MPJ Express Profiler

User Guide

15th April, 2014

Document Revision Track

Version	Updates	By
1.0	Initial version document	Rizwan Hanif, Amjad Aziz, Aamir Shafi
1.1	Complete configuration details, Profiles and Traces generation steps	Aleem Akhtar, Aamir Shafi, Mohsan Jameel

Table of Contents

List of Figures

1 Getting Started with MPJ Express

This section outlines how to configure MPJ Express Profiler and TAU with Java support. This section also covers how to generate profiles and traces of MPJ Express programs in multicore and cluster configuration using MPJ Express Profiler. Current version of MPJ Express Profiler does not support profiling of applications using native device.

Complete video guide can be found at http://vimeo.com/88968631

Note: This document only covers basic configuration of TAU with Java support and how to generate profiles and traces of MPJ Express applications. To use advance features of TAU i.e. manual instrumentation, Vampir output and PerfExplorer output, please follow TAU users guide and TAU reference guide. You can get these documentations from TAU official website.

1.1 Pre-requisites

- Java 1.6 (stable) or higher
- MPJ Express (version 0.40 or higher)
- TAU (version 2.23.1b or higher)
- Apache ant 1.6.2 or higher (For those who are interested in 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.

1.2 Configuration of MPJ Express Profiler

1. Download MPJ Express and setup environment to write MPJ Express applications. You can follow section-2 of "Linux User Guide" for setting up environment for MPJ Express.
2. Download TAU and unpack it in user directory.
3. Move to TAU root directory.
4. To configure TAU follow below mentioned steps or you can also follow README.JAVA to configure TAU with Java support.
   i. Configure TAU with this command

```
./configure –jdk=path_to_jdk
```

For example:

```
-bash-3.2$ cd tau-2.23
-bash-3.2$ ./configure -jdk=/export/home2/aleem.akhtar/jdk1.7.0_25
```

ii. After successful configuration message use “`make  clean`” command to clean previous configurations (if any)

```
Configuration complete!
   Please add  /export/home2/aleem.akhtar/tau-2.23/x86_64/bin  to your path
   Type "make install" to begin compilation
-bash-3.2$ make clean
```

iii. After successful clean of previous configurations use “`make install`” command to install new configuration

```
-bash-3.2$ make install
```

TAU configuration will create a new directory depending on architecture of your operating system in TAU root directory. For Linux 64 Bit OS, it will create x86_64 directory.

5. After successful configuration of TAU you can set following Environment Variables in `.bashrc` or `.bash_profile`

```
export MPJ_HOME=/export/home2/aleem.akhtar/mpj-v0_40
export TAU_HOME=/export/home2/aleem.akhtar/tau-2.23/x86_64
export LD_LIBRARY_PATH=$TAU_HOME/lib:$LD_LIBRARY_PATH
export CLASSPATH=.:$MPJ_HOME/lib/mpj.jar:$TAU_HOME/lib:$CLASSPATH
export PATH=$MPJ_HOME/bin:$JAVA_HOME/bin:$TAU_HOME/bin:$PATH
```

6. Once Environment Variables are set, test whether TAU has been configured correctly or not

i. Move to TAU root directory.

ii. Now move to examples/java/pi

iii. Use `tau_java` command to run Pi class. If it prints value of Pi then TAU has been configured properly otherwise check Environment Variables.

```
-bash-3.2$ cd tau-2.23
-bash-3.2$ cd examples/java/pi/
-bash-3.2$ tau_java Pi
The value of Increment is: 0.009999999776482582
The value of pi is: 3.1515759417729816
-bash-3.2$
```

At this point you have configured TAU and MPJ Express. Proceed with profiling and tracing of MPJ Express applications

1.3 Profiling of MPJ Express

User can generate profiles for MPJ Express applications in multicore mode and cluster mode.

1. Create a new directory for MPJ Express applications
2. Create a new MPJ Express application and write "HelloWorld" code in it.

```
import mpi.*;

public class HelloWorld {

      public static void main(String args[]) throws Exception {
            MPI.Init(args);
            int me = MPI.COMM_WORLD.Rank();
            System.out.println("Hi from <"+me+">");
            MPI.Finalize();
      }
}
```

3. Save and compile HelloWorld.java

```
-bash-3.2$ mkdir mpjtesting
-bash-3.2$ cd mpjtesting
-bash-3.2$ vi HelloWorld.java
-bash-3.2$ javac -cp .:$MPJ_HOME/lib/mpj.jar HelloWorld.java
-bash-3.2$
```

1.3.1 Profiling in Multicore Mode

Process of using profiler for any device is exactly same. Providing `"-profile"` switch in MPJ Express run command generates profiles for particular device.

For example following command is used for starting a job in multicore mode:

```
mpjrun.sh -np 4 -dev multicore HelloWorld
```

Similarly, for starting a job in multicore device with MPJ Express Profiler following command is used

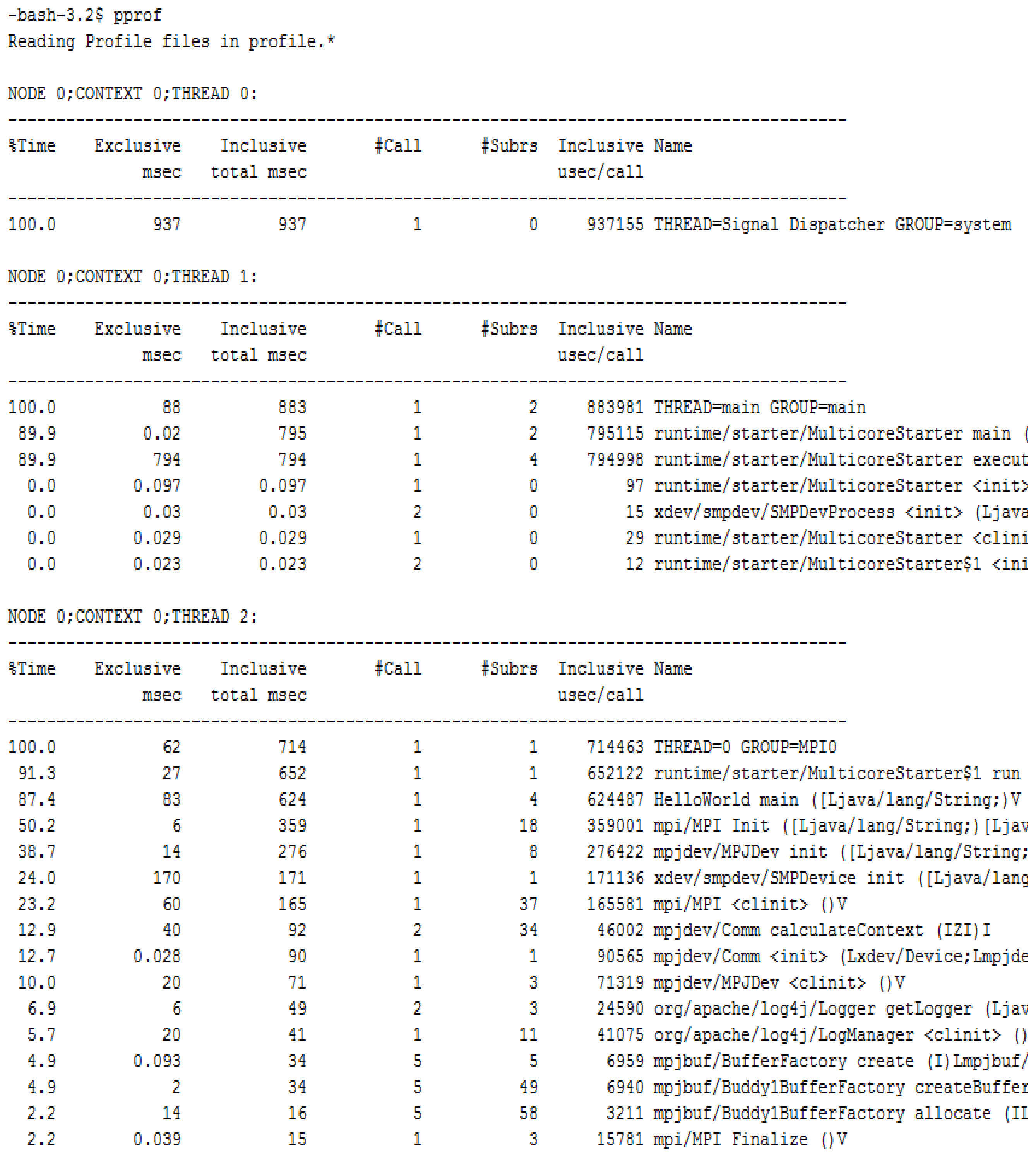
```
mpjrun.sh -np 4 -dev multicore -profile HelloWorld
```

This command will execute your application and will generate profiles for each process in the current directory.

You can use `pprof` to visualize profiles in command line format. Or you can use `paraprof` to visualize profiles in graphical format.

- **Using pprof**

  To visualize profiles in command line format type `pprof`



```
-bash-3.2$ pprof
Reading Profile files in profile.*

NODE 0;CONTEXT 0;THREAD 0:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0          937          937           1           0     937155 THREAD=Signal Dispatcher GROUP=system

NODE 0;CONTEXT 0;THREAD 1:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0           88          883           1           2     883981 THREAD=main GROUP=main
 89.9         0.02          795           1           2     795115 runtime/starter/MulticoreStarter main (
 89.9          794          794           1           4     794998 runtime/starter/MulticoreStarter execut
  0.0        0.097        0.097           1           0         97 runtime/starter/MulticoreStarter <init>
  0.0         0.03         0.03           2           0         15 xdev/smpdev/SMPDevProcess <init> (Ljava
  0.0        0.029        0.029           1           0         29 runtime/starter/MulticoreStarter <clini
  0.0        0.023        0.023           2           0         12 runtime/starter/MulticoreStarter$1 <ini

NODE 0;CONTEXT 0;THREAD 2:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs  Inclusive Name
              msec   total msec                          usec/call
---------------------------------------------------------------------------------------
100.0           62          714           1           1     714463 THREAD=0 GROUP=MPI0
 91.3           27          652           1           1     652122 runtime/starter/MulticoreStarter$1 run
 87.4           83          624           1           4     624487 HelloWorld main ([Ljava/lang/String;)V
 50.2            6          359           1          18     359001 mpi/MPI Init ([Ljava/lang/String;)[Ljav
 38.7           14          276           1           8     276422 mpjdev/MPJDev init ([Ljava/lang/String;
 24.0          170          171           1           1     171136 xdev/smpdev/SMPDevice init ([Ljava/lang
 23.2           60          165           1          37     165581 mpi/MPI <clinit> ()V
 12.9           40           92           2          34      46002 mpjdev/Comm calculateContext (IZI)I
 12.7        0.028           90           1           1      90565 mpjdev/Comm <init> (Lxdev/Device;Lmpjde
 10.0           20           71           1           3      71319 mpjdev/MPJDev <clinit> ()V
  6.9            6           49           2           3      24590 org/apache/log4j/Logger getLogger (Ljav
  5.7           20           41           1          11      41075 org/apache/log4j/LogManager <clinit> ()
  4.9        0.093           34           5           5       6959 mpjbuf/BufferFactory create (I)Lmpjbuf/
  4.9            2           34           5          49       6940 mpjbuf/Buddy1BufferFactory createBuffer
  2.2           14           16           5          58       3211 mpjbuf/Buddy1BufferFactory allocate (II
  2.2        0.039           15           1           3      15781 mpi/MPI Finalize ()V
```

Figure 1-1: Example of command line format

- **Using paraprof**

Type `paraprof` to visualize profiles in graphical format

```
-bash-3.2$ paraprof
```

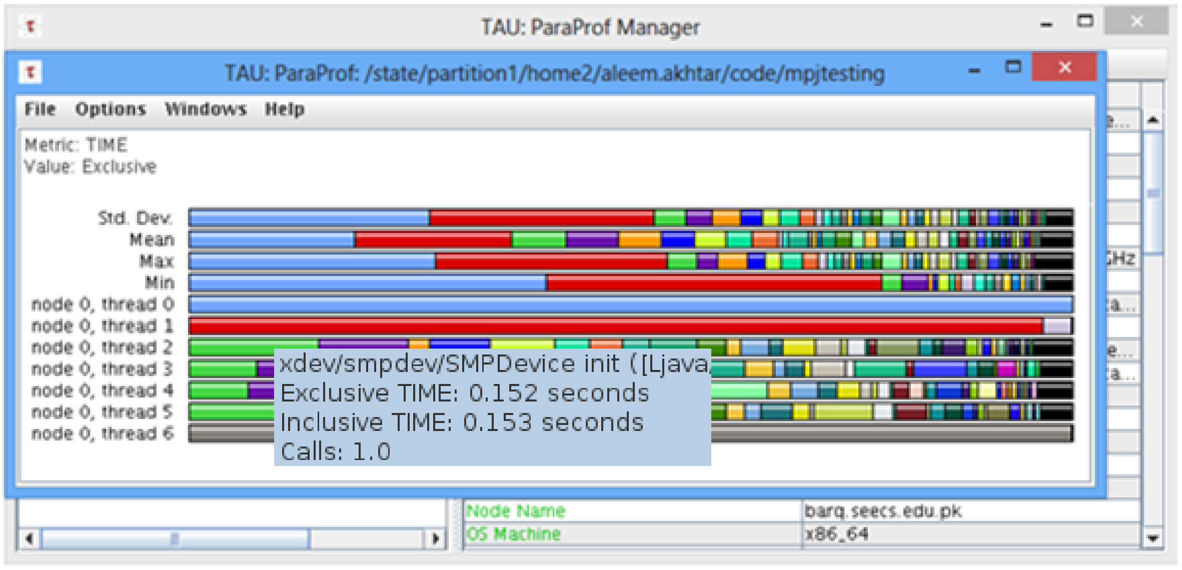


Figure 1-2: Example of graphical format in multicore mode

1.3.2 Profiling in Cluster Mode

This section outlines steps to generate profiles for MPJ Express applications in the cluster configuration with niodev communication device driver.

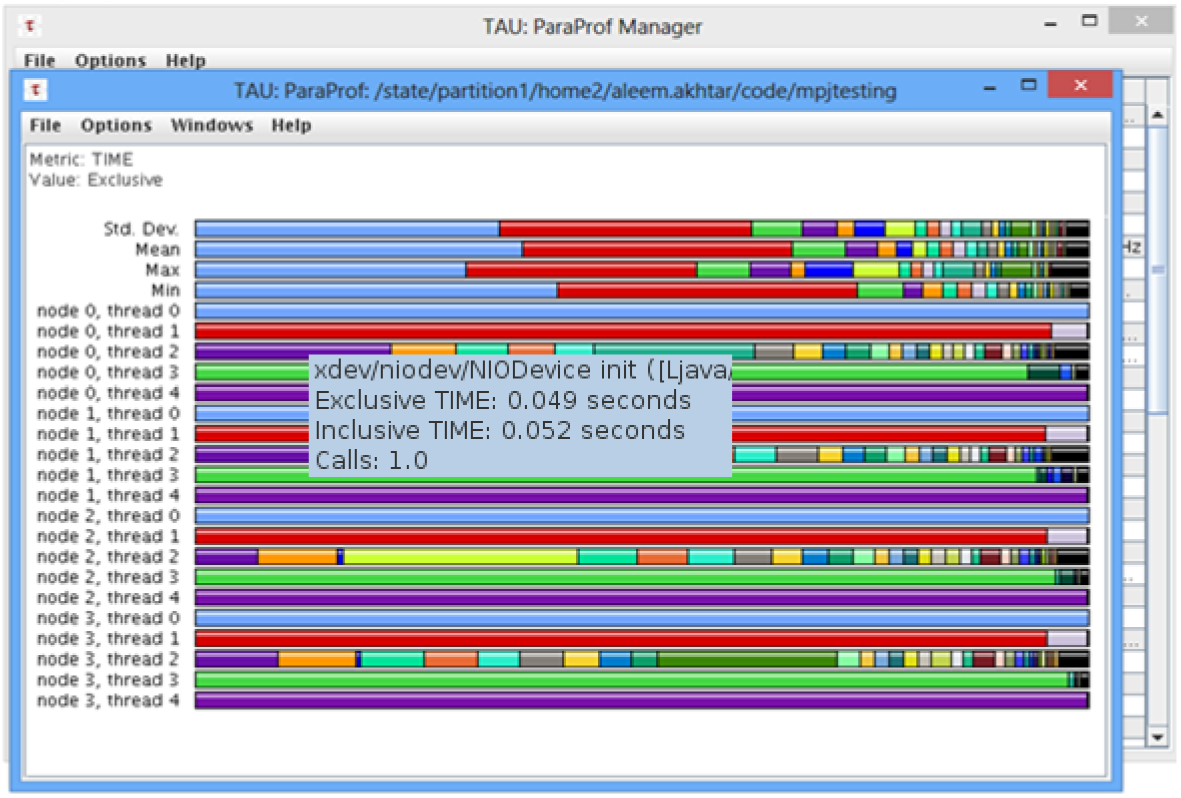
1. Create a machines file and write IP address or machines names in it and start daemons. You can follow section-2.3 of "Linux User Guide" on how to create a machines file and start daemons.
2. Once daemons get started, use this command to start application in cluster mode.

```
mpjrun.sh -np 4 -dev niodev -profile HelloWorld
```

This will generate profiles for all processes in the current working directory.

Once profiles are generated you can then use `pprof` or `paraprof` to visualize profiles.

```
-bash-3.2$ paraprof
```


TAU: ParaProf Manager
File Options Help
TAU: ParaProf: /state/partition1/home2/aleem.akhtar/code/mpjtesting
File Options Windows Help

Metric: TIME
Value: Exclusive

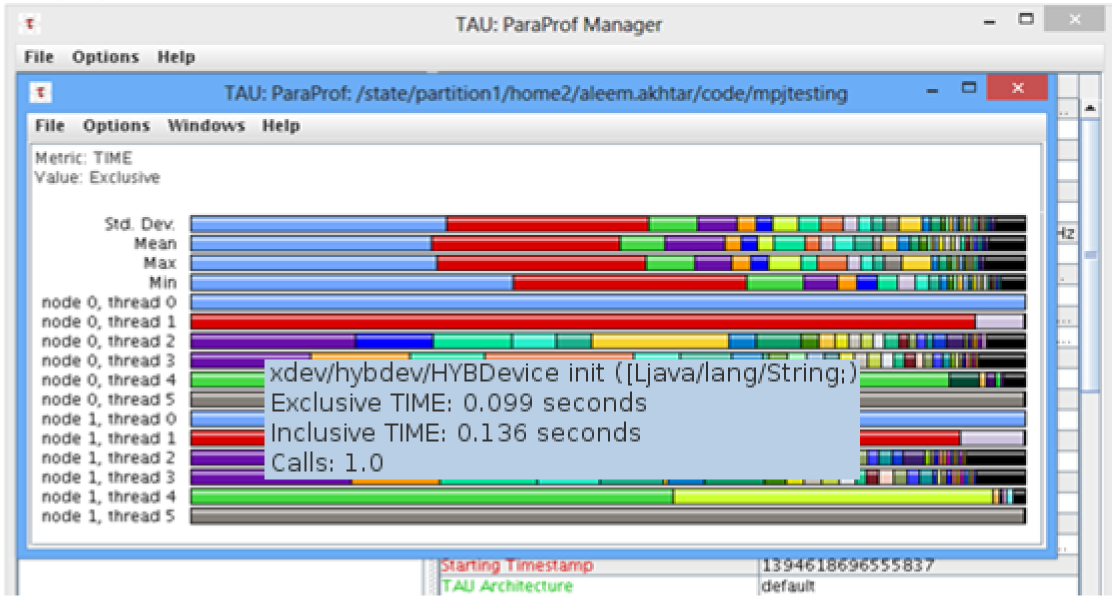
Std. Dev.
Mean
Max
Min
node 0, thread 0
node 0, thread 1
node 0, thread 2
node 0, thread 3
node 0, thread 4
node 1, thread 0
node 1, thread 1
node 1, thread 2
node 1, thread 3
node 1, thread 4
node 2, thread 0
node 2, thread 1
node 2, thread 2
node 2, thread 3
node 2, thread 4
node 3, thread 0
node 3, thread 1
node 3, thread 2
node 3, thread 3
node 3, thread 4

xdev/niodev/NIODevice init ([Ljava/
Exclusive TIME: 0.049 seconds
Inclusive TIME: 0.052 seconds
Calls: 1.0

Figure 1-3: Example of graphical format in niodev mode

Similarly using `mpjrun.sh -np 4 -dev hybdev -profile HelloWorld` will generate profiles for hybdev mode in current working directory.

```
-bash-3.2$ paraprof
```


TAU: ParaProf Manager
File Options Help
TAU: ParaProf: /state/partition1/home2/aleem.akhtar/code/mpjtesting
File Options Windows Help

Metric: TIME
Value: Exclusive

Std. Dev.
Mean
Max
Min
node 0, thread 0
node 0, thread 1
node 0, thread 2
node 0, thread 3
node 0, thread 4
node 0, thread 5
node 1, thread 0
node 1, thread 1
node 1, thread 2
node 1, thread 3
node 1, thread 4
node 1, thread 5

xdev/hybdev/HYBDevice init ([Ljava/lang/String;)
Exclusive TIME: 0.099 seconds
Inclusive TIME: 0.136 seconds
Calls: 1.0

Starting Timestamp 1394618696555837
TAU Architecture default

Figure 1-4: Example of graphical format in hybdev mode

1.4 Tracing of MPJ Express

Procedure to generate trace files for MPJ Express applications is the same as for profiling.

1. Enable Tracing in `.bashrc` or `.bash_profile` and disable Profiling

```
export TAU_TRACE=0
export TAU_PROFILE=1
```

2. Now you can generate trace files using same steps as explained above for profiling.

```
Multicore Mode: mpjrun.sh -np 4 -dev multicore -profile HelloWorld
```

```
Niodev Mode: mpjrun.sh -np 4 -dev niodev -profile HelloWorld
```

```
Hybdev Mode: mpjrun.sh -np 4 -dev hybdev -profile HelloWorld
```

3. Using any of above mentioned command will generate trace files in the current working directory. You can use `Jumpshot` to visualize trace files.

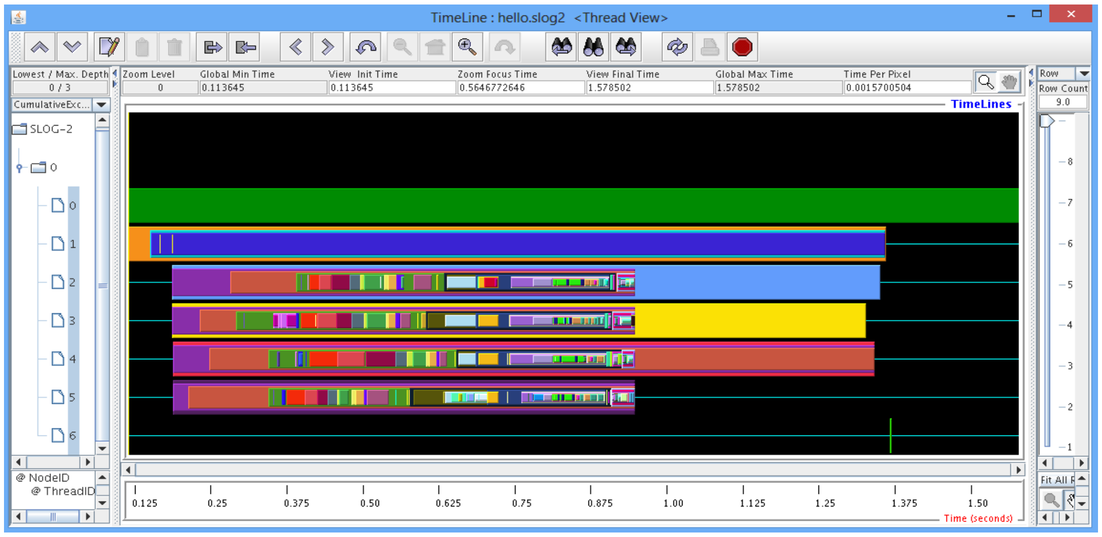
- **Using Jumpshot**

This section outlines how to visualize trace files using Jumpshot

1. Merge all trace files using `tau_treemerge.pl` command
2. `tau_treemerge.pl` command will merge all trace files into one `tau.trc` file and one `tau.edf` file in the same directory.
3. Now use `tau2slog2` command to create `slog2` file which is supported by Jumpshot

```
tau2slog2 tau.trc tau.edf -o hello.slog2
```

4. Type `jumpshot` on command line. Jumpshot window will open
5. Click File → Select → hello.slog2 file to visualize traces along timeline


TimeLine : hello.slog2 <Thread View>

Lowest / Max. Depth: 0 / 3
Zoom Level: 0
Global Min Time: 0.113645
View Init Time: 0.113645
Zoom Focus Time: 0.5646772646
View Final Time: 1.578502
Global Max Time: 1.578502
Time Per Pixel: 0.0015700504
Row Count: 9.0

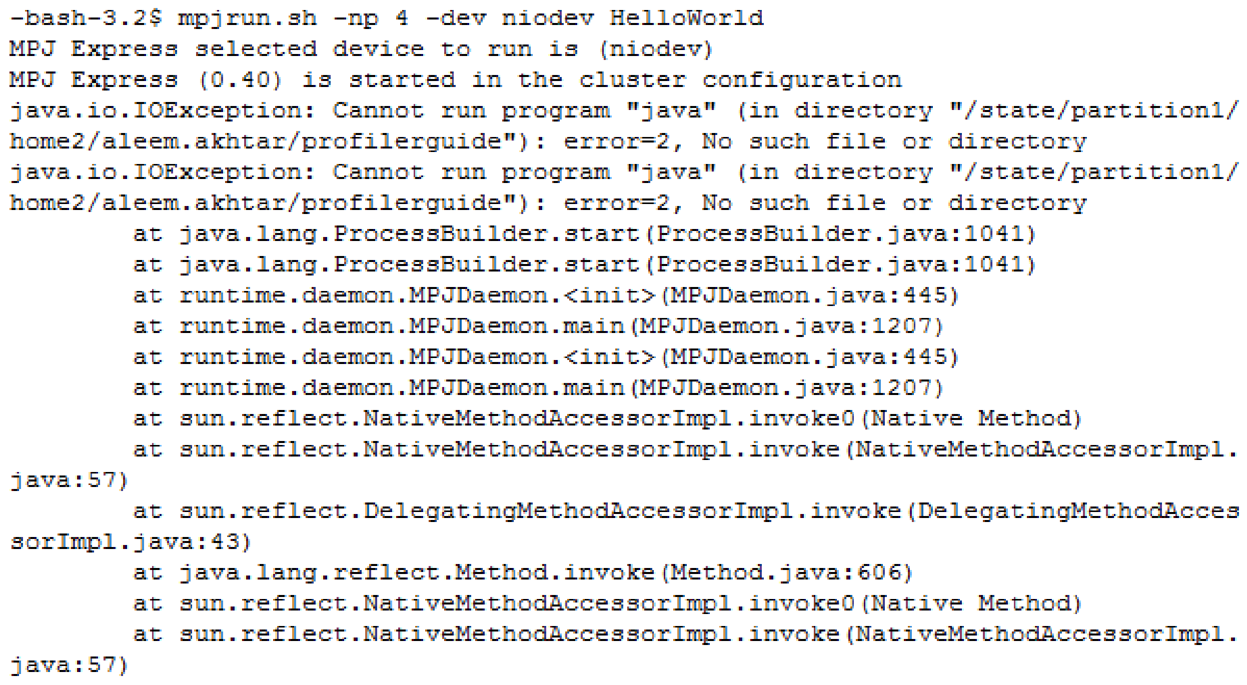
Figure 1-5: Example of Jumpshot window

2 Known Issues and Limitations

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

1. Users of the Windows Vista and 7 might find install mpjd-windows.bat script not working directly. In such case, try executing with “Run as Administrator”. If the issue still persists, try executing the script mpjdaemon.bat directly or turning off the firewall.

2. If you are running your applications on Rocks cluster then you might get below mentioned error. To solve this just add “-wdir =absolute path to your current working directory” switch in your mpjrun command.



```
-bash-3.2$ mpjrun.sh -np 4 -dev niodev HelloWorld
MPJ Express selected device to run is (niodev)
MPJ Express (0.40) is started in the cluster configuration
java.io.IOException: Cannot run program "java" (in directory "/state/partition1/
home2/aleem.akhtar/profilerguide"): error=2, No such file or directory
java.io.IOException: Cannot run program "java" (in directory "/state/partition1/
home2/aleem.akhtar/profilerguide"): error=2, No such file or directory
        at java.lang.ProcessBuilder.start(ProcessBuilder.java:1041)
        at java.lang.ProcessBuilder.start(ProcessBuilder.java:1041)
        at runtime.daemon.MPJDaemon.<init>(MPJDaemon.java:445)
        at runtime.daemon.MPJDaemon.main(MPJDaemon.java:1207)
        at runtime.daemon.MPJDaemon.<init>(MPJDaemon.java:445)
        at runtime.daemon.MPJDaemon.main(MPJDaemon.java:1207)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.
java:57)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcces
sorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:606)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.
java:57)
```

Figure 2-1: State Partition Error

```
mpjrun.sh -np 4 -dev niodev –wdir (absolute path to current working
directory) HelloWorld
```

For example:

```
-bash-3.2$ mpjrun.sh -np 4 -dev niodev -wdir /export/home2/aleem.akhtar/mpjdir/ HelloWorld
```

3 Contact and Support

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

1 Aamir Shafi (aamir.shafi@seecs.edu.pk)

2 Mohsan Jameel (mohsan.jameel@seecs.edu.pk)

3 Bryan Carpenter (bryan.carpenter@port.ac.uk)

4 Mark Baker (http://acet.rdg.ac.uk/~mab)

5 Guillermo Lopez Taboada (http://www.des.udc.es/~gltaboada)