COMP9319 Web Data Compression and Search

Web data compression & search in industry - case studies

Bigtable

From Wikipedia, the free encyclopedia

Bigtable is a compressed, high performance, proprietary data storage system built on Google File System, Chubby Lock Service, SSTable (log-structured storage like LevelDB) and a few other Google technologies. On May 6, 2015, a public version of Bigtable was made available as a service. Bigtable also underlies Google Cloud Datastore, which is available as a part of the Google Cloud Platform.[1][2]

Applications

- Storage system used by
  - Web indexing
  - MapReduce
  - Google App Engine
  - Google Cloud Datastore
  - and many many more…

Google's Motivation – Scale!

- Scale Problem
  - Lots of data
  - Millions of machines
  - Different project/applications
  - Hundreds of millions of users
- Storage for (semi-)structured data
- No commercial system big enough
  - Couldn't afford if there was one
- Low-level storage optimization help performance significantly
  - Much harder to do when running on top of a database layer

Bigtable

- Distributed multi-level map
- Fault-tolerant, persistent
- Scalable
  - Thousands of servers
  - Terabytes of in-memory data
  - Petabyte of disk-based data
  - Millions of reads/writes per second, efficient scans
- Self-managing
  - Servers can be added/removed dynamically
  - Servers adjust to load imbalance

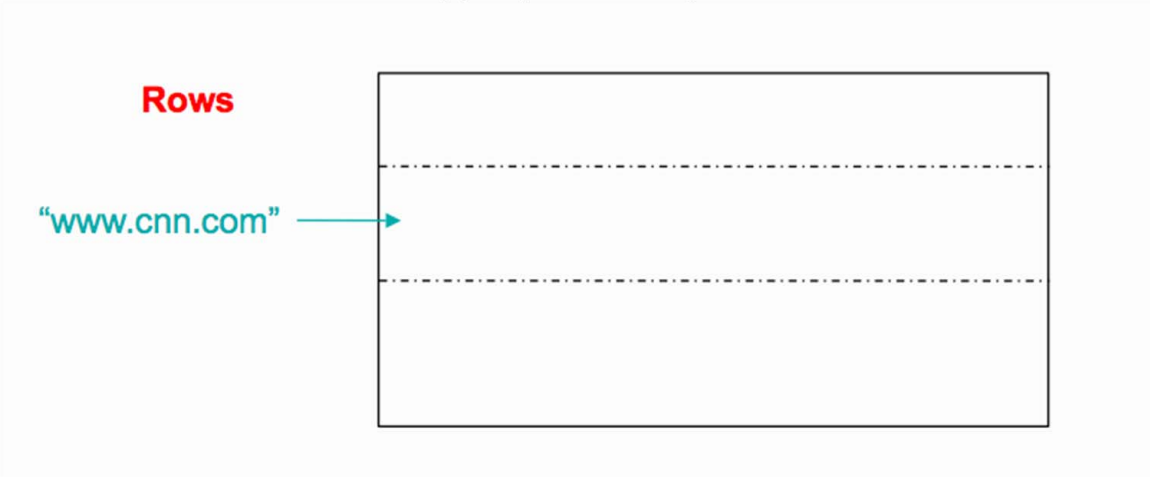
Data Model

- a sparse, distributed persistent multi-dimensional sorted map

(row, column, timestamp) -> cell contents

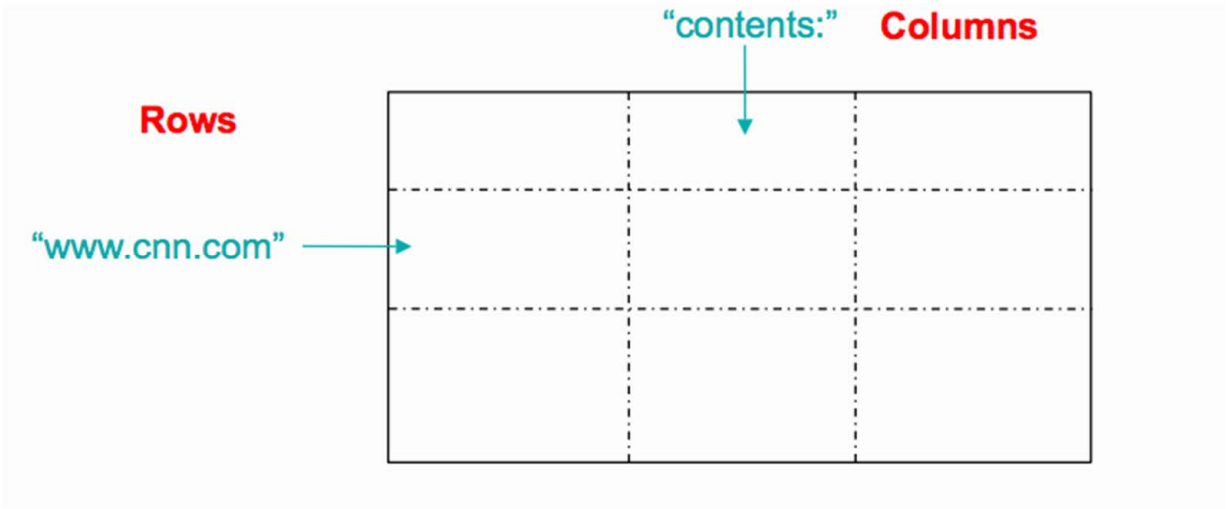
Data Model

- Rows
  - Arbitrary string
  - Access to data in a row is atomic
  - Ordered lexicographically



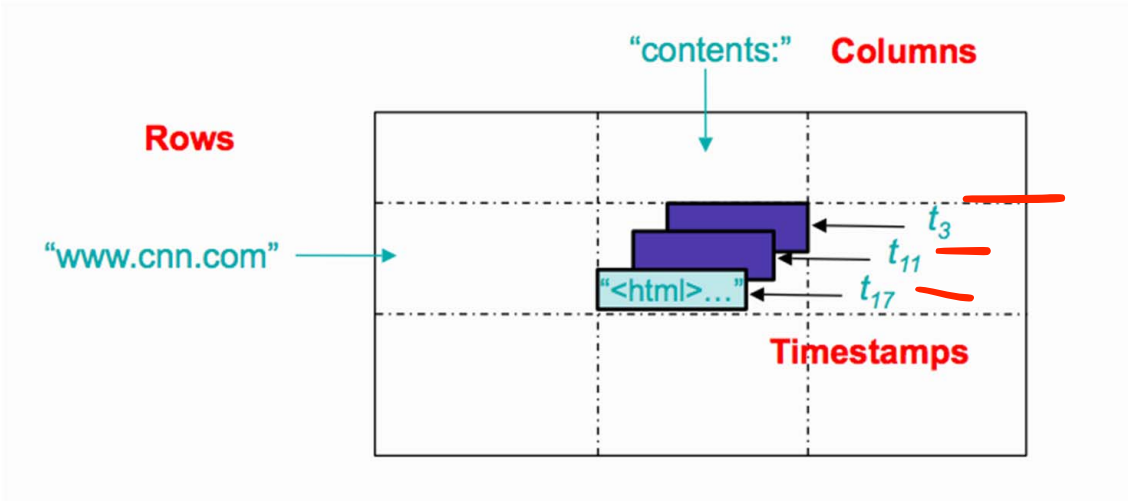
Data Model

- Column
  - Name structure:
    - family: qualifier
  - Column Family is the unit of access control



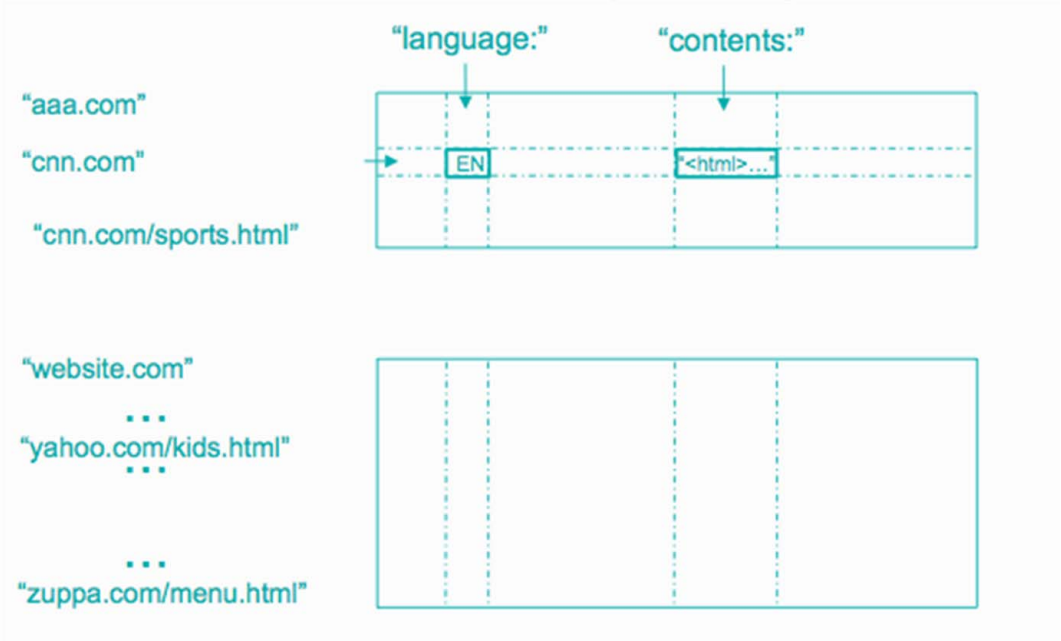
Data Model

- Timestamps
  - Store different versions of data in a cell
  - Lookup options
    - Return most recent K values
    - Return all values



Data Model

- The row range for a table is dynamically partitioned
- Each row range is called a tablet
- Tablet is the unit for distribution and load balancing



Slide from Jeff Dean, Google:

Compression

- Many opportunities for compression
  - Similar values in the same row/column at different timestamps
  - Similar values in different columns
  - Similar values across adjacent rows
- Within each SSTable for a locality group, encode compressed blocks
  - Keep blocks small for random access (~64KB compressed data)
  - Exploit fact that many values very similar
  - Needs to be low CPU cost for encoding/decoding
- Two building blocks: BMDiff, Zippy

Now called Snappy, LZW-like, 16K entry table, compress less but faster

Based on DCC'99: Data Compression Using Long Common Strings

Google

Slide from Jeff Dean, Google:

Compression Effectiveness

- Experiment: store contents for 2.1B page crawl in BigTable instance
  - Key: URL of pages, with host-name portion reversed
    - `com.cnn.www/index.html:http`
  - Groups pages from same site together
    - Good for compression (neighboring rows tend to have similar contents)
    - Good for clients: efficient to scan over all pages on a web site

- One compression strategy: gzip each page: ~28% bytes remaining
- BigTable: BMDiff + Zippy:

Type	Count (B)	Space (TB)	Compressed	% remaining
Web page contents	2.1	45.1 TB	4.2 TB	9.2%
Links	1.8	11.2 TB	1.6 TB	13.9%
Anchors	126.3	22.8 TB	2.9 TB	12.7%

Google

Real Applications

% remaining

Project name	Table size (TB)	Compression ratio	# Cells (billions)	# Column Families	# Locality Groups	% in memory	Latency-sensitive?
Crawl	800	11%	1000	16	8	0%	No
Crawl	50	33%	200	2	2	0%	No
Google Analytics	20	29%	10	1	1	0%	Yes
Google Analytics	200	14%	80	1	1	0%	Yes
Google Base	2	31%	10	29	3	15%	Yes
Google Earth	0.5	64%	8	7	2	33%	Yes
Google Earth	70	–	9	8	3	0%	No
Orkut	9	–	0.9	8	5	1%	Yes
Personalized Search	4	47%	6	93	11	5%	Yes

DATA OPTIMIZATION ON CLOUD

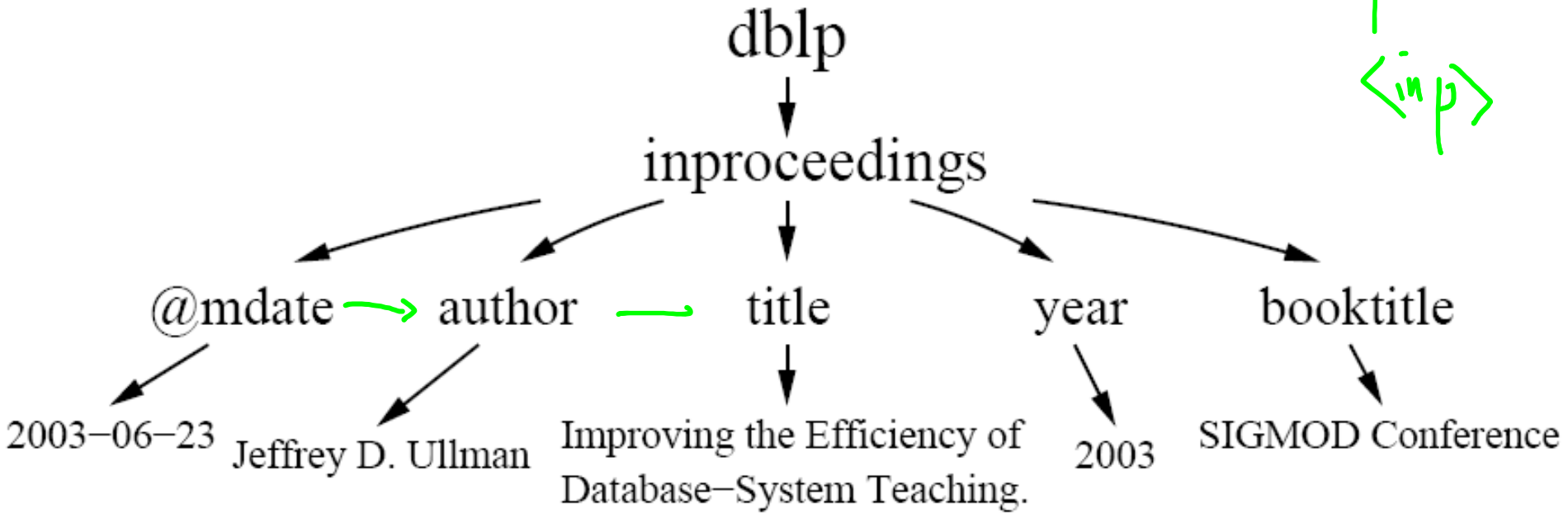
JSON

```
{
  "glossary": {
     "title": "example glossary”, "GlossDiv": {
        "title": "S","GlossList": {
           "GlossEntry": {
             "ID": "SGML",
             "SortAs": "SGML",
             "GlossTerm": "Standard Generalized Markup Language",
             "Acronym": "SGML",
             "Abbrev": "ISO 8879:1986",
             "GlossDef": {
                  "para": "A meta-markup language, used to create marku
languages such as DocBook.",
                 "GlossSeeAlso": ["GML", "XML"]
              },
              "GlossSee": "markup"
            }
         }
      }
   }
}
```

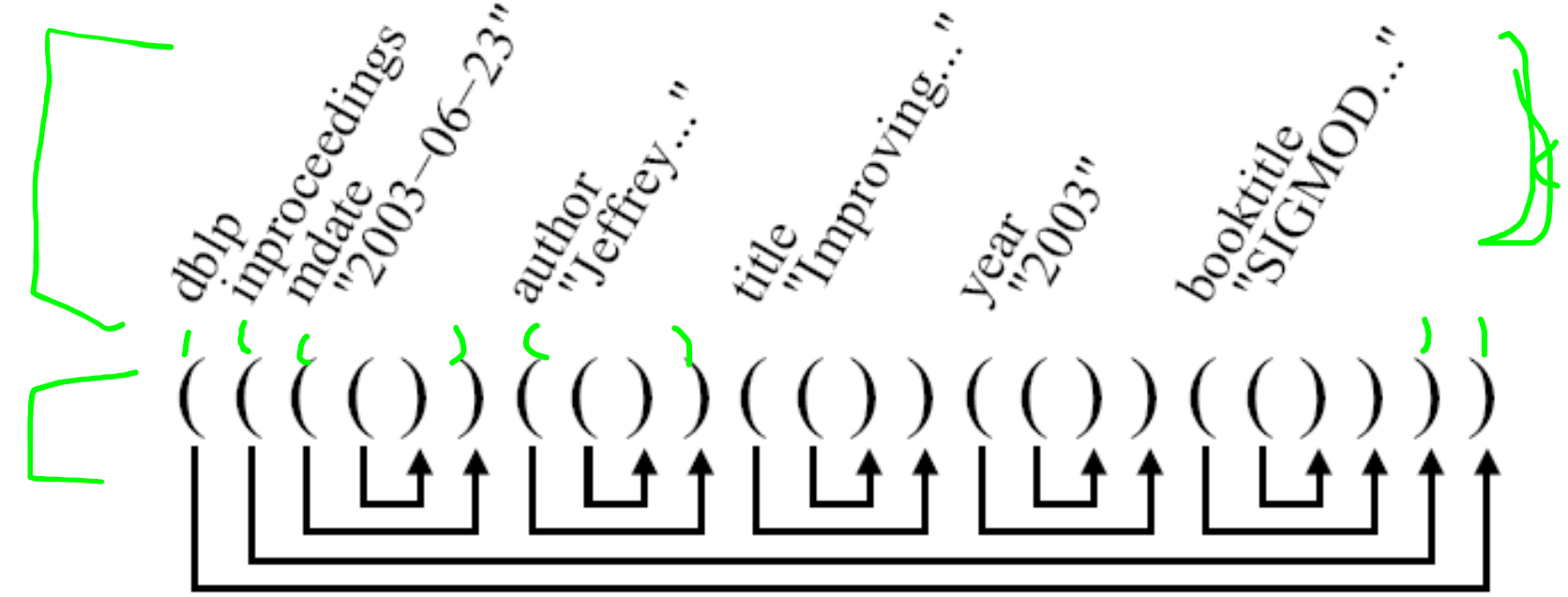
XML

```
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
 <glossary><title>example glossary</title>
  <GlossDiv><title>S</title>
   <GlossList>
    <GlossEntry ID="SGML" SortAs="SGML">
     <GlossTerm>Standard Generalized Markup Language</GlossTerm>
     <Acronym>SGML</Acronym>
     <Abbrev>ISO 8879:1986</Abbrev>
     <GlossDef>
      <para>A meta-markup language, used to create markup
languages such as DocBook.</para>
      <GlossSeeAlso OtherTerm="GML">
      <GlossSeeAlso OtherTerm="XML">
     </GlossDef>
     <GlossSee OtherTerm="markup">
    </GlossEntry>
   </GlossList>
  </GlossDiv>
 </glossary>
```

Sample DBLP XML Fragment

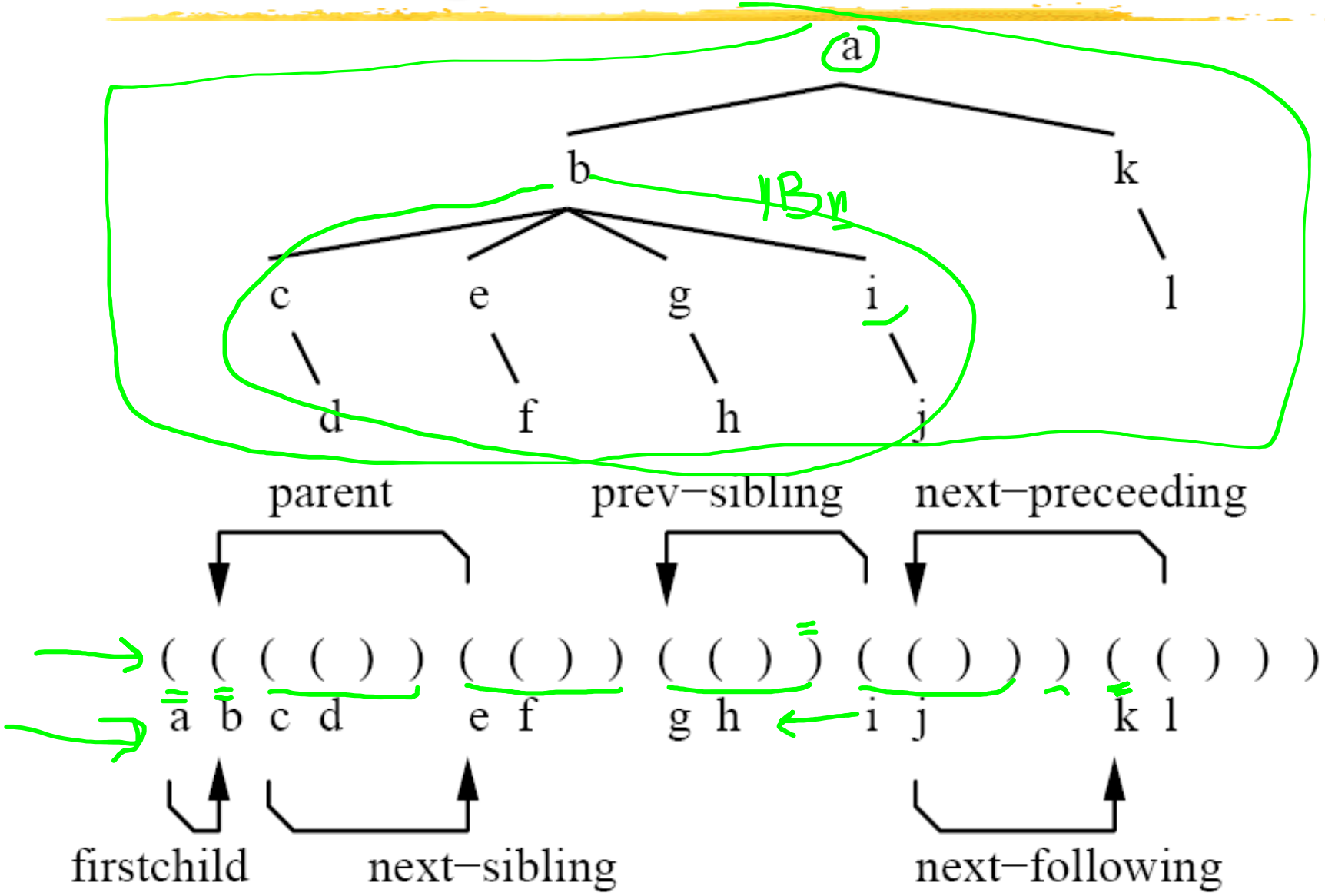


Balanced Parenthesis Encoding

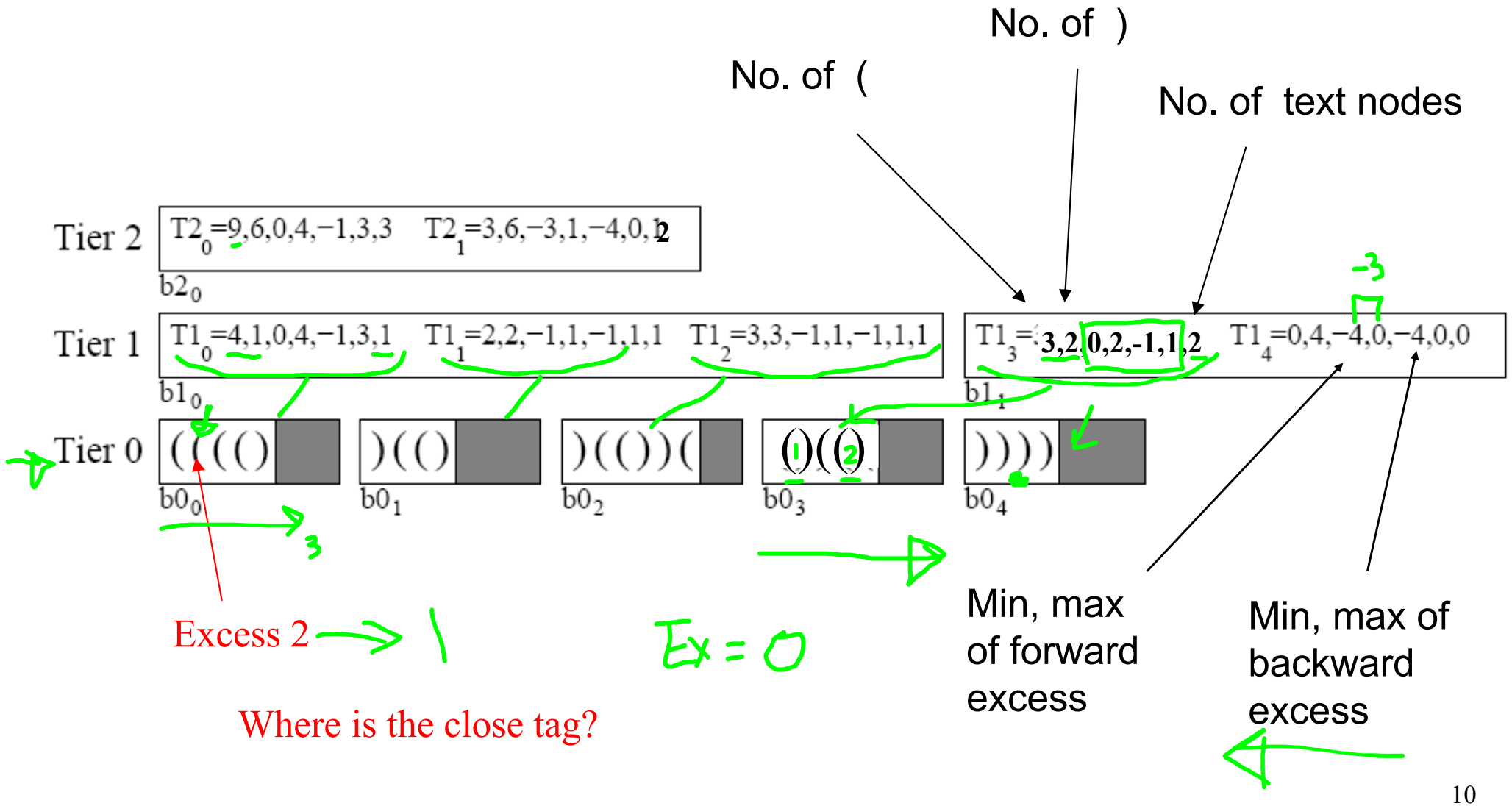


0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1

Node Navigations



Topology Tiers

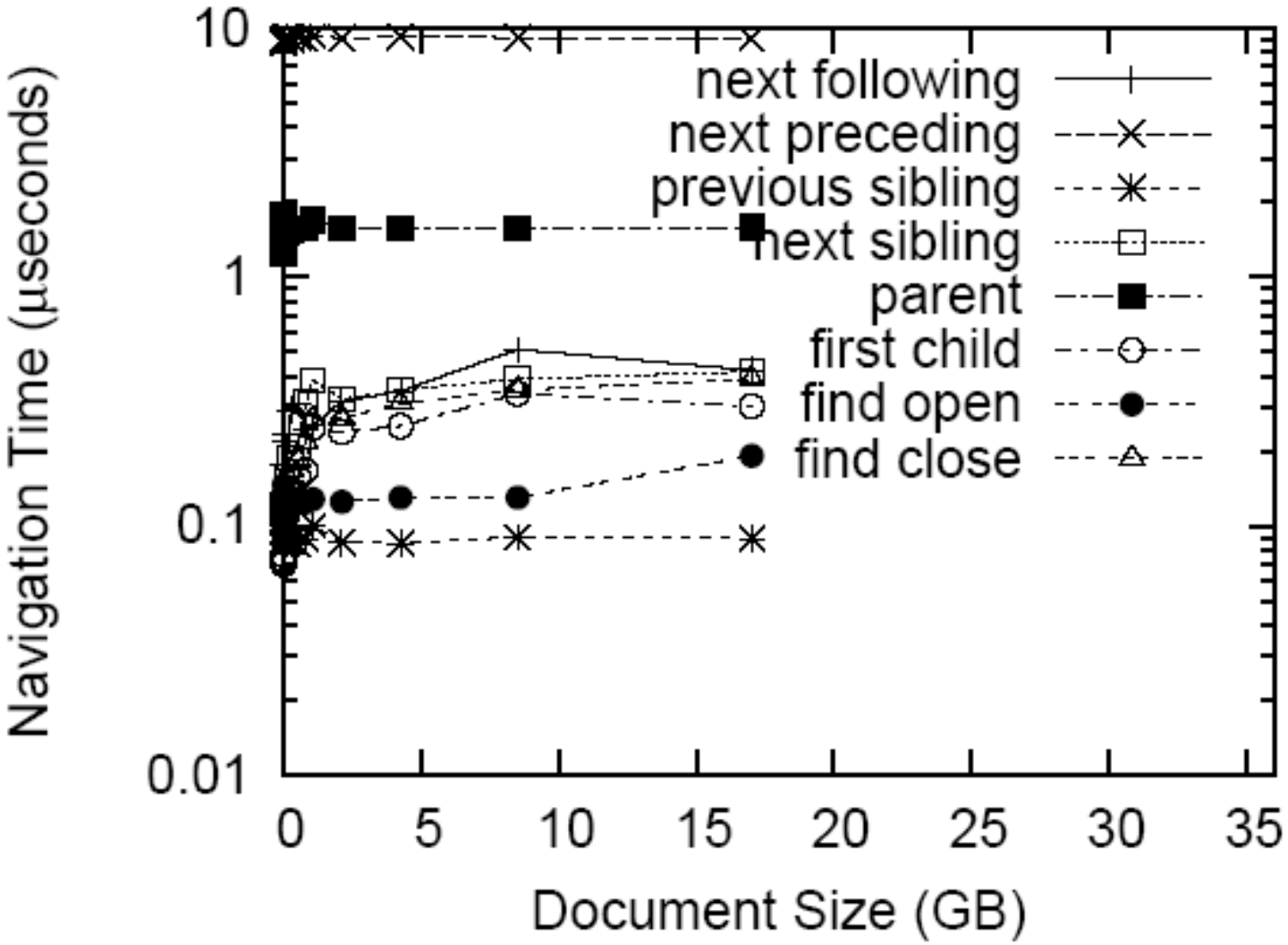


Experiments

Setup

- Fixed at **64MB memory buffer**
- Up to 16 GB XML document
- E.g. 16 GB DBLP contains > 770 million nodes
- **NO** index or query optimization has been employed for ISX (*except for ISX Stream where TurboXPath algorithm has been employed*)

Node Navigation



Content delivery

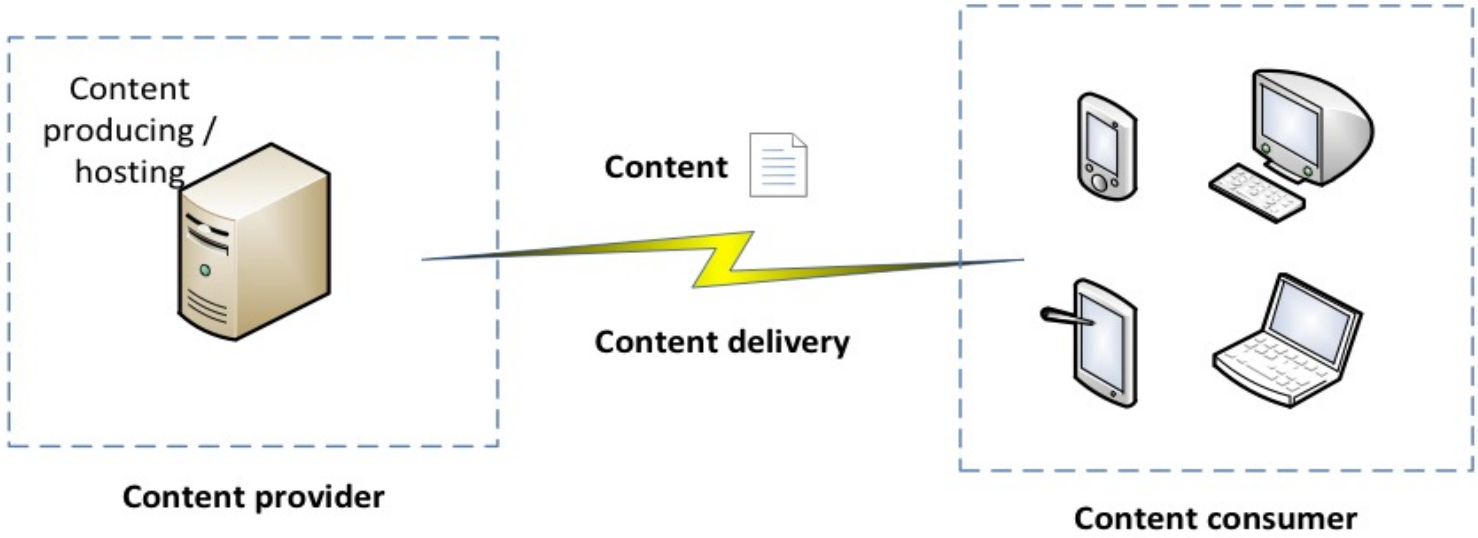


Figure 1. Content delivery from content provider to content consumer

Content optimization

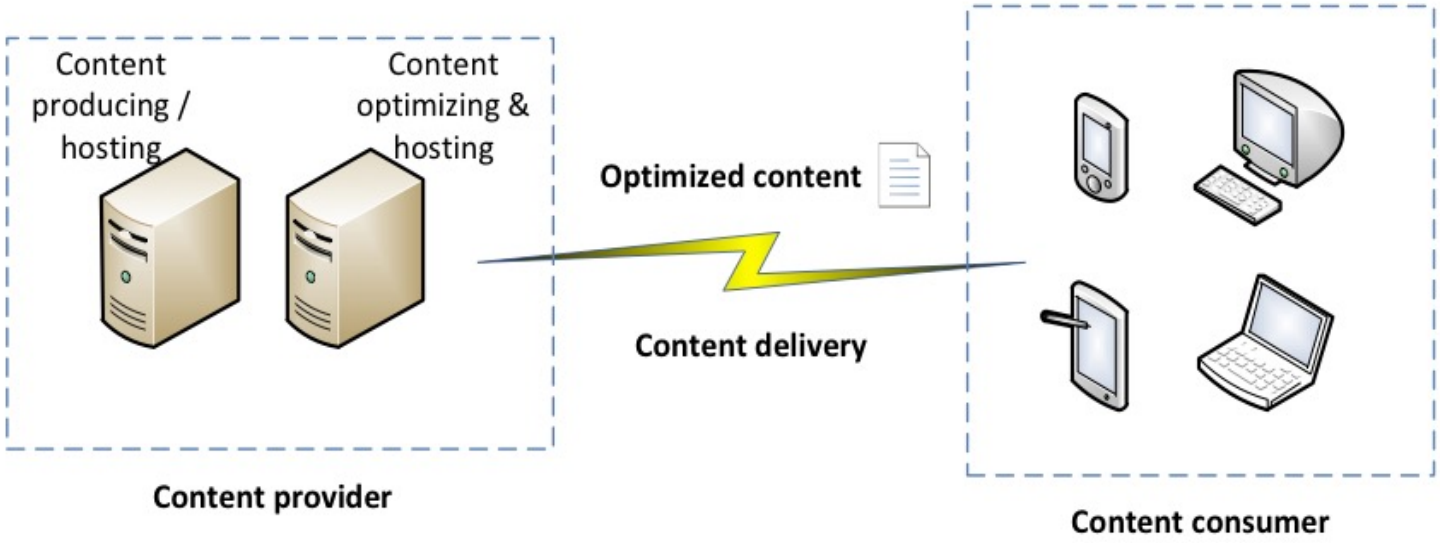


Figure 2. Delivery of content with content optimization

Table I
PRICING FOR AMAZON EC2 ON-DEMAND INSTANCES FOR LINUX/UNIX USAGE

Instances	Pricing /Hr (Oregon)	Pricing /Hr (Singapore)	Pricing /Hr (Tokyo)
Extra Large	0.320	0.360	0.368
Standard Extra Large	0.640	0.720	0.736
High-CPU Extra Large	0.660	0.744	0.760
High-Memory Quadruple	1.800	2.024	2.072

Table II
PRICING FOR AMAZON EC2 DATA TRANSFER

Data transfer out /month	Pricing /GB (US)	Pricing /GB (Singapore)	Pricing /GB (Tokyo)
First 1 GB	0.000	0.000	0.000
Up to 10 TB	0.120	0.190	0.201
Next 40 TB	0.090	0.150	0.158
Next 100 TB	0.070	0.130	0.137

Table III
PRICING FOR AMAZON S3 STANDARD STORAGE

Size /month	Pricing /GB (US / Singapore)	Pricing /GB (Tokyo)	Pricing /GB (Northern CA)
First 1 TB	0.125	0.130	0.140
Next 49 TB	0.110	0.115	0.125
Next 450 TB	0.095	0.100	0.115

Table IV
DATA COMPRESSION BENCHMARK FOR A 301MB FILE

Program	Compression ratio (%)	Compression time (sec)	Decompression time (sec)
7-Zip	72.00	49.2	7.1
GZip	63.51	15.5	10.2
BZip2	65.95	48.7	14.1
LZW	51.55	154	5.7

Mobile bandwidth cost in AU

- Pay As You Go: $2 / MB
- $69 per month plan: 12GB, excess $0.05 / MB
- Assume $10 per month plan: 1GB, excess $0.25 / MB, i.e., avg rate $0.01 / MB

Assumption

- 50TB static content
- Updated once a month (e.g., magazine)
- Each user accesses 100MB
- Hosted in Amazon Singapore

Table V
DATA COMPRESSION ON AMAZON CLOUD

	Original	7-Zip	GZip	BZip2	LZW
Size (TB)	50	14	18.245	17.025	24.225
Storage ($)	5515	1555	2021.95	1887.75	2679.75
Data transfer ($)	7900	2500	3136.75	2953.75	4033.75
Compression time (hrs)	0	2270.21	715.21	2247.14	7105.94
High-CPU EL ($)	0	1689.04	532.12	1671.87	5286.82
Mobile bandwidth per content item ($)	1.00	0.28	0.3649	0.3405	0.4845
Decompression time per content item (sec)	0	2.36	3.39	4.68	1.89

Findings

- Data transfer in is free
- CPU computation cost is more significant than storage & bandwidth costs

Table VI
COHESIVE DATA'S OPTIMIZATION PERFORMANCE FOR 250MB FILES

Encode time (sec)	72.09
Decode time (sec)	12.13
Compression ratio (%)	73.60
Encode time for 10MB file (sec)	3.07
Size of 10MB file encoded (MB)	2.66
Append time for 10MB file (sec)	2.32

Table VII
COHESIVE DATA'S OPTIMIZATION FOR WEB BROWSING

Website	Raw (KB)	Optimized (KB)	Compression ratio (%)	Rendering speedup
Amazon	920	271	70.54	250%
Yahoo	1073	197	81.64	220%
Ebay	1089	149	86.32	250%
Wikipedia	749	200	73.30	400%
Blogger	1882	945	49.79	211%
Fox Sports	1620	203	87.47	233%
ESPN	1159	106	90.85	165%
Weather.com	1140	88	92.28	157%
Best Buy	1320	139	89.47	243%
NY Times	1283	135	89.48	320%

Table VIII
PERFORMANCE OF CONTENT OPTIMIZATION ON AMAZON CLOUD

	Original	Cohesive
Size (TB)	50	13.2
Storage ($)	5515	1467
Data transfer ($)	7900	2380
Compression time (hrs)	0	4005
High-CPU EL ($)	0	2979.72
Mobile bandwidth cost per 10MB ($)	0.100	0.0264
Decompression time per 10MB (sec)	0	0.4852

Maximizing the value

- Need to be stored for a long period
- Will be transmitted many times
- Further processing on the cloud is needed
- Low-cost updates for dynamic content