SAS®9 on the IBM® System z™ Mainframe:

The path to mainframe modernization that preserves your application investment

THE POWER TO KNOW®

Table of Contents

The SAS and IBM relationship

Since its founding in 1976, SAS has had a technological relationship with IBM. This relationship has expanded to include joint solution development and marketing. Partnering with IBM is a true win-win situation for both SAS and IBM. SAS benefits from IBM's experience in integrating technologies, and IBM aligns itself with a market leader in business intelligence. As a result, joint customers get the most complete set of intelligence and analytical solutions on the market.

SAS®9

SAS® 9 is the result of a multi-year project to revise and extend SAS architecture to meet the demands of enterprise-class applications and solutions. As a result of this project, SAS offers its customers the SAS Business Intelligence (BI) Architecture.

This paper focuses on the role of the mainframe in SAS BI Architecture and the implications and considerations for current mainframe customers. This paper is written for the people who run production data centers and who are experienced in operational procedures. Perspectives and opinions are based on the authors' experiences with the software. Performance figures relate to the SAS internal data center environment, and the caveat "Your mileage might vary" applies.

The major goals of SAS BI Architecture are:

- Usability — the ability to employ SAS software to accomplish business objectives.
- Scalability — the ability to expand the volume of data that is processed (scale up) and to expand the amount of processing resources for that amount of data (scale out).
- Manageability — the ability to configure, monitor, and control the various software components easily.
- Interoperability — the ability to have diverse components on diverse hardware work together to accomplish business objectives.

- Performance —the ability to accomplish a given task using the minimum amount of processing and hardware resources.
- Upward Compatibility — the ability to move existing, user-written SAS software programs to a new release of SAS without having to change the source code.

SAS software powers enterprise-class solutions to work with your enterprise-wide resources. In addition, another significant result has been achieved — leveraging a current customer's investment in SAS software. MultiVendor Architecture™ (MVA) applications that were developed in SAS® 8 and use DATA steps, SAS procedures, and SAS Component Language (SCL), in batch and interactive modes, can move easily to SAS®9. In our experience, no source code changes have been required during migration.

IBM® System z™

IBM System z servers are designed to provide a rich set of features, which include world-class workload management technology, dynamic logical partitioning, and an array of self-managing, self-optimizing, and self-healing tools in both the hardware and the z/OS software. Following are highlights and characteristics of the System z servers.

Scale Up, Scale Out

With System z granular growth, highly scalable processing power and innovative Capacity-on-Demand technology may be able to provide flexible responsiveness to changing IT needs.

Virtualization

With support for a maximum of 60 logical partitions that run varied applications and operating systems, the BC and EC servers can offer a strong solution for server consolidation.

Autonomic Computing

With self-healing and self-managing features such as dynamic memory sparing and dynamic disk balancing, System z servers are fine-tuning to provide high levels of performance that are required by on-demand businesses. Unique to System z, Intelligent Resource Director (IRD) technologies can provide dynamic allocation of resources across partitions and servers based on customer-designed policies.

z/Architecture

System z is based on z/Architecture, which can reduce bottlenecks that are associated with the lack of addressable memory and automatically direct resources to priority work through the IRD. A 64-bit superset of ESA/390, z/Architecture is designed to facilitate IT business transformation and handle the challenges of business-to-business and business-to-customer application growth.

Security

System z servers may effectively manage access to resources within your enterprise and help protect the integrity of your data and applications. System z security features can include isolated logical partitions with a high international certification level, EAL5. System z servers can provide industry-leading encryption with FIPS 140-2, Level 4 certification.

Integration

System z can improve information flow and access to assets throughout your enterprise, and it can co-exist easily with other hardware and software formats.

System z Application Assist Processor (zAAP)

zAAP is a specialty engine on System z that is designed to deliver a cost-effective, specialized Java execution environment with price/performance benefits when there is a need to integrate Java technology-based Web applications with mission-critical data on the same platform.

Threaded kernel (TK) architecture

TK Architecture is a significant enhancement in SAS software that enables effective exploitation of multiprocessors, including the mainframe.

The TK Architecture in SAS® 9 improves scalability by enabling SAS tasks to take advantage of multiprocessors. Each thread is an independently dispatchable unit of work. In versions 6 and 8 of SAS, the MVA SAS Supervisor is single-threaded at the task level (a task being a DATA step or a SAS procedure). In SAS® 9, SAS procedures can take advantage of threaded-kernel services, in addition to the single-threaded services of the MVA SAS Supervisor.

In Figure 1, a SAS procedure has initiated two threads, each of which can be dispatched on a separate processor.

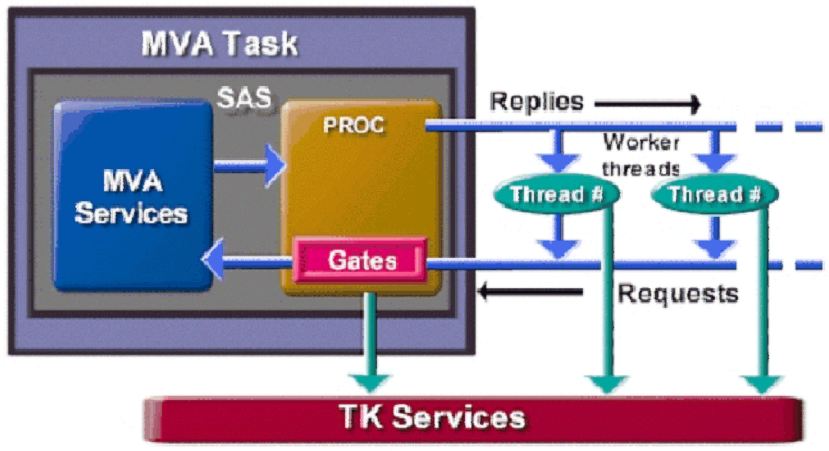


Figure 1 – Diagram of TK Architecture

How is SAS TK architecture implemented?

SAS TK Architecture is implemented for z/OS, via UNIX System Services (USS). Because USS is used, all SAS tasks (batch jobs, started tasks, and TSO commands) must run under RACF profiles that have an OpenMVS segment. The OpenMVS segment specifies a user ID (uid) and a group ID (gid). The RACF OpenMVS segment is necessary to support dubbing[1]. One consequence of SAS jobs running dubbed is that a CPU-time-limit-exceeded ABEND no longer shows as an S322, but instead shows as an SEC6.

Note: Threaded procedures require a special UTILLIB to handle multithreaded utility I/O. The UTILLIB should be implemented in a SAM-E data set or in an HFS or zFS file system.

What elements of SAS®9 implement TK?

The following procedures, which are listed under the SAS product that they're used in, have been rewritten to take advantage of TK Architecture.

In Base SAS®:

PROC SORT

PROC SUMMARY/MEANS

PROC TABULATE

PROC REPORT

PROC SQL

[1]"Dubbing" means to make an MVS address space known to UNIX System Services. Once dubbed, an address space is considered to be a process. Address spaces that are created by FORK() are automatically dubbed when they are created; other address spaces become dubbed when they invoke a z/OS UNIX service.

In SAS/STAT®:

PROC REG

PROC GLM

PROC LOESS

ROBUSTREG

In SAS® Enterprise Miner™:

PROC DMREG

PROC DMINE

The SAS system option THREADS/NOTHREADS controls whether these procedures will use multithreads. The option CPUCOUNT controls how many processors can be used simultaneously.

SAS®9 performs multithreaded (parallel) reads against Oracle, DB2, ODBC, and Sybase data stores. Multithreaded reads are used for fast data reads, using fast export in the SAS/ACCESS® Interface to Teradata.

The following SAS products exploit TK Architecture:

- SAS/ACCESS and PROC SQL support multithreaded queries.
- SAS/SHARE® TCP/IP access method optionally exploits multithreads in SAS®9 (this feature is experimental).
- SAS/CONNECT® uses TK services to implement Compute Services in SAS®9.
- MP CONNECT provides parallel processing of multiple submitted DATA step and procedure streams. Pipeline parallelism permits SAS DATA steps and procedures to pass data via TCP/IP socket pipes.
- The SAS® Scalable Performance Data Engine (SAS SPD Engine) is an important data engine in Base SAS.

Scalability

The SAS SPD Engine

The SAS SPD Engine is part of Base SAS®9. The SPD Engine takes advantage of parallelism to improve throughput of data loading, index creation, and data retrieval as shown in Figure 2.

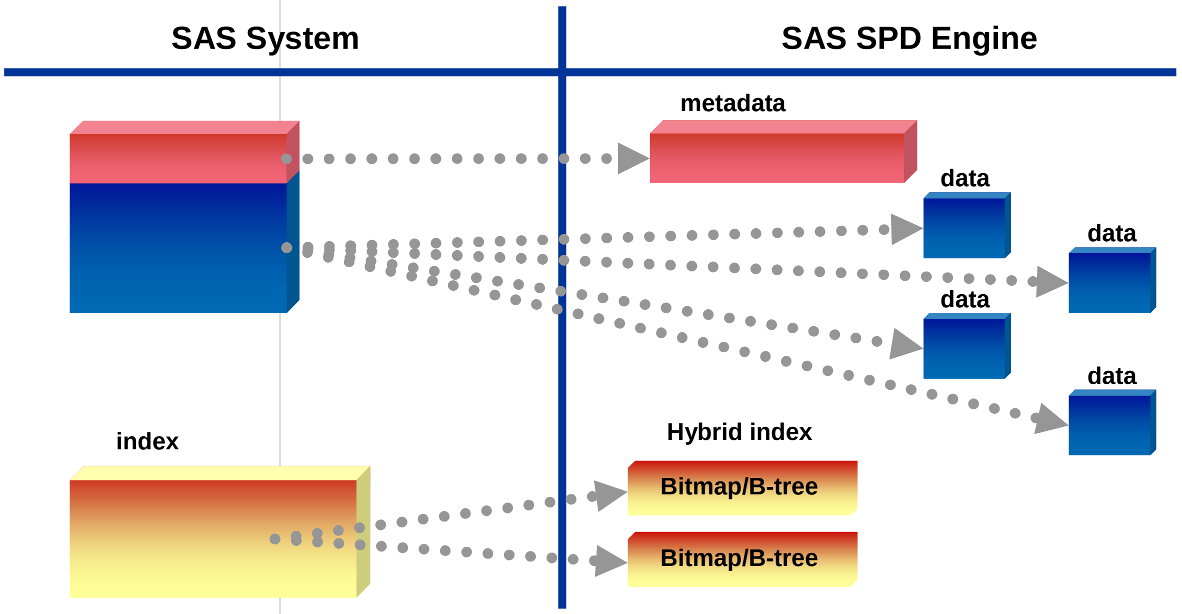


Figure 2 – SAS Input/Output versus the SAS SPD Engine

Standard SAS data set physical files consist of one file that contains data and metadata (for example, PROC CONTENTS information) and one file for each index.

The SAS SPD Engine data set separates metadata from data, stores data in multiple physical partitions, and has multiple index data sets. Index space utilization, storage, and retrieval are more efficient with the SAS SPD Engine.

The SAS SPD Engine uses multiple indices in the resolution of a WHERE clause; the standard data set engine uses only one index. In addition, where possible, the query is broken into parts that can be executed in parallel. All of these techniques greatly reduce reporting and query response times. Tests that were run on an IBM 2105-F20 ESS (Enterprise Storage Server) showed performance gains due to these features.

The SAS SPD Engine implemented on z/OS uses USS thread services and a USS directory-based file system: zFS, HFS, or NFS. To use the SAS SPD Engine, define a file system and change the engine name that is specified in the LIBNAME statement from LIBNAME to SPD.

The USS file systems HFS and zFS have been supported for SAS data libraries on z/OS for some time, however, this is the first time that these file systems are required for any purpose.

Initial exploitation of the SAS SPD Engine is easy. You only have to specify the engine name SPD in the LIBNAME statement. There are many options that can be used to configure and tune a SAS SPD Engine library, but starting with the default options is a reasonable first step.

Figure 3 shows an overview of the aspects of parallelism that are made possible by Threaded Kernel services in the SAS execution environment.

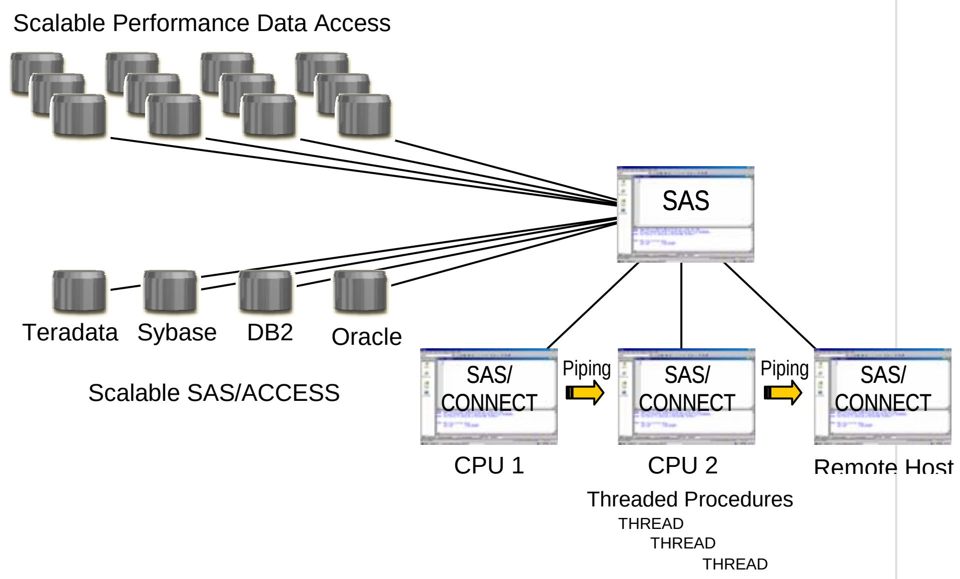


Figure 3 – SAS Scalable Architecture

For example, the white box labeled “SAS” could represent a z/OS SAS batch job step. In that box, parallelism can be achieved by using a threaded procedure such as PROC SUMMARY. As another example, SAS/CONNECT could launch multiple local or remote MP CONNECT processes (the SAS/CONNECT boxes CPU 1 and CPU 2 are local; the third SAS/CONNECT box is remote). These processes could, in turn, run threaded procedures, as shown under the SAS/CONNECT box in the middle.

Data could be piped among these MP CONNECT processes as depicted by the “Piping” arrows.

The gray “disk farms” on the left of the figure represent the added parallelism that is available through the SAS SPD and SAS/ACCESS engines. The spawned MP CONNECT processes could access these disk farms also, but connection lines were omitted for simplicity.

Upward compatibility

The ability to move existing, user-written SAS software programs to a new release of SAS, without having to change the source code, preserves a current customer's investment in SAS software. For example, SAS® IT Resource Management, 2.6, which is shipped with SAS 8.2, works unchanged with SAS®9.

SAS IT Resource Management makes heavy use of:

- The DATA step
- Graphics and Base SAS procedures
- SQL
- Macro language
- SCL (interactive and in batch)
- SAS/CONNECT

Experience shows that moving existing SAS applications from SAS 8 to SAS®9 should be smooth and should require few, if any, changes in source code. As a test, SAS IT Resource Management was moved to SAS®9 and no source code changes were required. This is impressive when you consider that SAS IT Resource Management uses many SAS features and many SAS components to create a complex layered application. Version 8 SAS/AF® application catalogs run on SAS®9 without changes, and there's no need to recompile SCL programs.

Performance

DATA step and procedure performance

In SAS®9, for a quick indicator of general SAS performance, MXG software was used as a representative and widely used SAS batch application. The "compile-only" test result indicates that customers who are running code-intensive but relatively light data might see a big reduction in CPU time. Analyzing the MXG results a little differently, the compilation/setup phase of the job showed a 30%

reduction in CPU time, and the execution phase showed a 12% reduction in CPU time. The MXG results also show the following:

compile-only BUILDPDB run — 30% reduction in CPU time

one day of SMF data — 20% reduction in CPU time

three days of SMF data — 15% reduction in CPU time

For a rewrite of vector math routines to exploit System z pipeline architecture, there have been reductions in CPU time of up to 25% to 35% for PROC GLM, PROC LOGISTIC, and PROC MIXED.

“For more detailed performance information see the white paper "Performance Test Suite Results for SAS® 9.1 Foundation on the IBM® zSeries™ Mainframe" at support.sas.com/rnd/papers/sugi30/Performance_Test_Suite_031805.pdf.

Added features

Base SAS software

- The **XML engine** supports importing data from an XML document into a SAS data set and exporting a SAS data set into an XML document. The XML engine allows data from an XML document to be imported directly into a DATA step or a SAS procedure without first being loaded into a SAS data set. Also, observations generated by a DATA step or a SAS procedure can be exported directly to an XML document without first being instantiated to a SAS data set.

- The **metadata engine** provides a single point of access to many heterogeneous data sources without the user having to be aware of engine-specific details. The metadata engine uses information retrieved from the SAS Metadata Server to construct a LIBNAME statement for the engine that is specified in the metadata (referred to as the underlying engine) and assigns appropriate options for the engine.

- **String handling functions** search and compare character strings and concatenate character strings. For more details about string handling functions, see *Appendix A – added functions*.

- The Perl regular expression **PRX functions** and **CALL routines** use a modified version of Perl as a pattern-matching language to enhance search-and-replace operations on text.

- **Format and informat** names can be 31 characters.
- The **DS2CSV** macro outputs a SAS data set into a Comma Separated Values (CSV) file. This feature was originally part of SAS/IntrNet® software.

In addition, there are many additional functions, formats, informats, and procedure enhancements.

Hash tables

SAS provides two predefined component objects for use in a DATA step: the hash object and the hash iterator object. These objects enable the user to quickly and efficiently store, search, and retrieve data that is based on lookup keys. The hash object keys and data are DATA step variables. Key and data values can be directly assigned constant values or SAS data set values.

A popular use of SAS formats has been as a table lookup mechanism. SAS formats were not designed for that purpose, but they work well in that capacity. However, there has been a need for a more general table lookup mechanism, specifically, a mechanism that allows entries to be dynamically added, updated, and replaced and that supports multiple variables in a single table entry. The DATA step hash table provides these capabilities.

A more complex implementation involves using table entries as holding cells for observations that are being constructed concurrently during the execution of a DATA step. When all the elements of a given observation have appeared, and the object is fully populated, then that observation can be written to an output data set.

This implementation example is based on a common problem when working with system performance data logs that contain multiple event records for multiple transactions. The output that you want is a single record per transaction into which all event information has been gathered. This approach is a scalable way to solve the problem, and it eliminates sorts and multiple passes of the data. The size of the hash table is based on the number of transactions that are in progress concurrently, not on the total number of transactions.

As related to parallelism, this DATA step can be implemented as a DATA step view, or it can be piped via MP CONNECT to a PROC SUMMARY, which would produce one or more summary tables based on transaction code, application, accounting codes, or whatever else you might want. For more details about DATA step hash tables, see *Appendix B – hash tables*.

Application response measurement (ARM)

Support for the ARM API was introduced with version 8 of SAS (Figure 4). The ARM API enables applications to report transaction start, stop, and update information via an open platform-neutral API. SAS 8 provided SAS language-specific macros that enabled SAS application developers to instrument code for performance measurement, operator alerting, or transaction accounting. Event information recorded through the API was handled by the SAS ARM subsystem and written to the SAS log or to a system management product such as those provided by Tivoli and HP.

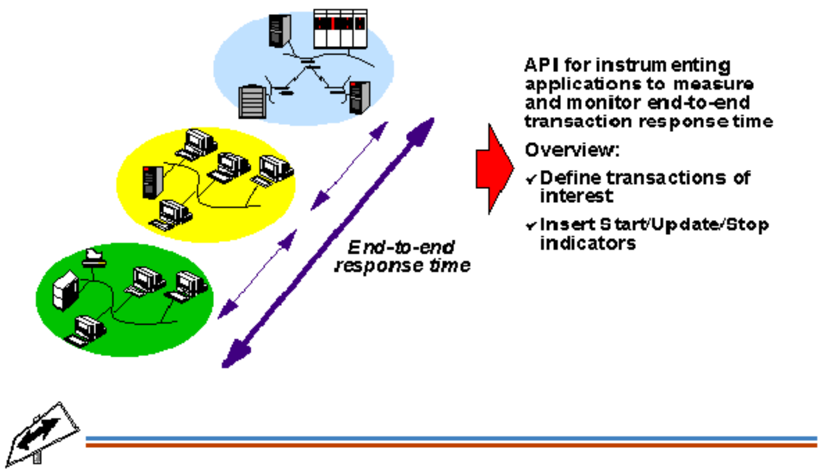

API for instrumenting applications to measure and monitor end-to-end transaction response time
Overview:
✓Define transactions of interest
✓Insert Start/Update/Stop indicators
End-to-end response time

Figure 4 – Application Response Measurement API Overview

ARM support is used for business transaction workload management. SAS®9 provides more than ARM support for customer applications. SAS®9 exploits ARM instrumentation within SAS components. The industry-standard ARM API enables programs that produce ARM records to log records by using any vendor's ARM agent that supports the standard API. SAS programs that produce ARM log records can register these records with the HP ARM agent on HP machines or with the IBM ARM agent on IBM machines. Distributed workload managers such as Enterprise Workload Manager (eWLM) from IBM use real-time ARM measurements to regulate and balance the workload on their respective machines.

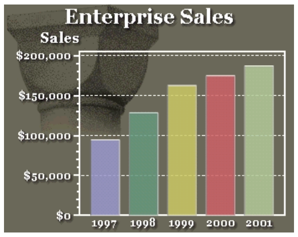
ARM support changes in SAS®9 include:

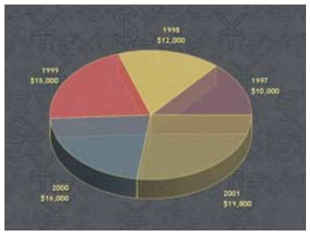
- SAS macros that support ARM 2
- Generalized ARM subsystem support that can be used by all SAS components
- ARM recording for SAS components that are controlled by SAS option settings.

For more details about ARM support, see *Appendix C – ARM support*.

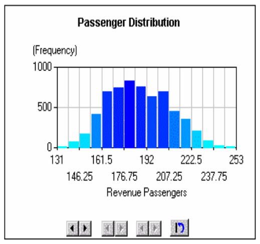
Improved graphics support under z/OS

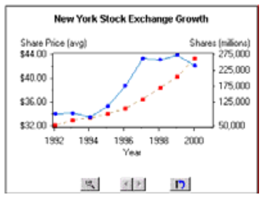
SAS®9 provides improved ActiveX and Java drivers. These drivers enable users to produce interactive graphics output for the Web (Figure 5). The graphics support zooming, rotation, rubber banding, drilling, and dynamic switching of presentation style.





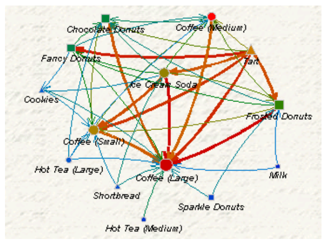
16 New Graphs Styles

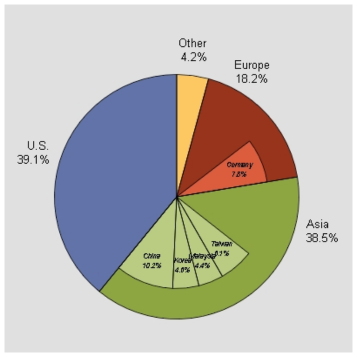




Graph-n-Go
Overlay Plot

Graph-n-Go
Histogram





Constellation Chart
Applet

Detail Pie

Figure 5 – New Graph Styles in SAS®9

Usability

Usability of the interactive interface on the mainframe has always been constrained by the capabilities of the 3270 terminal interface. While the 3270 terminal interface is fully supported by Display Manager and SAS/AF applications that run under z/OS, Windows, UNIX, and Web browser client interfaces can be used for a richer interaction with SAS applications that run on the mainframe (Figure 6).

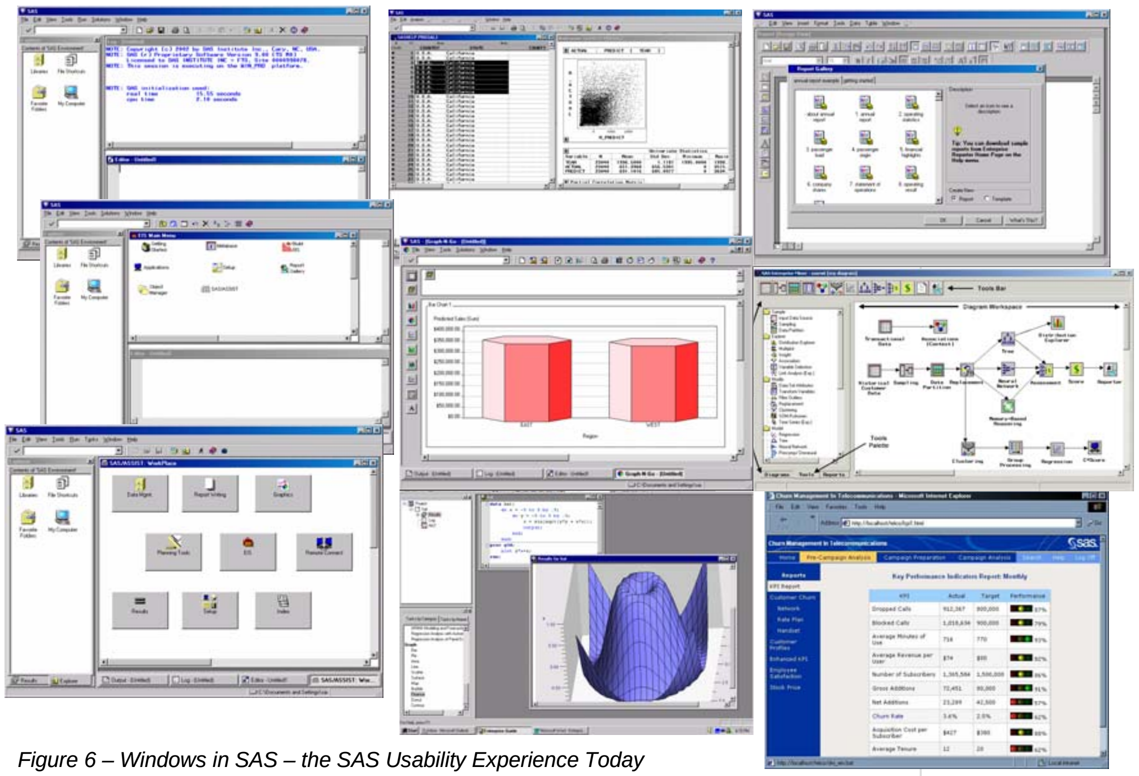


Figure 6 – Windows in SAS – the SAS Usability Experience Today

Accessibility and usability of batch-generated reports are greatly improved through enhanced and extended SAS®9 Web publishing features that are available in SAS® Foundation statements and procedures.

The Output Delivery System (ODS) has been enhanced in SAS®9 to meet Web publishing requirements. Support has been added for various output destinations, including PostScript, PDF, and Wireless Markup Language (WML).

The DOCUMENT procedure enables the user to re-arrange, duplicate, or remove output from the results of a procedure or a database query. Also, the user can generate output for one or more ODS destinations by using the newly transformed output hierarchy file.

The MARKUP statement produces SAS output that is formatted by using one of many different markup languages.

Using these SAS®9 features, customers can upgrade their batch report delivery to high-quality Web output that is prepared and delivered directly from z/OS.

SAS Foundation servers

All SAS Foundation servers are supported under z/OS (Figure 7).

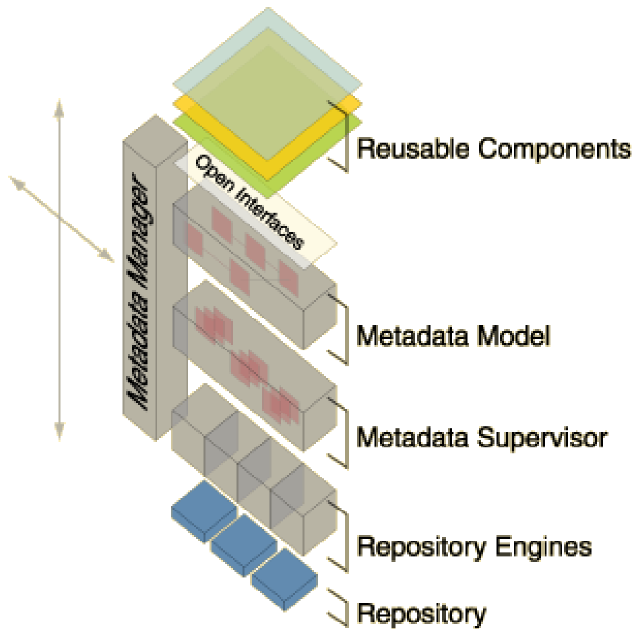


Figure 7 – SAS®9 Open Metadata Architecture Foundation Servers

SAS® Metadata Server

The SAS Metadata Server provides metadata repository services (via XML) to other SAS components, including other servers.

The layered architecture of the SAS Metadata Server consists of repositories, repository engines, the Metadata Supervisor, and the Metadata Model. Reusable components are the extensive library of metadata templates.

SAS® Workspace Server

The SAS Workspace Server spawns address spaces on behalf of clients (or other servers such as the SAS OLAP Server) in which SAS Foundation work is run.

SAS® Stored Processes Server

Stored processes are SAS programs that exist on a server. Most stored processes have macro variables in them. These macro variables can be set by clients to customize the execution of the job. Multiple clients can create and execute stored processes by using the appropriate interfaces.

SAS® OLAP Server

In SAS®9, the SAS OLAP Server (Figure 8) enables users to develop and deploy scalable OLAP applications. Automated data loading and cube building are available through the new administration interface called the SAS OLAP Server Administrator, which is based on Java technology. OLAP queries are performed using the MultiDimensional eXpressions (MDX) query language in client applications that are connected to the SAS OLAP Server. MDX is a standard OLAP cube query language; therefore, SAS OLAP Server can be accessed from other vendors' clients, as well as from SAS clients. Each client session uses separate threads. Where possible, queries are broken into concurrently executable parts. In-memory data caching is extensively used for frequently accessed cube rows or slices. Data storage is implemented with the SPD Engine, which is a highly parallel data access engine.

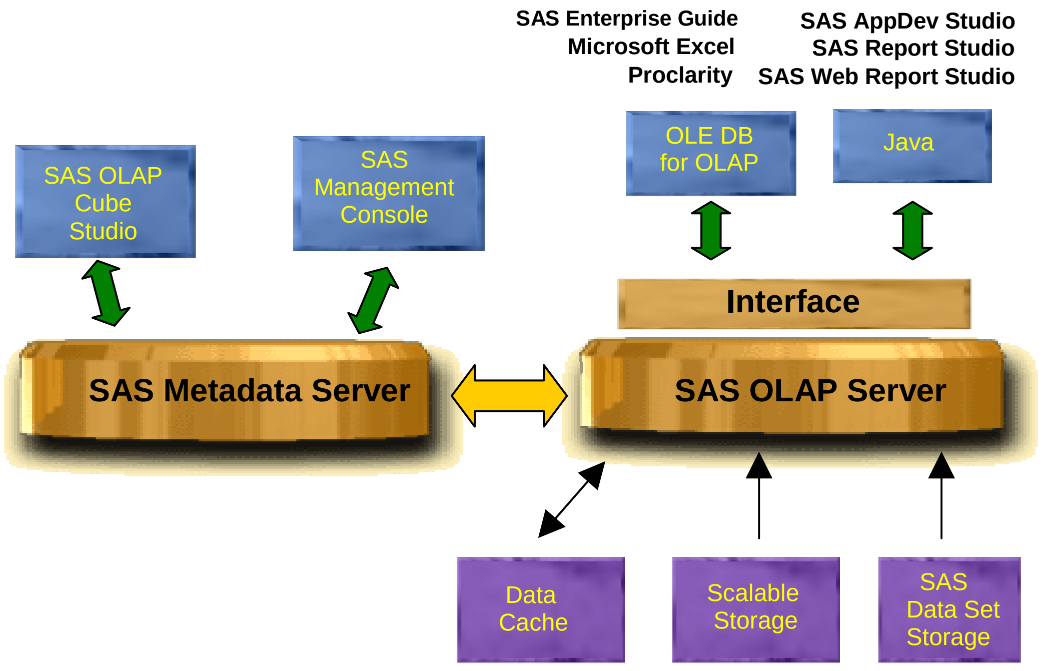


Figure 8 – SAS OLAP Server Architecture

SAS OLAP Server supercedes the SAS/MDDB Server, which was shipped in version 8 of the SAS System. The SAS OLAP Server's principal design goals are scalability and performance.

SAS®9 BI architecture

The SAS®9 BI Architecture is shown in Figure 9.

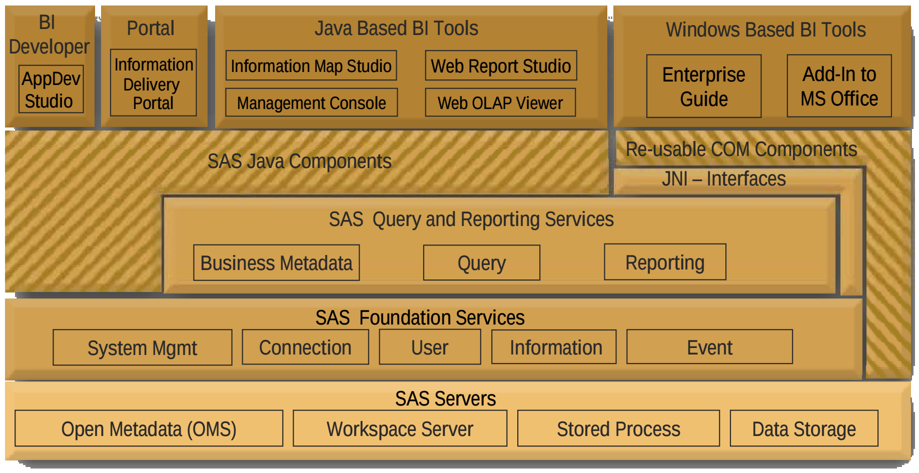


Figure 9 - Complete SAS®9 BI Architecture

SAS Client Services includes:

- A development environment — SAS® AppDev Studio™
- A portal — SAS® Information Delivery Portal
- BI tools based on Java technology — SAS® Information Map Studio, SAS Web® Report Studio, and SAS® Management Console
- BI tools based on Windows — the SAS® Add-In for Microsoft Office, SAS Web® Report Studio, and SAS® Enterprise Guide®

SAS BI Infrastructure includes:

- SAS Application Services — SAS Java Components, SAS COM Components, SAS JNI Interfaces, SAS Query and Reporting Services
- SAS Foundation Services

SAS Foundation Servers include:

- SAS Metadata Server
- SAS Workspace Server
- SAS Stored Processes Server
- Data Storage

The SAS Foundation products include:

- Base SAS
- SAS/CONNECT
- SAS/ACCESS
- SAS/STAT
- SAS/GRAPH®
- SAS/SHARE
- SAS/ETS®
- SAS/QC®

Summary

Experience strongly suggests that customers who are running SAS software under z/OS can confidently regression test their current SAS applications under SAS®9 without having to first make source code changes. Then, customers can deploy their applications in a production area with expectations of improved performance over SAS 8 software.

At that point, customers are in an excellent position to begin evaluating, testing, and deploying new SAS®9 features and components that are appropriate for their business needs and priorities.

SAS®9 and the System z mainframes create a powerful platform that improves the performance of existing applications enabling next generation BI solutions that will give enterprises better answers, faster.

Appendix A – added functions

String search functions

Functions with the ANY* prefix enable the user to search for any character in the specified class and return the position of the first occurrence in the string. For example, ANYALPHA searches for the first alphabetic character and returns its position in the string. ANYDIGIT searches for the first numeric digit and returns its position in the string.

String concatenation functions

Functions with the CAT* prefix concatenate two strings. For example, CAT<*function*> concatenates strings without removing any leading or trailing blanks. CATS<*function*> concatenates strings and removes leading and trailing blanks.

String validation functions

Functions with the NOT* prefix search a string for the first character that is NOT in the class of characters that are specified. For example, NOTALPHA searches for the first non-alphabetic character and returns its position in the string. NOTDIGIT searches for the first non-numeric digit and returns its position in the string.

Formats and informats

User-specified numeric format names can be 32 characters long, and user-specified character format names can be 31 characters long. Date formats and informats have been added that do not require a day value to be output.

Macro functions

Functions with the SYM* prefix determine the existence and scope of macro variables. Similar to SYMPUT and SYMGET, which are used to establish macro variable values at execution time, SYMEXIST indicates the existence of a macro variable, and SYMGLOBAL and SYMLOCAL indicate the scope of the macro variable at execution time.

Appendix B – hash tables

The SAS DATA step component interface enables the user to create and manipulate hash table component objects using statements, attributes, and methods.

For example:

```
declare Hash ht();          /* table definition */
ht.defineKey("ISBN");       /* key for Table    */
                            /* specify variables to be
stored */
ht.defineData("author", "title“    “price”);
ht.defineDone();            /* table definition complete
*/
                            /* define data for a single
line */
ISBN="0679722769";
author="Joyce"; title="Ulysses";
price=19.99;
rc=ht.add();
```

Appendix C – ARM support

There are two categories of ARM macros. There are ARM macros that are used to instrument applications, for example:

%ARMINIT

%ARMGTID

%ARMSTRT

%ARMUPDT

%ARMSTOP

%ARMEND

There are also ARM post-processing macros to use with the ARM log, for example:

%ARMPROC

%ARMJOIN

SAS ARM system options:

Syntax: OPTIONS ARMSUBSYS=(ARM_NONE | ARM_ALL<, ARM_DSIO, ARM_IOM, ARM_PROC, ARM_OLAP_SESSION>);

ARM_NONE – specifies that no internal SAS processing transactions are written to the ARM log. This is the default setting.

ARM_ALL – specifies that all available internal SAS processing transactions are written to the ARM log.

ARM_DSIO – collects SAS data set I/O processing information.

ARM_IOM – collects IOM Server processing information.

ARM_PROC – collects SAS procedure and DATA step processing information.

ARM_OLAP_SESSION – collects SAS OLAP Server session information.

THE POWER TO KNOW®

SAS INSTITUTE INC. WORLD HEADQUARTERS 919 677 8000

U.S. & CANADA SALES 800 727 0025 SAS INTERNATIONAL +49 6221 416-0 **WWW.SAS.COM**

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc.in the USA and other countries.
® indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2006, SAS Institute Inc.
All rights reserved. 437784.0407