Tema 1 – Caso de estudio: Wikipedia

Tecnología de los Sistemas de Información en la Red

Objetivos

- Presentar un caso de estudio que ilustre la necesidad de algunas tecnologías en los sistemas de información en la red.
- Analizar cuáles son los objetivos de estas tecnologías.
- Mostrar la arquitectura de algún sistema distribuido escalable.
- Conocer la evolución de ese sistema escalable.

Índice

1. Introducción

- Objetivo de TSR
  - Describir y utilizar tecnologías actuales de los sistemas de información en la red
    - Sistema de información en la red → Sistema distribuido (SD)
- Para este fin...
  - Necesitaremos algún caso de estudio que ilustre los problemas a resolver en un SD.
  - Tecnologías diferentes resuelven problemas diferentes.
    - Pero no hay relaciones uno a uno entre tecnologías y problemas.

1. Introducción

- Candidatos a casos de estudio:
  - Aquellos sistemas cuya envergadura plantee tantos problemas y retos como sea posible.
  - Ejemplos:
    - Servicio de búsqueda de Google. Utiliza más de un millón de servidores.
    - Facebook. Con más de 1000 millones de usuarios.
    - El sitio web oficial de los ferrocarriles de China. Más de 1000 millones de peticiones por día (en enero de 2012).
    - YouTube. Más de 1000 millones de usuarios y más de 7000 millones de peticiones por día.
  - Pero también necesitamos información pública fiable sobre su arquitectura y tecnologías...
    - ...y esa información no suele estar disponible (es confidencial en muchos casos).
  - La Wikipedia es una buena opción.

Índice

2. La Wikipedia hoy en día

- La Wikipedia es...
  - una enciclopedia digital creada en 2001,
  - escrita de manera colaborativa por editores voluntarios,
  - disponible en múltiples idiomas (actualmente, 288),
  - administrada por la Fundación Wikimedia
    - con solo 200 empleados!
- ...y tiene...
  - cerca de 5 millones de artículos en su edición inglesa,
    - 35 millones considerando todos los idiomas
  - 25 millones de usuarios registrados,
  - 73000 editores activos,
  - 500 millones de visitantes mensuales

2. La Wikipedia hoy en día

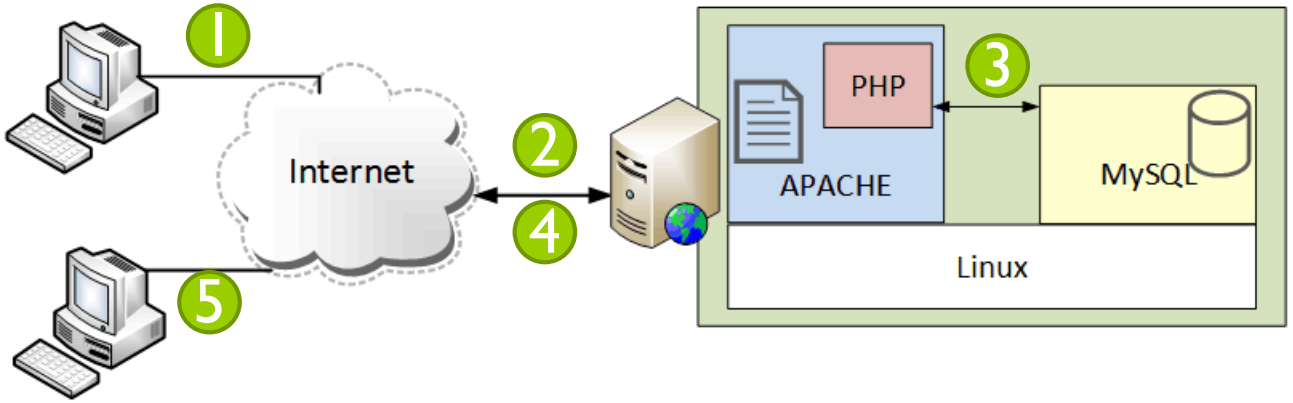
- Por tanto, la Wikipedia es un buen ejemplo de servicio distribuido de gran envergadura:
  - Basado en tecnología "wiki":
    - Edición colaborativa de contenido
    - Mantiene un histórico de las modificaciones aplicadas sobre cada página
    - Citando la entrada "wiki" de la Wikipedia...
      - *"**Un wiki permite que se escriban artículos colectivamente (co-autoría) por medio de un lenguaje de wikitexto editado mediante un navegador.** Una página wiki singular es llamada «página wiki», mientras que el conjunto de páginas (normalmente interconectadas mediante hipervínculos) es «el wiki». **Es mucho más sencillo y fácil de usar que una base de datos.**"*

Índice

Sistemas LAMP

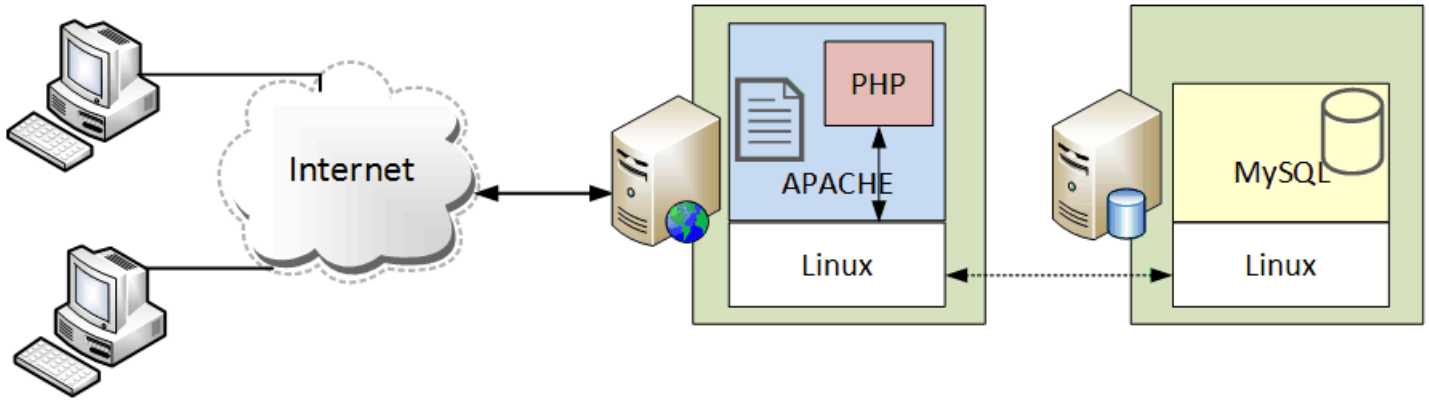
- El motor de la Wikipedia se llama MediaWiki.
- MediaWiki es un sistema LAMP.
  - LAMP = Linux + Apache + MySQL + PHP
    - Linux es un ejemplo de sistema operativo UNIX.
    - Apache es un servidor de web.
    - MySQL es un sistema gestor de bases de datos (SGBD) relacional.
    - PHP es un lenguaje de *scripting*.
  - Un sistema LAMP suele seguir una arquitectura de tres niveles
    - El nivel de **interfaz de usuario** se implanta mediante los documentos de hipertexto retornados por el servidor de web Apache y mostrados por el navegador del cliente.
    - El nivel de **aplicación** consta de algunas reglas de aplicación y está implantado en PHP.
    - El nivel de **datos** mantiene los datos persistentes y está soportado por el SGBD MySQL.
    - Todos los niveles se despliegan sobre nodos Linux.

Sistemas LAMP



- Una petición es atendida siguiendo estos pasos:
  1. Un cliente envía la petición al servidor Apache.
  2. Apache redirige esa petición a su módulo PHP interno.
  3. El programa PHP envía múltiples solicitudes al servidor MySQL.
  4. El servidor Apache construye un documento web con los resultados de esas solicitudes y se lo envía al cliente.
  5. El cliente muestra la página resultante en su navegador.

Sistemas LAMP



- Obsérvese que Apache y MySQL son dos procesos independientes:
  - Pueden ser desplegados en dos ordenadores distintos para mejorar su rendimiento.
  - El servidor Apache actúa como cliente de MySQL.

Índice

4. MediaWiki

- Tratemos de aproximar la carga máxima a soportar por MediaWiki en su implantación de la Wikipedia...
  - En picos de carga, se alcanzan 25000 accesos/s
  - Esos accesos multiplicados por el tamaño promedio de una página, proporcionará el ancho de banda máximo a facilitar.
    - Pero... ¿cuál es ese tamaño promedio de página?
      - No hay datos oficiales sobre ello.
      - Una página pequeña, como la entrada "yottabyte", ocupa 1210 KB.
      - Una página grande, como la entrada "United States", ocupa 5490 KB.
        - ¡Pero no son los tamaños mínimo y máximo!!
    - Esto implica que el ancho de banda promedio necesario durante un pico de carga estará entre 242 y 1098 Gbps.
      - 242 Gbps = 25000*1210*8 / 1000000 [acc/s * tamaño página* bits/byte / KB por GB]
      - 1098 Gbps = 25000*5490*8 / 1000000

4. MediaWiki

- ¿Podríamos soportar este ancho de banda con un solo canal de red y un solo ordenador?
  - No. Actualmente no parece posible.
  - El mayor ancho de banda facilitado por un proveedor de red comercial es 2000 Mbps (XFINITY en EEUU, diciembre 2017).
  - ¡En el peor caso necesitaríamos 549 canales de esa capacidad!
    - 1098000 / 2000 = 549
    - ¡Por tanto, parece que se necesiten más de 549 servidores para soportar esa carga!!

4. MediaWiki

- Se necesitan múltiples servidores en el sistema MediaWiki
  - Objetivo: Mejorar el rendimiento
  - Solución: Replicación
    - Será analizada en el Tema 5
    - ¡Pero se necesitan muchas réplicas (>549)!!! Surgen muchos problemas adicionales en esos casos...
      - ¿En cuántos centros de datos serán desplegados esos nodos?
      - ¿Cuántos puntos de entrada habrá para acceder a ellos?
      - ¿Cómo se propaga el tráfico entrante al resto de nodos servidores?
      - Si un usuario modifica una página de la wiki... ¿cómo y cuándo propagamos esos cambios al resto de réplicas?
        - ¡Habrá que tener cuidado! Esto compromete seriamente la consistencia y las prestaciones.
      - ¿Qué pasa si un nodo servidor falla mientras estaba procesando una petición?
      - Con tantas réplicas, ¿cuál es la probabilidad de que ninguna de ellas falle durante cierto intervalo?

4. MediaWiki

- Esbozaremos en las páginas siguientes cómo los componentes de un sistema LAMP podrán manejar cargas elevadas.
- Esos componentes son:
  - Servidor Apache
  - Servidor MySQL

4.1. Servidor Apache

- El servidor Apache espera peticiones HTTP de los clientes.
- Esas peticiones pueden ser:
  - Estáticas. Atendidas leyendo y retornando un documento desde un fichero.
  - Dinámicas. Atendidas mediante la ejecución de un programa PHP.
- El rendimiento en las peticiones estáticas puede mejorar mediante...
  - Más memoria en los servidores, ubicando en RAM todos los documentos.
    - ¡Demasiado caro!!
  - Uso de cachés.
  - Uso de *proxies* inversos.
    - Es un tipo de caché en la parte servidora.
    - Un conjunto de procesos intermedios (que actúan como servidores para los clientes) mantienen el contenido de las páginas más solicitadas.
      - Esos procesos se indexan de alguna manera; así las peticiones de los clientes se dirigen al proxy inverso adecuado.
    - **¡Cuidado!!** Cuando se añaden nuevos componentes tenemos que preocuparnos por si llegan a fallar.
      - Más componentes → Menor fiabilidad

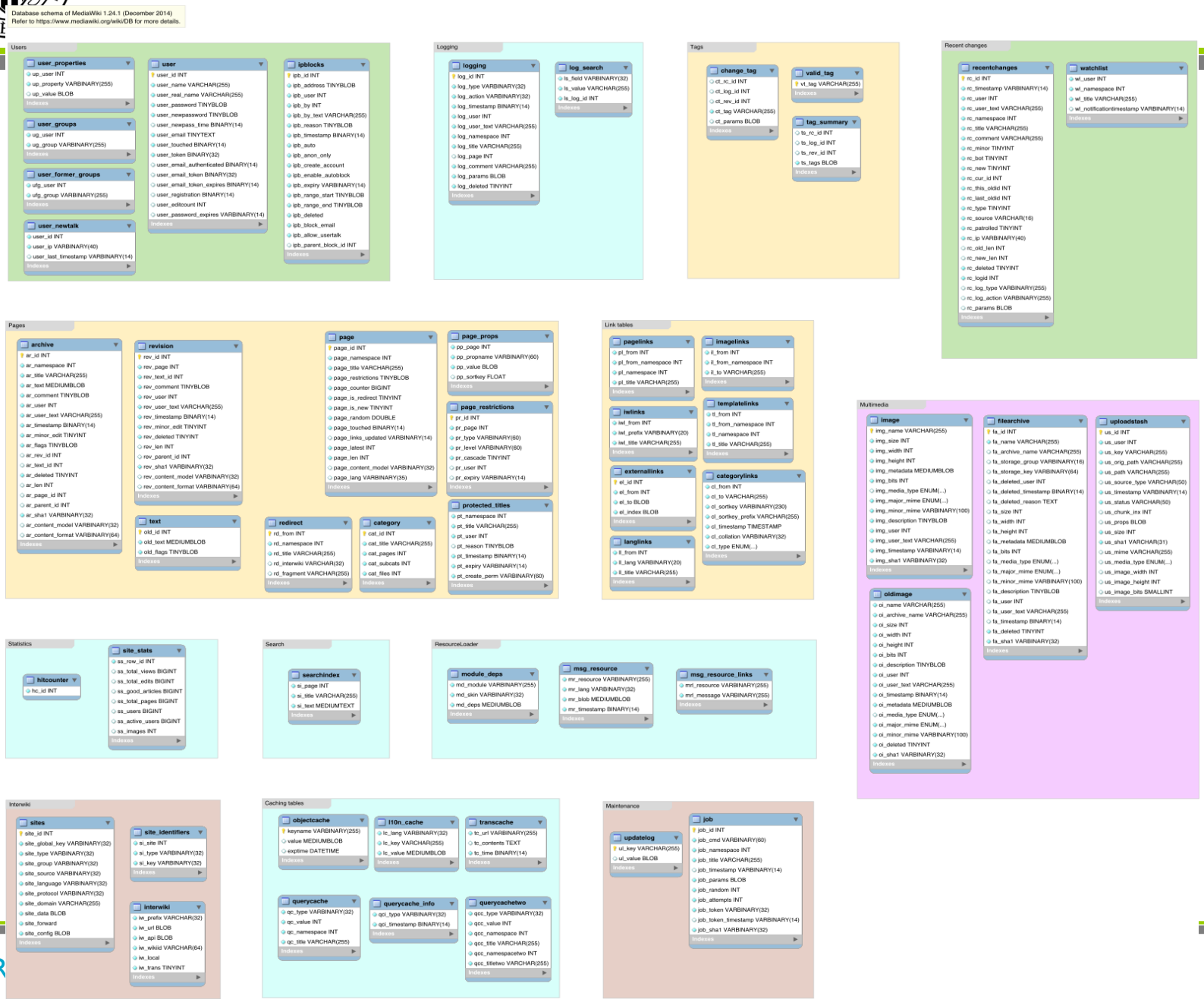
4.1. Servidor Apache

- El servicio de peticiones dinámicas depende del tipo de operación:
  - Modificación de páginas wiki:
    - Normalmente, una sola copia es modificada inicialmente.
    - Esa modificación debe propagarse a las demás réplicas y a las copias mantenidas en los *proxies* inversos.
      - Es una tarea compleja.
      - Soluciones: invalidación o propagación.
  - Páginas que dependen del usuario solicitante:
    - Algún tipo de gestión de sesiones será necesario.
    - Las peticiones en la misma sesión deben dirigirse a la misma réplica servidora.
      - Habrá problemas si falla esa réplica.
  - Consultas basadas en términos de búsqueda:
    - Pueden gestionarse mediante cachés en la parte servidora.
    - Gestión difícil si las peticiones añaden otros parámetros a la consulta.
  - Consultas de información asociada a la página:
    - Cuando una página wiki mantiene algún contenido asociado y ese contenido puede utilizarse o filtrarse en los accesos siguientes.
    - De nuevo, las cachés en la parte servidora serán útiles.

4.2. MySQL

- El servidor MySQL mantendrá los datos persistentes de Wikipedia: contenidos de sus páginas, información de usuarios registrados, enlaces entre páginas, estadísticas de uso, cambios recientes...
- En la primera generación de MediaWiki, ambos servidores (Apache y MySQL) fueron ubicados en un mismo ordenador.
- Otras arquitecturas fueron utilizadas posteriormente...
  1. Cada servidor en un nodo distinto.
  2. Se replicaron los servidores Apache. Esto exigió una distribución de las tablas MySQL entre varios ordenadores.
  3. En 2014, el esquema de MediaWiki mantenía 13 bases de datos distintas con un total de 50 tablas.
     - Cada base de datos puede ser ubicada en un ordenador diferente.
     - Aquellas bases de datos con carga alta de consultas pueden además replicarse en varios ordenadores.

4.2. MySQL (Esquema en 2014)



Índice

1. Introducción
2. La Wikipedia hoy en día
3. Sistemas LAMP
4. MediaWiki
5. Arquitectura de la Wikipedia
6. Conclusiones
7. Resultados de aprendizaje

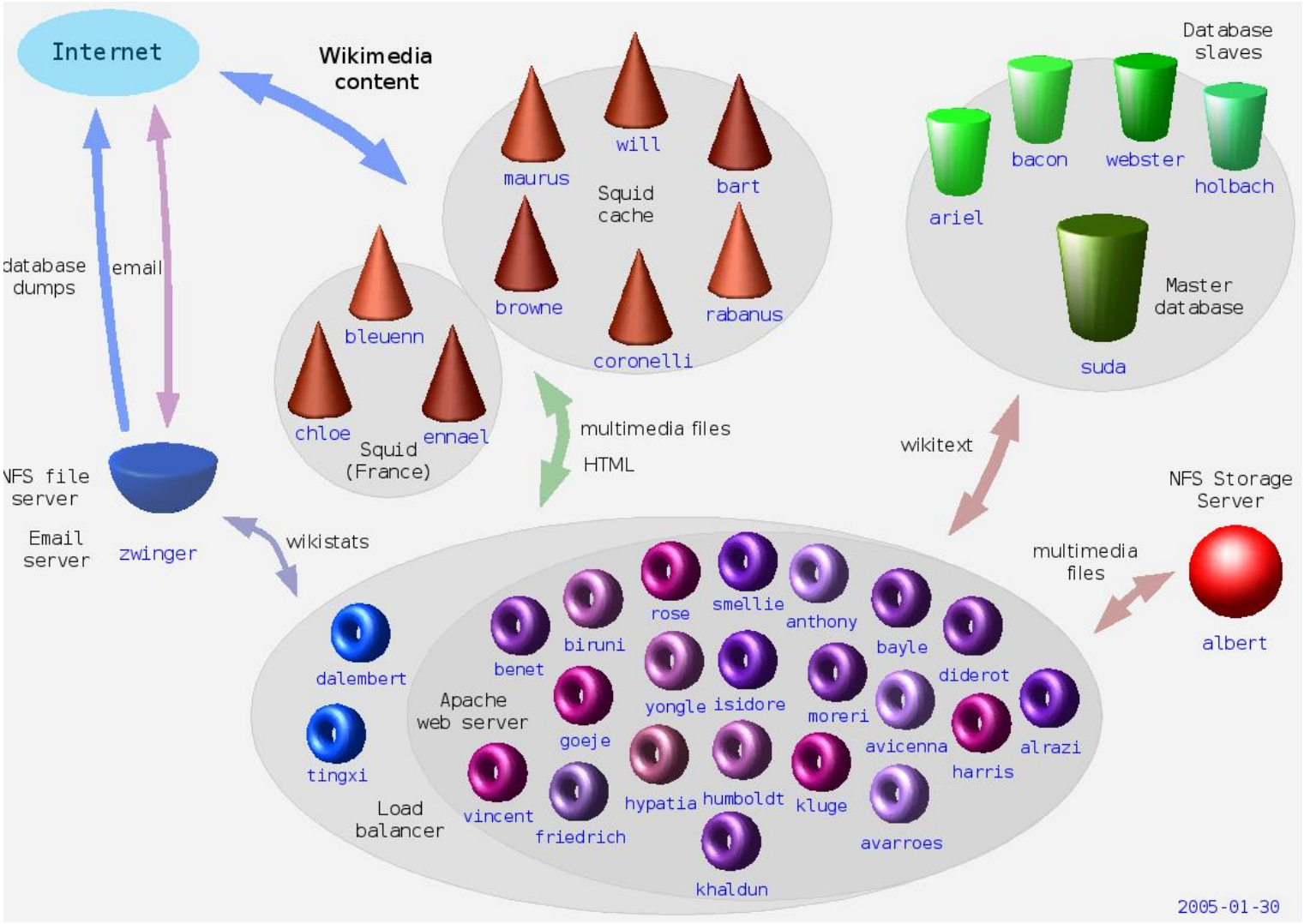
5. Arquitectura de la Wikipedia

- En sus dos primeras etapas, la arquitectura de este sistema consistía en:
  1. Un solo ordenador que mantenía todos los componentes de MediaWiki.
     - Incapaz de escalar horizontalmente.
  2. Dos ordenadores, cada uno mantiene un servidor distinto.
     - Apache + módulos PHP
     - MySQL
- El éxito de la Wikipedia forzó a la Fundación Wikimedia a diseñar arquitecturas más complejas, replicando y distribuyendo los componentes principales.
  - Se muestran tres ejemplos en las próximas hojas:
    - Arquitectura en 2005
    - Arquitectura en 2010
    - Arquitectura de componentes

5.1. Arquitectura en 2005

- Componentes:
  - Dos servicios de proxy inverso (Squid), con múltiples ordenadores en cada uno.
  - Cinco nodos en un servicio MySQL distribuido bajo un modelo de replicación pasivo.
    - La réplica primaria gestiona peticiones de lectura y escritura mientras las réplicas secundarias también aceptan lecturas y reciben las modificaciones generadas en la réplica primaria.
  - Muchos servidores Apache, con dos nodos asociados para encaminar y equilibrar la carga.
  - Los recursos multimedia se almacenan en ficheros normales mantenidos en un único servidor NFS.
  - Resumen: 39 ordenadores utilizados en este despliegue.

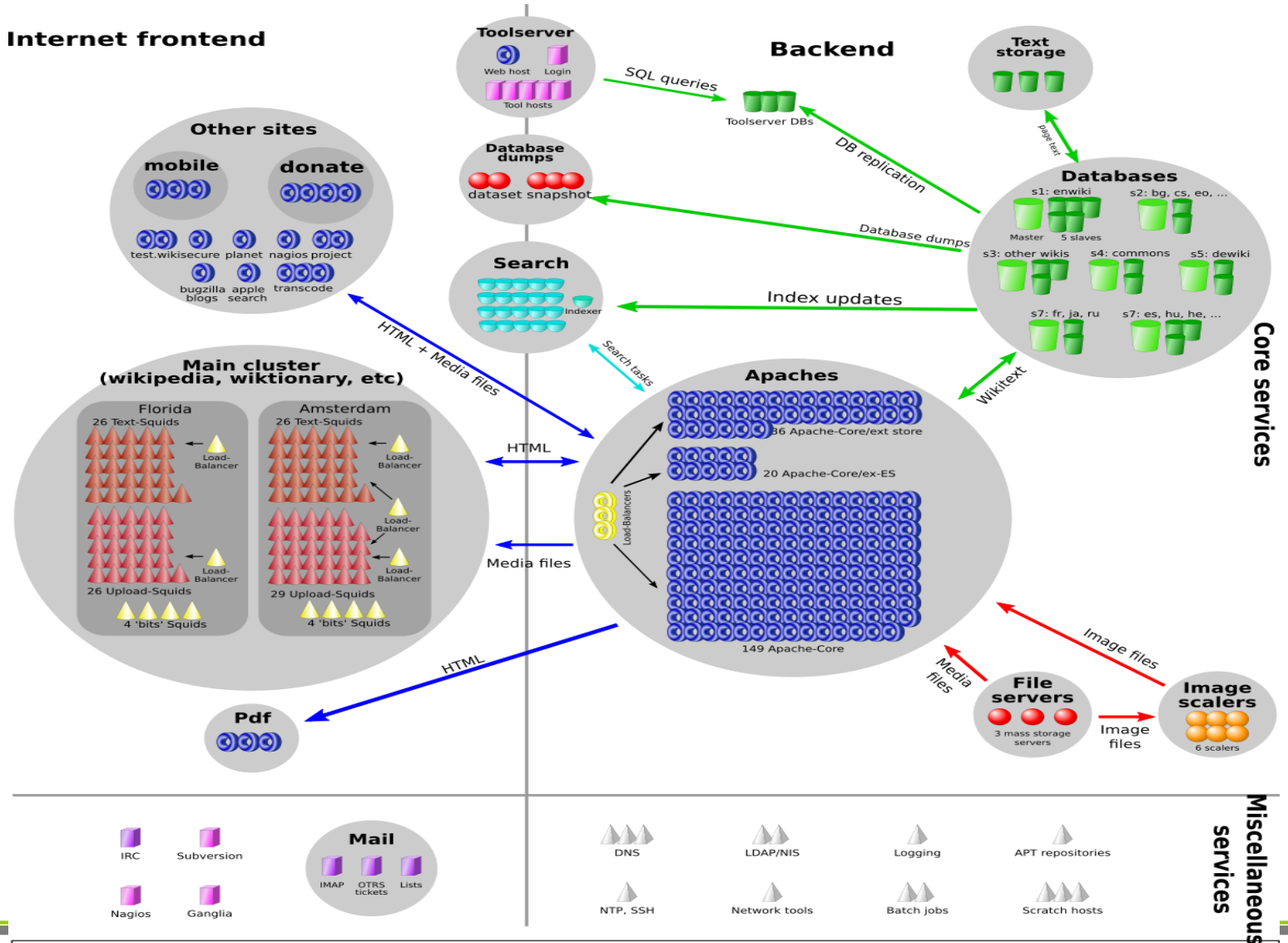
5.1. Arquitectura en 2005



5.2. Arquitectura en 2010

- El sistema necesita muchos más recursos que en 2005.
- Se han añadido varios servicios complementarios.
- Hay muchas más instancias de cada componente (proxies inversos, equilibradores de carga, servidores Apache, servidores MySQL...).
- Hay varios tipos de servidores Apache, dependiendo del tipo de recurso solicitado (páginas wiki en HTML, ficheros PDF, versiones de las páginas para dispositivos móviles, etc.).

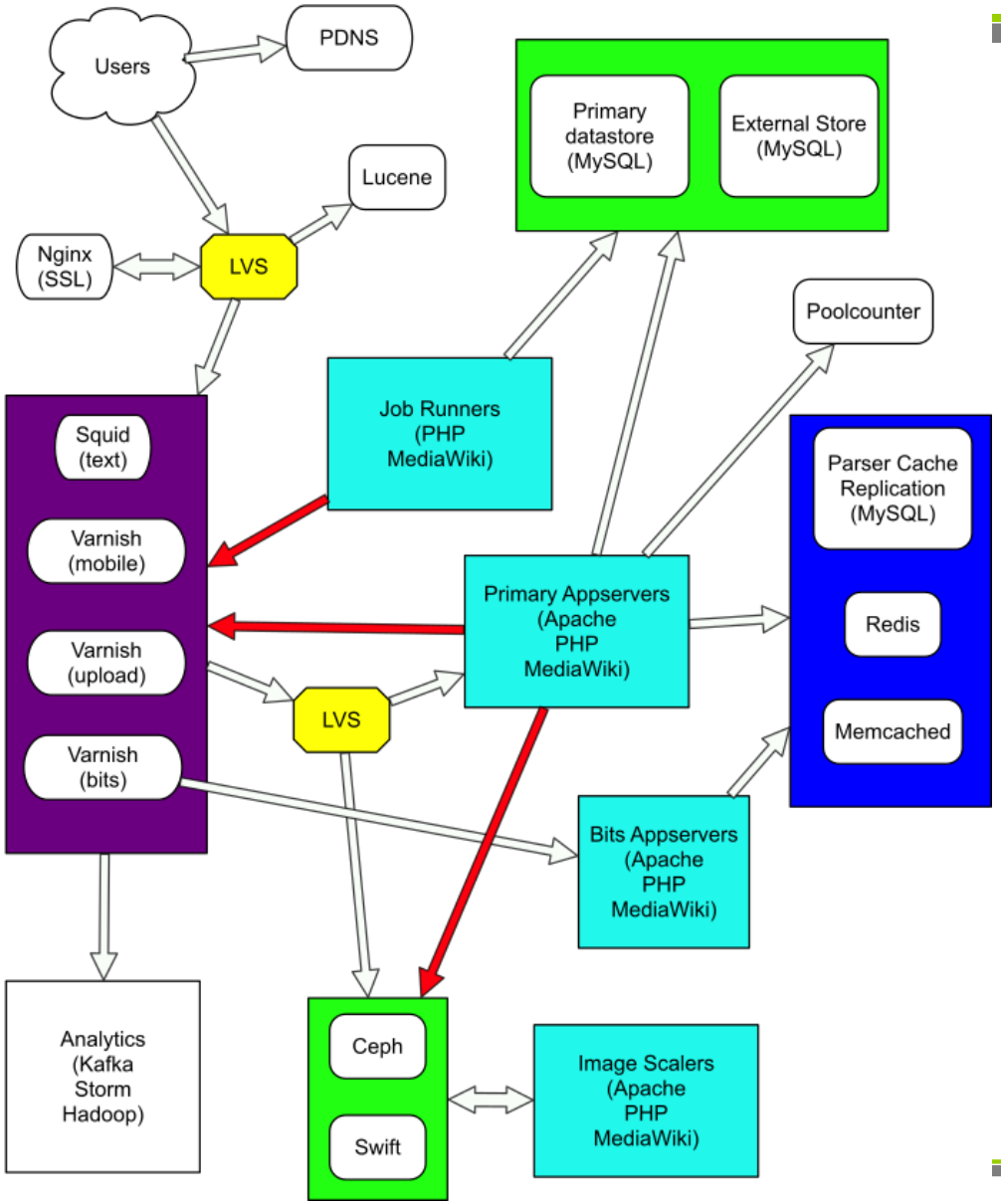
5.2. Arquitectura en 2010



5.3. Arquitectura de componentes

- En los dos ejemplos anteriores se prestaba atención a cuántos nodos se utilizaban durante el despliegue.
- En la próxima hoja nos centramos exclusivamente en las dependencias entre componentes.
  - Ésta es la arquitectura del sistema distribuido.
    - Centrada en la funcionalidad a obtener y en cómo obtenerla.
    - Más reciente → Mayor funcionalidad.
  - Los dos casos anteriores son ejemplos de despliegue.
    - Utilizaban una arquitectura más sencilla, pues el servicio proporcionado ha ido incorporando funcionalidad durante su evolución.
- Consultar la guía de estudio para entender la funcionalidad de cada componente.
  - Hay mucha información *extra* que no aparece en esta presentación

5.3. Arquitectura de componentes



Índice

7. Conclusiones

- Podemos encontrar varios ejemplos de servicios distribuidos que están siendo utilizados por millones de usuarios.
- Sus arquitecturas exigen un diseño cuidadoso que considere y resuelva varios retos.
- La carga soportada por estos servicios debe ser repartida entre tantos nodos como sea posible.
  - Así, cada nodo sólo será responsable de una parte proporcional (y reducida) de esa carga.
- Para asegurar la continuidad de servicio, los nodos servidores deben estar replicados.
  - En caso de fallo en algún nodo, otras réplicas compartirán su trabajo y completarán las peticiones de servicio ya iniciadas.
- Las cachés y los *proxies* inversos son otras técnicas que permiten mejorar la capacidad de servicio.

Índice

8. Resultados de aprendizaje

- Al finalizar el estudio de esta unidad, el alumno debería ser capaz de:
  - Identificar varios ejemplos de servicios distribuidos de gran envergadura (y, como tales, altamente escalables).
  - Identificar algunos de los problemas y retos a gestionar en esos servicios: estrategias de uso de cachés, propagación de peticiones, persistencia de la información, distribución de los datos, consistencia, fallos, continuidad de servicio...
  - Identificar algunas aproximaciones para afrontar esos retos: reparto de carga, replicación de servidores...