Grid Computing in Organizational Environments

Instituto Superior Técnico – Universidade de Lisboa

Guilherme de Sousa Aranha
guilherme.de.sousa@tecnico.ulisboa.pt

October 2015

Abstract

Unused computation power has increased in organizations due the continuous evolution of hardware capacity and the long periods of time workstations stay in idle. Grid solutions can be seen as a way to scavenge these wasted resources while requiring minimal or none investment in new equipment. However, it is of high priority not to interfere with the staffs' responsibilities while trying to provide this new service. Furthermore, the application of such a system can lead to higher energy costs due to superior machine uptimes, and consequently, more dissipated heat, that will require to be mitigated through air conditioning systems. This work suggests solutions towards integrating a Grid system, while taking into consideration the users' comfort, both temperature and noise wise, as well as attempting to reduce energy costs, applying the ideas in a proof of concept setup.

Keywords: High Throughput Computing, HTC, HTCondor, Sensor Network, Processing, Energetic efficiency, Scheduling

1 Introduction

Nowadays, information systems are everywhere. Regardless of what industry, the dependency on technology is present. More specifically, in organizations, the number of people that work directly with computers is even higher; in many cases, there is a factor of one computer per person, making IT resources proportional to the number of workers.
Nonetheless, it is also common that the computational resources existent in these computers are far greater than the needs of its users that end up using mostly interactive applications that stay idle most of the time, awaiting for input.

Even though the CPU is not receiving any instructions, clock cycles keep happening but, in these cases, they are just being wasted by not processing anything useful for the organization.

1.1 Motivation

Moreover, the time the user is not interacting with the computer (out of office hours, for instance), is also time that these machines are not being used. Assuming they are turned off in these periods, they do not represent an energy waste. However, they are still an investment that is not being used to its full potential.

Taking this into account, it would be of interest to take advantage of these computational resources in order to grant an organization with full capacity of its IT resources. The concept of Grid computing comes as a way to mitigate this misuse.
Much like a cluster, a Grid computing system connects computers in a network in order to process data in a distributed and sometimes parallelized environment, with the difference that it can be both heterogeneous (regarding the equipment used) and geographically dispersed. This project is aimed at exploring ways and adaptations that can be beneficial, in order to integrate this functionality seamlessly in an organizational environment, without disrupting the primarily job of the workstations, which is to serve the local (physical) users.
It is of paramount importance that one understands that the reason by which there is a computational waste is due to the need of the workers to be equipped with a workstation. If, by any means, the suggested system in this project interferes more than it should in their job, the main reason for the existence of those workstations is being jeopardized, making this work not an asset but a liability.
However, the transparency in discussion does not only concern the computer performance. As a consequence of a machine being used (remotely or locally), other factors become relevant. Room temperature increase, power consumption and noise are elements that need to be taken into consideration, since they also have impact on the local user.

1.2 Objectives

The main objective of this work is to present solutions to the problems stated above; primarily energy efficiency measures and non-intrusive behavior for the local users. In order to reduce energy consumption, a method to efficiently manage what machines should be woken up or turned off must be developed, as well as a temperature sensor network to input the grid with more environment knowledge about the room temperature where the workstations are located.

1.3 Document Structure

This document is divided in 7 main parts. The first one being the introductory section where it is given an overview of the problem and the objectives of this work in order to try to mitigate the non-desirable behaviors. Section 2 reviews related work on grid computing and its energy efficiency. Section 3 characterizes the system domain while proposing the needed alterations in desktop grids. Section 4 describes the experimental setup used to demonstrate the desired functionality. Section 5 estimates the costs of running an air-conditioning system for the case study environment and analyses the obtained results from the collected data. Section 5.2 interprets and comments the acquired data while revealing the most pertinent findings (Section 5.2.1). Section 6 concludes this document, with some insight on what more can be done in this field of work.

2 State of the Art

This section summarizes the most relevant research on the field of grid computing and the techniques developed to improve energy efficiency regarding this field. It starts with a an approach to classify and characterize the most pertinent differences between the existent systems, passing to a contextualization of the concept of High Throughput Computing. It concludes with some proof of concepts for energy efficient server clusters.

2.1 Characterizing and Classifying Desktop Grid

Choi et. al propose a taxonomy [1] for classifying desktop grids by organization, platform, scale and resource provider, as shown in Figure 1.

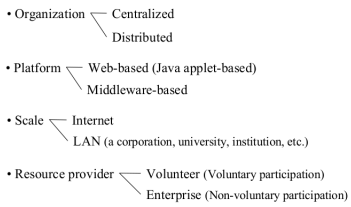

Figure 1: A taxonomy of a Desktop Grid [1]

The type of organization can either be “Centralized Desktop Grid” or “Distributed Desktop Grid”. In the first type, the grid is composed by clients, resource providers (or volunteers), and a server. The client submits a job to the server responsible for scheduling it to a resource provider. When the job is done, the resource provider returns the result to the server which subsequently returns it to the client. On the other side, in case it is distributed, there is no need for the centralized server. Each resource provider has knowledge of other resource providers (partial or complete). In a distributed way they are responsible for scheduling the received job from the client to an available resource provider that fits its needs. Platform-wise, there are both web and middleware based. When the platform is web-based, the client writes its job in Java and posts it as an Applet on the Web. Volunteers access the web page with their browsers that will automatically download the Applet and run it in their own machine. In middleware-based desktop grids, a service has to be installed in the resource providers. This makes the machines available to run the clients' submitted jobs. Grids can be scaled to be used in LANs or on the Internet. If scaling the system to the Internet, it becomes based on anonymous volunteers to run the jobs. This type of desktop grid is the one presenting more heterogeneity, most trust issues (malicious volunteers) and failure. In LAN based, the desktop grid is within an organization. In this case, the connectivity is more stable and the heterogeneity level is well known.
Finally, there are two models for resource providers: Volunteer and Enterprise. The first one is based on resource providers that voluntarily allow jobs from the grid to run on their target machines. When a desktop grid is Internet based in terms of scale, their resource providers are always from this type since there is no obligation to contribute to a grid. Typically, this type of system is more vulnerable to malicious responses, adding the need to some kind of result validation. In the second one, the resource providers are located in the same administrative domain, resulting in a more trustworthy system.

2.2 High Throughput Computing

With the need to process large volumes of data for long periods of time, there is a research field in distributed systems dedicated to the study of this type of problems; this area goes by the name of *High Throughput Computing* (HTC).
Contrary to *High Performance Computing* (HPC), in HTC, the objective is not to process the most amount of data in the shortest period of time (measured in floating point operations per second, *FLOP*), but to process large amounts of data in a month/year time period.
Even though, the use of HTC is more common in research and academic fields, with the rise of techniques like *Big Data* for instance, the integration of this type of systems can be of great value, for organizations that may need equivalent processing power.

2.2.1 Previous work

In “Experiments on Computation and Storage Scavenging in Institutional Environments using HTC and HPC Solutions”, R. Bruno *et al.* [2] explore a combination of cluster service configurations to harvest underutilized computational resources in university laboratories. A brief description of the environment, chosen services and their advantages is made, while explaining the path that was taken on the development of the current system state.
The system offers two main services: computation and storage. Both are delivered to the users through the cluster, supported by unused resources of the computers present in the labs. For HTC services, the first candidate was HTCondor which presents to be a system that consumes unused resources while re-

specting local users. The second one was Apache' Hadoop [3], an implementation of MapReduce programming model, highly used in processing and manipulating large amounts of data. Regarding HPC, the chosen solution was Open MPI [4], an MPI (message Passing Library) implementation, oriented for parallel computation.
In terms of priority over the computational resources, the local user comes first, since it requires more interaction with the machine, and also because the existence of that equipment in first place, is due to its needs. Respecting this requirement would be easy if a conservative approach was taken, limiting the available resources of the cluster to a point where it would never interfere with the local user. However, this would still result in considerable wasted computational power. The second objective of these experiments was to achieve a solution with enough elasticity in order to adapt the cluster available resources to the local workload. Therefore, this work represents a continuation of this project, aiming to bring new ideas, improvements and concerns that lack on this first iteration.

2.2.2 HTCondor

HTCondor is an opensource HTC middleware, developed in the University of Wisconsin–Madison. It can be used both in clusters and grids, with mechanisms to scavenge cycles from desktop computers in a non-intrusive way to the local user [5]. Because of this characteristic, I have considered it to be the most adequate solution for the study of my work.
From all the features HTCondor presents [6], there are three that I find more interesting for the environment in question: checkpoint and migration, sensitive to the desires of machine owners and *ClassAds*.

1. HTCondor assures job completion. In case of a node crash (where a certain job is being computed), HTCondor migrates the job to another machine as soon as one is available. In order to safeguard the already done computation, checkpoints are done.
2. When working with desktop grids, HTCondor can detect interaction of the local user with the computer, in order to halt the job on that machine. Local users can be positive about letting their machines do some computation when the computer is idle, but not when they are interacting with it.
3. *ClassAds* are a set of uniquely named expressions that provide a way to communicate with HTCondor. Through them, users can describe job requirements in terms of computation requisites (quantity of RAM, for instance) and computer owners can specify constrains on which remote users have priority to run jobs, or even which users are allowed to run them on that target. The combination of possible policies is extensive.

2.3 Energy efficient server clusters

In 2003, Elnozahy et. al. [7] evaluated five policies for reducing energy consumption on server clusters. The evaluation was supported by a simulation model of a web server cluster.
The policies are based on two mechanisms: dynamic voltage scaling and node vary-on/vary-off. As the names suggest, one characterizes itself by dynamically scaling the CPU voltage, and the other by turning the nodes on and off according to the needs of the clusters.
Later on, in 2011, Valentini et. al [8] investigate similar methods although suggesting a more load balancing-aware system. With load balancing techniques, the idea is to distribute the workload evenly between all the available nodes on the cluster. This way, we assure the lowest frequency possible is needed in all the machines, since ideally, they are doing as little computation as possible.
Just like Elnozahy et. al [7], Valentini et. al [8] also recommends that in case nodes are idle, policies for turning them dynamically on/off be applied.

2.4 Summary

High Throughput Computing has been around for over a decade now and is currently a standard technology in a wide range of research centers and universities. Desktop grid environments however, are not as common. Due to this, most of the work and development in energy saving systems is purely theoretical and test cases without real world and size application.
In what concerns HTC, HTCondor (Section 2.2.2) does a fair job deploying a grid that can satisfy the remote users needs without getting in the way of the physical desktop owners. Not only it offers a good default setup, but it also empowers the system administrator with a wide variety of configuration parameters and the possibility to expand its functionality.
From the energy efficiency point of view, some trials have been made to dynamically wake only the needed nodes and turn them off when they are not necessary. At the same time, a centralized voltage scaling system has been applied to coordinate the voltage between the machines stating that it could save some energy.

3 Grid Computing in Organizational Environments

Based on the related work previously mentioned and the objectives of this project (Section 1.2), a solution proposal is presented in the following section.

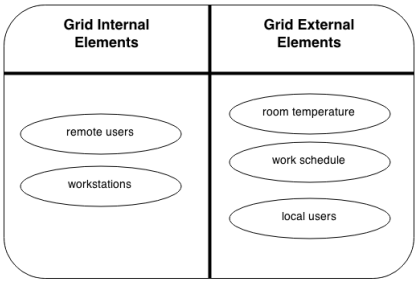

Figure 2: Grid Domain

3.1 Grid Internal Domain

The internal domain represents the well-known domain from the grid's perspective: it is aware of the remote users and the jobs they are submitting, as well as the available workstations and their resources.

3.1.1 Remote Users

Remote users can be described as the client of the system. They access the cluster from any location (from inside the organization or from the outside) using an SSH (secure shell) session and submitting their jobs to the grid. The interaction is done exclusively through command line interface, and after job submission no more input is possible. Considering this, the need for fast feedback is not necessarily our target, compared to a local user that expects the system to react almost automatically to interaction.
This is an important characteristic of this type of user because it lowers the importance of providing an highly responsive system, as long as the computation ends at some point with the wanted results.

3.1.2 Workstations

From the grid's perspective, the workstations are seen as nodes, responsible for computing the jobs delivered by the system (requested by the remote users). These computers exist primarily as work tools for the local users, but the unused resources are also made available to the grid.

3.2 Grid External Domain

External elements represent the components that influence the grid in a non-predictive way, and typically grids do not take them into consideration. In this domain I include the local users and their work schedule as well as the room temperature were the workstations are located. It is from this side were my work introduces its functionality.

3.2.1 Local Users

The availability of computational resources in the grid, directly depends on the type of activity the local users have on their workstations.

If a local user requires more computational power of its workstation that is currently being scavenged by the grid, one or more jobs (depending on the workload running) must be transferred to another node. On the contrary, if the local user is using little or no resources (in case it is absent) like working in mainly interactive applications (such as text processing or internet browsing), it can be considered a well suited node to run a job. Whenever possible, the grid should choose workstations that are not being used.

Work Schedule

Integrating a grid system into an organizational environment can be problematic if we consider that some time-frames can have higher priorities, as the local users can tend to do specific tasks in certain times of the day or week. Furthermore, these schedules may not apply equally to all users, or all sets of computers, limiting the availability of some.
As such, having a system that aggregates the work schedule of all the local users can reveal to be an important asset for a better elasticity of the system, providing the grid with more knowledge for better decision making in distributing the workload.

3.2.2 Room Temperature and Sensor Network

The use of computational resources in a workstation directly influences the dissipated temperature. Therefore, implementing a grid system to scavenge unused resources will result in more dissipated heat.
An increase in the machine temperature does not only affect the hardware, but also increases the room temperature and noise (workstations are air-cooled). As a result, local users tend to resort to air-conditioning systems increasing power consumption.

The solution proposed for this problem consists on implementing a sensor network which is responsible for measuring the room temperature and upload it to a database. When the grid receives a new job from a remote user, it consults the database and takes into consideration the temperature of the room for scheduling (Figure 3).
Scheduling jobs to machines that reside in rooms with lower temperature produce a double effect: it does not only prevent hot rooms from getting hotter, but it also heats rooms that are colder. This means that, for instance, in the summer the system will reduce the air-conditioning use by avoiding heat, and in the winter it will be also heating rooms that have lower temperatures. It is the same functionality, the same configuration, with opposite conditions.

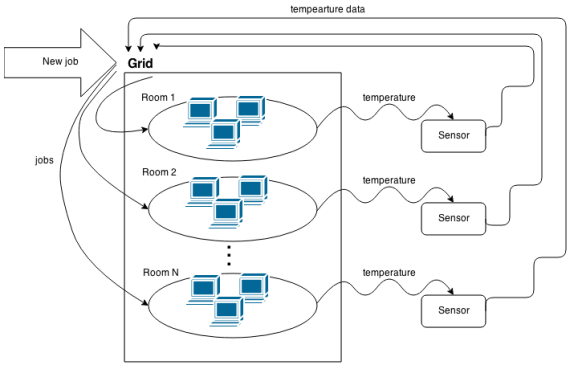

Figure 3: Temperature Sensor Network

3.3 Classifying the Grid

Based on the taxonomy referenced in Section 2.1, and the choices I have presented, the grid can now be formally classified.
Choosing HTCondor as a platform, already characterizes a part of the system; it's a centralized system from the organization perspective, middleware-based, and scales on LAN. However, the type of resource provider we have cannot be specified with a single option. Depending on the privileges of the local user we are dealing with, the resource provider will be considered "Vol-

unteer" or "Enterprise". Some users may be exclusively assigned to basic and light tasks like text processing; in this case we assume the "Enterprise" model. In other cases, due to the non-existence of a profile that fits the local user needs, the type of use may be too difficult to predict, and the system leaves the choice of volunteering to the local user.

3.4 Case Study

As a way to support and test the ideas proposed in this document, I will be implementing them in the RNL (Rede de Novas Licenciaturas) laboratories at Instituto Superior Técnico. In this scenario, students using the labs (named "rooms" until now) can be seen as the local users of the system, while the remote users can be anyone with an user account at Instituto Superior Técnico. The workers' work schedules can be easily mapped to the class schedules of each lab. The case study should be seen as a way to evaluate and test some of my work, as a proof of concept model.

3.4.1 Infrastructure

The grid comprises 90 machines distributed across 8 laboratories. It can be considered a relatively homogeneous system, as there are only two workstation configurations:

1. Type 1: Intel(R) Core(TM) i5 CPU 760 @ 2.80GHz, 8GB RAM, NVIDIA GeForce GT 220, 500GB HDD

2. Type 2: Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz, 8GB RAM, NVIDIA GeForce GT 630 D3, 1TB HDD

Roughly speaking, we are talking about a total of 360 CPU cores, 720GB of RAM memory, and 60TB of storage. Furthermore, both type 1 and type 2 are equipped with GPUs supporting CUDA technology [9] [10].

3.5 Summary

Based on the state of the art presented on this document and the current section, I can say it is clear that external domain attributes and variables have been disregarded and ignored.

Room temperature influences both machines and local users directly. From the perspective of the machine, it can accelerate the degradation of components, becoming the most crucial factor in this matter if we ignore the tear down of hard-disks due to their mechanical nature. On the other hand, local users are also influenced by temperature and can become unhappy if room temperature rises considerably due to remote users' workload on the grid. This brings us two a third factor, which is the extra energy consumption needed to cool the rooms in order to create a more "healthy" environment to both local users and desktops so they can cohabit in the same place.

4 Experimental Setup

Towards supporting the concepts referenced in the previous section, a proof of concept was developed. From the hardware standpoint, the main objective was to achieve a solution that would allow me to collect in real time the temperatures of the laboratories to provide the grid with this information. Such setup would ideally allow integration with a database to store the collected data for further analysis; be scalable to all the laboratories (and even more if possible, for bigger infrastructures); as affordable as possible.

There are multiple systems that can be deployed to achieve this result. The most simple one would be to acquire an "out of the box" temperature sensor with an ethernet interface; at the same time, it would also be the most expensive one and presenting more limitations. The price of one of these devices is currently superior to the price of the whole system I am presenting, and would require to be multiplied by the number of rooms/labs whereas mine does not. The functionality would also be limited to the specifications determined by the manufacturer (database support, number of readings per time, where data is saved, future updates, etc).
Therefore, I found that the best solution would be to rely on a microcontroller or a single-board computer. Considering that single-board computers are very cheap these days, expandable and able to run an operating system such as Linux, I decided to take the second option.

After exploring the Raspberry Pi and its GPIO (General Purpose Input/Output) interface it seemed like a good start as it had all the needed features and a fair price. At first I considered the possibility to install one of these equipments per room, but that would imply more costs and more systems to administrate. Alternatively, after some thought, installing the sensors through the CAT5 infrastructure along the building seemed more adequate.

4.1 Sensor deployment

There were two possible sensor protocols: I2C (Inter-Integrated Circuit) [11] and 1-Wire [12]. Apart from VCC and Ground, I2C uses two more lines, one for clock sync (SCL – Serial Clock) and another for data transfer (SDA – Serial Data). In a simple I2C system, there is one master (CPU or microcontroller) and one slave (in more complex implementations the slaves can scale to higher numbers) the master being responsible for the SCL line; the SDA line depends on the type of device we are connecting. If it is a readable device like a sensor, the slave is responsible for writing to the SDA; if it is a writeable device, such as an LCD screen, the master does the writing and the slave the reading.

With the need for two channels for data and clock, I2C is not well suited for long distance implementations as the clock tends to loose sync easily and the writes and reads on the SDA fail. After understanding this, I went on to investigate the 1-Wire protocol which I will explain next.

As the name states, the 1-wire protocol works with only one line (plus VCC and Ground). Contrary to the I2C protocol, 1-wire supports much longer distances. In shorter distances, it can also be wired with only

two wires since the devices include a capacitor to store charge supported by the data line (parasitic mode).

1-Wire has support to connect multiple devices to the same bus, as every device has a unique embedded 64-bit identifier. This way, the master can query (read) or write to any device by specific request. For error detection, the protocol also includes an 8-bit CRC(cyclic redundancy check).

Writes

In order to write data to a slave, the master starts by pulling the data line to 0 volts by connecting it to ground. This procedure is considered a reset pulse and lasts for at least 480μs. After that, the sensor(s) connected to the bus inform the master about their existence with a pulse (the bus is lowered for another 60μs). Next, if the master intends to write a 1 it lowers the bus to 0 volts for 1 to 15μs; if it is a 0 the procedure is the same but instead it waits for 60μs.

Reads

In case the slave device is a readable device, the reset and presence procedure still applies. When the master wants to receive its data, it sends a 0 volt signal in the bus for 1 to 15μs and waits for the slave's response. In case the bus turns high, it is a 1; in case it stays low for 1 to 15μs, then it represents a 0.

DS18B20 Temperature Sensor

The reason for using the DS18B20 [13] over other sensors was mainly due to being the most commonly used sensor implementing the 1-Wire protocol. It offers a fair precision of 9-bit to 12-bit Celsius temperature measurement, operating in a range between -55°C to +125°C, with an assurance of ±0.5°C inside the range of -10°C to +85°C.

It needs at least 3V to be stable and tolerates a maximum of 5V. For the assembly, a 4.7kω pull-up resistor needs to be placed between the power line and data bus.

The DS18B20 module (Figure 4a) has the advantage of already including the pull-up resistor, which not only simplifies the circuit, but can also reduce the downtimes drastically. In a setup with multiple sensors attached to one pull-up resistor, if it fails, the overall system goes down, while using a resistor per sensor reduces the chance by only failing one sensor at a time. In addition, it also provides a LED that informs that the module is working which is useful for debug. Alternatively, the resistor could also be assembled directly to each sensor, but that would result in a much more fragile system in case of someone touches the hardware by accident (for instance, passing by or dragging a chair near it). A prototype of the module assembled in a RJ45 connector through a CAT5 cable can be seen in Figure 4b.

Attempts, Problems and Solutions

Even though the 1-wire protocol presents itself as a good solution it still has some weaknesses. I learned from trial and error that the protocol is not very well

(a) DS18B20 module

(b) DS18B20 attatched to RJ45

Figure 4: DS18B20

suited for star-topology networks, which was what I was using (it is recommended to use a unique bus with the sensors connected along with it). The Raspberry Pi (and all its competitors I have analyzed) only have support for one 1-wire bus, and as soon as I tried to connect more than 2 sensors they started to act abnormally. Sometimes, some of them did not show up, in other occasions all the readings were wrong. At first, I thought it could be due to the weak connection through a breadboard and cabling so I soldered a stripboard with the same topology, with no success.

After some research, I found a shield from a British manufacturer (Sheepwalk Electronics) that was designed to be coupled to the Raspberry Pi (Figure 5) and would multiplex one of its I2C interface to multiple 1-Wire, giving me now a total of 8 independent 1-Wire buses. Since I only needed 7, this fitted my needs perfectly. From my experience, in cases where more sensors are needed, two sensors per bus work correctly, which enables the system to scale to 16 sensors. If there is a need for more, multiple Raspberry Pi's are also a possibility, as long as they are writing to the same database.

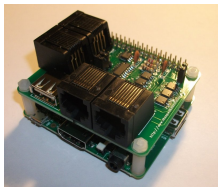

Figure 5: SheepWalk Electronics shield coupled with the Raspberry Pi

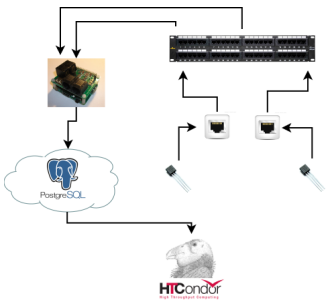

Figure 6: Connection Schematics

4.2 HTCondor Configuration

In order to interconnect the sensor network and the lab schedules with the cluster some extra configuration and scripts were added. Both the master and the nodes

needed adjustments.

In the nodes, two new *ClassAd* attributes were added: one to store the room temperature where the machines reside, plus a script to update the value from time to time (since the sensors write to the database in a 1 minute period I maintained this number); and another to mark the computer as being located or not in a lab where a lecture is happening.

For the master to take into consideration the temperatures and schedules and therefore opt, when possible, to forward the jobs to nodes where room temperature is lower and without lecture activity, the `NEGOTIATOR_POST_JOB_RANK` equation was changed to incorporate the *ClassAd* attributes announced by the nodes.

```
STARTD_CRON_JOBLIST = LAB_TEMPERATURE
STARTD_CRON_LAB_TEMPERATURE_EXECUTABLE = $(LIBEXEC)/labTemperature.sh
STARTD_CRON_LAB_TEMPERATURE_PERIOD = 60s

STARTD_CRON_JOBLIST = IN_LECTURE
STARTD_CRON_LAB_TEMPERATURE_EXECUTABLE = $(LIBEXEC)/inlecture.py
STARTD_CRON_LAB_TEMPERATURE_PERIOD = 300s
```

Figure 7: Configuration added to the nodes HTCondor configuration file

In Figure 8, the *Negotiator* starts by verifying if the node is running a job (`RemoteOwner =?= UNDEFINED`). If positive, the result of the expression results in 0, otherwise, if the node has no job running, the *Negotiator* will prioritize the node by its number of FLOPs (higher FLOPs produce a higher rank position). Finally, three negative factors are added to the rank value: (1) in case the node was asleep; (2) regarding room temperature; (3) and whether there is a lecture or not. As a safety measure, both new values (`LAB_TEMPERAETURE`, `IN_LECTURE`) are checked; in case they are undefined (something went wrong with the execution of the script, for example), a conservative value of 25°C is used for the temperature, and it is considered that the lecture did not occur.

Using `NEGOTIATOR_POST_JOB_RANK` instead of the `NEGOTIATOR_PRE_JOB_RANK` enables the system to take room temperatures and the schedules into consideration without deprioritizing all the performance variables that the HTCondor should evaluate.

```
NEGOTIATOR_POST_JOB_RANK =
(RemoteOwner =?= UNDEFINED)*
(ifthenElse(isUndefined(KFlops), 1000, Kflops)
-SlotID - 1.0e10*(Offline=?=True)
-1.0e10*(ifthenElse(isUndefined(LAB_TEMPERATURE)), 25, LAB_TEMPERATURE))
-1.0e10*(ifthenElse(isUndefined(IN_LECTURE)), 0, IN_LECTURE)))
```

Figure 8: Post job rank negotiator expression in HTCondor configuration file

4.3 Green Wake-on-Lan

Wake-on-Lan (also known as WoL) is a standard technology nowadays, being present in almost every desktop motherboard. Through a broadcasted packet containing the target machines MAC address delivered on the network (named magic packet), it allows the user to remotely wake a computer. Many systems, such as clusters use this technology to dynamically turn on nodes when the workload demands it. As a basic procedure, the choice of what machine to wake is typically random. However with further configuration, choosing better suited targets can be achieved (for instance, choosing a machine with lower/higher performance).
What is here suggested is the functionality of waking machines located in rooms with lower temperatures, avoiding or delaying the need of additional air conditioning. Figure 9 provides further detail on how it works.

```
function greenWakeonLan
    _labTemperatures[] = queryDB();
    sortLabsByTemperature(_labTemperatures);
    for all lab in _labTemperatures do
        for all pc in lab do
            if isOFF(pc) == true then
                wakeOnLan(pc);
                sleep(60s);
                if isON(pc) == true then
                    exit;
                end if
            end if
        end for
    end for
end function
```

Figure 9: Green Wake-on-Lan logic

From another perspective, GreenWoL also allows the grid to take advantage of working air conditioners as it will mostly wake computers nearby these equipment mitigating the need for more.

4.4 Summary

In theory, deploying a temperature sensor network is straightforward. However, in practice, it revealed to be more problematic. Scaling it to a wider setup with long distances revealed to be a much harder task than first expected.
Choosing the main technologies was not problematic per se; the complications came next. Deploying the sensor network with only 1-wire bus was a serious limitation and was preventing me to continue my work. Fortunately I found the Sheepwalk shield that allowed me to get more buses for the needed protocol and everything ended up working as initially expected.
There were three main changes/additions to the system: (1.) redirect jobs to nodes in colder rooms; (2.) wake nodes in colder rooms when the workload demands it; (3) prefer nodes not used by lectures at the current moment.

The result ended up as a remarkably cheap system (less than €120 of hardware at current market price) with the advantage of being a lot more flexible than any out of the box temperature sensor network solution that could be found, considering that any thermometer with an internet connection costs more than the stated price, with the need to be multiplied by every room.

5 Evaluation and Results

This section starts by explaining how to apply the British Thermal Unit to the case study and calculating

the corresponding energy costs. Sizing the air conditioner correctly is important for a good refrigeration but also ensures this equipment will not be overworking and wasting unneeded energy, which can happen when it does not have enough capacity. Then, an analysis based on the preliminary collected data is done, with some observations and recommendations for energy saving.

5.1 British Thermal Unit

The amount of heat generated by a certain closed system (a system that does not consider exterior lighting and its associated heat)is known as heat load.
Heat load can be calculated both in kilowatts(kW) or in the British Thermal Units (BTU), but I will be using the latest one since it is the most commonly adopted by the industry. Factors that influence the heat include:

- Room size;
- Heat generated by occupants in the room;
- Dissipated heat generated by electronic equipment;
- Dissipated heat generated by artificial light;
- Room positioning (for sunlight consideration).

For a matter of simplicity I will be overlooking both sunlight and artificial light in the room in the following calculations.

Room size

For calculating the BTUs generated for a certain room we apply the following formula:

$$RoomArea(BTU) = Length(m) \times Width(m) \times 337BTU$$

Occupants

For calculating the BTUs generated by N occupants:

$$Occupants(BTU) = N \times 400BTU$$

Electronic equipment

For calculating the BTUs generated by the electronic equipment, for N computers:

$$ElectronicEquipment(BTU) = N \times WattagePerComputer \times 3.5BTU$$

The conservative value of 300 watts was estimated. The value was calculated by the *"ASUS Power Supply Wattage Calculator"* [14] based on the the hardware specifications of the workstations.

Total Heat Load

Therefore, to calculate the total heat load we must sum all the previous BTU's:

$$TotalHeatLoad(BTU) = RoomArea(BTU) + Occupants(BTU) + ElectronicEquipment(BTU)$$

The values here stated are reference values calculated by the ASHRAE (American Society of Heating, Refrigerating, and Air-Conditioning Engineers) [15] [16] for air conditioning sizing.

5.1.1 Applying BTU to the Case Study

Predicting numbers or even estimations of how much energy can be saved by applying the suggested concepts is fairly difficult. The environment is non-deterministic in every aspect and does not allow me to directly compare scenarios where the system runs with and without the needed alterations. At the same time, that is what makes the system much more interesting as it has to dynamically adapt to the circumstances.

Lab	Size (m^2)	PCs	Room Area (BTU)	Electronic Equipment (BTU)	Total Heat Load (BTU)	kWh	Cost (€)
6	30	9	10110	9450	19560	5.732	0.734
7	32	9	10784	9450	20234	5.93	0.759
8	34	10	11458	10500	21958	6.435	0.824
10	35	10	11795	10500	22295	6.534	0.836
11	70	22	23590	23100	46690	13.683	1.751
13	46	10	15502	10500	26002	7.62	0.975
14	64	20	21568	21000	42568	12.475	1.597

Table 1: BTU Estimation per Lab

Therefore, I cannot precise an average kWh saving; although with all the context presented in this document I expect it to be empiric and deductible that the needed energy will be reduced. Table 1 elaborates some values on the average BTU consumption running an ideally dimensioned air conditioning system to each laboratory needs. Table 2 calculates the associated costs (the cost per kWh was taken from the EDP – Energias de Portugal website [17]).

Lab	Cost(€) Hour	Cost(€) Day	Cost(€) Month	Costs(€) Year
6	0.734	17.616	528.48	6342
7	0.759	18.216	546.48	6558
8	0.824	19.776	593.28	7119
10	0.836	20.064	601.92	7223
11	1.751	42.024	1260.72	15129
13	0.975	23.4	702	8424
14	1.597	38.328	1149.84	13798

Table 2: Associated costs of air conditioner based on the estimation

5.2 Data Analysis

By carefully looking at the generated charts [18] and crossing them with the class schedules, the misuse of energy through the air conditioner becomes very clear. In this section I will try to make it somewhat more obvious by analyzing some specific cases. In the future, with more complete collected data, a more deepful analysis could bring further conclusions, but for now, with the preliminary data, this is what I could come with so far.

Lab 8

By looking at Figure 11, Lab 8 presents itself very regular. The inside temperature values follow very closely the outside temperatures regardless the expected resistance to some minor changes. Despite some one or two very short occasions, the air conditioning system seems to be off all the time even during lecture periods (Figure 12).

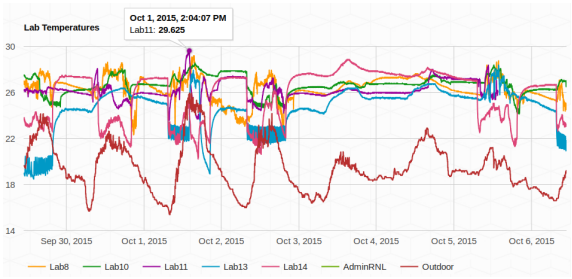

Figure 10: Data from 29/09/2015 to 06/10/2015 (1 week) [18]

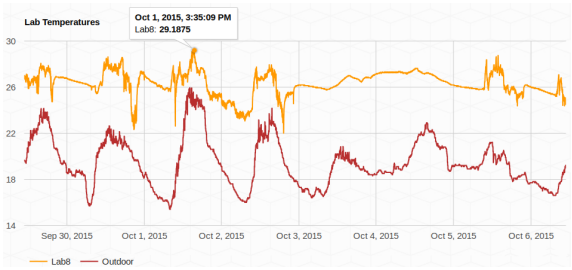

Figure 11: Lab 8 Data from 29/09/2015 to 05/10/2015 [18]

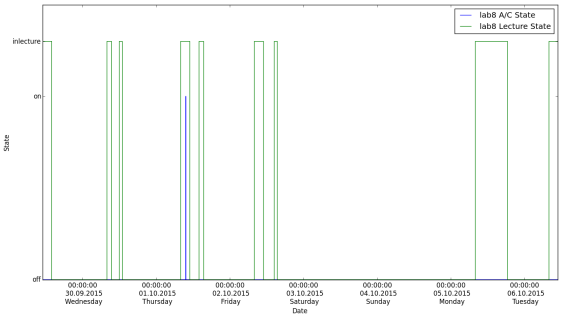

Figure 12: Lab 8 Air Conditioner State - 29/09/2015 – 30/09/2015

Lab 10

In terms of size, equipment and location in the building, Lab 10 is very similar to Lab 8 (for more detail check the plant in the appendix) with the exception of only having one facade, which reduces the cooling capacities through window airflow. Consequently, the need for artificial refrigeration increases which is revealed by an increase in the air conditioner use. Through the extrapolation (Figure 14) I calculated approximately a period of 20 hours of air conditioning used outside the lectures. It is always possible that the lab was being used by students since both Lab 8 and Lab 10 can both be opened by anyone (typically by this order), but it just seems the air conditioner was left on from the last lecture a few times.

Lab 11

Lab 11 is both the largest lab and the most populated in terms of available machines (22 computers total). It is normally only used for lectures, and considering its size and student attendance it is the one consuming less energy for refrigeration. This is probably due

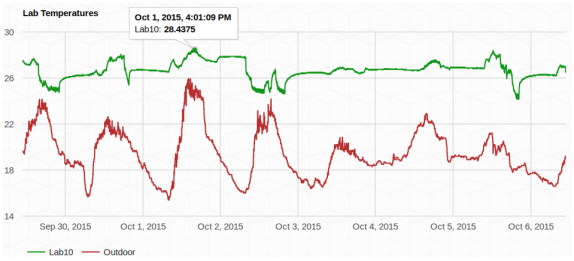

Figure 13: Lab 10 Data from 29/09/2015 to 05/10/2015 [18]

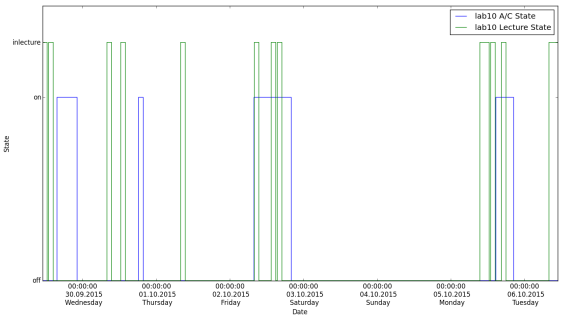

Figure 14: Lab 10 Air Conditioner State - 29/09/2015 – 30/09/2015

to it being in a more shady spot of the building and having two facades which allows for a better airflow, regulating the ambient temperature through the night. In this period, it used approximately 17 hours of air conditioning outside classes.

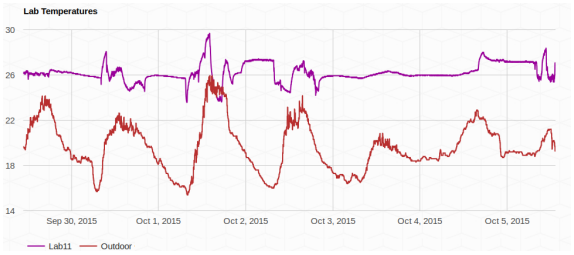

Figure 15: Lab 11 Data from 29/09/2015 to 05/10/2015 [18]

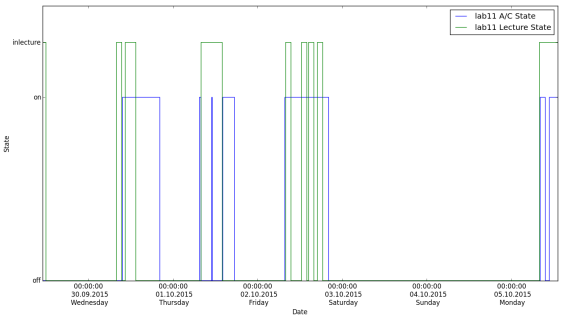

Figure 16: Lab 11 Air Conditioner State - 29/09/2015 – 30/09/2015

Lab 13

From the Lab 13 schedule and the air conditioner state analysis (Figure 18) there is a clear misuse of the the refrigeration system. Even though, there are no lectures most of the time, the air conditioner is always on trying to keep the room at an average of 18.5°C. As already stated, electronic equipment deal best with lower temperatures, however, there is no need to keep the room this cold especially if there where no users present; if they were, this wouldn't even be adequate as it is too cold. The lab totals 24 hours of meaningless air conditioning use.

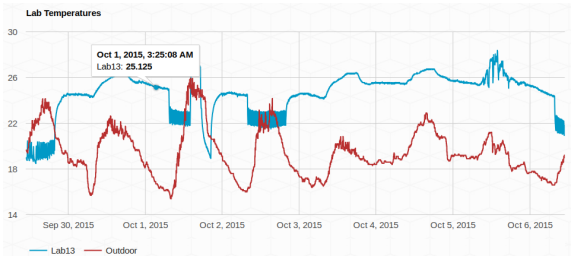

Figure 17: Lab 13 Data from 29/09/2015 to 05/10/2015 [18]

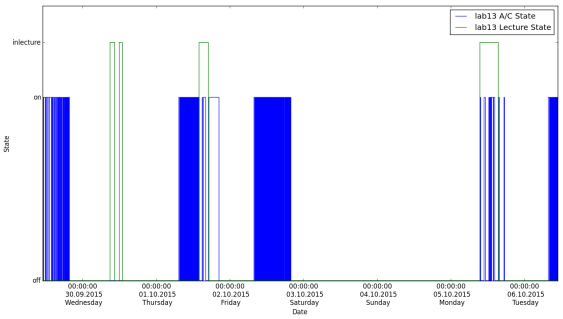

Figure 18: Lab 13 Air Conditioner State - 29/09/2015 – 30/09/2015

Lab 14

Lab 14 has a relatively high occupation rate for classes. Due to this, it is normal that the air conditioner works for longer hours. In the presented example most of the days have at least 4 lectures, except Wednesday (30/09/2015) with only 1 first in the morning. Like in the other cases, the air conditioner is left on very often in between classes even when there is a gap of more than 2 hours; in worst cases like on Wednesday, it is left running all day long. I calculated a total of approximately 28 hours wasted in this week period.

5.2.1 Observations

At first, some of the data was not clear or trivial to interpret and understand. The labs are not the most controlled environment: students enter and leave the rooms, open and close doors and windows, activate and deactivate the air conditioner, etc.
In order to carefully analyze and extrapolate more precise information some stress tests were done.
From these, what I could conclude was that machine uptime had considerably more influence in the room temperature than the quantity of workload these were subjected to. For instance, a lab with a stress test running for 3 hours raised less than 2°C. Yet, on the other hand, two similar labs with 25% of the machines running versus 100%, can easily raise the temperature up 3°C (for a more careful analysis consult the temperature chart stream [18]).

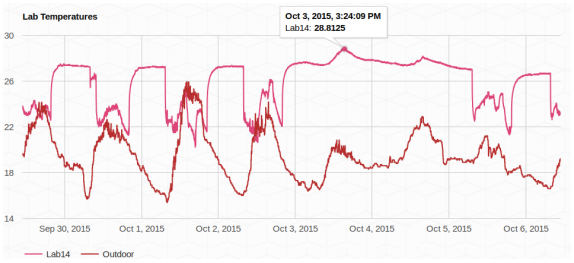

Figure 19: Lab 14 Data from 29/09/2015 to 05/10/2015 [18]

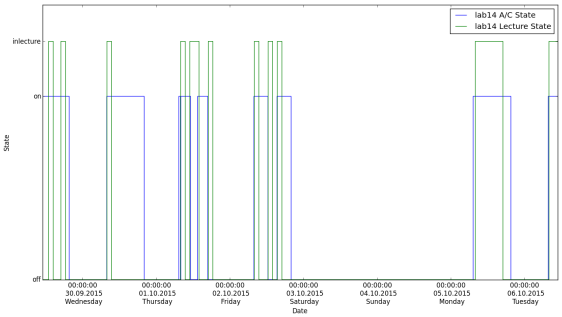

Figure 20: Lab 14 Air Conditioner State - 29/09/2015 – 30/09/2015

Automatically turning the machines off more frequently, when no local or remote work is being done is probably the most effective solution. It will not only help lower the temperature, but at the same time with the support of GreenWoL will force the machines to wake in better rooms when needed. This way, we assure to be maximizing the number of machines working in colder rooms. Furthermore, some policies to control the air conditioning should also be considered:

Policy 1:

The air conditioner remote control should only be lend to a user when the building security observes in the temperature chart stream [18] that the temperature is not adequate. Between 20°C and 24°C it is considered to be comfortable for a workplace condition. This will also avoid the air conditioner being left ON after the lecture.

Policy 2:

The building security is currently informed to turn on the air conditioner in the mornings in some labs in case they got to hot during night workloads; however, even though electronic equipment deals best with low temperatures, there is no need to set it at lower than 23°C just for refrigerating the computers (it is common

to see Lab13 for instance at 18°C).

5.3 Summary

Correctly sizing an air conditioner system is crucial to both understanding what is the correct equipment for the needs and its operational cost. In this matter, the BTU (formally, British Thermal Unit) is the industry standard. This section shows how these calculations are done and applies them to the specifications of the RNL labs providing us with an estimate of the energy costs. Due to the non-deterministic environment (weather alterations, students coming and leaving the rooms and their interaction with the computers and air-conditioners) it is difficult to estimate how many hours of working air conditioning we can reduce, nonetheless it is easy to understand that the purposed system will have a serious effect in this matter. From the collected data, I was able to understand the impact of the workload and machine *uptime* in the room temperature; the last one being somewhat more significant than the other, thus, reducing the machine *uptime* on inactivity will have a relevant impact.

6 Conclusions

Desktop Grid Computing presents itself as a solution for unused computational resources currently present at most organizations. Even though there are already some solutions to its seamless implementation, the connection between the system and real-world environment variables was yet lacking development. I believe this work touched a few of these points while enlightening to some of the possible solutions. The desktop grid can bring great advantages to organizations in terms of computation whereas it does not interfere with the users work, and as long as it does not dramatically increase the energy costs.

The two biggest aspects are the additional machine uptime and air refrigeration. For these, I believe the proposed solutions can make a big difference whilst requiring a very low investment and maintenance. Even though HTCondor had no support for these features, it proved to be as flexible as presented by its developers, allowing me to integrate the temperature sensor network and the lecture schedules to its node ranking calculations. The *ClassAd* mechanism is definitely very adaptable to any imaginable situation, empowering the system administrator to do what its imagination allows.

For future work, I believe that automating the suggested policies (Sections 5.2.1 and 5.2.1) would bring further advantages in reducing energy costs. This would allow the system to be much more deterministic, as the state would be predictable in specific conditions (for a certain room temperature, number of students logged in the room, etc).

References

[1] S. Choi, H. Kim, E. Byun, M. Baik, S. Kim, C. Park, and C. Hwang, "Characterizing and classifying desktop grid," in *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*. IEEE, 2007, pp. 743–748.

[2] R. Bruno, S. Bernardo, and D. Matos, "Experiments on computation and storage scavanging in institutional environments using htc and hpc solutions," 2014.

[3] "Hadoop," http://hadoop.apache.org/, visited 15-10-2015.

[4] "Openmpi," http://www.open-mpi.org/, visited 15-10-2015.

[5] "HTCondor - Overview," http://research.cs.wisc.edu/htcondor/overview/, visited 15-10-2015.

[6] "Htcondor - exceptional features," http://research.cs.wisc.edu/htcondor/manual/v7.8/1_3Exceptional_Features.html, visited 15-10-2015.

[7] E. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in *Power-Aware Computer Systems*. Springer, 2003, pp. 179–197.

[8] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej *et al.*, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Computing*, vol. 16, no. 1, pp. 3–15, 2013.

[9] "CUDA," http://www.nvidia.com/object/cuda_home_new.html, visited 15-10-2015.

[10] D. Kirk, "Nvidia cuda software and gpu parallel computing architecture," in *ISMM*, vol. 7, 2007, pp. 103–104.

[11] N. S. N.V., *I2C-bus specification and user manual*, ser. 6, 2014.

[12] M. Integrated, *Overview of 1-Wire Technology and Its Use*.

[13] M. Integrated, *DS18B20 Programmable Resolution 1-Wire Digital Thermometer*, 2008.

[14] "ASUS Power Supply Wattage Calculator," http://support.asus.com/powersupply.aspx, visited 15-10-2015.

[15] R. American Society of Heating and A.-C. Engineers, *2009 ASHRAE Handbook: Fundamentals*, ser. 2009 Ashrae Handbook - Fundamentals. American Society of Heating, Refrigeration and Air-Conditioning Engineers, 2009.

[16] R. ASHRAE American Society of Heating and I. Air-Conditioning Engineers, *ASHRAE - Mechanical Pocket Guide*, 2009.

[17] "Tarifas Energéticas – EDP," http://www.edpsu.pt/pt/tarifasehorarios/Pages/TarifasBTE.aspx, visited 15-10-2015.

[18] "RNL Temperature chart stream," http://web.ist.utl.pt/ist164766/web.html, visited 15-10-2015.

[19] L. F. Sarmenta, "Volunteer computing," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[20] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: The condor experience," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[21] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya *et al.*, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, no. 2, pp. 47–111, 2011.

[22] "Htcondor -*condor-compile*," http://research.cs.wisc.edu/htcondor/manual/current/condor_compile.html, visited 15-10-2015.