Möglichkeiten der SAS Software für die Analyse großer Datentabellen

Hans-Peter Altenburg
Deutsches Krebsforschungszentrum
Klinische Epidemiologie / C0500
Im Neuenheimer Feld 280
D-69120 Heidelberg
E-mail: hp.altenburg@dkfz.de

Übersicht

- Beispiele
- Optimierung der System Performance
  - I/O Optimierung
  - Optimierung der Speicherausnutzung
  - Optimierung der CPU-Performance
- Sortieren
  - Die Prozedur SORT und Alternativen

Beispiel 1

EPIC Studie

Multizentrische Kohortenstudie

über den Zusammenhang

Ernährungsverhalten ↔ Krebsentstehung

Lungenkrebs-AG:

n=417.717 Probanden

mit 600 Variablen

Beispiel 1

PC: 1 Ghz, 128MB RAM

```
1    %LET dset=lung.lyon061e  ;
2
3    PROC FREQ DATA=&dset  ;
4    TABLES
5           sex*caselung / nopercent
6            ;
7    RUN ;

NOTE: There were 417717 observations read from the data set
   LUNG.LYON061E.
NOTE: PROCEDURE FREQ used:
      real time           1:19.31
      cpu time            8.50 seconds
```

Beispiel 1

PC: 1 Ghz, 128MB RAM

```
8
9     PROC FREQ DATA=&dset (KEEP=sex caselung) ;
10    TABLES
11            sex*caselung / nopercent
12             ;
13    RUN ;

NOTE: There were 417717 observations read from the data set
   LUNG.LYON061E.
NOTE: PROCEDURE FREQ used:
      real time           1:18.67
      cpu time            9.40 seconds
```

Beispiel 1

PC: 1 Ghz, 128MB RAM

```
14
15   %MACRO tabs(row,col) ;
16   PROC FREQ DATA=&dset (KEEP=&row &col) ;
17   TABLES
18          (&row)*(&col) / NOPERCENT   ;
19   RUN ;
20   %MEND ;
21
22   %tabs(sex,caselung)

NOTE: There were 417717 observations read from the data set
   LUNG.LYON061E.
NOTE: PROCEDURE FREQ used:
      real time           1:18.48
      cpu time            9.18 seconds
```

Beispiel 2

Cox-Proportional Hazard-Modell:

- Prozedur PHREG
- gleiche Datenmenge
- ohne KEEP -Statement:

  Zeitdauer: `real time` `> 2 h (Abbruch)`

- mit KEEP-Statement:

  Zeitdauer: `real time` `~ 20 min`

Optimierung der System Performance

I/O Optimierung

Optimierung der System Performance

I/O <u>Optimierung</u>:

- wichtigster Faktor

  da DA-Prozess meist viele Zyklen mit Lesen / Schreiben und Manipulation von Datentabellen beinhaltet

- Performance wird deutlich verbessert, wenn die Zahl der Plattenzugriffe verringert werden kann

$\Rightarrow$

- SAS Programm so konzipieren, daß möglichst wenige (besser nur die erforderlichen) Variablen und Beobachtungen verwendet werden.

I/O Optimierung

- Lese nur die benötigten Felder:

```
DATA neu ;
INFILE 'D:\...\...\dsetraw.txt' ;
INPUT ID $ 1-10 G $ 11 @110 x1 10.0 x2 12.2 ;
RUN ;
```

- Speichere Daten in eine permanente SAS-Datei:

```
LIBNAME  st_0102 'G:\...\...\daten' ;
DATA st_0102.neu ;
INFILE 'D:\...\...\dsetraw.txt' ;
INPUT ID $ 1-10 G $ 11 @110 x1 10.0 x2 12.2
  ;
RUN ;
```

I/O Optimierung

- KEEP oder DROP-Statement oder Option
- WHERE-Ausdruck
- OBS= / FIRSTOBS=
- LENGTH

I/O Optimierung

Reduktion der Variablenanzahl:

- KEEP- bzw. DROP-
  - Statement im DATA-Step

```
DATA st_0102.new ; SET old ;
KEEP x ;
  sum_x + x ;
  . . .
RUN ;
```

weniger effizient als

```
DATA st_0102.new ; SET old (KEEP=x );
  sum_x + x ;
  . . .
RUN ;
```

I/O Optimierung

Reduktion der Variablenanzahl:

- KEEP- bzw. DROP-
  - Statement im DATA-Step

```
DATA st_0102.new ; SET old (KEEP=x )
                                END=fin ;
sum_x + x ;      nn+1;
IF fin THEN mean_x=sum_x/nn ;
DROP sum_x nn ;
RUN ;
```

I/O Optimierung

Reduktion der Variablenanzahl:

- KEEP- bzw. DROP-
  - als Data-Set-Option

```
DATA new ; SET old (KEEP=x ) END=fin ;
sum_x + x ; nn+1;
IF fin THEN mean_x=sum_x/nn ;
DROP sum_x nn ;
RUN ;
PROC PRINT DATA=new (KEEP=mean_x ) ;
RUN ;
```

I/O Optimierung

OUTPUT:

Obs	mean_x
1	.
2	.
3	.
4	.
5	.
6	.
7	.
8	.
9	.
10	.
11	.
12	.
13	.
14	.
15	.
16	.
17	.
18	.
19	.
20	0.45524

I/O Optimierung

Reduktion der Anzahl von Beobachtungen:

- WHERE-Ausdruck:
  - bilden einer Untermenge der Datei
  - Verwendung im DATA-Step oder als Datei-Option
  - Zeitpunkt der Ausführung:

    vor dem Einlesen der Daten in den „program data vector“ (PDV), $\Rightarrow$ Vorteile bei Dateien mit vielen Variablen!
  - Erlaubt weitere Operatoren wie z.B.

    LIKE, CONTAINS oder

    =* (phonetische Äquivalente)

I/O Optimierung

Reduktion der Anzahl von Beobachtungen:

- WHERE-Ausdruck:
  - Nachteile:
    - kann nur einmal am Anfang verwendet werden
    - Keine temporären Variablen
      (d.h. nicht _N_ / FIRST. / LAST. / IN etc.)

```
PROC PRINT DATA=new (WHERE=(sex=1));
PROC PRINT DATA=new; WHERE  sex=1 ;

PROC PRINT DATA=new
                    (WHERE=(30<=age<40));
```

I/O Optimierung

Reduktion der Anzahl von Beobachtungen:

bilden einer Untermenge der Datei (IF)

```
DATA new ; SET old  END=fin ;
IF sex=1 THEN DO; sum_x + x ; nn+1 ; END ;
IF fin   THEN mean_x=sum_x/nn ;
DROP sum_x nn ;
RUN ;
```

Mit WHERE-Bedingung:

```
DATA new ; SET old  END=fin ;
WHERE sex=1 ;
sum_x + x ; nn+1 ;
IF fin THEN mean_x=sum_x/nn ;
DROP sum_x nn ;
RUN ;
```

I/O Optimierung

Reduktion der Anzahl von Beobachtungen:

WHERE-Bedingung als DATA-Set-Option:

```
DATA new ; SET old (WHERE=(sex=1))
                    END=fin ;
sum_x + x ; nn+1 ;
IF fin THEN mean_x=sum_x/nn ;
DROP sum_x nn ;
RUN ;
```

I/O Optimierung

Beispiel:

```
DATA new ; SET old (KEEP=x ) END=fin ;
  sum_x + x ; nn+1;
  IF fin THEN mean_x=sum_x/nn ;
  DROP sum_x nn ;
RUN ;
PROC PRINT DATA=new (KEEP=mean_x
                     WHERE=(mean_x >.));
RUN ;
```

OUTPUT:

Obs	mean_x
20	0.45901

I/O Optimierung

Bilde Teil- / Untermengen in nur einem Data-Step:

```
DATA new_M ; SET old (KEEP= ...) ;
IF gender=1 ;
RUN ;
DATA new_F ; SET old (KEEP= ...) ;
IF gender=2 ;
RUN ;
```

Effizienter:

```
DATA new_M new_F ; SET old (KEEP= ...) ;
SELECT (gender) ;
WHEN (1) OUTPUT new_M ;
WHEN (2) OUTPUT new_F :
OTHERWISE ;
RUN ;
```

I/O Optimierung

Nutze „temporäre“ Arrays anstatt dem Erzeugen und Löschen von Variablen:

```
DATA index ; INPUT ... ;

ARRAY testvar {4} t1-t4 ;
ARRAY cutoffs {4} c1-c4 (20 40 60 80) ;
DO i=1 TO 4 ;
IF testvar{i}>=cutoffs{i} THEN ... ;
END ;
DROP c1-c4 ;
RUN ;
```

Effektiver:

```
DATA index ; INPUT ... ;

ARRAY testvar {4} t1-t4 ;
ARRAY cutoffs {4} _TEMPORARY_ c1-c4 (20 40 60 80) ;
DO i=1 TO 4 ;
IF testvar{i}>=cutoffs{i} THEN ... ;
END ;
RUN ;
```

I/O Optimierung

Reduktion der Anzahl von Beobachtungen:

Data-Set-Optionen

OBS = (Anzahl) und

FIRSTOBS = (Startwert)

Anwendung:

Verwendung einer temporären Datei, die nur die erforderlichen Beobachtungen enthält, kann die Anzahl der I/O-Operationen reduzieren.

I/O Optimierung

LENGTH-Statement

- reduziert ebenfalls die Beobachtungsgröße
- reduziert den erforderlichen Speicherbedarf pro Variable
- reduziert die Anzahl I/O Operationen in einem Datenverarbeitungsprozeß
- Bei RISC-Prozessoren dagegen können „kurze“ numerische Daten zu einem größeren Aufwand führen.

I/O Optimierung

Nutze die Möglichkeit übersetzte Programme abzuspeichern und später auszuführen:

```
LIBNAME lib 'D:\...\...' ;
DATA neu1 ;
SET lib.old1 ;

. . . weitere Rechenschritte

RUN ;
DATA neu2 ;
SET lib.old2 ;

. . . weitere Rechenschritte

RUN ;
```

I/O Optimierung

übersetztes Programm abspeichern:

```
LIBNAME lib 'D:\...\...' ;
LIBNAME plib 'D:\...\...' ;

DATA neu1 ;
SET lib.old1 ;
. . . ;
RUN pgm=plib.prog1 ;
```

und später ausführen:

```
DATA pgm=plib.prog1 ;
REDIRECT INPUT lib.old1=lib.old1 ;
REDIRECT OUTPUT neu1=neu1 ;
RUN ;

DATA pgm=plib.prog1 ;
REDIRECT INPUT lib.old1=lib.old2 ;
REDIRECT OUTPUT neu1=neu2 ;
RUN ;
```

Optimierung der Speichernutzung

Optimierung der Speichernutzung

- Lies nur die Felder, die benötigt werden
- Speichere nur die Variablen, die gebraucht werden
- Verkürze die Datenwerte durch Verwendung von Formaten
- Verwende Character anstelle numerischer Variablen
- Speichere Zahlen als 1-Byte Character-Werte

Optimierung der Speichernutzung

Verwende _NULL_ Data Sets:

```
DATA new ; SET old (KEEP=... WHERE=(...) ) END=fin ;
   FILE PRINT ;
   sum_x + x ;  nn+1 ;
IF fin THEN DO ;
            mean_x=sum_x/nn ;
            PUT 'Mean: ' mean_x 12.2 ;
            END ;
DROP sum_x nn ;
RUN ;

DATA _NULL_ ; SET old (KEEP=... WHERE=(...) ) END=fin ;
   FILE PRINT ;
   sum_x + x ;  nn+1 ;
IF fin THEN DO ;
            mean_x=sum_x/nn ;
            PUT 'Mean: ' mean_x 12.2 ;
            END ;
RUN ;
```

Optimierung der CPU-Performance

Optimierung der CPU-Performance

- Lies nur die Felder, die benötigt werden
- Speichere die Daten in eine SAS-Datei
- Verwende nur die Daten / Variablen, die benötigt werden
- Verwende Informate für Datentransformationen
- Logische Operatoren: Besser **IN** als **OR** Operator
- DO-Schleifen: Verwende die Schleife mit den geringsten Iterationen am weitesten außen
- Verwende SAS-Funktionen anstelle „eigener“ Programmschritte (z.B. mean_xyz=MEAN(OF x y z) ;)
- Verwende Macros für immer gleiche Programmschritte
- Verwende _NULL_ Data Sets

Optimierung der CPU-Performance

Benutze ein CLASS-Statement in Prozeduren:

```
PROC SORT  DATA=new ; BY group ; RUN ;
PROC MEANS DATA=new ; BY group ; RUN ;

PROC MEANS DATA=new ; CLASS group ;
RUN ;
```

Vorteil:

- CLASS-Statement hat keinen permanenten Effekt auf die Daten.
- Braucht zwar mehr Zeit als Verwendung von BY, aber weniger als ein separater Sortierschritt!

Optimierung der System Performance

Sortieren
Die Prozedur SORT und Alternativen

Optimierung der CPU-Performance

Plane das Sortieren von Data Sets:

```
PROC SORT  DATA=new ; BY group ; RUN ;
PROC MEANS DATA=new ; BY group ;
VAR a b c y1 y4 y10 ;
RUN ;

PROC SORT  DATA=new ; BY sex   ; RUN ;
PROC CHART DATA=new ; BY sex   ;
HBAR x1 ;
RUN ;

PROC SORT  DATA=new ; BY group ; RUN ;
PROC PRINT DATA=new ; BY group ;
VAR y1-y10 ;
RUN ;

PROC SORT  DATA=new ; BY sex agegr ; RUN ;
PROC MEANS DATA=new ; BY sex agegr ;
VAR z1-z5 ;
RUN ;
```

Optimierung der CPU-Performance

Besser:

```
PROC SORT  DATA=new ; BY group ; RUN ;
PROC MEANS DATA=new ; BY group ;
VAR a b c y1 y4 y10 ;
RUN ;
PROC PRINT DATA=new ; BY group ;
VAR y1-y10 ;
RUN ;

PROC SORT  DATA=new ; BY sex agegr ; RUN ;
PROC CHART DATA=new ; BY sex   ;
HBAR x1 ;
RUN ;

PROC MEANS DATA=new ; BY sex agegr ;
VAR z1-z5 ;
RUN ;
```

Optimierung der CPU-Performance

Prozedur SORT:

```
PROC SORT DATA=large;
BY group ;
RUN ;
```

Besser:

```
PROC SORT DATA=large NOEQUALS ;
BY group ;
RUN ;
```

Die Reihenfolge weiterer Variablen bleibt dabei nicht erhalten.

Prozedur SORT

Optimierung der Speichernutzung:

System (Prozedur) Option

SORTSIZE=*n*|*n*K|*n*M|*n*G|MIN|MAX|*hex*

- *n*|*n*K|*n*M|*n*G |*hex* spezifiziert die Größe in Byte, Kilobyte, Megabyte, Gigabyte oder als Hexadezimalzahl
- Spezifiziert die Speichergröße, die für die Prozedur SORT zur Verfügung steht
- Wert sollte kleiner als die physikalisch verfügbare Größe gewählt werden
- Kann die Sortierperformanz verbessern helfen

Prozedur SORT

Alternativen für große Datenmengen:

- Host Sortierroutine kann schneller sein:

  ```
  OPTIONS SORTPGM=host ;
  ```

  Wird allerdings ignoriert, wenn die Option TAGSORT in der Prozedur SORT verwendet wird.

- Prozedur SYNCSORT® lizensieren

  schneller als Standard-SORT

- TAGSORT-Option:

Prozedur SORT: Alternativen für große Datenmengen

TAGSORT-Option:

```
PROC SORT TAGSORT DATA=dat.m1 ;
BY v1 v2 v3 v4 v5 v6 v7 v8 v9 v10
   v11 v12 ;
RUN ;
```

- **Wirkung**: Es werden nur die Sortierschlüssel und Beobachtungsnummern (=„tags“) in einem temorären SAS-File abgelegt. Am Ende werden mit Hilfe dieser tags die Datensätze aus der Orginaldatei in die sortierte Reihenfolge gebracht.

TAGSORT-Option:

- **Vorteile**, wenn die totale Länge der Sortierkeys klein ist verglichen mit der Satzlänge.
- Temporärer Speicherplatz wird dramatisch reduziert.

Prozedur SQL:

```
PROC SQL ;
CREATE TABLE dat.sqlt1 AS
SELECT * FROM dat.m1
ORDER v1, v2, v3, v4, v5, v6, v7, v8,
  v9, v10, v11, v12 ;
QUIT ;
```

Erzeugt eine neue sortierte SAS-Datei.

Prozedur SORT: Alternativen für große Datenmengen

Prozedur DATASETS:

```
PROC DATASETS LIBRARY=dat ;
MODIFY m1 ;
INDEX CREATE idxlist=(v1 v2 v3 v4 v5 v6 v7
  v8 v9 v10 v11 v12 ) ;
RUN ;
```

Erzeugt einen Index (Name `idxlist`), der so genutzt werden kann, als ob die Datei vollständig sortiert ist.

- Indizieren ist schneller als Sortieren!
- **Nachteil**: u.U. wurde ein großer Indexfile erzeugt (30%)!

Fragen, Anregungen oder Kommentare?