Saving Portable Computer Battery Power through Remote Process Execution*

Alexey Rudenko[a] **Peter Reiher**[a] **Gerald J. Popek**[a,b] **Geoffrey H. Kuenning**[a]
arudenko@fmg.cs.ucla.edu *reiher@fmg.cs.ucla.edu* *popek@platinum.com* *geoff@fmg.cs.ucla.edu*
[a] Computer Science Department, University of California, Los Angeles, CA, USA
[b] Platinum *technology, inc.*, Inglewood, CA, USA

We describe a new approach to power saving and battery life extension on an untethered laptop through wireless remote processing of power-costly tasks. We ran a series of experiments comparing the power consumption of processes run locally with that of the same processes run remotely. We examined the trade-off between communication power expenditures and the power cost of local processing. This paper describes our methodology and results of our experiments. We suggest ways to further improve this approach, and outline a software design to support remote process execution.

I. Introduction

Power management is one of the most challenging problems in making portable computers more useful. Portable computers have their greatest utility when they can truly be used anywhere at any time, and one of the greatest limitations to that goal is battery power. Often AC power connections are not available and the portable computer must run off its battery. However, the battery life of existing and expected batteries is not sufficient for many situations. Users must either alter their behavior or limit their use of the portable computer to preserve the battery's charge. Any user whose portable computer has run out of power while in the middle of a long air flight understands the impact of insufficient batteries.

If the battery's power capacity cannot be improved, the other alternative is to find ways to use less power, preferably with no impact on the user. Many researchers have looked at this problem [3], [14], [19]. Solutions range from intelligent management of the disk and screen [4], [7], [8], [13] to slowing down the CPU clock rate [6], [23] or powering down components of the computer not currently in use. Many of these innovations have already found their way into commercial use, a strong indication of the importance of the problem.

Wireless communication devices are becoming increasingly common in portable computers, since they help solve one of the other fundamental problems of portable computing: that is, when wired connections are unavailable, wireless devices can maintain network connectivity, allowing remote file access, sending and receiving of email, and web browsing. Generally, wireless communication devices are regarded as contributing to the power management problem, rather than the solution, as they use significant power when sending and receiving. However, this paper demonstrates that wireless devices can sometimes be used to save a significant amount of battery power.

A portable computer's battery power is drained by performing tasks for the user. Some of these, by their nature, must be performed locally. For example, information must be displayed on the local screen. Other tasks, however, could be performed anywhere, provided the results came back to the portable computer. If the power cost of sending the task elsewhere and receiving the results back is lower than the cost of running it locally, remote process execution could save battery life for the portable computer.

This possibility makes little sense without wireless devices. If the user can plug in a wired Ethernet card, he could probably also plug in a power cord. But an untethered user can still communicate via wireless networks. One realistic scenario is an office in which untethered users can move around through a ubiquitous wireless network, migrating processes to server machines that have no power constraints.

In its simplest form, remote process execution for power management would involve moving new tasks from the portable computer to a server machine before the task start running. The server would execute the task and ship the results back to the portable computer. In the meantime, the portable computer would continue running other tasks, going idle if nothing else is to be done. If the user runs many tasks that drain a lot of power, and the costs of moving the tasks to and from a remote server are low enough, remote execution could save a large amount of power and allow portable computers to run untethered for much longer.

Since wireless cards themselves consume significant amounts of power, there is no guarantee that migrating typical realistic tasks would actually save power. This paper describes experiments that have proved that such power savings are possible, and that the size of the savings can be very significant. We compared the amount of power consumed by migrating various large tasks off portable computers over a wireless device to the amount of power consumed by running the tasks locally on the same portables. We discovered that significant power savings are possible for certain common tasks of realistic size. This paper describes our experimental methodology, presents our results, analyzes those results, and suggests what would be required to make remote process execution a feasible power management tool for realistic environments.

II. Experimental Methodology

We ran an extensive series of experiments to determine if remote process execution could save significant amounts of battery power. We identified a set of applications likely to profit from remote execution and ran them in both local and remote modes, measuring the battery power consumed in each case. The experiments consisted of requesting a task on an untethered client machine, and either running it locally or migrating

*This work was partially supported by the Advanced Research Agency Projects under contract DABT63-94-C-0080.

it to a tethered server machine. In the latter case, the server ran the application and shipped the results back to the client when completed.

This brief description ignores many important issues that might influence the outcome of the experiment. Here we describe all the conditions under which we ran the experiments. The results of the experiments are presented in Section 3.

The experiments were conducted on Dell Latitude XP portable computers running the Linux operating system. Dell's rechargeable Li-ion battery rating is 14.4V 2200mAh, 2A. Both the client and the server were Dell Latitude XPs in this experiment. In realistic situations the server would likely be a more powerful machine. We chose to use the same machine for both client and server for two reasons. First, it was convenient for experimental purposes, since the wireless devices available to us did not fit into any of our other machines, and it allowed us to avoid issues of possible incompatibility between the client and server machines. Second, it was a conservative choice, since in realistic circumstances the server would run faster, thus causing the client to lose less power while waiting for its results.

The wireless device used for the experiment was the 915 MHz AT&T Wavelan card [1]. This network adapter consumes 250 mW to achieve a data rate of 2 Mb/sec at ranges up to 800 feet. In the basic experiments, we took great care to insure that the Wavelan cards in the client and server were the only devices using that portion of the electromagnetic spectrum within range of each other. The experiments were performed in a location containing no other Wavelan cards or other wireless communications devices. For the baseline results, we needed to know whether remote execution could ever win, even in rather favorable circumstances. We added interference to the channel in later experiments, as described in Section 3.2.

We chose three applications for test cases. These applications were chosen because they were known to be fairly large and time-consuming, making them good candidates for power savings by remote execution. The three tasks were a compilation of a large program, text formatting of a 200-page document using LaTeX, and Gaussian solution of a system of linear algebraic equations. The compilation task performed significant CPU processing, along with a good deal of disk activity to read the source, write and read temporary files, and write the resulting object and executable files. The text formatting task performed a moderate amount of CPU processing, but relatively little disk activity. The Gaussian elimination problem performed trivial amounts of file access, but made very heavy demands on the CPU and on memory. The size of the matrix varied, but was never large enough to cause significant amounts of virtual memory activity.

In all cases we were able to vary the amount of information that had to be sent from the client to the server machine. In the cases of the compilation and text formatting tasks, we arranged it so that the client and server each had copies of the source, but that the client had altered some varying fraction of the source. Thus, to run the process at the server the client had to move only the altered fraction of the source. The client and server stored previously computed object files corresponding to unchanged sources, so the amount of work required to perform these tasks varied depending on the fraction of data changed. The amount of data shipped back by the compilation and text formatting jobs was constant, since the executable or Postscript document was always the same size. In the case of the Gaussian solution process, the entire source matrix had to be moved to the server, but we varied the size of the matrix, thus varying the amount of data shipped and the amount of work done. A larger source matrix required shipping a larger result vector as well.

Neither the client nor the server performed any other user-level activity during the course of the experiment. We made no attempt to prevent the operating system from performing its normal housekeeping activities, but we turned off many of the Linux daemons that would typically run periodically in the background. The periodic intrusion of these daemons into the experiment could cause an undesirable noisy impact on power-cost measurements. This decision allowed us to isolate the actual costs and benefits of executing the tasks locally and remotely. In normal conditions these daemons would slightly shorten the battery's life.

Existing power-management techniques can have a dramatic impact on the amount of power consumed by a portable computer running off its battery, so we controlled these techniques carefully. Our research showed that screen timeout and disk spindown were the most important power-management tools. We set the screen timeout and disk spindown intervals to one minute on the client machine. We also allowed the client to go into idle mode during remote execution. We did not permit the client machine to suspend, since the server would be unable to ship the results back to a suspended machine. Since the server was connected to AC power, it performed no power management. Usually, the client screen timeout would occur during local execution, but disk spindown would not. Typically, both screen timeout and disk spindown occurred on the client during remote execution. The client generally went into idle mode during remote execution. These power-management settings are similar to the ones our portable computer users typically set for normal situations.

We did not attempt to turn off the Wavelan card at any point in the experiment. The Wavelan consumes 1.48 watts even when it is neither sending nor receiving, so there were significant power costs to not turning the card off. However, if the card is turned off, timeout or human intervention would be required to turn it back on when the results were to be shipped back from the server. Doing so would have been neither realistic nor convenient for running the experiments.

Measurement of the power consumed by a task caused some problems. The most accurate way to measure power consumption would have been to insert appropriate electronic instrumentation between the battery and the computer it was driving. Practical problems with such instrumentation suggested the use of less direct methods. The readily available metrics for power currently in a battery are rather unreliable. The Dell Latitudes used in these experiments use the Advanced Power Management (APM) tools, which will report a battery's charge as a percentage of maximum. However, the APM measurements are not very reliable. In some cases, the amount of power reported by APM will go up over time, even though the machine has remained untethered. In addition, the amount of power expended to perform a particular task, as reported by APM, varies widely.

We considered two options for measuring the amount of power consumed by a task. First, we could interrogate the APM metric before the task, interrogate it again after the task completes, and report the difference. Alternately, we could completely charge a battery, repeatedly run the task until the battery dies, and divide 100% battery life by the number of executions required to drain the battery. Our experiments showed that the two methods produced substantially similar results, but the use of the APM metric gave more stable variances then the alternative method for the same number of runs. The results presented here rely on the APM metric.

Our experimental methodology was to fully charge the battery of the client portable, then repetitively perform tasks (either locally or remotely) until no battery power remained. We measured and recorded the power consumed for each task. Because of the noisiness of the APM metric, we performed numerous runs to achieve sufficiently low variance. Since the number of runs required caused the battery to discharge and recharge hundreds of times, we were concerned that the battery's power storage and consumption characteristics might change over time. However, measurements done at the beginning and the end of the experimental period showed no statistically significant difference in the battery's capacity.

III. Experimental Results

We ran 220 experiments, consuming approximately 900 hours, to obtain the data presented here. Typically, each point plotted on the curves represents four to eight hours of experimentation. All results are shown with 95% confidence intervals. The compilations used real software packages designed in our laboratory. The LaTeX texts were real papers and dissertations written in our laboratory. As a result, some file sizes are not round numbers.

A. Noiseless environment

The first part of the experiments was run in a noiseless environment. All major sources of noise (like other laptops equipped with wireless cards) were isolated from the room where the laptop experiments were performed. Figure 1 shows the power consumption for local and remote execution of the compilation process. The left bar of each pair shows the amount of power used for local execution of the task, and the right bar for full remote execution.[1] The X axis shows the number of kilobytes of C source code that were altered. The amount of work required to perform the compilation thus varies from point to point. Also, the amount of data shipped over the wireless link for remote execution varies with the amount of altered code, since only altered modules were shipped. In the remote-execution case, the server shipped back only stripped executables, object files were not shipped. In the local case, stripped executables (6.9MB) were saved directly to disk. (We performed the same experiment with unstripped executables. The results were qualitatively the same, though the transmission costs for the larger executables made the percentage improvements smaller.)

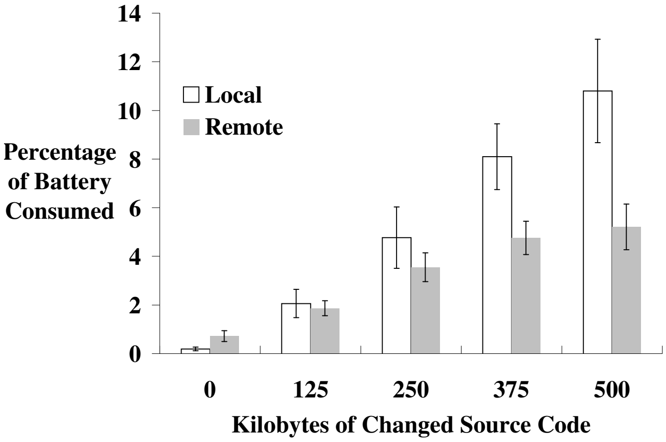

Figure 1: Power savings for remote execution of a compilation

For small amounts of changed code, which corresponds to small amounts of work to be done by the compilation, moving the task to the remote server consumed more power than local execution. The power cost of receiving the executables from the remote machine, combined with power wasted by the portable while waiting for the result, dominated any benefits. However, as the amount of work increased, the value of moving the work off the portable computer became clear. For 500 kilobytes of altered source, shipping the task to the remote server consumed less than half the battery power needed to compile locally.

Some of the power cost of remote execution is due to using the Wavelan card to move the data to the server and the results back to the portable computer, while other components of the cost reflect power wasted by the portable computer while waiting for the results to come back. If one assumed that the server had significantly more compute power than the portable computer, the waiting period would have been much shorter, since the more powerful server would have completed the compilation faster. To isolate this effect, we measured the costs of pure transmission of the required data and results. We precompiled the source code for each case and stored the results on the server. When the client portable requested remote execution of the task, instead of compiling we shipped the saved results back immediately. This experiment shows the effect of an extremely powerful server and gives some insight into the amount of power the portable computer uses for data transport versus the amount of power spent waiting idly for the server to complete the task.

Figure 2 shows the difference between the power consumed by local execution, by remote execution, and by simply shipping data and results back and forth, using the compilation application shown in Figure 1. The left bar of each group shows the amount of power used for local execution of the task, the middle bar for full remote execution, and the right bar for simply shipping the data and results over the wireless network.[2]

For zero bytes changed, the cost of remote execution is statistically indistinguishable from the cost of transmitting the data and the results. Since the server stores the results of the compilation with no changes made, this case is identical to the pure transmission case, though minor amounts of work are

[1] The regression equation is $y = 2.7x + 3.0$ for local execution and $y = 1.2x - 0.3$ for remote execution The R^2 values are 0.99 and 0.97, respectively.

[2] The regression equation is $y = 2.7x - 3.0$ for local execution, $y = 1.9x - 0.3$ for remote execution, and $y = 0.2x + 0.5$ for transportation cost. The R^2 values are 0.99, 0.97 and 0.88, respectively.

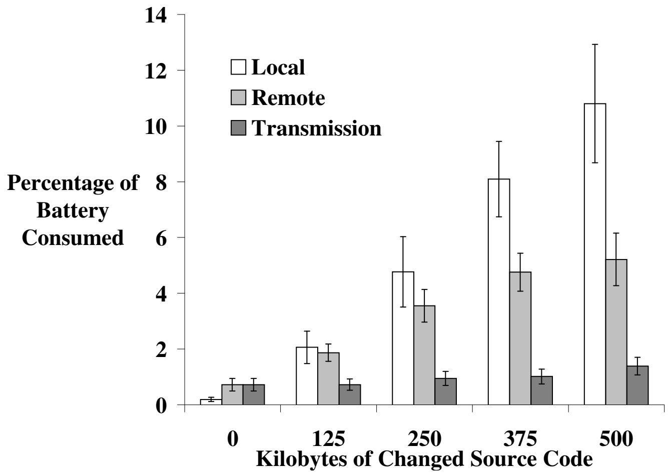

Figure 2: Total power cost of local and remote execution compared to transmission costs

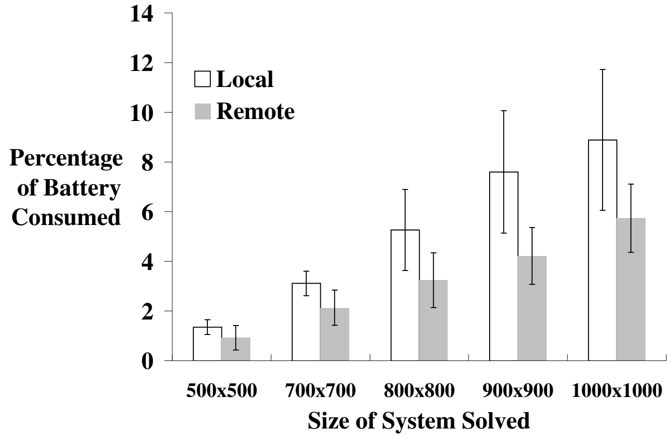

Figure 3: Power savings for remote execution of Gaussian solution of a system of linear algebraic equations

done on the server to verify through a make utility that no recompilation is required. In the other cases, however, note that the transmission costs are half or less of the costs of the remote execution. The remainder of the costs occurs because of various inefficiencies in how the portable computer behaves while waiting for the compilation to complete. For example, its Wavelan card enters idle mode for this time, but the idle mode still consumes about 1.48 watts of power. The disk continues to spin for at least one minute after it was last accessed, the screen displays an image for at least one more minute after the last key was pressed, and some other devices must be considered in that total as well. Thus, if the computer had a more efficient power-saving mode, remote execution would have saved significantly more power, bounded by the transmission costs shown in this figure.

Figure 3 shows the power saved by remote execution for the Gaussian solution of the system of algebraic equations.[3] In this case, the size of the task is controlled by the number of rows in the matrix. The entire matrix had to be shipped and solution vector returned, in the case of remote execution. For relatively small matrices, less than 500x500, the costs of moving the computation were greater than the benefits, or the results were statistically indistinguishable. But for larger matrices, the savings were as great as 45%.

Unlike compilation, the Gaussian application performed very little disk I/O. The savings shown on a large Gaussian solution thus demonstrate that performing tasks remotely can offer benefits even if the task is largely compute-bound. Transportation expenses for the shipment of the data grow slower than the workload, because they are quadratic with respect to the size of problem, while the workload grows at a rate of O(N3), where N is the number of simultaneous equations.

Figure 4 shows the results of migrating a text formatting application. This application used LaTeX to format 30- to 200-page documents containing multiple figures, references, and equations, making it a moderately large text formatting process.[4] Remote execution did not improve the power con-

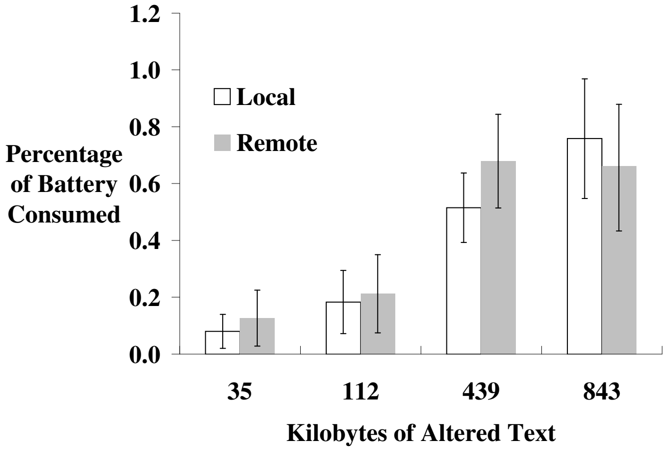

Figure 4: Power savings for remote execution of text formatting

sumed by this application. In most cases, there was no statistically significant difference between local and remote execution. Only for the case of 439 kilobytes of altered text was the difference significant at the 95% level, and in this case remote execution performed worse than local execution. There are several possible explanations for the text formatting application's failure to benefit from remote execution. The most obvious and most likely is that the application consumed less than 1% of the total battery power, even at its heaviest workload. With such minor power consumption, adding anything that itself consumes significant power (such as moving the result files back to the portable computer over the wireless link) is likely to have a major impact on the total power consumption. Note that the compilation and Gaussian elimination applications tended to consume 2% or more of the total battery power, leaving more room for paying an up-front penalty to reduce the overall costs.

The results presented so far were performed on an otherwise unused medium. The only traffic in the Wavelan frequencies in the testing area was generated by the migration of tasks and results. In a realistic environment, the Wavelan frequencies would be used for other tasks, including other machines also trying to execute processes remotely. We performed further experiments to determine the power- saving characteristics of large applications in the presence of noise on the wireless net-

[3] The regression equation is $y = 2.0x - 0.6$ for local execution and $y = 1.17x - 0.3$ for remote execution. The R^2 values are 0.99 and 0.99, respectively.

[4] The regression equation is y=0.24x-0 for local execution and y=0.2x-0.1 for remote execution. The R^2 values are 0.97 and 0.84, respectively.

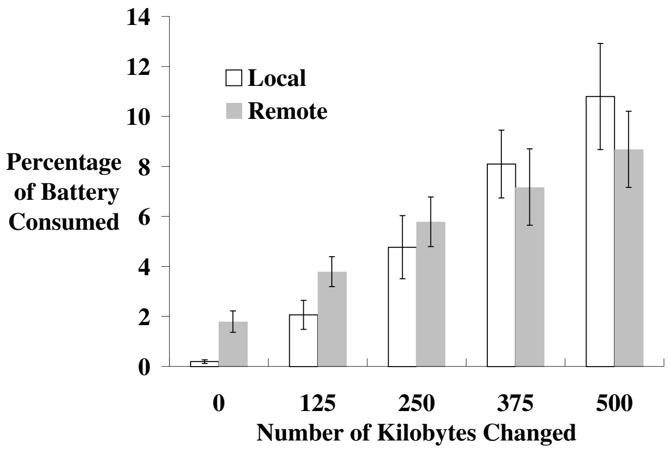

Figure 5: Power savings for remote execution of a compilation with noise in background

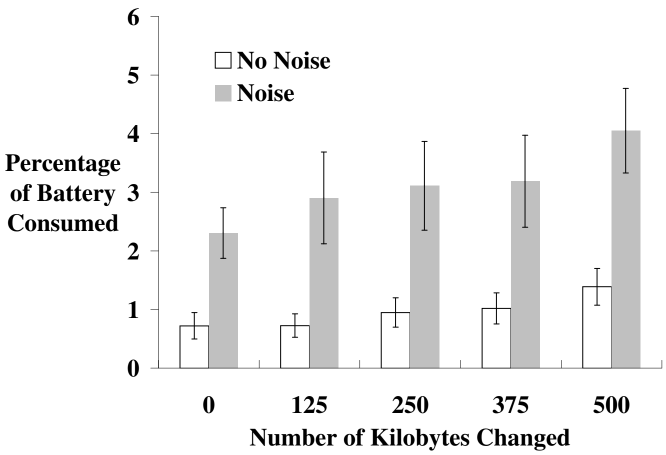

Figure 6: Power cost of transmission with and without noise for compilation

work. However, since text formatting did not show improvements in the noiseless case, we did not perform experiments for that application in the noisy environment.

B. Experimental data in noisy environment

Network noise has many different characteristics. Assuming multiple senders and receivers, and multiple patterns of traffic presented to the network, a vast number of different experiments are possible. Our goal was to examine a reasonable case, not to exhaustively examine the entire realm of possibilities. However, we did want a challenging case, not a trivial one, since the noise-free experiments had already given us best-case results. We chose to introduce two new machines in the environment. One of these machines opened a socket to the other new machine and sent data down that socket as fast as it could. We refer to this as saturated-socket noise. Since the Wavelan cards use an Ethernet-style protocol where collisions cause backoff and retransmission, we expected that the interfering with communications would significantly impede attempts to move data and results, but that it would not entirely block those activities.

Of course, it would be possible to introduce multiple pairs of communicating machines, or to direct some of the traffic to either the portable computer being tested or its server. Either of these options would be expected to cause even more problems, and either is a defensible ”realistic” situation, so in the future we may expand our experiments to include them. Other variations in the noise experiments are also possible, including more realistic forms of communication between the noise-making machines, such as file transfer or attempts by these machines to perform their own remote executions.

Figure 5 presents the effects of saturated-socket noise on the compilation task.[5] With this background noise, only the largest compilation saved a statistically significant amount of power by using remote execution, and its savings were only around 20%, as opposed to a 51% improvement without noise. Clearly a large amount of noise in the environment has a major effect on the power savings achievable by remote execution.

Again, the power consumed in this experiment is partly due to transmission costs and partly due to the power expended by the portable while waiting for its results. Given that the portable and the server now have to contend for the Wavelan radio spectrum with a very heavy consumer, more collisions and retransmissions occur while transmitting the remote execution data and results over the network, thus increasing the amount of power consumed. The remainder of power consumed is spent by the portable while waiting for the server to complete its task. This distinction is relevant because different techniques would be required to reduce the costs of each component. Reducing the power costs of transmitting in a noisy environment would involve changing the wireless protocol to cause fewer collisions, or to make collisions less costly. Reducing the power costs of waiting would involve either improvements in the portable's power management software, improvements in the wireless communications device's use of power, or reducing waiting times by using faster servers. We believe that there might be also other ways to improve the power cost characteristics of the remote processing.

The power-management tools available to us did not give any indication of which effect causes the use of power, just the amount of power used. Thus, we could not directly measure the fraction of power used for retransmissions versus the fraction wasted waiting for results. Instead, we indirectly measured the contributions of these effects.

The cost of retransmissions was approximated by measuring the power consumption of simple transmission. Figure 6 shows these costs. As in Figure 2, we measured the costs of transmitting the data, fetching the results off the server without recompiling, and transmitting the results back. Figure 6 shows the power consumed by this process without noise compared to the power consumed with noise.[6] These results are a reasonable approximation for determining the power cost of collisions and retransmissions due to noise. The figure shows that the increased cost of transmissions in the noisy medium consumed an additional 1.5% to 2.5% of the battery. We conjecture that the power consumed by increased transmissions is similar in the normal remote execution case.

These costs are significant, and they almost fully explain

[5] The regression equation is $y = 2.7x - 3.0$ for local execution and $y = 1.7x + 0.3$ for remote execution in noisy environment. The R^2 values are 0.99 and 0.99, respectively.

[6] The regression equation is y=3.8x+2.0 for the execution in a noisy environment and y=1.6x-0.48 for the execution in a noiseless environment. The R^2 values are 0.90 and 0.88, respectively.

the increased costs for remote execution in the face of noise as shown in Figure 5. The remaining component of cost is extra battery power consumed in the noisy case while waiting for the server to complete. This extra cost appeared to be almost negligible, so the portable's cost of waiting for a unit of time is approximately the same in the noisy and noiseless cases.

Due to space limitations, we do not show the results from the Gaussian elimination experiments with noise. They are similar to the results from the compilation experiments.

IV. Discussion and Future Work

Our results show that remote execution of large tasks can reduce their power consumption by up to 50%. Better power-management features added to portable computers in the future, and wireless communications devices that consume less power, will make the improvement even greater. The power expended by the portable computer while waiting for results to come back could be minimized by having idle modes that are entered either on command or very aggressively, and that consume very small amounts of power. The fact that there is a significant difference in the client portable computer's power consumption for actual remote execution versus pure transmission of data and results (Figure 2) shows that the amount of power expended purely on waiting is quite significant. Other experiments done earlier in our work suggested that if the user manually turns the portable completely off and turns it back on again when the computation is complete, the amount of power consumed by the portable is much, much lower. Of course, this is not a practical way to operate in the real world, but if the power-saving mode can come closer to expending zero power, remote execution will show more benefits.

A. Improving Power Management

The high cost of pure data transmission shown in Figures 1 and 6 and the cost of listening to the channel suggest that certain improvements in wireless communications devices could also improve the performance of remote execution. Wavelan wireless cards, and wireless communications cards in general, are fairly new phenomena that are not yet ubiquitous. As a result, they have not been heavily optimized. The Wavelan card, for example, operates in three modes: transmitting, receiving, and sleeping. These modes take approximately 3 watts, 1.48 watts, and .18 watts, respectively [1]. The transmission and receiving modes consume as much power as a typical disk drive, which is known to be one of the most power-consumtive devices in a typical portable computer. Other devices, such as the Metricom communications card, have reduced these figures by a significant amount. The Metricom card expends approximately 1 watt, .4 watts, and .1 watt for the same modes as the Wavelan card [22], but unfortunately at the cost of a much lower data rate than the WaveLAN. Proxim RDA radios spend 0.375 watts in transmitting and receiving modes and 0.001 watt in standby mode, with a data rate of 0.242 Mb/sec [16]. Future improvements can be expected.

One particular area for improvement is minimizing the amount of power spent by the local card in the power-costly receiving mode. The use of a broadcast medium, and the requirement that the portable computer receive messages that it might not be expecting (which is very useful for any number of purposes), mean that some power must be expended examining all incoming packets to see whether they are destined for the local node. Designs that are able to synchronize communication and allow the local card to spend some time in sleep mode without the loss of messages have promise for minimizing these power costs. Such designs will not only benefit power consumption for remote execution, but will generally allow portable computers with wireless cards to remain truly idle at minimal power cost in a shared wireless network environment. Communication cards that are able to remain alert even when the portable itself is in suspend mode, waking only when its own message arrives, would also be helpful.

B. A Power-Conserving Infrastructure

Although power savings are possible through remote execution, considerable work remains before these savings become available to average users. Our experiments used special scripts that performed each migration and sent the results back. Other scripts ensured that the server stored the appropriate data and that only the required data was sent from the client portable to the server. However, ordinary users cannot be expected to be able to create such tricky scripts themselves. Simply using normal remote execution has disadvantages as well. If the data files are stored on the portable, it will spend power moving them to the remote machine. If not, the portable cannot operate disconnected. Generally, a transparent facility would be preferable. Common use of remote execution for power management will require a user-friendly infrastructure.

This infrastructure will require several important components. First, it will require the ability to remotely execute a task and deliver the results back in an efficient way. Preliminary tests have shown that the mechanism used to transport the data from client to server and back can have a dramatic impact on the power expended.

Second, the infrastructure will require simple replication mechanisms that allow the client and server to synchronize replicas of the required data. Such replication mechanisms will allow users to ignore the difficult issues of exactly which pieces of data need to be sent to the server. Moreover, by running part of the replication algorithms when the portable computer is tethered (at least to power, perhaps to a wired network), the infrastructure can minimize the amount of data that would have to be sent to perform the task remotely. Intuitively, if an earlier replication operation had already moved all the data associated with one of the compilations in Figure 1, for example, the transportation costs shown in Figure 2 would have been lowered.

Third, proper remote execution of a job requires ensuring that all conditions at the server machine are the same as at the client. For example, if the server has an older version of a library than the client does, the resulting program will behave differently if compiled remotely than if compiled locally. Many other issues related both to the user's personal environment and preferences, and to the general system environment on the two machines, can make providing a consistent execution environment challenging. We hope to obtain insight on this problem from projects that execute processes remotely for different purposes [12], [17], [18]. These and similar projects have dealt successfully with these challenges.

Fourth, the infrastructure will need to assist the user in determining when to execute a task remotely. As the results in Section 3 show, only jobs above a certain size benefit from remote execution, and small jobs can actually waste power by executing them remotely. Therefore, the system must not execute all jobs remotely, and not even all jobs of particular types. The infrastructure can supply varying levels of support for remote execution. A very simple form of support would be to provide users with a command to execute a job remotely. While simple, this puts a heavy burden on the user to decide whether a particular job is suitable for remote execution. A more complicated alternative would be for the user to provide the system with hints about when a job is and is not likely to profit from remote execution. For example, makefiles could be augmented with hints about whether or not particular targets are likely to profit. In its most complex form, the infrastructure could attempt to deduce automatically whether particular jobs were likely candidates for remote execution.

A major component of this last problem is identifying exactly which characteristics of a task are likely to cause it to consume major amounts of battery power. Clearly, disk accesses are important, but it is less clear how many disk accesses are required for a job to be a good candidate. Very large processes that will require substantial amounts of virtual memory activity may perform few file system accesses, but may actually exercise the disk heavily. The results we obtained for Gaussian elimination clearly show that a sufficiently large CPU load alone may be enough to make a task power-expensive. A better understanding of which activities in which quantities consume a great deal of power must guide any approach to choosing jobs suitable for remote execution. Design of a suitable infrastructure for remote execution is the next phase of our research.

Substantial questions also exist in the realms of failure detection and recovery, security, and server design for remote execution support.

V. Related Work

Much research has been performed on power management, including measurement techniques, approaches, methods, technical tools, etc. Power measurement techniques for laptop devices and applications and benchmark strategies were discussed in [3], [14], and [22]. Many techniques to save laptop power are based on switching off or slowing down the most power-costly devices, such as the hard drive, CPU, and wireless network devices when they are not being used. [24], [19], [13], [4], [7], and [8] discuss different strategies to reduce hard drive power cost. Measurement results show significant power savings where real hard drive access patterns were successfully predicted. The prediction of the moment when a hard drive will be in use again is essential for all techniques based on this idea, and it is relevant to all other devices having inertia (floppy, CD, etc.). [23] suggests that power also can be saved through slowing down clock speed, with limited negative impact on performance. The authors of [2] use predictive caching to reduce contention on the narrow-bandwidth wireless channel, consuming less power and allowing a mobile laptop to keep working in circumstances of long and frequent disconnections. [6] proposes using idle periods to achieve a fairer distribution of the workloads of busy periods. This paper also includes a taxonomy of idle-detection algorithms and idleness predictors. The idle periods can be used to run some tasks whose results are needed in future, as well as for the disconnection of unused power-costly devices.

Wireless communication devices appear to be highly power-consumptive. [20] discusses a technique of transmission suspension at the moment when interference in the channel is detected. It is presumed that interference is stationary and ergodic. During the interference, the communication device can be suspended, and power consumption by this device reduced. [9] considered wireless data broadcasting as a way of disseminating information to a massive number of clients equipped with battery powered laptops. The user must periodically listen to the channel to obtain a consistent schedule of the data that will be transmitted in the near future. At other times, the user can disconnect his communication devices and save power.

Laptop and battery manufacturers have provided another effort towards reducing power consumption. Several specifications [10], [5], [11] address issues of power consumption. They are focused mainly on two points: providing system functions allowing connection/disconnection of any particular device from the power source, and getting statistics about current status of power consumption in the system, including remaining battery capacity. These functions serve as a hardware/software basis for multiple packages, such as Wildboar [21], which supplies a user with power management utilities that can be used in his scripts and applications.

[15] discussed a system designed around the InfoPad portable terminal, a network I/O device with no computation power, relying on network servers to run major processes. Local computing is not possible here, even when it is less power-costly.

For many years, process migration and delegation have been discussed in the computer community for various purposes: dynamic load balancing, improved reliability, reduced network traffic. [18], [17], and [12] discuss these issues. However, power consumption has not previously been identified as a benefit of process migration.

VI. Conclusions

The experimental results presented in this paper demonstrate that portable computers that execute their large tasks remotely can save significant amounts of battery power. Savings of up to 51% were observed. While the tasks in our experiments were large, they were not unrealistic. They represent tasks that are typically performed by ordinary users every day. Thus, assuming a suitable environment, remote execution has promise for providing better battery life for future users. This technique is largely orthogonal to other power-saving techniques, adding to any benefits they provide.

The savings shown in this study are not by any means the maximum savings possible. Larger tasks are likely to benefit even more. More efficient ways of moving the data may also provide greater savings. Improvements in wireless devices and power management will provide further benefits. Preliminary experiments under more optimistic conditions have shown up to a five-fold increase in battery life.

References

[1] AT&T Corp, WaveLAN/PCMCIA User's Guide.

[2] Daniel Barbar and Tomasz Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments," *ACM SIGMOD International Conference on Management of Data*, Minneapolis, Minnesota, May 1994.

[3] James W. Davis, "Power Benchmark Strategy for Systems Employing Power Management," *IEEE International Symposium on Electronics and Environment*, 1993.

[4] Fred Douglis, P. Krishnan, Brian March, "Thwarting the Power-Hungry Disk," *USENIX Conference Proceedings*, Winter 1994.

[5] Duracell Corp./Intel Corp, Smart Battery Data Specification, Revision 1.0, February 15, 1995.

[6] Richard Golding, Peter Bosch, Carl Staelin, Tim Sullivan and John Wilkes, "Idleness is Not Sloth," HP Laboratories, Technical Report HPL-CCD-95-1, Palo Alto, CA, January 1995.

[7] Paul M. Greenawalt, "Modeling Power Management for Hard Disks," *Mascots 94, International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, June 1994.

[8] David P. Helmbold, Darrell D. E. Long and Bruce Sherrod, "A Dynamic Disk Spin-down Technique for Mobile Computing," *Proceedings of the Second Annual ACM International Conference on Mobile Computing and Networking*, Rye, NY, November 1996.

[9] Tomasz Imielinski, S. Viswnathan, B.R. Badrinath, "Energy Efficient Indexing on Air," *Proceedings of ACM-SIGMOD Conference on Management of Data*, May 1994.

[10] Intel Corp. Pentium Processor (610
75) Power Consumption, Revision 1.1, Technical Marketing, October 1994.

[11] Intel & Microsoft Corp. Advanced Power Management (APM) BIOS Interface Specification, Revision 1.2, February 1996.

[12] Deitmar A. Kottmann, Ralph Wittman and Markus Posur, "Delegating Remote Operation Execution in a Mobile Computing Environment," *Mobile Networks and Applications*, Vol. 1 No. 4, pp. 387-397, January 1997.

[13] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson, "A Quantitative Analysis of Disk Drive Power Management in Portable Computers," *USENIX Conference Proceedings*, Winter 1994.

[14] Brian Marsh and Bruce Zenel, "Power Measurements of Typical Notebook Computers," Matsushita Information Technology Laboratory Technical Report MITL-TR-110-94, May 1994.

[15] Shankar Narayanaswamy et al., "Application and Network Support for InfoPad," *IEEE Personal Communications*, Vol. 3, No. 2, pp. 4-17, April 1996.

[16] Proxim Corp. Spread Spectrum RF Modem, User's Manual.

[17] M. Ranganathan, Anurag Acharua, Shamik D. Sharma and Joel Saltz, "Network-Aware Mobile Programs," *USENIX 1997 Annual Technical Conference*, January 1997.

[18] Michael Richmond and Michael Hitchens, "A New Process Migration Algorithm," *Operating Systems Review*, Vol. 31, No. 1, November 1997.

[19] Chris Ruemmler and John Wilkes. "UNIX Disk Access Patterns," Technical Report HPL-92-152, Computer Systems Laboratory, HP Laboratories, Palo Alto.

[20] John M. Rulnick and Nicholas Bambos. "Mobile Power Management for Maximum Battery Life in Wireless Communications Networks," *Proceedings of IEEE Infocom*, 1996.

[21] Yoichi Shinoda. Wildboar project, http://spider.fortune.co.jp/wildboar.

[22] Mark Stemm, Randy H. Katz. "Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices," *Proceedings of 3rd International Workshop on Mobile Multimedia Communications (MoMuC-3)*, Princeton, NJ, September 1996.

[23] Mark Weiser, Brent Welch, Alan Demers, Scott Shenker, "Scheduling for Reduced CPU Energy," USENIX Association, First Symposium on Operating Systems Design and Implementation, Monterey, CA, November 1994.

[24] John Wilkes, "Predictive power conservation," Tecnical report HPL-CSP-92-5. Concurrent Systems Project, Hewlett-Packard Laboratories, 14 February 1992.

Biographies

Alexey Rudenko is a research assistant in the File Mobility Group at the University of California, Los Angeles. He received his M.S. degree in mathematics and mechanics from Kiev University, Ukraine, in 1984 and the degree of Candidate of Science from Kiev Civil Aviation Institute in 1990. He published a number of papers on mathematical modeling. Before joining UCLA, he worked as a software designer in fields of mathematical modeling, distributed DBMS, and networking. His research interests focus on mobile computing and distributed file systems.

Peter Reiher is an Adjunct Associate Professor of Computer Science at UCLA. Dr. Reiher received his Ph.D. from UCLA in 1987. He has worked on several distributed operating system projects at JPL and UCLA. Dr. Reiher has published 30 professional articles on his research interests, which include mobile computing, distributed operating systems, optimistic and predictive computing, and security for distributed systems.

Gerald J. Popek has been a Professor of Computer Science at UCLA since 1973. His academic background includes a doctorate in computer science from Harvard University. He co-authored "The LOCUS Distributed System Architecture," MIT Press, 1985, and has written more than 70 professional articles concerned with computer security, system software, and computer architecture. Dr. Popek was the founder of Locus Computing Corporation, and is currently also the Chief of Technology Officer for PLATINUM technology, inc.

Geoffrey H. Kuenning received B.S. and M.S. degrees in computer science from Michigan State University, and a Ph.D. in computer science from UCLA. He is currently a researcher in the File Mobility Group at UCLA. Prior to joining UCLA, he worked as a industry consultant in the fields of operating systems, embedded systems, and graphics. Dr. Kuenning has published over 10 papers on various subjects. He is currently performing research in mobile computing, distributed file systems, prediction, and clustering methods. Dr. Kuenning is a member of IEEE, CPSR, ACM, and a number of ACM SIGs.