User Interface - Feature #5742

improve web client startup time

10/19/2021 12:29 PM - Greg Shah

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Tomasz Domin	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Runtime Infrastructure - Feature #5776: reduce memory requirements...	**Test**

History

#3 - 10/19/2021 12:41 PM - Greg Shah

We have recently seen two scenarios in which the web client has an extended startup time (multiple minutes).

- #5384 - on a slower AWS system (less CPU resources and less memory), the client startup is noticably slower
- #5656 - on a specific VM, the startup was in the 2 to 4 minute range but on the same VM host and resources a different VM had startup of less than a minute

The first case may be related to resources but the second case seems to be a low level issue in the software platform, rather than something in FWD.

However, in #5656 Sergey did find a list of items where most of the time was spent:

- transferring text metrics from the FWD server to the FWD client
- font initialization from the FWD client (there are slow calls to the FWD server and slow calls to the javascript code)

This task is intended to greatly reduce the overhead of these activities. The idea is that it will make FWD load much faster and the reduced burden will make it less likely to load slowly even in cases where there is slow hardware or some system level issue causing a slowdown.

My initial idea is that most (if not all) of the text metrics and font init can be done lazily instead of all up front. Let's discuss this idea and any others and then we will decide what approach makes sense.

#4 - 12/09/2021 12:22 PM - Greg Shah

- Assignee set to Constantin Asofiei

#6 - 05/03/2022 07:01 AM - Greg Shah

We have another case where a customer's system (based on Amazon Linux which is in turn based on RedHat/CentOS) showed this same behavior. See #6233. The same configuration on Ubuntu does not show this behavior. It seems this is related to something in RedHat/CentOS, probably code levels of some base library or subsystem.

The idea to reduce the amount of data transferred to the client is still a good one, which will greatly reduce the impact of this OS level bug.

#7 - 05/03/2022 07:02 AM - Greg Shah

- Assignee changed from Constantin Asofiei to Tomasz Domin

#8 - 05/03/2022 07:02 AM - Greg Shah

- Related to Feature #5776: reduce memory requirements for the FWD client added

#9 - 05/18/2022 06:04 AM - Tomasz Domin

- Status changed from New to WIP

#10 - 05/19/2022 12:42 PM - Tomasz Domin

Greg,
Should I continue working on this issue (and possibly looking for environmental issues) considering #6233#note-10 and #6233 being closed ?

#11 - 05/19/2022 01:24 PM - Greg Shah

Ignore the environmental issues.

What is left to do is the items in #5742-3 which reduce the startup time of the FWD client.

#12 - 05/25/2022 07:30 AM - Tomasz Domin

- % Done changed from 0 to 90

- File 5742a.diff added

- Status changed from WIP to Review

Greg Shah wrote:

> Ignore the environmental issues.
>
> What is left to do is the items in #5742-3 which reduce the startup time of the FWD client.

Dynamic text metrics cache has been implemented as described in #5776#note-29.
Results as in #5776#note-31:

Original application first startup time: 25s heap usage: 456MB
Original application second startup time: 17s heap usage: 456MB

Text metrics cached application first startup time: 16,8s heap usage: 62MB
Text metrics cached application second startup time: 7s-8s heap usage: 62MB

In order to activate the feature following configuration entry must be put into directory.xml (I put it after "custom-fonts"). If not present the old way will be used (pushing full text-metrics to client). Cache size is valid for both server and client side. It could have also value "full" so it will push full text-metrics to client as before.

```
        <node class="container" name="font-metrics">
          <node class="string" name="cache-size">
            <node-attribute name="value" value="1024"/>
           </node>
         </node>
       </node>
```

I've tested the change with hotel_gui embedded/virtual and customer app in virtual mode.

Attached please find a patch for 3821a (5742a.diff)
Please review.

#13 - 05/25/2022 10:42 AM - Greg Shah

Code Review 5742a.diff

1. I think the read access to sharedLegacyTextDimensionsCache uin FontTable.readLegacyTextMetrics() needs to be protected. As it is written, if I understand correctly, the static cache can be modified from other user sessions that call FontTable.getLegacyTextMetrics(). This means the underlying cache can be changing during the read operation which is not safe.

2. The intention here is to share the cache across all users?

3. There are code formatting issues. Several places have { that are not on their own line. Some cases of missing spacing around operators (int i=0; instead of int i = 0;) or in control structures (while(...) instead of while (...)).

#14 - 05/26/2022 07:17 AM - Tomasz Domin

Greg Shah wrote:

> To "one time" push some state from the client to the server at startup, we have void MainEntry.setClientParams(ClientParameters) which is called from ClientCore.start(). State can be added to the ClientParameters class.

I am not sure if **text-metrics** cache size parameter fits in there - I think these are 4GL parameters rather then runtime parameters.
I will try to push cache configuration with BootstrapConfig

Greg Shah wrote:

> Code Review 5742a.diff
>
> 1. I think the read access to sharedLegacyTextDimensionsCache uin FontTable.readLegacyTextMetrics() needs to be protected. As it is written, if I understand correctly, the static cache can be modified from other user sessions that call FontTable.getLegacyTextMetrics(). This means the underlying cache can be changing during the read operation which is not safe.

I wasn't sure if Directory entries can change at runtime. I presumed so until now, but it will be easier to assume it cant as most of code I've seen seems to share this approach. So cache configuration will only happen on startup, thus I will move initialization from FontTable.readLegacyTextMetrics to FontTable.initialize so only initiating session could change it.

> 2. The intention here is to share the cache across all users?

Yes. Global text-metrics is large and contains data which is never used (like metrics for font sizes never used in application).
The idea is to build **text-metrics** cache at runtime only from items clients are requesting - I guess real live client uses maybe 1-5% of **text-metrics** entries - in case of tested customer I had about 100 entries to push to client instead of 1600000.

> 3. There are code formatting issues. Several places have { that are not on their own line. Some cases of missing spacing around operators (int i=0; instead of int i = 0;) or in control structures (while(...) instead of while (...)).

Formatting corrected.
I will submit corrected patch once I'm done with transfer of configuration setting to client and some testing.

#15 - 05/26/2022 07:17 AM - Tomasz Domin

- Status changed from Review to WIP

#16 - 05/26/2022 07:48 AM - Greg Shah

> > To "one time" push some state from the client to the server at startup, we have void MainEntry.setClientParams(ClientParameters) which is called from ClientCore.start(). State can be added to the ClientParameters class.
>
> I am not sure if **text-metrics** cache size parameter fits in there - I think these are 4GL parameters rather then runtime parameters.
> I will try to push cache configuration with BootstrapConfig

I would rather set the cache size value in the directory.xml and if it needs to be sent down to the client, we can do that with the change I mentioned.

Is there a reason to configure this for every client? That seems messy and error prone. Also, the static nature of the cache on the server means that the size is server-wide, so it should not be determined by the client. Am I misunderstanding?

> > 1. I think the read access to sharedLegacyTextDimensionsCache uin FontTable.readLegacyTextMetrics() needs to be protected. As it is written, if I understand correctly, the static cache can be modified from other user sessions that call FontTable.getLegacyTextMetrics(). This means the underlying cache can be changing during the read operation which is not safe.
>
> I wasn't sure if Directory entries can change at runtime. I presumed so until now, but it will be easier to assume it cant as most of code I've seen seems to share this approach. So cache configuration will only happen on startup, thus I will move initialization from FontTable.readLegacyTextMetrics to FontTable.initialize so only initiating session could change it.

Directory entries can change during runtime but usually do not. And we don't have to honor those changes dynamically (most changes are not honored dynamically today). But my concern isn't with changing the size of the cache. I am worried about the contents of the cache changing during access from another session.

#17 - 05/26/2022 10:52 AM - Tomasz Domin

- % Done changed from 90 to 60

Let me explain the approach.
My goal is to balance between minimum client cache size and optimum performance not delayed by unnecessary network calls.

My idea is to replace full text metrics database on client side with three level cache:
1st level cache - client - **local text metrics cache** - initialized on startup with **optimized text metrics cache** retrieved from server
2nd level cache - server - **optimized text metrics cache** - managed by server based on client requests for text metrics items
3rd level cache - server = **full text metrics database** - a large Map as was implemented until now.

On start client retrieves optimized cache from server and updates it during runtime based on server queries or local metrics evaluation.
If **optimized text metrics cache** contains all needed text metrics items for typical client session -> no further calls to server are needed.
If any text metrics item is missing in client cache (1st level) it needs to be retrieved from server (from 2nd or 3rd level cache).
Based on client requests server can optimize **optimized text metrics cache** for later connecting clients

There are two parameters:

1. maximum cache size on client side - to be sent on client bootstrap to client, also will be known by server to limit number of items to be send to client
2. maximum cache size on server side - number of items to store on server side to choose optimum client initialization data from.

For LFUAgingCache 2. will be greater then 1., so server should send only the best items to client (for LFUAgingCache - the most frequently requested over time).

I started with a simpler setup (simple cache size on both sides, only LRUCache in use), but I think I need to go further and have introduced 2 cache sizes as above and LFUAgingCache on server side instead of LRUCache.

Greg Shah wrote:

> > I will try to push cache configuration with BootstrapConfig
>
> I would rather set the cache size value in the directory.xml and if it needs to be sent down to the client, we can do that with the change I mentioned.

It will be stored in directory.xml and it will be pushed to client as part of one-time BootstrapConfig like many other items in WebClient.

> Is there a reason to configure this for every client? That seems messy and error prone. Also, the static nature of the cache on the server means that the size is server-wide, so it should not be determined by the client. Am I misunderstanding?

There is one value of maximum client cache size for all clients, but this could be changed in runtime by directory.xml changes once a better value is found.

> Directory entries can change during runtime but usually do not. And we don't have to honor those changes dynamically (most changes are not honored dynamically today). But my concern isn't with changing the size of the cache. I am worried about the contents of the cache changing during access from another session.

If cache algorithm is chosen well and cache is big enough we would get optimized subset of all text metrics entries. Optimized based on actual client usage.
Its hard to estimate how sessions would affect each other until application usage patterns are known, but I hope following above approach may be a good start point.

#18 - 05/26/2022 11:05 AM - Greg Shah

I like the plan.

#19 - 05/27/2022 10:29 AM - Tomasz Domin

- % Done changed from 60 to 80

- File 5742b.diff added

- Status changed from WIP to Review

Implemented as planned
To activate following section needs to be added to directory.xml "server/default" section:

```
        <node class="container" name="font-metrics">
          <node class="integer" name="client-cache-size">
            <node-attribute name="value" value="1000"/>
          </node>
          <node class="integer" name="server-cache-size">
            <node-attribute name="value" value="100000"/>
          </node>
        </node>
```

server-cache-size - server side master cache LFUAgingCache maximum size. Note it also stores information about items missing in legacy text metrics (to send it to client to minimize communication)
client-cache-size - client side local cache LRUCache maximum size, initialized with the best subset of server cache

If section is not present the cache size is allocated for all text-metrics items similarly as before.

To **disable cache on server** size please set server-cache-size/value to -1

I've tested for application startup and memory (localhost only configuration):
old way - first startup time -> 30-32s
old way - subsequent startup time -> 19-22s, memory - 520MB

new way - first startup time -> 20-22s
new way - subsequent startup time -> 9-11s, memory - 30MB

The difference will be probably higher in case of distributed environment, as local optimized cache saves every client from transferring over 100mb of data for customer app.

I had to update LFUAgingCache to expose elements sorted by usage/age, wasnt sure if I should make an inner class.

Please review 5742b.diff patch for 3821c/13919

#20 - 05/27/2022 02:56 PM - Greg Shah

Code Review 5742b.diff

I think this is a really good update. I'm excited to see it in use in real applications. Please make the following modifications and then check it in to 3821c.

1. Let's set reasonable defaults so that this is active with a configuration that will work well for large desktop applications. If we have a small application (like Hotel GUI) where we can use a smaller set of values, then we can override it for those cases. The common case will be the big desktop GUI systems, where the defaults should be a good match.

2. Please handle these formatting issues:

- LFUAgingCache
  - Map<K,V> result=new HashMap<K,V>();
  - while(mfuElementsIterator.hasNext())
  - Entry<K, V> entry=mfuElementsIterator.next();
- WebClientBuilderOptions
  - cmd.add("client:text-metrics:cache-size="+options.get("clientTextMetricsCacheSize"));
- FontTable
  - sharedLegacyTextMetricsLock needs javadoc
  - if (textMetrics!=null)

#21 - 05/30/2022 10:35 AM - Tomasz Domin

- File 5742c.diff added

In meantime I've found an issue with the feature implementation.

There would be two types of setups of the feature:
- legacy/default - NOT having the feature configured - client-cache-size/server-cache-configured not present in directory.xml
- migrated - having the feature configured - having (client-cache-size/server-cache-configured) proper values present in directory.xml

The issue I've discovered is that in case when feature is not configured memory consumption goes significantly up.
To confirm my suspicion I've tested Client application and found following heap memory usage (after app startup):
- plain 3821c uses 457MB
- not properly configured 5742b - 518MB

So in case the feature is not configured a client would require additional ~60MB of memory (it probably would go down 50% after Garbage Collection).
The reason is that LRUCache requires more memory to store data then HashMap which makes difference in case of a real customer configuration when there are over 2000000 entries to cache and no server side optimization.

To fix this I've implemented a change which restores old way of storing Full Text Metrics data in case the feature is not configured. Full Text Metrics retrieved from server is not put into LRUCache - it is just stored and used as additional source of metrics.

Now text metrics data is retrieved in following order:

1. local LRUCache metrics cache
2. local Full Text Metrics retrieved from server at beginning, only if the feature is not configured
3. call to Server
4. Local metrics calculation

The result is stored back in LRUCache

Note that Full Text Metrics is not initialized and is empty in case the feature is configured.

LRUCache is still in use to store Local metrics calculation in case of Server misses.

After implementing the change I noticed following heap memory consumption:
- plain 3821c uses 457MB
- the feature not configured 5742b - 518MB
- the feature not configured 5742c - 459MB
- the feature properly configured 5742c - 31MB

So now the feature wont consume unnecessary memory in case its not configured in deployment.

Greg Shah wrote:

> Code Review 5742b.diff
> 1. Let's set reasonable defaults so that this is active with a configuration that will work well for large desktop applications. If we have a small application (like Hotel GUI) where we can use a smaller set of values, then we can override it for those cases. The common case will be the big desktop GUI systems, where the defaults should be a good match.

Actually I wanted to avoid the feature to be used without being configured first. In case there is no feature specific configuration application behaves the old way.
In case of large applications I'd let support configure parameters.
Anyway I've configured defaults for cache size - 100000 items for server and 10000 for client, which should be reasonable big.

> 2. Please handle these formatting issues:

Corrected, I've also configured additional formatting style checking tool, it needs tweaking but I hope it would help.

Attached please find 5742c.diff - the only functional change is different behavior of client in case client-cache-size is not configured.

#22 - 05/30/2022 10:42 AM - Greg Shah

You can merge 5742c.diff into 3821c.

#23 - 05/30/2022 10:43 AM - Greg Shah

Can you detect any performance difference in large applications (especially in open complex screens)?

#24 - 05/30/2022 12:04 PM - Tomasz Domin

- % Done changed from 80 to 100

Greg Shah wrote:

> Can you detect any performance difference in large applications (especially in open complex screens)?

No, I haven't noted performance difference as it was hard to compare - I was running apps on localhost only.
I've just noted less queries to server to get missing text-metrics.

Just a reminder - To activate following section needs to be added to directory.xml "server/default" section:

```
        <node class="container" name="font-metrics">
          <node class="integer" name="client-cache-size">
            <node-attribute name="value" value="10000"/>
          </node>
          <node class="integer" name="server-cache-size">
            <node-attribute name="value" value="100000"/>
          </node>
        </node>
```

server-cache-size - server side master cache LFUAgingCache maximum size. Note it also stores information about items missing in legacy text metrics.
client-cache-size - client side local cache LRUCache maximum size, initialized with the best subset of server cache

Merged into 3821c rev 13935.

#25 - 05/30/2022 12:32 PM - Greg Shah

- Status changed from Review to Test

#26 - 06/20/2022 08:35 AM - Roger Borrello

Tomasz, there is a minor javadoc issues to correct:

```
[ant:javadoc] /home/rfb/projects/fwd/3821c/src/com/goldencode/p2j/ui/client/FontManager.java:1505: warning - @
param argument "configured" is not a parameter name.
```

#27 - 06/20/2022 08:36 AM - Greg Shah

At least the TailScopedDictionary is not from Tomasz.

#28 - 06/20/2022 08:45 AM - Roger Borrello

Greg Shah wrote:

> At least the TailScopedDictionary is not from Tomasz.

I'm sorry.. I combined them incorrectly.

Files

5742a.diff	10.9 KB	05/25/2022	Tomasz Domin
5742b.diff	19.8 KB	05/27/2022	Tomasz Domin
5742c.diff	21.7 KB	05/30/2022	Tomasz Domin