SAS® / Firebird Performance Testing Strategy

Bill Oliver

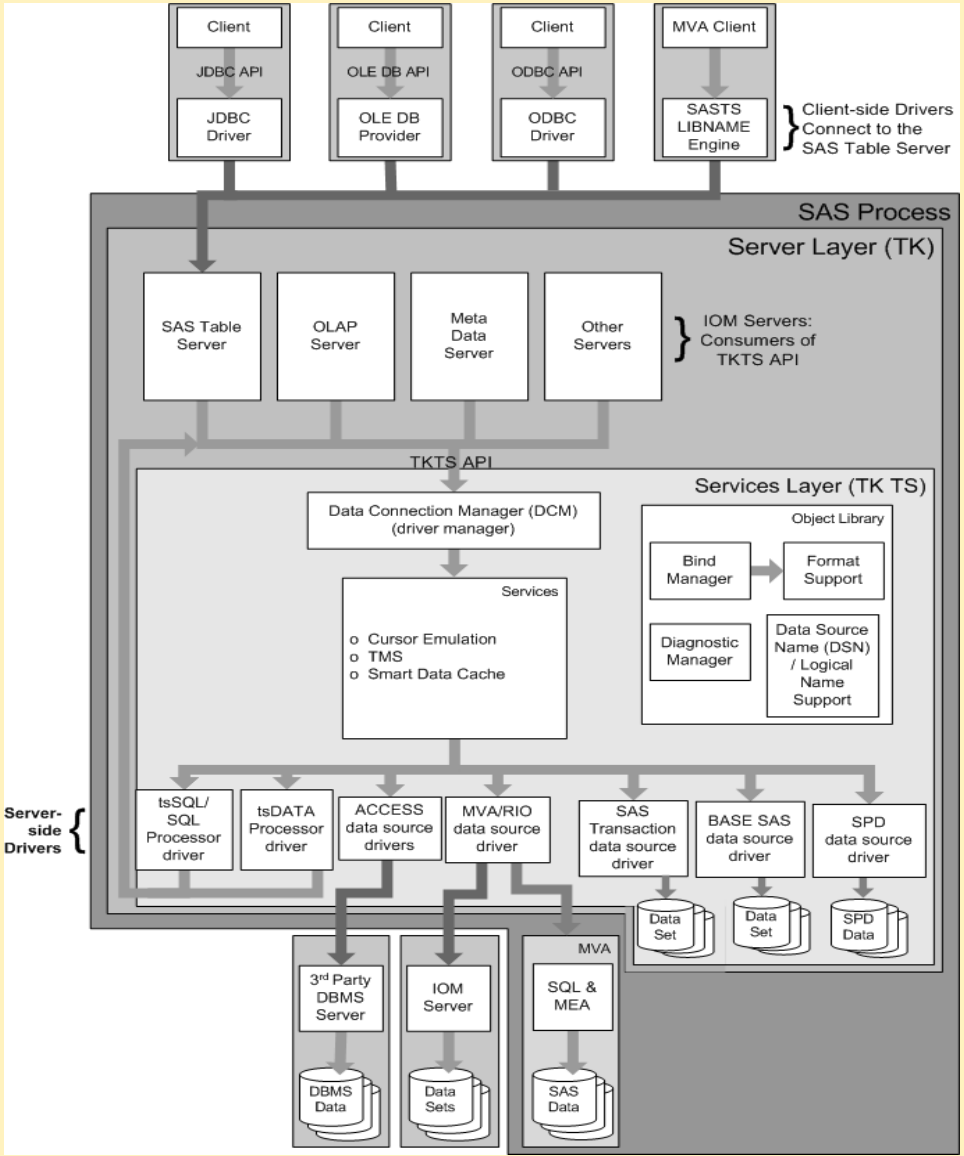
Product Specialist

November 2005

Agenda:

- Thumbnail sketch of SAS®'s use of Firebird.
- Very tiny discussion of SAS users' expectations.
- Real world discussion of customer use cases.
  - A J2EE/JDBC Application
  - A metadata Server (in-process consumer of TKTS API)
  - A forecasting solution (TSSQL, batch SAS jobs, reporting)
- Performance testing methodology and SAS Resources
  - Where are we today?
  - Performance test tools and methodology.
- Next steps and thanks!

SAS® 9.2 Table Server Big Picture



Who are my customers?

- External OEM customers, using our JDBC driver.
- SAS® Solutions – J2EE / JDBC.
- In-process consumers of TKTS API. TKTS API calls our Firebird driver, which calls Vulcan. Only 1 customer here so far, but with an outsize importance.
- Other SAS products which require ODBC or OLE-DB client side driver.
- Firebird is a small (but important) piece of an overall system that has to be tested for performance.

Our goal is for TKTS to become the preferred data access technique for SAS developers, with Firebird the preferred transactional store.

Case Study #1 – A JDBC/J2EE Application

- JDBC/J2EE Customer, uses Hibernate.
- Was developing/testing on Firebird 1.5 for next release.
- Deployment requires SAS Table Server infrastructure and SAS Table Server JDBC driver.
- Functional defects resolved, performance defect open on JDBC driver.

Current (problematic) results
Jaybird/Vulcan:
0: Total Time 57,375 ms
0: Total Time 54,452 ms (2nd run)

Table Server
0: Total Time 347,050 ms
0: Total Time 296,346 ms (2nd run)

Case Study #2: A Metadata Repository

- Shipping product
- Current Version:
  - Originally architected on top of non-transactional Base data sets
  - Transaction support was provided at the application level
  - In memory caching added to meet performance needs (IMDB).
- With TKTS/Firebird ...
  - A lead internal customer of ours for our next release.
  - Will rely on TKTS/Firebird for concurrency and transaction support
  - But... has not been rewritten to take advantage of a relational data model. Still does its own joins.
  - We've provided QMDB – an in memory database on top of Firebird as a work around for performance issues.

Case Study #3: A Forecasting Solution

Data Mart	*Model Data*	*Model_data_calib*
Rows (millions)	14598	14598
Data Size (GB)	915	1714
Index size (GB)	211	359

- Data model contains >= 1 row for each SKU.
- A small customer site has 4 TB of data, a large data store exceeds 7 TB.
- Model_data updated during weekly ETL process.
  - Fact tables updated
  - model_data is combination of simple facts extracted from inventory, sales, price, and promotion fact tables, calculated columns derived from simple facts, and aggregations over product or geography dimension.
  - Primary key is composite key of geography, product, and date surrogate keys.
  - Moving window of 156 weeks
- Reporting environment uses SAS Web Report Studio, Information Map Studio – J2EE architecture with Hibernate
- Current focus is 10+ concurrent users, hoping for 200 concurrent users
- Reports are pre built SQL queries, no ad-hoc queries permitted

Case Study #3: A Forecasting Solution

- How'd they do that?
  - Typical setup is a heterogeneous mixture of SAS® data sets and transactional tables (DB2, for example).
  - SAS data sets are partitioned by value - e.g. date or model group. SAS Base data sets and SPDE data sets do not support partition by value, so this is done at the application level.
  - Mix of analytical data (in SAS data sets) and transactional data is painful, since transactional queries often need that one column that is stored in SAS data sets. They would really like to move to a homogeneous, transactional, model.
  - In some cases desired performance can only be achieved through careful crafting of data creation and extraction algorithms. For example, they maintain many tables in sort order and use hand-optimized extraction algorithms.

Case Study #3: A Forecasting Solution

- Their requirements:
  - Support efficient extraction of fact data based on composite key (geo/prod/date)
  - Support efficient movement of window on week boundary – oldest week's data is archived and removed from the fact table and the new one is added
  - ETL process must complete during ***weekly window of 8 hours***
  - Efficient concurrent read and update access to **disjoint** sets of data contained in a single logical table (and associated objects) that exceeds 1 TB in size
  - Query mechanism should support efficient access through logical model – want to get away from application specific partitioning schemes
  - Data recovery to be no more than 50% of maintenance window
- Anticipated Difficulties:
  - Bulk loading support in Firebird, specifically turning off indexes and constraints
  - Lack of partitioned stored seen (by internal customer) as deficiency
  - Lack of clustered indexes, to leverage sorted tables
  - Would like ability to “pin” tables in memory – small tables used frequently

Time check!

- Where are we today, with respect to Firebird2?
- Close with performance test methodology, and thanks.

Vulcan Performance Goals

- Perform within 5% of Firebird 1.5 for single-user cases
- Provide 3x throughput boost on a 4-way CPU
- Single User tests are promising (see next chart)
- Additional requirements now provided by SAS® Forecasting Solutions

Benchw, an open source benchmark

Tuning Notes: All tests run with these settings:						
	defaultDbCacheSize=2048					
	SortMemBlockSize = 20971520					
	SortMemUpperLimit = 20971520					
	database page size=16834					
1st run Vulcan						
loadstart	12:48:44 PM	46124.0000000			***vs fb1.5***	***vs fb2***
indexstart	1:02:14 PM	46934.0000000	loaddur	810.0000	*106%*	*94%*
q0start	1:07:58 PM	47278.0000000	indexdur	344.0000	*107%*	*98%*
q1start	1:08:06 PM	47286.0000000	q0dur	8.0000	*200%*	*160%*
q2start	1:08:09 PM	47289.0000000	q1dur	3.0000	*100%*	*150%*
q3start	1:08:39 PM	47319.0000000	q2dur	30.0000	*111%*	*94%*
q4start	1:08:59 PM	47339.0000000	q3dur	20.0000	*87%*	*91%*
q4end	1:09:46 PM	47386.0000000	q4dur	47.0000	*104%*	*102%*
				108.0000	*106%*	*101%*
2nd run Vulcan						
q0start	1:09:46 PM	47386.0000000			***vs fb1.5***	***vs fb2***
q1start	1:09:51 PM	47391.0000000	q0dur	5.0000	*125%*	*250%*
q2start	1:09:53 PM	47393.0000000	q1dur	2.0000	*67%*	*100%*
q3start	1:10:13 PM	47413.0000000	q2dur	20.0000	*95%*	*91%*
q4start	1:10:33 PM	47433.0000000	q3dur	20.0000	*87%*	*95%*
q4end	1:10:55 PM	47455.0000000	q4dur	22.0000	*69%*	*100%*
				69.0000	*83%*	*100%*

threadtest.c Examples

- Btsload, 4-way windows 2003 server
- essentially a read only test (best case for no shared cache)

threadtest.c, our multi-user workhorse

- No share cache Vulcan Embedded
  C:\SASv9\vulcan\bin>threadtest vul.fdb
  Started 16 threads ...
  100000 rows selected from each of 16 tables in 15.172000 seconds.

- FB2A3EMBED
  C:\fb2a3embed>threadtest.exe TT.FDB
  Started 16 threads ...
  100000 rows selected from each of 16 tables in 130.453000 seconds.
  Fb2a3embed locked to 1 cpu using Intel's imagecfg
  Started 16 threads ...
  100000 rows selected from each of 16 tables in 315.000000 seconds.

- FB2a3 Classic
  Started 16 threads ...
  100000 rows selected from each of 16 tables in 55.266000 seconds.

- Firebird 1.5 embedded
  127 seconds when using 4 cpus
  75.375000 seconds (locked to 1 cpu)

Locking to 1 cpu helped Firebird 1.5 embedded but not Firebird2.

threadtest3.c

- Read/write test, worst case for no Share cache Vulcan
- No Share cache Vulcan:
  C:\SASv9\vulcan\bin>threadtest3 vul.fdb 16
  Starting 16 threads to update table
  1000 rows inserted and 16 threads each selected 500 rows in 23.110000 seconds.
- Firebird2a3, Classic:
  1000 rows inserted and 16 threads each selected 500 rows in 29.375000 seconds.

Methodology

- Hypothesis testing: Try to break down performance testing into unit test cases whenever possible and control the variables.
- Baseline performance works best, usually Firebird 2 in our cases.
- Have also benchmarked against MySQL (not recently) and SAS® 9.1 's PROC SQL.

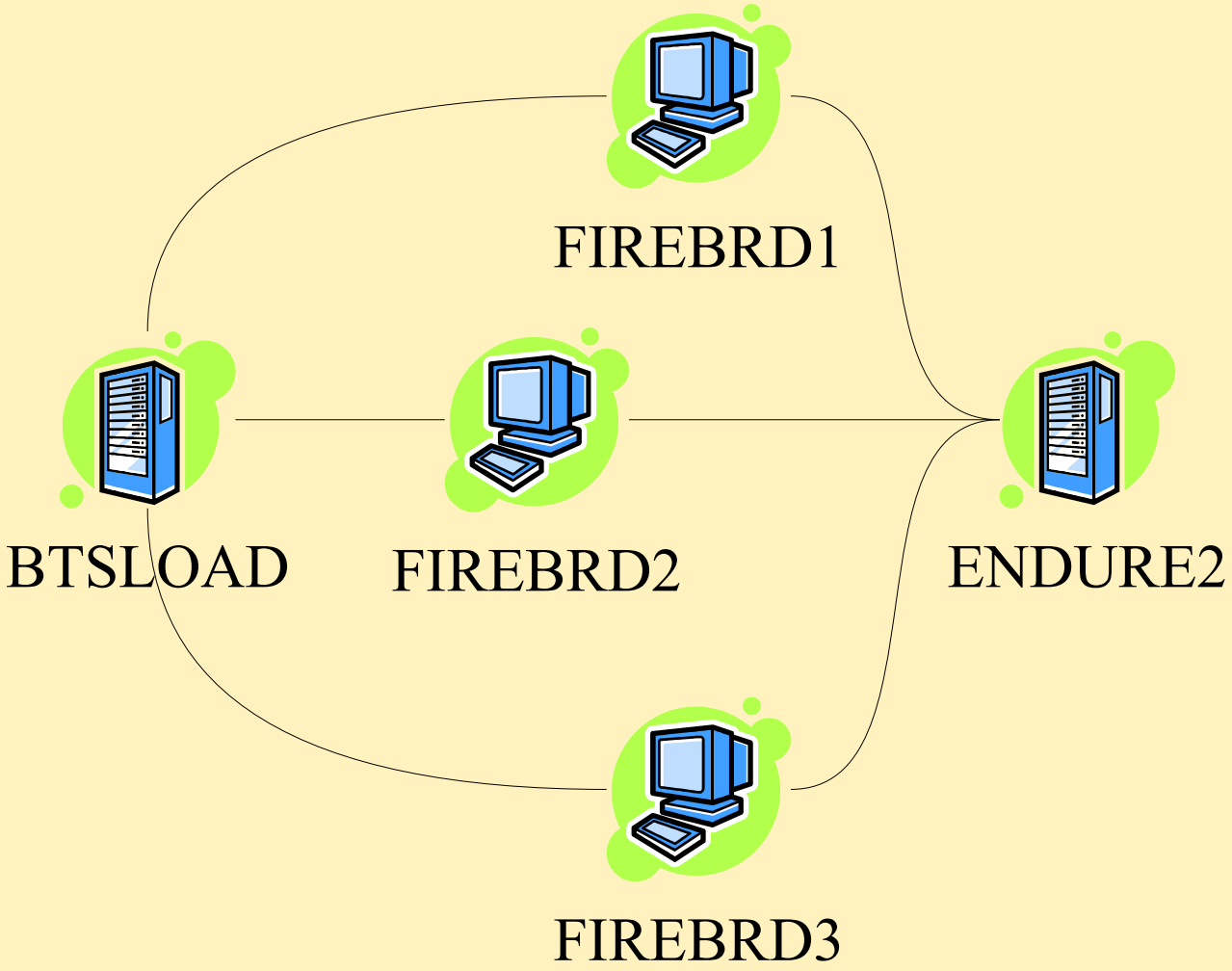
Baseline Performance Testing

- We've developed a test system that times queries and archives the times for trend line reporting. It also provides roll up reports.
- Currently, we've implemented SQL queries for TKTS Base and SPDE, and PROC SQL queries.
- Plan is to add Vulcan queries at some date in the future.

Performance / Endurance Testing

- We also have a Load Runner lab, that we've been using primarily to test long running server applications.
- Java Interface, currently JDBC driver is suspected bottleneck for throughput
- Current best results for SAS Table Server with embedded Vulcan:
  - 3 days, 200+ users, no think time (really a stress test)
  - Have tested 500 users for brief time periods.
- Expectation is 1000 concurrent users.
- Performance testing on Windows, S64 minimum.

SAS® Load Runner Performance Lab



Progress to Date

- Multi-user testing has focused on SAS® Table Server, with embedded Vulcan
- Vulcan has run stably for 3 days at 200 users
- Peak loads of 500 users have been demonstrated
- Tests are transactional, modeled on an internal customer's MySQL data model.

Test Environments

- ENDURE2: Windows 2003 Server, 4-way Xeon with Hyper threading (8 logical CPU's).
- BTSLOAD: Windows 2003 Server, 4-way.
- SIRIUS: Solaris 64-bit 8-way for performance testing
- Other test machines for LIX, LAX, AIX, H64, HUX, etc.

Thanks from the SAS® Team!

- Tom – Mr. Stored Procedure
- Steve – Mr. Lock Manager
- Gary – Mr. Cursor
- Bill – Mr. Test
- David S. – Mr. Manager
- Jim B. – Mr. Driver

Don't forget to tell Tom how nice his transitions are...

The Power to Know®