Subject: xsl transformation slow
Posted by semcycle on Wed, 20 Sep 2006 20:49:01 GMT
View Forum Message <> Reply to Message

a list screen of 100 records times out on a P3 server unless you allow the script to run longer than 30 seconds. then the page comes up with (XSLT = 34+ seconds)

to demonstrate this i have added records to the person database of the radicore demo

http://www.radicore.org/demo/

 if you go to the person_list screen and hit the list 100 link you end up with a lost network connection because the script times out.

is there something in the xsl transformation that can be optimized ?
the transformation should not be so hard on the processor, should it

Subject: Re: xsl transformation slow
Posted by AJM on Wed, 20 Sep 2006 22:19:20 GMT
View Forum Message <> Reply to Message

The more records you ask it to process the longer it takes, so the use of such large record sets should be avoided. Is there a good reason why you feel the need to display 100 records at a time?

Unfortunately the speed of XSL transformations with PHP 4 is quite slow compared with PHP 5, so I can only offer the following advice:

(a) Avoid such large record sets.
(b) Upgrade to PHP 5
(c) Upgrade your hardware.

Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 02:18:19 GMT
View Forum Message <> Reply to Message

Quote:(a) Avoid such large record sets.

Never underestimate the ability of an end user to do what should not be done.

Quote:
(b) Upgrade to PHP 5
(c) Upgrade your hardware.

This is not always an option for remotely hosted sites.

Actually, there is a very simple optimisation that will speed up the xsl transformation by a factor of around 300%.

I had a quick look at the std_std.data_field.xsl file and figured that the lions share of the processing would be done by the template called "display_horizontal" (for the list1 transaction pattern at least).

The thing that struck me right away was that inside the for_each loop a variable called "field" is declared and is selected starting from the root of the document every time. I've learned from hair-pulling experience that this is to be avoided whenever possible. It basically means that in order to create this variable, the ENTIRE xml document must be searched (which is what the "//" means) every time the variable is created. For large record sets this can be prohibitive. The obvious solution is to whittle down what needs to be searched before the variable is declared.

the offending line is this....

<xsl:variable name="field" select="//*[name()=$table][position()=$position]/*[name()=$fieldname]" />

Now most of that select changes on each iteration of the for_each loop due to the position() function, but the very beginning can be replaced by a variable declared outside the loop, e.g

<xsl:variable name="table_tmp" select="//*[name()=$table]" />

This replaces the corresponding code in the variable declaration for "field" like so...

<xsl:variable name="field" select="$table_tmp[position()=$position]/*[name()=$fieldname]" />

..and voila. A few very quick tests showed improvements from around 250-300% in xsl transformation times. I'm sure the same principle can be applied in other places for an even greater increase. The important thing to remember is to avoid using "//" repetitively if at all possible.

My apologies if this sounded like a lecture for those who know how xsl works. It was written for those who don't. Obviously Tony will know what I'm talking about which is all that counts

For those interested in learning more about XSL (and you should, it's very cool) I can thoroughly recommend a book I bought on the subject called "Beginning XSLT 2.0: From Novice to Professional" by Jeni Tennison. A most excellent tutorial style book that is worth its weight in gold.

Cheers,
Craig

Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 05:13:04 GMT
View Forum Message <> Reply to Message

After some further testing I've found the following:

a)The speed DECREASE due to record size is not linear, seems to be more like a squared increase in time. i.e. double the records = quadruple the time.

b) The speed INCREASE is consistent regardless of recordset size for a given table. It's as big as 400% on some tables in my test db. 15 seconds down from from over 60.

c)The size of the individual records also seems to affect it as much as the size of the recordset. This is probably a bigger worry than the large recordsets, since a user can't change the size of a record.

This would suggest one further optimisation from a developers perspective may be to limit the fields returned from a query if they're not all displayed on the list screen. i.e. hidden fields.

For example if you have 6 fields in a table, 3 of which are hidden(as many are in the radicore example system, mostly dates), then instead of the default ...

$sql_select = null;

which selects all fields in the table including hidden ones, you could perhaps have something like...

$sql_select = field1, field2, field2;

where fields 4, 5 & 6 are hidden and not displayed anyway. Naturally this requires more maintenance, so you'd need to weigh up the benefits of quicker display of larger recordsets versus the extra maintenance.

One thing I noticed during my testing was that the number of pages that benefited from changing 2 lines of code was huge. Every single page that shows multiple records in a horizontal format was affected. Further testament to Radicore's very cool design.

Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 05:59:23 GMT
View Forum Message <> Reply to Message

BTW, the server performing these transformations is a remote host and is running php4.

When I asked if there were any plans to upgrade to php5, well... their answer was polite, but you could tell they were laughing. Seems Tony's not the only who doesn't like php5's "improvements".

Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 07:25:11 GMT
View Forum Message <> Reply to Message

OK, I've found one further significant increase in speed, but it's entirely dependant on the answer to this question...

Is the relative position of the <structure> tag in any xml document that uses the "display_horizontal" template in it's xsl transformation ever going to be anything other than /root/structure ? If so then never mind.

If not then read on...

Again, the use of "//" has crippled the performance of the transformation. The xsl:for-each loop I referred to in my original post has the following select criteria...

<xsl:for-each select="//structure/*[name()=$zone]/row/cell[@field]">

what this selection does is to search the entire xml document,including the table data, for a <structure> tag at any level however deeply buried, before applying the remaining selection criteria of...

/*[name()=$zone]/row/cell[@field]

Now for smaller documents (i.e. recordsets) this is not an issue, but for very large documents the overhead is enormous and totally unneccesary since the table data, which is the vast majority of the information in the document, will never contain the <structure> tag.

If the relative position of the <structure> tag never changes, or at least is known at runtime, then the for_each criteria can be changed to this...

<xsl:for-each select="/root/structure/*[name()=$zone]/row/cell[@field]">

This allows the transformation engine to go directly to the to the <structure> tag without searching the entire document. It's like the difference between giving someone specific street directions to your house or simply telling them the suburb you live in and letting them work it out from there.

The effect of this is to give a further 200% increase in speed for a total of almost a whopping 600%. Yes, that's not a typo, nearly a 6-fold improvement in xsl transformation time. The transformation times of my largest tables for 100 records are now about 12 seconds down from over 60. Needless to say, smaller recordsets are now like greased lightning

If this logic is also applied to the table_tmp variable created in my original post

<xsl:variable name="table_tmp" select="/root/*[name()=$table]" />
instead of
<xsl:variable name="table_tmp" select="//*[name()=$table]" />
then the transformation time drops again to 8 seconds. But I suspect the data structure for this might not be so fixed, so this one is of dubious validity.

I have no idea if any of these changes have ramifications for the rest of the system, only Tony would know that, but isn't it worth investigating a 6-fold increase in speed for bugger-all effort?

Subject: Re: xsl transformation slow
Posted by AJM on Mon, 19 Mar 2007 09:20:33 GMT
View Forum Message <> Reply to Message

Very interesting. If the line of code which reads

<xsl:for-each select="//structure/*[name()=$zone]/row/cell[@field]">

can be made to run significantly faster by changing it to

<xsl:for-each select="/root/structure/*[name()=$zone]/row/cell[@field]">

then I will look into making this change in all the relevant XSL files. For most of the time I do actually know the full pathname of the node that is being processed, so changing "//" to "/root/" will not be a problem.

Thanks for the tip.

Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 09:25:28 GMT
View Forum Message <> Reply to Message

No problem.

Don't forget to check out the one in my first post as well. It seems to make the biggest difference.

Could you let us know how you get on? I'd be interested to know.

Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 14:28:53 GMT
View Forum Message <> Reply to Message

braingeyser wrote on Mon, 19 March 2007 03:25

If this logic is also applied to the table_tmp variable created in my original post

<xsl:variable name="table_tmp" select="/root/*[name()=$table]" />
instead of
<xsl:variable name="table_tmp" select="//*[name()=$table]" />

then the transformation time drops again to 8 seconds. But I suspect the data structure for this might not be so fixed, so this one is of dubious validity.

Ok scratch this one. As I suspected the data structure is not fixed sufficiently to make this work for all the places it's used. For example a link1, which has the link table data nested within the outer table data, making the above XPath incorrect for the nested data. The existing XPath is the better for keeping things generic.

If you wanted to make this work, you would need to pass in the context node from the external calling code. I tried this and it works fine, but it means adding an extra parameter in all of the places this template is called. Not a trivial amount of work I'd guess. Doing this makes the whole template a lot clearer though since you do away with most of the complicated xpaths.

If you add this...
<xsl:param name="context_node" />
then you don't need this...
<xsl:variable name="table_x" select="//*[name()=$table]" />

and you can replace this...
<xsl:variable name="field" select="$table_x[position()=$position]/*[name()=$fieldname]" />

with this...
<xsl:variable name="field" select="$context_node/*[name()=$fieldname]" />

The context node parameter (in the case of list1 for eg) comes from std.list1.xsl here...
<xsl:for-each select="//*[name()=$main][count(*)>0]">

```
    !-- display all the fields in the current row -->
    <xsl:call-template name="display_horizontal">
        <xsl:with-param name="zone" select="'main'"/>
        <xsl:with-param name="context_node" select="." />
    </xsl:call-template>
</xsl:for-each>
```

This is the code that would need to be changed in every template call in order to make it work. It speeds things up because your'e no longer getting the same table data node twice, once in the outer calling loop and once more in the display_horizontal template.

The extra time gains seem to be in the order of double, which is not a lot more than you would already get by implmenting the previous changes. In my case, I gained an extra 4 or 5 seconds on my originally 60+ sec page load, now down to between 7 & 8.

Subject: Re: xsl transformation slow
Posted by AJM on Mon, 19 Mar 2007 14:56:52 GMT
View Forum Message <> Reply to Message

I cannot guarantee that a table name will always be directly under the root node. In a parent/child

relationship the child table is under the parent node and not the root node.

Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 14:58:46 GMT
View Forum Message <> Reply to Message

heh, good timing. I just discovered that myself. I was editing the above post when you posted before me

There is a solution however...scroll up a bit

Even if you totally disregard this stuff, it's been an interesting exercise.

Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 15:18:15 GMT
View Forum Message <> Reply to Message

Just in case my ramblings have confused you a little, here is my final version of "display_horizontal"

```
<!-- display details horizontally -->
<xsl:template name="display_horizontal">
  <xsl:param name="zone"/>      <!-- could be 'main', 'inner', 'outer', etc -->
  <xsl:param name="table_record" />
  <xsl:param name="multiple"/>  <!-- set this for more than one occurrence -->

  <xsl:variable name="table" select="name()"/>          <!-- current table name -->
  <xsl:variable name="position" select="position()"/>   <!-- current row within table -->

  <tr>
    <!-- set the row class to 'odd' or 'even' to determine the colour -->
    <xsl:attribute name="class">
      <xsl:choose>
        <xsl:when test="position()mod 2">odd</xsl:when>
        <xsl:otherwise>even</xsl:otherwise>
      </xsl:choose>
    </xsl:attribute>

 <!-- step through the fields defined in the STRUCTURE element -->
    <xsl:for-each select="/root/structure/*[name()=$zone]/row/cell[@field]">

      <!-- get fieldname from the FIELD attribute -->
      <xsl:variable name="fieldname" select="@field" />

      <!-- select the field (identified in STRUCTURE) from the current row of the specified table -->
```

```
    <xsl:variable name="field" select="$table_record/*[name()=$fieldname]" />

    <td>
      <!-- process the named field from the current row -->
      <xsl:call-template name="datafield">
        <xsl:with-param name="item"     select="$field"/>
        <xsl:with-param name="itemname" select="$fieldname"/>
        <xsl:with-param name="path"     select="$table"/>
        <xsl:with-param name="multiple" select="$multiple"/>
        <xsl:with-param name="position" select="$position"/>
      </xsl:call-template>
    </td>

  </xsl:for-each>
 </tr>

</xsl:template> <!-- display_horizontal -->
```

and this is the change needed in the calling scripts to make it work...

std.link1.xsl

```
          <!-- process each non-empty row in the MAIN table of the XML file -->
          <xsl:for-each select="//*[name()=$main][count(*)&gt;0]">

            <!-- display all the fields in the current row -->
            <xsl:call-template name="display_horizontal">
               <xsl:with-param name="zone" select="'main'"/>
    <xsl:with-param name="table_record" select="." /> <!-- ADD THIS LINE TO ALL CALLING
SCRIPTS -->
            </xsl:call-template>
          </xsl:for_each>
```

similar change in std.link1.xsl

```
         <!-- process each non-empty row in the INNER/CHILD table of the XML file -->
          <xsl:for-each select="//*[name()=$link][count(*)&gt;0]">

            <!-- display all the fields in the current row -->
            <xsl:call-template name="display_horizontal">
              <xsl:with-param name="zone"     select="'link'"/>
   <xsl:with-param name="table_record" select="." />
              <xsl:with-param name="multiple" select="'y'"/>
            </xsl:call-template>

          </xsl:for-each>
```

etc etc...

Seems to work correctly. And it's lightning fast. Give it a whirl, what's the worst that could happen?

Subject: Re: xsl transformation slow
Posted by AJM on Mon, 19 Mar 2007 15:42:11 GMT
View Forum Message <> Reply to Message

I will put this on the top of my list as soon as I have completed my current task.

Subject: Re: xsl transformation slow
Posted by AJM on Fri, 23 Mar 2007 14:17:21 GMT
View Forum Message <> Reply to Message

I'm impressed. I've tried your suggestions on my old laptop (which runs PHP 4) and got the time for a 100 line list screen down from 20 seconds to under 5 seconds. On my relatively newer desktop (which runs PHP 5) the time dropped from 3.6 seconds to 0.8 seconds. Not too shabby, eh?

These improvements will be included in the next release. Thanks for the tip.

Subject: Re: xsl transformation slow
Posted by braingeyser on Fri, 23 Mar 2007 22:55:48 GMT
View Forum Message <> Reply to Message

Excellent results ! I'm relieved that it works on yours too. What do you think of the mods necessary for the fix? Manageable?

I'm happy to have contributed, in a small way, to Radicore's continued improvement. Keep up the good work.

Subject: Re: xsl transformation slow
Posted by AJM on Fri, 23 Mar 2007 23:50:52 GMT
View Forum Message <> Reply to Message

The mods were quite straightforward and easy to implement. Thanks for your contribution.