数据库文件增长对主机性能的影响分析

数据库文件增长对主机性能的影响

在使用 SQL Server 数据库时，我们经常会碰到数据文件或事务日志文件增长的情况，DBA 针对这一问题会制定一些技术维护计划或者直接对数据库文件进行缩减。在本技术专题中，我们将通过大量测试来验证数据库文件增长对主机性能的影响，其中包括数据文件和事务日志文件，通过测试结果，相信将会对 SQL Server 数据库性能的维护工作有所帮助。

SQL Server 数据文件增长检测

我经常能看到人们忙着建立 SQL Server 技术维护计划，定期制定关于缩减数据库文件（数据或者 T-Log）的工作。在客户调查数据库增长原因之前，我总是建议他们不要缩减数据库文件，特别是不要定期缩减。我将做一个测试来证明，由事务处理导致的数据库文件增长对 SQL Server 性能的影响有多大。

- SQL Server 数据文件增长检测（一）
- SQL Server 数据文件增长检测（二）
- SQL Server 数据文件增长检测（三）

SQL Server 中测试事务日志的自动增长

比起数据文件，事务日志的增长速度毫不逊色，因此数据库管理员会经常缩减删除事务日志。在本文中，我将对事务日志文件增长造成的影响做一个测试。而在本次测试中，我将使用和上一次测试相同的环境和配置。唯一的不同是，这一次我在 SQL Server 上安装了 SP3。

- 在 SQL Server 上测试事务日志的自动增长（一）
- 在 SQL Server 上测试事务日志的自动增长（二）
- 在 SQL Server 上测试事务日志的自动增长（三）

SQL Server 事务日志自动增长对性能的影响

在上一次测试中，我们验证了 SQL Server 中事务日志的自动增长特点，在本次测试中我们将对 SQL Server 中事务日志的自动增长对主机性能的影响作出分析，而测试环境同前两次的完全相同。

- SQL Server 中事务日志自动增长对性能的影响（上）
- SQL Server 中事务日志自动增长对性能的影响（下）

SQL Server 数据增长检测（一）

我经常能看到人们忙着建立 SQL Server 技术维护计划，定期制定关于缩减数据库文件（数据或者 T-Log）的工作。在客户调查数据库增长原因之前，我总是建议他们不要缩减数据库文件，特别是不要定期缩减。

我和好多人讨论过这个问题，他们更关心硬盘空间而不是主机性能，因此我决定做一个测试来向他们证明，由事务处理导致的数据库文件增长对 SQL Server 性能的影响有多大。

我将测试一下几个增长情况：

1、数据库文件增长；

2、数据库 T-Log 增长；

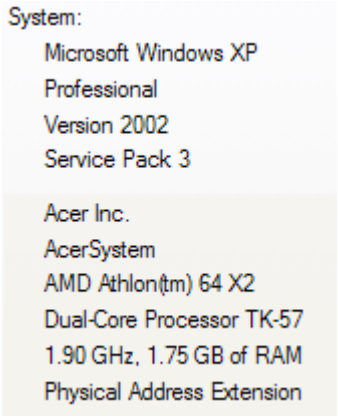
3、Tempdb 增长；

以下是 SQL Server 数据库文件增长一系列测试的第一部分：

测试环境

软硬件配置

本次测试的软硬件配置如下图所示：



图一：测试系统的软硬件配置

SQL Server 2005 SP2, Enterprise Evaluation Edition。

电脑有一个本地 iSCSI 逻辑单元号。

数据库配置

我创建了一个数据库，命名为 ShrinkDB，数据库配置如下图：

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
ShrinkDB	Data	PRIMARY	2	By 1 MB, unrestricted growth
ShrinkDB_log	Log	Not Applicable	10	By 1 MB, restricted growth to 2097152 MB

图 2：ShrinkDB 数据库配置

有以下查询语句：

```
select size from sysfiles where fileid = 1
Returns: 256
```

文件大小：256*8k=2MB。

在数据库中，我创建了下面的表：

```
create table ExpandDB （a varchar（8000））
```

预期目标

我的目标是做出基于事务大小和数据文件 Autogrowth 比的性能测试：

1、测试 1——小 autogrowth，少量事务，插入 X 行；

2.、测试 2——小 autogrowth，少量事务，插入 X*Y 行（比测试 1 插入的行数多）；

3、测试 3——大 autogrowth，少量事务，插入 X*Y 行（同测试 2 插入的行数一样多）；

4、测试 4——小 autogrowth，在一个事务中插入 X*Y 行；

5、测试 5——大 autogrowth（同测试 3），在一个事务中插入 X*Y 行；

6、测试 6——超大 autogrowth，少量事务；

7、测试 7——超大 autogrowth，大量事务；

我会分析不同的结果并最终得出结论。

注意：测试不包含并发性。

方法与代码说明

针对每个测试，我为数据库文件规定了文件大小个 autogrowth 比。例如：初始大小=2MB，autogrowth=1MB。

我为数据库文件还规定了一个目标文件大小， 所以在第一个执行的循环中，插入的行数应达到数据文件和目标文件大小相同的标准：

```
while （select size from sysfiles where fileid = 1） <=
  insert into ExpandDB select replicate （'a',8000））
```

这个循环向 ExpandDB 中插入了 8000 字节的行，直到数据库文件大小超出 X。ExpandDB 表是一个一般关系表，没有索引和约束条件，这样可以避免特殊优化和索引的开销。

在运行完第一个循环并达到目标文件大小时，为作比较我还要执行相同的插入但不增加文件。所以，我必须截短表并清理事务日志来确保 T-Log 也不增长。

输入以下查询：

```
select count（*） from ExpandDB
```

返回第一循环中插入行的数量，

```
select size from sysfiles where fileid = 1
```

返回新数据库中的数据文件大小。

以下语句将截短表和 T-Log：

```
truncate table ExpandDB
go
backup transaction ShrinkDB with truncate_only
go
```

下面一步是执行同样的插入命令：

```
declare @i int
set @i = 1
while @i <=
begin
insert into ExpandDB select replicate （'a',8000）
set @i = @i + 1
end
```

第一循环结束后，就是表中的行数。

另一件重要的事就是在执行第一第二循环之前一定要清除缓存，只有这样才能确保一个公平的性能比较并清除缓存造成的不平等：

DBCC DROPCLEANBUFFERS

最后，每个测试都要执行三次（三个循环）以保证结果的相容性。

所以，每次测试的第一步是确保数据文件达到初始大小。例如，缩减数据库文件：DBCC SHRINKFILE （N'ShrinkDB', 0, TRUNCATEONLY）

（作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国）

SQL Server 数据增长检测（二）

性能监控工具

性能由 SQL Profiler 来监控。

结果总结

测试的结果由表格呈现，对比了两次循环中的 CPU、读写和持续时间情况，第一次循环有文件增长而第二次没有。

有两行作比较：百分比差异（<第二循环 >/<第一循环 > *100）和总数差异（<第一循环 > -<第二循环 >）。如果结果是正面的，那么说明第二循环性能更佳。相反的说明第一循环性能更佳。在每个对比表的最后，对比的平均只包括了三个循环的平均值。

测试结果：

测试 1

初始文件大小= 256,

目标文件大小 = 8448,

少量事务（1 次一行）,

总行数: 8160,

文件增长 1 MB

目标文件 8448 比初始文件增长了 33 倍。

我运行了几次以下的代码：

```
--=====================================================================
-- Truncate the table to free space
truncate table ExpandDB
go
-- Shrink database to original size (2MB)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Check that the size was reduced properly
select size from sysfiles where fileid = 1
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
--=== First insert ==========
-- Fill the table + Grow the database file x 30
while (select size from sysfiles where fileid = 1) <= 8400
      insert into ExpandDB select replicate ('a',8000)
go
-- Make sure that the number of rows does not change
select count(*) from ExpandDB
go
-- What is the size of the file now?
select size from sysfiles where fileid = 1
go
-- Truncate the table to free space
truncate table ExpandDB
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
--=== Second insert ==========
-- Fill the table without growing the file
declare @i int
set @i = 1
while @i <= 8160     -- Same number of rows inserted in the first loop
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- What is the size of the file now?
select size from sysfiles where fileid = 1
go
```

运行以上代码后，SQL Profiler 显示的结果：

TextData	CPU	Reads	Writes	Duration
--==...	0	42	5	186
-- Shrink database to original size (2MB) DBCC SHRINKFILE (...	47	3142	8	703
-- Clear cache DBCC DROPCLEANBUFFERS	0	0	0	153
-- Clear transaction log (so it will not grow) backup trans...	0	59	2	94
--=== First insert ========== -- Fill the table + Grow the ...	2031	69850	8222	13045
-- Truncate the table to free space truncate table ExpandDB	0	43	6	176
-- Clear cache DBCC DROPCLEANBUFFERS	16	0	0	155
-- Clear transaction log (so it will not grow) backup trans...	15	183	6	236
--=== Second insert ========== -- Fill the table without gr...	750	55234	8210	7999
--==...	0	42	5	33
-- Shrink database to original size (2MB) DBCC SHRINKFILE (...	16	3142	9	154
-- Clear cache DBCC DROPCLEANBUFFERS	0	0	0	0
-- Clear transaction log (so it will not grow) backup trans...	16	59	2	100
--=== First insert ========== -- Fill the table + Grow the ...	1859	69853	8218	12735
-- Truncate the table to free space truncate table ExpandDB	0	43	5	176
-- Clear cache DBCC DROPCLEANBUFFERS	16	0	0	6
-- Clear transaction log (so it will not grow) backup trans...	0	183	6	175
--=== Second insert ========== -- Fill the table without gr...	1078	55533	8210	7230
--==...	0	42	5	177
-- Shrink database to original size (2MB) DBCC SHRINKFILE (...	31	3142	8	434
-- Clear cache DBCC DROPCLEANBUFFERS	0	0	0	0
-- Clear transaction log (so it will not grow) backup trans...	0	59	2	0
--=== First insert ========== -- Fill the table + Grow the ...	1985	69853	8221	13150
-- Truncate the table to free space truncate table ExpandDB	0	43	5	34
-- Clear cache DBCC DROPCLEANBUFFERS	0	0	0	7
-- Clear transaction log (so it will not grow) backup trans...	15	183	6	230
--=== Second insert ========== -- Fill the table without gr...	1360	55316	8210	8324

结果总结：

循环	步骤	CPU	读	写	持续时间 (ms)
1	文件增长	2031	69850	8222	13045
	文件不增长	750	55234	8210	7999
	% 改进	63.07237814	20.92483894	0.145949891	38.68148716
	差异	1281	14616	12	5046
2	文件增长	1859	69853	8218	12735
	文件不增长	1078	55533	8210	7230
	% 改进	42.01183432	20.50019326	0.097347286	43.22732627
	差异	781	14320	8	5505
3	文件增长	1985	69853	8221	13150
	文件不增长	1360	55316	8210	8324
	% 改进	31.4861461	20.81084563	0.133803674	36.69961977
	差异	625	14537	11	4826
	平均% 改进	45.52345285	20.74529261	0.125700284	39.5361444
	平均差异	895.6666667	14491	10.33333333	5125.666667

测试 2

初始文件大小 = 256,

最终文件大小 = 34816,

少量事务 （每次一行），

总行数：33373,

文件增长 1 MB

现在我们来测试更多的插入，代码如下：

```
--============================================================
-- Truncate the table to free space
truncate table ExpandDB
go
-- Shrink database to original size (2MB)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Check that the size was reduced properly
select size from sysfiles where fileid = 1
go
-- Clear cache
DBCC DROPCLEANBUFFERS
Go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
--=== First insert ==========
-- Fill the table + Grow the database file x 30
while (select size from sysfiles where fileid = 1) <= 33600
      insert into ExpandDB select replicate ('a',8000)
go
-- Truncate the table to free space
truncate table ExpandDB
go
-- Clear cache
DBCC DROPCLEANBUFFERS
Go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
--=== Second insert ==========
-- Fill the table without growing the file
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
```

结果总结：

循环	步骤	CPU	读	写	持续时间 (ms)
1	文件增长	6730	285208	33618	56148
	文件不增长	4250	225591	33567	35003
	% 改进	36.84992571	20.9029901	0.151704444	37.65940016
	差异	2480	59617	51	21145
2	文件增长	7031	285223	33619	53373
	文件不增长	4204	225364	33564	32472
	% 改进	40.20765183	20.98673669	0.163597965	39.16024956
	差异	2827	59859	55	20901
3	文件增长	6453	285278	33618	54189
	文件不增长	3844	225362	33564	33530
	% 改进	40.43080738	21.00267108	0.160628235	38.1239735
	差异	2609	59916	54	20659
	平均% 改进	39.16279497	20.96413262	0.158643548	38.31454107
	平均差异	2638.666667	59797.33333	53.33333333	20901.66667

测试 3

初始文件大小= 256,

最终文件大小 = 34816,

少量事务（每次一行），

文件增长 10 MB

在这个测试中，代码同测试 2 一样，但数据库数据文件被设定为 autogrow10MB。

结果总结：

循环	步骤	CPU	读	写	持续时间 (ms)
1	文件增长	8907	283930	33588	41354

	文件不增长	6297	225358	33565	37755
	% 改进	**29.30279555**	**20.62902828**	**0.068476837**	**8.702906611**
	差异	**2610**	**58572**	**23**	**3599**
2	文件增长	9078	283902	33588	46607
	文件不增长	5578	216971	33560	41202
	% 改进	**38.55474774**	**23.57538869**	**0.083363106**	**11.59697041**
	差异	**3500**	**66931**	**28**	**5405**
3	文件增长	9016	283909	33587	52515
	文件不增长	6015	225358	33565	38807
	% 改进	**33.28527063**	**20.62315742**	**0.065501533**	**26.10301819**
	差异	**3001**	**58551**	**22**	**13708**
	平均% 改进	**33.71427131**	**21.60919146**	**0.072447159**	**15.46763174**
	平均差异	**3037**	**61351.33333**	**24.33333333**	**7570.666667**

测试 4

初始文件大小 = 33664,

最终文件大小 = 66944,

大量事务 （33373 行），

文件增长 1 MB

在这个测试中，我再次将 ShrinkDB 表中的行插入到 ShrinkTable 中，是大量的事务（一次 33373 行）。文件增长了 1MB，我必须把 T-Log 增加到 200MB 才能在事务运行时不增长。我在测试大量事务如何影响文件增长。

代码如下：

```
--=====================================================
-- Truncate the table
truncate table ExpandDB
go
-- Shrink database as much as possible (to 2MB)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Fill up the table (this statement will not be compared)
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== First insert ==========
-- Insert all rows from ExpandDB table to ExpandDB again
insert into ExpandDB select * from ExpandDB
go
-- How many rows are now in the table?
select count(*) from ExpandDB
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Truncate the table again
truncate table ExpandDB
go
-- Fill up the table (this statement will not be compared)
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== Second insert ==========
-- Fill the table without growing the file
-- Insert all rows from ExpandDB table to ExpandDB again
insert into ExpandDB select * from ExpandDB
go
-- How many rows are now in the table?
select count(*) from ExpandDB
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
```

结果总结：

循环	步骤	CPU	读	写	持续时间 (ms)
1	文件增长	3969	880471	33381	36097
	文件不增长	3720	879426	33380	18104
	% 改进	6.273620559	0.118686476	0.002995716	49.84624761
	差异	249	1045	1	17993
2	文件增长	3750	880473	33381	35962
	文件不增长	3578	879435	33380	20006
	% 改进	4.586666667	0.117891179	0.002995716	44.36905623
	差异	172	1038	1	15956
3	文件增长	3657	880471	33381	36544
	文件不增长	3999	879422	33380	18802
	% 改进	-9.35192781	0.119140778	0.002995716	48.54969352
	差异	-342	1049	1	17742
	平均% 改进	0.502786472	0.118572811	0.002995716	47.58833245
	平均差异	26.33333333	1044	1	17230.33333

测试 5

初始文件大小= 33664,

最终文件大小 = 66944,

大量事务（33373 行），

文件增长 10 MB

在这个测试中，代码通测试 4 的相同，但是文件 autogrowth 被设定为 10MB。

结果总结：

循环	步骤	CPU	读	写	持续时间 (ms)
1	文件增长	4016	879535	33381	23003

	文件不增长	4016	879432	33380	21177
	% 改进	0	0.011710734	0.002995716	7.938095031
	差异	0	103	1	1826
2	文件增长	3672	879531	33381	22501
	文件不增长	3673	879439	33380	19965
	% 改进	-0.027233115	0.01046012	0.002995716	11.2706102
	差异	-1	92	1	2536
3	文件增长	3798	879544	33381	22366
	文件不增长	3782	879426	33380	18702
	% 改进	0.421274355	0.013416043	0.002995716	16.38200841
	差异	16	118	1	3664
	平均% 改进	0.13134708	0.011862299	0.002995716	11.86357121
	平均差异	5	104.3333333	1	2675.333333

(作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国)

SQL Server 数据增长检测（三）

测试 6

初始文件大小 = 256,

最终文件大小 = 128256,

少量事务（每次一行），

总行数：33373,

文件增长 1,000 MB

在这个测试中，我插入了 33373 行，一次让文件增长另一次不增长。

代码如下：

```
--============================================================
-- Truncate the table
truncate table ExpandDB
go
-- Shrink database as much as possible (2MB)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== First insert ==========
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Truncate the table
truncate table ExpandDB
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== Second insert ==========
-- Insert all rows from ExpandDB table again
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
```

结果总结：

循环	步骤	CPU	读	写	持续时间 (ms)
1	文件增长	3343	217006	33382	19335
	文件不增长	3500	216989	33383	18020
	% 改进	-4.696380497	0.007833885	-0.002995626	6.801137833

	差异	-157	17	-1	1315
2	文件增长	3344	217006	33385	30126
	文件不增长	3359	217001	33378	16499
	% 改进	-0.448564593	0.002304084	0.0209675	45.23335325
	差异	-15	5	7	13627
3	文件增长	3578	217006	33382	16884
	文件不增长	3234	216989	33382	21183
	% 改进	9.61430967	0.007833885	0	-25.46197584
	差异	344	17	0	-4299
	平均% 改进	1.489788193	0.005990618	0.005990625	8.857505083
	平均差异	57.33333333	13	2	3547.666667

测试 7

初始文件大小= 34816,

最终文件大小= 1314816,

大量事务（33373 行），

文件增长 1,000 MB

代码如下：

```
--=====================================================
-- Set autogrowth to 10MB
ALTER DATABASE [ShrinkDB] MODIFY FILE ( NAME = N'ShrinkDB', FILEGROWTH
= 10240KB )
go
-- Truncate the table
truncate table ExpandDB
go
-- Shrink database as much as possible (to 256)
DBCC SHRINKFILE (N'ShrinkDB' , 0, TRUNCATEONLY)
Go
-- Fill up the table (this statement will not be compared)
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Set autogrowth to 10000MB
ALTER DATABASE [ShrinkDB] MODIFY FILE ( NAME = N'ShrinkDB', FILEGROWTH
= 10240000KB )
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== First insert ==========
-- Insert all rows from ExpandDB table to ExpandDB again
insert into ExpandDB select * from ExpandDB
go
-- How many rows are now in the table?
select count(*) from ExpandDB
go
-- What is the size of the data file?
select size from sysfiles where fileid = 1
go
-- Truncate the table again
truncate table ExpandDB
go
-- Fill up the table (this statement will not be compared)
declare @i int
set @i = 1
while @i <= 33373
begin
      insert into ExpandDB select replicate ('a',8000)
      set @i = @i + 1
end
go
-- Clear transaction log (so it will not grow)
backup transaction ShrinkDB with truncate_only
go
-- Clear cache
DBCC DROPCLEANBUFFERS
go
--=== Second insert ==========
-- Fill the table without growing the file
-- Insert all rows from ExpandDB table to ExpandDB again
insert into ExpandDB select * from ExpandDB
go
-- How many rows are now in the table?
select count(*) from ExpandDB
go
```

结果总结：

循环	步骤	CPU	读	写	持续时间 (ms)
1	文件增长	3548	867705	45151	24096
	文件不增长	3250	878488	34317	20707
	% 改进	8.399098083	-1.242703453	23.99503887	14.06457503
	差异	298	-10783	10834	3389
2	文件增长	3579	879331	33542	21753
	文件不增长	3765	872841	39965	22775
	% 改进	-5.196982397	0.738061094	-19.14912647	-4.698202547
	差异	-186	6490	-6423	-1022
3	文件增长	3781	879411	33452	25014
	文件不增长	3422	870858	41918	21235
	% 改进	9.494842634	0.972582786	-25.30790386	15.10753978
	差异	359	8553	-8466	3779
	平均% 改进	4.23231944	0.155980142	6.82066382	8.157970755
	平均差异	157	1420	1351.66667	2048.666667

结论：

平均值分析：

测试 1—少量事务，小 autogrowth

	CPU	读	写	持续时间 (ms)
平均% 改进	45.52345285	20.74529261	0.125700284	39.5361444
平均差异	895.6666667	14491	10.33333333	5125.666667

测试 2——少量事务，小 autogrowth，多行

	CPU	读	写	持续时间 (ms)
平均% 改进	39.16279497	20.96413262	0.158643548	38.31454107
平均差异	2638.666667	59797.33333	53.33333333	20901.66667

测试 3——少量事务，大 autogrowth，行数同测试 2

	CPU	读	写	持续时间 (ms)
平均% 改进	33.71427131	21.60919146	0.072447159	15.46763174
平均差异	3037	61351.33333	24.33333333	7570.666667

测试 4——大量事务，小 autogrowth，行数同测试 2

	CPU	读	写	持续时间 (ms)
平均% 改进	0.502786472	0.118572811	0.002995716	47.58833245
平均差异	26.33333333	1044	1	17230.33333

测试 5——大量事务，大 autogrowth，行数同测试 2

	CPU	读	写	持续时间 (ms)
平均% 改进	0.13134708	0.011862299	0.002995716	11.86357121
平均差异	5	104.3333333	1	2675.333333

测试 6——少量事务，大 autogrowth，行数同测试 2

	CPU	读	写	持续时间 (ms)
平均% 改进	1.489788193	0.005990618	0.005990625	8.857505083
平均差异	57.33333333	13	2	3547.666667

测试 7——大量事务，大 autogrowth，行数同测试 2

	CPU	读	写	持续时间 (ms)
平均% 改进	4.23231944	0.155980142	-6.82066382	8.157970755
平均差异	157	1420	-1351.66667	2048.666667

结果显示，性能下降主要是由于文件增长，特别是在少量事务和小 autogrowth 情况下。同样，autogrowth 越大性能就越好。事务越多，性能受 autogrowth 的影响也就越少。所以，当你要缩减数据库文件，你应该问一下自己是否值得，这意味着文件在一段时间内不会再次自动增长，此外，你还要根据预期的数据库活动和增长来设定一个 autogrowth 门槛。

无论一个事物有无 autogrowth，CPU 以及读写速度的差距并不大，但事务的持续时间仍然受到文件增长的影响。

（作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国）

在 SQL Server 上测试事务日志的自动增长（一）

比起数据库文件，事务日志的增长速度毫不逊色，因此数据库管理员会经常缩减删除事务日志。在本文中，我将对事务日志文件增长造成的影响做一个测试。而在本次测试中，我将使用和上一次测试相同的环境和配置。唯一的不同是，这一次我在 SQL Server 上安装了 SP3。

数据库

本次测试数据库 ShrinkDB 设置为在执行事务时，数据库文件不增长，事务日志文件大小只有 2MB，而本次测试的目的就是让它在每次事务中都增长以衡量其影响：

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
ShrinkDB	Rows Data	PRIMARY	200000	None
ShrinkDB_log	Log	Not Applicable	2	By 1 MB, restricted growth to 2097152 MB

ShrinkDB 数据库的还原模型被设置为 FULL。

```
select size from sysfiles where fileid = 2
```

在以上的查询语句中，fileid 代表事务日志文件，返回值 256。这表明，事务日志文件大小为 2MB。

同上一篇文章一样，ExpandDB 表再次使用。

```
create table ExpandDB (a varchar(8000))
```

本次测试的目标是了解 INSERT, UPDATE 和 DELETE 命令是如何影响事务日志文件的增长的，还有事务日志文件增长反过来如何影响 INSERT, UPDATE 和 DELETE 日志的。

我把测试分成两个阶段：

在第一阶段，我将监测由于运行修改造成事务文件增长的一般行为。

在第二阶段，我将用那些在第一阶段造成事务日志增长的命令。在这个阶段，我将测试同第一部分类似的 autogrowth 情况。

由于更新，插入和删除都有所不同，在每个阶段都要包含三个部分，每次一个操作：UPDATES, INSERTS 和 DELETE。

第一阶段：事务日志增长行为

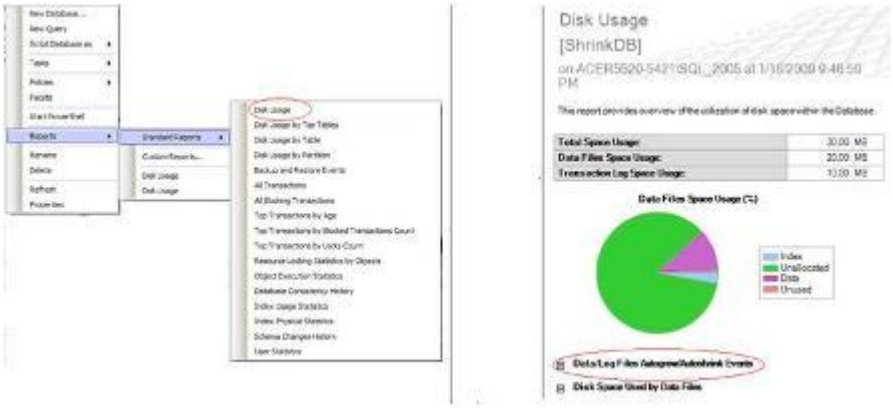
UPDATE 命令和事务日志增长

在这个测试中，我的代码将只向表中插入一行，知道事务日志文件增长到 8MB 后，它将更新此表。

```
-- Insert one row into the table
insert into ExpandDB select replicate ( 'a',8000)
--================================================================
-- Update the table until T-Log reaches 8MB
while (select size from sysfiles where fileid = 2) <= 1024
update ExpandDB set a = replicate ( 'a',8000)
go
```

结果令人惊讶！

我的循环无限地运行，事务日志文件没有增长而且在使用 dbcc sqlperf（日志空间）的空闲空间百分比一直保持 30%的增长。事务运行期间，事务日志将持续填充直至 70%，然后下降到 50%。为确保从 dbcc sqlperf 得到准确的数据，我同时在数据文件中运行查找 autogrowth 的报告：



此时报告在任何数据库中都没有 autogrowth!

然后我决定检查有少量更新的大量事务，一次一行，所以我加了一个开始事务：

```
-- Big transaction
begin tran
while (select size from sysfiles where fileid = 2) <= 1024
update ExpandDB set a = replicate ('a',8000)
go
commit tran
```

这一次，结果同样令人困惑。循环无限地运行，事务日志文件没有超过初始的 2MB 而且空闲空间百分比一直保持 60%的增长。我停止了循环并提交了事务。提交之后，事务日志文件的使用空间甚至少了一点。

在另一个 UPDATE 命令测试中，我向表中插入了 10，000 并运行了以下代码：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Check the size of the T-Log file (RESULT = 2MB)
select size from sysfiles where fileid = 2
go
-- Check % free space in T-Log
```

```
dbcc sqlperf(logspace)
go
-- Update 10000 rows at a time
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
update ExpandDB set a = a + 'a'
go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
```

结果类似。事务日志文件增长了 1MB，直到达到 10MB 大小，之后就不再增长了。无论我执行几次更新动作，空间填满之后又再次下降：

Database Name	Log Size (MB)	Log Space Used (%)	Status
ShrinkDB	9.992188	40.39777	0
ShrinkDB	9.992188	62.96423	0
ShrinkDB	9.992188	37.40227	0

我为此寻求解答，但在 SQL Server 2005 说明书中没找到答案。我还在其它机器上的 SQL Server 2005 中运行了同样的测试，但结果都一样。

(作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国)

在 SQL Server 上测试事务日志的自动增长（二）

我用 ApexSQL Log 工具分析了事务日志内容，结果显示，每次我看事务日志内容都会得到不同的更新数，这同我在事务日志大小上遇到的问题一样;即使不断增长与减少，但总大小依旧保持不变。

为了解事务日志的行为，我运行了一个事务，UPDATE 命令修改了行值：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShribkDB_Log', 0, TRUNCATEONLY)
Go
-- Insert one row into the table
insert into ExpandDB select replicate ('a',3)
--=============================================================
-- Update the table until T-Log reaches 8MB
-- Big transaction
begin tran
while (select size from sysfiles where fileid = 2) <= 1024
update ExpandDB set a = case len(a)
when 7999 then a else a + 'b' end
commit tran
go
-- Check status of T-log
dbcc sqlperf(logspace)
go
```

在这一情况下，事务日志增长到 4MB，使用空间在很长一段时间内保持在 94%。然而我没有收到一个错误指示字符串截断或溢出。当事务日志停止增长时，被 ApexSQL Log 工具监视的事务日志内容始终停留在大约 13，380 行。此外，SQL Profiler 监视语句表明在

事务日志停止增长之后没执行过任何命令，没出过任何错。我取消事务后，在错误日志中没有出现任何特殊信息表明 autogrowth 操作被取消或中止，事务似乎被卡住了。

在类似实例中，我在错误日志中有时会看到下面的信息：

"Autogrow of file '

但无论我执行多少次上面的代码，它都不会出现在测试中。

取消事务之后，事务日志中的使用空间减少到大约 45%。

在大量行数情况下，我决定运行以下 UPDATE 命令：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Check the size of the T-Log file (RESULT = 2MB)
select size from sysfiles where fileid = 2
go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Update 10000 rows at a time
update ExpandDB set a case len(a)
when 7999 then a else a 'b' end
go
update ExpandDB set a case len(a)
when 7999 then a else a 'b' end
go
update ExpandDB set a case len(a)
when 7999 then a else a 'b' end
go
update ExpandDB seta caselen(a)
when 7999 then a else a 'b' end
go
update ExpandDB set a case len(a)
when 7999 then a else a 'b' end
go
update ExpandDB set a case len(a)
```

```
when 7999 then a else a 'b' end
go
update ExpandDB set a case len(a)
when 7999 then a else a 'b' end
go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
```

结果表明，如果我一个接一个的运行升级，每次等待一到两秒，事务日志不会增长，空间像原来一样填满清空。然而当同时运行两个更新，在第二个事务运行之前空间不会填满或清空。这一情况下，事务日志将增长。

INSERT 命令和事务日志增长

在这个测试中，我的目标是使事务日志增长到 8MB，然后将数据文件增长比同事务日志文件增长比做一个比较。数据文件设定为没有 autogrowth，大小为 200G；事务日志文件设定为 2MB 大小，没有 autogrowth，然后运行以下代码：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log', 0, TRUNCATEONLY)
Go
-- Check the size of the T-Log file
select size from sysfiles where fileid go
-- Insert narrow rows into the table

while (select size from sysfiles where fileid = 2) insert into ExpandDB select
replicate ('a',3)
go
At the same time, I also executed the following code:
select getdatedbcc sqlperf (logspace)
go
```

```
waitfor delay '00:00:02'
go
select getdatedbcc sqlperf (logspace)
go
select getdatego
dbcc sqlperf (logspace)
go
waitfor delay '00:00:02'
Go
```

我的循环运行了很长时间，数据文件慢慢填充并清空，但事务日志文件始终保持在2MB。以上代码得到如下结果：

Results Messages

	(No column name)
1	2009-01-26 23:13:51.233

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	0.4921875	69.84127	0
2	tempdb	0.7421875	54.47368	0
3	model	0.4921875	50	0
4	msdb	1.992188	36.07843	0
5	ShrinkDB	1.992188	56.4951	0

	(No column name)
1	2009-01-26 23:13:53.420

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	0.4921875	69.84127	0
2	tempdb	0.7421875	54.47368	0
3	model	0.4921875	50	0
4	msdb	1.992188	36.07843	0
5	ShrinkDB	1.992188	39.33823	0

	(No column name)
1	2009-01-26 23:13:53.650

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	0.4921875	69.84127	0
2	tempdb	0.7421875	54.47368	0
3	model	0.4921875	50	0
4	msdb	1.992188	36.07843	0
5	ShrinkDB	1.992188	43.06372	0

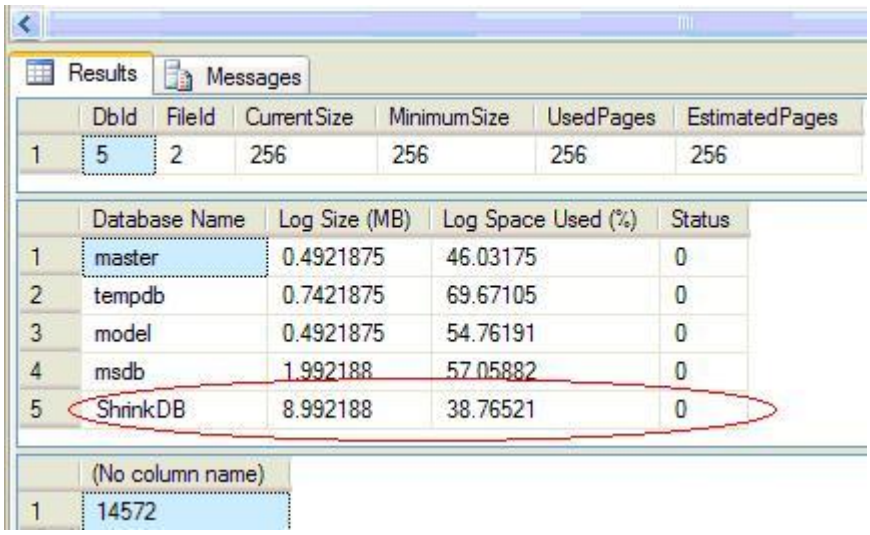
显然地，空闲空间减少了。在向测试插入 45 分钟之前，我的表中有一百八十万行，事务日志仍然是 2MB。在取消之后，事务日志还是 2MB 而且只有 39.88%的空间被占用。数据文件 31%被占用。

然后在每次插入一行的大量事务中测试：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log', 0, TRUNCATEONLY)
Go
-- Insert 100000 rows
-- Big transaction
begin tran

while (select size from sysfiles where fileid = 2) insert into ExpandDB select
replicate ('a',3)
go
commit tran
go
dbcc sqlperf(logspace)
go
select count(*) from ExpandDB
go
```

事务日志增长到 9MB，最后表中有 14，572 行：



Results | Messages

	DbId	FileId	CurrentSize	MinimumSize	UsedPages	EstimatedPages
1	5	2	256	256	256	256

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	0.4921875	46.03175	0
2	tempdb	0.7421875	69.67105	0
3	model	0.4921875	54.76191	0
4	msdb	1.992188	57.05882	0
5	ShrinkDB	8.992188	38.76521	0

	(No column name)
1	14572

为测试大量更新，我向表中插入 10，000 行并运行以下代码：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Check the size of the T-Log file (RESULT = 2MB)
select size from sysfiles where fileid = 2 go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Insert 10000 rows at a time
Insert into ExpandDB select * from ExpandDB
Go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Insert 10000 rows at a time
Insert into ExpandDB select * from ExpandDB Go --
Check % free space in T-Log
dbcc sqlperf(logspace)
go
-- Insert 10000 rows at a time
Insert into ExpandDB select * from ExpandDB
Go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
```

```
go
-- Insert 10000 rows at a time
Insert into ExpandDB select * from ExpandDB
Go
-- Check % free space in T-Log
dbcc sqlperf(logspace)
go
```

这次，每次插入后，事务日志都加倍了。

(作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国)

在 SQL Server 上测试事务日志的自动增长（三）

DELETE 命令和事务日志增长

这次的目标和前面一样。我向表中插入 10，000 行并运行以下代码来一次删除一行：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log', 0, TRUNCATEONLY)
Go
-- Insert 100000 rows
declare @i int
set @i = 1
while @i <= 10000
begin
insert into ExpandDB select replicate ('a',1000)
set @i = @i + 1
end
go
-- Delete rows one by one:
set rowcount 1

while (select size from sysfiles where fileid = 2) delete from ExpandDB
set rowcount 0
go
```

出现了同样的情况：事务日志并没增长，空间循环地填充和清空。

然后我尝试了大量的删除：

```
-- Truncate the table
truncate table ExpandDB
```

```
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log', 0, TRUNCATEONLY)
Go
-- Insert 100000 rows
declare @i int
set @i = 1
while @i <= 10000
begin
insert into ExpandDB select replicate ('a',1000)
set @i = @i + 1
end
go
-- Delete 10,000 at a time and monitor the T-Log size and free space:
begin tran
set rowcount 10000
while begin
delete from ExpandDB
dbcc sqlperf(logspace)
end set rowcount 0 commit tran go
```

结果表明事务日志增长：

Database Name	Log Size (MB)	Log Space Used (%)	Status
ShrinkDB	29.99219	54.32241	0
ShrinkDB	32.99219	51.68571	0
ShrinkDB	34.99219	50.18001	0
ShrinkDB	37.99219	48.60683	0
ShrinkDB	39.99219	46.85608	0
ShrinkDB	40.99219	66.63094	0
ShrinkDB	40.99219	58.30355	0
ShrinkDB	41.99219	53.7686	0
ShrinkDB	44.99219	70.38548	0

最后我运行了一个有少量操作的大量事务：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE (N'ShrinkDB_Log' , 0, TRUNCATEONLY)
Go
-- Insert 100000 rows
declare @i int
set @i = 1
while @i <= 10000
begin
insert into ExpandDB select replicate ('a',1000)
set @i = @i + 1
end
go
-- Delete 10,000 at a time and monitor the T-Log size and free space:
begin tran
set rowcount 1
while from ExpandDB) begin delete from ExpandDB end
```

结果显示事务日志增长了，尽管很慢。

这一切意味着什么

当执行少量分散的 INSERT, UPDATE 和 DELETE 命令时，事务日志并不会增长太多。但当我浏览事务日志文件时，这一测试生成了奇怪的动作：当达到大约 70%时，文件填充清空但不增长。

当大量操作被执行时，包括事务中的大量数据和事务日志都增长，但并不是预期的线性增长。似乎有些数据在事务中被压缩了。

(作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国)

SQL Server 中事务日志自动增长对性能的影响（上）

在本文中，我将对事务运行过程中事务日志增长对性能的影响做测试。我这次的测试环境和配置同前两次的相同。

数据库

本次测试中使用的 ShrinkDB 数据库要预先配置好，保证数据库足够大并在事务运行中不会增长。事务日志文件大小只有 2MB。测试的目的就是让它随着事务进行而增长，从而衡量它的影响：

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth
ShrinkDB	Rows Data	PRIMARY	200000	None
ShrinkDB_log	Log	Not Applicable	2	By 1 MB, restricted growth to 2097152 MB

ShrinkDB 数据库的还原模型设置为 FULL。

使用以下查询语句（fileid = 2 → the T-Log file）：

```
select size from sysfiles where fileid = 2
```

这是事务日志文件的大小（每页 8KB）。256*8=2MB。

原先测试用过的 ExpandDB 表在本次测试用依然使用。

```
create table ExpandDB （a varchar（8000））
```

测试

测试的目标就是检验让数据库事务日志自动增长的 INSERT, UPDATE 和 DELETE 命令如何影响性能。

由于升级，插入和删除可能会相互影响，所以这个测试包含三个部分，每次一个操作：

UPDATES

INSERTS

DELETE

注意：在原先的文章中，我证明了只有大量的事务才能使事务日志自动增长。

插入命令

在本次测试中，一次事务我都向表中插入了 10，000 行。每一步都执行相同的代码，但事务日志的 autogrowth 设定不同：

在第一步中， autogrowth 设置为 1MB;

在第二步中， autogrowth 设置为 10MB。

这两个步骤总共执行三次，SQL Profiler 来捕获执行统计。以下为代码：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB （initial size）:
DBCCSHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY
Go
-- Insert 10000 rows
-- Big transaction
begin tran
declare @i int
set @i = 1
while @i <= 10000
begin insert into ExpandDB select replicate （'a',8000）
set @i = @i + 1
```

```
end
commit
Go
-- Check size and % free space in T-Log
dbcc sqlperf (logspace
go
select count (*)  from ExpandDB
go
```

在第一步中，事务日志达到 110MB。在第二步中达到了 102MB。

以下是结果比较（只有插入）：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	985	65152	10115	30479
	File grows by 10MB	875	65111	10113	15416
	% improvement	11.16751269	0.062929764	1.008403361	49.42091276
	Difference	110	41	102	15063
2	File grows by 1MB	828	65155	10115	32109
	File grows by 10MB	875	65104	10118	13240
	% improvement	-5.676328502	0.078274883	0.958971824	58.76545517
	Difference	-47	51	97	18869
3	File grows by 1MB	969	65218	10118	37298
	File grows by 10MB	1062	65104	10012	15773
	% improvement	-9.59752322	0.174798369	1.047637873	57.71086922
	Difference	-93	114	106	21525
	Average % improvement	-1.36877968	0.105334339	1.005004353	55.29907905
	Average difference	-10	68.66666667	101.6666667	18485.66667

我又进行了第三次测试，事务日志文件设定为 130MB 并在事务运行期间不增长。这次我没有缩减事务日志，代码如下：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Insert 10000 rows
-- Big transaction
begin tran
```

```
declare @i int
set @i = 1
while @i <= 10000
begin
insert into ExpandDB select replicate ('a',8000)
set @i = @i + 1
end
commit
Go
-- Check size and % free space in T-Log
dbcc sqlperf (logspace)
go
select count (*)
from ExpandDB
go
```

现在我们有了上一次执行和第一次执行的比较，只有插入命令：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	985	65152	10115	30479
	File did not grow	812	65097	10003	5651
	% improvement	17.56345178	0.084417976	1.107266436	81.45936546
	Difference	173	55	112	24828
2	File grows by 1MB	828	65155	10115	32109
	File did not grow	719	65093	10007	8160
	% improvement	13.16425121	0.095157701	1.067721206	74.58656451
	Difference	109	62	108	23949
3	File grows by 1MB	969	65218	10118	37298
	File did not grow	985	65103	10003	6741
	% improvement	-1.651186791	0.176331688	1.136588259	81.926644862
	Difference	-16	115	115	30557
	Average % improvement	9.692172065	0.118635788	1.103858634	79.32419161
	Average difference	88.66666667	77.33333333	111.6666667	26444.66667

插入总结：

平均改进数据显示性能的增强和事务运行中事务日志增长数目有关。自动增长越少性能越佳，特别是对于整段时间而言（无增长比 1MB 增长性能改进了 80%）。

升级命令

在上一篇文章中显示，当每次都有很多行改变时，升级会令事务日志增长。为此，我执行了以下代码：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY）
Go
-- Update 10000 rows at a time
update ExpandDB set a = 'aaaaaaaaaaaaaaaaaaaaaaaa'
go
-- Check size and % free space in T-Log
dbcc sqlperf（logspace）
go
```

我运行了两次以确保事务日志按相同比例增长，令人困惑的是第一次运行中时事务日志增长了，而第二次一点没有增长（停留在 2MB）！

这可能意味着行值不变的话，事务日志也就不增长。为证明，我加了一个 Select 语句来输出被升级影响的行数：

```
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY）
Go
-- Update 10000 rows at a time
update ExpandDB set a = 'aaaaaaaaaaaaaaaaaaaaaaaa'
select @@rowcount go
-- Check size and % free space in T-Log
dbcc sqlperf（logspace）
go
```

返回的行值一直是 10000.

当我修改了升级值（update ExpandDB set a = 'abcde'），

```
-- #1 ITERATION:
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go -- Shrink T-Log back to 2MB:
DBCC SHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY）
Go
-- Update 10000 rows at a time
update ExpandDB set a = 'aaaaaaaaaaaaaaaaaaaabc'
go
-- Check size and % free space in T-Log
dbcc sqlperf（logspace）
go
-- #2 ITERATION:
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go -- Shrink T-Log back to 2MB:
DBCC SHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY）
Go
-- Update 10000 rows at a time
update ExpandDB set a = 'AAA'
go
-- Check size and % free space in T-Log
dbcc sqlperf（logspace）
go
-- #3 ITERATION:
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go -- Shrink T-Log back to 2MB:
DBCC SHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY）
Go
-- Update 10000 rows at a time
update ExpandDB set a = 'zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz'
go
-- Check size and % free space in T-Log
```

```
dbcc sqlperf（logspace）
go
```

不出所料，当事务日志设置为每次增长 1MB 时，它最后的大小是不同的。

以下是结果的对比：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	203	8842	50	271
	File did not grow	187	8840	94	192
	% improvement	7.881773399	0.022619317	-88	29.15129151
	Difference	16	2	-44	79
2	File grows by 1MB	156	13125	47	203
	File did not grow	157	13125	3	267
	% improvement	-0.641025641	0	93.61702128	-31.5270936
	Difference	-1	0	44	-64
3	File grows by 1MB	781	17919	100	827
	File did not grow	750	17918	31	804
	% improvement	3.969270166	0.005580669	69	2.781136638
	Difference	31	1	69	23
	Average % improvement	3.736672641	0.009399995	24.87234043	0.135111518
	Average difference	15.33333333	1	23	12.66666667

（作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国）

SQL Server 中事务日志自动增长对性能的影响（下）

升级总结：

看上去当事务日志不增长的时候，性能会得到改善，特别是写入时。

删除命令

这个测试的目标同删除一样，也有更新与插入。

我向表中插入 10，000 行然后运行以下代码来删除所有行：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY）
Go
-- Delete 10,000:
declare @i int
set @i = 1
while @i <= 10000
begin
insert into ExpandDB select replicate （'a',1000）
set @i = @i + 1
end
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY）
go
-- Delete 10,000:
```

```
delete from ExpandDB
-- Check size and % free space in T-Log
dbcc sqlperf（logspace）
go
```

在第一步中，事务日志被缩减，autogrowth 被设定为 1MB。在第二步中，事务日志没有缩减丙炔他设定的足够大以至于不会增长。

下面是结果对比：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	78	7224	1468	7890
	File did not grow	79	7196	1440	495
	% improvement	-1.282051282	0.387596899	1.907356948	93.72623574
	Difference	-1	28	28	7395
2	File grows by 1MB	93	7220	1468	8372
	File did not grow	62	7194	1441	435
	% improvement	33.33333333	0.360110803	1.839237057	94.80410893
	Difference	31	26	27	7937
3	File grows by 1MB	46	7220	1468	6657
	File did not grow	94	7192	1440	355
	% improvement	-104.3478261	0.387811634	1.907356948	94.66726754
	Difference	-48	28	28	6302
	Average % improvement	-24.098848	0.378506446	1.884650318	94.39920407
	Average difference	-6	27.33333333	27.66666667	7211.333333

删除总结：

再次证明，事务日志不增长，性能就会越好，特别是在整个期限里。CPU 差异呈现锯齿状情况（可能因为数量太少所以不精准）。

事务中的更多行

为测试大量事务的相关性能（事务日志自动增长），我重复了上面的测试，将行数改为 50，000 行（之前是 10，000 行），测试结果如下所示：

插入：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	2219	103797	7249	26237
	File did not grow	2110	103713	7152	5666
	% improvement	4.912122578	0.080927194	1.338115602	78.4045432
	Difference	109	84	97	20571
2	File grows by 1MB	2250	103797	7250	24083
	File did not grow	2063	103704	7157	3528
	% improvement	8.311111111	0.089597965	1.282758621	85.35066229
	Difference	187	93	93	20555
3	File grows by 1MB	2234	103804	7250	24880
	File did not grow	2078	103711	7157	6010
	% improvement	0.089591923	0.387811634	1.282758621	75.84405145
	Difference	156	93	93	18870
	Average % improvement	6.735407947	0.086705694	1.301210948	79.86641898
	Average difference	150.6666667	90	94.33333333	19998.66667

更新：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	313	50055	50023	4868
	File did not grow	250	50032	50000	386
	% improvement	20.12779553	0.045949456	0.04597885	92.07066557
	Difference	63	23	23	4482
2	File grows by 1MB	218	50048	50020	5273
	File did not grow	265	50028	50000	254
	% improvement	-21.55963303	0.039961637	0.039984006	95.18300778
	Difference	-47	20	20	5019
3	File grows by 1MB	297	50048	50020	4418
	File did not grow	266	50028	50000	282
	% improvement	10.43771044	0.039961637	0.039984006	93.61702128
	Difference	31	20	20	4136
	Average % improvement	3.001957646	0.041957576	0.041982288	93.62356487
	Average difference	15.66666667	21	21	4545.666667

删除：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	375	43054	7299	35894
	File did not grow	406	42911	7155	3138
	% improvement	-8.266666667	0.332141032	1.972872996	91.2575918
	Difference	-31	143	144	32756
2	File grows by 1MB	343	43053	7293	31672
	File did not grow	312	42911	7155	4933
	% improvement	9.037900875	0.329826028	1.892225422	84.42472847
	Difference	31	142	138	26739
3	File grows by 1MB	359	43057	7269	33549
	File did not grow	297	42907	7155	2990
	% improvement	17.27019499	0.348375409	1.568303756	91.08766282
	Difference	62	150	114	30559
	Average % improvement	6.013809731	0.336780823	1.811134058	88.9233277
	Average difference	20.66666667	145	132	30018

看上去，在事务日志不增长的情况下，事务越多性能改进越大。

更大的 autogrowth

为测试更大 autogrowth 的相关性能，我将事务日志的 autogrowth 设置为 50MB，然后连续三次运行以下代码（包括插入删除）：

```
-- Truncate the table
truncate table ExpandDB
go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
go
-- Shrink T-Log back to 2MB:
DBCC SHRINKFILE （N'ShrinkDB_Log', 0, TRUNCATEONLY）
Go
-- Insert 50,000 rows
begin tran
declare @i int
set @i = 1
```

```
while @i <= 50000
begin insert into ExpandDB select replicate ('a',1000)
set @i = @i + 1
end
commit
Go
-- Truncate the T-Log
backup transaction ShrinkDB with truncate_only
Go
-- Shrink T-
Log back to 2MB: DBCC SHRINKFILE (N'ShrinkDB_Log', 0, TRUNCATEONLY)
Go
-- Delete 50,000: delete from ExpandDB
-- Check size and % free space in T-Log
dbcc sqlperf (logspace)
Go
```

然后我对比了前两次的结果（50，000 行、不同的 autogrowth）：

插入：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	2219	103797	7249	26237
	File grows by 50MB	2281	103713	7158	8933
	File did not grow	2110	103713	7152	5666
	% improvement 50MB to no growth	7.496711968	0	0.083822297	36.57226016
	Difference	171	0	6	3267
2	File grows by 1MB	2250	103797	7250	24083
	File grows by 50MB	2203	103713	7158	7940
	File did not grow	2063	103704	7157	3528
	% improvement 50MB to no growth	6.354970495	0.008677794	0.013970383	55.56675063
	Difference	140	9	1	4412
3	File grows by 1MB	2234	103804	7250	24880
	File grows by 50MB	2312	103713	7159	12823
	File did not grow	2078	103711	7157	6010
	% improvement 50MB to no growth	10.12110727	0.001928399	0.027936863	53.13109257
	Difference	234	2	2	6813
	Average % improvement	7.99092991	0.003535397	0.041909847	48.42336779
	Average difference	181.6666667	3.666666667	3	4830.666667

删除：

ITERATION	Step	CPU	Reads	Writes	Duration (ms)
1	File grows by 1MB	375	43054	7299	35894
	File grows by 50MB	375	42914	7158	10920
	File did not grow	406	42911	7155	3138
	% improvement 50MB to no growth	-8.266666667	0.006990726	0.041911148	71.26373626
	Difference	-31	3	3	7782
2	File grows by 1MB	343	43053	7293	31672
	File grows by 50MB	407	42910	7152	10854
	File did not grow	312	42911	7155	4933
	% improvement 50MB to no growth	23.34152334	-0.002330459	-0.041946309	54.55131749
	Difference	95	-1	-3	5921
3	File grows by 1MB	359	43057	7269	33549
	File grows by 50MB	391	42910	7126	11133
	File did not grow	297	42907	7155	2990
	% improvement 50MB to no growth	24.04092072	0.006991377	-0.406960427	73.142908477
	Difference	94	3	-29	8143
	Average % improvement	13.03859246	0.003883881	-0.1356652	66.31932074
	Average difference	52.66666667	1.666666667	-9.66666667	7282

结果显示事务日志增长越少，性能越佳。

测试结果总结——第二阶段

根据测试结果，我可以证明当事务日志在运行中增长时，它会对性能产生负面影响：

1、事务越多，性能受影响越大;

2、更新的行数越多，声场的 T-Log 越大;

3、Autogrowth 越小，性能的影响越大。

注意我的测试只包含一个单独运行的事务。那么在多用户环境中，事务日志增长之后会发生什么？我将在今后的测试中继续研究。

另注：

同样的问题，测试证明了在一些数据库中，事务日志里的大量虚拟日志文件影响了数据修改的整体性能。这样的话，建议主动增长事务日志，不要让它自动增长。对于另外一些数据库，建议使用大的 autogrowth（指大小而不是百分比）。

(作者：Michelle Gutzait 译者：孙瑞 来源：TT 中国)