CCGrid2015 The 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 4-7, 2015, Shenzhen

Locality-Aware Stencil Computations using Flash SSDs as Main Memory Extension

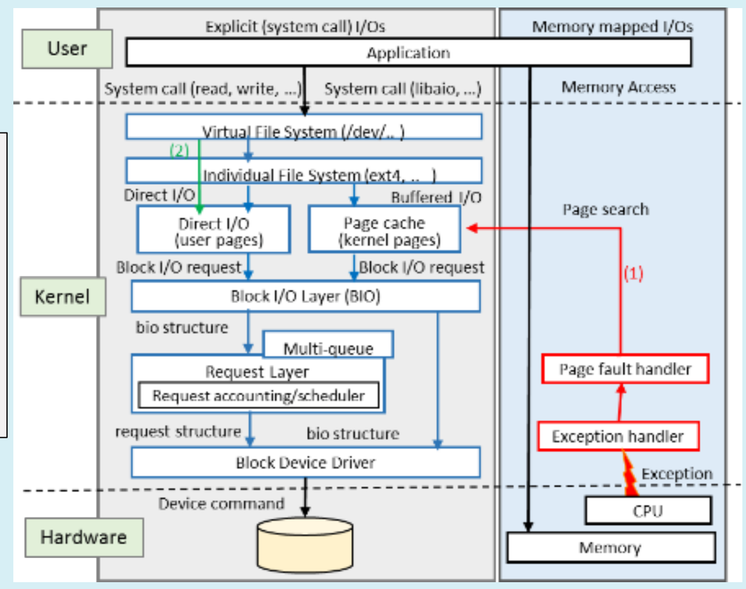
Hiroko Midorikawa, Hideyuki Tan, JST CREST & Seikei University, Tokyo Japan
midori@st.seikei.ac.jp http://www.ci.seikei.ac.jp/midori/paper

Out-of-core Stencil Algorithm using Flash SSDs

Three methods to access to Flash SSDs

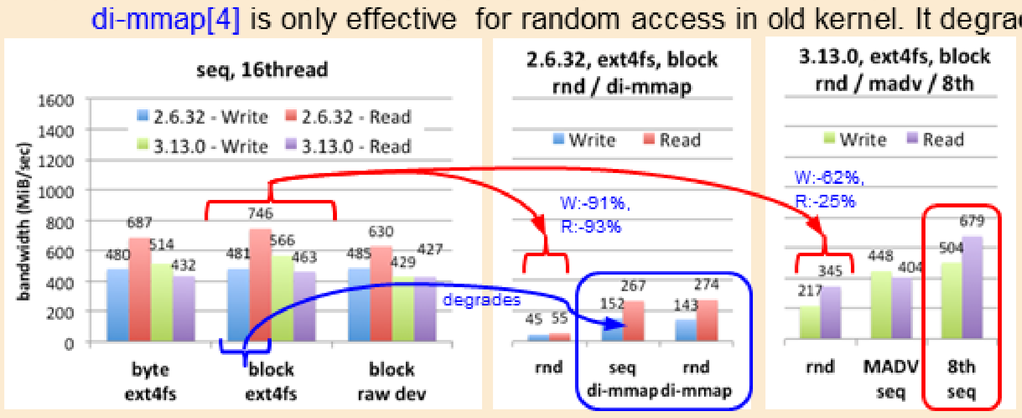
- swap method (swap system): swap device → transparent for app. slow. swapd overhead
- mmap method (file memory map): ext4 file system → easy to use. depends on page cache space, Out of mem killer
- aio method (asynchronous file io) : block device for direct I/O → fast, requires block-size-aligned access

Multiple access paths to a SSD device: mmap and direct aio

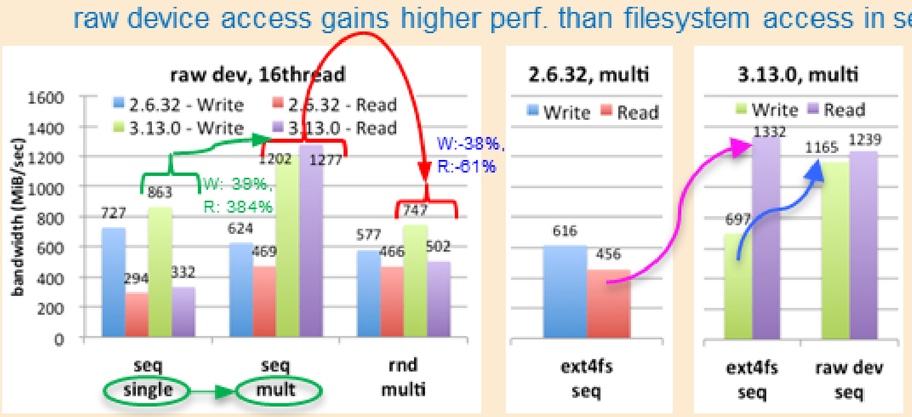


aio vs. mmap in access bandwidth to a flash SSD

mmap: random access perf. is very low compared to seq. access perf. in kernel(2.6.32)
di-mmap[4] is only effective for random access in old kernel. It degrades seq. access.



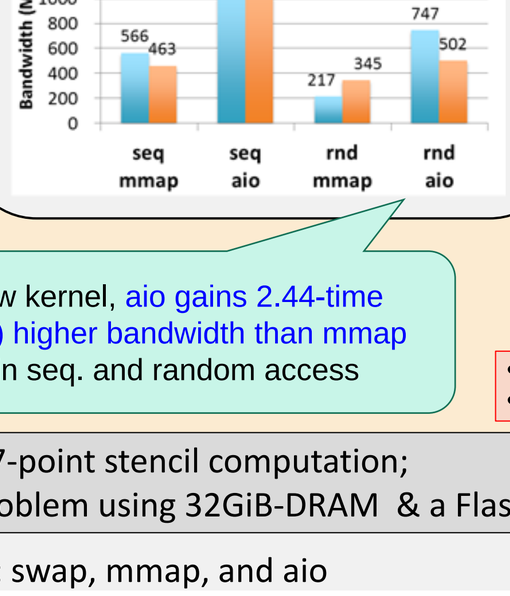
aio: multiple deeper queues in newer kernel(3.13) is effective for seq. access
raw device access gains higher perf. than filesystem access in seq. write



Micro benchmarks
(128GiB data/64GiB mem)

- Kernel version
- I/O queue depth (1 or 64K/thread)
- Access mode (seq, random)
- File system vs. Raw device
- I/O size (block(4KiB), byte)

aio vs. mmap in kernel 3.13.0



In new kernel, aio gains 2.44-time (ave.) higher bandwidth than mmap both in seq. and random access

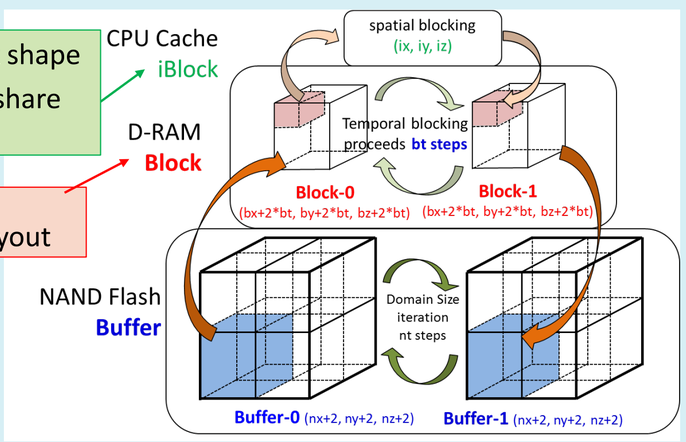
7-point stencil computation;
64GiB-problem using 32GiB-DRAM & a Flash

Locality-aware algorithm:
temporal and spatial blocking for *three-memory layers*

Tuning temporal blocking parameters for higher performance

- Internal block shape
- Thread work share scheme

- Block shape
- BLock memory layout

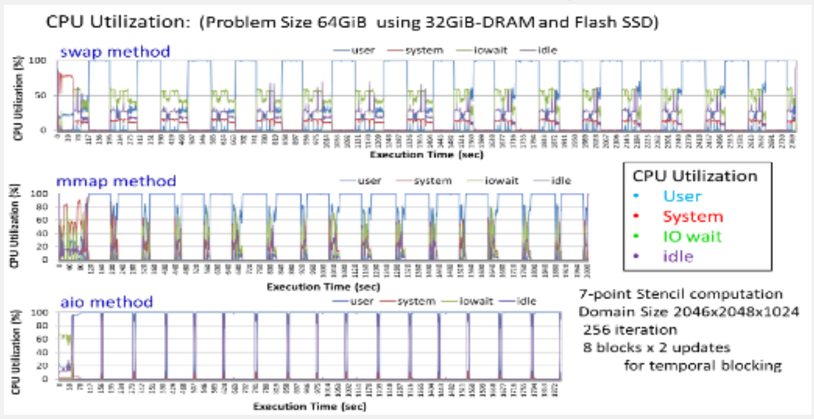


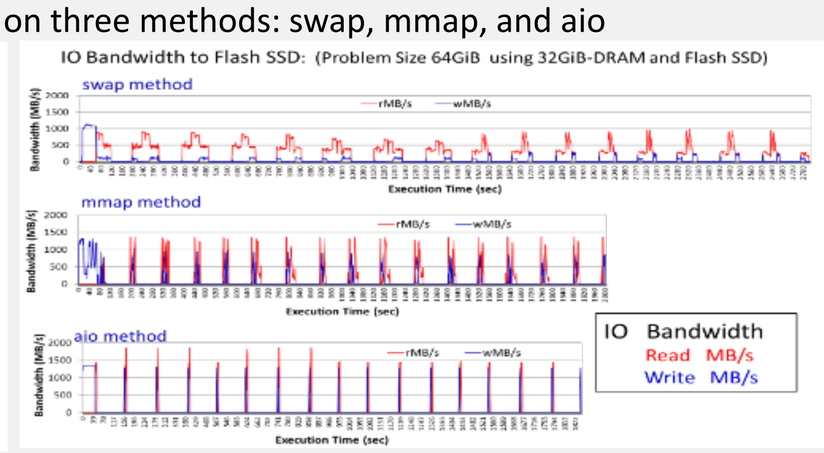
parameter select policy

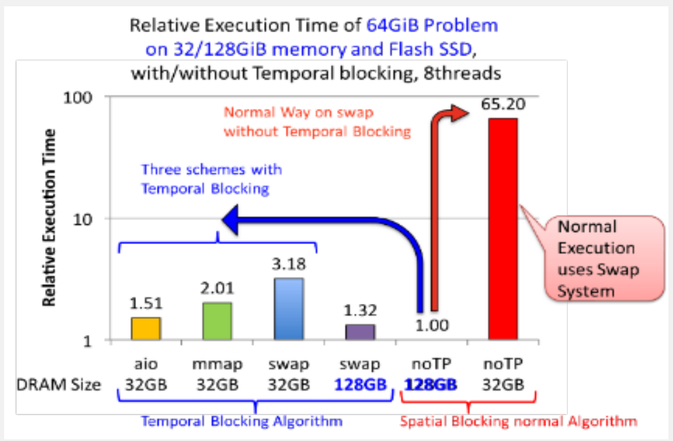
1. fit block/iblock volume in DRAM/L3 cache size
2. select block shape and a memory layout
   with device-block-size aligned fashion
   to increase sequential access
   to reduce memory access conflicts by pixel/page padding for block layout
   to increase work share efficiency for iBlock shape according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

Performance on UMA-Systems

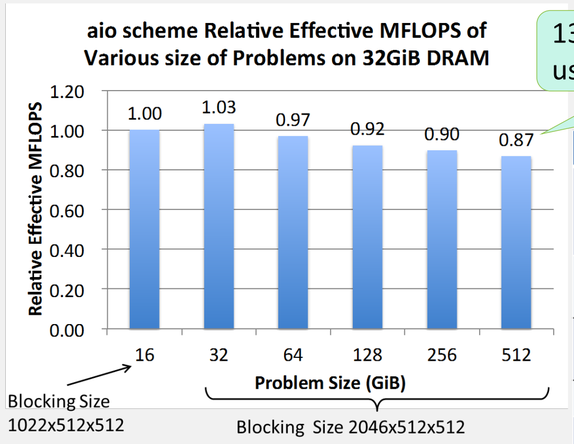
CPU Utilizations and I/O Bandwidths on three methods: swap, mmap, and aio







Relative Execution Times of three methods



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

13% degradation for 512GiB-problem using 32GiB memory

	UMA	NUMA
CPU	Xeon E5-2650 2.0GHz 1 socket (8 cores)	Xeon E5-2687W 3.1GHz 2 socket (8x2 cores)
Memory	DDR3-1600 ECC 8GiB x 4 (32GiB)	DDR3-1600 ECC 8GiB x 8 (64GiB)
Flash Storage (PCI Bus)	Fusion-io ioDrive2 MLC(PCIe2.0x4) 785GB, 1.2TB	
OS	kernel 3.13.0, CentOS6.4(x86_64)	
Compiler	gcc version 4.4.7 20120313 / -O3	

Runtime Auto Tuning

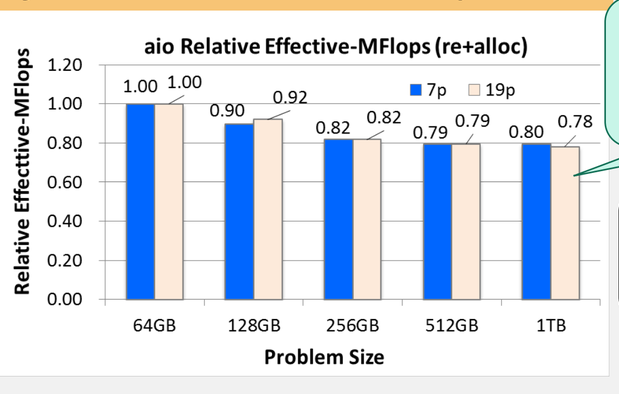
1. User command parameters
   domain size(nx,ny,nz), Time step(nt), Flash device path

```
% ./stencil7p -n 4094 4096 2048 -t 1000 -d /dev/sdc
----- autotune start -----
(bx,by,bz), bt = (4094,1024,512), 125
(ix,iy,iz) = (4094,1,40) : 20961280 B (19.99 MiB)
----- autotune end -----
        :
```

2. Get hardware information
   device capacity, device block size, DRAM size, L3 cache size, # of Cores, # of CPU sockets
3. Calculate optimal blocking sizes by adjusting to underlying hardware
   - Efficient spatial block size and shape, temporal block size (optimal bx,by,bz, bt for Blocks on DRAM)
   - Efficient spatial inner block shape and size (optimal ix,iy,iz for inner blocks on L3 cache)
4. Numa-aware computing
   - Blocks are divided into sub-block areas according to # of CPU sockets
   - memory-bind & cpu-bind between each sub-block to each local socket
   - locality-aware sub-block computation on each CPU socket using local cores

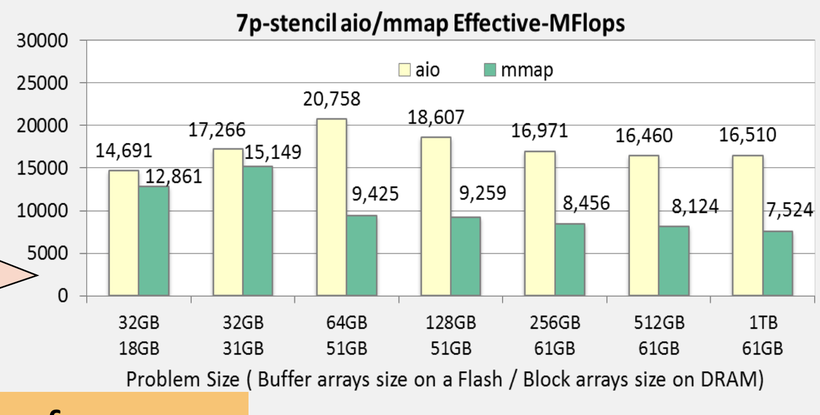
The auto tuning releases users from the burden of choosing parameters for individual systems they utilize.

Performance on NUMA-Systems



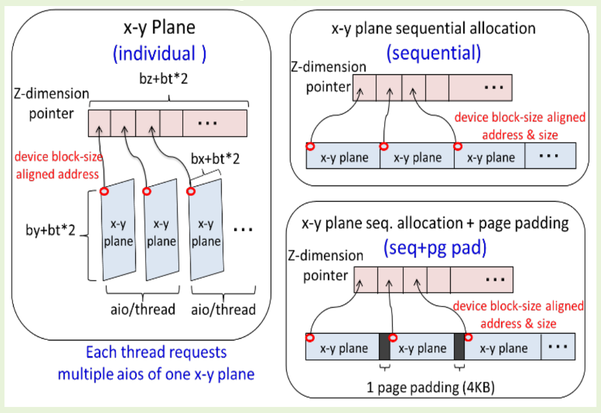
Only 20% degradation for 1TiB-problem using 64GiB memory

aio gains much better perf. than mmap

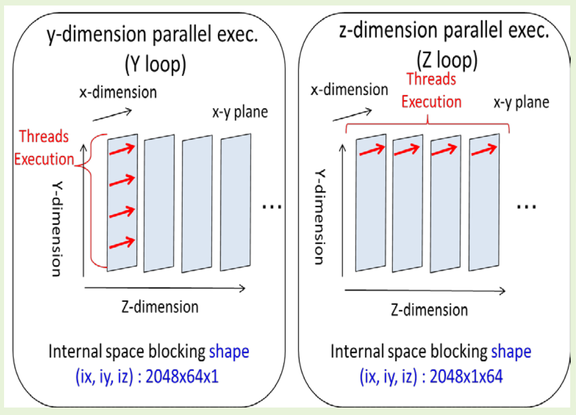


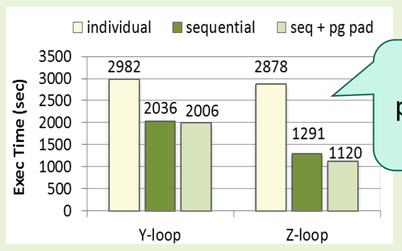
Optimization and parameter tuning for higher performance

Three memory layouts for Block

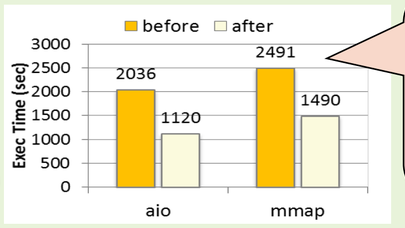


iBlock shape & work-share



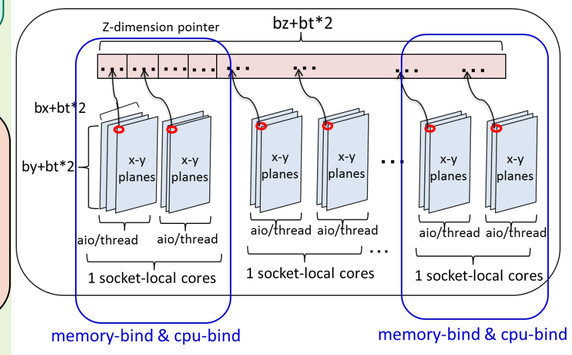


Z-dimension parallel with page padding



Optimizations reduce exe. times on aio and mmap methods HALF !

Tuning for Numa systems



Numa Tuning

1.55-time increase in MFlops