Dataflow Execution

Craig Knoblock
University of Southern California

This talk is based in part on slides from Greg Barish
Outline of talk

• Introduction
• Streaming dataflow execution systems
• Network Query Engines
• A streaming dataflow plan language
• Discussion
Motivation

• Problem
  • Information gathering may involve accessing and integrating data from many sources
  • Total time to execute these plans may be large

• Why?
  • Unpredictable network latencies
  • Varying remote source capabilities
  • Thus, execution is often I/O-bound

• Complicating factor: binding patterns
  • During execution, many sources cannot be queried until a previous source query has been answered
Traditional Approaches

• Executing information gathering plans
  • Generate a plan
  • Plan typically consists of a partial ordering of the operators
  • Execute the plan based on the given order
  • Operators process all of their input data before transmitting any results to consumer(s)
    • Operators as fast as their most latent input
  • Long delays due to the dependencies in the plan
Streaming Dataflow Execution Systems
Streaming Dataflow

- Plans consist of a network of operators
  - Each operator like a function
    - Example: Wrapper, Select, etc.
  - Operators produce and consume data
  - Operators “fire” when any part of any input data becomes available
  - Data routed between operators are relations
    - Zero or more tuples with one or more attributes

### Input

<table>
<thead>
<tr>
<th>City</th>
<th>State</th>
<th>Max Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Santa Monica</td>
<td>CA</td>
<td>200000</td>
</tr>
</tbody>
</table>

### Plan

- Wrapper
- Select
- Join

### Output

<table>
<thead>
<tr>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 Main St., Santa Monica, 90292</td>
</tr>
<tr>
<td>520 4th St. Santa Monica, 90292</td>
</tr>
<tr>
<td>2 Ocean Blvd, Venice, 90292</td>
</tr>
</tbody>
</table>
Dataflow vs Von-Neumann

\[((a + b) \times (c + d))\]
Parallelism of Streaming Dataflow

- **Dataflow (horizontal parallelism)**
  - Decentralized, independent operator execution
  - Enables "maximally parallel" operator execution
    - Also known as the "dataflow limit"

- **Streaming/pipelining (vertical parallelism)**
  - Producer emits tuples to consumer ASAP
    - Producer & consumer can process same relation simultaneously
  - Effective because information gathering latencies can be high – even at the tuple level
    - Data often "trickles" out of I/O-bound operators
Example: The RepInfo Agent

• INPUT
  • Any street address
e.g., 4767 Admiralty Way, Marina del Rey, CA, 90292

• OUTPUT
  • Federal reps
    • 2 senators,
    • 1 house member
  • For each rep:
    • Recent news
    • Real-time funding information
Vote-Smart:

- List of officials
Vote-Smart:
- List of officials

Yahoo
- Recent news
RepInfo Sources

Vote-Smart:
- List of officials

Yahoo
- Recent news

Open Secrets
- Funding graph
OpenSecrets – Navigation + Fetching!

Who's got the most juice on Capitol Hill? Here's a list of the top industries contributing to members of the 107th Congress during the 2001-2002 election cycle. The first list shows the overall 50 biggest industries. The other two highlight the top 25 industries giving to members of each of the two major parties. In all cases, the Top Recipient listed is the individual member of the 107th Congress who received the most from the industry. Totals shown here include only the money that went to current incumbents in Congress.

2002 Congressional Campaign Finance Profiles

<table>
<thead>
<tr>
<th>Rank</th>
<th>Industry</th>
<th>Total</th>
<th>Dem Pct</th>
<th>GOP Pct</th>
<th>Top Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lawyers/Law Firms</td>
<td>$8,383,066</td>
<td>38%</td>
<td>32%</td>
<td>Jean Carnahan (D-Mo)</td>
</tr>
<tr>
<td>2</td>
<td>Retired</td>
<td>$4,944,324</td>
<td>41%</td>
<td>59%</td>
<td>Paul Wellstone (D-Minn)</td>
</tr>
<tr>
<td>3</td>
<td>Health Professionals</td>
<td>$4,565,626</td>
<td>42%</td>
<td>57%</td>
<td>Greg Ganske (R-Iowa)</td>
</tr>
<tr>
<td>4</td>
<td>Real Estate</td>
<td>$4,146,221</td>
<td>52%</td>
<td>46%</td>
<td>Charles E. Schumer (D-NY)</td>
</tr>
<tr>
<td>5</td>
<td>Securities/Invested</td>
<td>$3,965,756</td>
<td>55%</td>
<td>45%</td>
<td>Charles E. Schumer (D-NY)</td>
</tr>
<tr>
<td>6</td>
<td>Insurance</td>
<td>$3,797,370</td>
<td>38%</td>
<td>61%</td>
<td>Max Baucus (D-Mont)</td>
</tr>
</tbody>
</table>
Politicians featured on opensecrets.org:

- Jane Harman (D-Calif)

Campaign Finance Profiles
- 2002 (Member of Congress)
- 1996 (Member of Congress)
- 1994 (Member of Congress)

Race Profiles
- 2000 Race

You can use our search engine to find more references to "Harman" on opensecrets.org.
OpenSecrets – Navigation + Fetching!

Jane Harman: 2002 Politician Profile - Microsoft Internet Explorer


**2001-2002 Profile**
- Total Raised
- Geographic Totals
  - Top Industries
  - Top Contributors
- 2001-2002 Data
  - List PAC Contributions
- Other Data
  - Personal Finances
  - 2000 Election
  - 1999-96 Profile
  - 1993-94 Profile (pdf file)
- Legislation (off-site)
- States Home
  - California Contribution Profile
- Politicians Home

**Representative (D - CA)**

**Jane Harman**

2001-2002 Total Raised: $335,117
2001-2002 Total Spent: $146,030
Cash on Hand: $229,101
First elected 2003

**Source of Funds:**

- Individual contributions: $131,990 (39.4%)
- PAC contributions: $202,985 (60.0%)
- Candidate self-financing: $0
- Other: $142 (0.0%)

**PAC Contribution Breakdown**

- Business: $139,770 (56.0%)
- Labor: $50,701 (23.9%)
- Other: $32,017 (12.7%)

Note: All the information on this page as of January 14, 2003 and based on Election Commission data released electronically on January 14, 2003. The numbers are up-to-date.
OpenSecrets – Navigation + Fetching!

JANE HARMAN (D-CA)

Contributions by Sector

PAC
- Agribusiness
- Community/Electronics
- Construction
- Defense
- Energy/Nat Resource
- Finance/Insur/RealEst
- Health
- Lawyers & Lobbyists
- Transportation
- Misc Business
- Labor
- Ideology/Single-Issue
- Other

Individual

Total Raised
- Geographic Totals
- Sector Totals
- Top Industries
- Top Contributors

HOW TO READ THIS
The chart on this page shows the member's contributions by sector. The chart breaks down the contributions into 13 main sectors:
- One for labor, one for ideological/single-issue, and one for "other." More detailed breakdowns are found in the chart that includes industries.

Historically, the financial sector has consistently been the biggest source of funding for elections. In 2000, financial interests gave over $133 billion in federal candidates. This included banks, insurance companies, and real estate industries. Accountants, lawyers, and lobbyists were the three exceptions. The catch-all "miscellaneous" category accounted for most of the money.
RepInfo agent plan

- **Wrapper** Vote-Smart
- **Select** senators, house reps
- **Wrapper** Yahoo News
- **Wrapper** OpenSecrets (names page)
- **Wrapper** OpenSecrets (member page)
- **Wrapper** OpenSecrets (funding page)
- **Join** name

- Combined results
- All officials:
  - George Bush
  - Dick Cheney
  - Barbara Boxer
  - Dianne Feinstein
  - Jane Harman
  - James Hahn

- Address
- Senators & House reps
- Recent news
- Funding URL
- Combined results
- Graph URL

- Barbara Boxer
- Dianne Feinstein
- Jane Harman

- Boxer: Anthrax investigation continues...
- Boxer: Bay area politicians meet...
- Feinstein: Bay area politicians meet...
- Harman: Life in LA is just too sunny...
Streaming Dataflow Systems for Network Environments

• Focus
  • Autonomous data sources on the Internet
  • Unpredictable network latencies

• Network Query Engines
  • Build plans to support queries
    • Tukwila
    • Telegraph
    • Niagara

• Agent-based Execution System
  • Support a richer plan language
    • Theseus
Network Query Engine -- Tukwila
Network Query Engines

- Focus on supporting streaming XML data
- Plan is defined by a query on the XML sources
  - Xquery is the emerging standard for XML querying
- Challenges
  - How to convert XML data into tuples for a streaming dataflow system
  - How to handle queries over graphs
  - How to optimize the query processing
- Here we focus on how Tukwila handles the first issue [Ives, Halevy, Weld, VLDB Journal, 2002]
Example XML Document

<db>
  <book publisher="mkp">
    <title>Readings in Database Systems</title>
    <editor>Stonebraker</editor>
    <editor>Hellerstein</editor>
    <isbn>123-456-X</isbn>
  </book>
  <book publisher="mkp">
    <title>Transaction Processing</title>
    <author>Bernstein</author>
    <author>Newcomer</author>
    <isbn>235-711-Y</isbn>
  </book>
  <company ID="mkp">
    <name>Morgan Kaufmann</name>
    <city>San Mateo</city>
    <state>CA</state>
  </company>
</db>
Graph Representation of XML
<result> { 
    POR $b IN document("books.xml")/db/book,
    $t IN $b/title/data(),
    $n IN $b/(editor|author)/data()
    RETURN <item>
        <person>{$n}</person>
        <pub>{$t}</pub>
    </item>
} </result>

<result>
    <item>
        <person>Stonebraker</person>
        <pub>Readings in Database Systems</pub>
    </item>
    <item>
        <person>Hellerstein</person>
        <pub>Readings in Database Systems</pub>
    </item>
    <item>
        <person>Bernstein</person>
        <pub>Transaction Processing</pub>
    </item>
    <item>
        <person>Newcomer</person>
        <pub>Transaction Processing</pub>
    </item>
</result>
Tukwila Architecture
Example Query

```
FOR $b IN document("books.xml")/db/book,
  $pID IN $b/@publisher,
  $t IN $b/title/data(),
  $pub IN
document("amazon.xml")(book/item,
  $t2 IN $pub/title/data(),
  $p IN $pub/source,
  $pi IN $p/@ID,
  $pr IN $pub/price/data()
WHERE $pr < 49.95
  AND $pID2 = $pID
  AND $t = $t2
RETURN  <book>
    <name>{$ t }</name>,
    <publisher>{$ p }</publisher>
  </book>
```
Query Plan
X-scan Processing

(a) X-scan process

(b) State machines for Fig. 3 query
## Operators in Tukwila

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-scan</td>
<td>streaming input</td>
<td>Match input path expression</td>
</tr>
<tr>
<td>web-join</td>
<td>streaming input</td>
<td>Query based on bound vars.</td>
</tr>
<tr>
<td>follow</td>
<td>path evaluation</td>
<td>Evaluate XPath over binding</td>
</tr>
<tr>
<td>select</td>
<td>combination/filter</td>
<td>Filter tuples by predicate</td>
</tr>
<tr>
<td>project</td>
<td>combination/filter</td>
<td>Discard bindings</td>
</tr>
<tr>
<td>hybrid hash join</td>
<td>combination/filter</td>
<td>Equijoin</td>
</tr>
<tr>
<td>pipelined hash join</td>
<td>combination/filter</td>
<td>Equijoin</td>
</tr>
<tr>
<td>merge join</td>
<td>combination/filter</td>
<td>Ordered equijoin</td>
</tr>
<tr>
<td>nested loops join</td>
<td>combination/filter</td>
<td>Order-preserving join</td>
</tr>
<tr>
<td>union</td>
<td>combination/filter</td>
<td>Relational-style union</td>
</tr>
<tr>
<td>collector</td>
<td>combination/filter</td>
<td>Union with fail-over</td>
</tr>
<tr>
<td>assign</td>
<td>combination/filter</td>
<td>Evaluate expression</td>
</tr>
<tr>
<td>distinct</td>
<td>2nd-order</td>
<td>Remove duplicates</td>
</tr>
<tr>
<td>sort</td>
<td>2nd-order</td>
<td>Reorder tuples</td>
</tr>
<tr>
<td>aggregate</td>
<td>2nd-order</td>
<td>Compute aggregate over group</td>
</tr>
<tr>
<td>nestChild</td>
<td>nesting</td>
<td>Correlated nesting of elements</td>
</tr>
<tr>
<td>group</td>
<td>nesting</td>
<td>Group and restructure sets of elements</td>
</tr>
<tr>
<td>output</td>
<td>result</td>
<td>Output binding to XML</td>
</tr>
<tr>
<td>element</td>
<td>result</td>
<td>Create XML element</td>
</tr>
<tr>
<td>attribute</td>
<td>result</td>
<td>Create XML attribute</td>
</tr>
</tbody>
</table>
Discussion

• Tukwila has
  • operators for streaming data into and out of XML
    • X-scan
    • Output, element, attribute
  • Standard relational operations
    • Select, project, join
    • Sort, aggregate, nest, group, etc.
• Focuses on the efficient processing of XML queries or streaming data sources
A Streaming Dataflow Plan Language
Theseus

• A **plan language** and **execution system** for Web-based information integration

  • Expressive enough for monitoring a variety of sources
  • Efficient enough for near-real-time monitoring

Theseus Executor

```plaintext
PLAN myplan {
  INPUT: x
  OUTPUT: y
  BODY {
    Op (x : y)
  }
}
```
Expressivity

• Basic relational-style operators
  • Select, Project, Join, Union, etc.

• Operators for gathering Web data
  • Xwrapper
    • Queries Web source via Fetch agent (returns XML)
  • Xquery, Rel2Xml, and Xml2Rel
    • XML processing utilities

• Operators for monitoring Web data
  • DbExport, DbQuery, DbAppend, DbUpdate
    • Facilitates the tracking of online data
  • Email, Phone, Fax
    • Facilitates asynchronous notification
Expressivity

• Operators for extensibility
  • **Apply**: single-row functions (e.g., UPPER)
  • **Aggregate**: multi-row functions (e.g., SUM)

• Operators for conditional plan execution
  • **Null**: Tests and routes data accordingly

• Subplans and recursion
  • Plans are named and have INPUT & OUTPUT
    • We can use them as operators (subplans) in other plans
  • Subplans make recursion possible
    • Makes it easy to follow arbitrarily long list of result pages that are each separated by a NEXT page link
  • Subplans encourage modularity & reuse
Operators

\[ \text{operator } (\text{Input}_1, \text{Input}_2, \ldots : \text{Output}_1, \text{Output}_2, \ldots) \]
\[ \text{WAIT: } \text{waitInput}_1, \text{waitInput}_2, \ldots \]
\[ \text{ENABLE: } \text{enableInput}_1, \text{enableInput}_2, \ldots \]

- **Data formats**
  - Operators pass relations
  - Relations are composed of tuples
  - Each attribute of a tuple can be primitive, relation, or XML object
• Operators support stream-oriented processing
  • Firing rule met when any input receives a tuple
    • This enables ASAP processing of data
  • End of data signaled by end-of-stream (EOS)

• Operators vary on when they can begin output:
  • Union: immediately (i.e., for each input)
  • Minus: after EOS for second input has arrived
  • Email: after EOS for all inputs have arrived
WRAPPER OPERATOR (Xwrapper)

**PURPOSE:** Extract data from web pages as relation

- **INPUT:**
  - **Name:** URL prefix of wrapper
  - **bind_map:** Wrapper binding map
  - **bind_dat:** Binding tuples

- **OUTPUT:**
  - **new_rel:** Incoming relation joined with new attributes

```plaintext
auth = USER  PASSWORD
      greg  secret

wrapper("http://fetch.com?wrapper=foo",
         "user=$user, pwd=$password", auth : quotes)

quotes = USER  PASSWORD  SYMBOL  PRICE
         greg  secret    ORCL    15.50
         greg  secret    CSCO    21.50
```
Plans and Subplans

```
plan planName
{
    input: planInput1, planInput2, ...
    output: planOutput1, planOutput2, ...
    body {
        operator (opInput1,... : opOutput1,...)
        operator ...
        ...
    }
}
```

- Plans can be called just like operators (subplans)
Example plan: TheaterLoc

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>CITY</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>187 Maxella</td>
<td>Venice</td>
<td>CA</td>
</tr>
<tr>
<td>AMC Movies</td>
<td>191 Maxella</td>
<td>Venice</td>
<td>CA</td>
</tr>
<tr>
<td>EOS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:
- WRAPPER Restaurants
- WRAPPER Theaters
- UNION
- WRAPPER Geocoder
- WRAPPER TigerMap
TheaterLoc Plan

PLAN theaterloc
{
    INPUT: city
    OUTPUT: latlons, map_url

    BODY
    {
        wrapper ("cuisinenet", "name, addr", city : restaurants)
        wrapper ("yahoo_movies", "name, addr" city : theaters)
        union (restaurants, theaters : addresses)
        wrapper ("geocoder", "name,lat,lon", addresses : latlons)
        wrapper ("tigermap", latlons : map_url)
    }
}
Transactions

- Enable
  - Concurrent plan access by multiple clients
  - Recursive plan execution
- Transactions each assigned unique ID
- Individual transactions can be aborted
- All transactions are assigned a “time to live”
  - Unprocessed data is garbage collected by Theseus
Conditionals and Recursion

• Conditional outputs are defined by enabling outputs depending on the action results

\[
\text{Null}(\text{inStream}, \text{inStreamTrue}, \text{inStreamFalse} : \\
\text{outStreamTrue}, \text{outStreamFalse})
\]

• Plans can be called recursively
  • Termination defined by conditional operators
  • Transactions support recursive calls in same execution environment
  • System provides tail-recursion optimization
New Listing:
3br 2bath 200K
Send Email Notification
Real Estate Plan

FIND_HOUSES

GET_URLS

criteria

WRAPPER
house-list

GET_URLS

WRAPPER
house-details

SELECT
(cond)

PROJECT
addr, price

Email

FORMAT
"price < %s AND beds = %s"

criteria

false

true

house results

DISTINCT
next_page_url

NULL

PROJECT
house_url

WRAPPER
house-list

GET_URLS

NULL

UNION
Parallel Remote Data Retrievals

Details Page Retrievals

Listings Page Retrievals

time (seconds)
Discussion

- Theseus, Tukwila, Telegraph, Niagara are all:
  - Streaming dataflow systems
  - Target network-based query execution
    - Large source latencies
    - Unknown characteristics of sources
  - Focus on techniques for improving the efficiency of plan execution
    - More on this in upcoming class