Ghoti.io Pool

0.1

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

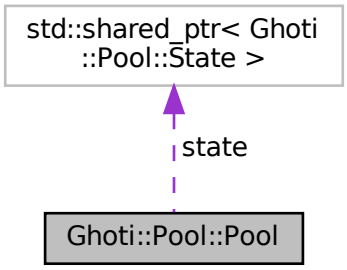
Class Documentation

3.1 Ghoti::Pool::Pool Class Reference

Represents a generalized thread pool.

`#include <pool.hpp>`

Collaboration diagram for Ghoti::Pool::Pool:



Public Member Functions

- Pool ()

  Default thread pool constructor.
- Pool (size_t threadCount)

  Thread pool constructor for a specific number of threads.
- ~Pool ()

  Thread pool destructor.
- **Pool** (const Ghoti::Pool::Pool &)=delete
- Ghoti::Pool::Pool & **operator=** (const Ghoti::Pool::Pool &)=delete
- bool enqueue (Task &&task)

  Enqueue a Task for the thread pool.

- void start ()

  Start the thread pool processing.
- void stop ()

  Stop the thread pool from dispatching new tasks and remove the existing threads.
- void join ()

  Stop the thread pool (if not already stopped) and join all threads.
- size_t getTaskQueueCount ()

  Returns the number of tasks currently in the task queue.
- void setThreadCount (size_t threadCount)

  Set the thread count.
- size_t getThreadCount () const

  Returns the number of threads that are created.
- size_t getWaitingThreadCount () const

  Returns the number of threads that are waiting.
- size_t getTerminatedThreadCount () const

  Returns the number of threads that are terminated.
- size_t getRunningThreadCount () const

  Returns the number of threads that are running.

Private Member Functions

- void createThreads ()

  Keep creating threads until the limit is reached.

Private Attributes

- std::shared_ptr< State > state

  Pointer to the shared state of the thread pool.

3.1.1 Detailed Description

Represents a generalized thread pool.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Pool() [1/2]

```
Pool::Pool ( )
```

Default thread pool constructor.

Will create as many threads as the total number of logical cores on the system.

3.1.2.2 Pool() [2/2]

```
Pool::Pool (
            size_t threadCount )
```

Thread pool constructor for a specific number of threads.

Parameters

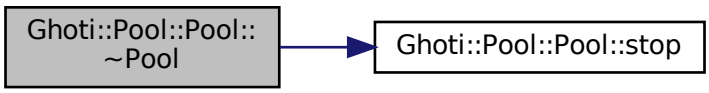
threadCount	The desired number of threads.

3.1.2.3 ~Pool()

```
Pool::~Pool ( )
```

Thread pool destructor.

When the pool is destroyed, it will signal all threads to stop. Here is the call graph for this function:



3.1.3 Member Function Documentation

3.1.3.1 enqueue()

```
bool Pool::enqueue (
            Task && task )
```

Enqueue a Task for the thread pool.

Parameters

task	A rvalue representing the Task to be enqueued.

Returns

True on success, False on failure.

3.1.3.2 getRunningThreadCount()

```
size_t Pool::getRunningThreadCount ( ) const
```

Returns the number of threads that are running.

Returns

The number of threads that are running.

3.1.3.3 getTaskQueueCount()

```
size_t Pool::getTaskQueueCount ( )
```

Returns the number of tasks currently in the task queue.

Returns

The number of tasks currently in the task queue.

3.1.3.4 getTerminatedThreadCount()

```
size_t Pool::getTerminatedThreadCount ( ) const
```

Returns the number of threads that are terminated.

Returns

The number of threads that are terminated.

3.1.3.5 getThreadCount()

```
size_t Pool::getThreadCount ( ) const
```

Returns the number of threads that are created.

Returns

The number of threads that are created.

3.1.3.6 getWaitingThreadCount()

```
size_t Pool::getWaitingThreadCount ( ) const
```

Returns the number of threads that are waiting.

Returns

The number of threads that are waiting.

3.1.3.7 setThreadCount()

```
void Pool::setThreadCount (
            size_t threadCount )
```

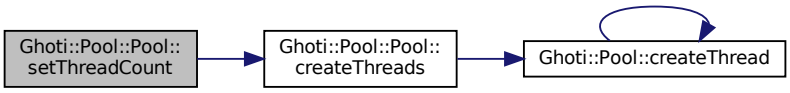
Set the thread count.

If the pool is not running, then no threads will be created. If the pool is running and the number specified is higher than the current pool size, then new threads will be created. If the number specified is lower than the current pool size, then threads will be removed as they finish their tasks. Threads will not be interrupted.

Parameters

threadCount	The desired thread count.

Here is the call graph for this function:



3.1.3.8 start()

```
void Pool::start ( )
```

Start the thread pool processing.

Will create threads as needed. Here is the call graph for this function:

3.1.3.9 stop()

```
void Pool::stop ( )
```

Stop the thread pool from dispatching new tasks and remove the existing threads.

Note: This will not halt any currently processing thread. It will only keep that thread from accepting a new Task from the queue.

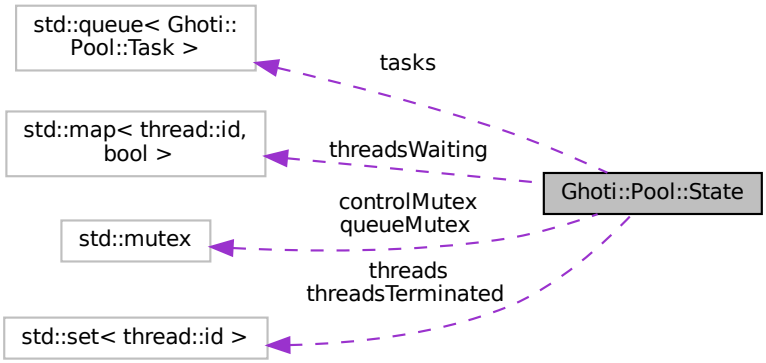
The documentation for this class was generated from the following files:

- include/pool.hpp
- src/pool.cpp

3.2 Ghoti::Pool::State Struct Reference

Structure to hold the state of the pool.

Collaboration diagram for Ghoti::Pool::State:



Public Attributes

- mutex queueMutex

  Mutex to control access to the Queue.
- mutex controlMutex

  Mutex to control access to the threadsWaiting map.
- set< thread::id > threads

  Collection of available threads.
- map< thread::id, bool > threadsWaiting

  Track the waiting state of each thread.
- set< thread::id > threadsTerminated

  Track the threads that have been terminated.
- queue< Task > tasks

  Queue of tasks waiting to be assigned to a thread.
- bool terminate

  Indicates whether or not the threads should terminate.
- std::condition_variable mutexCondition

  Allows threads to wait on new tasks or termination.
- size_t targetThreadCount

  The number of threads that the pool should manage.

3.2.1 Detailed Description

Structure to hold the state of the pool.

If the pool object is destroyed, then the threads must exit safely, but in order to do so, the synchronization mutexes and queues must still exist. To accomplish this, the State is provided to each thread as a shared pointer. As such, the State will be destroyed when the object pool and all associated threads are destroyed.

3.2.2 Member Data Documentation

3.2.2.1 targetThreadCount

```
size_t Ghoti::Pool::State::targetThreadCount
```

The number of threads that the pool should manage.

This defaults to the number of logical cores on the system.

The documentation for this struct was generated from the following file:

- src/pool.cpp

3.3 Ghoti::Pool::Task Struct Reference

Holds information about a task.

`#include <pool.hpp>`

Public Attributes

- std::function< void()> **function**

3.3.1 Detailed Description

Holds information about a task.

The documentation for this struct was generated from the following file:

- include/pool.hpp

Chapter 4

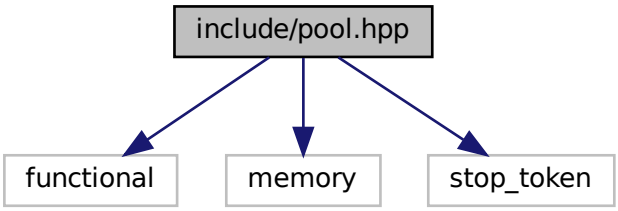
File Documentation

4.1 include/pool.hpp File Reference

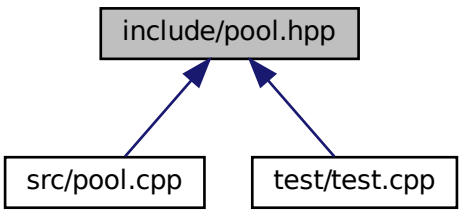
Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

```
#include <functional>
#include <memory>
#include <stop_token>
```

Include dependency graph for pool.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct Ghoti::Pool::Task

  Holds information about a task.
- class Ghoti::Pool::Pool

  Represents a generalized thread pool.

Typedefs

- using Ghoti::Pool::ThreadFunction = std::function< void(std::stop_token)>

  Function type used to create the thread pool threads.

Functions

- std::thread::id Ghoti::Pool::createThread (ThreadFunction func)

  Function that will ask the global thread pool to create an additional thread.
- void Ghoti::Pool::joinGlobalPool ()

  Function that must be called in order to terminate and join the Global thread pool.
- size_t Ghoti::Pool::getGlobalPoolThreadCount ()

  Get the total number of threads being tracked by the global thread pool.

4.1.1 Detailed Description

Header file supplied for use by 3rd party code so that they can easily include all necessary headers.

4.1.2 Function Documentation

4.1.2.1 createThread()

```
thread::id Ghoti::Pool::createThread (
            ThreadFunction func )
```

Function that will ask the global thread pool to create an additional thread.

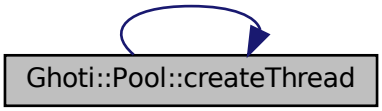
Parameters

func	The function which will be provided to the thread for execution.

Returns

The id of the thread that was created.

Here is the call graph for this function:



4.1.2.2 getGlobalPoolThreadCount()

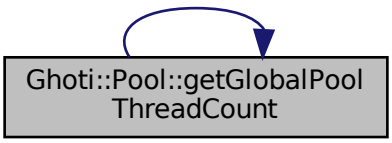
```
size_t Ghoti::Pool::getGlobalPoolThreadCount ( )
```

Get the total number of threads being tracked by the global thread pool.

Returns

The number of threads being tracked by the global thread pool.

Here is the call graph for this function:



4.1.2.3 joinGlobalPool()

```
void Ghoti::Pool::joinGlobalPool ( )
```

Function that must be called in order to terminate and join the Global thread pool.

The global thread pool must be ended before the program can terminate. The pool will terminate automatically, on its own, when all of its threads have self-terminated. It may be, however, that the threads do not know that they need to terminate.

This function will asynchronously request that all threads stop, and then block until all threads join.

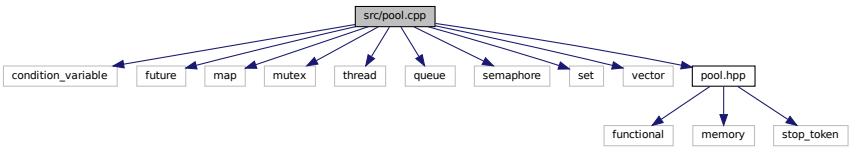
The main program will not end until all threads have terminated. It may not be necessary to call this function explicitly, depending on the design of your program. Here is the call graph for this function:

4.2 src/pool.cpp File Reference

Code for the Pool thread pool.

```
#include <condition_variable>
#include <future>
#include <map>
#include <mutex>
#include <thread>
#include <queue>
#include <semaphore>
#include <set>
#include <vector>
#include "pool.hpp"
```

Include dependency graph for pool.cpp:



Classes

- struct Ghoti::Pool::State

  Structure to hold the state of the pool.

Typedefs

- using Ghoti::Pool::ThreadInfo = pair< function< void()>, vector< promise< void > >>

  Container to hold the instance of the thread as well as a collection of promises that must be fulfilled when the thread terminates.

Functions

- std::thread::id Ghoti::Pool::createThread (ThreadFunction func)

  Function that will ask the global thread pool to create an additional thread.
- void Ghoti::Pool::joinGlobalPool ()

  Function that must be called in order to terminate and join the Global thread pool.
- size_t Ghoti::Pool::getGlobalPoolThreadCount ()

  Get the total number of threads being tracked by the global thread pool.

4.2.1 Detailed Description

Code for the Pool thread pool.

4.2.2 Function Documentation

4.2.2.1 createThread()

```
thread::id Ghoti::Pool::createThread (
            ThreadFunction func )
```

Function that will ask the global thread pool to create an additional thread.

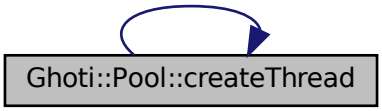
Parameters

func	The function which will be provided to the thread for execution.

Returns

The id of the thread that was created.

Here is the call graph for this function:



4.2.2.2 getGlobalPoolThreadCount()

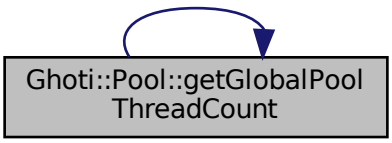
```
size_t Ghoti::Pool::getGlobalPoolThreadCount ( )
```

Get the total number of threads being tracked by the global thread pool.

Returns

The number of threads being tracked by the global thread pool.

Here is the call graph for this function:



4.2.2.3 joinGlobalPool()

```
void Ghoti::Pool::joinGlobalPool ( )
```

Function that must be called in order to terminate and join the Global thread pool.

The global thread pool must be ended before the program can terminate. The pool will terminate automatically, on its own, when all of its threads have self-terminated. It may be, however, that the threads do not know that they need to terminate.

This function will asynchronously request that all threads stop, and then block until all threads join.

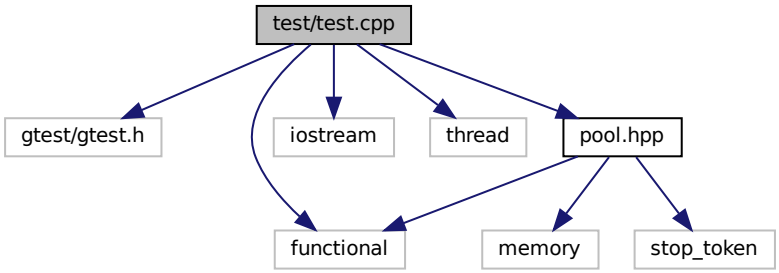
The main program will not end until all threads have terminated. It may not be necessary to call this function explicitly, depending on the design of your program. Here is the call graph for this function:

4.3 test/test.cpp File Reference

Test the general thread pool behavior.

```
#include <gtest/gtest.h>
#include <functional>
#include <iostream>
#include <thread>
#include "pool.hpp"
```

Include dependency graph for test.cpp:



Functions

- **TEST** (JoinGlobalPool, Repeated)
- **TEST** (JoinGlobalPool, AsJoin)
- **TEST** (Pool, IndependentThreads)
- **TEST** (PoolSize, Default)
- **TEST** (PoolSize, Specified)
- **TEST** (TaskQueue, Count)
- **TEST** (StopJoin, Compare)
- int **main** (int argc, char ∗∗argv)

Variables

- function **emptyFunc** = [ ](){}
- function **threadSleep**

4.3.1 Detailed Description

Test the general thread pool behavior.

4.3.2 Variable Documentation

4.3.2.1 threadSleep

```
function threadSleep
```

Initial value:

```
= [](chrono::milliseconds duration){
  return [=](){
    this_thread::sleep_for(duration);
  };
}
```

Index