Paper CC-12

Application Dispatcher: Some Tips and Tweaks

Carol A. Martell, Highway Safety Research Center, Chapel Hill, NC

ABSTRACT

In the process of converting a ColdFusion/MySQL dynamic website to one that uses SAS/IntrNet® Application Dispatcher to query data, we developed a few workarounds to cope with miscellaneous obstacles. The paper presents techniques enriching the dictionary tables and tricks for passing and parsing multiple values from a single form field.

This paper is for the intermediate to advanced SAS® user having familiarity with Application Dispatcher.

INTRODUCTION

Our campus statistical computing environment has moved from a large UNIX server to a cluster environment having a variety of flavors of UNIX. The cluster environment is great for holding off downtime, but presents some new challenges for the SAS programmer, encapsulated in the following SAS log message:

```
NOTE: Data file P.BIKECOLS.DATA is in a format native to another host or the file
encoding does not match the session encoding. Cross Environment Data Access will be
used, which may require additional CPU resources and reduce performance.
```

This message means that the SAS tables we are using were created under a different operating system than the one wherein we are running our current job. When appearing in the log for a program run from SAS/IntrNet Application Dispatcher, it usually translates into a much longer wait for the results of a dynamic query to appear. It can, unfortunately, mean that your results won't appear because the program does not generate output. If we cannot, for one reason or another, create our files in a format native to that of the Application Dispatcher, we need to find some other options to help us. We examine some workarounds that help alleviate our "cluster woes."

After dealing with the meta-issues of our dynamic website, we demonstrate our foray into finding a solution that sends multiple pieces of information from a single form field, or name/value pair. Specifically, we wish to use different formats with a single variable, and to override a default ORDER= parameter for our generated tables.

FOREIGN DATA

Our data reside in a library created in the UNIX cluster. There is a format catalog, and a table with variables assigned formats contained in that catalog. The dynamic website we are designing allows the user to select a variable. In our program, we would like to query the dictionary tables for information about the selected variable. Having received no output, we begin to debug by first querying DICTIONARY.TABLES, with good results, as seen in Figure 1:

```
3    +ods html file=_webout style=minimal;
NOTE: Writing HTML Body file: _WEBOUT
4    +proc sql;
5    +select memname from dictionary.tables where lowcase(libname)='p';
6    +quit;
NOTE: The PROCEDURE SQL printed page 1.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.16 seconds
      cpu time            0.17 seconds

7    +ods html close;
NOTE: %INCLUDE (level 1) ending.
NOTE: request has completed
```

Figure 1

```
3    +ods html file=_webout style=minimal;
NOTE: Writing HTML Body file: _WEBOUT
4    +proc sql;
5    +*select memname from dictionary.tables where lowcase(libname)='p'
6    +order by memname;
7    +select name from dictionary.columns where lowcase(libname)='p'
8    +and lowcase(memname)='pbcatbike'; quit;
NOTE: Data file P.BIKECOLS.DATA is in a format native to another host or t
      additional CPU resources and reduce performance.
NOTE: Data file P.BIKEDATA.DATA is in a format native to another host or t
      additional CPU resources and reduce performance.
NOTE: Data file P.FMTCNTL.DATA is in a format native to another host or th
      additional CPU resources and reduce performance.
NOTE: Data file P.PBCATBIKE.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.PBCATBIKE_MISSINGTIME0607.DATA is in a format native to
      may require additional CPU resources and reduce performance.
NOTE: Data file P.PBCATPED.DATA is in a format native to another host or t
      additional CPU resources and reduce performance.
NOTE: Data file P.PEDCOLS.DATA is in a format native to another host or th
      additional CPU resources and reduce performance.
NOTE: Data file P.PEDDATA.DATA is in a format native to another host or th
      additional CPU resources and reduce performance.
NOTE: Data file P.PEDDATAMAYBE.DATA is in a format native to another host
      additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.EUROSTAT.DATA is in a format native to another hos
      require additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.WEBBGT.DATA is in a format native to another host
      additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.WEBSALE.DATA is in a format native to another host
      additional CPU resources and reduce performance.
NOTE: The PROCEDURE SQL printed pages 1-2.


NOTE: PROCEDURE SQL used (Total process time):
      real time           28.49 seconds
      cpu time            5.09 seconds

9    +ods html close;
NOTE: %INCLUDE (level 1) ending.
NOTE: request has completed
```

Figure 2

We expand our query, asking for the members in our library as seen in Figure 2. Here we see our first 'foreign data' messages. Our CPU time has increased to 5 seconds. We try pulling the table into the WORK library first to see if that helps our CPU time. Since we have formats associated with our variables, we issue a FMTSEARCH option, as seen in Figure 3. This attempt fails to produce results, because SAS cannot use the format catalog prepared in a foreign operating system and we are not using the NOFMTERR option.

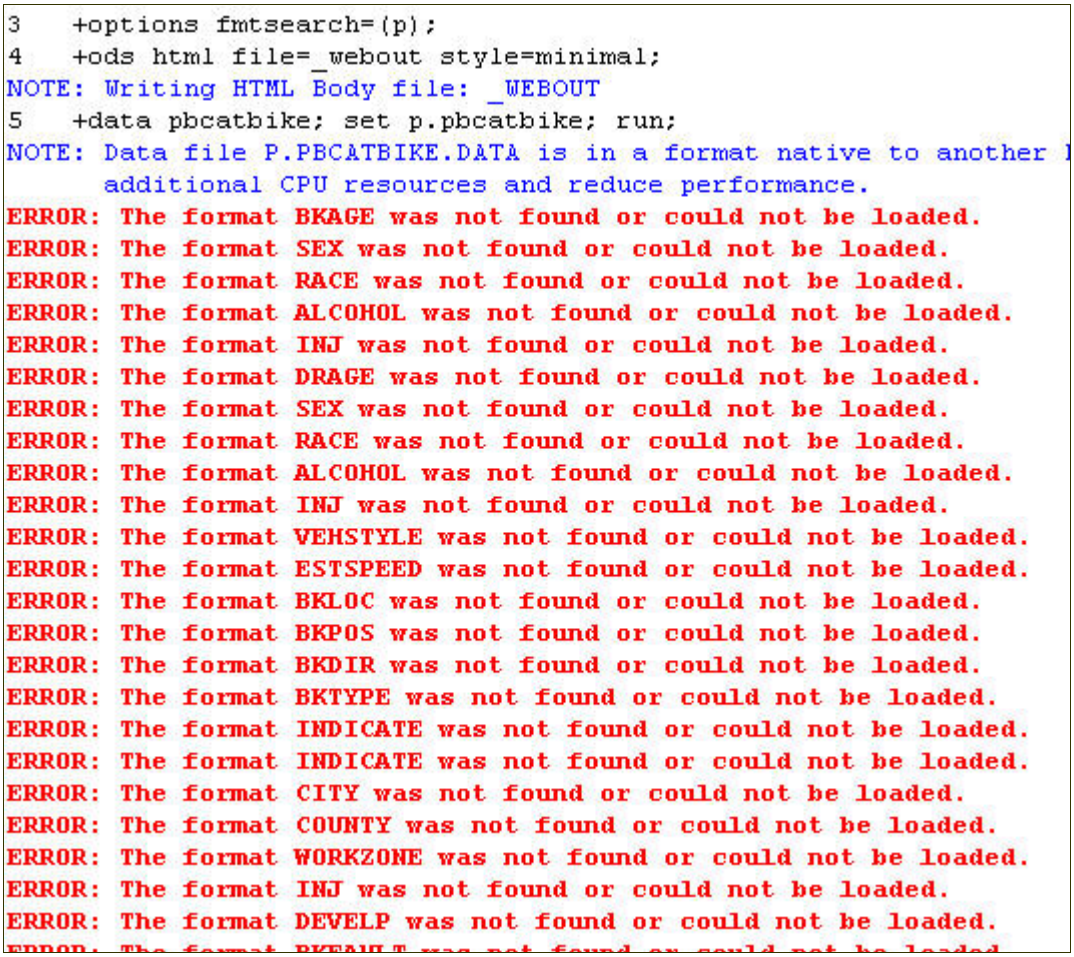

Figure 3

We try removing the formats from the variables, while pulling the table into WORK. Figure 4 shows that the job runs, with CPU totaling to 3.32 seconds, but we're still getting warning messages about all the members of the foreign library. In fact, we're getting warning messages about the SAMPDAT library, which we failed to notice at first.

We are, at least, happier with our CPU time, but we need to use our formats. We create a format control dataset in the P library and run a PROC FORMAT using the CNTLIN option to pull our user-defined formats into the WORK library. This log is shown in Figure 5. Our CPU time is now totaling 3.29 seconds, and we have our formats in place.

Those pesky warning messages are, however, still there. We need to bypass the dictionary tables completely if we want to get rid of them!

```
3    +ods html file=_webout style=minimal;
NOTE: Writing HTML Body file: _WEBOUT
4    +data pbcatbike;
5    +set p.pbcatbike;
NOTE: Data file P.PBCATBIKE.DATA is in a format native to another host o
      additional CPU resources and reduce performance.
6    +format _all_;
7    +run;
NOTE: There were 11842 observations read from the data set P.PBCATBIKE.
NOTE: The data set WORK.PBCATBIKE has 11842 observations and 52 variable
NOTE: DATA statement used (Total process time):
      real time           0.13 seconds
      cpu time            0.13 seconds

8    +proc sql;
9    +select name from dictionary.columns where lowcase(libname)='work'
10   +and lowcase(memname)='pbcatbike';
NOTE: Data file P.BIKECOLS.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.BIKEDATA.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.FMTCNTL.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.PBCATBIKE.DATA is in a format native to another host o
      additional CPU resources and reduce performance.
NOTE: Data file P.PBCATBIKE_MISSINGTIME0607.DATA is in a format native t
      may require additional CPU resources and reduce performance.
NOTE: Data file P.PBCATPED.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.PEDCOLS.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.PEDDATA.DATA is in a format native to another host or


      additional CPU resources and reduce performance.
NOTE: Data file P.PEDDATAMAYBE.DATA is in a format native to another hos
      additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.EUROSTAT.DATA is in a format native to another h
      require additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.WEBBGT.DATA is in a format native to another hos
      additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.WEBSALE.DATA is in a format native to another ho
      additional CPU resources and reduce performance.
11   +ods html close;
NOTE: %INCLUDE (level 1) ending.
NOTE: The PROCEDURE SQL printed pages 1-2.
NOTE: PROCEDURE SQL used (Total process time):
      real time           5.76 seconds
      cpu time            3.19 seconds
```

Figure 4

```
NOTE: There were 1486 observations read from the data set P.FMTCNTL.
NOTE: PROCEDURE FORMAT used (Total process time):
      real time           0.03 seconds
      cpu time            0.02 seconds

5   +data pbcatbike; set p.pbcatbike; run;
NOTE: Data file P.PBCATBIKE.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: There were 11842 observations read from the data set P.PBCATBIKE.
NOTE: The data set WORK.PBCATBIKE has 11842 observations and 52 variables
NOTE: DATA statement used (Total process time):
      real time           0.15 seconds
      cpu time            0.12 seconds

6   +proc sql;
7   +select name from dictionary.columns where lowcase(libname)='work'
8   +and lowcase(memname)='pbcatbike';
NOTE: Data file P.BIKECOLS.DATA is in a format native to another host or


      additional CPU resources and reduce performance.
NOTE: Data file P.BIKEDATA.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.FMTCNTL.DATA is in a format native to another host or t
      additional CPU resources and reduce performance.
NOTE: Data file P.PBCATBIKE.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.PBCATBIKE_MISSINGTIME0607.DATA is in a format native to
      may require additional CPU resources and reduce performance.
NOTE: Data file P.PBCATPED.DATA is in a format native to another host or
      additional CPU resources and reduce performance.
NOTE: Data file P.PEDCOLS.DATA is in a format native to another host or t
      additional CPU resources and reduce performance.
NOTE: Data file P.PEDDATA.DATA is in a format native to another host or t
      additional CPU resources and reduce performance.
NOTE: Data file P.PEDDATAMAYBE.DATA is in a format native to another host
      additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.EUROSTAT.DATA is in a format native to another ho
      require additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.WEBBGT.DATA is in a format native to another host
      additional CPU resources and reduce performance.
NOTE: Data file SAMPDAT.WEBSALE.DATA is in a format native to another hos
      additional CPU resources and reduce performance.
9   +ods html close;
NOTE: %INCLUDE (level 1) ending.
NOTE: The PROCEDURE SQL printed pages 1-2.
NOTE: PROCEDURE SQL used (Total process time):
      real time           5.82 seconds
      cpu time            3.15 seconds
```

Figure 5

```
3    +ods html file=_webout style=minimal;
NOTE: Writing HTML Body file: _WEBOUT
4    +proc sql;
5    +select name from p.bikecols;
NOTE: Data file P.BIKECOLS.DATA is in a format native to ano
      additional CPU resources and reduce performance.
6    +quit;
NOTE: The PROCEDURE SQL printed pages 1-2.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds

7    +ods html close;
NOTE: %INCLUDE (level 1) ending.
NOTE: request has completed
```

Figure 6

Our table structures are static, so obtaining the most up-to-date list of variables and their attributes from the dictionary tables is not necessary. Instead, we create a table, called BIKECOLS, in our P library that contains the information we are seeking from DICTIONARY.COLUMNS – specifically, the variable name, format and label. Figure 6 shows that when we query this table for the set of column names, our CPU time is only 0.01 seconds! We get a foreign table warning message, but only about the table in questions, rather than about every table in our P library and the SAMPDAT library. We could add some columns to our fake DICTIONARY.COLUMNS table, augmenting the information with useful tidbits like the preferred order to use in table output! The possibilities are freeing, rather than constricting.

TINKERING WITH FORM FIELD VALUES

For our project at hand, we are converting a dynamic website to use the SAS/IntrNet Application Dispatcher from another application server system. One of the most powerful tools in SAS is the format – the means to take a single variable value and present it any number of different ways by using different formats. To implement the same thing in the ColdFusion/MySQL version, a new variable had been created for each different representation. So, for example, if an age variable needed to be available in single year, 5-year groupings and 10-year groupings, three variables were created. For our dynamic website, we throw out all those extra iterations of a single variable and use formats instead. By associating formats with the variables in the table to be queried, most of the pretty display work is taken care of, as SAS will automatically use the formats in procedure output. A select list that allows the web user to choose a variable simply needs to pass the variable name from the web form to the SAS program. For those variables, such as age, that need format overrides, we incorporate the format name into the value for the selected variable.

In addition to the items in this variable select list, which would use default formats…:

```
<select name="rowvar">
<option value="sex">Driver Sex</option>
<option value="race">Driver Race</option>
<option value="city">Crash City</option>
<option value="county">Crash County</option>
<option value="region">Crash Region</option>
<option value="age">age</option>
</select>
```

we insert more age items that can be selected:

```
<option value="age,age5g.">5-year age group</option>
<option value="age,age10g.">10-year age group</option>
```

In our SAS program we can use the COUNTW function to determine how many parameters are included in the name/value pair, and therefore whether or not the default format should be overridden in the table. If so, the SCAN or %SCAN functions can extract the format name to use instead of the default format.

Some variables are best presented in alphabetical order using the FORMATTED value. Others, such as day of week, are better presented with ORDER=INTERNAL. It probably makes more sense to augment our fake DICTIONARY.COLUMNS table with a variable indicating the preferred display order, but for illustrative purposes, we will instead override the default ORDER= value using the form field.

We add a third parameter to our value, shown here with comma separators, providing name, format and order:

```
<option value="weekday,,internal">day of week</option>
```

There is a problem with using the above example. One would think that, not needing to override the format, putting the variable name followed by two commas and the overriding order would be appropriate. This syntax for the value could be used to supply a positional parameter string for a macro. All the other values in our web form, however, don't have three parameters, so the SAS code would need to be constructed accordingly. If the string 'weekday,,internal' were only fed to a macro when the count of parameters was 3, all would be well. However, COUNTW("weekday,,internal",",") would return the value 2, because the two comma delimiters in a row would count as a single delimiter. SCAN behaves in the same fashion. Alternatively, we could count the occurrences of the comma and use macro conditional logic to call different macros accordingly. Another possibility would be to use keyword parameter syntax to accommodate a varying number of values to pass.

For demonstration purposes, our workaround is to supply a value we will subsequently ignore:

```
<option value="weekday,default,internal">day of week</option>
```

We write our SAS code to ignore the value 'default' as an overriding format. The functions COUNTW, %SCAN and SCAN will count our parameters properly, and we can construct our output tables for the dynamic website successfully.

CONCLUSION

We have demonstrated some workarounds for dealing with mixed operating systems in a computing environment that is not under our control. It would be easiest to convert our data to the flavor of UNIX used by the Application Dispatcher. We would have no guarantees that the Application Dispatcher would not be moved to a different flavor of UNIX in the future, though, so our workaround is conservative and functional.

We have also shown one way, and suggested several ways, to pass multiple pieces of information using a web form field that normally would supply a single value.

CONTACT INFORMATION

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

Carol Martell
Highway Safety Research Center
730 Martin Luther King Jr Blvd
Chapel Hill, NC 27514
caol_martell@unc.edu