Embedded video pipeline

Marek Vašut

November, 2020

ToC

Uvod

- Prehravani videa je HW narocne
- Embedded systemy maji omezene HW prostredky
  - Nizka pametova propustnost
  - Nevykonny procesor
- Embedded systemy maji podpurny HW
  - Dedikovane en/dekodery videa
  - Ruzne jednotky na upravu video (rotace, scaling, ...)
- HW je potreba vyuzit efektivne
- Audio nebude (komplexita na jinou prednasku)

Video pipeline

Tri casti

- Vstup
  - Soubor
  - Kamera
- Zpracovani
  - Hardware
  - Software
- Vystup
  - Soubor
  - Display

Vstup

- Soubor je snadny (open() + read())
- Kamera je slozitejsi
  - V4L2 subsystem
  - Media controller (komplexni kamery)
- Kamera v embedded systemech
  - Sensor (mimo SoC)
  - Camera IP (v SoC)

Sensor

- Snimac mimo SoC
- Muze mit ISP (konverze pixel formatu, 3A)
- Interface:
  - Parallel
  - MIPI CSI
- Pixel format
  - RGB (s ISP)
  - YUV
  - Bayer SRGB

Camera IP

- IP v SoC
- Prijima data ze sensoru
- Zapisuje data do RAM
- Muze umet konvertovat pixel format sensoru
- Muze umet dalsi speciality (predzpracovani)

V4L2

- Kernelove API pro praci s media zarizenimi
- Typicky kamera (vstup), M2M (zpracovani), ale i vystup
- V4L2 drivery pro kamery
- Common V4L2 kod pro spravu bufferu
- Buffery
  - mmap
  - dmabuf
- Nastroj v4l2-ctl
  - `v4l2-ctl -d /dev/video0 --stream-mmap \`
    `--stream-count=1 --stream-to=/tmp/file.raw`
  - ffmpeg pro konverzi raw dat

Media controller

- API pro konfiguraci komplexnich kamer
- Komplexni kamera ma vice konfigurovatelnych casti
- Napr. zabudovana rotace obrazu, CSC, ...
- Viz media-ctl

V4L2 demo – alokace

```
1  int v4lfd = open("/dev/video0", O_RDWR);
2
3  struct v4l2_format fmt;
4  ioctl(v4lfd, VIDIOC_G_FMT, &fmt);
5
6  struct v4l2_requestbuffers rqbufs;
7  rqbufs.count = CONFIG_V4L2_BUF_COUNT;
8  rqbufs.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
9  rqbufs.memory = V4L2_MEMORY_DMABUF;
10 ioctl(v4lfd, VIDIOC_REQBUFS, &rqbufs);
11
12 for (i = 0; i < CONFIG_V4L2_BUF_COUNT; i++) {
13     struct v4l2_buffer buf = { 0 };
14     buf.index = i;
15     buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
16     buf.memory = V4L2_MEMORY_DMABUF;
17     ioctl(v4lfd, VIDIOC_QBUF, &buf);
18 }
19
20 int type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
21 ioctl(v4lfd, VIDIOC_STREAMON, &type);
```

V4L2 demo – dequeue/enqueue

```
1  struct pollfd pfd;
2  pfd.fd = v4lfd;
3  pfd.events = POLLIN;
4
5  poll(&pfd, 1, -1);
6  if (pfd.revents & POLLIN) {
7      struct v4l2_buffer buf = { 0 };
8      buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
9      buf.memory = V4L2_MEMORY_DMABUF;
10     ioctl(v4lfd, VIDIOC_DQBUF, &buf);
11     int index = buf.index;
12
13     ...
14
15     memset(&buf, 0, sizeof(buf));
16     buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
17     buf.index = index;
18     buf.memory = V4L2_MEMORY_DMABUF;
19     ioctl(v4lfd, VIDIOC_QBUF, &buf);
20 }
```

V4L2 demo – mmap

```
1  struct v4l2_buffer buf;
2  void *map;
3
4  for (i = 0; i < CONFIG_V4L2_BUF_COUNT; i++) {
5      memset(&buf, 0, sizeof(buf));
6      buf.index = i;
7      buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
8      buf.memory = V4L2_MEMORY_DMABUF;
9      ioctl(v4lfd, VIDIOC_QUERYBUF, &buf);
10
11     map = mmap(NULL, buf.length,
12                PROT_READ | PROT_WRITE, MAP_SHARED,
13                v4lfd, buf.m.offset);
14     if (map == MAP_FAILED)
15         exit(EINVAL);
16 ...
```

Zpracovani

- Hardware – V4L2 mem2mem zarizeni
  - Chova se velmi podobne jako V4L2 kamera (jen ma dve fronty bufferu)
  - M2M zarizeni cte buffer z RAM na vstupu
  - M2M zarizeni implementuje transformaci
  - M2M zapisuje buffer do RAM na vystupu
  - Typicky rotace, scaling, CSC, en/dekodovani videa
  - Konfigurace pomoci V4L2 nebo media controller
- Kamera v embedded systemech
  - Softwarove implementace filtru (treba pomoci SIMD)

Vystup

- Soubor je snadny (open() + write())
- Display je slozitejsi
  - DRM subsystem
  - fbdev emulace

DRM subsystem

- Kernelove API pro obsluhu zobrazovacich zarizeni, GPU, ...
- Sprava a konfigurace topologie a vystupu
- Sprava pameti a bufferu (vc. importu DMABUFu)
- Atomicke updatovani obrazu, ...
- libdrm – wrapper okolo DRM ioctl
- Buffer object (BO) – struktura okolo bufferu s datama

DRM demo – open

```
1  drmModeRes *resources;
2  drmModeConnector *connector = NULL;
3  drmModeEncoder *encoder = NULL;
4  int i, drmfd;
5
6  drmfd = open("/dev/dri/card0", O_RDWR | O_CLOEXEC);
7  resources = drmModeGetResources(drmfd);
8
9  for (i = 0; i < resources->count_connectors; i++) {
10     connector = drmModeGetConnector(drmfd, resources->connectors[i]);
11     if (connector->connection == DRM_MODE_CONNECTED)
12         break;
13     drmModeFreeConnector(connector);
14     connector = NULL;
15 }
16
17 for (i = 0; i < connector->count_modes; i++) {
18     drmModeModeInfo *current_mode = &connector->modes[i];
19     if (current_mode->type & DRM_MODE_TYPE_PREFERRED)
20         drmmode = current_mode;
21 }
```

DRM demo – open

```
1  for (i = 0; i < resources->count_encoders; i++) {
2      encoder = drmModeGetEncoder(drmfd, resources->encoders[i]);
3      if (encoder->encoder_id == connector->encoder_id)
4          break;
5      drmModeFreeEncoder(encoder);
6      encoder = NULL;
7  }
8
9  drmModeFreeResources(resources);
10
11 drmcrtc_id = encoder->crtc_id;
12 drmconnector_id = connector->connector_id;
```

DRM demo – framebuffer

```
1  uint32_t width, height, strides[4] = {0}, handles[4] = {0}, offsets[4]
2  const unsigned bo_attrs[] = { ... };
3  struct kms_driver *kd;
4  struct kms_bo *bo;
5
6  kms_create(drmfd, &kd);
7  kms_bo_create(kd, bo_attrs, &bo);
8
9  ...
10 kms_bo_map(...);
11 ...
12 kms_bo_unmap(...);
13 ...
14 kms_bo_get_prop(bo, KMS_HANDLE, handles);
15
16 drmModeAddFB2(drmfd, width, height, DRM_FORMAT_XRGB8888,
17               handles, strides, offsets, &fb_id, 0);
18
19 drmModePageFlip(drmfd, drmcrtc_id, fb_id,
20                 DRM_MODE_PAGE_FLIP_EVENT, &waiting_for_flip);
```

Import DMABUF

- V4L2 DMABUF jde exportovat a importovat primo do DRM
- Embedded GPU je typicky mem2mem zarizeni
  - Cte data z RAM, zapisuje data do RAM
  - Stejna RAM jako pouziva kamera, display, ...
- Je mozne importovat primo jako texturu do GPU

GBM

- libgbm – Generic Buffer Management
- Wrapper okolo alokatoru BO
- Spravu bufferu DRM zarizeni, hlavne GPU
- Soucast mesa3d (nicmene BSD, takze i soucast blobu)

GBM demo – alokace

```
1 static uint64_t modifiers[] = {DRM_FORMAT_MOD_LINEAR};
2 uint64_t *mods = modifiers;
3 int count = 1;
4
5 gbm_dev = gbm_create_device(drmfd);
6 gbm_surface = gbm_surface_create_with_modifiers(gbm_dev,
7                         drmmode->hdisplay, drmmode->vdisplay,
8                         GBM_FORMAT_XRGB8888, mods, count);
```

GBM demo – framebuffer

```
1  uint32_t width, height, strides[4] = {0}, handles[4] = {0}, offsets[4]
2  struct gbm_bo *bo;
3
4  bo = gbm_surface_lock_front_buffer(gbm_surface);
5
6  width = gbm_bo_get_width(bo);
7  height = gbm_bo_get_height(bo);
8
9  handles[0] = gbm_bo_get_handle(bo).u32;
10
11 drmModeAddFB2(drmfd, width, height, DRM_FORMAT_XRGB8888,
12                handles, strides, offsets, &fb_id, 0);
13
14 drmModePageFlip(drmfd, drmcrtc_id, fb_id,
15                  DRM_MODE_PAGE_FLIP_EVENT, &waiting_for_flip);
```

V4L2 a DRM demo – export a import DMABUF FD

```
1  struct v4l2_exportbuffer expbuf;
2  expbuf.index = i;
3  expbuf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
4  ioctl(v4lfd, VIDIOC_EXPBUF, &expbuf);
5  /* expbuf.fd je FD DMABUFu */
6
7  ...
8
9  eglCreateImageKHR(
10         display, EGL_NO_CONTEXT, EGL_LINUX_DMA_BUF_EXT, NULL,
11         (const EGLint []){
12                 EGL_WIDTH, fmt.pix.width,
13                 EGL_HEIGHT, fmt.pix.height,
14                 EGL_LINUX_DRM_FOURCC_EXT, DRM_FORMAT_YUYV,
15                 EGL_DMA_BUF_PLANE0_FD_EXT, expbuf.fd,
16                 EGL_DMA_BUF_PLANE0_OFFSET_EXT, 0,
17                 EGL_DMA_BUF_PLANE0_PITCH_EXT, fmt.pix.width * 2,
18                 EGL_NONE
19         });
```

Propustnost DRAM

- memcpy() spotrebovava DRAM bandwidth (viz Drepper `https://www.akkadia.org/drepper/cpumemory.pdf`)
- V4L2 M2M a GPU implicitne kopiruji
- Ostatni kopirovani je potreba eliminovat
- Zero-copy operace – bez neuzitecneho kopirovani
- Vyhledavani problemu, perf a HW DRAM countery

Propustnost DRAM – perf demo

```
$ perf list | grep ddr
  ddrperfm/activate_cnt/      [Kernel PMU event]
  ddrperfm/idle_cnt/          [Kernel PMU event]
  ddrperfm/read_cnt/          [Kernel PMU event]
  ddrperfm/time_cnt/          [Kernel PMU event]
  ddrperfm/write_cnt/         [Kernel PMU event]
  mem:<addr>[/len][:access]   [Hardware breakpoint]

$ perf stat -a -e ddrperfm/read_cnt/,ddrperfm/write_cnt/ sleep 1
 Performance counter stats for 'system wide':
          5701736      ddrperfm/read_cnt/
            19513      ddrperfm/write_cnt/
       1.013295459 seconds time elapsed

$ perf stat -a -e ddrperfm/read_cnt/,ddrperfm/write_cnt/ sleep 1
 Performance counter stats for 'system wide':
         19361468      ddrperfm/read_cnt/
          6673730      ddrperfm/write_cnt/
       1.058049131 seconds time elapsed
```

Propustnost CPU

- Embedded CPU neoplyva vykonem
- Pozor na SW implementace ruznych konverzi
- Pokud neni zbyti, SW implementace pomoci SIMD
- SIMD: arm neon, ppc altivec, x86 mmx/sse
- GStreamer a ORC / ffmpeg a swx

Gstreamer

- Framework pro praci s multimedii
- Casto se pouziva pro skryti detailu media pipeline
- Implicitne nemusi optimalne vyuzivat pipeline
- Demo gstreamer:

```
1 gst-launch-1.0 v4l2src device=/dev/video0 io-mode=dmabuf ! \
2     v4l2convert output-io-mode=dmabuf-import \
3                 capture-io-mode=dmabuf ! \
4     kmssink
```

Gstreamer videoconvert

- Element pro softwarovou konverzi video formatu
- Muze byt instanciovan implicitne
- Pouziva ORC pro SW akceleraci pomoc SIMD, pokud to jde
- Viz `GST_DEBUG="*:9"` a `ORC_DEBUG="9"`
- Casto lze nahradit lepsi konfiguraci pipeline / constraints:

```
gst-launch-1.0 v4l2src device=/dev/video0 io-mode=dmabuf ! \
    video/x-raw,format=YUY2 ! \
    v4l2convert output-io-mode=dmabuf-import \
                capture-io-mode=dmabuf ! \
    video/x-raw,format=RGBA ! \
    kmssink
```

Hledani problemu s CPU

- perf
- nahravani celeho systemu (potrebuje debug symboly)
  `perf record -a -g sleep 5`
- `perf report --stdio`
- popr. `perf top` na real-time monitorovani

Demo – perf top

```
$ gst-launch-1.0 videotestsrc num-buffers=6000 ! \
        video/x-raw,format=I420,width=1280,height=720 ! \
        videoconvert !
        video/x-raw,format=BGRA ! fakesink
```

```
Samples: 53K of event 'cycles', 4000 Hz, Event count (approx.): 13941453700 lost: 0/0 drop: 0/0
Overhead  Shared Object                  Symbol
  67.40%  libgstvideo-1.0.so.0.1801.0    [.] _backup_video_orc_convert_I420_BGRA
   7.21%  libgstvideo-1.0.so.0.1801.0    [.] _backup_video_orc_chroma_down_v2_u8
   3.56%  libgstvideotestsrc.so          [.] _backup_video_test_src_orc_splat_u32
   3.53%  libgstvideo-1.0.so.0.1801.0    [.] _backup_video_orc_pack_I420
   3.26%  libgstvideo-1.0.so.0.1801.0    [.] video_chroma_down_h2_cs_u8
   2.07%  libgstvideotestsrc.so          [.] videotestsrc_blend_line
   2.03%  libgstvideo-1.0.so.0.1801.0    [.] _backup_video_orc_pack_Y
   1.01%  libc-2.31.so                   [.] __memcpy_generic
   0.92%  libgstvideotestsrc.so          [.] gst_video_test_src_smpte
   0.47%  perf_5.9                       [.] 0x0000000000f4f6c
   0.34%  [kernel]                       [k] _raw_spin_unlock_irqrestore
   0.33%  [kernel]                       [k] lock_is_held_type
   0.31%  [kernel]                       [k] lock_acquire
   0.29%  [kernel]                       [k] _raw_spin_unlock_irq
   0.27%  libgstvideotestsrc.so          [.] video_test_src_orc_splat_u32
   0.23%  libgstvideotestsrc.so          [.] videotestsrc_convert_tmpline
   0.22%  libc-2.31.so                   [.] strchr
   0.19%  libgstvideotestsrc.so          [.] convert_hline_generic
   0.17%  libc-2.31.so                   [.] _int_malloc
   0.15%  libgstvideotestsrc.so          [.] paint_tmpline_AYUV
For a higher level overview, try: perf top --sort comm,dso
```

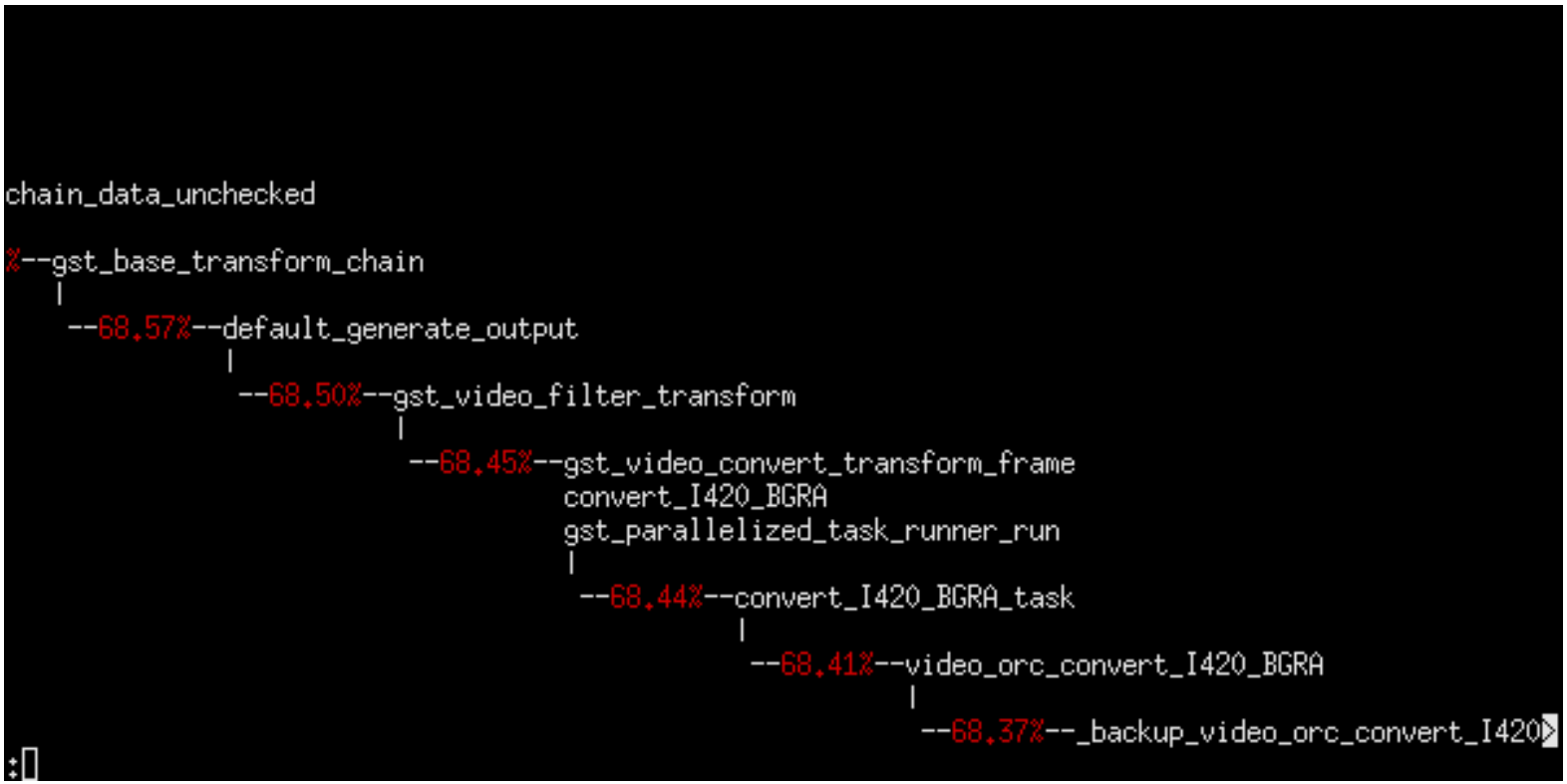
Demo – perf record (ORC backup)

```
$ perf record -a -g sleep 5
$ perf report --stdio
```

```
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 22K of event 'cycles'
# Event count (approx.): 7867169797
#
# Children      Self  Command          Shared Object                Symbol
# ........  ........  ...............  ...........................  ..............................
#
    94.02%     0.00%  videotestsrc0:s  libc-2.31.so                 [.] thread_start
           |
           ---thread_start
              start_thread
              0xffff880c1cac
              0xffff880c2640
              gst_task_func
              |
               --94.01%--gst_base_src_loop
                         |
                         |--68.79%--gst_pad_push
:
```

Demo – perf record (ORC backup)

```
$ perf record -a -g sleep 5
$ perf report --stdio
```



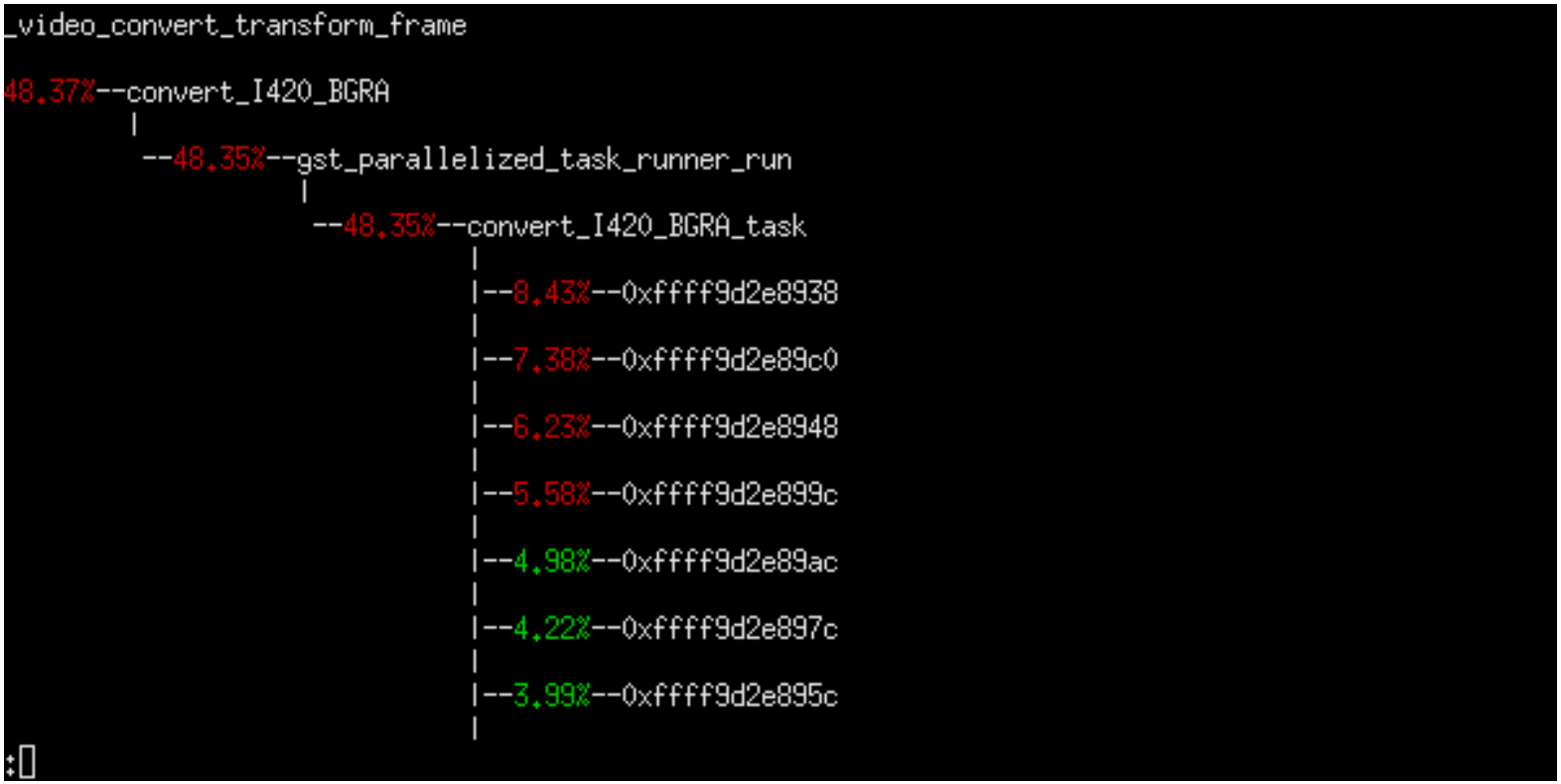
Demo – perf record (ORC SIMD)

```
$ perf record -a -g sleep 5
$ perf report --stdio
```

```
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 25K of event 'cycles'
# Event count (approx.): 8561629545
#
# Children      Self  Command          Shared Object                Symbol
# ........  ........  ...............  ...........................  ..............................
#
    87.44%     0.00%  videotestsrc0:s  libc-2.31.so                 [.] thread_start
           |
           ---thread_start
              start_thread
              0xffff9da40cac
              0xffff9da41640
              |
               --87.43%--gst_task_func
                         |
                          --87.42%--gst_base_src_loop
                                    |
:
```

Demo – perf record (ORC SIMD)

```
$ perf record -a -g sleep 5
$ perf report --stdio
```



Zaver

- memcpy() je problem
- pozor na CPU cykly
- optimalni konfigurace pipeline je dulezita

Dekuji za pozornost