ReuseDistance

0.01

Generated by Doxygen 1.6.3

Tue Sep 25 11:06:42 2012

Contents

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 files with brief descriptions:

Chapter 3

Class Documentation

3.1 ReuseDistance Class Reference

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

Public Member Functions

- ReuseDistance ()
- ReuseDistance (uint64_t w)
- ReuseDistance (ReuseDistance &h)
- ∼ReuseDistance ()
- void Print ()
- void Print (std::ostream &f)
- 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)
- uint64_t GetWindowSize ()
- void IncrementSequence (uint64_t count)
- void GetIndices (std::vector< uint64_t > &ids)
- void GetActiveAddresses (std::vector< uint64_t > &addrs)
- uint64_t GetSequenceValue (uint64_t addr)
- uint64_t GetCurrentSequence ()
- void Cleanup ()
- void SetCleanFrequency (uint64_t c)
- uint64_t GetCleanFrequency ()

Static Public Attributes

- static const uint64_t MinimumCleanFrequency = 1000000

3.1.1 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. See constructor documentation for more details.

Definition at line 185 of file ReuseDistance.hpp.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ReuseDistance::ReuseDistance ()

Contructs a ReuseDistance object. Default constructor. Uses an unlimited-size window.

Definition at line 25 of file ReuseDistance.cpp.

3.1.2.2 ReuseDistance::ReuseDistance (uint64_t *w*)

Contructs a ReuseDistance object.

Parameters

w The maximum size of the window of addresses that will be examined. Use 0 for no window, but be aware that this will use a potetially unlimited amount of memory that will be proportional to the number of unique addresses processed by this object.

Definition at line 33 of file ReuseDistance.cpp.

3.1.2.3 ReuseDistance::ReuseDistance (ReuseDistance & *h*)

Contructs a ReuseDistance object. Copy constructor. Performs a deep copy.

Parameters

h A reference to another ReuseDistance object. All state from this parameter is copied to the new ReuseDistance object, including window size, current addresses in that window and all tracked statistics.

Definition at line 41 of file ReuseDistance.cpp.

3.1.2.4 ReuseDistance::∼ReuseDistance ()

Destroys a ReuseDistance object.

Definition at line 72 of file ReuseDistance.cpp.

3.1.3 Member Function Documentation

3.1.3.1 void ReuseDistance::Cleanup ()

Clean up the address window. That is, reclaim memory for all addresses strictly outside the maximum window size. This is done for you periodically so you don't ever really need to call this yourself.

Returns

none

Definition at line 121 of file ReuseDistance.cpp.

3.1.3.2 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).

Returns

none

Definition at line 87 of file ReuseDistance.cpp.

3.1.3.3 uint64_t ReuseDistance::GetCleanFrequency () `[inline]`

Get the frequency with which Clean is called. Has no meaning when window size is unlimited.

Returns

The frequency with which Cleanup is called.

Definition at line 408 of file ReuseDistance.hpp.

3.1.3.4 uint64_t ReuseDistance::GetCurrentSequence () `[inline]`

Get this ReuseDistance object's current sequence.

Returns

This ReuseDistance object's current sequence.

Definition at line 380 of file ReuseDistance.hpp.

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

Get a reuse distance for a memory address without tracking statistics for it.

Parameters

addr The memory address to analyze.

Returns

The reuse distance for the memory address given by addr.

3.1.3.6 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).

Returns

none

Definition at line 79 of file ReuseDistance.cpp.

3.1.3.7 uint64_t ReuseDistance::GetSequenceValue (uint64_t *addr*)

Get the sequence value for an address currently in this ReuseDistance object's active window.

Parameters

addr An address. Addresses not in this object's active window will generate a return value of 0.

Returns

The sequence value for addr, or 0 if addr is not in this object's active window.

Definition at line 95 of file ReuseDistance.cpp.

3.1.3.8 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.

Definition at line 195 of file ReuseDistance.cpp.

3.1.3.9 uint64_t ReuseDistance::GetWindowSize () `[inline]`

Get the size of the window for this ReuseDistance object.

Returns

The size of the window for this ReuseDistance object.

Definition at line 328 of file ReuseDistance.hpp.

3.1.3.10 void ReuseDistance::IncrementSequence (uint64_t *count*) `[inline]`

Increment the internal sequence count for this ReuseDistance object. This has the effect of fast forwarding in the memory address stream. Possibly useful if you are using sampling on your memory address stream.

Parameters

count The amount of the increment.

Returns

none

Definition at line 339 of file ReuseDistance.hpp.

3.1.3.11 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 two tokens, which give [1] a reuse distance and [2] the number of times that reuse distance was observed.

Parameters

f The output stream to print results to.

Returns

none

3.1.3.12 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 103 of file ReuseDistance.cpp.

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

Process multiple memory addresses.

Parameters

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

Returns

none

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

Process multiple memory addresses.

Parameters

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

Returns

none

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

Process multiple memory addresses.

Parameters

addrs An array of structures describing memory addresses to process.

count The number of elements in addrs.

Returns

none

Definition at line 148 of file ReuseDistance.cpp.

3.1.3.16 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 168 of file ReuseDistance.cpp.

3.1.3.17 void ReuseDistance::SetCleanFrequency (uint64_t *c*)

Set the frequency with which Clean is called. By default this is defined using the minimum of ReuseDistance::MinimumCleanFrequency and the window size. Has no meaning when window size is unlimited.

Parameters

c The frequency with which to call Cleanup.

Returns

none

Definition at line 67 of file ReuseDistance.cpp.

3.1.4 Member Data Documentation

3.1.4.1 const uint64_t ReuseDistance::MinimumCleanFrequency = 1000000 `[static]`

Minimum value for the cleanup frequency. The cleanup frequency is set by the constructor to the maximum of this value and the window size.

Definition at line 212 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

3.2 ReuseEntry Struct Reference

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

Public Attributes

- uint64_t id
- uint64_t address

3.2.1 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 62 of file ReuseDistance.hpp.

3.2.2 Member Data Documentation

3.2.2.1 uint64_t ReuseEntry::address

Definition at line 64 of file ReuseDistance.hpp.

3.2.2.2 uint64_t ReuseEntry::id

Definition at line 63 of file ReuseDistance.hpp.

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

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

3.3 ReuseStats Class Reference

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

Public Member Functions

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

3.3.1 Detailed Description

ReuseStats holds a count of the number of times each reuse distance is observed.

Definition at line 72 of file ReuseDistance.hpp.

3.3.2 Constructor & Destructor Documentation

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

Contructs a ReuseStats object. Default constructor.

Definition at line 83 of file ReuseDistance.hpp.

3.3.2.2 ReuseStats::ReuseStats (ReuseStats & *r*)

Contructs a ReuseStats object. Copy constructor. Performs a deep copy.

Parameters

r A ReuseStats object from which to copy all state.

Definition at line 205 of file ReuseDistance.cpp.

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

Destroys a ReuseStats object.

Definition at line 95 of file ReuseDistance.hpp.

3.3.3 Member Function Documentation

3.3.3.1 uint64_t ReuseStats::CountDistance (uint64_t *low*, uint64_t *high*)

Count the number of times any distance within some range [low, high) has been observed.

Parameters

low The lower bound (inclusive) of the distance range to count.

high The upper bound (exclusive) of the distance range to count.

Returns

The number of times any distance within the range [low, high) has been observed.

Definition at line 250 of file ReuseDistance.cpp.

3.3.3.2 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 243 of file ReuseDistance.cpp.

3.3.3.3 uint64_t ReuseStats::GetAccessCount ()

Count the total number of distances observed.

Returns

The total number of distances observed.

Definition at line 218 of file ReuseDistance.cpp.

3.3.3.4 uint64_t ReuseStats::GetMaximumDistance ()

Get the maximum distance observed.

Returns

The maximum distance observed.

Definition at line 222 of file ReuseDistance.cpp.

3.3.3.5 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 119 of file ReuseDistance.hpp.

3.3.3.6 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

3.3.3.7 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 238 of file ReuseDistance.cpp.

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

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.

scale A vector holding the boundaries of bins used to aggregate the reuse distances.

Returns

none

3.3.3.9 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 233 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

Chapter 4

File Documentation

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

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

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

```
#include <assert.h>
#include <stdlib.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

This file is part of the ReuseDistance tool.

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 51 of file ReuseDistance.hpp.

4.2.4.2 #define reuse_map_type std::map

Definition at line 47 of file ReuseDistance.hpp.

4.2.4.3 #define TAB "\t"

Definition at line 50 of file ReuseDistance.hpp.