Gestion de la mémoire virtuelle dans Linux

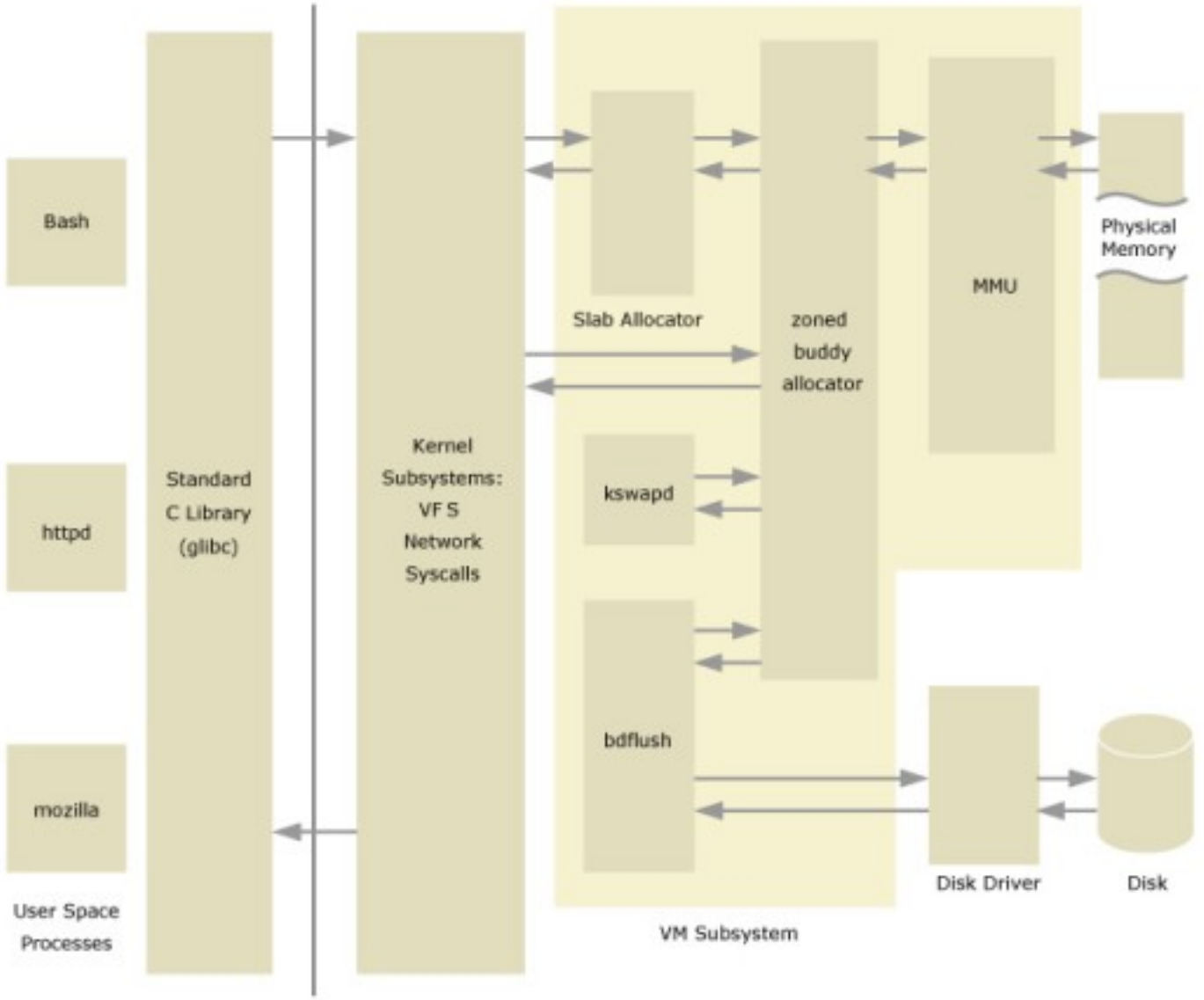
Imed Chihi <imed.chihi@gmail.com>
Décembre 2010

Agenda

- Définitions
- Le besoin de bien gérer la mémoire
- Organisation de la mémoire physique
- Organisation de la mémoire virtuelle
- Architecture et vue d'ensemble
- Avantages et exploitation en pratique
- Allocateurs de mémoire
- Quelques détails d'implémentation sous Linux

Définitions

Gestionnaire de la mémoire virtuelle
Ensemble de programmes et de matériel pour gérer la ressource “mémoire physique”



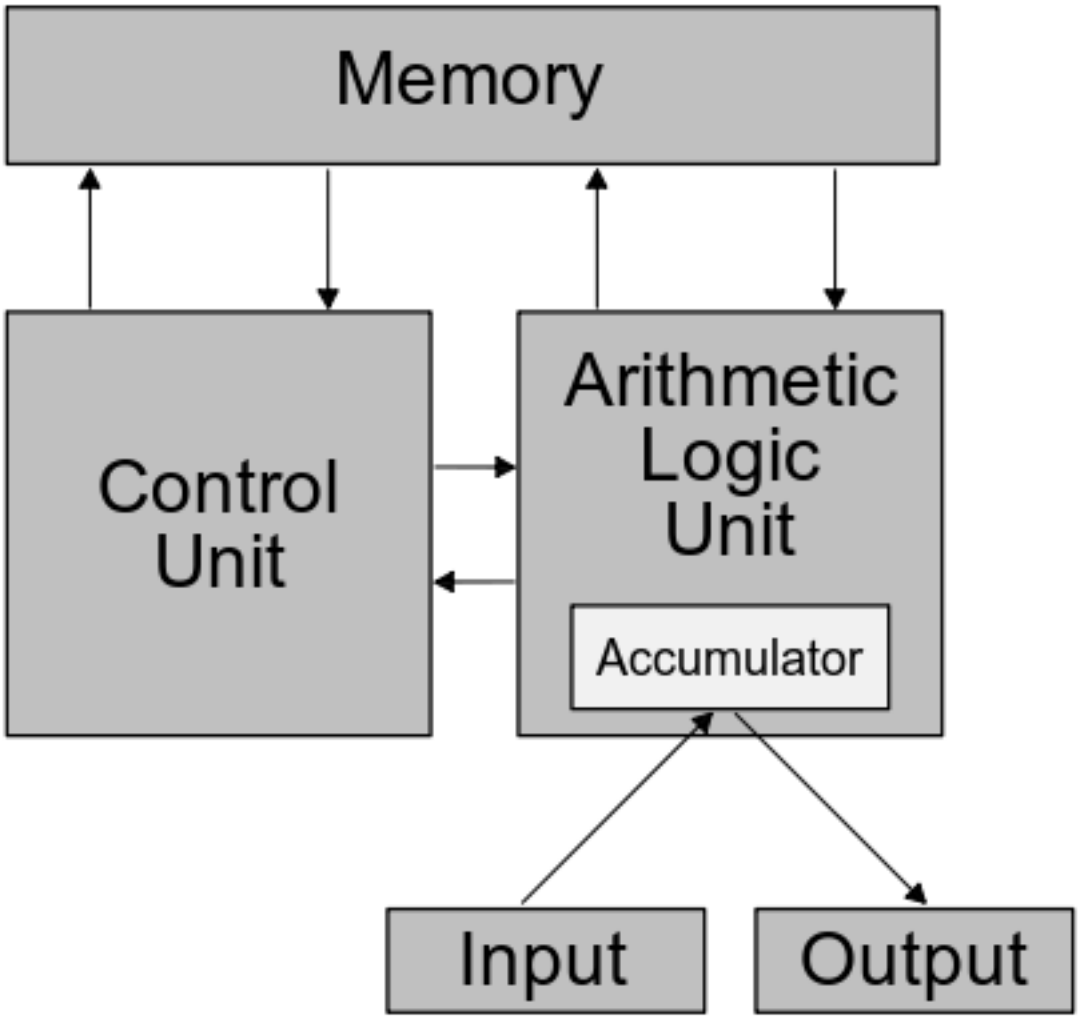
Source: Red Hat Magazine, Novembre 2004

Le besoin de bien gérer la mémoire

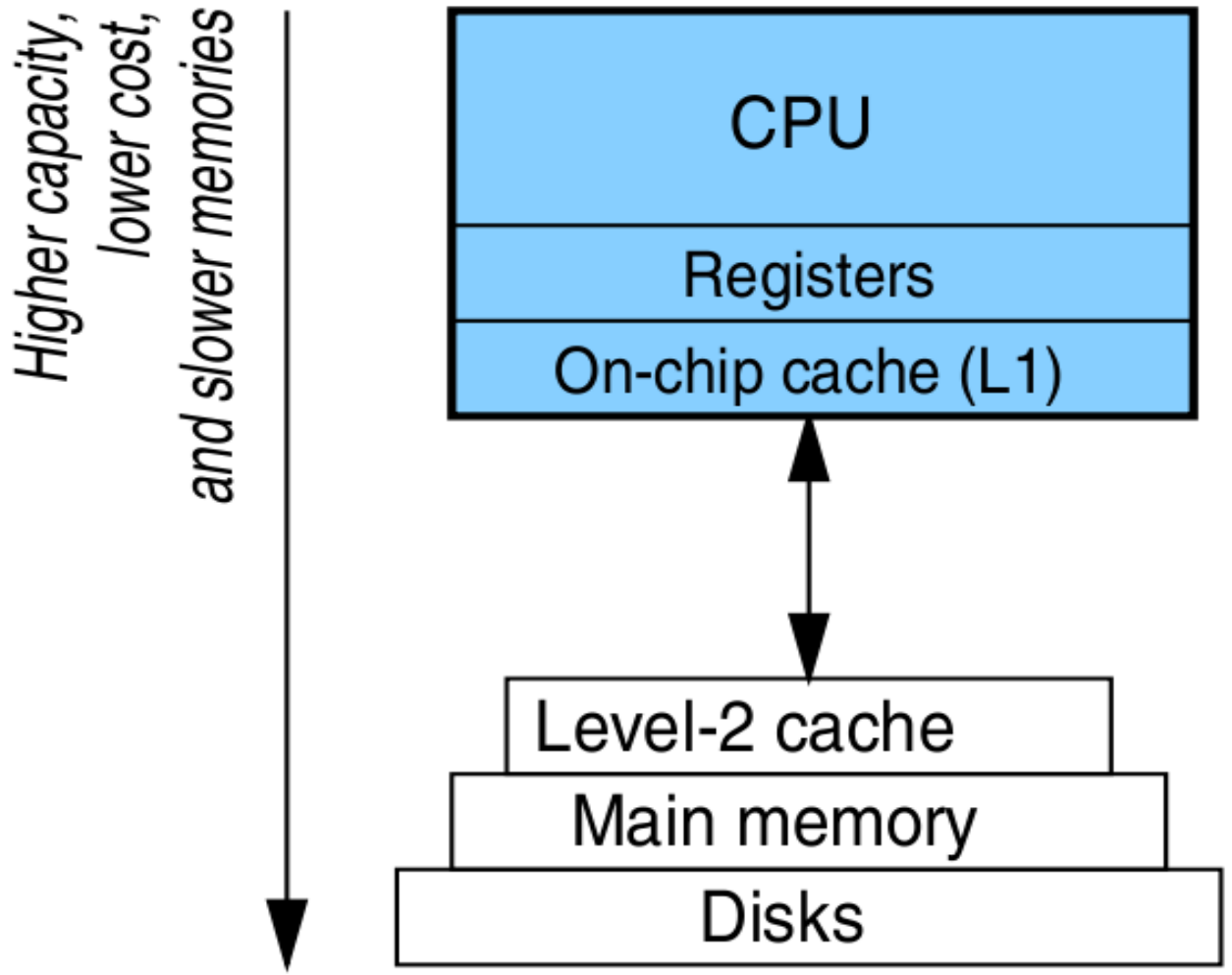
Mémoire physique
Eléments de stockage de données “hiérarchisés”

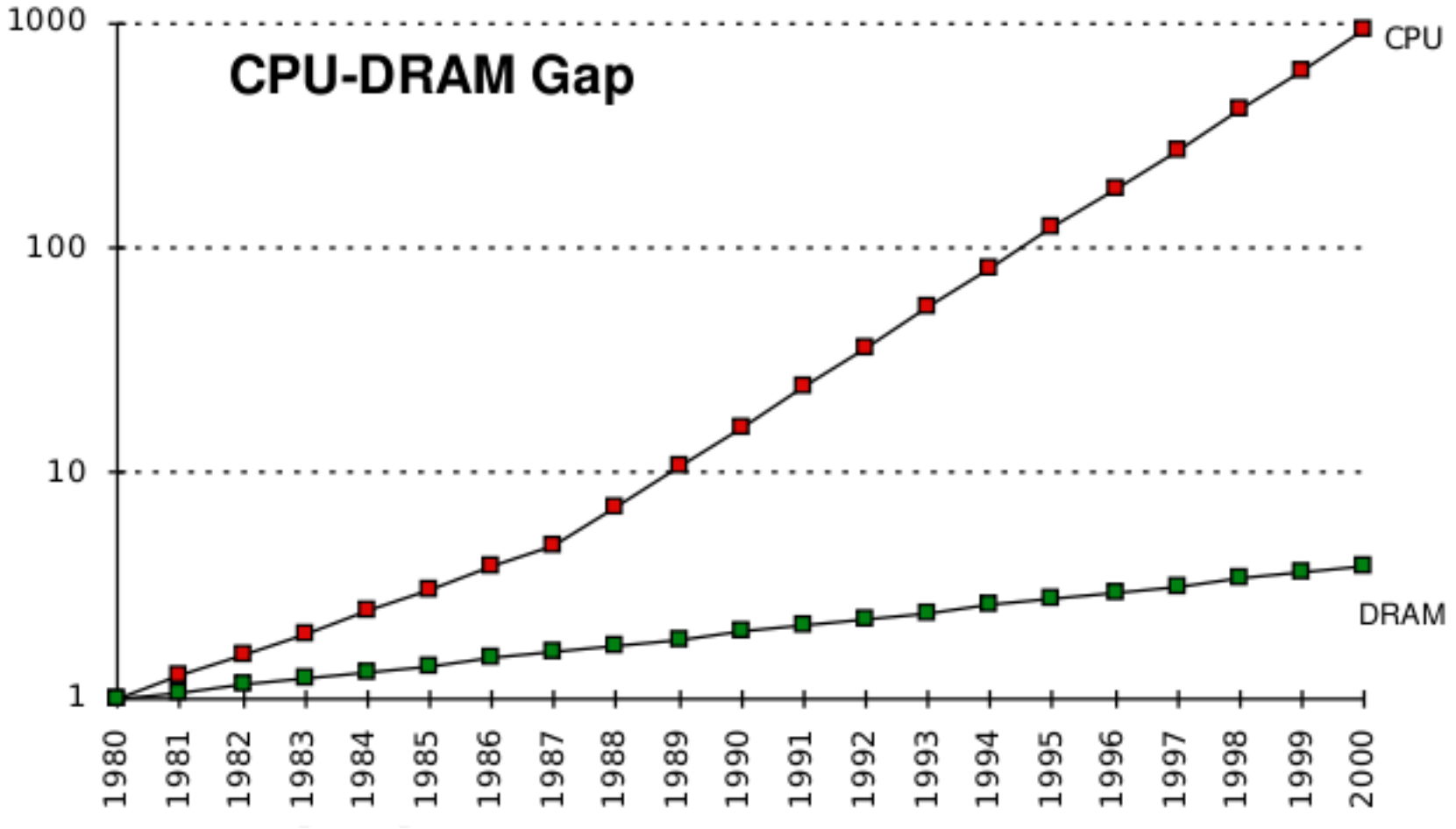
- *ressource cruciale*
- *hiérarchique*
- *pas assez rapide*

Le modèle Von Neumann

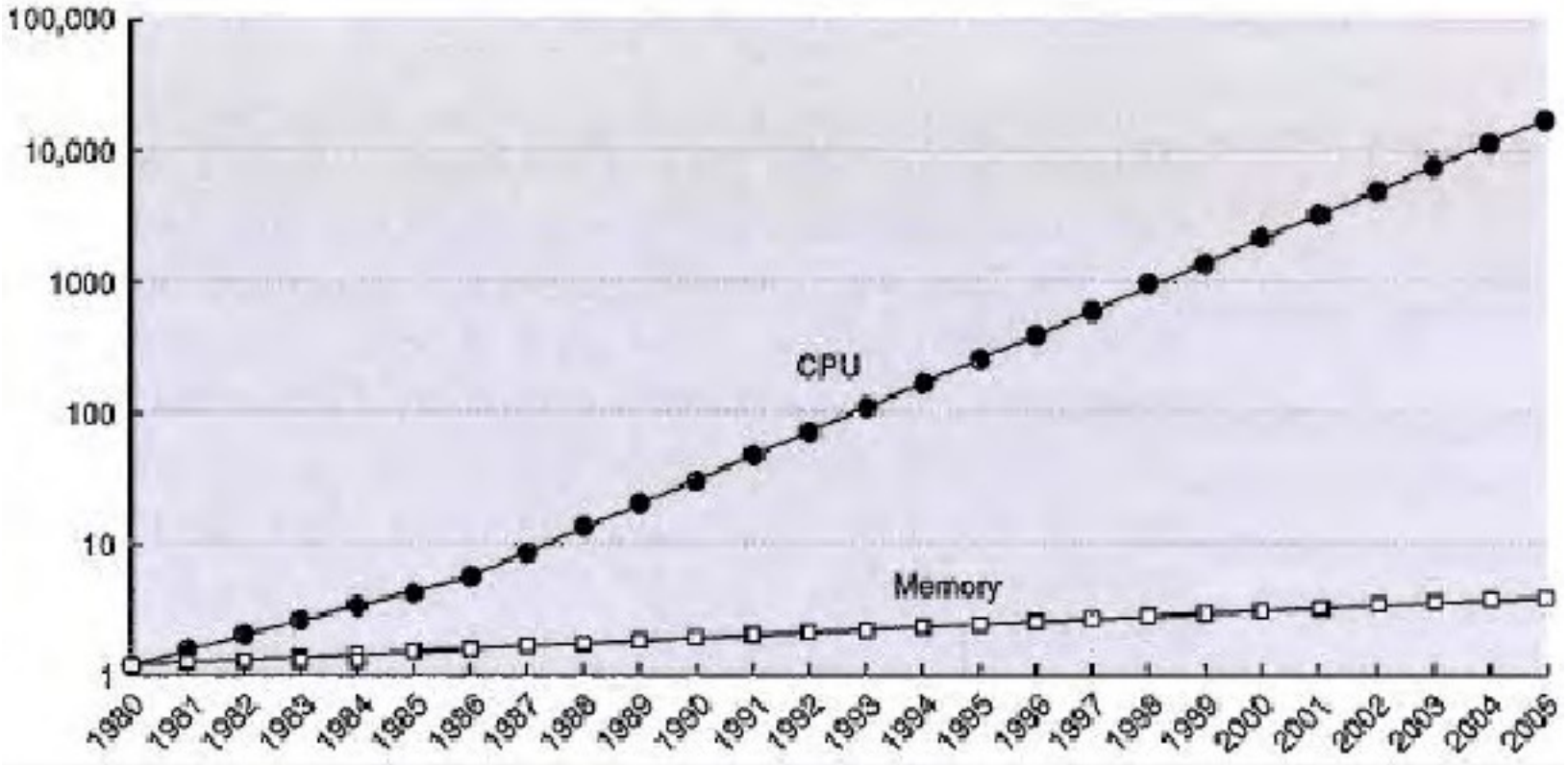


Source: Wikipedia.org


Higher capacity,
lower cost,
and slower memories
CPU
Registers
On-chip cache (L1)
Level-2 cache
Main memory
Disks


CPU-DRAM Gap
1000
100
10
1
CPU
DRAM
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

La tendence continue..



Source: Computer Architecture, AQA – Hennessy and Patterson

Organisation de la mémoire physique

- Espace d'adresses contiguës, ou presque
- Le BIOS fournit une carte de la mémoire
  - BIOS 0xe820 system memory map (int 0x15)

“Carte” de la mémoire physique

```
# dmesg
(..)
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009bc00 (usable)
 BIOS-e820: 000000000009bc00 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000e0000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 00000000bff60000 (usable)
 BIOS-e820: 00000000bff60000 - 00000000bff6a000 (ACPI data)
 BIOS-e820: 00000000bff6a000 - 00000000bff80000 (ACPI NVS)
 BIOS-e820: 00000000bff80000 - 00000000c0000000 (reserved)
 BIOS-e820: 00000000e0000000 - 00000000f0000000 (reserved)
 BIOS-e820: 00000000fec00000 - 00000000fec10000 (reserved)
 BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
 BIOS-e820: 00000000ff000000 - 0000000100000000 (reserved)
 BIOS-e820: 0000000100000000 - 0000000240000000 (usable)
(..)
Memory: 8160592k/9437184k available (2112k kernel code, 0k
reserved, 1305k data, 208k init)
```

Zones

- Les adresses mémoires ne sont pas “équivalentes”

⟶ Rassembler les adresses “similaires” dans des zones

```
# dmesg
(..)
 Zone PFN ranges:
   DMA      0x00000000 -> 0x00001000
   Normal   0x00001000 -> 0x000377fe
   HighMem  0x000377fe -> 0x000bb800
```

Non-Uniform Memory Access

```
# dmesg
(..)
Scanning NUMA topology in Northbridge 24
Number of nodes 2 (10010)
Node 0 MemBase 0000000000000000 Limit 000000003fffffff
Node 1 MemBase 0000000040000000 Limit 000000007fff0000
(..)
On node 0 totalpages: 262143
  DMA zone: 4096 pages, LIFO batch:1
  Normal zone: 258047 pages, LIFO batch:16
  HighMem zone: 0 pages, LIFO batch:1
On node 1 totalpages: 262128
  DMA zone: 0 pages, LIFO batch:1
  Normal zone: 262128 pages, LIFO batch:16
  HighMem zone: 0 pages, LIFO batch:1
```

Organisation de la mémoire virtuelle (VM)

- Présenter aux applications une vue artificielle de la mémoire physique
- Chaque processus “voit” un espace d'adressage qui lui est propre
- Le gestionnaire de la mémoire virtuelle (VMM) se charge de faire les correspondances

Paging

- Diviser la mémoire (physique et virtuelle) en des morceaux de taille fixe, appelés “**pages**”
- Maintenir des “**page tables**” pour faire l'association entre une page virtuelle et une physique
- Maintenir une “carte de mémoire” avec des informations sur chaque page physique
- Utiliser des composants matériels pour accélérer la conversion virtuel → physique

Espace d'adressage des processus

```
$ pmap -x 8375
8375:   gcalctool
Address   Kbytes     RSS    Anon  Locked Mode   Mapping
00110000    3892       -       -       - r-x--  libgtk-x11-2.0.so.0.2000.1
004dd000      16       -       -       - r----  libgtk-x11-2.0.so.0.2000.1
004e1000       8       -       -       - rw---  libgtk-x11-2.0.so.0.2000.1
004e3000       8       -       -       - rw---    [ anon ]
004e5000     256       -       -       - r-x--  libpango-1.0.so.0.2800.0
00525000       4       -       -       - -----  libpango-1.0.so.0.2800.0
00526000       4       -       -       - r----  libpango-1.0.so.0.2800.0
(..)
b7785000       4       -       -       - r----  LC_IDENTIFICATION
b7786000      28       -       -       - r--s-  gconv-modules.cache
b778d000       4       -       -       - r----  LC_NUMERIC
b778e000       8       -       -       - rw---    [ anon ]
bfe49000     112       -       -       - rw---    [ stack ]
-------- ------- ------- ------- -------
total kB   36232       -       -       -
```

Conversion des adresses mémoire

- Les “page tables” contiennent des “**page table entries**” (PTE)
- Chaque PTE contient un numéro de page et des informations sur son état
- Le “Translation Look-aside Buffer” (TLB) est un cache des résultats de la conversion virtuel → physique

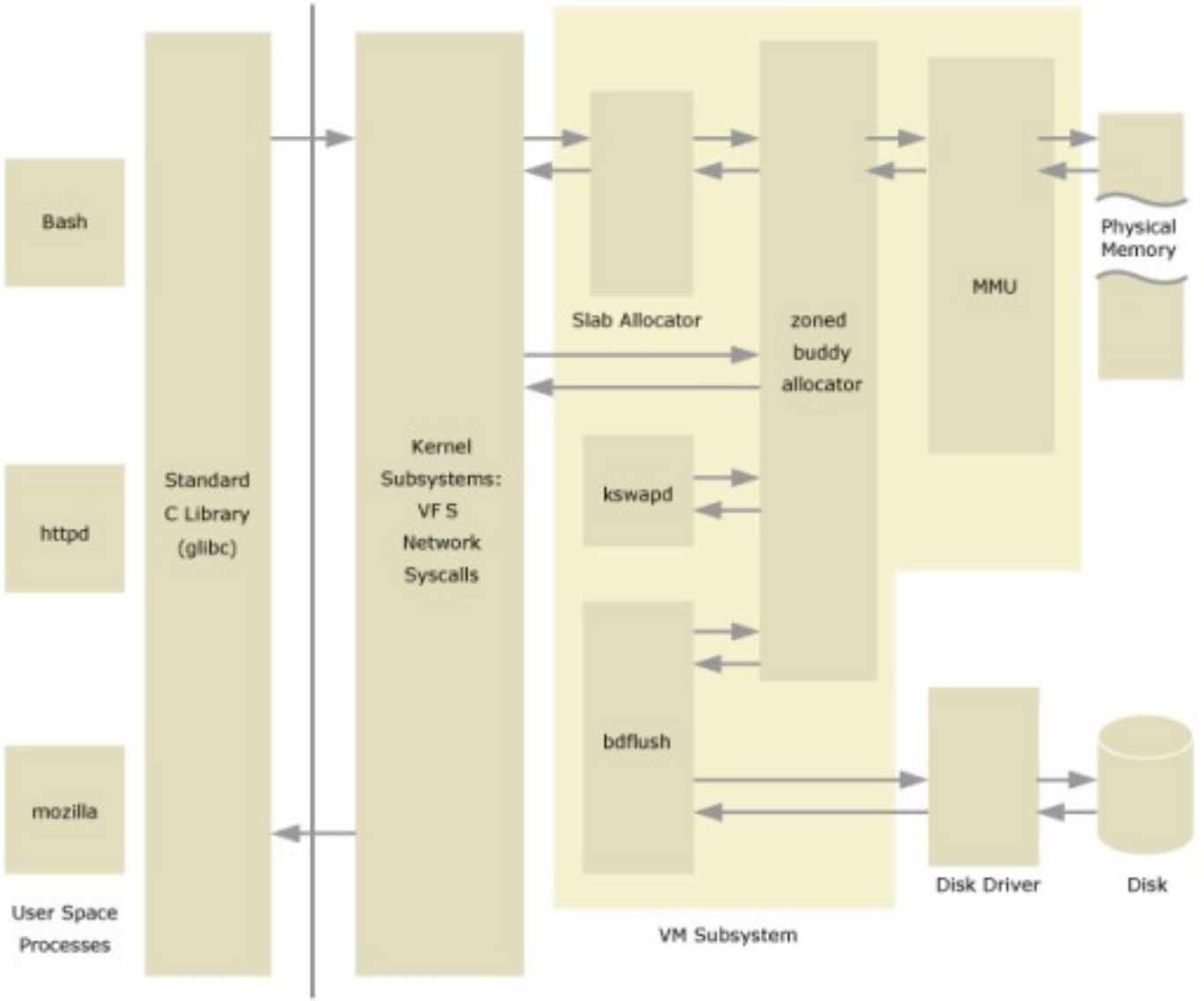
La VM, pour quoi faire?

- Les processus ont un espace d'adressage plus large
- La VM peut satisfaire des allocations qui dépsseraient la capacité physique: “**overcommit**”
- Partage de la mémoire entre plusieurs processus
- Prétendre satisfaire des allocations et retarder l'allocation réelle autant que possible: “**Copy-on-Write**” (COW) ou “**Lazy Allocation**”
- Prétendre satisfaire des chargements du disque et retarder l'accès réel autant que possible: “**demand paging**”

La VM, pour quoi faire? – cont.

- Servir de cache des accès disque (**page cache**); pas vraiment VM, mais..
- Permettre de “mapper” des fichiers à l'espace d'adressage d'un processus: assurer un “**zero-copy**”

Vue d'ensemble



Allocation de mémoire physique

- Les allocations utilisent un sous-système qui s'appelle: “**slab**”
- Alloue des objets de même taille de manière optimale

Allocation de mémoire physique

```
# cat /proc/slabinfo
slabinfo - version: 2.1
# name            <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab> :
tunables <limit> <batchcount> <sharedfactor> : slabdata <active_slabs> <num_slabs> <sharedavail>
fat_inode_cache      158    195    416   39    4 : tunables    0    0    0 : slabdata      5      5      0
fat_cache            340    340     24  170    1 : tunables    0    0    0 : slabdata      2      2      0
isofs_inode_cache     43     63    384   21    2 : tunables    0    0    0 : slabdata      3      3      0
nf_conntrack_expect      0      0    168   24    1 : tunables    0    0    0 : slabdata      0      0      0
nf_conntrack_c08d9a40     63    170    240   34    2 : tunables    0    0    0 : slabdata      5      5      0
kvm_vcpu                  0      0   9104    3    8 : tunables    0    0    0 : slabdata      0      0      0
kvm_mmu_page_header       0      0    136   30    1 : tunables    0    0    0 : slabdata      0      0      0
kmalloc_dma-512          32     64    512   32    4 : tunables    0    0    0 : slabdata      2      2      0
RAWv6                    34     46    704   23    4 : tunables    0    0    0 : slabdata      2      2      0
UDPLITEv6                 0      0    704   23    4 : tunables    0    0    0 : slabdata      0      0      0
UDPv6                    23     46    704   23    4 : tunables    0    0    0 : slabdata      2      2      0
tw_sock_TCPv6             0      0    256   32    2 : tunables    0    0    0 : slabdata      0      0      0
TCPv6                    25     48   1344   24    8 : tunables    0    0    0 : slabdata      2      2      0
flow_cache                0      0     80   51    1 : tunables    0    0    0 : slabdata      0      0      0
dm_raid1_read_record      0      0   1056   31    8 : tunables    0    0    0 : slabdata      0      0      0
kcopyd_job                0      0    272   30    2 : tunables    0    0    0 : slabdata      0      0      0
dm_uevent                 0      0   2464   13    8 : tunables    0    0    0 : slabdata      0      0      0
dm_rq_target_io           0      0    224   36    2 : tunables    0    0    0 : slabdata      0      0      0
(..)
```

Allocation de mémoire physique

- Le système slab utilise un algorithme d'allocation: “**buddy allocator**”
  - Allouer des blocs de taille en puissance de 2. Si indisponible, prendre l'ordre suivant
  - Refusionner les blocs en des blocs aussi gros que possible à la libération
- Différents types d'allocation: `GFP_ATOMIC, GFP_DMA, GFP_NOFAIL, GFP_KERNEL`

Allocation de mémoire physique

```
# echo 1 > /proc/sys/kernel/sysrq
# echo m > /proc/sysrq-trigger
$ dmesg
(..)
DMA: 6*4kB 4*8kB 4*16kB 6*32kB 4*64kB 2*128kB 4*256kB
1*512kB 2*1024kB 2*2048kB 0*4096kB = 8504kB
Normal: 10*4kB 8*8kB 4*16kB 6*32kB 2*64kB 1*128kB 3*256kB
3*512kB 2*1024kB 2*2048kB 126*4096kB = 525160kB
HighMem: 261*4kB 165*8kB 239*16kB 383*32kB 285*64kB
168*128kB 52*256kB 10*512kB 11*1024kB 15*2048kB 233*4096kB
= 1072972kB
```

Swapping et pages anonymes

- Une page peut être: **free**, **active**, **inactive** (**dirty**, **laundry**, **clean**)
- La Memory Management Unit (MMU) peut détourner certains accès à une page virtuelle: **page fault**. Mécanisme servant au demand paging et aux COW.
- Le système de fichiers et le swap sont des “**backing store**” pour les pages de la mémoire physique
- Les pages dont le backing store est le swap sont dites “**anonymous**”
- Les cas de grande demande sur la mémoire par les processus s'appelle: “**memory pressure**”

Swapping et pages anonymes

- En cas de pression mémoire, les pages éligibles sont **évictées** de la mémoire physique vers leurs backing stores respectifs
- Ce processus d'éviction et de re-attribution de mémoire est appelé: **memory reclaiming**
- Les pages “inactive” ne peuvent être évictées qu'une fois devenues “clean”

Quelques détails d'implémentation

- Physical Address Extension (PAE)
- Division espace utilisateur / espace noyau
- 32-bit vs. 64-bit
- Problèmes de la zone Normal dans les systèmes 32-bit
- Le Out Of Memory killer (OOM)
- Désactiver le pagecache avec “**O_DIRECT**” et les montages “**sync**”
- Vérrouillage de mémoire avec mlock() et applications temps réel