Runtime Infrastructure - Bug #6084

NIO SSL performance is slower than standard Java SSL

02/21/2022 08:43 AM - Greg Shah

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Igor Skornyakov	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#2 - 02/21/2022 08:45 AM - Greg Shah

As found in #6075-15, there is at least one case where there is a signficant slowdown when using NIO SSL. We need to resolve this or at least understand the cause. If it cannot be resovled, then we should make the standard Java SSL our default.

#3 - 04/25/2022 12:26 PM - Greg Shah

- Assignee set to Igor Skornyakov

#5 - 05/02/2022 08:10 AM - Greg Shah

In #5384, Hynek implemented Conscrypt support for the web client communication (Java client SSL to web client over the web socket). This represents a substantial performance increase for the web client over the standard SSL. Interestingly, it is somehow even faster than unsecured sockets for the web client.

I want us to look at using Conscrypt for the FWD server to/from FWD client connection (both sides).

Hynek: What do you think is needed to make that happen?

#6 - 05/05/2022 10:52 AM - Hynek Cihlar

Greg Shah wrote:

> In #5384, Hynek implemented Conscrypt support for the web client communication (Java client SSL to web client over the web socket). This represents a substantial performance increase for the web client over the standard SSL. Interestingly, it is somehow even faster than unsecured sockets for the web client.
>
> I want us to look at using Conscrypt for the FWD server to/from FWD client connection (both sides).
>
> Hynek: What do you think is needed to make that happen?

All the jar dependencies are in FWD. The majority of the work will be to modify SecurityContext.getSecureSocketContext to return the correct SSLContext instance, Conscrypt in this case.

In theory the Conscrypt security provider should be installed with the call to Security.insertProviderAt and SSLCOntext retrieved with SSLContext.getInstance.

Possibly also SecurityManager.filterCiphers will need to be revisited.

#7 - 06/13/2022 11:32 AM - Igor Skornyakov

- Status changed from New to WIP

#8 - 06/13/2022 12:00 PM - Igor Skornyakov

As far as I understand Conscrypt code is not pure Java - it contains native parts. Is it acceptable?
Thank you.

#9 - 06/13/2022 12:04 PM - Greg Shah

Yes, it is OK.

#10 - 06/14/2022 03:08 AM - Igor Skornyakov

I've an impression that Conscrypt is mostly used on the Android platform.
Moreover, it is based on BoringSSL. Here is an excerpt from the BoringSSL documentation (https://github.com/google/boringssl):

```
BoringSSL is a fork of OpenSSL that is designed to meet Google's needs.

Although BoringSSL is an open source project, it is not intended for general use, as OpenSSL is. We don't reco
mmend that third parties depend upon it. Doing so is likely to be frustrating because there are no guarantees
of API or ABI stability.

Programs ship their own copies of BoringSSL when they use it and we update everything as needed when deciding
to make API changes. This allows us to mostly avoid compromises in the name of compatibility. It works for us,
 but it may not work for you.

BoringSSL arose because Google used OpenSSL for many years in various ways and, over time, built up a large nu
mber of patches that were maintained while tracking upstream OpenSSL. As Google's product portfolio became mor
e complex, more copies of OpenSSL sprung up and the effort involved in maintaining all these patches in multip
le places was growing steadily.

Currently BoringSSL is the SSL library in Chrome/Chromium, Android (but it's not part of the NDK) and a number
 of other apps/programs.
```

I'm not sure that relying on the Conscrypt/BoringSSL is a good idea.
However, I continue working on adding support for it.

#11 - 06/14/2022 08:25 AM - Greg Shah

I appreciate the concern. Thank you for noting this, it is important.

On the other hand, I want us to have a choice of transport. This will allow us to measure the advantage of different stacks. Please ensure it is

configurable.

#12 - 06/14/2022 09:13 AM - Igor Skornyakov

Greg Shah wrote:

> I appreciate the concern. Thank you for noting this, it is important.
>
> On the other hand, I want us to have a choice of transport. This will allow us to measure the advantage of different stacks. Please ensure it is configurable.

I see. Thank you.

#13 - 06/24/2022 07:05 AM - Igor Skornyakov

- Status changed from WIP to Review

- File 6084.diff added

Added support for the Conscrypt SSL provider.
I've also changed the way for configuring non-default SSL providers.
Now one should use security:provider:name bootstrap configuration parameter. Supported values are **"bouncycastle"** and **"conscrypt"**. The previous way to configure using BouncyCastle (security:bouncycastle:use) is obsolete now (silently ignored).
Please review.
Thank you.

#14 - 06/24/2022 08:40 AM - Hynek Cihlar

Code review 6084.diff.

In ServerDriver.java:

```
      if (provider == null)
      {
      }
```

null should be handled similarly as the invalid value case. So that the condition is not silently ignored and the issue can be spotted right away.

In multiple locations: Conscript -> Conscrypt.

```
this.maxMessageSize = appBufferMax - 50 + delta;@
```

When delta is 0 maxMessageSize will become smaller than the value returned by getApplicationBufferSize. Is this expected?

```
      int delta = engine.getClass().getName().startsWith("org.conscrypt") ? 0 : 50;
```

Delta value of 50 is needed for a particular set of providers so the logic should be inverted. Instead of checking for Conscrypt, you should check for the providers where delta 50 is needed. This will help when we add another provider to the mix.

#15 - 06/24/2022 09:30 AM - Igor Skornyakov

Hynek Cihlar wrote:

> Code review 6084.diff.
>
> In ServerDriver.java:
>
> [...]
>
> null should be handled similarly as the invalid value case. So that the condition is not silently ignored and the issue can be spotted right away.

I cannot agree with you. provider == null means that no provider override was specified and the default one from JDK should be used. I think that this is actually the most common situation.

> In multiple locations: Conscript -> Conscrypt.

Fixed.

> [...]
> When delta is 0 maxMessageSize will become smaller than the value returned by getApplicationBufferSize. Is this expected?

Yes, this is expected.

> [...]
> Delta value of 50 is needed for a particular set of providers so the logic should be inverted. Instead of checking for Conscrypt, you should check for the providers where delta 50 is needed. This will help when we add another provider to the mix.

At this moment we support only 3 providers - a default JDK one, BouncyCastle, and Conscrypt. It is difficult to say what value of delta should be considered "default". At this moment the most common value is 50. The logic is very sensitive to the buffers' sizes so I suggest leaving it as-is for now.

#16 - 06/24/2022 09:44 AM - Hynek Cihlar

Igor Skornyakov wrote:

> Hynek Cihlar wrote:
>
> > Code review 6084.diff.
> >
> > In ServerDriver.java:
> >
> > [...]
> >
> > null should be handled similarly as the invalid value case. So that the condition is not silently ignored and the issue can be spotted right away.
>
> I cannot agree with you. provider == null means that no provider override was specified and the default one from JDK should be used. I think that this is actually the most common situation.

Actually I wasn't suggesting to handle it in any particular way but rather to handle the case at all (an exception, log output, etc.).

> > When delta is 0 maxMessageSize will become smaller than the value returned by getApplicationBufferSize. Is this expected?
>
> Yes, this is expected.

But doesn't it break the contract? The following is from the javadoc getApplicationBufferSize:

```
     * Gets the current size of the largest application data that is
     * expected when using this session.
     * <P>
     * SSLEngine application data buffers must be large
     * enough to hold the application data from any inbound network
     * application data packet received.  Typically, outbound
     * application data buffers can be of any size.
```

So when we decrease the buffer size by 50 or whatever we won't satisfy the above. This is counter-intuitive to me.

> > [...]
> > Delta value of 50 is needed for a particular set of providers so the logic should be inverted. Instead of checking for Conscrypt, you should check for the providers where delta 50 is needed. This will help when we add another provider to the mix.
>
> At this moment we support only 3 providers - a default JDK one, BouncyCastle, and Conscrypt.
> It is difficult to say what value of delta should be considered "default". At this moment the most common value is 50. The logic is very sensitive to the buffers' sizes so I suggest leaving it as-is for now.

Where does the value come from anyway? I think it will be useful to add code comments describing its origins.

#17 - 06/24/2022 09:58 AM - Igor Skornyakov

Hynek Cihlar wrote:

> null should be handled similarly as the invalid value case. So that the condition is not silently ignored and the issue can be spotted right away.

I cannot agree with you. provider == null means that no provider override was specified and the default one from JDK should be used. I think that this is actually the most common situation.

Actually I wasn't suggesting to handle it in any particular way but rather to handle the case at all (an exception, log output, etc.).

I decided that there is no point in logging the situation when we work in a default mode. Please note also that bootstrap configuration parameters are already logged.

> When delta is 0 maxMessageSize will become smaller than the value returned by getApplicationBufferSize. Is this expected?

Yes, this is expected.

But doesn't it break the contract? The following is from the javadoc getApplicationBufferSize:

[...]

So when we decrease the buffer size by 50 or whatever we won't satisfy the above. This is counter-intuitive to me.

I agree. However, this trick (and the value of delta) was added as a result of multiple experiments.

> [...]
> Delta value of 50 is needed for a particular set of providers so the logic should be inverted. Instead of checking for Conscrypt, you should check for the providers where delta 50 is needed. This will help when we add another provider to the mix.

At this moment we support only 3 providers - a default JDK one, BouncyCastle, and Conscrypt.
It is difficult to say what value of delta should be considered "default". At this moment the most common value is 50. The logic is very sensitive to the buffers' sizes so I suggest leaving it as-is for now.

Where does the value come from anyway? I think it will be useful to add code comments describing its origins.

Added the comment.

#18 - 06/24/2022 10:52 AM - Hynek Cihlar

Igor Skornyakov wrote:

> Hynek Cihlar wrote:
>
> > null should be handled similarly as the invalid value case. So that the condition is not silently ignored and the issue can be spotted right away.
>
> I cannot agree with you. provider == null means that no provider override was specified and the default one from JDK should be used. I think that this is actually the most common situation.

Actually I wasn't suggesting to handle it in any particular way but rather to handle the case at all (an exception, log output, etc.).

I decided that there is no point in logging the situation when we work in a default mode. Please note also that bootstrap configuration parameters are already logged.

I see. In that case please remove the empty branch as it adds noise to the code and do else if (provider != null) for the illegal state branch.

> So when we decrease the buffer size by 50 or whatever we won't satisfy the above. This is counter-intuitive to me.

I agree. However, this trick (and the value of delta) was added as a result of multiple experiments.

The think is that your change modifies the maxMessageSize for the non-conscrypt cases. So this invalidates the multiple experiments done for the the non-conscrypt cases, doesn't it?

#19 - 06/24/2022 10:58 AM - Igor Skornyakov

Hynek Cihlar wrote:

> I see. In that case please remove the empty branch as it adds noise to the code and do else if (provider != null) for the illegal state branch.

Hynek. Please take a look at this place again. The provider null is a perfectly valid situation. See a similar please in the SecurityManager.getSecureSocketContext where this branch is not empty.

> So when we decrease the buffer size by 50 or whatever we won't satisfy the above. This is counter-intuitive to me.

> I agree. However, this trick (and the value of delta) was added as a result of multiple experiments.

> The think is that your change modifies the maxMessageSize for the non-conscrypt cases. So this invalidates the multiple experiments done for the the non-conscrypt cases, doesn't it?

No, for non-conscrypt case maxMessageSize appBufferMax as it was before adding Conscrypt support.

#20 - 06/24/2022 11:04 AM - Hynek Cihlar

Igor Skornyakov wrote:

> Hynek Cihlar wrote:
>
> > I see. In that case please remove the empty branch as it adds noise to the code and do else if (provider != null) for the illegal state branch.
>
> Hynek. Please take a look at this place again. The provider == null is a perfectly valid situation. See a similar please in the ServerDriver.start where this branch is not empty.

If you remove the empty branch and add provider != null to the last branch, then null will naturally stay a valid situation. The advantage of this is that the noise will be gone.

> So when we decrease the buffer size by 50 or whatever we won't satisfy the above. This is counter-intuitive to me.

I agree. However, this trick (and the value of delta) was added as a result of multiple experiments.

> The think is that your change modifies the maxMessageSize for the non-conscrypt cases. So this invalidates the multiple experiments done for the the non-conscrypt cases, doesn't it?

No, for non-conscrypt case maxMessageSize == appBufferMax as it was before adding Conscrypt support.

True, it is OK then.

#21 - 06/24/2022 11:09 AM - Igor Skornyakov

- File 6084-1.diff added

Hynek Cihlar wrote:

> If you remove the empty branch and add provider != null to the last branch, then null will naturally stay a valid situation. The advantage of this is that the noise will be gone.

Oh, I see. Fixed.
Please find the final version attached.
Thank you.

#22 - 06/24/2022 12:11 PM - Hynek Cihlar

Code review 6084-1.diff. The changes are good.

#23 - 06/24/2022 12:19 PM - Igor Skornyakov

- Status changed from Review to Test

Hynek Cihlar wrote:

> Code review 6084-1.diff. The changes are good.

Thank you.
Committed to 3821c/14001.

#24 - 06/24/2022 02:33 PM - Greg Shah

Code Review Task Branch 3821c Revision 14001

1. BCProbe and BouncyCastle should not be referenced in SecurityManager$CSProbe.

2. Please move the code that is hard coded to specific SSL providers into a separate class com.goldencode.p2j.security.SecureSocketsRegistrar instead of leaving it in the SecurityManager. I would want to see that the BouncyCastle and Conscrypt imports can be removed from SecurityManager. I'd expect BCHolder, CSHolder, BCProbe and CSProbe to be outside of the SecurityManager as well. The provider registration can be done using SSLContext register(BootstrapConfig, boolean) where the 2nd boolean parameter causes a SSLContext.getInstance() and a return of that instance if true (and null if false). The registration method should be common code (please call it from ServerDriver instead of having a separate implementation there). This change is intended to better encapsulate the provider logic and to remove code duplication.

3. Instead of raising IllegalStateException when an invalid provider is configured, please just log a WARNING. I don't see why we need to make this a fatal error since it is recoverable (we should just use the default provider if things are misconfigured).

#25 - 06/24/2022 02:34 PM - Greg Shah

Please also implement any JMX instrumentation needed to help us measure the performance of the SSL processing.

#26 - 06/24/2022 02:37 PM - Igor Skornyakov

Greg Shah wrote:

> Code Review Task Branch 3821c Revision 14001
>
> 1. BCProbe and BouncyCastle should not be referenced in SecurityManager$CSProbe.
>
> 2. Please move the code that is hard coded to specific SSL providers into a separate class com.goldencode.p2j.security.SecureSocketsRegistrar instead of leaving it in the SecurityManager. I would want to see that the BouncyCastle and Conscrypt imports can be removed from SecurityManager. I'd expect BCHolder, CSHolder, BCProbe and CSProbe to be outside of the SecurityManager as well. The provider registration can be done using SSLContext register(BootstrapConfig, boolean) where the 2nd boolean parameter causes a SSLContext.getInstance() and a return of that instance if true (and null if false). The registration method should be common code (please call it from ServerDriver instead of having a separate implementation there). This change is intended to better encapsulate the provider logic and to remove code duplication.
>
> 3. Instead of raising IllegalStateException when an invalid provider is configured, please just log a WARNING. I don't see why we need to make this a fatal error since it is recoverable (we should just use the default provider if things are misconfigured).

OK. Will be done tomorrow.

#27 - 06/24/2022 02:37 PM - Igor Skornyakov

Greg Shah wrote:

> Please also implement any JMX instrumentation needed to help us measure the performance of the SSL processing.

Sure.

#28 - 06/26/2022 02:05 PM - Igor Skornyakov

Re-worked Conscrypt support based on the code review. Added JMC counters for send, wrap, and unwrap operations of the SSL class.
Committed to 3821c/14002.

Here are the values of the counters after a quick test of the large customer app with different JCE/JSSE providers.

Counter	default	Conscrypt	BouncyCastle
objToByteArray (count/ms):	13155/331	13175/304	13117/326
sslSend (count/ms):	15769/2425	15818/1981	15759/1517
sslWrap (count/ms):	18172/2405	15822/1968	15763/1500
sslUnwrap (count/ms):	13177/164	13198/160	13144/458
NetworkReads (count/total):	13155/2486590	13175/2484614	13117/2479905
NetworkWrites (count/total):	13155/49793798	13175/49397256	13117/49407508
WidgetAttrFlushes (count/total):	4032/4032	4060/4060	4032/4032

#29 - 06/27/2022 02:34 AM - Constantin Asofiei

Igor, in Eclipse I get a 'package not found' for this, in SecureSocketsRegistrar:

```
import org.apache.jasper.tagplugins.jstl.core.*;
```

#30 - 06/27/2022 03:10 AM - Igor Skornyakov

Constantin Asofiei wrote:

> Igor, in Eclipse I get a 'package not found' for this, in SecureSocketsRegistrar:
> [...]

Sorry, It was added by Eclipse. Strangely it does not cause problems in my environment.
Fixed in 14007.

#31 - 07/14/2022 11:44 AM - Igor Skornyakov

Added total time in ns to the NanoCounter.toStrng.
Committed to 3821c/ 14067.

Files

6084.diff	9.88 KB	06/24/2022	Igor Skornyakov
6084-1.diff	9.9 KB	06/24/2022	Igor Skornyakov