# An Evaluation of Flash SSDs as Main Memory Extension for Stencil Computation

JST CREST, Seikei University
Hiroko Midorikwa

JST CREST International Symposium
on Post Peta-scale System Software
Dec.2-4, 2014, Kobe

# Background

High Performance Computations always require

- Larger-size problem solving
- Higher resolution analysis

more memory, DRAM..

Traditional way to solve it :

- increase the amount of DRAM per computing node
- increase the number of computing nodes

Limitation of power consumption

# New Non Volatile Memory

Lower power consumption, Capacity, Speed, Cost, …

NON-Volatile DATA

Traditional Framework

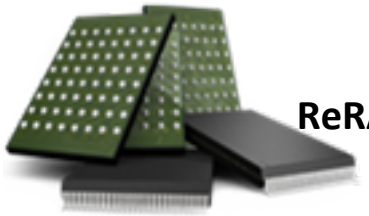
"Volatile main memory and

Non-volatile second storage"

NVMs give drastic changes

Not only in hardware, but also in

- OS managements in memory, IO subsystem and file system
- Programming model based on Memory R/W and FILE-IO

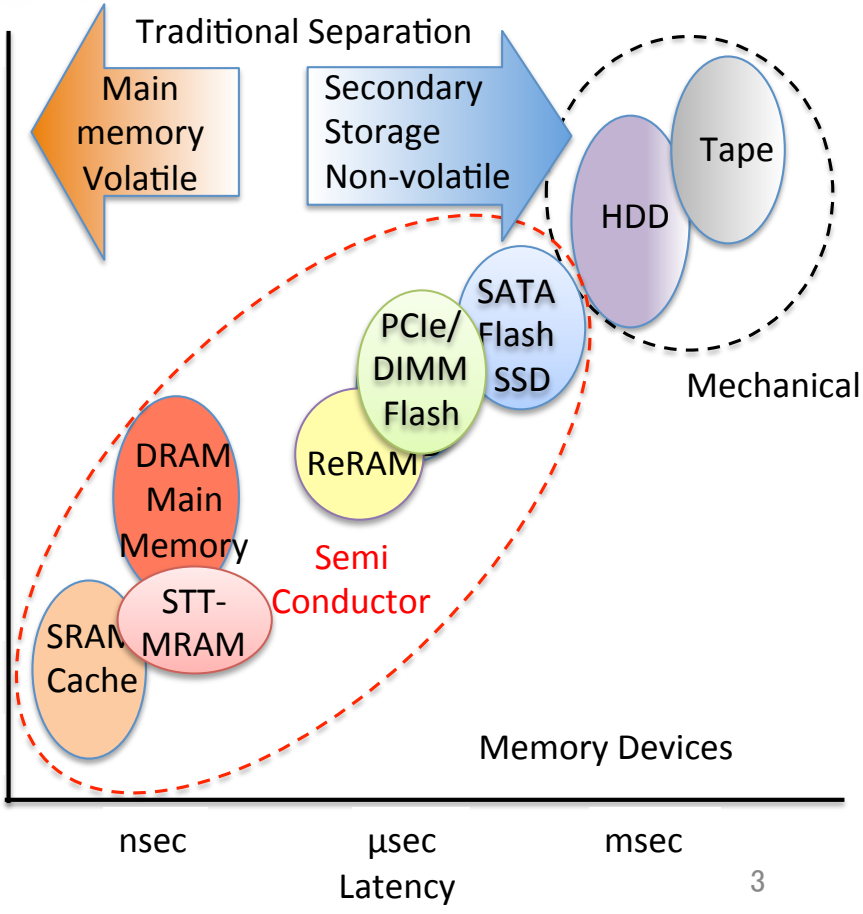


STT-MRAM

ReRAM

PCM-RAM

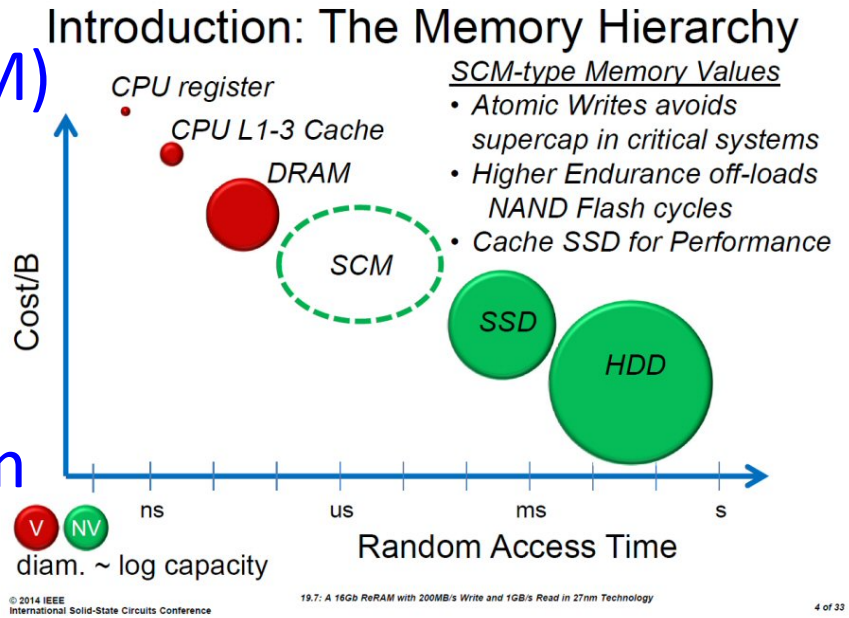
Flash DIMM



# A New Era: in which the worlds of storage and memory are colliding

## Storage Class Memory (SCM)（IBM)

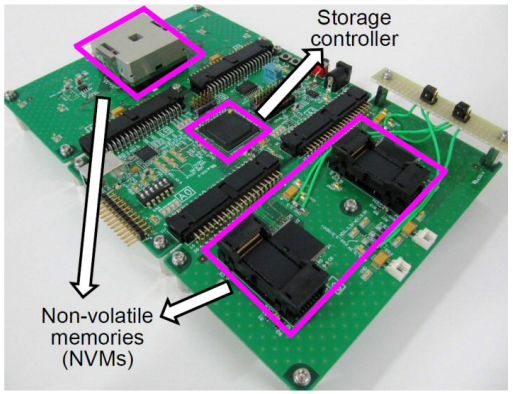
- M-type SCM, may close to memory
- S-type SCM, may close to storage




## File Storage device: Hybrid system

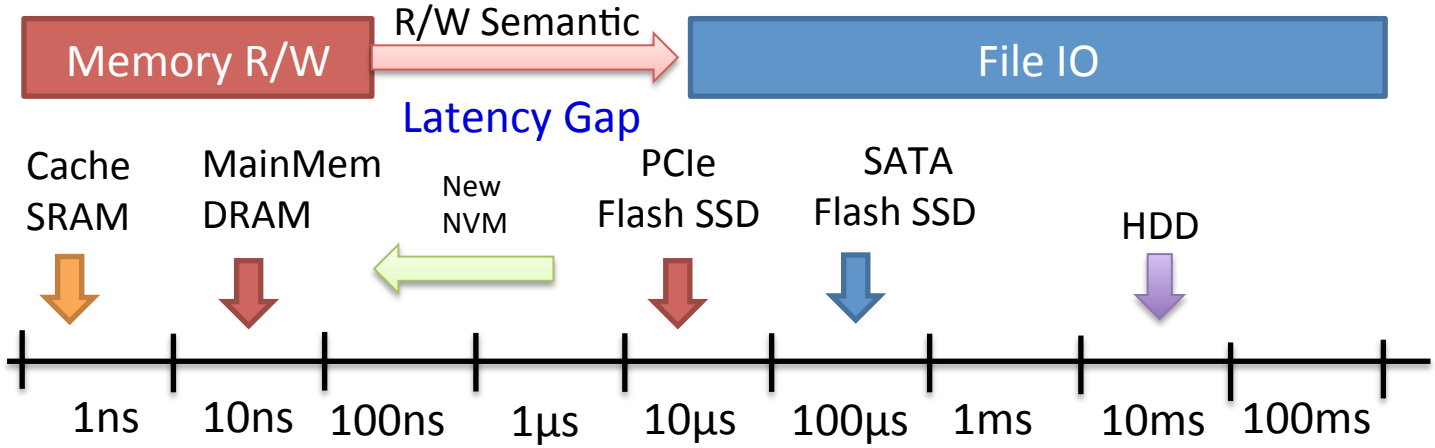
- HDD + Flash Capacity & Speed
- Flash + ReRAM Capacity & Speed & Endurance (Sony, Micron)
- Flash + STT-RAM (Toshiba,Baffalo)

MRAM cache & Flash
Baffalo,2012.6




Hybrid Storage of
ReRAM/TLC NAND Flash with RAID-5/6 ,Takeuchi, 2014.6

# Out of Core Computation Algorithm extracting Data Access Locality



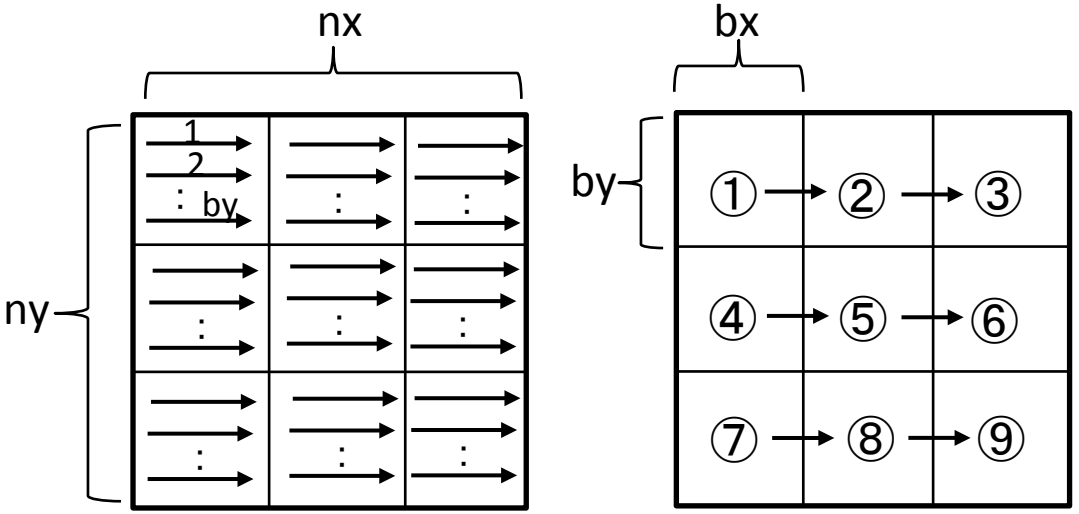

- The Gap between DRAM and PCIe-Flash SSD

  > the Gap between Cache and Main memory (DRAM)

Stencil Computation

The temporal blocking algorithm is newly designed for DRAM-Flash tier to bridge their latency divide.

## Spatial Blocking for Two-dimensional grid domain (nx x ny)



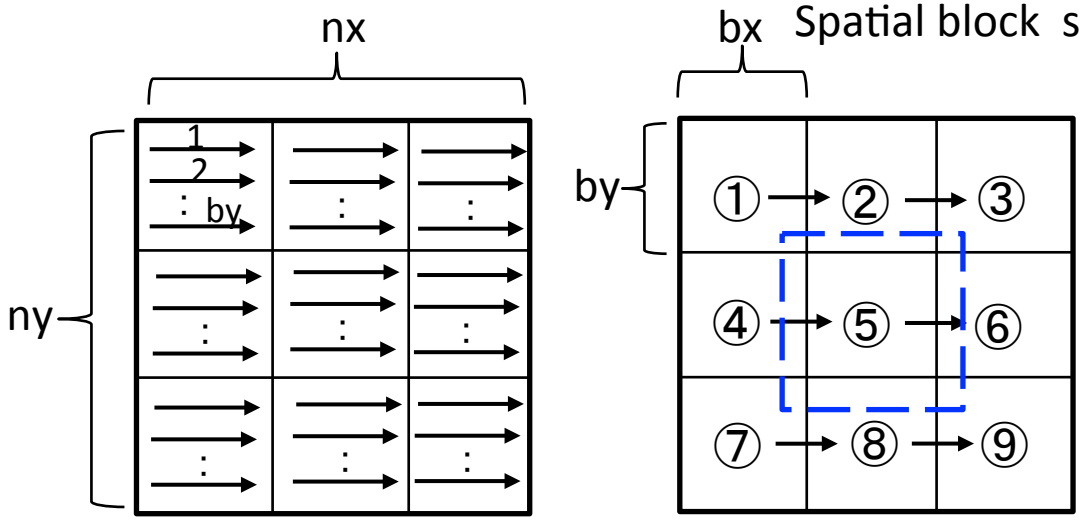

Spatial block size: (bx x by)

Update for one time step at each block

- Blocks from ① to ⑨ are calculated in order.
- After one block is calculated, the next block is processed.

Entire data domain sweeps : Nt times

## Temporal Blocking for Two-dimensional grid domain (nx x ny)



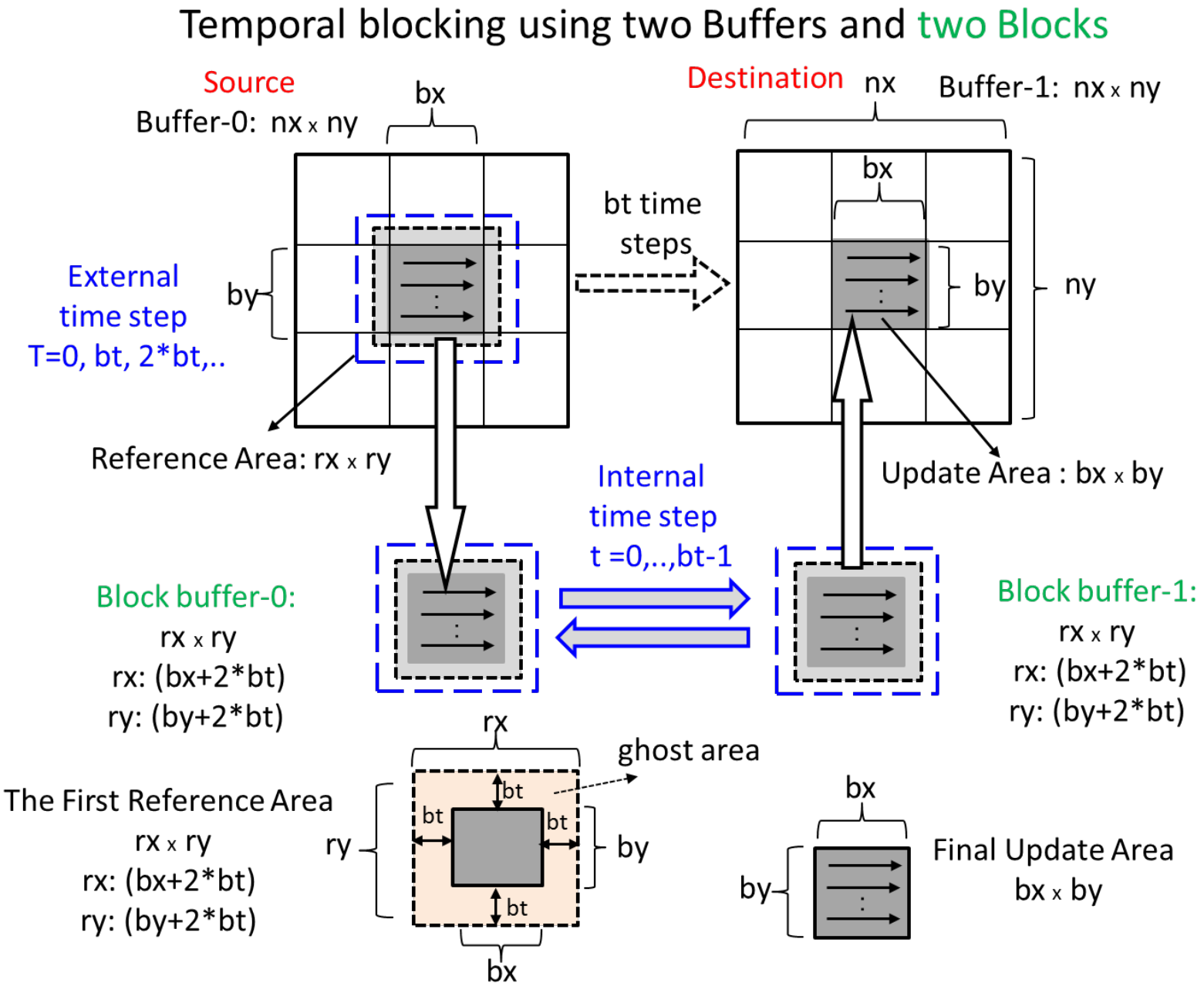

Temporal block size: bt

Update for bt time steps at each block

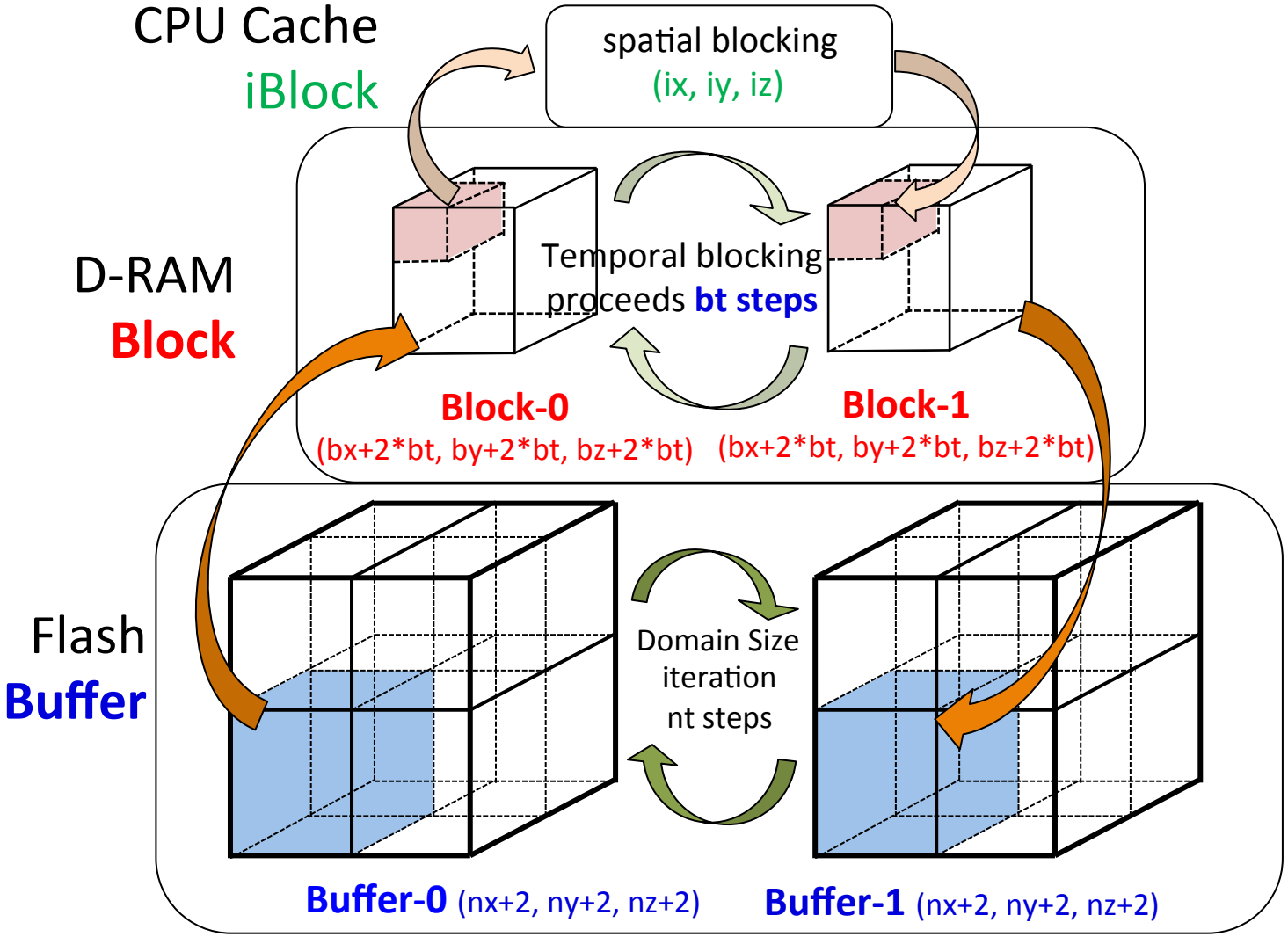
- Blocks from ① to ⑨ are calculated in order.
- After one block is calculated, the next block is processed.

Entire data domain sweeps : Nt/bt times

# Temporal Blocking for Stencil Comp.



# Temporal & Spatial Blocking for three-level Memory Hierarchy



# Stencil Comp. using Flash SSD

## for processing a larger-size problem than physical memory (DRAM) size

Three implementations

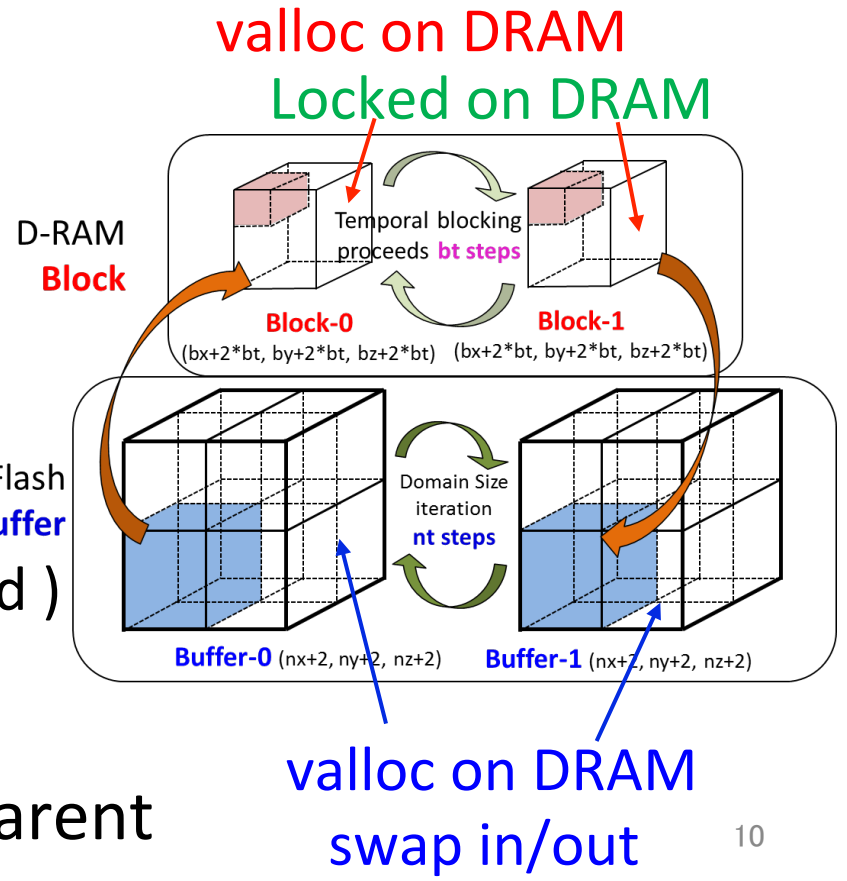
- swap method ( swap system)
  uses a flash as a swap device
- mmap method (file memory map)
  uses a flash as a ext4 file system
- aio method (asynchronous file io)
  uses a flash as a block device for direct input/output

# swap method ( swap system)

## A flash SSD is used as a swap device

( **Buffer** areas are swapped between DRAM and flash SSD)

- 2 **Buffers** and 2 **Blocks** are allocated in page-aligned with valloc()
- 2 **Blocks** are locked on DRAM with mlock()
  (**Blocks** never swap out)
- Linux kernel 3.13.0
  ( fastswap (3.6.0 patch) incorporated )
- No program modification
  Flash SSD is application-transparent



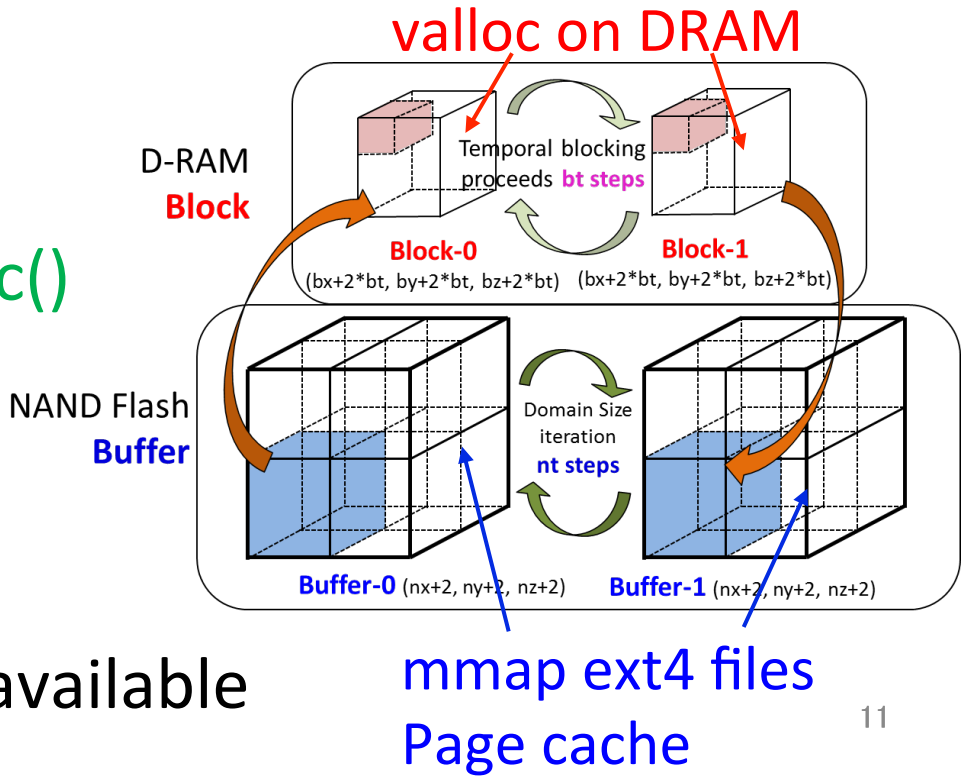
# mmap method (file memory map)

## A flash SSD is used as a ext4 file system

- 2 **Buffers** :
  two ext4 files, memory mapped with mmap().
  accessed by a memory semantic API.
  Page cache: files are cached in available main memory.

- 2 **Blocks** :
  allocated on DRAM
  in page aligned with valloc()

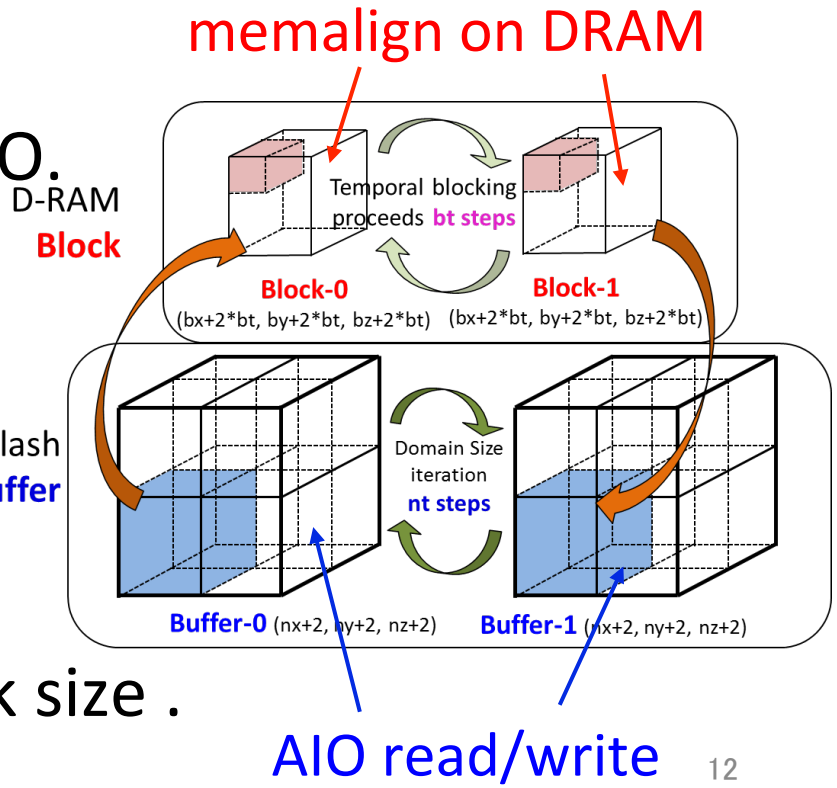
- Programs are almost the same as the programs using swap device,
  Initial & result files are available



# aio method ( Linux asynchronous io lib)

## A flash SSD is used as a block device

- 2 **Buffers** : in a block device.
  direct-accessed by an explicit IO.
- aio non-blocking IO,
  IOs between **Buffer** & **Block**
  overlap **Block** calculations
- AIO parameters, such as
  start adrs, offset and size,
  must be aligned in device block size .




- 2 **Blocks** : allocated in DRAM with posix_memalign(3)

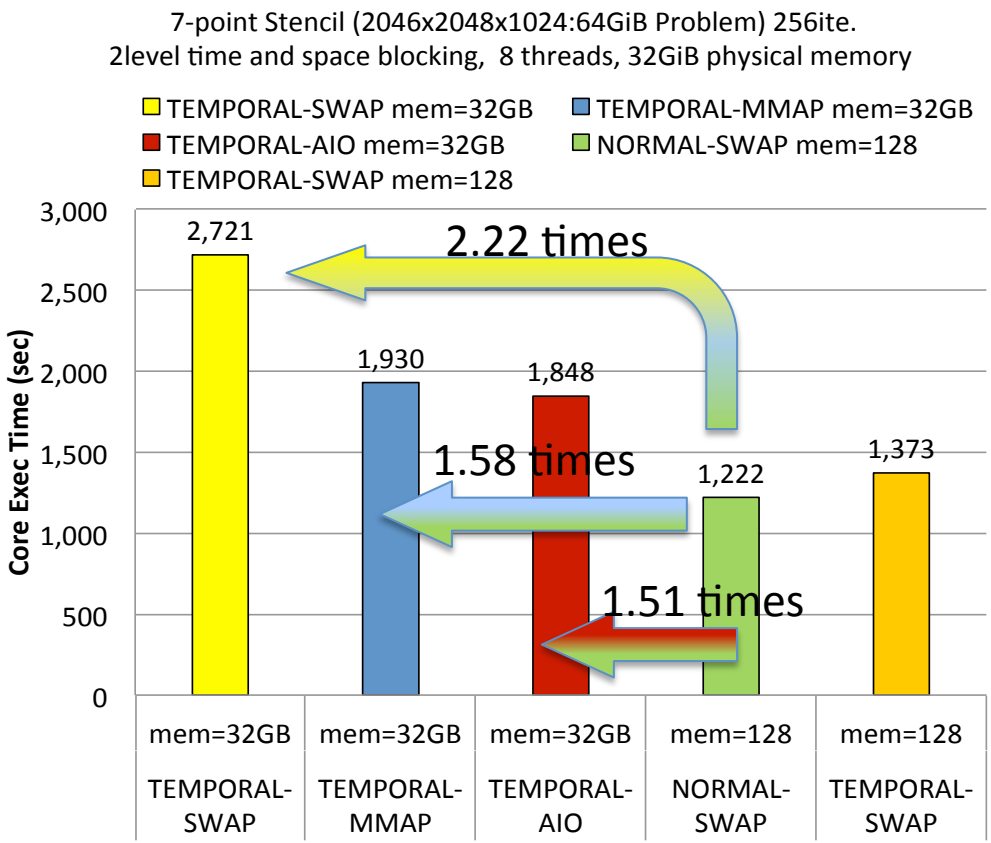
- Programs have to be modified with IO semantic.
  Data layout and access are under restrictions.


12

# Preliminary Evaluation of Three methods

- 7-point Stencil Computation
- Problem Size (64GiB-Problem)
  - 2046x2048x1024 ( 3D-grid domain), 256 time steps
    Grid element : double(8B), Total size of double Buffers: 64GiB
  - Spatial Blocking size : 2046x512x512 grid (8 blocks)
  - Temporal Blocking size : 128 time steps(2 updates of 8 blocks)
- Experimental Environment
  - CPU: Xeon E5-2650 2.0GHz x1 (1-socket, 8 cores)
  - L3 cache: 20MiB
  - Memory: DDR3-1600 ECC 8GiB x4 (32GiB)
  - Flash Storage: Fusion-io ioDrive2 MLC(785GB)
  - OS: kernel 3.13.0, CentOS6.4(x86_64)
  - Compiler: gcc version 4.4.7 20120313 / -O3

# Preliminary Evaluation
# Effect of Temporal blocking



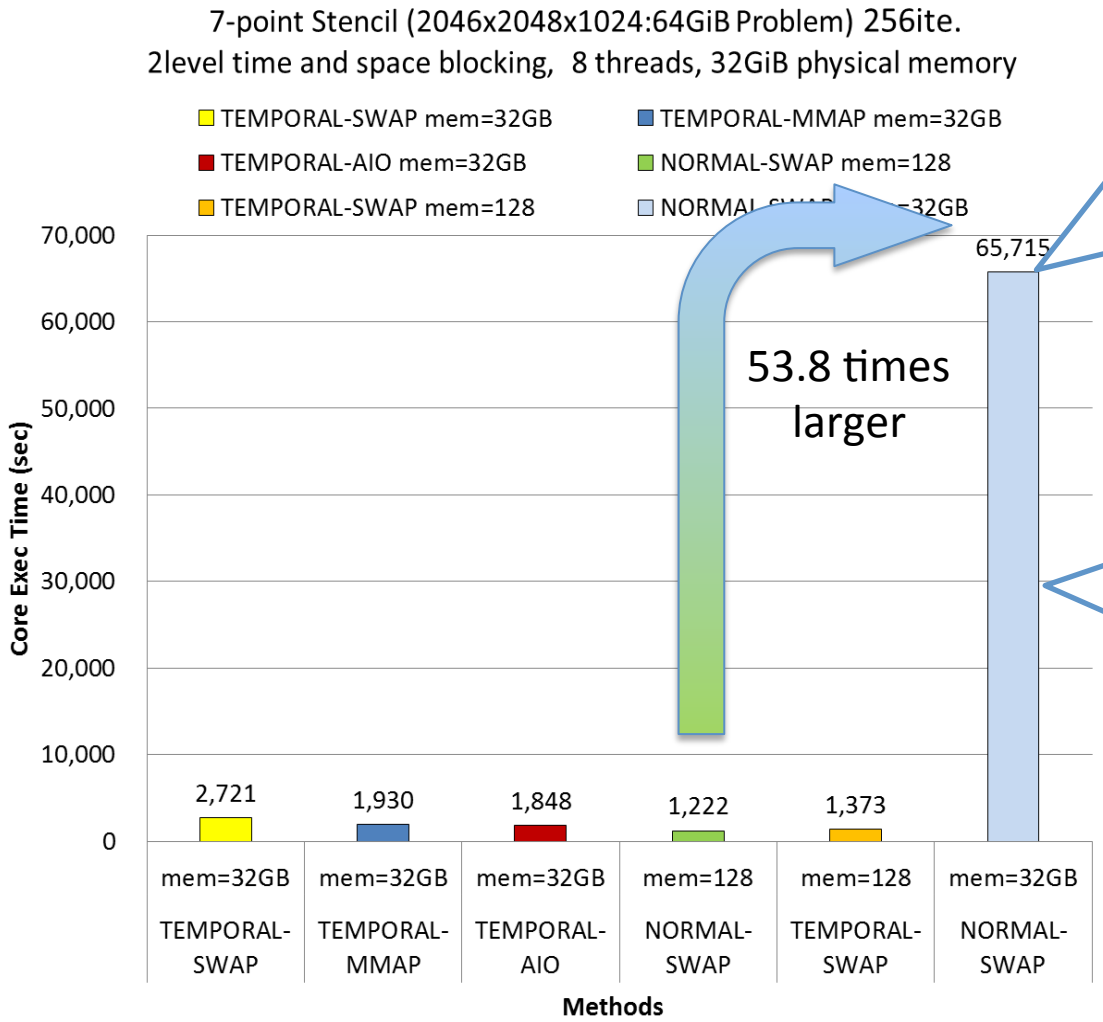

With spatial blocking and temporal blocking

- TEMPORAL-SWAP: 2.22 times larger
- TEMPORAL-MMAP: 1.58 times larger
- TEMPORAL-AIO: 1.51 times larger

7-point Stencil
(2046x2048x1024:64GiB Problem) 256ite.
2level time and space blocking,
8 threads, on 32GiB physical memory

| | |
|---|---|
| CPU | Xeon E5-2650 2.00GHz x1 (8cores) |
| Memory | 32GiB boot (Total memory 128GiB )<br>16GiB(DDR3 1600MHz ECC Reg) x 8 |
| OS kernel | 3.13.0 |
| Compiler | gcc 4.4.7 20120313 -O3 |
| SwapDevice | ioDrive2 (FusionIO) |

# Preliminary Evaluation
# Result without Temporal blocking



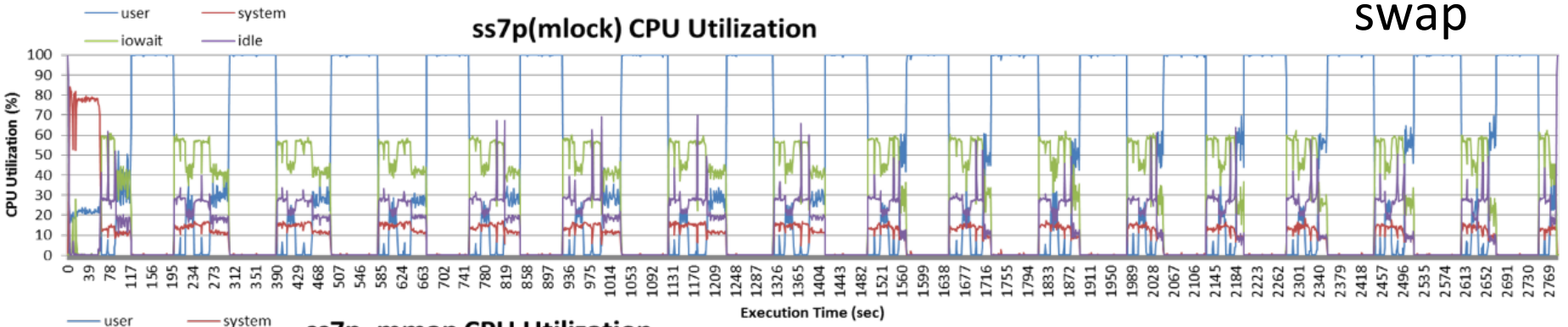

NORMAL-SWAP on 32GB memory with only spatial blocking takes 18 hours 16 minutes, 53.8 times longer time compared with the normal exec. time on 128GB.

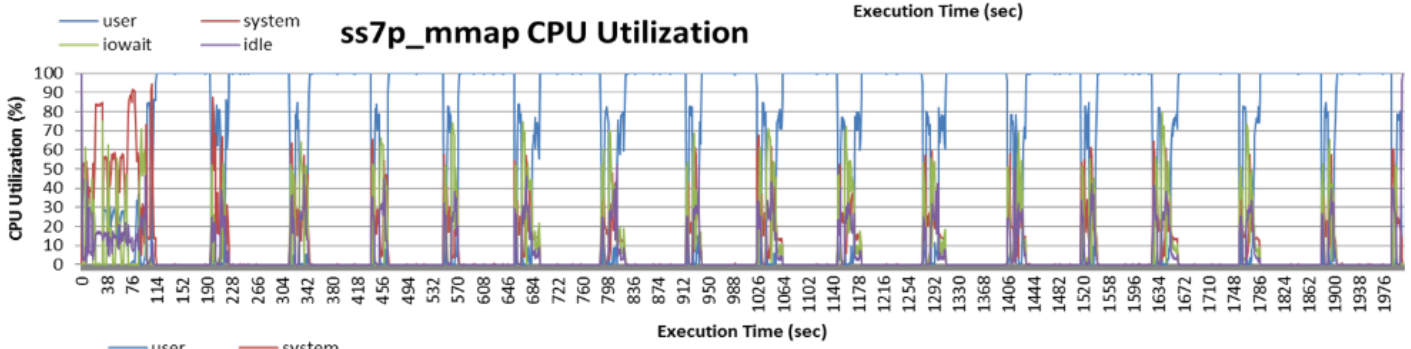
NORMAL-SWAP employs only spatial blocking (32x32) for cache hit

7-point Stencil (2046x2048x1024:64GiB Problem) 256ite.
2level time and space blocking, 8 threads, 32GiB physical memory

# Stencil computation
# CPU Utilization Profile

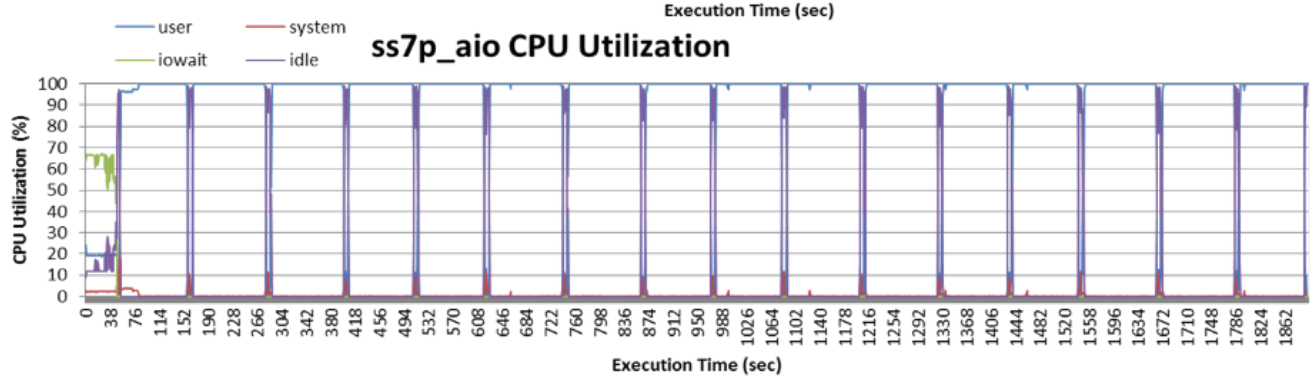
swap




mmap

8 blocks on DRAM
update 2 times



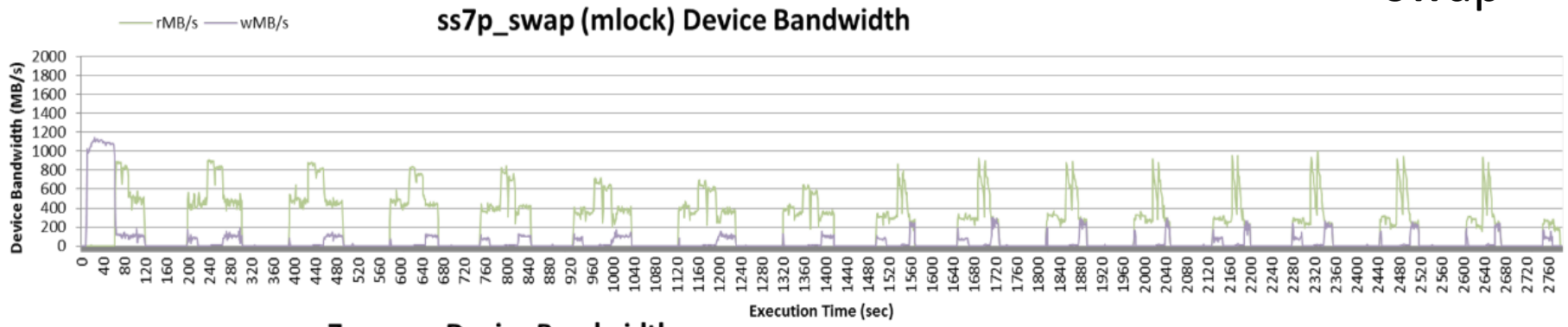

aio

| | Total Size(GiB) | | Elapsed Time |
|---|---|---|---|
| | read | write | (min.sec) |
| swap | 600.90 | 124.69 | 46:24.8 |
| mmap | 209.70 | 156.39 | 33:45.65 |
| aio | 110.25 | 128.19 | 31:36.5 |

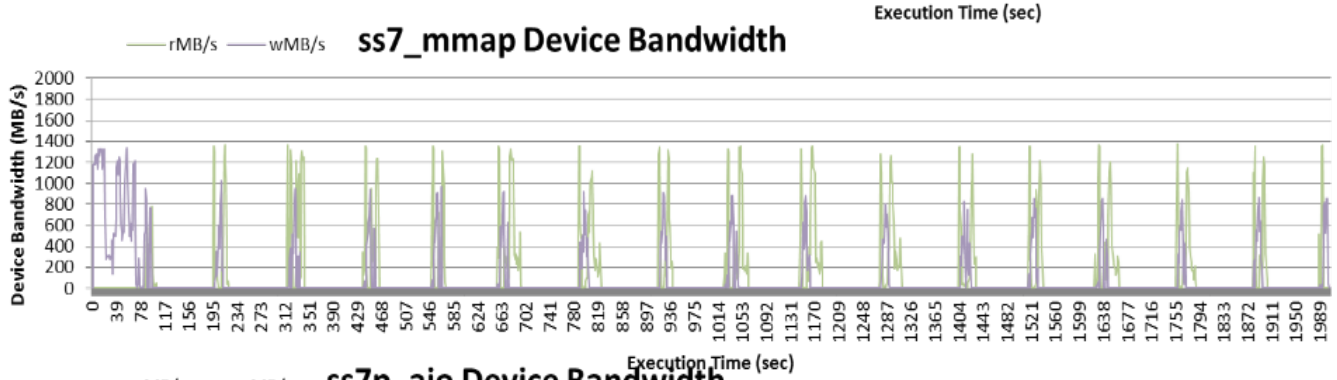
TIme

# Stencil computation
# Flash SSD IO Bandwidth profile

swap

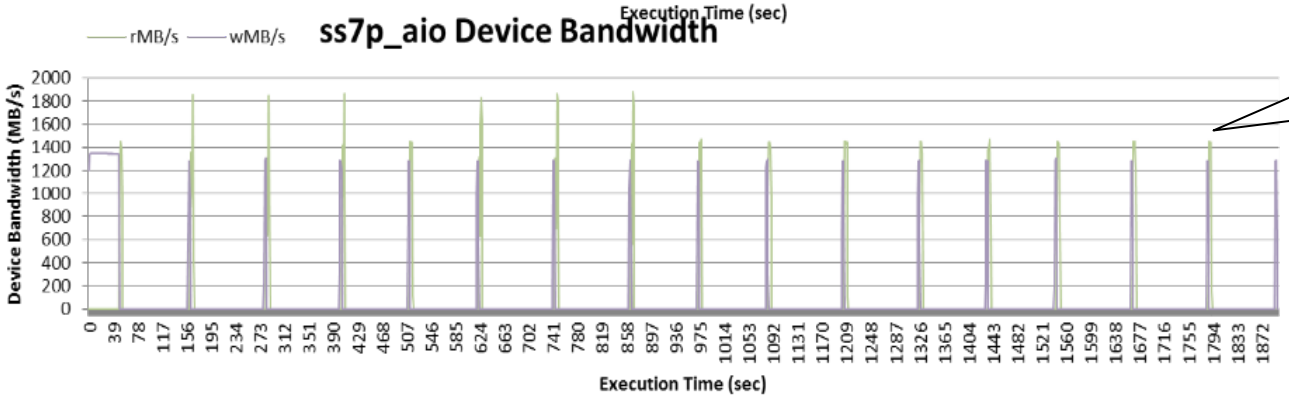



mmap



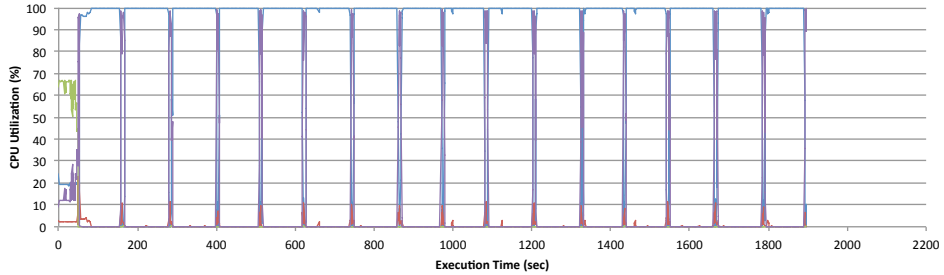

aio achieves almost maximum BW of the Flash device provides

aio

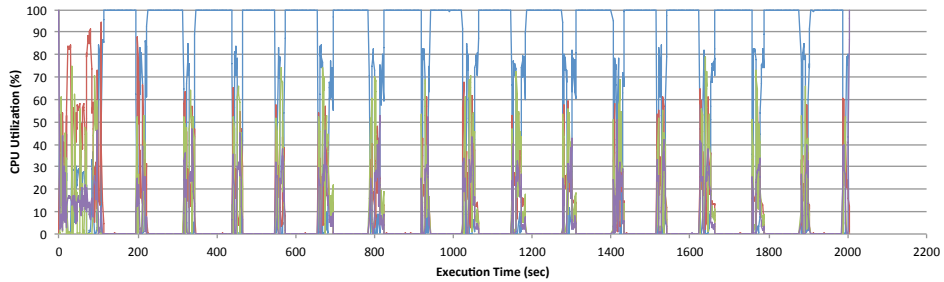


# Profiles of CPU utilization and IO for three methods

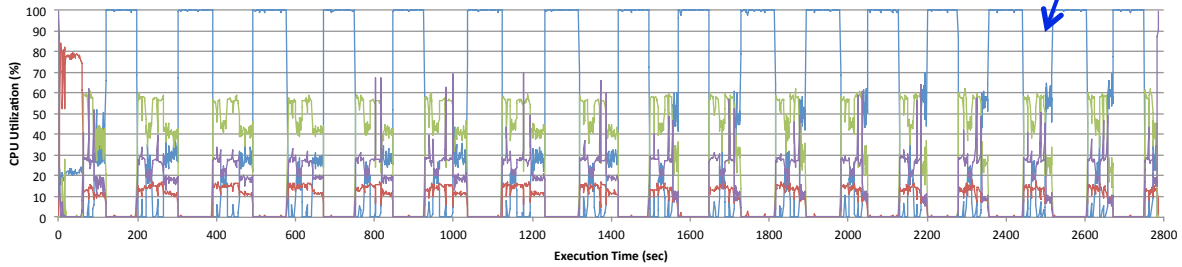
aio method




mmap method




Swap method




Problem Size 64GiB
Inner Loop block 32x32x32

| Methods | aio | mmap | swap |
|---|---|---|---|
| Exec. Time (sec) | 1847.55 | 1929.86 | 2720.58 |
| Effectice Mflops | 5945.38 | 5691.79 | 4037.51 |
| One Block Average Period Time (sec) | 114.67 | 118.47 | 169.13 |
| One Block Average IO Time (sec) (CPU utilization <90% ) | 7.73 | 29.73 | 85.27 |

※Average Time of 15 iterations excluding the first one.

IO time: aio < mmap

Total time: aio ≅ mmap

Optimization to reduce calculation time of aio

# Tuning temporal blocking parameters for higher performance

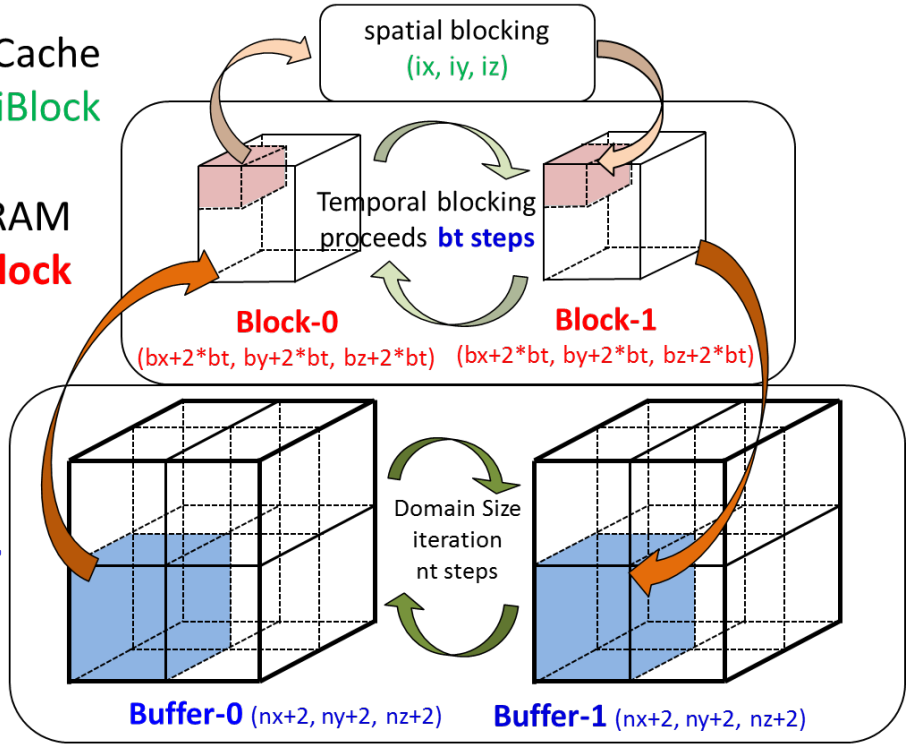
- Internal block shape
- Thread work share scheme

CPU Cache
iBlock

D-RAM
**Block**

- Block shape
- Block memory layout

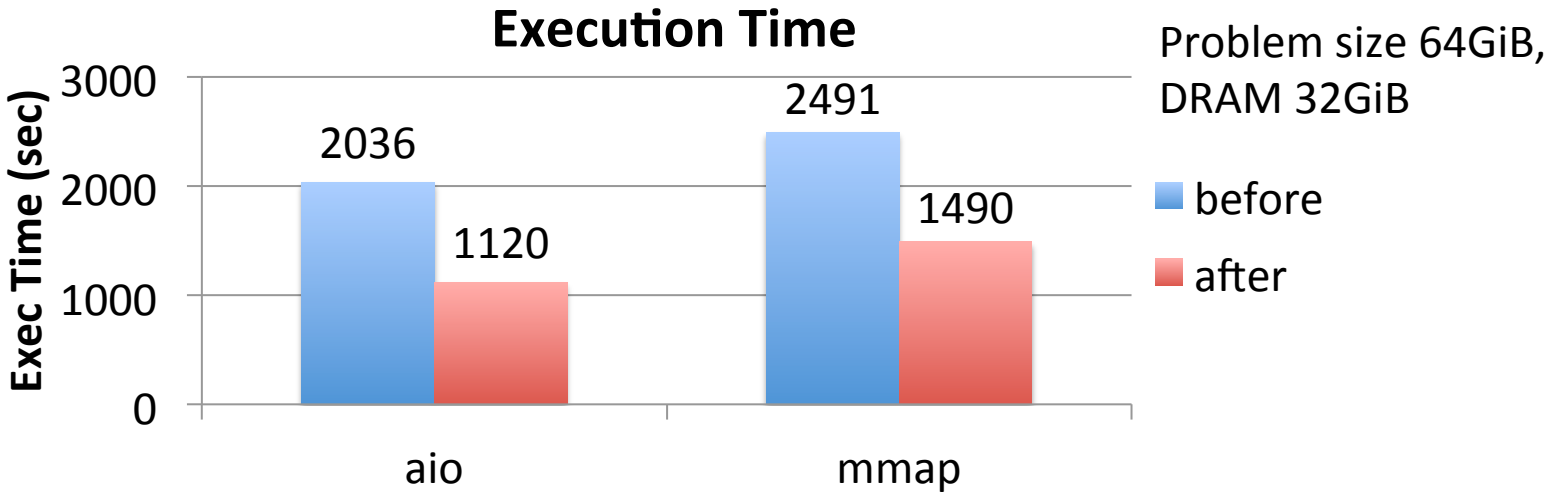
NAND Flash
**Buffer**




parameter select policy

1. fit block/iblock volume in DRAM/L3 cache size
2. select block shape and a memory layout with device-block-size aligned fashion
   to increase sequential access
   to reduce memory access conflicts by pixel/page padding for block layout
   to increase work share efficiency for iblock shape according to work share scheme
3. select work share scheme to increase core independent and parallel calculation

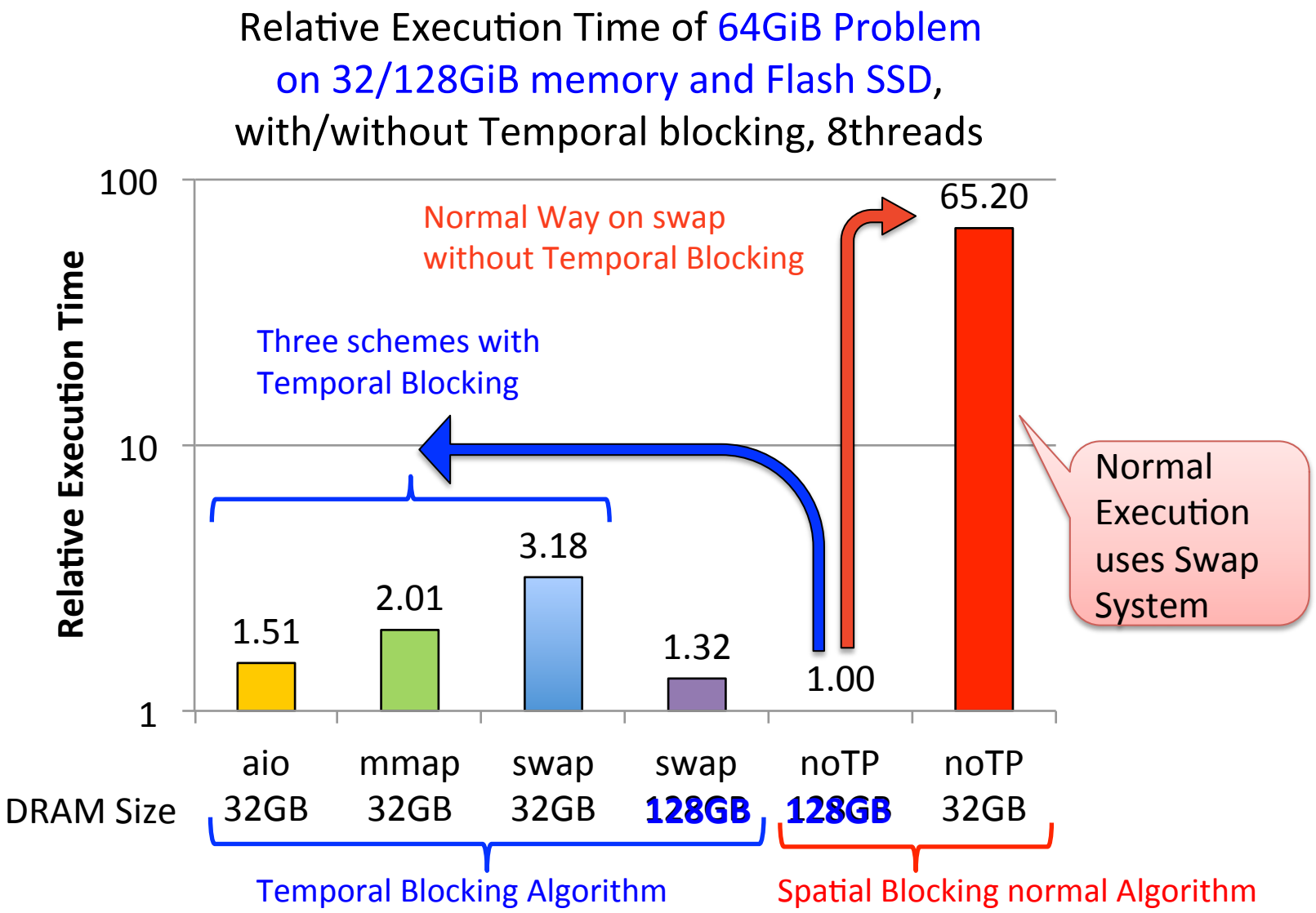
# Performance of Before/After tuning



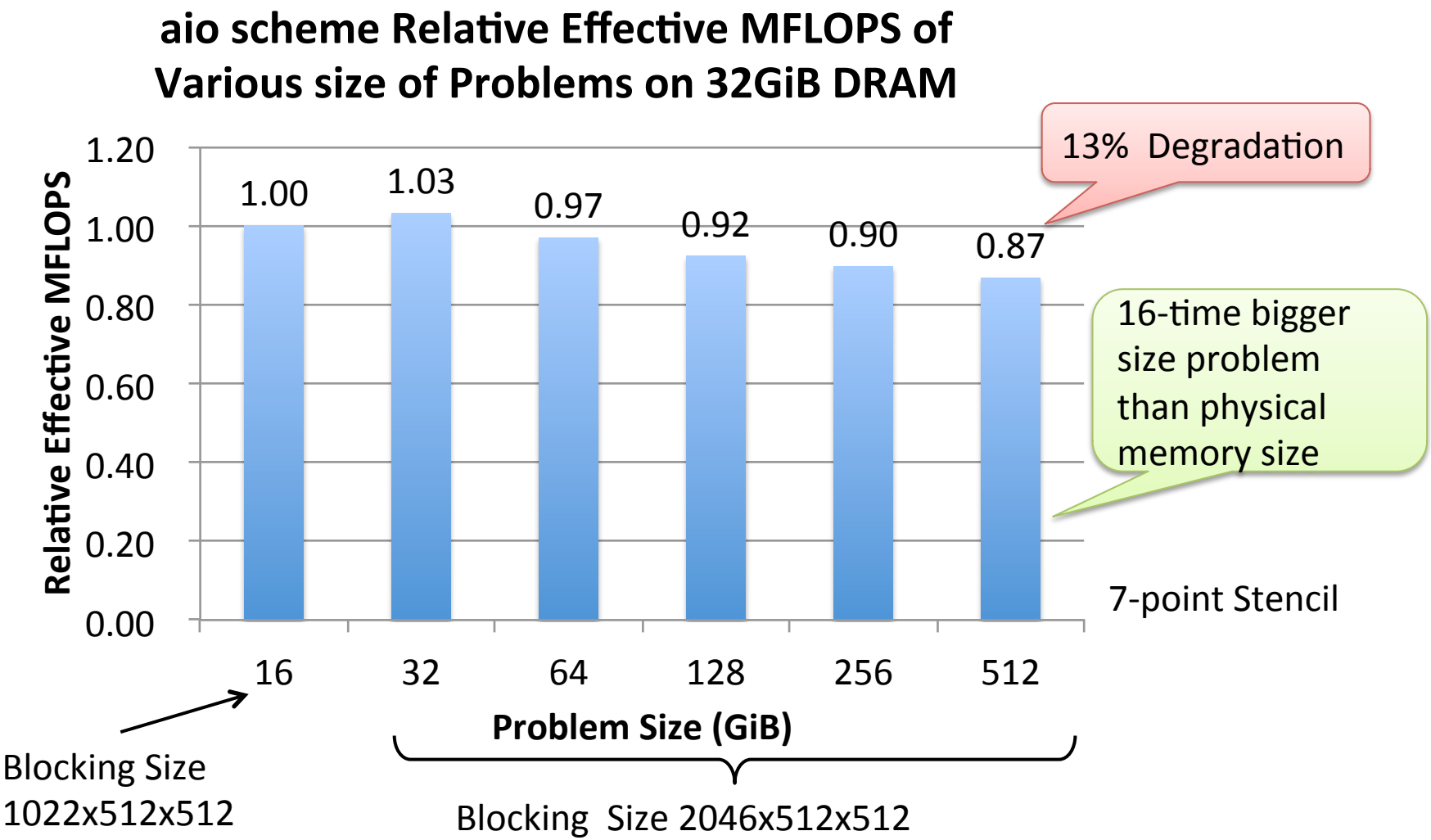

| Methods | aio Method | | mmap Method | |
|---|---|---|---|---|
| | Before Opt. | After Opt. | Before Opt. | After Opt. |
| Exec. Time (sec) | 2036.13 | 1119.17 | 2490.59 | 1492.25 |
| Effective Mflops | 5394.74 | 9814.72 | 4410.35 | 7360.97 |
| One Block Average Period Time (sec) | 126.40 | 69.13 | 154.47 | 91.53 |
| Relative Period Time | 1.00 | 0.55 | 1.00 | 0.59 |
| One Block Average IO Time (sec) (CPU utilization <90%) | 8.33 (6.6%) | 8.20 (11.9%) | 36.53 (23.7%) | 31.73 (34.7%) |

※Average Time of 15 iterations excluding the first one.

# Impact of Temporal blocking



# Performance of various-size Problems on fixed-size Physical memory



# Auto-Tune Example 2-socket, 64GiB-mem. 1/3

```
% ./ss7p -n 4094 4096 2048 -t 1000 -d /dev/sdc
----- autotune start -----
socket = 2
core   = 16
total  = 67667959808 B (64533.20 MiB)
L3     = 20971520 B (20.00 MiB)
L2     = 262144 B
L1     = 32768 B
reserve_base_size = 1073741824, reserve_core_size = 134217728
reserve_size = 3221225472 B, (3072.00 MiB)
usable_size = 64446734336 B, (61461.20 MiB)
(nx,ny,nz), nt = (4094,4096,2048), 1000
 buf_size = 275280691200 B (262528.12 MiB), buf[0][1]_size = 550561382400 B (525056.25 MiB)
  rd_buf_size = 0 B (0.00 MiB)
  buf_total_size = 550561382400 B (525056.25 MiB)

device = /dev/sdc
BLKSSZGET = 512
BLKGETSIZE64 = 4088798380032
b_yz = 991
(bx,by,bz), bt = (4094,1024,512), 0
  block_size = 17179869184 B (16384.00 MiB), block[0][1]_size = 34359738368 B (32768.00 MiB)
  rd_block_size = 0 B (0.00 MiB)
  block_total_size = 34359738368 B (32768.00 MiB)
  remain_size = 33308221440 B (31765.20 MiB)
```

User command

Get System Information
16 cores, 2 socket,
DRAM(64GiB), L3(20MiB)

User command parameters

Get device block size(512MB), capacity(4TiB)

select spatial block size

# Auto-tune Example 2-socket, 64GiB-mem. 2/3

```
remain_size = 33308221440 B (31765.20 MiB)
max_bt = 128
Update ite. = 8
(bx,by,bz), bt = (4094,1024,512), 125
  block_size = 31810781184 B (30337.12 MiB), block[0][1]_size = 63621562368 B (60674.25 MiB)
  rd_block_size = 0 B (0.00 MiB)
  block_total_size = 63621562368 B (60674.25 MiB)
  remain_size = 4046397440 B (3858.95 MiB)

l3_size          = 20971520 B (20.00 MiB)
usable_cache_size = 20970496 B (20.00 MiB)
(ix,iy,iz) = (4094,1,32) : 16769024 B (15.99 MiB)
(ix,iy,iz) = (4094,1,40) : 20961280 B (19.99 MiB)
  remain_cache_size = 10240 B (0.01 MiB)
----- autotune end -----
ss7p version $Id: stencil_aio.c,v 1.01 2014/06/30  Hiroko Midorikawa Exp $
(nx,ny,nz)=(4094,4096,2048), nt=1000
(bx,by,bz)=(4094,1024,512), bt=125
(cx,cy,cz,cc)=(0.100000,0.100000,0.100000,0.400000)
(ix,iy,iz)=(4094,1,40), 2620160 B
This process is using OpenMP.
num_thread in this process 16
```

Balacing bt over iterations

Final spatial block size and temporal size are fixed

Final inner block is fixed

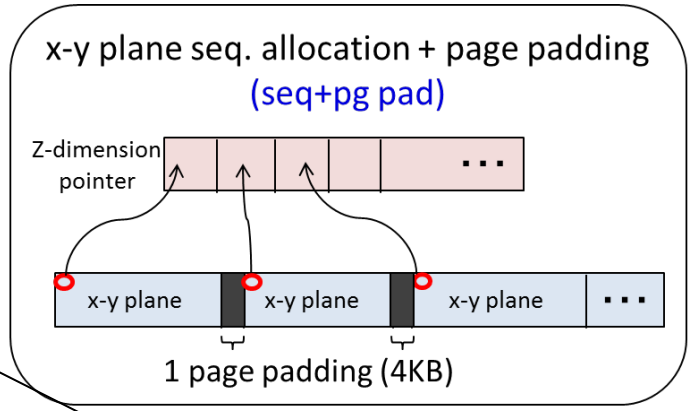
Execution starts with optimized tuned parameters

OpenMP uses 16 threads

# Auto-tune Example 2-socket, 64GiB-mem. 3/3

```
calc threads: 16
CPU_BIND
internal workshare loop is Z
block:(4344,1274,762) -> (4096,1274,762), start_x = 0
valloc p 6096 B
size_xy = 41746432 B
ALL_ALLOC
page_padding = 1, padding_size = 4096
alloc_size = 31813902336 B
mbind 0: x-y 0x7f4c70636000, 15906951168
mbind 0: z 0x20ff000, 4096
mbind 1: x-y 0x7f5024843000, 15906951168
mbind 1: z 0x20ffbe8, 3048
size_xy_total = 31813902336 B, 30340.10 MiB, 29.63 GiB
size_xyz_total = 31813908432 B, 30340.11 MiB, 29.63 GiB
                    : (Block1 log is omitted )
before g.buf_lxy = 16785408
after g.buf_lxy = 16785408, size = 134283264 B
buf:(4096,4098,2050) -> (4096,4098,2050), start_x = 0
buf size: 275280691200 B, 262528.12 MiB, 256.38 GiB
init...
use open(/dev/sdc, O_DIRECT)
```

Threads are bound to sockets




Block0 is created

Block0 is divided into sub-blocks and mbind to each socket

Block1 is created, divided into sub-blocks and bound to each socket

Buffer information

Execution starts with optimized tuned parameters

# Summary

- swap vs. mmap for memory access APIs
  - Usually mmap performs better than swap. Both are easy to use.
  - mmap performance partially depends on available kernel page cache size and mmap sometimes requires OS kernel tuning for page cache.
- Linux kernel aio for explicit IO
  - Aio gains higher level of parallelism for multi-core systems and non-blocking IO for overlapping to calculations.
  - Aio has a limitation in program data structures. IO parameters must be aligned in device-block size.
  - Aio can be controlled by application without little interference of the kernel.
- Application specific locality-aware algorithm
  - Flash are available for large and slow memory in practical use for stencil computations by the temporal blocking algorithm designed for flash.
  - Aio method is most effective among three methods for Flash SSDs.
  - The auto tuning relieves us from annoying parameter tuning for temporal blocking.

# future works & co-design

- Efficient Programming framework

  Enhancement of the auto tuning system by adding more flexibility and generality for temporal blocking stencil computations with programmer-friendly interface

  co-design: share application knowledge to extract generality in calculation & data structures for designing API

  co-design: share design & implementation knowledge for API , DSL, frameworks for the programming interface

- Enhancing multiple-nodes calculations on a cluster

  Combining of distributed DRAM, Flash and HDD for stencil computations.

  co-design: we want simple, portable & fast communication library with sufficient multi-thread support for system software, not app.

  Efficient file systems for input/output, chkpt/restart data structure for NVM.