**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

- For Offline Parameter Search and Analysis
- Hardware Information
- Search Problem Definition
- Tuning for Numa systems
- For Online Runtime Auto Tuning

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

**RunTime Parameter Tuning as a forefront**

1. User command parameters
   - domain size(nx,ny,nz), Time steps(nt), Flash device path
   - %_tuneTnp -n 4094 4096 2048 -t 11000 -dev/dev: 
     - _auto* start
     - (bx,by,bz), bt = (4094,1024,512), 125
   - _auto* end
2. Get hardware information
   - device capacity, device block size, DRAM size, L3 cache size, # of sockets, # of CPU cores, Flash SSD
3. Calculate optimal blocking sizes by adjusting to underlying hardware
   - Efficient spatial block size and shape, temporal block size
   - Efficient spatial inner block shape and size (optimal (ix,iy,iz)
   - Typical block array size for algorithm-1 with redundant calculations
   - Typical block array size for algorithm-2, aio3y and aio5y, temporal blocking without redundant calculations
4. numa-aware computing
   - Blocks are divided into sub-block areas according to # of CPU sockets
   - memory-bound and qup-bound between each sub-block to each local socket
   - locality-aware sub-block computation on each CPU socket using local cores

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

**Table I. Parameter examples selected by Blk-Tune**

<table>
<thead>
<tr>
<th>Problem size</th>
<th>nx</th>
<th>ny</th>
<th>nz</th>
<th>nt</th>
</tr>
</thead>
<tbody>
<tr>
<td>stencil0</td>
<td>32</td>
<td>128</td>
<td>256</td>
<td>20</td>
</tr>
<tr>
<td>stencil4</td>
<td>64</td>
<td>256</td>
<td>1024</td>
<td>200</td>
</tr>
<tr>
<td>stencil6</td>
<td>96</td>
<td>256</td>
<td>2048</td>
<td>500</td>
</tr>
</tbody>
</table>

**Table II. Platforms**

<table>
<thead>
<tr>
<th>Platform</th>
<th>CPU (GHz)</th>
<th>Mem (KiB)</th>
<th>Flash (KiB)</th>
<th>L3 Cache (KiB)</th>
<th>L2 Cache (KiB)</th>
<th>Socket</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>2.4 GHz</td>
<td>16 GB</td>
<td>32 GB</td>
<td>8 GB</td>
<td>16 GB</td>
<td>4</td>
</tr>
<tr>
<td>server2</td>
<td>2.4 GHz</td>
<td>16 GB</td>
<td>32 GB</td>
<td>8 GB</td>
<td>16 GB</td>
<td>4</td>
</tr>
</tbody>
</table>

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
      - S3: Level-2 cache, S2: Level-3 cache, S1: Main memory (DRAM) capacity
    - S4: Flash memory capacity, S5: Flash device block size
    - 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 minimizes IO traffic between Flash and DRAM
    - 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 (Block array)
- Premises
  - S5 > nx * ny * nz: x * y * z
    - (ix is constant, 2 for double buffer arrays, Ex is the data element size in byte)
    - nx is required to be a multiple of ix (Tune for asynchronous IO)

**REFERENCES**