**Introducción a las arquitecturas de sistemas distribuidos**

**Eduardo Gimeno Soriano 721615**

**Sergio Álvarez Peiro 740241**

**23 de octubre de 2019**

# Introducción

El sistema consta de tres escenarios diferentes y está formado por dos elementos diferenciados. Un cliente que realiza múltiples peticiones dependiendo del escenario y un servidor que las resuelve y envía la respuesta correspondiente.

La arquitectura del servidor cambia para cada escenario dependiendo de la carga de trabajo que implique cada escenario. Las arquitecturas que se han tenido en cuenta han sido servidor secuencial, servidor concurrente y master-worker.

# Sección Principal

Para cada escenario se ha creado un fichero elixir diferente, para el escenario 1 escenario1.exs, para el escenario 2 escenario2.exs y para el escenario 3 escenario3.exs.

## Características técnicas de las máquinas utilizadas

Para realizar esta práctica se han utilizado tres máquinas distintas:

* Máquinas lab102-XXX (siendo XXX un número comprendido entre 191 y 210)
  + - Núcleos: 4
    - Memoria: 11 GB
    - Disco: 1 TB (500 MB por usuario)
* Máquina ASUS (Eduardo)
  + - Núcleos: 4
    - Memoria: 8 GB
    - Disco: 1 TB
* /\* Máquina Sergio \*/

Las máquinas personal se han utilizado para probar únicamente que el código desarrollado funcionaba bien, para el despliegue del sistema se han utilizado las máquinas de laboratorio 1.02.

## Análisis gráfico de la carga de trabajo para cada escenario

## Patrones arquitecturales para los distintos escenarios

Para resolver los escenarios planteados en la práctica se ha seguido una metodología icremental, es decir, se ha comenzado desde la solución más sencilla y a partir de ahi se ha ido evolucionando según las necesidades de cada escenario.

Partiendo de lo anterior, para resolver el primer escenario se ha comenzado probando con una arquitectura cliente-servidor secuencial. Para comprobar que era la solución idónea se ha comprobado para cada petición del cliente que la QoS no era violada, en la sección Validación Experimental se informa más al detalle de la comprobación de cada solución. Para este primer escenario esta solución ha sido suficiente, ya que la QoS se respetaba en cada petición del cliente.

Para el segundo escenario se ha partido del escenario anterior, pero la solución cliente-servidor secuencial para este escenario no cumplía la QoS, por tanto se ha pasado a la siguiente arquitectura, cliente-servidor concurrente. Esta solución si ha sido suficente para este escenario, respetando la QoS para cada petición del cliente.

Para el úñtimo escenario se ha seguido el mismo procedimiento, se ha partido de la solución del escenario 2 y se ha comprobado que la QoS no se respetaba, por tanto se ha pasado a la siguiente arquitectura, master-worker. En una primera instancia se ha probado con un único worker. Como esta arquitectura presenta dos posibles soluciones, se ha optado por establecer el número de workers necesarios para garantizar la QoS en el momento de mayor carga de trabajo. En primer lugar se ha probado con un único worker, pero no ha sido suficiente para soportar dicho momento, por ello, se ha añadido un worker más y en esta ocasión, la QoS si era respetada para petición del cliente. Añadir que la implementación fibonacci\_tr siempre da problemas con la QoS, ya que esta está establecida para la implemtación fibonacci.

## Arquitectura software para cubrir los tres escenarios simultáneamente

La arquitectura necesaria para cubrir los tres casos simultáneamente sería la utilizada para el escenario 3, porque este escenario es el que genera mayor carga de trabajo para el servidor, por tanto, si se consigue garantizar la QoS para este escenario, para los dos anteriores también queda garantizada. El único aspecto que podría variar sería el número de workers necesario, ya que se pasaría a tener tres clientes realizando peticiones en vez de uno, siguiendo la solución realizada para el escenario 3, habría que establecer el número de workers necesario para soportar el momento con mayor carga de trabajo.

# Validación Experimental

Para validar la solución de cada escenario se han utilizado las máquinas de laboratorio 1.02. Se han tomado 10 mediciones de la ejecución con la implemetnación fibonacci del tiempo necesario para resolver la operación a realizar y el tiempo total que conlleva la petición. Para la implementación fibonacci\_tr se han tomado también algunas medidas (no consecutivas), pero teniendo más relevancia la implementación anterior.

# Conclusiones

# Bibliografía