Resource Usage of Windows Computer Laboratories

Patricio Domingues
ESTG – Leiria – Portugal
patricio@estg.ipleiria.pt

Paulo Marques
Univ. Coimbra – Portugal
pmarques@dei.uc.pt

Luis Silva
Univ. Coimbra – Portugal
luis@dei.uc.pt

Abstract

Studies focusing on Unix have shown that the vast majority of workstations and desktop computers remain idle for most of the time. In this paper we quantify the usage of main resources (CPU, main memory, disk space and network bandwidth) of Windows 2000 machines from classroom laboratories. For that purpose, 169 machines of 11 classroom laboratories were monitored over 77 consecutive days. Samples were collected from all machines every 15 minutes for a total of 583653 samples.

Besides evaluating availability of machines (uptime and downtime) and usage habits of users, the paper assesses usage of main resources, focusing on the impact of interactive login sessions over resource consumptions. Also, resorting to Self Monitoring Analysis and Reporting Technology (SMART) parameters of hard disks, the study estimates the average uptime per hard drive power cycle for the whole life of monitored computers.

Our results show that resources idleness in classroom computers is very high, with an average CPU idleness of 97.9%, unused memory averaging 42.1% and unused disk space of the order of gigabytes per machine. Moreover, this study confirms the 2:1 equivalence rule found out by similar works, with N non-dedicated resources delivering an average CPU computing power roughly similar to N/2 dedicated machines. These results confirm the potentiality of these systems for resource harvesting, especially for grid desktop computing schemes.

1. Introduction

Today, academic institutions frequently have large dozens of personal computers (PCs) in their classrooms and laboratories, with a large percentage of these PCs devoted mostly to teaching activities. However, it is known that much of this computing power simply goes unused. In fact, considering that these PCs are only used during extended office work, say from 8.00 am to 8.00 pm on weekday, this means that more than half of the time these machines are simply unused. If we sum up the unexploited idle CPU cycles we could count a considerable computing power, available to the companies, academia and institutions at free-costs. In fact, numerous environments exist to harness idle resources. Examples range from academic projects such as Bayanihan [1], BOINC [2], Condor [3] and XtremWeb [4] to commercial solutions like Entropia [5] and United Devices [6]. Recently some desktop grid schemes have emerged relying on virtual machines [7]. Examples include Virtual Cluster [8] and Non-Dedicated Distributed Environment (NDDE) [9].

PCs of classrooms have an attractive characteristic for resource harvesting: no individual user owns the machines. On the contrary, an office computer is generally assigned to an individual user who controls it, being very suspicious about possible invasion of the machine by foreign programs. Thus, stealing schemes need to deal with social issues beside engineering and technology [10]. Since classroom computers have no individual owner social issues involving their use in resource harvesting are weaker. In fact, Gupta et al. [11] analyze the relation between resource borrowing and interactive usage comfort concluding that resource stealing schemes can be quite aggressive without disturbing user comfort, particularly in the case of memory and disk.

Our main motivation for this study was to characterize the availability and pattern usage of academic classroom computers, quantifying the portion of important resources such as CPU, RAM, disk space and network bandwidth that is left unused. In particular, we were interested in differentiating resource usage according to the existence of interactive login sessions, and their impact on resource usage.

The remainder of this paper is organized as follows. Section 2 describes related work, while section 3 describes our monitoring methodology. Section 4 presents the monitoring experiment, with results being discussed in section 5. Finally, section 6 concludes the paper.

2. Motivation

Evaluation of computer resources usage has been a research topic since the emergence of networked computers in the late 80's. In fact, soon it was noticed that computer resources, noticeably CPU, were frequently underused, especially in machines primarily used for tasks dependent on human interaction, such as text processing or spreadsheet computations. Several studies have assessed the high level of resources idleness in networked computers, not only about CPU [12] [13] but also memory [14] and disk storage [15].

Through simulation Arpaci et al. [12] study the interaction of sequential and parallel workloads. They conclude that a 2:1 rule applies, meaning that N non-dedicated machines are roughly equivalent to N/2 dedicated machines. Interestingly, a 2:1 equivalence ratio was also observed in our set of machines.

The study presented in [13] focuses on the potentiality of using non-dedicated Solaris machines to execute parallel tasks in otherwise idle times, evaluating the machines availability and stability based upon a 14–day trace of computers primarily assigned to undergraduate students. The authors conclude that reasonably large idle clusters are available half the time noting, however, that such set of machines are not particularly stable, exhibiting frequent reboots.

Acharya and Setia [14] analyze main memory idleness and assesses its potential utility resorting to a two-week memory usage trace from two sets of Solaris workstations. One set includes machines fitted with a high amount of main memory (total of 5.2 GB for 29 machines), while the other set is more modest (total of 1.4 GB for 23 machines). The study shows that, on average, idle machines present around 50% of unused main memory.

Ryu et al. in [16] aims to harvest idle resources from what they define as non-idle machines, that is, machines that are lightly loaded by interactive usage. They conclude that a vast set of idle resources can be harvested without interfering on interactive users. However, their methodology requires modification at the kernel level and thus seems impractical for closed operating systems like Windows.

All of the aforementioned cited studies focus on UNIX environments and rely on somewhat reduced traces to draw their conclusions. Our work targets Windows machines, monitoring a medium-sized set of machines over a relatively long period of time (11 consecutive weeks).

Bolosky et al. [15] study a vast set of Windows machines, reporting availability, CPU load and file system usage in a corporate desktop environment. The study is oriented toward the demonstration of the viability of a serveless distributed file system. Thus issues such as main memory load, network usage and interactive sessions and its impact over resource usage are not reported. In contrast, our work focuses on categorizing all main resources usage.

Heap [17] studied resource usage of Unix and Windows servers, through 15-minute periodic gathering of monitoring data. The study found out that Windows servers had a CPU idleness average near 95%, while Unix servers averaged 85% CPU idleness.

The study of Kondo et al. [18] evaluates CPU availability from the perspective of grid desktop computing, running a benchmark probe as a regular task in the grid desktop computing system Entropia. Since the task only gets scheduled at Entropia clients when required idleness threshold conditions are met, this methodology measures effective resources availability from the perspective of a grid desktop system. A drawback of this approach is that the analysis is dependent on the used grid desktop system.

Our approach is distinct from previous works by focusing on academic classrooms fitted with Windows desktop machines. The workload traces were collected for 11 weeks over a medium-sized set of 169 Pentium III and Pentium 4 desktop machines. An analysis of main resource usage is conducted with emphasis in differentiating resource usage between machines occupied with interactive users and free machines.

We also present a novel approach to assess machines availability, combining collected samples with data extracted from the SMART counters of machines' hard disks, namely the *power on hour counts* and *power on cycle* [19] counters. Coupled with the collected traces, these SMART values permit to infer about machines power on pattern allowing a rough estimation of machines availability.

3. Overview

Our monitoring methodology resorts on periodically probing the remote machines. Every 15 minutes an attempt is made to perform a remote execution of a software probe (W32Probe) sequentially over the whole set of machines. W32Probe is a simple win32 console application that outputs, via standard output (*stdout*), several metrics such machine's uptime, CPU time consumed by the idle thread of the operating system since machine boot-up, existence of an interactive user session, amongst other metrics (see section 3.1).

To automate the periodic data collection over the surveyed machines we developed a framework, named Distributed Data Collector (DDC) [20], to support remote data collection in local area networked Windows machines. The framework aims to cover the needs that arise in distributed data collection at the scale of local area networks.

The remote probing solution was chosen because it avoided the installation of software in remote nodes, thus eliminating administrative and maintenance burdens that remote daemons and alike normally provoke. Another motivation for the remote probe approach was the possibility of tailoring the probe to our monitoring needs, capturing only the wanted metrics. Windows built-in remote monitoring capabilities like *perfmon* and Windows Management Interface (WMI) were discarded for several reasons. First, both mechanisms have high timeout values (order of seconds) when the remote machine to be monitored is not available. Also, both impose a high overhead on the network and on the remote machine.

DDC schedules the periodic execution of software probes in a given set of machines. The execution of probes is carried out remotely, that is, the probe binary is executed at the remote machine. For that purpose, DDC uses Sysinternal's psexec utility [21] that executes application in remote windows machines if appropriate access credentials are given. All executions of probes are orchestrated by DDC's central coordinator host, which is a normal PC.

As stated above, a probe is a win32 console application that uses its output channels to communicate its results. One of DDC's tasks is precisely to capture the output of the probe and to store it at the coordinator machine. Additionally, DDC allows the probe output to be processed by so-called post-collecting code (written in the Python language) which is supplied by DDC's user and specific to a probe. If defined, post-collecting code is executed at the coordinator site, immediately after a successful remote execution. This code receives as input arguments the content of both standard output and standard error channels, besides other context information such as the name of the remote machine. The purpose of the post-collecting code is to permit the analysis of the probe's output immediately after its execution, so that relevant data can be extracted and, if deemed necessary, saved in an appropriate format for future use. Figure 1 schematizes DDC execution. In step (1), the probe W32Probe is executed in a remote machine. Next (2), output results are returned to the coordinator machines. These results are post-processed at the coordinator's (3) and stored.

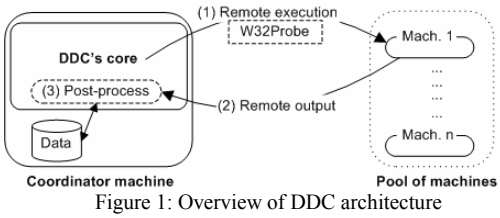

Figure 1: Overview of DDC architecture

The overhead induced by DDC is mostly dependent on the probe, since the remote execution mechanism requires minimal resources from the remote machine. Since W32Probe gathers its monitoring data mostly through win32 API calls, the probe requires practically no CPU and minimal memory for its execution.

For a more complete description of DDC interested readers are referred to [20].

3.1. Monitored metrics

W32Probe collects static and dynamic metrics characterizing the current state of the machine that is being monitored. Static metrics describe fixed characteristics that typically remain constant over time, while dynamic metrics measure runtime usage of main resources.

3.1.1 Static metrics

Static metrics comprise the following elements:

- Processor name, type and frequency: identifies the processor name and its operating frequency.
- Operating system: name and version.
- Main memory: size of installed main memory.
- Virtual memory: size of configured virtual memory.
- Hard disks: for every installed hard disk returns the serial number and the drive's size.
- Network interfaces: display MAC addresses for every installed network interface.

3.1.1 Dynamic metrics

Dynamic metrics include the following items:

- Boot time and uptime: system's boot time and respective uptime.
- CPU idle time: time consumed by the operating system idle thread since the computer was booted. This metric is further used to compute the average CPU idleness between two consecutive samples.
- Main memory load: main memory load (as returned by the field *dwMemoryLoad* filled by *GetMemoryStatus()* win32 API function).
- Swap memory load: analogue to main memory load metric but for the swap area.
- Free disk space: returns free disk space.
- Hard disk power cycle count: SMART's parameter that counts the number of disk's power cycles, i.e., the number of times the disk has been powered on/off since it was built.
- Hard disk power on hour counts: SMART parameter that counts the number of hours that a hard disk has been powered on since it was built.
- Network usage: this metric comprises "total received bytes" and "total sent bytes".
- Interactive user-login session: username and the session init time are returned if any user is interactively logged at the monitored machine.

4. Experiment

Using DDC and W32Probe, we conducted a 77-day (11 weeks) monitoring experiment using 169 computers of 11 classrooms of an academic institution.

4.1 Computing environment

The monitored classrooms are used for computer classes for several subjects, ranging from economics to computer sciences. When no classes are being taught, students can use the machines to perform their practical assignments and homework, as well as communication activities (e-mail, etc.). To avoid any changes of behavior that could false results, only system managers were aware of the monitoring.

All classrooms have 16 machines, except L09 which only has 9 machines. All machines run Windows 2000 professional edition (SP3) and are connected via a 100 Mbps Fast-Ethernet link. The main characteristics of the computers are summarized in Table 1 grouped by classrooms (from L01 to L11). The columns INT and FP refer respectively to NBench benchmark integer and floating-point performance indexes [22]. NBench, which is derived from the well-know Bytemark benchmark, was ported from Linux with its C source code compiled under Visual Studio .NET in release mode. NBench indexes can be used to assess relative performance among the monitored machines since the same benchmark binary was used to compute the indexes over the machines. NBench performance indexes were gathered with DDC using the corresponding benchmark probe.

Table 1: Main characteristics of machines

Lab	CPU (GHz)	RAM MB	Disk (GB)	INT / FP
L01	P4 (2.4)	512	74.5	30.5 / 33.1
L02	P4 (2.4)	512	74.5	30.5 / 33.1
L03	P4 (2.6)	512	55.8	39.3 / 36.7
L04	P4 (2.4)	512	59.5	30.6 / 33.2
L05	PIII (1.1)	512	14.5	23.2 / 19.9
L06	P4 (2.6)	256	55.9	39.2 / 36.7
L07	P4 (1.5)	256	37.3	23.5/ 22.1
L08	PIII (1.1)	256	18.6	22.3 / 18.6
L09	PIII (0.65)	128	14.5	13.7 / 12.1
L10	PIII (0.65)	128	14.5	13.7 / 12.2
L11	PIII (0.65)	128	14.5	13.7 / 12.2
Avg.	–	340.8 MB	40.3 GB	25.5 / 24.6

Combined together, the resources of the 169 machines are impressive: 56.62 GB of memory, 6.66 TB of disk and more than 98.6 GFlops of floating-point performance.

4.2 Settings and limitations

For the purpose of the monitoring experiment, the period for W32probe execution attempt over the set of machines was set to 15 minutes. This value was a compromise between the benefits of gathering frequent samples and the negative impact this strategy might cause on resources, especially on machines and on the network. A 15-minute interval between samples means that captured dynamic metrics are coarse-grained, with quick fluctuations of values escaping the monitoring system. For instance, a 5-minute memory activity burst using nearly 100% of main memory is undistinguishable from 10-minute 50% memory usage, since samples comprising both memory usage bursts will report the same average memory space usage. However, this is seldom a problem, since all metrics are relatively stable, and thus not prone to fluctuate widely in a 15-minute interval. The only exception is the CPU idleness percentage, which is prone to quick changes. But, precisely to avoid misleading instantaneous values, CPU usage is returned as the average CPU idleness percentage observed since machine was booted. Therefore, given the CPU idleness values for two consecutive samples it is straightforward to compute the average CPU idleness between these two samples, given that no reboot occurred in the meantime.

A subtle and unexpected limitation of our methodology was triggered by user habits, particularly with users who forget to logout. In fact, over the original 277513 samples captured on machines with an interactive session, we found out 87830 samples corresponding to user interactive session lasting 10 hours or more. Since classrooms remain open 20 hours per day, closing from 4 am to 8 am, these abnormal lengthy sessions have to do, most certainly, with users who had left their login session opened. To assert our hypothesis, we grouped the samples of interactive sessions upon their relative duration since the start of the corresponding interactive session. For instance, samples collected during the first hour of any interactive session were counted together and so on. Data are plotted in Figure 2 and permit to observe that the time interval [10-11[hour (samples collected during the 10th and 11th hour of any interactive session) is the first interval that presents an average CPU idleness above 99%, a very high value that indicates that no interactive activity existed when the samples were collected. Therefore, in order to avoid results biased by abnormally long interactive user sessions, we consider samples reporting an interactive user-session equal or above than 10 hours as being captured on non-occupied machines. Note that this threshold is a conservative approach, which means that real interactive usage is probably lower than reported.

An obvious conclusion of forgotten session is that very high level of CPU idleness (99% or above) is a good indicator of non-interactive usage on a machine, even if an interactive session is opened.

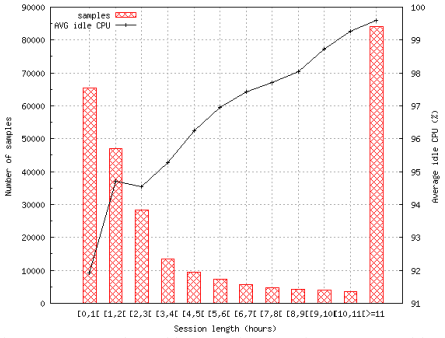

Figure 2: Samples of interactive sessions grouped by their relative time occurrence.

5. Results

During the experiment 6883 iterations were run over the whole machines with a total of 583653 samples collected. Main results of the monitoring are summarized in Table 2. The column "No Login" shows results captured when no interactive user-session existed, while column "With login" expresses samples gathered at user-occupied machines. Both results are combined in the final column "Both".

Table 2: Main results

Metric	No login	With login	Both
Samples	393970	189683	583653
Avg. uptime (%)	33.9	16.3	50.2
Avg. CPU idle (%)	99.7	94.2	97.9
Avg. RAM load (%)	54.8	67.6	58.9
Avg. SWAP load (%)	25.7	32.8	28.0
Avg. disk used (GB)	13.6	13.6	13.6
Avg. sent bytes (bps)	255.3	2601.8	1071.9
Avg. recv. Bytes (bps)	359.2	8662.1	3057.9

Machines responded to 50.2% of the sample attempts over the 77 days and in 393970 samples (33.9%) queried machine did not have any interactive login session. This means that for slightly more than one third of the time, machines were completely available and free for resources harvesting. In fact, unoccupied machines presented a high 99.7% CPU idle time, expressing almost full idleness. The presence of interactive session reduces CPU idleness to an average of 94.2%, meaning that an interactive session roughly requires 5.5% of CPU usage. This CPU idleness confirms other studies performed in academic classrooms running Unix environments [13], but with higher than values found by Bolosky et al. [15], who reported an average CPU usage rounding 15% for corporate machines. In fact, Bolosky stated that some of the machines presented a continuous 100% CPU usage, a fact that obviously raised mean CPU usage.

As expected, main memory demands increases roughly 12% when interactive usage occurs at a machine. This is a natural behavior, since an interactive session obviously means that interactive applications will be opened and thus consuming memory. Similarly, swap memory load rises by 5% when an interactive user is logged on the machine.

Used disk space is independent of the presence of interactive login sessions: average of 13.6 GB for both situations. The low variability respecting used disk space is a consequence of system usage policy: an interactive user is restricted to 100 MB to 300 MB of temporary local hard disk drive (the actual size depends on the capacity of the machine hard drive), meaning that it can be cleaned after an interactive session has terminated. In fact, users are fostered to keep their files in a central server with the benefit of being able to access their files independently of the desktop machine being used.

Relatively to network usage, occupied machines present average usages (sent and received) roughly 10 times superior to user-free machines. Also, confirming the client role of machines, received rates are nearly 4 times bigger than sent rates.

5.1 Machines availability

Figure 3 plots machines availability during the 11-week monitoring experiment. Figure 3a) (left) shows the number of powered on machines; Figure 3b) (right) traces the count of user-free machines, that is, machines powered on without interactive logged on user at sample time. In each plot, the black horizontal line displays average of samples, which are 84.87 for powered on machines, and 57.29 for session-free machines. This means that roughly, on average, 70% of the powered on machines are free of users, and thus fully available for foreign computation. Also, on average, slightly more than half of the set of 169 machines is powered on.

All plots exhibit a similar sharp pattern with high-frequency variations showing that machine counts fluctuate widely, except on weekends (note that the x-axis labels of the plots denote Mondays). Since classrooms are open on Saturdays, weekend slowdowns are more noticeable on Sundays. The high-frequency variations exhibited on weekdays mean that resources are volatile and thus harvesting such resources require highly tolerant and adaptable mechanisms.

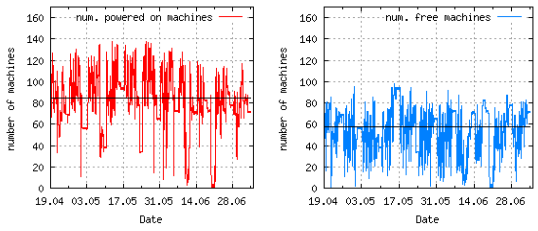

Figure 3: Number of machines powered on (left) and user-free (right) over the 77-day experiment

Left plot of Figure 4 shows two metrics related to uptime. The double cross curve, which appears almost as a straight line, represents machine availability measured in units of "nines" [34]. Nines are defined as log10 of the fraction of time a host is not available. The name of the unit comes from the number of nines in its availability ratio. For example, one nines means a 0.9 availability ratio, that is, –log10(1-0.9)= 1 nine. The simple cross curve displays the fraction of time each machine is up. In both curves, machines are sorted in descending order by their cumulated uptimes.

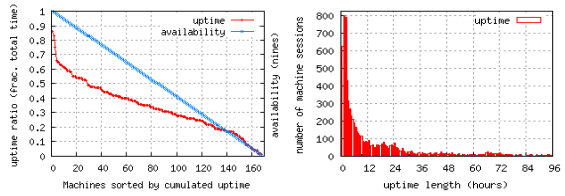

Figure 4: uptime ratio and availability in nines (left) and distribution of machines' uptime (right).

The ratio availability curve shows that only 30 machines have cumulated uptimes bigger than half the experiment period, that is, 37.5 days. Also, less than 10 machines have cumulated uptimes ratio higher than 0.8 and none was above 0.9. Comparatively to the Windows corporate environment depicted in [23] where more than 60% of machines presented an uptimes bigger than one nine, analyzed classroom machines present much lower uptime ratios. This is a consequence of the machines having no real owner and thus subject to the possibility of being powered off at the end of a class, contrary to corporate machines that split in two patterns: daytime and 24 hours. Daytime machines are powered on during office hours, while 24 hours machines remain powered on for long periods.

5.2. Machines stability

In this section we analyze machines' sessions, focusing on uptime length and reboot count. We define a machine session as the activity comprised between a boot and its corresponding shutdown.

5.2.1 Uptime

During the whole experiment 10688 sessions of machines were captured by our sampling methodology. It is important to note that due to the 15-minute period between consecutive samples, some short machine sessions might have not been captured. In fact, between two samples, DDC can only detect one reboot, since its reboot detection is based upon machine's uptime.

The average duration of the length of sessions was 15 hours and 55 minutes. This value exhibits a high standard deviation of 26.65 hours indicating that session length fluctuates widely. Since multiple reboots occurred between two samples escape DDC (only one is detected), the above given average duration exceeds the real value. The right plot of Figure 4 displays the distribution of machines' uptime length for sessions that lasted less or equal than 96 hours (4 days). These sessions accounted for 98.7% of all machine sessions and 87.93% of cumulated uptime. These data permit to conclude that most machine sessions are short, lasting few hours, another indication of the high volatility of machines.

5.2.2 Power on cycles

As stated before, due to the coarse-grained granularity of the sample methodology, some of the short machine sessions may go unnoticed. Thus, in order to have a detailed view respecting reboots, we recurred to SMART parameters [19]. By resorting to the "power cycle count" metric, it is possible to spot undetected machine sessions. For instance, if the "power cycle count" values of two consecutive samples of a machine differ by more than one unit, this means that at least one short machine session with its corresponding boot and shutdown sequence, occurred.

The cumulated count of hard disk power on cycles was 13871, with an average of 82.57 power cycles per machine and a standard deviation of 37.05 over the 77 days. This represents 1.07 power on cycle per day, a value 30% higher than the number of machine sessions counted by our monitoring analysis. This means that a significant percentage of power cycles are of very short duration (less than 15 minutes) escaping our sampling mechanism.

Resorting to the parameters "power on hour count" and "power cycles" it is possible to compute the average power on hours per power on cycle, henceforth referred as "uptime per power cycle". For the 77-day monitoring, uptime per power cycle was 13 hours and 54 minutes with a standard deviation of nearly 8 hours. The difference between the average uptime per power cycle and the average machine session length (see section 5.2.1) can be explained by the short-lived sessions that are not caught by our sampling methodology. Given the absolute count of power cycles and power on hours, it is possible to compute the uptime per power cycle for the whole disk life. Since machines are relatively new (all machines are less than 3 years old) the probability of machines conserving their original disk is high and thus average uptime per power cycle serves as a measure of average uptime. For our monitored system, the average power on hours per power on cycle was 6.46 hours with a standard deviation of 4.78 hours. This value is surprisingly lower than the one we found during our monitoring study.

5.3. Weekly analysis

Figure 6 aggregates two plots related to the weekly distribution of samples. Left plot displays weekly distribution of average percentage of CPU idleness (top

curve), RAM occupation (middle curve) and SWAP load (bottom curve). Right plot shows average network rates for received (top curve) and sent traffic (bottom curve). Besides following the night and weekend pattern, the weekly distribution of average CPU idleness presents a significant negative spike on Tuesdays afternoons, dropping below 91%. This CPU usage spike was due to a practical class which consumed an average of 50% of CPU, although we could not find the reasons behind this abnormal CPU usage. Confirming high CPU availability, average CPU idleness never drops below 90% and mostly ranges from 95% to 100%. The phases of near 100% average CPU idleness correspond to the periods when classrooms are closed: 4 am to 8 am during weekdays and from Saturday 9 pm to Monday 8 am for weekends.

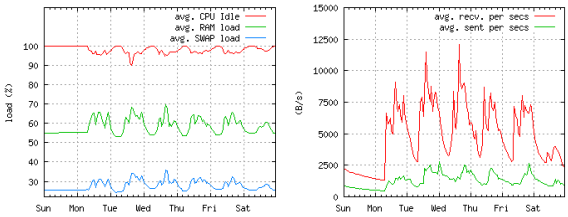

Figure 5: Weekly distribution of average CPU idleness and memory usage (left) and network traffic (right).

Both RAM and swap load curves exhibit the weekly pattern, although in a smoothed way. Note that RAM load never falls below 50%, meaning that a significant amount of RAM is required by operating system usage. Comparing RAM and swap usage, it can be observed that the swap curve roughly follows memory usage, although strongly attenuating high frequencies.

Network rates weekly distributions also exhibit the weekend and night pattern. Since we are plotting a rate, the drops originated by night periods and weekends appear as smoothed lines. Network client role of machines appears clearly visible, with received rates several times higher than sent rates.

Weekly distribution of resource usage permits to conclude that apart from weekends and the night interval between 4 am and 8 am, absolute system idleness is limited. However, even on working hours, idleness levels are quite high, permitting successful yields in resource scavenging schemes.

5.4 Equivalence ratio

Arpaci et al. [12] defines the equivalent parallel machine as a metric to gauge the usable performance of non-dedicated machines relatively to a parallel machine. Kondo et al. [18] adopt an analogue metric, calling it cluster equivalence metric. This metric aims to measure the fraction of a dedicated cluster CPU that a non-dedicated machine CPU is worth to an application. We apply this definition, computing the CPU availability of a machine for a given period accordingly to its measured CPU idleness over this period. For instance, a machine with 90% CPU idleness is viewed as a dedicated machine with 90% of its computing power. This methodology assumes that all idle CPU can be harvested. Thus, obtained results should be regarded as an upper limit of CPU resources that can be harvested. To cope with heterogeneity, machines' performances were normalized accordingly to their respective INT and FP indexes (a 50% weight was given to each index to compute a machine index).

Figure 6 plots the weekly distribution of the cluster equivalence ratio. The average cluster ratio is 0.26 for occupied machines and 0.25 for user-free machines, totaling a 0.51 cluster equivalence ratio. This means that the set of non-dedicated machines is roughly equivalent to a dedicated cluster with half the size, following the 1:2 rule found out by [12].

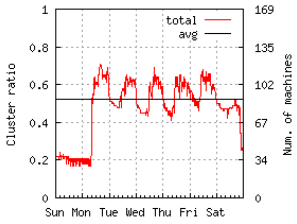

Figure 6: Weekly distribution of equivalence cluster

A more complete discussion of results is available in [24].

6. Conclusions

This paper presents the main results of a 77-day monitoring usage study of 169 Windows 2000 machines. The results show that resources idleness in academic classrooms comprised of Windows machines is very high, confirming previous works carried out in classrooms with Unix machines.

The average CPU idleness observed was impressive: 97.9%. Also, the 94.2% average CPU idleness measured in user occupied machines indicates that CPU harvest schemes should be profitable even when interactive usage of the machine exists. This is confirmed by the 0.26 cluster equivalence ratio potentially available from exploiting CPU idleness of interactively used machines and 0.51 when user-free machines are also considered.

Memory idleness is also noticeable especially in machines fitted with 512 MB of main memory. Coupled with a fast network technology such resources might be put to good use for network RAM schemes.

Due to the fact that every machine only contains the operating system installation plus specific software needed for classes, free space storage among monitored machines is impressive. A possible application for such disk space relates to distributed backups or to the implementation of local data grids.

Classrooms comprised of Windows machines seem appropriate for desktop grid computing not only limited to CPU, but also to main memory and free hard disk space. Beside wide resources availability, attractiveness of such environments for resources harvesting is strengthened by the fact that machines have no real personal owner.

A major concern in classroom environments relates to volatility, since a resource available at a given time might become unavailable few seconds later. Thus, efficient usage of idle resources requires survival techniques such as checkpointing, oversubscription and multiple executions.

We believe our results can be generalized to other academic classrooms which use Windows operating systems and which follow a similar classroom usage policy: shared machines for classes, with students accessing computers for work assignment and communication use.

In conclusions, this study confirms that resource idleness observed in classrooms with Windows computers is quite considerable, and carefully channeled could yield good opportunities for grid desktop computing.

Acknowledgements

This work was partially supported by PRODEP III/5.3 and by the Portuguese FCT through the R&D Unit 326/94 (CISUC).

References

[1] L. Sarmenta, "Bayanihan: Web-Based Volunteer Computing Using Java.," presented at 2nd International Conference on World-Wide Computing and its Applications (WWCA'98), Tsukuba, Japan, 1998.

[2] BOINC Project (http://boinc.berkeley.edu/)," 2005.

[3] Condor, "Condor Project Homepage (http://www.cs.wisc.edu/condor/)," 2005.

[4] G. Fedak, C. Germain, V. Neri, and F. Cappello, "XtremWeb: A Generic Global Computing System," CCGRID'01, Brisbane, 2001.

[5] A. Chien, B. Calder, S. Elbert, and K. Bhatia, "Entropia: architecture and performance of an enterprise desktop grid system," *Journal of Parallel and Distributed Computing*, vol. 63, 2003.

[6] UD, "United Devices, Inc. (http://www.ud.com)."

[7] R. Figueiredo, P. Dinda, and J. Fortes, "A Case For Grid Computing On Virtual Machines," ICDCS'03, Providence, Rhode Island, 2003.

[8] C. d. Rose, F. Blanco, N. Maillard, K. Saikoski, R. Novaes, O. Richard, and B. Richard, "The Virtual Cluster: A Dynamic Environment for Exploitation of Idle Network Resources," 14th Symposium on Computer Architecture and High Performance Computing, Brazil, 2002.

[9] R. Novaes, P. Roisenberg, R. Scheer, C. Northfleet, J. H. Jornada, and W. Cirne, "Non-Dedicated Distributed Environment: A Solution for Safe and Continuous Exploitation of Idle Cycles," AGridM 2003, Workshop on Adaptive Grid Middleware, 2003.

[10] T. E. Anderson, D. E. Culler, and D. Patterson, "A case for NOW (Networks of Workstations)," *Micro, IEEE*, vol. 15, pp. 54-64, 1995.

[11] A. Gupta, B. Lin, and P. A. Dinda, "Measuring and understanding user comfort with resource borrowing," 13th HPDC, Honolulu, USA, 2004.

[12] R. Arpaci, A. Dusseau, A. Vahdat, L. Liu, T. Anderson, and D. Patterson, "The interaction of parallel and sequential workloads on a network of workstations," presented at ACM SIGMETRICS, Ottawa, Ontario, Canada, 1995.

[13] A. Acharya, G. Edjlali, and J. Saltz, "The utility of exploiting idle workstations for parallel computation," ACM SIGMETRICS Seattle, Washington, USA, 1997.

[14] A. Acharya and S. Setia, "Availability and utility of idle memory in workstation clusters," ACM SIGMETRICS - Measurement and modeling of computer systems, Atlanta, Georgia, USA, 1999.

[15] W. Bolosky, J. Douceur, D. Ely, and M. Theimer, "Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs," SIGMETRICS - Measurement and modeling of computer systems, Santa Clara, USA, 2000.

[16] K. D. Ryu and J. K. Hollingsworth, "Unobtrusiveness and efficiency in idle cycle stealing for PC grids," 18th IPDPS, 2004.

[17] D. G. Heap, "Taurus - A Taxonomy of Actual Utilization of Real UNIX and Windows Servers," IBM White Paper GM12-0191, 2003.

[18] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien, "Characterizing and evaluating desktop grids: an empirical study," 18th IPDPS, 2004.

[19] G. Hughes, J. Murray, K. Kreutz-Delgado, and C. Elkan, "Improved Disk Drive Failure Warnings," *IEEE Transaction on Reliability*, 2002.

[20] P. Domingues, P. Marques, and L. Silva, "Distributed Data Collection through Remote Probing in Windows Environments," 13th Euromicro PDP, Lugano, Switzerland, 2005.

[21] M. Russinovich and B. Cogswell, "Sysinternals - PsTools (http://www.sysinternals.com/)," 2004.

[22] U. Mayer, "Linux/Unix nbench project page (http://www.tux.org/~mayer/linux/)," 2003.

[23] J. Douceur, "Is remote host availability governed by a universal law?" *SIGMETRICS Perform. Eval. Rev.*, vol. 31, pp. 25-29, 2003.

[24] P. Domingues, P. Marques, and L. Silva, "Resources Usage of Windows Computer Laboratories," CISUC, Portugal. Technical Report. www.cisuc.uc.pt/view_member.php?id_m=207, January 2005.