Profiler

Criando uma trace para análise de performance de um servidor SQL Server

1. **Análise preliminar**: identificando stored procedures e batchs com baixa performance.

Demora excessiva para conclusão de consultas, **deadlocks** em demasia e problemas com timeout nas aplicações – esse é o quadro de um servidor com sérios problemas de performance. Nosso objetivo inicial será efetuar um levantamento para determinar quais são os processos mais demorados, cronometrando a execução de batchs e stored ptrocedures nesse servidor.

Passos para realizar esta tarefa:

1) inicie o **Profiler**.
2) Escolha New Trace na opção File da barra de ferramentas.
3) Na tela de propriedade da Trace, confirme a seleção do template Standard na guia General. (Esse template captura a execução de batchs(TSQL\SQL:batch Completed e Stored Procedures\RPC:Completed).
4) Os eventos Security Audit e Sessions não são necessários para essa análise, podendo ser removidos.

As colunas e ordem de apresentação dos eventos do template Standard sofrerão duas alterações: na guia Data Columns ordenaremos as linhas capturadas segundo seu tempo de execução e eliminaremos algumas colunas desnecessárias nesse momento.

5) para ordenar as linhas do Profiler segundo a duração do comando executado, selecione Duration e clique em Up até que essa coluna fique logo abaixo de Groups.
6) As colunas removidas foram: NTUserName, Loginname, CPU, Reads, Writes e ClientProcessId.

Até esse momento, a trace irá capturar todos os batchs e sp´s executados nesse servidor. Seria interessante ligar um filtro de tempo de modo que só fossem capturados os processos com tempo de execução acima de um determinado limite.

7) Para conseguir esse efeito, vamos ligar o filtro Duartion na guia Filters, trabalhando com um limite de 2000 milisegundos.

8) Agora inicie a trace clicando <RUN>.
9) Para simular uma atividade no SQGBD, abra uma sessão no Query Analyzer, digite as linhas da **Listagem 1** e confirme o resultado na tela do Profiler.

```
use Northwind
go

create proc stp_teste
as
  select top1 * from orders
  select top 1 * from [order details]
  waitfor delay '00:00:03'
  select top 1 * from customers
go

exec stp_teste
go
select count(*) from [order details]
go
select top 10 * from orders
go
```

Listagem 1

2. Análise específica: identificando comandos T-SQL com baixa performance.

A trace gerada anteriormente forneceu a relação de batchs e sp´s que estão apresentando tempo de execução superior a 2 segundos. No caso da stored procedute stp_teste, temos que identificar qual comando T-SQL está sendo responsável pela lentidão. Para isto, vamos criar uma trace para analisar somente a sp stp_teste.

1) O ponto de partida é identificar o id do objeto que queremos selecionar. No Query Analyzer, digite:

```
Select object_id ('stp_teste')
----------------------------------
1637580872
```

2) Crie uma trace, selecionando os eventos SP:Completed e SP:StmtCompleted.

SP:Completed – irá gerar uma linha contendo a chamada para a procedure.

SP:StmtCompleted – irá gerar uma linha para cada comando T-SQL executado dentro da sp.

3) Na guia Data Columns, selecione o mesmo evento analisado anteriormente “Duration”.

4) Na guia Filters, siga até ObjectId e informe o id da procedure stp_teste obtida anteriormente. Veja a **Figura 1**.

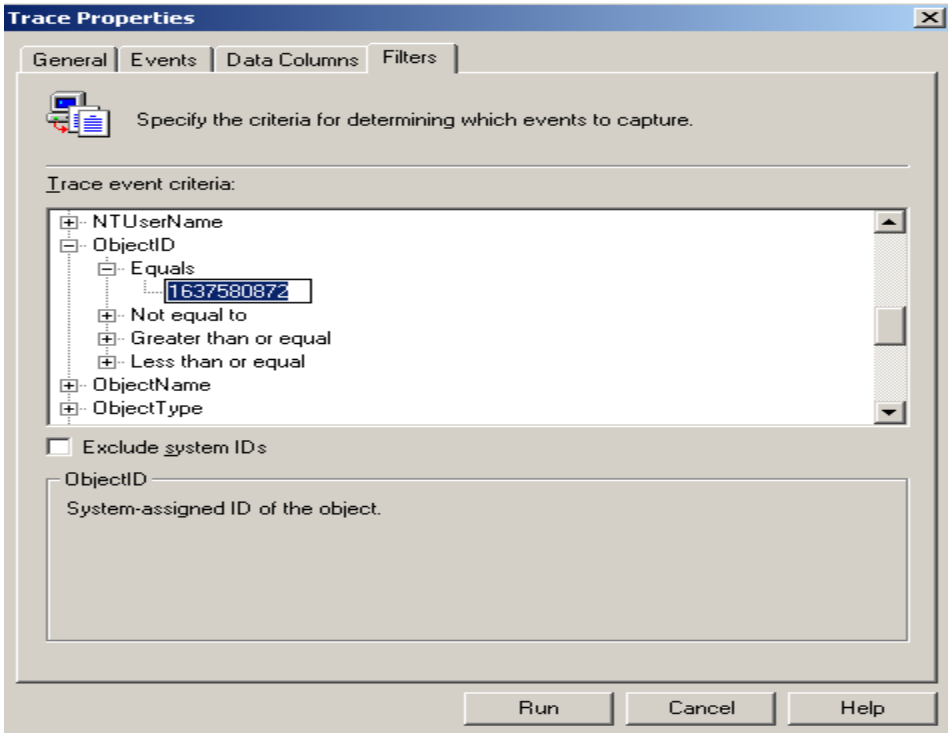
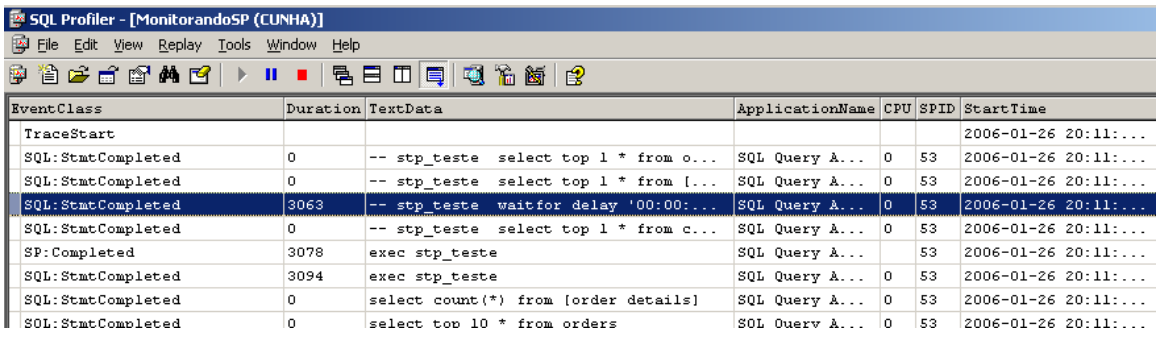


Figura 1: Ligando o filtro ObjectId para filtrar os comandos T-SQL executados na procedure stp_teste.

5) Rode a trace <RUN>
6) Execute a procedure stp_teste. Veja o resultado na tela profiler (**Figura 2**).



EventClass	Duration	TextData	ApplicationName	CPU	SPID	StartTime
TraceStart						2006-01-26 20:11:...
SQL:StmtCompleted	0	-- stp_teste select top 1 * from o...	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	0	-- stp_teste select top 1 * from [...	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	3063	-- stp_teste waitfor delay '00:00:...	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	0	-- stp_teste select top 1 * from c...	SQL Query A...	0	53	2006-01-26 20:11:...
SP:Completed	3078	exec stp_teste	SQL Query A...		53	2006-01-26 20:11:...
SQL:StmtCompleted	3094	exec stp_teste	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	0	select count(*) from [order details]	SQL Query A...	0	53	2006-01-26 20:11:...
SQL:StmtCompleted	0	select top 10 * from orders	SQL Query A...	0	53	2006-01-26 20:11:...

Figura 2: Visualização dos comandos executados na procedure stp_teste.

3) Rastreando processos envolvidos em Deadlocks

Deadlock é uma causa freqüente de má performance, pois requer que a conexão escolhida como vítima – e cuja execução de comandos foi abortada- envie novamente o comando para execução. O profiler possui dois eventos que nos ajudam a rastrear deadlocks:

- Lock:Deadlock – informa a ocorrência do erro#1205 associado ao deadlock.
- Lock:DeadlockChain – irá apontar o spid das conexões envolvidas no deadlock.

4) Evitando CacheMiss

Quando acionamos uma sp, seu plano de execução fica em memória. Nesse plano são armazenadas instruções de como a query deverá ser executada: que índice utilizar, o tipo de join selecionado, etc. antes de criar um plano de execução novo, o otimizador faz uma busca na área de cachê destinada a procedures procurando por um plano pré-existente. Se a busca for bem sucedida, o plano será aproveitado e a sp será executada de acordo com ele.

Existem algumas situações onde o otimizador é obrigado a fazer várias tentativas até encontrar o código pré-compilado. Cada uma dessas tentativas mal sucedidas dispara um evento conhecido por ***SP:CacheMiss***, que pode ser evitado seguindo-se as dicas abaixo:

- Não utilize o prefixo "sp_" para nomear suas sp´s. Quando o otimizador encontra uma procedure com o prefixo sp, sua execução é desencadeada no banco de dados master. Como a sp não existe nesse banco de dados, seu plano de execução não é encontrado na primeira tentativa, disparando um evento **SP:cacheMiss**. O processo de execução prossegue procurando a sp no banco de dados local, onde o código compilado é encontrado e a sp executada.
- Execute stored procedure qualificando seu dono (owner). Lembre-se que você pode possuir objetos com o mesmo nome, mas com owner diferentes. Troque exec stp_teste por exec dbo.stp_teste.

Principais comandos utilizados na manipulação de traces:

1) Para gerar o script de uma trace: na barra de ferramentas do profiler, selecione File...Save As SQL Script.
2) Verificando as traces ativas no servidor:

```
Select * from ::fn_trace_getInfo(default)
```

3) Pasa parar uma trace:

Sp_trace_setstatus 1,0

<1>: id da trace, levantado no item -2
<<0>: parâmetro para parar a trace.

4) Para iniciar (ou restaurar) uma trace:

Sp_trace_setstatus 1,1

<1>: id da trace, levantado no item -2
<1>: parâmetro para iniciar a trace.

5) Para encerrar uma trace, excluindo sua definição da memória do servidor:

St_trace_setstatus 1,2

<1>: id da trace, levantado no item -2
<2>: parâmetro para encerrar a trace.

PS: é necessário parar antes.

Nota: Funções internas no SQL Server que possuem o prefixo “fn_” precisam ser chamadas com a adição de “::” (duas vezes o sinal de pontuação dois pontos).