Newcastle University
Computer Science

Effective power consumption analysis of desktop computers

Troy Astarte, BSc Computing Science
11010935
t.astarte@gmail.com | t.astarte@ncl.ac.uk

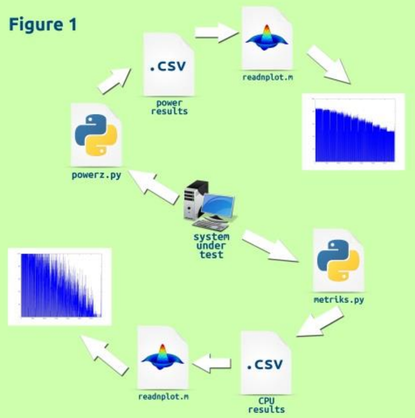
Supervisors: Dr Stephen McGough, Matthew Forshaw
Digital Institute, Systems research group
stephen.mcgough@ncl.ac.uk, m.j.forshaw@ncl.ac.uk

Introduction

As the processing power of computers grows, so too does their consumption of electrical power. It has become necessary to find ways to limit the power consumption of computing - not only for environmental reasons, but as current top-end supercomputers take such huge amounts of power, in order to create machines orders of magnitude faster, serious efficiency measures will be needed.

It is impossible to find ways to successfully reduce power consumption without good data on present power consumption; to that end, I spent my eight weeks investigating some of the tools used to measure power consumption, and creating some extra tools to help the process of gathering relevant, useful power consumption data.

Figure 1



Methods

I decided to use the SPECpower_ssj2008 (referred to hereafter as SPECpower) tool, as it stresses a computer (referred to as system under test or SUT) to 100% of its capability for five minutes, then 90%, and so on. This allows data to be gathered showing how the power profile of a computer changes with its load level, as those with good performance at high load may not be so efficient at lower load levels. [1][2]

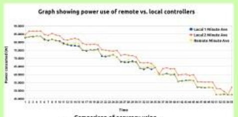
SPECpower supports a variety of USB temperature and power sensors, but these are all prohibitively expensive, so I used Plugwise power monitors[3] which are plugged between a computer and its power socket. As the provided software does not provide sufficiently high granularity (one hour minumum), I utilised plugwiser[4] to poll the monitors. This did not allow for iterative polling, so I wrote a wrapper script, named **`powerz.py`**, which polls a specified list of Plugwise units every second and writes the time and current power to a file.

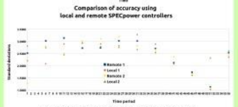
Another set of information in which I was interested is the hardware utilisation information of the SUT, so I wrote a python script based on the **`psutil`** module[5] which gathers system utilisation (CPU, network interface, memory, disk) metrics every second, and writes them to an output file. It is named **`metriks.py`**. The results produced by both **`metriks.py`** and **`powerz.py`** are in .csv format, and to produce a handy overview of them I wrote an Octave (GNU equivalent of Matlab) script to plot from these files, which I have called **`readnPlot.m`**.

At first, I used two different computers for the controller and test modules, as recommended by SPEC, but as this takes up two computers at once, I performed a test to see if running a local controller adversely affects results at all, and it does not. See results for local vs. remote controller test. Therefore, test methodology is to plug the computer into a Plugwise monitor, set up SPECpower, set up the **`metriks.py`** and **`powerz.py`** scripts to gather data, and run SPECpower. Once finished, cease collection of data, and use **`readnPlot.m`** to plot graphs for quick examination of results; spreadsheets may be used for more indepth analysis later. See figure one, presented left.

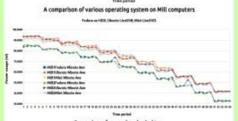
Results

As can be seen from the graphs right, there is no appreciable difference to using a local controller - in fact, there is more deviation between two remote controlled runs on the same machine than remote and local on the same machine.

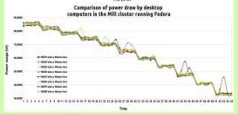




The results of varying the operating system on the same machine can be seen to the right - notice how much more power is consumed by Live operating systems.



Finally, the results of eight different computers running the same operating system in the same location are presented bottom right.



Conclusions

There are a few interesting points raised by this study. Firstly, it is noteworthy that SPECpower will run just as well with a local controller; this should make further testing done on this topic easier to manage. The utility scripts described in the method should also be useful for future researchers.

Secondly, despite the eight Mill computers having identical hardware and software, they have noticeably different power signatures. They have been treated roughly the same during their tenure in the cluster, so reasons for these differences warrant further research.

Finally, I believe the most interesting result is the huge (10W) increase in power consumption when using live media. This is more than could be expected from a typical optical or USB drive, and is curiously static across load levels. Further research here could be very useful in the live OS field.

References
[1] http://www.spec.org/power_ssj2008/
[2] http://impact.asu.edu/cse591sp11/Barroso07_EnergyProp-clean.pdf
[3] http://www.plugwise.com/idplugtype-g/home-basic
[4] https://github.com/infernix/plugwiser
[5] https://code.google.com/p/psutil/