Optimizing Adaptive Server Anywhere Performance Over a WAN

A whitepaper from iAnywhere Solutions, Inc., a subsidiary of Sybase, Inc.

Contents

Introduction

This document discusses tuning performance for an Adaptive Server Anywhere network server (version 8 or higher) that is running over a WAN. TCP/IP is the protocol of choice for WAN implementations and is the main focus of this document. Note that network performance tuning is an iterative process of determining what works best with a particular application and network.

The recommended steps for optimizing the performance of your application are:

1. Measure the performance of the network you plan to run your application on.
2. Based on the network performance, tune your application to reduce requests and/or to reduce the amount of data transferred.
3. Based on the network performance, tune Adaptive Server Anywhere's server options and connection parameters to maximize performance.
4. Consider the implications of your tuning on LANs or same-machine operation.

In many cases, tuning an application can have a more significan performance impact than tuning Adaptive Server Anywhere.

Measuring network performance

Latency and throughput can be used together to describe the performance of a network. Latency refers to the time delay between when one machine sends a packet of data and the second machine receives the data (for example, if the second machine receives the data 10 ms later than the firs machine sent it, the latency is 10 ms). Throughput refers to the amount of data that can be transferred in a given time (for example, if a one machine sends 1000 KB of data, and it takes 5 seconds for all of it to be received by the second machine, the throughput is 200 KB/s). On a LAN, latency is typically less than 1 ms, and throughput is typically more than 1 MB/s. On a WAN, the latency is typically significantl higher (perhaps 5 ms to 500 ms), and the throughput is typically significantl lower (perhaps 4 KB/s to 200 KB/s).

You can measure network latency between two machines by the round trip time reported by the system's ping utility. The round trip time is the latency to transfer data from one machine to a second machine plus the latency to transfer data from the second machine back to the firs machine. You can measure network throughput by copying a fil of a known size of at least 200 KB from one machine to a second machine and timing the copy. This copy could be performed as a regular fil copy, using FTP, or by downloading a fil using an Internet browser.

To get reasonable Adaptive Server Anywhere performance on a network that has high latency, but reasonable throughput, the number of requests made by the client must be minimized. If a network has reasonable latency, but low throughput, the amount of data transferred between the client and server must be minimized.

Tuning your application to improve WAN performance

Changing your application to implement the following suggestions generally reduces both the number of requests and the amount of data transferred. These suggestions can also help improve application performance in any environment (standalone, LAN, and WAN).

- Move logic that requires many SQL statements from the application to one or more stored procedures or functions.
- If the same basic SQL statement is used more than once, consider preparing the statement once and executing it multiple times with different parameters.
- Ensure your application is not executing queries or SQL statements more often than necessary. If a particular query is executed more than once, consider changing the application to cache the results the firs time the query is executed, and then use the cached values instead of re-executing the query.
- Combine queries that get one property, function, or variable value into one multiple-column query. Suppose you want to execute the following queries: `SELECT current user`, `SELECT @@servername`, and `SELECT connection_property( 'BytesSent' )`. Instead of executing three queries, use a single query:

```
SELECT current user, @@servername, connection_property(
        'BytesSent' )
```

- Avoid doing joins in the application with multiple queries if it is possible to do the join in the server. If an application executes a query, and then executes a second query using the result from the firs query, you may essentially be joining multiple queries together in your application. If it is possible to do the join using one query instead of many, this can significantl improve performance. As a simple example, if your application executed the query `SELECT T.x FROM T WHERE <conditions on T> ORDER BY <order>`, and then for each value of T.x, did `SELECT R.y FROM R WHERE R.z = <value of T.x>`, then this can be combined into the single query `SELECT T.x, R.y FROM T, R WHERE <conditions on T> AND R.z = T.x ORDER BY <order>`.
- When working with third-party development tools (for example, PowerBuilder, Visual Basic, or Delphi), check whether there are any application-specifi settings you can make to increase your performance. For example, in PowerBuilder, changing the BLOCK connection property sometimes improves application performance over a WAN.

Reducing the number of requests

The following suggestions may reduce the number of requests, which is particularly beneficia if your network has high latency. These suggestions may also improve performance on other networks.

- Use bound columns to fetch data instead of using get data. This reduces the number of requests, especially when fetching the firs row from a cursor.
- Combine SQL statements into a batch (a sequence of SQL statements separated by semicolons that are all executed as if they were one statement) where possible.
- Disable autocommit and commit explicitly *only* when necessary to eliminate extra commit requests (a commit or rollback should be performed in most cases before waiting on user input to avoid excessive blocking).
- Use wide fetches (also referred to as increasing the row set size) and wide inserts (also referred to as using arrays of parameter values). These fetch or insert multiple rows in one request, instead of using one request per row. Prefetch also fetches more than one row per request.

Reducing the amount of data transferred on the network

The following suggestions generally reduce the amount of data transferred, which is particularly beneficia if your network has low throughput. These suggestions are unlikely to decrease performance on other networks.

- Consider using stored procedures for large database queries. This eliminates the need to send a large statement to the server across the network by allowing you to send a small CALL statement to execute the query.
- If you know that you will only be fetching the firs row (or firs few rows) of a query, add a FIRST or TOP *n* clause to the query. If you want to skip the firs few rows in a query, use the START AT clause. These clauses prevent rows that you won't use from being transferred, particularly if prefetch is enabled. Adding these clauses can also help the query optimizer know how to execute the query more efficientl .

Tuning Adaptive Server Anywhere to improve WAN performance

The following suggestions can tune Adaptive Server Anywhere so your application runs better on a WAN, independent of your network's latency and throughput.

- Use the client connection parameters **"CommLinks=TCPIP(host=*w.x.y.z*;port=*p*);DoBroadcast=none"**. These options are often required when connecting over a WAN since the UDP packets used to fin servers that do not include these options are blocked or are not routed by fir walls. Even if you can connect without these options, using these options may speed up connection times. A hostname can be used instead of *w.x.y.z*. The port option is not required if the server is using on the default TCP/IP port (2638).
- Consider increasing the liveness timeout if your connections are dropped because of liveness. Increasing this value does not improve performance, but

you should increase this value if your connection keeps timing out due to liveness. The option can be set at the client (using the LivenessTimeout connection parameter) or the server (-tl). Using the LivenessTimeout connection parameter, you could change the liveness timeout only for WAN connections.

- Consider increasing the packet size (the server's -p option or the client's CommBufferSize connection parameter) to the largest value that makes sense for the WAN/protocol combination. 1460 (the default) is ideal for TCP/IP over most Ethernet hardware, including where the client and server both have Ethernet cards and are separated by a WAN. Higher values (for example 5000 or 10 000) can improve the performance of large fetches, multi-row fetches, or BLOB operations for networks that support transmitting data in larger packets. Note that most gigabit Ethernet hardware is capable of supporting 8 KB of data in one transmission packet using jumbo frames, but must be configure to use larger sizes. This only benefit transmission over a WAN if the WAN is also capable of transmitting larger packet sizes. Using the CommBufferSize connection parameter, you can change the packet size only for connections that would benefi from a larger packet size. Using an inappropriate packet size has the potential to decrease performance.

In most cases the following suggestions reduce the number of requests, which is particularly beneficia if your network has high latency:

- Consider changing the prefetch behavior. When your application fetches a row, Adaptive Server Anywhere may prefetch additional rows depending on the cursor type and other factors. Prefetch can reduce requests and significantl improve performance, particularly on high latency networks when a large number of rows are fetched from a cursor. Note that absolute fetches, relative negative or relative 0 fetches, rollbacks, and certain get data operations can cause prefetch rows to be discarded and re-fetched, which decreases performance. Prefetch is best used on forward-only, read-only cursors, when more than just the firs row is fetched. If your application fetches many rows from each cursor and only uses fetch next operations, using the PrefetchRows and PrefetchBuffer connection parameters may improve performance. Prefetching many rows when the application only fetches part of the result set could potentially decrease performance. Additionally, an ODBC or OLE DB application can benefi from the PrefetchOnOpen connection parameter if many cursor opens are done.
- Consider using the LazyClose connection parameter, particularly when many cursors are opened and closed. This eliminates an extra request when closing a cursor. Note that using this parameter is not recommended for applications where cancel requests are common.

The following suggestions may be beneficia if your network has low throughput:

- Consider using communication compression (the server's -pc option or the client's Compression connection parameter). Using communication compression reduces the amount of data transferred, and the number of packets transferred, but it does not reduce the number of requests. Communication compression can be particularly beneficia on large fetches,

multi-row fetches, or BLOB operations. Note that communication compression uses more CPU and memory on both the client and the server. As a result, it can cause poorer performance in some cases, and in most cases decreases performance on a LAN. By using the client's Compression connection parameter, you can enable communication compression for WAN connections only.

- Consider using the ReceiveBufferSize and SendBufferSize TCP/IP protocol options on both the client and the server. These options pre-allocate memory in the protocol stack for sending and receiving TCP/IP packets. Preallocation of memory inside the protocol stack can increase client/server performance for network-intensive applications. The default value is machine dependent, and values in the range of approximately 65 536 to 262 144 bytes are reasonable for experimentation.
- If your application only uses the firs row from a cursor, disabling prefetch using the DisableMultiRowFetch connection parameter can improve performance. Instead of using DisableMultiRowFetch, you may get better performance by changing the application to use the FIRST clause on queries where you only fetch the firs row as suggested in the application tuning section above. Using DisableMultiRowFetch can cause poorer performance in many cases, including on faster networks.

Here are some suggested settings you can use to start tuning. Start with these settings and adjust them in your test environment to see how they affect performance. Try adding and removing the server options and connection parameters mentioned above and see how they affect performance. It is not an exact science and requires some trial and error to get the best performance in your particular network environment with your particular application.

☞ For information about potential LAN performance implications, see "Performance implications for LANs and same-machine operation" on page 7.

Database server command-line options

```
dbsrv9 -x TCPIP(SendBufferSize=100000;ReceiveBufferSize=100000)
   -n server_name ...
```

Client connection parameters

Note that PrefetchOnOpen is for ODBC and OLE DB only.

```
"ServerName=server_name; Compression=Yes;
CommLinks=TCPIP(Host=w.x.y.z; DoBroadcast=NONE;
SendBufferSize=100000; ReceiveBufferSize=100000);
PrefetchRows=50; PrefetchBuffer=1000; PrefetchOnOpen=Yes;
LazyClose=Yes; ..."
```

Note
The options listed here are communication oriented. There are other options to consider that can make the server more efficien in all environments (for example, cache size and database page size). Refer to your SQL Anywhere Studio documentation for more information.

Performance implications for LANs and same-machine operation

If your application is also going to be running in a LAN environment, don't forget to test your Adaptive Server Anywhere communication options there as well, as there may be a need to adjust these options for that environment.

All of the application tuning suggestions mentioned in this document typically improve or at least maintain performance in LAN and same-machine configu ations.

The following Adaptive Server Anywhere WAN tuning suggestions may cause poorer performance on a LAN or for same-machine operation. They should only be enabled for WAN clients, or after LAN testing has verifie they also increase performance on the LAN:

- Changing the liveness timeout can cause severed connections to go undetected for long periods of time.
- Changing the packet size can decrease throughput when set incorrectly.
- Enabling communication compression often results in slower performance on LANs where the increased memory and CPU usage outweigh potential throughput gains.
- Changing the prefetch settings (PrefetchRows, PrefetchBuffer and DisableMultiRowPrefetch). All prefetch settings have the potential to increase or decrease performance depending on the application and environment.

Legal Notice

Contact Us

iAnywhere Solutions Worldwide Headquarters One Sybase Drive, Dublin, CA, 94568 USA

Phone 1-800-801-2069 (in US and Canada)

Fax 1-519-747-4971

World Wide Web *http://www.ianywhere.com*

E-mail contact.us@ianywhere.com