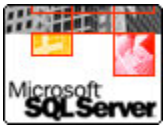
Artigos: Analisando a Performance do Servidor - CheckList

Uma das grandes dificuldades de muitos DBA's, principalmente aqueles que estão iniciando no SQL Server, é saber como e o quê verificar no momento de uma lentidão no sistema ou até mesmo no servidor como um todo. Neste artigo, descreverei um checklist básico que você poderá seguir para ajudar na hora de identificar a causa de uma lentidão.



Na verdade, a causa da lentidão em um servidor pode ser várias: uma query que não faz uso adequado dos índices, tabelas sem índices, proplemas de modelagem, falta de memória, discos ruins e assim vai...

Independente da causa da lentidão, alguns passos básicos podem ser seguidos para tentar identificar o problema. Neste artigo estarei descrevendo alguns desses passos e dizendo como interpretá-los durante a verificação de problemas de performance em seu servidor. Porém, quero deixar claro que os passos decritos aqui não é um padrão e que a metodologia de análise pode variar de DBA para DBA.

O PASSO-A-PASSO

Passo 1: Verificar as conexões existentes no servidor e se existe problemas de blocks.

O primeiro passo para identificar problemas de performance é saber o que está rolando no servidor. Quais são as conexões existentes? O que elas estão executando? Existem conexões bloqueadas? Quem é o causador do bloqueio? Quem está bloqueado? etc...

A procedure de sistema **sp_who** é uma procedure muito utilizada para responder estas questões, mas a principal pergunta a ser respondida aqui é: Existem conexões bloqueadas?

Para responder a esta pergunta execute a procedure **sp_who** e verifique se existe alguma conexão onde a coluna **Blk** seja diferente de 0 **(blk<>0)**. Se existir, já sabemos que estamos tendo problemas de block de conexões.

Passo 1.2: Havendo conexões bloqueadas, identificar o causador do bloqueio e a instrução SQL envolvida.

Uma vez detectado problemas de bloqueio, o próximo passo é identificar a conexão que está causando o bloqueio. Os bloqueios ocorrem porque normalmente duas ou mais conexões estão tentando utilizar um mesmo recurso. Imagine que temos duas conexões, uma com o SPID 64 e outra com o SPID 65. O SPID 65 efetua um lock exclusivo em uma tabela para efetuar uma atualização de dados e neste mesmo momento a conexão com SPID 64 tenta ler estas informações.

Neste caso, como a conexão SPID 65 abriu uma transação e efetuou um lock exclusivo, a conexão com SPID 64 ficará bloqueada pelo SPID 65 até que o mesmo conclua sua atualização. Uma das maiores causas de problemas de bloqueio são tabelas sem índices ou queries que não utilizam os índices de forma correta. Na maioria das vezes, a simples criação de uma índice clustered na tabela envolvida pode solucionar este tipo de problema.

Em grandes ambientes (ambientes com muitas conexões), a dificuldade aqui pode estar em identificar a conexão causadora do bloqueio. Isto porque muitas das vezes pode haver um certo encadeamento de bloqueio. Para facilitar este trabalho, você poderá estar utilizando a procedure **sp_usrheadblocker**.

Ao executá-la você obterá as seguintes informações:
--**SPID =>** Este é o identificador da conexão.
--**Blocked =>** informa o SPID que está bloqueando a conexão. Os SPIDs que possuirem Blocked=0 é o SPID que está causando os problemas de bloqueio.
--**ECID =>** Informa se a conexão esta fazendo uso de pararelismo.
--**WaitTimeMS =>** informa o tempo (em milessegundo) que a conexão está aguardando pelo recurso. Ou seja, a quanto tempo ela está bloqueada.
--**RunAs =>** informa a quanto tempo (em minutos) a conexão está no servidor.
--**Status =>** O status da conexão.
--**CPU =>** informa o consumo de CPU pela conexão.
--**Physical_IO =>** informa o consumo de I/O pela conexão.
--**HostName =>** O nome da máquina que estabeleceu a conexão.
--**LoginName =>** o nome do login que estabeleceu a conexão.
--**DBName =>** o nome da base de dados.
--**Last_Batch =>** a hora em que a conexão foi estabelecida ou executou a última instrução.
--**open_tran =>** informa se existe alguma transação aberta pela conexão. O número apresentado aqui é o número de

transações abertas pela conexão.
--**MemUsage =>** quantidade de memória sendo utilizada pela conexão.
--**TextBuffer =>** informa a instrução T-SQL sendo executada pela conexão.

Na maioria das vezes, a primeira linha sempre será a conexão causadora do bloqueio e as demais, as que estão aguardando pela liberação do recurso. Nos casos onde você tiver block em vários recursos, as conexões causadoras dos boqueios sempre terão a coluna **Blocked=0** e serão as primeiras linhas.

No exemplo abaixo, temos que as conexões 64 e 68 estão sendo bloqueadas pelas conexões 65 e 66 respectivamente. Na coluna **TextBuffer**, podemos ver que as conexões 65 e 64 estão utilizando o mesmo recurso (a tabela Products). O mesmo acontece com as conexões 66 e 68.

```
SPID   Blocked WaitTimeMS  TextBuffer
------ ------- ----------- ------------------------------------------------------------------------
65     0       0           begin tran Update products SET UnitsInStock= 38 where ProductID=1
66     0       0           begin tran Update Orders SET ShipCountry = 'Brasil' where OrderID=10248
64     65      1408218     select * from Products
68     66      134890      select * from orders
```

Obs: Algumas colunas foram excluídas para facilitar a visualização.

Neste caso, as conexões 64 e 68 não serão liberadas enquanto as conexões 65 e 66 não concluírem o processo de atualização (efetuarem um commit tran) ou forem derrubadas com a utilização do comando Kill. Um ponto importante a ser observado é que nem sempre o comando Kill finalizará o processo de forma imediata. Dependendo da instrução sendo executada e dos status das conexões, a conexão pode entrar em Rollback e demorar um certo tempo para ser finalizada.

Caso a coluna TextBuffer não esteja lhe fornecendo informações suficientes sobre a instrução T-SQL sendo executada pela conexão, você pode executar também o comando **DBCC Inputbuffer(<spid>)**. Porém, uma limitação do DBCC Inputbuffer é que nos casos em que a conexão estiver executando uma stored procedure ou view, ele lhe mostrará apenas o nome da procedure ou view. O ideal seria poder identificar o que realmente está sendo executado pela conexão naquele exato momento e para vencer esta limitação do comando DBCC Inputbuffer, você pode utilizar a procedure **sp_usrinputbuffer<spid>**.

Exemplo:

```
-- Executa o Comando DBCC InputBuffer no QA
dbcc inputbuffer (77)

EventType      Parameters EventInfo
-------------- ---------- --------------------------------
Language Event 0          exec SalesByCategory 'Beverages'


-- Para usar esta procedure o conexão deve estar com o status "Runnable"
Exec sp_usrinputbuffer 77

******** STATEMENT SENDO EXECUTADO NO MOMENTO ************
SELECT ProductName,
TotalPurchase=ROUND(SUM(CONVERT(decimal(14,2), OD.Quantity * (1-OD.Discount) * OD.UnitPrice)), 0)
FROM [Order Details] OD, Orders O, Products P, Categories C
WHERE OD.OrderID = O.OrderID
AND OD.ProductID = P.ProductID
AND P.CategoryID = C.CategoryID
AND C.CategoryName = @CategoryName
AND SUBSTRING(CONVERT(nvarchar(22), O.OrderDate, 111), 1, 4) = @OrdYear
GROUP BY ProductName
ORDER BY ProductName
```

Passo 2: Avaliar o estado do servidor.

Se o problema de lentidão ou performance não estiver associado com o bloqueio das conexões, o próximo passo a seguir é

fazer uma análise em alguns contadores do Perfmon (Performance Monitor) para tentar identificar possíveis gargalos no nível de hardware.

Alguns dos principais contadores a serem monitorados são listados abaixo:

A) Contadores de Memória

Memory: Available Bytes ==> Indica a quantidade de memória disponível no servidor.

Mamory: Pages/Sec ==> Indica se está havendo paginação. Em perfeitas condições deve-se ver pouca atividade neste contador. Se o SQL esta rodando em um servidor dedicado, toda a sua memória alocada (aquela configurada nas propriedades do servidor dentro do Enterprise Manager) será mapeada para a memória física e muito pouco swapping (paginação) deverá ocorrer. Se este contador estiver muito alto ou crescente, verifique a memória alocada para o SQL Server, aumentar este valor ou colocar o SQL Server para configurar sua memória dinamicamene pode ajudar a reduzir a paginação.

B) Contadores do SQL Server

SQL Server: Buffer Manager: Buffer Cache Hit Ratio ==> Indica o percentual de requisições a dados que são obtidas no próprio cache do sql e portanto, sem precisar acessar o disco. O ideal é estar maior que 90%.

SQL Server:Access Methods:Page Splits/sec ==> Mostra quantos page splits estão ocorrendo no servidor. Este valor deve ser o mais baixo possível. Se o valor estiver alto, configurar os índices com um fillfactor apropriado pode ajudar a reduzir este valor.

SQL Server: Memory Manager: Total Server Memory (KB) ==> Quando maior que a quantidade de memória física pode indicar falta de memória.

C) Contadores de CPU

Processor:% Processor Time:_Total ==> Indica o consumo de CPU no servidor (ideal abaixo de 80%).

Process: % Processor time: sqlservr ==> Indica o consumo do processador pelo processo do SQL Server.

System: Processor Queue Length ==> Indica o número de threads aguardando para execução no processador e nunca deve exceder 1 ou 2 (por processador) por um período superior a 10 minutos.

D) Contadores de Disco

PhysicalDisk: % idle Time ==> Indica o percentual de tempo que o disco esta ocioso. Supostamente este contador apresenta uma estimativa mais precisa sobre a utilização dos discos. Subtraindo o valor encontrado de 100, temos uma ideia do quanto o disco esta trabalhando.

PhysicalDisk: % Disk Time ==> Mostra o quanto o disco esta ocupado. Se valor > 60% durante 10 minutos, verificar os contadores **PhysicalDisk: % Disk Read Time** e **PhysicalDisk: % Disk Write Time** para identificar se é leitura ou escrita quem está causando maior utilização.

PhysicalDisk: Avg. Disk Queue Length ==> Mostra o número médio de requisições de I/O aguardando para acesso ao disco. Este contador nunca deve exceder 2 (por disco) por um extenso período de tempo.

PhysicalDisk: Avg. Disk Sec/Transfer ==> Taxa de transferência em bytes de ou para um disco durante operações de leitura ou escrita.

D) Contadores de Banco de Dados

SQL Server: Access Methods: Full Scans/sec ==> Mostra a quantidade de Table Scan sendo executada no SQL Server. Este valor deve ser o mais baixo possível e se for constantemente alto, use o Profile para identificar quais queries estão causando os scans.

SQL Server: Locks: Average Wait Time (ms) ==> Este contador mostra quanto tempo em milessegundos um processo está aguardando para obter lock no SQL Server. O valor ideal para este contador é zero. Se este valor estiver muito alto,

utilize o Profile para identificar quais processos estão mantendo locks por um período longo de tempo.

Como descrito no início deste artigo, o problema de permance pode estar associado a várias causas, este checklist segue alguns passos básicos para ajudar na identificação do problema. Também é importante observar que este checklist não é um padrão e que por isso, os métodos utilizados para analizar a performance de um servidor pode variar de DBA para DBA. Na verdade, no dia-a-dia de trabalho cada DBA acaba por criar a sua própria metodologia de análise.

Caso prefira, você pode estar baixando aqui um .xls (documento Excel) com os passos descritos neste artigo.

Um abraço a todos
Nilton Pinheiro