NAME

hwloc-annotate − Modify attributes in a XML topology

SYNOPSIS

hwloc-annotate [*options*] <*input.xml*> <*output.xml*> -- <*location1*> <*location2*> ... -- <*mode*> <*annotation*>

hwloc-annotate [*options*] <*input.xml*> <*output.xml*> <*location*> <*mode*> <*annotation*>

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.

OPTIONS

−−ri Remove all info attributes that exist with the same name before adding the new one. This option is only accepted in "info" mode. If the info value is omitted, existing infos are replaced with nothing.

−−ci Clear the existing info attributes in the target objects before annotating. If no new annotation has to be added after clearing, *mode* should be set to *none*.

−−cu Clear the existing userdata from the target objects. If nothing else has to be performed after clearing, *mode* should be set to *none*.

−−cd Clear the existing distances from the topology. If nothing else has to be performed after clearing, *mode* should be set to *none*.

−−version Report version and exit.

−h −−help Display help message and exit.

DESCRIPTION

hwloc-annotate loads a topology from a XML file, adds some annotations, and export the resulting topology to another XML file. The input and output files may be the same.

The annotation may be string info attributes. This is specified by the*mode*:

info <name> <value>
Specifies a new string info attribute whose name is *name* and value is *value*.

subtype <subtype>
Specifies that the subtype attribute of the object should now be *subtype*. If an empty string is given, the subtype is removed.

size <size>
Specifies the size of a cache or NUMA node. The value may be suffixed with **kB**, **MB**, **GB** or **TB**.

misc <name>
Specifies a new Misc object name.

memattr <name> <ags>
Register a new memory attribute whose name is *name* and flags is *flags*. *location* is ignored in this mode.

Flags may be given as numeric values or as a comma-separated list of flag names that are passed to *hwloc_memattr_register()*. Those names may be substrings of actual flag names as long as a single one matches. For instance, a value of **1** (or **higher**) means that highest values are considered best for this attribute.

memattr <name> <initiator> <value>
Set the memory attribute (whose name is *name*) from initiator *initiator* (either an object or a CPUset) to target NUMA node *location* to value *value*.

If this attribute does not require specific initiators, *initiator* is ignored.

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

```
$ lstopo --memattrs -i input.xml
```

cpukind <cpuset> <efficiency> <flags> [<infoname> <infovalue>]
Specifies the kind of CPU for PUs listed in the given cpuset. *location* is ignored in this mode.

efficiency is an abstracted efficiency value that will enforce ranking of kinds. It should be -1 if unknown.

flags must be 0 for now.

If *infoname* and *infovalue* are given and non-empty, they are added as info attributes to this kind of CPU.

See the function hwloc_cpukinds_register() for details.

distances <filename> [<flags>]
Specifies new distances to be added to the topology using specifications in <*filename*>. The optional *flags* (0 unless specified) corresponds to the flags given to the function **hwloc_distances_set()**. *location* is ignored in this mode.

The real first line of the pointed file must be a integer representing a distances **kind** as defined in **hwloc/distances.h**. The second line is the number of objects involved in the distances. The next lines contain one object each. The next lines contain one distance value each, or a single line may be given with a integer combination of format **x*y** or **x*y*z**. An optional line before all others may start with **name=** to specify the name of the distances structure if any.

distances-transform <name> links
Transform a bandwidth distances structure named <name> into links. See the documentation of HWLOC_DISTANCES_TRANSFORM_LINKS in hwloc/distances.h for details.

distances-transform <name> merge-switch-ports
When switches appear in the matrix as different ports, merge all of them into a single port for clarity. This currently only applies to the NVLinkBandwidth matrix between NVIDIA GPUs. See the documentation of HWLOC_DISTANCES_TRANSFORM_MERGE_SWITCH_PORTS in hwloc/distances.h for details.

distances-transform <name> transitive-closure
If objects are connected across a switch, apply a transitive-closure to report the bandwidth through that switch. This currently only applies to the NVLinkBandwidth matrix between NVIDIA GPUs. The bandwidth between all pairs of GPUs will be exposed instead of bandwidths between single GPUs and single NVSwitch ports. See the documentation of HWLOC_DISTANCES_TRANSFORM_TRANSITIVE_CLOSURE in hwloc/distances.h for details.

distances-transform <name> remove-obj <obj>
Remove the given object from the distances structure named <name>.

distances-transform <name> replace-objs <oldtype> <newtype>
Replace objects of type <oldtype> in distances structure named <name> with objects of type <newtype> with same locality. If <oldtype> or <newtype> are not object types, they are assumed subtypes of OS devices, e.g. "NVML" or "OpenCL". See the documentation of hwloc_get_obj_with_same_locality() in hwloc/helper.h for details.

If <newtype> is "NULL", objects are removed from the distances structure.

none No new annotation is added. This is useful when clearing existing attributes.

Annotations may be added to one specific object in the topology, all of them, or all of a given type. This is specified by the *location* (see also EXAMPLES below). Multiple locations may be affected if they are specified between --. Objects may be specified as location tuples, as explained in hwloc(7). However hexadecimal bitmasks are not accepted since they may correspond to multiple objects.

NOTE: The existing annotations may be listed with hwloc-info.

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-annotate utility.

EXAMPLES

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

Add an info attribute to all Core and PU objects:

```
$ hwloc-annotate input.xml output.xml -- Core:all PU:all -- info infoname infovalue
```

Only add to all Core objects:

```
$ hwloc-annotate input.xml output.xml Core:all info infoname infovalue
```

Add a Misc object named "foobar" under the root object of the topology and modify the input XML directly:

```
$ hwloc-annotate file.xml file.xml root misc foobar
```

Add an info attribute to OS device #2 and #3:

```
$ hwloc-annotate input.xml output.xml os:2-3 info infoname infovalue
```

Change package objects to green with red text in the lstopo graphical output:

```
$ hwloc-annotate topo.xml topo.xml package:all info lstopoStyle "Background=#00ff00;Text=#ff0000"
$ lstopo -i topo.xml
```

Set the memory attribute latency to 123 nanoseconds from the PUs in the first package to the first NUMA node:

```
$ hwloc-annotate topo.xml topo.xml numanode:0 memattr Latency $(hwloc-calc package:0) 123
```

Register a memory attribute **MyApplicationPerformance** (with flags specifying that it requires an initiator and reports higher values first) and set its value for initiator CPU-set 0x11 to NUMA node #2 to 2345:

```
$ hwloc-annotate topo.xml topo.xml ignored memattr MyApplicationPerformance need_init,higher
$ hwloc-annotate topo.xml topo.xml numanode:2 memattr MyApplicationPerformance 0x11 2345
```

To clarify that NUMA node #0 is DDR while NUMA node #1 is HBM:

```
$ hwloc-annotate topo.xml topo.xml numa:0 subtype DDR
$ hwloc-annotate topo.xml topo.xml numa:1 subtype HBM
```

Specify that PU 0-3 and PU 4-7 are of different kinds, and the latter is more efficient:

```
$ hwloc-annotate topo.xml topo.xml dummy cpukind 0x0f 0 0 CoreType Small
$ hwloc-annotate topo.xml topo.xml dummy cpukind 0xf0 1 0 CoreType Big
```

Replace NUMA nodes with Packages in the NUMALatency distances matrix, when they have the exact same locality.

$ hwloc-annotate topo.xml topo.xml -- dummy -- distances-transform NUMALatency replace-objs numanode packages

Remove NUMA node #3 from the NUMALatency distances matrix:

$ hwloc-annotate topo.xml topo.xml -- dummy -- distances-transform NUMALatency remove-obj numa:3

Merge all NVSwitch ports bandwidth information into a single port in the NVLinkBandwidth matrix:

$ hwloc-annotate topo.xml topo.xml -- dummy -- distances-transform NVLinkBandwidth merge-switch-ports

Apply a transitive closure to get inter-GPU bandwidth across NVSwitches in the NVLinkBandwidth matrix:

$ hwloc-annotate topo.xml topo.xml -- dummy -- distances-transform NVLinkBandwidth transitive-closure

RETURN VALUE

Upon successful execution, hwloc-annotate generates the output topology. The return value is 0.

hwloc-annotate 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)