Streaming Document Deltas
Tiles from 21.11 to 22.05 ...

Collabora Rocks
And this is why

Collabora Rocks

Collabora Rocks

Previously:
re-compress whole tile

requested: 2
rec-tiles: 1
recv-delta: 2

requested: 5
rec-tiles: 1
recv-delta: 3
best: 7 ms
avg: 12 ms
worst: 18 ms
last: 7 ms

requested: 5
rec-tiles: 1
recv-delta: 3
best: 13 ms
avg: 19 ms
worst: 23 ms
last: 13 ms

requested: 2
rec-tiles: 1
recv-delta: 2
best: 20 ms
avg: 22 ms
worst: 23 ms
last: 20 ms
Now for a new line:

Lets make a delta bytes eg.

c copy a span from previous delta.

12 count-of-rows
150 src-row

180 dest-row

d $x<y<pix-count>$

t Terminate delta

So hitting enter → small change ~8 bytes

Previously: PNG headers, compression - ~2k+ per simple tile.
PNG compression – threaded but ...

Numbers > 70% in reality – this is a debug build

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Binary</th>
<th>cycles (self)</th>
<th>cycles (incl.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>??</td>
<td>libz.so.1.2.11</td>
<td>59.3%</td>
<td>59.3%</td>
</tr>
<tr>
<td>png_write_row</td>
<td>libpng16.so.16.37.0</td>
<td>8.88%</td>
<td>68.6%</td>
</tr>
<tr>
<td>pthread_mutex_unlock@plt</td>
<td>libvcl.so</td>
<td>5.82%</td>
<td>5.82%</td>
</tr>
<tr>
<td>std::chrono::V2::steady_clock::now()@plt</td>
<td>coolforkit.so</td>
<td>5.8%</td>
<td>5.8%</td>
</tr>
<tr>
<td>SwTextNode::SetWrong(SwWrongList*, bool)</td>
<td>libswlo.so</td>
<td>4.58%</td>
<td>4.58%</td>
</tr>
<tr>
<td>??</td>
<td></td>
<td>1.87%</td>
<td>1.87%</td>
</tr>
<tr>
<td>Poco::LocalDateTime::determineTzd(bool)</td>
<td>libPocoFoundation.so</td>
<td>0.697%</td>
<td>2.38%</td>
</tr>
<tr>
<td>VclReferenceBase::acquire() const</td>
<td>libswlo.so</td>
<td>0.629%</td>
<td>0.629%</td>
</tr>
</tbody>
</table>

| Caller | Binary | cycles | | Caller | Binary | cycles | |
|--------|--------|--------| | | | | |
| deflate | libz.so.1.2.11 | 59.3% | ?? | 0.00042% | ?? | 59.3% | 59.3% |
| ?? | libz.so.1.2.11 | 23.6% | ?? | libz.so.1.2.11 | 23.6% |
Old PNG generation flow

Invalidation from Kit → WSD

- Something changed in <this> area.

WSD

- Choose: ask Kit for tile or Notify Browser ...

Kit: gets render request

- With ‘previous’ PNG hash ‘wireId’
  - WireId == ‘unique’ (Spooky) hash
- Sends tile ... if after render doesn’t match.
New Delta generation flow

Invalidation from Kit → WSD

- Something changed in <this> area.

WSD

- Choose: ask Kit for tile or Notify Browser ...
- Controls Delta vs. Not ...

Kit: gets render request

- With monotonically increasing oldWireId or – force-keyframe if none present.
- Send wire-id

Browser

Reliable tile-cache by id.

WSD

Compressed Keyframe + delta cache

Kit

Tile rendering + deltas: Uncompressed Tile cache
Creating deltas ...

**Faster:**
- Make a Delta faster than PNG compression!
  - Needs to be faster than zstd compression

**Managing cache size → memory ...**
- 4k screen → 8m pixels - ~100 document tiles
  - ~32Mb – big.
- Initially – just cache for deltas around editing

**Accelerated unpremult_copy**
- For long runs of the same pixels ...

**Updated:**
- Stores keyframes + a list of deltas.
- Generates a new keyframe as/when that seems sensible size-wise.
Other wins:

Keyframe + Delta <x2> application
- Slower / late-arriving clients:
  - Apply multiple deltas.

No potential for spooky-hash collision

Less complexity needed for slow clients
- Can push stream of small deltas cheaply.

75% lower bandwidth (estimate)

~10% lower CPU cost (estimate)

More profiling needed ...
- Where are the next gotchas?
- Getting the next 2x CPU use win should not be hard ...
Shipping in 22.05 ...

at the end it looks the same, but feels silkier
What we can do next: ...

Optimization:

- More profiling ... low hanging fruit
- RLE compression of previous tiles:
  - Save tile-cache size.
  - zstd / dictionary compression

Intelligent region merge & render

- If we can delta: have old in cache
- Re-render a sub-tile area ...

Decompress to Uint8ClampedArray

- zstd – allows us to allocate an array type of our own to de-compress into.
- Avoid a biggish copy in the JS core.

Better Deltas

- Currently no horizontal change detection / re-use between lines.
- Should shrink things more.

More Unit tests!
Thanks!

By Michael Meeks