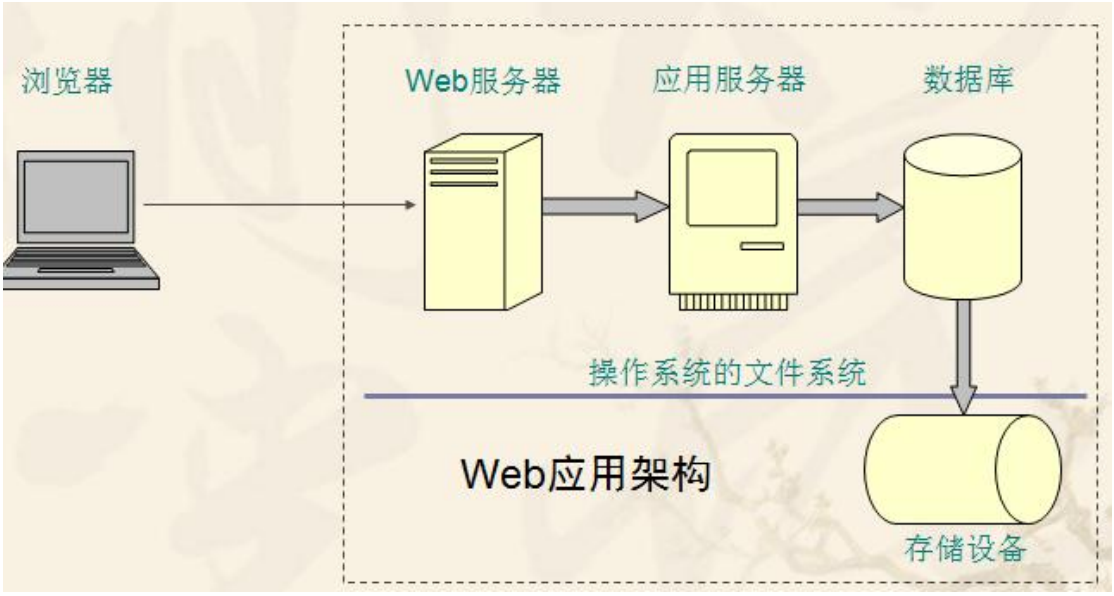
题目：缓存技术浅谈

1. 缓存是什么，解决什么问题？

- Cache是高速缓冲存储器 一种特殊的存储器子系统，其中复制了频繁使用的数据以利于快速访问
- 凡是位于速度相差较大的两种硬件/软件之间的，用于协调两者数据传输速度差异的结构，均可称之为 Cache

2.基于 Web 应用的系统架构图



3.Web 应用系统存在哪些速度差异？

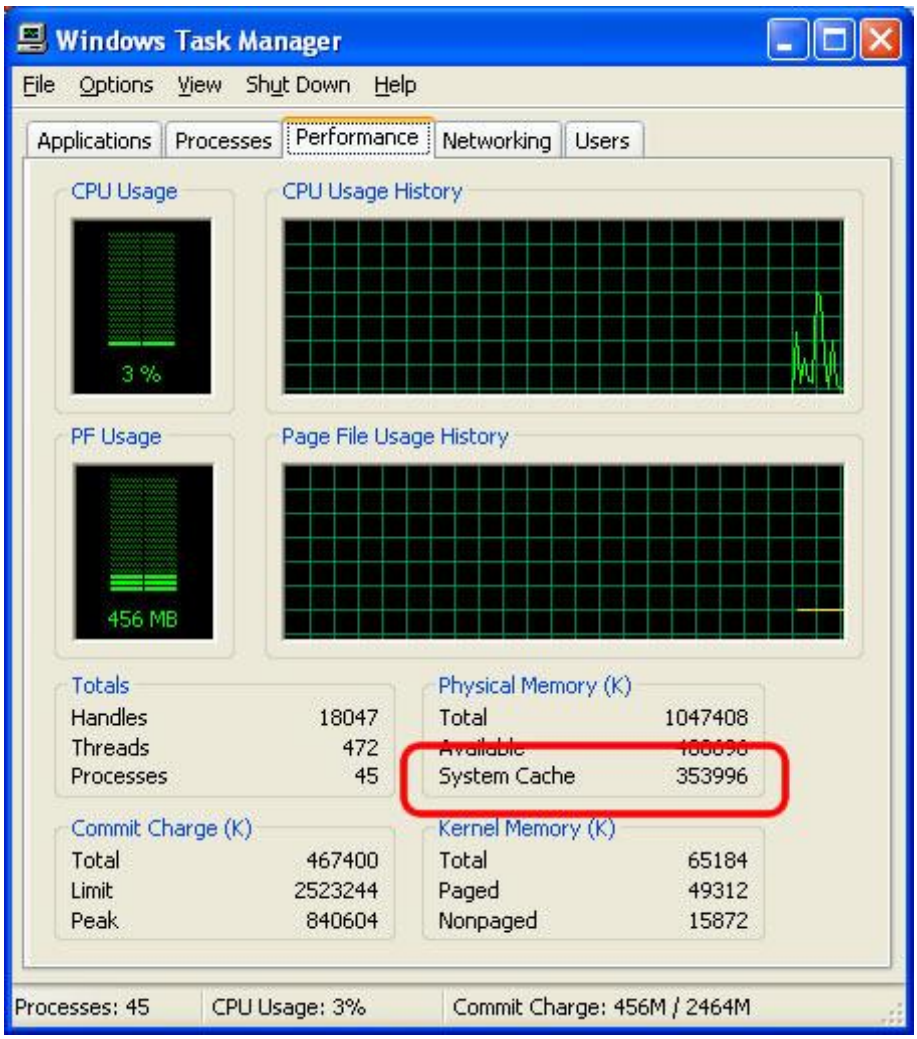
- 读取文件系统 → 读取磁盘
- 读取数据库内存 → 读取文件系统
- 读取应用内存 → 访问数据库服务器
- 读取静态文件 → 访问应用服务器
- 读取浏览器缓存 → 访问网站

4.缓存技术分类

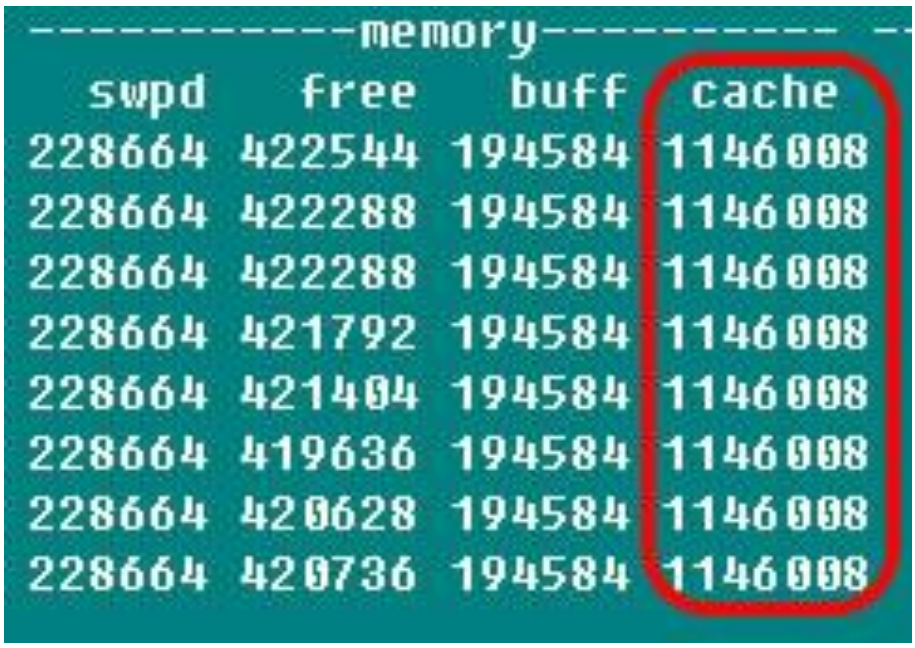
- 操作系统磁盘缓存 → 减少磁盘机械操作
- 数据库缓存 → 减少文件系统I/O
- 应用程序缓存 → 减少对数据库的查询
- Web服务器缓存 → 减少应用服务器请求
- 客户端浏览器缓存 → 减少对网站的访问

操作系统缓存概述

5.Windows 的 Disk Cache



6. Linux 的 Disk Cache



7.数据库缓存的重要性

- 为什么数据库非常依赖缓存？
  - 数据库通常是企业应用系统最核心的部分
  - 数据库保存的数据量通常非常庞大
  - 数据库查询操作通常很频繁，有时还很复杂
  - 以上原因造成数据库查询会引起非常频繁的磁盘I/O读取操作，迫使CPU挂起等待，数据库性能极度低下
- 数据库有哪些缓存策略？
  - Query Cache
  - Data Buffer

8.Query Cache

- 以SQL作为key值缓存查询结果集
- 一旦查询涉及的表记录被修改，缓存就会被自动删除
- 设置合适的Query Cache会极大提高数据库性能
- Query Cache并非越大越好，过大的Qquery Cache会浪费内存。
- MySQL: query_cache_size= 128M

9.MySQL Query Cache 监控工具

- show status like 'Qcache%';
- mysqlreport脚本
- MySQL Administrator

10. Query Cache 状态示例

```
__ Query Cache _____________________________________
Memory usage   25.77M of  64.00M  %Used:  40.26
Block Fragmnt  24.73%
Hits           6.98M     3.2/s
Inserts        100.87M    46.9/s
Insrt:Prune    34.15:1    45.5/s
Hit:Insert     0.07:1
```

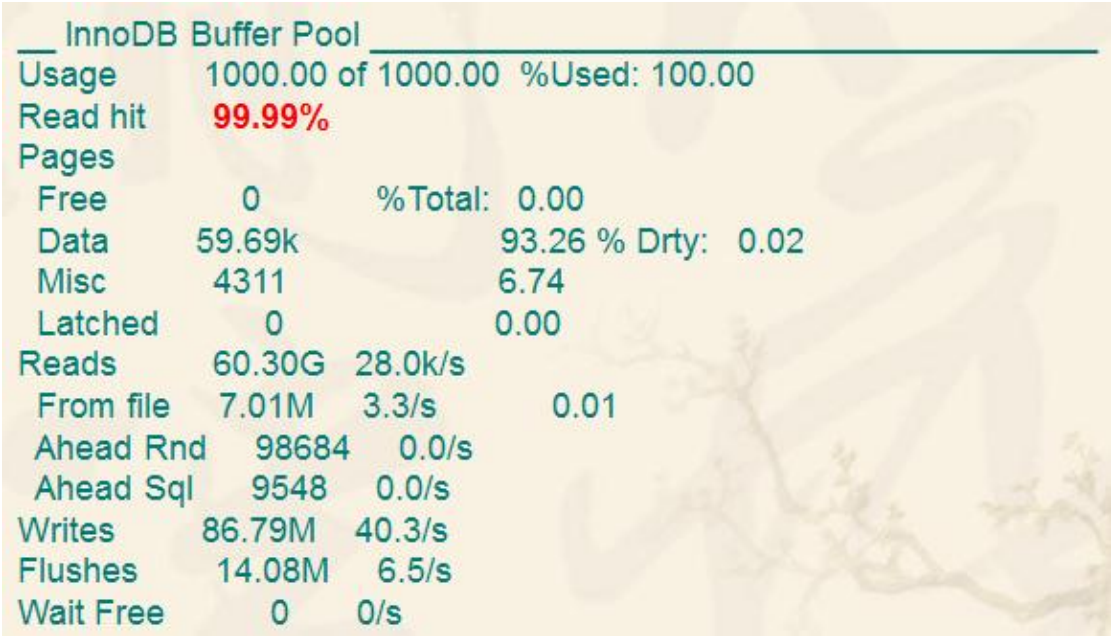
11. Data Buffer

- data buffer是数据库数据在内存中的容器
- data buffer的命中率直接决定了数据库的性能
- data buffer越大越好，多多益善
- MySQL的InnoDB buffer：
  innodb_buffer_pool_size = 2G
- MySQL建议buffer pool开大到服务器物理内存60-80%

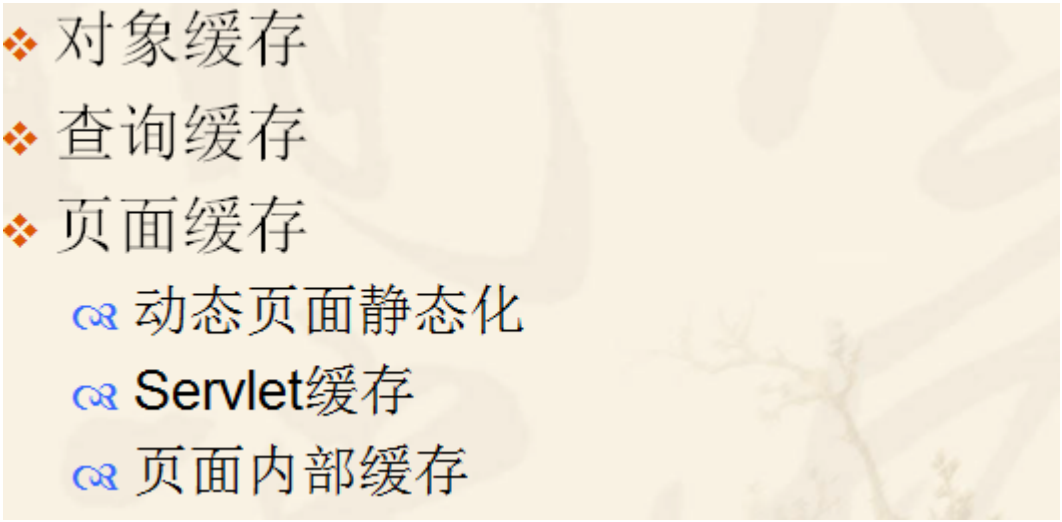
12. MySQL buffer 监控工具

- show innodb status\G
- show status like 'innodb%';
- mysqlreport脚本
- innotop

13. InnoDB buffer 状态示例



14. 应用程序缓存概述



15. 对象缓存

- 由O/R Mapping框架例如Hibernate提供，透明性访问，细颗粒度缓存数据库查询结果，无需业务代码显式编程，是最省事的缓存策略
- 当软件结构按照O/R Mapping框架的要求进行针对性设计，使用对象缓存将会极大降低Web系统对于数据库的访问请求
- 良好的设计数据库结构和利用对象缓存，能够提供极高的性能，对象缓存适合OLTP（联机事务处理）应用

16. 对象缓存分类

- 对映射数据库表记录的entity对象进行缓存
- 对1对n关系的集合进行缓存
- 对n对1关系的关联对象进行缓存

17. Hibernate 对象缓存配置

配置entity对象缓存

```
@Entity
@Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public class Forest { ... }
```

配置关联集合的缓存

```
@OneToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)
@JoinColumn(name="CUST_ID")
@Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public SortedSet<Ticket> getTickets() { return tickets; }
```

仅仅添加Annotation就可以了，无须编码，即可自动享受对象缓存。Hibernate会拦截对象的CRUD操作，针对对象读取操作进行缓存，针对对象修改操作自动清理缓存

18. Hibernate 二级缓存是提升 web 应用性能的法宝

- OLTP类型的web应用，由于应用服务器端可以进行群集水平扩展，最终的系统瓶颈总是逃不开数据库访问；
- 哪个框架能够最大限度减少数据库访问，降低数据库访问压力，哪个框架提供的性能就更高；
- 针对数据库的缓存策略：
  - 对象缓存：细颗粒度，针对表的记录级别，透明化访问，在不改变程序代码的情况下可以极大提升web应用的性能。对象缓存是ORM的制胜法宝。
  - 对象缓存的优劣取决于框架实现的水平，Hibernate是目前已知对象缓存最强大的开源ORM
  - 查询缓存：粗颗粒度，针对查询结果集，应用于数据实时化要求不高的场合

19. 查询缓存

- 对数据库查询结果集进行缓存，类似数据库的Query Cache
- 适用于一些耗时，但是时效性要求比较低的场景。查询缓存和对象缓存适用的场景不一样，是互为补充的
- 当查询结果集涉及的表记录被修改以后，需要注意清理缓存

20. Hibernate 查询缓存

- 在配置文件中打开Query Cache
  - hibernate.cache.use_query_cache true
- 在查询的时候显式编码使用Cache

```
List blogs = sess.createQuery("from Blog blog where blog.blogger
    = :blogger") .setEntity("blogger",blogger) .
setMaxResults(15) .
setCacheable(true) .
setCacheRegion("frontpages") .
list();
```

21. Hibernate 查询缓存特征

- 并非缓存整个查询结果集，而是缓存查询结果集entity对象的id集合
  - [blogId1, blogId2, blogId3,…]
  - 在遍历结果集的时候，再按照blogId去查询blog对象，例如 select blog.* from blog where id=?
  - 如果此时blog配置了对象缓存，则自动读取对象缓存
- Hibernate查询缓存会自动清理过期缓存
  - 一旦结果集涉及的entity被修改，查询缓存就被自动清理

22. 页面缓存

- 页面缓存的作用是什么？
  - 针对页面的缓存技术不但可以减轻数据库服务器压力，还可以减轻应用服务器压力
  - 好的页面缓存可以极大提高页面渲染速度
  - 页面缓存的难点在于如何清理过期的缓存
- 页面缓存技术有哪些？
  - 动态页面静态化
  - Servlet缓存
  - 页面局部缓存

23. 动态页面静态化

- 利用模板技术将访问过一次的动态页面生成静态html，同时修改页面链接，下一次请求直接访问静态链接页面
- 动态页面静态化技术的广泛应用于互联网CMS/新闻类Web应用，但也有BBS应用使用该技术，例如Discuz!
- 无法进行权限验证，无法显示个性化信息
- 可以使用AJAX请求弥补动态页面静态化的某些缺点

24. Servlet 缓存

- 针对URL访问返回的页面结果进行缓存，适用于粗粒度的页面缓存，例如新闻发布
- 可以进行权限的检查
- OScache提供了简单的Servlet缓存(通过web.xml中的配置)
- 也可以自己编程实现Servlet缓存

25. OSCache Servlet 缓存示例

```
<filter>
   <filter-name>CacheFilter</filter-name>
   <filter-class>com.opensymphony.oscache.web.filter.CacheFilter</filter-class>
   <init-param>
      <param-name>time</param-name>
      <param-value>600</param-value>
   </init-param>
   <init-param>
      <param-name>scope</param-name>
      <param-value>session</param-value>
   </init-param>
</filter>

<filter-mapping>
   <filter-name>CacheFilter</filter-name>
   <url-pattern>/news/*</url-pattern>
</filter-mapping>
```

26. 页面局部缓存

- 针对动态页面的局部片断内容进行缓存，适用于一些个性化但不经常更新的页面(例如博客)
- OSCache提供了简单的页面缓存
- 可以自行扩展JSP Tag实现页面局部缓存

27. OSCache 的页面局部缓存

```
<%@ taglib uri="http://www.opensymphony.com/oscache"  prefix="cache" %>
<cache:cache>
    ... some jsp content ...
</cache:cache>

<cache:cache key="foobar" scope="session">
    ... some jsp content ...
</cache:cache>

<cache:cache key="<%= product.getId() %>" time="1800" refresh="<%= needRefresh %>">
    ... some jsp content ...
</cache:cache>

<cache:cache key="<%= product.getId() %>" cron="0 2 * * *" refresh="<%= needRefresh %>">
    ... some jsp content ...
</cache:cache>
```

28. 应用缓存的缓存服务器

- EHCache
  - 适合充当对象缓存和Hibernate集成效果很好，Gavin King也是EHCache作者之一
- OSCache
  - 充当Servlet和页面缓存
  - 在一个Web应用当中可以同时使用OSCache和EHCache
- JBossCache
  - 在Java群集环境下使用
  - 支持缓存在节点之间的复制，在JBoss AS上被用来实现HTTP Session的内存复制功能

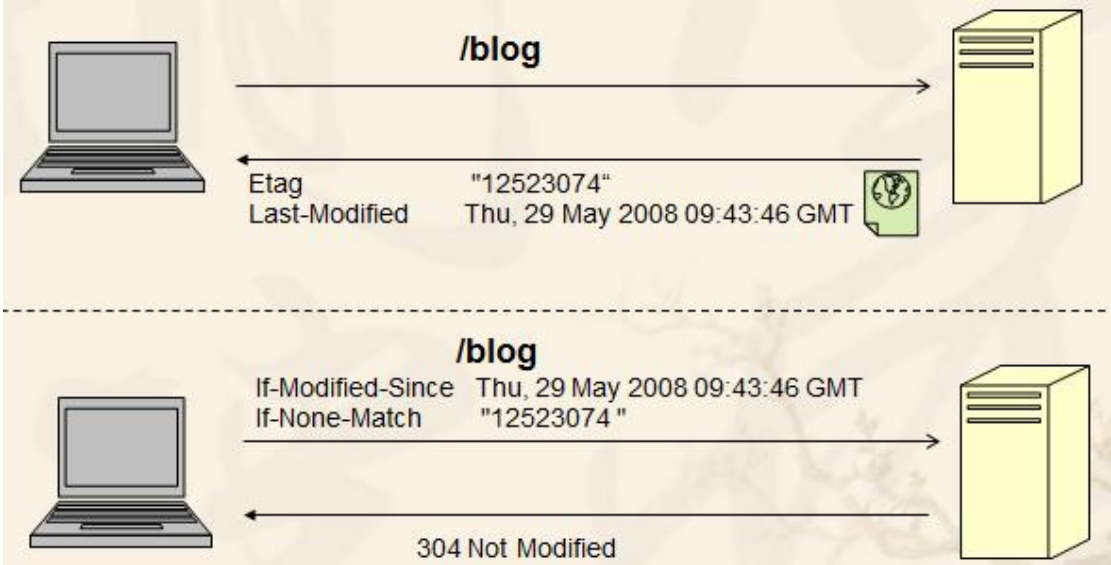
29. 非 Java 实现的通用缓存产品

- Memcached
  - 在大规模互联网应用下使用
  - 每秒支撑1.5万～2万次请求
- Tokyo Tyrant
  - 兼容memcached协议，可以持久化存储
  - 支持故障切换，对缓存服务器有高可靠性要求可以使用
  - 每秒支撑0.5万～0.8万次请求

30. 基于 AJAX 技术的浏览器缓存

- 使用AJAX调用的时候，将数据库在浏览器端缓存
- 只要不离开当前页面，不刷新当前页面，就可以直接读取缓存数据
- 只适用于使用AJAX技术的页面

31. 基于 HTTP 协议的资源缓存



32. 基于资源的缓存示例

```
<filter>
   <filter-name>CacheFilterStaticContent</filter-name>
   <filter-class>com.opensymphony.oscache.web.filter.CacheFilter</filter-class>
   <init-param>
      <param-name>expires</param-name>
      <param-value>time</param-value>
   </init-param>
</filter>

<filter-mapping>
   <filter-name>CacheFilterStaticContent</filter-name>
   <url-pattern>*.jsp</url-pattern>
</filter-mapping>
```