Comment analyser l'architecture matérielle d'un ordinateur sous Linux ?

Sid Touati
Professeur à l'université Nice Sophia Antipolis

Le but de ces exercices pratiques est de montrer comment analyser et décortiquer l'architecture matérielle d'un ordinateur sous Linux. Nous apprendrons comment faire la distinction entre processeurs, cœurs, *sockets*, hyperthreading, nœuds NUMA, etc. Je tiens à remercier mon ex doctorant, M. Mazouz, pour son aide à réaliser ces exercices.

Attention, les répertoires indiqués dans ces exercices peuvent varier d'une distribution Linux à une autre, d'une version Linux à une autre.

Table des matières

1 Analyse de l'architecture matérielle en étudiant les fichiers générés par Linux

1.1 Processeurs

Cette section vous explique comment analyser l'ensemble des CPU existant sur un ordinateur et vus par le noyau système. Attention, le système d'exploitation (comme Linux) peut confondre entre CPU, processeur, *socket* et cœurs.

1.1.1 Architectures SMT (*Simultaneous Multi-Threading*)

Avant de commencer, il faut vérifier si votre machine a des processeurs SMT (voir la documentation de votre ordinateur par exemple). Sur les architectures Intel, cela a le nom commercial d'Hyperthreading. Si cette option est activée par la Bios, le noyau système aura l'illusion que chaque cœur matériel (physique) est en fait deux cœurs logiques, car chaque cœur matériel pourra exécuter deux *threads* en parallèle. Cela veut dire que le noyau Linux détectera deux CPU distincts pour chaque cœur physique. À ma connaissance, pour savoir si l'option Hyperthreading est activée sur une architecture Intel, on peut :

1. soit consulter le menu du Bios au démarrage ;
2. soit analyser la ligne `siblings` dans le fichier `/proc/cpuinfo`. Si le nombre de cœurs = le nombre de `siblings` pour un "processor" listé dans `/proc/cpuinfo`, alors l'option Hyperthreading est désactivée. Le nombre de nombre de cœurs dans un "processor" est indiqué dans la ligne `cpu cores`.

1.1.2 Quelques exemples pour savoir si l'Hyperthreading est activé

La ligne de commande suivante vous permettrait d'afficher uniquement les lignes intéressantes du fichier `/proc/cpuinfo` pour détecter si l'option Hyperthreading est activée ou pas :
`/bin/cat /proc/cpuinfo` | /bin/egrep 'processor|model name|cache size|core|sibling|physical'

Exemple 1 : Un "processor" (CPU), 1 cœur, Hyperthreading désactivé

```
processor : 0
model name : AMD Duron(tm) processor
cache size : 64 KB
```

Exemple 2 : Un "processor" (CPU), 1 cœur, Hyperthreading activé Dans l'exemple ci-dessous, nous avons 2 siblings, mais seulement un cœur. L'identifiant du CPU physique est identique pour les deux "processors" : 0, ce qui veut dire qu'il n'y a qu'un seul cœur physique.

```
processor : 0
model name : Intel(R) Pentium(R) 4 CPU 2.80GHz
cache size : 1024 KB
physical id : 0
siblings : 2
core id : 0
cpu cores : 1

processor : 1
model name : Intel(R) Pentium(R) 4 CPU 2.80GHz
cache size : 1024 KB
physical id : 0
siblings : 2
core id : 0
cpu cores : 1
```

1.1.3 CPU détectés par le noyau linux

Le fichier texte `/proc/cpuinfo` contient la liste de tous les CPU vus par le noyau système, appelés "processor". Sur ma machine, je peux lire :

```
touati@maalouf:~> more /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
```

```
cpu family      : 6
model           : 26
model name      : Intel(R) Xeon(R) CPU           X5570  @ 2.93GHz
stepping        : 5
cpu MHz         : 2925.814
cache size      : 8192 KB
physical id     : 1
siblings        : 8
core id         : 1
cpu cores       : 4
apicid          : 18
initial apicid  : 18
fpu             : yes
fpu_exception   : yes
cpuid level     : 11
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts ac
iority ept vpid
bogomips        : 5851.89
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

...

processor : 15
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU           X5570  @ 2.93GHz
stepping : 5
cpu MHz : 2925.814
cache size : 8192 KB
physical id : 0
siblings : 8
core id : 3
cpu cores : 4
apicid : 7
initial apicid : 7
fpu : yes
fpu_exception : yes
cpuid level : 11
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxs
bogomips : 5851.98
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management:
```

Je lis donc que le noyau Linux croit qu'il a 16 CPU (appelés processeurs par Linux), numérotés de

processeur 0 à processeur 15 (ou de CPU0 à CPU15). Tous les processeurs sont identiques d'après Linux. Leur fréquence d'horloge est 2.93GHz, leur micro-architecture est Intel(R) Xeon(R) X5570. Cherchez des informations sur votre propre micro-architecture sur Internet pour avoir une idée de ses capacités micro-architecturales.

Le noyau Linux obtient les informations matérielles sur les processeurs Intel en exécutant une instruction spéciale appelée CPUID sur chaque cœur.

Dans ma machine, l'option Hyperthreading est activée. Cela veut dire qu'en fait, ma machine n'a pas 16 cœurs mais seulement 8. Mais comment détecter le nombre de *sockets* (puces distinctes) ? Une première solution serait d'ouvrir le capot de votre machine pour regarder à l'œil nu combien de *sockets* (processeurs physiques) y a t il. J'ai fait cela sur ma machine et j'ai trouvé deux *sockets*. Mais il m'est impossible de déduire à l'œil nu quel cœur appartient à quel *socket*. De plus, il est souvent impossible d'avoir l'accès ou l'autorisation d'ouvrir le capot d'un ordinateur. La section suivante vous explique comment faire pour analyser les *sockets* sur un ordinateur sous Linux.

1.1.4 Quelques exemples pour détecter les cœurs physiques

Exemple 1 : Une socket, quatre cœurs physiques Remarquez que chaque processeur a son propre identifiant de cœur (`core id`). Le nombre de siblings est égal au nombre de cœurs, donc l'Hyperthreading est désactivée. Remarquez aussi que la taille du cache affichée est 6 MB par cœur. Il n'est pas encore clair à ce niveau si ce cache est partagé entre les ccœurs ou pas (ce sera l'objet de la Section 1.2).

```
processor : 0
model name : Intel(R) Xeon(R) CPU           E5410  @ 2.33GHz
cache size : 6144 KB
physical id : 0
siblings : 4
core id : 0
cpu cores : 4

processor : 1
model name : Intel(R) Xeon(R) CPU           E5410  @ 2.33GHz
cache size : 6144 KB
physical id : 0
siblings : 4
core id : 1
cpu cores : 4

processor : 2
model name : Intel(R) Xeon(R) CPU           E5410  @ 2.33GHz
cache size : 6144 KB
physical id : 0
siblings : 4
core id : 2
cpu cores : 4

processor : 3
model name : Intel(R) Xeon(R) CPU           E5410  @ 2.33GHz
cache size : 6144 KB
physical id : 0
siblings : 4
```

```
core id : 3
cpu cores : 4
```

Exemple 2 : Une socket deux cœurs physiques Chaque processeur a son propre cœur, nous avons donc un ordinateurs avec deux cœurs physiques

```
processor : 0
model name : Intel(R) Pentium(R) D CPU 3.00GHz
cache size : 2048 KB
physical id : 0
siblings : 2
core id : 0
cpu cores : 2

processor : 1
model name : Intel(R) Pentium(R) D CPU 3.00GHz
cache size : 2048 KB
physical id : 0
siblings : 2
core id : 1
cpu cores : 2
```

Exemple 3 : Deux cœurs physiques et deux cœurs logiques, Hyperthreading activé Cet exemple montre que le processeur 0 et 2 partagent le même identifiant physique (`physical cpu`). Les processeurs 1 et 3 partagent tle même identifiant physique. Le nombre de siblings est le double du nombre de œurs, nous en déduisons que l'option Hyperthreading est activée.

```
processor : 0
model name : Intel(R) Xeon(TM) CPU 3.60GHz
cache size : 1024 KB
physical id : 0
siblings : 2
core id : 0
cpu cores : 1

processor : 1
model name : Intel(R) Xeon(TM) CPU 3.60GHz
cache size : 1024 KB
physical id : 3
siblings : 2
core id : 0
cpu cores : 1

processor : 2
model name : Intel(R) Xeon(TM) CPU 3.60GHz
cache size : 1024 KB
physical id : 0
siblings : 2
core id : 0
cpu cores : 1

processor : 3
```

```
model name : Intel(R) Xeon(TM) CPU 3.60GHz
cache size : 1024 KB
physical id : 3
siblings : 2
core id : 0
cpu cores : 1
```

1.1.5 Sockets

Linux, en se basant sur l'instruction spéciale CPUID sur architectures Intel, et en se basant sur le rapport du BIOS, est capable de donner des informations sur l'appartenance des cœurs aux *sockets*. Il faut lire les fichiers : `/sys/devices/system/cpu/cpu?/topology/core_id`

Je sais par exemple que CPU0 et CPU8 appartiennent au même cœur physique :

```
# cat /sys/devices/system/cpu/cpu0/topology/core_id
0
# cat /sys/devices/system/cpu/cpu8/topology/core_id
0
```

Similairement, CPU1 et CPU 9 appartiennent au même cœur physique :

```
# cat /sys/devices/system/cpu/cpu1/topology/core_id
1
# cat /sys/devices/system/cpu/cpu9/topology/core_id
1
```

Pour analyser les *sockets*, il faut lire le fichier `/cpu0/topology/physical_package_id`

```
#cat /sys/devices/system/cpu/cpu0/topology/physical_package_id
0
#cat /sys/devices/system/cpu/cpu8/topology/physical_package_id
0
#cat /sys/devices/system/cpu/cpu4/topology/physical_package_id
1
#cat /sys/devices/system/cpu/cpu12/topology/physical_package_id
1
```

Si je fais la lecture du fichier `physical_package_id` de tous les `cpu?` j'obtiens deux *sockets*.

Je pense que maintenant, vous pouvez récupérer les informations concernant les processeurs (CPU vus par le noyau système), les cœurs (processeur physique) et les *socket* (puce électronique qui contient éventuellement plusieurs cœurs).

1.1.6 Exemple pour distinguer entre sockets et cœurs physiques, Hyperthreading désactivé

Il est possible aussi de faire parfois quelques déductions concernants les sockets en examinant le fichier `/proc/cpuinfo`. Dans l'exemple ci dessous, il y a 4 cœurs en tout ; 2 cœurs dans 2 sockets séparées (examiner attentivement le `physical id` de chaque cœur). Chaque cœur a one taille de 4MB cache, on ne sait pas encore comment est organisée la hiérarchie mémoire.

```
processor : 0
model name : Intel(R) Xeon(R) CPU            5160  @ 3.00GHz
cache size : 4096 KB
physical id : 0
siblings : 2
core id : 0
```

```
cpu cores : 2

processor : 1
model name : Intel(R) Xeon(R) CPU            5160  @ 3.00GHz
cache size : 4096 KB
physical id : 0
siblings : 2
core id : 1
cpu cores : 2

processor : 2
model name : Intel(R) Xeon(R) CPU            5160  @ 3.00GHz
cache size : 4096 KB
physical id : 3
siblings : 2
core id : 0
cpu cores : 2

processor : 3
model name : Intel(R) Xeon(R) CPU            5160  @ 3.00GHz
cache size : 4096 KB
physical id : 3
siblings : 2
core id : 1
cpu cores : 2
```

1.1.7 Adressage mémoire (virtuelle, physique)

Le fichier `/proc/cpuinfo` contient aussi des informations importantes sur la capacité d'adressage mémoire d'un processeur. Plus précisément, la ligne `adresses sizes` indique combien de bits sont utilisés pour l'adressage de la mémoire virtuelle et combien de bits sont utilisés pour l'adressage de la mémoire physique (RAM). Soit l'exemple suivant ;

```
address sizes   : 40 bits physical, 48 bits virtual
```

La ligne ci-dessus indique que le nombre de bits utilisés par le processeur pour adresser la RAM est de 40. Ce qui veut dire que la taille maximale de RAM adressable par le processeur est 2^{40} octets= 1 tera octets. En pratique, la RAM est beaucoup plus réduite à cause de contraintes technologiques.

Aussi, la ligne ci-dessus indique que le nombre de bits utilisés par le processeur pour adresser la mémoire virtuelle est de 48. Ce qui veut dire que la taille maximale de la mémoire virtuelle d'un processus est 2^{48} octets= 128 tera octets.

1.2 Hiérarchie mémoire

Cette section vous explique comment analyser la hiérarchie mémoire de votre ordinateur : différents niveaux de caches (L1, L2, L3), nœuds NUMA, etc.

1.2.1 Caches partagés

Analysez le répertoire `/sys/devices/system/cpu` pour obtenir les informations sur les caches matériels accessibles par chaque cœur de la machine. Chaque fichier `cpu?` correspond à un cœur distinct. Accéder au répertoire "cpu ?/cache" donne des informations sur la hiérarchie cache du coeur comme suit :

— index0 L1 data cache

— index1 L1 instructions cache
— index2 L2 data + instructions
— index3 L3 data + instructions

Vous trouverez les informations de partage des caches entre les cœurs à l'intérieur du fichier `cpu?/cache/index?/shared_cpu_list`.

Par exemple, `cpu0/cache/index3/shared_cpu_list` donne la liste des cœurs avec lesquels le cœur 0 partage le cache de niveau L3.

1.2.2 Nœuds mémoire NUMA

Dans un modèle de machine von Neumann à mémoire partagée, il existe une mémoire principale unique. Linux, les programmeurs et les processus se basent sur cette vision de mémoire principale unique : il est ainsi supposé que tous les accès à la mémoire se font sur le même composant (mémoire principale). Cela est vrai du point de vue logique (un seul espace d'adressage virtuel pour chaque programme, un seul espace d'adressage physique partagé par tous les processeurs). Cependant, à cause des contraintes techniques et technologiques, certaines architectures matérielles organisent des mémoires physiques sous forme de nœuds (ou modules) mémoire séparés, *i.e.* des circuits séparés (à ne pas confondre avec des barrettes RAM). Ce modèle d'organisation de la mémoire centrale s'appelle NUMA (Non Uniform Memory Access) par opposition à UMA (Uniform Memory Access).

Un nœud NUMA est un ensemble de mémoires DRAM attachées à une ou à plusieurs de *sockets* par le biais d'un contrôleur mémoire dédié. Par conséquent, cela implique l'existence d'un bus unique reliant le/les *sockets* au nœud NUMA. Sachant que nous sommes tout de même dans des systèmes à mémoire partagée, des mécanismes existent pour permettre à un nœud NUMA de récupérer des données à partir d'un nœud NUMA voisin. À l'opposé, un modèle d'organisation UMA contient un seul nœud mémoire qui est relié avec un bus dédié unique à toutes les *sockets*.

Avoir plusieurs nœuds NUMA ne remet pas en cause la vision logique d'une mémoire centrale unique, par contre, les performances des accès mémoire deviennent différentes : le temps d'accès à une donnée en mémoire centrale dépend maintenant de son lieu de stockage, *i.e.* de son nœud NUMA.

Pour les informations NUMA, analysez le répertoire `/sys/devices/system/node`. Chaque répertoire `node?` dans `/sys/devices/system/node` représente un nœud NUMA distinct. À l'intérieur, vous trouverez les numéros des cœurs appartenant à ce nœud NUMA.

2 Analyse de l'architecture matérielle en utilisant des outils logiciels

Les outils, si vous leur faites confiance, peuvent vous donner rapidement un aperçu sur l'architecture matérielle d'un ordinateur. J'ai sélectionné deux outils pour vous illustrer cela. Ces outils sont déjà installés sur vos machines d'enseignement à l'université. Testez les, et vérifiez la concordance entre les informations affichées par ces outils, et les informations que vous avez pu déduire manuellement en examinant les fichiers Linux.

2.1 L'outil Likwid (*Lightweight performance tools*)

Vérifiez si l'outil `likwid` est installé sur votre machine (commande `locate`). Si non, téléchargez les sources de cet outil à partir d'internet, `http://code.google.com/p/likwid` La compilation de ce t outil engendre plusieurs outils. Vous seriez intéressés par `likwid-topology`

`./likwid-topology -g` donne la topologie de la machine de manière graphique.

2.2 L'outil hwloc (*portable hardware locality*)

Vérifiez si l'outil `hwloc` est installé sur votre machine (commande `locate`).

Si non, téléchargez les sources de cet outil à partir d'internet, `http://www.open-mpi.org/software/hwloc`

La compilation de ces codes source engendre plusieurs outils. Vous seriez intéressés par les outils `lstopo` ou `lstopo-no-graphics`.