Paper 393-2008

Chickens, Eggs, and SAS ® BI: An Evolutionary Approach

Donald Atkinson, Patricia Konstantinopoulos, Ian Huggins

ABSTRACT

Business Intelligence wisdom stresses the need for careful planning before deploying "*Your BI Solution,*" but successful BI deployments present a chicken/egg dilemma: it's difficult to know what you need to know until you've gained usage and deployment experience. It's hard to know if you'll get it right the first time because BI is multifaceted and each deployment is sensitive to local conditions and contexts. Our approach was to build a small prototype and use that to discover what we did not know. We also needed to better understand likely avenues of BI usage that would let us play to our strengths. We faced severe constraints in terms of budget, staff and specific knowledge. With this in mind we chose Solaris 10 Containers for our BI install as the best way to gain knowledge and prove concept. Chief among the things to be demonstrated was that achieving some of the returns promised by BI need not be as daunting, and strictly major-league, as they may initially seem. Successful deployments don't have to be limited to large, well funded, well staffed, expert efforts. Even your most novice attempts can produce useful artifacts that can be leveraged forward. This paper discusses our approach, specific kinds of challenges, typical problems and the successes that helped us to decide to push out a production iteration and to expand our SAS BI initiative without the need for the usual sales pitch.

WHAT LED US TO START DOWN THIS PATH (You Have to Start Somewhere)

When we first got the idea of "doing BI" we found the whole thing both intriguing and daunting. We tried asking Google "what is Business Intelligence" and got 30,000 hits. We wanted to deploy more web stuff but we had "staffing issues." We also feared we were falling behind the times. Numerous presentations and articles stressed the need for rigorous upfront planning and specification. We looked for a roadmap to take us "from here, to there." The problem was that not much seemed to fit our circumstance (think tiny and poor). As we considered our own investment in SAS code, our backgrounds, and the potential for "good things to happen" from BI, our notions of "here" and "there" began to change.

In 2006 we attended SUGI 31 and set about evaluating SAS® BI in terms of our known organizational needs and what parts of the architecture we thought we could handle. We wanted to move quickly and with a low profile. We also were not sure how much of a support load our server group would undertake without balking. In resource constrained environments, the balking zone can be a major consideration.

This conference was extremely valuable in that we came away with some answers for our most pressing concern, which was that we didn't have an implementable development path. We talked to a lot of SAS staff at the campground and bounced around ideas that seemed to always start with "Do you think you could …." and ended with "…..theoretically that should work, I don't see why not." It was like wandering through the parts store picking what we thought we could use. Our curiosity was piqued; we were becoming strangely comfortable with a fuzzy definition of BI.

Even as interest in the potential for BI was running high, staffing and dollars were running low. We had existing SAS programming skill sets and a code base used to generate various reports and also to massage data going into and out of our Oracle databases. The prospect of moving SAS paper reports to a quicker, cooler, dynamically generated web distribution was very appealing. As Homer Simpson might say "Multiple Oracle databases, SAS BI, no paper, easy connections – Hmmmmm."

THE INITIAL CHALLENGE OR SMALL SHOP BLUES (Connecting the Local Dots)

The fear that the whole thing is overly ambitious must be overcome. You have to decide who to get on board first, your management or your staff. The important thing is that someone gets enthused. We chose staff. Fortunately dynamic report generation appealed to our programmers' love of not keeping track of static web content. We identified some real business needs. Meeting these forced us to ask ourselves if this was really something we could do. You assess your strengths and weaknesses and identify achievable connections between the skills your staff currently possess, and the skills which you will need them to acquire. It's the basic SWOT – Strengths, Weaknesses, Opportunities, and Threats review.

From a small-group perspective, it's important that the pile of tasks not seem like such an unscalable mountain that nobody wants to start. Consider how to get people excited about taking on more work using toolsets that are loosely related to old familiar tools. One way of reducing dissonance is to approach BI as the natural extension of the existing framework. Let your people inform you about what seems "most natural" to them. This channels initial effort through known pathways and opens a dialog. Communication is essential.

To introduce new activities into the work environment is to encounter "culture trumps." Pretty much culture can trump whatever.[1] In our case, cultural issues included the current technological focus and the existing toolset of our IT staff. There was already grousing about the demands the increasing frequency of vendor-mandated upgrades placed on staff. In a similar way, security concerns followed suit. There was no real consensus about BI except as a new buzzword which had already acquired several levels of meaning including "odbc" and "spread marts." If your local culture elevates frugality to an art form then money for new hardware may also be an obstacle to be overcome. You have to watch out for that moment when not enough bodies, knowledge, hardware, or time to achieve anything approaching "necessary and sufficient" for as complex an undertaking as BI, creeps up on you. Resist it.

BUILDING AN INITIAL ROADMAP (Getting and Gaining Familiarity, Confidence, and a User)

In an interview with Christine Perfetti, Scott Berkun said "… there are cultures and management styles that allow for innovation at any scale - and that's the real thing to focus on - culture and management, not size."[2] Many in our group cut their teeth on IBM mainframes. We were not trying to be especially inventive – in truth we wanted a cook book. We decided to ignore our lack of size and concentrate on a mutual enculturation with BI. Eventually we realized we would have to make up some of this for ourselves. I think the key was to frame the new thing (SAS BI) as a really good way to make the existing technology stack more powerful. Like Tim the Tool Man's[3] favorite phrase, it's hard to argue with "More Power!" Inspiration can come from a lot of places – SUGI sessions, Jack Sparrow[4] movies, and old MacGyver[5] reruns.

We did have a Swiss Army knife and some duct tape but, in this case we also "found" a single low end Sun Fire V240 server running Solaris 9. One of our takeaways from SUGI was the"theoretical knowledge" that we should be able to use Solaris 10 to set up local zones in a Solaris 10 Container environment. This would allow us to emulate a four server setup on a single box.

The process of prototyping a SAS BI deployment as proof-of-concept was the beginning. The idea was to allow staff to become familiar with the architecture and terminology without making a commitment of equipment and staff resources that did not exist. The old saying is "Time, Quality and Price – Pick two." The "two" we chose seemed to vary according to the needs of the moment. In a world where everything is accounted for and attributable, this deployment fits conveniently under the heading of professional development and competitive position.

BABY STEPS – NUMEROUS BABY STEPS (Use SUGI to Learn What You Need To Know)

An invaluable take away from SUGI and SGF was a better understanding of what we didn't know. For example, the server staff needed to learn about putting up Solaris 10 Containers. The development staff needed a single target to focus their attention. We needed to connect our pushy users' wish list items to our nascent BI skills which included takeaway knowledge that Java could be used to execute SAS® Stored Processes in the BI environment.

The "ah-ha moment" was when we decided to approach our first SAS BI project in a way that mimicked something we were already doing – just injecting SAS BI technology. It was a short jump to convince our wary developers to try their Java skills to get a quick success that leveraged SAS code, repackaged as a stored process, and a familiar Java framework. The only new thing we needed to do was set up the SAS BI environment. We expected to learn about the "reach-exceeding-grasp" thing that Robert Browning liked to push.[6]

QUICKEST WIN FOR US (Opportunity Risk as a Positive Thing)

In order to create an environment for SAS Stored Processes and make Metadata connections to Oracle we needed to get a SAS BI license. Once we had this we could install SAS BI, learn to administer the environment, and write some stored processes for Java to execute on a workspace server. That was a great meeting.

We worked with SAS to obtain a BI license. This was a bit of a back-and-forth as the deployment we were proposing was somewhat theoretical and might best be described as Micro-BI. The "Traditional," or "Standard," license met our need and helped keep costs within our ad-hoc budget. We were comforted to know there was an upgrade path in case this whole thing worked. We chose Apache HTTP Server –Tomcat as our web server solution since we had prior experience with it and it was within our budget. The SAS® Education course "Creating and Distributing SAS Stored Processes" was extremely relevant to this case.

GETTING GOING (First Make a Roux – It's Easy)

Relying on some of that "I don't see why not" information from SUGI, we reasoned that once we implemented a multiserver environment on our single Sun Microsystems SunFire V240 using the Solaris 10 Operating Environment (OE) virtualization technology (Solaris Containers), we could scale out the individual containers to separate machines as needed.

The SAS BI environment was established upon a global zone named sassafras with four local zones: metasas, websas, svc1sas and svc2sas. Based upon our model of the anticipated network I/O characteristics of the components of the SA BI environment, we allocated one physical network interface to the global zone, another to metasas, and one to websas. The compute servers had separate logical network interfaces but shared a physical interface.

The RAID 1 disk partitioning design provided tolerance for single disk failure and isolated the partitions used by the global zone from the local zones, and the local zones from each other. In terms of the operating system, the file system layout was designed to reduce the impact of file space over-allocation and simplify system recovery. We planned to correct disk I/O performance bottlenecks identified once the environment was operating by moving partitions to external SCSI bus attached disk storage.

File System mount points

Device Mirror	File System
d80	/
d81	<swap>
d83	/usr
d84	/var
d85	/home
d90	/home/metasas
d91	/home/websas
d93	/home/svc1sas
d94	/home/svc2sas

Solstice Disk Suite Metadevices

Submirror Set A	Attached Disk Slice	Submirror Set B	Attached Disk slice	Mirror Device
d40	c1t1d0s0	d0	c1t0d0s0	d80
d41	c1t1d0s1	d1	c1t0d0s1	d81
d43	c1t1d0s3	d3	c1t0d0s3	d83
d44	c1t1d0s4	d4	c1t0d0s4	d84
d45	c1t1d0s5	d5	c1t0d0s5	d85
d50	c1t3d0s0	d10	c1t2d0s0	d90
d51	c1t3d0s1	d11	c1t2d0s1	d91
d53	c1t3d0s3	d13	c1t2d0s3	d93
d54	c1t3d0s4	d14	c1t2d0s4	d94

(1) The Solaris disk partitioning (slicing) software reserves slice 2 to represent the entire hard drive, and only allows 8 slices per physical drive.
(2) Slice 6 was unutilized, and slice 7 on each drive was reserved for the Solstice Disk Suite metadatabases.

SUPPORT SOFTWARE FOR SAS BI (Proving the Value of Good Coffee)

An awareness of the SAS install instructions even as you prepare the Solaris environment first is helpful. For example, "Third-Party Software for SAS® 9.1.3 Foundation" [7] (at the time Service Pack 3) indicated we needed to install the 2.0.45, or higher, release of the Apache HTTP Server and the 4.1.18 release of the Tomcat Java servlet container applications prior to installing the SAS BI product. We selected the then current 2.0.59 release of the 2.0 HTTP Server development tree. We added support for WebDAV HTTP Extensions for Distributed Authoring to the HTTP Server build, and utilized Litmus,[8] a WebDAV server test suite which facilitates determining if a server is compliant with the WebDAV protocol in RFC 2518. [9] Since the Tomcat application server required a Java runtime environment we installed j2sdk1.4.2_05.

In the Solaris OE, the Apache HTTP server has package dependencies upon the db-4.2.52.NC, expat, gdbm, libiconv, openssl-0.9.8f, zlib, and several of those packages are dependent in turn upon the libgcc-3.3 or gcc-3.3.2 packages being installed. Fortunately Steve Christensen has built many of these packages for various Solaris OE releases and made them available at the Sun Freeware website.[10]

SSL CERTIFICATE FOR WEBSAS TOMCAT AND HTTP SERVER

We added SSL support to the HTTP Server build so that web browser communication with the SAS BI environment would be able to utilize Digest authentication, Basic authentication, and TLS/SSL encryption. OpenSSL allows you to create and manage key and certificate security principals as individual files. Tomcat however, uses a Java subsystem, JSSE, to manage security principals and requires you to use the keytool utility to store the security principals in a keystore file. Sharing a private key with both the HTTP Server and Tomcat is not directly possible. The solution we found to this problem was to obtain the ExportPriv.java program to decrypt the keystore security principals.[11] We used Verisign as our certificate authority. Since Verisign introduced an intermediate trusted certificate authority layer, we had to import its trustcacerts file into the keystore. Until this was done, SSL negotiation could not complete and client browsers would generate “no compatible protocols” messages.

The Keytool utility is required only when utilizing a standalone Tomcat Application Server. Tomcat, out of the box, comes with its own web server. It works well, but is not as robust as the Apache Web Server. Tomcat can be modified to utilize Apache as its front-end web server with Apache passing off java requests to Tomcat via its MOD_PROXY and/or MOD_JK plugins. When Tomcat is utilized in this configuration, SSL needs be configured only in Apache using OpenSSL, with the Tomcat listener configured to accept only local requests for service.

An interesting gotcha with the SSL protocol under Solaris 10 was that web clients were not initially able to establish SSL sessions. We did a protocol analysis of SSL negotiation attempts, and directed errors to a log file at an elevated logging level. This revealed a cryptic message, "TLS1_SETUP_KEY_BLOCK:cipher or hash unavailable". In a different discussion thread for postfix mail, we found out that this error was related to a restriction in the 128 bit encryption provided by the Solaris 10 operating system versus 256 bit encryption available through OpenSSL. Postfix.org told us that the issue is that the server offers the client a list of high encryption ciphers that are disabled in the SunOS 5.10 libcrypto.[12]

We temporarily changed the httpd-ssl.conf file to remove the capability of the HTTP Server to negotiate high encryption through OpenSSL, dropping it to medium. Web clients were then able to engage in SSL sessions. Then we updates the Solaris cryptography library with the "Solaris 10 Data Encryption Supplement" which provides cryptographic algorithms for meeting strong encryption requirements. We re-enabled high encryption successfully.

Once the third party support software installation was complete we created the SAS BI software depot in the file system /home/sasdepot which was visible to each non-global zone as well as the global zone (sassafras) where it physically resided.

INSTALL DAY(Z) (They Also Serve Who Only Stand and Wait)

While we prepared the Solaris environment we also obtained our SAS BI and SAS® Analytics Suite licenses. At this point we were ready to go ahead and do an install. Just kidding – SAS made sure we understood that we could watch while they worked. Even though we were not doing a SAS® Enterprise Edition install, and had an experienced in-house staff, we spent three days watching our SAS installation consultant work. This was worth the cost and the effort. The best reason to “assist” while SAS takes the lead on new installs is that the installs are knowledge intensive events. The expense of gaining this specialized install knowledge versus the number of times you will use it speaks in favor of letting SAS lead your install effort.

A gotcha issue we ran into during the install related to public-private address translation with Network Address Translation (NAT) in our network design. This is covered in Usage Note 18503.[13] This issue results in an inability to connect to a Stored Process Server on UNIX from client applications that are on Windows when NAT is used. There are also separate usage notes for the management console and stored process server. What happens is that you can’t run a stored process, or even make a connection, when you try to access a Stored Process Server. You get a message along the lines of “… there is no server listening for connections or the server is too busy to access any new connections." Windows clients that experience this problem include Enterprise Guide and Add-In for Microsoft Office.

What makes this problem interesting is that when you do the same Stored Process Server test from the SAS Management Console on the UNIX host or a Web client, then it succeeds. The cause is that the IP address, as it appears to the Windows client, is not the actual IP address on the server due to NAT translation of the IP address. The SAS support solution is to specify the -lbUseHostName option when starting the Object Spawner on UNIX (SAS Notes SN -V9-015049).[14] By using this option at spawner invocation, it uses the actual host name instead of the host's IP address.

The moral here is to “try it everywhere.” Once we got by this little detour, we were the proud owners of a brand new baby BI environment. We were eager to play, but like new parents, we were going to get some on-the-job training.

OLD DOGS DO NEW TRICKS (Can We Have Our Biscuit Now)

Our plan was to use the knowledge from our SAS course on Stored Processes to turn legacy code into web executables. The next step was to get Java to interact with the SAS environment and run these Stored Processes. We relied heavily on the SAS 9.1.3 Integration Technologies Developer's Guide[15] to point us in the right direction. This is a critical document if you want Java to “front end” the SAS BI environment.

We found we could run stored processes without creating any users beyond SAS users that were authorized to access the library containing the stored process. We were bad: we used the SASDemo user and native connections to Oracle in the stored process code. In retrospect, we would not recommend doing this. What we did was to temporarily bypass most of what you need to learn about Metadata administration, while we focused on running a stored process from Java. Our SAS “Users” were actually Java stored process runners authorized to run particular collections of stored processes. We were very focused on doing this one thing so we could show an actual result and build confidence. It worked.

For a side activity this seemed pretty cool. The total 25% to 30% part time staff commitment was:

- 2 Sun Server/Apache staff
- 1 Developer doing Java and Java script
- 1 SAS Programmer
- 1 Oracle DBA who “was almost ready to be volunteered” to do Metadata Administration
- 1 installer from SAS for 3 days

We had collectively spent parts of 5 FTEs time over a four month period. Total hours were about 600. We now had a basic Business Intelligence environment.

A typical problem for a small shop, doing a lead-from-the-middle,[16] fringe project – is that life intervenes. Ongoing operations required our time and we were swamped with other non-SAS work. The Metadata authorization structure still loomed big as “Something we need to look at.” We were able to truthfully say to our management that we had moved beyond the theoretical aspects of BI.

PIN YOUR LOCAL FRAMEWORK (Can You Find Yourself on the Map?)

While our SAS efforts were temporarily stymied by the crush of other business needs, we did acquire formal BI training from The Data Warehouse Institute.[16] This helped widen our perspective on other aspects of BI such as Data & BI Governance, and Master Data Management. We also saw SAS representatives more than hold their own in a “shoot-out” with other leading BI vendors. Even as success and an increasing technical understanding of your deployment move you to “bond” with your implementation, it is important to maintain outside connections to the “ecology” of the BI environment.

Although you might maintain a low profile as you gather BI expertise, eventually you need to work with (or help found) your BI Governance group. The characteristics of this group are highly dependent on your local corporate culture. You need to participate in the group as issues like data source inclusion, scalability, metadata structures, layers and authorizations are considered. It would be both a tedious and an error prone activity to push too far down an assumed path and then have to backtrack to a different structure. This is another area where a chicken and egg approach can minimize the extent of any errors you make as you grow your implementation.

SECURITY, JAVA AND DEFENSE IN DEPTH (Oh Brave New World)

We got back into the swing of things after SAS Global Forum 2007. A vacant Java programming position was filled which helped us clarify our delivery framework. Our java coders were tasked to develop a stored process runner with the stipulation that a defense-in-depth strategy was a primary concern. They developed Java/SAS front end applications that functioned to provide a single point of end user entry into an applicable subset of SAS stored processes. In the SAS® Management Console group level permissions were used to define access to the underlying Stored Processes. Individual User Authentication and Authorization were set to be maintained at the Java Application level. Depending on the application, any additional parameters required by the SAS® Stored Process that is invoked are gathered at the Java level. This two-tiered approach to SAS Stored Process access allows for simplified account maintenance at the SAS level, as well as seamless integration of disparate authentication/authorization mechanisms that exist at the enterprise level.

In addition to the logical separation of authentication from application authorization within the Java front ends, Java acts as a proxy to the SAS Stored Process Server. Only the Java Application Server is given firewall access to SAS, thus acting as a separate, preferably physically separate, gatekeeper. You can also front the Java Application Server with an HTTP reverse proxy server to regulate communications with the end user. By convention, in a reverse proxy scenario, the end users are on the "outside" of the reverse proxy and the application server is on the "inside." The HTTP reverse proxy examines URLs in HTML documents passing through it and rewrites the HTTP response headers to point to the reverse proxy for outbound packets bound for end user consumption. The HTTP request headers are rewritten to point to the application server for inbound packets.

Thus, ultimately, three physical tiers exist between the end user and the SAS Stored Process being invoked. With SAS accessing the Enterprise Database, your company's data stores are separated by yet another application layer. The physically separate server strategy is an implementation of the "defense-in-depth" philosophy, whereby a potential hacker would need to gain access through multiple physical security layers before gaining access to your company's data.

We use Apache Tomcat to power our SAS-Web applications. Our basic SAS-Web application bundle is a collection of the following items:

- Servlets
- JavaServer Pages
- Utility Classes
- Static Documents including XHTML, images, etc.
- Client side classes
- Meta information that describes the web application

The resources are bundled and run on multiple containers that reside in the web layer of the application.

We knew that some of our planned applications could require access from anywhere. To address security issues such as insecure communications, compromised data protection, parameter tampering and revealing too much about our internal structure, we implemented features of the Java Cryptography Extension (JCE) API. This provides a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. Support for encryption includes symmetric, asymmetric, block, and stream ciphers.

While the Java delivery mechanism was being coded and tested the SAS coders set out to lure that segment of the client community for whom IT means Microsoft Office. Teaser demonstrations of the SAS® Add-in for Microsoft Office (AMO) let us try out the same stored processes that were destined for internet use from MS Word and Excel. This helped generate some buzz about the new SAS and how it could boost the power and reach of the MS Office Suite. One of our problems has been overcoming the old-SAS/new-SAS ingrained perception problem.

Still webification (actual technical term) was what we were most interested in. We wanted to reach out and touch the biggest audience. We chose to defer our interaction with the desktop power-user for a later time.

END OF PHASE II (A New Round Of Learning)

July of 2007 found us twelve months into this project’s timeline. Staff had acquired new skill sets, and as a group we were feeling more comfortable with the new BI environment. Without really noticing, acculturation had crept up on us. Group conversations began to include BI terminology in an unstudied way.

As we looked ahead the same knowledge that had improved our comfort level led us to realize there was more that needed to be done. In order to create a production deployment we needed to firm up front-end security elements and in our case authenticate to our campus level access authorization structure. We also had a cranky object spawner balancing setup that we did not understand very well. There was a further concern that our deployment would not meet the graphical output demands of the end user group we had identified for our initial rollout. We realized there was a lot we did not know, but knew where to look.

TUNING AND VALIDATION (Expert Advice from SAS Consulting)

Deployments of new technologies run the risk of within-group self-referencing – meaning the group forms a consensus about how things should be by talking only to each other. In order to avoid that trap we agreed the time was right for external advice and validation. We turned to SAS Consulting to fill a very specific need. We asked SAS if they could send us someone for two weeks who could role-play “as if” they were an employee in our group who had been in charge of the SAS BI environment but who was leaving the company. Our role playing consultant would then need to review the relevant aspects of the deployment with the appropriate staff member(s). Together they would engage in knowledge transfer and turnover.

This was our single biggest line item purchase to date and the knowledge transfer was just what the doctor ordered. As a result, we tightened up our Java interaction with the stored process server, expanded load balancing, reviewed and set up an authorization structure that accommodates several downstream possibilities, and successfully moved one of our Solaris Containers to a separate Sun box without disturbing the install.

Each of the above activities took about two days. The key lesson from this experience is that expert advice, in your local context, at the appropriate time, is invaluable. There was a one week break in the middle of the two weeks. This really focused the second week's questions and added a sense of urgency.

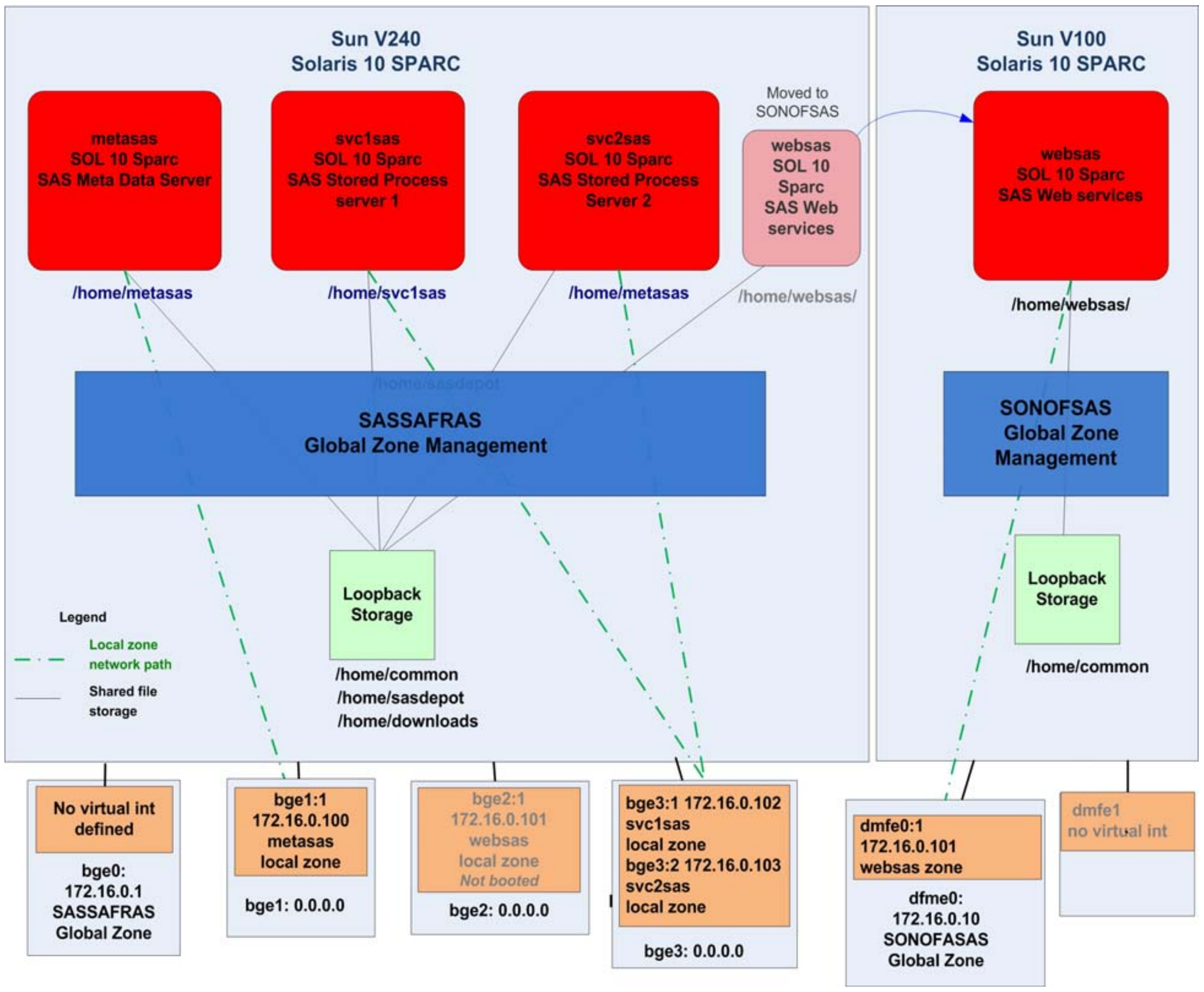
SOMETHING WONDERFUL RIGHT AWAY? (A New BI Opportunity)

As we began the process of transitioning from a prototype to production, we received a request to "maybe do something with this BI stuff." The request was to participate in a demonstration for a Translational Research project that had a missing presentation and analysis component once the build of a warehouse was complete. We worked with our new project partners. Thirty days after the warehouse came live we demonstrated access to several stored processes developed for secure query from browsers, MS Office and SAS® Enterprise Guide. This confirmed the value of the deployment and helped to demonstrate the versatility of SAS BI. The important lesson for us was that once you have set up a framework along these lines, it's pretty versatile – with component techniques that are easily reused.

SUMMARY

By starting small, and pacing a trial BI deployment, you can gain considerable insight into many aspects of the care and feeding of your BI efforts. You can also test the receptiveness and readiness of various elements of your user community. This will guard against great deployments that have no fan base. Correctly gauging the inclination and ability of your local culture is at least as important as getting the architectural aspects of the deployment right. You don't need to concern yourself with getting everything right the first time so much as you need to make sure you learn the important lessons that each step on the way has to offer. In this way you can move from learning what you don't know to knowing what you need to know. Having a small prototype to play around with also allows for quick demonstrations and extends your ability to sensibly "what if" your future plans. You will discover new strengths and weaknesses. A Solaris 10 containers environment for BI installs is a very inexpensive and flexible avenue for small groups to gain knowledge and experience of the many ways that SAS BI can be utilized. It also provides scalability should one of your quick demonstrations take off. By taking a phased approach that generates a return of knowledge at each phase, you can alter plans to meet conditions on the ground, without sacrificing the experience you have gained and avoid the all-or-nothing trap.

THE BIGGER EGG – WEBSAS CONTAINER TRANSPORTED TO ITS OWN SERVER


Sun V240
Solaris 10 SPARC
metasas SOL 10 Sparc SAS Meta Data Server
svc1sas SOL 10 Sparc SAS Stored Process server 1
svc2sas SOL 10 Sparc SAS Stored Process Server 2
Moved to SONOFSAS
websas SOL 10 Sparc SAS Web services
/home/metasas
/home/svc1sas
/home/metasas
/home/websas/
/home/sasdepot
SASSAFRAS Global Zone Management
Loopback Storage
/home/common
/home/sasdepot
/home/downloads
Legend
Local zone network path
Shared file storage
Sun V100
Solaris 10 SPARC
websas SOL 10 Sparc SAS Web services
/home/websas/
SONOFSAS Global Zone Management
Loopback Storage
/home/common
No virtual int defined
bge0: 172.16.0.1 SASSAFRAS Global Zone
bge1:1 172.16.0.100 metasas local zone
bge1: 0.0.0.0
bge2:1 172.16.0.101 websas local zone Not booted
bge2: 0.0.0.0
bge3:1 172.16.0.102 svc1sas local zone
bge3:2 172.16.0.103 svc2sas local zone
bge3: 0.0.0.0
dmfe0:1 172.16.0.101 websas zone
dfme0: 172.16.0.10 SONOFASAS Global Zone
dmfe1 no virtual int

SAS BI Server Components

Sassafras	Sonofasas
1 GHz UltraSPARC-IIIi CPU, 64 bit RISC 4-way Superscalar Pipeline SPARC® V9 Architecture (scalable to 2 CPUs)	500 MHz UltraSPARC-IIe CPU, 64 bit RISC 4-way Superscalar Pipeline SPARC® V9 Architecture
64 KB data and 32 KB instruction L1 cache and 1 MB integrated L2 cache	16 KB data and 16 KB instruction L1 cache and 512 KB integrated L2 cache
128-bit memory interface with 1776 MB/second throughput (single CPU configuration)	64-bit memory interface with 800 MB/second throughput
4GB 133 MHz 128 bit ECC DIMM memory (scalable to 16 GB)	2 GB 133 MHz 64bit ECC SDRAM memory (maximum compliment)
4 10/100/1000 BaseT Ethernet interfaces	2 10/100 BaseT Ethernet interfaces
1 10BaseT Ethernet server management port	
1 Integrated Ultra160 SCSI multimode (SE/LVD) host adapter, dual channel (internal and external interfaces)	100 MBps Ultra-DMA mode 5 (Ultra/100) 16.7 MB/sec burst rate EIDE peripheral interface
4 internal hot-swap Ultra320 SCSI hard drives: 2 10K rpm 36GB drives, 8 MB cache, 5 ms average seek time; 2 10K rpm 73GB drives, 8 MB cache, 4.5 ms average seek time	2 7200 rpm IDE disks, 40 GB Barracuda ATA IV drive, 2 MB cache, 9 msec average seek time
Dual redundant power supplies	

REFERENCES

1. DiRomualdo, Tony. "Don't rely on information technology to change corporate culture." 2004 http://wistechnology.com/article.php?id=1308
2. Perfetti, Christine. "Debunking the Myths of Innovation: An Interview with Scott Berkun" 2007 http://www.uie.com/articles/myths_of_innovation/

   Kawasaki Guy "Ten Questions with Scott Berkun,Author of "The Myths of Innovation" 2007 http://blog.guykawasaki.com/2007/06/ten-questions-w.html

3. Home Improvement (TV series) http://en.wikipedia.org/wiki/Home_Improvement
4. Jack Sparrow http://en.wikipedia.org/wiki/Jack_Sparrow
5. List of problems solved by MacGyver http://en.wikipedia.org/wiki/List_of_problems_solved_by_MacGyver
6. Quotes by Browning, Robert http://quotationsbook.com/quote/2053/
7. Third Party Software Requirements for use with SAS® Products http://support.sas.com/documentation/configuration/thirdpartysupport/v913sp4/thirdparty913sp4.html#httpserver
8. litmus - WebDAV server protocol compliance test suite http://www.webdav.org/neon/litmus/
9. Internet Engineering Task Force (IETF) http://www.ietf.org/rfc/rfc2518.txt
10. Solaris Freeware Project http://www.sunfreeware.com/
11. Sun Developer Network - Java Secure Socket Extension (JSSE)- How to export private key from keystore http://forum.java.sun.com/thread.jspa?threadID=154587&messageID=449486&forumID=2
12. Olafsen, Lars: Postfix-users, Readlist.com, 2007, 'TLS1_SETUP_KEY_BLOCK:cipher or hash unavailable' problem on Solaris 10 + possible solution http://readlist.com/lists/postfix.org/postfix-users/15/79988.html
13. KNOWLEDGE BASE / SAMPLES & SAS NOTES "Usage Note 18503: Cannot connect to Stored Process Server on UNIX from client applications on Windows when NAT is used" http://support.sas.com/kb/18/503.html
14. KNOWLEDGE BASE / SAMPLES & SAS NOTES "Problem Note 15049: Running the Stored Process Server on a network that uses NAT (Network Address Translation)" http://support.sas.com/kb/15/049.html
15. KNOWLEDGE BASE/SAMPLES& SAS NOTES "SAS 9.1.3 Integration Technologies » Developer's Guide" http://support.sas.com/rnd/itech/doc9/dev_guide/dist-obj/javaclnt/found/found_stprocess.html
16. "Leading From The Middle: Issues and Answers On Leadership For Middle Managers." http://www.babsoninsight.com/contentmgr/showdetails.php/id/594
17. The Data Warehousing Institute™ http://www.tdwi.org/

ACKNOWLEDGMENTS

Phil McDermott philm@uic.edu Java Coding & Web Environment

Donnie Atkinson donalda@uic.edu Solaris Setup and Split out

Suman Suman suman@uic.edu Java and Security Coding

RECOMMENDED READING

M. Andrews, J. Whittaker, How to Break Web Software. Boston, Addison-Wesley 4th Printing, 2006

J. Dyché, E. Levy, Customer Data Integration: Reaching a Single Version of the Truth, Hoboken, Wiley 2006

The Open Web Application Security Project (OWASP), 2008,
http://www.owasp.org/index.php/CLASP_Security_Principles

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Donald Atkinson, Patricia Konstantinopoulos, Ian Huggins
University of Illinois at Chicago - College of Medicine
1853 W. Polk St, MC 785
Chicago, IL. 60643
Work Phone: 312-996-1627, 312-996-9602, 312-996-7472
Fax: 312-355-0253
E-mail: atkinson@uic.edu, pkonstan@uic.edu, huggins@uic.edu