Power-Aware Network Swapping for Wireless Palmtop PCs

Andrea Acquaviva Emanuele Lattanzi Alessandro Bogliolo

STI - Università di Urbino
61029 Urbino - Italy

Abstract

Virtual memory is considered to be an unlimited resource in desktop or notebook computers with high storage memory capabilities. However, in wireless mobile devices like palmtops and personal digital assistants (PDA), storage memory is limited or absent due to weight, size and power constraints. As a consequence, swapping over remote memory devices can be considered as a viable alternative. Nevertheless, power hungry wireless network interface cards (WNIC) may limit the battery lifetime and application performance if not efficiently exploited. In this work we explore performance and energy of network swapping in comparison with swapping on local micro-drives and flash memories. Our study points out that remote swapping over power-manageable WNICs can be more efficient than local swapping and that both energy and performance can be optimized through power-aware reshaping of data requests. Experimental results show that our optimization technique can save up to 60% of communication energy while improving performance.

1. Introduction

Mass storage devices provide to desktop and laptop computers the support to implement a virtual memory that can be viewed as an unlimited resource to be used to extend the main memory whenever needed. However, in wireless mobile devices like palmtops and personal digital assistants (PDAs), storage memory is limited or absent due to weight, size and power constraints, thus limiting the application of virtual memory. On the other hand, if a wireless network interface card (WNIC) is available, unlimited swapping space could be found on remote devices made available by a server and managed by the operating system as either *network file systems* (NFS) or *network block devices* (NBD). However, swapping over a power hungry WNIC may limit the battery lifetime and application performance if not efficiently exploited.

Network swapping has been object of research in the past decade because of its application in wired networked clusters of computers. In this context, remote swapping has been found to be more efficient than local swapping [7, 5] when high-bandwidth network links are available. In fact, accessing the physical memory of a remote server can be more efficient than accessing a local disk. These results are not directly applicable to wireless palmtop PCs, because of bandwidth limitations and energy constraints.

Remote swapping for handheld computing systems is a recent research topic that has not been extensively studied so far. The problem of energy consumption of network swapping in mobile devices has been faced by Hom and Kremer [6]. They propose a compilation framework aimed at reducing the energy by switching the communication device on and off by means of specific instructions inserted at compile time based on a partial knowledge of the memory footprint of the application. Remote memory resources have been also exploited in advanced file systems for multimedia devices capable of smart storage in remote servers with automatic backup [8].

In this paper we report the results of extensive experiments conducted to evaluate and optimize the performance and power efficiency of different local and remote swap devices for wireless PDAs (namely, a compact flash (CF), a micro drive (HD) and two different WNICs). The contribution of the paper is three-fold. First we characterize all swap devices in terms of time and energy inherently required to swap a single page. Second, we test the effectiveness of the dynamic power management (DPM) support made available by each device. Third, we show that dummy data accesses can be preemptively inserted in the source code to reshape page requests in order to significantly improve the effectiveness of DPM.

Experimental results show that WNICs are less efficient than local devices both in terms of energy and time per page. However, the DPM support provided by WNICs is much more efficient than that of local micro drives, making network swapping less expensive than local swapping for real-world applications with non-uniform page requests. Finally, we show that application-level reshaping of page requests can be used in conjunction with DPM to save up to 60% of energy while improving performance.

2. Swap devices

We refer to the page-based swapping support provided by the Linux OS. Linux performs a page swap in two situations: *i*) when a kernel daemon, activated once per second, finds that the number of free pages has fallen below a given threshold; *ii*) when a memory request cannot be satisfied. The page to be swapped-out is selected in a global way, independently from the process that made the request. The page replacement algorithm is based on an approximation of least recently used (LRU) policy [3].

Modern operating systems equipping palmtops and PDAs make possible to define heterogeneous support for swapping. Swapping can be performed both locally to the PDA and remotely, by exploiting server storage capabilities and network connections. More than one swap units can be enabled at the same time, with assigned priority. The unit with the highest priority is selected by default until it becomes insufficient.

2.1. Local devices

On-board non-volatile memory is usually available in palmtop PCs to store the bootloader and the filesystem. Magnetic disks can be added to extend file storage capabilities. Swap can be made locally in palmtops as in desktop PCs. A dedicated partition can be defined in hard drives or flash memories, where the filesystem resides. Alternatively, some OSes allow the user to define a swap file that does not need a dedicated partition. Either way, the swap area comes at the price of decreasing the space available for actual storage purposes.

Compact Flash.
Palmtop PCs are equipped with on-board flash memories, but additional memory chips can be installed as an expansion if an external slot is present. Memory Technology Device (MTD) drivers allow to define swap partitions or swap files on flash memories. However, being read-most devices, flash memories are not the ideal support for swapping. Nevertheless, we evaluate their swapping performance since they are always present in palmtop PCs, being sometimes the only alternative to network swapping.

Hard Disk.
Today's technology make available hand-sized magnetic disks (called *mini* of *micro drives*) suitable to be installed in palmtop computers. Currently they provide a storage capability up to 5GBytes. Like traditional hard disks (HD), micro drives provide a seek time much longer than the access time to sequential blocks. For this reason, access to these kinds of devices is usually performed in bursts whenever possible by exploiting on-board hardware buffers in order to compensate for the initial transfer delay. The OS tries to limit the delays by filtering disk accesses using software caches, whose size is limited by the available space in main memory. When a micro drive uses as a swap device, this trade-off is even more critical, since increasing the memory space allocated for caching increases the number of swap requests.

2.2. Network Devices

In order to provide the performance required to fully exploit the channel bandwidth, remote swap files can be mapped in the main memory of a remote server. This is the choice we made for our experiments.

Network File System.
NFS (*Network File System*) is used in a network to enable file sharing among different machines on a local area. The communication protocol is based on a UDP stack, while data transfers between NFS server and clients are based on Remote Procedure Calls (RPCs). The idea of using NFS to support network swapping is relatively recent [10]. To this purpose, a remote file must be configured as a swap area. This is made possible by modern operating systems that allow the user to specify either a device or a file as a swap unit.

Network Block Device.
A *Network Block Device* (NBD) [4] offers to the OS and to the applications running on top of it the illusion of using a local block device, while data are not stored locally but sent to a remote server. As in case of NFS, the virtual local device is mapped in a remote file, but the swap unit is viewed as a device, rather than as a file.

This is made possible by a kernel level driver (or module) that communicates to a remote user-level server. The first time the network connection is set-up, a NDB user-level client negotiates with the NBD server the size and the access granularity of the exported file. After initialization, the user-level NBD client does not take part to remaining transactions that directly involve the kernel NBD driver and the NBD server. No RPCs are required in this case, thus reducing the software overhead. Differently from NFS, the underlying network stack is TCP instead of UDP. This increases the reliability of network transfers, at the cost of increasing the protocol overhead.

3. Experimental Setup

We performed our experiments on a HP's IPAQ 3600 handheld PC, equipped with a Strong-ARM-1110 processor, 32MB SDRAM and 16MB of FLASH. Our benchmarks were executed on the palmtop on top of the Linux operating system, Familiar release 6.0. The WNICs used to provide network connectivity were a COMPAQ WL110 [11] (hereafter denoted by NIC_{COMPAQ}) and a CISCO AIRONET

```
/*************** Benchmark 1 ***************/
double A[ROW][COL];
initialize(A,ROW,COL);
t0 = time();
read_by_column(A,ROW,COL);
t1 = time();
/*******************************************/
```

Figure 1. Pseudo-code of the benchmark used to characterize swap devices.

350 [13] (NIC$_{CISCO}$), while the AP connected to the remote swapping server was a CISCO 350 Series base station [12]. The remote server was installed on a Athlon 4 Mobile 1.2 GHz notebook. For local swap experiments we used a 340 MB IBM Microdrive (HD) and a 64 MB Compaq-Sundisk Compact Flash Memory (CF) [14, 15]. Power consumption of both WNICs and local devices was measured using a *Sycard Card Extender* that allowed us to monitor the time behavior of the supply current drawn by the card. The current waveforms were then digitized using a *National Instruments Data Acquisition Board* connected to a PC. A *Labview* software running on the PC was used to coordinate the acquisition and bufferize current samples to compute power and energy consumption.

The remote swap NBD server was instrumented in order to collect time-stamped traces of swapping activity during benchmarks execution.

4. Characterization of Swapping Costs

To characterize the inherent cost of a page swap we developed a suite of benchmarks accessing data structures much larger than the available main memory, without performing any computation on them. This kind of benchmark is suitable to characterize swapping cost since the computation time is negligible with respect of the time spent in swapping and the devices under characterization are always busy serving page requests. The pseudo-code of the benchmark is shown in Figure 1. A large matrix is allocated and initialized and then read by column in order to maximize the number of page faults. Different benchmarks were generated by changing the number of columns and rows of the matrix in order to change the number of page faults while keeping the total size of the matrix unchanged. This allowed us to cross-validate experimental results and reduce characterization errors.

A second set of benchmarks was obtained by replacing the `read_by_column` procedure with a `write_by_column` procedure, and used to characterize the swapping cost in case of write-back.

Characterization results.
Experimental results are reported in Table 1 in terms of time, energy and power required by each local and remote device to swap a page of 4096 bytes. Both *read-only* and *write-back* results are reported.

In general, write-back doubles the cost (in energy and time) of a read-only swap, since it involves two data transfers. As expected, local devices are more efficient than WNICs and CF has an energy-per-page more than 10 times lower than all other devices.

It is also worth noting that, for a given WNIC, NBD provides greater performance than NFS, at the cost of slightly higher power consumption. Since the time reduction overcomes the additional power consumption, the energy per page required by NBD is lower than that required by NFS.

5. Power Optimization

In the previous section we have characterized swap devices in terms of time and energy requirements per swap page. To this purpose we designed a set of benchmarks that simply accessed data structures much larger than the main memory without performing any computation on them.

Although useful for characterization purposes, the benchmarks of Figure 1 are unrealistic for two main reasons. First, computation time is usually non-negligible, so that page requests are spaced in time according to a distribution that depends on the workload and on the state of the main memory. Second, the total size of the data structures accessed by each application usually does not exceed the size of the main memory, otherwise the performance degradation would not be acceptable.

In most cases of practical interest, swapping is mainly needed after a context switch to bring in main memory data structures the first time they are used by the active process. Moreover, in handheld devices there are often only a few processes running concurrently, so that both main memory and peripherals are mainly used by a single process at a time. In this situation, the usage pattern of swapping devices are significantly different from those used for characterizing swapping costs because of the presence of long idle periods between page swaps.

Since swapping devices spend power while waiting for page requests, the effective energy per page is larger than that reported in Table 1. On the other hand, idleness can be dynamically exploited to save power by putting the devices in low-power operating modes, or by turning them off. Dynamic power management (DPM) significantly impacts the performance and energy trade-off offered by each device under bursty workloads.

In this section, we first analyze the DPM supports provided by each swapping device, then we show how to increase their effectiveness by means of software optimiza-

Swap device		Read-only			Write-back		
Type	Mode	Time [ms]	Energy [mJ]	Power [mW]	Time [ms]	Energy [mJ]	Power [mW]
CF	local	4.1	0.201	49	8.2	0.402	49
HD	local	3.0	1.911	637	6.4	4.061	637
NIC$_{CISCO}$	NBD	7.0	5.934	848	14.0	10.319	735
NIC$_{CISCO}$	NFS	8.5	6.123	720	14.6	10.516	720
NIC$_{COMPAQ}$	NBD	8.0	5.626	578	15.0	8.599	573
NIC$_{COMPAQ}$	NFS	10.0	5.243	524	22.0	10.672	485

Table 1. Power consumption and performance of local and remote swap devices.

tion techniques aimed at reshaping the distribution of page requests.

5.1. Dynamic Power Management

Device	State	Power [mW]	Timeout [ms]	WU-time [ms]	WU-pow [mW]
CF	Read	107			
	Write	156			
	Wait	4.5			
HD	Read	946			
	Write	991			
	Wait	600			
	Sleep	24	2s	$4.5s \pm 1.98s$	1s
NIC CISCO	RX	755			
	TX	1136			
	Wait	525			
	Doze	113	15	14	400
	Off	0	any	370	451
NIC COMP	RX	548			
	TX	798			
	Wait	407			
	Doze	38	100	1	800
	Off	0	any	270	357

Table 2. Power states of swapping devices.

The DPM support provided by each swap device is schematically represented in Table 2. For each device, the key features of active and inactive operating modes are reported. Active modes are characterized only in terms of power consumption, while inactive modes are also characterized in terms of *timeout* to be waited before entering the inactive state, *wake-up time* and *wake-up power*. The data reported in the Table have been obtained by analyzing the current profiles provided by the measurement setup described in Section 3.

First of all we remark that the average power consumptions measured during page swaps (reported in Table 1) are not equal to the power consumptions measured for the devices during read/receive or write/transmit. In fact, for instance, a page swap across a wireless link entails the transmission of the page request, a waiting time corresponding to the latency of the remote device, the reception of the page and, possibly, the write-back of a swapped-out page. The average swapping power comes from the weighted average of all these contributions.

The CF has no inactive states. This is because its power consumption in wait mode is negligible, making inactive low-power states not necessary. On the contrary, NICs and HDs consume a large amount of power while waiting for service requests, so that it is worth switching them to low-power inactive states during long idle periods.

The sleep state of the HD has the lowest power consumption, but the highest wakeup cost in terms of power (higher than 1W) and time (in the order of several seconds). Moreover, the wakeup time is highly unpredictable, its measured standard deviation being almost 2 seconds.

According to the IEEE802.11b standard [1], WNICs provide MAC-level DPM support that can be enabled via software. The actual implementation of the DPM support depends on the WNIC. The protocol policy (PSP) consists in placing the card in a low-power state called *doze mode*, in which it sleeps but wakes-up periodically to keep synchronized with the network and to check the access point (AP) for outstanding data. A polling frame must be transmitted by the card for each packet to be retrieved. PSP mode provides power savings at a cost of a noticeable performance hit. To increase performance, a variation of this policy is implemented by CISCO cards. They automatically switch from PSP to CAM (*Constant Awake Mode*) when a large amount of traffic is detected. In this case no polling frame is needed between packets since the reception and transmission happen in active mode.

Even if the power consumption in sleep state is low, it is not negligible. Moreover, the card is sensitive to broadcast traffic. A more aggressive policy would require to completely shut-off the card when no needed by any active application in the system. Thus, more power can be saved, at the price of a larger wake-up delay needed by network reassociation. OS-level policies can be implemented to this purpose based on a power management infrastructure recently developed for Linux OS [2]. This infrastructure is composed by a power manager module that handles requests from applications and keeps track of their resource needs. On the other side, upon a request, the power manager can directly switch off a peripheral (WNIC in our case) if no other applications are using it. Switch off request may come from user applications through dedicated APIs or directly by another kernel module. We exploited this feature to let the NBD driver module switch on and off the card between swapping requests.

The features of doze and power-off modes are reported for both WNICs in Table 2. We observe that the MAC-level DPM support of NIC$_{COMPAQ}$ is more efficient that that of NIC$_{CISCO}$, but the DPM policy if more conservative (the timeout being 100ms).

—

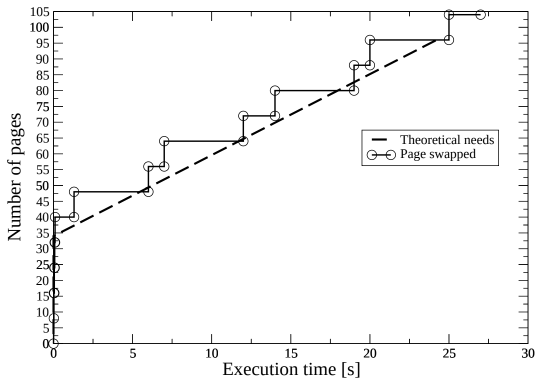

Figure 2. Distribution of page requests.

5.2. Re-shaping Swap Requests

The effectiveness of any DPM strategy strongly depends on workload statistics. Regardless of the DPM policy, the higher the burstiness of the workload, the higher the power savings. In fact, long idle periods can be effectively exploited to switch to the deepest inactive states, while long activity bursts amortize the cost of wake up.

Although caching and buffering can be performed by the OS to perform a low-level reshaping of page requests, a typical trace of swapping traffic shows small bursts of a few pages followed by short periods of inactivity. Increasing the granularity of page swaps could increase the burstiness of the workload, but also increases the risk of preemptively swapping-in unused pages.

On the other hand, in many cases data pre-fetching could be deterministically performed at the application-level in order to reshape swapping traffic. This can be done by inserting dummy accesses to the data structures right before they are used. Dummy accesses generate bursts of page requests for two main reasons: first, they are not delayed by any computation; second, a single access is sufficient to fetch an entire page.

5.3. Case Study

We use matrix multiplication as a case study to evaluate the effectiveness of the DPM strategies implemented by the swap devices and to demonstrate the feasibility of application-level reshaping of swap requests.

```
/*************** Benchmark 2 ***************/
double dummy[2048][2048], C[128][128];
double A[128][128], B[128][128];
initialize(A,128,128);
initialize(B,128,128);
initialize(C,128,128);
initialize(dummy,2048,2048); //swap out
t0 = time();
compute_product(A,B,C);
t1 = time();
/*******************************************/
```

Figure 3. Pseudo-code of the case study.

The pseudo-code of the case study is reported in Figure 3: it simply computes the product of two square matrices A and B and puts the result if a third matrix C. The total size of the three matrices fits in main memory, but we use a dummy matrix, exceeding the size of the physical memory, to force swapping activity. Matrices A, B and C are first allocated and initialized, then the dummy matrix is initialized in order to swap A, B and C out from main memory. In practice, the initialization of the dummy matrix creates boundary conditions similar to those possibly caused by the execution of other applications. Then we monitor the execution time and the swapping energy caused by the execution of the `compute_product` procedure.

The distribution of swap requests is shown in Figure 2. The expected distribution is also plotted for comparison. The large number of pages requested at the beginning corresponds to the upload of the entire matrix B. In fact, the first column of B has to be read in order to compute the first entry of C. Since matrices are stored in memory by rows, reading the first column entails swapping in the entire matrix. Subsequent page requests are spaced in time according to the time required to compute 512×128 floating point products.

Comparing the actual requests with the theoretical needs we observe that the OS swaps 8 pages at a time, thus increasing the opportunity for DPM. However, the total number of pages request by the OS is 104, while the three matrices fit into 96 pages.

To reshape swap requests we inserted dummy accesses to the three matrices between the computation of initial time t0 and the computation of the matrix product. Dummy accesses were performed by a routing, called `access_one_per_page`, that reads one matrix entry every 512 (i.e., one entry per page).

5.4. Experimental results and conclusions

Experimental results obtained by executing the case study with and without traffic reshaping are reported in Ta-

Device	Exec. time [s]					Energy [mJ]				
	Original		Reshaped		Ratio	Original		Reshaped		Ratio
	Avg.	St.Dev.	Avg.	St.Dev.		Avg.	St.Dev.	Avg.	St.Dev.	
RAM	25	0	25.25	0.5	1.01	-	-	-	-	-
CF	25.5	0.57	25.75	0.5	1.01	0.143	0.003	0.161	0.025	1.12
HD	*25.312*	-	*25.812*	-	*1.01*	*15.199*	-	*15.499*	-	*1.01*
(sleep 2000)	**37.75**	5.91	**27.75**	0.957	**0.73**	**19.426**	5.311	**6.207**	0.846	**0.32**
NIC$_{Cisco}$	26.5	0.58	27	0.82	1.02	15.591	0.32	15.533	1.48	0.996
(doze 15)	**27.5**	0.58	**26.5**	0.58	**0.96**	**7.303**	0.26	**4.692**	0.13	**0.64**
(off 100)	**28.75**	0.5	**26.0**	0.82	**0.93**	**2.473**	0.09	**0.890**	0.051	**0.36**
NIC$_{Compaq}$	30.25	0.5	28.5	1.0	0.94	13.597	0.56	12.701	0.51	0.93
(doze 100)	**30.0**	0	**27.75**	0.5	**0.92**	**2.542**	0.096	**2.187**	0.083	**0.86**
(off 100)	**30.0**	0	**28.25**	0.5	**0.94**	**1.760**	0.08	**0.722**	0.045	**0.41**

Table 3. Execution time and swapping energy required to run the case study of Figure 3.

ble 3. Based on the results reported in Table 1 we decided to use NBD for remote swapping.

Each device was tested with and without DPM. The two WNICs were tested with both MAC-level DPM (doze) and OS-level PDM (power-off). The DPM mode and the corresponding timeouts are reported in the first column. The performance of the CF and the CPU time obtained by running the application with data available in main memory are also reported in the first two rows for reference. The DPM of the HD was enabled by default, so that data reported on row HD are computed from previous characterization. All other data were obtained from real measurements, by repeating each experiment 4 times.

Interestingly, even without DPM, the HD consumes more energy than WNICs. This is because of its higher power consumption when idle. When DPM is enabled, WNICs become much more convenient than HD. In particular, the DPM of the HD is counterproductive both in terms of time and energy under this traffic conditions because of the large wakeup cost. On the contrary, MAC-level DPM of NIC$_{Cisco}$ and NIC$_{Compaq}$ saves respectively more than 50% and more than 80% of the swapping energy. If the power-off state is exploited, power savings become of 85% and 94%, respectively, with negligible performance loss.

When DPM policies are enabled, traffic reshaping provides further advantages both in terms of energy and execution time. For the HD, traffic reshaping makes DPM effective to save more than 60% of energy consumption. For WNICs, traffic reshaping provides additional energy savings while further reducing the performance loss. The ratios between results obtained with and without traffic reshaping are reported in the table. In particular, when the power-off state is exploited, traffic reshaping leads to additional energy savings around 60%. It is also worth noting that reshaping provides only a marginal benefit when combined to the MAC-level DPM of NIC$_{Compaq}$, because of the highly efficient DPM support provided by that card.

The overall effect of DPM and traffic reshaping makes network swapping much more energy efficient than local swapping on a HD (the overall energy being almost 10 times lower) with a performance penalty of about 5%.

In conclusions, our experiments demonstrate the feasibility and the energy efficiency of network swapping from wireless palmtop PCs. The effectiveness of the DPM support provided by WNICs compensate makes them more efficient than local HDs and open the field to optimization strategies (like swap reshaping) that may further improve energy efficiency and performance.

References

[1] LAN/MAN Standards Committee of the IEEE Computer Society. *Part 11: Wireless LAN MAC and PHY Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, IEEE, 1999.

[2] A. Acquaviva, T. Simunic, V. Deolalikar, S. Roy, ”Remote Power Control of Wireless Network Interfaces,” *Proceedings of PATMOS*, Turin, Italy, Sept. 2003.

[3] D. Bovet, M. Cesati, ”Understanding the Linux Kernel,” *OŔeally & Associates*, Sebastopol, CA, Jan. 2001.

[4] P. T. Breuer, A. Marin Lopez, A. Garcia Ares, ”The Network Block Device,” *Linux Journal*, Issue 73, May 2000.

[5] M. D. Flouris, E. P. Markatos, ”The Network RamDisk: Using Remote Memory on Heterogeneous NOWs,” *Cluster Computing*, pp. 281-293, 1999, Baltzer Science Publishers.

[6] J. Hom, U. Kremer, ”Energy Management of Virtual Memory on Diskless Devices,” *Proceedings of COLP*, Barcelona, Spain, Sept. 2001.

[7] T. Newhall, S. Finney, K. Ganchev, M. Spiegel, ”Nswap: A Network Swapping Module for Linux Clusters,” *Proceedings of Euro-Par*, Klagenfurt, Austria, August 2003.

[8] M. Satyanarayanan, ”The Evolution of Coda,” *ACM TOCS*, Vol. 20, Issue 2, Pages: 85–124, May 2002.

[9] A. Silbershatz, P. Galvin, G. Gagne, ”Operating System Concepts, 6th Edition,” *Addison-Wesley*, 2002.

[10] ”Swapping via NFS for Linux,” http://www.nfs-swap.dot-heine.de

[11] HP, *WL110*, http://h18004.www1.hp.com/products/wireless/wlan/wl110.html, 2003.

[12] Cisco System, *Cisco Aironet 350 Series Access Points*, http://www.cisco.com/univercd/cc/td/doc/product/wireless/airo_350/accsspts/index.htm, 2003.

[13] Cisco System, *Cisco Aironet 350 Series Wireless LAN Adapters*, http://www.cisco.com/univercd/cc/td/doc/product/wireless/airo_350/350cards/index.htm, 2003.

[14] IBM, *340MB Microdrive Hard Drive*, http://www.storage.ibm.com/hddredirect.html?/micro/index.html, 2003.

[15] Compaq, *compact flash cards*, http://www.hp.com/products1/storage/products/storagemedia/flash_cards/index.html, 2003.