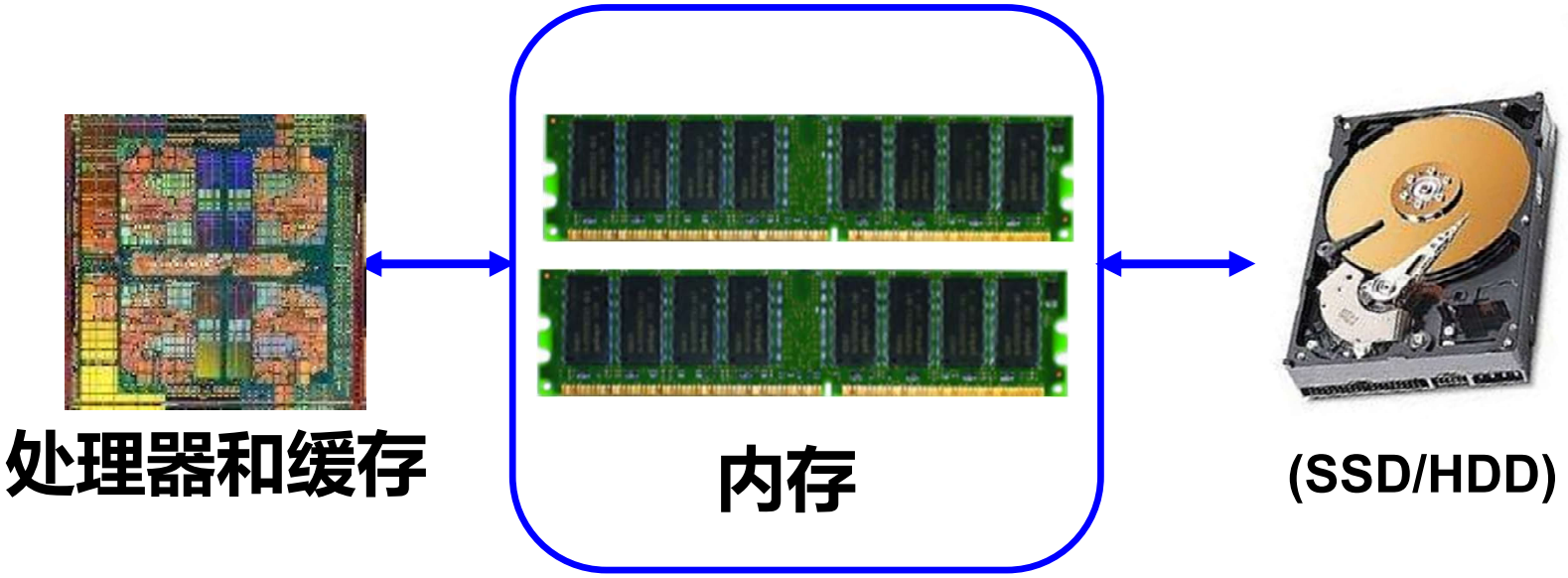
内存 (Main Memory)

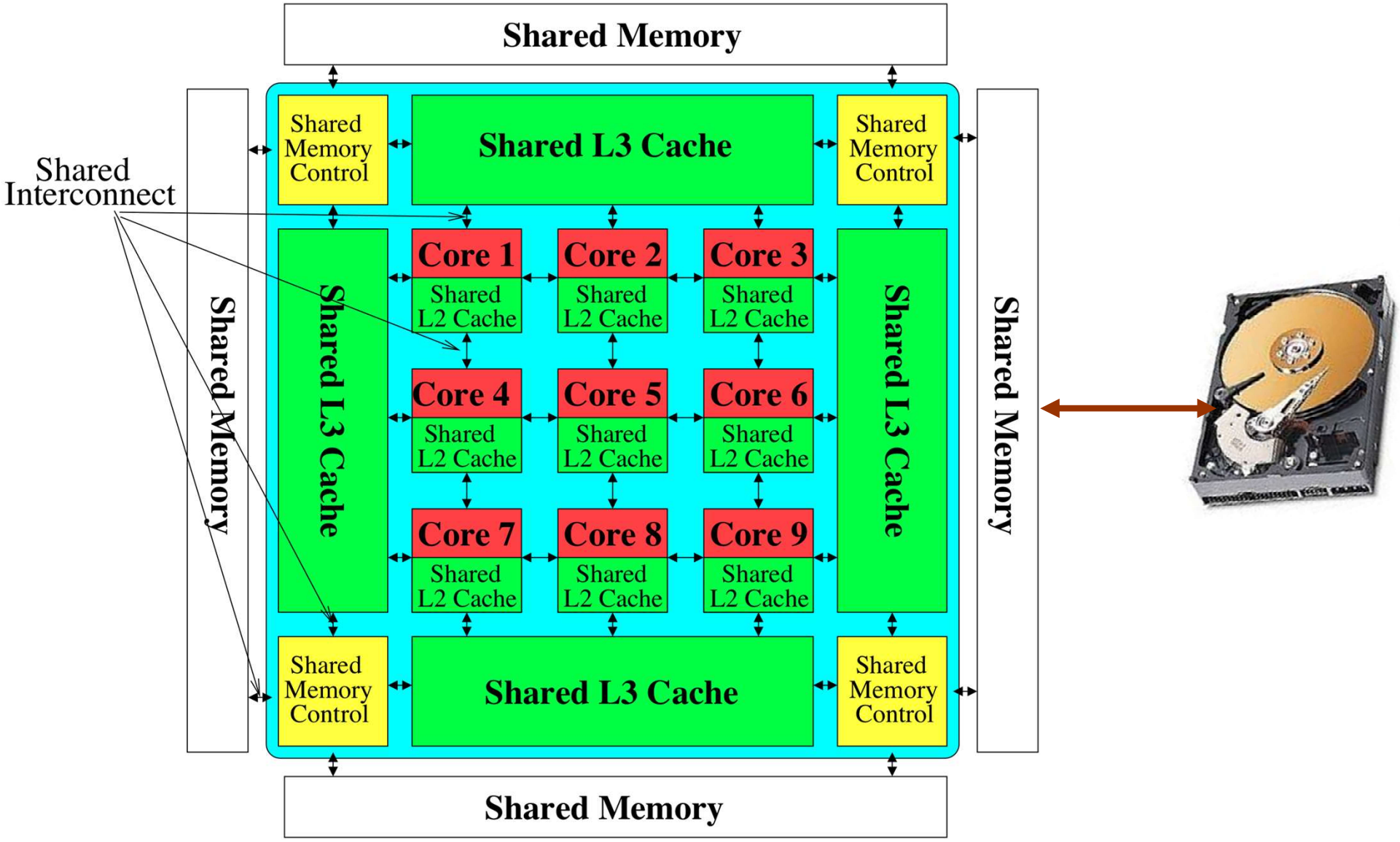
Sep. 25, 2025

内存系统



- 内存（主存）是所有计算系统的关键组件：服务器、移动设备、嵌入式系统、台式机、传感器。
- 内存系统必须具备可扩展性（在尺寸、技术、效率、成本和管理算法方面），才能保持性能增长和技术扩展优势。

内存系统：共享资源视角


Shared Memory
Shared Memory Control
Shared L3 Cache
Shared Memory Control
Shared Interconnect
Shared Memory
Core 1
Core 2
Core 3
Shared L2 Cache
Shared L2 Cache
Shared L2 Cache
Shared L3 Cache
Shared Memory
Core 4
Core 5
Core 6
Shared L2 Cache
Shared L2 Cache
Shared L2 Cache
Core 7
Core 8
Core 9
Shared L2 Cache
Shared L2 Cache
Shared L2 Cache
Shared Memory Control
Shared L3 Cache
Shared Memory Control
Shared Memory

内存系统的现状

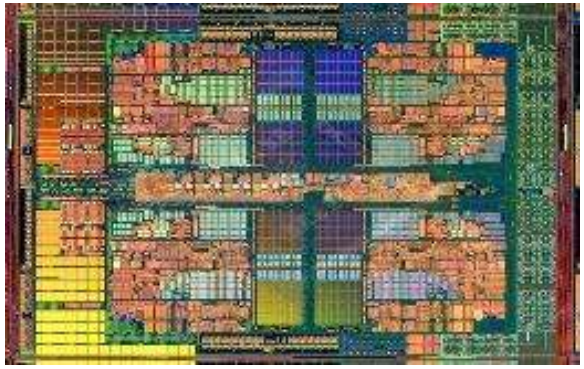
- 最近的技术、架构和应用趋势：
  - 导致了新的需求
  - 加剧了旧有的需求
- 目前的 **DRAM** 和内存控制器，不太可能满足所有的需求。
- 一些新兴的非易失性内存技术（例如，**PCM**）带来了新的机会：内存与存储的融合。
- 我们需要重新思考/重新设计内存系统：
  - 解决 DRAM 的问题，并支持新兴技术
  - 满足所有需求

影响内存的主要趋势（I）

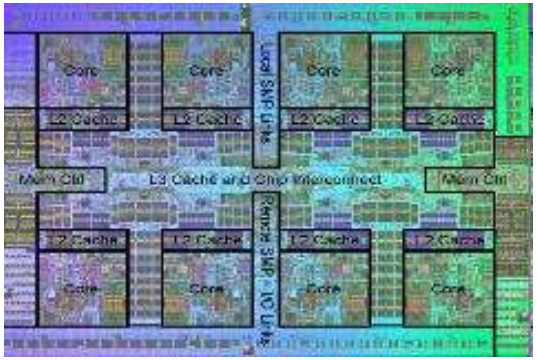
- 对主内存容量、带宽、**QoS**（服务质量）的需求日益增加
- 主内存的能量/功耗成为关键系统设计问题
- **DRAM** 技术的扩展已接近结束

对内存容量的需求

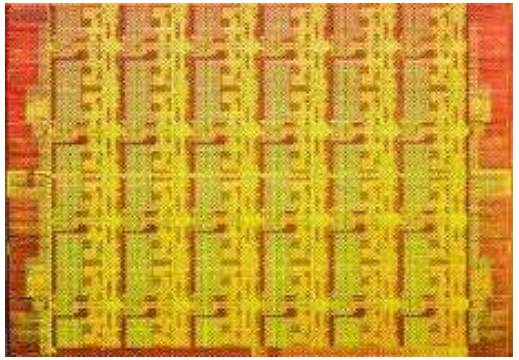
- 更多核心 → 更多并发 → 更大的工作集



AMD Barcelona: 4 cores



IBM Power7: 8 cores

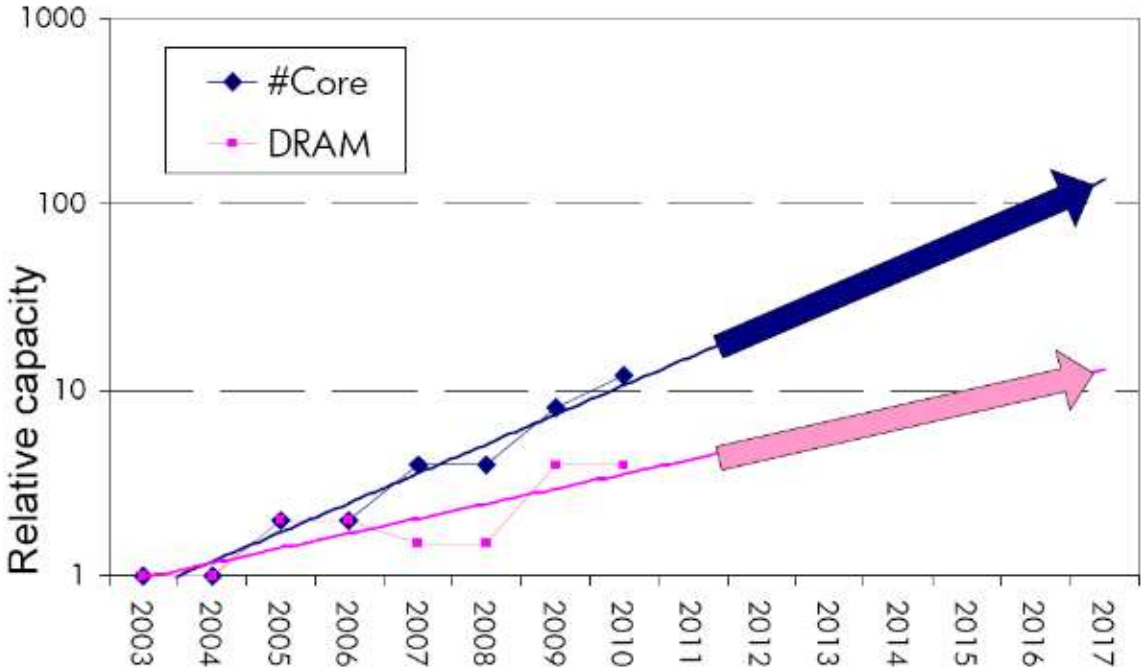


Intel SCC: 48 cores

- 现代应用程序（越来越）数据密集
- 许多应用程序/虚拟机（将）共享内存
  - 云计算/服务器：通过整合提升效率
  - 移动设备：交互式与非交互式应用的整合
  - 。。。

内存容量差距

- 核心数每两年翻一番
- DRAM容量每三年翻一番



- 每核心的内存容量预计每两年减少30%
- 每核心的内存带宽趋势更差

影响内存的主要趋势（II）

- 对内存容量、带宽、QoS（服务质量）需求增加
  - 多核： 核心/代理数量增加
  - 数据密集型应用：对数据的需求/渴望增加
  - 整合： 云计算、GPU、移动设备、异构性

影响内存的主要趋势（III）

- 对内存容量、带宽、**QoS**（服务质量）需求增加

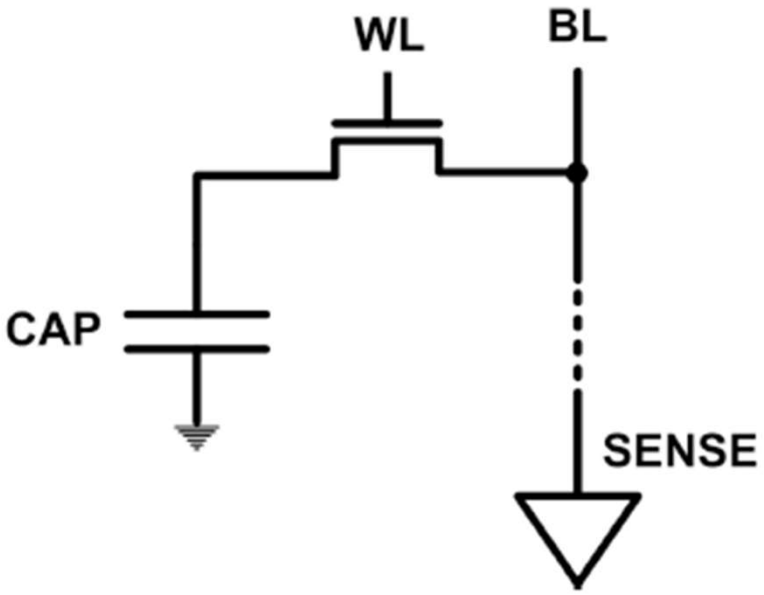
- 内存的能量/功耗是关键的系统设计问题
  - IBM 服务器：大约 50% 的能量消耗用于外部内存层次结构【Lefurgy，IEEE Computer 2003】
  - DRAM 在空闲时消耗功率，并且需要周期性刷新

影响内存的主要趋势（III）

- 对内存容量、带宽、**QoS**（服务质量）需求增加
- 内存的能量/功耗是关键的系统设计问题
- **DRAM** 技术的扩展即将结束
  - ITRS 预测 DRAM 很难在 X nm 以下继续扩展
  - 扩展带来了许多好处：更高的容量、更高的密度、更低的成本、较低的能耗

DRAM 扩展问题

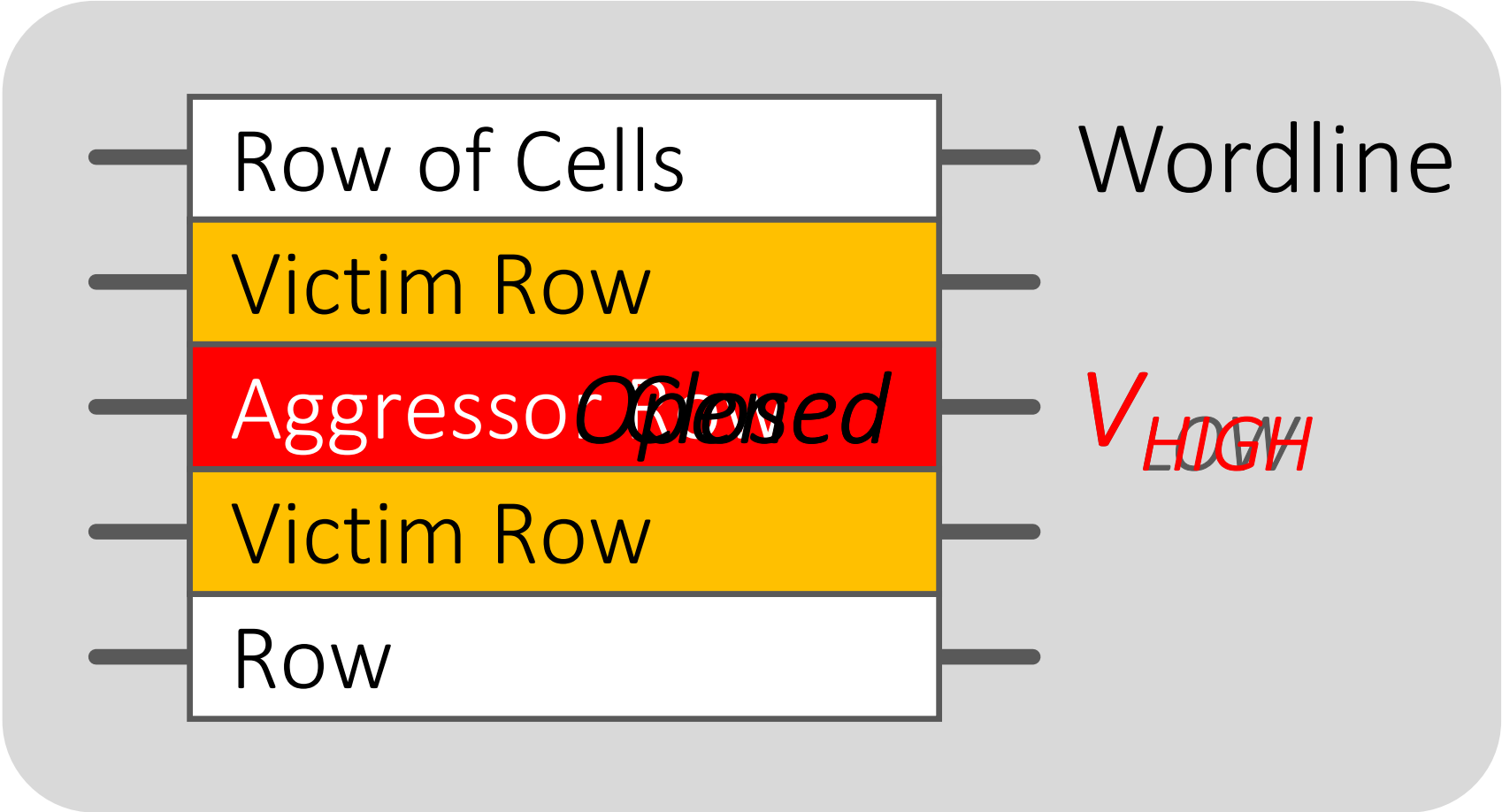
- DRAM 在电容器中存储电荷（基于电荷的内存）
  - 电容器必须足够大，以确保可靠的感应
  - 存取晶体管应足够大，以确保低泄漏和较长的保持时间
  - 在 40-35nm 以下的扩展是具有挑战性的【ITRS，2009】



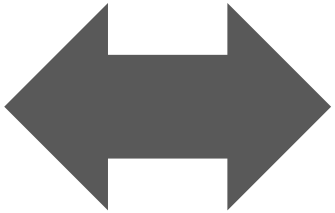
- DRAM 的容量、成本和能效难以扩展

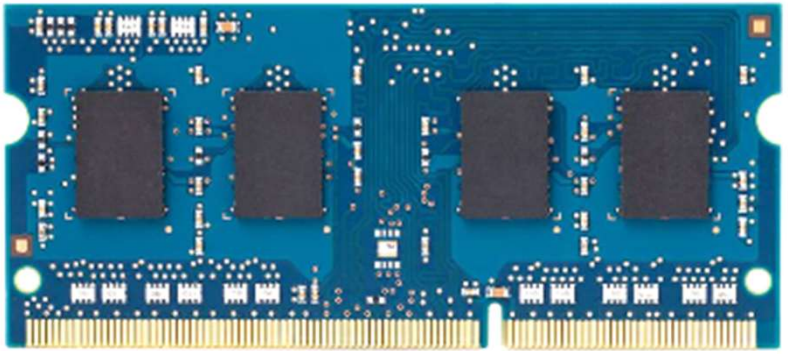
DRAM 扩展问题的证据

- 在刷新间隔内反复开启和关闭一行足够多次，会导致大多数现有 DRAM 芯片中相邻行的干扰错误【Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014】。

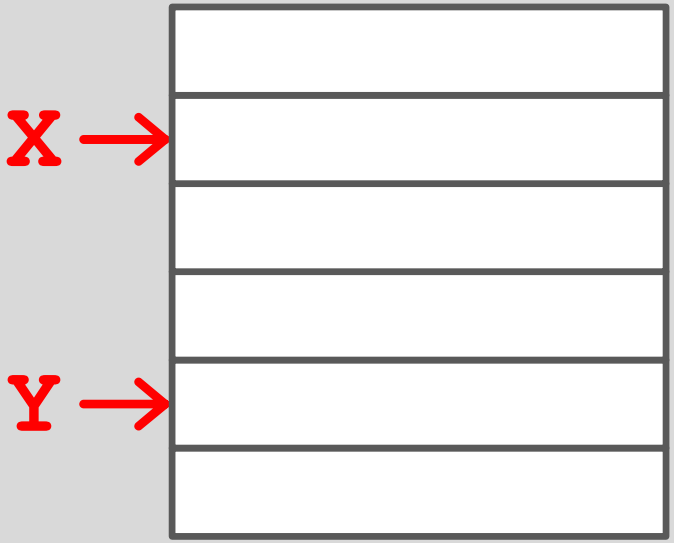


x86 CPU DRAM Module

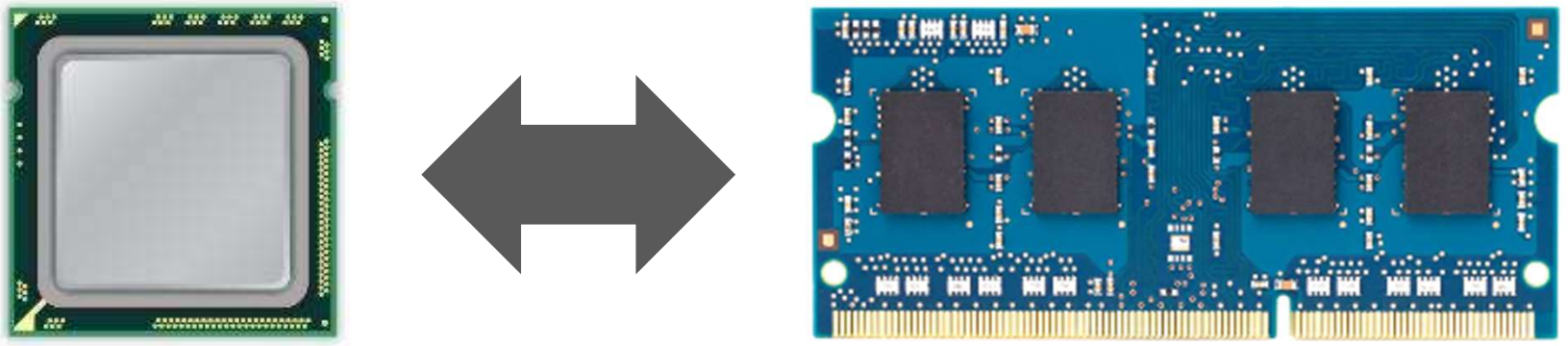




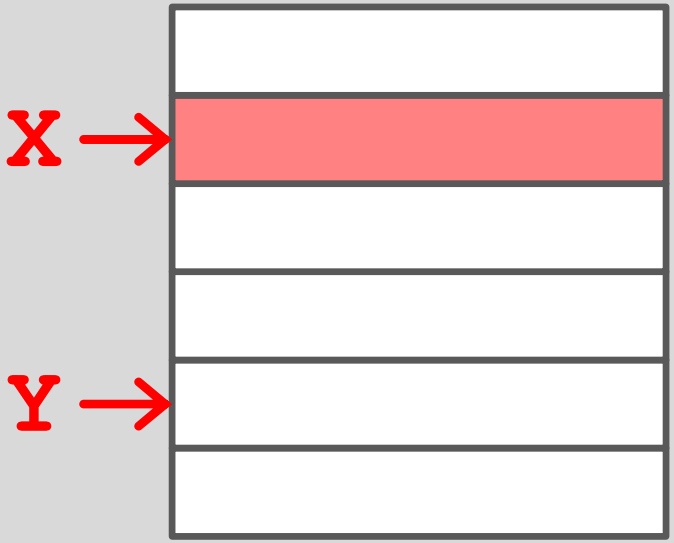
```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



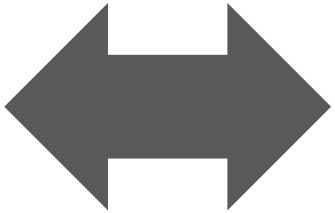
x86 CPU DRAM Module

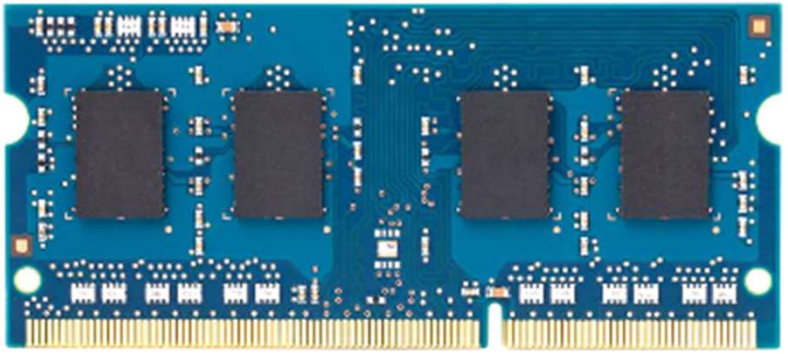


```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

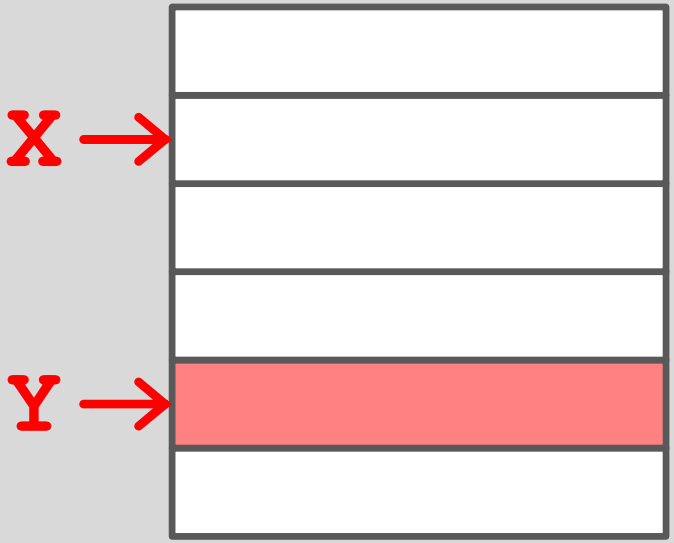


x86 CPU DRAM Module

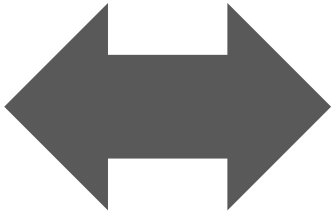


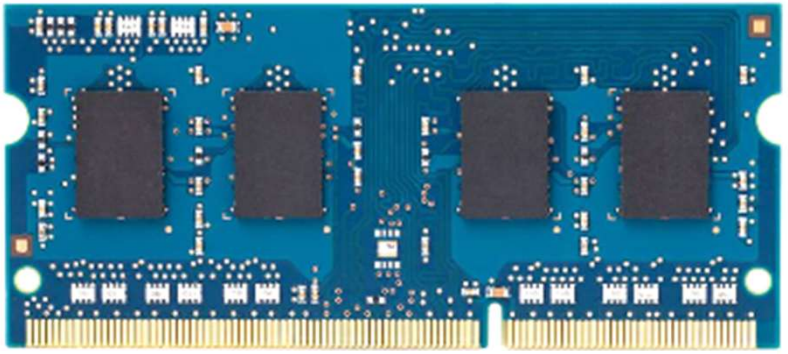


```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```



x86 CPU DRAM Module





```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```