Técnicas de Auto-optimización de Rutinas Básicas de Álgebra Lineal en Sistemas Multicore+MultiGPU

Francisco José Herrera Zapata

Trabajo dirigido por:

Antonio Javier Cuenca Muñoz
Domingo Giménez Cánovas

Codirector:
Luis Pedro García González (UPCT)

Universidad de Murcia - Facultad Informática
sep, 2017

- Introducción
- Multiplicación de Matrices
- Técnicas de auto-optimización en la Multiplicación de Matrices
- Experimentos Multicore+multiGPU
- Conclusiones y Trabajos Futuros

- Introducción
- Multiplicación de Matrices
- Técnicas de auto-optimización en la Multiplicación de Matrices
- Experimentos Multicore+multiGPU
- Conclusiones y Trabajos Futuros

Sistema utilizado en este trabajo:

- 1 nodo con dos hexa-cores Intel Xeon E5-2620, a 2.00 GHz, con 32 GB de memoria RAM y 6 GPUs. Agrupadas en 2 GPUs NVIDIA Fermi Tesla C2075 y 4 GPUs NVIDIA GTX 590. Nodo perteneciente al clúster Heterosolar de la UMU.

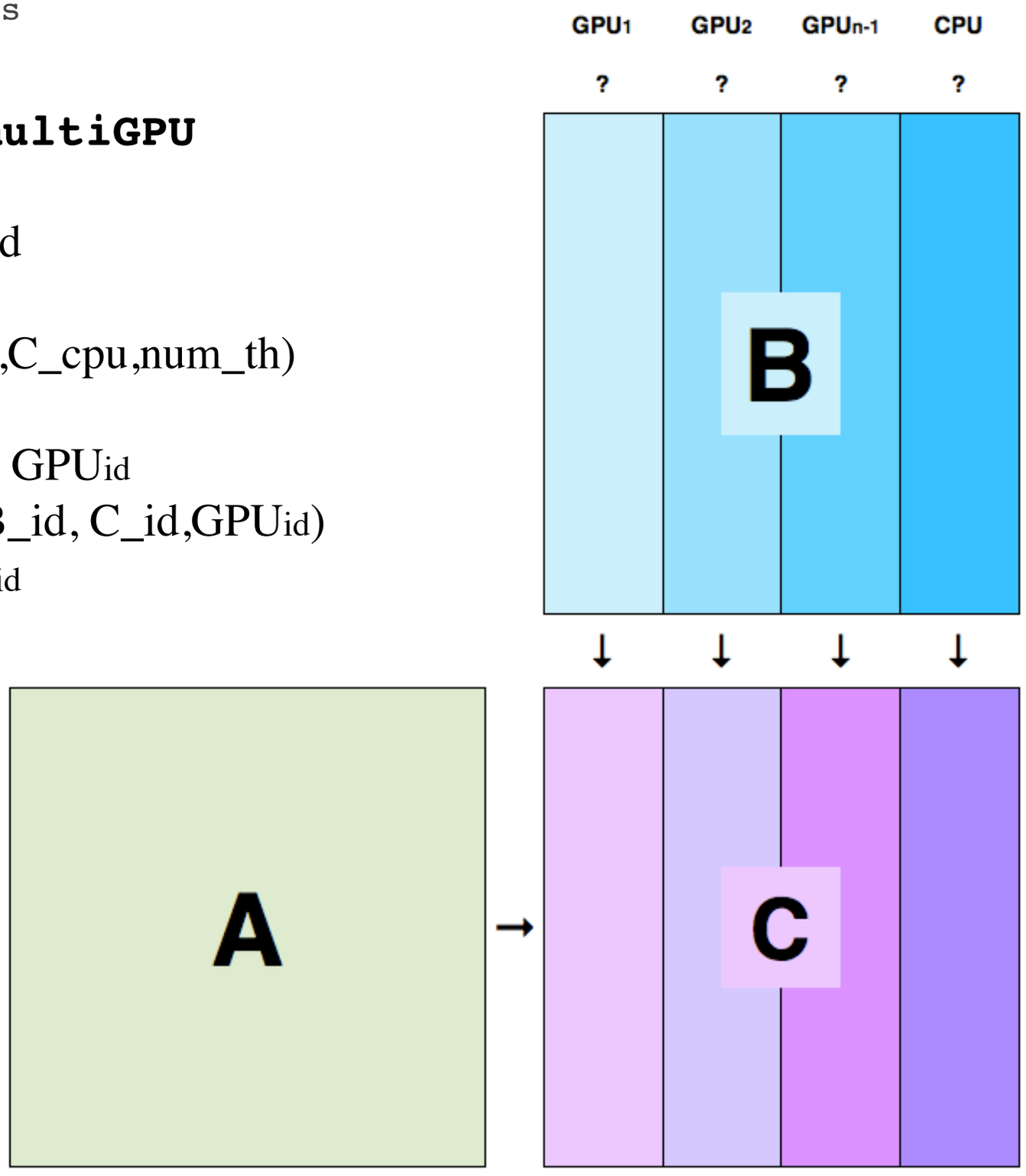
- 1 nodo de 16 Intel Xeon 64GB RAM y 2 coprocesadores GPU NVIDIA Tesla K40m. Dentro del clúster Prometeo de la UPCT.

- Librerías BLAS:
  - Intel MKL multicore.
  - cuBLAS GPU.

- Introducción
- **Multiplicación de Matrices**
- Técnicas de auto-optimización en la Multiplicación de Matrices
- Experimentos Multicore+multiGPU
- Conclusiones y Trabajos Futuros

MM multicore+multiGPU

obtenemos nuestro tid
si (tid==0)
  mm_mkl(A,B_cpu,C_cpu,num_th)
sino si (tid!=0)
  enviar A, B_{id}, C_{id} a GPU_{id}
  mm_cuBLAS(A, B_id, C_id,GPU_{id})
  recibir C_{id} de GPU_{id}
finsi



$$t(n,p,GPU) = Máx\{mm\_cuBlas(B_1),\ldots,mm\_cuBlas(B_{n-1}),mm\_mkl(B_n)\}$$

<N, gpu_1, gpu_2, … , gpu_{n-1}, CPU,th>

- Introducción
- Multiplicación de Matrices
- **Técnicas de auto-optimización en la Multiplicación de Matrices**
- Experimentos Multicore+multiGPU
- Conclusiones y Trabajos Futuros

Técnicas de auto-optimización MM. Proceso Instalación.

Conjunto Instalación	Búsqueda	Configuración óptima
n_0	Exhaustiva - Guiada	<n_0,gpu1,...gpu$_{n-1}$,cpu,th>
n_1		<n_1,gpu1,...gpu$_{n-1}$,cpu,th>
...		...
n_m		<n_m,gpu1,...gpu$_{n-1}$,cpu,th>

Parámetros de entrada:

- Conjunto instalación.
- tb Tamaño del bloque reparto.
- Número de threads disponible.

Salida:

- Rutina auto-optimizada MM con librerías BLAS.

Técnicas de auto-optimización MM.
Uso rutina auto-optimizada MM con librerías BLAS.

auto_optimizada_MM_BLAS(A,B → C)

- Seleccionar la configuración óptima de las obtenidas en el proceso de instalación que mejor se ajuste.

- Realizar la operación MM con la configuración seleccionada y los datos de entrada.

Búsqueda Exhaustiva

- Explorar **todo** el árbol de soluciones y así obtener la configuración **óptima** para el tamaño **n1**.

- El proceso de instalación con búsqueda exhaustiva se realiza sobre todos los elementos del conjunto de instalación.

- Algoritmo de backtracking, búsqueda en profundidad y retroceso.

- Problema: Coste computacional muy elevado. Inasumible en la práctica.

Búsqueda Exhaustiva

Ejemplo del Problema:

- **m = 9216** Tamaño de la matriz del elemento **nI.**

- **tb = 192** Tamaño del bloque de reparto.

- **k = 48 = m/tb** bloques que se reparten entre las unidades de cómputo.

- **n = 7** Número de dispositivos de cómputo. 6 GPUs + 1 CPU.

$$busqueda\_completa(m, tb, k, n) = \binom{n+k-1}{n}$$

≈ 26 Millones Comprobaciones
Y sólo es un nI del conjunto instalación.

Búsqueda Guiada

- Reducir el número de comprobaciones explorando solamente aquellos nodos del árbol de soluciones que creamos que contienen configuraciones óptimas.

- Rebajar el tiempo de instalación, prohibitivo en el caso de realizar una instalación complenta usando búsqueda exhaustiva.

- Con ayuda de metaheurística guiar al algoritmo hacia soluciones potencialmente óptimas.

- Búsqueda local por ascensión de colinas utilizando posiciones tabú.

- Utilizar como condición de parada estar por debajo de un **umbral** determinado respecto a la mejor configuración.

Búsqueda Guiada

Instalación_Guiada{n_1,n_2,…,n_m}

Busqueda_exahustiva(n1) -> v_conf_best$_1$

Para i: 2..m

InciarBúsqueda(v_conf_best$_{i-1}$) -> v_conf_best$_i$ -> v_conf$_i$

hacer

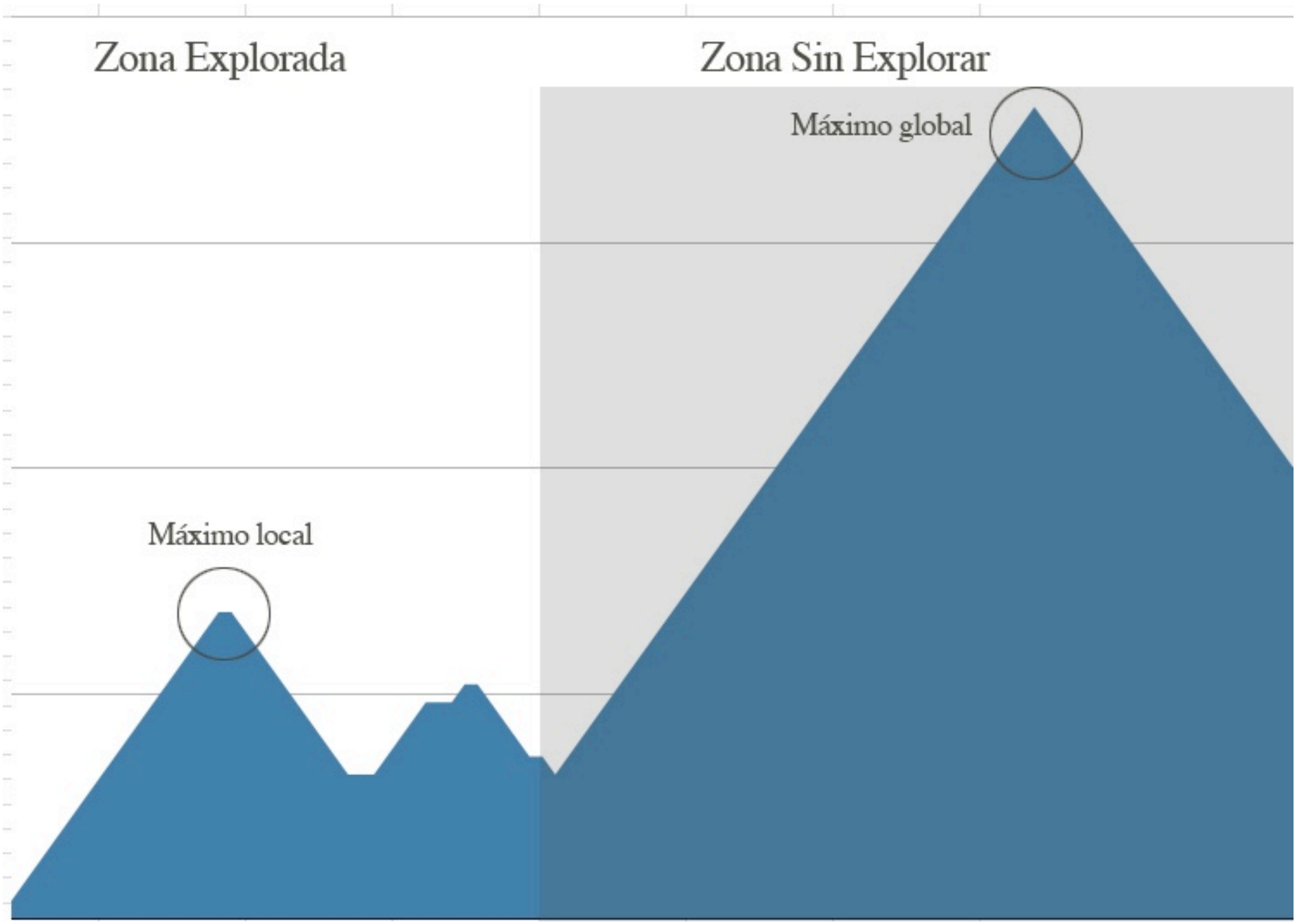
mejorVecino(v_conf$_i$)-> v_conf$_i$

si **(v_conf$_i$** > v_conf_best$_i$)

v_conf$_i$-> v_conf_best$_i$

mientras (v_conf$_i$ < **umbral)**

Búsqueda Guiada



- Introducción
- Multiplicación de Matrices
- Técnicas de auto-optimización en la Multiplicación de Matrices
- **Experimentos Multicore+multiGPU**
- Conclusiones y Trabajos Futuros

Tiempos de Instalación

Conjunto instalación homogénea = {768, 1536, ..., 10752}
conjunto instalacion heterogénea = {768, 1536, ..., 7680}
tb=192
th_homogéneo = 16
th_heterogéneo =12

Instalación Homogénea Multicore + 2 GPUs		
	Random	**Completa**
2%	4’17”	5’39”
5%	3’45”	9’33”
10%	4’14”	9’18”
Instalación Heterogénea Multicore + 6 GPUs		
	Random	**Completa**
2%	3’27”	17’01”
5%	5’30”	15’01”
10%	6’29”	1h 05’04”

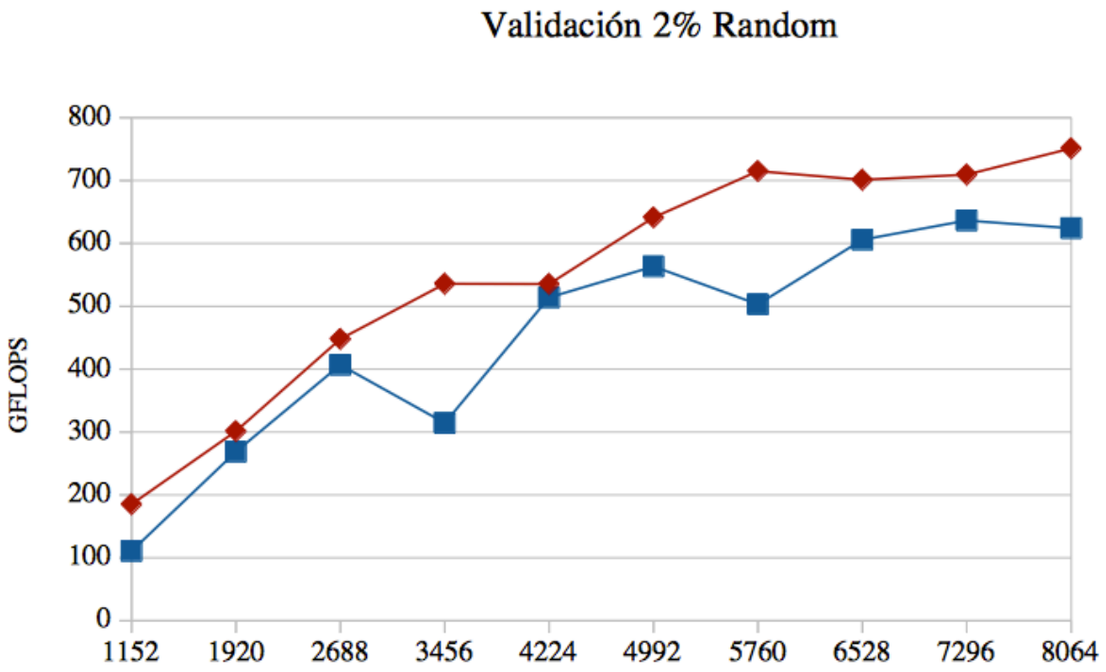
Proceso de Validación

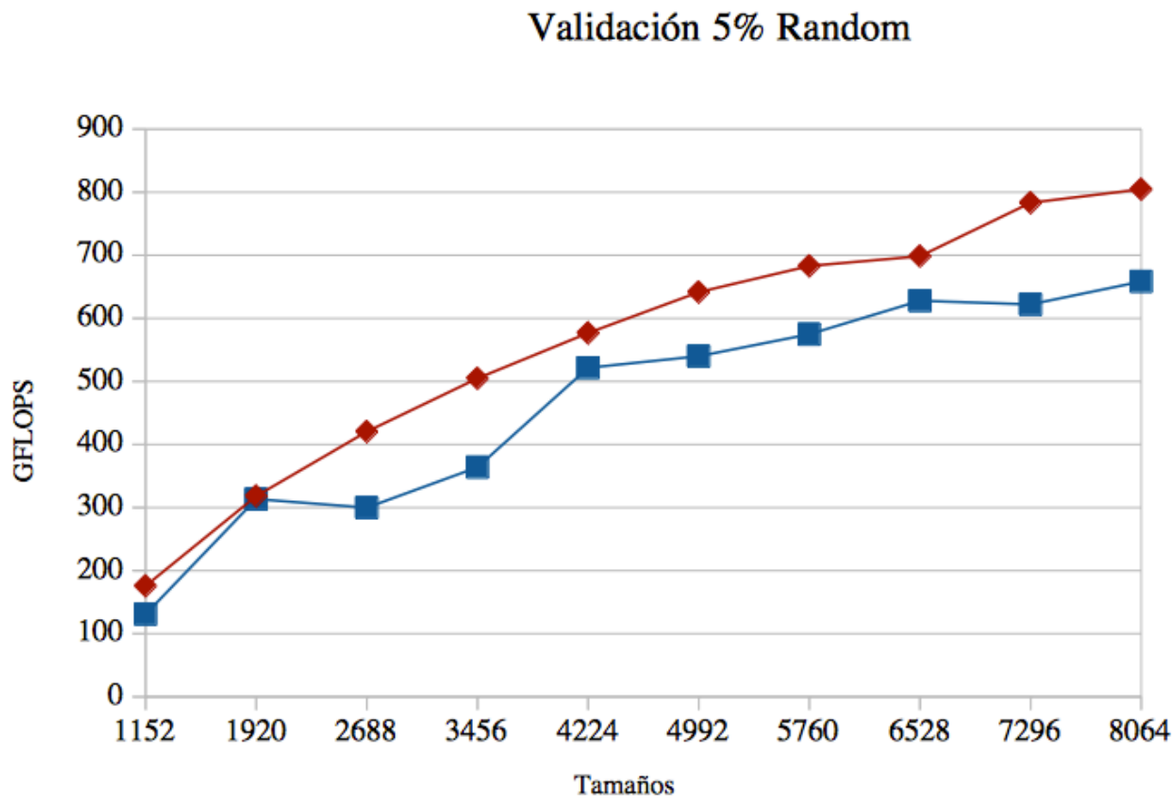
1. Instalación guiada *{conjunto instalación, tb, th}* → rutina MM auto-optimizada.

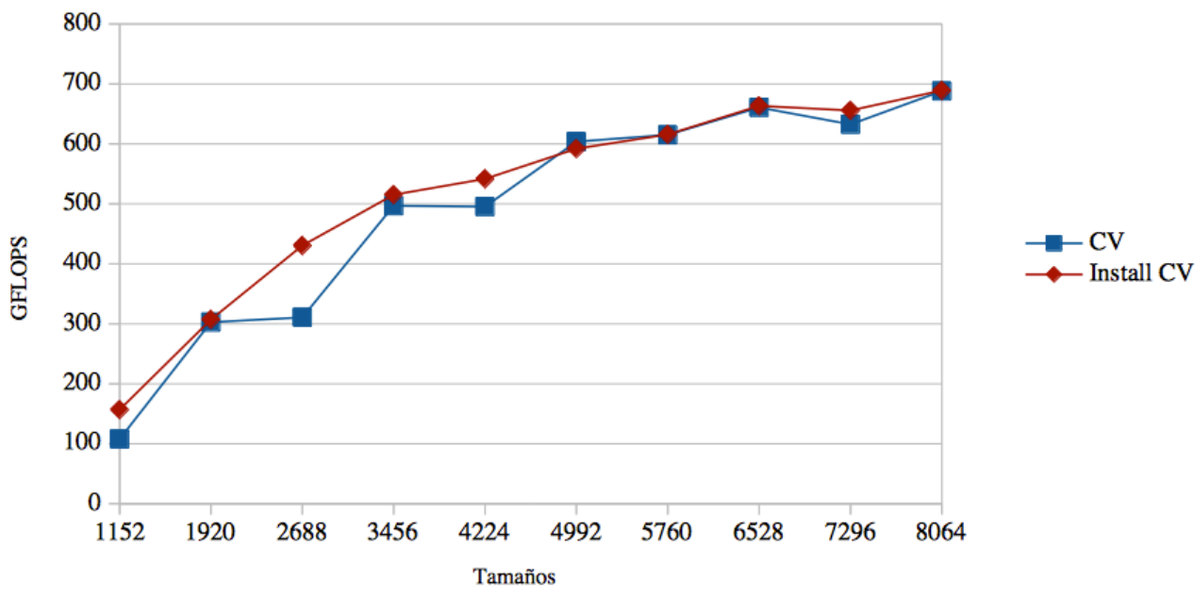
2. MM auto-optimizada sobre un conjunto de datos **cv**. Midiendo el rendimiento de la operación para cada tamaño de **cv**.

3.Comparar rendimiento paso 2 con una configuración óptima para el mismo conjunto de datos. Esta configuración óptima se obtiene realizando el proceso de instalación sobre este conjunto de datos **install_cv**.

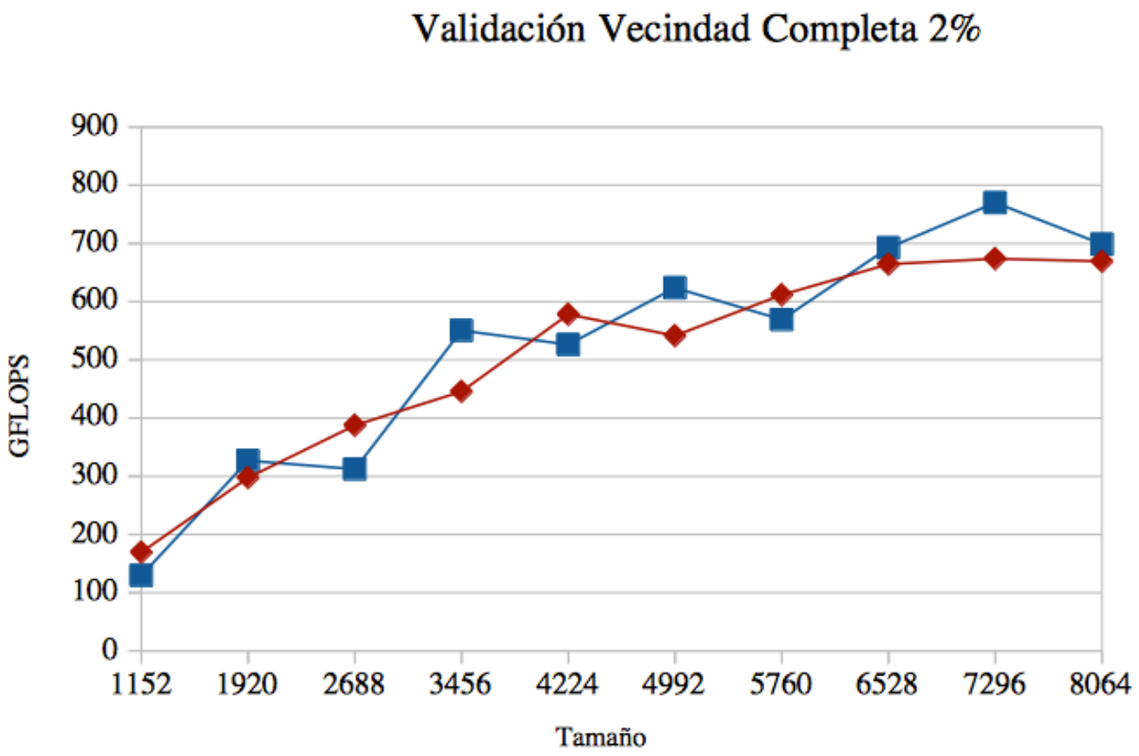
Proceso de Validación - Vecindad Aleatoria

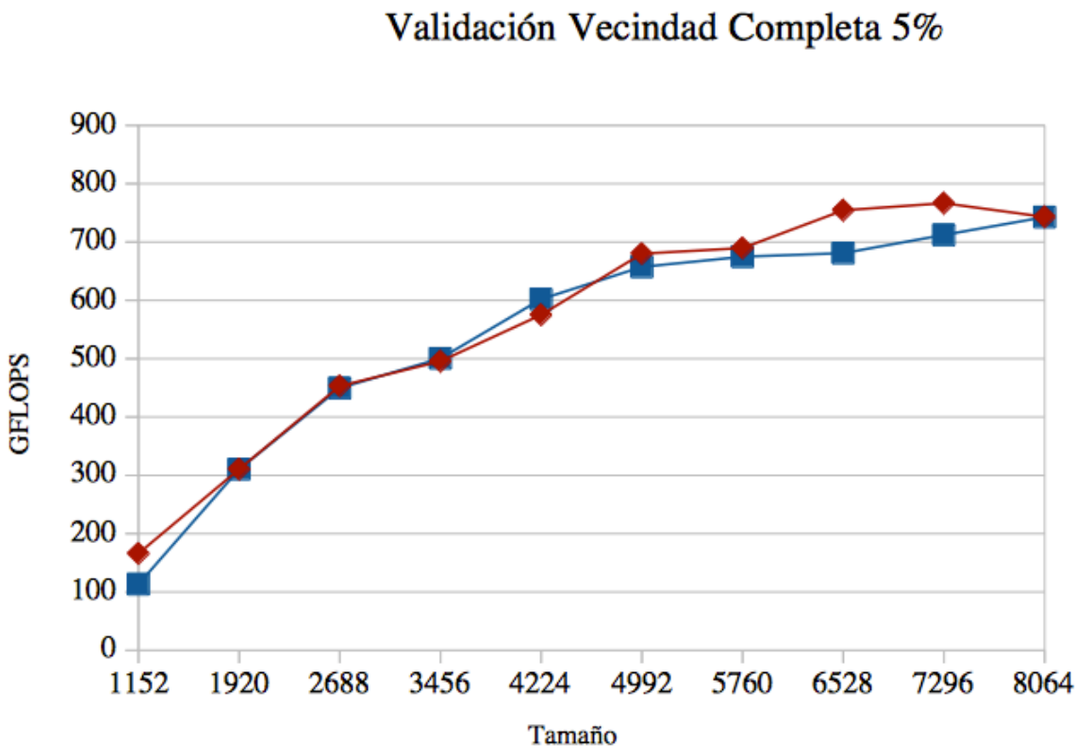


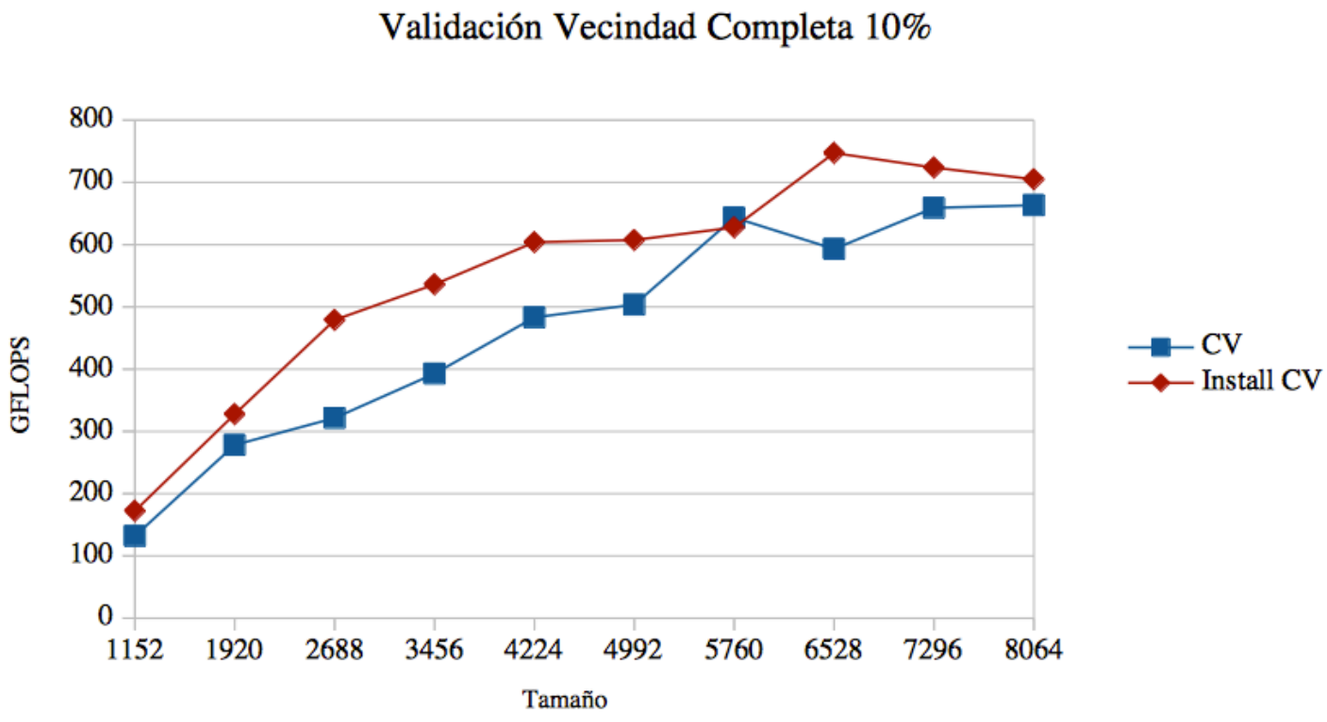




Proceso de Validación - Vecindad Completa







- Introducción
- Multiplicación de Matrices
- Técnicas de auto-optimización en la Multiplicación de Matrices
- Experimentos Multicore+multiGPU
- **Conclusiones y Trabajos Futuros**

Conclusiones

- Ya que los sistemas cada vez son más grandes y heterogéneos.
- Estos necesitan de ayuda software para obtener el mayor rendimiento de los sistemas.
- Donde las librerías BLAS no son de suficiente ayuda, optimizan las operaciones dentro del dispositivo de cómputo, pero, no son capaces de distribuir el trabajo según la capacidad de cómputo de cada dispositivo

Las ***rutinas de auto-optimización**, que sí son capaces de distribuir el trabajo con una simple instalación previa, han demostrado ser una gran herramienta en este sentido.

* Javier Cuenca, Luis-Pedro García, Domingo Giménez, and Francisco-José Herrera. Guided installation of basic linear algebra routines in a cluster with manycore components. Concurrency and Computation: Practice and Experience, 29(15), 2017.

Trabajos Futuros

- Utilizar esta multiplicación de matrices ya optimizada sobre rutinas de un nivel superior y estudiar el aumento de rendimiento.

- Utilizar técnicas de auto-optimización sobre otro tipo de rutinas, tal vez de mayor nivel.

- Utilizar esta técnica de auto-optimización de la operación multiplicación matricial en un sistema totalmente heterogéneo compuesto por multicore+multiGPU+multi- MIC.

- Y por último, adaptar las técnicas a clusters con múltiples nodos multicore+multi- GPU+ multiMIC.

Muchas Gracias. ¿Preguntas?