XiTAO Documentation

Generated by Doxygen 1.8.13

Contents

1 Hierarchical Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 VecAdd Class Reference 7

4.1.1 Detailed Description 7

4.1.2 Constructor & Destructor Documentation 7

4.1.2.1 VecAdd() 7

4.1.3 Member Function Documentation 8

4.1.3.1 cleanup() 8

4.1.3.2 execute() 8

4.1.4 Member Data Documentation 8

4.1.4.1 in 8

4.1.4.2 len 8

4.1.4.3 out 9

4.2 VecMulDyn Class Reference 9

4.2.1 Detailed Description 9

4.2.2 Constructor & Destructor Documentation 9

4.2.2.1 VecMulDyn() 9

4.2.3 Member Function Documentation 10
4.2.3.1 cleanup() 10
4.2.3.2 execute() 10
4.2.4 Member Data Documentation 10
4.2.4.1 A 10
4.2.4.2 B 11
4.2.4.3 blocks 11
4.2.4.4 blocksize 11
4.2.4.5 C 11
4.2.4.6 len 11
4.2.4.7 next 11
4.3 VecMulSta Class Reference 11
4.3.1 Detailed Description 12
4.3.2 Constructor & Destructor Documentation 12
4.3.2.1 VecMulSta() 12
4.3.3 Member Function Documentation 13
4.3.3.1 cleanup() 13
4.3.3.2 execute() 13
4.3.4 Member Data Documentation 13
4.3.4.1 A 13
4.3.4.2 B 13
4.3.4.3 block 13
4.3.4.4 C 14
4.3.4.5 len 14

5 File Documentation **15**

5.1 taos_dotproduct.h File Reference 15

5.2 xitao_api.h File Reference 15

5.2.1 Detailed Description 16

5.2.2 Macro Definition Documentation 16

5.2.2.1 goTAO_fini 16

5.2.2.2 goTAO_init 16

5.2.2.3 goTAO_push 16

5.2.2.4 goTAO_start 16

5.2.3 Function Documentation 16

5.2.3.1 gotao_barrier() 16

5.2.3.2 gotao_fini() 17

5.2.3.3 gotao_init() 17

5.2.3.4 gotao_init_hw() 17

5.2.3.5 gotao_push() 17

5.2.3.6 gotao_start() 18

6 Example Documentation **19**

6.1 dotprod.cxx 19

Index **21**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AssemblyTask
- VecAdd 7
- VecMulDyn 9
- VecMulSta 11

Chapter 2

Class Index

2.1 Class List

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

VecAdd 7
VecMulDyn 9
VecMulSta 11

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

taos_dotproduct.h
Contains the TAOs needed for the dot product example 15
xitao_api.h
Interfaces to the XiTAO Runtime 15

Chapter 4

Class Documentation

4.1 VecAdd Class Reference

```
#include <taos_dotproduct.h>
```

Public Member Functions

- VecAdd (double ∗_in, double ∗_out, int _len, int width)

  VecAdd TAO constructor.
- int cleanup ()

  Inherited pure virtual function that is called by the runtime to cleanup any resources (if any), held by a TAO.
- int execute (int threadid)

  Inherited pure virtual function that is called by the runtime upon executing the TAO.

Public Attributes

- double ∗ in
- double ∗ out
- int len

4.1.1 Detailed Description

this TAO will take a set of doubles and add them all together

Examples:

dotprod.cxx.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 VecAdd()

```
VecAdd::VecAdd (
              double ∗ _in,
              double ∗ _out,
              int _len,
              int width )  [inline]
```

VecAdd TAO constructor.

Parameters

_in	is the input vector for which the elements should be accumulated
_out	is the output element holding the summation
_len	is the length of the vector
width	is the number of resources used by this TAO

4.1.3 Member Function Documentation

4.1.3.1 cleanup()

```
int VecAdd::cleanup ( )  [inline]
```

Inherited pure virtual function that is called by the runtime to cleanup any resources (if any), held by a TAO.

4.1.3.2 execute()

```
int VecAdd::execute (
            int threadid )  [inline]
```

Inherited pure virtual function that is called by the runtime upon executing the TAO.

Parameters

threadid	logical thread id that executes the TAO. For this TAO, we let logical core 0 only do the addition to avoid reduction

4.1.4 Member Data Documentation

4.1.4.1 in

```
double* VecAdd::in
```

TAO implementation specific double vector that holds the input to be accumulated

4.1.4.2 len

```
int VecAdd::len
```

TAO implementation specific integer that holds the number of elements

4.1.4.3 out

```
double∗ VecAdd::out
```

TAO implementation specific double point to the summation

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

- taos_dotproduct.h

4.2 VecMulDyn Class Reference

```
#include <taos_dotproduct.h>
```

Public Member Functions

- VecMulDyn (double ∗_A, double ∗_B, double ∗_C, int _len, int width)

    VecMulDyn TAO constructor.
- int cleanup ()

    Inherited pure virtual function that is called by the runtime to cleanup any resources (if any), held by a TAO.
- int execute (int threadid)

    Inherited pure virtual function that is called by the runtime upon executing the TAO.

Public Attributes

- int blocks
- int blocksize
- int len
- double ∗ A
- double ∗ B
- double ∗ C
- atomic< int > next

4.2.1 Detailed Description

this TAO will take two vectors and multiply them. This TAO implements internal dynamic scheduling.

Examples:

dotprod.cxx.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 VecMulDyn()

```
VecMulDyn::VecMulDyn (
              double ∗ _A,
              double ∗ _B,
              double ∗ _C,
              int _len,
              int width )  [inline]
```

VecMulDyn TAO constructor.

Parameters

_A	is the A vector
_B	is the B vector
_C	is the Result vector
_len	is the length of the vector
width	is the number of resources used by this TAO The constructor computes the number of elements per thread and overdecomposes the domain using PSLACK parameter In this simple example, we do not instatiate a dynamic scheduler (yet)

4.2.3 Member Function Documentation

4.2.3.1 cleanup()

```
int VecMulDyn::cleanup ( )  [inline]
```

Inherited pure virtual function that is called by the runtime to cleanup any resources (if any), held by a TAO.

4.2.3.2 execute()

```
int VecMulDyn::execute (
            int threadid )  [inline]
```

Inherited pure virtual function that is called by the runtime upon executing the TAO.

Parameters

threadid	logical thread id that executes the TAO This assembly can work totally asynchronously

4.2.4 Member Data Documentation

4.2.4.1 A

```
double* VecMulDyn::A
```

TAO implementation specific double array that holds the A vector

4.2.4.2 B

```
double* VecMulDyn::B
```

TAO implementation specific double array that holds the B vector

4.2.4.3 blocks

```
int VecMulDyn::blocks
```

TAO implementation specific integer that holds the number of blocks per TAO

4.2.4.4 blocksize

```
int VecMulDyn::blocksize
```

TAO implementation specific integer that holds the number of elements per block

4.2.4.5 C

```
double* VecMulDyn::C
```

TAO implementation specific double array that holds the result vector

4.2.4.6 len

```
int VecMulDyn::len
```

TAO implementation specific integer that holds the vector length

4.2.4.7 next

```
atomic<int> VecMulDyn::next
```

TAO implementation specific atomic variable to provide thread safe tracker of the number of processed blocks

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

- taos_dotproduct.h

4.3 VecMulSta Class Reference

```
#include <taos_dotproduct.h>
```

Public Member Functions

- VecMulSta (double ∗_A, double ∗_B, double ∗_C, int _len, int width)

  VecMulSta TAO constructor.
- int cleanup ()

  Inherited pure virtual function that is called by the runtime to cleanup any resources (if any), held by a TAO.
- int execute (int threadid)

  Inherited pure virtual function that is called by the runtime upon executing the TAO.

Public Attributes

- int block
- int len
- double ∗ A
- double ∗ B
- double ∗ C

4.3.1 Detailed Description

this TAO will take two vectors and multiply them. This TAO implements internal static scheduling.

Examples:

dotprod.cxx.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 VecMulSta()

```
VecMulSta::VecMulSta (
            double ∗ _A,
            double ∗ _B,
            double ∗ _C,
            int _len,
            int width )  [inline]
```

VecMulSta TAO constructor.

Parameters

_A	is the A vector
_B	is the B vector
_C	is the Result vector
_len	is the length of the vector
width	is the number of resources used by this TAO The constructor computes the number of elements per thread. In this simple example, we do not instatiate a dynamic scheduler (yet)

4.3.3 Member Function Documentation

4.3.3.1 cleanup()

```
int VecMulSta::cleanup ( )  [inline]
```

Inherited pure virtual function that is called by the runtime to cleanup any resources (if any), held by a TAO.

4.3.3.2 execute()

```
int VecMulSta::execute (
             int threadid )  [inline]
```

Inherited pure virtual function that is called by the runtime upon executing the TAO.

Parameters

threadid	logical thread id that executes the TAO

4.3.4 Member Data Documentation

4.3.4.1 A

```
double* VecMulSta::A
```

TAO implementation specific double array that holds the A vector

4.3.4.2 B

```
double* VecMulSta::B
```

TAO implementation specific double array that holds the B vector

4.3.4.3 block

```
int VecMulSta::block
```

TAO implementation specific integer that holds the number of blocks per TAO

4.3.4.4 C

```
double* VecMulSta::C
```

TAO implementation specific double array that holds the result vector

4.3.4.5 len

```
int VecMulSta::len
```

TAO implementation specific integer that holds the vector length

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

- taos_dotproduct.h

Chapter 5

File Documentation

5.1 taos_dotproduct.h File Reference

Contains the TAOs needed for the dot product example.

```
#include "tao.h"
#include <chrono>
#include <iostream>
#include <atomic>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

Include dependency graph for taos_dotproduct.h:

5.2 xitao_api.h File Reference

Interfaces to the XiTAO Runtime.

Macros

- #define goTAO_init gotao_init
- #define goTAO_start gotao_start
- #define goTAO_fini gotao_fini
- #define goTAO_push gotao_push

Functions

- int gotao_init_hw (int nthr, int thrb, int nhwc)

  Initialize the XiTAO Runtime.
- int gotao_init ()

  Initialize the XiTAO Runtime using the environment variables GOTAO_NTHREADS, GOTAO_THREAD_BASE and GOTAO_HW_CONTEXTS respectively.
- int gotao_start ()

  Triggers the start of the TAODAG execution.
- int gotao_fini ()

  Finalize the runtime and makes sure that all workers have finished.
- int gotao_push (PolyTask ∗pt, int queue=-1)
- void gotao_barrier ()

  Block master thread until DAG execution is finished, without having to finalize.

5.2.1 Detailed Description

Interfaces to the XiTAO Runtime.

This header define the interfaces of the runtime initialization, preperation of the TAODAG, finalization, etc.

5.2.2 Macro Definition Documentation

5.2.2.1 goTAO_fini

```
#define goTAO_fini gotao_fini
```

5.2.2.2 goTAO_init

```
#define goTAO_init gotao_init
```

5.2.2.3 goTAO_push

```
#define goTAO_push gotao_push
```

5.2.2.4 goTAO_start

```
#define goTAO_start gotao_start
```

5.2.3 Function Documentation

5.2.3.1 gotao_barrier()

```
void gotao_barrier ( )
```

Block master thread until DAG execution is finished, without having to finalize.

5.2.3.2 gotao_fini()

```
int gotao_fini ( )
```

Finalize the runtime and makes sure that all workers have finished.

Examples:

dotprod.cxx.

5.2.3.3 gotao_init()

```
int gotao_init ( )
```

Initialize the XiTAO Runtime using the environment variables GOTAO_NTHREADS, GOTAO_THREAD_BASE and GOTAO_HW_CONTEXTS respectively.

Examples:

dotprod.cxx.

5.2.3.4 gotao_init_hw()

```
int gotao_init_hw (
             int nthr,
             int thrb,
             int nhwc )
```

Initialize the XiTAO Runtime.

Parameters

nthr	The number of XiTAO threads
thrb	The logical thread id offset from the physical core mapping
nhwc	The number of hardware contexts

5.2.3.5 gotao_push()

```
int gotao_push (
             PolyTask * pt,
             int queue = -1 )
```

Push work into Polytask queue. if no particular queue is specified then try to determine which is the local. queue and insert it there. This has some overhead, so in general the programmer should specify some queue

Parameters

pt	The TAO to push
queue	The queue to be pushed to (< GOTAO_NTHREADS)

Examples:

dotprod.cxx.

5.2.3.6 gotao_start()

```
int gotao_start ( )
```

Triggers the start of the TAODAG execution.

Examples:

dotprod.cxx.

Chapter 6

Example Documentation

6.1 dotprod.cxx

An example parallization of 2 vectors dot product using XiTAO

```
#include "taos_dotproduct.h"
int
main(int argc, char *argv[])
{
  double *A, *B, *C, D;
  if(argc != 4) {
    std::cout << "./a.out <veclength> <TAOwidth> <blocklength>" << std::endl;
    return 0;
  }

  int len = atoi(argv[1]);
  int width = atoi(argv[2]);
  int block = atoi(argv[3]);

  // For simplicity, only support only perfect partitions
  if(len % block){
    std::cout << "len is not a multiple of block!" << std::endl;
    return 0;
  }

   // no topologies in this version
  A = new double[len];
  B = new double[len];
  C = new double[len];

  // initialize the vectors with some numbers
  for(int i = 0; i < len; i++){
    A[i] = (double) (i+1);
    B[i] = (double) (i+1);
  }

  // init XiTAO runtime
  gotao_init();

  // create numvm TAOs
  int numvm = len / block;

  // static or dynamic internal TAO scheduler
#ifdef STATIC
  VecMulSta *vm[numvm];
#else
  VecMulDyn *vm[numvm];
#endif
  VecAdd *va = new VecAdd(C, &D, len, width);

  // Create the TAODAG
  for(int j = 0; j < numvm; j++){
#ifdef STATIC
    vm[j] = new VecMulSta(A+j*block, B+j*block, C+j*block, block, width);
#else
    vm[j] = new VecMulDyn(A+j*block, B+j*block, C+j*block, block, width);
#endif
    //Create an edge
    vm[j]->make_edge(va);
    //Push current root to assigned queue
    gotao_push(vm[j], j % gotao_nthreads);
  }
  //Start the TAODAG exeuction
  gotao_start();

  //Finalize and claim resources back
  gotao_fini();

  std::cout << "Result is " << D << std::endl;
  std::cout << "Done!\n";
  std::cout << "Total successful steals: " << tao_total_steals << std::endl;
}
```

Index

A
    VecMulDyn, 10
    VecMulSta, 13

B
    VecMulDyn, 10
    VecMulSta, 13
block
    VecMulSta, 13
blocks
    VecMulDyn, 11
blocksize
    VecMulDyn, 11

C
    VecMulDyn, 11
    VecMulSta, 13
cleanup
    VecAdd, 8
    VecMulDyn, 10
    VecMulSta, 13

execute
    VecAdd, 8
    VecMulDyn, 10
    VecMulSta, 13

goTAO_fini
    xitao_api.h, 16
goTAO_init
    xitao_api.h, 16
goTAO_push
    xitao_api.h, 16
goTAO_start
    xitao_api.h, 16
gotao_barrier
    xitao_api.h, 16
gotao_fini
    xitao_api.h, 16
gotao_init
    xitao_api.h, 17
gotao_init_hw
    xitao_api.h, 17
gotao_push
    xitao_api.h, 17
gotao_start
    xitao_api.h, 18

in
    VecAdd, 8

len
    VecAdd, 8
    VecMulDyn, 11
    VecMulSta, 14

next
    VecMulDyn, 11

out
    VecAdd, 8

taos_dotproduct.h, 15

VecAdd, 7
    cleanup, 8
    execute, 8
    in, 8
    len, 8
    out, 8
    VecAdd, 7
VecMulDyn, 9
    A, 10
    B, 10
    blocks, 11
    blocksize, 11
    C, 11
    cleanup, 10
    execute, 10
    len, 11
    next, 11
    VecMulDyn, 9
VecMulSta, 11
    A, 13
    B, 13
    block, 13
    C, 13
    cleanup, 13
    execute, 13
    len, 14
    VecMulSta, 12

xitao_api.h, 15
    goTAO_fini, 16
    goTAO_init, 16
    goTAO_push, 16
    goTAO_start, 16
    gotao_barrier, 16
    gotao_fini, 16
    gotao_init, 17
    gotao_init_hw, 17
    gotao_push, 17

gotao_start, 18