Delta OTA Update with SWUpΔate
• Which is a Δ (incremental) update
• FOSS projects for Δ
• Δ and SWUpdate
SWUpdate: OTA Update Agent

https://swupdate.org
Reasons for \( \Delta \) updates

- Size of SW is increasing
- Bandwidth constraints (GSM, etc.)
- Cost for device owner / hosting server
Ways in SWUpdate

• Split OS and application
  – Update of application is smaller
  – Consistency ??
• Delta update based on libr-sync
appimage :(
{
    filename = "myapp.tgz";
type = "archive";
device = "dev to be used";
path = "/";
filesystem = "ext4";
sha256 =<computed hash>;
}
);

Split OS + App
Use librsync handler

- Prepare delta during build
- \( \Delta \) File for each source version

```json
images: (

{

type = "rdiff_image";
filename = "image.rdiff.delta";
device = "/dev/mmcblk0p2";
properties: {

rdiffbase = ["/dev/mmcblk0p1"];
}

});
```
- \( Rdiff \rightarrow \Delta \)
- OK rolling releases
- Prebuilt \( \Delta \) images
- No suitable for update from any release to any release
• Independent from type of sources
  – $\Delta$ is smaller if type is known
• Low Resources to create the destination
• $\Delta + \text{SRC} \rightarrow \text{DST(device)} == \text{DST(Build)}$
Deployment Server

It becomes a proprietary solution!
Different concept

- **Δ on Server**
  - Server holds all versions (old and new)
  - Generates multiple Δ files
  - Check based just on info sent by device
  + Less CPU Load on device
  + Update faster

- **Δ on Device**
  + Server holds one version
  + Δ on device → same server
  + Δ from any X to any Y
  + Crypto check
  - More CPU Load on device
  - Update slower
FOSS to build Δ

- X-delta
  - Resulting image is built in RAM
  - Not suitable for embedded
  - Δ (like libr sync) specific for each version
- Librsync → already treated
- casync
Casync with SWUpdate?

- Each chunk is a separate file
  - Device must download hundreds of separate files
  - Sometimes FW stored by another entity
  - HTTP(S) requests are expensive on small devices.
- It is a complex project
  - Integration difficult.
  - Library was planned, never implemented.
  - Breaks security („privilege separation“)
Zchunk

- Developed by Jonathan Dieter
  - https://github.com/zchunk/zchunk
- Used on Fedora Project
- Define a new file format
  - Self contained
  - Meta (index) is part of delivered file
The more I looked at casync, the more obvious it became that it’s designed for a different use-case (delivering full filesystem images), and, while close, wasn’t quite what I needed.
Changes in Zchunk

- Embedded friendly
  - Runtime errors instead assert / exit
  - No load file in memory to build ZCK
- Add uncompressed Hashing
- Extend API
  - To build index
  - To return list of chunks meta for old and new version
- Extend format (fields for uncompressed, etc.)
- Merged by Jonathan since 1.2.0
Zchunk format

- Header
- ZST Chunk 0
- ZST Chunk 1
- ZST Chunk n

- Lead
- Preface
- Index

- ID
- CHKS Type
- Header
- Size
- Header Chksum

- Data Chksum
- Flags
- Comp Type
- Optional Elements

- Index Size
- Chunk CHKS Type
- Chunk count
- Chunk 0
- Chunk Chksum
- Uncomp Chunk Chksum
- Chunk Length
- Uncomp Chunk Length

- Chunk n
- Chunk Chksum
- Uncomp Chunk Chksum
- Chunk Length
- Uncomp Chunk Length
Build

Rootfs.ext4 \(\xrightarrow{\text{zck tool}}\) Rootfs.ext4.zck

- Extract Header
- Header
- Build SWU (new SW)

Uploaded
Delta in SWU

images: (
  {
    filename = "myimage.rootfs.ext4.zck.header";
    type = "delta";
    device = "/dev/mmcblk0p2";
    properties: {
      url = "https://examples.com/my.rootfs.ext4.zck";
      chain = "raw";
      source = "/dev/mmcblk0p1";
      zckloglevel = "error";
      debug-chunks = "true";
    }
  }
)
Analyze source

```
mmcblk0p1
```

```
Library
```

```
Zchunk
```

```
Header
```

- Lead
- Preface
- Index Hdr
- Hdr Chunk 0
- Hdr Chunk m
Look up with UT Hash

Generated
https://troydhanson.github.io/uthash/

Downloaded in SWU
RFC 7233 – Range request

Multiple chunks in a single HTTP GET

Servers have a max range number

- Device must queue multiple requests
GET /rootfs.ext4.zck HTTP/1.1
Host: example.com
Range: bytes=24568-345678,435678-980123

HTTP/1.1 206 Partial Content
Content-Type: multipart/byteranges; boundary=ab1234cde5678
Content-Length: 321110

--ab1234cde5678
Content-Type: bytes
Content-Range: bytes 24568-345678

→ binary data
--ab1234cde5678

.................
Downloader

Bytes Request

Delta Handler

Source

Chain Handler
<table>
<thead>
<tr>
<th>Index</th>
<th>Typ</th>
<th>HASH</th>
<th>START(chunk)</th>
<th>SIZE(uncomp)</th>
<th>Pos(Device)</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>SRC</td>
<td>6e7f552dad1491de3d6b8a0e4cdb9de3f0b8c8b887079e0caf29671eb104fe</td>
<td>679747</td>
<td>9568</td>
<td>2709834</td>
</tr>
<tr>
<td>43</td>
<td>DST</td>
<td>22e4804633181c025ae3a0b783c8f556d68bc563d439fe18962a8d892c46f</td>
<td>680251</td>
<td>11762</td>
<td>2719402</td>
</tr>
<tr>
<td>44</td>
<td>SRC</td>
<td>f276c49bda860b0272e0d7545b9a8edf722f41469873f6e619388663c73f5</td>
<td>685407</td>
<td>8970</td>
<td>2837074</td>
</tr>
<tr>
<td>45</td>
<td>DST</td>
<td>dfdf9718联合会2c8f1f46d6b781ca64b9da516d23de466abf8fe0440a07a711f029d6</td>
<td>685867</td>
<td>131072</td>
<td>2846044</td>
</tr>
<tr>
<td>46</td>
<td>DST</td>
<td>6c7dec2b250ae3181ae3ce13b5f252e5d2716a895f3da9be0e2d151ace8f5</td>
<td>691353</td>
<td>131072</td>
<td>2977116</td>
</tr>
<tr>
<td>47</td>
<td>DST</td>
<td>7bed48b7773290c2b6a898ad0ce7b1e9f9c7d298b6d1327f956e9c315f9</td>
<td>697023</td>
<td>24640</td>
<td>3108188</td>
</tr>
<tr>
<td>48</td>
<td>DST</td>
<td>d61c71712d1c64cb879fbeb713f40cd4bf61a1079f4e0188eaf38be92fd439</td>
<td>698176</td>
<td>20678</td>
<td>3273321</td>
</tr>
<tr>
<td>49</td>
<td>DST</td>
<td>b2b9f696b5c73526c22c5c95efb44b8cb7a56b5a502239fd637b366012a05</td>
<td>698678</td>
<td>131072</td>
<td>3142249</td>
</tr>
<tr>
<td>50</td>
<td>DST</td>
<td>878f8d79219a9f0a0d510f03fcd2158f308f356e59f275104155e8b49d9</td>
<td>704303</td>
<td>20678</td>
<td>3273321</td>
</tr>
<tr>
<td>51</td>
<td>DST</td>
<td>7d389d45a011832dbc9188d84c61393b59691b5264846cfc3e193759be0fc5</td>
<td>705335</td>
<td>131072</td>
<td>3293999</td>
</tr>
<tr>
<td>52</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711188</td>
<td>131072</td>
<td>3425071</td>
</tr>
<tr>
<td>53</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711210</td>
<td>131072</td>
<td>3556143</td>
</tr>
<tr>
<td>54</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711232</td>
<td>131072</td>
<td>3687215</td>
</tr>
<tr>
<td>55</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711254</td>
<td>131072</td>
<td>3818287</td>
</tr>
<tr>
<td>56</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711276</td>
<td>131072</td>
<td>3949359</td>
</tr>
<tr>
<td>57</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711298</td>
<td>131072</td>
<td>4080431</td>
</tr>
<tr>
<td>58</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711320</td>
<td>131072</td>
<td>4211503</td>
</tr>
<tr>
<td>59</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711342</td>
<td>131072</td>
<td>4342575</td>
</tr>
<tr>
<td>60</td>
<td>SRC</td>
<td>fa43239bcee7b9c797a62f007cc6848756a39e1974f3de7486db3f98de8e71</td>
<td>711364</td>
<td>131072</td>
<td>4473647</td>
</tr>
</tbody>
</table>

**INFO**: Total bytes to be reused: 637800925

**INFO**: Total bytes to be downloaded: 197733

**INFO**: Size of artifact to be installed: 641728512
Bandwidth at build time

- Generate ZCK
  - zck -u -h sha256 -v -o rootfs.ext4.zck rootfs.ext4
- Tool to get difference (exposed as test in Zchunk)
  - zck_cmp_uncomp rootfs.ext4.old rootfs.ext4.zck

```
8107 SRC fa43239bcee7b97ca62f007cc68487560a39e19f7f4f3dde7486db3f98df8e471 83540064 131072
8108 SRC fa43239bcee7b97ca62f007cc68487560a39e19f7f4f3dde7486db3f98df8e471 83540066 131072
8109 SRC fa43239bcee7b97ca62f007cc68487560a39e19f7f4f3dde7486db3f98df8e471 83540068 131072
8110 SRC fa43239bcee7b97ca62f007cc68487560a39e19f7f4f3dde7486db3f98df8e471 83540070 131072
8111 SRC fa43239bcee7b97ca62f007cc68487560a39e19f7f4f3dde7486db3f98df8e471 83540072 131072
8112 SRC fa43239bcee7b97ca62f007cc68487560a39e19f7f4f3dde7486db3f98df8e471 83540074 131072
8113 SRC fa43239bcee7b97ca62f007cc68487560a39e19f7f4f3dde7486db3f98df8e471 83540076 131072
8114 SRC fa43239bcee7b97ca62f007cc68487560a39e19f7f4f3dde7486db3f98df8e471 83540078 131072
8115 DST d69ae673b9891676f11ece248286fc0eb773e1ae799c1869aabf9f74ca744805 83540240 122229
```

Total to be reused : 602976803
Total to be downloaded : 9008805
Todo

- It was commissioned for Debian / ISAR
  - Missing support in meta-swupupdate
  - Methods to generate .zck
  - Build of SWU
- Optimizations?
Contributions

- Δ handler with dual-copy update.
- No changes on Server side
- Just store the ZCK file
- Bytes to be downloaded known at build time