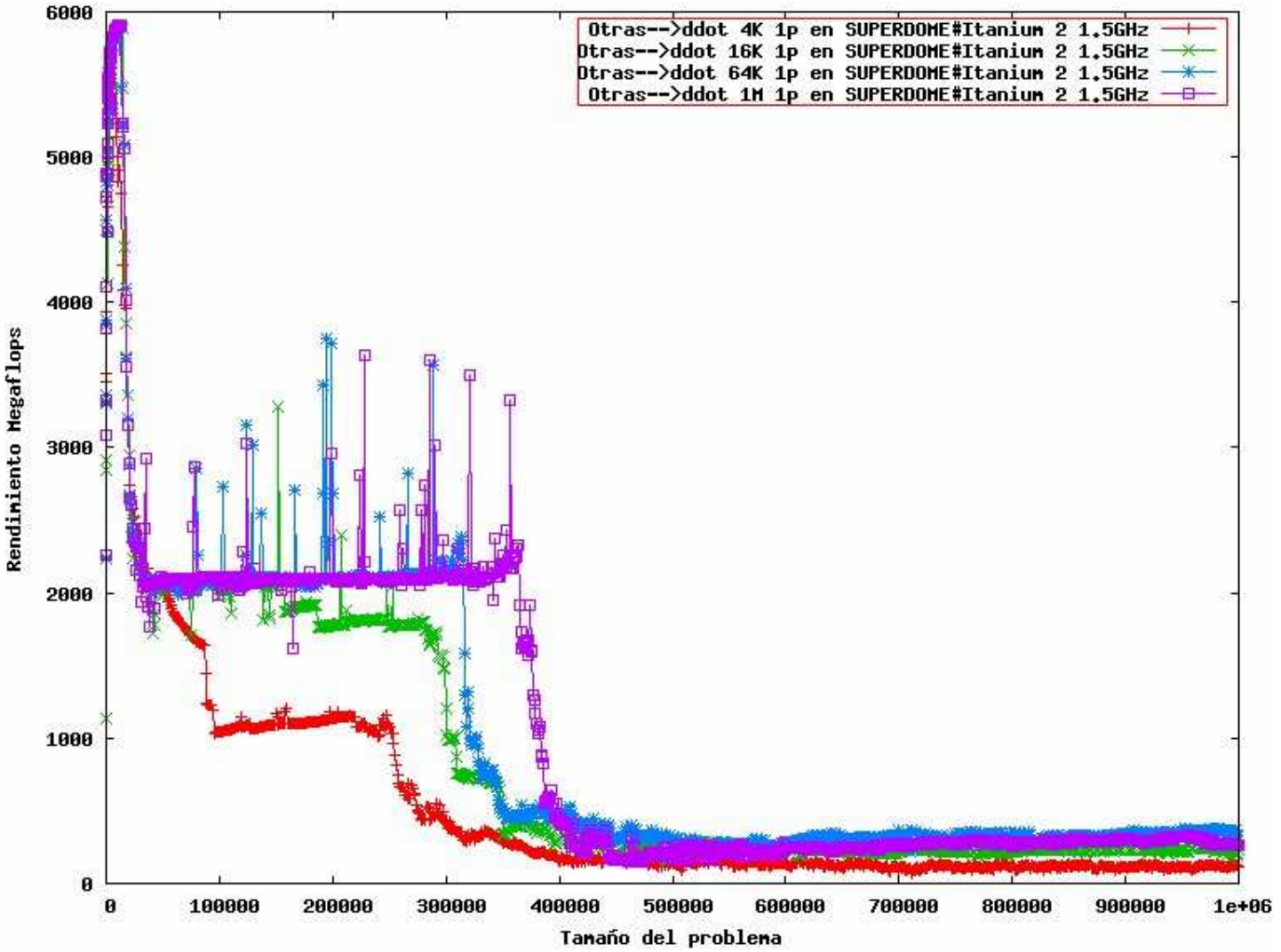
- **[+pd size]** cambia el tamaño de página de los segmentos de datos.
  - Posibles: D, L, 4K, 16K, 64K, 256K,1M, 4M, 16M, 64M, 256M, 1G y 4G.
  - Es una solicitud al S.O.. El valor final puede variar en el momento de la ejecución.
- **[+pi size]** Cambia el tamaño de página de los segmentos de código.

- **[+id flag]** Controla el tipo de memoria
  - Enable: memoria interleaved
  - Disable: memoria local a la celda.

- **[+mergeseg]** Controla si se juntan los segmentos de datos de las librerías dinámicas.

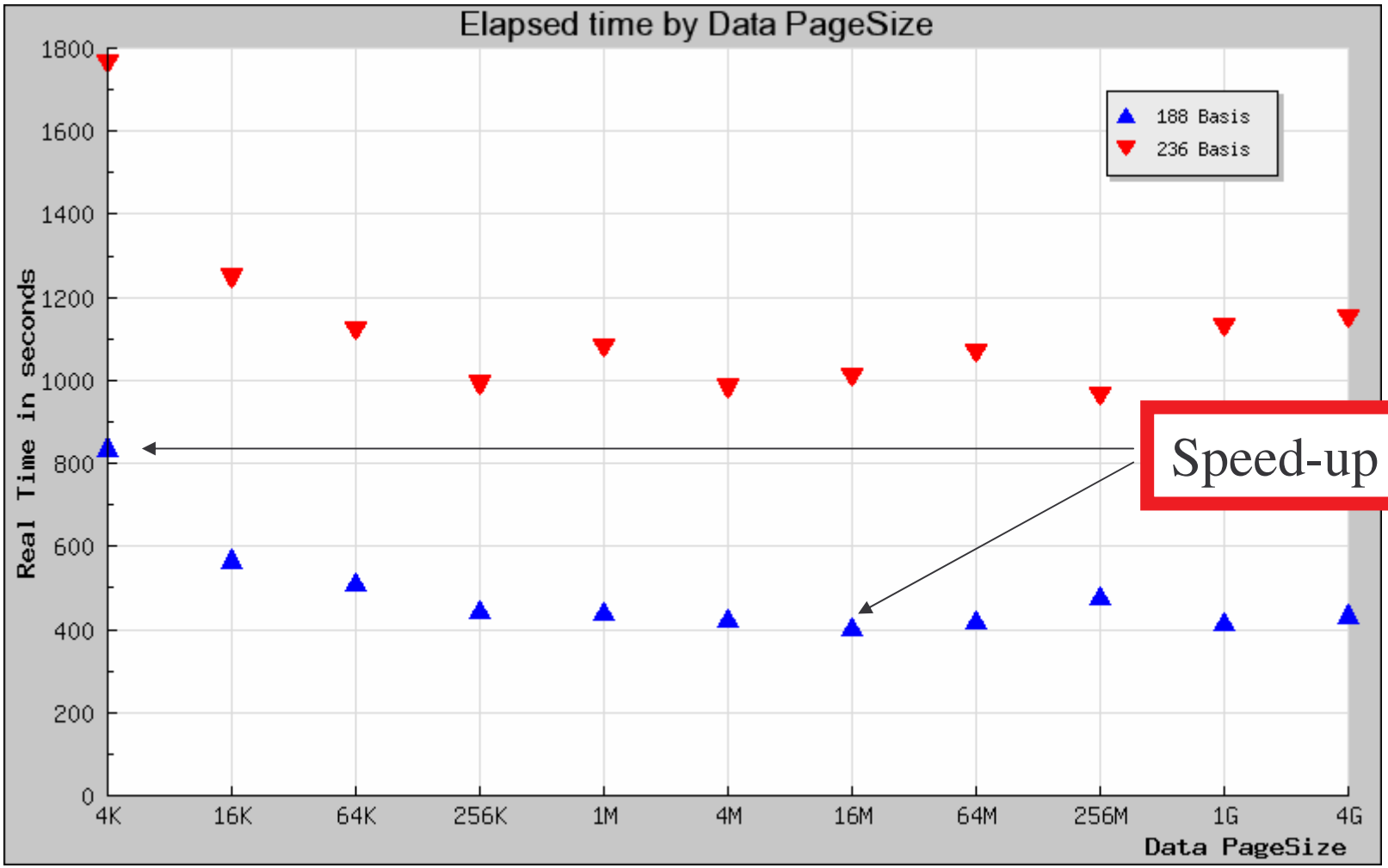
CESGA

Ejecución y Rendimiento: Pagesize



CESGA

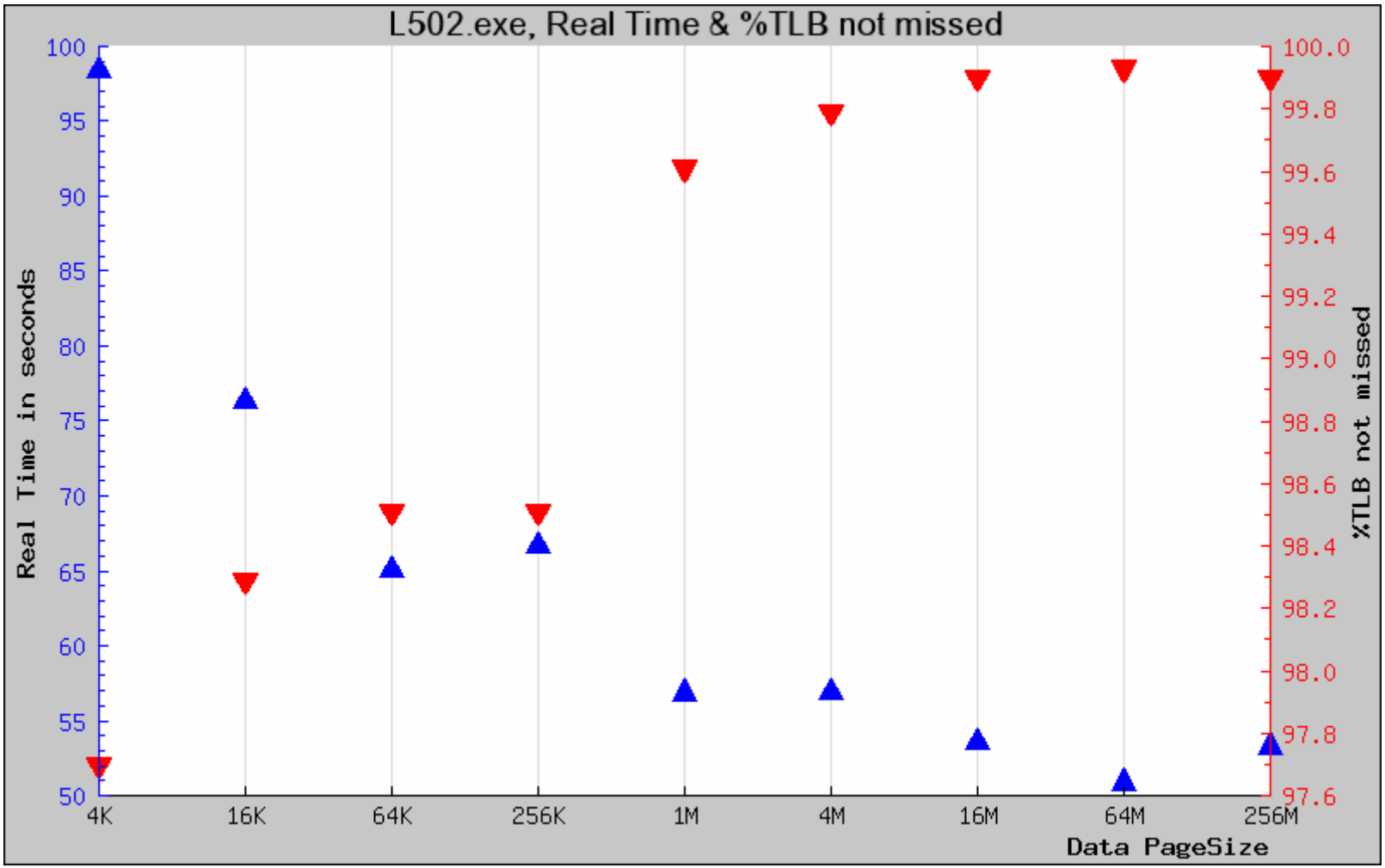
Ejecución y Rendimiento: Pagesize



Ejemplo cálculo en Gaussian Incore

CESGA

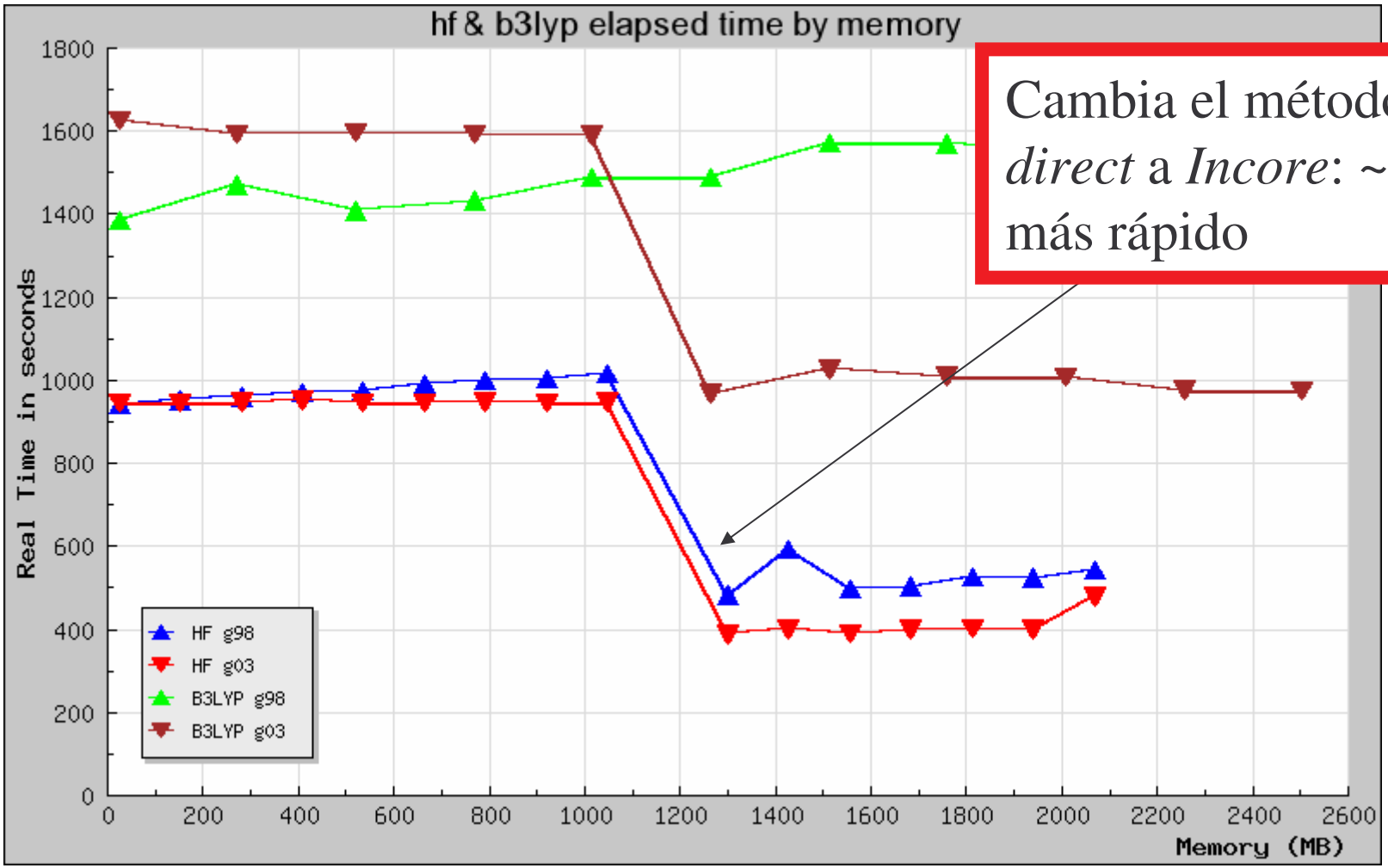
Ejecución y Rendimiento: Pagesize



Ejemplo cálculo en Gaussian Incore 256 bases

CESGA

Ejecución y Rendimiento: Método



Cambia el método de *direct* a *Incore*: ~50% más rápido

CESGA

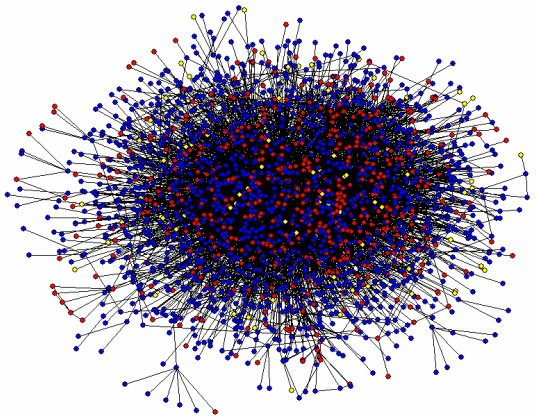
Ejecución y Rendimiento: Método

Cálculo centralidad de la red:

$$SC(i)=\sum_j [\gamma(i)]^2 \cdot e^{\lambda_j}$$

γ Autovectores

λ Autovalores



- Ineficiente:

```
for (i=1;i<N;i++)
{
        sc[i]=0.0;
        for (j=1;j<N;j++)
                sc[i]=sc[i]+gamma[i,j]*gamma[i,j]*exp(lambda[j]);

}
```

- Eficiente: $\Rightarrow SC=\Gamma^2 \cdot e^{\Lambda}$

```
dgemm("N","N",&M,&N,&L,&alpha,A,&LDA,EL,&LDB,&beta,C,&LDC,sizeof(char),sizeof(char));
dgemm("N","T",&M,&N,&L,&alpha,C,&LDA,A,&LDB,&beta,V2,&LDC,sizeof(char),sizeof(char));
```

CESGA

Ejecución y Rendimiento: Método

$$SC(i) = \sum_j [\gamma_j(i)]^2 \cdot e^{\lambda_j} \quad \Rightarrow \quad SC = \Gamma^2 \cdot e^{\Lambda}$$

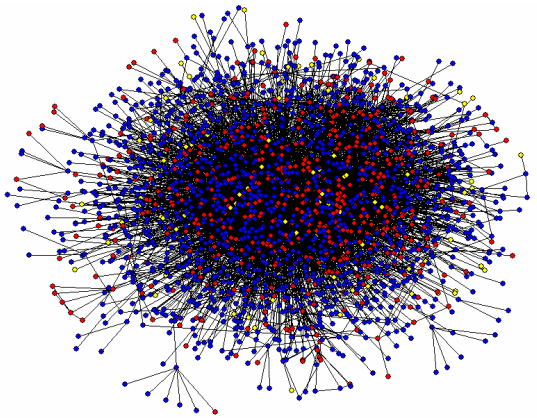
$$SC_{odd}(i) = \sum_j [\gamma_j(i)]^2 \cdot \sinh(\lambda_j) \quad \Rightarrow \quad SC_{odd} = \Gamma^2 \cdot \sinh(\Lambda)$$

$$SC_{even}(i) = \sum_j [\gamma_j(i)]^2 \cdot \cosh(\lambda_j) \quad \Rightarrow \quad SC_{even} = \Gamma^2 \cdot \cosh(\Lambda)$$

$$S = -\sum_i e_1(i) \cdot \log_2 e_1(i) \quad \Rightarrow \quad S = -E_1 \cdot \log_2(E_1^T)$$

$$C(p,q) = \sum_j \gamma_j(p) \cdot \gamma_j(q) \cdot e^{\lambda_j} \quad \Rightarrow \quad C = \Gamma \cdot e^{\Lambda} \cdot \Gamma^T$$

$$C_{conn}(G) = \frac{1}{2 \cdot N_{p,q}} \sum_j e^{\lambda_j} (v_j A v_j^T)$$



+100 Horas → menos de 5 minutos

CESGA

Compilación

Compiladores

	SD	SC	SVG
F77	f90	f77	pgf77/ifc/g77
F90	f90	f90	pgf90/ifc
F95		f95	ifc
HPF		f90 -hpf	pghpf
C89	c89	cc	pgcc/gcc/icc
C99	cc +AC99		
C++	aCC	cxx	pgCC/g++/icc

CESGA

COMPILACIÓN: ¿32 O 64 BITS?

- 64 bits necesarios cuando se necesitan más de 2GB de RAM
- En coma flotante, no hay casi diferencia de rendimiento
- En FORTRAN, pocos problemas. Pero cuidado con la autopromoción de tipos (+i8)
- En C más problemático, ya que:

TIPO	32bits	64bits
short	2	2
int	4	4
long	**4**	**8**
float	4	4
double	8	8
pointer	**4**	**8**

- En C los ejecutables son más grandes y pueden necesitar más memoria (p.e., por alineamiento de estructuras)

CESGA

Compiladores: algunos consejos

- Detectar las funciones que más consumen con cgprof o caliper
- Concentrarse en las funciones que más consumen
- No utilizar compilación con *profile* para cálculo científico
- Cuidado con las relajaciones en el cálculo (+Ofltacc=relaxed)
- Compilar siempre que se pueda en estático
- Utilizar compilación con instrucciones nativas para la máquina
- Es posible (y recomendable) compilar los programas grandes en batch
- Los scripts de revisión de tests a veces engañan.

CESGA

Conclusiones

Conclusiones

- CESGA da soporte de programación, compilación y ejecución de aplicaciones
- Gastar tiempo en el algoritmo puede hacer que ganemos tiempo.
- Estudiar adecuadamente cual es mejor ordenador para cada cálculo
- El resultado no tiene que ser exactamente igual entre arquitecturas
- Utilizar siempre que se pueda librerías nativas
- Es posible mejorar el rendimiento de una aplicación binaria
- La paralelización no siempre ayuda. A veces hay métodos alternativos mejores.

CESGA

APLICACIONES CIENTÍFICAS, LIBRERÍAS Y COMPILADORES

CESGA

¿PREGUNTAS?

CENTRO DE SUPERCOMPUTACIÓN DE GALICIA