Class CPUResourceMonitor

Table of contents

- Defined in File cpu_resource_monitor.hpp

Class Documentation

class CPUResourceMonitor

CPUResourceMonitor class.

This class is responsible for monitoring the CPU resources. It provides the information about the CPU resources (through `holoscan::CPUInfo`) to the SystemResourceManager class.

The following `holoscan::CPUMetricFlag` flags are supported:

- `DEFAULT`: Default CPU metrics (`CORE_COUNT | CPU_COUNT | AVAILABLE_PROCESSOR_COUNT`)
- `CORE_COUNT`: Number of cores (num_cores)
- `CPU_COUNT`: Number of CPUs (num_cpus)
- `AVAILABLE_PROCESSOR_COUNT`: Number of available processors (num_processors)
- `CPU_USAGE`: CPU usage (cpu_usage)
- `MEMORY_USAGE`: Memory usage (memory_total, memory_free, memory_available, memory_usage)
- `SHARED_MEMORY_USAGE`: Shared memory usage (shared_memory_total, shared_memory_free, shared_memory_available, shared_memory_usage)
- `ALL`: All CPU metrics

CPU usage information is based on the `/proc/stat` file and it calculates the CPU usage between the current and the previous CPU usage information, so it is necessary to call `update()` method at least once before calling `cpu_info()` method.

Example:

```
#include <chrono> // std::chrono::seconds #include <thread> //
std::this_thread::sleep_for #include
<holoscan/core/system/system_resource_manager.hpp> #include
<holoscan/logger/logger.hpp> ... holoscan::Topology topology; topology.load();
holoscan::CPUResourceMonitor cpu_resource_monitor(topology.context());
holoscan::CPUInfo cpu_info =
cpu_resource_monitor.cpu_info(holoscan::CPUMetricFlag::ALL);
std::this_thread::sleep_for(std::chrono::seconds(1)); // wait for cpu usage update
// Update CPU usage info cpu_info = system_resource_manager.cpu_monitor()-
>update(holoscan::CPUMetricFlag::CPU_USAGE); HOLOSCAN_LOG_INFO("CPU
cores: {}", cpu_info.num_cores); HOLOSCAN_LOG_INFO("CPU cpus: {}",
cpu_info.num_cpus); HOLOSCAN_LOG_INFO("CPU processors: {}",
cpu_info.num_processors); HOLOSCAN_LOG_INFO("CPU usage: {}",
cpu_info.cpu_usage); HOLOSCAN_LOG_INFO("CPU memory total: {}",
cpu_info.memory_total); HOLOSCAN_LOG_INFO("CPU memory free: {}",
cpu_info.memory_free); HOLOSCAN_LOG_INFO("CPU memory available: {}",
cpu_info.memory_available); HOLOSCAN_LOG_INFO("CPU memory usage: {}",
cpu_info.memory_usage); HOLOSCAN_LOG_INFO("CPU shared memory total:
{}", cpu_info.shared_memory_total); HOLOSCAN_LOG_INFO("CPU shared
memory free: {}", cpu_info.shared_memory_free); HOLOSCAN_LOG_INFO("CPU
shared memory available: {}", cpu_info.shared_memory_available);
HOLOSCAN_LOG_INFO("CPU shared memory usage: {}",
cpu_info.shared_memory_usage);
```

Public Functions

explicit CPUResourceMonitor(void *context, uint64_t metric_flags = kDefaultCpuMetrics)

Construct a new CPUResourceMonitor object.

This constructor creates a new CPUResourceMonitor object.

Parameters

- **context** – The topology context (from `holoscan::Topology::context()`).
- **metric_flags** – The metric flags (default: `CORE_COUNT | CPU_COUNT | AVAILABLE_PROCESSOR_COUNT`)

virtual ~CPUResourceMonitor() = default

uint64_t metric_flags() const

Get metric flags.

This function returns the metric flags.

Returns

The metric flags.

void metric_flags(uint64_t metric_flags)

Set metric flags.

This function sets the metric flags.

Parameters

metric_flags – The metric flags

CPUInfo update(uint64_t metric_flags = CPUMetricFlag::DEFAULT)

Update the CPU information and cache it.

This function updates information for the CPU with the given metric flags and returns the CPU information. If the metric flags are not provided, the existing metric flags are used. It also caches the CPU information.

Parameters

metric_flags – The metric flags.

Returns

The CPU information.

CPUInfo &update(CPUInfo &cpu_info, uint64_t metric_flags = CPUMetricFlag::DEFAULT)

Update the CPU information.

This function fills the CPU information given as the argument based on the given metric flags and returns the CPU information. If the metric flags are not provided, the existing metric flags are used.

Parameters

- **cpu_info** – The CPU information.
- **metric_flags** – The metric flags.

Returns

The CPU information filled with the updated values (same as the argument).

CPUInfo cpu_info(uint64_t metric_flags = CPUMetricFlag::DEFAULT)

Get the CPU information.

If the metric flags are provided, it returns the CPU information based on the given metric flags. If the metric flags are not provided, it returns the cached CPU information.

Parameters

metric_flags – The metric flags.

Returns

The CPU information.

cpu_set_t cpu_set() const

Get the CPU set.

The returned value (`cpu_set_t`) can be used to check whether the given CPU is available or not.

See https://linux.die.net/man/2/sched_getaffinity for more details.

Example:

```
#include <holoscan/core/system/system_resource_manager.hpp>
#include <holoscan/logger/logger.hpp> ... holoscan::Topology topology;
topology.load(); holoscan::CPUResourceMonitor
cpu_resource_monitor(topology.context()); holoscan::CPUInfo cpu_info =
cpu_resource_monitor.cpu_info(); cpu_set_t cpu_set =
cpu_resource_monitor.cpu_set(); for (int i = 0; i < cpu_info.num_cpus; i++)
{ if (CPU_ISSET(i, &cpu_set)) { HOLOSCAN_LOG_INFO("CPU {} is available",
i); } }
```

Returns

The CPU set.

Protected Attributes

void *context_ = nullptr

The context of the CPU resource monitor.

uint64_t metric_flags_ = kDefaultCpuMetrics

The metric flags.

bool is_cached_ = false

The flag to indicate whether the CPU information is cached.

CPUInfo cpu_info_ = {}

The cached CPU information.

cpu_set_t cpu_set_ = {}

The cached CPU set

bool is_last_total_stats_valid_ = false

The flag to indicate whether the last total cpu usage stats are valid.

uint64_t last_total_stats_[4] = {0}

The last total cpu usage stats.