The `nswap` module for Network Swap

1 Authors

The `nswap` module was and continues to be developed by Professor Tia Newhall, Sean Finney, Kuzman Ganchev, Matti Klock and Michael Spiegel as an undergraduate research project. The students presenting the poster will be Kuzman Ganchev and Sean Finney:

Name	School	e-mail
Sean Finney	Swarthmore College	`finney@cs.swarthmore.edu`
Kuzman Ganchev	Swarthmore College	`kuzman@sccs.swarthmore.edu`
Matti Klock	Swarthmore College	`klock@cs.swarthmore.edu`
Tia Newhall	Swarthmore College	`newhall@cs.swarthmore.edu`
Michael Spiegel	Swarthmore College	`spiegel@cs.swarthmore.edu`

2 Sponsor

This poster is sponsored by Professor Tia Newhall of Swarthmore College, who may be contacted by electronic mail at `newhall@cs.swarthmore.edu` or by phone at (610) 690 5637.

3 Abstract

Swapping between nodes in a cluster of workstations to improve system performance has been discussed by several research papers in the recent past [2] [3]. Studies of workstation usage patterns show that most of the resources in a cluster of workstation are free most of the time even at peak workloads [1]. As a result, there will almost always be idle remote memory for an overloaded node to use as swap space. In addition, faster networks and slowly increasing disk speeds make swapping to remote memory an increasingly attractive option. We present a network swapping system called "`nswap`" implemented as a loadable kernel module for for the Linux kernel. `nswap` features a rapid response time even in the case of out of date information about remote hosts, a load balancing scheme through page migrations, and smooth fail-over to disk in case network swap space becomes scarce. The `nswap` module, conceptually divided into a pseudo-device driver client and a memory server, achieves these goal using a set of protocols described in the poster.

When the `nswap` module is used as a swap device, the kernel will write a page to it. The client receives the write request and initiates a `PUTPAGE` to a server. At some later time the kernel will require the data on the page and issue a read request to the device. The client will then use the `GETPAGE` protocol to retrieve the page. Eventually the client will use the `INVALIDATE` protocol to inform the server that the page will no longer be needed and its memory will be reused. When a server node becomes overloaded, or needs to reclaim some of its memory for its own paging system, it can migrate some of the pages it is caching to other nswap server nodes. Page migration is implemented using the `PUNTPAGE` protocol, which involves the use of `UPDATE` to alert the client to the new location of the page and `INVALIDATE` to inform the server to drop the page.

4 References

References

[1] Anurag Acharya and Sanjeev Setia. Availability and utility of idle memory on workstation clusters. In *ACM SIGMETRICS Conference on Measuring and Modeling of Computer Systems*, pages 35–46, May 1999.

[2] Michael J. Feeley et al. Implementing global memory management in a workstation cluster. In *15th ACM Symposium on Operating Systems Principles*, December 1995.

[3] Evangelos P. Markatos and George Dramitinos. Implementation ofa reliable remote memory pager. In *USENIX 1996 Annual Technical Conference*, 1996.