ReuseDistance

0.01

Generated by Doxygen 1.6.1

Wed Oct 17 23:23:23 2012

Contents

Chapter 1

Class Index

1.1 Class List

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

node234_Tag . . . . . ??
ReuseDistance . . . . . ??
ReuseEntry . . . . . ??
ReuseStats . . . . . ??
tree234_Tag . . . . . ??

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/home/michaell/software/ReuseDistance/ReuseDistance.cpp . . . . . ??
/home/michaell/software/ReuseDistance/ReuseDistance.hpp . . . . . ??
/home/michaell/software/ReuseDistance/tree234.c . . . . . ??
/home/michaell/software/ReuseDistance/tree234.h . . . . . ??

Chapter 3

Class Documentation

3.1 node234_Tag Struct Reference

Public Attributes

- node234 * parent
- node234 * kids [4]
- int counts [4]
- void * elems [3]

3.1.1 Detailed Description

Definition at line 52 of file tree234.c.

3.1.2 Member Data Documentation

3.1.2.1 int node234_Tag::counts[4]

Definition at line 55 of file tree234.c.

3.1.2.2 void* node234_Tag::elems[3]

Definition at line 56 of file tree234.c.

3.1.2.3 node234* node234_Tag::kids[4]

Definition at line 54 of file tree234.c.

3.1.2.4 node234* node234_Tag::parent

Definition at line 53 of file tree234.c.

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

- /home/michaell/software/ReuseDistance/tree234.c

3.2 ReuseDistance Class Reference

`#include <ReuseDistance.hpp>`

Public Member Functions

- ReuseDistance (uint64_t w, uint64_t b)
- ReuseDistance (uint64_t w)
- ~ReuseDistance ()
- void Print (std::ostream &f)
- void Print ()
- void Process (ReuseEntry &addr)
- void Process (ReuseEntry *addrs, uint64_t count)
- void Process (std::vector< ReuseEntry > rs)
- void Process (std::vector< ReuseEntry * > addrs)
- uint64_t GetDistance (ReuseEntry &addr)
- ReuseStats * GetStats (uint64_t id)
- void GetIndices (std::vector< uint64_t > &ids)
- void GetActiveAddresses (std::vector< uint64_t > &addrs)

Static Public Attributes

- static const uint64_t DefaultBinIndividual = 32
- static const uint64_t Infinity = 0

3.2.1 Detailed Description

Tracks reuse distances for a memory address stream. Keep track of the addresses within a specific window of history, whose size can be finite or infinite. For basic usage, see the documentation for the constructors, the Process methods and the Print methods. Also see the simple test file test/test.cpp included in the source package.

Definition at line 171 of file ReuseDistance.hpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ReuseDistance::ReuseDistance (uint64_t *w*, uint64_t *b*)

Contructs a ReuseDistance object.

Parameters:

w The maximum window size, or alternatively the maximum possible reuse distance that this tool will find. No window/distance limit is imposed if ReuseDistance::Infinity is used, though you could easily run of of memory.

b All distances not greater than b will be tracked individually. All distances are tracked individually if b == ReuseDistance::Infinity. Beyond individual tracking, distances are tracked in bins whose boundaries are the powers of two greater than b (and not exeeding w, of course).

Definition at line 49 of file ReuseDistance.cpp.

3.2.2.2 ReuseDistance::ReuseDistance (uint64_t *w*)

Contructs a ReuseDistance object. Equivalent to calling the other constructor with b == ReuseDistance::DefaultBinIndividual

Parameters:

w The maximum window size, or alternatively the maximum possible reuse distance that this tool will find. No window/distance limit if ReuseDistance::Infinity is used, though you could easily run out of memory.

Definition at line 53 of file ReuseDistance.cpp.

3.2.2.3 ReuseDistance::~ReuseDistance ()

Destroys a ReuseDistance object.

Definition at line 57 of file ReuseDistance.cpp.

3.2.3 Member Function Documentation

3.2.3.1 void ReuseDistance::GetActiveAddresses (std::vector< uint64_t > & *addrs*)

Get a std::vector containing all of the addresses currently in this ReuseDistance object's active window.

Parameters:

addrs A std::vector which will contain the addresses. It is an error to pass this vector non-empty (that is addrs.size() == 0 is enforced at runtime).

Returns:

none

Definition at line 79 of file ReuseDistance.cpp.

3.2.3.2 uint64_t ReuseDistance::GetDistance (ReuseEntry & *addr*)

Get a reuse distance for a ReuseEntry without updating any internal state.

Parameters:

addr The memory address to analyze.

Returns:

The reuse distance for the memory address given by addr.

Definition at line 135 of file ReuseDistance.cpp.

3.2.3.3 void ReuseDistance::GetIndices (std::vector< uint64_t > & *ids*)

Get a std::vector containing all of the unique indices processed by this ReuseDistance object.

Parameters:

ids A std::vector which will contain the ids. It is an error to pass this vector non-empty (that is addrs.size() == 0 is enforced at runtime).

Returns:

none

Definition at line 71 of file ReuseDistance.cpp.

3.2.3.4 ReuseStats * ReuseDistance::GetStats (uint64_t *id*)

Get the ReuseStats object associated with some unique id.

Parameters:

id The unique id.

Returns:

The ReuseStats object associated with parameter id, or NULL if no ReuseStats is associate with id.

Definition at line 241 of file ReuseDistance.cpp.

3.2.3.5 void ReuseDistance::Print ()

Print statistics for this ReuseDistance to std::cout. See the other version of ReuseDistance::Print for information about output format.

Returns:

none

Definition at line 89 of file ReuseDistance.cpp.

3.2.3.6 void ReuseDistance::Print (std::ostream & *f*)

Print statistics for this ReuseDistance to an output stream. The first line of the output is five tokens which are [1] the string literal REUSESTATS, [2] the unique id, [3] the window size (0 == unlimited) [4] the total number of accesses for that unique id and [5] the number of accesses from that id which were not found within the active address window either because they were evicted or because of cold misses. Each additional line of output contains three tokens, which give [1] the minimum of a reuse distance range (inclusive), [2] the maximum of a reuse distance range (inclusive) and [2] the number of times a reusedistance in that range was observed.

Parameters:

f The output stream to print results to.

Returns:

none

3.2.3.7 void ReuseDistance::Process (std::vector< ReuseEntry * > *addrs*)

Process multiple memory addresses. Equivalent to calling Process on each element of the input vector.

Parameters:

addrs A std::vector of memory addresses to process.

Returns:

none

3.2.3.8 void ReuseDistance::Process (std::vector< ReuseEntry > *rs*)

Process multiple memory addresses. Equivalent to calling Process on each element of the input vector.

Parameters:

addrs A std::vector of memory addresses to process.

Returns:

none

3.2.3.9 void ReuseDistance::Process (ReuseEntry * *addrs*, uint64_t *count*)

Process multiple memory addresses. Equivalent to calling Process on each element of the input array.

Parameters:

addrs An array of structures describing memory addresses to process.

count The number of elements in addrs.

Returns:

none

Definition at line 115 of file ReuseDistance.cpp.

3.2.3.10 void ReuseDistance::Process (ReuseEntry & *addr*)

Process a single memory address.

Parameters:

addr The structure describing the memory address to process.

Returns:

none

Definition at line 176 of file ReuseDistance.cpp.

3.2.4 Member Data Documentation

3.2.4.1 const uint64_t ReuseDistance::DefaultBinIndividual = 32 [static]

Definition at line 197 of file ReuseDistance.hpp.

3.2.4.2 const uint64_t ReuseDistance::Infinity = 0 [static]

Definition at line 198 of file ReuseDistance.hpp.

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

- /home/michaell/software/ReuseDistance/ReuseDistance.hpp
- /home/michaell/software/ReuseDistance/ReuseDistance.cpp

3.3 ReuseEntry Struct Reference

`#include <ReuseDistance.hpp>`

Public Attributes

- uint64_t id
- uint64_t address

3.3.1 Detailed Description

ReuseEntry is used to pass memory addresses into a ReuseDistance.

id The unique id of the entity which generated the memory address. Statistics are tracked seperately for each unique id. address A memory address.

Definition at line 63 of file ReuseDistance.hpp.

3.3.2 Member Data Documentation

3.3.2.1 uint64_t ReuseEntry::address

Definition at line 65 of file ReuseDistance.hpp.

3.3.2.2 uint64_t ReuseEntry::id

Definition at line 64 of file ReuseDistance.hpp.

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

- /home/michaell/software/ReuseDistance/ReuseDistance.hpp

3.4 ReuseStats Class Reference

`#include <ReuseDistance.hpp>`

Public Member Functions

- ReuseStats ()
- ~ReuseStats ()
- void Update (uint64_t dist)
- void Miss ()
- uint64_t GetMissCount ()
- void Print (std::ostream &f, uint64_t uniqueid, uint64_t binindividual)
- void GetSortedDistances (std::vector< uint64_t > &dists)
- uint64_t GetMaximumDistance ()
- uint64_t CountDistance (uint64_t dist)
- uint64_t GetAccessCount ()

3.4.1 Detailed Description

ReuseStats holds count of observed reuse distances.

Definition at line 74 of file ReuseDistance.hpp.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 ReuseStats::ReuseStats () `[inline]`

Contructs a ReuseStats object. Default constructor.

Definition at line 85 of file ReuseDistance.hpp.

3.4.2.2 ReuseStats::~ReuseStats () `[inline]`

Destroys a ReuseStats object.

Definition at line 90 of file ReuseDistance.hpp.

3.4.3 Member Function Documentation

3.4.3.1 uint64_t ReuseStats::CountDistance (uint64_t *dist*)

Count the number of times some distance has been observed.

Parameters:

dist The distance to count.

Returns:

The number of times d has been observed.

Definition at line 270 of file ReuseDistance.cpp.

3.4.3.2 uint64_t ReuseStats::GetAccessCount ()

Count the total number of distances observed.

Returns:

The total number of distances observed.

Definition at line 245 of file ReuseDistance.cpp.

3.4.3.3 uint64_t ReuseStats::GetMaximumDistance ()

Get the maximum distance observed.

Returns:

The maximum distance observed.

Definition at line 249 of file ReuseDistance.cpp.

3.4.3.4 uint64_t ReuseStats::GetMissCount () `[inline]`

Get the number of misses. This is equal to the number of times Miss() is called plus the number of times Update(0) is called. These two calls are functionally equivalent but Miss() is faster.

Definition at line 114 of file ReuseDistance.hpp.

3.4.3.5 void ReuseStats::GetSortedDistances (std::vector< uint64_t > & *dists*)

Get a std::vector containing the distances observed, sorted in ascending order.

Parameters:

dists The vector which will hold the sorted distance values. It is an error for dists to be passed in non-empty (that is, dists.size() == 0 is enforced).

Returns:

none

3.4.3.6 void ReuseStats::Miss ()

Increment the number of misses. That is, addresses which were not found inside the active address window. This is equivalent Update(0), but is faster.

Returns:

none

Definition at line 265 of file ReuseDistance.cpp.

3.4.3.7 void ReuseStats::Print (std::ostream & *f*, uint64_t *uniqueid*, uint64_t *binindividual*)

Print a summary of the current reuse distances and counts.

Parameters:

f The stream to receive the output.

uniqueid An identifier for this ReuseStats object.

binindividual The maximum value for which bins are kept individually. Helps print things prettily.

Returns:

none

3.4.3.8 void ReuseStats::Update (uint64_t *dist*)

Increment the counter for some distance.

Parameters:

dist A reuse distance observed in the memory address stream.

Returns:

none

Definition at line 260 of file ReuseDistance.cpp.

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

- /home/michaell/software/ReuseDistance/ReuseDistance.hpp
- /home/michaell/software/ReuseDistance/ReuseDistance.cpp

3.5 tree234_Tag Struct Reference

Public Attributes

- node234 * root
- cmpfn234 cmp

3.5.1 Detailed Description

Definition at line 47 of file tree234.c.

3.5.2 Member Data Documentation

3.5.2.1 cmpfn234 tree234_Tag::cmp

Definition at line 49 of file tree234.c.

3.5.2.2 node234* tree234_Tag::root

Definition at line 48 of file tree234.c.

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

- /home/michaell/software/ReuseDistance/tree234.c

Chapter 4

File Documentation

4.1 /home/michaell/software/ReuseDistance/ReuseDistance.cpp File Reference

```
#include <ReuseDistance.hpp>
```

Defines

- #define debug_assert(...) assert(__VA_ARGS__)
- #define __seq id

Functions

- int reusecmp (void *va, void *vb)
- uint64_t ShaveBitsPwr2 (uint64_t val)

4.1.1 Define Documentation

4.1.1.1 #define __seq id

Definition at line 27 of file ReuseDistance.cpp.

4.1.1.2 #define debug_assert(...) assert(__VA_ARGS__)

Definition at line 25 of file ReuseDistance.cpp.

4.1.2 Function Documentation

4.1.2.1 int reusecmp (void * *va*, void * *vb*)

Definition at line 28 of file ReuseDistance.cpp.

4.1.2.2 uint64_t ShaveBitsPwr2 (uint64_t *val*) [inline]

Definition at line 157 of file ReuseDistance.cpp.

4.2 /home/michaell/software/ReuseDistance/ReuseDistance.hpp File Reference

```
#include <assert.h>
#include <stdlib.h>
#include <tree234.h>
#include <algorithm>
#include <iostream>
#include <ostream>
#include <set>
#include <vector>
#include <map>
```

Classes

- struct ReuseEntry
- class ReuseStats
- class ReuseDistance

Defines

- #define reuse_map_type std::map
- #define TAB "\t"
- #define ENDL "\n"

4.2.1 Detailed Description

Author:

Michael Laurenzano <michaell@sdsc.edu>

Version:

0.01

4.2.2 LICENSE

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

4.2.3 DESCRIPTION

The ReuseDistanceHandler class allows for calculation and statistic tracking for finding memory reuse distances given a stream of memory addresses and ids.

Definition in file ReuseDistance.hpp.

4.2.4 Define Documentation

4.2.4.1 #define ENDL "\n"

Definition at line 52 of file ReuseDistance.hpp.

4.2.4.2 #define reuse_map_type std::map

Definition at line 48 of file ReuseDistance.hpp.

4.2.4.3 #define TAB "\t"

Definition at line 51 of file ReuseDistance.hpp.

4.3 /home/michaell/software/ReuseDistance/tree234.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "tree234.h"
```

Classes

- struct tree234_Tag
- struct node234_Tag

Defines

- #define smalloc malloc
- #define sfree free
- #define mknew(typ) ((typ *) smalloc (sizeof (typ)))
- #define LOG(x)

Typedefs

- typedef struct node234_Tag node234

Functions

- tree234 * newtree234 (cmpfn234 cmp)
- void freetree234 (tree234 *t)
- int count234 (tree234 *t)
- void * add234 (tree234 *t, void *e)
- void * addpos234 (tree234 *t, void *e, int index)
- void * index234 (tree234 *t, int index)
- void * findrelpos234 (tree234 *t, void *e, cmpfn234 cmp, int relation, int *index)
- void * find234 (tree234 *t, void *e, cmpfn234 cmp)
- void * findrel234 (tree234 *t, void *e, cmpfn234 cmp, int relation)
- void * findpos234 (tree234 *t, void *e, cmpfn234 cmp, int *index)
- void * delpos234 (tree234 *t, int index)
- void * del234 (tree234 *t, void *e)

4.3.1 Define Documentation

4.3.1.1 #define LOG(x)

Definition at line 42 of file tree234.c.

4.3.1.2 #define mknew(typ) ((typ *) smalloc (sizeof (typ)))

Definition at line 37 of file tree234.c.

4.3.1.3 #define sfree free

Definition at line 35 of file tree234.c.

4.3.1.4 #define smalloc malloc

Definition at line 34 of file tree234.c.

4.3.2 Typedef Documentation

4.3.2.1 typedef struct node234_Tag node234

Definition at line 45 of file tree234.c.

4.3.3 Function Documentation

4.3.3.1 void* add234 (tree234 * *t*, void * *e*)

Definition at line 389 of file tree234.c.

4.3.3.2 void* addpos234 (tree234 * *t*, void * *e*, int *index*)

Definition at line 395 of file tree234.c.

4.3.3.3 int count234 (tree234 * *t*)

Definition at line 106 of file tree234.c.

4.3.3.4 void* del234 (tree234 * *t*, void * *e*)

Definition at line 896 of file tree234.c.

4.3.3.5 void* delpos234 (tree234 * *t*, int *index*)

Definition at line 891 of file tree234.c.

4.3.3.6 void* find234 (tree234 * *t*, void * *e*, cmpfn234 *cmp*)

Definition at line 550 of file tree234.c.

4.3.3.7 void* findpos234 (tree234 * *t*, void * *e*, cmpfn234 *cmp*, int * *index*)

Definition at line 556 of file tree234.c.

4.3.3.8 void* findrel234 (tree234 * *t*, void * *e*, cmpfn234 *cmp*, int *relation*)

Definition at line 553 of file tree234.c.

4.3.3.9 void* findrelpos234 (tree234 * *t*, void * *e*, cmpfn234 *cmp*, int *relation*, int * *index*)

Definition at line 446 of file tree234.c.

4.3.3.10 void freetree234 (tree234 * *t*)

Definition at line 82 of file tree234.c.

4.3.3.11 void* index234 (tree234 * *t*, int *index*)

Definition at line 407 of file tree234.c.

4.3.3.12 tree234* newtree234 (cmpfn234 *cmp*)

Definition at line 62 of file tree234.c.

4.4 /home/michaell/software/ReuseDistance/tree234.h File Reference

Typedefs

- typedef struct tree234_Tag tree234
- typedef int(∗ cmpfn234)(void ∗, void ∗)

Enumerations

- enum {
  REL234_EQ, REL234_LT, REL234_LE, REL234_GT,
  REL234_GE }

Functions

- tree234 ∗ newtree234 (cmpfn234 cmp)
- void freetree234 (tree234 ∗t)
- void ∗ add234 (tree234 ∗t, void ∗e)
- void ∗ addpos234 (tree234 ∗t, void ∗e, int index)
- void ∗ index234 (tree234 ∗t, int index)
- void ∗ find234 (tree234 ∗t, void ∗e, cmpfn234 cmp)
- void ∗ findrel234 (tree234 ∗t, void ∗e, cmpfn234 cmp, int relation)
- void ∗ findpos234 (tree234 ∗t, void ∗e, cmpfn234 cmp, int ∗index)
- void ∗ findrelpos234 (tree234 ∗t, void ∗e, cmpfn234 cmp, int relation, int ∗index)
- void ∗ del234 (tree234 ∗t, void ∗e)
- void ∗ delpos234 (tree234 ∗t, int index)
- int count234 (tree234 ∗t)

4.4.1 Typedef Documentation

4.4.1.1 typedef int(∗ cmpfn234)(void ∗, void ∗)

Definition at line 36 of file tree234.h.

4.4.1.2 typedef struct tree234_Tag tree234

Definition at line 34 of file tree234.h.

4.4.2 Enumeration Type Documentation

4.4.2.1 anonymous enum

Enumerator:

REL234_EQ

REL234_LT

REL234_LE

REL234_GT

REL234_GE

Definition at line 126 of file tree234.h.

4.4.3 Function Documentation

4.4.3.1 void* add234 (tree234 * *t*, void * *e*)

Definition at line 389 of file tree234.c.

4.4.3.2 void* addpos234 (tree234 * *t*, void * *e*, int *index*)

Definition at line 395 of file tree234.c.

4.4.3.3 int count234 (tree234 * *t*)

Definition at line 106 of file tree234.c.

4.4.3.4 void* del234 (tree234 * *t*, void * *e*)

Definition at line 896 of file tree234.c.

4.4.3.5 void* delpos234 (tree234 * *t*, int *index*)

Definition at line 891 of file tree234.c.

4.4.3.6 void* find234 (tree234 * *t*, void * *e*, cmpfn234 *cmp*)

Definition at line 550 of file tree234.c.

4.4.3.7 void* findpos234 (tree234 * *t*, void * *e*, cmpfn234 *cmp*, int * *index*)

Definition at line 556 of file tree234.c.

4.4.3.8 void* findrel234 (tree234 * *t*, void * *e*, cmpfn234 *cmp*, int *relation*)

Definition at line 553 of file tree234.c.

4.4.3.9 void* findrelpos234 (tree234 * *t*, void * *e*, cmpfn234 *cmp*, int *relation*, int * *index*)

Definition at line 446 of file tree234.c.

4.4.3.10 void freetree234 (tree234 * *t*)

Definition at line 82 of file tree234.c.

4.4.3.11 void* index234 (tree234 * *t*, int *index*)

Definition at line 407 of file tree234.c.

4.4.3.12 tree234* newtree234 (cmpfn234 *cmp*)

Definition at line 62 of file tree234.c.