

creating jobs programmatically causes deadlocks

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.server/2004-10/0178.html>

From: dwaine (*dwaine_at_nospam.nospam*)

Date: 10/01/04

Date: Fri, 1 Oct 2004 09:15:28 -0700

I'm using the transactional sproc below to create jobs in an automated fashion that will run cmdexecs after a configurable pause. This is potentially HIGH VOLUME! The sproc is run from a VB.NET EXE using the ADO.NET command.executenonquery method during the processing of text files to determine what the job should do.

I have two essentially identical servers (4 X 3.2GHz CPU, 3GB RAM). On one server, dropping 30 trigger files results in ~150 jobs being created in ~30 seconds (I have a thread.sleep(200) between each job creation due to another issue). On another server dropping just 10 of the trigger files causes deadlocks consistently. Both servers are running SQL 2000 SP3 on WinServer 2003 Standard edition and the assembly is written against DNF 1.1.

Is the logic in the sproc faulty? The transaction is necessary since the job needs all the properties to do it's job.

What server configurations affect deadlock timeouts, etc. Any help would be appreciated.

Sproc follows:

--creates single use, self deleting job

```
CREATE PROCEDURE usp_RunCmdJobSoon (  
    @JobName as varchar(50),  
    @CmdText as varchar(1000),  
    @DelaySeconds as int = 30,  
    @RunOnIdle as bit = 0,  
    @DeleteWhenDone as bit = 0  
)
```

```
AS  
BEGIN TRANSACTION  
DECLARE @JobID BINARY(16)  
DECLARE @ReturnCode INT  
SELECT @ReturnCode = 0  
declare @Date as datetime  
declare @NewDate as int  
declare @NewTime as int
```

microsoft.public.sqlserver.server: creating jobs programmatically causes deadlocks

```
DECLARE @String char(30)
SET @String = 'sa'

/*
SELECT @JobID = job_id
FROM msdb.dbo.sysjobs
WHERE (name = @JobName)

IF (@JobID IS NOT NULL)
BEGIN
    -- Check if the job is a multi-server job
    IF (EXISTS (SELECT *
                FROM msdb.dbo.sysjobsservers
                WHERE (job_id = @JobID) AND (server_id <> 0)))
    BEGIN
        -- There is, so abort the script
        RAISERROR ('Unable to import job %s since there is already a
multi-server job with this name.', 16, 1,@JobName) WITH LOG
        GOTO QuitWithRollback
    END
    ELSE
        RAISERROR ('Unable to create job %s since it already exists.', 16,
1,@JobName) WITH LOG
        GOTO QuitWithRollback
    END
*/
-- Add the job
EXECUTE @ReturnCode = msdb.dbo.sp_add_job @job_id = @JobID OUTPUT ,
@job_name = @JobName, @owner_login_name = @String, @description = N'Automated
Job Run', @category_name = N'ODS Automated', @enabled = 1,
@notify_level_email = 0, @notify_level_page = 0, @notify_level_netsend = 0,
@notify_level_eventlog = 3, @delete_level= @DeleteWhenDone
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

-- Add the job steps
EXECUTE @ReturnCode = msdb.dbo.sp_add_jobstep @job_id = @JobID, @step_id =
1, @step_name = @JobName, @command = @CmdText, @database_name = N'', @server
= N'', @database_user_name = N'', @ subsystem = N'CmdExec',
@cmdexec_success_code = 0, @flags = 0, @retry_attempts = 2, @retry_interval =
5, @output_file_name = N'', @on_success_step_id = 0, @on_success_action = 1,
@on_fail_step_id = 0, @on_fail_action = 2
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXECUTE @ReturnCode = msdb.dbo.sp_update_job @job_id = @JobID,
@start_step_id = 1
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

-- Add the job schedules
--in case of restart
EXECUTE @ReturnCode = msdb.dbo.sp_add_jobschedule @job_id = @JobID, @name
= N'Run on start', @enabled = 1, @freq_type = 64
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
```

microsoft.public.sqlserver.server: creating jobs programmatically causes deadlocks

--in case of Idle condition

```
IF (@RunOnIdle = 1)
BEGIN
EXECUTE @ReturnCode = msdb.dbo.sp_add_jobschedule @job_id = @JobID, @name
= N'Run on Idle', @enabled = 1, @freq_type = 128
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
END
-- Delay start
SET @String = 'Run job soon'
set @Date = dateadd(ss,@DelaySeconds,Getdate())
set @NewDate = cast(convert(varchar(8), @Date, 112) as int)
set @NewTime = cast(replace(convert(varchar(8), @Date, 108), ':', '') as
int)
EXECUTE @ReturnCode = msdb.dbo.sp_add_jobschedule @job_id = @JobID, @name
= @String, @enabled = 1, @freq_type = 1, @active_start_date = @NewDate,
@active_start_time = @NewTime
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
-- Add the Target Servers
EXECUTE @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @JobID,
@server_name = N'(local)'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
COMMIT TRANSACTION
GOTO EndSave
```

QuitWithRollback:

```
RAISERROR ('Error %s while creating JobName= %s ', 16, 1,@@Error,
@JobName) WITH LOG
IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
```