CPU Core vs Thread Analysis

Author: Craig Shallahamer (craig@orapub.com), Version 1f, 10-May-2011

Background and Purpose

The purpose of this notepad is to:

1. Demonstrate that sometimes there is a difference in CPU utilization calculations from different sources; Oracle and the operating system.

2. Is the difference significant?

3. Why is there a difference? (This is not detailed in the notepad.)

Currently, on must Oracle system is little discrepancy between the Oracle and OS based utilization. However, on a few occasions I have personal seen examples where there is clearly a difference. This motivated me to write a simple script to gather and record both Oracle and OS based utilization data, ask for volunteers to run the script on their production systems, and send me the resulting data. I emailed about 50 people asking for participation and I received seven sample sets back. (While this may seem like a low response, considering how busy DBAs are and I'm asking them to gather data from a real production environment, I'm not only thankful for their response but also very pleased with the response.)

The core utilization formula is:

U = R/C

where: R are the requirements and C is the capacity.

Oracle as a data source. Starting in Oracle 10g, the v$osstat view's busy_time provides operating system cpu time consumption in centi-seconds...the requirements (R). The v$osstat view also contains information about the number of cpu cores. While the statistic name is still inconsistent, taking the largest value from any stat that contains, cpu, core, socket but does not contain the word thread appears to provide the number of cpu cores. The capacity (C) is simply then the number of cores multiplied by the sample interval. In a formula structure, it looks like this:

U = busy time / (interval X cores)

Note: The busy time and the interval must use the same unit of time, which is typically seconds. That is, the busy time is the delta; the ending busy_time - starting busy_time.

Operating system as a data source. Without getting into the operating system's data source and timing details, we can gather the operating system based utilization by simply using the widely available, **vmstat** command. Anytime OS utilization is mentioned, assume the utilization was gathered and based on vmstat. The utilization could be calculated based on cores or threads, more specifically the capacity is based on threads and not cores creating a much large capacity figure when compared to core based capacity. Depending on the number and distribution of threads, if threads are used to determine CPU capacity the Oracle and OS based utilization calculations can be significantly different. For more details, see my blog entry on this subject.

Data Loading

All the data sets are contained at the bottom of this notebook along with their associated variable name. Make sure to set the appropriate data source in the section and then ***re-evaluate the entire notebook twice***. One re-evaluation will not work because the data resides at the bottom of the notebook.

In[48]:=
```
UtilOracle = AG1UtilOracle;
UtilOs = AG1UtilOs;
```

Basic Statistics

In this section I calculate the basic statistics, such as the mean and median. My objective is to ensure the data has been collected and entered correctly and also to compare the two datasets to see if they appear to be different.

Here I calculate the utilization error, which is actual the difference between the Oracle (v$osstat based) and OS (vmstat based) values. The UtilDiff is the pure difference. The UtilError is formatted to easily create a residual (i.e., error graph). (Although *Mathematica* does have some other cool ways to do this, I didn't use it...out of time.)

In[50]:=
```
UtilError = {};
UtilDiff = {};
Table[
  AppendTo[UtilDiff, UtilOracle[[i]] - UtilOs[[i]]];
  AppendTo[UtilError, {UtilOracle[[i]], UtilDiff[[i]]}];
  , {i, 1, Length[UtilOracle]}
 ];
UtilDiff
UtilError
```

Out[53]=
```
{0.036, 0.0726, 0.0305, 0.0709, 0.0617, 0.0641, 0.0399, 0.0309, 0.0537, 0.0242,
 0.0083, 0.0186, 0.025, -0.0023, 0.009, 0.0125, 0.0155, 0.0286, 0.0272, 0.0366,
 0.0353, 0.0514, 0.0586, 0.1166, 0.1149, 0.1485, 0.1542, 0.175, 0.156, 0.1472,
 0.1575, 0.1545, 0.1534, 0.1506, 0.152, 0.1483, 0.1468, 0.1306, 0.1421, 0.1154,
 0.1041, 0.0757, 0.0687, 0.0521, 0.0529, 0.077, 0.0734, 0.0264, 0.004, 0.0311,
 -0.0178, 0.0465, 0.0362, 0.0379, 0.0299, 0.0277, 0.057, 0.0187, 0.0172, 0.0134}
```

Out[54]=
```
{{0.356, 0.036}, {0.3426, 0.0726}, {0.3505, 0.0305}, {0.4609, 0.0709}, {0.4517, 0.0617},
 {0.3941, 0.0641}, {0.3299, 0.0399}, {0.2209, 0.0309}, {0.1437, 0.0537},
 {0.1142, 0.0242}, {0.1083, 0.0083}, {0.1086, 0.0186}, {0.115, 0.025}, {0.1277, -0.0023},
 {0.169, 0.009}, {0.1325, 0.0125}, {0.1655, 0.0155}, {0.1886, 0.0286}, {0.2572, 0.0272},
 {0.2966, 0.0366}, {0.3353, 0.0353}, {0.3814, 0.0514}, {0.4786, 0.0586},
 {0.5666, 0.1166}, {0.6249, 0.1149}, {0.6585, 0.1485}, {0.6642, 0.1542},
 {0.655, 0.175}, {0.666, 0.156}, {0.6472, 0.1472}, {0.6475, 0.1575}, {0.6345, 0.1545},
 {0.6534, 0.1534}, {0.6506, 0.1506}, {0.642, 0.152}, {0.6483, 0.1483}, {0.6268, 0.1468},
 {0.5806, 0.1306}, {0.5721, 0.1421}, {0.5254, 0.1154}, {0.4541, 0.1041},
 {0.4157, 0.0757}, {0.3887, 0.0687}, {0.3321, 0.0521}, {0.3029, 0.0529}, {0.297, 0.077},
 {0.1734, 0.0734}, {0.1964, 0.0264}, {0.204, 0.004}, {0.2111, 0.0311}, {0.2222, -0.0178},
 {0.3765, 0.0465}, {0.3562, 0.0362}, {0.3479, 0.0379}, {0.3299, 0.0299},
 {0.2077, 0.0277}, {0.167, 0.057}, {0.1187, 0.0187}, {0.1372, 0.0172}, {0.1034, 0.0134}}
```

Next I calculate basic statistics and display them so we can visually see if there are any significant numerical

differences. The correlation (actually Pearson's correlation coefficient, r) for *all the data* between the two data sources is also computed. The worst is 0 and the best correlation is 1.

In[55]:=
```
MaxUtilOracle = Max[UtilOracle];
MeanUtilOracle = Mean[UtilOracle];
MedianUtilOracle = Median[UtilOracle];
StdUtilOracle = StandardDeviation[UtilOracle];
CountUtilOracle = Length[UtilOracle];
MaxUtilOs = Max[UtilOs];
MeanUtilOs = Mean[UtilOs];
MedianUtilOs = Median[UtilOs];
StdUtilOs = StandardDeviation[UtilOs];
CountUtilOs = Length[UtilOs];
MaxUtilDiff = MaxUtilOracle - MaxUtilOs;
MeanUtilDiff = MeanUtilOracle - MeanUtilOs;
MedianUtilDiff = MedianUtilOracle - MedianUtilOs;
StdUtilDiff = StdUtilOracle - StdUtilOs;
Grid[
 {{"Data Source", "Util Max", "Util Avg", "Util Median", "Util Std Dev", "Samples"},
  {"Oracle", MaxUtilOracle, MeanUtilOracle,
   MedianUtilOracle, StdUtilOracle, CountUtilOracle},
  {"OS", MaxUtilOs, MeanUtilOs, MedianUtilOracle, StdUtilOs, CountUtilOs},
  {"Diff (Ora-OS)", MaxUtilDiff, MeanUtilDiff, MedianUtilDiff, StdUtilDiff}
 }, Frame → All]
CorrelUtil = Correlation[UtilOracle, UtilOs]
```

Out[69]=

Data Source	Util Max	Util Avg	Util Median	Util Std Dev	Samples
Oracle	0.666	0.367242	0.34525	0.192862	60
OS	0.51	0.298833	0.34525	0.144915	60
Diff (Ora-OS)	0.156	0.0684083	0.04025	0.0479474	

Out[70]=
```
0.989723
```

Utilization Difference (i.e., error) Analysis

I have a hunch that in certain circumstances the two utilization calculations will result in different values. I want to check if any of our samples exhibit this behavior. It's not a simple as it may seem because sometimes a difference may seem significant but it in reality randomness can explain the difference. To tell if the utilization data sets are indeed different, we need to perform some tests. Below is a series of tests and observations to, hopefully, tell us if the two utilization calculations can result in different utilizations.

1. Hypothesis test to statistically tell if our sample sets are significant different.
2. Histogram is visually look to see if the difference in the utilizations is normally distributed.
3. Scatter plot so we can visually observe the utilization difference.
4. Residual (i.e., error or utilization difference) graph to see if the difference changes based on the utilization.

Hypothesis test to check if the utilization differences can be explained by randomness.

Each sample set consists of 60 sample of two utilizations. The Oracle utilization was calculated bast on v$osstat and is core based, not thread based. The OS utilization was gathered from vmstat. I want to know if they are really and truly different. To do this, I need to perform a statistical hyposthesis test.

While I suspect some of our sample set's OS calculate CPU utilization based on threads and not cores AND I calculate Oracle's CPU utilization based on cores, is the end result different? Perhaps it makes no differ-

ence; 58% compared to say 61%. We can perform a hypothese test is check if statistically there is so much of a difference it can not be expalined by randomness and so something must be involved...like the underlying calculations.

Hypothesis test details:

We want to test if our two sample sets are statistically different. If they are different, then we know any difference is not caused by randomness. The difference could be caused any number of things, but not randomness. But it's a little more complicated then is usual... We can't expect all the Oracle utilization data values to be normally distributed. Think about it; there could easily be two clusters of values, say around 20% and 50% busy. This would result in a non-nomal distirbution. The same is true for the vmstat utilization data values. Because of this non-normality, we can't simply perform a t-test.

All is not lost. According to the central limit therom, the new sample set based on the differences between the utilization for each collection (one value for Oracle based and another value from vmstat) will be normal...if any differences can be explained by randomness. One way to do this is to compare our new sample set (created from the utilization differences) to the normal distribution. If they are significantly different, we know the utilization difference are not caused by randomness. This is what I'm doing below via *Mathematica*.

In[71]:=
```
generatedDataSet = RandomVariate[
   NormalDistribution[Mean[UtilDiff], StandardDeviation[UtilDiff]], 10 000];
α = 0.05;

h = DistributionFitTest[UtilDiff, generatedDataSet, "HypothesisTestData"];
h["TestDataTable", All]
If[AndersonDarlingTest[UtilDiff, generatedDataSet] > α,
 Print["Null hypo accepted: Data is
    statistically similar. Any error is likely due to randomness."],
 Print["Null hypo rejected: Data is statistically different. The
    results cannot be explained by randomness, so there
    must be something else causing the differences."]
]
```

Out[74]=

	Statistic	P-Value
Anderson-Darling	2.68734	0.0396669
Cramér-von Mises	0.441583	0.0562396
Kolmogorov-Smirnov	1.18237	0.122083
Kuiper	0.283717	0.00150498
Pearson χ^2	40.2442	0.0000324928
Watson U^2	0.4092	0.00636816

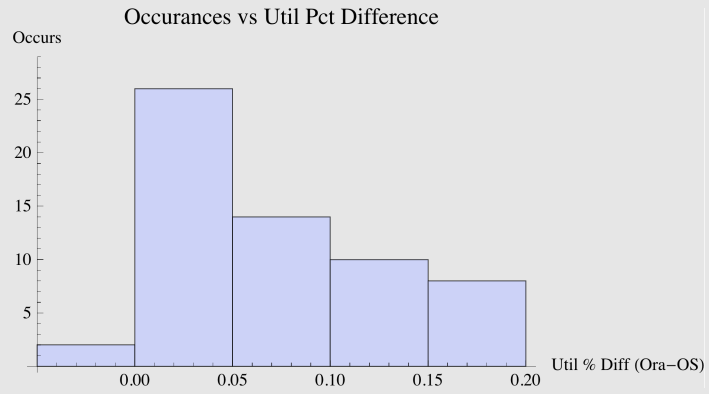
```
Null hypo rejected: Data is statistically different. The results cannot be explained
  by randomness, so there must be something else causing the differences.
```

Histogram of error to visually look to see if the error is normally distributed.

A hypothesis is one of the ways to check if the difference in utilization values (statically called the error) is normally distributed, but another method is to visually look at a error in a histogram format. Viewing the Histogram, we can also learn about the skewness of the error and trends. If the utilization calculation error is pretty much the same and hovering around the average, we would expect to see the typical bell curve.

In[76]:=
```
Histogram[UtilDiff, PlotLabel → "Occurances vs Util Pct Difference",
 AxesLabel → {"Util % Diff (Ora-OS)", "Occurs"}]
```

Out[76]=

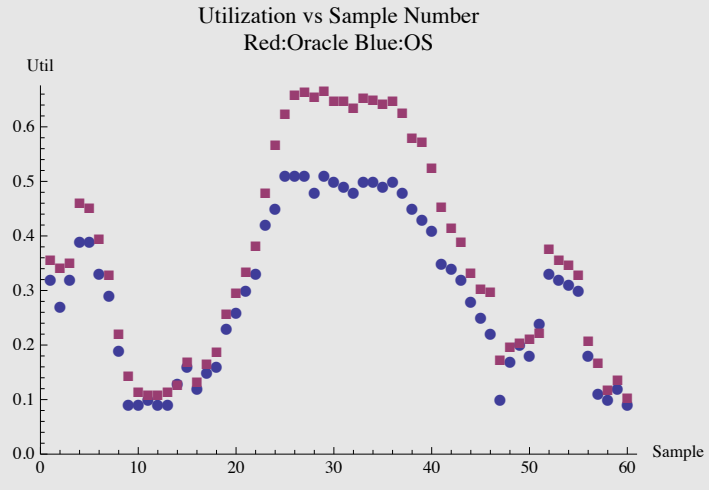


Scatter plot by time to observe if the error changed based on sample gathering time and also utilization.

This scatter plot shows each sample sets' utilization based on Oracle (red squares) and the OS (blue circles). The vertically further apart the points are the greater the utilization difference. This is a good way to check our data set for outliers but also to see if utilization difference (i.e.., error) trends and is highly variant. We may also get a glimpse of an increased utilization gap as utilization increases.

In[77]:=
```
ListPlot[{UtilOs, UtilOracle},
 PlotLabel → "Utilization vs Sample Number\nRed:Oracle Blue:OS",
 AxesLabel → {"Sample", "Util"}, PlotMarkers → {Automatic, Small},
 PlotRange → {{0, Length[UtilOracle] + 1}, {0, Max[UtilOracle, UtilOs] + .01}}]
```

Out[77]=



Residual plot to clearly see if there is a trend to the error based on utilization.

Residual analysis is a powerful tool. The horizontal axis is the Oracle calculated utilization. The vertical axis is the utilization difference (i.e., error or residual) between Oracle (v$osstat based) and the OS (vmstat based). A residual graph makes is very visually easy to see if there are error patterns or trends. An example trend could be, that as the Oracle utilization increases, the difference between the Oracle utilization and OS utilization decreases. If there was no difference between the utilization calculations, the error trend line would be flat, that is, having a slope of zero.

For easy visual comparisons, I only plotted differences up to plus/minus 5% (0.050). This is truly a compro-

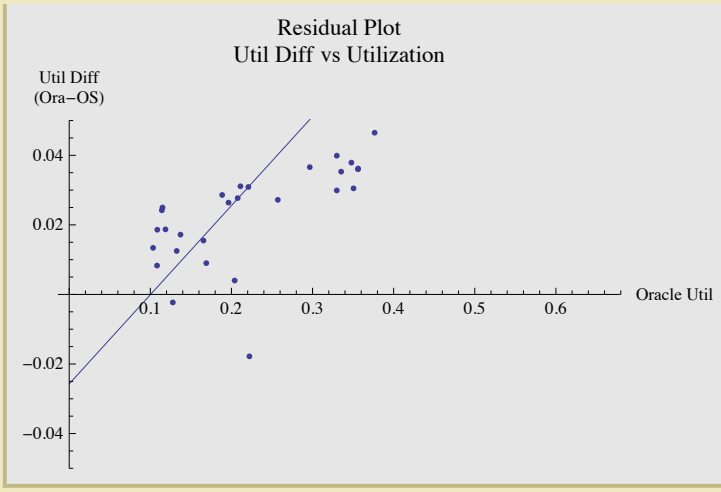
mise because some data sets display better with a wider range (e.g., AG1), most display best with only a 1% plus/minus range.

The trend line equation is also displayed. The slope has the variable *x* after it. A slope of zero means the trend line is flat. A slope of 0.25 means that every increase percent in the Oracle utilization, the difference between the utilization calculations increases by 1/4 of a percent (0.0025), which is extremely small. So while the slope may look shockingly massive, from an impact perspective it may be extremely flat.

In[78]:=
```
yRange = .050;
model = LinearModelFit[UtilError, x, x]
Show[ListPlot[{UtilError}, PlotLabel → "Residual Plot\nUtil Diff vs Utilization",
  AxesLabel → {"Oracle Util", "Util Diff\n(Ora-OS)"},
  AxesOrigin → {0, 0}, PlotRange → {-yRange, yRange}]
 , Plot[model["BestFit"], {x, 0, Max[UtilOracle]}], PlotRange → {-yRange, yRange}]
```

Out[79]=
```
FittedModel[ -0.0257273 + 0.256332 x ]
```

Out[80]=



Experimental Data

Instead of reading data from the actual data collection output files (finalOut.txt), I decided to place all the various samples into the notebook. This allows me to comment on each sample set below and also when making notebook changes...there is only one!

Because I placed the experimental data at the bottom of the notepad, you will must **evaluate the notebook twice** or evaluate the data set first and then evaluate the section of interest.

To keep the data contributors anonymity secret (after all these are production machines), I gave each contributed a code, such as AG. I know their real names, emails, and other pertinent details.

Data Set: AG1

This data set was captured by AG and emailed to Craig on 4-May-2011. It's an AIX 6.1.6.3 box with 16 cores each with 2 threads, so AIX considers there to be 32 CPUs.

In[81]:=
```
AG1UtilOracle = {.3560, .3426, .3505, .4609, .4517, .3941, .3299, .2209,
   .1437, .1142, .1083, .1086, .1150, .1277, .1690, .1325, .1655, .1886,
   .2572, .2966, .3353, .3814, .4786, .5666, .6249, .6585, .6642, .6550,
   .6660, .6472, .6475, .6345, .6534, .6506, .6420, .6483, .6268, .5806, .5721,
   .5254, .4541, .4157, .3887, .3321, .3029, .2970, .1734, .1964, .2040, .2111,
   .2222, .3765, .3562, .3479, .3299, .2077, .1670, .1187, .1372, .1034}
AG1UtilOs = {.3200, .2700, .3200, .3900, .3900, .3300, .2900, .1900, .0900,
   .0900, .1000, .0900, .0900, .1300, .1600, .1200, .1500, .1600, .2300,
   .2600, .3000, .3300, .4200, .4500, .5100, .5100, .5100, .4800, .5100,
   .5000, .4900, .4800, .5000, .5000, .4900, .5000, .4800, .4500, .4300,
   .4100, .3500, .3400, .3200, .2800, .2500, .2200, .1000, .1700, .2000,
   .1800, .2400, .3300, .3200, .3100, .3000, .1800, .1100, .1000, .1200, .0900}
```

Out[81]=
```
{0.356, 0.3426, 0.3505, 0.4609, 0.4517, 0.3941, 0.3299, 0.2209, 0.1437, 0.1142,
 0.1083, 0.1086, 0.115, 0.1277, 0.169, 0.1325, 0.1655, 0.1886, 0.2572, 0.2966,
 0.3353, 0.3814, 0.4786, 0.5666, 0.6249, 0.6585, 0.6642, 0.655, 0.666, 0.6472,
 0.6475, 0.6345, 0.6534, 0.6506, 0.642, 0.6483, 0.6268, 0.5806, 0.5721, 0.5254,
 0.4541, 0.4157, 0.3887, 0.3321, 0.3029, 0.297, 0.1734, 0.1964, 0.204, 0.2111,
 0.2222, 0.3765, 0.3562, 0.3479, 0.3299, 0.2077, 0.167, 0.1187, 0.1372, 0.1034}
```

Out[82]=
```
{0.32, 0.27, 0.32, 0.39, 0.39, 0.33, 0.29, 0.19, 0.09, 0.09, 0.1, 0.09, 0.09, 0.13, 0.16,
 0.12, 0.15, 0.16, 0.23, 0.26, 0.3, 0.33, 0.42, 0.45, 0.51, 0.51, 0.51, 0.48, 0.51, 0.5,
 0.49, 0.48, 0.5, 0.5, 0.49, 0.5, 0.48, 0.45, 0.43, 0.41, 0.35, 0.34, 0.32, 0.28, 0.25,
 0.22, 0.1, 0.17, 0.2, 0.18, 0.24, 0.33, 0.32, 0.31, 0.3, 0.18, 0.11, 0.1, 0.12, 0.09}
```

Data Set: AG2

This data set was captured by AG and emailed to Craig on 4-May-2011. It's an unbranded server running Linux 2.6.9-67.ELsmp 32 bit with 4 cores, each with 4 threads, for a total of 16 threads.

In[83]:=
```
AG2UtilOracle = {.1018, .0992, .1097, .0935, .1003, .0933, .1026, .0986,
   .0905, .0972, .0964, .1252, .1023, .0909, .1112, .1065, .0921, .1084,
   .1272, .1063, .1447, .2068, .3341, .3490, .3410, .1242, .1274, .1051,
   .1132, .1023, .1240, .1049, .1218, .1056, .1081, .1202, .1085, .1045, .1109,
   .0981, .1081, .1065, .1171, .1130, .0983, .0970, .0909, .0836, .0958, .0940,
   .1050, .0926, .1050, .0967, .1100, .0881, .0896, .0970, .0875, .1014}
AG2UtilOs = {.1000, .1000, .1100, .0900, .1000, .0900, .1000, .1000, .0900,
   .1000, .1000, .1300, .1000, .0900, .1100, .1100, .0900, .1100, .1300,
   .1100, .1500, .2100, .3400, .3500, .3500, .1200, .1300, .1000, .1200,
   .1000, .1300, .1000, .1300, .1000, .1100, .1200, .1100, .1000, .1200,
   .1000, .1000, .1000, .1200, .1200, .1000, .1000, .0900, .0900, .0900,
   .0900, .1000, .0900, .1000, .1000, .1200, .0900, .0900, .1000, .0900, .1000}
```

Out[83]=
```
{0.1018, 0.0992, 0.1097, 0.0935, 0.1003, 0.0933, 0.1026, 0.0986, 0.0905, 0.0972,
 0.0964, 0.1252, 0.1023, 0.0909, 0.1112, 0.1065, 0.0921, 0.1084, 0.1272, 0.1063,
 0.1447, 0.2068, 0.3341, 0.349, 0.341, 0.1242, 0.1274, 0.1051, 0.1132, 0.1023,
 0.124, 0.1049, 0.1218, 0.1056, 0.1081, 0.1202, 0.1085, 0.1045, 0.1109, 0.0981,
 0.1081, 0.1065, 0.1171, 0.113, 0.0983, 0.097, 0.0909, 0.0836, 0.0958, 0.094,
 0.105, 0.0926, 0.105, 0.0967, 0.11, 0.0881, 0.0896, 0.097, 0.0875, 0.1014}
```

Out[84]=
```
{0.1, 0.1, 0.11, 0.09, 0.1, 0.09, 0.1, 0.1, 0.09, 0.1, 0.1, 0.13, 0.1, 0.09, 0.11, 0.11,
 0.09, 0.11, 0.13, 0.11, 0.15, 0.21, 0.34, 0.35, 0.35, 0.12, 0.13, 0.1, 0.12, 0.1,
 0.13, 0.1, 0.13, 0.1, 0.11, 0.12, 0.11, 0.1, 0.12, 0.1, 0.1, 0.1, 0.12, 0.12, 0.1,
 0.1, 0.09, 0.09, 0.09, 0.09, 0.1, 0.09, 0.1, 0.1, 0.12, 0.09, 0.09, 0.1, 0.09, 0.1}
```

Data Set: GO1

This data set was captured by GO and emailed to Craig on 3-May-2011. It's an unbranded server running HPUX 10.2.0.4.0 64 bit with 3 cores, each with 2 threads, for a total of 6 threads.

In[85]:=
```
GO1UtilOracle = {.0565, .0481, .0535, .1467, .0502, .0479, .0500, .1508,
   .0457, .0417, .0928, .1217, .0398, .0527, .0678, .0893, .0593, .0508,
   .0573, .0517, .0489, .0585, .0720, .0841, .0790, .1171, .1195, .1925,
   .0826, .0553, .0780, .0888, .0845, .0958, .1319, .1457, .1039, .0917, .1001,
   .1124, .1128, .1035, .1217, .0933, .1621, .1975, .1866, .1994, .1551, .0878,
   .0929, .1278, .1155, .0947, .1211, .1164, .1035, .1223, .1105, .1174}
GO1UtilOs = {.0600, .0500, .0500, .1400, .0500, .0500, .0500, .1600, .0600,
   .0500, .1000, .1200, .0400, .0500, .0700, .0900, .0600, .0500, .0600,
   .0500, .0500, .0600, .0700, .0900, .0800, .1200, .1200, .1900, .0900,
   .0500, .0800, .0900, .0900, .1000, .1300, .1500, .1100, .1000, .1000,
   .1100, .1100, .1000, .1200, .1000, .1600, .2000, .1900, .2100, .1500,
   .0900, .0900, .1300, .1200, .1000, .1200, .1100, .1100, .1200, .1100, .1200}
```

Out[85]=
```
{0.0565, 0.0481, 0.0535, 0.1467, 0.0502, 0.0479, 0.05, 0.1508, 0.0457, 0.0417,
 0.0928, 0.1217, 0.0398, 0.0527, 0.0678, 0.0893, 0.0593, 0.0508, 0.0573, 0.0517,
 0.0489, 0.0585, 0.072, 0.0841, 0.079, 0.1171, 0.1195, 0.1925, 0.0826, 0.0553,
 0.078, 0.0888, 0.0845, 0.0958, 0.1319, 0.1457, 0.1039, 0.0917, 0.1001, 0.1124,
 0.1128, 0.1035, 0.1217, 0.0933, 0.1621, 0.1975, 0.1866, 0.1994, 0.1551, 0.0878,
 0.0929, 0.1278, 0.1155, 0.0947, 0.1211, 0.1164, 0.1035, 0.1223, 0.1105, 0.1174}
```

Out[86]=
```
{0.06, 0.05, 0.05, 0.14, 0.05, 0.05, 0.05, 0.16, 0.06, 0.05, 0.1, 0.12, 0.04, 0.05, 0.07,
 0.09, 0.06, 0.05, 0.06, 0.05, 0.05, 0.06, 0.07, 0.09, 0.08, 0.12, 0.12, 0.19, 0.09, 0.05,
 0.08, 0.09, 0.09, 0.1, 0.13, 0.15, 0.11, 0.1, 0.1, 0.11, 0.11, 0.1, 0.12, 0.1, 0.16,
 0.2, 0.19, 0.21, 0.15, 0.09, 0.09, 0.13, 0.12, 0.1, 0.12, 0.11, 0.11, 0.12, 0.11, 0.12}
```

Data Set: LZ1

This data set was captured by LZ and emailed to Craig on 4-May-2011. It's an HP server running HPUX B.11.31 64 bit with 64 cores. It appears threads are not being used.

In[87]:=
```
LZ1UtilOracle = {.2864, .1745, .1426, .1427, .1504, .1507, .1204, .1170,
   .1246, .1496, .1371, .1356, .1498, .2273, .2826, .3665, .3806, .4375,
   .4468, .4305, .4571, .4711, .4792, .4666, .4438, .2583, .5461, .5669,
   .5527, .5835, .5629, .5758, .5903, .6159, .6034, .6555, .6097, .6348, .6184,
   .5997, .5222, .3183, .3006, .4659, .2522, .2653, .2683, .2703, .3442, .5389,
   .3281, .2853, .2949, .2806, .2817, .3281, .2946, .2837, .2843, .3642}
LZ1UtilOs = {.2900, .1700, .1500, .1400, .1500, .1500, .1200, .1200, .1200,
   .1500, .1400, .1300, .1500, .2200, .2800, .3700, .3800, .4400, .4500,
   .4300, .4600, .4700, .4800, .4700, .4400, .2600, .5500, .5600, .5500,
   .5900, .5700, .5700, .5900, .6200, .6000, .6500, .6100, .6300, .6200,
   .6000, .5200, .3100, .3000, .4600, .2500, .2600, .2700, .2700, .3500,
   .5400, .3300, .2900, .3000, .2800, .2800, .3300, .2900, .2900, .2800, .3600}
```

Out[87]=
```
{0.2864, 0.1745, 0.1426, 0.1427, 0.1504, 0.1507, 0.1204, 0.117, 0.1246, 0.1496,
 0.1371, 0.1356, 0.1498, 0.2273, 0.2826, 0.3665, 0.3806, 0.4375, 0.4468, 0.4305,
 0.4571, 0.4711, 0.4792, 0.4666, 0.4438, 0.2583, 0.5461, 0.5669, 0.5527, 0.5835,
 0.5629, 0.5758, 0.5903, 0.6159, 0.6034, 0.6555, 0.6097, 0.6348, 0.6184, 0.5997,
 0.5222, 0.3183, 0.3006, 0.4659, 0.2522, 0.2653, 0.2683, 0.2703, 0.3442, 0.5389,
 0.3281, 0.2853, 0.2949, 0.2806, 0.2817, 0.3281, 0.2946, 0.2837, 0.2843, 0.3642}
```

Out[88]=
```
{0.29, 0.17, 0.15, 0.14, 0.15, 0.15, 0.12, 0.12, 0.12, 0.15, 0.14, 0.13, 0.15, 0.22, 0.28,
 0.37, 0.38, 0.44, 0.45, 0.43, 0.46, 0.47, 0.48, 0.47, 0.44, 0.26, 0.55, 0.56, 0.55, 0.59,
 0.57, 0.57, 0.59, 0.62, 0.6, 0.65, 0.61, 0.63, 0.62, 0.6, 0.52, 0.31, 0.3, 0.46, 0.25,
 0.26, 0.27, 0.27, 0.35, 0.54, 0.33, 0.29, 0.3, 0.28, 0.28, 0.33, 0.29, 0.29, 0.28, 0.36}
```

Data Set: LZ2

This data set was captured by LZ and emailed to Craig on 4-May-2011. It's an HP server running HPUX B.11.31 64 bit with 64 cores. It appears threads are not being used.

In[89]:=
```
LZ2UtilOracle = {.0335, .0445, .1084, .1252, .0345, .0343, .1620, .0695,
   .0324, .0322, .0327, .0331, .0393, .1238, .1550, .1482, .1169, .1248,
   .1129, .0896, .0913, .0879, .0703, .0624, .0681, .0354, .0764, .0840,
   .0704, .0717, .0629, .0567, .0598, .0545, .0469, .0499, .1418, .1348, .0769,
   .1423, .0477, .0453, .0447, .0443, .0440, .0470, .0513, .0505, .0499, .1135,
   .0543, .0545, .0573, .0536, .0539, .0551, .0537, .0506, .0486, .0453}
LZ2UtilOs = {.0400, .0400, .1100, .1300, .0400, .0400, .1600, .0700, .0300,
   .0300, .0300, .0300, .0400, .1200, .1600, .1400, .1200, .1200, .1100,
   .0900, .0900, .0900, .0700, .0700, .0700, .0300, .0700, .0800, .0700,
   .0700, .0600, .0500, .0600, .0500, .0500, .0500, .1400, .1300, .0700,
   .1400, .0500, .0400, .0400, .0400, .0400, .0400, .0500, .0500, .0500,
   .1200, .0500, .0500, .0500, .0500, .0500, .0500, .0500, .0500, .0500, .0400}
```

Out[89]=
```
{0.0335, 0.0445, 0.1084, 0.1252, 0.0345, 0.0343, 0.162, 0.0695, 0.0324, 0.0322,
 0.0327, 0.0331, 0.0393, 0.1238, 0.155, 0.1482, 0.1169, 0.1248, 0.1129, 0.0896,
 0.0913, 0.0879, 0.0703, 0.0624, 0.0681, 0.0354, 0.0764, 0.084, 0.0704, 0.0717,
 0.0629, 0.0567, 0.0598, 0.0545, 0.0469, 0.0499, 0.1418, 0.1348, 0.0769, 0.1423,
 0.0477, 0.0453, 0.0447, 0.0443, 0.044, 0.047, 0.0513, 0.0505, 0.0499, 0.1135,
 0.0543, 0.0545, 0.0573, 0.0536, 0.0539, 0.0551, 0.0537, 0.0506, 0.0486, 0.0453}
```

Out[90]=
```
{0.04, 0.04, 0.11, 0.13, 0.04, 0.04, 0.16, 0.07, 0.03, 0.03, 0.03, 0.03, 0.04, 0.12, 0.16,
 0.14, 0.12, 0.12, 0.11, 0.09, 0.09, 0.09, 0.07, 0.07, 0.07, 0.03, 0.07, 0.08, 0.07, 0.07,
 0.06, 0.05, 0.06, 0.05, 0.05, 0.05, 0.14, 0.13, 0.07, 0.14, 0.05, 0.04, 0.04, 0.04, 0.04,
 0.04, 0.05, 0.05, 0.05, 0.12, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.04}
```

Data Set: NN1

This data set was captured by NN and emailed to Craig on 4-May-2011. It's an Solaris (UltraSPARC T1 server) with 4 cores, each with 2 threads, for a total of 8 threads.

In[91]:=
```
NN1UtilOracle = {.2622, .2954, .3093, .2882, .3229, .2782, .3112, .2771,
   .3065, .2292, .2524, .2465, .2767, .2717, .2836, .2985, .3684, .2904,
   .2978, .2664, .3516, .3902, .3468, .3362, .3557, .4173, .3968, .3162,
   .3199, .3040, .3586, .3493, .3795, .3982, .3070, .2515, .2865, .2508, .3389,
   .2329, .2065, .1925, .2679, .2611, .2374, .2293, .2240, .2208, .2487, .2864,
   .2499, .2478, .2473, .1967, .3250, .4378, .1984, .1793, .1857, .2629}
NN1UtilOs = {.2600, .2900, .3100, .2900, .3200, .2800, .3100, .2800, .3000,
   .2300, .2500, .2500, .2700, .2700, .2800, .3000, .3700, .2900, .3000,
   .2600, .3500, .3900, .3500, .3300, .3600, .4200, .4000, .3200, .3200,
   .3000, .3600, .3400, .3800, .3900, .3000, .2500, .2900, .2500, .3400,
   .2400, .2100, .1900, .2700, .2600, .2300, .2300, .2200, .2200, .2500,
   .2800, .2500, .2500, .2500, .1900, .3200, .4300, .2000, .1800, .1800, .2600}
```

Out[91]=
```
{0.2622, 0.2954, 0.3093, 0.2882, 0.3229, 0.2782, 0.3112, 0.2771, 0.3065, 0.2292,
 0.2524, 0.2465, 0.2767, 0.2717, 0.2836, 0.2985, 0.3684, 0.2904, 0.2978, 0.2664,
 0.3516, 0.3902, 0.3468, 0.3362, 0.3557, 0.4173, 0.3968, 0.3162, 0.3199, 0.304,
 0.3586, 0.3493, 0.3795, 0.3982, 0.307, 0.2515, 0.2865, 0.2508, 0.3389, 0.2329,
 0.2065, 0.1925, 0.2679, 0.2611, 0.2374, 0.2293, 0.224, 0.2208, 0.2487, 0.2864,
 0.2499, 0.2478, 0.2473, 0.1967, 0.325, 0.4378, 0.1984, 0.1793, 0.1857, 0.2629}
```

Out[92]=
```
{0.26, 0.29, 0.31, 0.29, 0.32, 0.28, 0.31, 0.28, 0.3, 0.23, 0.25, 0.25, 0.27, 0.27, 0.28,
 0.3, 0.37, 0.29, 0.3, 0.26, 0.35, 0.39, 0.35, 0.33, 0.36, 0.42, 0.4, 0.32, 0.32, 0.3,
 0.36, 0.34, 0.38, 0.39, 0.3, 0.25, 0.29, 0.25, 0.34, 0.24, 0.21, 0.19, 0.27, 0.26, 0.23,
 0.23, 0.22, 0.22, 0.25, 0.28, 0.25, 0.25, 0.25, 0.19, 0.32, 0.43, 0.2, 0.18, 0.18, 0.26}
```

Data Set: RB1

This data set was captured by RB and emailed to Craig on 29-April-2011. It's an Red Hat Enterprise Linux Server release 5.6 (Tikanga) 2.6.18 238.1.1.el5 (64-bit) with 8 cores (Quad-Core AMD Opteron(tm) Proces-

sor 2382) and we're not sure how many threads there are or threads are being used.

In[93]:=
```
RB1UtilOracle = {.0642, .0747, .0943, .0495, .0705, .0728, .0724, .0622,
   .0552, .0475, .0471, .0386, .0401, .0376, .0436, .0526, .0622, .0519,
   .0673, .0558, .0610, .0496, .0700, .0633, .0626, .0615, .0584, .0323,
   .0864, .1582, .1478, .1474, .1424, .1403, .0989, .0154, .0137, .0142, .0138,
   .0141, .0142, .0133, .0180, .0163, .0822, .0599, .0170, .0156, .0448, .0141,
   .0221, .0228, .0172, .0157, .0162, .0176, .0911, .0250, .0523, .1384}
RB1UtilOs = {.0700, .0700, .0900, .0500, .0700, .0700, .0700, .0600, .0500,
   .0500, .0500, .0400, .0400, .0400, .0400, .0500, .0600, .0500, .0600,
   .0500, .0600, .0500, .0700, .0600, .0700, .0600, .0500, .0300, .0900,
   .1600, .1500, .1500, .1500, .1400, .1000, .0200, .0200, .0200, .0200,
   .0200, .0200, .0200, .0200, .0200, .0900, .0600, .0200, .0200, .0500,
   .0200, .0300, .0300, .0200, .0200, .0200, .0200, .0900, .0300, .0500, .1400}
```

Out[93]=
```
{0.0642, 0.0747, 0.0943, 0.0495, 0.0705, 0.0728, 0.0724, 0.0622, 0.0552, 0.0475,
 0.0471, 0.0386, 0.0401, 0.0376, 0.0436, 0.0526, 0.0622, 0.0519, 0.0673, 0.0558,
 0.061, 0.0496, 0.07, 0.0633, 0.0626, 0.0615, 0.0584, 0.0323, 0.0864, 0.1582,
 0.1478, 0.1474, 0.1424, 0.1403, 0.0989, 0.0154, 0.0137, 0.0142, 0.0138, 0.0141,
 0.0142, 0.0133, 0.018, 0.0163, 0.0822, 0.0599, 0.017, 0.0156, 0.0448, 0.0141,
 0.0221, 0.0228, 0.0172, 0.0157, 0.0162, 0.0176, 0.0911, 0.025, 0.0523, 0.1384}
```

Out[94]=
```
{0.07, 0.07, 0.09, 0.05, 0.07, 0.07, 0.07, 0.06, 0.05, 0.05, 0.05, 0.04, 0.04, 0.04, 0.04,
 0.05, 0.06, 0.05, 0.06, 0.05, 0.06, 0.05, 0.07, 0.06, 0.07, 0.06, 0.05, 0.03, 0.09, 0.16,
 0.15, 0.15, 0.15, 0.14, 0.1, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.09,
 0.06, 0.02, 0.02, 0.05, 0.02, 0.03, 0.03, 0.02, 0.02, 0.02, 0.02, 0.09, 0.03, 0.05, 0.14}
```