Making Statistical Sense out of Time Series Data with “Home Grown” Perl Scripts

July, 2010
by James F. Brady

About the Author

James F Brady

Jim has worked over 30 years in the telecommunications and computer industries for GTE, Tandem Computers, Siemens, and currently is the Capacity Planner for the State Of Nevada. At GTE he worked in both Data Center Capacity Planning and Digital Switching Traffic Capacity determination. While at Siemens he obtained EU and US patents for a traffic overload control mechanism used in multiple products including a VoIP Switch. He holds BS and MS degrees in Operations Research from The Ohio State University.

Introduction

Often the analyst has time series data available www.en.wikipedia.org/wiki/Time_series and needs to make statistical sense out of it for capacity planning purposes or to better understand a system’s performance characteristics. For example, a twenty-four hour frequency plot produced by averaging several days of time series data provides a statistically stable representation of a resource’s traffic handling characteristics and facilitates easy identification of the busy period during the day.

This paper describes an approach to accomplishing this data transformation using “home grown” Perl scripts, www.perl.com, to process the time series data in a manner that converts it from the time to the frequency domain producing the desired statistical orientation in graphical and text format. The three examples that follow focus on illustrating these frequency distribution results with only cursory comments being made regarding the time series data file manipulations performed in the Perl environment used to complete the task.

AIX Unix and Linux Nmon Example

Time series data is usually collected for system monitoring purposes at fixed sampling intervals with a focus on status reporting and troubleshooting. Figure 1 shows a typical time series graph where number of processors used is plotted as a function of time for a twenty-four hour period.

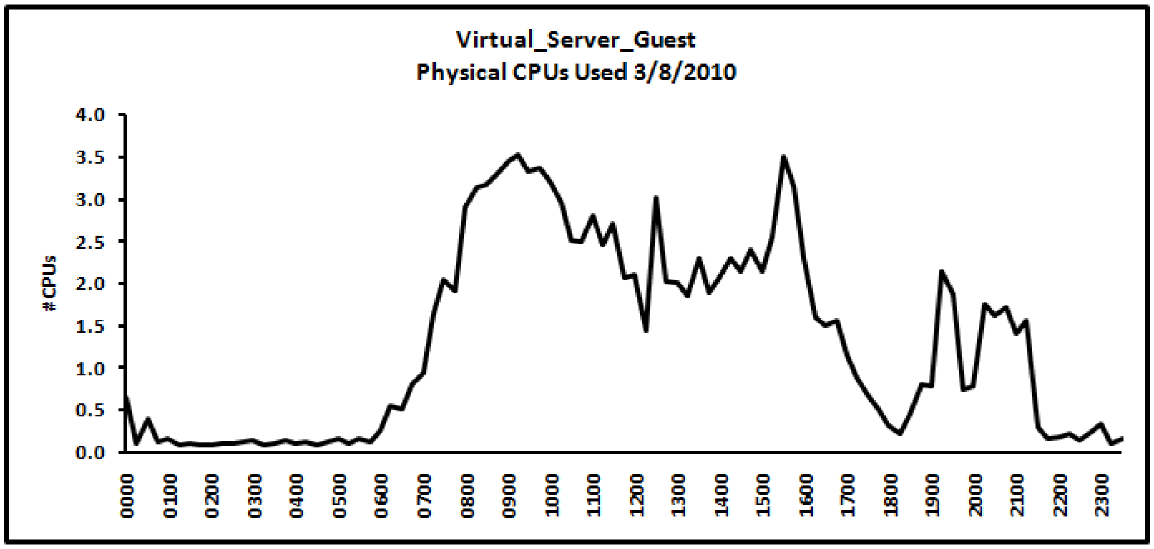

Figure 1: Time Series – CPUs Used Vs Time

This illustration provides useful information regarding CPUs used for the specific day represented, but does not yield sufficient insight into CPU consumption for capacity planning purposes. Figure 2 supports this argument with a second day of data for the same CPU environment one week earlier by showing the variation that occurs even when both graphs represent the same day of week.

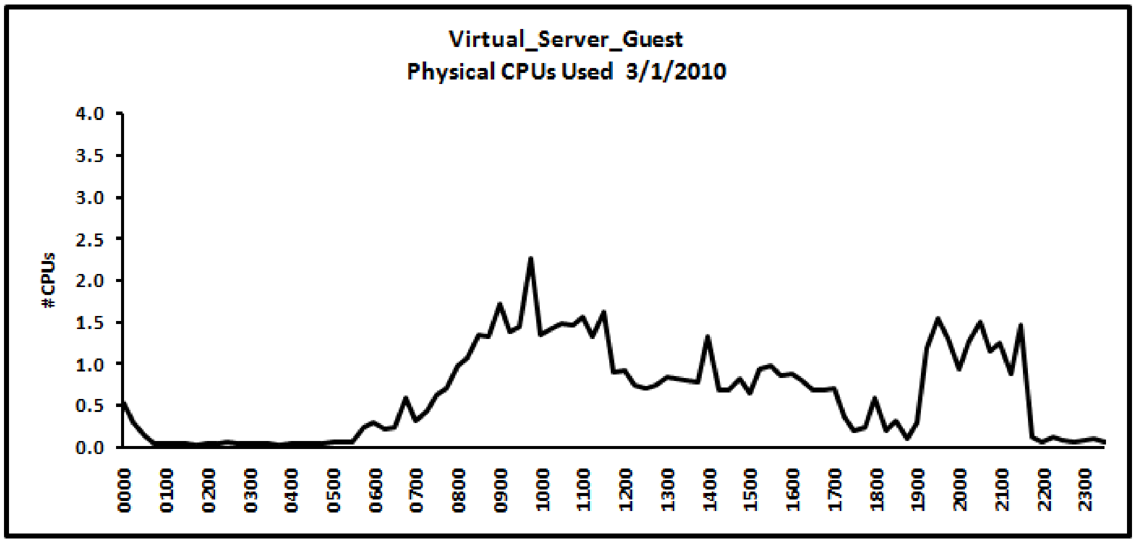

Figure 2: Time Series – CPUs Used Vs Time for Day 2

Given the differences between these two samples, more daily data is required to provide a statistically representative view of CPUs used across the day. Figure 3 is such a capacity planning oriented graph showing the average number of CPUs used on a twenty-four hour basis for all weekdays in March 2010. It is produced with a Perl script which processes a month of daily resource utilization files and constructs M-F results in this bar chart format along with comma separated value (csv) listings for spreadsheet import (not shown).

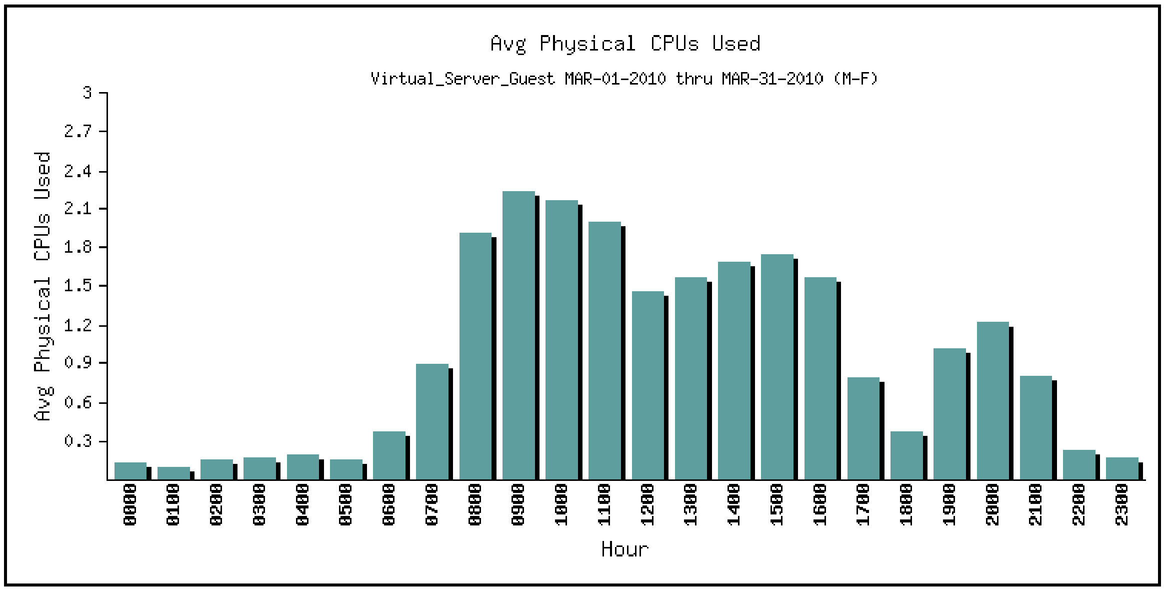

Figure 3: Average CPUs Used for March 2010.

Figure 4 contains the peak hour statistics for the Figure 3 time period and represents a rudimentary indication of data dispersion from the average since it displays the highest hour observed for all days sampled. Figure 3 shows approximately 2.2 physical processors are consumed during the average busy hour for the month (9:00 AM – 10:00 AM) while the peak for the same hour of the day is 4.1 CPUs.

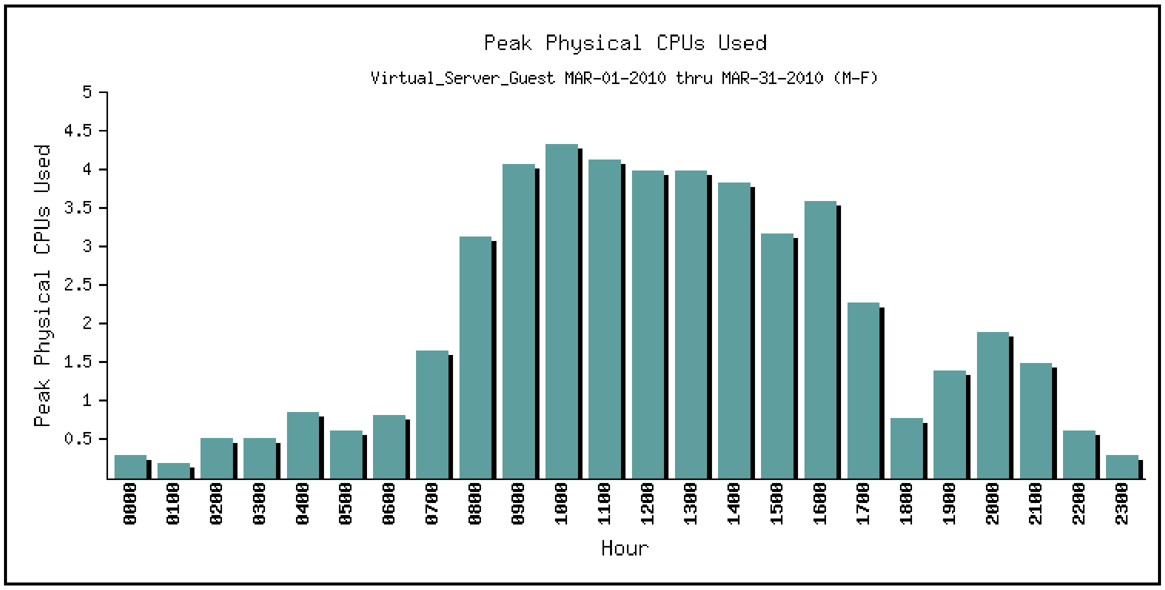

Figure 4: Peak CPUs Used for March 2010.

The automated data collection software that generates the input data for Figure 1 through 4 is Nmon, a free tool used to analyze AIX Unix and Linux performance on IBM pSeries systems, http://www.ibm.com/developerworks/aix/library/au-analyze_aix . Nmon produces graphs and daily data files based on a parameter driven sampling interval which in this case is set to fifteen minutes. Individual daily files generated by this data collector can be imported into a companion spreadsheet tool called the nmon analyser, http://www.ibm.com/developerworks/aix/library/au-nmon_analyser, and the results viewed as illustrated in Figure 1 and 2.

Network Circuit Example

Another resource that requires the application of statistical analysis techniques for proper capacity planning is data network circuits. Figure 5 is a time series chart produced by a popular data network analysis tool representing % Bandwidth Used by a particular circuit during the last four weeks in March 2010 on a one hour sampling interval basis.

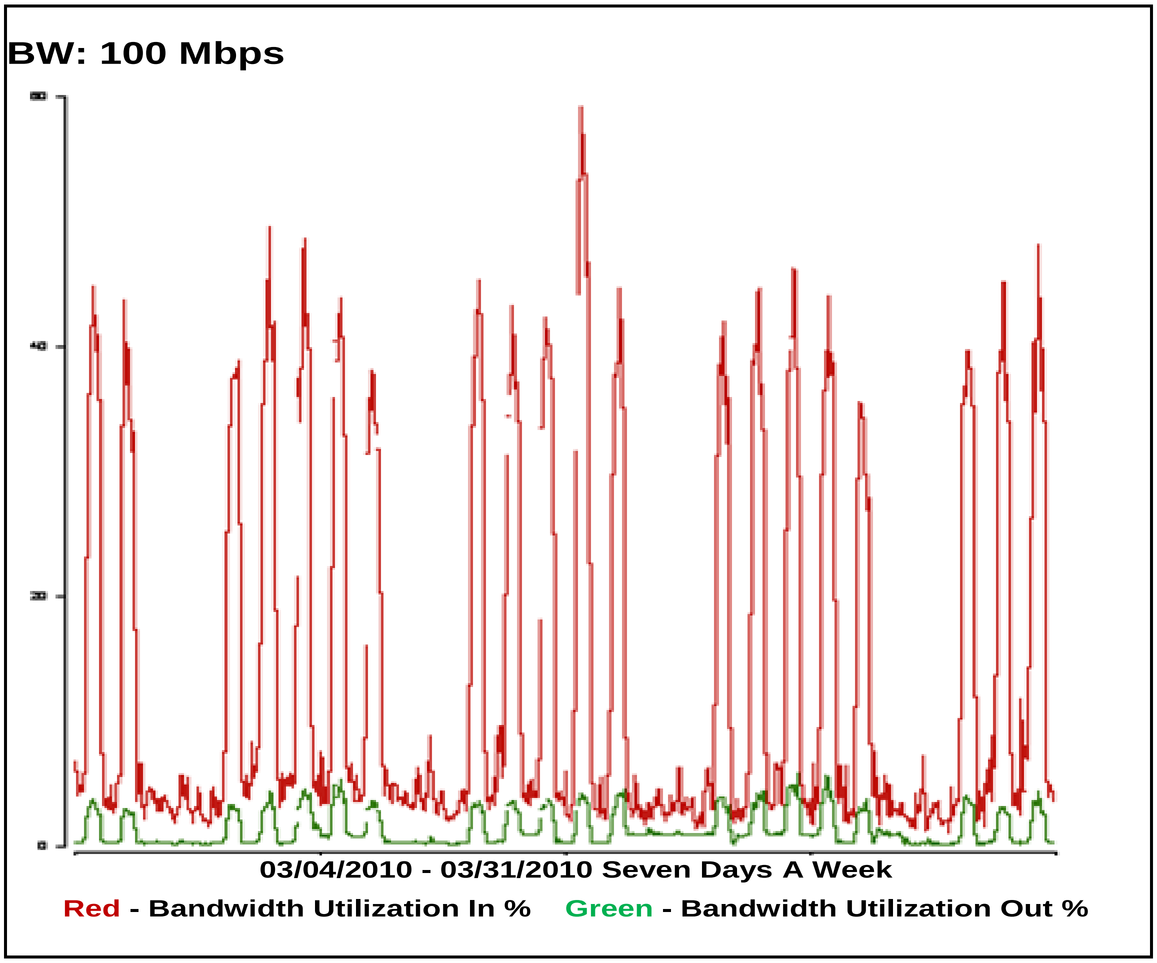

Figure 5: Time Series - Network Circuit % Bandwidth Used For March 2010

The key statistic of interest from a capacity planning perspective is busy hour bandwidth utilization based on a significant sample size. Figure 5 and its companion csv file contain enough data for this purpose, i.e., four weeks of hourly samples, but the data is not represented in a statistically meaningful way. Identifying the busy hour of the day is difficult from this time series orientation and the weekend data makes statistical manipulations complex.

Figure 6 and 7 provide the needed statistical clarity. They show this circuit to be “Prime Time” oriented (8:00 AM – 5:00 PM) with a busy hour average of around 43% occurring from 11:00 AM to 12:00 PM for the incoming (dominant direction) traffic. Figure 7 provides a rudimentary indication of data dispersion from the average in the same manner that Figure 4 does for Figure 3.

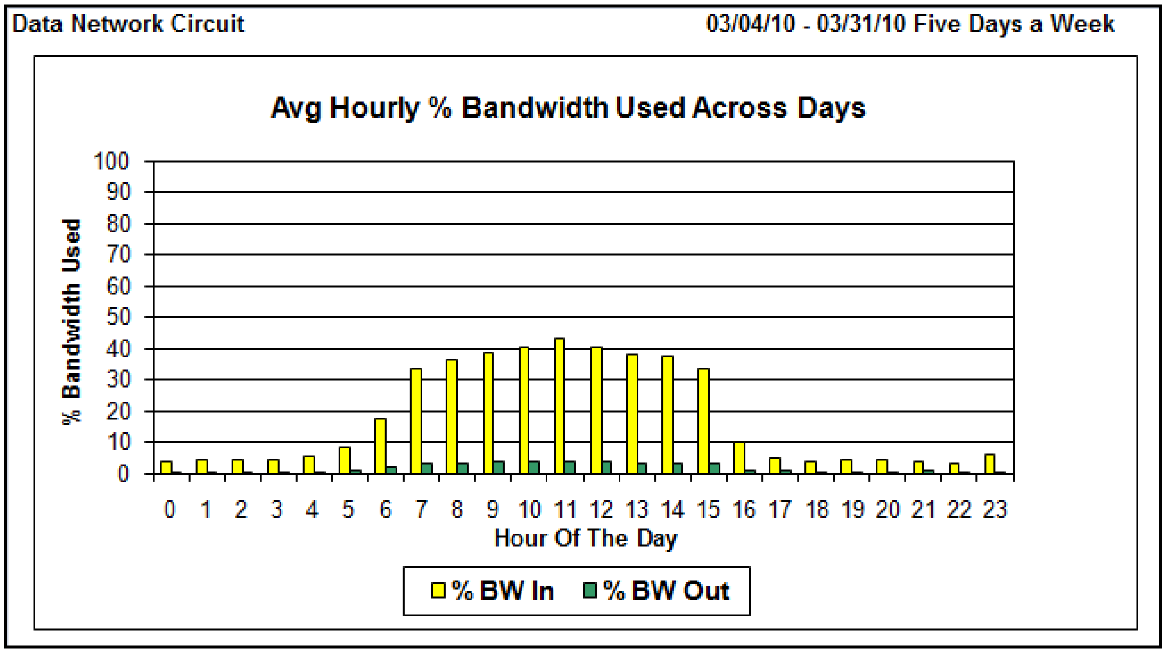

Figure 6: Average Hourly % Bandwidth Used for March 2010 (M-F)

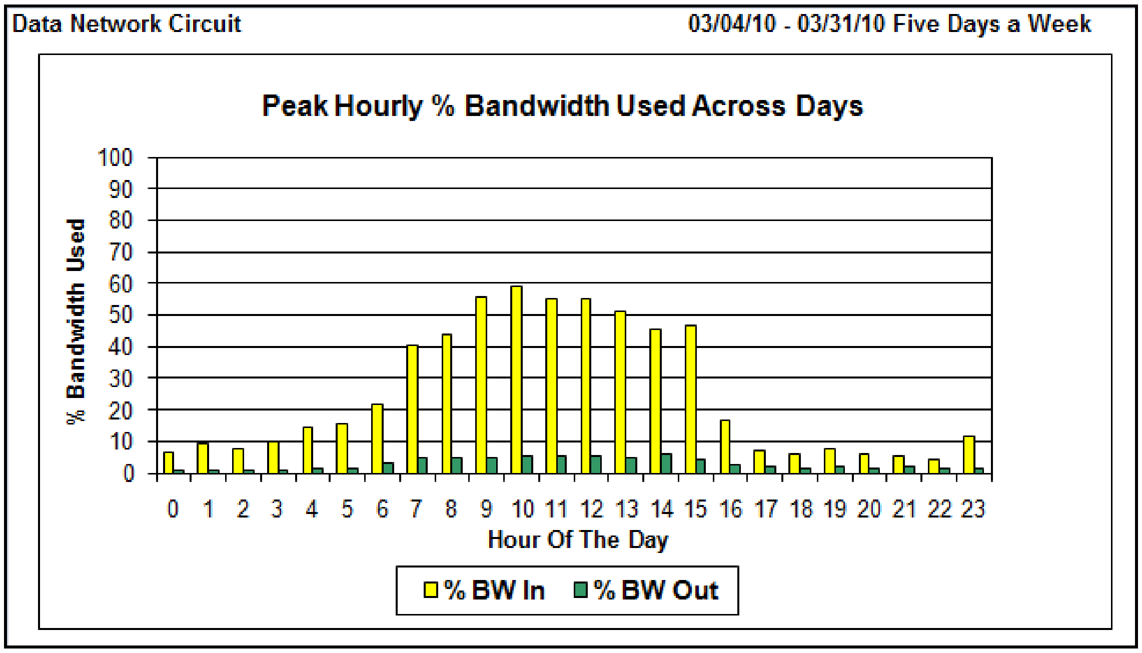

Figure 7: Peak Hourly % Bandwidth Used for March 2010 (M-F)

Figure 6 and 7 are produced using a structured spreadsheet and a Perl script. The Perl script reads the network analysis tool's csv file containing a row of data for each hour interval sampled, extracts the weekday data, reorients that data to match the spreadsheet layout, and creates a csv file whose contents are imported into a spreadsheet set up to calculate statistics and display the two graphs. The Perl script produces a set of graphical bar charts in png format as well (not shown) that are structured like Figure 6 and 7.

Linux Guest Performance Characteristics Example

Sometimes the focus of the time series data manipulation is to gain an understanding of a system's performance characteristics instead of for capacity planning purposes. Figure 8 represents an example of this focus which takes advantage of Perl's prevalence on Unix/Linux Systems. Displayed are graphs of Linux Guest resource consumption levels for several contiguous hours of a specific day. These graphs are produced by processing vmstat, iostat, netstat, and process status (ps) output files with a set of "stat" specific Perl scripts. These output files are generated hourly by running a Perl script that executes the Unix/Linux utilities at approximately one minute sampling intervals organizing the results in hour of day files, e.g., 0900.vmstat is 09:00 AM – 10:00 AM, and placing them in day of week directories, mon, tue, etc. The process status, ps, utility executes at the start and end of each hour to generate the CPU Process Percent graph.

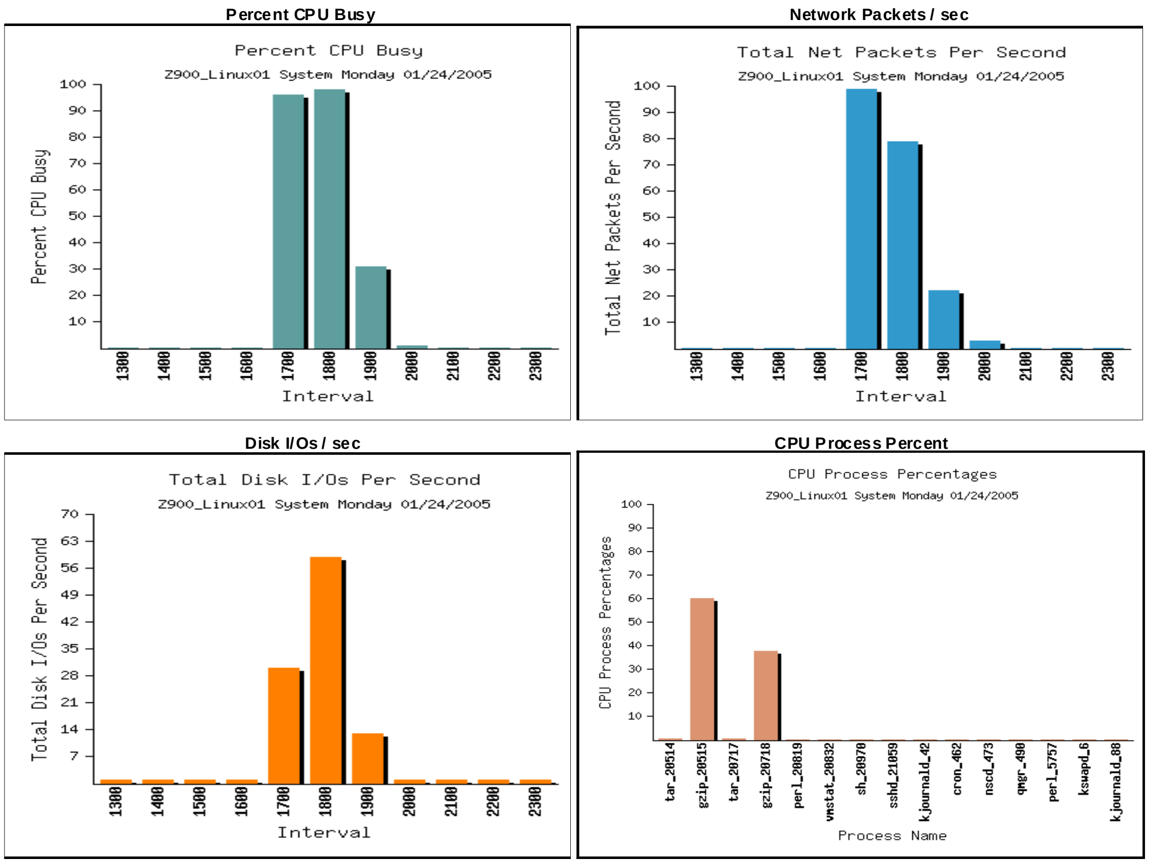

Figure 8: Linux Guest Resource Utilization Statistics

The Figure 8 graphs are extracted from an article that appears in the Fall 2005 CMG Journal [Bra05]. This article describes the behavior of one virtualized environment when contention for physical CPU resources caused significantly long delays in processing to occur.

Summary

This paper contains three examples of how to make statistical sense out of time series data using “home grown” Perl scripts. The general approach is to process one or more time series text files with a user developed Perl script structured to create bar charts and generate comma separated files for spreadsheet import. These Perl script outputs represent statistical aggregations of time series data for capacity planning purposes or system performance characterization requirements. For details regarding the structure of the Perl scripts used to perform the manipulations illustrated in these examples contact this author, jfbrady@doit.nv.gov.

There are a number of tools which can be used to perform this data aggregation task. This author has chosen Perl for this purpose because of its data parsing capabilities, its inclusion in Unix/Linux operating system distributions, and its free user community edition.

The methodology summarized above is consistent with Dr. Neil Gunther’s Guerilla techniques which emphasize tactical approaches to planning focused on opportunity and the use of lightweight tools such as spreadsheets [Gun07]. This author believes that Perl scripts, like those discussed in this article, should be included in this list of lightweight tools as well.

References

[Bra05] J. Brady, “Virtualization and CPU wait times in a Linux guest environment”, CMG Journal, 116:3:8, Fall 2005.

[Gun07] N.J. Gunther, “Guerrilla Capacity Planning”, Springer-Verlag, Berlin Heidelberg, 2007.

Copyrights and Trademarks