Technical Report

Resources Usage of Windows Computer Laboratories

Patricio Domingues[α], Paulo Marques[β], Luis Silva[β]

Email:patricio@estg.ipleiria.pt,{pmarques,luis}@dei.uc.pt

[α]Escola Superior de Tecnologia e Gestão - Instituto Politécnico de Leiria – Portugal
[β]Departamento Eng. Informática, Universidade de Coimbra – Portugal

January 2005

Document available at: http://www.cisuc.uc.pt/view_member.php?id_m=207

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 of an academic institution 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, recurring 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. The paper also analyzes the potential of non-dedicated classroom Windows machines for distributed and parallel computing, evaluating the mean stability of group of machines.

Our results show that resources idleness in classroom computers is very high, with an average CPU idleness of 97.93%, unused memory averaging 42.06% 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. However, the efficient exploitation of the computational power of these environments requires adaptive fault-tolerance schemes to overcome the high volatility of resources.

Keywords: monitoring, resource usage, cycle stealing, desktop grid-computing.

1. Introduction

In the last two decades personal computers (PCs) have invaded organisations, becoming essential working tools in offices, laboratories and classrooms. The continuous and somewhat massive deployment of more and more powerful PCs means that more resources go unused, especially since many machines are primarily devoted to low-demanding electronic office applications. At the same time, affordable and fast network technology, especially for local area network, permits distributed resource harvesting to focus not only on CPU, but also on idle main memory and in otherwise unused disk space.

Today, academic institutions frequently have large dozens of 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, except for the additional electric power consumption [1]. In fact, numerous environments exist to harness idle resources. Examples ranges from academic projects such as Bayanihan [2], BOINC [3], Charlotte [4], Condor [5], GLUnix [6], Harmony [7], I-Cluster [8], Intergrade [9], Javelin [10], JET [11], SuperWeb [12], XtremWeb [13] to commercial solutions like Entropia [14], United Devices [15], Platform [16] and Data Synapse [17]. More recently, some desktop grid schemes have emerged relying on virtual machines [18]. Examples include Virtual Cluster [19] and Non-Dedicated Distributed Environment (NDDE) [20].

PCs of classrooms have an attractive characteristic for resource harvesting: no individual user owns the PCs. Office's computers are generally affected to an individual user (after all they are called "personal" computers). The owner of the machine controls it, being very suspicious about possible invasion of the machine by foreign programs. In fact, frequently, individual owners do not tolerate resource stealing schemes. So the use of individual PCs for distributed and parallel computing has to deal with social issues, beside engineering and technology [21]. Being centrally managed, classroom computers have no individual owner, and thus social issues involving their use in resource harvesting are weaker. However, even without individual owner, care must be taken to avoid undermining regular user comfort, since resources are primarily devoted to interactive users. Gupta et al. [22] analyse 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.

Windows operating system variants hold an important part of the desktop market. Statistics gathered from browsers usage [23] show that 89.7% of requests come from machines running Windows. Similarly, statistical data from the SETI@Home project reveals that 81.5% of the results received were computed by Windows machines. Figure 1 plots the percentage distribution of computed workunits of the SETI@Home projects in the month of December 2004.

Despite this predominance, few studies analyze real resource usage of Windows machine, and even fewer quantify resource idleness existing in classrooms fitted with Windows machines.

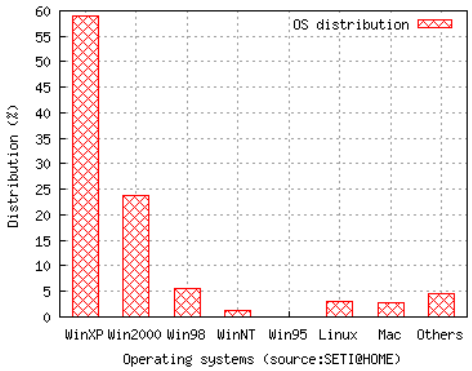

Figure 1: Percentage of SETI@Home's computed workunits distributed by operating systems (source: SETI@Home, December 2004)

The remainder of this report is organised 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 report and outlines future work.

2. Related work

Evaluation of computer resources usage has been a research topic since the wide acceptance 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 [24] [25] but also memory [26] and disk storage [27].

Through simulation Arpaci et al. [24] study the interaction of sequential and parallel workloads. They conclude that for their workloads a 2:1 rule applies, meaning that N non-dedicated machines are roughly equivalent to N/2 dedicated machines.

The study presented in [25] 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, that is, machines frequently go down.

Acharya and Setia [26] 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, a large fraction of the total memory installed on a machine is available, with idle machines presenting around 50% of unused main memory.

Ryu et al. in [28] 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 much interference on interactive users. However, their methodology requires modification at the kernel level and therefore 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 up-to-date machines over a relatively long period of time (11 consecutive weeks).

Bolosky et al. [27] study a vast set of Microsoft Corporate desktop machines, reporting availability, CPU load and file system usage in a corporate environment. The study is oriented toward the demonstration of the viability of a serveless distributed file system. Thus important 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 resources usage, obtaining results substantially different, especially respecting a much lower CPU load than observed in the corporate environment depicted in [27].

Heap [29] studied the resource usage of Unix and Windows servers, through 15-minute periodic gathering of monitoring data. He found out that Windows servers had a CPU idleness average near 95%, while Unix servers averaged 85% CPU idleness.

The study of P. Cicotti et al. [30] 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, Entropia in this case.

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 up-to-date desktop machines (Pentium III and Pentium 4). An analysis of main resource usage is conducted with emphasis in differentiating resource usage between machines occupied with interactive users and free machines, assessing the effect of interactive user over resources consumption.

We also present a novel approach to assess machines availability, combining collected samples with data extracted from the Self Monitoring Analysis and Reporting Technology (SMART) counters of machines' hard disks, namely the "*power on hour counts*" and "*power on cycle*" [31] counters. Coupled

with the collected traces, these SMART values permit to infer about machines "*power on pattern*" allowing a rough estimation of machines availability since hard disk was installed, which is, for most of the computers, the date they were built.

3. Methodology

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).

The need to automate the periodic data collection over the machines to survey fostered us to develop a framework to support remote data collection in local area networked Windows machines. The framework was named Distributed Data Collector (DDC) [32] and aims to cover the needs that arise in distributed data collection at the scale of local area networks of Windows PCs.

The remote probing solution was chosen in part 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) [33] 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. *Perfmon*'s overhead is caused mostly by the need to transfer the remotely read data to the local machine while WMI's overhead is a consequence of its dependence over Distributed COM (DCOM) for accessing a 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 [34] 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 (both *stdout* and *stderr*) to communicate its results. One of DDC 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* which is a supplied by DDC's user and specific to a probe. This code needs to be written in the Python programming language. 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 and coordinator's file system directory used to store data respecting remote executions. The purpose of the post-collecting code is to permit analysis of 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 2 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.

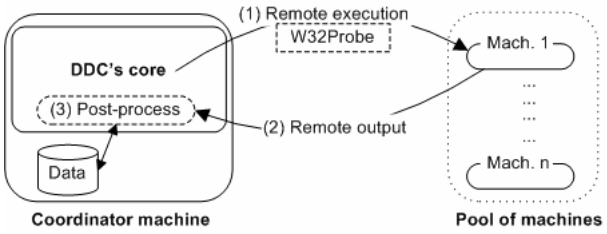

Figure 2: Overview of DDC architecture

DDC schedules probe executions in iterations. An iteration consists of the execution attempt of every defined probe (DDC supports multiple probes) and attached post-collecting code (if any) over the whole pool of configured machines. After having completed an iteration by processing all defined machines, DDC waits until a timer expires to start the next iteration.

One of the strengths of DDC lies in its support for data collection over machines with transient availability, that is, machines that have no requirement of being permanently powered on and that offer no guarantee of being switched on at a given moment. For that purpose, before an execution is attempted, DDC pings a machine with a sub-second timeout. This way, an unavailable remote machine is quickly detected with execution attempt being switched to the next machine.

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. Also, to minimise network traffic, the output produce by the probe contains only the data deemed necessary for the monitoring experiment.

DDC was a central piece in the monitoring infrastructure, allowing us to collect the whole trace usage without installing any software at remote machines.

DDC software is available under the GNU Public License (GPL) For a more complete description interested readers are referred to [32].

3.1 Monitored metrics

For the purpose of this study we developed W32Probe. This probe collects several metrics characterizing the current state of the machine that is being monitored. These metrics are grouped in two categories: *static* and *dynamic*. Static metrics describe fixed characteristics that typically remain constant over time. Examples of such metrics include CPU name and type and amount of installed main memory. Dynamic metrics are related to current computing activity, measuring target machine main resources usage. Dynamic metrics include CPU idleness percentage, memory load, available free disk space and whether an interactive session exists at the machine. Next, a brief description of both categories of metrics is given.

3.1.1 Static metrics

Static metrics comprise the following elements:

- Processor name, type and frequency: this identifies the processor name and its operating frequency.
- Operating system: name, version and service pack version, if any.
- Amount of main memory: size of installed main memory.
- Amount of virtual memory: size of configured virtual memory.
- Hard disks: for every installed hard disk returns a descriptive string, the serial identification number and the size of the drive.
- Network interfaces: display MAC address and description strings for every installed network interface.

3.1.2 Dynamic metrics

Dynamic metrics collected from W32Probe include the following items:

- Boot time and uptime: system's boot time and respective uptime corresponding to the moment when probe was run.
- 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.
- CPU idle percentage: CPU idle percentage since machine was booted. This metric simply corresponds to the division of "*CPU idle time*" by machine's uptime.

- Main memory load: main memory load (as returned by the field *dwMemoryLoad* filled by win32's *GetMemoryStatus()* 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 parameter that counts the number of disk's power cycles, i.e., the number of times the disk has been powered on/powered 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 two main values and two derived ones. Main values are "*total received bytes*" and "*total sent bytes*". Derived values are "*received byte rate*" and "*sent byte rate*" which are simply computed respectively from *"total received bytes"* and *"total sent bytes"*.
- Interactive user-login session: if any user is interactively logged at the monitored machine, the username and domain name (if any), along with the session init time are returned.

4. Experiment

4.1 Computing environment

Using DDC and W32Probe, we conducted a 77-day monitoring experiment using 169 computers of 11 classrooms of an academic institution from midnight 19th April to midnight 5th July 2004, for a total of 11 complete weeks, with only the last week corresponding to off-classes. Monitored classrooms are used for regular classes. When no classes are being taught students use the machines to perform practical assignments and homework, as well as for personal use (*e-mail*, etc.). To avoid any changes of behaviour that could false results, only system managers were aware of the monitoring experiment.

All classrooms have 16 machines, except L09 which only has 9 machines. All machines run Windows 2000 professional edition (service pack 3) 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 [35] integer and floating-point performance indexes. NBench, which is derived from the well-know Bytemark [36] benchmark, was ported from Linux with its C source code compiled under Visual Studio .NET in release mode. Like its predecessor, NBench relies on well-know algorithms to summarize computer performance with two numerical indexes: INT for integer performance and FP to expose floating-point performance. It is important to note that presented indexes are not suitable for absolute comparisons with NBench original values, since both operating system and the compiler are different. However, indexes can be used to

assess relative performance among the monitored machines, since the same benchmark binary was used to compute the indexes. The final column of Table 1 expresses FLOPS performance as given by Linpack [37] benchmark compiled with Visual Studio .Net in release mode. All performance indexes were gathered with the DDC framework using the corresponding benchmark probe (NBench for INT and FP, Linpack for MFlops). Figure 3 plots the machines sorted by their INT (left) and FP (right) performance indexes. The plots present similar shape, indicating that a single index (either INT or FP) is enough for comparing performances of a set of machines.

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.

	Qty	CPU (GHz)	RAM (MB)	Disk size (GB)	INT	FP	Linpack (MFlops)
L01	16	P4 (2.4)	512	74.5	30.53	33.12	850.31
L02	16	P4 (2.4)	512	74.5	30.46	33.08	851.19
L03	16	P4 (2.6)	512	55.8	39.29	36.71	903.18
L04	16	P4 (2.4)	512	59.5	30.55	33.15	847.23
L05	16	PIII (1.1)	512	14.5	23.19	19.88	389.49
L06	16	P4 (2.6)	256	55.9	39.24	36.65	899.32
L07	16	P4 (1.5)	256	37.3	23.45	22.10	520.10
L08	9	PIII (1.1)	256	18.6	22.27	18.64	396.52
L09	16	PIII (0.65)	128	14.5	13.65	12.21	227.37
L10	16	PIII (0.65)	128	14.5	13.68	12.22	227.33
L11	16	PIII (0.65)	128	14.5	13.68	12.22	227.32
Total	**169**	**–**	**56.25 GB**	**6.66 TB**	**4315.69**	**4164.98**	**98654.12**
Avg.	**–**	**–**	**340.83 MB**	**40.33 GB**	**25.54**	**24.64**	**583.75**

Table 1: Main characteristics of monitored machines.

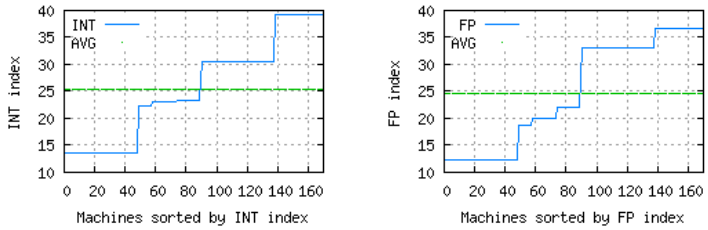

Figure 3: Machines sorted by computing power. INT index (left) and FP index (right).

4.2 Settings and limitations

For the purpose of the monitoring experiment, the period for W32probe execution attempt over the set of machines was configured 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 (if a reboot occurred, then the sample taken after the reboot reports the average CPU idleness since machine was booted).

A subtle and unexpected limitation of our methodology was due to 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 with users who had left their login session opened. To assert our hypothesis, we grouped the samples of interactive sessions upon their relative time occurrence 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. For every time interval the average and standard deviation of CPU idleness was computed. Table 2 presents these data, with the first column corresponding to the time intervals, the second column holding the count of samples and the third displaying average CPU idleness jointly with standard deviation. The data permits to observe that the time interval [10-11[hour (samples collected during the 10^{th} and 11^{th} hour of any interactive session) is the first one that presents an average CPU idleness above 99% (99.27%), a very high value that indicates that no interactive activity existed when the samples were collected. Therefore, in order to avoid results biased by such abnormal interactive user sessions, in this study 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 in this study. Figure 4 plots the number of samples (left y-axis) and the average percentage of CPU idleness (right y-axis) of data shown in Table 2.

Length of session (hour)	Number of samples	Avg. CPU idleness (stdev)
[0-1[	65521	91.93% (12.69)
[1-2[	47057	94.72% (11.08)
[2-3[	28374	94.54% (11.76)
[3-4[	13387	95.28% (12.57)
[4-5[	9514	96.24% (11.40)
[5-6[	7334	96.95% (10.28)
[6-7[	5654	97.43% (9.51)
[7-8[	4754	97.70% (9.24)
[8-9[	4181	98.03% (8.61)
[9-10[	3907	98.73% (6.09)
[10-11[	3637	99.27% (3.84)
>=11	84193	99.61% (1.65)

Table 2: Samples from interactive sessions grouped by their relative time occurrence.

An important conclusion to drawn from the forgotten sessions is that verification of user logins does not seem enough to assess machine interactive usage. Metrics like keyboard idleness and mouse usage should be used as a complementary diagnosis. However, in Windows environments monitoring of keyboard and mouse require, to the best of our knowledge, usage of driver hooks, which not only forces software installation at remote machines, but also require the software to be run at a high privilege level. Interestingly, very high level of CPU idleness (99% or above) also seems to be a good indicator of non-interactive usage on a machine, even if an interactive session is opened. Finally, another conclusion to be drawn from forgotten user sessions is the need to configure classroom computers to detect long unused user sessions and to automatically logout.

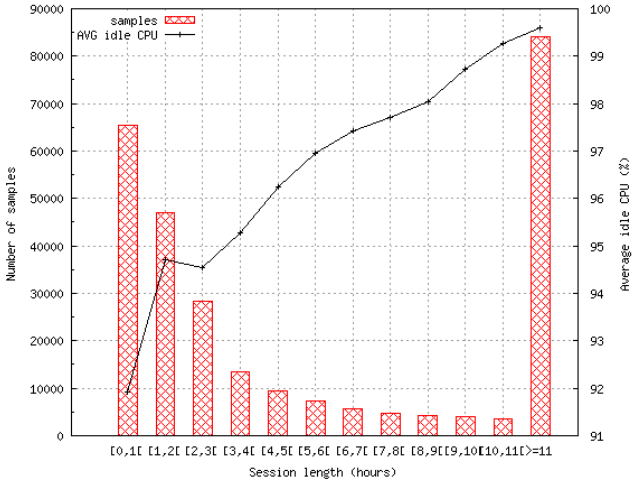

Figure 4: Samples from interactive sessions grouped by their relative time occurrence. Left y-axis depicts number of samples, right y-axis plots average CPU idleness.

5. Results

During the 77 days of the experiment, 6883 iterations were run 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". On rows that display average values, standard deviation is given in parenthesis.

	No login	With login	Both
Samples (avg. uptime %)	393970 (33.87%)	189683 (16.31%)	583653 (50.18%)
Avg. CPU idle	99.71% (1.99)	94.24% (11.20)	97.93% (4.99)
Avg. RAM load	54.81% (8.45)	67.53% (11.95)	58.94% (9.59)
Avg. SWAP load	25.74% (4.28)	32.83% (7.86)	28.04% (5.44)
Avg. disk used	13.64 GB (3.30)	13.64GB (4.31)	13.64GB (3.63)
Avg. sent bytes	255.32 Bps (7029.56)	2601.79 Bps (31241.85)	1017.91Bps (14898.38)
Avg. received bytes	359.17Bps (5754.62)	8662.07Bps (47604.81)	3057.86 Bps (19357.18)

Table 3: Global monitoring resource usage.

Machines responded to 50.18% of the sample attempts over the 77 days and in 393970 samples (33.87%) queried machine did not have any interactive login session. This means that during the 77 days, for slightly more than one third of the time, machines were completely available and free for resources harvesting. In fact, unoccupied machines presented 99.71% CPU idle time, expressing almost full idleness. The presence of interactive session reduces CPU idleness to an average of 94.24%, 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 [25], but with higher than values found by Bolosky et al. [27], who reported an average CPU usage rounding 15%. In fact, Bolosky analysed corporate machines stating that some of the machines presented an almost continuous 100% CPU usage, a fact that obviously raised mean CPU usage.

Main memory usage values are difficult to interpret recurring only to global averages, since installed main memory of the assessed machines ranged from 128 MB to 512 MB. However, as expected, main memory demands increases roughly 12% when interactive usage occurs at a machine. This is a natural behaviour, since an interactive session obviously means that interactive applications will be opened and thus consuming memory. Even though, the broad conclusion is that a significant amount of memory goes unused. Again, as a consequence of higher main memory usage verified during interactive sessions, swap memory load raises 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 13.64 GB for both situations. The low variability respecting used disk space, confirmed by the low standard deviation of 3.63, 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. This policy restricts users from cluttering disks, also avoiding that personal files can mistakenly be forgotten, or that *trojan* programs and alike be maliciously dropped in shared machines. 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.

5.1 Machines availability

Figure 5 plots machines availability during the 11-week monitoring experiment. Figure 5a) (left) shows the number of powered on machines; Figure 5 b) (middle) displays the count of occupied machines, that is, machines with an interactive session. Figure 5c) (right) traces the count of user-free machines, that is, machines powered on without interactive logged on user at sample time. In each graph, the black horizontal line displays average of samples, which is 84.87 for powered on machines, 27.58 for

occupied 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.

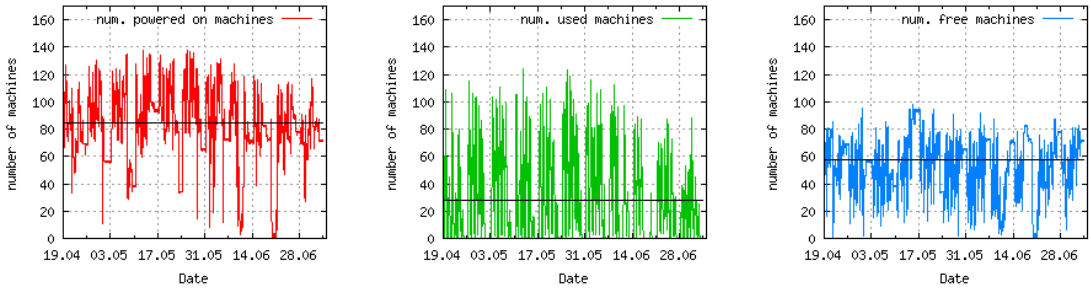

Figure 5: Number of machines powered on (left), with user-session (middle) and user-free (right).

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, except for Saturdays 1st May and 22nd May, both of them holidays. The negative spike that can be found around 10th June corresponds to another holiday, while the other negative spike on 21st June was motivated by a somewhat long maintenance period on the power circuits that feed the classrooms. The high-frequency variations exhibited on weekdays mean that resources are volatile and thus harvesting such resources require highly tolerant and adaptable mechanisms.

Left plot of Figure 6 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" [38]. Nines are defined as $-\log_{10}$ 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, $-\log_{10}(1-0.9)= 1$ 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.

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 Douceur [38] 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 being subject to the possibility of being powered off at the end of a class, contrary to corporate machines that divide in two patterns: daytime and "24 hours". Daytime are machines powered on during office hours, while "24 hours" machines remain powered on for long periods.

5.2 Machines stability

An important factor in resource harvesting concerns machine stability, that is, how long a given group of machines will be available for intensive computation. We define two levels of stability: a light level, where group stability is broken by a machine shutdown or reboot, and a more demanding level, which add to the non-reboot policy, the need for a machine to remain session-free.

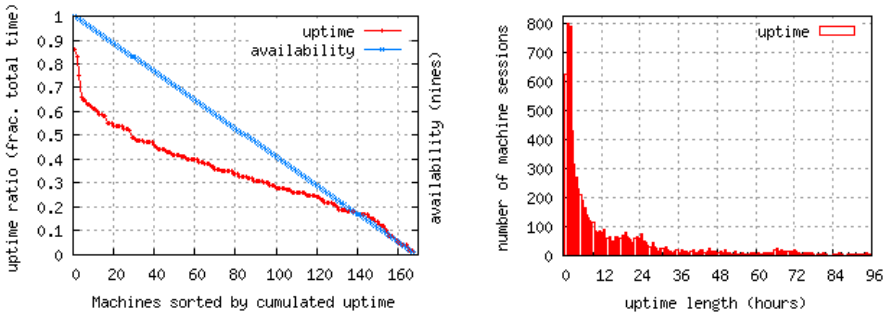

Figure 6: machines uptime ratio and availability in nines (left). Distribution of machines' uptime (right).

5.2.1 Machines Uptime

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

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 our monitoring setup (only one is detected), the above given average duration exceeds the real value. The right plot of Figure 6 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 relatively short, lasting few hours, another indication of the high volatility of machines.

5.2.2 Machine power on cycles

Networked personal computers, especially Windows machines, have a reputation of instability, requiring frequent reboots, for resolving system crashes, completing software installations or simply to refresh system resources. However, it is important to note that one of the main drivers of Windows 2000 development was to reduce the high rate of reboots of its predecessor [39].

As stated before, since our sampling methodology has a coarse-grained granularity of 15 minutes, some of the short machine sessions may go unnoticed. Thus, in order to have a detailed view respecting reboots, we recurred to SMART parameters [31]. By resorting to SMART "*power cycle count*" metric it is possible to spot undetected machine sessions. For instance, if two consecutive samples of a machine have a difference in "*power cycle count*" parameters bigger than one, this means that at least one short machine session, with its corresponding boot and shutdown sequence, occurred without being noticed by the monitoring mechanism.

An important issue of SMART is that parameters vary among disk manufacturers, not only in its availability, but also in units used to represent the metrics [40]. For instance, in the set of machines we observed, two machines had disks that counted "*power on hour*" in minutes using a 16-bit value that overflows after 45.5 days of usage. A third machine expressed this metric in seconds, but using a 32-bit register. Also, another machine had a disk that simply did not provide the *power on hour count* parameter.

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. The number of power on cycles is 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.

Recurring 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. Histogram of average uptime per power cycle grouped by 15 minutes intervals is displayed in Figure 7a) (left plot).

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 (being the oldest 3 years old, and the newest 9 months 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 77-day monitoring. Figure 7b) (right) plots the distribution of average uptime per power cycle considering disks whole lifetime.

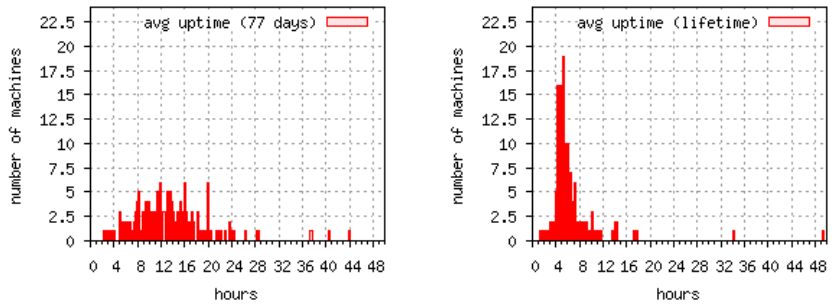

Figure 7: Histogram of average uptime per power cycle observed during the study (left) and along the lifetime of the machines (right).

5.2.3 Group stability

An important issue when executing parallel applications in network of non-dedicated personal computers is group stability, that is, how long a given set of machines will be available for foreign computation. In fact, some parallel environments are very sensitive to changes in the machine pool. For instance, Message Passing Interface (MPI) environments are required to interrupt computation when one or more machines fail [41]. Our definition of stability is similar to Acharya's [25], which characterizes stability of a set of K machines as how long all the K machines remain usable for parallel computation.

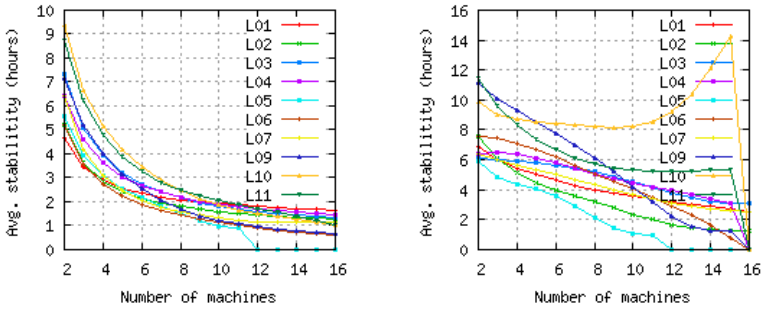

Figure 8: Average stability per classroom, considering powered on machines (left) and user-free machines (right).

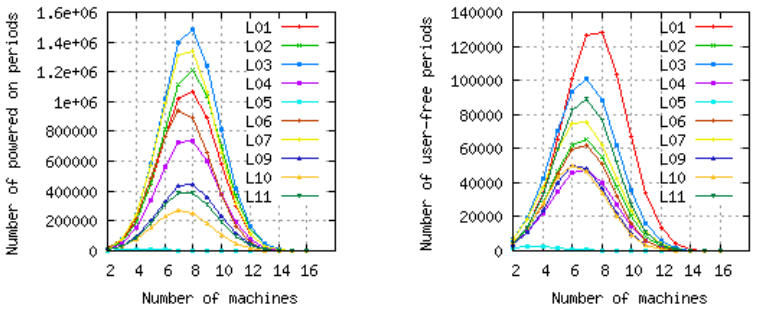

Figure 9: Number of stability periods with N machines. Left for powered on periods, right for user-free periods.

We limited our stability computation to the 10 classrooms that have 16 machines (L08 was therefore excluded). Figure 8 aggregates two plots related to stability for every classroom. Left plot depicts average power on stability length ranging from 2 to 16 machines. All curves present a similar shape with, as expected, average stability decreasing as the number of machines in the set increases. Even so, on

average, the powered on machines stability is quite interesting since that even with 16 machines, almost all classrooms presents an average stability above one hour.

Figure 8b) (right) presents the average user-free stability. This metric differs from average power on stability since it requires that a machine remains user free. That is, if a user logs on a machine, this machine is no longer considered available and thus the stability of the set in which the machine was integrated terminates. The curves for average user-free stability are much more heterogeneous than the power on stability. Surprisingly, the average stability periods are significantly bigger than average length of power on stability period (plots have different Y-scales). In fact, this is mostly a consequence of the number of stable periods, which are, quite logically, much less for user-free periods than for power on periods. For instance, the user-free spike for classroom L10 which presents a 14 hours average stable period for 15 machines is somewhat misleading since its corresponds to a single occurrence.

The number of stable periods for both situations is shown in Figure 9 (left for powered on, right for user-free). The shape of Figure 9 reflects the fact that the number of machine combinations reaches its maximum around 8 machines. Since user-free machines are a subset of powered on machines, the number of stable periods is much higher for powered on machines than for user-free machines.

5.2.4 User sessions

23224 interactive user-sessions from 1684 different logins were recorded along the monitoring period. Average session length was 8942 seconds, a little bit more than 2 hours and 29 minutes. This value is similar to the duration of most classes which last around 3 hours. However, session duration fluctuates widely, with an observed standard deviation of 9083 seconds, roughly 2 hours and 32 minutes. The histogram of the duration of the user sessions grouped by quarters of hour is plotted in Figure 10. The high number of short sessions that exist, with more than 2200 login sessions that lasted less than 15 minutes and slightly less than 2000 lasting between 15 and 30 minutes might be due to students login in for checking their e-mail. Otherwise, most session durations are distributed almost evenly between 30 minutes and 3 hours. For duration bigger than 3 hours, session counts drops smoothly and linearly. The spike of nearly 900 sessions that occurs at the right end of the plot is due to our truncation policy of user-session bigger than 11 hours.

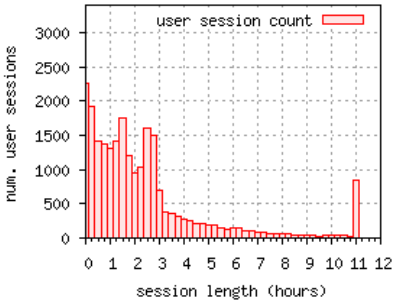

Figure 10: Distribution of user-session lengths.

5.3 Global resource usage

Figure 11 plots averages percentage of CPU idleness over the whole 11 weeks. Left plot displays the CPU idleness when an interactive user session exists, the middle plot represents idleness of user-free machines, and right plot refers to whole machines, independently of the existence or not of user-sessions.

All plots exhibit a similar trend: a convergence near 100% CPU idleness, obviously more noticeable in user-free machines. Weekends are clearly identifiable in all plots, with idleness stabilizing near 100%.

In the plot focusing on used machines (left), some drops in global average CPU idleness below 70% are visible. These drops are mostly caused by the existence of a relatively busy single machine when the number of occupied computers is very low (less than 10 units).

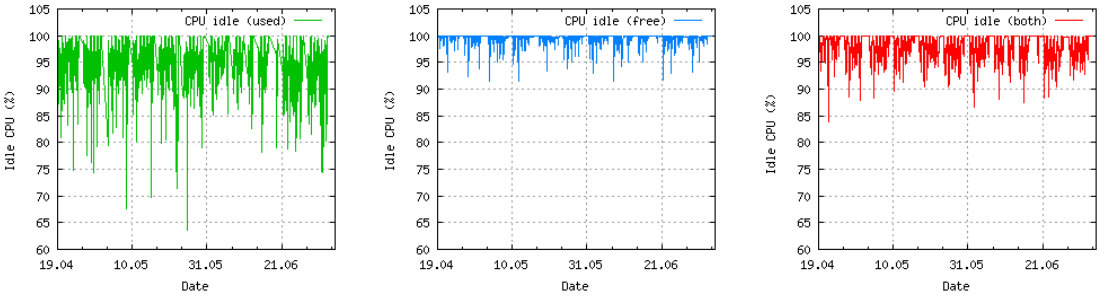

Figure 11: Average CPU idleness percentage while used (left), user-free (middle) and combined (right).

An interesting issue respects idleness of user-free machines (center plot): numerous negative spikes occur, with idleness percentage dropping near 90%, a surprising condition for unused machines. After some investigation, we found out that these negative spikes occur when a significant number of machines are powered on at the same time, for instance, in the first class of the day taught in a classroom. In fact, machine start-up consumes a significant amount of resources and since a user can only log on after a certain amount of time, if the machine is sampled shortly after startup, but before a user has had the opportunity to log on, observed average CPU idleness will be relatively low, and since no user is yet logged on, the machine will be reported as being free. To support our hypothesis, we found that average CPU idleness in user-free machines with less than 5 minutes uptime was 86.05%. The average CPU

idleness percentage ranging from 0 to 5 minutes uptime with no user sessions is given in Table 4 and plotted in Figure 12.

Seconds since boot	Number of samples (no user session)	Avg. CPU idleness (%)
[0,60]	19	77.70
]60,120]	154	83.89
]120,180]	65	86.60
]180,240]	49	90.44
]240,300]	43	91.60
Total	**330**	**86.05**

Table 4: Average CPU idleness right after machine boot with no user logged on.

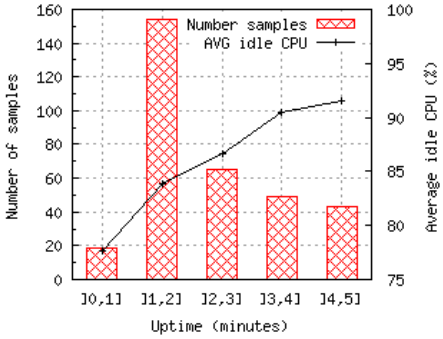

Figure 12: Average CPU idleness right after machine boot with no user logged on. Left y-axis depicts number of samples, right y-axis plots average CPU idleness.

Interestingly, even for machines with interactive usage, average of CPU idleness seldom drops below 75% and mostly fluctuates around 95%. This confirms the potentiality of resources harvesting even when interactive sessions exist as demonstrated by Ryu [28].

Figure 13 shows the sum of free memory in gigabytes over the 11-week observation. Left plot focuses on unused memory of occupied machines; middle plot displays the same metric for user-free machine, while right plot comprises all machines. The plots reflect mostly machines usage pattern, especially the number of powered on machines at a given time. In fact, the high frequency fluctuations in all plots derive essentially from the number of available machines, constituting a proof of machines high volatility.

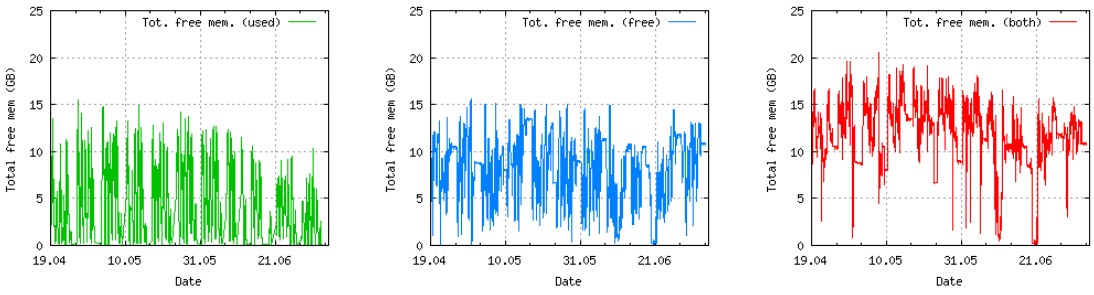

Figure 13: Sum of free memory for occupied machines (left), user-free (middle) and both (right).

Plots of free disk space cumulated from all machines, shown in Figure 14, exhibit the sharp high-frequencies characteristics of high volatile environments. The differences of cumulated available

space between used and unused machines are dependent of the ratio of machines in either condition. On weekdays, especially during work time, since most machines powered on are being interactively used, this set of machines presents normally more than 1.5 TB of available disk space. The cumulated available disk space, even if highly volatile, rarely drops under 1 TB. This is an impressive figure indicator of the dimension of resources that can be exploited in classroom machines.

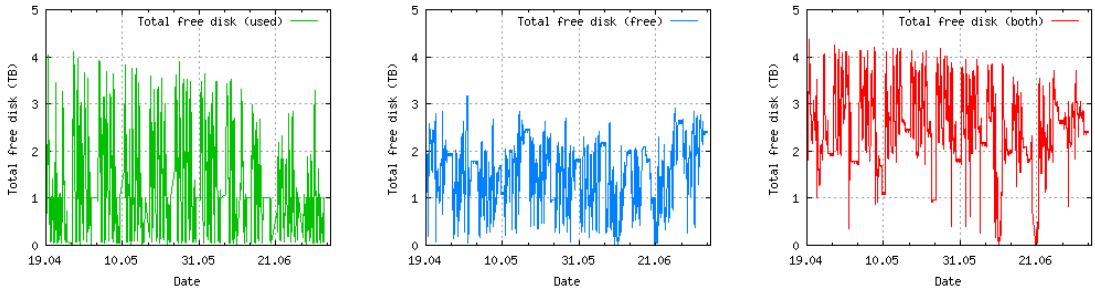

Figure 14: Cumulated free disk space for occupied machines (left), user-free (middle) and both (right).

5.4 Weekly analysis

Figure 15 aggregates two plots related to weekly distribution of samples. Left plot depicts the average number of powered on machines (top curve) as well as the standard deviation (bottom curve). Right plot shows the weekly distribution of average CPU idleness. Both plots reflect the week pattern with stable curves on weekends, especially Sundays, and during the nights.

Besides following the night and weekend pattern, the week distribution of average CPU idleness presents a significant negative spike on Tuesdays afternoons, dropping below 91%. Although we could trace back the Tuesdays afternoon CPU usage spike to a practical class which was taken in a given classroom and consumed an average of 50% of CPU, we could not find the reasons behind this abnormal CPU usage.

Confirming high CPU availability for resource scavenging, 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.

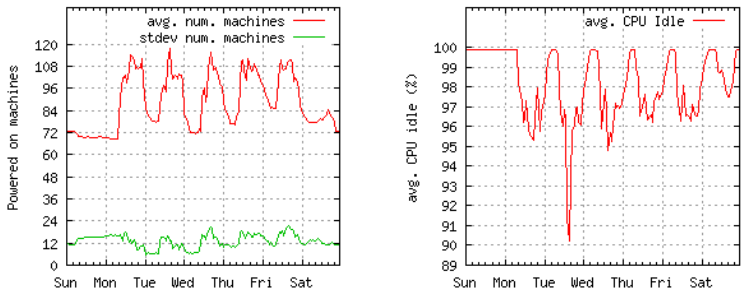

Figure 15: Weekly distribution of powered on machines (left) and average CPU idleness (right).

Figure 16 shows on left plot the average memory distribution along the week with top line representing average RAM load and bottom line showing average swap load. Right plot depicts average network rates for received (top line) and sent traffic (bottom line).

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 reserved 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. This is a natural consequence of how memory systems are organized.

Network rates weekly distributions plotted in Figure 16b) are another example of 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 classroom machines appears clearly visible, with received rates several times higher than sent rates.

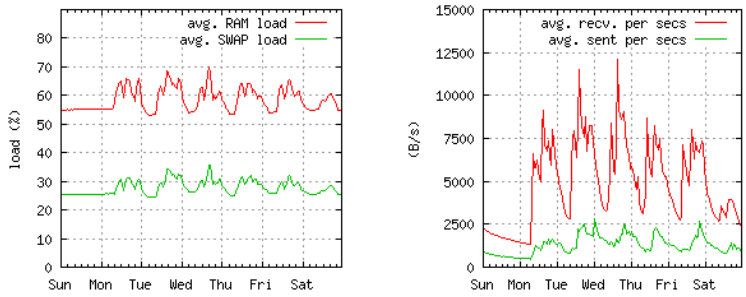

Figure 16: Weekly distribution of memory usage (left) and network traffic (right).

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.

5.5 Equivalence ratio

Arpaci et al. [24] 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. [42] adopt an analogue metric, based upon clusters, and which they appropriately call the cluster equivalence metric. The cluster equivalence aims to measure the fraction of a dedicated cluster CPU that a non-dedicated machine CPU is worth to an application. In this study 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% of CPU idleness is viewed as a dedicated machine with 90% of its computing power. It is important to note that 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). For instance, a machine of L03 (INT:39.29, FP:36.71) is worth 1.19 of a L01's machine (INT:30.53, FP:33.12).

Figure 17 plots the cluster equivalence ratio for the 77-day experiment (left) and its weekly distribution (right). The average cluster ratio is 0.26 for occupied machines and 0.25 for user-free machines. Combining together occupied and unoccupied machines yields a 0.51 cluster equivalence ratio, meaning 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 [24].

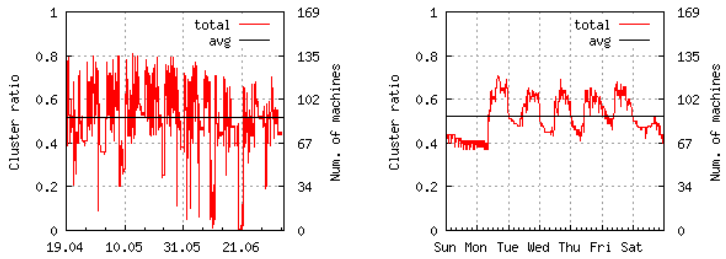

Figure 17: Equivalence cluster ratio over the 77-day monitoring (left) and weekly distribution (right).

6. Conclusions and future work

This report presents the main results of a 77-day monitoring usage study of 11 academic classrooms for a total of 169 Windows 2000 machines. Our study shows that resources idleness in academic classrooms comprised of Windows machines is very high. This confirms previous works carried out in classrooms with Unix machines, and also with Windows server machine [29].

CPU idleness is impressively high with an average of 97.93% observed along the 77-day monitoring study. Also, the 94.24% average CPU idleness measured in user occupied machines indicates that CPU harvest schemes should be profitable not only when a machine is unoccupied but also when interactive usage of the machine exists. This is confirmed by the 0.26 cluster equivalence potentially available from exploiting CPU idleness of interactively used machines and 0.51 when user-free machines are also considered. Another interesting result associated to CPU, is that interactive usage of the machine can be sensed by the level of CPU idleness: CPU idleness above 98% almost certainly means that the machine is not being interactively used, even if a login session exists.

Memory idleness is also noticeable, with respectable amounts of unused memory, especially in machines fitted with 512 MB of main memory. Coupled with a relatively fast network technology such 100 Mb/s switched LAN, such resources might be put to good use for network RAM schemes.

Due to the fact that the single hard disk of all machines only contains the operating system installation plus specific software needed for classes, totalling an average of 13.64 GB used per machine, free space storage among monitored machines is impressive. And with hard disks exponential growth, the tendency is that more and more unused disk space becomes available, at least in sites whose management policy restricting use of shared machines' disks to limited temporary storage. A possible application for such disk space relates to distributed backups, or to the implementation of a local data grid.

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 with minimal disk space for interactive user, and student off-class access to computers for work assignment and communication use.

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", being managed by a central authority. This ease the deployment of resource harvesting schemes, since an authorization from the central authority gives access to the shared machines. Obviously, it is mandatory for harvesting environments to respect interactive users, guaranteeing that interactive usage is not degraded by harvesting schemes.

Another major concern in classroom environments relates to volatility, since a resource available at a given time might disappear right after. Thus, efficient usage of idle resources requires survival techniques such as checkpointing, oversubscription and multiple executions. As expected in any distributed system, security is another major issue to address.

In conclusions, this study confirms that resource idleness observed in classrooms Windows 2000 environments is quite considerable, and carefully channelled could yield good opportunities for grid desktop computing.

As future work, we plan to assess if absence of interactive usage can be sensed by high level of CPU idleness as detected in this study, and if so, what is the threshold value to mark a machine as unused.

Another future task will be to analyse the resource usage of "owned" machines, that is, environments where every monitored desktop machine is assigned to an individual user. The purpose of the study will be to compare the resource usage patterns of "owned" machines against the shared machines found in academic classrooms. We anticipate that a major challenge of the study will be to recruit volunteers, a task that we hope to achieve by a transparent and anonymous monitoring methodology.

Finally, we also plan to use the collected traces in trace-driven simulations to study several scheduling issues concerning desktop grid systems, taking advantage that trace-driven simulations are credited as being more reliable than assessments based on analytical load distribution models [43].

References

[1] E.Pinheiro, R. Bianchini, E. Carrera, and T. Heath, "Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems," presented at Workshop on Compilers and Operating Systems for Low Power, 2001.

[2] 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.

[3] BOINC, "Berkeley Open Infrastructure for Network Computing - http://boinc.berkeley.edu," 2004.

[4] A. Baratloo, M. Karaul, Z. Kadem, and P. Wyckoff, "Charlotte: Metacomputing on the Web," presented at 9th International Conference on Parallel and Distributed Systems (PDCS-96), Dijon, France, 1996.

[5] M. Litzkow, M. Livny, and M. Mutka, "Condor - A Hunter of Idle Workstations," presented at 8th International Conference of Distributed Computing Systems, San José, California, 1988.

[6] D. Ghormley, D. Petrou, S. Rodrigues, A. Vahdat, and T. Anderson, "GLUnix: A Global Layer Unix for a Network of Workstations," *Software, Practice and Experience*, vol. 28, pp. 929-961, 1998.

[7] V. Naik, S. Sivasubramanian, and D. Bantz, "Harmony: A Desktop Grid for Delivering Enterprise Computations," presented at 4th International Workshop on Grid Computing, Phoenix, Arizona, 2003.

[8] B. Richard and P. Augerat, "I-Cluster: Intense computing with untapped resources," presented at 4th International Conference on Massively Parallel Computing Systems (MPCS'02), Ischia, Italy, 2002.

[9] A. Goldchleger, K. Kon, A. Goldman, M. Finger, and G. Bezerra, "InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines," *Concurrency and Computation: Practice and Experience*, vol. 16, pp. 449-459, 2004.

[10] P. Cappello, B. Christiansen, M. Ionescu, M. Neary, K. Schauser, and D. Wu, "Javelin: Internet-Based Parallel Computing Using Java," presented at 6th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 1997.

[11] H. Pedroso, L. Silva, and J. Silva, "Web-based Metacomputing with JET," *Concurrency - Practice and Experience*, vol. 9, pp. 1169 - 1173, 1997.

[12] A. Alexandrov, M. Ibel, K. Schauser, and C. Scheiman, "SuperWeb: research issues in Java-based global computing"," *Concurrency - Practice and Experience*, vol. 9, pp. 535-553, 1997.

[13] G. Fedak, C. Germain, V. Neri, and F. Cappello, "XtremWeb: A Generic Global Computing System," presented at 1st Int'l Symposium on Cluster Computing and the Grid (CCGRID'01), Brisbane, 2001.

[14] 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, pp. 597-610, 2003.

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

[16] Platform, "Platform, Inc. (http://www.platform.com)."

[17] DataSynapse, "DataSynapse, Inc. (http://www.datasynapse.com)."

[18] R. Figueiredo, P. Dinda, and J. Fortes, "A Case For Grid Computing On Virtual Machines," presented at 23rd Int'l Conf. on Distributed Computing Systems (ICDCS 2003), Providence, Rhode Island, 2003.

[19] 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," presented at 14th Symposium on Computer Architecture and High Performance Computing, Brazil, 2002.

[20] 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," presented at AGridM 2003 - Workshop on Adaptive Grid Middleware, 2003.

[21] T. E. Anderson, D. E. Culler, and D. Patterson, "A case for NOW (Networks of Workstations)," *Micro, IEEE*, vol. 15, pp. 54-64, 1995.
[22] A. Gupta, B. Lin, and P. A. Dinda, "Measuring and understanding user comfort with resource borrowing," presented at 13th IEEE International Symposium on High Performance Distributed Computing, Honolulu, Hawaii USA, 2004.
[23] "Browser Usage Stats (http://www.w3schools.com/browsers/browsers_stats.asp-values)," 2004.
[24] 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 joint international conference on measurement and modeling of computer systems, Ottawa, Ontario, Canada, 1995.
[25] A. Acharya, G. Edjlali, and J. Saltz, "The utility of exploiting idle workstations for parallel computation," presented at ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Seattle, Washington, United States, 1997.
[26] A. Acharya and S. Setia, "Availability and utility of idle memory in workstation clusters," presented at ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Atlanta, Georgia, United States, 1999.
[27] W. Bolosky, J. Douceur, D. Ely, and M. Theimer, "Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs," presented at SIGMETRICS international conference on Measurement and modeling of computer systems}, Santa Clara, California, United States, 2000.
[28] K. D. Ryu and J. K. Hollingsworth, "Unobtrusiveness and efficiency in idle cycle stealing for PC grids," presented at 18th Intl' Parallel and Distributed Processing Symposium, 2004.
[29] D. G. Heap, "Taurus - A Taxonomy of Actual Utilization of Real UNIX and Windows Servers," IBM White Paper GM12-0191, 2003.
[30] P. Cicotti, M. Taufer, and A. Chien, "DGMonitor: a performance monitoring tool for sandbox-based desktop grid platforms," presented at Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, 2004.
[31] B. Allen, "Monitoring Hard Disks with SMART," in *Linux Journal*, vol. Nº117, January 2004.
[32] P. Domingues, P. Marques, and L. Silva, "Distributed Data Collection through Remote Probing in Windows Environments," presented at 13th Euromicro Parallel, Distributed and Network-Based Processing, Lugano, Switzerland, 2005.
[33] M.Lavy and A. Meggitt, *Windows Management Instrumentation (WMI)*: Sams, 2001.
[34] M. Russinovich and B. Cogswell, "Sysinternals - PsTools (http://www.sysinternals.com/)," 2004.
[35] U. Mayer, "Linux/Unix nbench project page (http://www.tux.org/~mayer/linux/bmark.html)," 2003.
[36] BYTE, "BYTEmark project page (http://www.byte.com/bmark/bmark.htm)," Byte, 1996.
[37] R. Longbottom, "PC Benchmark Collection (http://homepage.virgin.net/roy.longbottom/#anchorBenchNT)," September 2004.
[38] J. Douceur, "Is remote host availability governed by a universal law?" *SIGMETRICS Perform. Eval. Rev.*, vol. 31, pp. 25-29, 2003.
[39] B. Murphy and B. Levidow, "Windows 2000 Dependability," Microsoft, Microsoft Research Technical Report MSR-TR-2000-56, June 2000.
[40] G. Hughes, J. Murray, K. Kreutz-Delgado, and C. Elkan, "Improved Disk Drive Failure Warnings," *IEEE Transaction on Reliability*, 2002.
[41] "MPI-2: Extensions to the Message Passing Interface (http://www.mpi-forum.org)," 1997.
[42] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien, "Characterizing and evaluating desktop grids: an empirical study," presented at 18th International Parallel and Distributed Processing Symposium, 2004.
[43] P. Dinda, "The Statistical Properties of Host Load (Extended Version)," School of computer Science - Carnegie Mellon University. Technical report CMU-CS-98-175, 1999.