NAME

hwloc - General information about hwloc ("hardware locality").

DESCRIPTION

hwloc provides command line tools and a C API to obtain the hierarchical map of key computing elements, such as: NUMA memory nodes, shared caches, processor packages, processor cores, and processor "threads". hwloc also gathers various attributes such as cache and memory information, and is portable across a variety of different operating systems and platforms.

Definitions

hwloc has some specific definitions for terms that are used in this man page and other hwloc documentation.

hwloc CPU set:

A set of processors included in an hwloc object, expressed as a bitmask indexed by the physical numbers of the CPUs (as announced by the OS). The hwloc definition of "CPU set" does not carry any of the same connotations as Linux's "CPU set" (e.g., process affinity, cgroup, etc.).

hwloc node set:

A set of NUMA memory nodes near an hwloc object, expressed as a bitmask indexed by the physical numbers of the NUMA nodes (as announced by the OS).

Linux CPU set:

See http://www.mjmwired.net/kernel/Documentation/cpusets.txt for a discussion of Linux CPU sets. A super-short-ignoring-many-details description (taken from that page) is:

"Cpusets provide a mechanism for assigning a set of CPUs and Memory Nodes to a set of tasks."

Linux Cgroup:

See http://www.mjmwired.net/kernel/Documentation/cgroups.txt for a discussion of Linux control groups. A super-short-ignoring-many-details description (taken from that page) is:

"Control Groups provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behaviour."

To be clear, hwloc supports all of the above concepts. It is simply worth noting that they are different things.

Location Specification

Locations refer to specific regions within a topology. Before reading the rest of this man page, it may be useful to read lstopo(1) and/or run lstopo on your machine to see the reported topology tree. Seeing and understanding a topology tree will definitely help in understanding the concepts that are discussed below.

Locations can be specified in multiple ways:

Tuples: Tuples of hwloc "objects" and associated indexes can be specified in the form *object:index*. hwloc objects represent types of mapped items (e.g., packages, cores, etc.) in a topology tree; indexes are non-negative integers that specify a unique physical object in a topology tree. Both concepts are described in detail, below.

Indexes may also be specified as ranges. *x-y* enumerates from inde x *x* to *y*. *x:y* enumerates *y* objects starting from index *x* (wrapping around the end of the index range if needed). *x-* enumerates all objects starting from index *x*. *all*, *odd*, and *even* are also supported for listing all objects, or only those with odd or even indexes.

Chaining multiple tuples together in the more general form *object1:index[.object2:index2[...]]* is permissable. While the first tuple's object may appear anywhere in the topology, the Nth tuple's object must have a shallower topology depth than the (N+1)th tuple's object. Put simply: as you move right in a tuple chain, objects must go deeper in the topology tree. When using logical indexes (which is the default), indexes specified in chained tuples are relative to the scope of the parent object. For example, "package:0.core:1" refers to the second core in the first package.

When using OS/physical indexes, the first object matching the given index is used.

PCI and OS devices may also be designed using their identifier. For example, "**pci=02:03.1**" is the PCI device with bus ID "02:03.1". "**os=eth0**" is the network interface whose software name is "eth0". PCI devices may also be filtered based on their vendor and/or device IDs, for instance "**pci[15b3:]:2**" for the third Mellanox PCI device (vendor ID 0x15b3). OS devices may also be filtered based on their subtype, for instance "**os[gpu]:all**" for all GPU OS devices.

Hex: For tools that manipulate object as sets (e.g. hwloc-calc and hwloc-bind), locations can also be specified as hexidecimal bitmasks prefixed with "0x". Commas must be used to separate the hex digits into blocks of 8, such as "0xffc0140,0x00020110". Leading zeros in each block do not need to be specified. For example, "0xffc0140,0x20110" is equivalent to the prior example, and "0x0000000f" is exactly equivalent to "0xf". Intermediate blocks of 8 digits that are all zeoro can be left empty; "0xff0,,0x13" is equivalent to "0xff0,0x00000000,0x13". If the location is prefixed with the special string "0xf...f", then all unspecified bits are set (as if the set were infinite). For example, "0xf...f,0x1" sets both the first bit and all bits starting with the 33rd. The string "0xf...f" -- with no other specified values -- sets all bits.

"all" and "root" are special locations consisting in the root object in tree. It contains the entire current topology.

Some tools directly operate on these objects (e.g. hwloc-info and hwloc-annotate). They do not support hexadecimal locations because each location may correspond to multiple objects. For instance, there can be exactly one L3 cache per package and NUMA node, which means it's the same location. If multiple locations are given on the command-line, these tools will operation on each location individually and consecutively.

Some other tools internally manipulate objects as sets (e.g. hwloc-calc and hwloc-bind). They translate each input location into a hexidecimal location. When I/O or Misc objects are used, they are translated into the set of processors (or NUMA nodes) that are close to the given object (because I/O or Misc objects do not contain processors or NUMA nodes).

If multiple locations are specified on the command-line (delimited by whitespace), they are combined (the overall location is wider). If prefixed with "~", the given location will be cleared instead of added to the current list of locations. If prefixed with "x", the given location will be and'ed instead of added to the current list. If prefixed with "^", the given location will be xor'ed.

More complex operations may be performed by using *hwloc-calc* to compute intermediate values.

hwloc Objects

Objects in tuples can be any of the following strings (listed from "biggest" to "smallest"):

machine A set of processors and memory.

numanode
A NUMA node; a set of processors around memory which the processors can directly access. If **hbm** is used instead of **numanode** in locations, command-line tools only consider high-bandwidth memory nodes such as Intel Xeon Phi MCDRAM.

package Typically a physical package or chip, that goes into a package, it is a grouping of one or more processors.

l1cache ... l5cache
A data (or unified) cache.

l1icache ... l3icache
An instruction cache.

core A single, physical processing unit which may still contain multiple logical processors, such as hardware threads.

pu Short for *processor unit* (not *process*!). The smallest physical execution unit that hwloc recognizes. For example, there may be multiple PUs on a core (e.g., hardware threads).

osdev, **pcidev**, **bridge**, and **misc** may also be used to specify special devices although some of them have dedicated identification ways as explained in **Location Specification**.

Finally, note that an object can be denoted by its numeric "depth" in the topology graph.

hwloc Indexes

Indexes are integer values that uniquely specify a given object of a specific type. Indexes can be expressed either as *logical* values or *physical* values. Most hwloc utilities accept logical indexes by default. Passing **--physical** switches to physical/OS indexes. Both logical and physical indexes are described on this man page.

Logical indexes are relative to the object order in the output from the lstopo command. They always start with 0 and increment by 1 for each successive object.

Physical indexes are how the operating system refers to objects. Note that while physical indexes are non-negative integer values, the hardware and/or operating system may choose arbitrary values -- they may not start with 0, and successive objects may not have consecutive values.

For example, if the first few lines of lstopo -p output are the following:

```
 Machine (47GB)
   NUMANode P#0 (24GB) + Package P#0 + L3 (12MB)
     L2 (256KB) + L1 (32KB) + Core P#0 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#1 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#2 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#8 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#9 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#10 + PU P#0
   NUMANode P#1 (24GB) + Package P#1 + L3 (12MB)
     L2 (256KB) + L1 (32KB) + Core P#0 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#1 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#2 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#8 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#9 + PU P#0
     L2 (256KB) + L1 (32KB) + Core P#10 + PU P#0
```

In this example, the first core on the second package is logically number 6 (i.e., logically the 7th core, starting from 0). Its physical index is 0, but note that another core *also* has a physical index of 0. Hence, physical indexes may only be relevant within the scope of their parent (or set of ancestors). In this example, to uniquely identify logical core 6 with physical indexes, you must specify (at a minimum) both a package and a core: package 1, core 0.

Index values, regardless of whether they are logical or physical, can be expressed in several different forms (where X, Y, and N are positive integers):

X The object with index value X.

X-Y All the objects with index values >= X and <= Y.

X- All the objects with index values >= X.

X:N N objects starting with index X, possibly wrapping around the end of the level.

all A special index value indicating all valid index values.

odd A special index value indicating all valid odd index values.

even A special index value indicating all valid even index values.

REMEMBER: hwloc's command line tools accept *logical* indexes for location values by default. Use **--physical** and **--logical** to switch from one mode to another.

SEE ALSO

hwloc's command line tool documentation: lstopo(1), hwloc-bind(1), hwloc-calc(1), hwloc-distrib(1), hwloc-ps(1).

hwloc has many C API functions, each of which have their own man page. Some top-level man pages are also provided, grouping similar functions together. A few good places to start might include: hwlocality_objects(3), hwlocality_types(3), hwlocality_creation(3), hwlocality_cpuset(3), hwlocality_information(3), and hwlocality_binding(3).

For a listing of all available hwloc man pages, look at all "hwloc*" files in the man1 and man3 directories.