Optimization Services
hookup Language (OShL)

-- API for invoking Remote Solver Service
-- Think of HTTP/HTML, JDBC/ODBC
-- Open Source
-- Platform Independent
-- Programming Language Independent

Jun Ma
Robert Fourer
Northwestern University
Kipp Martin
University of Chicago

Jun Ma
INFORMS, Pittsburgh
11/08/2006
OUTLINE

1. Optimization Services and OSP (OSxL’s)
2. Invoking a remote solver service
3. Optimization Services hookup Language (OShL)
4. Conclusion
XML-based standard

Optimization Services (OS)
THE Optimization Internet

Modeler

Parser to OSiL

OS Server

Solver

Web address

browser

html form

socket

http/html

Max \( f(x) \)

s.t. \( l_b_1 \leq g_1(x) \leq u_b_2 \)

\( l_b_2 \leq g_2(x) \leq u_b_2 \)

f(x) can be \( \sin(x(1)) + x(x(2)) \)

g_1(x) can be \( \text{if}(x(1)>0) \text{then} x(2) \text{else} \text{cost}(x(2)) \)

g_2(x) can be a metric from a finite element simulation (non-closed form black box function evaluator)

OShL – Optimization Services hookup Language

Analyzer

Objective Registry

Solver

Solver

OS/OSxL

CGI

OS Server

OS Server

OS Server

OS Server

Model/Data

AMPL

Lingo

Robert Fourer, Jun Ma, Kipp Martin
Copyright 2006
Invoking remote solver service (1)

**Requirement**
1. Platform Independent
2. Language Independent
3. Protocol Independent
4. Type compatibility
5. Simple
6. Built-in state/session maintenance
7. Request and response/Blocking

**Solution**
- OS is Web services based
- OShL is XML Based WSDL
- OShL is in SOAP envelope
- Uses all strings as arguments
- 6 methods, 2 args, only 1 method impl., no network knowledge req.
- jobID
- solve(instance, option) -> result
- getJobID(option) -> jobID

Robert Fourer, Jun Ma, Kipp Martin
Copyright 2006
Invoking remote solver service (2)

Requirement
8. Communication only
9. Truly asynchronous (server side)
10. Retrieve anytime, anywhere
11. Must stop remote process

Solution
No specification on arguments

Argument specification -> xml schemas -> OSiL, OSoL, OSrL etc.

send(instance, option) -> true/false
retrieve(option) -> result
kill(option) -> killProcessResult

Send (≠ stop on the browser)
**Invoking remote solver service (3)**

**Requirement**

8. Dynamic process information (heartbeat)  

9. Extendable

**Solution**

Knock

2 inputs, 1 output (leverages on OSpL)

- `knock(inputProcess, option) -> outputProcess`
- `ping`  
- `getServiceStatistics`  
- `getJobStatistics`  
- `getOptimizationStatistics`  
- `getAll`  
- `notifyJobCompletion`  
- `setServiceStatistics`  
- `setJobStatistics`  
- `setOptimizationStatistics`  
- `setAll`  
- `requestJob`
Invoking remote solver service (4)

knock(inputProcess, option) -> outputProcess
Optimization Services hookup Language (OShL)

getJobID (String OSoL) -> jobID
solve (String OSiL, String OSoL) -> OSrL
send (String OSiL, String OSoL) -> true/false
retrieve (String OSoL) -> OSrL
kill (String) -> OSpL
knock(String OSpL, String OSoL) -> OSpL
Our OShL-compatible solver hosting SERVER reference impl.

Remote job submission, management and control
Remote retrieval of previously submitted jobs
Session and state maintenance
Synchronous and asynchronous solver invocation
Killing long jobs over the remote server
Checking and managing service status and job statistics
Automatic job completion notification via most common protocols including emails
Persistence between service starts
Service logging
Automatic notification of critical service information to admin
Centralized user configuration
Directory and file cleanup
Disk, memory and process cleanup
Critical data backing up
Waiting job queue management
Long computational job handling
Job dependency handling
Keeping track of service utilization and preparing periodic report
Automatic input and output validation and processing
Support of machines with multiple CPUs
Support of all major operating systems
Authentication and authorization
Security
Conclusion

• Optimization Services, OS Protocols (OSP => OSxL’s)
• Design Requirement in Invoking Remote Solver Service
• Optimization Services

• Optimization Services will be released end of the year or beginning of next year.
• Almost all major parties (commercial, open source, research projects) are adopting it (It’s a private process now!)
• Next generation NEOS
• Critical role in Cyber-infrastructure
• A Mega COIN project – touches nearly all the major COIN optimization-related projects
• “Run-Time” COIN (COMputational INfrastructure)
• Optimization Internet
• Contact us in private