CNCC

新型硬件环境下大数据处理技术

当内存计算遇上NVM

陈世敏

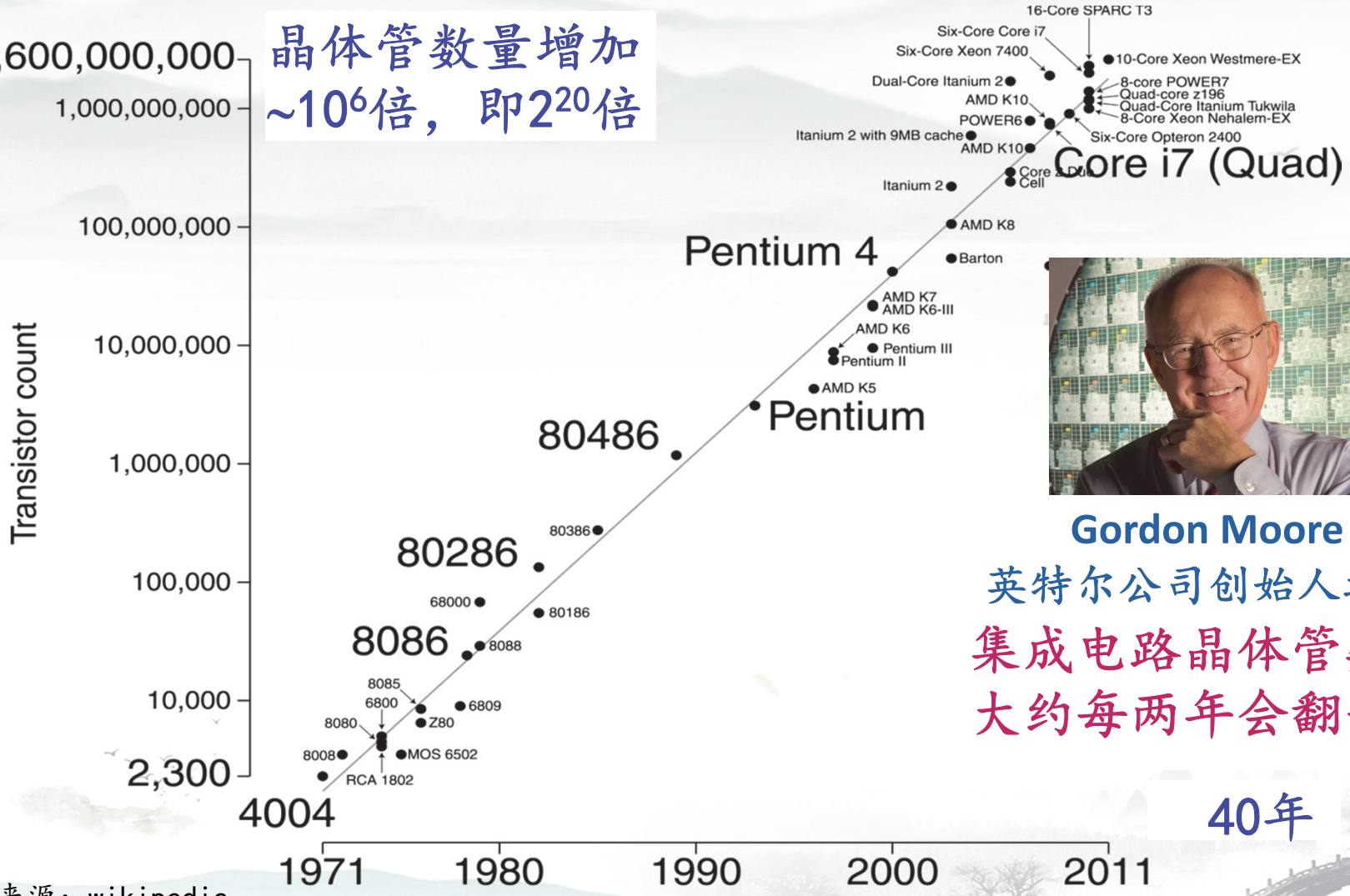
中科院计算所

Outline

- 内存计算
- NVM
- 内存计算+NVM

摩尔定律 (Moore's Law)

晶体管数量增加
~10^6倍，即2^{20}倍



Gordon Moore

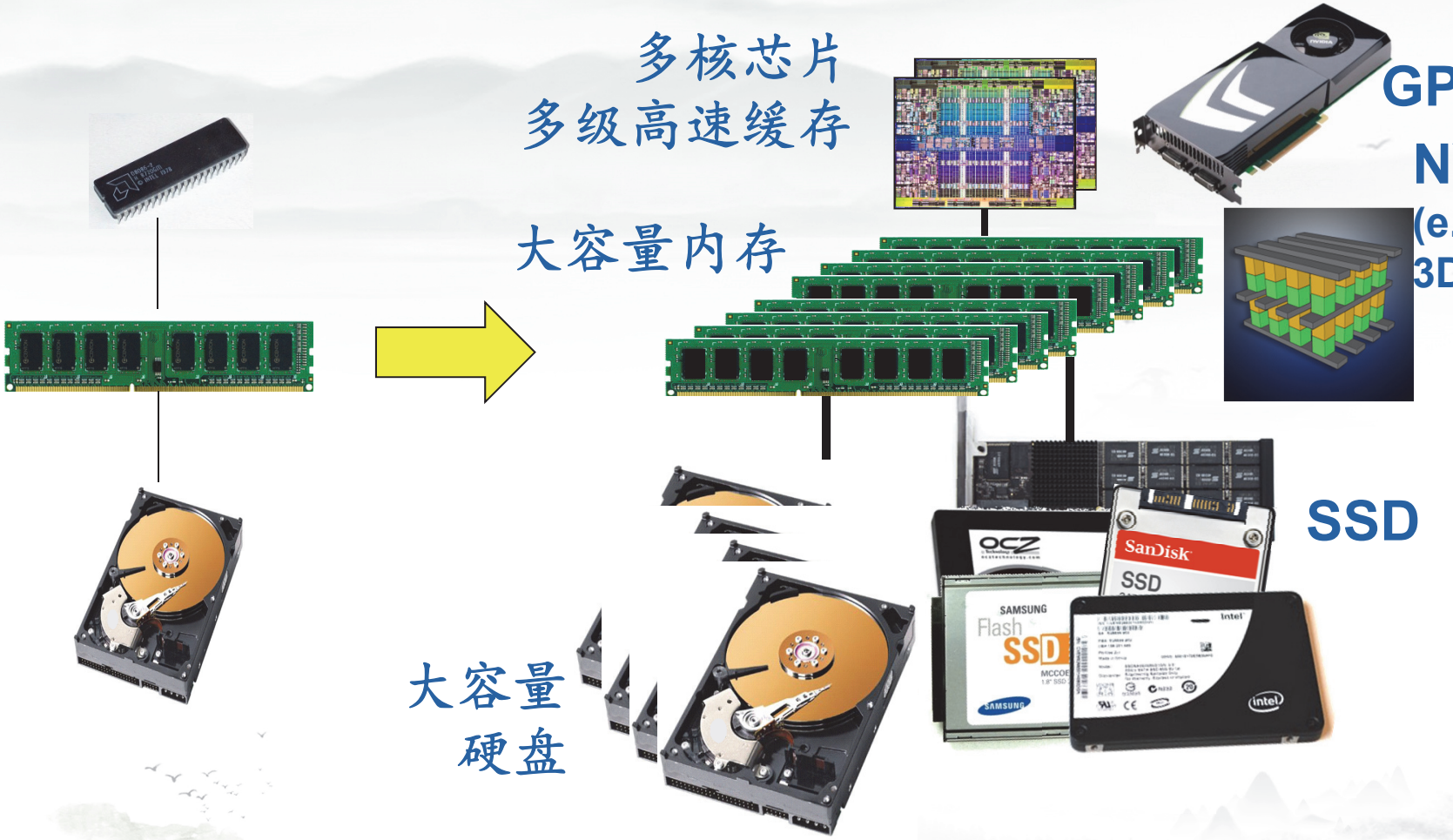
英特尔公司创始人之一

集成电路晶体管数量
大约每两年会翻一番

40年

数据来源：wikipedia

体系结构和硬件技术的巨大发展



内存计算

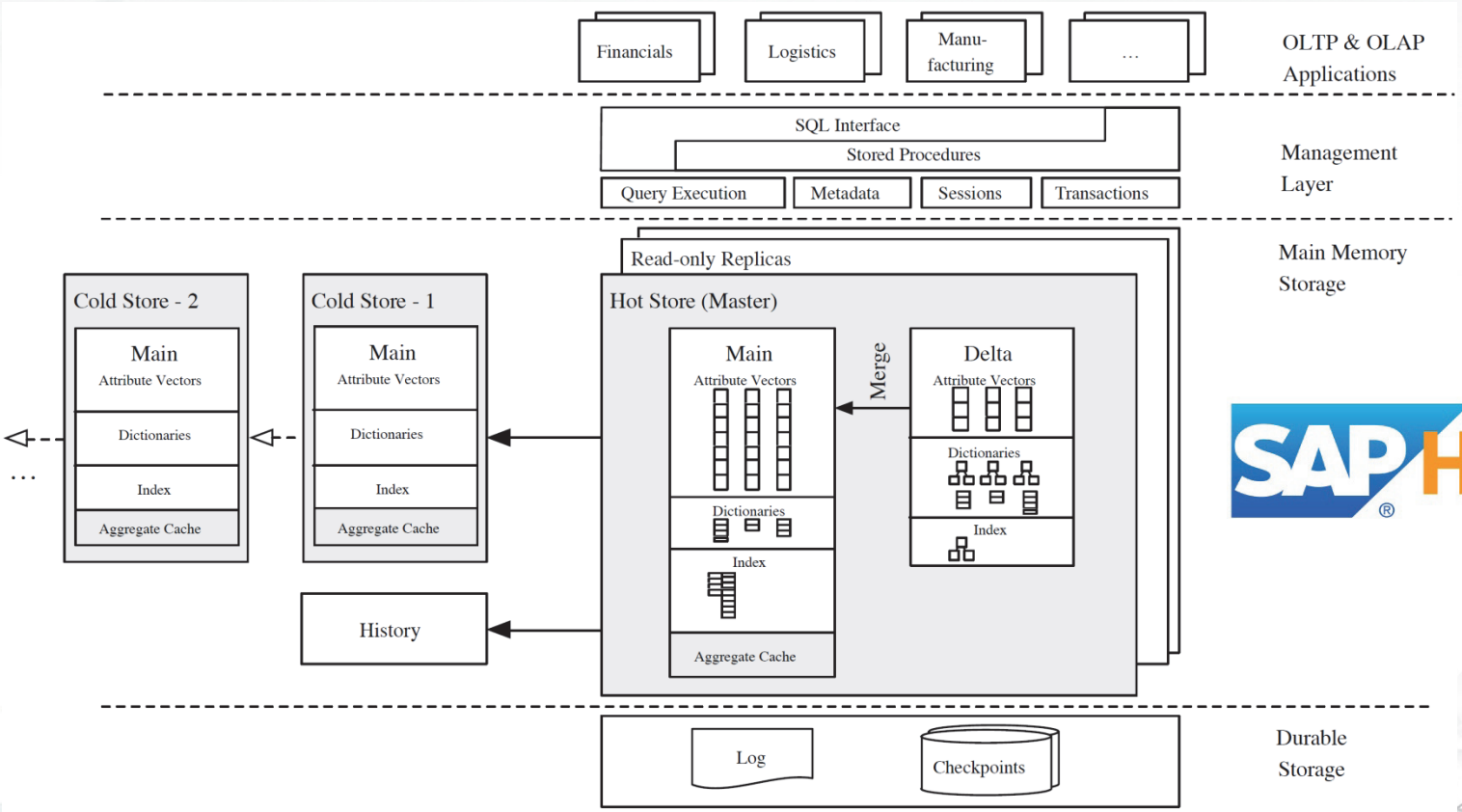
- 随着内存容量的指数级增加
  - 每台服务器256GB，一组刀片16台，就是4TB
  - 越来越大的数据集可以完全存放在内存中
- 内存计算的优点
  - 去除了外存访问的开销
  - 提高了处理速度

内存计算已经遍地开花

- 内存数据库系统
- 内存KV系统
- 内存大数据系统
- 内存图计算系统
- 。。。。。。

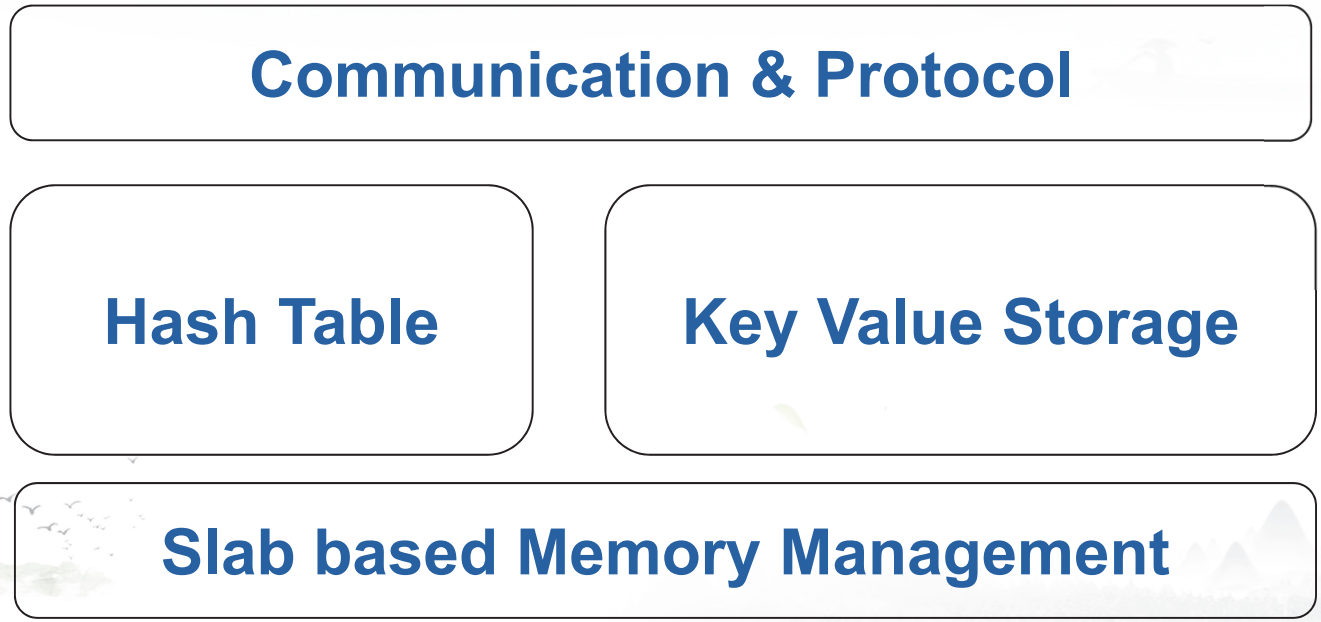
内存数据库系统

- 近年来，主流数据库公司纷纷增加了内存数据库引擎
  - Microsoft Hekaton & Apollo, SAP HANA, IBM BLU, 等

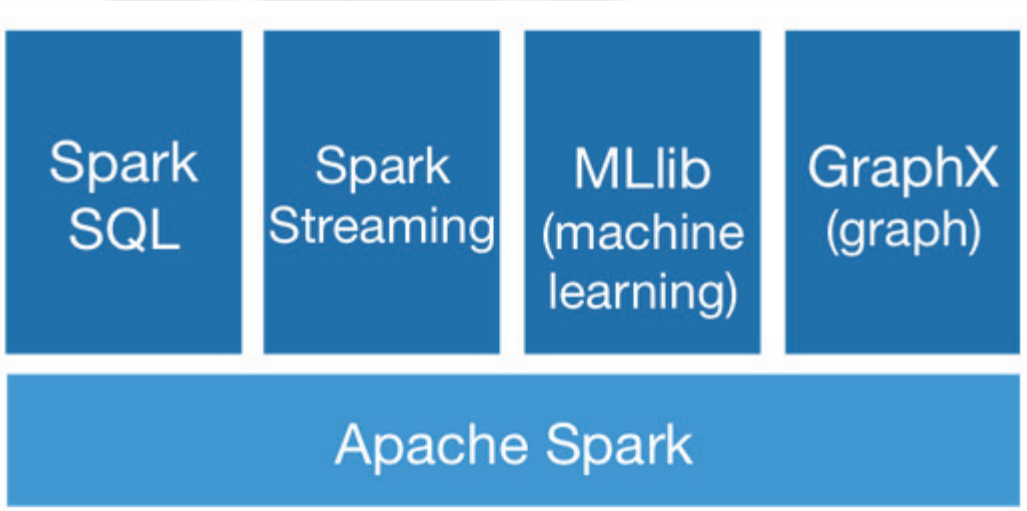


内存KV系统

- Memcached与Redis
- 支持put(key, value), get(key)

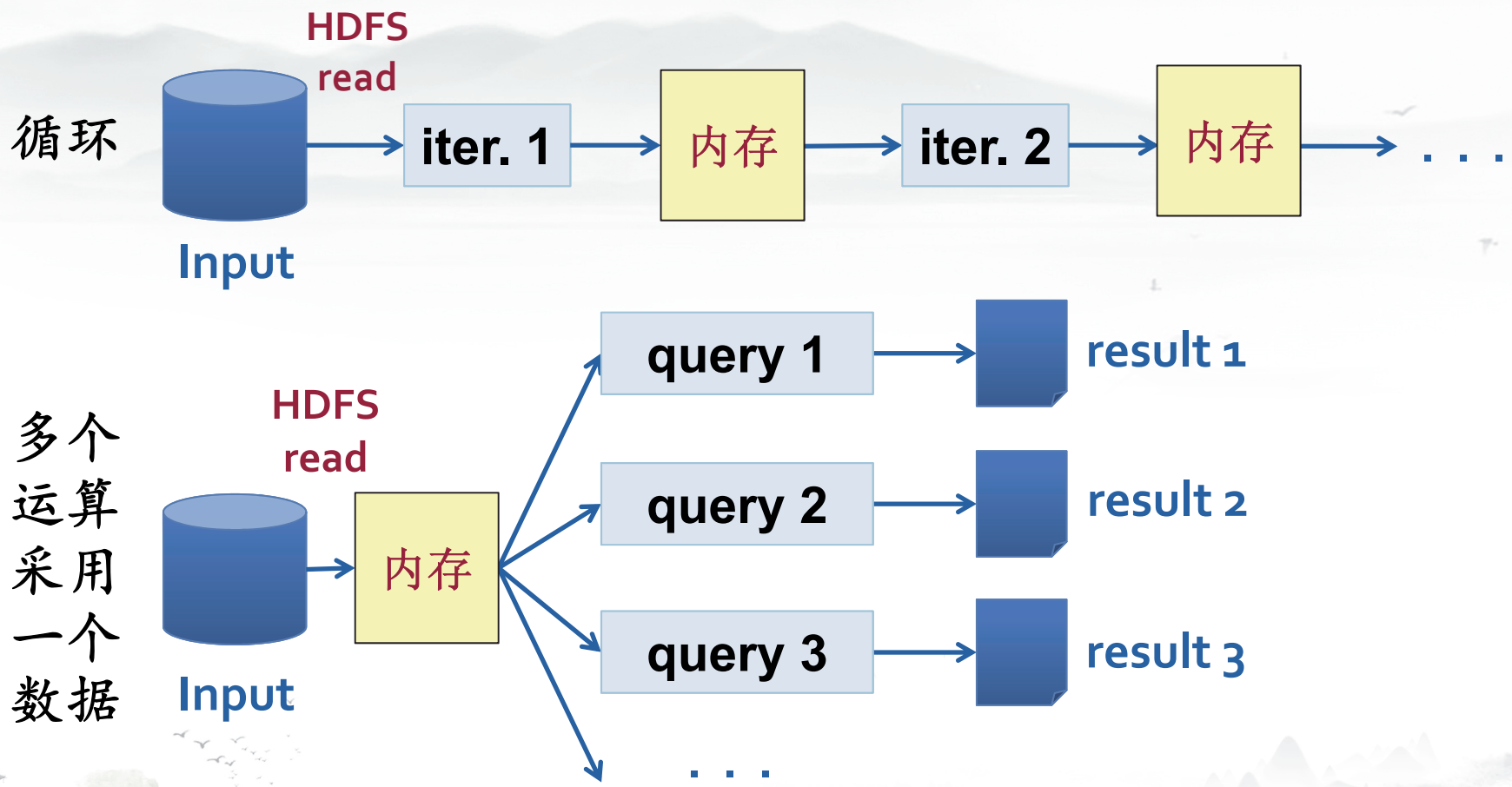


Spark: 面向大数据分析的内存系统

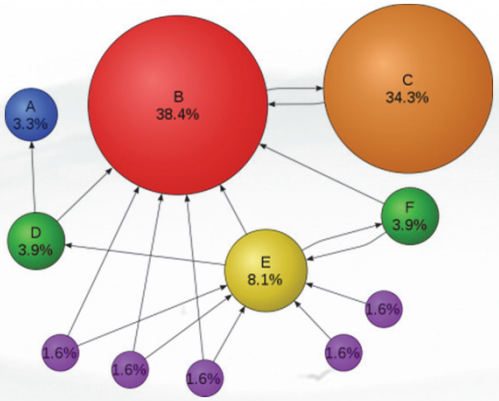


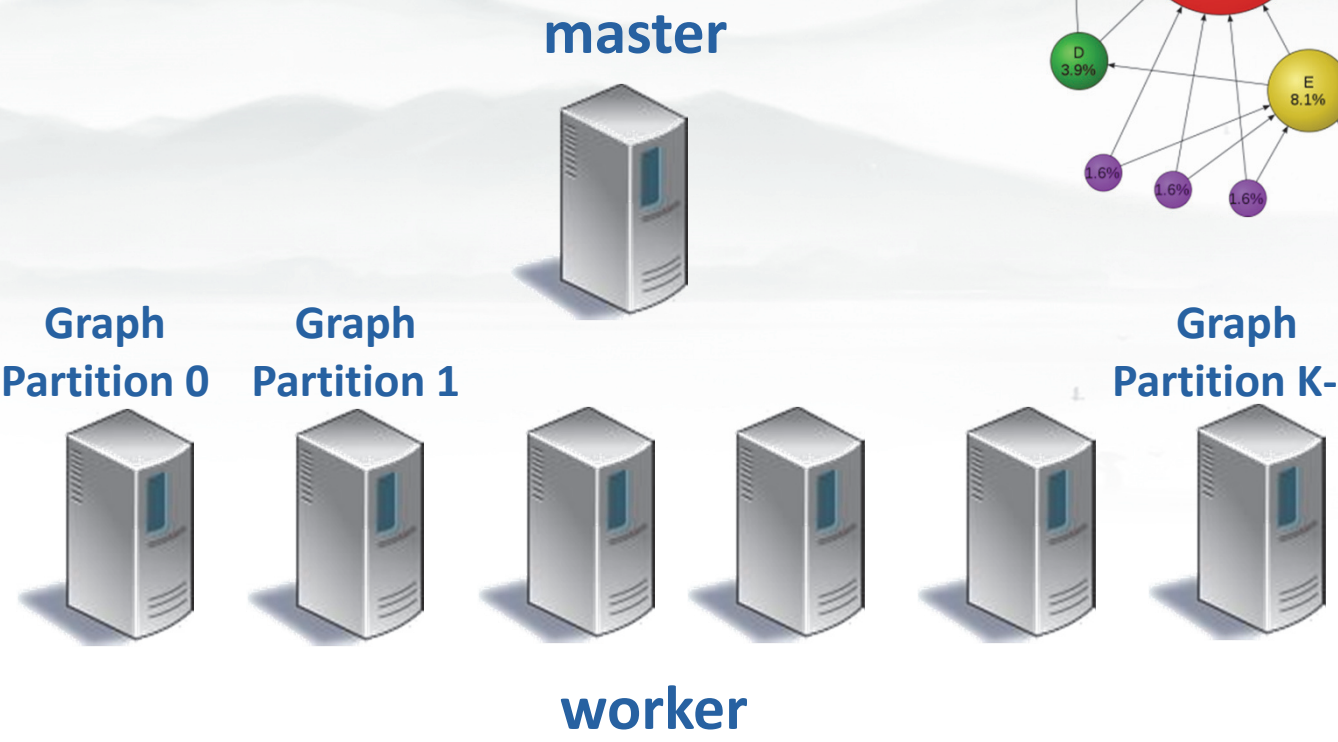
- Berkeley AMP Lab研发
- “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing ”, NSDI’12

Spark的思路：减少外存访问



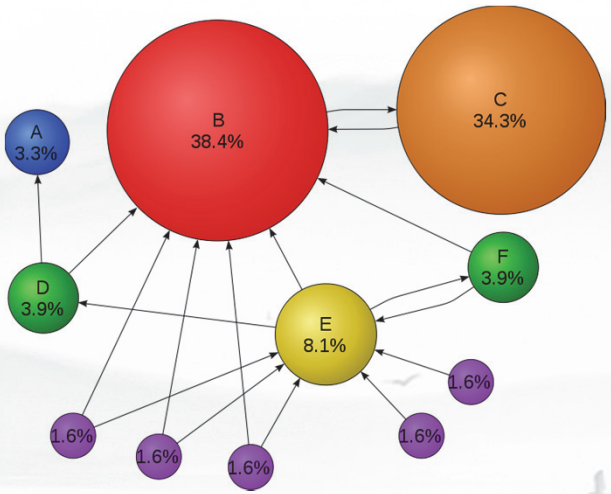
内存图计算系统





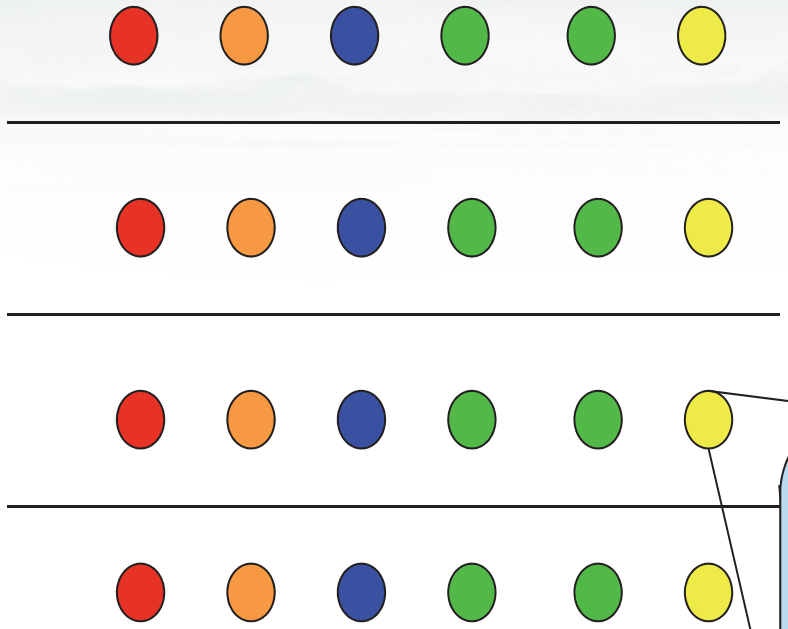
- 主要类型
  - Pregel类型的同步图计算：Pregel, Giraph, GraphLite等
  - GAS图计算：PowerGraph, GraphX等

Pregel图计算模型



超步内，并行执行每个顶点

运算分成多个超步



超步间全局同步

顶点算法通常步骤

1) 接收上个超步发出的in-neighbor的消息
2) 计算当前顶点的值
3) 向out-neighbor发消息

Outline

- 内存计算
- NVM
- 内存计算+NVM

集成电路向下扩展：特征尺寸不断变小

数据来源：wikipedia, ITRS

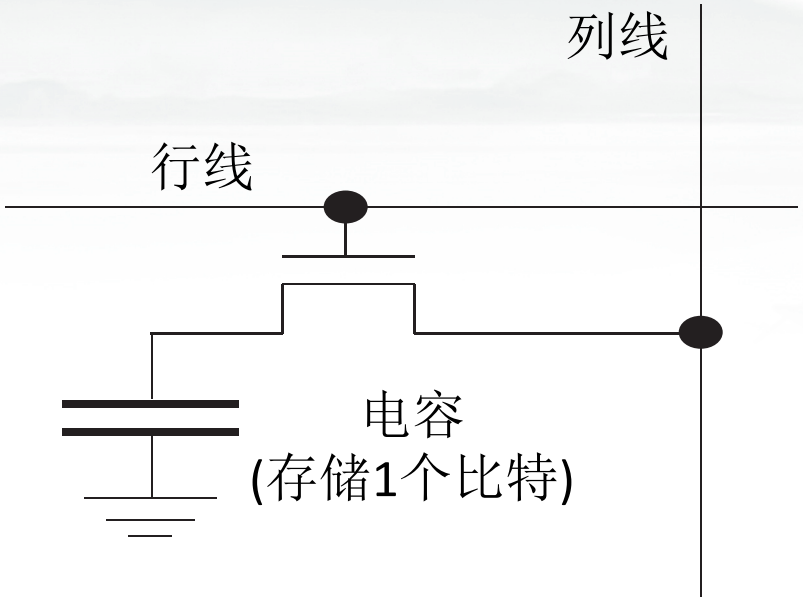
1971	1974	1977	1982	1985	1989	1994	1995	1997	1999
10μm	6μm	3μm	1.5μm	1μm	800nm	600nm	350nm	250nm	180nm
2001	**2004**	**2006**	**2008**	**2010**	**2012**	**2014**	**2017**	**2019**	**2021?**
130nm	90nm	65nm	45nm	32nm	22nm	14nm	**10nm**	**7nm**	**5nm**

- **1nm(纳米) = 10^{-9}米，原子的直径：~0.1nm**

☞**DRAM面积将越来越小！**

DRAM

- DRAM单元是由电容存储电荷，表示0/1



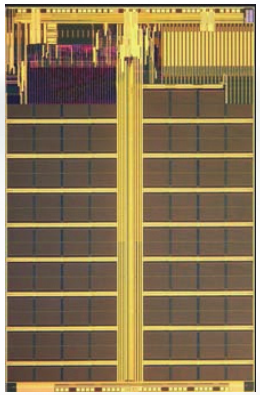
- 向下扩展遇到难题：面积变小➔存储电荷变少

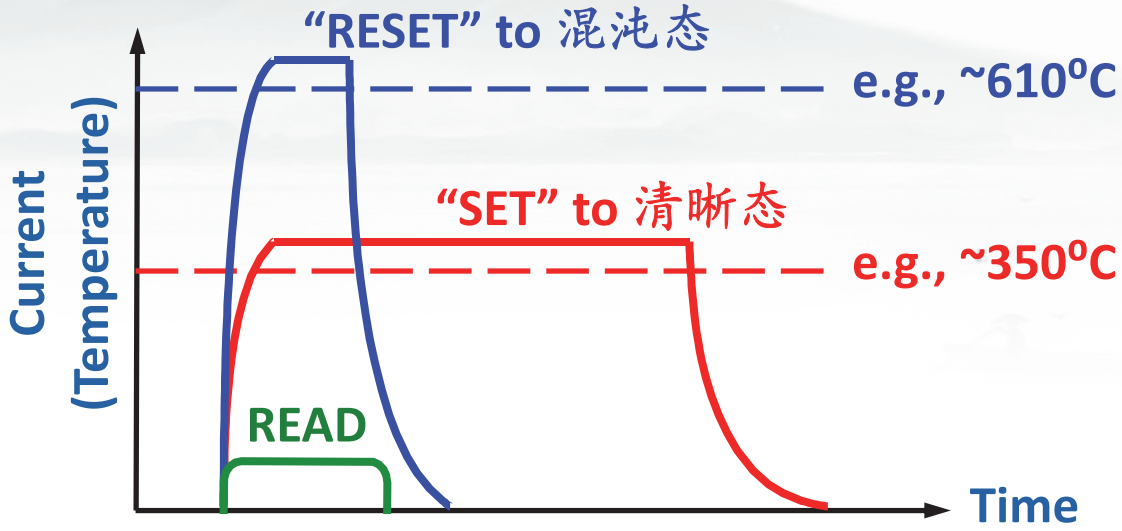
新的NVM(Non-Volatile Memory)技术

- 解决方法：电阻表示0/1
  - 通过某种物理过程改变存储单元的电阻值

- 代表技术
  - **PCM** (Phase Change Memory)：相变存储器
  - **STT-RAM** (Spin-Torque-Transfer RAM)：自旋扭矩转换-存储器
  - **Memristor / RRAM** (Resistive RAM)：忆阻器
  - **3D Xpoint**

PCM (Phase Change Memory)
相变存储器

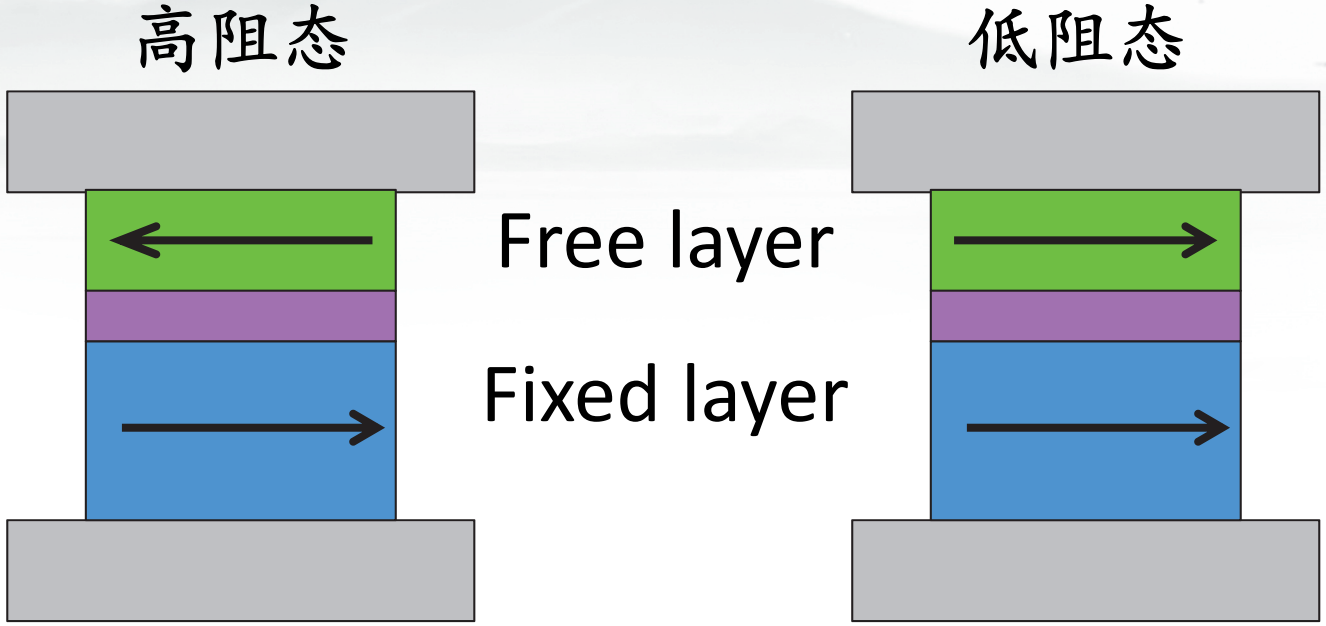




- 类似可写光盘的材料
- 利用材料的电性能
  - 混沌态：高电阻，清晰态：低电阻
- 发展得最成熟的NVM技术

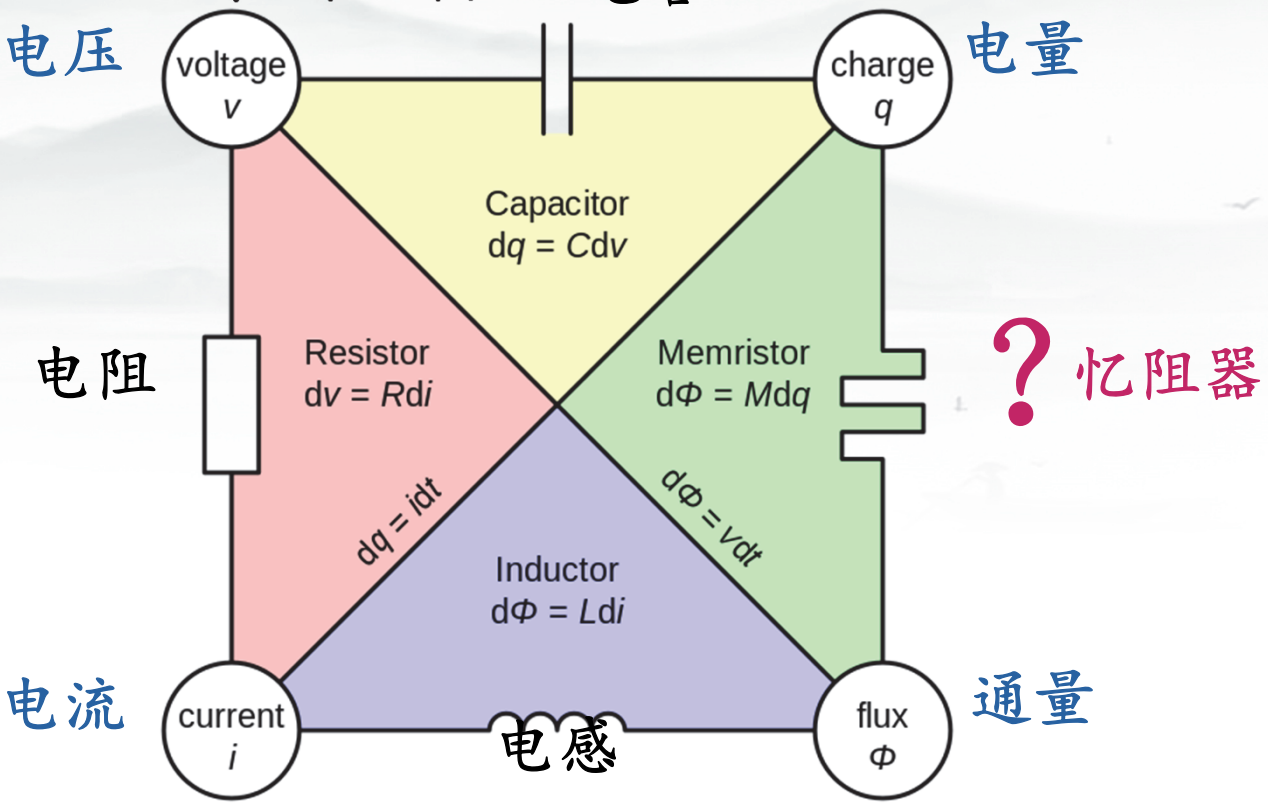
STT-RAM (Spin-Torque-Transfer RAM)

自旋扭矩转换-存储器



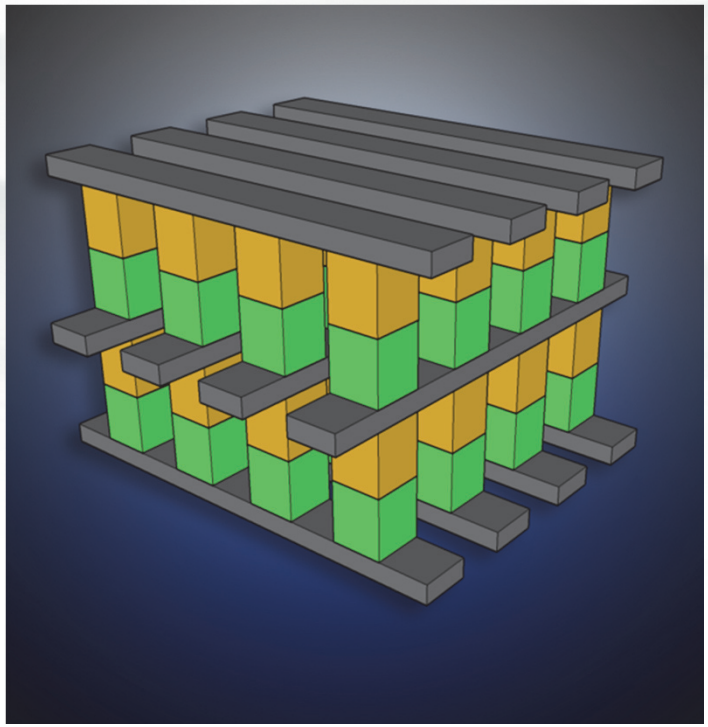
- 通过改变free layer的磁化方向，改变单元的电阻

Memristor：忆阻器



- 1971年，UC Berkeley蔡少棠(Leon Ong Chua)首先提出这种器件的假设
- 2008年，HP Labs发表了关于Memristor器件的论文
  - 通过施加不同方向的电流，可以改变电阻

3D XPoint



- 2015.7, Intel & Micron发布了3D XPoint技术
- 2016.4, IDF上演示比较了3D XPoint的SSD与Flash SSD的性能
- 2017年后期，3D Xpoint内存芯片的测试机
- 预期近期将有NVM内存条产品发售

NVM的特征

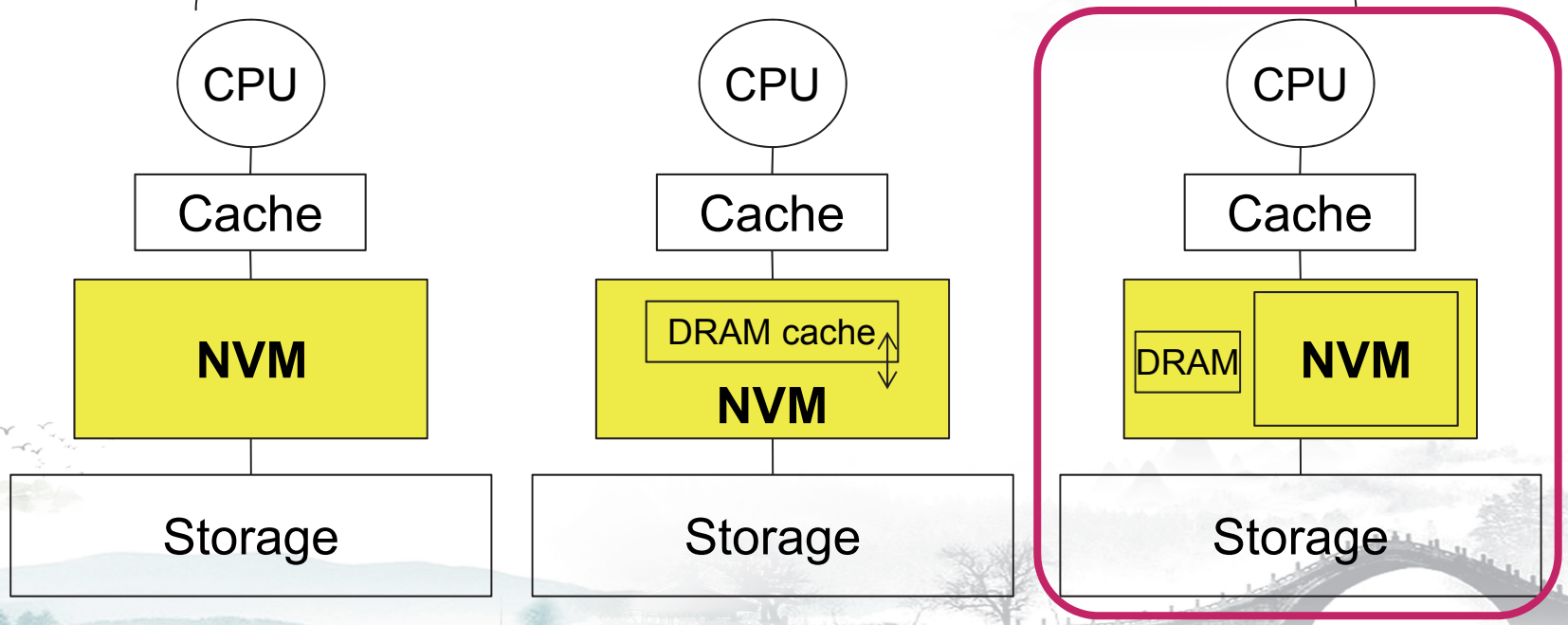
DRAM	Flash	HDD
10-100ns	10-100us	10ms

- 优点
  - 性能：接近DRAM
  - 粒度：接近DRAM，字节可寻址，可以覆盖写
  - 非易失：掉电不消失，不需要像DRAM周期性刷新
  - 容量：远大于DRAM，例如10倍
- 挑战
  - 比DRAM慢
  - 读写访问非对称（不同的物理过程）
    - 延迟、功耗、写寿命问题

NVM在系统中的位置

存储块设备
(Storage)

内存
(Memory)

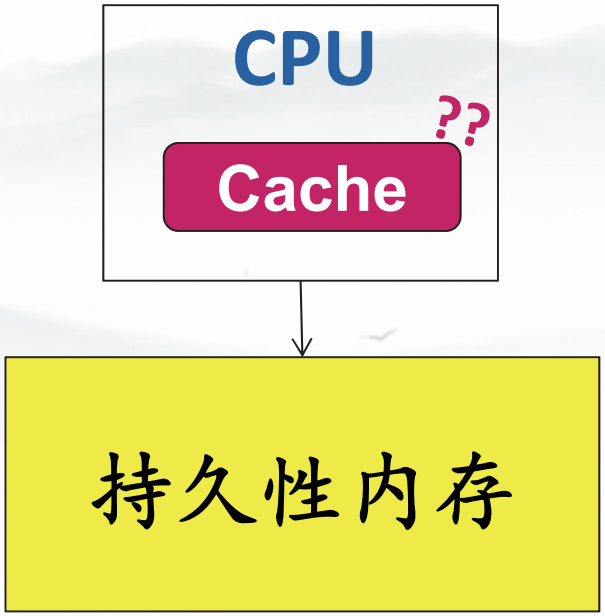


Persistent Memory（持久性内存）

- Persistent Memory (PM)：用于主存的NVM
- 为了支持PM
  - CPU的改变
  - 内存控制器的改变
  - 内存条接口定义
  - 操作系统的改变
  - 编程方式的改变

CPU的改变

- CPU Cache掉电后内容丢失
  - 从Cache → NVM基本由硬件控制
  - 写回的时机未知
  - 写回的顺序未知

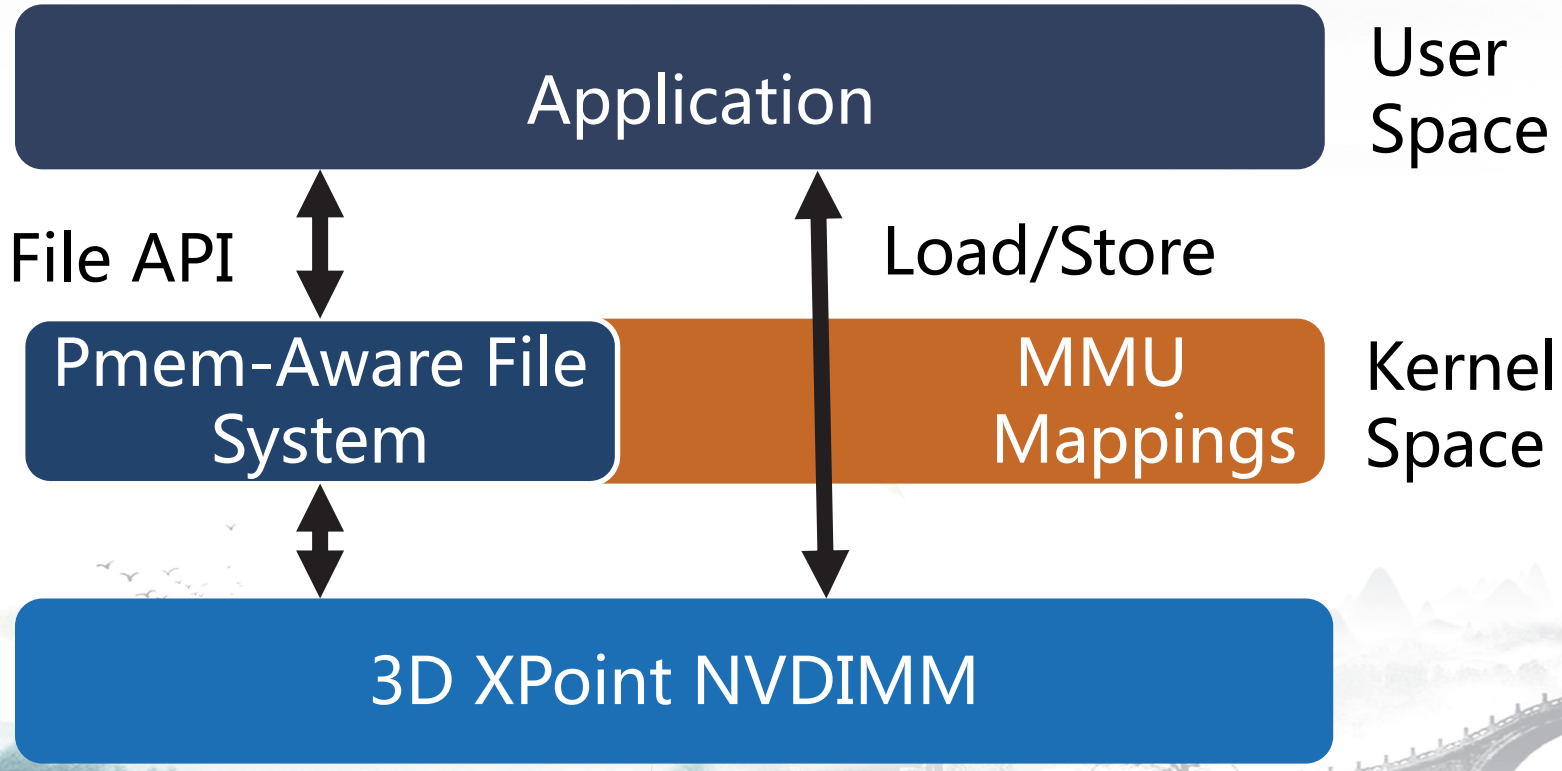


- 需要特殊指令保证数据写回PM
  - 强制写回Cache line: clflush, clwb等
  - 保证写顺序: sfence等

```
MOV X1, 10
MOV X2, 20
.
.
CLWB X1
CLWB X2
.
.
SFENCE
```

操作系统（Linux）DAX（Direct Access）

- 代替传统的Block Device Driver
- 直接返回NVM的物理地址



编程方式：PMDK

```
/* 打开文件，mmap到虚存空间，采用了DAX */
if ((pmemaddr = pmem_map_file(PATH, PMEM_LEN, PMEM_FILE_CREATE,
                              0666, &mapped_len, &is_pmem)) == NULL) {
        perror("pmem_map_file");
        exit(1);
}

/* 直接进行内存访问，与访问DRAM方法相同 */
strcpy(pmemaddr, "Hello, persistent memory!");

if (is_pmem) pmem_persist(pmemaddr, mapped_len); /* clwb, sfence */
else         pmem_msync(pmemaddr, mapped_len);

pmem_unmap(pmemaddr, mapped_len); /* munmap */
```


陈世敏（chensm@ict.ac.cn）

Outline

- 内存计算
- NVM
- 内存计算+NVM

内存计算遇到NVM

- 内存数据库系统
- 内存KV系统
- 内存大数据系统
- 内存图计算系统

NVM主存

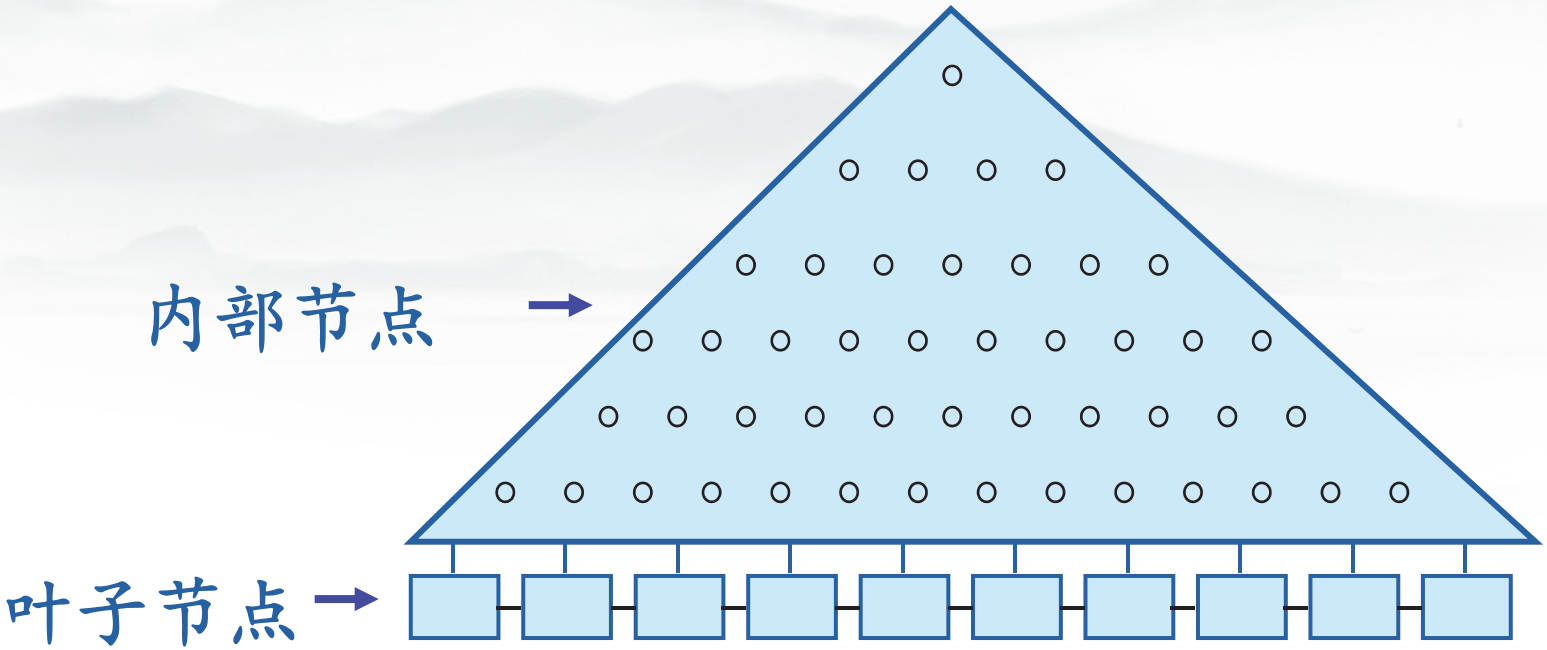
NVM新增的访存问题

- 读写非对称性问题
  - 如何尽量减少写？
  - 写分布均匀？
- 宕机一致性问题
  - 持久性数据结构
  - 如何保证其一致性？在突然掉电后，能够恢复？
- 访问性能问题
  - 如何更好地利用有限的DRAM实现更高的性能？

下面以B+-Tree为例

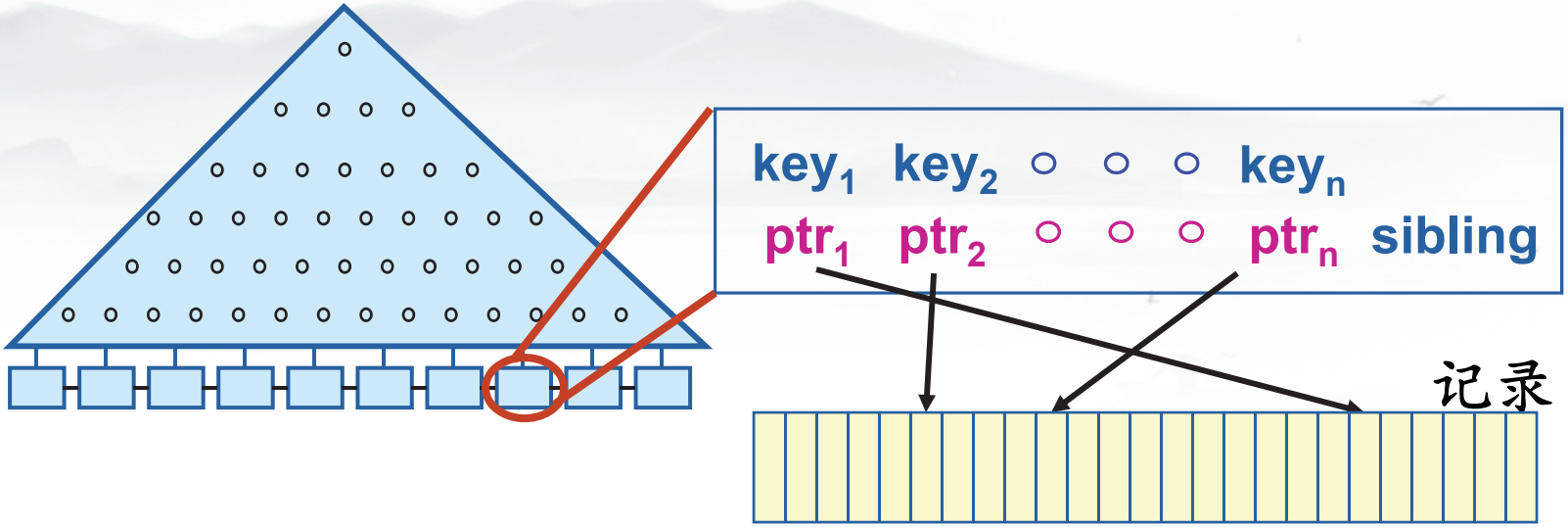
- 什么是B+-Tree?
- NVM上的B+-Tree

B+-Trees



- 每个节点是一个page
- 所有key存储在叶子节点
- 内部节点完全是索引作用

叶子节点

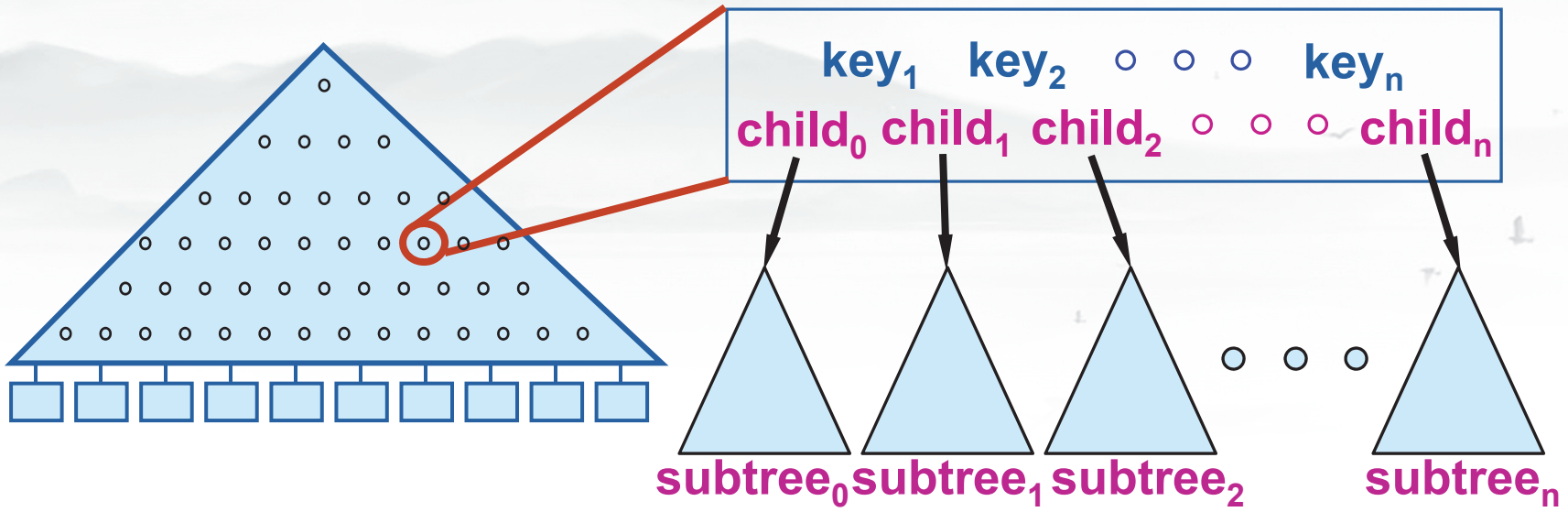


Keys 按照从小到大顺序排列: $key_1 < key_2 < \cdots < key_n$

叶节点自左向右也是从小到大排序，以sibling pointer链起来

(ptr= record ID; sibling = page ID)

内部节点

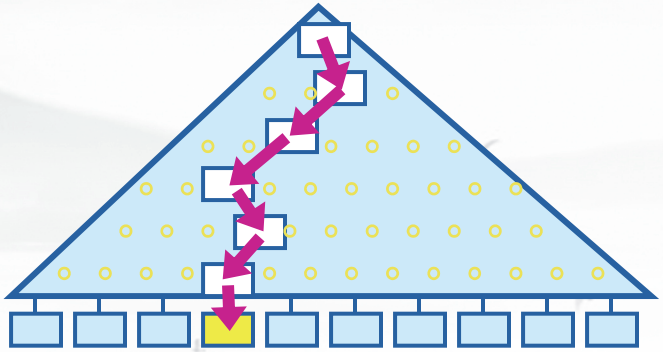


$$subtree_0 < key_1 \leq subtree_1 < key_2 \ldots < key_n$$

B+-Tree: Search

Search:

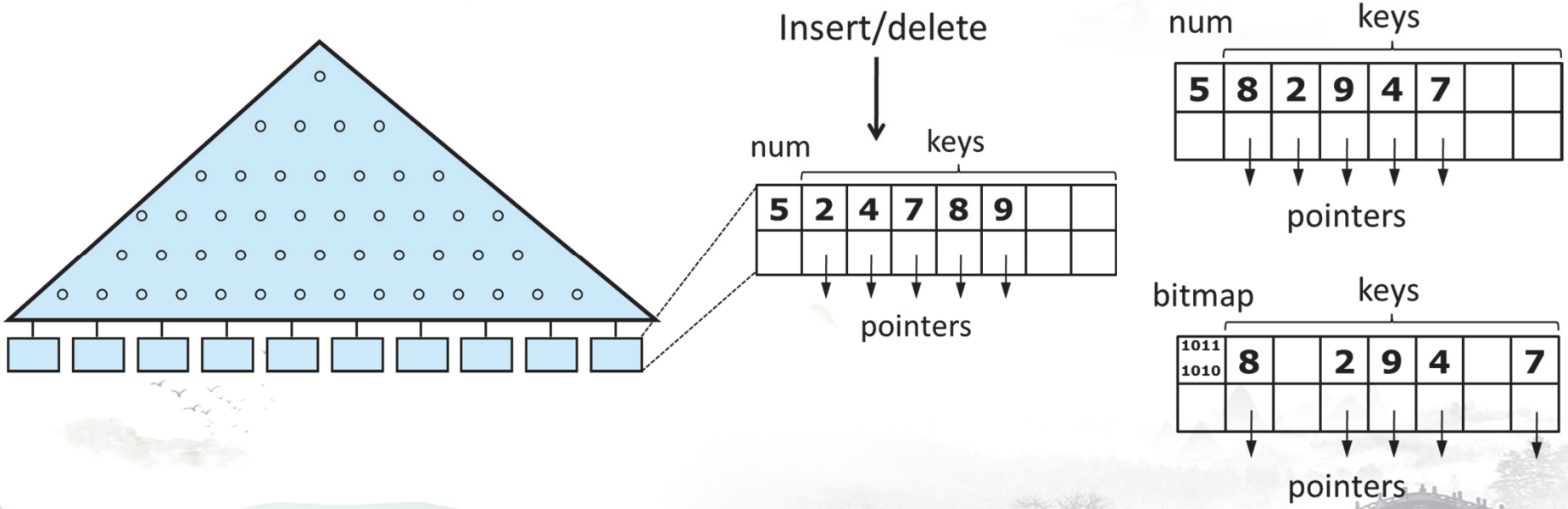
- 从根节点到叶节点
- 每个节点中进行二分查找
  - 内部节点：找到包括search key的子树
  - 叶节点：找到匹配



PCM-Friendly B+-Tree

[Chen et al. CIDR'11]

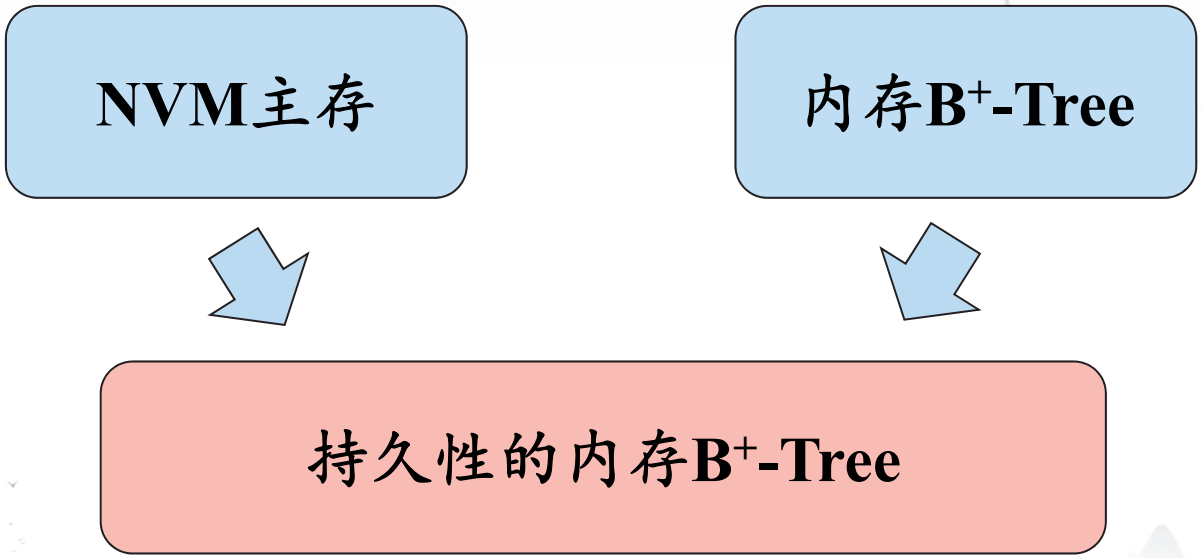
- 问题: insertion/deletion in sorted nodes
  - 引起大量的写操作，而写操作在NVM可能比读慢10倍!
- 提出： unsorted nodes



Persistent B+-Trees in Non-Volatile Main Memory

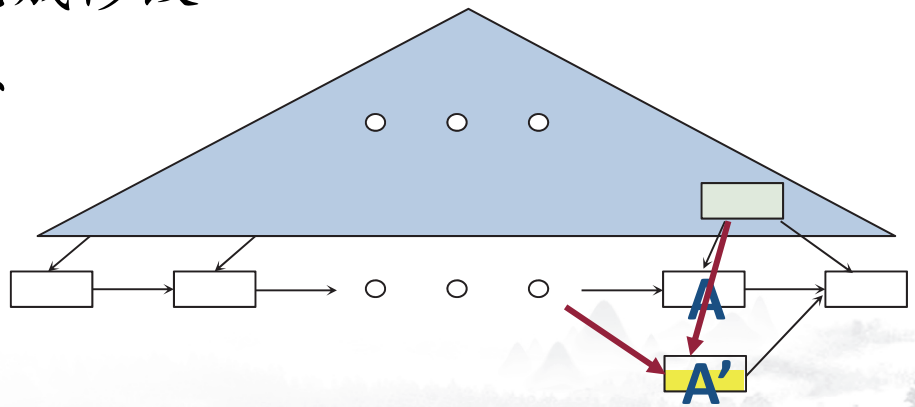
[Chen et al. VLDB'15]

- 在前面工作基础上，希望B+-Tree持久化
- 解决宕机一致性问题



现有方案会引起大量NVM写或特殊指令

- 写前日志（Write-Ahead Logging）
  - 8B的NVM写➔记录日志（8B地址，8B旧值，8B新值）
  - 特殊指令强制写日志到NVM
- 影子副本（Shadowing, Copy-on-Write）
  - 先拷贝一个叶子节点
  - 在新拷贝的叶子节点上完成修改
  - 特殊指令强制写叶子节点
  - 换指针



Write-Atomic B$^+$-Tree

- 我们的方案：Write-Atomic B$^+$-Trees (wB$^+$-Trees)
  - 改进B$^+$-Tree节点的结构，通常的插入、删除操作可以用atomic write完成，从而避免logging或shadowing开销
  - 同时保持B$^+$-Tree良好的查询性能

- 实验结果
  - 多种设计，定长与变长的键值类型，不同种类的NVM
  - 插入/删除性能与Logging和Shadowing方案相比
    - PCM (采用模拟)： wB$^+$-Tree提高性能**1.5 ~ 27.1**倍
    - DRAM-like NVM (在真实机器上运行)： wB$^+$-Tree提高性能**1.2 ~ 8.8**倍
  - 查找性能与cache-optimized B$^+$-tree相似

总结

- Moore’s law推动了内存计算
  - 已经遍地开花
- Moore’s law向下扩展的要求推动了新一代NVM技术
  - 多种技术：PCM, STT-RAM, Memristor, 3D Xpoint等
    ☒存储电荷 ☑改变单元的电阻
  - 非易失，字节寻址，访问速度接近DRAM，读写非对称
- 对内存计算系统带来了新的挑战和机遇
  - 机遇：近似DRAM + 大容量 + 非易失
  - 挑战：比DRAM慢 + 读写非对称 + 宕机一致性

HardBD&Active'19：大数据系统+新硬件

- Workshop在ICDE 2019（澳门）举行
- http://www.carch.ac.cn/~ictdb/HardBD-Active-2019/
- 截稿日期：2019/1/18

HardBD&Active'18 @
ICDE 2018, Paris

CNCC

谢谢！