Streaming Big Data with Self-Adjusting Computation

Umut A. Acar
CMU

Yan Chen
MPI-SWS
Motivation

Incoming
Stream Processing

Time of arrival

Time reflected

Response time
Stream Processing

Time of arrival

Immediate

Time reflected
Challenge

That's a huge gap.

Big data

Incremental Big data

©Gabriel Laposi 06-12-07
## Incremental Big Data

<table>
<thead>
<tr>
<th>System</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percolator [OSDI’10]</td>
<td>General purpose</td>
<td>Hand-crafted update algorithms</td>
</tr>
<tr>
<td>Incoop [SoCC’11]</td>
<td>Transparent</td>
<td>MapReduce</td>
</tr>
<tr>
<td>Naiad [CIDR’13]</td>
<td>Transparent</td>
<td>Flat data parallel</td>
</tr>
</tbody>
</table>
Challenge

Self-Adjusting Computation

Incremental Big data

That's a huge gap.

Big data
MapReduce: Map-Shuffle-Reduce

mapper: in seq $\rightarrow$ KV.t seq

reducer: KV.key*KV.val seq $\rightarrow$ out

out seq
MapReduce Example: Wordcount

mapper

(a,1)(c,1)(c,1)(a,1)(c,1)
(b,1)(c,1)(b,1)(c,1)

reducer

(a,[1 1]) (b,[1 1]) (c,[1 1 1 1 1])

(a,2) (b,2) (c,5)
Incremental MapReduce

mapper

reducer

out seq

in seq
Level Types

• type int = int

• datatype list = nil |
  cons of int * list

• datatype α seq = nil |
  cons of α * α seq
Type Signature

mapper: in seq $\rightarrow$ KV.t seq $\rightarrow$

reducer: KV.key*KV.val seq $\rightarrow$ out

out seq $\rightarrow$
Implicit Self-Adjusting Computation

[ICFP’11, PLDI’12]
Benchmarks & Evaluation

• Prototype Implementation
  – Single machine
  – Sequential

• Benchmark
  – MapReduce
    • Wordcount, Pagerank
  – Sparse matrix vector multiply
    • Pagerank, Graph connectivity
Benchmark: Wordcount

- Dataset: Wikipedia (1M, 8000 lines)
  - Hadoop: >13 hours
Non-incremental MapReduce
Incremental MapReduce
Words per Line

![Graph showing the number of words per line](image-url)
Comparison

<table>
<thead>
<tr>
<th></th>
<th>Experiment</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-incremental</strong></td>
<td>Hadoop</td>
<td>&gt; 13 hours</td>
</tr>
<tr>
<td></td>
<td>ML</td>
<td>12.5 hours</td>
</tr>
<tr>
<td><strong>Incremental</strong></td>
<td>ML</td>
<td>1 min 40 sec</td>
</tr>
<tr>
<td></td>
<td>Perl</td>
<td>34 sec</td>
</tr>
</tbody>
</table>
Challenge: Memory Usage

• Non-incremental MapReduce: 350M
• Incremental MapReduce: 16G
Work in Progress

• Granularity Control
  – Partition into large block
    • Wordcount: memory 16G $\rightarrow$ 2G
  – Traceable Data Type [PLDI’10]

• More Benchmarks
  – Pagerank
  – Graph Algorithms

• Parallelism