Configurações de performance no SQL Server 2005

José Antônio da Cunha

CEFET-RN

Configuração de performance

Para obter o máximo de performance , DBAs configuram o SQL Server para atender às suas necessidades de negócio e muitas vezes acabam alterando configurações de forma equivocada.

Número de configurações possíveis de alterar:

- No SQL Server 2000 = 37
- No SQL Server 2005 = 64

Configuração de performance

Conhecendo as configurações:

A maneira mais fácil de obter a lista das configurações que o banco de dados possui é executando uma stored procedure de servidor chamada S**P_CONFIGURE**.

Esta stored procedure irá listar as configurações que estão sendo utilizadas atualmente no banco de dados, e seu resultado pode variar dependendo da versão do SQL Server.

```
EXEC SP_CONFIGURE
```

Repare que para cada configuração, temos cinco colunas de informações:

Configuração de performance

- **Name**: nome da configuração;
- **Minimun:** valor mínimo permitido para a configuração listada;
- **Maximun:** valor máximo permitido para a configuração listada;
- **Config_value**: valor configurado;
- **Run_value:** valor que está sendo executado. Pode ser que o valor que está sendo executado não seja o valor configurado, isto acontece porque determinadas configurações, quando alteradas, só entrarão em vigor após o servidor ser reiniciado.

Configuração de performance

Alterando uma configuração

A stored procedure de servidor SP_CONFIGURE, além de exibir as configurações, pode ser utilizada também para alterá-las. A sintaxe para alterar uma configuração é a seguinte:

```
SP_CONFIGURE [‘nome da configuração’], [novo valor a ser configurado]
GO
RECONFIGURE WITH OVERRIDE
GO
```

Observe que mesmo utilizando o comando **RECONFIGURE WITH OVERRIDE** que serve para alterar uma configuração em tempo de execução, algumas configurações só serão realmente executadas ao reiniciar o servidor, como é o caso do ‘recovery interval’ (intervalo de recuperação do banco de dados).

Configuração de performance

Como exemplo, vamos alterar a configuração **clr enabled** para desabilitar o suporte ao **CLR** (common langauge runtime) no banco de dados, alterando o valor de 1 para 0.

```
SP_CONFIGURE ‘clr enabled’, 0
GO
RECONFIGURE WITH OVERRIDE
GO
```

```
Para confirmar alteração
SP_CONFIGURE ‘clr enabled’
```

Configuração de performance

Configurações que influencia a performance

Dentre as configurações listadas através da stored procedure de servidor SP_CONFIGURE, vale destacar algumas configurações importantes para o conhecimento e manuteção do banco de dados e, com isso, obter um ganho considerável de desempenho:

- **Recovery Interval,**
- **Network Packet Size,**
- **Min Memory Per Query,**
- **Max Degree Of Parallelism**,
- **Priority Boost**,
- **Max Server Memory,**
- **Min Server Memory,**
- **Index Create Memory,**
- **Nested Triggers**,
- **Query Governor Cost Limit**.

Configuração de performance

Recovery Internal – alterar o intervalo de recuperação dos objetos que compõem o banco de dados quando alguma falha acontecer. Se o seu SQL Server é muito ativo, recebendo uma carga enorme de transações e mantém o monitor de performance em **100%** na maior parte do tempo, esta é uma boa configuração a ser alterada. Por padrão, o SQL Server tem o **valor 0** nesta configuração, o que significa que ele terá até 1 minuto para realizar uma recuperação após reiniciar, o que pode não ser suficiente. Para este caso, podemos aumentar este intervalo para 5 minutos, como no exemplo abaixo:

```
Exec SP_CONFIGURE ‘recovery interval’, 5
GO
RECONFIGURE WITH OVERRIDE
GO
```

Configuração de performance

Network Packet Size: altera o tamanho padrão de pacotes enviados pelas transações do SGBD. O tamanho padrão é 4096 bytes para cada pacote. Esta configuração pode ser alterada para um valor maior quando a sua aplicação armazena e trafega com freqüência dados com um grande número de bytes, por exemplo, imagens e longos trechos de texto.

Aumentar o número de bytes, temos menos pacotes trafegando e assim aceleramos o tráfego na rede. Por outro lado, se o banco de dados trafega dados em pouca quantidade, aumentar o tamanho dos pacotes pode ser prejudicial.

```
Exec SP_CONFIGURE ‘network packet size’, 6000
GO
RECONFIGURE WITH OVERRIDE
GO
```

Configuração de performance

Min Memory Per Query: altera o valor mínimo da memória para cada query. O valor mínimo padrão é 1024Kb, podendo ser aumentado somente quando o servidor possui muita memória RAM disponível e se existe a certeza de que as queries disparadas ocupam um valor bem maior que o padrão. Ou seja, as chamadas consultas “pesadas”.

Como exemplo, considere uma consulta que traga milhares de registros de várias tabelas, e estes registros precisem ser ordenados pelo usuário. Como o espaço em memória requerido para esta tarefa é grande e sempre irá ultrapassar o tamanho padrão de 1024Kb, pode-se configurar um tamanho mínimo padrão de memória maior.

```
Exec SP_CONFIGURE ‘min memory per query’, 3072
GO
RECONFIGURE WITH OVERRIDE
GO
```

Configuração de performance

Max Degree Of Parallelism: altera o grau máximo de paralelismo. Por padrão, o valor 0 indica que o paralelismo está ativado e pode usar os CPU´s existentes no servidor. Se você alterar esta definição para 1, o paralelismo é desativado para todas as CPU´s.

Devemos desativar o paralelismo somente quando for desnecessário o uso de múltiplas CPU´s, por exemplo, aplicações que executam consultas que retornam poucos registros e fazem pouco uso de processamento. Para estas, aconselha-se utilizar somente uma CPU, poupando as outras CPU´s para serviços com maiores prioridades.

```
Exec SP_CONFIGURE ‘max degree of parallelism’, 0
GO
RECONFIGURE WITH OVERRIDE
GO
```

Nota: Paralelismo é a capacidade do Query Optimizer de usar múltiplas CPUS para executar uma única consulta.

Configuração de performance

Priority Boost: altera a prioridade dos processos referentes ao SQL Server no servidor. Aqui temos uma configuração muito interessante: os processos oriundos do SQl server terão uma prioridade maior de execução que de outros aplicativos no servidor, podendo ser utilizado quando temos a certeza de que existem no servidor outras aplicações rodando além do SQL Server, e que sejam aplicações com baixo grau de importância quanto à performance. Não o utilize quando tiver apenas o SQl Server rodando no Servidor.

```
Exec SP_CONFIGURE ‘priority boost’, 1
GO
RECONFIGURE WITH OVERRIDE
GO
```

O valor padrão é 0 → sem prioridade

1 → prioridade maior

Configuração de performance

Max Server Memory e Min Server Memory: configuram o valor máximo e mínimo de memória para um servidor SQL, respectivamente. Esta opção pode ser interessante quando, no servidor onde reside o SQL Server, existirem aplicações que em determinados momentos consomem muita memória, deixando pouca para o SQL Server. Caso o seu banco de dados seja a única aplicação no servidor, aconselha-se não alterar esta configuração, porque o uso da memória já estará otimizado.

```
Exec SP_CONFIGURE ‘min server memory (MB)’, 2
GO
RECONFIGURE WITH OVERRIDE
GO
```

```
Exec SP_CONFIGURE ‘max server memory (MB)’, 3
GO
RECONFIGURE WITH OVERRIDE
GO
```

Configuração de performance

INDEX Create Memory: define o tamanho de memória utilizado para criação de índices. O valor padrão 0 diz para o SQL Server determinar automaticamente o valor ideal a ser utilizado. Em praticamente todos os casos, o SQL Server irá configurar a quantidade de memória otimizada, por exemplo, em índices cluster. Em outros casos, pode acontecer de um índice ocupar espaço demais em uma tabela e tornar as consultas lentas. Somente se essa ocorrência for verificada, é que se torna importante definir um espaço em memória para os índices.

```
Exec SP_CONFIGURE ‘index create memory’, 2048
GO
RECONFIGURE WITH OVERRIDE
GO
```

Configuração de performance

NESTED TRIGGERS: configura o uso ou não de triggers aninhadas. Por padrão é configurada a opção 1, que significa que o uso de triggers aninhadas está ativo. Sabemos que eles consomem uma boa parte do processamento de uma transação, então caso deseje que os desenvolvedores não usem este tipos de triggers, basta setar esta configuração para o valor 0.

```
Exec SP_CONFIGURE ‘nested triggers’, 0
GO
RECONFIGURE WITH OVERRIDE
GO
```

Configuração de performance

Query Governor Cost Limit: configura o limite de custo de uma consulta. Esta opção é uma das poucas que normalmente sofrem alterações por parte dos DBAs, pois permite que seja gerenciado o tempo em segundos para uma consulta ser disparada. O valor padrão para essa configuração é 0, o que significa que não há limites para quanto tempo uma consulta pode ser executada. Deixar o banco de dados com esta configuração pode ser perigoso para consultas lentas, principalmente quando há diversos usuários simultâneos acessando o banco de dados.

```
Exec SP_CONFIGURE 'query governor cost limit', 50 → segundos
GO
RECONFIGURE WITH OVERRIDE
GO
```

Observação: Com o tempo em segundos limitado, os desenvolvedores serão levados a criar consultas mais rápidas.

Configuração de performance

Configurações avançadas

O SQL Server 2005 é um SGBD onde a maioria das configurações já vêm ajustadas para se obter a melhor performance possível. Elas podem ser alteradas conforme a regra de negócio, mas no geral é aconselhável que se mantenha o valor padrão já configurado no banco. As configurações avançadas só podem ser alteradas quando a configuração **show advanced options** estiver setada com o valor 1.

Conheça algumas destas configurações, onde na maioria dos casos, aconselha-se manter o valor padrão.

Configuração de performance

AWE ENABLED: esta é a configuração utilizada somente em situações que o SQL Server 2005 esteja rodando em um ambiente de 32 bits. Se o servidor onde o SQL Server reside possui 4GB de memória RAM ou menos, aconselha-se manter o valor padrão (0, zero). Isto faz com que a memória AWE não seja utilizada. A **API AWE** permite executar aplicações que foram desenvolvidas para usar mais de 4GB de memória RAM no Windows 2003 Enterprise Server ou Windows 2003 Datacenter Serrver.

Lembrando que se o sistema operacional for Windows 2003 Server, tanto o SQL Server 2005 Standard com o Enterprise Edition podem usar até **32GB** de memória RAM. Se o sistema operacional for Windows 2003 Datacenter Server, estes dois SGBDs podem utilizar até **64GB.**

Atenção! para aplicações de pequeno e médio porte, aconselha-se utilizar o valor 0.

Nota: A AWE (Address Windows Extensions) é uma API específica do Windows que realiza o acesso direto aos endereços de memória RAM com tamanho acima de 4GB por aplicações que fazem alto uso de memória.

Configuração de performance

AFFINITY I/O MASK: a affinity I/O mask permite que possamos configurar um conjunto de CPUs multiprocessadas para um servidor. O valor a ser configurado depende do número de CPUs a serem utilizadas, conforme a **Tabela 1**.

Valor	Descrição
1	Configura o uso de até 8 CPUs em um computador.
2	Configura o uso de até 16 CPUs em um computador.
3	Configura o uso de até 24 CPUs em um computador.
4	Configura o uso de até 32 CPUs em um computador.

Tabela 1. Opções da configuração Affinity I/O Mask.

O valor 0(zero) significa que o SQL Server está programado para ser executado em qualquer número de CPUs disponíveis.

Configuração de performance

AFFINITY MASK: Se o Affinity I/O Mask informa o número total de CPUs utilizadas no servidor, é o Affinity Mask que define quais processadores o SQL Server usará, podendo então escolher 4 entre 6 processadores disponíveis, por exemplo. Isto pode ser interessante em servidores que rodam múltiplas instâncias do SQL Server e/ou outros aplicativos , onde você pode definir que processador utilizar para cada um.

O valor padrão é 0(zero). Caso você só tenha o SQL Server rodando em um servidor, então não altere este valor. Isto permite utilizar um algoritmo do sistema operacional específico para determinar que thread executar, em que CPU, e quando mover uma thread de uma CPU para outra.

Configuração de performance

Alterar as configurações do SQL Server pode ser uma ótima maneira de otimizar desempenho, porém requer um considerável conhecimento da estrutura e funcionamento do banco de dados. Isto é bom quando conhecemos estas configurações e podemos utilizá-las para adaptar o nosso banco a cada situação, mas também pode ser perigoso, pois uma série de recursos que foram automaticamente configurados para uma boa performance podem ser alterados de forma errada, gerando gravíssimos problemas no banco de dados. O ideal é realizar as configurações em um servidor de teste, e caso sucesso, implementá-las no banco em produção.