MappedByteBuffer.hurray()!

Programming the Linux Framebuffer in Java

Christopher Friedt
Principle Embedded Firmware Engineer

chris@mmbnetworks.com
chrisfriedt@gmail.com

code available at http://github.com/cfriedt
Overview

History
Hypothesis
Apparatus
Methods
Observations
Conclusions
Thanks
Questions
History

A long time ago… in a galaxy far, far away…

JAR WARS: Revenge of the Disc
Oracle vs. Google
Oracle claimed that Android had fragmented the Java API

- Android used the Java for apps (at least syntactically)
- new, different, and widely adopted Java windowing API
- Java ME ... wot ??
History

- I thought it would be a good idea to create an **AWT** port for Android
- **AWT - Abstract Windowing Toolkit**
- Each platform must implement AWT in order to support windows, buttons, forms, ...
- Oracle would have a lesser basis for their lawsuit against Google / Android OHA
Eventually saw the **Caciocavallo Project**

- [http://openjdk.java.net/projects/caciocavallo](http://openjdk.java.net/projects/caciocavallo)
- Framework for developing an AWT port
- Enables one to (possibly) create an AWT port entirely in Java
- Java is great for rapid prototyping
After some effort, I had written a few widgets, and then experienced **deja vu**.

- Ready to start painting to a screen!
- but then I couldn’t map the Linux Framebuffer
  - Needed to **use JNI to map** /dev/fb0
- and then I remembered Video For Linux (V4L) several years prior
  - Needed to **use JNI to map** /dev/video0
- Why duplicate code? Why doesn’t FileChannel.map() **Just Work™**?
  - because /dev nodes are “special”
History

Back to the Oracle vs. Google...

- Luckily, Google (kind of) won that lawsuit - for all of us!
- "So long as the specific code used to implement a method is different, anyone is free under the Copyright Act to write his or her own code to carry out exactly the same function or specification of any methods used in the Java API. It does not matter that the declaration or method header lines are identical."
- U.S. Copyright Act: 102(b) ... "system or method of operation."
- I was free to put my just for fun project on the back burner...
Hypothesis
Hypothesis

“With a small bit of hacking on weekends, I can get this to work in no time at all !!~!”

-- some idiot, 5 years ago
Hypothesis

- *Every OS implements mmap(2)*
- Java has **MappedByteBuffer**, via `FileChannel.map()`, extends ByteBuffer
- Able to get a FileChannel object with `RandomAccessFile.getChannel()`
- All of the above *works with a plain text file*, but do not work on `/dev/XXX`
- Unable to get **MappedByteBuffer.array()** object()
  - kind of required for direct pixelpushing
- The above is true for **any JVM**

* that I care to use
** that I have been able to test
Apparatus
Apparatus

JamVM
- Most familiar with hacking
- Squashing into embedded since 2006
- Easy to modify
- Multi-platform
- Multi-classpath
- Met the author at FOSDEM’12
- … straaangely similar to Dalvik …

GNU Classpath
- Most familiar with hacking
- Squashing into embedded since 2006
- Good code structure
- Several existing AWT implementations
- Easy to add a new, Framebuffer AWT
Apparatus

**VMWare**
- Would prefer to *not* need a separate test machine
- Easier for others to test on Mac / Linux / Windows
- Can easily run Linux in VMWare
- VMWare has *a* Linux framebuffer driver

**Linux**
- Most familiar with hacking
- I have been squashing it into stuff since 1998
- Pretty OK code structure…
- Linux runs on a few things…
- Easy to add a new stuff

* quasi-functional
Apparatus

Java
FB4JFrameBuffer fb = new FrameBuffer();
FB4JVarScreenInfo vinfo = fb.getVarScreenInfo();
vinfo.setXresVirtual( vinfo.getXres() );
vinfo.setYresVirtual( 2 * vinfo.getYres() );
fb.putVarScreenInfo( vinfo );
FB4JFixScreenInfo finfo = fb.getFixScreenInfo();
int[] pixel = fb.asByteBuffer().asIntBuffer().array();
final int w = vinfo.getXres();
final int h = vinfo.getYres();
final int hmax = vinfo.getYresVirtual();

Peanut Gallery
// ioctl via JNI / JNA
// ioctl via JNI / JNA
// relies on ByteBuffer .read() / .put()!
// relies on ByteBuffer .read() / .put()!
// ioctl via JNI / JNA
// ioctl via JNI / JNA
// calls FileChannel.map() on special node
Java
for( int yoffs = h ;; yoffs += h, yoffs %= hmax ) {
    // draw stuff
    fb.flip();
}

Peanut Gallery
// initially draw to 1 / N back buffers, loop 4 EVAR!!

// ioctl via JNI / JNA
// code available at http://github.com/cfriedt/fb4j
Methods
Ensure **VMWare / Linux Framebuffer** is accessible and can flip pages natively

- **Methods**
  
  ```c
  #include <sys/mman.h>
  void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);
  int munmap(void *addr, size_t length);
  ```

  ```c
  #include <sys/ioctl.h>
  int ioctl(int d, unsigned long request, ...);
  ```

  ```c
  #include <linux/fb.h>
  ```
Ensure VMWare / Linux Framebuffer is accessible and can flip pages natively

- int fd = open("/dev/fb0", O_RDWR);

- struct fb_var_screeninfo vinfo = {};
  ioctl( fd, FBIOGET_VSCREENINFO, &vinfo );

- vinfo.xres_virtual = vinfo.xres;
  vinfo.yres_virtual = 2 * vinfo.yres; // front & back buffer
  ioctl( fd, FBIOPUT_VSCREENINFO, &vinfo );

- size_t maplen = vinfo.xres_virtual * vinfo.yres_virtual * vinfo.bits_per_pixel / 8;
Ensure VMWare / Linux Framebuffer is accessible and can flip pages natively

- `uint8_t *map = mmap( NULL, maplen, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0 );`

- `uint16_t w = vinfo.yres;
  uint16_t h = vinfo.xres;
  for( uint16_t yoffs = h ;; yoffs += h, yoffs %= vinfo.yres_virtual ) {
    uint8_t *pixel = &map[ yoffs * w ];
    vinfo.yoffset = yoffs;
    // … draw stuff to back buffer
    ioctl( fd, FBIOPAN_DISPLAY, &vinfo );
  }`
Methods

Linux VMWare Framebuffer Driver

Two problems:

1) `fb_fix_screeninfo.line_length`

2) `FBIOPAN_DISPLAY` broken

```c
int vmw_fb_pan_display(
    struct fb_var_screeninfo *var,
    struct fb_info *info) {
    return 0;
}
// should at least return -ENOSYS!
```

https://bugs.gentoo.org/show_bug.cgi?id=494794
Methods (fix problem 1 in vmwgfx_fb.c)

Previously, the vmwgfx_fb driver would allow users to call FSET_VINFO, but it would not adjust the PINFO properly, resulting in distorted screen rendering. The patch corrects that behaviour.

See [https://bugs.gentoo.org/show_bug.cgi?id=494794](https://bugs.gentoo.org/show_bug.cgi?id=494794) for examples.

```c
if (vm_kms_validate_mode_vram(vm_priv,
               info->fix.line_length,
               var->xres * var->bits_per_pixel / 8,
               var->yoffset + var->yres)) {
   DRM_ERROR("Requested geom can not fit in framebuffer\n");
   return -EINVAL;
}
```
Methods (fix problem 2 in vmwgfx_fb.c)

https://bugs.gentoo.org/show_bug.cgi?id=496516
Methods

Ensure **Classpath** is able to

1) **mmap(2)** special files
2) FileDescriptor / VMChannel provides integer file descriptor, for JNI FB ioctls
   a) Available in most VMs for a very long time* (search kfu.com java file descriptor)
   b) Android may use a slightly different field name (other than “fd”)
3) Pass Pointer object from classpath to VM, portably, so VM can allocate array object
   a) New classes: **VMFlexArray, VMFlexArrayInfo**
4) **sun.misc.Unsafe** support
5) **Buffer** and subclasses
6) Object LifeCycle
Methods: Special Files

- `mmap(2)` special files
- `VMChannel.map()`
- Simple check to see if the file is special using `S_ISCHR()`, `S_ISBLK()`
- Aligning up with `mmap(2)` is not necessary on
  - Linux
  - Mac OS X

https://github.com/cfriedt/classpath/compare/mmap-special-files
Methods: File Descriptor

integer file descriptor

- VMChannel( final int native_fd )
- public so that it is accessible from FileDescriptor class

https://github.com/cfriedt/classpath/compare/add-integer-filedescriptor
Methods: File Descriptor

integer file descriptor

- `int fd`
- 2 new constructors required for FileChannelImpl VMChannel
- `isValid()`

https://github.com/cfriedt/classpath/compare/add-integer-filedescriptor
Methods: File Descriptor

- Make constructor visible to FileDescriptor
- Uncomment getNativeFD()

https://github.com/cfriedt/classpath/compare/add-integer-filedescriptor
Methods: VMFlexArray

VMFlexArray

- flexible because they allow both regularly allocated java arrays and arrays defined by arbitrary pointers
- uses system property `gnu.classpath.flexarray.enable`
- uses `sun.misc.Unsafe`, reflection
- statically initialized once upon VM init
- `static Object pointerToArray(Pointer address, int capacity, int array_offset, Class<?> cls);`

VMFlexArrayInfo

- Private static interface `IVMFlexArrayInfo`
- A VMFlexArrayInfo is needed for each VM that supports VMFlexArray
- String `jamvminfo = System.getProperty("java.vm.info");`
  `flexible = null == jamvminfo ? false : jamvminfo.contains( "flexarray" );`
- `private static interface IVMFlexArrayInfo {`
  `  boolean isArrayObjectFlexible();`
  `  int arraySizeOffset();`
  `  int dataPointerOffset();`
  `}`

https://github.com/cfriedt/classpath/compare/use-sun-misc-unsafe-for-pointer-arrays
Methods: sun.misc.Unsafe

Unsafe

Requires two additional method declarations in Classpath. Implementations are in the VM.

1) addressSize()
   a) used by VMFlexArrayInfo, VMFlexArray

2) allocateInstance()
   a) used by VMFlexArray

https://github.com/cfriedt/classpath/compare/use-sun-misc-unsafe-for-pointer-arrays
### Methods: Buffers, Views et al

#### Buffers & Views

ByteBuffer, CharBuffer, ShortBuffer, … really just rely on VMFlexArray.DirectpointerToArray() 

Views allow one type of buffer to be interpreted as having different types of elements, not unlike a cast.

---

**Buffers & Views**

ByteBuffer, CharBuffer, ShortBuffer, … really just rely on VMFlexArray.DirectpointerToArray()

Views allow one type of buffer to be interpreted as having different types of elements, not unlike a cast.

[https://github.com/cfriedt/classpath/compare/use-sun-misc-unsafe-for-pointer-arrays](https://github.com/cfriedt/classpath/compare/use-sun-misc-unsafe-for-pointer-arrays)
Methods: Mapped Array LifeCycle

VMDirectByteBuffer

- MappedByteBuffers umap their memory upon finalization – same as before
- DirectByteBuffers only free memory they allocate (and only if there is no backing buffer) upon finalization – same as before

https://github.com/cfriedt/classpath/compare/buffers
Ensure JamVM

1) Implements methods in sun.misc.Unsafe
2) Supports VMFlexArray
Methods: sun.misc.Unsafe

Additional Methods

- `int addressSize()`
  - 32-bit (4-bytes), 64-bit (8-bytes)
- `Object allocateInstance(Class<?> cls)`
  - allocate but do not initialize an object

https://github.com/cfriedt/jamvm/compare/additional-unsafe-methods
Methods: VMFlexArray

- Objects in JamVM:
  ```c
  typedef struct object {
    uintptr_t lock;
    Class *class;
  } Object;
  ```
- The Class type is simply an Object with data allocated after for the ClassBlock and MethodBlock – i.e. the Class type wraps around the Object type.

https://github.com/cfriedt/jamvm/compare/array-object-modifications
Methods: VMFlexArray Example: int[]

**Normal Array Object Layout @ 0x0**
- Object: 2 words: lock, Class *
- Array Length: 1 word
- Array Data₀: 1 word
- ... 
- Array Dataₙ₋₁: 1 word

**FlexArray Object Layout @ 0x0**
- Object: 2 words: lock, Class *
- Array Length: 1 word
- Array Pointer: 1 word: 0x16
- Array Data₀: 1 word @ 0x16
- ... 
- Array Dataₙ₋₁: 1 word

**Mapped FlexArray Object Layout @ 0x0**
- Object: 2 words: lock, Class *
- Array Length: 1 word
- Array Pointer: 1 word: 0x200
- Array Data₀: 1 word @ 0x200
- ... 
- Array Dataₙ₋₁: 1 word
Methods: VMFlexArray

- Similarly, array Objects simply wrap the Object structure
- Previously, JamVM was always responsible for allocating arrays contiguously
- 1 additional dereference is performed
- A pointer is reserved directly before the array data. If array is contiguous, pointer points to next word. Otherwise, points elsewhere (e.g. mmap’d data)

https://github.com/cfriedt/jamvm/compare/array-object-modifications
Methods: Build Configuration & Running

JamVM

- ./configure --enable-vm-flexarray …

Classpath

- ./configure --enable-vm-flexarray …

Linux

- Patch 1/2 of VMWare patchset is upstream
- I believe it rolls out with current Ubuntu
- Patch 2/2 (page flip) applied manually
- .Waiting for feedback from lkml

VMWare

- Download VM image
- The root password is empty
- SSH in from terminal
- modprobe vmwgfx
- fbset
  - xres 640 –yres 480
  - vxres 640 –vyres 960
- . ~/.bashrc; testfb4j
Observations
Observations

- Compared to respective Buffer.put(), using byte[] foo = MappedByteBuffer.array() and performing regular Java operations array elements speeds up Java code by a factor of up to 150x!
With JamVM, the bouncing ball demo achieves 21 up to 48 fps.

In C, the bouncing ball demo achieves up to 700 fps :-/
Conclusions
Conclusions

- Code works end-to-end with JamVM and GNU Classpath
- Very interested to try VMFlexArray with e.g. other class libraries, other VMs
  - would be useful to have JDWP and Java Profiling Agent support
- Large difference between Java and C fps indicates that JamVM needs performance optimizations
  - backport Dalvik’s JIT?
- Demo anyone?
Thanks!

- MMB Networks, for
  - having our monthly HackDay
  - encouraging me to do this sort of thing for fun
- Robert Lougher for writing JamVM, the Classpath developer community
- David Airlie (RedHat), and Thomas Hellström (VMWare) for
  - reviewing my patches to go upstream in the Linux Kernel
- Mario Torre an Roman Kennke
  - for organizing the Java DevRoom, speakers, etc
  - for starting the Caciocavallo project
- The Audience!