Microsoft SQL Server 2005

SQL Server 2005 대기 및 큐

번역자: 성대중 (djsung@feelanet.com)

목차

소개

이 문서에서는 대기 및 큐 분석 방법론을 사용하여 Microsoft® SQL Server™ 성능을 튜닝하는 과정을 소개합니다. 대기 및 큐 분석 방법론은 성능을 향상시킬 수 있는 최적의 방안을 도출하는데 도움을 줄 수 있습니다. 이러한 성능 향상의 측면에는 성능 튜닝을 위해 소비하는 시간을 획기적으로 절약할 수 있다는 것도 포함됩니다. 대기 및 큐 분석 방법론은 두 가지 측면에서 성능 문제를 찾아 내고, 성능 문제를 발생시키는 정확한 원인이 되는 부분을 찾아내는데 도움을 줍니다. 성능 문제를 해결해야 하는 담당자는 대기와 큐라는 두 가지 관점과 관련된 데이터를 수집하여 분석함으로써 문제의 원인을 찾아낼 수 있습니다. 대기 분석은 SQL Server 가 대기 시간을 어떤 작업을 위해 사용했는지를 알아내기 위해 수행합니다. 가장 많은 대기가 발생하는 대기 유형은 현재 작업부하에 상태에서, 의미가 있고 중요한 큐(성능 모니터 카운터 및 기타 다른 데이터) 정보와 관련이 되어 있습니다. 대기 분석을 통해, 수집된 성능 카운터 정보 중에서 의미가 있고 중요한 카운터 정보를 제외한 나머지 성능 카운터 정보는 분석 대상에서 제외시킬 수 있고, 문제의 원인이 되는 정확한 리소스를 찾아내는데 도움을 줄 수 있습니다.

대기 및 큐 분석을 통한 성능 튜닝 방법론은 사용자가 어플리케이션이나 솔루션에 포함되어 있는 새로운 성능 문제 또는 잠재적인 성능 문제의 원인이 될 수 있는 부분을 단순한 어림짐작이 아니라 실질적인 방법으로 찾을 수 있도록 해 주기 때문에, 어플리케이션 성능 문제를 찾아내고 해결하는데 빠르고 효과적인 방법이 될 수 있습니다.

개요

어플리케이션과 솔루션에 대한 성능 튜닝은 이미 오랜 기간 수행되어 왔습니다. SQL Server 2005 데이터베이스 어플리케이션의 성능은 몇 가지 관점으로 평가되어야 합니다. 각 관점은 전체 성능을 보장하기 위한 작업을 몇 가지 하위 단계로 구분한 것입니다. 각 단계는 성능 개선을 위한 전체적인 그림의 일부분이 되며, 필요에 따라 각 관점간의 상관관계를 살펴보는 것도 중요한 의미를 가집니다. 성능을 보장하기 위한 방법론에는 어플리케이션과 SQL Server 의 관점에서 각각에 관련된 시스템 또는 리소스에 대한 고려사항이 포함됩니다. 그 중에 대기 및 큐 분석 방법론이 가장 기초적인 역할을 수행하게 됩니다.

일부 병목현상은 다른 병목현상에 비해 비교적 쉽게 해결할 수 있습니다. 예를 들어, 임의(ad hoc) 쿼리가 쿼리 실행계획을 재사용하지 못하는 문제를 해결하기 위해서는 1) sp_executesql 을 사용하여 임의 쿼리를 매개변수화된 쿼리로 변경하거나, 2) 임의 쿼리를 저장 프로시저로 변경하면 됩니다. 하지만, 이러한 변경작업을 수행하기 위해서는 어플리케이션의 변경작업이 필요하기 때문에, 추가적인 코드변경 및 테스트에 대한 시간 및 공수가 투입되어야 합니다.

목적

이 문서의 목적은 개발자와 데이터베이스 관리자에게 SQL Server 2005 와 관련된 어플리케이션이나 솔루션의 성능을 개선하는데 도움을 줄 수 있는 접근방법을 제공하기 위함입니다. 이 문서에서는 어플리케이션의 성능 문제의 원인을 찾아내기 위한 프로세스와 가이드라인을 제공하며, 성능을 개선할 수 있는 권장사례 방법론에 대해서도 소개합니다. 이 문서에 포함된 각 개념은 이전 버전 SQL Server 환경에도 적용될 수 있는 것이지만, 일부 SQL Server 2005 에서 제공하는 새로운 기능을 기반으로 한 예제들은 이전 버전 SQL Server 에서는 동작하지 않을 수도 있습니다.

이 문서에서 소개하는 성능 튜닝 방법론은 다소 모호할 수 있는 성능 문제를 찾아내고, 이에 대한 근본적인 원인을 좀 더 신속하게 찾아낼 수 있도록 해 줍니다. SQL Server 의 다양한 버전에 대한 성능 튜닝 및 최적화와 관련된 문서와 서적이 많이 있습니다. 이 문서에는 SQL Server 2005 에 초점을 맞추어 논의를 진행하게 됩니다. 이 문서에 포함된 내용은 SQL Server 개발팀과 여러 전문가의 전문지식을 근거로 하여 작성된 것입니다.

대상: 이 문서를 읽어야 하는 대상

이 문서는 SQL Server 플랫폼에서 운영되는 어플리케이션이나 솔루션을 개발하고 성능 튜닝 작업을 수행하는 개발자, 테스터, 데이터베이스 관리자를 위해 작성된 것입니다. 이 문서를 읽는 대상은 이미 SQL Server 의 기본적인 명령과 어플리케이션 성능 튜닝에 대한 기초 지식을 가지고 있다고 가정합니다. 하지만, 개발 중에 있는 어플리케이션의 성능을 테스트 자체를 대체하거나, 다른 성능 최적화 관련 문서를 대체하는 것은 아닙니다.

대기 및 큐: 성능 튜닝 방법론

어플리케이션에서와 마찬가지로, SQL Server 도 사용자의 쿼리를 실행하기 위해서 시스템 리소스를 요청하고, 요청한 리소스를 획득할 때까지 대기합니다. 대기(Waits)는 SQL Server 대기 통계정보를 통해 표현됩니다. SQL Server 2005 에서는 사용자 연결 또는 특정 session_id 가 대기중 상태에 있을 동안에는 언제든지 대기 정보를 추적할 수 있습니다. 현재 주어진 작업부하 상태에서 성능에 대한 상태정보를 얻기 위해서, 현재 모든 연결에 대한 대기 정보를 요약하고 범주별로 구분하는 것이 도움이 될 수 있습니다. 그러므로, SQL Server 에서는 대기 유형을 사용하여 어플리케이션의 작업부하나 사용자의 관점에서 현재 특정 사용자(또는 스레드)에 대한 대기 정보를 구분하고, 범주화합니다.

큐(Queue)는 시스템 리소스와 사용률을 측정하기 위해 사용합니다. 성능에 대한 큐 정보는 성능 모니터의 성능 개체와 카운터에 의해 표현됩니다. 성능 모니터 카운터는 디스크의 전송률이나 CPU 사용률과 같은 다양한 성능 데이터를 측정하기 위해 사용됩니다. 성능 모니터의 SQL Server 관련 카운터 정보는 *sys.dm_os_performance_counters* 동적 관리 뷰(DMV)를 사용하여 수집됩니다. 그러므로, 성능 모니터 카운터는 해당 시점의 리소스에 대한 성능 정보를 나타냅니다.

성능 카운터와 대기유형과의 연관성과 상관관계를 분석하고, 문제가 되는 성능 카운터 수치와 비교하는 작업을 통해, 전체적인 성능에 대한 전체적인 구도를 잡아갈 수 있습니다. 대기와 큐 사이의 연관성을 분석하고, 해당 범위 안에서 서로 연관관계가 없는 카운터를 제거함으로써, 문제 영역에서 좀 더 효과적으로 성능 병목지점을 찾아낼 수 있습니다. 특정 카운터를 다른 카운터와 비교하는 작업은 올바른 결론을 추론하기 위한 합리적인 관점을 제공합니다. 예를 들어, 운영 환경의 작업부하 상태에서 1,000 개의 잠금 대기가 발생하였다고 가정합니다. 이러한 경우, 성능에 지대한 악영향을 미치는 원인이 된다고 할 수 있을까요? 현재의 대기 상태가 전체적인 성능에 영향을 주는지를 판단하기 위해서는, 전체 잠금 요청 수(전체적으로 얼마나 많은 잠금 요청이 발생하는지 vs. 얼마나 많은 잠금 대기가 발생하는지), 잠금으로 인한 전체 대기시간과 잠금 대기 시간의 지속기간, 테스트 시간대 등이 모두 함께 고려되어야 합니다. 만약, 전체 잠금 요청수가 1,000 만 건인 상태에서, 잠금 대기 수가 1,000 이라면 그리 심각한 상태가 아닐 것입니다. 또한, 총 8 시간 동안 발생한 작업부하 상태에서, 잠금 대기 관련 대기 시간이 총 50 초였다면, 전혀 문제가 없는 상황이라고 판단할 수 있습니다. 반면에, 1,000 개의 잠금 대기마다 평균 50 초씩의 잠금 대기시간이 발생하는 상황이라면, 분명하게 잠금과 관련한 문제가 있는 상태로 판단할 수 있습니다. 결론적으로, 전체적인 성능의 타당성을 검증하기 위해서는, 반드시 대기유형과 성능 카운터 간의 연관성이나 상관관계가 고려되어야 합니다.

SQL Server 의 대기 정보와, 시스템 또는 리소스 큐에 대한 정보를 살펴보면, 좀 더 쉽게 어플리케이션의 성능에 대해서 설명할 수 있습니다. SQL Server 2005 에서는, *sys.dm_os_wait_stats* DMV 를 통해, 어플리케이션의 관점에서 유용하게 사용할 수 있는 대기 정보를 얻을 수 있습니다.

시스템 리소스 사용량에 대한 상세 정보를 제공하는 성능 모니터의 성능 카운터와 기타 데이터 원본을 통해, 시스템이나 리소스 관점에서 유용하게 사용할 수 있는 리소스 큐 정보를 얻을 수 있습니다. 대기와 큐 분석 방법론을 병행함으로써, 어플리케이션 관점과 리소스 관점에서 시스템에 발생하는 병목현상을 찾아내고, 병목현상과 관련이 없는 성능 정보를 제거할 수 있습니다.

실행 모델 (단순화)

SQL Serve 의 실행 모델과 가장 유사한 사례는, 식료품 상점의 계산대 대기열에서 찾을 수 있습니다. 계산대는 CPU 의 역할을 합니다. 현재 계산대에서 계산하고 있는 고객은 현재 실행중인 세션에 비유할 수 있습니다. 계산대 대기열에서 대기하고 있는 고객은 실행가능한(runnable) 큐에 대기하고 있는 세션에 비유할 수 있습니다. 만약, 현재 계산대에서 계산하고 있던 고객 1 이 특정 상품에 대한 가격 조회를 요청한 경우, 고객 1 은 가격 조회가 완료될 때까지 대기해야만 합니다. 고객 1 이 가격조회가 완료될 때까지 대기하는 동안, 해당 대기열에 다음 차례로 대기하고 있던, 고객 2 가 계산대에서 계산작업을 수행할 수 있습니다. 고객 1 의 가격조회 작업이 완료되면, 계산대에서 다시 고객 1 에 대한 계산 작업을 수행할 수 있습니다. 이러한 과정이 SQLOS 라고 불리는 SQL Server 실행 모델에서 수행하는 작업절차를 잘 표현하고 있습니다.

SQL Server SQLOS 에서는 사용자 요청에 대한 실행을 관리하기 위해 스케줄러를 사용합니다. SQLOS 스케줄러는 CPU 와 매핑되어 있습니다. 4 CPU 서버에서는, 기본적으로 4 개의 SQLOS 스케줄러가 존재합니다. 다음 다이어그램은 단일 SQLOS 스케줄러를 사용하는 SQL Server 의 실행 모델을 단순화하여 나타낸 것입니다. 그림 1 의 실행모델은 SQL Server 가 사용자 요청이나 세션(SPID 로 표현됨)을 어떤 단계로 실행하는지를 잘 표현하고 있습니다.

그림 1: 실행 모델 – 실행 중(Running), 실행대기(runnable), 보류(suspended) 상태 프로세스가 포함된 실행대기(Runnable) 큐와 대기 목록

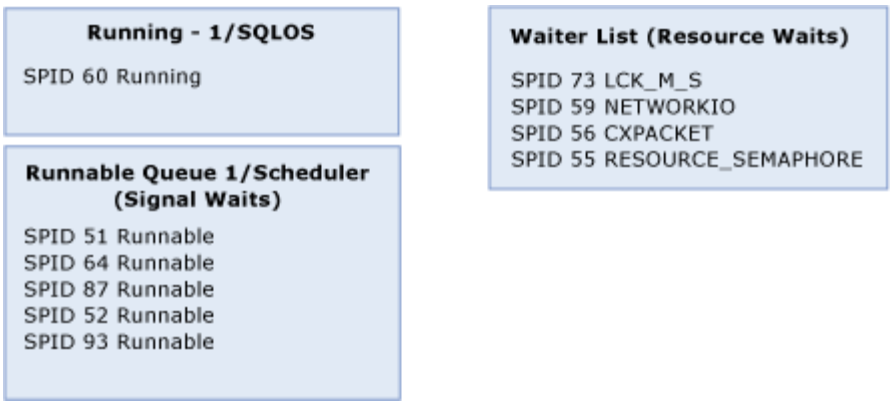
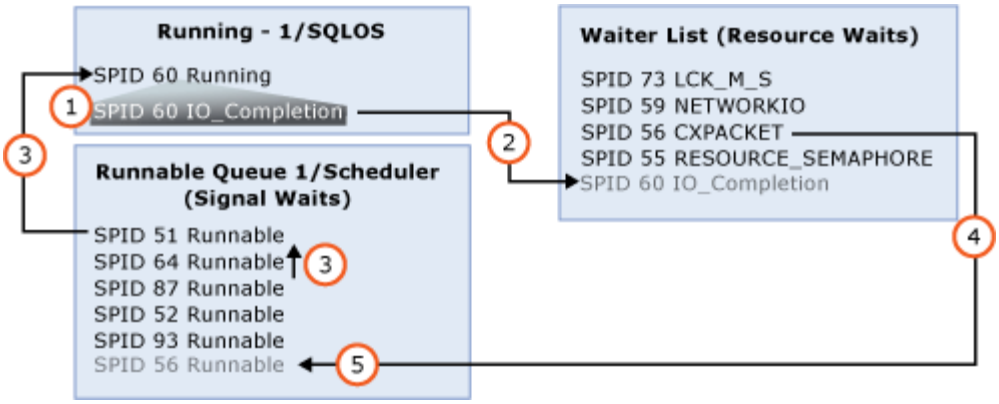


그림 2 는 SQL Server 가 특정 세션의 상태를 실행중(running; 스케줄러 당 실행 중인 세션은 하나만 존재), 실행가능(Runnable; CPU 에 대기하고 있는 세션), 보류중(suspended)으로 어떻게 전환하는지를 나타냅니다. 보류중 상태의 SPID 는 요청된 리소스가 가용한 상태가 될 때까지 대기 목록에 위치하게 됩니다. 실행중인 세션이 버퍼 캐시에 존재하지 않는 데이터 페이지를 요청한 경우나, 다른 사용자가 잠금을 설정한 페이지를 요청하여 차단현상이 발생한 경우에는, 해당 세션을

대기 목록으로 이동시킵니다. 그 다음, 실행가능 큐에 존재하는 다음 순서의 SPID 또는 session_id 가 실행되도록 스케줄링합니다.

그림 2: 실행 모델 – SPID 의 상태 변경 과정

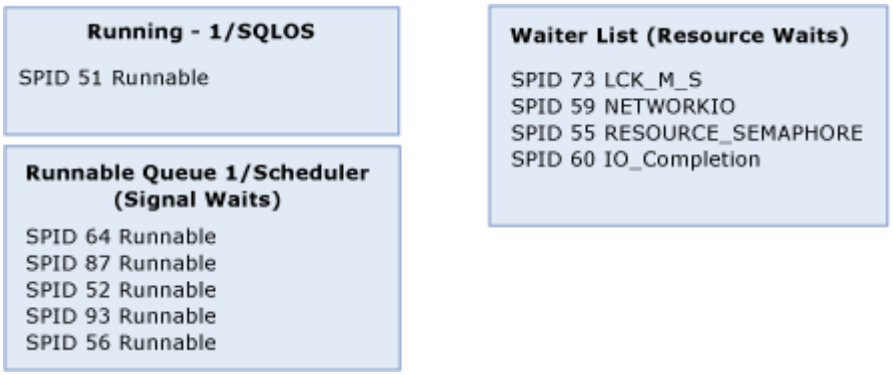


세션의 상태가 변경되는 순서는 다음과 같습니다.

1. SPID60 가 버퍼 캐시에 존재하지 않는 페이지를 요청합니다. 디스크로부터 메모리로 데이터를 가져올 때까지 대기하기 위해, SPID60의 상태를 실행중(Running)에서 IO_Completion 대기유형을 가진 보류중(Suspended) 상태로 변경합니다.
2. SPID60 를 대기자 목록으로 이동시킵니다.
3. 실행가능 큐에 대기중이던 실행가능(Runnable) 상태의 SPID51를 실행중(Running) 상태로 이동시키고, SPID64를 실행가능 큐의 최상위로 이동시킵니다.
4. SPID56 는 병렬처리가 완료되기까지 대기하고 있는 상태입니다. 병렬처리가 완료되면, SPID56의 상태가 CXPACKET 대기유형을 가진 보류중(Suspended) 상태에서 실행가능(Runnable) 상태로 변경됩니다.
5. SPID56 를 실행가능 큐의 맨 아래 위치로 이동시킵니다.

그림 3 은 상태가 변경된 이후의 실행 모델 상태를 나타냅니다.

그림 3: 실행 모델 – 상태 변경 이후



대기 목록은 리소스를 기다리고 있는 스레드의 집합을 의미합니다. 리소스 대기에는 IO 가 완료할 때까지의 대기, 잠금이 해제될 때까지의 대기, 메모리를 할당받을 때까지의 대기 등이 포함됩니다. 세션이 대기 목록으로 이동되면, 대기유형이 할당되고, 해당 대기유형에 대한 대기 시간이 누적됩니다. 리소스가 가용한 상태가 되면, 스레드는 다시 실행가능 큐로 이동되고, CPU 가 사용한 상태가 되자마자 바로 실행됩니다. 사용자 요청을 완료할 때까지, 실행중, 실행가능, 보류중 상태를 전환하는 작업이 지속됩니다.

대기자 목록과 대기 유형

SQL Server 2005 에서 특정 session_id 가 보류중 상태로 전환되면, 해당 session_id 가 보류중 상태로 전환된 이유를 나타내는 대기유형이 지정됩니다. 대기자 목록은, 현재 보류중 상태로 설정된 세션 정보(session_id 칼럼)와 보류 중 상태로 전환 이유(wait_type 칼럼), 현재 대기유형으로 대기한 누적 대기 시간(wait_duration_ms 칼럼) 등의 정보를 제공하는 sys.dm_os_waiting_tasks DMV 를 사용하여 조회할 수 있습니다. 만약, 다른 세션이 잠금을 설정하고 있어서, 해당 잠금이 해제될 때까지 대기하는 차단현상이 발생하고 있는 상황이라면, 차단주체가 되는 잠금을 설정하고 있는 세션정보(blocking_session_id 칼럼)와 차단된 리소스 정보(resource_description 칼럼)가 추가로 조회됩니다.

sys.dm_os_waiting_tasks DMV 에서는 현재 대기 목록을 확인할 수 있습니다. 현재 실행가능 큐 정보는 sys.dm_exec_requests DMV 에서 상태가 "runnable"인 세션만 조회하면 됩니다. sys.dm_os_waiting_tasks 에서 현재까지 대기한 총 대기시간 정보는 wait_time_ms 칼럼에 나타나고, 현재까지 실행가능 큐에서 CPU 를 기다리고 있는 동안 발생한 대기시간 정보는 signal_wait _time_ms 칼럼에 나타납니다. 리소스 대기시간은 총 대기시간(wait_time_ms)에서 시그널 대기시간(signal_wait_time_ms)을 차감하여 계산합니다. 실행가능 큐는 동일한 유형의 트랜잭션이 매우 많이 반복되는 OLTP 작업부하 상황에서는 피할 수 없습니다. 실행가능 큐에 머무는 시간보다는 CPU 에 대한 대기시간과 대기목록의 리소스 대기시간을 비교하는 것이 더 중요한 관건이 됩니다. 리소스 대기시간과 시그널 대기시간의 차이는 전체적인 성능측면에서 CPU 병목현상이 발생하고 있는지 여부를 판단하는 근거가 될 수 있습니다. 만약, 전체 리소스 대기시간 대비 시그널 대기시간의 비율이 낮은 수치(시그널 대기시간이 전체 대기시간의 25%미만인 경우)로 유지되면, 현재 CPU 에 대한 병목현상은 발생하지 않고 있다고 판단할 수 있습니다.

동적 관리 뷰 (DMVs)와 동적 관리 함수 (DMFs)

동적 관리 뷰(DBVs)와 동적 관리 함수(DMFs)는 현재 서버에서 세션, 트랜잭션, 사용자 요청이 얼마나 발생하고 있는지에 대한 서버 상태 정보의 변경 내역을 제공합니다. 동적 관리 뷰와 동적 관리 함수는 특정 서버 프로세스에서 발생하고 있는 내역과, 서버의 모든 세션에서 발생하고 있는 내역을 모두 반영합니다. 그렇기 때문에, 동적 관리 뷰와 동적 관리 함수를 사용하여, 현재 모든 세션에 대한 정보를 진단하고, 메모리와 각 프로세스에 대한 성능을 개선하고, 모니터링할 수 있습니다.

성능 튜닝 목적에 유용한 DMV 에는 sys.dm_exec_requests, sys.dm_os_waiting_tasks, sys.dm_os_wait_stats 가 있습니다.

Sys.dm_exec_requests

모든 SQL Server 세션은 sys.dm_exec_requests DMV 에 고유한 session_id 를 가집니다. sp_who2 저장 프로시저는 세션 목록과 함께, 명령, 리소스, 대기 유형, 대기 시간, 상태 등의 정보를 제공합니다. 사용자 쿼리에 대한 session_id 는 50 보다 큰 값으로 할당됩니다. 실행 모델 섹션에서 설명한 바와 같이, 각 세션의 상태는 일반적으로, 'running', 'runnable' 'suspended'으로 설정됩니다. 세션의 상태가 'Sleeping'으로 나타나는 경우는, SQL Server 가 다음 명령을 수신하기 위해 대기하고 있는 상태를 나타냅니다.

Sys.dm_os_waiting_tasks

sys.dm_os_waiting_tasks DMV 는 모든 대기 중인 세션과 대기 유형을 나타내는 대기자 목록을 제공합니다. session_id, 대기 유형, 이와 관련된 대기 시간 정보도 함께 제공됩니다. 또한, 특정 세션이 다른 세션에서 설정한 잠금으로 인해 차단이 발생하는 경우에는, 잠금을 설정한 차단 주체에 대한 세션 정보(blocking_session_id 칼럼)와 차단된 리소스(resource_description)에 대한 정보도 제공됩니다.

Sys.dm_os_wait_stats

Sys.dm_os_wait_stats DMV 는 최근 SQL Server 재시작 시점 또는 DBCC SQLPERF ('sys.dm_os_wait_stats', CLEAR) 명령을 사용하여 수작업으로 재설정한 시점으로부터 현재까지 모든 세션에서 발생한 대기에 대한 통계 정보의 누적값을 제공합니다. 테스트를 수행하기 전이나, 대량의 작업부하가 발생하기 전에 대기 통계정보를 재설정하면, 해당 작업으로 인해 발생한 대기 통계정보를 쉽게 파악할 수 있습니다.

특정 세션이 리소스에 대한 대기를 필요로 하는 경우, 해당 세션은 리소스에 관련한 대기유형이 지정된 상태로 대기 목록으로 이동됩니다. sys.dm_os_waiting_tasks DMV 에서 제공하는 대기 목록은 주어진 시점에 대한 정보입니다. 이에 비해, sys.dm_os_wait_stats DMV 에서 제공하는 대기 정보는 모든 세션에서 발생한 대기를 집계한 통계정보입니다.

track_waitstats_2005 저장 프로시저와 *get_waitstats_2005* 저장 프로시저를 사용하면, 주어진 작업부하에 대한 대기 통계정보를 측정할 수 있습니다.

Track_waitstats_2005 저장 프로시저

Track_waitstats_2005 저장 프로시저는 sys.dm_os_wait_stats DMV 로부터 대기 통계정보를 수집하고, 이를 대기유형별 퍼센트를 기준으로 역순으로 정렬합니다. 성능개선에 가장 큰 도움이 되는 대기 유형을 찾아내기 위해서 사용합니다. **Track_waitstats_2005** 저장 프로시저에 대한 스크립트는 다음 링크를 참고하십시오

http://www.microsoft.com/technet/scriptcenter/scripts/sql/sql2005/waitstats/sql05vb048.mspx

Get_waitstats_2005 저장 프로시저

get_waitstats_2005 저장 프로시저는 track_waitstats_2005 저장 프로시저에 의해 수집한 대기 유형의 정보를 조회하는 역할을 합니다. get_waitstats_2005 저장 프로시저는 track_waitstats 저장 프로시저가 실행 중인 동안에도 실행될 수 있으며, track_waitstats 저장 프로시저의 실행이 완료된 상태에서도 실행할 수 있습니다. track_waitstats_2005 저장 프로시저를 실행하는 동안, get_waitstats_2005 저장 프로시저를 실행하면, 중간 단계의 결과를 반환하는데 반해, track_waitstats_2005 저장 프로시저의 실행이 완료된 다음에 get_waitstats_2005 저장 프로시저를 실행하면, 최종 상태의 대기 통계 정보를 반환합니다.

get_waitstats_2005 저장 프로시저의 실행 결과는 측정한 대상 시간 동안 발생한 여러 가지 유형의 대기 유형에 대한 상세 정보와 각 대기 유형별 대기 시간을 제공합니다.

Get_waitstats_2005 저장 프로시저의 실행결과는 대기에 대한 정보를 제공합니다. 전체 대기 시간은 리소스 대기와 시그널 대기로 구성됩니다. 리소스 대기는 전체 대기시간에서 시그널 대기 시간을 차감하여 계산합니다. 시그널 대기 시간은 CPU 리소스를 획득하기 위해 실행가능 큐에서 대기한 총 시간을 의미하며, CPU 병목현상이 발생하고 있는지를 판단하는 기준이 됩니다. 전체 대기 시간 중에서 시그널 대기 시간이 점유하는 비율을 계산하면, 현재 어플리케이션에 CPU 병목현상이 발생하고 있는지 여부를 판단할 수 있습니다. **get_waitstats_2005** 저장 프로시저에 대한 스크립트는 다음 링크를 참조하십시오.

http://www.microsoft.com/technet/scriptcenter/scripts/sql/sql2005/perf/sql05vb021.mspx

성능 및 튜닝 청사진

이 백서에서는 다양한 유형의 어플리케이션에서 리소스를 어떻게 사용하는지, 다양한 작업부하 상태에서 어플리케이션의 성능을 어떻게 개선할 수 있는지에 대해서 살펴볼 것입니다. OLTP 작업부하와 관계형 데이터웨어하우스 또는 리포팅 어플리케이션의 작업부하는 완전히 다른 것으로 고려되어야 하며, 어플리케이션의 목적에 따라 성능 개선을 위한 접근 방법도 달라지게 됩니다.

리소스의 병목현상이 각 어플리케이션마다 다르다고 할지라도, 대기 및 큐 분석 성능 튜닝 방법론을 사용하면, 매우 정확한 진단이 가능하며, 재현가능한 결과를 얻을 수 있습니다. 하나의 병목현상을 해결하게 되면, 어플리케이션의 처리량이 증가하여 다른 부분에 병목현상이 발생할 가능성이 있습니다.

OLTP 작업부하는 일반적으로 소수의 동일한 트랜잭션이 매우 많이 반복 실행되는 특징을 갖습니다. 반면에, 데이터웨어하우스나 리포팅 어플리케이션은 서로 다른 트랜잭션이 비교적 적게 반복 실행되는 특징을 갖습니다. 이러한 차이점이 있기 때문에, 각 어플리케이션의 목적에 따라 서로

다른 리소스 사용량 구성을 가지게 됩니다. 각 어플리케이션 환경에 대한 청사진에는 이러한 각 어플리케이션의 특성이 반영되어야 합니다.

OLTP 청사진

OLTP 어플리케이션은 주로 SELECT, INSERT, UPDATE, DELETE 등의 작업으로 구성된, 소수의 동일한 트랜잭션이 매우 많이 반복 실행되는 특징을 갖습니다. 여러 개의 CPU 를 사용하여 대용량 쿼리를 좀 더 작은 단위로 나누어서 병렬처리를 해야 하는 대용량 데이터웨어하우스나 리포팅 어플리케이션과 달리, OLTP 환경의 소규모 쿼리에서는 병렬처리가 필요하지 않습니다. 병렬처리란 여러 개의 CPU 를 사용하여, 대용량 쿼리를 좀 더 작은 단위로 분할하여 실행하는 것을 말합니다.

쿼리를 분할하여 여러 개의 CPU 를 사용하여 실행하면 좀 더 빠르게 실행할 수 있는 반면에, 최종 결과집합을 만들기 위해, 각 CPU 에서 분할하여 실행한 결과를 정렬하고 병합하는 과정이 필요하기 때문에, 그만큼 CPU 리소스가 더 소비됩니다. OLTP 트랜잭션은 비교적 소규모이기 때문에, 좀 더 빠르게 실행하기 위해 부가적인 CPU 와 메모리 리소스를 사용해야 하는 병렬처리 작업을 수행할 필요가 없습니다. 게다가, OLTP 환경에서는 대량의 트랜잭션을 처리해야 하기 때문에, CPU 리소스가 낭비되지 않도록 하는 것이 매우 중요합니다. 병렬처리는 쿼리가 대용량이지만, 비교적 적은 수로 실행되는 데이터웨어하우스나 리포팅 어플리케이션에서 좀 더 적합하다고 할 수 있습니다.

데이터베이스의 설계, 리소스의 사용량, 시스템 성능은 서로 밀접하게 연관되어 있습니다.

OLTP 환경에 적합한 성능 청사진을 제시하기 위한 다음의 표에 나타난 리소스 관련 이슈 중, 어느 것 하나라도 일치되는 상황이 발생하고 있다면, 성능과 확장성의 문제가 발생하고 있다고 판단할 수 있습니다.

주의 **값(value)** 칼럼에 있는 값은 단지 기준값입니다. 실제 환경에서는 다양한 값으로 나타날 수 있습니다.

OLTP환경에서 피해야 할 일반적인 시나리오

데이터베이스 설계

규칙	설명	값	원천정보	문제 설명
1	여러 개의 테이블이 조인되는 쿼리가 자주 반복 실행되는 경우	>4	Sys.dm_exec_sql_text Sys.dm_exec_cached_plans	여러 개의 테이블이 조인되는 쿼리가 자주 반복 실행되는 경우는, 높은 수준의 OLTP 확장성을 위해, 데이터 모델을 과도하게 정규화한 것이 원인이 될 수

				있습니다..
2	여러 개의 인덱스가 설정된 테이블에 자주 업데이트가 발생하는 경우	>3	Sys.indexes sys.dm_db_operational_index_stats	OLTP 환경에 적합하지 않게 과도하게 많은 인덱스가 설정된 경우
3	대량 IO 가 발생하는 쿼리, 테이블 스캔, 범위 스캔이 발생하는 쿼리	>1	성능 모니터 개체 SQL Server Access Methods Sys.dm_exec_query_stats	누락된 인덱스로 인해 데이터 캐시가 모두 비워질만큼의 대용량 IO 가 발생하는 경우
4	사용되지 않는 인덱스	Sys.dm_db_index_usage_stats DMV 에 나타나지 않는 인덱스. 인덱스가 전혀 사용되지 않으면, sys.dm_db_index_usage_stats DMV 에 나타나지 않습니다.		사용되지 않는 인덱스를 정리하지 않은 경우

CPU

규칙	설명	값	원천정보	문제 설명
1	시그널 대기	>25%	Sys.dm_os_wait_stats	실행가능(runable) 큐에 순수한 CPU 대기가 있는 경우
2	쿼리 실행계획 재사용	<90%	성능 모니터 개체 SQL Server Statistics	OLTP 환경의 동일 유형 트랜잭션에서는 실행계획 재사용률이 95% 이상으로 유지되어야 이상적인 상태
3	병렬처리: Cxpacket 대기	>5%	Sys.dm_os_wait_stats	병렬처리는 OLTP 처리량(throughput)을 줄어들게 하는 원인이 됩니다. CXPACKET 대기는 여러 개의 CPU 를 사용하여 쿼리를 좀 더 작은 단위로 분할하여 실행한다는 것을 나타냅니다. 일반적으로, 인덱스가 누락되어 있거나, WHERE 절이 제대로 구성되어 있지 않거나, 쿼리가 OLTP

				트랜잭션이라고 보기 어려울 정도로 복합한 대량 쿼리가 아니라면, 잘 튜닝된 OLTP 어플리케이션에서는 병렬처리가 발생하지 않습니다.

메모리

규칙	설명	값	원천정보	문제설명
1	Page life expectancy	<300 초	성능 모니터 개체 SQL Server Buffer Manager SQL Server Buffer Nodes	Page life expectancy 는 데이터 페이지가 데이터 캐시 상에 머무는 평균 시간을 초 단위로 나타냅니다. 이 성능 카운터 값이 낮게 나타난다면, 대량 IO 를 발생시키는 쿼리가 데이터 캐시를 비우고 다시 데이터를 캐시에 등록하고 있다는 것을 나타냅니다. 순수한 OLTP 작업부하에서는 이러한 대량 IO 를 발생하는 쿼리가 발생해서는 않는 것이 바람직하기 때문에, 인덱스 누락되지 않았는지 확인해 볼 필요가 있습니다.
2	Page life expectancy	50%이하로 떨어짐	Perfmon object SQL Server Buffer Manager	Page life expectancy 는 데이터 페이지가 데이터 캐시 상에 머무는 평균 시간을 초 단위로 나타냅니다. 이 성능 카운터 값이 낮게 나타난다면, 대량 IO 를 발생시키는 쿼리가 데이터 캐시를 비우고 다시 데이터를 캐시에 등록하고 있다는 것을 나타냅니다. 순수한 OLTP 작업부하에서는 이러한 대량 IO 를 발생하는 쿼리가 발생해서는 않는 것이 바람직하기 때문에, 인덱스 누락되지 않았는지 확인해 볼 필요가 있습니다.
3	Memory Grants Pending	>1	성능 모니터 개체 SQL Server Memory Manager	현재 작업영역(workspace) 메모리를 할당받기 위해 대기 중인 프로세스의 수

4	SQL cache hit ratio	<90%	SQL cache hit ratio 성능 카운터 수치가 90% 이하로 떨어져 60 초 이상 지속되는 경우.	버퍼 캐시를 한꺼번에 비우는 대량 스캔을 발생시키는 쿼리가 있다는 것을 의미합니다.

IO

규칙	설명	값	원천정보	문제설명
1	Average Disk sec/read	>20 ms	성능 모니터 개체 Physical Disk	IO 병목현상이 발생하지 않는다면, 읽기 작업은 4-8 ms 내외로 완료되어야 합니다.
2	Average Disk sec/write	>20 ms	성능 모니터 개체 Physical Disk	트랜잭션 로그에 대한 쓰기(순차적 쓰기) 작업은 1 ms 내외로 완료되어야 합니다.
3	대용량 IO, 테이블 스캔, 범위 스캔을 발생시키는 쿼리	>1	성능 모니터 개체 SQL Server Access Methods	누락된 인덱스로 인해, 대량 IO 를 발생시켜 버퍼 캐시를 비우는 현상이 발생하는 경우
4	상위 2 위의 대기 유형에 다음과 같은 대기유형이 포함되는 경우: ASYNCH_IO_COMPLETION IO_COMPLETION LOGMGR WRITELOG PAGEIOLATCH_x	상위 2 위	Sys.dm_os_wait_stats	상위 2 위의 대기 유형에 IO 에 관련된 대기유형이 포함되면, IO 병목현상이 발생하고 있음을 나타냅니다.
5	초당 읽기/쓰기 바이트 수가 낮은 경우		성능 모니터 개체 Physical Disk	

차단(Blocking)

규칙	설명	값	원천정보	문제설명
1	차단 발생 퍼센트	>2%	Sys.dm_db_index_operational_stats	차단이 발생하는 빈도
2	Block process report	30 초	Sp_configure 저장 프로시저, SQL Server 2005 프로파일러	차단이 발생하는 쿼리가 이벤트로 보고됩니다.
3	Average Row Lock Waits	>100ms	Sys.dm_db_index_operational_stats	차단의 지속시간

4	상위 2 위의 대기유형에 다음과 같은 유형의 대기유형이 포함되는 경우: LCK_M_BU LCK_M_IS LCK_M_IU LCK_M_IX LCK_M_RIn_NL LCK_M_RIn_S LCK_M_RIn_U LCK_M_RIn_X LCK_M_RS_S LCK_M_RS_U LCK_M_RX_S LCK_M_RX_U LCK_M_RX_X LCK_M_S LCK_M_SCH_M LCK_M_SCH_S LCK_M_SIU LCK_M_SIX LCK_M_U LCK_M_UIX LCK_M_X	상위 2 위	Sys.dm_os_wait_stats	상위 2 위의 대기 유형에 IO 에 관련된 대기유형이 포함되면, 차단관련 병목현상이 발생하고 있음을 나타냅니다.
5	데드락이 다수 발생하는 경우	시간당 >5	추적플래그 1204 를 사용하여 에러로그에 데드락 정보를 기록하거나, 프로파일러를 사용하여 deadlock graph 이벤트를 수집	동일 쿼리내에서 또는 동시에 실행되는 여러 개의 명령 사이에서 데드락이 발생하면, 잠금과 관련된 문제가 발생하고 있다는 것을 나타냅니다.

네트워크

규칙	설명	값	원천정보	문제설명
1	데이터베이스로의 라운드 트립이 자주 발생하면서, 높은 네트워크 지연현상이 있는 경우	Output queue length >2	성능모니터 개체: Network Interface	어플리케이션 서버와 데이터베이스 서버간에 네트워크 지연이 발생하고 있음을 나타냅니다. 어플리케이션 서버와 SQL Server 인스턴스간에 통신을

				담당하는 네트워크 인프라에 문제의 원인이 있을 가능성이 있습니다.
2	네트워크 대역폭이 거의 대부분 사용되는 경우	Packets Outbound Discarded Packets Outbound Errors Packets Received Discarded Packets Received Errors	성능모니터 개체: Network Interface	손실된 패킷이 나타나는 경우

요약해보면, 소수의 동일한 유형의 트랜잭션이 많이 반복되는 특징을 가지는 OLTP 환경에서는, 초당 트랜잭션 처리량과 리소스의 사용량을 향상시키기 위해 다음의 권고사항을 고려해야 합니다.

1. 데이터베이스 설계 상 INSERT, UPDATE, DELETE 작업이 발생할 때마다 인덱스에 대한 유지관리 작업이 발생하기 때문에, 반드시 기능적으로 꼭 필요한 최소한의 수만큼만 인덱스를 생성해야 합니다.
2. 실행계획을 재사용할 수 있게 하고, 조인되는 대상 테이블의 수를 줄여줌으로써, CPU 사용률을 줄일 수 있습니다.
3. 인덱스 전략의 최적화, 조인되는 대상 테이블 개수의 축소, 높은 page life expectancy 수치 유지를 통해서 IO 성능을 개선할 수 있습니다.
4. 메모리에 대해서는 Page Life Expectancy 성능 카운터 수치가 갑자기 떨어지는 현상이 발생하지 않는 것이 최적인 상태입니다.
5. 정렬 순서가 인덱스의 사용을 제한시킬 수도 있습니다. 즉, 특정 순서로 정렬하는 작업은 해당 정렬순서(내림차순 또는 오름차순)와 동일한 조건으로 정렬된 상태로 구성된 인덱스를 사용하여 수행될 수 있습니다.
6. 인덱스 전략 최적화, 트랜잭션 길이의 단축 등의 방법으로 차단현상을 줄일 수 있습니다.

데이터웨어하우스 청사진

OLTP 환경에 비해, 데이터웨어하우스 어플리케이션은 규모가 큰 트랜잭션이 비교적 적은 양 반복해서 수행되는 특징을 갖습니다. 이러한 작업부하 특성은 정확하게 OLTP 작업부하의 정반대의 특징입니다. 데이터웨어하우스 또는 리포팅 어플리케이션에서는 주로 대용량 SELECT 또는 읽기전용 작업이 수행됩니다. 데이터베이스의 설계, 리소스의 사용량, 시스템 성능은 서로 밀접하게 연관되어 있습니다. 인덱스의 조각화, 캐시의 회전율, IO 성능은 이러한 작업부하 상황에 매우 중요한 요소로 작용합니다.

데이터웨어하우스 환경에 적합한 성능 청사진을 제시하기 위한, 다음의 표에 나타난 리소스 관련 이슈 중에 어느 것 하나라도 일치되는 상황이 발생하고 있다면, 성능과 확장성의 문제가 발생하고 있다고 판단할 수 있습니다

주의 value 칼럼에 포함된 실제 값은 논란의 여지가 있을 수 있지만, 데이터웨어하우스 또는 리포팅 어플리케이션에서 발생할 수 있는 일반적인 성능문제를 인식하기 위한 기준값으로 사용될 수 있습니다.

데이터웨어하우스에서 피해야 할 일반적인 시나리오

데이터베이스 설계

규칙	설명	값	원천정보	문제상황
1	커버된 인덱스를 사용하여, 과도한 정렬작업이나 RID 검색 작업이 발생할 가능성을 최소화		Sys.dm_exec_sql_text Sys.dm_exec_cached_plans	대용량 데이터웨어하우스에서는 인덱스가 많으면 많을수록 효과를 볼 수 있습니다. 인덱스를 사용하면 부가적인 정렬 작업을 피할 수 있으며, 커버된 쿼리가 되도록 할 수 있습니다. 데이터웨어하우스 쿼리는 대부분 읽기전용 쿼리이기 때문에, 인덱스로 인한 오버헤드는 데이터를 로드할 때에만 발생하게 됩니다.
2	과도한 인덱스 조각화 문제: Average fragmentation_in_percent 칼럼이 25% 미만으로 유지	>25%	sys.dm_db_index_physical_stats	인덱스 재생성작업을 통해 인덱스의 조각화를 제거하면, 데이터웨어하우스 또는 리포팅 시나리오에서 일반적으로 발생하는 대량 범위 스캔 작업의 성능을 향상시킬 수 있습니다..
3	스캔과 범위스캔은 데이터웨어하우스 환경에서는 일반적인 경우이지만, 필요한 인덱스가 누락되었는지는 확인해야 함	>= 1	Perfmon object SQL Server Access Methods Sys.dm_db_missing_index_group_stats Sys.dm_db_missing_index_groups Sys.dm_db_missing_index_details	인덱스가 누락되면, 대량 IO 를 발생시켜, 데이터 버퍼 캐시가 비워지는 원인이 될 수 있습니다.
4	사용되지 않는 인덱스	Sys.dm_db_index_usage_stats DMV 에 나타나지 않는 인덱스. 인덱스가 전혀 사용되지 않으면, sys.dm_db_index_usage_stats DMV 에 나타나지 않습니다.		사용되지 않는 인덱스를 정리하지 않은 경우

리소스 이슈: CPU

규칙	설명	값	원천정보	문제상황

1	시그널 대기	> 25%	Sys.dm_os_wait_stats	실행가능(runnable) 큐에 순수한 CPU 대기가 있는 경우
2	실행계획 재사용 회피	> 25%	성능 모니터 개체 SQL Server Statistics	데이터웨어하우스 환경에서는 OLTP 환경에서보다 비교적 적은 수의 트랜잭션이 실행되지만, 각 트랜잭션은 비교적 대용량 IO 를 발생시키는 특성을 갖습니다. 그러므로, 실행계획을 재사용하는 것보다는 개별 쿼리에 좀 더 최적화된 실행계획을 가질 수 있도록 하는 것이 중요합니다. OLTP 와는 달리, 데이터웨어하우스 쿼리는 동일한 유형의 쿼리가 반복되어 실행되지 않습니다.
3	병렬처리: Cxpacket 대기	<10%	Sys.dm_os_wait_stats	병렬처리는 데이터웨어하우스 또는 리포팅 작업부하 환경에서는 바람직한 현상입니다.

리소스 이슈: 메모리

규칙	설명	값	원천정보	문제상황
1	Memory grants pending	>1	성능 모니터 개체 SQL Server Memory Manager	쿼리를 실행하기 위해 필요한 메모리를 할당받을 수 없는 경우. 충분한 메모리 여유공간이 있는지 여부와 page life expectancy 성능 카운터 수치를 확인해야 합니다.
2	Page life expectancy	50%이하로 떨어짐	성능 모니터 개체 SQL Server Buffer Manager	Page life expectancy 는 데이터 페이지가 데이터 캐시 상에 머무는 평균 시간을 초 단위로 나타냅니다. 이 성능 카운터 값이 낮게 나타난다면, 대량 IO 를 발생시키는 쿼리가 데이터 캐시를 비우고 다시 데이터를 캐시에 등록하고 있다는 것을 나타냅니다. 필요한 인덱스 누락되지 않았는지 확인해 볼 필요가 있습니다.

리소스 이슈: IO

규칙	설명	값	원천정보	문제상황
1	Average Disk sec/read	>20 ms	성능 모니터 개체 Physical Disk	IO 병목현상이 발생하지 않는다면, 읽기 작업은 4-8 ms 내외로 완료되어야 합니다.
2	Average Disk sec/write	>20 ms	성능 모니터 개체	트랜잭션 로그에

			Physical Disk	대한 쓰기(순차적 쓰기) 작업은 1 ms 내외로 완료되어야 합니다.
3	대량 스캔	>1	성능 모니터 개체 SQL Server Access Methods	누락된 인덱스로 인해, 대량 IO 를 발생시켜 버퍼 캐시를 비우는 현상이 발생하는 경우
4	상위 2 위의 대기 유형에 다음과 같은 대기유형이 포함되는 경우: ASYNCH_IO_COMPLETION IO_COMPLETION LOGMGR WRITELOG PAGEIOLATCH_x	상위 2 위	Sys.dm_os_wait_stats	상위 2 위의 대기 유형에 IO 에 관련된 대기유형이 포함되면, IO 병목현상이 발생하고 있음을 나타냅니다.

리소스 이슈: 차단(Blocking)

규칙	설명	값	원천정보	문제상황
1	차단 발생 퍼센트	>2%	Sys.dm_db_index_operational_stats	차단이 발생하는 빈도
2	Block process report	30 초	Sp_configure, profiler	차단이 발생하는 쿼리가 이벤트로 보고됩니다.
3	Average Row Lock Waits	>100ms	Sys.dm_db_index_operational_stats	차단의 지속시간
4	상위 2 위의 대기유형에 다음과 같은 유형의 대기유형이 포함되는 경우: LCK_M_BU LCK_M_IS LCK_M_IU LCK_M_IX LCK_M_RIn_NL LCK_M_RIn_S LCK_M_RIn_U LCK_M_RIn_X	상위 2 위	Sys.dm_os_wait_stats	상위 2 위의 대기 유형에 IO 에 관련된 대기유형이 포함되면, 차단관련 병목현상이 발생하고 있음을 나타냅니다.

	LCK_M_RS_S LCK_M_RS_U LCK_M_RX_S LCK_M_RX_U LCK_M_RX_X LCK_M_S LCK_M_SCH_M LCK_M_SCH_S LCK_M_SIU LCK_M_SIX LCK_M_U LCK_M_UIX LCK_M_X			

정확하게 OLTP 어플리케이션과 반대의 특성을 나타내는, 관계형 데이터웨어하우스 또는 리포팅 어플리케이션은 서로 다른 대용량 트랜잭션이 비교적 적은 수로 반복하여 실행되는 특성을 갖습니다. 대부분의 쿼리는 SELECT 작업 중심으로 수행됩니다. 데이터베이스의 설계, 리소스의 사용량, 시스템 성능은 서로 밀접하게 연관되어 있습니다.

리포팅과 데이터웨어하우스 환경에서는, 성능을 보장하기 위해 다음의 권고사항을 고려해야 합니다.

1. 데이터웨어하우스 및 관계형 데이터웨어하우스 설계에서는 인덱스의 유지관리 비용이 배치로 데이터를 업데이트하는 절차에서만 발생하기 때문에, 좀 더 많은 인덱스를 만들 것을 고려해 보아야 합니다.
2. 일반적으로, 데이터웨어하우스 환경에선 실행계획을 재사용하지 않도록 해야 합니다. 실행계획을 재사용하게 되면, 다른 데이터 분포를 기준으로 최적화한 다른 쿼리의 실행계획을 재사용하게 될 수 있기 때문에, 현재 실행하는 데이터웨어하우스 쿼리에는 적합하지 않은 실행계획이 될 가능성이 있습니다. 대용량 데이터웨어하우스 쿼리에서는 실행계획 생성을 위한 시간이 소요되더라도 반드시 해당 쿼리에 최적화된 실행계획을 수립할 수 있도록 해야 합니다.
3. 정렬 작업은 올바른 인덱스를 사용하게 함으로써 최소화할 수 있습니다.
4. 누락된 인덱스가 있는 조사하고, 이를 적절하게 수정해 주어야 합니다.
5. 범위 스캔과 같이 대량 IO 를 발생시키는 작업은 디스크 상의 데이터 페이지의 연속성에 영향을 받게 됩니다. 그러므로, 인덱스 조각화 정도를 주기적으로 모니터링하고, 인덱스 재생성 작업을 통해, 인덱스 조각화가 최소화될 수 있도록 관리해야 합니다.
6. 데이터웨어하우스 트랜잭션은 대부분 읽기 작업이기 때문에, 차단 현상은 일반적으로 발생하는 상황은 아닙니다.
7. 데이터웨어하우스 어플리케이션에서는 병렬처리가 일반적으로 권장되는 작업입니다.

리소스 병목현상의 유형

리소스 병목현상은 대기와 큐 정보간의 상관관계를 통해 찾아낼 수 있습니다. 병목현상의 유형에는 메모리 압박, IO, CPU, 네트워크, 차단 등이 포함됩니다. 어플리케이션에 따라, 리소스가 서로 다른 유형으로 사용되며, 서로 다른 리소스에서 성능 병목현상이 발생할 수 있습니다. 어플리케이션의

특성을 분석하는 과정을 통해, 데이터베이스 설계, 리소스의 사용량, 성능에 대한 목표를 찾아낼 수 있다.

대기 및 큐 방법론에 대한 좀 더 자세한 정보를 얻기 위해서, 다음 링크에서 관련 DMV 스크립트와 예제를 참조하여 활용할 수 있습니다. (http://www.microsoft.com/technet/scriptcenter/scripts/sql/sql2005/default.mspx)

메모리 압박 및 IO 서브시스템 이슈

디스크로부터 메모리로 데이터를 로드하는 과정에서 지연현상이 발생하게 되면, PageIOLatch 대기가 나타냅니다. 메모리 압박이나, 디스크 서브시스템에 문제가 발생한 경우에도, PageIOLatch 대기가 증가합니다. 사용자가 버퍼 캐시에서 필요로 하는 데이터 페이지를 찾을 수 없는 경우, SQL Server 는 먼저 데이터 버퍼 페이지를 할당한 다음, 필요로 하는 페이지를 디스크로부터 데이터 캐시로 로드하는 동안 해당 버퍼 페이지에 PageIOLatch_ex 배타적 래치를 설정합니다. 그 동안에, 다른 사용자가 해당 버퍼 페이지에 대한 읽기 요청을 하게 되면, SQL Server 는 사용자를 대신 하여 사용자가 요청한 버퍼 페이지에 대해 PageIOLatch_sh 래치를 요청하고 대기합니다. 캐시에 대한 쓰기 작업이 완료되면, PageIOLatch_Ex 래치가 해제됩니다. PageIOLatch_Ex 래치가 해제되면, 비로소 사용자가 해당 버퍼 페이지를 읽을 수 있도록 허용하며, PageIOLatch_sh 래치도 해제됩니다. 결론적으로, PageIOLatch_Ex 대기유형과 PageIOLatch_sh 대기유형이 높은 수치로 나타나면, IO 서브시스템에 문제가 발생하고 있다는 것을 나타냅니다. 관련된 성능 카운터로는 Physical disk: disk seconds/read, Physical disk: disk seconds/write, SQL Server Buffer Manager: Page Life Expectancy 등이 있습니다. 좀 더 자세한 정보는 각 성능 카운터에 대한 정보를 참조하십시오.

IO 대기

테이블값을 반환하는 sys.dm_io_virtual_file_stats 동적 관리 함수는 특정 데이터베이스 파일이나 트랜잭션 로그 파일에 발생한 읽기, 쓰기, io_stall 값 정보를 제공합니다. IO_Stall 값은, 최근에 SQL Server 를 재시작한 시점부터 현재까지, 사용자가 해당 파일에 IO 작업을 완료하기까지 대기한 총 누적시간을 밀리초 단위로 나타낸 것입니다.

- *Select * from sys.dm_io_virtual_file_stats (dbid,file#)*
- *Select * from sys.dm_io_virtual_file_stats (dbid,NULL)* 지정된 데이터베이스의 전체 파일 목록을 표시

만약, 하나 또는 그 이상의 파일의 IO_Stall 값이 비정상적으로 높은 경우라면, 디스크 병목현상이 발생하고 있거나, 하나의 드라이브에 매우 많은 읽기/쓰기 작업이 실행되고 있다는 것을 나타냅니다. 읽기당 평균 IO 대기 시간이나, 쓰기 당 평균 IO 대기 시간 수치를 통해, 높은 IO 대기가 지속적으로 발생하였는지, 일시적인 IO 스파이크 현상이 발생하였는지를 구분할 수 있습니다. 특정 드라이브에 IO Stall 에 대한 평균 값이 매우 높게 나타나는 경우, 해당 드라이브에 지속적으로 높은 IO 요청이 발생하고 있다는 것을 나타냅니다. 이는 성능 모니터의 Physical Disk: Average Disk Seconds/Read , Average Disk Seconds/Write 카운터를 통해 확인할 수 있습니다. 다음 스크립트는

sys.dm_io_virtual_file_stats 동적관리함수(DMF)를 사용하여, Average Disk Seconds/Read 와 Average Disk Seconds/Write 를 계산합니다.

```
---- 0 으로 나누기 오류를 방지하기 위해, 읽기, 쓰기 당 평균 stall 값에 1.0 을 더함
select database_id, file_id
    ,io_stall_read_ms
    ,num_of_reads
    ,cast(io_stall_read_ms/(1.0+num_of_reads) as numeric(10,1)) as 'avg_read_stall_ms'
    ,io_stall_write_ms
    ,num_of_writes
    ,cast(io_stall_write_ms/(1.0+num_of_writes) as numeric(10,1)) as 'avg_write_stall_ms'
    ,io_stall_read_ms + io_stall_write_ms as io_stalls
    ,num_of_reads + num_of_writes as total_io
    ,cast((io_stall_read_ms+io_stall_write_ms)/(1.0+num_of_reads + num_of_writes) as
numeric(10,1)) as 'avg_io_stall_ms'
from sys.dm_io_virtual_file_stats(null,null)
order by avg_io_stall_ms desc
```

인덱스의 누락 및 잘못 설정된 인덱스

인덱스가 누락되었거나, 인덱스가 잘못 설정된 경우에는, 과도한 메모리 압박과 데이터 캐시가 비워지게 되는 문제의 원인이 될 수 있습니다. SQL Server 2005 쿼리 최적화 프로그램에서는 특정 쿼리(그림 1)를 실행하기 위해, 잠재적으로 유용한 인덱스를 찾습니다. 인덱스의 효과를 계산한 수치가 *avg_user_impact(제안된 인덱스를 사용함으로써 얻을 수 있는 성능 향상 퍼센트)* 칼럼에 나타냅니다. 이러한 성능향상 퍼센트는 INSERT, UPDATE, DELETE 작업을 통해 인덱스 유지관리 비용이 발생하는 경우에만, 개별 쿼리에 적용됩니다.
다음 스크립트는 자주 사용되는 인덱스 목록을 나타냅니다.

```
-- 잠재적으로 유용한 인덱스
select d.*
        , s.avg_total_user_cost
        , s.avg_user_impact
        , s.last_user_seek
        ,s.unique_compiles
from sys.dm_db_missing_index_group_stats s
        ,sys.dm_db_missing_index_groups g
        ,sys.dm_db_missing_index_details d
```

```
where s.group_handle = g.index_group_handle
and d.index_handle = g.index_handle
order by s.avg_user_impact desc
go
--- 인덱스에 포함되어야 할 칼럼과 사용량
declare @handle int

select @handle = d.index_handle
from sys.dm_db_missing_index_group_stats s
        ,sys.dm_db_missing_index_groups g
        ,sys.dm_db_missing_index_details d
where s.group_handle = g.index_group_handle
and d.index_handle = g.index_handle

select *
from sys.dm_db_missing_index_columns(@handle)
order by column_id
```

대량 IO를 발생시키는 쿼리

sys.dm_db_missing_index_columns DMV 에 의해서 제안된 인덱스의 목적은, 문제가 되는 쿼리에서 발생하는 대량 IO 를 줄이기 위한 것입니다. 그러므로, 대량 IO 가 발생하는 쿼리의 우선순위를 결정하기 위해서 해당 쿼리를 찾아내야 합니다. 대량 IO 가 발생하는 쿼리를 찾기 위해서는, 다음 예제 코드를 실행합니다.

```
--- IO 기준 상위 50 위 쿼리
SELECT TOP 50
        (qs.total_logical_reads + qs.total_logical_writes) /qs.execution_count as [Avg IO],
        substring (qt.text,qs.statement_start_offset/2,
         (case when qs.statement_end_offset = -1
          then len(convert(nvarchar(max), qt.text)) * 2
              else qs.statement_end_offset end -     qs.statement_start_offset)/2)
        as query_text,
    qt.dbid,
    qt.objectid
FROM sys.dm_exec_query_stats qs
```

```
cross apply sys.dm_exec_sql_text (qs.sql_handle) as qt
ORDER BY [Avg IO] DESC
```

쿼리 최적화 프로그램, 쿼리 실행계획과 통계

SQL Server 2005 쿼리 최적화 프로그램은 사용자의 쿼리를 실행하기 위한 쿼리 실행계획을 컴파일합니다. 컴파일은 SQL Server 쿼리 최적화 프로그램이 데이터를 조회하거나 변경하기 위한 최소 비용 전략을 수립하는 작업입니다. 쿼리 실행계획에는 쿼리를 실행하기 위해 수행해야 하는 일련의 단계와 전략이 포함되어 있습니다.

SQL Server 2005 쿼리 최적화 프로그램, 쿼리 실행계획, 통계 정보에 대해서는 다음 링크의 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/qrystats.mspx.)

쿼리 실행계획 재사용 관련 DMV

대량의 트랜잭션을 처리하는 OLTP 어플리케이션 환경에서는 높은 쿼리 실행계획 재사용률을 유지하는 것이 중요합니다. 실행계획을 재사용하면, 쿼리가 실행될 때마다 동일한 실행계획을 위한 최적화 단계를 수행하기 위해 소비하는 CPU 비용을 절감할 수 있습니다. 다음 코드를 실행하면, 관련 DMV 를 사용하여, 실행계획 재사용률이 가장 낮은 쿼리를 찾아낼 수 있습니다.

```
--- 가장 낮은 실행계획 재사용률을 나타내는 쿼리를 반환
---
SELECT TOP 50
        qs.sql_handle
        ,qs.plan_handle
        ,cp.cacheobjtype
        ,cp.usecounts
        ,cp.size_in_bytes
        ,qs.statement_start_offset
        ,qs.statement_end_offset
        ,qt.dbid
        ,qt.objectid
        ,qt.text
        ,SUBSTRING(qt.text,qs.statement_start_offset/2,
             (case when qs.statement_end_offset = -1
            then len(convert(nvarchar(max), qt.text)) * 2
            else qs.statement_end_offset end -qs.statement_start_offset)/2)
```

```
        as statement
FROM sys.dm_exec_query_stats qs
cross apply sys.dm_exec_sql_text(qs.sql_handle) as qt
inner join sys.dm_exec_cached_plans as cp on qs.plan_handle=cp.plan_handle
where cp.plan_handle=qs.plan_handle
and qt.dbid = db_id()    ----- 조회 대상 database ID 를 지정
ORDER BY [Usecounts] ASC
```

쿼리 실행계획 재사용 관련 성능 카운터

성능 모니터의 SQL Server:SQL Statistiscs 개체에는 실행계획 재사용을 진단하기 위한 성능카운터가 포함되어 있습니다. 전체 배치 요청 대비 초기 컴파일 비율을 비교함으로써 매우 유용한 정보를 얻을 수 있습니다. 실행계획이 플랜 캐시에 존재하지 않는 경우에 초기 컴파일이 발생합니다. OLTP 어플리케이션에서는 실행계획 재사용률이 90% 이상으로 유지되어야 합니다.

초기 컴파일 = SQL Compilations/sec – SQL Re-Compilations/sec

실행계획 재사용률 = (Batch requests/sec – Initial Compilations/sec) / Batch requests/sec

메모리 압박이 쿼리 실행계획을 캐시에서 제거하는 원인이 될 수 있으며, 이로 인해 실행계획 재사용률이 저하될 수 있습니다. 메모리 압박의 증상 섹션을 참고하십시오.

문 단위 재컴파일 관련 DMVs

SQL Server 2005 에서는 저장 프로시저에 포함된 개별 문 단위로 재컴파일될 수 있습니다. 문 단위 재컴파일은 매우 효율적입니다. 예를 들어, MyTable 이라는 테이블을 만들고, 데이터를 입력한 다음, MyTable 과 다른 테이블을 조인하는 쿼리를 실행하는 저장 프로시저가 있다고 가정합니다. MyTable 를 만들고 데이터를 입력하는 작업은 초기 컴파일이후에 실행되는 작업이기 때문에, MyTable 의 행의 최종 크기와 행 수는 실행시점이 될 때까지 알 수 없습니다. MyTable 과 다른 테이블을 조인하는 시점에는 MyTable 에 1 백만행 이상이 저장되어 있을 수도 있습니다. SQL Server 에서는 MyTable 에 대한 통계정보를 조회하고, MyTable 에 대한 새로운 통계 정보를 반영하기 위해 조인 명령을 재컴파일합니다. 좀 더 자세한 정보는 SQL Server 2005 최적화 프로그램과 통계정보 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/qrystats.mspx)

재컴파일이 발생하는 것이 항상 바람직한 것은 아닙니다. 예를 들어, 재컴파일한 실행계획이 원본 실행계획이 동일한 경우라면, 재컴파일은 불필요한 절차가 됩니다. 이를 확인하기 위해서, 재컴파일되는 문에 대한 정보를 확인해야 합니다. SQL Server 2005 재컴파일 이슈에 대한 좀 더 자세한 정보는 다음 링크의 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx)

다음 스크립트는 재컴파일이 발생한 문장을 반환합니다.

```
---- Recompilation and SQL.sql
---- plan_generation_num 칼럼과 SQL 문의 관계
---- 문이 재컴파일이 발생할 때마다, plan_generation_num 칼럼의 값이 증가
----
select top 25
    --sql_text.text,
    sql_handle,
    plan_generation_num,
    substring(text,qs.statement_start_offset/2,
              (case when qs.statement_end_offset = -1
             then len(convert(nvarchar(max), text)) * 2
             else qs.statement_end_offset end - qs.statement_start_offset)/2)
        as stmt_executing,
    execution_count,
    dbid,
    objectid
from sys.dm_exec_query_stats as qs
    Cross apply sys.dm_exec_sql_text(sql_handle) sql_text
where plan_generation_num >1
order by sql_handle, plan_generation_num
```

차단 이슈 원인 추적

SQL Server 2005 프로파일러와 sp_configure 저장 프로시저를 사용하여, 오랫동안 지속되는 차단현상을 추적할 수 있습니다. sp_configure 저장 프로시저를 사용하여, blocked process threshold 옵션 값을 설정하면, 설정된 값(초 단위)보다 오랫동안 차단현상이 발생한 프로세스에 대한 정보가 이벤트로 보고됩니다. 'blocked process threshold' 옵션 값을 너무 낮게 설정하게 되면, 너무 많은 이벤트가 수집될 수 있기 때문에 유의해야 합니다. 'blocked process threshold' 옵션 값을 적절하게 설정한 다음, Errors and Warnings 개체의 Blocked Process Report 이벤트를 사용하여 차단주체가 되는 프로세스와 차단된 프로세스를 추적할 수 있습니다.

차단 문제가 발생하고 있는 주요 개체를 찾아내기 위해, 다음의 코드를 실행하면, 가장 많은 차단현상이 발생하고 있는 테이블과 인덱스 목록을 반환합니다.

```
----행 잠금 대기를 검색
declare @dbid int
select @dbid = db_id()
Select dbid=database_id, objectname=object_name(s.object_id)
```

```
, indexname=i.name, i.index_id  --, partition_number
, row_lock_count, row_lock_wait_count
, [block %]=cast (100.0 * row_lock_wait_count / (1 + row_lock_count) as numeric(15,2))
, row_lock_wait_in_ms
, [avg row lock waits in ms]=cast (1.0 * row_lock_wait_in_ms / (1 + row_lock_wait_count) as
numeric(15,2))
from sys.dm_db_index_operational_stats (@dbid, NULL, NULL, NULL) s,          sys.indexes i
where objectproperty(s.object_id,'IsUserTable') = 1
and i.object_id = s.object_id
and i.index_id = s.index_id
order by row_lock_wait_count desc
```

평균 차단 시간은 밀리초 단위로 반환됩니다. sp_configure 저장 프로시저를 사용하여, 'blocked process threshold' 옵션을 설정할 때에는, 이 값을 초 단위로 변환하여 지정해야 합니다. 'blocked process threshold' 옵션을 얼마로 설정해야 할 것인지에 대한 명확한 기준이 없는 상태에서는 위의 코드를 실행한 결과 반환되는 평균 차단 시간을 참조로 활용할 수 있습니다. 하지만, 'blocked process threshold' 옵션값을 너무 낮게 설정하면, 문제가 될 수 있다는 것을 다시 한 번 기억해야 합니다. 'blocked process threshold' 옵션으로 설정된 값을 초과하는 모든 명령이 Blocked Process Report 추적 이벤트로 발생하기 때문입니다.

대기자 리스트에 있는 SQL 문 조회

get_statements_in_waiter_list 저장 프로시저는 생략가능한 매개변수인 @wait_type 을 입력받아, 입력한 매개변수와 일치하는 대기유형의 대기자 리스트을 조회합니다. @wait_type 매개변수에 NULL 값이 입력되면, 대기자 리스트에 있는 모든 SQL 문이 조회됩니다. 대기자 리스트에 있는 SQL 문을 조회하기 위해서, 언제라도 **get_statements_in_waiter_list** 저장 프로시저를 실행할 수 있습니다. 예를 들어, 병렬처리와 관련한 대기유형에 해당하는 대기자 리스트을 조회하기 위해서는, 다음과 같은 명령을 실행하면 됩니다.

```
Exec get_statements_in_waiter_list @wait_type = 'CXPACKET'
```

get_statements_in_waiter_list 저장 프로시저에 대한 스크립트는 다음 링크에서 확인할 수 있습니다.
http://www.microsoft.com/technet/scriptcenter/scripts/sql/sql2005/sqlos/sql05vb033.mspx
http://www.microsoft.com/technet/scriptcenter/scripts/sql/sql2005/waitstats/default.mspx. 각 대기유형에 대해서는 "SQL Server 2005 대기유형" 섹션을 참조하십시오.

SQL Server 2005 대기 유형과 관련 성능 정보

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
ASYNC_DISKPOOL_LOCK	IO	True	파일을 만들거나 파일에 대한 초기화 작업을 수행하는 동안 병렬 스레드를 동기화하려고 시도하는 경우에 발생합니다. SQL Server 2000: 백업 및 복원 작업 스레드가 병렬로 작업을 수행하는 동안(예를 들어, 페이지를 0 으로 초기화하는 작업하는 동안) 발생 SQL Server 2005: 더 이상 복원작업을 수행하기 전에 데이터 파일을 초기화(예를 들어, 0 으로 초기화)하는 작업을 수행하지 않음	디스크 병목현상이 발생하고 있을 가능성이 있습니다. 디스크 관련 성능카운터의 수치를 확인하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
ASYNC_IO_COMPLETION	IO	True	작업이 비동기 I/O 가 완료될 때까지 대기하는 경우에 발생합니다. 성능 모니터 카운터, 프로파일러, sys.dm_io_virtual_file_stats, SHOWPLAN 옵션 등을 사용하여 디스크 병목현상이 발생하고 있는지 점검해야 합니다. ASYNC_IO_COMPLETION 대기유형을 줄이기 위해서는 다음의 작업을 수행해야 합니다. 1. 추가적인 IO 대역폭을 추가 2. 각 디스크 드라이브간의 IO 를 균등하게 분산 3. 적절한 인덱스 전략을 통해, IO 를 줄여줌 4. 잘못된 쿼리 실행계획이 존재하는지 점검 5. 메모리 압박이 존재하는지 점검	"메모리 압박과 디스크 IO 서브시스템 이슈" 섹션을 참조하십시오. 디스크의 병목현상이 존재하는지 점검하기 위해, 성능모니터의 Physical Disk 성능 카운터를 점검하십시오.: 1. Disk sec/read 2. Disk sec/write 3. Disk queues 메모리 압박성능 현상이 존재하는지 점검하기 위해, 성능모니터의 SQLServer:Buffer Manager 성능 카운터를 점검하십시오.: 1. Page Life Expectancy 2. Checkpoint pages/sec 3. Lazy writes/sec 적절한 인덱스 전략이 수립되었는지 점검하기 위해, 성능모니터의 SQLServer: Access Methods 성능 카운터를 점검하십시오: 1. Full Scans/sec 2. Index seeks/sec SQL 프로파일러를 실행하여, 스캔을 발생시키는 T-SQL 문을 찾아냅니다. 프로파일러에서 scans 이벤트 클래스의 scan:stopped 이벤트를 선택합니다. 데이터 칼럼 탭에서 objectid 칼럼을 추가합니다. 추적을 실행합니다. 프로파일러 추적 결과를 추적 테이블에 저장하고, scans 이벤트를 조사합니다. 또는, duration, reads, writes 칼럼 값이 높게 나타나는 이벤트를 조사합니다. 잘못된 쿼리 실행계획이 존재하는지 점검하기 위해 SHOWPLAN 옵션을 점검하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
ASYNC_NETWORK_IO 새로 추가	Network	True	작업이 네트워크로 인해 차단되고 있는 상황에서, 네트워크 상에 존재하는 리소스에 쓰기 작업을 수행하려고 시도하는 발생합니다. 클라이언트가 서버의 데이터를 처리하고 있는지 점검하십시오.	네트워크 어댑터의 대역폭을 점검하십시오. 100 Mbit 보다는 1 Gbit 이 더 좋습니다. 10 Mbit 보다는 100 Mbit 이 더 좋습니다.
BACKUP 새로 추가	Backup	True	백업 프로세스의 일부에 의해서 작업이 차단되었을 때 발생합니다.	
BACKUP_CLIENTLOCK 새로 추가	Backup	True	내부 전용	
BACKUP_OPERATOR 새로 추가	Backup	True	작업이 백업 테이프가 탑재될 때까지 대기할 때 발생합니다. 백업 테이프의 상태를 확인하기 위해서, sys.dm_io_backup_tapes DMV 를 사용할 수 있습니다. 탑재 작업이 보류된 상태가 아니라면, 백업 테이프 드라이브에 하드웨어적인 문제가 있음을 나타내고 있는 것입니다.	백업 테이프 드라이브 점검
BACKUPBUFFER 새로 추가	Backup	True	백업 작업이 데이터를 기다리거나, 데이터를 저장할 버퍼를 기다리는 경우에 발생합니다. 이 유형은 백업 테이프 탑재를 대기하는 경우를 제외하고는 일반적인 경우는 아닙니다.	백업 테이프 드라이브 점검

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
BACKUPIO	Backup	True	백업 작업이 데이터를 기다리거나, 데이터를 저장할 버퍼를 기다리는 경우에 발생합니다. 이 유형은 백업 테이프 탑재를 대기하는 경우를 제외하고는 일반적인 경우는 아닙니다.	백업 테이프 드라이브 점검
BACKUPTHREAD	Backup	True	작업이 백업이 완료될 때까지 대기하는 경우에 발생합니다. 대기 시간은 몇 분에서부터 몇 시간이 소요될 가능성이 있습니다. 대기 중인 작업이 I/O 프로세스 상태에 있는 경우에는, 이 유형은 크게 문제가 되지 않습니다.	
BAD_PAGE_PROCESS	Memory	True	손상된(suspect) 페이지가 대량으로 발생한 경우, 백그라운드로 동작하는 손상된 페이지 로그기록기(suspect page logger)가 매 5 초보다 더 긴 간격으로 실행되는 것을 방지하려고 시도하는 경우에 발생합니다.	손상된(suspect) 페이지는 msdb 데이터베이스의 dbo.suspect_pages 시스템 테이블에 저장됩니다. 손상된 페이지는 페이지 수준 온라인 복원을 통해서 복원될 수 있습니다.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
BROKER_CONNECTION_RECEIVE_TASK 새로 추가	Service Broker	False	연결 끝점(endpoint)에서 메시지를 수신하기 위한 액세스를 대기하는 경우에 발생합니다. 연결 끝점에 대한 메시지 수신 액세스는 직렬화(serialized) 됩니다.	
BROKER_ENDPOINT_STATE_MUTEX 새로 추가	Service Broker	False	Service broker 연결 끝점의 상태에 액세스하기 위한 경합현상이 존재하는 경우에 발생합니다. 변경 내용의 상태에 대한 액세스는 직렬화(serialized) 됩니다.	
BROKER_EVENTHANDLER 새로 추가	Service Broker	False	작업이 Service broker 의 기본 이벤트 처리기 에서 대기하는 경우에 발생합니다. 이 대기 유형은 반드시 매우 짧게 나타나야 합니다.	
BROKER_INIT 새로 추가	Service Broker	False	각 활성 데이터베이스에서 Service Broker 를 초기화하는 경우에 발생합니다. 이 대기유형은 매우 드물게 나타나야 합니다.	
BROKER_MASTERSTART 새로 추가	Service Broker	False	Service broker 를 시작하기 위해 기본 이벤트 처리기 작업이 대기하고 있는 경우에 발생합니다. 이 대기 유형은 반드시 매우 짧게 나타나야 아 합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
BROKER_RECEIVE_WAITFOR 새로 추가	Service Broker	True	RECEIVE WAITFOR 명령에 의해서 대기중인 경우에 발생합니다. 주로 수신할 준비가 된 메시지가 존재하지 않는 경우에 발생합니다.	
BROKER_REGISTERALLENDPOINTS 새로 추가	Service Broker	False	Service Broker 연결 끝점을 초기화하는 동안 발생합니다. 이 대기 유형은 반드시 매우 짧게 나타나야 아합니다.	
BROKER_SHUTDOWN	Service Broker	False	계획된 Service Broker 종료시 발생합니다. 이 대기 유형은 반드시 매우 짧게 나타나야 아합니다.	
BROKER_TRANSMITTER 새로 추가	Service Broker	False	Service Broker 메시지 전송기가 작동될 때까지 대기하는 경우에 발생합니다.	
BUILTIN_HASHKEY_MUTEX 새로 추가		True	인스턴스를 시작한 다음에, 내부 데이터 구조를 초기화하는 동안 발생할 수 있습니다. 데이터 구조를 초기화한 다음에는 다시 발생하지 않습니다.	
CHECKPOINT_QUEUE		False	체크포인트 작업이 다음 번 체크포인트 요청이 있기까지 대기하는 경우에 발생합니다.	체크포인트는 변경된 데이터 페이지와 로그 페이지를 디스크에 기록합니다. 디스크에 문제가 발생하고 있는지 점검하십시오. 성능모니터의 Physical Disk performance 성능카운터를 점검하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
CHKPT		True	서버가 시작되는 동안 체크포인트 스레드를 시작할 수 있다는 것을 알려주기 위해서 발생합니다.	
CLR_AUTO_EVENT 새로 추가	CLR	True	CLR 실행 작업이 수행되는 동안, 특정 자동 이벤트가 시작될 때까지 대기하는 경우에 발생합니다.	
CLR_CRST 새로 추가	CLR	True	CLR 실행 작업이 수행되는 동안, 현재 다른 작업에 의해서 사용중인 중요한(critical) 섹션을 시작하기 위해 대기하는 경우에 발생합니다.	
CLR_JOIN 새로 추가	CLR	True	CLR 실행 작업이 수행되는 동안, 다른 작업이 종료될 때까지 대기하는 경우에 발생합니다. 이 대기 유형은 각 작업간의 조인이 존재하는 경우에 발생합니다.	
CLR_MANUAL_EVENT 새로 추가	CLR	True	CLR 실행 작업이 수행되는 동안, 특정 수동이벤트가 시작될 때까지 대기하는 경우에 발생합니다.	
CLR_MONITOR 새로 추가	CLR	True	CLR 실행 작업이 수행되는 동안, 모니터링 잠금을 획득하기 위해서 대기하는 경우에 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
CLR_RWLOCK_READER 새로 추가	CLR	True	CLR 실행 작업이 수행되는 동안, 판독기(reader) 잠금을 획득하기 위해 대기하는 경우에 발생합니다.	
CLR_RWLOCK_WRITER 새로 추가	CLR	True	CLR 실행 작업이 수행되는 동안, 기록기(writer) 잠금을 획득하기 위해서 대기하는 경우에 발생합니다.	
CLR_SEMAPHORE 새로 추가	CLR	True	CLR 실행 작업이 수행되는 동안, 세마포어(semaphore) 를 위해 대기하는 경우에 발생합니다.	
CLR_TASK_START 새로 추가	CLR	False	CLR 작업 시작이 완료될 때까지 대기하는 경우에 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
CMEMTHREAD	Memory	True	작업이 스레드-안정성 메모리 개체를 획득하기 위해 대기하는 경우에 발생합니다. 여러 개의 작업이 동일한 메모리 개체로부터 메모리를 할당받고자 경합하는 현상이 발생하면, 대기시간이 증가할 수 있습니다.	사용자가 메모리 개체로부터 메모리를 할당받거나, 해제하는 작업을 수행하는 동안에는 직렬화(serialization)가 보장됩니다. 하나의 작업을 수행하는 동안 동일한 작업을 수행하는 나머지 서버 프로세서 ID(SPID)는 대기해야 합니다. 이러한 경우, 대기하는 SPID 에는 CMEMTHREAD 대기유형이 설정됩니다. 이 대기유형이 나타나는 다양한 시나리오가 있을 수 있습니다. Ad hoc 쿼리 실행계획이 매우 많은 연결로부터 실행되어, 매우 빨리 프로시저 캐시로 등록되는 경우에 발생할 수 있습니다. 이러한 병목현상은 쿼리 실행계획을 재사용할 수 있도록 쿼리를 명시적으로 매개변수화 하거나, 적절하게 저장 프로시저로 전환하는 것을 통해, 프로시저 캐시에 등록되거나 제거되는 데이터를 제한함으로써 해결할 수 있습니다.
CURSOR		True	비동기 커서 스레드	
CURSOR_ASYNC		True	내부 전용	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
CXPACKET		True	쿼리 프로세서에서 교환(exchange) 반복기의 처리를 동기화하려고 하는 경우에 발생합니다. 이 대기 유형의 경합이 문제가 되는 상황이라면, 병렬처리수준 옵션(degree of parallelism)을 더 낮은 값으로 설정할 수 있습니다. 병렬처리에 관련된 대기는 때로 데이터가 왜곡되어 있을 때 발생할 수 있습니다. 이러한 경우, 특정 병렬처리 스레드에서는 좀 더 많은 행을 처리하고, 다른 병렬처리 스레드에서는 더 적은 행을 처리하는 문제가 발생할 수 있습니다. OLTP 환경에서 CXPACKET 대기가 과도하게 발생하면, 다른 OLTP 트래픽의 처리량에 영향을 미칠 수 있습니다. 데이터웨어하우스 환경에서는, 다중 처리 환경을 위해 CXPACKET 대기가 필요할 수 있습니다.	sp_Configure "max degree of parallelism"을 사용하여, 병렬처리 옵션을 점검하십시오. max degree of parallelism 옵션이 0 으로 설정되어 있다면, 다음의 대안 중 하나를 채택해 볼 것을 고려할 수 있습니다. 1. 모두 OLTP 작업부하 상태라면, max degree of parallelism 옵션을 1 로 설정하여, 병렬처리를 완전히 비활성화할 수 있습니다. 2. max degree of parallelism 옵션을 설정하여, 전체 CPU 개수보다 적은 수의 CPU 만 병렬처리에 사용될 수 있도록 제한할 수 있습니다. 예를 들어, CPU 가 8 개인 서버에서 max degree of parallelism 옵션을 4 이하로 설정할 수 있습니다.
DBMIRROR_DBM_EVENT 새로 추가	DBM	True	내부 전용	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
DBMIRROR_DBM_MUTEX 새로 추가	DBM	True	내부 전용	
DBMIRROR_EVENTS_QUEUE 새로 추가	DBM	False	데이터베이스 미러링이 이벤트가 처리될 때까지 대기하는 경우에 발생합니다.	
DBMIRROR_SEND 새로 추가	DBM	True	작업이 네트워크 계층의 통신 백로그가 지워져서 메시지를 보낼 수 있을 때까지 대기하는 경우에 발생합니다. 통신 계층에 과부하가 생겨 데이터베이스 미러링 처리량에 영향을 주기 시작했음을 나타냅니다.	
DBMIRROR_WORKER_QUEUE 새로 추가	DBM	False	데이터베이스 미러링 작업자 스레드가 추가 작업을 대기하는 경우에 발생합니다.	
DBMIRRORING_CMD 새로 추가	DBM	True	로그 레코드가 디스크로 플러시 될 때까지 대기하는 경우에 발생합니다.이 대기 유형은 오랜 시간 동안 발생할 가능성이 있습니다.	
DBTABLE			내부 전용 새로운 체크포인트 요청이 먼저 발생한 체크포인트 요청이 완료될 때까지 기다리는 경우에 발생합니다.	SQL Buffer Manager 성능 카운터를 점검하십시오: 1. Page Life Expectancy 2. Checkpoint pages/sec 3. Lazy writes/sec

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
DEADLOCK_ENUM_MUTEX	Lock	True	데드락 모니터와 sys.dm_os_waiting_tasks DMV 가 SQL Server 에서 동시에 여러 개의 데드락 탐색 작업을 수행하지 않도록 보장하기 위해서 발생합니다.	
DEADLOCK_TASK_SEARCH	Lock	True	이 대기 유형에 대한 대기 시간이 크게 나타나면, 서버에 sys.dm_os_waiting_tasks DMV 와 관련된 쿼리가 실행되고 있다는 것을 나타내며, 이러한 유형의 쿼리가 데드락 탐색 작업을 실행하는 동안에는 데드락 모니터가 실행되지 못하도록 차단하게 됩니다. (특정 시점에, 쿼리 또는 데드락 모니터 둘 중 하나만 데드락 탐색 작업을할 수 있습니다.). sys.dm_os_waiting_tasks 쿼리에서는 DEADLOCK_ENUM_MUTEX 대기유형을 사용하고, 데드락 모니터는 DEADLOCK_TASK_SEARCH 대기유형을 사용합니다.	
DEBUG		True	T-SQL 및 CLR 디버깅 작업을 수행하는 동안 내부 동기화를 위해 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
DISABLE_VERSIONING 새로 추가		True	SQL Server 에서 가장 오래된 활성 트랜잭션의 타임스탬프가 상태변경이 시작될 때의 타임스탬프보다 나중인지 확인하기 위해 버전 관리 트랜잭션 관리자를 폴링하는 경우에 발생합니다. 이러한 경우에서는, 모든 스냅숏 트랜잭션이 실행된 ALTER DATABASE 명령이 실행되기 전에 시작된 스냅숏 트랜잭션은 모두 완료된 상태이어야 합니다. 이 대기 유형은 SQL Server 에서 ALTER DATABASE 명령을 사용하여 행 버전 관리 기능을 비활성화하는 경우에 사용됩니다.	
DISKIO_SUSPEND	IO	True	외부 백업이 활성화된 상태에서, 파일에 액세스 하기 위해 대기하는 작업이 존재할 때 발생합니다. 이는 대기중인 각 사용자 프로세스에 대해서 보고됩니다. 사용자 프로세스마다 5 이상의 대기 수가 나타나면, 외부 백업 작업을 완료하는데 너무 많은 시간이 소요된다는 것을 나타냅니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
DLL_LOADING_MUTEX	XML	False	XML 파서 DLL 가 로드될 때까지 대기하는 경우에 발생합니다.	
DROPTEMP		True	이전에 시도가 실패한 상태에서, 임시 개체를 다시 한 번 삭제하고자 시도하는 사이에 발생합니다. 이 대기 유형의 대기시간은 임시 개체에 대한 삭제 시도가 증가할 때마다 지수로 증가하게 됩니다.	
DTC	DTC	True	작업이 상태 변이를 관리하기 위해서 사용되는 이벤트에 대해 대기하는 경우에 발생합니다. 이 대기 상태는 MS DTC(Microsoft Distributed Transaction Coordinator) 서비스가 가용하지 않은 상태가 되었다는 알림을 수신한 이후에, MS DTC 트랜잭션을 복구하는 경우에 발생합니다. 또한, 이 대기 상태는 MS DTC 트랜잭션의 커밋 작업이 SQL Server 에 의해 초기화된 다음, SQL Server 가 MS DTC 트랜잭션의 커밋 작업이 완료될 때까지 대기하고 있는 경우를 나타냅니다. MS DTC 관련 대기를 나타냅니다.	트랜잭션 격리수준 점검

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
DTC_ABORT_REQUEST	DTC	True	MS DTC 작업자 세션이 MS DTC 트랜잭션에 대한 소유권을 얻기 위해서 대기하는 경우에 발생합니다. MS DTC 가 해당 트랜잭션에 대한 소유권을 얻은 다음에는, 필요에 따라 해당 트랜잭션을 롤백할 수 있습니다. 일반적으로 MS DTC 작업자 세션은 해당 트랜잭션을 사용하고 있는 다른 세션에 대해 대기합니다.	
DTC_RESOLVE	DTC	True	여러 개의 데이터베이스에 걸쳐진 트랜잭션내에서 트랜잭션의 결과에 쿼리할 수 있도록 하기 위해서, 복구 작업이 master 데이터베이스에 대해 대기하는 경우에 발생합니다.	
DTC_STATE	DTC	True	내부적으로 관리되는 MS DTC 전역 상태 개체의 상태를 변경하는 작업을 보호하기 위한 이벤트에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 매우 짧은 기간 동안만 나타나야 합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
DTC_TMDOWN_REQUEST	DTC	True	SQL Server 에서 MS DTC 서비스가 사용할 수 없게 되었다는 알림을 받는 경우에 MS DTC 작업자 세션에서 발생합니다. 먼저 작업자는 MS DTC 복구 프로세스가 시작될 때까지 기다렸다가 작업자가 작업하고 있는 분산 트랜잭션의 결과물을 획득하기 위해 대기합니다. 이것은 MS DTC 서비스와의 연결이 다시 설정될 때까지 계속될 수 있습니다.	
DTC_WAITFOR_OUTCOME	DTC	True	복구 작업이 준비된(prepared) 트랜잭션의 결과를 활성화하기 위해서 MS DTC 가 활성 상태가 되기까지 대기하는 경우에 발생합니다.	
DUMP_LOG_COORDINATOR		True	주 작업이 데이터를 생성하기 위한 하위 작업에 대해 대기하는 경우에 발생합니다. 일반적인 상황에서는, 이 대기 유형은 발생하지 않을 수도 있습니다. 이 대기 유형이 오랫동안 발생한다는 것은 예상치 못한 차단이 발생하고 있다는 것을 의미합니다. 하위작업에 대해서 반드시 확인해 보아야 합니다.	
EC			내부전용	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
EE_PMOLOCK		True	명령을 실행하는 동안 메모리 할당 작업을 동기화하기 위해서 발생합니다.	
EE_SPECPROC_MAP_INIT		False	내부적인 해시 테이블 생성을 동기화하기 위해서 발생합니다. 이 대기 유형은 SQL Server 2005 인스턴스가 시작된 이후 해시 테이블에 맨 처음 액세스하는 경우에만 발생합니다.	
ENABLE_VERSIONING		True	SQL Server 에서 ALTER DATABASE 명령을 사용하여 특정 데이터베이스를 스냅숏 격리수준을 활성화한 상태로 변경하기 전에, 해당 데이터베이스 내에서 발생한 모든 변경 트랜잭션이 완료되기까지 대기하는 경우에 발생합니다.	
ERROR_REPORTING_MANAGER				

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
EXCHANGE		True	병렬처리 쿼리를 실행하는 동안, 쿼리 프로세서 교환(exchange) 연산자 내부의 동기화를 위해 대기하는 경우에 발생합니다.	sp_Configure "max degree of parallelism"을 사용하여, 병렬처리 옵션을 점검하십시오. max degree of parallelism 옵션이 0 으로 설정되어 있다면, 다음의 대안 중 하나를 채택해 볼 것을 고려할 수 있습니다. 4. 모두 OLTP 작업부하 상태라면, max degree of parallelism 옵션을 1 로 설정하여, 병렬처리를 완전히 비활성화할 수 있습니다. 5. max degree of parallelism 옵션을 설정하여, 전체 CPU 개수보다 적은 수의 CPU 만 병렬처리에 사용될 수 있도록 제한할 수 있습니다. 예를 들어, CPU 가 8 개인 서버에서 max degree of parallelism 옵션을 4 이하로 설정할 수 있습니다.
EXECSYNC		True	병렬 쿼리가 실행되는 동안, 교환 연산자에 의해서 처리되지 않는 영역에 대해 동기화 작업을 수행하는 경우에 발생합니다. 교환 연산자에 의해서 처리되지 않는 영역에는 비트맵(Bitmap), 대량 바이너리 개체(BLOBs)와 스풀(Spool) 연산자 등이 포함됩니다. LOB 를 처리하기 위해 이 대기 유형을 자주 사용되게 됩니다. 비트맵이나 스풀 연산자는 경합현상의 원인이 되지 않습니다.	
Failpoint		True	내부전용	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
FCB_REPLICA_READ		True	스냅숏 또는 DBCC 명령에 의해 생성되는 임시 스냅숏 스파스(sparse) 파일에 대한 읽기 작업을 동기화하는 경우에 발생합니다.	
FCB_REPLICA_WRITE		True	스냅숏 또는 DBCC 명령에 의해 생성되는 임시 스냅숏 스파스(sparse) 파일에 페이지를 밀어넣기하거나 끌어오기하는 작업을 동기화하는 경우에 발생합니다.	
FT_RESTART_CRAWL		True	일시적인 장애상황으로 인해, 마지막 성공 시점으로부터 전체 텍스트의 데이터 탐색 작업을 다시 시작해야 하는 경우에 발생합니다. 이 대기 유형은 현재 수행 중인 데이터 집적 작업을 현재 단계에서 완료하고 종료하기 위해서 발생합니다.	
FT_RESUME_CRAWL		True	전체 텍스트의 데이터 집적 작업을 잠시동안 중지하기 위해, 기존의 작업이 완료하기까지 대기하는 경우에 발생합니다.	
HTTP_ENDPOINT_COLLCREATE		True	내부전용	
HTTP_ENUMERATION		True	시스템이 시작될 때, HTTP 를 시작할 HTTP 끝점을 열거하기 위해 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
IMP_IMPORT_MUTEX		True	내부전용	
IMPPROV_IOWAIT		True	SQL Server 에서 대량로드 I/O 가 완료될 때까지 대기하는 경우에 발생합니다.	
INDEX_USAGE_STATS_MUTEX		True	내부전용	
IO_AUDIT_MUTEX		True	추적 이벤트 버퍼를 동기화하는 동안에 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
IO_COMPLETION		True	IO 작업이 완료될 때까지 대기하는 동안에 발생합니다. 이 대기 유형은 주로 비-데이터 페이지 I/O 를 표현합니다. 데이터 페이지 I/O 완료에 대한 대기 유형은 PAGEIOLATCH_* 대기유형으로 나타납니다. 성능 모니터, 프로파일러, sys.dm_io_virtual_file_stats DMV, SHOWPLAN 옵션 등을 사용하여 잘못된 쿼리 실행계획이 생성되었는지 여부를 점검합니다. 이 대기유형을 줄이기 위해서는 다음 대안을 고려하십시오. 1. IO 대역폭 추가 2. 드라이브간의 IO 균등 배분 3. 적절한 인덱스 전략을 통한, IO 경감 4. 잘못된 쿼리 실행계획여부 점검	디스크 관련 성능 카운터를 참조하십시오. 1. Disk sec/read 2. Disk sec/write 3. Disk queues SQL Buffer Manager 관련 성능 카운터를 참조하십시오. 1. Page Life Expectancy 2. Checkpoint pages/sec 3. Lazy writes/sec 인덱스 전략을 확인하기 위해 SQL Access Methods 관련 성능카운터를 참조하십시오. 1. Full Scans/sec 2. Index seeks/sec 메모리 관련 성능카운터를 참조하십시오 • Page faults/sec IO 병목현상이 발생하고 있는지 점검하기 위해 Io_Stalls 섹션을 참조하십시오. SQL 프로파일러를 사용하여 스캔을 발생시키는 T-SQL 구문을 찾을 수 있습니다. Scans 이벤트 클래스의 scan:started 이벤트와 scan:completed 이벤트로 추적할 수 있습니다. ObjectId 데이터 칼럼을 포함시킵니다. 프로파일러 추적을 추적 테이블로 저장하고, scans 관련 이벤트를 검색할 수 있습니다. scan:completed 이벤트는 높은 reads, writes, duration 가 발생하는 쿼리를 찾기 위해, 관련 IO 정보를 제공합니다. 잘못된 실행계획이 존재하는지 여부를 판단하기 위해 SHOWPLAN 옵션을 점검하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
KTM_ENLISTMENT		True	내부전용	
KTM_RECOVERY_MANAGER		True	내부전용	
KTM_RECOVERY_RESOLUTION		True	내부전용	
LATCH_x			래치(Latches)는 매우 짧은 기간 동안 발생하는 경량 동기화 개체입니다. 래치는 트랜잭션 지속시간동안 유지되지 않습니다. "일반적인(Plain)" 래치는 일반적으로 IO 와 관련되지 않습니다. 이러한 유형의 래치는 매우 다양한 용도로 사용되지만, 버퍼 페이지에 대한 액세스를 동기화하기 위해서는 사용되지 않습니다.(PAGELATCH_x 대기유형이 버퍼 페이지의 동기화를 위해 사용됩니다.) 주로 힙이나 텍스트에서 사용하는 내부 캐시 (버퍼 풀 페이지가 아닌)의 경합을 해결하기 위한 동기화 개체로 사용됩니다.	LATCH_x 에 대한 수치가 높게 나타난다면, 다음의 성능 모니터를 사용하여 다음 문제를 점검하십시오. 1. 메모리 압박 2. SQL Latch waits (ms) LOG 와 Pagelatch_UP 대기유형을 참조하십시오. Latch_x 대기유형은 주로 LOG 와 Pagelatch_UP 대기유형 경합현상을 해결함으로써 제거할 수 있습니다. LOG 또는 PAGELATCH_UP 경합현상이 존재하지 않는 경우라면, 여러 개의 캐시(인덱스마다 캐시가 존재)를 생성하기 위해서 분할 테이블/인덱스를 사용할 수 있습니다.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LATCH_DT		True	DT (destroy) 래치로 인해 대기하는 동안에 발생합니다. 이 대기 유형에는 버퍼 래치나 트랜잭션 표시 래치 등은 포함되지 않습니다. Latch_*에 대한 정보는 sys.dm_os_latch_stats DMV 를 통해 확인할 수 있습니다. sys.dm_os_latch_stats DMV 에서는 LATCH_NL, LATCH_SH, LATCH_UP, LATCH_EX, LATCH_DT 대기 유형을 함께 그룹화하여 표시해 줍니다.	LATCH_x 를 참조하십시오.
LATCH_EX		True	EX (exclusive)래치로 인해 대기하는 동안에 발생합니다. 이 대기 유형에는 버퍼 래치나 트랜잭션 표시 래치 등은 포함되지 않습니다. Latch_*에 대한 정보는 sys.dm_os_latch_stats DMV 를 통해 확인할 수 있습니다. sys.dm_os_latch_stats DMV 에서는 LATCH_NL, LATCH_SH, LATCH_UP, LATCH_EX, LATCH_DT 대기 유형을 함께 그룹화하여 표시해 줍니다.	LATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LATCH_KP		True	KP(Keep) 래치로 인해 대기하는 동안에 발생합니다. 이 대기 유형에는 버퍼 래치나 트랜잭션 표시 래치 등은 포함되지 않습니다. Latch_*에 대한 정보는 sys.dm_os_latch_stats DMV 를 통해 확인할 수 있습니다. sys.dm_os_latch_stats DMV 에서는 LATCH_NL, LATCH_SH, LATCH_UP, LATCH_EX, LATCH_DT 대기 유형을 함께 그룹화하여 표시해 줍니다.	LATCH_x 를 참조하십시오.
LATCH_NL		True	내부전용	LATCH_x 를 참조하십시오.
LATCH_SH		True	SH(share) 래치로 인해 대기하는 동안에 발생합니다. 이 대기 유형에는 버퍼 래치나 트랜잭션 표시 래치 등은 포함되지 않습니다. Latch_*에 대한 정보는 sys.dm_os_latch_stats DMV 를 통해 확인할 수 있습니다. sys.dm_os_latch_stats DMV 에서는 LATCH_NL, LATCH_SH, LATCH_UP, LATCH_EX, LATCH_DT 대기 유형을 함께 그룹화하여 표시해 줍니다.	LATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LATCH_UP		True	UP(Update) 래치로 인해 대기하는 동안 발생합니다. 이 대기 유형에는 버퍼 래치나 트랜잭션 표시 래치 등은 포함되지 않습니다. Latch_*에 대한 정보는 sys.dm_os_latch_stats DMV 를 통해 확인할 수 있습니다. sys.dm_os_latch_stats DMV 에서는 LATCH_NL, LATCH_SH, LATCH_UP, LATCH_EX, LATCH_DT 대기 유형을 함께 그룹화하여 표시해 줍니다.	LATCH_x 를 참조하십시오.
LAZYWRITER_SLEEP		True	Lazy writer 작업이 보류되는 동안에 발생합니다. 이 대기 유형의 시간 측정값은 백그라운드 작업으로서 대기한 시간을 의미합니다. 사용자 영역에서 이 대기유형이 나타난다면, 무시할 수 있습니다.	
LCK_x			발생가능한 트랜잭션 관리 이슈 1. 공유 잠금에 대해서는, 트랜잭션 격리수준을 점검하십시오. 2. 트랜잭션의 길이를 가능한 짧게 유지하십시오.	SQL Locks 성능 카운터를 점검하십시오. • Lock wait time (ms) 힌트: 물리적인 IO 를 발생시켜서 트랜잭션과 잠금의 지속시간을 길게 하는 원인이 되는 메모리 압박 현상이 발생하고 있는지 점검하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LCK_M_BU			대량 업데이트 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_IS			내재된 공유 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_IU			내재된 업데이트 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_IX			내재된 배타적 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LCK_M_RIn_NL			현재 키 값에 대한 NULL 잠금과, 현재 키 값과 이전 키 값 사이의 insert 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_RIn_S			현재 키 값에 대한 공유 잠금과 현재 키 값과 이전 키 값 사이의 insert 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_RIn_U			현재 키 값에 대한 업데이트 잠금과 현재 키 값과 이전 키 값 사이의 insert 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LCK_M_RIn_X		True	현재 키 값에 대한 배타적 잠금과 현재 키 값과 이전 키 값 사이의 insert 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_RS_S		True	현재 키 값에 대한 공유 잠금과 현재 키 값과 이전 키 값 사이의 공유 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_RS_U		True	현재 키 값에 대한 업데이트 범위 잠금과 현재 키 값과 이전 키 값 사이의 공유 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LCK_M_RX_S		True	현재 키 값에 대한 공유 잠금과 현재 키 값과 이전 키 값 사이의 배타적 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_RX_U		True	현재 키 값에 대한 업데이트 잠금과 현재 키 값과 이전 키 값 사이의 배타적 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_RX_X		True	현재 키 값에 대한 배타적 잠금과 현재 키 값과 이전 키 값 사이의 배타적 범위 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_S		True	공유 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LCK_M_SCH_M		True	스키마 변경 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_SCH_S		True	스키마 변경 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_SIU		True	내재된 업데이트 잠금과 함께 공유 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_SIX		True	내재된 배타적 잠금과 함께 공유 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_U		True	업데이트 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LCK_M_UIX		True	내재된 배타적 잠금과 함께 업데이트 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LCK_M_X		True	배타적 잠금을 얻기 위해 대기하는 경우에 발생합니다. 잠금 호환성 매트릭스를 확인하기 위해서는 sys.dm_tran_locks 토픽을 참조하십시오.	Lck_x 를 참조하십시오.
LOGBUFFER		True	로그 레코드를 로그 버퍼에 저장하기까지 대기하는 경우에 발생합니다. 이 대기 유형에 지속적으로 높은 수치로 유지되면, 로그를 저장하기 위한 디바이스가 현재 서버에서 발생하고 있는 트랜잭션 로그의 정보의 부하를 감당할 수 없다는 것을 의미합니다.	디스크 관련 성능 카운터를 점검하십시오: 1. Disk sec/read 2. Disk sec/write 3. Disk queues

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LOGMGR		True	데이터베이스를 닫는 동안, 트랜잭션 로그를 종료하기 전에, 발생한 트랜잭션 로그의 I/O 가 완료되기까지 대기하는 경우에 발생합니다. 성능 모니터 카운터, 프로파일러, sys.dm_io_virtual_file_stats, SHOWPLAN 옵션 등을 사용하여 디스크 병목현상이 발생하고 있는지 점검해야 합니다. LOGMGR 대기유형을 줄이기 위해서는 다음의 작업을 수행해야 합니다. 1. 추가적인 IO 대역폭 추가 2. 각 디스크 드라이브간의 IO 를 균등하게 분산 3. 트랜잭션 로그 파일을 전용 디스크 드라이브로 이동/격리	디스크 관련 성능 카운터를 점검하십시오: 1. Disk sec/read 2. Disk sec/write 3. Disk queues SQL Buffer Manager 관련 성능 카운터를 점검하십시오: 1. Page Life Expectancy 2. Checkpoint pages/sec 3. Lazy writes/sec 트랜잭션 로그 파일에 대한 Io_stall 값을 점검하십시오. • select * from sys.dm_io_virtual_file_stats(dbid,file#)
LOGMGR_FLUSH		True	내부 전용	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
LOGMGR_RESERVE_APPEND		True	로그 잘라내기 작업으로 인해, 새로운 로그 레코드를 저장하기 위한 로그 공간이 확보되었는지 여부를 확인하는 경우에 발생합니다. 이 대기유형을 줄이기 위해서, 관련된 데이터베이스의 트랜잭션 로그 파일의 크기를 증가시킬 것을 고려해 보아야 합니다.	
LOWFAIL_MEMMGR_QUEUE		True	메모리가 사용가능한 상태가 될까지 대기하는 경우에 발생합니다.	
MIRROR_SEND_MESSAGE		True	내부전용	
MISCELLANEOUS		True	모든 대기 유형에 관련	
MSQL_DQ		True	분산 쿼리 작업이 완료될 때까지 대기하는 경우에 발생합니다. 이 유형은 다중 활성 결과 집합(MARS) 어플리케이션의 데드락을 검색하기 위해서도 사용됩니다. 이 대기 유형은 분산 쿼리 호출 작업이 완료될 때 종료됩니다.	
MSQL_SYNC_PIPE		True	내부전용	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
MSQL_XACT_MGR_MUTEX		True	세션 트랜잭션 관리자의 소유권을 획득하여, 세션 수준 트랜잭션 작업을 수행하기 위해 대기하는 경우에 발생합니다.	
MSQL_XACT_MUTEX		True	세션 수준 트랜잭션의 처리를 동기화를 위해서 발생합니다. 세션 수준 트랜잭션 작업에 대한 요청은 트랜잭션을 사용하기 위해 반드시 뮤텍스(MUTEX)를 획득해야 합니다.	
MSQL_XP		True	확장 저장 프로시저의 실행이 완료될 때까지 대기하는 경우에 발생합니다. SQL Server 에서는 이 대기 유형을 잠재적인 MARS 어플리케이션 데드락 탐색 작업에서도 사용합니다. 이 대기 유형은 확장 저장 프로시저의 호출이 완료될 때 종료됩니다.	
MSSEARCH		True	전체 텍스트 검색 작업이 수행되는 동안에 발생합니다. 이 대기 유형은 전체 텍스트 작업이 완료될 때 종료됩니다. 이 대기 유형은 전체 텍스트 검색 작업에 경합현상이 발생하고 있는 것을 나타내는 것이 아니라, 전체 텍스트 작업의 지속시간을 나타냅니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
NET_WAITFOR_PACKET		True	네트워크 읽기 작업을 수행하는 동안 연결이 네트워크 패킷을 대기하는 경우에 발생합니다.	
OLEDB			SQL Server 가 Microsoft SQL Native Client OLE DB 공급자를 호출하는 경우에 발생합니다. 이 대기 유형은 동기화 작업을 위해 사용되지 않으며, 대신, OLE DB 공급자에 대한 호출의 지속시간을 나타냅니다. 이 대기 유형에는 다음과 같은 항목이 포함됩니다: 4 part name 호출, 원격 프로시저 호출, OPENQUERY, OPENROWSET 쿼리 등을 포함한 연결된 서버에 대한 호출 DMV 에 액세스하는 쿼리(DMV 에 액세스하는 쿼리는 OLE DB 행집합 공급자로서 구현되어 있기 때문) 대량 프로파일러 추적	1. 클라이언트에 존재하는 파일의 입력 작업, SQL Server 데이터 파일 또는 로그 파일에 대한 읽기 작업을 수행하는 클라이언트 어플리케이션이 존재하는지 점검하십시오. 성능 모니터의 disk sec/read , disk sec/write 성능 카운터를 점검하십시오. disk sec/read 성능 카운터가 높은 수치로 나타나면, IO 대역폭을 추가하거나, 각 디스크 드라이브의 IO 의 균형을 맞추거나, 해당 데이터베이스 파일이나 트랜잭션 로그 파일을 전용 디스크 드라이브로 이동/격리시기는 대책을 적용할 것을 고려해 보아야 합니다. 2. 원격프로시저 호출로 실행되는 T-SQL 코드, 분산 쿼리(연결된 서버 쿼리), 전체 텍스트 검색 쿼리를 조사하십시오. SQL Server 에서 이러한 유형의 쿼리를 지원하기는 하지만, 때로 이러한 유형의 쿼리가 성능 병목현상의 원인이 될 수 있습니다. 3. OLE DB 대기가 발생하고 있는 SQL 문을 조회하기 위해서는, "대기자 리스트에 있는 SQL 문 조회" 섹션을 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
PAGEIOLATCH_x			래치(Latches)는 매우 짧은 기간 동안 발생하는 경량 동기화 개체입니다. 버퍼 페이지에 대한 액세스를 동기화하기 위해 사용됩니다. PageIOLatch 는 디스크로부터 메모리로부터 데이터 페이지를 전송하는 과정에서 사용됩니다.	이 대기 유형이 전체 대기유형 중 높은 비율을 점유한다면, 디스크 IO 서브시스템에 문제가 발생하고 있다는 것을 의미합니다. 디스크 관련 성능 카운터를 점검하십시오.
PAGEIOLATCH_DT		True	I/O 요청 작업과 관련된 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 DT(Destory) 모드로 요청된 래치 요청에서 사용됩니다. 이 대기 유형에 대한 대기 시간이 길게 나타나는 경우, 디스크 서브시스템에 문제가 있다는 것을 나타냅니다.	PAGEIOLATCH_x 를 참조하십시오.
PAGEIOLATCH_EX		True	I/O 요청 작업과 관련된 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 배타적(Exclusive) 모드로 요청된 래치 요청에서 사용됩니다. 이 대기 유형에 대한 대기 시간이 길게 나타나는 경우, 디스크 서브시스템에 문제가 있다는 것을 나타냅니다.	PAGEIOLATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
PAGEIOLATCH_KP		True	I/O 요청 작업과 관련된 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 KEEP 모드로 요청된 래치 요청에서 사용됩니다. 이 대기 유형에 대한 대기 시간이 길게 나타나는 경우, 디스크 서브시스템에 문제가 있다는 것을 나타냅니다.	PAGEIOLATCH_x 를 참조하십시오.
PAGEIOLATCH_NL		True	내부전용	PAGEIOLATCH_x 를 참조하십시오.
PAGEIOLATCH_SH			I/O 요청 작업과 관련된 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 공유(SHARED) 모드로 요청된 래치 요청에서 사용됩니다. 이 대기 유형에 대한 대기 시간이 길게 나타나는 경우, 디스크 서브시스템에 문제가 있다는 것을 나타냅니다.	PAGEIOLATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
PAGEIOLATCH_UP			I/O 요청 작업과 관련된 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 업데이트(UPDATE) 모드로 요청된 래치 요청에서 사용됩니다. 이 대기 유형에 대한 대기 시간이 길게 나타나는 경우, 디스크 서브시스템에 문제가 있다는 것을 나타냅니다.	PAGEIOLATCH_x 를 참조하십시오.
PAGELATCH_x			래치(Latches)는 매우 짧은 기간 동안 발생하는 경량 동기화 개체입니다. 래치는 트랜잭션 지속시간동안 유지되지 않습니다. 주로 PAGELATCH_x 관련 래치 작업은 특정 행을 메모리로 전송하는 과정과, 행 오프셋 테이블에 대한 변경작업을 통제하기 위해서 사용됩니다. 그러므로, 래치의 지속시간은 주로 가용한 메모리 양에 민감하게 관련되어 있습니다.	이 대기 유형이 전체 대기유형 중 높은 비율을 점유한다면, 캐시에 대한 경합현상이 발생하고 있다는 것을 의미합니다.
PAGELATCH_DT		True	I/O 요청 작업과 관련되지 않은 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 DT(Destroy) 모드로 요청된 래치 요청에서 사용됩니다.	PAGELATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
PAGELATCH_EX		True	I/O 요청 작업과 관련되지 않은 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 배타적 모드로 요청된 래치 요청에서 사용됩니다. 이러한 경합현상은 IO 또는 메모리 성능이 아닌, 다른 이유 때문에 발생하게 됩니다. 예를 들어, 동일한 인덱스 범위에 동시 INSERT 작업이 수행되면, 이러한 경합현상이 발생할 수 있습니다. 대량의 INSERT 작업이 동일한 페이지에 대해 발생하게 되면, 이 래치 유형을 사용하여 각 INSERT 작업을 직렬화합니다. 또한, 동일한 범위에 대해 대량의 INSERT 작업이 발생하면, 래치가 설정된 상태의 인덱스 안에서 페이지 분할이 발생하여 새로운 페이지를 할당하는 작업(일정 시간이 소요됨)이 발생할 수 있습니다. 동일 페이지에 대한 INSERT 작업과 같이, 동일 범위에 대한 읽기 액세스가 발생하면 래치간에 충돌현상이 발생할 수 있습니다. 이러한 문제를 해결하기 위해서는 적절한 인덱스 전략을 통해 INSERT 작업을 분산시켜야 합니다.	PAGELATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
PAGELATCH_KP		True	I/O 요청 작업과 관련되지 않은 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 KEEP 모드로 요청된 래치 요청에서 사용됩니다.	PAGELATCH_x 를 참조하십시오.
PAGELATCH_NL		True	내부전용	PAGELATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
PAGELATCH_SH		True	I/O 요청 작업과 관련되지 않은 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 공유 모드로 요청된 래치 요청에서 사용됩니다. 이러한 경합현상은 IO 또는 메모리 성능이 아닌, 다른 이유 때문에 발생하게 됩니다. 예를 들어, 동일한 인덱스 범위에 동시 INSERT 작업이 수행되면, 이러한 경합현상이 발생할 수 있습니다. 대량의 INSERT 작업이 동일한 페이지에 대해 발생하게 되면, 이 래치 유형을 사용하여 각 INSERT 작업을 직렬화합니다. 또한, 동일한 범위에 대해 대량의 INSERT 작업이 발생하면, 래치가 설정된 상태의 인덱스 안에서 페이지 분할이 발생하여 새로운 페이지를 할당하는 작업(일정 시간이 소요됨)이 발생할 수 있습니다. 동일 페이지에 대한 INSERT 작업과 같이, 동일 범위에 대한 읽기 액세스가 발생하면 래치간에 충돌현상이 발생할 수 있습니다. 이러한 문제를 해결하기 위해서는 적절한 인덱스 전략을 통해 INSERT 작업을 분산시켜야 합니다.	PAGELATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
PAGELATCH_UP		True	I/O 요청 작업과 관련되지 않은 버퍼에 대한 래치 요청에 대해 대기하는 경우에 발생합니다. 이 대기 유형은 업데이트 모드로 요청된 래치 요청에서 사용됩니다. 이 대기유형은 페이지에 관련한 할당 작업에 대해서만 사용되며, 이 대기유형에 대한 경합현상이 발생하는 경우, 주로 좀 더 많은 데이터 파일이 필요하다는 것을 나타냅니다. 여러 개의 데이터 파일이 존재하면, 페이지에 대한 할당 작업을 여러의 파일로 분산될 수 있어서, 각 파일에 페이지를 저장하기 위한 요청이 줄어들 수 있습니다. 이러한 경합현상은 IO 성능과 관련된 것이 아니라, 해당 페이지에 액세스하기 위한 내부 할당 작업에 대한 경합현상입니다. 이러한 경합현상은 스핀들을 추가하거나, 좀 더 빠른 디스크로 해당 파일을 이동시키는 대책은 도움이 되지 않으며, 메모리를 추가하는 것도 도움이 되지 않습니다.	PAGELATCH_x 를 참조하십시오.

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
PRINT_ROLLBACK_PROGRESS			데이터베이스에 연결된 사용자 프로세스가 ALTER DATABASE 명령의 즉시 연결종료(termination) 옵션으로 인해 종료되는 동안, 대기하기 위해서 사용됩니다. 좀 더 자세한 정보는 온라인 설명서의 ALTER DATABASE (Transact-SQL) 부분을 참조하십시오.	
QNMANAGER_ACQUIRE			내부전용	
QPJOB_KILL			비동기 통계 업데이트 작업이 KILL 명령이 실행됨에 의해서 취소되었다는 것을 나타냅니다. 종료되는 스레스는 일시중지 상태로, KILL 명령을 수신할 수 있는 상태가 될 때까지 대기하게 됩니다. 1 초 미만의 값으로 유지되어야 합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
QPJOB_WAITFOR_ABORT			비동기 통계 업데이트 작업이 KILL 명령이 실행됨에 의해서 취소되었다는 것을 나타냅니다. 비동기 통계 업데이트 작업은 완료된 상태이지만, 스레드 메시지 조정 작업이 완료될 때까지 일시 중지됩니다. 이 대기 유형은 일반적으로 발생할 수 있지만, 매우 드물게 발생하며, 매우 짧은 기간 동안 발생해야 합니다. 1 초 미만의 값으로 유지되어야 합니다.	
QRY_MEM_GRANT_INFO_MUTEX		True	쿼리가 실행될 때 사용하는 메모리 관리하는 기능이 정적 권한부여 정보 목록에 대한 액세스를 통제하고자 하는 경우에 발생합니다. 이 대기 유형은 현재 할당되었거나 대기하고 있는 메모리 요청에 대한 정보를 나타냅니다. 이 대기 유형은 단순한 액세스 통제 상태를 나타냅니다. 이 대기 유형은 매우 짧은 시간동안만 나타나야 합니다. 이 뮤텍스(mutex)가 해제되지 않으면, 새로운 메모리를 필요로 하는 모든 쿼리가 응답하지 않는 상태가 될 수 있습니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
QUERY_NOTIFICATION_MGR_MUTEX		True	쿼리 알림 관리자(Query Notification Manager)에서 가비지(garbage) 컬렉션 큐에 대한 동기화 작업을 수행하는 동안에 발생합니다.	
QUERY_NOTIFICATION_SUBSCRIPTION_MUTEX		True	쿼리 알림 관리자에서 트랜잭션 상태에 대한 동기화 작업을 수행하는 동안에 발생합니다.	
QUERY_NOTIFICATION_TABLE_MGR_MUTEX		True	쿼리 알림 관리자 에서 수행되는 동기화 과정 동안에 발생합니다.	
QUERY_NOTIFICATION_UNITTEST_MUTEX		True	내부전용	
QUERY_OPTIMIZER_PRINT_MUTEX		False	쿼리 최적화 프로그램의 진단 결과를 생성하는 동안에 발생합니다. 이 대기 유형은 Microsoft Product Support 팀의 요청에 따라 진단을 위한 설정값을 활성화한 상태에서만 나타납니다.	
QUERY_TRACEOUT		True	내부전용	
RECOVER_CHANGEDB		True	웜(warm) 대기 데이터베이스에 대한 동기화 작업을 수행하는 동안 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
REPL_CACHE_ACCESS		True	복제 아티클 캐시에 대한 동기화 작업을 수행하는 동안에 발생합니다. 이 대기 유형이 지속되는 동안에는 복제 로그 판독기의 동작이 제한되며, 게시된 테이블에 대한 DDL 작업도 차단됩니다.	
REPL_SCHEMA_ACCESS		Yes	복제 아티클 캐시에 대한 동기화 작업을 수행하는 동안에 발생합니다. 이 대기 유형이 지속되는 동안에는 복제 로그 판독기의 동작이 제한되며, 게시된 테이블에 대한 DDL 작업도 차단됩니다.	
REPLICA_WRITES		True	작업이 데이터베이스 스냅숏 또는 DBCC 복제본에 대한 페이지 쓰기가 완료될 때까지 대기하는 동안 발생합니다.	
REQUEST_DISPENSER_PAUSE			스냅숏 백업을 위해 파일 I/O 를 고정할 수 있도록 작업이 처리 중인 모든 I/O 가 완료될 때까지 대기하는 경우에 발생합니다	
RESOURCE_QUEUE			다양한 내부 리소스 큐 동기화 중에 발생합니다.	동기화 개체

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
RESOURCE_SEMAPHORE		True	다른 동시 쿼리로 인해 쿼리 메모리 요청을 즉시 허용할 수 없는 경우에 발생합니다. 대기 수가 많고 대기 시간이 길면 동시 쿼리 수 또는 메모리 요청 양이 과도하게 많은 것입니다 작업부하가 의사결정지원시스템(DSS)와 유사하고, 해시 조인과 같은 대량 쿼리가 많은 환경에서는 대용량 쿼리를 실행하기 전에 해당 쿼리를 실행하기 위해 필요한 메모리를 할당받기 위해 대기해야 합니다.	SQL Memory Mgr 관련 성능 카운터를 점검하십시오. 1. Memory Grants Pending 2. Memory Grants Outstanding
RESOURCE_SEMAPHORE_MUTEX		True	쿼리가 스레드 예약 요청이 수행될 때까지 대기하는 동안 발생합니다. 또한 쿼리 컴파일 및 메모리 부여 요청을 동기화하는 경우에 발생합니다	
RESOURCE_SEMAPHORE_QUERY_COMPILE		True	동시 쿼리 컴파일 수가 조절 한계에 도달한 경우에 발생합니다. 대기 수가 많고 대기 시간이 길면 컴파일, 재컴파일 또는 캐싱할 수 없는 계획 수가 과도하게 많은 것입니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
RESOURCE_SEMAPHORE_SMALL_QUERY		True	다른 동시 쿼리로 인해 작은 쿼리의 메모리 요청을 즉시 허용할 수 없는 경우에 발생합니다. 서버는 몇 초 내에 요청된 메모리를 부여하지 못하면 주 쿼리 메모리 풀로 요청을 전송하기 때문에 대기 시간은 몇 초를 넘지 않아야 합니다. 대기 수가 많으면 대기 중인 쿼리가 주 메모리 풀을 차단하는 동안 작은 동시 쿼리 수가 과도하게 많은 것입니다.	
SEC_DROP_TEMP_KEY		True	임시 보안 키 삭제 시도가 실패한 후 다시 시도하기 전에 발생합니다.	
SERVER_IDLE_CHECK		True	리소스 모니터가 SQL Server 인스턴스를 유휴 상태나 켜는 중 상태로 선언하는 경우 SQL Server 인스턴스 유휴 상태 동기화 중에 발생합니다.	
SLEEP_BPOOL_FLUSH		True	체크포인트가 디스크 하위 시스템의 폭주를 방지하기 위해 새 I/O 의 실행을 조절하는 경우에 발생합니다	
SLEEP_SYSTEMTASK		True	tempdb 시작이 완료될 때까지 대기하는 동안 백그라운드 작업 시작 중에 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
SLEEP_TASK		True	일반 이벤트가 발생할 때까지 대기하는 동안 작업이 중지되는 경우에 발생합니다.	
SNI_HTTP_ACCEPT		True	내부전용	
SNI_HTTP_WAITFOR_0_DISCON		True	처리 중인 HTTP 연결이 종료될 때까지 대기하는 동안 SQL Server 종료 중에 발생합니다.	
SOAP_READ		True	HTTP 네트워크 읽기가 완료될 때까지 대기하는 동안 발생합니다	
SOAP_WRITE		True	HTTP 네트워크 쓰기가 완료될 때까지 대기하는 동안 발생합니다	
SOS_CALLBACK_REMOVAL		True	콜백을 제거하기 위해 콜백 목록에 대한 동기화를 수행하는 동안 발생합니다. 서버 초기화가 완료된 후에는 이 카운터가 변경되지 않습니다	
SOS_LOCALLALLOCATORLIST		True	SQL Server 메모리 관리자의 내부 동기화 중에 발생합니다	
SOS_OBJECT_STORE_DESTROY_MUTEX		True	풀에서 개체를 삭제하는 경우 메모리 풀의 내부 동기화 중에 발생합니다	
SOS_PROCESS_AFFINITY_MUTEX		True	프로세스 선호도(affinity) 설정에 대한 액세스 동기화 중에 발생합니다	
SOS_RESERVEDMEMBLOCKLIST		True	SQL Server 메모리 관리자의 내부 동기화 중에 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
SOS_SCHEDULER_YIELD		True	다른 작업이 실행될 수 있도록 작업이 자발적으로 스케줄러를 양보하는 경우에 발생합니다. 이 대기 중에 작업은 해당 퀀텀이 갱신될 때까지 대기합니다.	
SOS_STACKSTORE_INIT_MUTEX		True	내부 저장소 초기화 동기화 중에 발생합니다.	
SOS_SYNC_TASK_ENQUEUE_EVENT		True	작업이 동기 방식으로 시작되는 경우에 발생합니다. SQL Server 2005 의 작업은 대부분 비동기 방식으로 시작되므로 작업 요청이 작업 큐에 배치된 후 제어가 즉시 시작으로 돌아갑니다.	
SOS_VIRTUALMEMORY_LOW		True	메모리 할당이 리소스 관리자가 가상 메모리를 해제할 때까지 대기하는 경우에 발생합니다.	
SOSHOST_EVENT	SOS	True	CLR 과 같은 호스팅된 구성 요소가 SQL Server 2005 이벤트 동기화 개체를 대기하는 경우에 발생합니다.	
SOSHOST_INTERNAL	SOS	True	CLR 과 같은 호스팅된 구성 요소에 사용되는 메모리 관리자 콜백 동기화 중에 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
SOSHOST_MUTEX	SOS	True	Occurs when a hosted component, such as CLR, waits for a SQL Server 2005 mutex synchroniza CLR 과 같은 호스팅된 구성 요소가 SQL Server 2005 뮤텍스 동기화 개체를 대기하는 경우에 발생합니다 tion	
SOSHOST_RWLOCK	SOS	True	CLR 과 같은 호스팅된 구성 요소가 SQL Server 2005 읽기/쓰기 동기화 개체를 대기하는 경우에 발생합니다	
SOSHOST_SEMAPHORE	SOS	True	CLR 과 같은 호스팅된 구성 요소가 SQL Server 2005 세마포어 동기화 개체를 대기하는 경우에 발생합니다	
SOSHOST_SLEEP	SOS	True	일반 이벤트가 발생할 때까지 대기하는 동안 호스팅된 작업이 중지되는 경우에 발생합니다. 호스팅된 작업은 CLR 과 같은 호스팅된 구성 요소에 사용됩니다.	
SOSHOST_TRACELOCK	SOS	True	추적 스트림에 대한 액세스 동기화 중에 발생합니다.	
SOSHOST_WAITFORDONE	SOS	True	CLR 과 같은 호스팅된 구성 요소가 작업이 완료될 때까지 대기하는 경우에 발생합니다	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
SQLCLR_APPDOMAIN	CLR	True	CLR 이 응용 프로그램 도메인 시작이 완료될 때까지 대기하는 동안 발생합니다	
SQLCLR_ASSEMBLY	CLR	True	Sql appdomain 안에서, 로드된 어셈블리 목록에 대한 액세스를 대기하는 동안 발생합니다	
SQLCLR_DEADLOCK_DETECTION	CLR	True	CLR 이 교착 상태 감지가 완료될 때까지 대기하는 동안 발생합니다	
SQLCLR_QUANTUM_PUNISHMENT	CLR	True	CLR 작업이 실행 퀀텀을 초과하여 조절되는 경우에 발생합니다. 이러한 조절은 리소스를 많이 사용하는 이 작업이 다른 작업에 미치는 영향을 줄이기 위해 수행됩니다.	
SQLSORT_NORMMUTEX		True	내부 정렬 구조를 초기화하는 동안 내부 동기화 중에 발생합니다.	
SQLSORT_SORTMUTEX		True	내부 정렬 구조를 초기화하는 동안 내부 동기화 중에 발생합니다.	
SQLTRACE_BUFFER_FLUSH		True	SQLTrace 플러시 작업이 플러시 작업의 간격사이에 잠간 멈춰있는 상태에서 발생합니다. 이 대기는 필요한 것이고, 대기 시간이 길다고 해서 문제가 되지는 않습니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
SQLTRACE_LOCK		True	파일 추적에서 추적 버퍼 동기화 중에 발생합니다.	
SQLTRACE_SHUTDOWN		True	추적 종료가 처리 중인 추적 이벤트가 완료될 때까지 대기하는 동안 발생합니다	
SQLTRACE_WAIT_ENTRIES		True	SQL Trace 이벤트 큐가 패킷이 큐에 도착할 때까지 대기하는 동안 발생합니다	
SRVPROC_SHUTDOWN		True	프로세스가 올바르게 종료하기 위해 내부 리소스가 해제될 때까지 대기하는 동안 발생합니다.	
TEMPOBJ		True	임시 개체 삭제가 동기화되는 경우에 발생합니다. 이 대기는 드물게 발생하며 작업이 임시 테이블 삭제에 대한 액세스를 과도하게 요청한 경우에만 발생합니다.	
THREADPOOL		True	작업이 작업자가 실행될 때까지 대기하는 경우에 발생합니다. 이는 최대 작업자 설정이 너무 낮거나 해당 배치 실행이 비정상적으로 오래 수행되어 다른 배치에 사용할 수 있는 작업자 수가 감소한 것입니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
TRAN_MARKLATCH_DT		True	표시된 트랜잭션에 대한 Destroy 모드 래치를 대기하는 경우에 발생합니다. 트랜잭션 표시 래치는 표시된 트랜잭션의 커밋 동기화에 사용됩니다.	
TRAN_MARKLATCH_EX		True	표시된 트랜잭션에 대한 배타 모드 래치를 대기하는 경우에 발생합니다. 트랜잭션 표시 래치는 표시된 트랜잭션의 커밋 동기화에 사용됩니다.	
TRAN_MARKLATCH_KP		True	표시된 트랜잭션에 대한 KEEP 모드 래치를 대기하는 경우에 발생합니다. 트랜잭션 표시 래치는 표시된 트랜잭션의 커밋 동기화에 사용됩니다.	
TRAN_MARKLATCH_NL		True	내부전용	
TRAN_MARKLATCH_SH		True	표시된 트랜잭션에 대한 공유 모드 래치를 대기하는 경우에 발생합니다. 트랜잭션 표시 래치는 표시된 트랜잭션의 커밋 동기화에 사용됩니다.	
TRAN_MARKLATCH_UP		True	표시된 트랜잭션에 대한 업데이트 모드 래치를 대기하는 경우에 발생합니다. 트랜잭션 표시 래치는 표시된 트랜잭션의 커밋 동기화에 사용됩니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
TRANSACTION_MUTEX		True	트랜잭션에 대한 여러 개의 배치의 액세스 동기화 중에 발생합니다.	
UTIL_PAGE_ALLOC		True	트랜잭션 로그 검색이 메모리 부족 시 메모리를 사용할 수 있을 때까지 대기하는 경우에 발생합니다.	
VIEW_DEFINITION_MUTEX		True	캐시된 뷰 정의에 대한 액세스 동기화 중에 발생합니다.	
WAIT_FOR_RESULTS		True	쿼리 알림이 트리거될 때까지 대기하는 경우에 발생합니다.	
WAITFOR		True	WAITFOR Transact-SQL 문의 결과로 발생합니다. 대기 시간은 문의 매개 변수에 의해 결정됩니다. 이 대기는 사용자가 시작합니다.	"waitfor delay" 문을 실행하는 T-SQL 코드를 조사하십시오.
WORKTBL_DROP		True	작업 테이블 삭제가 실패한 후 다시 시도하기 전에 일시 중지하는 동안 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
WRITELOG			로그 플러시가 완료될 때까지 대기하는 동안 발생합니다. 로그 플러시를 발생시키는 일반적인 작업은 검사점과 트랜잭션 커밋입니다 성능 모니터 카운터, 프로파일러, sys.dm_io_virtual_file_stats, SHOWPLAN 옵션 등을 사용하여 디스크 병목현상이 발생하고 있는지 점검해야 합니다 이 대기유형을 줄이기 위해서는 다음의 작업을 수행해야 합니다. 1. 추가적인 IO 대역폭 추가, 2. 각 디스크 드라이브간의 IO 를 균등하게 분산 3. 트랜잭션 로그 파일을 전용 디스크 드라이브로 이동/격리	디스크 관련 성능 카운터를 점검하십시오: 1. Disk sec/read 2. Disk sec/write 3. Disk queues SQL Buffer Manager 관련 성능 카운터를 점검하십시오: 1. Page Life Expectancy 2. Checkpoint pages/sec 3. Lazy writes/sec 트랜잭션 로그 파일에 대한 Io_stall 값을 점검하십시오. • select * from sys.dm_io_virtual_file_stats(dbid,file#)
XACT_OWN_TRANSACTION		True	트랜잭션 소유권을 획득하기 위해 대기하는 동안 발생합니다.	
XACT_RECLAIM_SESSION		True	세션의 현재 소유자가 세션 소유권을 해제할 때까지 대기하는 동안 발생합니다.	

대기 유형	범주	Wait stats 테이블 포함여부	설명	관련 정보
XACTLOCKINFO		True	트랜잭션 잠금 목록에 대한 액세스 동기화 중에 발생합니다. 트랜잭션 자체 외에도 교착 상태 감지, 페이지 분할 중 잠금 마이그레이션 등의 작업이 트랜잭션 잠금 목록에 액세스합니다.	
XACTWORKSPACE_MUTEX		True	트랜잭션에서 제거 및 트랜잭션 참여 멤버 간의 데이터베이스 잠금 전환에 대한 동기화 중에 발생합니다	

큐 (성능모니터 카운터)

대기 및 큐 분석 방법론에서 큐에 대한 분석은 리소스의 사용률에 대한 정보를 제공하는 성능모니터(PERFMON)의 성능 카운터와 기타 데이터 원본을 기준으로 분석합니다. 성능모니터 카운터는 각 리소스의 측면에서 시스템의 성능에 대한 현황을 나타내는 지표가 됩니다.

성능 모니터의 성능 카운터 및 상호연관성, 추론가능한 결과, 대응방안

리소스 구성요소: 디스크

성능 개체: Physical Disk

모니터대상 카운터	설명	가능한 결론 / 대응방안
Current Disk Queue Length	지속적으로 높은 수치로 나타나면, 현재 IO 서브시스템의 용량이 부족하다는 것을 의미합니다.	disk sec/read 카운터와 disk sec/write 카운터를 통해 IO 이슈가 있는지를 확인하십시오. Waitstats 상관관계: 1. IO_COMPLETION 2. ASYNC_IO_COMPLETION 3. WRITELOG 4. LOGMGR
Average Disk	시간에 따른 디스크 큐의 평균 값을	disk sec/read 카운터와 disk sec/write

모니터대상 카운터	설명	가능한 결론 / 대응방안
Queue Length	의미합니다. 평균 디스크 큐 값이 지속적으로 높게 나타나고, disk sec/read 카운터와 disk sec/write 카운터도 높게 나타나면, IO 대역폭이 부족하다는 것을 의미합니다.	카운터를 통해 IO 이슈가 있는지를 확인하십시오. Waitstats 상관관계: 1. IO_COMPLETION 2. ASYNC_IO_COMPLETION 3. WRITELOG 4. LOGMGR
Disk Sec/Read	이상적인 환경에서라면, 읽기는 4-8 ms 이내에 완료되어야 합니다.(하드웨어 벤더에 실제 읽기 시간에 대한 기준 정보를 확인하십시오.) 디스크 큐가 지속적으로 높은 수치로 나타날 때는, disk sec/read 값이 디스크 큐에 영향을 받아서 높게 나타날 수 있습니다. disk sec/read 값이 높다는 것은, 현재 IO 서브시스템이 대역폭이 전체 요청을 처리하기에 부족하다는 것을 의미합니다. 여러 개의 디스크 드라이브가 있는 경우는 각 드라이브의 성능을 개별적으로 확인해야 합니다. 만약, 모든 디스크 드라이브에 전반적으로 문제상황이 발생하고 있는 경우라면, IO 서브시스템의 대역폭이 부족하다고 할 수 있습니다. 이러한 경우, 좀 더 많은 디스크 드라이브를 추가하면, 성능에 도움이 됩니다. 만약, 특정 디스크 드라이브에만 부하가 매우 집중되는 경우라면, 해당 디스크에 어떤 작업(예를 들어, 페이징 파일, 데이터베이스 파일, 트랜잭션 로그 파일, 다른 읽기/쓰기 작업)이 발생하고 있는지를 확인해야 합니다.	disk sec/read 카운터의 수치가 일반적인 읽기 시간(이상적인 읽기 시간은 하드웨어 벤더에 확인하십시오.)보다 더 높은 경우라면, 다음의 옵션을 점검해 보아야 합니다. 1. IO 병목현상을 해결하기 위해, 디스크 드라이브를 추가하거나, IO 를 다른 디스크 드라이브로 분산합니다. 예를 들어, 읽기 쓰기 작업이 집중적으로 발생하는 데이터베이스 파일이나 트랜잭션 로그 파일, 다른 어플리케이션 파일을 다른 곳으로 이동시킵니다. 2. 메모리 압박상황을 점검하기 위해, 메모리 관련 카운터를 점검하십시오. 3. SQL 테이블에 적절한 인덱스가 존재하는지 점검합니다. 제대로 설정된 인덱스는 IO 를 경감시켜 줍니다. SQL 쿼리 실행계획을 점검하여, 스캔이나 정렬과 같은 작업이 발생하고 있는지 확인합니다. 실행계획보기 기능을 활용하여 부가적인 정렬이 발생하고 있는 단계를 찾아냅니다. 4. SQL 프로파일러를 실행하여, 스캔을 발생시키는 T-SQL 문을 찾아냅니다. 프로파일러에서 scans 이벤트 클래스의 scan:stopped 이벤트를 선택합니다. 데이터 칼럼 탭에서 objectid 칼럼을 추가합니다. 추적을 실행합니다. 프로파일러 추적 결과를 추적 테이블에 저장하고, scans 이벤트를 조사합니다. 또는, duration, reads, writes 칼럼 값이 높게 나타나는 이벤트를 조사합니다. Waitstats 상관관계: 1. IO_COMPLETION 2. ASYNC_IO_COMPLETION 3. WRITELOG 4. LOGMGR

모니터대상 카운터	설명	가능한 결론 / 대응방안
Disk Sec/Write	이상적인 환경에서라면, 쓰기는 4-8 ms 이내에 완료되어야 합니다.(하드웨어 벤더에 실제 읽기 시간에 대한 기준 정보를 확인하십시오.) 디스크 큐가 지속적으로 높은 수치로 나타날 때는, disk sec/write 값이 디스크 큐에 영향을 받아서 높게 나타날 수 있습니다. disk sec/write 값이 높다는 것은, 현재 IO 서브시스템이 대역폭이 전체 요청을 처리하기에 부족하다는 것을 의미합니다. SAN 환경이라면, 쓰기가 1-2 ms 이내로 완료될 수도 있습니다.	disk sec/writes 카운터를 점검하십시오. 높은 성능(대량 insert, update, delete 작업 발생)을 보장하기 위해서는, 트랜잭션 로그 파일은 데이터베이스 파일과 분리하여, 별도의 디스크 드라이브에 위치시켜야 합니다. Waitstats 상관관계: 1. IO_COMPLETION 2. ASYNC_IO_COMPLETION 3. WRITELOG 4. LOGMGR

리소스 구성요소: 메모리 / 캐시

성능 개체: Memory

모니터대상 카운터	설명	가능한 결론 / 대응방안
Page Faults/sec	하드 페이지 폴트(디스크에 액세스해야 하는 페이지 폴트)와 소프트 페이지 폴트(페이지 폴트가 발생한 페이지를 물리적 메모리에서 찾을 수 있는 페이지 폴트)가 모두 포함됩니다. 대부분의 프로세서는 특별한 성능상의 문제없이 대량의 소프트 페이지 폴트를 처리할 수 있습니다. 하지만, 디스크에 직접 액세스가 필요한 하드 페이지 폴트 경우에는 분명한 지연현상의 원인이 될 수 있습니다. 좀 더 자세한 정보는 디스크 관련 카운터 정보를 참조하십시오.	메모리 압박, 낮은 데이터 페이지 적중률, memory grants pending 등이 발생하고 있는지 점검합니다.(SQL Server buffer manager 관련 성능 카운터 참고)
Pages/sec	하드 페이지 폴트를 해결하기 위해, 디스크로부터 읽거나 디스크에 쓴 페이지 수. 하드 페이지 폴트는 페이지를 가져오기 위해 물리적인 IO 가 필요한 페이지 폴트입니다.	Page Faults/sec 카운터 수치와 비교합니다. 메모리 압박, 낮은 데이터 페이지 적중률, memory grants pending 등이 발생하고 있는지 점검합니다.(SQL Server buffer manager 관련 성능 카운터 참고)

리소스 구성요소: CPU

성능 개체: Processor

모니터대상 카운터	설명	가능한 결론 / 대응방안
% User	SQL Server 가 User mode 로	% user time 카운터는 전체 CPU 사용률 중에

모니터대상 카운터	설명	가능한 결론 / 대응방안
Time	실행되는 동안 사용한 CPU 시간의 퍼센트입니다. Privileged mode 는 하드웨어나 메모리에 직접 액세스할 수 있는 운영체제 관련 시스템 컴포넌트가 사용한 CPU 시간입니다.	70%이상이 되어야 합니다. 작업관리자 (taskmgr.exe)를 사용하여, sqlservr.exe 가 얼마만큼 CPU 를 사용하고 있는지 점검하십시오. % user time 카운터가 70% 미만이라면, %Processor Time 카운터와 % Privileged time 카운터를 함께 점검하십시오.
% Privileged Time	운영체제에서는 운영체제 서비스에 액세스하기 위해서 어플리케이션 스레드를 Privileged mode 로 전환합니다.	전체 CPU 사용률 중에 20% 미만이 되어야 합니다. 작업관리자 (taskmgr.exe)를 사용하여, sqlservr.exe 가 얼마만큼 CPU 를 사용하고 있는지 점검하십시오. % privileged time 카운터가 20% 이상이라면, %Processor Time 카운터와 % Privileged time 카운터를 함께 점검하십시오.
% Processor Time	샘플링 간격 동안 사용된 CPU 사용률 퍼센트입니다.	CPU 리소스를 사용하는 일반적인 작업: 1. 컴파일과 재컴파일 작업은 CPU 리소스를 사용합니다. 실행계획 재사용과 매개변수화를 통해, 컴파일 작업 때문에 사용되는 CPU 리소스를 최소화할 수 있습니다. 컴파일, 재컴파일, 매개변수화, 실행계획 재사용에 대한 좀 더 자세한 정보는, 다음 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx) 재사용되는 실행계획은 다음 쿼리에서 usecounts 칼럼의 값이 1 이상이어야 합니다. Select cacheobjtype, objtype, usecounts, or refcounts from sys.dm_exec_cached_plans and order by usecounts 성능 모니터의 관련 성능 카운터: 1. System: Processor Queue length 2. SQL Statistics: Compilations/sec 3. SQL Statistics: Re-Compilations/sec 4. SQL Statistics: Requests/sec 다음 두 가지 항목이 모두 발생하고 있다면, 현재 CPU 병목현상이 발생하고 있다고 할 수 있습니다. 1. CPU 사용률(%)가 평균 85% 이상 2. Context switches (system 성능개체 참조)가 초당 20,000 이상 Light weight pooling 은 약 15%의 성능향상을 제공합니다. Light weight pooling (또는 파이버(fiber) 모드)은 하나의 스레드를 10 개의 파이버로 나누어서 작업을 수행합니다. 개별 스레드의 오버헤드보다는

모니터대상 카운터	설명	가능한 결론 / 대응방안
		파이버의 오버헤드가 더 적습니다.
% Idle Time	샘플링 간격 동안 CPU 가 유휴상태로 있었던 시간입니다.	
Interrupts/ sec	Interrupts/sec 카운터는 프로세서가 받았거나 서비스한 인터럽트의 초당 평균 발생 건입니다.	IO, Network 과 같은, 다른 성능카운터와 상관관계를 점검하십시오.

리소스 구성요소: Thread

성능 개체: Process

모니터대상 카운터	설명	가능한 결론 / 대응방안
Page Faults/sec	하드 페이지 폴트(디스크에 액세스해야 하는 페이지 폴트)와 소프트 페이지 폴트(페이지 폴트가 발생한 페이지를 물리적 메모리에서 찾을 수 있는 페이지 폴트)가 모두 포함됩니다. 대부분의 프로세서는 특별한 성능상의 문제없이 대량의 소프트 페이지 폴트를 처리할 수 있습니다. 하지만, 디스크에 직접 액세스가 필요한 하드 페이지 폴트 경우에는 분명한 지연현상의 원인이 될 수 있습니다. 좀 더 자세한 정보는 디스크 관련 카운터 정보를 참조하십시오.	메모리 압박, 낮은 데이터 페이지 적중률, memory grants pending 등이 발생하고 있는지 점검합니다.(SQL Server buffer manager 관련 성능 카운터 참고)

리소스 구성요소: System

성능 개체: System

모니터대상 카운터	설명	가능한 결론 / 대응방안
Processor Queue Length		CPU 시간을 스케줄링받기 위해서 대기하고 있는 스레드의 수입니다. 다음과 같은 일반적인 CPU 리소스를 사용하는 작업을 피해야 합니다. 1. 불필요한 컴파일과 재컴파일. 매개변수화와 실행계획 재용을 통해, CPU 리소스의 소모를 줄여줄 수 있습니다. 다음 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx) 2. 메모리 압박 3. 적절한 인덱스의 누락
Context Switches/sec		

리소스 구성요소: SQL Server
성능 개체: SQLServer:Access Method

모니터대상 카운터	설명	가능한 결론 / 대응방안
Forwarded Records/sec	forwarded record 포인터가 발생한 레코드 수입니다. 클러스터형 인덱스가 없는 테이블에서만 발생합니다. 최초 짧은 행으로 테이블에 추가된 다음, 추후에 더 폭이 넓은 행으로 업데이트된 경우, 해당 업데이트 작업이 현재 데이터 페이지에 저장할 수 없을 만큼 큰 경우에 발생합니다. forwarded record 포인터는 해당 행이 위치한 다른 데이터 페이지의 주소입니다.	최초 짧은 행으로 테이블에 추가된 다음, 추후에 더 폭이 넓은 행으로 업데이트하는 코드가 있는지 점검하십시오. forwarded record 는 다음과 같은 방법으로 제거할 수 있습니다. 1. 칼럼에 기본값을 사용하십시오. 그리하여, 더 폭이 넓은 행으로 업데이트되지 않게 하여, forwarded record 가 발생하는 근본적인 원인을 제거하십시오. 2. varchar 데이터형 대신 char 데이터형을 사용하십시오. 고정 길이 데이터형을 사용하여, 더 폭이 넓은 행으로 업데이트되지 않게 하십시오.
Full Scan/sec	테이블 또는 인덱스를 전체 스캔할 때 발생합니다. 스캔작업은 인덱스가 있는 상황이라고 할지라도 과도한 IO 를 발생시키는 원인이 됩니다.	SQL 프로파일러를 사용하여, 스캔이 발생하는 T-SQL 문을 찾을 수 있습니다. Scans 이벤트 클래스의 scan:started 이벤트와 scan:stopped 이벤트를 선택합니다. objectid 데이터 칼럼을 추가합니다. 추적 결과를 추적 테이블에 저장하고, 스캔작업에 대한 이벤트를 조사합니다. scan:stopped 이벤트는 높은 duration, reads, writes 이 발생한 이벤트를 찾아낼 수 있도록, 관련된 IO 정보를 제공해 줍니다.
Index Searches/ sec	index search 의 수. Index search 는 인덱스 안에 저장된 단일 인덱스 레코드에 대한 검색이나, 범위 스캔 작업에 의해서 사용됩니다.	Full Scan/sec 카운터와 비교하십시오. index search 값이 더 높은 수치로 나와야 바람직합니다.
Page Splits/sec	인덱스 페이지 오버플로우의 결과로, 페이지 분할이 발생한 수입니다. 주로, 클러스터형 인덱스와 비클러스터형 인덱스의 리프(leaf) 페이지에서 발생합니다.	페이지 분할은 랜덤 INSERT 작업의 결과로 발생하며, 부가적인 IO 오버헤드를 발생시킵니다. 데이터 페이지에 여유공간이 없는 상태에서 해당 페이지에 행이 추가되는 경우(인덱스의 순서를 맞추기 위해), SQL Server 는 새로운 행을 추가하기 위해 해당 페이지를 분할하고, 페이지의 일부 행을 새로운 페이지로 이동시킵니다. Disk: page sec/write 카운터와 비교하십시오. page sec/write 카운터가 매우 높은 수치로 나타나면, 그 원인이 되는

모니터대상 카운터	설명	가능한 결론 / 대응방안
		인덱스를 재구성함으로써, 일시적으로 페이지 분할을 줄일 수 있습니다. Fillfactor 를 지정하여, insert 를 위한 여유공간을 페이지에 남겨두도록 설정할 수 있습니다.

리소스 구성요소: SQL Server
성능 개체: SQLServer:Memory Mgr

모니터대상 카운터	설명	가능한 결론 / 대응방안
Memory Grants Pending	메모리 리소스는 각 사용자 요청에 의해 사용됩니다. 메모리가 부족한 경우에는, 쿼리를 실행하기 위해 필요한 메모리를 얻기 위해 대기해야 합니다.	Memory grants outstanding 카운터와 비교하십시오. Memory grants pending 카운터의 수치가 증가하면, 다음과 같은 조치를 취할 수 있습니다. 1. SQL Server 에 메모리를 추가로 할당합니다. 2. 서버에 물리적인 메모리를 추가합니다. 3. 메모리 압박여부를 점검합니다. "사용가능한 메모리가 부족합니다." 상황이 발생한다면, 인덱스 전략이 제대로 되어 있는지 점검하십시오. 관련된 대기유형 1. RESOURCE_SEMAPHORE

리소스 구성요소: SQL Server
성능 개체: SQLServer:Buffer Manager

모니터대상 카운터	설명	가능한 결론 / 대응방안
Buffer cache hit ratio	이미 캐시된 페이지가 요청된 경우에 대한 퍼센트입니다.	메모리 압박이 존재하는지 점검하십시오. Checkpoint pages/sec, Lazy writes/sec Page life expectancy 카운터를 함께 점검하십시오.
Checkpoint pages/sec	체크포인트 작업이 수행되는 동안 디스크에 기록된 페이지 수입니다. 체크포인트 작업은 SQL 캐시를 비우는 작업을 수행합니다.	Page life expectancy 카운터가 300 초보다 낮은 수치로 나타나고, 매우 높은 *lazy writes/sec* 카운터와 *checkpoint pages/sec* 카운터가 나타나는 경우에는, 메모리 압박이 발생하고 있다는 것을 나타냅니다.
Lazy writes/sec	lazy writer 에 의해서 디스크에 기록된 페이지 수입니다. 체크포인트 작업은 SQL 캐시를 비우는 작업을 수행합니다.	Page life expectancy 카운터가 300 초보다 낮은 수치로 나타나고, 매우 높은 *lazy writes/sec* 카운터와 *checkpoint pages/sec* 카운터가 나타나는 경우에는, 메모리 압박이 발생하고 있다는 것을 나타냅니다.
Page life expectancy	데이터 페이지가 평균적으로 SQL 캐시에 머물러 있는	Page life expectancy 카운터가 300 초보다 낮은 수치로 나타나고, 매우 높은 *lazy writes/sec*

모니터대상 카운터	설명	가능한 결론 / 대응방안
	시간(단위:초)입니다. Page life expectancy 카운터가 300 초보다 낮은 수치로 나타나면, 다음과 같은 문제가 발생하고 있다는 것을 나타냅니다. (1) 디스크로부터 데이터 페이지를 가져와서 SQL 캐시에 등록, (2) 메모리 문제 (3) 인덱스 누락. Lazy writes/sec 카운터와 Checkpoint pages/sec 카운터와의 상관관계를 점검하십시오.	카운터와 *checkpoint pages/sec* 카운터가 나타나는 경우에는, 메모리 압박이 발생하고 있다는 것을 나타냅니다. 누락된 인덱스나 잘못된 쿼리 실행계획(SQL 프로파일러에서 Scans 개체로 추적)이 존재하는지 점검하십시오. page faults/sec 카운터가 높은지 점검하십시오.
Readahead pages/sec	메모리 부족현상이 존재하거나, 디스크로부터 데이터 페이지를 가져와 SQL 캐시에 등록해야 하거나, 낮은 캐시 적중률이 발생하는 상황에서는, SQL Server 가 캐시 적중률을 높이기 위해, 미리읽기(사전에 미리 데이터페이지를 캐시로 읽어들임) 작업을 수행할 수 있습니다. 사용자가 서로 계속 데이터 페이지를 캐시에서 제거하는 경우만 아니라면, 미리읽기 작업 자체만으로는 문제가 되지 않습니다.	메모리 압박상황에서는, 다음과 같은 SQL buffer manager 관련 성능카운터와의 상관관계를 점검하십시오: buffer cache hit ratio, page life expectancy, lazy writes, checkpoint pages. 적절한 인덱스가 존재하는지, 잘못된 쿼리 실행계획이 존재하지는 않는지 점검하십시오.(SQL 프로파일러의 scans 개체 사용)

리소스 구성요소: SQL Server

성능 개체: SQLServer:Plan Cache

모니터대상 카운터	설명	가능한 결론 / 대응방안
Cache Hit Ratio	프로시저 실행계획 페이지가 이미 캐시에 존재하는 경우에 대한 퍼센트입니다. 예를 들어, 프로시저 캐시가 적중되는 경우입니다. 즉, 얼마나 자주 컴파일된 프로시저	메모리 압박상황이 존재하는지 점검하십시오. Checkpoint pages/sec, Lazy writes/sec Page life expectancy 카운터를 함께 점검하십시오. SQL 프로파일러의 Stored Procedure 개체의 CacheHit, CacheMiss, CacheInsert 이벤트를 사용하여, 어떤 저장 프로시저의 쿼리 실행계획이 캐시에서 적중(Hit)되는지, 캐시에 존재하지 않는지(Miss, Insert)를 점검하십시오. 적절한 쿼리 실행계획 재사용이 발생하고 있는지를 점검하십시오. "쿼리 실행계획 재사용" 섹션을 참고하십시오. 항상은 아니지만, 유사한 SQL 문은 실행계획을

	실행계획이 재컴파일이 발생하지 않고 프로시저 캐시에서 찾아지는지를 의미합니다.	재사용하는 것이 바람직합니다. 쿼리 실행계획 재사용을 점검하기 위해, SQL Statistics 성능 개체의 Compilations/sec 카운터 수치를 함께 점검하십시오. 메모리 압박상황이 발생하게 되면, 다른 데이터 또는 저장 프로시저 실행계획을 위한 공간을 캐시하기 위한 공간을 마련하기 위해, 자주 사용되지 않는 실행계획이 캐시에서 제거됩니다.

리소스 구성요소: SQL Server

성능 개체: SQLServer:Databases

모니터대상 카운터	설명	가능한 결론 / 대응방안
Log Flush Wait Time	트랜잭션 로그를 기록하는 동안 대기한 시간 (ms)	디스크 성능 관련 카운터를 참조하십시오. sys.dm_io_virtual_file_stats(dbid, file#) 함수를 사용하여, 트랜잭션 로그파일의 Io_stall(단위 :ms) 수치를 점검하십시오.
Log Flush Waits/sec	로그가 기록될 때까지 대기하고 있는 커밋된 건수.	디스크 성능 관련 카운터를 참조하십시오. sys.dm_io_virtual_file_stats 함수를 사용하여, Io_stall(단위 :ms) 수치를 점검하십시오.
Log Growths	Microsoft Windows® 시스템은 insert, update, delete 작업에 적합한 트랜잭션 로그 파일의 크기를 자동으로 증가시킵니다.	일반적으로, Windows 가 트랜잭션 로그 파일의 크기를 증가시키는 작업을 수행하는 동안에는 트랜잭션 로그에 대한 쓰기 작업이 임시적으로 중단됩니다. 트랜잭션 로그 파일의 자동 증가 크기가 충분하게 크게 설정되었는지 점검하십시오. 자동 증가 크기가 너무 작게 설정되어 있는 경우에는, 트랜잭션 로그 파일에 대한 자동 증가 작업이 너무 자주 수행되어 성능을 저하시킬 가능성이 있습니다.
Transactions /sec	초당 SQL Server 트랜잭션 수입니다.	

리소스 구성요소: SQL Server

성능 개체: SQLServer:General Statistics

모니터대상 카운터	설명	가능한 결론 / 대응방안
Logins/sec	초당 로그인 수입니다.	사용자 연결
Logout/sec	초당 로그아웃 수입니다.	
User connections	사용자 연결 수입니다.	

리소스 구성요소: SQL Server

성능 개체: SQLServer:Latches

모니터대상 카운터	설명	가능한 결론 / 대응방안
Average Latch Wait Time(ms)	래치(Latches)는 매우 짧은 기간 동안 발생하는 경량 동기화 개체입니다. 래치는 트랜잭션 지속시간동안 유지되지 않습니다. 주로 래치는 메모리에서 행을 처리하는 동안, 행 오프셋 테이블에 대한 변경작업을 통제하기 위해서 사용됩니다.	Average Latch Wait Time(ms) 카운터가 높은 수치로 나타나면, 성능 모니터의 DISK 및 MEMORY 관련 개체에서 다음과 같은 항목을 점검하십시오. 1. IO 병목현상 2. 메모리 압박 주로 메모리를 추가하거나, 디스크 서브시스템의 대역폭을 추가함으로써 래치 대기 시간을 줄일 수 있습니다.
Latch Waits/sec	Average Latch Wait Time(ms) 카운터를 참조하십시오.	
Total Latch Wait Time(ms)	래치(Latches)는 매우 짧은 기간 동안 발생하는 경량 동기화 개체입니다. 래치는 트랜잭션 지속시간동안 유지되지 않습니다. 주로 래치는 메모리에서 행을 처리하는 동안, 행 오프셋 테이블에 대한 변경작업을 통제하기 위해서 사용됩니다.	Total Latch Wait Time(ms) 카운터가 높은 수치로 나타나면, 성능 모니터의 DISK 및 MEMORY 관련 개체에서 다음과 같은 항목을 점검하십시오. 1. IO 병목현상 2. 메모리 압박 주로 메모리를 추가하거나, 디스크 서브시스템의 대역폭을 추가함으로써 래치 대기 시간을 줄일 수 있습니다.

리소스 구성요소: SQL Server

성능 개체: SQLServer:Locks

모니터대상 카운터	설명	가능한 결론 / 대응방안
Average Wait Time(ms)	다른 사용자를 차단하지 않도록 제한하기 위해서, 트랜잭션은 가능한 한 짧게 유지되어야 합니다.	힌트: 물리적 IO 를 발생시키는 원인이 되는, 메모리 압박이 존재하는지 점검하십시오. 메모리 압박상황에서는 트랜잭션 및 잠금의 지속시간이 더 연장될 수 있습니다.
Lock Wait Time(ms)	다른 사용자를 차단하지 않도록 제한하기 위해서, 트랜잭션은 가능한 한 짧게 유지되어야 합니다.	힌트: 물리적 IO 를 발생시키는 원인이 되는, 메모리 압박이 존재하는지 점검하십시오. 메모리 압박상황에서는 트랜잭션 및 잠금의 지속시간이 더 연장될 수 있습니다.
Lock Waits/sec	다른 사용자를 차단하지 않도록 제한하기 위해서, 트랜잭션은 가능한 한 짧게 유지되어야 합니다.	힌트: 물리적 IO 를 발생시키는 원인이 되는, 메모리 압박이 존재하는지 점검하십시오. 메모리 압박상황에서는 트랜잭션 및 잠금의 지속시간이 더 연장될 수 있습니다.

리소스 구성요소: SQL Server

성능 개체: SQLServer:SQL Statistics

모니터할 대상 카운터/설명	가능한 결론 / 대응방안
SQL Compilations/sec	SQL 문이 실행되기 전에, 쿼리 최적화 프로그램에서는 쿼리 실행계획을 생성해야 합니다. 쿼리 실행계획은 주어진 SQL 문장의 결과값을 반환하기 위해 수행해야 하는 단계로 구성됩니다. SQL Server 2005 쿼리 최적화 프로그램과 통계 정보에 대한 좀 더 자세한 정보는 다음의 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/qrystats.mspx) Compilations/sec 에는 초기 컴파일과 이후에 발생한 재컴파일이 모두 포함됩니다. 컴파일과 재컴파일 작업은 CPU 리소스를 많이 소모하는 CPU 집중 작업입니다. 쿼리 실행계획 재사용을 통해, 불필요한 컴파일을 제거할 수 있습니다. "쿼리 실행계획 재사용" 섹션을 참조하십시오. 항상은 아니지만, 유사한 SQL 문은 실행계획을 재사용하는 것이 바람직합니다. 매개변수화는 쿼리 실행계획 재사용을 위해 매우 중요한 역할을 합니다. 재컴파일의 일부 유형은 발생하지 않도록 피할 수 있습니다. SQL Server 2005 재컴파일과 관련한 좀 더 자세한 정보는 다음의 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx) 초기 컴파일 수만 계산하기 위해서는 compilations/sec 카운터 값에서 recompilations/sec 카운터 값을 차감하면 됩니다. batch requests/sec 카운터와 비교하여, 전체 배치 대비 컴파일의 점유율을 점검하십시오.
SQL Re-Compilations/sec	재컴파일 된 수만 포함됩니다. SQL 프로파일러는 재컴파일에 어떤 프로세서가 사용되었는지, 재컴파일 대상이 된 문장, 재컴파일의 원인에 대한 정보를 제공합니다. SQL 프로파일러에서, stored procedure 이벤트 클래스의 SP:recompilation 이벤트를 선택하고, eventsubclass 데이터 칼럼을 추가합니다. Eventsubclass 칼럼이 1 에서 6 까지인 이벤트를 조사하십시오. Eventsubclass 칼럼이 1 에서 6 까지인 이벤트는 재컴파일을 발생시키는 원인이 됩니다. 재컴파일과 관련한 좀 더 자세한 정보는, 다음의 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx)
Batch Requests/sec	전체 배치 요청 수를 반드시 compilations/sec 카운터와 비교하십시오.
Auto-Param Attempts/sec	Auto-Param Attempts/sec 카운터를 반드시 failed auto-params/sec 카운터와 비교하십시오. 적절한 매개변수화는 실행계획 재사용에 매우 중요한 역할을 합니다. Sp_executeSQL 저장 프로시저를 사용하여 임의 쿼리를 실행하는 것이 도움이 될 수 있습니다. 재컴파일과 관련된 좀 더 자세한 정보는 다음 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/200

	5/recomp.mspx)
Failed Auto-Params/sec	Auto-Param Attempts/sec 카운터를 반드시 failed auto-params/sec 카운터와 비교하십시오. . 적절한 매개변수화는 실행계획 재사용에 매우 중요한 역할을 합니다. Sp_executeSQL 저장 프로시저를 사용하여 임의 쿼리를 실행하는 것이 도움이 될 수 있습니다. 재컴파일과 관련된 좀 더 자세한 정보는 다음 기사를 참조하십시오.(http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx)

관심을 두고 확인해야 할 성능모니터 카운터 간의 비율 비교

성능 모니터의 일부 카운터는 다른 카운터와 비교함으로써 좀 더 효과적인 진단 결과를 도출할 수 있습니다. 다음에서 소개하는 성능 카운터 간의 비율과 비교는 절대적인 것은 아니지만, 그럼에도 불구하고, 올바른 진단 결과를 도출하기 위한 방향을 제시하는데 도움을 줍니다.

1. **Batch requests/sec 대 SQL Compilations/sec.** 동일한 유형의 트랜잭션이 매우 많은 양 실행되는 경향을 갖는 OLTP 작업부하 환경에서는, 실행계획을 재사용하는 것은 매우 바람직한 현상입니다. "쿼리 실행계획과 성능 카운터" 섹션을 참고하십시오. 전체 배치 요청 중에서 컴파일이 발생하는 비율이 매우 높은 경우에는, 메모리 압박현상이 발생하게 되고, 이는 다른 작업을 하기 위한 메모리 여유공간을 확보하기 위해, 프로시저 캐시에서 쿼리 실행계획이 빠르게 제거되는 원인이 됩니다. 임의(ad hoc) SQL 실행계획의 재사용에 중요한 역할을 하는 매개변수화가 수행되지 않으면, 마찬가지로 실행계획이 빠르게 제거되는 원인이 될 수 있습니다. 매개변수화란 리터럴 값 대신에 변수를 사용하는 것을 의미합니다. Sp_executesql 저장 프로시저를 사용하여 임의 SQL 문을 매개변수화할 수 있습니다. 성능 카운터의 SQL Server:SQL Statistiscs:Batch Requests/sec 과 SQL Server:SQL Statistiscs:SQL Compilations/sec 카운터를 비교합니다. SQL Server 2005 쿼리 최적화 프로그램과 통계에 대한 좀 더 자세한 정보는 다음의 기사를 참조하십시오.http://www.microsoft.com/technet/prodtechnol/sql/2005/qrystats.mspx
2. **SQL Compilations/sec 대 SQL Re-Compilations/sec.** SQL Compilations/sec 성능카운터에는 모든 컴파일 유형(초기 컴파일과 재컴파일)이 포함되지만, SQL Re-Compilations/sec 에는 재컴파일(초기 컴파일은 제외됨)만 포함됩니다. "쿼리 실행계획과 성능카운터" 섹션을 참조하십시오. 전체 컴파일 중 재컴파일에 비해 초기 컴파일 점유율이 낮은 경우(SQL Compilations – SQL Re-Compilations), 재컴파일과 관련한 문제가 발생하고 있을 가능성이 있습니다. 재컴파일이 발생하는 실제 SQL 문을 찾는 방법은 "쿼리 실행계획과 DMV"섹션을 참조하십시오. 성능카운터의 SQLServer:SQL Statistcs: SQL Compilations/sec 과 SQLServer:SQL Statistics:SQL Re-Compilations/sec 을 비교합니다. SQL Server 2005 재컴파일과 관련된 좀 더 자세한 정보는 다음의 기사를 참조하십시오. http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx
3. **Kernel CPU 대 User CPU.** (Kernel CPU/ User CPU) 비율이 25 % 이상이면, 네트워크, 디스크 드라이버, 하드웨어 이슈가 발생하고 있다는 것을 나타냅니다. 네트워크 및 디스크 IO 는 커널 모드로 서비스됩니다. SQL Server 는 사용자 모드로 서비스 됩니다. 작업 관리자를 확인하십시오. 성능 카운터의 Processor:%Processor Time, Processor:%User time 과 Processor:%Interrupt time 을 비교합니다.
4. **Context switches/sec .** 이 성능 카운터 수치는 20000 보다 낮은 값으로 유지되어야 합니다. 만약, 이 임계값보다 더 큰 값으로 나타난다면, 컨텍스트 전환이 너무 많이 발생하고 있다는 것을 나타냅니다. 이 성능 카운터 수치가 50000 보다 높은 값으로 나타난다면, 현재 서버에서는 실제로 주어진 작업을 수행하는 스레드보다 컨텍스트 전환하는 스레드에 더 많은 CPU 시간을 사용하고 있다는 것을 나타냅니다. 이러한 문제 상황은 Microsoft Windows 2000 Server 기반

운영체제를 사용하는 서버에서 발견됩니다. Windows Server® 2003 에서는 이러한 문제가 여전히 발생하기는 하지만, 그 발생 빈도가 적습니다.

5. **Disk Queue Length 대 Disk sec/Transfer.** Disk Queue Length 성능 카운터 수치가 증가하면, disk sec/transfer 성능 카운터 수치도 함께 증가합니다. 성능 모니터의 PhysicalDisk:Avg Disk Queue Length 와 PhysicalDisk:Avg Disk sec/Transfer 를 비교합니다.
6. **Page life expectancy, checkpoint pages/sec, lazy writes/sec 비교.** page life expectancy 성능 카운터 수치가 낮게 나타나고, checkpoint pages /sec 성능카운터의 수치와 lazy writes/sec 성능카운터의 수치가 높게 나타난다면, 메모리 압박현상이 발생하고 있다는 것을 의미합니다. 성능에 좋지 않은 영향을 줄 수 있는 메모리 압박 상황을 해결하기 위해서는, 다음의 대안 중 하나 또 그 이상의 대안을 적용할 것을 고려해야 합니다.
   a. 현재 서버에 메모리를 추가로 증설
   b. SQL 서버에서 사용하는 메모리를 증가
   c. 적절한 인덱스 전략을 통해 테이블 스캔 또는 인덱스 스캔이 발생하지 않도록 제한
7. **SQL buffer cache hit ratio.** 이 성능 카운터 수치가 지속적으로 90% 미만으로 나타난다면, 버터 캐시가 자주 플러시되고 있음을 나타냅니다.
8. **시그널 대기 와 실행가능한 큐 비교.** 실행 모델(단순화)의 기본동작은 다음과 같습니다.
   a. 특정 세션 ID 가 실행 중이고, 수행 중인 작업에 필요한 리소스에 잠금이 설정되어 있어 동시에 사용이 불가능한 상태라면, 해당 리소스에 대한 대기 리스트로 이동되어 대기합니다.(T_0 시점).
   b. 리소스가 가용한 상태가 되면 시그널이 발생하며, 해당 SPID 는 T_1 시점에 실행가능한 큐로 이동합니다.
   c. SPID 는 실행가능 큐에서 CPU 가 해당 SPID 순서에 도착하여 실행 중 상태가 되는 시점(T_2)까지 대기합니다.
   d. 리소스 대기 시간은 T_0 시점에서 T_1 까지 실제로 해당 리소스가 가용한 상태가 될 때까지 대기한 시간입니다.
   e. 시그널 대기 시간은 실행가능한 큐에서 대기한 시간입니다. 리소스가 가용한 상태가 된 시점 (T_1)으로부터 프로세스가 실행 중 상태가 된 시점(T_2)까지의 대기 시간을 의미합니다. 그러므로, 시그널 대기 시간은 T_2-T_1 으로 계산합니다.

      $T_1 - T_0$ 가 큰 값으로 나타나면, 특정 리소스의 가용성이 제한되어 병목이 발생하고 있음을 나타냅니다. 또한, 잠금에 대한 대기가 발생하고 있다면, 차단현상이 발생하고 있을 가능성이 높습니다. IO 에 대한 대기가 발생하고 있다면, 디스크 서브시스템에 병목현상이 발생하고 있을 가능성이 높습니다.

      $T_2 - T_1$ 는 CPU 압박상황을 나타냅니다. 이 값이 증가하면 실행가능한 큐에서 실행되기까지 대기하는 시간이 증가한다는 것을 나타냅니다. 실행가능한 큐에 존재하는 Session_id 는 CPU 리소스만을 대기합니다. 이 대기가 전체 대기의 25%이상인 경우라면, CPU 병목현상이 발생하고 있음을 나타냅니다.
   f. 핵심 질문: 리소스 대기 시간과 시그널 대기 시간 중 어느 부분에 문제가 발생하고 있습니까?
      - 가장 높은 대기 시간이 발생하는 부분이 확장성 문제를 해결하기 위해 반드시 해결해야 하는 병목지점을 나타냅니다.
      - 일반적으로 시그널 대기 시간 점유율이 낮게 나타나면, CPU 가 작업부하를 처리하고 있다는 것을 나타냅니다. 예를 들어, session_id 가 실행가능한 큐를 통해 빠르게 이동하고 있다는 것을 나타냅니다.

- 시그널 대기 시간 점유율이 높게 나타나면, CPU 에 병목현상이 발생하고 있음을 나타내며, SPID 가 실행가능한 큐로 이동된 이후에 실행 중 상태가 될 때까지 대기하는 시간이 오래 걸린다는 것을 나타냅니다.

9. **Network: Current bandwidth, bytes total/sec, packets/sec.** 네트워크 대역폭 문제는 반드시 bytes total/sec 성능 카운터 수치과 비교하여 확인해야 합니다. [Network interface: bytes total/sec] / [Network interface: Current Bandwidth] 비율이 60%이상인 경우에는, 네트워크 병목현상이 발생할 수 있다는 것을 의미합니다.
10. **Page Faults/sec 대 Pages/sec.** 페이지 폴트에는 하드 페이지 폴트(디스크 액세스가 필요한 페이지 폴트)와 소프트 페이지 폴트(물리적인 메모리 상에서 폴트가 발생한 페이지를 찾을 수 있는 페이지 폴트)가 모두 포함됩니다. 대부분의 프로세서는 성능의 문제없이 대량의 소프트 폴트를 처리할 수 있습니다. 하지만, 하드 페이지 폴트의 경우에는, 디스크에 액세스가 필요하기 때문에, 심각한 지연현상의 원인이 될 수 있습니다. Pages/sec 은 페이지를 메모리로 가져오기 위해 물리적인 IO 가 필요한 하드 페이지 폴트 수를 나타냅니다.

메모리 이슈

SQL Server 관계형 데이터베이스 시스템은 내부적으로 다양한 목적으로 메모리를 사용합니다. SQL Server 에서 사용하는 메모리에 대한 좀 더 자세한 정보는 다음 기사를 참조하십시오. http://www.winnetmag.com/Article/ArticleID/43419/43419.html.

SQL Server 가 사용하는 메모리를 요약하여 정리하면 다음과 같습니다.

1. 데이터베이스 페이지(테이블/인덱스)를 캐시하기 위해서 사용하는, 데이터베이스 페이지 캐시
2. 해시, 정렬과 같은, 메모리 집중 쿼리 연산을 수행하기 위해서 사용하는, 쿼리 워크스페이스 메모리
3. 실행계획을 재사용하기 위해 캐시하는데 사용하는 플랜 캐시
4. 잠금, 연결 메모리, 스레드 스택, 백업/복원과 같이, 유틸리티 등이 사용하는 기타 메모리
5. SQL Server 프로세스에 연결된 기타 구성요소(확장 저장 프로시저, OLE-DB 공급자 등)에서 사용하는 메모리. 이 메모리는 주로 MemToLeave 영역의 메모리로 할당되며, 이 영역의 메모리를 연결된 다른 구성요소에서 사용할 수 있도록 하기 위해, SQL Server 는 이 영역의 메모리를 할당해서 사용하지 않습니다.

32 비트 메모리 아키텍처와 64비트 메모리 아키텍처의 비교

물론, 메모리의 1 차적인 사용용도인 데이터베이스 페이지 캐시는 32 비트 시스템에서도 AWE 메모리를 사용할 수 있습니다. 하지만, 데이터베이스 페이지 캐시 외의 나머지 다른 용도에서는 가상 메모리를 필요로 하기 때문에, 2 GB 범위(boot.ini 파일에 /3GB 옵션을 지정한 경우에는 3 GB 까지)로 제한됩니다. 어플리케이션이 SQL Server 에서 데이터베이스 페이지 캐시 이외에 하나 또는 그 이상의 용도로 메모리를 사용해야 하는 상황에서, 32 비트 가상 메모리 제한에서 처리할 수 있는 범위를 초과하는 포인터가 필요한 경우에서는, 64 비트 옵션을 고려해 보아야 합니다. 64 비트 옵션을 고려해야 하는 경우인지 여부를 판단하기 위해서는, 다음의 단계를 따라 필요한 고려사항을 점검해 보아야 합니다.

1. 전체 서버 메모리: *SQL Server:Memory Manager* 관련 성능 카운터를 점검하십시오. 만약, *Total Server Memory* 성능 카운터가 *Target Server Memory* 성능 카운터보다 안정적인 상태로 낮게 나타난다면, 메모리 압박 상황은 존재하지 않는다는 것을 의미합니다. 이러한 경우처럼, 메모리로 인한 아무런 성능 문제가 발생하지 않는 상황에서라면, SQL Server 64 비트 시스템을 고려할 이유가 없습니다. 하지만, 만약, 반대의 경우라면, 다음에서 제시하는 단계를 수행하여, 메모리 압박 상황의 원인이 무엇인지 좀 더 면밀하게 살펴볼 필요가 있습니다. 물론, 메모리

압박 상황이 발생하고 있다면, 이미 SQL Server 에 추가 메모리를 증설하고, AWE 옵션을 활성화하여 사용하고 있을 수 있습니다. 이러한 경우, SQL Server 가 시작될 때 Max Server Memory 로 설정된 메모리를 할당하기 때문에, *Total Server Memory* 가 동적으로 변경되지 않습니다. 이러한 상황에서도, 메모리 압박 현상이 계속되고 있다면, 다음 단계의 권고사항을 적용해 보아야 합니다.

2. 또한, 버퍼 캐시 적중률이 정상적인 값으로 유지되는지 점검해야 합니다. 주로, 메모리 관련 문제가 발생하고 있음을 알려주는 가장 대표적인 증상 중 하나입니다.
3. 쿼리 워크스페이스 메모리: *SQL Server:Memory Manager* 관련 성능 카운터를 점검하십시오. *Memory Grants Outstanding* 성능 카운터와 *Memory Grants Pending* 성능 카운터를 점검하십시오. 만약, *Memory Grants Pending* 성능 카운터가 *Memory Grants Outstanding* 성능 카운터에 비해 더 높게 나타난다면, 현재 쿼리 워크스페이스 메모리로 인해 메모리 압박이 발생하고 있다고 할 수 있습니다. 현재 실행 중인 쿼리에 할당된 전체 메모리 양을 나타내는 *Granted Workspace Memory (KB)* 성능 카운터를 확인함으로써 메모리 압박현상이 발생하고 있는지 확인할 수 있습니다. 현재 발생하고 있는 메모리 압박현상의 원인이 쿼리 워크스페이스 메모리 때문이라고 판단하기 위해서는, 쿼리 워크스페이스 메모리가 SQL Server 에서 사용할 수 있는 가상 메모리 중에서 적어도 25%이상은 점유해야만 합니다. 메모리 압박상황이 극심해지면, 서버에서는 701 또는 8645 오류를 반환합니다. 이러한 오류가 발생한다면, SQL Server 64 비트 버전을 사용할 것을 고려할 수 있는 좋은 근거자료가 될 수 있습니다.
4. 플랜 캐시: *SQL Server:Buffer Manager:Procedure Cache Pages* 성능 카운터는 플랜 캐시에 존재하는 전체 페이지 수를 나타냅니다. 이 성능 카운터 수치가 버퍼 풀에 존재하는 전체 페이수에 비해서 상당한 비율(주로, 25 퍼센트 이상)로 나타나는 경우라면, 어플리케이션이 플랜 캐시를 매우 많이 사용하고 있다는 것을 나타냅니다. 하지만, 어플리케이션에서 플랜 캐시를 매우 많이 사용한다고 해서, SQL Server 64 비트 시스템으로 이행해야 하는 당위성이 확보된 것은 아닙니다. 만약, 플랜 캐시에 등록된 쿼리 실행계획 중 대부분이 거의 재사용되고 있지 않아 플랜 캐시의 크기가 증가한 경우라면, SQL Server 64 비트 시스템으로 이행하는 것은 거의 도움이 되지 않을 것입니다.(앞에서 언급한 바와 같이, 플랜 캐시의 크기가 증가하게 되면 오히려 성능에 좋지 않은 영향을 미칠 가능성도 있습니다.) 플랜 캐시에 저장된 쿼리 실행계획의 종류는 "쿼리 실행계획 재사용" 섹션의 내용을 참조하고, 해당 실행계획이 재사용되는지를 확인하기 위해서는 usecounts 칼럼을 참조하십시오. 플랜 캐시에 저장된 대부분의 쿼리가 재사용되고, 여전히 메모리 압박상황이 발생한다면, 해당 어플리케이션은 좀 더 큰 가상 메모리를 필요로 한다는 의미이기 때문에, SQL Server 64 비트 시스템이 고려할 수 있는 바람직한 대안이 될 수 있습니다.
5. MemToLeave 영역에 메모리 압박은 주로 확장 저장 프로시저나 OLE-DB 공급자로 인해 발생합니다. MemToLeave 영역에 메모리 압박이 발생하면, 주로 7399, 17802, 17803 과 같은 오류가 나타납니다. 이러한 경우, MemToLeave 영역을 증가시켜 주기 위해, 시작옵션 –g 매개변수의 설정값을 증가시킬 수 있습니다. 하지만, 이러한 작업은 다른 메모리 영역에 메모리 압박상황을 발생시키는 원인으로 전환될 수 있는 가능성이 존재합니다.
6. 높은 CPU 비용을 사용하는 AWE 메모리: 일부 경우에서는, AWE 메모리가 주로 데이터베이스 페이지 캐시를 구성하기 위해 사용된다고 하더라도, AWE 매커니즘을 사용하여 데이터베이스 페이지를 매핑하고, 매핑해제하는 작업을 수행하기 위해 사용하는 CPU 비용이 매우 높게 나타난다는 것을, 높은 커널 CPU 시간을 통해 확인할 수 있습니다. 특히, 시스템의 CPU 수가 8 개 이상이고/이거나, 32 비트 시스템에서 물리적 메모리가 32GB 를 초과하는 상황에서는 특히 문제가 될 수 있습니다. 이러한 경우라면, 64 비트 시스템을 사용할 것을 고려할 수 있습니다.

64-비트 메모리 아키텍처 vs. 더 빠른 32-비트 CPU 성능

앞에서 살펴본 바와 같이, 메모리 압박 상황에는 순수한 메모리 압박상황이 있고, 다른 하나는 SQL Server 64 비트 시스템을 도입하면 개선할 수 있는 메모리 압박상황이 있습니다. 하지만, 32 비트

시스템에 대한 대안으로 64 비트 시스템을 선택하는 것이 항상 바람직한 대안은 아닐 수 있습니다. Xeon 기반의 32 비트 시스템의 클럭속도에 비해, Itanium 기반 64 비트시스템의 클럭 속도가 상대적으로 매우 느립니다. 물론, Itanium 프로세서는 동시에 여러 개의 명령을 실행할 수 있기 때문에, 클럭 속도의 차이 중 어느 정도는 완화되는 효과를 얻을 수 있습니다. 하지만, 어플리케이션에서 CPU 를 많이 사용해야 하는 상황이고, 비교대상이 되는 32 비트 시스템에서 처리하는 것과 동일한 작업부하를 처리해야 한다면, 64 비트 시스템에서는 좀 더 많은 CPU 가 필요할 수 있습니다. 64 비트 플랫폼에 대한 투자효과를 검증하기 위해서, 프로토타입이나 PoC(Proof of Concept) 프로젝트를 통해, 선택가능한 두 개의 옵션(32 비트 플랫폼과 64 비트 플랫폼)에 대한 성능을 검증할 것을 권장합니다.

어플리케이션 디자인 이슈

어플리케이션 설계 측면에서 대기 및 큐 분석 방법론을 적용하기 위한 고려사항이 존재합니다. 다음 표는 어플리케이션 설계 측면에서 고려해야 할 고려사항 중 일부입니다.

관점	어플리케이션 이슈	가능한 대안
높은 IO 대기	데이터베이스 설계 메모리 압박	잘못된 인덱스 전략으로 인한 잘못된 실행계획 IO 를 최소화할 수 있는 적절한 인덱스를 추가 메모리를 증설
높은 CPU 사용률	메모리 압박 실행계획 재사용 매개변수화	올바른 실행계획 재사용, 매개변수화, 재컴파일과 관련한 이슈을 점검하기 위해서는, 다음 기사를 참조하십시오.http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx
높은 차단/잠금	트랜잭션 관리	트랜잭션 재실행 관리 적절한 트랜잭션 격리 수준 사용

권장사항

대기 및 큐 분석 방법론은 성능 문제를 진단하고 해결하기 위해 매우 효과적인 기법을 제공하기 때문에 반드시 적용할 것을 권고합니다. 소위, "한방에 큰 건을 건지는(biggest bang for the buck)", 실제로 성능을 향상시킬 수 있는 최상의 기회를 제공해 줍니다. 이러한 성능 향상을 통해, 성능 튜닝을 위해 투자해야 하는 시간 중 상당부분을 절감할 수 있습니다.

결론

SQL Server 의 성능 정보를 제공해 주는 두 가지 상호 보완적인 데이터 원본이 존재합니다. 대기 유형은 어플리케이션 관점에서 전체 시스템의 성능을 분석하기 위한 매우 유용한 단서를 제공합니다. 대기 유형이 SQL 스레드 입장에서 시스템 성능 정보를 제공한다면, 성능 모니터는 리소스 측면에서 시스템 성능 정보를 제공합니다.

대기 통계정보는 성능모니터의 관련된 리소스에 대한 성능 가운터 수치와 함께 분석하고 확인해야 합니다. 예를 들어, SQL Server 에 시그널 대기가 높은 점유율로 나타난다면, 프로세서, IO 서브시스템, 네트워크 등과 같은 기존 리소스에 대한 성능 모니터의 관련 성능 카운터 조사도

병행하여 수행해야 합니다. 대기유형과 성능카운터 간의 연관관계를 분석하는 작업과, 서로 연관된 성능 카운터간의 비율을 분석하는 작업은 어플리케이션 성능에 대해 광범위한 정보를 제공합니다.

숙련된 성능 전문가라고 할지라도 근본적인 문제를 찾기 위해서는 반드시 단순한 증상을 초월하는 부분까지 면밀하게 조사해야 합니다. 비록 완벽한 결과를 보장해 주지는 못할지라도, 상호 연관된 성능 정보, 추론가능한 결과와 대응방안, 성능 정보간의 비율 계산 및 비교 작업을 통해, 주어진 증상에 대한 실제적인 근본 문제를 조명할 수 있는 정보를 얻을 수 있습니다. 이 문서에서 소개한 대기 및 큐 분석 방법론을 적용하면, 시스템의 병목지점을 찾아내고, 부가적인 정보와 추론가능한 결론을 얻을 수 있습니다.

결론적으로, 대기 및 큐 분석 방법론은 대기 통계 정보와 성능 모니터의 성능 카운터 정보, 기타 관련 정보로 구성된 종합적인 성능 정보를 제공해 주며, 이는 어플리케이션 성능에 대한 전반적인 프로필을 제공합니다. 또한, 이러한 성능 정보는 성능 문제를 해결하기 위해 정확한 병목지점을 도출하는데 매우 유용한 도구로서 사용됩니다.

추가 정보:

http://www.microsoft.com/technet/prodtechnol/sql/bestpractice/default.mspx