Semantic Compression & Light-weight Disc Models

Adam Crume, Carlos Maltzahn

University of California, Santa Cruz

October 23, 2012
Outline

1. Semantic Compression
   - Hadoop intermediate data
   - Byte-level
   - Logical-level

2. Lightweight Disc Models
   - Background
   - Fitting
   - Implementation
Hadoop internal data flow

Mapper → Combiner → Disk → network transfer → Disk → Sort → Reducer
MapReduce Job CDF

- Mappers for 22 Reducers (SH)
- Mappers for 22 Reducers (SS)
- 22 Reducers (SH)
- 22 Reducers (SS)
Linear sequences

00 00 00 00 00 00 00 00 00 00 00 0a 77 69 6e |.............win|
64 73 70 65 64 31 00 00 00 01 00 00 01 6c 00 |dspeed1........l|
88 6c 80 01 01 1f 0e 00 00 00 01 00 00 00 8c 00 |.l..............|
00 00 00 00 00 00 00 00 00 00 01 0a 77 69 6e 64 |............wind|
73 70 65 64 31 00 00 00 01 00 00 00 01 6c 88 |speed1........l.|
6c 8a 01 01 1f 0e 00 00 00 01 00 00 00 8c 00 00 |l..............|
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |..............|
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |..............|
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |..............|
65 65 64 31 00 00 00 01 00 00 01 6c 88 6c 9e |eed1........l.l|
01 01 1f 0e 00 00 00 00 04 00 00 00 8c 00 00 00 |..............|
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |..............|
0a 77 69 6e 64 73 70 65 64 31 00 00 00 01 6c |...........winds|
### Predictive coding

<table>
<thead>
<tr>
<th>Keys:</th>
<th>(1,1)</th>
<th>(1,2)</th>
<th>(1,3)</th>
<th>(1,4)</th>
<th>(1,5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(2,1)</td>
<td>(2,2)</td>
<td>(2,3)</td>
<td>(2,4)</td>
<td>(2,5)</td>
</tr>
</tbody>
</table>

| Original: | 1   | 1   | 1   | 2   | 1   | 3   | 1   | 4   | 1   | 5   | 2   | 1   |
| Predictions: | 1   | 4   | 1   | 5   | 1   | 6   |
| Delta (output): | 1   | 1   | 1   | 2   | 1   | 3   | 0   | 0   | 0   | 0   | 0   | 1   | -7   |
Semantically-informed byte-level compression (results)

File size by compression method

- Original: 100%
- gzip: 13.6%
- transform+gzip: 0.28%
- bzip2: 4.27%
- transform+bzip2: 0.0039%
Key redundancy
N-dimensional aggregation

Optimal choice is not obvious
Space-filling curves

Cells are numbered with a space-filling curve, and contiguous numbers are collapsed into ranges.

1–5, 7, 9–10, 13

1–5, 8-9, 14–15
Key splitting

Overlapping ranges are split on the overlap boundaries
Still in progress, but compared to SciHadoop with \textit{no} aggregation, SciHadoop \textit{with} aggregation yields:

- 80\% reduction in intermediate data
- 60\% reduction in runtime
Both approaches described in:
Outline

1. Semantic Compression
   - Hadoop intermediate data
   - Byte-level
   - Logical-level

2. Lightweight Disc Models
   - Background
   - Fitting
   - Implementation
Introduction

- Cluster simulation is important for architecture design
- Storage is expensive to simulate
- Willing to trade some accuracy for performance
- Analytical models can be difficult to find
- Need models for hard drives, SSDs, RAID arrays, etc.
- Our work directly models any block device
Contribution

Our work fits an analytic model to device traces with genetic programming. After training, the model is used at high speed in simulation.
1. Population initialized with random models
2. Each model evaluated by comparing to the actual device
3. Poor models are discarded, good models are mated and mutated

\[ 2x \text{ error 10\%} \quad x + 1 \text{ error 20\%} \quad x^2 \text{ error 50\%} \]

\[ 2x \text{ error 10\%} \quad x + 1 \text{ error 20\%} \quad 2x + 1.1 \text{ error 5\%} \]

4. Go to step 2, repeat until error is low
Computing error/fitness

- Multiple plausible error metrics available
- Metric determines the output model
- Per-request accuracy would be ideal, but seems impossible
- Currently using mean relative demerit, as defined by Ruemmler
- May factor in model speed, to penalize slow models
Computing error/fitness: Demerit
Execution

- Requests fed through sequentially
- Results available immediately
- Calculation is fast
- State is only a few bytes

\[
\begin{align*}
\text{request}_0 & \rightarrow \text{model} & \rightarrow \text{state}_0 & \rightarrow \text{time}_0 \\
\text{request}_1 & \rightarrow \text{model} & \rightarrow \text{state}_1 & \rightarrow \text{time}_1 \\
\text{request}_2 & \rightarrow \text{model} & \rightarrow \text{state}_2 & \rightarrow \text{time}_2 \cdots
\end{align*}
\]
Models are arrays of variables
- Each variable is computed by an expression tree
- State is kept by referencing old variables
- The last variable gives the predicted access time

```
var 0:  var 1:
  *       +
    -     old var 1
  0.8     var 0

start block  old start block
```
Results

- Model is 300x faster than DiskSim
- Time to read one million events is 7.5 seconds, versus time to simulate them is 0.25 seconds

![Access time for small random reads](chart.png)
Questions?
References

Adam Crume, Joe Buck, Carlos Maltzahn, and Scott Brandt.
Compressing intermediate keys between mappers and reducers in sciadopop.

C. Ruemmler and J. Wilkes.
An introduction to disk drive modeling.