2012-07-12

2012年 前学期 TOKYO TECH

計算機アーキテクチャ 第一 (E)

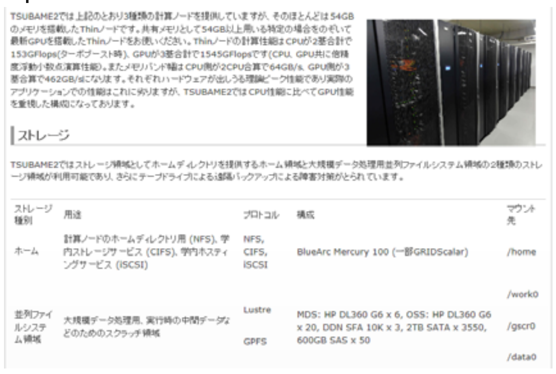
磁気ディスク, RAID

吉瀬 謙二　計算工学専攻
kise_at_cs.titech.ac.jp
W641講義室　木曜日13:20 － 14:50

Acknowledgement

- Lecture slides for Computer Organization and Design, Third Edition, courtesy of **Professor Mary Jane Irwin**, Penn State University
- Lecture slides for Computer Organization and Design, third edition, Chapters 1-9, courtesy of **Professor Tod Amon,** Southern Utah University.

TSUBAME 2.0



TSUBAME2では上記のとおり3種類の計算ノードを提供していますが、そのほとんどは54GBのメモリを搭載したThinノードです。共有メモリとして54GB以上用いる特定の場合をのぞいて最新GPUを搭載したThinノードをお使いください。Thinノードの計算性能はCPUが2基合計で153GFlops(ターボブースト時)、GPUが3基合計で1545GFlopsです(CPU、GPU共に倍精度浮動小数点演算性能)。またメモリバンド幅はCPU側が2CPU合算で64GB/s、GPU側が3基合算で462GB/sになります。それぞれハードウェアが出しうる理論ピーク性能であり実際のアプリケーションでの性能はこれに劣りますが、TSUBAME2ではCPU性能に比べてGPU性能を重視した構成になっております。

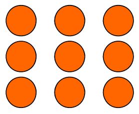
ストレージ

TSUBAME2ではストレージ領域としてホームディレクトリを提供するホーム領域と大規模データ処理用並列ファイルシステム領域の2種類のストレージ領域が利用可能であり、さらにテープドライブによる遠隔バックアップによる障害対策がとられています。

ストレージ種別	用途	プロトコル	構成	マウント先
ホーム	計算ノードのホームディレクトリ用 (NFS)、学内ストレージサービス (CIFS)、学内ホスティングサービス (iSCSI)	NFS, CIFS, iSCSI	BlueArc Mercury 100 (一部GRIDScalar)	/home
並列ファイルシステム領域	大規模データ処理用、実行時の中間データなどのためのスクラッチ領域	Lustre GPFS	MDS: HP DL360 G6 x 6, OSS: HP DL360 G6 x 20, DDN SFA 10K x 3, 2TB SATA x 3550, 600GB SAS x 50	/work0 /gscr0 /data0

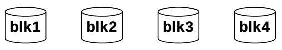
RAID: Disk Arrays

Redundant Array of Inexpensive Disks



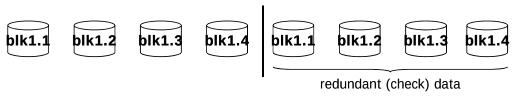
- Arrays of small and inexpensive disks
  - Increase potential **throughput** by having many disk drives
    - Data is spread over multiple disk
    - Multiple accesses are made to several disks at a time
- **Reliability** is lower than a single disk
- But **availability** can be improved by adding redundant disks (RAID)

RAID: Level 0 (RAID 0, 冗長性なし, ストライピング)



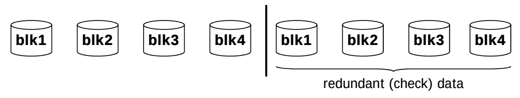
- Multiple smaller disks as opposed to **one big disk**
  - Spreading the blocks over multiple disks – **striping** – means that multiple blocks can be accessed in parallel increasing the performance
    - 4 disk system gives four times the throughput of a 1 disk system
  - Same cost as one *big* disk – assuming 4 small disks cost the same as one big disk
- No redundancy, so what if one disk fails?

RAID: Level 1 (Redundancy via Mirroring)



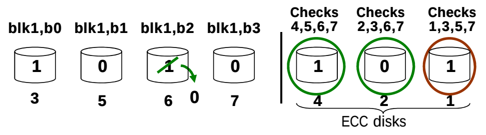
- Uses twice as many disks for redundancy so there are always two copies of the data
  - The number of redundant disks = the number of data disks so twice the cost of one big disk
    - writes have to be made to both sets of disks, so writes would be only 1/2 the performance of RAID 0
- What if one disk fails?
  - If a disk fails, the system just goes to the "**mirror**" for the data

RAID: **Level 0+1** (RAID01, Striping with Mirroring)



- Combines the best of RAID 0 and RAID 1, data is striped across four disks and mirrored to four disks
  - Four times the throughput (due to striping)
  - # redundant disks = # of data disks so twice the cost of one big disk
    - writes have to be made to both sets of disks, so writes would be only 1/2 the performance of RAID 0
- What if one disk fails?
  - If a disk fails, the system just goes to the "**mirror**" for the data

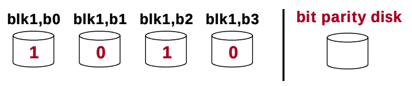
RAID: Level 2 (Redundancy via ECC)



誤り訂正コード (ECC, error-correcting code) disks 4 and 2 point to either data disk 6 or 7, but ECC disk 1 says disk 7 is okay, so disk 6 must be in error

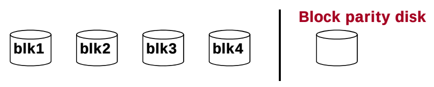
- ECC disks contain the parity of data on a set of distinct overlapping disks
  - # redundant disks = log (total # of data disks) so almost twice the cost of one big disk
    - writes require computing parity to write to the ECC disks
    - reads require reading ECC disk and confirming parity

RAID: Level 3 (Bit/Byte-Interleaved **Parity**)



- Cost of higher availability is reduced to 1/N where N is the number of disks in a protection group（保護グループ）
  - # redundant disks = 1 × # of protection groups
    - **writes** require writing the new data to the data disk as well as computing the parity, meaning reading the other disks, so that the parity disk can be updated
    - **reads** require reading all the operational data disks as well as the parity disk to calculate the missing data that was stored on the **failed disk**

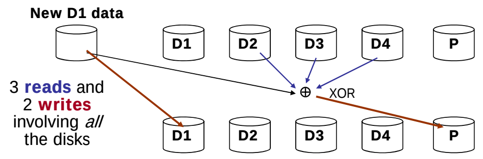
RAID: Level 4 (Block-Interleaved Parity)



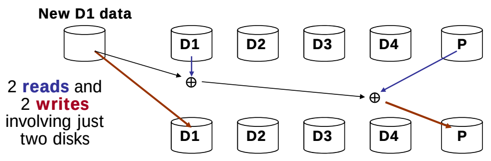
- Cost of higher availability still only 1/N but the parity is stored as **blocks** associated with sets of data blocks
  - Four times the throughput (striping)
  - # redundant disks = 1 × # of protection groups
  - Supports "small reads" and "small writes" (reads and writes that go to just one (or a few) data disk in a protection group)

Small Writes

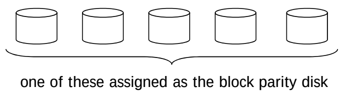
- RAID 3



- RAID 4 small writes

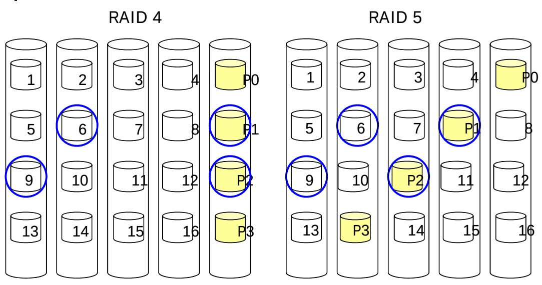


RAID: Level 5 (Distributed Block-Interleaved Parity)



- Cost of higher availability still only 1/N but the parity block can be located on any of the disks so there is no single bottleneck for writes
  - Still four times the throughput (striping)
  - # redundant disks = 1 × # of protection groups
  - Supports "**small reads**" and "**small writes**" (reads and writes that go to just one (or a few) data disk in a protection group)
  - Allows **multiple simultaneous writes**

Distributing Parity Blocks



- By distributing parity blocks to all disks, some small writes can be performed **in parallel**

Disk and RAID **Summary**

- Four components of disk access time:
  - Seek Time: advertised to be 3 to 14 ms but lower in real systems
  - Rotational Latency: 5.6 ms at 5400 RPM and 2.0 ms at 15000 RPM
  - Transfer Time: 30 to 80 MB/s
  - Controller Time: typically less than .2 ms
- RAIDs can be used to improve availability
  - RAID 0 and RAID 5 – widely used in servers, one estimate is that 80% of disks in servers are RAIDs
  - RAID 1 (mirroring) – EMC, Tandem, IBM
  - RAID 4 – Network Appliance
- RAIDs have enough redundancy to allow continuous operation

Exercise

- 磁気ディスク（ハードディスク）の信頼性を向上させる技術 RAID(Redundant Array of Inexpensive Disks)がある. 5台ハードディスクの中の1台をパリティとして用いるRAID-4の構成を示せ. また, このシステムでデータが破壊される典型的な例を議論せよ.
- mean time to failure (MTTF), mean time to repair (MTTR)を用いて, アベイラビリティの式を示せ. 先のRAID-4のアベイラビリティを向上させるために、有効な手段を述べよ。

氏名, 学籍番号,
学籍番号マーク欄

2012-07-12

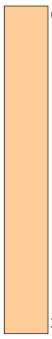
計算機アーキテクチャ 第一 (E)

仮想記憶

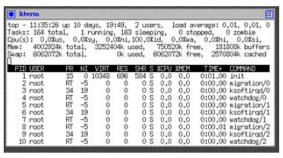
吉瀬 謙二 計算工学専攻
kise_at_cs.titech.ac.jp
W641講義室 木曜日13:20 － 14:50

例：32ビット（4GB）のメモリ空間

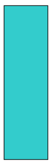
0x00000000 00000000 00000000 00000000 00000000_2 = 0_{10}



2GB Memory !



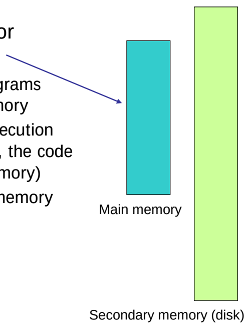
```
kterm
top - 11:35:26 up 10 days, 19:49,  2 users,  load average: 0.01, 0.01, 0
Tasks: 164 total,   1 running, 163 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,
Mem:   4002924k total,  3252404k used,   750520k free,   181808k buffers
Swap:  6062072k total,        0k used,  6062072k free,  2570804k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      15   0 10348  696  584 S  0.0  0.0   0:01.00 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/1
    6 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/1
    7 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/1
    8 root      RT  -5     0    0    0 S  0.0  0.0   0:00.01 migration/2
    9 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/2
   10 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/2
```



0xFFFFFFFF 11111111 11111111 11111111 11111111_2 = 4,294,967,296 - 1_{10}

Virtual Memory（仮想記憶）

- Use main memory as a “**cache**” for secondary memory
  - Provides the ability to easily run programs **larger** than the size of physical memory
  - **Simplifies** loading a program for execution by providing for code relocation (i.e., the code can be loaded anywhere in main memory)
  - Allows efficient and safe sharing of memory among **multiple programs**

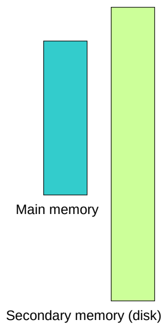


Virtual Memory（仮想記憶）

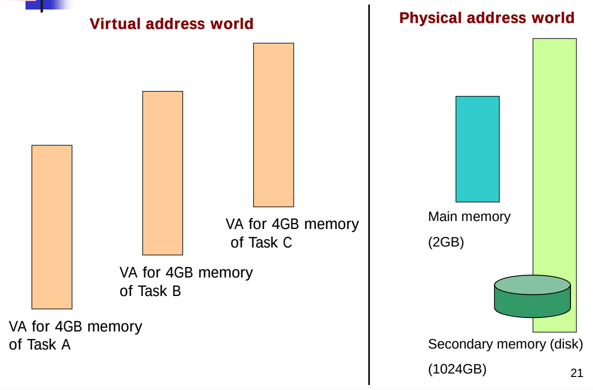
- What makes it work? – again the **Principle of Locality**
  - A program is likely to access a relatively small portion of its address space during any period of time

Virtual Memory（仮想記憶）

- Each program is compiled into its own address space – a “virtual address (VA)” space
- Physical address (PA) for the access of physical devices
  - During run-time each **virtual address, VA**（仮想アドレス）must be translated to a **physical address, PA**（物理アドレス）

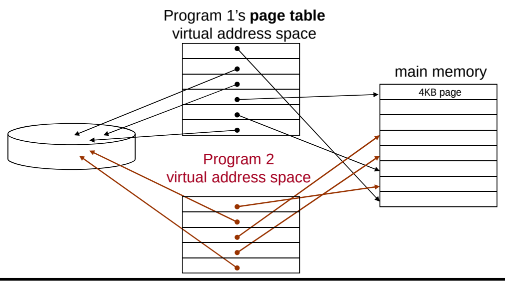


Virtual Memory（仮想記憶）



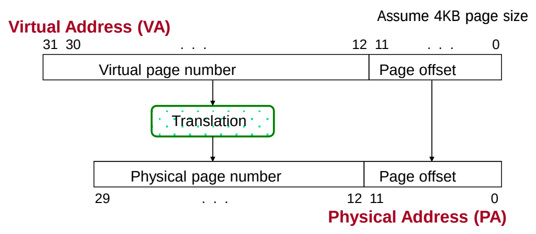
Two Programs Sharing Physical Memory

- A program’s address space is divided into **pages** (all one fixed size) or **segments** (variable sizes)
  - The starting location of each page (either in **main memory** or in **secondary memory**) is contained in the program’s **page table**



Address Translation

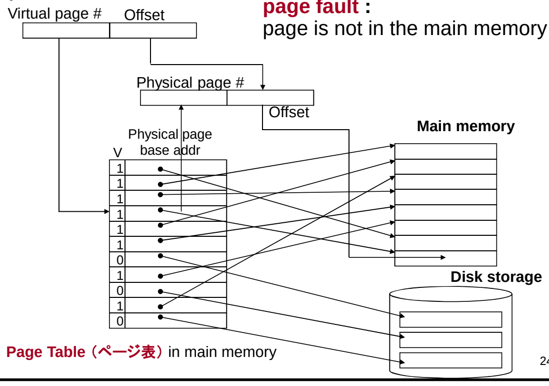
- A virtual address is translated to a physical address by a combination of hardware and software



- So each memory request **first** requires an **address translation** from the virtual space to the physical space

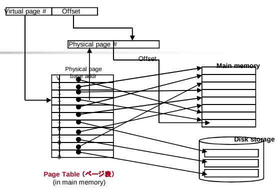
Address Translation Mechanisms

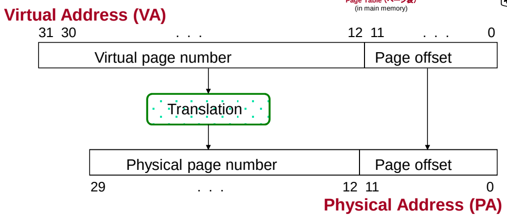
page fault :
page is not in the main memory



Page Table（ページ表）in main memory

Address Translation



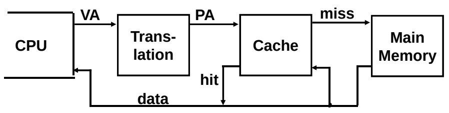


- ページサイズ4KBの場合，シンプルなページ表のメモリサイズは？

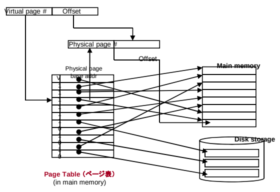
25

Virtual Addressing with a **Cache**

- Thus it takes an ***extra*** memory access to translate a virtual address to a physical address



- This makes memory (cache) accesses **very expensive** (if every access was really ***two*** accesses)



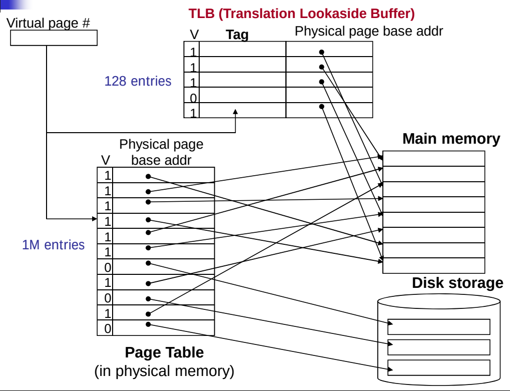
26

Virtual Addressing, the hardware fix

- The hardware fix is to use a **Translation Lookaside Buffer (TLB)**（アドレス変換バッファ）
  - a small cache that keeps track of recently used address mappings to avoid having to do a page table lookup

27

Making Address Translation Fast



28

Translation Lookaside Buffers (TLBs)

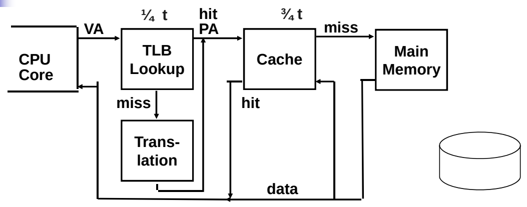
- Just like any other cache, the TLB can be organized as fully associative, set associative, or direct mapped

V	Virtual Page #	Physical Page #			

- TLB access time is **typically smaller** than cache access time (because TLBs are much smaller than caches)
  - TLBs are typically not more than 128 to 256 entries even on high end machines

29

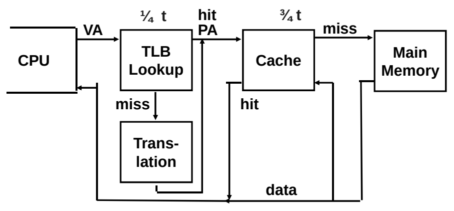
A TLB in the Memory Hierarchy



- **A TLB miss** – is it a page fault or a TLB miss ?
  - If the page is in main memory, then the TLB miss can be handled (in hardware or software) by loading the translation information from the page table into the TLB
    - Takes 10's of cycles to find and load the translation info into the TLB
  - If the page is not in main memory, then it's a true page fault
    - Takes 1,000,000's of cycles to service a page fault

30

A TLB in the Memory Hierarchy



- **page fault** : page is not in physical memory
- **TLB misses** are much more frequent than true page faults

31

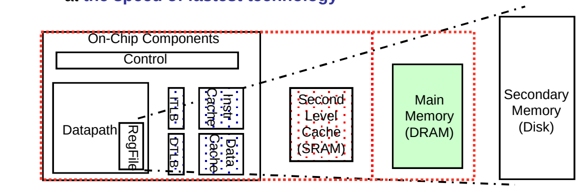
Two Machines' TLB Parameters

	Intel P4	AMD Opteron
TLB organization	1 TLB for instructions and 1TLB for data Both 4-way set associative Both use ~LRU replacement Both have 128 entries TLB misses handled in hardware	2 TLBs for instructions and 2 TLBs for data Both L1 TLBs fully associative with ~LRU replacement Both L2 TLBs are 4-way set associative with round-robin LRU Both L1 TLBs have 40 entries Both L2 TLBs have 512 entries TBL misses handled in hardware

32

A Typical Memory Hierarchy

- By taking advantage of **the principle of locality**（局所性）
  - Present **much memory** in **the cheapest technology**
  - at **the speed of fastest technology**



Speed (%cycles):	½'s	1's	10's	100's	1,000's
Size (bytes):	100's	K's	10K's	M's	G's to T's
Cost:	highest				lowest

33

The Hardware/Software Boundary

- What parts of the virtual to physical address translation is done by or assisted by the hardware?
  - **Translation Lookaside Buffer (TLB)** that caches the recent translations
    - TLB access time is part of the cache hit time
    - May cause an extra stage in the pipeline for TLB access
  - Page table storage, fault detection and updating
    - **Page faults** result in **interrupts (precise)** that are then handled by the **OS**
    - Hardware must support (i.e., update appropriately) **Dirty** and **Reference bits (e.g., ~LRU)** in the Page Tables

34

アナウンス

- 講義スライドおよびスケジュール
  - www.arch.cs.titech.ac.jp
  - 講義日程が変更になることがあるので頻繁に確認すること.

35