Paper 8080-2016

The HPSUMMARY® Procedure: An Old Friend's Younger (and Brawnier) Cousin

Anh P. Kellermann, Jeffrey D. Kromrey
University of South Florida, Tampa, FL

ABSTRACT

The HPSUMMARY procedure provides data summarization tools to compute basic descriptive statistics for variables in a SAS dataset. It is a high-performance version of the SUMMARY procedure in Base SAS. Although the PROC SUMMARY is popular with data analysts, the PROC HPSUMMARY is still "a new kid on the block." This paper provides an introduction to PROC HPSUMMARY by comparing it with its well-known counterpart, PROC SUMMARY. General syntax differences as well as performance in terms of processing time and memory utilization of the two procedures are examined. Simulated data of different sizes were used to observe the performance of the two procedures. Experiment results indicate that there was no clear difference in real time between the PROC SUMMARY and its high performance counterpart. The HP version utilized more memory but provided better memory management in a limited-memory environment than the legacy version did. When the number of available cores increases, the HPSUMMARY utilized all cores available to it, reducing real time substantially.

Keywords: HPSUMMARY, SUMMARY, high-performance analytics procedures

INTRODUCTION

PROC SUMMARY in Base SAS is one of the most popular resources in the toolbox of most SAS data analysts. It is the identical twin of the popular PROC MEANS in all aspects except some difference in the VAR statement and default output display. With over three dozen of available statistic-keywords provided in the SUMMARY procedure, users can conveniently obtain statistics of a dataset for observation or for saving in a temporary or permanent dataset for subsequent analysis. The HPSUMMARY procedure is a high-performance (HP) version of the SUMMARY procedure. In contrast to PROC SUMMARY which is provided in Base SAS, the HP version requires one or more of the traditional SAS products like SAS/STAT. Along with other SAS high performance analytics procedures, the HPSUMMARY procedure was developed to meet the growing needs for efficient tools to work on large data sets. Like other SAS HP procedures, the HPsummary procedure is designed to utilize the existing capacity of the machine it runs on; it will exploit all cores available to it. The HPsummary procedure runs on both high performance cluster systems as well as on a single-user machines.

The HPSUMMARY procedure provides functionality similar to that of the SUMMARY procedure in Base SAS, and its syntax, options, and underlying concepts are also similar (Base SAS 9.4 procedure guide); however, there are some important differences between the two. To assist users in getting acquainted with this evolutionary procedure, this paper provides an introduction to PROC HPSUMMARY by comparing it with its more familiar counterpart, PROC SUMMARY. Based on SAS documentation and the authors' empirical experiments with the procedures, the comparison will focus on the differences in options/syntax and performance in terms of execution time and memory usage between the two procedures.

OPTION/SYNTAX DIFFERENCES

As mentioned above, the HPSUMMARY procedure provides functionality similar to that of the SUMMARY procedure. In general, the HPSUMMARY procedure calculates descriptive statistics, calculates and estimates quantiles including the median, calculates confidence limits for the mean, identifies extreme values, and performs a *t* test. Like the SUMMARY procedure, the HPSUMMARY procedure computes these descriptive statistics for variables across all observations or within groups of observations using a

CLASS statement as the SUMMARY procedure does. Both procedures share the same options of statistic-keywords for computing these statistics. However, the HPSUMMARY procedure is designed to be used in both single machine and in a distributed system where computation is done on different machines; and this difference in purpose gives rise to their syntax differences. Figure 1 shows the general syntax for the HPSUMMARY and SUMMARY procedures which can be found in Base SAS 9.4 procedures guides. The key difference between the syntax of two procedures is the PERFORMANCE statement in the PROC HPSUMMARY.

Unlike the SUMMARY procedure which can only run on a single machine, the HPSUMMARY procedure can run on both a single machine and a distributed system. The PERFORMANCE statement facilitates setups for the HPSUMMARY to run on a distributed environment. The PERFORMANCE statement is used to (1) define performance parameters for multithreaded and distributed computing, (2) pass variables that describe the distributed computing environment, and (3) request detailed results about the performance characteristics of the HPSUMMARY procedure. For example, using the NODES option of the PERFORMANCE statement to specify the number of separate grid nodes that participate in the execution of the HPSUMMARY procedure, using the HOST option to specify the name of the appliance host, or using the NTHREADS option to specify how many threads are used by the HPSUMMARY procedure instance that runs on each node, etc. The PERFORMANCE statement can also be used to control whether PROC HPSUMMARY executes in single-machine or distributed mode (e.g., Specifying NODES=0 causes PROC HPSUMMARY to execute in single-machine mode only). More detailed information about setting up the PERFORMANCE statement of the PROC HPSUMMARY can be found in Base SAS® 9.4 Procedures Guide.

PROC HPSUMMARY <*options*> <*statistics-keywords*>; **CLASS** *variables* </ *options*>; **FREQ** *variable*; **OUTPUT** <**OUT** = *SAS-dataset*> <*output-statistic-specifications*> </ **AUTONAME**>; **PERFORMANCE** *performance-options*; **TYPES** *requests*; **VAR** *variables* </ **WEIGHT**=*weight-variable*>; **WAYS** *list*; **WEIGHT** *variable*;	**PROC SUMMARY** <option(s)> <statistic-keyword(s)>; BY <DESCENDING> variable-1<...<DESCENDING> variable-n> <NOTSORTED>; **CLASS** variable(s) </ option(s)>; **FREQ** variable; ID variable(s); **OUTPUT** <**OUT**=*SAS-data-set*><*output-statistic-specification*(s)> <id-group-specification(s)> <maximum-id-specification(s)> <minimum-id-specification(s)></ option(s)> ; **TYPES** request(s); **VAR** variable(s)</ **WEIGHT**=weight-variable>; **WAYS** list; **WEIGHT** variable;

Figure 1. General Syntax for PROC HPSUMMARY and PROC SUMMARY

Other important syntax differences between the HPSUMMARY and SUMMARY procedure relate to display and output. Regarding display, the SUMMARY procedure's output can be displayed in the output window using a PRINT option whereas the HPSUMMARY procedure does not display output to the output window; but its computed statistics can only be stored in a temporary SAS dataset in the Work library or in a permanent dataset using the OUTPUT statement; hence with HPSUMMARY procedure, statistic-keywords are only specified in the OUTPUT statement as seen in the following code snippet:

```
proc hpsummary data=grades;
   var Score;
   class Status Year;
   output out=result_HP mean= median= min= max=/AUTONAME;
run;
```

In contrast, with PROC SUMMARY, statistic-keywords can be specified in the PROC statement or in the OUTPUT statement as seen in the code below. In the below sample code, statistic Keywords are specified in the PROC statement and outputs are displayed in the output window using the PRINT option; the same statistic-keywords are also specified in the OUTPUT statement and their computed statistics are also stored in a SAS temporary dataset named result:

```
proc summary data=grades mean median min max PRINT;
   var Score;
   class Status Year;
   output out=result mean= median= min= max=/AUTONAME;
run;
```

Regarding the difference in output, by default HPSUMMARY procedure generates n-way results (i.e., displaying the highest value of _TYPE_ only; for example, if there are two categorical variables in the CLASS statement, only statistics of _TYPE_ = 3 are displayed) whereas, by default, the SUMMARY procedure generates all class variable combination types. To generate the desired statistics for all class combination types using the HPSUMMARY proc, the ALLTYPES or ALLWAYS option is used. Below are the syntax and the content of the output dataset generated by proc HPSUMMARY by default (Output 1) and with an ALLWAYS option (Output 2):

```
proc hpsummary data=grades;
   var Score;
   class gender Status;
   output out=result_HP mean= median= std= /AUTONAME;
run;
```

Obs	Gender	Status	_TYPE_	_FREQ_	Score_ Mean	Score_ Median	Score_ StdDev
1	F	2	3	2	86.0	86.0	5.65685
2	F	1	3	2	86.5	86.5	3.53553
3	M	2	3	2	84.5	84.5	4.94975
4	M	1	3	4	86.5	88.0	6.45497

Output 1. Content of the Output Dataset Generated by HPSUMMARY by Default

```
proc hpsummary data=grades ALLWAYS ;
   var Score;
   class gender Status;
   output out=result_HP mean= median= std= /AUTONAME;
run;
```

Obs	Gender	Status	_TYPE_	_FREQ_	Score_ Mean	Score_ Median	Score_ StdDev
1			0	10	86.0000	86.5	4.71405
2		2	1	4	85.2500	85.0	4.42531
3		1	1	6	86.5000	87.0	5.24404
4	F		2	4	86.2500	86.5	3.86221
5	M		2	6	85.8333	86.5	5.56477
6	F	2	3	2	86.0000	86.0	5.65685
7	F	1	3	2	86.5000	86.5	3.53553
8	M	2	3	2	84.5000	84.5	4.94975
9	M	1	3	4	86.5000	88.0	6.45497

Output 2. Content of the Output Dataset Generated by HPSUMMARY with an ALLWAYS Option

Below are the syntax and the content of the output dataset generated by PROC SUMMARY by default:

```
proc summary data=grades;
   var Score;
   class gender Status;
   output out=result mean= median= std= /AUTONAME;
run;
```

Obs	Gender	Status	_TYPE_	_FREQ_	Score_ Mean	Score_ Median	Score_ StdDev
1			0	10	86.0000	86.5	4.71405
2		1	1	6	86.5000	87.0	5.24404
3		2	1	4	85.2500	85.0	4.42531
4	F		2	4	86.2500	86.5	3.86221
5	M		2	6	85.8333	86.5	5.56477
6	F	1	3	2	86.5000	86.5	3.53553
7	F	2	3	2	86.0000	86.0	5.65685
8	M	1	3	4	86.5000	88.0	6.45497
9	M	2	3	2	84.5000	84.5	4.94975

Output 3. Content of the Output Dataset Generated by PROC SUMMARY by Default

PERFORMANCE COMPARISON

HPSUMMARY vs. SUMMARY

To compare the performance of HPSUMMARY AND SUMMARY procedures, average memory usage and average processing time (real time and CPU time) of the two procedures on a limited capacity single-user machine as well as on a high capacity node in a cluster system were observed for evaluation.

Experimental design: PROC SUMMARY and HPSUMMARY were set up to run against 24 simulated datasets with number of variables *k* set at 50, 100, 500, and 1000; and the number of records established at 10,000, 50,000, 100,000, 500,000, 1,000,000, and 10,000,000. In terms of volume the datasets range from 0.004Gb to 76Gb. Each process was replicated 10 times to obtain the average measures of interest. The experiments were conducted on two different platforms with a large difference in capacity: (1) a Windows 7 Professional laptop which has quad core with 2.80GHz each and a total of 8 Gb MEM, (2) a 32Gb-memory node with 16 2.6GHZ-CPU's and 20Mg cache per core in a Linux cluster. SAS 9.4 was used in the experiment. The syntax options are set so that the two procedures will generate the same output (i.e., same statistic-keywords for both procs, PROC SUMMARY ran with the NWAY option while PROC HPSUMMARY ran with default option, and both proc saved output to a temporary dataset).

On the Windows setting, each of the procedures was run on its default thread setting (i.e., threaded is on for the SUMMARY procedure, and number of thread=4 for the HPSUMMARY procedure). SAS logs were used to examine the completeness of each process and to obtain the average memory usage and processing times. For all cases the HPSUMMARY procedure completed successfully; however the SUMMARY procedure failed when the number of records and number of variables increased. Table 1 summarizes the results of this observation.

N of Obs	Number of Variables							
	50		100		500		1000	
	HP	Sum.	HP	Sum.	HP	Sum.	HP	Sum.
10,000	✓	✓	✓	✓	✓	✓	✓	✓
50,000	✓	✓	✓	✓	✓	✓	✓	Failed
100,000	✓	✓	✓	✓	✓	Failed	✓	Failed
500,000	✓	✓	✓	✓	✓	Failed	✓	Failed
1,000,000	✓	✓	✓	✓	✓	Failed	✓	Failed
10,000,000	✓	✓	✓	✓	✓	Failed	✓	Failed

Table 1. The Completion of the HPSUMMARY and SUMMARY Procedures Running against Different Datasets on a Windows machine with limited memory

Examining average real time and CPU time (across 10 replications) for each of the two procedures when both were finished successfully reveals that there is little difference between the average processing times of the two procedures. Figure 1 shows that for the dataset of 50 variables, the real time of the two procedures are virtually the same across datasets of different records when running on this four-core, single-user machine. If there is any difference, PROC SUMMARY's average real time is slightly less than PROC HPSUMMARY's when the data set gets larger. Similar observations were seen for the average CPU time as seen in Figure 2.

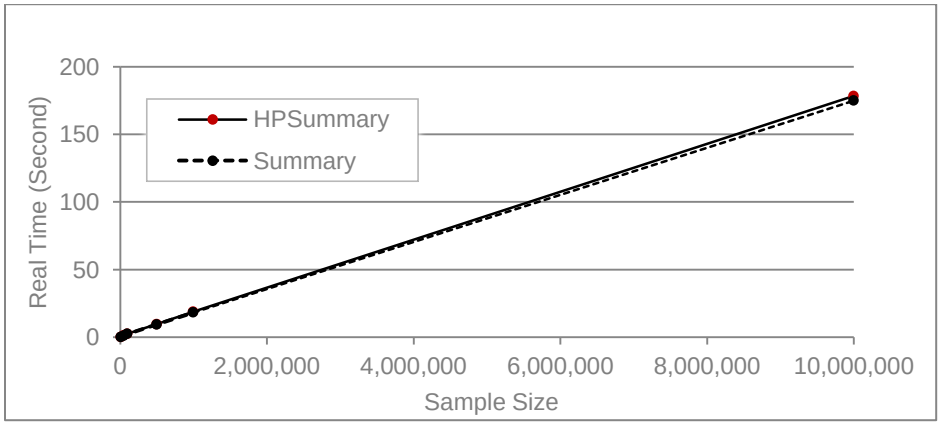

Figure 1: Average Real Time for HPSUMMARY and SUMMARY Procedure (k=50) on Windows Laptop

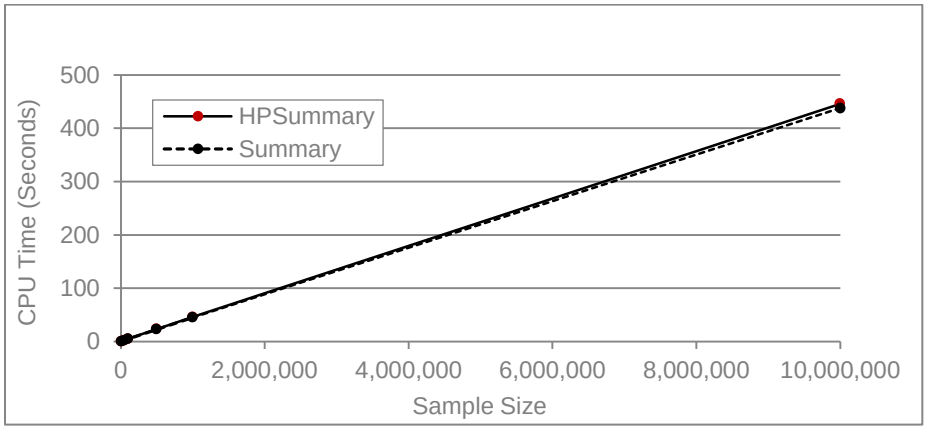

Figure 2: Average CPU Time for HPSUMMARY and SUMMARY Procedure (k=50) on Windows Laptop

In the Linux environment, PROC HPSUMMARY was set to run on eight threads and PROC SUMMARY was set with threaded on. Examining average real time and CPU time (across 10 replications) of the two procedures running in this setting shows that the relative performance of the HPSUMMARY and SUMMARY procedures is consistent across different dataset sizes (e.g., HPSUMMARY's processing time is slightly larger than SUMMARY's), and PROC HPSUMMARY consumed a little more CPU time than PROC SUMMARY did as sample size increases. Figure 3 and 4 describes the average real time and CPU time of the two procedures running with different dataset sizes (ranging from 0.08Gb to 76Gb) on this setting. The patterns are very similar to those obtained from the previous experiment on a limited-capacity setting: there is no clear difference in real time.

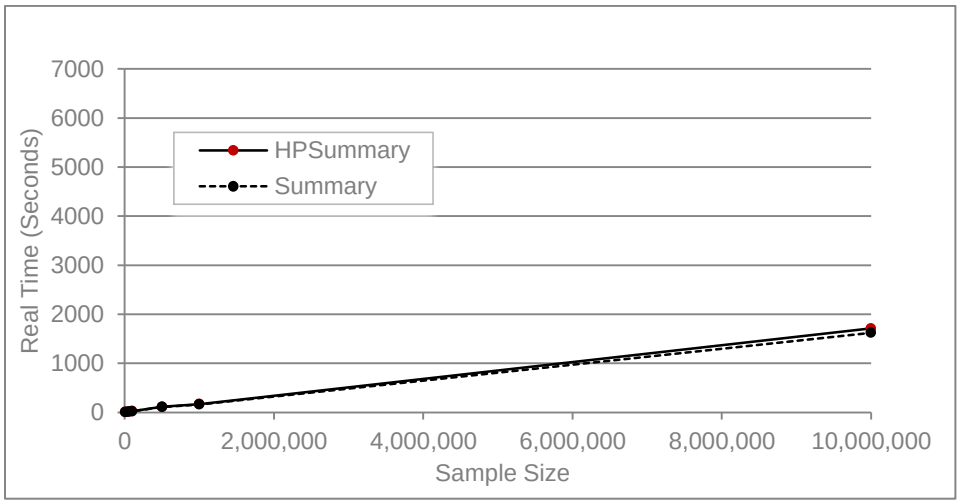

Figure 3: Real Time for HPSUMMARY and SUMMARY Procedure (k=1000) on Linux

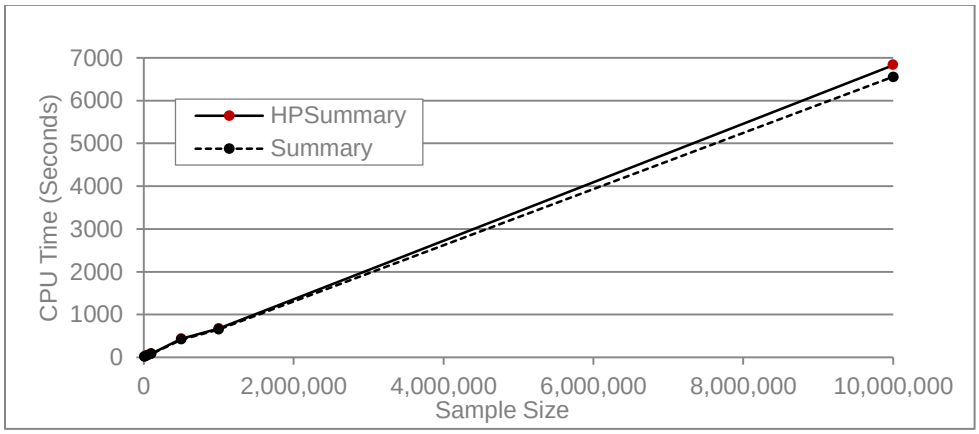

Figure 4: CPU Time for HPSUMMARY and SUMMARY Procedure (k=1000) on Linux

Examining the average amount of memory usage of each of the two procedures across different data volumes reveals that the HP version utilized more memory than the legacy version did, and the difference increases as the data volume increases. Figure 5 depicts average memory utilization by PROC HPSUMMARY and PROC SUMMARY. Though HPSUMMARY utilized more memory than SUMMARY, it seems to provide better memory management in a limited-memory environment than the legacy version did (e.g., in the limited memory machine, PROC SUMMARY failed when the dataset was large). It is also observed that each of the two procedures efficiently reserved the amount of memory needed for their

executions (e.g., there is not much difference between amount of memory utilized and memory requested). Figure 6 describes the maximum amount of memory each of the two procedures required from the system.

HPSUMMARY ON DIFFERENT THREAD COUNTS

To examine how efficient the HPSUMMARY procedure is in processing a large volume of data, the procedure was set up to run in single- machine mode on a Linux-based cluster system. It was set up to run on 4, 8, and 16 threads sequentially with three different data set sizes, small (50 variables and 10,000 rows ≈ 0.004 Gb), medium (500 variables x 10,000,000 rows ≈ 38Gb), and large (1000 variables x 10,000,000 rows ≈ 76Gb). Figure 7 shows that when the dataset was small, the real times are virtually the same for all thread counts; and when the data volume increases, increasing in thread counts substantially reduces the real time required by PROC HPSUMMARY. Based on the experiment real time, the HP procedure shows to be more efficient with larger data volume (e.g. the discrepancy between the average real time between the high thread count vs. low thread counts in a larger data volume is bigger than that on a smaller data volume). However, the procedure's efficiency in terms of processing time appears limited when factoring in the consumed CPU time. Figure 8 shows that the consumed CPU time increasing rate is steep as the data volume increase while the decreasing rate of real time is more moderate (Figure 7).

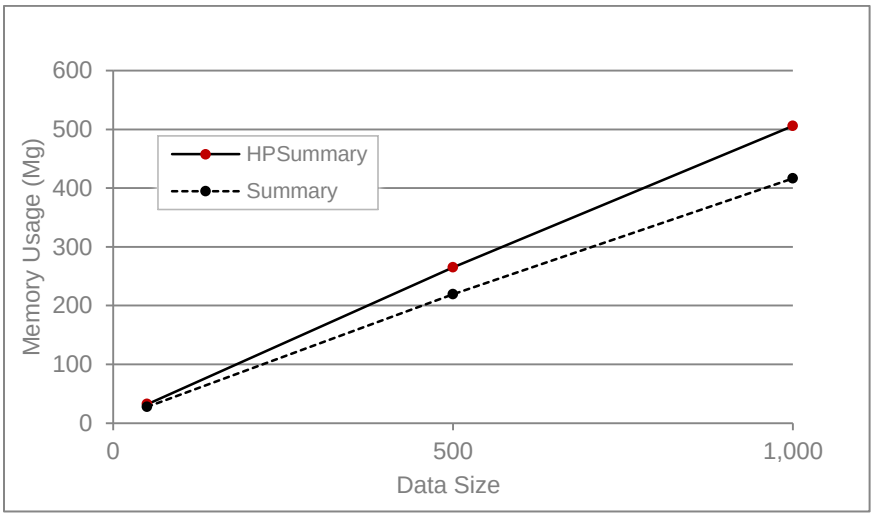

Figure 5: HPSUMMARY and SUMMARY's Memory Usage by Data Size

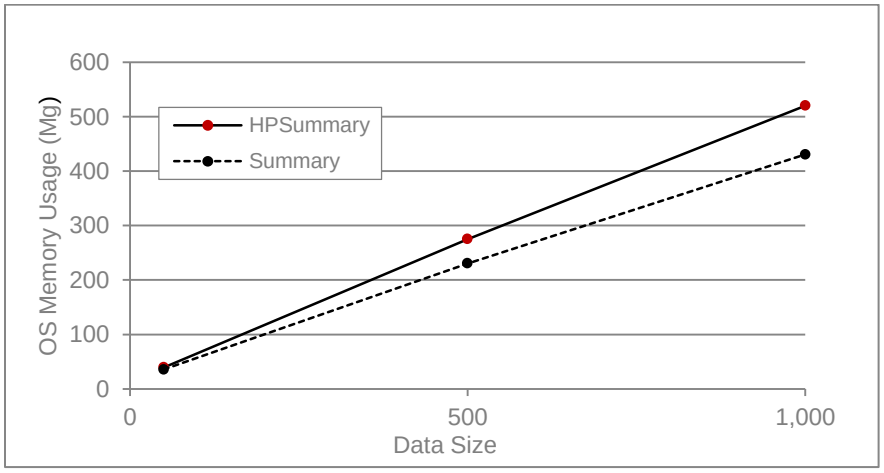

Figure 6: HPSUMMARY and SUMMARY's OS Memory by Data Size

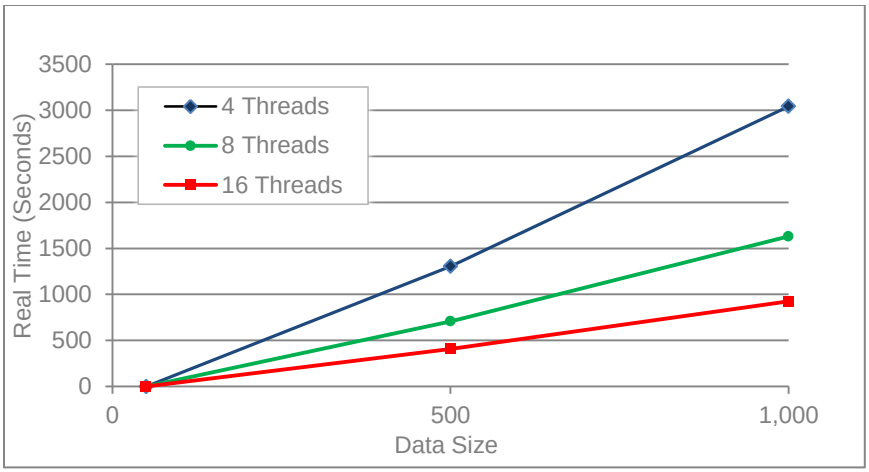

Figure 7: HPSUMMARY's Real Time for Different Thread Counts

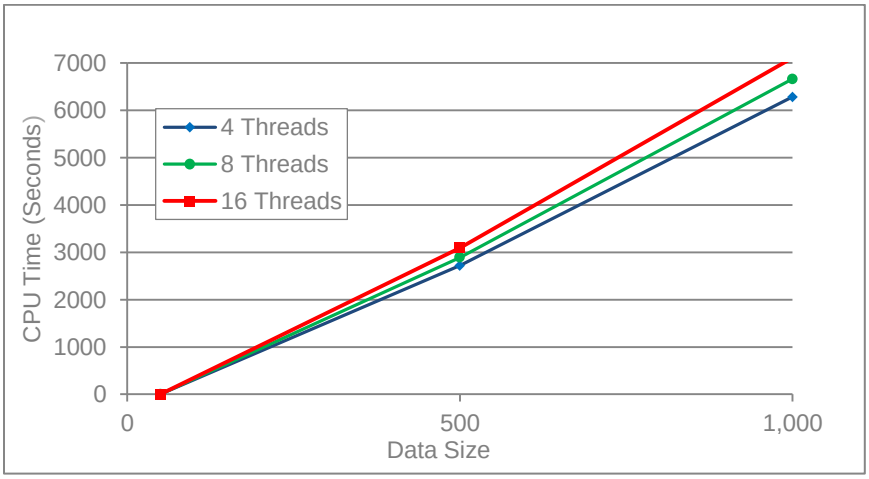

Figure 8: HPSUMMARY's CPU Time for Different Thread Counts

CONCLUSION

Besides the advantage that the HPSUMMARY procedure can run on a distributed system whereas the SUMMARY procedure cannot, the HPSUMMARY procedure provides better memory management than the SUMMARY procedure does when running on a memory-limited, single-user machine. In addition, on a high performance system with large-number-of-CPU machines, the HPSUMMARY utilized all cores available to it, substantially reducing real time when the number of available cores increases. However, based on this experiment, the HPSUMMARY procedure does not seem to provide any clear execution time advantage over its legacy version when running on either setting of a limited-capacity, single-user machine or on a high performance machine. The implications are (1) regarding memory management in a limited-memory, single-user machine environment, if available, HPSUMMARY can be a preferred choice to SUMMARY; and (2) regarding real time, in a busy, shared environment where requesting large computing resource requires a long waiting time, trading off between waiting time and shorter real time should be taken into consideration as the HPSUMMARY's decrease in real time is not proportional with its increase in consumed resources.

CONTACT INFORMATION

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

Anh P. Kellermann
University of South Florida
4202 East Fowler Ave., EDU 354
Tampa, FL 33620
Fax: (813) 974-4495
Email: napham@mail.usf.edu