CSE544: Principles of Database Systems

Review of MapReduce
Datalog (cont’d)
Announcements

• Review 7 (Datalog) was due yesterday

• Project Milestone due next Friday

• HW3 is due the following Monday
MapReduce Review

• What is the map function?

• What is the reduce function?

• What is a map task?

• What is a reduce task?

• What is a mapreduce job?
MapReduce Review

• What is the **map function**?
  – Takes \((k,v)\) returns \{\((k_1,v_1)\),\((k_2,v_2)\),\ldots\}\n
• What is the **reduce function**?
  – Takes \((k,\{v_1,v_2,\ldots\})\) returns any result

• What is a **map task**?
  – A set of \((k,v)\) pairs that are scheduled as a unit

• What is a **reduce task**?
  – A set of \((k,\{v_1,\ldots\})\) pairs scheduled as a unit

• What is a **mapreduce job**?
  – The entire map reduce program
Anatomy of a Query Execution

• Running Part B of HW2

• 20 nodes = 1 master + 19 workers

• Using PARALLEL 50

• Let’s see what happened
Only 19 reducers active, out of 50. Why?

1h 16min

Some errors start to occur. Watch this…

Completed. Sorting, and the rest of Reduce may proceed now

100

1h 16min

3h 50min

Copying by 19 reducers in parallel with mappers.

How will the other 31 reducers be scheduled?
Job Cleanup:

- Started at: Sun Mar 04 19:08:29 UTC 2012
- Job-ACLs: All users are allowed
- Job-Submit Host Address: ip-10-203-30-146.ec2.internal
- User: PigLatin:DefaultJobName

### Job Counters

<table>
<thead>
<tr>
<th>Kind</th>
<th>% Complete</th>
<th>Num Tasks</th>
<th>Pending</th>
<th>Running</th>
<th>Complete</th>
<th>Killed</th>
<th>Failed/Killed Task Attempts</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>100.00%</td>
<td>15816</td>
<td>0</td>
<td>0</td>
<td>15816</td>
<td>0</td>
<td>0 / 18</td>
</tr>
<tr>
<td>reduce</td>
<td>37.72%</td>
<td>50</td>
<td>19</td>
<td>22</td>
<td>9</td>
<td>0</td>
<td>0 / 0</td>
</tr>
</tbody>
</table>

### Counter Summary

- **Bytes Read:** 5,790,488,764,416
- **Bytes Written:** 2,501,793,030
- **Bytes Spilled:** 5,587,893
- **Bytes Copied:** 199,575,247,017
- **Bytes Transferred:** 530,591,875,823
- **Spilled Records:** 10,467,550,117
- **Reduce Input Records:** 8,020,133,888
- **Reduce Output Records:** 3,008,761,856
- **Map Output Records:** 4,020,133,888
- **Split Raw Bytes:** 7,311,305,131
- **Pending:** 4,020,133,888
- **Running:** 11074
- **Killed:** 14238

### Status

- **% Complete:** 100
- **Num Tasks:** 15816
- **Pending:** 0
- **Running:** 0
- **Complete:** 15816
- **Killed:** 0
- **Failed/Killed:** 0

### Graphs

- **Completion Graph:** Some of the 19 reducers have finished...
- **Next Batch of Reducers,** Next Batch of 19 reducers
Several servers failed: “fetch error”. Their map tasks need to be rerun. All reducers are waiting….

Mappers finished, reducers resumed.

Why did we lose some reducers?
Hadoop job_201203041905_0001 on ip-10-203-30-146

User: hadoop
Job Name: PigLatin:DefaultJobName
Submit Host: ip-10-203-30-146.ec2.internal
Submit Host Address: 10.203.30.146
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Sun Mar 04 19:08:29 UTC 2012
Finished at: Mon Mar 05 02:28:39 UTC 2012
Finished in: 7hrs, 20mins, 10sec
Job Cleanup: Successful
Black-listed TaskTrackers: 3

<table>
<thead>
<tr>
<th>Kind</th>
<th>% Complete</th>
<th>Num Tasks</th>
<th>Pending</th>
<th>Running</th>
<th>Complete</th>
<th>Killed</th>
<th>Failed/Killed Task Attempts</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>100.00%</td>
<td>15816</td>
<td>0</td>
<td>0</td>
<td>15816</td>
<td>0</td>
<td>26 / 5968</td>
</tr>
<tr>
<td>reduce</td>
<td>100.00%</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>0 / 14</td>
</tr>
</tbody>
</table>

Success! 7hrs, 20mins.
Degree Sequence
Famous Example of Big Data Analysis

Kumar et al., *The Web as a Graph*

- Question 1: is the Web like a “random graph”?
- Question 2: how does the Web graph look like?
Graph as Databases

Many large databases are graphs

- The Web
- The Internet
- Social Networks
- Flights btw. Airports
- Etc, etc, etc
Data Analytics on Big Graphs

Queries expressible in SQL:
- How many nodes (edges)?
- How many nodes have > 4 neighbors?
- Which are the “most connected nodes”?

Queries requiring recursion:
- Is the graph connected?
- What is the diameter of the graph?
- Compute \textit{PageRank}
- Compute the \textit{Centrality} of each node

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
Source & Target \\
\hline
a & b \\
b & a \\
a & f \\
b & f \\
b & e \\
b & d \\
d & e \\
d & c \\
e & g \\
g & c \\
c & g \\
\hline
\end{tabular}
\end{table}
Histogram of a Graph
a.k.a. Degree Sequence

- **Outdegree** of a node = number of outgoing edges
- For each d, let n(d) = number of nodes with outdegree d
- The outdegree histogram of a graph = the scatterplot (d, n(d))

```
<table>
<thead>
<tr>
<th>d</th>
<th>n(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
```
What can you say about these graphs?

Histograms Tell Us Something About the Graph
Exponential Distribution

- $n(d) \approx c/2^d$ (generally, $cx^d$, for some $x < 1$)
- A *random graph* has exponential distribution
- Best seen when $n$ is on a log scale
Zipf Distribution

- \( n(d) \approx 1/d^x \), for some value \( x > 0 \)
- Human-generated data has Zipf distribution: letters in alphabet, words in vocabulary, etc.
- Best seen in a log-log scale
The Histogram of the Web

Late 1990’s
200M Webpages

Exponential ?
Zipf ?

Figure 2: In-degree distribution.
The Bowtie Structure of the Web

Figure 4: The web as a bowtie. SCC is a giant strongly connected component. IN consists of pages with paths to SCC, but no path from SCC. OUT consists of pages with paths from SCC, but no path to SCC. TENDRILS consists of pages that cannot surf to SCC, and which cannot be reached by surfing from SCC.
Review

• What is datalog?

• What is the naïve evaluation algorithm?

• What is the seminaive algorithm?
Discussion in Class

Assume a linear graph:

0 → 1 → 2 → 3 → ... → n

• How many iterations are needed by each algorithm?

• How many times is the tuple T(0,n) discovered by each algorithm?

Right linear TC

\[
T(x,y) :\!\!: R(x,y) \\
T(x,y) :\!\!: R(x,z), T(z,y)
\]

Non-linear TC

\[
T(x,y) :\!\!: R(x,y) \\
T(x,y) :\!\!: T(x,z), T(z,y)
\]
Discussion in Class

The *Declarative Imperative* paper:

- What are the extensions to datalog in Dedalus?
- What is the main usage of Dedalus described in the paper? Discuss some applications, discuss what’s missing.
Semantics of a Datalog Program

Three different, equivalent semantics:

• Minimal model semantics

• Least fixpoint semantics

• Proof-theoretic semantics
Minimal Model Semantics

To each rule $r$: $P(x_1...x_k) :- R_1(...), R_2(...), ...$
Minimal Model Semantics

To each rule $r$: $P(x_1...x_k) :- R_1(...), R_2(...), ...$

Associate the logical sentence $\Sigma_r$: $\forall z_1... \forall z_n. [(R_1(...) \land R_2(...) \land ...) \Rightarrow P(...)]$

All variables in the rule
Minimal Model Semantics

To each rule r: 
\[ P(x_1...x_k) : - R_1(...), R_2(...), ... \]

Associate the logical sentence \( \Sigma_r: \)
\[ \forall z_1...\forall z_n. [(R_1(...) \land R_2(...) \land ...) \implies P(...)] \]

All variables in the rule

Same as:
\[ \forall x_1...\forall x_k. [\exists y_1...\exists y_m.(R_1(...) \land R_2(...) \land ...) \implies P(...)] \]

Head variables

Existential variables
Minimal Model Semantics

To each rule $r$:  $P(x_1...x_k) :- R_1(...), R_2(...), ...$

Associate the logical sentence $\Sigma_r$:  $\forall z_1...\forall z_n. [(R_1(...)] \land R_2(...) \land ... \Rightarrow P(...)]$

Same as:  $\forall x_1...\forall x_k. [\exists y_1...\exists y_m. (R_1(...)] \land R_2(...) \land ... \Rightarrow P(...)]$

**Definition.** If $P$ is a datalog program, $\Sigma_P$ is the set of all logical sentences associated to its rules.
Minimal Model Semantics

To each rule r: $P(x_1...x_k) :- R_1(...), R_2(...), ...$

Associate the logical sentence $\Sigma_r$: $\forall z_1...\forall z_n. [(R_1(...)) \land (R_2(...)) \land ...] \implies P(...)]$

Same as: $\forall x_1...\forall x_k. [\exists y_1...\exists y_m. (R_1(...)) \land (R_2(...)) \land ...] \implies P(...)]$

**Definition.** If $P$ is a datalog program, $\Sigma_P$ is the set of all logical sentences associated to its rules.

Example. Rule: $T(x,y) :- R(x,z), T(z,y)$

Sentence: $\forall x. \forall y. \forall z. (R(x,z) \land T(z,y) \implies T(x,y))$\[\equiv \forall x. \forall y. (\exists z. R(x,z) \land T(z,y) \implies T(x,y))\]
Minimal Model Semantics

**Definition.** A pair \((I,J)\) where \(I\) is an EDB and \(J\) is an IDB is a *model* for \(P\), if \((I,J) \models \Sigma_P\).

**Definition.** Given an EDB database instance \(I\) and a datalog program \(P\), the minimal model, denoted \(J = P(I)\) is a minimal database instance \(J\) s.t. \((I,J) \models \Sigma_P\).

**Theorem.** The minimal model always exists, and is unique.
**Minimal Model Semantics**

**Definition.** A pair $(I, J)$ where $I$ is an EDB and $J$ is an IDB is a *model* for $P$, if $(I, J) \models \Sigma_P$

**Definition.** Given an EDB database instance $I$ and a datalog program $P$, the minimal model, denoted $J = P(I)$ is a minimal database instance $J$ s.t. $(I, J) \models \Sigma_P$

**Theorem.** The minimal model always exists, and is unique.

---

**Example:**

Which of these IDBs are *models*? Which are *minimal models*?

$$R = \begin{array}{|c|c|} \hline 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ \hline \end{array}$$

$$T = \begin{array}{|c|c|} \hline 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 1 & 3 \\ 2 & 4 \\ 3 & 5 \\ \hline \end{array}$$

$$T(x,y) :- R(x,y)$$

$$T(x,y) :- R(x,z), T(z,y)$$
**Minimal Model Semantics**

**Definition.** A pair \((I,J)\) where \(I\) is an EDB and \(J\) is an IDB is a *model* for \(P\), if \((I,J) \models \Sigma_P\)

**Definition.** Given an EDB database instance \(I\) and a datalog program \(P\), the minimal model, denoted \(J = P(I)\) is a minimal database instance \(J\) s.t. \((I,J) \models \Sigma_P\).

**Theorem.** The minimal model always exists, and is unique.

Example:

Which of these IDBs are *models*? Which are *minimal models*?

\[
\begin{array}{c|c}
R & T \\
\hline
1 & 2 \\
2 & 3 \\
3 & 4 \\
4 & 5 \\
\end{array}
\]

\[
\begin{array}{c|c}
T= & T(x,y) \ :- \ R(x,y) \\
\hline
1 & 2 \\
2 & 3 \\
3 & 4 \\
4 & 5 \\
\end{array}
\]

\[
\begin{array}{c|c}
T(x,y) \ :- \ R(x,z), T(z,y) \\
\hline
1 & 3 \\
2 & 4 \\
3 & 5 \\
1 & 4 \\
2 & 5 \\
1 & 5 \\
\end{array}
\]
Minimal Model Semantics

**Definition.** A pair \((I,J)\) where \(I\) is an EDB and \(J\) is an IDB is a *model* for \(P\), if \((I,J) \models \Sigma_P\)

**Definition.** Given an EDB database instance \(I\) and a datalog program \(P\), the minimal model, denoted \(J = P(I)\) is a minimal database instance \(J\) s.t. \((I,J) \models \Sigma_P\)

**Theorem.** The minimal model always exists, and is unique.

Example:

Which of these IDBs are *models*? Which are *minimal models*?

\(T(x,y) :- R(x,y)\)
\(T(x,y) :- R(x,z), T(z,y)\)

![Diagram of nodes and edges]
**Definition.** Fix an EDB $I$, and a datalog program $P$. The *immediate consequence* operator $T_P$ is defined as follows. For any IDB $J$:

$$T_P(J) = \text{all IDB facts that are immediate consequences from } I \text{ and } J.$$ 

**Fact.** For any datalog program $P$, the immediate consequence operator is monotone. In other words, if $J_1 \subseteq J_2$ then $T_P(J_1) \subseteq T_P(J_2)$. 

**Minimal Fixpoint Semantics**
Definition. Fix an EDB I, and a datalog program P. The *immediate consequence* operator $T_P$ is defined as follows. For any IDB J:

$$T_P(J) = \text{all IDB facts that are immediate consequences from I and J.}$$

Fact. For any datalog program P, the immediate consequence operator is monotone. In other words, if $J_1 \subseteq J_2$ then $T_P(J_1) \subseteq T_P(J_2)$.

Theorem. The immediate consequence operator has a unique, minimal fixpoint J: $\text{fix}(T_P) = J$, where J is the minimal instance with the property $T_P(J) = J$.

Proof: using Knaster-Tarski’s theorem for monotone functions. The fixpoint is given by:

$$\text{fix } (T_P) = J_0 \cup J_1 \cup J_2 \cup \ldots \quad \text{where } J_0 = \emptyset, \quad J_{k+1} = T_P(J_k)$$
Minimal Fixpoint Semantics

\[ R = \]

\[
\begin{array}{c|c|c}
1 & 2 & \\
2 & 3 & \\
3 & 4 & \\
4 & 5 & \\
\end{array}
\]

\[ T = \]

\[
\begin{array}{c|c|c}
1 & 2 & \\
2 & 3 & \\
3 & 4 & \\
4 & 5 & \\
\end{array}
\]

\[ J_0 = \emptyset \]

\[ J_1 = T_P(J_0) \]

\[
\begin{array}{c|c|c}
1 & 2 & \\
2 & 3 & \\
3 & 4 & \\
4 & 5 & \\
\end{array}
\]

\[ J_2 = T_P(J_1) \]

\[
\begin{array}{c|c|c}
1 & 2 & \\
2 & 3 & \\
3 & 4 & \\
4 & 5 & \\
\end{array}
\]

\[ J_3 = T_P(J_2) \]

\[
\begin{array}{c|c|c}
1 & 2 & \\
2 & 3 & \\
3 & 4 & \\
4 & 5 & \\
\end{array}
\]

\[ J_4 = T_P(J_3) \]

\[
\begin{array}{c|c|c}
1 & 2 & \\
2 & 3 & \\
3 & 4 & \\
4 & 5 & \\
\end{array}
\]

\[ T(x,y) :\neg R(x,y) \]

\[ T(x,y) :\neg R(x,z), T(z,y) \]
Proof Theoretic Semantics

Every fact in the IDB has a *derivation tree*, or *proof tree* justifying its existence.

R =

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Derivation tree of T(1,4)

T(x,y) :- R(x,y)
T(x,y) :- R(x,z), T(z,y)
Adding Negation: Datalog\textsuperscript{−}

**Example:** compute the complement of the transitive closure

\[
\begin{align*}
T(x,y) & :- R(x,y) \\
T(x,y) & :- T(x,z), R(z,y) \\
CT(x,y) & :- \text{Node}(x), \text{Node}(y), \text{not} \ T(x,y)
\end{align*}
\]

What does this mean??
Recursion and Negation
Don’t Like Each Other

EDB: \( I = \{ R(a) \} \)

\[
S(x) :- R(x), \text{ not } T(x) \\
T(x) :- R(x), \text{ not } S(x)
\]

Which IDBs are models of \( P \)?

\[ J_1 = \{ \} \quad J_2 = \{ S(a) \} \quad J_3 = \{ T(a) \} \quad J_4 = \{ S(a), T(a) \} \]
Recursion and Negation
Don’t Like Each Other

EDB: \( I = \{ R(a) \} \)

\[
\begin{align*}
S(x) & : - R(x), \text{not } T(x) \\
T(x) & : - R(x), \text{not } S(x)
\end{align*}
\]

Which IDBs are models of \( P \)?

\( J_1 = \{ \} \)  
\( J_2 = \{ S(a) \} \)  
\( J_3 = \{ T(a) \} \)  
\( J_4 = \{ S(a), T(a) \} \)

No: both rules fail
Yes: the facts in \( J_2 \) are \( R(a), S(a), \text{not } T(a) \)
and both rules are true.
Yes
Yes

There is no \textit{minimal} model!
Recursion and Negation

Don’t Like Each Other

EDB: \( I = \{ R(a) \} \)

\[
\begin{align*}
S(x) & :\!- R(x), \text{ not } T(x) \\
T(x) & :\!- R(x), \text{ not } S(x)
\end{align*}
\]

Which IDBs are models of \( P \)?

\( J_1 = \{ \} \)

\( J_2 = \{ S(a) \} \)

\( J_3 = \{ T(a) \} \)

\( J_4 = \{ S(a), T(a) \} \)

No: both rules fail

Yes: the facts in \( J_2 \) are \( R(a), S(a), \neg T(a) \) and both rules are true.

Yes

Yes

There is no \textit{minimal} model!

There is no minimal fixpoint! (Why does Knaster-Tarski’s theorem fail?)
Adding Negation: $\text{datalog}^-$

- **Solution 1: Stratified Datalog$^-$**
  - Insist that the program be *stratified*: rules are partitioned into strata, and an IDB predicate that occurs only in strata $\leq k$ may be negated in strata $\geq k+1$

- **Solution 2: Inflationary-fixpoint Datalog$^-$**
  - Compute the fixpoint of $J \cup T_P(J)$
  - Always terminates (why?)

- **Solution 3: Partial-fixpoint Datalog$^-$**
  - Compute the fixpoint of $T_P(J)$
  - May not terminate
A datalog⁻ program is *stratified* if its rules can be partitioned into $k$ strata, such that:

- If an IDB predicate $P$ appears negated in a rule in stratum $i$, then it can only appear in the head of a rule in strata 1, 2, ..., $i-1$.

Note: a datalog⁻ program either is stratified or it ain’t!

Which programs are stratified?

- $T(x,y) :- R(x,y)$
- $T(x,y) :- T(x,z), R(z,y)$
- $CT(x,y) :- Node(x), Node(y), \text{not } T(x,y)$
- $S(x) :- R(x), \text{not } T(x)$
- $T(x) :- R(x), \text{not } S(x)$
Stratified datalog\(^{-}\)

- Evaluation algorithm for stratified datalog\(^{-}\):

- For each stratum \(i = 1, 2, \ldots\), do:
  
  - Treat all IDB’s defined in prior strata as EBS
  
  - Evaluate the IDB’s defined in stratum \(i\), using either the naïve or the semi-naïve algorithm

Does this compute a minimal model?

\[
\begin{align*}
T(x, y) &: R(x, y) \\
T(x, y) &: T(x, z), R(z, y) \\
CT(x, y) &: \text{Node}(x), \text{Node}(y), \text{not} T(x, y)
\end{align*}
\]
Stratified datalog

• Evaluation algorithm for stratified datalog: 
  
• For each stratum $i = 1, 2, \ldots$, do: 
  
  – Treat all IDB’s defined in prior strata as EBS
  – Evaluate the IDB’s defined in stratum $i$, using either the naïve or the semi-naïve algorithm

Does this compute a minimal model?

NO:
$J_1 = \{ T = \text{transitive closure}, \ CT = \text{its complement} \}$
$J_2 = \{ T = \text{all pairs of nodes}, \ CT = \text{empty} \}$

T(x, y) :- R(x, y)
T(x, y) :- T(x, z), R(z, y)
CT(x, y) :- Node(x), Node(y), not T(x, y)
Inflationary-fixpoint datalog\(^{-}\)

Let \( P \) be any datalog\(^{-}\) program, and \( I \) an EDB.
Let \( T_P(J) \) be the *immediate consequence* operator.
Let \( F(J) = J \cup T_P(J) \) be the *inflationary immediate consequence* operator.

Define the sequence: \( J_0 = \emptyset, J_{n+1} = F(J_n) \), for \( n \geq 0 \).

**Definition.** The inflationary fixpoint semantics of \( P \) is \( J = J_n \) where \( n \) is such that \( J_{n+1} = J_n \)

Why does there always exists an \( n \) such that \( J_n = F(J_n) \)?

Find the inflationary semantics for:

\[
\begin{align*}
T(x,y) & : - R(x,y) \\
T(x,y) & : - T(x,z), R(z,y) \\
CT(x,y) & : - \text{Node}(x), \text{Node}(y), \text{not} \ T(x,y) \\
\end{align*}
\]

\[
\begin{align*}
S(x) & : - R(x), \text{not} \ T(x) \\
T(x) & : - R(x), \text{not} \ S(x) \\
\end{align*}
\]
Inflationary-fixpoint datalog

- Evaluation for Inflationary-fixpoint datalog

- Use the naïve, of the semi-naïve algorithm

- Inhibit any optimization that rely on monotonicity (e.g. out of order execution)
Partial-fixpoint datalog\(^-\),*

Let \( P \) be any datalog\(^-\) program, and \( I \) an EDB.
Let \( T_P(J) \) be the *immediate consequence* operator.

Define the sequence: \( J_0 = \emptyset \), \( J_{n+1} = T_P(J_n) \), for \( n \geq 0 \).

**Definition.** The partial fixpoint semantics of \( P \) is \( J = J_n \) where \( n \) is such that \( J_{n+1} = J_n \), if such an \( n \) exists, undefined otherwise.

Find the partial fixpoint semantics for:

- \( T(x,y) :\!-\! R(x,y) \)
- \( T(x,y) :\!-\! T(x,z), R(z,y) \)
- \( CT(x,y) :\!-\! Node(x), Node(y), \text{not } T(x,y) \)

Note: there may not exists an \( n \) such that \( J_n = F(J_n) \)
Discussion

• Which semantics does Daedalus adopt?
Discussion

Comparing datalog\(^{-}\)

• Compute the complement of the transitive closure in inflationary datalog\(^{-}\)

• Compare the expressive power of:
  – Stratified datalog\(^{-}\)
  – Inflationary fixpoint datalog\(^{-}\)
  – Partial fixpoint datalog\(^{-}\)