ANTs DATA SERVER:

White Paper on Design and Use Release 3.4

ANTs Software

Table of Contents

WHAT IS THE ANTs DATA SERVER 3

HOW DOES THE ANTs DATA SERVER COMPARE WITH OTHER DATABASE SYSTEMS, SUCH AS ORACLE? 3

HOW DOES THE ANTs DATA SERVER COMPARE WITH “IN MEMORY” DATABASE SYSTEMS, SUCH AS ORACLE TIMESTEN? 6

WHERE SHOULD THE ANTs DATA SERVER BE USED? 7

HOW FAST IS THE ANTs DATA SERVER? 7

TEST 1 8

TEST 2 9

TEST 3 10

WHICH DATABASE PERFORMANCE PROBLEMS DOES THE ANTs DATA SERVER ADDRESS? 10

CONCLUSION 11

APPENDIX: 12

This document presents an overview of the ANTs Data Server, version 3.4. Comparisons are made to other database systems and advantages are indicated.. Some performance results are presented and internal architecture innovations are discussed.

What is the ANTs Data Server?

The ANTs Data Server is a fully transactional, persistent relational database server, supporting ANSI SQL, ODBC and JDBC database standards. "Transactional" means that SQL statements can be grouped into entities that are treated as atomic units. "Persistent" means that changes made by committed transactions are guaranteed to be permanent, even if a failure occurs at a later time.

The ANTs Data Server is especially designed for use in *high update volume* or *high contention* situations with *multiple database clients*. "High update volume" means that there is a high rate of inserts/changes/deletes to one or more tables. "High contention" means that multiple clients often compete for overlapping or related data. "Multiple database clients" means that there are at least ten or more active connections to the database with ANTs demonstrating more scalability with more users. In such cases, the ANTs Data Server is significantly faster than other databases. ANTs is ideal for database of choice for applications where performance is important, or multiple client connections are making frequent changes to the data.

The ANTs Data Server currently runs on Windows, Linux (32-bit Pentium and 64 bit AMD Opteron or Intel Xeon64) and Solaris. 64-bit Windows on Opteron/Xeon64 will be available soon. Additional platforms are planned in the future.

How does the ANTs Data Server compare with other database systems, such as Oracle?

The ANTs Data Server is similar in many respects to traditional database systems such as Oracle, DB2, or SQL-Server. Like many of these database systems, the ANTs Data Server:

- Supports the complete SQL query language and the ODBC, JDBC, and OCI (Oracle call interface) database APIs
- Supports stored procedures and triggers. Currently three stored procedure/trigger languages are natively supported: Oracle PL/SQL, Microsoft and Sybase Transact SQL.
- Is fully transactional and persistent. Durability of committed transactions is guaranteed, even in the event of a failure.
- Is a database server, not an application library
- Can handle data which is larger than the computer's main memory
- Can be used either as a networked server, or co-located with its applications.
- Supports online backup
- Can be monitored for administration, either by a provided SNMP agent or by SQL queries against server variables

However, the internal architecture of the ANTs Data Server is quite different from that of other database systems. Major differences include:

- *Multi-database* or *"Universal Compatibility"*. The ANTs Data Server provides native support for the SQL statements, stored procedure languages, and triggers of many popular database systems, including Oracle, Microsoft SQL-Server and Sybase. This means that applications on these databases can be migrated to ANTs with minimal effort. A migration support group in ANTs software can even assist with the migration, or perform it completely, as desired.

- *Lock-free internal datastructures.* All indexes, queues, and other internal ANTs datastructures are fully concurrent, with no lock waiting ever. This allows them to have much greater performance when multiple clients are updating the database. For example, any number of readers and/or writers can simultaneously access ANTs' B-Tree based indexes; no one ever waits for access. This is ANTs software proprietary technology and is the subject of granted and pending patents.

- *Almost no database row locking.* Like Oracle, the ANTs Data Server uses a *multi-version* concurrency model for data access, in which rows are never read-locked. But unlike Oracleand other databases, the ANTs Data Server rarely write-locks rows either.

  All databases other than the ANTs Data Server lock rows (if not larger entities) for writing. In these databases, if a transaction has a pending (not yet committed) change to a row, it will hold an exclusive lock on that row. The lock is typically held until the transaction is committed (or rolled back). Unlike other databases, the ANTs Data Server supports concurrent updates *even to the same row*, provided the outcome will be deterministic (independent of operation order). Only in race conditions, where the outcome cannot be predicted in advance, will locks be used, but properly designed applications should generally avoid race conditions[1]. The ANTs Data Server handles all of this automatically and transparently.

- *Compilation to machine language.* The ANTs Data Server includes a built-in compiler which converts SQL statements, and even entire stored procedures, into machine language functions. Then, the machine language function is simply called to execute the statement or procedure. Other databases only parse to an intermediate form, which must be interpreted each time the statement or procedure is executed. This is another reason ANTs is faster.

- *Efficient thread scheduling.* Although the ANTs Data Server is a multi-threaded process, no operating system thread primitives, other than thread creation, are ever used. The ANTs Data Server uses a single "worker" thread per processor; work items are scheduled into these worker threads by the ANTs micro-scheduler with no requirement for context switching or even stack switching.

- *Handling of data on disk.* Oracle and similar databases were architected at a time when main memories were small and expensive. Main memory was too small to hold a database of even moderate size, and so the principal repository for the data is the disk. Main memory in these systems is usually just a big cache for disk blocks. New or updated data must be pushed back to the disk eventually, but in the background.

  Note that the data on the disk in an Oracle environment may not be the most current data. The transaction log is the guarantor of persistence, since the data itself is not necessarily up-to-date on the disk. If a failure occurs with Oracle, the transaction log is automatically replayed upon startup to recover the data.

1 An example of a race condition is: two clients simultaneously set the same datum to two different values. The final result will depend on which client went first. But if two clients concurrently set a datum to the same value, then that is not a race condition, and the ANTs Data Server will allow them to run concurrently.

Like Oracle, the ANTs Data Server uses a synchronized transaction log to guarantee data persistence. But the ANTs Data Server uses main memory not as a big cache, but as the up-to-date location of new or recently changed data. Old data will eventually be pushed out to disk if there is insufficient room in main memory, but data that is continually in use need never be pushed out except at checkpoint times. And like Oracle, failure recovery is accomplished by automatic log replay at startup time. Periodic checkpoints, which can be scheduled automatically based on size of the log file, assure that the log file size (and time to recover from a failure) remains manageable, since a new log file is started at each checkpoint time[2].

- *Large numbers of clients are supported.* Unlike databases that require a TP Monitor, an Application server, or other techniques for many clients, the ANTs Data Server can easily handle hundreds, even thousands, of active clients, with superior performance. In fact, as the number of clients is increased, the opportunities for concurrent operation (and better performance) increase as well. Low per-client overhead, efficient scheduling of operations and network aggregation technology (see below) make this possible.

- *Multiple replication scenarios are supported.* The ANTs Data Server supports three forms of replication, any or all of which can be used as needed. None of these forms has a significant impact on the performance of the primary server[3]. These include:

    - *High Availability* replication with automatic failover. The ANTs server can automatically replicate to a second hot standby server, which will automatically take over should the primary server fail. All client connections are automatically and transparently failed over to the standby with no disruption. No committed changes are ever lost. And recovery from a failure is easy— just restart the failed server and it joins and synchronizes automatically.

    - *"Speed Clusters" for Disaster Recovery and/or reporting servers.* The ANTs server can automatically replicate to one or more *readable replicas.* A readable replica can be remotely located for disaster recovery without significant performance impact. Readable replicas can also be used for reporting, where it is desired to offload work from the primary ANTs server. (Of course, the ANTs Data Server is so fast, and handles concurrent read and write operations so well, that this is often unnecessary.)

    - *The ANTs Log Miner (ALM).* The ALM is an API provided for use where the other forms of replication above do not meet mission requirements. It allows you to inspect the ANTs transaction log, and optionally replicate some or all of the logged operations.

  The ALM can be either on the same machine as the primary ANTs Data Server or on another machine. Unlike the other supported forms of replication the ANTs ALM does not happen automatically, but it allows customized replication (or simply transaction log inspection) in almost any imaginable scenario. The Log Miner allows you to filter log entries ("show me only updates to these tables") and then choose which operations to replicate and where to replicate. If a replication target is another ANTs Data Server, then the transaction log record itself is sent over. Or, you can ask the ALM to generate an equivalent SQL statement, which can be used to replicate the operation to any SQL database such as Oracle.

- *Advanced connection management.* When ANTs is used as a networked server, database connections to it are automatically and transparently *aggregated* into a single physical TCP/IP connection per application. This substantially reduces networking overhead.

2 Although a checkpoint logically represents a snapshot of the database at a single instant of time, it is taken while the database continues to run, and is actually a "fuzzy" checkpoint

3 A server which is replicating will usually perform at higher speed compared with the same server and no replication.

These unique and advance features as well as some other differences are the reason for the remarkable performance of the ANTs Data Server. Additional technical details on many of the above features appear in the Appendix.

How does the ANTs Data Server compare with “in memory” databases such as Oracle TimesTen?

TimesTen and similar databases are primarily memory-only *program libraries,* as opposed to independent database servers. As a library, TimesTen is linked-in to a specific application and is private to that application.
Only threads from within that application[4] can access the database.
This has some significant limitations:

- All clients of the database must be a single process (or possibly several processes) on the same single computer as the database. This means that users cannot scale up capacity by adding additional computers, generally the most cost-effective way to increase capacity.

- There is no real “firewall” between the application and the data. A bug in the application has the potential of crashing the database, or of overwriting or corrupting the data in the database.

- There is no opportunity for true Highly Available configurations, in which clients can fail over to a replica database in the event of a failure. TimesTen supports replication (at significant performance cost), but switchover to use the replica must be accomplished manually.

Of course, as an in-memory database, TimesTen cannot handle data larger than the size of memory. *Everything*—client code and data, database code, database data—must fit into main memory. Finally, TimesTen uses conventional datastructure locks and row locks, and therefore loses performance when there is contention among multiple database sessions.

By comparison, the ANTs Data Server (like most Relational Databases) is most commonly an independent server process, which uses networking or shared-memory IPC (inter-process communication) to communicate with its clients. Clients can be on separate machines or on the same machine, or both. Client failures cannot crash the server, or corrupt data within the database. The data within the database can be larger than main memory. Yet the ANTs Data Server is actually *faster* than TimesTen in many cases. Generally speaking, an application which uses Times Ten in any mode other than as an internal library will run faster on the ANTs Data Server using client-server IPC. Applications which link with TimesTen will usually run at least as fast when migrated to the ANTs Data Server, and may be much faster if there is significant update activity.

A feature of the ANTs Data Server is that it can be *embedded* in the same process as its client application, much like TimesTen. This is not a special version of the ANTs database: it is purely a configuration option for an ANTs client. At client connect time, you request that the ANTs server be embedded in your process, and that happens automatically. Yet even in this mode, the ANTs Data Server is not an in-memory database— it can handle data larger than main memory, and other applications can still connect to it via networking or shared-memory IPC. Even replication and failover is supported.

4 TimesTen does support a mode in which multiple processes on a single computer can share a common database, but at significant performance cost. Most, if not all, of the published TimesTen performance results are for TimesTen as a program library.

Where should the ANTs Data Server be used?

The ANTs Data Server is a general-purpose relational database. It conforms to the SQL[5] , ODBC, and JDBC standards, and so can usually be substituted for another relational database with relatively little effort. In “database of record” scenarios, customers have chosen to use the ANTs Data Server as a *second* “hotspot” database: to retain another relational database as the primary database of record, but move certain performance-critical tables over to the ANTs Data Server.

The ANTs Data Server can be used to best advantage in applications with some or all of the following attributes:

- *The cost of database licenses and maintenance is > $50,000 per year.*
- *The data is frequently updated.* Since the ANTs Data Server is essentially lock-free, it can achieve much greater concurrency and performance than other databases when there is a significant load of INSERTs, UPDATEs, and/or DELETEs.
- *The data is read frequently.* The ANTs Data Server is more than 3 times the speed of other databases even for read-only data, It has even greater advantages when many users are reading the same data.
- *The cost of performing each transaction is important to profitability.*
- *Many servers are deployed in distributed locations.*
- *Real time analytics are needed.*
- Applications are co-located with and distributed from the database.
- *There is data contention.* If multiple clients tend to access the same or related data, then the ANTs Data Server will be ideal. Consider, as one example, an application that wishes to keep running totals or other on-the-fly statistics. Each update is accompanied by an addition to one or more summary quantities. Suppose also that multiple clients are doing this concurrently. In any other database, the running totals will become a severe performance bottleneck, because only one transaction at a time can have a pending addition. But the ANTs Data Server recognizes these operations as non-race-conditions, and allows any number of simultaneous add/subtract operations against a single quantity. More generally, “contention” can be understood as any situation in which an ordinary database would need to wait for locks, either rows or indexes. The ANTs Data Server does neither.

How fast is the ANTs Data Server?

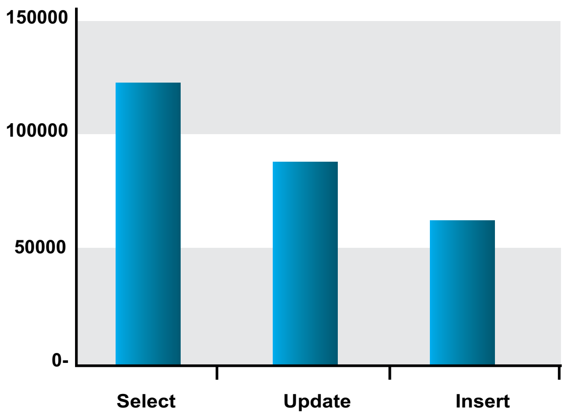
We present some selected examples of the speed of the ANTs Data Server. The following basic SQL tests were designed to be as basic as possible in order to determine the performance of the core database engines. Client systems executed the following SQL DML statements for basic SQL tests. These tests and drivers are included with ANTs download. More detailed information can be found in the Basic SQL Operations Benchmark paper available from the ANTs software website, www.ants.com.

5 SQL-92 “entry level” plus some features from higher levels, plus select features from SQL-99 (e.g. BLOBS and CLOBS).

Test 1

Read/Write transactions test

(1b.) 2 CPU Dual Core 2 GHz Opteron, ops/sec



#connections	Selects/s	Updates/s	Inserts/s
1	5596	4760	4511
4	20737	15003	13605
16	57590	36625	29393
32	78495	52356	40099
64	105027	72616	49112
128	113550	83530	62343
256	121146	91662	67759
512	131022	96364	72113
1024	130461*	96362	70883

*Client Limited

Database Server Systems Used: 2-CPU, AMD Opteron 2 GHz, 4.0 GB DRAM, Dual RAID controllers log file to one RAID drive, and data to the other. Windows 2000 Professional. *Select was client limited.

SELECT Statement

```
SELECT SELECT P_NUM_AVAILABLE FROM P_INVENTORY
WHERE P_PART_NAME =<randomly chosen part>
```

In the SELECT test, each client connection queries a single table in a database, which then returns the result to the client.

INSERT Statement

```
INSERT INSERT INTO P_INVENTORY

VALUES (<unique part number>, 1000000,
'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA')
```

In the INSERT test, each client connection inserts ten rows (using 10 INSERT statements) into the same table of the database and then commits the transaction (this sequence is counted as ten operations).

Update Statement No Contention

```
UPDATE P_INVENTORY SET P_NUM_AVAILABLE = P_NUM_AVAILABLE -1
WHERE P_PART_NAME =<randomly chosen part>
```

Since each client connection always updates its own distinct row, there is never any contention in the UPDATE test.

UPDATE Statement with Contention

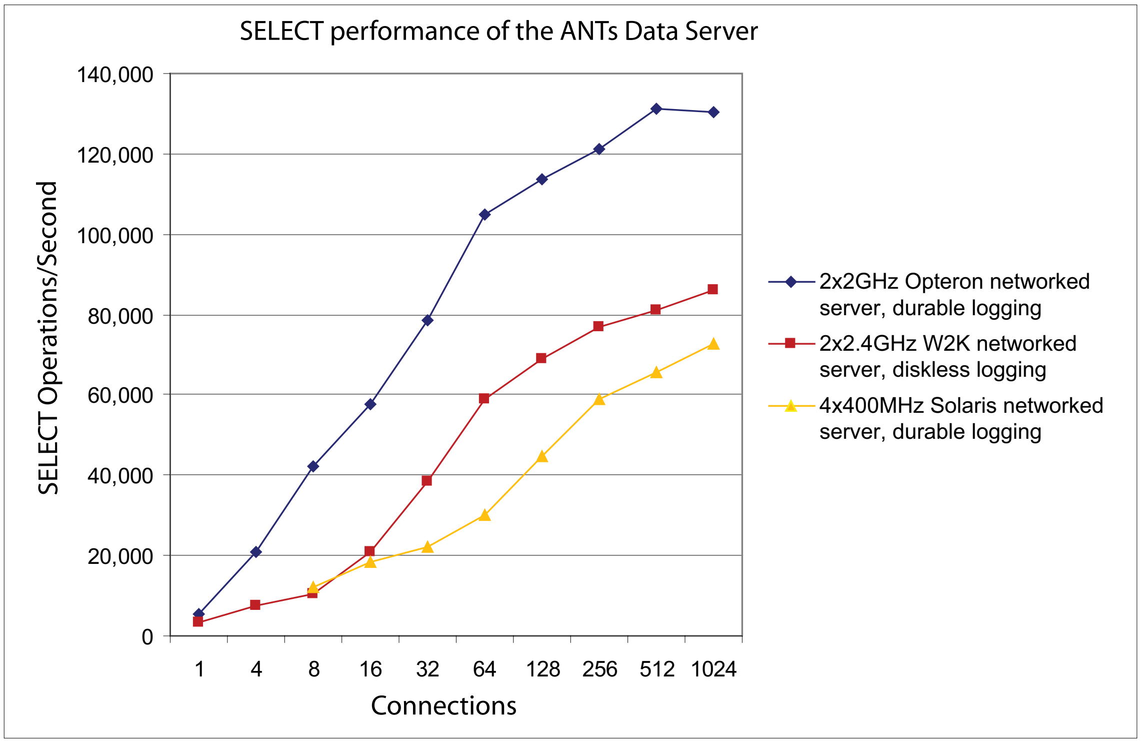
```
UPDATE P_INVENTORY SET P_NUM_AVAILABLE = P_NUM_ AVAILABLE -1
WHERE P_PART_NAME =<randomly chosen part> OR P_PART _NAME =<2nd
randomly chosen part>
```

In the UPDATE with contention test, each client connection updates two randomly chosen rows of a common 100-row table (using a single SQL UPDATE statement), and then commits the transaction. All tests were performed using ‘Read Committed’ transaction isolation.

The UPDATE test (without contention) was performed in autocommit mode: there was an implicit commit operation after every update. The UPDATE-with-contention test did not use autocommit: an explicit commit was issued after every update operation. The complete source code of the benchmark program used is available.

Test 2

The chart below gives some examples of ANTs Data Server query performance, for various platforms and logging options[6]. Note that in the co-located server case shown, only one of the four CPUs is actually dedicated to ANTs Data Server processing; the other three

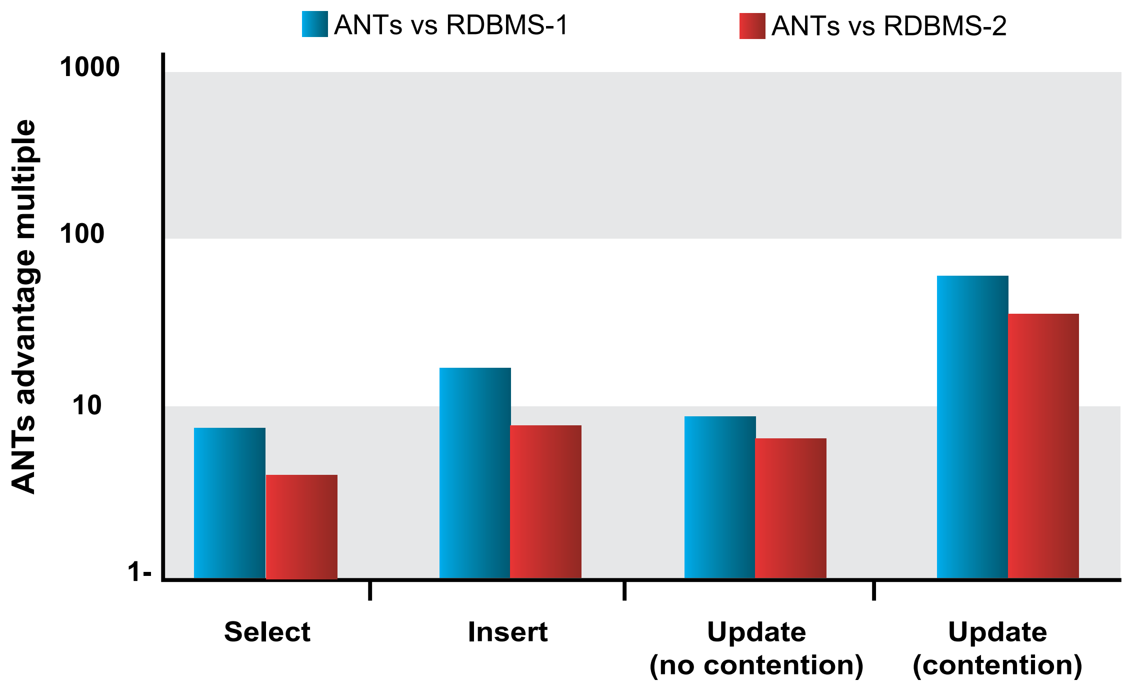


are used primarily for client processing. Although not shown in the graph below, a 4-way 2GHZ networked server will achieve more than 250,000 `SELECT` operations per second.

6 With durable logging, a transaction commit request does not return to the application until the transaction’s log record is physically written to disk, thus guaranteeing durability. With diskless logging, the log records are kept only in memory; this configuration option is useful for applications for which loss of some committed changes is tolerable.

Test 3

As the following comparison chart demonstrates, the ANTs Data Server delivers extraordinary performance compared to other RDBMS products. “RDBMS-1” and “RDBMS 2” are two leading relational database systems. All experiments were performed with the ANTs Data Server on a 4-way Windows 2000 networked server, using durable logging. “RDBMS-1” and “RDBMS 2” were run on the identical hardware. All comparisons are for 512 total client connections, except for “`UPDATE` (contention)” which was for 128 total connections. Complete details of these experiments are presented in the Basic SQL *Operations Benchmark* paper referenced above.



Which database performance problems does the ANTs Data Server address?

There are a number of possible reasons for poor database performance. These include:

Reason for poor performance	Explanation	Can the ANTs Data Server improve performance?
Locking, at the table row and/or the index level	Many updates (INSERT, UP-DATE, DELETE operations) per second to the same table(s) from multiple connections. Many updates and queries are forced to wait for locks.	**Yes**. The ANTs Data Server is virtually lock-free; all connections have unimpeded access.

Reason for poor performance	Explanation	Can the ANTs Data Server improve performance?
Complex SQL statements	The application uses lots of JOIN operations, or other SQL statements which are inherently complex to calculate	**Yes.** The ANTs Data Server is very fast at most of these, in part because it compiles SQL statements into machine language functions (other databases such as Oracle must interpret a query plan each time the statement is executed).
Networking Overhead	Performance is limited by networking overhead between clients and the database server	**Yes**. ANTs Data Server network aggregation technology greatly improves network performance in heavy-load situations, by transparently combining multiple messages into a single physical network message.
Replication or Backup Overhead	Performance is limited by the need to replicate data (for failover in case of a failure), or by the overhead of online backups.	**Yes**. ANTs Data Server replication impacts performance by less than five percent. Incremental Online backups, which will be available in the next release, will be similarly inexpensive.
Disk I/O Overhead	The database is so much larger than available main memory that SQL operations are constantly accessing the disk. (NOTE: this is less common than one might think. Even very large databases tend to use only a small portion of the data at any given time, and keep that in use data in main memory.)	**No**. Although the ANTs Data Server can accommodate databases larger than main memory, at least the “working set” (data currently in use) should be able to fit into main memory, or ANTs Data Server performance will be no better than that of other databases.
Distributed Database Overhead	There are two or more separate databases in use, or a federation of databases, and transactions often span more than one database. Expensive distributed-transaction protocols are likely performance bottlenecks here.	**No.** The ANTs Data Server can not increase distributed database performance.

Conclusion

The ANTs Data Server is a very fast general purpose database system, and can replace traditional databases such as Oracle. ANTs compatibility even makes this easy to do, with little or no application or SQL changes. ANTs can also be deployed as a “helper database”, moving only the hotspot tables into ANTs (perhaps using the ANTs Log Miner to propagate changes back to the traditional database as required). ANTs is fully featured, with high availability and disaster recovery support built in, as well as online backup and other administrative features. ANTs will greatly lower your cost of ownership and improve your performance.

Appendix: Some Technical Details of ANTs Data Server Operation

The internal architecture of the ANTs Data Server includes a number of innovative (and in some cases patent pending) features that account for its superior performance. These include:

Lock-free internal datastructures

Database users are usually familiar with "database locking", a traditional means of ensuring correct operation by (typically) taking out locks on rows. But databases other than the ANTs Data Server also lock internal datastructures such as indexes, queues, etc., and these locks can significantly limit performance when the database is highly dynamic.

Consider the most common index structure, the "B-Tree", used by Oracle, the ANTs Data Server (in modified form), and most other databases. To update a B-Tree index, most databases must acquire locks on nodes of the tree. Concurrent changes to the tree by other clients are forced to wait for their turn. If there is a high volume of updates, these index locks can significantly reduce performance. A particular contention point is often the root of the tree, which can be a lock-contention hotspot.

In multiprocessor servers, internal locks cause a second form of performance degradation due to CPU hardware cache coherency. The physical datum representing the lock must continually be moved from one CPU's cache to another, as threads on different processors contend for the lock. This often causes even more performance loss than simple lock waiting, as we have demonstrated by experiment.

By contrast the ANTs Data Server uses no internal datastructure locks at all. Accesses to indexes, queues, and other internal datastructures are fully concurrent; no one ever waits for a lock. Any number of readers and/or writers can be simultaneously accessing the datastructure, without interference. And since there are no lock words, there is less cache coherency performance loss.

How is this possible? Programmers are taught as a first principle that if concurrent threads are sharing mutable data, "mutual exclusion" locks must be used to protect the data. Without such locks, corruption is possible. But ANTs software's patent pending technology allows lock-free operation, yet with guaranteed correctness.

Modern computers provide hardware instructions to help us. These instructions read and write a word in certain ways atomically. The particular operation we need is of the general form:

Compare the contents of a specified word to the value X. If it is equal to X, then atomically change it to another value Y. If it is not equal to X, leave it unchanged. In any case, tell me whether you changed the value or not.

In Intel computers, this instruction is called Interlocked Compare and Exchange. In Sun Microsystems Sparc computers it is called Compare and Swap. The instruction is always guaranteed to be atomic, which means that no one else can change the word between the time you test its value and set its value. Such instructions are heavily exploited by the ANTs Data Server to support lock-free concurrent access. These instructions are used purely to detect concurrency conflicts (two threads trying to modify the same word at the same time), and not to "roll our own locks". For example, if two threads attempt to insert a new key into the same place in an index, the second thread in will detect that someone else has beat it to the punch, and find an alternative place to insert. This sort of conflict happens rarely in a large index, but is handled correctly when it does occur.

However, atomic instructions can detect conflicts only against a single word in memory, and therefore cannot handle all types of concurrency conflicts. For example, atomic instructions can protect insert operations on ANTs indexes, but they cannot protect delete operations. If an index key is deleted, the key is simply marked “invalid”, but is not actually deleted. This means that the index will fill up with more and more invalid entries, unless it is periodically cleaned up. Also, an index tree can become unbalanced— some paths in the tree are much longer than other paths— which is undesirable because some searches will then take too long.

To deal with such problems, the ANTs Data Server uses phased execution. All operations in the ANTs Data Server are divided into two groups, each of which can be mutually concurrent. For ANTs indexes the groups are:

Phase 1 Operations	Phase 2 Operations
Insert into an index tree Search an index tree Mark a deleted index entry “invalid”	Delete invalid entries and rebalance an index as needed

The ANTs Data Server alternates continually between Phase 1 and Phase 2, usually thousands of times per second. All real work is done in Phase 1; Phase 2 is primarily for cleanup. In fact, most Phase 2’s do nothing. Indexes are scheduled for cleanup/rebalancing only as necessary, when they accumulate too many invalid entries or become too unbalanced. (And even then, only subtrees of an index are processed, not the entire index.) This is patent pending technology.

Almost no database row locking

Most modern database systems, other than the ANTs Data Server, lock rows for update. In these databases, if a transaction has a pending (not yet committed) change to a row, it will hold an exclusive lock on that row. The lock is typically held until the transaction is committed (or rolled back). Should another concurrent transaction wish to write (or for many database systems, even read) the row, it must wait.

Oracle avoids read locking by means of an architecture known as multi-version concurrency control. With most databases, if you wish to read a value on which some other transaction has a pending update, you wait for the other transaction to commit and then read the new value. With Oracle, you do not wait, but read the existing current committed value. The ANTs Data Server has adopted this approach, so like Oracle, the ANTs Data Server never uses read locks. But unlike Oracle, the ANTs Data Server rarely uses write locks either, yet still guarantees correct performance. In fact, the ANTs Data Server will always yield a same end result as Oracle, for any combination of concurrent transactions.

The ANTs Data Server automatically detects commutative operations and allows them to run concurrently. Operations are “commutative” if the outcome of doing all of them does not depend on the order of execution. Examples of commutative operations include:

- Adds/subtracts to a numeric value
- Setting different columns of a row
- Setting the same column of a row to the same value

These and other operations are allowed to run concurrently in the ANTs Data Server, but not in other database systems. Concurrency, and therefore performance, is improved, yet correct operation is still guaranteed. This is patent pending technology.

Commutative add/subtract operations enable an application to maintain transactional real-time subtotals or rollups of rapidly changing numeric data. Each time a detailed entry is inserted/modified/deleted, one or more corresponding subtotals can be modified in the same transaction. With other databases, this would become a severe bottleneck, since only one of these sub-total update operations could be pending (issued but not yet committed) at any time. In the ANTs Data Server, any number of them can be concurrently pending. This allows a form of "dynamic OLAP"—analytic processing of data in real time—if the application is written to maintain the rollup values. And since read locks are never used, it is possible to compute the sum of some column of a rapidly changing table without delaying other concurrent operations on the table.

On the other hand, operations that are not mutually commutative are by definition race conditions, since the end result will be timing dependent. The ANTs Data Server will make a transaction wait if, for example, it attempts to set a cell to some value, and there is already a pending set to a different value. Or there is already a pending add/subtract against that cell (sets and adds are not commutative). Or the transaction is setting a column with a uniqueness constraint, and there is already a pending set of that identical value. In each of these example cases, the result depends on the order of operations, and cannot be predicted in advance. A well-written application will probably avoid doing such things. But in the (hopefully rare) event that they are do happen, the ANTs Data Server will force one or more sessions to wait, to guarantee correctness.

Preparation by compilation

Although the ANTs Data Server obtains most of its enhanced performance through non-locking and efficient thread scheduling, it gets an additional boost because it compiles all SQL statements, and all stored procedures, to machine language functions. ANTs software has developed a completely in-memory and in-process compiler/assembler/linker to do this. Both the ANTs ODBC client and the ANTs Data Server have prepare caches, so if the statement (or a similar statement) has recently been prepared it will not be re-prepared.

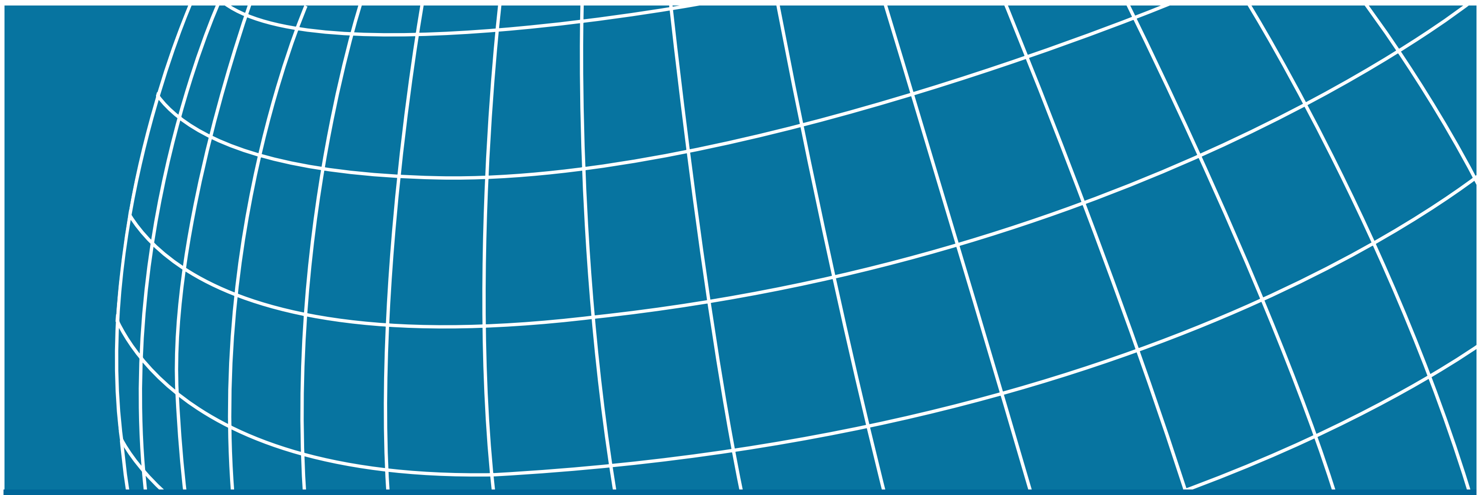
Advanced Network Connection Management

The ANTs Data Server can support literally thousands of connections ("sessions") from clients. Each session is encapsulated in a Session object, but is not assigned permanently to any thread. The next available CPU worker thread will be assigned to requests from any session.

However, thousands of connections does not mean thousands of network sockets for the ANTs Data Server. This many sockets would cause serious problems in most operating systems, not to mention the ANTs Data Server. Instead, the ANTs ODBC driver and ANTs Data Server support network aggregation. All sessions from a given client application are physically transmitted on a single socket between client and server. This saves socket overhead, and allows longer (and therefore more efficient) TCP/IP messages to be built up. Of course, this is all transparent to the application.

For more information call ANTs Software today. We will arrange a briefing to show you how a low-risk implementation of the ANTs Data Server would dramatically reduce your database costs.

Call 650.931.0537 or visit http://www.ants.com/savings or e-mail info@ants.com



ANTs Software

Document published April, 21, 2006