Sintonización Dinámica de Aplicaciones MPI

Presentado por:
Andrea Martínez Trujillo

Directores:
Anna B. Morajko
Joan Sorribes

13 de Julio 2010

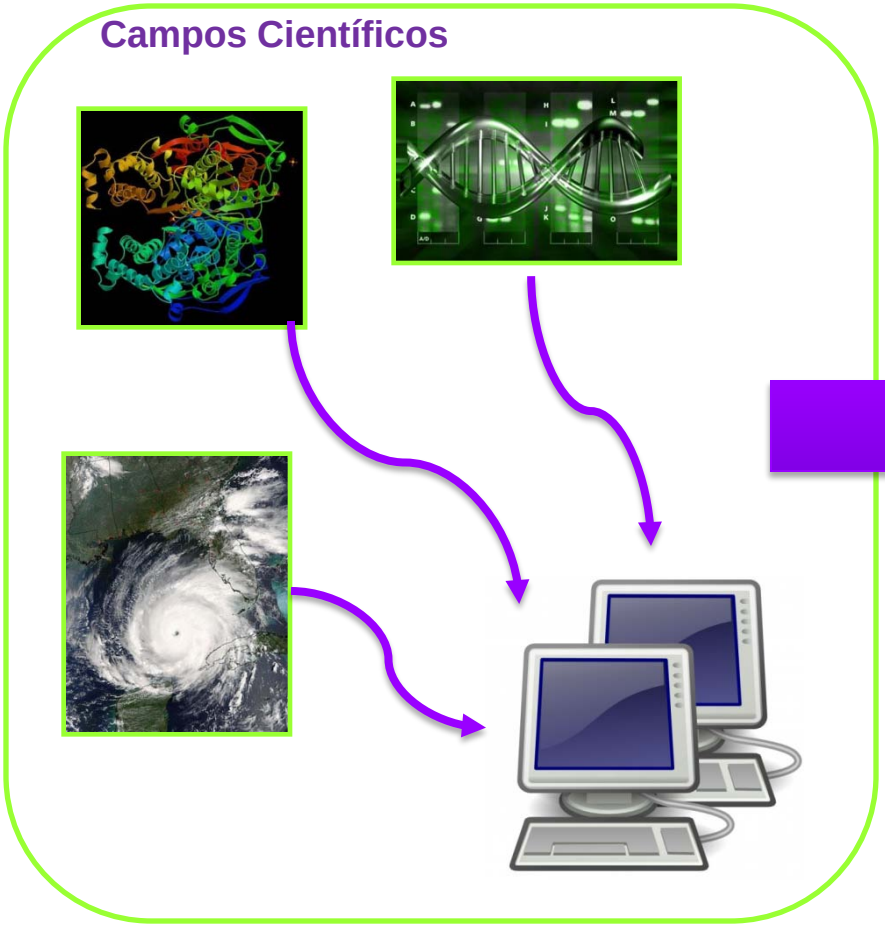
Departamento de Arquitectura de Computadores y Sistemas Operativos
Universidad Autónoma de Barcelona

Contenido

- Motivación
- Objetivo
- Proceso de sintonización dinámica
- Conclusiones
- Trabajo futuro

Motivación

Contexto



Paralelas / Distribuidas

Gr ...s

HPC

Sofisticados cálculos complejos

Desafío

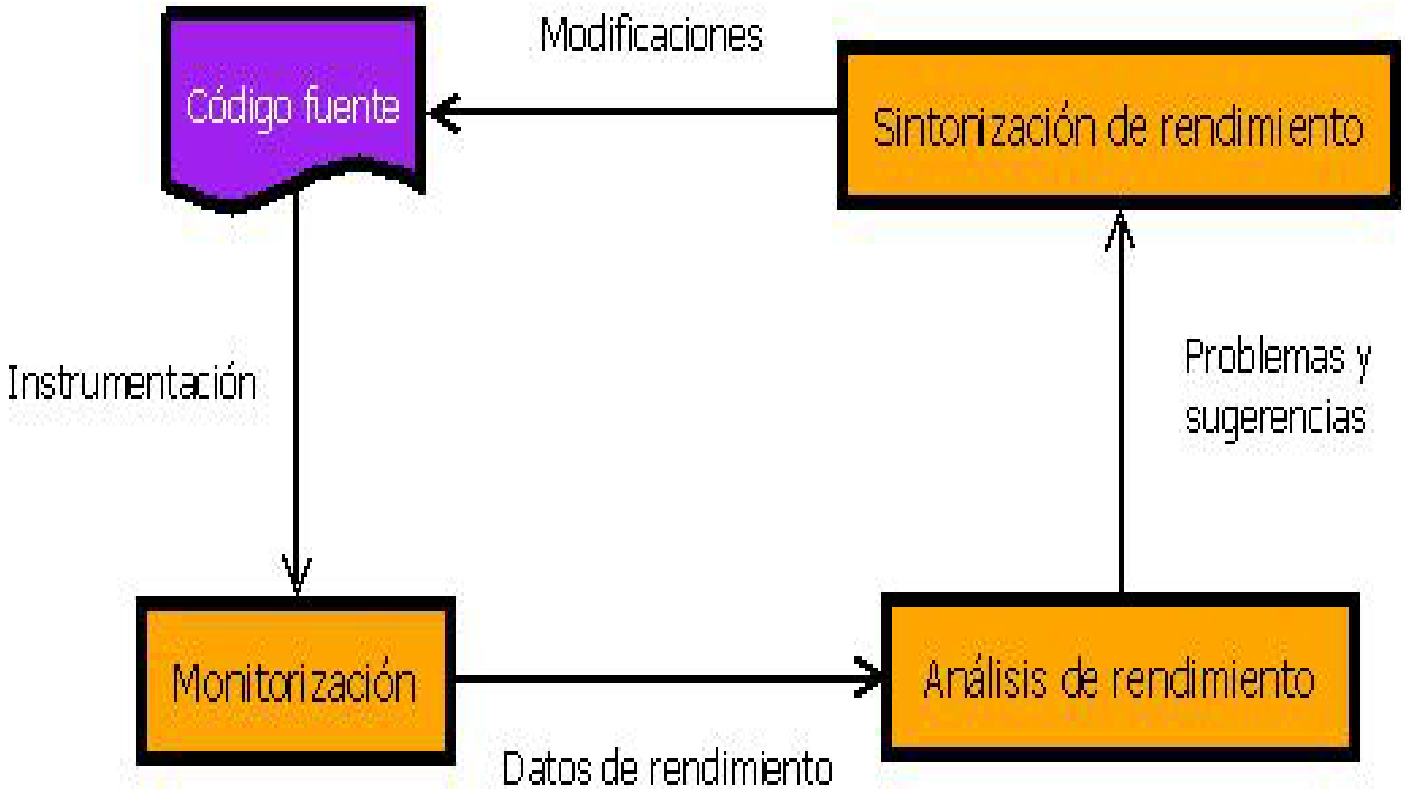
Eficiencia

- Aprovechar los recursos y altas capacidades computacionales

Rendimiento

- Punto clave
- Requiere un alto grado de experiencia

Problema


Código fuente
Modificaciones
Sintorización de rendimiento
Instrumentación
Problemas y sugerencias
Monitorización
Análisis de rendimiento
Datos de rendimiento

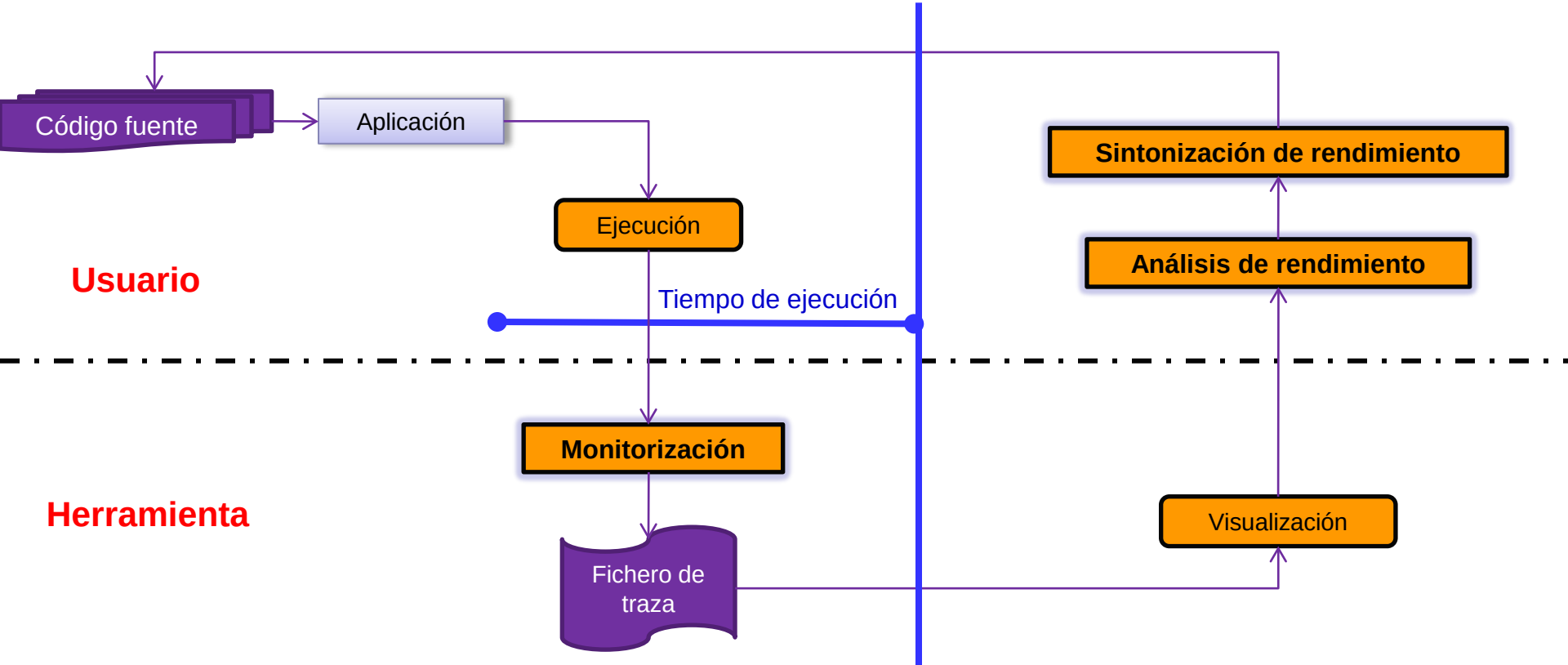
Análisis Clásico

Aplicación

Comportamiento estático

Herramientas

- MPICL
- PARAGRAPH
- PABLO
- VAMPIR



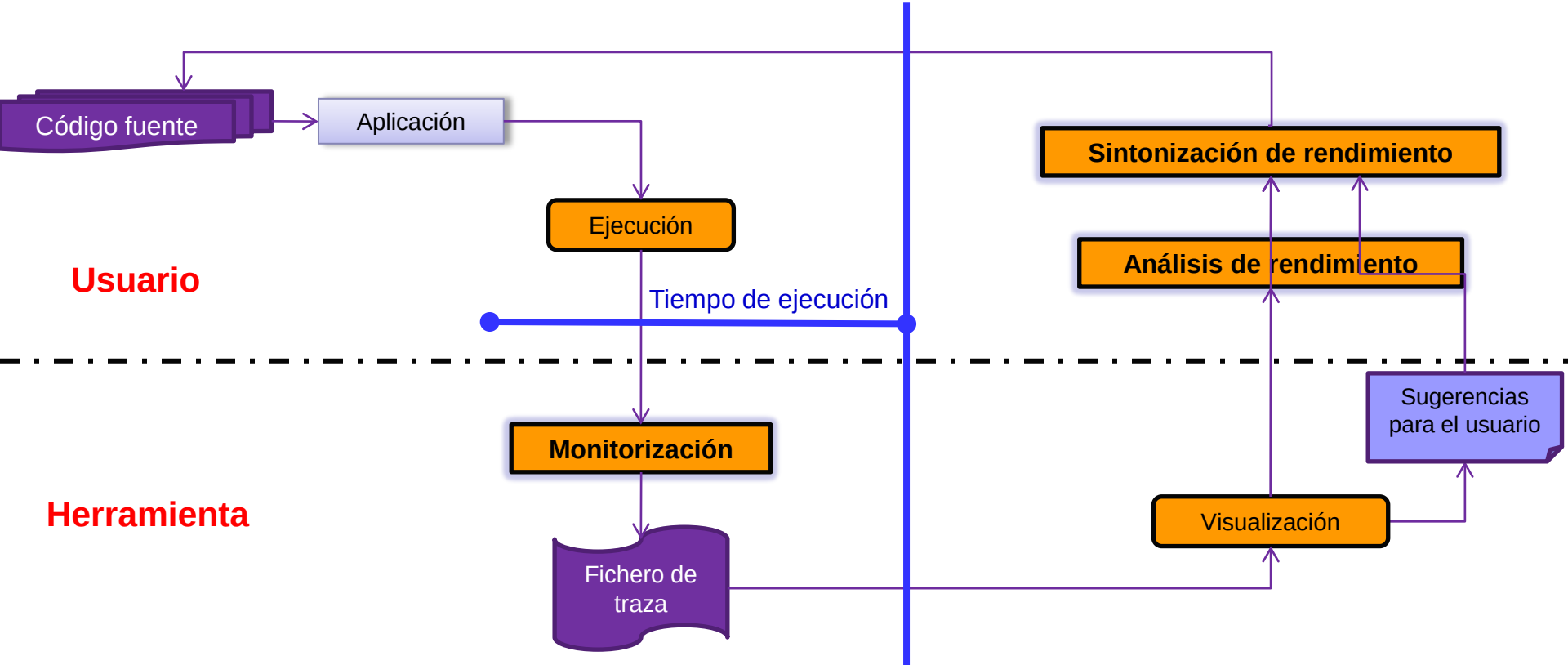
Análisis Automático

Aplicación

Comportamiento estático

Herramientas

- SCALASCA
- TAU
- PERISCOPE
- KAPPAPI



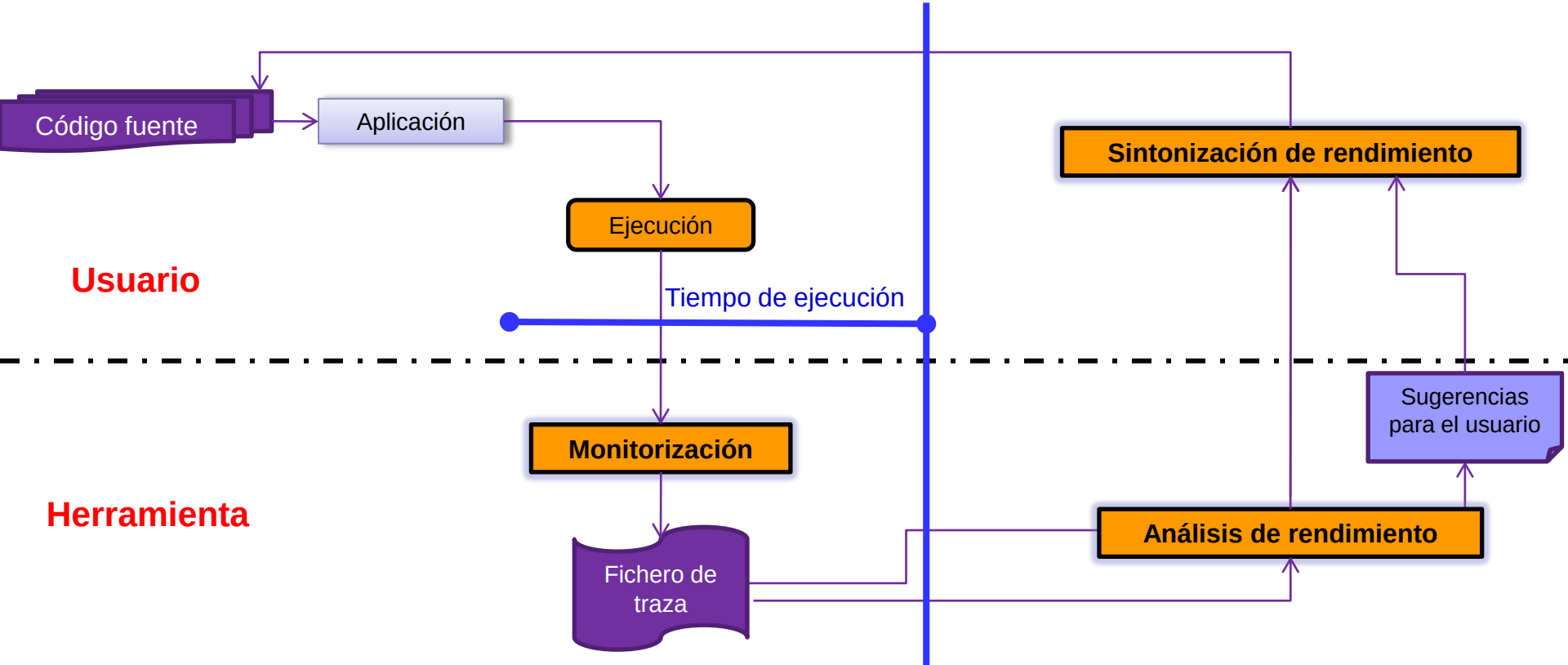
Análisis Dinámico

Aplicación

Comportamiento estático

Herramientas

•PARADYN



Sintonización Dinámica

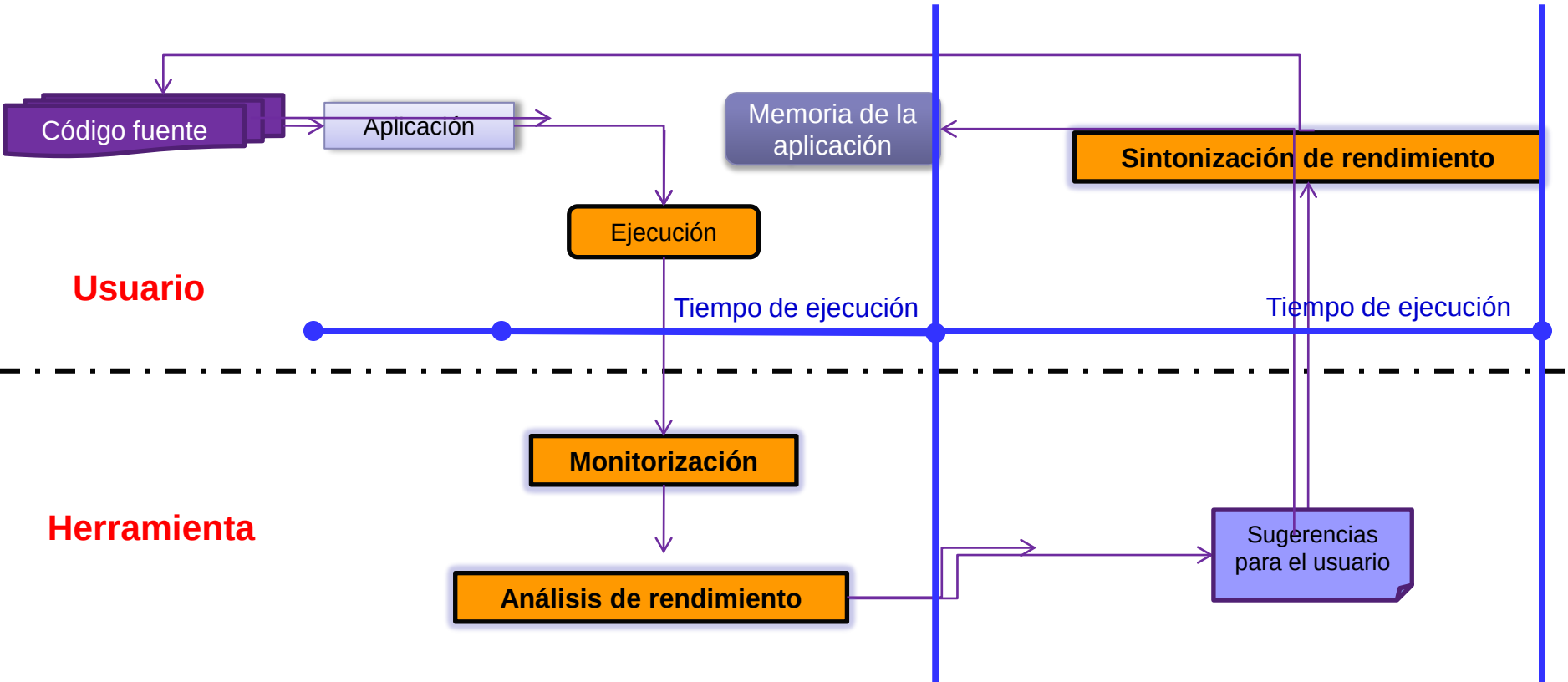
Aplicación

Comportamiento dinámico

Herramientas

- AUTOPILOT
- ACTIVE HARMONY
- PERCO

MATE



Sintonización dinámica

MATE — Monitoring, Analysis and Tuning Enviroment

- Diseñada en la UAB, en el departamento de Arquitectura de Computadores y Sistemas Operativos.
- Basada en la técnica de instrumentación dinámica.
- Inicialmente estaba pensada para sintonizar aplicaciones PVM paralelas/distribuidas desarrolladas en C/C++ ejecutándose en plataformas UNIX.
- Actualmente ha sido desarrollada para sintonizar aplicaciones MPI.

Objetivo

Objetivo

Sintonizar dinámicamente mediante **MATE** una aplicación MPI empleada en computación de altas prestaciones que siga un paradigma **Master/Worker**

Proceso de sintonización dinámica

Departamento de Arquitectura de Computadores y Sistemas Operativos
Universidad Autónoma de Barcelona

Estudio

- MATE
- Modelo de rendimiento
- Aplicación a sintonizar: **Xfire**

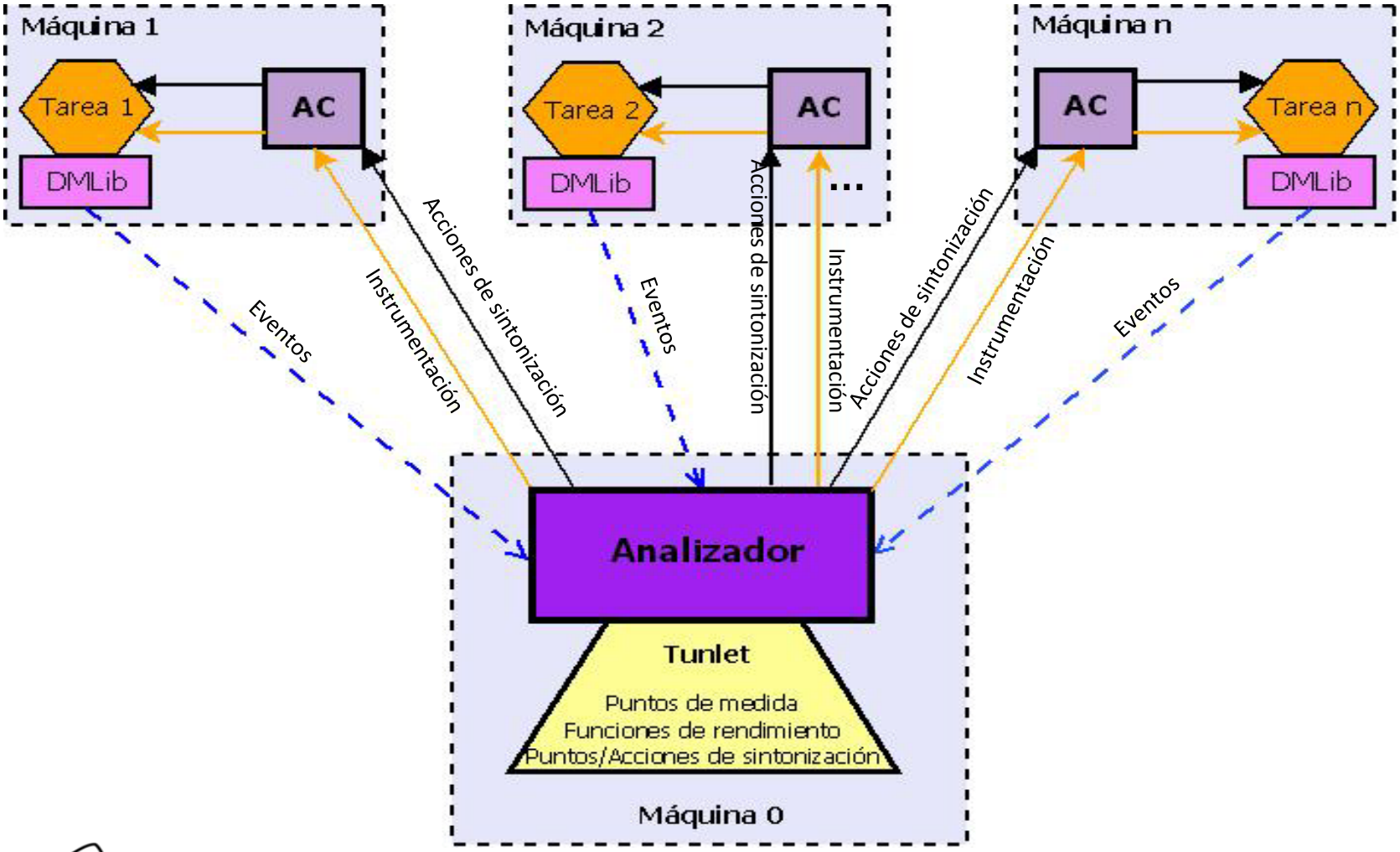
Diseño y desarrollo

- Lógica de análisis de rendimiento

Experimentación

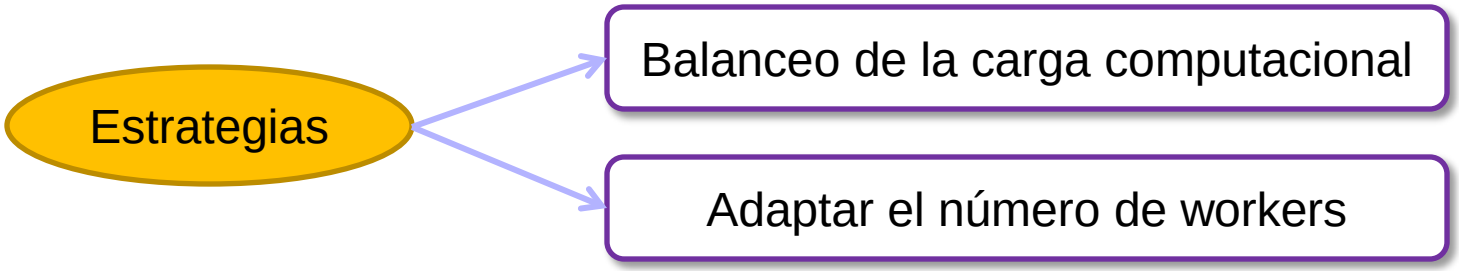
- Evaluar sobrecarga introducida por MATE
- Estudiar la mejora de rendimiento en la aplicación sintonizada

Estudio de MATE



Modelo de rendimiento

E. Cesar, A. Moreno, J. Sorribes, E. Luque, ***Modeling master/worker applications for automatic performance tuning***. *Parallel Comput.* 32, 7, pp. 568-589 (2006).



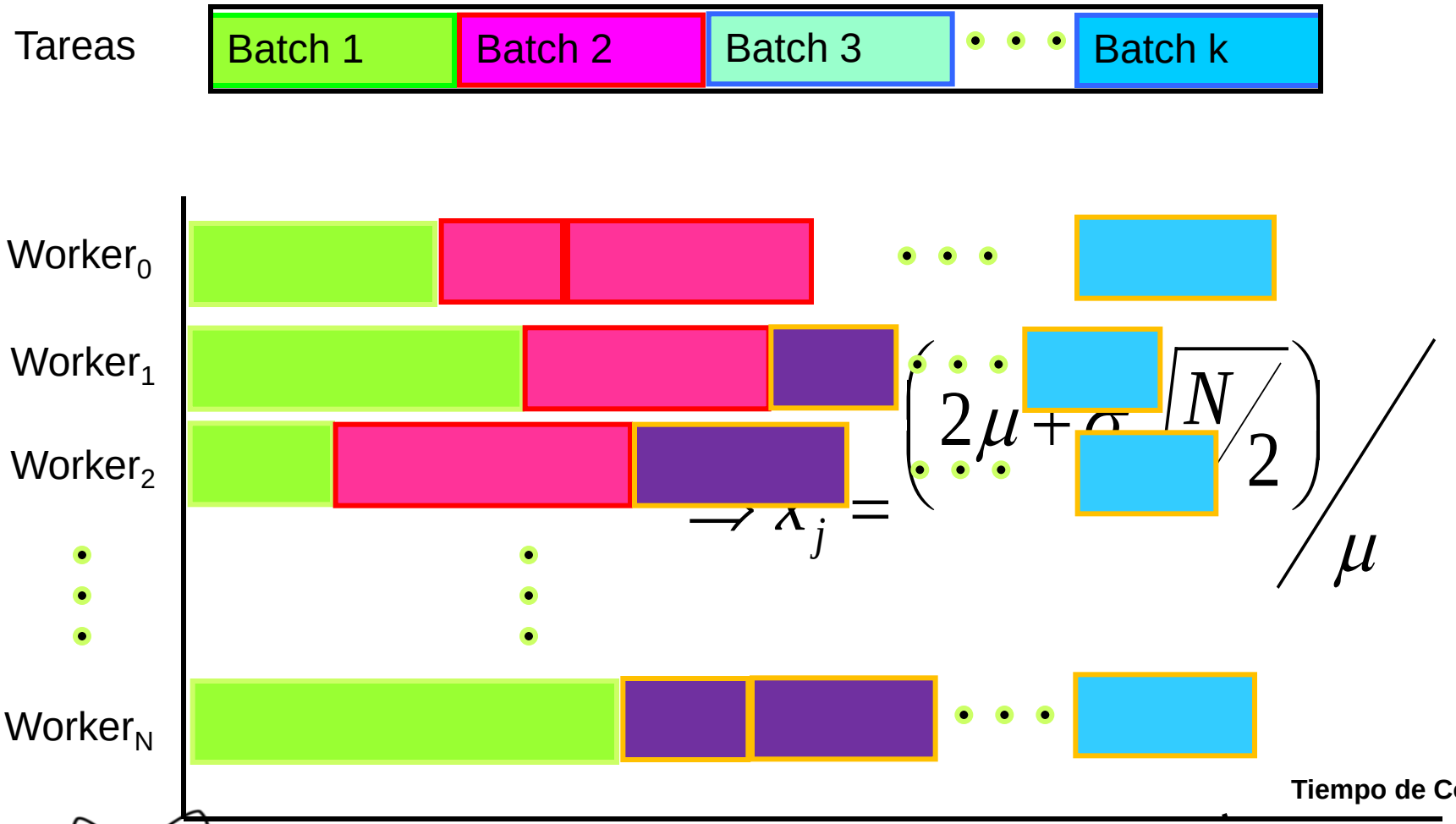
Balanceo de Carga

Conjunto completo de tareas

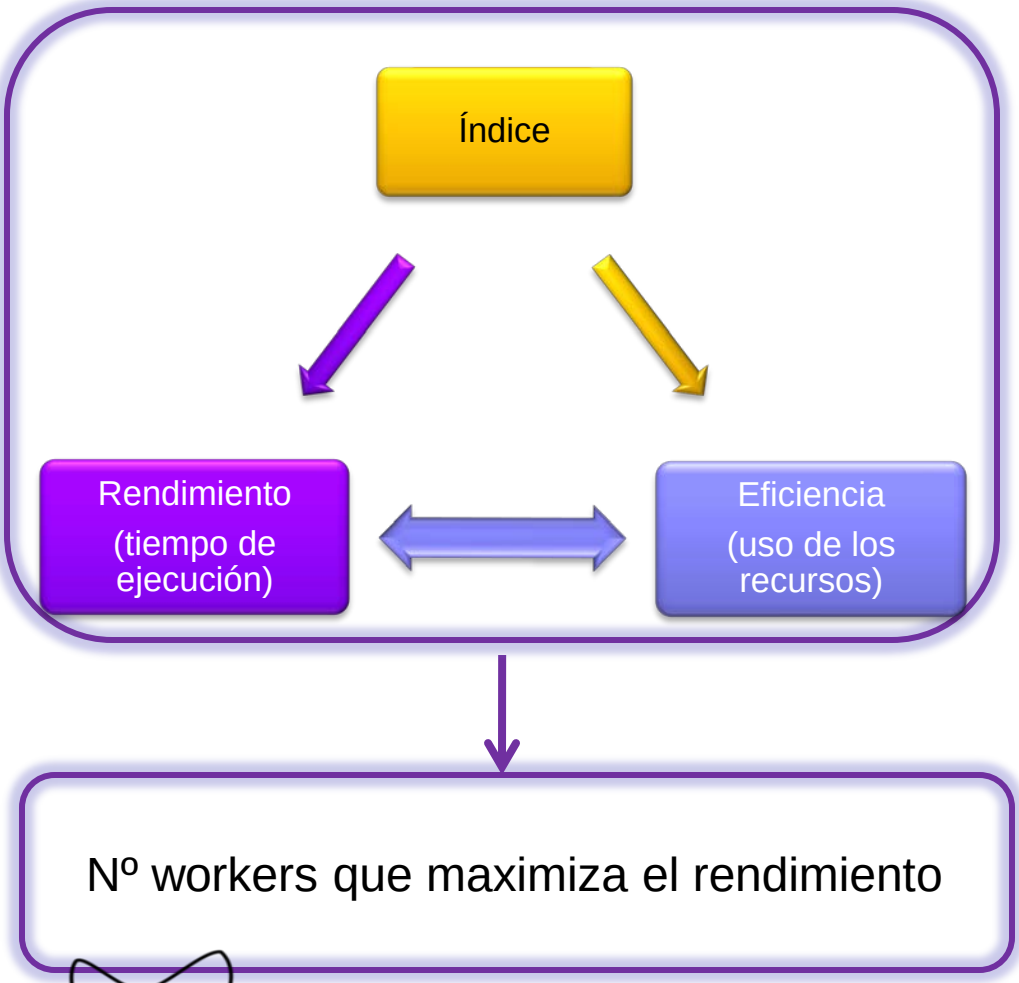
Dynamic Adjusting Factoring

El Master realiza una distribución parcial dividiendo el conjunto de tareas en diferentes porciones llamadas batch

Modelo de rendimiento



Adaptar número de Workers



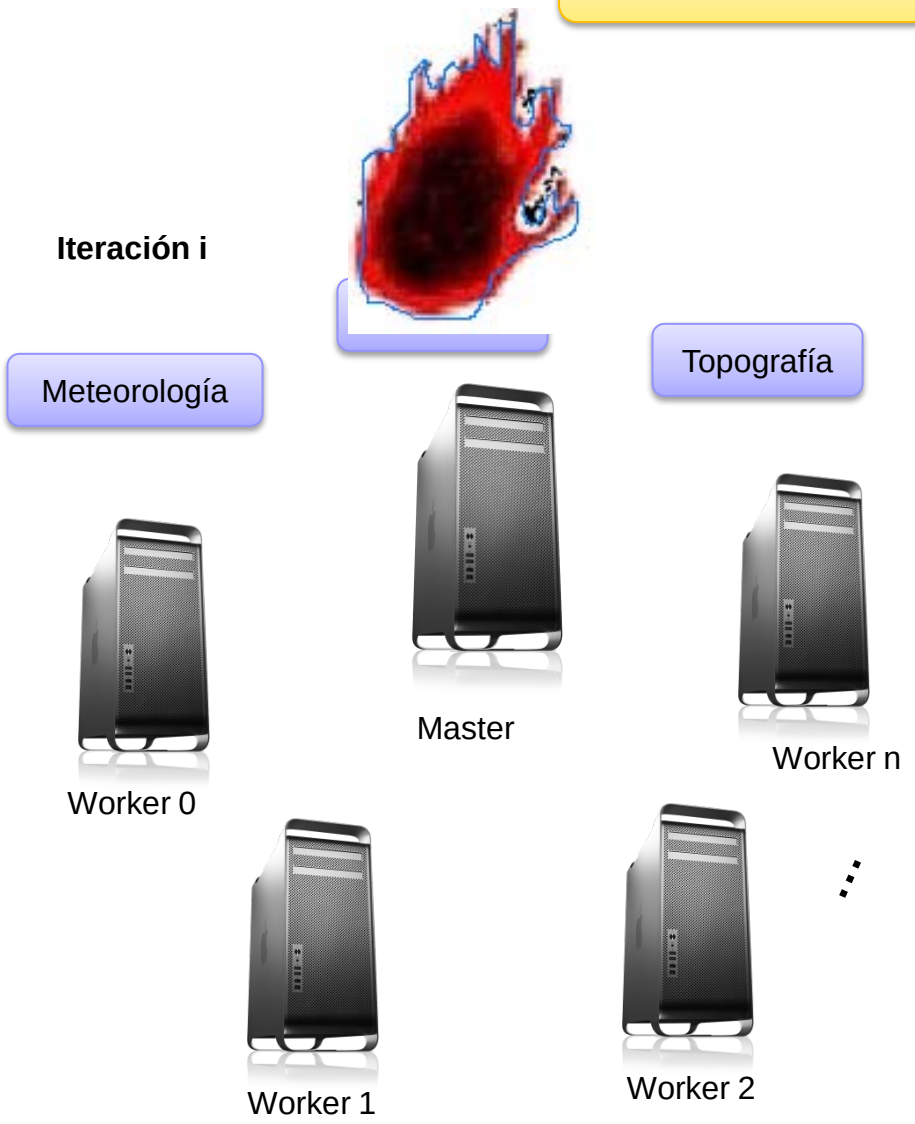
$$Tt = 2m_o + \frac{\left[\left(\left(n-1\right)\alpha + 1\right)\lambda V + Tc\right]}{n}$$

$$E(x) = \frac{Tc}{xTt(x)}$$

$$Pi(x) = \frac{Tt(x)}{E(x)}$$

Xfire

- Aplicación paralela desarrollada en CAOS
- Sigue un paradigma Master/Worker
- Emplea la librería de paso de mensajes MPI
- Simula la propagación de la línea de fuego en incendios forestales



Estudio

MATE

Modelo de rendimiento

Aplicación a sintonizar: **Xfire**

Diseño y desarrollo

Lógica de análisis de rendimiento

Experimentación

Evaluar sobrecarga introducida por MATE

Estudiar la mejora de rendimiento en la aplicación sintonizada

Fases de desarrollo

I. Se ha implementado en el proceso Master de Xfire la lógica que hay que aplicar para realizar el balanceo de carga dinámico.

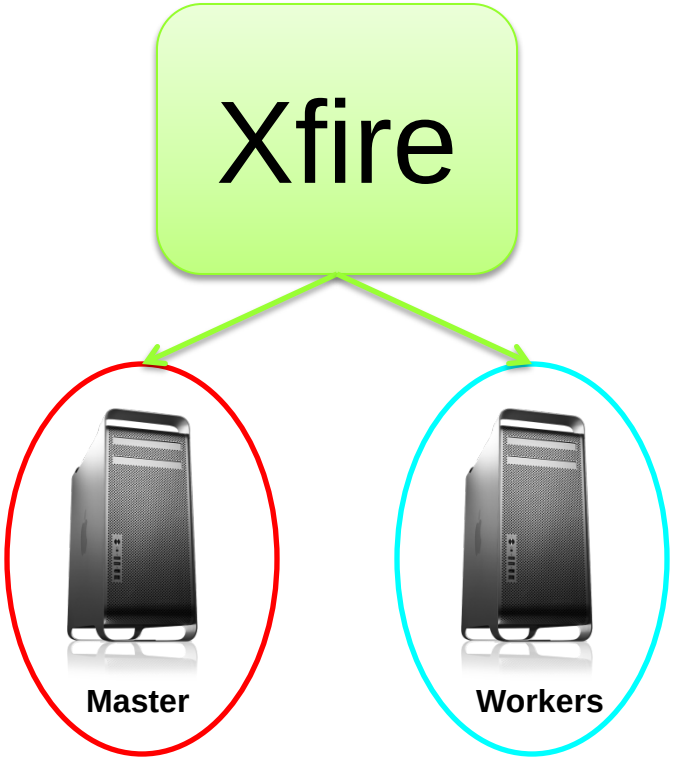
II. Se ha diseñado y desarrollado el tunlet que constituye la lógica de análisis de rendimiento de MATE.

Tunlet

- Define e implementa una técnica particular de sintonización.

Identificación de los procesos de la aplicación

- Para cada proceso:
  - Distinta instrumentación.
  - Distintos eventos capturados



Identificación de información/variables/valores

¿Qué sintonizamos?

- Nº Workers
- Factor de partición para el balanceo de carga

¿Cuándo sintonizamos?

- Nº Workers → Inicio de iteración.
- Factor de partición → A lo largo de la iteración

¿Qué valores necesitamos?

Identificación de información/variables/valores

Adaptar Nº Workers

$$T_t = 2m_o + \frac{\left[\left(\left(n-1\right)\alpha+1\right)\beta V + Tc\right]}{n}$$

+

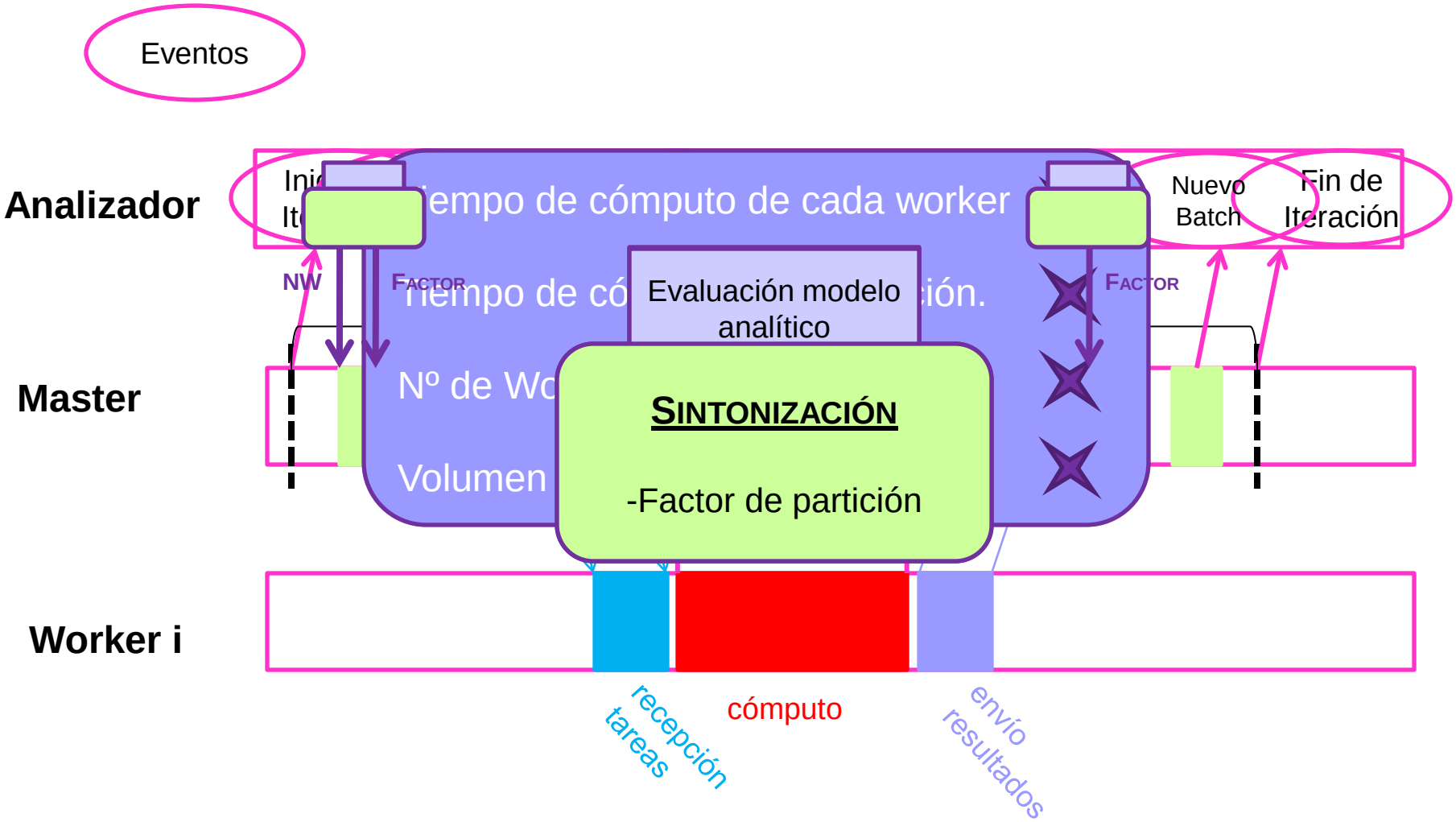
Datos de control

Balanceo de Carga

$$= \mu + \sigma\sqrt{N/2}$$

- Tiempo de cómputo de cada worker.
- Tiempo de cómputo total iteración.
- Nº de Workers
- Volumen total de comunicación

Identificación de eventos



Estudio

MATE

Modelo de rendimiento

Aplicación a sintonizar: **Xfire**

Diseño y desarrollo

Lógica de análisis de rendimiento

Experimentación

Evaluar sobrecarga introducida por MATE

Estudiar la mejora de rendimiento en la aplicación sintonizada

Experimentación

Plataforma de cómputo

- Clúster homogéneo (clúster D):
  - 10 nodos → 8 son de cómputo.
  - Procesador: Intel Pentium 4, 3.0GHz
  - Red 1Gbps.

Experimentos

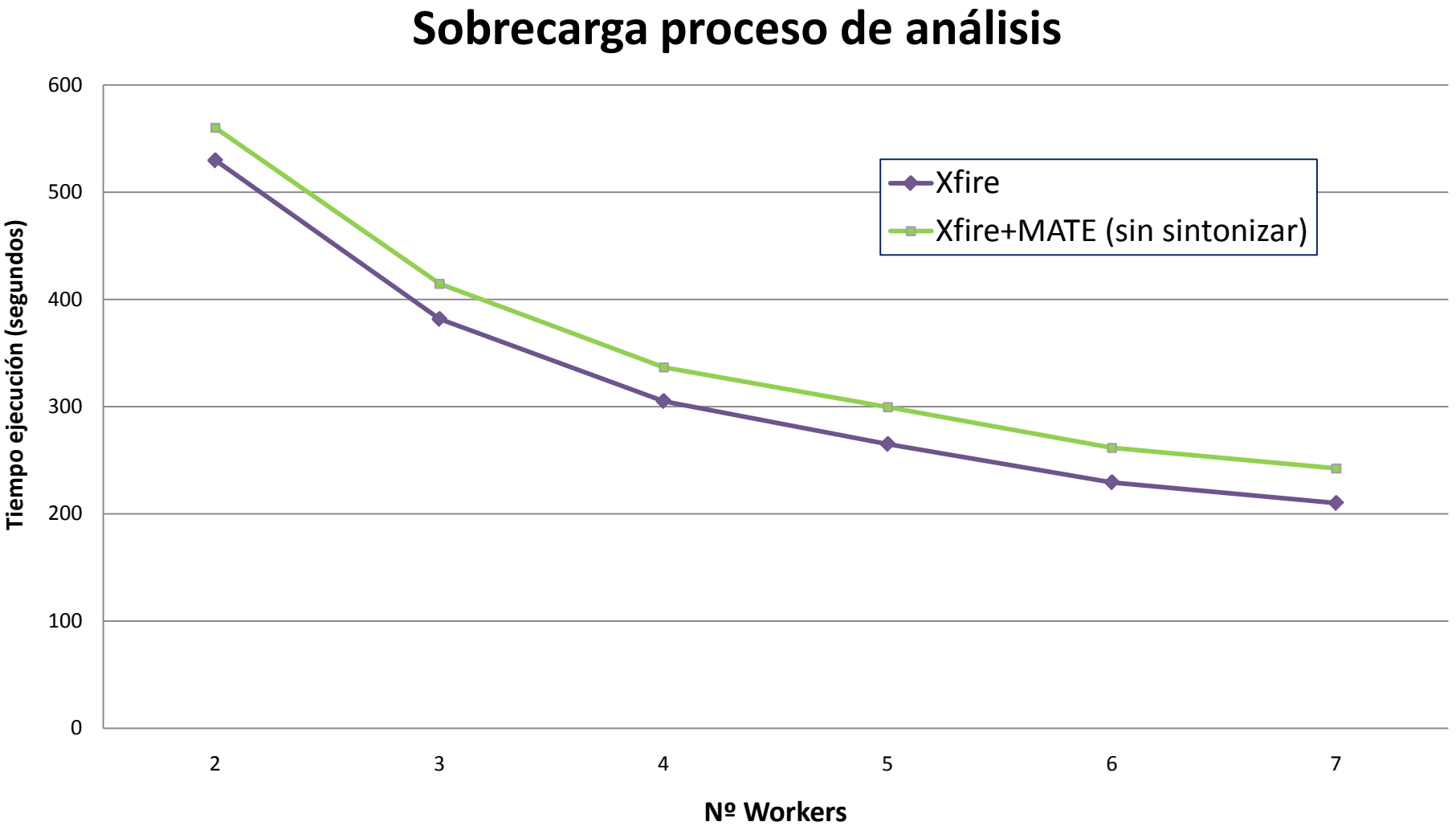
- Datos de entrada: línea de fuego 786420 puntos.
- Nº de workers: 2, 3, 4, 5, 6, 7.

Estudio de la sobrecarga generada por MATE

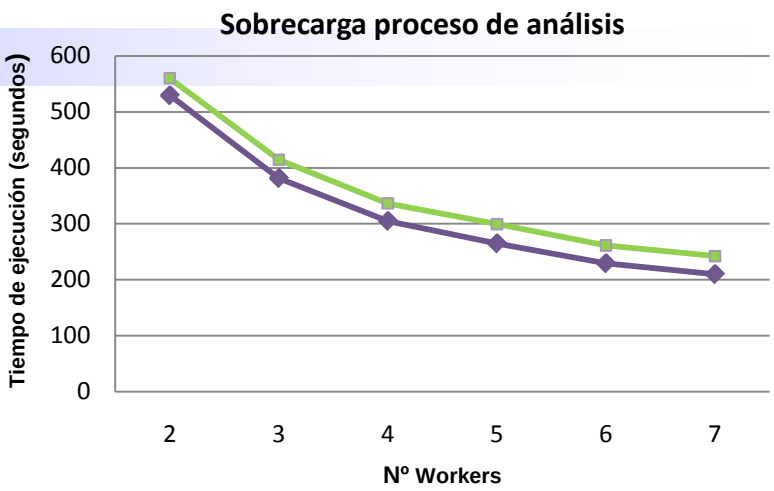
Escenario 1: Xfire

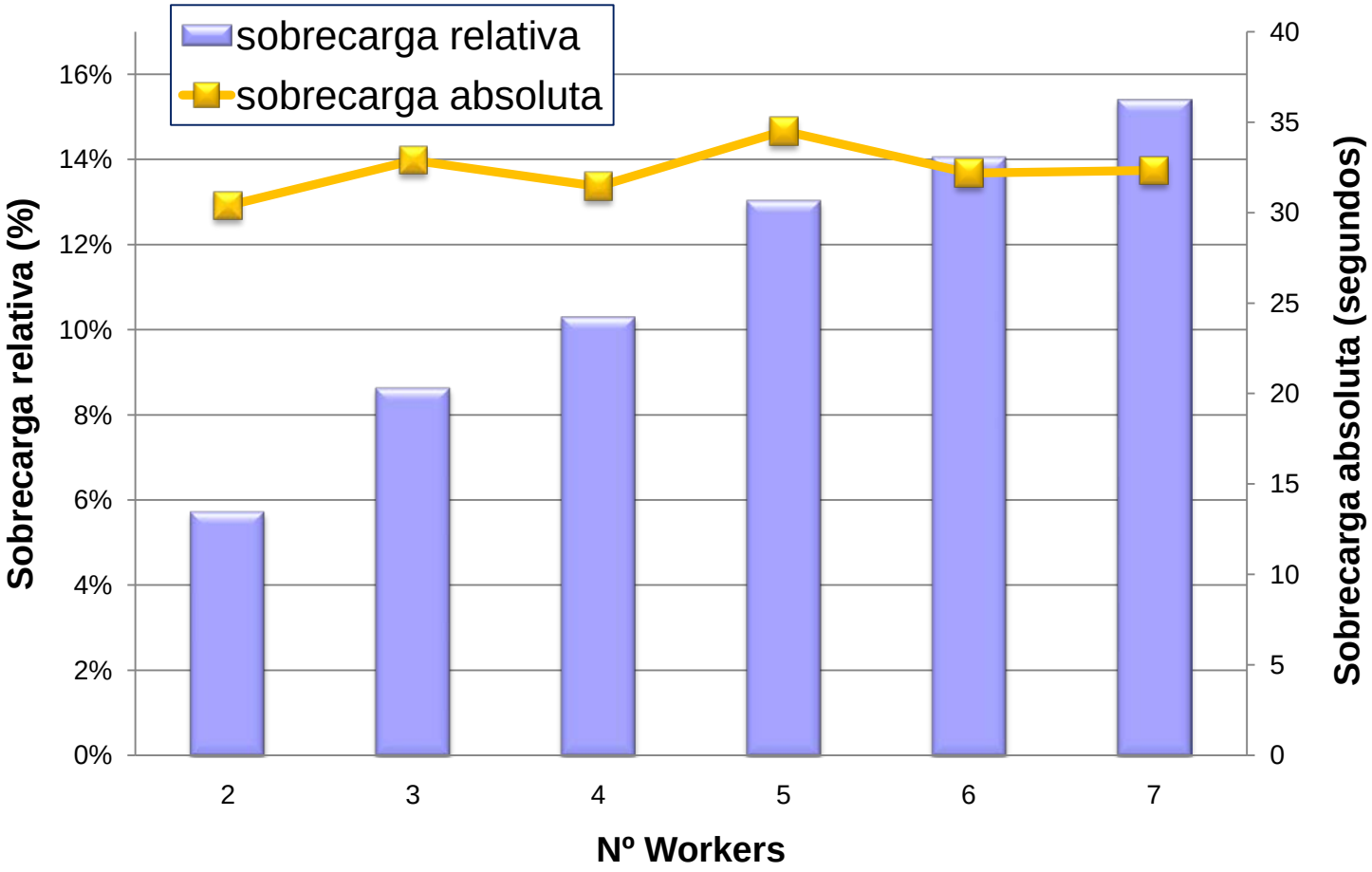
Escenario 2: Xfire+MATE (sin sintonizar)

Estudio de la sobrecarga generada por MATE



Estudio de la sobrecarga generada por MATE



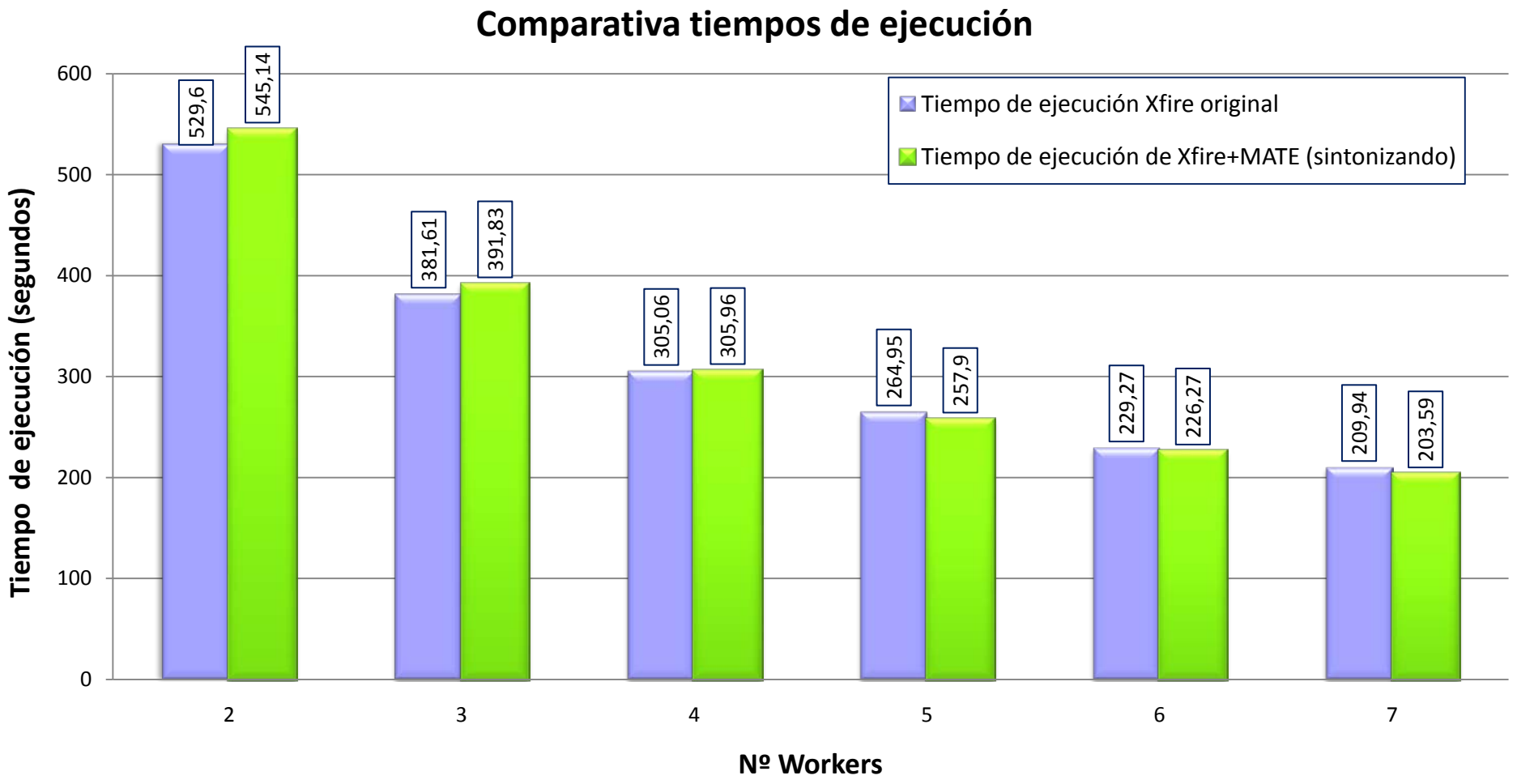


Estudio del rendimiento de Xfire con sintonización

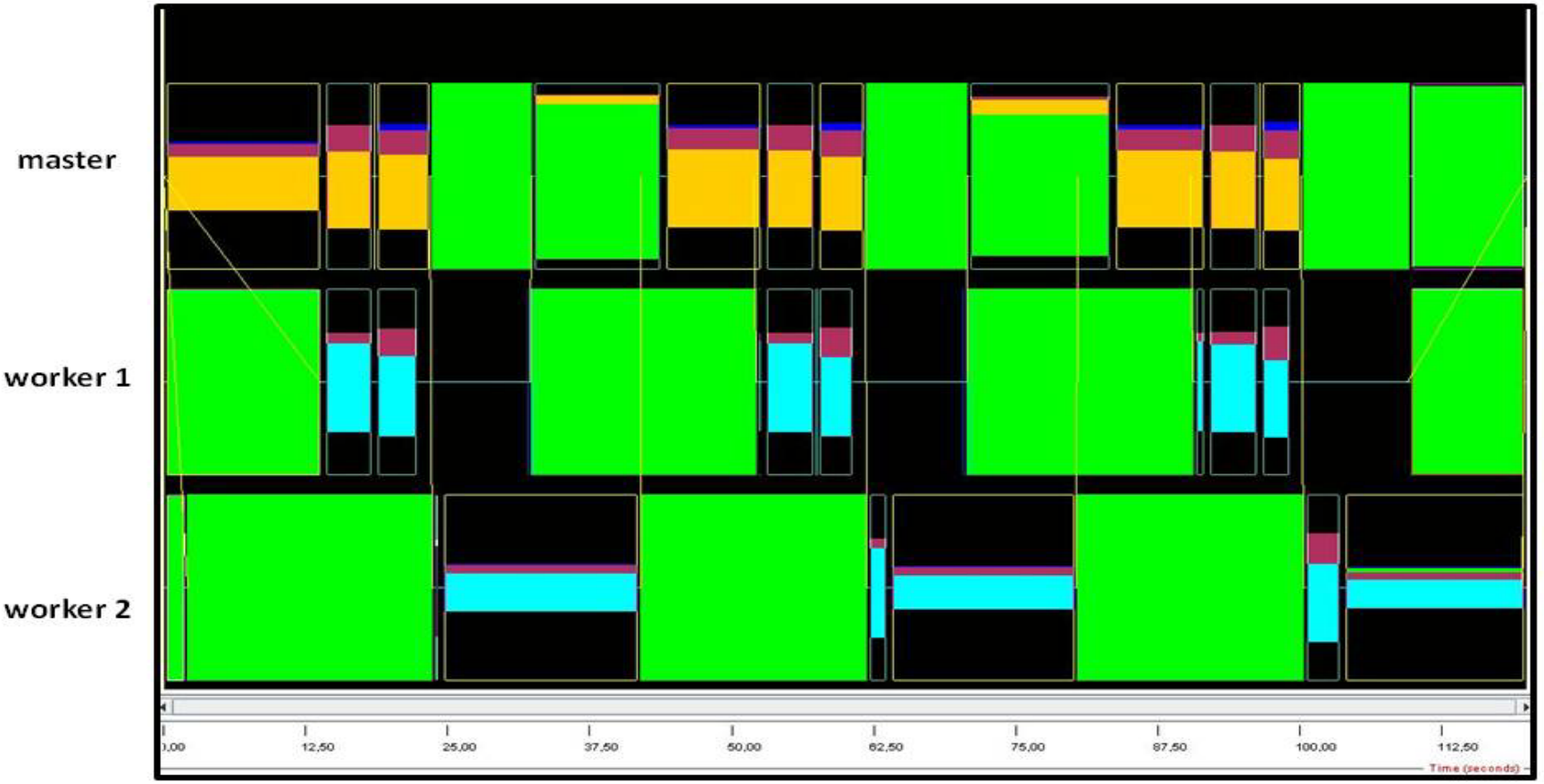
Escenario 1: Xfire

Escenario 2: Xfire+MATE (sintonizando)

Estudio del rendimiento de Xfire con sintonización



Estudio del rendimiento de Xfire con sintonización



Estados de espera

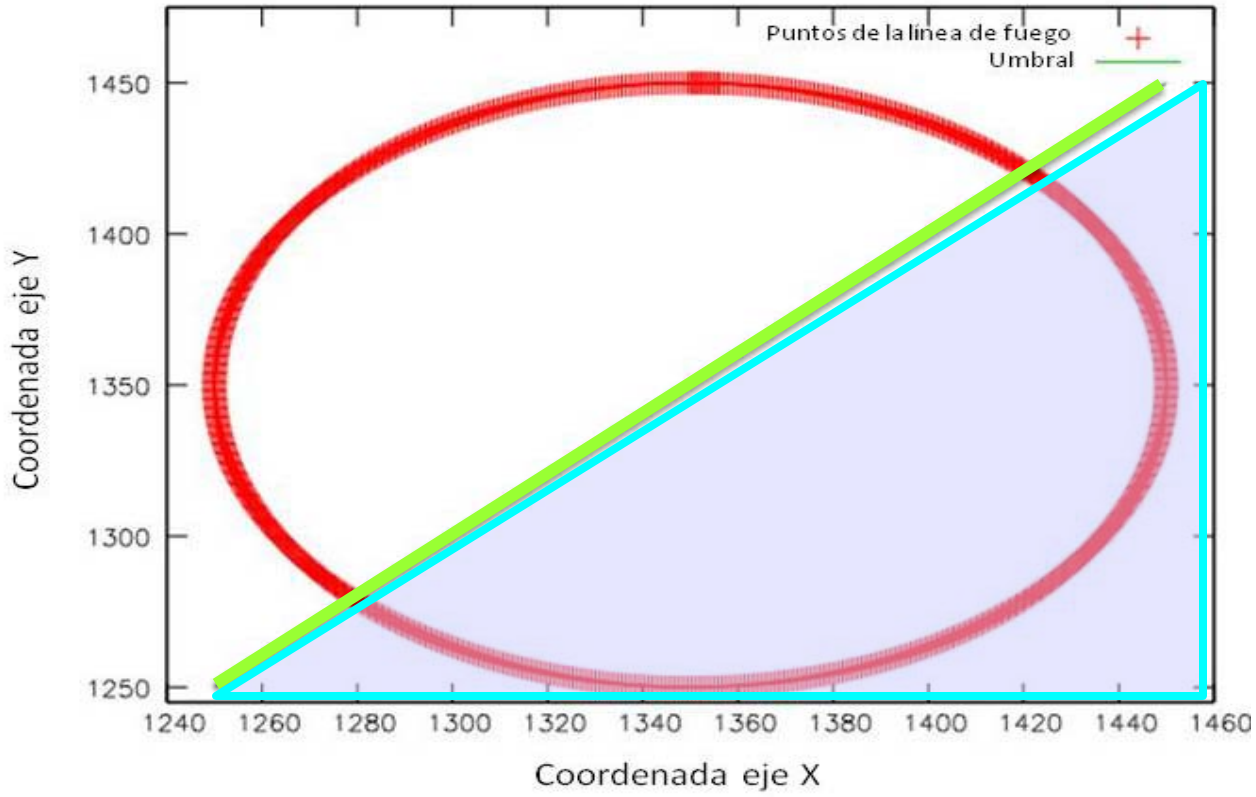
Preparación de datos

Estudio del rendimiento de Xfire con sintonización

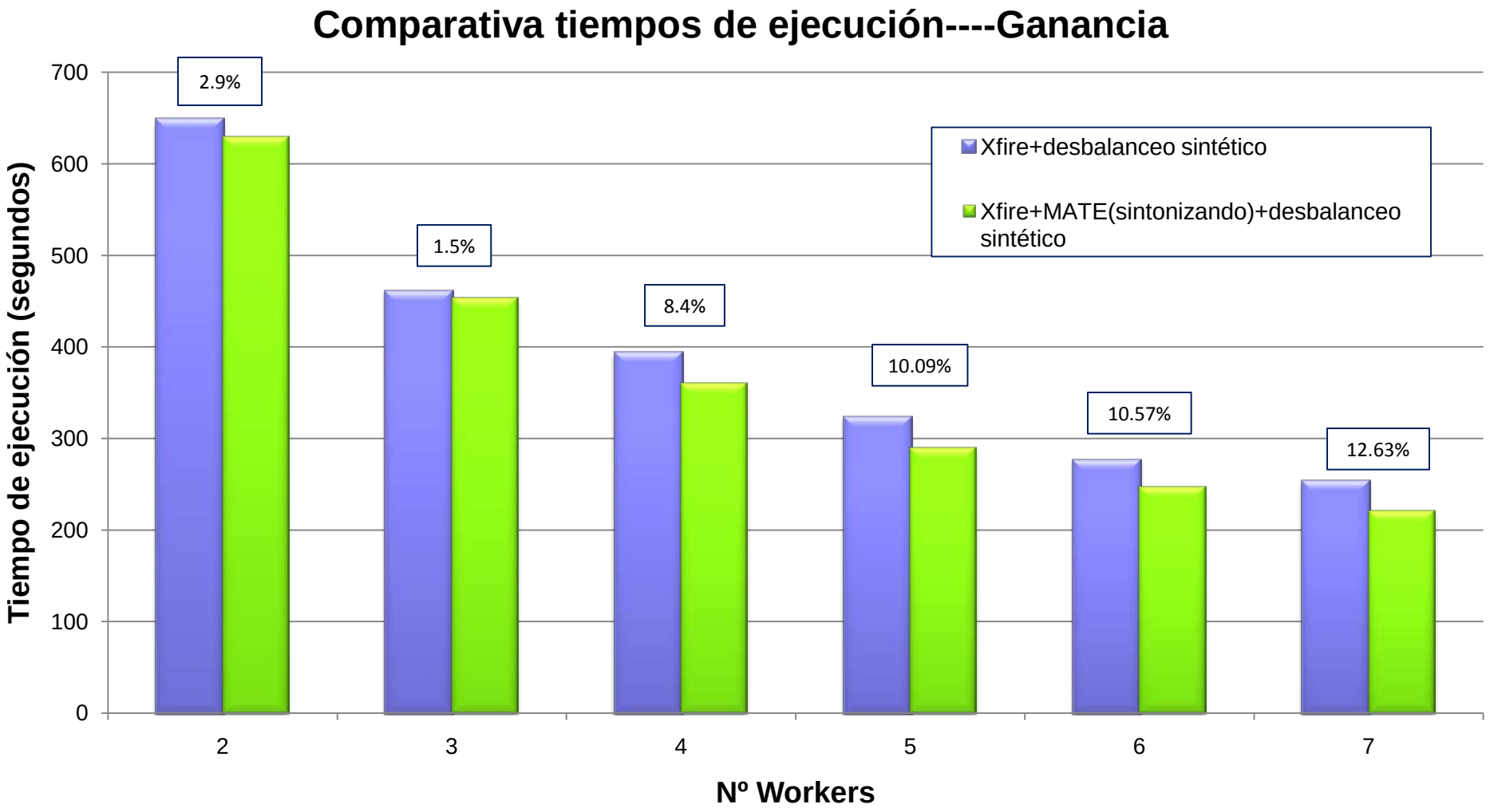
Idea: acentuar el desbalanceo de Xfire

- Tratar algunos puntos de la línea de fuego como tareas más pesadas (necesitan más tiempo de cómputo).
- Estudio de la línea de fuego.

Estudio del rendimiento de Xfire con sintonización



Estudio del rendimiento de Xfire con sintonización



Conclusiones

Conclusiones

- Se ha adquirido el conocimiento necesario para trabajar con MATE, como entorno de sintonización de aplicaciones paralelas/distribuidas.
- Se ha adaptado la política de balanceo de carga de Xfire para ser aplicada a la lógica del modelo de rendimiento.
- Se han diseñado e implementado el tunlet que contiene la lógica de análisis del modelo de rendimiento estudiado.

Conclusiones

- Se han realizado pruebas experimentales:
  - Sobrecarga de MATE es constante e independiente del nº de workers que participan en la ejecución de la aplicación.
  - El tunlet implementado es eficaz ya que los resultados muestran una mejora en el rendimiento de Xfire cuando es ejecutada bajo MATE.

Trabajo Futuro

Trabajo Futuro

- A corto plazo
  - Terminar la depuración de la técnica de sintonización para adaptar el número de workers y realizar la
- A largo plazo
  - Estudio y mejora de la escalabilidad de MATE:
    - Esquema de comunicación jerarquizado.
    - Análisis de rendimiento distribuido.

  escalabilidad de MATE.
  - Aplicar la estrategia de sintonización desarrollada sobre otras aplicaciones

Gracias por su atención

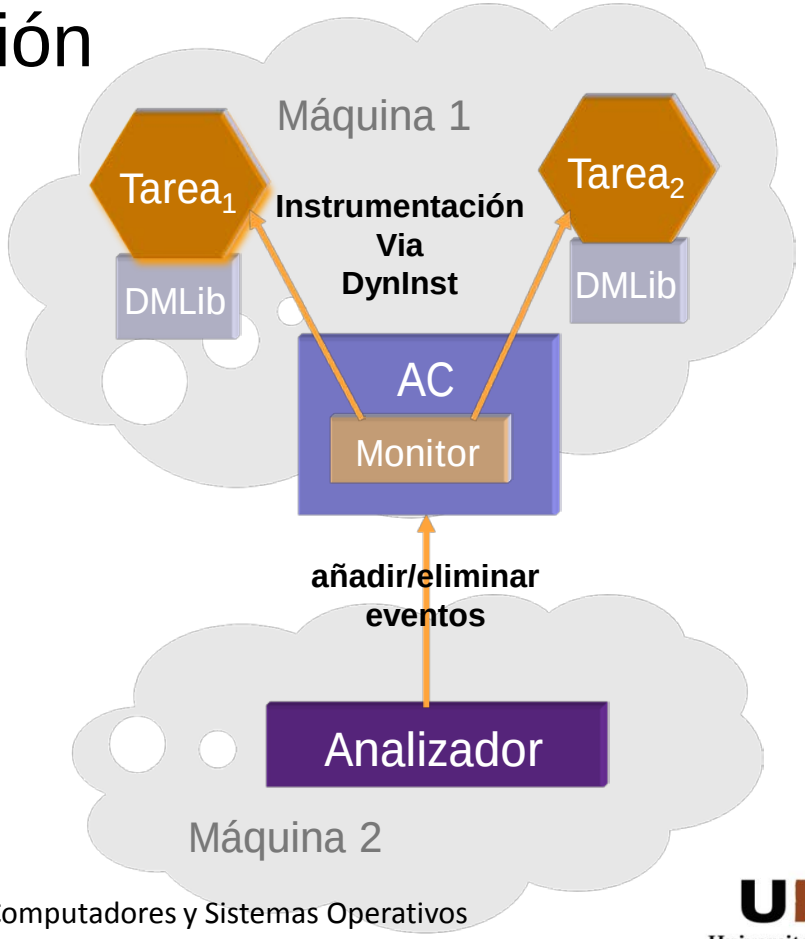
Andrea Martínez Trujillo
amartinez@caos.uab.es

Departamento de Arquitectura de Computadores y Sistemas Operativos
Universidad Autónoma de Barcelona

Estudio de MATE

- Estudiar la herramienta MATE
- Modificar la implementación de MATE para que sintonice aplicaciones MPI
- Localizar una aplicación paralela/distribuida que siga un paradigma Master/Worker
- Estudio del modelo de rendimiento Master/Worker
- Diseño y desarrollo del tunlet que contiene la lógica asociada al modelo de rendimiento usado
- Realizar experimentación

- Controlador de Aplicación
  - Monitor
    - Carga DMLib
    - Crea snippets
    - Inserta/Elimina snippets



Estudio de MATE

- Estudiar la herramienta MATE
- Modificar la implementación de MATE para que sintonice aplicaciones MPI
- Localizar una aplicación paralela/distribuida que siga un paradigma Master/Worker
- Estudio del modelo de rendimiento Master/Worker
- Diseño y desarrollo del tunlet que contiene la lógica asociada al modelo de rendimiento usado
- Realizar experimentación

- Controlador de Aplicación
  - Monitor
  - Sintonizador
    - Genera los snippets de sintonización
    - Inserta los snippets de sintonización