Optimization Services: A Framework For Distributed Optimization

Robert Fourer
Northwestern University
Jun Ma
Northwestern University
Kipp Martin
University of Chicago

November 4, 2007

Outline

Web Page

Project Wiki:

`projects.coin-or.org/OS`

see also

`www.optimizationservices.org`

What Is OS?

- A set of XML-based standards for representing information relevant to the practice of optimization, most importantly optimization instances (OSiL), optimization results (OSrL), and optimization solver options (OSoL).
- Open source libraries that support and implement many of the standards.
- A robust API for both solver algorithms and modeling systems. Corresponding to an OSiL instance representation there is an in-memory object, `OSInstance`, along with a set of `get()`, `set()`, and `calculate()` methods for accessing and creating problem instances. The API is for linear, integer, and general nonlinear programs.

What Is OS?

- A command line executable `OSSolverService` for reading problem instances (in OSiL format, AMPL `nl` format, or MPS format) and calling a solver either locally or on a remote server.
- Utilities that convert MPS files and AMPL `nl` files into the OSiL XML-based format.
- Standards that facilitate the communication between clients and optimization solvers using Web Services and libraries that support these standards.

What Is OS?

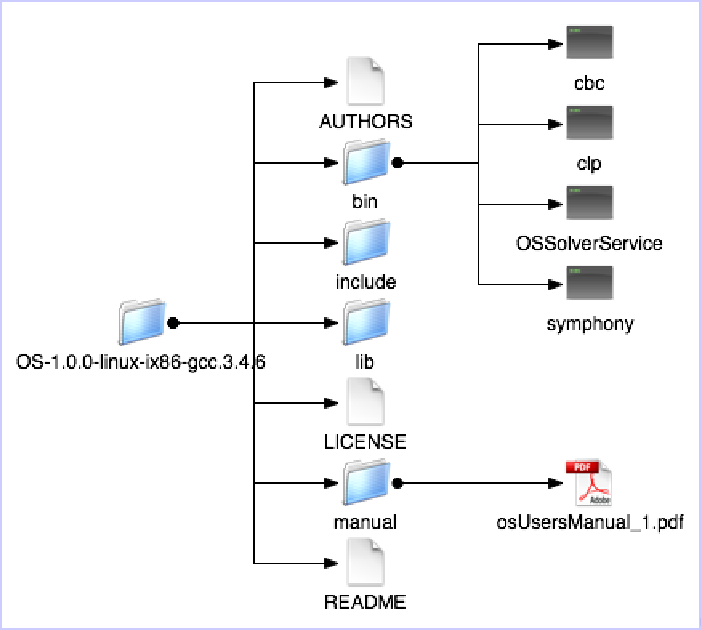
- An executable program `amplClient` that is designed to work with the AMPL modeling language. The amplClient appears as a “solver” to AMPL and, based on options given in AMPL, contacts solvers either remotely or locally to solve instances created in AMPL.

- Server software that works with Apache Tomcat and Apache Axis. This software uses Web Services technology and acts as middleware between the client that creates the instance, and solver on the server that optimizes the instance and returns the result.

Downloading the OS Project

1. Binary format
2. Use subversion (SVN) for source code
3. Download the source code in a tarball or zip file

Downloading the OS Project Binary



Downloading the OS Project Binary

Binary Format Available for:

- Windows with Microsoft Visual Studio cl compiler
- GNU/Linux 32 bit gcc 3.4.6
- Mac OS X (Intel) gcc 4.0.1

Downloading the OS Project Binary

Binary Format Also Available For Server Software:

OS-1.0.0-server-distribution.tgz

- os-server-1.0.0 (OS software + Apache Tomcat)
- os.war (OS software)

Contains an `OSSolverService` for Linux, Windows, and Mac.

More from Jun Ma in session SD29.

Downloading the OS Project Source Code

You can checkout the source code using subversion.

Get a release version:

```
svn co https://projects.coin-or.org/svn/OS/releases/1.0.0 OS
```

Get a stable version:

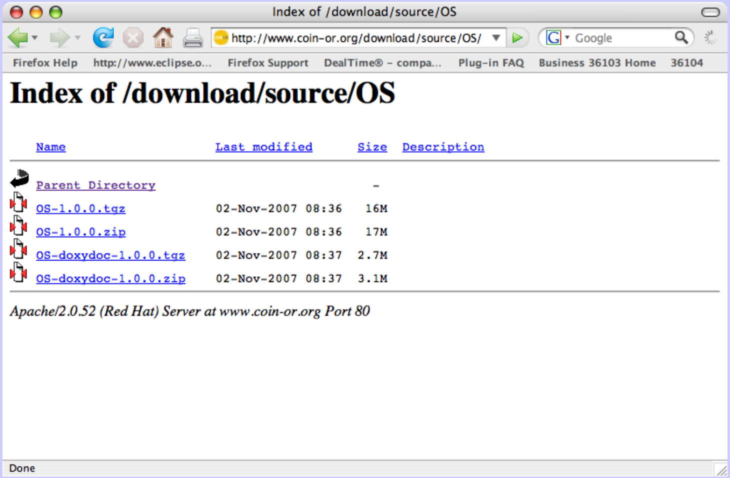
```
svn co https://projects.coin-or.org/svn/OS/stable/1.0 OS
```

Get a trunk version:

```
svn co https://projects.coin-or.org/svn/OS/trunk OS
```

Downloading the OS Project Source Code

You can checkout the source code as tarballs or zip files.

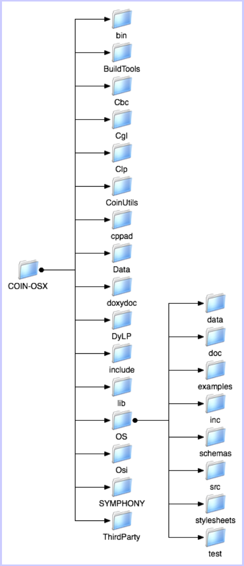


Index of /download/source/OS

```
    Name                          Last modified      Size  Description
--------------------------------------------------------------------------
    Parent Directory                                   -
    OS-1.0.0.tgz                  02-Nov-2007 08:36   16M
    OS-1.0.0.zip                  02-Nov-2007 08:36   17M
    OS-doxydoc-1.0.0.tgz          02-Nov-2007 08:37  2.7M
    OS-doxydoc-1.0.0.zip          02-Nov-2007 08:37  3.1M
--------------------------------------------------------------------------
```

Apache/2.0.52 (Red Hat) Server at www.coin-or.org Port 80

OS Root

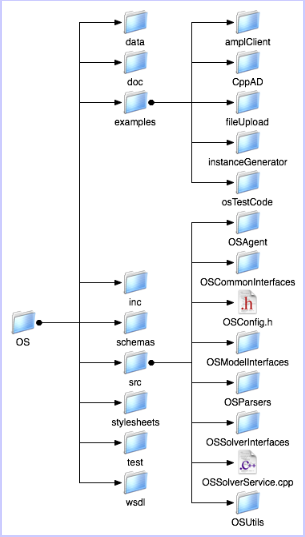


COIN-OSX
bin
BuildTools
Cbc
Cgl
Clp
CoinUtils
cppad
Data
doxydoc
DyLP
include
lib
OS
Osi
SYMPHONY
ThirdParty
data
doc
examples
inc
schemas
src
stylesheets
test

OS Project Dependencies

- Buildtools
- CoinUtils
- Cbc
- Cgl
- Clp
- CppAD
- DyLP
- Ipopt
- Osi
- SYMPHONY
- Vol

OS Project Root



Building the OS Project

Build Flavors:

- Pure Unix
- Hybrid Microsoft-Unix
- Pure Microsoft

Building the OS Project – Unix

Project is designed to work with autotools:

After downloading, do:

```
./configure
make
make test
make install
```

There is a fairly extensive `unitTest`.

Note: may wish to build without Ipopt if you don't have FORTRAN 95

```
./configure  COIN_SKIP_PROJECTS=Ipopt
```

Building the OS Project - Pure Windows

Use the Version 7 or Version 8 Microsoft Visual Studio **Solution** and **Project** files that download with the project. There are project files for:

- The OS lib
- The OSSolver Service
- The OS unitTest

Note: Project files do not include any Third Party software (nothing outside of COIN-OR)

Building the OS Project - Windows-Unix Hybrids

Hybrid in the following sense: we are using the Unix auto tools (which are not Windows native) to do an OS build.

- Cygwin/gcc
- Cygwin/cl
- MINGW/gcc
- MSYS/cl

OS builds on all of the above

Platforms

Summary: OS has been built successfully on:

- Various flavors of GNU/Linux
- Windows using Microsoft Visual Studio
- Windows using MSYS and Microsoft cl
- Windows using MINGW and gcc
- Windows using Cygwin and gcc
- Windows using Cygwin and cl (an excellent platform for users with too much free time)
- Mac OS X (both Intel and Power PC)

OS Library Components

- OSAgent
- OSCommonInterfaces
- OSModelInterfaces
- OSParsers
- OSSolverInterfaces
- OSUtils

Using the API

The **OSCommon** library provides in-memory representation of an optimization instance, **OSInstance**. It is an API that has three types of methods:

- **get() methods:** a set of methods to get information about the problem instance
- **set() methods:** a set of methods to create/modify a problem instance
- **calculate() methods:** a set of methods for performing Algorithmic Differentiation (based upon the COIN-OR CppAD – see talk in Session SD29 by Brad Bell).

get() Methods

get() methods: a set of methods to get information about the problem

For example, the **CoinSolver** class takes and **OSInstance** object and creates an instance for an COIN Osi compatible solver.

```
osinstance->getVariableNumber()
```

```
osinstance->getConstraintUpperBounds()
```

```
osinstance->getLinearConstraintCoefficientsInColumnMajor()->values
```

You can also use **get()** methods to get the problem in **postfix** or **prefix** format.

set() Methods

set() methods: a set of methods to get information about the problem

See **OS/examples/instanceGenerator** for an example of creating a problem instance using the set() methods

```
osinstance->setVariableNumber( 2);

osinstance->addVariable(1, "x1", 0, 1, 'B', OSNAN, "");
```

calculate() Methods

calculate() methods: a set of methods to calculate constraint and objective function

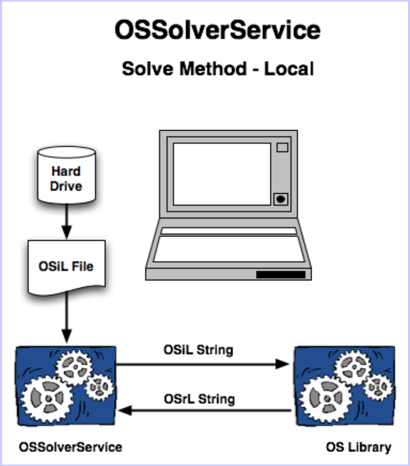
- values
- gradients
- Hessians

```
sparseJacobian = osinstance->getJacobianSparsityPattern();
sinstance->calculateAllConstraintFunctionGradients();
```

```
osinstance->getLagrangianHessianSparsityPattern( );
osinstance->calculateLagrangianHessian()
```

Using the OSSolverService

The OS build includes the **OSSolverService** executable. This executable can be called locally, or on a remote server.
A local call:



Using the OSSolverService

Here is the local call

```
OSSolverService -config
                ../data/configFiles/testlocal.config
```

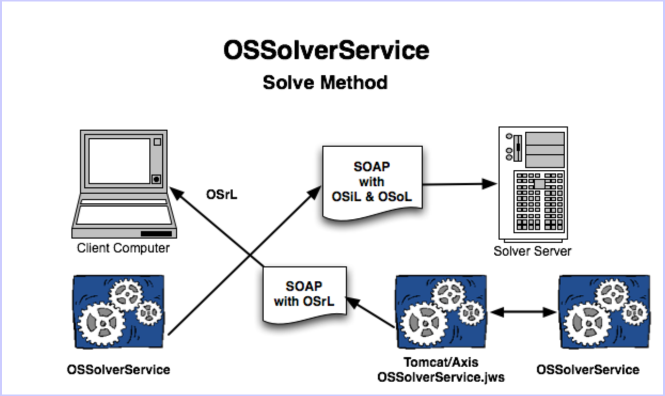
where **testlocal.config** is

```
-osil ../data/osilFiles/parincLinear.osil
-solver ipopt
-serviceMethod solve
```

Options at command line override options in the configure file.

Using the OSSolverService

A remote call:



Using the OSSolverService

Here is the local call

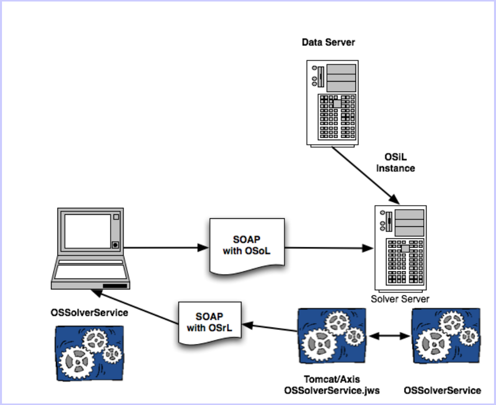
```
OSSolverService -config
                ../data/configFiles/testremote.config
```

where **testremote.config** is

```
-serviceLocation
    http://gsbkip.chicagogsb.edu/os/OSSolverService.jws
-osil ../data/osilFiles/parincLinear.osil
```

Using the OSSolverService

A remote call with data solver server and data server:



Using the OSSolverService

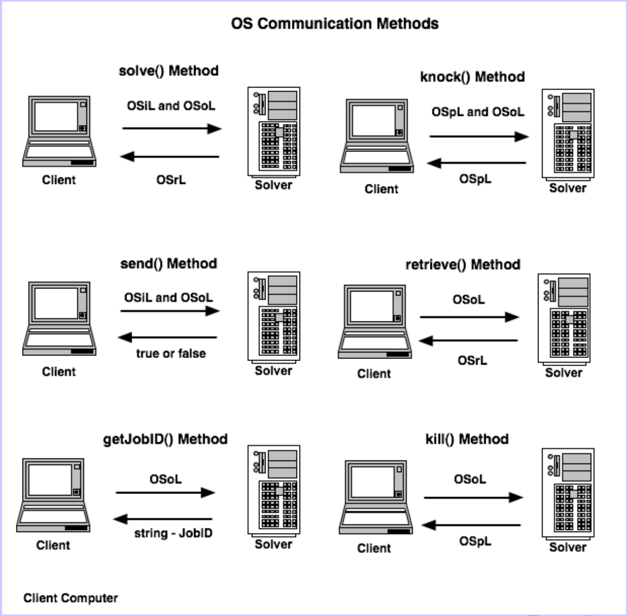
To have the solver server call a data server for the model instance send it some OSoL with the

```
<instanceLocation>
```

specified

```
<general>
<instanceLocation locationType="http">
http://www.coin-or.org/OS/parincLinear.osil</instanceLocati
</general>
<optimization>
     <other name="os_solver">ipopt</other>
</optimization>
```

Using the OSSolverService with OShL



Solvers

OSSolverService has an interface for the following solvers:

- Clp (COIN-OR Osi Interface)
- Cbc (COIN-OR Osi Interface)
- Cplex (COIN-OR Osi Interface)
- Dylp (COIN-OR Osi Interface)
- Glpk (COIN-OR Osi Interface)
- Ipopt
- Knitro
- Lindo
- SYMPHONY (COIN-OR Osi Interface)
- Vol (COIN-OR Osi Interface)

Examples

In the **OS** directory, there is an **examples** directory with:

- **algorithmicDiff**
- **amplClient**
- **fileUpload**
- **instanceGenerator**

Examples – amplClient

To invoke a solver locally using AMPL and amplClient:

```
# take in problem 71 in Hock and Schittkowski
model hs71.mod;
# tell AMPL that the solver is amplClient
option solver amplClient;
# now tell amplClient to use Ipopt
option amplClient_options "solver ipopt";
# the name of the nl file (this is optional)
write gtestfile;
# now solve the problem
solve;
```

Examples – amplClient

To invoke a solver remotely using AMPL and amplClient, after the command

```
option amplClient_options "solver ipopt";
```

Next, set the solver `service` option to the address of the remote solver service.

```
option ipopt_options
"service http://gsbkip.chicagogsb.edu/os/OSSolverService.jws";
```

Documentation

- OS User's Manual in pdf format
- OS User's Manual online
- Doxygen
- See also **www.optimizationservices.org**