Succinct
Fast Interactive Queries via “Homomorphic” Compression
Rachit Agarwal, Anurag Khandelwal, Ion Stoica

Succinct in a Nutshell
A distributed data store that supports:
- Queries on a compressed representation of input
- Read-optimized, append-only updates
- Low memory (input in compressed form; no secondary indexes)
- Low Latency (no scans, no decompression)
- Limitations: CPU (data access), lower scan throughput, no in-place updates

Succinct, natively...
- Supports queries on flat files
- Easily extensible to support queries on semi-structured data (key-value pairs, tables, etc.)
- Three basic queries: extract(offset, len), count(string str), search(string str)
- Extended to support range, wildcard and regular expression queries

Succinct for Semi-structured data
- Logical collection of records transformed to flat files
- Problem A simplified problem:
  - Searching on a compressed representation of input data
  - Important operators
    - extract(string)
    - count(string)
    - search(string)

  Example: Extract suffixes from input data

  Problem: Total space to store the locations is nlogn.

  Step 1: Extract suffixes from input data.
  Step 2: Sort suffixes lexicographically.
  Step 3: Obtain locations of suffixes in the input data:

  Problem: How to lookup unsampled values?

  Step 4: Store sampled values only.

  Problem: Successor array still takes up nlogn space.

  Step 5: Maintain pointers to index of successor.

  Step 6: Compress the increasing sequences.

  - More complex operations to convert these values into contiguous sequences, which are more compressible.

Technique Overview

Preliminary Results

Succinct requires 11x lower memory than systems with similar functionality.

When datasets fit in memory, Succinct achieves comparable performance using 11x lower memory.

When datasets do not fit in memory, Succinct achieves significantly lower latency.

Preliminary evaluation suggests that RegEx queries using Succinct can achieve lower latency than existing systems, while using 4–7x lower memory.

Related Work

- Traditional Compression: GZip, LZ, Snappy, etc.
- Sampling Data: BlinkDB, for approximate computations in data analytics
- Column-Stores: Use of opportunistic compression
- Full-Text Indexes: Compressed Suffix Arrays, FM-Index, etc.

(anuragk, ragarwal, istoica)@eecs.berkeley.edu