Fast Dictionary-based Compression for Inverted Indexes

Giulio Ermanno Pibiri
The University of Pisa
and ISTI-CNR
Pisa, Italy

Matthias Petri
The University of Melbourne
Melbourne, Australia

Alistair Moffat
The University of Melbourne
Melbourne, Australia

12/02/2019
We focus on compression effectiveness and decoding speed for **inverted indexes**.

The inverted index is the *de-facto* data structure at the basis of every large-scale retrieval system.
We focus on compression effectiveness and decoding speed for **inverted indexes**.

The inverted index is the *de-facto* data structure at the basis of every large-scale retrieval system.

$V = \{\text{always, boy, good, house, hungry, is, red, the}\}$

$L_{t_1} = [1, 3]$
$L_{t_2} = [4, 5]$
$L_{t_3} = [1]$
$L_{t_4} = [2, 3]$
$L_{t_5} = [3, 5]$
$L_{t_6} = [1, 2, 3, 4, 5]$
$L_{t_7} = [1, 2, 4]$
$L_{t_8} = [2, 3, 5]$
Huge research corpora describing different space/time trade-offs.

- Elias gamma/delta
- Variable-Byte family
- Binary Interpolative Coding
- Simple family
- PForDelta

- Optimized PForDelta
- Elias-Fano
- Partitioned Elias-Fano
- Clustered Elias-Fano
- Asymmetric Numeral Systems
Many solutions

Huge research corpora describing different space/time trade-offs.

- Elias gamma/delta
- Variable-Byte family
- Binary Interpolative Coding
- Simple family
- PForDelta

- Optimized PForDelta
- Elias-Fano
- Partitioned Elias-Fano
- Clustered Elias-Fano
- Asymmetric Numeral Systems

Space

Interpolative

\[ \sim 3X \] smaller

Time

Variable-Byte + SIMD

\[ \sim 4.5X \] faster
Many solutions

Huge research corpora describing different space/time trade-offs.

- Elias gamma/delta
- Variable-Byte family
- Binary Interpolative Coding
- Simple family
- PForDelta

- Optimized PForDelta
- Elias-Fano
- Partitioned Elias-Fano
- Clustered Elias-Fano
- Asymmetric Numeral Systems

**Space**

- Interpolative
  - ~3X smaller

**Time**

- Variable-Byte + SIMD
  - ~4.5X faster

Can we inherit both advantages?
A crucial fact

Patterns of $d$-gaps are repetitive.
A crucial fact

Patterns of $d$-gaps are *repetitive*.
DINT — Dictionary of INTegers

- Encode a whole pattern with a single dictionary reference of $b$ bits
- Decode a whole pattern with a single dictionary access

**input stream**

$2^b$

$\ell + 1$

*fixed-to-fixed arrangement*
DINT — Dictionary of INTegers

- Encode a whole pattern with a *single* dictionary reference of *b* bits
- Decode a whole pattern with a *single* dictionary access

```plaintext

\[ \ell + 1 \]

\[ 2^b \]

\[
\begin{align*}
\text{copy}(D, c, \text{output}) & \\
\text{begin} &= c \times (\ell + 1) \\
\text{copy} 4 \times \ell \text{ bytes starting from } D[\text{begin}] \text{ to } \text{output} \\
\text{end} &= \text{begin} + \ell \\
\text{size} &= D[\text{end}] \\
\text{return size}
\end{align*}
\]

\[
\begin{align*}
\text{decode}(D, \text{input}, \text{output}) & \\
\text{for } i = 0; i < B; & \\
\quad c &= \text{get_16bits}(\text{input}) \\
\quad \text{size} &= 1 \\
\quad \text{if } c > 2 & \\
\quad \quad \text{size} &= \text{copy}(D, c, \text{output}) \\
\quad \text{else} & \\
\quad \quad \text{e} &= 0 \\
\quad \quad \text{if } c == 1 & \\
\quad \quad \quad \text{e} &= \text{get_32bits}(\text{input}) \\
\quad \quad \text{else} & \\
\quad \quad \quad \text{e} &= \text{get_16bits}(\text{input}) \\
\quad \text{copy e to } \text{output} & \\
\quad i &= i + \text{size}
\end{align*}
\]
```
DINT — Dictionary of INTegeRs

- Encode a whole pattern with a single dictionary reference of $b$ bits
- Decode a whole pattern with a single dictionary access

```
1 copy(D, c, output)
2 begin = c x (l + 1)
3 copy 4 x l bytes starting from D[begin] to output
4 end = begin + l
5 size = D[end]
6 return size
```

<table>
<thead>
<tr>
<th></th>
<th>Variable-length</th>
<th></th>
<th>Constant-length</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>docs</td>
<td>freqs</td>
<td>docs</td>
<td>freqs</td>
</tr>
<tr>
<td>instructions ($\times 10^9$)</td>
<td>53.63</td>
<td>35.02</td>
<td>41.72</td>
<td>28.35</td>
</tr>
<tr>
<td>instructions/cycle</td>
<td>1.16</td>
<td>1.13</td>
<td>1.28</td>
<td>1.24</td>
</tr>
<tr>
<td>cache-misses ($\times 10^7$)</td>
<td>10.77</td>
<td>9.06</td>
<td>8.21</td>
<td>7.60</td>
</tr>
<tr>
<td>branch-misses (%)</td>
<td>3.40</td>
<td>2.79</td>
<td>2.24</td>
<td>0.35</td>
</tr>
<tr>
<td>nanoseconds/int</td>
<td>1.82</td>
<td>1.08</td>
<td>1.12</td>
<td>0.73</td>
</tr>
</tbody>
</table>
DINT — Dictionary of INTegers

- Encode a whole pattern with a single dictionary reference of \( b \) bits
- Decode a whole pattern with a single dictionary access

```plaintext
1  copy(D, c, output)
2      begin = c × (\ell + 1)
3  copy 4 × \ell bytes starting from D[begin] to output
4  end = begin + \ell
5  size = D[end]
6  return size
```

<table>
<thead>
<tr>
<th></th>
<th>Variable-length docs</th>
<th>Variable-length freqs</th>
<th>Constant-length docs</th>
<th>Constant-length freqs</th>
</tr>
</thead>
<tbody>
<tr>
<td>instructions (× 10^9)</td>
<td>53.63</td>
<td>35.02</td>
<td>41.72</td>
<td>28.35</td>
</tr>
<tr>
<td>instructions/cycle</td>
<td>1.16</td>
<td>1.13</td>
<td>1.28</td>
<td>1.24</td>
</tr>
<tr>
<td>cache-misses (× 10^7)</td>
<td>10.77</td>
<td>9.06</td>
<td>8.21</td>
<td>7.60</td>
</tr>
<tr>
<td>branch-misses (%)</td>
<td>3.40</td>
<td>2.79</td>
<td>2.24</td>
<td>0.35</td>
</tr>
<tr>
<td>nanoseconds/int</td>
<td>1.82</td>
<td>1.08</td>
<td>1.12</td>
<td>0.73</td>
</tr>
</tbody>
</table>

1/3 of the time is saved
Refinements

1. Packed dictionary structure
   Exploiting string overlap

2. Optimal block parsing

3. Multiple dictionaries
Experimental results: setting

Datasets

<table>
<thead>
<tr>
<th>Collection</th>
<th>Lists</th>
<th>Postings</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gov2</td>
<td>39,180,840</td>
<td>5,880,709,591</td>
<td>25,205,179</td>
</tr>
<tr>
<td>CCNEWS</td>
<td>43,844,574</td>
<td>20,150,335,440</td>
<td>43,530,315</td>
</tr>
</tbody>
</table>

Machine
Intel Xeon 6144 processor, 512 GiB RAM, Linux 4.13.0

Compiler
gcc 7.2.0 (with all optimizations)

C++ code available at https://github.com/jermp/dint
### Experimental results: compression effectiveness

<table>
<thead>
<tr>
<th>Method</th>
<th>Gov2</th>
<th></th>
<th></th>
<th>CCNEWS</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GiB</td>
<td>docids</td>
<td>freqs</td>
<td>GiB</td>
<td>docids</td>
<td>freqs</td>
</tr>
<tr>
<td>Varint-GB</td>
<td>14.48</td>
<td>11.04</td>
<td>10.04</td>
<td>48.68</td>
<td>10.72</td>
<td>10.01</td>
</tr>
<tr>
<td>Varint-G8IU</td>
<td>12.77</td>
<td>9.90</td>
<td>8.69</td>
<td>43.87</td>
<td>9.75</td>
<td>8.93</td>
</tr>
<tr>
<td>VByte</td>
<td>11.85</td>
<td>9.22</td>
<td>8.02</td>
<td>39.65</td>
<td>8.88</td>
<td>8.00</td>
</tr>
<tr>
<td>QMX</td>
<td>5.59</td>
<td>4.99</td>
<td>3.11</td>
<td>19.20</td>
<td>5.12</td>
<td>3.04</td>
</tr>
<tr>
<td>Simple16</td>
<td>5.28</td>
<td>4.84</td>
<td>2.81</td>
<td>16.85</td>
<td>4.70</td>
<td>2.46</td>
</tr>
<tr>
<td>Opt-PFOR</td>
<td>4.55</td>
<td>4.33</td>
<td>2.26</td>
<td>15.50</td>
<td>4.49</td>
<td>2.10</td>
</tr>
<tr>
<td>DINT</td>
<td>4.29</td>
<td>4.22</td>
<td>1.98</td>
<td>15.09</td>
<td>4.48</td>
<td>1.92</td>
</tr>
<tr>
<td>PEF</td>
<td>4.16</td>
<td>3.85</td>
<td>2.23</td>
<td>13.75</td>
<td>3.89</td>
<td>1.97</td>
</tr>
<tr>
<td>Clust-EF</td>
<td>4.02</td>
<td>3.66</td>
<td>2.16</td>
<td>13.44</td>
<td>3.79</td>
<td>1.92</td>
</tr>
<tr>
<td>Interp</td>
<td>3.86</td>
<td>3.54</td>
<td>2.04</td>
<td>12.80</td>
<td>3.64</td>
<td>1.79</td>
</tr>
<tr>
<td>ANS, 2d</td>
<td>3.71</td>
<td>3.56</td>
<td>1.86</td>
<td>12.58</td>
<td>3.67</td>
<td>1.69</td>
</tr>
</tbody>
</table>
## Experimental results: compression effectiveness

![Table of compression effectiveness results](table.png)

- **Method**
- **Gov2**
  - GiB
  - docids
  - freqs
- **CCNEWS**
  - GiB
  - docids
  - freqs

<table>
<thead>
<tr>
<th>Method</th>
<th>Gov2 GiB</th>
<th>Gov2 docids</th>
<th>Gov2 freqs</th>
<th>CCNEWS GiB</th>
<th>CCNEWS docids</th>
<th>CCNEWS freqs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Varint-GB</td>
<td>14.48</td>
<td>11.04</td>
<td>10.04</td>
<td>48.68</td>
<td>10.72</td>
<td>10.01</td>
</tr>
<tr>
<td>Varint-G8IU</td>
<td>12.77</td>
<td>9.90</td>
<td>8.69</td>
<td>43.87</td>
<td>9.75</td>
<td>8.93</td>
</tr>
<tr>
<td>VByte</td>
<td>11.85</td>
<td>9.22</td>
<td>8.02</td>
<td>39.65</td>
<td>8.88</td>
<td>8.00</td>
</tr>
<tr>
<td>QMX</td>
<td>5.59</td>
<td>4.99</td>
<td>3.11</td>
<td>19.20</td>
<td>5.12</td>
<td>3.04</td>
</tr>
<tr>
<td>Simple16</td>
<td>5.28</td>
<td>4.84</td>
<td>2.81</td>
<td>16.85</td>
<td>4.70</td>
<td>2.46</td>
</tr>
<tr>
<td>Opt-PFOR</td>
<td>4.55</td>
<td>4.33</td>
<td>2.26</td>
<td>15.50</td>
<td>4.49</td>
<td>2.10</td>
</tr>
<tr>
<td><strong>DINT</strong></td>
<td><strong>4.29</strong></td>
<td><strong>4.22</strong></td>
<td><strong>1.98</strong></td>
<td><strong>15.09</strong></td>
<td><strong>4.48</strong></td>
<td><strong>1.92</strong></td>
</tr>
<tr>
<td>PEF</td>
<td>4.16</td>
<td>3.85</td>
<td>2.23</td>
<td>13.75</td>
<td>3.89</td>
<td>1.97</td>
</tr>
<tr>
<td>Clust-EF</td>
<td>4.02</td>
<td>3.66</td>
<td>2.16</td>
<td>13.44</td>
<td>3.79</td>
<td>1.92</td>
</tr>
<tr>
<td>Interp</td>
<td>3.86</td>
<td>3.54</td>
<td>2.04</td>
<td>12.80</td>
<td>3.64</td>
<td>1.79</td>
</tr>
<tr>
<td>ANS, 2d</td>
<td>3.71</td>
<td>3.56</td>
<td>1.86</td>
<td>12.58</td>
<td>3.67</td>
<td>1.69</td>
</tr>
</tbody>
</table>
Experimental results: effectiveness/efficiency plot

![Plot showing effectiveness vs efficiency with different markers for Interp, PEF, Opt-PFOR, QMX, Stream-VByte, DINT ℓ = 16, and DINT ℓ = 8. The x-axis represents space in GiB, and the y-axis represents time in ns/INT.](image-url)
Experimental results: effectiveness/efficiency plot

![Graph showing effectiveness versus efficiency with various data points and markers.](image)
Experimental results: effectiveness/efficiency plot

- Interp
- PEF
- Opt-PFOR
- QMX
- Stream-VByte
- DINT $\ell = 16$
- DINT $\ell = 8$
Further readings

Chapter 6 and 7 of my Ph.D. thesis.

(more datasets, comparisons, query timings)

Thanks for your attention, time, patience!

Any questions?