**Cloud Lab Exercises/Experiments in C++**

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.**

Note: This tutorial uses .NET 4.6 or lower version

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 4.6 (CLR)
   1. Go to File -> New -> Project.
   2. From New Project Dialog select Visual C++ -> CLR Application - > windows .
   3. Select .NET 4.6/4.6.1/4.6.2 (only one of these) 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.5.0\Tools\SDK\Common
* C:\Program Files\Manjrasoft\Aneka.5.0\Tools\SDK\Runtime
* C:\Program Files\Manjrasoft\Aneka.5.0\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 .cpp file by replacing the code in the appropriate place

**Source code**

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace System;

using namespace std;

using namespace Aneka;

using namespace Aneka::Threading;

using namespace Aneka::Entity;

[Serializable]

ref class HelloWorld

{

public:

int result;

HelloWorld() {}

void PrintHello()

{

result = 1;

}

};

int main()

{

AnekaApplication<AnekaThread^, ThreadManager^>^ app;

try {

Logger::Start();

Aneka::Entity::Configuration^ conf = Aneka::Entity::Configuration::GetConfiguration("C:/Aneka/conf.xml");

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

HelloWorld^ hw = gcnew HelloWorld();

AnekaThread^ th = gcnew AnekaThread(hw->PrintHello(), app);

th->Start();

th->Join();

hw = (HelloWorld^)th->Target;

std::cout << "Value: " << hw->result;

}

catch(int e){

;

}

finally{

app->StopExecution();

Logger::Stop();

}

return 0;

}

**Ouput :**

Value: 1

**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?**

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace System;

using namespace std;

using namespace Aneka;

using namespace Aneka::Threading;

using namespace Aneka::Entity;

[Serializable]

ref class HelloWorld

{

public:

int result;

HelloWorld() {}

void PrintHello()

{

result = 1;

}

};

int main()

{

AnekaApplication<AnekaThread^, ThreadManager^>^ app;

try {

Logger::Start();

Aneka::Entity::Configuration^ conf = Aneka::Entity::Configuration::GetConfiguration("C:/Aneka/conf.xml");

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

HelloWorld^ hw = gcnew HelloWorld();

AnekaThread^ th = gcnew AnekaThread(hw->PrintHello(), app);

th->Start();

th->Join();

hw = (HelloWorld^)th->Target;

std::cout << "Value: " << hw->result;

}

catch(int e){

;

}

finally{

app->StopExecution();

Logger::Stop();

}

return 0;

}

**Qn : Write a program to print “Hello World” using Aneka Thread Programming model on 5 threads?**

**Procedure**

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

**Source code**

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace System;

using namespace std;

using namespace Aneka;

using namespace Aneka::Threading;

using namespace Aneka::Entity;

[Serializable]

ref class HelloWorld

{

public:

int result;

HelloWorld() {}

void PrintHello()

{

result = 1;

}

};

int main()

{

AnekaApplication<AnekaThread^, ThreadManager^>^ app;

try {

Logger::Start();

Aneka::Entity::Configuration^ conf = Aneka::Entity::Configuration::GetConfiguration("C:/Aneka/conf.xml");

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

HelloWorld^ hw = gcnew HelloWorld();

AnekaThread^ th = gcnew AnekaThread(hw->PrintHello(), app);

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

th->Start();

th = gcnew AnekaThread(hw->PrintHello(), app);

}

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

th->Join();

hw = (HelloWorld^)th->Target;

std::cout << "Value: " << hw->result;

}

}

catch(int e){

;

}

finally{

app->StopExecution();

Logger::Stop();

}

return 0;

}

**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 4.6 (CLR)
   1. Go to File -> New -> Project.
   2. From New Project Dialog select Visual C++ -> CLR Application - > windows .
   3. Select .NET 4.6/4.6.1/4.6.2 (only one of these) 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.5.0\Tools\SDK\Common
* C:\Program Files\Manjrasoft\Aneka.5.0\Tools\SDK\Runtime
* C:\Program Files\Manjrasoft\Aneka.5.0\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 .cpp file by replacing the code in the appropriate place

**Program**

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace System;

using namespace std;

using namespace Aneka;

using namespace Aneka::Threading;

using namespace Aneka::Entity;

using namespace Aneka::Tasks;

[Serializable]

ref class HelloWorld

{

public:

int result;

HelloWorld() {}

void Execute()

{

result = 1;

}

};

int main()

{

AnekaApplication<AnekaTask^, ThreadManager^>^ app;

try {

Logger::Start();

Aneka::Entity::Configuration^ conf = Aneka::Entity::Configuration::GetConfiguration("C:/Aneka/conf.xml");

app = gcnew AnekaApplication<AnekaTask^, ThreadManager^>(conf);

HelloWorld^ hw = gcnew HelloWorld();

AnekaTask^ th = gcnew AnekaTask(hw);

app->ExecuteWorkUnit(th);

}

catch(int e){

;

}

finally{

app->StopExecution();

Logger::Stop();

}

return 0;

}