Técnicas de Modelado y Optimización del Tiempo de Ejecución de Rutinas Paralelas de Álgebra Lineal

Autor: Luis Pedro García González

Directores: Javier Cuenca y Domingo Giménez

Universidad de Murcia

julio de 2012

Motivación

- La simulación numérica de problemas en ciencias e ingeniería precisa la **resolución** de problemas de gran tamaño con más exactitud y en **menor tiempo**.
- Se requiere **optimizar** las aplicaciones con el fin de obtener **rendimientos** cercanos al pico teórico.
- La optimización de código es un procedimiento muy **exigente**.
- Además el trabajo realizado para una plataforma **no** tiene por qué ser **válido** para otras.
- Proporcionar una metodología de modelado que permite desarrollar técnicas de **óptimización automática**.

Objetivos y Motivación

- Desarrollar métodos para la obtención de **modelos** detallados que **reflejen** el comportamiento del software de álgebra lineal y permitan su **optimización automática**.
- Completar la metodología de modelado analítico con la aplicación de técnicas de **remodelado**.
- Extender la aplicación de la metodología propuesta para sistemas homogéneos a **sistemas heterogéneos**.
- Ampliar la metodología de modelado y optimización a rutinas de álgebra lineal multithread para plataformas **multicore**.

Índice

Índice

Antecedentes

Punto de partida

- Modelo parametrizado que en su formulación genérica: $T = f(n, AP, SP)$
  - n: tamaño del problema a resolver.
  - AP: parámetros del algoritmo. SP: parámetros del sistema.

Computación

Rutinas disponibles en BLAS o librería con una estructura similar. Clasificación:

- Nivel 1: operaciones vector-vector, escalar-vector, coste $O(n)$.
- Nivel 2: operaciones matriz-vector, coste $O(n^2)$.
- Nivel 3: operaciones matriz-matriz, coste $O(n^3)$.

Parámetro coste de una operación: k_1, k_2 y k_3.

Comunicación

Rutinas de paso de mensajes disponibles en MPI

- Coste en transferir n palabras: $t_s + nt_w$.
- Parámetros: t_s y t_w.

Parámetros del sistema. *SP*

Descripción *SP*

- Su valor se **determinará experimentalmente** para cada plataforma y software básico de computación y comunicación.
- En general su valor dependerá de n y de los AP: $SP = g(n, AP)$.

Requisito de los *SP*

- Reflejan las características de la plataforma paralela y software básico al ejecutar una rutina paralela de álgebra lineal.

Mejoras en el modelado

Coste asociado con las operaciones de computación. SP_{compu}

- Diferentes costes para rutinas básicas que realizan **operaciones distintas**. $k_{i,operacion}$.
- Variaciones en el valor del coste para una misma rutina. **Influencia del algoritmo**.
- Asociar costes distintos de las rutinas básicas en cada librería. **Influencia de la librería**.
- Descomponer el modelo de tiempo de ejecución. Refleja la **variación** de los *SP* con el tamaño, forma y localidad de los datos:
  - Modelo inicial: $\frac{2}{3}kn^3$
  - Descomposición: $\frac{2}{9}k_{\frac{n}{6}}n^3 + \frac{2}{9}k_{\frac{3n}{6}}n^3 + \frac{2}{9}k_{\frac{5n}{6}}n^3$
  - donde $k_{\frac{in}{6}}$ sería el valor del *SP* cuando opera con un tamaño de problema entre $\frac{(i-1)n}{6}$ y $\frac{(i+1)n}{6}$.

Mejoras en el modelado

Coste asociado con las operaciones de comunicación. SP_{comu}

- **Diferentes** costes para los parámetros (t_s, t_w) asociados con **cada** una de las **rutinas** básicas de comunicación.
- **Variación** de los SP_{comu} con n, y los AP: p, topología lógica de la malla de procesos, tamaño de bloque.
- **Considerar** distintos SP_{comu} para cada librería y red de interconexión.

Obtención valores de los SP_{compu} y SP_{comu}

- $SP = g(n, AP)$ por medio de **rutinas específicas** que utilizarán las operaciones aritméticas y de comunicación básicas empleadas.

Modelo tiempo de ejecución

Aplicación de la propuesta

- Selección óptima de los AP de 6 rutinas $C = AB$.

Algoritmos para la multiplicación de matrices. $C = AB$

BSR

CBSR

Cannon

Strassen

StPBLAS

StMS

Para cada rutina

- Modelo de tiempo de ejecución $T = f(n, AP, SP)$.
- t_s y t_w rutinas de comunicación: **MPI_Send**, **MPI_Bcast**, **MPI_Allgatherv**...
- $k_{3,dgemm}$ multiplicación descomposición matriz A y B.
- Reutilización de los SP.

Modelo analítico de una rutina de álgebra lineal. Ejemplo rutina StPBLAS.

Algoritmo de Strassen

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Pre-adiciones

$$\begin{aligned} S_1 &= A_{11} + A_{22} \\ S_2 &= A_{21} + A_{22} \\ S_3 &= A_{11} + A_{12} \\ S_4 &= A_{21} - A_{11} \\ S_5 &= A_{12} - A_{22} \end{aligned} \qquad \begin{aligned} T_1 &= B_{11} + B_{22} \\ T_2 &= B_{12} - B_{22} \\ T_3 &= B_{21} - B_{11} \\ T_4 &= B_{11} + B_{12} \\ T_5 &= B_{21} + B_{22} \end{aligned}$$

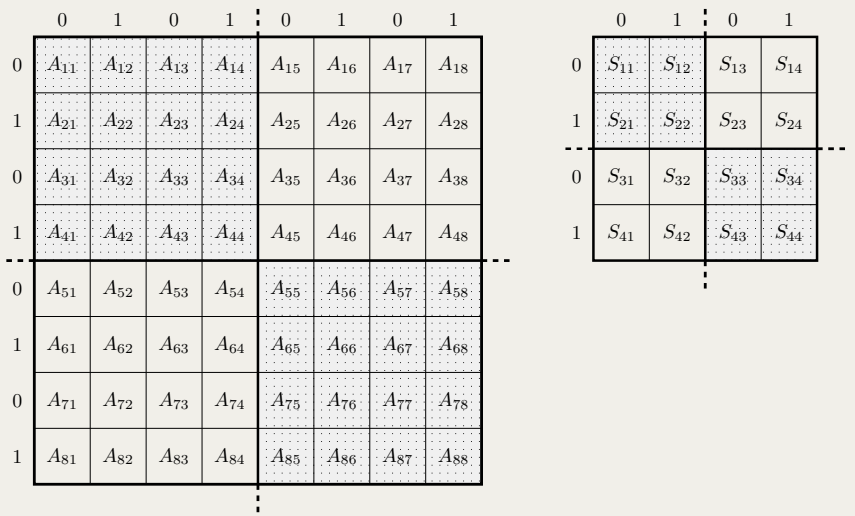
Llamadas recursivas

$$\begin{aligned} P_1 &= S_1 T_1 \\ P_2 &= S_2 B_{11} \\ P_3 &= A_{11} T_2 \\ P_4 &= A_{22} T_3 \\ P_5 &= S_3 B_{22} \\ P_6 &= S_4 T_4 \\ P_7 &= S_5 T_5 \end{aligned}$$

Post-adiciones

$$\begin{aligned} C_{11} &= P_1 + P_4 - P_5 + P_7 \\ C_{12} &= P_3 + P_5 \\ C_{21} &= P_2 + P_4 \\ C_{22} &= P_1 + P_3 - P_2 + P_6 \end{aligned}$$

Distribución de datos cíclica por bloques



Distribución de A con 8×8 bloques en 2×2 procesos. Distribución de la adición S_1 en el siguiente nivel de recursión.

Modelo de tiempo de ejecución de la rutina StPBLAS

- $$T = 7^l \frac{2}{p} \left(\frac{n}{2^l}\right)^3 k_{3,dgemm} + 18 \frac{n^2}{p} \sum_{i=1}^{l} \frac{7^{i-1}}{4^i} k_{2,add} + 7^l \left(2 \left\lceil \frac{n}{2^l b} \right\rceil t_s^{\sqrt{p}-\sqrt{p}} + \frac{2n^2}{4^l \sqrt{p}} t_w^{\sqrt{p}-\sqrt{p}}\right)$$

SP : $k_{3,dgemm}$, $k_{2,add}$, $t_s^{\sqrt{p}-\sqrt{p}}$, $t_w^{\sqrt{p}-\sqrt{p}}$ $\qquad$ AP : l, p, b

Modelo analítico. Selección de los *AP*

Valores de los *SP*. Plataforma P4net

	$k_3, dgemm$ en PBLAS			
n	512	1024	1536	2560
	0.000996	0.000995	0.000988	0.000996

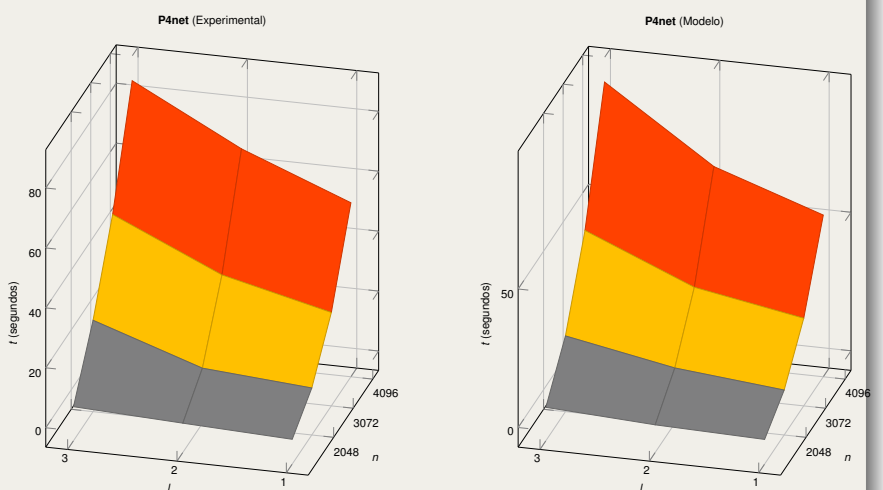
- $t_s = 370.524$ μsegundos, $t_w = 0.709810$ μsegundos

Valores de los *SP*. Plataforma HPC160

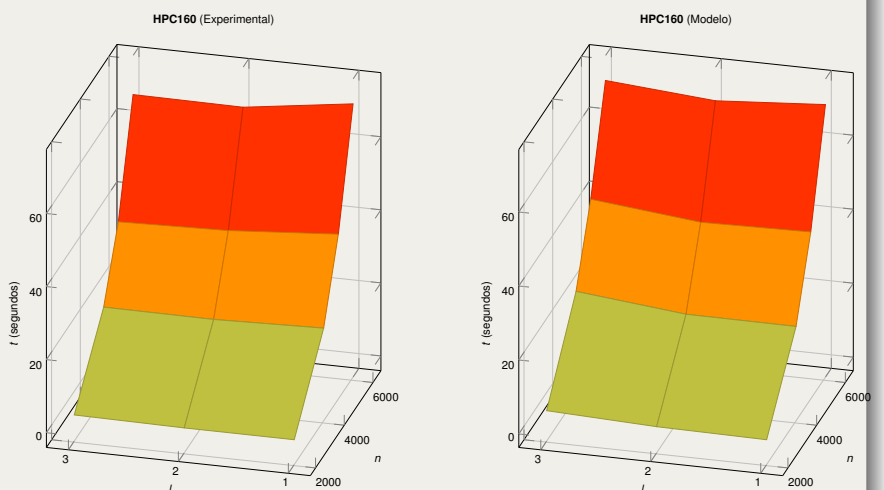
	$k_3, dgemm$ en PBLAS			
n	512	1024	1536	2560
	0.000644	0.000640	0.000995	0.000988

- $t_s = 92.696$ μsegundos, $t_w = 0.065002$ μsegundos

Rutina StPBLAS en P4net. $p = 2 \times 2$, $b = 4$



Rutina StPBLAS en HPC160. $p = 2 \times 2$, $b = 4$



Rutina StPBLAS. Selección del *AP*

Tiempo experimental y aproximación modelo. Plataforma P4net

n	$l = 1$	$l = 2$	$l = 3$
	Tiempo experimental (segundos)		
1024	**1.815157**	3.296943	4.901434
2048	**9.222891**	11.931264	24.009070
3072	**24.715618**	33.355596	49.610680
4096	**51.600152**	65.679293	84.528084
	Tiempo teórico (segundos)		
1024	**1.827532**	2.877208	4.984113
2048	**9.142147**	12.899089	20.246833
3072	**24.567550**	31.535856	47.827317
4096	**51.296548**	64.581415	90.880069
	Desviación (%)		
1024	0.68	12.73	1.69
2048	0.88	8.11	15.67
3072	0.60	5.46	3.59
4096	0.59	1.67	7.51

Tiempo experimental y aproximación modelo. Plataforma HPC160

n	$l = 1$	$l = 2$	$l = 3$
	Tiempo experimental (segundos)		
2048	**2.767520**	2.788013	3.119059
4096	21.333430	**20.499468**	20.651807
5120	41.086701	38.851529	**38.037099**
6144	70.749382	**66.608620**	66.857656
	Tiempo teórico (segundos)		
2048	**3.059522**	3.412719	4.568527
4096	**21.856261**	21.980797	24.988895
5120	41.504228	**40.949497**	43.996547
6144	70.003190	**67.857265**	70.174509
	Desviación (%)		
2048	10.55	22.41	46.47
4096	2.45	7.23	21.00
5120	1.02	5.40	15.67
6144	1.05	1.87	4.96

Modelo rutinas de multiplicación de matrices

- Desviaciones entre el modelo y el experimental (%)

Plataforma P4net.

n	BLASopt	PBLAS	BSR	CBSR	Cannon	Strassen	StPBLAS
512	16.09	0.30	23.11	1.21	2.29	0.18	22.52
1024	2.01	0.28	3.38	0.67	4.27	0.62	0.68
2048	5.12	0.14	2.44	1.62	5.58	0.14	0.88
4096	8.18	0.35	4.20	2.77	4.46		0.59
5120	9.65	0.03	4.33	0.38	0.47		

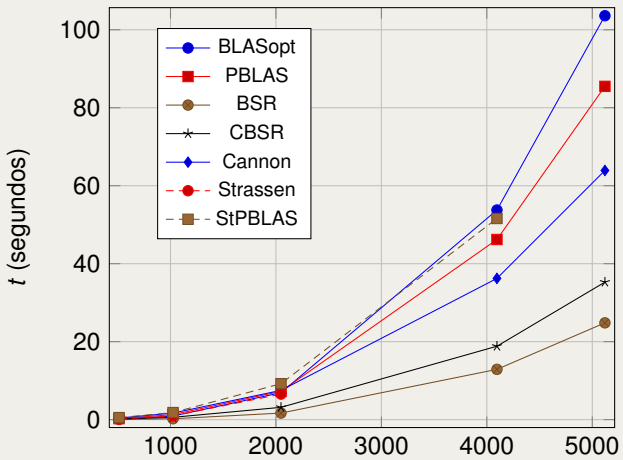
Plataforma HPC160.

n	BLASopt	PBLAS	BSR	CBSR	Cannon	Strassen	StPBLAS
512	4.58	22.95	7.96	4.41	48.31	9.83	
1024	7.39	1.17	16.75	9.37	0.81	3.66	21.64
2048	7.26	4.17	2.98	0.99	2.08	4.94	10.55
4096	1.56	3.52	2.90	5.30	0.27	8.80	2.45
5120	9.57	6.72	0.50	0.64	0.31	5.78	3.40

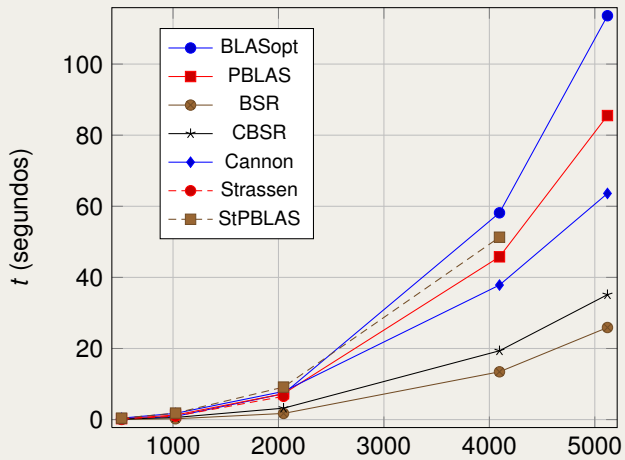
Multiplicación de Matrices. Resultados Experimentales

Comparación entre el tiempo experimental y el teórico. Selección de la rutina

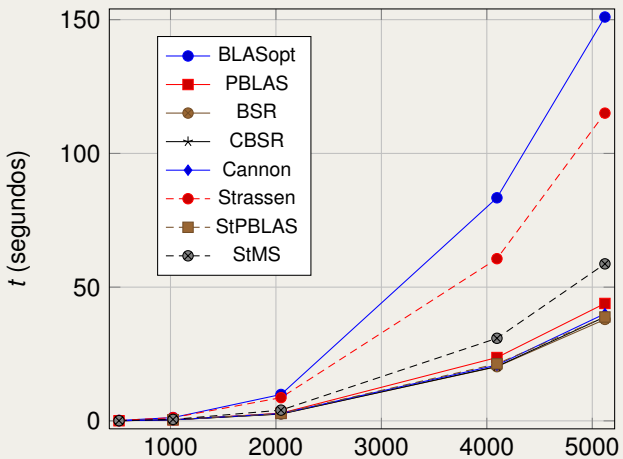
P4net - Experimental



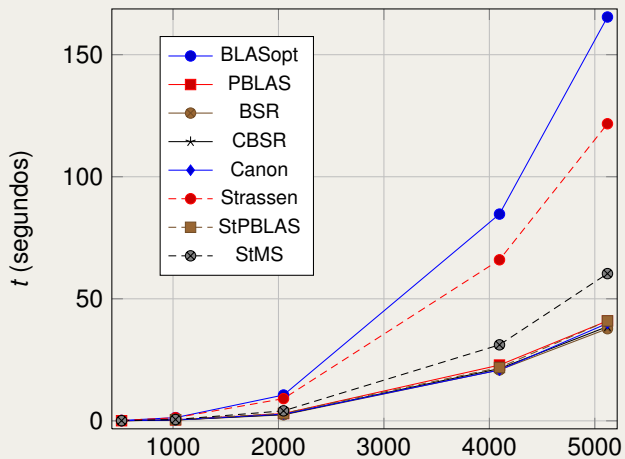
P4net - Teórico



HPC160 - Experimental



HPC160 - Teórico



Factorización de Cholesky $A = GG^T$

Descomposición del modelo de tiempo de ejecución

- Rutina de nivel superior sin bloques que emplea rutinas básicas de nivel 1 y 2.
- Modelo simplificado no proporciona una buena aproximación.

Versión producto matriz vector

- $T = 2\sum_{i=1}^{n-1} i(n-i)k_{2,dgemv} \approx \frac{1}{3}n^3 k_{2,dgemv}$
- Rutina **dgemv** en matrices de tamaño $(n-i) \times i$, para $i = 1, 2, \ldots, n-1$

SP y Modelo

- Valores de $k_{2,dgemv}$ para $\frac{n}{8}$, $\frac{3n}{8}$, $\frac{5n}{8}$ y $\frac{7n}{8}$

$$T = \frac{5}{96}n^3 k_{2,dgemv_{\frac{7n}{8}\times\frac{n}{8}}} + \frac{11}{96}n^3 k_{2,dgemv_{\frac{5n}{8}\times\frac{3n}{8}}} + \frac{11}{96}n^3 k_{2,dgemv_{\frac{3n}{8}\times\frac{5n}{8}}} + \frac{5}{96}n^3 k_{2,dgemv_{\frac{n}{8}\times\frac{7n}{8}}}$$

Factorización de Cholesky $A = GG^T$

Versión producto vector escalar

- $T = 2\sum_{i=1}^{n-1} i(n-i)k_{1,daxpy} \approx \frac{1}{3}n^3 k_{1,daxpy}$
- Rutina **daxpy** actualiza columnas de tamaño $(n-i)$, para $i = 1, 2, \ldots, n-1$

SP y Modelo

- Valores de $k_{1,daxpy}$ para $k_{1,daxpy_{\frac{7n}{8}}}$, $k_{1,daxpy_{\frac{5n}{8}}}$, $k_{1,daxpy_{\frac{3n}{8}}}$ y $k_{1,daxpy_{\frac{n}{8}}}$

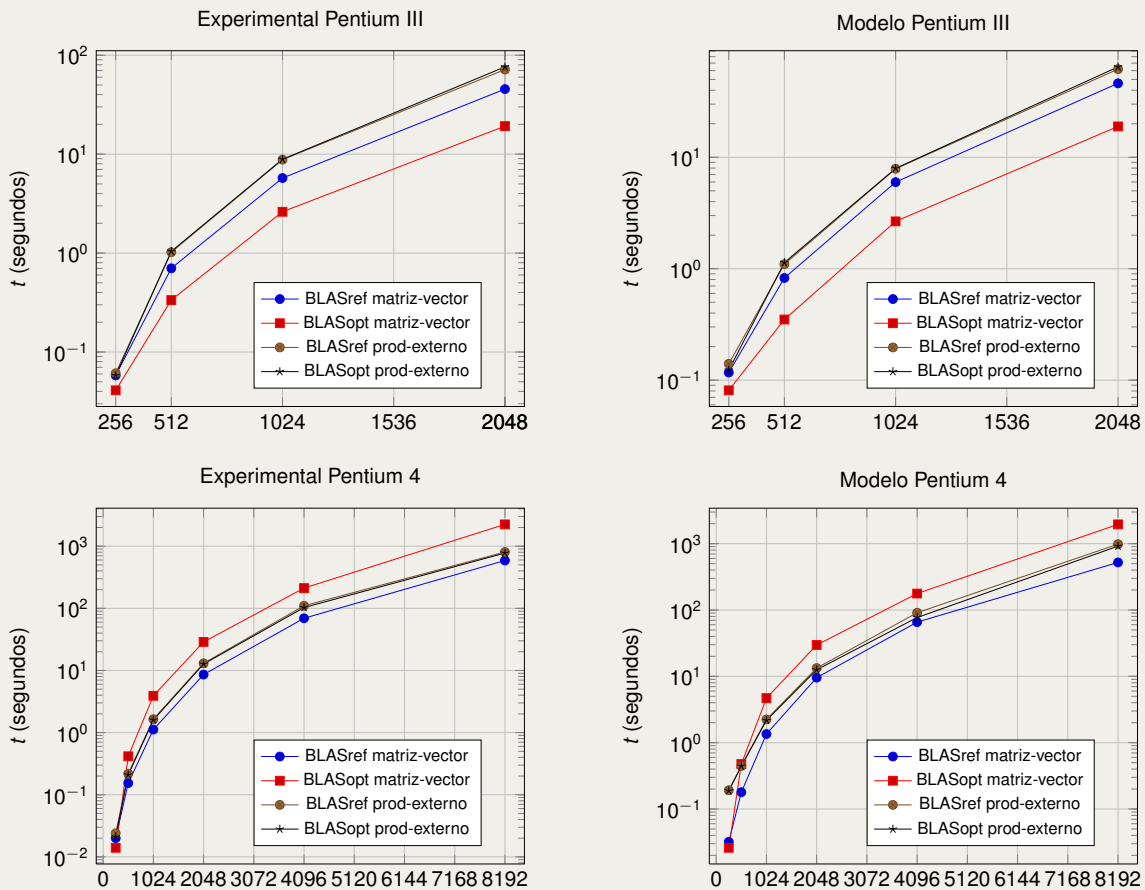
$$T = \frac{5}{96}n^3 k_{1,daxpy_{\frac{7n}{8}}} + \frac{11}{96}n^3 k_{1,daxpy_{\frac{5n}{8}}} + \frac{11}{96}n^3 k_{1,daxpy_{\frac{3n}{8}}} + \frac{5}{96}n^3 k_{1,daxpy_{\frac{n}{8}}}$$

Selección de los *AP*

- Selección de la combinación algoritmo y librería básica que proporciona menores tiempos de ejecución. Ahora el *AP* es la librería básica.

Resultados Experimentales

Selección del algoritmo y librería
Estudio en sistemas Pentium III y Pentium 4. Librerías BLASref y BLASopt



Pentium III

Orden de selección:

1. matriz vector con BLASopt.
2. matriz vector con BLASref.
3. producto escalar con BLASref.
4. producto escalar con BLASopt.

con $n = 256$ el orden de los dos últimos se intercambia.

Pentium 4

Orden de selección:

1. matriz vector con BLASref.
2. producto escalar con BLASopt.
3. producto escalar con BLASref.
4. matriz vector con BLASopt.

con $n = 256$ mejor matriz vector con BLASopt.

Factorización de Cholesky $A = GG^T$ paralela por bloques

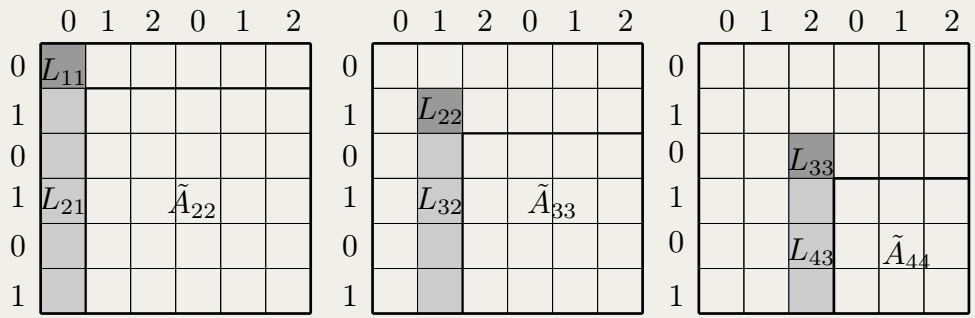
Descripción

1. Calculo del factor L_{11} por el proceso {0,0}. Rutina de nivel 2 de LAPACK **dpotf2**.
2. Operación $L_{21} = A_{21}(L_{11}^T)^{-1}$, realizada por los procesos de la columna 0 de la malla. Rutina nivel 3 de BLAS **dtrsm**.
3. Operación $\tilde{A}_{22} = A_{22} - L_{21}L_{21}^T = L_{22}L_{22}^T$, realizada por todos los procesos en la malla. Rutinas nivel 3 de BLAS **dsyrk** y **dgemm**.

- *AP*: $p = r \times c$ y b.
- *SP*: coste rutinas de BLAS, LAPACK y de rutinas de MPI. Determinación de su dependencia respecto de n y los *AP*.

Distribución de los cálculos en la rutina de Cholesky con $\frac{n}{b} = 6$ y $p = 2 \times 3$

- Versión por bloques con una distribución cíclica en una malla de dos dimensiones de $p = r \times c$ procesos.



(a) Primer paso (b) Segundo paso (c) Tercer paso

Modelo tiempo de ejecución $T = f(n, AP, SP) = T_{compu} + T_{comu}$

- **Computación:**

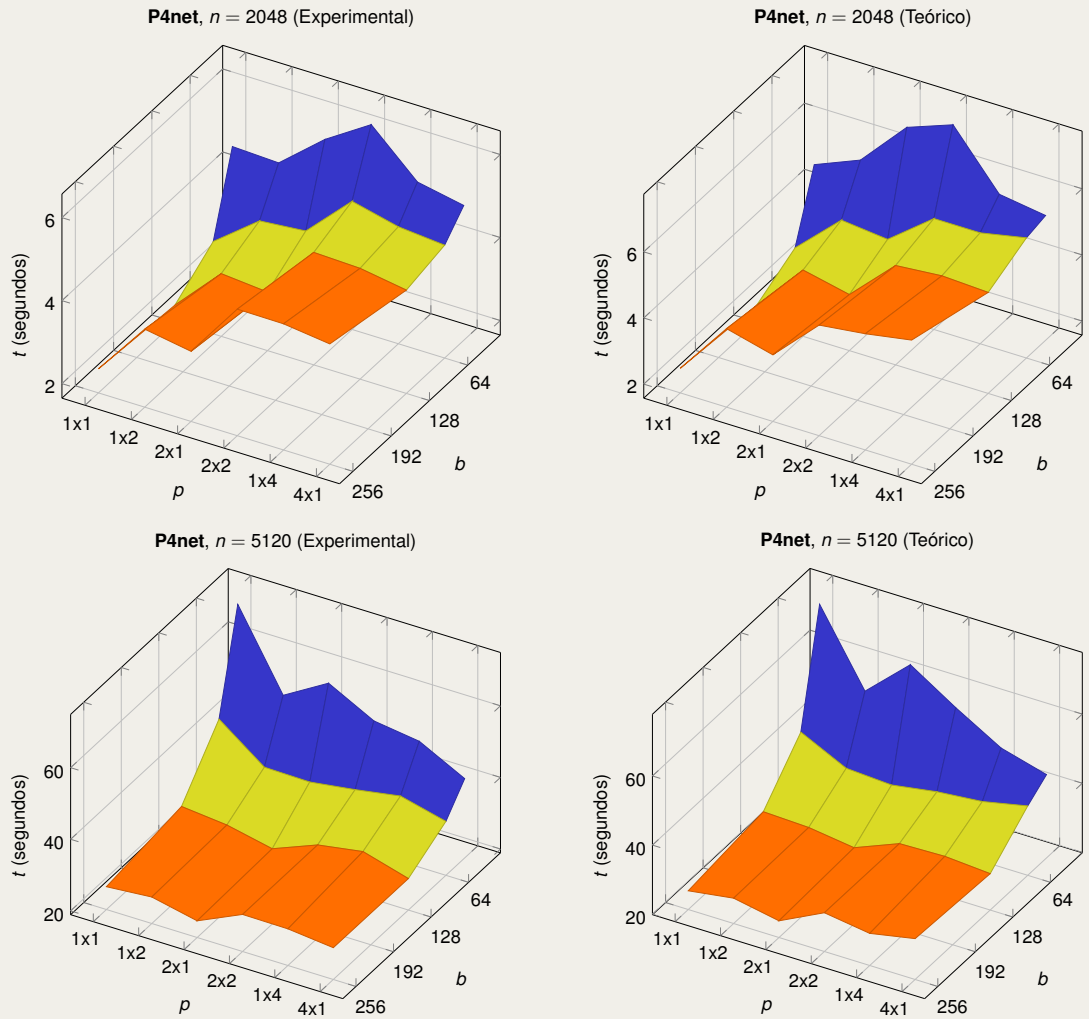
$$k_{2,dpotf2}\frac{nb^2}{3} + k_{3,dtrsm}\left[\frac{n}{r}(r-1) + \frac{n}{2}\left(\frac{n}{rb}-1\right)\right]b^2 + k_{3,dsyrk}\left[\frac{1}{\sqrt{p}}\right]\left(\frac{n^2-nb}{2}\right)(b+1) + \frac{2}{p}k_{3,dgemm}\left(\frac{n^3}{6} - \frac{n^2b}{2} + \frac{nb^2}{3}\right)$$

- **Comunicación:**

$$\left(\frac{n}{b}-1\right)(t_s + b^2 t_{w_d}) + \frac{n}{2b}\left(\frac{n}{b}-1\right)\left(t_s + b^2 t_{w_d}\right) + \left(\frac{n}{b}-1\right)\left(bt_s + \frac{nb}{2}t_w\right)$$

Resultados Experimentales

Comparación tiempos experimentales y modelo en P4net



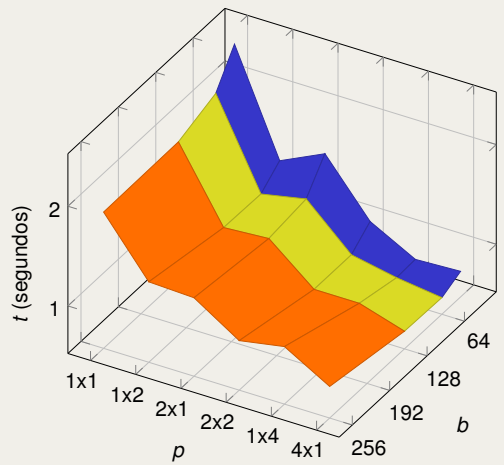
Discusión resultados

- El tiempo de ejecución varía con $p = r \times c$.
- Para un $p = r \times c$ con b.
- Valores óptimos de $p = r \times c$ y b con n.
- AP óptimos varían con la plataforma.
- $T = f(n, SP, AP)$ refleja las variaciones.

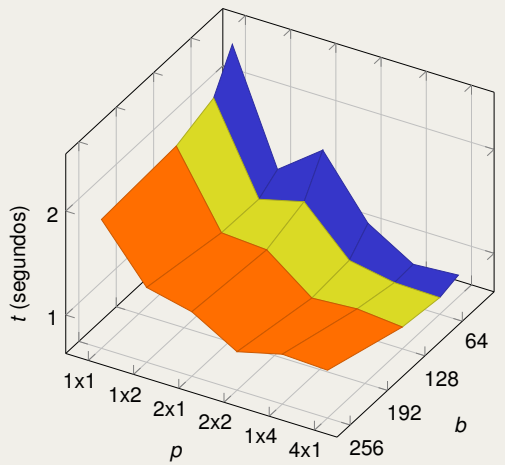
Resultados Experimentales

Comparación tiempos experimentales y modelo en HPC160

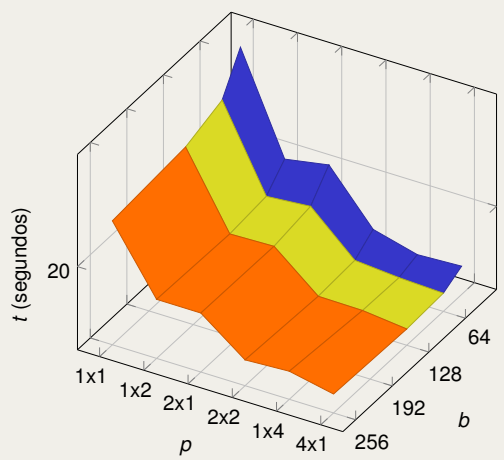
HPC160smp, $n = 2048$ (Experimental)



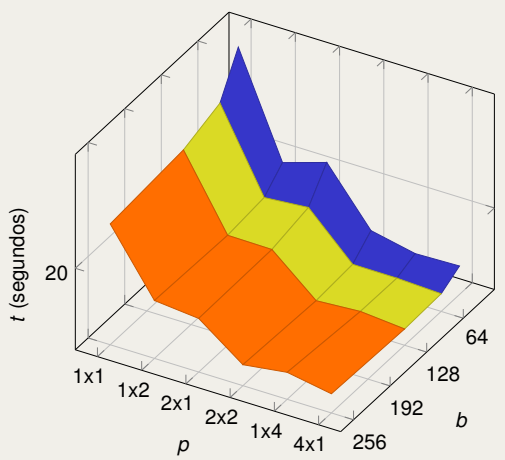
HPC160smp, $n = 2048$ (Teórico)



HPC160smp, $n = 5120$ (Experimental)



HPC160smp, $n = 5120$ (Teórico)



Discusión resultados

- El tiempo de ejecución varía con $p = r \times c$.
- Para un $p = r \times c$ con b.
- Valores óptimos de $p = r \times c$ y b con n.
- AP óptimos varían con la plataforma.
- $T = f(n, SP, AP)$ refleja las variaciones.

Selección de los *AP*

AP óptimos en P4net

P4net

	$p=1$			$p=2$					$p=4$				
	opt.	mod.	dev.	opt.		mod.		dev.	opt.		mod.		dev.
n	b	b	%	b	$r \times c$	b	$r \times c$	%	b	$r \times c$	b	$r \times c$	%
512	64	64	0	64	1×2	128	1×2	2.0	128	1×4	128	1×4	0
1024	128	128	0	128	1×2	64	1×2	3.4	64	4×1	64	1×4	1.2
2048	128	128	0	128	2×1	128	2×1	0	128	1×4	128	2×2	9.6
4096	256	256	0	256	2×1	128	2×1	0.1	128	4×1	128	4×1	0
5120	256	256	0	256	2×1	256	2×1	0	128	4×1	128	4×1	0
Media			**0**					**1.1**					**2.2**

AP óptimos en HPC160

	HPC160smp $p=4$					HPC160mc $p=4$					HPC160smp-mc $p=8$				
	opt.		mod.		dev.	opt.		mod.		dev.	opt.		mod.		dev.
n	b	$r \times c$	b	$r \times c$	%	b	$r \times c$	b	$r \times c$	%	b	$r \times c$	b	$r \times c$	%
512	32	4×1	32	4×1	0	32	4×1	32	2×2	81	32	2×4	32	2×4	0
1024	64	4×1	64	4×1	0	64	4×1	32	2×2	62	32	2×4	32	2×4	0
2048	64	4×1	64	4×1	0	64	4×1	32	2×2	1.3	64	2×4	32	2×4	17.2
4096	128	4×1	128	4×1	0	128	2×2	128	4×1	1.6	128	2×4	128	2×4	0
5120	128	4×1	128	4×1	0	128	2×2	128	2×2	0	64	2×4	64	2×4	0
7168	128	4×1	128	4×1	0	128	2×2	128	2×2	0	64	2×4	64	2×4	0
Media					**0**					**24.3**					**2.9**

Índice

Remodelado

Descripción

Construcción de un nuevo modelo de tiempo de ejecución cuando:

- El $T = f(n, AP, SP)$ y el valor de los SP no selecciona los AP óptimos.
- No se pueda obtener un $T = f(n, AP, SP)$. Desconocemos el algoritmo seguido por la rutina a modelar.

Propuesta construcción modelo de tiempo de ejecución

- Modelo basados en funciones matemáticas combinación de n y los AP.
- Multiplicación de matrices $C = AB$ de tamaño $n \times n$ con un coste de $O(n^3)$

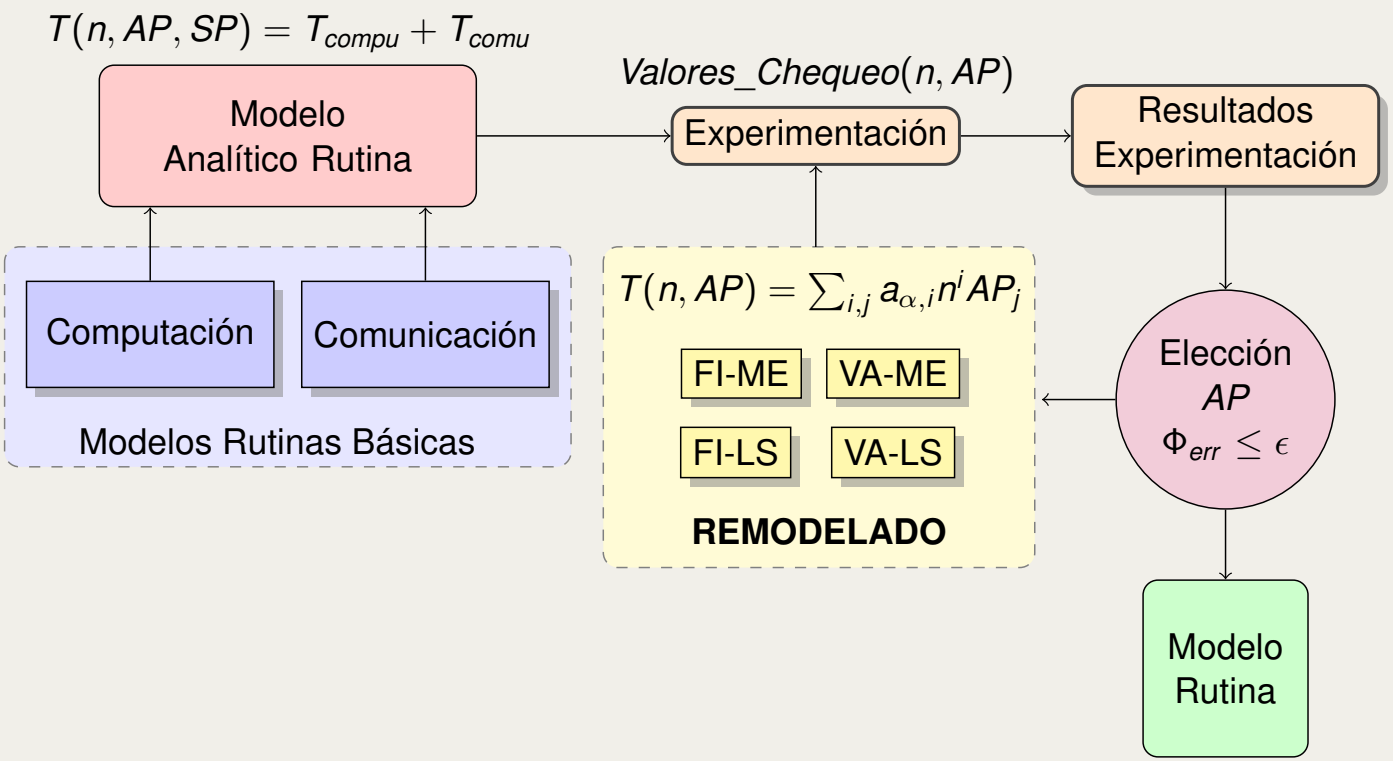
$$T(n) = \beta_3 n^3 + \beta_2 n^2 + \beta_1 n + \beta_0$$

- Si la multiplicación es por bloques de tamaño b:

$$T(n) = \beta_{3,1} n^3 b + \beta_{3,0} n^3 + \beta_{3,-1} \frac{n^3}{b} + \beta_{2,1} n^2 b + \beta_{2,0} n^2 + \beta_{2,-1} \frac{n^2}{b} + \beta_{1,1} nb + \beta_{1,0} n + \beta_{1-1} \frac{n}{b} + \beta_{0,0}$$

Remodelado. Esquema general

Obtención del modelo de tiempo de ejecución



Valor de Φ_{err}

Se aplica un mayor peso a las aproximaciones realizadas por el modelo para tamaños de problema grandes.

$$\Phi_{err} = \left[1 - \left(\frac{\sum Valores\_Chequeo\ t_{teo}}{\sum Valores\_Chequeo\ t_{exp}}\right)\right] 100$$

Estimación de los Coeficientes

Métodos para la estimación de los coeficientes

- **FIxed Minimal Executions (FI-ME)**: Un único polinomio aproxima el tiempo de ejecución. Sistema de d ecuaciones con d incógnitas, utilizando d combinaciones distintas del tamaño del problema y de los parámetros del algoritmo (n, AP).
- **VAriable Minimal Executions (VA-ME)**: Se seleccionan i regiones de posibles combinaciones de (n, AP). En cada una se aplica **FI-ME**.
- **FIxed Least Square (FI-LS)**: Un único polinomio. Coeficientes se obtienen con regresión lineal para un conjunto de valores de (n, AP) seleccionados.
- **VAriable Least Square (VA-LS)**: Se utilizan i polinomios. Los polinomios se obtienen usando el método **FI-LS** en las i regiones de combinaciones de (n, AP).

Multiplicación de Strassen de matrices

Remodelado de rutinas básicas

- A partir de $T = f(n, AP, SP)$:

$$T = 7^l 2 \left(\frac{n}{2^l}\right)^3 k_{3,dgemm} + \frac{18}{4} n^2 \sum_{i=1}^{l} \left(\frac{7}{4}\right)^{i-1} k_{2,add}$$

- En lugar de los SP de las rutinas básicas, aparecen sus tiempos de ejecución:

$$T = 7^l t_{mult}\left(\frac{n}{2^l}\right) + 18 \sum_{i=1}^{l} 7^{i-1} t_{add}\left(\frac{n}{2^i}\right)$$

Aproximación de $t_{mult}(\frac{n}{2^l})$ y $t_{add}(\frac{n}{2^l})$

- El método FIxed Least Square (FI-LS) obtiene buenos resultados.
- t_{mult} polinomio de grado tres. Coste teórico de $O(n^3)$.
- t_{add} polinomio de grado seis. Coste teórico de $O(n^2)$.

Resultados experimentales

Strassen - Sol

n	*l*	mod.	exp.	des. (%)
3072	1	**11.75**	**12.86**	8.58
3072	2	13.90	13.63	1.99
3072	3	37.04	15.76	135.06
4096	1	**27.21**	**29.71**	8.41
4096	2	28.59	30.10	5.02
4096	3	48.76	33.34	46.26
5120	1	**53.14**	56.83	6.51
5120	2	53.53	**56.43**	5.13
5120	3	71.08	60.19	18.09
6144	1	96.48	96.32	0.17
6144	2	**95.39**	**93.69**	1.82
6144	3	110.40	98.39	12.21

Strassen - HPC160

n	*l*	mod.	exp.	des. (%)
3072	1	29.96	29.70	0.89
3072	2	28.54	27.82	2.57
3072	3	**17.55**	**27.61**	36.46
4096	1	69.85	70.85	1.43
4096	2	66.04	64.55	2.30
4096	3	**57.82**	**62.56**	7.58
5120	1	135.03	134.67	0.26
5120	2	125.76	123.38	1.92
5120	3	**118.122**	**118.45**	0.28
6144	1	229.786	232.268	1.07
6144	2	211.104	210.876	0.11
6144	3	**201.150**	**199.326**	0.92

Remodelado rutina de Strassen

Remodelado

- A partir del modelo teórico, se define un conjunto de funciones polinomiales de grado tres:

$$T(n,l) = 7^l 2 \left(\frac{n}{2^l}\right)^3 M(l) + \frac{9}{2} n^2 A(l) \sum_{i=1}^{l} \left(\frac{7}{4}\right)^{i-1}$$

Coeficientes

- Coeficientes $M(l)$ y $A(l)$. Coste multiplicación y adición de matrices.
- Se fija l y se varía n.
- Para cada l obtenemos una mejor aproximación de $M(l)$ y $A(l)$. Método FI-LS.
- Modelo de tiempo de ejecución para cada l.

Remodelado rutina de Strassen

Un único modelo

- Un único modelo teórico para cualquier combinación de (n, AP) de la rutina de Strassen.
- Ajuste por mínimos cuadros del conjunto de valores de $M(l)$ y $A(l)$:

$$M(l) = 1.907 \cdot 10^{-10} + 4.580 \cdot 10^{-11} \cdot l - 1.445 \cdot 10^{-11} \cdot l^2$$

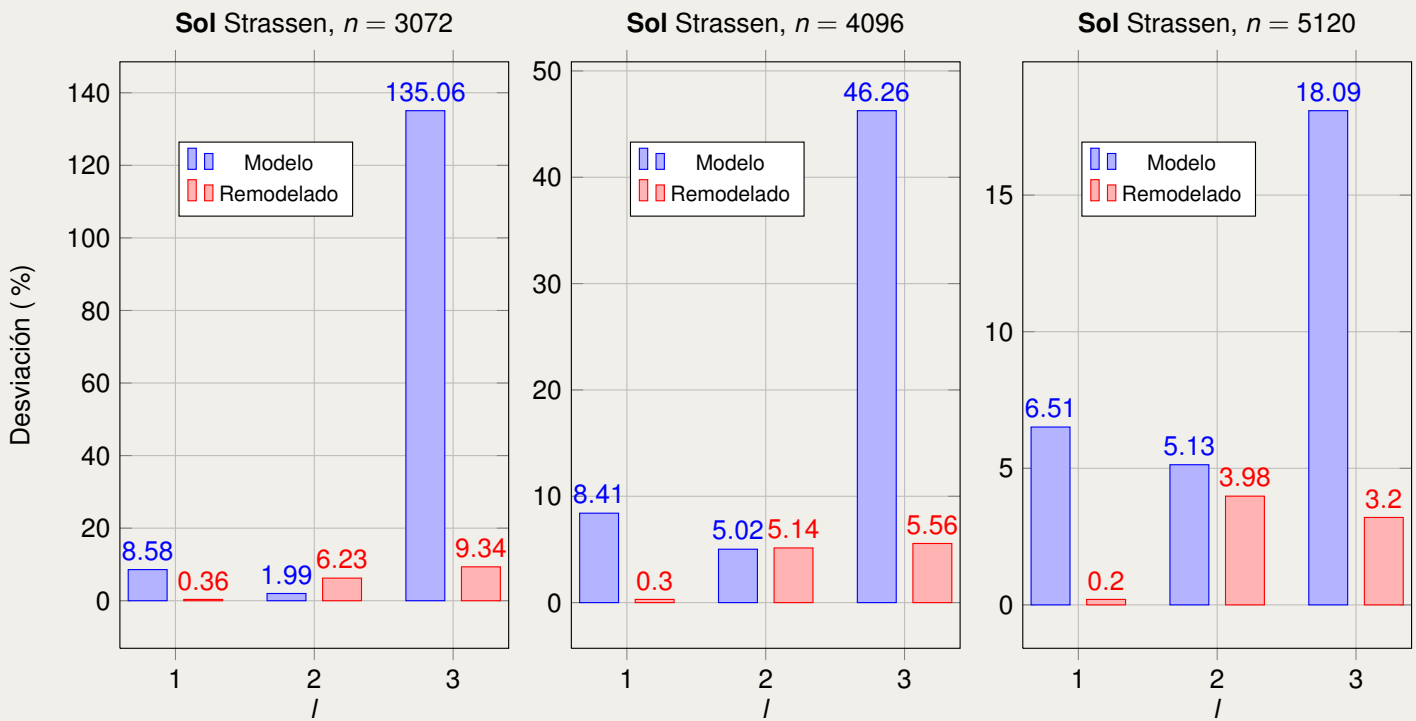
$$A(l) = 4.378 \cdot 10^{-08} - 5.131 \cdot 10^{-09} \cdot l$$

Resultados experimentales

Strassen - Sol

n	l	exp.	mod.
1664	1	**1.99**	**2.27**
1664	2	2.37	2.73
1664	3	3.12	3.27
2176	1	**4.27**	**4.83**
2176	2	4.77	5.51
2176	3	6.08	6.25
2688	1	**7.87**	**8.80**
2688	2	8.40	9.67
2688	3	10.28	10.52
3200	1	**13.02**	**14.51**
3200	2	13.56	15.51
3200	3	16.00	16.30
5120	1	56.80	56.71
5120	2	**56.44**	57.01
5120	3	60.04	**55.09**
5632	1	75.78	74.92
5632	2	73.50	74.56
5632	3	**71.70**	**70.97**

Comparación entre las desviaciones

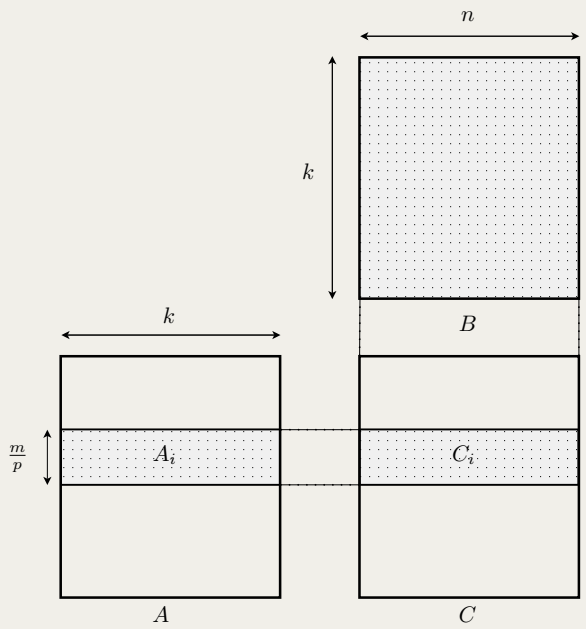


Remodelado rutinas paralelas

Multiplicación paralela de matrices con *A* distribuida *rowwise block-striped* y *B* replicada (BSR)

- Modelo teórico: $T = \frac{2n^3}{p} k_{3,dgemm} + t_{s,send} + \frac{n^2}{p} t_{w,send} + t_{s,bcast} + n^2 t_{w,bcast}$
- Remodelado rutinas básicas: $T = t_{mult}\left(\frac{m}{p}\right) + (p-1)t_{send}\left(\frac{n^2}{p}\right) + t_{bcast}\left(n^2\right)$

Distribución de las matrices



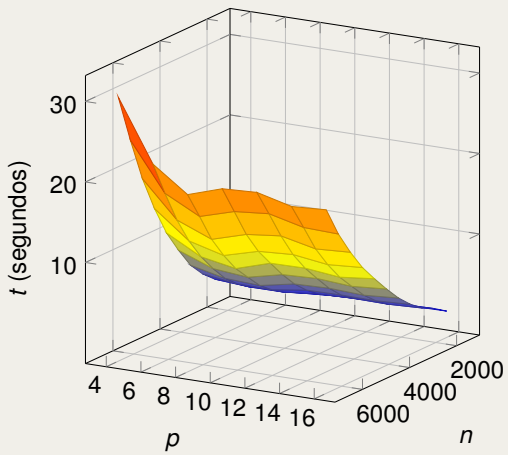
Modelo computación y comunicación

- Computación: FI-LS y polinomio de tercer grado.
- t_{send}: Método FI-LS es suficiente.
- t_{bcast}: Necesario utilizar método VA-LS.

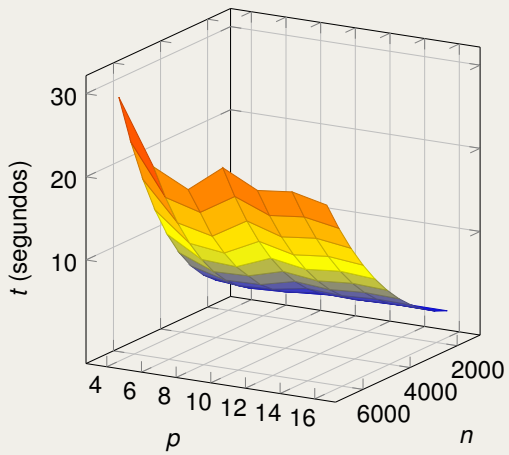
Resultados experimentales

Multiplicación BSR - Rosebud

Rosebud, Rutina BSR (Experimental)



Rosebud, Rutina BSR (Modelo)



Orden de selección de p en Rosebud

n	$p=4$ exp.	$p=4$ mod.	$p=4$ dev. (%)	$p=8$ exp.	$p=8$ mod.	$p=8$ dev. (%)	$p=12$ exp.	$p=12$ mod.	$p=12$ dev. (%)	$p=16$ exp.	$p=16$ mod.	$p=16$ dev. (%)
1024	$2.^o$	$1.^o$	2.49	$1.^o$	$2.^o$	2.49	$3.^o$	$3.^o$	0	$4.^o$	$4.^o$	0
2048	$2.^o$	$2.^o$	0	$1.^o$	$1.^o$	0	$3.^o$	$3.^o$	0	$4.^o$	$4.^o$	0
2560	$2.^o$	$2.^o$	0	$1.^o$	$1.^o$	0	$3.^o$	$3.^o$	0	$4.^o$	$4.^o$	0
4096	$4.^o$	$4.^o$	0	$1.^o$	$1.^o$	0	$2.^o$	$3.^o$	0.37	$3.^o$	$2.^o$	0.37
5120	$4.^o$	$4.^o$	0	$1.^o$	$1.^o$	0	$3.^o$	$3.^o$	0	$2.^o$	$2.^o$	0
5632	$4.^o$	$4.^o$	0	$1.^o$	$1.^o$	0	$3.^o$	$3.^o$	0	$2.^o$	$2.^o$	0
6144	$4.^o$	$4.^o$	0	$1.^o$	$1.^o$	0	$3.^o$	$3.^o$	0	$2.^o$	$2.^o$	0
6656	$4.^o$	$4.^o$	0	$1.^o$	$1.^o$	0	$3.^o$	$3.^o$	0	$2.^o$	$2.^o$	0

Remodelado rutina BSR

Aplicación del método

- Multiplicación de matrices de coste $O(n^3)$. El *AP* es el número de procesos p.
- Función polinomial de grado tres con las posibles combinaciones de n y p:

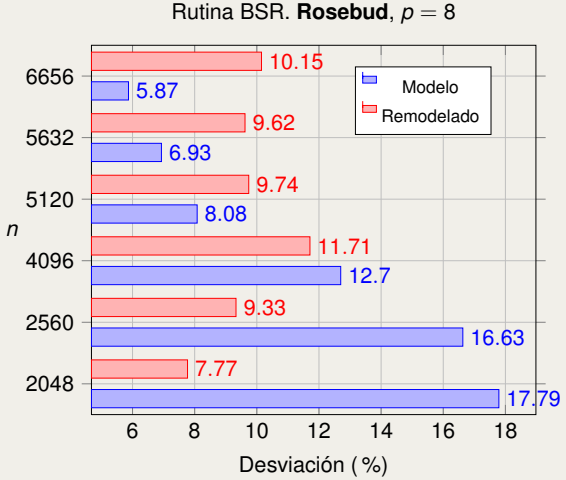
$$T(n,p) = a_{3,1}n^3p + a_{3,0}n^3 + a_{3,-1}\frac{n^3}{p} + a_{2,1}n^2p + a_{2,0}n^2 + a_{2,-1}\frac{n^2}{p} + a_{1,1}np + a_{1,0}n + a_{1,-1}\frac{n}{p} + a_{0,1}p + a_{0,0} + \frac{a_{0,-1}}{p}$$

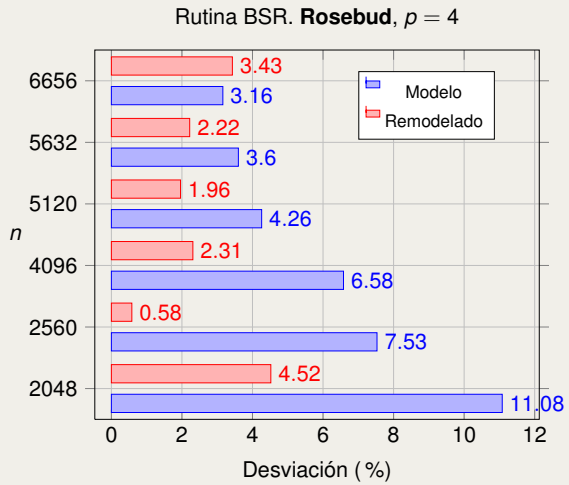
- Necesario el cálculo de 12 coeficientes.
- Método FI-LS con $p = \{2, 4, 6\}$ y $n = \{1000, 1500, \cdots, 4000\}$.

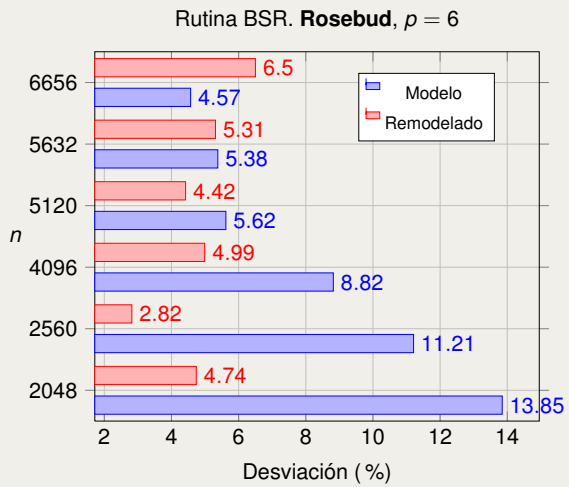
Resultados experimentales

Comparación resultados experimentales y remodelado en Rosebud

	n	1024	2048	3584	4608	5632	6656
$p = 4$							
exp.		0.167	1.147	5.181	10.497	18.640	30.135
mod.		0.164	1.118	5.237	10.490	18.316	29.205
$p = 6$							
exp.		**0.159**	1.109	4.227	8.202	14.145	22.453
mod.		**0.153**	0.978	4.117	7.935	13.497	21.117
$p = 8$							
exp.		0.165	**1.066**	**4.029**	**7.559**	**12.517**	**19.499**
mod.		0.157	**0.962**	**3.647**	**6.793**	**11.322**	**17.490**







Índice

Introducción

Planteamiento

- Buen rendimiento rutinas de álgebra lineal diseñadas para sistemas homogéneos, sin necesidad de modificar su código.
- Si procesadores de distinta capacidad de cómputo. La distribución del trabajo tiene que ser heterogénea:
  - **HoHe**: Cada procesador se encarga de ejecutar un sólo proceso. Distribución de datos heterogénea sobre los procesos. Algoritmos heterogéneos.
  - **HeHo**: Distribución de procesos sobre procesadores heterogénea y distribución de datos homogénea. Algoritmo homogéneo.

Propuesta

- $T = f(n, AP, SP)$ realizará la selección de los AP.
- Se combina con estrategias de asignación de procesos a procesadores.
- Selección de los AP homogéneos (P, $p = r \times c$, b).
- Procesos a generar y número de procesos asignados a cada procesador.

Modelo de tiempo de ejecución en plataformas heterogéneas

- *AP* heterogéneos
  - $d = (d_1, \ldots, d_P)$, en el que cada d_i representa el número de procesos asignados a cada procesador. P número de procesadores disponibles en el sistema.
  - $D = \sum_{i=1}^{P} d_i$ el número total de procesos generados para ejecutar la rutina.
- Valor de los *SP*
  - $SP = (SP_1, \ldots, SP_P)$
  - Variación de los SP_i con el número de procesos por procesador.
    - El procesador i con d_i procesos un valor de $SP_i \times d_i$.
    - Se considerará el valor de cada SP_{compu} como el máximo para todos los procesadores.
    - El máximo entre pareja de procesadores para los SP_{comu}.
    - Un valor global de los *SP*.

Factorización LU. Modelo de tiempo de ejecución

Descripción

- Versión paralela por bloques, equivalente a la rutina **pdgetrf** disponible en la librería ScaLAPACK.
- Distribución cíclica por bloques entre los $r \times c$ procesos.
- Llamadas a rutinas de PBLAS y LAPACK para los cálculos aritméticos y de MPI para las comunicaciones.

Algoritmo homogéneo

1. Factorización LU en el bloque A_{11} por el proceso {0,0}. Rutina de nivel 2 de LAPACK **dpotf2**.
2. Resolución sistemas de ecuaciones: $L_{21}\,U_{11} = A_{21}$ y $L_{11}\,U_{12} = A_{12}$, realizada por los procesos de la fila 0 y columna 0 de la malla. Rutina de nivel 3 de PBLAS **pdtrsm**.
3. Resolver la ecuación: $L_{21}\,U_{12} + L_{22}\,U_{22} = A_{22}$, reformulada como $\tilde{A}_{22} = A_{22} - L_{21}\,U_{12}$ realizada por todos los procesos en la malla. Rutinas nivel 3 de PBLAS **pdgemm**.

$T = f(n, AP, SP)$

- **Computación:**
  $$T_{COMPU} = \frac{2}{3}\frac{n^3}{p}k_{3,dgemm} + \frac{r+c}{p}n^2 b k_{3,dtrsm} + \frac{1}{3}nb^2 k_{2,dgetf2}$$
- **Comunicación:** $T_{COMU} = \frac{2ng}{b}t_s + \frac{2n^2 g}{p}t_w$
- $T = T_{COMPU} + T_{COMU}$

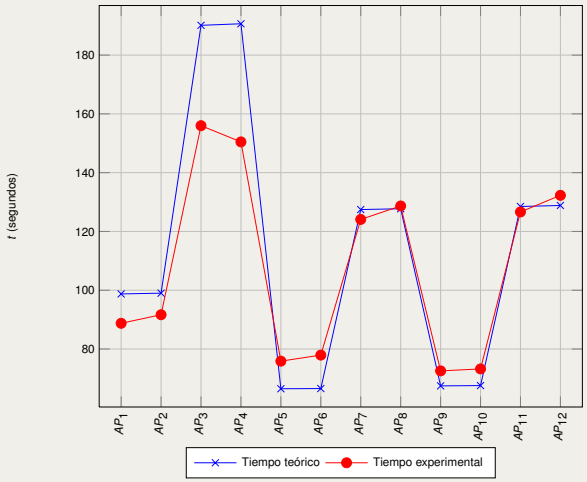
AP y *SP*

- *SP*: Parámetros del Sistema
  - $k_{3,dgemm}$, $k_{3,dtrsm}$, $k_{2,dgetf2}$
  - t_s, t_w
- *AP*: Parámetros del Algoritmo
  - b: tamaño de bloque
  - p: número de procesos
  - $r \times c$: forma de la malla lógica 2D
  - d: asignación de procesos a procesadores

Comparativa tiempo experimental y modelado

Plataforma SUNEt. $p = 8, P = 6, n = 2048$

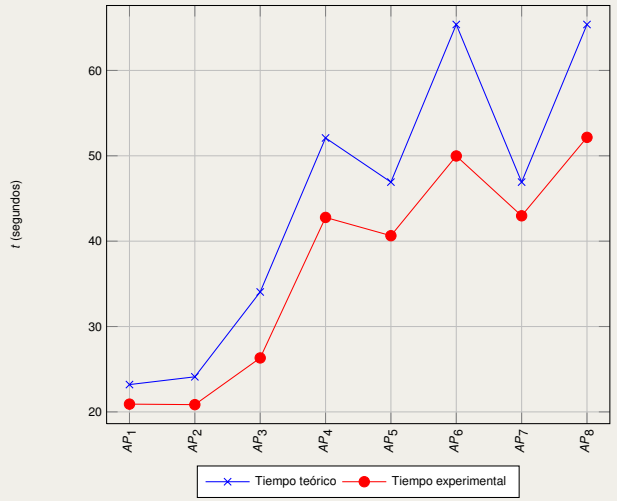
- SUNEt: 5 SUN Ultra 1 (P_0 - P_4) y 1 SUN Ultra 5 (P_5)



	Asignación $p = 8, P = 6$ $d = (P_0, ..., P_5)$	$p = r \times c$	b
AP_1	(3,1,1,1,1,1)	2×4	32
AP_2	(3,1,1,1,1,1)	2×4	64
AP_3	(3,1,1,1,1,1)	1×8	32
AP_4	(3,1,1,1,1,1)	1×8	64
AP_5	(2,1,1,1,1,2)	2×4	32
AP_6	(2,1,1,1,1,2)	2×4	64
AP_7	(2,1,1,1,1,2)	1×8	32
AP_8	(2,1,1,1,1,2)	1×8	64
AP_9	(1,1,1,1,2,2)	2×4	32
AP_{10}	(1,1,1,1,2,2)	2×4	64
AP_{11}	(1,1,1,1,2,2)	1×8	32
AP_{12}	(1,1,1,1,2,2)	1×8	64

Plataforma TORC. $p = 8, P = 19, n = 4096$

- TORC: 8 Intel duales (P_0 - P_{15}), 1 Intel Pentium III a 600 MHz (P_{16}),1 AMD Athlon (P_{17}) y 1 Intel Pentium 4 a 1.7 GHz (P_{18}).

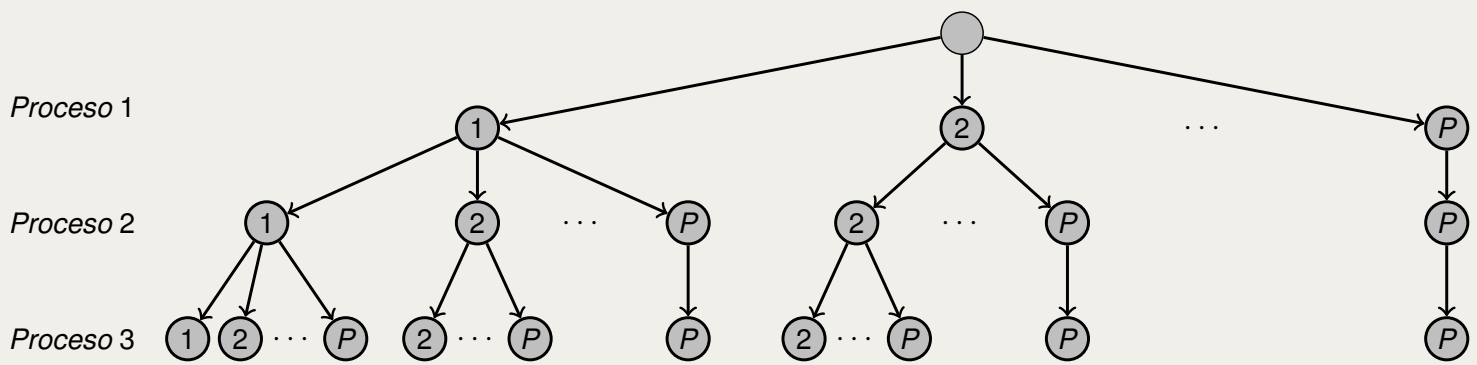


	Asignación $p = 8, P = 19$ $d = (P_0, ..., P_{18})$	$p = r \times c$	b
AP_1	(1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,0)	4×2	32
AP_2	(1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,0)	8×1	32
AP_3	(1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,2,0)	4×2	32
AP_4	(1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,2,0)	8×1	32
AP_5	(1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,2,2,1)	4×2	32
AP_6	(1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,2,2,1)	8×1	32
AP_7	(1,0,1,0,1,0,1,0,1,0,0,0,0,0,2,0,0)	4×2	32
AP_8	(1,0,1,0,1,0,1,0,1,0,0,0,0,0,2,0,0)	8×1	32

Árbol de asignación

Planteamiento

- El espacio de posibles soluciones se representa por un árbol.
- Cada nivel corresponde a un proceso y cada nodo, uno de los P procesadores.
- Árbol más pequeño. Pero hay que encontrar la mejor topología lógica ($r \times c$).
- Función a optimizar tiempo de ejecución de la rutina.
- $T = f(n, AP, SP)$. Valor de los SP para D y d. Encontrar el resto de AP: tamaño de bloque y topología lógica 2D.



Estimaciones en cada nodo del árbol

Búsqueda de una solución

- Encontrar una solución general resultaría costoso. Estrategias para eliminar nodos en el árbol.
- Cada nodo del árbol tiene asociadas tres estimaciones:
  - $EET(nodo)$
  - $LET(nodo)$
  - $GET(nodo)$
- $LET(nodo)$ y $GET(nodo)$. Cota inferior y superior para la solución óptima de sus descendientes. Limitan número de nodos evaluados y altura del árbol.
- *Backtracking* o *Branch and Bound*:
  - $MEET = \min_{nodo \in \text{nodos evaluados}} GET(nodo)$
  - Si $LET(nodo) \geqslant MEET$ no se continuará la búsqueda.
- Encontrar el menor valor para $EET(nodo)$.
- Donde $EET(nodo)$ sería $T = f(n, AP, SP)$

Estrategias de búsqueda automática

Método de Backtraking (**BTM**)

- $GET = EET$.
- $LET = LET_{ari} + LET_{com}$.

Método de Backtraking con Greedy (**BGRM**)

- *GET* se obtiene con esquema *greedy*. Descendiente con menor *EET*.
- *LET* como en **BTM**.

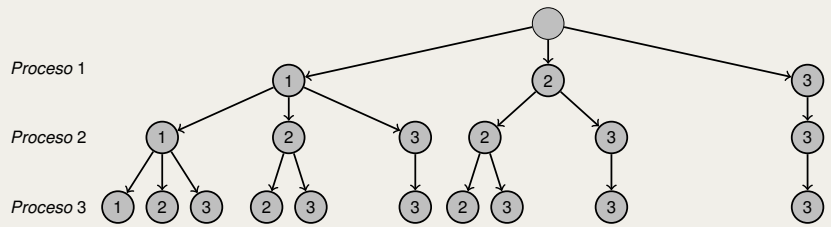
Método Greedy (**GRM**)

- *GET* se obtiene con esquema *greedy*.
- *LET* se obtiene con esquema *greedy*. Descendiente que menos incremente el coste aritmético.

Método Greedy con árbol Combinatorio (**GCTM**)

- Como en **BGRM** pero con búsqueda hasta alcanzar nivel máximo en el árbol combinatorio con repeticiones.

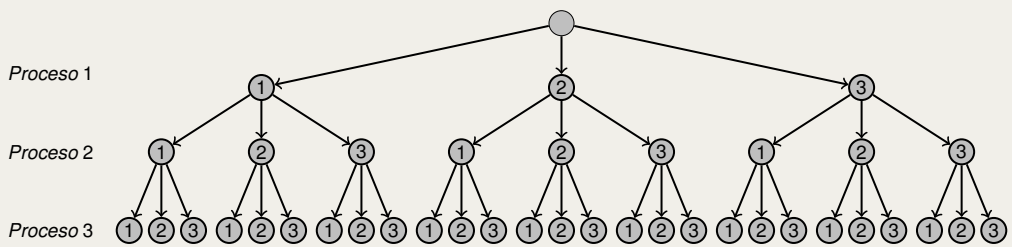
Árbol Combinatorio con repeticiones



Método Greedy con árbol Permutacional (**GPTM**)

- Esquema *greedy* con árbol de permutaciones con repeticiones.

Árbol permutacional con repeticiones



LU. Comparación selección automática y usuarios

Plataforma SUNEt. $n = 7680$

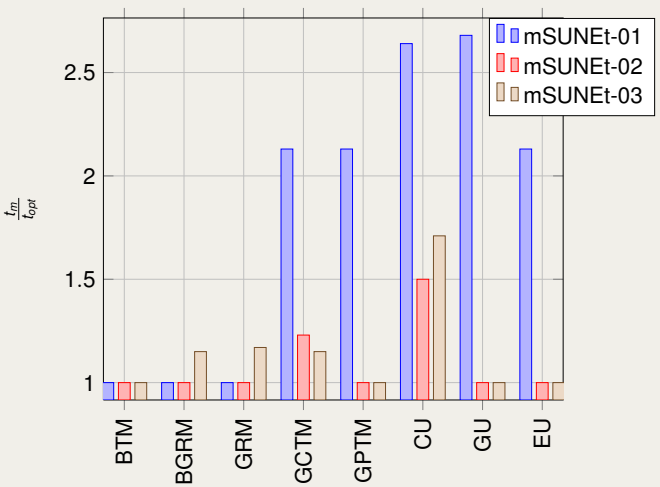
Método	asignación de procesos a procesadores $d = (P_0, \ldots, P_5)$	b	topología lógica	tiempo solución	t.e.s	nivel
BTM	(1,1,1,1,1,1)	64	2×3	718.94	0.02	25
BGRM	(1,1,1,1,1,1)	64	2×3	718.94	0.04	25
GRM	(1,1,1,1,1,1)	64	2×3	718.94	0.02	25
GCTM	(1,1,0,0,0,1)	128	1×3	887.85	0.0001	25
GPTM	(1,1,0,0,0,1)	128	1×3	887.85	0.0005	25
CU	(1,1,0,0,0,1)	128	1×3	1047.13		
GU	(1,1,1,1,1,1)	64	2×3	887.85		
EU	(1,1,1,1,1,1)	64	2×3	887.85		

Plataforma TORC. $n = 2048$

Método	asignación de procesos a procesadores $d = (P_0, \ldots, P_{18})$	b	topología lógica	tiempo solución	t.e.s	nivel
BTM	(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0)	64	3×5	17.91	3.08	15
BGRM	(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0)	64	3×5	17.91	3.08	15
GRM	(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0)	64	4×4	15.27	0.06	25
GCTM	(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0)	64	1×1	43.16	0.0012	30
GPTM	(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0)	64	4×4	15.27	0.01	30
CU	(1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1)	64	3×3	23.77		
GU	(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)	32	1×19	33.57		
EU	(1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1)	64	3×3	23.77		

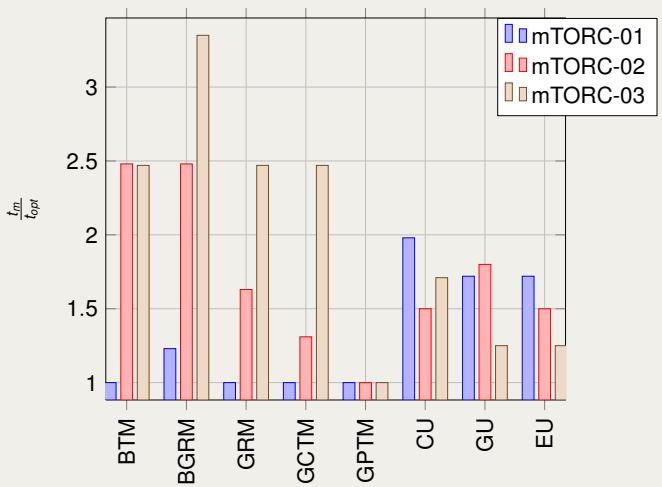
LU. Comparación selección automática y usuarios

Plataformas simuladas SUNEt. $n = 20000$



Sistema	Método	t.e.s	nivel
mSUNEt-01	BTM	3.7	20
	BGRM	0.02	20
	GRM	0.03	25
	GCTM	0.0005	25
	GPTM	0.0025	25
mSUNEt-02	BTM	46.7	20
	BGRM	88.4	20
	GRM	0.5	20
	GCTM	0.0004	25
	GPTM	0.004	25
mSUNEt-03	BTM	110.9	20
	BGRM	251.1	20
	GRM	0.04	25
	GCTM	0.0005	25
	GPTM	0.006	25

Plataformas simuladas TORC. $n = 20000$



Sistema	Método	t.e.s	nivel
mTORC-01	BTM	20.4	15
	BGRM	59.5	15
	GRM	0.7	20
	GCTM	0.0007	25
	GPTM	0.012	25
mTORC-02	BTM	259.4	15
	BGRM	792.3	15
	GRM	7.5	25
	GCTM	0.01	30
	GPTM	0.07	30
mTORC-03	BTM	109.2	10
	GRM	169.7	10
	GRBM	1274.3	5
	GCTM	0.08	25
	GPTM	2.3	40

Índice

Propósito

- Ejecución eficiente de rutinas de álgebra lineal con código multithread.
- Selección del código generado por los diferentes compiladores OpenMP disponibles.
- Ejecutar la versión compilada con mejores tiempos de ejecución.

Justificación

Compiladores con diferentes capacidades:

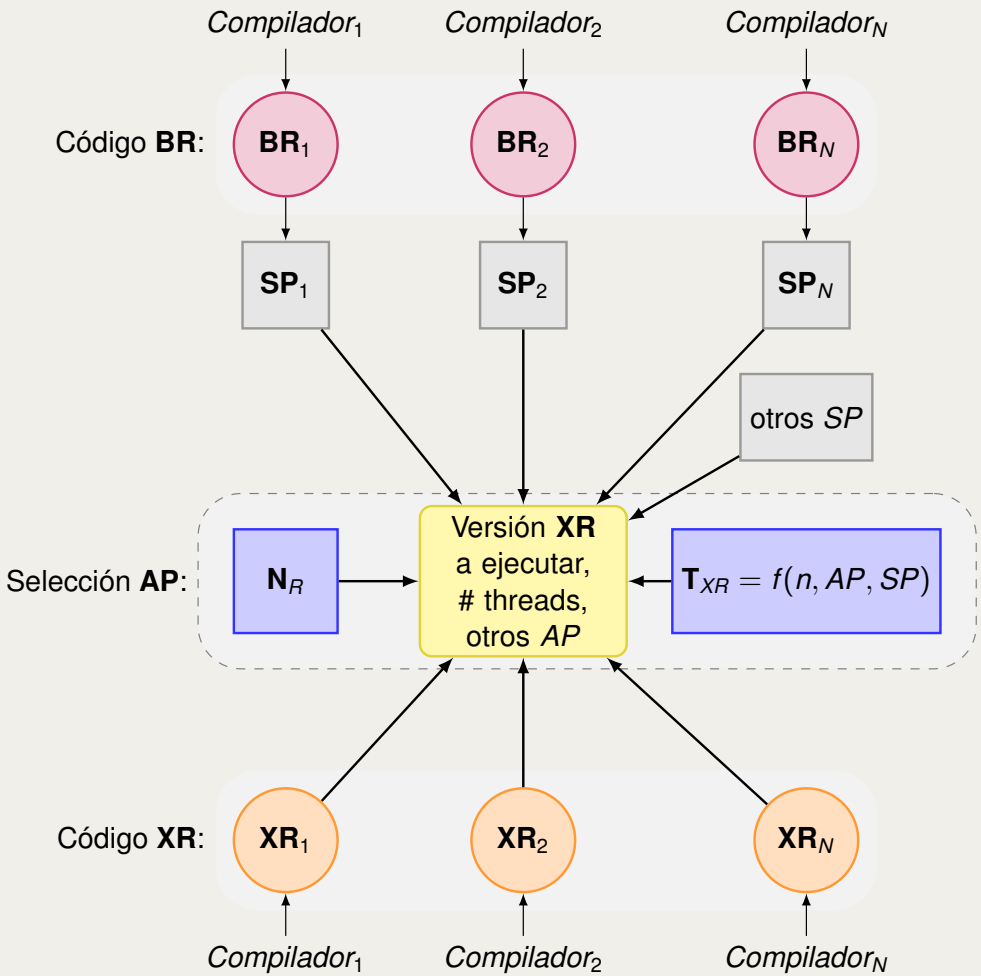
- Un compilador genera código secuencial de forma eficiente.
- Otro optimizará mejor el uso de múltiples threads.
- El código generado se ejecuta eficientemente si el nº theads < cores
- El rendimiento decrezca si nº threads > cores.

Propuesta de selección

Propuesta

- Modelado de rutinas con un nuevo *AP*: elección de la versión del código generada por cada compilador.
- Comparar las capacidades de los compiladores en la generación de código multithread.
- Conjunto de rutinas de *benchmarking* (*BR*).
- Llamadas a directivas OpenMP.

Metodología propuesta



Esquema para una rutina XR

- Plataforma con varios compiladores ($Compilador_1$, $Compilador_2$,..., $Compilador_N$)
- Se obtiene para cada compilador una versión de las BR (BR_1, BR_2,..., BR_N)
- La ejecución de las BR caracteriza las capacidades de gestión y generación de threads de cada compilador, por medio de los correspondientes SP (SP_1, SP_2,..., SP_N), junto con otros SP que aparecen en rutinas de álgebra lineal.
- El $T_{XR} = f(n, AP, SP)$ de la rutina XR multithread, para un tamaño de problema N_R, permitirá realizar la selección de la mejor versión de XR (XR_1, XR_2,..., XR_N), el número de threads a generar, y otros AP de la rutina.

Rutinas de *benchmarking*

Descripción

- Rutinas básicas con las que se compara las capacidades de los compiladores OpenMP en la creación y gestión de threads:
  - **R-generate**
    - Crea una serie de threads con una cantidad fija de trabajo por thread.
    - El propósito es comparar el tiempo de **creación y gestión** de threads.
  - **R-pfor**
    - Consiste en paralelizar un bucle `for` sencillo. En cada iteración se realiza una cantidad de trabajo significativo.
    - Comparar el tiempo empleado en **distribuir dinámicamente** un conjunto de tareas homogéneas entre threads.
  - **R-barriers**
    - Imponer una primitiva de barrera, a continuación de una área de trabajo paralelo.
    - Comparar el tiempo en realizar una **sincronización global** de todos los threads.

Modelo tiempo de ejecución rutinas *BR*. **R-generate**

Rutina *R-generate*

- número de threads $\leqslant$ número de cores disponibles

$$T_{R-generate} = PT_{gen} + NT_{work}$$

- número de threads $>$ número de cores disponibles

$$T_{R-generate} = PT_{gen} + NT_{work}\frac{P}{C}\left(1 + \frac{T_{sw}}{T_{cpu}}\right)$$

Parámetros OpenMP

- T_{gen}: tiempo en generar un thread.
- T_{work}: tiempo unitario de trabajo.
- T_{sw}: tiempo necesario en realizar un cambio de contexto.
- T_{cpu}: tiempo de CPU asignado a cada thread entre dos cambios de contexto.

Modelo tiempo de ejecución rutinas *BR*. **R-pfor**

Rutina *R-pfor*

- número de threads $\leqslant$ número de cores disponibles

$$T_{R-pfor} = PT_{gen} + \frac{N_t}{P} T_{work}$$

- número de threads $>$ número de cores disponibles

$$T_{R-pfor} = PT_{gen} + \frac{N_t}{C} T_{work} \left(1 + \frac{T_{sw}}{T_{cpu}}\right)$$

Parámetros OpenMP

- C: número de cores.
- N_t: tamaño total del problema.

Modelo tiempo de ejecución rutinas *BR*. **R-barriers**

Rutina *R-barriers*

- número de threads $\leqslant$ número de cores disponibles

$$T_{R-barriers} = PT_{gen} + NT_{work} + PT_{syn}$$

- número de threads $>$ número de cores disponibles

$$T_{R-pfor} = PT_{gen} + NT_{work}\frac{P}{C}\left(1 + \frac{T_{sw}}{T_{cpu}}\right) + PT_{syn}$$

Parámetros OpenMP

- T_{syn}: tiempo de sincronización por thread.

Modelo de tiempo de ejecución de la rutina OpenMP de Strassen

número de threads generados $\leq$ número de cores disponibles

$$T = PT_{gen} + 7\frac{2(\frac{n}{2})^3}{P}T_{mul} + \frac{9n^2}{4}T_{add}$$

$$T = P_1P_2T_{gen} + 49\frac{2(\frac{n}{2^2})^3}{P_2P_1}T_{mul} + 7\frac{\frac{18n^2}{4^2}}{P_1}T_{add} + \frac{9n^2}{2}T_{add}$$

número de threads generados $>$ número de cores disponibles

$$T = PT_{gen} + 7\frac{2(\frac{n}{2})^3}{C}T_{mult}\left(1 + \frac{T_{sw}}{T_{cpu}}\right) + \frac{9n^2}{4}T_{add}$$

$$T = P_1P_2T_{gen} + 49\frac{2(\frac{n}{2^2})^3}{C}T_{mul}\left(1 + \frac{T_{sw}}{T_{cpu}}\right) +$$
$$7\frac{\frac{18n^2}{4^2}}{\min(P_1, C)}T_{add}\left(1 + \frac{T_{sw}}{T_{cpu}}\right) + \frac{9n^2}{2}T_{add}$$

Medición de los *SP*

Estimación valores de los *SP* en diferentes plataformas y con diferentes compiladores

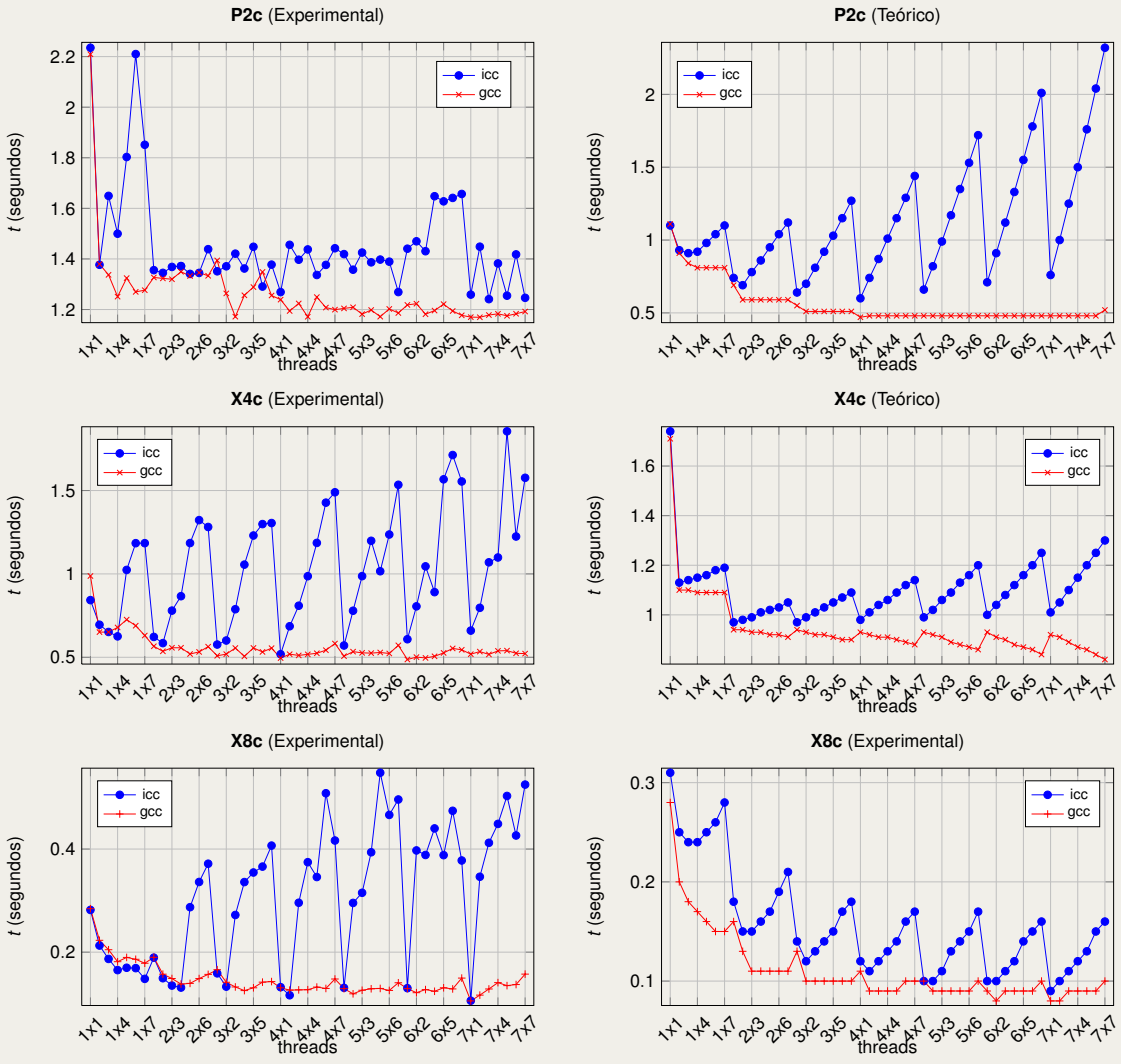
Plataforma		T_{gen} (μs)	$\frac{T_{sw}}{T_{cpu}}$	T_{add} (ns)	T_{mul} (ps)
P2c	icc	75	$2 + 0.01P$	$20 + 0.05P$	$400 + 100P$
	gcc	25	$7 - 0.01P$	20	$400 + 0.1P$
X4c	icc	75	$0.9 + 0.3P$	$23 + 0.3P$	$140 + 10P$
	gcc	25	$0.9 + 0.01P$	$30 - 0.3P$	$140 - P$
A4c	cc	75	$0.8 + 0.2P$	$40 + P$	60
	gcc	25	$0.8 + 0.02P$	$40 - 0.1P$	$60 - 0.5P$
X8c	icc	75	$6 + 0.05P$	10	100
	gcc	25	$0.5 + 0.01P$	10	100

T_{sw} / T_{cpu}. Coste cambio de contexto

- Su valor depende del número de threads P. Nueva *BR* para su cálculo. Adición de vectores con una cantidad de trabajo fijo y variando P.
- Se obtiene el % de incremento entre tiempo experimental y tiempo teórico de ésta *BR*.

Comparación tiempos experimentales y teóricos

Rutina R-strassen con $n = 1000$ y $P_1 \times P_2$ threads



Selección de los *AP*

Selección P_1, P_2 y código compilado

	P2c	X4c	A4c	X8c
versión compilada	gcc	gcc	gcc	gcc
número threads P_1	7	4	4	7
número threads P_2	7	1	1	2

Tiempos de ejecución de R-strassen con $n = 1000$

Plataforma	OP	MOD	HW	SW	dev MOD (%)	dev HW (%)	dev SW (%)
P2c	1.17	1.19	1.37	1.22	2	17	4
X4c	0.49	0.50	0.55	1.31	2	12	167
A4c	0.45	0.49	0.65	1.20	9	44	167
X8c	0.11	0.16	0.12	0.32	45	9	191

Índice

Conclusiones

- Las mejoras propuestas permiten abordar el desarrollo de técnicas de optimización automática y ajuste del código modelado.
- Selección adecuada de los parámetros ajustables. Selección del mejor algoritmo y selección de la mejor librería básica.
- El uso de técnicas de remodelado permite obtener modelos cuando el modelo analítico no permite la elección correcta de los *AP* o no puede ser obtenido.
- La metodología de modelado junto a estrategias de asignación de procesos a procesadores permite ejecutar una rutina homogénea en un sistema heterogéneo, sin necesidad de modificar el código de la misma.
- La elección automática de la mejor versión del ejecutable de una rutina con código OpenMP mejora las prestaciones del código de álgebra lineal multithread para plataformas multicore.

Trabajo futuro

- Estudio del remodelado con otras funciones (spline discretas) y mejora incremental de la estimación obtenida.
- Aplicación de metodologías de selección de modelos como el Criterio de Información de Akaike (AIC).
- Optimización automática de rutinas con paralelismo anidado OpenMP y BLAS multithread.
- Desarrollar modelos para rutinas que se ejecuten en sistemas híbridos (multicore y GPU).
- Aplicación de la metodología de modelado a librerías diseñadas para plataformas heterogéneas como HeteroScaLAPACK.
- Métodos metaheurísticos como búsqueda dispersa, algoritmos genéticos, GRASP (*Greedy Randomized Adaptative Search Procedure*). Analizar la posibilidad de aplicar de forma combinada las técnicas metaehurísticas mencionadas.