Applied Physics Computing using Java Parallel Processing Framework in clusters environment of parallel computing

Deepak Agnihotri *
Department of Computer Applications,
National Institute of Technology Raipur,
G.E. Road Raipur (C.G.) India-492010

Mithilesh Atulkar
Department of Computer Applications,
National Institute of Technology Raipur,
G.E. Road Raipur (C.G.) India-492010

Harendra Bikrol
Department of Computer Applications,
National Institute of Technology Raipur,
G.E. Road Raipur (C.G.) India-492010

Abstract

This paper gives a brief introduction of research in the field of parallel and grid computing using java parallel processing framework (JPPF). It is observed that applied physics computations like Moment of Inertia, velocity, viscosity of liquid, elasticity etc. requires lot of calculations and also it should be precise to high degree of accuracy. This application calculates the Moment of Inertia of 11 types of bodies symmetrical in nature along with given axis of rotation. These 11 tasks are running parallel in JPPF grid in clusters environment of parallel computing. This application promotes the solution of such engineering problems with great accuracy and speed as platform independent parallel applications. The experimental results are obtained with the help of JPPF GUI monitoring and administration tool.

Keywords

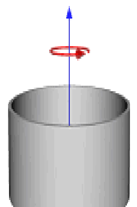
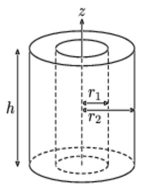
JPPF, node, drivers, MOI, axis of rotation, load balancing algorithms.

Introduction

In this application there are 11 tasks for Moment of inertia calculation of 11 symmetrical bodies are running parallel in JPPF grid. JPPF is an open source Grid Computing platform written in Java [8] that makes it easy to run applications in parallel, and speed up their execution by orders of magnitude. Write once, deploy once, and execute everywhere! [1].The experimental results are obtained with the help of JPPF GUI administration, monitoring and management control tool .This tool helps to get average execution time, average node execution time, average queue size, average transport time etc. Before running this application, we must have a JPPF driver and at least one node running. This paper is organized in five sections: First section, describes the application implemented parallel in JPPF grid for Moment of Inertia Calculations of 11 symmetrical bodies. Second section, describes JPPF and its various features, required software's for running parallel application using JPPF framework, and other technologies in java for parallel and grid computing. Third section, describes other related works in the field; in the Fourth section, the authors have given the results of their experiments and discussion about the results, and at last section authors have given the conclusion of their paper.

1. Moment of Inertia of different bodies symmetrical in nature

The following table 1-1 depicts the different types of bodies', formulas for Moment of Inertia calculations with respect to given axis of rotations [6, 7].

S.No.	Description	figure
01	Thin cylindrical shell with open ends,of radius r and mass m,Moment(s) of Inertia $I=mr^2$ **Comments:** This expression assumes the shell thickness is negligible. It is a special case of the next object for $r1=r2$. Also, a point mass (m) at the end of a rod of length r has this same moment of inertia and the value r is called the radius of gyration.	
02	Thick-walled cylindrical tube with open ends, of inner radius $r1$, outer radius $r2$, length h and mass m $I_z=½m(r_1{}^2+r_2{}^2)$ $I_x=I_y=1⁄12\ m[3(r_2{}^2+r_1{}^2)+h^2]$ or when defining the normalized thickness $tn = t/r$ and letting $r = r2$, then $I_z = mr^2\left(1-t_n+\frac{1}{2}t_n{}^2\right)$ **Comments:** With a density of ρ and the same geometry $I_z = \frac{1}{2}\pi\rho h\left(r_2{}^4 - r_1{}^4\right)$	

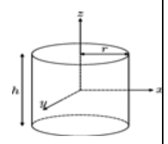
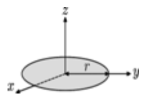
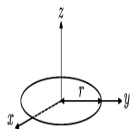
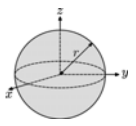
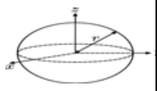
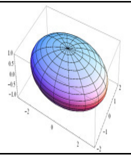
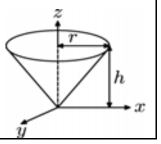
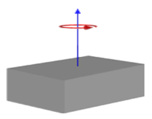
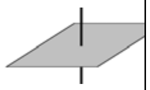
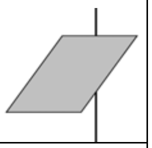
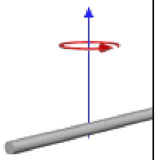
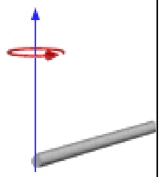
03	Solid cylinder of radius *r*, height *h* and mass *m* $I_z = \frac{mr^2}{2}$ $I_x = I_y = \frac{1}{12}m\left(3r^2 + h^2\right)$ **Comments:** This is a special case of the previous object for *r*1=0. (Note: X-Y axis should be swapped for a standard right handed frame)	
04	Thin, solid disk of radius *r* and mass *m* $I_z = \frac{mr^2}{2}$ $I_x = I_y = \frac{mr^2}{4}$ **Comments:** This is a special case of the previous object for *h*=0.	
05	Thin circular hoop of radius *r* and mass *m* $I_z = mr^2$ $I_x = I_y = \frac{mr^2}{2}$ This is a special case of a torus for *b*=0. (See below.), as well as of a thick-walled cylindrical tube with open ends, with *r*1=*r*2 and *h*=0.	
06	Ball (solid) of radius *r* and mass *m* $I = \frac{2mr^2}{5}$ **Comments:** A sphere can be taken to be made up of a stack of infinitesimal thin, solid discs, where the radius differs from 0 to *r*.	
07	Sphere (hollow) of radius *r* and mass *m* $I = \frac{2mr^2}{3}$ **Comments:** Similar to the solid sphere, only this time considering a stack of infinitesimal thin, circular hoops.	
08	Ellipsoid (solid) of semiaxes *a*, *b*, and *c* with axis of rotation *a* and mass *m* $I_a = \frac{m(b^2 + c^2)}{5}$	
09	Right circular cone with radius *r*, height *h* and mass *m* $I_z = \frac{3}{10}mr^2$ $I_x = I_y = \frac{3}{5}m\left(\frac{r^2}{4} + h^2\right)$	
10	Solid cuboid of height *h*, width *w*, and depth *d*, and mass *m* $I_h = \frac{1}{12}m\left(w^2 + d^2\right)$ $I_w = \frac{1}{12}m\left(h^2 + d^2\right)$ $I_d = \frac{1}{12}m\left(h^2 + w^2\right)$ **Comments:** For a similarly oriented cube with sides of length s, $I_{CM} = \frac{ms^2}{6}$	
11	Thin rectangular plate of height *h* and of width *w* and mass *m* $I_c = \frac{m(h^2 + w^2)}{12}$	
12	Thin rectangular plate of height *h* and of width *w* and mass *m* (Axis of rotation at the end of the plate) $I_e = \frac{mh^2}{3} + \frac{mw^2}{12}$	
13	Rod of length *L* and mass *m* $I_{center} = \frac{mL^2}{12}$ Comments: This expression assumes that the rod is an infinitely thin (but rigid) wire. This is a special case of the previous object for *w* = *L* and *h* = *0*.	
14	Rod of length *L* and mass *m* (Axis of rotation at the end of the rod) $I_{end} = \frac{mL^2}{3}$ Comments: This expression assumes that the rod is an infinitely thin (but rigid) wire. This is also a special case of the thin rectangular plate with axis of rotation at the end of the plate: *h* = *L* and *w* =*0*.	

Table 1-1

Now the following table 1-2 shows 11 tasks, implemented parallel in JPPF grid for Moment of Inertia calculations.

Task 1::Parallel Calculation of Moment of Inertia of a Thin cylindrical shell with open ends, of radius=14.38 and mass=7.74is I=1600.5112560000002

Task 2::[1]Parallel Calculation of The Moment of inertia of a Thick-walled cylindrical tube with open ends,of inner radius=14.38,outer radius=12.38,length=12.21 and mass=7.74,with respect to x-axis and y-axis is Ix=Iy=792.8536725
[2] Parallel Calculation of The Moment of inertia of a Thick-walled cylindrical tube with open ends, of inner radius=14.38,outer radius=12.38 and mass=7.74,with respect to z-axis is Iz=1393.3888560000003***

Task 3::[1]Parallel Calculation of The Moment of inertia of a Solid cylinder of radius=14.38,height=12.21 and mass=7.74,with respect to x-axis and y-axis is Ix=Iy=688.6055475000002
[2]The Moment of inertia of a Solid cylinder of radius=14.38,height=12.21 and mass=7.74,with respect to z axis is Iz= 800.2556280000001 ***

***Task 4::[1]Parallel Calculation of The Moment of inertia of a Thin solid disk of radius=14.38 and mass=7.74
, with respect to x and y axis is Ix=Iy=400.12781400000006 ***
[2]The Moment of inertia of a Thin solid disk of radius=14.38 and mass=7.74, with respect to z axis is Iz= =800.2556280000001 ***

Task 5::[1]Parallel Calculation of The Moment of inertia of a Thin circular hoop with radius=14.38 and mass=7.74,with resect to x and y axis is Ix=Iy= =800.2556280000001
[2]The Moment of inertia of a Thin circular hoop with radius=14.38 ,
and mass=7.74,with resect to z axis is Iz=1600.5112560000002 ***

***Task 6::Parallel Calculation of The Moment of inertia of a Ball (solid) with radius=14.38and mass=7.74 I= =640.2045024000001 ***

***Task 7::Parallel Calculation of] The Moment of inertia of an Ellipsoid (solid) of semi axis a=4.23,b=4.38,
and c=5.23,with axis of rotation a=4.23 and mass=7.74,I=72.0397404***

***Task 8::[1] Parallel Calculation of The Moment of inertia of a Right circular cone with radius=4.38,height=5.23 and mass=7.74with resect to x and y axis is Ix=Iy=149.299956 ***
[2] The Moment of inertia of a Right circular cone with radius=4.38,height=5.23 and mass=7.74, with resect to z axis is Iz=44.5461768***

***Task 9::[1]Parallel Calculation of the Moment of inertia of a Solid cuboid of height=5.23,width=12.21, and depth=8.45, and mass=7.74 with resect to Height Ih=142.21385700000002 ***
[2] Parallel Calculation of the Moment of inertia of a Solid cuboid of height=5.23,width=12.21, and depth=8.45, and mass=7.74, the with resect to Width Iw==63.697233 ***
[3] Parallel Calculation of the Moment of inertia of a Solid cuboid of height=5.23,width=12.21, and depth=8.45, and mass=7.74 The Moment of inertia with resect to Depth Id=113.80186500000003***

***Task 10::[1]Parallel Calculation of The Moment of Inertia of a Thin rectangular plate of height h=5.23 and of width w=12.21 and
mass=7.74,when axis of rotation is at the end of the plate Ie=166.72972650000003***
[2]The Moment of Inertia of a Thin rectangular plate of height h= and of width w=12.21and
mass=7.74,when axis of rotation is at the center of the plate Ic=113.80186500000003 ***

***Task 11::[1]Parallel Calculation of The Moment of inertia of a Rod of length=5.23 and mass=7.74,when Axis of rotation is at the center of the rod Ic=17.642620500000003,
[2]The Moment of inertia of a Rod of length=5.23 and mass=7.74,when Axis of rotation is at the end of the rod Ie=70.57048200000001 ***

Table 1-2

2. JPPF Features

JPPF stands for Java Parallel Processing Framework. JPPF has many outstanding features such as :- a JPPF grid can be up and running in minutes; highly scalable, distributed framework for the execution of Java tasks; leverages JCA 1.5 to integrate with leading J2EE application servers; easy programming model that abstracts the complexity of distributed and parallel processing; graphical and programmatic tools for fine-grained monitoring and administration; reliability through redundancy, recovery and failover capabilities; a set of fully documented sample applications, demonstrating the use of JPPF on real-life problems; very flexible and non-constraining open-source licensing [1, 2, 3,4,5,and 9].

2.1 Required Software's

We need to download and install the following JPPF components:
JPPF application template: this is the JPPF-2.0-application-template.zip file
JPPF driver: this is the JPPF-2.0-driver.zip file
JPPF node: this is the JPPF-2.0-node.zip file
JPPF administration console: this is the JPPF-2.0-admin-ui.zip file.

These files are all available from the JPPF installer and/or from the JPPF download page ([1]). In addition to this, Java 1.5 or later and Apache Ant 1.7.0 or later should already be installed on your machine. We have created a new folder called "JPPF", in which all these components are unzipped [1]. Thus, we should have the following folder structure:

```
» JPPF
  » JPPF-2.0-admin-ui
  » JPPF-2.0-application-template
  » JPPF-2.0-driver
  » JPPF-2.0-node
```

The JPPF-2.0-admin-ui folder is used for executing JPPF framework's GUI monitoring and administration tool. We have placed our all java files for this application in the JPPF-2.0-application-template folder. The JPPF-2.0-application-template folder is organized with the following directory structure:

root directory: contains the scripts to build and run the application

src: this is where the sources of the application are located

classes: the location where the Java compiler will place the built sources

config: contains the JPPF and logging configuration files

lib: contains the required libraries to build and run the application

The JPPF-2.0-driver and JPPF-2.0-node folders are required for executing driver and nodes these two folders containing all the necessary files for running successfully the driver and nodes should be installed in each client machine which you want to use for your application. Before running this application, we must have a JPPF server and at least one node running. We have used Apache Ant 1.7.0 for running this application.

2.2 Other Technologies in Java for Parallel and Grid Computing

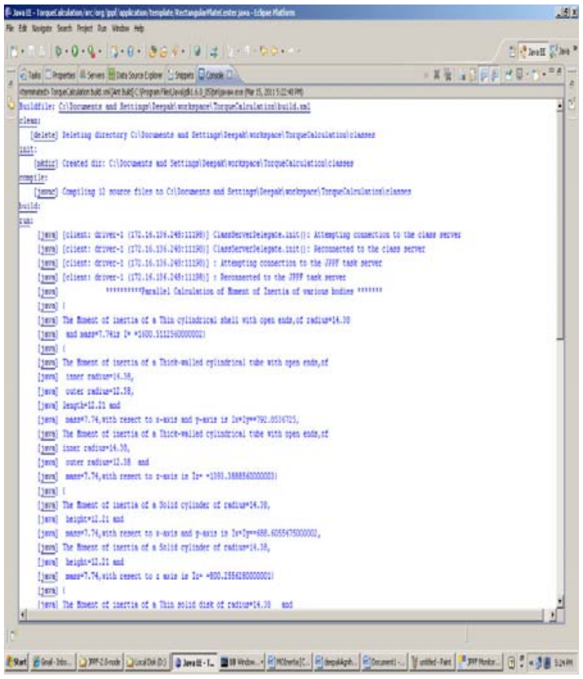
There are various other technologies in Java for Parallel and Grid Computing, some of them are Gridgain, HPJava, JPVM, Java Aglets using Mobile agents etc. JPPF might be the main competitor for the GridGain for grid computing applications using java. There are various features of JPPF frameworks such as TCP port multiplexer, J2ee connector, JPPF GigaSpaces etc which makes it better than other java technologies for Parallel and Grid Computing applications. The one important feature form the JPPF framework is TCP port multiplexer, very useful to connect various nodes in case of proxy servers and firewalls.

3. Other Related Works

There are various other applications which are running with JPPF framework some of them are come with JPPF sample applications like Matrix Multiplication, DNA/Protein Sequence alignment etc., which can be downloaded freely from the JPPF website[1]. Many researchers are using this framework for executing their research applications parallel in JPPF grid. Mostly Scientific and Mathematical applications like Numerical Integration, Solution of Linear algebraic equations, Numerical physics applications etc uses JPPF framework for parallel task execution in JPPF grid, which requires so much calculation with great precision.

4. Results and Discussions

JPPF now has 4 different algorithms auto tuned, manual, and proportional and rl algorithms to compute the distribution of tasks to the nodes, each with its own configuration parameters. The distribution of the tasks to the nodes is performed by the JPPF driver. This work is actually the main factor of the observed performance of the framework. It consists essentially in determining how many tasks will go to each node for execution, out of a set of tasks sent by the client application. Each set of tasks sent to a node is called a "bundle" and the role of the load balancing (or task scheduling) algorithm is to optimize the performance by adjusting the number of task sent to each node [1]. The application is executed for 2, 4, and 8 nodes using auto tuned and proportional load balancing algorithms. The results tell that they depend on number of nodes and load balancing algorithm.



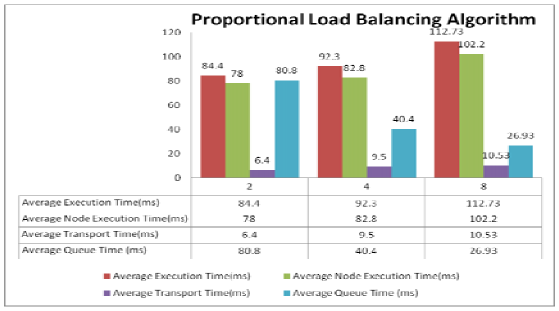
```
Buildfile: C:\Documents and Settings\Deepak\workspace\TorqueCalculation\build.xml
clean:
   [delete] Deleting directory C:\Documents and Settings\Deepak\workspace\TorqueCalculation\classes
init:
    [mkdir] Created dir: C:\Documents and Settings\Deepak\workspace\TorqueCalculation\classes
compile:
    [javac] Compiling 12 source files to C:\Documents and Settings\Deepak\workspace\TorqueCalculation\classes
build:
run:
     [java] [client: driver-1 (172.16.136.248:11198)] ClassServerDelegate.init(): Attempting connection to the class server
     [java] [client: driver-1 (172.16.136.248:11198)] ClassServerDelegate.init(): Reconnected to the class server
     [java] [client: driver-1 (172.16.136.248:11198)] : Attempting connection to the JPPF task server
     [java] [client: driver-1 (172.16.136.248:11198)] : Reconnected to the JPPF task server
     [java]             **********Parallel Calculation of Moment of Inertia of various bodies *******
     [java] |
     [java] The Moment of inertia of a Thin cylindrical shell with open ends,of radius=14.38
     [java]  and mass=7.74is I= =1600.5112560000002|
     [java] |
     [java] The Moment of inertia of a Thick-walled cylindrical tube with open ends,of
     [java]  inner radius=14.38,
     [java]  outer radius=12.38,
     [java] length=12.21 and
     [java]  mass=7.74,with resect to x-axis and y-axis is Ix=Iy==792.0516725,
     [java] The Moment of inertia of a Thick-walled cylindrical tube with open ends,of
     [java] inner radius=14.38,
     [java]  outer radius=12.38  and
     [java]  mass=7.74,with resect to z-axis is Iz= =1391.3888540000003|
     [java] |
     [java] The Moment of inertia of a Solid cylinder of radius=14.38,
     [java]  height=12.21 and
     [java]  mass=7.74,with resect to x-axis and y-axis is Ix=Iy==688.6055475000002,
     [java] The Moment of inertia of a Solid cylinder of radius=14.38,
     [java]  height=12.21 and
     [java]  mass=7.74,with resect to z axis is Iz= =800.2556280000001|
     [java] |
     [java] The Moment of inertia of a Thin solid disk of radius=14.38   and
```

Fig 4-1: output of the application using eclipse IDE and ant builder

The figure Fig 4-1 shows the output of the application using eclipse IDE and ant builder. Table 4-1 and Table 4-2 show the proportional and auto tuned load balancing algorithm data for 2, 4, and 8 nodes. Graph 4-1 and Graph 4-2 are the graphs for showing the relationships among data's of Table 4-1 and Table 4-2 respectively.

Number of Nodes	Average Execution Time(ms)	Average Node Execution Time(ms)	Average Transport Time(ms)	Average Queue Time (ms)
2	84.40	78.00	6.40	80.80
4	92.30	82.80	9.50	40.40
8	112.73	102.20	10.53	26.93

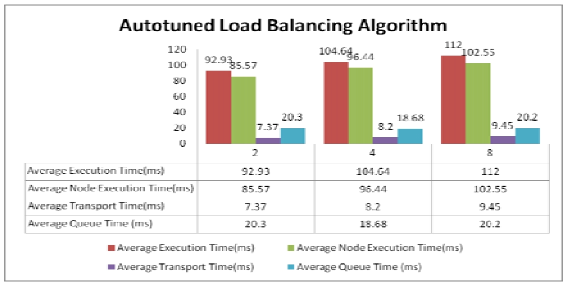
Table -4-1: proportional load balancing algorithm data



Graph -4-1: proportional load balancing algorithm data and graph

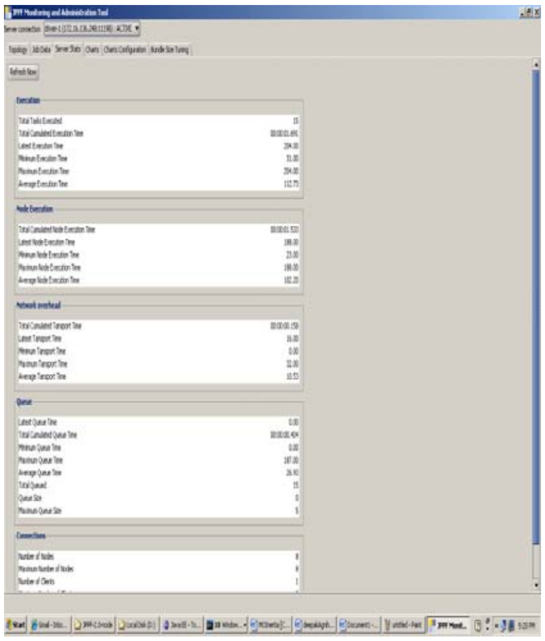
Number of Nodes	Average Execution Time(ms)	Average Node Execution Time(ms)	Average Transport Time(ms)	Average Queue Time (ms)
2	92.93	85.57	7.37	20.30
4	104.64	96.44	8.20	18.68
8	112.00	102.55	9.45	20.20

Table -4-2: auto tuned load balancing algorithm data



Graph -4-2: auto tuned load balancing algorithm

If this application is executed using sequential algorithm that is without using JPPF grid on clusters then it takes about 14-16 seconds ,whether if load balancing algorithm is used in cluster environment then it takes 3-4 seconds. We can say using JPPF grid on cluster environment this parallel application's speedup and efficiency is high over sequential implementation of the application. The figure Fig4-2 and Fig 4-3 shows the server stats, bar chart respectively using JPPF administration and monitoring tool. The figure Fig 4-3 shows the output for sequential implementation i.e. without distributing the task on JPPF grid.



Execution

Total Tasks Executed	15
Total Cumulated Execution Time	00:00:01.691
Latest Execution Time	204.00
Minimum Execution Time	31.00
Maximum Execution Time	204.00
Average Execution Time	112.73

Node Execution

Total Cumulated Node Execution Time	00:00:01.533
Latest Node Execution Time	188.00
Minimum Node Execution Time	23.00
Maximum Node Execution Time	188.00
Average Node Execution Time	102.20

Network overhead

Total Cumulated Transport Time	00:00:00.158
Latest Transport Time	16.00
Minimum Transport Time	0.00
Maximum Transport Time	32.00
Average Transport Time	10.53

Queue

Latest Queue Time	0.00
Total Cumulated Queue Time	00:00:00.404
Minimum Queue Time	0.00
Maximum Queue Time	187.00
Average Queue Time	26.93
Total Queued	15
Queue Size	0
Maximum Queue Size	5

Connections

Number of Nodes	8
Maximum Number of Nodes	8
Number of Clients	1

Fig 4-2: Server stats using JPPF administration and monitoring tool.

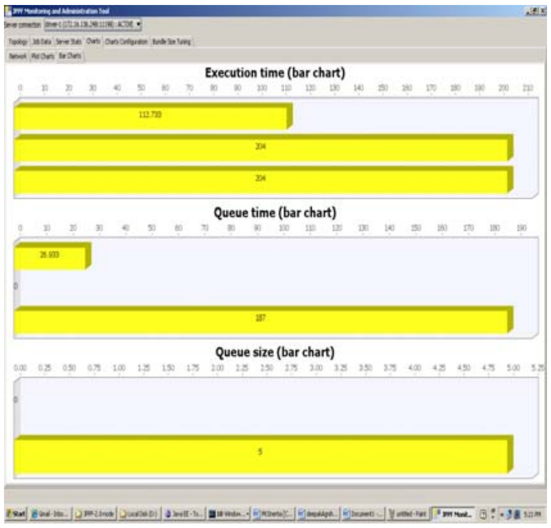


Fig 4-3: bar chart using JPPF administration and monitoring tool.

```
Buildfile: C:\Documents and Settings\Deepak\workspace\TorqueCalculation\build.xml
clean:
  [delete] Deleting directory C:\Documents and
Settings\Deepak\workspace\TorqueCalculation\classes
init:
   [mkdir] Created dir: C:\Documents and
Settings\Deepak\workspace\TorqueCalculation\classes
compile:
   [javac] Compiling 12 source files to C:\Documents and
Settings\Deepak\workspace\TorqueCalculation\classes
build:
run:
    [java] **********Parallel Calculation of Moment of Inertia of various bodies *******
    [java] {
    [java] The Moment of inertia of a Thin cylindrical shell with open ends,of radius=14.38
    [java]  and mass=7.74is I= =1600.5112560000002}
    [java] {
    [java] The Moment of inertia of a Thick-walled cylindrical tube with open ends,of
    [java]  inner radius=14.38,
    [java]  outer radius=12.38,
    [java] length=12.21 and
    [java]  mass=7.74,with resect to x-axis and y-axis is Ix=Iy==792.8536725,
    [java] The Moment of inertia of a Thick-walled cylindrical tube with open ends,of
    [java] inner radius=14.38,
    [java]  outer radius=12.38  and
    [java]  mass=7.74,with resect to z-axis is Iz= =1393.3888560000003}
    [java] {
    [java] The Moment of inertia of a Solid cylinder of radius=14.38,
    [java]  height=12.21 and
    [java]  mass=7.74,with resect to x-axis and y-axis is Ix=Iy==688.6055475000002,
    [java] The Moment of inertia of a Solid cylinder of radius=14.38,
    [java]  height=12.21 and
    [java]  mass=7.74,with resect to z axis is Iz= =800.2556280000001}
    [java] {
    [java] The Moment of inertia of a Thin solid disk of radius=14.38        and
    [java] mass=7.74
    [java] , with resect to x and y axis is
    [java]  Ix=Iy= =400.12781400000006,
    [java] The Moment of inertia of a Thin solid disk of radius=14.38        and
    [java] mass=7.74
    [java] , with resect to z axis is Iz= =800.2556280000001}
    [java] {
    [java] The Moment of inertia of a Thin circular hoop with radius=14.38
    [java]  and mass=7.74,with resect to x and y axis is Ix=Iy= =800.2556280000001,
    [java] The Moment of inertia of a Thin circular hoop with radius=14.38 ,
    [java]  and mass=7.74,with resect to z axis is Iz= =1600.5112560000002}
    [java] {
    [java] The Moment of inertia of a Ball (solid) with radius=14.38and
    [java]  mass=7.74 I= =640.2045024000001}
    [java] {
    [java] The Moment of inertia of an Ellipsoid (solid) of semiaxes a=4.23,
    [java]  b=4.38,
    [java]  and c=5.23,with axis of rotation a=4.23 and mass=7.74I= =72.0397404}
    [java] {
    [java]  The Moment of inertia of a Right circular cone with radius=4.38,height=5.23 and
mass=7.74with resect to x and y axis isIx=Iy==149.299956,
    [java] The Moment of inertia of a Right circular cone with radius=4.38,height=5.23 and
mass=7.74with resect to z axis isIz==44.5461768}
    [java] {
    [java] Solid cuboid of height=5.23,width=12.21, and depth=8.45, and mass=7.74
    [java] the Moment of inertia with resect to Height Ih==142.21385700000002,
    [java] Solid cuboid of height=5.23,width=12.21, and depth=8.45, and mass=7.74
    [java] the Moment of inertia with resect to Width Iw==63.697233,
    [java] Solid cuboid of height=5.23,width=12.21, and depth=8.45, and mass=7.74 The
Moment of inertia with resect to Depth Id==113.80186500000003}
    [java] {
    [java] The Moment of Inertia of a Thin rectangular plate of height h=5.23 and of width
w=12.21and
    [java]     mass=7.74,when axis of rotation is at the end of the plate
Ie==166.72972650000003,
    [java] The Moment of Inertia of a Thin rectangular plate of height h= and of width
w=12.21and
    [java]  mass=7.74,when axis of rotation is at the center of the plate
Ic==113.80186500000003}
    [java] {
    [java] The Moment of inertia of a Rod of length=5.23 and
    [java]     mass=7.74,when Axis of rotation is at the center of the rod
Ic==17.642620500000003,
    [java] nThe Moment of inertia of a Rod of length=5.23 and
    [java]  mass=7.74,when Axis of rotation is at the end of the rodIe==70.57048200000001}
BUILD SUCCESSFUL
Total time: 14 seconds
```

Fig 4-4: output for sequential implementation i.e. without distributing the task on JPPF grid

5. Conclusions

JPPF framework is cost effective, platform independent and easy to implement applications parallel. It is easy to implement for those having prior knowledge of java. All the concepts and utilities of java along with JPPF features can be implemented using this framework. This Framework can be best utilized for those mathematical applications which require lot of calculations. The performance issue varies as nodes increases; it depends upon various factors such as network overhead, distribution of the tasks to the nodes made by the driver, and nodes which use pool of threads to perform the execution of the multiple tasks etc.. Speedup and efficiency of parallel application is great using JPPF grid on cluster environment.

If this application is executed using sequential algorithm that is without using JPPF grid on clusters then it takes about 14-16 seconds ,whether if load balancing algorithm is used in cluster environment then it takes 3-4 seconds. We can say using JPPF grid on cluster environment this parallel application's speedup and efficiency is high over sequential implementation of the application.

This application promotes with great accuracy and speed as platform independent parallel applications: the solution of engineering problems like Moment of Inertia, velocity, viscosity of liquid, elasticity etc. requires lot of calculations and also it should be precise to high degree of accuracy.

References

[1] All about JPPF at http://www.jppf.org/

[2] For JPPF features website is http://linux.softpedia.com/get/Utilities/JPPF-47576.shtml

[3] For JPPF features website is http://www.theserverside.com/news/thread.tss?thread_id=47941

[4] JPPF features at http://grid-comp.blogspot.com/2008/12/jppf-and-gridgain-two-java.html

[5] JPPF features at http://linux.softpedia.com/get/Programming/Libraries/Java-2-Standard-Edition-Runtime-Environment-6-16344.shtml

[6] Moment of Inertia at http://en.wikipedia.org/wiki/Moment_of_inertia

[7] Moment of Inertia at http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html

[8] The Complete Reference Java Seventh edition,Herbert Schildt

[9] Java Parallel Programming Framework: JPPF by Murray Foote & John HetheringtonBOPPOLY Taranga, NZ at http://naccq08.unitec.ac.nz/proceedings/papers/351.pdf