NAME

hwloc-calc − Operate on cpu mask strings and objects

SYNOPSIS

hwloc-calc [*topology options*] [*options*] *<location1> [<location2> [...]]*

Note that hwloc(7) provides a detailed explanation of the hwloc system and of valid <location> formats; it should be read before reading this man page.

TOPOLOGY OPTIONS

All topology options must be given before all other options.

−−no−smt, **−−no−smt=<N>**
Only keep the first PU per core in the input locations. If *<N>* is specified, keep the <N>-th instead, if any. PUs are ordered by physical index during this filtering.

−−cpukind <n>
−−cpukind <infoname>=<infovalue> Only keep PUs whose CPU kind match. Either a single CPU kind is specified as an index, or the info name/value keypair will select matching kinds.

When specified by index, it corresponds to hwloc ranking of CPU kinds which returns energy-efficient cores first, and high-performance power-hungry cores last. The full list of CPU kinds may be seen with *lstopo --cpukinds*.

−−restrict <cpuset>
Restrict the topology to the given cpuset.

−−restrict nodeset=<nodeset>
Restrict the topology to the given nodeset, unless **−−restrict−flags** specifies something different.

−−restrict−flags <flags>
Enforce flags when restricting the topology. Flags may be given as numeric values or as a comma-separated list of flag names that are passed to *hwloc_topology_restrict()*. Those names may be substrings of actual flag names as long as a single one matches, for instance **bynodeset,memless**. The default is **0** (or **none**).

−−disallowed
Include objects disallowed by administrative limitations.

−i <file>, **−−input** <file>
Read topology from XML file <file> (instead of discovering the topology on the local machine). If <file> is "−", the standard input is used. XML support must have been compiled in to hwloc for this option to be usable.

−i <directory>, **−−input** <directory>
Read topology from <directory> instead of discovering the topology of the local machine. On Linux, the directory may contain the topology files gathered from another machine topology with hwloc-gather-topology. On x86, the directory may contain a cpuid dump gathered with hwloc-gather-cpuid.

−i <specification>, **−−input** <specification>
Simulate a fake hierarchy (instead of discovering the topology on the local machine). If <specification> is "node:2 pu:3", the topology will contain two NUMA nodes with 3 processing units in each of them. The <specification> string must end with a number of PUs.

−−if <format>, **−−input−format** <format>
Enforce the input in the given format, among **xml**, **fsroot**, **cpuid** and **synthetic**.

OPTIONS

All these options must be given after all topology options above.

−p −−physical
Use OS/physical indexes instead of logical indexes for both input and output.

−l −−logical
Use logical indexes instead of physical/OS indexes for both input and output (default).

−−pi −−physical−input
Use OS/physical indexes instead of logical indexes for input.

−−li −−logical−input
Use logical indexes instead of physical/OS indexes for input (default).

−−po −−physical−output
Use OS/physical indexes instead of logical indexes for output.

−−lo −−logical−output
Use logical indexes instead of physical/OS indexes for output (default, except for cpusets which are always physical).

−n −−nodeset
Interpret both input and output sets as nodesets instead of CPU sets. See **−−nodeset−output** and **−−nodeset−input** below for details.

−−no −−nodeset−output
Report nodesets instead of CPU sets. This output is more precise than the default CPU set output when memory locality matters because it properly describes CPU-less NUMA nodes, as well as NUMA-nodes that are local to multiple CPUs.

−−ni −−nodeset−input
Interpret input sets as nodesets instead of CPU sets.

−N −−number−of <type|depth>
Report the number of objects of the given type or depth that intersect the CPU set. This is convenient for finding how many cores, NUMA nodes or PUs are available in a machine.

When combined with **−−nodeset** or **−−nodeset-output**, the nodeset is considered instead of the CPU set for finding matching objects. This is useful when reporting the output as a number or set of NUMA nodes.

−I −−intersect <type|depth>
Find the list of objects of the given type or depth that intersect the CPU set and report the comma-separated list of their indexes instead of the cpu mask string. This may be used for determining the list of objects above or below the input objects.

When combined with **−−physical**, the list is convenient to pass to external tools such as taskset or numactl **−−physcpubind** or **−−membind**. This is different from −−largest since the latter requires that all reported objects are strictly included inside the input objects.

When combined with **−−nodeset** or **−−nodeset-output**, the nodeset is considered instead of the CPU set for finding matching objects. This is useful when reporting the output as a number or set of NUMA nodes.

−H −−hierarchical <type1>.<type2>...
Find the list of objects of type <type2> that intersect the CPU set and report the space-separated list of their hierarchical indexes with respect to <type1>, <type2>, etc. For instance, if *package.core* is given, the output would be *Package:1.Core:2 Package:2.Core:3* if the input contains the third core of the second package and the fourth core of the third package.

Only normal CPU-side object types may be used. NUMA nodes cannot.

−−largest Report (in a human readable format) the list of largest objects which exactly include all input objects (by looking at their CPU sets). None of these output objects intersect each other, and

the sum of them is exactly equivalent to the input. No largest object is included in the input This is different from −−intersect where reported objects may not be strictly included in the input.

−−local−memory
Report the list of NUMA nodes that are local to the input objects.

This option is similar to **−I numa** but the way nodes are selected is different: The selection performed by **−−local−memory** may be precisely configured with **−−local−memory−flags**, while **−I numa** just selects all nodes that are somehow local to any of the input objects.

−−local−memory−flags
Change the flags used to select local NUMA nodes. Flags may be given as numeric values or as a comma-separated list of flag names that are passed to *hwloc_get_local_numanode_objs()*. Those names may be substrings of actual flag names as long as a single one matches. The default is **3** (or **smaller,larger**) which means NUMA nodes are displayed if their locality either contains or is contained in the locality of the given object.

This option enables **−−local−memory**.

−−best−memattr <name>
Enable the listing of local memory nodes with **−−local−memory**, but only display the local node that has the best value for the memory attribute given by <*name*> (or as an index).

If the memory attribute values depend on the initiator, the hwloc-calc input objects are used as the initiator.

Standard attribute names are *Capacity*, *Locality*, *Bandwidth*, and *Latency*. All existing attributes in the current topology may be listed with

```
$ lstopo --memattrs
```

−−sep <sep>
Change the field separator in the output. By default, a space is used to separate output objects (for instance when **−−hierarchical** or **−−largest** is given) while a comma is used to separate indexes (for instance when **−−intersect** is given).

−−single Singlify the output to a single CPU.

−−taskset Display CPU set strings in the format recognized by the taskset command-line program instead of hwloc-specific CPU set string format. This option has no impact on the format of input CPU set strings, both formats are always accepted.

−q −−quiet
Hide non-fatal error messages. It mostly includes locations pointing to non-existing objects.

−v −−verbose
Verbose output.

−−version Report version and exit.

−h −−help Display help message and exit.

DESCRIPTION

hwloc-calc generates and manipulates CPU mask strings or objects. Both input and output may be either objects (with physical or logical indexes), CPU lists (with physical or logical indexes), or CPU mask strings (always physically indexed). Input location specification is described in hwloc(7).

If objects or CPU mask strings are given on the command-line, they are combined and a single output is printed. If no object or CPU mask strings are given on the command-line, the program will read the standard input. It will combine multiple objects or CPU mask strings that are given on the same line of the

standard input line with spaces as separators. Different input lines will be processed separately.

Command-line arguments and options are processed in order. First topology configuration options should be given. Then, for instance, changing the type of input indexes with **−−li** or changing the input topology with **−i** only affects the processing the following arguments.

NOTE: It is highly recommended that you read the hwloc(7) overview page before reading this man page. Most of the concepts described in hwloc(7) directly apply to the hwloc-calc utility.

EXAMPLES

hwloc-calc's operation is best described through several examples.

To display the (physical) CPU mask corresponding to the second package:

```
$ hwloc-calc package:1
0x000000f0
```

To display the (physical) CPU mask corresponding to the third pacakge, excluding its even numbered logical processors:

```
$ hwloc-calc package:2 ~PU:even
0x00000c00
```

To convert a cpu mask to human-readable output, the -H option can be used to emit a space-delimited list of locations:

```
$ echo 0x000000f0 | hwloc-calc -H package.core
Package:1.Core1 Package:1.Core:1 Package:1.Core:2 Package:1.Core:3
```

To use some other character (e.g., a comma) instead of spaces in output, use the --sep option:

```
$ echo 0x000000f0 | hwloc-calc -H package.core --sep ,
Package:1.Core1,Package:1.Core:1,Package:1.Core:2,Package:1.Core:3
```

To combine two (physical) CPU masks:

```
$ hwloc-calc 0x0000ffff 0xff000000
0xff00ffff
```

To display the list of logical numbers of processors included in the second package:

```
$ hwloc-calc --intersect PU package:1
4,5,6,7
```

To bind GNU OpenMP threads logically over the whole machine, we need to use physical number output instead:

```
$ export GOMP_CPU_AFFINITY=‘hwloc-calc --physical-output --intersect PU all‘
$ echo $GOMP_CPU_AFFINITY
0,4,1,5,2,6,3,7
```

To display the list of NUMA nodes, by physical indexes, that intersect a given (physical) CPU mask:

```
$ hwloc-calc --physical --intersect NUMAnode 0xf0f0f0f0
0,2
```

To find how many cores are in the second CPU kind (those cores are likely higher-performance and more

power-hungry than cores of the first kind):

```
$ hwloc-calc --cpukind 1 -N core all
4
```

To display the list of NUMA nodes, by physical indexes, whose locality is exactly equal to a Package:

```
$ hwloc-calc --local-memory-flags 0 pack:1
4,7
```

To display the best-capacity NUMA node, by physical indexe, whose locality is exactly equal to a Package:

```
$ hwloc-calc --local-memory-flags 0 --best-memattr capacity pack:1
4
```

Converting object logical indexes (default) from/to physical/OS indexes may be performed with **--intersect** combined with either **--physical-output** (logical to physical conversion) or **--physical-input** (physical to logical):

```
$ hwloc-calc --physical-output PU:2 --intersect PU
3
$ hwloc-calc --physical-input PU:3 --intersect PU
2
```

One should add **--nodeset** when converting indexes of memory objects to make sure a single NUMA node index is returned on platforms with heterogeneous memory:

```
$ hwloc-calc --nodeset --physical-output node:2 --intersect node
3
$ hwloc-calc --nodeset --physical-input node:3 --intersect node
2
```

To display the set of CPUs near network interface eth0:

```
$ hwloc-calc os=eth0
0x00005555
```

To display the indexes of packages near PCI device whose bus ID is 0000:01:02.0:

```
$ hwloc-calc pci=0000:01:02.0 --intersect Package
1
```

To display the list of per-package cores that intersect the input:

```
$ hwloc-calc 0x00003c00 --hierarchical package.core
Package:2.Core:1 Package:3.Core:0
```

To display the (physical) CPU mask of the entire topology except the third package:

```
$ hwloc-calc all ~package:3
0x0000f0ff
```

To combine both physical and logical indexes as input:

```
$ hwloc-calc PU:2 --physical-input PU:3
0x0000000c
```

To synthetize a set of cores into largest objects on a 2-node 2-package 2-core machine:

```
$ hwloc-calc core:0 --largest
Core:0
$ hwloc-calc core:0-1 --largest
Package:0
$ hwloc-calc core:4-7 --largest
NUMANode:1
$ hwloc-calc core:2-6 --largest
Package:1 Package:2 Core:6
$ hwloc-calc pack:2 --largest
Package:2
$ hwloc-calc package:2-3 --largest
NUMANode:1
```

To get the set of first threads of all cores:

```
$ hwloc-calc core:all.pu:0
$ hwloc-calc --no-smt all
```

This can also be very useful in order to make GNU OpenMP use exactly one thread per core, and in logical core order:

```
$ export OMP_NUM_THREADS=‘hwloc-calc --number-of core all‘
$ echo $OMP_NUM_THREADS
4
$ export GOMP_CPU_AFFINITY=‘hwloc-calc --physical-output --intersect PU --no-smt all‘
$ echo $GOMP_CPU_AFFINITY
0,2,1,3
```

To export bitmask in a format that is acceptable by the resctrl Linux subsystem (for configuring cache partitioning, etc), apply a sed regexp to the output of hwloc-calc:

```
$ hwloc-calc pack:all.core:7-9.pu:0
0x00000380,,0x00000380   <this format cannot be given to resctrl>
$ hwloc-calc pack:all.core:7-9.pu:0 | sed -e ’s/0x//g’ -e ’s/,,/,0,/g’ -e ’s/,,/,0,/g’
00000380,0,00000380
# echo 00000380,0,00000380 > /sys/fs/resctrl/test/cpus
# cat /sys/fs/resctrl/test/cpus
00000000,00000380,00000000,00000380   <the modified bitmask was corrected parsed by resctrl>
```

RETURN VALUE

Upon successful execution, hwloc-calc displays the (physical) CPU mask string, (physical or logical) object list, or (physical or logical) object number list. The return value is 0.

hwloc-calc will return nonzero if any kind of error occurs, such as (but not limited to): failure to parse the command line.

SEE ALSO

hwloc(7), lstopo(1), hwloc-info(1)