Using Flash SSDs as Main Memory Extension with a Locality-aware Algorithm

Hiroko Midorikawa

midori@st.seikei.ac.jp,
Seikei University, JST-CREST, Tokyo, Japan

I. Introduction

Most scientific computation often require a lager memory to tackle big-sized problems and/or for higher resolution data analysis. One of the common solutions is aggregating the distributed memories over cluster nodes. Typically, it is accomplished by increasing the amount of DRAM per node and the number of nodes in one cluster. However, there is a limit to the extent to which DRAM can be increased in main memory, because there is a limited number of memory slots on server boards, limited power consumption, and other resource limitations.

The advent of various kinds of NVM brings us a new era in memory organizations and memory-related software [1]. It influences not only the traditional memory hierarchy but also the basic idea of memory read/write and file IO. It has a potential to change traditional programming models and application programs drastically. Of various NVMs, flash memory is already widely available to end-users. Its access time is not as short as that of DRAM, but it provides a much greater capacity at a lower cost and with less power consumption. With regard to these points, flash memory is one of the candidates as a DRAM extension to the main memory[2].

Recent PCIe bus-connected flash SSDs [3][4] achieve several hundred times faster latency than HDDs, but it is about one thousand time slower than DRAM, as shown in Fig. 1. The gap between DRAM and flash SSD is much bigger than the gap between L3 cache and memory, which is only 3 to 10 times difference in latency. Thus, the large latency gap between DRAM and flash SSD makes it difficult to use the latter as a main memory extension for applications.

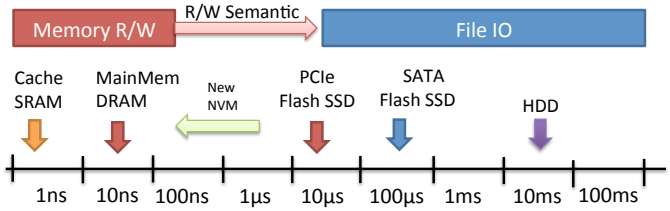

Fig. 1 Access latency in various memory devices

This paper investigates the potential of PCIe flash as large and slow memory behind DRAM for a stencil computation, which is one of the most typical and important computation kernels in various scientific and engineering simulations. A locality-aware, hierarchical out-of-core computation algorithm using data structure blocking techniques is newly applied to the stencil computation to bridge the DRAM-Flash latency divide. Our novel application of hierarchical temporal blocking optimization on stencil computation with a flash SSD performs satisfactorily for practical use. We find that 7-point stencil computations for a 512GiB problem (16 times bigger size than that of the DRAM) using only a 32 GiB of DRAM and a flash SSD, in Mflops attain 87% of the performance achieved in execution using only DRAM.

II. The Locality-aware algorithm for using Flash SSD as Main Memory Extension

Stencil computation is one of the most popular and important types of processing in various scientific and engineering simulations. It performs iterative calculations on a limited data set, typically the nearest neighbor data. It sweeps the entire data – e.g., three-dimensional (3D) physical data space – and updates them at each time step. Fig. 2 shows a typical stencil computation for a 3D data grid – a 7-point stencil computation using the six nearest neighbor points and a 19-point stencil computation using the 18 nearest neighbor points.

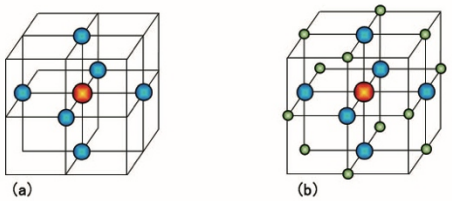

Fig.2 7-point and 19-point stencil computations on 3D data

Temporal blocking algorithm, which extracts not only spatial locality but also temporal locality for iterative applications, is newly applied to flash and DRAM memory layers for stencil computations [6]. Typical temporal blocking algorithm for 3-dimensional data domain is shown in Fig. 6. We introduce three-layered data structures corresponding to Flash, DRAM, L3-cache memory hierarchy in the algorithm to extract memory access locality for each layer in memory hierarchy, as shown in Fig.3.

There are several options in the use of flash SSD as memory extension from applications [5]. Three methods are evaluated in this experiment, (1) swap method, (2) mmap method and (3) aio method, shown in Fig. 3. In the swap method, memory allocation, malloc() function is used in applications and a flash SSD is used as a swap device under the virtual memory system of the OS. In the mmap method, a file memory map *mmap()*

function is used in applications and a flash SSD is used as a file system. In the aio method, Linux kernel asynchronous Input/Output library functions are used in applications and a flash SSD is used as a block device.

Fig. 4 shows relative execution times of a 7-point stencil 64GiB-problem for each method using limited DRAM, 32GiB, and sufficient DRAM, 128GiB. aio method with temporal blocking is most effective for the execution under limited DRAM. With aio method, computation time for 64GiB problem using only 32GiB of DRAM and Flash takes only 1.5 time larger than that of normal execution using sufficient DRAM, 128GiB. Without temporal blocking algorithm, its computation time using 32GiB DRAM takes 65.2 time larger than the normal execution as shown in Fig. 4. Fig. 5 shows relative effective Mflops of various-size problems using 32GiB. In the 1TiB problem execution using only 32GiB DRAM achieves 87% performance of normal execution using sufficient DRAM.

We also optimized the algorithm for NUMA systems and found that it achieves sufficient performance in stencil computations for using flash SSD as a main memory extension.

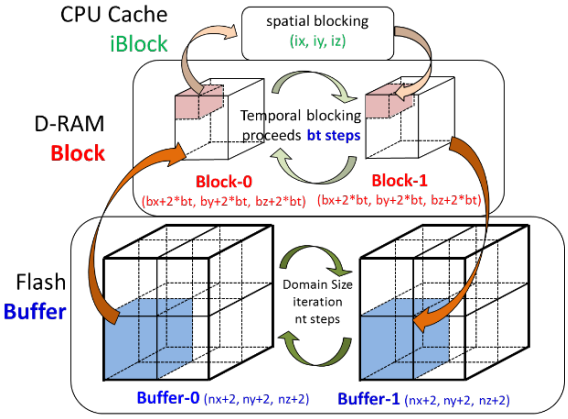

Fig.3 Three-layered data structure for locality extraction

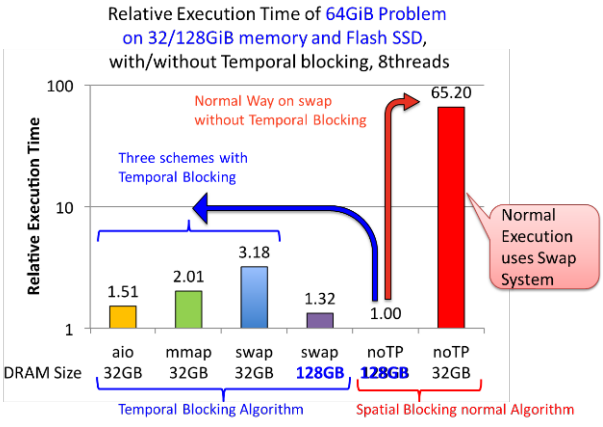

Fig.4 Relative times for various methods

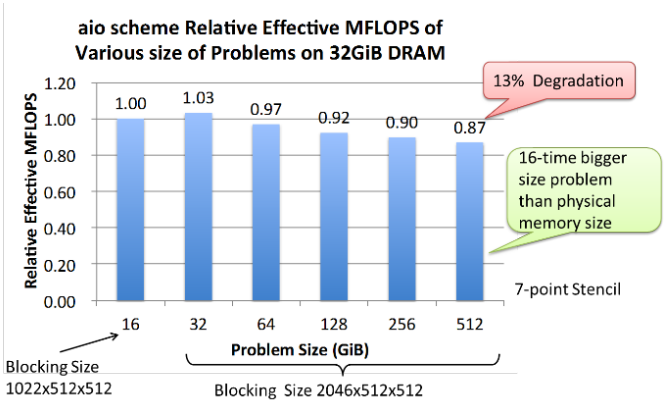

Fig. 5 Performance in various-size of problems

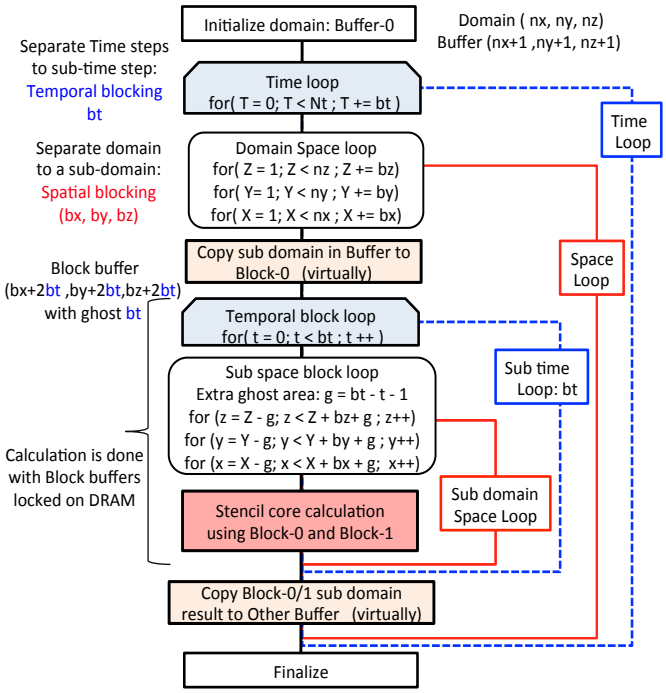

Fig.6 1-level temporal blocking algorithm: pseudo codes for a 3D domain.

REFERENCES

[1] Anirudh Badam, "How Persistent Memory Will Change Software Systems", IEEE Computer ,pp45-51, Aug. 2013

[2] Kshitij Sudan, Anirudh Badam, Dvid Nellans, "NAND-Flash: Fast Storage or Slow Memory?", NVM Workshop 2012

[3] ioDrive2 (FusionIO) https://www.fusionio.com/products/iodrive2/

[4] Intel SSD910 http://www.intel.com/content/www/us/en/solid-state-drives/solid-state-drives-910-series.html

[5] Hiroko Midorikawa, "Using a Flash as Large and Slow Memory for Stencil Computations", Flash Memory Summit 2014, (2014.8) http://www.ci.seikei.ac.jp/midori/paper/20140807_301D_Midorikawa.pdf

[6] Hiroko Midorikawa, Hideyuki Tan and Toshio Endo:"An Evaluation of the Potential of Flash SSD as Large and Slow Memory for Stencil Computations", Proceedings of the 2014 International Conference on High Performance Computing and Simulation (IEEE HPCS2014) (ISBN 978-1-4799-5311-0), pp.268-277, 2014-7 IEEE-HPCS2014