Factores de Rendimiento en Entornos Multicore

César Allande Álvarez

callande@caos.uab.es

Computer Architecture & Operating Systems Department (CAOS)
Barcelona, Spain

Director: Eduardo César Galobardes

14 de julio de 2010

Índice

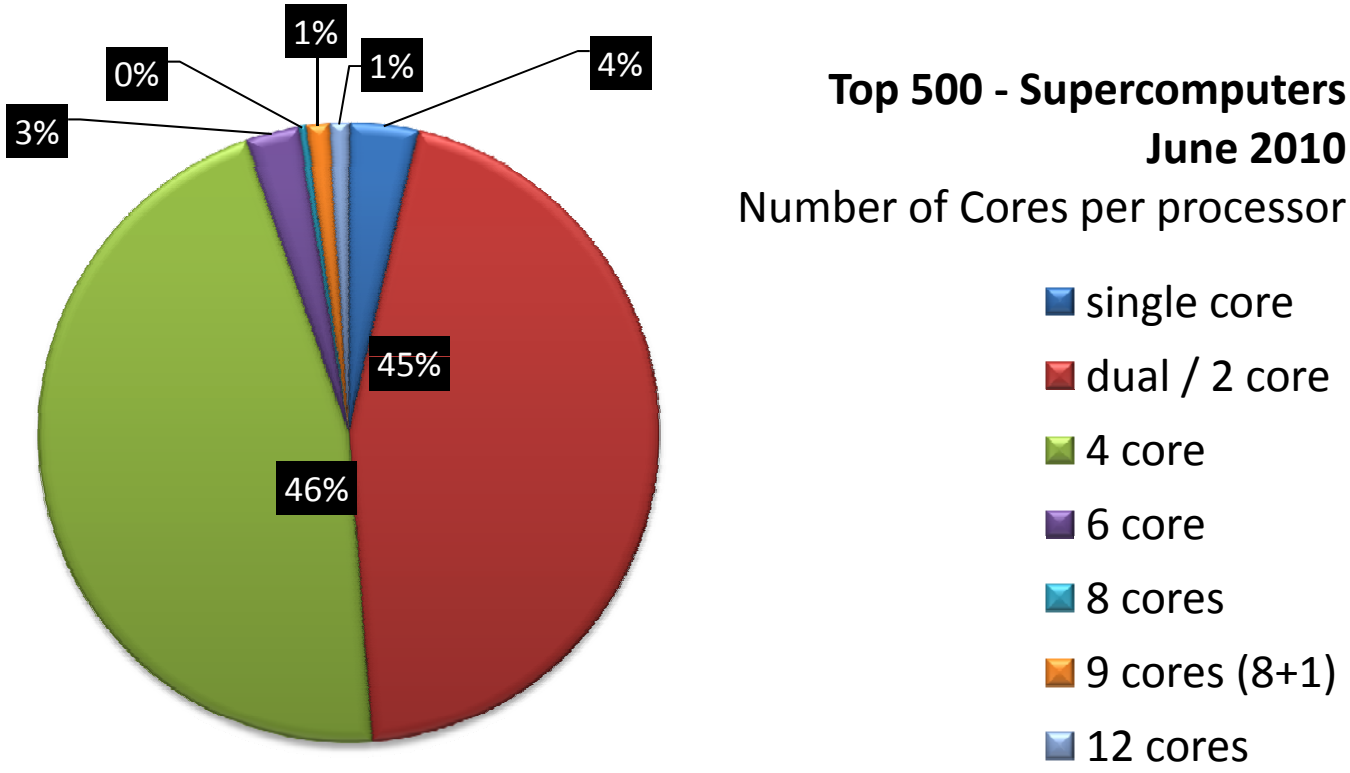
- Introducción
- Objetivos
- Desarrollo del Estudio
- Análisis y Experimentación
- Conclusiones
- Trabajo actual y Líneas abiertas

Introducción

- ¿Qué?
  - Integración de sistemas multicore en HPC
- ¿Por qué?
  - Mejora de eficiencia, rendimiento y consumo
- ¿Cómo?
  - Sintonizando aplicaciones paralelas
    - Teniendo en cuenta el modelo de programación
    - Teniendo en cuenta la arquitectura
    - Considerando el patrón de comportamiento de la aplicación

Introducción – ¿ Multicore en HPC ?

- Multicore está implantado en HPC



Introducción

Ley de Moore

- La potencia de los ordenadores se duplica cada dos años, reduciendo además su coste.

Ley de Amdahl

- El incremento de velocidad de un programa utilizando múltiples procesadores en computación distribuida está limitada por la fracción secuencial del programa

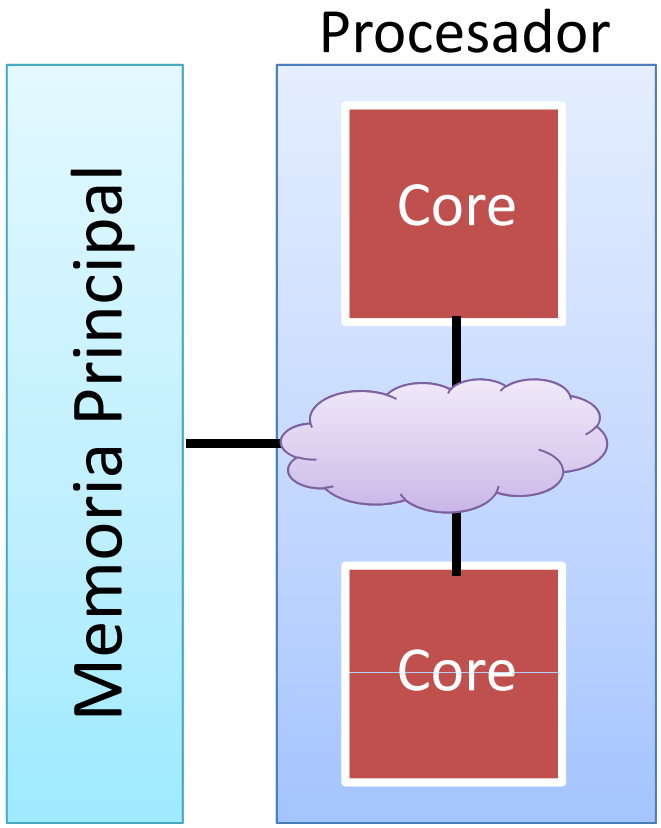
Ley de Gustafson

- Cualquier problema suficientemente grande puede ser eficientemente paralelizado

Kill Rule

- Se debe incrementar el tamaño del recurso solamente si por cada 1% de aumento del área se obtiene al menos un 1% de mejora de rendimiento del core

Introducción – Taxonomía multicore



Arquitectura

- Multicore vs. Manycore
- Homogénea / Heterogénea

Interconexión

- Comunicación
- Topología

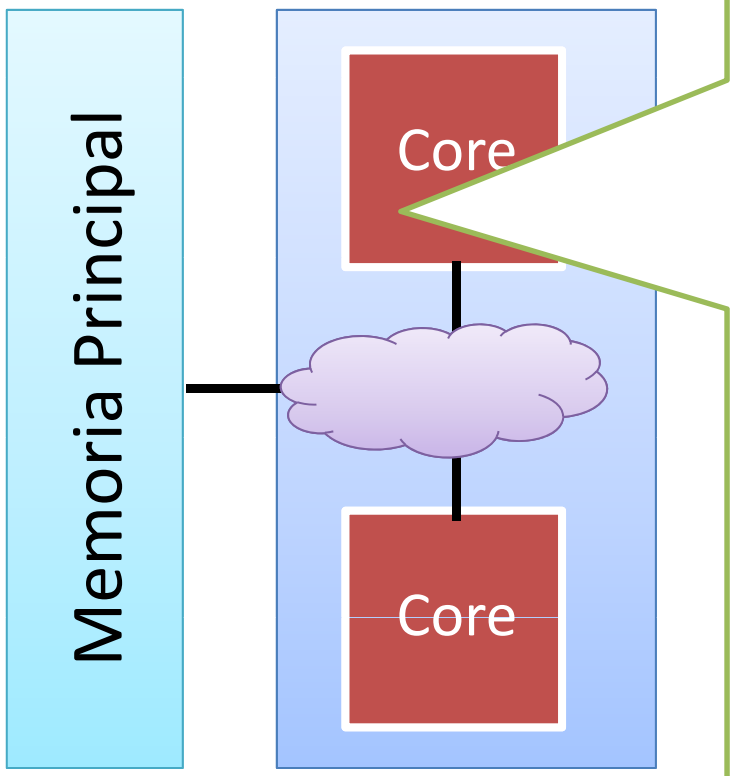
Acceso a Memoria compartida

- Sección Crítica
- Transaccional

Modelos de programación

- Paralelismo de datos
- Paralelismo funcional

Introducción – Taxonomía multicore

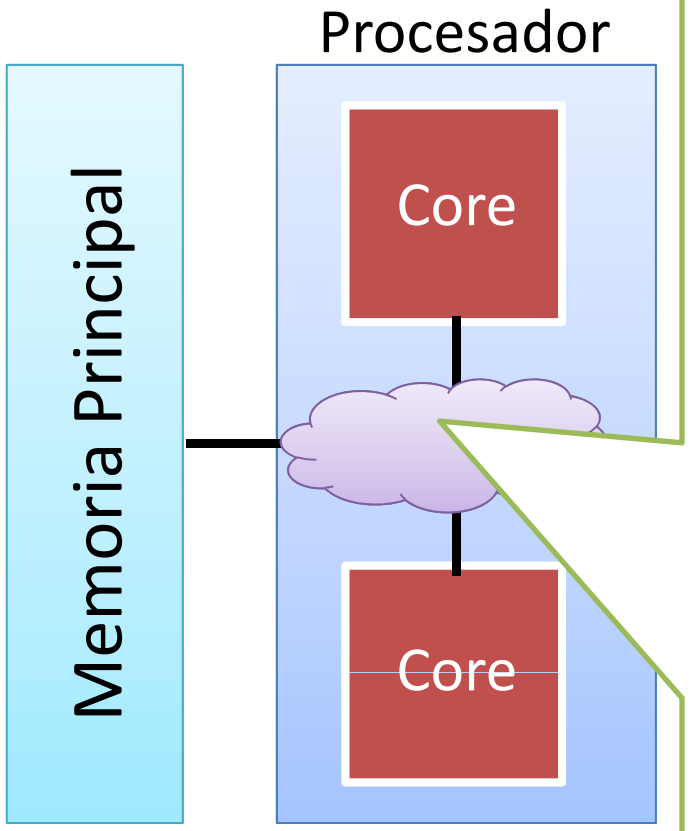


Arquitectura

Arquitectura

- Escalabilidad
  - ➢ - cores + complejos (multicore)
  - ➢ + cores - complejos (manycore)
- Tipos de cores (funcionalidad)
  - ➢ Homogéneos
    - ▪ Intel Core
  - ➢ Heterogéneos
    - ▪ PPE & SPE (PowerXCell)

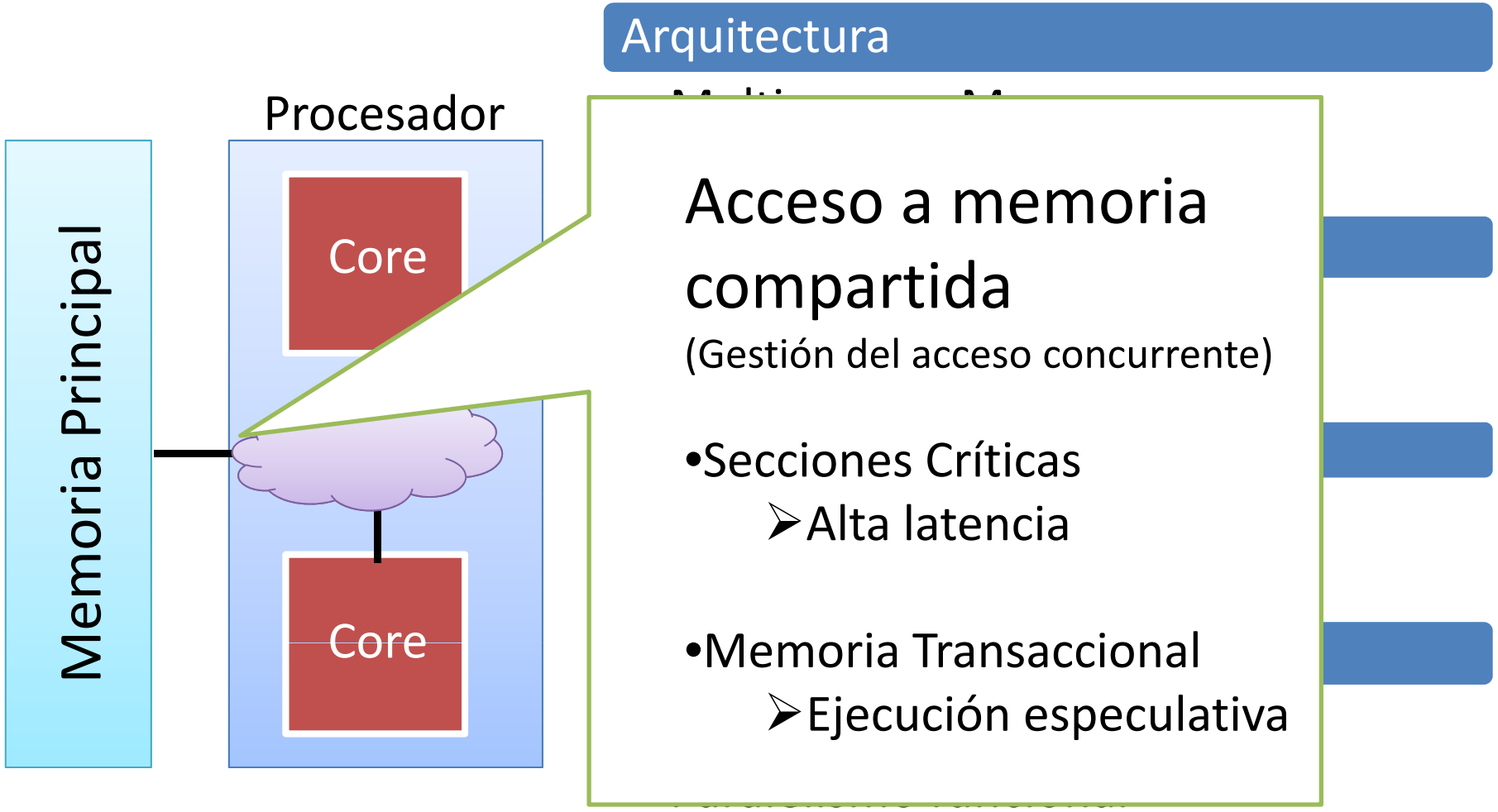
Introducción – Taxonomía multicore



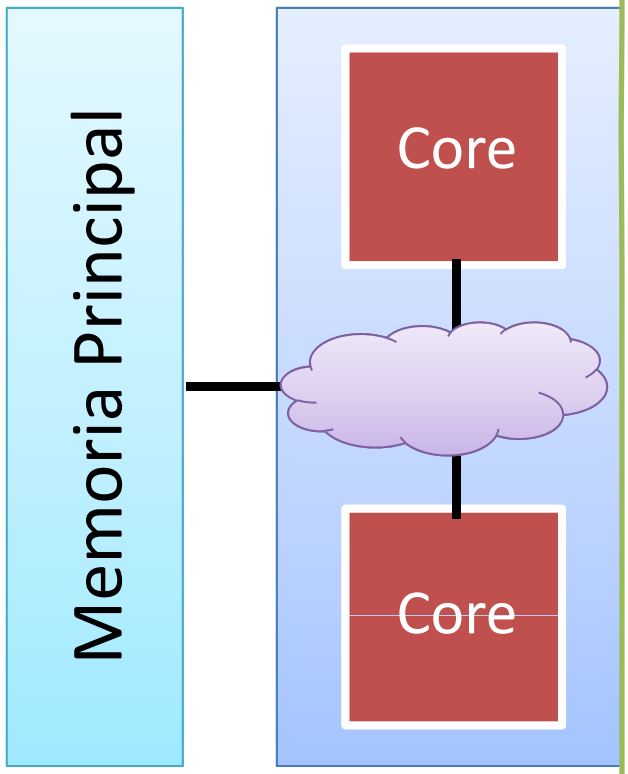
Interconexión

- Comunicación intracore
  - Memoria compartida
  - Paso de mensajes
    - rMPI
- Topologías (Network On Chip)
  - Bus, anillo
  - Mallas, Crossbar, switched networks
  - Jerárquica por niveles (mixta)

Introducción – Taxonomía multicore



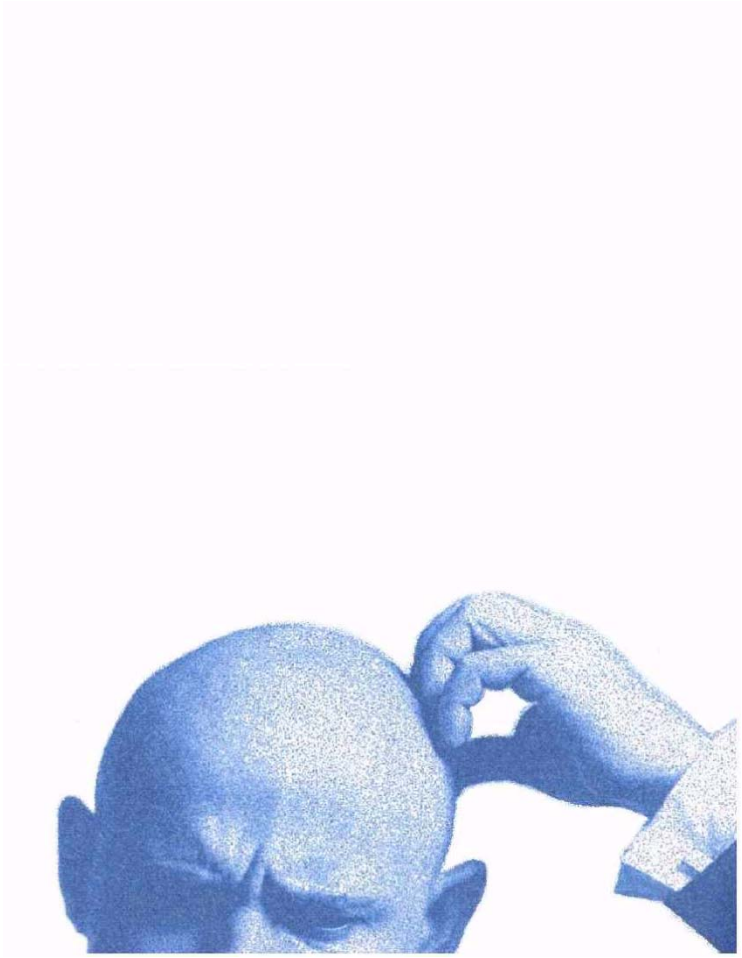
Introducción – Taxonomía multicore

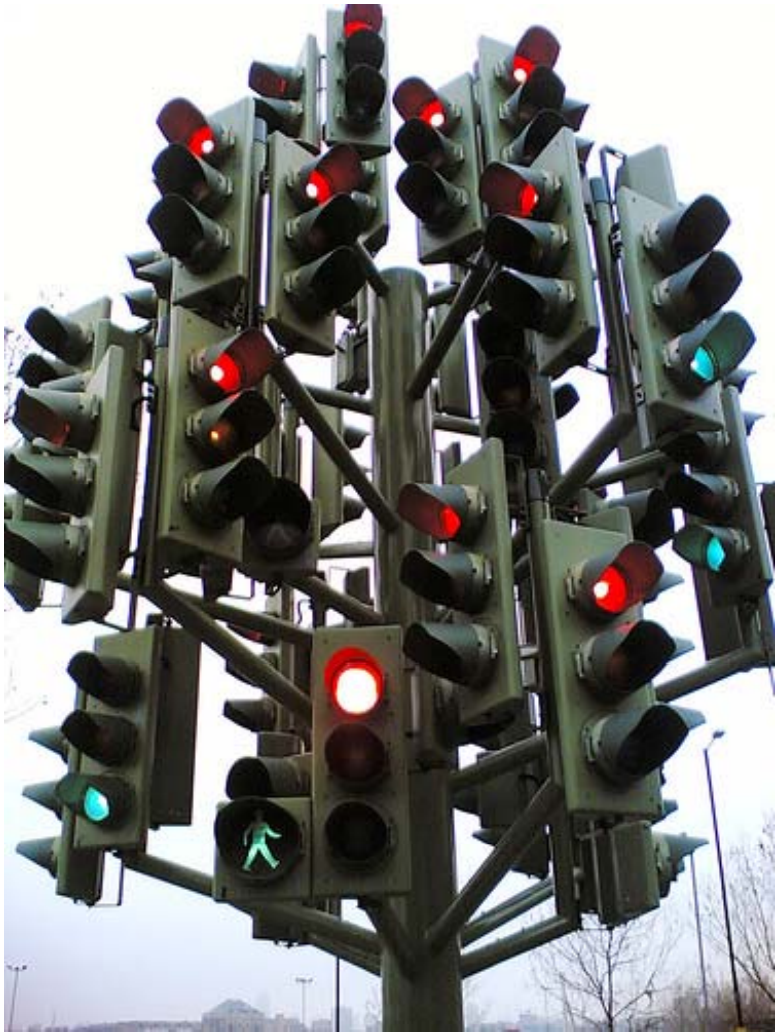


Paradigmas de programación

- Paralelismo de datos
  - Unidad: thread
  - Gestión: nivel de S.O. o usuario
  - Ejemplos: Pthreads, OpenMP 2.5
- Paralelismo funcional
  - Unidad:tarea
  - Gestión: DAGs
  - Ejemplos: Cilk, OpenMP v3.0, *Ss

CAOS





Cesar.Allande@ caos .uab.es

UAB

Definir un marco de trabajo acotado

Paradigma de programación paralela basado en OpenMP v2.5

Paralelismo de datos

Herramientas de acceso concurrente a memoria

Planificación de threads a nivel de Sistema Operativo

Procesadores homogéneos

Dual(2 cores)

memoria compartida jerárquica

L2 unificada (2MB)

L1 datos e instrucciones (32KB)

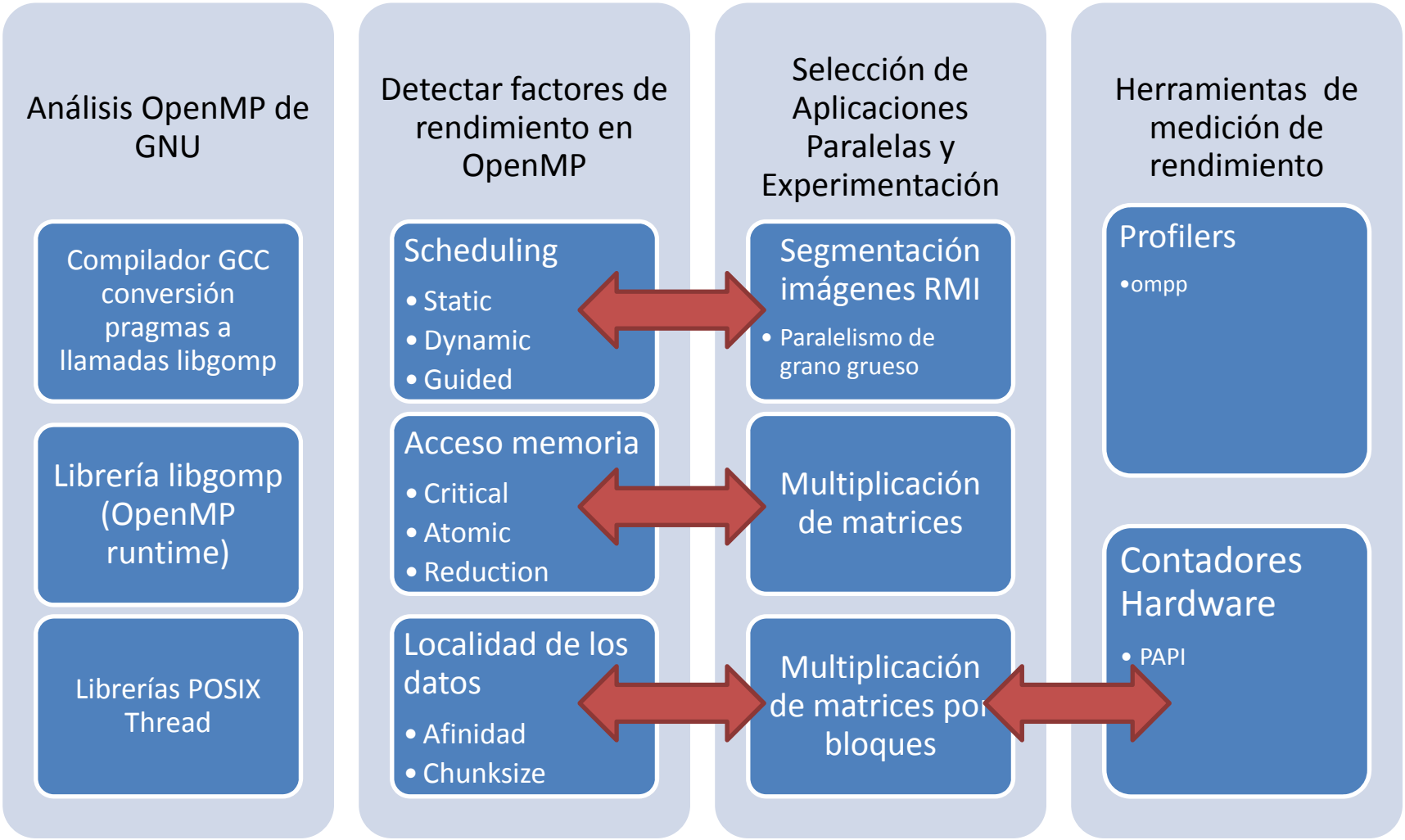
Asociatividad por bloques (8 vías)

Objetivos

Objetivos

- Identificar factores de rendimiento
- Estudio de los parámetros sintonizables de una API multicore
- Estudio del patrón de comportamientos de una aplicación y evaluación de los factores de rendimiento

Desarrollo del Estudio



Análisis I - factor de rendimento
planificación de la carga de trabajo

Scheduling
- Static
- Dynamic
- Guided

Gestión de planificación de la carga de trabajo

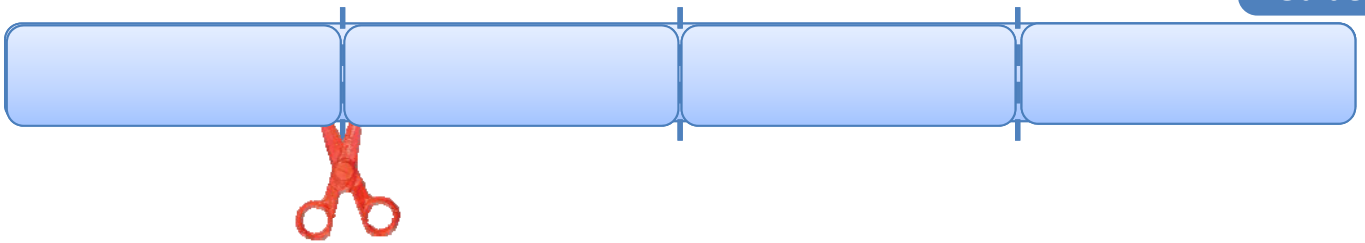
- La sintonización de las políticas de balanceo de carga puede ser determinante para el rendimiento de la aplicación

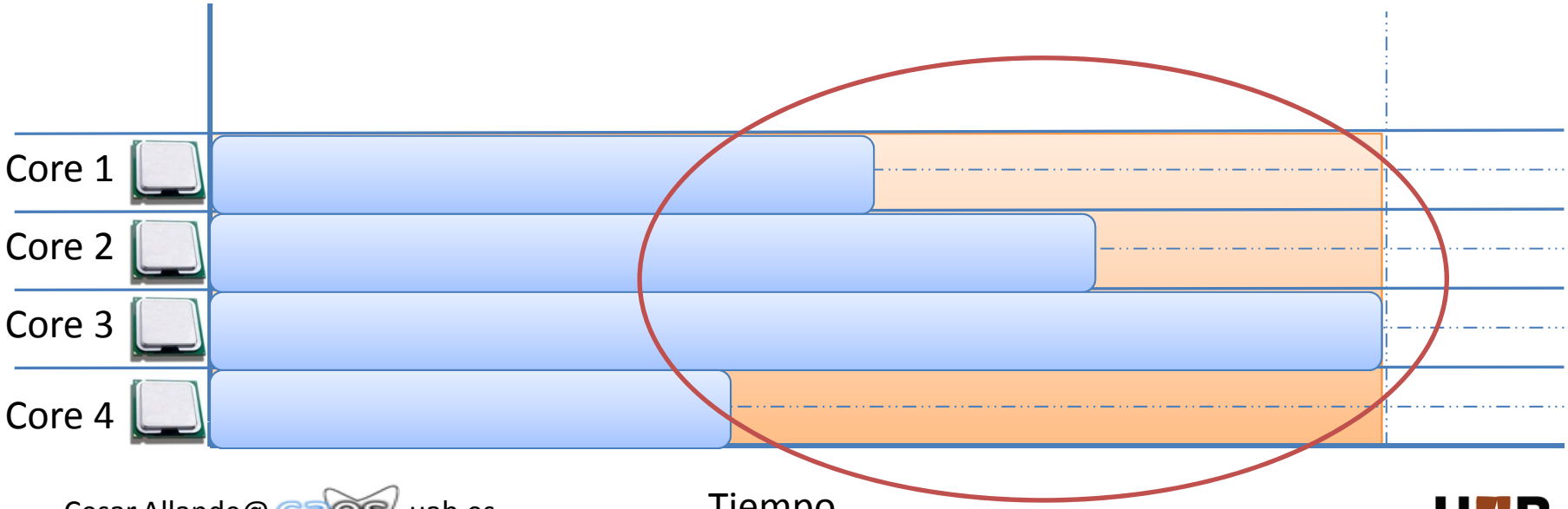
¿En qué medida?

Análisis I – Scheduling I (static)

Scheduling
- Static
- Dynamic
- Guided

WorkLoad = Iteraciones / Num Threads

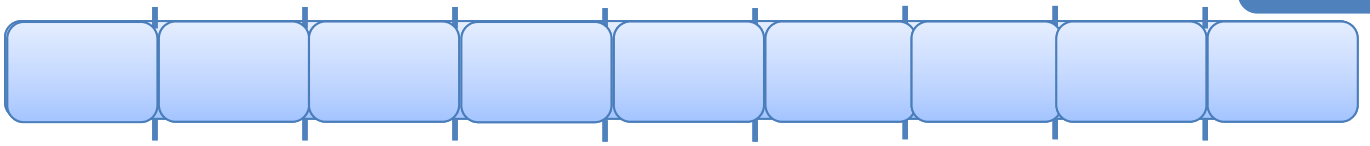


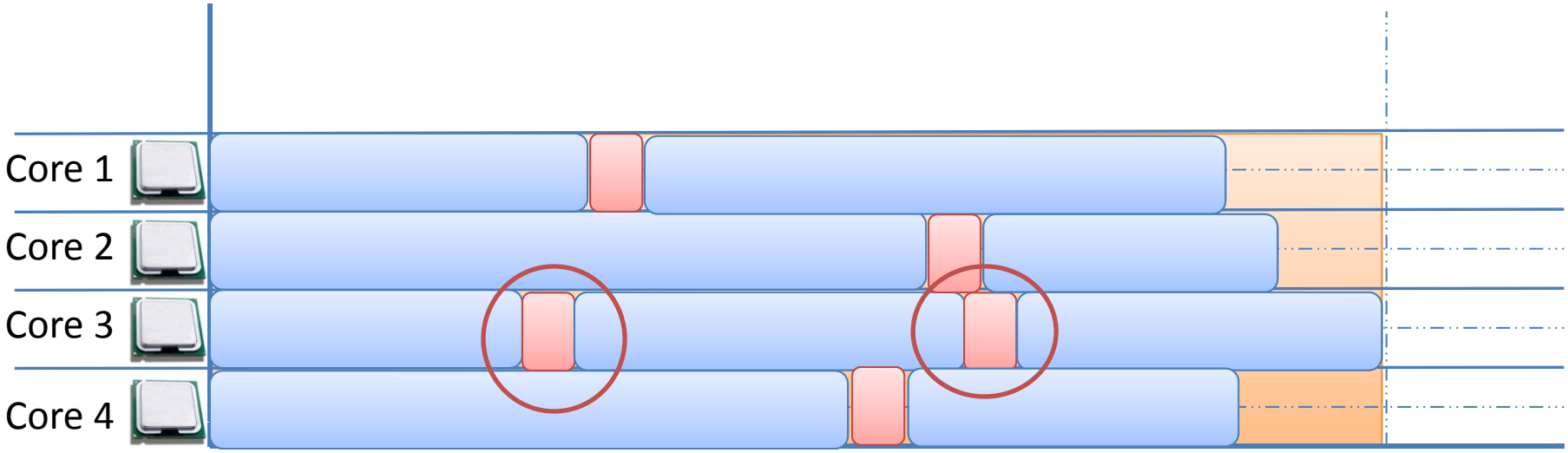


Análisis I – Scheduling II (dynamic)

Scheduling
- Static
- Dynamic
- Guided

WorkLoad = Chunksize (1)

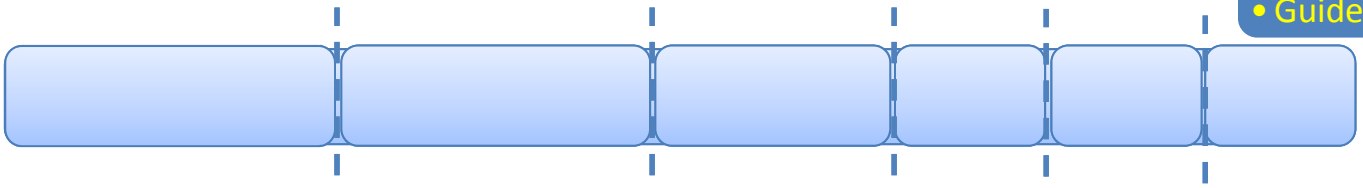




Análisis I – Scheduling III (guided)

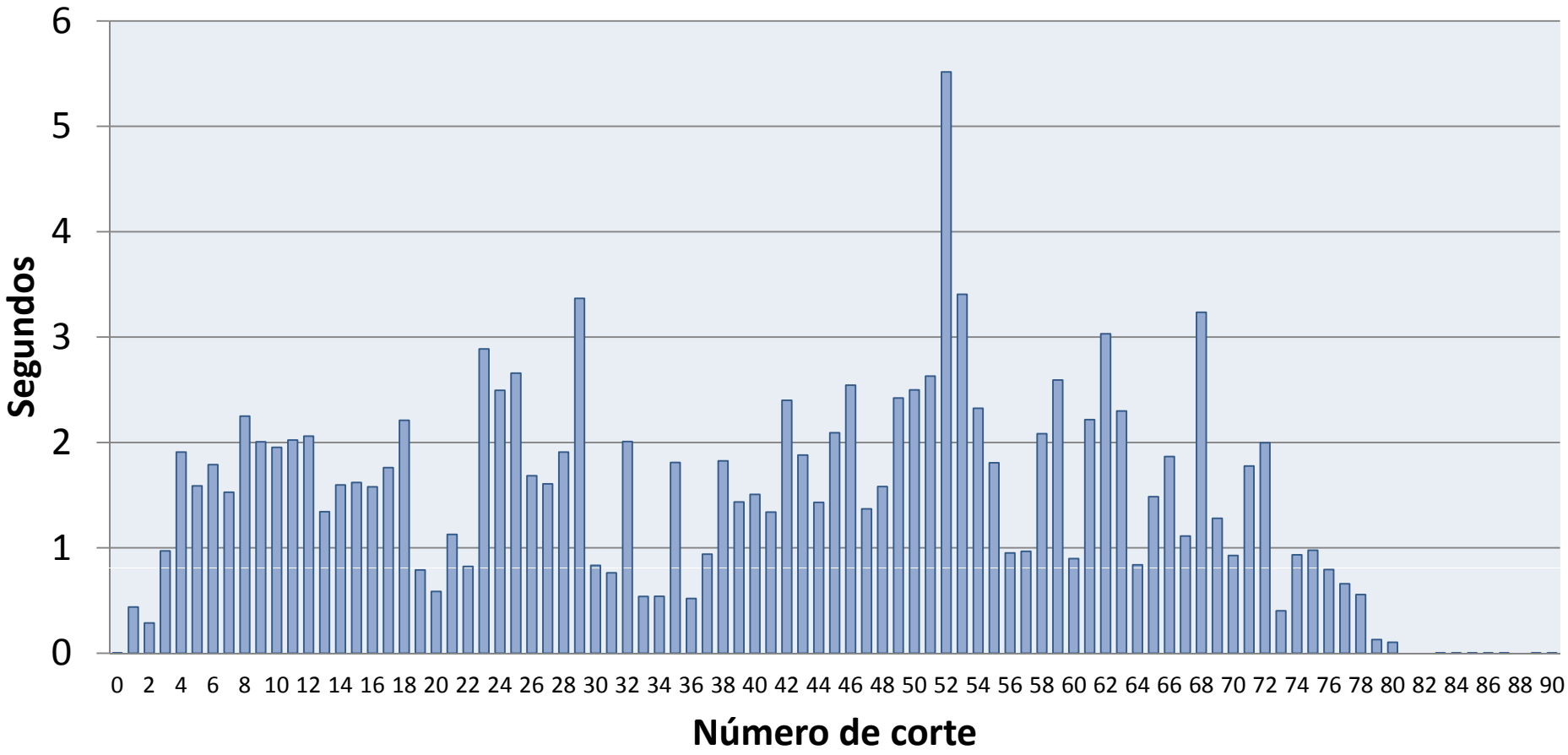
Scheduling
- Static
- Dynamic
- Guided

WorkLoad = Remaining iterations / Num Threads



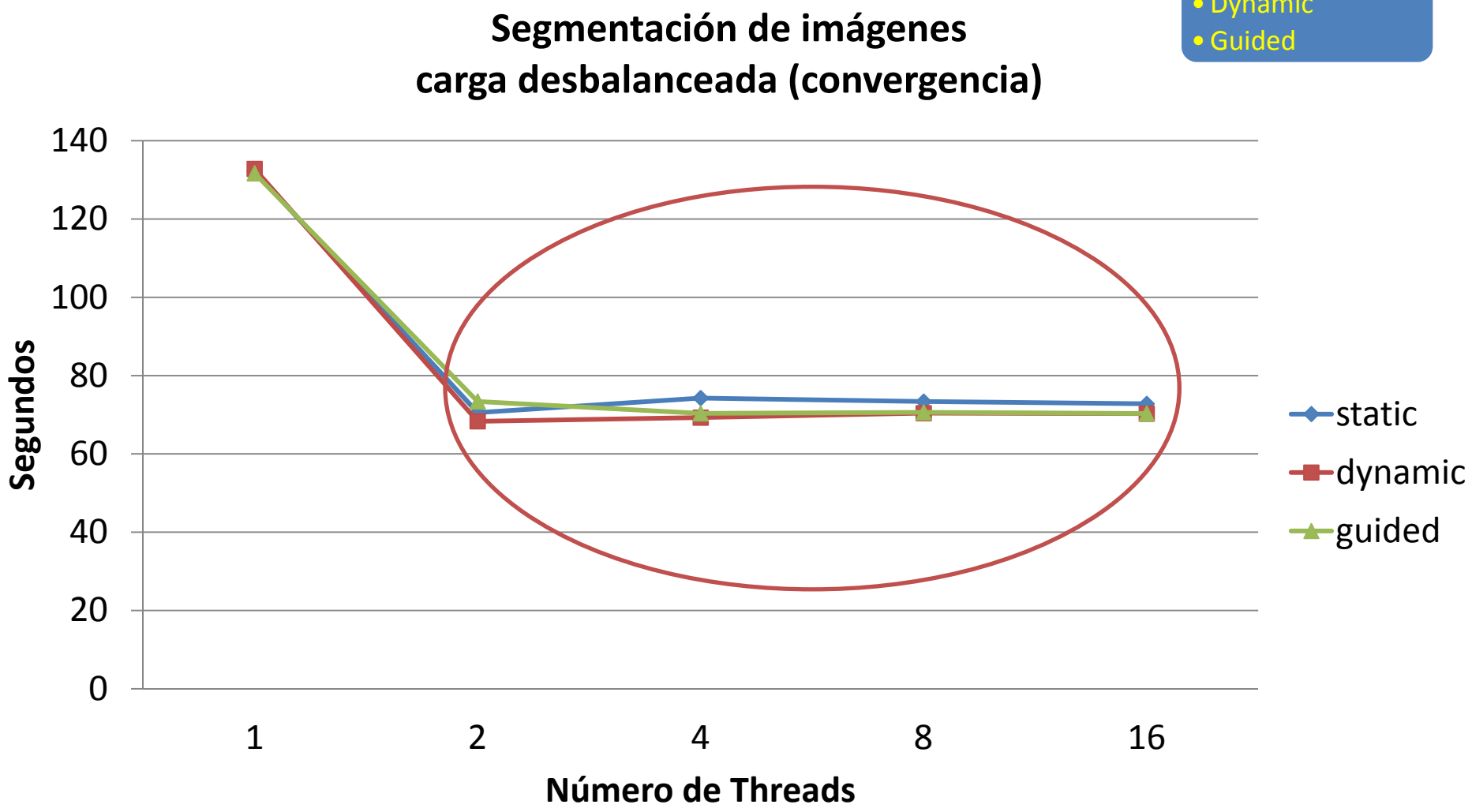
Experimentación I
scheduling - segmentación RMI (I)

Scheduling
- Static
- Dynamic
- Guided



Experimentación I
scheduling - segmentación RMI (II)

Scheduling
- Static
- Dynamic
- Guided

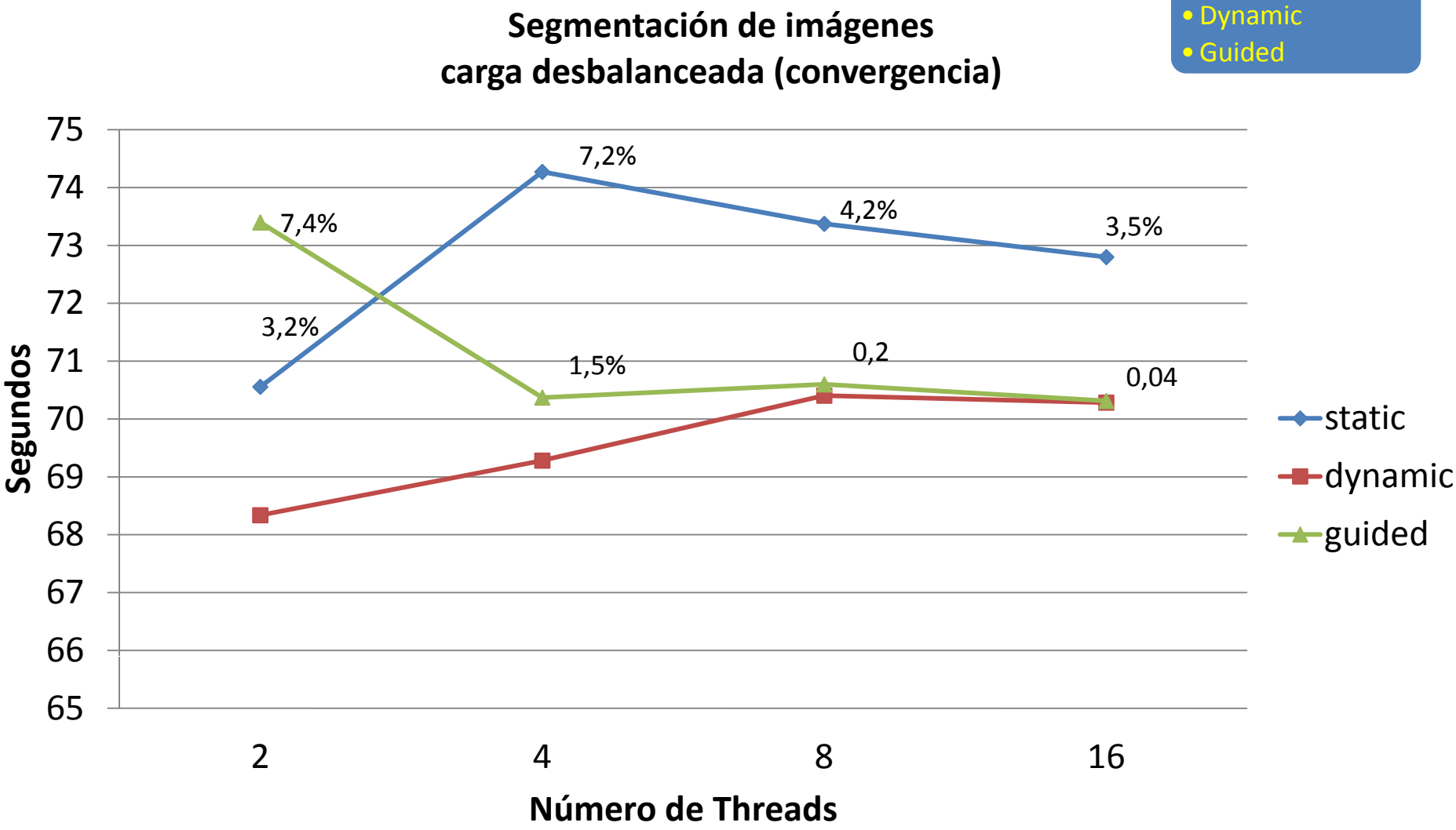


Cesar.Allande@ caos .uab.es

UAB

Experimentación I
scheduling - segmentación RMI (II)

Scheduling
- Static
- Dynamic
- Guided

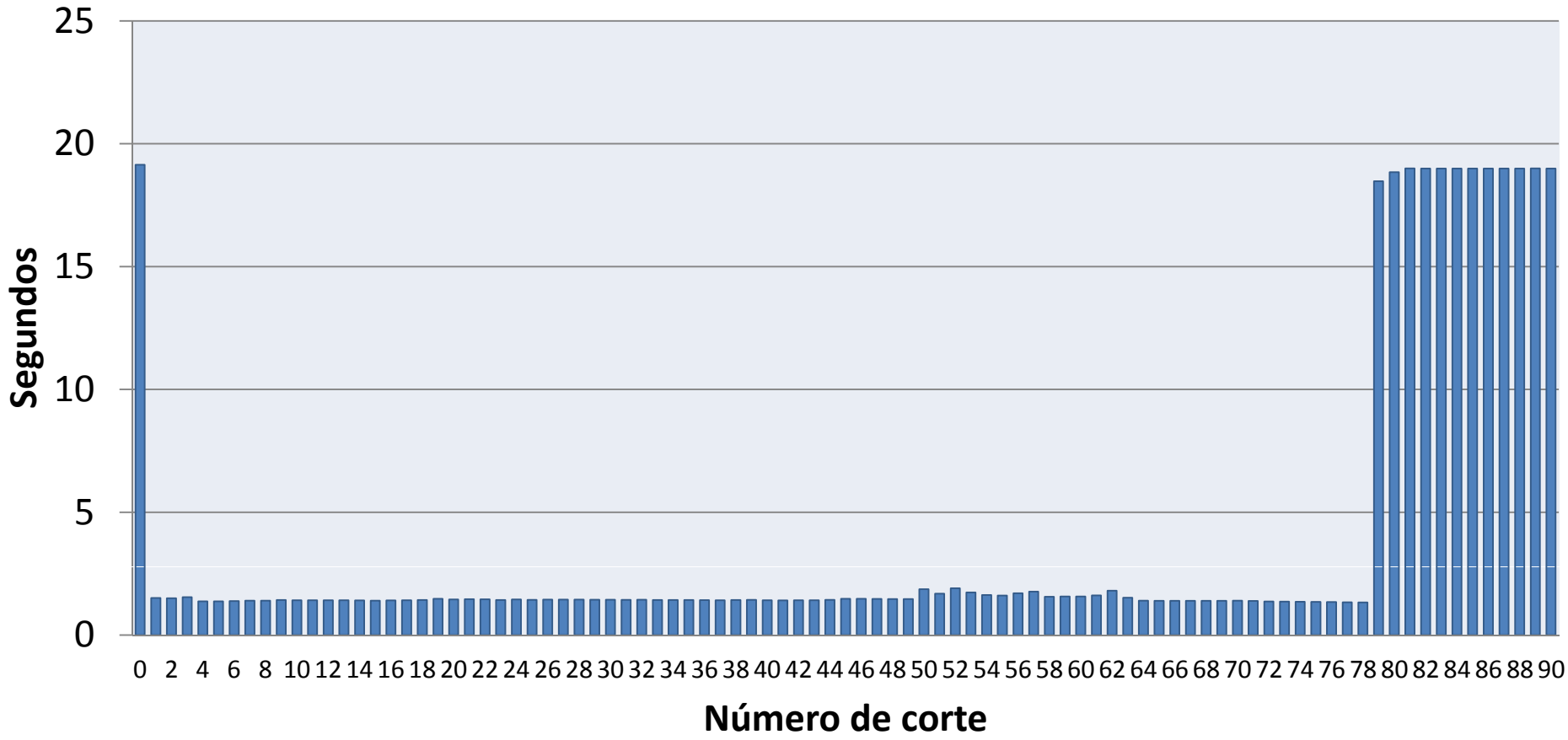


Cesar.Allande@ caos .uab.es

UAB

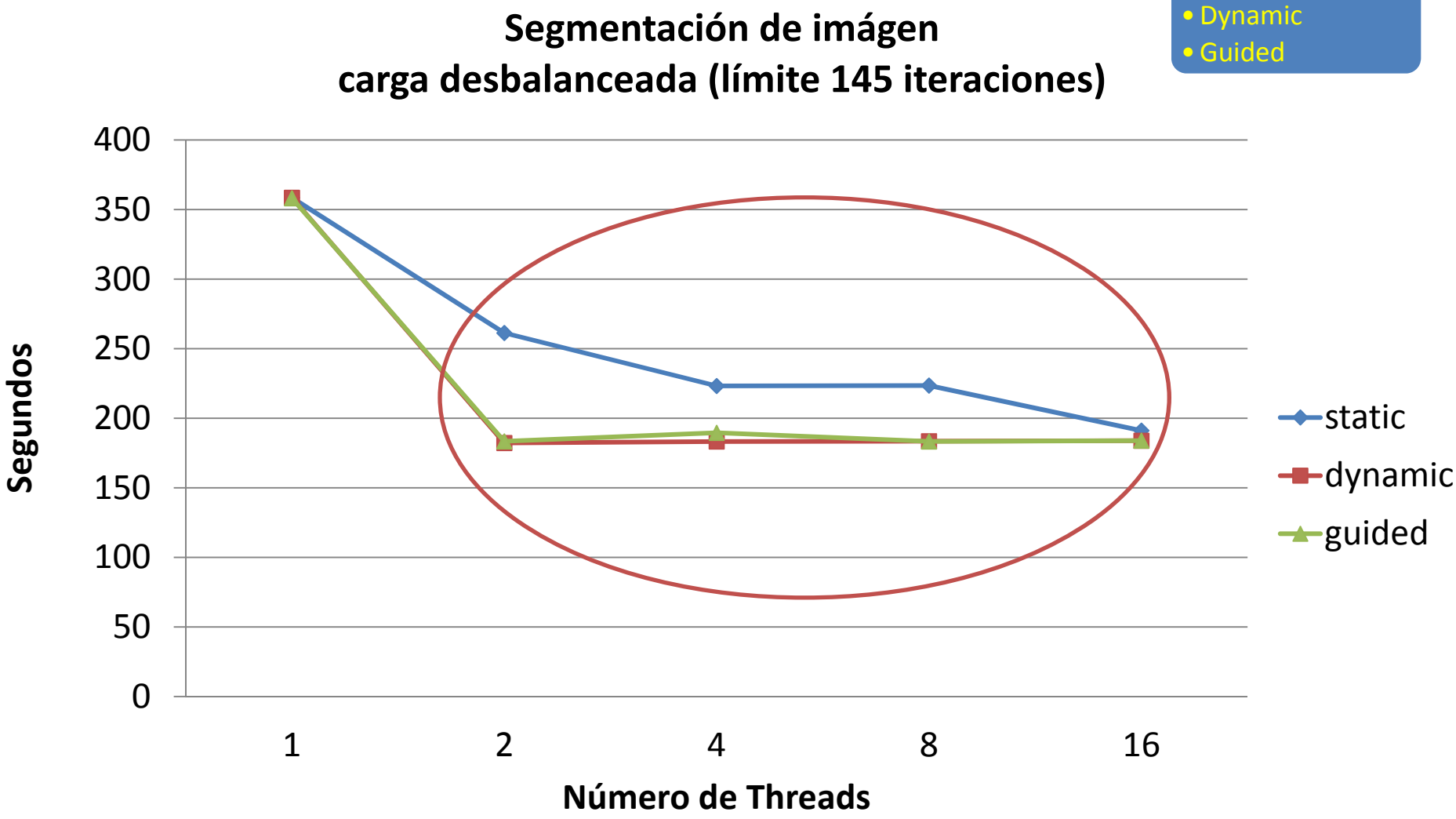
Experimentación I
scheduling - segmentación RMI (III)

Scheduling
- Static
- Dynamic
- Guided



Experimentación I
scheduling - segmentación RMI (IV)

Scheduling
- Static
- Dynamic
- Guided

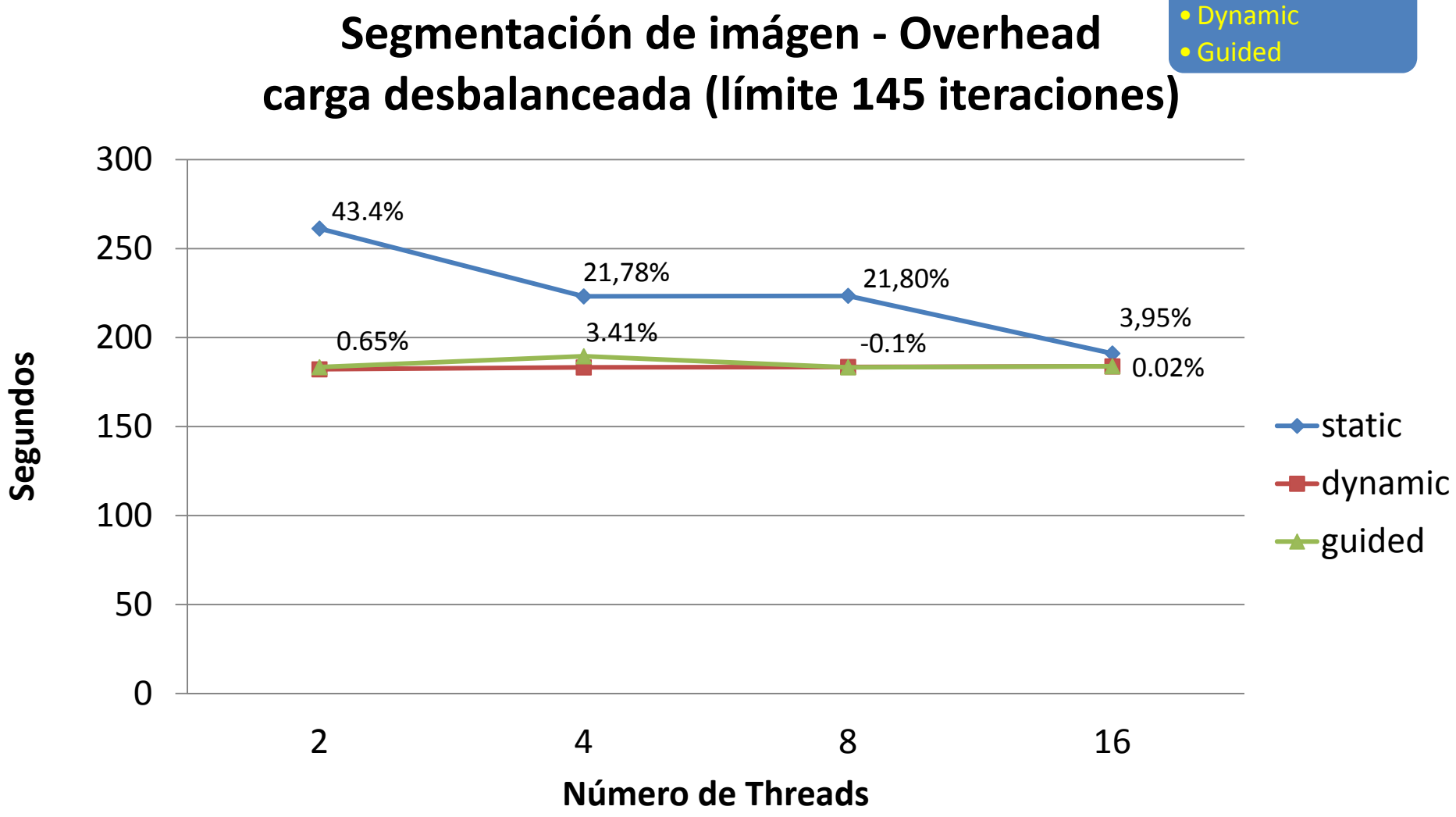


Cesar.Allande@ caos .uab.es

UAB

Experimentación I
scheduling - segmentación RMI (IV)

Scheduling
- Static
- Dynamic
- Guided



Cesar.Allande@ caos .uab.es

UAB

Análisis II - factor de rendimiento
acceso concurrente a memoria

Acceso memoria
- Critical
- Atomic
- Reduction

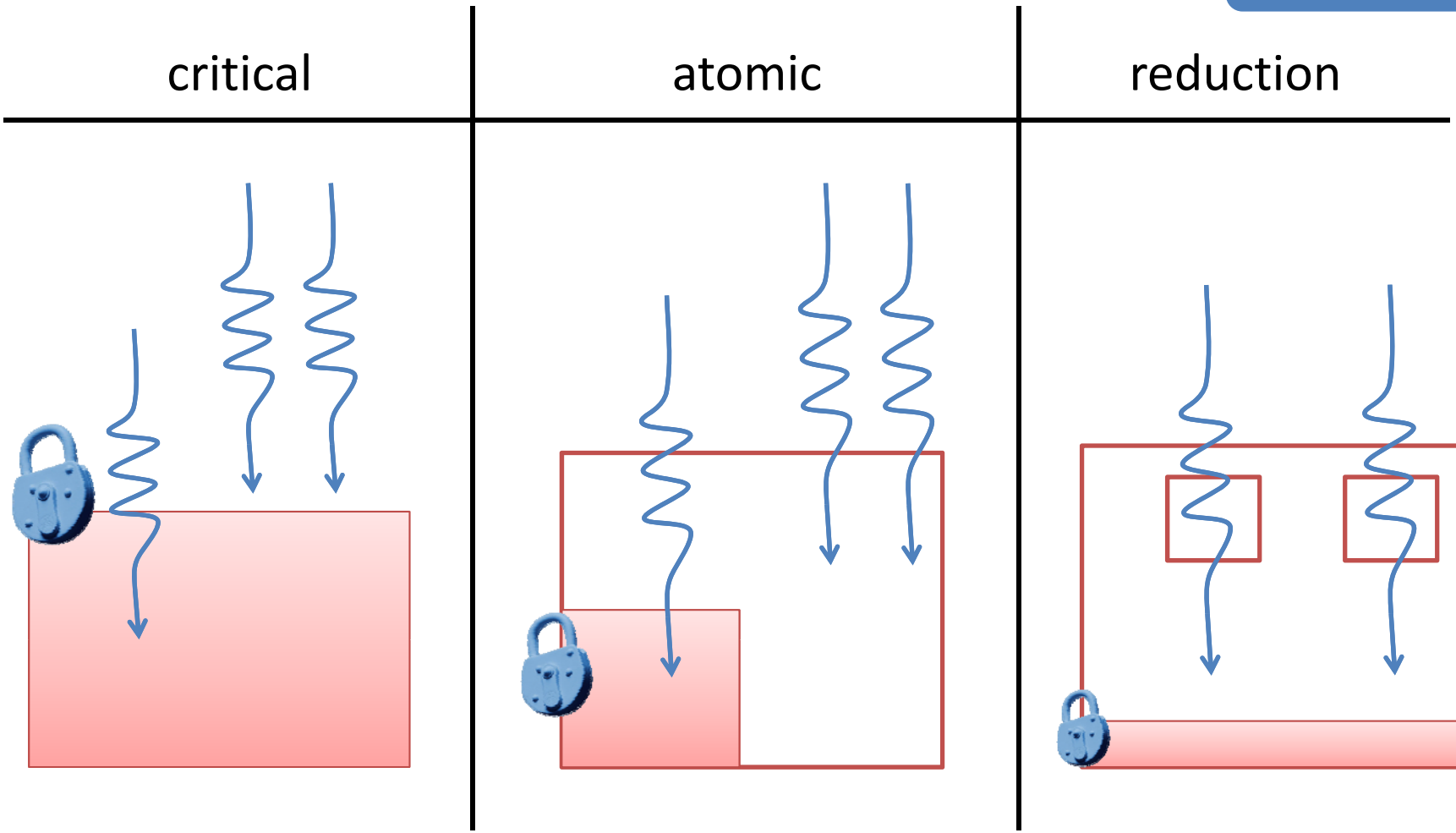
Sincronización de hilos

- El mecanismo de sincronización de acceso a memoria compartida genera una sobrecarga que afecta al rendimiento

¿En qué medida?

Análisis II
sincronización de acceso a memoria

Acceso memoria
- Critical
- Atomic
- Reduction

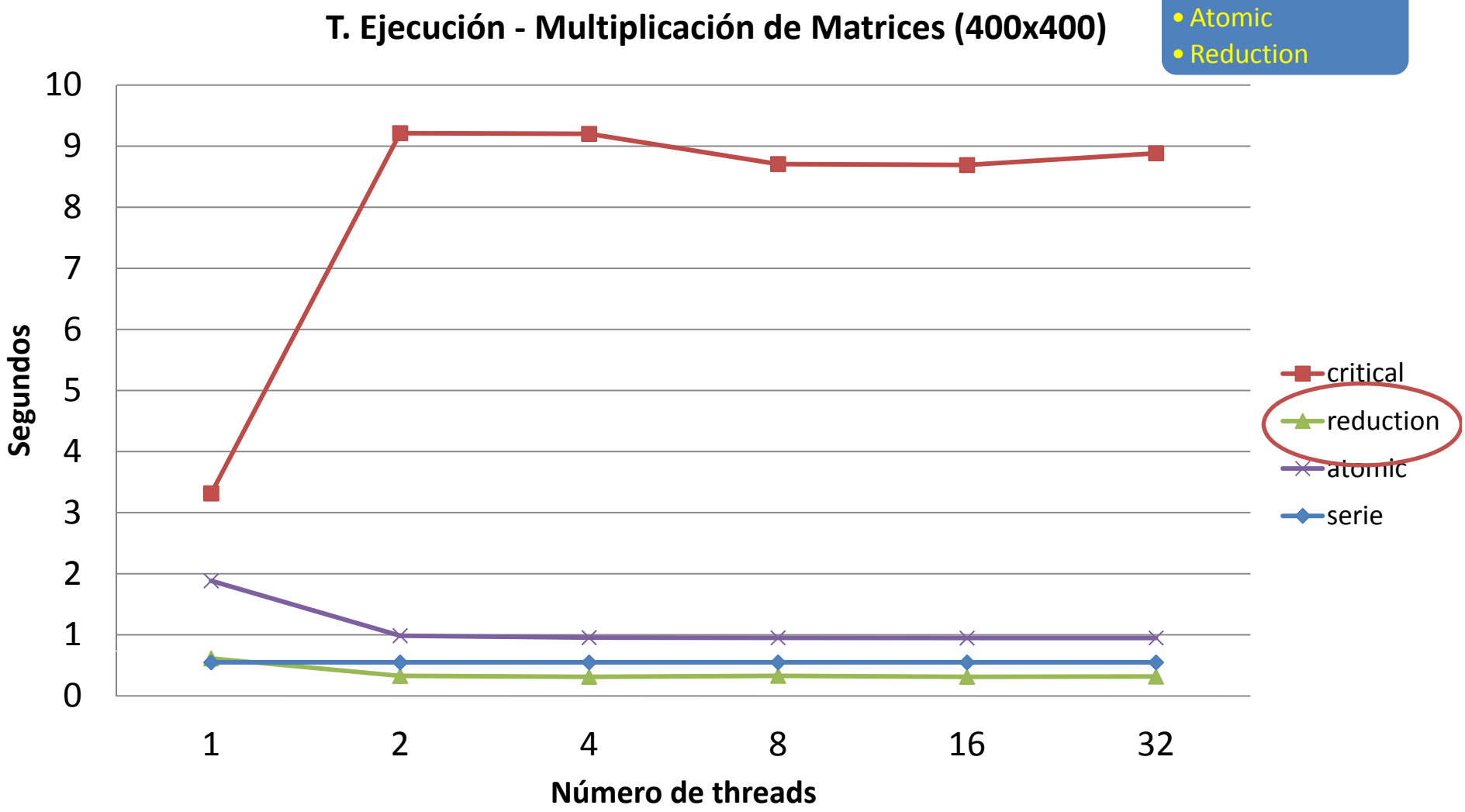


Cesar.Allande@ caos .uab.es

UAB

Experimentación II
sincronización – Mat. Mul. (400x400)

Acceso memoria
- Critical
- Atomic
- Reduction

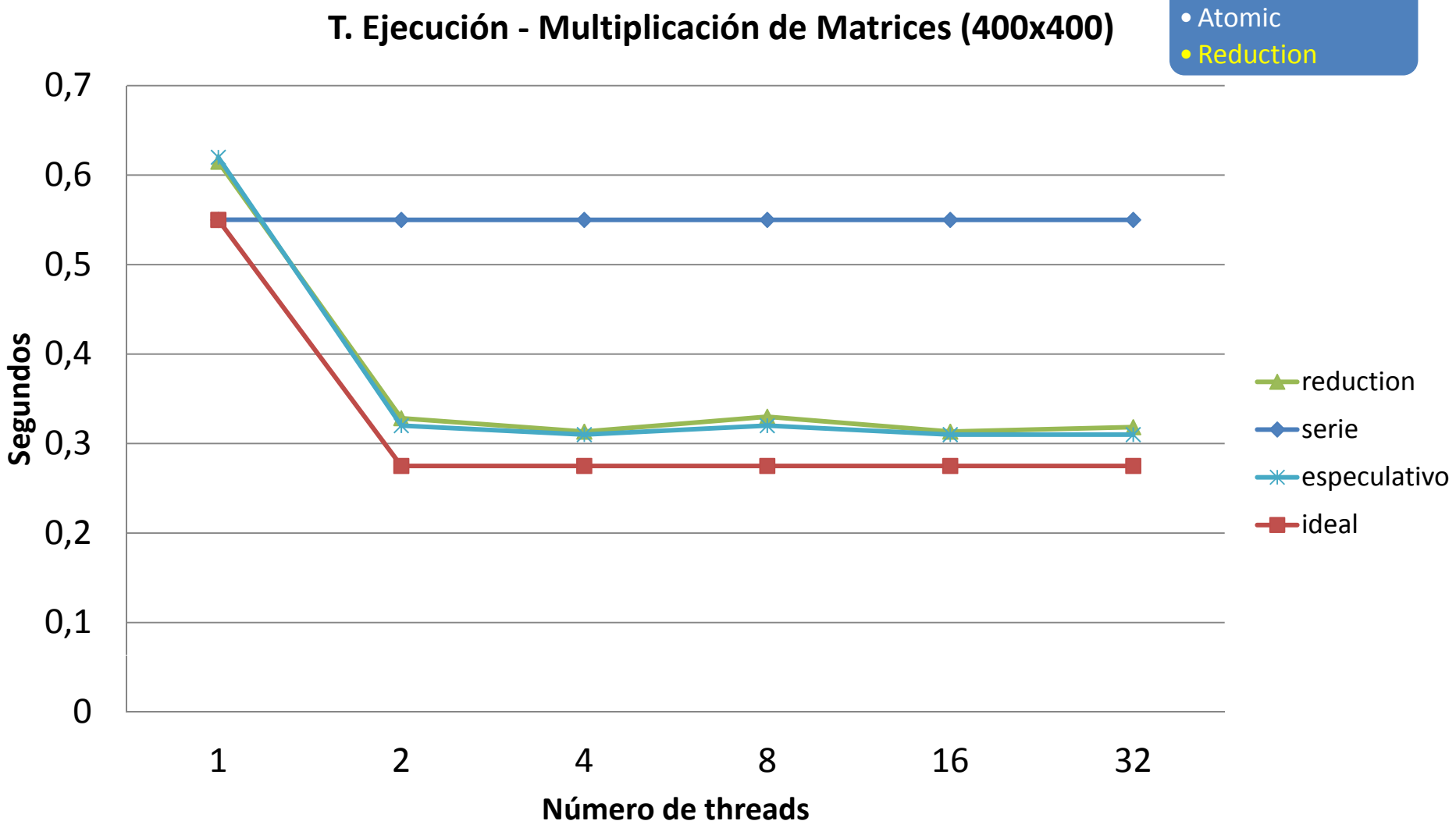


Cesar.Allande@ caos .uab.es

UAB

Experimentación II
sincronización – Mat. Mul. (400x400)

Acceso memoria
- Critical
- Atomic
- Reduction



Cesar.Allande@ caos .uab.es

UAB

Análisis III - factor de rendimiento
localidad espacial y temporal

Localidad de los datos
- Afinidad
- Chunksize

Localidad de los datos

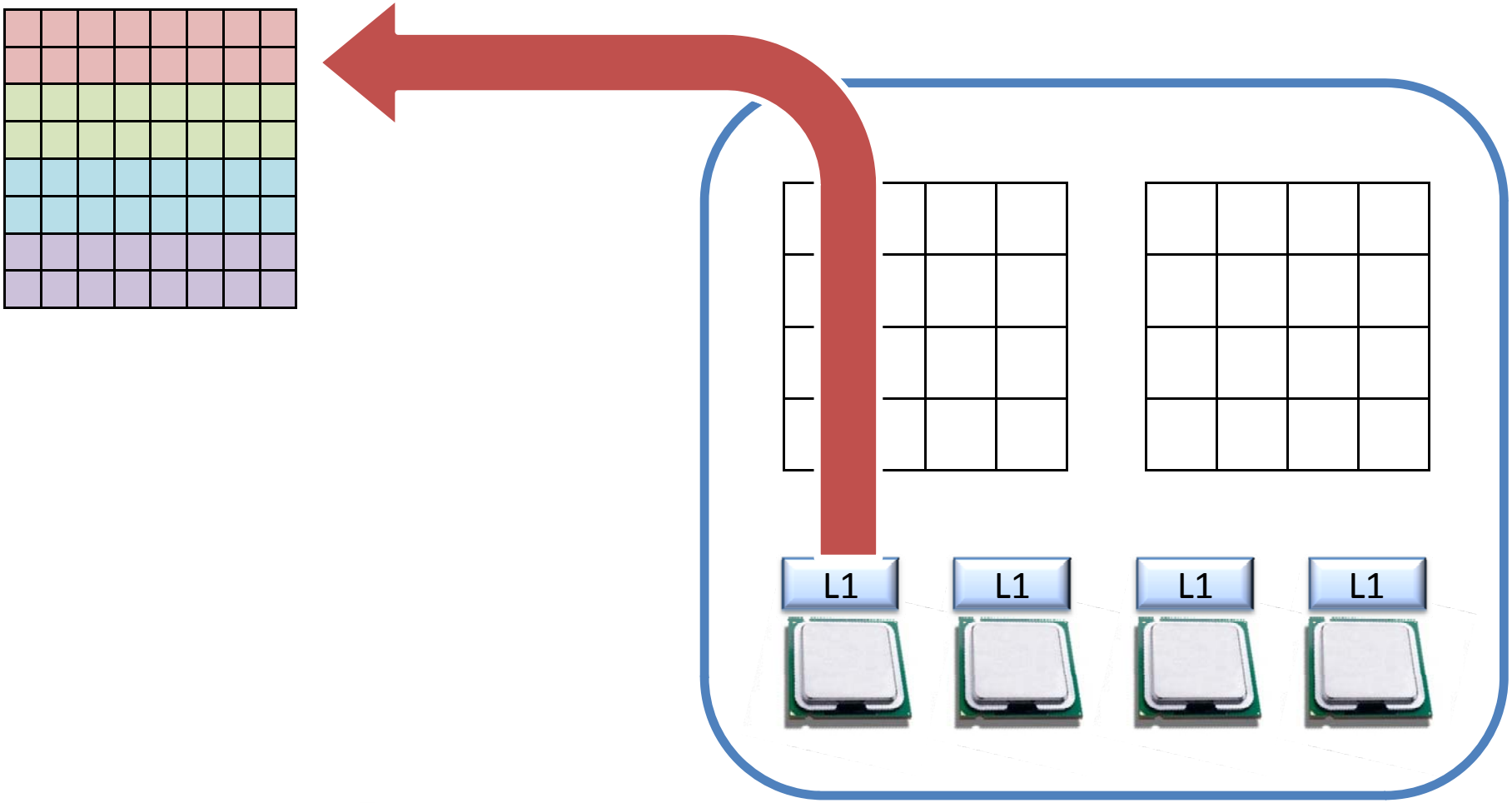
- Las estructuras jerárquicas de memoria con diferentes niveles de latencia y etapas de acceso a los datos se benefician de la localidad espacial y temporal

¿En qué medida?

Análisis III - Localidad de los datos

Localidad de los datos

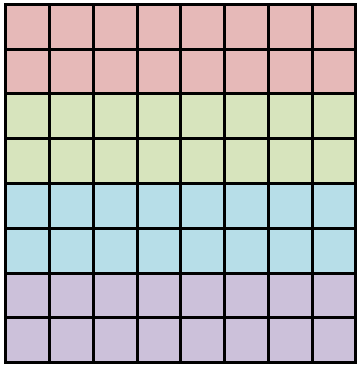
- Afinidad
- Chunksize

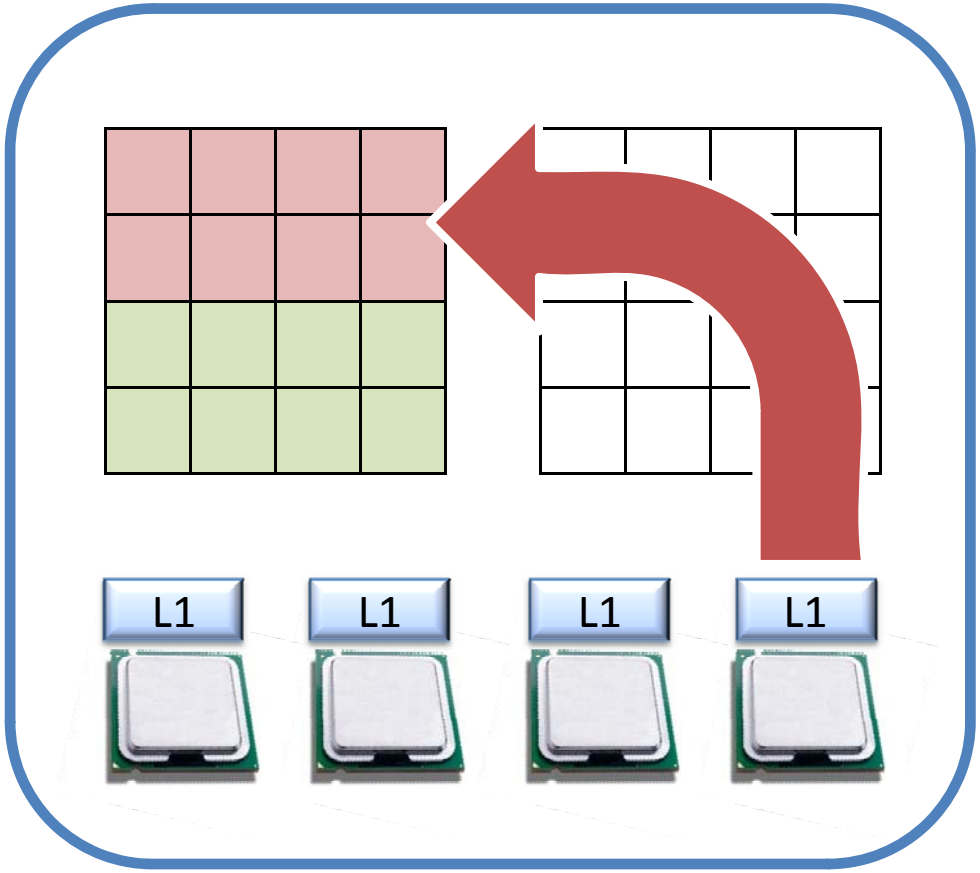


Análisis III - Localidad de los datos

Localidad de los datos

- Afinidad
- Chunksize

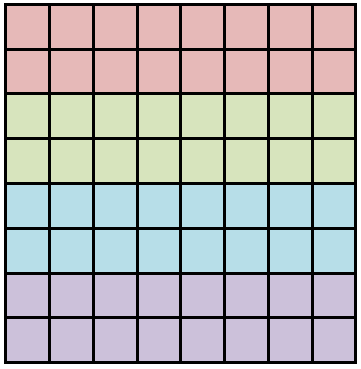


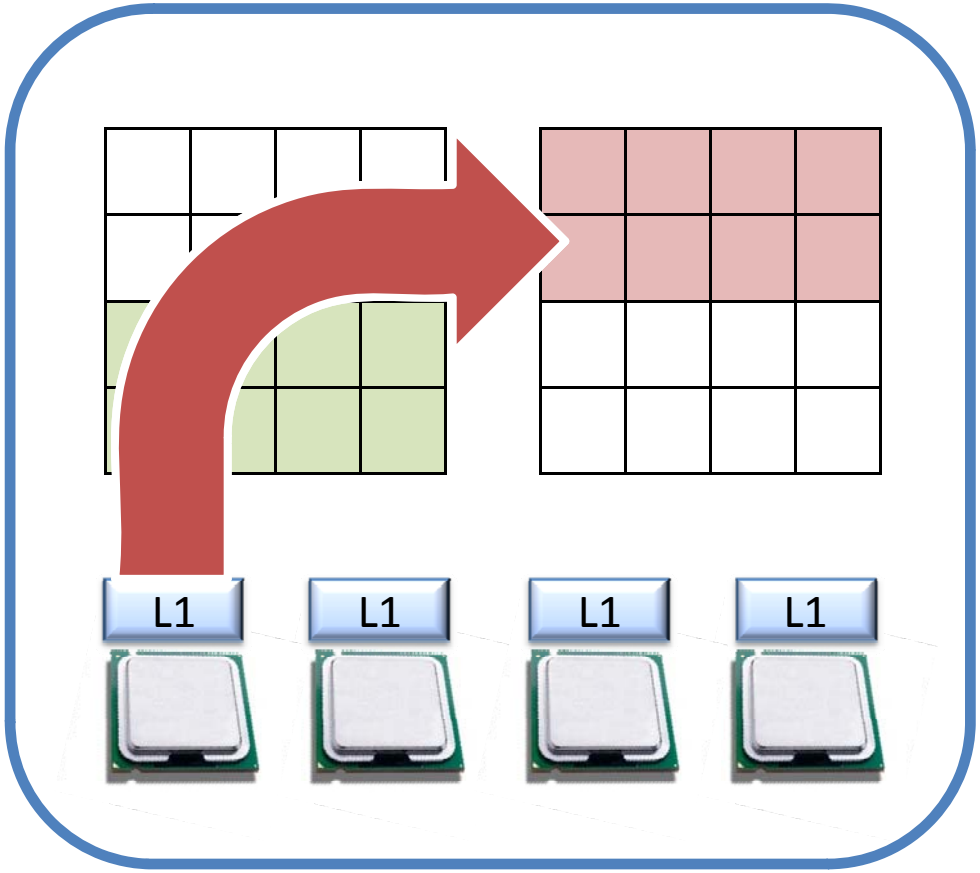


Análisis III - Localidad de los datos

Localidad de los datos

- Afinidad
- Chunksize





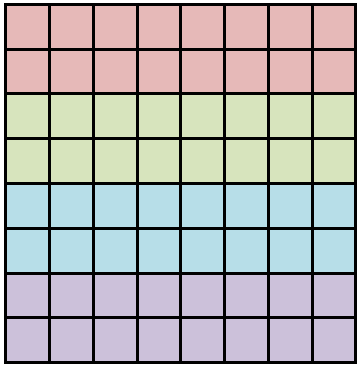
Cesar.Allande@ caos .uab.es

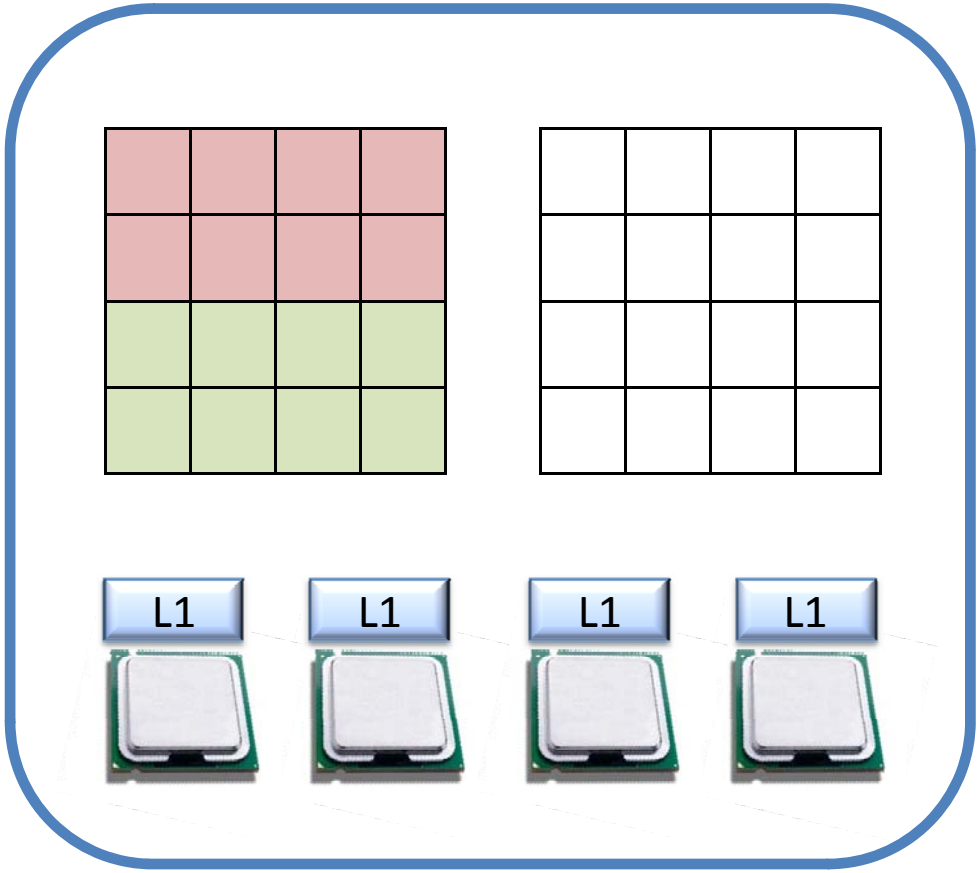
UAB

Análisis III - Localidad de los datos

Localidad de los datos

- Afinidad
- Chunksize





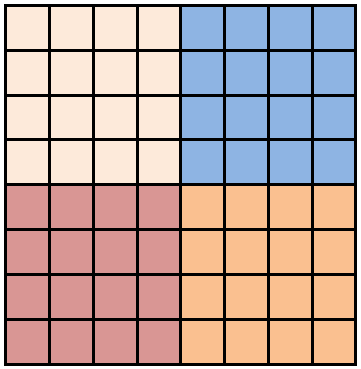
Cesar.Allande@ caos .uab.es

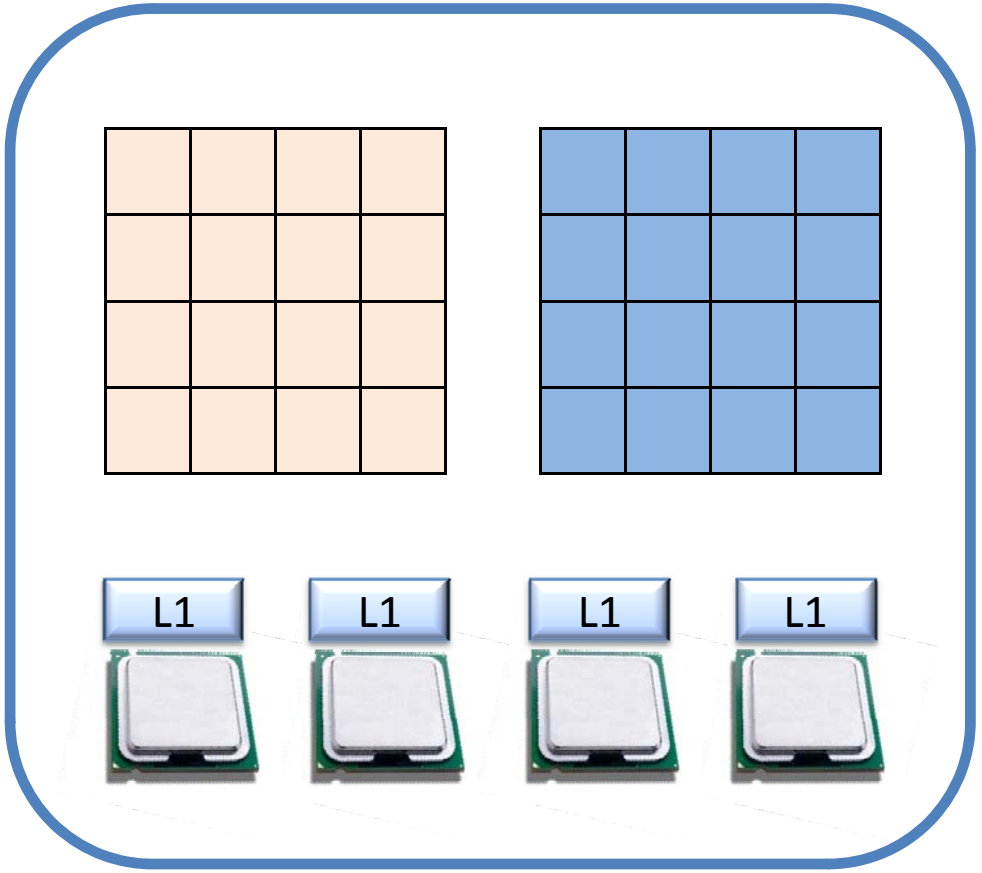
UAB

Análisis III - Localidad de los datos

Localidad de los datos

- Afinidad
- Chunksize



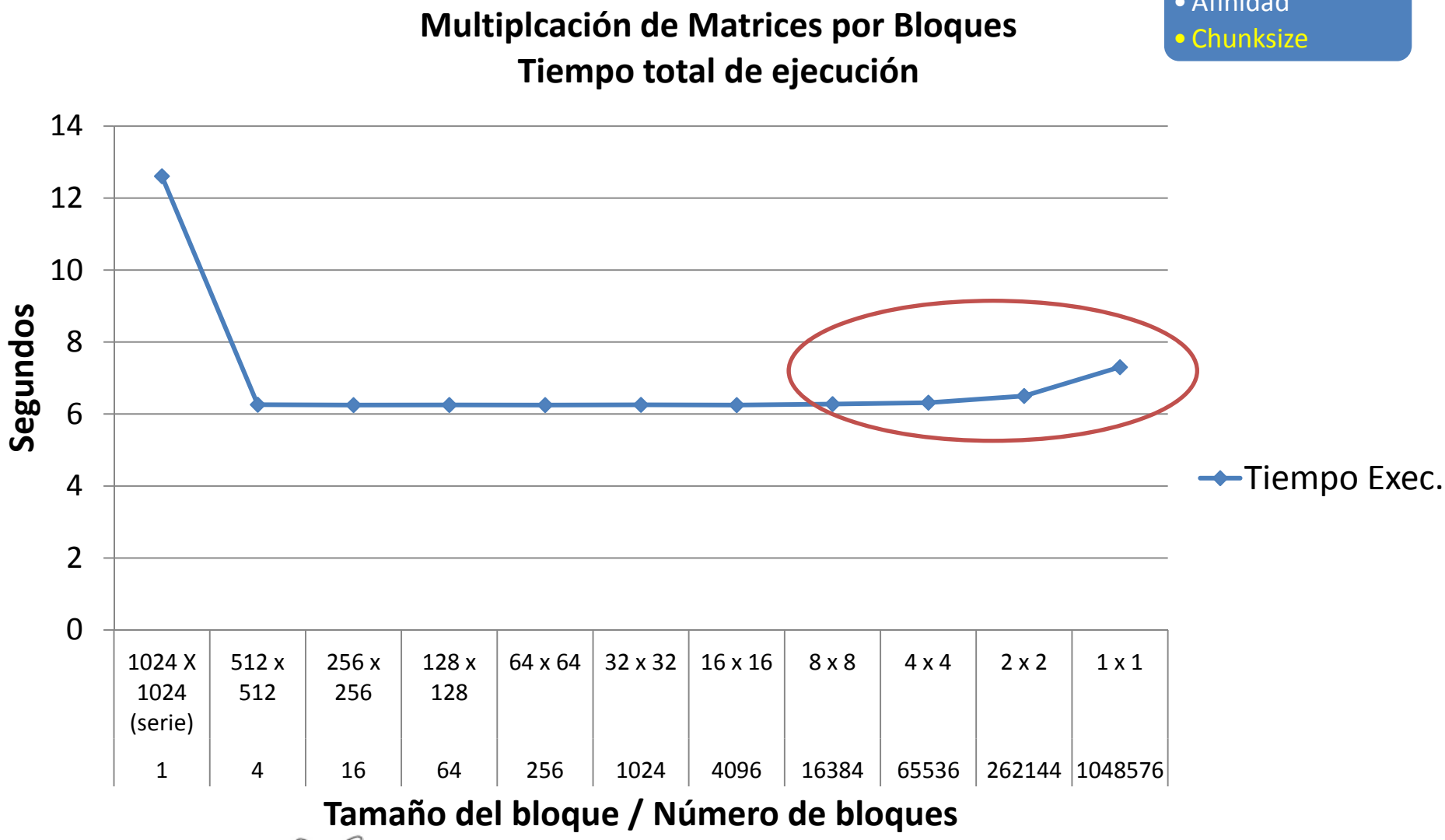


Cesar.Allande@ caos .uab.es

UAB

Experimentación III
localidad datos – Mat.Mul x Bloques

Localidad de los datos
- Afinidad
- Chunksize

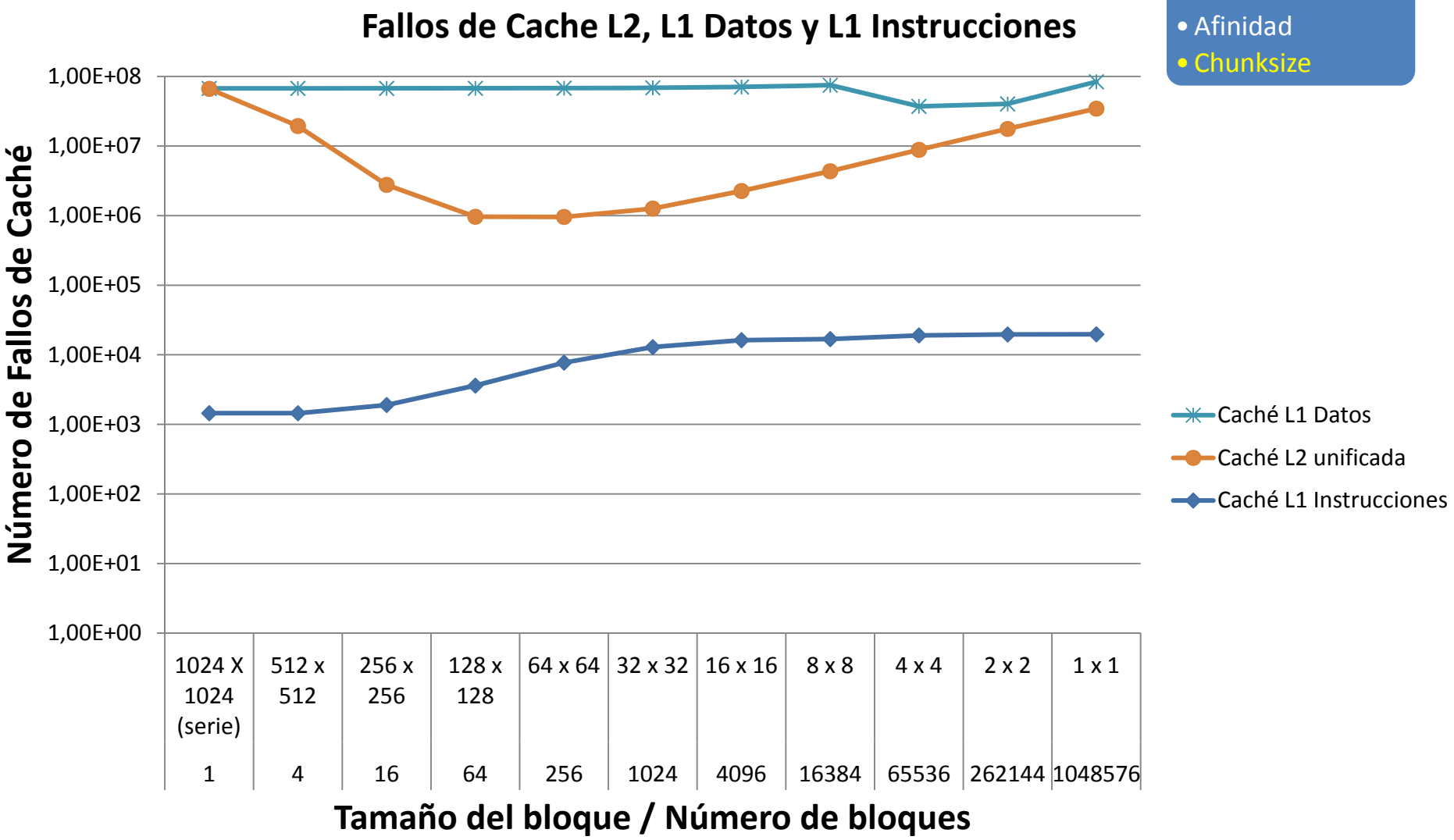


Cesar.Allande@ caos .uab.es

UAB

Experimentación III
localidad datos – Mat.Mul x Bloques

Localidad de los datos
- Afinidad
- Chunksize



Cesar.Allande@ caos .uab.es

UAB

Experimentación III
localidad datos – Mat.Mul x Bloques

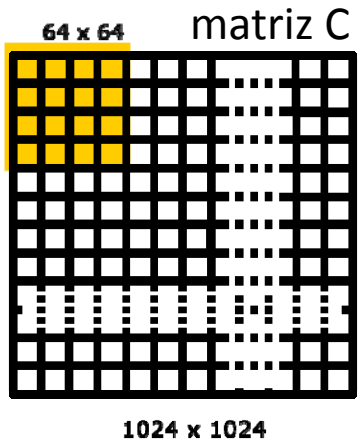
Localidad de los datos
- Afinidad
- Chunksize

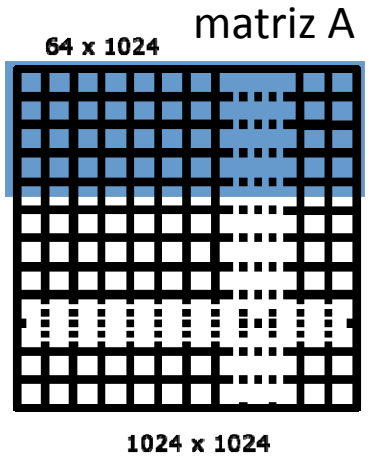
- Modificación del algoritmo para ajustar el acceso a datos para 2 niveles de caché
  - Bloques de 64x64 elem (256 KB), ajuste L2
  - Sub-bloques de 2x2 elem.(4KB), ajuste L1.

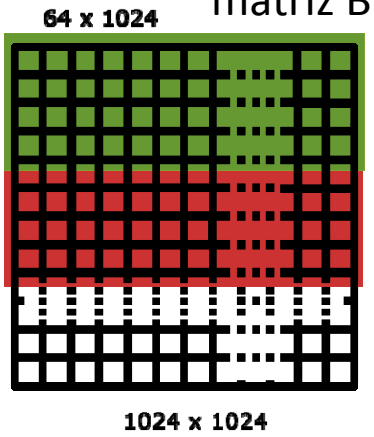
Experimentación III

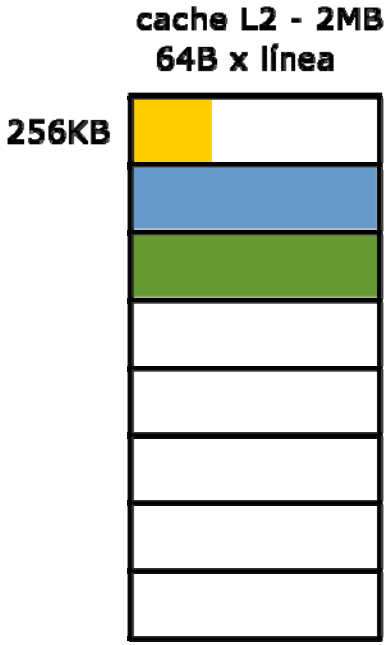
localidad datos – Mat.Mul x Bloques (mejora)

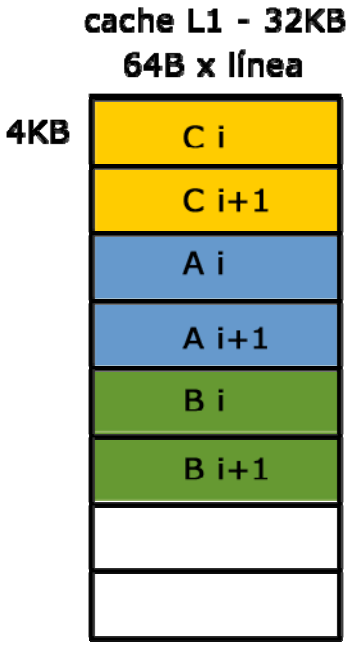
Localidad de los datos
- Afinidad
- Chunksize





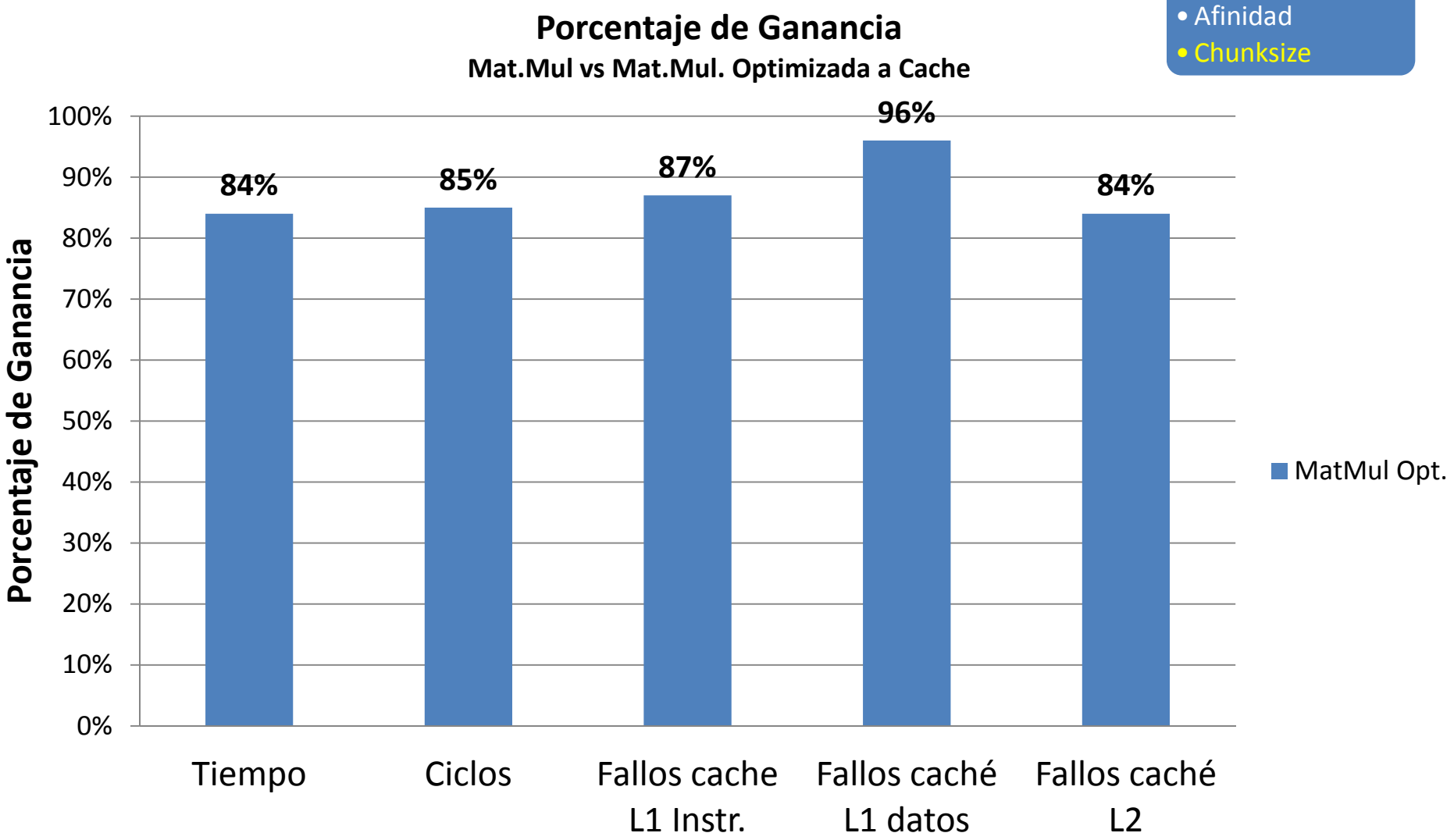






Experimentación III
localidad datos – Mat.Mul x Bloques

Localidad de los datos
- Afinidad
- Chunksize



Cesar.Allande@ caos .uab.es

UAB

Conclusiones (I)

- Se han estudiado factores de rendimiento para un entorno multicore
- Se ha estudiado la implementación GNU OpenMP que ha permitido detectar factores de rendimiento del modelo de programación
- Se ha analizado el patrón de comportamiento de varias aplicaciones, y se ha evaluado el impacto de sintonizar factores de rendimiento

Conclusiones (II)

- Scheduling de la carga de trabajo.
  - Sintonización: dificultad moderada.
  - No se ha conseguido mejorar el rendimiento modificando el tipo de planificación para diferentes cargas de trabajo
    - Static. Poca adaptación con carga desbalanceada
    - Guided y Dinamic. Consiguen balanceo de carga

- Sincronismo de acceso concurrente a memoria.
  - Sintonización: depende de la funcionalidad.
  - Los primitivas ofrecen diferente latencia y flexibilidad
  - Existe un overhead en la utilización de OpenMP.

Conclusiones (III)

- Localidad de los datos
  - Sintonización: dificultad alta
  - Efectuando un cambio en la granularidad de los datos por fuerza bruta, no se ha conseguido una mejora significativa
  - Sintonizando la aplicación para hacer un ajuste en los dos niveles de caché, se ha conseguido una ganancia de 6x. (respecto a la versión paralela previa)

Trabajo actual y Líneas abiertas

- Trabajo actual
  - Ejecución en un entorno Quad (o más cores), para evaluar el rendimiento en la definición de grupos afines en una jerarquía de memoria modular
- Líneas abiertas
  - Continuar con el estudio de impacto de los factores que afectan al rendimiento
  - Estudiar la posibilidad de alterar parámetros de la aplicación de forma dinámica para mejorar su rendimiento
  - Estudiar los factores que será necesario monitorizar para detectar los problemas de rendimiento
  - Sentar las bases que permitan definir un modelo de rendimiento para entornos multicore

¡ Gracias por vuestra atención !

Cesar.Allande@ caos .uab.es

UAB

Experimentación IV
localidad datos – Mat.Mul x Bloques

Localidad de los datos
- Afinidad
- Chunksize

Tam Bloque	Tiempo	Ciclos	Cache L1 Fallos Instr.	Cache L1 Fallos Datos	Cache L2 Fallos
64x 64	**6,24 seg.**	**1,62E+10**	**3,86E+03**	**3,40E+07**	**4,79E+05**
64 x 64 opt.	**0,956 seg.**	**2,42E+09**	**4,75E+02**	**1,10E+06**	**7,41E+04**