GStreamer
The road to 1.0

Wim Taymans (wim@collabora.com)
Edward Hervey (edward@collabora.com)
8 aug 2011 – Desktop Summit 2011
• Reworked memory model
• Buffer Metadata
• Dynamic pipeline changes
  - Probes
  - Negotiation
  - Timing changes
• First class GstMemory object
  - Refcounted block of memory
  - Resize/copy
  - Map/unmap
• GstAllocator makes those blocks
  - Can add new allocators
  - Identified with a string name
• GstBuffer has list of GstMemory objects
• Buffer operations operate on underlying memory objects
  - Copy/resize
  - Map/unmap
So what?

*Blank Stare*
• Some DSPs need to store video planes in different memory blocks
• Scatter gather buffer data
Why explicit map/unmap GstMemory?
• GstMemory map/unmap to get access to the data
  - Keep track of who reads/writes
  - Cache flushes (between DSP/GPU)
  - Might actually do mmap/munmap or equivalent
• New memory model should improve
  - Integration with DSP/GPU
  - Integration with vaapi/vdpau
  - ...

Collabora
• GstMeta
  - Attach arbitrary structures to buffers
  - Extra properties
  - Extra methods
  - Well defined API, multiple implementations possible
But.. we want examples !
XImage information associated with GstBuffer

GstBuffer

X shared memory

GstMetaXImage
{
  ...
  Ximage *
  XshmSegmentInfo
  ...
}
GstMetaVideo describing video buffers

GstBuffer

image data

GstMetaVideo

{  
  ...  
  guint n_planes;  
  gsize offset[MAX_PLANES];  
  gint stride[MAX_PLANES];  
  
  gpointer (*map) (...);
  gpointer (*unmap) (...);
  ...  
}
- GstMetaVideo also has API

```c
 gpointer gst_meta_video_map (GstMetaVideo *meta,
                               guint plane,
                               gint *stride,
                               GstMapFlags flags);

gboolean gst_meta_video_unmap (GstMetaVideo *meta,
                               guint plane,
                               gpointer data);
```
• GstMetaCrop as an example of an operation
  - Instead of changing data, attach info about what to change and do the change later (maybe combined with other operations)
• But how can we know what metadata is supported in the pipeline
  – Does downstream understand cropping metadata or do we have to do the cropping ourselves?
• Consider decoder ! videosink

Does the videosink Understand cropping Metadata ??
- ALLOCATION query

videodec does the downstream ALLOCATION query to find out
• The ALLOCATION query:
  - How to allocate memory blocks (the supported allocators)
  - Alignment/prefix
  - Min/max amount of buffers
  - Supported metadata
  - But also: an optional GstBufferPool object
GstBufferPool ?
• Preallocate buffers
  - min/max amount of buffers
  - Prefix alignment
  - Reuse buffers
  - That's how some hardware wants it
  - That's how some API's want/prefer it (v4l2, OpenMax, ..)

• ...
• Most awesome feature of GstBufferPool is to do extensive configuration of the allocated buffers
  - Enabled/quered with extensible bufferpool options

... An example?
• Ask bufferpool to attach metadata to buffers
  – Because you can deal with it ( GstMetaVideo, for example)
• Ffmpeg without EMU_EDGE flag
  – Sink bufferpool supports extra config option for padding and stride_alignment
  – Ffmpegdec configures and sink allocates bigger area
NEGOTIATION

It's not always an option now, is it?

VERY DEMOTIVATIONAL.COM
• Renegotiation now with a RECONFIGURE event
  - No more piggyback on buffer_alloc

Allows us to remove all the complicated code from basetransform
Improved support for dynamic pipelines
- Sticky events
  - Define context of stream (caps, tags, timing info...)
  - Stored on pads
  - Passed to newly linked pads automatically
• Tweaked GstSegment to include the accumulated time (base)
  - No more segment accumulation
  - Segment accumulation was only useful for looping

• Add API to change offset on pads
  - Can adjust running-time on a per pad basis
• Improved pad probes
  - Merged probes and pad block
  - Can get notify about datapassing
  - Notify when no data is flowing on the pad (pad_block on steroids)
• New video GstCaps:
  - `video/x-raw-rgb, bpp=16, depth=15, endianness=1234, red_mask=31744, green_mask=992, blue_mask=31`
  - => `video/x-raw, format=RGB15`
Current state

• Core/Base/gst-ffmpeg working, some plugins from Good and Ugly too.
• First 0.11.0 release is out!
• Port plugins and applications!!
• API not 100% stable yet but getting close
  – There is a porting document
What's not quite working

• Bufferpool renegotiation is not yet well understood/implemented
• Dynamic pipeline features not so much tested
  – Probes API still misses interesting bits.
• We need to port more plugins to make it useful
• We need to make more plugins use the new features
• Some more Goals
  - Remove GstPropertyProbe
  - More base classes
  - Split parsers from decoders
What's next

- We'll be porting more apps and plugins
- We'll be doing more 0.11.x releases

On track for a 1.0 release later this year
Questions ?