第 11 讲：Scalable Synchronization on Shared-Memory Multiprocessors

第一节：Introduction

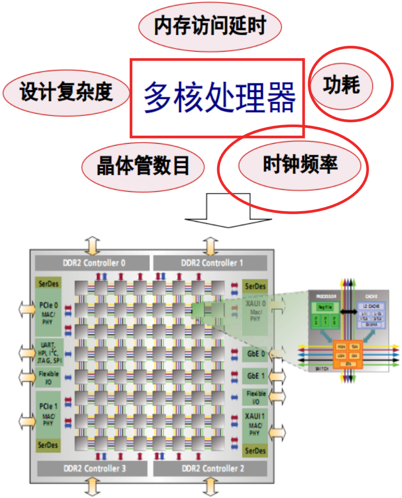
陈渝

清华大学计算机系

yuchen@tsinghua.edu.cn

2020 年 4 月 27 日

Introduction

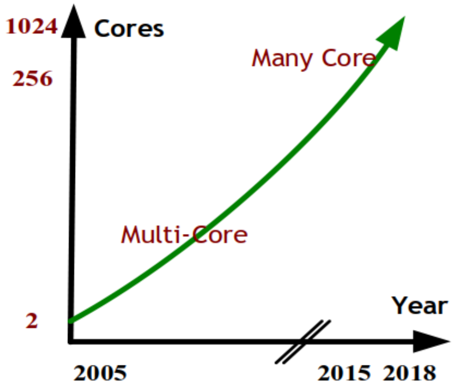


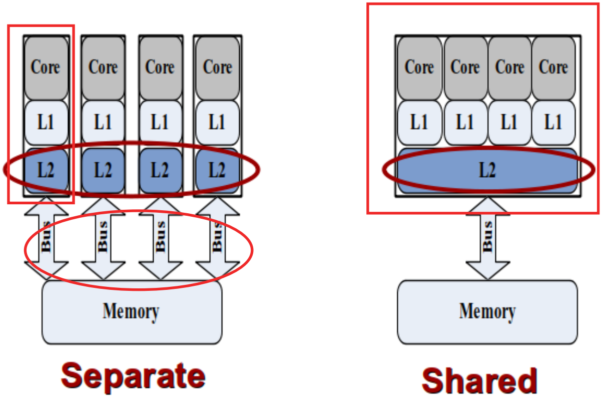
- 工业界普遍采用
- Intel/AMD/IBM/ARM/RISC-V
- 服务器/PC/笔记本/嵌入式系统

ref: Some info are from
Paul McKenney (IBM) Tom Hart (University of Toronto), Frans Kaashoek (MIT), Daniel J. Sorin ”A Primer on Memory Consistency and Cache Coherence”, Fabian Giesen ”Cache coherency primer”, Mingyu Gao(Tsinghua),Yubin Xia(SJTU)

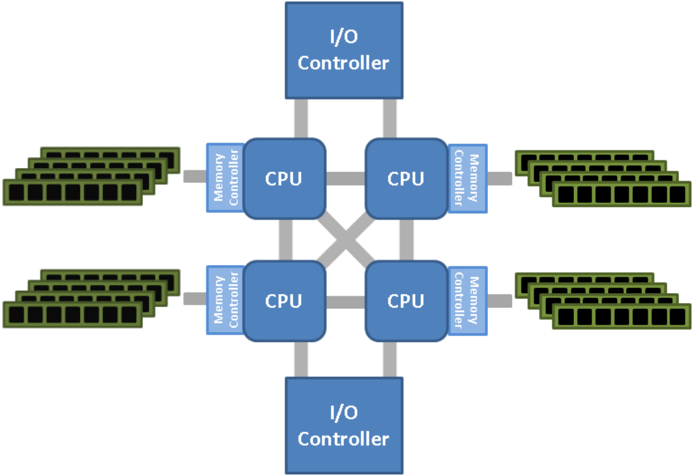
多核 (CMP) V.S. 对称多处理器 (SMP)

硬件资源共享 (e.g., 最后级缓存)



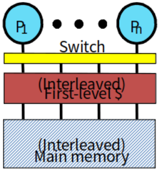


Introduction

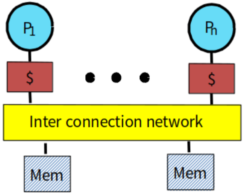


NUMA 架构

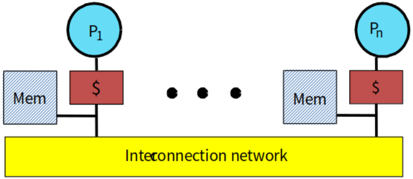
Introduction



Shared Cache



Centralized Memory
Dance Hall, UMA



Distributed Memory (NUMA)

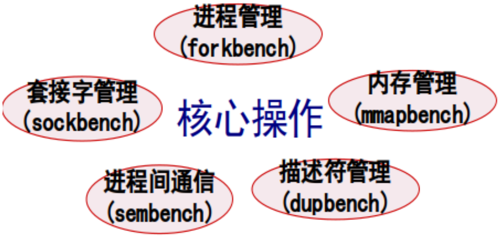
一些重要的体系结构参数数据 (cycles)

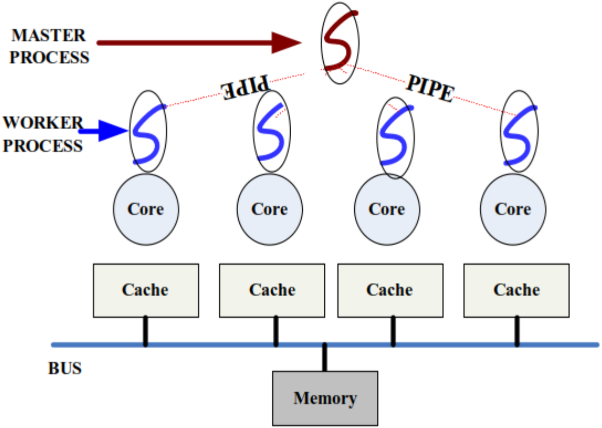
	Opteron	Xeon	Niagara	Tilera
L1	3	5	3	2
L2	15	11		11
LLC	40	44	24	45
RAM	136	355	176	118

Inst.	0-hop	1-hop	2-hop
80-core Intel Xeon machine			
Load	117	271	372
Store	108	304	409
64-core AMD Opteron machine			
Load	228	419	498
Store	256	463	544

需要分析的问题

- 操作系统在多核 +NUMA 平台上的可扩展性能如何?

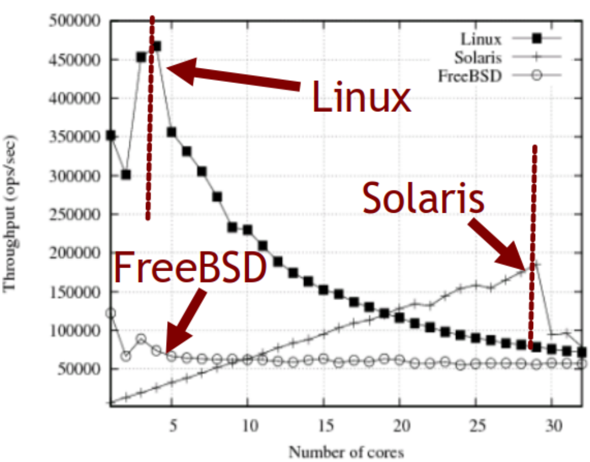


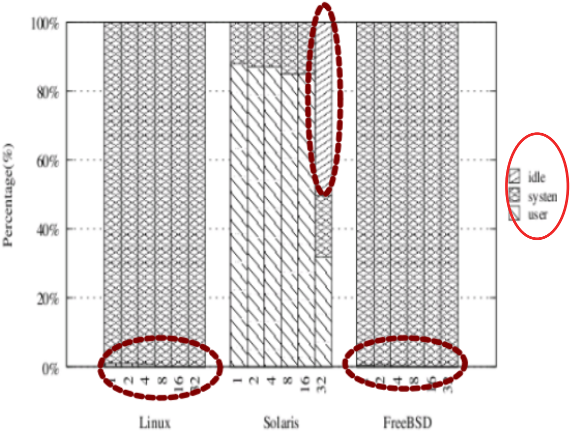


需要分析的问题

- 操作系统在多核 +NUMA 平台上的可扩展性能如何?

mmapbench





Execution time breakdown with the number of cores

Introduction

	Linux	Solaris	FreeBSD
forkbench	建立、删除VMA导致保护内存映射文件的锁竞争	缺页异常在内存映射文件的读写锁竞争	缺页异常在内存映射文件的互斥锁竞争
mmapbench	建立、删除VMA导致保护内存映射文件的锁竞争	设置内存放置策略导致内存映射文件读写锁竞争	更新和查找vnode信息在内存映射文件互斥锁竞争
dupbench	完全可扩展	关闭文件描述符在哈希表的自适应互斥锁上竞争	witness开销随着核数线性增加(去掉则完全可扩展)
sembench	保护全局信号量的读锁有竞争	完全可扩展	完全可扩展
sockbench	全局目录缓存和全局inode链表的自旋锁竞争	建立和删除流导致网络协议栈的引用计数竞争	保护全局的协议控制块链表的读写锁竞争

- 操作系统中保护共享数据结构的同步原语是影响可扩展性的重要因素
- 锁竞争可能导致可扩展性随着核数的增加而下降 (锁颠簸现象)

Introduction

What is scalability?

- Amdahl's law

Amdahl's Law (1967)

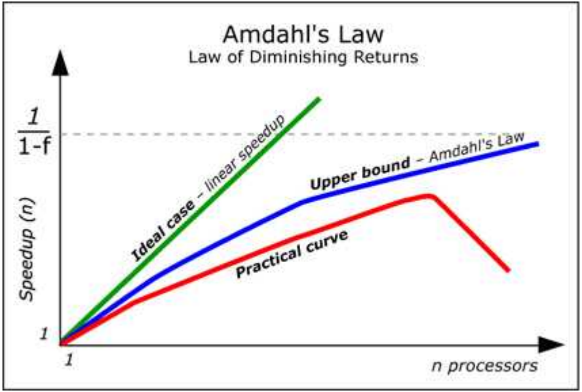
$$t_s = \text{Serial time} = 1\ \text{processor time}$$

$$f = \text{Serial code fraction}$$

$$(1-f)t_s = \text{Parallel time}$$

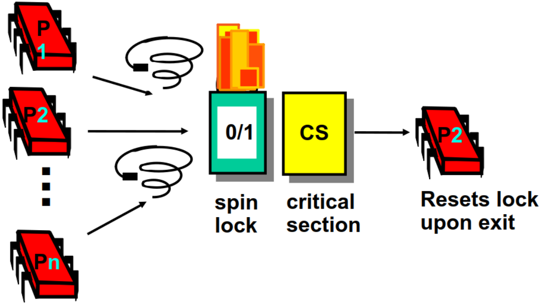
$$t_p = f \cdot t_s + (1-f)t_s / n = t_s(1+(n-1)f)/n$$

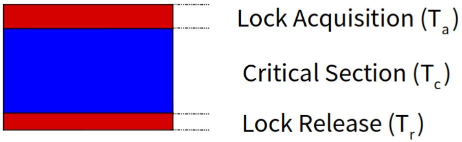
$$S(n) = \frac{t_s}{t_p} = \frac{n}{f(n-1)+1}$$



Locking/Mutex/Synchronization

- Acquire lock
- critical section
- Release lock





$$\text{Critical-section efficiency} = \frac{T_c}{T_c + T_a + T_r}$$

Introduction

Various types of synchronization techniques used by the Linux

Technique	Description	Scope
Per-CPU variables	Duplicate a data structure among the CPUs	All CPUs
Atomic operation	Atomic read-modify-write instruction to a counter	All CPUs
Memory barrier	Avoid instruction reordering	Local CPU or All CPUs
Spin lock	Lock with busy wait	All CPUs
Semaphore	Lock with blocking wait (sleep)	All CPUs
Seqlocks	Lock based on an access counter	All CPUs
Local interrupt disabling	Forbid interrupt handling on a single CPU	Local CPU
Local softirq disabling	Forbid deferrable function handling on a single CPU	Local CPU
Read-copy-update (RCU)	Lock-free access to shared data structures through pointers	All CPUs