



OS期末习题课

主讲助教：胡永康

中国科学院大学计算机与控制学院
中国科学院计算技术研究所

2026-1-14





8.1 虚存分段分配

一台机器虚存采用分段机制，物理内存当前的空闲空间如下(按物理地址由小到大的顺序):12MB, 5MB, 18MB, 20MB, 8MB, 9MB, 10MB和15MB。

此时要为三个段分配空间(按时间先后顺序): 段A申请12MB, 段B申请10MB, 段C申请9MB。

请分别给出采用Best Fit, Worst Fit, First Fit 和 Next Fit算法下，每次分配成的空闲空间状态，以及每次分配所需的比较次数。

- 考察四种算法的概念
- 外碎片和算法时间开销是一对 trade-off
- **注意**
 - Best Fit和Worst Fit是要对链表排序的
 - 20MB分配了12MB，还剩8MB，是要留在空闲段链表里的
 - 10MB分配完了，是要从空闲段列表里删除的



8.2* 虚存页表

假设一台计算机使用32-bit的虚拟地址空间和三级页表，虚地址的划分为 8-bit | 6-bit | 6-bit | 12-bit
(注：8 bit对应为第一级页表的地址，以此类推)，请计算：

1. 该计算机系统的页大小是多少？
2. 该三级页表一共能索引多少个页？
3. 现有一个程序的代码段大小为132KB，数据段为86KB，栈大小为8KB，则在使用上述三级页表时，最少需要占用多少个物理页框？最多会占用多少个物理页框？（注：假设程序各段在地址空间中的布局可以自行决定）
4. 在上述（3）中，假设该计算机使用一级页表进行地址空间管理，则（3）中的程序需要占用多少个物理页框？

1. 4KB
2. $2^{(8+6+6)}$
3. 物理页框包括：已分配的物理页，指向这些物理页的多级页表；
 - 注意不同级页表不能放在同一页内
 - 最少： $57 + 1 + 1 + 1 = 60$
 - 最多：
 - $57 + 6 + 6 + 1 = 70$
(Assumption: 各段内存连续、分配时按4KB对齐)
4. 注意页表是用于索引虚拟地址空间，而不是索引物理地址空间
 $2^{20} * 4B / 4KB = 1024$ 个一级页表
 $1024 + 57 = 1081$



8.3 虚存 TLB

假设一台计算机上运行一个进程A，该进程的地址空间大小为8 MB（页大小为4KB）。该计算机使用线性页表记录进程A的虚实映射关系，并且将A的页表都保存在内存中。该计算机CPU的TLB大小为64项，每项4B，一次TLB查询或TLB填充的延迟均为5 ns，请计算：

- 假设该计算机使用软件处理TLB miss，且操作系统进行一次页表查询的平均延迟为100 ns，如果能让虚实地址映射的平均延迟为40 ns，那么 TLB的命中率应为多少？如果能让虚实地址映射的平均延迟不超过20 ns，那么TLB的命中率应为多少？（上述各项操作的延迟不变）

- Latency of TLB hit: 5ns
- Latency of TLB miss: 5+100+5+5
 - E.g., store a register to memory
 - Check TLB, find it miss 5ns
 - Raise a exception to find 120ns
 - Fill TLB 5ns
 - Redo the instruction of store, check TLB, hit 5ns
- 如果硬件处理呢？在页表中找到后填充到TLB里，没有异常发生，store指令只运行一次
- 计算命中率
 - $5x + 115(1-x) = 40$
 - $5x + 115(1-x) = 20$



8.4 虚存

现有如下C程序

```
uint32 X[N];
```

```
int step = M, i = 0;
```

```
for(i=0;i<N;i+=step) X[i] = X[i] + 1;
```

请计算：

- 假设该程序运行在一台计算机上，该计算机的虚址空间为32-bit，物理地址空间为2 GB，页大小为4 KB，如果采用一级页表，则该页表的页表项一共有多少？
- 假设该计算机的CPU的TLB大小为32项，每项4B，那么题述程序中的M和N取值为多少时，会使得程序中循环的每一次执行都会触发TLB miss？（假设TLB初始为空）
- 在（2）中，M和N取值多少时，会使得程序中的循环执行时TLB hit最多？（假设TLB初始为空）

- $2^{32}/2^{12}=2^{20}$ 个
- $M * 4B > 4KB$
- $M = 1$



9.1 页替换

假设一台计算机上运行的一个进程其地址空间有8个虚页（每个虚页大小为4KB，页号为1至8），操作系统给该进程分配了4个物理页框（每个页框大小为4KB），该进程对地址空间中虚页的访问顺序为 1 2 4 5 4 3 7 3 7 8 6 1。假设分配给进程的4个物理页框初始为空，请计算：

(1) 如果操作系统采用CLOCK算法管理内存，那么该进程访存时会发生多少次page fault？当进程访问完上述虚页后，物理页框中保存的是哪些虚页？

(2) 如果操作系统采用LRU算法管理内存，请再次回答（1）中的两个问题。请回答虚页保存情况时，写出LRU链的组成，标明LRU端和MRU端。

- 模拟算法，略



9.2 页替换

假设一台计算机给每个进程都分配4个物理页框，每个页框大小为1KB。现有一个程序对一个二维整数数组（`uint32 X[32][32]`）进行赋值操作，该程序的代码段占用一个固定的页框，并一直存储在内存中。程序使用剩余3个物理页框存储数据。该程序操作的数组X以列存储形式保存在磁盘上。该程序有如下两种写法。

写法1:

```
for(int i=0;i<32;i++)  
    for(int j=0;j<32;j++)  
        X[i][j] = 0
```

写法2:

```
for(int j=0;j<32;j++)  
    for(int i=0;i<32;i++)  
        X[i][j] = 0
```

请分析使用这两种写法时，各自会产生多少次page fault？（注：请写出分析或计算过程）

- 256, 8



9.3 内存页权限

假设一个程序有两个段，其中段0保存代码指令，段1保存读写的数据。段0的权限是可读可执行，段1的权限是可读可写，如下所示。该程序运行的内存系统提供的虚址空间为14-bit空间，其中低10-bit为页内偏移，高4-bit为页号。

当有如下的访存操作时，请给出每个操作的实际访存物理地址或是产生的异常类型（例如缺页异常、权限异常等）

- 读取段1中page 1的offset为3的地址
- 向段0中page 0的offset为16的地址写入
- 读取段1中page 4的offset为28的地址
- 跳转至段1中page 3的offset为32的地址

Segment 0		Segment 1	
Read/Execute		Read/Write	
Virtual Page #	Page frame #	Virtual Page #	Page frame #
0	2	0	On Disk
1	On Disk	1	14
2	11	2	9
3	5	3	6
4	On Disk	4	On Disk
5	On Disk	5	13
6	4	6	8
7	3	7	12

- 无
- 权限异常
- 缺页异常
- 权限异常



9.4* 页替换 缓存算法

假设一个程序对其地址空间中虚页的访问序列为

0,1,2, ...,511,422,0,1,2,...,511, 333,0,1,2,..., 即访问一串连续地址（页0到页511）后会随机访问一个页（页422或页333），且这个访问模式会一直重复。请分析说明：

- 假设操作系统分配给该程序的物理页框为500个，那么，LRU, Second Chance和FIFO这三种算法中哪一个会表现较好（即提供较高的命中率），或是这三种算法都表现不佳？为什么？

- 注意：正常的“缓存”命中率
 - For caches in CPU: $\sim 80\%$
 - For cache systems in web services: $60\% \sim 99\%$
 - For swap system: should be higher
- 答案：均不佳 ($<1\%$)
- 为什么？
 - Cache Size vs. Working Set Size
 - Scan workload
 - 因为扫描工作流程大小大于缓存大小
- Scan-workload resistance
 - Juncheng Yang, etc.,. FIFO queues are all you need for cache eviction. SOSP '23



10.1* buddy system

- 现有一个内存空间分配器，采用伙伴算法。假设物理内存总共 64 KB，
 - 1) 请给出第一级的一对伙伴块的起始地址
 - 2) 请给出第二级的二对伙伴块的起始地址
 - 3) 地址 0xa700，已知它位于第 7 级伙伴块中，请问该块的伙伴块的起始地址
- 0x0000; 0x8000
 - 0x0000 0x4000 0x8000 0xc000
 - 第七级块大小： $64\text{KB}/(2^7) = 512\text{B}$ ，每块0x200对齐，每对伙伴块0x400对齐
0xa600，伙伴块地址为0xa400



10.2 HDD

- 现有一块磁盘，扇区大小为512B，假设其平均寻道时间是4ms，旋转速率是15000 RPM（每分钟15000转），传输带宽是200MB/s，请计算：
- 1) 当某一用户程序分别从磁盘上的一个文件中读取256B，2KB，1MB的数据时，这三种情况下的有效带宽各是多少？（注意：用户程序发送的文件数据读写请求经过文件系统处理后，发往磁盘的最小请求粒度是4KB。计算有效带宽时不考虑软件层的时间开销）
- 2) 如果希望该用户程序读写该磁盘的有效带宽达到180MB/s，则该程序的读写粒度应为多大？
- 有效带宽 = **数据块大小** / 读取数据块总时间
- 读取数据块总时间 = 磁盘**平均**旋转延迟 + 寻道时间 + 数据传输时间
- 磁盘**平均**旋转延迟（等待目标扇区旋转到磁头下方时间）= 转一圈需要的时间 / 2（统计平均）
- 注意：**磁盘传输的最小粒度是扇区**



10.3 HDD

- 现有一块有240个磁道的磁盘，假设其磁头当前位于第103磁道，正在向磁道序号增加的方向移动。现有一个磁盘访问请求序列，其访问的磁道号依次为33, 50, 8, 69, 110, 150, 173, 202，请计算：
- 1) 当分别采用FIFO、SSF和C-SCAN三种磁盘调度算法执行上述磁盘请求序列时，三种情况下的寻道距离各是多少？
- 注意：SCAN 和 C-SCAN 算法中，扫描的首尾边界不是磁盘边界，而是请求边界
- 略



11.1* RAID

现有一个由5块磁盘组成的磁盘阵列，采用RAID-5模式，如下图所示。

该磁盘阵列每块盘的磁盘块 (block) 大小为4KB，每条 (strip) 含一个块；磁盘的平均寻道时间是5ms，旋转速度是12000 RPM（每分钟12000转），传输带宽是200MB/s，请计算：

1) 平均来说，从该RAID5阵列上读出一个条带 (stripe) 的时间是多少？

2) 当向该RAID5阵列中写入连续的两个4KB数据块时，平均来说，所需的时间是多少？请考虑这两个数据块属于同一个条带和不同条带的两种情况。

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

- 第一问
 - 寻道时间 5ms
 - 平均旋转延迟 $60s/12000/2 = 2.5ms$
 - 传输时间 $4KB / 200MB/s = 0.02 ms$
 - 总时间 7.52ms
- 第二问
 - 读 -> 算 -> 写
 - 同一条带：并发读一个条带，并发写一个条带 $2 * 7.52ms$
 - 不同条带：并发读两个连续条带，并发写两个连续条带 $2 * 7.54ms$

每块磁盘需要在一次 IO 中处理 2 个块，如7末尾+P2开头，P1末尾+8开头



11.2* SSD

现有一块320GB的SLC SSD，它的擦写上限(P/E cycles)是200,000 次。假设SSD FTL能将写均匀分布在所有的闪存页上，若以每秒发 300,000 个4KB写请求的速率写，请问多长时间这块SSD 会被磨穿？

- 页数 $320\text{G}/4\text{KB} = 80 * 10^6$
- $80 * 10^6 * 200000 / 300000 \text{ s}$
 $= 5.3 * 10^7 \text{ s}$
 $= 14814.8 \text{ h}$
 $= 617 \text{ d}$
- 请注意
 - 仅对于RAM、和部分文件系统
 $\text{GB} = 2^{30} \text{ Byte}$
 - 对于SSD、HDD、网络和其他的场景
 $\text{GB} = 10^9 \text{ Byte}$
 $\text{GiB} = 2^{30} \text{ Byte}$
 - P/E cycle 是指一轮 擦+写



11.3 SSD

现有一块SSD，每个擦除块有256页，且它读一页的延迟是20微秒，写一页的延迟是60微秒，擦除一块的时间是1毫秒。如果该SSD的FTL采用混合映射，分下面3种情况，计算回收一个块需要的时间。

- Switch merge
- Partial merge 且假设块中有效页为 50%
- Full merge

注意：SSD 只支持 Plane 间并行，**Block 内的读、写、擦除均为串行**

- Switch: 1ms
- Partial: $256 * 0.5 * 80\mu s + 1ms = 11.240ms$
- Full merge: $256 * 80\mu s + 2ms = 22.48ms$



12.1 FS

现有一个文件系统，它的文件块索引采用多级间址。该文件系统的inode，包含12个直接指针，1个一级间址指针，1个二级间址指针和1个三级间址指针。假设文件块大小为4KB，每个文件块对应的磁盘块地址为4B。

- 请问该索引结构能够索引的最大文件是多大？
- 请问一个 1GB 的文件需要几级间址？它总共有多少间址块？其中，各级间址块分别是多少？如何找到第5,000块？

- 1)
 - 一级间址：指向一个全是直接指针的块
 - N 级间址：指向一个全是 N-1 级间址的块
 - 最大文件
 $(12 + 1024 + 1024^2 + 1024^3) * 4KB$
约为 4TB
- 2)
 - 使用顺序：直接->一级->二级->三级
 - 1GB 需要二级间址
 - $(10^6/4 - 12) / 1024$
 - 256个1级，1个二级（实际上是255+1）
 - 全局的第5000块数据块，在二级间址下的第5000 - 12 - 1024块



12.2* FS

某用户X刚挂载了一个文件系统（假设此时该文件系统的所有inode已被加载到内存），该文件系统使用的磁盘块大小为4KB，能用到的page cache大小最大为256MB。随后，该用户执行如下所示程序A。请分析（请写出分析过程）

注：假设（1）所有目录都只需1个磁盘块保存其内容；（2）fs01.ppt和fs02.ppt两个文件已在文件系统中存在。

- 1) 当程序A打开fs02.ppt文件时，文件系统需要从磁盘读取几个磁盘块？
- 3) 程序A执行完成后，用户Y再次运行该程序，当程序A打开fs02.ppt时，文件系统需要从磁盘读取几个磁盘块？
- 4) 用户Y将程序A中打开的文件修改为/home/os25/fs01.ppt，并编译执行程序A，那么当程序A打开fs01.ppt时，文件系统需要从磁盘读取几个磁盘块？

注意：打开文件/打开目录不读取磁盘块（inode已在内存）
但traverse树，需要读每层目录项，一层一个数据块（假设目录足够小）

目录也是”文件“，存目录项的数据块也被page cache缓存

- 1) 根目录、/home/、/home/os25 共三个
- 3) 0个
- 4) 0个

```
#define MAX (1024)
char buf[MAX];
int fd = open("/home/os24/fs02.ppt", O_CREAT | O_RDWR, 0666);
int n = 0, i = 0;
if (fd < 0)
{
    perror("open");
    exit(-1);
}
for (i = 0; i < MAX; i++)
{
    bzero(buf, sizeof(buf));
    sprintf(buf, "%3d\n", i);
    n = write(fd, buf, strlen(buf));
    printf("len=%d\n", strlen(buf));
    if (n != strlen(buf))
    {
        perror("write");
        printf("length=%d, buf=[%s]", strlen(buf), buf);
    }
}
close(fd);
```



12.2* FS

某用户X刚挂载了一个文件系统（假设此时该文件系统的所有inode已被加载到内存），该文件系统使用的磁盘块大小为4KB，能用到的page cache大小最大为256MB。随后，该用户执行如下所示程序A。请分析（请写出分析过程）

注：假设（1）所有目录都只需1个磁盘块保存其内容；（2）fs01.ppt和fs02.ppt两个文件已在文件系统中存在。

2) 假设该文件系统采用write through的缓存策略，当程序A完成对fs02.ppt的写入操作后，文件系统写入几个磁盘块？如果该文件系统采用的是write back缓存策略，那么程序A在写完fs02.ppt还未关闭文件时，文件系统写入几个磁盘块？

注意： `sprintf(buf, "%3d\n", i);` //长度为四/五个字节

Write 4B 1000次，Write 5B 24次，共4120B

2) write through 写了1025（有一个写请求跨页）块，占用2块write back 0块

（考虑inode的更新，也可以是1025+1024）

```
#define MAX (1024)
char buf[MAX];
int fd = open("/home/os24/fs02.ppt", O_CREAT | O_RDWR, 0666);
int n = 0, i = 0;
if (fd < 0)
{
    perror("open");
    exit(-1);
}
for (i = 0; i < MAX; i++)
{
    bzero(buf, sizeof(buf));
    sprintf(buf, "%3d\n", i);
    n = write(fd, buf, strlen(buf));
    printf("len=%d\n", strlen(buf));
    if (n != strlen(buf))
    {
        perror("write");
        printf("length=%d, buf=[%s]", strlen(buf), buf);
    }
}
close(fd);
```



13.1 FS

- 现有一个文件系统，在其使用文件缓存的情况下，某个应用创建了一个文件
“/home/OS25/fs03.pdf”，并往该文件中写入了33 KB的数据，请分析该过程需要写几个块？分别写哪几个块？如果在任意时刻发生宕机，会出现哪些不一致？请详细列出所有不一致的情况。（注：假设home和OS25目录都已存在）

FS 不一致, space leak

以课件为准的答案，**不使用日志**
用fsck保证崩溃一致性：自下而上写

- 创建过程
 - 分配元数据块：inode bitmap 块
 - 写元数据块：fs03.pdf 的 inode 块
 - 写目录数据：os25 的数据块
 - 写目录元数据：os25 的 inode 块
- 写入过程
 - 分配数据块：data block bitmap块
 - 写数据块：fs03.pdf 的数据块 (9个)
 - 写元数据块：fs03.pdf 的 inode 块

目的：fsck可以**自上而下**地进行恢复



13.1 FS

- 现有一个文件系统，在其使用文件缓存的情况下，某个应用创建了一个文件
“/home/OS25/fs03.pdf”，并往该文件中写入了12 KB的数据，请分析该过程需要写几个块？分别写哪几个块？如果在任意时刻发生宕机，会出现哪些不一致？请详细列出所有不一致的情况。（注：假设home和OS25目录都已存在）

对应现实中的fs：ext2

用fsck保证崩溃一致性：自下而上写

- 创建过程

- 分配元数据块：inode bitmap 块
- 写元数据块：fs03.pdf 的 inode 块
- 写目录数据：os25 的数据块
- 写目录元数据：os25 的 inode 块

- 写入过程

- 分配数据块：data block bitmap块
- 写数据块：fs03.pdf 的数据块（3个）
- 写元数据块：fs03.pdf 的 inode 块

目的：fsck可以自上而下地进行恢复



13.2 FS 日志

某个文件系统在磁盘上保存了一个大小为20 KB的文件 A，现有一个进程打开文件 A，并调用write函数一次性向文件 A 的文件块0 和文件块1写入新数据。假设该文件系统使用文件缓存，且宕机可能发生在任意时刻。请分析

- 1) 如果文件系统采用数据日志，宕机恢复后，文件A的内容是什么？请分不同情况讨论(即在什么样的宕机情况下，文件A的内容是什么)；
- 2) 如果文件系统采用元数据日志，并且采用先改数据再改元数据的方式，宕机恢复后，文件 A 的内容是什么？请分不同情况讨论(即在什么样的宕机情况下，文件 A 的内容是什么)。

- 对应现实中的fs: ext4
用WAL保证崩溃一致性：每个“操作”都是原子的
- 数据日志（对应 ext4 data=journal）的写流程
 - 写 TxB、写日志（数据块 0、数据块 1、inode、bitmap）、写 TxE
 - Checkpoint(实际执行)
 - 清除日志
- 元数据日志（对应 ext4 data=ordered）的写流程：
 - 写数据块 0、写数据块 1
 - 写 TxB、写日志（inode、bitmap）、写 TxE
 - Checkpoint
 - 清除日志



13.2 FS 日志

进程调用write函数overwrite两个文件块

解答：

1. 文件 A 的内容，或为 write 之前的内容，或为 write 之后的内容
 - 写入 TxE 前，任意时刻宕机视作未发生修改，于是得到旧内容
 - 写入 TxE 后，任意时刻宕机都会从 TxB 开始重新执行，最终得到新内容
2. 先写数据块，再写元数据。文件A的内容有四种情况：
 - 写块 0 前宕机，文件内容和 inode 均未改变
 - 写块 1 前宕机，文件内容部分改变，inode 未改变
 - 写 TxE 前宕机，文件内容全改变，inode 未改变
 - 写 TxE 后宕机，文件内容全改变，inode 改变

因此：元数据日志不能保证原子的overwrite



13.3 FS LFS

LFS 的imap和CR都采用类似数组的结构，下标是ino或imap块号，每一项保存对应i-node或imap块的磁盘地址。例如，imap[k]记录ino为k的i-node的磁盘地址；CR[n]记录第n个imap块的磁盘地址。假设一个LFS的块大小为 4KB，磁盘地址占4B。如果已经分配了100万个i-node，请问：

- 1) 该LFS的imap有多少个块？请给出计算过程；
- 2) 该LFS的CR有多少个块？请给出计算过程；
- 3) 如何查ino=356302的inode的磁盘地址？请给出查找和计算过程。

1. imap 块数 = inode 数 * 磁盘地址长度 / 块大小
2. CR 块数 = $2 * \text{imap 块} * \text{磁盘地址长度} / \text{块大小}$
3. 查找流程：CR -> imap -> inode
 - CR : $356302 / (1024 * 1024) = 0$
(CR 索引 1M 个 inode)
 - imap : $(356302 / 1024) \bmod 1024 = 347$ (imap 索引 1K 个 inode)
 - inode : $356302 - 1024 * 347 = 974$



13.4 FS

一个LFS的块大小为4KB，segment大小是4MB。文件块采用多级索引，即包含10个直接指针，以及一、二、三级间接指针各1个。每个指向数据块的指针占4字节。该LFS中已经有一个12MB的文件foo，请分析：

- 1) 给出文件foo的文件块索引结构，即文件foo使用了哪些指针？
- 2) 在该LFS中写文件foo的第2560块(假设它在磁盘块 A_i 中， A_i 为磁盘逻辑块号)，需要写哪些块？需要几次I/O？请给出它们写在磁盘上的顺序；
- 3) 如果是Fast FS (其块大小也为4KB)，写文件foo的第2560块，需要写哪些块？需要几次I/O？
- 4) 如果是日志文件系统，只记录元数据日志，且日志不采用批量提交，则写文件foo的第2560块，需要写哪些块？需要几次I/O？

- LFS：重写数据块、**间接索引块**和inode块、imap块
- FFS：自下而上直接修改数据块和inode块
- 元数据日志：直接修改数据块，先写日志再写inode块
- 目标是12MB文件的第2560块，所以不是追加写而是覆盖写
- **只有LFS会改变数据块位置**，所以也只有LFS需要写索引块



13.4 FS

一个LFS的块大小为4KB，segment大小是4MB。文件块采用多级索引，即包含10个直接指针，以及一、二、三级间接指针各1个。每个指向数据块的指针占4字节。该LFS中已经有一个10MB的文件foo，请分析：

- 1) 给出文件foo的文件块索引结构，即文件foo使用了哪些指针？
- 2) 在该LFS中写文件foo的第2560块(假设它在磁盘块 A_i 中， A_i 为磁盘逻辑块号)，需要写哪些块？需要几次I/O？请给出它们写在磁盘上的顺序；
- 3) 如果是Fast FS (其块大小也为4KB)，写文件foo的第2560块，需要写哪些块？需要几次I/O？
- 4) 如果是日志文件系统，只记录元数据日志，且日志不采用批量提交，则写文件foo的第2560块，需要写哪些块？需要几次I/O？

(1) $12\text{MB} / 4\text{KB} = 3072$ 块

10 个直接指针 \rightarrow 10 个

一个一级间址 $\rightarrow 2^{10}$ 个

一个二级间址 $\rightarrow 2^{20}$ 个

一个三级间址 $\rightarrow 2^{30}$ 个

(2) 5 个块、1 次写 (可认为 CR 一直被缓存，且异步写回)

写：数据块、一级间址、二级间址、inode、imap
(一次顺序写)

(3) 2个块、2 次写 (不考虑找 inode 位置的过程)

写：数据块、inode

(4) 六个块、5(6)次写 (不考虑找 inode 位置的过程)

写：数据块、写日志 (TxB块、inode日志块)、写TxE块、inode、清除日志



其他注意事项

- 考试范围
 - 以上传的所有课件和实例分析题为准
 - 选择题期中后80%，期中前20%
 - 大题期中前后内容大约各占一半
- 实例分析部分
 - 上传的PPT内容不保证100%正确
 - 重功能理解，轻实现细节



谢谢大家

Q&A

