One-pager 3

Martin Vassor

1 Memory design

NVM seems suitable for long-lasting large-size read-only data, while DRAM seems suitable for heavily modified short-lifetime data.

The idea proposed is to use NVM as *read oriented* main memory and as an intermediate paging level (Figure 1). Since the NVM is 20× denser, splitting the original main memory capacity C into C_{NVM} and C_{DRAM} with $C = C_{\text{NVM}} + C_{\text{DRAM}}$ provides a swap extension of $19 \cdot C_{\text{NVM}}$. NVM as swap is suitable in term of both energy (Subsection 1.1) and latency (e.g. nanosecond versus millisecond).

The proportion of NVM and DRAM are chosen according to the relative sizes of the *read-oriented* and *write-oriented* working sets, which are determined according to the following subsections.

1.1 Energy

Let K_k, K_r and K_w be the energy cost of keeping a unit of data for a unit of time, reading and writing a unit of data in DRAM. Let N_r, N_w and N_l be the number of reads, writes and loads from disk. Let $||d||$ be the size of the data, t its lifetime. The overall energy cost in DRAM and NVM are

$$E_{\text{DRAM}} = ||d|| \cdot (t \cdot K_k + N_r \cdot K_r + (N_w + N_l) \cdot K_w)$$

$$E_{\text{NVM}} = ||d|| \cdot (N_r \cdot K_r + (N_w + N_l) \cdot K_w \cdot 50) \quad .$$

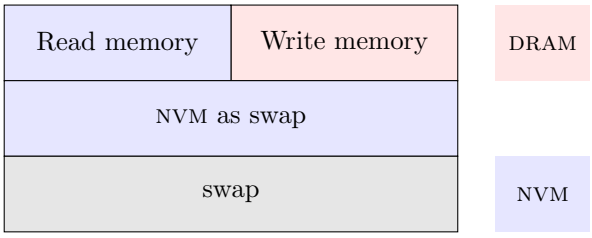

Figure 1: Memory organisation

Hence,

$$\begin{aligned} & E_{\text{NVM}} < E_{\text{DRAM}} \\ \Leftrightarrow & (N_w + N_l) \cdot K_w \cdot 50 < t \cdot K_k + (N_w + N_l) \cdot K_w \\ \Leftrightarrow & 50 < \frac{t \cdot K_k}{(N_w + N_l) \cdot K_w} + 1 \end{aligned}$$

Hence, NVM is energy efficient for long-lasting (swap extension) or *read-oriented* data.

1.2 Latency

With N_r, N_w and N_l as above, L_w and L_r the latencies of a single write and read in DRAM. The DRAM and NVM overall latencies are

$$L_{\text{DRAM}} = N_r \cdot L_r + (N_w + N_l) \cdot L_w$$

$$L_{\text{NVM}} = N_r \cdot L_r + (N_w + N_l) \cdot L_w \cdot 12 \quad .$$

Hence, if NVM is used by read-oriented data, it does not suffer from the writing latency penalty. The loading latency is certainly worse than DRAM, but the disk latency bottlenecks in both cases.

2 Software design

In virtual memory, the software is responsible of handling page placement between two locations. Our goal here is to spread the data between three locations, as the memory is now divided into two parts.

2.1 Semantic separation

Usual programs are divided into segments. When loading a program, the operating system can safely place read-only data (such as the `.text`, the `.data` segments) in NVM memory and read-write data (such as the `.bss` segment, the `stack`, etc.) in DRAM memory.

Similarly, dynamically allocated memory can be placed depending on the right accesses. If some data is `mprotect` with write access, the memory page should be put in DRAM and vice-versa.

This separation is *semantic* as it is based on the expected behaviour of the data based on some *hints*.

2.2 Dynamic moves

As there is currently a dirty bit in the MMU to measure accesses, one can imagine to split it into two counters, namely a *read access counter* and a *write access counter*. Hence, it would be easy to adapt the current page-placement algorithm to dynamically guess the best place for each page, and eventually move them during the execution.

3 Conclusion

Replacing some DRAM by NVM serves two purposes:

1. Reducing the cost of read-oriented data, handled by semantic hints.
2. Providing an intermediate level for virtual memory, due to NVM superior capacity, handled by extended MMU.

Word count: 269 + 193 + 38 = 500, including footnotes, titles, caption, 1 word per equation, excluding references