GStreamer 1.0

No longer compromise flexibility for performance

Edward Hervey
Senior Multimedia Architect
edward@collabora.com
ELC 2012
GStreamer

- Open Source Multimedia Framework
- Set of libraries and plugins
- Direct Acyclic Graphs of elements
- API for plugins (to export features)
- API for applications

“Flexibility and performance”
Gstreamer usage

- Desktop, embedded, server, TV, ...
- Playback, recording, real-time communication, transcoding, ...
- Linux, Windows, MacOSX, iOS, Android, ...
- This talk is being recorded by GStreamer (UbiCast)
GStreamer 0.10

- 0.10 series (0.10.0 Dec 5 2005)
- Used widely and continuously improved
- More popular and solid than anticipated
0.10 Limitations

- Performance issues
- Some use-case very cumbersome to handle (hw-accel)
- Missing information
- Caps tightly coupled to buffer/memory
- Deprecated API
Enter GStreamer 1.0

- Talked about since 2007
- New challenges
  - Embedded Platforms
  - GPU
  - Dynamic pipelines
  - Re-negotiation
Goals

- Improve performance
- Allow more use-cases
- Avoid vendor 'hacks'
- Minimize downstream patches
GStreamer 1.0

- API/ABI cleanups and speedups
- Memory Management
- (Re)Negotiation
- Dynamic Pipelines
- Open the road to better performance
- More flexible/open API
Memory management

- 0.10
  - One buffer => One 'data' field (pointer)
  - Only accessible memory
  - Content entirely specified by caps
  - No control over memory access

- Problems
  - Different content layout => new caps
  - More fields => Override data (or subclass)
Memory management

• 0.10 Examples
  ▪ Stride
    • video/x-raw-yuv-strided,stride=4096,...
    • Incompatible with all existing video elements :(
  ▪ Non-contiguous planes
    • GstVendorBufferIncompatible
    • Also need specific caps to avoid other elements from prodding into (invalid/unknown) 'data' field
  ▪ <Insert the hack you had to do>

• => Incompatibility/Maintenance Hell
GStreamer hacker reviewing downstream patches
Memory management

1.0

- Memory separated from GstBuffer
- Caps separated from GstBuffer
- Generic Metadata system for GstBuffer
GstMemory

• Abstraction of memory
  ▪ flags (read only, not sharable, ...)
  ▪ refcount (MT-safe)
  ▪ size, maxsize, alignment, offset

• Buffer can point to many GstMemory

• No direct access
  ▪ gst_memory_map() / _unmap()
  ▪ GST_MAP_READ, GST_MAP_WRITE
GstMemory

System memory

GPU memory

DRM memory

DMA buf

offline Memory ?
GstAllocator

- GstAllocator provides GstMemory
  - .alloc(), .mem_free()
  - .mem_map(), .mem_unmap()
  - .mem_copy()
  - .mem_share()
  - .mem_is_span()
- => Explicit memory control
Inter-plugin communication

- Problem:
  - How do I communicate information to other plugins?
  - How do I do this in a transparent way?

- 0.10:
  - GstBuffer subclass and custom event
    - See previous rant about that
GstMeta

- Describes properties of a GstBuffer content
  - Video information (planes, strides, ...)
  - Extra buffer data (system context, ...)
  - Processing information (crop, pan, ...)
  - Anything you want really (but don't abuse it)

- query-able

- Can be ignored by elements
GstMeta

- C structure
- Stored in the GstBuffer memory
- `gst_buffer_get_meta()`
- `gst_buffer_add_meta()`
Ex: GstVideoMeta

```c
struct _GstVideoMeta {
    GstMeta meta;
    GstBuffer *buffer;
    GstVideoFlags flags;
    GstVideoFormat format;
    gint id;
    guint width;
    guint height;
    guint n_planes;
    gsize offset[GST_VIDEO_MAX_PLANES];
    gint stride[GST_VIDEO_MAX_PLANES];
    gboolean (*map) (GstVideoMeta *meta, guint plane,
                     GstMapInfo *info, gint *stride,
                     GstMapFlags flags);
    gboolean (*unmap) (GstVideoMeta *meta, guint plane,
                       GstMapInfo *info);
};
```
GstMeta

- Inter-plugin communication
- Ways to create new use-case-/field-specific APIs
- Stay compatible with other plugins
(Re)Negotiation

- 0.10
  - Linked with buffer allocation (comes from downstream)

- Problems
  - Slow
  - Doesn't work when upstream provides the buffers (ex: v4l2src)
(Re)Negotiation

- In 1.0, negotiation is entirely decoupled from buffer allocation
- GST_QUERY_ALLOCATION
- GST_EVENT_RECONFIGURE
(Re)Negotiation

- **GST_QUERY_ALLOCATION**
  - **Upstream**
    - caps, need_pool
  - **Downstream**
    - Creates pool if needed
    - Min/max buffers, alignment info, ...
    - GstMeta handled
    - GstAllocator
  - Back to upstream who decides what to do
GstBufferPool

- Provides a pool of re-usable buffers
- Avoid free/alloc overhead
- Control allocation
- Shared between elements
- Generic API
(Re)Negotiation

- GST_EVENT_RECONFIGURE
- Sent upstream
  - By elements when changes happen
  - By pads when (un-)linked
- Faster response
- Handle-able by all elements
Impact of change

- Application porting minimal
- 'Naive' plugin porting minimal
- Use fast-path without disturbance
- Allow usage of your plugins/hardware in all use-cases
- “Throw away the hacks”
  - Re-use existing features
Current status

- No more massive API/ABI breaks
- Freeze “really soon now” (tm) (c)
- All freedesktop modules ported
- Some external modules ported
- Applications ported
  - Problem of 0.10/1.0 dual usage
Example: TI PandaBoard 1.0

- Strided caps => GstVideoMeta
- Custom elements => gone
- Custom query => GST_QUERY_QUERY_ALLOCATION
- V4l2sink works out of the box
- pvrvideosink in -bad
- gst-ducati ported
Questions?

- http://gstreamer.freedesktop.org/
- Thank you!
- Bon appétit!