Report: hierarchical workstealing in XKAAPI

XKAAPI team

Contents

1 Introduction

TODO: describe the need for HWS

2 Enabling HWS in XKAAPI

2.1 Build configuration

No new configuration has been added to the build system. However, the runtime has to be compiled with *hwloc* and *numa* support for HWS to be enabled:

```
./configure --with-hwloc --with-numa
```

2.2 Environment variables

The steal request emission routine has to be specialized by setting the *KAAPI_EMITSTEAL* environment variable to “hws”:

```
$> KAAPI_HWS_LEVEL=hws ./a.out
```

Specific memory heriarchy levels can be used by setting the *KAAPI_HWS_LEVELS* environment variable. It consists of a comma separated list of one or more of the following values:

- ALL: enables all the levels,
- NONE: disables all the levels,
- L3: enable the L3 cache level,
- NUMA: enable the numa level,
- SOCKET: enable the socket level,
- MACHINE: enable the machine level,
- FLAT: enable the flat level.

For instance:

```
$> KAAPI_HWS_LEVEL=hws KAAPI_HWS_LEVELS=FLAT,NUMA ./a.out
```

Not setting this variable enables the NUMA, SOCKET, MACHINE and FLAT memory levels.

3 HWS implementation in XKAAPI

3.1 Overview

The *src/hws* directory has been added to the runtime sourcecode. It implements:

- subsystem initialization (*kaapi_hws_initialize.c*),
- task pushing (*kaapi_hws_pushtask.c*),
- steal request emission (*kaapi_hws_emitsteal.c*),
- workstealing queue implementation (*kaapi_ws_queue_xxx.c*),
- related performance counters (*kaapi_hws_counters.c*),
- scheduler synchronization (*kaapi_hws_sched_sync.c*) .

3.2 Hierarchy construction

TODO

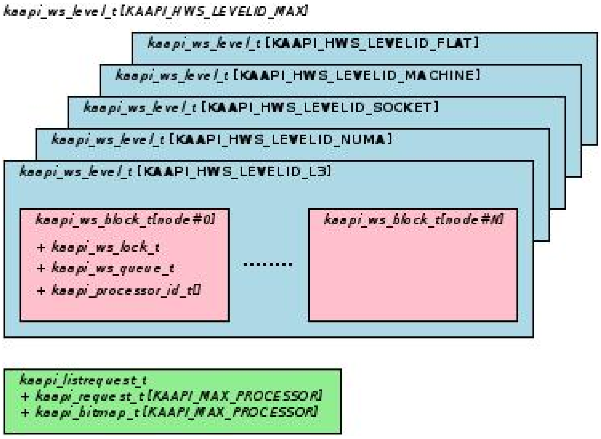
kaapi_ws_level_t [KAAPI_HWS_LEVELID_MAX]
kaapi_ws_level_t [KAAPI_HWS_LEVELID_FLAT]
kaapi_ws_level_t [KAAPI_HWS_LEVELID_MACHINE]
kaapi_ws_level_t [KAAPI_HWS_LEVELID_SOCKET]
kaapi_ws_level_t [KAAPI_HWS_LEVELID_NUMA]
kaapi_ws_level_t [KAAPI_HWS_LEVELID_L3]
kaapi_ws_block_t[node#0]
+ kaapi_ws_lock_t
+ kaapi_ws_queue_t
+ kaapi_processor_id_t[]
....
kaapi_ws_block_t[node#N]
kaapi_listrequest_t
+ kaapi_request_t [KAAPI_MAX_PROCESSOR]
+ kaapi_bitmap_t [KAAPI_MAX_PROCESSOR]

Figure 1: HWS data structure relations

3.3 Steal request emission algorithm

TODO

3.4 Workstealing queue virtualization

TODO

3.5 Emisteal routine virtualization

The *emitsteal* routine has been virtualized. The *KAAPI_EMITSTEAL* environment variable controls the implementation used:

- hws: use the hierarhical workstealing implemented by the *kaapi_hws_sched_emitsteal* routine,
- any other value: use the default *kaapi_sched_emitsteal* routine.

4 Benchmarks

TODO