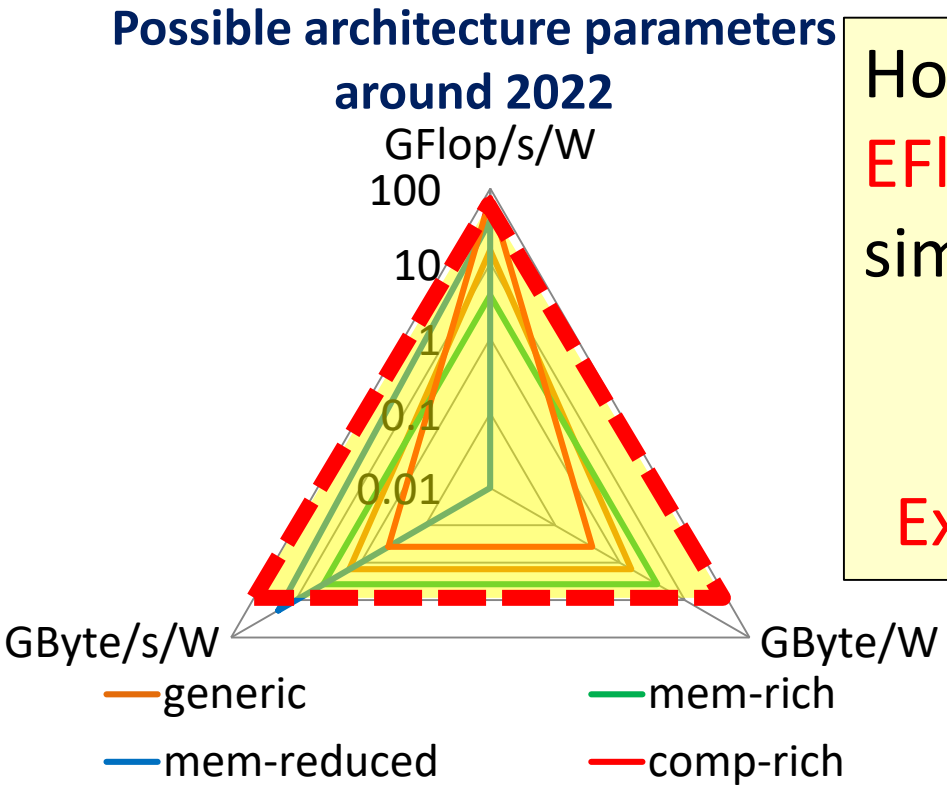
Software Technology that Deals with Deeper Memory Hierarchy in Post-petascale Era

PI: Toshio Endo (Tokyo Tech)
Co-PI: Hiroko Midorikawa (Seikei Univ)

Yukinori Sato (Tokyo Tect)

Motivation: Memory Wall Problem in Post-Peta Era

- HPC community is going to Post-peta/Exascale era
- "Memory wall problem" introduces trade-off among "Flop/s", "Byte/s" and "Byte"(problem size)



How can we reach (virtually) EFlop/s, EByte/s and EByte simultaneously?

Realizing Extreme Fast&Big Simulations

MemoryCREST: Overview

Goal: To achieve extremely high-speed&large simulations by the co-design approach

Co-design

Algorithms with locality improvement

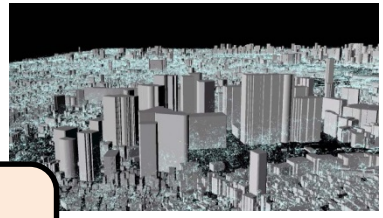
+

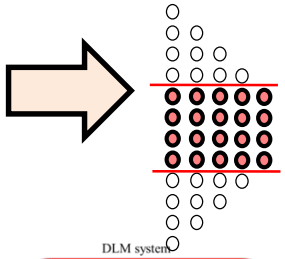
Memory Profiling

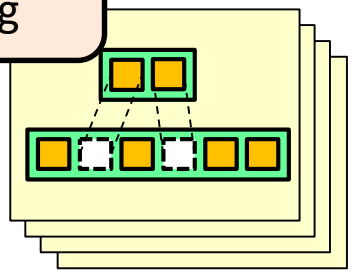
System software for mem hierarchy mgmt

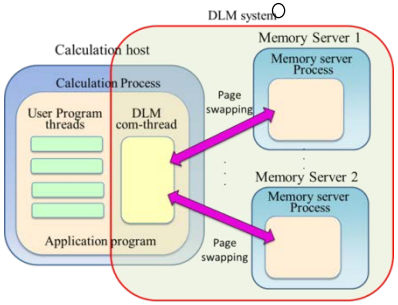
+

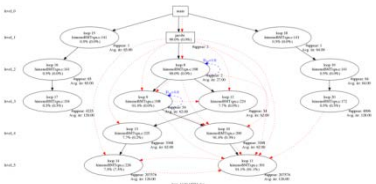
HPC Architecture with hybrid memory devices

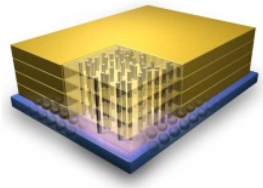




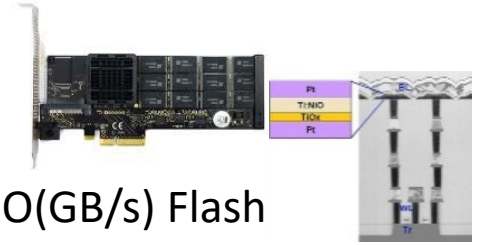






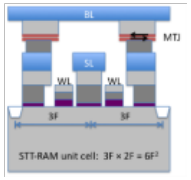


HMC, HBM



O(GB/s) Flash

NVDIMM



PCM, STT-MRAM…

Algorithms with locality improvement

\+

Memory Profiling

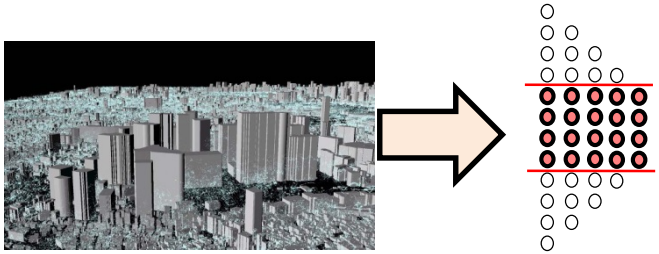
System software for mem hierarchy mgmt

\+

HPC Architecture with hybrid memory devices

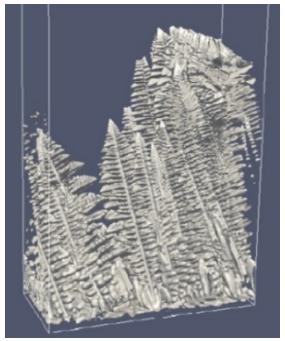
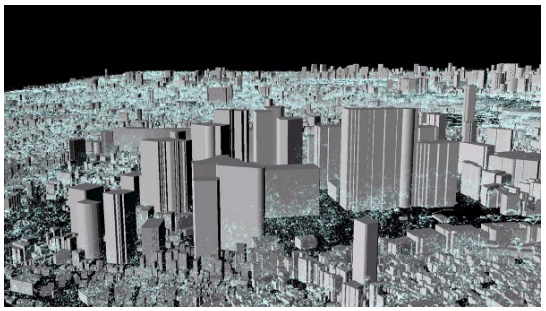
Algorithm layer

- Stencil kernels w/ temporal blocking

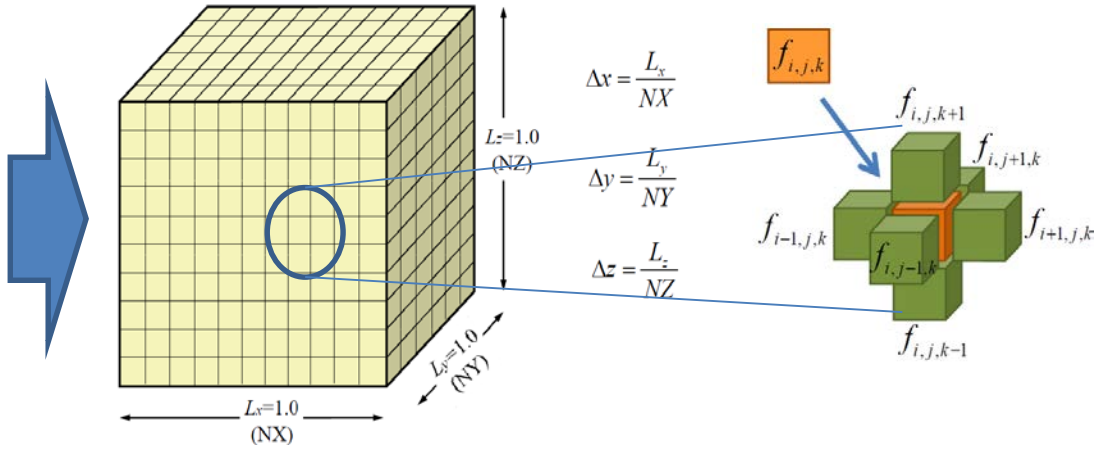


Stencil Computations

Important kernels for various simulations: CFD, Materials…

Figures by Prof. T. Aoki (Tokyo Tech)



Naïve implementations lacks memory access locality

→ How can we harness memory hierarchy for fast&large simulations?

Locality Improvement in GPGPU Stencils

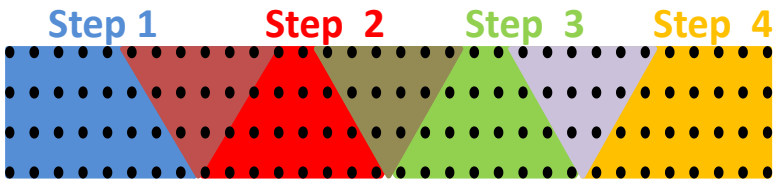
Endo

[IEEE Cluster 13]

Blocking/tiling techniques are good, but “spatial” blocking is still insufficient

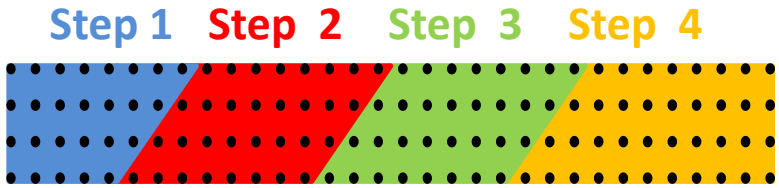
⇒ Temporal blocking (TB) is a key technology [Wolf 91] [Datta 08]…

w/ Temporal blocking (3 steps)



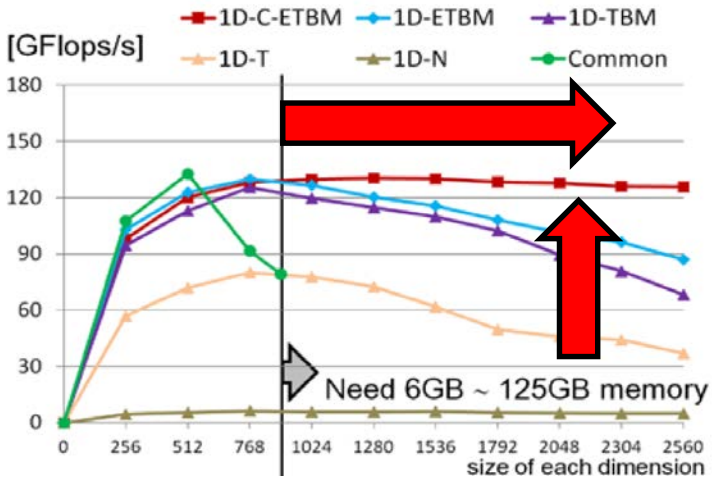
Simulated time

Optimized temporal blocking



- TB is originally for cache usage improvement. We adopted it to exceed GPU memory capacity

7P-stencil on a K20X GPU



With optimized TB, 20x larger domain size is successfully computed with very little overhead

6GB (GPU mem size) → 125GB

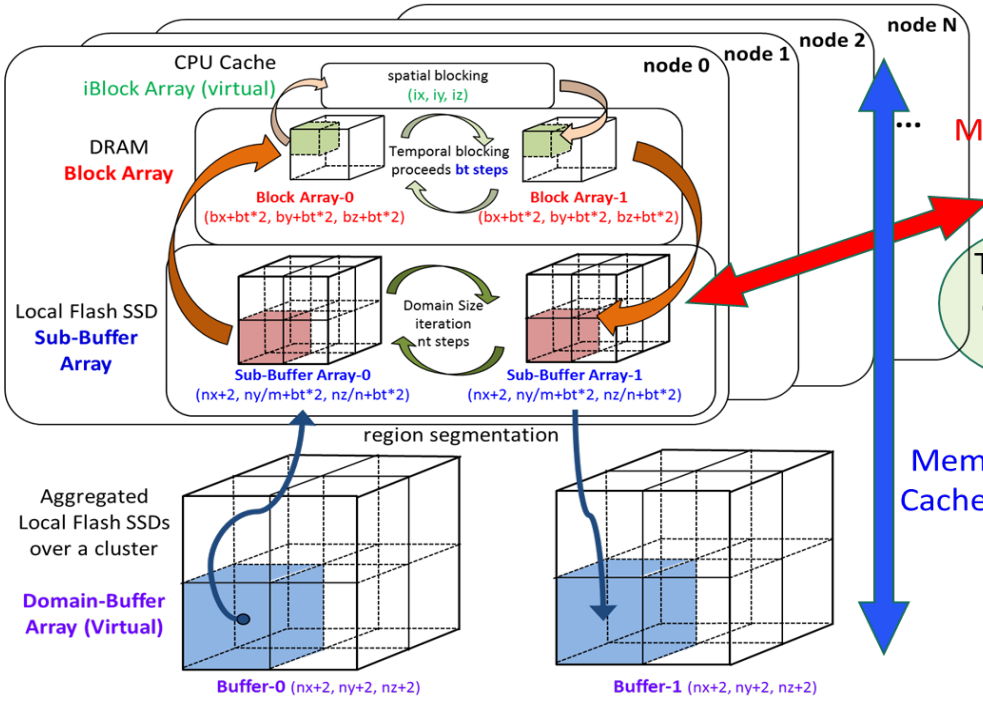
Horizontal and Vertical Memory Extensions for Large Data Applications

Midori

An Out-of-core Stencil Algorithm using Cache, DRAM, and Flash SSDs

Large Problem Size beyond the Total Capacity of DRAM of Nodes in a Cluster

Maximum Problem Size = Local SSD Capacity x Num of Nodes

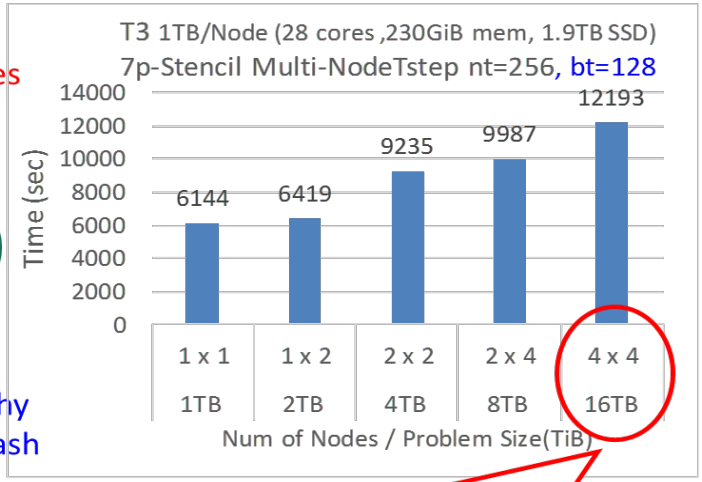


Horizontal
Multiple nodes
with MPI

Temporal
& Spatial
Blocking

Vertical
Memory Hierarchy
Cache+DRAM+Flash

Distributed SSDs in a cluster are used for out-of-core stencil computations



Tsubame3 : (256GB-DRAM, 2TB-Flash) / node
Only 16 nodes are sufficient for 16TB-Stencil problem.
When using only DRAM without our algorithm,
128 nodes are required for the 16TB Stencil problem

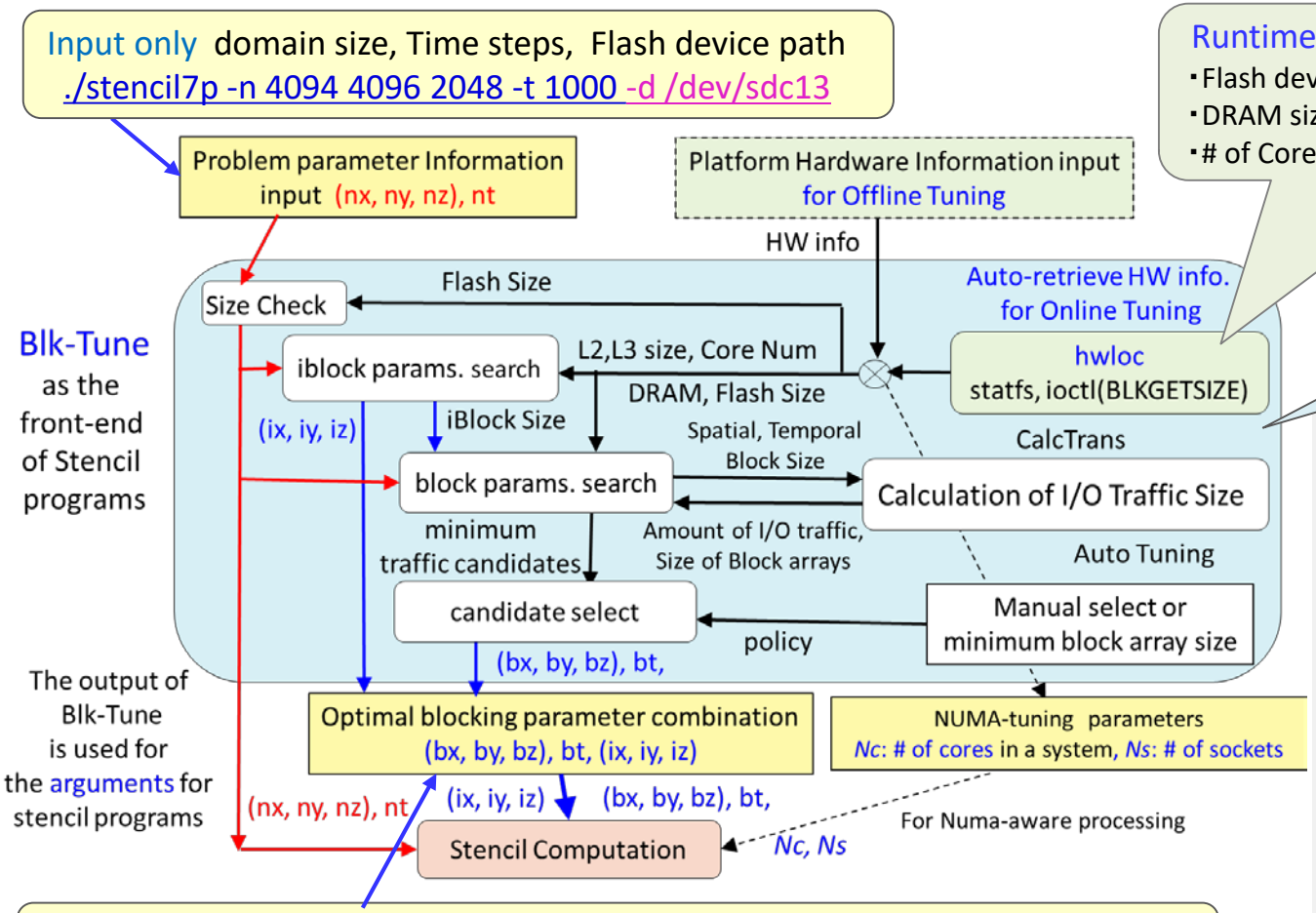
Tsubame3 : 1TiB-array/Node
(28-core, 120GiB-DRAM, 1.9TB-SSD)

Blk-Tune

Midori

Just-in-Time Automatic Blocking Size Setting to increase data access locality for Flash-based Stencil Computing

In runtime, the Blk-Tune selects the optimal set of spatial and temporal blocking sizes for given platforms and problem parameters to minimize the amount of I/O traffic to Flash



Advantages

・No preliminary program executions to gather information using various parameter combinations, which are common in ordinary auto-tune systems.

・More dynamic than compiler-based tuning systems that use only static information of programs without any input parameters or platform information.

Towards Automatic Temporal Blocking by Extending Polly/LLVM [LLVM-HPC'17]

Endo

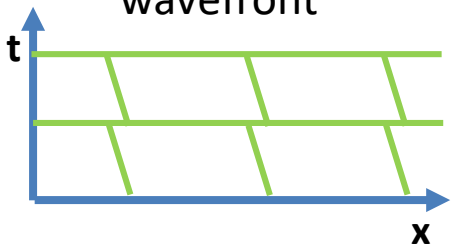
w/ Matsuoka Lab

- Temporal blocking is expressed as loop transformation

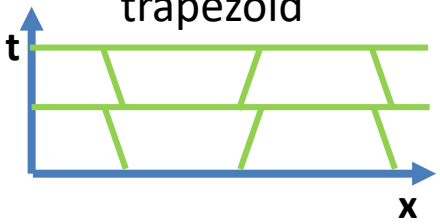
→ in order to reduce programming costs, polyhedral compilers are promising

  - Pluto, Polly on LLVM

wavefront



trapezoid



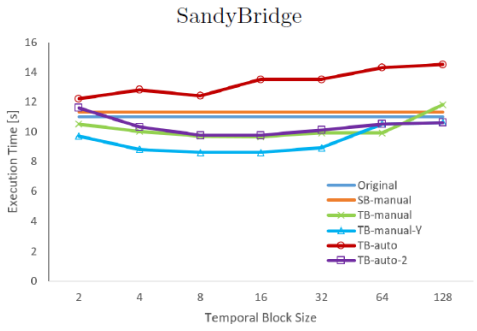
But … temporal blocking introduces "skewed" block shapes, which are not supported in all those compilers

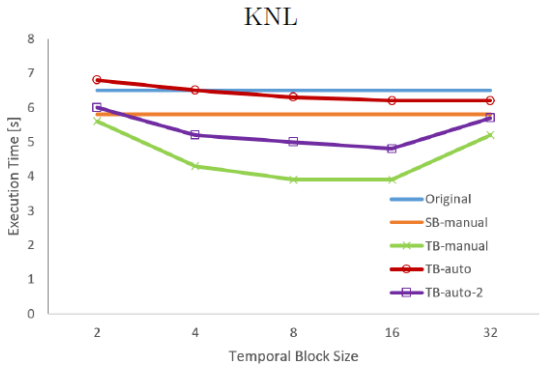
→ We extended Polly/LLVM!

Our new tool creates a loop scheduling that expresses TB. 1D 3Point case is:

[n, timestep] -> { S1[t, x] -> [T, 0, bx, t, 0, x] : (T % 13 = 0 and T <= t < T + 13 and ((x + (1 * (12 – (t - T)))) % 624 < 312 + 2 * (1 * (12 - (i0 - T))) and bx = floor((x + (1 * (12 - (t - T)))) / 624))) and 2*floor((t)/2) = t and 2 <= t < timestep ;
S1[t, x] -> [T, 1, bx, t, 0, x] : (T % 13 = 0 and T <= t < T + 13 and ((x + (1 * (12 - (t - T)))) % 624 >= 312 + 2 * (1 * (12 - (t - T))) and bx = floor((x + (1 * (12 - (t - T)))) / 624))) and 2*floor((t)/2) = t and 2 <= t < timestep}

Performance of 2D 5point Stencil





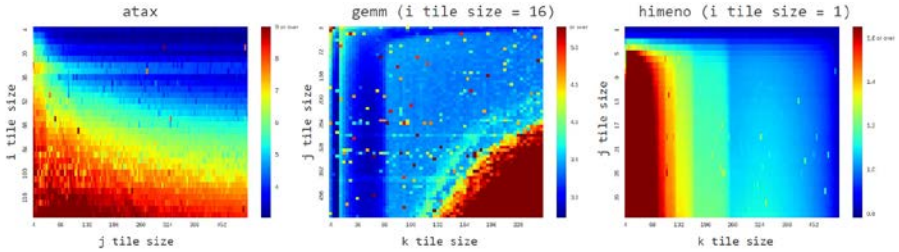
- codes with TB are successfully generated
- Similar performance with hand-tuned code

Future: Transforming real apps!!

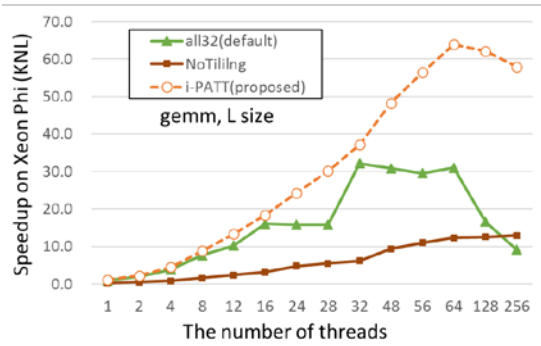
PATT: Polyhedral compilation based AuTo Tile size optimizer

Sato

Loop tiling size selection is still a open problem because it depends on application's memory access patterns and the underlying hierarchical memories on each platform

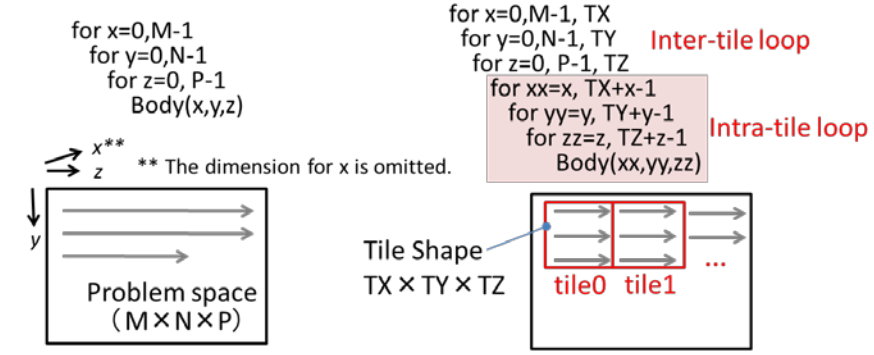


Performance widely varies with loop tile size



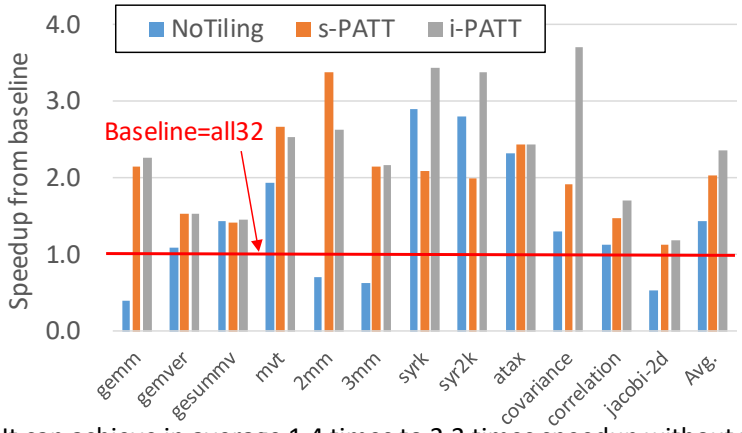
As a preliminary evaluation on many core CPUs (KNL), we observed the performance degradation due to scalability issues

We propose an autotuning mechanism capable of load balancing among threads running on a many core CPU



(a) A typical space traversal

(b) A space traversal with loop tiling



It can achieve in average 1.4 times to 2.3 times speedup without the case of no tile size consideration (32x32x32, Polly's default tile size)

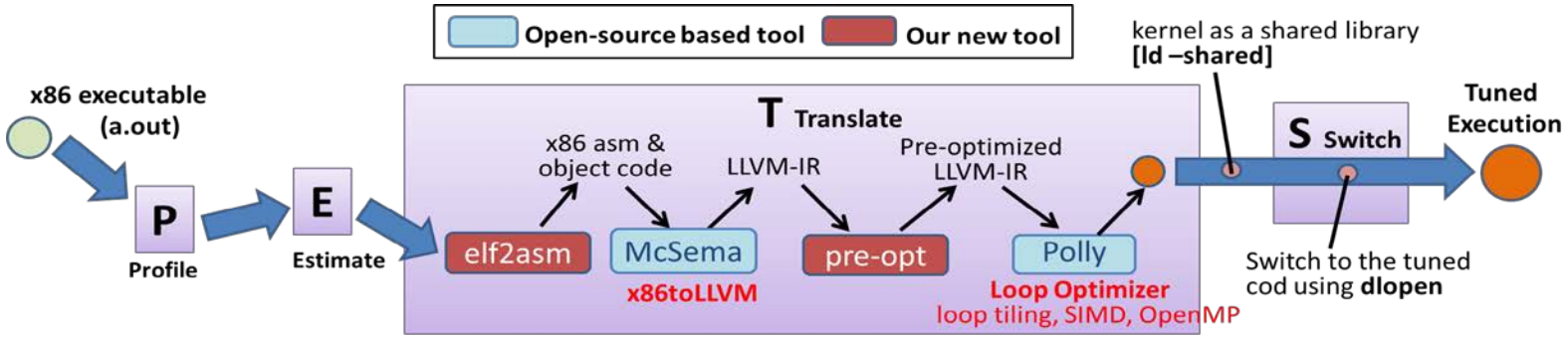
PATT (Polyhedral compilation based AuTo Tile size optimizer): Automate tiling using open source Polyhedral compiler and search the optimal parameters for tile size coupling with autotuning technique

Transparent performance tuning on ExanaDBT

Sato

ExanaDBT: A dynamic binary optimizer based on Polyhedral model

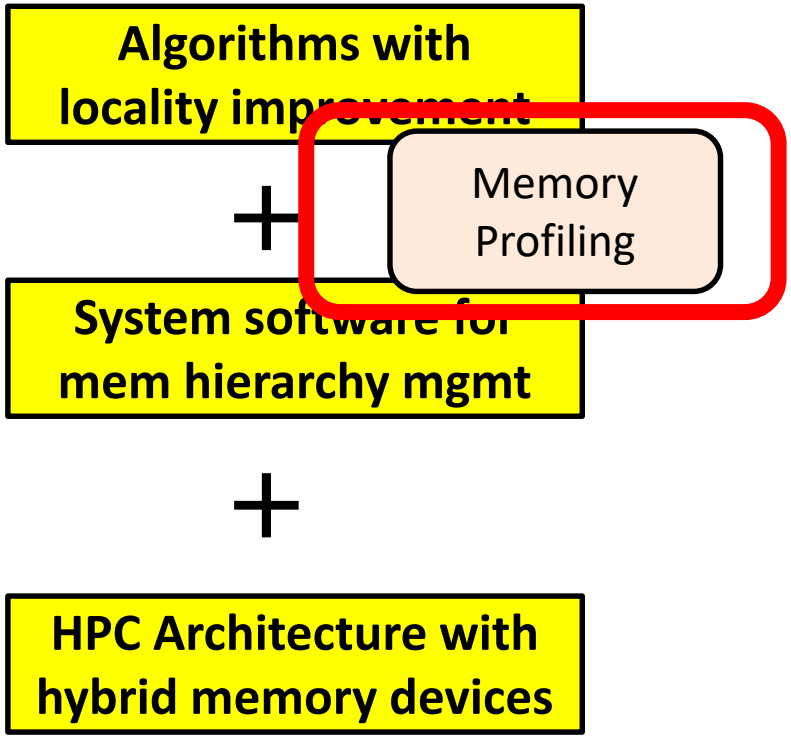
- It transparently optimizes executable binary code at runtime
- Implemented based on LLVM compiler toolchains
- Just connecting these tools, polyhedral optimizer always failed due to the structural gaps among x86 ISA and LLVM IR
- We investigate the reasons behind them and attempt to perform transparent polyhedral optimization including loop tiling, vectorization and parallelization



We have developed a toolchain that lifts up the binaries to the one capable of polyhedral optimization

- From the results, we find that ExanaDBT successfully performs dynamic optimization
- It contribute to 3.2x in 1-thread and 11.9x speedup in 16-thread execution in average from unoptimized serial code.

[CF'17] Yukinori Sato, Tomoya Yuki and Toshio Endo, "ExanaDBT: A Dynamic Compilation System for Transparent Polyhedral Optimizations at Runtime", ACM International Conference on Computing Frontiers 2017. 10 pages.



Exana series

Our toolchain for performance tuning

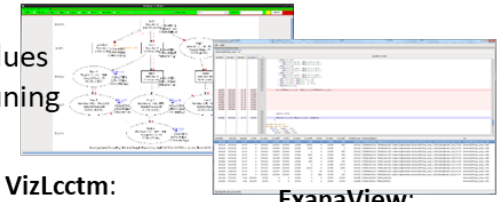
Sato

Source code | Executable code | Profiling | Performance tuning

Exana

Memory locality profiler

Provide hints and clues for performance tuning



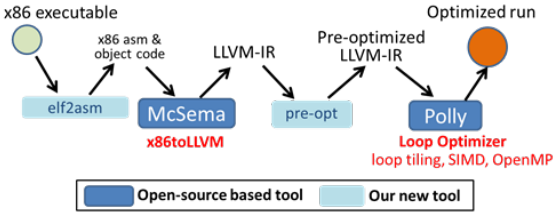
VizLcctm: Visualizing LCCT+M

ExanaView: Diagnosing bottlenecks of source code

ExanaDBT

Transparent code translator

Dynamic binary translation to locality optimized code [ACM CF'17]



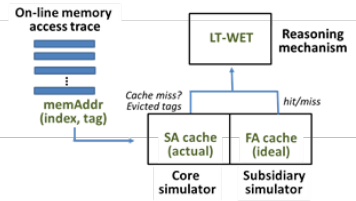
Feedback to source code

Transparent tuning

Exana-C2Sim

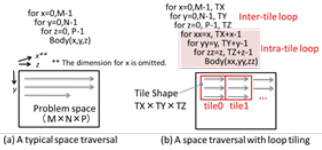
Cache-line conflict detector

Detect line conflicts and analyze root cases [EuroPar'17]



PATT

Automatic loop tile size optimizer



Advanced locality optimization for hierarchical memory using a polyhedral compiler

Memory locality analysis using Exana tool

Sato

Exana: A memory locality profiler for assisting performance tuning

How to use : Put options before target application's command

% Exana [option] -- ./a.out

Outputs are exana.out and result files for each analysis

	Options	Result file
Loop-call nest detection [CF'11]	-mode LCCT	lcct.dat
Memory dependence analysis [IISWC'12]	-mode LCCT+M	lcctm.dat
Memory access pattern analysis [WANC'14]	-mempat 1	mempat.dat
Cache-line conflict simulator [EuroPar'17]	-cacheSim 1	cacheSim.dat
Working set analysis [SEPS'15]	-workingSetAna 1	lcct.dat

For MPI programs

% mpirun –np 16 Exana –cacheSim 1 -- ./openmx Methane.dat

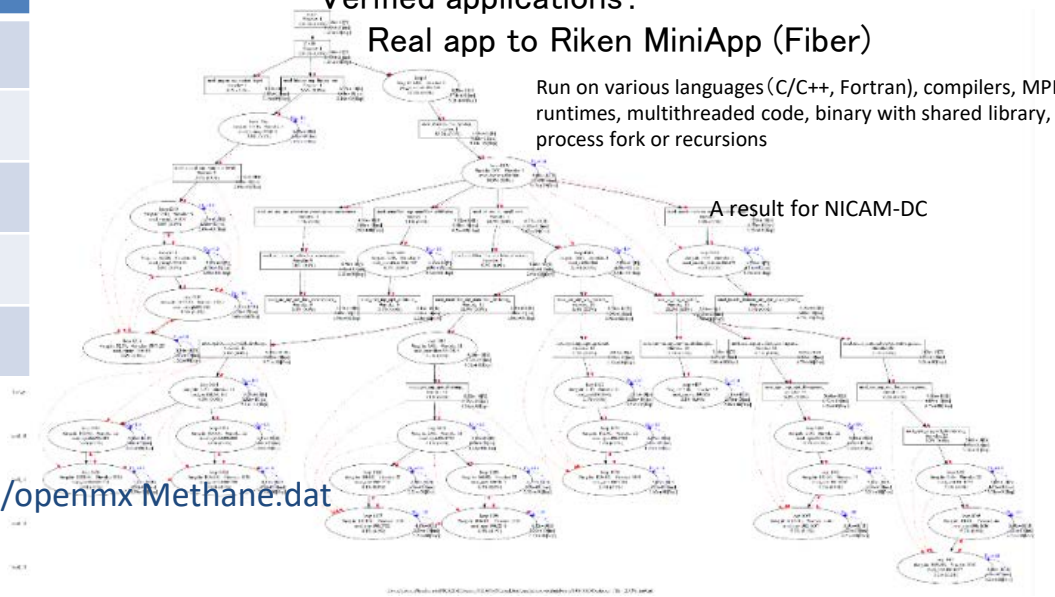
Verification for Exana tool set

Custom HPC	TSUBAME 3.0
Custom HPC	SGI Altix UV
Custom HPC	Cray XC30
Accelerators	Xeon Phi (KNC, KNL)
Generic	x86 Linux (CentOS)

Verified applications :

Real app to Riken MiniApp (Fiber)

Run on various languages（C/C++, Fortran), compilers, MPI runtimes, multithreaded code, binary with shared library, process fork or recursions

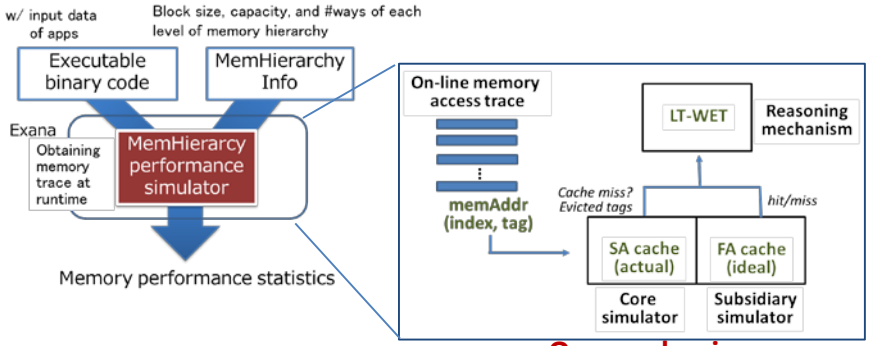


A result for NICAM-DC

Yukinori Sato, Shimpei Sato and Toshio Endo. Exana: An Execution-driven Application Analysis Tool for Assisting Productive Performance Tuning. The Second Workshop on Software Engineering for Parallel Systems (SEPS), co-located with SPLASH 2015.

Exana-C2Sim: A cache-line conflict detector and its application to performance tuning

Sato



Our mechanism

Ratio of cache conflicts and their source of occurrences

We can detect memory object level locations of conflicts

Also, we can link them to the locations in source code

```
Memory object-relative view:
malloc[#3] total=...
  --> malloc[#3] cnt= 308503621, 17620466, 0, originPC= 4008a2
  --> malloc[#11] cnt= 9951731, 107283, 0, originPC= 400898
  --> malloc[#2] ...
  --> Stack(7ff...) ...
malloc[#1] total=106...
  --> malloc[#3] cnt= 10603920, 0, 0, originPC= 4008a2
malloc[#2] total=115953 conflictMissPC= 400888 4008c8
  --> malloc[#3] cnt= 0, 115953, 0, originPC= 4008a2

Reason classification view:
             sum          inter-array  intra-array  scalar  unknown
#conflict    347018930:   20894839     326124087    4       0
Ratio                     6.02%        93.98%       0.00%   0.00%
```

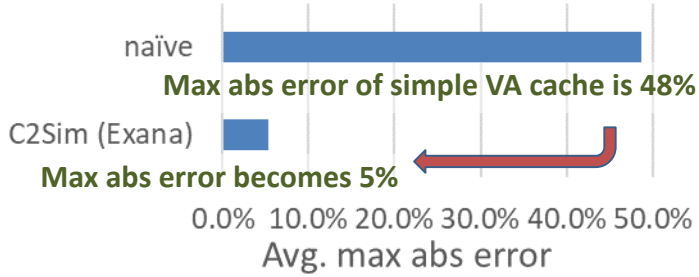
We can classify the sources into intra-array or inter-array

We develop a special cache simulator for diagnosing occurrences of conflict misses and a simple workflow to get rid of them

- **Concurrent dual cache simulation:** Detecting conflict misses using both of the ideal Fully Associative cache and the actual Set Associative cache
- **Reasoning mechanism:** Revealing the source of conflict misses together with memory-object relative profiling
- **Advanced cache modeling:** Modeling slice structures in L3 cache and physical address translation

Code tuning strategy for avoiding cache conflicts

	Tuning strategy	When
Opt.1	Intra-array padding insertion	To resolve intra-array conflicts
Opt.2	Use of hugetlbfs (2MB page)	To resolve conflicts in L2/L3 cache
Opt.3	Inter-array padding insertion	To resolve inter-array conflicts



Effects of advanced cache modeling

Cache optimizations in doitgen

	Speedup
Original	1.00
Opt.1	1.19
Opt.1+Opt.2	1.21
Opt.2	1.02

Scalability for parallel threads and sensitivity to HW prefetch in Himeno

	HW PF	Speedup (**)
1 thread	off	1.62
	on	1.75
16 threads	off	1.50
	on	1.70

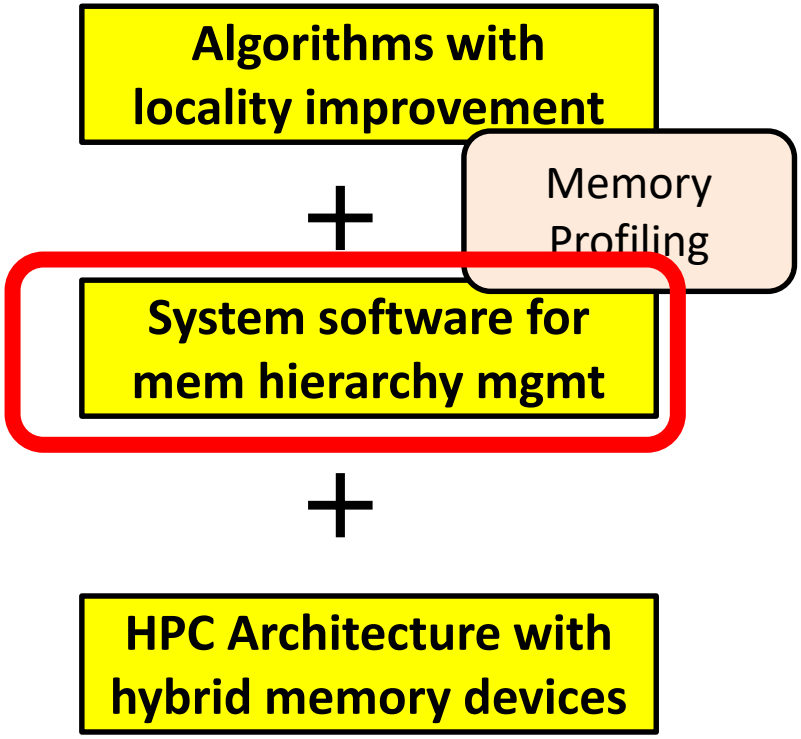
(**) **Opt.3** is performed

Cache optimizations in 3D-FDTD

	Speedup
Original	1.00
Opt.3	1.32

Based on the results of reasoning mechanism, we can productively find the locations for refactoring and seamlessly achieve performance gain

[EuroPar'17] Yukinori Sato and Toshio Endo. An Accurate Simulator of Cache-line Conflicts to Exploit the Underlying Cache Performance. The 22nd International Conference on Parallel and Distributed Computing, Euro-Par 2017.



System software/Middleware

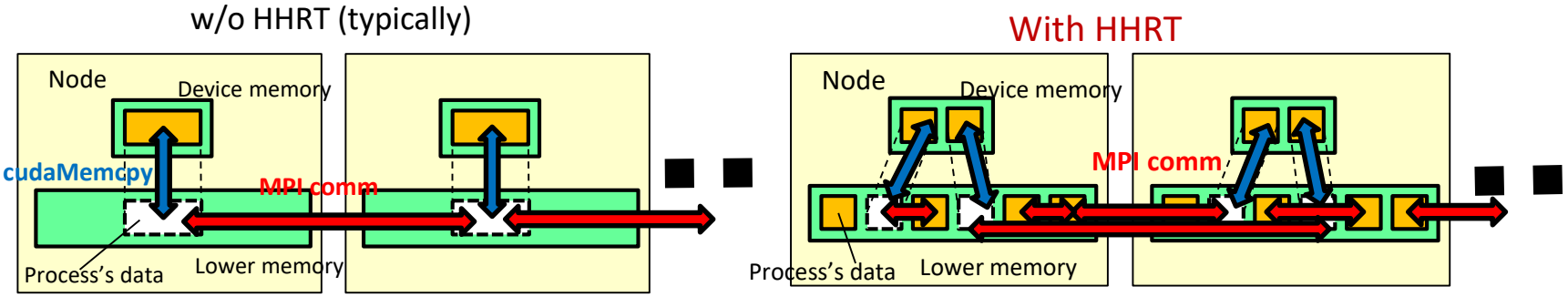
- HHRT (GPU⇔Host⇔SSD)
- mDLM
- mSMS
- vGASNet

HHRT for CUDA/MPI Apps

[Cluster '14]

Endo

- Hybrid Hierarchical RunTime: A wrapper library for MPI + CUDA
  - It provides data swapping facility in memory hierarchy → Expands data size visible to applications
  - *github.com/toshioendo/hhrt*



- Execution model : Several (*n*) MPI processes share a single GPU
- *m*, # of processes that can run simultaneously, is smaller than *n*
  - Swapping is done per process (not per page)
- Process's data are swapped out to lower memory (host memory or Flash SSD)

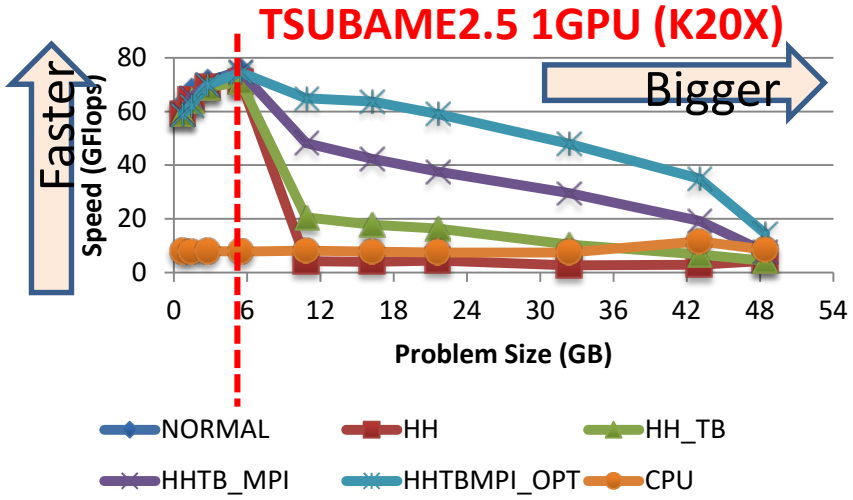
HHRT does swapping, but does not locality improvement
→ Programmers still need to implement locality improvement

Expanding Domain Sizes of Real-World Stencil Applications [IEEE ICPADS '15]

Endo
w/ MaruyamaT

A city wind simulator based on Lattice-Boltzmann method
by Onodera, Aoki

- A stencil application written in MPI+CUDA
- The simulation domains are allocated on GPU memory

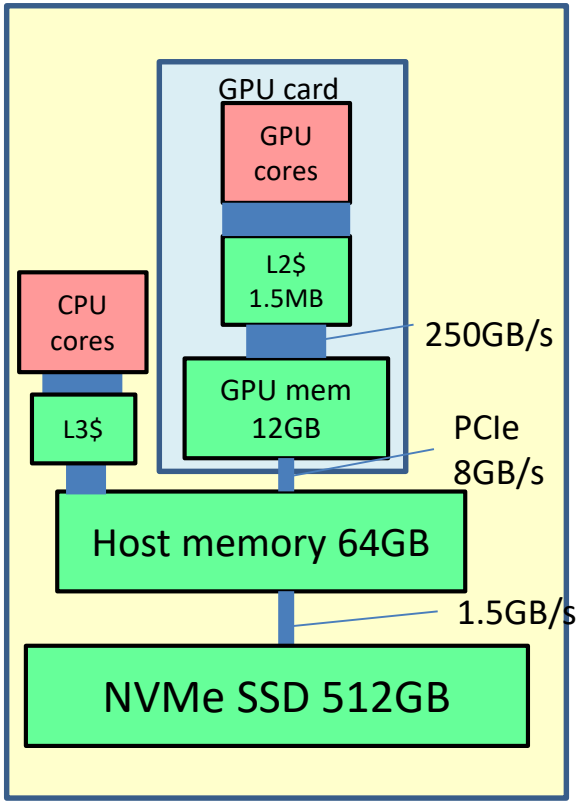


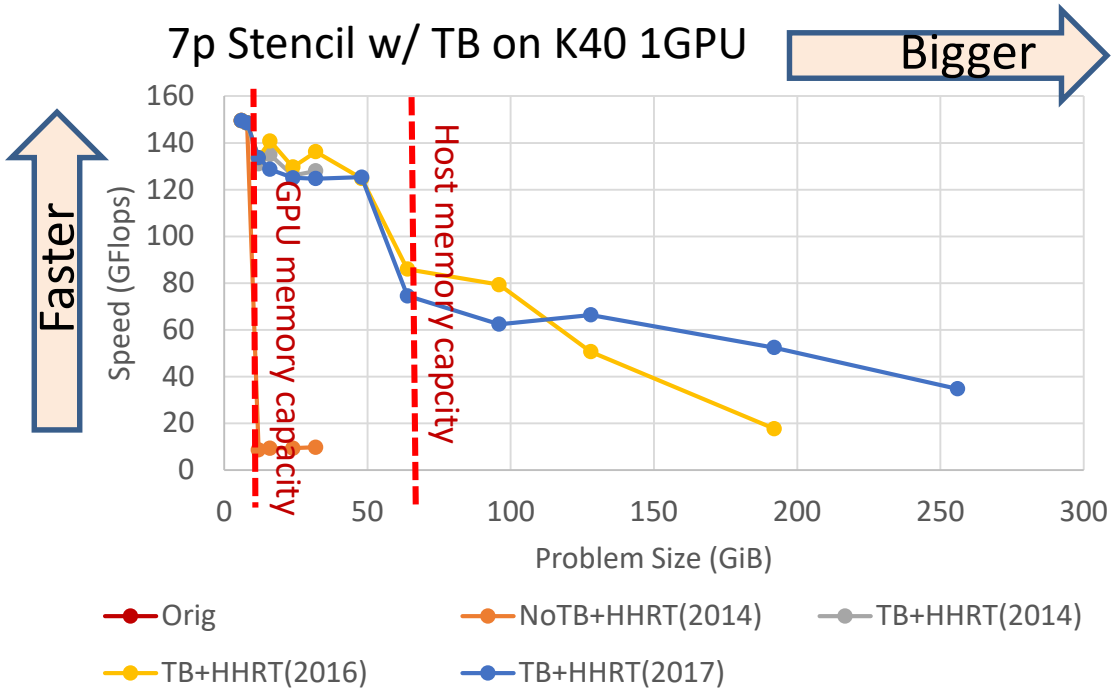
Fast &Large simulations are achieved by the co-design approach!

- *GPU device memory (fast but small) and host memory (large but slow) are automatically harnessed on top of HHRT middleware*
- *TBD: We see still overhead with large domains*

Harnessing 3-Tier memory on HHRT: GPU⇔Host⇔SSD [Cluster '16]

Endo





20x larger stencil domains can be computed

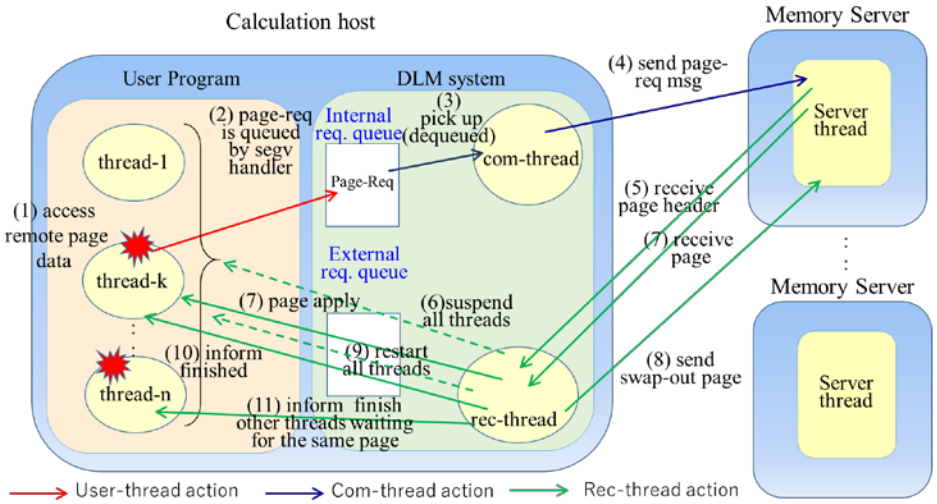
TBD: ~50% overhead with >64GB should be mitigated

mDLM (Distributed Large Memory)

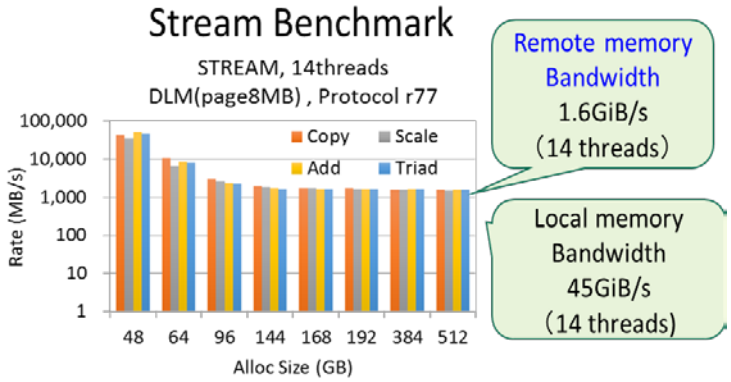
Midori

User-level remote memory paging for multi-thread programs

Page-Swap Protocol in mDLM

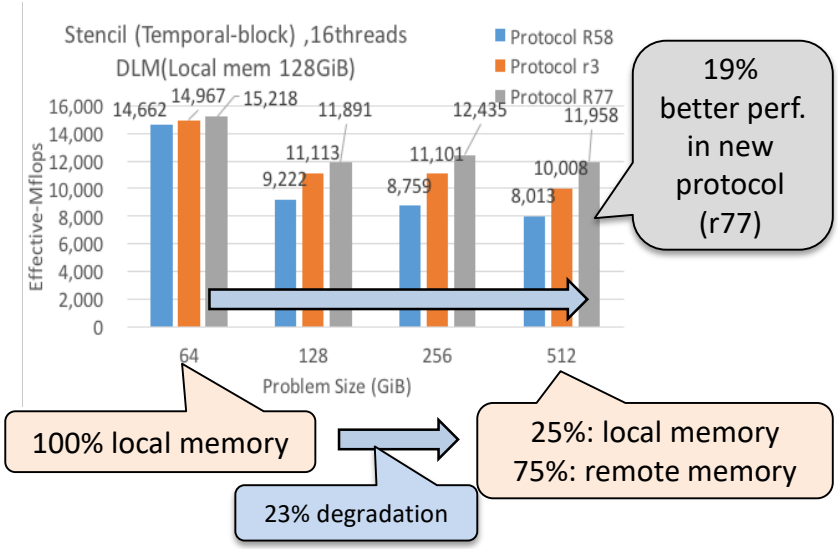


Stream Benchmark



7-point temporal-blocking Stencil Computing

(128GiB mem/node, problem size : 64GiB – 512GiB)



Programming Interface for mDLM

Dynamic data allocation
for remote paging
by changing malloc with dlm_alloc

```
int main(int argc, char *argv[] )
{
  dlm_startup(&argc, &argv);
  array = (int*) dlm_alloc(sizeof(int) * NUM);
#pragma omp parallel for
  for ( i = 0; i < NUM; i++ ) {
        array[i] = i;     .... Multi-thread processing
  }
  dlm_shutdown();
}
```

Static array declaration
for remote paging
by DLM-C Translator

```
#define N 10000
dlm int a[N][N];
int main (int argc, char *argv[])
{
  int i, j;
#pragma omp parallel for private(j)
  for ( i = 0; i < N; i++)
    for ( j = 0; j < N; j++) {
      a[i][j] = i*j;  ... Multi-thread processing
    }
}
```

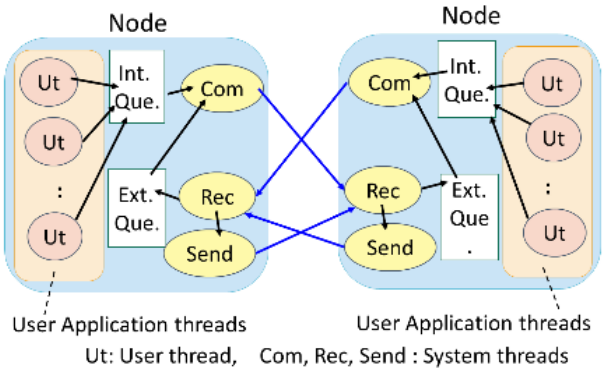
dlm array declaration

OpenMP and pthread programming are available

mSMS (Distributed Shared Memory) for multi-node & multi-thread parallel programs

Midori

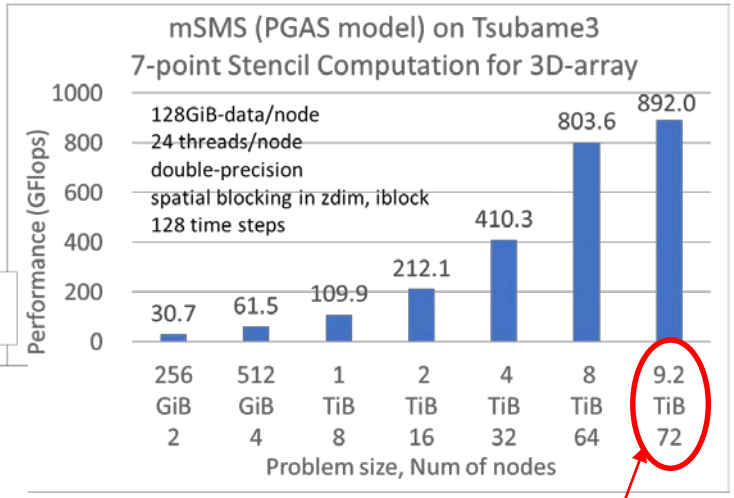
Communications in mSMS system threads



Preliminary experiments in Tsubame3

7-point Simple Spatial-blocking Stencil Computing

(128GiB data/node, 16-thread/node, 2 – 72 nodes
double precision, problem size : 256GiB – 9.2TiB)



9.2 TiB-problem on 72 nodes achieves 892 GFlops, which is expected more when using a temporal-blocking algorithm

Large size problems can be easily implemented and executed on a cluster with highly productive programming environment

Programming Interface for mSMS

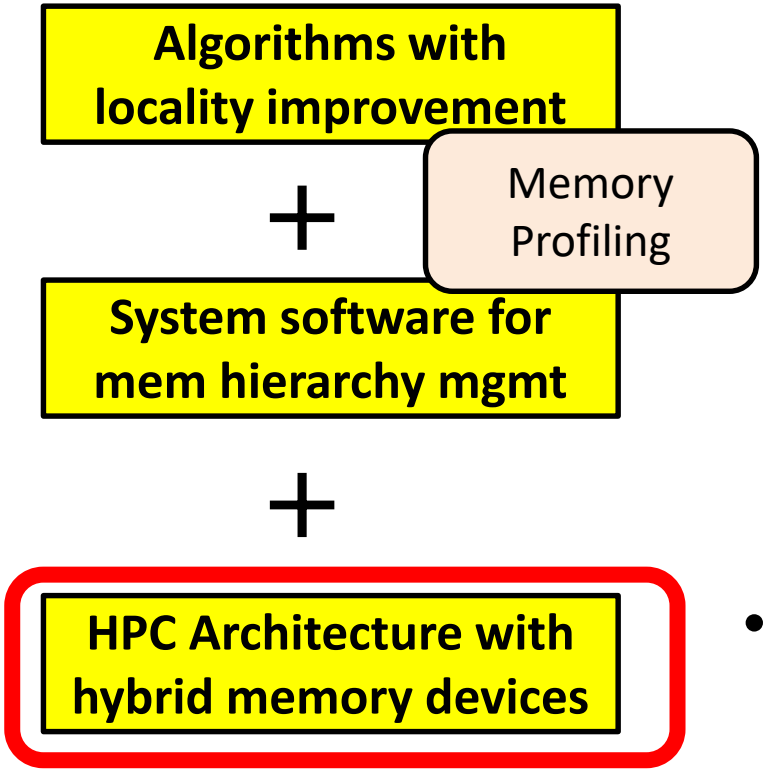
Dynamic data allocation by changing malloc with sms_alloc

```
int main( )
{
   sms_startup(&argc, &argv);
   // data allocation on a specific node
   array = (int*) sms_alloc(sizeof(int) * N, node);
  //or distributed data allocation on some nodes
   array = (int*) sms_mapalloc( dim, div, .....);

if (sms_pid == 0 ){       //   the first node
 #pragma omp parallel for
   for ( i = 0; i < N/sms_nproc; i++ ) {
        array[i] = i;     .... multi-thread processing
  }
}
else if (sms_pid == sms_nproc-1) //the last node
{   :
 }
 else{  // other nodes
   :
 }
 sms_shutdown();
}
```

Distributed static array declaration by MpC Translator

```
shared int a[NZ][NY][NX]::[NPROCS][1][1](0,NPROCS);
int main (int argc, char *argv[])
{
  int i, j, k;
  int size = NZ / NPROCS;
  int st = MYPID * size, en = (MYPID+1)*size;

  mpc_init ();
  #pragma omp parallel for private( j,k)
   for ( i = st; i< en; i++) //node parallel in z-dimension
   {
    for ( j = 0; j < NY; j++)
      for ( k = 0; k < NX; k++)
          a[i][j][k] =  .... multi-thread processing
   }
   mpc_barrier();
   mpc_exit();
}
```

OpenMP and pthread programming are available

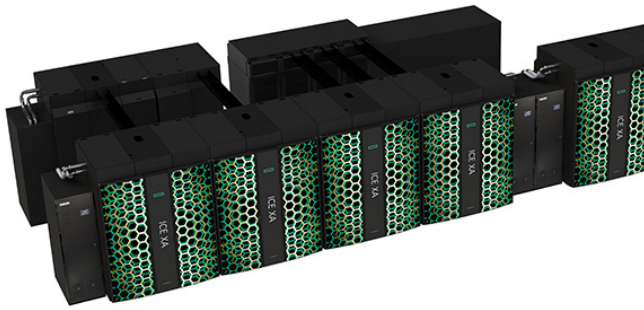


- Feedback to design of TSUBAME3 supercomputer

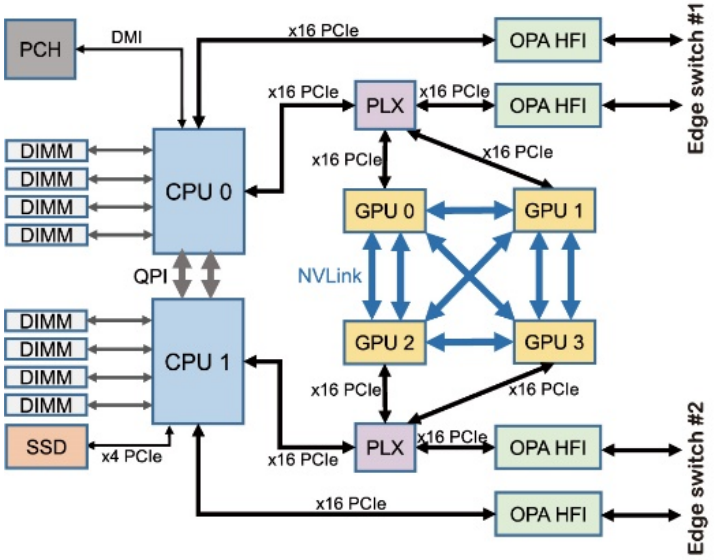
Feedback of Results of Projects to New Supercomputer, TSUBAME3.0

Endo

- Operation of Tokyo Tech TSUBAME3.0 started in Aug 2017
  - 12 PFlops (DP) computation, 16PByte storage
  - 3-Tier memory hierarchy is expected to realize fast&large simulations



A TSUBAME3ノード
- Computation: 22TFlops
  - 4 P100 GPU, 2 Broadwell CPUs
- Memory Hierarchy
  - GPU: 16GB × 4, 0.7TB/s × 4
  - Host: 256GB, 154GB/s
  - NVMe SSD: 2TB, 2.6GB/s



Events

- Sep 17, 2014: 1st Memory Plus Workshop
  - 7 Invited talks, ~80 participants
- Aug 31, 2016: 2nd Memory Plus Workshop
  - Invited talks from Intel, NVIDIA, Toshiba. ~45 participants

Summary

Towards Extreme Big Simulations

- Architecture: Hierarchical Hybrid memory
- System software: Reducing programming cost
- App. Algorithm: Reducing communication

Co-design is the key