

«РЕАЛИЗАЦИЯ ФУНКЦИЙ КЛАСТЕРНОГО ПАКЕТА MPI/MPICH»

АВТОР : АЛЬ-ХУЛАЙДИ АБДУЛМАДЖИД АХМЕД(ЙЕМЕН)
РОССИЯ , ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ, Abdulmajed_83@mail.ru

Аннотация. В данной работе приводится реализация функций кластерного пакета MPI/MPICH, которые позволяют обеспечить более высокие параллельные вычисления в многопроцессорных, кластерных системах.

Ключевые слова: MPI, MPICH, кластер.

Задачи исследования

В последние годы за рубежом и в нашей стране активно ведутся работы, связанные с использованием технологии параллельной обработки. С использованием параллельной обработки, возможно, значительно повысить скорость выполнения работы. Поскольку суперкомпьютеры остаются достаточно дорогими, только большие компании и институты способны обладать такими вычислительными возможностями[1,2]. Кластерный пакет MPI/MPICH , используется для повышения эффективности параллельных вычислений. Пакет состоит из следующих функций:

1- MPI_Init(&argc, &argv);

//функция инициализации – создает коммуникационное пространство, вызов обязателен.

2- MPI_Finalize();// функция освобождения коммуникационного пространства

3- int MPI_Comm_size(MPI_Comm comm, int *size)//функция присваивает переменной size количество процессов

4- int MPI_Comm_rank(MPI_Comm comm, int *rank)//функция присваивает переменной rank номер процесса

5- // Блокирующая передача осуществляется с помощью функции MPI_Send.

6- //Функция MPI_Recv реализует блокирующий прием данных.

7- MPI_ERRORS_ARE_FATAL //обработчик, который после вызова прерывает работу программы на всех процессах.

8- MPI_ERRORS_RETURN // обработчик не делает ничего, кроме представления кода ошибки пользователю.

9- MPI_NULL_COPY_FN// есть функция, которая не делает ничего другого, кроме возвращения flag = 0 и MPI_SUCCESS.

10- MPI_DUP_FN //есть простая функция копирования, которая устанавливает flag = 1, возвращает значение attribute_val_in в attribute_val_out и возвращает MPI_SUCCESS.

11- MPI_NULL_DELETE_FN //не делает ничего другого, кроме возвращения MPI_SUCCESS.

12- MPI_COMM_DUP// вызываются (в произвольном порядке) все установленные на этот момент функции обратного вызова для копирования атрибутов. Всякий раз, когда коммутатор удаляется функцией MPI_COMM_FREE, вызываются все установленные функции обратного вызова для удаления атрибутов.

13- MPI_COMM_DUP_FN // Возвращает значение attribute_val_in в attribute_val_out и устанавливает MPI_SUCCESS

14-MPI_COMM_NULL_COPY_FN// Возвращает flag = false и MPI_SUCCESS

15- MPI_ABORT(comm, errorcode) //Прекращает выполнение операций

16- MPI_ADDRESS(location, address)// Устарела, вместо нее используется MPI_GET_ADDRESS

17- MPI_ALLGATHER(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvttype, comm)
 //Собирает данные от всех процессов и распределяет их всем процессам.

18- MPI_ALLGATHERV(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs, recvttype, comm)//Собирает данные от всех процессов и поставляет их всем процессам

19- MPI_ALLREDUCE(sendbuf, recvbuf, count, datatype, op, comm) //Выполняет глобальную операцию над данными от всех процессов и результат посылает обратно всем процессам.

20- MPI_ALLTOALL(sendbuf, sendcount, sendtype, recvbuf, recvcount,recvttype, comm)
 //Посылает данные от всех процессов всем процессам

21- MPI_ALLTOALLV (sendbuf, sendcounts, sdispls, sendtype, recvbuf, recvcounts, rdispls, recvttype, comm)// Посылает данные от всех процессов всем процессам со смещением

22- MPI_ATTR_DELETE (comm, keyval)// Удаляет связанное с ключом значение. Устарела, вместо нее используется MPI_COMM_DELETE_ATTR

23- MPI_ATTR_GET(comm, keyval, attribute_val, flag)//Обрабатывает значение атрибута по ключу Устарела, вместо нее используется MPI_COMM_GET_ATTR

24- MPI_ATTR_PUT(comm, keyval, attribute_val)//Хранит связанное с ключом значение атрибута устарела, вместо нее используется MPI_COMM_SET_ATTR

25- MPI_BARRIER (comm) //Блокирует дальнейшее исполнение, пока все процессы не достигнут этой функции

26- MPI_BCAST(buffer, count, datatype, root, comm)//Широковещательная рассылка сообщения от одного процесса всем

27- MPI_BSEND (buf, count, datatype, dest, tag, comm)//основная операция посылки сообщения с определенным пользователем буфером

28- MPI_BSEND_INIT (buf, count, datatype, dest, tag, comm, request)//дескриптор для буферизованной посылки.

29- MPI_BUFFER_ATTACH (buffer, size) //Создает определяемый пользователем буфер для посылки

30- MPI_BUFFER_DETACH (buffer_addr, size)// Удаляет существующий буфер.

31- MPI_CANCEL (request) Отменяет коммуникационный запрос.

32- MPI_CART_COORDS (comm, rank, maxdims, coords)// Определяет координаты процесса в картезианской топологии в соответствии с его номером.

33-MPI_CART_CREATE (comm_old, ndims, dims, periods, reorder, comm_cart) // Создает новый коммуникатор по заданной топологической информации.

34- MPI_CART_GET (comm, maxdims, dims, periods, coords)// Получает связанную с коммуникатором информацию о картезианской топологии.

35- MPI_CART_MAP (comm, ndims, dims, periods, newrank)//Отображает процесс в соответствии с топологической информацией.

36- MPI_CART_RANK (comm, coords, rank)// Определяет номер процесса в коммуникаторе с картезианской топологией.

37- MPI_CART_SHIFT (comm, direction, disp, rank_source, rank_dest)//Возвращает новые номера процессов отправителя и получателя после сдвига в заданном направлении.

38- MPI_CART_SUB (comm, remain_dims, newcomm)// Разделяет коммуникатор на подгруппы, которые формируют картезианские решетки меньшей размерности.

39- MPI_CARTDIM_GET (comm, ndims)//Получает информацию о связанной с коммуникатором картезианской топологии.

40- MPI_COMM_COMPARE (comm1, comm2, result)//Сравнивает два коммуникатора.

41- MPI_COMM_CREATE (comm, group, newcomm)//Создает новый коммуникатор.

42- MPI_COMM_CREATE_ERRHANDLER (function, errhandler)//MPI-2:создает обработчик ошибок в стиле MPI.

43-MPI_COMM_CREATE_KEYVAL (Comm_copy_attr_fn,comm_delete_attr_fn, comm_keyval, extra_state)//MPI-2: генерирует новый ключ атрибута.

44- MPI_COMM_DELETE_ATTR (comm,comm_keyval)//MPI-2: удаляет связанное с ключом значение атрибута.

45- MPI_COMM_DUP (comm, newcomm)//Создает новый коммуникатор путем дублирования существующего совсеми его параметрами.

46- MPI_COMM_FREE (comm)// Маркирует коммуникатор для удаления.

47- MPI_COMM_FREE_KEYVAL (comm_keyval) MPI-2:освобождает ключ атрибута.

48- MPI_COMM_GET_ATTR (comm,comm_keyval, attribute, flag)// MPI-2: получает значение атрибута по ключу.

49- MPI_COMM_GET_ERRHANDLER (comm, errhandler)//MPI-2:Получает обработчик ошибок для коммуникатора.

50- MPI_COMM_GROUP (comm, group)/Осуществляет доступ к группе, связанной с данным коммуникатором.

51- MPI_COMM_RANK (comm, rank) //Определяет номер процесса в коммуникаторе.

51- MPI_COMM_REMOTE_GROUP (comm, group)// Возвращает удаленную группу в коммуникатор.

52- MPI_COMM_REMOTE_SIZE (comm, size)// Возвращает номер процесса в удаленной группе.

53- MPI_COMM_SET_ATTR (comm, comm_keyval, attribute_val)//Запоминает связанное с ключом значение атрибута.

54- MPI_COMM_SET_ERRHANDLER (comm, errhandler)//MPI-2: устанавливает обработчик ошибок для коммуникатора.

55- MPI_COMM_SIZE (comm, size) Определяет размер связанной с коммуникатором группы.

56- MPI_COMM_SPLIT (comm, color, key, newcomm)//Создает новый коммуникатор на основе признаков и ключей.

57- MPI_COMM_TEST_INTER (comm, flag)//Проверяет, является ли данный коммуникатор интеркоммуникатором.

58- MPI_DIMS_CREATE (nnodes, ndims, dims) Распределяет процессы по размерностям.

59- MPI_ERRHANDLER_CREATE (function, errhandler) Создает обработчик ошибок в стиле MPI Устарела, вместо нее используется MPI_COMM_CREATE_ERRHANDLER.

60- MPI_ERRHANDLER_FREE (errhandler)//Освобождает обработчик ошибок в стиле MPI.

61- MPI_ERRHANDLER_GET (comm, errhandler)//Создает обработчик ошибок для коммуникатора.

62- MPI_ERRHANDLER_SET (comm, errhandler)//Устанавливает обработчик ошибок для коммуникатора устарела,вместо нее используется MPI_COMM_SET_ERRHANDLER.

63- MPI_ERROR_CLASS (errorcode, errorclass)// Преобразует код ошибки в класс ошибки.

64- MPI_ERROR_STRING (errorcode, string, resultlen) //Возвращает строку для кода данной ошибки.

65- MPI_FINALIZE ()// Завершает выполнение программы MPI.

66- MPI_GATHER (sendbuf, sendcount, sendtype, recvbuf, recvcount,recvtype, root, comm)// Собирает в один процесс данные от группы процессов.

67- MPI_GATHERV (sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs, recvtype, root, comm) //Векторный вариант функции GATHER.

68- MPI_GET_ADDRESS (location, address)// MPI-2: получает адрес ячейки в памяти.

69- MPI_GET_COUNT (status, datatype, count)// Получает номер старших элементов.

70- MPI_GET_ELEMENTS (status, datatype, count)//Возвращает номер базового элемента в типе данных.

71- MPI_GET_PROCESSOR_NAME (name, resultlen)//Получает номер процессора.

72- MPI_GET_VERSION (version, subversion)// Возвращает версию MPI.

73- MPI_GRAPH_CREATE (comm_old, nnodes, index, edges, reorder,comm_graph)// Создает новый коммуникатор согласно топологической информации.

74- MPI_GRAPH_GET(comm, maxindex, maxedges, index, edges)//Получает информацию о связанной с коммуникатором графовой топологии.

75- MPI_GRAPH_MAP (comm, nnodes, index, edges, newrank)//Размещает процесс согласно информации о топологии графа.

76- MPI_GRAPH_NEIGHBORS_COUNT(comm, rank, nneighbors)//Возвращает число соседей узла в графовой топологии.

77- MPI_GRAPH_NEIGHBORS (comm, rank, maxneighbors, neighbors)//Возвращает соседей узла в графовой топологии.

78- MPI_GRAPHDIMS_GET (comm, nnodes, nedges)//Получает информацию о связанной с коммуникатором топологии.

79- MPI_GROUP_COMPARE (group1, group2, result) //Сравнивает две группы.

80- MPI_GROUP_DIFFERENCE (group1, group2, newgroup)//Создает группу по разности двух групп.

81- MPI_GROUP_EXCL (group, n, ranks, newgroup)//Создает группу путем переупорядочивания существующей группы и отбора только тех элементов, которые не указаны в списке.

82- MPI_GROUP_FREE (group)// Освобождает группу.

83- MPI_GROUP_INCL (group, n, ranks, newgroup)//Создает группу путем переупорядочивания существующей группы и отбора только тех элементов, которые указаны в списке.

84- MPI_GROUP_INTERSECTION (group1, group2, newgroup)// Создает группу на основе пересечения двух групп.

85- MPI_GROUP_RANGE_EXCL (group, n, ranges, newgroup)//Создает группу путем исключения ряда процессов из существующей группы.

86- MPI_GROUP_RANGE_INCL (group, n, ranges, newgroup)//Создает новую группу из ряда номеров существующей группы.

87- MPI_GROUP_RANK (group, rank) //Возвращает номер процесса в данной группе.

88- MPI_GROUP_SIZE (group, size)// Возвращает размер группы.

89- MPI_GROUP_TRANSLATE_RANKS (group1, n, ranks1, group2, ranks2)// Переводит номер процесса в одной группе в номер в другой группе.

90- MPI_GROUP_UNION(group1, group2, newgroup)// Создает новую группу путем объединения двух групп.

91- MPI_IBSEND (buf, count, datatype, dest, tag, comm, request)//Запускает неблокирующую буферизованную посылку.

92- MPI_INIT ()// Инициализация параллельных вычислений.

93- MPI_INITIALIZED (flag)// Указывает, был ли выполнен MPI_INIT.

94- MPI_INTERCOMM_CREATE (local_comm, local_leader, peer_comm, remote_leader, tag, newintercomm)// Создает интеркоммуникатор из двух интракоммуникаторов.

95- MPI_INTERCOMM_MERGE (intercomm, high, newintracomm)//Создает интракоммуникатор из интеркоммуникатора.

96- MPI_IPROBE (source, tag, comm, flag, status)// Неблокирующий тест сообщения.

97- MPI_Irecv (buf, count, datatype, source, tag, comm, request)//Начинает неблокирующий прием.

98- MPI_IRSEND (buf, count, datatype, dest, tag, comm, request)//Запускает неблокирующую посылку по готовности.

99- MPI_ISEND (buf, count, datatype, dest, tag, comm, request)//Запускает неблокирующую посылку.

100- MPI_ISSEND (buf, count, datatype, dest, tag, comm, request)//Запускает неблокирующую синхронную передачу.

101- MPI_KEYVAL_CREATE (copy_fn, delete_fn, keyval, extra_state)//Генерирует новый ключ атрибута Устарела, вместо нее используется MPI_COMM_CREATE_KEYVAL.

102- MPI_KEYVAL_FREE (keyval)// Освобождает ключ атрибута устарела, вместо нее используется MPI_COMM_FREE_KEYVAL.

103- MPI_OP_CREATE (function, commute, op)//Создает определенный пользователем дескриптор функции.

104-MPI_OP_FREE (op)// Освобождает определенный пользователем дескриптор функции.

105-MPI_PACK (inbuf, incount, datatype, outbuf, outsize, position, comm)//Упаковывает данные в непрерывный буфер.

106-MPI_PACK_SIZE (incount, datatype, comm, size)//Возвращает размер, необходимый для упаковки типа данных.

107-MPI_PCONTROL (level, ...) //Управляет профилированием.

108-MPI_PROBE (source, tag, comm, status) //Блокирующий тест сообщения.

109-MPI_RECV (buf, count, datatype, source, tag, comm, status)//Основная операция приема.

110-MPI_RECV_INIT(buf, count, datatype, source, tag, comm, request)//Создает дескриптор для приема.

111-MPI_REDUCE (sendbuf, recvbuf, count, datatype, op, root, comm)//Выполняет глобальную операцию над значениями всех процессов и возвращает результат в один процесс.

112-MPI_REDUCE_SCATTER (sendbuf, recvbuf, recvcounts, datatype,op, comm)// Выполняет редукцию и рассылает результаты.

113-MPI_REQUEST_FREE (request)//Освобождает объект коммуникационного запроса.

114-MPI_RSEND (buf, count, datatype, dest, tag, comm) //Операция отправки по готовности.

115-MPI_RSEND_INIT (buf, count, datatype, dest, tag, comm, request)// Создает дескриптор для отправки по готовности.

116-MPI_SCAN (sendbuf, recvbuf, count, datatype, op, comm)// Вычисляет частичную редукцию данных на совокупности процессов.

117-MPI_SCATTER (sendbuf, sendcount, sendtype, recvbuf, recvcount,recvtype, root,comm) //Рассылает содержимое буфера одного процессавсем процессам в группе.

118-MPI_SCATTERV(sendbuf, sendcounts, displs, sendtype, recvbuf,recvcount, recvtype, root, comm)// Рассылает части буфера одногопроцесса всем процессам в группе.

119-MPI_SEND_INIT (buf, count, datatype, dest, tag, comm, request)//Создает дескриптор для стандартной отправки.

120-MPI_SENDRECV (sendbuf, sendcount, sendtype, dest, sendtag,recvbuf, recvcount, recvtype, source, recvtag, comm, status) //Посылает ипринимает сообщение.

121-MPI_SENDRECV_REPLACE (buf, count, datatype, dest, sendtag,source, recvtag,comm, status)// Посылает и принимает сообщение,используя один буфер.

122-MPI_SSEND_INIT (buf, count, datatype, dest, tag, comm, request)//Создает дескриптор для синхронной передачи.

123-MPI_STARTALL (count, array_of_requests)// Запускает совокупность запросов

124-MPI_TEST(request, flag, status) Проверяет завершение отправки или приема.

125-MPI_TESTALL (count, array_of_requests, flag, array_of_statuses)//Проверяет завершение всех ранее начатых операций обмена.

126-MPI_TESTANY (count, array_of_requests, index, flag, status)//Проверяет завершение любой ранее начатой операции.

127-MPI_TESTSOME (incount, array_of_requests, outcount,array_of_indices,array_of_statuses) //Проверяет завершение заданных операций.

128-MPI_TEST_CANCELLED (status, flag)// Проверяет отмену запроса.

129-MPI_TOPO_TEST (comm, status) //Определяет тип связанной с коммуникатором топологии.

130-MPI_TYPE_COMMIT(datatype)// Объявляет тип данных.

131-MPI_TYPE_CONTIGUOUS (count, oldtype, newtype)// Создает непрерывный тип данных.

132-MPI_TYPE_EXTENT(datatype, extent)// Создает экстенст типа данных.

133-MPI_TYPE_FREE (datatype)// Отмечает объект типа данных для удаления.

134-MPI_TYPE_HINDEXED (count, array_of_blocklengths,array_of_displacements,oldtype, newtype)// MPI-2: создает индексированный тип данных со смещением в байтах.

135-MPI_TYPE_HVECTOR (count, blocklength, stride, oldtype, newtype)//MPI-2: создает векторный тип данных со смещением в байтах.

136-MPI_TYPE_INDEXED (count, array_of_blocklengths, array_of_displacements, oldtype, newtype)// Создает индексированный тип данных.

137-MPI_TYPE_LB (datatype, displacement)//Возвращает нижнюю границу типа данных Устарела, используется MPI_TYPE_GET_EXTENT.

138-MPI_TYPE_SIZE (datatype, size)//Возвращает число байтов, занятых элементами типа данных.

140-MPI_TYPE_STRUCT (count, array_of_blocklengths,array_of_displacements, array_of_types, newtype)// Создает новый тип данных Устарела, используется MPI_TYPE_CREATESTRUCT.

141-MPI_TYPE_UB (datatype, displacement)//Возвращает верхнюю границу типа данных Устарела, используется MPI_TYPE_GET_EXTENT.

142-MPI_UNPACK (inbuf, insize, position, outbuf, outcount, datatype,comm)// Распаковывает данные из непрерывного буфера.

143-MPI_WAIT (request, status) //Ожидает завершения посылки или приема.

144-MPI_WAITANY (count, array_of_requests, index, status)//Ожидает завершения любой из описанных посылки или приема.

145-MPI_WAITSSOME (incount, array_of_requests, outcount, array_of_indices, array_of_statuses)// Ожидает завершения некоторых заданных обменов.

146-MPI_WTIME ()//Возвращает полное время выполнения операций на используемом процессоре.

147- MPI_COMM_NULL_COPY_FN Возвращает flag = false и MPI_SUCCESS.

148-MPI_COMM_DUP_FN//Возвращает значение attribute_val_in в attribute_val_out и устанавливает MPI_SUCCESS.

149-MPI_COMM_NULL_DELETE_FN //Возвращает MPI_SUCCESS.

150-MPI_FILE_OPEN// открывает файл с именем filename для всех процессов из группы коммутатора comm.

151-int MPI_File_close(MPI_File *fh) MPI_FILE_CLOSE(FH, IERROR) INTEGER FH, IERROR void MPI::File::Close() MPI_FILE_CLOSE //сначала синхронизирует состояние файла(эквивалент исполнению MPI_FILE_SYNC), затем закрывает файл, ассоциированный с fh.

152-MPI_FILE_DELETE// удаляет файл, определяемый именем файла filename. Если такого файла не существует, MPI_FILE_DELETE генерирует ошибку класса MPI_ERR_NO_SUCH_FILE.

153-MPI_FILE_SET_SIZE// изменяет размер файла, ассоциированного с дескриптором fh. Размер измеряется в байтах от начала файла.
MPI_FILE_SET_SIZE //коллективная; все процессы в группе должны устанавливать одно и то же значение size. Если новый размер меньше, чем текущий размер файла, файл урезается до позиции, определяемой новым размером. Для реализации необязательно удалять из памяти участки файла, расположенные за этой позицией.

154-MPI_FILE_PREALLOCATE// обеспечивает пространство для хранения первых size байтов файла, ассоциированного с fh.
MPI_FILE_PREALLOCATE //коллективная; все процессы в группе должны устанавливать одно и то же значение size.

155-MPI_FILE_GET_SIZE// возвращает в size текущий размер в байтах файла, ассоциированного с дескриптором файла fh. Что касается семантики согласованности, MPI_FILE_GET_SIZE - операция доступа к данным.

156-MPI_FILE_GET_GROUP// возвращает дубликат группы коммутатора, использованной для открытия файла, ассоциированного с fh. Эта группа возвращается в group.
Ответственность за освобождение памяти group лежит на пользователе.

157-mpi_file_get_atomicity//возвращает атомарный режим.

158- mpi_file_get_amode// возвращает режим доступа к файлу.

159-mpi_file_get_byte_offset// возвращает абсолютную позицию байта в файле, соответствующую смещению в типах элемента , относительно текущего представления.

- 160- `mpi_file_get_info`//возвращает подсказки для файла ,которыефактически используются `mpi`.
- 161- `mpi_file_get_position`//возвращает текущую позицию индивидуального указателя файла,измеренную в единицах типа элемента,относительно текущего представления.
- 162- `mpi_file_get_position_shared`//возвращает текущую позицию разделяемого указателя файла в единицах типа элемента,относительно текущего представления.
- 163-`mpi_file_get_type_extent`//возвращает протяженность типа данных в файле.
- 164-`mpi_file_get_view`//возвращает представление файла .
- 165- `mpi_file_iread`// неблокирующее чтение ,использующее индивидуальный указатель файла.
- 166- `mpi_file_iread_at`//неблокирующее чтение , использующее явное смещение.
- 167-`mpi_file_iread_shared`// неблокирующее чтение, использующее разделяемый указатель файла.
- 168- `mpi_file_iwrite`//неблокирующая запись , использующая индивидуальный указатель файла.
- 169- `mpi_file_iwrite_at`// неблокирующая запись, использующая явное смещение.
- 170- `mpi_file_iwrite_shared`- неблокирующая запись ,использующая разделяемый указатель файла.
- 171- `mpi_file_read`-чтение и использованием индивидуального указателя файла.
- 172- `mpi_file_read_all`- коллективное чтение ,использующее индивидуальный указатель файла.
- 173- `mpi_file_read_all_begin` //начинает раздельное коллективное чтение ,использующее индивидуальный указатель файла.
- 174- `mpi_file_read_all_end`//завершает раздельное коллективное чтение, использующее индивидуальный указатель файла.
- 175- `mpi_file_read_at`// чтение с использованием явного смещения.
- 176- `mpi_file_read_at_all`// коллективное чтение ,использующее явное смещение.
- 177- `mpi_file_read_at_all_begin`//начинает раздельное коллективное чтение ,использующее явное смещение.
- 178- `mpi_file_read_at_all_end`//завершение раздельное коллективное чтение, использующее явное смещение.
- 179-`mpi_file_read_ordered`//коллективное чтение , использующее разделяемый указатель файла.
- 180- `mpi_file_read_ordered_begin`// начинает раздельное коллективное чтение , использующее разделяемый указатель файла.
- 181- `mpi_file_read_ordered_end`//завершает раздельное коллективное чтение ,использующее разделяемый указатель файла.
- 182- `mpi_file_read_shared`//чтение с использованием разделяемого указателя файла.
- 183- `mpi_file_seek`//модифицирует индивидуальный указатель файла.
- 184- `mpi_file_seek_shared`// модифицирует разделяемый указатель файла.
- 185-`mpi_file_set_atomicity`// устанавливает атомарный режим .
- 186- `mpi_file_set_info`// устанавливает новые значения для подсказок ,связанных с файлами.
- 187- `mpi_file_set_size`-//устанавливает размер файла.
- 188- `mpi_file_set_view`//устанавливает представление файла.
- 189-`mpi_file_sync`// заставляет все предыдущие записи быть перемещенными на устройство хранения .
- 190- `mpi_file_write`// запись, использующая индивидуальный указатель файла.
- 191- `mpi_file_write_all`// коллективное запись, использующая индивидуальный указатель файла.
- 192- `mpi_file_write_all_begin`// начинает раздельную коллективную запись, использующую индивидуальный указатель файла.

- 193- `mpi_file_write_all_end`//завершает отдельную коллективную запись, использующую индивидуальный указатель файла.
- 194- `mpi_file_write_at`//запись ,использующая явное смещение.
- 195- `mpi_file_write_at_all`//коллективная запись ,использующая явное смещение.
- 196- `mpi_file_write_at_all_begin`// начинает отдельную коллективную запись,использующую явное смещение.
- 197- `mpi_file_write_at_all_end`// завершает отдельную коллективную запись, использующую явное смещение .
- 198- `mpi_file_write_ordered`//коллективная запись ,использующая разделяемый указатель файла.
- 199- `mpi_file_write_ordered_begin`// начинает отдельную коллективную запись ,использующую разделяемый указатель файла.
- 200- `mpi_file_write_ordered_end`//завершает отдельную коллективную запись, использующую разделяемый указатель файла.
- 201- `mpi_file_write_share`// запись ,использующая разделяемый указатель файла.
- 202- `mpi_graph_create`//создает новый коммуникатор ,к которому присоединяется информация о топологии.
- 203- `mpi_graph_get`//возвращает информацию о топологии графа,связанной с коммуникатором.
- 204- `mpi_graph_map`//отображает процесс на информацию о топологии графа .
- 205- `mpi_graph_neighbors`// возвращает соседей узла, связанного с топологией графа.
- 206- `mpi_graph_neighbors_count`//возвращает число соседей узла, связанного с топологией графа.
- 207- `mpi_graphdims_get`// возвращает информацию о топологии графа,связанной с коммуникатором.
- 208-`mpi_info_create`//создает новый информационный объект.
- 209-`mpi_info_delete`//удаляет пару (ключ, значение) из информации.
- 210-`mpi_info_dup`//возвращает дубликат информационного объекта.
- 211-`mpi_info_free`// освобождает информационный объект.
- 212-`mpi_info_get`// возвращает значение, связанное с ключом.
- 213-`mpi_info_get_nkeys`//возвращает число определенных на текущий момент ключей в информации.
- 214-`mpi_info_get_nthkey`//возвращает n-ый определенный ключ в информации.
- 215- `mpi_info_get_valuelen`// возвращает длину значения, связанного с ключом.
- 216-`mpi_info_set`//добавляет пару (ключ, значения) к информации.
- 217-`mpi_intercomm_create`//создает интеркоммуникатор из двух интеркоммуникаторов .
- 218- `mpi_intercomm_merge`//создает интракоммуникатор из интеркоммуникатора .
- 219-`mpi_keyval_create`//генерирует новый ключ атрибута.
- 220- `mpi_keyval_free`//освобождает ключ атрибута для атрибута КЭШа коммуникатора.
- 221- `Mpi_abort`// прерывает выполнение `mpi` седы.
- 222- `Mpi_address`// Получает адрес расположения в памяти.
- 223- `Mpi_allgatherv` //собирает данные из всех задач и распределяет их всем задачам.
- 224- `Mpi_allgather` //собирает данные из всех задач и поставит их всем задачам.
- 225-`Mpi_allreduce`//объединяет значения из всех процессов и распределяет результат всем процессам.
- 226-`Mpi_alltoall`//посылает данные от всех процессов всем.
- 227- `Mpi_alltoallv` // посылает данные от всех процессов всем , со смещением.
- 228- `Mpi_barrier`//блокирует процесс ,пока все процессы достигнут этой подпрограммы.
- 229- `Mpi_bcast`//рассылает сообщение из процесса с номером `l` корень `l` всем другим процессам группы.
- 230-`MPI_Bsend`//основная посылка сообщения с указанной пользователем буферизацией.
- 231- `MPI_Bsend_init`//формирует дескриптор для буферизованной

посылки.

232- `Mpi_buffeer_attach`//присоединяет определяемый пользователем буфер к посылке.

233-`Mpi_buffeer_datach`//удаляет существующий буфер (для использования в `mpi_bsend` т.п.)

234-`Mpi_cancel`// отменяет запрос связи.

235- `Mpi_buffeer_attach`//сравнивает два коммуникатора.

236-`Mpi_buffeer_create`// создает новый коммуникатор.

237- `MPI_Comm_dup`//дублирует существующий коммуникатор со всей кешируемой информацией.

238-`MPI_Comm_free`//помечает объект коммуникатора для освобождения.

239-`MPI_Comm_group`// выбирает группу ,связанную с данным коммуникатором.

240-`Mpi_comm_rank`//определяет номер вызывающего процесса в коммуникаторе.

241-`Mpi_comm_remote_group`//выбирает удаленную группу связанную с данным интеркоммуникатором.

242-`MPI_Comm_remote_group`//выбирает удаленную группу,связанную с данным интеркоммуникатором.

243-`Mpi_comm_remote_size`//определяет размер удаленной группы связанной с интеркоммуникатором.

244-`Mpi_comm_split`// создает новые коммуникаторы ,основываясь на ветках и ключах(`split-разбиение`).

245-`MPI_GATHER`// собирает вместе значения из группы процессов.

246-`Mpi_gatherv`// собирает в определенные места из всех процессов в группе.

247-`MPI_Get_count`//получает число элементов верхнего уровня.

248-`MPI_Get_element`// возвращает число основных элементов в типе данных.

249-`MPI_Get_processor_name`// получает имя процессора.

250-`MPI_Get_version`//получает версию `mpi`.

251-`Mpi_group_compare` // Сравнивает две группы.

252-`MPI_Group_difference`// делает группу из разности двух групп.

253-`Mpi_group_excl`// производит группу ,переупорядочивая существующую группу и принимая только перечисленные члены.

254-`MPI_Group_free`//освобождает группу .

255-`Mpi_group_incl`//производит группу, переупорядочивая существующую группу и принимая только перечисленные члены.

256-`MPI_Group_intersection`//производит группу как пересечение двух существующих групп.

257-`Mpi_group_range_excl`//производит группу ,исключая диапазоны процессов из существующей группы.

258-`Mpi_group_range_incl`// создает новую группу из диапазонов номеров в существующей группе.

259-`Mpi_group_rank`//возвращает номер текущего процесса в заданной группе.

260-`Mpi_group_size`//возвращает размер группы.

261-`MPI_GROUP_TRANSLATE_RANKS`// транслирует номера процессов в одной группе в их номера в другой группе.

262-`MPI_Group_union`//производит группу ,объединяя две группы.

263-`MPI_Ibsend`// начинает нелокирующую буферизированную посылку.

264-`MPI_Init_thread`//инициализирует выполнение среды `mpi`.

265-`MPI_Iprobe`// неблокирующий тест для сообщения .

266-`Mpi_irecv`//начинает неблокирующий прием.

267-`Mpi_irsend`//начинает неблокирующую готовую посылку.

268-`Mpi_isend`//начинает неблокирующую посылку.

169-`Mpi_issend`//начинает неблокирующую синхронную посылку.

270-`Mpi_op_create`//создает определяемый пользователем дескриптор комбинационной функции.

271-Mpi_or_free// освобождает определяемый пользователем дескриптор комбинационной функции.

272-Mpi_pack//упаковывает тип данных в непрерывную память.

273-Mpi_pack_size// возвращает верхнюю границу пространства, необходимого для упаковки сообщения.

Mpi_pcontrol//управление профилированием.

274-Mpi_probe//блокирующий тест для сообщения .

275-Mpi_recv//основной прием сообщения.

276-Mpi_recv_init//формирует дескриптор для приема.

277-Mpi_reduce//приводит значения из всех процессов к одиночному значению.

278-Mpi_reduce_scatter//объединяет значения и рассеивает результаты.

279-Mpi_request_free//освобождает объект запроса связи.

280-Mpi_rsend// Основная готовая посылка сообщения .

281-Mpi_rsend_init//формирует дескриптор для готовой посылки.

282-Mpi_scan// вычисляет развертку данных для совокупности процессов.

283-Mpi_scatter// Посылает данные из одной задачи всем остальным задачам в группе.

284-Mpi_scatterv//рассеивает буфер по частям всем задачам в группе.

285-Mpi_send// выполняет основную посылку сообщения.

286-Mpi_send_init// формирует дескриптор для стандартной посылки.

287-Mpi_sendrecv//посылает и получает сообщение.

288-Mpi_sendrecv_replace//посылка и прием ,использующие одиночный буфер.

289-Mpi_ssend//основная синхронная посылка сообщения.

290-Mpi_ssend_init// формирует дескриптор для синхронной посылки.

291-Mpi_start//инициализирует связь с постоянным дескриптор запроса.

292-Mpi_startall//стартует совокупность запросов.

293-MPI_Status_set_cancelled//устанавливает непрозрачную часть.

294-MPI_Status так// чтобы Status_test_cancelled возвратил флажок.

295-MPI_status_set_elements // устанавливает непрозрачную часть.

296-MPI_status// так, чтобы MPI_status_get_elements возвратил счетчик.

297-Mpi_test//тест завершения посылки или приема.

298-Mpi_test_cancelled// тест , определяющий был ли отменен запрос.

299-Mpi_testall//тест завершения всех предварительно инициализированных связей.

300-MPI_Testany//тест завершения любой из предварительно инициализированных связей.

301-MPI_Testsome//тест завершения каких-либо из указанных связей.

302-Mpi_type_commit//регистрирует тип данных .

303-Mpi_type_contiguous//создает непрерывный тип данных.

304-Mpi_type_create_darray//создает тип данных, соответствующий распределенному многомерному массиву.

305-Mpi_type_create_indexed_block//создает индексированный тип данных с блоками постоянного размера.

306-Mpi_type_create_subarray//создает тип данных , описывающий под массив многомерного массива.

307-Mpi_type_extent//возвращает протяженность типа данных.

308-Mpi_type_free//освобождает тип данных.

309-Mpi_type_get_contents//возвращает фактические параметры , использованные в вызове при создании типа данных.

310-Mpi_type_get_envelope//возвращает информацию относительно номера и типа входных параметров, использованных при вызове, который создал тип данных.

311-Mpi_type_hindexed//создает индексированный тип данных со смещениями в байтах.

312-Mpi_type_hvector//создает векторный (прореженный) тип данных со смещением в байтах.

313-Mpi_type_indexed//создает индексированный данных.
314-Mpi_type_lb//возвращает нижнюю границу типа .
315-Mpi_type_size//возвращает число байтов, занятых записями в типе данных.
316-Mpi_type_struct//создает структурный тип данных.
317-Mpi_type_ub//возвращает верхнюю границу типа .
318-Mpi_type_vector//создает векторный тип.
319-Mpi_unpack//распаковывает тип данных из непрерывной памяти.
320-Mpi_wait//ждет завершения посылки или приема.
321-Mpi_waitall// ждет завершения всех указанных связей.
322-Mpi_waitany// ждет завершения любых из указанных посылок или приемов.
323- Mpi_waitsome// ждет завершения каких-либо из указанных связей.
324-Mpi_wtick//возвращает разрешающую способность Mpi_wtime.
325-Mpi_wtime// возвращает прошедшее время на вызывающем процессоре.

Список литературы

1. Аль-хулайди А.А. Анализ существующих пакетов в кластерных сетях / А.А. Аль-хулайди, Н.Н. Садовой // Вестник Донского гос. техн. ун-та. — 2010. — Т. 10. — №3 (46). — С. 303–310.
2. Аль-хулайди А.А. «распределенные вычисления (кластерные вычисления) с использованием пакета параллельного программирования MPI»//журнал "Современные наукоемкие технологии" ,2010. – №4.