An Appropriate Computing System and its System Parameter Selection based on Bottleneck Prediction of Applications

Kazuhiko Komatsu, Takumi Kishitani,
Masayuki Sato, Hiroaki Kobayashi
Tohoku University

24 May, 2019

IPDPS Workshop IWAPT2019

Background

- Various designs of recent computing systems
  - Multi/many-core technology
    - 10~100 cores in a processor
  - Vector processing technology
    - simultaneous calculations of multiple elements
      - The number of elements tends to be increased
  - Hybrid memory architecture
    - High bandwidth memories with the conventional DDR memory
      - MCDRAM&DDR in KNL, HBM in Volta and SX-Aurora TSUBASA

It is not easy to judge an appropriate computing system

Background (cont.d)

- Performance tuning are necessary for recent computing systems
  - A number of system parameters to be tuned due to its complexity
    - Difficult to find the best parameters in a practical time
      - 300+ patterns for all parameter combinations in KNL
  - Long time of each application execution
    - Execution time of an HPC application tends to increase due to more advanced, detailed, precise simulations
    - Each execution time drastically affects tuning times
      - An app is executed many times to find an appropriate parameter sets

A brute force approach for various computing systems and system parameters is not practical

Objective and approach

- Objective
  - Reduce the time for processor selection and parameter tuning
    - An appropriate parameter combination is found in a practical time
- Approach
  - Select an appropriate **computing system** in advance
    - Predict a bottleneck
    - Select computing system to relieve the bottleneck
  - Narrow a search space of **system parameters**
    - Select system parameters to solve the bottleneck

Overview of the proposed method

1. Database construction
   - Understand characteristics of system parameters on systems
     - Relationship between each system parameter and performance
     - This information used for system parameter selections in Steps 3 and 4
2. Prediction of bottleneck candidate
   - Identify bottleneck considering both a computing system and an application
3. Computing system selection
   - Select **computing systems that might best solve** the predicted bottleneck
4. System parameter selection
   - Select **only system parameters that are effective to solve** the predicted bottleneck

   → <u>Narrow search space of system parameter combinations</u>

Step1: Database construction

- Understand effects of system parameters on each computing system by evaluating benchmarks with various parameters
  - GEMM
    - Identify contribution of each system parameter to **computational** performance
  - Stream
    - Identify contribution of each system parameter to **memory bandwidth** performance
  - Any other benchmarks
    - Other information can be used for system parameter selection
  - **Only once** when a system is installed
    - This cost can be amortized

Step2: Bottleneck prediction

- Identify bottleneck candidates
  - Bottleneck is utilized to select an appropriate systems and to reduce search space of parameter combinations
  - How to identify?
    - This paper uses **Bytes/Flop** ratios of a system and an app
      - Code B/F < System B/F => **Computational bound**
      - Code B/F > System B/F => **Memory bandwidth bound**

        * Code B/F = (necessary data in Byte) / (# floating operations)

        * System B/F = (memory bandwidth) / (peak performance)

Step3: Computing system selections

- What computing system should be select to achieve high performance?
  - Select **only computing systems that are effective to solve** the predicted bottleneck
    - Comp bound=> systems effective to computation
    - Mem bound => systems effective to memory
    - Characteristics of system parameters are clarified in Step 1

Step4: System parameter selections

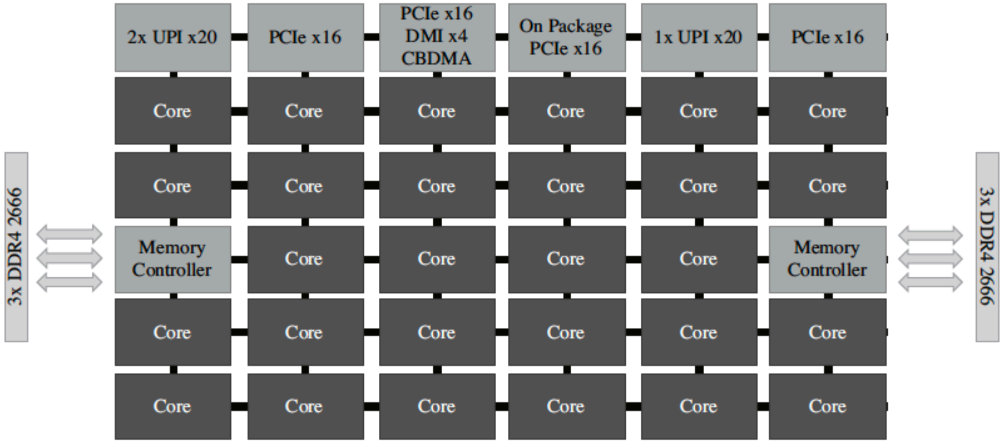
- Wat system parameters should be select to narrow search space?
  - Select **only effective parameter combinations that solve** the predicted bottleneck
    - Comp bound=> parameters effective to computation
    - Mem bound => parameters effective to memory

Case study: target application kernels

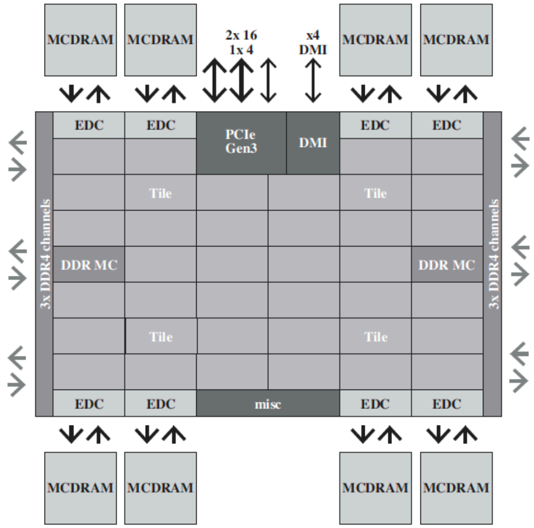
Kernels	Fields	Methods	Memory access	Mesh size	Code B/F
Land mine	Electro magnetic	FDTD	Sequential	100x750x750	5.15
Earthquake	Seismolo gy	Friction Law	Sequential	2047x2047x256	4.00
Turbulent Flow	CFD	Navier-Stokes	Sequential	512x16384x512	0.35
Antenna	Electro magnetic	FDTD	Sequential	252755x9x97336	0.98
Plasma	Physics	Lax-Wendroff	Indirect	20,048,000	0.075
Turbine	CFD	LU-SGS	Indirect	480x80x80x10	0.0084

Case study: four target systems

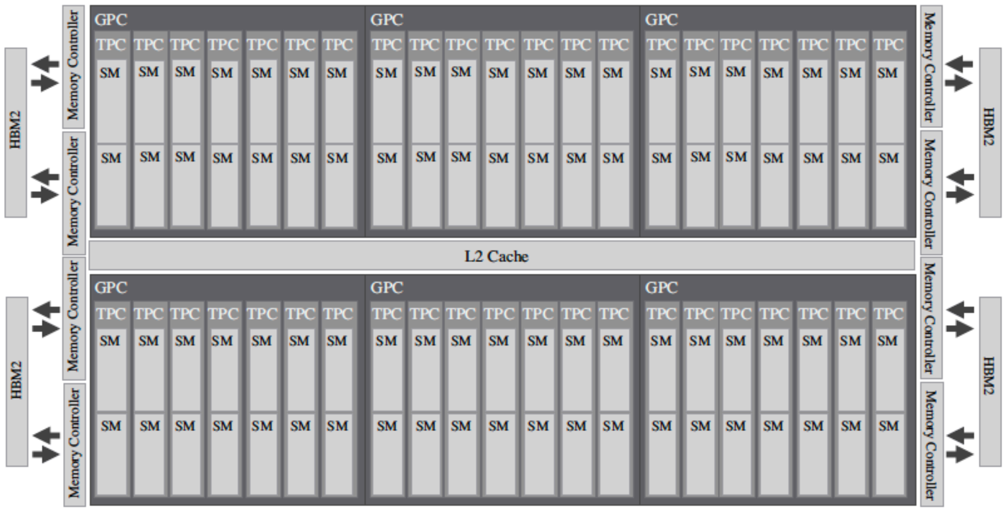
Skylake



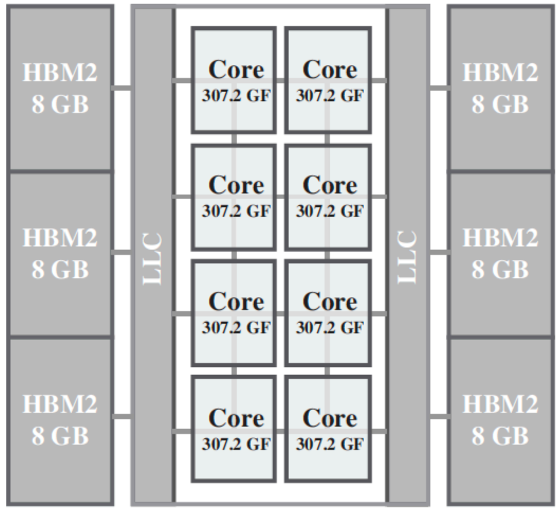
KNL



Nvidia V100



SX-Aurora TSUBASA



Experimental environments

Processor	SX-Aurora Type 10B	Xeon Gold 6126	Tesla V100	Xeon Phi KNL 7290
Frequency	1.4 GHz	2.6 GHz	1.245 GHz	1.5 GHz
# of cores	8	12	**5120**	72
DP flop/s (SP flop/s)	2.15 TF (4.30 TF)	998.4 GF (1996.8 GF)	**7 TF (14 TF)**	3.456 TF (6.912 TF)
Memory subsystem	**HBM2 x6**	DDR4 x6ch	HBM2 x4	MCDRAM DDR4
Memory BW	**1.22 TB/s**	128 GB/s	900 GB/s	450+ GB/s 115.2 GB/s
Memory capacity	48 GB	**96 GB**	16 GB	16 GB 96 GB
LLC BW	**2.66 TB/s**	N/A	N/A	N/A
LLC capacity	16 MB shared	19.25 MB shared	6 MB shared	1 MB shared by 2 cores

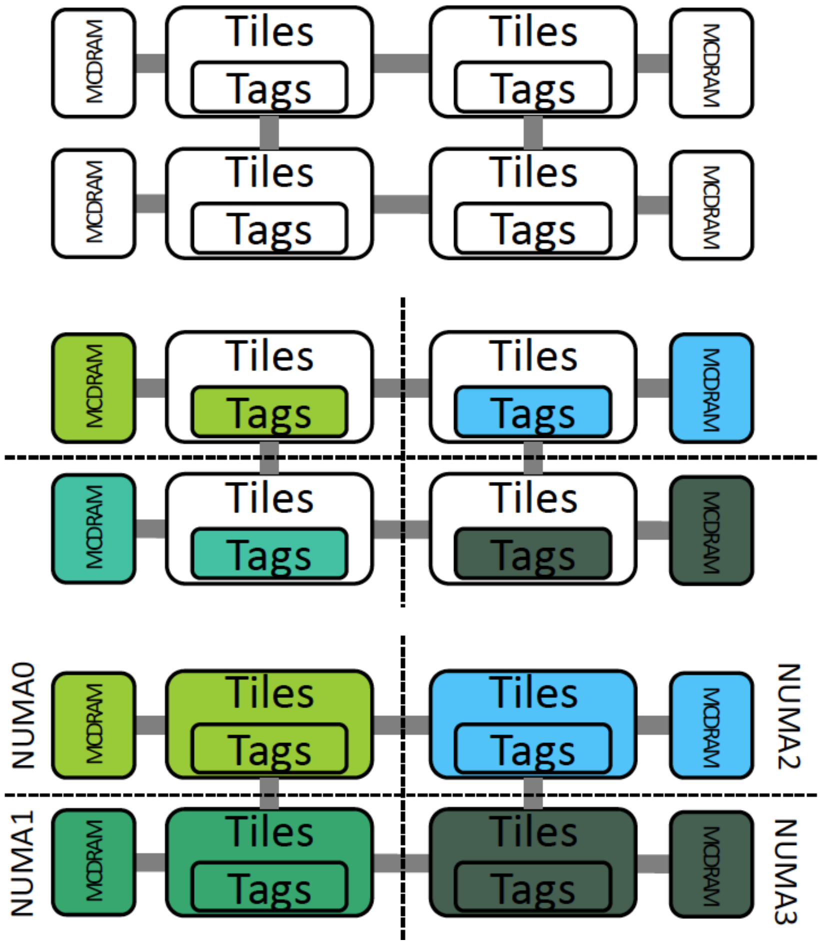
Experimental environments cont.

- Configurable system parameters

Processor	SX-Aurora Type 10B	Xeon Gold 6126	Tesla V100	Xeon Phi KNL 7290
# threads	1~8	1~12	2^5~2^{10}	72, 144, 216, 288
Thread affinity	N/A	compact, scatter	N/A	compact, scatter, balanced
Cluster mode	N/A	N/A	N/A	a2a, quad, hemi, SNC-2or4
Memory mode	N/A	N/A	N/A	flat, cache, hybrid
# thread blocks	N/A	N/A	2^0~2^{16}	N/A

Configurable system parameters (1)

- ***Cluster*** mode
  - Decides how to logically divides tiles and memory into virtual regions
  - *all-to-all*
    - Not divided into virtual regions
    - UMA
  - *Quadrant, Hemisphere*
    - Divides into 4 or 2 virtual regions
    - UMA
    - Smaller latency than all-to-all
  - *Sub-NUMA Cluster(SNC)-4, SNC-2*
    - Divides into 4 or 2 virtual regions
    - NUMA
      - Nume-aware optimization is necessary



Configurable system parameters (2)

- ***Memory*** mode
  - Decides how to use MCDRAM and DDR
  - ***Flat*** mode
    - MCDRAM and DDR have the same address space
      - An application needs to explicitly allocate data in MCDRAM
  - ***Cache*** mode
    - All MCDRAM is treated as a cache of DDR
      - Cache is hardware-managed, so no explicit programming:)
  - ***Hybrid*** mode
    - Combination of flat mode and cache mode
      - 25% or 50% of MCDRAM is used as cache.
      - Remaining MCDRAM is used as allocatable memory

Configurable system parameters (3)

- **Thread affinity**
  - *Compact*
    - A thread is assigned to a core as close as possible to its adjacent thread
    - Suitable for **computation-intensive** applications
  - *Scatter*, *balanced*
    - Threads are distributed across all cores as much as possible
    - Balanced affinity assigns a close thread to the same core
    - Suitable for **memory-intensive** applications
- **Number of threads**
  - Up to 4 threads can be assigned to one core
    - 72, 144, 216, 288 are candidates to use full cores in KNL

The number of combinations of system parameters reaches 300

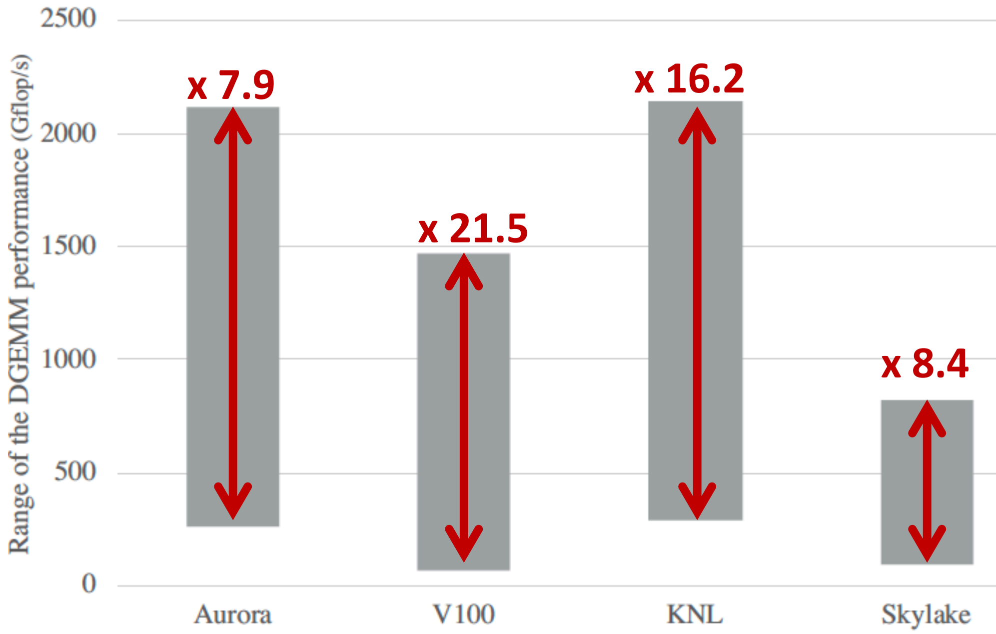
Experimental environments cont.

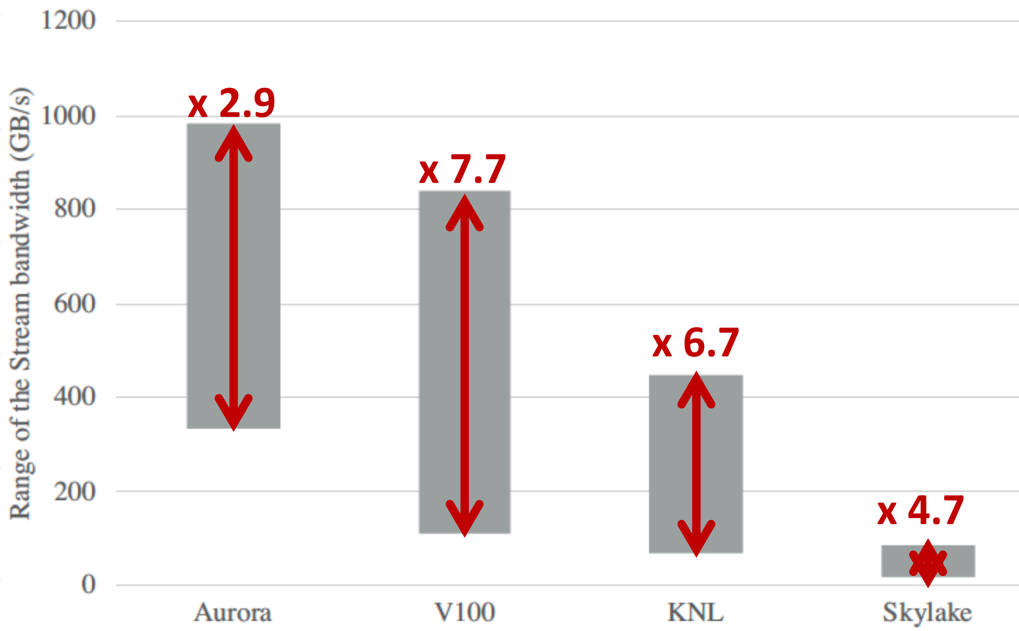
- Configurable system parameters

Processor	SX-Aurora Type 10B	Xeon Gold 6126	Tesla V100	Xeon Phi KNL 7290
# threads	1~8	1~12	2^5~2^{10}	72, 144, 216, 288
Thread affinity	N/A	compact, scatter	N/A	compact, scatter, balanced
Cluster mode	N/A	N/A	N/A	a2a, quad, hemi, SNC-2or4
Memory mode	N/A	N/A	N/A	flat, cache, hybrid
# thread blocks	N/A	N/A	2^0~2^{16}	N/A

1029 parameter combinations for all computing systems

Step1: Database construction (DGEMM/Stream performance)





- Aurora: 8 threads
- V100: 512+ blocks & 128+ threads/block
  - Hand written code
- KNL: 4 clusters, 72 threads, scatter/balanced
- Skylake: 12 threads

- Aurora: 6 threads
- V100: 8192+ blocks & 512+ threads/block
- KNL: 4 clusters, 72 or 144 threads, scatter/balanced
- Skylake: 12 threads

Big performance gap on simple benchmarks => construct the database from the results

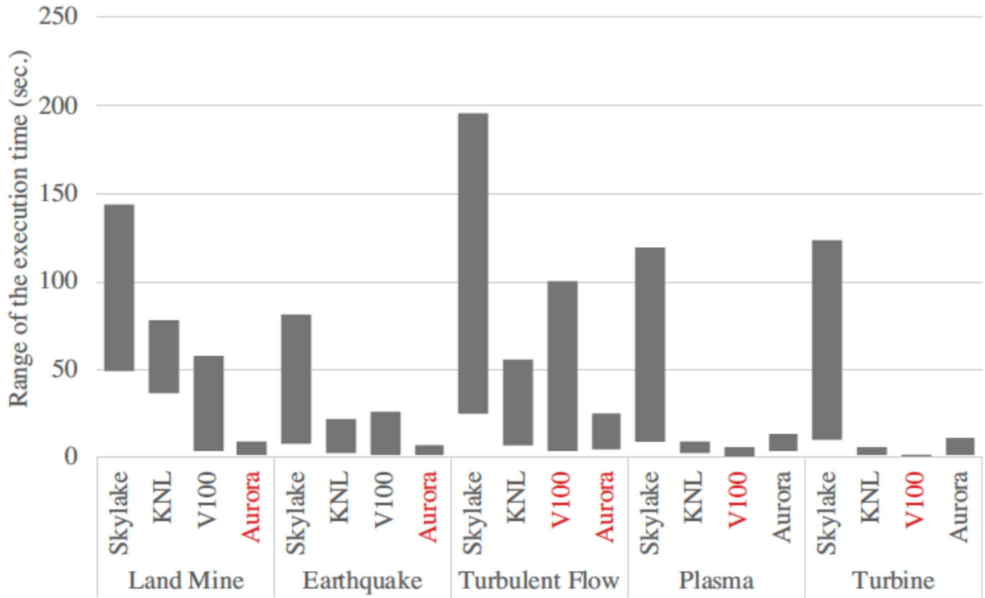
Step2: Bottleneck prediction
Step3: Computing system selections

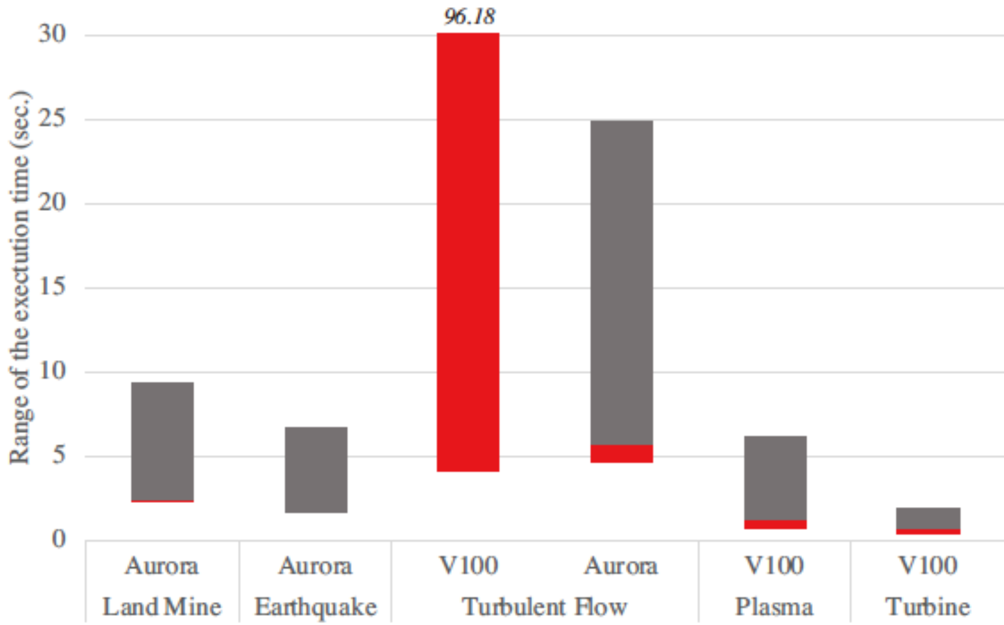
- This paper predicts the bottleneck using **Bytes/Flop** ratios
  - System B/F = (memory BW) / (peak flops)
  - Code B/F of applications = (necessary data) / (# flops)

Systems	System B/F
Xeon	0.128
KNL	0.130
Tesla V100	0.128
SX-Aurora TSUBASA	0.567

Kernels	Code B/F	Selections
Land mine	5.15	SX-Aurora
Earthquake	4.00	SX-Aurora
Turbulent Flow	0.35	Aurora or V100
Antenna	0.98	V100
Plasma	0.075	V100
Turbine	0.0084	V100

Step3: Computing system selections
Step4: System parameter selections





- Appropriate systems can be selected
- System parameters can be narrowed
  - In Land mine, Earthquake, Turbulent flow, Plasma, Turbine, the number of the system parameter candidates can be reduced to **3, 3, 283, 280, 280** from **1029**, respectively.

Conclusions

- Toward reduction in performance tuning time
  - As the numbers of **computing systems** and **system parameters** increase, a long time is necessary for performance tuning
- Approach
  - Select appropriate system and narrow a search space of system parameters
    - Predict a bottleneck and select appropriate system and its system parameters
- Future work
  - More detailed evaluations
    - Not only the full search but also the conventional tuning algorithms
    - Other application such as practical applications