

Appendix

Chart A. The following table is a complete file listing of the source code that makes up the JPM application.

File Name	Package	Description
Migrate.java	edu.fsu.jpm.jpmp	JPM process framework. The Migrate class is extended by other Java classes that want to "plug" into the JPM application for logging and migration.
jpmd.java	edu.fsu.jpm.jpmd	JPM Daemon. The jpmd class is the main program or daemon the is executed. This class is responsible for starting all of the other sub-component threads.
jpmd-pps.java	edu.fsu.jpm.jpmd	JPM Daemon Sub-Component. The jpmd-pps class represents the Pre-Process Scheduler sub-component. The jpmd-pps is responsible for managing the process registration request, suspending, and serializing the processes.
jpmd-ps.java	edu.fsu.jpm.jpmd	JPM Daemon Sub-Component. The jpmd-ps class represents the Process Logger sub-component. The jpmd-ps is responsible for logging the processes together for migration to another machine.
jpmd-pe.java	edu.fsu.jpm.jpmd	JPM Daemon Sub-Component. The jpmd-pe class represents the Process Execution sub-component. The jpmd-pe is responsible for reconstituting the serialized processes and executing the processes on the migrated machine.
jpmd-ijpm.java	edu.fsu.jpm.jpmd	JPM Daemon Sub-Component. The jpmd-ijpm class represents the Inter-JPM communicator sub-component. The jpmd-ijpm is responsible for managing communication with the other machines in the cluster of known machines. Also, this class is responsible for migrating the process blocks of processes to the other machines.
jpmd-ijpm-nodemgr.java	edu.fsu.jpm.jpmd	JPM Daemon Sub-Component. The jpmd-ijpm-nodemgr class is a sub-component to the jpmd-ijpm sub-component. The jpmd-ijpm-nodemgr is responsible for managing the connection with the other machines with a jpmd running.
jpmd-pm.java	edu.fsu.jpm.jpmd	JPM Daemon Sub-Component. The jpmd-pm class represents the Process Monitor sub-component. The jpmd-pm is responsible for the JPMD GUI and all user interaction with the JPMD.
JpmCommand.java	edu.fsu.jpm.jpmd	The JpmCommand class is used to implement the communication protocol between the JPMD and the JPM process. This class has the different message definitions that can be sent between the jpmd-pps and the Migrate framework.

JpmIjpmIf.java	edu.fsu.jpj.jpmd	The JpmIjpmIf class is the RMI interface class that defines the remote interface used between the jpmd processes running on different machines.
JpmIjpmImpl.java	edu.fsu.jpj.jpmd	The JpmIjpmImpl class is the RMI method implementations of the methods defined in the JpmIjpmIf RMI interface class.
JpmNode.java	edu.fsu.jpj.jpmd	The JpmNode class holds information about a node or rather machine in the cluster of know machines. Most importantly, this class holds a reference to the RMI interface for that machine. Also, this class has a status indicator used to indicate if the node is active or inactive.
JpmProcess.java	edu.fsu.jpj.jpmd	The JpmProcess class holds information about a process. Most importantly, this class holds a reference to the serialized JPM Process object.
JpmProcessBlock.java	edu.fsu.jpj.jpmd	The JpmProcessBlock class is a extension of the Java LinkedList and represents a block of logged processes to be migrated to another machine.
JpmProcessLogger.java	edu.fsu.jpj.jpmd	The JpmProcessLogger class is responsible for the management of the process blocks.
JpmSharedMemory.java	edu.fsu.jpj.jpmd	The JpmSharedMemory class represents the shared memory in the jpmd. All the jpmd sub-components have a reference to the shared memory. Most importantly, this class has the pre-process, process-blocks, and migrated process-blocks collections.
jpj_constants.java	edu.fsu.jpj	The jpj_constants class contains the constants used throughout the jpmd and Migrate classes.

Chart B. The following table is a table of commands defined in JpmCommand, all of which Migrate needs to be able to handle.

Command	Description
JPM_NONE	This command indicates no command. This is used when the JPM process is first registering with the JPMD.
JPM_STATUS	This command from the JPMD to the JPM process requests that the process send a status report back to the JPMD.
JPM_START	This command from the JPMD to the JPM process requests that the process start executing. This is used in two instances. One, if the process is rejected by the JPMD, the JPMD will issue this command to have the process execute locally. Second, once the process has been migrated and reconstituted, the JPMD will issue this command to the process to have the process execute on the migrated machine.
JPM_SUSPEND	This command from the JPMD to the JPM process commands

	the process to prepare itself for migration and suspends the process.
JPM_SERIALIZE	This command from the JPMD to the JPM process requests that the process be serialized. In return, the serialized object bytes are sent back to the JPMD.
JPM_RESUME	This command from the JPMD to the JPM process commands the process to resume itself after being suspended.
JPM_COMPLETE	This command from the JPMD to the JPM process commands the process to complete itself and exit gracefully. This is used once the process has been returned from being migrated and needs to display output, etc.

Chart C. The following table shows the sequence of events during the registration of a JPM process with the JPMD.

JPM process	JPM daemon
	JPM daemon Starts – The daemon starts, initializes, opens the port for communication, and begins to listen for JPM process registration requests.
JPM process Starts – The process registers itself with the daemon. The framework method register() attaches to the well known port and sends a registration request to the JPM daemon.	
	JPM daemon receives JPM process registration request – Examines process by examining the information sent in the registration request. Determines that the process is “migratable” and issues the suspend command to the process.
JPM process receives the signal to suspend itself and calls the suspend framework method. The process prepares itself for migration. Once suspension activity is complete, the serialized process is sent to the JPM daemon via socket communication. The JPM process then gracefully exits and stops execution.	
	The Java-PM daemon receives the serialized JPM process and queues the process for migration.

JPM daemon Graphical User Interface (GUI) Process Monitor (PM) Guide

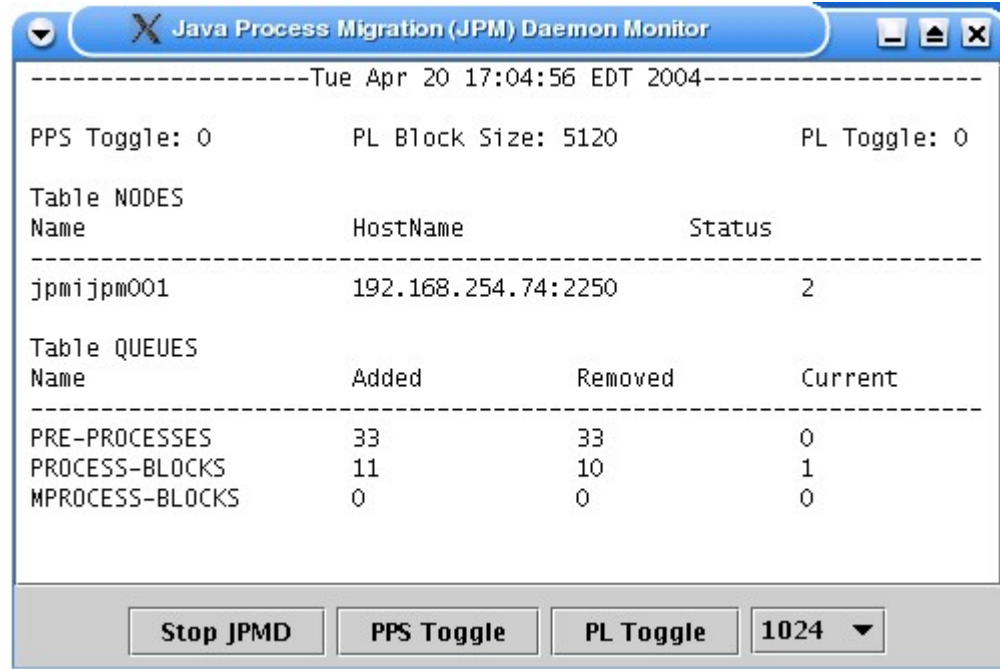


Figure 1

Figure 1 shows a screen shot of the JPM daemon PM during execution. At the top of the display window is the current date and time. The next line in the display window shows the current state of the Pre-Process Scheduler (PPS) toggle, the Process Logger (PL) maximum block size, and the current state of the PL toggle. The first table in the display window, NODES, lists all of the nodes that this JPM daemon knows of and the current status of that node. The second table, QUEUES, shows the current state of each of the queues contained in shared memory. Specifically how many elements have been added, removed, and are currently contained in that particular queue.

The buttons on the bottom of the dialog window allow the user to interact with the JPM daemon. The “Stop JPMD” button will stop the execution of the daemon including the PM, which will terminate the monitor dialog window. The “PPS Toggle” button will toggle the PPS execution. A toggle state of ‘0’ means that the PPS is on and a toggle state of ‘1’ means that the PPS is off. The “PL Toggle” button will toggle the PL execution. Again, ‘0’ is on and ‘1’ is off. The drop down widget displaying the 1024 number allows the user to select the size of the PL maximum block size in bytes.

JPM Experiment Graphical User Interface (GUI) Guide

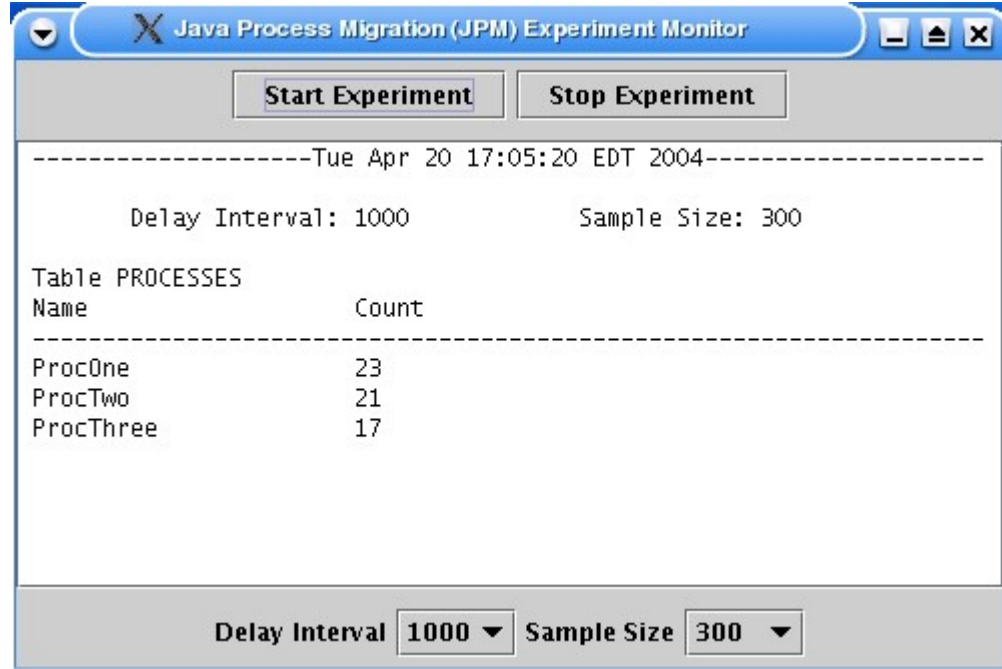


Figure 2

Figure 2 shows a screen shot of the JPM Experiment application during execution. The buttons at the top of the dialog window allow the user to start and stop an experiment as the button labels suggest.

The display window contains the date and time on the first line. The next line shows the current setting of the experiment delay interval and sample size. The delay interval is the time in milliseconds between the creation and execution of each process in the experiment. The sample size indicates how many processes will be created and executed for the experiment.

The table PROCESSES shows the count of each of the three JPM processes that have been created and executed.