Image speed tests

Image used in the resize tests 1&2 - 1,500px wide x 2,000px high x 1,987kb

Test 1

Speed tests for resizing a photo to a 200px high jpg thumbnail keeping the aspect ratio on a web server.
Imagemagick - resize, thumbnail and thumbnail with a jpg hint.
Imagick - thumbnailImage and resizeImage
GD – ImageCopyResampled

Test 2

Speed tests for resizing a photo to a 600px high jpg keeping the aspect ratio on a web server.
Imagemagick - resize, thumbnail and thumbnail with a jpg hint.
Imagick - thumbnailImage and resizeImage
GD - ImageCopyResampled

Test 3

Loop through the Imagemagick resize code to get 5 lots of data.
Using a web server and localhost on a PC.

Test 4

Create a text image
Using Imagemagick, Imagick and GD on a PC localhost and web server .

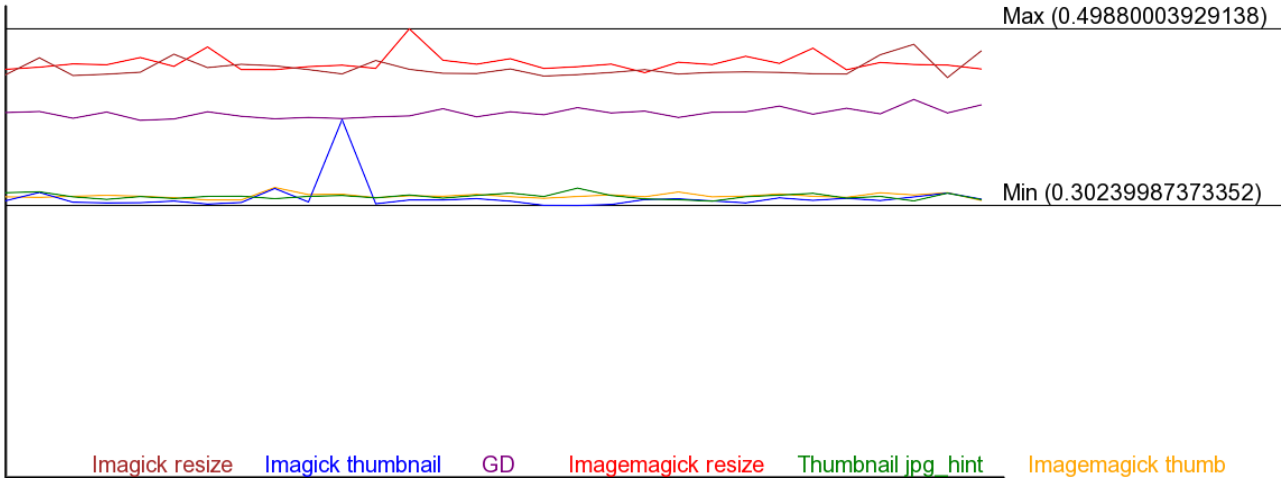
Test 1

Speed tests for resizing a photo to a 200px high jpg thumbnail keeping the aspect ratio.
The resize code was run 30 times for each resize method on a web server.

Method	Average time
Imagick thumbnail	0.31130666732788
php exec() imagemagick jpg hint -thumbnail	0.31260333061218
php exec() -thumbnail	0.31316999594371
GD	0.40514667034149
Imagick resize	0.45480666955312
php exec() Imagemagick -resize	0.46112333138784

Method	File size
GD	7492 bytes
Imagick thumbnail	30493 bytes
php exec() -thumbnail	30493 bytes
php exec() imagemagick jpg hint -thumbnail	30493 bytes
php exec() Imagemagick -resize	41352 bytes
Imagick resize	41352 bytes

Graph showing times to process each image using all 6 methods



The code used:

exec() Imagemagick -resize **:**

```
exec("/usr/local/bin/convert $image -resize 200x200 $output ");
```

exec() Imagemagick -thumbnail **:**

```
exec("/usr/local/bin/convert $image -thumbnail 200x200 $output ");
```

exec() Imagemagick –thumbnail with jpg hint **:**

```
exec("/usr/local/bin/convert -define jpeg:size=200x200  $image -thumbnail 200x200 $output
");
```

Before v6.5.6-0 you would have used -size 200x200 instead of -define jpeg:size=200x200

GD:

```
// Get the size of the original image into an array
$size = getimagesize( $image );
// Set the new width of the image
$thumb_width = "150";
// Calculate the height of the new image to keep the aspect ratio
$thumb_height = ( int )(( $thumb_width/$size[0] )*$size[1] );
// Create a new true color image in the memory
$thumbnail = ImageCreateTrueColor( $thumb_width, $thumb_height );
// Create a new image from file
$src_img = ImageCreateFromJPEG( $image );
// Create the resized image
ImageCopyResampled( $thumbnail, $src_img, 0, 0, 0, 0, $thumb_width, $thumb_height,
$size[0], $size[1] );
// Save the image as resized.jpg
ImageJPEG( $thumbnail, $output );
// Clear the memory of the tempory image
ImageDestroy( $thumbnail );
```

Imagick thumbnail:

```
$thumbnail = new Imagick( $image );
$thumbnail->thumbnailImage( 200,200, 1 );
$thumbnail->writeImage( $output );
```

Imagick resize:

```
$thumbnail = new Imagick( $image );
$thumbnail->resizeImage( 150,200, imagick::FILTER_LANCZOS, 1 );
$thumbnail->writeImage( $output );
```

Notes:

Imagick thumbnail keeps the aspect ratio automatically but resize does not. There is a best fit option but it gives me an error; Imagick needs more documentation

The standard Imagemagick filter is Lanczos which was specified in Imagick resize.
I do not know what the standard filter for Imagick thumbnail and GD and there is no method to specify the filter.

The quality settings for each method were left as standard as that is how most users will leave it.
Imagemagick - The default 92
Imagick – assume the same as Imagemagick
GD - The default is the default IJG quality value about 75.

The thumbnail images had a smaller file size than resize in Imagemagick and Imagick as the EXIF data is striped from the image although the colour profile is left in later versions.

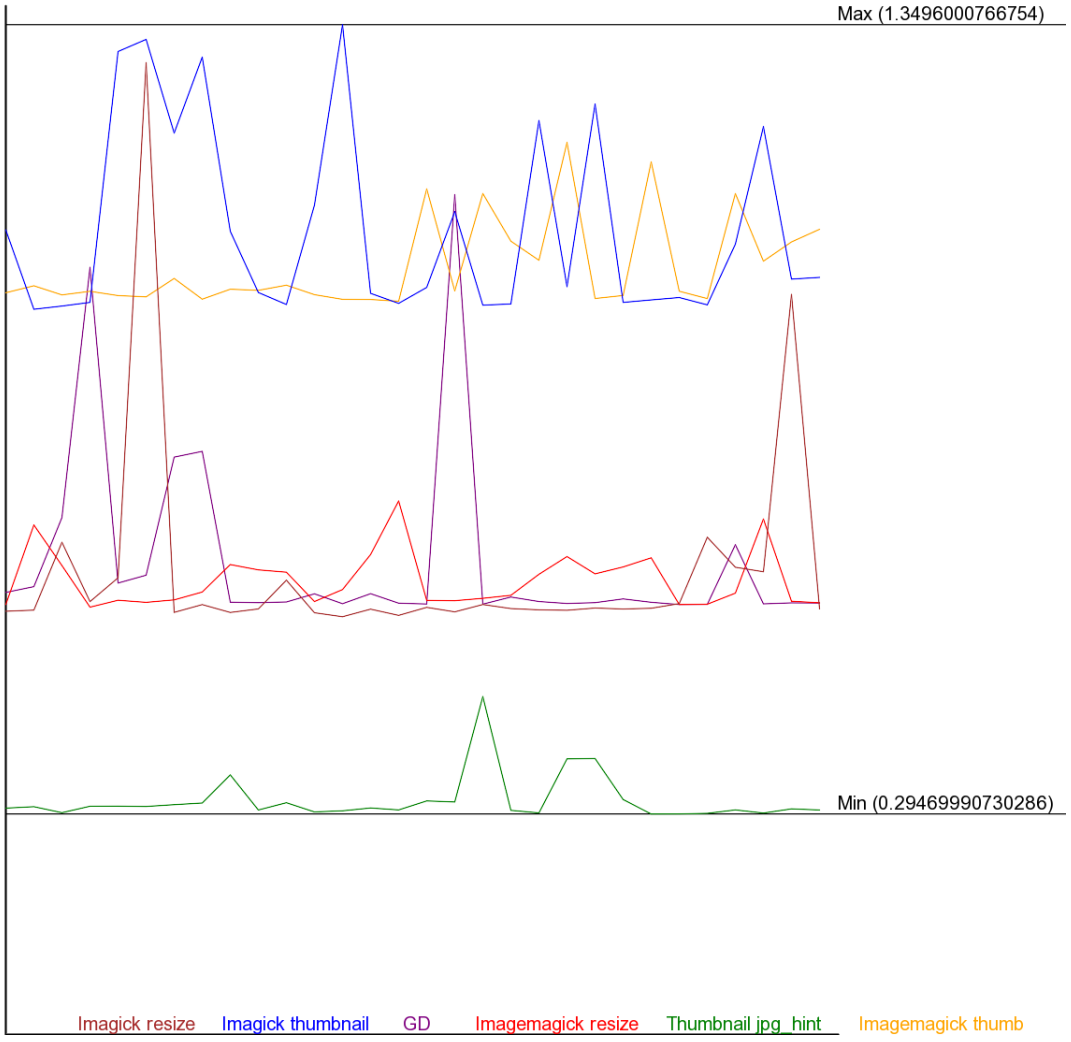
Test 2

Speed tests for resizing a photo to a 600px high jpg keeping the aspect ratio.
The resize code was run 30 times for each resize method on a web server.

Method	Average time
php exec() imagemagick jpg hint -thumbnail	0.31332666873932
php exec() Imagemagick -resize	0.60761333306630
Imagick resize	0.61915999253591
GD	0.63410999774933
php exec() -thumbnail	1.02709000110630
Imagick thumbnail	1.07819999853770

Method	File size
GD	39684 bytes
Imagick thumbnail	204875 bytes
php exec() -thumbnail	204875 bytes
php exec() imagemagick jpg hint -thumbnail	204982 bytes
php exec() Imagemagick -resize	216005 bytes
Imagick resize	216005 bytes

Graph showing times to process each image using all 6 methods



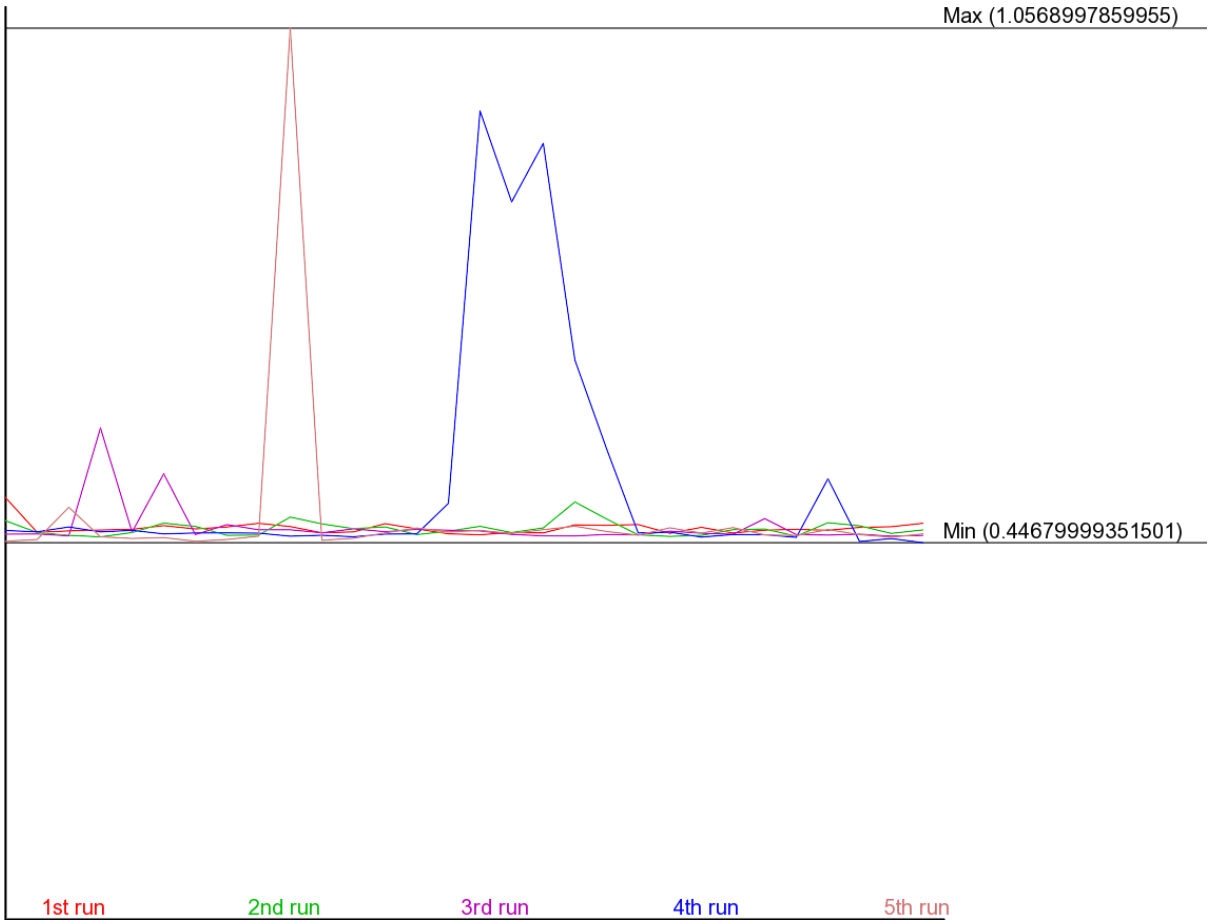
Test 3

Loop through the Imagemagick resize 30 times code to get 5 lots of data

The results from a server:

1st run	0.46436332861582 average time
2nd run	0.46346000035604 average time
3rd run	0.46554333368937 average time
4th run	0.51569666862488 average time
5th run	0.47780333360036 average time

Graph showing times to process each image on the server

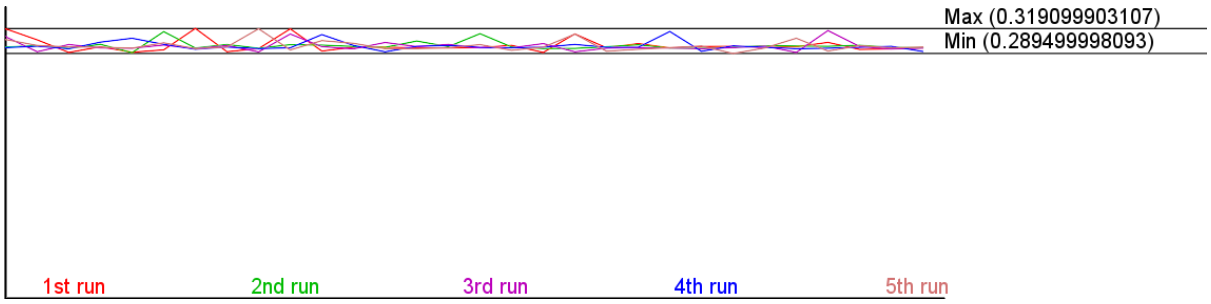


The results from a PC:

(Running php through XAMPP with no other programs running.)

1st run	0.299100001653 average time
2nd run	0.298720002174 average time
3rd run	0.298156666756 average time
4th run	0.298293328285 average time
5th run	0.298613333702 average time

Graph showing times to process each image on the PC



Test 4

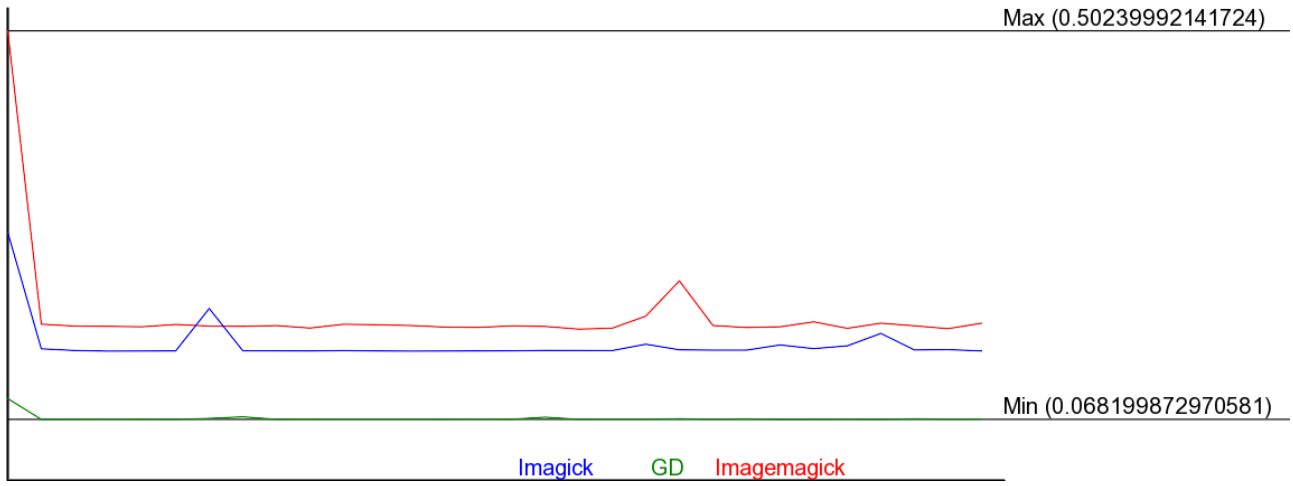
Create this text image using Imagemagick, Imagick and GD.

The code was run 30 times for each method on a PC running php through XAMPP with no other programs running and a server.

The results from a server:

GD	0.0440500020981 average time
Imagick	0.15247333049774 average time
Imagemagick	0.18567333221436 average time

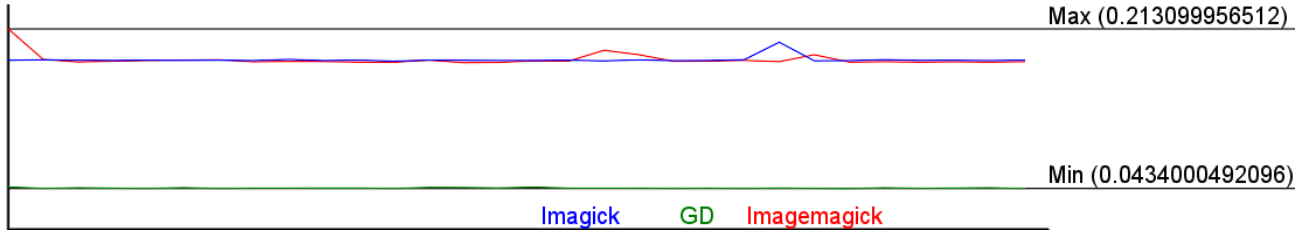
Graph showing times to process each image on the server



The results from a PC:

GD	0.069430001576742 average time
Imagick	0.18038333257 average time
Imagemagick	0.180669999123 average time

Graph showing times to process each image on the PC



The code used:

Imagemagick:

```
exec(" convert -size 933x93 xc:white -fill skyblue -stroke black -strokewidth 1 -undercolor lightpink -font ./arial.ttf -pointsize 96 -gravity center -annotate 0x0+0+0 \"THIS IS SOME TEXT\" -trim -bordercolor white -border 10 +repage $output.png");
```

Imagick:

```
$im = new Imagick();
$draw = new ImagickDraw();
$draw->setFont('./arial.ttf');
$draw->setFontSize( 96 );
$fillcolor = new ImagickPixel( "skyblue" );
$draw->setFillColor( $fillcolor );
$draw->setStrokeWidth( 1 );
$strokecolor = new ImagickPixel( "black" );
$draw->setStrokeColor( $strokecolor );
$undercolor = new ImagickPixel( "pink" );
$draw->setTextUnderColor( $undercolor );
$draw->setGravity( Imagick::GRAVITY_CENTER );
$bgcolor = new ImagickPixel( "white" );
$text = 'THIS IS SOME TEXT';
$im->newImage($width, $height, $bgcolor );
$im->annotateImage($draw, 0, 0, 0, $text);
$im->trimImage(0);
$bordercolor = new ImagickPixel( "white" );
$im->borderImage( $bordercolor, 10, 10);
$page = array();
$page = $im->getImagePage();
$owidth = $page["width"];
$oheight = $page["height"];
$im->setImagePage( $owidth, $oheight, 0, 0);
$page = array();
$page = $im->getImagePage();
$im->setImageFormat("png");
$im->writeImage( $output );
```

GD:

```
// Function to add the black border around the text
// http://www.johnciacia.com/2010/01/04/using-php-and-gd-to-add-border-to-text/
function imagettfstroketext(&$image, $size, $angle, $x, $y, &$textcolor, &$strokecolor, $fontfile, $text, $px) {

    for($c1 = ($x-abs($px)); $c1 <= ($x+abs($px)); $c1++)
        for($c2 = ($y-abs($px)); $c2 <= ($y+abs($px)); $c2++)
            $bg = imagettftext($image, $size, $angle, $c1, $c2, $strokecolor, $fontfile, $text);

   return imagettftext($image, $size, $angle, $x, $y, $textcolor, $fontfile, $text);
}

// Path to the font you are going to use
$font = "arial.ttf";
// Text to write
```

```
$text = "THIS IS SOME TEXT";
// Font size
$size = "70";
// Calculate the size of the text to center on the background
// If php has been setup without ttf support this will not work.
// Fudge added to center text as it was offset in x and y
$box = imagettfbbox( $size, 0, $font, $text );
$x = ((933 - ($box[2] - $box[0])) / 2)-20;
$y = ((93 - ($box[1] - $box[7])) / 2);
$y -= $box[7];
// Canvas size
$width = ($box[2] - $box[0]);
$height = ($box[1] - $box[7]);
// Create the canvas
$canvas = imagecreate( ($width+30), ($height+20) );
// First colour - this will be the default colour for the canvas
$white = imagecolorallocate( $canvas, 255, 255, 255 );
// The second colour - to be used for the text background
$pink = imagecolorallocate( $canvas, 255, 182, 193 );
// The third colour - to be used for the text outline
$black = imagecolorallocate( $canvas, 0, 0, 0 );
// The fourth colour - to be used for the text
$blue = imagecolorallocate( $canvas, 173, 216, 230 );
// Create the pink background
imagefilledrectangle($canvas, 10, 10, ($width+20), ($height+10), $pink);
// Add the text to the image
imagettfstroketext( $canvas, $size, 0, $x, $y, $blue, $black, $font, $text, 2 );
// Save as Text.png
imagejpeg( $canvas, "Text.png" );
// Clear the memory of the tempory image
ImageDestroy( $canvas );
```

Notes:

The GD version had to use a user function and the code is not as neat as the Imagemagick version.

I also had to add a lot of “fudges” to get a similar output image to the Imagemagick version.

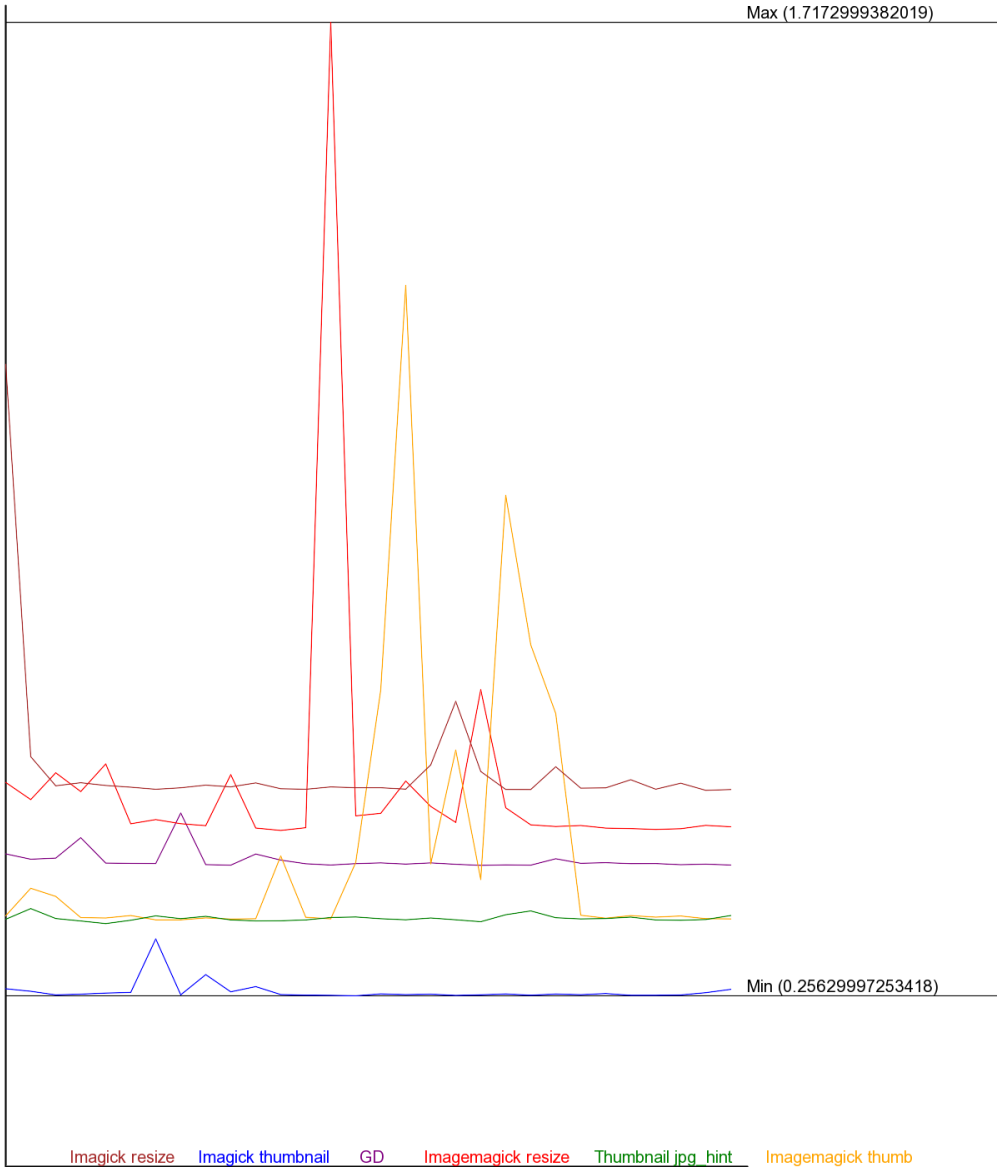
Even with all the extra code GD was a lot faster.

Conclusions

1/ Server load can have a large effect on the times as can be seen in this re-run of Test 1

Method	Average time
Imagick thumbnail	0.26339000066121
php exec() -thumbnail	0.48310666879018
php exec() imagemagick jpg hint -thumbnail	0.37271333535512
php exec() Imagemagick -resize	0.57697333494822
GD	0.46004333496094
Imagick resize	0.59951999982198

Graph showing times to process each image using all 6 methods



2/ If you are only resizing one image the time can be longer than when the time is averaged out over a batch.

3/ It was noticed in some previous use of Imagemagick that the next resize etc. can start before the previous one has finished.

4/ Some of the results are very close and another run may swap a couple of the close methods.

5/ Nearly twice as quick on the PC localhost but the server hardware/setup may not be as good as the PC.

6/ Depending what action you are using can depend on the software/method to use.

7/ The results from test 1 and 2 have very different results and the only change was to the final image size.