

**Differences in SAS Computation Time between
the AMD Opteron and Intel Xeon when running a
Computationally Intense SAS Randomization
Program**

James D. Abbey
June 30, 2004

1 - Introduction

1.1 – Purpose of Experiment

This experiment sought differences in computational speed between the AMD Opteron and Intel Xeon server processors under nearly identical conditions when running a completely self-contained and highly computationally intense SAS randomization. Both CPU architectures were only tested at the two processor level. The computational speed was measured in seconds. The architectures showed a statistically significant difference and demonstrated that for this I/O intense randomization program the AMD Opteron is the better CPU.

1.2 – Audience for the Results

The target audience for this research includes SAS power users and SAS server administrators. Also, this research can be used by MIS and technical people in other fields. However, some familiarity with SAS and mathematical concepts is required for full understanding.

1.3 – Testbeds

Operating System: Windows 2000 sp 4. Hyperthreading was disabled in the Xeon because Windows 2000 does not support this feature.

Intel Xeon Hardware:

Processors: Two Intel Xeon 2.4Ghz 533Mhz Quad Pumped Bus Socket 604

Memory: 2GB Buffalo Technology PC3200 Registered ECC DDR (Samsung Chips)

Motherboard: Supermicro X5DP8-G2 Intel E7501 Dual Xeon

RAID Card: Adaptec 2010S 48MB Onboard Cache

Hard Drives: Fujitsu MAS 15,000rpm 36.4GB (2 in JBOD)

Power Supply: Antec 550Watt 24-pin Power Supplies

Chassis: Lian-Li Cases PC-73SL

AMD Opteron Hardware:

Processors: Two AMD Opteron 242 (1.6Ghz) 800Mhz Hyper Transport Socket 940

Memory: 2 GB Buffalo Technology PC3200 Registered ECC DDR (Samsung Chips)

Motherboard: Tyan S2882UG3NR K8SPro AMD 8131/8111 Chipsets

RAID Card: Adaptec 2015S 48MB Onboard Cache

Hard Drives: Fujitsu MAS 15,000rpm 36.4GB (2 in JBOD)

Power Supply: Antec 550Watt 24-pin Power Supplies

Chassis: Lian-Li Cases PC-73SL

Comparable cost drove the choice of hardware. The cost of each server totaled around \$2600 each in April, 2004. The Xeon was slightly more expensive due solely to a \$15 per processor higher cost. All other components are identical or nearly so except the system motherboards. Finally, though the servers cost \$2600 each for the experimenter

to *build by hand*, nearly identical servers from IBM (Opteron) and Dell (Xeon) cost almost twice as much when purchased pre-made.

2 – Experiment Details

2.1 – Experimental Program

This program was created by an Iowa State University Statistics graduate student, Ling Huang. Ms. Huang created this program to generate a conglomerate random sample of deer in Iowa to estimate the number of deer infected with Chronic Wasting Disease (CWD). The program manipulates a database of 2.1 million * 5 variables or 10.5 million pieces of pertinent information. Further, the program uses multiple PROC SORT functions as well as i=1 to 1000 macros, which entail multiple functions including PROC SORTS, PROC SQLs and PROC SURVEYSELECTs in the macros themselves. According to the SAS help files, PROC SORT does use multithreading when available. Such multithreading was apparent in the Windows 2000 task manager, though the second processor was only used at around the 10-20% level.

2.2 – Experimental Design

This experiment used the completely randomized experimental design. A breakdown of the factors follows:

- *Response*: Time to complete the programs in seconds (CPU time in Windows 2000 task manager)
- *Sole Factor of Interest*: CPU Architecture [AMD Opteron or Intel Xeon]

2.3 – Experimental Details and Description

Experimental Material: The testbeds and program information can be found above in sections 1.3 and 2.1, respectively.

Experimental Control: The environment for this experiment is a stable temperature, dehumidified basement computer room. Both systems sit in identical Lian-Li PC-73SL cases. Both cases use the same style of Antec 550 Watt power supply. Nearly all components are identical across systems to keep external variables from altering the results. At the time this experiment was conducted, the Fujitsu MAS 36.5GB 15,000 rpm drives were the fastest server drives on the market according to storagereview.com.¹

Experimental Replication: A total of 24 runs were completed. Thus, each treatment level had 12 replicates. Hence, the experiment had a treatment level $\alpha = 0.05$, $\beta = 0.05$ and significant difference of 1.6 deviations.

Experimental Randomization: To randomize in a completely randomized experiment, you simply randomize the order of all runs. The order of runs was randomized by

¹ Storagereview.com: http://www.storagereview.com/articles/200304/20030429MAS3735_1.html

entering the run names (Opteron*12 and Xeon*12) in column one, and next creating and then sorting by random digits in column two. The order of the runs was as follows:

CPU Architecture	Random Digits
Opteron	0.06637608
Opteron	0.07189302
Xeon	0.18252131
Xeon	0.30227965
Opteron	0.33443062
Xeon	0.44313184
Opteron	0.44773139
Xeon	0.4794709
Xeon	0.55062306
Opteron	0.62685095
Opteron	0.66485226
Opteron	0.69027736
Opteron	0.7115528
Xeon	0.74171293
Opteron	0.74475463
Xeon	0.76363492
Xeon	0.76646352
Xeon	0.7800264
Xeon	0.79015576
Opteron	0.80096628
Opteron	0.84460186
Xeon	0.88244515
Xeon	0.9440356
Opteron	0.96797132

2.3.1 – Random Order Table: Displays random order of runs.

Experimental Process: Each run entailed opening a fresh copy of SAS by double clicking the desktop icon. Next, any changes to the output file and log file names were made, and the program was saved. The output and log files were sent to an external file due to size overflows in the SAS window. Note that each run of the program generates over a gigabyte of total files due to the macros, log and output files.

Unfortunately, SAS’s own internal timer could not be relied upon to give the overall time for the experiment as each function was measured, but not the program as a whole. Even when FULLSTIMER was set to 1, SAS merely recorded when the user shutdown SAS, not when the program was complete. Hence, the run times were measured by the CPU time reported in Windows 2000 task manager. After each run, SAS was closed and the process repeated. A sample screen shot from an AMD Opteron run follows:

Image Name	PID	CPU	CPU Time	Mem Usage	Peak Mem Usage	VM Size	Threads	I/O Read Bytes	I/O Write Bytes
winmgmt.exe	756	00	0:00:03	496 K	5,760 K	816 K	5	6,155,434	7,951,922
WINLOGON.EXE	160	00	0:00:00	2,044 K	8,696 K	5,872 K	15	971,627	4,456
vstskmgr.exe	568	00	0:00:00	292 K	5,144 K	5,072 K	11	435,656	12
UpdaterUI.exe	1092	00	0:00:00	388 K	5,000 K	2,164 K	4	4,967	0
taskmgr.exe	1000	00	0:00:04	1,164 K	3,020 K	892 K	3	0	0
System Idle Process	0	99	139:28:08	16 K	16 K	0 K	2	0	0
System	8	00	0:01:40	216 K	652 K	24 K	44	33,616	22,558,692
svchost.exe	1248	00	0:00:00	8,040 K	8,124 K	5,288 K	6	7,210	442,983
svchost.exe	800	00	0:00:01	13,496 K	14,612 K	11,588 K	7	1,754,014	984,692
svchost.exe	472	00	0:00:00	9,048 K	9,152 K	5,032 K	28	616,794	372,036
svchost.exe	412	00	0:00:00	4,104 K	4,124 K	1,508 K	8	269,580	312
SPOOLSV.EXE	440	00	0:00:00	5,588 K	6,472 K	3,744 K	11	40	12
smss.exe	148	00	0:00:00	384 K	2,048 K	1,076 K	6	18,754,048	39,936
SideCar.exe	1200	00	0:00:00	2,364 K	2,480 K	776 K	2	1,020	0
shstat.exe	1136	00	0:00:00	432 K	4,796 K	4,580 K	6	0	0
services.exe	216	00	0:00:00	5,856 K	5,984 K	2,924 K	40	23,399,285	29,135,994
sas.exe	1436	00	0:25:00	46,444 K	104,288 K	35,448 K	23	476,610,595,454	5,619,016,650
regsvc.exe	688	00	0:00:00	916 K	924 K	268 K	2	54	12
PRONoMgr.exe	1124	00	0:00:00	2,892 K	2,900 K	908 K	3	0	0
naPrdMgr.exe	596	00	0:00:00	1,144 K	4,588 K	4,536 K	4	3,813	219
mstask.exe	704	00	0:00:00	3,244 K	3,256 K	1,112 K	6	9,770	444
MDM.EXE	652	00	0:00:00	2,892 K	2,916 K	792 K	4	5,072	548
mcshield.exe	552	00	0:07:36	15,092 K	15,092 K	14,752 K	27	2,947,289,513	435,624
LSASS.EXE	228	00	0:00:00	1,112 K	5,172 K	2,376 K	16	984,122	269,748
krbcc32s.exe	1216	00	0:00:00	1,216 K	1,216 K	308 K	3	0	0
FrameworkServic	500	00	0:00:00	7,640 K	7,940 K	4,552 K	11	164,459	994,850
explorer.exe	1056	00	0:00:15	3,780 K	11,608 K	7,012 K	10	16,902,557	2,372,561
CTEMO.NE	1160	00	0:00:06	2,084 K	2,168 K	512 K	1	0	0

2.3.2 *Opteron Run Example CPU Time*: Displays Windows 2000 task manager data.

A closer view at the data from the SAS run follows:

Image Name	PID	CPU	CPU Time	Mem Usage	Peak Mem Usage	VM Size	Threads	I/O Read Bytes	I/O Write Bytes
sas.exe	1436	00	0:25:00	46,444 K	104,288 K	35,448 K	23	476,610,595,454	5,619,016,650

2.3.3 *Opteron Run Example CPU Time Zoom Shot*: Displays Windows 2000 task manager data for SAS.

As you can see, Windows 2000 generated more information than the CPU time alone, but that is a topic covered later in this report. In this particular instance, the randomization program took 25:00 minutes of CPU time to complete.

3 – CPU Time – Data Analysis and Discussion

3.1 –CPU Time Model Significance

Overall Model Significance: The following shows the R-Square and t-test/F-stats for the experimental model:

Oneway Anova	
Summary of Fit	
Rsquare	0.996345
Adj Rsquare	0.996179
Root Mean Square Error	7.609553
Mean of Response	1616.792
Observations (or Sum Wgts)	24

3.1.1 – *Summary of Fit Display*: Displays R, R adj and other information.

The R-square and adjusted R-Square indicate that the model appears to be very decisive in accounting for variability in the experimental material. However, the linear model may not explain all the variance as well as the R value indicates. The “Observations” display that 24 runs were incorporated in the statistical analysis.

F-Stat (ANOVA) and t-test:

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio	Prob > F
CPU Type	1	347282.04	347282	5997.413	<.0001
Error	22	1273.92	58		
C. Total	23	348555.96			

t Test

Opteron-Xeon

Assuming equal variances

Difference	-240.58	t Ratio	-77.443
Std Err Dif	3.11	DF	22
Upper CL Dif	-234.14	Prob > t	0.0000
Lower CL Dif	-247.03	Prob > t	1.0000
Confidence	0.95	Prob < t	0.0000

3.1.2 Treatment F-Stat and t-test: Displays treatment level F-Stats and p-values along with t-test.

Both the F-test and the t-test show the same result: the AMD Opteron is statistically significantly different from the Intel Xeon. The formal test is H_0 : No statistically significant difference in the amount of CPU time to complete the randomization program between the AMD Opteron and Intel Xeon processors. The F-Ratio of 5997.413 and accompanying Prob > F of <0.0001 indicate that pure chance is unlikely to have caused this result. Further, the t-test under the same H_0 indicates that the probability of chance alone causing the observed result is Prob < t of 0.0000. Hence, a statistically significant difference exists in the amount of CPU time consumed between the AMD Opteron and Intel Xeon processors.

3.2 - CPU Time Treatment Means and Test Meaning

So, a statistically significant difference between the AMD Opteron and Intel Xeon exists as shown in section 3.1. The treatment means, displayed below, show that the AMD Opteron was the far faster architecture when running this randomization program on the nearly identical servers:

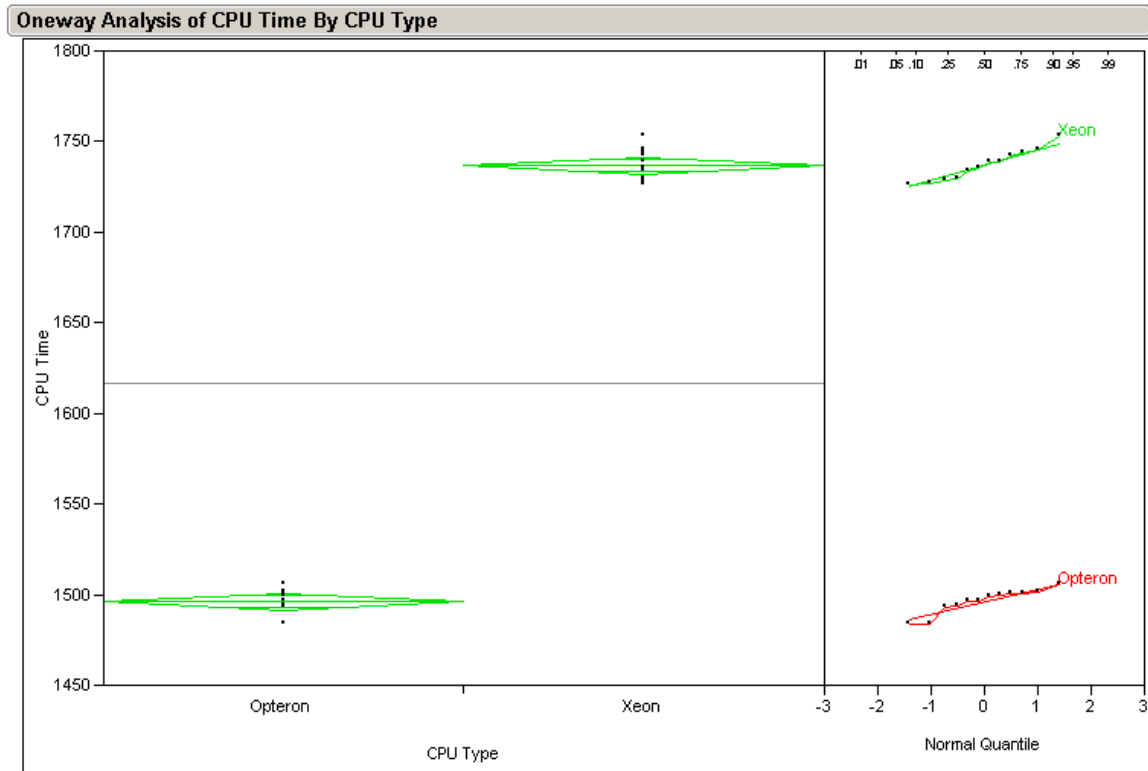
Means for Oneway Anova

Level	Number	Mean	Std Error	Lower 95%	Upper 95%
Opteron	12	1496.50	2.1967	1491.9	1501.1
Xeon	12	1737.08	2.1967	1732.5	1741.6

Std Error uses a pooled estimate of error variance

3.2.1 – CPU Architecture Treatment Means

As you can see, the AMD Opteron achieved an average time of only 1496.50 seconds or approximately 24:56 minutes. The Intel Xeon took a somewhat longer 1737.08 seconds or approximately 28:57 minutes. Hence, the AMD Opteron completes the randomization program in approximately 86.2% of the time the Intel Xeon took. Whether this time savings is meaningful in the real world setting is up to you. However, in general, the lower the CPU time, the more efficient the architecture. Hence, the AMD Opteron appears to be the better chip at handling this I/O intensive randomization program. See the scatter and quantile plots, below:



3.2.2 – CPU Architecture Opteron vs. Xeon Scatter Plot and Quantile Plot

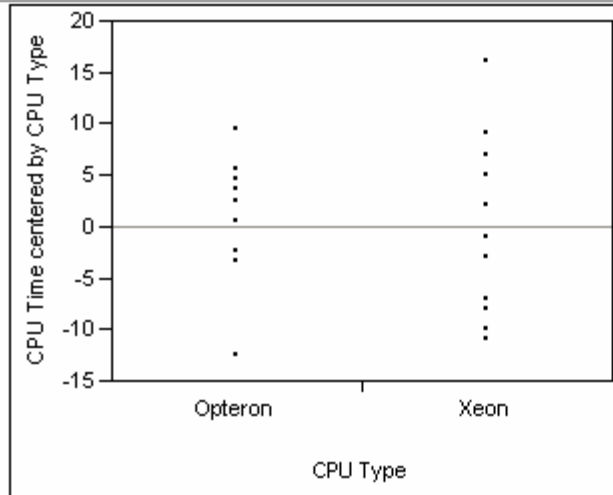
The above plots simply show that the AMD Opteron was significantly below the Intel Xeon in CPU time to run the program. Further, you can see how tightly the two processors clustered around their respective mean CPU times.

4 - Residuals Verification and Analysis

4.1 – ANOVA Assumptions Residual Analysis for CPU Time

Residual by Predicted Plot: One of the six ANOVA assumptions is that the variance of any one predicted value should be no more than three times larger than the smallest variance of a predicted value. The following plot allows graphical analysis of the assumption:

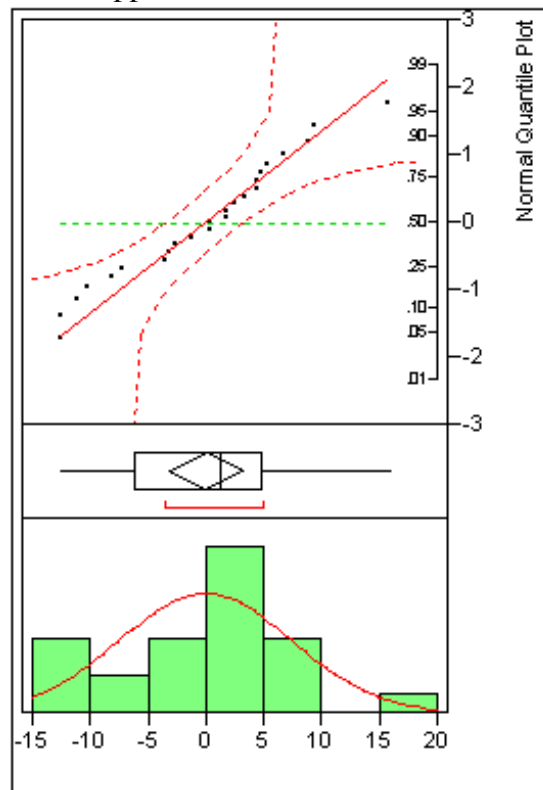
Oneway Analysis of CPU Time centered by CPU Type By CPU Type



4.1.1 – Residual by Predicted Plot Real Time

The condition of equal variance appears met. The distribution of the residuals does not display that the Opteron's residual cluster was even close to three times smaller than the Xeon's residual cluster.

Normality of the Residual Distribution: Another of the ANOVA assumptions is that the residuals are normally distributed. To check for this normality, the normal quantile plot and histogram of the residuals appears below:



4.1.2 – Residual Distributions in Histogram and Quantile Plot

The normality assumption appears met for this data. A quick look at the quantile plot shows that one outlier may exist, but the remainder of the data looks quite good. Further, the pattern of points follows a bit of an up and down pattern but nothing severe enough to be called systematic. Also, the histogram shows a moderately normal distribution that is perhaps a bit lopsided to the left. Overall, even if normality is not present, the F-Stat and t-test above appear unharmed.

5- Conclusions

In sum, the AMD Opteron was the clear winner in CPU Time consumed when running this randomization program. Further, because the servers are nearly identical in all ways, except the motherboards and processors, you should likely choose the AMD Opteron if you are primarily interested in programs such as this I/O intensive SAS randomization program.