Energy Saving in Future Home Environments

Helmut Hlavacs
and Roman Weidlich
and Thomas Treutner
University of Vienna,
Department of Distributed and Multimedia Systems
Lenaug. 2/8, 1080 Vienna, Austria

Email: {roman.weidlich | helmut.hlavacs }@univie.ac.at, a0306814@unet.univie.ac.at

***Abstract*—Already, hundreds of millions of PCs are found in homes, offering high computing capacity without being adequately utilized. This paper reveals the potential for energy saving in future home environments, which can be achieved by sharing resources, and concentrating 24/7 computation on a small number of PCs. We present three evaluation methods for assessing the expected performance. A newly created prototype is able to interconnect an arbitrary number of homes by using the free P2P library FreePastry. The prototype is able to carry out task virtualization by sending virtual machines (VMs) from one home to another, most VMs being of size around 4 MB. We present measurement results from the prototype. We then describe a general model for download sharing, and compare performance results from an analytical model to results obtained from a discrete event simulator. The simulation results demonstrate that it is possible to reach almost optimal energy efficiency for this scenario.**

I. Introduction

Future home environments are full of equipment offering a multitude of functions making our life more comfortable. Parts of this equipment runs continuously in order to be instantly used when needed, or to run services that require 24/7 operations. Today, especially PCs and notebooks often continuously run and consume energy, but remain underutilized. In our research approach we want to harvest unused cycles in home environments, and automatically decrease the power consumption of the whole distributed system, while still offering the same service level.

II. Related Work

Energy consumption in computers has always been an important issue, that however has gained increasing attention in the last couple of years. Most important, the climate change has stimulated the idea of Green Computing or Green IT, leading manufacturers to continuously develop low power components in order to decrease the global energy consumption caused by computers. In fact, the estimated 1.5 billion computers running worldwide use an estimated total of 90 GW of electrical energy, comprising a share of 10% of the total energy consumption of mankind. Furthermore, this energy consumption already produces an amount of CO_2 comparable to the one of the entire airline industry. In total, end devices in the *home* are contributing to a large portion of the electricity consumption growth in the EU for instance [1].

Another reason for trying to save power consumption caused by computers is the considerable cost, an ordinary off-the-shelf computer running on a 24/7 basis in a European home may well cause energy costs of several hundred Euros per year, although this computer not necessarily computes meaningful tasks, but might be idle most of the time.

The same can be said for professional infrastructure found for instance in offices, or in server farms, which have increased their power consumption significantly over the last years [2]. In the latter, energy costs are not only caused by the consumption of the servers themselves, but also by the need for cooling the servers, and long term electricity consumption costs are steadily approaching the cost of the hardware itself [3].

In the past manufacturers have focused on optimizing the power consumption of single components like CPUs, harddisks, mainboards, etc. Strategies introduced in the past include SpeedStep [4], PowerNow, Cool'n'Quiet, and Demand Based Switching. These measures enable slowing down the clock speeds (Clock Gating), or powering off parts of the chips (Power Gating), if they are idle [5], [6]. All of the above techniques must be regarded as *local* energy saving techniques.

This, however is contrasted by the enormous growth of the IT industry, and the increasing numbers of new PCs and devices worldwide. Of course, the growth of running PCs incurs also a growth of the potential computing power available. Initiatives like Seti@Home[1], Folding@Home[2] and many others try to exploit idling hardware at millions of homes in order to maximize the computing output. These and similar approaches like Grid computing [3] are fully centralized in order to work, depending on central instances for distributing work, security, management etc. On the other hand, there have been no initiatives to share computing resources in a decentralized, self-organized way and on a global scale in order to decrease the total power consumption.

In our work we developed models and schemes in order to demonstrate the feasibility of sharing computing resources of idling home devices to substantially decrease the total energy consumption. Our approach fully relies on self-organization

[1] http://setiathome.berkeley.edu/
[2] http://folding.stanford.edu/
[3] http://www.gridcomputing.com/

and self-management in a decentralized way. The contribution of this paper is the presentation of measurement and simulation results showing this energy saving is indeed possible.

III. Virtual Home Environments

The specifically targeted research project Virtual Home Environments (VHE) [7], [8], funded by the European network of excellence EuroFGI [4], joins researchers from the University of Vienna, Passau and Cantabria in order to understand how possibly millions of home devices can be interconnected in a *decentralized* way, in order to gain common advantage, like for instance lowering the total energy consumption. The basic VHE architecture is shown in Figure 1.

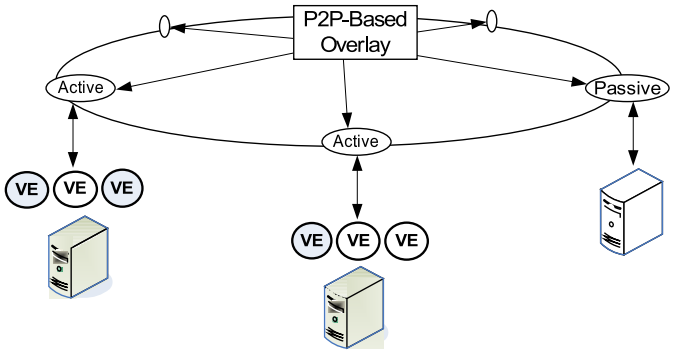

Fig. 1. Basic VHE architecture.

The envisioned architecture interconnects possibly millions of homes by using a P2P system like Chord or Pastry. Each home consists of at least one intelligent router and a PC. The routers must run a special firmware enabling a fixed continuous entry in the global P2P overlay. This way, the churn due to hibernating or woken-up PCs that otherwise might cause high overhead can be decreased dramatically. In our current work, PCs are required to run the Linux operating system and additionally a middleware holding the local VHE intelligence. A more detailed overview over the middleware and its tasks can be found in [7], [8].

In each home users may create tasks that can either be run locally, or may be offloaded to another PC. In this case the system middleware must identify a suitable host in some other home and transfer the task to this home where it is executed. Once the task is finished, the results then must be transferred back the respective *task owner*. Tasks are put into dynamically created virtual machines (VMs). This is done in order to allow the execution of arbitrary tasks, independent of the availability of special software/libraries at the host, and to encapsulate the task into a closed environment that acts as a security shield, protecting the host from possibly malicious tasks.

In an environment like this, which is based on cooperation, besides important issues like security and quality of service, the principle of *indirect reciprocity* (a fundamental prerequisite of cooperation) must be ensured, meaning that participants using resources from other computers must themselves eventually allow the usage of their own resources [9]. For our system we have already developed a fairness model that takes into account different types of resources like CPU, access network bandwidth or disk space [10].

We identified a set of feasible applications with potential for energy saving that usually use computers continuously without need of user input once started.

Download Sharing. In this scenario, home computers sometimes download large files from the Internet, possibly using a file sharing tool or any other means. Alternatively, computers may record live streams from some live source, like Web radio or Web TV. Power saving is done by concentrating downloads on a small number of computers.

Music/Video Encoding. Music/video encoding is a task that requires a very high computational power. Computers from different generations and cost categories exhibit a high variability with respect to their performance. It might make sense to offload an encoding task from a weak computer to a much stronger computer, in order to significantly lower both the encoding time and the power consumption.

Home Management. Home management denotes managing sensors and actuators in a home, like temperature control, light, rain sensors, etc., and requires a computing system running on a 24/7 basis. In this scenario one computer would be used to manage N homes, receiving the data from their sensors, and returning back control messages for changing actuator states.

Avatar Hosting. In the future many virtual worlds are to be expected, and many people will be represented there by a virtual copy of themselves, an avatar. In order to keep their avatar alive, people might want to run client applications on their home computer. Similar to the home management scenario, such clients might be concentrated on a small number of computers.

Server Hosting. This includes hosting private servers like Web or music servers, which would be required to run on a 24/7 basis, but which can be concentrated on a small number of computers.

P2P TV. In this scenario, clients contribute access bandwidth in order to distribute live TV or video-on-demand movies in a P2P fashion.[5]

Disk Sharing. In this scenario home PCs might contribute their disk space in order to provide large data repositories (as is done already in commercial products), or sharing music libraries in a legal and reliable way.

We concentrate on the following resource types that can be shared: (i) CPU, (ii) access network, (iii) disk storage, and (iv) sensors. The above applications do not use all three types of resources identically. To consolidate the resources of an underutilized computer, many tasks must be aggregated. Thus, under the assumption there exists a suitable software connecting many computers in a way they can exchange tasks, then load can be outsourced. While the load is outsourced, the idle computer could hibernate to save energy.

[4]http://www.eurofgi.org

[5]http://www.joost.com/ or http://zattoo.com/de

IV. A Prototypical Implementation

The idea of *task virtualization* is to migrate a VM containing only a specific task. Currently, task virtualization is only possible within expert environments, since due to the complexity of decentralized systems, normal users are not able to create tasks for remote execution. Solving this usability problem for home users would enable virtually all user groups to participate in effective resource sharing and is an important part of our future work.

We have developed a first prototype called "vPastry" that allows easy to use task virtualization for home environments. The prototype connects to other instances by using the open source library FreePastry. On the first start the user is prompted to provide some basic information about the system he uses. A minimalistic resource/performance model is used, comprised of information about

- CPU: Number and performance of CPU cores.
- RAM: Size and speed of main memory.
- Disk: Disk space and speed.
- Energy efficiency of the system as a whole (e.g., power supply).
- Connection: Bandwidth of connection to the overlay.

CPU, RAM and disk metrics can be understood as how much a user is willing or able to contribute. This is important as it enables other participants of the overlay to find a host with desired abilities for a specific task. For finding appropriate hosts for a task a user would select the desired characteristics and click on the search button. vPastry then uses the "anycast" message of "Scribe", a publish/subscribe messaging system offered by FreePastry.

When a node receives an invitation for a task it requested, the user is prompted to select a virtual machine to send while a file transfer channel is established in the background. We currently develop VMs following the *just enough operating system* (Jeos) approach, here trying to reduce the size of the VM to a minimum. Currently available task VMs include VMs for performance tests, converting an MP3 file to OGG Vorbis, a "Personal Stream Recorder" (PSR) capturing N seconds of a predefined radio webstream and saving it as MP3, stress tests for stability analysis, downloading a specified file using BitTorrent, and downloading a specified file using the command *wget*. Though all VMs contain a fully bootable Linux machine, sizes of compressed versions vary between only 3.8 and 4.7 MB. The only exception is given by the VM containing an MP3 file, which amounts to 43 MB.

Task VMs consist of a minimal Linux kernel image, a minimal root filesystem built by OpenEmbedded and a task filesystem layer, which is put on top of the read-only root filesystem by UnionFS. The VM's task filesystem shelters a shell script which is executed after the VM has started and diskspace for installing required packages (e.g., `lame`) and writing resulting data (e.g., an MP3 file). When the executor receives a VM, it is decompressed into a temporary place and started by a runtime execution using the Kernel-based Virtual Machine (KVM) driver and QEMU (Figure 2).

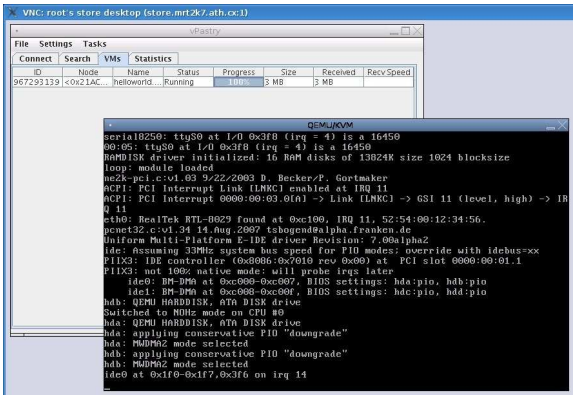

Fig. 2. vPastry: Executors view. A VM is received, decompressed and started with KVM and QEMU.

As KVM's primary user interface is the command line, it is easy to start VMs with KVM from within other programs as for example vPastry. The executed command is currently from the following scheme and can be adapted to future work in terms of task-specific memory reservation or use of different network models provided by KVM:

```
/usr/bin/kvm -kernel bzImage -hda rootfs.ext2
    -hdb task.ext2 -m 32 -append "root=/dev/hda"
    -net nic,vlan=0,macaddr=XX:XX:XX:XX:XX:XX
    -net tap,vlan=0
```

Once a task is finished (which basically means that the respective VM has exited), the VM task filesystem is compressed and can be sent back to the task owner. Only the task layer is sent back, as the root filesystem is mounted read-only and has encountered no changes (and never will).

V. VHE Performance Evaluation

In this section it is investigated how energy can be saved by sharing home resources, here using download sharing (DS) as an example. In this scenario N homes are interconnected with gateways, each home offering one PC. For simplicity we assume a homogeneous environment. A home's downlink bandwidth B_d and uplink bandwidth B_u to outside is shared between all local computers. For future homes we assume a synchronous access with either $B_d = B_u = 50$ Mbit/s, or $B_d = 8$ Mbit/s downlink and $B_u = 4$ Mbit/s uplink bandwidth. The local bandwidth within the home is 100 Mbit/s. A download task uses on average $B_l = 2500$ Kbit/s of the available access bandwidth and has on average a size of $F = 700$ MB. It follows that a home can carry out $M = B_d/B_l$ downloads in parallel. Following a Poisson arrival process, home users create DS-tasks with arrival rate λ, the service rate is given by $\mu = B_l/F$.

A. Prototype Measurements

In order to validate the resource consumption of executing VMs we carried out the following experiments for the download sharing scenario. A PC (AMD Phenom 9550 Quad-Core Processor (2.20 GHz) with 8 GB main memory) was

setup to act as a host. A client instance of vPastry then sent $1 \leq N \leq 5$ download tasks (VMs) to the host, which decompressed and ran the VMs. The tasks then downloaded a file via `wget` at 300 KByte/s from an FTP server connected via GigabitEthernet. The total upstream bandwidth of the FTP server was limited to 8 Mbit/s to simulate a typical ADSL access network with 8 Mbit/s downstream bandwidth, the upstream bandwidth of a single FTP connection was limited to 300 KByte/s to simulate a download that does not fully utilize the available total downstream bandwidth. The intention was to assess the resource demands of such a scenario. Figure 3 (fat lines) shows the results, consisting of the long term resource utilization of the CPU, the main memory and the network card of the host PC. As a result,

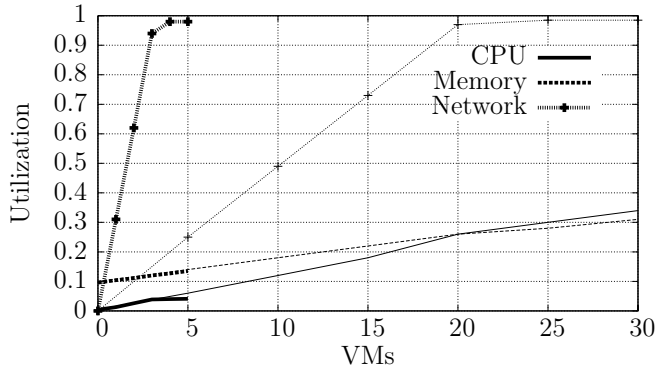

Fig. 3. Resource utilization (CPU, memory, network) of a download scenario. Fat lines denote results for a 8 Mbit/s access network (i.e., saturation occurs for $\lceil 8/2.4 \rceil = 4$ parallel downloads), thin lines for a 50 Mbit/s access network.

resource utilization clearly depends linearly on the number of VMs being hosted. Memory and CPU utilization grow slowly, due to the little resource demands of our minimalistic Linux operating system. This indicates that when hosting download sharing on a modern PC, the system bottleneck indeed is the network, but CPU and memory can be used to host other types of tasks. When the network is beginning to be the bottleneck of the system, an interesting behavior can be observed: With three VMs downloading a file at 300 KByte/s (2.4 Mbit/s) each, the resulting downstream utilization is clearly within the available downstream bandwidth of 8 Mbit/s and the network is not yet a bottleneck. Beginning with four VMs, the network becomes a bottleneck, as the desirable downstream utilization would be 9.6 Mbit/s, and for five VMs, 12 Mbit/s. As the downstream bandwidth is limited to 8 MBit/s, the incoming amount of data and therefore, the amount of data that has to be written to disk, is limited too. In our experiments we observed that the CPU utilization rises more slowly when the system bottleneck is fully utilized. At this point, there is no sense in adding more VMs, as the resource utilization rises with no benefit of work being done.

In a second experiment, the FTP server's total upstream bandwidth was limited to 50 Mbit/s to simulate a future Internet access like FTTH (see thin lines in Figure 3). A client instance of vPastry sent $5 \leq N \leq 30$ VMs containing a download task (`wget`, 300 KByte/s). The intention was to verify the linear resource demand of a download sharing scenario also for *future* Internet access bandwidths.

As a result, our approach scales well to a greater number of VMs being hosted. Even then, the system bottleneck is the network, with enough CPU and memory resources being available for hosting other types of tasks. Furthermore, a large number of VMs, possibly >100, can be hosted with modern PCs, which is important for other scenarios like home management, where only little CPU and network resources might be needed. When the network begins to be the bottleneck of the system, which happens in this case between 20 and 25 VMs, the above mentioned behavior is to be observed here too.

B. Simulation Model

In [7] we developed an analytical model that denotes the average number of active computers for the download scenario, which is (assuming the parameters given above) given by N_l in the local case:

$$N_l = N \left[1 - \left(1 + \sum_{k=1}^{M} \frac{1}{k!} \left(\frac{\lambda}{\mu} \right)^k \right)^{-1} \right], \qquad (1)$$

and by N_d in the distributed case:

$$N_d = \sum_{a=1}^{N} a \sum_{k=(a-1)M+1}^{a\,M} \pi_k + \frac{B_l}{B_u} \sum_{k=1}^{MN} k\,\pi_k \, \frac{N - \lceil k/M \rceil}{N}, \qquad (2)$$

with

$$\pi_k = \pi_0 \frac{1}{k!} \left(\frac{N\lambda}{\mu} \right)^k, 1 \leq k \leq N\,M,$$

and

$$\pi_0 = \left[1 + \sum_{k=1}^{N\,M} \frac{1}{k!} \left(\frac{N\lambda}{\mu} \right)^k \right]^{-1}.$$

Equations (1) and (2) model a somewhat ideal scenario since they do not capture the efforts for maintaining a P2P network. In order to include also middleware overhead, we currently develop a discrete event simulator for VHE. The simulator models home resources, home users, and the network infrastructure in between. Home computers may be in any of the following states: (i) passive, (ii) active blocked, (iii) active, and (iv) active blocked content. A *passive* (P) computer is assumed to be in hibernating mode, thus consuming no energy. In this case, the computer is quasi switched off and does not contribute its resources to the VHE system. A user deciding to create a task may wake up the computer, thus setting it into state *active blocked* (AB). Once a task is created the VHE system looks for an *active* (A) computer that is ready to take over the task. A computer is called active if it is running and executing tasks for its own home user, or any other remote user. If no remote active computer is found, the local computer itself goes into active state and computes the task, but also from this time on accepts tasks from other homes. After the executer has finished a task for some remote user he sends

back the result to the owner. If the owner at this time is in state P, the owner must switch to state *active blocked content* (ABC) for the time the upload of the content from the executer to the owner takes. After a successful transfer of the content to the owner the task is finally completed. We can see the home's state cycle in Figure 4. To introduce fairness and to ensure that

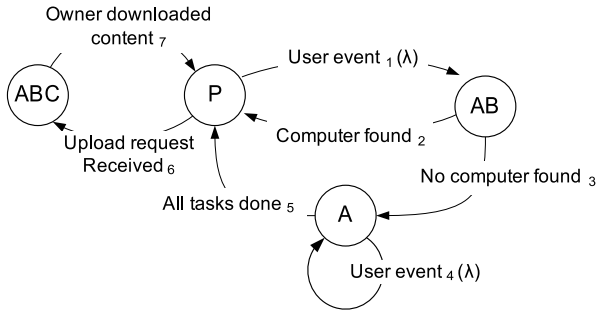

Fig. 4. State cycles.

the load is equally distributed over all homes in state A, we introduce a fairness parameter called *service time* (ST). The ST is the time period the home must accept remote tasks, beginning at the time of state change into state A. After the ST has gone by, incoming task requests are refused. The ST thus guarantees that each contributer may eventually change from state A into state P.

C. Simulation Results

In order to assess the download sharing scenario in more detail we ran several simulation experiments. There the ST was set to 8 hours, and the simulation time to one year.

In [8] we compared the analytical model (1) and (2) to a first simulation scenario which already showed good accuracy, but due to its implementation choices very much depended on the fairness parameter ST. We improved this simulation model in the following by making it possible to delay result transfers back to the task owners. The results for this new version are shown in Figure 5 for N=100 and 200 homes. The x-axis shows the maximal number of new tasks per week (load) and the y-axis the number of active computers necessary to carry out all generated tasks (cope with the whole load). Thick lines denote simulation results, while thin lines denote the analytical model (1) and (2). Furthermore, for each application the local and distributed case is compared. In the *local* case no task will be outsourced, thus all homes have to carry out the tasks themselves. In the *distributed* case resources are shared by task outsourcing.

It can be clearly seen that for the local case far more computers must be up and running compared to the distributed case. Furthermore, analytical and simulation results are quite close to each other, meaning that the overhead of our distributed algorithm keeping up the P2P connection is almost minimal.

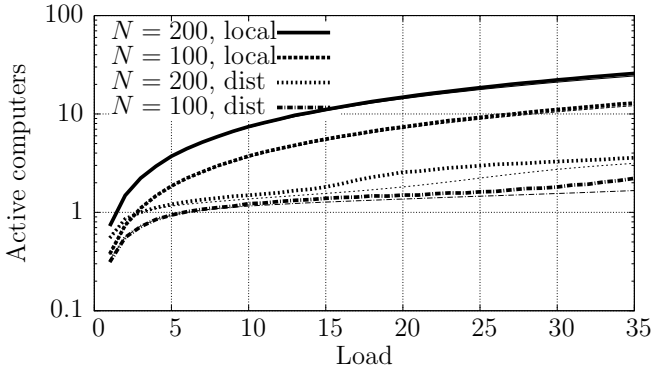

Fig. 5. Average number of active computers only for the download sharing (DS) scenario ($N = 100$ and 200 homes). Thick lines denote simulation results, thin lines denote the analytical model.

VI. Conclusion

In this paper we show how energy consumption of home services can be reduced if homes share their resources amongst each other. We presented the first prototype for sharing resources called vPastry, being based on Pastry and KVM. Using the prototype we have investigated the resource utilization of the download sharing scenario, showing that CPU and Memory are only lightly utilized. For the same scenario we present simulation results for an advanced communication model, which shows near optimal efficiency when compared to an optimal analytical model.

Our future work will focus on implementing more and more application types inside VMs, and the general usability of the prototype, as well as simulation models for further scenarios.

References

[1] P. Bertoldi and B. Atanasiu, "Electricity Consumption and Efficiency Trends in the Enlarged European Union," Institute for Environment and Sustainability, European Commission Report EUR 22753 EN, Tech. Rep., 2007.

[2] J. Koomey, "Estimating Total Power Consumption by Servers in the US and the World," Lawrence Berkeley National Laboratory and Stanford University, Tech. Rep., 2007.

[3] IBM Virtualizatin View, "Virtualization Can Help Power Efficiency," http://www-03.ibm.com/systems/virtualization/view/011607.html, January 2007.

[4] Intel white paper 30057701, "Wireless Intel SpeedStep Power Manager: Optimizing Power Consumption for the Intel PXA27x Processor Family," 2004.

[5] Intel, "Energy Star* System Implementation," www.intel.com/cd/channel/reseller/asmo-na/eng/339085.htm, 2007.

[6] C. Windeck, "Energy Star 4.0," *C't German Magazine for Computer Techniques*, vol. Vol 14, pp. Pages 52–53, 2007.

[7] H. Hlavacs, K. Hummel, R. Weidlich, A. Houyou, A. Berl, and H. de Meer, "Energy efficiency in future home environments: A distributed approach," in *IFIP TC6's and IEEE's 1 st Home Networking Conference, Paris, France*, December 2007.

[8] H. Hlavacs, K. Hummel, R. Weidlich, A. Houyou, and H. de Meer, "Distributed energy efficiency in future home environments," *Annals of Telecommunications*, 2008, special Issue on Home Networks.

[9] M. Nowak and K. Sigmund, "The dynamics of indirect reciprocity," *Journal of Theoretical Biology*, vol. 194, no. 4, pp. 561–574, 1998.

[10] A. Garcia, A. Berl, K. Hummel, R. Weidlich, K. Hackbarth, H. de Meer, and H. Hlavacs, "An economical cost model for fair resource sharing in virtual home environments," in *Proceedings of the EuroNGI conference 2008*, 2008.