Understanding Linux /proc/cpuinfo

Source : http://www.richweb.com/cpu_info

A **hyperthreaded processor** has the same number of function units as an older, non-**hyperthreaded processor**. It just has two execution contexts, so it can maybe achieve better function unit utilization by letting more than one program execute concurrently. On the other hand, if you're running two programs which compete for the same function units, there is no advantage at all to having both running "concurrently." When one is running, the other is necessarily waiting on the same function units.

A **dual core processor** literally has two times as many function units as a single-core processor, and can really run two programs concurrently, with no competition for function units.

A **dual core processor** is built so that both cores share the same **level 2 cache**. A **dual processor** (separate physical CPUs) system differs in that each CPU will have its own **level 2 cache**. This may sound like an advantage, and in some situations it can be but in many cases new research and testing shows that the shared cache can be faster when the CPUs are sharing the same or very similar tasks.

In general **Hyperthreading** is considered older technology and is no longer supported in newer CPUs. **Hyperthreading** can provide a marginal (10%) for some server workloads like MySQL, but dual core technology has essentially replaced **hyperthreading** in newer systems.

A dual core CPU running at 3.0GHz should be faster than a dual CPU (separate core) system running at 3.0GHz due to the ability to share the cache at higher bus speeds.

The examples below detail how we determine what kind of CPU(s) are present.

The kernel data Linux exposes in `/proc/cpuinfo` will show each logical CPU with a unique processor number. A logical CPU can be a **hyperthreading** sibling, a shared core in a dual or quad core, or a separate physical CPU. We must look at the siblings, CPU cores and core id to tell the difference.

If the number of cores = the number of siblings for a given physical processor, then hyperthreading is OFF.

```
/bin/cat /proc/cpuinfo | /bin/egrep 'processor|model name|cache
size|core|sibling|physical'
```

Example 1: Single processor, 1 core, no Hyperthreading

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

Example 2: Single processor, 1 core, Hyperthreading is enabled.

Notice how we have 2 siblings, but only 1 core. The physical CPU id is the same for both: 0.

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

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

Example 3. Single socket Quad Core

Notice how each processor has its own core id. The number of siblings matches the number of cores so there are no **Hyperthreading** siblings. Also notice the huge **L2 cache** - 6 MB. That makes sense though, when considering 4 cores share that **L2 cache**.

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

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

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

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

Example 3a. Single socket Dual Core

Again, each processor has its own core so this is a dual core system.

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

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

Example 4. Dual Single core CPU, Hyperthreading ENABLED

This example shows that processer 0 and 2 share the same physical CPU and 1 and 3 share the same physical CPU. The number of siblings is twice the number of cores, which is another clue that this is a system with **hyperthreading** enabled.

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

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

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

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

Example 5. Dual CPU Dual Core No hyperthreading

Of the 5 examples this should be the most capable system processor-wise. There are a total of 4 cores : 2 cores in 2 separate socketed physical CPUs. Each core shares the 4MB cache with its sibling core. The higher clock rate (3.0 GHz vs 2.3GHz) should offer slightly better performance than example 3.

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

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

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

```
physical id : 3   ← different physical id
siblings    : 2
core id     : 0   ← different core id
cpu cores   : 2

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