Totem Unified Database Access Service

System Prototype

Introduction
This prototype is introduced in order to familiarize with main concepts of using TUDAS system. We provide server application which is constantly working on pc1otem34 machine, ready to handle client requests, as well as client-side libraries with simple examples written in all three supported languages (java, c++, python). In this document we would like to guide you through these client-side prototypes, showing how to install and run them.
Detailed description of prototype interfaces can be found in document:

Installing TUDAS
All you need to do is to checkout project from svn tag:

```
svn co svn+ssh://svn.cern.ch/reps/totem/trunk/tudas
```

Downloaded package includes all source code of our project. Essential part containing ready to use libraries and examples is located in directory:

```
./release/
```

Here you can find client-side modules grouped in language specific directories.

Running Java example
1. Open your terminal and navigate to java directory:

```
   cd  ./release/java
```

2. Compile ExampleClient.java attaching tudas.jar library:

```
javac -cp ./lib/tudas.jar ExampleClient.java
```

3. Run compiled class:

```
java -cp ./;lib/tudas.jar ExampleClient
```

Running python example
1. Open your terminal and navigate to python directory:

```
   cd ./release/python
```

Authors: Bartłomiej Alberski, Michał Idzik, Bartosz Niemczura
2. Update PYTHONPATH variable:
   
   ```bash
   export PYTHONPATH=$PYTHONPATH;/lib/tudas;/lib/ice
   ```

3. Run example_client.py:
   
   ```bash
   python example_client.py
   ```

**Running c++ example**

NOTE: All TUDAS c++ libraries were compiled on Scientific Linux CERN 5. If you would like to use another distribution to test our prototype, please contact us.

1. Open your terminal and navigate to python directory:
   
   ```bash
   cd ./release/c++
   ```

2. Update LD_LIBRARY_PATH variable:
   
   ```bash
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$LD_LIBRARY_PATH:/lib
   ```

4. Compile example_client.cpp:
   
   ```bash
   g++ -I./include -L./lib -ltudas example_client.cpp -o example_client
   ```

5. Run example_client:
   
   ```bash
   ./example_client
   ```

**Using CMSSW module**

In order to compile CMSSW with access to database, one should do following steps:

1. Create new CMSSW project workspace:
   
   ```bash
   export RFIO_USE_CASTOR_V2=YES
   export STAGE_HOST=castorpublic
   export STAGE_SVCCLASS=default
   export SCRAM_ARCH=slc5_amd64_gcc434
   source /afs/cern.ch/cms/cmsset_default.sh scram project CMSSW CMSSW_4_2_4
   ```

2. Check out version of CMSSW with Tudas from SVN branch:
svn co
svn+ssh://svn.cern.ch/reps/totem/branches/CMS_4_2_4_with_tudas
CMSSW_4_2_4/src/

3. Create tool that will install needed external libraries:

```
cd CMSSW_4_2_4/config/toolbox/slc5_amd64_gcc434/tools/selected

Create following file:

<tool name="Ice" version="3.4.2">
  <lib name="Ice"/>
  <lib name="IceUtil"/>

  <client>
    <environment name="ICE_BASE" default="/afs/cern.ch/exp/totem/soft/
database/ice"/>
    <environment name="LIBDIR" default="$ICE_BASE/lib64"/>
    <environment name="INCLUDE" default="$ICE_BASE/include"/>
  </client>

  <runtime name="ICE_HOME" value="$ICE_BASE"/>
</tool>
```

4. Compile CMSSW:

```
cd ../../../src
eval `scram runtime -sh`
sccm setup ice
tsccm b -j 4
```

5. To use Tudas module in other modules, below line should be added info BuildFile.xml file:

```
<use name="TotemUnifiedDatabaseAccessService/Tudas"/>
```

Example of Tudas usage:

```
In TotemAlignment/RPDataFormats/plugin/TotemRPIncludeAlignments.cc we add following line:

#include "TotemUnifiedDatabaseAccessService/Tudas/interface/DatabaseAccessProvider.h"
```

Authors: Bartłomiej Alberski, Michał Idzik, Bartosz Niemczura
and change method:

TotemRPIncludeAlignments::TotemRPIncludeAlignments(const edm::ParameterSet &pSet) :
    verbosity(pSet.getUntrackedParameter<unsigned int>("verbosity", 1))
{
    std::cout << "------------------- Open connection to the database ---------------"
;
    DatabaseAccessProvider* serviceProvider = new DatabaseAccessProvider();
    serviceProvider->initialize();

    RomanPotManager* rpManager = serviceProvider->getRomanPotManager();

    cout<< "\n\n loadOffset:begin"
;
    map<string, double> resultOffsets = rpManager->loadOffsets(12341, "beta90");
    cout<< "\n\n loadOffset:end"
;
    serviceProvider->close();
    std::cout << "------------------- Close connection to the database ---------------";

    PrepareSequence("Measured", acsMeasured, pSet.getParameter< vector<string> >("MeasuredFiles");
    PrepareSequence("Real", acsReal, pSet.getParameter< vector<string> >("RealFiles");
    PrepareSequence("Misaligned", acsMisaligned, pSet.getParameter< vector<string> >("MisalignedFiles");

    setWhatProduced(this, &TotemRPIncludeAlignments::produceMeasured);
    setWhatProduced(this, &TotemRPIncludeAlignments::produceReal);
    setWhatProduced(this, &TotemRPIncludeAlignments::produceMisaligned); 

    findingRecord<RPMeasuredAlignmentRecord>();
    findingRecord<RPRealAlignmentRecord>();
    findingRecord<RPMisalignedAlignmentRecord>();
}

Into the BuildFile.xml we added line:

<use name="TotemUnifiedDatabaseAccessService/Tudas">

Then, one can use this plugin while running cmsRun with below configuration:

import FWCore.ParameterSet.Config as cms
process = cms.Process("GeometryOpticsInfo")

# minimum of logs 
process.load("Configuration.TotemCommon.LoggerMin_cfi")

# geometry 
process.load("Configuration.TotemCommon.geometryRP_real_cfi")
process.XMLIdealGeometryESSource.geomXMLFiles.append('Geometry/TotemRPData/data/2012_07_07_2/RP_Dist_Beam_Cent.xml')
#process.TotemRPGeometryESModule.verbosity = 0 

# include alignments, if any 
process.load("TotemAlignment.RPDataFormats.TotemRPIncludeAlignments_cfi")

Authors: Bartłomiej Alberski, Michał Idzik, Bartosz Niemczura
process.TotemRPIncludeAlignments.RealFiles = cms.vstring('
  'TotemAlignment/RPData/LHC/2012_07_07_2/sr+hsx/56_220.xml'
)

# no events to process
process.source = cms.Source("EmptySource")
process.maxEvents = cms.untracked.PSet(
  input = cms.untracked.int32(1)
)

process.GeomInfo = cms.EDAnalyzer("GeometryInfoModule",
  geometryType = cms.untracked.string("real")
)

process.p = cms.Path(
  process.GeomInfo
)