NAME

hwloc-bind − Launch a command that is bound to specific processors and/or memory, or consult the binding of an existing program

SYNOPSIS

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

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 before binding. If *<N>* is specified, keep the <N>-th instead, if any. PUs are ordered by physical index during this filtering.

−−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.

−−best−memattr <name>
Select the best NUMA node among the given memory binding set by looking at the memory attribute given by *<name>* (or as an index).

If the memory attribute values depend on the initiator, the CPU binding set is 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
```

−−hbm Only take high bandwidth memory nodes (Intel Xeon Phi MCDRAM) in account when looking for NUMA nodes in the input locations.

This option must be combined with NUMA node locations, such as *--hbm numa:1* for binding on the second HBM node. It may also be written as *hbm:1*.

−−no−hbm
Ignore high bandwidth memory nodes (Intel Xeon Phi MCDRAM) when looking for NUMA nodes in the input locations.

OPTIONS

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

−−cpubind
Use following arguments for CPU binding (default).

−−membind
Use following arguments for memory binding. If **−−mempolicy** is not also given, the default policy is bind.

−−mempolicy <policy>
Change the memory binding policy. The available policies are default, firsttouch, bind, interleave and nexttouch. This option is only meaningful when an actual binding is also given with **−−membind**. If**−−membind** is gi ven without **−−mempolicy**, the default policy is bind.

−−get Report the current bindings. The output is an opaque bitmask that may be translated into objects with hwloc-calc (see EXAMPLES below).

When a command is given, the binding is displayed before executing the command. When no command is given, the program exits after displaying the current binding.

When combined with **−−membind**, report the memory binding instead of CPU binding.

No location may be given since no binding is performed.

−−nodeset Report binding as a NUMA memory node set instead of a CPU set if −−get was given. This is useful for manipulating CPU-less NUMA nodes since their cpuset is empty while their nodeset is correct.

Also parse input bitmasks as nodesets instead of cpusets.

When this option is not passed, individual input bitmasks may still be parsed as nodesets if they are prefixed with *nodeset=*.

−e −−get−last−cpu−location
Report the last processors where the process ran. The output is an opaque bitmask that may be translated into objects with hwloc-calc (see EXAMPLES below).

Note that the result may already be outdated when reported since the operating system may move the process to other processors at any time according to the binding.

When a command is given, the last processors is displayed before executing the command. When no command is given, the program exits after displaying the last processors.

This option cannot be combined with **−−membind**.

No location may be given since no binding is performed.

−−single Bind on a single CPU to prevent migration.

−−strict Require strict binding.

−−pid <pid>
Operate on pid <pid>

−−tid <tid>
Operate on thread <tid> instead of on an entire process. The feature is only supported on Linux for thread CPU binding, or for reporting the last processor where the thread ran if **−e** was also passed.

−p −−physical
Interpret input locations with OS/physical indexes instead of logical indexes. This option does not apply to the output, see **−−get** above.

−l −−logical
Interpret input locations with logical indexes instead of physical/OS indexes (default). This option does not apply to the output, see **−−get** above.

−−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.

−f −−force Launch the executable even if binding failed.

−q −−quiet
Hide non-fatal error messages. It includes locations pointing to non-existing objects, as well as failure to bind. This is usually useful in addition to **−−force**.

−v −−verbose
Verbose output.

−−version Report version and exit.

−h −−help Display help message and exit.

DESCRIPTION

hwloc-bind execs an executable (with optional command line arguments) that is bound to the specified location (or list of locations). Location specification is described in hwloc(7). Upon successful execution, hwloc-bind simply sets bindings and then execs the executable over itself.

If a bitmask location is given with prefix *nodeset=*, then it is considered a nodeset instead of a CPU set. See also **−−nodeset**.

If multiple locations are given, they are combined in the sense that the binding will be wider. The process will be allowed to run on every location inside the combination.

The list of input locations may be explicitly ended with "--".

If binding fails, or if the binding set is empty, and **−−force** was not given, hwloc-bind returns with an error instead of launching the executable.

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-bind utility.

EXAMPLES

hwloc-bind's operation is best described through several examples. More details about how locations are specified on the hwloc-bind command line are described in hwloc(7).

To run the echo command on the first logical processor of the second package:

```
$ hwloc-bind package:1.pu:0 -- echo hello
```

which is exactly equivalent to the following line as long as there is no ambiguity between hwloc-bind option names and the executed command name:

```
$ hwloc-bind package:1.pu:0 echo hello
```

To bind the "echo" command to the first core of the second package and the second core of the first package:

```
$ hwloc-bind package:1.core:0 package:0.core:1 -- echo hello
```

To bind on the first PU of all cores of the first package:

```
$ hwloc-bind package:0.core:all.pu:0 -- echo hello
$ hwloc-bind --no-smt package:0 -- echo hello
```

To bind on the memory node local to a PU with largest capacity:

```
$ hwloc-bind --best-memattr capacity --cpubind pu:23 --membind pu:23 -- echo hello
```

To bind memory on the first high-bandwidth memory node on Intel Xeon Phi:

```
$ hwloc-bind --membind hbm:0 -- echo hello
$ hwloc-bind --hbm --membind numa:0 -- echo hello
```

Note that binding the "echo" command to multiple processors is probably meaningless (because "echo" is likely implemented as a single-threaded application); these examples just serve to show what hwloc-bind can do.

To run on the first three packages on the second and third nodes:

```
$ hwloc-bind node:1-2.package:0:3 -- echo hello
```

which is also equivalent to:

```
$ hwloc-bind node:1-2.package:0-2 -- echo hello
```

Note that if you attempt to bind to objects that do not exist, hwloc-bind will not warn unless *-v* was specified.

To run on processor with physical index 2 in package with physical index 1:

```
$ hwloc-bind --physical package:1.core:2 -- echo hello
```

To run on odd cores within even packages:

```
$ hwloc-bind package:even.core:odd -- echo hello
```

To run on the first package, except on its second and fifth cores:

```
$ hwloc-bind package:0 ~package:0.core:1 ~package:0.core:4 -- echo hello
```

To run anywhere except on the first package:

```
$ hwloc-bind all ~package:0 -- echo hello
```

To run on a core near the network interface named eth0:

```
$ hwloc-bind os=eth0 -- echo hello
```

To run on a core near the PCI device whose bus ID is 0000:01:02.0:

```
$ hwloc-bind pci=0000:01:02.0 -- echo hello
```

To bind memory on second memory node and run on first node (when supported by the OS):

```
$ hwloc-bind --cpubind node:1 --membind node:0 -- echo hello
```

The --get option can report current bindings. This example shows nesting hwloc-bind invocations to set a binding and then report it:

```
$ hwloc-bind node:1.package:2 -- hwloc-bind --get
0x00004444,0x44000000
```

hwloc-calc can also be used to convert cpu mask strings to human-readable package/core/PU strings; see

the description of -H in hwloc-calc(1) for more details. The following example binds to all the PUs in a specific core, uses the --get option to retrieve where the process was actually bound, and then uses hwloc-calc to display the resulting cpu mask in space-delimited list of human-readable locations:

```
$ hwloc-bind package:1.core:2 -- hwloc-bind --get | hwloc-calc -H package.core.pu
Package:1.Core:2.PU:0 Package:1.Core:2.PU:1
```

hwloc-calc may convert this output into actual objects, either with logical or physical indexes:

```
$ hwloc-calc --physical -I pu ‘hwloc-bind --get‘
26,30,34,38,42,46
$ hwloc-calc --logical -I pu ‘hwloc-bind --get‘ --sep " "
24 25 26 27 28 29
```

Locations may also be specified as a hex bit mask (typically generated by hwloc-calc). For example:

```
$ hwloc-bind 0x00004444,0x44000000 -- echo hello
$ hwloc-bind ‘hwloc-calc node:1.package:2‘ -- echo hello
```

The current memory binding may also be reported:

```
$ hwloc-bind --membind node:1 --mempolicy interleave -- hwloc-bind --get --membind
0x000000f0 (interleave)
```

HINT

If the graphics-enabled lstopo is available, use for instance

```
$ hwloc-bind core:2 -- lstopo --pid 0
```

to check what the result of your binding command actually is. lstopo will graphically show where it is bound to by hwloc-bind.

RETURN VALUE

Upon successful execution, hwloc-bind execs the command over itself. The return value is therefore whatever the return value of the command is.

hwloc-bind will return nonzero if any kind of error occurs, such as (but not limited to): failure to parse the command line, failure to retrieve process bindings, or lack of a command to execute.

SEE ALSO

hwloc(7), lstopo(1), hwloc-calc(1), hwloc-distrib(1)