Alps

trunk

Generated by Doxygen 1.8.1.2

Mon Mar 16 2015 20:20:17

Contents

1 Class Index

1.1 Class Hierarchy

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

std::basic_ifstream< wchar_t >
std::basic_ios< char >
std::basic_ios< wchar_t >
std::basic_iostream< char >
std::basic_iostream< wchar_t >
std::basic_istream< char >
std::basic_istream< wchar_t >
std::basic_istringstream< char >
std::basic_istringstream< wchar_t >
std::basic_ofstream< char >
std::basic_ofstream< wchar_t >
std::basic_ostream< char >
std::basic_ostream< wchar_t >
std::basic_ostringstream< char >
std::basic_ostringstream< wchar_t >
std::basic_string< char >
std::basic_string< wchar_t >
std::basic_stringstream< char >
std::basic_stringstream< wchar_t >

2 Class Index

2.1 Class List

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

3 Class Documentation

3.1 ALPS_PS_STATS Struct Reference

3.1.1 Detailed Description

Definition at line 179 of file Alps.h.

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

- Alps.h

3.2 AlpsEncoded Class Reference

This data structure is to contain the packed form of an encodable knowledge.

`#include <AlpsEncoded.h>`

Public Member Functions

- void make_fit (const int addSize)

  Reallocate the size of encoded if necessary so that at least `addsize_` number of additional bytes will fit into the encoded.
- void clear ()

  Completely clear the encoded.
- template<class T >
  AlpsEncoded & writeRep (const T &value)

  Write a single object of type `T` in `repsentation_` .
- template<class T >
  AlpsEncoded & readRep (T &value)

  Read a single object of type `T` from `repsentation_` .
- template<class T >
  AlpsEncoded & writeRep (const T *const values, const int length)

  Write a C style array of objects of type `T` in `repsentation_`.
- template<class T >
  AlpsEncoded & readRep (T *&values, int &length, bool needAllocateMemory=true)

  *Read an array of objects of type `T` from `repsentation_`, where T **must** be a built-in type (ar at least something that can be copied with memcpy).*
- AlpsEncoded & writeRep (std::string &value)

  Read a `std::string` in `repsentation_` .
- AlpsEncoded & readRep (std::string &value)

  Read a `std::string` from `repsentation_` .
- template<class T >
  AlpsEncoded & writeRep (const std::vector< T > &vec)

  Write a `std::vector` into `repsentation_` .
- template<class T >
  AlpsEncoded & readRep (std::vector< T > &vec)

  Read a `std::vector` from `repsentation_` .

Constructors and destructor

- AlpsEncoded ()
  The default constructor creates a buffer of size 16 Kbytes with no message in it.
- AlpsEncoded (int t)
  Useful constructor.
- AlpsEncoded (int t, int s, char *&r)
  Useful constructor.
- ~AlpsEncoded ()
  Destructor.

Query methods

- int **type** () const
- int **size** () const
- const char * **representation** () const

3.2.1 Detailed Description

This data structure is to contain the packed form of an encodable knowledge.

It servers two purposes:

- used as a buffer when passing messages
- allow Alps to manipulate the user derived knowledge

Definition at line 25 of file AlpsEncoded.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 AlpsEncoded::AlpsEncoded ( ) [inline]

The default constructor creates a buffer of size 16 Kbytes with no message in it.

Definition at line 65 of file AlpsEncoded.h.

3.2.2.2 AlpsEncoded::AlpsEncoded (int *t*) [inline]

Useful constructor.

Definition at line 75 of file AlpsEncoded.h.

3.2.2.3 AlpsEncoded::AlpsEncoded (int *t*, int *s*, char *& *r*) [inline]

Useful constructor.

Take over ownership of r.

Definition at line 85 of file AlpsEncoded.h.

3.2.2.4 AlpsEncoded::~AlpsEncoded ( ) [inline]

Destructor.

Definition at line 95 of file AlpsEncoded.h.

3.2.3 Member Function Documentation

3.2.3.1 void AlpsEncoded::make_fit (const int *addSize*) [inline]

Reallocate the size of encoded if necessary so that at least `addsize_` number of additional bytes will fit into the encoded.

Definition at line 130 of file AlpsEncoded.h.

3.2.3.2 void AlpsEncoded::clear ( ) [inline]

Completely clear the encoded.

Delete and zero out `type_, size_, pos_`.

Definition at line 146 of file AlpsEncoded.h.

3.2.3.3 template<class T > AlpsEncoded& AlpsEncoded::writeRep (const T & *value*) [inline]

Write a single object of type `T` in `repsentation_` .

Copies `sizeof(T)` bytes from the address of the object.

Definition at line 163 of file AlpsEncoded.h.

3.2.3.4 template<class T > AlpsEncoded& AlpsEncoded::readRep (T & *value*) [inline]

Read a single object of type `T` from `repsentation_` .

Copies `sizeof(T)` bytes to the address of the object.

Definition at line 173 of file AlpsEncoded.h.

3.2.3.5 template<class T > AlpsEncoded& AlpsEncoded::writeRep (const T *const *values, const int *length*) [inline]

Write a C style array of objects of type `T` in `repsentation_`.

First write the length, then write the content of the array

Definition at line 189 of file AlpsEncoded.h.

3.2.3.6 template<class T > AlpsEncoded& AlpsEncoded::readRep (T *& *values*, int & *length*, bool *needAllocateMemory* = `true`) [inline]

Read an array of objects of type `T` from `repsentation_`, where T **must** be a built-in type (ar at least something that can be copied with memcpy).

If the third argument is true then memory is allocated for the array and the array pointer and the length of the array are returned in the arguments.

If the third argument is false then the arriving array's length is compared to `length` and an exception is thrown if they are not the same. Also, the array passed as the first argument will be filled with the arriving array.

Definition at line 216 of file AlpsEncoded.h.

3.2.3.7 AlpsEncoded& AlpsEncoded::writeRep (std::string & *value*) [inline]

Read a `std::string` in `repsentation_` .

Definition at line 281 of file AlpsEncoded.h.

3.2.3.8 AlpsEncoded& AlpsEncoded::readRep (std::string & *value*) [inline]

Read a `std::string` from `repsentation_` .

Definition at line 295 of file AlpsEncoded.h.

3.2.3.9 template<class T > AlpsEncoded& AlpsEncoded::writeRep (const std::vector< T > & *vec*) [inline]

Write a `std::vector` into `repsentation_` .

Definition at line 304 of file AlpsEncoded.h.

3.2.3.10 template<class T > AlpsEncoded& AlpsEncoded::readRep (std::vector< T > & *vec*) [inline]

Read a `std::vector` from `repsentation_` .

Definition at line 318 of file AlpsEncoded.h.

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

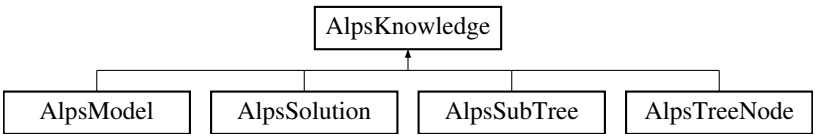
- AlpsEncoded.h

3.3 AlpsKnowledge Class Reference

The abstract base class of any user-defined class that Alps has to know about in order to encode/decode.

`#include <AlpsKnowledge.h>`

Inheritance diagram for AlpsKnowledge:



Public Member Functions

- virtual AlpsEncoded ∗ encode () const

  This method should encode the content of the object and return a pointer to the encoded form.
- virtual AlpsReturnStatus encode (AlpsEncoded ∗encoded)

  Pack into a encode object.
- virtual AlpsKnowledge ∗ decode (AlpsEncoded &encoded) const

  This method should decode and return a pointer to a brand new object, i.e., the method must create a new object on the heap from the decoded data instead of filling up the object for which the method was invoked.
- AlpsEncoded ∗ getEncoded () const

  Get/set encoded.

3.3.1 Detailed Description

The abstract base class of any user-defined class that Alps has to know about in order to encode/decode.

These classes must all be registered so that the proper decode method can be called.

Definition at line 51 of file AlpsKnowledge.h.

3.3.2 Member Function Documentation

3.3.2.1 virtual AlpsEncoded* AlpsKnowledge::encode ( ) const [virtual]

This method should encode the content of the object and return a pointer to the encoded form.

NOTE: This default implementation can not be used when the memory of data members is not continously allocated, for example, some data members are pointers, STL set, map, etc.

Reimplemented in AlpsSubTree.

3.3.2.2 virtual AlpsReturnStatus AlpsKnowledge::encode (AlpsEncoded * *encoded*) [inline], [virtual]

Pack into a encode object.

Definition at line 83 of file AlpsKnowledge.h.

3.3.2.3 virtual AlpsKnowledge* AlpsKnowledge::decode (AlpsEncoded & *encoded*) const [virtual]

This method should decode and return a pointer to a *brand new object*, i.e., the method must create a new object on the heap from the decoded data instead of filling up the object for which the method was invoked.

NOTE: This default implementation can not be used when the memory of data members is not continously allocated, for example, some data members are pointers, STL set, map, etc.

Reimplemented in AlpsSubTree.

3.3.2.4 AlpsEncoded* AlpsKnowledge::getEncoded ( ) const [inline]

Get/set encoded.

Definition at line 98 of file AlpsKnowledge.h.

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

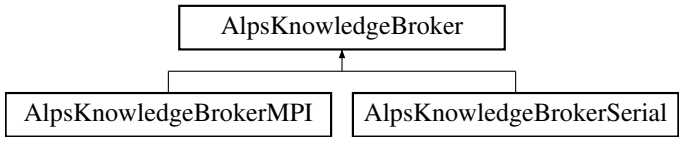
- AlpsKnowledge.h

3.4 AlpsKnowledgeBroker Class Reference

The base class of knowledge broker class.

`#include <AlpsKnowledgeBroker.h>`

Inheritance diagram for AlpsKnowledgeBroker:



Public Member Functions

- AlpsKnowledgeBroker ()

  Default constructor.
- virtual ~AlpsKnowledgeBroker ()

  Destructor.

- int getTreeDepth ()
  Get tree depth.
- void setPeakMemory (double size)
  Set peak memory usage.
- double getPeakMemory ()
  Get peak memory usage.
- virtual int getProcRank () const
  Qeury the global rank of process.
- virtual int getMasterRank () const
  Query the global rank of the Master.
- virtual AlpsProcessType getProcType () const
  Query the type (master, hub, or worker) of the process.

Funcitons related to register knowledge.

- void registerClass (int name, AlpsKnowledge *userKnowledge)
  Every user derived knowledge class must register.
- const AlpsKnowledge * decoderObject (int name)
  This method returns the pointer to an empty object of the registered class `name`.

Funcitons related to exploring subtree.

- virtual void initializeSearch (int argc, char *argv[], AlpsModel &model)=0
  Do some initialization for search.
- virtual void rootSearch (AlpsTreeNode *root)=0
  Explore the tree rooted as the given root.
- virtual void search (AlpsModel *model)
  Search best solution for a given model.

Get/set phase.

- AlpsPhase **getPhase** ()
- void **setPhase** (AlpsPhase ph)
- AlpsModel * **getModel** ()
- void **setModel** (AlpsModel *m)

Interface with the knowledge pools

- void setupKnowledgePools ()
  Set up knowledge pools for this broker.
- void addKnowledgePool (AlpsKnowledgeType kt, AlpsKnowledgePool *kp)
  Add a knowledge pool into the Knowledge pools.
- AlpsKnowledgePool * getKnowledgePool (AlpsKnowledgeType kt) const
  Retrieve a knowledge pool in the Knowledge base.
- virtual int getNumKnowledges (AlpsKnowledgeType kt) const
  Query the number of knowledge in the given type of a knowledge pool.
- virtual int getMaxNumKnowledges (AlpsKnowledgeType kt) const
  Query the max number of knowledge can be stored in a given type of knowledge pools.
- virtual void setMaxNumKnowledges (AlpsKnowledgeType kt, int num)
  Set the max number of knowledge can be stored in a given type o fknowledge pools.
- virtual bool hasKnowledge (AlpsKnowledgeType kt) const
  Query whether there are knowledges in the given type of knowledge pools.

- virtual std::pair
  < AlpsKnowledge *, double > getKnowledge (AlpsKnowledgeType kt) const
  Get a knowledge, but doesn't remove it from the pool.
- virtual void popKnowledge (AlpsKnowledgeType kt)
  Remove the a knowledge from the given type of knowledge pools.
- virtual std::pair
  < AlpsKnowledge *, double > getBestKnowledge (AlpsKnowledgeType kt) const
  Get the best knowledge in the given type of knowledge pools.
- virtual void getAllKnowledges (AlpsKnowledgeType kt, std::vector< std::pair< AlpsKnowledge *, double > > &kls) const
  Get all knowledges in the given type of knowledge pools.
- virtual void addKnowledge (AlpsKnowledgeType kt, AlpsKnowledge *kl, double value)
  Add a knowledge in the given type of knowledge pools.

Querty and set statistics

- int getNumNodesProcessed () const
  Query the number of node processed by this process.
- int getNumNodesBranched () const
  Query the number of node processed by this process.
- int getNumNodesDiscarded () const
  Query the number of node processed by this process.
- int getNumNodesPartial () const
  Query the number of node in the queue that are pregnant.
- int getNumNodesProcessedSystem () const
  Query the number of node processed by the system.
- virtual int updateNumNodesLeft ()
  Update the number of left nodes on this process.
- virtual AlpsTreeNode * getBestNode () const
  Query the best node in the subtree pool.
- AlpsExitStatus getSolStatus () const
  Query search termination status.
- void setExitStatus (AlpsExitStatus status)
  Set terminate status.
- AlpsTimer & timer ()
  Query timer.
- AlpsTimer & subTreeTimer ()
  Query subtree timer.
- AlpsTimer & tempTimer ()
  Query secondary timer.
- virtual void searchLog ()=0
  Search statistics log.

Query and set the approximate memory size of a tree node

- int **getNodeMemSize** ()
- void **setNodeMemSize** (int ms)

Query and set the approximate node processing time

- double **getNodeProcessingTime** ()
- void **setNodeProcessingTime** (double npTime)

Report the best result

- virtual double getIncumbentValue () const =0
  The process queries the objective value of the incumbent that it stores.
- virtual double getBestQuality () const =0
  The process (serial) / the master (parallel) queries the quality of the best solution that it knows.
- virtual double getBestEstimateQuality ()
  Get best estimalted quality in system.
- virtual int **getNumNodeLeftSystem** ()
- virtual void printBestSolution (char *outputFile=0) const =0
  The process (serial) / the master (parallel) outputs the best solution that it knows to a file or std::out.

Query and set node index

- AlpsNodeIndex_t nextNodeIndex ()
  Query the next index assigned to a newly created node, and then increment the nextIndex_ by 1.
- AlpsNodeIndex_t getNextNodeIndex () const
  Query the next index assigned to a newly created node.
- void setNextNodeIndex (AlpsNodeIndex_t s)
  Set nextIndex_.
- AlpsNodeIndex_t getMaxNodeIndex () const
  Queriy the upper bound of node indices.
- void setMaxNodeIndex (AlpsNodeIndex_t s)
  Set the upper bound of node indices.

Query and set comparision

- AlpsSearchStrategy
  < AlpsSubTree * > * **getSubTreeSelection** () const
- void **setSubTreeSelection** (AlpsSearchStrategy< AlpsSubTree * > *tc)
- AlpsSearchStrategy
  < AlpsTreeNode * > * **getNodeSelection** () const
- void **setNodeSelection** (AlpsSearchStrategy< AlpsTreeNode * > *nc)
- AlpsSearchStrategy
  < AlpsTreeNode * > * **getRampUpNodeSelection** () const
- void **setRampUpNodeSelection** (AlpsSearchStrategy< AlpsTreeNode * > *nc)

Message and log file handling

- void passInMessageHandler (CoinMessageHandler *handler)
  Pass in Message handler (not deleted at end).
- void newLanguage (CoinMessages::Language language)
  Set language.
- void **setLanguage** (CoinMessages::Language language)
- CoinMessageHandler * messageHandler () const
  Return handler.
- CoinMessages messages ()
  Return messages.
- CoinMessages * messagesPointer ()
  Return pointer to messages.
- int getMsgLevel ()
  Return msg level.
- int getHubMsgLevel ()
  Return msg level.
- int getMasterMsgLevel ()
  Return msg level.
- int getlogFileLevel ()

Return log file level.

- int getNumNodeLog () const

  Get times that node log has been printed.

- void setNumNodeLog (int num)

  Get times that node log has been printed.

Protected Attributes

- std::string instanceName_

  The instance name.

- AlpsModel * model_

  Pointer to model.

- AlpsPhase phase_

  Alps phase.

- int nodeMemSize_

  The approximate memory size (bytes) of a node with full description.

- double nodeProcessingTime_

  The approximately CPU time to process a node.

- int largeSize_

  The size of largest message buffer can be sent or received.

- bool userBalancePeriod_

  Has user input balance period.

- int numNodeLog_

  Times that node log is printed.

knowledge pools

- AlpsSubTreePool * subTreePool_

  A subtree pool holding a collection of subtrees.

- AlpsSolutionPool * solPool_

  A solution pool containing the solutions found.

- std::map< AlpsKnowledgeType, AlpsKnowledgePool * > * pools_

  The collection of pools managed by the knowledge broker.

Exploring subtree

- AlpsSubTree * workingSubTree_

  Point to the subtree that being explored.

- bool needWorkingSubTree_

  Indicate whether need a new subtree.

- AlpsNodeIndex_t nextIndex_

  The index to be assigned to a new search tree node.

- AlpsNodeIndex_t maxIndex_

  The maximum index can been assigned on this process.

Statistics

- AlpsTimer timer_

  Main timer.

- AlpsTimer subTreeTimer_

Subtree timer.

- AlpsTimer tempTimer_
  Secondary timer.
- int solNum_
  The number of solutions found.
- int nodeProcessedNum_
  The number of nodes that have been processed.
- int nodeBranchedNum_
  The number of nodes that have been branched.
- int nodeDiscardedNum_
  The number of nodes that have been discarded before processing.
- int nodePartialNum_
  The number of nodes that are pregnant.
- int systemNodeProcessed_
  To record how many nodes processed by the system (used in parallel code).
- int nodeLeftNum_
  The number of nodes left.
- int treeDepth_
  The depth of the tree.
- int bestSolNode_
  The number of nodes pocessed to find the solution.
- double peakMemory_
  Peak memory usage.
- AlpsExitStatus exitStatus_
  The status of search when terminated.

Search strategy

- AlpsSearchStrategy
  $<$ AlpsSubTree $*>*$ treeSelection_
  Tree selection criterion.
- AlpsSearchStrategy
  $<$ AlpsTreeNode $*>*$ nodeSelection_
  Node selection criterion.
- AlpsSearchStrategy
  $<$ AlpsTreeNode $*>*$ rampUpNodeSelection_
  Node selection criterion.

message handling

- CoinMessageHandler $*$ handler_
  Message handler.
- CoinMessages messages_
  Alps messages.
- int msgLevel_
  The leve of printing message to screen of the master and general message.
- int hubMsgLevel_
  The leve of printing message to screen of hubs.
- int workerMsgLevel_
  The leve of printing message to screen of workers.
- int logFileLevel_
  The degree of log file.
- std::string logfile_
  The log file.

3.4.1 Detailed Description

The base class of knowledge broker class.

Definition at line 48 of file AlpsKnowledgeBroker.h.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 AlpsKnowledgeBroker::AlpsKnowledgeBroker ( )

Default constructor.

3.4.2.2 virtual AlpsKnowledgeBroker::~AlpsKnowledgeBroker ( ) [virtual]

Destructor.

3.4.3 Member Function Documentation

3.4.3.1 void AlpsKnowledgeBroker::registerClass (int *name*, AlpsKnowledge * *userKnowledge*) [inline]

Every user derived knowledge class must register.

The register methods register the decode method of the class so that later on we can decode objects from buffers. Invoking this registration for class `foo` is a single line:

`foo().registerClass(name, userKnowledge).` NOTE: take over user knowledge's memory ownership, user doesn't need free memory.

Definition at line 230 of file AlpsKnowledgeBroker.h.

3.4.3.2 const AlpsKnowledge* AlpsKnowledgeBroker::decoderObject (int *name*) [inline]

This method returns the pointer to an empty object of the registered class `name`.

Then the `decode()` method of that object can be used to decode a new object of the same type from the buffer. This method will be invoked as follows to decode an object whose type is `name`:

```
obj = AlpsKnowledge::decoderObject(name)->decode(buf)
```

Definition at line 253 of file AlpsKnowledgeBroker.h.

3.4.3.3 virtual void AlpsKnowledgeBroker::initializeSearch (int *argc*, char * *argv[]*, AlpsModel & *model*) [pure virtual]

Do some initialization for search.

Implemented in AlpsKnowledgeBrokerMPI, and AlpsKnowledgeBrokerSerial.

3.4.3.4 virtual void AlpsKnowledgeBroker::rootSearch (AlpsTreeNode * *root*) [pure virtual]

Explore the tree rooted as the given root.

Implemented in AlpsKnowledgeBrokerMPI, and AlpsKnowledgeBrokerSerial.

3.4.3.5 virtual void AlpsKnowledgeBroker::search (AlpsModel * *model*) [inline], [virtual]

Search best solution for a given model.

Reimplemented in AlpsKnowledgeBrokerMPI.

Definition at line 273 of file AlpsKnowledgeBroker.h.

3.4.3.6 void AlpsKnowledgeBroker::setPeakMemory (double *size*) [inline]

Set peak memory usage.

Definition at line 298 of file AlpsKnowledgeBroker.h.

3.4.3.7 double AlpsKnowledgeBroker::getPeakMemory ( ) [inline]

Get peak memory usage.

Definition at line 301 of file AlpsKnowledgeBroker.h.

3.4.3.8 void AlpsKnowledgeBroker::setupKnowledgePools ( )

Set up knowledge pools for this broker.

3.4.3.9 virtual int AlpsKnowledgeBroker::getNumKnowledges (AlpsKnowledgeType *kt*) const [virtual]

Query the number of knowledge in the given type of a knowledge pool.

3.4.3.10 virtual int AlpsKnowledgeBroker::getMaxNumKnowledges (AlpsKnowledgeType *kt*) const [inline], [virtual]

Query the max number of knowledge can be stored in a given type of knowledge pools.

Definition at line 339 of file AlpsKnowledgeBroker.h.

3.4.3.11 virtual void AlpsKnowledgeBroker::setMaxNumKnowledges (AlpsKnowledgeType *kt,* int *num*) [inline], [virtual]

Set the max number of knowledge can be stored in a given type o fknowledge pools.

Definition at line 351 of file AlpsKnowledgeBroker.h.

3.4.3.12 virtual bool AlpsKnowledgeBroker::hasKnowledge (AlpsKnowledgeType *kt*) const [inline], [virtual]

Query whether there are knowledges in the given type of knowledge pools.

Definition at line 363 of file AlpsKnowledgeBroker.h.

3.4.3.13 virtual void AlpsKnowledgeBroker::popKnowledge (AlpsKnowledgeType *kt*) [inline], [virtual]

Remove the a knowledge from the given type of knowledge pools.

Definition at line 384 of file AlpsKnowledgeBroker.h.

3.4.3.14 virtual std::pair< AlpsKnowledge *, double > AlpsKnowledgeBroker::getBestKnowledge (AlpsKnowledgeType *kt*) const [virtual]

Get the best knowledge in the given type of knowledge pools.

3.4.3.15 virtual void AlpsKnowledgeBroker::getAllKnowledges (AlpsKnowledgeType *kt,* std::vector< std::pair< AlpsKnowledge *, double > > & *kls*) const [inline], [virtual]

Get all knowledges in the given type of knowledge pools.

Definition at line 399 of file AlpsKnowledgeBroker.h.

3.4.3.16 virtual void AlpsKnowledgeBroker::addKnowledge (AlpsKnowledgeType *kt,* AlpsKnowledge * *kl,* double *value*) [inline], [virtual]

Add a knowledge in the given type of knowledge pools.

Definition at line 412 of file AlpsKnowledgeBroker.h.

3.4.3.17 int AlpsKnowledgeBroker::getNumNodesProcessed ( ) const [inline]

Query the number of node processed by this process.

Definition at line 429 of file AlpsKnowledgeBroker.h.

3.4.3.18 int AlpsKnowledgeBroker::getNumNodesBranched ( ) const [inline]

Query the number of node processed by this process.

Definition at line 434 of file AlpsKnowledgeBroker.h.

3.4.3.19 int AlpsKnowledgeBroker::getNumNodesDiscarded ( ) const [inline]

Query the number of node processed by this process.

Definition at line 439 of file AlpsKnowledgeBroker.h.

3.4.3.20 int AlpsKnowledgeBroker::getNumNodesPartial ( ) const [inline]

Query the number of node in the queue that are pregnant.

Definition at line 444 of file AlpsKnowledgeBroker.h.

3.4.3.21 int AlpsKnowledgeBroker::getNumNodesProcessedSystem ( ) const [inline]

Query the number of node processed by the system.

Definition at line 449 of file AlpsKnowledgeBroker.h.

3.4.3.22 virtual int AlpsKnowledgeBroker::updateNumNodesLeft ( ) [virtual]

Update the number of left nodes on this process.

3.4.3.23 virtual AlpsTreeNode* AlpsKnowledgeBroker::getBestNode ( ) const [virtual]

Query the best node in the subtree pool.

Return NULL if no node exits.

3.4.3.24 AlpsExitStatus AlpsKnowledgeBroker::getSolStatus ( ) const [inline]

Query search termination status.

Definition at line 460 of file AlpsKnowledgeBroker.h.

3.4.3.25 void AlpsKnowledgeBroker::setExitStatus (AlpsExitStatus *status*) [inline]

Set terminate status.

Definition at line 465 of file AlpsKnowledgeBroker.h.

3.4.3.26 AlpsTimer& AlpsKnowledgeBroker::timer ( ) [inline]

Query timer.

Definition at line 470 of file AlpsKnowledgeBroker.h.

3.4.3.27 AlpsTimer& AlpsKnowledgeBroker::subTreeTimer ( ) [inline]

Query subtree timer.

Definition at line 475 of file AlpsKnowledgeBroker.h.

3.4.3.28 AlpsTimer& AlpsKnowledgeBroker::tempTimer ( ) [inline]

Query secondary timer.

Definition at line 480 of file AlpsKnowledgeBroker.h.

3.4.3.29 virtual void AlpsKnowledgeBroker::searchLog ( ) [pure virtual]

Search statistics log.

Implemented in AlpsKnowledgeBrokerMPI, and AlpsKnowledgeBrokerSerial.

3.4.3.30 virtual double AlpsKnowledgeBroker::getIncumbentValue ( ) const [pure virtual]

The process queries the objective value of the incumbent that it stores.

Implemented in AlpsKnowledgeBrokerMPI, and AlpsKnowledgeBrokerSerial.

3.4.3.31 virtual double AlpsKnowledgeBroker::getBestQuality ( ) const [pure virtual]

The process (serial) / the master (parallel) queries the quality of the best solution that it knows.

Implemented in AlpsKnowledgeBrokerMPI, and AlpsKnowledgeBrokerSerial.

3.4.3.32 virtual double AlpsKnowledgeBroker::getBestEstimateQuality ( ) [inline], [virtual]

Get best estimalted quality in system.

Reimplemented in AlpsKnowledgeBrokerMPI.

Definition at line 519 of file AlpsKnowledgeBroker.h.

3.4.3.33 virtual void AlpsKnowledgeBroker::printBestSolution (char * *outputFile* = 0) const [pure virtual]

The process (serial) / the master (parallel) outputs the best solution that it knows to a file or std::out.

Implemented in AlpsKnowledgeBrokerMPI, and AlpsKnowledgeBrokerSerial.

3.4.3.34 virtual int AlpsKnowledgeBroker::getProcRank ( ) const [inline], [virtual]

Qeury the global rank of process.

Note: not useful for serial code.

Reimplemented in AlpsKnowledgeBrokerMPI.

Definition at line 529 of file AlpsKnowledgeBroker.h.

3.4.3.35 virtual int AlpsKnowledgeBroker::getMasterRank ( ) const [inline], [virtual]

Query the global rank of the Master.

Reimplemented in AlpsKnowledgeBrokerMPI.

Definition at line 532 of file AlpsKnowledgeBroker.h.

3.4.3.36 AlpsNodeIndex_t AlpsKnowledgeBroker::nextNodeIndex ( ) [inline]

Query the next index assigned to a newly created node, and then increment the nextIndex_ by 1.

Definition at line 544 of file AlpsKnowledgeBroker.h.

3.4.3.37 AlpsNodeIndex_t AlpsKnowledgeBroker::getNextNodeIndex ( ) const [inline]

Query the next index assigned to a newly created node.

Definition at line 547 of file AlpsKnowledgeBroker.h.

3.4.3.38 void AlpsKnowledgeBroker::setNextNodeIndex (AlpsNodeIndex_t *s*) [inline]

Set nextIndex_.

Definition at line 550 of file AlpsKnowledgeBroker.h.

3.4.3.39 AlpsNodeIndex_t AlpsKnowledgeBroker::getMaxNodeIndex ( ) const [inline]

Queriy the upper bound of node indices.

Definition at line 553 of file AlpsKnowledgeBroker.h.

3.4.3.40 void AlpsKnowledgeBroker::setMaxNodeIndex (AlpsNodeIndex_t *s*) [inline]

Set the upper bound of node indices.

Definition at line 556 of file AlpsKnowledgeBroker.h.

3.4.3.41 void AlpsKnowledgeBroker::passInMessageHandler (CoinMessageHandler * *handler*)

Pass in Message handler (not deleted at end).

3.4.3.42 void AlpsKnowledgeBroker::newLanguage (CoinMessages::Language *language*)

Set language.

3.4.3.43 CoinMessageHandler* AlpsKnowledgeBroker::messageHandler ( ) const [inline]

Return handler.

Definition at line 598 of file AlpsKnowledgeBroker.h.

3.4.3.44 CoinMessages AlpsKnowledgeBroker::messages ( ) [inline]

Return messages.

Definition at line 601 of file AlpsKnowledgeBroker.h.

3.4.3.45 CoinMessages* AlpsKnowledgeBroker::messagesPointer ( ) [inline]

Return pointer to messages.

Definition at line 604 of file AlpsKnowledgeBroker.h.

3.4.3.46 int AlpsKnowledgeBroker::getMsgLevel ( ) [inline]

Return msg level.

Definition at line 607 of file AlpsKnowledgeBroker.h.

3.4.3.47 int AlpsKnowledgeBroker::getHubMsgLevel ( ) [inline]

Return msg level.

Definition at line 610 of file AlpsKnowledgeBroker.h.

3.4.3.48 int AlpsKnowledgeBroker::getMasterMsgLevel ( ) [inline]

Return msg level.

Definition at line 613 of file AlpsKnowledgeBroker.h.

3.4.3.49 int AlpsKnowledgeBroker::getlogFileLevel ( ) [inline]

Return log file level.

Definition at line 616 of file AlpsKnowledgeBroker.h.

3.4.4 Member Data Documentation

3.4.4.1 std::string AlpsKnowledgeBroker::instanceName_ [protected]

The instance name.

Definition at line 61 of file AlpsKnowledgeBroker.h.

3.4.4.2 AlpsModel* AlpsKnowledgeBroker::model_ [protected]

Pointer to model.

Definition at line 64 of file AlpsKnowledgeBroker.h.

3.4.4.3 AlpsPhase AlpsKnowledgeBroker::phase_ [protected]

Alps phase.

(RAMPUP, SEARCH, RAMPDOWN)

Definition at line 67 of file AlpsKnowledgeBroker.h.

3.4.4.4 AlpsSubTreePool* AlpsKnowledgeBroker::subTreePool_ [protected]

A subtree pool holding a collection of subtrees.

For serial version, there is only one subtree in the pool.

Definition at line 75 of file AlpsKnowledgeBroker.h.

3.4.4.5 AlpsSolutionPool* AlpsKnowledgeBroker::solPool_ [protected]

A solution pool containing the solutions found.

Definition at line 78 of file AlpsKnowledgeBroker.h.

3.4.4.6 std::map<AlpsKnowledgeType, AlpsKnowledgePool*>* AlpsKnowledgeBroker::pools_ [protected]

The collection of pools managed by the knowledge broker.

Definition at line 81 of file AlpsKnowledgeBroker.h.

3.4.4.7 AlpsSubTree* AlpsKnowledgeBroker::workingSubTree_ [protected]

Point to the subtree that being explored.

Definition at line 89 of file AlpsKnowledgeBroker.h.

3.4.4.8 bool AlpsKnowledgeBroker::needWorkingSubTree_ [protected]

Indicate whether need a new subtree.

Definition at line 92 of file AlpsKnowledgeBroker.h.

3.4.4.9 AlpsNodeIndex_t AlpsKnowledgeBroker::nextIndex_ [protected]

The index to be assigned to a new search tree node.

Definition at line 95 of file AlpsKnowledgeBroker.h.

3.4.4.10 AlpsNodeIndex_t AlpsKnowledgeBroker::maxIndex_ [protected]

The maximum index can been assigned on this process.

Definition at line 98 of file AlpsKnowledgeBroker.h.

3.4.4.11 AlpsTimer AlpsKnowledgeBroker::timer_ [protected]

Main timer.

Do not touch.

Definition at line 107 of file AlpsKnowledgeBroker.h.

3.4.4.12 AlpsTimer AlpsKnowledgeBroker::subTreeTimer_ [protected]

Subtree timer.

Do not touch.

Definition at line 110 of file AlpsKnowledgeBroker.h.

3.4.4.13 AlpsTimer AlpsKnowledgeBroker::tempTimer_ [protected]

Secondary timer.

Definition at line 113 of file AlpsKnowledgeBroker.h.

3.4.4.14 int AlpsKnowledgeBroker::solNum_ [protected]

The number of solutions found.

Definition at line 116 of file AlpsKnowledgeBroker.h.

3.4.4.15 int AlpsKnowledgeBroker::nodeProcessedNum_ [protected]

The number of nodes that have been processed.

Definition at line 119 of file AlpsKnowledgeBroker.h.

3.4.4.16 int AlpsKnowledgeBroker::nodeBranchedNum_ [protected]

The number of nodes that have been branched.

Definition at line 122 of file AlpsKnowledgeBroker.h.

3.4.4.17 int AlpsKnowledgeBroker::nodeDiscardedNum_ [protected]

The number of nodes that have been discarded before processing.

Definition at line 125 of file AlpsKnowledgeBroker.h.

3.4.4.18 int AlpsKnowledgeBroker::nodePartialNum_ [protected]

The number of nodes that are pregnant.

Definition at line 128 of file AlpsKnowledgeBroker.h.

3.4.4.19 int AlpsKnowledgeBroker::systemNodeProcessed_ [protected]

To record how many nodes processed by the system (used in parallel code).

Definition at line 132 of file AlpsKnowledgeBroker.h.

3.4.4.20 int AlpsKnowledgeBroker::nodeLeftNum_ [protected]

The number of nodes left.

Definition at line 135 of file AlpsKnowledgeBroker.h.

3.4.4.21 int AlpsKnowledgeBroker::treeDepth_ [protected]

The depth of the tree.

Definition at line 138 of file AlpsKnowledgeBroker.h.

3.4.4.22 int AlpsKnowledgeBroker::bestSolNode_ [protected]

The number of nodes pocessed to find the solution.

Definition at line 141 of file AlpsKnowledgeBroker.h.

3.4.4.23 double AlpsKnowledgeBroker::peakMemory_ [protected]

Peak memory usage.

Definition at line 144 of file AlpsKnowledgeBroker.h.

3.4.4.24 AlpsExitStatus AlpsKnowledgeBroker::exitStatus_ [protected]

The status of search when terminated.

Definition at line 147 of file AlpsKnowledgeBroker.h.

3.4.4.25 AlpsSearchStrategy<AlpsSubTree*>* AlpsKnowledgeBroker::treeSelection_ [protected]

Tree selection criterion.

Definition at line 155 of file AlpsKnowledgeBroker.h.

3.4.4.26 AlpsSearchStrategy<AlpsTreeNode*>* AlpsKnowledgeBroker::nodeSelection_ [protected]

Node selection criterion.

Definition at line 158 of file AlpsKnowledgeBroker.h.

3.4.4.27 AlpsSearchStrategy<AlpsTreeNode*>* AlpsKnowledgeBroker::rampUpNodeSelection_ [protected]

Node selection criterion.

Definition at line 161 of file AlpsKnowledgeBroker.h.

3.4.4.28 CoinMessageHandler* AlpsKnowledgeBroker::handler_ [protected]

Message handler.

Definition at line 169 of file AlpsKnowledgeBroker.h.

3.4.4.29 CoinMessages AlpsKnowledgeBroker::messages_ [protected]

Alps messages.

Definition at line 172 of file AlpsKnowledgeBroker.h.

3.4.4.30 int AlpsKnowledgeBroker::msgLevel_ [protected]

The leve of printing message to screen of the master and general message.

(0: no; 1: basic; 2: moderate, 3: verbose)

Definition at line 176 of file AlpsKnowledgeBroker.h.

3.4.4.31 int AlpsKnowledgeBroker::hubMsgLevel_ [protected]

The leve of printing message to screen of hubs.

(0: no; 1: basic; 2: moderate, 3: verbose)

Definition at line 180 of file AlpsKnowledgeBroker.h.

3.4.4.32 int AlpsKnowledgeBroker::workerMsgLevel_ [protected]

The leve of printing message to screen of workers.

(0: no; 1: basic; 2: moderate, 3: verbose)

Definition at line 184 of file AlpsKnowledgeBroker.h.

3.4.4.33 int AlpsKnowledgeBroker::logFileLevel_ [protected]

The degree of log file.

(0: no; 1: basic; 2: moderate, 3: verbose)

Definition at line 188 of file AlpsKnowledgeBroker.h.

3.4.4.34 std::string AlpsKnowledgeBroker::logfile_ [protected]

The log file.

Definition at line 191 of file AlpsKnowledgeBroker.h.

3.4.4.35 int AlpsKnowledgeBroker::nodeMemSize_ [protected]

The approximate memory size (bytes) of a node with full description.

Definition at line 195 of file AlpsKnowledgeBroker.h.

3.4.4.36 double AlpsKnowledgeBroker::nodeProcessingTime_ [protected]

The approximately CPU time to process a node.

Definition at line 198 of file AlpsKnowledgeBroker.h.

3.4.4.37 int AlpsKnowledgeBroker::largeSize_ [protected]

The size of largest message buffer can be sent or received.

Definition at line 201 of file AlpsKnowledgeBroker.h.

3.4.4.38 int AlpsKnowledgeBroker::numNodeLog_ [protected]

Times that node log is printed.

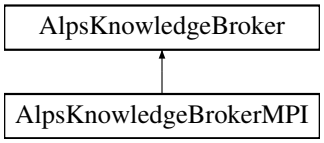
Definition at line 207 of file AlpsKnowledgeBroker.h.

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

- AlpsKnowledgeBroker.h

3.5 AlpsKnowledgeBrokerMPI Class Reference

Inheritance diagram for AlpsKnowledgeBrokerMPI:



Public Member Functions

- AlpsKnowledgeBrokerMPI ()
  Default construtor.
- AlpsKnowledgeBrokerMPI (int argc, char *argv[], AlpsModel &model)
  Useful construtor.
- ~AlpsKnowledgeBrokerMPI ()
  Destructor.
- virtual int getProcRank () const
  Query the global rank of the process.
- virtual int getMasterRank () const
  Query the global rank of the Master.
- virtual AlpsProcessType getProcType () const
  Query the type (master, hub, or worker) of the process.
- void initializeSearch (int argc, char *argv[], AlpsModel &model)
  This function.
- void search (AlpsModel *model)
  Search best solution for a given model.
- void rootSearch (AlpsTreeNode *root)
  This function.

Report search results.

- virtual double getIncumbentValue () const
  The process queries the quality of the incumbent this process stores.
- virtual double getBestQuality () const
  The master queries the quality of the best solution it knowns.
- virtual double getBestEstimateQuality ()
  Get best estimalted quality in system.
- virtual void printBestSolution (char *outputFile=0) const
  Master prints out the best solution that it knows.
- virtual void searchLog ()
  Log search statistics.

Knowledge sharing functions

- void sendKnowledge (AlpsKnowledgeType type, int sender, int receiver, char *&msgBuffer, int msgSize, int msgTag, MPI_Comm comm, bool blocking)
  Set knowlege.
- void receiveKnowledge (AlpsKnowledgeType type, int sender, int receiver, char *&msgBuffer, int msgSize, int msgTag, MPI_Comm comm, MPI_Status *status, bool blocking)
  Receive knowlege.
- void requestKnowledge (AlpsKnowledgeType type, int sender, int receiver, char *&msgBuffer, int msgSize, int msgTag, MPI_Comm comm, bool blocking)
  Request knowlege.

Protected Member Functions

- void init ()
  Initialize member data.
- AlpsReturnStatus doOneUnitWork (int unitWork, double unitTime, AlpsExitStatus &exitStatus, int &numNodesProcessed, int &numNodesBranched, int &numNodesDiscarded, int &numNodesPartial, int &depth, bool &betterSolution)
  Explore a subtree from subtree pool for certain units of work/time.
- void processMessages (char *&buffer, MPI_Status &status, MPI_Request &request)
  Processing messages.
- void rootInitMaster (AlpsTreeNode *root)
  Static load balancing: Root Initialization.
- void spiralMaster (AlpsTreeNode *root)
  Static load balancing: spiral.
- void deleteSubTrees ()
  Delete subTrees in pools and the active subtree.
- void sendModelKnowledge (MPI_Comm comm, int receiver=-1)
  Set generated knowlege (related to model) to receiver.
- void receiveModelKnowledge (MPI_Comm comm)
  Receive generated knowlege (related to model) from sender.
- void masterForceHubTerm ()
  Master tell hubs to terminate due to reaching limits or other reason.
- void hubForceWorkerTerm ()
  Hub tell workers to terminate due to reaching limits or other reason.
- void changeWorkingSubTree (double &changeWorkThreshold)

Change subtree to be explored if it is too worse.

- void sendErrorCodeToMaster (int errorCode)

  Send error code to master.
- void recvErrorCode (char *&bufLarge)

  Receive error code and set solution status.
- void spiralRecvProcessNode ()

  Unpack the node, explore it and send load info to master.
- void spiralDonateNode ()

  Unpack msg and donate a node.

Core member functions for master, hubs and workers.

- void masterMain (AlpsTreeNode *root)

  Master generates subtrees and sends them to hubs in Round-Robin way.
- void hubMain ()

  Hub generates subtrees and sends them to workers in Round-Robin way.
- void workerMain ()

  Worker first receive subtrees, then start to explore them.

Load balancing member functions

- void masterAskHubDonate (int donorID, int receiverID, double receiverWorkload)

  Master asks a hub to donate its workload to another hub.
- void hubAskWorkerDonate (int donorID, int receiverID, double receiverWorkload)

  Hub asks a worker to donate its workload to another worker.
- void updateWorkloadInfo ()

  Calculate the work quality and quantity on this process.
- virtual int **getNumNodeLeftSystem** ()
- void donateWork (char *&buf, int tag, MPI_Status *status, int recvID=-1, double recvWL=0.0)

  A worker donate its workload to the specified worker.
- void hubAllocateDonation (char *&buf, MPI_Status *status)

  Hub allocates the donated workload to its workers.
- void hubBalanceWorkers ()

  Hub balances the workloads of its workers.
- void hubSatisfyWorkerRequest (char *&buf, MPI_Status *status)

  Hub satisfies the workload rquest from a worker.
- void hubReportStatus (int tag, MPI_Comm comm)

  A hub reports its status (workload and msg counts) to the master.
- void hubUpdateCluStatus (char *&buf, MPI_Status *status, MPI_Comm comm)

  A hub unpacks the status of a worker from buffer.
- void hubsShareWork (char *&buf, MPI_Status *status)

  Two hubs share their workload.
- void masterBalanceHubs ()

  Master balance the workload of hubs.
- void masterUpdateSysStatus (char *&buf, MPI_Status *status, MPI_Comm comm)

  Master unpack the status of a hub from buf and update system status.
- void refreshSysStatus ()

  The master re-calculate the system status.
- void refreshClusterStatus ()

  A hub adds its status to the cluster's status.
- void workerReportStatus (int tag, MPI_Comm comm)

  A worker report its status (workload and msg counts) to its hub.

Node index functions // msg counts is modified inside

- void workerAskIndices ()
  A worker ask for node index from master.
- void workerRecvIndices (char *&bufLarge)
  A worker receive node index from master.
- void masterSendIndices (char *&bufLarge)
  Master send a batch of node indices to the receiving worker.

Other message passing member functions

- void broadcastModel (const int id, const int source)
  Broadcast the model from source to other processes.
- void sendIncumbent ()
  Sent the incumbent value and rank to its two child if eixt.
- bool unpackSetIncumbent (char *&buf, MPI_Status *status)
  unpack the incumbent value, then store it and the id of the process having the incumbent in AlpsDataPool.
- void collectBestSolution (int destination)
  Send the best solution from the process having it to destination.
- void tellMasterRecv ()
  Inform master that a proc has received workload during a load balance initialized by master.
- void tellHubRecv ()
  Inform hub that a proc has received workload during a load balance initialized by a hub.
- void packEncoded (AlpsEncoded *enc, char *&buf, int &size, int &position, MPI_Comm comm)
  Pack an AlpsEncoded instance into buf.
- AlpsEncoded * unpackEncoded (char *&buf, int &position, MPI_Comm comm, int size=-1)
  Unpack the given buffer into an AlpsEncoded instance.
- void receiveSizeBuf (char *&buf, int sender, int tag, MPI_Comm comm, MPI_Status *status)
  Receive the size of buffer, allocate memory for buffer, then receive the message and put it in buffer.
- void receiveRampUpNode (int sender, MPI_Comm comm, MPI_Status *status)
  First receive the size and the contend of a node, then construct a subtree with this received node.
- void receiveSubTree (char *&buf, int sender, MPI_Status *status)
  Receive a subtree from the sender process and add it into the subtree pool.
- void sendSizeBuf (char *&buf, int size, int position, const int target, const int tag, MPI_Comm comm)
  Send the size and content of a buffer to the target process.
- void sendRampUpNode (const int target, MPI_Comm comm)
  Send the size and the content of the best node of a given subtree to the target process.
- void sendNodeModelGen (int receiver, int doUnitWork)
  Send a node from rampUpSubTree's node pool and generated model knowledge.
- bool sendSubTree (const int target, AlpsSubTree *&st, int tag)
  Send a given subtree to the target process.
- void sendFinishInit (const int target, MPI_Comm comm)
  Send finish initialization signal to the target process.

Change message counts functions

- void incSendCount (const char *how, int s=1)
  Increment the number of sent message.
- void decSendCount (const char *how, int s=1)
  Decrement the number of sent message.
- void incRecvCount (const char *how, int s=1)
  Increment the number of received message.
- void decRecvCount (const char *how, int s=1)
  Decrement the number of sent message.

Protected Attributes

- bool forceTerminate_

  Terminate due to reaching limits (time and node) or other reason.
- bool blockTermCheck_

  Indicate whether do termination check.
- bool blockHubReport_

  Indicate whether a hub need to report state to master.
- bool blockWorkerReport_

  Indicate whether a worker need to report state to its hub.
- bool blockAskForWork_

  Indicate whether a worker need to as for work from its hub.
- char * attachBuffer_

  Buffer attached to MPI when sharing generated knowledge.
- char * largeBuffer_

  Large message buffer.
- char * largeBuffer2_

  Large message buffer.
- char * smallBuffer_

  Small message buffer.
- double masterBalancePeriod_

  The period that master do load balancing.
- double hubReportPeriod_

  The period that a hub load balancing and report cluster status.
- int modelGenID_

  The global rank of the process that share generated model knowledge.
- int modelGenPos_

  Size of the shared knowledge.
- AlpsSubTree * rampUpSubTree_

  A subtree used in during up.
- int unitWorkNodes_

  Number of nodes in one unit of work.
- int haltSearch_

  Temporily halt search.

Process information

- int processNum_

  The Number of processes launched.
- int hubNum_

  The Number of hubs.
- int globalRank_

  The rank of the process in MPI_COMM_WORLD.
- MPI_Comm clusterComm_

  Communicator of the cluster to which the process belongs.
- MPI_Comm hubComm_

  Communicator consists of all hubs.
- MPI_Group hubGroup_

  MPI_Group consists of all hubs.

- int clusterSize_
  The actual size of the cluster to which the process belongs.
- int userClusterSize_
  The user reqested size of a cluster.
- int clusterRank_
  The local rank of the process in clusterComm_.
- int * hubRanks_
  The global ranks of the hubs.
- int myHubRank_
  The global rank of its hub for a worker.
- int masterRank_
  The global rank of the master.
- AlpsProcessType processType_
  The AlpsProcessType of this process.
- AlpsProcessType * processTypeList_
  The AlpsProcessType of all process.
- bool hubWork_
  Whether hub should also work as a worker.
- MPI_Request subTreeRequest_
  Send subtree request.
- MPI_Request solRequestL_
  Send model knoledge request.
- MPI_Request **solRequestR_**
- MPI_Request modelKnowRequestL_
  Send model knoledge request.
- MPI_Request **modelKnowRequestR_**
- MPI_Request forwardRequestL_
  Forward model knoledge request.
- MPI_Request **forwardRequestR_**

Incumbent data

- double incumbentValue_
  Incumbent value.
- int incumbentID_
  The process id that store the incumbent.
- bool updateIncumbent_
  Indicate whether the incumbent value is updated between two checking point.

Workload balancing

- double workQuality_
  The workload quality of the process.
- double clusterWorkQuality_
  The workload quality of the cluster to which the process belong.
- double systemWorkQuality_
  The workload quality of the whole system.
- double * hubWorkQualities_
  The workload qualities of hubs.
- double * workerWorkQualities_
  The workload qualities of workers in the cluster to which this proces belongs.
- double workQuantity_
  The workload quantity of the workload on the process.
- double clusterWorkQuantity_
  The workload quantity of the cluster to which the process belongs.

- double systemWorkQuantity_
  The workload quantity of the whole system.
- double systemWorkQuantityForce_
  The workload quantity of the whole system before forcing termination.
- double * hubWorkQuantities_
  The workload quantities of all clusters/hubs.
- double * workerWorkQuantities_
  The workload quantities of workers in the cluster to which this proces belongs.
- bool * workerReported_
  Indicate which worker has been reported its work.
- bool * hubReported_
  Indicate which hub has been reported its work.
- bool allHubReported_
  Indicate whether all hubs have reported status to master at least once.
- int masterDoBalance_
  Whether master do load balance.
- int hubDoBalance_
  Whether a hub do load balance.
- int * workerNodeProcesseds_
  To record how many nodes processed for each worker in a cluster.
- int clusterNodeProcessed_
  To record how many nodes by a cluster.
- int * hubNodeProcesseds_
  To record how many nodes processed for each hub.

Message counts

- int sendCount_
  The number of new messages sent by the process after last survey.
- int recvCount_
  The number of new messages received by the process after last survey.
- int clusterSendCount_
  The number of new messages sent by the processes in clusterComm_ after last survey.
- int clusterRecvCount_
  The number of new messages received by the processes in clusterComm_ after last survey.
- int systemSendCount_
  The total number of messages sent by the all processes.
- int systemRecvCount_
  The total number of messages sent by the all processes.

Node index

- int **masterIndexBatch_**

Parallel statistics

- AlpsTimer masterTimer_
  Master timer.
- AlpsTimer hubTimer_
  Hub timer.
- AlpsTimer workerTimer_
  Worker timer.
- double rampUpTime_
  The time spent in ramp up.

- double rampDownTime_
  The time spent in ramp down.
- double idleTime_
  The time spent waiting for work.
- double msgTime_
  The time spent processing messages (include idle).
- AlpsPsStats psStats_
  More statistics.

3.5.1 Detailed Description

Definition at line 41 of file AlpsKnowledgeBrokerMPI.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 AlpsKnowledgeBrokerMPI::AlpsKnowledgeBrokerMPI ( ) [inline]

Default construtor.

NOTE: must call initializeSearch() later.

Definition at line 579 of file AlpsKnowledgeBrokerMPI.h.

3.5.2.2 AlpsKnowledgeBrokerMPI::AlpsKnowledgeBrokerMPI (int *argc*, char * *argv[]*, AlpsModel & *model*) [inline]

Useful construtor.

Definition at line 587 of file AlpsKnowledgeBrokerMPI.h.

3.5.2.3 AlpsKnowledgeBrokerMPI::~AlpsKnowledgeBrokerMPI ( )

Destructor.

3.5.3 Member Function Documentation

3.5.3.1 void AlpsKnowledgeBrokerMPI::init ( ) [protected]

Initialize member data.

3.5.3.2 void AlpsKnowledgeBrokerMPI::masterMain (AlpsTreeNode * *root*) [protected]

Master generates subtrees and sends them to hubs in Round-Robin way.

Master periodically do inter-cluster load balancing, and termination check.

3.5.3.3 void AlpsKnowledgeBrokerMPI::hubMain ( ) [protected]

Hub generates subtrees and sends them to workers in Round-Robin way.

Hub do intra-cluster load balancing.

3.5.3.4 void AlpsKnowledgeBrokerMPI::workerMain ( ) [protected]

Worker first receive subtrees, then start to explore them.

Worker also peroidically check message and process message.

3.5.3.5 AlpsReturnStatus AlpsKnowledgeBrokerMPI::doOneUnitWork (int *unitWork,* double *unitTime,* AlpsExitStatus & *exitStatus,* int & *numNodesProcessed,* int & *numNodesBranched,* int & *numNodesDiscarded,* int & *numNodesPartial,* int & *depth,* bool & *betterSolution*) [protected]

Explore a subtree from subtree pool for certain units of work/time.

3.5.3.6 void AlpsKnowledgeBrokerMPI::processMessages (char *&* *buffer,* MPI_Status & *status,* MPI_Request & *request*) [protected]

Processing messages.

3.5.3.7 void AlpsKnowledgeBrokerMPI::masterAskHubDonate (int *donorID,* int *receiverID,* double *receiverWorkload*) [protected]

Master asks a hub to donate its workload to another hub.

3.5.3.8 void AlpsKnowledgeBrokerMPI::hubAskWorkerDonate (int *donorID,* int *receiverID,* double *receiverWorkload*) [protected]

Hub asks a worker to donate its workload to another worker.

3.5.3.9 void AlpsKnowledgeBrokerMPI::updateWorkloadInfo ( ) [protected]

Calculate the work quality and quantity on this process.

3.5.3.10 void AlpsKnowledgeBrokerMPI::donateWork (char *&* *buf,* int *tag,* MPI_Status * *status,* int *recvID* = -1, double *recvWL* = 0.0) [protected]

A worker donate its workload to the specified worker.

3.5.3.11 void AlpsKnowledgeBrokerMPI::hubAllocateDonation (char *&* *buf,* MPI_Status * *status*) [protected]

Hub allocates the donated workload to its workers.

3.5.3.12 void AlpsKnowledgeBrokerMPI::hubBalanceWorkers ( ) [protected]

Hub balances the workloads of its workers.

3.5.3.13 void AlpsKnowledgeBrokerMPI::hubSatisfyWorkerRequest (char *&* *buf,* MPI_Status * *status*) [protected]

Hub satisfies the workload rquest from a worker.

3.5.3.14 void AlpsKnowledgeBrokerMPI::hubReportStatus (int *tag,* MPI_Comm *comm*) [protected]

A hub reports its status (workload and msg counts) to the master.

3.5.3.15 void AlpsKnowledgeBrokerMPI::hubUpdateCluStatus (char *&* *buf,* MPI_Status * *status,* MPI_Comm *comm*) [protected]

A hub unpacks the status of a worker from buffer.

3.5.3.16 void AlpsKnowledgeBrokerMPI::hubsShareWork (char *&* *buf,* MPI_Status * *status*) [protected]

Two hubs share their workload.

3.5.3.17 void AlpsKnowledgeBrokerMPI::masterBalanceHubs ( ) [protected]

Master balance the workload of hubs.

3.5.3.18 void AlpsKnowledgeBrokerMPI::masterUpdateSysStatus (char *& *buf,* MPI_Status * *status,* MPI_Comm *comm*) [protected]

Master unpack the status of a hub from buf and update system status.

3.5.3.19 void AlpsKnowledgeBrokerMPI::refreshSysStatus ( ) [protected]

The master re-calculate the system status.

3.5.3.20 void AlpsKnowledgeBrokerMPI::refreshClusterStatus ( ) [protected]

A hub adds its status to the cluster's status.

3.5.3.21 void AlpsKnowledgeBrokerMPI::workerReportStatus (int *tag,* MPI_Comm *comm*) [protected]

A worker report its status (workload and msg counts) to its hub.

3.5.3.22 void AlpsKnowledgeBrokerMPI::workerAskIndices ( ) [protected]

A worker ask for node index from master.

3.5.3.23 void AlpsKnowledgeBrokerMPI::workerRecvIndices (char *& *bufLarge*) [protected]

A worker receive node index from master.

3.5.3.24 void AlpsKnowledgeBrokerMPI::masterSendIndices (char *& *bufLarge*) [protected]

Master send a batch of node indices to the receiving worker.

3.5.3.25 void AlpsKnowledgeBrokerMPI::broadcastModel (const int *id,* const int *source*) [protected]

Broadcast the model from source to other processes.

3.5.3.26 bool AlpsKnowledgeBrokerMPI::unpackSetIncumbent (char *& *buf,* MPI_Status * *status*) [protected]

unpack the incumbent value, then store it and the id of the process having the incumbent in AlpsDataPool.

3.5.3.27 void AlpsKnowledgeBrokerMPI::collectBestSolution (int *destination*) [protected]

Send the best solution from the process having it to destination.

3.5.3.28 void AlpsKnowledgeBrokerMPI::tellMasterRecv ( ) [protected]

Inform master that a proc has received workload during a load balance initialized by master.

3.5.3.29 void AlpsKnowledgeBrokerMPI::tellHubRecv ( ) [protected]

Inform hub that a proc has received workload during a load balance initialized by a hub.

3.5.3.30 void AlpsKnowledgeBrokerMPI::packEncoded (AlpsEncoded * *enc,* char *& *buf,* int & *size,* int & *position,* MPI_Comm *comm*) [protected]

Pack an AlpsEncoded instance into buf.

Return filled buf and size of packed message. position: where to start if buf is allocated.

3.5.3.31 **AlpsEncoded*** **AlpsKnowledgeBrokerMPI::unpackEncoded (char *& *buf,* int & *position,* MPI_Comm *comm,* int *size =* −1)** [protected]

Unpack the given buffer into an AlpsEncoded instance.

3.5.3.32 **void AlpsKnowledgeBrokerMPI::receiveSizeBuf (char *& *buf,* int *sender,* int *tag,* MPI_Comm *comm,* MPI_Status * *status*)** [protected]

Receive the size of buffer, allocate memory for buffer, then receive the message and put it in buffer.

3.5.3.33 **void AlpsKnowledgeBrokerMPI::receiveRampUpNode (int *sender,* MPI_Comm *comm,* MPI_Status * *status*)** [protected]

First receive the size and the contend of a node, then construct a subtree with this received node.

3.5.3.34 **void AlpsKnowledgeBrokerMPI::receiveSubTree (char *& *buf,* int *sender,* MPI_Status * *status*)** [protected]

Receive a subtree from the sender process and add it into the subtree pool.

3.5.3.35 **void AlpsKnowledgeBrokerMPI::sendSizeBuf (char *& *buf,* int *size,* int *position,* const int *target,* const int *tag,* MPI_Comm *comm*)** [protected]

Send the size and content of a buffer to the target process.

3.5.3.36 **void AlpsKnowledgeBrokerMPI::sendRampUpNode (const int *target,* MPI_Comm *comm*)** [protected]

Send the size and the content of the best node of a given subtree to the target process.

3.5.3.37 **bool AlpsKnowledgeBrokerMPI::sendSubTree (const int *target,* AlpsSubTree *& *st,* int *tag*)** [protected]

Send a given subtree to the target process.

3.5.3.38 **void AlpsKnowledgeBrokerMPI::sendFinishInit (const int *target,* MPI_Comm *comm*)** [protected]

Send finish initialization signal to the target process.

3.5.3.39 **void AlpsKnowledgeBrokerMPI::deleteSubTrees ( )** [protected]

Delete subTrees in pools and the active subtree.

3.5.3.40 **void AlpsKnowledgeBrokerMPI::sendModelKnowledge (MPI_Comm *comm,* int *receiver =* −1)** [protected]

Set generated knowlege (related to model) to receiver.

3.5.3.41 **void AlpsKnowledgeBrokerMPI::receiveModelKnowledge (MPI_Comm *comm*)** [protected]

Receive generated knowlege (related to model) from sender.

3.5.3.42 **void AlpsKnowledgeBrokerMPI::incSendCount (const char * *how,* int *s =* 1)** [protected]

Increment the number of sent message.

3.5.3.43 **void AlpsKnowledgeBrokerMPI::decSendCount (const char * *how,* int *s =* 1)** [protected]

Decrement the number of sent message.

3.5.3.44 void AlpsKnowledgeBrokerMPI::incRecvCount (const char * *how,* int *s* = 1) [protected]

Increment the number of received message.

3.5.3.45 void AlpsKnowledgeBrokerMPI::decRecvCount (const char * *how,* int *s* = 1) [protected]

Decrement the number of sent message.

3.5.3.46 void AlpsKnowledgeBrokerMPI::masterForceHubTerm ( ) [protected]

Master tell hubs to terminate due to reaching limits or other reason.

3.5.3.47 void AlpsKnowledgeBrokerMPI::hubForceWorkerTerm ( ) [protected]

Hub tell workers to terminate due to reaching limits or other reason.

3.5.3.48 void AlpsKnowledgeBrokerMPI::changeWorkingSubTree (double & *changeWorkThreshold*) [protected]

Change subtree to be explored if it is too worse.

3.5.3.49 void AlpsKnowledgeBrokerMPI::sendErrorCodeToMaster (int *errorCode*) [protected]

Send error code to master.

3.5.3.50 void AlpsKnowledgeBrokerMPI::recvErrorCode (char *& *bufLarge*) [protected]

Receive error code and set solution status.

3.5.3.51 void AlpsKnowledgeBrokerMPI::spiralRecvProcessNode ( ) [protected]

Unpack the node, explore it and send load info to master.

3.5.3.52 void AlpsKnowledgeBrokerMPI::spiralDonateNode ( ) [protected]

Unpack msg and donate a node.

3.5.3.53 virtual int AlpsKnowledgeBrokerMPI::getProcRank ( ) const [inline], [virtual]

Query the global rank of the process.

Reimplemented from AlpsKnowledgeBroker.

Definition at line 601 of file AlpsKnowledgeBrokerMPI.h.

3.5.3.54 virtual int AlpsKnowledgeBrokerMPI::getMasterRank ( ) const [inline], [virtual]

Query the global rank of the Master.

Reimplemented from AlpsKnowledgeBroker.

Definition at line 604 of file AlpsKnowledgeBrokerMPI.h.

3.5.3.55 virtual AlpsProcessType AlpsKnowledgeBrokerMPI::getProcType ( ) const [inline], [virtual]

Query the type (master, hub, or worker) of the process.

Reimplemented from AlpsKnowledgeBroker.

Definition at line 607 of file AlpsKnowledgeBrokerMPI.h.

3.5.3.56 void AlpsKnowledgeBrokerMPI::initializeSearch (int *argc*, char * *argv[]*, AlpsModel & *model*) [virtual]

This function.

- initializes the message environment;
- the master reads in ALPS and user's parameter sets. If the model data is input from file, then it reads in the model data.
- sets up user params and model;
- broadcast parameters from the master to all other processes;
- creates MPI communicators and groups;
- classifies process types, sets up subtree and pools
- determines their hub's global rank for workers

Implements AlpsKnowledgeBroker.

3.5.3.57 void AlpsKnowledgeBrokerMPI::search (AlpsModel * *model*) [virtual]

Search best solution for a given model.

Reimplemented from AlpsKnowledgeBroker.

3.5.3.58 void AlpsKnowledgeBrokerMPI::rootSearch (AlpsTreeNode * *root*) [virtual]

This function.

- broadcasts model data from the master to all other processes;
- calls its associated main function to explore the sub tree;
- collects the best solution found.

Implements AlpsKnowledgeBroker.

3.5.3.59 virtual double AlpsKnowledgeBrokerMPI::getIncumbentValue ( ) const [inline], [virtual]

The process queries the quality of the incumbent this process stores.

Implements AlpsKnowledgeBroker.

Definition at line 638 of file AlpsKnowledgeBrokerMPI.h.

3.5.3.60 virtual double AlpsKnowledgeBrokerMPI::getBestQuality ( ) const [inline], [virtual]

The master queries the quality of the best solution it knowns.

Implements AlpsKnowledgeBroker.

Definition at line 650 of file AlpsKnowledgeBrokerMPI.h.

3.5.3.61 virtual double AlpsKnowledgeBrokerMPI::getBestEstimateQuality ( ) [inline], [virtual]

Get best estimalted quality in system.

Reimplemented from AlpsKnowledgeBroker.

Definition at line 665 of file AlpsKnowledgeBrokerMPI.h.

3.5.3.62 virtual void AlpsKnowledgeBrokerMPI::printBestSolution (char * *outputFile* = 0) const [virtual]

Master prints out the best solution that it knows.

Implements AlpsKnowledgeBroker.

3.5.3.63 virtual void AlpsKnowledgeBrokerMPI::searchLog ( ) [virtual]

Log search statistics.

Implements AlpsKnowledgeBroker.

3.5.3.64 void AlpsKnowledgeBrokerMPI::sendKnowledge (AlpsKnowledgeType *type,* int *sender,* int *receiver,* char *& *msgBuffer,* int *msgSize,* int *msgTag,* MPI_Comm *comm,* bool *blocking*)

Set knowlege.

3.5.3.65 void AlpsKnowledgeBrokerMPI::receiveKnowledge (AlpsKnowledgeType *type,* int *sender,* int *receiver,* char *& *msgBuffer,* int *msgSize,* int *msgTag,* MPI_Comm *comm,* MPI_Status * *status,* bool *blocking*)

Receive knowlege.

3.5.3.66 void AlpsKnowledgeBrokerMPI::requestKnowledge (AlpsKnowledgeType *type,* int *sender,* int *receiver,* char *& *msgBuffer,* int *msgSize,* int *msgTag,* MPI_Comm *comm,* bool *blocking*)

Request knowlege.

3.5.4 Member Data Documentation

3.5.4.1 int AlpsKnowledgeBrokerMPI::processNum_ [protected]

The Number of processes launched.

Definition at line 55 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.2 int AlpsKnowledgeBrokerMPI::hubNum_ [protected]

The Number of hubs.

Definition at line 58 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.3 int AlpsKnowledgeBrokerMPI::globalRank_ [protected]

The rank of the process in MPI_COMM_WORLD.

Definition at line 61 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.4 MPI_Comm AlpsKnowledgeBrokerMPI::clusterComm_ [protected]

Communicator of the cluster to which the process belongs.

Definition at line 64 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.5 MPI_Comm AlpsKnowledgeBrokerMPI::hubComm_ [protected]

Communicator consists of all hubs.

Definition at line 67 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.6 MPI_Group AlpsKnowledgeBrokerMPI::hubGroup_ [protected]

MPI_Group consists of all hubs.

Definition at line 70 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.7 int AlpsKnowledgeBrokerMPI::clusterSize_ [protected]

The actual size of the cluster to which the process belongs.

Definition at line 73 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.8 int AlpsKnowledgeBrokerMPI::userClusterSize_ [protected]

The user reqested size of a cluster.

Definition at line 76 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.9 int AlpsKnowledgeBrokerMPI::clusterRank_ [protected]

The local rank of the process in clusterComm_.

Definition at line 79 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.10 int* AlpsKnowledgeBrokerMPI::hubRanks_ [protected]

The global ranks of the hubs.

Definition at line 82 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.11 int AlpsKnowledgeBrokerMPI::myHubRank_ [protected]

The global rank of its hub for a worker.

Definition at line 85 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.12 int AlpsKnowledgeBrokerMPI::masterRank_ [protected]

The global rank of the master.

Definition at line 88 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.13 AlpsProcessType AlpsKnowledgeBrokerMPI::processType_ [protected]

The AlpsProcessType of this process.

Definition at line 91 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.14 AlpsProcessType* AlpsKnowledgeBrokerMPI::processTypeList_ [protected]

The AlpsProcessType of all process.

Definition at line 94 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.15 bool AlpsKnowledgeBrokerMPI::hubWork_ [protected]

Whether hub should also work as a worker.

Definition at line 97 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.16 MPI_Request AlpsKnowledgeBrokerMPI::subTreeRequest_ [protected]

Send subtree request.

Definition at line 100 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.17 MPI_Request AlpsKnowledgeBrokerMPI::solRequestL_ [protected]

Send model knoledge request.

Definition at line 103 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.18 MPI_Request AlpsKnowledgeBrokerMPI::modelKnowRequestL_ [protected]

Send model knoledge request.

Definition at line 107 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.19 MPI_Request AlpsKnowledgeBrokerMPI::forwardRequestL_ [protected]

Forward model knoledge request.

Definition at line 111 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.20 double AlpsKnowledgeBrokerMPI::incumbentValue_ [protected]

Incumbent value.

Definition at line 120 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.21 int AlpsKnowledgeBrokerMPI::incumbentID_ [protected]

The process id that store the incumbent.

Definition at line 123 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.22 bool AlpsKnowledgeBrokerMPI::updateIncumbent_ [protected]

Indicate whether the incumbent value is updated between two checking point.

Definition at line 127 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.23 double AlpsKnowledgeBrokerMPI::workQuality_ [protected]

The workload quality of the process.

Definition at line 135 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.24 double AlpsKnowledgeBrokerMPI::clusterWorkQuality_ [protected]

The workload quality of the cluster to which the process belong.

Definition at line 138 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.25 double AlpsKnowledgeBrokerMPI::systemWorkQuality_ [protected]

The workload quality of the whole system.

Definition at line 141 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.26 double* AlpsKnowledgeBrokerMPI::hubWorkQualities_ [protected]

The workload qualities of hubs.

Definition at line 144 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.27 double* AlpsKnowledgeBrokerMPI::workerWorkQualities_ [protected]

The workload qualities of workers in the cluster to which this proces belongs.

Number of nodes is used as the quantities criteria.

Definition at line 148 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.28 double AlpsKnowledgeBrokerMPI::workQuantity_ [protected]

The workload quantity of the workload on the process.

Definition at line 151 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.29 double AlpsKnowledgeBrokerMPI::clusterWorkQuantity_ [protected]

The workload quantity of the cluster to which the process belongs.

Definition at line 154 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.30 double AlpsKnowledgeBrokerMPI::systemWorkQuantity_ [protected]

The workload quantity of the whole system.

Definition at line 157 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.31 double AlpsKnowledgeBrokerMPI::systemWorkQuantityForce_ [protected]

The workload quantity of the whole system before forcing termination.

Definition at line 160 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.32 double* AlpsKnowledgeBrokerMPI::hubWorkQuantities_ [protected]

The workload quantities of all clusters/hubs.

Definition at line 163 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.33 double* AlpsKnowledgeBrokerMPI::workerWorkQuantities_ [protected]

The workload quantities of workers in the cluster to which this proces belongs.

Definition at line 167 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.34 bool* AlpsKnowledgeBrokerMPI::workerReported_ [protected]

Indicate which worker has been reported its work.

Definition at line 170 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.35 bool* AlpsKnowledgeBrokerMPI::hubReported_ [protected]

Indicate which hub has been reported its work.

Definition at line 173 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.36 bool AlpsKnowledgeBrokerMPI::allHubReported_ [protected]

Indicate whether all hubs have reported status to master at least once.

Definition at line 176 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.37 int AlpsKnowledgeBrokerMPI::masterDoBalance_ [protected]

Whether master do load balance.

0: do; >0: blocked.

Definition at line 179 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.38 int AlpsKnowledgeBrokerMPI::hubDoBalance_ [protected]

Whether a hub do load balance.

0: do; >0: blocked.

Definition at line 182 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.39 int* AlpsKnowledgeBrokerMPI::workerNodeProcesseds_ [protected]

To record how many nodes processed for each worker in a cluster.

Definition at line 185 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.40 int AlpsKnowledgeBrokerMPI::clusterNodeProcessed_ [protected]

To record how many nodes by a cluster.

Definition at line 188 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.41 int AlpsKnowledgeBrokerMPI::sendCount_ [protected]

The number of new messages sent by the process after last survey.

Definition at line 199 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.42 int AlpsKnowledgeBrokerMPI::recvCount_ [protected]

The number of new messages received by the process after last survey.

Definition at line 202 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.43 int AlpsKnowledgeBrokerMPI::clusterSendCount_ [protected]

The number of new messages sent by the processes in clusterComm_ after last survey.

Definition at line 206 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.44 int AlpsKnowledgeBrokerMPI::clusterRecvCount_ [protected]

The number of new messages received by the processes in clusterComm_ after last survey.

Definition at line 210 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.45 int AlpsKnowledgeBrokerMPI::systemSendCount_ [protected]

The total number of messages sent by the all processes.

Definition at line 213 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.46 int AlpsKnowledgeBrokerMPI::systemRecvCount_ [protected]

The total number of messages sent by the all processes.

Definition at line 216 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.47 double AlpsKnowledgeBrokerMPI::rampUpTime_ [protected]

The time spent in ramp up.

Definition at line 240 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.48 double AlpsKnowledgeBrokerMPI::rampDownTime_ [protected]

The time spent in ramp down.

Definition at line 243 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.49 double AlpsKnowledgeBrokerMPI::idleTime_ [protected]

The time spent waiting for work.

Definition at line 246 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.50 double AlpsKnowledgeBrokerMPI::msgTime_ [protected]

The time spent processing messages (include idle).

Definition at line 249 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.51 bool AlpsKnowledgeBrokerMPI::forceTerminate_ [protected]

Terminate due to reaching limits (time and node) or other reason.

Definition at line 256 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.52 char* AlpsKnowledgeBrokerMPI::attachBuffer_ [protected]

Buffer attached to MPI when sharing generated knowledge.

Definition at line 271 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.53 char* AlpsKnowledgeBrokerMPI::largeBuffer_ [protected]

Large message buffer.

Definition at line 274 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.54 char* AlpsKnowledgeBrokerMPI::largeBuffer2_ [protected]

Large message buffer.

Used for sharing model knowledge

Definition at line 277 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.55 char* AlpsKnowledgeBrokerMPI::smallBuffer_ [protected]

Small message buffer.

Definition at line 280 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.56 double AlpsKnowledgeBrokerMPI::masterBalancePeriod_ [protected]

The period that master do load balancing.

It changes as search progresses.

Definition at line 284 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.57 double AlpsKnowledgeBrokerMPI::hubReportPeriod_ [protected]

The period that a hub load balancing and report cluster status.

It changes as search progresses.

Definition at line 288 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.58 int AlpsKnowledgeBrokerMPI::modelGenID_ [protected]

The global rank of the process that share generated model knowledge.

Definition at line 291 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.59 int AlpsKnowledgeBrokerMPI::modelGenPos_ [protected]

Size of the shared knowledge.

Definition at line 294 of file AlpsKnowledgeBrokerMPI.h.

3.5.4.60 AlpsSubTree* AlpsKnowledgeBrokerMPI::rampUpSubTree_ [protected]

A subtree used in during up.

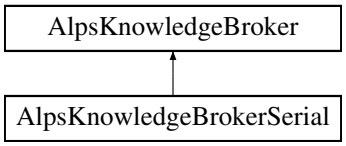
Definition at line 297 of file AlpsKnowledgeBrokerMPI.h.

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

- AlpsKnowledgeBrokerMPI.h

3.6 AlpsKnowledgeBrokerSerial Class Reference

Inheritance diagram for AlpsKnowledgeBrokerSerial:



Public Member Functions

- AlpsKnowledgeBrokerSerial ()
  Default constructor.
- AlpsKnowledgeBrokerSerial (AlpsModel &model)
  Useful constructor.
- AlpsKnowledgeBrokerSerial (int argc, char *argv[], AlpsModel &model)
  Userful constructor.

- virtual ~AlpsKnowledgeBrokerSerial ()

  Destructor.
- virtual void initializeSearch (int argc, char *argv[], AlpsModel &model)

  Reading in Alps and user parameter sets, and read in model data.
- virtual void rootSearch (AlpsTreeNode *root)

  Search for best solution.

Report the search results.

- virtual void searchLog ()

  Search log.
- virtual double getIncumbentValue () const

  The process queries the quality of the incumbent that it stores.
- virtual double getBestQuality () const

  The process queries the quality of the best solution that it finds.
- virtual void printBestSolution (char *outputFile=0) const

  The process outputs the best solution and the quality that it finds to a file or std::out.

Additional Inherited Members

3.6.1 Detailed Description

Definition at line 35 of file AlpsKnowledgeBrokerSerial.h.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 AlpsKnowledgeBrokerSerial::AlpsKnowledgeBrokerSerial ( ) [inline]

Default constructor.

Definition at line 42 of file AlpsKnowledgeBrokerSerial.h.

3.6.2.2 AlpsKnowledgeBrokerSerial::AlpsKnowledgeBrokerSerial (AlpsModel & *model*) [inline]

Useful constructor.

Note need read in parameters and data seperately.

Definition at line 49 of file AlpsKnowledgeBrokerSerial.h.

3.6.2.3 AlpsKnowledgeBrokerSerial::AlpsKnowledgeBrokerSerial (int *argc*, char * *argv[]*, AlpsModel & *model*) [inline]

Userful constructor.

Read in parameters from arguments. Also read in data.

Definition at line 58 of file AlpsKnowledgeBrokerSerial.h.

3.6.2.4 virtual AlpsKnowledgeBrokerSerial::~AlpsKnowledgeBrokerSerial ( ) [inline], [virtual]

Destructor.

Definition at line 69 of file AlpsKnowledgeBrokerSerial.h.

3.6.3 Member Function Documentation

3.6.3.1 virtual void AlpsKnowledgeBrokerSerial::searchLog ( ) [virtual]

Search log.

Implements AlpsKnowledgeBroker.

3.6.3.2 virtual double AlpsKnowledgeBrokerSerial::getIncumbentValue ( ) const [inline], [virtual]

The process queries the quality of the incumbent that it stores.

Implements AlpsKnowledgeBroker.

Definition at line 81 of file AlpsKnowledgeBrokerSerial.h.

3.6.3.3 virtual double AlpsKnowledgeBrokerSerial::getBestQuality ( ) const [inline], [virtual]

The process queries the quality of the best solution that it finds.

Implements AlpsKnowledgeBroker.

Definition at line 87 of file AlpsKnowledgeBrokerSerial.h.

3.6.3.4 virtual void AlpsKnowledgeBrokerSerial::printBestSolution (char * *outputFile* = 0) const [inline], [virtual]

The process outputs the best solution and the quality that it finds to a file or std::out.

Implements AlpsKnowledgeBroker.

Definition at line 98 of file AlpsKnowledgeBrokerSerial.h.

3.6.3.5 virtual void AlpsKnowledgeBrokerSerial::initializeSearch (int *argc*, char * *argv[]*, AlpsModel & *model*) [virtual]

Reading in Alps and user parameter sets, and read in model data.

Implements AlpsKnowledgeBroker.

3.6.3.6 virtual void AlpsKnowledgeBrokerSerial::rootSearch (AlpsTreeNode * *root*) [virtual]

Search for best solution.

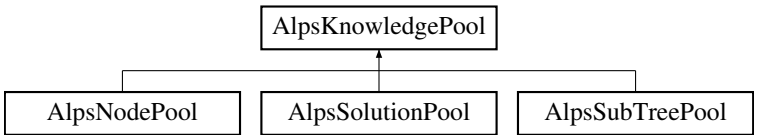
Implements AlpsKnowledgeBroker.

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

- AlpsKnowledgeBrokerSerial.h

3.7 AlpsKnowledgePool Class Reference

Inheritance diagram for AlpsKnowledgePool:



Public Member Functions

- virtual void addKnowledge (AlpsKnowledge *nk, double priority)=0

  Add a knowledge to pool.
- virtual int getNumKnowledges () const =0

  Query how many knowledges are in the pool.
- virtual std::pair < AlpsKnowledge *, double > getKnowledge () const =0

  Query a knowledge, but doesn't remove it from the pool.
- virtual void popKnowledge ()

  Remove the queried knowledge from the pool.
- virtual bool hasKnowledge () const

  Check whether the pool has knowledge.
- virtual void setMaxNumKnowledges (int num)

  Set the quantity limit of knowledges that can be stored in the pool.
- virtual int getMaxNumKnowledges () const

  Query the quantity limit of knowledges.
- virtual std::pair < AlpsKnowledge *, double > getBestKnowledge () const

  Query the best knowledge in the pool.
- virtual void getAllKnowledges (std::vector< std::pair< AlpsKnowledge *, double > > &kls) const

  Get a reference to all the knowledges in the pool.

3.7.1 Detailed Description

Definition at line 36 of file AlpsKnowledgePool.h.

3.7.2 Member Function Documentation

3.7.2.1 virtual int AlpsKnowledgePool::getNumKnowledges ( ) const [pure virtual]

Query how many knowledges are in the pool.

Implemented in AlpsSolutionPool, AlpsNodePool, and AlpsSubTreePool.

3.7.2.2 virtual bool AlpsKnowledgePool::hasKnowledge ( ) const [inline], [virtual]

Check whether the pool has knowledge.

Reimplemented in AlpsNodePool, AlpsSolutionPool, and AlpsSubTreePool.

Definition at line 61 of file AlpsKnowledgePool.h.

3.7.2.3 virtual void AlpsKnowledgePool::setMaxNumKnowledges (int *num*) [inline], [virtual]

Set the quantity limit of knowledges that can be stored in the pool.

Reimplemented in AlpsSolutionPool.

Definition at line 67 of file AlpsKnowledgePool.h.

3.7.2.4 virtual int AlpsKnowledgePool::getMaxNumKnowledges ( ) const [inline], [virtual]

Query the quantity limit of knowledges.

Reimplemented in AlpsSolutionPool.

Definition at line 75 of file AlpsKnowledgePool.h.

3.7.2.5 virtual std::pair<AlpsKnowledge*, double> AlpsKnowledgePool::getBestKnowledge ( ) const [inline], [virtual]

Query the best knowledge in the pool.

Reimplemented in AlpsSolutionPool.

Definition at line 83 of file AlpsKnowledgePool.h.

3.7.2.6 virtual void AlpsKnowledgePool::getAllKnowledges (std::vector< std::pair< AlpsKnowledge *, double > > & *kls*) const [inline], [virtual]

Get a reference to all the knowledges in the pool.

Reimplemented in AlpsSolutionPool.

Definition at line 89 of file AlpsKnowledgePool.h.

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

- AlpsKnowledgePool.h

3.8 AlpsMessage Class Reference

Public Member Functions

Constructors etc

- **AlpsMessage** (Language language=us_en)

3.8.1 Detailed Description

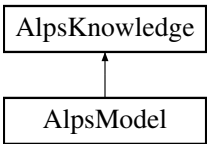
Definition at line 116 of file AlpsMessage.h.

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

- AlpsMessage.h

3.9 AlpsModel Class Reference

Inheritance diagram for AlpsModel:



Public Member Functions

- AlpsModel ()

  Default construtor.
- virtual ~AlpsModel ()

  Destructor.
- AlpsKnowledgeBroker * getKnowledgeBroker ()

  Get knowledge broker.
- void setKnowledgeBroker (AlpsKnowledgeBroker *b)

  Set knowledge broker.
- std::string getDataFile () const

  Get the input file.
- void setDataFile (std::string infile)

  Set the data file.
- AlpsParams * AlpsPar ()

  Access Alps Parameters.
- virtual void readInstance (const char *dateFile)

  Read in the instance data.
- virtual void readParameters (const int argnum, const char *const *arglist)

  Read in Alps parameters.
- void writeParameters (std::ostream &outstream) const

  Write out parameters.
- virtual bool setupSelf ()

  Do necessary work to make model ready for use, such as classify variable and constraint types.
- virtual void preprocess ()

  Preprocessing the model.
- virtual void postprocess ()

  Postprocessing results.
- virtual AlpsTreeNode * createRoot ()

  Create the root node.
- virtual void modelLog ()

  Problem specific log.
- virtual void nodeLog (AlpsTreeNode *node, bool force)

  Node log.
- virtual bool fathomAllNodes ()

  Return true if all nodes on this process can be fathomed.
- AlpsReturnStatus encodeAlps (AlpsEncoded *encoded) const

  Pack Alps portion of node into an encoded object.
- AlpsReturnStatus decodeAlps (AlpsEncoded &encoded)

  Unpack Alps portion of node from an encoded object.
- virtual void decodeToSelf (AlpsEncoded &encoded)

  Decode model data from the encoded form and fill member data.
- virtual void registerKnowledge ()

  Register knowledge class.
- virtual void sendGeneratedKnowledge ()

  Send generated knowledge.
- virtual void receiveGeneratedKnowledge ()

Receive generated knowledge.

- virtual AlpsEncoded * packSharedKnowlege ()

  Pack knowledge to be shared with others into an encoded object.

- virtual void unpackSharedKnowledge (AlpsEncoded &)

  Unpack and store shared knowledge from an encoded object.

Protected Attributes

- AlpsKnowledgeBroker * broker_

  Knowledge broker.

- std::string dataFile_

  Data file.

- AlpsParams * AlpsPar_

  The parameter set that is used in Alps.

3.9.1 Detailed Description

Definition at line 36 of file AlpsModel.h.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 AlpsModel::AlpsModel ( ) [inline]

Default construtor.

Definition at line 57 of file AlpsModel.h.

3.9.2.2 virtual AlpsModel::~AlpsModel ( ) [inline], [virtual]

Destructor.

Definition at line 62 of file AlpsModel.h.

3.9.3 Member Function Documentation

3.9.3.1 AlpsKnowledgeBroker* AlpsModel::getKnowledgeBroker ( ) [inline]

Get knowledge broker.

Definition at line 65 of file AlpsModel.h.

3.9.3.2 void AlpsModel::setKnowledgeBroker (AlpsKnowledgeBroker * *b*) [inline]

Set knowledge broker.

Definition at line 68 of file AlpsModel.h.

3.9.3.3 std::string AlpsModel::getDataFile ( ) const [inline]

Get the input file.

Definition at line 71 of file AlpsModel.h.

3.9.3.4 void AlpsModel::setDataFile (std::string *infile*) [inline]

Set the data file.

Definition at line 74 of file AlpsModel.h.

3.9.3.5 AlpsParams* AlpsModel::AlpsPar ( ) [inline]

Access Alps Parameters.

Definition at line 77 of file AlpsModel.h.

3.9.3.6 virtual void AlpsModel::readInstance (const char * *dateFile*) [inline], [virtual]

Read in the instance data.

At Alps level, nothing to do.

Definition at line 80 of file AlpsModel.h.

3.9.3.7 virtual void AlpsModel::readParameters (const int *argnum,* const char *const * *arglist*) [virtual]

Read in Alps parameters.

3.9.3.8 void AlpsModel::writeParameters (std::ostream & *outstream*) const

Write out parameters.

3.9.3.9 virtual bool AlpsModel::setupSelf ( ) [inline], [virtual]

Do necessary work to make model ready for use, such as classify variable and constraint types.

Definition at line 93 of file AlpsModel.h.

3.9.3.10 virtual void AlpsModel::preprocess ( ) [inline], [virtual]

Preprocessing the model.

Definition at line 96 of file AlpsModel.h.

3.9.3.11 virtual void AlpsModel::postprocess ( ) [inline], [virtual]

Postprocessing results.

Definition at line 99 of file AlpsModel.h.

3.9.3.12 virtual AlpsTreeNode* AlpsModel::createRoot ( ) [inline], [virtual]

Create the root node.

Default: do nothing

Definition at line 102 of file AlpsModel.h.

3.9.3.13 virtual void AlpsModel::modelLog ( ) [inline], [virtual]

Problem specific log.

Definition at line 108 of file AlpsModel.h.

3.9.3.14 virtual void AlpsModel::nodeLog (AlpsTreeNode * *node,* bool *force*) [virtual]

Node log.

3.9.3.15 virtual bool AlpsModel::fathomAllNodes ( ) [inline], [virtual]

Return true if all nodes on this process can be fathomed.

Definition at line 114 of file AlpsModel.h.

3.9.3.16 AlpsReturnStatus AlpsModel::encodeAlps (AlpsEncoded * *encoded*) const

Pack Alps portion of node into an encoded object.

3.9.3.17 AlpsReturnStatus AlpsModel::decodeAlps (AlpsEncoded & *encoded*)

Unpack Alps portion of node from an encoded object.

3.9.3.18 virtual void AlpsModel::decodeToSelf (AlpsEncoded & *encoded*) [inline], [virtual]

Decode model data from the encoded form and fill member data.

Definition at line 127 of file AlpsModel.h.

3.9.3.19 virtual void AlpsModel::registerKnowledge ( ) [inline], [virtual]

Register knowledge class.

Definition at line 130 of file AlpsModel.h.

3.9.3.20 virtual AlpsEncoded* AlpsModel::packSharedKnowlege ( ) [inline], [virtual]

Pack knowledge to be shared with others into an encoded object.

Return NULL means that no knowledge can be shared.

Definition at line 140 of file AlpsModel.h.

3.9.3.21 virtual void AlpsModel::unpackSharedKnowledge (AlpsEncoded &) [inline], [virtual]

Unpack and store shared knowledge from an encoded object.

Definition at line 147 of file AlpsModel.h.

3.9.4 Member Data Documentation

3.9.4.1 AlpsKnowledgeBroker* AlpsModel::broker_ [protected]

Knowledge broker.

Definition at line 46 of file AlpsModel.h.

3.9.4.2 std::string AlpsModel::dataFile_ [protected]

Data file.

Definition at line 49 of file AlpsModel.h.

3.9.4.3 AlpsParams* AlpsModel::AlpsPar_ [protected]

The parameter set that is used in Alps.

Definition at line 52 of file AlpsModel.h.

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

- AlpsModel.h

3.10 AlpsNodeDesc Class Reference

A class to refer to the description of a search tree node.

`#include <AlpsNodeDesc.h>`

Public Member Functions

- virtual AlpsReturnStatus encode (AlpsEncoded *encoded) const
  Pack node description into an encoded.
- virtual AlpsReturnStatus decode (AlpsEncoded &encoded)
  Unpack a node description from an encoded.

Protected Attributes

- AlpsModel * model_
  A pointer to model.

3.10.1 Detailed Description

A class to refer to the description of a search tree node.

FIXME : write a better doc...

Definition at line 35 of file AlpsNodeDesc.h.

3.10.2 Member Function Documentation

3.10.2.1 virtual AlpsReturnStatus AlpsNodeDesc::encode (AlpsEncoded * *encoded*) const [inline], [virtual]

Pack node description into an encoded.

Definition at line 55 of file AlpsNodeDesc.h.

3.10.2.2 virtual AlpsReturnStatus AlpsNodeDesc::decode (AlpsEncoded & *encoded*) [inline], [virtual]

Unpack a node description from an encoded.

Fill member data.

Definition at line 63 of file AlpsNodeDesc.h.

3.10.3 Member Data Documentation

3.10.3.1 AlpsModel* AlpsNodeDesc::model_ [protected]

A pointer to model.

Definition at line 41 of file AlpsNodeDesc.h.

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

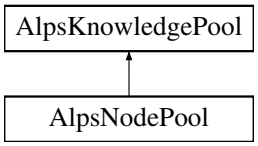
- AlpsNodeDesc.h

3.11 AlpsNodePool Class Reference

Node pool is used to store the nodes to be processed.

`#include <AlpsNodePool.h>`

Inheritance diagram for AlpsNodePool:



Public Member Functions

- int getNumKnowledges () const
  Query the number of nodes in the node pool.
- double getBestKnowledgeValue () const
  Get the "best value" of the nodes in node pool.
- AlpsTreeNode * getBestNode () const
  Get the "best" nodes in node pool.
- bool hasKnowledge () const
  Check whether there are still nodes in the node pool.
- std::pair< AlpsKnowledge *, double > getKnowledge () const
  Get the node with highest priority.
- void popKnowledge ()
  Remove the node with highest priority from the pool.
- void addKnowledge (AlpsKnowledge *node, double priority)
  Remove the node with highest priority from the pool and the elite list.
- const AlpsPriorityQueue < AlpsTreeNode * > & getCandidateList () const
  Get a constant reference to the priority queue that stores nodes.
- void setNodeSelection (AlpsSearchStrategy< AlpsTreeNode * > &compare)
  Set strategy and resort heap.
- void deleteGuts ()
  Delete all the nodes in the pool and free memory.
- void clear ()
  Remove all the nodes in the pool (does not free memory).

3.11.1 Detailed Description

Node pool is used to store the nodes to be processed.

Definition at line 37 of file AlpsNodePool.h.

3.11.2 Member Function Documentation

3.11.2.1 int AlpsNodePool::getNumKnowledges ( ) const [inline], [virtual]

Query the number of nodes in the node pool.

Implements AlpsKnowledgePool.

Definition at line 55 of file AlpsNodePool.h.

3.11.2.2 double AlpsNodePool::getBestKnowledgeValue ( ) const [inline]

Get the "best value" of the nodes in node pool.

Definition at line 58 of file AlpsNodePool.h.

3.11.2.3 AlpsTreeNode* AlpsNodePool::getBestNode ( ) const [inline]

Get the "best" nodes in node pool.

Definition at line 74 of file AlpsNodePool.h.

3.11.2.4 bool AlpsNodePool::hasKnowledge ( ) const [inline], [virtual]

Check whether there are still nodes in the node pool.

Reimplemented from AlpsKnowledgePool.

Definition at line 93 of file AlpsNodePool.h.

3.11.2.5 std::pair<AlpsKnowledge*, double> AlpsNodePool::getKnowledge ( ) const [inline], [virtual]

Get the node with highest priority.

Doesn't remove it from the pool

Implements AlpsKnowledgePool.

Definition at line 96 of file AlpsNodePool.h.

3.11.2.6 void AlpsNodePool::addKnowledge (AlpsKnowledge * *node*, double *priority*) [inline], [virtual]

Remove the node with highest priority from the pool and the elite list.

Add a node to node pool.

Implements AlpsKnowledgePool.

Definition at line 110 of file AlpsNodePool.h.

3.11.2.7 const AlpsPriorityQueue<AlpsTreeNode*>& AlpsNodePool::getCandidateList ( ) const [inline]

Get a constant reference to the priority queue that stores nodes.

Definition at line 124 of file AlpsNodePool.h.

3.11.2.8 void AlpsNodePool::setNodeSelection (AlpsSearchStrategy< AlpsTreeNode * > & *compare*) [inline]

Set strategy and resort heap.

Definition at line 127 of file AlpsNodePool.h.

3.11.2.9 void AlpsNodePool::deleteGuts ( ) [inline]

Delete all the nodes in the pool and free memory.

Definition at line 132 of file AlpsNodePool.h.

3.11.2.10 void AlpsNodePool::clear ( ) [inline]

Remove all the nodes in the pool (does not free memory).

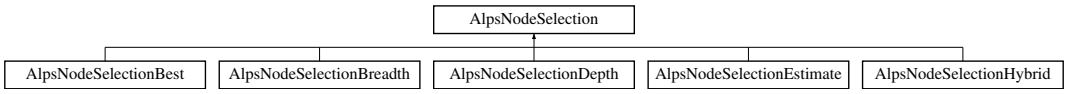
Definition at line 142 of file AlpsNodePool.h.

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

- AlpsNodePool.h

3.12 AlpsNodeSelection Class Reference

Inheritance diagram for AlpsNodeSelection:



Public Member Functions

- AlpsNodeSelection ()
  Default Constructor.
- virtual ~AlpsNodeSelection ()
  Default Destructor.
- virtual bool compare (AlpsTreeNode *x, AlpsTreeNode *y)=0
  This returns true if the depth of node y is lesser than that of node x.

3.12.1 Detailed Description

Definition at line 49 of file AlpsSearchStrategy.h.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 AlpsNodeSelection::AlpsNodeSelection ( ) [inline]

Default Constructor.

Definition at line 53 of file AlpsSearchStrategy.h.

3.12.2.2 virtual AlpsNodeSelection::~AlpsNodeSelection ( ) [inline],[virtual]

Default Destructor.

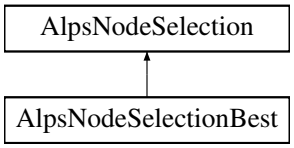
Definition at line 56 of file AlpsSearchStrategy.h.

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

- AlpsSearchStrategy.h

3.13 AlpsNodeSelectionBest Class Reference

Inheritance diagram for AlpsNodeSelectionBest:



Public Member Functions

- AlpsNodeSelectionBest ()
  Default Constructor.
- virtual ~AlpsNodeSelectionBest ()
  Default Destructor.
- virtual bool compare (AlpsTreeNode *x, AlpsTreeNode *y)
  This returns true if quality of node y is better (the less the better) than that of node x.

3.13.1 Detailed Description

Definition at line 139 of file AlpsSearchStrategy.h.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 AlpsNodeSelectionBest::AlpsNodeSelectionBest ( ) [inline]

Default Constructor.

Definition at line 143 of file AlpsSearchStrategy.h.

3.13.2.2 virtual AlpsNodeSelectionBest::~AlpsNodeSelectionBest ( ) [inline],[virtual]

Default Destructor.

Definition at line 146 of file AlpsSearchStrategy.h.

3.13.3 Member Function Documentation

3.13.3.1 virtual bool AlpsNodeSelectionBest::compare (AlpsTreeNode * x, AlpsTreeNode * y) [inline], [virtual]

This returns true if quality of node y is better (the less the better) than that of node x.

Implements AlpsNodeSelection.

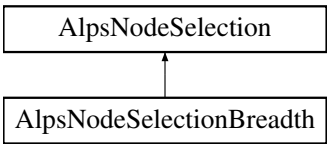
Definition at line 150 of file AlpsSearchStrategy.h.

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

- AlpsSearchStrategy.h

3.14 AlpsNodeSelectionBreadth Class Reference

Inheritance diagram for AlpsNodeSelectionBreadth:



Public Member Functions

- AlpsNodeSelectionBreadth ()
  Default Constructor.
- virtual ~AlpsNodeSelectionBreadth ()
  Default Destructor.
- virtual bool compare (AlpsTreeNode *x, AlpsTreeNode *y)
  This returns true if the depth of node y is lesser than that of node x.

3.14.1 Detailed Description

Definition at line 157 of file AlpsSearchStrategy.h.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 AlpsNodeSelectionBreadth::AlpsNodeSelectionBreadth ( ) [inline]

Default Constructor.

Definition at line 161 of file AlpsSearchStrategy.h.

3.14.2.2 virtual AlpsNodeSelectionBreadth::~AlpsNodeSelectionBreadth ( ) [inline], [virtual]

Default Destructor.

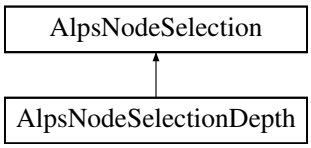
Definition at line 164 of file AlpsSearchStrategy.h.

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

- AlpsSearchStrategy.h

3.15 AlpsNodeSelectionDepth Class Reference

Inheritance diagram for AlpsNodeSelectionDepth:



Public Member Functions

- AlpsNodeSelectionDepth ()

  Default Constructor.
- virtual ∼AlpsNodeSelectionDepth ()

  Default Destructor.
- virtual bool compare (AlpsTreeNode ∗x, AlpsTreeNode ∗y)

  This returns true if the depth of node y is greater than that of node x.

3.15.1 Detailed Description

Definition at line 175 of file AlpsSearchStrategy.h.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 AlpsNodeSelectionDepth::AlpsNodeSelectionDepth ( ) [inline]

Default Constructor.

Definition at line 179 of file AlpsSearchStrategy.h.

3.15.2.2 virtual AlpsNodeSelectionDepth::∼AlpsNodeSelectionDepth ( ) [inline], [virtual]

Default Destructor.

Definition at line 182 of file AlpsSearchStrategy.h.

3.15.3 Member Function Documentation

3.15.3.1 virtual bool AlpsNodeSelectionDepth::compare (AlpsTreeNode ∗ *x*, AlpsTreeNode ∗ *y*) [inline], [virtual]

This returns true if the depth of node y is greater than that of node x.

Implements AlpsNodeSelection.

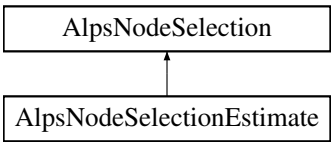
Definition at line 186 of file AlpsSearchStrategy.h.

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

- AlpsSearchStrategy.h

3.16 AlpsNodeSelectionEstimate Class Reference

Inheritance diagram for AlpsNodeSelectionEstimate:



Public Member Functions

- AlpsNodeSelectionEstimate ()

  Default Constructor.
- virtual ~AlpsNodeSelectionEstimate ()

  Default Destructor.
- virtual bool compare (AlpsTreeNode *x, AlpsTreeNode *y)

  This returns true if the estimate quality of node y is better (the lesser the better) than that of node x.

3.16.1 Detailed Description

Definition at line 193 of file AlpsSearchStrategy.h.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 AlpsNodeSelectionEstimate::AlpsNodeSelectionEstimate ( ) [inline]

Default Constructor.

Definition at line 197 of file AlpsSearchStrategy.h.

3.16.2.2 virtual AlpsNodeSelectionEstimate::~AlpsNodeSelectionEstimate ( ) [inline], [virtual]

Default Destructor.

Definition at line 200 of file AlpsSearchStrategy.h.

3.16.3 Member Function Documentation

3.16.3.1 virtual bool AlpsNodeSelectionEstimate::compare (AlpsTreeNode * *x*, AlpsTreeNode * *y*) [inline], [virtual]

This returns true if the estimate quality of node y is better (the lesser the better) than that of node x.

Implements AlpsNodeSelection.

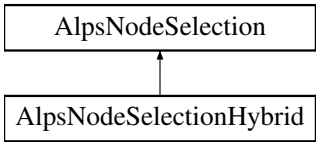
Definition at line 204 of file AlpsSearchStrategy.h.

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

- AlpsSearchStrategy.h

3.17 AlpsNodeSelectionHybrid Class Reference

Inheritance diagram for AlpsNodeSelectionHybrid:



Public Member Functions

- AlpsNodeSelectionHybrid ()

  Default Constructor.
- virtual ~AlpsNodeSelectionHybrid ()

  Default Destructor.
- virtual bool compare (AlpsTreeNode *x, AlpsTreeNode *y)

  This returns true if the quality of node y is better (the lesser the better) than that of node x.

3.17.1 Detailed Description

Definition at line 211 of file AlpsSearchStrategy.h.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 AlpsNodeSelectionHybrid::AlpsNodeSelectionHybrid ( ) [inline]

Default Constructor.

Definition at line 215 of file AlpsSearchStrategy.h.

3.17.2.2 virtual AlpsNodeSelectionHybrid::~AlpsNodeSelectionHybrid ( ) [inline], [virtual]

Default Destructor.

Definition at line 218 of file AlpsSearchStrategy.h.

3.17.3 Member Function Documentation

3.17.3.1 virtual bool AlpsNodeSelectionHybrid::compare (AlpsTreeNode * x, AlpsTreeNode * y) [inline], [virtual]

This returns true if the quality of node y is better (the lesser the better) than that of node x.

Implements AlpsNodeSelection.

Definition at line 222 of file AlpsSearchStrategy.h.

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

- AlpsSearchStrategy.h

3.18 AlpsParameter Class Reference

This parameter indeintifies a single parameter entry.

`#include <AlpsParameterBase.h>`

Public Member Functions

Constructors / Destructor

- AlpsParameter ()
  The default constructor creates a phony parameter.
- AlpsParameter (const AlpsParameterT t, const int i)
  Constructor where members are specified.
- ~AlpsParameter ()
  The destructor.

Query methods

- AlpsParameterT type () const
  Return the type of the parameter.
- int index () const
  Return the index of the parameter within all parameters of the same type.

3.18.1 Detailed Description

This parameter indeintifies a single parameter entry.

Definition at line 77 of file AlpsParameterBase.h.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 AlpsParameter::AlpsParameter ( ) [inline]

The default constructor creates a phony parameter.

Definition at line 93 of file AlpsParameterBase.h.

3.18.2.2 AlpsParameter::AlpsParameter (const AlpsParameterT *t*, const int *i*) [inline]

Constructor where members are specified.

Definition at line 95 of file AlpsParameterBase.h.

3.18.2.3 AlpsParameter::~AlpsParameter ( ) [inline]

The destructor.

Definition at line 98 of file AlpsParameterBase.h.

3.18.3 Member Function Documentation

3.18.3.1 AlpsParameterT AlpsParameter::type ( ) const [inline]

Return the type of the parameter.

Definition at line 104 of file AlpsParameterBase.h.

3.18.3.2 int AlpsParameter::index ( ) const [inline]

Return the index of the parameter within all parameters of the same type.

Definition at line 107 of file AlpsParameterBase.h.

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

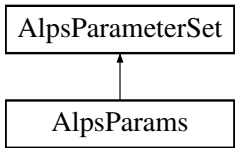
- AlpsParameterBase.h

3.19 AlpsParameterSet Class Reference

This is the class serves as a holder for a set of parameters.

`#include <AlpsParameterBase.h>`

Inheritance diagram for AlpsParameterSet:



Public Member Functions

- void setEntry (const AlpsParameter key, const char *val)
  First, there is the assignment operator that sets the whole parameter set at once.
- void readFromStream (std::istream &parstream)
  Read the parameters from the stream specified in the argument.
- void readFromFile (const char *paramfile)
  Read parameters from a file.
- void readFromArglist (const int argnum, const char *const *arglist)
  Read parameters from the command line.
- void writeToStream (std::ostream &outstream) const
  Write keyword-value pairs to the stream specified in the argument.
- AlpsParameterSet (int c, int i, int d, int s, int sa)
  The constructor allocate memory for parameters.
- virtual ~AlpsParameterSet ()
  The destructor deletes all data members.

Pure virtual functions that must be defined for each parameter set.

If the user creates a new parameter set, she must define these two methods for the class.

- virtual void createKeywordList ()=0
  Method for creating the list of keyword looked for in the parameter file.
- virtual void setDefaultEntries ()=0
  Method for setting the default values for the parameters.

Pack and unpack

- virtual void pack (AlpsEncoded &buf)

Pack the parameter set into the buffer.

- virtual void unpack (AlpsEncoded &buf)

  Unpack the parameter set from the buffer.

Protected Attributes

Data members. All of them are protected.

- std::vector< std::pair < std::string, AlpsParameter > > keys_

  The keyword, parameter pairs.
- std::vector< std::string > obsoleteKeys_

  list of obsolete keywords.
- bool * bpar_

  The bool parameters.
- int * ipar_

  The integer parameters.
- double * dpar_

  The double parameters.
- std::string * spar_

  The string (actually, std::string) parameters.
- int numSa_

  The "vector of string" parameters.
- std::vector< std::string > * **sapar_**

3.19.1 Detailed Description

This is the class serves as a holder for a set of parameters.

For example, Alps stores has a parameter set for each process. Of course, the user can use this class for her own parameters. To use this class the user must

- first derive a subclass with the names of the parameters (see, e.g., AlpsParams.)
- then define the member functions `createKeywordList()` and `setDefaultEntries()`. For an example look at the file `AlpsParams.cpp`. Essentially, the first method defines what keywords should be looked for in the parameter file, and if one is found which parameter should take the corresponding value; the other method specifies the default values for each parameter.

After this the user can read in the parameters from a file, she can set/access the parameters in the parameter set.

Definition at line 134 of file AlpsParameterBase.h.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 AlpsParameterSet::AlpsParameterSet (int *c*, int *i*, int *d*, int *s*, int *sa*) [inline]

The constructor allocate memory for parameters.

Definition at line 243 of file AlpsParameterBase.h.

3.19.2.2 virtual AlpsParameterSet::~AlpsParameterSet ( ) [inline], [virtual]

The destructor deletes all data members.

Definition at line 253 of file AlpsParameterBase.h.

3.19.3 Member Function Documentation

3.19.3.1 virtual void AlpsParameterSet::createKeywordList ( ) [pure virtual]

Method for creating the list of keyword looked for in the parameter file.

Implemented in AlpsParams.

3.19.3.2 virtual void AlpsParameterSet::setDefaultEntries ( ) [pure virtual]

Method for setting the default values for the parameters.

Implemented in AlpsParams.

3.19.3.3 virtual void AlpsParameterSet::pack (AlpsEncoded & *buf*) [inline], [virtual]

Pack the parameter set into the buffer.

Reimplemented in AlpsParams.

Definition at line 182 of file AlpsParameterBase.h.

3.19.3.4 virtual void AlpsParameterSet::unpack (AlpsEncoded & *buf*) [inline], [virtual]

Unpack the parameter set from the buffer.

Reimplemented in AlpsParams.

Definition at line 187 of file AlpsParameterBase.h.

3.19.3.5 void AlpsParameterSet::setEntry (const AlpsParameter *key,* const char * *val*) [inline]

First, there is the assignment operator that sets the whole parameter set at once.

Individual members of the parameter set can be set for using the overloaded setEntry() method. Using the example in the class documentation the user can set a parameter with the "<code>param.setEntry(USER_par::parameter_name, param_value)</code>" expression.

Definition at line 205 of file AlpsParameterBase.h.

3.19.3.6 void AlpsParameterSet::readFromStream (std::istream & *parstream*)

Read the parameters from the stream specified in the argument.

The stream is interpreted as a lines separated by newline characters. The first word on each line is tested for match with the keywords specified in the createKeywordList() method. If there is a match then the second word will be interpreted as the value for the corresponding parameter. Any further words on the line are discarded. Every non-matching line is discarded.

If the keyword corresponds to a non-array parameter then the new value simply overwrites the old one. Otherwise, i.e., if it is a StringArrayPar, the value is appended to the list of strings in that array.

3.19.3.7 void AlpsParameterSet::readFromFile (const char * *paramfile*)

Read parameters from a file.

3.19.3.8 void AlpsParameterSet::writeToStream (std::ostream & *outstream*) const

Write keyword-value pairs to the stream specified in the argument.

Each keyword-value pair is separated by a newline character.

3.19.4 Member Data Documentation

3.19.4.1 std::vector< std::pair<std::string, AlpsParameter> > AlpsParameterSet::keys_ [protected]

The keyword, parameter pairs.

Used when the parameter file is read in.

Definition at line 140 of file AlpsParameterBase.h.

3.19.4.2 std::vector<std::string> AlpsParameterSet::obsoleteKeys_ [protected]

list of obsolete keywords.

If any of these is encountered a warning is printed.

Definition at line 144 of file AlpsParameterBase.h.

3.19.4.3 bool* AlpsParameterSet::bpar_ [protected]

The bool parameters.

Definition at line 147 of file AlpsParameterBase.h.

3.19.4.4 int* AlpsParameterSet::ipar_ [protected]

The integer parameters.

Definition at line 150 of file AlpsParameterBase.h.

3.19.4.5 double* AlpsParameterSet::dpar_ [protected]

The double parameters.

Definition at line 153 of file AlpsParameterBase.h.

3.19.4.6 std::string* AlpsParameterSet::spar_ [protected]

The string (actually, std::string) parameters.

Definition at line 156 of file AlpsParameterBase.h.

3.19.4.7 int AlpsParameterSet::numSa_ [protected]

The "vector of string" parameters.

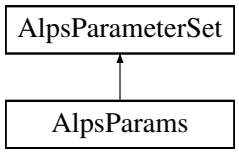
Definition at line 159 of file AlpsParameterBase.h.

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

- AlpsParameterBase.h

3.20 AlpsParams Class Reference

Inheritance diagram for AlpsParams:



Public Types

- enum boolParams {
  checkMemory, deleteDeadNode, interClusterBalance, intraClusterBalance,
  printSolution }

  Character parameters.
- enum intParams {
  bufSpare, clockType, eliteSize, hubInitNodeNum,
  hubMsgLevel, hubNum, largeSize, logFileLevel,
  masterInitNodeNum, masterReportInterval, hubWorkClusterSizeLimit, mediumSize,
  msgLevel, nodeLimit, nodeLogInterval, printSystemStatus,
  processNum, staticBalanceScheme, searchStrategy , smallSize,
  solLimit, unitWorkNodes, workerMsgLevel }

  Integer paramters.
- enum dblParams {
  changeWorkThreshold, donorThreshold, hubReportPeriod, masterBalancePeriod,
  needWorkThreshold, receiverThreshold, timeLimit, tolerance,
  unitWorkTime, zeroLoad }

  Double parameters.
- enum strParams { instance, logFile }

  String parameters.
- enum strArrayParams

  There are no string array parameters.

Public Member Functions

- virtual void createKeywordList ()

  Method for creating the list of keyword looked for in the parameter file.
- virtual void setDefaultEntries ()

  Method for setting the default values for the parameters.
- void setEntry (const boolParams key, const char *val)

  char is true(1) or false(0), not used*
- void setEntry (const boolParams key, const char val)

  char is true(1) or false(0), not used
- void setEntry (const boolParams key, const bool val)

  This method is the one that ever been used.

Constructors.

- AlpsParams ()

  The default constructor creates a parameter set with from the template argument structure.

Query methods

For user's application: Copy following code exactly (till the end of this class) and do NOT change anything.

The reason can not put following functions in base class `AlpsParameterSet` *is:*

`boolParams` *and* `endOfBoolParams` *etc. can NOT be declared in base class. They are different types for each derived classes. The members of the parameter set can be queried for using the overloaded entry() method. Using the example in the class documentation the user can get a parameter with the "<code>param.entry(USER_par::parameter_name)</code>" expression.*

- bool **entry** (const boolParams key) const
- int **entry** (const intParams key) const
- double **entry** (const dblParams key) const
- const std::string & **entry** (const strParams key) const
- const std::vector< std::string > & **entry** (const strArrayParams key) const

Packing/unpacking methods

- void pack (AlpsEncoded &buf)
  Pack the parameter set into buf.
- void unpack (AlpsEncoded &buf)
  Unpack the parameter set from buf.

Additional Inherited Members

3.20.1 Detailed Description

Definition at line 36 of file AlpsParams.h.

3.20.2 Member Enumeration Documentation

3.20.2.1 enum AlpsParams::boolParams

Character parameters.

All of these variable are used as booleans (ture = 1, false = 0).

Enumerator:

checkMemory Check memory. Default: false

deleteDeadNode Remove dead nodes or not. Default: true.

interClusterBalance Master balances the workload of hubs: centralized. Default: true.

intraClusterBalance Hub balances the workload of workers: receiver initialized. Default: true

printSolution Print solution to screen and log if have a solution and msgLevel and logFileLevel permits. Default: false.

Definition at line 40 of file AlpsParams.h.

3.20.2.2 enum AlpsParams::intParams

Integer paramters.

Enumerator:

bufSpare The size of extra memory allocated to a message buffer. Default: 256 byte

clockType Type of clock when timing rampup, rampdown, etc. CPU or Wallclock. default: wallclock

eliteSize Number of the "elite" nodes that are used in determining workload. Default: 1

hubInitNodeNum The number of nodes initially generated by each hub. Default: 2

hubMsgLevel Message level of the hub specific messages. (0: no print to screen; 1: summary; 2: moderate; 3: verbose) Default: 0

hubNum The number of hubs. Default: 1

largeSize The size of memory allocated for large size message. Default: 10485760

logFileLevel The level of log file. (0: no log file; 1: summary; 2: moderate; 3: verbose) Default: 0

masterInitNodeNum The number of nodes initially generated by the master. Default: 2

masterReportInterval The interval between master report system status. Default: 10

hubWorkClusterSizeLimit If the number of processes in a cluster is less than it, the hub also work as a worker. Default: 0 (Hub does NOT work)

mediumSize The size of memory allocated for medium size message. Default: 4096

msgLevel The level of printing messages on screen. Used to control master and general messages. (0: no print to screen; 1: summary; 2: moderate; 3: verbose) Default: 2

nodeLimit The max number of nodes can be processed. Default: ALPS_INT_MAX

nodeLogInterval Node log interval. Default: 100

printSystemStatus Print system status: 0: do not print, 1: print. Default: 1;

processNum The total number of processes that are launched for parallel code. Default: 2 Not used since can get actual number of processes from MPI.

staticBalanceScheme Static load balancing scheme – root initialization (0) – spiral (1)

searchStrategy Search strategy – best-first (0) – best-first-estimate (1) – breadth-first (2) – depth-first (3) – hybrid (4) Default: hybrid.

smallSize The size of memory allocated for small size message. Default: 1024

solLimit The max num of solution can be stored in a solution pool. Default: ALPS_INT_MAX

unitWorkNodes The size/number of nodes of a unit work. Default: 50

workerMsgLevel Message level of the worker specific messages. (0: no print to screen; 1: summary; 2: moderate; 3: verbose) Default: 0

Definition at line 63 of file AlpsParams.h.

3.20.2.3 enum AlpsParams::dblParams

Double parameters.

Enumerator:

changeWorkThreshold The threshold of workload below which a worker will change the subtree that is working on. Default: 0.05

donorThreshold It is between 1.0 - infty. When the workload in process is more than the average workload timing donorThreshold, it is a donor in load balancing. Defaut: 0.1

hubReportPeriod The time period (sec) for hubs to process messages. Default: 0.1

masterBalancePeriod The time period for master to do loading balance/termination check. Default: 0.05

needWorkThreshold The threshold of workload below which a process will ask for workload Default: 2.

receiverThreshold It is between 0.0 - 1.0. When the workload in process is less than the average workload timing receiverThreshold, it is a receiver. Default: 0.1

timeLimit The time limit (in seconds) of search. Default: ALPS_DBL_MAX

tolerance The numeric tolerance. Default: 1e-6

unitWorkTime The time length of a unit work. Default: 0.5

zeroLoad If less than this number, it is considered zero workload. Default: 1e-6

Definition at line 158 of file AlpsParams.h.

3.20.2.4 enum AlpsParams::strParams

String parameters.

Enumerator:

instance The instance to be solved. Default: "NONE"

logFile The name of log file. Default: "Alps.log "

Definition at line 199 of file AlpsParams.h.

3.20.2.5 enum AlpsParams::strArrayParams

There are no string array parameters.

Definition at line 212 of file AlpsParams.h.

3.20.3 Constructor & Destructor Documentation

3.20.3.1 AlpsParams::AlpsParams ( ) [inline]

The default constructor creates a parameter set with from the template argument structure.

The keyword list is created and the defaults are set.

Definition at line 227 of file AlpsParams.h.

3.20.4 Member Function Documentation

3.20.4.1 virtual void AlpsParams::createKeywordList ( ) [virtual]

Method for creating the list of keyword looked for in the parameter file.

Implements AlpsParameterSet.

3.20.4.2 virtual void AlpsParams::setDefaultEntries ( ) [virtual]

Method for setting the default values for the parameters.

Implements AlpsParameterSet.

3.20.4.3 void AlpsParams::pack (AlpsEncoded & *buf*) [inline], [virtual]

Pack the parameter set into buf.

Reimplemented from AlpsParameterSet.

Definition at line 341 of file AlpsParams.h.

3.20.4.4 void AlpsParams::unpack (AlpsEncoded & *buf*) [inline],[virtual]

Unpack the parameter set from buf.

Reimplemented from AlpsParameterSet.

Definition at line 355 of file AlpsParams.h.

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

- AlpsParams.h

3.21 AlpsPriorityQueue< T > Class Template Reference

Public Member Functions

- const std::vector< T > & getContainer () const
  Return a const reference to the container.
- void setComparison (AlpsSearchStrategy< T > &c)
  Set comparison function and resort heap.
- T top () const
  Return the top element of the heap.
- void push (T x)
  Add a element to the heap.
- void pop ()
  Remove the top element from the heap.
- bool empty () const
  Return true for an empty vector.
- size_t size () const
  Return the size of the vector.
- void clear ()
  Remove all elements from the vector.

3.21.1 Detailed Description

template<class T>class AlpsPriorityQueue< T >

Definition at line 34 of file AlpsPriorityQueue.h.

3.21.2 Member Function Documentation

3.21.2.1 template<class T> const std::vector<T>& AlpsPriorityQueue< T >::getContainer ( ) const [inline]

Return a const reference to the container.

Definition at line 50 of file AlpsPriorityQueue.h.

3.21.2.2 template<class T> void AlpsPriorityQueue< T >::setComparison (AlpsSearchStrategy< T > & *c*) [inline]

Set comparison function and resort heap.

Definition at line 53 of file AlpsPriorityQueue.h.

3.21.2.3 template<class T> T AlpsPriorityQueue< T >::top ( ) const [inline]

Return the top element of the heap.

Definition at line 59 of file AlpsPriorityQueue.h.

3.21.2.4 template<class T> void AlpsPriorityQueue< T >::push (T *x*) [inline]

Add a element to the heap.

Definition at line 62 of file AlpsPriorityQueue.h.

3.21.2.5 template<class T> void AlpsPriorityQueue< T >::pop ( ) [inline]

Remove the top element from the heap.

Definition at line 68 of file AlpsPriorityQueue.h.

3.21.2.6 template<class T> bool AlpsPriorityQueue< T >::empty ( ) const [inline]

Return true for an empty vector.

Definition at line 74 of file AlpsPriorityQueue.h.

3.21.2.7 template<class T> size_t AlpsPriorityQueue< T >::size ( ) const [inline]

Return the size of the vector.

Definition at line 79 of file AlpsPriorityQueue.h.

3.21.2.8 template<class T> void AlpsPriorityQueue< T >::clear ( ) [inline]

Remove all elements from the vector.

But not delete them.

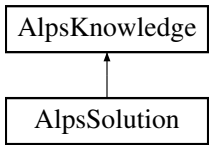
Definition at line 84 of file AlpsPriorityQueue.h.

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

- AlpsPriorityQueue.h

3.22 AlpsSolution Class Reference

Inheritance diagram for AlpsSolution:



Public Member Functions

- AlpsSolution ()
  Default constructor.
- AlpsSolution (const AlpsNodeIndex_t i, const int d)

Constructor to set index and depth.

- virtual ~AlpsSolution ()

  Destructor.

- AlpsNodeIndex_t getIndex ()

  Get index where solution was found.

- void setIndex (const AlpsNodeIndex_t i)

  Set index where solution was found.

- int getDepth ()

  Get depth where solution was found.

- void setDepth (const int d)

  Set depth where solution was found.

- virtual void print (std::ostream &os) const

  Print out the solution.

3.22.1 Detailed Description

Definition at line 35 of file AlpsSolution.h.

3.22.2 Constructor & Destructor Documentation

3.22.2.1 AlpsSolution::AlpsSolution ( ) [inline]

Default constructor.

Definition at line 51 of file AlpsSolution.h.

3.22.2.2 AlpsSolution::AlpsSolution (const AlpsNodeIndex_t *i*, const int *d*) [inline]

Constructor to set index and depth.

Definition at line 59 of file AlpsSolution.h.

3.22.2.3 virtual AlpsSolution::~AlpsSolution ( ) [inline], [virtual]

Destructor.

Definition at line 67 of file AlpsSolution.h.

3.22.3 Member Function Documentation

3.22.3.1 virtual void AlpsSolution::print (std::ostream & *os*) const [inline], [virtual]

Print out the solution.

Definition at line 82 of file AlpsSolution.h.

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

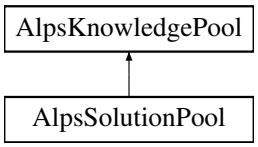
- AlpsSolution.h

3.23 AlpsSolutionPool Class Reference

In the solution pool we assume that the lower the priority value the more desirable the solution is.

`#include <AlpsSolutionPool.h>`

Inheritance diagram for AlpsSolutionPool:



Public Member Functions

- int getNumKnowledges () const

  query the current number of solutions
- bool hasKnowledge () const

  return true if there are any solution stored in the solution pool
- std::pair< AlpsKnowledge *, double > getKnowledge () const

  Get a solution from solution pool, doesn't remove it from the pool.
- void popKnowledge ()

  Remove a solution from the pool.
- void addKnowledge (AlpsKnowledge *sol, double priority)

  Append the solution to the end of the vector of solutions.
- int getMaxNumKnowledges () const

  query the maximum number of solutions
- void setMaxNumKnowledges (int maxsols)

  reset the maximum number of solutions
- std::pair< AlpsKnowledge *, double > getBestKnowledge () const

  Return the best solution.
- void getAllKnowledges (std::vector< std::pair< AlpsKnowledge *, double > > &sols) const

  Return all the solutions of the solution pool in the provided argument vector.
- void clean ()

  Delete all the solutions in pool.

3.23.1 Detailed Description

In the solution pool we assume that the lower the priority value the more desirable the solution is.

Definition at line 33 of file AlpsSolutionPool.h.

3.23.2 Member Function Documentation

3.23.2.1 std::pair<AlpsKnowledge*, double> AlpsSolutionPool::getKnowledge ( ) const [inline], [virtual]

Get a solution from solution pool, doesn't remove it from the pool.

It is implemented same as getBestKnowledge().

Implements AlpsKnowledgePool.

Definition at line 80 of file AlpsSolutionPool.h.

3.23.2.2 void AlpsSolutionPool::addKnowledge (AlpsKnowledge * *sol*, double *priority*) [inline], [virtual]

Append the solution to the end of the vector of solutions.

The solution pool takes over the ownership of the solution

Implements AlpsKnowledgePool.

Definition at line 104 of file AlpsSolutionPool.h.

3.23.2.3 std::pair<AlpsKnowledge*, double> AlpsSolutionPool::getBestKnowledge ( ) const [inline], [virtual]

Return the best solution.

The callee must not delete the returned pointer!

Reimplemented from AlpsKnowledgePool.

Definition at line 155 of file AlpsSolutionPool.h.

3.23.2.4 void AlpsSolutionPool::getAllKnowledges (std::vector< std::pair< AlpsKnowledge *, double > > & *sols*) const [inline], [virtual]

Return all the solutions of the solution pool in the provided argument vector.

The callee must not delete the returned pointers!

Reimplemented from AlpsKnowledgePool.

Definition at line 173 of file AlpsSolutionPool.h.

3.23.2.5 void AlpsSolutionPool::clean ( ) [inline]

Delete all the solutions in pool.

Definition at line 183 of file AlpsSolutionPool.h.

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

- AlpsSolutionPool.h

3.24 AlpsStrLess Struct Reference

A function object to perform lexicographic lexicographic comparison between two C style strings.

`#include <AlpsKnowledge.h>`

3.24.1 Detailed Description

A function object to perform lexicographic lexicographic comparison between two C style strings.

Definition at line 38 of file AlpsKnowledge.h.

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

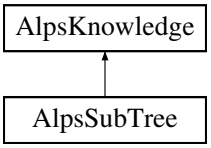
- AlpsKnowledge.h

3.25 AlpsSubTree Class Reference

This class contains the data pertaining to a particular subtree in the search tree.

`#include <AlpsSubTree.h>`

Inheritance diagram for AlpsSubTree:



Public Member Functions

- AlpsSubTree ()

  Default constructor.
- AlpsSubTree (AlpsKnowledgeBroker *kb)

  Useful constructor.
- virtual ~AlpsSubTree ()

  Destructor.
- AlpsTreeNode * activeNode ()

  Get pointer to active node.
- void setActiveNode (AlpsTreeNode *activeNode)

  Set pointer to active node.
- void createChildren (AlpsTreeNode *parent, std::vector< CoinTriple< AlpsNodeDesc *, AlpsNodeStatus, double > > &children, AlpsNodePool *kidNodePool=NULL)

  Create children nodes from the given parent node.
- AlpsSubTree * splitSubTree (int &returnSize, int size=10)

  The function split the subtree and return a subtree of the specified size or available size.
- virtual AlpsReturnStatus exploreSubTree (AlpsTreeNode *root, int nodeLimit, double timeLimit, int &numNodesProcessed, int &numNodesBranched, int &numNodesDiscarded, int &numNodesPartial, int &depth)

  Explore the subtree from `root` as the root of the subtree for given number of nodes or time, depending on which one reach first.
- AlpsReturnStatus exploreUnitWork (bool leaveAsIt, int unitWork, double unitTime, AlpsExitStatus &solStatus, int &numNodesProcessed, int &numNodesBranched, int &numNodesDiscarded, int &numNodesPartial, int &depth, bool &betterSolution)

  Explore the subtree for certain amount of work/time.
- virtual int rampUp (int minNumNodes, int requiredNumNodes, int &depth, AlpsTreeNode *root=NULL)

  Generate required number (specified by a parameter) of nodes.
- virtual AlpsEncoded * encode () const

  This method should encode the content of the subtree and return a pointer to the encoded form.
- virtual AlpsKnowledge * decode (AlpsEncoded &encoded) const

This method should decode and return a pointer to a brand new object, i.e., the method must create a new object on the heap from the decoded data instead of filling up the object for which the method was invoked.

- virtual AlpsSubTree * newSubTree () const

  Create a AlpsSubtree object dynamically.
- void clearNodePools ()

  Remove nodes in pools in the subtree.
- void nullRootActiveNode ()

  Set root and active node to null.
- void reset ()

  Move nodes in node pool, null active node.

query and set member functions

- AlpsTreeNode * getRoot () const

  Get the root node of this subtree.
- void setRoot (AlpsTreeNode *r)

  Set the root node of this subtree.
- AlpsNodePool * nodePool ()

  Access the node pool.
- AlpsNodePool * diveNodePool ()

  Access the node pool.
- void setNodePool (AlpsNodePool *np)

  Set node pool.
- void changeNodePool (AlpsNodePool *np)

  Set node pool.
- double getBestKnowledgeValue () const

  Get the quality of the best node in the subtree.
- AlpsTreeNode * getBestNode () const

  Get the "best" node in the subtree.
- AlpsKnowledgeBroker * getKnowledgeBroker () const

  Get the knowledge broker.
- void setKnowledgeBroker (AlpsKnowledgeBroker *kb)

  Set a pointer to the knowledge broker.
- double getQuality () const

  Get the quality of this subtree.
- double getSolEstimate () const

  Get the emtimated quality of this subtree.
- void incDiveDepth (int num=1)

  Increment dive depth.
- int getDiveDepth ()

  Get dive depth.
- void setDiveDepth (int num)

  Set dive depth.
- double calculateQuality ()

  Calcuate and return the quality of this subtree, which is measured by the quality of the specified number of nodes.
- int **nextIndex** ()
- int getNextIndex () const

  Get the index of the next generated node.
- void setNextIndex (int next)

  Set the index of the next generated node.
- int getNumNodes () const

  Return the number of nodes on this subtree.
- void setNodeSelection (AlpsSearchStrategy< AlpsTreeNode * > *nc)

  Set the node comparision rule.

Protected Member Functions

- void removeDeadNodes (AlpsTreeNode *&node)

  The purpose of this method is to remove nodes that are not needed in the description of the subtree.
- void replaceNode (AlpsTreeNode *oldNode, AlpsTreeNode *newNode)

  This function replaces `oldNode` with `newNode` in the tree.
- void fathomAllNodes ()

  Fathom all nodes on this subtree.

Protected Attributes

- AlpsTreeNode * root_

  The root of the sub tree.
- AlpsNodePool * nodePool_

  A node pool containing the leaf nodes awaiting processing.
- AlpsNodePool * diveNodePool_

  A node pool used when diving.
- AlpsSearchStrategy < AlpsTreeNode * > * diveNodeRule_

  Diving node comparing rule.
- int diveDepth_

  Diving depth.
- AlpsTreeNode * activeNode_

  The next index to be assigned to a new search tree node.
- double quality_

  A quantity indicating how good this subtree is.
- AlpsKnowledgeBroker * broker_

  A pointer to the knowledge broker of the process where this subtree is processed.

3.25.1 Detailed Description

This class contains the data pertaining to a particular subtree in the search tree.

In order to improve scalability, we will try to deal with entire subtrees as much as possible. They will be the basic unit of work that will be passed between processes.

Definition at line 47 of file AlpsSubTree.h.

3.25.2 Constructor & Destructor Documentation

3.25.2.1 AlpsSubTree::AlpsSubTree ( )

Default constructor.

3.25.2.2 AlpsSubTree::AlpsSubTree (AlpsKnowledgeBroker * *kb*)

Useful constructor.

3.25.2.3 virtual AlpsSubTree::~AlpsSubTree ( ) [virtual]

Destructor.

3.25.3 Member Function Documentation

3.25.3.1 void AlpsSubTree::removeDeadNodes (AlpsTreeNode *& *node*) [protected]

The purpose of this method is to remove nodes that are not needed in the description of the subtree.

The argument node must have status `fathomed`. First, the argument node is removed, and then the parent is examined to determine whether it has any children left. If it has none, then this function is called recursively on the parent. This removes all nodes that are no longer needed.

3.25.3.2 void AlpsSubTree::replaceNode (AlpsTreeNode * *oldNode,* AlpsTreeNode * *newNode*) [protected]

This function replaces `oldNode` with `newNode` in the tree.

3.25.3.3 void AlpsSubTree::fathomAllNodes ( ) [protected]

Fathom all nodes on this subtree.

Set activeNode_ and root_ to NULL.

3.25.3.4 void AlpsSubTree::createChildren (AlpsTreeNode * *parent,* std::vector< CoinTriple< AlpsNodeDesc *, AlpsNodeStatus, double > > & *children,* AlpsNodePool * *kidNodePool* = `NULL`)

Create children nodes from the given parent node.

3.25.3.5 AlpsTreeNode* AlpsSubTree::getRoot ( ) const [inline]

Get the root node of this subtree.

Definition at line 129 of file AlpsSubTree.h.

3.25.3.6 void AlpsSubTree::setRoot (AlpsTreeNode * *r*) [inline]

Set the root node of this subtree.

Definition at line 132 of file AlpsSubTree.h.

3.25.3.7 AlpsNodePool* AlpsSubTree::nodePool ( ) [inline]

Access the node pool.

Definition at line 135 of file AlpsSubTree.h.

3.25.3.8 AlpsNodePool* AlpsSubTree::diveNodePool ( ) [inline]

Access the node pool.

Definition at line 138 of file AlpsSubTree.h.

3.25.3.9 void AlpsSubTree::setNodePool (AlpsNodePool * *np*) [inline]

Set node pool.

Delete previous node pool and nodes in pool if exit.

Definition at line 141 of file AlpsSubTree.h.

3.25.3.10 void AlpsSubTree::changeNodePool (AlpsNodePool * *np*) [inline]

Set node pool.

Delete previous node pool, but not the nodes in pool.

Definition at line 150 of file AlpsSubTree.h.

3.25.3.11 double AlpsSubTree::getBestKnowledgeValue ( ) const

Get the quality of the best node in the subtree.

3.25.3.12 AlpsTreeNode* AlpsSubTree::getBestNode ( ) const

Get the "best" node in the subtree.

3.25.3.13 AlpsKnowledgeBroker* AlpsSubTree::getKnowledgeBroker ( ) const [inline]

Get the knowledge broker.

Definition at line 169 of file AlpsSubTree.h.

3.25.3.14 void AlpsSubTree::setKnowledgeBroker (AlpsKnowledgeBroker * *kb*) [inline]

Set a pointer to the knowledge broker.

Definition at line 172 of file AlpsSubTree.h.

3.25.3.15 double AlpsSubTree::getQuality ( ) const [inline]

Get the quality of this subtree.

Definition at line 178 of file AlpsSubTree.h.

3.25.3.16 double AlpsSubTree::getSolEstimate ( ) const [inline]

Get the emtimated quality of this subtree.

Definition at line 181 of file AlpsSubTree.h.

3.25.3.17 double AlpsSubTree::calculateQuality ( )

Calcuate and return the quality of this subtree, which is measured by the quality of the specified number of nodes.

3.25.3.18 int AlpsSubTree::getNextIndex ( ) const

Get the index of the next generated node.

3.25.3.19 void AlpsSubTree::setNextIndex (int *next*)

Set the index of the next generated node.

3.25.3.20 int AlpsSubTree::getNumNodes ( ) const [inline]

Return the number of nodes on this subtree.

Definition at line 214 of file AlpsSubTree.h.

3.25.3.21 void AlpsSubTree::setNodeSelection (AlpsSearchStrategy< AlpsTreeNode * > * *nc*) [inline]

Set the node comparision rule.

Definition at line 228 of file AlpsSubTree.h.

3.25.3.22 AlpsSubTree* AlpsSubTree::splitSubTree (int & *returnSize,* int *size* = 10)

The function split the subtree and return a subtree of the specified size or available size.

3.25.3.23 virtual AlpsReturnStatus AlpsSubTree::exploreSubTree (AlpsTreeNode * *root,* int *nodeLimit,* double *timeLimit,* int & *numNodesProcessed,* int & *numNodesBranched,* int & *numNodesDiscarded,* int & *numNodesPartial,* int & *depth*) [virtual]

Explore the subtree from `root` as the root of the subtree for given number of nodes or time, depending on which one reach first.

Only for serial code.

3.25.3.24 AlpsReturnStatus AlpsSubTree::exploreUnitWork (bool *leaveAsIt,* int *unitWork,* double *unitTime,* AlpsExitStatus & *solStatus,* int & *numNodesProcessed,* int & *numNodesBranched,* int & *numNodesDiscarded,* int & *numNodesPartial,* int & *depth,* bool & *betterSolution*)

Explore the subtree for certain amount of work/time.

leaveAsIt means exit immediately after reseach limits: do not put activeNode_ in pool, do not move nodes in divePool_ in regular pool.

3.25.3.25 virtual int AlpsSubTree::rampUp (int *minNumNodes,* int *requiredNumNodes,* int & *depth,* AlpsTreeNode * *root* = NULL) [virtual]

Generate required number (specified by a parameter) of nodes.

This function is used by master and hubs.

3.25.3.26 virtual AlpsEncoded* AlpsSubTree::encode ( ) const [virtual]

This method should encode the content of the subtree and return a pointer to the encoded form.

Only parallel code need this function.

Reimplemented from AlpsKnowledge.

3.25.3.27 virtual AlpsKnowledge* AlpsSubTree::decode (AlpsEncoded & *encoded*) const [virtual]

This method should decode and return a pointer to a *brand new object*, i.e., the method must create a new object on the heap from the decoded data instead of filling up the object for which the method was invoked.

Only parallel code need this function.

Reimplemented from AlpsKnowledge.

3.25.3.28 virtual AlpsSubTree* AlpsSubTree::newSubTree ( ) const [inline], [virtual]

Create a AlpsSubtree object dynamically.

Only parallel code need this function.

Definition at line 285 of file AlpsSubTree.h.

3.25.3.29 void AlpsSubTree::clearNodePools ( ) [inline]

Remove nodes in pools in the subtree.

Do not free memory.

Definition at line 290 of file AlpsSubTree.h.

3.25.3.30 void AlpsSubTree::reset ( ) [inline]

Move nodes in node pool, null active node.

Definition at line 306 of file AlpsSubTree.h.

3.25.4 Member Data Documentation

3.25.4.1 AlpsTreeNode* AlpsSubTree::root_ [protected]

The root of the sub tree.

Definition at line 52 of file AlpsSubTree.h.

3.25.4.2 AlpsNodePool* AlpsSubTree::nodePool_ [protected]

A node pool containing the leaf nodes awaiting processing.

Definition at line 55 of file AlpsSubTree.h.

3.25.4.3 AlpsNodePool* AlpsSubTree::diveNodePool_ [protected]

A node pool used when diving.

Definition at line 58 of file AlpsSubTree.h.

3.25.4.4 AlpsSearchStrategy<AlpsTreeNode*>* AlpsSubTree::diveNodeRule_ [protected]

Diving node comparing rule.

Definition at line 61 of file AlpsSubTree.h.

3.25.4.5 AlpsTreeNode* AlpsSubTree::activeNode_ [protected]

The next index to be assigned to a new search tree node.

This is the node that is currently being processed. Note that since this is the worker, there is only one.

Definition at line 71 of file AlpsSubTree.h.

3.25.4.6 double AlpsSubTree::quality_ [protected]

A quantity indicating how good this subtree is.

Definition at line 74 of file AlpsSubTree.h.

3.25.4.7 AlpsKnowledgeBroker* AlpsSubTree::broker_ [protected]

A pointer to the knowledge broker of the process where this subtree is processed.

Definition at line 79 of file AlpsSubTree.h.

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

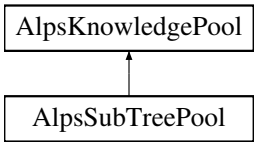
- AlpsSubTree.h

3.26 AlpsSubTreePool Class Reference

The subtree pool is used to store subtrees.

`#include <AlpsSubTreePool.h>`

Inheritance diagram for AlpsSubTreePool:



Public Member Functions

- int getNumKnowledges () const

  Query the number of subtrees in the pool.
- bool hasKnowledge () const

  Check whether there is a subtree in the subtree pool.
- std::pair< AlpsKnowledge *, double > getKnowledge () const

  Get a subtree from subtree pool, doesn't remove it from the pool.
- void popKnowledge ()

  Remove a subtree from the pool.
- void addKnowledge (AlpsKnowledge *subTree, double priority)

  Add a subtree to the subtree pool.
- const AlpsPriorityQueue < AlpsSubTree * > & getSubTreeList () const

  Return the container of subtrees.
- void setComparison (AlpsSearchStrategy< AlpsSubTree * > &compare)

  Set comparison function and resort heap.
- void deleteGuts ()

  Delete the subtrees in the pool.
- double getBestQuality ()

  Get the quality of the best subtree.

3.26.1 Detailed Description

The subtree pool is used to store subtrees.

Definition at line 32 of file AlpsSubTreePool.h.

3.26.2 Member Function Documentation

3.26.2.1 int AlpsSubTreePool::getNumKnowledges ( ) const [inline], [virtual]

Query the number of subtrees in the pool.

Implements AlpsKnowledgePool.

Definition at line 49 of file AlpsSubTreePool.h.

3.26.2.2 bool AlpsSubTreePool::hasKnowledge ( ) const [inline], [virtual]

Check whether there is a subtree in the subtree pool.

Reimplemented from AlpsKnowledgePool.

Definition at line 52 of file AlpsSubTreePool.h.

3.26.2.3 void AlpsSubTreePool::addKnowledge (AlpsKnowledge * *subTree,* double *priority*) [inline], [virtual]

Add a subtree to the subtree pool.

Implements AlpsKnowledgePool.

Definition at line 67 of file AlpsSubTreePool.h.

3.26.2.4 const AlpsPriorityQueue< AlpsSubTree*>& AlpsSubTreePool::getSubTreeList ( ) const [inline]

Return the container of subtrees.

Definition at line 74 of file AlpsSubTreePool.h.

3.26.2.5 void AlpsSubTreePool::setComparison (AlpsSearchStrategy< AlpsSubTree * > & *compare*) [inline]

Set comparison function and resort heap.

Definition at line 77 of file AlpsSubTreePool.h.

3.26.2.6 void AlpsSubTreePool::deleteGuts ( ) [inline]

Delete the subtrees in the pool.

Definition at line 82 of file AlpsSubTreePool.h.

3.26.2.7 double AlpsSubTreePool::getBestQuality ( ) [inline]

Get the quality of the best subtree.

Definition at line 90 of file AlpsSubTreePool.h.

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

- AlpsSubTreePool.h

3.27 AlpsTimer Class Reference

Public Member Functions

- void reset ()
  Reset.
- void start ()
  Start to count times.
- void stop ()
  Stop timer and computing times.
- double getCpuTime ()
  Get cpu timee.
- double getWallClock ()
  Get cpu timee.

- double getTime ()

  Get time depends on clock type.
- int getClockType ()

  Get/Set clock type.
- bool reachCpuLimit ()

  Check if cpu time reach limit.
- bool reachWallLimit ()

  Check if wallclock time reach limit.

Public Attributes

- double limit_

  Time limit.
- double cpu_

  Cpu time.
- double wall_

  Wall clock time.

3.27.1 Detailed Description

Definition at line 75 of file AlpsTime.h.

3.27.2 Member Function Documentation

3.27.2.1 void AlpsTimer::reset ( ) [inline]

Reset.

Definition at line 101 of file AlpsTime.h.

3.27.2.2 void AlpsTimer::start ( ) [inline]

Start to count times.

Definition at line 111 of file AlpsTime.h.

3.27.2.3 void AlpsTimer::stop ( ) [inline]

Stop timer and computing times.

Definition at line 117 of file AlpsTime.h.

3.27.2.4 double AlpsTimer::getCpuTime ( ) [inline]

Get cpu timee.

Definition at line 130 of file AlpsTime.h.

3.27.2.5 double AlpsTimer::getWallClock ( ) [inline]

Get cpu timee.

Definition at line 137 of file AlpsTime.h.

3.27.2.6 double AlpsTimer::getTime ( ) [inline]

Get time depends on clock type.

Definition at line 144 of file AlpsTime.h.

3.27.2.7 bool AlpsTimer::reachCpuLimit ( ) [inline]

Check if cpu time reach limit.

Definition at line 164 of file AlpsTime.h.

3.27.2.8 bool AlpsTimer::reachWallLimit ( ) [inline]

Check if wallclock time reach limit.

Definition at line 176 of file AlpsTime.h.

3.27.3 Member Data Documentation

3.27.3.1 double AlpsTimer::limit_

Time limit.

Definition at line 82 of file AlpsTime.h.

3.27.3.2 double AlpsTimer::cpu_

Cpu time.

Definition at line 90 of file AlpsTime.h.

3.27.3.3 double AlpsTimer::wall_

Wall clock time.

Definition at line 93 of file AlpsTime.h.

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

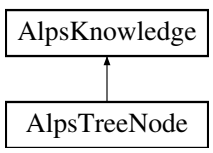
- AlpsTime.h

3.28 AlpsTreeNode Class Reference

This class holds one node of the search tree.

`#include <AlpsTreeNode.h>`

Inheritance diagram for AlpsTreeNode:



Public Member Functions

- AlpsNodeDesc * modifyDesc ()

  Access the desc so that can modify it.
- AlpsKnowledgeBroker * getKnowledgeBroker () const

  Functions to access/set the knwoledge broker.
- virtual AlpsTreeNode * createNewTreeNode (AlpsNodeDesc *&desc) const =0

  The purpose of this function is be able to create the children of a node after branching.
- void removeChild (AlpsTreeNode *&child)

  Remove the pointer to given child from the list of children.
- void addChild (AlpsTreeNode *&child)

  Add a child to the list of children for this node.
- void removeDescendants ()

  Removes all the descendants of the node.

- AlpsNodeStatus getStatus () const

  Query/set the current status.

- bool isCandidate () const

  Query functions about specific stati.

- bool isActive () const

  Query/set node in-process indicator.

- AlpsNodeIndex_t getIndex () const

  Query/set node identifier (unique within subtree).

- int getDepth () const

  Query/set what depth the search tree node is at.

- double getSolEstimate () const

  Query/set the solution estimate of the node.

- double getQuality () const

  Query/set the quality of the node.

- int getNumChildren () const

  Query/set what the number of children.

- AlpsTreeNode * getChild (const int i) const

  Query/set pointer to the ith child.
- void setChild (const int i, AlpsTreeNode *node)

  Returns a const pointer to the ith child.

- AlpsTreeNode * getParent () const

  Get/set subtree.

- AlpsNodeIndex_t getParentIndex () const

  Get/set the index of the parent of the node.

- int getExplicit () const

  Get/set the indication of whether the node has full or differencing description.

- virtual void convertToExplicit ()

  Convert explicit description to difference, and vise-vesa.

- int getDiving () const

  If the this node is in a diving process.

- int getSentMark () const

  Various marks used in parallel code.

Protected Attributes

- bool active_

  The subtree own this node.
- AlpsNodeIndex_t index_

  The unique index of the tree node (across the whole search tree).
- int depth_

  The depth of the node (in the whole tree – the root is at depth 0).
- double solEstimate_

  The solution estimate.
- double quality_

  The quality of this node.
- AlpsTreeNode * parent_

  The parent of the tree node.
- AlpsNodeIndex_t parentIndex_

  The index of parent of the tree node.
- int numChildren_

  The number of children.
- int explicit_

  Indicate whether the node description is explicit(1) or relative(0).
- AlpsNodeDesc * desc_

  The actual description of the tree node.
- AlpsNodeStatus status_

  The current status of the node.
- AlpsKnowledgeBroker * knowledgeBroker_

  A pointer to the knowledge broker of the process where this node is processed.
- int sentMark_

  Various mark used in splitting and passing subtrees.
- bool diving_

  When processing it, if it is in the diving processing.

3.28.1 Detailed Description

This class holds one node of the search tree.

Note that the generic search procedure doesn't know anything about the nodes in the tree other than their index, lower bound, etc. Other application-specific data is contained in derived classes, but is not needed for the basic operation of the search tree.

Definition at line 50 of file AlpsTreeNode.h.

3.28.2 Member Function Documentation

3.28.2.1 AlpsNodeDesc* AlpsTreeNode::modifyDesc ( ) [inline]

Access the desc so that can modify it.

Definition at line 155 of file AlpsTreeNode.h.

3.28.2.2 virtual AlpsTreeNode* AlpsTreeNode::createNewTreeNode (AlpsNodeDesc *& *desc*) const [pure virtual]

The purpose of this function is be able to create the children of a node after branching.

3.28.2.3 AlpsNodeStatus AlpsTreeNode::getStatus ( ) const [inline]

Query/set the current status.

Definition at line 176 of file AlpsTreeNode.h.

3.28.2.4 bool AlpsTreeNode::isCandidate ( ) const [inline]

Query functions about specific stati.

Definition at line 182 of file AlpsTreeNode.h.

3.28.2.5 bool AlpsTreeNode::isActive ( ) const [inline]

Query/set node in-process indicator.

Definition at line 198 of file AlpsTreeNode.h.

3.28.2.6 AlpsNodeIndex_t AlpsTreeNode::getIndex ( ) const [inline]

Query/set node identifier (unique within subtree).

Definition at line 204 of file AlpsTreeNode.h.

3.28.2.7 int AlpsTreeNode::getDepth ( ) const [inline]

Query/set what depth the search tree node is at.

Definition at line 210 of file AlpsTreeNode.h.

3.28.2.8 double AlpsTreeNode::getSolEstimate ( ) const [inline]

Query/set the solution estimate of the node.

Definition at line 216 of file AlpsTreeNode.h.

3.28.2.9 double AlpsTreeNode::getQuality ( ) const [inline]

Query/set the quality of the node.

Definition at line 222 of file AlpsTreeNode.h.

3.28.2.10 int AlpsTreeNode::getNumChildren ( ) const [inline]

Query/set what the number of children.

Definition at line 228 of file AlpsTreeNode.h.

3.28.2.11 AlpsTreeNode* AlpsTreeNode::getChild (const int *i*) const [inline]

Query/set pointer to the ith child.

Definition at line 251 of file AlpsTreeNode.h.

3.28.2.12 void AlpsTreeNode::setChild (const int *i,* AlpsTreeNode * *node*) [inline]

Returns a const pointer to the ith child.

Definition at line 258 of file AlpsTreeNode.h.

3.28.2.13 void AlpsTreeNode::removeChild (AlpsTreeNode *& *child*)

Remove the pointer to given child from the list of children.

This method deletes the child as well. An error is thrown if the argument is not a pointer to a child.

3.28.2.14 void AlpsTreeNode::addChild (AlpsTreeNode *& *child*)

Add a child to the list of children for this node.

3.28.2.15 void AlpsTreeNode::removeDescendants ( )

Removes all the descendants of the node.

We might want to do this in some cases where we are cutting out a subtree and replacing it with another one.

3.28.2.16 AlpsTreeNode* AlpsTreeNode::getParent ( ) const [inline]

Get/set subtree.

Get/set the parent of the node

Definition at line 281 of file AlpsTreeNode.h.

3.28.2.17 AlpsNodeIndex_t AlpsTreeNode::getParentIndex ( ) const [inline]

Get/set the index of the parent of the node.

Used in decode subtree.

Definition at line 287 of file AlpsTreeNode.h.

3.28.2.18 int AlpsTreeNode::getExplicit ( ) const [inline]

Get/set the indication of whether the node has full or differencing description.

Definition at line 295 of file AlpsTreeNode.h.

3.28.2.19 int AlpsTreeNode::getDiving ( ) const [inline]

If the this node is in a diving process.

Definition at line 307 of file AlpsTreeNode.h.

3.28.2.20 int AlpsTreeNode::getSentMark ( ) const [inline]

Various marks used in parallel code.

Definition at line 313 of file AlpsTreeNode.h.

3.28.3 Member Data Documentation

3.28.3.1 bool AlpsTreeNode::active_ [protected]

The subtree own this node.

Whether the node is being worked on at the moment

Definition at line 60 of file AlpsTreeNode.h.

3.28.3.2 AlpsNodeIndex_t AlpsTreeNode::index_ [protected]

The unique index of the tree node (across the whole search tree).

Definition at line 63 of file AlpsTreeNode.h.

3.28.3.3 int AlpsTreeNode::depth_ [protected]

The depth of the node (in the whole tree – the root is at depth 0).

Definition at line 66 of file AlpsTreeNode.h.

3.28.3.4 double AlpsTreeNode::solEstimate_ [protected]

The solution estimate.

The smaller the better.

Definition at line 69 of file AlpsTreeNode.h.

3.28.3.5 double AlpsTreeNode::quality_ [protected]

The quality of this node.

The smaller the better.

Definition at line 72 of file AlpsTreeNode.h.

3.28.3.6 AlpsTreeNode* AlpsTreeNode::parent_ [protected]

The parent of the tree node.

Definition at line 75 of file AlpsTreeNode.h.

3.28.3.7 AlpsNodeIndex_t AlpsTreeNode::parentIndex_ [protected]

The index of parent of the tree node.

Used in decoding sub tree.

Definition at line 78 of file AlpsTreeNode.h.

3.28.3.8 int AlpsTreeNode::numChildren_ [protected]

The number of children.

Definition at line 81 of file AlpsTreeNode.h.

3.28.3.9 int AlpsTreeNode::explicit_ [protected]

Indicate whether the node description is explicit(1) or relative(0).

Default is relative.

Definition at line 92 of file AlpsTreeNode.h.

3.28.3.10 AlpsNodeDesc* AlpsTreeNode::desc_ [protected]

The actual description of the tree node.

Definition at line 95 of file AlpsTreeNode.h.

3.28.3.11 AlpsNodeStatus AlpsTreeNode::status_ [protected]

The current status of the node.

Definition at line 98 of file AlpsTreeNode.h.

3.28.3.12 AlpsKnowledgeBroker* AlpsTreeNode::knowledgeBroker_ [protected]

A pointer to the knowledge broker of the process where this node is processed.

Definition at line 103 of file AlpsTreeNode.h.

3.28.3.13 int AlpsTreeNode::sentMark_ [protected]

Various mark used in splitting and passing subtrees.

Definition at line 107 of file AlpsTreeNode.h.

3.28.3.14 bool AlpsTreeNode::diving_ [protected]

When processing it, if it is in the diving processing.

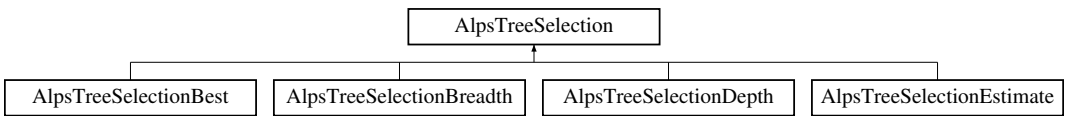
Definition at line 110 of file AlpsTreeNode.h.

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

- AlpsTreeNode.h

3.29 AlpsTreeSelection Class Reference

Inheritance diagram for AlpsTreeSelection:



Public Member Functions

- AlpsTreeSelection ()

  Default Constructor.
- virtual ~AlpsTreeSelection ()

  Default Destructor.
- virtual bool compare (AlpsSubTree *x, AlpsSubTree *y)=0

  This returns true if the quality of the subtree y is better (the less the better) than that the subtree x.

3.29.1 Detailed Description

Definition at line 33 of file AlpsSearchStrategy.h.

3.29.2 Constructor & Destructor Documentation

3.29.2.1 AlpsTreeSelection::AlpsTreeSelection ( ) [inline]

Default Constructor.

Definition at line 37 of file AlpsSearchStrategy.h.

3.29.2.2 virtual AlpsTreeSelection::~AlpsTreeSelection ( ) [inline], [virtual]

Default Destructor.

Definition at line 40 of file AlpsSearchStrategy.h.

3.29.3 Member Function Documentation

3.29.3.1 virtual bool AlpsTreeSelection::compare (AlpsSubTree * *x*, AlpsSubTree * *y*) [pure virtual]

This returns true if the quality of the subtree y is better (the less the better) than that the subtree x.

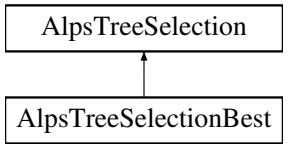
Implemented in AlpsTreeSelectionEstimate, AlpsTreeSelectionDepth, AlpsTreeSelectionBreadth, and AlpsTreeSelectionBest.

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

- AlpsSearchStrategy.h

3.30 AlpsTreeSelectionBest Class Reference

Inheritance diagram for AlpsTreeSelectionBest:



Public Member Functions

- AlpsTreeSelectionBest ()

  Default Constructor.
- virtual ~AlpsTreeSelectionBest ()

  Default Destructor.
- virtual bool compare (AlpsSubTree *x, AlpsSubTree *y)

  This returns true if the quality of the subtree y is better (the less the better) than that the subtree x.

3.30.1 Detailed Description

Definition at line 73 of file AlpsSearchStrategy.h.

3.30.2 Constructor & Destructor Documentation

3.30.2.1 AlpsTreeSelectionBest::AlpsTreeSelectionBest ( ) [inline]

Default Constructor.

Definition at line 77 of file AlpsSearchStrategy.h.

3.30.2.2 virtual AlpsTreeSelectionBest::~AlpsTreeSelectionBest ( ) [inline], [virtual]

Default Destructor.

Definition at line 80 of file AlpsSearchStrategy.h.

3.30.3 Member Function Documentation

3.30.3.1 virtual bool AlpsTreeSelectionBest::compare (AlpsSubTree * *x*, AlpsSubTree * *y*) [virtual]

This returns true if the quality of the subtree y is better (the less the better) than that the subtree x.

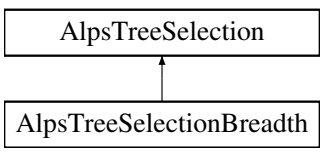
Implements AlpsTreeSelection.

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

- AlpsSearchStrategy.h

3.31 AlpsTreeSelectionBreadth Class Reference

Inheritance diagram for AlpsTreeSelectionBreadth:



Public Member Functions

- AlpsTreeSelectionBreadth ()
  Default Constructor.
- virtual ~AlpsTreeSelectionBreadth ()
  Default Destructor.
- virtual bool compare (AlpsSubTree *x, AlpsSubTree *y)
  This returns true if the depth of the root node in subtree y is smaller than that of the root node in subtree x.

3.31.1 Detailed Description

Definition at line 89 of file AlpsSearchStrategy.h.

3.31.2 Constructor & Destructor Documentation

3.31.2.1 virtual AlpsTreeSelectionBreadth::~AlpsTreeSelectionBreadth ( ) [inline], [virtual]

Default Destructor.

Definition at line 96 of file AlpsSearchStrategy.h.

3.31.3 Member Function Documentation

3.31.3.1 virtual bool AlpsTreeSelectionBreadth::compare (AlpsSubTree * x, AlpsSubTree * y) [virtual]

This returns true if the depth of the root node in subtree y is smaller than that of the root node in subtree x.

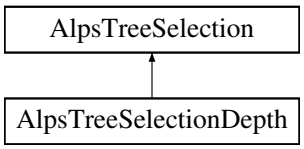
Implements AlpsTreeSelection.

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

- AlpsSearchStrategy.h

3.32 AlpsTreeSelectionDepth Class Reference

Inheritance diagram for AlpsTreeSelectionDepth:



Public Member Functions

- AlpsTreeSelectionDepth ()
  Default Constructor.
- virtual ~AlpsTreeSelectionDepth ()
  Default Destructor.
- virtual bool compare (AlpsSubTree *x, AlpsSubTree *y)
  This returns true if the depth of the root node in subtree y is greater than that of the root node in subtree x.

3.32.1 Detailed Description

Definition at line 105 of file AlpsSearchStrategy.h.

3.32.2 Constructor & Destructor Documentation

3.32.2.1 virtual AlpsTreeSelectionDepth::~AlpsTreeSelectionDepth ( ) [inline], [virtual]

Default Destructor.

Definition at line 112 of file AlpsSearchStrategy.h.

3.32.3 Member Function Documentation

3.32.3.1 virtual bool AlpsTreeSelectionDepth::compare (AlpsSubTree * *x*, AlpsSubTree * *y*) [virtual]

This returns true if the depth of the root node in subtree y is greater than that of the root node in subtree x.

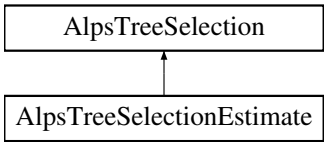
Implements AlpsTreeSelection.

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

- AlpsSearchStrategy.h

3.33 AlpsTreeSelectionEstimate Class Reference

Inheritance diagram for AlpsTreeSelectionEstimate:



Public Member Functions

- AlpsTreeSelectionEstimate ()
  Default Constructor.
- virtual ~AlpsTreeSelectionEstimate ()
  Default Destructor.
- virtual bool compare (AlpsSubTree *x, AlpsSubTree *y)
  This returns true if the estimated quality of the subtree y is better (the less the better) than that the subtree x.

3.33.1 Detailed Description

Definition at line 121 of file AlpsSearchStrategy.h.

3.33.2 Constructor & Destructor Documentation

3.33.2.1 AlpsTreeSelectionEstimate::AlpsTreeSelectionEstimate ( ) [inline]

Default Constructor.

Definition at line 125 of file AlpsSearchStrategy.h.

3.33.2.2 virtual AlpsTreeSelectionEstimate::~AlpsTreeSelectionEstimate ( ) [inline], [virtual]

Default Destructor.

Definition at line 128 of file AlpsSearchStrategy.h.

3.33.3 Member Function Documentation

3.33.3.1 virtual bool AlpsTreeSelectionEstimate::compare (AlpsSubTree * *x*, AlpsSubTree * *y*) [virtual]

This returns true if the estimated quality of the subtree y is better (the less the better) than that the subtree x.

Implements AlpsTreeSelection.

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

- AlpsSearchStrategy.h

3.34 DeletePtrObject Struct Reference

3.34.1 Detailed Description

Definition at line 62 of file AlpsHelperFunctions.h.

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

- AlpsHelperFunctions.h

3.35 TotalWorkload Class Reference

A functor class used in calulating total workload in a node pool.

`#include <AlpsHelperFunctions.h>`

3.35.1 Detailed Description

A functor class used in calulating total workload in a node pool.

Definition at line 38 of file AlpsHelperFunctions.h.

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

- AlpsHelperFunctions.h

Index