System Architecture Support for Green Enterprise Computing

Maria Kazandjieva
Stanford University
mariakaz@cs.stanford.edu

Chinmayee Shah
Stanford University
chinmayee.shah@stanford.edu

Ewen Cheslack-Postava
Stanford University
ewencp@cs.stanford.edu

Behram Mistree
Stanford University
bmistree@stanford.edu

Philip Levis
Stanford University
pal@cs.stanford.edu

Abstract—This paper proposes a novel system for enterprise computing that reduces energy consumption without sacrificing performance or putting devices to sleep. It uses a hybrid architecture composed of multiple device classes and it runs an application in the most energy-efficient location. Our prototype, called Anyware, provides desktop-class performance while reducing energy consumption by 75% through a combination of lightweight clients and a small number of servers. Providing such an elastic system invisibly to end users requires solving many challenges, including deciding where to run applications while simultaneously making it appear they all run locally. Anyware's hybrid design suggests a new way to think about using the modern spectrum of personal computing devices.

Keywords— energy efficiency, system architecture, hybrid computing

I. Introduction

This paper argues for taking an *architectural approach* to green computing in the enterprise (office) environment. Rather than focus on one component (e.g., CPU) or device (e.g., server) within a computing system, redesigning the system as a whole can have greater benefits with lower costs. Considering the different power/performance curves of personal computers, servers, and networking infrastructure, we propose Anyware: an elastic, hybrid computing architecture that reduces desktop computing energy consumption by over 75% without sacrificing performance and, in some cases, even improving it.

Anyware leverages the fact that personal computers today have a non-linear power/performance tradeoff. By running most undemanding applications on low power, efficient personal computers such as laptops, one can reduce each person's personal computing energy consumption by over 80%. But this benefit comes by sacrificing performance and therefore productivity. For some demanding tasks, a lightweight personal computer such as an Eee PC can take twice as long as a reasonable desktop. To provide both low energy consumption and high performance, Anyware uses a hybrid architecture. Some applications run locally on users' low power computers and others run remotely on powerful shared servers. The architecture is elastic since more servers can be added to scale to users' workload needs.

Anyware's goal is to reduce energy consumption without harming productivity. It is therefore completely invisible to its users. This is in contrast with existing approaches such as LiteGreen [1] and SleepServer [2], which put computers to sleep that users have to wake them up when needed. Applications running on top of Anyware look and run identically (configuration, preferences, plugins, etc.) whether they are run locally or remotely. All of a user's files are always accessible. Anyware requires no centralized application authority: a user can install a new application on her file system and use it with Anyware immediately. This invisibility comes without any modifications to user applications or operating system kernels.

Although Anyware is a synthesis of many existing system design concepts targeted at these problems, its contributions lie in answering six research questions:

1) In modern enterprise environments, what are the highly underutilized resources one can cut to reduce idle energy or use more of to increase efficiency?
2) How can one run a locally installed application on a remote server without modifications or user intervention, or even a user noticing?
3) How does a client find remote computing resources for offloading applications?
4) When should a client offload applications and when should it run them locally?
5) What does a computing system designed to take advantage of Anyware look like?
6) How does Anyware affect application performance and energy consumption?

Anyware trades off local computing power for a small number of powerful servers and increased network utilization. The former is amortized across many clients, while the latter essentially comes for free as network power is independent of its utilization [7]. Anyware applications run remotely by having very thin virtual machines hosted on remote servers that use the client's local file system. Clients can automatically find Anyware offload resources using DNS with service discovery, such that all a user has to do is connect to a network and Anyware auto-configures and starts running applications remotely. We derive an offload algorithm based on a logistic regression of

978–1–4799–6177–1/14/$31.00

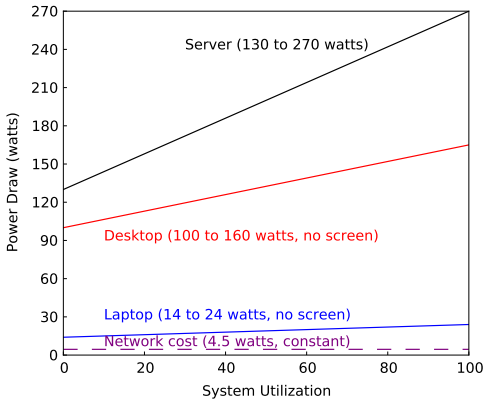

Figure 1. Relationship between power draw and utilization for various pieces of office equipment. Note that the network cost is low and constant.

multiple application properties and a user study of perceived performance. Anyware's design reveals the importance of efficient but fast I/O, suggesting a future personal computing device centered around solid-state drives, energy efficient processors, and GPUs for local applications. Anyware's approach of distributed execution and centralized storage, potentially backed up to the cloud, is a novel point in end-user computing system architectures, motivated by the changing tradeoffs between energy, network, and workloads we see in enterprises today.

II. Enterprises Today

Improving the energy-efficiency of enterprise computing requires first understanding it. This motivates our first question:

> *In modern enterprise environments, what are the highly underutilized resources one can cut to reduce idle energy or use more of to increase efficiency?*

Systems have poor power proportionality. Figure 1 shows how power draw increases with system utilization for representative examples of four different computing elements within the enterprise: a Mac laptop, a Dell desktop, an Intel Xeon server, and a single active port on an HP network switch. All power data in this paper are empirical measurements of devices within the Stanford Computer Science building, gathered using custom wireless power meters. All devices have poor power proportionality [3]: their idle power draw is 48% to 100% of their draw at full utilization. One important observation is that the switch's power draw is completely independent of its utilization. Therefore, a computing system that increases network traffic can still reduce the aggregate energy consumption.

Small increases in performance can have a high power cost. Power and performance have a non-linear relationship. Beyond a certain point, small increases in performance cost a lot of energy. Figure 2 illustrates these diminishing returns

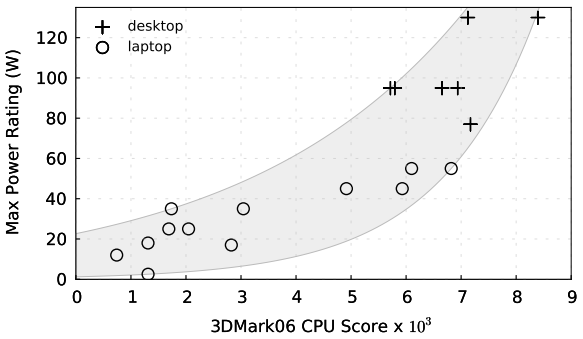

Figure 2. Processor performance as rated the 3DMark benchmark versus maximum power dissipation. For modern processors, power draw increases non-linearly with performance.

for several modern mobile and desktop processors [4]. The performance is rated using the 3DMark benchmarking tool [5]. The power ratings reflect the maximum processor power dissipation, an upper bound for active power draw.

A typical mobile CPU draws about of 15-20 watts during operation, while a desktop processor reaches 100 watts or more — four to five times as much [6], [7]. While the difference in power draw is significant, this four-fold increase in energy cost does not come with a four-fold increase in performance. We could build more energy-efficient systems by using multiple points from the design space.

User systems have low average utilization but occasionally need high performance. Prior work [7], [8], [9] has consistently shown that PCs are often underutilized. It is not uncommon for a machine's CPU to be under 25% utilization for more than 75% of the time. Office workloads only occasionally tax a machine's resources, but systems are provisioned for those uncommon, bursty events. The non-linear relationship between power and performance means this over-provisioning has a significant cost: hundreds of machines consume energy for the rare cases when one of them needs high performance.

Networks are significantly overprovisioned.
Enterprise networks are also highly underutilized [7]. Cumulative traffic is often only several percent of capacity, with the peak at about 20% or less. This unused capacity within the enterprise LAN can be put to use without increasing the energy cost of the network.

Current power saving techniques trade off productivity for efficiency.
Thin clients are one common approach to reducing energy consumption. They consist of an end user system which is typically no more than a display and a lightweight processor for graphics draw calls. They provide no local compute or storage resources. Instead, a small number of powerful back-end servers run all user programs and sessions, assuming that multiplexing the server resource across users improves average utilization without reducing peak performance. Thin

clients have two major limitations. First, the lack of local compute resources means they cannot handle graphics-heavy tasks. Second, thin clients draw as much power as lightweight computers such as laptops (15–20W) [10].

Recent research on enterprise energy efficiency has concentrated on putting idle desktops to sleep by using sleep proxies and migrating user sessions [1], [2], [8], [11]. A common limitation to these approaches is that they require users to wait for their compute environment to become available. This fundamentally trades off convenience and productivity for improved energy efficiency. Furthermore, putting personal computers to sleep is not always possible. For example, in our organization, computers need to be awake at night for nightly backups, and as prior work has pointed out [12], network wakeup mechanisms are fragile and difficult to rely on in practice.

III. Running Applications Remotely

The very low average PC utilization in the enterprise means that most work can be handled by lower power, lightweight computers such as Eee PCs, laptops, and Mac Minis. For example, if laptops have an average power draw of 20 watts while desktops have an average draw of 100 watts, simply replacing all desktops in the enterprise with laptops will reduce idle power by 80%, more than any existing sleep solution. Furthermore, the power/performance curve of these lower power devices means that they can provide a reasonable fraction of desktop performance at a fraction of the cost.

But some occasional workloads need the performance of desktop, and running them on a laptop can be significantly slower. To conserve energy without harming productivity, an enterprise computing system needs multiple classes of computing devices that appear as a single enterprise system. This leads to our second question:

> *How can one run a locally installed application on a remote server without modifications, user intervention, or even a user noticing?*

To answer this question we present Anyware– a hybrid, elastically-provisioned system architecture for enterprise computing provides . It is hybrid because a single user can be executing some tasks on a *local low-power client*, while other workloads are offloaded to a more powerful *remote server*. It is elastically-provisioned because in the spirit of cloud computing, resources can grow as needed by adding more backend servers. Anyware requires neither OS nor application changes; it exports directories via NFS and reassigns MIME types via a configuration file.

An Anyware client machine has a processor with a very high performance-per-joule, such as a laptop or a low-end PC. Anyware shares a small number of servers across many users, amortizing their cost similarly to thin client systems. But unlike them, an Anyware client is a fully operational computer that can function disconnected from the server, if necessary. Unlike sleep approaches, the user's work environment can always remain on, instituting no wakeup latency or penalty.

Our approach requires only one change to the OS configuration and allows Anyware to run entirely in user space. The approach trivially extends to other Unix-like systems and can be engineered for other OSes.

An Anyware client maintains a full operating system, with all user files, configuration data, libraries, and applications. A small number of high-performance servers on the enterprise cloud host virtual machines. A given client can have at most one VM per server but may have VMs on multiple servers. Each server VM contains a minimal OS installation, configured only to be able to bootstrap execution of the user's software. The Anyware client is an NFS server, exporting data and program directories. Anyware server VMs mount the necessary directories via NFS with the sync option. All other communication between the Anyware client and VM are over SSH. X11 forwarding allows remote-execute applications to appear as if running on the client machine. On the server side, certain core system configurations also need to be unique. For instance, the client and the VM do not share the same `fstab` file or log files. They also do not share network-related configurations.

Anyware flexibly places applications by intercepting all user-initiated program executions through a level of indirection to operating system MIME type associations. It introduces two shadow files, `mimeinfo.cache` and `mimeapps.list`, that cause all file extensions and application shortcuts to be associated with the Anyware executable. Once intercepted, the Anyware daemon decides where to run the application and invokes the original program on that host.

IV. Finding Remote Resources

A careful composition of user-level settings such as MIME type bindings and exported filesystems allow Anyware to execute workloads remotely in a way that is invisible to the user. This goal of creating a minimally invasive computing system, raises an additional challenge:

> *How does a client find available remote computing resources for offloading applications?*

Anyware adopts a zero-configuration (Zeroconf) methodology that takes advantage of DNS-based Service Discovery (DNS-SD) [13]. DNS-SD uses standard DNS queries to allow network clients to look up a service within a specific name domain. For example, Bonjour is an implementation of DNS-SD for Mac OS, enabling services such as iTunes sharing on the same network. This technology is well suited for Anyware which operates on enterprise LANs.

The current implementation of Anyware uses the Avahi Zeroconf implementation for Linux and BSD [14]. Remote Anyware servers use it to advertise themselves by placing a configuration file in the `/etc/avahi/services` folder. A new Anyware client can bootstrap the entire process

via one of these advertisements; it requires a thin layer of software to execute an Avahi query and parses the returned advertisement, which has the server's IP address and port. The client then uses them to establish a socket connection and download the rest of the setup scripts from the server.

First, the client takes several local configuration steps. It creates a public/private key pair (currently, without a passphrase) and modifies `/etc/exports` to export its directories. Next, it sends information about its operating system and architecture so that the server can instantiate a new virtual machine. During this step the server can pass additional information to the VM, including the client's username and IP address. After the VM is up and running the client's NFS directories are mounted. In addition to giving the VM access to applications and user data, the mount also implicitly sets up the SSH connection: both the private and public keys live in the same folder and are accessed by the server and client respectively. From the client's perspective this is equivalent to a `ssh localhost`. At this point, the VM signals that the setup is complete.

Taking a zero configuration approach to Anyware is key to being invisible to the user. A client only needs to execute a simple program to look for advertisements in order to start using Anyware; everything past the first step is an automated exchange of network messages, letting users focus on more productive tasks. From a system's perspective, Zeroconf allows flexibility on both the remote and local sides of Anyware. Additional server resources can be added on demand, by initiating new advertisements. Clients can automatically find who to offload applications to and have the ability to choose one or more servers to connect to.

V. Execution Placement

Sections III and IV described the mechanisms by which an Anyware client can invisibly run applications remotely. However, we cannot expect enterprise users to be experts in the needs or performance of applications and force them to decide where to run applications. Instead, the system needs to decide so automatically, leading to the question:

> *When should a client offload applications and when should it run them locally?*

The centralization worldview, argued by the administrators of a lab filled with thin clients, is that remote execution is almost always better because it can use high-end server hardware. The local execution worldview, argued by our local system administrators, posits that the added latency and bottleneck of the network mean local execution will be generally better. Where does the balance lie?

We perform a small-scale user study in which subjects complete a wide range of computer tasks using either a laptop or remote Anyware execution. The results from the study (Section V-B) together with a set of application features allow us to develop a statistical model. This model, described in Sections V-C and validated in Section V-D, shows how a few application properties can be used to predict user preferences for remote and local execution.

Table I
Application features collected in order to build a predictive model for Anyware.

Identifier	Explanation	Collection mechanism
`instr`	Instructions executed ($\times 10^9$)	`perf`
`ipc`	Instructions per cycle	`perf`
`llc`	% last-level cache misses	`perf`
`xmsg`	Number of X messages	`xtrace`
`procs`	Number of processes spawned	`strace`
`opencalls`	Number of `open` system calls	`strace`
`mbread`	Data read (MB)	`strace`
`mbin`	Data sent over the network, from server to laptop (MB)	`iptables`
`mbout`	Data sent over the network, from laptop to server (MB)	`iptables`

Table II
Execution placement classes determined from user experiments.

Class	Description
Local only	Remote execution is unusable, local is usable
Local preferred	Both executions are usable, local is better
Either	Both remote and local were acceptable, no difference
Remote preferred	Both executions are usable, remote is better
Remote only	Local execution is unusable, remote is usable
Fail	Neither local nor remote is usable

A. Methodology

Five subject participated in the study by performing 39 unique tasks in two conditions. In the *laptop condition*, task were executed locally. In the *Anyware condition*, tasks were offloaded to a server using Anyware. The jobs span a variety of applications that capture the workloads of an enterprise environments. These include office programs (e.g., create a slide presentation, edit photos, read and fill out PDFs, compress files), social communication (e.g., send an instant message or an email), and games. After performing a task in either the laptop or Anyware condition, the subject answered whether the application performance was acceptable (yes or no). At the end of each workload, the subject answered whether the first or second execution was better (first, second, or no difference).

Lastly, we profiled each workload separately to collect application features. Table I summarizes the features and how they were measured. Based on the user responses, we classified applications into six classes, shown in Table II.

B. Experimental Results

While there was some disagreement between test subjects on whether there was no difference between conditions or one condition was better, the results were never inconsistent. In no case did one subject mark a task as better with Anyware's remote execution and another subject mark it as better running locally. Most applications were rated as usable in both the local and remote scenario. A majority of

users classified PDF viewing and photo organizing as *local preferred*. Users preferred remote execution for many of the image processing tasks, as well as the Open Office document manipulation tasks. One user placed Thunderbird in the *fail* class due to its slow response to user interactions.

Only two applications were classified as *local only* by two or more people – video playback and Google Earth. These tasks highlight the failure case of systems in which all computation is remote (thin clients or remote VMs). Both applications update screen graphics at a high rate, making it difficult to run remotely. The model derived in the next delves deeper into this issue and show an evaluation of one of these *local only* tasks in Section VII-C.

C. Logistic Regression Model

We use the features in Table I and user classifications to create a model of where Anyware should run applications. The model is built using logistic regression, a type of regression analysis that is well-suited for predicting a boolean value [15] from predictor variables. To build a model, one specifies a data set of numeric prediction variables and their corresponding boolean results. Given a set of prediction variables, a model produces a number between 0 and 1, indicating the predicted probability that the result is true.

In the case of Anyware, the predictor variables are application features and the boolean value is whether to run an application locally. For model training, *local only* and *local preferred* are local execution (value is 1), while *either*, *remote only*, and *remote preferred* are remote execution (value is 0). The *either* case defaults to remote in order to enable thinner, lower power clients. Anyware runs an application locally if the output of the model is ≥ 0.5.

To generate a more compact model, reduce the number of features Anyware must collect, and gain insight into which features are most important, we simplify the model using single-term deletions. Single-term deletion removes individual features or combination of features whose removal does not lead to a statistically significant difference (at $p - value < 0.05$) in the model's predictive power.

D. Model Training and Validation

In order to train and test a user-independent model that determines an application's execution environment, we need to collapse the five user's ratings into one. Section V-B discussed how different user responses diverged. For example, four users place a task in the *either* category, while the last user places it in the *local preferred*. In these situations, we assign the task to the category that majority of users chose.

To evaluate the effectiveness of logistic regression based on these features and classes, we randomly select 29 of the tasks to be in the training set and test the resulting model on the remaining 10. We generate and test 20 such models, each with a different division of the 39 workloads. This evaluates whether training on a reasonable set of tasks can generate a model that is broadly applicable to many more tasks. All resulting reduced-feature models use the following same variables:

$-k \times$ instructions
$+l \times$ MB send
$+m \times$ MB received

The constants (k, l, m) depend on the training data set and the units for each feature, e.g. megabytes versus kilobytes. What is important is the *sign* of each contributing variable. A positive coefficient increases the output of the function, such that a higher value for that feature will push the application to run locally. Conversely, a negative coefficient means a higher value will push the application to run remotely.

This highly simplified model captures the inherent tradeoff between processing and I/O capabilities. The number of instructions an application executes is a proxy for how CPU intensive the task is; higher instruction counts bias towards remote execution that can use the server's more powerful CPU. Conversely, tasks that perform substantial I/O between server and client (MB_sent and MB_received) can become network latency bound in comparison to local systems.

The average prediction accuracy is 88%, the minimum is 60% (in one of the 20 models), and a maximum of 100% accuracy (in five models). In only one case, there is an application assigned to an unusable environment – Google Earth which is classified as *local only* is predicted to be offloaded to the server. The remaining errors are tasks in the *local preferred* class, which the model predicted as remote. These include firefox, thunderbird, unzip, and full-screen text editing – all tasks with a high number of instructions executed. On the flip side, the FreeCiv game which users did not have a preference for, was predicted as local by the model, due to its higher network traffic.

VI. Architectural Support for Anyware

The Anyware architecture trades off an increase in I/O for the ability to choose where to run an application. Given that the client serves user data, a question emerges:

> *What does a client system designed to take advantage of Anyware look like?*

Laptops and low-end PCs often have low performance storage systems, such as low-speed (5400RPM) disks. This hardware decision becomes a performance bottleneck for the I/O-centric nature of Anyware clients. Table III shows the time in seconds it takes on average (over ten runs) to execute three sample workloads with warm caches. Section VII-A describes the experimental methodology in detail; in this context, the important point is that Anyware client performance lags behind a desktop, in some cases drastically.

The Gnumeric workload, which is a mixture of CPU and I/O, has a 31% increase in completion time on the laptop, while the Eee PC gives an 76% increase, compared to the desktop. This verifies that simply switching out equipment

Table III
EVEN WITH ANYWARE, REPLACING A DESKTOP WITH LOWER-END MACHINE HAS A LARGE NEGATIVE EFFECT ON SYSTEM PERFORMANCE.

Runtime in seconds	Gnumeric	GIMP	Kate
Desktop	6.93	42.19	2.42
Laptop only	8.72 (+31.5%)	51.06 (+21.0%)	2.72 (+12.4%)
Anyware on Laptop	8.21 (+18.5%)	37.11 (-12.0%)	4.75 (+96.3%)
Eee PC only	12.25 (+76.8%)	155.39 (+268%)	4.33 (+78.9%)
Anyware on Eee PC	9.48 (+36.8%)	44.13 (+4.6%)	4.48 (+85.1%)

is not an acceptable path to energy savings. Anyware makes up for some of the slowdown by taking advantage of the faster CPU, but there is still an 18% and 37% increase in completion time relative to the base case.

The advantage of the server-side CPU is more visible in the CPU-bound GIMP workload. In the extreme case, The Eee PC execution sees a 200% increase in completion time. The combination of a single-core slow CPU, a slow disk, and not enough memory is devastating for the task. Anyware's remote run remedies this. When used on the laptop, Anyware performs better than the power-hungry desktop; on the Eee PC, it is close to the desktop.

The last workload has higher I/O demand that the previous two since it requires the reading and writing of a 3.2MB file. The slower hard disks affect execution time by 12% and 80% for the two clients, and adding Anyware makes things even worse. NFS served from a slow drive become a serious bottleneck for Anyware, resulting in almost 100% increase in execution time compared to the desktop. Since Anyware uses NFS mounts with the sync option, every write has to hit the laptop disk, before the application can proceed. The Kate workload causes NFS to send 72 'WRITE' remote procedure calls over the network, compared to 12 and 6 for Gnumeric and GIMP, respectively.

On one hand, the data in Section V was collected on the laptop and users rated application performance as satisfactory. Yet, the data in Table III illustrate that the slowdown is not insignificant.

Even though a workload can benefit from the fast CPU at the server, it turns out that with careless hardware configurations the local hard-drive become bottlenecks. Given that many user workloads are not CPU-bound but rather I/O-bound, Anyware's reliance on client storage becomes a liability. Therefore, Anyware clients, follow the trend of storage today and replace their hard disk drives with solid state drives. In our prototype, we replace the laptop and Eee PC drives with OCZ Vertex 4 256GB SSDs. We calculate that a low-end laptop ($500) with an SSD ($200-300) and its share of the server cost ($200) is comparable to current desktop enterprise purchase budges of about $1000 to $1300.

Table IV
HARDWARE USED TO EVALUATE ANYWARE. NON-DESKTOP CLIENTS HAVE SLOWER CPU AND I/O PERFORMANCE.

Machine	CPU	Mem (GB)	HDD (RPM)
Desktop	Intel Core2 Quad, 2.40GHz	4	7200 or SSD
Mac Laptop	Intel Core2 Duo, 1.6GHz	4	5400 or SSD
Eee PC	Atom D425, 1.8GHz	2	5400 or SSD
Server	Intel Xeon, 12 cores, 3GHz	48	7200
User VM	4 cores	4	7200

VII. EVALUATION

This section quantitatively evaluates how closely Anyware can mimic desktop performance, answering the question

How does Anyware affect application performance and energy consumption?

The results in Table III suggest that storage I/O is critically important: Section VII-B evaluates Anyware using SSDs rather than HDDs, showing it matches and in some cases exceeds desktop performance. Section VII-C illustrates the benefits of a hybrid approach by allowing applications with I/O bound workloads to run locally. Next, Section VII-D discusses how Anyware's performance is affected when multiple users share the server's resources. The section concludes with an analysis of Anyware's energy savings.

A. Experimental Methodology

Our experimental Anyware setup has a client at the user's desk and a backend virtual machine running on a shared server. We use a midrange desktop to compare Anyware to a more traditional setup. The laptop, desktop, and VM all run on 64-bit Ubuntu 11.10. The server has Ubuntu 10.04 LTS. All machines are on the same VLAN; the client machines are located on the second floor, while the server resides in the basement server room. Average ping time between client and server is 270us with a standard deviation of 44us. The server is a 12-core, 3GHz Xeon Anyware server with 7200RPM drives and 48GB of RAM. Clients are a Macintosh PowerBook laptop or an Eee PC. Choosing two points in the low-end PC design space allows us to see how client hardware variations affect performance. Table IV contains the details on each device.

To simulate realistic user interactions with applications, we use a Perl package that automates interaction with X11 GUIs [16]. Because we are interested in performance bottlenecks, the script executes commands as quickly as possible by waiting for and firing X11 events. For example, to open a file the script may send the keys `Ctrl+O` to the application, wait for the file open dialog to open, then send keys to open the correct file.

We use three sample workloads used to evaluate Anyware. The first is in the Gnumeric spreadsheet application. It

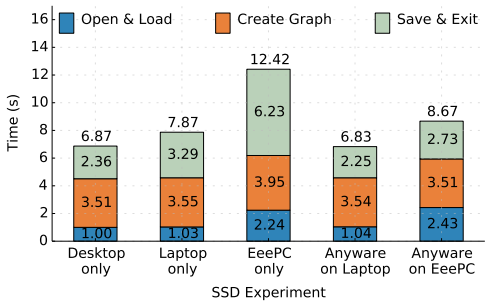

Figure 3. On the Gnumeric Anyware performs identically to the desktop. The results are averaged over 10 runs, which are all within 0.6% of each other.

involves opening a data file, selecting the data, and creating and saving a graph. The second is CPU-bound. It consists of opening a large image in GIMP and applying the 'Van Gogh' visual filter to the picture. The third task involves opening a 3.2MB text file, editing it by adding a few sentences, then saving and exiting the Kate text editor application. In addition, we also play a video to evaluate a case in which remote application execution is not desirable. The following section presents Anyware's performance on these workloads.

B. Anyware Client Performance

Spreadsheet. Figure 3 shows the average performance of the Gnumeric workload for different setups, all using SSDs. Comparing to the data in Table III reveals that the desktop does not benefit tremendously from the SSD addition since it already has a relatively fast hard drive. All other setups, on the other hand, see a significant reduction in completion time. Most notably, adding an SSD to Anyware makes its performance on the laptop comparable to the desktop's; it takes 6.87 seconds to create and save a graph on the desktop and 6.83 seconds using Anyware.

Since it is a hybrid setup, Anyware gets the best of both worlds – a fast data drive on the local side and a fast processor on the remote side. Delays due to the network transfers are negligible. While the laptop spends 3.29 seconds saving a figure (which involved CPU due to image compression), Anyware is able to do the same over the network in 30% less time, outperforming even the power-hungry desktop. The Eee PC+Anyware setup is within 10% of the desktop – a performance degradation that will be acceptable to most user.

Image Editing. Both Anyware configurations complete the GIMP task faster than the 100+ watt desktop. Figure 4 summarizes the results; data are averaged over ten runs and total application execution time is within 3% for all iterations. The Eee PC sees an improvement of 10 seconds over the hard-drive case, to a total execution time of 147 seconds (bar omitted in figure.) This is an example of a task that makes the most out of the additional resources made available by Anyware.

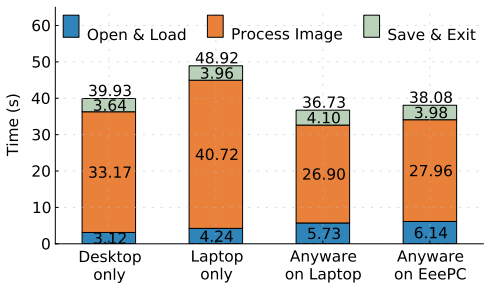

Figure 4. The Van Gogh GIMP filter is a CPU-intensive task so both the laptop and the Eee PC benefits from Anyware's access to the server. The missing 'Eee PC only' bar is at 147 seconds.

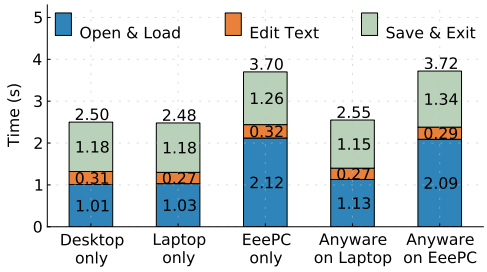

Figure 5. Editing a 3.2MB text file with the Kate Editor has comparable performance for all three setups, with variation between runs of at most 0.25 seconds.

Text Editing. Lastly, Figure 5 presents data on the text edit workload. Anyware performance match that of the powerful desktop. The Eee PC is slower due to its lower memory capacity; the entire slowdown is due to application startup, which does not affect productivity on longer tasks.

Cold Cache. To present a complete picture of Anyware's performance profile, we execute the three task on just-booted hardware, with empty data, instruction, and NFS caches. This is similar to starting an application immediately after booting a desktop. Figure 6 shows these worst-case data for the spreadsheet and text edit workloads. The GIMP workload is not affected, compared to running on the desktop because of the remote CPU gains. The Gnumeric task can be slowed down by as much as 35% (using the Eee PC) and the text

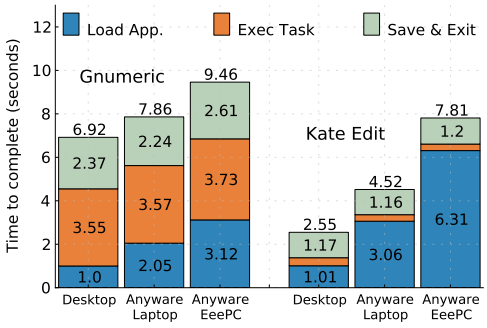

Figure 6. Executing applications with a cold cache represents worst-case performance for Anyware, similar to a just-booted desktop.

Table V
REMOTE VIDEO PLAYBACK RESULTS IN LOW FRAMES-PER-SECOND AND DEGRADED VIEWING EXPERIENCE. ANYWARE HAS THE BENEFIT OF BOTH LOCAL AND REMOTE RESOURCES SO SUCH TASKS WILL REMAIN LOCAL.

Setup	Not Decoded	Dropped	Total	% Not Displ.	FPS
Desktop	1	2	14308	0	24
Laptop	0	5	14309	0	24
Remote	1670	2923	12639	32	18.08

edit suffers the most due to to the large amount of data it needs to load. The user will experience such performance on rare occasions. We leave it as future work to modify Anyware so that it pre-caches applications to produce the performance of a warm cache at all times.

Summary. The time-to-completion data in this subsection highlight two important points. First, the switch from hard drives to SSDs illustrates how choosing the right hardware in combination with a new system design can produce the needed performance. Second, testing Anyware on two different client machines shows that flexibility in the client setup can yield different results. In a practical deployment, the IT staff and users can make a hardware decision based on a balance between hardware, price, and energy.

C. Local-only Applications

Section V revealed that some applications cannot be offloaded. Workloads that require large transfer of data over the network or that take advantage of special hardware, e.g. acceleration for graphics, will perform better locally. Google Earth's data traffic and heavy graphics makes remote-execution unsatisfactory. Another such application is video.

In order to quantify what a user might call 'unsatisfactory performance', we play a 10 minute video through mplayer. Table V shows the result when the video is played locally and remotely (using Anyware.) The percent of undisplayed frames is a user-centric metric for quality. The video player can skip displaying a frame for two reasons – decoding takes too long or it is too late to even start decoding the frame.

Running mplayer remotely using Anyware results the video playing at 18 frames per second. In contrast, the laptop and desktop deliver 24 FPS. The unacceptable remote-run performance of the video task is not a drawback for Anyware. Instead, it validates the need for a hybrid computing approach that makes use of local resources when those are needed. Thin clients cannot accommodate such tasks, leading to unsatisfactory user experience.

D. Sharing Server Resources

Section VII-B showed that Anyware can achieve acceptable performance for a variety of workloads. However, the measurements so far have been collected under ideal conditions with no additional load on the server. One of the keys ways in which Anyware saves energy is by consolidating

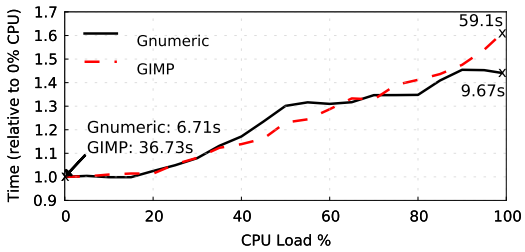

Figure 7. As pre-existing CPU load on the server increases, so does the execution time of workloads. An offloading policy should take into account current utilization before assigning tasks for remote execution.

user workloads on the server. Therefore, it is important to understand how a single user's experience will be affected by sharing server resources. We stress three components of the server – the memory, CPU, and network and rerun the three test workloads using the laptop+SSD hardware.

To introduce server memory pressure, we use a perl script that allocates memory and keeps it live by reading it. The memory usage increases from 0% to 95% (4.8GB). There is no observed effect on execution time for the Gnumeric and Kate workloads and an increase of one second in the GIMP execution time at 90% of memory use.

To generate pre-existing CPU load, the server runs multiple threads of a python script that does a variety of computations. The CPU utilization increases in steps of 5%, until it reaches 99%. The I/O-bound text edit task is not affected. The other two tasks experience increasing execution time as CPU utilization goes up.

Figure 7 shows the performance of the Gnumeric and GIMP tasks at different points of CPU load. For Gnumeric, execution time increases from 6.7 seconds to 9.7 second at 99% CPU utilization. The effect of sharing the CPU is even more visible in the case of GIMP, which is a CPU-intensive task. There is a steady increase in execution time which accelerates past 20% utilization. At 30% utilization, Anyware still completes the GIMP task as fast as a desktop (40 seconds). At its worst, when the 99% of the CPU is occupied by other tasks, it takes 60% (20 seconds) longer to execute the task, compared to what it would have on a standard desktop.

We use *iperf* to generate additional network traffic going into the server. The traffic load increases in steps of 100Mbps to maximum of 960Mbps. There is no effect for the GIMP application and minimal effect for the spreadsheet one. As expected, the text edit task is affected. Its completion time becomes visibly variable, between 2.5 and 3.4 seconds, when the traffic on the link is past 500Mbps. A breakdown reveals that the extra 0.9 second is from opening files.

VIII. ENERGY SAVINGS

Anyware has a low energy footprint because server resources are shared among users. We estimate that one server can support about 25 users VMs. The choice of this number is backed by data from three real-world thin client IT

Table VI
AVERAGE POWER DRAW OF DIFFERENT TYPES OF COMPUTING EQUIPMENT IN WATTS. THE PER-USER SERVER VALUES ASSUME 25 VMS PER ONE PHYSICAL SERVER.

Equipment	Idle (W)	100% CPU (W)
Desktop	100	165
Server (total)	130	270
Server (per user)	5	11
Laptop	14	24
Eee PC	13	22
Anyware (per user)	**18-19**	**33-35**

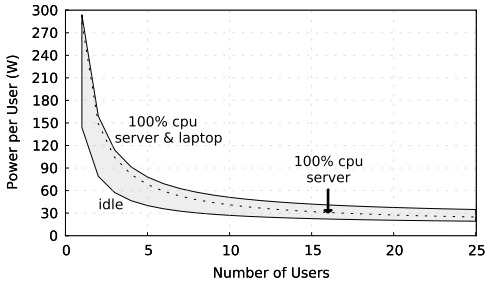

Figure 8. Anyware's energy savings depend on how many users can share the same server. Total per-user power draw lies within the shaded region.

setups. Two of them are in administrative departments at a university; the other is an academic department.

Using a thin client setup for making Anyware estimates is acceptable because in those types of deployments all work is done at the server. Therefore, if the typical thin client office can support 25 users with no local processing power, Anyware will be able to support *at least as many*, with a subset of tasks being completed on local machines. The three compute setups that inform the 25-VM assumption are supported by the following hardware:

- 20 user VMs: two E5450 Xeon quad-core 3GHz CPUs with 32GB RAM;
- 30 user VMs: two dual-core AMD Opteron 2.8GHz CPUs with 16GB RAM;
- 55 user VMs: Dell PowerEdge R810 with four 6-core 2.0GHz processors and 256 GB RAM.

Table VI summarizes the power draw of the equipment used to evaluate Anyware. The data is from empirical measurements collected at the device plug level; these numbers are lower than the data in Figure 2 which used maximum power dissipation data. The data do not include LCD screens; assume that users will use the same display regardless of PC choice.

In the best case scenario, when the setup is idle, the Anyware setup consumes fewer than 20 watts. This is an 80% decrease from the desktop running at 100W. In the worst-case scenario from an energy perspective, at 100% utilization, the hybrid setup will still draw only 35 watts.

Anyware's energy savings are subject to our specific equipment setup and the assumption of 25 VMs per server. Figure 8 shows the total per-user power draw changes as the number of users per server varies. Even a system that can support 10 clients per server will have a power draw of roughly 35 watts, or a 65% reduction over desktops.

IX. RELATED WORK

Sleep systems. Many energy saving techniques for PCs involve switching a PC to sleep mode when idle. In practice, however, users rarely activate these features and IT departments disable them to make patching and backups easier.

Sleep proxies were proposed [17] as a way to facilitate access to machines in sleep mode. Hardware [12] and software [8] proxies can reply to network packets on behalf of the host or wake it up. A real-world deployment of one such architecture achieved energy savings of about 20% [11]. More recently, GreenUp [18] has demonstrated that a distributed proxy can save 31% over a setup with always-on desktops. SleepServer [2], in contrast, proxies applications in trimmed-down virtual machines, reducing energy consumption by up to 60%. LiteGreen [1] runs the entire user desktop environment in a virtual machine, which can be migrated to a server so the desktop can be put to sleep, achieving savings of up to 74%.

Anyware's key difference is that it achieves comparable savings without requiring machines to sleep. This means that users never have to worry about whether their work environment is immediately available.

Remote Execution. The idea of offloading execution has been explored in the past [19], [20]. Most recently, migration of code at runtime has been used in smartphone-based systems such as MAUI [21] and CloneCloud [22]. These systems migrate portions of an executable to more powerful machines in the cloud. This results in lower energy use on the mobile device and faster completion of workloads. Programmers using MAUI are required to annotate methods for offloading. CloneCloud eliminates this need for special support for applications running in a application-layer VM. This assumption simplifies the task of migrating pieces of code on mobile devices but makes it impractical for office computing environments. To overcome these challenges, Anyware migrates a complete application to a minimally configured VM on the server, along with the relevant executables configuration files.

X. DISCUSSION

Anyware shows that it is possible to build an invisibly different enterprise computing infrastructure that consumes only one fifth the energy of systems today. This hybrid computing system, by using both highly energy efficient clients as well as high performance servers, achieves these savings without sacrificing performance and therefore productivity. Furthermore, Anyware supports disconnected operation. If a

person takes their client home or the network fails, all that happens is some tasks that would run better remotely run locally. The system also provides flexibility in its elasticity. The ratio of servers to clients can scale with the degree of demanding applications a particular enterprise runs.

The current Anyware implementation uses existing computing devices. One interesting question to ask is, were Anyware to become a common computing model, how would one design a client for it? Section VI argued that optimizing I/O is the most critical step. Section VII demonstrated that an Anyware setup that has a high performance I/O architecture can significantly improve performance; it can also reduce the cost of cold caches. Multiple SSD drives, each on a separate I/O channel, can improve read/write throughput. Such an approach would more closely resemble embedded systems and storage area network architectures (e.g. HP's 3PAR [23]) than traditional clients. Solid state drives mean improved performance does not have to cost energy.

Mobile devices have been tremendously successful at bringing compute resources at a very low energy cost. Many modern laptop CPU chipsets draw about 15 watts when idle and 20-30 when active. At the same time, tablets and smartphones are getting increasingly more capable at power draws of under 10 watts. An Anyware client can take advantage of the progress made in mobile computing, in combination with features common on standard PC to enable an energy-efficient system with elastic resources. Essentially, the Anyware client becomes a storage device, with a low-power processor attached for local applications, and GPU or perhaps ASICs for video processing since many local tasks are graphical in nature.

Anyware is a novel system architecture that improves enterprise computing efficiency by exploiting the non-linear relationship of power and performance of computing systems and the generally low demand of enterprise workloads. With right choice of hardware, Anyware does not require to sacrifice performance. Unlike sleep techniques, Anyware is the first system to provide users with an always-on environment whose performance is comparable to a powerful desktop but at a fraction of the energy cost. Anyware's effectiveness presents an alternative direction for future personal computers, as storage-centric devices able to run some applications locally while syncing with the cloud.

REFERENCES

[1] T. Das, P. Padaala, V. Padmanabhan, R. Ramjee, and K. Shin, "LiteGreen: Saving Energy in Networked Desktops Using Virtualization," *USENIX Annual Technical Conference*, 2009.

[2] R. G. Yuvraj Agarwal, Stefan Savage, "SleepServer: A Software-Only Approach for Reducing the Energy Consumption of PCs within Enterprise Environments," *USENIX Annual Technical Conference*, 2010.

[3] L. A. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[4] "Processor Performance and Power Data." http://www.notebookcheck.net.

[5] "3D Computer Benchmarking." http://en.wikipedia.org/wiki/3DMark.

[6] S. Dawson-Haggerty, S. Lanzisera, J. Taneja, R. Brown, and D. Culler, "@scale: Insights from a Large, Long-Lived Appliance Energy WSN," *Conference on Information Processing in Sensor Networks, SPOTS Track*, 2012.

[7] M. Kazandjieva, B. Heller, O. Gnawali, P. Levis, and C. Kozyrakis, "Green Enterprise Computing Data: Assumptions and Realities," in *International Green Computing Conference*, 2012.

[8] S. Nedevschi, S. Ratnasamy, J. Chandrashekar, B. Nordman, and N. Taft, "Skilled in the Art of Being Idle: Reducing Energy Waste in Networked Systems," in *Networked Systems Design and Implementation*, 2009.

[9] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta, "Somniloquy: augmenting network interfaces to reduce pc energy usage," NSDI'09, (Berkeley, CA, USA), pp. 365–380, USENIX Association, 2009.

[10] M. Kazandjieva, B. Heller, O. Gnawali, W. Hofer, P. Levis, and C. Kozyrakis, "Software or hardware: The future of green enterprise computing," Tech. Rep. CS TR 20011-02, Stanford, July 2011.

[11] J. Reich, M. Goraczko, A. Kansal, and J. Padhye, "Sleepless in Seattle No Longer," *USENIX Annual Technical Conference*, 2010.

[12] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta, "Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage," in *NSDI'09*, 2009.

[13] S. Cheshire and M. Krochmal, "DNS-Based Service Discovery." RFC 6763 (Proposed Standard), Feb. 2013.

[14] "Avahi Zeroconf Software." http://en.wikipedia.org/wiki/Avahi_(software).

[15] "Logistic Regression." http://en.wikipedia.org/wiki/Logistic_regression.

[16] "X11::GUI Test - Perl Package for User Emulation." http://sourceforge.net/projects/x11guitest/.

[17] B. Nordman and K. Christensen, "Improving the Energy Efficiency of Ethernet-Connected Devices: A Proposal for Proxying," *Ethernet Alliance*, 2007.

[18] S. Sen, J. R. Lorch, R. Hughes, C. G. J. Suarez, B. Zill, W. Cordeiro, and J. Padhye, "Sleep Over Availability: The GreenUp Decentralized Wakeup Service," in *Networked Systems Design and Implementation*, 2012.

[19] J. K. Ousterhout, A. R. Cherenson, F. Douglis, M. N. Nelson, and B. B. Welch, "The sprite network operating system," *Computer*, vol. 21, pp. 23–36, Feb. 1988.

[20] A. S. Tanenbaum and S. J. Mullender, "An overview of the amoeba distributed operating system," *SIGOPS Oper. Syst. Rev.*, vol. 15, pp. 51–64, July 1981.

[21] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer with Code Offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10.

[22] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic Execution between Mobile Device and Cloud," in *Proceedings of the sixth conference on Computer systems*, EuroSys '11, ACM, 2011.

[23] "HP 3Par Architecture." http://h18006.www1.hp.com/storage/solutions/3par/architecture.html.