



Cloud Computing in Banking

COBOL replatforming and Virtualization case study

Presenter: **Raffaele Sgherri**
Solutions Manager, Avanade Italy

Date: May 21, 2008

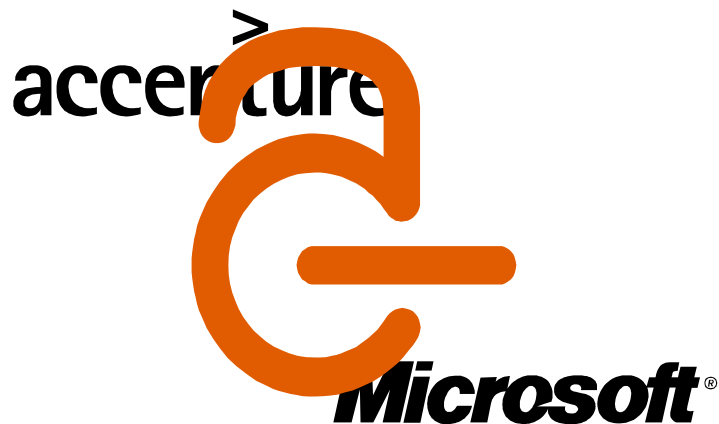
Agenda

- ▶ Who we are
 - Our offering in Grid space
 - Our experience in COBOL replatforming
- ▶ Case Study
 - COBOL replatforming test case
 - Computing node Virtualization
- ▶ Avanade Italy progress as EGEE EBA

Who we are

Offering on Grid space and COBOL replatforming

Our beginning



In 2000 Accenture and Microsoft came together to create Avanade.

A joint venture designed to meet the growing demand for Microsoft-based services among enterprises worldwide.

Eight years of growth and global expansion

Delivering value in 22 countries



More than

- ▶ 8,000 professionals
- ▶ 3,700 in India, Philippines
- ▶ \$593m revenue in FY07
- ▶ 2,900 customers

A diverse, global customer base

- ▶ *Forbes* Global 2000
- ▶ *Fortune* Global 1000
- ▶ All industries
- ▶ Government agencies



Wm. **WRIGLEY** Jr. Company



London
STOCK EXCHANGE



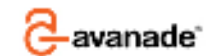
THE UNIVERSITY OF TEXAS
MD ANDERSON
CANCER CENTER



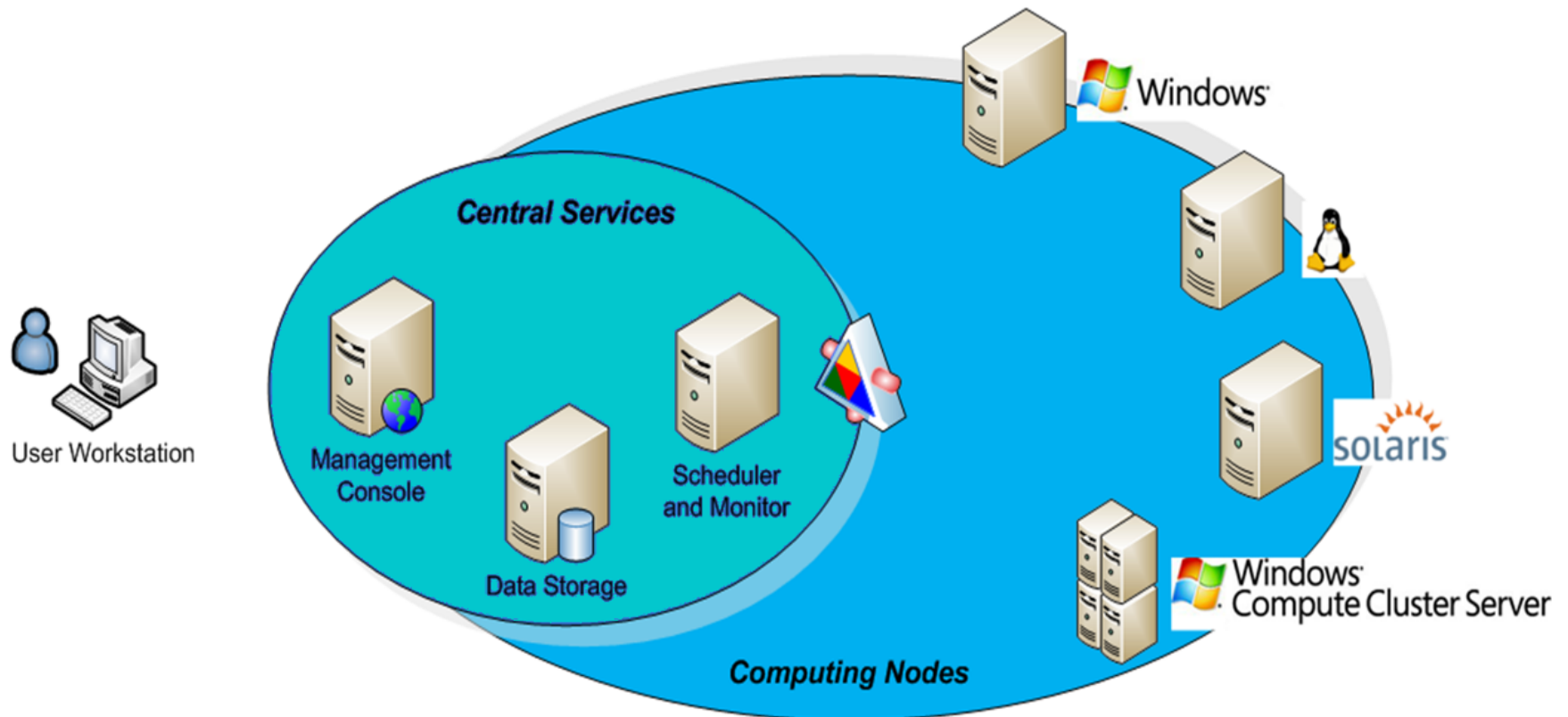
Chicago
Federal Home Loan Bank



Technology that liberates



AGA: Avanade Grid Architecture



AGA Mainframe

- ▶ Is a solution that enables companies to migrate large-scale, enterprise-wide applications from legacy environments to Microsoft .NET web environment, leveraging AGA to manage scalability and computing needs.
- ▶ Has been developed by evolving a set of tools previously created for a major Italian re-platforming project for the Italian Social Security.
 - (see Gartner report G00146104 published 28 March 2007).

AGA Mainframe: Key points

- ▶ TCO Reduction
- ▶ Legacy SW & HW dismissing (AS400 and Mainframe)
- ▶ Applications transformation into native .NET components (no inefficient wrapping or re-hosting)
- ▶ Smooth transition toward the web-based model by preserving applications logic and behavior
- ▶ Customer's developers workforce skills reuse
- ▶ Leverages AGA Grid technology
- ▶ Existing happy customers (INPS & a major Italian Bank)

The Solution: Component Inventory

Grid Infrastructure

- AGA .NET
- Custom Grid Tasks for integration with run-time migration architecture

Run-time architecture

- Batch Architecture

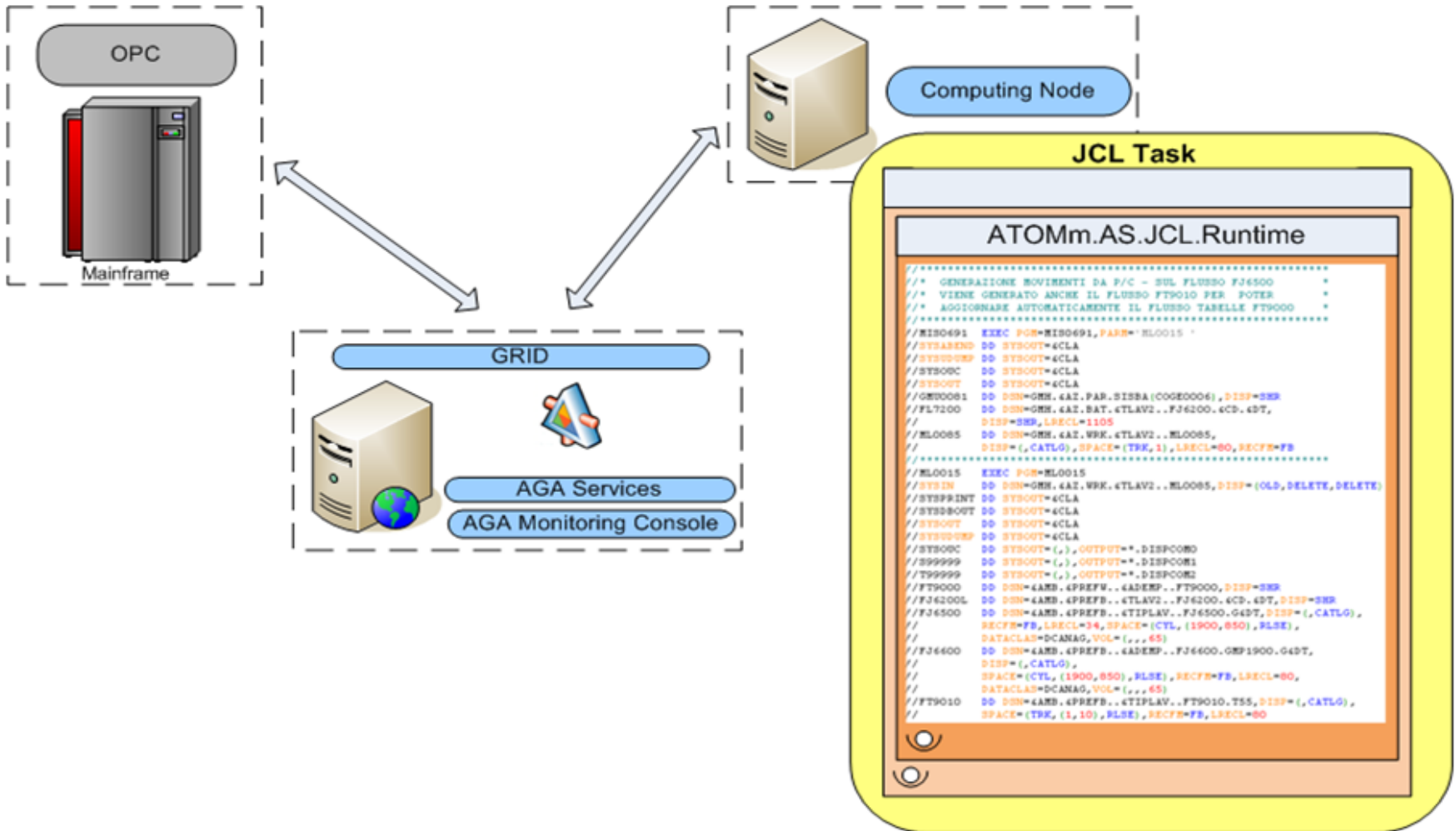
Migration Tools

- Code Migration
- Data Migration
- Solution Builder

Monitoring Tools

- OPC
- AGA Monitor
- AGA Mainframe Console

AGA Mainframe: Execution Architecture



Case Study

COBOL replatforming and Virtualization case study

AGA reference installation summary data

- ▶ Production grid in a major Italian bank
 - 250+ Windows machines
 - 80+ Linux machines
 - Adding 60 more, dedicated to Finance business unit
 - 1100+ CPUs available
 - 20.000+ Tasks executed each day
- ▶ Running mission critical applications
 - Basel II compliancy (VaR computation)
 - IAS 32 / IAS 39 instrument categorization
- ▶ Using idle workstations at night and recovery backup machines, otherwise not used

Test case COBOL procedure (1/2)

- ▶ AGA Mainframe was used to migrate a custom software package that creates detailed and aggregated reports about daily bank activities
- ▶ The package includes 4 different steps
 - 1: Extract data for each single bank customer
 - 2: Correlate user data with managing office
 - 3. Aggregate data for each managing office
 - 4. Create user reports
- ▶ We migrated the second step to let the customer test the AGA Mainframe package

Test case COBOL procedure (2/2)

Source Code

- ▶ 1 JCL (33 LoC)
- ▶ 1 JCL procedure (46 LoC)
- ▶ 3 include (14 LoC)
- ▶ 95 COBOL programs (46.299 LoC)
- ▶ 2 copy (39 LoC)
- ▶ 3 assembler programs (1.490 LoC)

Data Files

- ▶ Input (150 Gb.)
 - 1 sequential file
 - 1 GDG with 30 different data formats
 - 1 VSAM
- ▶ Output (450 Gb.)
 - 3 sequential files
 - 2 display files (including SYSOUT)

Application flow

CFG

- Database connection string & AGA Computing Node setup
- Deployment of assemblies to be executed

IN

- Data transfer through FTP
- Data format conversion (e.g. EBCDIC -> ANSI)

EXEC

- Converted COBOL assembly execution
- Each Task include all steps from original JCL
- 30 Tasks executed at the same time

OUT

- Data format conversion (e.g. ANSI -> EBCDIC)
- Data transfer trough FTP
- Clean up

Performance results

- ▶ On Bank Mainframe, the program required about 2 hours of CPU time
 - (elapsed time vary from 3 to 10 hours)
- ▶ Using 30 concurrent JCL execution Tasks, on AGA the program requires 3 hours
 - (elapsed time is 3 hours)
 - FTP transfer time is about 8 minutes, included
 - Runs on 12 non dedicated CPUs (about 2 GHz. each)
 - 4 machines with 2 CPUs quadcore
 - 2 machines with 2 CPUs dualcore

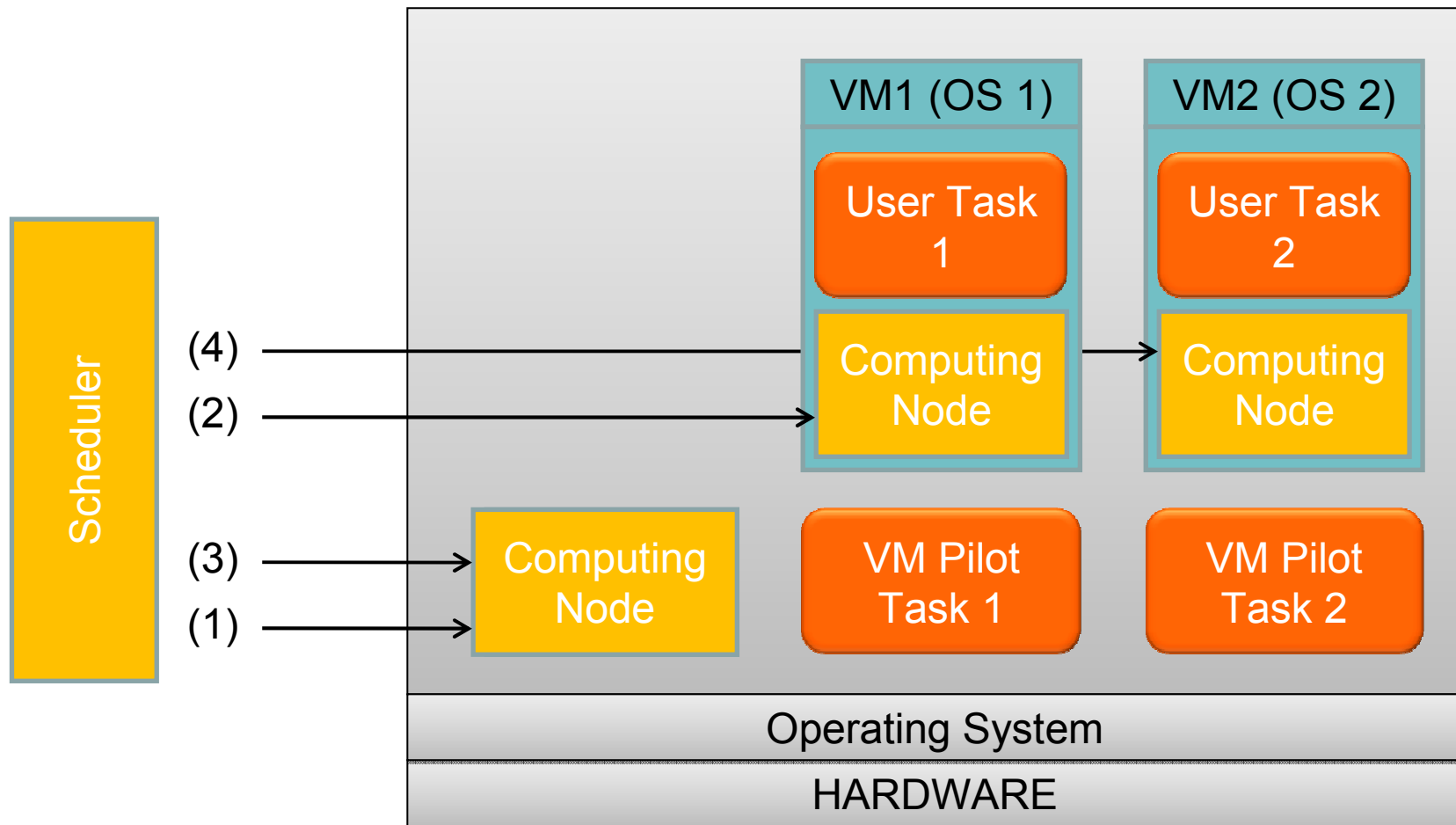
Virtualization

- ▶ Set of complete AGA Computing Node VMs with different setups
 - Available packages and package versions
 - Different Operating Systems and versions

- ▶ Management tool set
 - Easy clone of available virtual machines
 - Patch management of master virtual machines

- ▶ Task execution on Virtualized Computing Nodes
 - 1. Pilot AGA Task wake up the required VM on the physical machine
 - 2: Job execution AGA Task gets allocated on the VM machine
 - 3. Each VM is reverted to initial status

Task execution on Virtualized Computing Nodes



Virtualization Benefits

- ▶ One set of physical machines can change behavior and serve multiple Tasks
 - Different Operating Systems
 - Different packages versions
- ▶ Freedom of choice
 - Select the best Operating System for each Task
 - Availability on demand
- ▶ Low impact on performance
 - Para-virtualization: Xen, Hyper-V

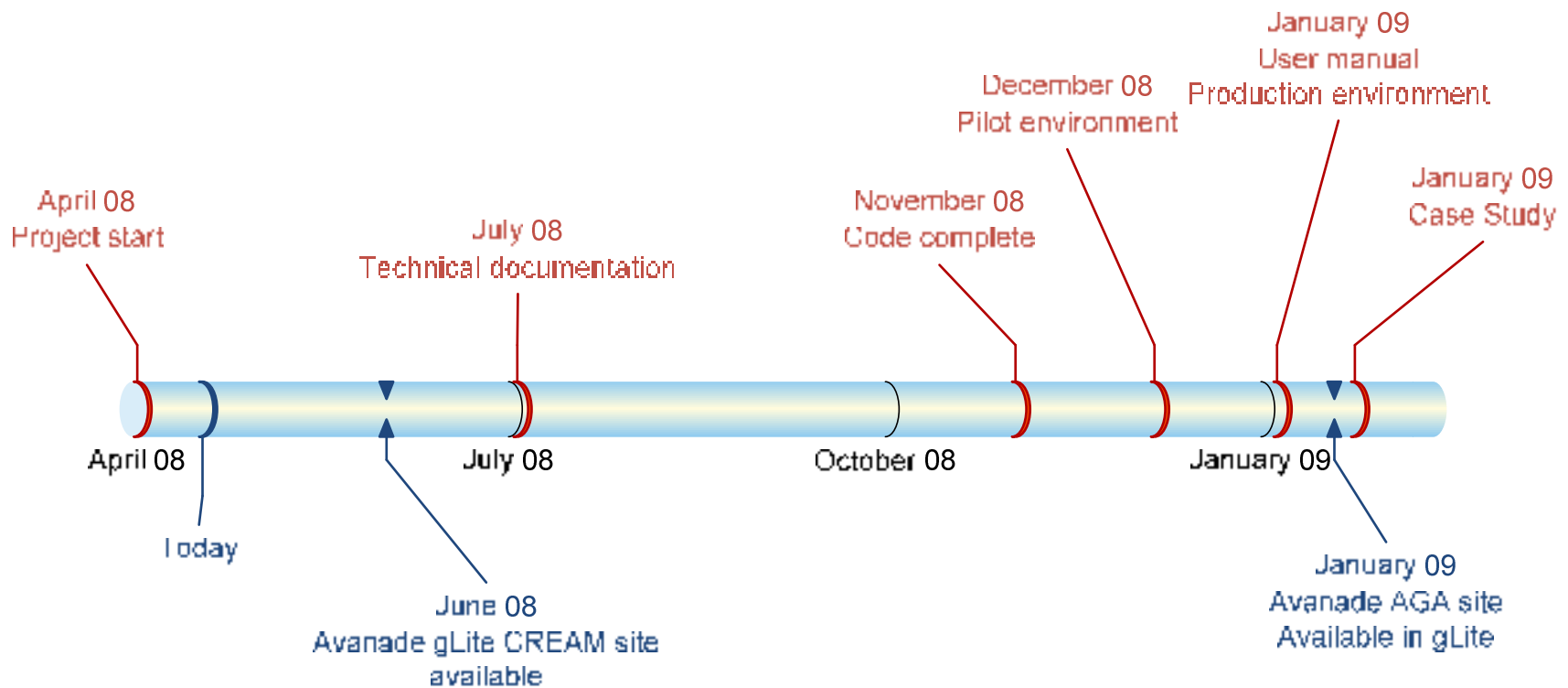
Avanade Italy as EGEE EBA

Activities and goal

Activities as EGEE EBA

- ▶ Implement a compatibility layer for gLite
 - Manage AGA Computing Nodes as gLite CREAM CNs
 - Use gLite CNs as part of AGA installations
- ▶ Setup and run a gLite site
 - Let some Avanade CPUs be part of gLite environment
 - Manage an AGA site as part of gLite
- ▶ Participate to EGEE events
 - Share Avanade experiences and results
 - Sponsor the Grid adoption in enterprises

EGEE EBA activity plan



Questions?

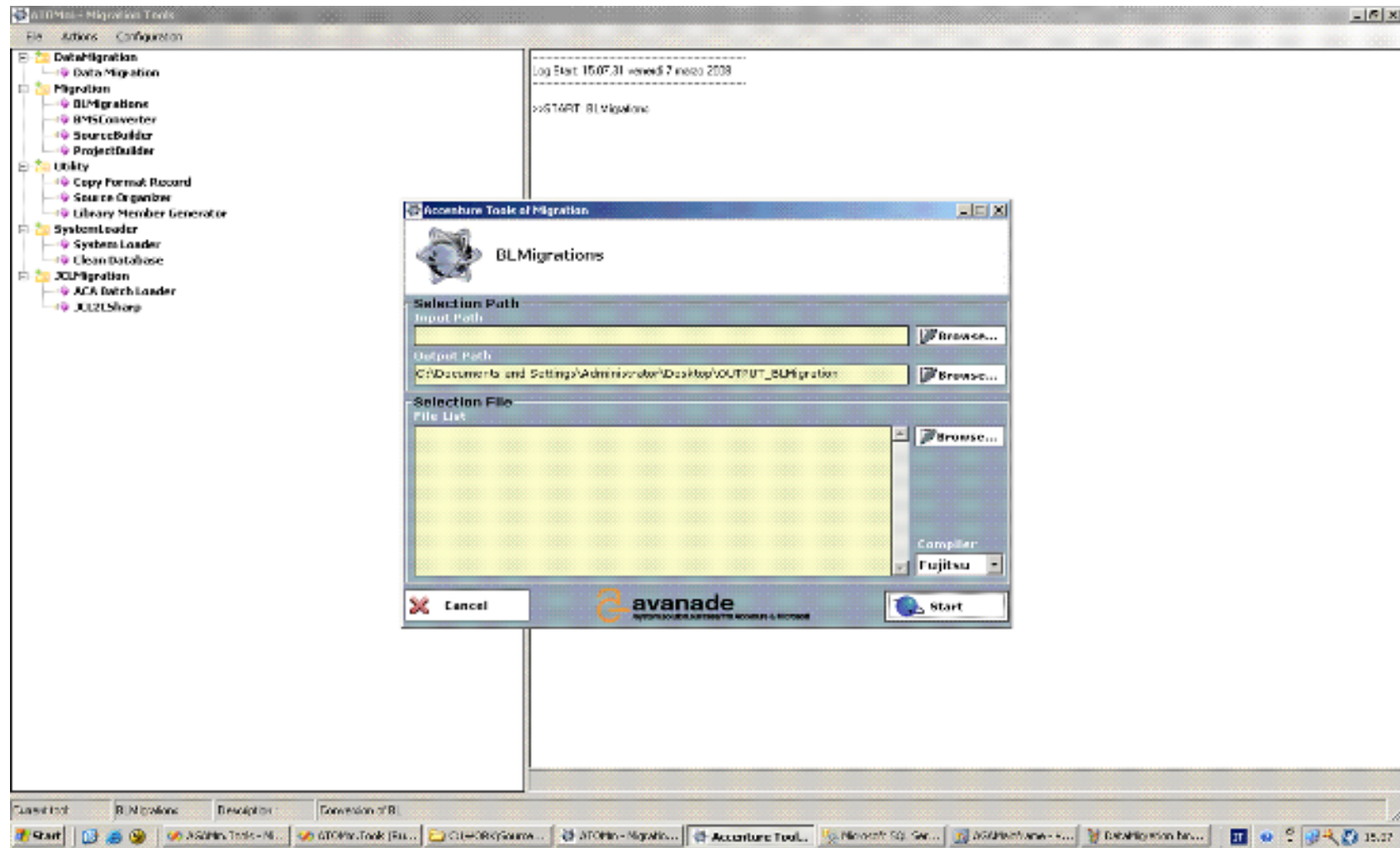


- ▶ For any further information
 - Raffaele Sgherri (raffaele.sgherri@avanade.com)
AGA Technical Lead, EGEE EBA primary contact
 - Roberto Cappuccio (roberto.cappuccio@avanade.com)
Director of Florence office

APPENDIX

Migration Tools & Samples

Code Migration: BLMigration



Code Migration: BLMigration

Source Code

```
* ENVIROMENT DIVISION statement
SPECIAL-NAMES.
    DECIMAL-POINT IS COMMA.

* FILE CONTROL statement
SELECT FJ6500 ASSIGN TO FJ6500
    FILE STATUS FLS.

* Data Types
01 FJ6500-110-REC.
03 FJ6500-110-KEY-SORT.
05 FJ6500-110-C-RESP PIC 9(6) COMP-3.
03 FJ6500-110-VALORE PIC S9(18) COMP.

* CALL statement
CALL "RXX555" USING RXX555-CAMPI.

* WRITE statement
WRITE FJ6500-010-REC.
```

Migrated code

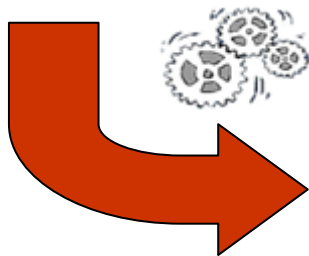
```
* ENVIROMENT DIVISION statement
AT02A OBJECT-COMPUTER.
AT02A    PC PROGRAM COLLATING SEQUENCE IS TRUE-EBCDIC.
        SPECIAL-NAMES.
        DECIMAL-POINT IS COMMA
        ALPHABET TRUE-EBCDIC COPY COPY_ATOM_EBCDIC.

* FILE CONTROL statement
AT07A    SELECT FJ6500 ASSIGN TO FJ6500_ATOM-TOOL.
AT07B*   SELECT FJ6500 ASSIGN TO FJ6500 FILE STATUS FLS.

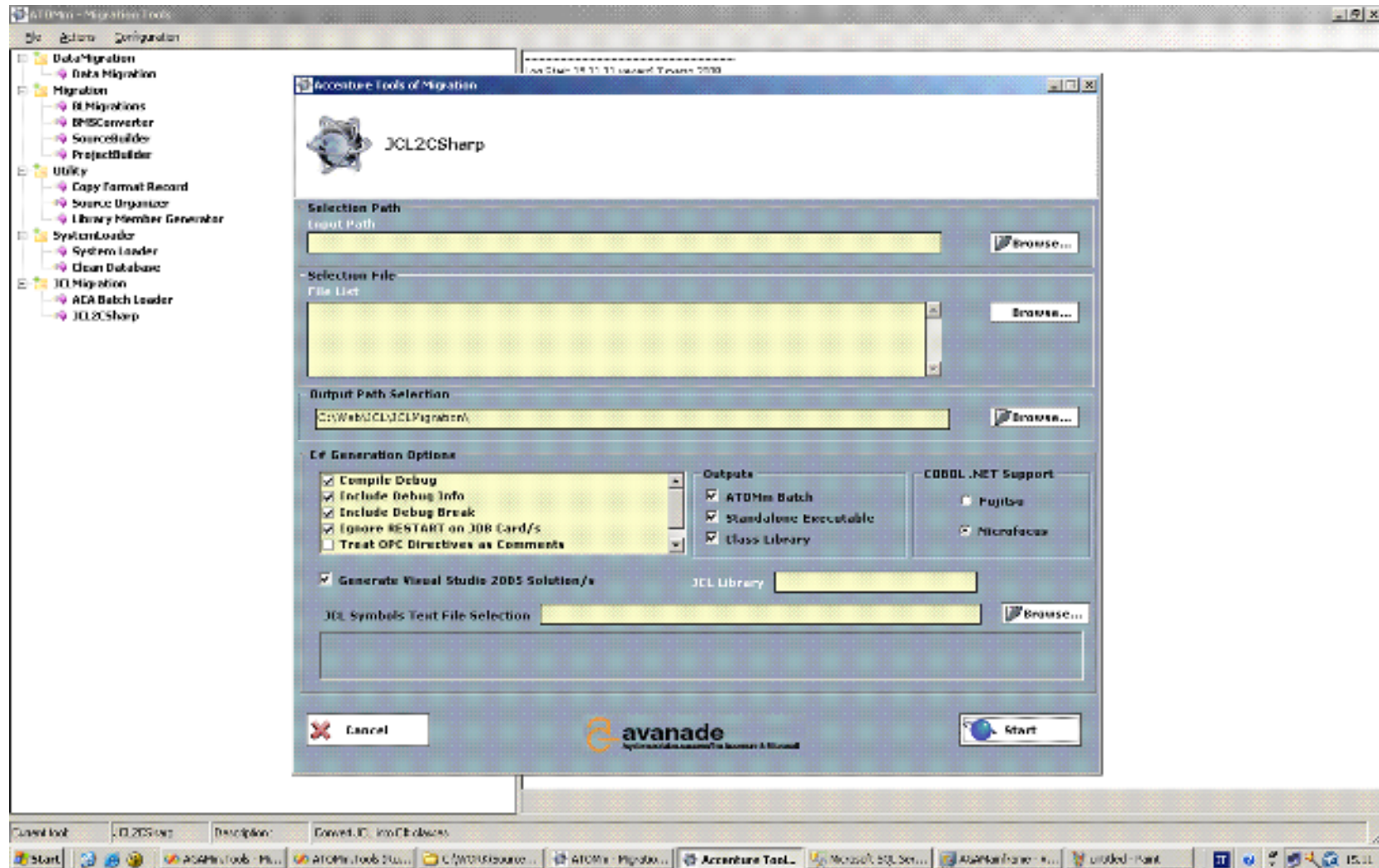
* Data Types
01 FJ6500-110-REC.
03 FJ6500-110-KEY-SORT.
05 FJ6500-110-C-RESP          PIC 9(6)    COMP-3.
AT08B*   03 FJ6500-110-VALORE          PIC S9(18) COMP.
AT08A    03 FJ6500-110-VALORE          PIC S9(18) BINARY.

* CALL statement
AT01B*   CALL "RXX555" USING RXX555-CAMPI.
AT01A    MOVE "RXX555" TO WS-ATOM-CALL
AT01A    INVOKE OBJ-WS-CUR-LNGENV "ResolveCall" USING WS-ATOM-CALL
        RETURNING WS-ATOM-CALL
AT01A    CALL WS-ATOM-CALL USING RXX555-CAMPI

* WRITE statement
AT13B*   WRITE FJ6500-010-REC.
AT13A    SET PROP-RECORDFILEFJ OF OBJ-FILE-FJ6500 TO FJ6500-010-REC OF FJ6500
AT13A    INVOKE OBJ-FILE-FJ6500 "WriteRecord" RETURNING WS-FS-ATOM-RETCODE
AT13A    MOVE WS-FS-ATOM-RETCODE TO FLS
AT13A    EVALUATE TRUE
AT13A        WHEN PROP-ERRORIO OF OBJ-FILE-FJ6500 EQUAL B"1"
AT13A        PERFORM DISPL-ERR-FJ6500 OF ERR-FILE-FJ6500
AT13A    END-EVALUATE.
```



Code Migration: JCL2CSharp



Code Migration: JCL2CSharp

Source Code

```
//*****  
//ML0015 EXEC PGM=ML0015  
//SYSIN DD DSN=GMH.&AZ.WRK.&TLAV2..ML0085,  
// DISP=(OLD,DELETE,DELETE)  
//SYSPRINT DD SYSOUT=&CLA  
//SYSDBOUT DD SYSOUT=&CLA  
//SYSOUT DD SYSOUT=&CLA  
//SYSUDUMP DD SYSOUT=&CLA  
//SYSOUC DD SYSOUT=(,),OUTPUT=*.DISPCOM3  
//S99999 DD SYSOUT=(,),OUTPUT=*.DISPCOM1  
//T99999 DD SYSOUT=(,),OUTPUT=*.DISPCOM2  
//FT9000 DD DSN=&AMB.&PREFW..&ADEMP..FT9000,DISP=SHR  
//FJ6200L DD DSN=&AMB.&PREFB..&TLAV2..FJ6200.&CD.&DT,  
// DISP=SHR  
//FJ6500 DD DSN=&AMB.&PREFB..&TIPLAV..FJ6500.G&DT,  
// DISP=(,CATLG),  
// RECFM=FB,LRECL=34,SPACE=(CYL,(1900,850),RLSE),  
// DATACLAS=DCANAG,VOL=(,,65)  
//FJ6600 DD DSN=&AMB.&PREFB..&ADEMP..FJ6600.GMP1900.G&DT,  
// DISP=(,CATLG),  
// SPACE=(CYL,(1900,850),RLSE),RECFM=FB,LRECL=80,  
// DATACLAS=DCANAG,VOL=(,,65)  
//FT9010 DD DSN=&AMB.&PREFB..&TIPLAV..FT9010.T55,  
// DISP=(,CATLG),  
// SPACE=(TRK,(1,10),RLSE),RECFM=FB,LRECL=80
```

Migrated code

```
public class GMP1900 : JCLProcedure  
{  
    public ML0015 ML0015;  
    private void InternalExecute()  
    {  
        if (base.Evaluate("ABEND = false"))  
        {  
            // //CE010AB EXEC  
            //PGM=CE#S010,PARM='&AMB.&PREFB..&TIPLAV..FJ6500.G&DT..&GDGVER'  
            CE010AB.Execute(  
                base.ResolveSymbols("'&AMB.&PREFB..&TIPLAV..FJ6500.G&DT..&GDGVER'"));  
        }  
    }  
}  
  
public class ML0015 : JCLStep  
{  
    public ML0015(JCLObject parent) : base(parent)  
    {  
        base.Name = "ML0015";  
    }  
    protected override void Run()  
    {  
        try  
        {  
            base.BeginTransactionSession("ML0015", "ML0015");  
  
            //FT9000 DD DSN=&AMB.&PREFW..DIP.&ADEMP..FT9000,DISP=SHR  
            ZosDataSet varDsL20 = ZosDataSetFactory.CreateZosDataSet(  
                base.ResolveSymbolsForDataSetName(  
                    "&AMB.&PREFW..DIP.&ADEMP..FT9000"), "NO", "SHR", "END", "", null);  
            base.AddDataSet("FT9000", varDsL20);  
  
            base.PerformInitialOperations();  
  
            base.RC = base.ExecuteProgram();  
        }  
    }  
}
```

