SAS Base – Darf's ein bisschen schneller sein?

Sebastian Reimann
viadee Unternehmensberatung GmbH
Anton-Bruchausen-Str. 8
48147 Münster
sebastian.reimann@viadee.de

Zusammenfassung

Das ETL Programm ist fertig und es läuft! Und läuft. Und läuft -- und läuft... Wer kennt das Problem nicht? Analysiert man das SAS Log, findet man schnell die Schritte mit langer Laufzeit. Beobachtet man jedoch während der Ausführung der relevanten Schritte die Systemauslastung, stellt sich häufig die Frage, warum der Schritt so lange läuft, wenn doch die System-CPU nur zu wenigen Prozent ausgelastet ist.

SAS Visual Analytics und SAS High Performance Analytics können ein Ausweg sein. Aber: Gibt es nicht auch im klassischen SAS BASE Wege, um die Ausführungszeiten der Programme zu optimieren?

Im Folgenden werden verschiedene Möglichkeiten beleuchtet, wie SAS Programme optimiert werden können, um die Laufzeit zu optimieren.

Schlüsselwörter: SAS Base, SPDE, Parallelisierung, SYSTASK, SAS/Connect, ETL, xCommands

1 Motivation

Wer kennt das Problem nicht. Bei der Ausführung von ETL Jobs kommt es häufig auf jede Minute Ausführungszeit an. Der neue ETL Job ist fertig entwickelt und getestet, doch in Produktion läuft der Job viel zu lange.

Analysiert man anschließend das SAS Log, so lassen sich schnell die Schritte ermitteln, die zu der erhöhten Joblaufzeit führen. Eine Analyse zeigt dabei häufig, dass die lange Ausführungszeit nicht auf fehlende Rechenressourcen zurückzuführen ist; die im System genutzte CPU wird i.d.R. nur zu einem Bruchteil ausgelastet.

Im Rahmen dieser Ausarbeitung soll das Phänomen analysiert werden und es sollen mögliche Lösungsalternativen vorgestellt werden, um die Ausführungszeit zu optimieren. Dabei soll der Fokus explizit nicht auf die neusten Technologien von SAS (High Performance Analytics) gelegt werden. Vielmehr soll untersucht werden, wie mit klassischen SAS-Mitteln eine Lösung gefunden werden kann.

2 Technische Grundlagen

Der Grund für die Ausführungszeit eines SAS Programms ist u.a. in der Art und Weise zu finden, wie SAS das Programm abarbeitet. Sämtliche Schritte des Programms werden sequentiell nacheinander verarbeitet. Für die Verarbeitung der einzelnen Schritte werden jedoch nicht immer alle im Rechner zur Verfügung stehenden Ressourcen genutzt. Viele SAS Prozeduren und vor allem der DATA Step werden nicht mit Threads (Ausführungsreihenfolge in der Abarbeitung eines Programms) auf mehrere CPUs verteilt sondern arbeiten single-threaded nur auf einer CPU. Die übrigen CPUs des Rechners bleiben ungenutzt.

Mit einer neuen SAS Version ist es möglich, dass für einzelne Prozeduren eine Optimierung seitens SAS erfolgt, so dass diese Prozedur künftig Multi-Thread-Verarbeitung unterstützt. Seit der Einführung der SAS High Performance Analytics ist es jedoch häufig zu erkennen, dass nicht die klassische SAS Prozedur optimiert wird, sondern dass eine neue High Performance Prozedur bereitgestellt wird, die alternativ zur klassischen Prozedur genutzt werden kann. Ein Beispiel hierfür wäre z.B. die Prozedur PROC HPLOGISTICS die als High Performance Pendant zur PROC LOGISTICS genutzt werden kann[1]. Problematisch bei dieser Vorgehensweise ist jedoch, dass hierfür i.d.R. zusätzliche SAS Lizenzen bzw. spezielle Hardwarekomponenten (SAS LASR Analytics Server) notwendig sind.

Seit Version SAS 9.1 wurden auch einzelne SAS Prozeduren um multi-threading Fähigkeiten ergänzt. So unterstützen bspw. folgende Prozeduren inzwischen mehrere Threads und können so aktuelle Hardwarekomponenten effektiver ausnutzen:

- PROC MEANS
- PROC REPORT
- PROC SORT
- PROC SQL
- PROC SUMMARY
- PROC TABULATE

Dennoch bleiben eine Vielzahl von SAS Prozeduren übrig, die auch in der aktuellen SAS Version 9.4 auf einen Thread begrenzt sind. Besonders zu erwähnen ist hier der klassische DATASTEP, der in einer Vielzahl an ETL Prozessen Verwendung findet.

Um auch mit dem DATASTEP die gesamte Performance eines Rechners ausnutzen zu können sind Erweiterungen nötig, die im Folgenden analysiert werden sollen.

[1] Vgl. http://support.sas.com/documentation/cdl/en/stathpug/66410/HTML/default/viewer.htm#stathpug_hplogistic_overview02.htm

3 Das Praxisbeispiel

Im Folgenden sollen verschiedene Optimierungsmöglichkeiten an einem Praxisbeispiel vorgestellt werden. Bei dem Beispielprozess handelt es sich um ein einen eher trivialen ETL-Prozess, der im Alltagsgeschäft häufig vorzufinden ist.

Im Rahmen des ETL-Prozesses sollen die Kundendaten aus verschiedenen Quellsystemen zu einem konsolidierten Kundenstamm zusammengefügt werden. Die Eingangsdaten stehen jeweils als Textdatei zur Verfügung. Modelliert man den Einleseprozess, so erhält man in Etwa folgende schematische Darstellung.

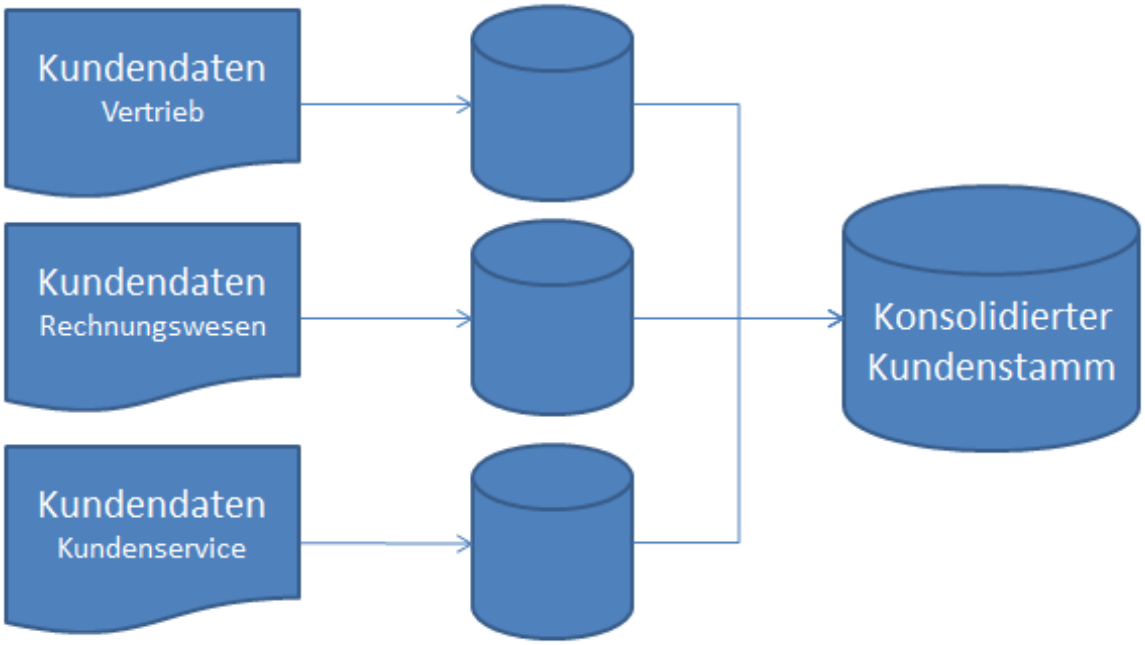

Abbildung 1: Beispielprozess zur Erstellung eines konsolidierten Kundenstamms

Im klassischen SAS Programm werden alle Steps sequentiell nacheinander ausgeführt und später zu einem gemeinsamen Bestand zusammengefasst.

In der Analyse des Prozesses lassen sich zwei Bereiche zur Optimierung erkennen. Zum einen kann es aufgrund von I/O-Beschränkungen zu einem Performanceverlust kommen. Auf der anderen Seite kann die CPU ein limitierender Faktor sein, für die Optimierungsmöglichkeiten gesucht werden müssen.

4 BASE-Engine vs. SPD-Engine

Traditionell erfolgt die Datenhaltung in SAS mit der SAS BASE Engine. Hierbei werden Tabellen in Form von FlatFiles innerhalb eines Verzeichnisses auf der Festplatte gespeichert. Jede Tabelle wird als einzelne Datei gespeichert. Gerade bei großen Tabellen kann die Dateigröße stark anwachsen.

Soll eine derartige Datei verarbeitet werden, ist das Lesen und Schreiben der Daten durch die Performance der Festplatte beschränkt. Hierbei kann die Performance der einzelnen Festplatte, des RAID-Verbundes, des SAN Speichers oder der Datenanbindung über FibreChannel oder iSCSI als limitierender Faktor ermittelt werden.

Eine Performance-Verbesserung kann erreicht werden, indem der Datenzugriff auf mehrere Prozesse und mehrere Speichermedien aufgeteilt wird. An dieser Stelle kommt die SAS BASE Engine schnell an ihre Grenzen.

Im regulären Funktionsumfang von SAS steht die SPD-Engine als alternative Datenzugriffsmethode zur Verfügung. Die Syntax eines entsprechenden LIBNAME Statements sieht wie folgt aus:

```
LIBNAME libref SPDE (SPDE options);
```

Die wichtigsten SPDE Optionen sind in der folgenden Tabelle 1 dargestellt:

Tabelle 1: Wichtige SPDE Optionen im LIBNAME Statement

Option	Beschreibung
Partsize	Daten werden beim Schreiben in kleine Dateien aufgeteilt. Mit Partsize wird die Größe der gesplitteten Dateien angegeben. Default: 128 MB
Datapath	Der Datapath gibt die Verzeichnisse an, in denen die Daten gespeichert werden. Idealerweise werden hier mehrere Ordner auf verschiedenen Laufwerken angegeben, um optimale Performance zu erreichen. Daten werden entsprechend der Partsize sequentiell auf diese Ordner verteilt.
Metapath	Gibt den Ordner an, in dem die Metadaten zu den Tabellen gespeichert werden
Indexpath	Gibt den Ordner an, in dem die Indexdaten zu den Tabellen der Bibliothek gespeichert werden.

Weitere Optionen zur Performance-Optimierung sind in der SAS Dokumentation[2] aufgeführt.

Neben den SPDE Optionen, die beim Zuweisen der Bibliothek vorgegeben werden, können einige Einstellungen auch global über SAS Systemoptionen definiert werden. Hierzu zählt vor allem die Option SPDEMAXTHREADS, über die die maximale Anzahl an Thread vorgegeben wird, die beim Zugriff mit der SPD-Engine genutzt werden.

[2] Vgl. http://support.sas.com/rnd/scalability/spde/syntax.html

5 Parallelisierung

5.1 Überblick

Stellt sich bei der Analyse heraus, dass nicht I/O, sondern die CPU der limitierende Faktor in der Verarbeitung ist, so sollte man die CPU Nutzung näher analysieren. Aufgrund der Arbeitsweise von SAS wird gerade beim DataStep lediglich ein CPU Kern genutzt obwohl heutige Rechner viel mehr CPU Kerne zur Verfügung stellen.

Betrachtet man den obigen Beispielprozess, so ist schnell ersichtlich, dass das Einlesen der Quelldateien voneinander unabhängig passieren kann. Abhängig davon, wie viele Berechnungen im Rahmen des Einleseprozesses ausgeführt werden und wie hoch die CPU Last hierdurch ist, kann Parallelisierung eine deutliche Performancesteigerung bringen.

Zum Starten paralleler Prozesse bieten sich bei SAS grundsätzlich mehrere Möglichkeiten. Im Standard-Funktionsumfang stehen xCommands zur Verfügung, über die aus einem DataStep heraus weitere Betriebssystemprozesse gestartet werden können. Diese Prozesse wiederum können problemlos weitere SAS Sessions starten und somit parallele SAS Prozesse steuern.

Lizensiert man das Zusatzprodukt SAS/Connect, so besteht die Möglichkeit, hierüber parallele Prozesse zu starten. Dabei besteht sowohl die Möglichkeit, die parallelen Prozesse auf demselben Rechner zu starten oder auch auf einem Remote-System, welches über eine Netzwerkverbindung erreichbar ist.

5.2 xCommands / SYSTASK

Über xCommands können Betriebssystemkommandos aus einem DataStep ausgeführt werden. Die Systemoption XWAIT bzw. NOXWAIT steuert dabei, ob diese Betriebssystemprozesse synchron oder asynchron ausgeführt werden. Nachteil des einfachen xCommands ist, dass ReturnCodes der Prozesse nicht individuell abgefragt werden können. Auch ist eine Steuerung von Abhängigkeiten nicht ohne weiteres möglich.

Eine Alternative stellt der SYSTASK dar, der analog zum einfachen xCommand verwendet werden kann. Im Gegensatz um xCommand können jedoch weitere Optionen an den Aufruf übergeben werden. Diese legen beispielsweise fest, ob in einer Makrovariablen der ReturnCode des Tasks gespeichert werden soll.

```
data _null_;
  systask command "sas.exe -sysin ""d:\ksfe\systask.sas""
                           -log ""d:\ksfe\systaks_log1.txt"""
          nowait taskname=T1 status=T1S;
  systask command "sas.exe -sysin ""d:\ksfe\systask.sas""
```

```
                          -log ""d:\ksfe\systaks_log2.txt"""
        nowait taskname=T2 status=T2S;
  systask command "sas.exe -sysin ""d:\ksfe\systask.sas""
                          -log ""d:\ksfe\systaks_log3.txt"""
        nowait taskname=T3 status=T3S;
  waitfor _all_ T1 T2 T3;
  putlog "Task 1: RC = &T1S.";
  putlog "Task 2: RC = &T2S.";
  putlog "Task 3: RC = &T3S.";
run;
```

Der abgebildete DataStep startet drei parallele SAS Prozesse, die jeweils das SAS Source-Member systask.sas aufrufen. Jeder Prozess wird asynchron (nowait) gestartet. Jedem Prozess wird ein individueller Name zugewiesen. Über die Anweisung waitfor wird auf die Beendigung aller Teilprozesse gewartet. Die Ausführung des DataStep wird erst fortgesetzt, wenn alle drei Prozesse abgeschlossen sind. Zum Abschluss werden die ReturnCodes der einzelnen Prozesse im SAS Log ausgegeben.

Vergleicht man die sequentielle Ausführung der drei Teilprogramme mit der Ausführung über drei parallele Systask, so lässt sich schnell der Performancegewinn erkennen.

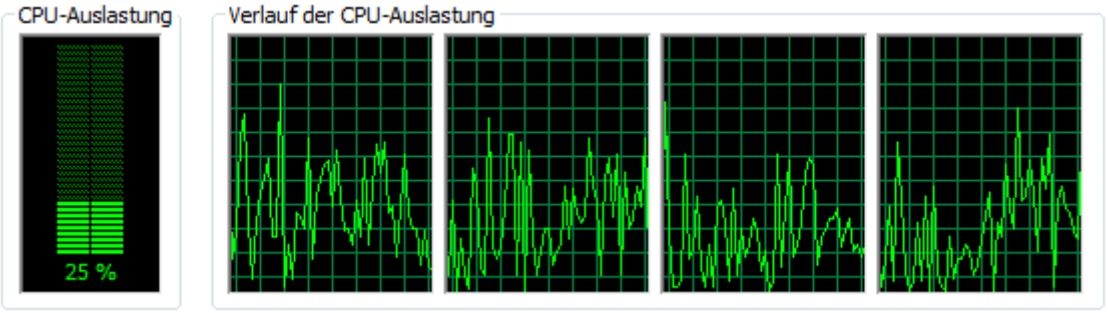

Abbildung 2: CPU-Auslastung bei sequentieller Ausführung

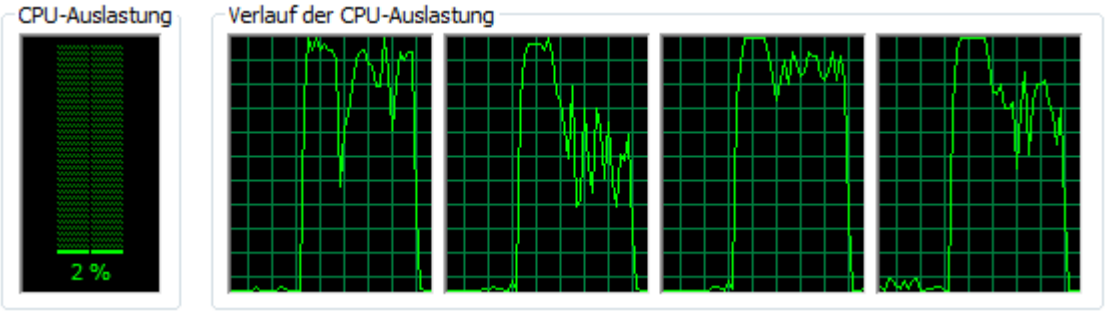

Abbildung 3: CPU-Auslastung bei paralleler Ausführung

Abbildung 2 zeigt die sequentielle Ausführung des Programms. Die reale Laufzeit beträgt 3x 28 Sekunden wobei die CPU Auslastung während dieser Zeit bei durchschnittlichen 25 % liegt. Hier erkennt man, dass von den 4 vorhandenen CPU Kernen im Schnitt nur ein Kern genutzt wurde. Abbildung 3 zeigt die Ausführung derselben drei

Programme in drei parallelen Prozessen. Die gesamte Ausführungszeit beträgt nur 28 Sekunden. Es wurden somit 2/3 der Ausführungszeit eingespart. Die Diagramme zur CPU-Auslastung zeigen, dass während der Ausführung die Auslastung des Rechners bei ca. 80% lag. Dies erklärt sich aus den drei parallelen Prozessen, die jeweils einen CPU Kern komplett nutzten. Der steuernde, übergeordnete Prozess nutzte die restlichen 5 % der CPU.

Auch mit dem SYSTASK sind die Steuerungsmöglichkeiten der gestarteten SAS Sessions relativ gering. Die Steuerung beschränkt sich auf die Abfrage des ReturnCodes und auf die Überwachung der Prozessbeendigung. Sollen Informationen aus dem übergeordneten Prozess in die Child-Prozesse übergeben werden, so sind diese in die SAS Sourcefiles zu generieren, die ausgeführt werden.

5.3 SAS/Connect

SAS/Connect bietet einen deutlich erweiterten Funktionsumfang. Prinzipiell wird mittels SAS/Connect eine Verbindung zu einer neuen SAS Session hergestellt. Ob diese Session auf dem lokalen Rechner oder auf einem entfernten Rechner gestartet wird, ist im Prinzip nicht relevant. Aus der steuernden SAS Session kann mit dem Sprachkonstrukt RSUBMIT und ENDRSUBMIT SAS Code in der Remote Session ausgeführt werden.

```
options sascmd="!sascmd";

signon task1;
signon task2;
signon task3;

rsubmit task1 wait=no log="d:\ksfe\task1.log"
                    output="d:\ksfe\task1.lst";
  data _null_;
    …;
    …;
  run;
endrsubmit;

rsubmit task2 wait=no log="d:\ksfe\task2.log”
                    output="d:\ksfe\task2.lst";
  data _null_;
    …;
    …;
  run;
endrsubmit;
```

```
rsubmit task3 wait=no log="d:\ksfe\task3.log"
                    output="d:\ksfe\task3.lst";
  data _null_;
    …;
    …;
  run;
endrsubmit;

waitfor _all_ task1 task2 task3;

signoff task1;
signoff task2;
signoff task3;
```

Das Ergebnis der Ausführung ist identisch zur parallelen Ausführung mittels SYS-TASK. Betrachtet man bei gleicher Programmausführung auch hier die CPU-Auslastung, so ist festzustellen, dass das Programm innerhalb von 28 Sekunden abgearbeitet werden konnte. Die CPU-Auslastung lag wieder im Schnitt bei 80 %, was auf die drei parallelen Prozesse mit voller CPU-Auslastung und den steuernden Prozess mit minimaler CPU-Auslastung zurückzuführen ist. Die Auslastungsdaten sind der folgenden Abbildung zu entnehmen.

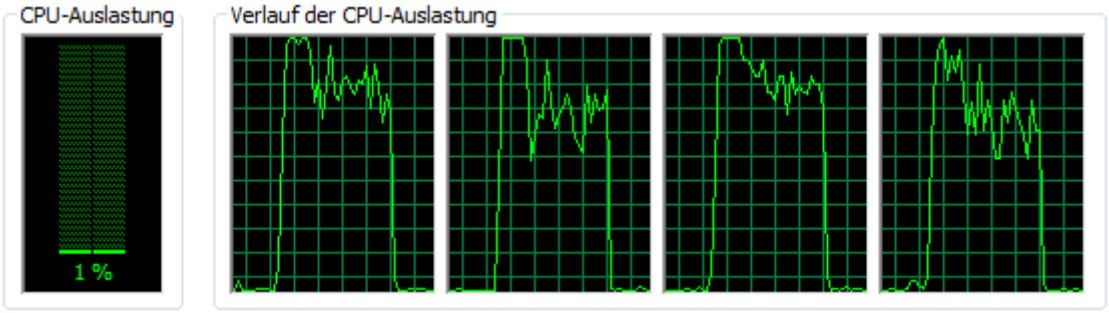

Abbildung 4: CPU-Auslastung bei paralleler Ausführung mit SAS/Connect

Um Informationen aus der steuernden Session in die Remote Sessions zu übertragen kann die Funktion %SYSLPUT genutzt werden.

```
%SYSLPUT REMOTEVAR=WERT;
```

Wird nach dem Signon ein %SYSLPUT ausgeführt kann innerhalb des RSUBMIT auf diese Variable zugegriffen werden. Auf diese Weise ist es einfach möglich, den Programmablauf oder auch Daten aus der steuernden Session zu übertragen. Eine Möglichkeit wäre beispielsweise die Übertragung des Pfades der Work-Bibliothek an die Client-Session, damit aus dieser Session auf Tabellen der steuernden Session zurückgegriffen werden kann.

Mit Hilfe der Funktion %SYSRPUT können auf dieselbe Weise Informationen aus den Child Sessions an die steuernde Session zurückgegeben werden.

```
rsubmit task1 wait=no sysrputsync=yes;
   %sysrput pathtask1=%sysfunc(pathname(work));
endrsubmit;
```

Im Beispiel wird der Pfad der Work Bibliothek der Client Session an die steuernde Session zurückübermittelt.

Seit SAS Version 9 besteht mit SAS Connect weiterhin die Möglichkeit, Daten zwischen SAS/Connect Sessions über die PIPE-Engine zu übertragen. Auf diese Weise wird der Festplattenzugriff komplett unterbunden und die Daten werden ausschließlich im Hauptspeicher übertragen.[3]

```
options sascmd="!sascmd";

signon p1;
rsubmit p1  wait=no;
  libname outLib sasesock ":8080";
  data outLib.Intermediate;
    set sashelp.class;
    do i=1 to 5; put 'Writing row ' i; output; end;
  run;
endrsubmit;

signon p2;
rsubmit p2  wait=no;
  libname inLib sasesock ":8080";
  data work.Final;
    set inLib.Intermediate;
    do j=1 to 5; put 'Adding data ' j; n2 = j*2; output; end;
  run;
endrsubmit;

waitfor _all_ p1 p2;

signoff p1;
signoff p2;
```

In diesem Beispiel wird die Tabelle SASHELP.CLASS über den Port 8080 vom Prozess P1 in den Prozess P2 übertragen. Dabei werden die Datensätze im ersten Prozess jeweils um 5 Sätze vervielfältigt. Der zweite Prozess greift die Datensätze auf, vervielfältigt die

[3] Vgl. http://support.sas.com/rnd/scalability/tricks/connect.html#multisign

Sätze erneut um den Faktor 5 und speichert die Daten in einer Tabelle. Dieses Vorgehen kann genutzt werden, um eine Tabelle mittels DataStep auf mehrere Prozesse aufzuteilen, um dort CPU intensive Berechnungen parallel durchzuführen.

Zu erwähnen ist an dieser Stelle, dass die Methode zum Piping von Daten zwischen Prozessen nicht auf lokale Prozesse beschränkt ist. Es ist problemlos möglich, über eine IP-Adresse Daten an einen entfernten Prozess zu übertragen um sie anschließend auf demselben Weg zurück zu übertragen.

6 Fazit

Die dargestellten Methoden zeigen, dass auch mit dem klassischen SAS Base Optimierungen genutzt werden können, um die Ausführungszeit von Programmen zu verkürzen.

Moderne Rechner bieten mit mehreren Rechenkernen die Möglichkeit, parallele Prozesse effizient abzuarbeiten. Durch die Nutzung der neuen SPD-Engine besteht auch bei I/O-intensiven Vorgängen die Möglichkeit, vom Multithreading zu profitieren und so mehr Performance aus dem Speichersubsystem zu erhalten.

Um die Optimierungsmöglichkeiten zu erkennen ist eine individuelle Analyse des Programmablaufs notwendig. Jede Optimierung muss auf den jeweiligen Anwendungsfall zugeschnitten sein.

Literatur

[1] SAS Dokumentation zur SPD-Engine:
http://support.sas.com/rnd/scalability/spde/index.html

[2] SAS Dokumentation zu SAS/Connect:
http://support.sas.com/rnd/scalability/connect/index.html.

[3] SAS Dokumentation zum SYSTAKS
http://support.sas.com/documentation/cdl/en/hostwin/67279/HTML/default/viewer.htm#p09xs5cudl2lfin17t5aqvpcxkuz.htm