SAS Migration Host > AIX

Daniel Schulte
viadee
Unternehmensberatung
GmbH
Anton-Bruchausen-Straße 8
48147 Münster
daniel.schulte@viadee.de

Carsten Zaddach
BDE
Business Datawarehouse
Engineering GmbH
Weg am Krankenhaus 2
15711 Königs Wusterhausen
info@bde-gmbh.de

Zusammenfassung

Um Prozesse vom Großrechner auf andere Architekturen zu migrieren, sind meist umfangreiche Umstellungsaktivitäten einzuplanen. Datenbestände müssen identifiziert, Jobs auf die neue Plattform gebracht, Prozessabläufe modifiziert und Daten und Schnittstellen synchron gehalten werden.
Das Z2AIX Framework soll diese Aktivitäten weitgehend automatisieren. Hierbei werden zentrale Komponenten auf dem Host und der AIX modifiziert bzw. ergänzt und so eine Jobausführung - gesteuert vom Host - auf der neuen Plattform ermöglicht. Neben der Datensynchronisierung zwischen den beiden Welten werden Ausgaben für Nachfolgejobs wieder auf den Host zurück geführt und Statusinformationen sowie Logs ins Beta92 gebracht.
Erfahren Sie mehr über mögliche Einsatzszenarien, potentielle Einsparmöglichkeiten und Grundlagen, welche für einen erfolgreichen Einsatz benötigt werden.

Schlüsselwörter: Host, zOS, AIX, Migration, Verteilte Daten, Datensynchronisation

1 Ausgangslage

In über 20 Jahren SAS Nutzung auf dem Host unter zOs ist ein komplexes Netz aus mehr als 2600 Jobs mit umfangreichen Datenbeständen (ca. 85 TB) entstanden, welches nun auf AIX migriert werden soll. Hierbei sollte aber die Jobnetzkontrolle weiterhin auf dem Host verbleiben, die Verarbeitung allerdings auf die AIX wandern und das damit notwendige SAS Paket nur noch auf einer Plattform verwaltet und lizensiert werden müssen.
Normalerweise müssen für die Migration Daten / Sourcen und Steuerroutinen auf die neue Plattform gebracht und hier neu eingerichtet werden. Bei dem ermittelten Umfang war dies nicht als “BigBang” durchführbar. Bei einer schleichenden Migration sind aber auch die Vor- und Nachfolger einzelner Jobs zu berücksichtigen, sprich: Datenbestände müssen auf beiden Plattformen synchron gehalten werden!

2 Ausgangslage

Um zunächst einen Überblick über die genutzten Jobs mit den verwendeten Bibliotheken und Daten zu bekommen, wurde eine Routine als “TERMSTMT” in den SAS Aufruf zentral implementiert, um diese Daten zu erfassen. Hierbei wird zum Ende eines jeden SAS Jobs das Data Dictionary ausgelesen und in einen eigenen Datenbestand fortgeschrieben. Da ein Großteil der Allocationen über die JCL (Job Control Language) vorgenommen wird, sind diese Informationen auch noch bei Job Ende verfügbar und werden damit erfasst.

Hier sind neben den Informationen aus den Tabellen dictionary.libnames und dictionary.extfiles auch noch folgende Metadaten interessant:

- Jobname (Macrovariable sysjobid)
- Startdatum (Macrovariable sysdate9)
- Startzeit (Macrovariable systime)
- User (Macrovariable sysuserid)
- ggf. Knoten (Macrovariable systcpiphostname)

Was nicht erfasst wird sind dynamische Allokationen, welche erst im Job zur Laufzeit vorgenommen und manuell bereinigt werden. Um auch diese Daten zu erfassen und die notwendige Dynamik für die Ansprache des Dateisystems zu erhalten (zOS: QUALI.FIER.DURCH.PUNKTE.GETRENNT AIX: /Pfad/mit/Schrägstrichen/) wird statt des FILENAME bzw. LIBNAME Statemens ein Makro verwendet, welches die Pfadauflösung je nach Ausführungsumgebung parametrisiert.
Neben der Information, welche Jobs, welche Daten und Dateien benutzen, können so auch Informationen zur Dateigröße bzw. deren Auslastung ermittelt werden.

3 Die Idee

Da jeder SAS Aufruf in der JCL über eine zentrale Aufrufroutine abgewickelt wird, sollte dies der Ansatzpunkt für die Migration werden. Wenn man hier nun entscheiden könnte, wo ein Job ausgeführt wird - eine Art Weiche - könnten sukzessive einzelne Jobs auf die gewünschte AIX Umgebung gebracht werden und die angrenzenden Jobs würden davon nichts mitbekommen. Dies bedingt, das mit dieser “Weiche” auch eine Datensynchronisierung einhergeht, um Ausgaben der Vorläufer und Eingaben für Nachfolger auf beiden Umgebungen auf dem richtigen Stand zu haben. Weiter muss eine Zugriffsschicht für sequenzielle Files und Sourcen eingeführt werden um diese Konsistent vorzuhalten. Modifikationen sollten möglichst gering gehalten werden. Plattformspezifischer Code muss “dynamisiert” werden, damit z.B. ein gemeinsam genutzter Code weiterhin auf beiden Plattformen funktioniert. Hierzu können die Ausprägungen der Makrovariable SYSSCP genutzt werden.

4 Architektur

Um allen Anforderungen gerecht zu werden muss das Framework mehrere Komponenten mit sich bringen.

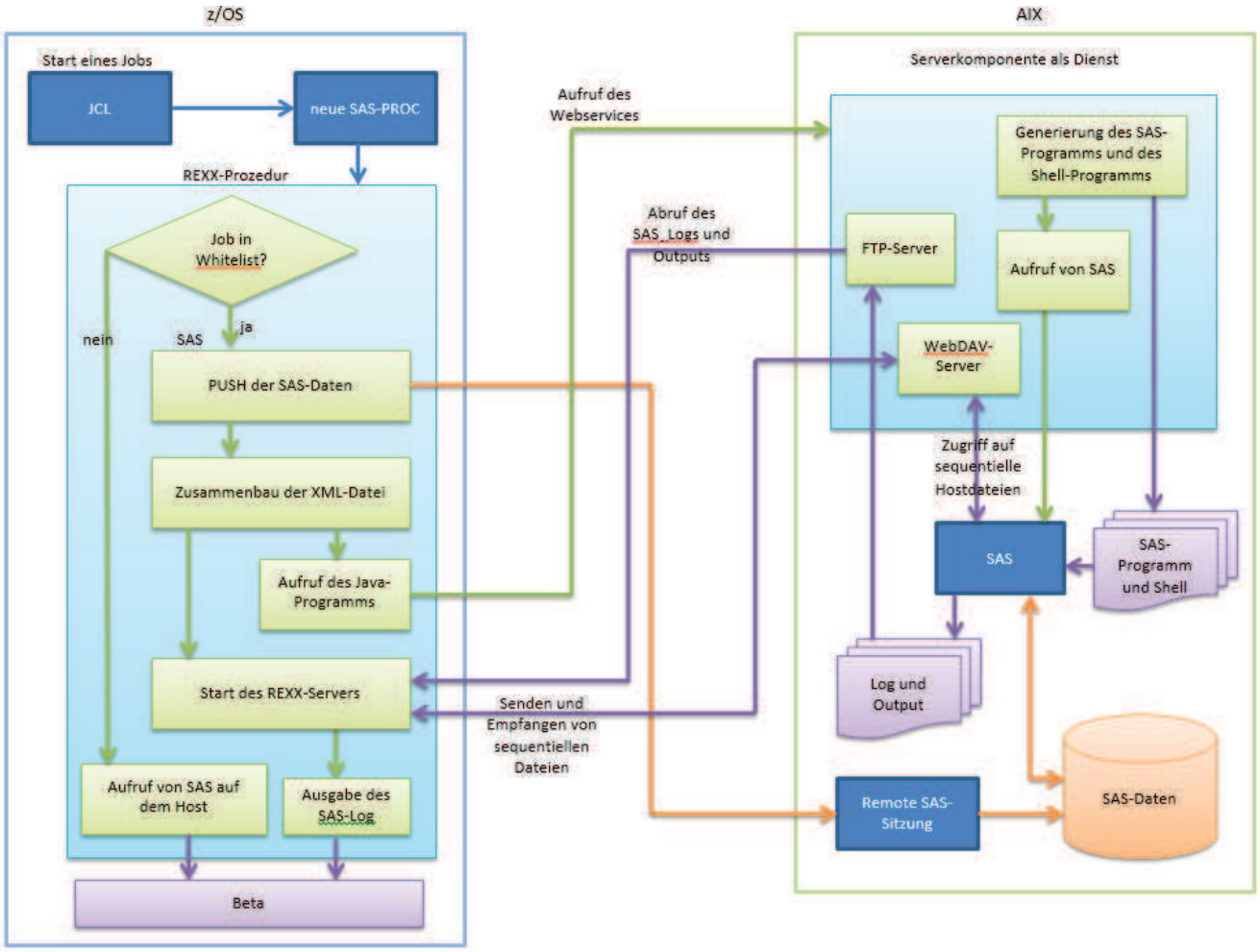

Abbildung 1: Architekturüberblick

4.1 REXX Prozedur (HOST)

Die REXX Prozedur ist der hostseitige Kernbestandteil des Frameworks. Diese wird in der systemseitigen Proclib auf dem Host eingespielt und ersetzt die bestehende SAS Prozedur. Damit laufen nun alle Jobs in diese Routine hinein. Um nun festzulegen, ob ein Job auf dem Host oder AIX ausgeführt werden soll, wird eine Liste mit Jobnamen zur Steuerung eingesetzt. Über ein internes Flag wird die Steuerdatei entweder als White- oder als Blacklist interpretiert. Als Whitelist werden nur Jobs, die darin vermerkt sind, auf die AIX geschickt. Als Blacklist werden alle Jobs bis auf die dort aufgeführten der AIX übergeben.
Ein Job, der weiterhin auf dem Host ausgeführt werden soll, wird nun der bisherigen SAS Start Routine übergeben und wie vorher ausgeführt. Zur Ausführung auf der AIX wird als nächstes die JCL analysiert, welche DD Statements auf SAS Bibliotheken verweisen. Zur Identifizierung wird die SAS typische Signatur der sequenziellen Daten

verwendet. Diese und weiter bekannte Datenquellen (z.B. dynamische Allokationen) werden dann dem Synchronisierungsprozess übergeben. Die kompletten Daten der JCL werden dann per XML Stream an die Java Server Komponente überstellt, welche sich dann um die weitere Ausführung kümmert.
Ergebnisse der Ausführung werden im Nachgang dann auf der Hostseite wieder entgegen genommen. Logs und Status Informationen wandern wie gewohnt in das SDSF und ggf. in Archivsysteme wie BETA92.

4.2 Java Server Komponenten (AIX)

Auf der AIX ist die andere Kernkomponente des Frameworks als Java Webservice zu finden. Diese ist als dauerhafter Dienst eingerichtet und wartet auf Input von der Host Komponente. Wird ein Job empfangen, wird aus dem XML Stream ein Shell Script und eine SAS Source gebaut, welches dann auf den synchronisierten Datenbestand ausgeführt wird. Status Informationen werden ebenfalls per XML Stream zurück kommuniziert, Logfiles werden transportiert und Daten werden im Nachgang erneut synchonisiert.

4.3 Datentransfer (FTP / WebDAV)

Für den Datentransfer von Sequenziellen Files sind zwei Transportwege eingerichtet. Zugriffe auf Sourcen und sequenzielle Dateien werden direkt per WebDAV durchgeführt. Für größere Dateien ist WebDAV aber eher ungeeignet und daher wird für diese noch der Weg per FTP zur Verfügung gestellt. Dieser ist der Standard-Weg für LOG und OUTPUT der SAS Verarbeitung.

4.4 Synchronisierung

Um die Daten auf beiden Plattformen synchron zu halten, wird der Jobausführung ein Prozess vor- bzw. nachgeschaltet, der dies übernimmt. Bei der allerersten Ausführung wird dieser Prozess eine leere AIX Lib vorfinden und den führenden Host bestand. In diesem Fall werden die Daten komplett übertragen, was natürlich zunächst der Laufzeit negativ beiträgt. Ist der Datenbestand allerdings auf der AIX schon vorhanden, werden lediglich Tabellen übertragen, die auf dem Host aktueller sind. Damit wird der zusätzliche Aufwand erheblich reduziert. Nach der Ausführung wird wieder synchronisiert und die geänderten AIX Tabellen wandern wieder auf den Host.
Bei der Ausführung von nur einzelnen Jobs einer Kette auf der AIX ist durch den erhöhten Transferaufwand sicher keine Laufzeitwunder für den kompletten Job zu erwarten. Wird jedoch eine ganze Kette als Strang migriert, sinken die notwendigen Transferzeiten.

5 Herausforderungen

5.1 Ressourcenverwaltung von zOS

Die Ressourcenverwaltung von zOS ist für die Ausführung von mehreren Jobs, die auf die gleiche Ressource zugreifen wollen, ein Segen, aber genau dieser Segen wurde zum Fluch. Ist eine Ressource für den schreibenden Zugriff durch den ausgeführten Job exklusiv gesperrt, so kann niemand außer dieser Job auf die Ressource zugreifen. Somit war es technisch nicht möglich, auf dem Host eine zentrale Serverkomponente zu installieren, die alle Anfragen der AIX-Komponenten bearbeitet, da diese Komponente keinen Zugriff auf die Ressourcen gehabt hätte.
So musste stattdessen innerhalb eines jeden abgeschickten Jobs eine eigener kleiner Server implementiert und gestartet werden.

5.2 Analyse der Job-JCL

Die Analyse der Job-JCL gestaltete sich als eine der größten Herausforderungen. Eine textbasierte Analyse der JCL war nicht möglich, da die Quelle der JCL zum Zeitpunkt der Ausführung nicht bekannt ist. Zum Glück verwendet zOS eine Vielzahl von internen Tabellen/Control Blocks für die Verwaltung von (fast) allen Ressourcen und für das Ablegen von Informationen über einen Job. Sind die Formate der Control Blocks und der Ort an dem sie sich befinden bekannt, so können die enthaltenen Informationen über ein REXX-Programm relativ problemlos ausgelesen werden.

5.3 Unterschiedliche Zeichensätze

Die unterschiedlichen Zeichensätze von zOS und AIX waren, speziell mit Blick auf einige Sonderzeichen, ein Problem während der Umsetzung. Die fehlerhafte Umsetzung von Zeichen führte in der Folge zu Fehlern, die durch Analyse der Logdateien auf den ersten Blick nicht sichtbar waren, sondern erst in einer aufwendigen Tiefenanalyse erkennbar wurden.
Als Lösung wurde für eine handvoll Zeichen in der Host-Komponente eine Funktion implementiert, die die Konvertierung dieser Zeichen übernahm.

5.4 WebDAV-Funktionalität

SAS unterstützt als filename-Engine eine Reihe von unterschiedlichen Zugriffsmethoden, darunter auch WebDAV. Der Vorteil von WebDAV gegenüber anderen Engines für den entfernten Zugriff auf Dateien ist, das WebDAV auch das schreiben von Dateien und die Ausgabe von “Verzeichnissen” unterstützt. Leider gibt es eine Reihen von WebDAV-Implementierungen für unix-ähnliche Betriebssysteme, aber keine Implementierung für zOS. Ein Grund dafür mag die oben angesprochene Ressourcenverwaltung sein.

Somit musste ein komplett neuer WebDAV-Server unter AIX implementiert werden, der die Verwaltung von MVS-Namen unterstützt und mit der Komponente auf dem Host zusammenarbeitet.

5.5 Identifizierung/Überführung aller SAS-Tape-Bestände

Neben den normalen SAS-Datenbeständen, deren Identifizierung im Abschnitt "Datenerfassung" beschrieben ist, wurden eine Reihe von SAS-DataSets auf Bänder ausgelagert. Hintergrund dafür waren in der Regel gesetzliche Vorschriften. Diese Daten mussten natürlich auch auf die neue Plattform migriert werden. Im Gegensatz zu den normalen SAS-Beständen wurde diese Bestände aber nicht automatisch durch das Analyseprogramm (siehe "Datenerfassung") erfasst. In vielen Fällen waren diese Bestände auch nicht mehr im Katalog des zOS-Systems vorhanden. Erst eine Analyse mit Unterstützung der Systemabteilung brachte diese Bestände wieder ans Licht.

5.6 Identifizierung/Umstellung von plattformspezifischem Code

Ein großer Vorteil von SAS-Programmen ist deren überwiegende Plattformunabhängigkeit. Jedoch gibt es einige Befehle in SAS, für die die Plattformunabhängigkeit nicht gilt. Als klassische Vertreter dieser Befehle gelten die filename- und die libname Anweisungen. Durch die Allokation von Dateien und Bibliotheken in der JCL sind bei einem Großteil der Programme diese Probleme schon im Vorfeld beseitigt. Jedoch gibt es einige Programme, in denen die Allokation dynamisch erfolgt. Diese und andere Fälle mussten durch einen Codescanner gefunden werden, um schon im Vorfeld den plattformspezifischen Code durch spezielle Makros und somit eine reibungslosere Übernahme der Programme nach AIX zu ermöglichen.

6 Fazit / Ausblick

Mit der gezeigten Lösung kann auf dem Weg zum langfristigen Ziel - der Host soll „sterben" - ein guter Schritt nach vorne gemacht werden. Dadurch, dass die Umstellung schleichend erfolgen kann, sind die Außenwirkungen sehr gering. Die notwendigen Skripte werden auf der Zielplattform automatisch generiert und können in einer Überarbeitung der Jobplanung direkt als lauffähige Startpunkte verwendet werden. Quellcode und Jobs können zunächst auf dem Host verbleiben, aber die Pfade lassen sich einfach auf einen lokalen Pfad in der Zielumgebung anpassen. Die Adressierung erfolgt im SAS Code typischerweise nach dem Muster Logischername(Membername). Wird dieser logische Name angepasst, läuft der SAS Code einfach weiter, da das Mapping zwischen Logik und Physik bei der Initialisierung erledigt wird. Ebenfalls zu diesem Zeitpunkt kümmert sich das Framework um die Synchronisation der SAS Daten. Diese wird auf Tabellenbasis vor und nach einem Joblauf durchgeführt um sicher zu sein, dass beide Umgebungen auf dem gleichen Datenstand sind. Auch die Rückführung der Logs und Ausgaben in Archivsysteme wie Beta 92 ist implementiert. Die revisionssichere Nachverfolgbarkeit von Jobläufen ist damit auch gewährleistet!