Inspector Gadget: A Framework for Custom Monitoring and Debugging of Distributed Dataflows

Christopher Olston and Benjamin Reed
Yahoo! Research
Web Scale problems

• Lots of servers, users, and data
• Fun to have power at your fingertip
• Sucks when things go wrong
Map/Reduce

Input Dataset

Per record Processing & Partitioning

Map

Map

Map

Output Dataset

Per Partition Processing

Reduce

Reduce

Reduce
Example Pig Workflow

Pages = load 'webpages'
UserViews = load 'userclicks'
NerdPages = filter Pages by NerdFilter(content)
NerdPageViews = join NerdPages, UserViews by url
NerdUsers = group NerdPageViews by user
Counts = foreach NerdUsers generate user, COUNT(NerdPageViews)
store Counts into 'nerdviewcounts'
Motivated by User Interviews

Interviewed 10 Yahoo dataflow programmers (mostly Pig users; some users of other dataflow environments)
Asked them how they (wish they could) debug
<table>
<thead>
<tr>
<th># of requests</th>
<th>feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>crash culprit determination</td>
</tr>
<tr>
<td>5</td>
<td>row-level integrity alerts</td>
</tr>
<tr>
<td>4</td>
<td>table-level integrity alerts</td>
</tr>
<tr>
<td>4</td>
<td>data samples</td>
</tr>
<tr>
<td>3</td>
<td>data summaries</td>
</tr>
<tr>
<td>3</td>
<td>memory use monitoring</td>
</tr>
<tr>
<td>3</td>
<td>backward tracing (provenance)</td>
</tr>
<tr>
<td>2</td>
<td>forward tracing</td>
</tr>
<tr>
<td>2</td>
<td>golden data/logic testing</td>
</tr>
<tr>
<td>2</td>
<td>step-through debugging</td>
</tr>
<tr>
<td>2</td>
<td>latency alerts</td>
</tr>
<tr>
<td>1</td>
<td>latency profiling</td>
</tr>
<tr>
<td>1</td>
<td>overhead profiling</td>
</tr>
<tr>
<td>1</td>
<td>trial runs</td>
</tr>
</tbody>
</table>
Running Pig
Running Pig
Running Pig
Running Pig

Detective

Pig

Error!
Running Pig
Our Approach

**Goal:** a programming framework for adding debugging features to Pig

**Precept:** avoid modifying Pig or tampering with data flowing through Pig

**Approach:** perform Pig script rewriting – insert special (User Defined Functions) UDFs that look like no-ops to Pig
Pig w/ Inspector Gadget

IG coordinator

load
IG agent
filter
IG agent
join
IG agent
group
IG agent
count
IG agent
store
Row Integrity

- **IG coordinator**
- **IG agent**
- **load**
- **filter**
- **join**
- **group**
- **count**
- **store**

Bad records flow from **IG coordinator** to **IG agent**.
Example: Forward Tracing

load

filter

join

IG agent

IG agent

IG agent

IG agent

store

count

group

IG agent

IG agent

IG agent

IG coordinator

report traced records to user

traced records

tracing instructions

load
Example: 
*Crash Culprit Determination* 

```
IG coordinator

- load
  - IG agent
  - filter
  - IG agent
  - join
    - IG agent
    - group
    - IG agent
    - count
      - IG agent
      - store
```
Crash Culprit Sending every 5th IG coordinator
Crash Culprit Sending every 5th
Crash Culprit sending every 5th
Crash Culprit Sending 5th
Crash Culprit Sending every 2nd

IG coordinator
Crash Culprit Sending every 2nd
Crash Culprit Sending every tuple
Crash Culprit Sending every tuple
Agent & Coordinator APIs

### Agent Class
- `init(args)`
- `tags = observeRecord(record, tags)`
- `receiveMessage(source, message)`
- `finish()`

### Coordinator Class
- `init(args)`
- `receiveMessage(source, message)`
- `output = finish()`

### Agent Messaging
- `sendToCoordinator(message)`
- `sendToAgent(agentId, message)`
- `sendDownstream(message)`
- `sendUpstream(message)`

### Coordinator Messaging
- `sendToAgent(agentId, message)`
# Applications Developed Using IG

<table>
<thead>
<tr>
<th># of requests</th>
<th>feature</th>
<th>lines of code (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>crash culprit determination</td>
<td>141</td>
</tr>
<tr>
<td>5</td>
<td>row-level integrity alerts</td>
<td>89</td>
</tr>
<tr>
<td>4</td>
<td>table-level integrity alerts</td>
<td>99</td>
</tr>
<tr>
<td>4</td>
<td>data samples</td>
<td>97</td>
</tr>
<tr>
<td>3</td>
<td>data summaries</td>
<td>130</td>
</tr>
<tr>
<td>3</td>
<td>memory use monitoring</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>backward tracing (provenance)</td>
<td>237</td>
</tr>
<tr>
<td>2</td>
<td>forward tracing</td>
<td>114</td>
</tr>
<tr>
<td>2</td>
<td>golden data/logic testing</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>step-through debugging</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>latency alerts</td>
<td>168</td>
</tr>
<tr>
<td>1</td>
<td>latency profiling</td>
<td>136</td>
</tr>
<tr>
<td>1</td>
<td>overhead profiling</td>
<td>124</td>
</tr>
<tr>
<td>1</td>
<td>trial runs</td>
<td>93</td>
</tr>
</tbody>
</table>
In Paper

Semantics under parallel/distributed execution
Messaging & tagging implementation
Limitations
Performance experiments
Related work
Performance Experiments

15-machine Pig/Hadoop cluster (1G network)
Four dataflows over a small web crawl sample (10M URLs):

<table>
<thead>
<tr>
<th>Dataflow Program</th>
<th>Early Projection Optimization</th>
<th>Early Aggregation Optimization</th>
<th>Number of Map-Reduce Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinct Inlinks</td>
<td>N</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>Frequent Anchortext</td>
<td>Y</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>Big Site Count</td>
<td>Y</td>
<td>Y</td>
<td>1</td>
</tr>
<tr>
<td>Linked By Large</td>
<td>N</td>
<td>Y</td>
<td>2</td>
</tr>
</tbody>
</table>
Dataflow Running Times

The chart shows the running times for different dataflow scripts. The x-axis represents different script categories: Distinct Inlinks, Frequent Anchor Text, Big Site Count, and Linked by Large. The y-axis represents running time in seconds. The lines and bars indicate the performance of various algorithms, with colors representing different dataflow implementations.
Related Work

XTrace, etc.
taint tracking
aspect-oriented programming
Summary / Status

- Users have a long wish-list for “debuggability”
  - Make a general framework rather than tool for each
  - Addressed most features with few lines of code
- Rather than implement them as separate features in the Pig core, we built a layer on top
- IG (called Penny) is open source. Accepted into Apache Pig v0.9 release (http://pig.apache.org)
The End