Alps

0.1.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

mainpage

1.1 ALPINiSM: Abstraction Layer for Programming Persistent Shared Memory

ALPINiSM provides a low-level abstraction layer that reliefs the user from the details of mapping, addressing, and allocating persistent shared memory (also known as fabric-attached memory). This layer can be used as a building block for building higher level abstractions and data structures such as heaps, logs, objects, etc.

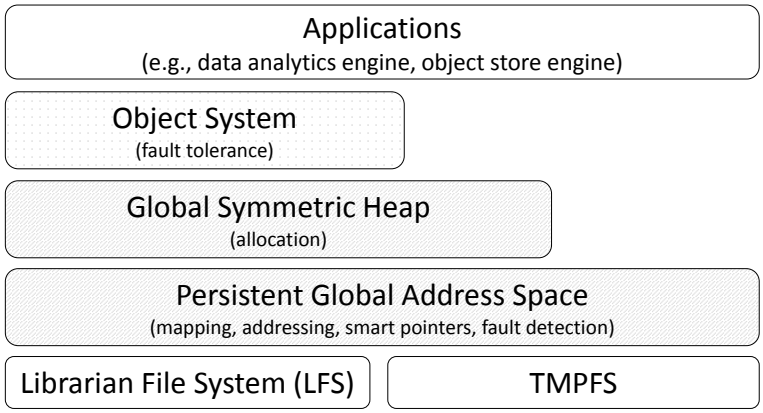


Figure 1.1 ALPINiSM layers

Currently, we provide two layers (or API classes). First, a PErsistent Global Address Space for Universal Sharing (PEGASUS) layer provides a shared address space between multiple worker processes. Second, a Global Symmetric Heap layer for allocating variable-size chunks of shared persistent memory (a.k.a. fabric-attached memory).

The APIs strive to be as generic as possible. Thus, we do not hardcode policy but instead seek providing generic mechanisms that can be used to support higher level policies.

Example Programs

ALPINiSM comes with several samples in the `examples` directory.

This Document

This document is written in Doxygen and maintained in the ALPINiSM git instance at:

git://git-pa1.labs.hpecorp.net/ssftm/alps

Generating the documentation

We include the ALPINiSM documentation as part of the source (as opposed to using a hosted wiki, such as the github wiki, as the definitive documentation) to enable the documentation to evolve along with the source code and be captured by revision control (currently git). This way the code automatically includes the version of the documentation that is relevant regardless of which version or release you have checked out or downloaded.

```
$ cd $ALPS/doc
$ doxygen
```

Reporting issues

Please report feedback, including performance and correctness issues and extension requests, through the ALPS Jira instance:

https://jira-pa1.labs.hpecorp.net/browse/ALPS/

Stable download locations

The most recent version is published at: N/A

Contact information

Author

Haris Volos haris.volos@hpe.com

Chapter 2

pegasus

2.1 PEGASUS: Persistent Global Address Space for Universal Sharing

A PErsistent Global Address Space for Universal Sharing (PEGASUS) object (alps::AddressSpace) provides a shared address space between multiple worker processes. The address space comprises multiple persistent memory regions, which are contiguous segments of the address space that are backed by fabric-attached memory (FAM). Each process has its own PEGASUS object instance that it can use to map and access shared persistent memory regions. For emulation purposes, we also provide an implementation of persistent memory regions on top of an in-memory file system (TMPFS).

Persistent regions (alps::Region) are instantiated by binding and mapping region files (alps::RegionFile) into the address space. As each process may memory map the region file at a different location, each region type provides smart pointers (Smart Pointers) for referencing locations within the region in a manner that is independent of mapping address.

Chapter 3

Todo List

Class alps::MemoryManager

Provide an API for picking address hints:

- best effort mapping: find the largest hole in the address space where we can map regions this can guide segment map to pick the right segment size
- guide underlying OS by picking an address_hint based on our past knowledge about persistent regions to minimize address space fragmentation similar to the Mnemosyne runtime (this could be implemented in the map method)

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

Module Documentation

7.1 Compiler Specific Optimizations

A few macros and functions to exploit compiler specific optimizations.

Macros

- #define LIKELY(x) (x)

  Hints that x is highly likely true. GCC's __builtin_expect.
- #define UNLIKELY(x) (x)

  Hints that x is highly likely false. GCC's __builtin_expect.
- #define NO_INLINE

  A function suffix to hint that the function should never be inlined. GCC's noinline.
- #define ALWAYS_INLINE

  A function suffix to hint that the function should always be inlined. GCC's always_inline.
- #define ASSUME_ALIGNED(x, y) x

  Wraps GCC's __builtin_assume_aligned.
- #define MAY_ALIAS

  *Wraps GCC's **attribute**((**may_alias**)).*
- #define RESTRICT_ALIAS

  Wraps GCC's __restrict.

7.1.1 Detailed Description

This file contains a few macros and functions analogous to linux/compiler.h

Example: likely/unlikely macros

For example, Linux kernel wraps GCC's __builtin_expect as follows.

```
// from linux/compiler.h.
#define likely(x)      __builtin_expect(!!(x), 1)
#define unlikely(x)    __builtin_expect(!!(x), 0)
...
if (likely(var == 42)) {
  ...
}
```

We provide macros/functions to do equivalent optimizations here. **However**, we should minimize the use of such detailed and custom optimizations. In general, compiler is good enough to predict branches/inlining, and, even if it isn't, we should rather use -fprofile-arcs. So far, we use it only for a few places that are VERY important and VERY easy to predict by human.

See also

http://blog.man7.org/2012/10/how-much-do-builtinexpect-likely-and.html

7.1.2 Macro Definition Documentation

7.1.2.1 #define MAY_ALIAS

This is *QUITE* important for us. This keyword is not for performance but for correctness. I'm seeing weird behaviors even with -fno-strict-aliasing. This keyword might help. Otherwise, we have to memcpy to type-pun everything. uggggrrr.

7.1.2.2 #define RESTRICT_ALIAS

OTOH, this explicitly helps compiler auto-vectorize and do other stuffs, saying that the variable is never aliased in the function. Seems like older gcc ignored __restrict when fno-strict-aliasing is specified, but recent gcc doesn't.

Attention

DO NOT USE THIS if you don't know what __restrict means.

7.2 C++11 Keywords in Public Headers

Defines macros for hiding C++11 features in public headers for clients that use C++98.

Macros

- #define DISABLE_CXX11_IN_PUBLIC_HEADERS

  If defined, our public headers must hide all C++11 dependent APIs.
- #define CXX11_FUNC_DELETE

  Used in public headers in place of " = delete" of C++11.
- #define CXX11_FUNC_DEFAULT

  Used in public headers in place of " = default" of C++11.
- #define CXX11_CONSTEXPR

  Used in public headers in place of "constexpr" of C++11.
- #define CXX11_FINAL

  Used in public headers in place of "final" of C++11.
- #define CXX11_NULLPTR NULL

  Used in public headers in place of "nullptr" of C++11.
- #define CXX11_NOEXCEPT

  Used in public headers in place of "noexcept" of C++11.
- #define CXX11_OVERRIDE

  Used in public headers in place of "override" of C++11.
- #define CXX11_STATIC_ASSERT(expr, message)

  Used in public headers in place of "static_assert" of C++11.

7.2.1 Detailed Description

C++11 in libalps

We basically **do assume C++11** and our library provides the best flexibility when the client program enables C++11. For example, the client program can simply contain alps-common as a subfolder and statically link to it if C++11 is enabled. However, some client program might have to stick to C++98. In that case, we provide our library as an external shared library which comes with public headers that at least compile in C++98. Thus, we will make sure C++11 keywords and classes do not directly appear in public header files. The macros defined in this file are for that switching.

DISABLE_CXX11_IN_PUBLIC_HEADERS macro

This macro is defined if __cplusplus $<$ 201103L, meaning the compiler option for the programs that include this header file (note: which is different from compiler option for libalps) disables C++11. If defined, our public headers must hide all C++11 dependent APIs. So, there are several ifdefs on this macro in public headers.

stdint.h vs cstdint

For the same reason, we include stdint.h rather than cstdint. cstdint is a C++11 extension, which defines those integer types in std namespace (eg std::int32_t). The integer types in global namespace are more concise to use, too.

C++11 in cpp and non-public1 headers

Remember, this is only for public headers. We anyway compile our library with C++11. We can freely use C++11 keywords/features in cpp and non-public header files, such as xxx_impl.hpp, and xxx_pimpl.hpp. In other words, client programs must not include them unless they turn on C++11. Also, impl/pimpl header files often include too much details for client programs to rely on. They might change in next versions.

7.2.2 Macro Definition Documentation

7.2.2.1 #define CXX11_CONSTEXPR

Note

C++98 : nothing.

7.2.2.2 #define CXX11_FINAL

Note

C++98 : nothing.

7.2.2.3 #define CXX11_FUNC_DEFAULT

Note

C++98 : nothing.

7.2.2.4 #define CXX11_FUNC_DELETE

Note

C++98 : nothing.

7.2.2.5 #define CXX11_NOEXCEPT

Note

C++98 : nothing.

7.2.2.6 #define CXX11_NULLPTR NULL

Note

C++98 : NULL.

7.2.2.7 #define CXX11_OVERRIDE

Note

C++98 : nothing.

7.2.2.8 #define CXX11_STATIC_ASSERT(*expr, message*)

Note

C++98 : nothing.

7.3 Error codes, messages, and stacktraces

Error codes (alps::ErrorCode), their error messages defined in error_code.xmacro, and stacktrace information (ErrorStack) returned by our API functions.

Classes

- class alps::ErrorStack

  Brings error stacktrace information as return value of functions.

Macros

- #define CHECK_ERROR_CODE(x)

  *This macro calls **x** and checks its returned error code. If the code is NOT kErrorCodeOk, it immediately returns from the current function or method, returning the error code code. For example, use it as follows:*
- #define CHECK_ERROR_CODE2(x, cleanup)

  *This macro calls **x** and checks its returned error code. If the code is NOT kErrorCodeOk, it executes the associated cleanup method, and immediately returns from the current function or method, returning the error code code. For example, use it as follows:*
- #define ERROR_STACK(e) alps::ErrorStack(__FILE__, __FUNCTION__, __LINE__, e)

  Instantiates ErrorStack with the given alps::error_code, creating an error stack with the current file, line, and error code.
- #define ERROR_STACK_MSG(e, m) alps::ErrorStack(__FILE__, __FUNCTION__, __LINE__, e, m)

  Overload of ERROR_STACK(e) to receive a custom error message.
- #define CHECK_ERROR(x)

  *This macro calls **x** and checks its returned value. If an error is encountered, it immediately returns from the current function or method, augmenting the stack trace held by the return code. For example, use it as follows:*
- #define WRAP_ERROR_CODE(x)

  Same as CHECK_ERROR(x) except it receives only an error code, thus more efficient.
- #define UNWRAP_ERROR_STACK(x)

  Similar to WRAP_ERROR_CODE(x), but this one converts ErrorStack to ErrorCode. This reduces information, so use it carefully.
- #define CHECK_ERROR_MSG(x, m)

  Overload of ERROR_CHECK(x) to receive a custom error message. For example, use it as follows:
- #define CHECK_OUTOFMEMORY(ptr)

  *This macro checks if **ptr** is nullptr, and if so exists with kErrorCodeOutofmemory error stack. This is useful as a null check after new/malloc. For example:*
- #define COERCE_ERROR(x)

  *This macro calls **x** and aborts if encounters an error. This should be used only in places that expects no error. For example, use it as follows:*
- #define COERCE_ERROR_CODE(x)

  Same as COERCE_ERROR(x) except this received ErrorCode, not ErrorStack.

Enumerations

- enum alps::ErrorCode { **alps::kErrorCodeOk** = 0, **X** }

  Enum of error codes defined in error_code.xmacro.

Functions

- const char * alps::get_error_name (ErrorCode code)

  Returns the names of ErrorCode enum defined in error_code.xmacro.
- const char * alps::get_error_message (ErrorCode code)

  Returns the error messages corresponding to ErrorCode enum defined in error_code.xmacro.

Variables

- const ErrorStack alps::kRetOk

  Normal return value for no-error case.

7.3.1 Detailed Description

What it is

We define all error codes and their error messages here. Whenever you want a new error message, add a new line in error_code.xmacro like existing lines. This file is completely independent and header-only. Just include this file to use.

X-Macros

To concisely define error codes, error names, and error messages, we use the so-called "X Macro" style, which doesn't require any code generation.

See also

http://en.wikipedia.org/wiki/X_Macro
http://www.drdobbs.com/the-new-c-x-macros/184401387

ErrorCode vs ErrorStack

alps::ErrorCode is merely an integer to identify the type of error. You can get a correponding error message and name of the error via get_error_name() and get_error_message(), but you can't get stacktrace information. For lightweight functions used internally, it might be enough. However, public API methods might need stacktrace information for ease of use. In that case, you should return ErrorStack, which additionally contains stacktrace and custom error message. ErrorStack is much more costly if it returns an error (if it's kErrorCodeOk, very efficient) and especially when it contains a custom error message (See ErrorStack for more details).

How to use ErrorStack

To use ErrorStack, you should be familiar with how to use the following macros: alps::kRetOk, CHECK_ERRO↩R_CODE(x), CHECK_ERROR(x), ERROR_STACK(e), COERCE_ERROR(x), and a few others. For example, use it as follows:

```
ErrorStack your_func() {
  if (out-of-memory-observed) {
     return ERROR_STACK(kErrorCodeOutofmemory);
  }
  CHECK_ERROR_CODE(another_func());
  CHECK_ERROR_CODE(yet_another_func());
  return kRetOk;
}
```

Current List of ErrorCode

See alps::ErrorCode.

7.3.2 Macro Definition Documentation

7.3.2.1 #define CHECK_ERROR(*x*)

Value:

```
{\
  alps::ErrorStack __e(x);\
  if (UNLIKELY(__e.is_error())) {\
    return alps::ErrorStack(__e, __FILE__, __FUNCTION__, __LINE__);\
  }\
}
```

```
ErrorStack your_func() {
  CHECK_ERROR(another_func());
  CHECK_ERROR(yet_another_func());
  return kRetOk;
}
```

Note

The name is CHECK_ERROR, not CHECK, because Google-logging defines CHECK.

7.3.2.2 #define CHECK_ERROR_CODE(*x*)

Value:

```
{\
  alps::ErrorCode __e = x;\
  if (UNLIKELY(__e != kErrorCodeOk)) {\
    return __e;\
  }\
}
```

```
ErrorCode another_func();
ErrorCode yet_another_func();
ErrorCode your_func() {
  CHECK_ERROR_CODE(another_func());
  CHECK_ERROR_CODE(yet_another_func());
  return kErrorCodeOk;
}
```

This macro is used in performance-critical functions that do not return ErrorStack but returns ErrorCode to save overheads. For a function that is called billion times per second, ErrorStack **does** cause bottleneck, especially because it requires to allocate hundreds bytes on stack, which would purge other data from cache lines. We actually did observe such situations in a few experiments. If your CPU profiling tells that ErrorStack-related methods cause more than 10% cpu costs, replace ErrorStack with ErrorCode.

See also

WRAP_ERROR_CODE(x)

7.3.2.3 #define CHECK_ERROR_CODE2(*x, cleanup*)

Value:

```
{\
  alps::ErrorCode __e = x;\
  if (UNLIKELY(__e != kErrorCodeOk)) {\
    cleanup; \
    return __e;\
  }\
}
```

```
ErrorCode another_func();
ErrorCode yet_another_func();
ErrorCode your_func() {
  CHECK_ERROR_CODE2(another_func(),cleanup_statement);
  CHECK_ERROR_CODE2(yet_another_func(),cleanup_statement);
  return kErrorCodeOk;
}
```

This macro is used in performance-critical functions that do not return ErrorStack but returns ErrorCode to save overheads. For a function that is called billion times per second, ErrorStack **does** cause bottleneck, especially because it requires to allocate hundreds bytes on stack, which would purge other data from cache lines. We actually did observe such situations in a few experiments. If your CPU profiling tells that ErrorStack-related methods cause more than 10% cpu costs, replace ErrorStack with ErrorCode.

See also

WRAP_ERROR_CODE(x)

7.3.2.4 #define CHECK_ERROR_MSG(*x, m*)

Value:

```
{\
  alps::ErrorStack __e(x);\
  if (UNLIKELY(__e.is_error())) {\
    return alps::ErrorStack(__e, __FILE__, __FUNCTION__, __LINE__, m);\
  }\
}
```

```
ErrorStack your_func() {
  CHECK_ERROR_MSG(another_func(), "I was doing xxx");
  CHECK_ERROR_MSG(yet_another_func(), "I was doing yyy");
  return kRetOk;
}
```

7.3.2.5 #define CHECK_OUTOFMEMORY(*ptr*)

Value:

```
if (UNLIKELY(!ptr)) {\
  return alps::ErrorStack(__FILE__, __FUNCTION__, __LINE__, kErrorCodeOutofmemory);\
}
```

```
ErrorStack your_func() {
  int* ptr = new int[123];
  CHECK_OUTOFMEMORY(ptr);
  ...
  delete[] ptr;
  return kRetOk;
}
```

7.3.2.6 #define COERCE_ERROR(x)

Value:

```
{\
  alps::ErrorStack __e(x);\
  if (UNLIKELY(__e.is_error())) {\
    __e.dump_and_abort("Unexpected error happened");\
  }\
}
```

```
void YourThread::run() {
  // the signature of thread::run() is defined elsewhere, so you can't return ErrorStack.
  // and you are sure an error won't happen here, or an error would be anyway catastrophic.
  COERCE_ERROR(another_func());
}
```

7.3.2.7 #define COERCE_ERROR_CODE(x)

Value:

```
{\
  alps::ErrorCode __e = x;\
  if (UNLIKELY(__e != alps::kErrorCodeOk)) {\
    ERROR_STACK(__e).dump_and_abort("Unexpected error happened");\
  }\
}
```

7.3.2.8 #define ERROR_STACK(e) alps::ErrorStack(__FILE__, __FUNCTION__, __LINE__, e)

For example, use it as follows:

```
ErrorStack your_func() {
  if (out-of-memory-observed) {
     return ERROR_STACK(kErrorCodeOutofmemory);
  }
  return kRetOk;
}
```

7.3.2.9 #define ERROR_STACK_MSG(e, m) alps::ErrorStack(__FILE__, __FUNCTION__, __LINE__, e, m)

For example, use it as follows:

```
ErrorStack your_func() {
  if (out-of-memory-observed) {
     std::string additional_message = ...;
     return ERROR_STACK_MSG(kErrorCodeOutofmemory, additional_message.c_str());
  }
  return kRetOk;
}
```

7.3.2.10 #define UNWRAP_ERROR_STACK(*x*)

Value:

```
{\
  alps::ErrorStack __e = x;\
  if (UNLIKELY(__e.is_error())) { return __e.get_error_code(); }\
}
```

See also

WRAP_ERROR_CODE(x)

7.3.2.11 #define WRAP_ERROR_CODE(*x*)

Value:

```
{\
  alps::ErrorCode __e = x;\
  if (UNLIKELY(__e != alps::kErrorCodeOk)) {return ERROR_STACK(__e);}\
}
```

Note

Unlike CHECK_ERROR_CODE(x), this returns ErrorStack.

See also

CHECK_ERROR_CODE(x)

7.3.3 Enumeration Type Documentation

7.3.3.1 enum alps::ErrorCode

This is often used as a return value of lightweight functions. If you need more informative information, such as error stack, use ErrorStack. But, note that returning this value is MUCH more efficient.

7.3.4 Variable Documentation

7.3.4.1 alps::kRetOk

Const return code that indicates no error. This is the normal way to return from a method or function.

7.4 Smart Pointers

Smart pointers for referencing locations within persistent regions.

Classes

- class alps::BaseRelativePointer::IPtr< RegionType, PointedType >

  Intermediate representation of a relocatable pointer.
- class alps::BaseRelativePointer::TPtr< RegionType, PointedType >

  Represents a transient pointer.
- class alps::BaseRelativePointer::PPtr< RegionType, PointedType >

  Represents a linear persistent pointer.

7.4.1 Detailed Description

Transient pointers are useful for temporarily referencing persistent memory locations.

Persistent pointers are useful for persistently storing references in persistent regions (region files).

A special null_ptr handle refers to the null pointer that points nowhere.

INVARIANTS

- Invariant 1: The virtual and linear persistent pointers do not cross regions. The invariant is met at runtime by ensuring that pointers never get assigned a memory location that is mapped to a different region.

Chapter 8

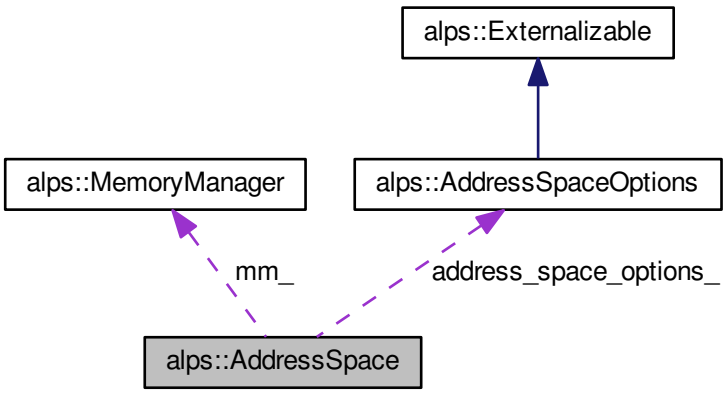
Class Documentation

8.1 alps::AddressSpace Class Reference

Logical address space.

`#include <address_space.hh>`

Collaboration diagram for alps::AddressSpace:



Public Member Functions

- **AddressSpace** (const AddressSpaceOptions &address_space_options)
- ErrorCode **init** ()
- template<class RegionT >
  ErrorCode map (RegionFile *region_file, RegionT **pregion)

  Maps a region file into the global address space.
- template<class RegionT >
  ErrorCode bind (RegionT *region, RegionId region_id)

Binds a region to a shorthand name identifier.

- template<class RegionT >
  ErrorCode unmap (RegionT *region)

  Unmaps a previously mapped region.
- Region * region (RegionId region_id)

  Returns region identified by region identifier region_id.
- ErrorCode rtrans (void *vaddr, Region **pregion, LinearAddr *offset)

  Performs reverse translation of a virtual address to a <pregion, offset> pair.
- MemoryManager * **mm** ()

Public Attributes

- AddressSpaceOptions address_space_options_

  Address space runtime options.
- MemoryManager * mm_

  Memory manager mapping regions into virtual memory.
- std::multimap< RegionId, Region * > regions_

  A table mapping region_id to region.

8.1.1 Detailed Description

Represents a logical address space where region files can be bound and mapped to. There should be a single AddressSpace object instance per process even though nothing prevents a user from having multiple instances.

Mapping and binding a region file to a global address space results in a named region. The global address space supports different modes of region mappings to allow users select different performance and flexibility tradeoffs. These modes are supported through template polymorphism rather than virtual polymorphish to reduce the runtime overhead for accessing regions.

Mapping modes include:

- single segment direct relocatable mapping, where the whole region is directly mapped to an arbitraty address. This mode enables using relative offset pointers (similar to offset_ptr) for flexible mapping and addressing at the expense of performance (i.e., some overhead to perform simple pointer arithmetic relative to the base of region).

8.1.2 Member Function Documentation

8.1.2.1 template<class RegionT > ErrorCode alps::AddressSpace::bind (RegionT * *region*, RegionId *region_id*)

Parameters

in	*region*	The region object to bind.
in	*region_id*	The shorthand name identifier to bind the region to.

After binding, pointers can identify the region using the shorthand name identifier.

8.1.2.2 template<class RegionT > template ErrorCode alps::AddressSpace::map< RRegion > (RegionFile * *region_file,* RegionT ** *pregion*)

Parameters

in	*region_file*	The region file to map into the global address space.
out	*pregion*	An object representing the region of the global address space where the file is mapped to.

8.1.2.3 Region * alps::AddressSpace::region (RegionId *region_id*)

Parameters

in	*region_id*	Region identifier

8.1.2.4 ErrorCode alps::AddressSpace::rtrans (void * *vaddr,* Region ** *pregion,* LinearAddr * *offset*)

Parameters

in	*vaddr*	The virtual address to reverse translate.
out	*pregion*	The region the virtual address is mapped to.
out	*offset*	The offset relative to the region base.

8.1.2.5 template<class RegionT > template ErrorCode alps::AddressSpace::unmap< RRegion > (RegionT * *region*)

Parameters

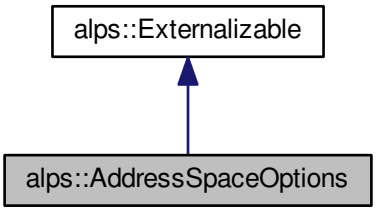
in	*region*	The region object to unmap.

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

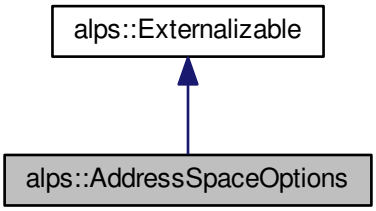
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/address_space.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/address_space.cc

8.2 alps::AddressSpaceOptions Struct Reference

Inheritance diagram for alps::AddressSpaceOptions:



Collaboration diagram for alps::AddressSpaceOptions:



Public Member Functions

- AddressSpaceOptions ()

Public Attributes

- bool **kDefaultAllowDuplicateMappings** = false
- uint64_t **allow_duplicate_mappings**

Additional Inherited Members

8.2.1 Constructor & Destructor Documentation

8.2.1.1 alps::AddressSpaceOptions::AddressSpaceOptions ( ) [inline]

Constructs option values with default values

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/address↩ _space_options.hh

8.3 alps::BacktraceContext Struct Reference

Classes

- struct GlibcBacktraceInfo
- struct LibBacktraceInfo

Public Types

- enum **Constants** { **kMaxDepth** = 64 }

Public Member Functions

- void **release** ()
- void **call_glibc_backtrace** ()
- void **on_libbt_create_state_error** (const char ∗msg, int errnum)
- void **on_libbt_full_error** (const char ∗msg, int errnum)
- void **on_libbt_full** (uintptr_t pc, const char ∗filename, int lineno, const char ∗function)
- std::vector< std::string > **get_results** (uint16_t skip)

Public Attributes

- std::string **error_**
- std::vector< GlibcBacktraceInfo > **glibc_bt_info_**
- std::vector< LibBacktraceInfo > **libbt_info_**

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/common/rich↩
_backtrace.cc

8.4 alps::BaseRelativePointer Class Reference

Classes

- class IPtr

  Intermediate representation of a relocatable pointer.
- class PPtr

  Represents a linear persistent pointer.
- class TPtr

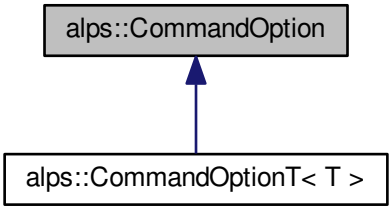
  Represents a transient pointer.
- class ZPtr

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/pointer.↩
hh

8.5 alps::CommandOption Struct Reference

Inheritance diagram for alps::CommandOption:



Public Member Functions

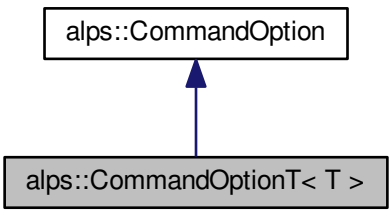
- virtual void **add_option** (boost::program_options::options_description &desc)=0
- virtual std::string **argname** ()=0
- virtual void **set_value** (const boost::program_options::variable_value &val)=0

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

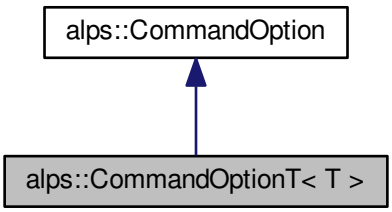
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/common/externalizable.hh

8.6 alps::CommandOptionT< T > Struct Template Reference

Inheritance diagram for alps::CommandOptionT< T >:



Collaboration diagram for alps::CommandOptionT< T >:



Public Member Functions

- **CommandOptionT** (std::string argname, T *out, std::string desc)
- std::string **argname** ()
- void **add_option** (boost::program_options::options_description &desc)
- void **set_value** (const boost::program_options::variable_value &val)

Public Attributes

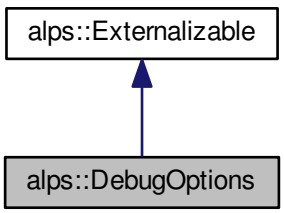
- std::string **argname_**
- T * **out_**
- std::string **desc_**

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

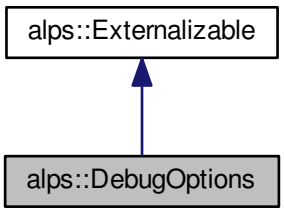
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/common/externalizable.hh

8.7 alps::DebugOptions Struct Reference

Inheritance diagram for alps::DebugOptions:



Collaboration diagram for alps::DebugOptions:



Public Member Functions

- DebugOptions ()

Public Attributes

- std::string **kDefaultLogFilename** = "sample.log"
- std::string **kDefaultLogLevel** = "error"
- std::string log_filename
- std::string log_level

Additional Inherited Members

8.7.1 Constructor & Destructor Documentation

8.7.1.1 alps::DebugOptions::DebugOptions ( ) [inline]

Constructs option values with default values

8.7.2 Member Data Documentation

8.7.2.1 std::string alps::DebugOptions::log_filename

Store log messages to this file

8.7.2.2 std::string alps::DebugOptions::log_level

Log messages at or above this level

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/common/debug↩
_options.hh

8.8 alps::ErrorStack Class Reference

Brings error stacktrace information as return value of functions.

`#include <error_stack.hh>`

Public Types

- enum Constants { kMaxStackDepth = 8 }

Public Member Functions

- ErrorStack ()
- ErrorStack (ErrorCode code)

  Instantiate a return code without a custom error message nor stacktrace.
- ErrorStack (const char *filename, const char *func, uint32_t linenum, ErrorCode code, const char *custom_message=CXX11_NULLPTR)

  Instantiate a return code with stacktrace and optionally a custom error message.
- ErrorStack (const ErrorStack &other)
- ErrorStack (const ErrorStack &other, const char *filename, const char *func, uint32_t linenum, const char *more_custom_message=CXX11_NULLPTR)
- ErrorStack & operator= (const ErrorStack &other)
- ~ErrorStack ()
- bool is_error () const
- ErrorCode get_error_code () const
- const char * get_message () const
- const char * get_custom_message () const
- void copy_custom_message (const char *message)
- void clear_custom_message ()
- void append_custom_message (const char *more_custom_message)
- uint16_t get_stack_depth () const
- uint32_t get_linenum (uint16_t stack_index) const
- const char * get_filename (uint16_t stack_index) const
- const char * get_func (uint16_t stack_index) const
- int get_os_errno () const
- void verify () const
- void output (std::ostream *ptr) const
- void dump_and_abort (const char *abort_message) const

Static Public Member Functions

- static std::string get_recent_dump_and_abort ()

Friends

- std::ostream & **operator**<< (std::ostream &o, const ErrorStack &obj)

8.8.1 Detailed Description

This is returned by many API functions. As it brings stacktrace information, it's more informative than just returning ErrorCode. However, note that instantiating and augmenting this stack object has some overhead.

Why not exception

A couple of reasons:

- Performance
- Portability
- Our Google C++ Coding Style overlord said so

We are not even sure what exceptions would look like in future environments. So, we don't throw or catch any exceptions in our program.

Macros to help use ErrorStack

In most places, you should use kRetOk, CHECK_ERROR(x), or ERROR_STACK(e) to handle this class. See the doucments of those macros.

Forced return code checking

An error code must be checked by some code, else it will abort with an "error-not-checked" error in stderr. We might later make this warning instead of aborting, but we should keep the current setting for a while to check for undesired coding. Once you get used to, making it sure is quite effortless.

Maximum stack trace depth

When the return code is an error code, we propagate back the stack trace for easier debugging. We could have a linked-list for this and, to ameriolate allocate/delete cost for it, a TLS object pool. Unfortunately, it causes issues in some environments and is not so readable/maintainable. Instead, we limit the depth of stacktraces stored in this object to a reasonable number enough for debugging; kMaxStackDepth. We then store just line numbers and const pointers to file names. No heap allocation. The only thing that has to be allocated on heap is a custom error message. However, there are not many places that use custom messages, so the cost usually doesn't happen.

Moveable/Copiable

This object is *copiable*. Further, the copy constructor and copy assignment operator are equivalent to *move*. Although they take a const reference, we *steal* its checked_ and custom_message_. This might be confusing, but much more efficient without C++11. As this object is copied so many times, we take this approach.

This class is header-only **except** output(), dump_and_abort(), and std::ostream redirect.

8.8.2 Member Enumeration Documentation

8.8.2.1 enum alps::ErrorStack::Constants

Constant values.

Enumerator

kMaxStackDepth Maximum stack trace depth.

8.8.3 Constructor & Destructor Documentation

8.8.3.1 alps::ErrorStack::ErrorStack ( ) [inline]

Empty constructor. This is same as duplicating kRetOk.

8.8.3.2 alps::ErrorStack::ErrorStack (ErrorCode *code*) [inline], [explicit]

Parameters

in	*code*	Error code, either kErrorCodeOk or real errors.

This is the most (next to kRetOk) light-weight way to create/propagate a return code. Use this one if you do not need a detail information to debug the error (eg, error whose cause is obvious, an expected error that is immediately caught, etc).

8.8.3.3 alps::ErrorStack::ErrorStack (const char * *filename*, const char * *func*, uint32_t *linenum*, ErrorCode *code*, const char * *custom_message* = CXX11_NULLPTR) [inline]

Parameters

in	*filename*	file name of the current place. It must be a const and permanent string, such as what "__FILE__" returns. Note that we do NOT do deep-copy of the strings.
in	*func*	functiona name of the current place. Must be a const-permanent as well, such as "__FUNCTION__" of gcc/MSVC or C++11's **func**.
in	*linenum*	line number of the current place. Usually "__LINE__".
in	*code*	Error code, must be real errors.
in	*custom_message*	Optional custom error message in addition to the default one inferred from error code. If you pass a non-NULL string to this argument, we do deep-copy, so it's EXPENSIVE!

8.8.3.4 alps::ErrorStack::ErrorStack (const ErrorStack & *other*) [inline]

Copy constructor.

8.8.3.5 alps::ErrorStack::ErrorStack (const ErrorStack & *other*, const char * *filename*, const char * *func*, uint32_t *linenum*, const char * *more_custom_message* = CXX11_NULLPTR) [inline]

Copy constructor to augment the stacktrace.

8.8.3.6 alps::ErrorStack::~ErrorStack ( ) [inline]

Will warn in stderr if the error code is not checked yet.

8.8.4 Member Function Documentation

8.8.4.1 void alps::ErrorStack::append_custom_message (const char * *more_custom_message*) [inline]

Appends more custom error message at the end.

8.8.4.2 void alps::ErrorStack::clear_custom_message ( ) [inline]

Deletes custom message from this object.

8.8.4.3 void alps::ErrorStack::copy_custom_message (const char * *message*) [inline]

Copy the given custom message into this object.

8.8.4.4 void alps::ErrorStack::dump_and_abort (const char * *abort_message*) const

Describe this object to std::cerr and then abort. This also leaves the dump information in static variable so that a signal handler pick it up.

8.8.4.5 const char * alps::ErrorStack::get_custom_message ( ) const [inline]

Returns the custom error message.

8.8.4.6 ErrorCode alps::ErrorStack::get_error_code ( ) const [inline]

Return the integer error code.

8.8.4.7 const char * alps::ErrorStack::get_filename (uint16_t *stack_index*) const [inline]

Returns the file name of the given stack position.

8.8.4.8 const char * alps::ErrorStack::get_func (uint16_t *stack_index*) const [inline]

Returns the function name of the given stack position.

8.8.4.9 uint32_t alps::ErrorStack::get_linenum (uint16_t *stack_index*) const [inline]

Returns the line number of the given stack position.

8.8.4.10 const char * alps::ErrorStack::get_message ( ) const [inline]

Returns the error message inferred by the error code.

8.8.4.11 int alps::ErrorStack::get_os_errno ( ) const [inline]

Global errno of the system as of instantiation of this error stack.

8.8.4.12 std::string alps::ErrorStack::get_recent_dump_and_abort ( ) [static]

Signal handler can get the dump information via this.

8.8.4.13 uint16_t alps::ErrorStack::get_stack_depth ( ) const [inline]

Returns the depth of stack this error code has collected.

8.8.4.14 bool alps::ErrorStack::is_error ( ) const [inline]

Returns if this return code is not kErrorCodeOk.

8.8.4.15 ErrorStack & alps::ErrorStack::operator= (const ErrorStack & *other*) [inline]

Assignment operator.

8.8.4.16 void alps::ErrorStack::output (std::ostream * *ptr*) const

Describe this object to the given stream.

8.8.4.17 void alps::ErrorStack::verify ( ) const [inline]

Output a warning to stderr if the error is not checked yet.

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

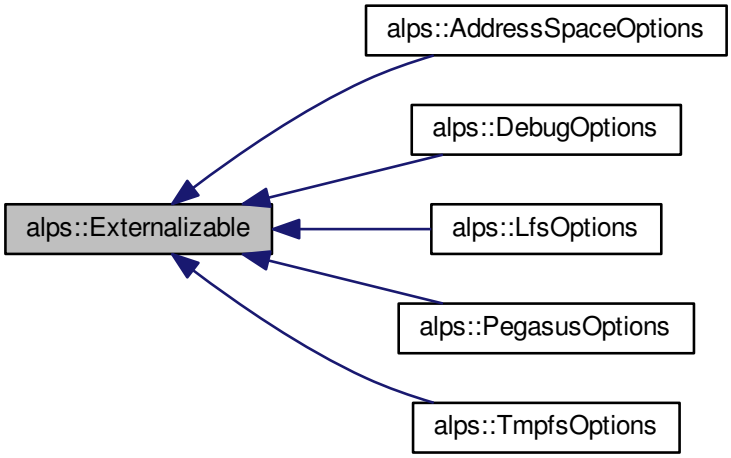
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/common/error↩ _stack.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/common/error↩ _stack.cc

8.9 alps::Externalizable Struct Reference

Represents an object that can be written to and read from files/bytes in YAML format.

`#include <externalizable.hh>`

Inheritance diagram for alps::Externalizable:



Public Member Functions

- virtual ErrorStack load (YAML::Node *node, bool ignore_missing=false)=0

  Reads the content of this object from the given YAML element.
- virtual ErrorStack save (YAML::Emitter *out) const =0

  Writes the content of this object to the given YAML element.
- virtual ErrorStack add_command_options (CommandOptionList *cmdopt)=0

  Adds command options for parsing.
- virtual const char * get_tag_name () const =0

  Returns a YAML tag name for this object as a root element.
- virtual void assign (const alps::Externalizable *other)=0

  Polymorphic assign operator. This should invoke operator= of the derived class.
- ErrorStack load_from_string (const std::string &yaml, bool ignore_missing=false)

  Load the content of this object from the given YAML string.
- void save_to_stream (std::ostream *ptr) const

  Invokes save() and directs the resulting YAML text to the given stream.
- ErrorStack load_from_file (const fs::path &path, bool ignore_missing=false)

  Load the content of this object from the specified YAML file.
- ErrorStack save_to_file (const fs::path &path) const

  Atomically and durably writes out this object to the specified YAML file.
- ErrorStack load_from_command_options (int argc, char *argv[ ])

  Load content from command line options.

Static Public Member Functions

- static ErrorStack **insert_comment** (YAML::Emitter *out, const std::string &comment)
- static ErrorStack **append_comment** (YAML::Emitter *parent, const std::string &comment)
- template<typename T >
  static ErrorStack add_element (YAML::Emitter *out, const std::string &tag, const std::string &comment, T value, bool seq=false)
- template<typename T >
  static ErrorStack add_element (YAML::Emitter *out, const std::string &tag, const std::string &comment, const std::vector< T > &value, bool seq=true)
- template<typename ENUM >
  static ErrorStack add_enum_element (YAML::Emitter *out, const std::string &tag, const std::string &comment, ENUM value)
- static ErrorStack add_child_element (YAML::Node *parent, const std::string &tag, const std::string &comment, const Externalizable &child)
- template<typename T >
  static ErrorStack get_element (YAML::Node *parent, const std::string &tag, T *out, bool ignore_↩
  missing=false, bool optional=false, T value=0)
- static ErrorStack get_element (YAML::Node *parent, const std::string &tag, std::string *out, bool ignore_↩
  missing=false, bool optional=false, const char *value="")
- template<typename ENUM >
  static ErrorStack get_enum_element (YAML::Node *parent, const std::string &tag, ENUM *out, bool ignore↩
  _missing=false, bool optional=false, ENUM default_value=static_cast< ENUM >(0))
- template<typename SIZE >
  static ErrorStack get_size_element (YAML::Node *parent, const std::string &tag, SIZE *out, bool ignore_↩
  missing=false, bool optional=false, SIZE default_value=static_cast< SIZE >(0))
- template<typename T >
  static ErrorStack get_element (YAML::Node *parent, const std::string &tag, std::vector< T > *out, bool ignore_missing=false, bool optional=false)
- static ErrorStack get_child_element (YAML::Node *parent, const std::string &tag, Externalizable *child, bool ignore_missing=false, bool optional=false)
- template<typename T >
  static ErrorStack **add_command_option** (CommandOptionList *cmdlist, const std::string &tag, T *out, std↩
  ::string desc)

8.9.1 Detailed Description

Derived classes must implement load() and save().

8.9.2 Member Function Documentation

8.9.2.1 static ErrorStack alps::Externalizable::add_child_element (YAML::Node * *parent*, const std::string & *tag*, const std::string & *comment*, const Externalizable & *child*) [static]

child Externalizable version

8.9.2.2 template<typename T > ErrorStack alps::Externalizable::add_element (YAML::Emitter * *out*, const std::string & *tag*, const std::string & *comment*, T *value*, bool *seq* = false) [static]

Only declaration in header. Explicitly instantiated in cpp for each type this func handles.

8.9.2.3 template<typename T > ErrorStack alps::Externalizable::add_element (YAML::Emitter * *out*, const std::string & *tag*, const std::string & *comment*, const std::vector< T > & *value*, bool *seq* = true) [static]

vector version

Only declaration in header. Explicitly instantiated in cpp for each type this func handles.

8.9.2.4 template<typename ENUM > static ErrorStack alps::Externalizable::add_enum_element (YAML::Emitter * *out*, const std::string & *tag*, const std::string & *comment*, ENUM *value*) [inline], [static]

enum version

8.9.2.5 virtual void alps::Externalizable::assign (const alps::Externalizable * *other*) [pure virtual]

Parameters

in	*other*	assigned value. It must be dynamic-castable to the assignee class.

8.9.2.6 ErrorStack alps::Externalizable::get_child_element (YAML::Node * *parent*, const std::string & *tag*, Externalizable * *child*, bool *ignore_missing* = false, bool *optional* = false) [static]

child Externalizable version

8.9.2.7 template<typename T > ErrorStack alps::Externalizable::get_element (YAML::Node * *parent*, const std::string & *tag*, T * *out*, bool *ignore_missing* = false, bool *optional* = false, T *value* = 0) [static]

Only declaration in header. Explicitly instantiated in cpp for each type this func handles.

8.9.2.8 ErrorStack alps::Externalizable::get_element (YAML::Node * *parent*, const std::string & *tag*, std::string * *out*, bool *ignore_missing* = false, bool *optional* = false, const char * *value* = " ") [static]

string type is bit special.

8.9.2.9 template<typename T > ErrorStack alps::Externalizable::get_element (YAML::Node * *parent*, const std::string & *tag*, std::vector< T > * *out*, bool *ignore_missing* = false, bool *optional* = false) [static]

vector version. Only declaration in header. Explicitly instantiated in cpp for each type this func handles.

8.9.2.10 template<typename ENUM > static ErrorStack alps::Externalizable::get_enum_element (YAML::Node * *parent*, const std::string & *tag*, ENUM * *out*, bool *ignore_missing* = false, bool *optional* = false, ENUM *default_value* = static_cast<ENUM>(0)) [inline], [static]

enum version

8.9.2.11 template<typename SIZE > static ErrorStack alps::Externalizable::get_size_element (YAML::Node * *parent*, const std::string & *tag*, SIZE * *out*, bool *ignore_missing* = false, bool *optional* = false, SIZE *default_value* = static_cast<SIZE>(0)) [inline], [static]

size version

8.9.2.12 virtual const char* alps::Externalizable::get_tag_name ( ) const [pure virtual]

We might want to give a different name for same externalizable objects, so this is used only when it is the root element of yaml.

8.9.2.13 virtual ErrorStack alps::Externalizable::load (YAML::Node * *node*, bool *ignore_missing* = false) [pure virtual]

Parameters

in	*node*	the YAML node that represents this object

Expect errors due to missing-elements, out-of-range values, etc.

8.9.2.14 ErrorStack alps::Externalizable::load_from_command_options (int *argc*, char * *argv[]*)

Parameters

in	*argc*	number of strings that make up the command line.
in	*argv*	arguments passed to a program through the command line.

Expect errors due to missing-elements, out-of-range values, etc.

8.9.2.15 ErrorStack alps::Externalizable::load_from_file (const fs::path & *path,* bool *ignore_missing* = `false`)

Parameters

in	*path*	path of the YAML file.
in	*ignore_missing*	whether to ignore missing options.

Expect errors due to missing-elements, out-of-range values, etc.

8.9.2.16 ErrorStack alps::Externalizable::load_from_string (const std::string & *yaml,* bool *ignore_missing* = `false`)

Parameters

in	*yaml*	YAML string.
in	*ignore_missing*	whether to ignore missing options.

Expect errors due to missing-elements, out-of-range values, etc.

8.9.2.17 virtual ErrorStack alps::Externalizable::save (YAML::Emitter * *out*) const [pure virtual]

Parameters

in	*node*	the YAML node that represents this object

Expect only out-of-memory error. We receive the YAML node this object will represent, so this method does not determine the YAML node name of itself. The parent object determines children's tag names because one parent object might have multiple child objects of the same type with different YAML element name.

8.9.2.18 ErrorStack alps::Externalizable::save_to_file (const fs::path & *path*) const

Parameters

in	*path*	path of the YAML file.

If the file exists, this method atomically overwrites it via POSIX's atomic rename semantics. If the parent folder doesn't exist, this method automatically creates the folder. Expect errors due to file-permission (and other file I/O issue), out-of-memory, etc.

8.9.2.19 void alps::Externalizable::save_to_stream (std::ostream * *ptr*) const

Parameters

in	*ptr*	ostream to write to.

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

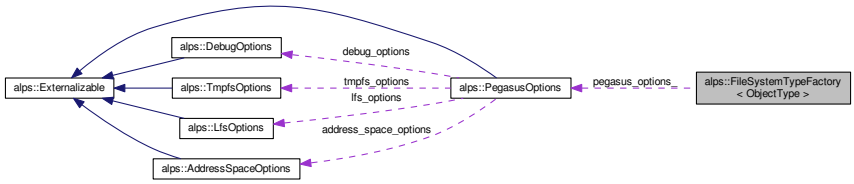
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/common/externalizable.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/common/externalizable.cc

8.10 alps::FileSystemTypeFactory< ObjectType > Class Template Reference

A factory class for constructing objects based on the type of the underlying file system.

`#include <fstype_factory.hh>`

Collaboration diagram for alps::FileSystemTypeFactory< ObjectType >:



Public Types

- typedef ObjectType *(* **ConstructCallback**) (const boost::filesystem::path &pathname, const PegasusOptions &pegasus_options)

Public Member Functions

- **FileSystemTypeFactory** (const PegasusOptions &pegasus_options)
- ErrorCode **construct** (const boost::filesystem::path &pathname, ObjectType **object)
- ErrorCode **register_fstype** (const std::string &fstype, ConstructCallback)

Protected Attributes

- ConstructCallbackMap **callbacks_**
- PegasusOptions **pegasus_options_**

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/fstype_factory.hh

8.11 alps::FRDNode Class Reference

Public Member Functions

- **FRDNode** (unsigned int node_id)
- **FRDNode** (unsigned int rack, unsigned int encl, unsigned int node)
- unsigned int **id** () const
- bool **operator==** (const FRDNode &other)
- bool **operator!=** (const FRDNode &other)
- unsigned int **operator-** (const FRDNode &other)

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/lfs↩ _topology.hh

8.12 alps::BacktraceContext::GlibcBacktraceInfo Struct Reference

Public Member Functions

- void **parse_symbol** ()

Public Attributes

- void * **address_**
- std::string **symbol_**
- std::string **function_**
- std::string **binary_path_**
- std::string **function_offset_**

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/common/rich↩ _backtrace.cc

8.13 alps::Hex Struct Reference

Convenient way of writing hex integers to stream.

`#include <assorted_func.hh>`

Public Member Functions

- template<typename T >
  Hex (T val, int fix_digits=-1)

Public Attributes

- uint64_t **val_**
- int **fix_digits_**

Friends

- std::ostream & **operator**<< (std::ostream &o, const Hex &v)

8.13.1 Detailed Description

Use it as follows.

```
std::cout << Hex(1234) << ...
// same output as:
// std::cout << "0x" << std::hex << std::uppercase << 1234 << std::nouppercase << std::dec << ...
```

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/common/assorted_func.hh

8.14 alps::HexString Struct Reference

Equivalent to std::hex in case the stream doesn't support it.

`#include <assorted_func.hh>`

Public Member Functions

- **HexString** (const std::string &str, uint32_t max_bytes=64U)

Public Attributes

- std::string **str_**
- uint32_t **max_bytes_**

Friends

- std::ostream & **operator**<< (std::ostream &o, const HexString &v)

8.14.1 Detailed Description

Use it as follows.

```
std::cout << Hex("aabc") << ...
// will output "0x61616263".
```

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/common/assorted_func.hh

8.15 alps::InvertedTable Class Reference

Public Types

- typedef boost::icl::interval_map< uintptr_t, VmArea * >::iterator **iterator**
- typedef boost::icl::interval_map< uintptr_t, VmArea * >::const_iterator **const_iterator**

Public Member Functions

- void **insert_vmarea** (VmArea *vma)
- void **remove_vmarea** (VmArea *vma)
- VmArea * **find_vmarea** (uintptr_t addr)

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

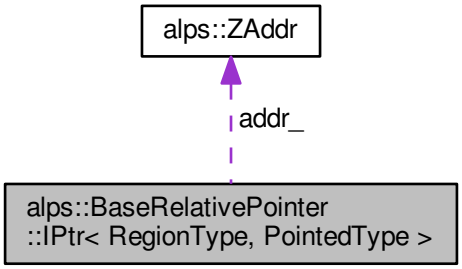
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/invtbl.hh

8.16 alps::BaseRelativePointer::IPtr< RegionType, PointedType > Class Template Reference

Intermediate representation of a relocatable pointer.

`#include <pointer.hh>`

Collaboration diagram for alps::BaseRelativePointer::IPtr< RegionType, PointedType >:



Public Member Functions

- **IPtr** (PointedType ∗from)
- **IPtr** (RegionType ∗pregion, LinearAddr offset)

Public Attributes

- RegionType ∗ **region_**
- ZAddr **addr_**

8.16.1 Detailed Description

template<typename RegionType, typename PointedType>
class alps::BaseRelativePointer::IPtr< RegionType, PointedType >

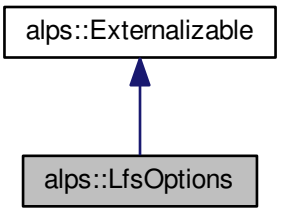
Not a full-fledged smart pointer but rather an intermediate representation that simplifies cross-assignment between transient and persistent pointers.

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

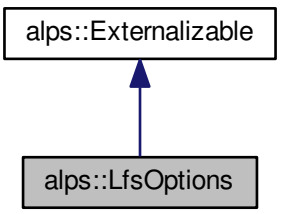
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/pointer.↩ hh

8.17 alps::LfsOptions Struct Reference

Inheritance diagram for alps::LfsOptions:



Collaboration diagram for alps::LfsOptions:



Public Member Functions

- LfsOptions ()

Public Attributes

- unsigned int **kDefaultNode** = 1
- unsigned int **kDefaultNodeCount** = 1
- size_t **kDefaultBookSizeBytes** = 8∗1024LLU∗1024LLU∗1024LLU
- unsigned int node

  the current node I am running on
- unsigned int node_count

  total number of nodes
- size_t book_size_bytes

  book size

Additional Inherited Members

8.17.1 Constructor & Destructor Documentation

8.17.1.1 alps::LfsOptions::LfsOptions ( ) [inline]

Constructs option values with default values

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

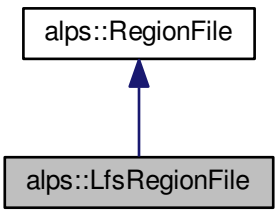
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/lfs↩
  _options.hh

8.18 alps::LfsRegionFile Class Reference

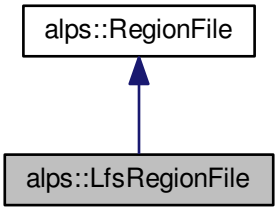
Represents a region file backed by the Librarian FS.

`#include <lfs_region_file.hh>`

Inheritance diagram for alps::LfsRegionFile:



Collaboration diagram for alps::LfsRegionFile:



Public Types

- enum **InterleavePolicy** { **kPreciseAllocate** = 0, **kRoundRobin** }

Public Member Functions

- **LfsRegionFile** (const boost::filesystem::path &pathname, const PegasusOptions &pegasus_options)
- ErrorCode **create** (mode_t mode)
- ErrorCode **open** (int flags, mode_t mode)
- ErrorCode **open** (int flags)
- ErrorCode **unlink** ()
- ErrorCode **close** ()
- ErrorCode **truncate** (loff_t length)
- ErrorCode **size** (loff_t ∗length)
- ErrorCode **map** (void ∗addr_hint, size_t length, int prot, int flags, loff_t offset, void ∗∗mapped_addr)
- ErrorCode **unmap** (void ∗addr, size_t length)
- ErrorCode **getxattr** (const char ∗name, void ∗value, size_t size)
- ErrorCode **setxattr** (const char ∗name, const void ∗value, size_t size, int flags)
- size_t **booksize** ()

Static Public Member Functions

- static RegionFile ∗ **construct** (const boost::filesystem::path &pathname, const PegasusOptions &pegasus↩ _options)

Static Public Attributes

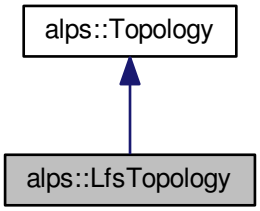
- static InterleavePolicy **interleave_policy_**

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

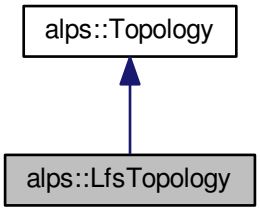
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/lfs↩ _region_file.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/lfs↩ _region_file.cc

8.19 alps::LfsTopology Class Reference

Inheritance diagram for alps::LfsTopology:



Collaboration diagram for alps::LfsTopology:



Public Member Functions

- **LfsTopology** (const boost::filesystem::path &path, const PegasusOptions &pegasus_options)
- InterleaveGroup max_interleave_group ()

  Returns the highest node number available in the system.
- InterleaveGroup nearest_ig ()

  Returns the nearest interleave group to the node the calling process is running on.

Static Public Member Functions

- static Topology * **construct** (const boost::filesystem::path &pathname, const PegasusOptions &pegasus_↩ options)

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/lfs↩ _topology.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/lfs↩ _topology.cc

8.20 alps::BacktraceContext::LibBacktraceInfo Struct Reference

Public Attributes

- uintptr_t **address_**
- std::string **srcfile_**
- int **srclineno_**
- std::string **function_**

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

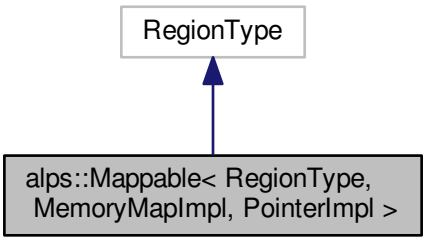
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/common/rich↩ _backtrace.cc

8.21 alps::Mappable< RegionType, MemoryMapImpl, PointerImpl > Class Template Reference

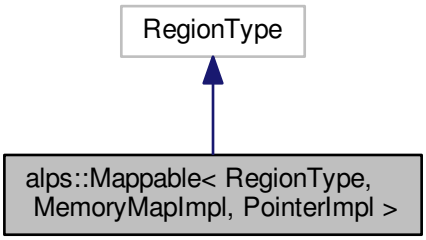
A template mixin class for defining mappable region types.

`#include <mappable.hh>`

Inheritance diagram for alps::Mappable< RegionType, MemoryMapImpl, PointerImpl >:



Collaboration diagram for alps::Mappable< RegionType, MemoryMapImpl, PointerImpl >:



Public Types

- template<class T >
  using **TPtr** = typename PointerImpl::template TPtr< Mappable, T >
- template<class T >
  using **PPtr** = typename PointerImpl::template PPtr< Mappable, T >
- template<class T >
  using **ZPtr** = typename PointerImpl::template ZPtr< Mappable, T >

Public Member Functions

- **Mappable** (AddressSpace *address_space, RegionFile *file)
- ErrorCode **map** ()
- ErrorCode **unmap** ()
- template<class T >
  TPtr< T > **base** (LinearAddr offset)
- uintptr_t **trans** (LinearAddr offset)

Friends

- class **AddressSpace**

8.21.1 Detailed Description

template<class RegionType, class MemoryMapImpl, class PointerImpl>
class alps::Mappable< RegionType, MemoryMapImpl, PointerImpl >

The MemoryMapImpl class defines the mapping policy that implements mapping of the underlying region file(s) onto the logical address space.

The PointerImpl class defines smart pointer types for naming (addressing) and referencing locations within the region.

There is no notion of a root pointer baked into the API. However, users may agree on a convention where offset 0x0 represents a root, and instantiate a smart pointer that points to offset 0x0 and use that as a root pointer.

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/mappable.↩
  hh

8.22 alps::MemoryManager Class Reference

The memory manager provides a mechanism for directly mapping persistent region files to virtual memory regions.

```
#include <mm.hh>
```

Public Member Functions

- ErrorCode **map** (Region *region, off_t offset, size_t length, void *addr_hint, int prot, int flags, VmArea **vmarea)
- ErrorCode **unmap** (Region *region, void *addr, size_t length)
- ErrorCode **rtrans** (void *addr, Region **pregion, LinearAddr *offset)
- ErrorCode **rtrans** (uintptr_t addr, Region **pregion, LinearAddr *offset)

8.22.1 Detailed Description

The manager is intended for use by a per-region memory mapper (MemoryMap) that implements mapping policy.

Todo Provide an API for picking address hints:

- best effort mapping: find the largest hole in the address space where we can map regions this can guide segment map to pick the right segment size
- guide underlying OS by picking an address_hint based on our past knowledge about persistent regions to minimize address space fragmentation similar to the Mnemosyne runtime (this could be implemented in the map method)

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

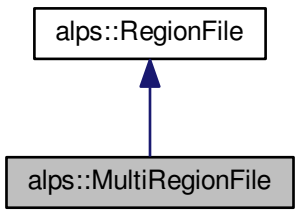
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/mm.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/mm.cc

8.23 alps::MultiRegionFile Class Reference

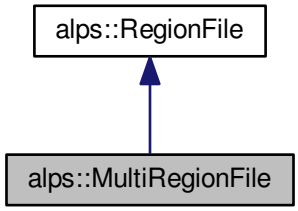
Represents a pseudo region file comprising multiple region files.

`#include <multi_region_file.hh>`

Inheritance diagram for alps::MultiRegionFile:



Collaboration diagram for alps::MultiRegionFile:



Public Member Functions

- **MultiRegionFile** (const std::vector< RegionFile * > region_files)
- ErrorCode create (mode_t mode)

  Operation not supported for this region file type.
- ErrorCode **open** (int flags, mode_t mode)
- ErrorCode **open** (int flags)
- ErrorCode unlink ()

  Operation not supported for this region file type.
- ErrorCode **close** ()
- ErrorCode truncate (loff_t length)

  Operation not supported for this region file type.
- ErrorCode **size** (loff_t *length)
- ErrorCode **map** (void *addr_hint, size_t length, int prot, int flags, loff_t offset, void **mapped_addr)
- ErrorCode **unmap** (void *addr, size_t length)
- ErrorCode getxattr (const char *name, void *value, size_t size)

  Operation not supported for this region file type.
- ErrorCode setxattr (const char *name, const void *value, size_t size, int flags)

  Operation not supported for this region file type.
- size_t **booksize** ()
- ErrorCode **set_interleave_group** (loff_t offset, loff_t length, const std::vector< InterleaveGroup > &vig)
- ErrorCode **interleave_group** (loff_t offset, loff_t length, std::vector< InterleaveGroup > *vig)

8.23.1 Detailed Description

We provide this pseudo region file as a helper class for glueing and mapping multiple existing region files as a single contiguous file. This class does not support operations that require modifying file-system metadata of underlying region files, including create, truncate, unlink, set/get attributes.

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/multi↩_region_file.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/multi↩_region_file.cc

8.24 alps::PAddr Struct Reference

Public Member Functions

- **PAddr** (LinearAddr linear_addr)
- bool **operator==** (const PAddr &other) const
- bool **operator!=** (const PAddr &other) const
- void **stream_to** (std::ostream &os) const

Public Attributes

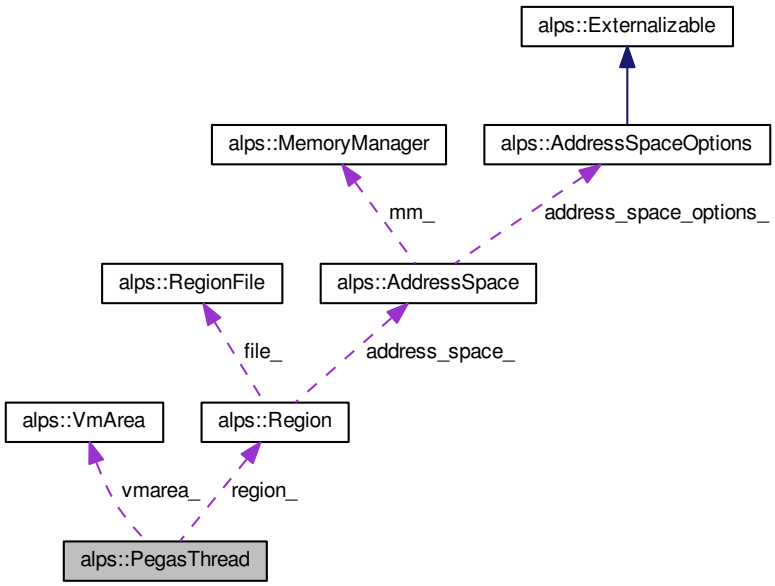
- LinearAddr **linear_addr_**

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/addr.hh

8.25 alps::PegasThread Class Reference

Collaboration diagram for alps::PegasThread:



Public Member Functions

- void **set_active_pregion** (Region *region)

Public Attributes

- Region * **region_**
- uint64_t **vmarea_version_**
- VmArea * **vmarea_**

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

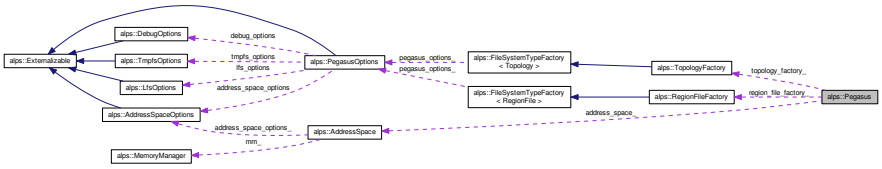
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/pegasthread.hh

8.26 alps::Pegasus Class Reference

Pegasus environment.

`#include <pegasus.hh>`

Collaboration diagram for alps::Pegasus:



Static Public Member Functions

- static ErrorStack load_options (const char *config_file, bool use_system_wide_conf, bool use_environ, PegasusOptions *pegasus_options)

  Loads configuration options from configuration file into object pointed by pegasus_options.
- static ErrorStack init (const char *config_file, bool use_system_wide_conf, bool use_environ)

  Initialize Pegasus singleton class by loading configuration options from a file.
- static ErrorStack init (const PegasusOptions &pegasus_options)

  Initialize Pegasus singleton class by loading configuration options from options object pegasus_options.
- static ErrorCode **create_region_file** (const char *pathname, mode_t mode, RegionFile **region_file)
- static ErrorCode **open_region_file** (const char *pathname, int flags, mode_t mode, RegionFile **region_file)
- static ErrorCode **open_region_file** (const char **pathnames, int npathnames, int flags, mode_t mode, RegionFile **region_file)
- static ErrorCode **open_region_file** (const char *pathname, int flags, RegionFile **region_file)
- static ErrorCode **open_region_file** (const char **pathnames, int npathnames, int flags, RegionFile **region_file)
- static AddressSpace * **address_space** ()
- static RegionFileFactory * **region_file_factory** ()
- static TopologyFactory * **topology_factory** ()

Static Protected Attributes

- static AddressSpace * **address_space_**
- static RegionFileFactory * **region_file_factory_**
- static TopologyFactory * **topology_factory_**

8.26.1 Detailed Description

A static singleton class that encapsulates a Pegasus environment

8.26.2 Member Function Documentation

8.26.2.1 ErrorStack alps::Pegasus::init (const char * *config_file*, bool *use_system_wide_conf*, bool *use_environ*) [static]

A shorthand for load_options() followed by init(PegasusOptions*).

Load configurations options in the order described under load_options.

8.26.2.2 ErrorStack alps::Pegasus::load_options (const char * *config_file*, bool *use_system_wide_conf*, bool *use_environ*, PegasusOptions * *pegasus_options*) [static]

Caller is responsible to pass a valid PegasusOptions object

Load configuration options in the following order:

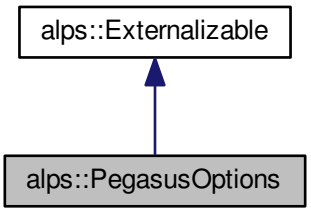
- If *use_system_wide_conf* is then load configuration options from the system wide configuration file (/etc/default/alps.yml), and then
- If path *use_environ* is set then load configuration options from the file declared by environment variable ALPS_CONF, and then
- Load configuration options from *config_file*.

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

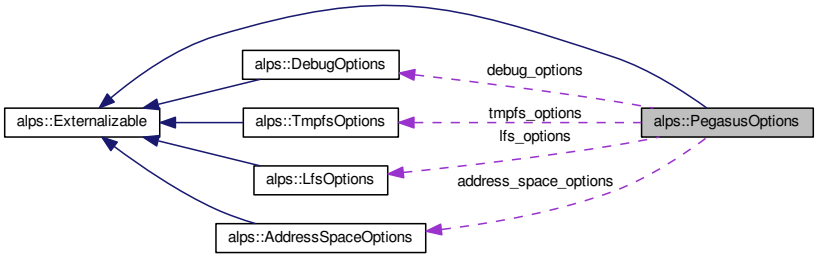
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/pegasus.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/pegasus.cc

8.27 alps::PegasusOptions Struct Reference

Inheritance diagram for alps::PegasusOptions:



Collaboration diagram for alps::PegasusOptions:



Public Member Functions

- **EXTERNALIZABLE** (PegasusOptions)

Public Attributes

- AddressSpaceOptions **address_space_options**
- DebugOptions **debug_options**
- LfsOptions **lfs_options**
- TmpfsOptions **tmpfs_options**

Additional Inherited Members

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

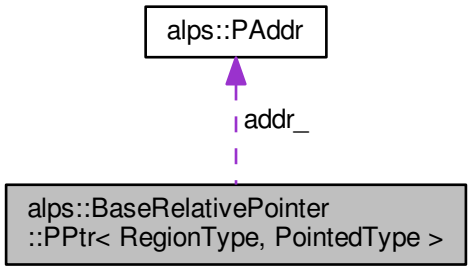
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/pegasus↩ _options.hh

8.28 alps::BaseRelativePointer::PPtr< RegionType, PointedType > Class Template Reference

Represents a linear persistent pointer.

`#include <pointer.hh>`

Collaboration diagram for alps::BaseRelativePointer::PPtr< RegionType, PointedType >:



Public Types

- typedef PointedType * **pointer**
- typedef boost::add_reference< PointedType >::type **reference**

Public Member Functions

- **PPtr** (LinearAddr offset)
- **PPtr** (const PPtr< RegionType, void > &other)
- **PPtr** (IPtr< RegionType, PointedType > &other)
- **operator IPtr< RegionType, PointedType >** ()
- PPtr & **operator=** (PointedType *from)
- PPtr & **operator=** (const IPtr< RegionType, PointedType > &other)
- PPtr & **operator=** (const TPtr< RegionType, PointedType > &tptr)
- pointer **get** () const
- pointer **operator->** () const
- reference **operator*** () const
- bool **operator==** (const IPtr< RegionType, PointedType > &other_iptr) const
- bool **operator!=** (const IPtr< RegionType, PointedType > &other_iptr) const
- reference **operator[ ]** (std::ptrdiff_t idx) const
- TPtr< RegionType, PointedType > **operator+** (std::ptrdiff_t offset) const
- void **stream_to** (std::ostream &os) const

Public Attributes

- PAddr **addr_**

8.28.1 Detailed Description

template<typename RegionType, typename PointedType>
class alps::BaseRelativePointer::PPtr< RegionType, PointedType >

Represents a smart persistent pointer that points to a persistent logical linear address (i.e., the pointer stores a persistent logical linear address). The pointer is valid for exchanging and sharing a reference to memory location among multiple processes mapping a region. It works with both fixed virtual address mappings and relocatable regions. It does not support inter-region pointers.

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/pointer.↩ hh

8.29 alps::ProcessMap Class Reference

Public Member Functions

- **ProcessMap** (pid_t pid)
- std::pair< size_t, size_t > **range** (const std::string name)

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

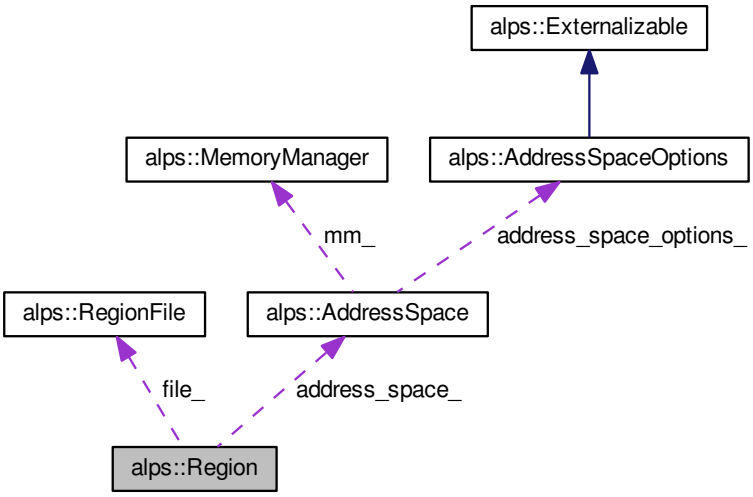
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/common/os.↩ hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/common/os.cc

8.30 alps::Region Class Reference

Persistent memory region.

`#include <region.hh>`

Collaboration diagram for alps::Region:



Public Member Functions

- **Region** (AddressSpace *address_space, RegionFile *region_file)
- RegionFile * file ()

  Returns the region file backing this region.
- loff_t **length** ()
- RegionId id ()

  Returns a global identifier identifying the region.
- AddressSpace * address_space ()

  Returns the Address Space object the region is mapped to.
- ErrorCode set_interleave_group (loff_t offset, loff_t length, const std::vector< InterleaveGroup > &vig)

  Sets interleave group hint for pages mapped in the range [offset, offset + length).
- ErrorCode interleave_group (loff_t offset, loff_t length, std::vector< InterleaveGroup > *vig)

  Returns assigned interleave group for pages mapped in the range [offset, offset + length).

Protected Attributes

- RegionId **region_id_**
- AddressSpace * **address_space_**
- loff_t length_

  cached value of the backing file's length
- RegionFile * file_

  the file backing the persistent memory region

8.30.1 Detailed Description

A base class that represents a persistent memory region in the logical address space (alps::AddressSpace).

A region is instantiated by mapping (and optionally binding) a region file (or a set of region files) to an AddressSpace object.

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

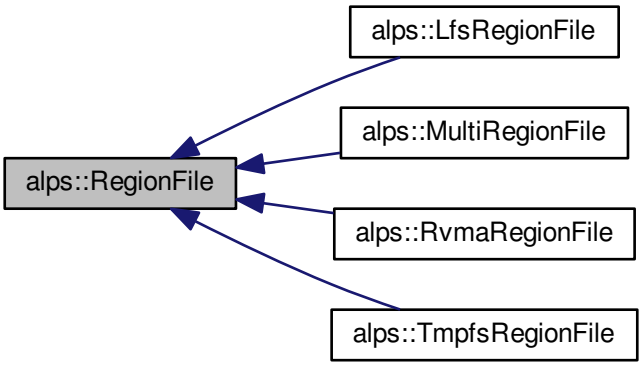
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/region.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/region.cc

8.31 alps::RegionFile Class Reference

Persistent region file.

`#include <region_file.hh>`

Inheritance diagram for alps::RegionFile:



Public Member Functions

- virtual ErrorCode **unlink** ()=0
- virtual ErrorCode **close** ()=0
- virtual ErrorCode **truncate** (loff_t length)=0
- virtual ErrorCode **size** (loff_t *length)=0
- virtual ErrorCode **map** (void *addr_hint, size_t length, int prot, int flags, loff_t offset, void **mapped_addr)=0
- virtual ErrorCode **unmap** (void *addr, size_t length)=0
- virtual ErrorCode **getxattr** (const char *name, void *value, size_t size)=0
- virtual ErrorCode **setxattr** (const char *name, const void *value, size_t size, int flags)=0
- virtual size_t **booksize** ()=0
- virtual ErrorCode **create** (mode_t mode)=0
- virtual ErrorCode **open** (int flags, mode_t mode)=0
- virtual ErrorCode **open** (int flags)=0
- virtual ErrorCode **set_interleave_group** (loff_t offset, loff_t length, const std::vector< InterleaveGroup > &vig)
- virtual ErrorCode **interleave_group** (loff_t offset, loff_t length, std::vector< InterleaveGroup > *vig)

Friends

- class **RegionFileFactory**

8.31.1 Detailed Description

This base class implements a common interface to region files backed by different types of memory file systems (e.g., TMPFS, LFS, EXT4+DAX). The API method names are self-described; they do what you expect them to do.

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

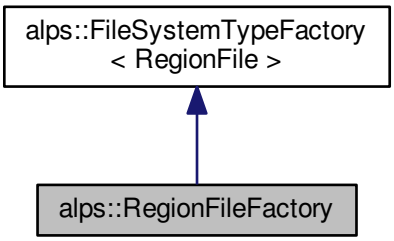
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/region_file.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/region_file.cc

8.32 alps::RegionFileFactory Class Reference

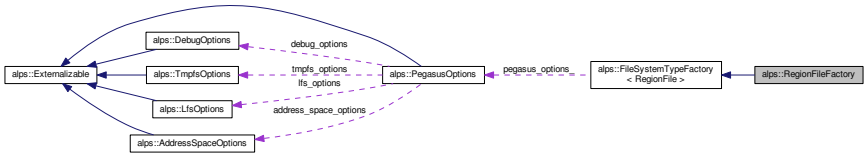
Region file factory.

`#include <region_file_factory.hh>`

Inheritance diagram for alps::RegionFileFactory:



Collaboration diagram for alps::RegionFileFactory:



Public Member Functions

- **RegionFileFactory** (const PegasusOptions &pegasus_options)
- ErrorCode **create** (const boost::filesystem::path &pathname, mode_t mode, RegionFile **region_file)
- ErrorCode **open** (const boost::filesystem::path &pathname, int flags, mode_t mode, RegionFile **region_file)
- ErrorCode **open** (const std::vector< boost::filesystem::path > &pathnames, int flags, mode_t mode, RegionFile **region_file)
- ErrorCode **open** (const boost::filesystem::path &pathname, int flags, RegionFile **region_file)
- ErrorCode **open** (const std::vector< boost::filesystem::path > &pathnames, int flags, RegionFile **region↩ _file)

Additional Inherited Members

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

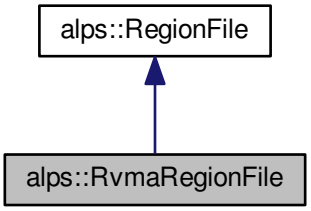
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/region↩ _file_factory.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/region↩ _file_factory.cc

8.33 alps::RvmaRegionFile Class Reference

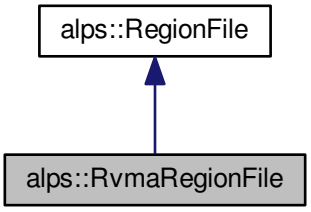
Represents a region file backed by an RVMA context.

`#include <rvma_region_file.hh>`

Inheritance diagram for alps::RvmaRegionFile:



Collaboration diagram for alps::RvmaRegionFile:



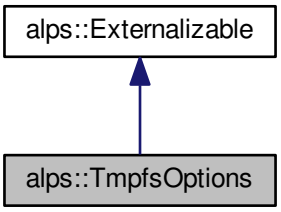
Additional Inherited Members

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

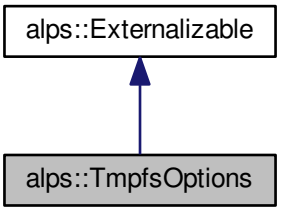
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/rvma↩_region_file.hh

8.34 alps::TmpfsOptions Struct Reference

Inheritance diagram for alps::TmpfsOptions:



Collaboration diagram for alps::TmpfsOptions:



Public Member Functions

- TmpfsOptions ()

Public Attributes

- std::size_t **kDefaultBookSizeBytes** = 8∗1024∗1024LLU
- size_t **book_size_bytes**

Additional Inherited Members

8.34.1 Constructor & Destructor Documentation

8.34.1.1 alps::TmpfsOptions::TmpfsOptions ( ) [inline]

Constructs option values with default values

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

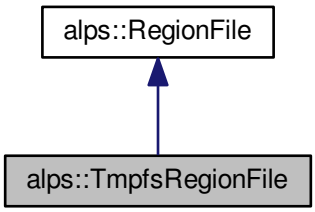
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/tmpfs↩ _options.hh

8.35 alps::TmpfsRegionFile Class Reference

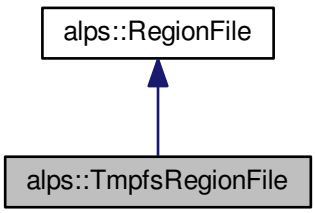
Represents a region file backed by TMPFS.

`#include <tmpfs_region_file.hh>`

Inheritance diagram for alps::TmpfsRegionFile:



Collaboration diagram for alps::TmpfsRegionFile:



Public Types

- enum **InterleavePolicy** { **kPreciseAllocate** = 0, **kRoundRobin** }

Public Member Functions

- **TmpfsRegionFile** (boost::filesystem::path pathname, boost::filesystem::path xpathname, const PegasusOptions &pegasus_options)
- ErrorCode **create** (mode_t mode)
- ErrorCode **open** (int flags, mode_t mode)
- ErrorCode **open** (int flags)
- ErrorCode **unlink** ()
- ErrorCode **close** ()

- ErrorCode **truncate** (loff_t length)
- ErrorCode **size** (loff_t *length)
- ErrorCode **map** (void *addr_hint, size_t length, int prot, int flags, loff_t offset, void **mapped_addr)
- ErrorCode **unmap** (void *addr, size_t length)
- ErrorCode **getxattr** (const char *name, void *value, size_t size)
- ErrorCode **setxattr** (const char *name, const void *value, size_t size, int flags)
- size_t **booksize** ()

Static Public Member Functions

- static RegionFile * **construct** (const boost::filesystem::path &pathname, const PegasusOptions &pegasus↩ _options)

Static Public Attributes

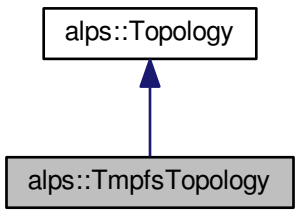
- static const char * **kLockFileName** = "/dev/shm/@@lockfile@@"
- static const char * **kXattrExtension** = ".xattr"
- static InterleavePolicy **interleave_policy_** = TmpfsRegionFile::kPreciseAllocate

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

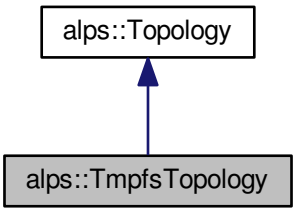
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/tmpfs↩ _region_file.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/tmpfs↩ _region_file.cc

8.36 alps::TmpfsTopology Class Reference

Inheritance diagram for alps::TmpfsTopology:



Collaboration diagram for alps::TmpfsTopology:



Public Member Functions

- **TmpfsTopology** (const boost::filesystem::path &path, const PegasusOptions &pegasus_options)
- InterleaveGroup max_interleave_group ()

  Returns the highest node number available in the system.
- InterleaveGroup nearest_ig ()

  Returns the nearest interleave group to the node the calling process is running on.
- int **run_on_node** (int n)

Static Public Member Functions

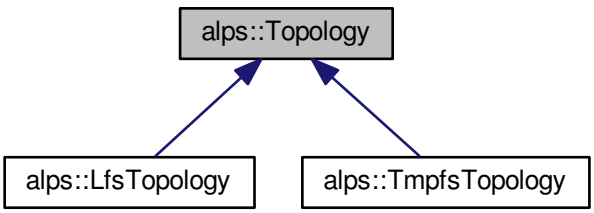
- static Topology * **construct** (const boost::filesystem::path &pathname, const PegasusOptions &pegasus_↩ options)

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/tmpfs↩ _topology.hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/tmpfs↩ _topology.cc

8.37 alps::Topology Class Reference

Inheritance diagram for alps::Topology:



Public Member Functions

- virtual InterleaveGroup max_interleave_group ()=0

  Returns the highest node number available in the system.
- virtual InterleaveGroup nearest_ig ()=0

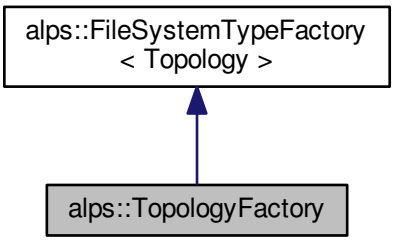
  Returns the nearest interleave group to the node the calling process is running on.

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

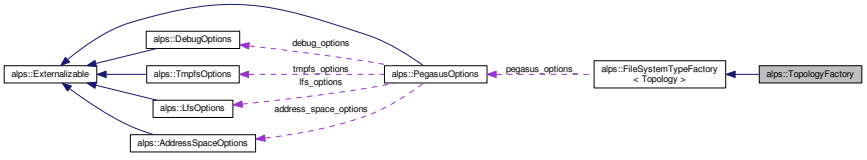
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/topology.↩ hh
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/topology.↩ cc

8.38 alps::TopologyFactory Class Reference

Inheritance diagram for alps::TopologyFactory:



Collaboration diagram for alps::TopologyFactory:



Public Member Functions

- **TopologyFactory** (const PegasusOptions &pegasus_options)

Additional Inherited Members

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

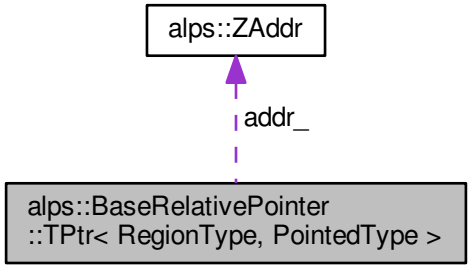
- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/topology↩
_factory.hh

8.39 alps::BaseRelativePointer::TPtr< RegionType, PointedType > Class Template Reference

Represents a transient pointer.

`#include <pointer.hh>`

Collaboration diagram for alps::BaseRelativePointer::TPtr< RegionType, PointedType >:



Public Types

- typedef PointedType * **pointer**
- typedef boost::add_reference< PointedType >::type **reference**

Public Member Functions

- **TPtr** (ZAddr zaddr)
- **TPtr** (RegionId region_id, LinearAddr offset)
- **TPtr** (RegionType *pregion, LinearAddr offset)
- **TPtr** (const TPtr< RegionType, void > &other)
- **TPtr** (const IPtr< RegionType, PointedType > &other)
- **TPtr** (PointedType *from)
- **operator IPtr< RegionType, PointedType >** () const
- TPtr & **operator=** (const IPtr< RegionType, PointedType > &other)
- TPtr & **operator=** (const typename RegionType::template TPtr< PointedType > &tptr)
- TPtr & **operator=** (PointedType *from)
- bool **operator==** (const ZAddr &other_zaddr) const

- bool **operator!=** (const ZAddr &other_zaddr) const
- **operator ZAddr** () const
- pointer **get** () const
- RegionId **region_id** () const
- LinearAddr **offset** () const
- RegionType * **region** ()
- pointer **operator->** () const
- reference **operator*** () const
- bool **operator<** (const TPtr rhs) const
- reference **operator[ ]** (std::ptrdiff_t idx) const
- void **inc_offset** (std::ptrdiff_t bytes)
- void **dec_offset** (std::ptrdiff_t bytes)
- TPtr **operator+** (std::ptrdiff_t diff) const
- std::ptrdiff_t **operator-** (const TPtr &other) const
- TPtr **operator-** (std::ptrdiff_t diff) const
- TPtr & **operator+=** (std::ptrdiff_t diff)
- TPtr & **operator-=** (std::ptrdiff_t diff)
- TPtr & **operator++** ()
- TPtr **operator++** (int)
- TPtr & **operator--** ()
- TPtr **operator--** (int)
- void **stream_to** (std::ostream &os) const

Public Attributes

- RegionType * **region_**
- ZAddr **addr_**

8.39.1 Detailed Description

template<typename RegionType, typename PointedType>
class alps::BaseRelativePointer::TPtr< RegionType, PointedType >

Represents a smart transient pointer that points to a persistent logical address (i.e., the pointer stores a persistent logical address). It may also be used as an intermediate pointer representation

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/pointer.↩ hh

8.40 alps::VmArea Class Reference

Public Member Functions

- **VmArea** (Region *region, LinearAddr offset, uintptr_t vm_start, uintptr_t vm_end)
- Region * **region** ()
- LinearAddr **offset** ()
- uintptr_t **vm_start** () const
- uintptr_t **vm_end** () const

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/src/pegasus/vmarea.↩ hh

8.41 alps::ZAddr Struct Reference

Public Member Functions

- **ZAddr** (RegionId region_id, LinearAddr linear_addr)
- bool **operator==** (const ZAddr &other) const
- bool **operator!=** (const ZAddr &other) const
- void **stream_to** (std::ostream &os) const

Public Attributes

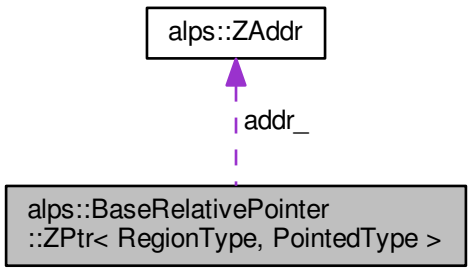
- RegionId **region_id_**
- LinearAddr **linear_addr_**

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/addr.hh

8.42 alps::BaseRelativePointer::ZPtr< RegionType, PointedType > Class Template Reference

Collaboration diagram for alps::BaseRelativePointer::ZPtr< RegionType, PointedType >:



Public Types

- typedef PointedType * **pointer**
- typedef boost::add_reference< PointedType >::type **reference**

Public Member Functions

- **ZPtr** (ZAddr zaddr)
- **ZPtr** (RegionId region_id, LinearAddr offset)
- **ZPtr** (const ZPtr< RegionType, void > &other)
- **ZPtr** (const typename RegionType::template TPtr< PointedType > &tptr)
- ZPtr & **operator=** (const typename RegionType::template TPtr< PointedType > &tptr)
- **operator TPtr< RegionType, PointedType >** ()

Public Attributes

- ZAddr **addr_**

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

- /home/yuan/Benchmarks/whisper/mnemosyne-gcc/usermode/library/pmalloc/include/alps/include/alps/pegasus/pointer.↩
  hh

Index