sqlserver基础知识:SQL Server 连接基础知识

引言 该堆栈的顶部是 API 或对象库层。应用程序通过对象库公开的 API 函数或接口连接到 Microsoft® SQL Server。用于访问 SQL Server 的 API 示例包括 ODBC 和 DB-Library。用于访问 SQL Server 的对象库示例包括 OLE DB、ADO 和 ADO.NET。由于 ADO 最终使用 OLE DB 与服务器通信，因此 Windows 应用程序在与 SQL Server 通信时实际上只使用两个常用的对象库，即 OLE DB 和 ADO.NET。由于通过 ADO 或 ADO.NET 进行连接通常比通过 ODBC 进行连接更普遍（但 SQL Server 的查询分析器和企业管理器仍通过 ODBC 进行连接），因此本文将从 ADO/OLE DB 和 ADO.NET 的角度介绍 SQL Server 连接体系结构的客户端。如今，大多数应用程序均通过对象库（而非 ODBC 或类似 API）连接到 SQL Server。

ADO 和 OLE DB OLE DB 客户端（也称作使用者）通过客户端提供程序与服务器以及其他后端程序进行通信。此提供程序是一组 COM 组件（一个或多个），用于将应用程序请求转换为网络进程间通信 (IPC) 请求。在使用 SQL Server 的情况下，最常用的 OLE DB 提供程序是 SQLOLEDB，它是 Microsoft 为 SQL Server 提供的 OLE DB 提供程序。SQLOLEDB 随附于 SQL Server 中，并作为 Microsoft 数据访问组件 (MDAC) 库的一部分安装。 为了使用 ADO 与 SQL Server 进行通信，应用程序首先使用 Connection 对象建立与服务器的连接。ADO 的 Connection 对象接受一个连接字符串，该字符串指定要使用的 OLE DB 提供程序以及传递给它的参数。如果应用程序使用 SQLOLEDB 提供程序连接到 SQL Server，则该字符串中将显示“SQLOLEDB”。 ADO 应用程序还可以通过 ODBC 连接到 SQL Server。为此，应用程序将使用适用于 ODBC 的 OLE DB 提供程序，并指定在其连接字符串中引用目标 SQL Server 的 ODBC 数据源。这种情况下，应用程序与 OLE DB 进行通信，同时 ODBC 的 OLE DB 提供程序调用相应的 ODBC API，以便与 SQL Server 进行会话。 ADO.NET ADO.NET 应用程序通常使用 .NET Framework Data Provider for SQL Server 连接到 SQL Server。该本机提供程序使 ADO.NET 对象能够与 SQL Server 直接进行通信。通常，应用程序使用 SqlConnection 对象建立连接，然后使用 SqlCommand 对象向服务器发送命令，并接收服务器返回的结果。SqlDataAdapter 和 SqlDataReader 类通常与 SqlCommand 一起使用，以便通过托管的代码应用程序与 SQL Server 进行交互。 通过 OleDbConnection 类，ADO.NET 应用程序还可以使用 SQLOLEDB OLE DB 提供程序与 SQL Server 进行交互。此外，它们可以通过 OdbcConnection 类使用 ODBC 访问 SQL Server。因此，仅通过托管代码，您就有三种不同的方法从应用程序访问 SQL Server。从故障排除的角度而言，了解这些方法是非常有用的，因为它可以帮助您将遇到的与连接相关的问题归结到特定的数据访问层或库。 客户端 Net-Library 该堆栈中的下一层是 Net-Library。Net-Library 在 API 或对象库（应用程序使用它与 SQL Server 进行通信）与网络协议（用于与网络交换数据）之间提供了一个通道。SQL Server 为所有主要的网络协议提供了 Net-Library。这些库以透明方式将客户端发出的请求发送到 SQL Server，并将服务器发出的响应返回给客户端。可以使用 SQL Server 的客户端网络实用程序配置适用于特定客户端的 Net-Library。支持的客户端协议包括 TCP/IP、命名管道、NWLink、多协议 (RPC) 和其他一些协议。 尤其值得一提的 Net-Library 是共享内存 Net-Library。顾名思义，该 Net-Library 使用 Windows 的共享内存功能在 SQL Server 客户端与服务器之间进行通信。显然，这意味着客户端与服务器必须

位于同一台物理计算机上。 由于它能够绕过物理网络堆栈，因此共享内存 Net-Library 要比其他 Net-Library 快得多。对共享内存区域的访问受到同步对象的保护，因此客户端与服务器之间的通信速度主要受限于 Windows 对内核对象进行调度的能力，以及进程与共享内存区域之间进行数据复制的能力。
可以在连接时将某个时间段或（本地）指定为您的计算机名，来指示使用共享内存 Net-Library。也可以在连接时为计算机\实例名加上前缀 lpc:，来指示要使用共享内存 Net-Library。 注意，即使连接到同一台计算机上的 SQL Server，共享内存 Net-Library 也未必就是最佳的连接选项。在某些情况下，客户端与服务器之间的直接连接可能限制它的扩展性。与应用程序整体体系结构中的其他元素一样，应始终对给定技术解决方案进行全面的测试，然后才能判断它是否有良好的扩展性以及是否比其他方法更快。 连接 客户端进行连接时，SQL Server 的用户模式计划程序 (UMS) 组件将它指定给特定的计划程序。启动时，SQL Server 为系统上的每个 CPU 创建一个单独的 UMS 计划程序。当客户端连接到服务器时，这些客户端将指定给具有最少连接数的计划程序。连接后，客户端将不会更换计划程序 - 它将始终受到指定计划程序的控制，直到连接断开。
这对与服务器建立多个连接的应用程序很重要。如果应用程序性能较差，或无法在它的多个连接上平均分配工作，则在该应用程序的某些连接之间可能造成不必要的 CPU 资源争用，而其他连接实际上却处于空闲状态。 例如，应用程序与双处理器计算机上运行的 SQL Server 建立了四个连接，连接 1 和 3 隶属于处理器 0，连接 2 和 4 隶属于处理器 1。如果应用程序的大部分工作通过连接 1 和 3 执行，则这两个连接将争用 CPU 0，而 CPU 1 实际上可能仍处于空闲状态。这种情况下，应用程序只能断开某些连接或重新连接某些连接，并希望连接 1 和 3 隶属于不同的 CPU（连接时无法指定处理器隶属关系），或在它的连接上重新分配工作负荷，以便每个连接的工作负荷更加均衡。当然，后一种情况要远好于前一种情况。 连接内存 SQL Server 为客户端请求的每个连接保留三个数据包缓冲区。每个缓冲区的大小取决于 sp_configure 存储过程指定的默认网络数据包大小。如果默认网络数据包大小小于 8 KB，则这些数据包的内存将由 SQL Server 的缓冲池提供。否则，该内存将由 SQL Server 的 MemToLeave 区域分配。 值得一提的是，.NET Framework Data Provider for SQL Server 的默认网络数据包大小为 8KB，因此，与托管代码客户端连接关联的缓冲区通常由 SQL Server 的 MemToLeave 区域提供。而典型的 ADO 应用程序却不同，它们的默认数据包大小为 4 KB，因此缓冲区将由 SQL Server 缓冲池分配。 事件 连接后的客户端请求通常分为两种广泛类别：语言事件和远程过程调用。尽管还存在其他类别，但大多数由 SQL Server 客户端发送到服务器的请求由以下两种类型之一构成：语言事件是从客户端发送到服务器的 一组 T-SQL。例如，如果调用 ADO Command 对象（其 CommandText 属性设置为 T-SQL 查询，CommandType 属性设置为 adCmdText）的 Execute 方法，则查询将作为语言事件提交给服务器。同样，如果将 CommandType 设置为 adCmdTable 并调用 Execute 方法，则 ADO 将生成一个内部查询（它将选择 CommandText 属性标识的表中的所有列），并将它作为语言事件提交给服务器。另一方面，如果将 CommandType 设置为 adStoredProc，则调用 Execute 将使 ADO 向服务器提交一个远程过程调用请求，以执行 CommandText 属性中列出的存储过程。 为何要关心将请求作为语言事件还是作为 RPC 提交给服务器呢？通常，这是因为 RPC 的功能更为出色，特别是在重复调用具有不同筛选值的同一查询时。尽管 SQL Server 可以自动将普通的语言事件请求参数化，但这种能力非常有限。它从不尝试自动将某些类型的查询参数化。这可能会导致基本相同的查询产生不同的执行，从而只因为这些不同的执行提供不同的值，而导致在服务器上白白浪费计划编译的成本。这通常不是您所希望的结果 - 您希望针对查询的第一次执行编译一个新的计划，然后将该计划重复

用于具有不同参数的执行。 而 RPC 则通过显式参数化查询（而不是依赖服务器参数化查询）来支持计划重复使用。为过程的第一次执行生成一个计划后，随后的执行将自动重复使用该计划，即使它们提供的参数值不同。与通过语言事件调用存储过程相比，使用 RPC 调用存储过程不仅节省了计划编译所需的执行时间和 CPU 资源，还增强了 SQL Server 内存资源的利用率，因为它避免了冗余执行计划所浪费的内存。 在执行动态 T-SQL 时，通常首选 sp_executesql 而不是 EXEC() 也出于同样的原因。Sp_executesql 的工作方式是：使用指定的查询创建一个存储过程，然后使用提供的参数调用它。与 EXEC() 不同，sp_executesql 提供了一个允许您参数化动态 T-SQL 并支持计划重复使用的机制。使用 sp_executesql 执行的动态查询比使用 EXEC() 的查询能够在更大程度上避免不必要的编译和资源消耗。 TDS 从客户端发送到 SQL Server 的 RPC、语言事件和其他类型的请求被格式化为称作表格数据流 (TDS) 的 SQL Server 特定数据格式。TDS 是 SQL Server 客户端和服务器之间使用的“语言”。对于它的确切格式将不作介绍，但是，如果客户端要与 SQL Server 进行通信，就必须使用 TDS。 目前，SQL Server 支持三种版本的 TDS：TDS 8.0（适用于 SQL 2000 客户端）、TDS 7.0（适用于 SQL Server 7.0 客户端）和 TDS 4.2（适用于 SQL Server 4.2、6.0 和 6.5 客户端）。完全支持所有 SQL Server 2000 功能的版本只有 TDS 8.0。其他版本保持向后兼容。
服务器端 Net-Library 在服务器端，客户端请求最初由 SQL Server 为侦听特定网络协议而建立的侦听器接收。这些侦听器由服务器上的网络库以及服务器端的 Net-Library（在它们与服务器之间提供管道）构成。您可以使用 SQL Server 网络实用程序配置服务器侦听的协议。SQL Server 与客户端支持同样范围的网络协议（处理群集的情况除外）。对于群集化的 SQL Server，只有 TCP/IP 和命名管道可用。 SQL Server 为侦听客户端请求所使用的每个网络协议设置一个线程，并使用 Windows 的 I/O 完成端口机制等待和有效处理请求。从网络接收到 TDS 数据包时，Net-Library 侦听器将其重新汇编为它们的原始客户端请求，并将这些请求传递到 SQL Server 的命令处理层，即开放式数据服务 (ODS)。 将结果返回到客户端 服务器在准备将特定客户端请求的结果返回时，将使用最初接收请求时所用的网络堆栈。它通过服务器端 Net-Library 将结果发送到相应的网络协议，随后这些结果将通过网络以 TDS 格式返回到客户端。 在客户端上，客户端 Net-Library 将从服务器接收的 TDS 数据包从 IPC 层重新汇编，并将其继续转发到初始化该请求的 API 或对象库。 小结 尽管涉及了所有组件，但 SQL Server 客户端与服务器之间的往返过程却相当快 - 特别是在使用内存 Net-Library 时，亚秒响应时间非常普遍。构建和调整您自己的 SQL Server 客户端应用程序时，以下几个与数据相关的问题值得注意： • 如果应用程序与 SQL Server 运行在同一台计算机上，则建议您使用共享内存 Net-Library（如果尚未使用它）。基于共享内存 Net-Library 的连接通常比其他类型的连接快很多。在注意上述内容的同时，还应：始终全面测试解决方案并将它与其他可行方案进行对比，这样才能判断它是否确实更好或更快。事实胜于雄辩。 • 由于客户端在第一次连接时将指定给特定的 UMS 计划程序，并只有在断开连接后，才会摆脱该计划程序的控制，因此确保在应用程序与服务器建立的连接上均衡分配工作负荷非常重要。工作负荷不均衡可导致不必要的 CPU 争用并降低资源使用率。 • 在服务器上配置的默认网络数据包大小以及客户端在连接时指定的网络数据包大小将直接影响它们在服务器上所需的内存量和分配内存的池。对服务器进行扩展性和速度配置时，应记住这一点。还应记住，默认情况下，ADO.NET 应用程序的网络数据包大小比 ADO 应用程序的更大。
• 通常，在向服务器发送请求时，应首选 RPC 而非语言事件。为此，应在使用的 ADO 或 ADO.NET 对象中设置相应的属性。 • 执行动态 T-SQL 时，应在可能的情况下使用 sp_executesql 代替

EXEC()。唯一例外的情况是，当使用 EXEC() 的功能将查询片断连接而成的动态查询字符串的大小超过单个本地变量的存储大小时（这种情况非常少见）。 • 当遇到客户端问题，并且怀疑它可能和连接服务器时所用的对象库或 API 有关时，可以使用的一个故障排除技巧就是更改所用的客户端机制，这样可以将问题归结为特定的组件。例如，假设您升级 MDAC 并开始在 SQL Server 错误日志中看到 17805 错误，这表明客户端 ADO 应用程序发送的 TDS 数据包的格式不正确。您可能尝试让应用程序转为使用 ODBC 的 OLE DB 提供程序，如果您可以较为容易地做到这一点，应看看该问题是否与 SQLOLEDB 提供程序有一定的关系。相反，如果基于 ADO 的应用程序一直通过 ODBC 进行连接，则可以切换到 SQLOLEDB，看看这是否能解决问题，或至少帮助您缩小问题的范围。 • 同样，在对连接问题进行故障排除时，更改正在使用的 Net-Library 有时会有所帮助。如果使用 TCP/IP，命名管道也许值得一试。例如，如果 DHCP 服务器出现问题，并且没有有效的 IP 地址，则您将无法使用 TCP/IP 连接到 SQL Server。通过切换到命名管道，可以快速地将问题归结为 TCP/IP 特定的因素上。另一方面，如果在切换 Net Library 后仍存在同样的问题，则可以排除 Net-Library 方面的问题。问题的原因可能是服务器已关闭，或在您与服务器之间的某处网络基础设施无法正常工作。最后，还可以容易地更改应用程序使用的 Net-Library，而不必更改应用程序本身，这样就为您提供一个帮助缩小问题范围的工具。尽管从长远角度而言，使用某一特定 Net-Library 并不可行，但让客户端临时使用它可以帮助您缩小连接相关问题的范围。 2008-11-28 11:33:37