Logging and Recovery
日志与恢复

事 务（Transaction）

- 逻辑单位
  - START
  - COMMIT
- 包含有限的数据库操作序列
  - READ(A)
  - WRITE(A, a)

事 务（Transaction）

- 逻辑单位
  - START
  - COMMIT
- 包含有限的数据库操作序列
  - READ(A)
  - WRITE(A, a)

用户A向用户B转账x

Transfer(A,B,x)

```
START
a_bal = READ(A)
WRITE(A, a_bal - x)
b_bal = READ(B)
WRITE(B, b_bal + x)
COMMIT
```

ACID

ACID

- A (Atomicity): 原子性
- C (Consistency): 一致性
- I (Isolation): 隔离性
- D (Durability): 持久性

ACID

- A (Atomicity): 原子性
  - each transaction be "all or nothing"
- C (Consistency): 一致性
  - any transaction will bring the database from one valid state to another
- I (Isolation): 隔离性
  - the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially
- D (Durability): 持久性
  - once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors

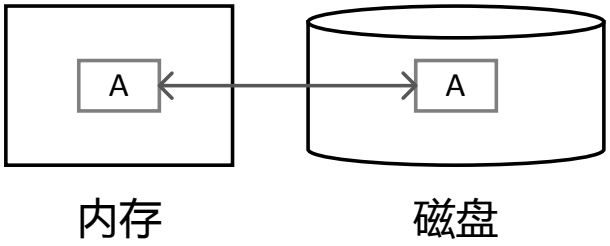
故 障（Failure）

- 存储介质故障（Media failure）
- 灾难性故障（Catastrophic failure）
- **系统故障**（System failure）
  - 断电
  - 软件中止
  - 操作系统中止

故 障（Failure）

- 存储介质故障（Media failure）
- 灾难性故障（Catastrophic failure）
- **系统故障**（System failure）
  - 断电
  - 软件中止
  - 操作系统中止

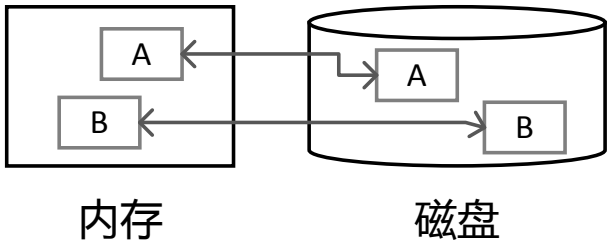
（1）计算机体系结构



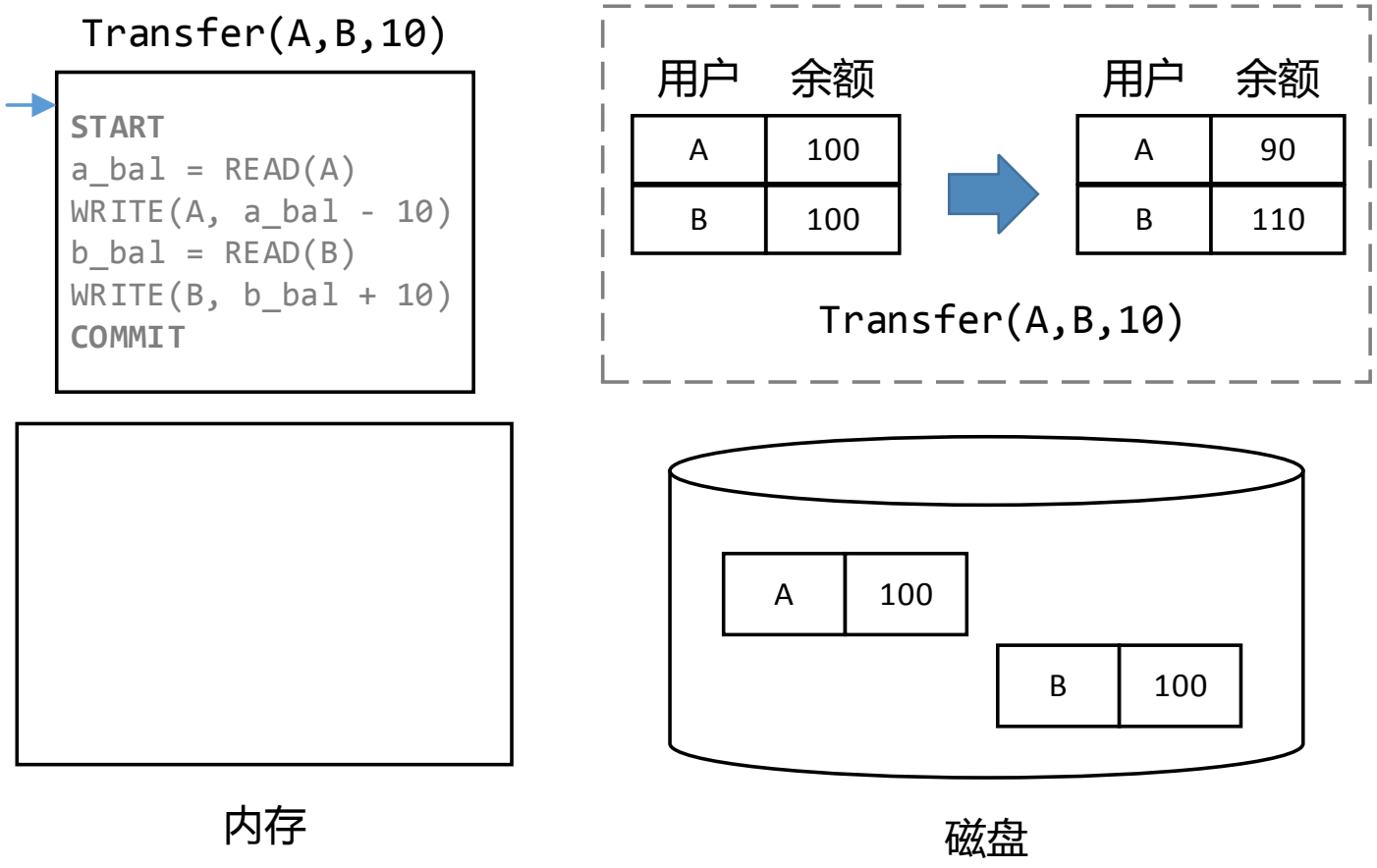
故 障（Failure）

- 存储介质故障（Media failure）
- 灾难性故障（Catastrophic failure）
- **系统故障**（System failure）
  - 断电
  - 软件中止
  - 操作系统中止

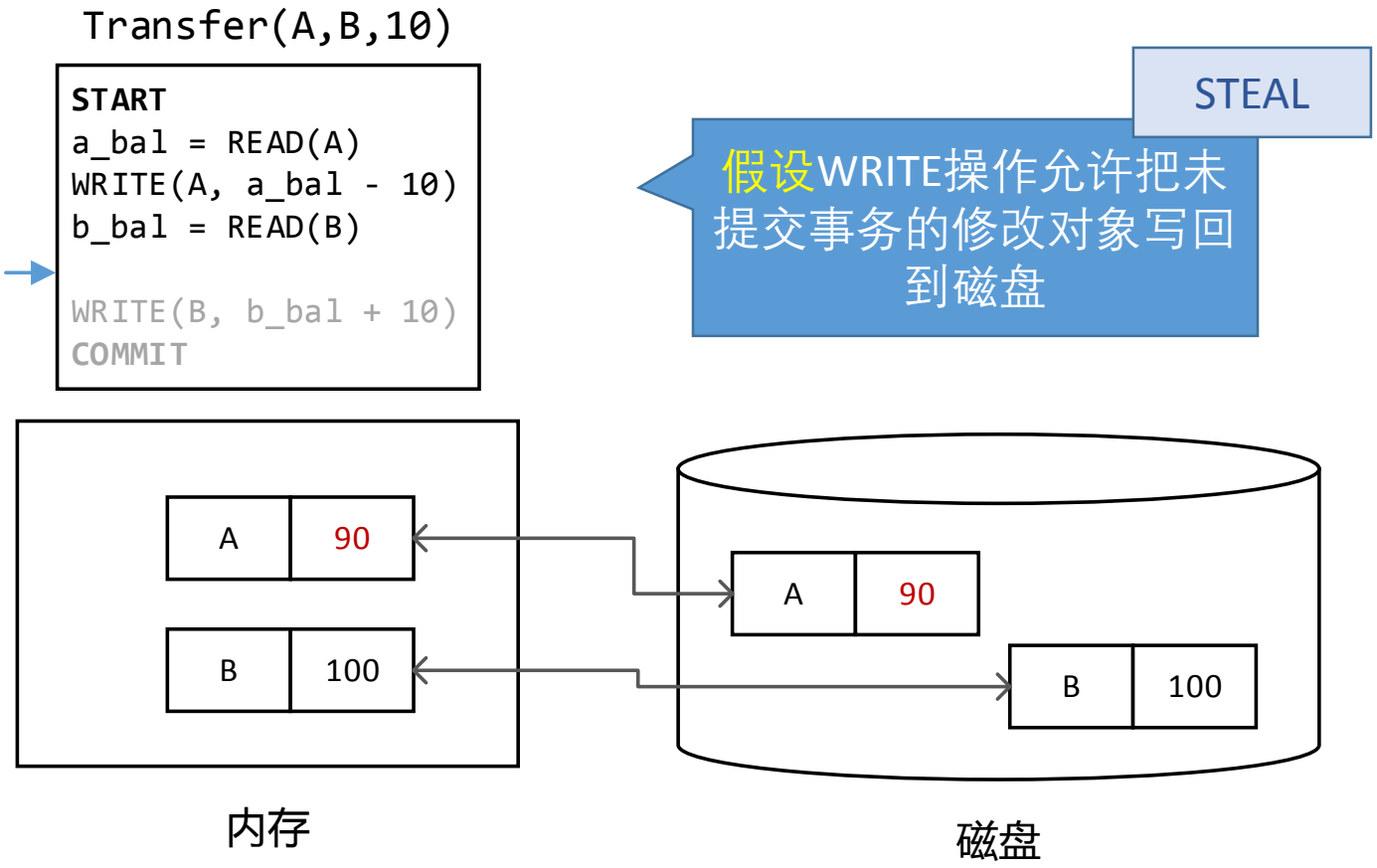
（1）计算机体系结构
（2）单个事务包含多个修改对象



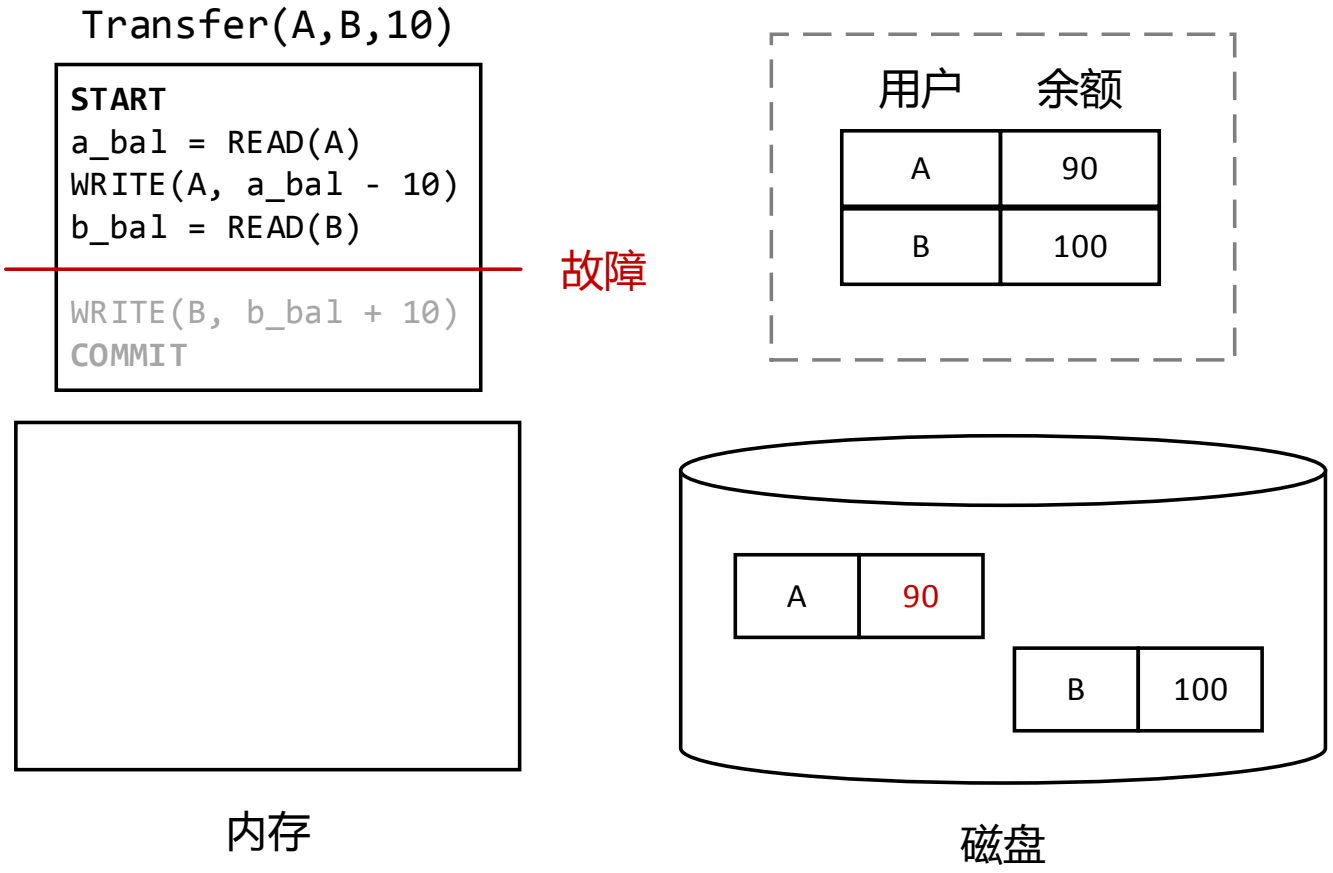
故 障（Failure）


Transfer(A,B,10)
START
a_bal = READ(A)
WRITE(A, a_bal - 10)
b_bal = READ(B)
WRITE(B, b_bal + 10)
COMMIT
用户
余额
A
100
B
100
用户
余额
A
90
B
110
Transfer(A,B,10)
A
100
B
100
内存
磁盘

故 障（Failure）


Transfer(A,B,10)
START
a_bal = READ(A)
WRITE(A, a_bal - 10)
b_bal = READ(B)
WRITE(B, b_bal + 10)
COMMIT
STEAL
假设WRITE操作允许把未提交事务的修改对象写回到磁盘
A
90
B
100
A
90
B
100
内存
磁盘

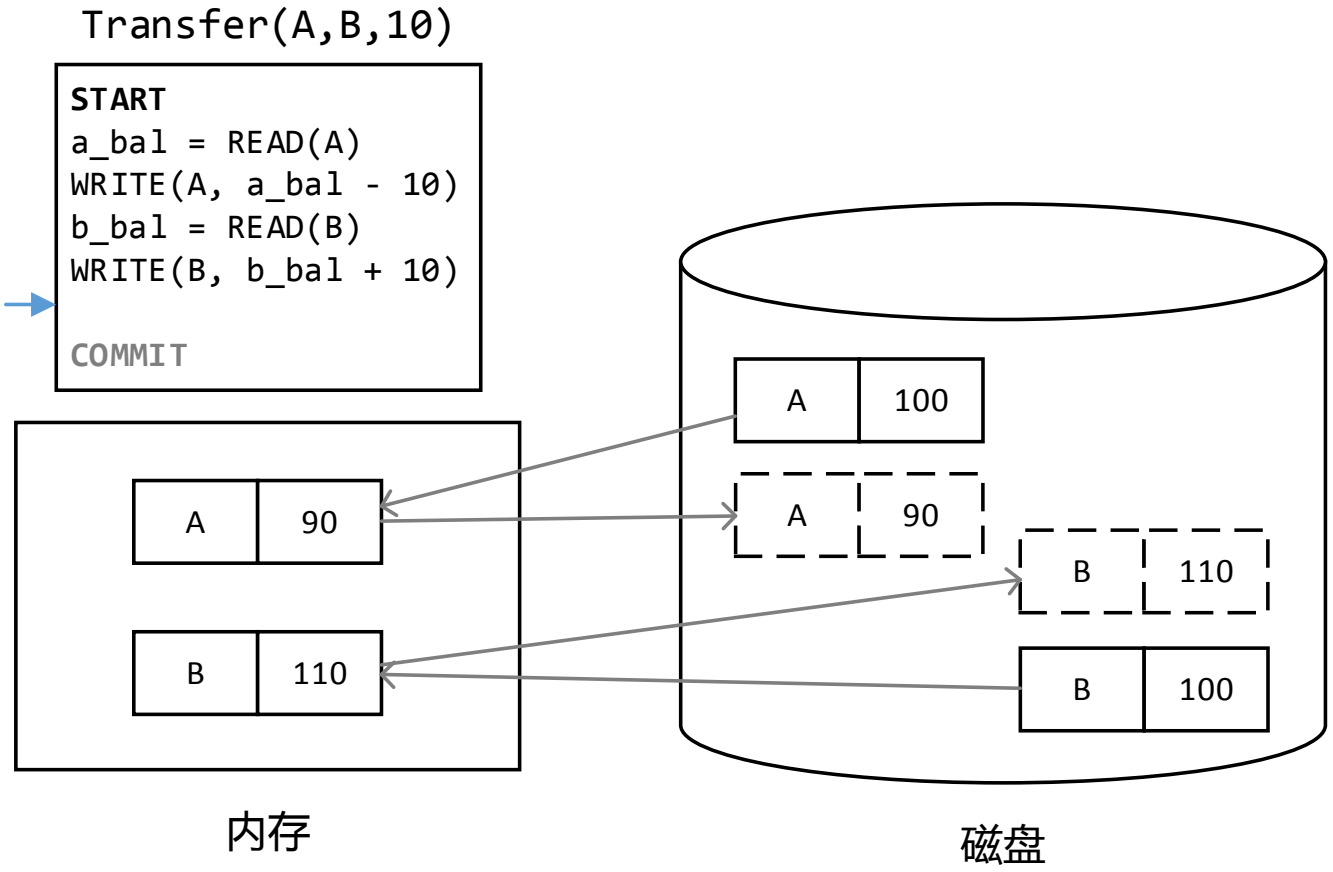
故 障（Failure）


Transfer(A,B,10)
START
a_bal = READ(A)
WRITE(A, a_bal - 10)
b_bal = READ(B)
WRITE(B, b_bal + 10)
COMMIT
故障
内存
用户
余额
A
90
B
100
A
90
B
100
磁盘

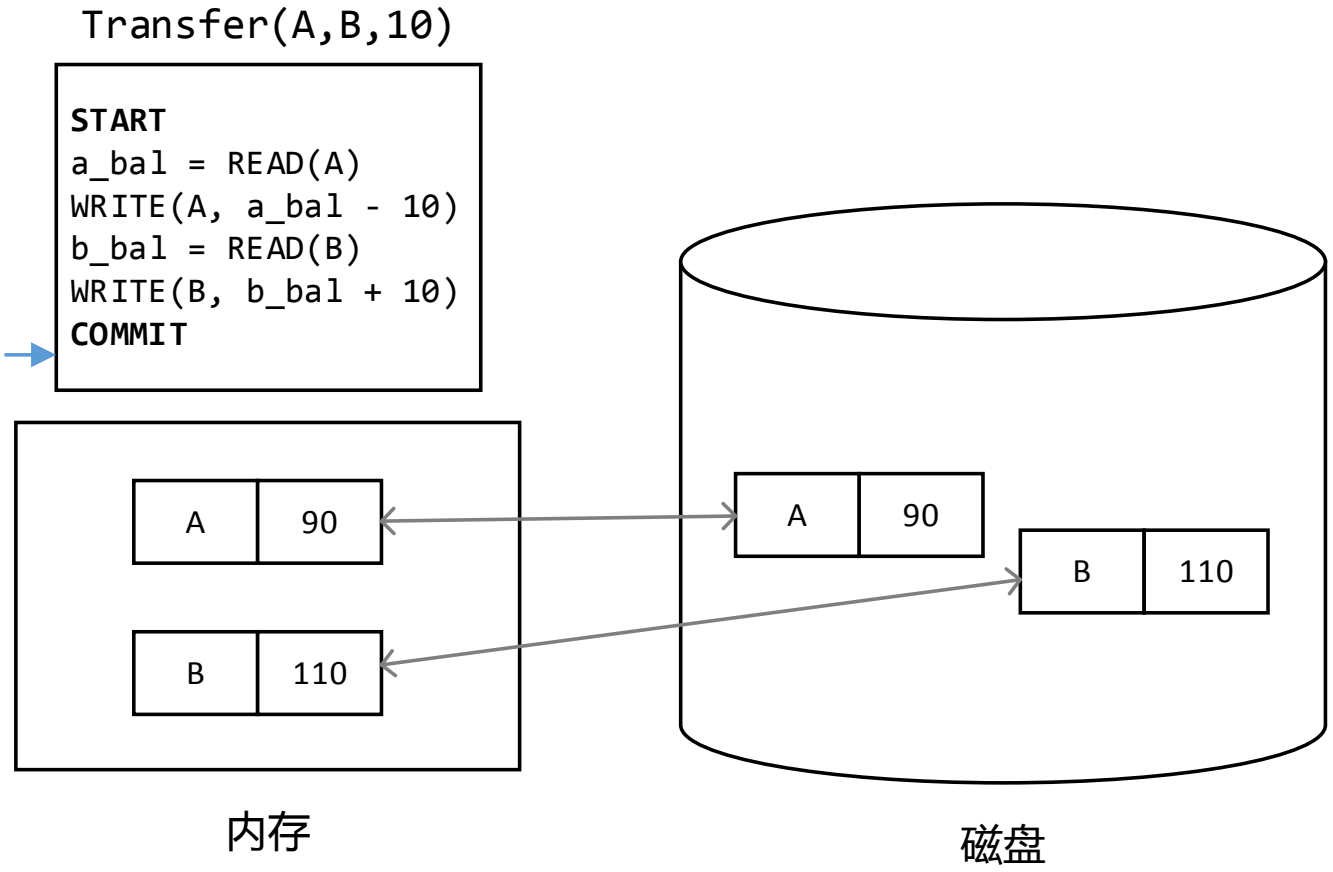
故障恢复

- Shadow paging technique
- 先写日志WAL (write-ahead logging)

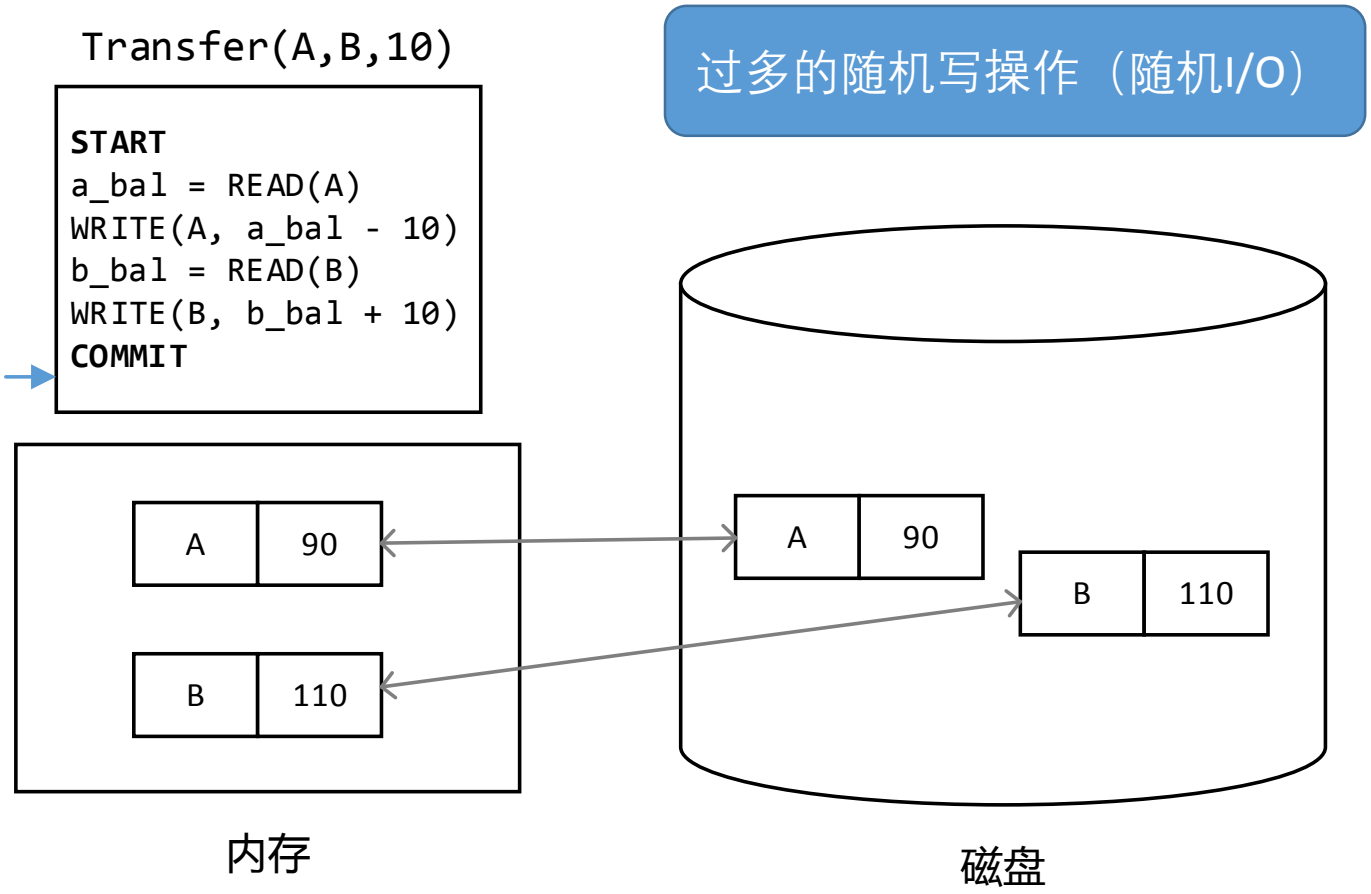
Shadow Paging



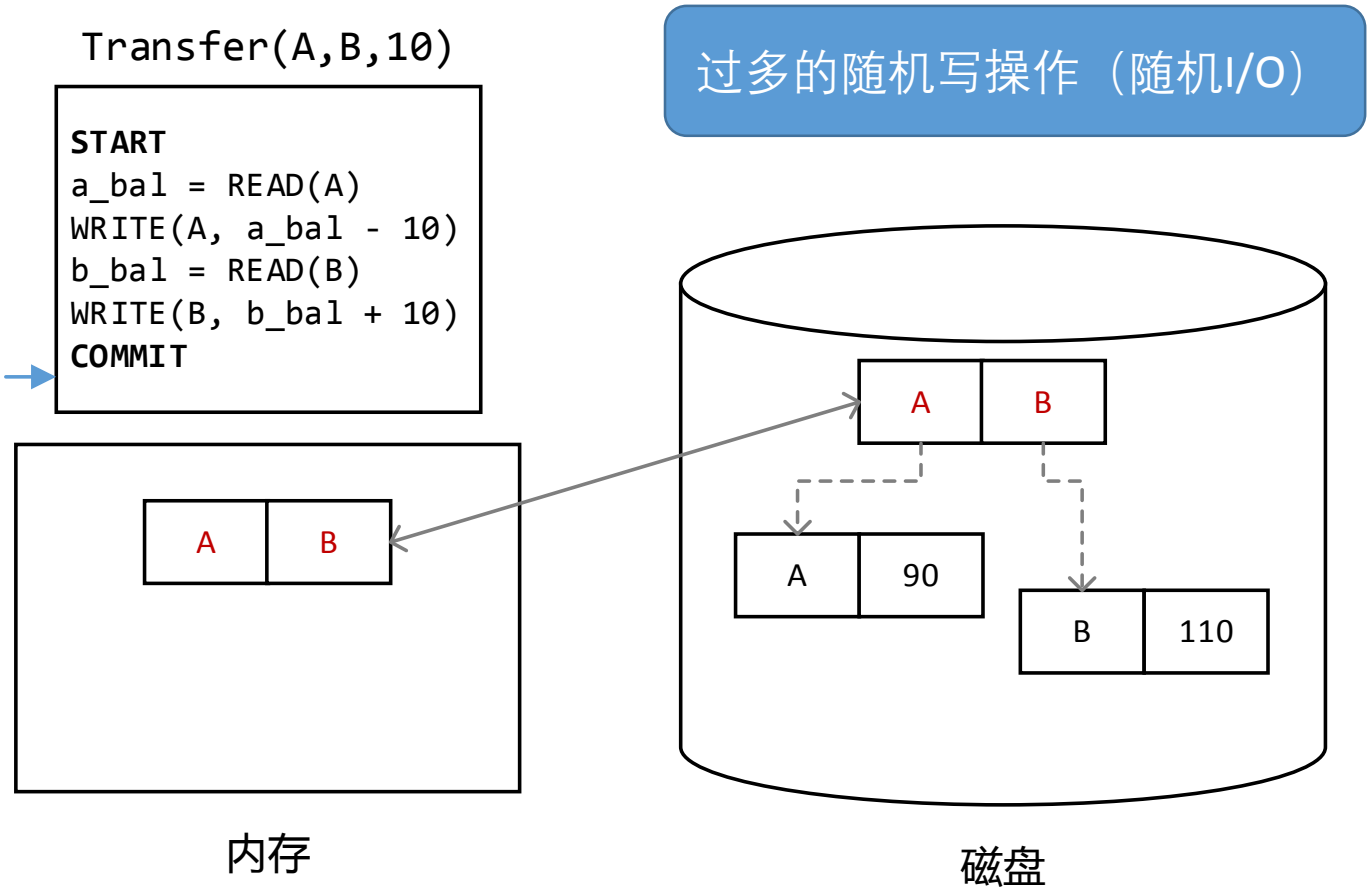
Shadow Paging



Shadow Paging



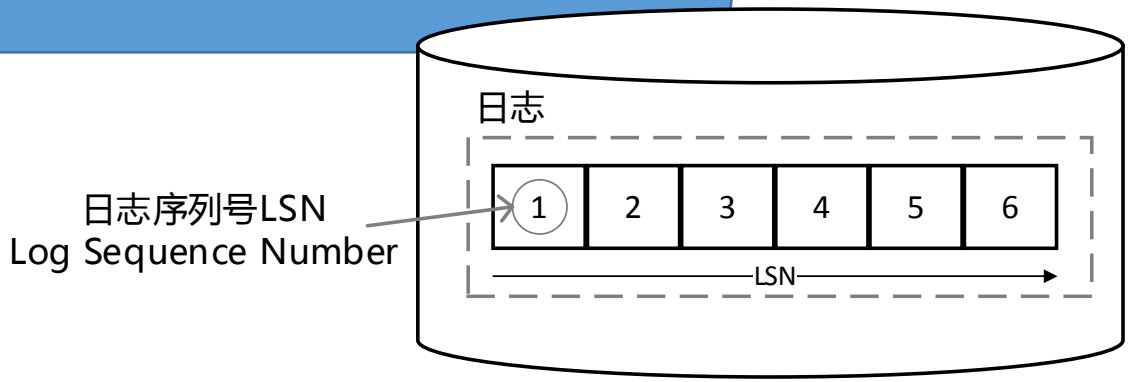
Shadow Paging



先写日志WAL

- 撤销日志（Undo logging）技术
- 重做日志（Redo logging）技术

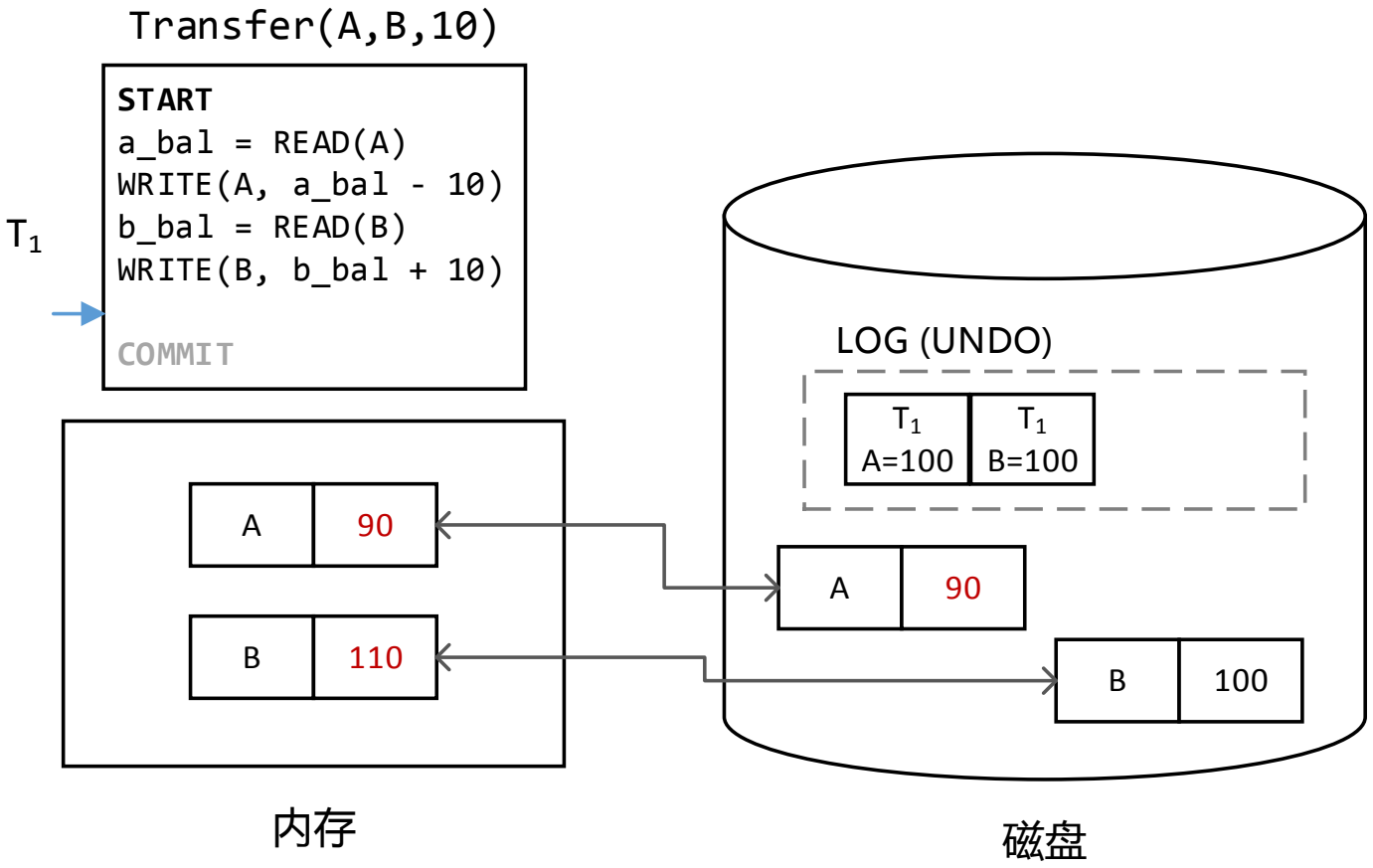
1. 将对数据库的修改记录在单独的存储空间中（日志）
2. 日志只支持追加操作（顺序I/O）
3. 修改的数据对象持久化之前，需要保证其对应的修改已记录在日志文件中（WAL）



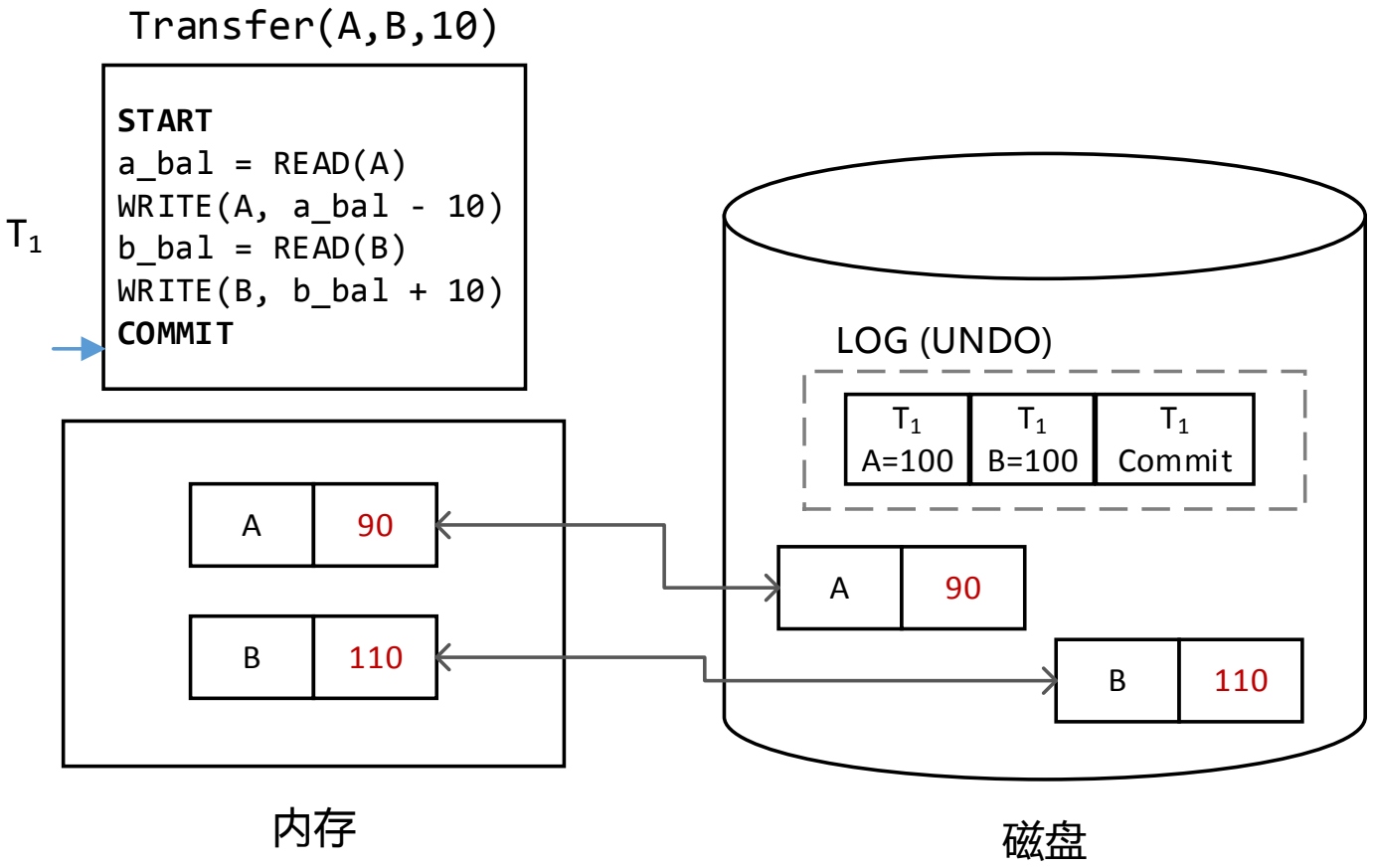
撤销日志（Undo Logging）

- 事务的每一个修改操作产生一条撤销日志记录
  - 包含对象修改之前的值
- 事务修改的对象在持久化到磁盘之前，对应的撤销日志记录必须已持久化到磁盘
  - WAL (write-ahead logging)
- 在事务的提交日志（commit）写入到日志之前，该事务修改过的对象必须已持久化到磁盘。
  - Force

撤销日志


Transfer(A,B,10)
START
a_bal = READ(A)
WRITE(A, a_bal - 10)
b_bal = READ(B)
WRITE(B, b_bal + 10)
COMMIT
T1
A
90
B
110
内存
LOG (UNDO)
T1
A=100
T1
B=100
A
90
B
100
磁盘

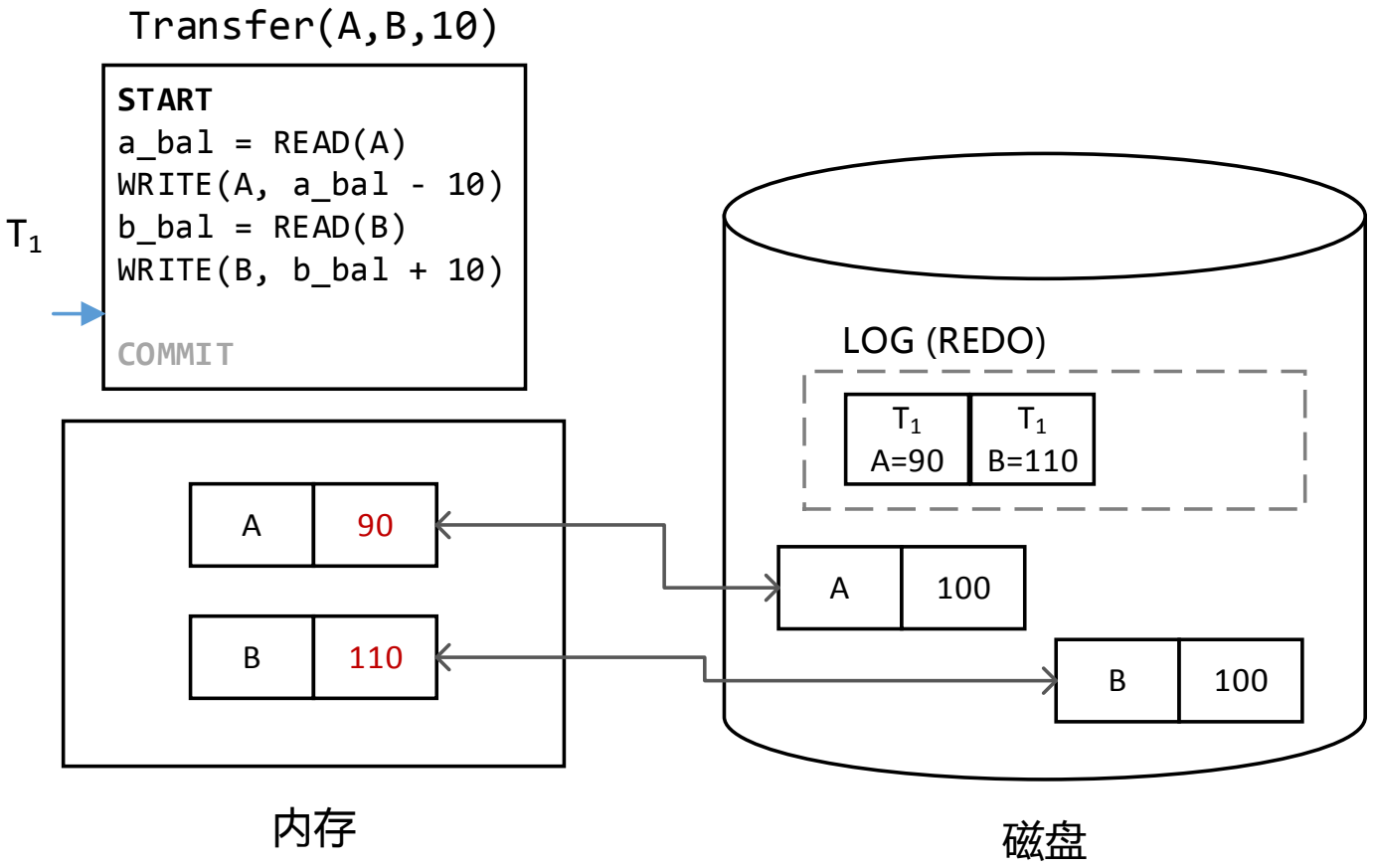
撤销日志


Transfer(A,B,10)
START
a_bal = READ(A)
WRITE(A, a_bal - 10)
b_bal = READ(B)
WRITE(B, b_bal + 10)
COMMIT
T1
LOG (UNDO)
T1 A=100
T1 B=100
T1 Commit
A
90
B
110
A
90
B
110
内存
磁盘

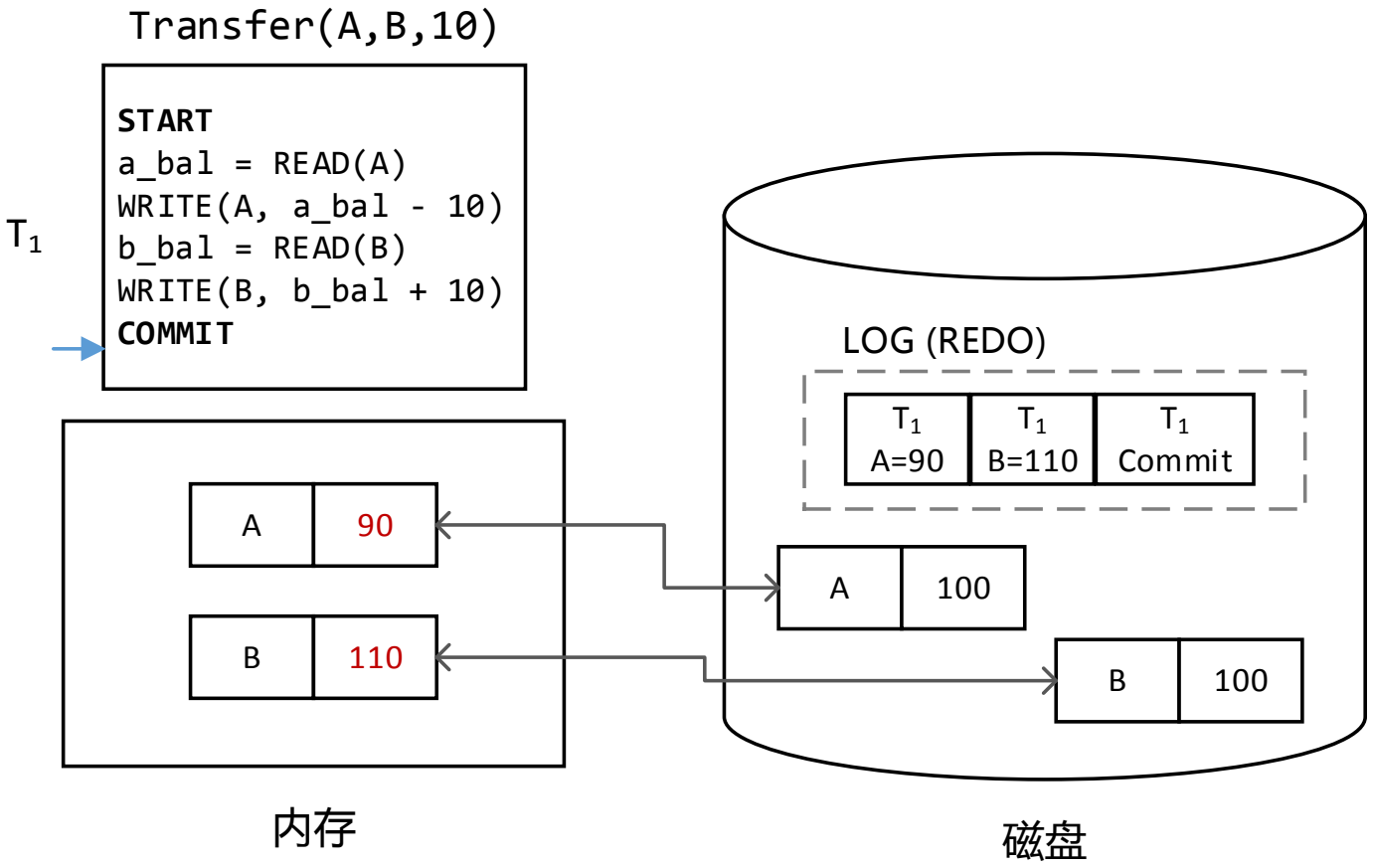
重做日志（Redo Logging）

- 事务的每一个修改操作产生一条重做日志记录
  - 包含对象修改之后的值
- 事务修改的对象在持久化到磁盘之前，对应的重做日志记录和提交日志必须已持久化到磁盘
  - WAL (write-ahead logging)
  - No-Force

重做日志


Transfer(A,B,10)
START
a_bal = READ(A)
WRITE(A, a_bal - 10)
b_bal = READ(B)
WRITE(B, b_bal + 10)
COMMIT
T1
A
90
B
110
内存
LOG (REDO)
T1
A=90
T1
B=110
A
100
B
100
磁盘

重做日志


Transfer(A,B,10)
START
a_bal = READ(A)
WRITE(A, a_bal - 10)
b_bal = READ(B)
WRITE(B, b_bal + 10)
COMMIT
T1
LOG (REDO)
T1 A=90
T1 B=110
T1 Commit
A
90
B
110
A
100
B
100
内存
磁盘

重做/撤销日志（Redo/Undo Logging）

- 撤销日志
  - 磁盘I/O增加
- 重做日志
  - 需要的内存空间增加

Redo/Undo
日志记录

1. 先写日志WAL
2. 提交日志落盘后，对应的事务即可提交

高级日志技术

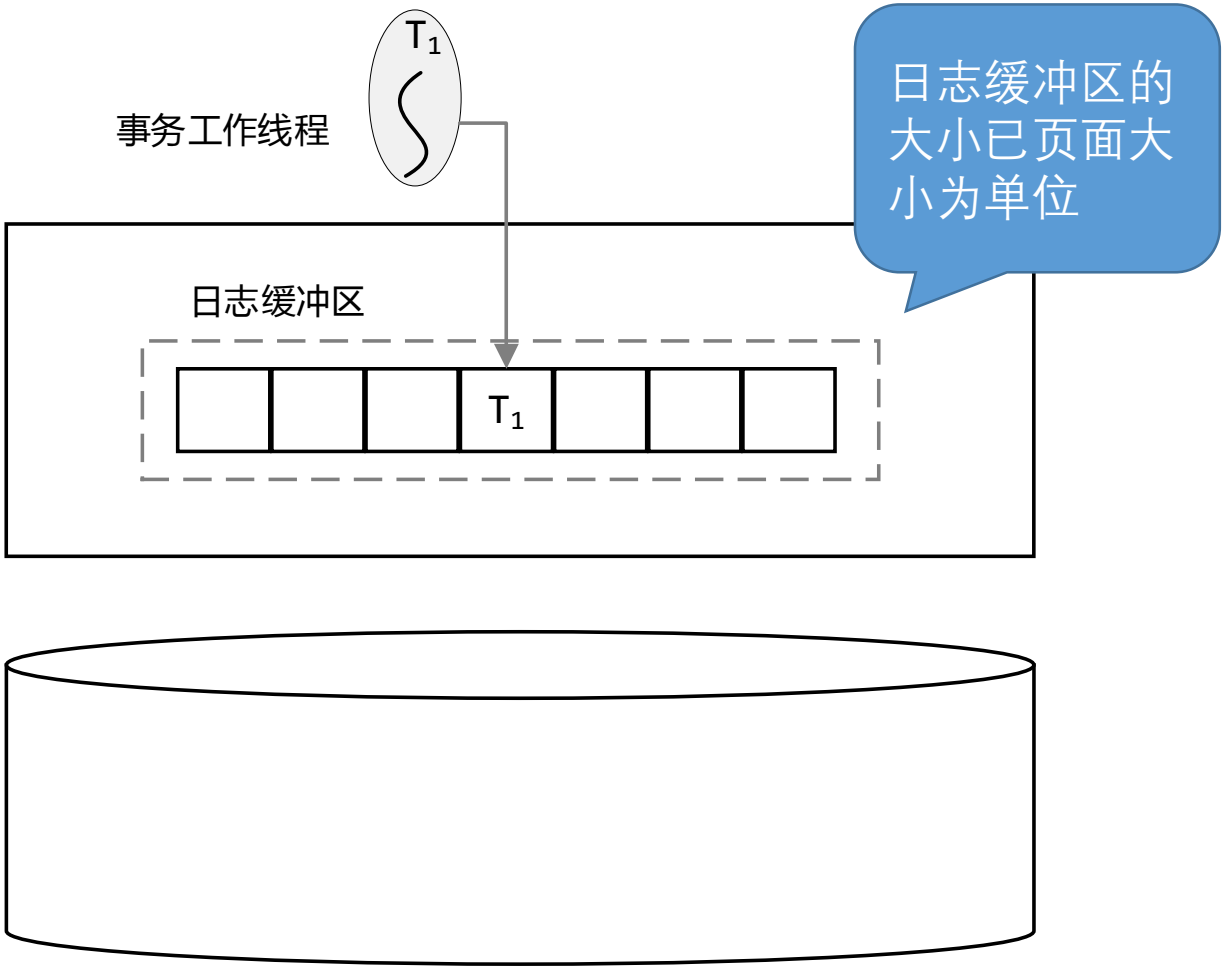
- 组提交
- ARIES

组提交（Group Commit）

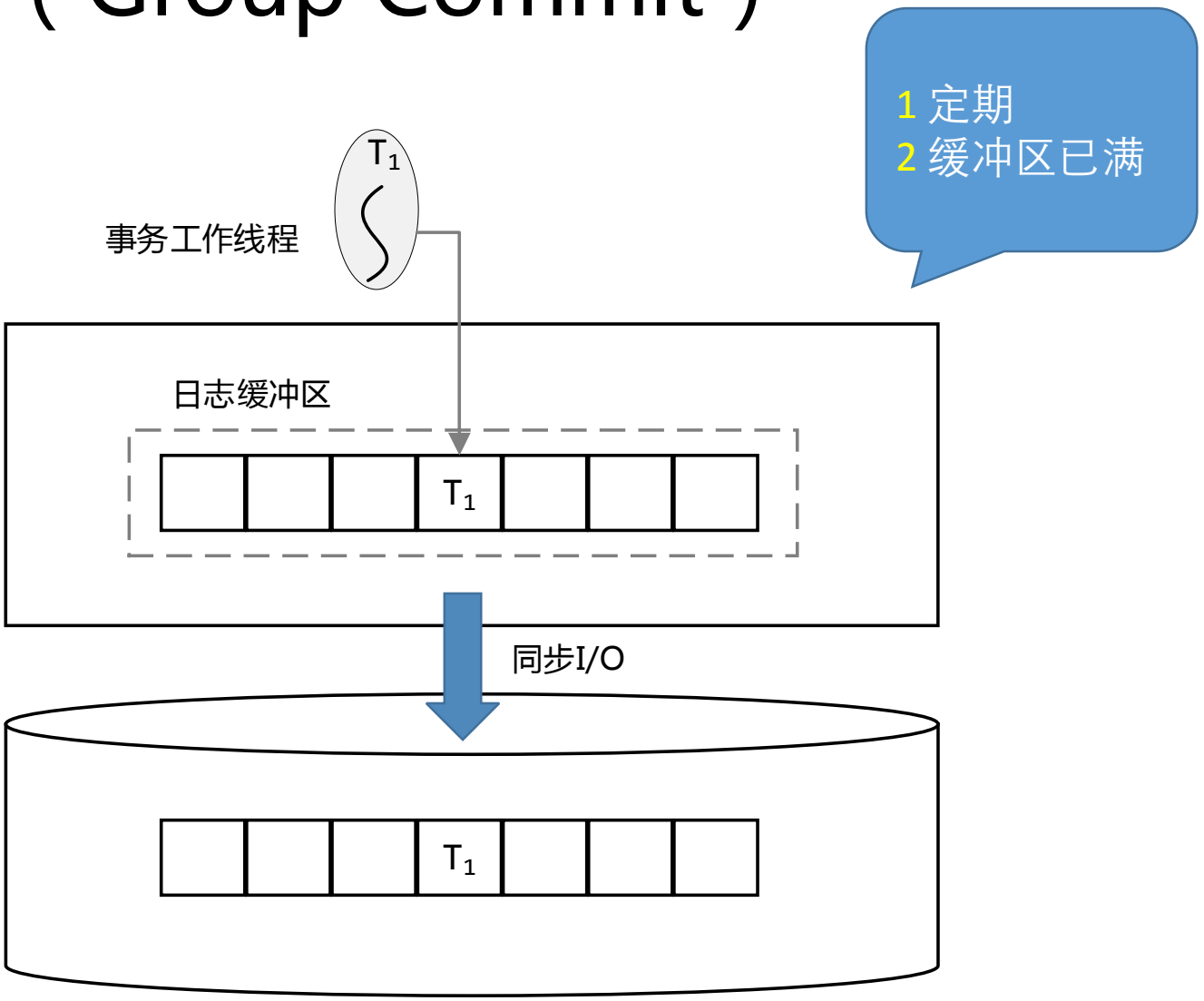
- 存储设备访问的基本单位“页”
- 存储设备的访问次数有限

假设：
单页面大小4K字节，单日志记录512字节
磁盘每秒能执行10次I/O

组提交（Group Commit）



组提交（Group Commit）



ARIES

- Algorithm for Recovery and Isolation Exploiting Semantics
- 关键特征
  - 先写日志WAL
  - Physical redo (page-oriented redo)
  - Logical undo
  - 记录页面状态 (page_LSN)
  - 补偿日志记录 (CLR, compensation log record)

ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging
[C. Mohan] [TODS 1992]

ARIES

- 正常日志流程
  1. fix the page containing the target object
  2. latch the page in X mode
  3. perform the operation
  4. generate the log record
  5. update page_LSN to the LSN of the log record
  6. unlatch the page
  7. unfix the page

ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging
[C. Mohan] [TODS 1992]

ARIES

- 恢复流程
  1. 分析日志（按照日志顺序）
  2. 重做所有日志（按照日志序，包含失败事务的日志）
  3. 撤销失败事务（按照日志反序）

ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging
[C. Mohan] [TODS 1992]

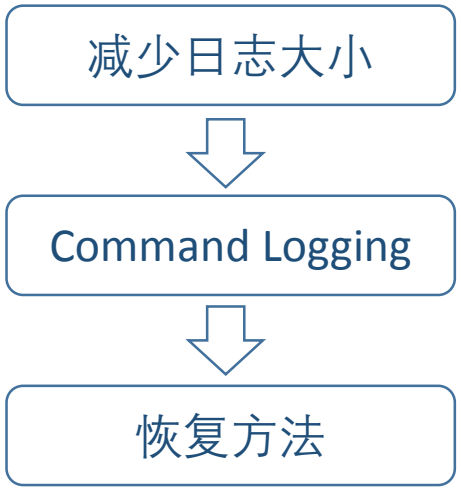
局限性

局限性

- 存储设备瓶颈
  - 有限的I/O
  - 有限的带宽
- 集中式日志缓冲区
- 全序日志

局限性

- 存储设备瓶颈
  - 有限的I/O
  - 有限的带宽
- 集中式日志缓冲区
- 全序日志



主题1：新型内存数据库恢复机制

Command Logging	HStore
Rethinking Main Memory OLTP Recovery [Nirmesh Malviya] [ICDE 2014]	

Command Logging	Peloton
Fast Failure Recovery for Main-Memory DBMSs on Multicores [Yingjun Wu] [SIGMOD 2017]	

局限性

- 存储设备瓶颈
  - 有限的I/O
  - 有限的带宽
- 集中式日志缓冲区
- 全序日志

高I/O
低延迟
高吞吐

非易失性存储

替代内存或磁盘

针对性的数据结构

主题2：面向新硬件的恢复机制

适用于ARIES

NVM, Log

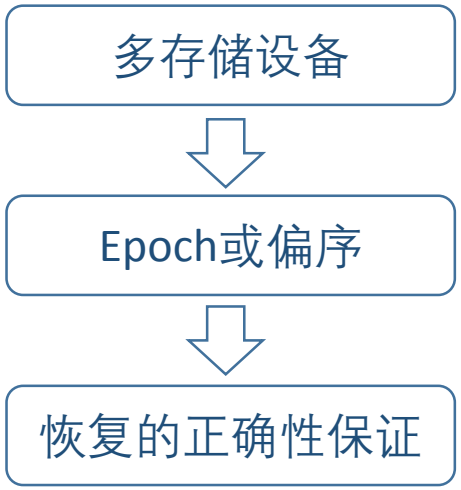
NVRAM-aware Logging in Transaction Systems
[Jian Hunag] [VLDB 2015]

NVM, Log, multi-core

Scalable Database Logging for Multicores
[Hyungsoo Jung] [VLDB 2018]

局限性

- 存储设备瓶颈
  - 有限的I/O
  - 有限的带宽
- 集中式日志缓冲区
- 全序日志

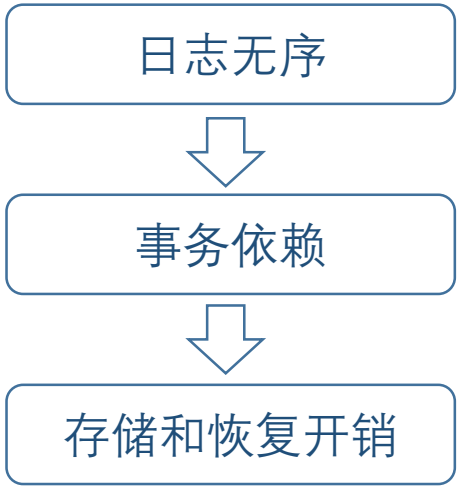


主题3：面向多存储设备的恢复机制

Main memory, multi-core Fast Databases with Fast Durability and Recovery Through Multicore Parallelism [Wenting Zheng] [OSDI 2014]	SiloR
NVM, multi-core Scalable logging through emerging non-volatile memory [Tianzheng Wang] [VLDB 2014]	

局限性

- 存储设备瓶颈
  - 有限的I/O
  - 有限的带宽
- 集中式日志缓冲区
- 全序日志



主题4：基于事务依赖的并行日志恢复机制

仅适用于内存数据库

Log

DBx1000

Taurus: A Parallel Transaction Recovery Method Based on Fine-Granularity Dependency Tracking
[Xiangyao Yu] []

主题安排

主题	汇报人	日期
新型内存数据库恢复机制	史琪锴	2018年3月21日
面向新硬件的恢复机制	周欢	2018年3月28日
面向多存储设备的恢复机制	王嘉豪	2018年4月4日
基于事务依赖的并行日志恢复机制	肖冰	2018年4月11日

Thank you!
Q & A