

High Performance AIR Applications

Mihai Corlan

Adobe Evangelist

December 3rd, 2008



This talk is not:

- A to do list of techniques
- Only about AIR applications

This talk:

- Can help you optimize any application
- Demonstrates generic tools and techniques
- Demonstrates AIR-specific tools and techniques

Poor questions about performance:

- Will my application be fast?
- Is AIR fast enough for writing real applications?
- Is AIR too slow to do x?

Better questions about performance:

- Which operations are performance-sensitive?
- What metric can I use to measure this sensitivity?
- How can I optimize my application to that metric?

Examples of performance-sensitive operations:

- Signal processing (image, sound, video)
- Rendering (large datasets, 3D)
- Searching
- Getting the data from the server
- Responding to user input

Good metrics are:

- **quantifiable**—you can produce a number for it
- **consistent**—when measured repeatedly
- **meaningful**—it correlates with what you care about

Problem:

- Apply a filter to a 100 MB image
- Display progress and allow user to cancel

Metrics:

- throughput (bytes/msec)
- memory use (bytes)
- response time (msec)

Measuring throughput:

- `start_msec = new Date().time`
- `filter();`
- `end_msec = new Date().time`
- `rate = imageSize_bytes / (end_msec - start_msec)`

Measuring memory use:

- Use a monitoring tool such as Activity Monitor on Mac, Process Explorer on Win
- Measure resident set size or working set private bytes

WS private bytes is the amount of RAM the OS is dedicating exclusively to your application in order to keep it executing

Response time:

- Start stopwatch when user takes action
- Wait for application to respond
- Stop stopwatch when UI shows response
- Latency of 100 msec or more is perceptible

Repeat until satisfied:

- Measure
- Analyze
- Modify

Broadly speaking you may modify either:

- Your design (high impact)
- Code (low impact)

Consider working with purpose-built test applications

- Full applications are complex and hard to understand
- Most of your application doesn't need to be optimized
- Keeps instrumentation out of your application
- You do need to validate that results port back


```
var timer:Timer = new Timer( 1, 1 );  
  
...  
  
var start:Number = new Date().time;  
while( workRemaining ) {  
    doWork();  
  
    var now:Number = new Date().time();  
    if(( now - start ) > TIME_LIMIT ) {  
        if( workRemaining )  
            timer.start();  
        break;  
    }  
}
```

What you are optimizing for may change

- New features may make different metrics important
- User expectations also evolve

Optimizations may have secondary impacts

- You may need to balance between various metrics

Optimizations effects may change

- Compiler changes might make alternate implementations even faster
- Libraries and codecs you depend on might change their characteristics
- Hardware changes

Flash Player 10 introduced Pixel Bender, 3D Transformations, vectors – these might improve your apps performance

Pixel Bender:

- Uses a highly efficient programming language for processing in parallel pixels
- In Flash Player runs only on CPU (there is no GPU processing)
- Runs in parallel on all the CPU's cores available

You can use it for crunching numbers. Instead of applying on bitmaps, you can apply on ByteArrays or Vectors:

- Mixing sounds, processing large sets of data
- Used in this mode, it runs in a separate thread than the main ActionScript thread -> you get an UI responsive for free

BlackBookSafe

- AIR 1.5 application
- Created with HTML/JavaScript
- Uses as3corelib for hashing
- Uses Pixel Bender, 3D transformations,

For example, if you do apps that works with mp3 files, it might be interesting for you to know that Flash Player works internally with these files at a 44Khz sample rate.

Thus, if you give other sample rate, the Player will resample – waste of CPU cycles.

Oliver Goldman's blog:

<http://blogs.adobe.com/simplicity/>

About Pixel Bender (Tinic Uro's blog):

<http://www.kaourantin.net/2008/05/adobe-pixel-bender-in-flash-player-10.html>

Examples of using Pixel Bender in Flex (also applied on sounds; Miti's blog):

<http://miti.pricope.com/2008/11/10/playing-with-pixel-bender/>

BlackBookSafe:

http://www.adobe.com/devnet/air/ajax/articles/blackbooksafe_anatomy.html

Optimizing Adobe AIR for Code Execution, Memory, and Rendering:

<http://www.craftymind.com/2008/11/20/max-2008-session-material/>

Some server changes can save your app:

<http://gregsramblings.com/2008/11/22/how-i-almost-crashed-the-adobe-max-2008-keynote-with-tour-de-flex/>

Thank you!

Email: mihai.corlan@adobe.com

Blog: <http://corlan.org>