🔟 アプリケーション自動化総合コミュニティフォーラム (Japan)





[ExtendScript] 高速化へのアプローチ

Posted by 10 A in アプリケーション自動化総合コミュニティフォーラム (Japan) on Apr 22, 2018 8:35:00 PM

正直に言います。Adobe社のアプリケーションに搭載されているJavaScriptはExtendScriptというECMA Scriptのver.3あたりを拡張した代物ですからアーキテクチャーがとても古くてインタプリタの性能も低く実行速度がとても遅いです。

しかしながら、そんな事ばかり言っているだけでは前に進めませんので色々と試す事はあります。今回はExtendScriptの弱点を踏まえた上でどう対応していくかの一例を用意しました。

Base64エンコーディング

Base64エンコーディングというのはバイナリファイルをAsciiテキストとして表現する処理です。これはメール送信時等原則的にテキストしか扱えない処理系でバイナリファイルを扱うためには必須のエンコーディング処理です。

処理の基本的な部分はとても単純で3 byte毎のコードを並べて24bitにして、それを6 bitづつに切り分けた上で特定のキャラクタに当てはめてテキスト化するという処理になります。です から

3byte(3×8bit=24bit)のバイナリはAsciiキャラクタ4コの表現となり32bitのデータとなります。メールに添付した画像の容量が実際の送信時に容量が大きくなるのはこの処理の為で、Base64処理を通るとファイル容量は4/3倍に増加します。このあたりの情報はウィキ等検索をかけると沢山出てきますから興味のある方は調べてみて下さい。

ExtendScriptでBase64処理を行う

では本題です。このBase64エンコーディング処理をExtendScriptでやってみるとどうなるか見てみます。 サンプルとして用意したコードはAdobe社のサンプルに含まれるものです。シンプルに書かれていますの で順番に追っていけば容易に理解できると思います。

```
EmailWithAttachment.prototype.encodeAttachment = function(binaryString)
01.
02.
03.
          // Accepted b64 chars
          var keyStr = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";
04.
05.
06.
07.
          var encoded = "";
          var c1, c2, c3;
08.
09.
          var e1, e2, e3, e4;
10.
          var i = 0;
11.
          while(i < binaryString.length)</pre>
12.
13.
              // Get 3 chars starting at the begining of the binary string
14.
15.
               c1 = binaryString.charCodeAt(i++);
16.
               c2 = binaryString.charCodeAt(i++);
```

```
17.
               c3 = binaryString.charCodeAt(i++);
18.
19.
20.
               // Create 4 sets of 6 bits from the 3 ascii bytes
21.
               e1 = c1 >> 2;
22.
               e2 = ((c1 \& 3) << 4) | (c2 >> 4);
23.
               e3 = ((c2 \& 15) << 2) | (c3 >> 6);
               e4 = c3 \& 63;
24.
25.
26.
27.
               // Check that chrs 2 and 3 are correct, if not replace with special padding char
28.
               if (isNaN(c2))
29.
30.
                   e3 = e4 = 64;
31.
               }
32.
               else if (isNaN(c3))
33.
               {
34.
                   e4 = 64;
35.
               }
36.
37.
               // Add encoded chars to the return string
38.
39.
               encoded = encoded + keyStr.charAt(e1) + keyStr.charAt(e2) +
40.
                   keyStr.charAt(e3) + keyStr.charAt(e4);
41.
          }
42.
43.
44.
          return encoded;
45.
      }
01.
      EmailWithAttachment.prototype.encodeAttachment = function(binaryString)
02.
03.
          // Accepted b64 chars
04.
          var keyStr = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";
05.
06.
           var encoded = "";
07.
08.
          var c1, c2, c3;
09.
           var e1, e2, e3, e4;
10.
           var i = 0;
11.
12.
          while(i < binaryString.length)</pre>
13.
               // Get 3 chars starting at the begining of the binary string
14.
15.
               c1 = binaryString.charCodeAt(i++);
               c2 = binaryString.charCodeAt(i++);
16.
17.
               c3 = binaryString.charCodeAt(i++);
18.
19.
               // Create 4 sets of 6 bits from the 3 ascii bytes
20.
21.
               e1 = c1 >> 2;
               e2 = ((c1 \& 3) << 4) | (c2 >> 4);
22.
23.
               e3 = ((c2 \& 15) << 2) | (c3 >> 6);
               e4 = c3 \& 63;
24.
25.
26.
27.
               // Check that chrs 2 and 3 are correct, if not replace with special padding char
28.
               if (isNaN(c2))
29.
               {
30.
                   e3 = e4 = 64;
31.
               }
```

```
32.
               else if (isNaN(c3))
33.
34.
                   e4 = 64;
35.
               }
36.
37.
38.
               // Add encoded chars to the return string
               encoded = encoded + keyStr.charAt(e1) + keyStr.charAt(e2) +
39.
40.
                    keyStr.charAt(e3) + keyStr.charAt(e4);
           }
41.
42.
43.
44.
           return encoded;
45.
       }
```

whileループでバイナリを順番に処理しencoded変数に収容するといった処理になっています。 今回用意したテスト用の画像が300kbです。これを上記のスクリプトを使いESTK上で処理してみます。

Base64 encode execution time = 251(sec)

となります。4分以上処理にかかるという結果となりました。実際にヒットカウンターを表示させると 以下のようになります。

```
\Theta \Theta \Theta
                                                                             base64.jsx
   ExtendScript Toolkit 2
                                                                    *
                                                                          main
                                                                                                                                   +
                                                                                                                                                    29
30
                 // Accepted b64 chars
31
                 var keyStr = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopgrstuvwxyz0123456789+/=";
32
33
                 var encoded = "";
34
                 var c1, c2, c3;
35
                 var e1, e2, e3, e4;
36
                 var i = 0;
37
38
                 while(i < binaryString.length)
39
40
                    // Get 3 chars starting at the begining of the binary string
41
                    c1 = binaryString.charCodeAt(i++);
42
                    c2 = binaryString.charCodeAt(i++);
43
                   c3 = binaryString.charCodeAt(i++);
44
                    // Create 4 sets of 6 bits from the 3 ascii bytes
45
                    e1 = c1 >> 2;
46
                    e2 = ((c1 & 3) << 4) | (c2 >> 4);
47
         39912
48
        148414
                    e3 = ((c2 & 15) << 2) | (c3 >> 6);
49
                    e4 = c3 & 63;
50
51
                    // Check that chrs 2 and 3 are correct, if not replace with special padding char
52
                    if (isNaN(c2))
53
                    {
54
                      e3 = e4 = 64;
55
56
                    else if (isNaN(c3))
57
                    {
58
                      e4 = 64;
59
60
61
                    // Add encoded chars to the return string
62
                    encoded = encoded + keyStr.charAt(e1) + keyStr.charAt(e2) +
63
                      keyStr.charAt(e3) + keyStr.charAt(e4);
64
65
                 var endDate = new Date();
66
                 $.writeIn("Base64 encode finish time = " + endDate);
67
                 return encoded;
68
                                                                                                                                  行65 列5
```

encoded変数へのアクセスで非常に負荷がかかっているのがわかります。

処理を変更してみる

上のスクリプトでenocded変数へ文字列を追加していく部分で大きな負荷がかかっている事がわかりました。300kbという容量は文字数にしてみると30万文字となります。もちろんアプリケーションキャッシュ等の環境にも関係する部分なのですが、基本的にはスクリプトエンジンに扱わせるには荷が重いデータです。数キロバイト程度の容量のファイルでテストしてみるとこういったパフォーマンスの低下は見られません。

これらを踏まえて書いたのが以下のコードです。

```
function base64(fileNm) {
01.
02.
          var theFile = new File(fileNm);
03.
           theFile.encoding = "binary";
04.
           theFile.open("r");
05.
           var binStr = theFile.read();
          theFile.close();
06.
           var flNm = "/tmpfile.txt";
07.
          var tmpFile = new File(flNm);
08.
09.
          tmpFile.open("w");
          szB64 = new Array;
10.
           var b64Str = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
11.
12.
           for (j=0;j<64;j++){
13.
               szB64[j] = keyStr.charAt(j);
14.
               }
15.
           var ecdStr = "";
16.
           var cntr = 0;
17.
           var binLen = binStr.length - (binStr.length % 3);
           for (i=0;i<binLen;i+=3) {</pre>
18.
19.
               ecdStr += szB64[binStr.charCodeAt (i) >> 2];
               ecdStr += szB64[((binStr.charCodeAt (i) & 0x3) << 4) | (binStr.charCodeAt (i+1) >>
20.
               ecdStr += szB64[((binStr.charCodeAt (i+1) & 15) << 2) | (binStr.charCodeAt (i+2) >
21.
22.
               ecdStr += szB64[binStr.charCodeAt (i+2) & 0x3f];
23.
               cntr +=4;
24.
               if(cntr > 75) {
25.
                   tmpFile.writeln (ecdStr);
                   ecdStr = "";
26.
27.
                   cntr = 0;
28.
29.
               }
30.
           switch (binStr.length % 3){
31.
               case 2: //padding 2 characters.
32.
               ecdStr += szB64[binStr.charCodeAt (i) >> 2];
               ecdStr += szB64[((binStr.charCodeAt (i) & 0x3) << 4) | (binStr.charCodeAt (i+1) >>
33.
               ecdStr += szB64[((binStr.charCodeAt (i+1) & 15) << 2) | (binStr.charCodeAt (i+2) >
34.
               ecdStr += "=";
35.
               tmpFile.writeln (ecdStr);
36.
37.
               break;
38.
               case 1: //padding 1 character.
39.
               ecdStr += szB64[binStr.charCodeAt (i) >> 2];
               ecdStr += szB64[((binStr.charCodeAt (i) & 0x3) << 4) | (binStr.charCodeAt (i+1) >>
40.
               ecdStr += "=="
41.
               tmpFile.writeln (ecdStr);
42.
43.
               break;
44.
               default:
45.
               tmpFile.writeln (ecdStr);
               break;
46.
47.
               }
          tmpFile.close();
48.
49.
           }
50.
          }
```

```
51.
           tmpFile.close();
52.
           }
01.
      function base64(fileNm) {
          var theFile = new File(fileNm);
02.
03.
          theFile.encoding = "binary";
04.
          theFile.open("r");
05.
           var binStr = theFile.read();
          theFile.close();
06.
07.
           var flNm = "/tmpfile.txt";
           var tmpFile = new File(flNm);
08.
09.
           tmpFile.open("w");
10.
           szB64 = new Array;
           var b64Str = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
11.
12.
           for (j=0;j<64;j++){
13.
               szB64[j] = keyStr.charAt(j);
14.
           var ecdStr = "";
15.
          var cntr = 0;
16.
17.
           var binLen = binStr.length - (binStr.length % 3);
           for (i=0;i<binLen;i+=3) {</pre>
18.
19.
               ecdStr += szB64[binStr.charCodeAt (i) >> 2];
20.
               ecdStr += szB64[((binStr.charCodeAt (i) & 0x3) << 4) | (binStr.charCodeAt (i+1) >>
               ecdStr += szB64[((binStr.charCodeAt (i+1) & 15) << 2) | (binStr.charCodeAt (i+2) >
21.
22.
               ecdStr += szB64[binStr.charCodeAt (i+2) & 0x3f];
23.
               cntr +=4;
               if(cntr > 75) {
24.
25.
                   tmpFile.writeln (ecdStr);
                   ecdStr = "";
26.
27.
                   cntr = 0;
                   }
28.
29.
30.
           switch (binStr.length % 3){
31.
               case 2: //padding 2 characters.
               ecdStr += szB64[binStr.charCodeAt (i) >> 2];
32.
33.
               ecdStr += szB64[((binStr.charCodeAt (i) & 0x3) << 4) | (binStr.charCodeAt (i+1) >>
34.
               ecdStr += szB64[((binStr.charCodeAt (i+1) & 15) << 2) | (binStr.charCodeAt (i+2) >
               ecdStr += "=";
35.
               tmpFile.writeln (ecdStr);
36.
37.
               break;
38.
               case 1: //padding 1 character.
39.
               ecdStr += szB64[binStr.charCodeAt (i) >> 2];
40.
               ecdStr += szB64[((binStr.charCodeAt (i) & 0x3) << 4) | (binStr.charCodeAt (i+1) >>
               ecdStr += "=="
41.
42.
               tmpFile.writeln (ecdStr);
43.
               break;
44.
               default:
45.
               tmpFile.writeln (ecdStr);
46.
               break;
47.
48.
          tmpFile.close();
49.
           }
50.
51.
           tmpFile.close();
          }
52.
```

前出のものとの大きな違いは処理したエンコード済文字列は1行分溜まったらテンポラリファイルに出力してしまって、変数上で大きなデータを保持しないようにしたところです。こちらもトリッキーな処理は行っていませんからスクリプトの構造は容易に追えるとおもいます。

これを先ほどと同じ画像を処理してみます。

Base64 encode execution time = 7(sec)

実行時間が大幅に短縮されたことがお分かりいただけるでしょう。実際にヒットカウンターを見てみる と2ケタ低下しているのが分かります。

```
🖶 Base64_filestream.jsx
\Theta \Theta \Theta
                                                                                                                                +
ExtendScript Toolkit 2
                                                                        main
2
               function base64(fileNm) {
3
                var startDate = new Date();
                 $.writeIn("Base64 encode start time= " + startDate);
 4
 5
                 var the File = new File (file Nm);
 6
                 theFile.encoding = "binary";
                 theFile.open("r");
                 var binStr = theFile.read();
                 theFile.close();
10
                 var flNm = "/tmpfile.txt";
                 var tmpFile = new File(flNm);
11
12
                 tmpFile.open("w");
13
                 szB64 = new Array;
14
                 var keyStr = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"; // Accepted b64 chars
                 for (j=0;j<64;j++){
15
16
                   szB64[j] = keyStr.charAt(j);
17
                 var ecdStr = "";
18
19
                 var counter = 0:
20
                 var binLen = binStr.length - (binStr.length % 3);
21
                 for (i=0;i<binLen;i+=3) {
                                              // Get 3 chars starting at the begining of the binary string
22
        123550
                    ecdStr += szB64[binStr.charCodeAt (i) >> 2];
23
                    ecdStr += szB64[((binStr.charCodeAt (i) & 0x3) << 4) | (binStr.charCodeAt (i+1) >> 4)];
                   ecdStr += szB64[((binStr.charCodeAt (i+1) & 15) << 2) | (binStr.charCodeAt (i+2) >> 6)];
24
25
                    ecdStr += szB64[binStr.charCodeAt (i+2) & 0x3f];
        121769
26
         15822
                    counter +=4;
27
                    if(counter > 75) {
                                          // when hit the limit, write the line and reset paramater
28
         24036
                      tmpFile.writeIn (ecdStr);
29
          1167
                      ecdStr = "";
30
                      counter = 0;
31
32
33
34
                 switch (binStr.length % 3){
35
                    case 2: //padding 2 characters
                    ecdStr += szB64[binStr.charCodeAt (i) >> 2];
36
37
                    ecdStr += szB64[((binStr.charCodeAt (i) & 0x3) << 4) | (binStr.charCodeAt (i+1) >> 4)];
                    ecdStr += szB64[((binStr.charCodeAt (i+1) & 15) << 2) | (binStr.charCodeAt (i+2) >> 6)];
38
                    ecdStr += "=";
39
40
                    if (ecdStr.length>75){
41
                      tmpFile.writeIn(ecdStr.slice (0, 76));
42
                      tmpFile.writeIn(ecdStr.slice (76));
43
                      } else {
                         tmpFile.writeIn (ecdStr);
45
                         }
46
                   break:
                    case 1: //padding 1 character.
47
                                                                                                                                行 28 列 36
```

この様にExtendScriptでは変数に大きなデータを喰わせると極端にパフォーマンスが低下する場合があります。もしスクリプトの実行時間が遅く感じたら、まずはヒットカウンターをチェックして、何処の処理に負荷がかかっているかを検証することが大事です。

しかしながら、ExtendScriptはインタープリタであり実行時にスクリプトエンジンがコードを逐一読み込みながら実行されるものです。基本的に実行時間が遅いものだと考えなければいけません。

もし、さらなる高速化を望む場合はCEPエクステンションやExternalObject等を検討する必要がある事も覚えておきましょう。

954 Views Tags: javascript, extendscript