



# Escalabilidad en grandes plataformas

(o cómo servir 16.000.000.000 de páginas al mes sin morir en el intento)

# ¿Qué es la escalabilidad?

## **Escalar**

Cambiar de tamaño manteniendo la proporción

## **Escalabilidad**

El poder aumentar de forma significativa la carga de un sistema sin degradar la calidad del servicio

# Algunos datos sobre

16.000.000.000 páginas vistas / mes

2.000.000 páginas / día en tuenti móvil

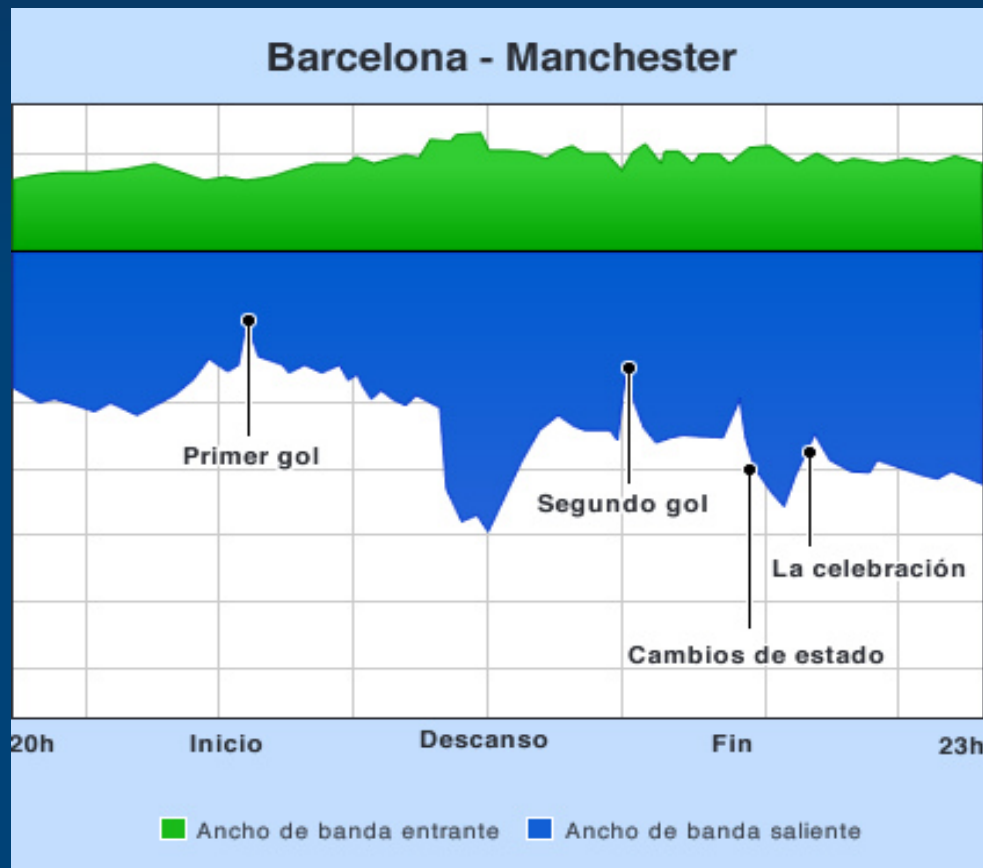
1.900.000 fotos / día

1.700.000 plays YouTube / día

1 hora / día



# ¿Qué aspecto tiene el enemigo?



# ¿Qué implica la escalabilidad?

Herramientas de calidad

Saber utilizarlas

Una arquitectura de datos sólida

Ingeniería de software

Tener una visión exacta e inmediata de lo que está pasando

# Herramientas

Conocer las herramientas que usamos en profundidad

Estar al tanto de alternativas

Pruebas, pruebas, pruebas

Usar software libre



# 100% software libre



LIGHTTPD  
fly light.



# MyWebApp v1.0

Uno o pocos servidores para toda la aplicación

Estructura monolítica

Cuello de botella: lectura de base de datos



# Memcached

Cambiamos accesos pesados a BBDD por accesos ligeros a caché

Reducción brutal de carga en BBDD

Una página típica tiene 0 - 5 queries, y 100+ accesos a Memcached

Cuello de botella: igual al anterior

# Replicación

Arquitectura maestro / esclavo

División de lecturas y escrituras

Configuraciones optimizadas

Cuello de botella: volumen de datos



# Separación de datos

División del esquema en bloques lógicos

Un cluster por cada tipo de dato

Más complicado de gestionar en código

Cuello de botella: sobrecarga en un tipo de datos

# Particionado de datos

Dividir una sola tabla en trozos y ubicarlos en varios clusters

Escalabilidad casi sin límites

No es gratis - aumenta la cantidad de código de soporte

Cuello de botella: volumen de datos

# Archivado de datos

Almacenar datos que se usan con menor frecuencia en soportes alternativos

Mantiene el tamaño de las tablas principales dentro de límites manejables

Flexibilidad a la hora de escoger el soporte alternativo

# Particionado de Memcached

El particionado no se tiene que limitar a bases de datos

Memcacheds y frontales particionados

Mejora el rendimiento de red interna

Esquema de particionado cruzado entre memcached y base de datos

# Client-side routing

Utilizar lógica en el navegador para acceder a granjas de frontales

Descargamos de trabajo a los servidores

Quitamos un posible punto único de fallo



# Content Delivery Networks

Contenidos estáticos

Optimizados para escalas masivas

Todo tiene un precio - escoger bien las optimizaciones



# Técnicas de desarrollo

Evitar queries pesadas usando consistencia eventual

Consistencia de caché en tiempo real

Caché paginada

Desnormalización de modelo de datos

Ser pragmático



# Rendimiento

Código y proceso enfocado al buen rendimiento desde el principio

El ahorro en ciclos de CPU se traduce directamente a un número de máquinas y a un número de euros

Es importante distinguir las optimizaciones que valen la pena

# Métricas

Imprescindible para tener una imagen global de lo que está pasando

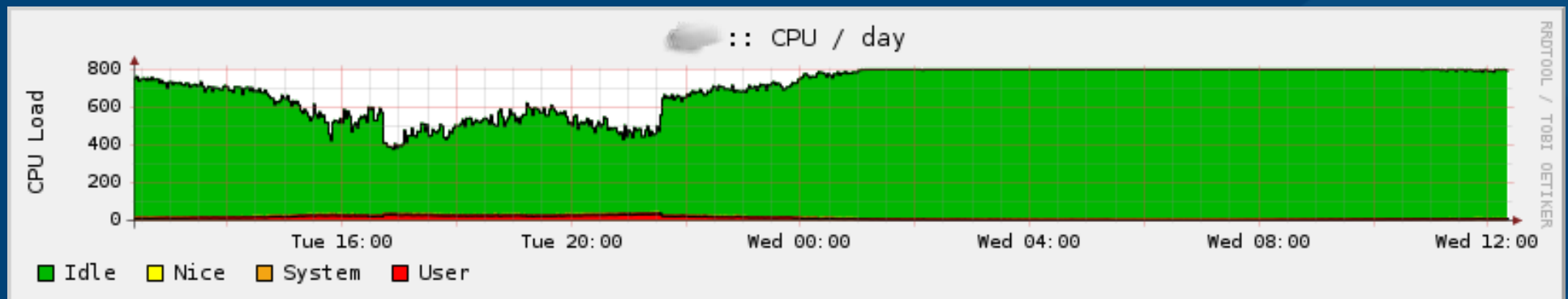
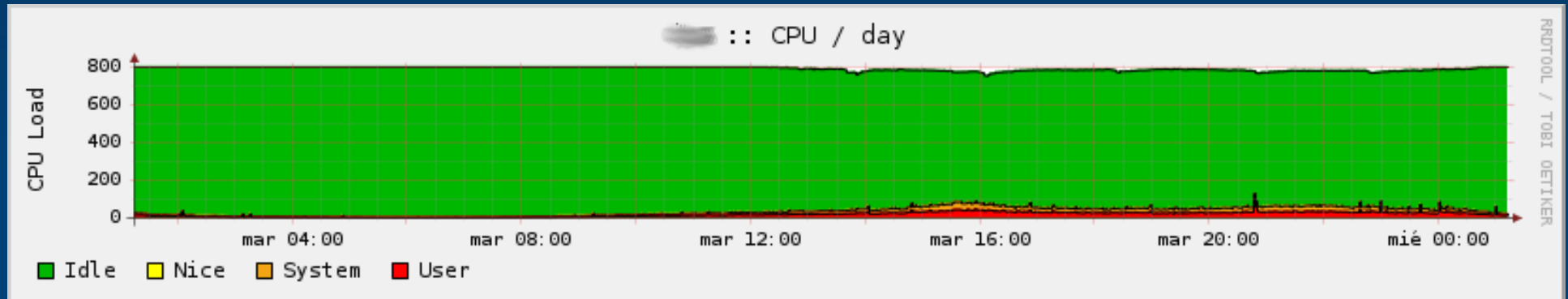
Cuanto más máquinas haya, más difícil es buscar problemas y cuellos de botella a mano

Todo se puede cuantificar, resumir, procesar...

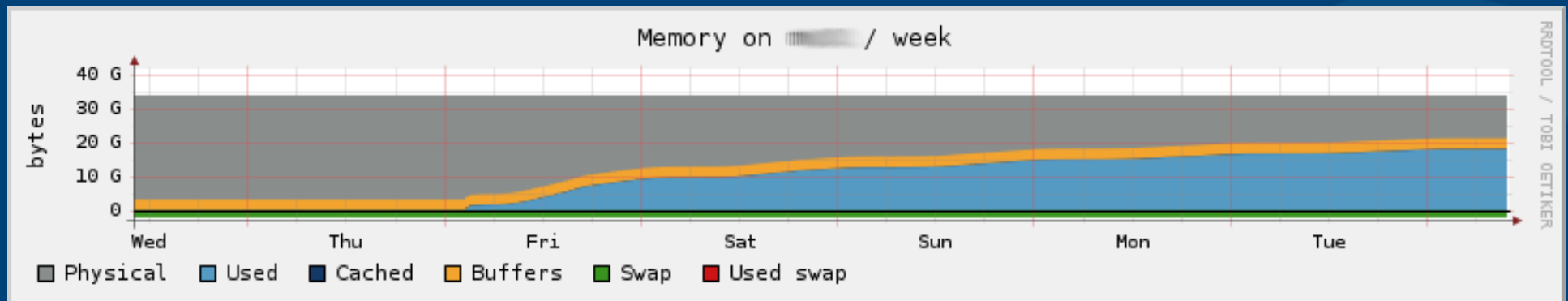
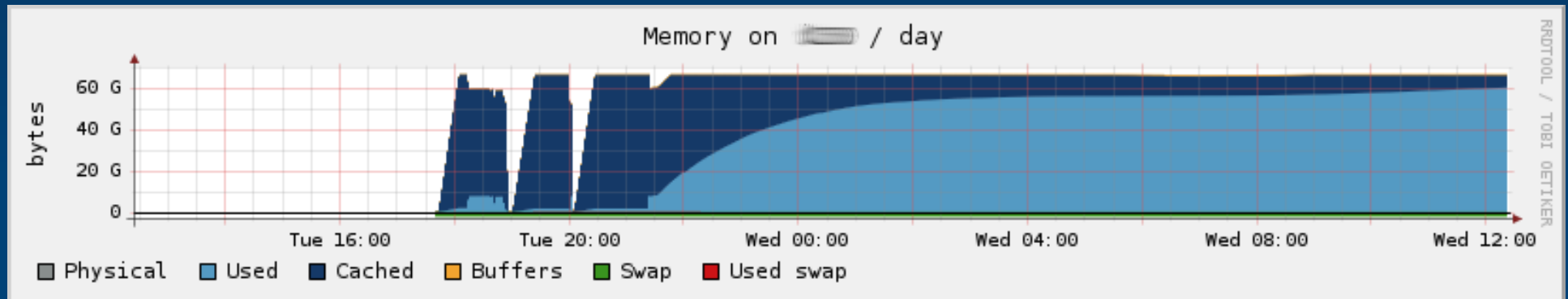
Visualmente se entiende mejor



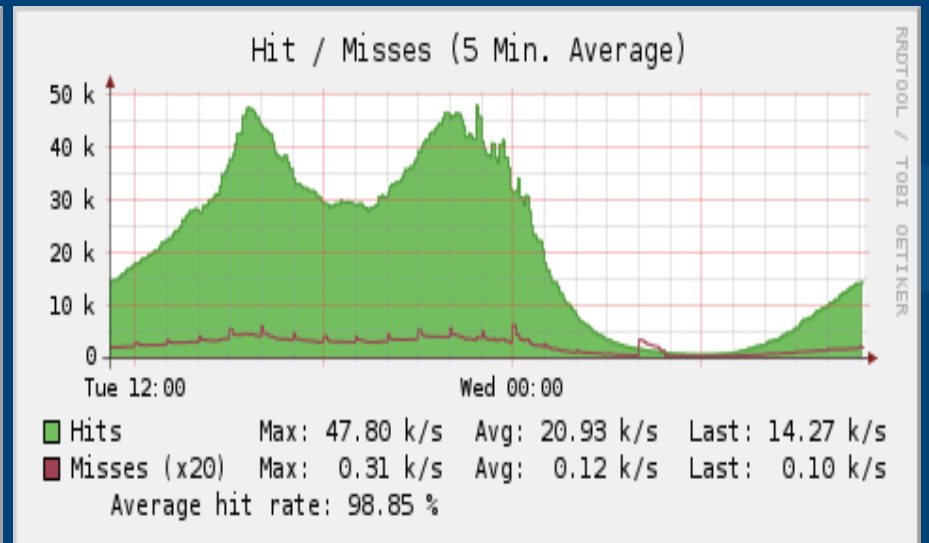
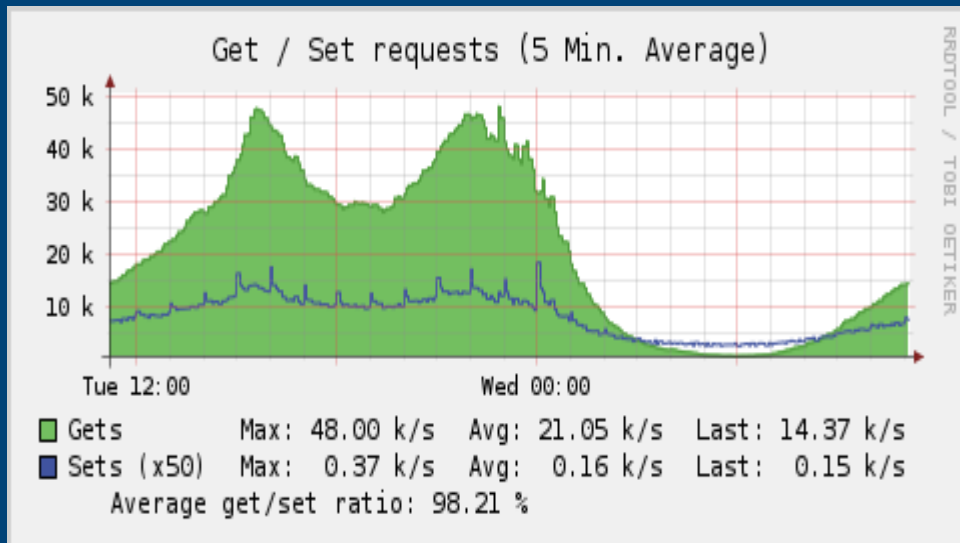
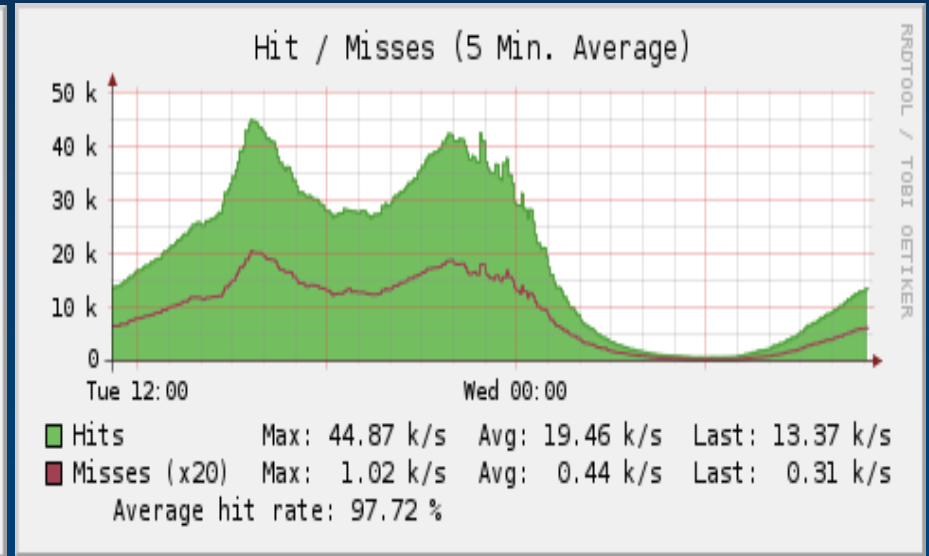
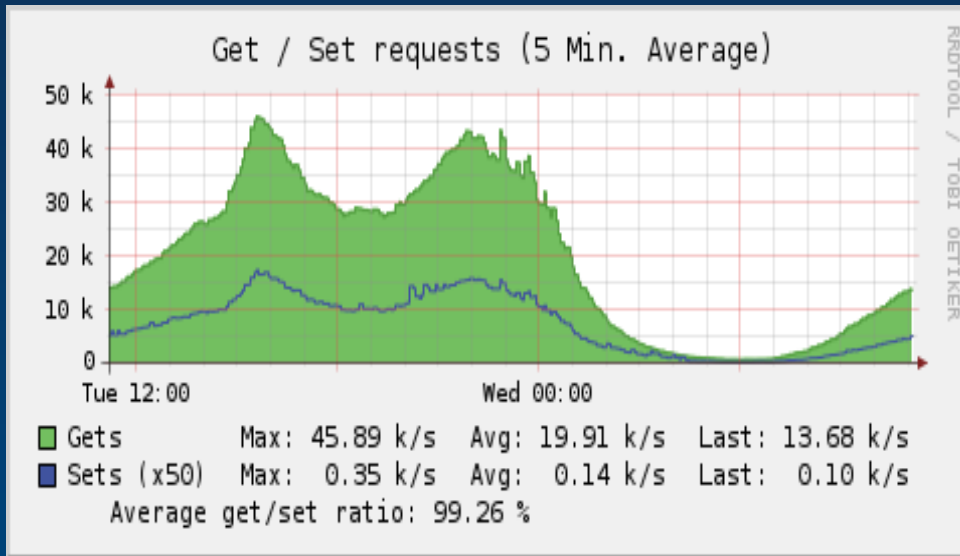
# CPU en BBDD



# Consumo de RAM



# Memcached



**Si esto te ha parecido interesante...**



<http://talento.tuenti.com>