Corso di Sistemi Distribuiti

Corso di Laurea Magistrale in Ingegneria Informatica A.A. 2014/2015

DIMES - Università degli Studi della Calabria

DEVELOPMENT SERVICES ON WINDOWS AZURE

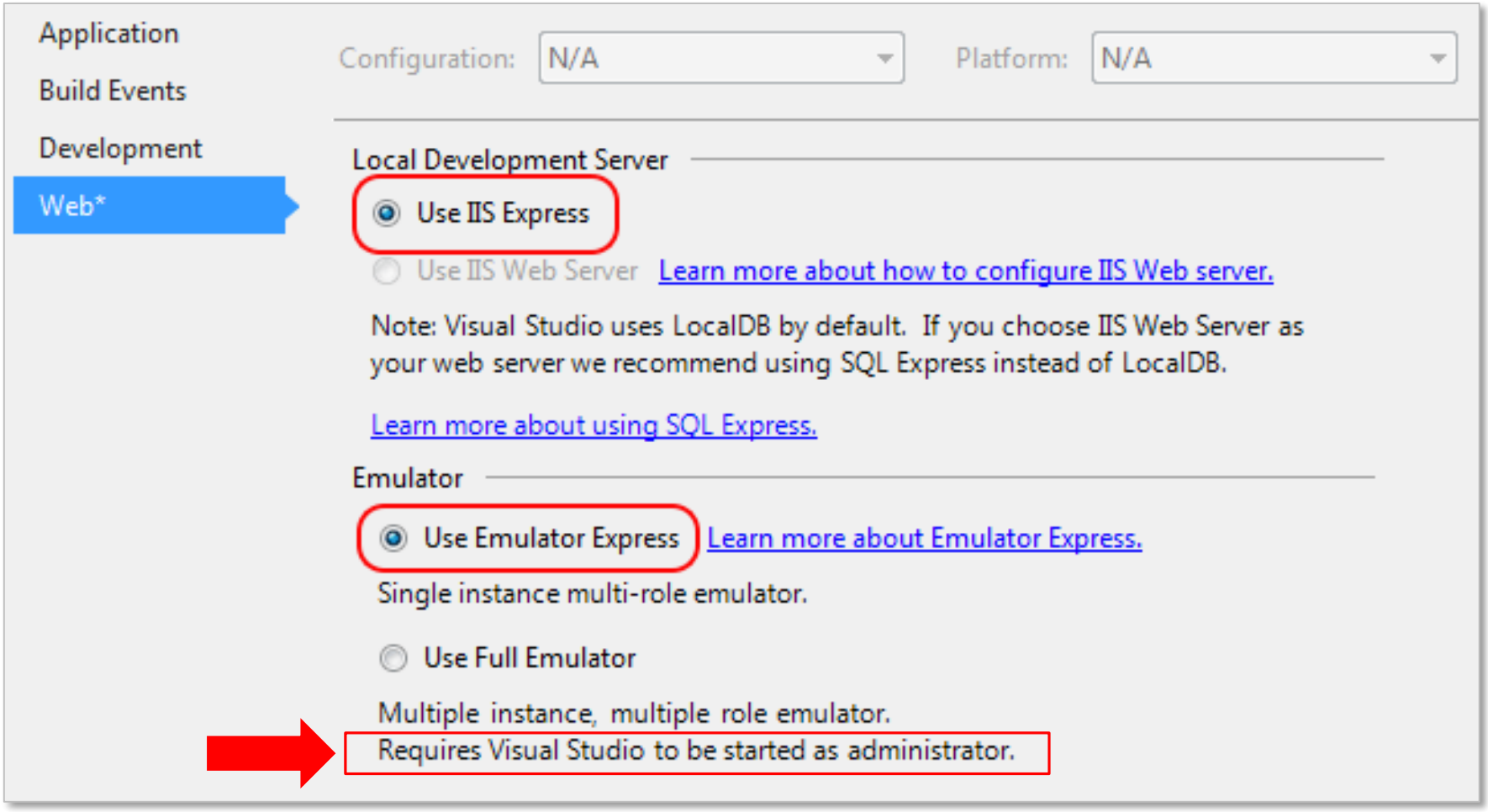
STRING INVERSION EXPLOITING TWO QUEUES

MATRIX MULTIPLICATION

Ing. Loris Belcastro

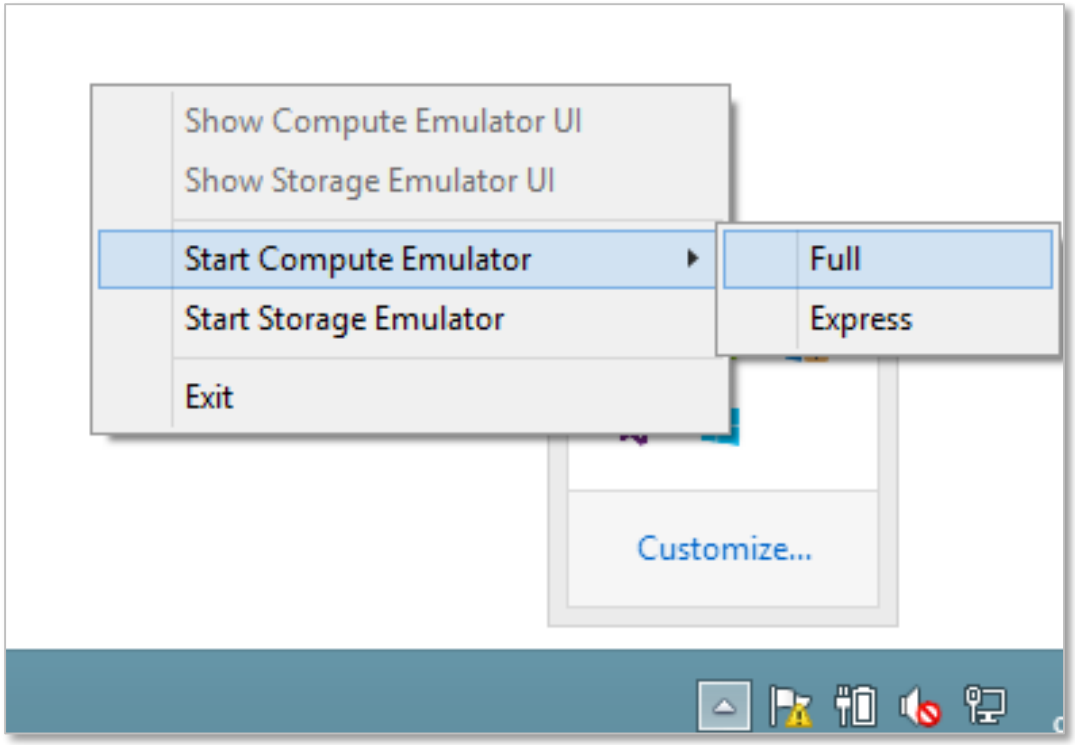
How to run multiple instances per role

- To run more than one instance per role you need to use the **Full Emulator**
- On the shortcut menu for the Azure project, choose Properties, and then choose the Web tab.



How to run Compute Full Emulator

- If emulator is running, then first kill it by going into task manager.
- Right click on "csmonitor.exe" in "C:\Program Files\Microsoft SDKs\Windows Azure\Emulator" directory and run that as Administrator.
- Next right click on "Azure Emulator" icon in the system tray and start compute emulator in "Full" mode as shown in screenshot below.



Summary

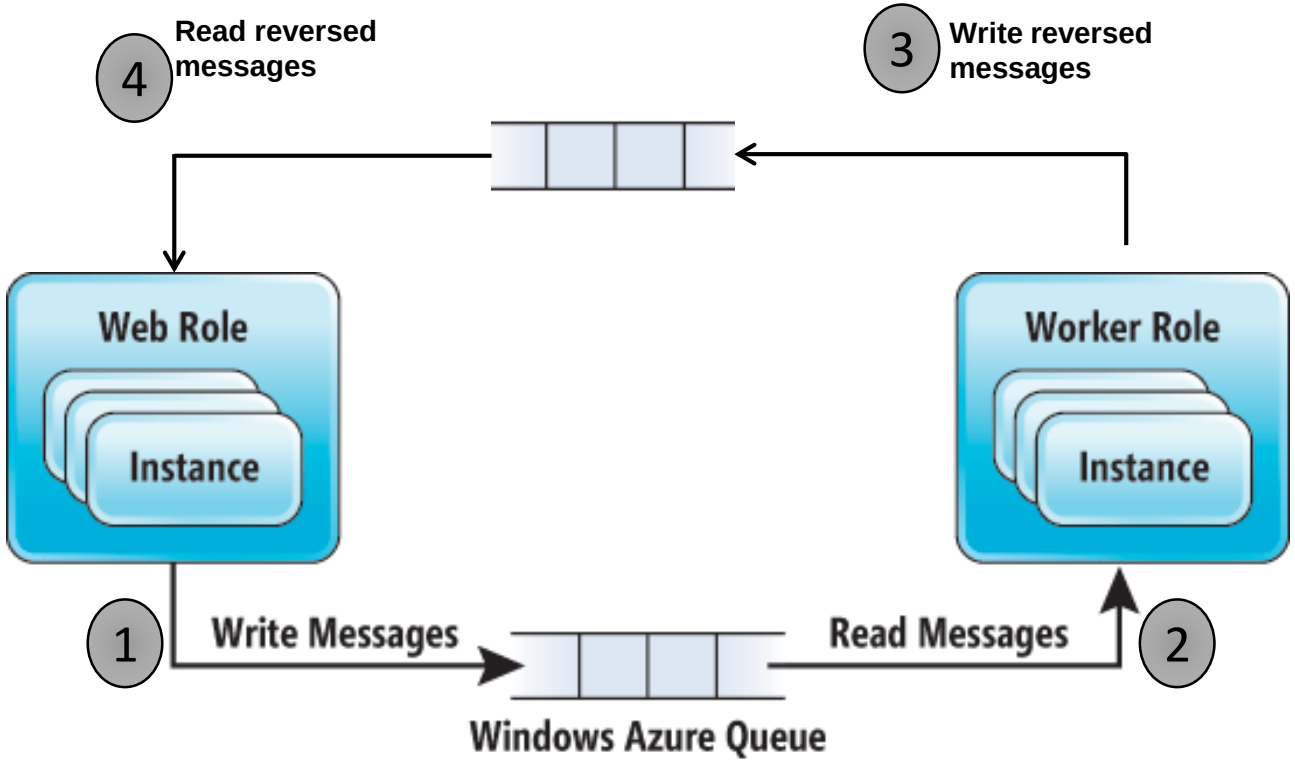
- Reversing a string, using input and output queues
- Matrix multiplication (more workers)

Summary

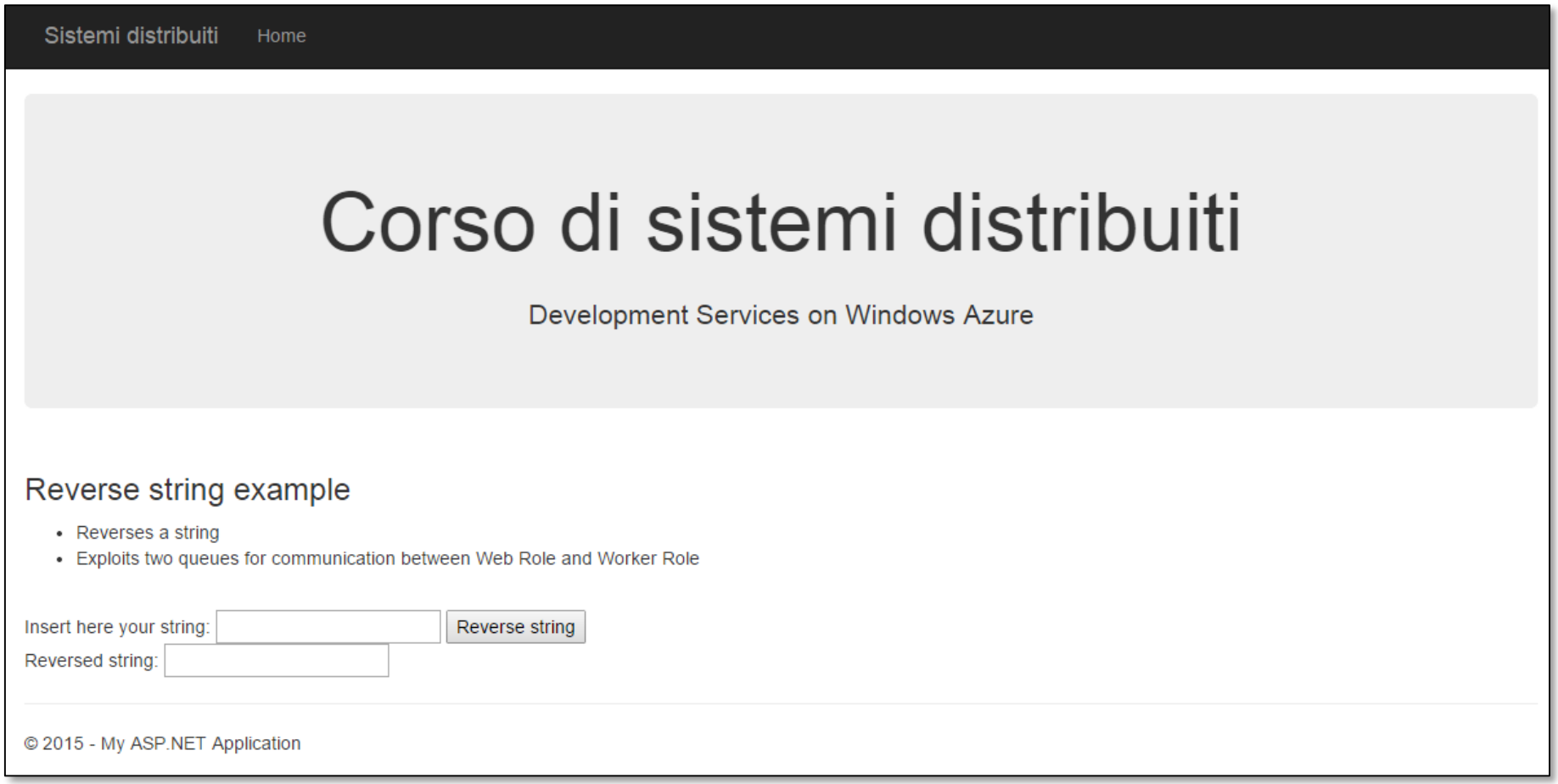
- Reversing a string, using input and output queues
- Matrix multiplication (more workers)

Reverse string, with Input and Output Queues

- Create a simple Cloud Service that:
  - reverses a string
  - exploits two queues for communication between Web Role and Worker Role



Web Interface



Code snippets – Web Role

- Default.aspx file

```
...
  <div class="col-md-12">
            <asp:Label ID="Label1" runat="server" Text="Insert here your string:"></asp:Label>
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <asp:Button ID="Button2" runat="server" Text="Reverse string" OnClick="btnSend_Click" />
        </div>
        <div class="col-md-12">
            <asp:Label ID="Label2" runat="server" Text="Reversed string:"></asp:Label>
            <asp:TextBox ID="TextBox3" runat="server></asp:TextBox>
        </div>
 ...
```

Code snippets – Web Role

- Default.aspx.cs file

```
…
public partial class _Default : Page
    {
        private CloudQueue inputQueue;
        private CloudQueue outputQueue;

        protected void Page_Load(object sender, EventArgs e)
        {
            CloudStorageAccount storageAccount =
                        CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("StorageConnectionString"));

            // Create the queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

            // Retrieve a reference to input and output queues
            inputQueue = queueClient.GetQueueReference("inputqueue");
            inputQueue.CreateIfNotExists();
            outputQueue = queueClient.GetQueueReference("outputqueue");
            outputQueue.CreateIfNotExists();

        }//Page_Load
    …
```

Code snippets – Web Role

```
...
public partial class _Default : Page {
...
        protected void btnSend_Click(object sender, EventArgs e) {
            // Create a message and add it to the queue.
            CloudQueueMessage msg = new CloudQueueMessage(txtMessage.Text);
            inputQueue.AddMessage(msg);
            txtMessage.Text = string.Empty;

            // retrieve messages and write them to the compute emulator log
            bool b = false;
            while (!b) {
                Thread.Sleep(500);
                msg = outputQueue.GetMessage();
                if (msg != null) {
                    responseMessage.Text = msg.AsString;
                    outputQueue.DeleteMessage(msg);
                    b = true;
                }//if
            }//while
        }//btnSend_Click
  }
...
```

Code snippets – Worker Role

- WorkerRole.cs file (onStart() method)

```
…
 public class WorkerRole : RoleEntryPoint {
        private CloudQueue inputQueue;
        private CloudQueue outputQueue;

        public override bool OnStart() {
            // Impostare il numero massimo di connessioni simultanee
            ServicePointManager.DefaultConnectionLimit = 12;
            CloudStorageAccount storageAccount =
               CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("StorageConnectionString"));

            // Create the queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

            // Retrieve a reference to input and output queues
            inputQueue = queueClient.GetQueueReference("inputqueue");
            inputQueue.CreateIfNotExists();
            outputQueue = queueClient.GetQueueReference("outputqueue");
            outputQueue.CreateIfNotExists();

            return base.OnStart();
        }
…
```

Code snippets – Worker Role

- WorkerRole.cs file (Run() method)

```
...
 public override void Run() {
          Trace.TraceInformation("ReverseStringWorkerRole entry point called");
          int workerInstanceId = getWorkerId();

           // retrieve messages and write them to the compute emulator log
           while (true) {
               Thread.Sleep(500);
               if (inputQueue.Exists()) {
                   var msg = inputQueue.GetMessage();
                   if (msg != null) {
                       Trace.TraceInformation(string.Format(" Message '{0}' received by the worker
                                '{1}'.", msg.AsString, workerInstanceId));
                       string str = msg.AsString;
                       string reversed_str = reverseString(str);
                       Trace.TraceInformation(" Reversed message built: " + reversed_str + "\n");
                       inputQueue.DeleteMessage(msg);

...
```

Code snippets – Worker Role

- WorkerRole.cs file (Run() method)

```
...
 public override void Run() {
           ...
                        //create and add the return message
                        var reversedMsg = new CloudQueueMessage(reversed_str);
                        outputQueue.AddMessage(reversedMsg);
                        Trace.TraceInformation(
                                "Inserted the response " + reversed_str + " for the Web Role");

                    }//if
                }//if
            }//while

 }//Run
...
```

Code snippets – Worker Role

- WorkerRole.cs file (utility methods)

```
...
        //return the instance worker ID
        private int getWorkerId() {
            string instanceId = RoleEnvironment.CurrentRoleInstance.Id;
            int instanceIndex;
            //if (int.TryParse(instanceId.Substring(instanceId.LastIndexOf(".") + 1), out
                    instanceIndex)) // On cloud.
            int.TryParse(instanceId.Substring(instanceId.LastIndexOf("_") + 1), out instanceIndex);
            // On compute emulator.
            // instanceIndex is begin from 0. The instanceIndex of the first instance is 0.
            return instanceIndex;
        } //getId

        private string reverseString(string s) {
            char[] chars = s.ToCharArray();
            char [] reversedChars = new char[chars.Length];
            for (int i = 0; i < chars.Length; i++)
                reversedChars[reversedChars.Length - 1 - i] = chars[i];
            string rs = new string(reversedChars);
            return rs;
        } //reverseString
...
```

Demo

- Demo in classroom

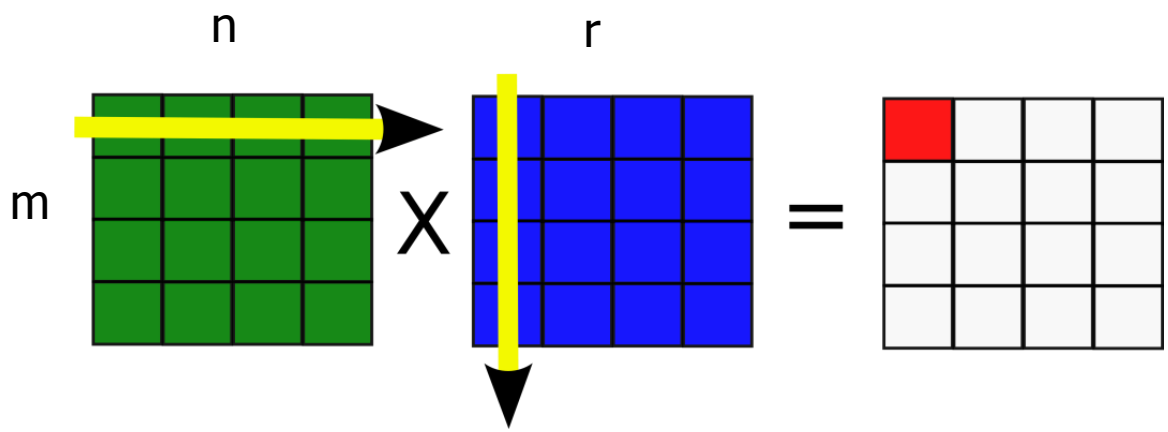
- Any Questions ?

Summary

- Reversing a string, using input and output queues
- Matrix multiplication (more workers)

Matrix Multiplication

- Example: matrix multiplication
  - Matrix1's dimension: m*n
  - Matrix2's dimension: n*r
  - (Matrix1 * Matrix2) dimension: m*r



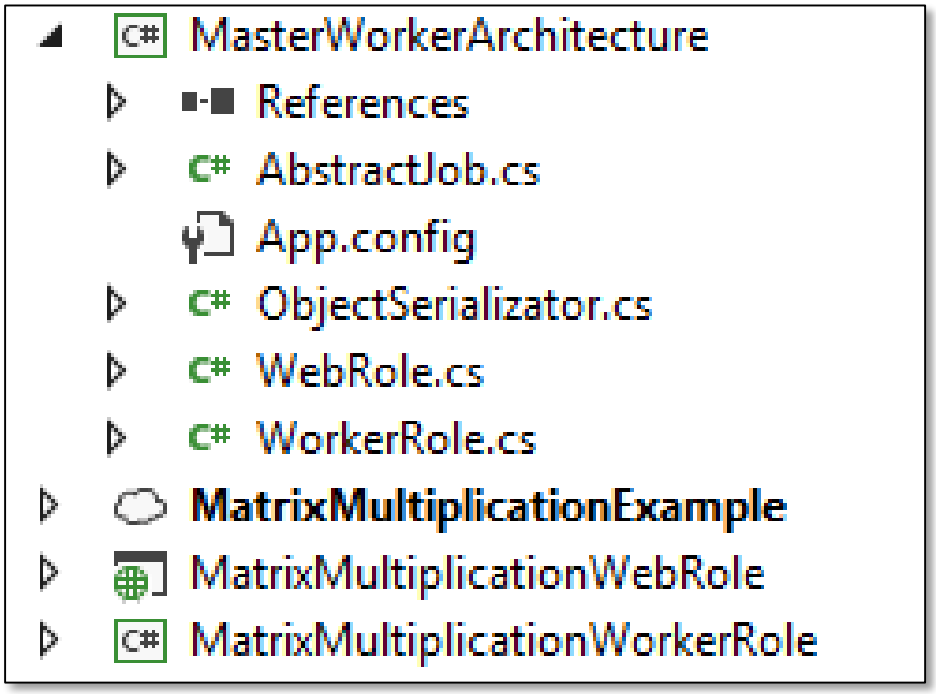
- Each (row-col)-multiplication can be assigned to a WorkerRole

Matrix Multiplication

- MasterWorker-Architecture
  - Define the infrastucture (WebRole+WorkerRole), an abstract type for the job and serialization class
- MatrixMultiplication
  - The MatrixMultiplication and ScalarProduct classes, that extend the abstract job class
- WebRole for the matrix multiplication
- WorkerRole for the matrix multiplication

MatrixMultiplication Example in MS Azure

- MasterWorker-Architecture
  - Create a new empty C# project, called MasterWorkerArchitecture
  - Add AbstractJob.cs
  - Add ObjectSerializator.cs
  - Add WebRole.cs
  - Add WorkerRole.cs



AbstractJob (1/2)

```
namespace AzureArchitecture
{
    [Serializable]
    public abstract class AbstractJob : IComparable<AbstractJob> {
        protected List<object> parameters;          // input elements
        protected List<object> result;              // output elements
        private int id;                             // id; it need to sort the sub-jobs
        protected List<AbstractJob> subJobs;

        public AbstractJob() {
            parameters = new List<object>();
            result = new List<object>();
            subJobs = new List<AbstractJob>();
        }
        public void addParamater(object o) {
            parameters.Add(o);
        }
        public List<object> getResult() {
            return result;
        }
        public void setId(int i) {
            id = i;
        }
…
```

In c#, the easiest way to make a class serializable is to mark it with the "Serializable" attribute.

"Icomparable" interface, needs to implement the "compareTo(…)" method

AbstractJob (2/2)

```
…

        public int getId() {
            return id;
        }

        public int CompareTo(AbstractJob j) {
            return id - j.id;
        }

        public abstract void splitMainJob();

        public abstract void mergeJobResults();

        public List<AbstractJob> getSubJobList() {
            return subJobs;
        }

        public void setSubJobList( List<AbstractJob> l ) {
            subJobs = l;
        }

        public abstract void run();
    }
}
```

ObjectSerializator

```
namespace AzureArchitecture
{
    public class ObjectSerializator
    {
        public byte[] serialize(object j) {
            MemoryStream m = new MemoryStream();
            BinaryFormatter bf = new BinaryFormatter();
            bf.Serialize(m, j);
            byte[] b = m.ToArray();
            m.Close();
            return b;
        }

        public object deserialize(byte[] b) {
            MemoryStream m = new MemoryStream(b);
            BinaryFormatter bf = new BinaryFormatter();
            object o = bf.Deserialize(m);
            m.Close();
            return o;
        }

    }
}
```

AzureArchitecture WebRole (1/4)

```
namespace AzureArchitecture
{
    public class WebRole
    {
        CloudQueue inputQueue; //WebRole-->WorkerRole
        CloudQueue outputQueue;//WorkerRole-->WebRole
        ObjectSerializator os;

        AbstractJob mainJob;

        protected List<AbstractJob> jobs;           //jobs to be executed
        protected List<AbstractJob> jobsExecuted;   //jobs executed
        int id;

        public WebRole()
        {
            os = new ObjectSerializator();
            initializeQueue();
            jobs = new List<AbstractJob>();
            jobsExecuted = new List<AbstractJob>();
            id = 0;
        }

        public void setMainJob(AbstractJob j)
        {
            mainJob = j;
        }
…
```

AzureArchitecture WebRole (2/4)

```
…
        private void initializeQueue() {
            CloudStorageAccount storageAccount =
                           CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("StorageConnectionString"));
            // Create the queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
            // Retrieve a reference to input and output queues
            inputQueue = queueClient.GetQueueReference("input");
            inputQueue.CreateIfNotExists();
            outputQueue = queueClient.GetQueueReference("output");
            outputQueue.CreateIfNotExists();
        }


        private void submitJob(AbstractJob j)
        {   // submit a job in the input queue ( webRole --> workerRole )
            inputQueue.AddMessage(new CloudQueueMessage(os.serialize(j)));
            Trace.WriteLine("job numero " + j.getId() + " inserito nella coda di input");
        }

        private AbstractJob getJob()
        {    //get a resolved job by the output queue ( workerRole --> webRole )
            CloudQueueMessage cqm = outputQueue.GetMessage();
            if (cqm == null) return null;
            outputQueue.DeleteMessage(cqm);
            AbstractJob j = (AbstractJob)os.deserialize(cqm.AsBytes);
            Trace.WriteLine("job numero " + j.getId() + " prelevato dalla coda di output");
            return j;
        }
…
```

AzureArchitecture WebRole (3/4)

```
…
        public void addSubJob(AbstractJob j)
        {   // add a new unresolved job
            j.setId(id); jobs.Add(j); id++;
        }

        public List<AbstractJob> getResolvedJobs()
        {
            return jobsExecuted;
        }

        public void run()
        {   // submit all the sub-jobs to the workers
            // get all the sub-jobs resolved by the workers
            mainJob.splitMainJob();

            //create the 'jobs' list
            foreach (AbstractJob j in mainJob.getSubJobList())
                addSubJob(j);               // per mantere l'ordine (id)

            //add the jobs to the 'inputQueue'
            foreach (AbstractJob j in jobs)
                submitJob(j);

            // job inviati ai worker...
…
```

AzureArchitecture WebRole (4/4)

```
…

            // job inviati ai worker...

            //get the executed jobs from the 'outputQueue'
            //and add them to the 'j'
            while (jobsExecuted.Count < jobs.Count) {
                AbstractJob e = getJob();
                if (e != null)
                    jobsExecuted.Add(e);
            }//while

            jobsExecuted.Sort();
            mainJob.setSubJobList(jobsExecuted);
            mainJob.mergeJobResults();
        }

        public AbstractJob getMainJob(){
             return mainJob;
        }

    }
}
```

AzureArchitecture WorkerRole (1/3)

```
namespace AzureArchitecture
{
    public class WorkerRole
    {
        CloudQueue inputQueue;
        CloudQueue outputQueue;
        ObjectSerializator os;

        public WorkerRole() {
            os = new ObjectSerializator();
            initializeQueue();
        }//constructor

        public void run() {
            // get an unresolved job and resolve it
            // submit the resolved job to the webRole
            while (true) {
                AbstractJob j = getJob();
                if (j != null) {
                    j.run();
                    returnResolvedJob(j);
                }//if
            }//while
        }//run
…
```

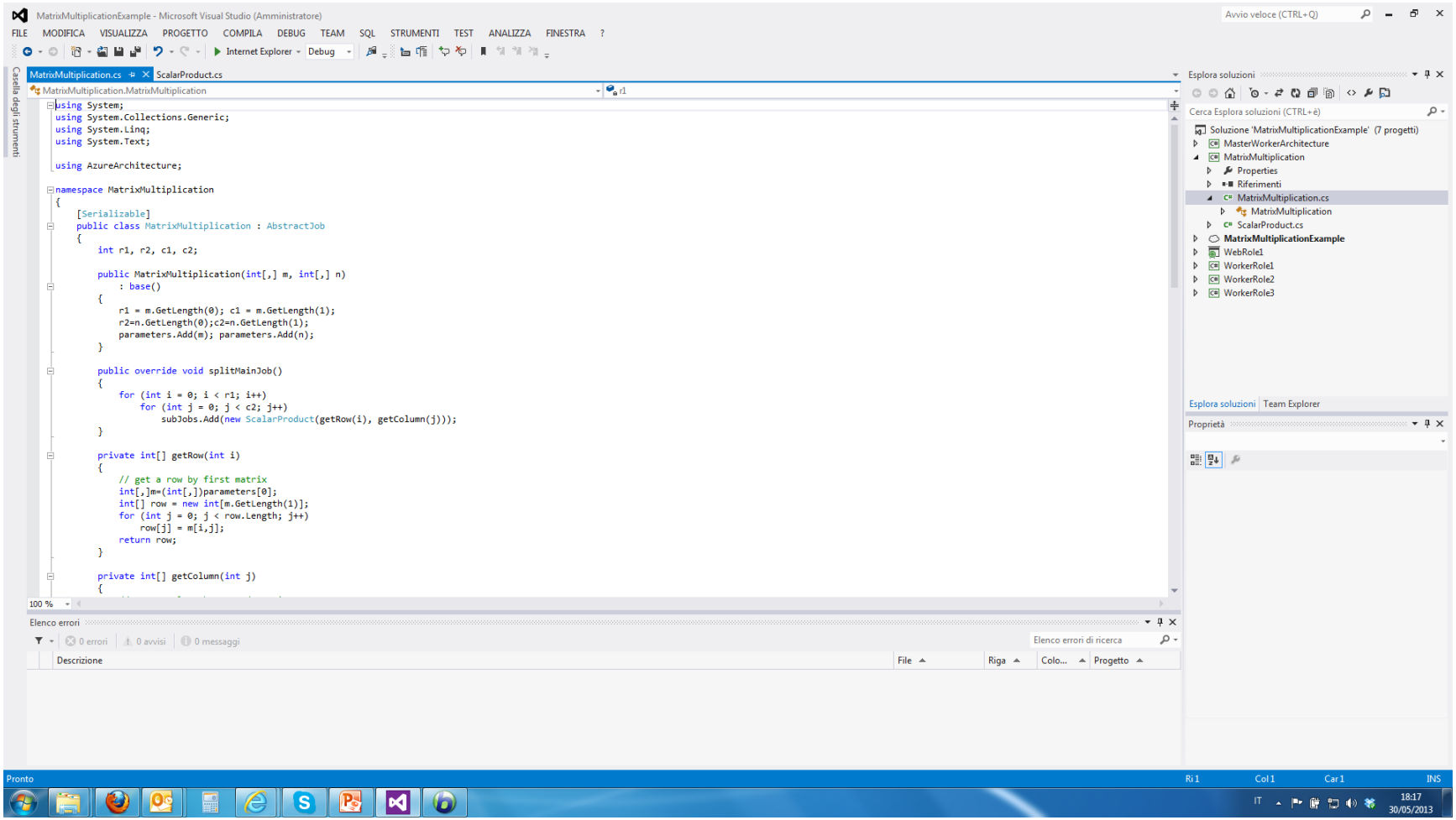
WorkerRole (2/3)

```
…
        private void initializeQueue() {
           CloudStorageAccount storageAccount =

           CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("StorageConnectionString"));
            // Create the queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
            // Retrieve a reference to input and output queues
            inputQueue = queueClient.GetQueueReference("input");
            inputQueue.CreateIfNotExists();
            outputQueue = queueClient.GetQueueReference("output");
            outputQueue.CreateIfNotExists();
        }//initializeQueue

        private void returnResolvedJob(AbstractJob j) {
            // submit a resolved job in the output queue ( workerRole --> webRole )
            outputQueue.AddMessage(new CloudQueueMessage(os.serialize(j)));
            Trace.WriteLine("job numero " + j.getId() + " inserito nella coda di output");
        }//returnResolvedJob
…
```

WorkerRole (3/3)

```
…
        private AbstractJob getJob() {
            //get an unresolved job by the input queue ( webRole --> workerRole )
            CloudQueueMessage cqm = inputQueue.GetMessage();
            if (cqm == null)
                return null;
            inputQueue.DeleteMessage(cqm);
            AbstractJob j = (AbstractJob)os.deserialize(cqm.AsBytes);
            Trace.WriteLine("job numero " + j.getId() + " prelevato dalla coda di input");
            return j;
        }//getJob

    }
}
```

MatrixMultiplication



ScalarProduct

```
namespace MatrixMultiplication
{
    [Serializable]
    class ScalarProduct : AbstractJob
    {
        public ScalarProduct(int[]m,int[] n)
            : base()
        {
            parameters.Add(m);  parameters.Add(n);
        }

        public override void splitMainJob() {}

        public override void mergeJobResults() {}

        public override void run()
        {
            int res = 0;
            int[]x=(int[])parameters[0],y=(int[])parameters[1];
            for (int i = 0; i < x.Length; i++)
                res += x[i] * y[i];
            result.Add(res);
        }
    }
}
```

MatrixMultiplication

```
namespace MatrixMultiplication
{
    [Serializable]
    public class MatrixMultiplication : AbstractJob
    {
        int r1, r2, c1, c2;

        public MatrixMultiplication(int[,] m, int[,] n)
            : base()
        {
            r1 = m.GetLength(0); c1 = m.GetLength(1);
            r2=n.GetLength(0);c2=n.GetLength(1);
            parameters.Add(m); parameters.Add(n);
        }

        public override void splitMainJob()
        {
            for (int i = 0; i < r1; i++)
                for (int j = 0; j < c2; j++)
                    subJobs.Add(new ScalarProduct(getRow(i), getColumn(j)));
        }
…
```

MatrixMultiplication

```
…
        private int[] getRow(int i)
        {
            // get the 'i-th' row of the first matrix
            int[,]m=(int[,])parameters[0];
            int[] row = new int[m.GetLength(1)];
            for (int j = 0; j < row.Length; j++)
                row[j] = m[i,j];
            return row;
        }

        private int[] getColumn(int j)
        {
            // get the 'j-th' column of the second matrix
            int[,] n = (int[,])parameters[1];
            int[] column = new int[n.GetLength(0)];
            for (int i = 0; i < column.Length; i++)
                column[i] = n[i,j];
            return column;
        }

        public override void mergeJobResults()
        {
            int[,] res = new int[r1, c2];
            for (int i = 0; i < r1; i++)
                for (int j = 0; j < c2; j++)
                    res[i, j] = (int)subJobs[i * c2 + j].getResult()[0];
            result.Add(res);
        }
…
```

MatrixMultiplication

```
…

        public override void run()
        {
            int[,] res = new int[r1, c2];
            int scalarProduct;
            for (int i = 0; i < r1; i++)
                for (int j = 0; j < c2; j++) {
                    scalarProduct=0;
                    for( int k=0;k<c1;k++ ) {
                        int[,]m=(int[,])parameters[0],n=(int[,])parameters[1];
                        scalarProduct += m[i, k] * n[k, j];
                    }
                    res[i, j] = scalarProduct;
                }
            result.Add(res);
        }
    }
}
```

WorkerRole

```
namespace WorkerRole
{
    public class WorkerRole : RoleEntryPoint
    {
        public override void Run()
        {   // Implementazione di lavoro di esempio.
            //Sostituire con la logica personalizzata.
            Trace.WriteLine("WorkerRole entry point called", "Information");
            new AzureArchitecture.WorkerRole().run();
        }

        public override bool OnStart()
        {
            // Impostare il numero massimo di connessioni simultanee
            ServicePointManager.DefaultConnectionLimit = 12;

            return base.OnStart();
        }
    }
}
```

Default.aspx

```
…
<div class="col-md-12">
        <asp:Label ID="LabelA" runat="server" Text="MatrixA"></asp:Label>
        <asp:TextBox ID="TextBoxA" TextMode="multiline" runat="server"
         Rows="5"></asp:TextBox>

        <asp:Label ID="LabelB" runat="server" Text="MatrixB"></asp:Label>
        <asp:TextBox ID="TextBoxB" TextMode="multiline" runat="server"
         Rows="5"></asp:TextBox>
</div>

<div class="col-md-12"><br />
        <asp:Label ID="Label4" runat="server" Text="Result:"></asp:Label>
        <asp:TextBox ID="TextBoxJobResult" TextMode="multiline"
         runat="server" Rows="5"></asp:TextBox>

         <asp:Button ID="ButtonMultiplicate" runat="server" Text="Run"
         OnClick="btnSend_Click"/>
         <br /><br />
</div>

…
```

Default.aspx

Default.aspx.cs

```
namespace MatrixMultiplicationWebRole
{
    public partial class _Default : Page
    {

        AzureArchitecture.WebRole web;
        int[,] matrixA, matrixB;

        protected void Page_Load(object sender, EventArgs e)
        {
            //Create a new WebRole Object (Azure Architecure class)
            web = new AzureArchitecture.WebRole();

            //Define a matrix
            matrixA = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };
            matrixB = new int[,] { { 1, 2, 3 }, { 3, 4, 5 } };

            //Display matrices on page
            TextBoxA.Text = getMatrixString(matrixA);
            TextBoxB.Text = getMatrixString(matrixB);

        }
…
```

Default.aspx.cs

```
…

        protected void btnSend_Click(object sender, EventArgs e)
        {
            //Create main job
            MatrixMultiplication.MatrixMultiplication job = new
MatrixMultiplication.MatrixMultiplication(matrixA, matrixB);
            web.setMainJob(job);

            //Run main job
            web.run();

            //Get and diplay main job result
            int[,] res = (int[,])web.getMainJob().getResult().ElementAt(0);
            TextBoxJobResult.Text = getMatrixString(res);
        }
…
```

Default.aspx.cs

```
…
 public string getMatrixString(int[,] res)
        {

            int rows = res.GetLength(0);
            int columns = res.GetLength(1);

            String result = "";
            for (int i = 0; i < rows; i++)
            {
                result = result + "[";
                for (int j = 0; j < columns; j++)
                    result = result + " " + string.Concat(res[i, j]);
                result = result + "]\r\n";
            }
            return result;
        }
    }
}
```

Demo

Demo in classroom