

MPJ Express: An Implementation of MPI in Java Windows User Guide 18th July 2014

Document Revision Track

Version	Updates	Ву
1.0	Initial version document	Aamir Shafi
1.1	A new device 'hybdev' is added for executing parallel Java	Aleem Akhtar, Mohsan
	applications exploiting hybrid parallelism.	Jameel, Aamir Shafi
1.2	A new device 'native' is added for executing parallel Java	Bibrak Qamar, Mohsan
	applications on top of a native MPI library.	Jameel, Aamir Shafi
1.3	Runtime updated and support for running Java	Aleem Akhtar, Aamir
	applications on non-shared file system added. New scripts	Shafi, Mohsan Jameel
	for daemons are also added.	
1.4	Improved collective primitives are added in MPJ Express.	Aleem Akhtar, Aamir
	Some minor bugs are fixed	Shafi, Mohsan Jameel

Table of Contents

1	Int	roduction	5
	1.1	Configurations	.5
	1.1.	1 Multicore configuration	.6
	1.1.	2 Cluster configuration	.6
2	Ge	tting Started with MPJ Express	8
	2.1 Pr	e-requisites	.8
	2. 2 In	stalling MPJ Express	.9
	2.3 Co	mpiling User Applications	13
	2.4 Ru	nning MPJ Express in the Multi-core Configuration1	13
	2.5 Ru	nning MPJ Express in the Cluster Configuration1	٤4
	2.5.	1 Cluster Configuration with niodev1	٤4
	2.5.	2 Cluster Configuration with hybdev1	۱5
	2.5.	4 Cluster Configuration with native device (using a native MPI library)	۱5
	2.6 Ac	lvanced Options to mpjrun.bat2	24
3	MI	PJ Express Debugging2	5
	3.1	The mpjrun Script	25
	3.2	Core Library2	25
	3.3	MPJ Express Daemons (Cluster configuration only)2	25
4	Kn	own Issues and Limitations2	6
5	Co	ntact and Support2	7
A	open	dices2	8
		idix A: Running MPJ Express on non-shared file system	
		idix B: Running MPJ Express without the runtime (manually)	
	1.1.2.		

Appendix C: Changing protocol limit switch	30
Appendix D: MPJ Express Testsuite	31
Compiling source code and Testsuite	31
Running Testsuite	31
Appendix E: Useful scripts for MPJ Daemons	32
Appendix F: Switching to Old Collectives	34

1 Introduction

MPJ Express is a reference implementation of the mpiJava 1.2 API, which is an MPI-like API for Java defined by the Java Grande forum. The mpiJava 1.2 API is the Java equivalent of the MPI 1.1 specification document (<u>http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html</u>).

This release of the MPJ Express software contains the core library and the runtime infrastructure. The software also contains a comprehensive test suite that is meant to test the functionality of various communication functions.

MPJ Express is a message passing library that can be used by application developers to execute their parallel Java applications on compute clusters or network of computers. Compute clusters is a popular parallel platform, which is extensively used by the High Performance Computing (HPC) community for large scale computational work. MPJ Express is essentially a middleware that supports communication between individual processors of clusters. The programming model followed by MPJ Express is Single Program Multiple Data (SPMD).

Although MPJ Express is designed for distributed memory machines like network of computers or clusters, it is possible to efficiently execute parallel user applications on desktops or laptops that contain shared memory or multicore processors.

1.1 Configurations

The MPJ Express software can be configured in two ways, as shown in Figure 1. The first configuration—known as the multicore configuration—is used to execute MPJ Express user programs on laptops and desktops. The second configuration—known as the cluster configuration—is used to execute MPJ Express user programs on clusters or network of computers. The cluster configuration relies on devices for communication. Currently there are four communication devices for the cluster configuration:

- 1. Java New I/O (NIO) device known as niodev: niodev is used to execute MPJ Express user programs on clusters using Ethernet.
- 2. Myrinet device known as mxdev: mxdev is used to execute MPJ Express user programs on clusters connected by Myrinet express interconnects. Currently mxdev is not supported under windows.
- 3. Hybrid device known as hybdev: hybdev is used to execute MPJ Express user programs on clusters of multicore computers.

4. Native device known as native: native is used to execute MPJ Express user programs on top of a native MPI library (MPICH, Open MPI or MS-MPI).

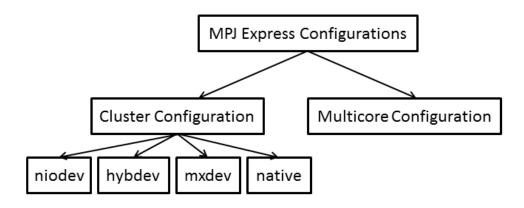


Figure 1: MPJ Express configurations

1.1.1 Multicore configuration

The multicore configuration is meant for users who plan to write and execute parallel Java applications using MPJ Express on their desktops or laptops—typically such hardware contains shared memory and multicore processors. In this configuration, users can write their message passing parallel application using MPJ Express and it will be ported automatically on multicore processors. We envisage that users can first develop applications on their laptops and desktops using multicore configuration, and then take the same code to distributed memory platforms including clusters. Also this configuration is preferred for teaching purposes since students can execute message passing code on their personal laptops and desktops. It might be noted that user applications stay the same when executing the code in multicore or cluster configuration.

Under the hood, the MPJ Express library starts a single thread to represent an MPI process. The multicore communication device uses efficient inter-thread mechanism.

1.1.2 Cluster configuration

The cluster configuration is meant for users who plan to execute their parallel Java applications on distributed memory platforms including clusters or network of computers.

As an example, consider a cluster or network of computers shown in Figure 2. It shows shows six compute nodes connected to each other via private interconnect. The MPJ Express cluster

configuration will start one MPJ Express process per node, which communicates to each other using message passing.

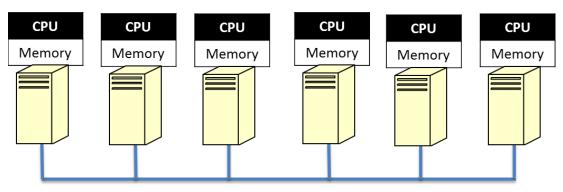


Figure 2: MPJ Express Cluster Configuration Targets the Distributed Memory Platforms Including Clusters and Network of Computers

Application developers can opt to use either of the four communication devices in the cluster configuration:

- 5. Java New I/O (NIO) device driver known as niodev
- 6. Myrinet device driver known as mxdev
- 7. Hybrid device driver known as hybdev
- 8. Native device driver known as native

The Java NIO device driver (also known as niodev) can be used to execute MPJ Express programs on clusters or network of computers. The niodev device driver uses Ethernet-based interconnect for message passing. On the other hand, many clusters today are equipped with high-performance low-latency networks like Myrinet. MPJ Express also provides a communication device for message passing using Myrinet interconnect—this device is known as mxdev and is implemented using the Myrinet eXpress (MX) library by Myricom. These communication drivers can be selected using command line switches.

Modern HPC clusters are mainly equipped with multicore processors (Figure 3). The hybrid device is meant for users who plan to execute their parallel Java applications on such a cluster of multicore machines. Hybrid device transparently uses both multicore configuration and cluster configuration for intra-node communication and cluster configuration (NIO device only) for inter-node communication, respectively.

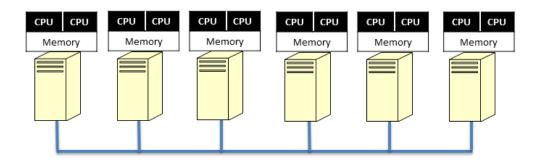


Figure 3: MPJ Express Hybrid Configuration Targeting Cluster of Multicore Machines

The fourth device—native device—is meant for users who plan to execute their parallel Java applications using a native MPI implementation for communication. With this device bulk of messaging logic is offloaded to the underlying MPI library. This is attractive because MPJ Express can exploit latest features, like support for new interconnects and efficient collective communication algorithms, of the native MPI library. Under Windows, this device is currently tested and supported for MS-MPI—as the underlying native MPI library.

2 Getting Started with MPJ Express

This section shows how MPJ Express programs can be executed in the multicore, cluster and hybrid configuration

2.1 Pre-requisites

- Java 1.6 (stable) or higher (Mandatory).
- Apache ant 1.6.2 or higher (Optional): ant is required for compiling MPJ Express source code.
- Perl (Optional): MPJ Express needs Perl for compiling source code because some of the Java code is generated from Perl templates. The build file will generate Java files from Perl templates if it detects perl on the machine. It is a good idea to install Perl if you want to do some development with MPJ Express.
- A native MPI library (Optional): Native MPI library such as MS-MPI is required for running MPJ Express in cluster configuration with native device.

• Visual Studio (Optional): MPJ Express needs Visual Studio to build JNI wrapper library for the native device.

2. 2 Installing MPJ Express

This section outlines steps to download and install MPJ Express software.

- 1. Download MPJ Express and unpack it
- 2. Assuming unpacked 'mpj express' is in 'c:\mpj', Right-click My Computer→Properties→Advanced tab→Environment Variables and export the following system variables (user variables are not enough)
 - a. Set the value of variable MPJ_HOME as c:\mpj [see Fig 4,Fig 5 and Fig 6]
 - b. Append the value of variable Path as c:\mpj\bin [see Fig 7]

See the snapshots below

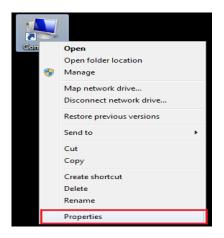


Figure 4: Right click on my computer and select Properties

💮 💬 – 🕎 ト Control Panel 🕨		Q
Control Panel Home Device Manager Remote settings System protection Advanced system settings 	System Properties Computer Name Hardware Advanced System Protection Remote You must be logged on as an Administrator to make most of these changes. Performance Visual effects, processor scheduling, memory usage, and virtual memory Visual effects, processor scheduling, memory usage, and virtual memory Settings User Profiles Desktop settings related to your logon	•
	Startup and Recovery System startup, system failure, and debugging information Settings Hz 3.00 GHz Play	III
See also Action Center Windows Update	OK Cancel Apply Change settings Computer description: Workgroup: WORKGROUP	
Performance Information and Tools	Windows activation Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now Windows now	-

Figure 5: Select Environment Variables to Add/Edit variables

	Syst	em Properties		X
	En	wironment Variable	25	× 1
		User variables for M	1 Yasir Shafi	
New System Variab	le	23	Value	
			C:\Program Files (x86)\Foxit Software\ %USERPROFILE%\AppData\Local\Temp	
Variable name:	MPJ_HOME		%USERPROFILE% AppData Local Temp	
Variable value:	c: \mpj			
	(cr trib)		New Edit Delete	
	OK	Cancel		
		Variable	Value	<u> </u>
		ANT_HOME	C:\MPJ\apache-ant-1.9.2	
		CLASSPATH	C.))// [
		ComSpec FP_NO_HOST_C	C:\Windows\system32\cmd.exe NO	-
			New Edit Delete	
			OK Cance	

Figure 6: Add MPJ_HOME as new Environment Variable

System Properties		×
Environment Variab	les	×
Edit System Variable		
Variable name:	Path	
Variable value:	ME%\bir; %MPJ_HOME%\bin;	
	OK Cancel]
System variables		
Variable	Value	
Path	C:\Windows\system32;C:\Windows;C:\	
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;	J
PROCESSOR_A.		
PROCESSOR_ID	Intel64 Family 6 Model 23 Stepping 10, 🔻	
	New Edit Delete	
	OK Cancel	

Figure 7: Append Path variable

3. For windows with Cygwin (assuming 'mpj express' is in 'c:\mpj')

The recommended way to is to set variables as in Windows

If you want to set variables in cygwin shell

export MPJ_HOME="c:\\mpj" export PATH=\$PATH:"\$MPJ_HOME\\bin"

- 4. Create a new working directory for MPJ Express programs. This document assumes that the name of this directory is "mpj-user".
- 5. Compile the MPJ Express library (Optional): cd %MPJ_HOME%; ant

2.3 Compiling User Applications

This section shows how to compile a simple Hello World parallel Java program.

1. Write Hello World MPJ Express program and save it as 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. Compile: javac -cp .;%MPJ_HOME%/lib/mpj.jar HelloWorld.java

2.4 Running MPJ Express in the Multi-core Configuration

This section outlines steps to execute parallel Java programs in the multicore configuration.

- 1. Assuming the user has successfully carried out Section 2.2 and Section 2.3
- 2. Running HelloWorld

Execute: mpjrun.bat -np 2 HelloWorld

- 3. Running test cases (Optional) [Test suite is provided with MPJ Express]
 - a. Compile (Optional): cd %MPJ_HOME%/test; ant
 - b. Execute: mpjrun.bat -np 2 -jar %MPJ_HOME%/lib/test.jar

2.5 Running MPJ Express in the Cluster Configuration

This section outlines steps to execute parallel Java programs in the cluster configuration with three communication device drivers including niodev, hybdev and native.

2.5.1 Cluster Configuration with niodev

This section outlines steps to execute parallel Java programs in the cluster configuration with niodev communication device driver.

- 1. Assuming the user has successfully carried out Sections 2.2 and 2.3.
- 2. Write a machines file stating machines name, IP addresses, or aliases of the nodes where you wish to execute MPJ Express processes. Save this file as 'machines' in mpj-user directory. This file is used by scripts like mpjboot, mpjhalt, mpjrun.bat and mpjrun.sh to find out which machines to contact.

Suppose you want to run a process each on 'machine1' and 'machine2', then your machines file would be as follows

machine1 machine2

Note that in real world, 'machine1' and 'machine2' would be fully qualified names, IP addresses or aliases of your machine

3. Start the daemons: mpjdaemon.bat -boot

This should work if <code>%MPJ_HOME%/bin</code> has been successfully added to <code>%PATH%</code> variable. You will need to run this command on each machine to start daemons. If logging is enabled then each daemon produces a log file named <code>daemon-<machine_name>.log</code> in <code>%MPJ_HOME%/logs</code> directory.

4. Running HelloWorld

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

5. Running test cases (Optional) [Test suite is provided with MPJ Express]

Execute: mpjrun.bat -np 2 -dev niodev -jar %MPJ_HOME%/lib/test.jar

6. Stop the daemons: mpjdaemon.bat -halt

After you are done with executing all the programs, make sure that you halt the daemons at each machine.

2.5.2 Cluster Configuration with hybdev

This section outlines steps to execute parallel Java programs in the hybrid configuration using multicore and cluster configurations. Hybrid configuration depends on Multicore configuration and Cluster configuration. Make sure that document sections **2.4** and **2.5.1** are completed successfully.

- 1. Start the daemons: mpjdaemon.bat -boot
- 2. Running HelloWorld

Execute: mpjrun.bat -np 2 -dev hybdev HelloWorld

3. Running test cases (Optional) [Test suite is provided with MPJ Express]

Execute: mpjrun.bat -np 2 -dev hybdev -jar %MPJ_HOME%/lib/test.jar

4. Stop the daemons: mpjdaemon.bat -halt

2.5.4 Cluster Configuration with native device (using a native MPI library)

This section outlines steps to execute parallel Java programs in the cluster configuration with native device.

- 1. Assuming the user has successfully carried out Section 2.2 and Section 2.3.
- 2. Since MPJ Express native device relies on a native MPI it is assumed that the user has installed and tested the native MPI library. Better to run a simple helloworld like program to test the native MPI. Currently MPJ Express is only tested on MS-MPI (under Windows).

By design MPJ Express should work with any native MPI library. If you have a different native MPI library installed on your system, please feel free to test it and let us know.

3. Compile the JNI wrapper library (Mandatory):

This requires Visual Studio to generate a dynamic library (nativempjdev.dll) to be used by MPJ Express to interface with the native MPI library. Open Visual Studio and follow the steps provided below:

New Project				2 X
▶ Recent	.NE	T Framework 4.5 - Sort by: Default	- # 1	Search Installed Templates (Ctrl+E)
 Installed Templates Visual Basic Visual C# Visual C++ ATL CLR General MFC Test Win32 Visual F# TypeScript Python Other Project T Samples 		Win32 Console Application MFC Application Win32 Project	Visual C++ Visual C++ Visual C++ Visual C++ Visual C++	Type: Visual C++ A project for creating a Win32 application, console application, DLL, or static library
▶ Online		<u>Click here to go online ar</u>	<u>id find templates.</u>	
Name:	nativempjdev			
Location:	c:\users\n	documents\visual studio 2013\Projects\mp	oj 🔹	Browse
Solution name:	nativempjdev			Create directory for solution Add to source control OK Cancel

a. File \rightarrow New \rightarrow Project: Create a Win32 Project with the name of **nativempjdev**

Figure 15: Create a Win32 Project with the name of nativempjdev

b. Click next→ set Application type as DLL and in Additional options tick Empty project → finish

Win32 Application Wizard - nat	ivempjdev	? ×
Applica	tion Settings	
Overview Application Settings	Application type:	Add common header files for:
	<pre></pre>	vious Next > Finish Cancel

Figure 16: Set Application type as DLL and in Additional options tick Empty project

c. Right click on project **nativempjdev** in the Solution Explorer and go to properties. Set Additional Include Directories

Configuration:	Release	▼ Platform: Active(x64)	Configuration Manager	
Genera Debug VC++ C/C++ Gen Op Pre Coi Lar Pre Ou Bro Ou Bro Adi	ging Directories Etimization processor de Generation iguage compiled Heade tput Files wase Information vanced Options mmand Line	Additional Include Directories Additional #using Directories Debug Information Format Common Language RunTime Support Consume Windows Runtime Extension Suppress Startup Banner Warning Level Treat Warnings As Errors SDL checks Multi-processor Compilation	Additional Include Directories C:\Program Files\Java\jdk1.6.0_45\include\Vin32 C:\Program Files\Java\jdk1.6.0_45\include C:\Program Files\Microsoft MPI\Inc\amd64 C:\Program Files\Microsoft MPI\Inc III	8 8 8

Figure 17: Set Additional Include Directories

d. Set Additional Library Directories in the Linker

onfiguration:	Release	Platform:	Active(x6	i4) 🔹	Configuration Mar	nager	
Out	put Files	Output File		\$(OutDir)\$(TargetName)\$(T	argetExt)		
Brow	wse Information	Show Progress		Not Set		199	
Adv	anced	Version					
	Options	Enable Incremental Link	ting	No (/INCREMENTAL:NO)			
	nmand Line	Suppress Startup Banne	r	Yes (/NOLOGO)			
▲ Linker		Ignore Import Library		No		E	
	eral	Register Output		No			
Inpu	Mark and a second s	Per-user Redirection		No			
0.077	nifest File	Additional Library Direc	tories	~		ິ	Explore
	ugging	Link Library Dependenc		Yes			
Syst		Use Library Dependency		Additional Library Directo			2
	imization	Link Status	No. All Chi	Additional Library Directo	nes	(
	dows Metadata	Prevent DII Binding					
	anced	Treat Linker Warning As	Errors		A 1 198 1 1 1 1 1 1 1 1 1 1 1 1		
	Options	Additional Library Director		C:\Program Files\Micros	oft MPI\Lib\amd64		
	nmand Line +	Allows the user to override		ne			
	100 m 100	Allows the user to overhide	are environi				F.
c <u> </u>				-			
				Inherited values:			
		and a second					
	- (1 = 1 =						
						-	
				Inherit from parent or	project defaults	1	Macros>
				10 March 10		1000	

Figure 18: Set Additional Include Directories

e. Set Additional Dependencies (msmpi.lib) in the Linker

onfiguration:	Release	▼ Platforr	m: Active(x64)	▼ Configu	iration Manager	
Brov Adv All C	out Files vse Information anced Dptions mmand Line	Additional Dependence Ignore All Default Libr Ignore Specific Default Module Definition File Add Module to Assen Embed Managed Resc	aries t Libraries e nbly	iser32.lib;gdi32.lib;winspo Il Dependencies	ol.lib;comdlg32.lib;	8 ×
	ifest File ⋿ ugging	Force Symbol Referen Delay Loaded Dils Assembly Link Resour	ces msmpi.	J		
Emb Win	mization vedded IDL dows Metadata anced		Inherited			
	Options Immand Line +	Additional Dependencie Specifies additional items	s user32.l	ib o ol.lib		Ĩ
	•]1 = 1 =		Inheri	t from parent or project d	lefaults	Macros>>

f. Right click on 'Header Files'→Add→Existing Item... to add Header Files

			Solutio	on Explorer	▼ ₽		
			66) 🟠 🔞 - 🤄 🗇 🕼 🖌 -	-		
			Search	Solution Explorer (Ctrl+;)	\$		
			Solution 'nativempjdev' (1 project) Mathematical Structure (1 project) Mathematical Structure (1 project)				
				🚛 External Dependencies			
°	New Item	Ctrl+Shift+A		Add 🦾	+		
Co.	Existing Item	Shift+Alt+A	e.	Class Wizard	Ctrl+Shift+X		
*	New Filter			Scope to This			
**	Class			New Solution Explorer View			
**	Resource		Ж	Cut	Ctrl+X		
			D	Сору	Ctrl+C		
			ĉ	Paste	Ctrl+V		
			×	Delete	Del		
			X:::	Rename			
			لكر	Properties	Alt+Enter		
			_				

Figure 19: Navigate to Header Files under the solution nativempjdev

g. Browse into %MPJ_HOME% \src\mpjdev lib and select the header files (*.h)

Organize 🔻 🛛 New fold	ler		III •	
Microsoft Visual St	Name	Date modified	Туре	Size
Projects	🕌 build	3/28/2014 4:39 PM	File folder	
	h mpjdev_natmpjdev_Comm.h	3/19/2014 5:34 AM	C/C++ Header	1
🔆 Favorites	h mpjdev_natmpjdev_Group.h	3/19/2014 5:34 AM	C/C++ Header	1
📃 Desktop	mpjdev_natmpjdev_Intercomm.h	3/19/2014 5:34 AM	C/C++ Header	j
📕 Downloads 🛛 🗉	h mpjdev_natmpjdev_Intracomm.h	3/19/2014 5:34 AM	C/C++ Header	1
强 Recent Places	npjdev_natmpjdev_NativeRecvRequest.	h 3/19/2014 5:34 AM	C/C++ Header	
	mpjdev_natmpjdev_NativeSendRequest.	h 3/19/2014 5:34 AM	C/C++ Header	1
浔 Libraries	h mpjdev_natmpjdev_shared.h	B/19/2014 5:34 AM	C/C++ Header	
Documents Music Pictures Videos				
Computer	•	m		

Figure 20: Add header files

a. Right click on 'Source Files'→Add→Existing Item... to add Source Files

Organize 👻 New folde	er		== -	
Microsoft Visual St	Name	Date modified	Туре	Size
📙 Projects	\mu build	3/28/2014 4:39 PM	File folder	
	C mpjdev_natmpjdev_Comm.c	4/3/2014 10:40 AM	C Source	2
😭 Favorites	C mpjdev_natmpjdev_Group.c	3/19/2014 5:34 AM	C Source	1
🧮 Desktop	C mpjdev_natmpjdev_Intercomm.c	3/19/2014 5:34 AM	C Source	3
🚺 Downloads 🛛 🗉	C mpjdev_natmpjdev_Intracomm.c	3/19/2014 5:34 AM	C Source	2
📃 Recent Places	C mpjdev_natmpjdev_NativeRecvRequest.c	3/19/2014 5:34 AM	C Source	1
	C mpjdev_natmpjdev_NativeSendRequest.c	3/19/2014 5:34 AM	C Source	
Documents Music Pictures Videos				
Computer	٠ [
		atmpidev Group 👻 🖌	All Files (*.*)	

Figure 21: Add source files

b. Right click on nativempjdev Solution and Build. This creates the dynamic library (nativempjdev.dll)

			on Explorer		▼ ₽ 3
		00	© ÷ ≈ @	P -	
		Search	Solution Explorer (Ctr	rl+;)	P
		s S	olution 'nativempjdev	' (1 project)	
E	Build	2	nativempjdev		
	Rebuild		Header Files	encies	
	Clean			npjdev_Comm.h	
	View	,	> D mpjdev_natr		
	Analyze	•	14	npjdev_Intercomm.h npjdev_Intracomm.h	
	Project Only	,		npjdev_Intracomm.n npjdev_NativeRecvReque	st.h
	Scope to This			npjdev_NativeSendReque	
Ē	New Solution Explorer View		14	npjdev_shared.h	
	Profile Guided Optimization	•	Resource Files		
	Build Dependencies		++ mpjdev_natr	npjdev_Comm.c	
	Add	r	> ++ mpjdev_natr	12 - 1	
₽.	Add Class Wizard	Ctrl+Shift+X		npjdev_Intercomm.c	
₩.		Ctri+Shirt+X	1 12 -	npjdev_Intracomm.c npjdev_NativeRecvReque:	et c
-	Manage NuGet Packages		14	npjdev_NativeSendReque	
ø					
	Debug	•	-		
	Source Control	•	_		
ж	Cut	Ctrl+X	Explorer Team Exp	olorer Class View	
â	Paste	Ctrl+V	es		т ф :
×	Remove	Del	pjdev Project Prop	erties	
χ.	Rename		- <u>-</u>		
	Unload Project				
	Rescan Solution		e)	nativempjdev	
\$	Open Folder in File Explorer		t Dependencies		
æ	Properties	Alt+Enter	:t File		docume
		ROC	n Namespace	nativempjdev	

Figure 22: Build

c. Install the newly created 'native mpjdev.dll'. Copy from your project \folder \x64 Release (or whatever x32 Release or \x64 Debug etc)

R Looks	Indian Triangle of Street,	1.10			_	
😋 🔵 🗢 🗾 « Visual St	udio 2013 🕨 Projects 🕨 mpj	j ▶ native	mpjdev ▶ x64 ▶ Release	▼ 4 ₇	Search Releas	se
Organize 🔻 🔳 Oper	n with Share with 🔻	Burn	New folder			
🔆 Favorites 💻 Desktop	Documents libra Release	iry				Arra
Downloads 🗐 Recent Places	Name		Date modified	Туре	Size	
Tecent Places	🐏 nativempjdev.pdb		4/3/2014 11:35 AM	Program Debug D	267 KB	
😂 Libraries	🔛 nativempjdev.lib		4/3/2014 11:35 AM	Object File Library	21 KB	
Documents	🗐 nativempjdev.exp		4/3/2014 11:35 AM	Exports Library File	13 KB	
Music Pictures Videos Computer Local Disk (C:) Local Disk (D:) Network	ativempjdev.dl			Annlication extension * and email	28 KB	

Figure 23: Copy nativempjdev.dll

d. Paste 'nativempjdev.dll' into MPJ_HOME lib

Organize 🔻 🛛 Inclu	ıde in	library 🔻 Share with	▼ Burn New for	older	
🔆 Favorites	^	Name #1 10g4]-1,2,111,jar	Date modified	Type Executable Jar File	Size 543 Kt
E Desktop		🖃 mpi.jar	3/31/2014 4:50 PM	Executable Jar File	160 KE
Downloads		mpiExp.jar	3/31/2014 4:50 PM	Executable Jar File	6 KE
🔠 Recent Places		📓 mpj.jar	3/31/2014 4:50 PM	Executable Jar File	296 KE
	- Miles	🔟 mpjbuf.jar	3/31/2014 4:50 PM	Executable Jar File	39 KE
📕 Libraries		🔳 mpjdev.jar	3/31/2014 4:50 PM	Executable Jar File	34 KE
Documents		📓 mxdev.jar	3/31/2014 4:50 PM	Executable Jar File	9 KI
J Music		lativempjdev.dll	4/3/2014 11:35 AM	Application extens	28 KE
Pictures		🖃 niodev.jar	3/31/2014 4:50 PM	Executable Jar File	39 KE
Videos		🔳 smpdev.jar	3/31/2014 4:50 PM	Executable Jar File	23 KE
		📓 starter.jar	3/31/2014 4:50 PM	Executable Jar File	24 KE
Computer		📓 test.jar	3/31/2014 2:55 PM	Executable Jar File	483 KE
Local Disk (C:)		📕 tools.jar	10/22/2006 8:08 AM	Executable Jar File	55 KE
👝 Local Disk (D:)		🚳 wrapper.dll	6/23/2007 5:49 AM	Application extens	80 KE
		📓 wrapper.jar	6/23/2007 5:49 AM	Executable Jar File	82 KE
📭 Network	+	xdev.jar	3/31/2014 4:50 PM	Executable Jar File	2 KE

Figure 24: Paste nativempjdev.dll into %MPJ_HOME%\lib

4. Running HelloWorld

Execute: mpjrun.bat -np 2 -dev native HelloWorld

- 5. Running test cases (Optional)
 - a. Compile :

cd ``%MPJ HOME%"/test/nativetest

compile.bat

- b. Execute: cd ``%MPJ_HOME%"/test/nativetest runtest.bat
 - i. To supply a machine file provide full path in the first argument of this script: runtest.bat /full/path/to/machinefile

Advanced Options:

Running directly with mpiexec to use options provided by native MPI library

This is for the advanced user who wants to run parallel Java programs using custom options to the native MPI library.

The mpjrun.bat script provides a wrapper to native mpiexec (mpirun) command. The user can bypass mpjrun.bat and directly call mpiexec using the following template.

```
mpiexec -np <number of processes> -machinefile <path\to\file\filename> java -cp
"%MPJ_HOME%"/lib/mpj.jar;. -Djava.library.path="%MPJ_HOME%"/lib HelloWorld 0 0
native userarg1 userarg2 userarg3
```

The above template consists of three parts: **mpiexec**, **java** and **user application**. In this way the user has flexibility to supply three kinds of options:

- 1. mpiexec: these are supplied to native MPI library bootstrapping framework (a.k.a mpirun or mpiexec), for example -np and -machinefile
- 2. java: these are supplied to the JVM for example -cp and -Djava.library.path and more.
- 3. user application: these are supplied to the user application for example userarg1 userarg2 userarg3 in the above template. The three arguments 0 0 native following user application (classname or jar) are reserved for MPJ Express and are to be kept intact. MPJ Express for conventional reasons searches for device name on argument index 3 (i.e args[2]).

2.6 Advanced Options to mpjrun.bat

1. JVM arguments (Optional): JVM arguments may be specified to the mpjrun script that passes these directly to the executing MPJ Express processes. For example, the following command modifies the JVM heap size:

mpjrun.bat -np 2 -Xms512M -Xmx512M HelloWorld

 Application Arguments (Optional): Users may pass arguments to their parallel applications by specifying them after "-jar <jarname>" or "classname" in the mpjrun script:

a. The user may pass three arguments "a", "b", "c" to the application as follows: <code>mpjrun.bat -np 2 HelloWorld a b c</code>

b. Application arguments can be accessed in the program by calling the string[] MPI.Init(String[] args) method. The returned array stores user arguments [a,b,c].

String appArgs[] = MPI.Init(args);

3 MPJ Express Debugging

This section shows how to debug various modules of the MPJ Express software. It is possible to debug MPJ Express on three levels:

- 1. The mpjrun Script: This script allows bootstrapping MPJ Express programs in cluster of multicore configuration.
- 2. Core Library: Internals of the MPJ Express Software
- 3. MPJ Express Daemons: While running the cluster configuration, daemons execute on compute nodes and are responsible for starting and stopping MPJ Express processes when contacted by the mpjrun script.

3.1 The mpjrun Script

To turn ON debugging for the mpjrun script, follow these steps:

- 1. Edit %MPJ_HOME%/conf/mpjexpress.conf file.
- 2. Change the value of mpjexpress.mpjrun.loglevel from "OFF" to "DEBUG".
- 3. The mpjrun script relevant log file is /current/directory/mpjrun.log file

3.2Core Library

To turn ON debugging for the core library, follow these steps:

- 1 Edit $MPJ_HOME\%/conf/mpjexpress.conf$ file
- 2 Change the value of mpjexpress.mpi.loglevel from "OFF" to "DEBUG"
- 3 If the total number of MPJ Express processes is two, then the relevant log files will be %MPJ_HOME%/logs/user_name-mpj-0.log and %MPJ_HOME%/logs/user_name-mpj-1.log for processes 0 and 1 respectively.

3.3 MPJ Express Daemons (Cluster configuration only)

The MPJ Express daemons running on compute nodes can be debugged using following steps:

- 1. Edit %MPJ_HOME%/conf/mpjexpress.conf file.
- 2. Change the value of mpjexpress.mpjdaemon.loglevel from "OFF" to "DEBUG".

3. Now log files can be seen in <code>\$MPJ_HOME/logs/daemon-<machine_name>.log file.</code>

4 Known Issues and Limitations

A list of known issues and limitations of the MPJ Express software are listed below.

- 1. The merge operation is implemented with limited functionality. The processes in localgroup and remote-group *have* to specify 'high' argument. Also, the value specified by local-group processes should be opposite to remote-group processes.
- 2. Any message sent with MPI.PACK can only be received by using MPI.PACK as the datatype. Later, MPI.Unpack(..) can be used to unpack different datatypes
- 3. Using 'buffered' mode of send with MPI.PACK as the datatype really does not use the buffer specified by MPI.Buffer_attach(..) method.
- 4. Cartcomm.Dims_Create(..) is implemented with limited functionality. According to the MPI specifications, non-zero elements of 'dims' array argument will not be modified by this method. In this release of MPJ Express, all elements of 'dims' array are modified without taking into account if they are zero or non-zero.
- 5. Request.Cancel(..) is not implemented in this release.
- 6. MPJ applications should not print more than 500 characters in one line. Some users may use system.out.print(..) to print more than 500 characters. This is not a serious problem, because printing 100 characters 5 times with system.out.println(..) will have the same effect as printing 500 characters with one system.out.print(..)
- 7. Some users may see this exception while trying to start the mpjrun module. This can happen when the users are trying to run mpjrun.bat script. The reason for this error is that the mpjrun module cannot contact the daemon and it tries to clean up the resources it has. In doing so, it tries to delete a file named 'mpjdev.conf' using File.deleteOnExit() method. This method appears not to work on Windows possibly because of permission issues.

```
Exception in thread "main" java.lang.RuntimeException: Another mpjrun module is
already running on this machine
at runtime.starter.MPJRun.(MPJRun.java:135)
at runtime.starter.MPJRun.main(MPJRun.java:925)
```

This issue can be resolved by deleting mpjdev.conf file. This file would be present in the directory, where your main class or JAR file is present. So for example, if the users are trying to run "-jar ../lib/test.jar", then this file would be present in ../lib directory.

8. The MPJ Express infrastructure does not deal with security. The MPJ Express daemons could be a security concern, as these are Java applications listening on a port to execute user-code. It is therefore recommended that the daemons run behind a suitably configured firewall, which only listens to trusted machines. In a normal scenario, these daemons would be running on the compute-nodes of a cluster, which are not accessible to outside world. Alternatively, it is also possible to start MPJ Express processes 'manually', which could help avoid runtime daemons. In addition, each MPJ Express process starts at least one server socket, and thus is assumed to be running on machine with configured firewall. Most MPI implementations assume firewalls as protection mechanism from the outside world.

5 Contact and Support

For help and support, join and post on the MPJ Express mailing list (<u>https://lists.sourceforge.net/lists/listinfo/mpjexpress-users</u>). Alternatively, you may also contact us directly:

- 1 Aamir Shafi (aamir.shafi@seecs.edu.pk)
- 2 Mohsan Jameel (<u>mohsan.jameel@seecs.edu.pk</u>)
- 3 Bryan Carpenter (<u>bryan.carpenter@port.ac.uk</u>)
- 4 Muhammad Ansar Javed (<u>muhammad.ansar@seecs.edu.pk</u>)
- 5 Bibrak Qamar (<u>bibrak.qamar@seecs.edu.pk</u>)
- 6 Aleem Akhtar (aleem.akhtar@seecs.edu.pk)

Appendices

Appendix A: Running MPJ Express on non-shared file system

MPJ Express applications can be executed on both shared file system and non-shared file system. Steps to run on both file systems are quite similar. Current version of MPJ Express supports running of MPJ Express applications in cluster mode on non-shared file system with niodev, hybdev and mxdev devices. Following steps should be performed to execute MPJ Express applications on non-shared file system:

- 1. Install MPJ Express on all machines where you want to execute your application. You can follow section 2.1, 2.2 and 2.3 for setting up environment for MPJ Express on each machine.
- 2. Once MPJ Express is installed, use mpjdaemon.bat script (see Appendix D) to boot daemons on each machine. You will need to manually boot daemons on each machine.
- 3. Write machines file on your host system from where you want to run your application and write down machine name, IP addresses, or aliases of the machines where you wish to execute MPJ Express processes. Make sure daemons are running that those machines.
- 4. Use -src switch with mpjrun script to enable working of MPJ Express on non-shared file system. Example commands are given below:

-- niodev: mpjrun.bat -np 2 -dev niodev -src HelloWorld --- hybdev: mpjrun.bat -np 2 -dev hybdev -src HelloWorld

Using -src switch will zip all the content of current working directory and will send to all machines listed in machines file. Since zipping of files is done and then that zipped file is sent to all machines through TCP so this feature should only be used for smaller projects.

5. Once job is finished you can stop MPJ daemons running at machines.

Appendix B: Running MPJ Express without the runtime (manually)

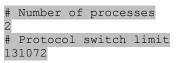
There are two fundamental ways of running MPJ Express applications. The first, and the recommended way is using the MPJ Express runtime infrastructure, alternatively the second way

involves the 'manual' start-up of MPJ Express processes. We do not recommend starting programs manually as normal procedure. This section documents the procedure for manual startup, mainly to allow developers the flexibility to create their own initiation mechanisms for MPJ Express programs. The runmpj.sh script can be considered one example of such a mechanism

- 1. cd mpj-user
- 2. This document is assuming mpj-user as the working directory for users. The name mpjuser itself has no significance.
- 3. Write a configuration file called 'mpj.conf' as follows.
 - a. A typical configuration file that would be used to start two MPJ Express processes is as follows. Note the names 'machine1' and 'machine2' would be replaced by aliases/fully-qualified-names/ IP-addresses of the machines where you want to start MPJ Express processes

```
# Number of processes
2
# Protocol switch limit
131072
# Entry in the form of machinename@port@rank@debug_port
machine1@20000@20001@0@0
machine2@20000@20001@1@0
```

- b. The lines starting with '#' are comments. The first entry which is a number ('2' above) represents total number of processes. The second entry, which is again a number ('131072' above) is the protocol switch limit. At this message size, MPJ Express changes its communication protocol from eager-send to rendezvous. There are a couple of entries, one for each MPJ Express process, and each in the form of machine name (OR)IP@READ_PORT@WRITE_PORT@RANK@DEBUG_PORT. Using this, the users of MPJ Express can control where each MPJ Express process runs, what server port it uses, and what should be the rank of each process. The rank specified here should exactly match the rank argument provided while manually starting MPJ Express processes (using java command). When the users decide to run their code using mpjrun, this file is generated programmatically.
- c. Sample configuration files can be found in <code>%MPJ_HOME%/conf</code> directory. If you wish to start MPJ processes on <code>localhost</code>, see <code>%MPJ_HOME%/conf/local2.conf</code> file.
- d. Each MPJ process uses two ports. Thus, do not use consecutive ports if you are trying to execute multiple MPJ Express processes on same node. A sample file for running two MPJ Express processes on same machine would be



#	Entry	in	the	form	of				
ma	achiner	name	e@rea	ad_poi	ct@	write_	_port@ra	ink@debug	_port
lc	calhos	st@2	20000	002000	010	000			
lc	calhos	st@2	20002	202000	030	100			

- 4. Running your MPJ Express program.
 - a. Running class files

For all the machines listed in mpj.conf, login to each Windows machine, change directory to <code>%MPJ HOME%</code>

java -cp .;%MPJ_HOME%/lib/mpj.jar World <rank> mpj.conf niodev

The <rank> argument should be 0 for process 0 and 1 for process 1. This should match to what has been written in configuration file (mpj.conf). Check the entry format in the configuration file to be sure of the rank

b. Running JAR files

For all the machines listed in mpj.conf, login to each Windows or Linux machine

java -jar hello.jar <rank> mpj.conf niodev

The <rank> argument should be 0 for process 0 and 1 for process 1. This should match to what has been written in configuration file (mpj.conf). Check the entry format in the configuration file to be sure of the rank.

Appendix C: Changing protocol limit switch

MPJ Express uses two communication protocols: the first is 'eager-send', which is used for transferring small messages. The other protocol is rendezvous protocol useful for transferring large messages. The default protocol switch limit is 128 KBytes. This can be changed prior to execution in following ways depending on whether you are running processes manually or using the runtime.

- 1. Running MPJ Express applications manually (without using runtime): The users may edit configuration file (for e.g. %MPJ_HOME%/conf/mpj2.conf) to change protocol switch limit. Look at the comments in this configuration file. The second entry, which should be 131072 if you have not changed it, represents protocol switch limit
- 2. Running MPJ Express applications with the runtime: Use -psl <val> switch to change the protocol switch limit

Appendix D: MPJ Express Testsuite

MPJ Express contains a comprehensive test suite to test the functionality of almost every MPI function. This test suite consists mainly of mpiJava test cases, MPJ JGF benchmarks, and MPJ microbenchmarks. The mpiJava test cases were originally developed by IBM and later translated to Java. As this software follows the API of mpiJava, these test cases can be used with a little modification. MPJ JGF benchmarks are developed and maintained by <u>EPCC at the University of Edingburgh</u>. MPJ Express is redistributing these benchmarks as part of its test suite. The original copyrights and license remain intact as can be seen in source-files of these benchmarks in <u>\$MPJ_HOME/test/jgf_mpj_benchmarks</u>. Further details about these benchmarks can be seen here. MPJ Express also redistributes micro-benchmarks developed by <u>Guillermo Taboada</u>. Further details about these benchmarks can be obtained <u>here</u>

Compiling source code and Testsuite

- 1. Compiling MPJ Express source code
 - a. Being in %MPJ_HOME% directory, execute ant

 $Produces \; \texttt{mpj.jar}, \; \texttt{daemon.jar}, \; \texttt{and} \; \texttt{starter.jar} \; in \; \texttt{lib} \; directory$

- 2. Compiling MPJ Express test-code
 - a. cd test; ant This produces test.jar in lib directory.

Running Testsuite

The suite is located in <code>%MPJ_HOME%/tests</code> directory. The test cases have been changed from their original versions, in order to automate testing. <code>TestSuite.java</code> is the main class that calls each of the test case present in this directory. The build.xml file present in test directory, compiles all test cases, and places test.jar into the lib directory. By default, JGF MPJ benchmarks and MPJ microbenchmarks are disabled. Edit <code>%MPJ_HOME%/test/TestSuite.java</code> to uncomment these tests and execute them. Note, after changing <code>TestSuite.java</code>, you will have to recompile the testsuite by executing <code>'ant'</code> in test directory.

1. cd mpj-user

With Runtime

- 1. Write a machines file
- mpjrun.bat -np 2 -jar %MPJ_HOME%/lib/test.jar

Without Runtime

- 1. Write a configuration file called 'mpj.conf'. Further details about writing configuration file and its format can be found <u>here</u>
 - a. Start the tests

For all the machines listed in mpj.conf, login to each Windows or Linux machine, type,

java -jar %MPJ HOME%/lib/test.jar <rank> mpj.conf niodev

The <rank> argument should be 0 for process 0 and 1 for process 1. This should match to what has been written in configuration file (mpj.conf). Check the entry format in the configuration file to be sure of the rank.

Appendix E: Useful scripts for MPJ Daemons

Following new scripts have been added in MPJ Express to check status of daemons or clean daemons. Details of each script are outlined below:

mpjboot <machines_file>

This command will boot MPJ Express daemons at compute nodes specified in machines file.

```
-bash-4.1$ mpjboot machines
[compute-0-2] MPJ Deamon started successfully with process id: 10904
[compute-0-5] MPJ Deamon started successfully with process id: 10225
-bash-4.1$
```

mpjhalt <machines_file>

This command will halt MPJ Express daemons at compute nodes specified in machines file.

-bash-4.1\$ mpjhalt machines [compute-0-2] MPJ Deamon stopped [compute-0-5] MPJ Deamon stopped -bash-4.1\$

mpjstatus <machines_file>

This command will display current status of MPJ Express daemons at compute nodes specified in machines file.

```
-bash-4.1$ mpjstatus machines
[compute-0-2] MPJ Deamon is running with process id: 11004
[compute-0-5] MPJ Deamon is running with process id: 20640
-bash-4.1$
```

mpjclean <machines_file>

This command will clean all java process at compute nodes specified in machines file.

-bash-4.1\$ mpjclean machines [compute-0-2] Killed all java processes [compute-0-5] Killed all java processes -bash-4.1\$

mpjinfo <machines_file>

This command will display all java process at compute nodes specified in machines file.

```
-bash-4.1$ mpjinfo machines
[aleem.akhtar @ compute-0-2] 11084 MPJDaemon 10050
[aleem.akhtar @ compute-0-5] 30352 MPJDaemon 10050
-bash-4.1$
```

mpjdaemon <query> <hostnames>

This command takes one of the following queries and will perform respective operation on specified hosts

-boot:	start MPJ Express daemons
-halt:	stop MPJ Express daemons
-status:	display current status of MPJ Express daemons
-clean:	clean all java process
-info:	display all java process

For example, this command will boot daemons at localhost.

```
-bash-4.1$ mpjdaemon -boot localhost
[localhost] MPJ Deamon started successfully with process id: 7883
-bash-4.1$
```

And this command will halt daemons at two hosts

```
-bash-4.1$ mpjdaemon -halt compute-0-2 compute-0-5
[compute-0-2] MPJ Deamon stopped
[compute-0-5] MPJ Deamon stopped
-bash-4.1$
```

Mpjdaemon command can be used to directly perform daemon operations without specifying machines file. Default value for hostname is set as localhost.

mpjdaemon.bat <query>

This command is for Windows Operating System and will perform respective operation on localhost only. Following operations are available with this command.

-boot:	start MPJ Express daemons
-halt:	stop MPJ Express daemons
-status:	display current status of MPJ Express daemons

For example to boot/start daemons, following command will be used

D:\work> mpjdaemon -boot
[localhost] MPJ Deamon started successfully with process id: 5052
Or to halt/stop daemons, following command will be used
D:\work> mpjdaemon -halt
[localhost] MPJ Deamon stopped

Note that mpjdaemon.bat only work for localhost.

Appendix F: Switching to Old Collectives

MPJ Express supports running of parallel Java applications using two types of collective primitives. Old collectives are implemented using linear algorithms and were used in earlier versions (0.42 and previous) of MPJ Express. Improved collectives are implemented using Minimum Spanning Tree (MST) and Bucket (BKT) Algorithms. In current version of MPJ Express, new collectives are used by default. To switch back to old collectives follow these steps:

- 1. Edit $MPJ_HOME/conf/mpjexpress.conf file.$
- 2. Change the value of mpjexpress.mpi.old.collectives from "false" to "true".
- 3. Old collectives will be used in next launch of MPJ Express job.