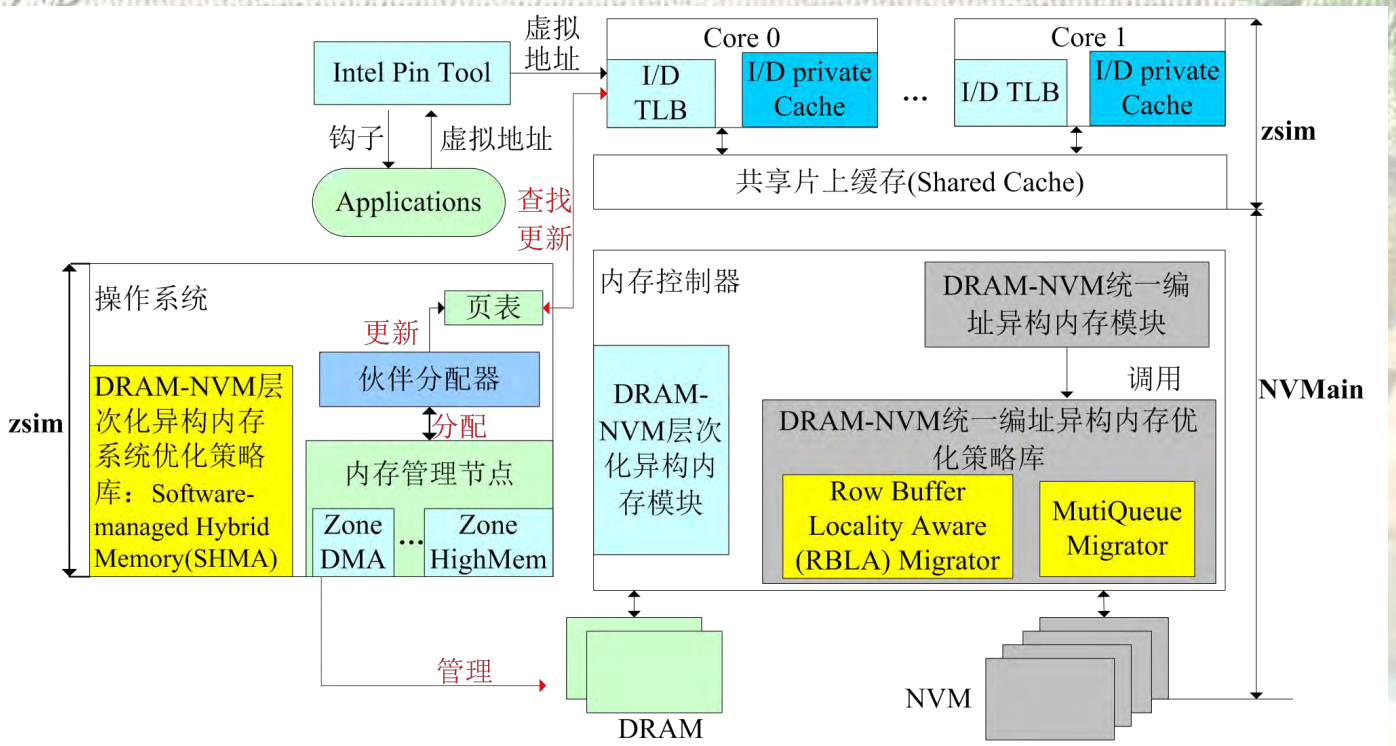
混合内存计算系统模拟器和仿真器

基于Zsim和NVMain的全系统混合模拟器，支持对多种存储介质和存储架构的模拟



https://github.com/CGCL-codes/SHMA

两种架构：

1.DRAM/NVM统一编址异构内存架构

2.DRAM/NVM层次化异构内存架构

集成多种策略库：

1. 基于DRAM-NVM平行异构内存架构策略库

（1）考虑到行缓冲区局部性的（*RBLA*）动态页迁移策略

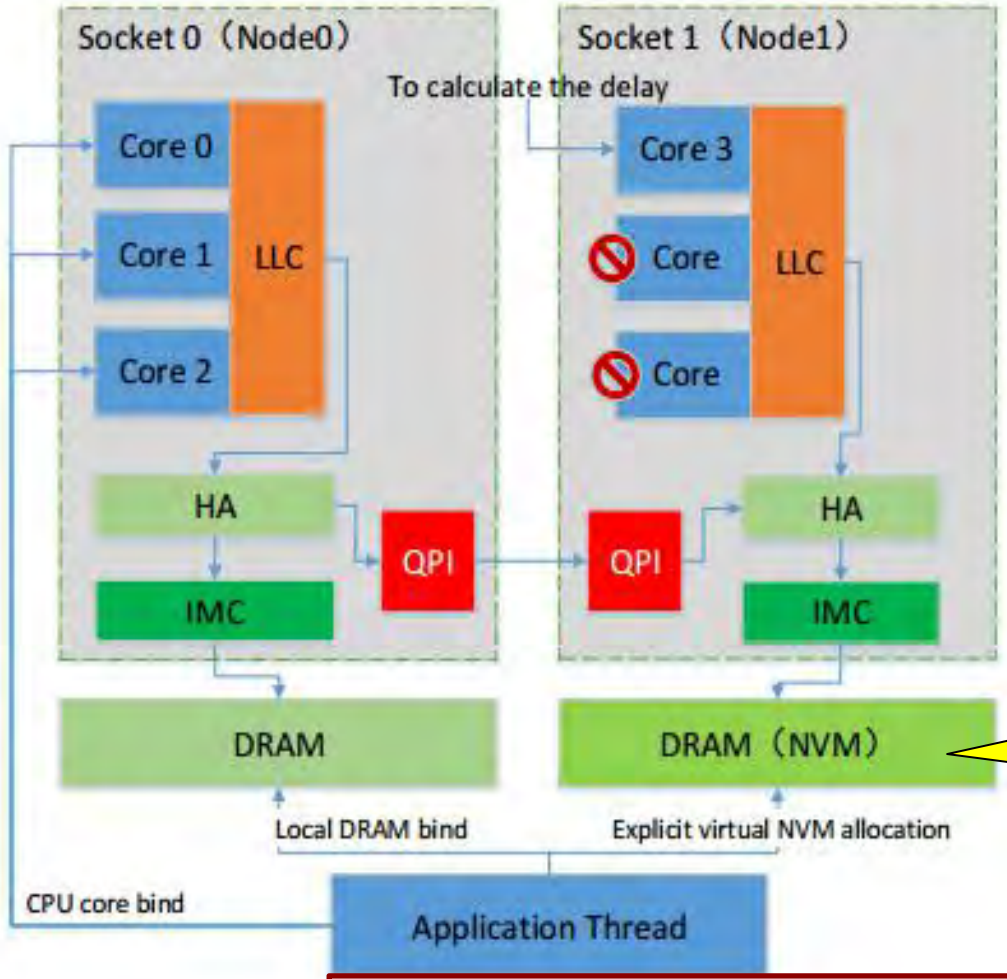
（2）基于多级队列的(*MQ*)动态热页迁移策略

2. 基于DRAM-NVM层次化异构内存架构策略库： 软件管理的DRAM缓存

HME：轻量级的异构内存仿真器

基于NUMA架构的异构内存仿真环境

https://github.com/CGCL-codes/HME

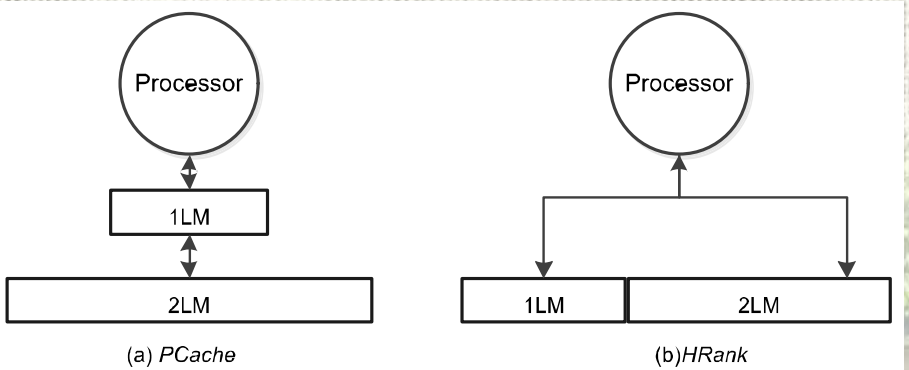


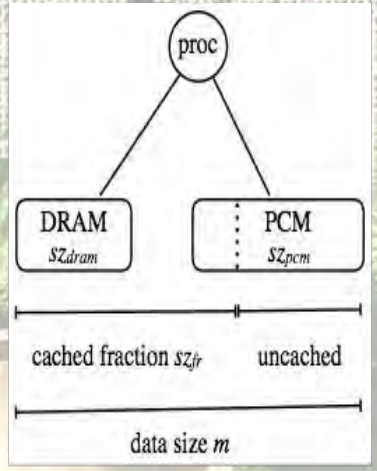
- **延时模拟**：统计访问远端内存的读写计数，通过IPI核间中断来给执行程序的本地core注入空转延迟
- **带宽模拟**：限制远端channel上内存控制器中单位时间内的内存请求数量来进行节流

通过软件注入延迟的方式将远端节点的DRAM模拟成NVM

“HME: A Lightweight Emulator for Hybrid Memory”, DATE 2018

Fraction Cache: Memory Equalizer for Lateral Management of Heterogeneous Memory



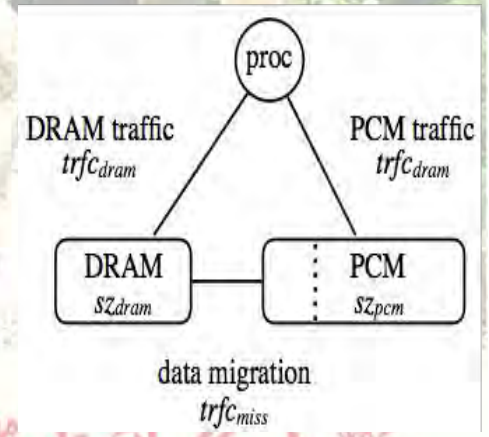


Fraction Cache

- Two level exclusive cache
- DRAM caches only a fraction of program data
- Three parameters: DRAM size, PCM size, cached fraction

Management

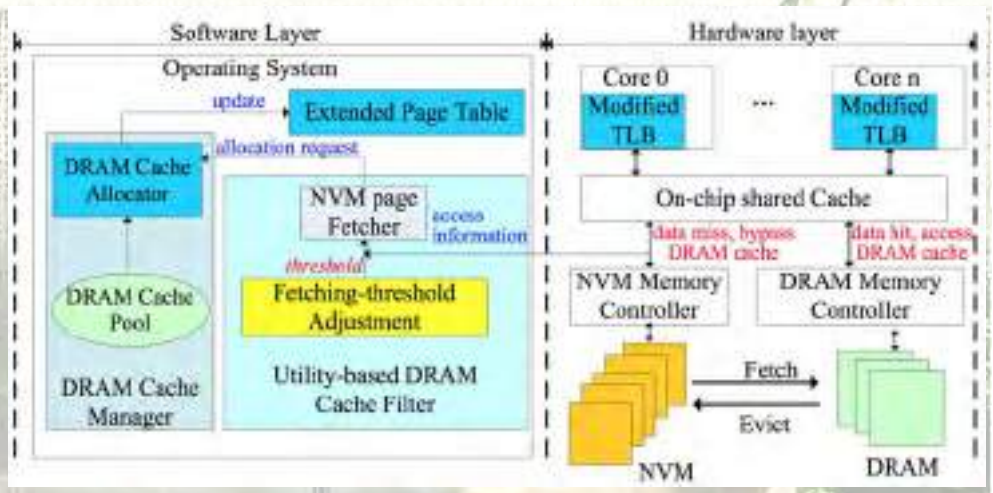
- Performance parameters: traffic between core and DRAM, core and PCM, DRAM and PCM
- Users specify performance objectives, leave the tuning and optimization to an automatic tool
- As an demonstration, we use fraction cache to minimize the need for DRAM while limiting the cost of data migration between DRAM and PCM.

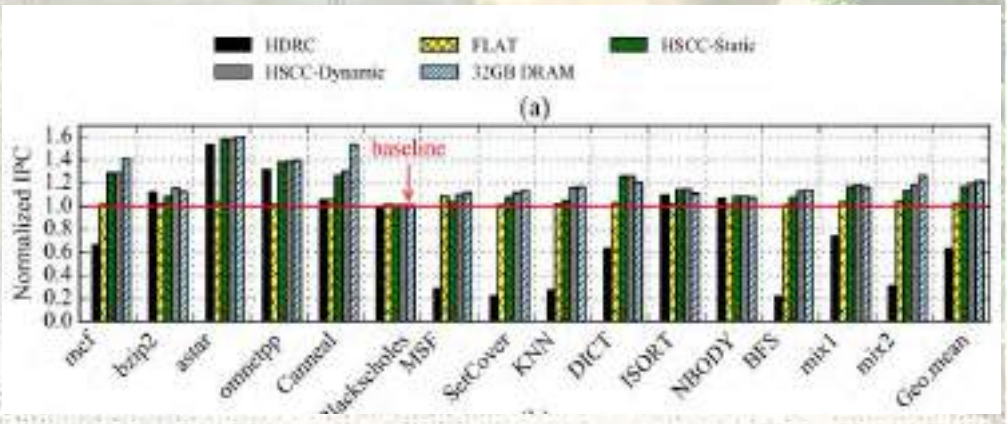


"Memory Equalizer for Lateral Management of Heterogeneous Memory", MEMSYS 2017

软硬件协同管理的混合内存架构

- 硬件支持的层次架构存在的问题
  - 硬件开销大，灵活性低
  - 缓存策略忽略数据热度，易造成缓存污染
  - 组相连结构导致DRAM利用率不高
- 解决方案：在平行架构上通过软硬件协同设计逻辑上实现层次架构
  - 扩展TLB和页表结构，维护DRAM到NVM的直接映射
  - 软件管理DRAM，实现全相联的Cache
  - 基于效用的DRAM Cache过滤机制
  - 支持DRAM-bypass



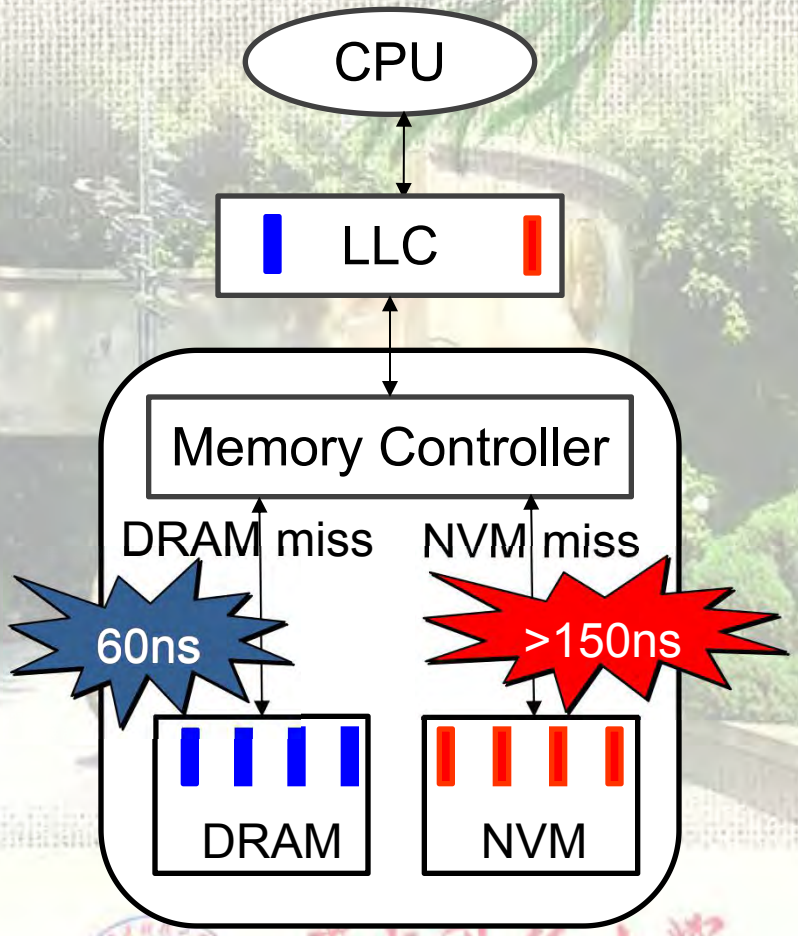


实验结果：对比硬件管理的DRAM Cache层次架构，可以提高某些应用（MSF）性能达4.7倍，平均提高93%，对比平行架构，可以提高某些应用性能达55%，平均21%。

“Hardware/Software Cooperative Caching for Hybrid DRAM/NVM Memory Architectures”, ICS 2017

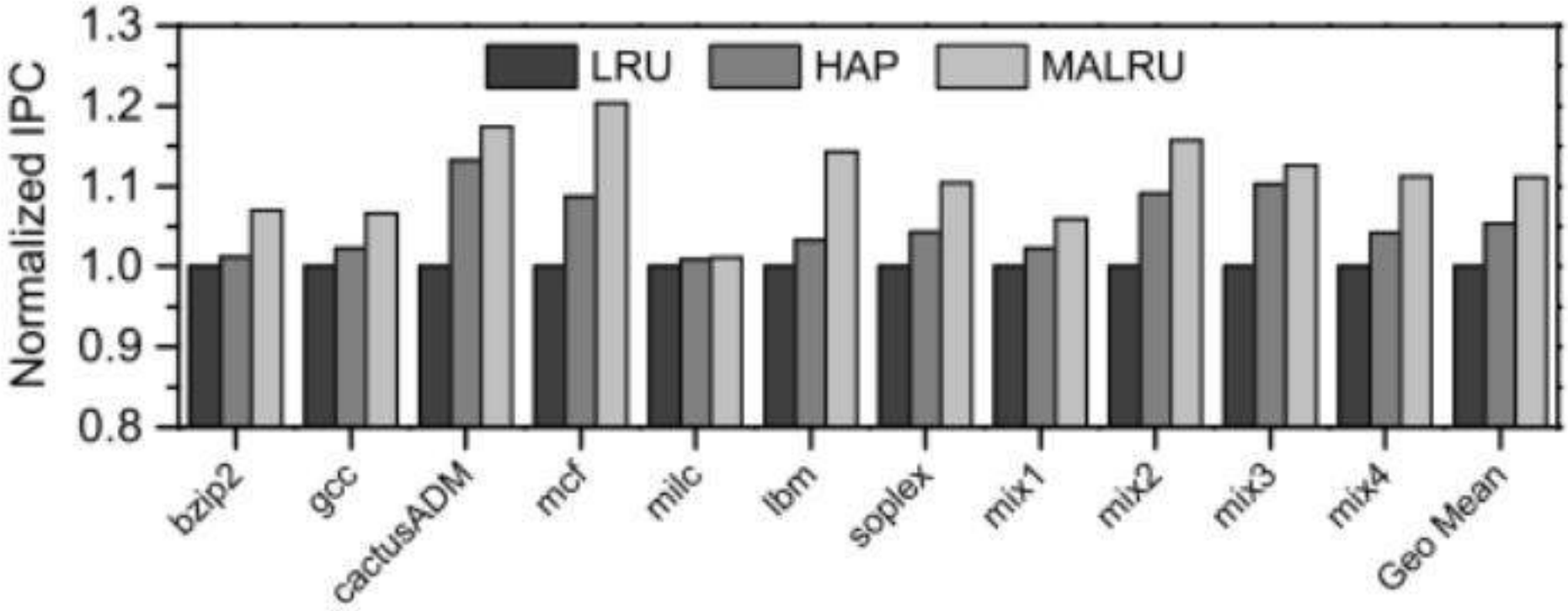
混合内存中时延感知的缓存替换策略

- 问题：传统基于命中率的Cache替换算法在混合内存中效率低下
- 根源：异构内存架构下，LLC缺失代价的不对称性
- 思路：Cache替换时兼顾局部性和替换代价，时延感知的Cache替换策略（MALRU）
- 提出新的评价存储系统性能的通用指标：平均访存时延Average Memory Access Time (AMAT)
- Cache替换原则：（1）尽量使NVM数据块更久地驻留在LLC中；（2）尽量使访问频繁的DRAM数据块驻留在LLC中。



混合内存中时延感知的缓存替换策略

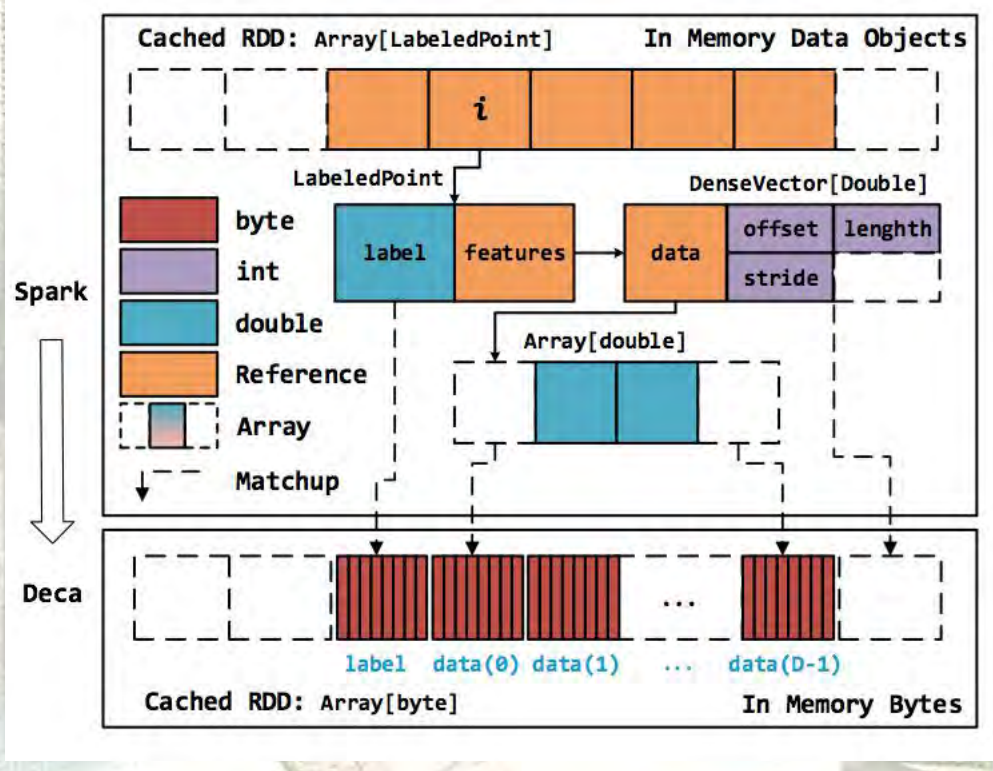
- MALRU相比HAP最高有10.7%的性能提升，平均为5.7%
- MALRU相比LRU最高有22.8%的性能提升，平均为11.1%
- 对于cache thrashing的应用，MALRU提高了NVM的命中；对于recency-friendly的应用，MALRU降低了平均的不命中开销



"MALRU: Miss-penalty Aware LRU-based Cache Replacement", DATE 2017

Deca:基于数据生命期的内存计算大数据处理系统

- 自动分析用户自定义函数和数据类型，获得数据对象的生命周期，根据生命周期来分配和释放内训空间，减少因垃圾回收所带来的开销
- 系统按照相似的生命周期透明地分解和组合数据对象，对于同一生命周期的对象一起释放内存空间
- 与Apache Spark相比，可加速22.7倍到41.6倍

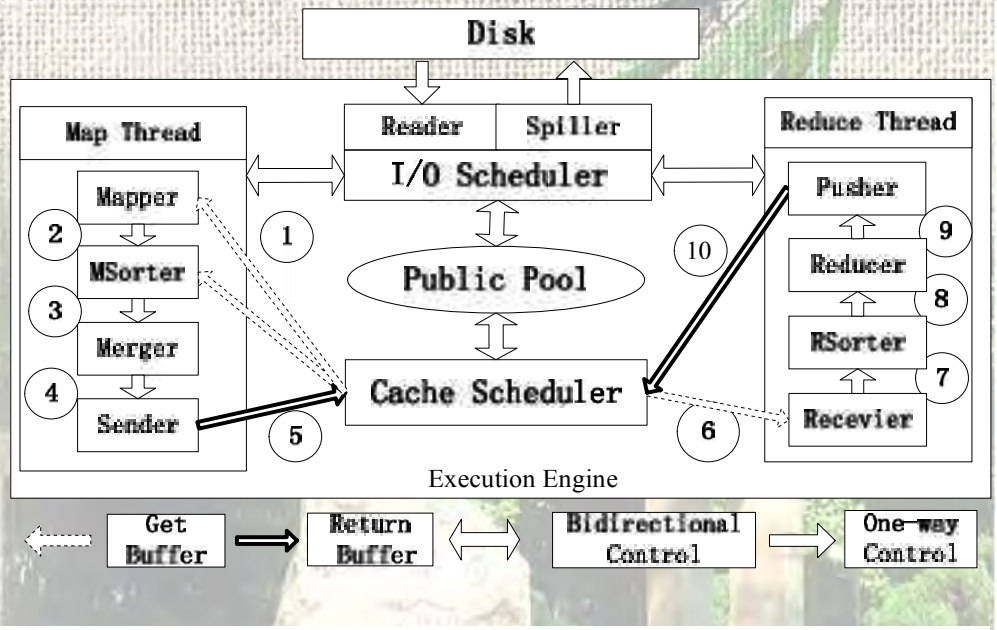


App	Spark			Deca	
	exec.	gc	ratio	gc	reduction
WC: 150GB	4980s	2016s	40.5%	12.2s	99.4%
LR: 80GB	2820s	2069.9s	73.4%	2.5s	99.9%
KMeans: 80GB	5443s	4294.8s	78.9%	7.2s	99.8%
PR: 20GB	1434s	517.9s	36.1%	85.4s	83.5%
CC: 20GB	2320s	919.2s	39.6%	37.6s	95.9%

"Lifetime-Based Memory Management for Distributed Data Processing Systems", VLDB 2016

Mammoth: 以内存为中心的MapReduce系统

- 以内存为中心调度数据，细粒度分配内存，以内存为缓冲串行化硬盘访问
- 研发并开源了系统Mammoth，在国际开源社区ASF以及Github开源
- 效果：比国际上主流的分布式数据处理系统Hadoop快2-5倍

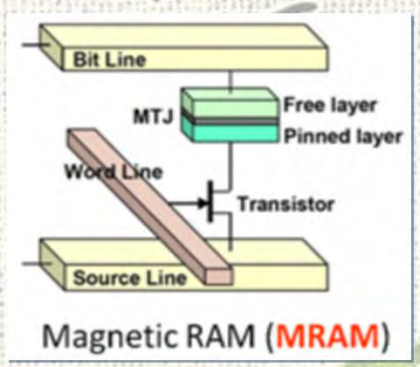


Application	Running Time		Speedup
	Mammoth	Hadoop	
PageRank	2760s	13090s	4.74x
Concmpt	2380s	12364s	5.19x
DegDist	438s	737s	1.68x
Radius	3365s	12250s	3.64x
CloudBurst	1395s	2313s	1.66x

"Mammoth: Gearing Hadoop Towards Memory-Intensive MapReduce Applications", IEEE TPDS 2015

Landscape of Disk-Based and In-Memory Data Management Systems (2014)

总结：内存计算的发展阶段



萌芽阶段

- 探讨是否能用

- 介质的研制
- NVM结构设计
- 模拟器/策略库

探索阶段

- 探讨如何使用

- 缓存管理
- 数据管理
- 编程模型

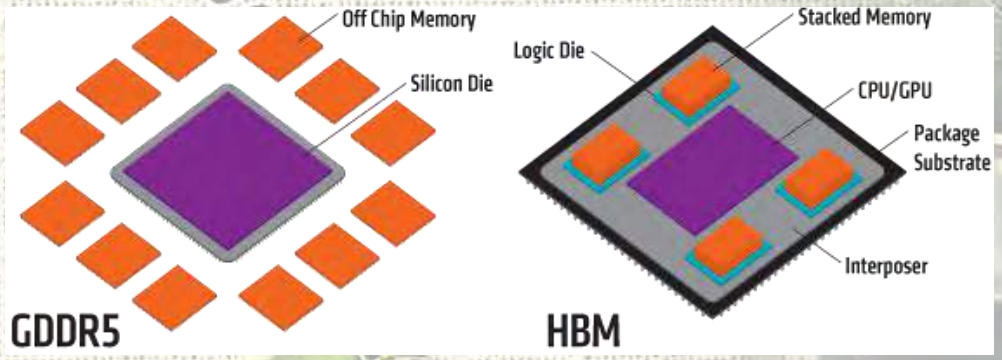
实用阶段

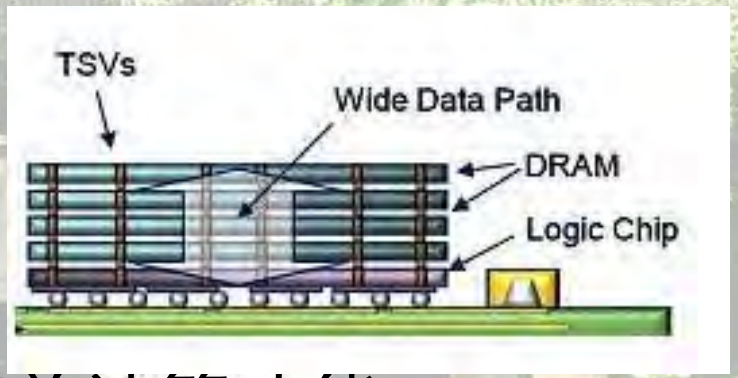
- 探讨提升应用性能

- 应用敏感的内存管理
- 与AI/大数据等应用的结合

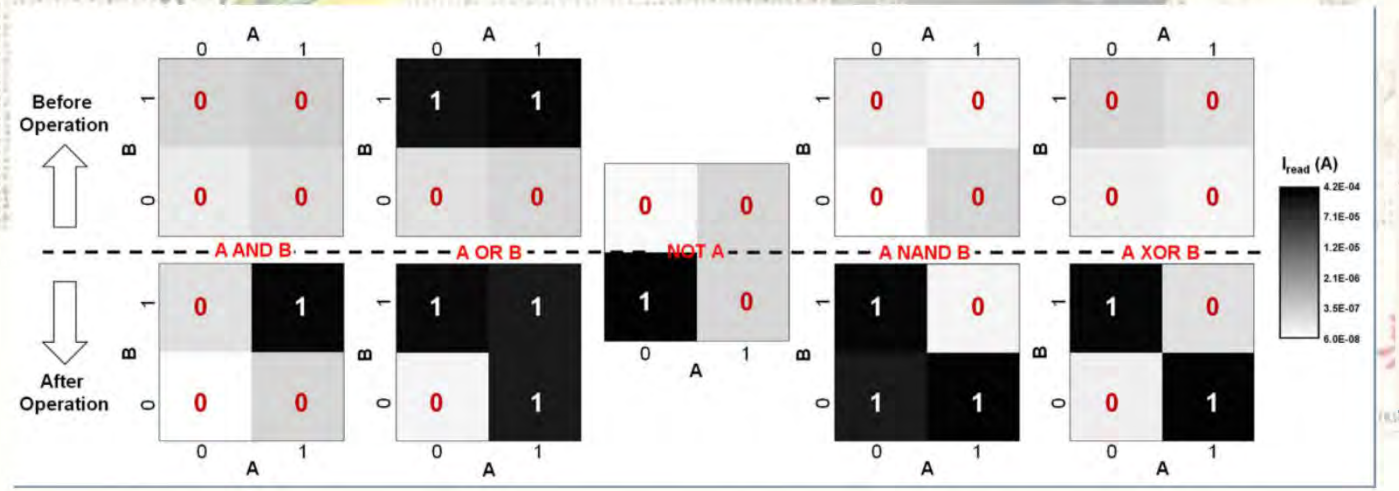
内存计算发展趋势：存算一体

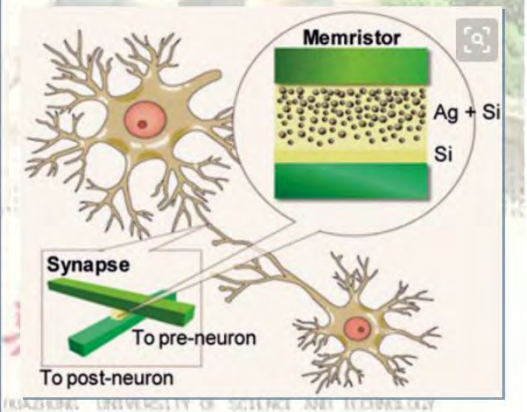
- 利用3D堆叠技术，CPU 核心和内存放在一块芯片上
  - HBM/HMC比DDR内存带宽提高~10倍，延迟和能耗更低





- 利用NVM的存储原理，实现简单计算功能





忆阻器(memristor)


谢谢！