

Re: Help – Timing Logic

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vb/2006-12/msg00092.html>

- *From:* "Stephany Young" <noone@localhost>
 - *Date:* Thu, 30 Nov 2006 01:32:20 +1300
-

Ok. Just in case you're interested, here's how we did something that sounds very very similar to what you are trying to achieve.

Our requirement was to have a central message lodgement resource that had to be able to handle up to 10,000 messages during a 9 hour working day. A given message would be for 1 and only one user but we had to cater for up to 200 users.

Messages were automatically 'lodged' by an application that could be run by any user in the system.

Any given user may not be logged in at any given time, but any given user could be logged into multiple computers simultaneously. This meant that to deliver a message to a given user we had to deliver it to all machines they were logged into. If a user was not logged, we, of course, could not deliver a message to them but it had to be delivered to them at the first available opportunity.

The way we approached this was to have a client application that ran in the users start menu. The application, when it started, 'registered' the user to the message lodgement resource along with the machine name. When the application terminated, (i.e. when the user logged off), the application 'de-registered' the user/machine combination.

The user also lodged messages for other users via this application (including users who were not online).

The application also set up a TCP Listener on a predefined port for receiving messages.

The message lodgement resource, comprised a SQL Server 2005 database and a server application, both of which ran on the same box.

One part of the server application was dedicated to receiving messages from the client applications and 'lodging' them in the database.

Another part of the server application was dedicated to retrieving messages from the database and distributing them to the clients.

Re: Help – Timing Logic

A third part of the server application was dedicated to displaying throughput and status information.

The database comprised 3 main tables. One table held static user information, a second table held dynamic user/machine information and the third table was the message store.

The structure of the message store table was, essentially:

```
create table messagestore(  
ms_id bigint not null identity(1,1),  
ms_timestamp datetime not null,  
ms_sender nvarchar(50) not null,  
ms_recipient nvarchar(50) not null,  
ms_message nvarchar(max) not null,  
ms_status int not null constraint df_messagestore_01 default (0),  
constraint pk_messagestore primary key (ms_timestamp,ms_id)  
)
```

The ms_id column was used purely to keep messages in the sequence that they were lodged if the value for the ms_timestamp column was not unique.

When a message was lodged, a value for ms_status was NOT specified in the insert statement therefore it started 'life' with a status of 0 which meant that it was available for sending.

The retrieval of messages was done via a stored procedure as follows:

```
create procedure retrievemessage as  
  
set nocount on  
  
declare @id bigint  
  
begin transaction  
  
select top 1 @id=ms_id from messagestore where status=0 order by  
ms_timestamp,ms_id  
  
update messagestore set ms_status=1 where ms_id=@id  
  
select  
ms_id,,  
ms_timestamp,  
ms_sender,  
ms_recipient,  
ms_message  
from  
messagestore  
where
```

Re: Help – Timing Logic

ms_id=@id

commit transaction

go

The server transaction had a thread with a loop that continually executed that stored procedure via a datareader. If there was no row returned then the thread slept for 50 milliseconds before continuing the loop. If a row was returned, the thread dealt with the row and continued the loop immediately. If there were 10 iterations of the loop without there being a 50 ms sleep then a 50 ms sleep was forced. This allowed the thread to 'do its stuff' on a timely basis without becoming unresponsive to any control 'commands'.

The prime intent was to send all available messages in the message store as quickly as possible after they were lodged.

The thread spawned very short-lived worker threads to do the actual sending passing the necessary values as parameters.

A single instance of a worker thread handled one and only one message.

The worker thread determined if the user was logged in. If so it sent a copy of the message to every machine that the user was logged in to. If it successfully sent the message it updated the ms_status value in the appropriate database row from 1 to 2. If it was unable to send the message for any reason it updated the ms_status value in the appropriate database row from 1 to 3.

When we stress tested this we had various users logged into a total of 20 machines and we preloaded the message store with 50,000 messages, (5 days worth), for those users but about 5,000 messages were for users who were not logged in. Two of the users were logged into 5 machines each. We then started the retrieval thread.

The time it took that thread to deal with all 50,000 messages was just over 24 seconds. Of all the worker thread instances, the longest lived thread was just over 300 milliseconds, and, as expected, that was one of the threads handling a message for one of the users that was logged in to 5 machines. The average length of the ms_message value was about 200 Kilobytes.

Certainly the hardware we were dealing with was reasonably beastly but we were more than happy with the performance of the techniques we used, (we had not expected it to be quite that good).

So the moral of the story is: Don't get all bitter and twisted about worst case performance expectations. Give it a suck and see and you may find that your fears about performance issues might be groundless.

Good luck!!!!

Re: Help – Timing Logic

Re: Help – Timing Logic

I
have
a
multi
threaded
VB.NET
application
(4
threads)
that
I
use
to
send
text
messages
to
many,
many
employees
via
system.timer
at
a
5
second
interval.
Basically,
I
look
in
a
SQL
table
(queue)
to
determine
who
needs
to
receive
the
text
message
then
send
the
message

Re: Help – Timing Logic

to
the
address.
Only
problem
is,
the
employee
may
receive
up
to
4
of
the
same
messages
because
each
thread
gets
the
recors
then
sends
the
message.
I
need
somehow
to
prevent
this...
just
can't
think
of
how.
Somehow
I
need
the
other
threads
to
know
that
another
thread
is
already

Re: Help – Timing Logic

using
that
record
and
move
on
to
the
next
record.
I
thought
of
getting
the
record
then
marking
it
(column
value)
as
in
use
and
writing
the
code
to
look
for
records
only
where
not
in
use...
problem
is
all
4
run
fast
enough
to
all
use/mark
the
row.
Any
thoughts?

Re: Help – Timing Logic

"Jay" <someone@xxxxxxxxxxxxxx> wrote in message
news:OGJfd9xEHHA.3188@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Thanks Jeff, answers under your questions...

"jeff" <jhersey at allnorth dottt com> wrote in message
news:uekJE1xEHHA.2356@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

I know it is a time critical system ... employees need the information ASAP... but what I do not understand is ...

– where is the application running ... on a server ... on a users work station?

Server (2003)

– who will be responsible for executing / sending messages ... one computer or many different computers?

One computer. One multithreaded vb.net app.

– will this message sender reside on a server / database server / or a user's workstation ... ???

Remote SQL Server 2005 (EE)

– what database are you using ???

SQL Server 2005 EE

– is this functionality wrapped up in a large application?

Not very large. The logic around this loop cycle is pretty much the entire application.

– why 4 threads ?

Re: Help – Timing Logic

Allows sending 4 messages at a time.

– how does you program determine which message to send ... parameter list please.

Read from a table in the db server. A Queue table. (employeeid, messageaddress)

If this information is so mission critical, and employees must get their message ASAP ... put a trigger on the database table.

Cant have a trigger execute a function in a remote vb.net app.

... wrap the necessary functionality in a small stand alone exe application

Exactly what I need to do.

...
... build your application so it receives commandline parameters / string

Considering it.

...
... build a trigger on the database .. fire on inserts ... have it call your message program with a command parameter (messageID) ...
... install your application on the database server this will speed up the connection / retrievals and so on ... no network latency.

Each time a message is inserted in the table, the trigger fires, calling your EXE with the appropriate parameter string, exe starts, fires off message ... done. Small exe can run as many times on the server...

Re: Help – Timing Logic

If you are using a timestamp for determining which messages to send ...
incorporate another table ... LastTimeExecuted ...

table: MessengerExecution
field: LastDateTimeFired

begin transactions ... lock table.
select lastdatetimedfired from this table...in a variable
update field with thread date/time ...
end transaction.
return the select value and the update value

select messages where date is between lastdatetimedfired and the
updateddatetime I just used...

Again, there are many solutions ... just do not completely understand
what you are doing...

Bottom line ...

– you have determined that you need to run a mutli-threaded process for
this.

– so, in order to avoid DUPLICATE MESSAGES you either have to
employee...

Transaction and Database Locking – look at isolation levels / settings

or

A booker type of system...

have the broker continually pool the table ... this will only work if 1
machine is designated for messaging! If more than one machine will be
used for messaging ... you will have to roll up your sleeves and look at
Transactions and Isolation Levels.

CheckMessage

```
Do Until Company.Revenues < 0  
MessegeID = broker.getNextMessage()  
broker.sendMessage(MessageID)  
Loop
```

SendMessage(MessageID)

Create a new thread...

Re: Help – Timing Logic

SendMessage(MessageID) in this new thread...
Return to the MessageBroker.CheckMessage...while the other thread is
preparing and sending the message...

This will continuous poll the database server for new messages! ...

However, if it is mission critical users receive this information ASAP
... TRIGGER on database table! If the user can wait 10 seconds .. build
a BROKER ... and have it spawn as many threads needed to send the
messages in the QUEUE ... have it control the process...and sending!

Again, many solutions, depends on your needs ... code sample, table
structure ... some thing to trying and figure out how exactly your
'threads' are getting 1 message...

Jeff.

"Jay" <someone@xxxxxxxxxxxxxx> wrote in message
news:OLc3ODxEHHA.996@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

It is extremely important for the employees to receive the
messages
almost immediately after identifying (time val in table). So,
threading
ensures I can execute multiple processes at the same time...
only
problem is these threads read from the same queue table and
have the
potential to send the same message to the same guy x number
of times.
where x is the number of threads running.

"jeff" <jhersey at allnorth dottt com> wrote in message
news:uQ7%23Y4wEHHA.4404@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Still do not know why you need 4 threads ?
Speed ? Does runnig the
process in one thread not work fast enough
for you since timing is
import ? Why 4 threads why not 6,12 , 3, 36
... ? How do you
determine which row a 'thread retrieves' ???
does it simply select the
next available row from the DATABASE
??? How does the thread determine

Re: Help – Timing Logic

'the next row'???

If mutli-threading is duplicating 'send messages' it is not working properly...

If you are multi-threading to improve performance ... maybe look at the design...

If you need to implement a locking mechanism / or / logging mechanism / or / a checking mechanism to avoid duplicate messages caused by multi-threading ... these will all come at a cost ... performance cost!

What is bottle necking your process that you need 4 threads? Is it the READ from the database ... is it the SENDING the text message? Is it connecting to the database ??? do not know here ...

Look at what is causing problem ...

– you need better performance ... so, multiple thread it! However, mutli-threading the entire process is causing issues ... duplicate records ... now you either need to incorporate a locking procedure ... or a checking procedure to avoid duplicate messages... all this has a cost to the overall performance ... adding more threads may in fact negatively impact the overall performance....

Maybe implement a message broker ...

– message broker gets all the necessary messages or message id's from the database to be sent ... in-memory list...
– message broker loops through list of messageIDs ...
– message broker starts another thread for processing the sendtextmessage functionality for each messageID...
– message broker will include the messageID so the process knows which message to get and process...
– message broker will ensure each database

Re: Help – Timing Logic

row is only processed
once...

It is very hard to help you without knowing
exactly what you are
doing...

Again, I will ask, what rational / reasoning
did you use for using 4
threads ? Performance ? Speed ? ... Where is
the bottle neck in the
process that requires you to multi-thread it?
Maybe just move the
'bottle neck section' to another thread ... ?

Grasping at straws ...

Code would be nice how to see what you are
doing ...

Jeff

"Jay" <someone@xxxxxxxxxxxx> wrote in
message
news:uBX7QewEHHA.1224@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Thanks. I do need to select
an individual row at a time
and all 4
threads need to do this.
What trasaction isolation
level would you
recommend? Perhaps a
stored proc may be faster to
execute and return
the values as opposed to
building the transaction in

Re: Help – Timing Logic

the code. What has to happen is every 5 seconds, and for each thread, a sub runs to get a single row then send the message to `dtr("numbertosendto")`. Because this app heavily relies on timing it is important that all threads run and only one distinct row can be returned at a time for each thread.

"jeff" <jhersey at allnorth dottt com> wrote in message
news:%23qVFIYwEHHA.3576@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

are you
selecting
individual
rows at a
time from
the database
table

...

if so ... use
a
transaction

...

begin
transaction

...

select a row
from
database ...

update the
row in table
set flag =
'Processed'

end

Re: Help – Timing Logic

transaction

this will
lock the row
until the
end of
transaction
is issued ...
as
long as the
isolation
level is set
accordingly...

Again,
please let us
know how
you are
getting the
information
from
the
database...
then we can
help!

If you are
reading a
bunch of
rows in one
statement,
storing the
rows
in
in-memory
datasets on
the
workstation,
looping
through the
rows
one by one
... then you
may need to
either ...

implement
as above
locking
only the
records you

Re: Help – Timing Logic

retrieve /
update –
need to
watch out
here for
table
locking ...
may impact
performance,
implement
using an
update flag
at sent and
'check
before send
method'...
...
lots of
options
here, just
need to
know how
you are
retrieving
your
data...and
how you are
processing
the data.

Question,
why do you
need 4
threads
running ...
are they
doing
'different
processes',
sending
different
'types' of
messages ...
sending the
same
message ...
just need 4
to make it
faster ...
what
is the logic

Re: Help – Timing Logic

...

Tying a
record to a
thread my
cause
problems in
the future ...
what
happens
when a
thread stops
or hangs ...
those
messages
will not be
processed...
When
happens if
somebody
changes the
ThreadID ...
in the
program ...

trying to
help ..

Jeff

PS: you can
lock
individual
rows ... look
at how
database
transactions
work and
incorporate
it in you
program ...

"Jay"
<someone@xxxxxxxxxxxxxx>

Re: Help – Timing Logic

wrote in
message
news:%23f70LJwEHHA.2328@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Thanks
for
the
reply.

I
do
need
all
4
threads
running
(maybe
even
more
in
the
future).

I
can
not
delete
the
row
from
the
table...
it
needs
to
be
there
for
later
update
use.

I
am
considering
marking
each
row
at
insert
(1
–
4)

Re: Help – Timing Logic

then
having
each
of
the
4
threads
only
select
a
row
based
on
the
mark.
It's
becoming
a
little
tricky...
it
would
be
great
if
I
could
lock
a
single
row
when
selecting
it.

"jeff"
<jhersey
at
allnorth
dottt
com>
wrote
in
message
news:OPCyvDwEHHA.4740@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Question:

What

Re: Help – Timing Logic

is
the
reason
for
the
4
threads?

Do
you
need
all
these
threads
processing
the
same
dataset?

Just
asking,
maybe
you
could
change
your
approach
to
avoid
this
problem...

If
not,
investigate
Record
Locking
/
Transactions...

How
are
you
retrieving
your
records
in
to
memory
for
processing

Re: Help – Timing Logic

...

ie,
–
are
you
selecting
on
the
entire
contents
of
a
'message'
table
and
looping
through
an
in-memory
dataset...

–

do
you
retrieve
'chunks'
or
rows
from
the
message
table
based
on
parameters...

You
could
use
the
internal
locking
mechanisms
of
the
database
to
achieve
this

...

Re: Help – Timing Logic

but
procede
with
caution...

Start
Transaction...

SELECT
*
FROM
MESSAGE
TABLE

...
stuff
this
into
a
data
table
on
the
desktop

...
DELETE
*

FROM
MESSAGE
TABLE

...
punt
/
clean
the
table.

...
End
Transaction.

The
transactions
places
locks
on
the
table

...
but
selecting
the
entire
tables

Re: Help – Timing Logic

contents

...

you

are

essentially

'LOCKING'

the

entire

table

until

the

transaction

is

over...

So,

each

thread's

'retrieve'

process

will

have

to

wait

in

line

until

the

select

and

delete

are

processed.

As

well,

anything

triggering

'New

Messages

to

be

Added'

will

be

delayed

until

the

Transaction

is

completed...

Re: Help – Timing Logic

You
may
impact
the
overall
performance
of
you
application
...
need
to
investigate.

If
locking
does
not
work
for
you,
you
will
need
to
implement
some
type
of
logging
/
checking
approach...

ie...
have
a
log
table...once
a
process
has
sent
the
message,
write
to
a
log
table
...

Re: Help – Timing Logic

message
sent.
Before
each
message
is
compiled
and
sent,
check
the
log
table
to
see
if
another
process
has
sent
the
message...if
not,
send
you
message...

A
snip
of
code
would
be
great
to
figure
out
how
you
are
retrieving
a
list
of
messages
and
how
you
are
processing

Re: Help – Timing Logic

each
record...

Jeff.

"Jay"

<someone@xxxxxxxxxxxxxx>

wrote

in

message

news:O2phzwvEHHA.4280@xxxxxxxxxxxxxxxxxxxx

I
have
a
multi
threaded
VB.NET
application
(4
threads)
that
I
use
to
send
text
messages
to
many,
many
employees
via
system.timer
at
a
5
second
interval.
Basically,
I
look
in
a
SQL
table
(queue)
to
determine
who

Re: Help – Timing Logic

needs
to
receive
the
text
message
then
send
the
message
to
the
address.
Only
problem
is,
the
employee
may
receive
up
to
4
of
the
same
messages
because
each
thread
gets
the
recors
then
sends
the
message.
I
need
somehow
to
prevent
this...
just
can't
think
of
how.
Somehow
I
need

Re: Help – Timing Logic

the
other
threads
to
know
that
another
thread
is
already
using
that
record
and
move
on
to
the
next
record.
I
thought
of
getting
the
record
then
marking
it
(column
value)
as
in
use
and
writing
the
code
to
look
for
records
only
where
not
in
use...
problem
is
all
4

Re: Help – Timing Logic

run
fast
enough
to
all
use/mark
the
row.
Any
thoughts?

Thanks
a
lot.

Jay