# Лабораторная работа №5 Использование технологии mpjexpress для распределенных вычислений

## 2.1 Цель работы

Установка и настройка библиотеки mpjexpress под операционными системами Windows и Linux/Unix. Конфигурирование библиотеки для разработки и выполнения распределенных приложений Java на вычислительных кластерах или компьютерных сетях.

## 2.2 Теоретические сведения

MPJ Express (message passing library) – библиотека передачи сообщений, которая может быть использована для разработки и выполнения параллельных приложений Java на вычислительных кластерах или компьютерных сетях.

Вычислительные кластеры являются популярной параллельной платформой, которая широко используется сообществами высокопроизводительных вычислений (HPC) для крупномасштабной вычислительной работы.

MPJ Express – это промежуточное ПО которое поддерживает связь между отдельными процессорами кластеров.

Хотя MPJ Express предназначен для машин с распределенной памятью, таких, как сеть компьютеров или кластеров, можно эффективно выполнять параллельные пользовательские приложения на настольных компьютерах или ноутбуках, которые содержат общую память или многоядерные процессоры.

## 2.2.1 Конфигурация

Программное обеспечение MPJ Express можно настроить двумя способами, как показано на рисунке 2.1. Первый способ конфигурации известный как многоядерная конфигурация – используется для выполнения пользовательских MPJ Express программ на ноутбуках и настольных ПК. Второй способ конфигурации известный как кластерная конфигурация –используется для выполнения пользовательских программ MPJ Express на кластерах или сетях компьютеров. Кластерная конфигурация основывается на устройствах для общения. В настоящее время существует четыре устройства связи для кластерной конфигурации:

1. Java New I/O (NIO) Device Driver – устройство, известное как niodev: niodev используется для выполнения пользовательских MPJ Express программ на кластерах используя Ethernet.
2. Myrinet Device Driver – устройство, известное как mxdev: mxdev используется для выполнения пользовательских MPJ Express программ на кластерах соединенных высокоскоростными межсоединениями Myrinet сети. В настоящее время mxdev недоступен под ОС windows.
3. Hybrid Device Driver – устройство, известное как hybdev: hybdev используется для выполнения пользовательских MPJ Express программ на кластерах из многоядерных компьютеров.
4. Native Device Driver – устройство, известное как native: native используется для выполнения пользовательских MPJ Express программ как надстройку над родной библиотекой MPI (MPICH, Open MPI or MS-MPI).

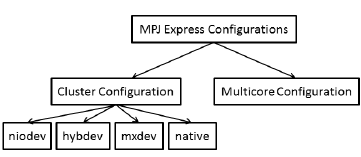


Рисунок 2.1 – Конфигурации MPJ Express

#### 2.2.2 Многоядерная конфигурация

Многоядерная конфигурация предназначена для пользователей, которые планируют писать и выполнять параллельные Java приложения, использующие MPJ Express на своих ПК или ноутбуках, как правило, такое аппаратное обеспечение содержит общую память и многоядерные процессоры. В этой конфигурации, пользователи могут создавать свои параллельные приложения, используя MPJ Express и он будет автоматически перенесен на многоядерные процессоры. Пользователи могут сначала разработать приложения на своих ноутбуках и настольных ПК с использованием многоядерной конфигурации, а затем использовать тот же самый код на платформах с распределенной памятью в том числе кластерах. Также эта конфигурация является предпочтительной в учебных целях, так как студенты могут выполнить код на своих личных ноутбуках и настольных ПК. Стоит отметить, что пользовательские приложения остаются неизменными при выполнении кода в многоядерной или кластерной конфигурации.

#### 2.2.3 Кластерная конфигурация

Кластерная конфигурация предназначается для пользователей, которые планируют выполнить свои параллельные Java приложения на платформах с распределенной памятью, включая кластеры или сети компьютеров.

В качестве примера, рассмотрим кластер или сеть компьютеров, показанный на рисунке 2.2. Он показывает шесть вычислительных узлов, связанных друг с другом с помощью частного межсоединения. Кластерная конфигурация MPJ Express начнет один MPJ Express процесс для каждого узла, которые взаимодействуют друг с другом с помощью передачи сообщений.

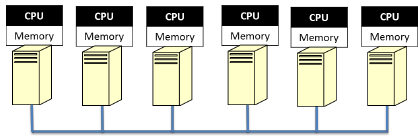


Рисунок 2.2 – Кластерная конфигурация MPJ Express нацеленная на платформы с распределенной памятью, включая кластеры или сети компьютеров

Разработчики приложений могут выбрать для использования любой из четырех устройств связи в кластерной конфигурации:

1. Java New I/O (NIO) device driver (niodev)

2. Myrinet device driver (mxdev)

3. Hybrid device driver (hybdev)

4. Native device driver (native)

## 2.3 Выполнение работы

Для использования MPJ Express должны быть предустановлены:

• Java 1.6 (stable) или выше.

• Apache ant 1.6.2 или выше (необязательно): ant необходим для компиляции исходных кодов MPJ Express.

• Perl (необязательно): MPJ Express необходим Perl для компилирования исходных кодов, потому что некоторые Java коды генерируются из Perl шаблонов. Файл сборки генерирует Java файлы из Perl шаблонов если он обнаруживает perl на машине. Это хорошая идея, установить Perl если вы хотите разрабатывать используя MPJ Express.

• Библиотека MPI (необязательно): библиотека MPI, такая как MS-MPI необходима для запуска MPJ Express в кластерной конфигурации с устройством native.

## 2.3.1 Настройка MPJ Express под Linux/Unix

#### Установка MPJ Express

В этом разделе описываются действия для загрузки и установки MPJ Express на Linux/Unix:

1. Загрузите и распакуйте MPJ Express.
2. Установите значение переменных среды MPJ\_HOME и PATH.

a. export MPJ\_HOME=/path/to/mpj/

b. export PATH=$MPJ\_HOME/bin:$PATH

Данные строки должны быть добавлены в файл “.bashrc”. Убедитесь в том, что интерпретатор, в котором вы устанавливаете переменные является интерпретатором по умолчанию. Например, если вашим интерпретатором по умолчанию является ‘bash’, тогда вы можете настраивать переменные среды в .bashrc. Если же вы используете ‘tcsh’ или какой-либо другой интерпретатор, тогда устанавливайте переменные среды в соответствующих файлах.

4. Создайте новую рабочую директорию для MPJ Express программ. Предполагается, что рабочая директория – “mpj-user”.

5. Скомпилируйте MPJ Express библиотеку (необязательно):

cd $MPJ\_HOME; ant

#### Компилирование пользовательской программы

В этом разделе показано, как скомпилировать простую, распределенную Java программу.

1. Напишите Hello World MPJ Express программу и сохраните ее как HelloWorld.java

import mpi.\*;

public class HelloWorld {

public static void main(String args[]) throws Exception {

MPI.Init(args);

int me = MPI.COMM\_WORLD.Rank();

int size = MPI.COMM\_WORLD.Size();

System.out.println("Hi from <"+me+">");

MPI.Finalize();

}

}

2. Компилирование: javac –cp .:$MPJ\_HOME/lib/mpj.jar HelloWorld.java

#### Выполнение MPJ Express в многоядерной конфигурации

В данном разделе описаны шаги для выполнения распределенной Java программы в многоядерной конфигурации.

1. Предположим, что пользователь успешно прошел предыдущие этапы.

2. Для запуска HelloWorld

Выполнение: mpjrun.sh -np 2 HelloWorld

3. Запуск тестов (необязательно) [Тестовые утилиты поставляются вместе с MPJ Express]

a. Компилирование (необязательно): cd $MPJ\_HOME/test; ant

b. Выполнение: mpjrun.sh -np 2 -jar $MPJ\_HOME/lib/test.jar

#### Выполнение MPJ Express в кластерной конфигурации

В данном разделе описаны шаги для выполнения распределенной Java программы в кластерной конфигурации с использованием драйверов niodev , mxdev, hybdev и native.

1. Предположим, что пользователь успешно прошел предыдущие этапы.
2. Напишите файл машин указанием названий машин, IP адресов или псевдонимов узлов, на которых вы хотите выполнить MPJ Express процессы. Сохраните этот файл как 'machines' в директории mpj-user. Этот файл используется такими скриптами : mpjboot, mpjhalt, mpjrun.sh для того, чтобы узнать с какими машинами связаться.

Предположим, вы хотите запустить процессы на каждой из машин 'machine1' и 'machine2 ', тогда ваш файл машин будет представлен в виде:

machine1

machine2

Запомните, что в реальном мире 'machine1' и 'machine2' должны быть полностью квалифицированные имена, IP адреса или псевдонимы ваших машин.

1. Запустите демон: mpjdaemon.bat –boot

Это должно работать, если $MPJ\_HOME/bin был успешно добавлен в переменную $PATH. Этот скрипт подключается по ssh к каждой машине указанной в файле машин и запускает демоны. Если включено ведение логов, тогда каждый демон создаст лог файл с именем daemon-<machine\_name>.log в $MPJ\_HOME/logs директории.

1. Запуск HelloWorld

Выполнение : mpjrun.sh -np 2 -dev niodev HelloWorld

1. Запуск тестов (необязательно)

Выполнение: mpjrun.sh -np 2 –dev niodev –jar $MPJ\_HOME/lib/test.jar

1. Остановка демонов : mpjhalt machines

Для того, чтобы использовать один из драйверов niodev , mxdev, hybdev или native укажите его название после ключа –dev.

Для того, чтобы задать количество машин, на которых будет выполняться программа, укажите необходимое количество после ключа –np.

## 2.3.2 Настройка MPJ Express под Windows

#### Установка MPJ Express

В этом разделе описываются действия для загрузки и установки MPJ Express на Windows:

1. Загрузите и распакуйте MPJ Express.
2. Предпологая, что 'mpj express' распакован в 'c:\mpj', нажмите правой кнопкой мыши на иконке My Computer, перейдите в My Computer PropertiesAdvanced tabEnvironment Variables and export the following system variables.

a. Установите значение переменной MPJ\_HOME в c:\mpj [рис. 2.3]

b. Дополните переменную Path значением c:\mpj\bin [рис 2.4]

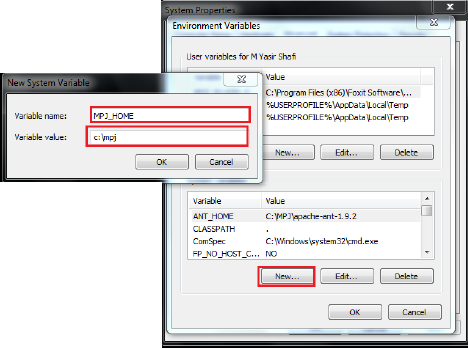


Рисунок 2.3 – Добавление переменной среды MPJ\_HOME

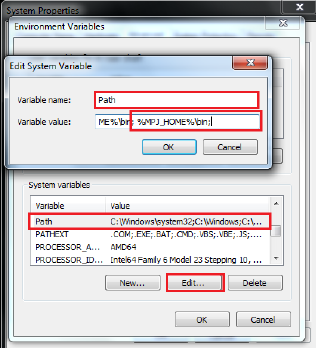


Рисунок 2.4 – Добавление значение в переменную Path

1. Для windows с использованием Cygwin (полагая „mpj express‟ в „c:\mpj‟) . Если вы хотите, вы можете установить переменные в cygwin shell

export MPJ\_HOME="c:\\mpj"

export PATH=$PATH:"$MPJ\_HOME\\bin"

4. Создайте новую рабочую директорию для MPJ Express программ. Предполагается, что рабочая директория – “mpj-user”.

5. Скомпилируйте MPJ Express библиотеку (необязательно):

cd %MPJ\_HOME%; ant

#### Компилирование пользовательской программы

В этом разделе показано, как скомпилировать простую, распределенную Java программу.

1. Напишите Hello World MPJ Express программу и сохраните ее как HelloWorld.java

import mpi.\*;

public class HelloWorld {

public static void main(String args[]) throws Exception {

MPI.Init(args);

int me = MPI.COMM\_WORLD.Rank();

int size = MPI.COMM\_WORLD.Size();

System.out.println("Hi from <"+me+">");

MPI.Finalize();

}

}

2. Компилирование: javac –cp .;%MPJ\_HOME%/lib/mpj.jar HelloWorld.java

#### Выполнение MPJ Express в многоядерной конфигурации

В данном разделе описаны шаги для выполнения распределенной Java программы в многоядерной конфигурации.

1. Предположим, что пользователь успешно прошел предыдущие этапы.

2. Для запуска HelloWorld

Выполнение: mpjrun.bat -np 2 HelloWorld

3. Запуск тестов (необязательно) [Тестовые утилиты поставляются вместе с MPJ Express]

a. Компилирование (необязательно): cd %MPJ\_HOME%/test; ant

b. Выполнение: mpjrun.bat -np 2 -jar %MPJ\_HOME%/lib/test.jar

#### Выполнение MPJ Express в кластерной конфигурации

В данном разделе описаны шаги для выполнения распределенной Java программы в кластерной конфигурации с использованием драйверов niodev, hybdev и native.

1. Предположим, что пользователь успешно прошел предыдущие этапы.
2. Напишите файл машин указанием названий машин, IP адресов или псевдонимов узлов, на которых вы хотите выполнить MPJ Express процессы. Сохраните этот файл как 'machines' в директории mpj-user. Этот файл используется такими скриптами : mpjboot, mpjhalt, mpjrun.bat и mpjrun.sh для того, чтобы узнать с какими машинами связаться.

Предположим, вы хотите запустить процессы на каждой из машин 'machine1' и 'machine2 ', тогда ваш файл машин будет представлен в виде:

machine1

machine2

Запомните, что в реальном мире 'machine1' и 'machine2' должны быть полностью квалифицированные имена, IP адреса или псевдонимы ваших машин.

1. Запустите демон: mpjdaemon.bat –boot

Это должно работать, если %MPJ\_HOME%/bin был успешно добавлен в переменную %PATH%. Вы должны выполнить эту команду на каждой машине для запуска демона. Если включено ведение логов, тогда каждый демон создаст лог файл с именем daemon-<machine\_name>.log в %MPJ\_HOME%/logs директории.

1. Запуск HelloWorld

Выполнение : mpjrun.bat -np 2 -dev niodev HelloWorld

1. Запуск тестов (необязательно)

Выполнение: mpjrun.bat -np 2 –dev niodev –jar %MPJ\_HOME%/lib/test.jar

1. Остановка демонов : mpjdaemon.bat -halt

Для того, чтобы использовать один из драйверов niodev, hybdev или native укажите его название после ключа –dev.

Для того, чтобы задать количество машин, на которых будет выполняться программа, укажите необходимое количество после ключа –np.

## Примеры пользовательских программ

Пример файла mashines:

192.168.56.10

192.168.56.11

Пример программы HelloWorld с передачей и приемом на разных машинах:

package main;

import javax.swing.text.MaskFormatter;

import mpi.\*;

public class HelloWorld {

static public void main(String[] args) {

MPI.Init(args) ;

int myrank = MPI.COMM\_WORLD.Rank() ;

if(myrank == 0) {

char [] message = "Hello, there".toCharArray() ;

MPI.COMM\_WORLD.Send(message, 0, message.length, MPI.CHAR, 1, 99) ;

}

else {

char [] message = new char [20] ;

MPI.COMM\_WORLD.Recv(message, 0, 20, MPI.CHAR, 0, 99) ;

System.out.println("received:" + new String(message) + ":") ;

}

MPI.Finalize();

}

}

Пример программы с передачей текста от нескольких машин и множественным приемом одной из машин:

import mpi.\*;

import java.io.\*;

import java.nio.ByteBuffer;

public class MPITest

{

public static void main(String[] args)

{

MPI.Init(args);

int me = MPI.COMM\_WORLD.Rank();

int tasks = MPI.COMM\_WORLD.Size();

MPI.COMM\_WORLD.Barrier();

if(me == 0)

{

Cat cat = new Cat("Tom", 15);

cat.Speak();

ByteBuffer byteBuff = ByteBuffer.allocateDirect(2000 + MPI.SEND\_OVERHEAD);

MPI.Buffer\_attach(byteBuff);

try

{

ByteArrayOutputStream bos = new ByteArrayOutputStream();

ObjectOutput out = null;

out = new ObjectOutputStream(bos);

out.writeObject(cat);

byte[] bytes = bos.toByteArray();

System.out.println("Serialized to " + bytes.length);

MPI.COMM\_WORLD.Isend(bytes, 0, bytes.length, MPI.BYTE, 1, 0);

}

catch(IOException ex)

{

}

}

else

{

byte[] bytes = new byte[2000];

Cat recv = null;

MPI.COMM\_WORLD.Recv(bytes, 0, 2000, MPI.BYTE, MPI.ANY\_SOURCE, 0);

ByteArrayInputStream bis = new ByteArrayInputStream(bytes);

ObjectInput in = null;

try

{

in = new ObjectInputStream(bis);

Object obj = in.readObject();

recv = (Cat)obj;

recv.Speak();

}

catch(IOException ex)

{

}

catch(ClassNotFoundException cnf)

{

}

}

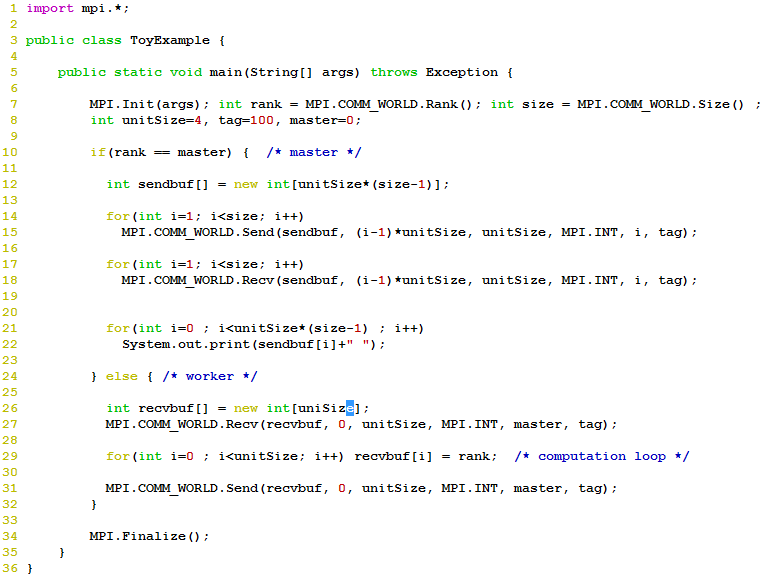
MPI.COMM\_WORLD.Barrier();

MPI.Finalize();

}

}

Пример программы обмена информацией между несколькими машинами:



alex@alex:~/mpj-user$ mpjrun.sh -np 5 ToyExample

MPJ Express (0.43) is started in the multicore configuration

1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4

Пример программы с коллективной передачей:

