

Running SAP® Applications on Sybase's ASE Database

A Technologist's Perspective on ASE's Resource Efficiency to Deliver Low Total Cost of Ownership

Peter F. Thawley
Senior Director & Architect
Office of the CTO
Sybase, an SAP Company

TABLE OF CONTENTS

- 1 Introduction
- 2 Efficiency — From the Inside Out
 - 2 Gaining Resource Efficiency through Hybrid Threading in the ASE Kernel
 - 4 Ensuring Business SLAs through *Virtualized Resource Management*™
- 7 Conclusions

TABLE OF FIGURES

- 3 Figure 1. ASE 15.7's "Virtual Server Architecture" using a Hybrid Threading Model
- 4 Figure 2. Virtualized Resource Management in an ASE Deployment
- 5 Figure 3. Virtualized Resource Management of Memory Caches in an ASE System
- 5 Figure 4. Fetch & Discard Cache Strategy Optimizes Memory Residency of Data in ASE
- 6 Figure 5. Variable-size Disk I/O Optimizes Storage Access for Concurrent OLTP & DSS

INTRODUCTION

In today's highly competitive global economy, enterprise business applications need to be deployed on an extremely reliable and high-performing database platform that ensures data availability and transaction processing as data volume, number of users, and business complexity continue to grow. Sybase's Adaptive Server® Enterprise (ASE) is a high performance relational database management system for mission-critical, data intensive environments that has been meeting these needs for nearly three decades.

Sybase® ASE has been a technology leader since its first release in the late 1980s as one of the first database platforms offering innovations such as a multi-threaded architecture, stored procedures, database triggers, and two-phase commit. Its' early technology leadership guided many large enterprises, especially in the financial services sector, to embrace ASE for mission-critical, high-volume transaction processing applications. In its' most recent Wave Report for Enterprise Database Management Systems, Forrester Research positioned Sybase ASE in the "Leader" category¹, and stated that "Sybase ASE is known for its reliability, robustness, and availability, supports large applications with hundreds and thousands of concurrent users, and is well-suited for customized transactional applications."

From those of you new to the Sybase's technology, Sybase has always been a bit of a "heretic" in the database industry. Prior to Sybase's founding in 1984, most transaction processing was done on large mainframes, predominately using non-interactive, batch applications. Sybase's founder, Bob Epstein, had a different idea — to democratize information by building a database engine that could support transaction processing applications for an "on-line" community of users, predominately using Personal Computers and Workstations across local and wide area networks. The idea of an "On-line Enterprise", as Sybase referred to it, was poised to explode because of the speed and relative agility it offered for large, often decentralized, enterprise companies around the world to grow their businesses and better service their customers.

However to enable the On-line Enterprise required a radical shift in database engineering because at the time, the state-of-the-art in open-systems hardware was considerably underpowered by today's standards. Servers had a single processor — Symmetric Multi-Processing (SMP), cores, and hardware threads hadn't even been invented yet! Processors ran over 100 times slower — 25 MHz versus 2.5 GHz today! Databases were much smaller, yet servers had 10,000 times less memory — 64 Megabytes versus 640 Gigabytes, or more, today!

So Sybase set out to build a relational database engine that could support hundreds of concurrent users, running transactional types of applications, accessed by client computers running across a network. To do this required a singular focus on resource efficiency and TCO. While 25 years later we may have more powerful hardware, Sybase has always kept resource efficiency and TCO as core tenants because it just didn't seem right to ask customers to upgrade hardware just because the database software didn't emphasize efficiency.

This white paper will review the key technologies and capabilities provided by Sybase ASE that make it ideal for running SAP applications because of its high performance and low TCO.

EFFICIENCY — FROM THE INSIDE OUT

Over the years, database software engineers and vendors have long debated the generalized notion of efficiency in database software. Most vendors took the position that a traditional separation of responsibilities between the operating system (OS) and the database software (DBMS) was ideal. For them, this meant the OS took on the core roles of process & user scheduling, memory management, and disk & network i/o while the DBMS focused entirely on query optimization and execution. This made perfect sense historically because timesharing systems on the mainframe had to juggle processing requirements across many applications of varying workloads. However with Sybase's sight set firmly on the needs of an on-line and geographically distributed enterprise, it would have been impossible to provide the high OLTP (On-line Transaction Processing) performance for large numbers of concurrent users under this approach.

For Sybase, the path was clear — we had to effectively build a mini-operating system within the DBMS where the DBMS kernel did its' own scheduling and dispatching of user tasks, disk and network i/o, and memory management, amongst all the other things that databases must do.

Gaining Resource Efficiency through Hybrid Threading in the ASE Kernel

When we first released the ASE product back in 1986-87 (when it was formerly known as the Sybase SQL Server), there were no such things as threads in operating systems. Since Sybase's goal was to build a high performance database engine that could support 100's to 1,000's of concurrent users on uniprocessor systems, we built our own "threads" that were tightly bound to the ASE kernel. This made for very low memory requirements for each user, under 50 Kilobytes at that time, and lightening fast context switches between users, all without the overhead of the operating system. To this day, we still find that context switching between ASE tasks remains significantly faster than could be achieved if we used OS threads for each user.

Over the years, Sybase recognized that a process-centric model where we provided our own threading made for highly efficient query processing but wouldn't leverage more modern processor designs that offer significant hardware parallelism through multiple cores and threads. Accordingly, Sybase altered its kernel architecture to bifurcate the threading model between query processing and system services. This would allow us to retain the efficiencies of our internal threading for users with its low memory requirements and fast context switching while leveraging native operating system threads for system services to increase parallelism within the processors' cores and threads. This also has the positive side-effect of reducing latency for i/o and messaging related services where latency is key to good performance.

Perhaps more importantly to IT and end-users alike, having our own kernel complete with its own internal threads and scheduler allows ASE to do some very innovative things that enable applications to more predictably meet their Service Level Agreements (SLAs) to the business community they serve. To better understand all of these concepts, Figure 1 depicts the architecture of ASE and its unique and innovative approach to resource efficiency.

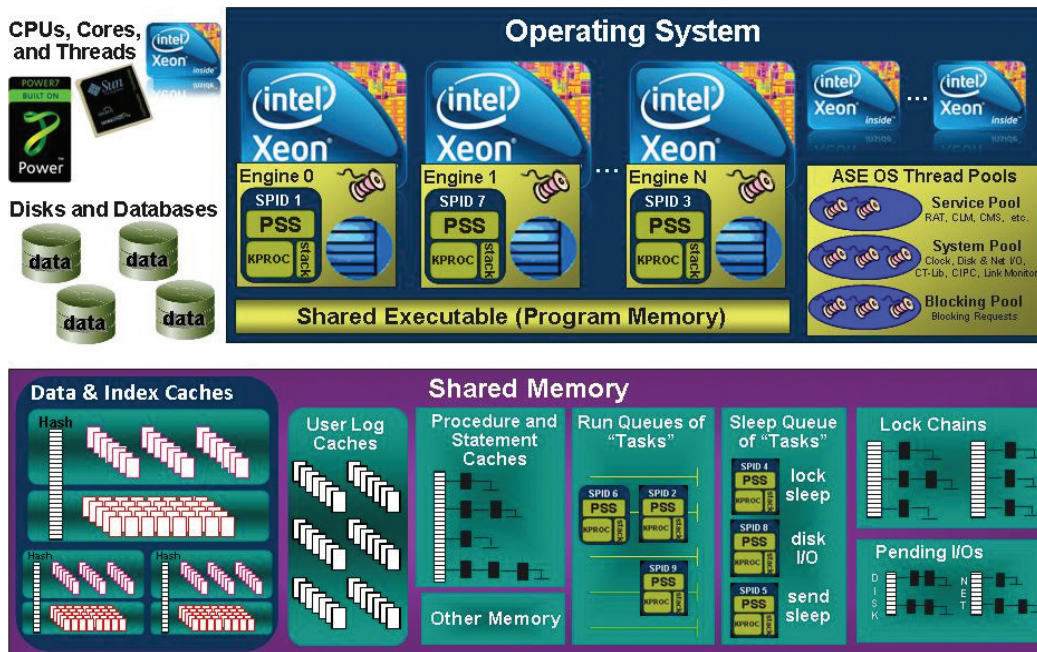


Figure 1. ASE 15.7's "Virtual Server Architecture" using a Hybrid Threading Model

As Figure 1 shows, ASE has the concept of an "Engine" which is an OS thread that provides the run-time execution vehicle to run users' queries. The user is represented within the ASE kernel as a SPID (Server Process ID) which has a number of data structures (PSS and KPROC above) associated with it as well as the SQL queries sent to ASE on its behalf. The ASE Engine uses its own scheduler to choose the next user query to service by looking for the highest priority user on its "Run Queues". When that user must wait for a resource like a database lock or an event to complete like a disk or network i/o, it is placed on a "Sleep Queue" until that resource is available or that event completes. Once ready, the user is "woken" and placed on the appropriate "Run Queue" to wait for an Engine to continue processing its' request. As you can imagine, this approach is incredibly lightweight, effectively taking a few assembler instructions to context switch between users.

Also shown, ASE also uses a configurable number of OS threads in each of three "Thread Pools" to provide each of the various system services required by a database. This provides flexibility in fine-tuning performance of services such as replication agents, disk i/o services, and network i/o services by leveraging more system CPU capacity (core and threads) as required to meet the applications' SLAs. It is this hybrid threading approach of using internal ASE threads for users' query processing and native operating system threads for system services within the database that make ASE highly resource efficient while still providing the performance and scalability that businesses demand. While all of this demonstrates that ASE provides the database industry's most resource efficient processing architecture, the reality of IT and business today is that databases must provide mechanisms to help ensure SLAs. Sybase ASE is unique in its ability to provide a resource reservation model for specific named applications to ensure enough system capacity is available to meet the performance requirements within the business SLAs!

Ensuring Business SLAs through Virtualized Resource Management™

It is now commonplace for customers to have many different applications, often each with their own workload characteristics, serviced by the same DBMS instance. Different workloads place wildly different demands on a DBMS though — some might be OLTP applications with strict response time requirements and which generally touch a very small amount of data at one time while others might be reporting or dashboard applications which often scan incredibly large amounts of data to summarize information. Since ASE controls decisions about which user task to run in our own scheduler, we have done some very innovative things to optimize the system, both for optimal throughput (e.g., transactions per second) and query response time. For example, when a user has been waiting for a transaction lock, we give it preference by basically running it at a higher priority. More interesting though to technologists and business users alike are the capabilities ASE provides to ensure specific levels of system capacity (CPU) to different applications. The key innovation Sybase created is known as *Virtualized Resource Management™* (VRM).

As Figure 2 below shows, DBA's managing an ASE deployment can group engines together as a unit called an "Engine Group". Applications (or users) can then be bound to those groups such that the execution of all requests made by those applications (or users) are performed only by engines within that group. In this common, but powerful example below, applications running OLTP transactions with strict performance SLAs could be segregated from applications running DSS (Decision Support) workloads which are almost always less critical in comparison. This approach to virtualizing resources is entirely managed within the ASE kernel itself and offers the dynamic elasticity necessary to enable the system to cope with temporary spikes in demand by moving engines between groups dynamically at run-time.

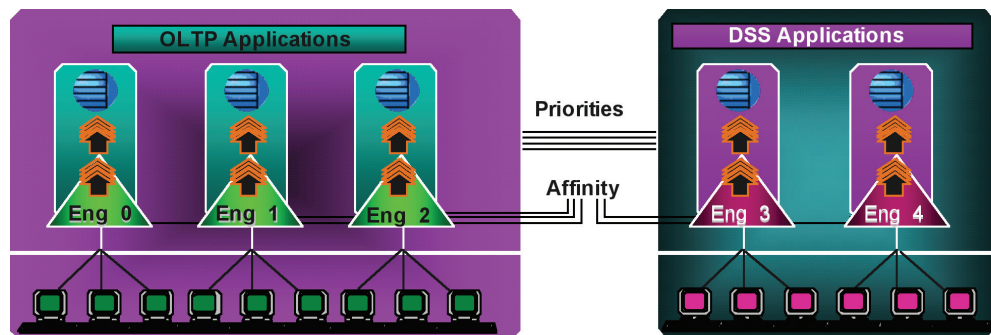


Figure 2. Virtualized Resource Management in an ASE Deployment

This "on-demand elasticity" of resources virtualized within the DBMS kernel itself is unique within the database industry. It can provide a significant reduction in TCO by allowing resources to be balanced as required to meet applications' peak load requirements without sizing every server for each application's peak load requirements as one normally would.

So far, we have been discussing the reduced TCO that ASE's Virtualized Resource Management technology brings to bear in the context of CPU usage to help ensure business applications' performance SLAs. However, there are other resources that are critical to increasing the efficiency of a DBMS for reduced TCO. The next most critical resources at which we need to look are memory and the storage subsystems. Memory in DBMS systems is used for many purposes but certainly one of the key usages related to performance SLAs has to do with the "caches" used to hold data and index pages in memory for fast access. In general, databases want to cache data in memory if it has a high probability of being reused or if the data itself is used in business critical processes which have very fast response time requirements. It may seem obvious, but it is important to state that the opposite is equally true — that is, the DBMS should not want to cache data in memory if there is a low probability of it being reused.

To facilitate these goals, ASE provides the flexibility to create multiple “named caches”, independently sized, to which tables and/or indices can be bound as shown in Figure 3 below.

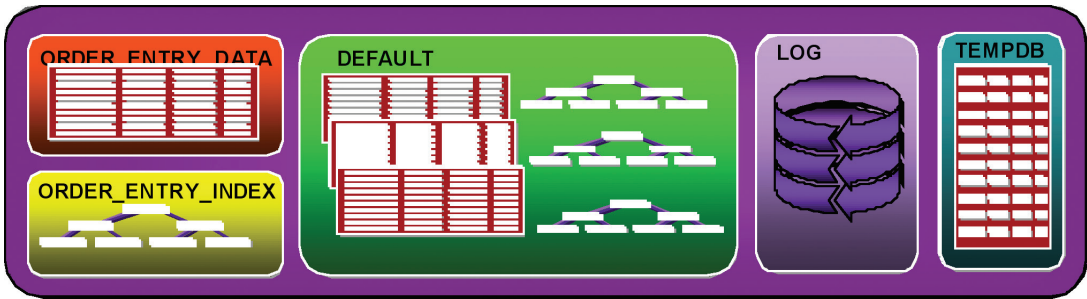


Figure 3. Virtualized Resource Management of Memory Caches in an ASE System

This provides a model to guarantee memory consumption for specific high priority data. For example, you might want to cache certain tables such as reference data which is used constantly by many different applications. To ensure that this data is memory-resident, one can simply create a “named cache” of the right size and bind the appropriate table(s) to it. As requests bring that data into memory, it will eventually be fully cached and will never be aged out of the cache. By selectively creating multiple caches to optimize memory usage, ASE helps to minimize the amount of memory required by the system to maintain a specific performance SLA by making sure the “right” data is cached, for as long as necessary.

The inverse may also be important to implement. For example, if you have a table that is involved in low priority business processes, why should the system waste a lot of memory caching it when you may not care if transactions or business processes which access it take longer amounts of time. In this case, it would be perfectly reasonable to create a very small cache to which that table is bound to ensure it never consumes much memory in cache.

Recall on the previous page that databases should not cache data that has a low probability of being reused or is not critical to the SLA of a business process or transaction. ASE provides a way to enable this through a capability referred to as “Fetch & Discard” caching which is an optimization especially useful when a system has differing workloads it must service. Typically, large sequential scans or full table scans bring large amounts of data into memory that generally will not be used again.

The ASE query optimizer recognizes this pattern and will generally choose a “Fetch & Discard” cache strategy to ensure that large scans do not “push out” other, more important (hot) data. In Figure 4 below, data brought in from disks are stored on the “MRU” (Most Recently Used) side of this logical diagram of a cache if the ASE optimizer thinks this page will be reused in the future. Alternatively, if it is following the Fetch & Discard strategy, the buffer will be put on the “LRU” (Least Recently Used) side of the diagram so it stays in cache for a much shorter amount of time before the buffer is reused for other data coming in from the storage subsystem.

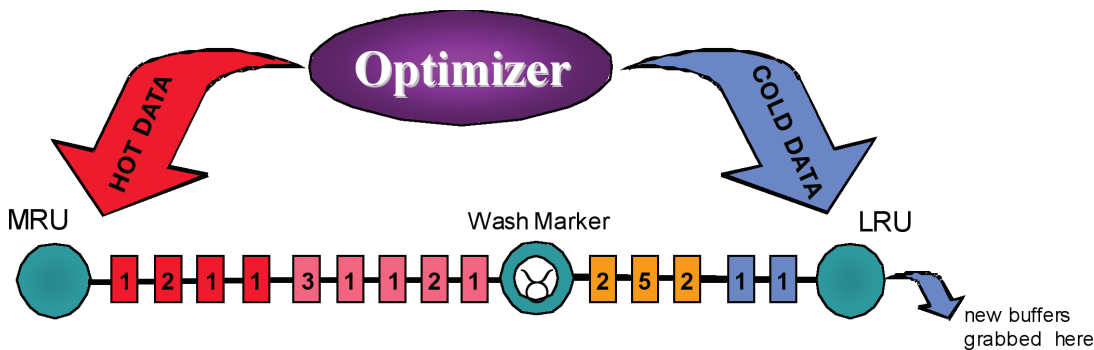


Figure 4. Fetch & Discard Cache Strategy Optimizes Memory Residency of Data in ASE

Finally, to optimally support differing OLTP and Decision Support (DSS) workloads on the same system at the same time, ASE has optimized the storage access layer to be able to do the small I/O sizes for OLTP workloads and large I/O sizes for DSS workloads. This is beneficial to OLTP workloads because OLTP transactions tend to touch small numbers of rows, often from a relatively small number of tables. Therefore, smaller i/o sizes tend to minimize the buffer cache memory requirements as well as helping to ensure better cache coherency since only small amounts of cache are “displaced” (i.e., aged out) every time another page from disk is brought into cache.

On the other hand, large DSS system scanning very large tables can not possible get an adequate amount of bandwidth and performance from the storage subsystem unless they can issue very large i/o's. This is because most magnetic disk-based storage systems have two upper bounds on performance — I/O's per second (IOPS) and Throughput. Since the fastest magnetic disks on the market spins at a rate of about 15,000 revolutions per minutes, typically you can only get about 180 IOPS from each disk. Since this is roughly independent of the i/o size, it is easy to see why doing larger block i/o of 128KB will yield significantly more i/o throughput than say an 8KB i/o.

To support this optimization for mixed workloads, ASE provides a mechanism known as buffer pools. In essence, a single cache can be split in multiple pools of differently sized buffers. These are a function of ASE Server's configured page size where buffers can be either 1, 2, 4, or 8 pages in size. At a page size of 2KB, this implies i/o sizes ranging from 2KB, 4KB, 8KB, or 16KB. At a page size of 16KB, this implies i/o sizes ranging from 16KB, 32KB, 64KB, or 128KB. Figure 5 below shows a sample named cache configured with both 2KB and 16KB buffer pools.

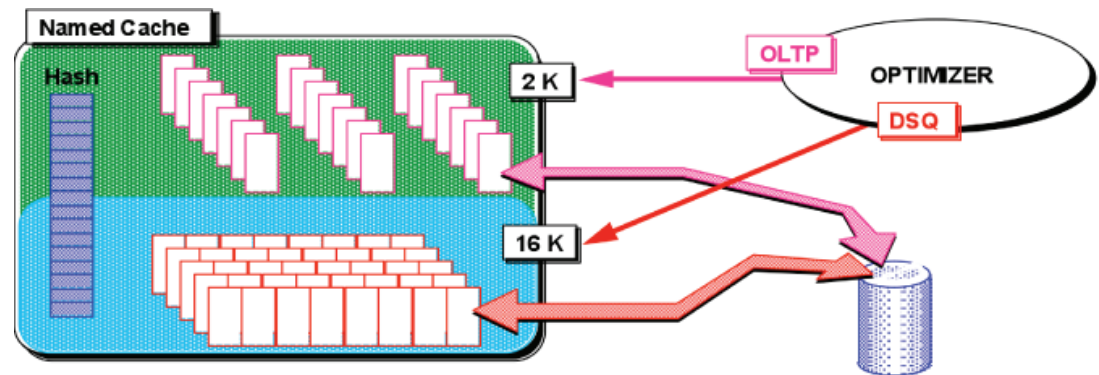


Figure 5. Variable-size Disk I/O Optimizes Storage Access for Concurrent OLTP & DSS


As this paper shows, ASE has optimized all the layers of a DBMS' kernel — CPU threading, memory management, disk and network i/o — to provide the most resource efficient Relational Database Management System in the Enterprise market.

CONCLUSIONS

Sybase's Adaptive Server Enterprise (ASE) is a mature, reliable, and high performance relational database management system for mission-critical, data intensive environments. For nearly three decades, it has been meeting the needs of data availability and transaction processing as data volume, number of users, and business complexity continue to grow. One of the primary reasons Sybase customers have remained loyal to ASE is because it provides one of the lowest total cost of ownership models of all the Enterprise RDBMS products.

Since the first line of code was written in 1984, resource efficiency and low TCO have been the core tenants which form the essence of our on-going engineering design philosophy. Over the years, we have sought to add greater capabilities within ASE to better support the needs of our customers and to leverage the shifts in hardware and software architecture. However, we have always tried our hardest to squeeze the most performance, the most availability, and the most scalability out of the hardware and operating systems as possible. Perhaps in the past, that has made Sybase less "interesting" to some of the server hardware companies in the industry. Irrespective of this, we strive to earn your respect and business as a database vendor who keeps efficiency at the top of our minds, always.

We hope this provides you with a deeper insight into Sybase's ASE database and what it will bring to you and your company as a credible and cost-effective platform for SAP applications and custom applications alike.



For information on our comprehensive
Consulting and Education Services to
support your Sybase technology initiatives,
visit us at www.sybase.com/consulting.

SYBASE, INC.
WORLDWIDE HEADQUARTERS
ONE SYBASE DRIVE
DUBLIN, CA 94568-7902
U.S.A.
1 800 8 SYBASE

www.sybase.com

Copyright © 2011 Sybase, Inc. All rights reserved. Unpublished rights reserved under U.S. copyright laws. Sybase, the Sybase logo, Adaptive Server Enterprise and Virtualized Resource Management are trademarks of Sybase, Inc. or its subsidiaries. ® indicates registration in the United States of America. SAP and the SAP logo are the trademarks or registered trademarks of SAP AG in Germany and in several other countries. All other trademarks are the property of their respective owners. 04/11

SYBASE®
An **SAP** Company