Query Processing in a WSMS

Utkarsh Srivastava
Stanford University

Joint work with
Jennifer Widom, Kamesh Munagala, Rajeev Motwani, and Gene Pang
What are Web Services?

Highly standardized method of sharing data and functionality

Discovery and Description

User/Client \(\xrightarrow{\text{UDDI, WSDL}}\) Web Services
What are Web Services?

Highly standardized method of sharing data and functionality

Communication

User/Client  SOAP  Web Services
Example

StockQuotes

Click [here](http://ws.invesbot.com/stockquotes.asmx?op=GetQuotes) for a complete list of operations.

GetQuotes

Enter symbols, separated by space, Quotes delayed in 20 minutes.

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbols:</td>
<td>GOOG</td>
</tr>
</tbody>
</table>

Invoke
Result of Invocation

Address: http://ws.invesbot.com/stockquotes.asmx/GetQuotes

```xml
<?xml version="1.0" encoding="utf-8" ?>
<StockQuotes>
  <StockQuote>
    <Symbol>COOC</Symbol>
    <Price><big><b>348.19</b></big></Price>
    <Time>4:00PM ET</Time>
    <Change>
      <img width="10" height="14" border="0" src="http://us.i1.yimg.com/us.yimg.com/i/us/fi/03rd/talt="Up"" style="color:#008800;"/>8.40 (2.47%)</Change>
    <PrevClose>339.79</PrevClose>
    <Open>342.01</Open>
    <Bid>348.05</Bid> x 100</small></Bid>
    <Ask>348.19</Ask> x 1300</small></Ask>
    <YearTarget>482.01</YearTarget>
    <DayRange>341.54 - 350.09</DayRange>
    <YearRange>177.25 - 475.11</YearRange>
    <Volume>10,415,814</Volume>
    <AvgVol>13,940,800</AvgVol>
    <MarketCap>102.91B</MarketCap>
    <PE>69.35</PE>
    <EPS>5.02</EPS>
    <DivYield>N/A (N/A)</DivYield>
  </StockQuote>
</StockQuotes>
```
Query Across Web Services

find info. about all companies whose stock had >10% change

Complex

User/Client

Query

Results

WS₁

symbol

company info.

symbol

stock price info.

WS₂

NASDAQ
Taking a Cue from DBMSs

Simple

Declarative Interface

All the complexity

User/Client

Query

Results

Database Management System

Data
Web Service Management System

User/Client

Web Service Management System (WSMS)

Query Results

symbol

company info.

stock price info.

WS1

WS2

symbol

NASDAQ
Web Service Management System

Declarative Interface | WSMS | WS Invocations

Metadata Component
- WS Registration
- Schema Mapper

Query Processing Component
- Plan Selection
- Plan Execution

Profiling and Statistics Component
- Response Time Profiler
- Statistics Tracker

Query and Input Data
Client
Results

WS1
WS2
... 
WSn
Query over Web Services – An Example

Credit card company wishes to send offers to those
a) Who have a credit rating > 50, and,
b) Who have a payment history = “Good” on a prior card.

Company has at its disposal
- L: List of potential recipient SSNs
- WS$_1$: SSN ! credit rating
- WS$_2$: SSN ! card no(s)
- WS$_3$: card no. ! payment history
Plan 1

WSMS  SSN  
     SSN, cr

Filter on cr, keep SSN

WS1  SSN ! cr

SSN  cr
1  60
2  70

WS2  SSN ! ccn

SSN ccn
1  xx1
2  xx2

WS3  ccn ! ph

ccn ph
xx1 B
xx2 G

Client  

Results  

L (SSN)

SSN
1
2

SSN
1
2

SSN
2

SSN, ccn

Filter on ph, keep SSN

Note pipelined processing
Simple Representation of Plan 1

L → WS_1 → WS_2 → WS_3 → Results
Plan 2

Client

WSMS

Filter on cr, keep SSN

SSN, cr

SSN

Join

SSN, ccn

Filter on ph, keep SSN

SSN, ccn, ph

Results

SSN

1

2

L (SSN)

WSMS

SSN, cr

SSN

WS1

SSN ! cr

SSN

1

60

2

70

WS2

SSN ! ccn

SSN, ccn

WS3

crn ! ph

ccn

xx1

xx2

ph

B

G

ccn

xx1

xx2
Simple Representation of Plan 2

\[ L \xrightarrow{\text{WS}_1} \text{Results} \xrightarrow{\text{WS}_2} \text{WS}_3 \]
Quiz

Cost Metric: Steady-state throughput

Which plan is better?

Plan 1

L $\rightarrow$ $WS_1$ $\rightarrow$ $WS_2$ $\rightarrow$ $WS_3$ $\rightarrow$ Results

Plan 2

L $\rightarrow$ $WS_1$ $\rightarrow$ $WS_2$ $\rightarrow$ $WS_3$ $\rightarrow$ Results

In Plan 1, $WS_2$ has to process only filtered SSNs
In Plan 2, $WS_2$ has to process all SSNs
Query Planning Recap

- Possible plans $P_1, \ldots, P_n$
- Statistics $S$
- Cost Metric $\text{cost}(P_i, S)$
- Want to find least-cost plan
Class of Queries Considered

- "Select-Project-Join" queries over input data and set of web services
- Precedence constraints
  - Input for $WS_i$ may be provided by the output of $WS_j$
    e.g., $WS_2$: SSN ! ccn and $WS_3$: ccn ! ph
  - Precedence constraints impose a DAG.
Statistics: Response Time

- $c_i$: per-tuple response time of $WS_i$ from client

Assume independent response times
Statistics: Selectivity

- $s_i$: selectivity of $WS_i$
  
  Average number of output tuples per input tuple to $WS_i$

  a) $WS_1$: SSN ! cr
      
      If 90% individuals have cr > 50, $s_1 = 0.9$

  b) $WS_2$: SSN ! ccn
      
      If on average each SSN holds 2 credit cards, $s_2 = 2$

- Assume independent selectivities
Bottleneck Cost Metric

Lunch Buffet

Overall per-item processing time = Response time of slowest or bottleneck stage in pipeline
Cost Expression for Plan $P$

$R_i(P)$: Predecessors of $WS_i$ in plan $P$

$\downarrow$

Fraction of input tuples seen by $WS_i = \prod_{j \mid j \in R_i(P)} s_j$

$\downarrow$

Response time per original input tuple at $WS_i$

$\left( \prod_{j \mid j \in R_i(P)} s_j \right) \cdot c_i$

$\downarrow$

$\text{cost}(P) = \max_{1 \leq i \leq n} \left( \prod_{j \mid j \in R_i(P)} s_j \right) \cdot c_i$

Assumption: WSMS cost is not the bottleneck
Problem Statement

Input:
- Set of web services $WS_1, \ldots, WS_n$
- Response times $c_1, \ldots, c_n$
- Selectivities $s_1, \ldots, s_n$
- Precedence constraints among web services

Output:
- Arrange web services into a plan $P$
- $P$ respects all precedence constraints
- $\text{cost}(P)$ by the bottleneck metric is minimized
No Precedence Constraints

- All selectivities · 1
  - Theorem: Optimal to linearly order by increasing $c_i$

- General case

![Diagram showing proliferative and selective web services joined at WSMS leading to results.]

Local Join at WSMS
With Precedence Constraints

\[
\text{cost}(P) = \sum_{1 \leq i \leq n} \left( \prod_{j \mid j \in R_i(P)} s_j \right) \cdot c_i
\]

Sum cost metric

- Hard to approximate to within a factor \(O(n^\theta)\)

Bottleneck cost metric

- Surprisingly, solvable in polynomial time
- Developed an \(O(n^5)\) algorithm
  - Adds one WS at a time to the plan
  - WS to be added is chosen by solving a linear program.
Isn’t this the same as …?

- **Web Service Composition**
  - Targeted towards workflow-oriented applications
  - Don’t give provably optimal strategies

- **Parallel and Distributed Query Optimization**
  - Freedom to move query operators around
  - Much larger space of execution plans

- **Data Integration, Mediators**
  - Integrate general sources of data
  - Primarily optimize the cost at the integration system itself
Implementation

- Building a prototype general-purpose WSMS
  - Written in Java
  - Uses Apache Axis, an open-source implementation of SOAP
  - Implements query planning and execution
Future Directions

- Monetary cost of invoking web services
  - Optimize combination of response time and cost

- Variations in web service response times
  - Depends on provisioning, load, network conditions
  - Consider adaptive plans and/or robust plans

- Statistics Collection
  - Self-tuning histograms are relevant

- Extension to optimizing workflows
Conclusion

http://infolab.stanford.edu/wsms

User/Client

Query

Results

WS

Questions?

Web Services