Lab 07: Distributed Threading with Aneka

Dhiraj K. Pandey, PhD

Assistant Professor [1]

[1] Central Department of CSIT, Tribhuvan University, Nepal

Distributed Threading with Aneka

- **Objective**: Implement distributed trigonometric computations using AnekaThreads
- **Tools**:
  - Aneka SDK
  - Serializable classes
- **Prerequisite**: Basic threading knowledge
- **Resource**: `https://www.manjrasoft.com/ParallelProgrammingWithAnekaThreads-Chapter.pdf`
- **Duration**: 90 minutes

Lab Objectives

1. Understand differences between AnekaThreads and .NET Threads
2. Implement serializable worker classes for remote execution
3. Deploy and synchronize distributed threads across cloud nodes
4. Combine results from parallel remote computations
5. Analyze distributed vs local execution performance (conceptually)

AnekaThreads Overview

- **Distributed Thread Model**:
  - Extends .NET threading to cloud nodes
  - Requires `AnekaApplication` gateway
- **Execution Flow**:
  - Client creates and starts threads
  - Aneka routes to worker nodes
  - Results aggregated post-completion

Problem Statement

Compute the equation:

$$p = \sin(x) + \cos(y) + \tan(z)$$

Parallelization Strategy:

- Three independent trigonometric operations
- Each function executes in separate AnekaThread
- Threads distributed to different worker nodes
- Main thread aggregates results after completion

Key Insight:

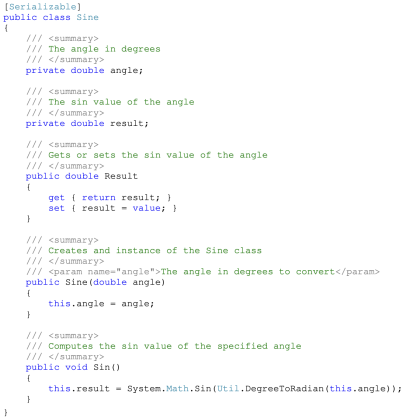
- No dependencies between computations
- Embarrassingly parallel problem

Serializable Worker Classes

Requirements:

- Mark with `[Serializable]` attribute
- Encapsulate both data and computation logic
- Parameterless worker methods

Example - Sine Class:

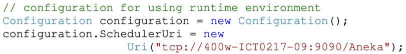


```
[Serializable]
public class Sine
{
    /// <summary>
    /// The angle in degrees
    /// </summary>
    private double angle;

    /// <summary>
    /// The sin value of the angle
    /// </summary>
    private double result;

    /// <summary>
    /// Gets or sets the sin value of the angle
    /// </summary>
    public double Result
    {
        get { return result; }
        set { result = value; }
    }

    /// <summary>
    /// Creates and instance of the Sine class
    /// </summary>
    /// <param name="angle">The angle in degrees to convert</param>
    public Sine(double angle)
    {
        this.angle = angle;
    }

    /// <summary>
    /// Computes the sin value of the specified angle
    /// </summary>
    public void Sin()
    {
        this.result = System.Math.Sin(Util.DegreeToRadian(this.angle));
    }
}
```

AnekaThread Initialization

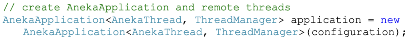
Implementation Steps:

1. Configure Aneka runtime:



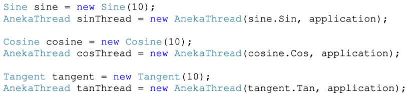
```
// configuration for using runtime environment
Configuration configuration = new Configuration();
configuration.SchedulerUri = new
                    Uri("tcp://400w-ICT0217-09:9090/Aneka");
```

2. Create application context:



```
// create AnekaApplication and remote threads
AnekaApplication<AnekaThread, ThreadManager> application = new
   AnekaApplication<AnekaThread, ThreadManager>(configuration);
```

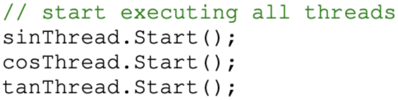
3. Initialize threads:



```
Sine sine = new Sine(10);
AnekaThread sinThread = new AnekaThread(sine.Sin, application);

Cosine cosine = new Cosine(10);
AnekaThread cosThread = new AnekaThread(cosine.Cos, application);

Tangent tangent = new Tangent(10);
AnekaThread tanThread = new AnekaThread(tangent.Tan, application);
```

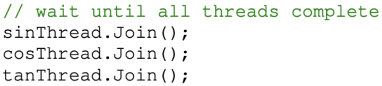
Thread Execution Flow

1. Start distributed execution:



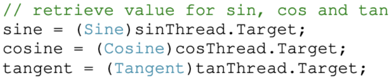
```
// start executing all threads
sinThread.Start();
cosThread.Start();
tanThread.Start();
```

2. Synchronize completion:



```
// wait until all threads complete
sinThread.Join();
cosThread.Join();
tanThread.Join();
```

3. Retrieve results:



```
// retrieve value for sin, cos and tan
sine = (Sine)sinThread.Target;
cosine = (Cosine)cosThread.Target;
tangent = (Tangent)tanThread.Target;
```

Lab Tasks & Submission

Required Tasks:

1. Implement the trigonometric computation
2. Extend to support user-input angles
3. Compare local vs distributed execution time

Submission:

- Source code (.cs files)
- Screenshot of summed result
- 1-page report containing:
  - Performance observations
  - Challenges faced
  - Answers to reflection questions