HPDC'16 The 25th International Symposium on High Performance Parallel and Distributed Computing
Kyoto, Japan, May 31 - June 4, 2016

Blk-Tune
Blocking Parameter Auto-Tuning for Flash-based Out-of-Core Stencil Computations

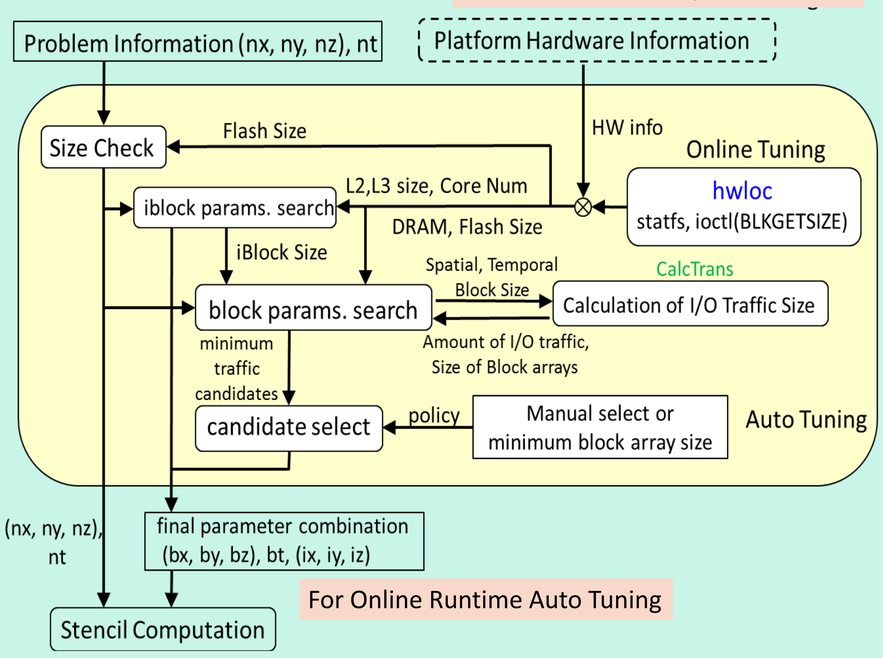
Hiroko Midorikawa JST CREST & Seikei University, Tokyo Japan
midori@st.seikei.ac.jp http://www.ci.seikei.ac.jp/midori/paper

Abstract

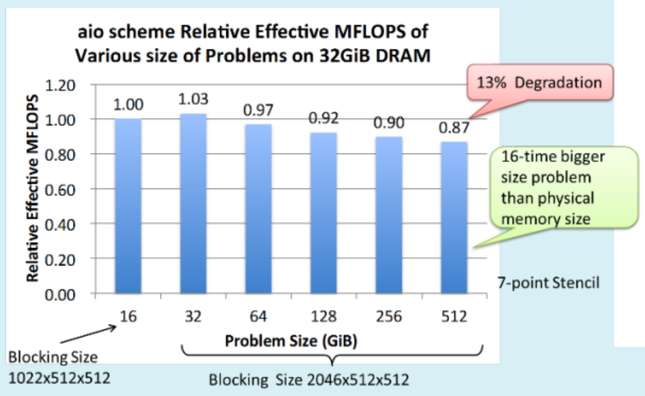
Blk-Tune is a runtime-based blocking parameter auto-tuning system that enables the use of flash memory as an extension of main memory. It obtains platform hardware information in runtime by using Portable Hardware Locality (hwloc) [3], which is widely portable to various kinds of OS and CPU architectures. By calculating the total amount of IO traffic to a flash device, it selects an optimal combination of spatial and temporal blocking sizes to suit the capacity of each memory layer (flash, DRAM, level-3 cache, and level-2 cache). A selected parameter combination minimizes the amount of data transferred between the flash device and DRAM, which is the dominant factor affecting the performance of out-of-core algorithms using flash. ***Blk-Tune*** allows users to easily achieve maximum performance of large-scale stencil computations for particular platforms and application settings.

Overview of Blk-Tune

Auto Tuning System Diagram

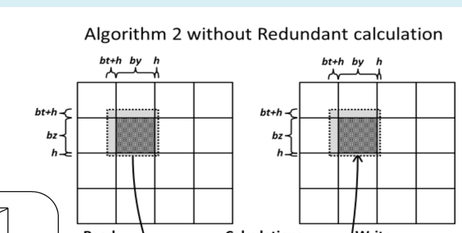


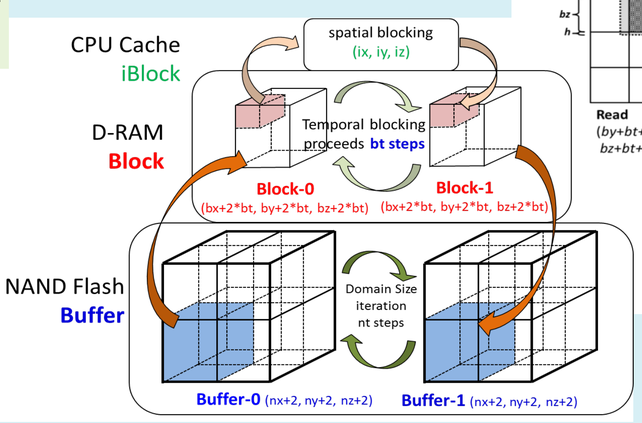
Blk-Tune: Auto-tuning system for out-of-core stencil computations using flash, DRAM, and cache



Out-of-core Stencil Algorithm using Flash SSDs

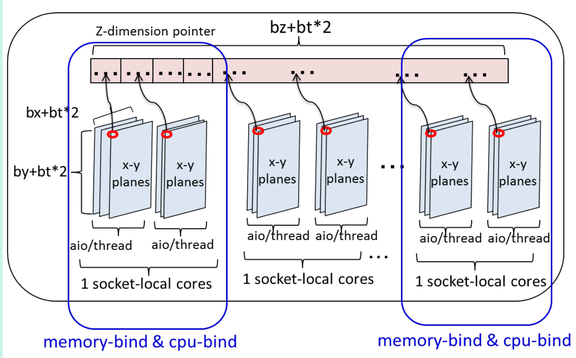
Locality-aware algorithm:
temporal and spatial blocking for *memory layers*





Three data arrays in memory layers, *Buffer arrays* in Flash SSD, *Block arrays* in DRAM, and *iBlock arrays* in the cache for locality extraction.

Tuning for Numa systems



Numa Tuning ⇩ 1.55-time increase in MFlops

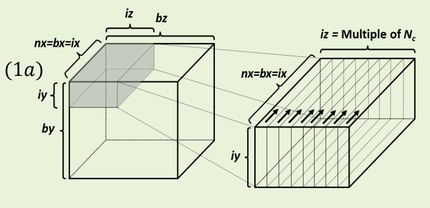
Search procedures

Step1 *find optimal iBlock size (ix, iy, iz)*

- ***Objective Function***

$$min\ (\ \ |\ k * iBsz(ix,iy,iz) * Esz - Sc2 * Nc\ |\ \ ) \quad (1a)$$
$$where\ \ iBsz(ix,iy,iz) = ix * iy * iz.$$

- ***Constraint conditions***
  1. ***iz must be a multiple of Nc.***
  2. ***ix has to equal nx.***



Step2 *find the optimal Block size (bx, by, bz) and bt*

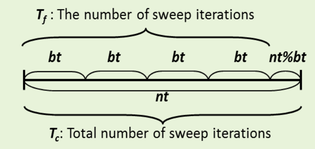
- ***Objective function***

$$\min(Total\_A) = \min\left(\sum_{i}^{T_c} Ai\right) = \min\left(\sum_{i}^{T_c}\sum_{X,Y,Z}^{nx,ny,nz} B(X,Y,Z,i)\right) \quad (2a)$$

$$\sum_{i}^{T_c} Ai = \sum_{X,Y,Z}^{nx,ny,nz} \left(B(X,Y,Z,1) * T_f + B(X,Y,Z,T_c) * (T_c - T_f)\right) \quad (2d)$$

$where\ \ 1 \le i \le T_c, \quad 1 \le X \le ny\ \ and\ \ X\ is\ a\ multiple\ of\ bx,$
$1 \le Y \le ny\ \ and\ \ Y\ is\ a\ multiple\ of\ by,\ 1 \le Z \le nz\ and\ Z\ is\ a\ multiple\ of\ bz$

- ***Constraint conditions***
  1. *bz must be multiple of iz*
  2. *by must be multiple of iy*
  3. *bx must equal nx, bx= nx*
  4. *bt must satisfy the following*
     $2 \le bt \le nt,\ \ bt \le \frac{bx}{2},\ bt \le \frac{by}{2},\ and\ \ bt \le \frac{bz}{2}$
     bt = 2: temporal blocking for minimum size
  5. *bt must satisfy the following*
     Total size of block arrays must not exceed DRAM capacity (Sm)
     *k * Bsz_max * Esz < Sm – Km*
     Bsz_max : The largest Block size, k: constant, Km : constant memory capacity reserved for OS
     $Bsz_{max} = Max_{X,Y,Z}\left(Max\left(Bsz_{w(X,Y,Z,1)}, Bsz_{r(X,Y,Z,1)}\right)\right)$



RunTime Parameter Tuning as a frontend

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, Flash info.

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 L2 and 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

Search Problem Definition

Search of a Globally Optimal set to Minimize the IO traffic between Flash and DRAM

Inputs to the auto-tuning system

1. ***Hardware information (from hwloc)***
   Capacity of each memory layer
   Sc2: level-2 cache, *Sc3*: level-3 cache, *Sm*: Main memory (DRAM) capacity,
   Sf: Flash memory capacity, *Sb:* Flash device block size
   CPU configurations
   Nc: # of cores in a system, *Ns*: # of sockets
2. ***Problem information (User input)***
   Domain data size & Number of iterations for stencil computations
   nx, ny, nz: 3D array, *nt*: (time steps)

Outputs from the auto-tuning system

1. ***An optimal combination of temporal and spatial block size***
   bx, by, bz: spatial blocking size (*Block Array* size), ***bt***: temporal size
   It minimize IO traffic between DRAM and flash memory
   the volume of the *Block array* can be accommodated in DRAM capacity.
2. ***Semi-optimal internal spatial block size combination***
   ix, iy, iz :spatial blocking size (iBlock array)

Premises

*Sf > nx * ny * nz * k * Esz*
(*k* is constant, 2 for double buffer arrays. *Esz* is the data element size in byte)
nx is required to be a multiple of Sb. (tune for asynchronous IO)

Amount of IO traffic for a Block: B(X, Y, Z, i)

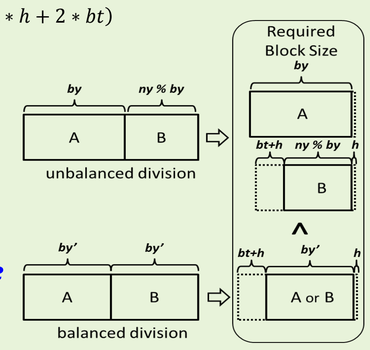
- *Typical block array size for algorithm-1 with redundant calculations*

$$Bsz1\_r = (bx + 2*h + 2*bt) * (by + 2*h + 2*bt) * (bz + 2*h + 2*bt)$$
$$Bsz1\_w = (bx) * (by) * (bz)$$

- *Typical block array size for algorithm-2 without redundant calculations*

$$Bsz2_r = Bsz2_w = (bx + 2*h + bt) * (by + 2*h + bt) * (bz + 2*h + bt)$$

Division scheme and boundary effects in block size



Blk-Tune searches an optimal spatial & temporal blocking parameter set for particular platforms and application settings

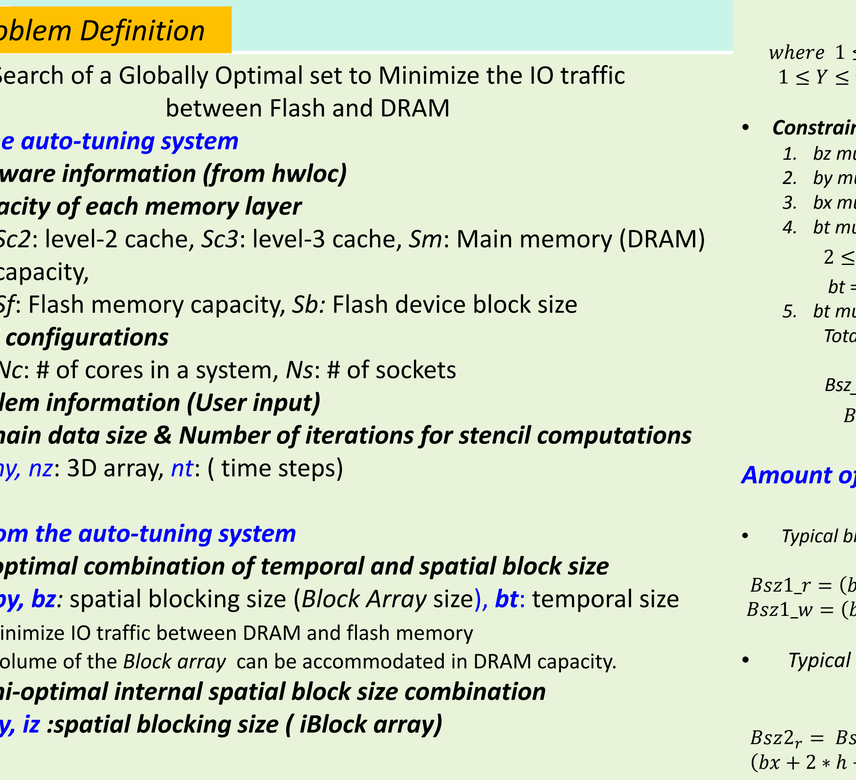
Table I. Parameter examples selected by Blk-Tune

Problem size	nx	ny	nz	nt		
128GiB-problem	2048	2048	2048	500		
Server (DRAM, cores)	bx, ix	by	bz	bt	iy	iz
crest6 (128GiB, 20)	2048	1025	2048	500	1	160
crest4 (64GIB, 16)	2048	1025	1280	500	1	128
crest0 (32GiB, 8)	2048	513	1152	250	1	64

Problem size	nx	ny	nz	nt		
256GiB-problem	2048	2048	4096	1000		
Server (DRAM, cores)	bx, ix	by	bz	bt	iy	iz
crest6 (128GiB, 20)	2048	1025	2240	500	1	160
crest4 (64GIB, 16)	2048	683	1536	334	1	128
crest0 (32GiB, 8)	2048	513	832	250	1	64

Table II. Platforms

server	L1 cache (KiB)	L2 cache (KiB)	L3 cache (MiB)	Phys Mem (GiB)	Flash Mem (TiB)	CPU Xeon E5, (GiHz)	Total cores	socket	cores/ socket
crest0	32	256	20	32	1.2	2650, (2)	8	1	8
crest4	32	256	20	64	0.785	2687W, (3.1)	16	2	8
crest6	32	256	25	128	1	2687W v3,(3.1)	20	2	10



Manual selection vs. *Blk-Tune* for various-size problems on crest6 server
for two algorithms, aio3y and aio5y, temporal blocking without redundant calculations.

REFERENCES
1. H. Midorikawa, et.al, "An Evaluation of the Potential of Flash SSD as Large and Slow Memory for Stencil Computations," Proc. IEEE Int. Conf. on High Performance Computing and Simulation IEEE-HPCS2014, pp. 268-277
2. H. Midorikawa et.al, "Locality-Aware Stencil Computations using Flash SSDs as Main Memory Extension," Proc. IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing CCGrid2015, pp. 1163-1168
3. Portable Hardware Locality (hwloc). https://www.open-mpi.org/projects/hwloc/
4. H. Midorikawa: "Blk-Tune: Blocking Parameter Auto-Tuning to Minimize Input-Output Traffic for Flash-based Out-of-Core Stencil Computations", IEEE Int. Parallel and Distributed Process. Symp (IPDPS) Workshop iWAPT , May 2016