# Consolidating memory locality information obtained from static and dynamic analysis of code for performance tuning in source code

Yukinori Sato, Toshio Endo (Tokyo Institute of Technology / JST CREST)

## Background of current HPC systems

Applications keep scaling their size and complexity
Multi-physics simulation
High-precision simulation

Applications become over 10 thousands lines of complex code

Just parallelizing programs with MPI or OpenMP is not enough
Memory locality tuning is mandatory for performance

In the actual HPC field, performance tuning is done manually and empirically by hands of a few experts

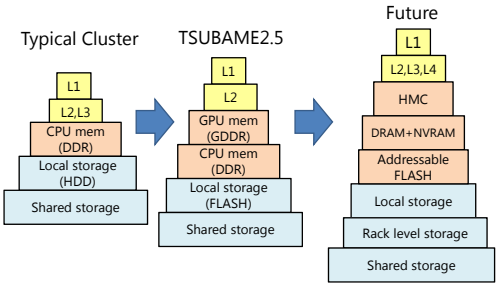
**Extremely low productivity**

For rewarding sustainable performance increasing, we need to solve this productivity issue on tuning

Deeply hierarchical memory

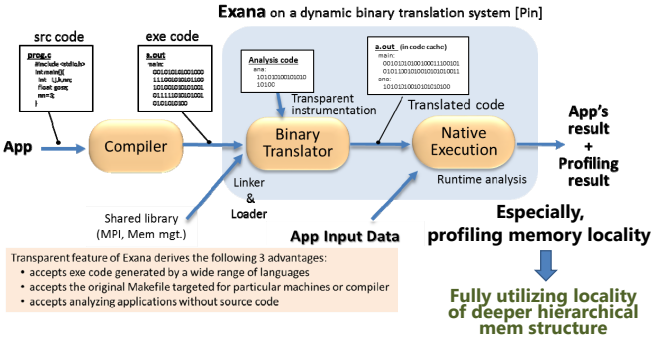
HPC systems with accelerators or many-core CPUs with 3D-stacked memory are becoming common

Deeper memory hierarchy must be managed for sustainalble performance




## Exana: An Execution-driven Application Analysis Tool [1]

Exana is a really integrated tool for transparently analyzing application code using dynamic binary translation technique.




Fully utilizing locality of deeper hierarchical mem structure

Verified applications working with Exana

| | Application | Note |
|---|---|---|
| Full app | NICAM-DC | Riken |
| | OpenMX | JAIST |
| | Numerical Turbine | Tohoku Univ |
| | Graph500 | Kyushu Univ |
| Mini app | NICAM-DC-min | fiber |
| | ffvc-mini | fiber |
| | ccs-qcd-mini | fiber |
| | modylus-mini | fiber |
| | CloverLeaf | Mantevo |
| | CoMD | Mantevo |
| Benchmark | SEPC CPU 2006 | |
| | NAS Parallel Benchmark | |
| | HimenoBMT | |
| | Stencils | |

For analyzing MPI program:
% mpirun –np 16 Exana –cacheSim 1 -- ./openmx Methane.dat

Ready for users

TSUBAME 2.5 supercomputer

Verified platforms running Exana

| | |
|---|---|
| Custom HPC | Cray XC30 |
| Custom HPC | SGI Altix UV |
| HPC with GPU | TSUBAME 2.5 |
| Accelerator | Xeon Phi |
| commodity | x86 Linux |

Exana is installed in the experimental user service directory of Tsubame 2.5 @ Tokyo Tech
http://tsubame.gsic.titech.ac.jp/labs

## Issue: How can the results feedback to source code?

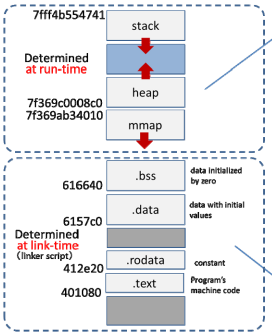
1) Location of source code and object names defined in source code should be linked to the analysis results
2) Memory locality information is generated through different profilers and tools
3) Outputs are located across different files

**Add new analyses for correlating executable code with its source code and make a tool to consolidate target analyses results**

## Memory layout analysis

Memory layout
- Not determined at compile time
- For compilers, it is hard to optimize it without post-link or runtime information

Combining link-time and run-time information to extract memory layout




Memory layout determined at run-time is extracted by dynamic analysis on Exana
- Hook malloc function
  - Detect locations of caling malloc
  - Correlated with source code location using debug information inside binary code
- Replace the malloc function for analysis
  - Into a wrapper function of the malloc
  - Get augments and a return value to malloc

An example of obtained malloc information

Memory layout determined at link-time is extracted by static analysis of code (readelf, objdump)

## Memory access range analysis

Online memory access trace

MemPat — Online access pattern analysis on Exana
Make a local memory access pattern for each instruction with memory accesses

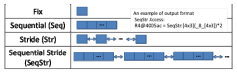
RangeAna
Calculate maximum and minimum addresses of each local pattern from outputs of MemPat

Address ranges of accessed memory for each instruction

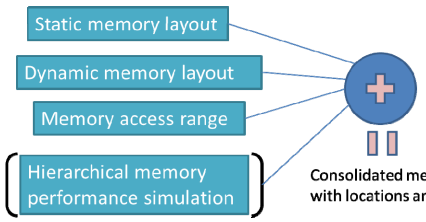
Example of a local memory access pattern

R4@4006ab = {
base pattern
+offset+ base pattern
⋮
+offset+ base pattern
}

** base pattern: Fix/Seq/Str/SeqStr




## Consolidating memory locality info




memConsol script (Python) + Visualizer (ExanaView)

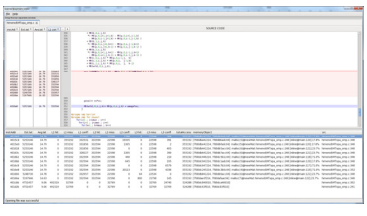
Consolidated memory locality info correlated with locations and object names of source code

## Evaluation

Target: Himeno benchmark (dynamic allocation, OpenMP, C lang version)
Machine: Intel Core i7-2700K CPU, CentOS 6.7 (on VMWare WS player), gcc4.4.7, Single thread execution

1. Perform profiling with Exana and result consolidation
2. Visualize using ExanaView

- We can detect source code locations where cache-line conflicts heavily occur
- Based on object name information, we find where we should perform padding for arrays by putting extra space (+1) for all of dimensions (i, j, k).
- Score of Himeno benchmark:
  1570MFLOPS → 1991MFLOPS (1.3x speedup)



ExanaView

[1] 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.

**For downloading Exana tool kit:**
**http://www.el.gsic.titech.ac.jp/~yukinori/Exana.html**