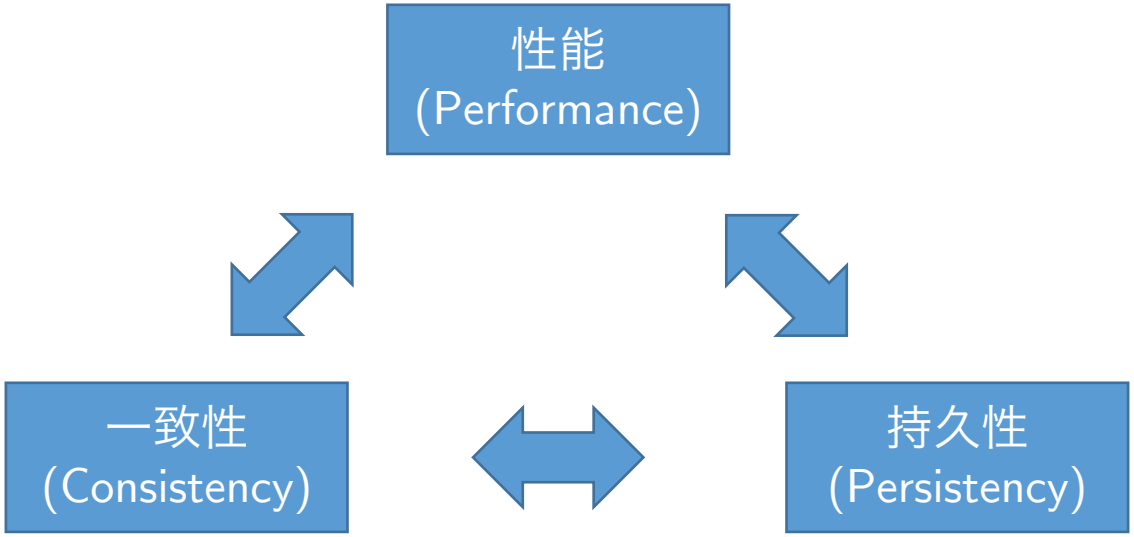
持久数据的可靠性

蒋炎岩

南京大学 | 计算机软件研究所 | 系统与软件分析研究组

复习：崩溃一致性

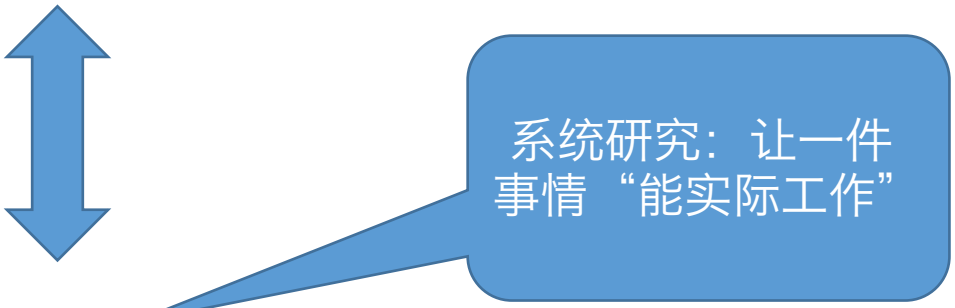
- 在系统能够崩溃的时候保证数据结构的正确性
  - 体现了性能、一致性、持久性之间的“权衡” (trade-off)
  - 有多种实现方法(FSCK, Journaling, CoW, ...)



持久数据可靠性

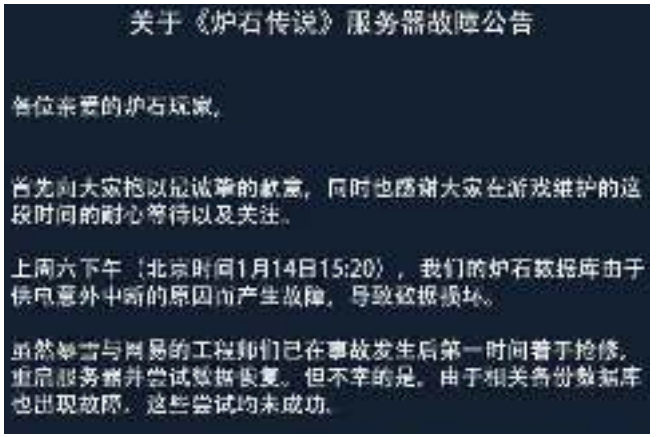
计算机系统：实际 = 更大的挑战

- OJ题(overly simplified)
  - 输入格式、数据规模等都是固定好的
  - 一旦有点不对，就不是自己的责任



- 实际系统(非常复杂)
  - 所有的部件(软件/硬件)都不绝对可靠(断电只是小问题)
  - 黑客虎视眈眈
  - ……

持久数据可靠性：数据是宝贵的


关于《炉石传说》服务器故障公告

各位亲爱的炉石玩家，

首先向大家抱以最诚挚的歉意，同时也感谢大家在游戏维护的这段时间的耐心等待以及关注。

上周六下午（北京时间1月14日15:20），我们的炉石数据库由于供电意外中断的原因而产生故障，导致数据损坏。

虽然暴雪与网易的工程师们已在事故发生后第一时间着手抢修，重启服务器并尝试数据恢复。但不幸的是，由于相关备份数据库也出现故障，这些尝试均未成功。

- Persistency对我们来说意味着保障
  - 虽然你用的软件都说“我们完全不负责”

- 但实际上很多数据丢不起
  - 《炉石传说》数据丢失——今晚没法玩了
  - 手机数据损坏无法开机——今天没法过了
  - 电脑硬盘坏了——这几周没法过了，OS作业没法交了
  - 教务系统崩坏——这几年没法过了，重修所有课程
  - 银行/支付宝弄丢了你的账号——破产了，没法活了

持久数据可靠性：世界是残酷的

- 计算机本身不可靠
  - 软件有bug (crash, Kernel Panic)
  - 硬件会崩溃 (系统异常, 断电, ...)
  - 存储器也不可靠
    - 宇宙射线击中内存
    - 磁盘不仅可能断开(断电等)，还可能出现各种问题

- 即便计算机不可靠，这些系统还得造啊！

文件系统：可靠性

- 应用运行在复杂的计算机系统上
  - 磁盘 – 文件系统 – 数据库 – 应用程序
  - 没有一个是绝对可靠的

- 可靠性存在于各个层级
  - 磁盘
  - 文件系统

  } 本次关注

  - 数据库
  - 应用程序

Redundant Arrays of Inexpensive Disks (RAIDs)

用不可靠的磁盘构造更可靠的磁盘

- 你的磁盘每天都有可能会*坏掉*
  - 坏掉就是数据好像从世界上消失了
  - 你还敢用这个硬盘做{教务系统，支付宝，银行，…}吗？
- 总有一天，一定有磁盘会坏掉的
  - 而且坏事件一旦发生，就有数据会丢失
- 而且……
  - 永远不丢失数据似乎是不可能的，但我们需要降低出问题的概率
    - (三体人进攻地球……摧毁所有数据中心……)
  - 怎样用不可靠的磁盘构造更可靠的磁盘？

Redundant Arrays of Inexpensive Disks

- The term RAID was invented by David Patterson, Garth A. Gibson, and Randy Katz at the University of California, Berkeley in 1987
  - Turing Award Winner

- 想法：虚拟化
  - 进程抽象：把一个CPU虚拟成多个CPU
  - 虚存抽象：把一份内存虚拟成多个地址空间
  - RAID：把多个磁盘虚拟成一个磁盘

- RAID：可以完全透明地实现(在物理上实现冗余)

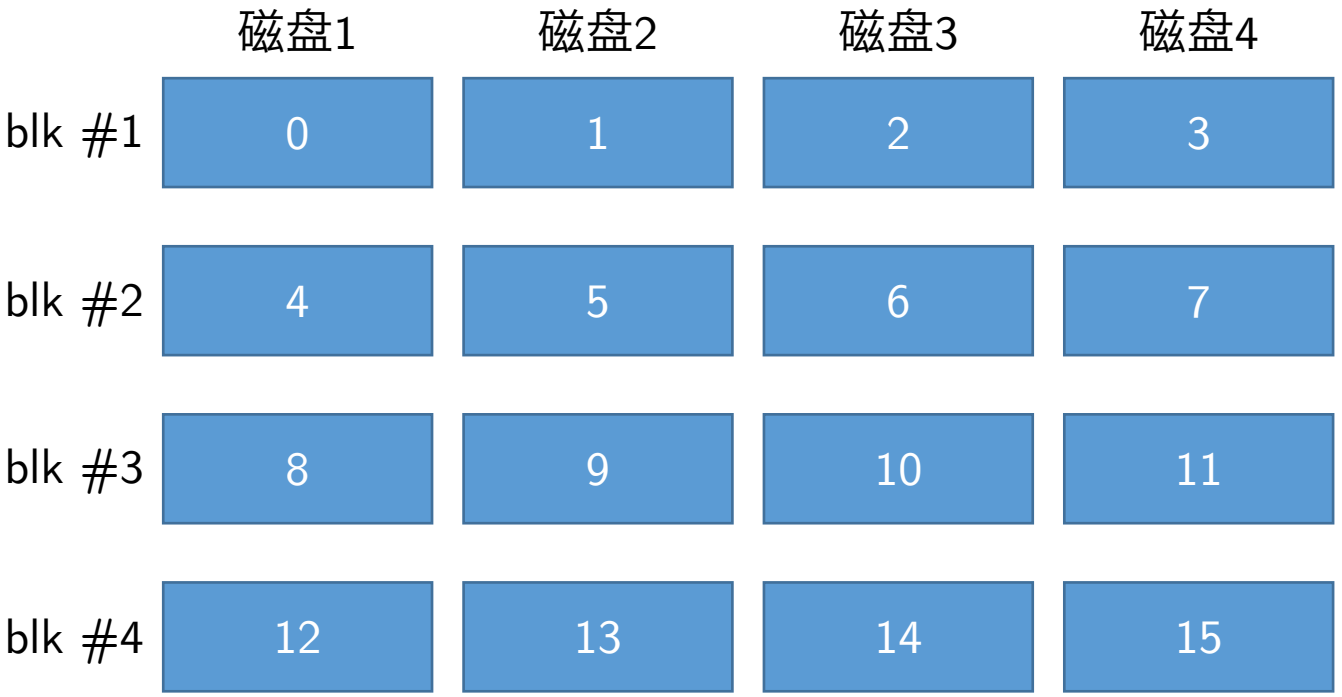
回顾：磁盘是什么？

- 一系列编号的数据块
  - `std::map<int,char[512]>`

- 虚拟化也很简单
  - 我们有多个编号是$\{0, 1, 2, 3\}$ … 的磁盘
  - 构造一个虚拟磁盘，把x映射到$\langle f(x), g(x)\rangle$
    - $f(x)$是对应的磁盘编号
    - $g(x)$是该磁盘的扇区编号

RAID-0 (0 = 没有) Redundancy

- 4 x 1TB = 4TB的磁盘，应该选用哪一种方案?
  - $f(x) = x/4,\ g(x) = x \bmod 4$
  - $f(x) = x \bmod 4,\ g(x) = x/4$



RAID-1：镜像

- 4 x 1TB = 2TB，容量减半，可靠性++++
  - $x \mapsto \{\langle f(x), g(x)\rangle, \langle f(x)+1, g(x)\rangle\}$
  - RAID1获得了多少可靠性？性能呢？

	磁盘1	磁盘2	磁盘3	磁盘4
blk #1	0	0	1	1
blk #2	2	2	3	3
blk #3	4	4	5	5
blk #4	6	6	7	7

RAID-1：崩溃一致性

- $x \mapsto \{\langle f(x), g(x)\rangle, \langle f(x)+1, g(x)\rangle\}$带来的问题
  - 每次read都可以选择*任意*一个磁盘(提高了性能)
  - 每次write必须写入两个磁盘(保证可靠性)

- 不要忘记系统可能断电……
  - 可能$\langle f(x), g(x)\rangle$更新了而 $\langle f(x)+1, g(x)\rangle$没有，怎么办?

解决办法：用电池维护journal

RAID-0, 1, 10, 01

- RAID-10 = RAID-1+0 = RAID-1 (不是ten)
- RAID-01 = RAID-0+1 (交换Disk #2, #3)

- 那么， RAID2, RAID3, RAID4, RAID5是什么……呢?

- 还有别的RAID方法么?

岔开话题

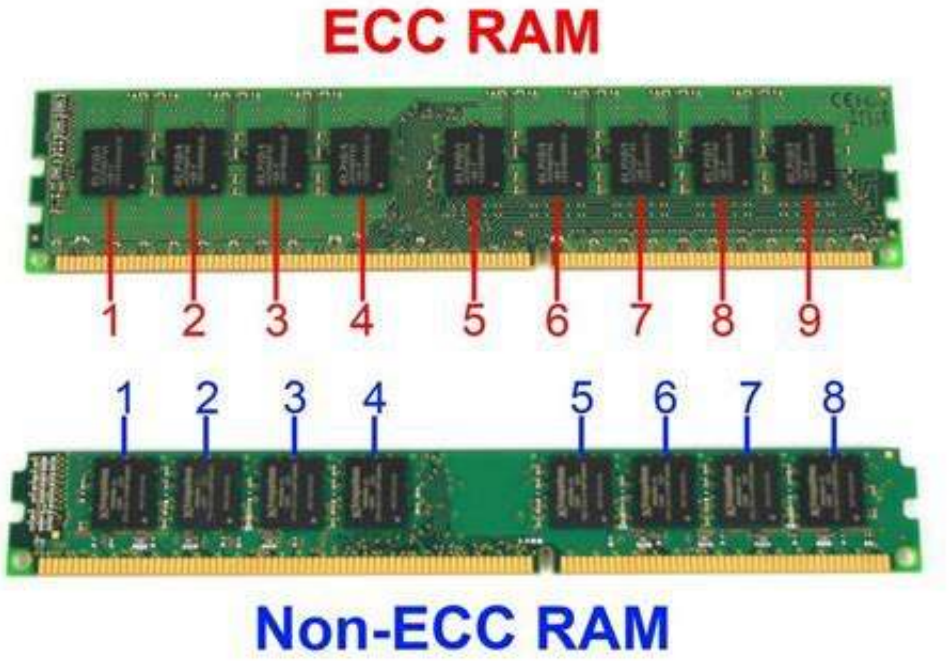
面试题

- 假设$S = \{1,2,\dots,n\} \setminus \{x\}$ $(x \in \{1,2,\dots,n\})$
  - 扫描S中的所有元素(顺序未知)，用$O(1)$的空间找出x

- 假设$S = \{1,2,\dots,n\} \setminus \{x,y\}$ $(x,y \in \{1,2,\dots,n\})$
  - 扫描S中的所有元素(顺序未知)，用$O(1)$的空间找出x,y

Error Correction Code (ECC)

- 计算机系统基础讲编码时学过
  - 想起来了吧！十分类似
  - 可以用来容忍偶尔射入内存的宇宙射线(...)



跟RAID有什么关系?

- 这不就能抵抗1个磁盘的损坏了吗?

- 任何n块磁盘中的一块坏了，都可以用剩下的把数据算出来
  - 哇哦!

Redundant Arrays of Inexpensive Disks (RAIDs)

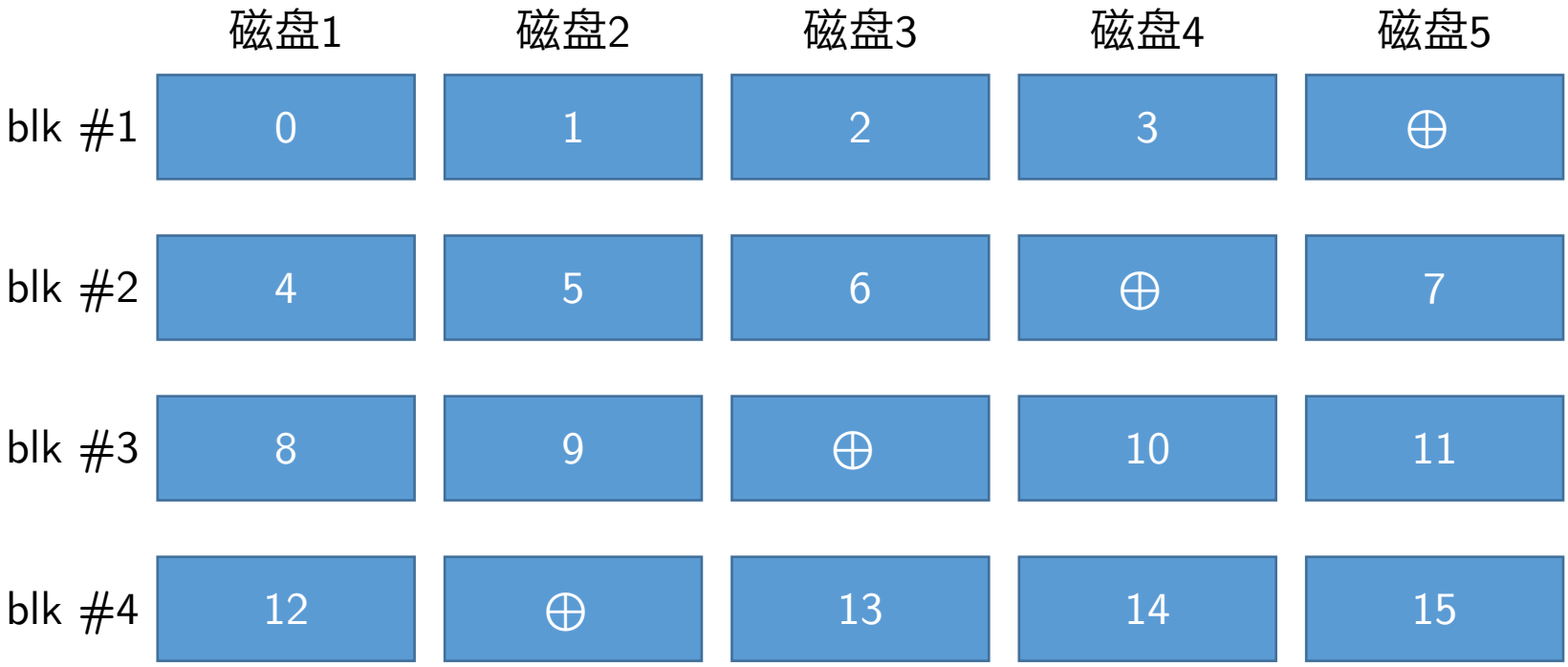
RAID-4：校验

- 多加一块盘，获得抵抗任意一块盘(包括P盘)出错的能力
  - 问题：如何维护磁盘P的数据?
  - 另外：磁盘P同样需要崩溃一致性

	磁盘1	磁盘2	磁盘3	磁盘4	磁盘P
blk #1	0	1	2	3	⊕
blk #2	4	5	6	7	⊕
blk #3	8	9	10	11	⊕
blk #4	12	13	14	15	⊕

RAID-5：负载均衡

- 让磁盘P不再成为瓶颈
  - 在实际中广泛使用



RAID-6 容忍两块盘的损坏

- 需要两个Parity Disks (P, Q)
  - $P = D_0 \oplus D_1 \oplus \cdots \oplus D_{n-1}$
  - $Q = g^0 D_0 \oplus g^1 D_1 \oplus \cdots \oplus g^{n-1} D_{n-1}$

- 在缺少i, j的时候
  - $A = D_i \oplus D_j$ (通过P推算)
  - $B = g^i D_i \oplus g^j D_j$ (通过Q推算)
  - $g^{n-i} B \oplus A = \left(g^{n-i+j} \oplus 1\right) D_j$

- 最后还需要理论救命

RAID：小结

RAID	磁盘数量	冗余空间	容错
RAID-0	n	$0\ (0)$	0
RAID-1	$2n$	$n\ (\frac{1}{2})$	1 -- n
RAID-4	$n+1$	$1\ (\frac{1}{n+1})$	1
RAID-5	$n+1$	$1\ (\frac{1}{n+1})$	1
RAID-6	$n+2$	$2\ (\frac{2}{n+2})$	2

n: 存放实际数据的磁盘数量

持久数据可靠性：更多问题

刚才我们做了一个假设

- 磁盘要么是“好的”，要么是“坏的”
  - 坏的盘就是“dead drive” ← fail stop
- 但实际上磁盘是悄悄坏掉的
  - 表面上看是好的(能写入/读出数据)
  - 但实际上数据已经不对了..... ← silent faults
- 你可能会觉得：碰上了是我rp不好呗
  - 但如果是数据中心里发生这种问题.....
  - 如果是某个批次的磁盘容易发生这种问题.....

更多的出错可能性

- Block corruption
  - 写入了莫名其妙的数据
- Misdirected writes
  - 写了x，磁盘写进了y
- Lost writes
  - 写了x，磁盘返回了，结果数据没进去(白忙活)
- 物理磁盘 – 虚拟磁盘 – 文件系统共同解决可靠性问题