**Cloud Lab Exercises/Experiments**

1. **Write a program to understand the differences between conventional Thread programming and Cloud programming using Aneka Thread Model.**
2. **Write a program to print “Hello World” based on Thread model and use exactly five threads also print the executor node information along with the Submission Time and Completion Time?**
3. **Write a program to compute the following mathematical equation using Aneka Threads(Note: Consider each trigonometric function in independent thread)?**

**P= sin(x) + cos(y) + tan(z)**

1. **Write a program to compute the matrix addition using Aneka Thread Programming Model**
2. **Write a program for parallel execution of *Mandelbrot generation algorithm in parallel using Aneka Threads***
3. **Write a program to compute the matrix multiplication using Aneka Thread Programming Model**
4. **Write a program to decompose the image into 25 parts(5X5) and apply histogram(dynamic stretch)**
5. **Write a program to parse the log files using Map/Reduce or Thread Programming Model**
6. **Write a program using Map/Reduce to count the words in the given input set**
7. **Write a program for Image Convolution using Task Programming Model**
8. **Write a program for sorting large number of records (say 1000 entries) stored in a file. You can create N threads (say 4 or 10) and each Thread taking responsibility for sorting a part of file (e.g., Thread1 can sort first 1000/N record; and Thread 2 can sort next set and continues) and a Master Thread can merge all these sorted sub-set of records.**
9. **Write a program for the tabulation of the Gaussian function by using simple threads and the convert it to Aneka threads.**
10. **Write a program that executes operations of data mining algorithms in parallel on Clouds.**
11. **Write a program for parallel execution of Ray Tracing operations. You can pick up sequential Ray tracking programs such as PovRay for rendering and convert it for parallel execution using Aneka Task programming model.**
12. **Write a program for parallel execution of bioinformatics algorithms such as Basic Local Alignment Search Tool (BLAST) used for comparing primary biological sequence information**. **You can select a suitable Cloud programming model and also implement a graphical user interface for start execution of applications.**

Answers to Selected Programming Exercises

**Thread Model**

**Qn : Write a program to print “Hello World” using Aneka Thread Programming model use Single Thread?**

**Procedure**

Follow the below mentioned steps to create the project and the necessary references

1. Creating a C# Console Application in .NET 2.0
   1. Go to File -> New -> Project.
   2. From New Project Dialog select visual C# - > windows .
   3. Select .NET 2.0 and Console Application.
   4. Enter the Solution Name and press OK.
2. Add References to the project from the following folders under Solution Explorer
   1. Right Click on the references
      1. Click on Add Reference
      2. Click on Browse Tab
      3. Select the following folders one after the other and add them

* C:\Program Files\Manjrasoft\Aneka.3.0\Tools\SDK\Common
* C:\Program Files\Manjrasoft\Aneka.3.0\Tools\SDK\Runtime
* C:\Program Files\Manjrasoft\Aneka.3.0\Tools\SDK\Thread Model

1. Place the conf.xml in the folder and mention the same path in the code.
2. Add the below mentioned code to .CS file by replacing the code in the appropriate place

**Source code**

using System;

using System.Collections.Generic;

using System.Text;

using Aneka;

using Aneka.Threading;

using Aneka.Entity;

using System.Threading;

namespace AnekaThreadPractise1

{

[Serializable]

public class HelloWorld

{

public string result;

public HelloWorld()

{

}

public void PrintHello()

{

result = "HelloWorld";

}

}

class Program

{

static void Main(string[] args)

{

AnekaApplication<AnekaThread, ThreadManager> app = null;

try

{

Logger.Start();

Configuration conf = Configuration.GetConfiguration(@"C:\Users\raghav\Documents\CloudComputing-Lectures\conf.xml");

app = new AnekaApplication<AnekaThread, ThreadManager>(conf);

HelloWorld hw = new HelloWorld();

AnekaThread th = new AnekaThread(hw.PrintHello, app);

th.Start();

th.Join();

hw = (HelloWorld)th.Target;

Console.WriteLine("Value : " + hw.result);

}

finally

{

app.StopExecution();

Logger.Stop();

}

}

}

}

**Ouput :**

Value : Helloworld

**Qn : Write a program to print “Hello World” based on Thread model and use exactly five threads also print the executor node information along with the Submission Time and Completion Time?**

using System;

using System.Collections.Generic;

using System.Text;

using Aneka;

using Aneka.Threading;

using Aneka.Entity;

using System.Threading;

namespace AnekaThreadsfive

{

[Serializable]

public class HelloWorld

{

public string result;

public HelloWorld()

{

}

public void PrintHello()

{

result = "HelloWorld";

}

}

class Program

{

static void Main(string[] args)

{

AnekaApplication<AnekaThread, ThreadManager> app = null;

try

{

Logger.Start();

Configuration conf = Configuration.GetConfiguration(@"C:\Users\raghav\Documents\CloudComputing-Lectures\conf.xml");

app = new AnekaApplication<AnekaThread, ThreadManager>(conf);

HelloWorld hw = new HelloWorld();

AnekaThread[] th = new AnekaThread[5];

for (int i = 0; i < 5; i++)

{

th[i] = new AnekaThread(hw.PrintHello, app);

th[i].Start();

}

for (int i = 0; i < 5; i++)

{

th[i].Join();

hw = (HelloWorld)th[i].Target;

Console.WriteLine("Value : {0} , NodeId:{1},Submission Time:{2},Completion Time{3}", hw.result, th[i].NodeId,th[i].SubmissionTime,th[i].CompletionTime);

}

}

finally

{

app.StopExecution();

Logger.Stop();

}

}

}

}

**Qn : Write a program to print “Hello World” using Aneka Thread Programming model and Conventional Thread and Understand the differences?**

**Procedure**

Follow the steps mentioned from 1 to 4 of mentioned in the procedure of the previous Question

**Source code**

using System;

using System.Collections.Generic;

using System.Text;

using Aneka;

using Aneka.Threading;

using Aneka.Entity;

using System.Threading;

namespace AnekaThreadPractise1

{

[Serializable]

public class HelloWorld

{

public string result;

public HelloWorld()

{

}

public void PrintHello()

{

Console.WriteLine("inside printHello : HelloWorld");

result = "HelloWorld";

}

}

class Program

{

static void Main(string[] args)

{

AnekaApplication<AnekaThread, ThreadManager> app = null;

try

{

Logger.Start();

Configuration conf = Configuration.GetConfiguration(@"C:\Users\raghav\Documents\CloudComputing-Lectures\conf.xml");

app = new AnekaApplication<AnekaThread, ThreadManager>(conf);

HelloWorld hw = new HelloWorld();

AnekaThread th = new AnekaThread(hw.PrintHello, app);

th.Start();

th.Join();

hw = (HelloWorld)th.Target;

Console.WriteLine("Value : " + hw.result);

HelloWorld conventionalhw = new HelloWorld();

Thread conventionalThread = new Thread(conventionalhw.PrintHello);

conventionalThread.Start();

}

finally

{

app.StopExecution();

Logger.Stop();

}

}

}

}

**Ouput :**

Value : HelloWorld

inside printHello : HelloWorld

**Qn : Write a program to print “Hello World” using Aneka Thread Programming model use Five Threads , also print the NodeIds on which the threads are executed and submission time and Completion Time of the Threads?**

**Procedure**

1.Followthe steps mentioned from 1 to 4 of mentioned in the procedure of the previous Question

**Source code**

using System;

using System.Collections.Generic;

using System.Text;

using Aneka;

using Aneka.Threading;

using Aneka.Entity;

using System.Threading;

namespace AnekaThreadsfive

{

[Serializable]

public class HelloWorld

{

public string result;

public HelloWorld()

{

}

public void PrintHello()

{

result = "HelloWorld";

}

}

class Program

{

static void Main(string[] args)

{

AnekaApplication<AnekaThread, ThreadManager> app = null;

try

{

Logger.Start();

Configuration conf = Configuration.GetConfiguration(@"C:\Users\raghav\Documents\CloudComputing-Lectures\conf.xml");

app = new AnekaApplication<AnekaThread, ThreadManager>(conf);

HelloWorld hw = new HelloWorld();

AnekaThread[] th = new AnekaThread[5];

for (int i = 0; i < 5; i++)

{

th[i] = new AnekaThread(hw.PrintHello, app);

th[i].Start();

}

for (int i = 0; i < 5; i++)

{

th[i].Join();

hw = (HelloWorld)th[i].Target;

Console.WriteLine("Value : {0} , NodeId:{1},Submission Time:{2},Completion Time{3}", hw.result, th[i].NodeId,th[i].SubmissionTime,th[i].CompletionTime);

}

}

finally

{

app.StopExecution();

Logger.Stop();

}

}

}

}

**output**

Value : HelloWorld , NodeId:raghav-PC-9090,Submission Time:20-Nov-12 11:19:45 PM,Completion Time20-Nov-12 11:19:45 PM

Value : HelloWorld , NodeId:raghav-PC-9090,Submission Time:20-Nov-12 11:19:45 PM,Completion Time20-Nov-12 11:19:45 PM

Value : HelloWorld , NodeId:raghav-PC-9090,Submission Time:20-Nov-12 11:19:45 PM,Completion Time20-Nov-12 11:19:45 PM

Value : HelloWorld , NodeId:raghav-PC-9090,Submission Time:20-Nov-12 11:19:45 PM,Completion Time20-Nov-12 11:19:45 PM

Value : HelloWorld , NodeId:raghav-PC-9090,Submission Time:20-Nov-12 11:19:45 PM,Completion Time20-Nov-12 11:19:45 PM

**Qn : Write a program to compute the following mathematical equation using Aneka Threads(Note: Consider each trigonometric function in independent thread)?**

**P= sin(x) + cos(y) + tan(z)**

**Procedure**

using System;

using System.Collections.Generic;

using System.Text;

using Aneka;

using Aneka.Threading;

using Aneka.Entity;

namespace AnekaThreadTrigonometric

{

[Serializable]

public class Sine

{

double x;

public double result;

public Sine(double x)

{

this.x = x;

}

public void sineCompute()

{

result = System.Math.Sin(x\*System.Math.PI/180);

}

}

[Serializable]

public class Cosine

{

double x;

public double result;

public Cosine(double x)

{

this.x = x;

}

public void cosineCompute()

{

result = System.Math.Cos(x \* System.Math.PI / 180);

}

}

[Serializable]

public class Tangent

{

double x;

public double result;

public Tangent(double x)

{

this.x = x;

}

public void tangentCompute()

{

result = System.Math.Tan(x \* System.Math.PI / 180);

}

}

class Program

{

static void Main(string[] args)

{

AnekaApplication<AnekaThread, ThreadManager> app = null;

try

{

Logger.Start();

Configuration conf = Configuration.GetConfiguration(@"C:\Users\raghav\Documents\CloudComputing-Lectures\conf.xml");

app = new AnekaApplication<AnekaThread, ThreadManager>(conf);

Sine sine = new Sine(10);

AnekaThread thsine = new AnekaThread(sine.sineCompute, app);

thsine.Start();

thsine.Join();

sine = (Sine)thsine.Target;

Cosine cosine = new Cosine(20);

AnekaThread thcosine = new AnekaThread(cosine.cosineCompute, app);

thcosine.Start();

thcosine.Join();

cosine = (Cosine)thcosine.Target;

Tangent tanget = new Tangent(20);

AnekaThread thtangent = new AnekaThread(tanget.tangentCompute, app);

thtangent.Start();

thtangent.Join();

tanget = (Tangent)thtangent.Target;

Console.WriteLine("P = sin(10)+cos(20)+tan(20) = {0}", sine.result + cosine.result + tanget.result);

}

finally

{

app.StopExecution();

Logger.Stop();

}

}

}

}

**Output**

P = sin(10)+cos(20)+tan(20) = 1.47731103271904

**Task Model**

**Qn : Write a program to print “Hello World” using Aneka Task Programming model ?**

**Procedure**

Follow the below mentioned steps to create the project and the necessary references

1. Creating a C# Console Application in .NET 2.0
   1. Go to File -> New -> Project.
   2. From New Project Dialog select visual C# - > windows .
   3. Select .NET 2.0 and Console Application.
   4. Enter the Solution Name and press OK.
2. Add References to the project from the following folders under Solution Explorer
   1. Right Click on the references
      1. Click on Add Reference
      2. Click on Browse Tab
      3. Select the following folders one after the other and add them

* C:\Program Files\Manjrasoft\Aneka.3.0\Tools\SDK\Common
* C:\Program Files\Manjrasoft\Aneka.3.0\Tools\SDK\Runtime
* C:\Program Files\Manjrasoft\Aneka.3.0\Tools\SDK\TaskModel

1. Place the conf.xml in the folder and mention the same path in the code.
2. Add the below mentioned code to .CS file by replacing the code in the appropriate place

**Program**

using System;

using System.Collections.Generic;

using System.Text;

using Aneka;

using Aneka.Tasks;

using Aneka.Entity;

using System.Threading;

namespace AnekaTaskDemo

{

[Serializable]

public class HelloTask : ITask

{

public string result;

public void Execute()

{

result = "Hello";

Console.WriteLine("Result is set : "+result);

Console.WriteLine("HelloWorld");

}

}

class Program

{

static AutoResetEvent semaphore = null;

static AnekaApplication<AnekaTask, TaskManager> app = null;

static void Main(string[] args)

{

try

{

Logger.Start();

semaphore = new AutoResetEvent(false);

Configuration conf = Configuration.GetConfiguration(@"C:\Users\raghav\Documents\CloudComputing-Lectures\conf.xml");

// conf.SingleSubmission = false;

app = new AnekaApplication<AnekaTask, TaskManager>(conf);

app.WorkUnitFailed += new EventHandler<WorkUnitEventArgs<AnekaTask>>(app\_WorkUnitFailed);

app.WorkUnitFinished += new EventHandler<WorkUnitEventArgs<AnekaTask>>(app\_WorkUnitFinished);

app.ApplicationFinished += new EventHandler<ApplicationEventArgs>(app\_ApplicationFinished);

HelloTask ht = new HelloTask();

AnekaTask gt = new AnekaTask(ht);

app.ExecuteWorkUnit(gt);

semaphore.WaitOne();

}

finally

{

Logger.Stop();

}

}

static void app\_ApplicationFinished(object sender, ApplicationEventArgs e)

{

semaphore.Set();

}

static void app\_WorkUnitFailed(object sender, WorkUnitEventArgs<AnekaTask> e)

{

Console.WriteLine("WorkUnit Failed");

}

static void app\_WorkUnitFinished(object sender, WorkUnitEventArgs<AnekaTask> e)

{

Console.WriteLine("Inside WorkUnit finished");

HelloTask ht = e.WorkUnit.UserTask as HelloTask;

Console.WriteLine(ht.result);

app.StopExecution();

}

}

}

**Qn : Write a program to sum the two numbers using Aneka Task Programming model ?**

**Program**

using System;

using System.Collections.Generic;

using System.Text;

using Aneka;

using Aneka.Entity;

using Aneka.Tasks;

using System.Threading;

namespace AnekaTaskPractise

{

[Serializable]

public class MyTask : ITask

{

public int a, b;

public int sum;

public MyTask(int a, int b) { this.a = a; this.b = b; }

public void Execute()

{

Console.WriteLine("Inside Execute");

sum = a + b;

}

}

class Program

{

static AutoResetEvent semaphore = null;

static AnekaApplication<AnekaTask, TaskManager> app = null;

static void Main(string[] args)

{

Configuration conf=null;

AnekaTask gt = null;

try

{

Logger.Start();

semaphore = new AutoResetEvent(false);

conf = Configuration.GetConfiguration(@"C:\Users\raghav\Documents\CloudComputing-Lectures\conf.xml");

conf.SingleSubmission = false;

app = new AnekaApplication<AnekaTask, TaskManager>(conf);

app.WorkUnitFailed += new EventHandler<WorkUnitEventArgs<AnekaTask>>(app\_WorkUnitFailed);

app.WorkUnitFinished += new EventHandler<WorkUnitEventArgs<AnekaTask>>(app\_WorkUnitFinished);

app.ApplicationFinished += new EventHandler<ApplicationEventArgs>(app\_ApplicationFinished);

MyTask task = new MyTask(10,20);

gt = new AnekaTask(task);

app.ExecuteWorkUnit(gt);

semaphore.WaitOne();

}

finally

{

Logger.Stop();

}

}

static void app\_ApplicationFinished(object sender, ApplicationEventArgs e)

{

semaphore.Set();

}

static void app\_WorkUnitFinished(object sender, WorkUnitEventArgs<AnekaTask> e)

{

Console.WriteLine("Workunit finished:"+((MyTask)e.WorkUnit.UserTask).sum);

app.StopExecution();

}

static void app\_WorkUnitFailed(object sender, WorkUnitEventArgs<AnekaTask> e)

{

}

}

}