

Microsoft SQL Server 2005 性能调优

SQL Server 2005 性能调优

- SQL Server 应用程序性能调优
 - 压力测试
 - 系统及数据库性能监控
 - 优化软硬件配置

压力测试

- 压力测试工具——脚本录制
 - LoadRunner
 - VS.Net ATC(Application Center Test)
- 模拟实际生产环境
 - 并发数量
 - 思考时间
 - 参数化

压力测试工具

- LoadRunner
 - Virtual User Generator
 - 录制应用脚本
 - 架构
 - 协议
 - Transaction
 - 参数化
 - Controller
 - 模拟并发操作
 - 设置think time
 - Scheduler
 - 监控服务器性能参数
 - Analysis
 - 分析测试结果
 - 表格方式
 - 图表方式

系统及数据库性能监控

- Windows 性能监视器
 - 添加性能参数
 - 设置监控计划
- SQL Server Profiler——跟踪 SQL Server 事件
 - 查询计划
 - T-SQL
 - 存储过程
 - 表锁
 - 游标
- DMV (动态管理视图)
 - 遵循 dm_* 命名约定
 - 使用 sys 架构作为视图或函数名称的前缀
 - SELECT wait_type, wait_time_ms
 - FROM sys.dm_os_wait_stats
 - GO

降低SQL Server速度的原因

- 资源瓶颈：
 - CPU, 内存, 和I/O瓶颈
- Tempdb 瓶颈
 - 每个SQL Server 实例只有一个tempdb
 - 性能和磁盘空间的瓶颈
 - 不好的应用程序
 - 过多的DDL和DML操作
- 缓慢运行的用户查询：
 - 改变统计信息可以导致现有查询的较差的查询计划
 - 全表扫描，降低查询性能
 - 即使资源利用正常由于阻塞也可以导致应用程序运行缓慢
 - 不良的应用程序设计或架构设计
 - 错误的事务隔离级别

资源瓶颈

- CPU 瓶颈
 - 过多的编译和重编译
 - 效率低的查询计划
 - 内部查询的并行
 - 拙劣游标使用
- 内存瓶颈
 - 外部物理内存压力
 - 外部虚拟内存压力
 - 内部物理内存压力
- IO瓶颈

CPU瓶颈

- 过多的编译和重编译
 - SQL Server: **SQL Statistics: Batch Requests/sec**
 - SQL Server: **SQL Statistics: SQL Compilations/sec**
 - SQL Server: **SQL Statistics: SQL Recompilations/sec**
 - 被重编译次数最多的25个存储过程
 - select top 25
 - sql_text.text,
 - sql_handle,
 - plan_generation_num,
 - execution_count,
 - dbid,
 - objectid
 - from
 - sys.dm_exec_query_stats a
 - cross apply sys.dm_exec_sql_text(sql_handle) as sql_text
 - where
 - plan_generation_num >1
 - order by plan_generation_num desc

CPU瓶颈

- 如何确定存储过程的过度重编译
 - 确定过度重编译的存储过程
 - 确定导致重编译的原因
- 如何确定
 - 使用**SQL Server Profiler**监控事件:
SP:Starting,SP:Completed,SP:stmtStarting,
SP:StmtCompleted
 - 使用**SQL Profiler**抓取信息
EventSubClass,TextData

存储过程的重编译原因代码

SubEvent	原因
1	架构、绑定或权限发生了更改
2	统计数据发生了更改
3	引用对象丢失
4	设置选项发生了更改
5	临时表的架构，绑定或权限发生了更改
6	远程行集的架构，绑定或权限发生了更改

统计数据变化的界定

表类型	空表条件	空表的界定	非空表的条件	非空表的界定
固定表	<500行	≥ 500 行	≥ 500 行	20%数据集行数
临时表	<6行	≥ 6 行	≥ 500 行	20%数据集行数
表变量	无	无	无	行

如何减少过度重编译

- 避免在操作中修改选项（SET OPTION）
- 存储过程中需要使用临时表的，相关的架构语句在存储过程的开始部分完成
- 有大量行变动的临时表可是考虑使用表变量替代
- 对于临时表的SELECT加上KEEP PLAN选项可避免6行的界定（500行）
- KEEP FIXEDPLAN可以避免统计数据对重编译的影响，但不能解决架构变动对重编译的影响

如何减少过度重编译

- SQL 2000:
 - 不支持存储过程中语句级的重编译（SQL 2005 支持）
 - 大的存储过程可以使用**sp_executesql**分拆
 - 将导致重编译的语句移出大的存储过程，建立独立的存储过程
- 使用**sp_autostats**关闭自动更新统计可以避免统计数据对重编译的影响

CPU瓶颈

- 效率低的查询计划
 - 前50个累计使用**CPU**时间最多的 查询
 - select
 - highest_cpu_queries.plan_handle,
 - highest_cpu_queries.total_worker_time,
 - q.dbid,
 - q.objectid,
 - q.number,
 - q.encrypted,
 - q.[text]
 - from
 - (select top 50
 - qs.plan_handle,
 - qs.total_worker_time
 - from
 - sys.dm_exec_query_stats qs
 - order by qs.total_worker_time desc) as highest_cpu_queries
 - cross apply sys.dm_exec_sql_text(plan_handle) as q
 - order by highest_cpu_queries.total_worker_time desc

CPU瓶颈

- 内部查询的并行

- SQL Server: SQL Statistics – Batch Requests/sec 计数器
- select
 - p.* ,
 - q.* ,
 - cp.plan_handle
- from
 - sys.dm_exec_cached_plans cp
 - cross apply sys.dm_exec_query_plan(cp.plan_handle) p
 - cross apply sys.dm_exec_sql_text(cp.plan_handle) as q
- where
 - cp.cacheobjtype = 'Compiled Plan' and
 - p.query_plan.value('declare namespace
p="http://schemas.microsoft.com/sqlserver/2004/07/showplan";
max(/p:RelOp/@Parallel)', 'float') > 0

内部查询的并行

CPU时间大于持续时间：

select

```
qs.sql_handle,    qs.statement_start_offset,  
qs.statement_end_offset,   q.dbid,   q.objectid,   q.number,  
q.encrypted,   q.text
```

from

```
sys.dm_exec_query_stats qs
```

```
cross apply sys.dm_exec_sql_text(qs.plan_handle) as q
```

where

```
qs.total_worker_time > qs.total_elapsed_time
```

SQL Trace

Look for the following signs of parallel queries, which could be either statements or batches that have CPU time greater than the duration.

内部查询的并行

CPU时间大于持续时间:

select

EventClass,

TextData

from

::fn_trace_gettable('c:\temp\high_cpu_trace.trc', default)

where

EventClass in (10, 12) -- RPC:Completed, SQL:BatchCompleted

and CPU > Duration/1000 -- CPU is in milliseconds, Duration in

microseconds Or can be Showplans (un-encoded) that have Parallelism
operators in them

select

EventClass,

TextData

from

::fn_trace_gettable('c:\temp\high_cpu_trace.trc', default)

where

TextData LIKE '%Parallelism%'

CPU瓶颈

- 拙劣游标使用
 - SQL Server: Cursor Manager By Type – Cursor Requests/Sec
- CPU瓶颈:
 - Processor: % Processor Time < 80%

内存瓶颈

- 外部物理内存压力
 - 任务管理器: Physical Memory—Available>100M
 - 系统监视器 : Memory: Available Bytes
- 外部虚拟内存压力
 - 任务管理器: Commit Charge
 - Total 与 Limit 是否接近
- 内部物理内存压力
 - 任务管理器 Mem Usage 列
 - SQL Server: **Buffer Manager** 对象
 - Low Buffer cache hit ratio
 - Low Page life expectancy
 - High number of Checkpoint pages/sec
 - High number Lazy writes/sec

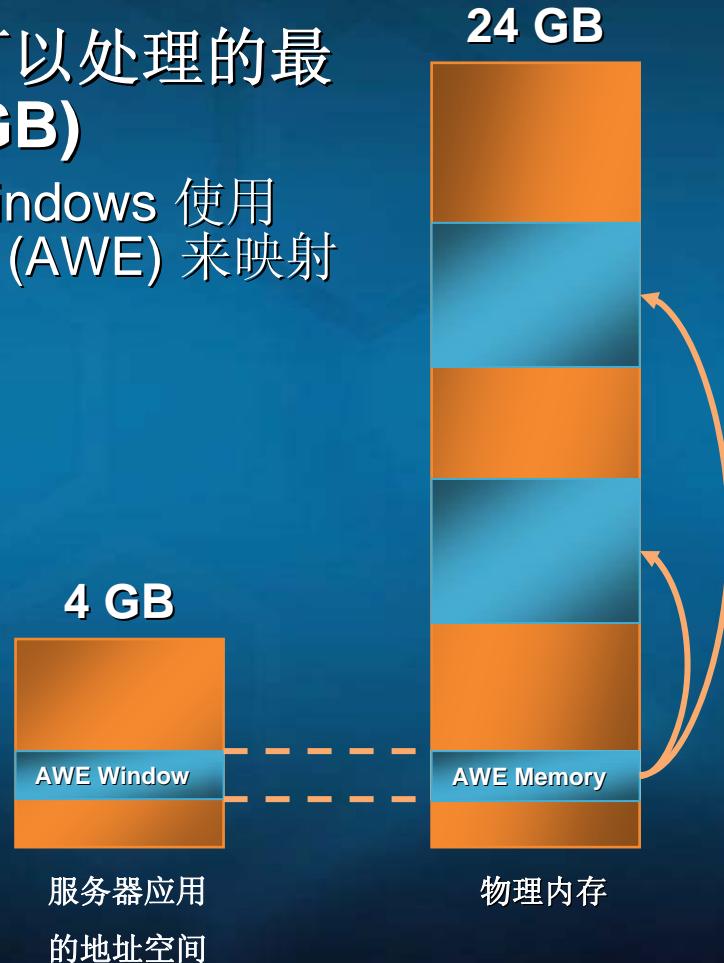
SQL Server 2005支持的最大内存

配置	VAS	最大物理内存	AWE/locked pages 支持
Native 32-bit on 32-bit OS with /3GB boot parameter	2 GB 3 GB	64 GB 16 GB	支持 支持
32-bit on x64 OS (WOW)	4 GB	64 GB	支持
32-bit on IA64 OS (WOW)	2 GB	2 GB	不支持
Native 64-bit on x64 OS	8 terabyte	1 terabyte	支持
Native 64-bit on IA64 OS	7 terabyte	1 terabyte	不支持

64-bit 计算

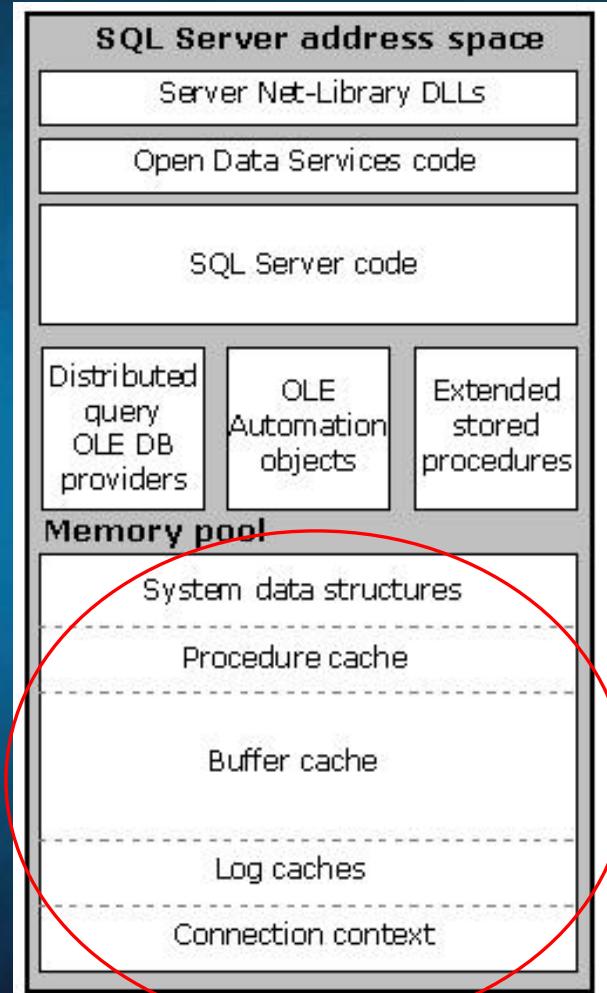
32位系统使用更多内存的方法

- 使用 **32-bit** 操作系统，用户可以处理的最大内存空间为 **4 GB ($2^{32} = 4 \text{ GB}$)**
 - 为了使用更多的内存，**32-bit Windows** 使用 **Address Windowing Extensions (AWE)** 来映射更多的内存
 - 这种额外的在物理和虚拟内存的转换会对性能有一些影响



64位关系型数据库系统(RDBMS) 优势

- 增大的可直接寻址空间
 - 消除了对 AWE 层的需求
 - 所有对数据库的操作都可从中获益
 - 并不仅仅只是对数据缓冲的操作
 - 提高了对并发用户的 support (游标, 事务, 锁等)
 - 大的查询缓存, 过程缓存, 排序堆, 缓冲池, 开放的数据库对象, 临时结果集 (复杂查询计划) 等
- 处理器的体系结构能够更好地处理并发
 - 对数据库的并行处理有益
 - 在一个时钟周期内可以做更多的工作
- 提高了总线与 I/O 的带宽与吞吐量



• DBCC MEMORYSTATUS

- Target
- Target是SQL Server计算出它在不导致分页时可以提交的8-KB每页的页数。Target是被定期的重新计算的来反映内存的低或高。在常规服务负载下target页面过低可能预示出现了外部内存压力。

Buffer Counts

Committed

Target

Hashed

Reserved Potential

Stolen Potential

External Reservation

Min Free

Visible

Available Paging File

Buffers

201120

201120

166517

143388

173556

0

256

201120

460640

内存压力

压力	外部	内部
物理	<p>物理内存(RAM)运行值低。这导致系统整理当前运行的工具集，导致整体性能下降。</p> <p>SQL Server监测到这种条件，依赖于配置，可以减少缓存池的目的提交并开始清理内部缓存。</p>	<p>SQL Server检测内部较高的内存消耗，导致在不同内部组件间的内存重新分配。</p> <p>内部内存压力可以导致：</p> <ul style="list-style-type: none">导致外部内存压力（SQL Server设置地的内存使用能力）。改变内存设置（例如'max server memory'）。改变内部组件的内存分布（导致预留的高百分比并从缓存池中获取页）。
虚拟的	<p>在系统页面文件运行在较低值。这样可以导致系统分配内存失败。不能扩展当前的内存分配。这可以导致着整个系统响应很慢或者可能导致系统关机。</p>	<p>在VAS运行值低，导致分页（很多VAS可用，但是被分为小块）与/或消耗（直接分配，DLL加载到SQL VAS，大量的线程）。</p> <p>SQL Server检测到这种条件并可以释放VAS中保留的区域，减少缓存池提交的目标并开始收缩缓存。</p>

内存压力的解决

压力

内部

外部

物理

找到主要的系统内存消耗组件。
尝试消除消耗(如果可能)。
检查适当的系统RAM和考虑添加额外
RAM (通常需要更仔细研究)

识别SQL Server内主要的内存消耗
确认系统配置。
进一步操作依赖于研究；检查负载；
可能出现的设计问题；其他的资源
瓶颈。

虚拟

增加交换文件大小。
检查主要物理内存的使用和外部物理
内存压力调用步骤。

外部物理内存压力调用步骤。

I/O瓶颈

- PhysicalDisk Object: Avg. Disk Queue Length > 2
- **Avg. Disk Sec/Read (write)** 是平均每次从磁盘读取(写入)数据的时间
 - 小于10 ms – 很好
 - 在 10 - 20 ms 之间- 正常
 - 在20 - 50 ms 之间- 缓慢, 需要注意
 - 大于 50 ms – 严重的I/O 瓶颈
- Physical Disk: %Disk Time 所选磁盘驱动器用于服务于读或写请求的总共时间的百分比。
 - 大于50%, 则表现为I/O瓶颈
- Avg. Disk Reads (write) /Sec 表现磁盘上读操作的速度
 - < 85%的磁盘设计能力
- RAID配置时, 你需要使用下列公式调整结果值
 - Raid 0 -- I/Os per disk = $(\text{reads} + \text{writes}) / \text{number of disks}$
 - Raid 1 -- I/Os per disk = $[\text{reads} + (2 * \text{writes})] / 2$
 - Raid 5 -- I/Os per disk = $[\text{reads} + (4 * \text{writes})] / \text{number of disks}$
 - Raid 10 -- I/Os per disk = $[\text{reads} + (2 * \text{writes})] / \text{number of disks}$

I/O瓶颈

- 解决方法：
 - 增加I/O带宽
 - 添加更多的物理驱动器
 - 添加快速或额外的I/O控制器
 - 查看那个计划占用了更多的I/O
 - select top 5
(total_logical_reads/execution_count) as avg_logical_reads,
(total_logical_writes/execution_count) as avg_logical_writes,
(total_physical_reads/execution_count) as avg_phys_reads,
Execution_count,
statement_start_offset as stmt_start_offset,
sql_handle,
plan_handle
 - from sys.dm_exec_query_stats
 - order by
(total_logical_reads + total_logical_writes) Desc
 - 可以通过下列查询或取执行查询的文本
 - select text
 - from sys.dm_exec_sql_text(
PlanHandle)

Tempdb 瓶颈

- **Tempdb**磁盘空间不足
- **tempdb**中的I/O瓶颈
- 过多DDL操作导致系统表的瓶颈
- 分配争夺

Tempdb

用户对象

这些对象被用户会话显示创建并在系统目录中被跟踪。这包括：

- 表和索引
- 全局临时表(##t1)和索引
- 本地临时表(#t1)和索引
- 会话范围
- 存储过程范围内创建
- 表变量(@t1).
- 会话范围
- 存储过程范围内创建

内部对象

这些有语句范围的对象，通过SQL Server处理的查询创建和销毁。这些对象不能被系统目录跟踪。这包括：

- 工作文件(hash join)
- 排序
- 工作表 (游标, 池 和临时大对象数据类型 (LOB)存储)

版本存储

用于存储行版本。MARS，在索引因操作，触发器和快照隔离级别都是基于行版本。这是SQL Server 2005中新的特性。

Tempdb

tempdb用户使用的空间和内部组件对象

Select

```
    SUM (user_object_reserved_page_count)*8 as user_objects_kb,  
    SUM (internal_object_reserved_page_count)*8 as  
internal_objects_kb,  
    SUM (version_store_reserved_page_count)*8 as  
version_store_kb,  
    SUM (unallocated_extent_page_count)*8 as freespace_kb  
From sys.dm_db_file_space_usage  
Where database_id = 2
```

user_objs_kb internal_objects_kb version_store_kb
freespace_kb

8736

128

64

448

tempDB

- 用户对象
 - `exec sp_spaceused @objname='<user-object>'`
 - 枚举所有tempdb对象：

枚举所有tempdb对象

```
DECLARE userobj_cursor CURSOR FOR
select
    sys.schemas.name + '.' + sys.objects.name
from sys.objects, sys.schemas
where object_id > 100 and
    type_desc = 'USER_TABLE' and
    sys.objects.schema_id =
    sys.schemas.schema_id
go
open userobj_cursor
go

declare @name varchar(256)
fetch userobj_cursor into @name
while (@@FETCH_STATUS = 0)
begin
    exec sp_spaceused @objname = @name
    fetch userobj_cursor into @name
end
close userobj_cursor
```

tempDB

- 版本存储
 - 触发器
 - MARS
 - 联机索引
 - 基于行版本隔离级别：需要在数据库级设置选项
- 行版本需要跨会话共享。当行版本被回收时，行版本的创建者没有控制权。你需要找到并杀掉阻止行版本清理的运行最长的事务。

tempDB

- 版本存储运行最长的2个事务

```
select top 2
    transaction_id,
    transaction_sequence_num,
    elapsed_time_seconds
from sys.dm_tran_active_snapshot_database_transactions
order by elapsed_time_seconds DESC
```

transaction_id	transaction_sequence_num	elapsed_time_seconds
8609	3	6523
20156	25	783

tempDB

- 内部对象
 - 内部对象在每个会话或任务中被创建和销毁
 - 如果你注意到有大量的**tempdb**空间分配，你将需要了解那个会话或任务占用了空间
 - **sys.dm_db_session_space_usage**
 - 分配给个别会话的空间
 - **sys.dm_db_task_space_usage**
 - 分配给个别任务所用的**tempdb**空间

tempDB

- 分配对象最多的会话，包括正在运行的任务

SELECT

```
t1.session_id,  
(t1.internal_objects_alloc_page_count + task_alloc) as allocated,  
(t1.internal_objects_dealloc_page_count + task_dealloc) as  
deallocated  
from sys.dm_db_session_space_usage as t1,  
(select session_id,      sum(internal_objects_alloc_page_count)  
     as task_alloc,    sum (internal_objects_dealloc_page_count) as  
task_dealloc  
     from sys.dm_db_task_space_usage group by session_id) as t2  
where t1.session_id = t2.session_id and t1.session_id >50  
order by allocated DESC
```

session_id	allocated	deallocated
52	5120	5136
51	16	0

tempDB

```
select
    t1.session_id,
    t1.request_id,
    t1.task_alloc,
    t1.task_dealloc,
    t2.sql_handle,
    t2.statement_start_offset,
    t2.statement_end_offset,
    t2.plan_handle
from (Select session_id,
            request_id,
            sum(internal_objects_alloc_page_count) as task_alloc,
            sum (internal_objects_dealloc_page_count) as task_dealloc
        from sys.dm_db_task_space_usage
       group by session_id, request_id) as t1,
     sys.dm_exec_requests as t2
where t1.session_id = t2.session_id and
      (t1.request_id = t2.request_id)
order by t1.task_alloc DESC
```

tempDB



```
select text from sys.dm_exec_sql_text(@sql_handle)  
select * from sys.dm_exec_query_plan(@plan_handle)
```

TempDB优化

- 过多的**DLL**和分配操作
 - 如果你使用存储过程范围内的临时表，考虑是否这些表可以移动到存储过程外。否则每次执行存储过程将会导致创建/删除临时表。
 - 查看查询计划，是否一些计划创建大量的临时对象，池，排序或工作表。你需要评估一些临时对象。例如，在一个列上创建一个用于**ORDER BY**操作的索引可以除去查询时的排序

阻塞

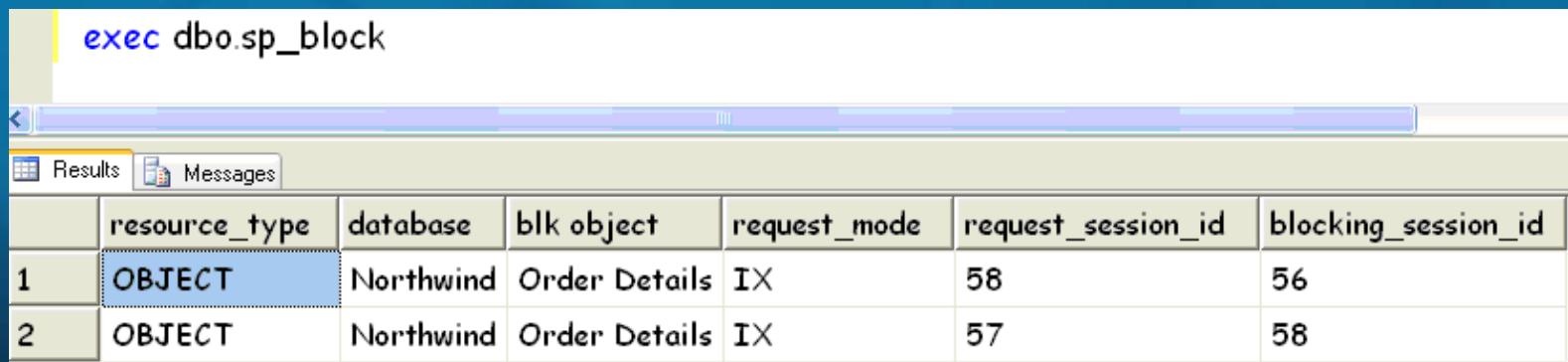
- Exec dbo.sp_lock
- sys.dm_os_wait_statistics
- sys.dm_os_waiting_tasks
- Dead Lock Graph
- Activity Monitor

阻塞

- 阻塞信息: **sys.dm_os_waiting_tasks**
select * from sys.dm_os_waiting_tasks
where session_id=56
- 等待锁的会话信息:
 - **sys.dm_tran_locks**

阻塞

- 连接上面的2个DMV， 使用存储过程 sp_block锁展示的。
- 阻塞报告列出了被阻塞的会话和阻塞它的会话。



```
exec dbo.sp_block
```

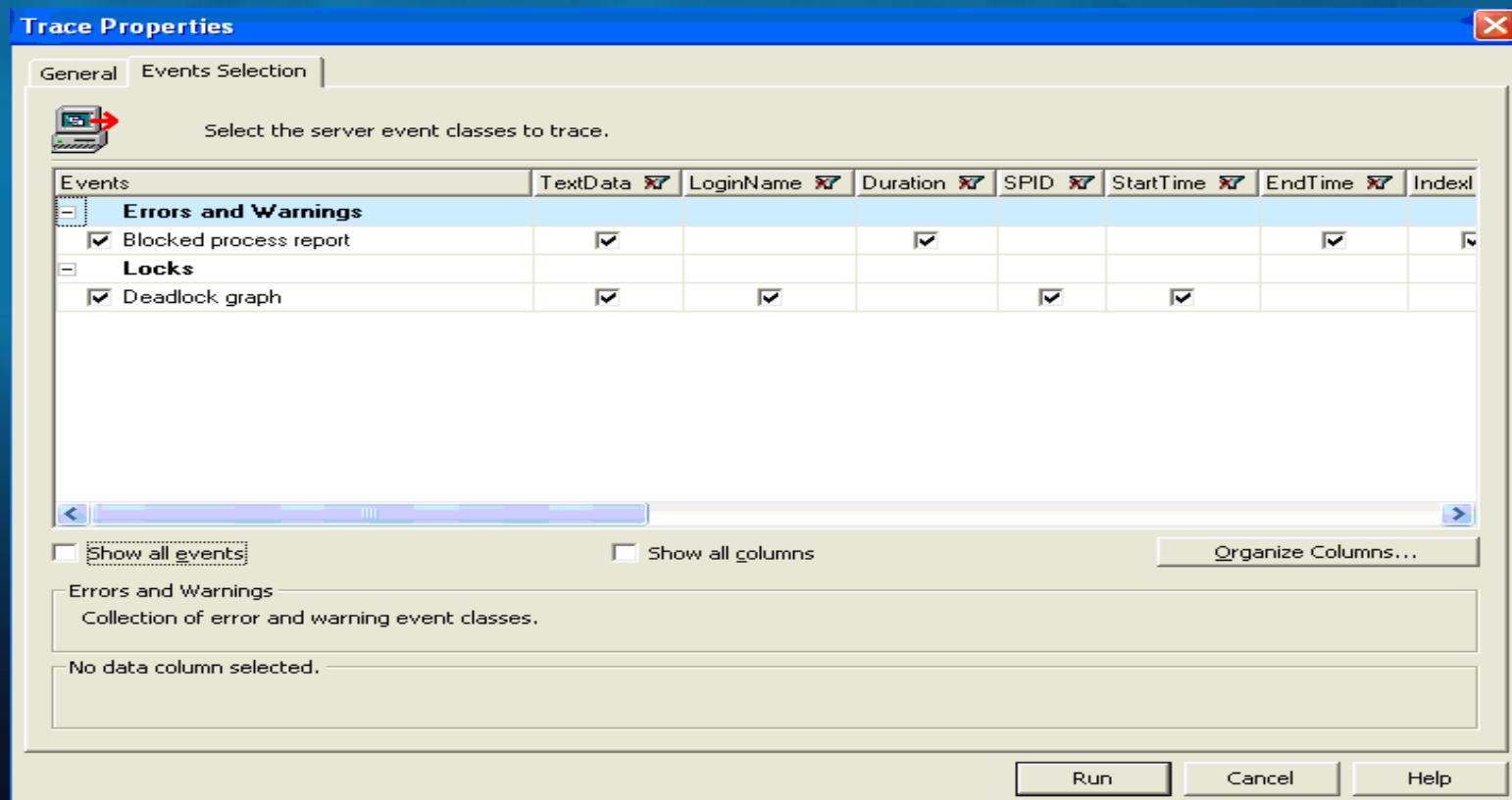
	resource_type	database	blk object	request_mode	request_session_id	blocking_session_id
1	OBJECT	Northwind	Order Details	IX	58	56
2	OBJECT	Northwind	Order Details	IX	57	58

识别长时间的阻塞

- 识别长时间的阻塞
- 配置阻塞阀值：
 - Execute Sp_configure 'blocked process threshold', 200
 - **Reconfigure with override**

识别长时间的阻塞

- 如果使用SQL Server Profiler，选择 Blocked Process Report 事件类（在 Error 和 Warnings 对象下）



优化软硬件配置

- 硬件：
 - C P U , 内存 , I O
- 数据库
 - 表分区
 - 将数据文件, 临时表空间存放在不同磁盘
 - 调整 S Q L S e r v e r 内存分配
 - 修改事务隔离级别
 - 数据库优化顾问