Query Processing in a WSMS

Utkarsh Srivastava
Stanford University

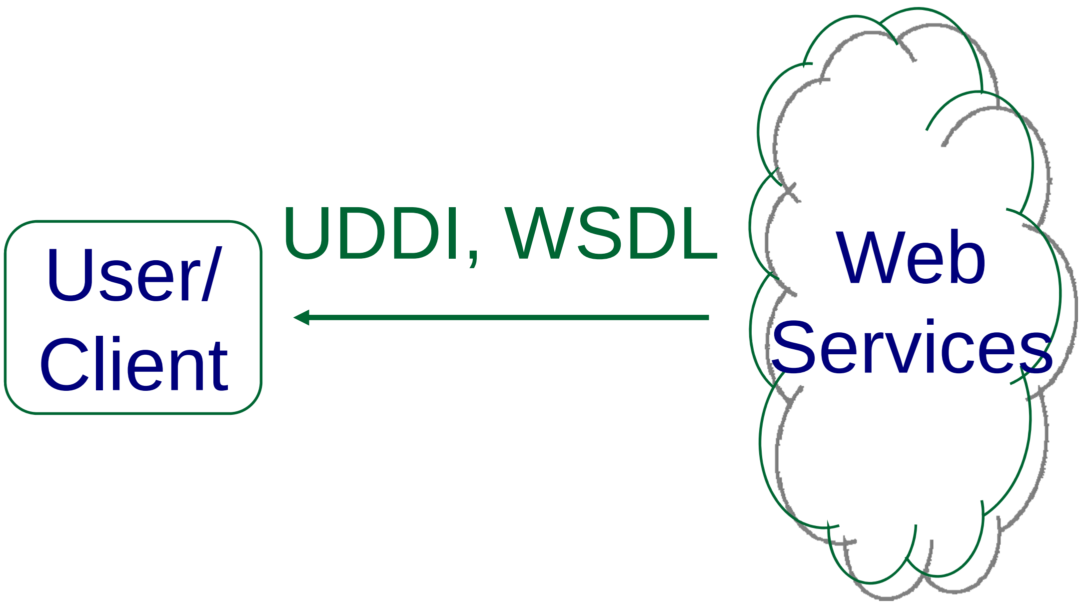
Joint work with

Jennifer Widom, Kamesh Munagala, Rajeev Motwani, and Gene Pang

What are Web Services?

Highly standardized method of sharing data and functionality

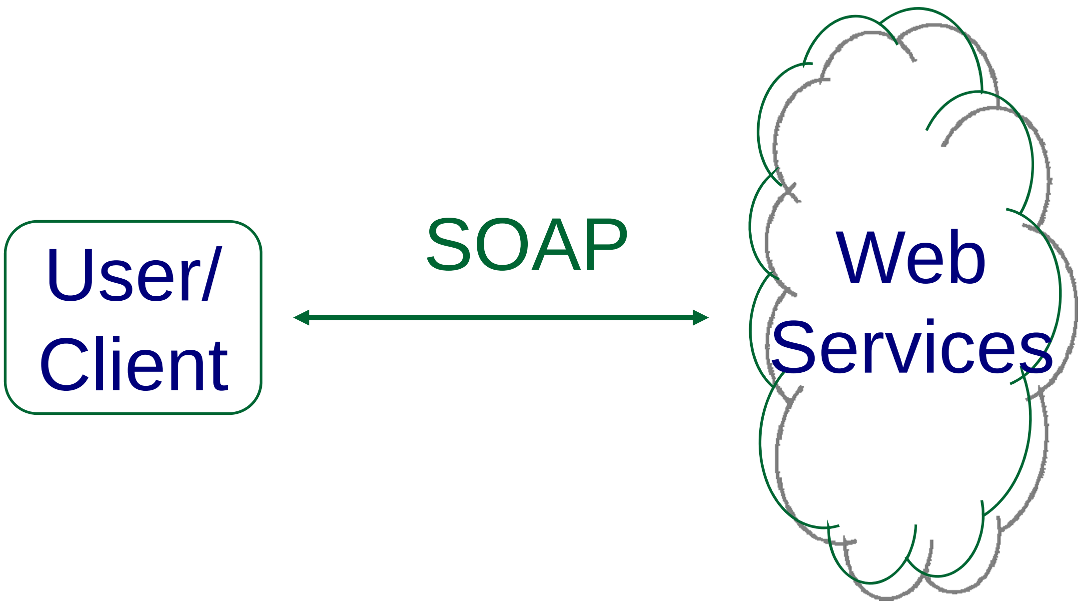
Discovery and Description



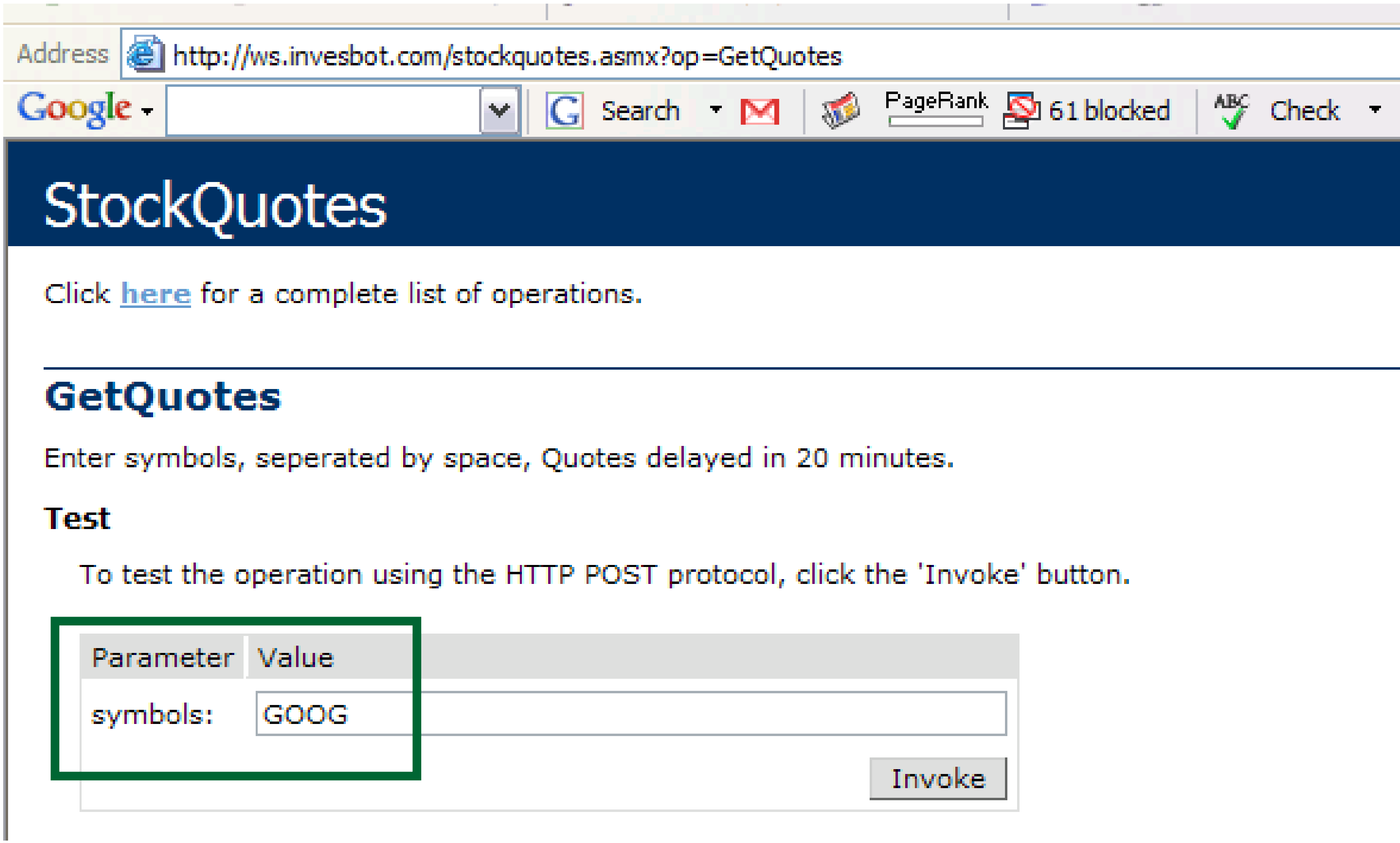
What are Web Services?

Highly standardized method of sharing data and functionality

Communication



Example


Address
http://ws.invesbot.com/stockquotes.asmx?op=GetQuotes
Google
Search
PageRank
61 blocked
Check
StockQuotes
Click here for a complete list of operations.
GetQuotes
Enter symbols, seperated by space, Quotes delayed in 20 minutes.
Test
To test the operation using the HTTP POST protocol, click the 'Invoke' button.
Parameter
Value
symbols:
GOOG
Invoke

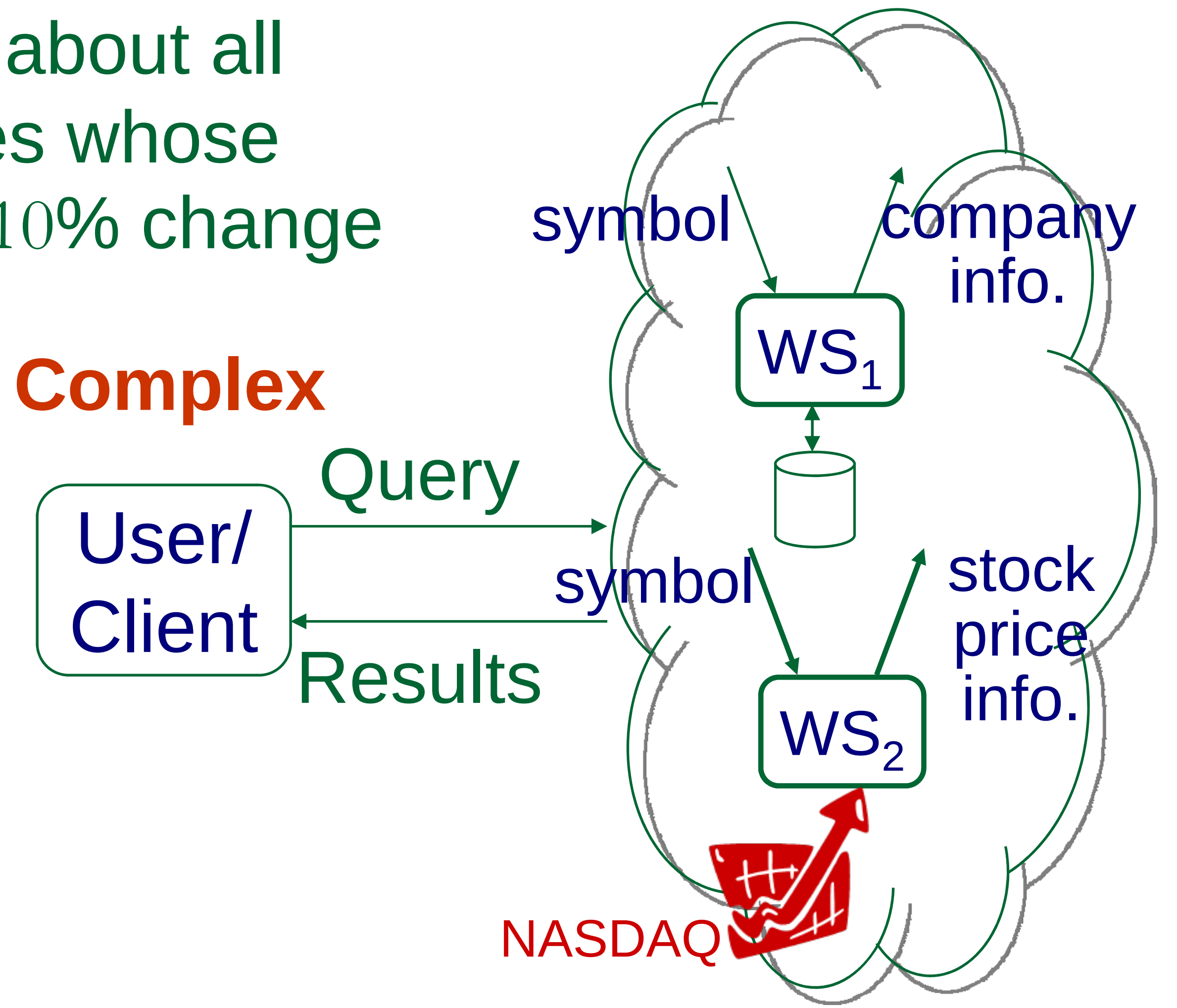
Result of Invocation

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

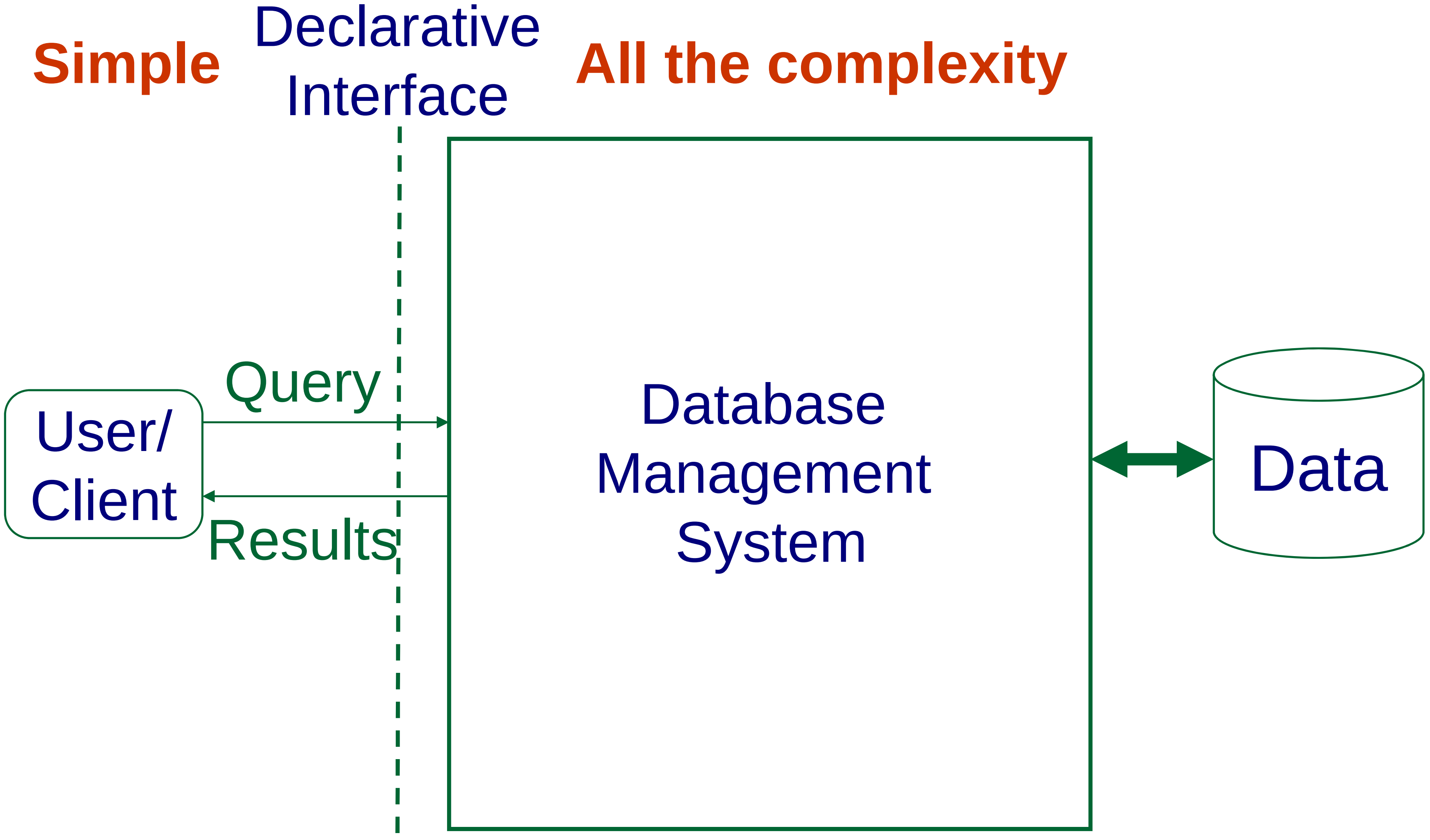

```
<?xml version="1.0" encoding="utf-8" ?>
- <StockQuotes>
  - <StockQuote>
      <Symbol>GOOG</Symbol>
      <Price><big><b>348.19</b></big></Price>
      <Time>4:00PM ET</Time>
      <Change><img width="10" height="14" border="0" src="http://us.i1.yimg.com/us.yimg.com/i/us/fi/03rd/u
        alt="Up"> <b style="color:#008800;">8.40 (2.47%)</b></Change>
      <PrevClose>339.79</PrevClose>
      <Open>342.01</Open>
      <Bid>348.05<small> x 100</small></Bid>
      <Ask>348.19<small> 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

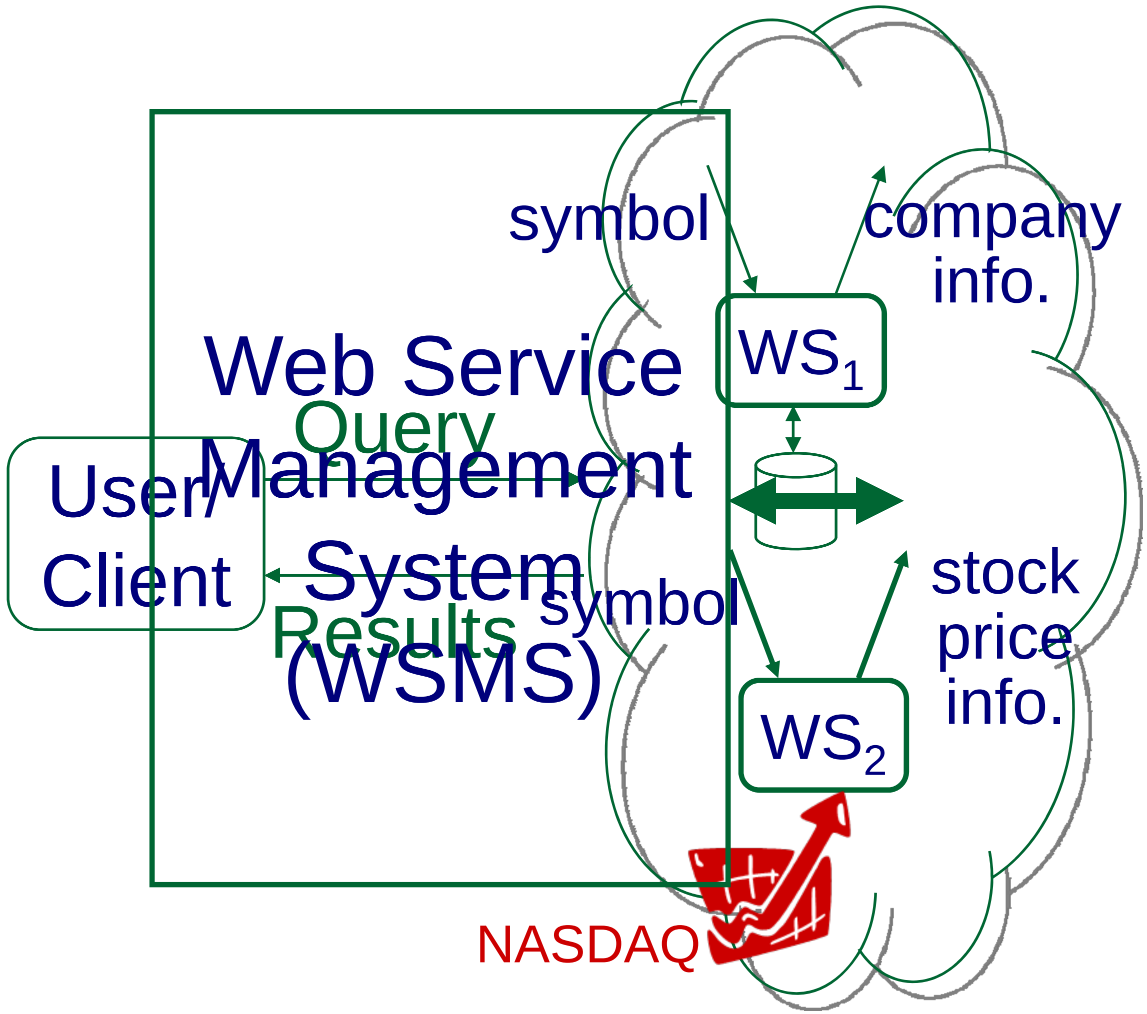
find info. about all companies whose stock had >10% change



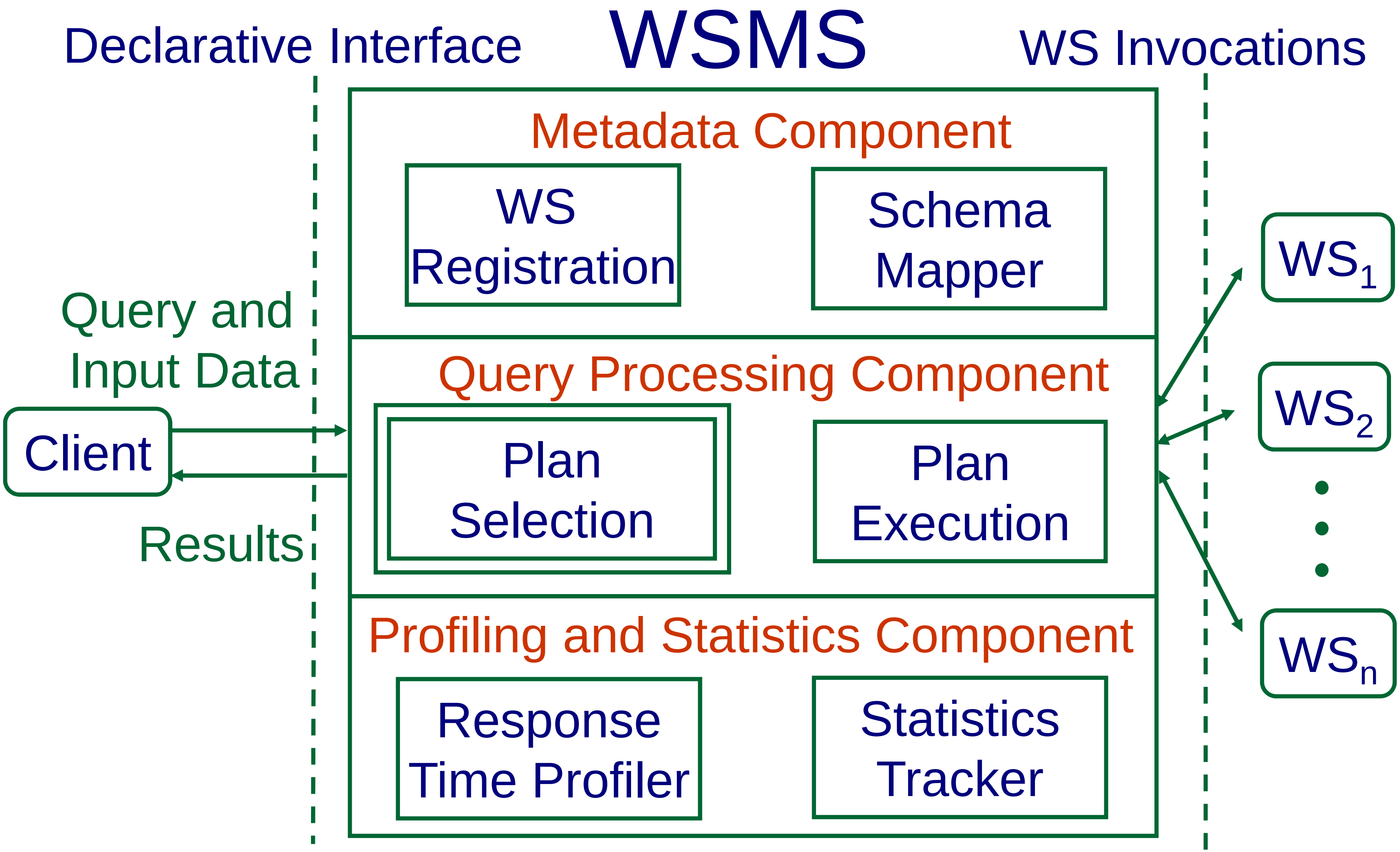
Taking a Cue from DBMSs



Web Service Management System



Web Service Management System

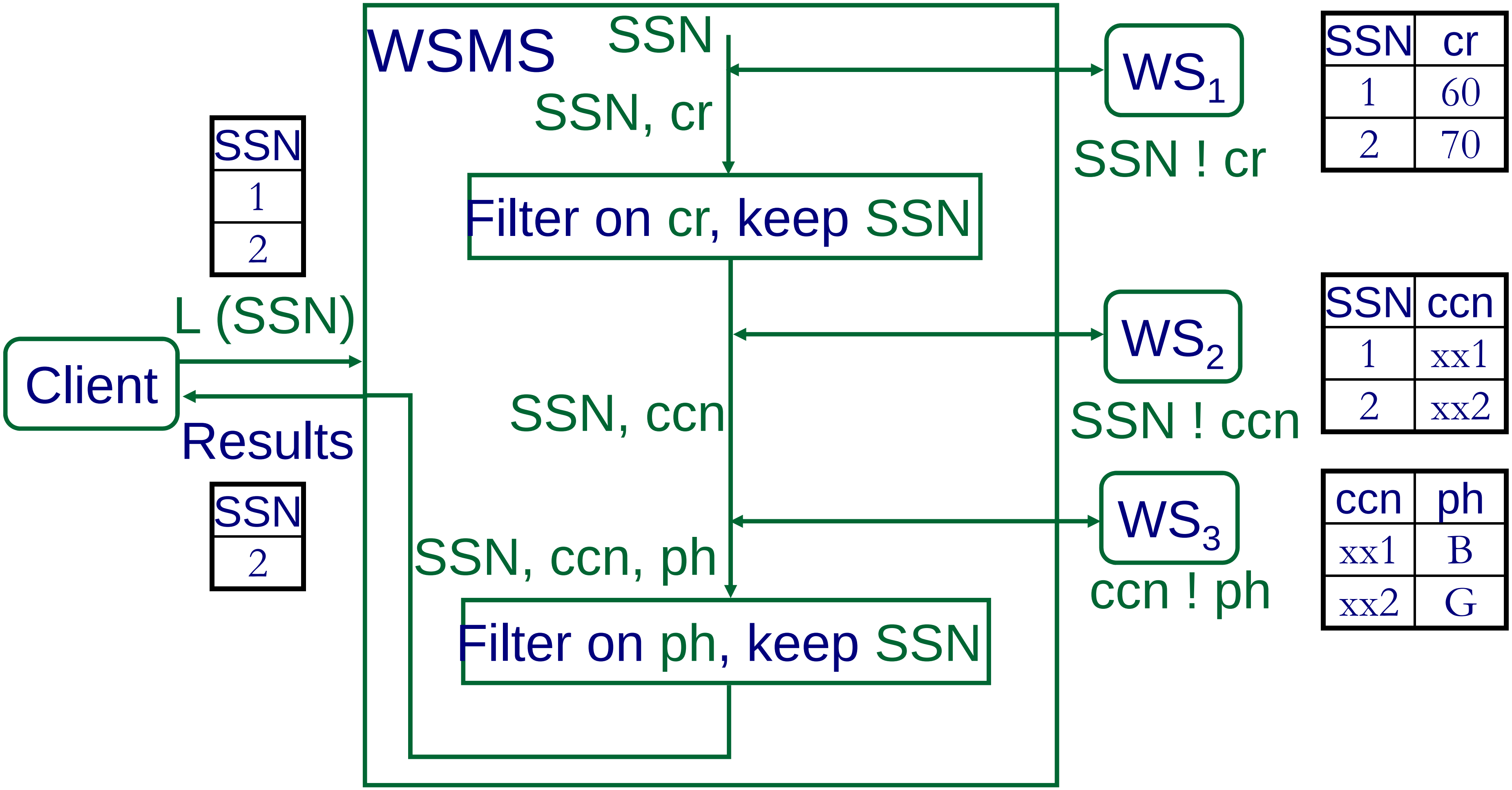


Query over Web Services – An Example

- 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

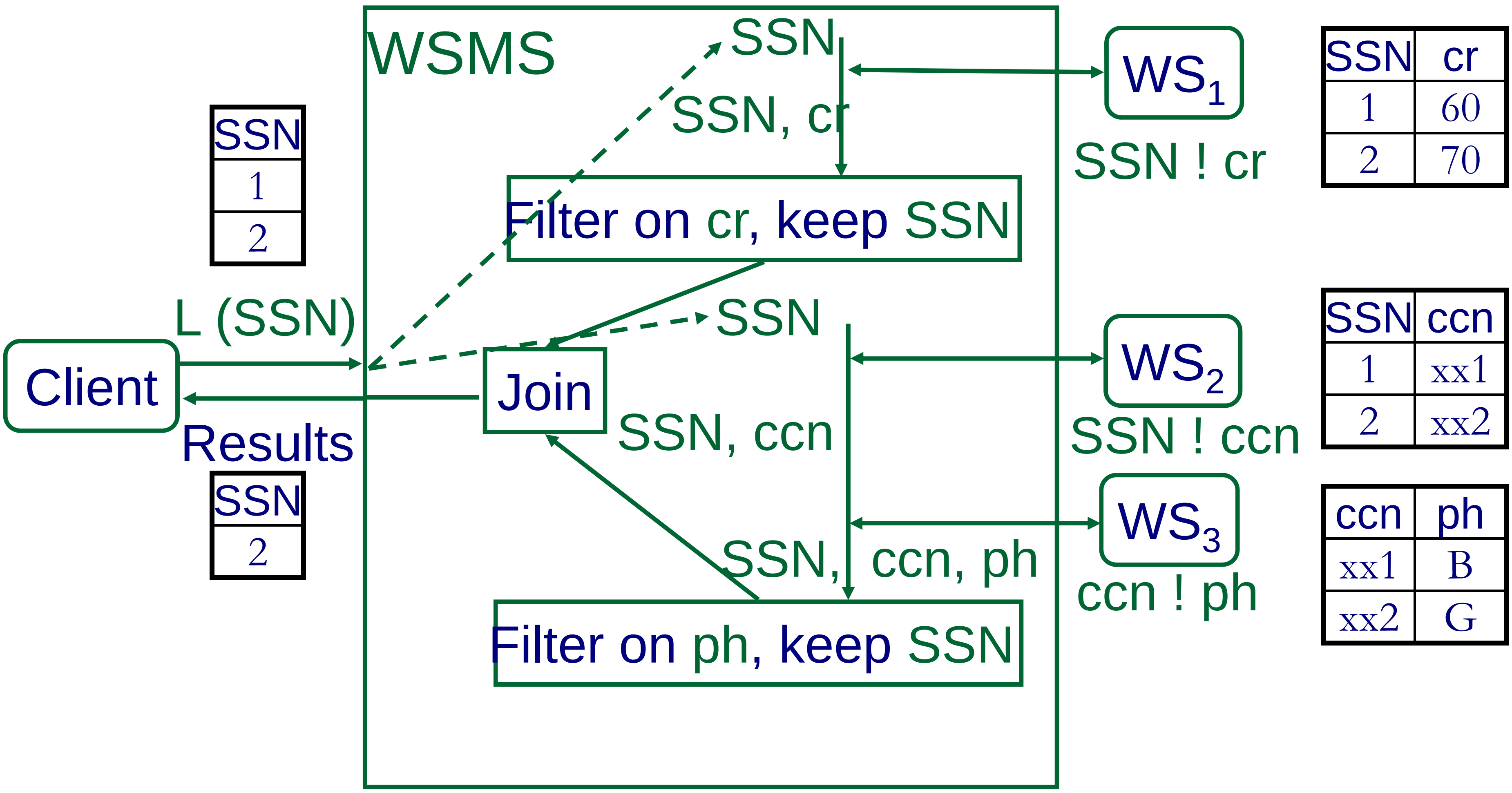


Note pipelined processing

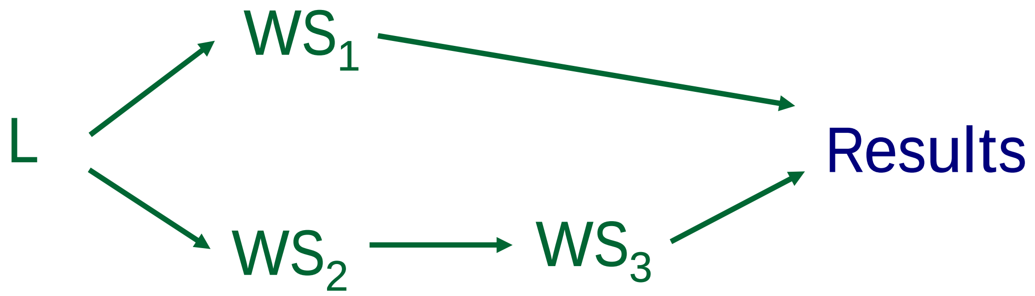
Simple Representation of Plan 1

$L \longrightarrow WS_1 \longrightarrow WS_2 \longrightarrow WS_3 \longrightarrow$ Results

Plan 2


WSMS
SSN
SSN, cr
Filter on cr, keep SSN
SSN
Join
SSN, ccn
SSN, ccn, ph
Filter on ph, keep SSN
Client
L (SSN)
Results
SSN
1
2
SSN
2
WS1
SSN ! cr
WS2
SSN ! ccn
WS3
ccn ! ph
SSN cr
1 60
2 70
SSN ccn
1 xx1
2 xx2
ccn ph
xx1 B
xx2 G

Simple Representation of Plan 2



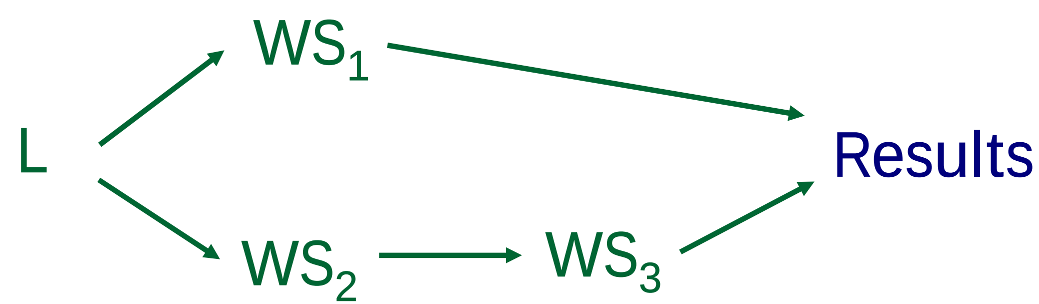
Quiz

Cost Metric: Steady-state throughput

Which plan is better?

Plan 1

Plan 2



In Plan 1, WS_2 has to process only filtered SSNs

In Plan 2, WS_2 has to process all SSNs

Query Planning Recap

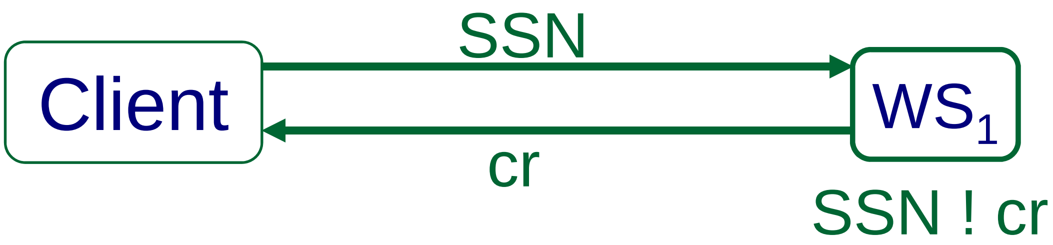
- Possible plans $P_1, \ldots, P_n$
- Statistics S
- Cost Metric $cost(P_i, S)$
- Want to find least-cost plan

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, $s_1 = 0.9$

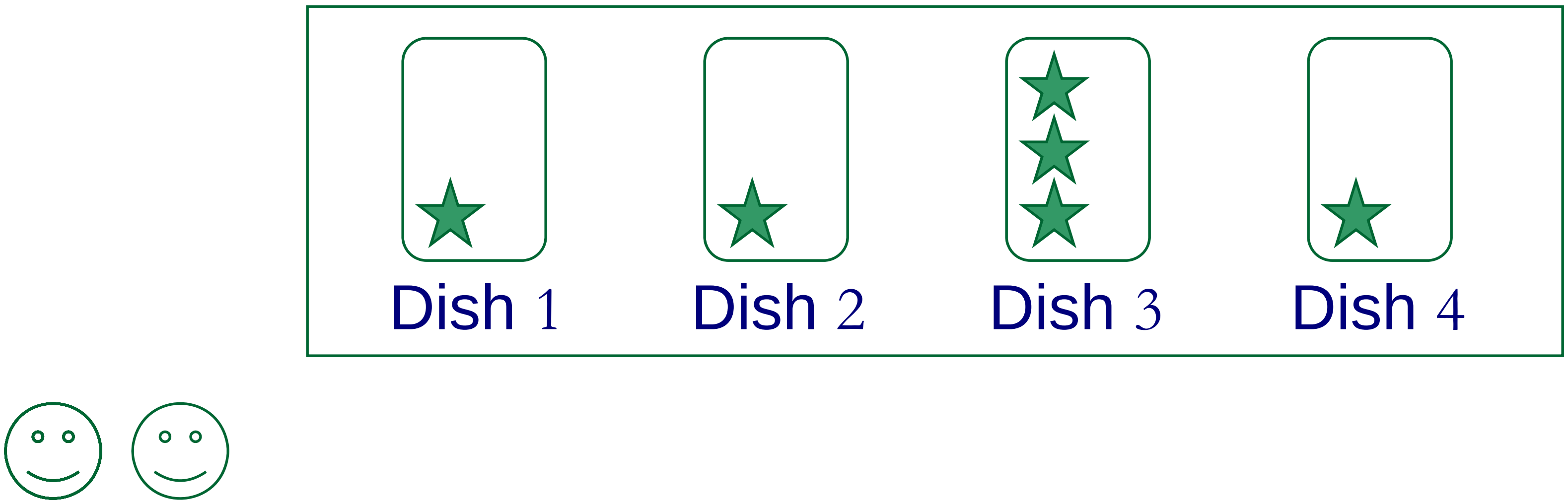
  b) WS_2: SSN ! ccn

  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

⇩

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

⇩

Response time per original input tuple at WS_i

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

⇩

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

Assumption: WSMS is not the bottleneck

Contrast with sum cost metric

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

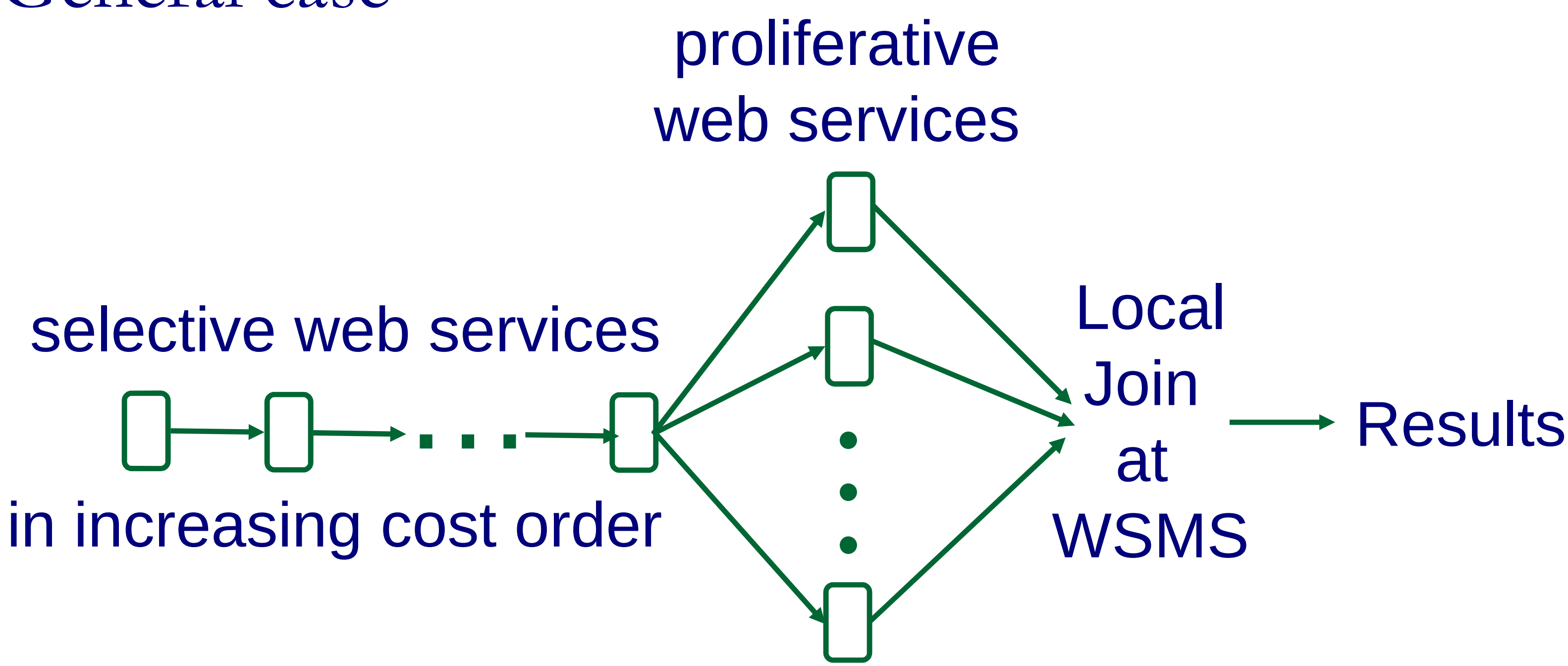
Output:

- Arrange web services into a plan P
- P respects all precedence constraints
- cost(P) by the bottleneck metric is minimized

No Precedence Constraints

- All selectivities · 1
  - Theorem: Optimal to linearly order by increasing $\mathbf{c}_i$

- General case



With Precedence Constraints

$$\text{cost}(P) = \sum_{1 \le i \le n} \Big(\prod_{j \,|\, j \in R_i(P)} s_j \Big) \cdot c_i$$

<u>Sum cost metric</u>

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

<u>Bottleneck cost metric</u>

- 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