opt_jr_doc

Generated by Doxygen 1.8.5

Tue Dec 12 2017 04:12:17

Contents

1 PACS_PROJECT **1**

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 Application Class Reference 7

4.1.1 Detailed Description 9

4.1.2 Constructor & Destructor Documentation 9

4.1.2.1 Application 9

4.1.3 Member Function Documentation 9

4.1.3.1 computeAlphaBeta 9

4.1.4 Member Data Documentation 9

4.1.4.1 alpha 9

4.1.4.2 app_id 9

4.1.4.3 baseFO 9

4.1.4.4 beta 9

4.1.4.5 bound 9

4.1.4.6 boundIterations 9

4.1.4.7 chi_0 10

4.1.4.8 chi_C 10

4.1.4.9 csi 10

4.1.4.10 currentCores_d 10

4.1.4.11 datasetSize 10

4.1.4.12 Deadline_d 10

4.1.4.13 index 10

4.1.4.14 initialBaseFO 10

4.1.4.15 m 10

4.1.4.16 M 10

4.1.4.17 mode . . . 10
4.1.4.18 nCores_DB_d . . . 10
4.1.4.19 nu_d . . . 10
4.1.4.20 R_d . . . 11
4.1.4.21 session_app_id . . . 11
4.1.4.22 stage . . . 11
4.1.4.23 term_i . . . 11
4.1.4.24 V . . . 11
4.1.4.25 v . . . 11
4.1.4.26 vm . . . 11
4.1.4.27 w . . . 11
4.2 Batch Class Reference . . . 11
4.2.1 Detailed Description . . . 13
4.2.2 Constructor & Destructor Documentation . . . 13
4.2.2.1 Batch . . . 13
4.2.3 Member Function Documentation . . . 13
4.2.3.1 calculate_nu . . . 13
4.2.3.2 fixInitialSolution . . . 13
4.2.3.3 initialize . . . 14
4.2.4 Member Data Documentation . . . 15
4.2.4.1 APPs . . . 15
4.3 Bounds Class Reference . . . 15
4.3.1 Detailed Description . . . 16
4.3.2 Member Function Documentation . . . 16
4.3.2.1 calculateBounds . . . 16
4.4 Candidate Class Reference . . . 17
4.4.1 Detailed Description . . . 18
4.4.2 Constructor & Destructor Documentation . . . 18
4.4.2.1 Candidate . . . 18
4.4.3 Member Data Documentation . . . 18
4.4.3.1 app_i . . . 18
4.4.3.2 app_j . . . 18
4.4.3.3 delta_i . . . 18
4.4.3.4 delta_j . . . 18
4.4.3.5 deltaFO . . . 18
4.4.3.6 newCoreAssignment_i . . . 18
4.4.3.7 newCoreAssignment_j . . . 18
4.4.3.8 real_i . . . 19
4.4.3.9 real_j . . . 19
4.5 ObjFun Class Reference . . . 19

4.5.1 Detailed Description . . . 19
4.5.2 Member Function Documentation . . . 19
4.5.2.1 ObjFunctionComponent . . . 19
4.5.2.2 ObjFunctionComponentApprox . . . 20
4.5.2.3 ObjFunctionGlobal . . . 21
4.6 optJrParameters Class Reference . . . 21
4.6.1 Detailed Description . . . 22
4.6.2 Constructor & Destructor Documentation . . . 22
4.6.2.1 optJrParameters . . . 22
4.6.3 Member Function Documentation . . . 22
4.6.3.1 get_cache . . . 22
4.6.3.2 get_debug . . . 23
4.6.3.3 get_filename . . . 23
4.6.3.4 get_globalFOcalculation . . . 24
4.6.3.5 get_K . . . 24
4.6.3.6 get_maxIteration . . . 24
4.6.3.7 get_number . . . 24
4.6.3.8 get_numberOfThreads . . . 25
4.6.3.9 get_simulator . . . 25
4.6.3.10 set_numberOfThreads . . . 26
4.7 Search Class Reference . . . 26
4.7.1 Detailed Description . . . 26
4.7.2 Member Function Documentation . . . 26
4.7.2.1 localSearch . . . 26
4.8 Statistic Class Reference . . . 27
4.8.1 Detailed Description . . . 28
4.8.2 Constructor & Destructor Documentation . . . 28
4.8.2.1 Statistic . . . 28
4.8.3 Member Function Documentation . . . 28
4.8.3.1 get_FO_Total . . . 28
4.8.3.2 get_iteration . . . 28
4.8.3.3 get_size . . . 28

5 File Documentation **29**

5.1 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/appByWeight.cpp File Reference . . . 29
5.1.1 Function Documentation . . . 29
5.1.1.1 addApplicationPointer . . . 29
5.2 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/appByWeight.hh File Reference . . . 30
5.2.1 Typedef Documentation . . . 31
5.2.1.1 appByWeight . . . 31

5.2.2 Function Documentation . . . 31
5.2.2.1 addApplicationPointer . . . 31
5.3 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/application.cpp File Reference . . . 31
5.4 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/application.hh File Reference . . . 32
5.4.1 Macro Definition Documentation . . . 32
5.4.1.1 R_ALGORITHM . . . 32
5.5 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/batch.cpp File Reference . . . 32
5.6 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/batch.hh File Reference . . . 33
5.7 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/bounds.cpp File Reference . . . 34
5.8 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/bounds.hh File Reference . . . 34
5.9 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/candidates.cpp File Reference . . . 35
5.9.1 Function Documentation . . . 36
5.9.1.1 addCandidate . . . 36
5.9.1.2 invokePredictorOpenMP . . . 36
5.10 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/candidates.hh File Reference . . . 37
5.10.1 Typedef Documentation . . . 38
5.10.1.1 sCandidates . . . 38
5.10.2 Function Documentation . . . 38
5.10.2.1 addCandidate . . . 38
5.10.2.2 invokePredictorOpenMP . . . 38
5.11 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/db.cpp File Reference . . . 39
5.11.1 Function Documentation . . . 40
5.11.1.1 DBclose . . . 40
5.11.1.2 DBerror . . . 40
5.11.1.3 DBopen . . . 41
5.11.1.4 executeSQL . . . 41
5.12 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/db.hh File Reference . . . 42
5.12.1 Function Documentation . . . 43
5.12.1.1 DBclose . . . 43
5.12.1.2 DBerror . . . 43
5.12.1.3 DBopen . . . 44
5.12.1.4 executeSQL . . . 44
5.13 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/debugmessage.cpp File Reference . . . 45
5.13.1 Function Documentation . . . 45
5.13.1.1 debugMessage . . . 45
5.14 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/debugmessage.hh File Reference . . . 46
5.14.1 Function Documentation . . . 47
5.14.1.1 debugMessage . . . 47
5.15 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor.cpp File Reference . . . 48
5.15.1 Function Documentation . . . 48

5.15.1.1 invokePredictor . . . 49
5.16 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor.hh File Reference . . . 49
5.16.1 Macro Definition Documentation . . . 50
5.16.1.1 RESIDUAL_DAGSIM . . . 50
5.16.1.2 WHOLE_DAGSIM . . . 50
5.16.2 Function Documentation . . . 50
5.16.2.1 invokePredictor . . . 51
5.17 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor_helper.cpp File Reference 51
5.17.1 Function Documentation . . . 52
5.17.1.1 _run . . . 52
5.17.1.2 extractRowMatchingPattern . . . 53
5.17.1.3 extractRowN . . . 53
5.17.1.4 extractWord . . . 53
5.17.1.5 ls . . . 54
5.17.1.6 readFile . . . 54
5.17.1.7 readFolder . . . 54
5.17.1.8 replace . . . 55
5.17.1.9 writeFile . . . 55
5.18 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor_helper.hh File Reference 55
5.18.1 Macro Definition Documentation . . . 57
5.18.1.1 BIG_LINE . . . 57
5.18.1.2 BIG_TEXT . . . 57
5.18.2 Function Documentation . . . 57
5.18.2.1 _run . . . 57
5.18.2.2 extractRowMatchingPattern . . . 57
5.18.2.3 extractRowN . . . 58
5.18.2.4 extractWord . . . 58
5.18.2.5 ls . . . 58
5.18.2.6 readFile . . . 59
5.18.2.7 readFolder . . . 59
5.18.2.8 replace . . . 59
5.18.2.9 writeFile . . . 60
5.19 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/main.cpp File Reference . . . 60
5.19.1 Function Documentation . . . 60
5.19.1.1 main . . . 61
5.20 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/objectiveFunction.cpp File Reference . . 62
5.21 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/objectiveFunction.hh File Reference . . . 62
5.22 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParam_helper.cpp File Reference . . 63
5.22.1 Function Documentation . . . 63
5.22.1.1 parseArg . . . 63

5.22.1.2 Usage . . . 64
5.23 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParam_helper.hh File Reference . . . 64
5.23.1 Macro Definition Documentation . . . 66
5.23.1.1 ARGS . . . 66
5.23.1.2 CACHE . . . 66
5.23.1.3 DEBUG . . . 66
5.23.1.4 FILENAME . . . 66
5.23.1.5 GLOBAL_FO_CALCULATION . . . 66
5.23.1.6 LIST_LIMIT . . . 66
5.23.1.7 MAX_ITERATIONS . . . 66
5.23.1.8 NO . . . 66
5.23.1.9 NUM_N . . . 66
5.23.1.10 NUMBER . . . 66
5.23.1.11 SIMULATOR . . . 66
5.23.1.12 STRING . . . 66
5.23.1.13 YES . . . 66
5.23.1.14 YES_NO . . . 66
5.23.2 Function Documentation . . . 66
5.23.2.1 parseArg . . . 66
5.23.2.2 Usage . . . 67
5.24 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrparameters.cpp File Reference . . . 67
5.25 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParameters.hh File Reference . . . 68
5.25.1 Macro Definition Documentation . . . 68
5.25.1.1 DAGSIM . . . 69
5.25.1.2 LUNDSTROM . . . 69
5.26 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/read_app_file.cpp File Reference . . . 69
5.26.1 Function Documentation . . . 69
5.26.1.1 getfield . . . 69
5.26.1.2 readAppFile . . . 70
5.27 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/read_app_file.hh File Reference . . . 70
5.27.1 Macro Definition Documentation . . . 71
5.27.1.1 MAX_APP_LENGTH . . . 71
5.27.2 Function Documentation . . . 71
5.27.2.1 getfield . . . 72
5.27.2.2 readAppFile . . . 72
5.27.3 Variable Documentation . . . 72
5.27.3.1 _APP_ID . . . 72
5.27.3.2 _CHI_0 . . . 72
5.27.3.3 _CHI_C . . . 72
5.27.3.4 _D . . . 72

5.27.3.5 _Dsz . . . 73
5.27.3.6 _M . . . 73
5.27.3.7 _m . . . 73
5.27.3.8 _SESSION_APP_ID . . . 73
5.27.3.9 _St . . . 73
5.27.3.10 _V . . . 73
5.27.3.11 _v . . . 73
5.27.3.12 _W . . . 73
5.27.3.13 MAX_LINE_LENGTH . . . 73
5.27.3.14 PARAMETERS . . . 73
5.28 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/readConfigurationFile.cpp File Reference 73
5.28.1 Function Documentation . . . 74
5.28.1.1 extractItem . . . 74
5.28.1.2 readConfigurationFile . . . 74
5.29 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/readConfigurationFile.hh File Reference . 74
5.29.1 Typedef Documentation . . . 75
5.29.1.1 sConfiguration . . . 75
5.29.2 Function Documentation . . . 75
5.29.2.1 extractItem . . . 76
5.29.2.2 readConfigurationFile . . . 76
5.30 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/search.cpp File Reference . . . 76
5.31 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/search.hh File Reference . . . 77
5.32 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/Statistics.cpp File Reference . . . 78
5.32.1 Function Documentation . . . 79
5.32.1.1 addStatistics . . . 79
5.32.1.2 readStatistics . . . 79
5.33 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/statistics.hh File Reference . . . 79
5.33.1 Typedef Documentation . . . 80
5.33.1.1 sStatistics . . . 81
5.33.2 Function Documentation . . . 81
5.33.2.1 addStatistics . . . 81
5.33.2.2 readStatistics . . . 81
5.34 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/utility.cpp File Reference . . . 81
5.34.1 Function Documentation . . . 82
5.34.1.1 doubleCompare . . . 82
5.34.1.2 elapsedTime . . . 82
5.35 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/utility.hh File Reference . . . 83
5.35.1 Function Documentation . . . 84
5.35.1.1 doubleCompare . . . 84
5.35.1.2 elapsedTime . . . 84

5.35.2 Variable Documentation 84
5.35.2.1 epsilon 84
5.36 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/writeResults.cpp File Reference 85
5.36.1 Function Documentation 85
5.36.1.1 writeResults 85
5.37 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/writeResults.hh File Reference 86
5.37.1 Function Documentation 87
5.37.1.1 writeResults 87
5.38 /vagrant/PROJECT_SPARK/PACS_PROJECT/README.MD File Reference 88

Chapter 1

PACS_PROJECT

Program that manage soft deadline application when heavy load occurs. The program reassign the number of core and VM to each application. The project is already build in C and the goal is to re-write it in C++ with some parallelization (using MPI and openMP).

The original project is available at: https://github.com/eubr-bigsea/opt_jr

BUILD DOCUMENTATION:

run in the doc directory: doxygen opt_jr_doxy requirments to build documentation: doxygen and graphviz (sudo yum install ..)

Chapter 2

Class Index

2.1 Class List

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

Application 7
Batch 11
Bounds 15
Candidate 17
ObjFun 19
optJrParameters 21
Search 26
Statistic 27

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/vagrant/PROJECT_SPARK/PACS_PROJECT/README.MD . . . 88
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/appByWeight.cpp . . . 29
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/appByWeight.hh . . . 30
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/application.cpp . . . 31
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/application.hh . . . 32
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/batch.cpp . . . 32
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/batch.hh . . . 33
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/bounds.cpp . . . 34
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/bounds.hh . . . 34
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/candidates.cpp . . . 35
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/candidates.hh . . . 37
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/db.cpp . . . 39
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/db.hh . . . 42
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/debugmessage.cpp . . . 45
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/debugmessage.hh . . . 46
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor.cpp . . . 48
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor.hh . . . 49
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor_helper.cpp . . . 51
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor_helper.hh . . . 55
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/main.cpp . . . 60
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/objectiveFunction.cpp . . . 62
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/objectiveFunction.hh . . . 62
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParam_helper.cpp . . . 63
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParam_helper.hh . . . 64
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrparameters.cpp . . . 67
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParameters.hh . . . 68
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/read_app_file.cpp . . . 69
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/read_app_file.hh . . . 70
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/readConfigurationFile.cpp . . . 73
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/readConfigurationFile.hh . . . 74
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/search.cpp . . . 76
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/search.hh . . . 77
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/Statistics.cpp . . . 78
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/statistics.hh . . . 79
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/utility.cpp . . . 81
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/utility.hh . . . 83
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/writeResults.cpp . . . 85
/vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/writeResults.hh . . . 86

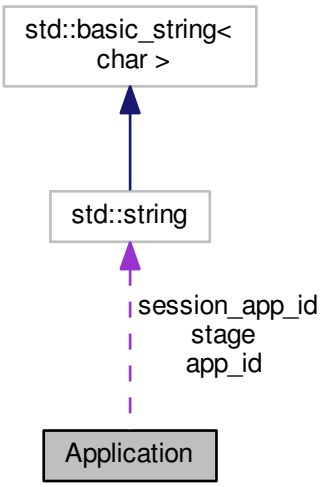
Chapter 4

Class Documentation

4.1 Application Class Reference

`#include <application.hh>`

Collaboration diagram for Application:



Public Member Functions

- Application (std::string session_app_id, std::string app_id, double w, double chi_0, double chi_C, double m, double M, double V, double v, double D, double csi, std::string St, int DatasetSize)

  Constructor expects all static values.

- void computeAlphaBeta (int nCores_n, double R_n)

  This function evaluates the Hyperbolic interpolation for alpha and beta (from the second time it is invoked).

Public Attributes

- int mode =R_ALGORITHM

  How the objective function is calculated (currently redundant)
- std::string session_app_id

  Session identifier.
- std::string app_id

  Application identifier.
- double w

  Weight application.
- double term_i

  Used to calculate nu index.
- double chi_0

  Machine learning parameter.
- double chi_C

  Machine learning parameter.
- double m

  Ram of a container for this application.
- double M

  Total Ram available at the YARN NodeManager.
- double V

  Total vCPUs available at the YARN NodeManager.
- double v

  vCPUs of a container for this application
- double Deadline_d

  Deadline for the application.
- double csi
- std::string stage

  Application's stage (used in case of residual time)
- int datasetSize

  Size of the dataset.
- double nu_d

  nu value
- int currentCores_d

  Initialized to nu_i.
- int nCores_DB_d

  Initialized to the value from look-up table.
- int bound

  Bound (number of cores)
- double R_d

  Value of R as per the predictor.
- double baseFO

  base FO value (used to calculate the delta)
- double initialBaseFO

  copy of base FO value (used to reset the value)
- int boundIterations

  Metrics.
- int vm

  Read from OPTIMIZER_CONFIGURATION_TABLE.
- double alpha

  First parameter for Hyperbolic interpolation.

- double beta

  Second parameter for Hyperbolic interpolation.
- int index =0

  Index for Hyperbolic interpolation.

4.1.1 Detailed Description

In the Application class all the data of one application are stored ; it's provided also a method to evaluate Hyperbolic interpolation for alpha and beta.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Application::Application (std::string *session_app_id,* std::string *app_id,* double *w,* double *chi_0,* double *chi_C,* double *m,* double *M,* double *V,* double *v,* double *D,* double *csi,* std::string *St,* int *DatasetSize*)

Constructor expects all static values.

4.1.3 Member Function Documentation

4.1.3.1 void Application::computeAlphaBeta (int *nCores_n,* double *R_n*)

This function evaluates the Hyperbolic interpolation for alpha and beta (from the second time it is invoked).

4.1.4 Member Data Documentation

4.1.4.1 double Application::alpha

First parameter for Hyperbolic interpolation.

4.1.4.2 std::string Application::app_id

Application identifier.

4.1.4.3 double Application::baseFO

base FO value (used to calculate the delta)

4.1.4.4 double Application::beta

Second parameter for Hyperbolic interpolation.

4.1.4.5 int Application::bound

Bound (number of cores)

4.1.4.6 int Application::boundIterations

Metrics.

4.1.4.7 double Application::chi_0

Machine learning parameter.

4.1.4.8 double Application::chi_C

Machine learning parameter.

4.1.4.9 double Application::csi

4.1.4.10 int Application::currentCores_d

Initialized to nu_i.

4.1.4.11 int Application::datasetSize

Size of the dataset.

4.1.4.12 double Application::Deadline_d

Deadline for the application.

4.1.4.13 int Application::index =0

Index for Hyperbolic interpolation.

4.1.4.14 double Application::initialBaseFO

copy of base FO value (used to reset the value)

4.1.4.15 double Application::m

Ram of a container for this application.

4.1.4.16 double Application::M

Total Ram available at the YARN NodeManager.

4.1.4.17 int Application::mode =R_ALGORITHM

How the objective function is calculated (currently redundant)

4.1.4.18 int Application::nCores_DB_d

Initialized to the value from look-up table.

4.1.4.19 double Application::nu_d

nu value

4.1.4.20 double Application::R_d

Value of R as per the predictor.

4.1.4.21 std::string Application::session_app_id

Session identifier.

4.1.4.22 std::string Application::stage

Application's stage (used in case of residual time)

4.1.4.23 double Application::term_i

Used to calculate nu index.

4.1.4.24 double Application::V

Total vCPUs available at the YARN NodeManager.

4.1.4.25 double Application::v

vCPUs of a container for this application

4.1.4.26 int Application::vm

Read from OPTIMIZER_CONFIGURATION_TABLE.

4.1.4.27 double Application::w

Weight application.

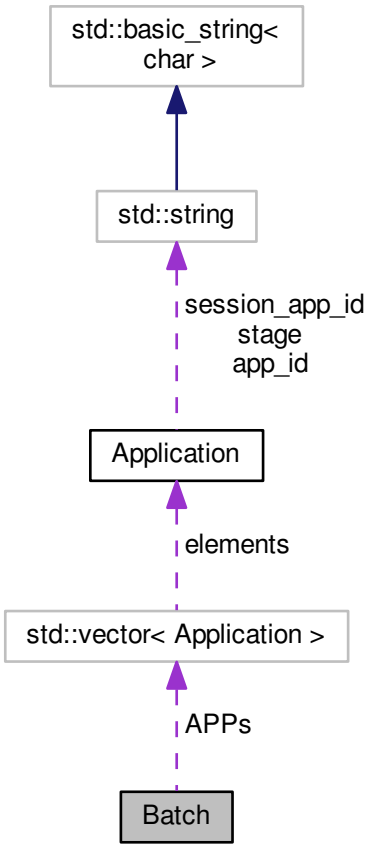
The documentation for this class was generated from the following files:

- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/application.hh
- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/application.cpp

4.2 Batch Class Reference

`#include <batch.hh>`

Collaboration diagram for Batch:



Public Member Functions

- Batch (std::vector< Application > apps)

  Constructor expects a vector of application which should be given by the "readAppFile" function declared in "read_app_file.hh".
- void calculate_nu (optJrParameters &par)

  It calculates nu indices for each application and stores it in each "Application" object.
- void initialize (sConfiguration &configuration, MYSQL *conn, optJrParameters &par)

  For each application, a base value for the objective function is calculated.
- void fixInitialSolution (optJrParameters &par)

  It fixes the initial solution by reallocating the residual cores to the applications that may need more resources.

Public Attributes

- std::vector< Application > APPs

  The vector stores application data.

4.2.1 Detailed Description

This class manages the applications; it stores applications data in a vector and it provides methods useful to apply before executing the localSearch

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Batch::Batch (std::vector< Application > *apps*) [inline]

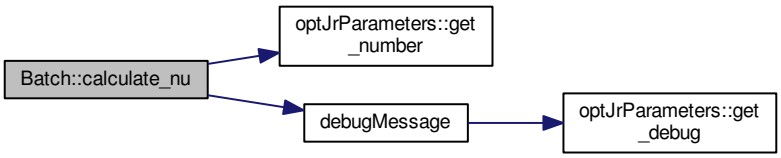
Constructor expects a vector of application which should be given by the "readAppFile" function declared in "read_app_file.hh".

4.2.3 Member Function Documentation

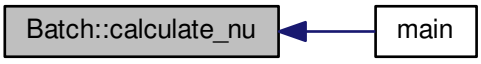
4.2.3.1 void Batch::calculate_nu (optJrParameters & *par*)

It calculates nu indices for each application and stores it in each "Application" object.

Here is the call graph for this function:



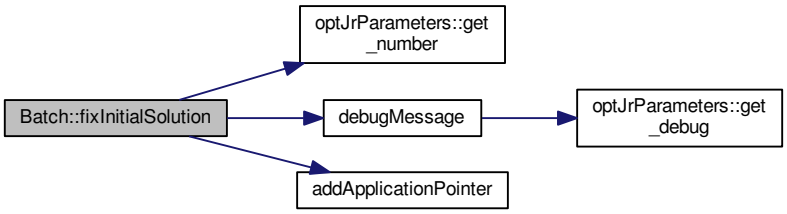
Here is the caller graph for this function:



4.2.3.2 void Batch::fixInitialSolution (optJrParameters & *par*)

It fixes the initial solution by reallocating the residual cores to the applications that may need more resources.

Here is the call graph for this function:

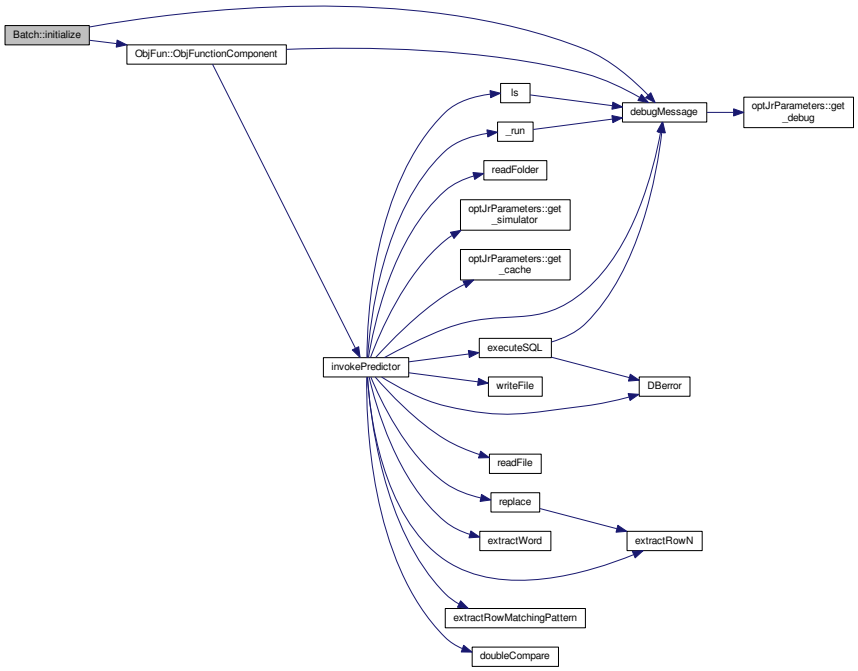


Here is the caller graph for this function:

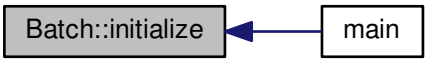
4.2.3.3 void Batch::initialize (sConfiguration & *configuration,* MYSQL * *conn,* optJrParameters & *par*)

For each application, a base value for the objective function is calculated.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.4 Member Data Documentation

4.2.4.1 std::vector<Application> Batch::APPs

The vector stores application data.

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

- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/batch.hh
- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/batch.cpp

4.3 Bounds Class Reference

`#include <bounds.hh>`

Static Public Member Functions

- static void calculateBounds (Batch &app_manager, sConfiguration &configuration, MYSQL ∗conn, optJrParameters &par)

4.3.1 Detailed Description

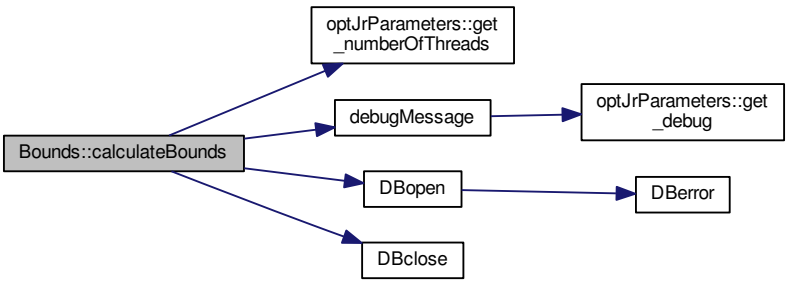
Bounds class provide a method to evaluate the bound for the applications in BATCH i.e. the minimal number of cores necessary to finish the execution before the deadline

4.3.2 Member Function Documentation

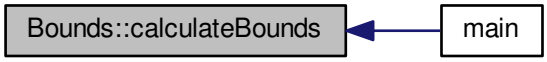
4.3.2.1 void Bounds::calculateBounds (Batch & *app_manager,* sConfiguration & *configuration,* MYSQL ∗ *conn,* optJrParameters & *par*) [static]

calculateBounds evaluates the bound for the applications in BATCH i.e. the minimal number of cores necessary to finish the execution before the deadline. The function looks before if the result is already stored in the database, otherwise it invokes the predictor doing a "HILL CLIMBING". If the number of threads in the configuration file is greater than 0, it does the computations in parallel (using openMP).

Here is the call graph for this function:



Here is the caller graph for this function:



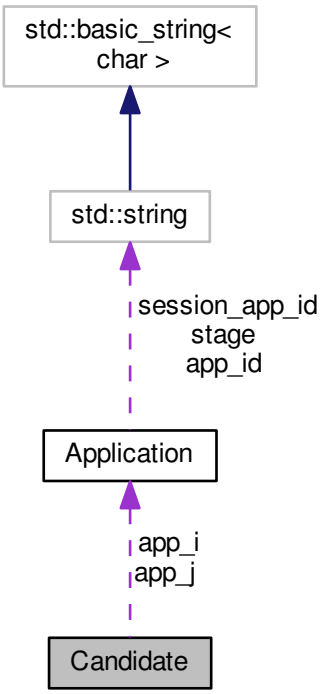
The documentation for this class was generated from the following files:

- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/bounds.hh
- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/bounds.cpp

4.4 Candidate Class Reference

`#include <candidates.hh>`

Collaboration diagram for Candidate:



Public Member Functions

- Candidate (Application *i, Application *j, int NCi, int NCj, double D_FO, int d_i, int d_j)

  Constructor.

Public Attributes

- Application * app_i

  Pointer to the first Application.
- Application * app_j

  Pointer to the second Application.
- int newCoreAssignment_i

  Cores after the move (first application)
- int newCoreAssignment_j

  Cores after the move (second application)
- double deltaFO

  Delta Objective Function following the move.
- int delta_i

  Delta cores following the move (first application)

- int delta_j

  Delta cores following the move (second application)
- double real_i

  Real predictor value calculated (MPI) after the interpolation (first application)
- double real_j

  Real predictor value calculated (MPI) after the interpolation (second application)

4.4.1 Detailed Description

Candidate class is an auxiliary class used by localSearch; it stores data about pairs of application and the consequent changes on the objective function after cores exchange.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Candidate::Candidate (Application * *i*, Application * *j*, int *NCi*, int *NCj*, double *D_FO*, int *d_i*, int *d_j*) [inline]

Constructor.

4.4.3 Member Data Documentation

4.4.3.1 Application* Candidate::app_i

Pointer to the first Application.

4.4.3.2 Application* Candidate::app_j

Pointer to the second Application.

4.4.3.3 int Candidate::delta_i

Delta cores following the move (first application)

4.4.3.4 int Candidate::delta_j

Delta cores following the move (second application)

4.4.3.5 double Candidate::deltaFO

Delta Objective Function following the move.

4.4.3.6 int Candidate::newCoreAssignment_i

Cores after the move (first application)

4.4.3.7 int Candidate::newCoreAssignment_j

Cores after the move (second application)

4.4.3.8 double Candidate::real_i

Real predictor value calculated (MPI) after the interpolation (first application)

4.4.3.9 double Candidate::real_j

Real predictor value calculated (MPI) after the interpolation (second application)

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

- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/candidates.hh

4.5 ObjFun Class Reference

`#include <objectiveFunction.hh>`

Static Public Member Functions

- static double ObjFunctionComponent (sConfiguration &configuration, MYSQL ∗conn, Application &app, optJrParameters &par)
- static double ObjFunctionComponentApprox (Application &App, optJrParameters &par)
- static double ObjFunctionGlobal (sConfiguration &configuration, MYSQL ∗conn, Batch &App_manager, optJrParameters &par)

4.5.1 Detailed Description

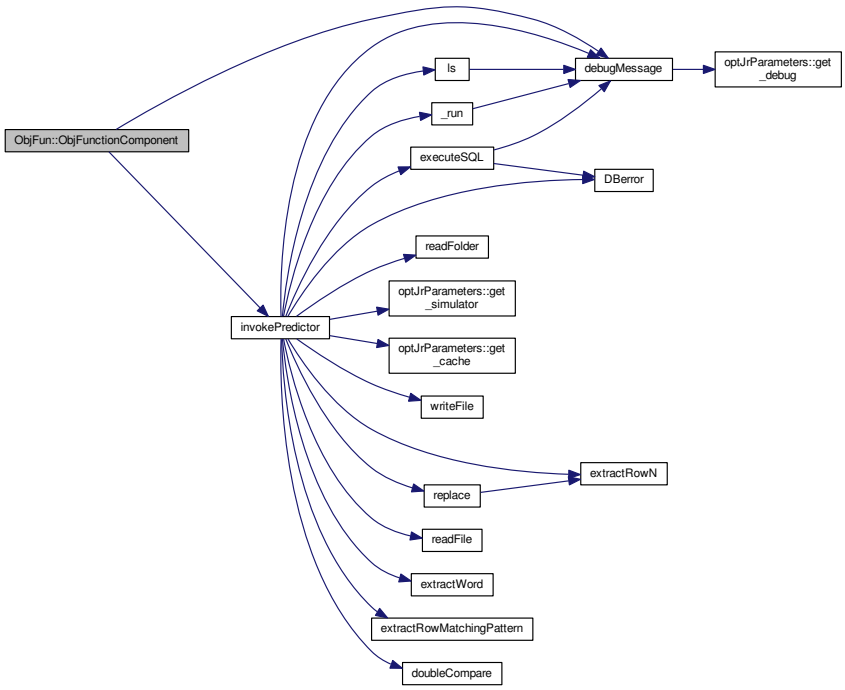
This class provides methods to evaluate the objective function in different ways

4.5.2 Member Function Documentation

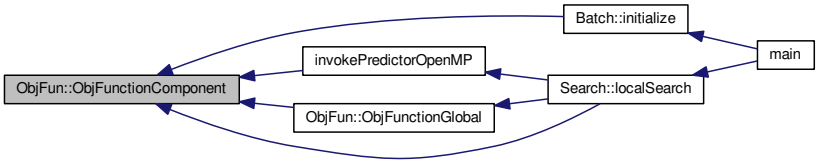
4.5.2.1 double ObjFun::ObjFunctionComponent (sConfiguration & *configuration,* MYSQL ∗ *conn,* Application & *app,* optJrParameters & *par*) [static]

ObjFunctionComponent evaluates the contribution to the calculation of the objective function of one application. Currently, only one method is supported. Note that the algorithm's choice is stored in the "mode" field of the application structure.

Here is the call graph for this function:



Here is the caller graph for this function:

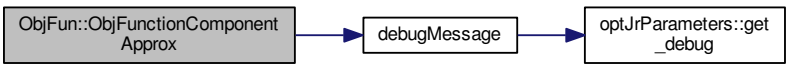


4.5.2.2 double ObjFun::ObjFunctionComponentApprox (Application & *App,* optJrParameters & *par*) [static]

ObjFunctionComponentApprox computes an approximation of the objective function (and update R_d)

Name: ObjFunctionComponentApprox Output parameters: a double The value of the approximated objective function Description It computes an approximation of the objective function (and update R_d)

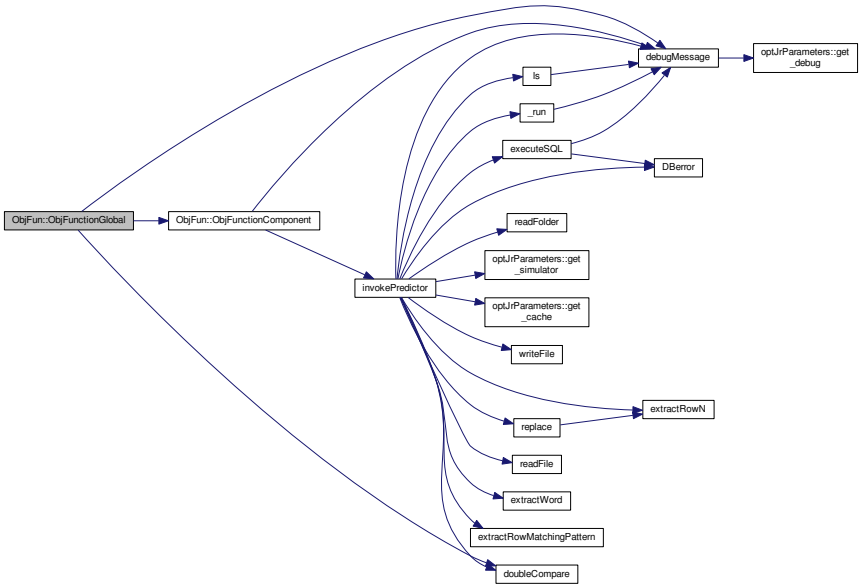
Here is the call graph for this function:



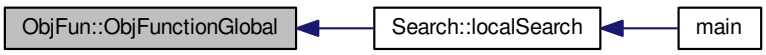
4.5.2.3 double ObjFun::ObjFunctionGlobal (sConfiguration & *configuration,* MYSQL * *conn,* Batch & *App_manager,* optJrParameters & *par*) [static]

ObjFunctionGlobal computes the value of the total objective function

Here is the call graph for this function:



Here is the caller graph for this function:



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

- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/objectiveFunction.hh
- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/objectiveFunction.cpp

4.6 optJrParameters Class Reference

`#include <optjrParameters.hh>`

Public Member Functions

- optJrParameters (char **args, int argc)

  The constructor takes in input all the input from command line.
- void set_numberOfThreads (sConfiguration &configuration)

Set the number of threads: it looks in configuration file (0== "no parallelization")

- const std::string get_filename ()

  Returns the name of the file with applications.

- const int get_debug ()

  Returns the debug option (1==YES, 0==NO)

- const int get_cache ()

  Returns the cache option (1==YES, 0==NO)

- const int get_globalFOcalculation ()

  Returns the option globalFOcalculation (1==YES, 0==NO)

- const int get_K ()

  Returns K.

- const int get_simulator ()

  Returns which simulator is used.

- const int get_number ()

  Returns the available number of cores.

- const int get_maxIteration ()

  Returns the maximum number of iteration for localSearch.

- const int get_numberOfThreads ()

  Returns the number of threads to use in parallelization; if it is 0 there is no parallelization.

4.6.1 Detailed Description

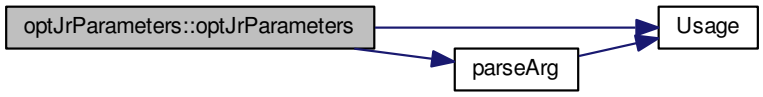
optJrParameters saves parameters received from command line; once they are saved they are visible with public get_∗() functions

4.6.2 Constructor & Destructor Documentation

4.6.2.1 optJrParameters::optJrParameters (char ∗∗ *args*, int *argc*)

The constructor takes in input all the input from command line.

Here is the call graph for this function:

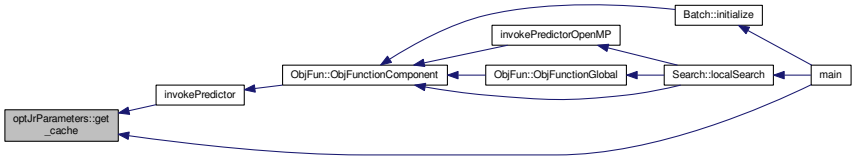


4.6.3 Member Function Documentation

4.6.3.1 const int optJrParameters::get_cache ( )

Returns the cache option (1==YES, 0==NO)

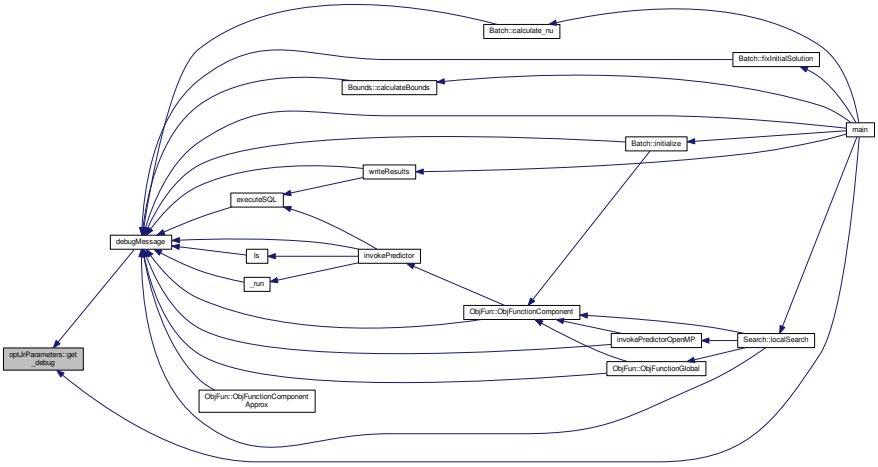
Here is the caller graph for this function:



4.6.3.2 const int optJrParameters::get_debug ( )

Returns the debug option (1==YES, 0==NO)

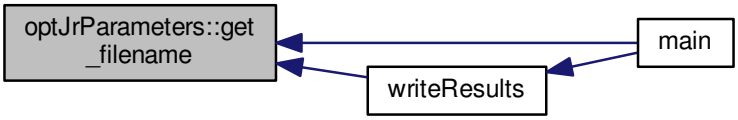
Here is the caller graph for this function:



4.6.3.3 const std::string optJrParameters::get_filename ( )

Returns the name of the file with applications.

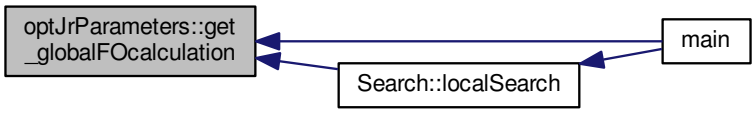
Here is the caller graph for this function:



4.6.3.4 const int optJrParameters::get_globalFOcalculation ( )

Returns the option globalFOcalculation (1==YES, 0==NO)

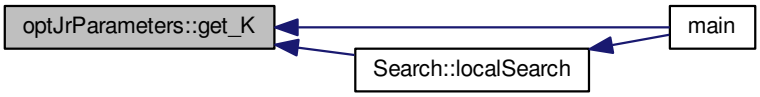
Here is the caller graph for this function:



4.6.3.5 const int optJrParameters::get_K ( )

Returns K.

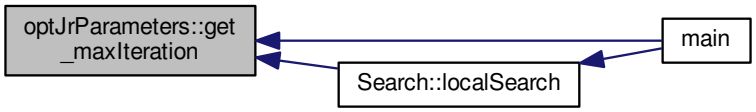
Here is the caller graph for this function:



4.6.3.6 const int optJrParameters::get_maxIteration ( )

Returns the maximum number of iteration for localSearch.

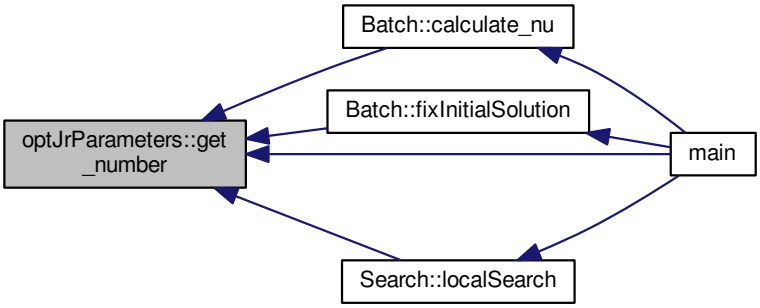
Here is the caller graph for this function:



4.6.3.7 const int optJrParameters::get_number ( )

Returns the available number of cores.

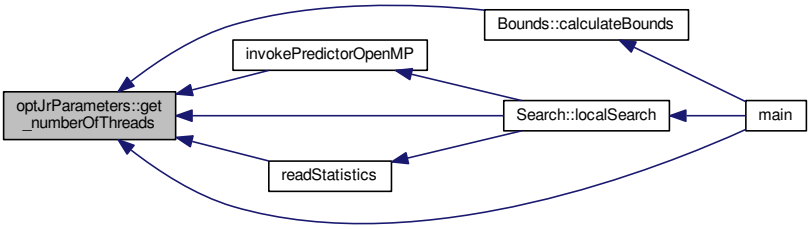
Here is the caller graph for this function:



4.6.3.8 const int optJrParameters::get_numberOfThreads ( )

Returns the number of threads to use in parallelization; if it is 0 there is no parallelization.

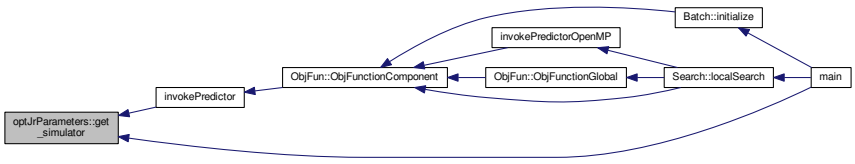
Here is the caller graph for this function:



4.6.3.9 const int optJrParameters::get_simulator ( )

Returns which simulator is used.

Here is the caller graph for this function:



4.6.3.10 void optJrParameters::set_numberOfThreads (sConfiguration & *configuration*)

Set the number of threads: it looks in configuration file (0== "no parallelization")

Here is the caller graph for this function:

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

- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParameters.hh
- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrparameters.cpp

4.7 Search Class Reference

`#include <search.hh>`

Static Public Member Functions

- static void localSearch (sConfiguration &configuration, MYSQL ∗conn, Batch &App_manager, optJrParameters &par)

4.7.1 Detailed Description

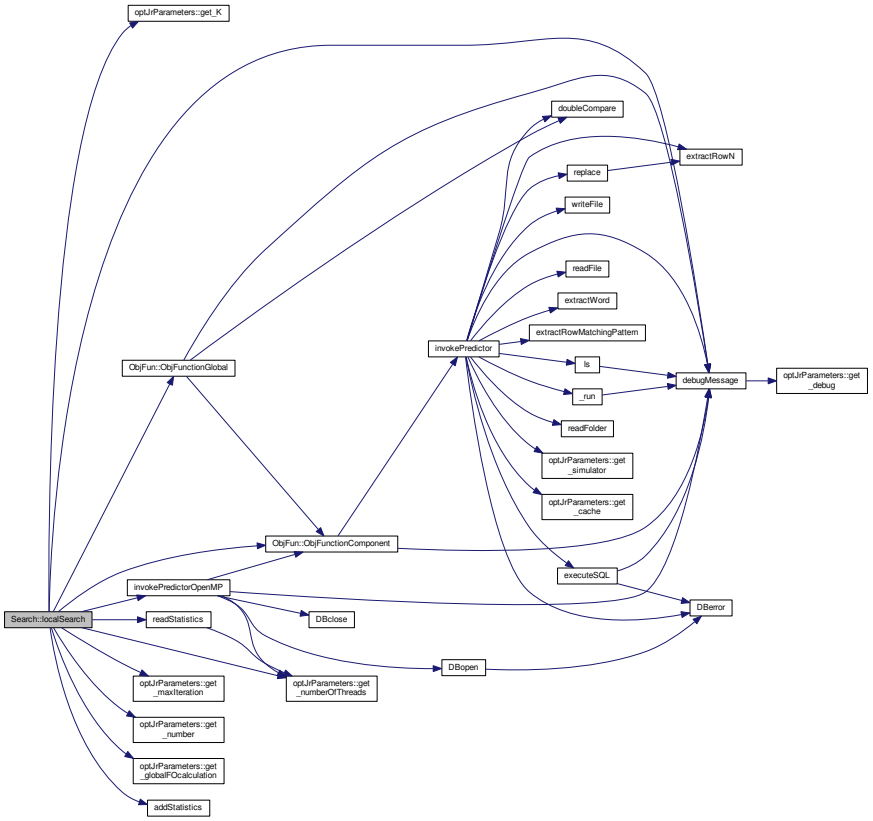
"Search" class provides methods to find a solution minimizing the objective function. Actually only one method is supported.

4.7.2 Member Function Documentation

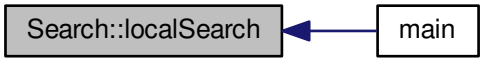
4.7.2.1 void Search::localSearch (sConfiguration & *configuration,* MYSQL ∗ *conn,* Batch & *App_manager,* optJrParameters & *par*) [static]

localSearch perform a local search of a solution minimizing the objective function; it performs cores exchanges between pairs of application and chooses the best pair. The search stops when no improvements are possible or the maximum number of iteration is reached. The function looks before at approximated values of objective function and then for the potential best pairs it invokes the predictor.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/search.hh
- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/search.cpp

4.8 Statistic Class Reference

`#include <statistics.hh>`

Public Member Functions

- Statistic (int iter, int s, double FO)

- int get_iteration ()
- int get_size ()
- double get_FO_Total ()

4.8.1 Detailed Description

Statistic includes relevant statistical information about a single iteration in localSearch

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Statistic::Statistic (int *iter,* int *s,* double *FO*) [inline]

4.8.3 Member Function Documentation

4.8.3.1 double Statistic::get_FO_Total ( )

4.8.3.2 int Statistic::get_iteration ( )

4.8.3.3 int Statistic::get_size ( )

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

- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/statistics.hh
- /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/Statistics.cpp

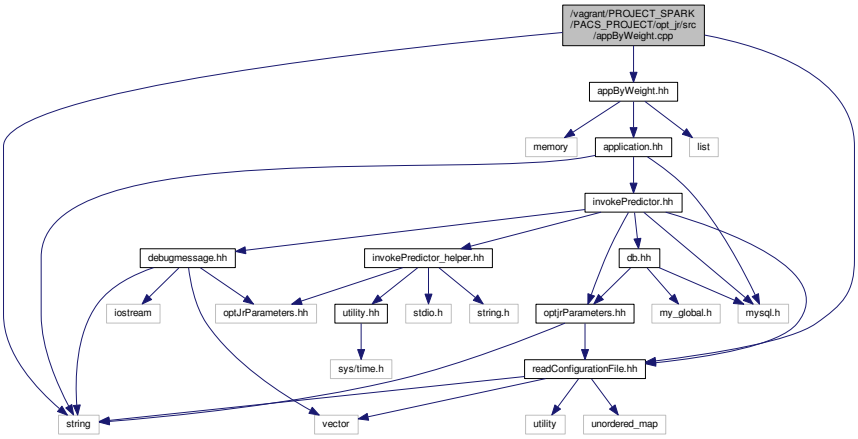
Chapter 5

File Documentation

5.1 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/appByWeight.cpp File Reference

```
#include "appByWeight.hh"
```

Include dependency graph for appByWeight.cpp:



Functions

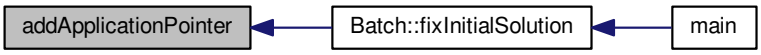
- void addApplicationPointer (appByWeight &LP, Application &App)

5.1.1 Function Documentation

5.1.1.1 void addApplicationPointer (appByWeight & *LP*, Application & *App*)

It saves Application∗ in decreasing weight "w" order

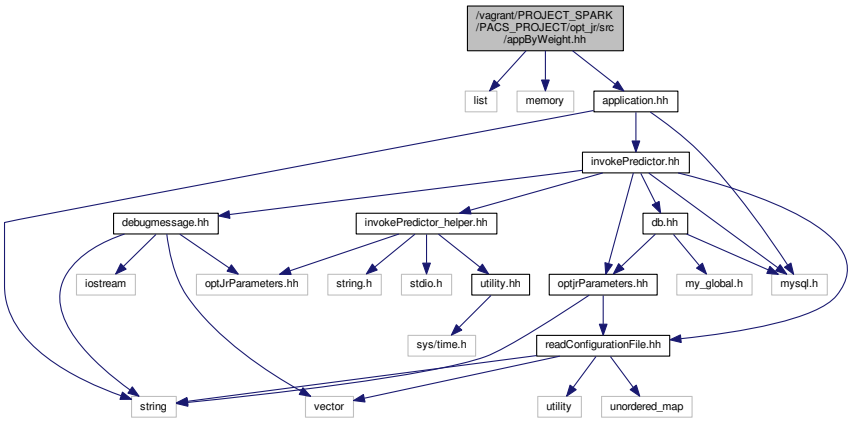
Here is the caller graph for this function:



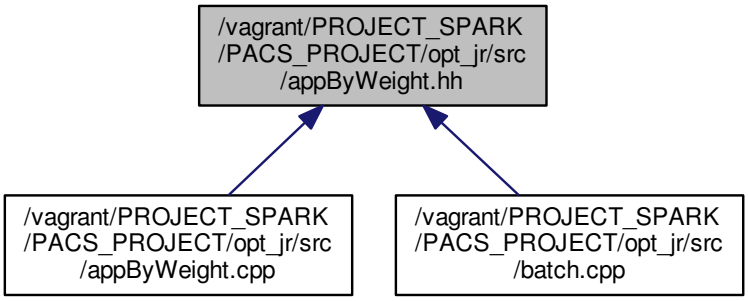
5.2 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/appByWeight.hh File Reference

```
#include <list>
#include <memory>
#include "application.hh"
```

Include dependency graph for appByWeight.hh:



This graph shows which files directly or indirectly include this file:



Typedefs

- using appByWeight = std::list< Application * >

Functions

- void addApplicationPointer (appByWeight &LP, Application &App)

5.2.1 Typedef Documentation

5.2.1.1 using appByWeight = std::list< Application* >

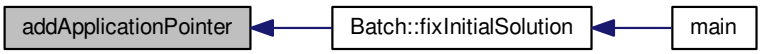
appByWeight is an auxiliary list of Application* used by fixInitialSolution; through the addApplicationPointer it stores Application* ordered by weight "w"

5.2.2 Function Documentation

5.2.2.1 void addApplicationPointer (appByWeight & *LP*, Application & *App*)

It saves Application* in decreasing weight "w" order

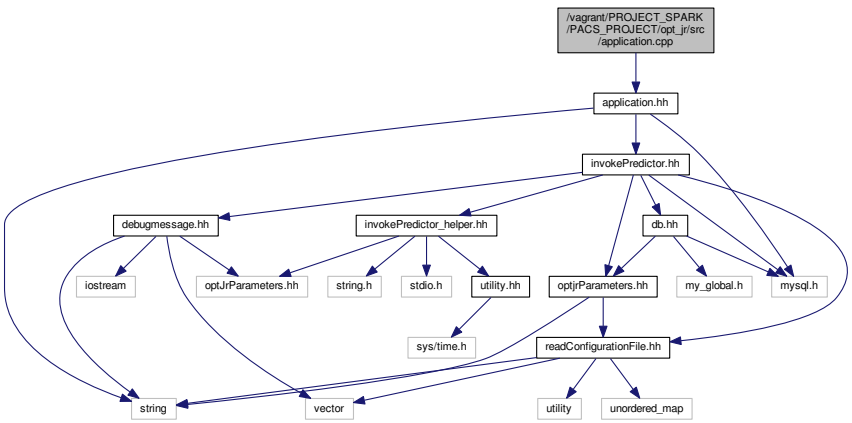
Here is the caller graph for this function:



5.3 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/application.cpp File Reference

```
#include "application.hh"
```

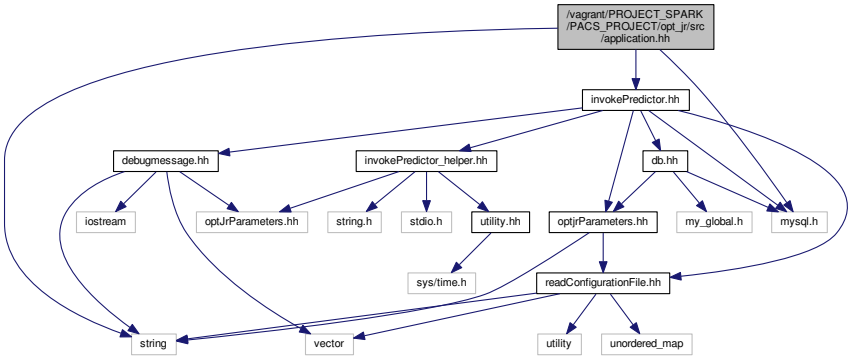
Include dependency graph for application.cpp:



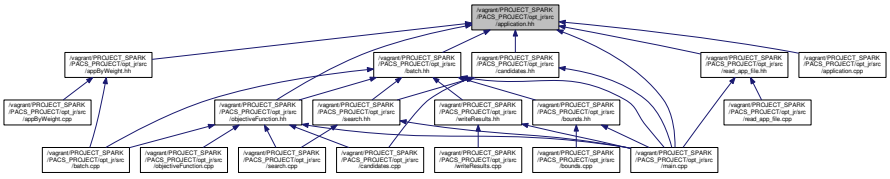
5.4 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/application.hh File Reference

```
#include <string>
#include <mysql.h>
#include "invokePredictor.hh"
```

Include dependency graph for application.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class Application

Macros

- #define R_ALGORITHM 0

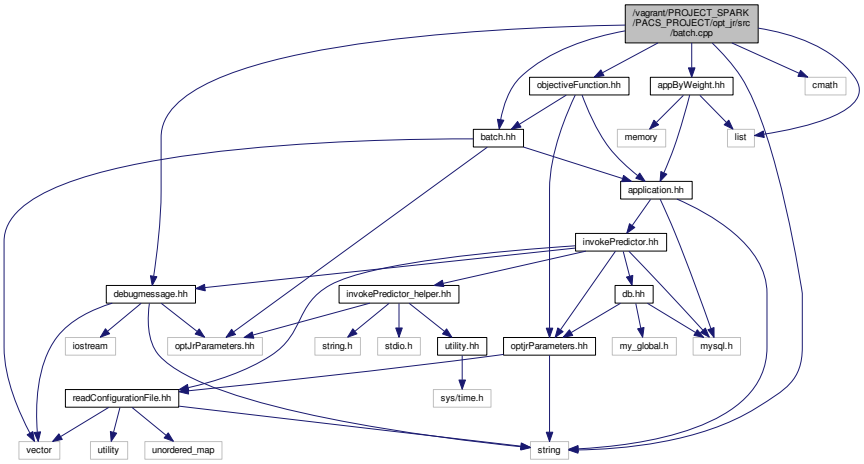
5.4.1 Macro Definition Documentation

5.4.1.1 #define R_ALGORITHM 0

5.5 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/batch.cpp File Reference

```
#include "batch.hh"
#include "debugmessage.hh"
#include "objectiveFunction.hh"
#include "appByWeight.hh"
#include <string>
#include <cmath>
#include <list>
```

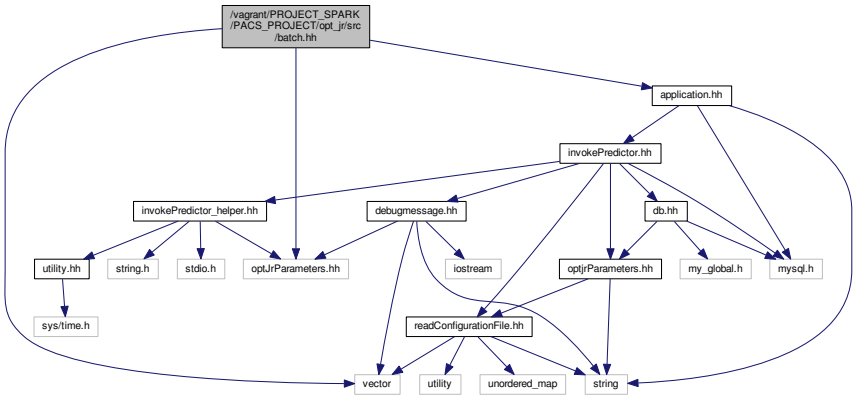
Include dependency graph for batch.cpp:



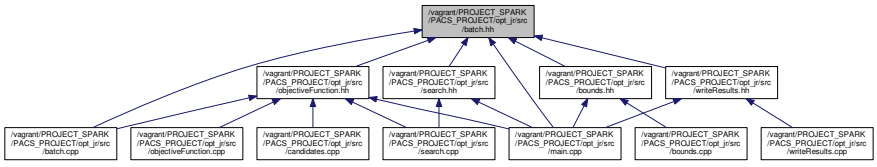
5.6 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/batch.hh File Reference

```
#include <vector>
#include "optJrParameters.hh"
#include "application.hh"
```

Include dependency graph for batch.hh:



This graph shows which files directly or indirectly include this file:



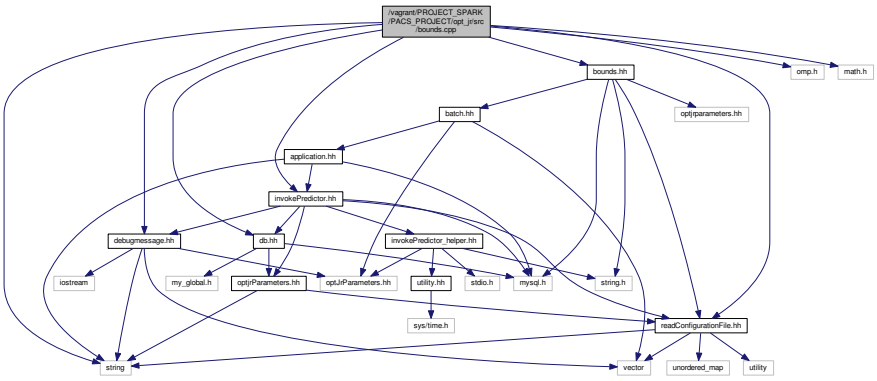
Classes

- class Batch

5.7 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/bounds.cpp File Reference

```
#include "bounds.hh"
#include "debugmessage.hh"
#include "db.hh"
#include "invokePredictor.hh"
#include <omp.h>
#include <math.h>
```

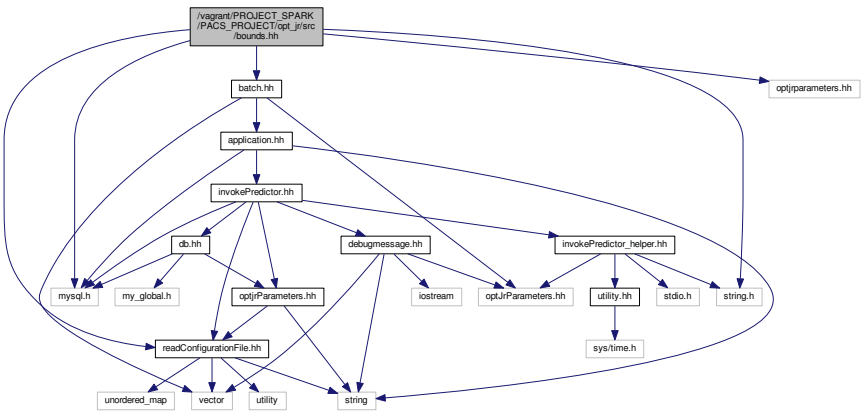
Include dependency graph for bounds.cpp:



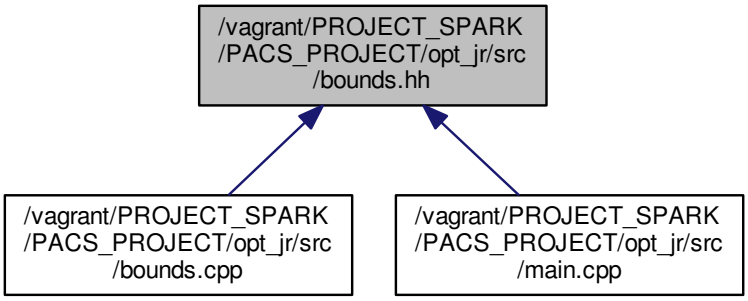
5.8 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/bounds.hh File Reference

```
#include "batch.hh"
#include "readConfigurationFile.hh"
#include "optjrparameters.hh"
#include <mysql.h>
#include <string.h>
```

Include dependency graph for bounds.hh:



This graph shows which files directly or indirectly include this file:



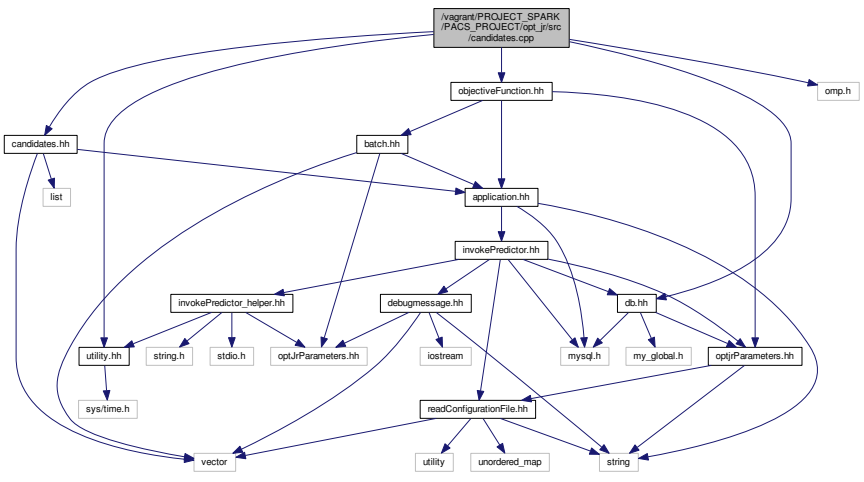
Classes

- class Bounds

5.9 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/candidates.cpp File Reference

```
#include "candidates.hh"
#include "utility.hh"
#include "db.hh"
#include "objectiveFunction.hh"
#include <omp.h>
```

Include dependency graph for candidates.cpp:



Functions

- void addCandidate (sCandidates &cand, Application &app_i, Application &app_j, int contr1, int contr2, double delta, double delta_i, double delta_j)
- void invokePredictorOpenMP (sCandidates &candidates, optJrParameters &par, sConfiguration &configuration)

5.9.1 Function Documentation

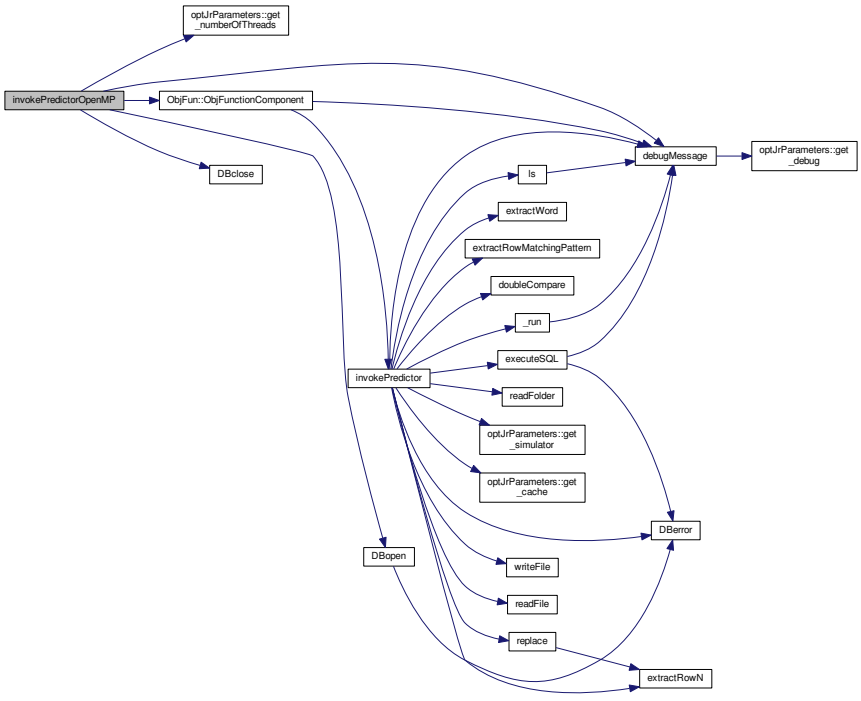
5.9.1.1 void addCandidate (sCandidates & *cand,* Application & *app_i,* Application & *app_j,* int *contr1,* int *contr2,* double *delta,* double *delta_i,* double *delta_j*)

"addCandidate" stores build a "Candidate" object and stores it in a sCandidates container ordered by increasing delta FO

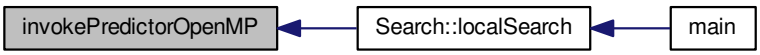
5.9.1.2 void invokePredictorOpenMP (sCandidates & *candidates,* optJrParameters & *par,* sConfiguration & *configuration*)

"invokePredictorOpenMP" calls in parallel the ObjFunctionComponent for each pair of application and it stores the results for each pair in real_i, real_j.

Here is the call graph for this function:



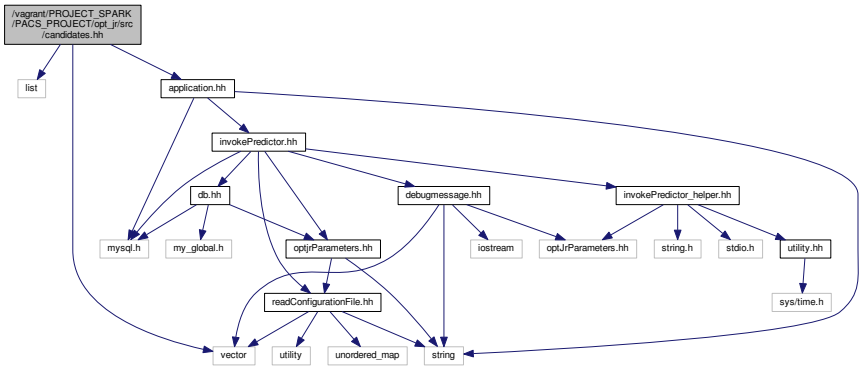
Here is the caller graph for this function:



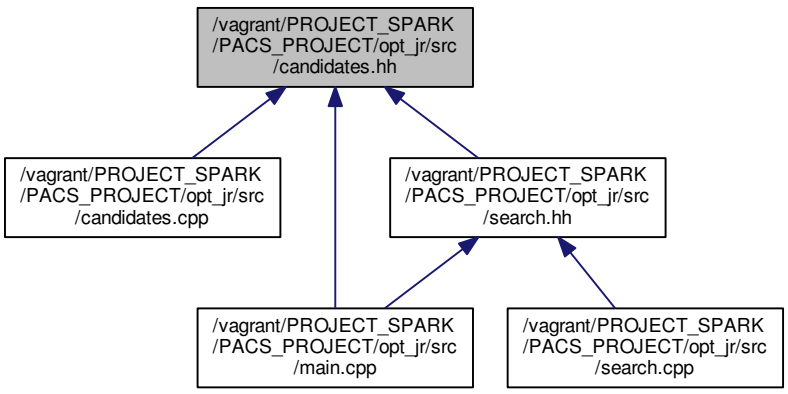
5.10 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/candidates.hh File Reference

```
#include <list>
#include <vector>
#include "application.hh"
```

Include dependency graph for candidates.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class Candidate

Typedefs

- using sCandidates = std::list< Candidate >

Functions

- void addCandidate (sCandidates &cand, Application &app_i, Application &app_j, int contr1, int contr2, double delta, double delta_i, double delta_j)
- void invokePredictorOpenMP (sCandidates &candidates, optJrParameters &par, sConfiguration &configuration)

5.10.1 Typedef Documentation

5.10.1.1 using sCandidates = std::list<Candidate>

List container used in localSearch to store Candidates with increasing deltaFO

5.10.2 Function Documentation

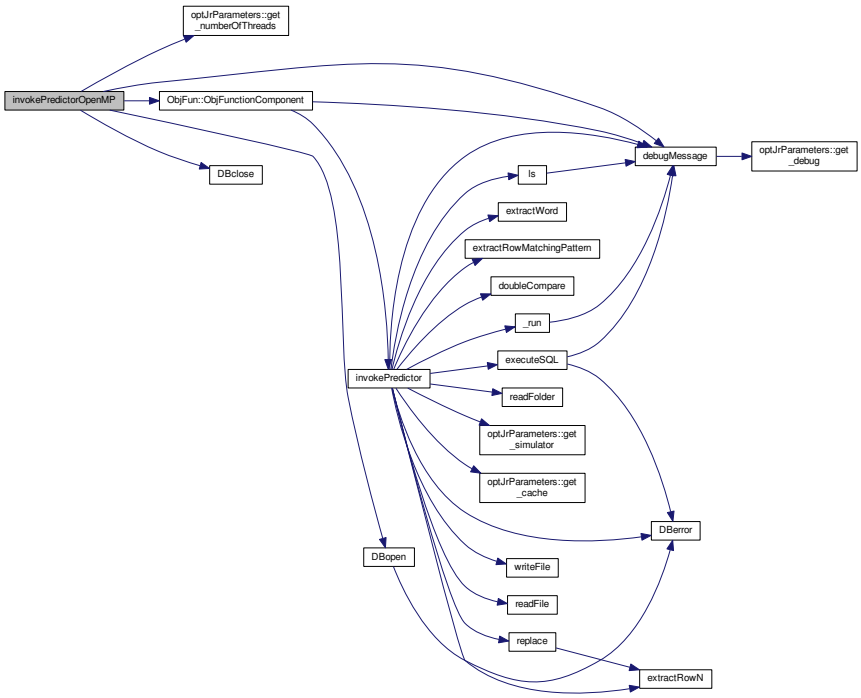
5.10.2.1 void addCandidate (sCandidates & *cand,* Application & *app_i,* Application & *app_j,* int *contr1,* int *contr2,* double *delta,* double *delta_i,* double *delta_j*)

"addCandidate" stores build a "Candidate" object and stores it in a sCandidates container ordered by increasing delta FO

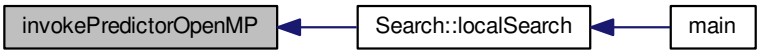
5.10.2.2 void invokePredictorOpenMP (sCandidates & *candidates,* optJrParameters & *par,* sConfiguration & *configuration*)

"invokePredictorOpenMP" calls in parallel the ObjFunctionComponent for each pair of application and it stores the results for each pair in real_i, real_j.

Here is the call graph for this function:



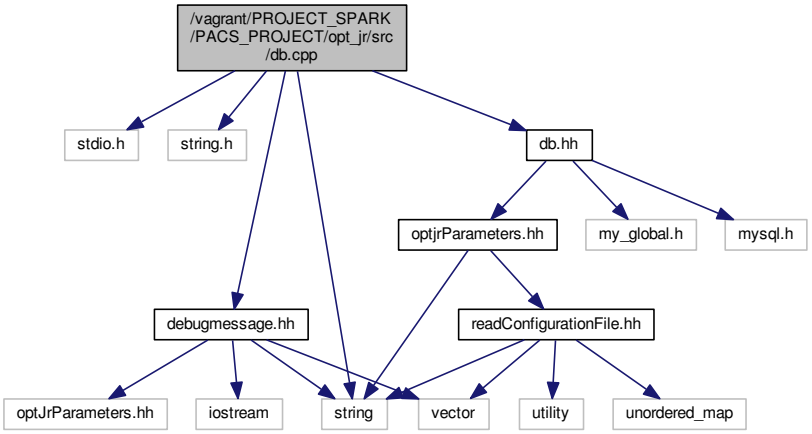
Here is the caller graph for this function:



5.11 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/db.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <string>
#include "db.hh"
#include "debugmessage.hh"
```

Include dependency graph for db.cpp:



Functions

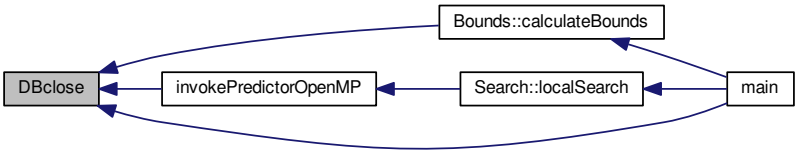
- void DBerror (MYSQL ∗conn, char ∗msg)
- MYSQL_ROW executeSQL (MYSQL ∗conn, char ∗statement, optJrParameters par)
- MYSQL ∗ DBopen (char ∗host, char ∗port, char ∗login, char ∗passw, char ∗dbName)
- void DBclose (MYSQL ∗conn)

5.11.1 Function Documentation

5.11.1.1 void DBclose (MYSQL ∗ *conn*)

Close DB connection (not substantially changed from original C version)

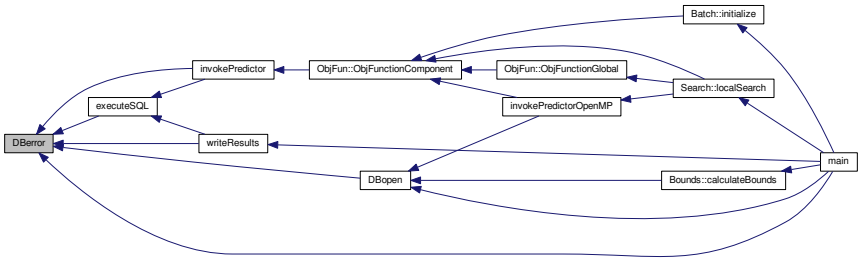
Here is the caller graph for this function:



5.11.1.2 void DBerror (MYSQL ∗ *conn,* char ∗ *msg*)

Standard error procedure for DB operations (not substantially changed from original C version)

Here is the caller graph for this function:

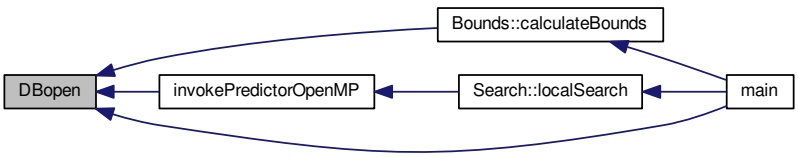


5.11.1.3 MYSQL* DBopen (char * *host,* char * *port,* char * *login,* char * *passw,* char * *dbName*)

Open a DB connection (not substantially changed from original C version)

Here is the call graph for this function:

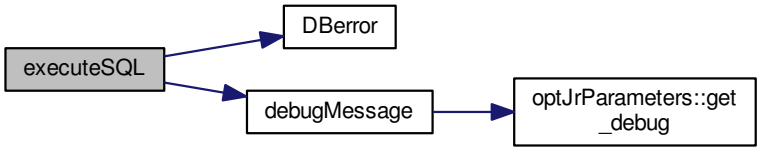
Here is the caller graph for this function:



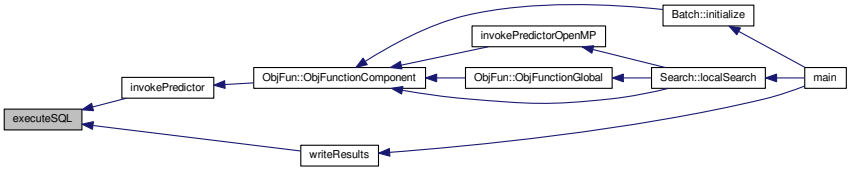
5.11.1.4 MYSQL_ROW executeSQL (MYSQL * *conn,* char * *statement,* optJrParameters *par*)

Execute SQL statement (not substantially changed from original C version)

Here is the call graph for this function:



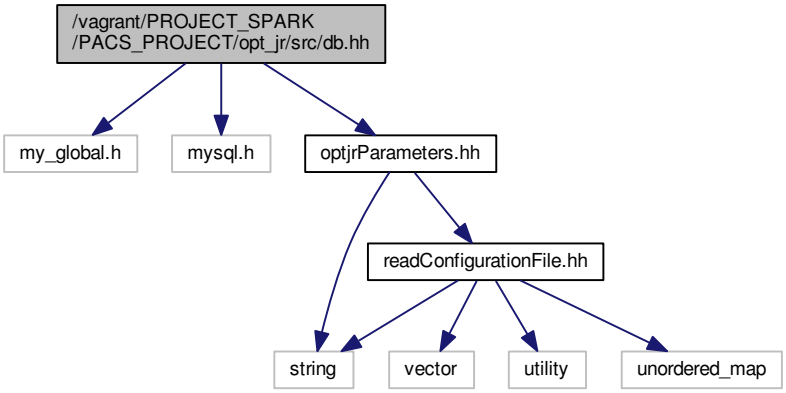
Here is the caller graph for this function:



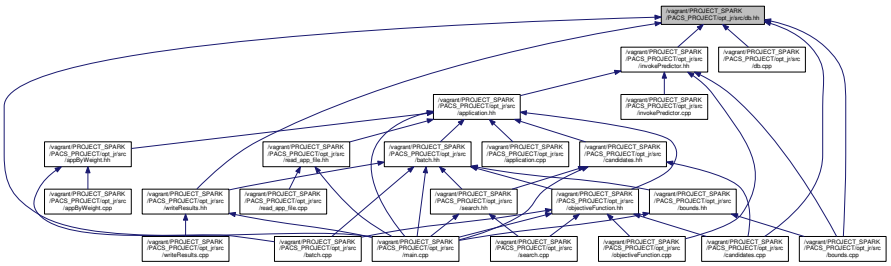
5.12 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/db.hh File Reference

```
#include <my_global.h>
#include <mysql.h>
#include "optjrParameters.hh"
```

Include dependency graph for db.hh:



This graph shows which files directly or indirectly include this file:



Functions

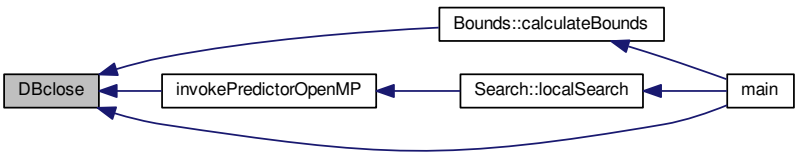
- void DBerror (MYSQL ∗conn, char ∗msg)
- MYSQL_ROW executeSQL (MYSQL ∗conn, char ∗statement, optJrParameters par)
- MYSQL ∗ DBopen (char ∗host, char ∗port, char ∗login, char ∗passw, char ∗dbName)
- void DBclose (MYSQL ∗conn)

5.12.1 Function Documentation

5.12.1.1 void DBclose (MYSQL ∗ *conn*)

Close DB connection (not substantially changed from original C version)

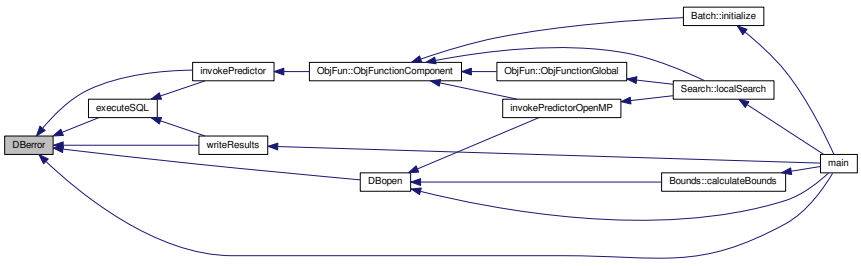
Here is the caller graph for this function:



5.12.1.2 void DBerror (MYSQL ∗ *conn*, char ∗ *msg*)

Standard error procedure for DB operations (not substantially changed from original C version)

Here is the caller graph for this function:

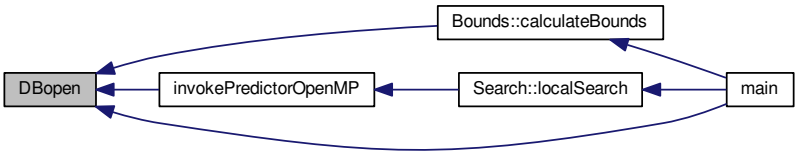


5.12.1.3 MYSQL* DBopen (char * *host*, char * *port*, char * *login*, char * *passw*, char * *dbName*)

Open a DB connection (not substantially changed from original C version)

Here is the call graph for this function:

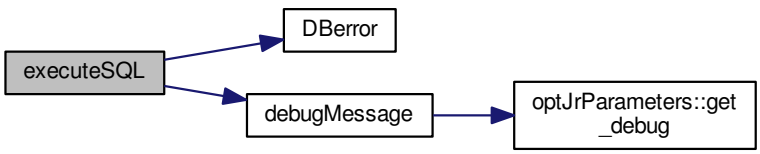
Here is the caller graph for this function:



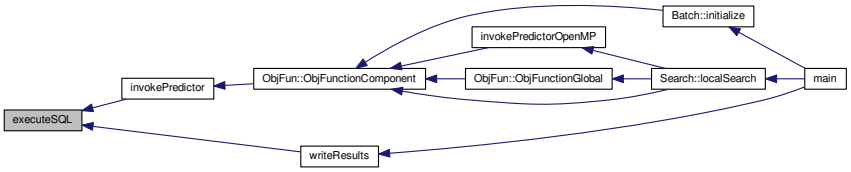
5.12.1.4 MYSQL_ROW executeSQL (MYSQL * *conn*, char * *statement*, optJrParameters *par*)

Execute SQL statement (not substantially changed from original C version)

Here is the call graph for this function:

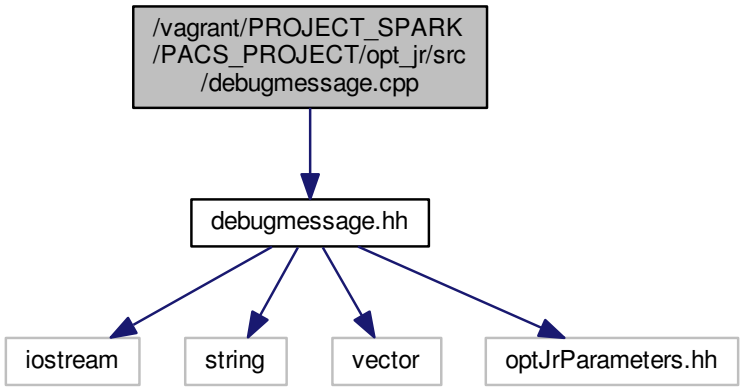


Here is the caller graph for this function:



5.13 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/debugmessage.cpp File Reference

`#include "debugmessage.hh"`
Include dependency graph for debugmessage.cpp:



Functions

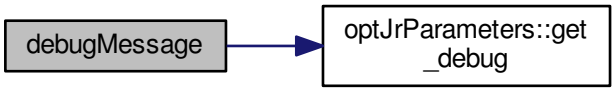
- void debugMessage (std::string &string, optJrParameters &par)

5.13.1 Function Documentation

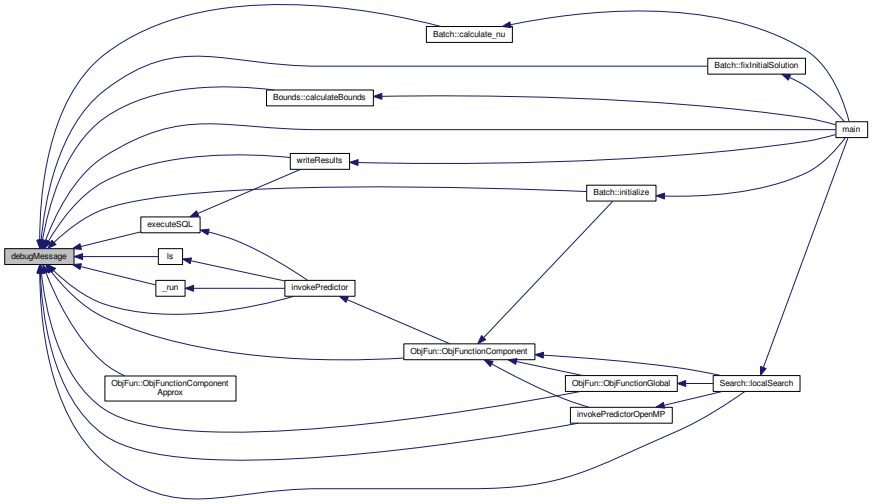
5.13.1.1 void debugMessage (std::string & *string*, optJrParameters & *par*)

debug function: if debugging is activated it shows the message in string

Here is the call graph for this function:



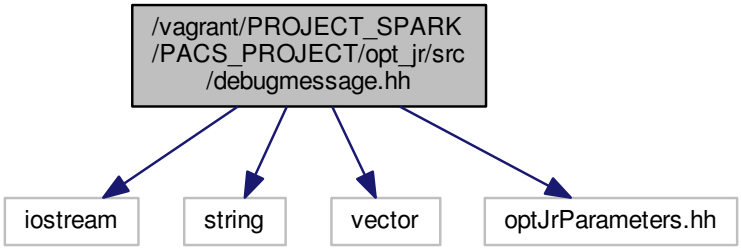
Here is the caller graph for this function:



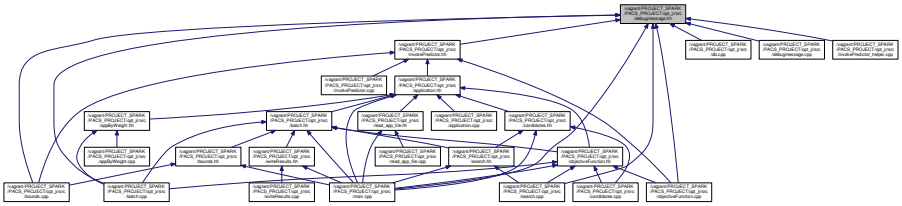
5.14 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/debugmessage.hh File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include "optJrParameters.hh"
```

Include dependency graph for debugmessage.hh:



This graph shows which files directly or indirectly include this file:



Functions

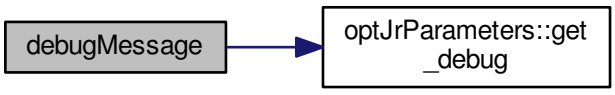
- void debugMessage (std::string &string, optJrParameters &par)

5.14.1 Function Documentation

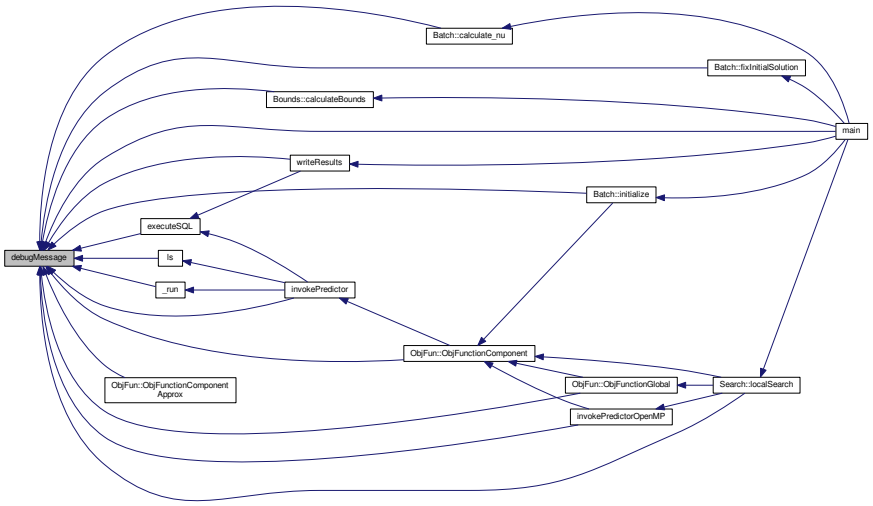
5.14.1.1 void debugMessage (std::string & *string*, optJrParameters & *par*)

debug function: if debugging is activated it shows the message in string

Here is the call graph for this function:



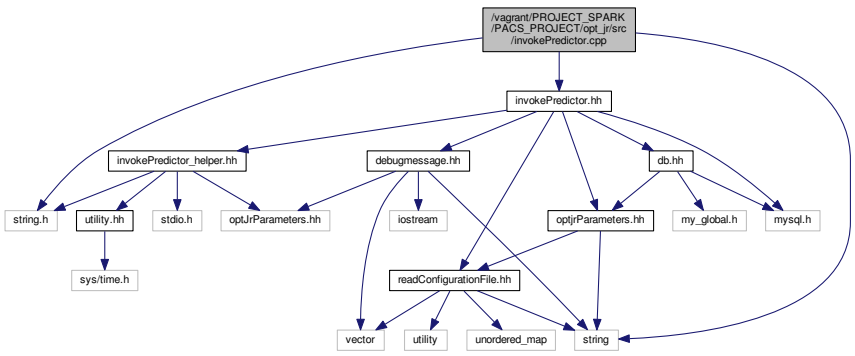
Here is the caller graph for this function:



5.15 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor.cpp File Reference

```
#include "invokePredictor.hh"
#include <string>
#include <string.h>
```

Include dependency graph for invokePredictor.cpp:



Functions

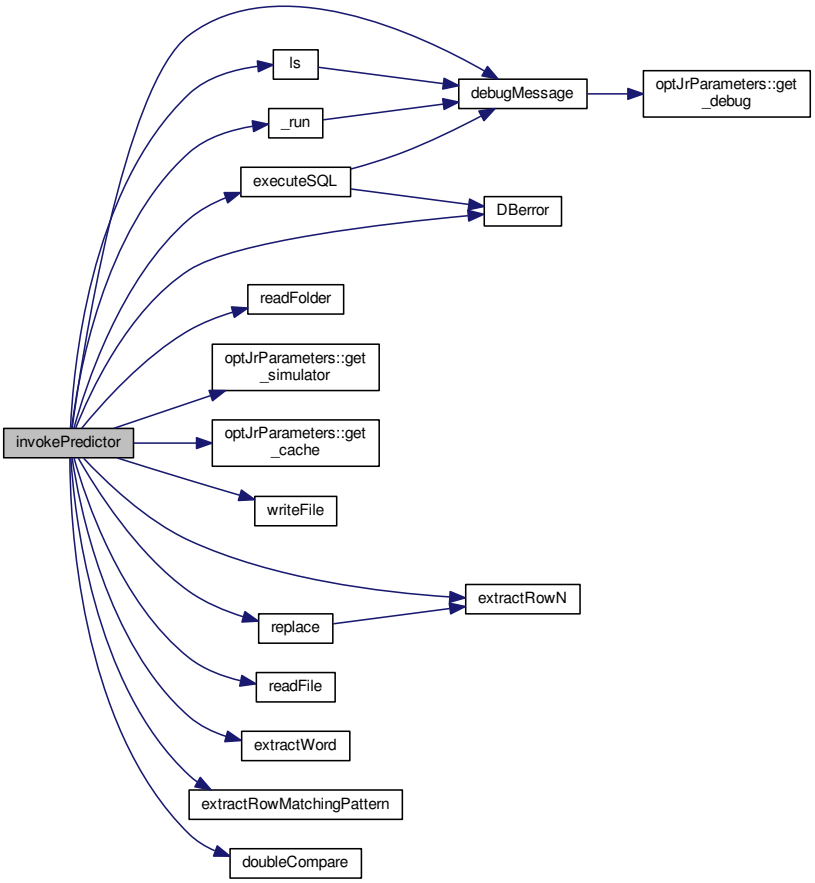
- char * invokePredictor (sConfiguration &configuration, MYSQL *conn, int nNodes, int currentCores, char *memory, int datasize, char *sessionId, char *appId, char *stage, optJrParameters &par, int flagDagsim)

5.15.1 Function Documentation

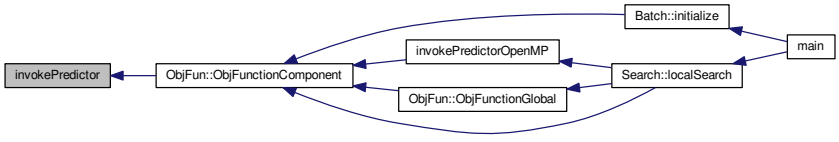
5.15.1.1 char* invokePredictor (sConfiguration & *configuration,* MYSQL * *conn,* int *nNodes,* int *currentCores,* char * *memory,* int *datasize,* char * *sessionId,* char * *appId,* char * *stage,* optJrParameters & *par,* int *flagDagsim*)

"invokePredictor" invokes a predictor (dagSim/Lundstrom). First it checks if an estimate of the execution time is already stored in the DB; if not, it invokes the actual predictor and stores the result on DB cache table.

Here is the call graph for this function:



Here is the caller graph for this function:

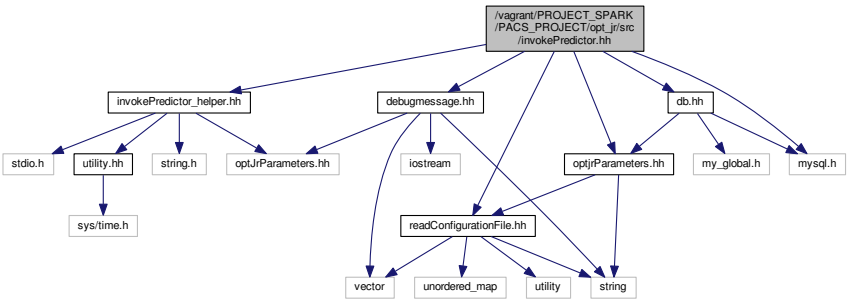


5.16 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor.hh File Reference

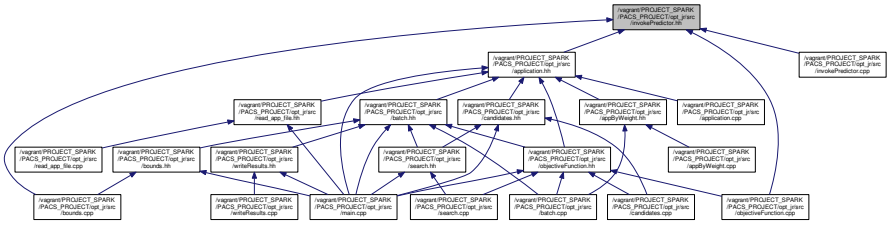
```
#include "invokePredictor_helper.hh"
```

```
#include "readConfigurationFile.hh"
#include "optjrParameters.hh"
#include "debugmessage.hh"
#include "db.hh"
#include <mysql.h>
```

Include dependency graph for invokePredictor.hh:



This graph shows which files directly or indirectly include this file:



Macros

- #define WHOLE_DAGSIM 0
- #define RESIDUAL_DAGSIM 1

Functions

- char * invokePredictor (sConfiguration &configuration, MYSQL *conn, int nNodes, int currentCores, char *memory, int datasize, char *sessionId, char *appId, char *stage, optJrParameters &par, int flagDagsim)

5.16.1 Macro Definition Documentation

5.16.1.1 #define RESIDUAL_DAGSIM 1

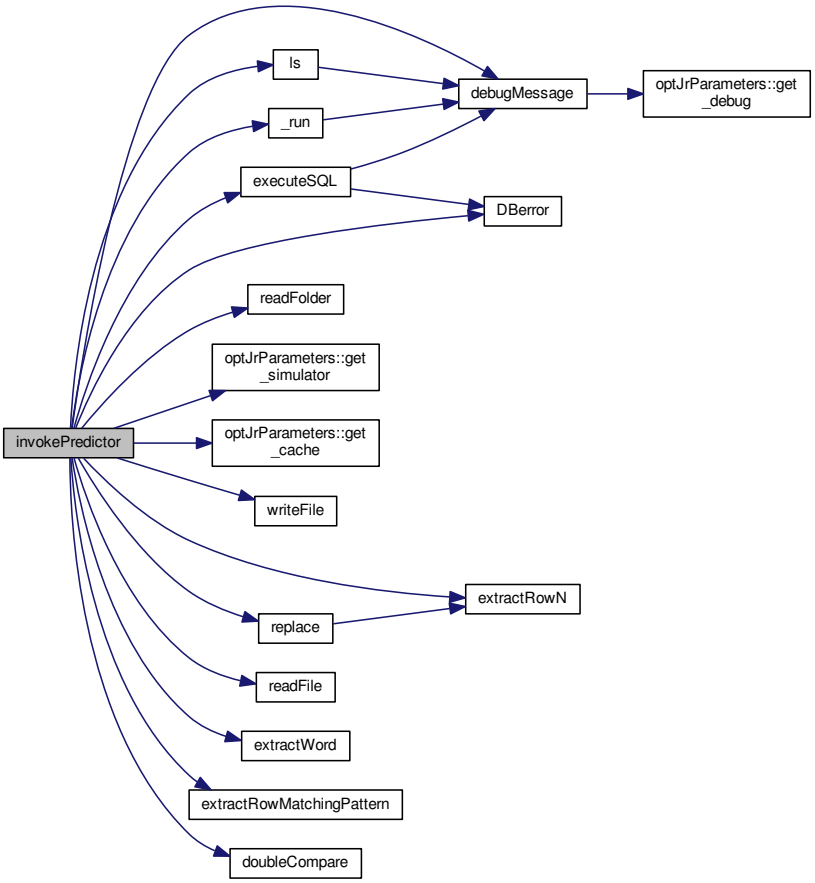
5.16.1.2 #define WHOLE_DAGSIM 0

5.16.2 Function Documentation

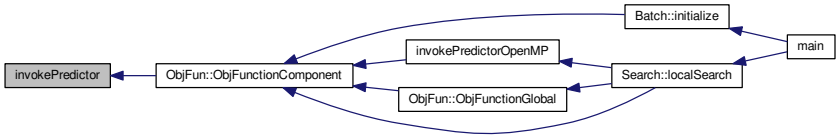
5.16.2.1 char* invokePredictor (sConfiguration & *configuration,* MYSQL * *conn,* int *nNodes,* int *currentCores,* char * *memory,* int *datasize,* char * *sessionId,* char * *appId,* char * *stage,* optJrParameters & *par,* int *flagDagsim*)

"invokePredictor" invokes a predictor (dagSim/Lundstrom). First it checks if an estimate of the execution time is already stored in the DB; if not, it invokes the actual predictor and stores the result on DB cache table.

Here is the call graph for this function:



Here is the caller graph for this function:

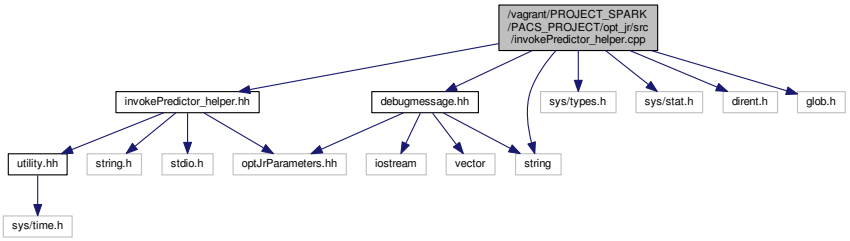


5.17 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor_helper.cpp File Reference

```
#include "invokePredictor_helper.hh"
```

```
#include "debugmessage.hh"
#include <string>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <glob.h>
```

Include dependency graph for invokePredictor_helper.cpp:



Functions

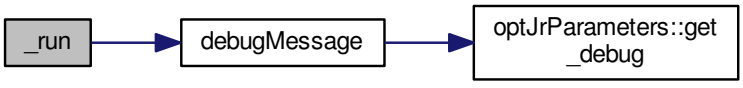
- char ∗ readFolder (char ∗path)
- void writeFile (const char ∗filepath, const char ∗data)
- char ∗ ls (char ∗pattern, optJrParameters &par)
- char ∗ extractRowN (char ∗text, int row)
- char ∗ replace (char ∗text, char ∗newLine)
- char ∗ readFile (char ∗filename)
- char ∗ _run (char ∗cmd, optJrParameters &par)
- char ∗ extractWord (char ∗line, int pos)
- char ∗ extractRowMatchingPattern (char ∗text, char ∗pattern)

5.17.1 Function Documentation

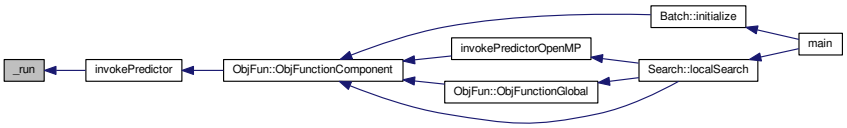
5.17.1.1 char∗ _run (char ∗ *cmd,* optJrParameters & *par*)

Name: _run Output parameters: The output provided by the executed command Description: This function executes a command ("cmd")

Here is the call graph for this function:

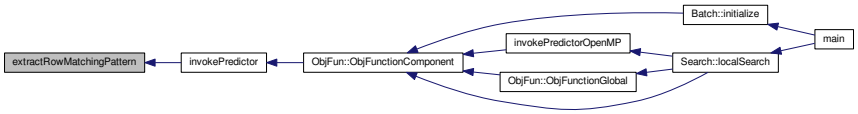


Here is the caller graph for this function:



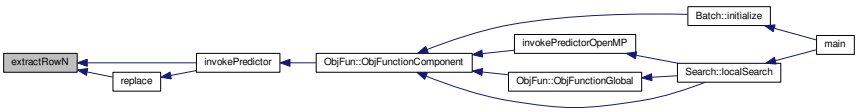
5.17.1.2 char* extractRowMatchingPattern (char * *text*, char * *pattern*)

Here is the caller graph for this function:



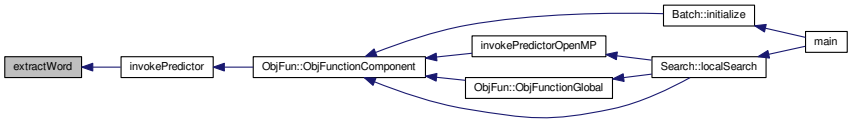
5.17.1.3 char* extractRowN (char * *text*, int *row*)

Here is the caller graph for this function:



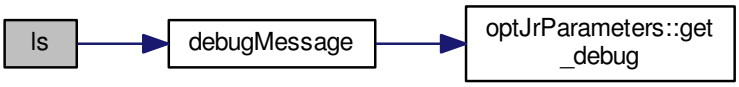
5.17.1.4 char* extractWord (char * *line*, int *pos*)

Here is the caller graph for this function:

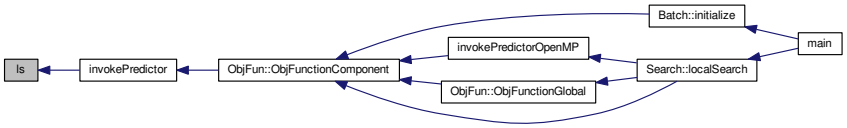


5.17.1.5 char* ls (char * *pattern*, optJrParameters & *par*)

Here is the call graph for this function:

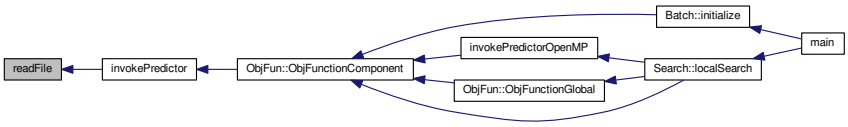


Here is the caller graph for this function:



5.17.1.6 char* readFile (char * *filename*)

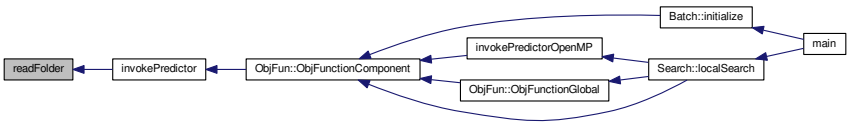
Here is the caller graph for this function:



5.17.1.7 char* readFolder (char * *path*)

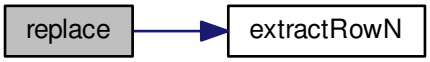
Name: readFolder Input parameters: A path to a folder Output parameters: The name of subfolder contained in the folder corresponding to the folder in "path" Description: It's an helper function used by invoke predictor; this function returns the first subFolder in the folder corresponding to "path"

Here is the caller graph for this function:

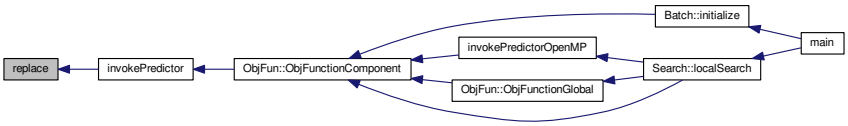


5.17.1.8 char* replace (char * *text*, char * *newLine*)

Here is the call graph for this function:

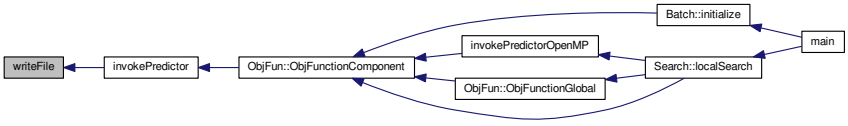


Here is the caller graph for this function:



5.17.1.9 void writeFile (const char * *filepath*, const char * *data*)

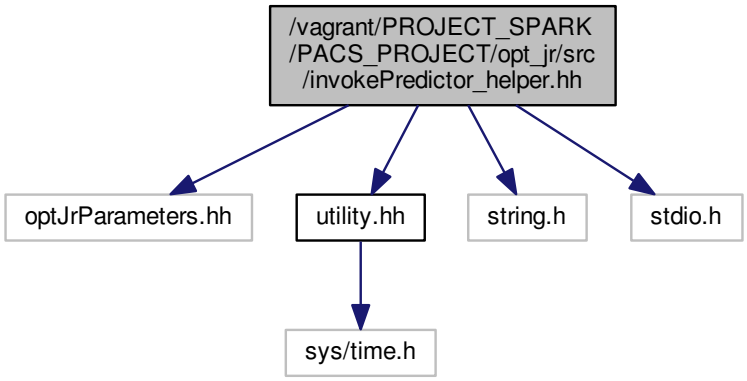
Here is the caller graph for this function:



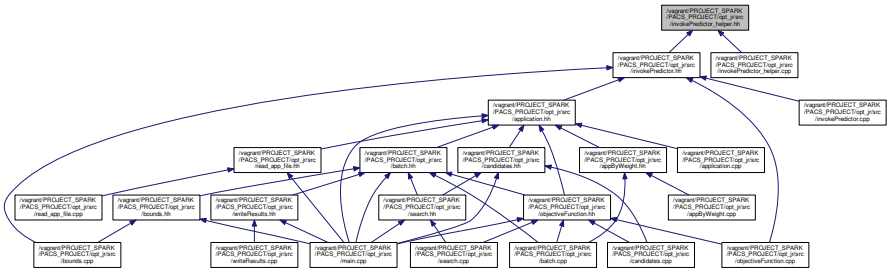
5.18 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/invokePredictor_helper.hh File Reference

```
#include "optJrParameters.hh"
#include "utility.hh"
#include <string.h>
#include <stdio.h>
```

Include dependency graph for invokePredictor_helper.hh:



This graph shows which files directly or indirectly include this file:



Macros

- #define BIG_LINE 4000
- #define BIG_TEXT 20000

Functions

- char * readFolder (char *path)
- char * _run (char *cmd, optJrParameters &par)
- void writeFile (const char *filepath, const char *data)
- char * ls (char *pattern, optJrParameters &par)
- char * extractRowN (char *text, int row)
- char * replace (char *text, char *newLine)
- char * readFile (char *filename)
- char * extractWord (char *line, int pos)
- char * extractRowMatchingPattern (char *text, char *pattern)

5.18.1 Macro Definition Documentation

5.18.1.1 #define BIG_LINE 4000

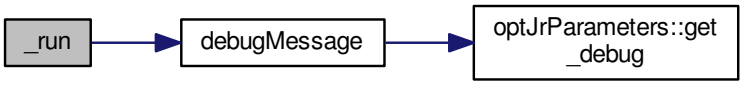
5.18.1.2 #define BIG_TEXT 20000

5.18.2 Function Documentation

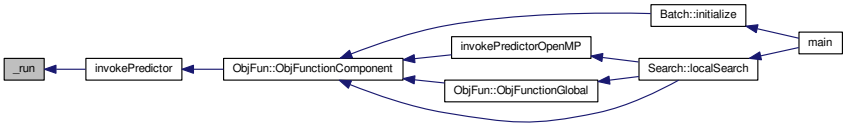
5.18.2.1 char* _run (char * *cmd,* optJrParameters & *par*)

Name: _run Output parameters: The output provided by the executed command Description: This function executes a command ("cmd")

Here is the call graph for this function:

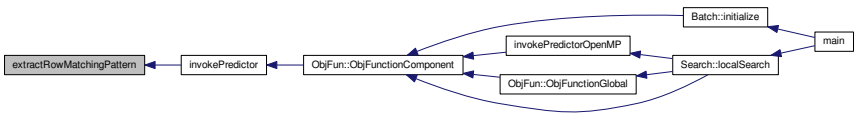


Here is the caller graph for this function:



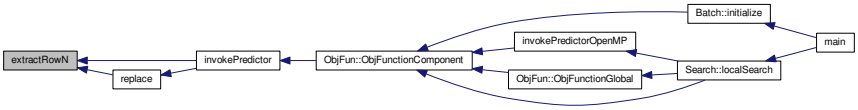
5.18.2.2 char* extractRowMatchingPattern (char * *text,* char * *pattern*)

Here is the caller graph for this function:



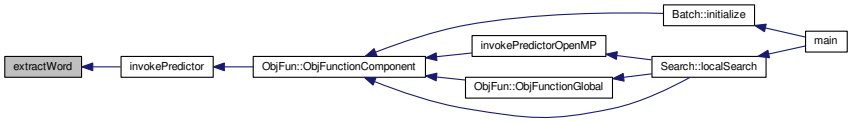
5.18.2.3 char* extractRowN (char * *text*, int *row*)

Here is the caller graph for this function:



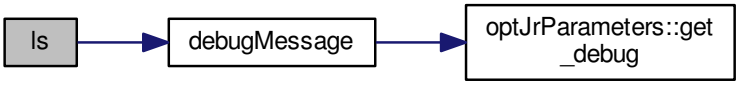
5.18.2.4 char* extractWord (char * *line*, int *pos*)

Here is the caller graph for this function:

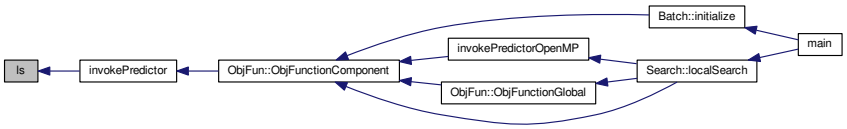


5.18.2.5 char* ls (char * *pattern*, optJrParameters & *par*)

Here is the call graph for this function:

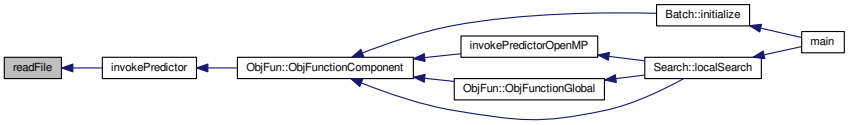


Here is the caller graph for this function:



5.18.2.6 char* readFile (char * *filename*)

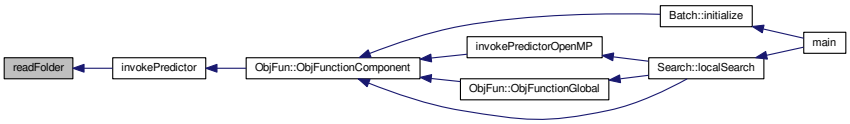
Here is the caller graph for this function:



5.18.2.7 char* readFolder (char * *path*)

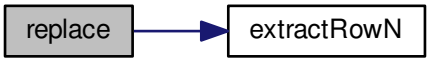
Name: readFolder Input parameters: A path to a folder Output parameters: The name of subfolder contained in the folder corresponding to the folder in "path" Description: It's an helper function used by invoke predictor; this function returns the first subFolder in the folder corresponding to "path"

Here is the caller graph for this function:

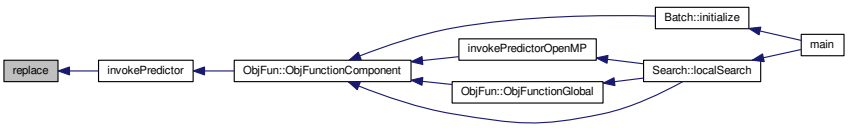


5.18.2.8 char* replace (char * *text,* char * *newLine*)

Here is the call graph for this function:

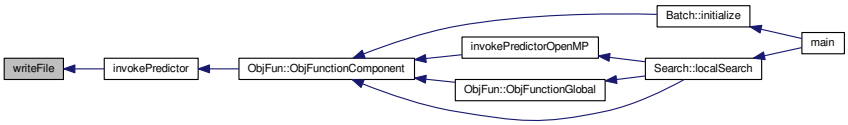


Here is the caller graph for this function:



5.18.2.9 void writeFile (const char * *filepath,* const char * *data*)

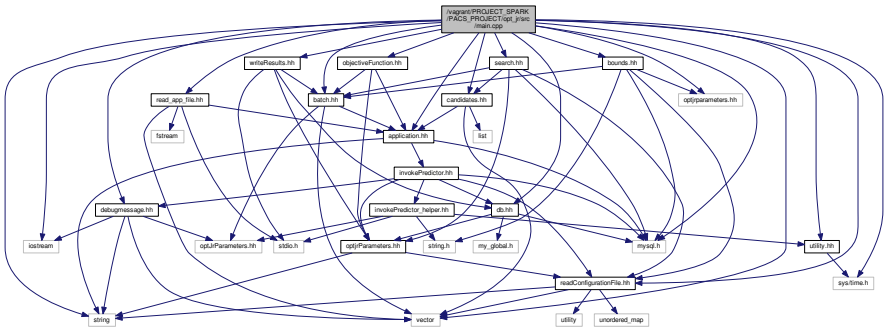
Here is the caller graph for this function:



5.19 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/main.cpp File Reference

```
#include <iostream>
#include <string>
#include <mysql.h>
#include <vector>
#include <sys/time.h>
#include "optjrparameters.hh"
#include "readConfigurationFile.hh"
#include "debugmessage.hh"
#include "db.hh"
#include "application.hh"
#include "read_app_file.hh"
#include "batch.hh"
#include "bounds.hh"
#include "search.hh"
#include "objectiveFunction.hh"
#include "candidates.hh"
#include "utility.hh"
#include "writeResults.hh"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char **argv)

5.19.1 Function Documentation

5.19.1.1 int main (int *argc*, char ∗∗ *argv*)

1) read informations from "wsi_config.xml" file and save it in a "sConfiguration" object (which is unordered_-map(string,string))

2) Read execution parameters from command line and configuration file and save them in an "optJrParameters" object

3) Connect to the Database

4) Open ∗.csv file with Applications data, and save it in a "Batch" object

5) Calculate bounds for each application loaded (with the calculateBounds method of Bounds class)

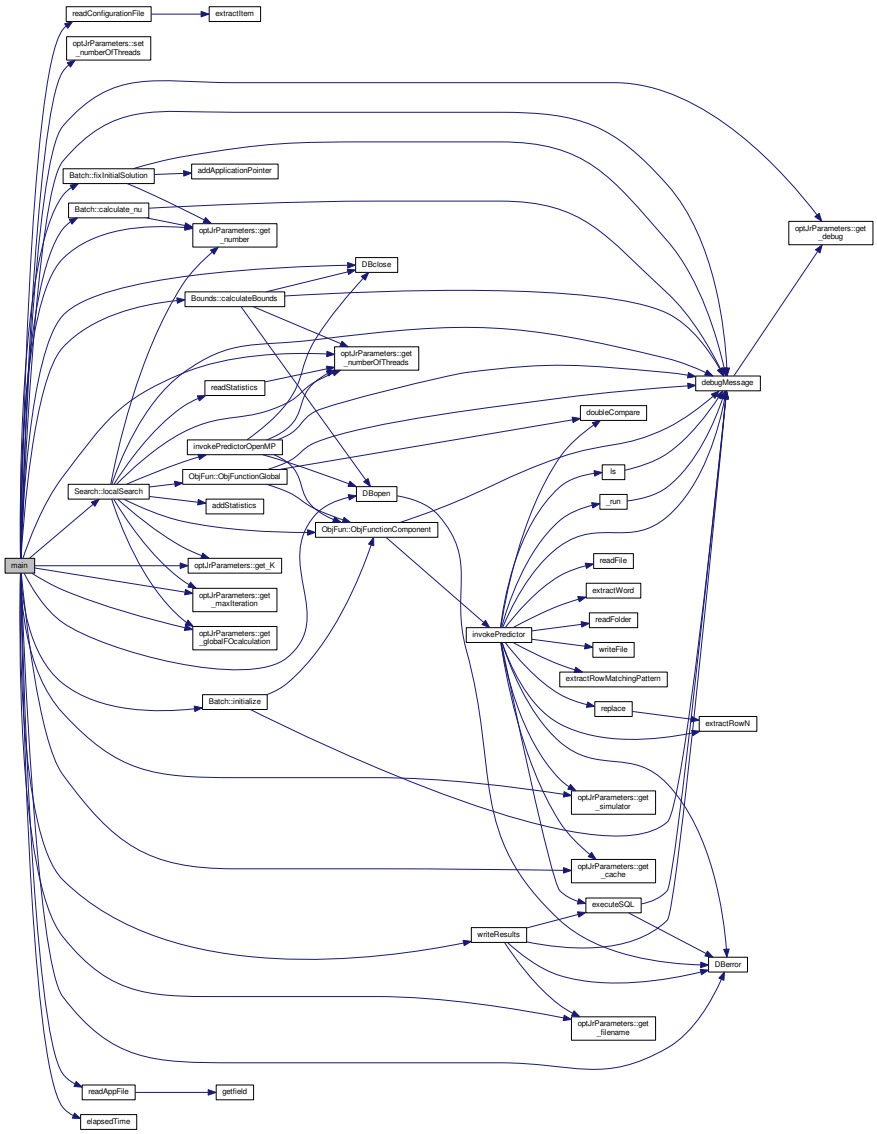
6) Calculate nu indices for each application (with the calculate_nu method of Batch class)

7) Fix initial solution (with the fixInitialSolution method of Batch class)

8) Initialize Objective Function evaluation for each application (with the initalize method of Batch class)

9) Find an "optimal" solution invoking "localSearch" method (of "Search" class)

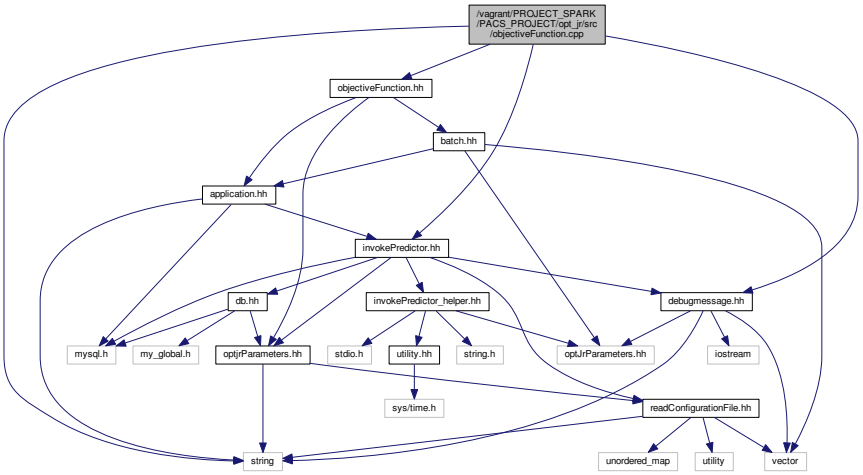
Here is the call graph for this function:



5.20 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/objectiveFunction.cpp File Reference

```
#include "objectiveFunction.hh"
#include <string>
#include "debugmessage.hh"
#include "invokePredictor.hh"
```

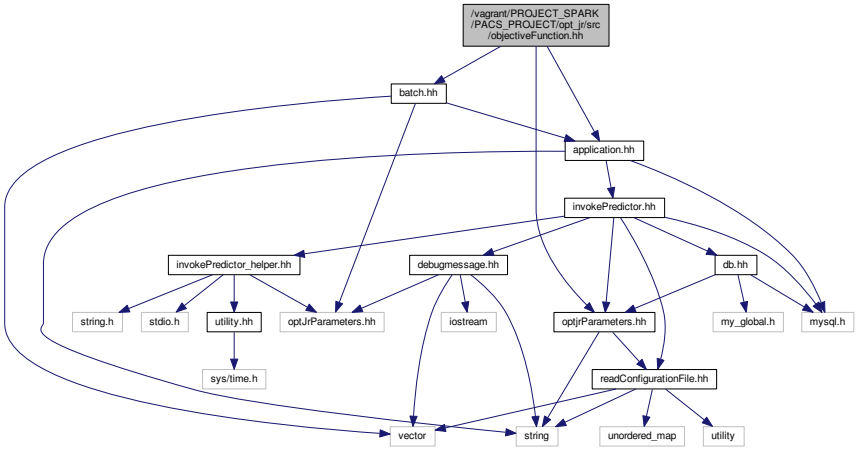
Include dependency graph for objectiveFunction.cpp:



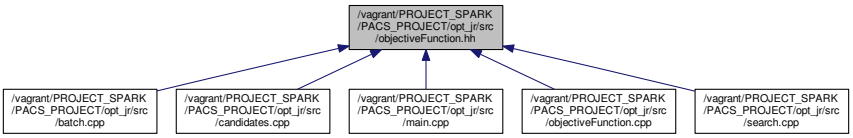
5.21 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/objectiveFunction.hh File Reference

```
#include "application.hh"
#include "optjrParameters.hh"
#include "batch.hh"
```

Include dependency graph for objectiveFunction.hh:



This graph shows which files directly or indirectly include this file:



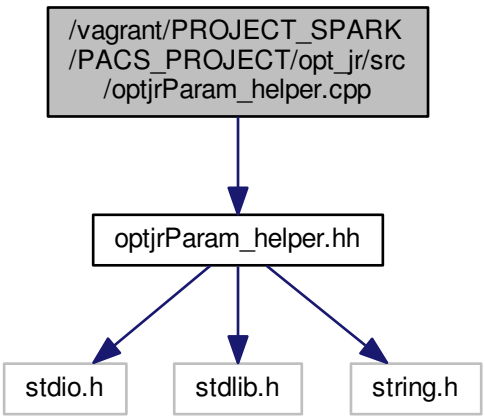
Classes

- class ObjFun

5.22 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParam_helper.cpp File Reference

```
#include "optjrParam_helper.hh"
```

Include dependency graph for optjrParam_helper.cpp:



Functions

- void Usage (int argc)
- char * parseArg (char *string, char *gap, int type, int argc)

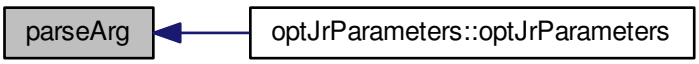
5.22.1 Function Documentation

5.22.1.1 char* parseArg (char * *string*, char * *gap*, int *type*, int *argc*)

Function to parse argument from command line; Invoked by optJrParameters constructor

Here is the call graph for this function:

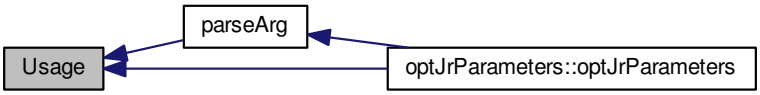
Here is the caller graph for this function:



5.22.1.2 void Usage (int *argc*)

Explain usage of the OPT_JR_CPP program Invoked if the number of parameters send at command line is incorrect

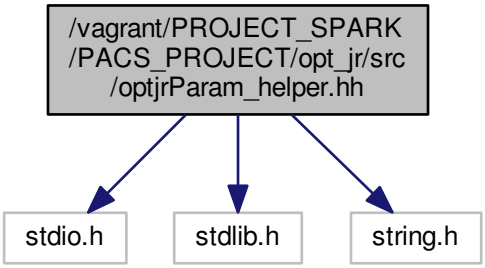
Here is the caller graph for this function:



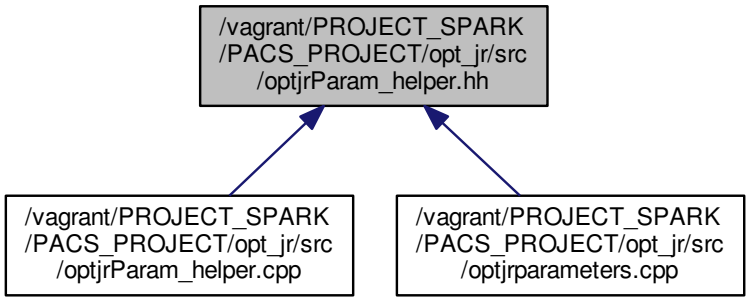
5.23 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParam_helper.hh File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for optjrParam_helper.hh:



This graph shows which files directly or indirectly include this file:



Macros

- #define ARGS 9

  number of arguments from command line
- #define FILENAME "-f="
- #define NUM_N "-n="
- #define LIST_LIMIT "-k="
- #define DEBUG "-d="
- #define MAX_ITERATIONS "-i="
- #define SIMULATOR "-s="
- #define GLOBAL_FO_CALCULATION "-g"
- #define CACHE "-c"
- #define NUMBER 0
- #define STRING 1
- #define YES_NO 2
- #define NO 0
- #define YES 1

Functions

- void Usage (int argc)
- char ∗ parseArg (char ∗string, char ∗gap, int type, int argc)

5.23.1 Macro Definition Documentation

5.23.1.1 #define ARGS 9

number of arguments from command line

5.23.1.2 #define CACHE "-c"

5.23.1.3 #define DEBUG "-d="

5.23.1.4 #define FILENAME "-f="

5.23.1.5 #define GLOBAL_FO_CALCULATION "-g"

5.23.1.6 #define LIST_LIMIT "-k="

5.23.1.7 #define MAX_ITERATIONS "-i="

5.23.1.8 #define NO 0

5.23.1.9 #define NUM_N "-n="

5.23.1.10 #define NUMBER 0

5.23.1.11 #define SIMULATOR "-s="

5.23.1.12 #define STRING 1

5.23.1.13 #define YES 1

5.23.1.14 #define YES_NO 2

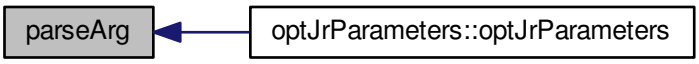
5.23.2 Function Documentation

5.23.2.1 char∗ parseArg (char ∗ *string,* char ∗ *gap,* int *type,* int *argc*)

Function to parse argument from command line; Invoked by optJrParameters constructor

Here is the call graph for this function:

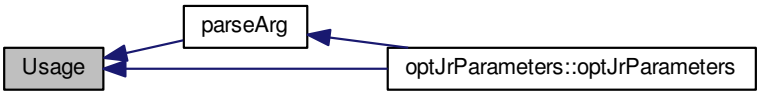
Here is the caller graph for this function:



5.23.2.2 void Usage (int *argc*)

Explain usage of the OPT_JR_CPP program Invoked if the number of parameters send at command line is incorrect

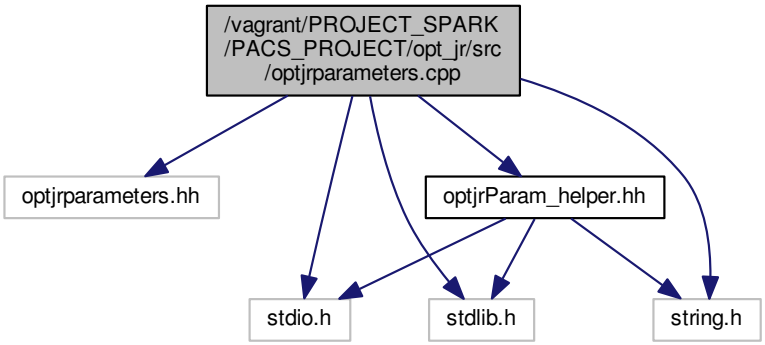
Here is the caller graph for this function:



5.24 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrparameters.cpp File Reference

```
#include "optjrparameters.hh"
#include "optjrParam_helper.hh"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

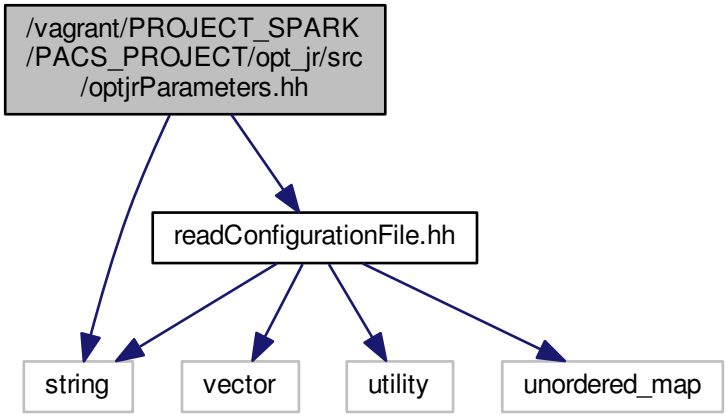
Include dependency graph for optjrparameters.cpp:



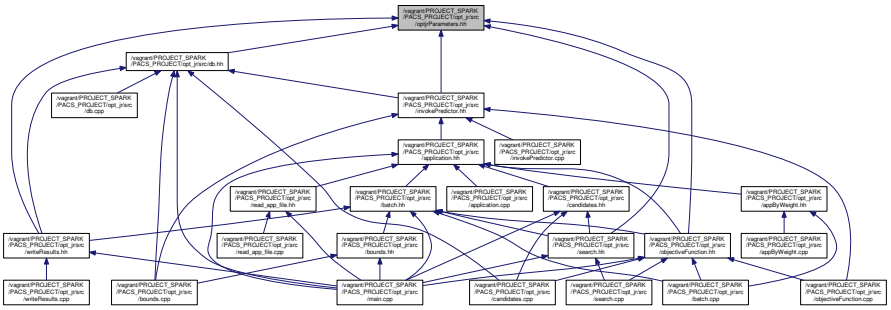
5.25 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/optjrParameters.hh File Reference

```
#include <string>
#include "readConfigurationFile.hh"
```

Include dependency graph for optjrParameters.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class optJrParameters

Macros

- #define DAGSIM 0
- #define LUNDSTROM 1

5.25.1 Macro Definition Documentation

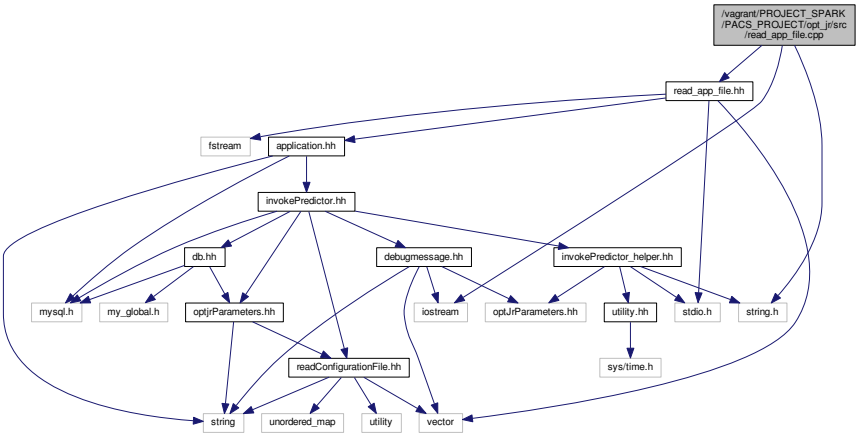
5.25.1.1 #define DAGSIM 0

5.25.1.2 #define LUNDSTROM 1

5.26 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/read_app_file.cpp File Reference

```
#include "read_app_file.hh"
#include <string.h>
#include <iostream>
```

Include dependency graph for read_app_file.cpp:



Functions

- char * getfield (char *line, int num)
- std::vector< Application > readAppFile (FILE *stream)

5.26.1 Function Documentation

5.26.1.1 char* getfield (char * *line*, int *num*)

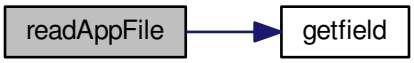
Name: getfield Input parameters: char * source, int num Output parameters: A word Description: it extracts values from the csv file

Here is the caller graph for this function:

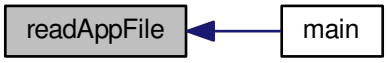
5.26.1.2 std::vector<Application> readAppFile (FILE * *stream*)

This function given a file* with data of application returns the vector of "Application" objects

Here is the call graph for this function:



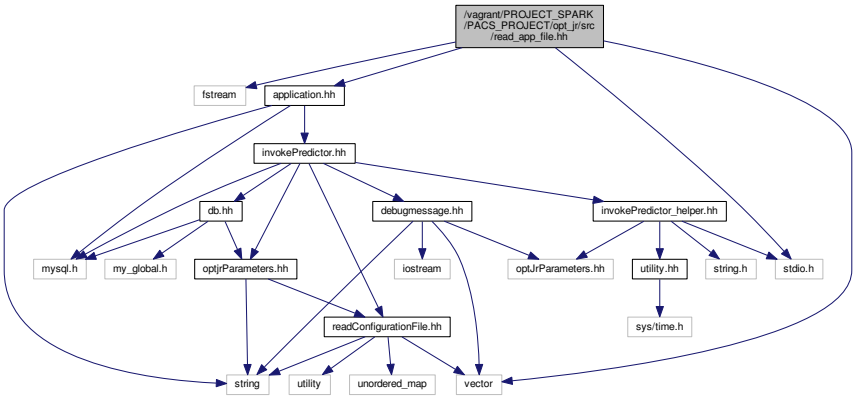
Here is the caller graph for this function:



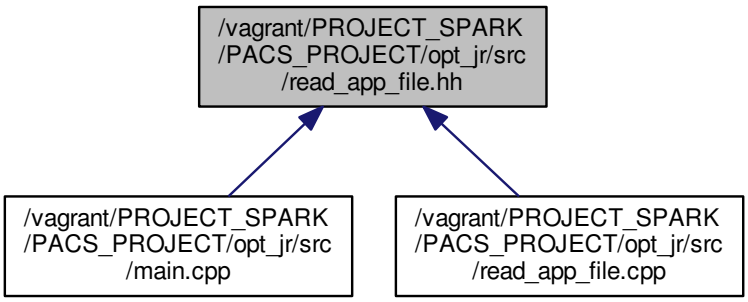
5.27 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/read_app_file.hh File Reference

```
#include <fstream>
#include <stdio.h>
#include <vector>
#include "application.hh"
```

Include dependency graph for read_app_file.hh:



This graph shows which files directly or indirectly include this file:



Macros

- #define MAX_APP_LENGTH 1024

Functions

- char * getfield (char *line, int num)
- std::vector< Application > readAppFile (FILE *stream)

Variables

- const int MAX_LINE_LENGTH = 1024
- const int _SESSION_APP_ID = 1
- const int _APP_ID = 2
- const int _W = 3
- const int _CHI_0 = 4
- const int _CHI_C = 5
- const int _M = 6
- const int _m = 7
- const int _V = 8
- const int _v = 9
- const int _D = 10
- const int _St = 11
- const int _Dsz = 12
- const int PARAMETERS = 12

5.27.1 Macro Definition Documentation

5.27.1.1 #define MAX_APP_LENGTH 1024

5.27.2 Function Documentation

5.27.2.1 char* getfield (char * *line,* int *num*)

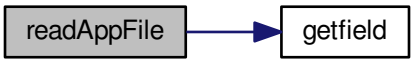
Name: getfield Input parameters: char * source, int num Output parameters: A word Description: it extracts values from the csv file

Here is the caller graph for this function:

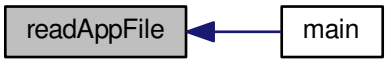
5.27.2.2 std::vector<Application> readAppFile (FILE * *stream*)

This function given a file* with data of application returns the vector of "Application" objects

Here is the call graph for this function:



Here is the caller graph for this function:



5.27.3 Variable Documentation

5.27.3.1 const int _APP_ID = 2

5.27.3.2 const int _CHI_0 = 4

5.27.3.3 const int _CHI_C = 5

5.27.3.4 const int _D = 10

5.27.3.5 const int _Dsz = 12

5.27.3.6 const int _M = 6

5.27.3.7 const int _m = 7

5.27.3.8 const int _SESSION_APP_ID = 1

5.27.3.9 const int _St = 11

5.27.3.10 const int _V = 8

5.27.3.11 const int _v = 9

5.27.3.12 const int _W = 3

5.27.3.13 const int MAX_LINE_LENGTH = 1024

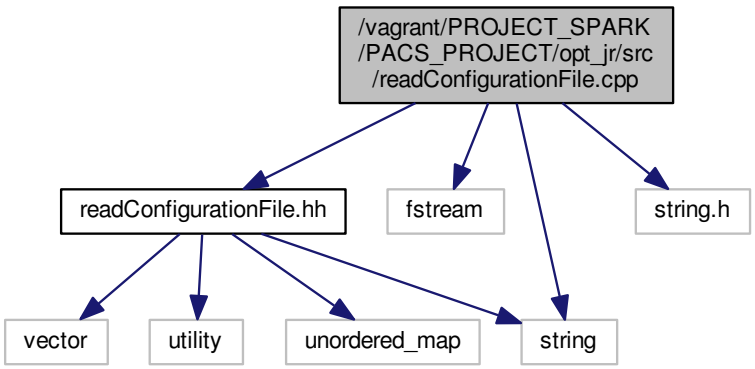
5.27.3.14 const int PARAMETERS = 12

Number of values in the csv file

5.28 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/readConfigurationFile.cpp File Reference

```
#include "readConfigurationFile.hh"
#include <fstream>
#include <string>
#include <string.h>
```

Include dependency graph for readConfigurationFile.cpp:



Functions

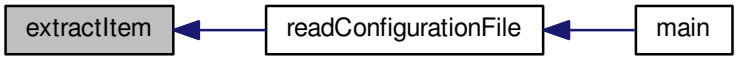
- sConfiguration readConfigurationFile ()
- char ∗ extractItem (char ∗const string, char ∗const left, const char ∗const right)

5.28.1 Function Documentation

5.28.1.1 char* extractItem (char *const *string,* char *const *left,* const char *const *right*)

This is an helper function used by readConfigurationFile; it's used to parse input from configuration file. (not changed from original C version)

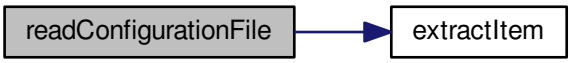
Here is the caller graph for this function:



5.28.1.2 sConfiguration readConfigurationFile ( )

This function inizializes the sConfiguration container; it reads the file defined in the environmental variable WSI_CONFIG_FILE

Here is the call graph for this function:

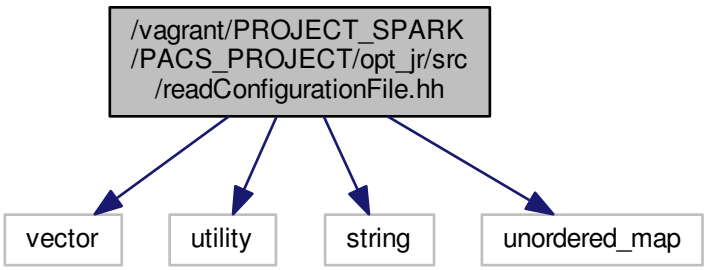


Here is the caller graph for this function:

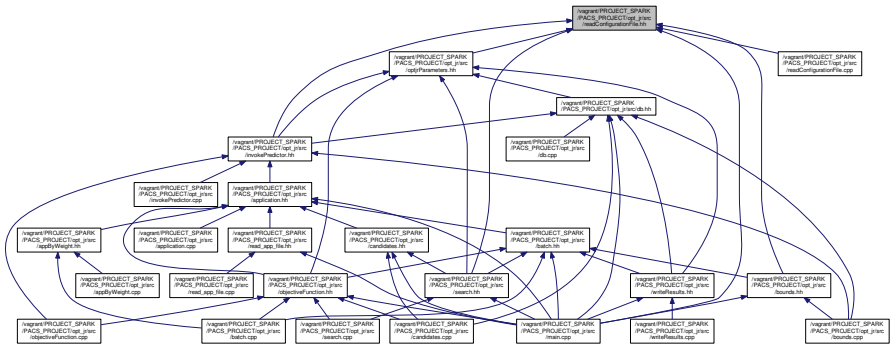
5.29 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/readConfigurationFile.hh File Reference

```
#include <vector>
#include <utility>
#include <string>
#include <unordered_map>
```

Include dependency graph for readConfigurationFile.hh:



This graph shows which files directly or indirectly include this file:



Typedefs

- using sConfiguration = std::unordered_map< std::string, std::string >

Functions

- sConfiguration readConfigurationFile ()
- char * extractItem (char *const string, char *const left, const char *const right)

5.29.1 Typedef Documentation

5.29.1.1 using sConfiguration = std::unordered_map<std::string,std::string>

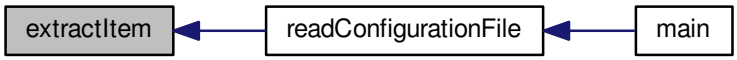
sConfiguration is a container which memorize data from the configuration file; it's an unordered map

5.29.2 Function Documentation

5.29.2.1 char* extractItem (char *const *string*, char *const *left*, const char *const *right*)

This is an helper function used by readConfigurationFile; it's used to parse input from configuration file. (not changed from original C version)

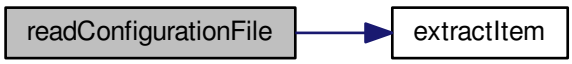
Here is the caller graph for this function:



5.29.2.2 sConfiguration readConfigurationFile ( )

This function inizializes the sConfiguration container; it reads the file defined in the environmental variable WSI_CONFIG_FILE

Here is the call graph for this function:

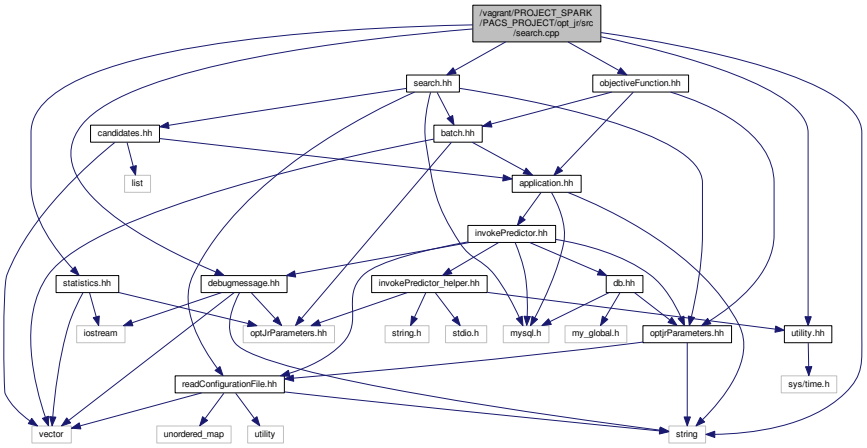


Here is the caller graph for this function:

5.30 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/search.cpp File Reference

```
#include "search.hh"
#include "debugmessage.hh"
#include "utility.hh"
#include "objectiveFunction.hh"
#include "statistics.hh"
#include <string>
```

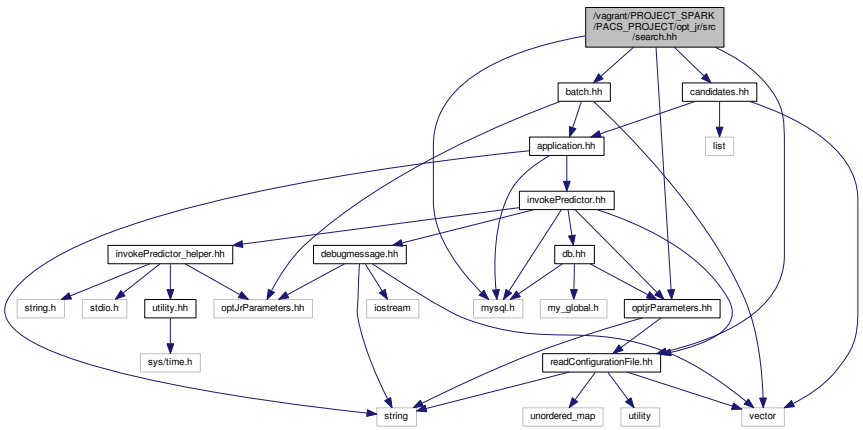
Include dependency graph for search.cpp:



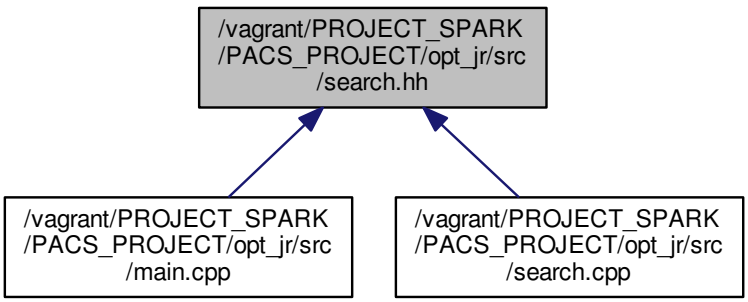
5.31 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/search.hh File Reference

```
#include "readConfigurationFile.hh"
#include "batch.hh"
#include "optjrParameters.hh"
#include "candidates.hh"
#include <mysql.h>
```

Include dependency graph for search.hh:



This graph shows which files directly or indirectly include this file:



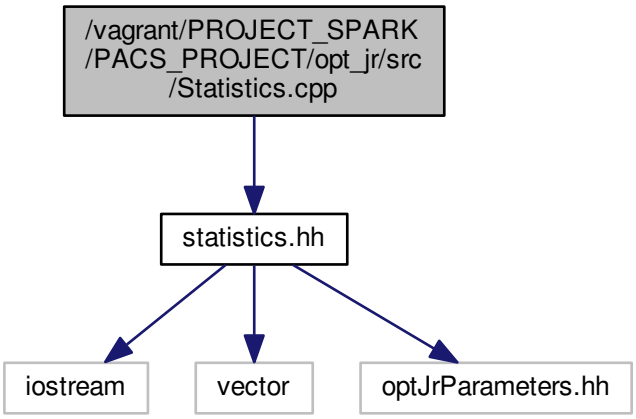
Classes

- class Search

5.32 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/Statistics.cpp File Reference

```
#include "statistics.hh"
```

Include dependency graph for Statistics.cpp:



Functions

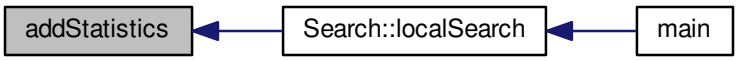
- void addStatistics (sStatistics &statistics, int iteration, int size, double FO_total)
- void readStatistics (sStatistics &statistics, optJrParameters &par)

5.32.1 Function Documentation

5.32.1.1 void addStatistics (sStatistics & *statistics*, int *iteration*, int *size*, double *FO_total*)

"addStatistics" is used to add information about an iteration to the "sStatistics" object.

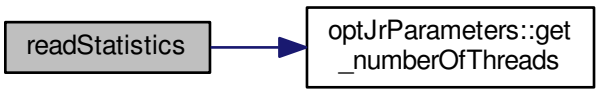
Here is the caller graph for this function:



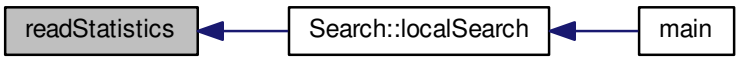
5.32.1.2 void readStatistics (sStatistics & *statistics*, optJrParameters & *par*)

"readStatistics" shows the statistics about localsearch iterations.

Here is the call graph for this function:



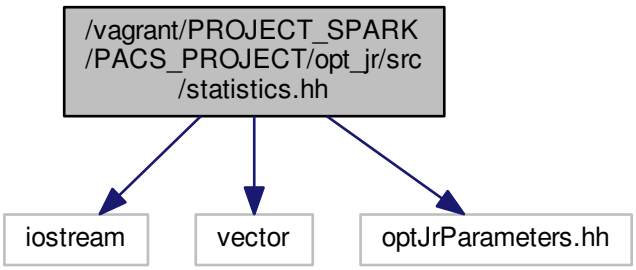
Here is the caller graph for this function:



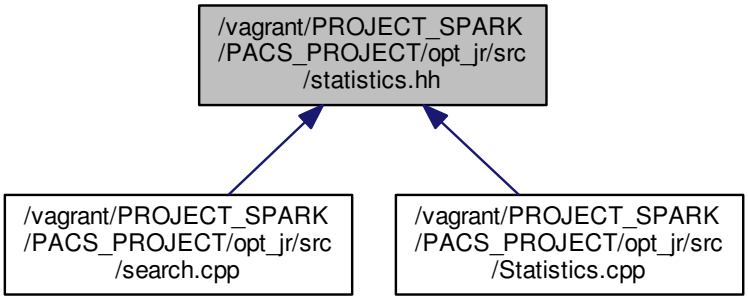
5.33 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/statistics.hh File Reference

```
#include <iostream>
#include <vector>
#include "optJrParameters.hh"
```

Include dependency graph for statistics.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class Statistic

Typedefs

- using sStatistics = std::vector< Statistic >

Functions

- void addStatistics (sStatistics &statistics, int iteration, int size, double FO_total)
- void readStatistics (sStatistics &statistics, optJrParameters &par)

5.33.1 Typedef Documentation

5.33.1.1 using sStatistics = std::vector<Statistic>

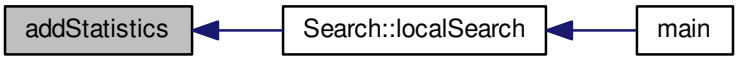
A vector of "Statistic" object is used to store statistical information about the local search.

5.33.2 Function Documentation

5.33.2.1 void addStatistics (sStatistics & *statistics*, int *iteration*, int *size*, double *FO_total*)

"addStatistics" is used to add information about an iteration to the "sStatistics" object.

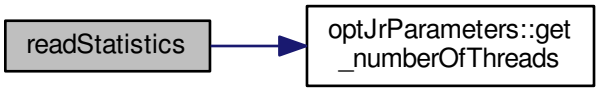
Here is the caller graph for this function:



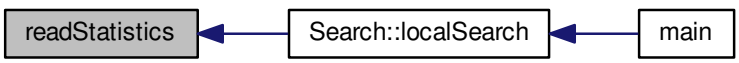
5.33.2.2 void readStatistics (sStatistics & *statistics*, optJrParameters & *par*)

"readStatistics" shows the statistics about localsearch iterations.

Here is the call graph for this function:



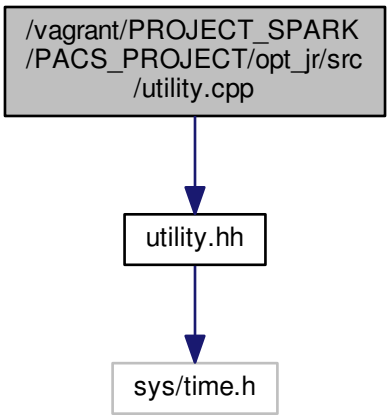
Here is the caller graph for this function:



5.34 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/utility.cpp File Reference

```
#include "utility.hh"
```

Include dependency graph for utility.cpp:



Functions

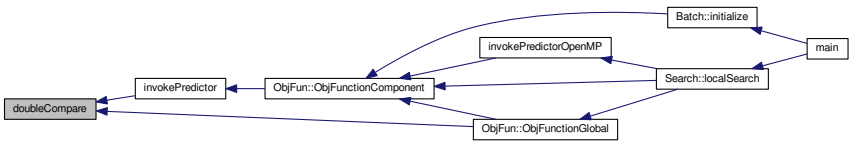
- double elapsedTime (struct timeval tv1, struct timeval tv2)
- int doubleCompare (double a, double b)

5.34.1 Function Documentation

5.34.1.1 int doubleCompare (double *a*, double *b*)

"doubleCompare" compare two doubles according to a certain precision (epsilon)

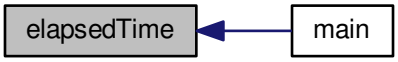
Here is the caller graph for this function:



5.34.1.2 double elapsedTime (struct timeval *tv1*, struct timeval *tv2*)

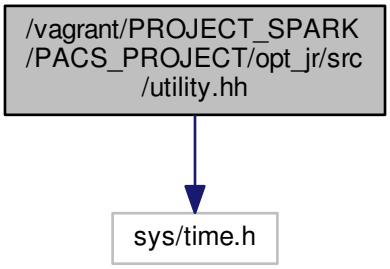
Measures the elapsed time

Here is the caller graph for this function:

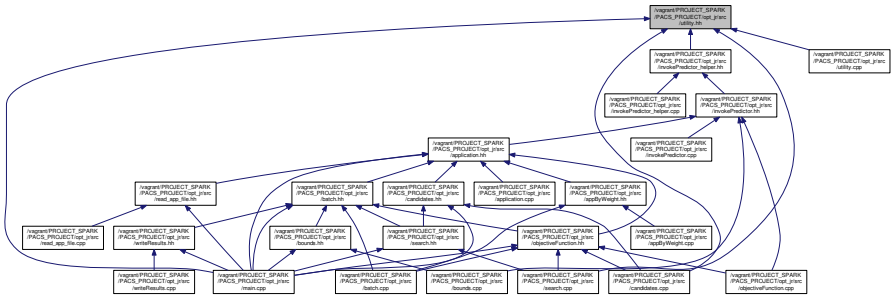


5.35 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/utility.hh File Reference

`#include <sys/time.h>`
Include dependency graph for utility.hh:



This graph shows which files directly or indirectly include this file:



Functions

- double elapsedTime (struct timeval tv1, struct timeval tv2)
- int doubleCompare (double a, double b)

Variables

- const double epsilon = 0.001

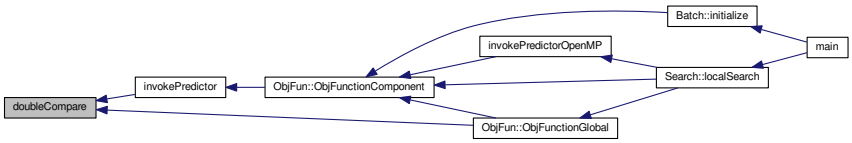
  precision in doubleCompare

5.35.1 Function Documentation

5.35.1.1 int doubleCompare (double *a*, double *b*)

"doubleCompare" compare two doubles according to a certain precision (epsilon)

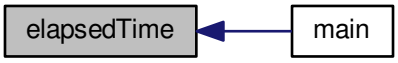
Here is the caller graph for this function:



5.35.1.2 double elapsedTime (struct timeval *tv1*, struct timeval *tv2*)

Measures the elapsed time

Here is the caller graph for this function:



5.35.2 Variable Documentation

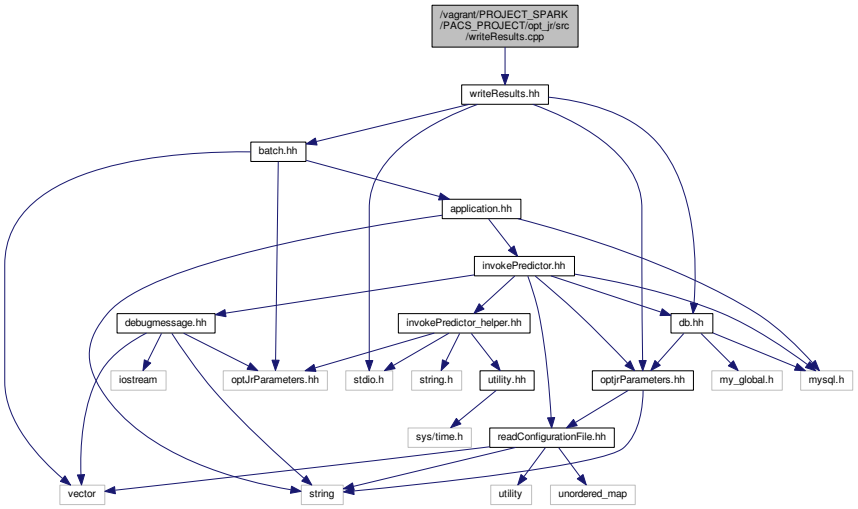
5.35.2.1 const double epsilon = 0.001

precision in doubleCompare

5.36 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/writeResults.cpp File Reference

```
#include "writeResults.hh"
```

Include dependency graph for writeResults.cpp:



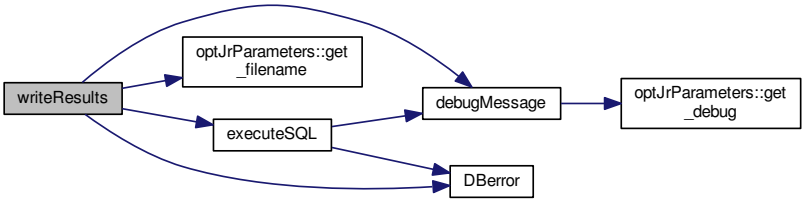
Functions

- void writeResults (MYSQL ∗conn, char ∗dbName, Batch &App_manager, optJrParameters &par)

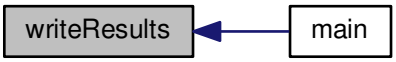
5.36.1 Function Documentation

5.36.1.1 void writeResults (MYSQL ∗ *conn,* char ∗ *dbName,* Batch & *App_manager,* optJrParameters & *par*)

Here is the call graph for this function:



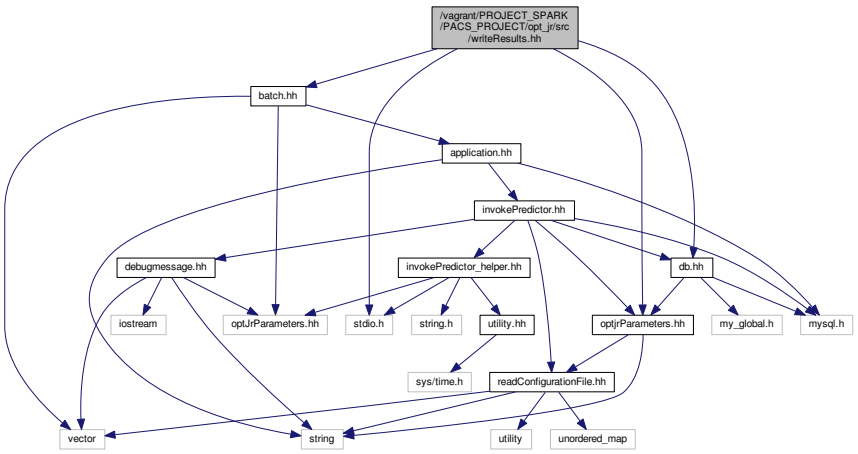
Here is the caller graph for this function:



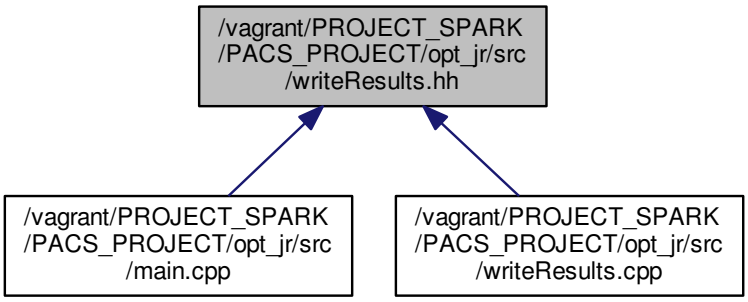
5.37 /vagrant/PROJECT_SPARK/PACS_PROJECT/opt_jr/src/writeResults.hh File Reference

```
#include <stdio.h>
#include "db.hh"
#include "batch.hh"
#include "optjrParameters.hh"
```

Include dependency graph for writeResults.hh:



This graph shows which files directly or indirectly include this file:



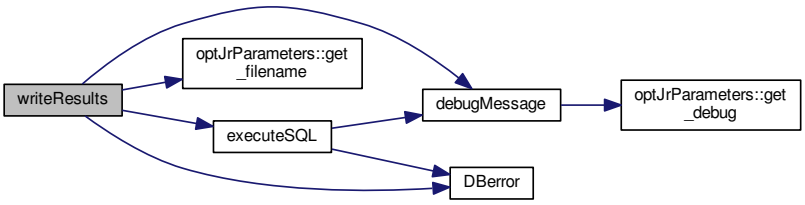
Functions

- void writeResults (MYSQL ∗conn, char ∗dbName, Batch &App_manager, optJrParameters &par)

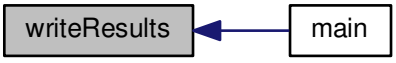
5.37.1 Function Documentation

5.37.1.1 void writeResults (MYSQL ∗ *conn,* char ∗ *dbName,* Batch & *App_manager,* optJrParameters & *par*)

Here is the call graph for this function:



Here is the caller graph for this function:



5.38 /vagrant/PROJECT_SPARK/PACS_PROJECT/README.MD File Reference