

A case study on modeling the performance of dense matrix computation: Tridiagonalization in the EigenExa eigensolver on the K computer

**The 13th International Workshop on Automatic Performance Tuning
(iWAPT2018)**

May 25, 2018

held in conjunction with IEEE IPDPS 2018

@ JW Marriott Parq Vancouver, Vancouver, Canada

Takeshi Fukaya^{1, 2}, Toshiyuki Imamura², Yusaku Yamamoto³

1: Hokkaido University

2: RIKEN Center for Computational Science

3: The University of Electro-Communications

Background

◆ Importance of performance modeling

- Performance modeling is one of important tasks in HPC including automatic performance tuning.
- Well constructed models can reduce the cost of tuning and help one to efficiently allocate limited computational resources.

◆ Difficulties in performance modeling

- Performance behavior of a program (on large-scale parallel systems) is complicated; many factors (cost for arithmetic and communication) effect the execution time.
- Available information for modeling is usually limited.
- Verifying models is also not easy; comparison with actual (measured) timing results is impractical in some cases.

Related work

- **Dackland et al., 1996:**

Performance modeling of the routine in ScaLAPACK by accumulating the estimation of low-level routines.

(Model parameters was configured by theoretical performance of the target system and experiments.)

- **J. Demmel et al., 2012:**

Performance model was used for evaluating the effectiveness of CA-QR/LU factorization on future machines.

(Model was simple, and parameters was determined from expected theoretical peak performance.)

- **A. Calotoiu et al., 2013:**

General techniques for empirical performance modeling

(Automatically selecting basis functions in model based on sampling data)

Overview of this study

◆ Motivation

- During the development of the EigenExa eigensolver on the K computer, we have accumulated a lot of measured timing data.
- For a certain program, comparison of different approaches to performance modeling has not been sufficiently investigated.

◆ Contribution

- For the tridiagonalization routine in EigenExa (named `eigen_trd`), we attempt to construct performance models.
- We consider four approaches to performance modeling and evaluate them by comparing with measured timing data.
- We also present some observations from measured data, which will be informative for further research for modeling.

Outline

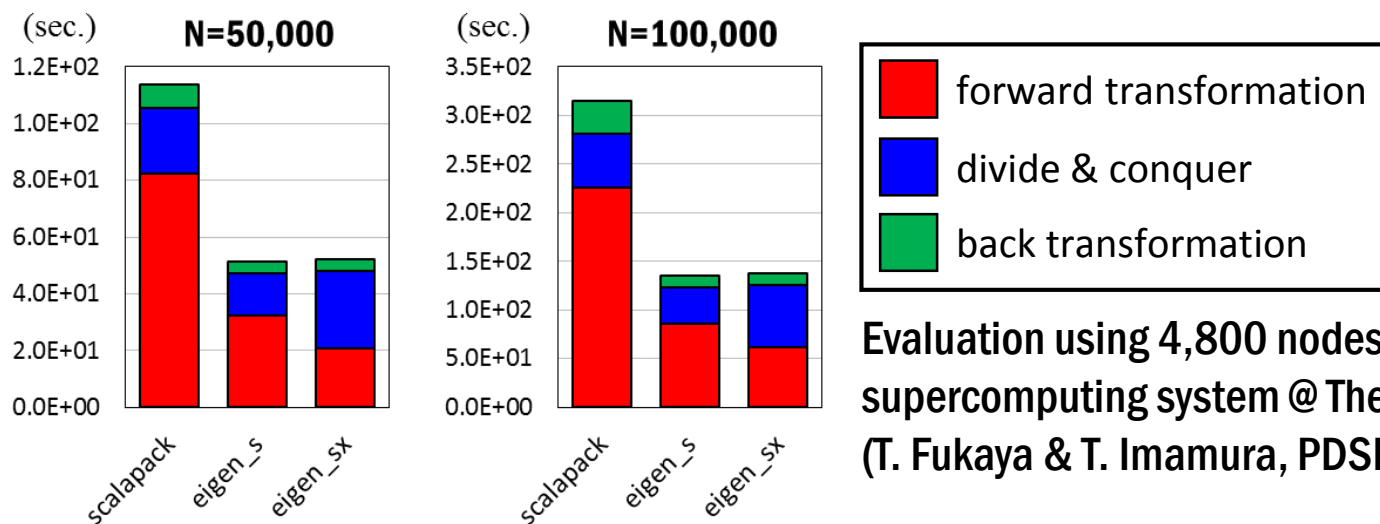
1. Introduction
- 2. Overview of eigen_trd**
- 3. Measured results on the K computer**
- 4. Approaches to performance modeling**
- 5. Evaluation of the performance models**
- 6. Conclusion**

Overview of eigen_trd

Overview of EigenExa

◆ EigenExa eigensolver (main developer: T. Imamura)

- Target problem: real symmetric dense eigenvalue problem
- Target system: K computer and post-petascale systems
- Two driver routines:
 - ✓ eigen_s: traditional approach based on tridiagonalization
 - ✓ eigen_sx: new approach employing penta-diagonalization

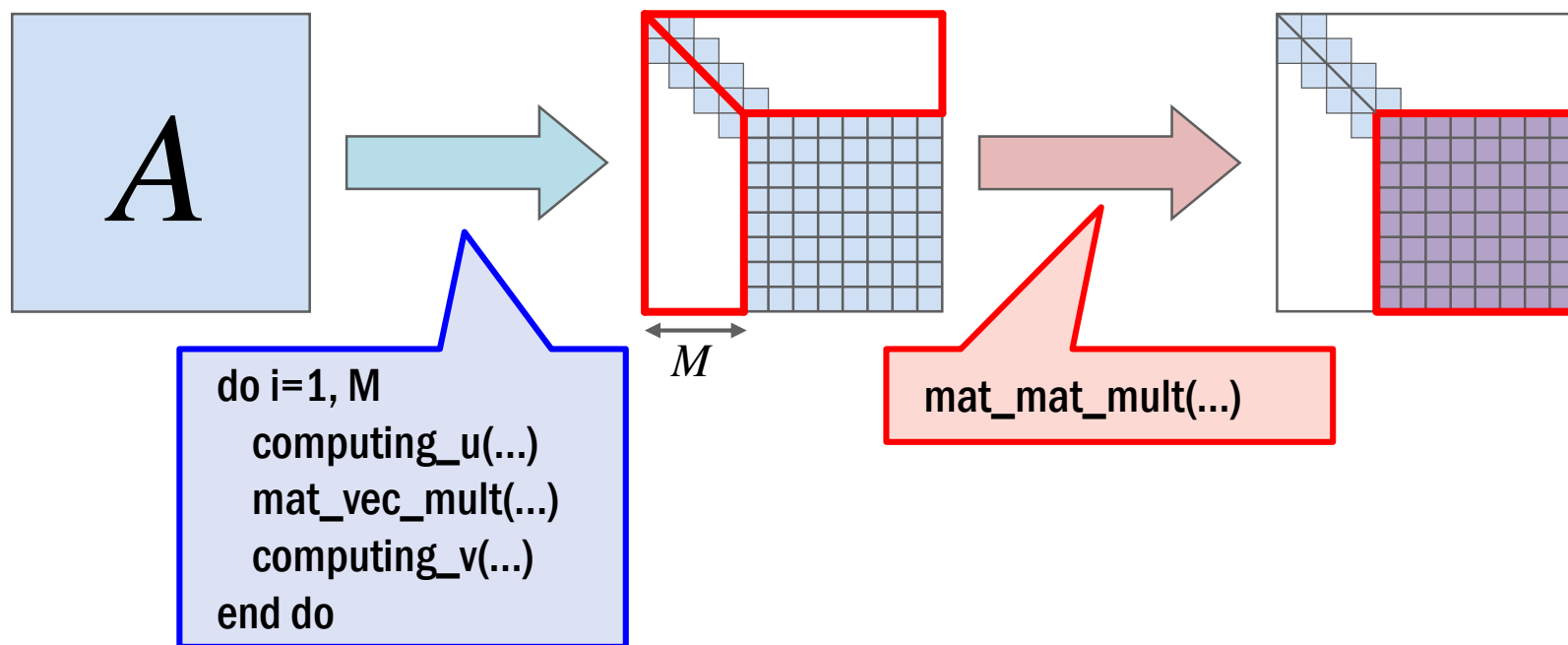


Evaluation using 4,800 nodes of Oakleaf-FX supercomputing system @ The Univ. of Tokyo (T. Fukaya & T. Imamura, PDSEC2015)

Overview of eigen_trd

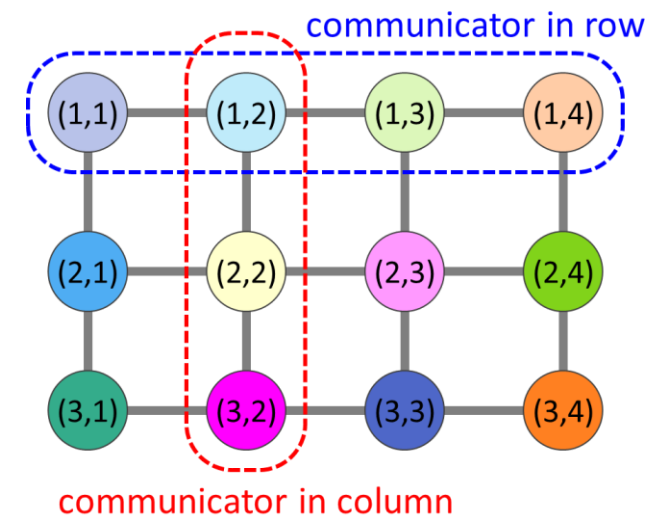
◆ eigen_trd: tridiagonalization routine in EigenExa

- Transform an input symmetric matrix into a tridiagonal matrix using orthogonal Householder transformations.
- Employs Dongarra's method to reduce the memory access cost in symmetric matrix-vector multiplications.



Arithmetic & communication cost

Computation	#flops per process	MPI_Bcast (\sqrt{P} processes)		MPI_Allreduce (\sqrt{P} processes)	
		#issues	#words	#issues	#words
computing_u	$O\left(\frac{N^2}{\sqrt{P}}\right)$	$2N$	$\frac{N^2}{\sqrt{P}}$	N	$O(N)$
mat_vec_mult	$\frac{2N^3}{3P}$			$2N$	$\frac{N^2}{\sqrt{P}}$
computing_v	$O\left(\frac{N^2}{\sqrt{P}}\right)$	N	$\frac{N^2}{2\sqrt{P}}$	$2N$	$O(N)$
mat_mat_mult	$\frac{2N^3}{3P}$				



process grid in EigenExa

(N : matrix size, P : # of processes, process grid: $\sqrt{P} \times \sqrt{P}$)

Measured results on the K computer

Evaluation conditions

◆ Specifications of the K computer

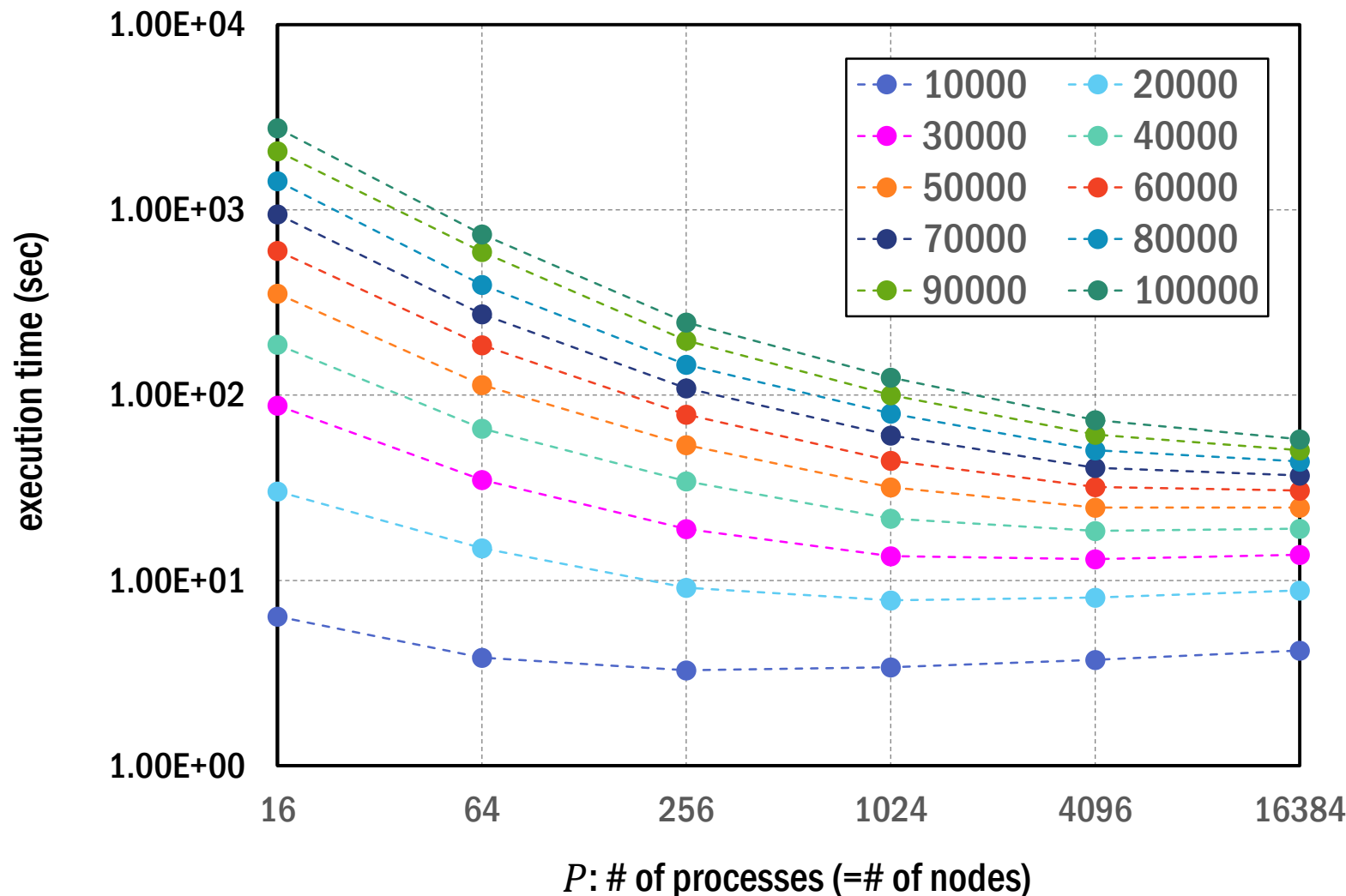
Item	Specification/values
CPU	SPARC64 VIIIfx (2.0GHz, 8cores)
Memory	DDR3 SDRAM (64GB/s)
Node	1 CPU and 16 GB memory
Network	Torus fusion 6D mesh/torus (Tofu), 5 GB/s/link, bidirectional
System	88,128 nodes (compute nodes: 82,944)
Peak FLOPS/system	11.28PFLOPS

◆ Evaluation conditions (for details, see our paper)

- Fujitsu Fortran90 (mpifrtpx) compiler with “-Kfast,openmp -Cpp -Cfpp” and Fujitsu BLAS/LAPACK MPI libraries.
- Test matrix: $A_{i,j} = N + 1 - \max(i, j)$

Measured timing data

Totally 60 timing results of eigen_trd on the K computer

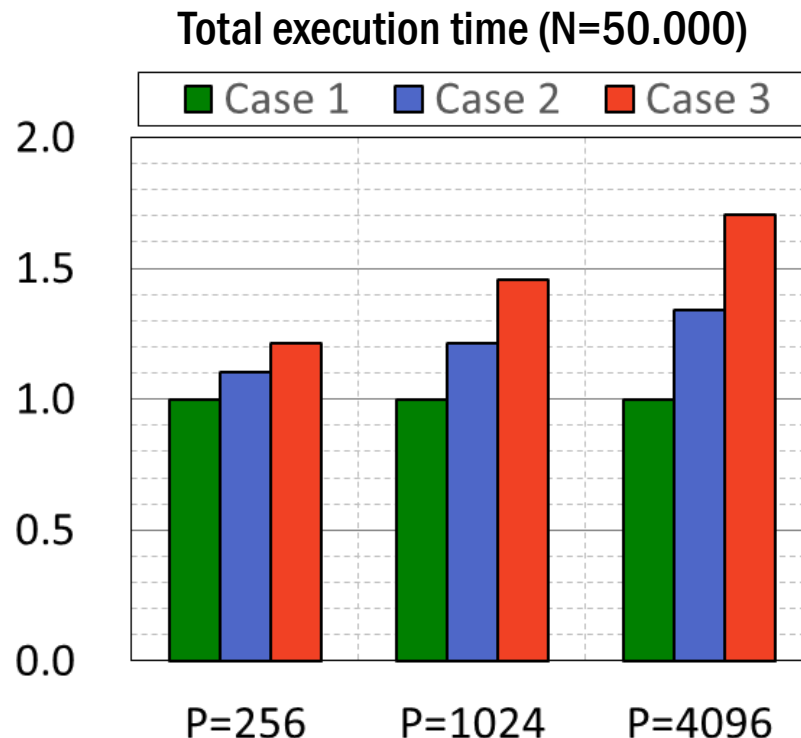


Difficulty in detailed measurement

```
subroutine function1(...)
  MPI_Barrier(...) --- (*)
  t1 = MPI_Wtime()
  [computation]
  MPI_Barrier(...) --- (**)
  ct1 = MPI_Wtime()
  MPI_Allreduce(...)
  MPI_Barrier(...) --- (**)
  ct2 = MPI_Wtime()
  g_ctime1 = g_ctime1 + (ct2 - ct1)
  [computation]
  MPI_Barrier(...) --- (*)
  t2 = MPI_Wtime()
  g_time1 = g_time1 + (t2 - t1)
end subroutine
```

Effect of synchronization on the measured timing results.

- Case 1: no barrier
- Case 2: barrier at (*)
- Case 3: barrier at (*) & (**)

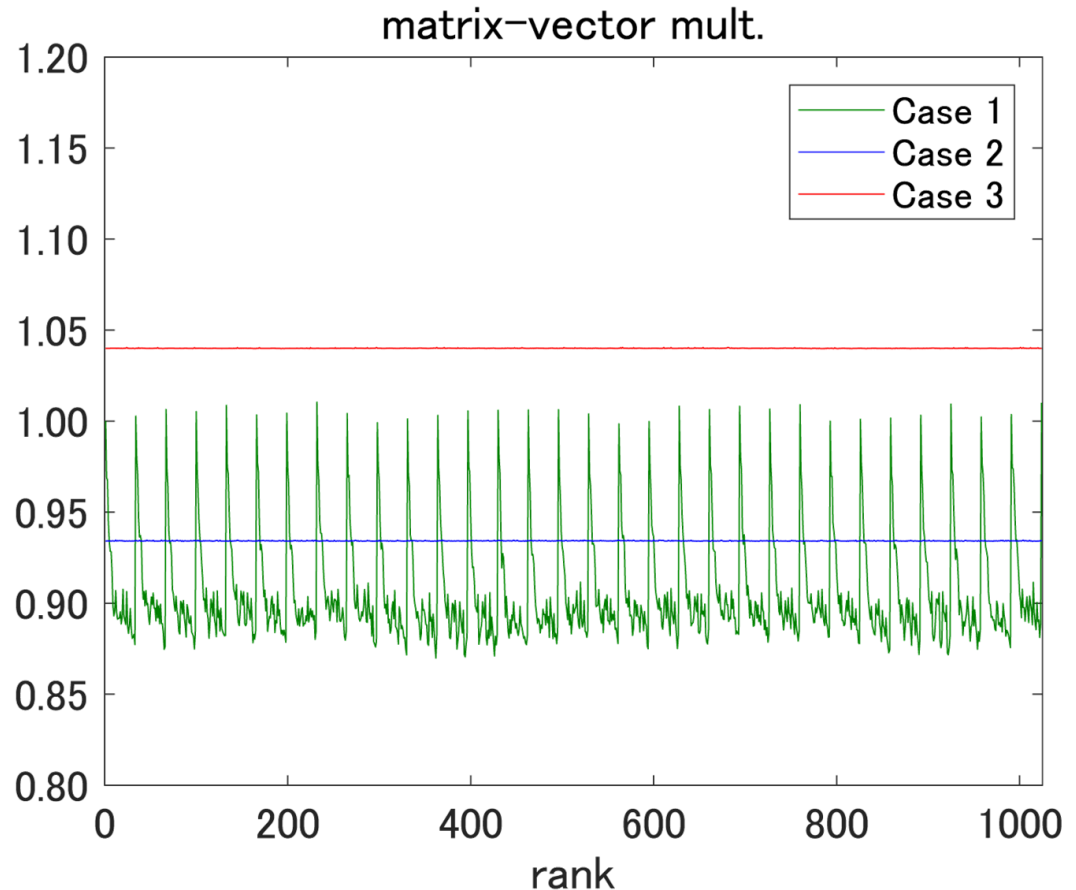


Variation of measured time on each rank

- Case 1: no barrier
- Case 2: barrier at (*)
- Case 3: barrier at (*) & (**)

N=50,000 P=1,024
(normalized by the value of 1st rank in Case 1)

```
subroutine function1(...)
  MPI_Barrier(...) --- (*)
  t1 = MPI_Wtime()
  [computation]
  MPI_Barrier(...) --- (**)
  ct1 = MPI_Wtime()
  MPI_Allreduce(...)
  MPI_Barrier(...) --- (**)
  ct2 = MPI_Wtime()
  g_ctime1 = g_ctime1 + (ct2 - ct1)
  [computation]
  MPI_Barrier(...) --- (*)
  t2 = MPI_Wtime()
  g_time1 = g_time1 + (t2 - t1)
end subroutine
```

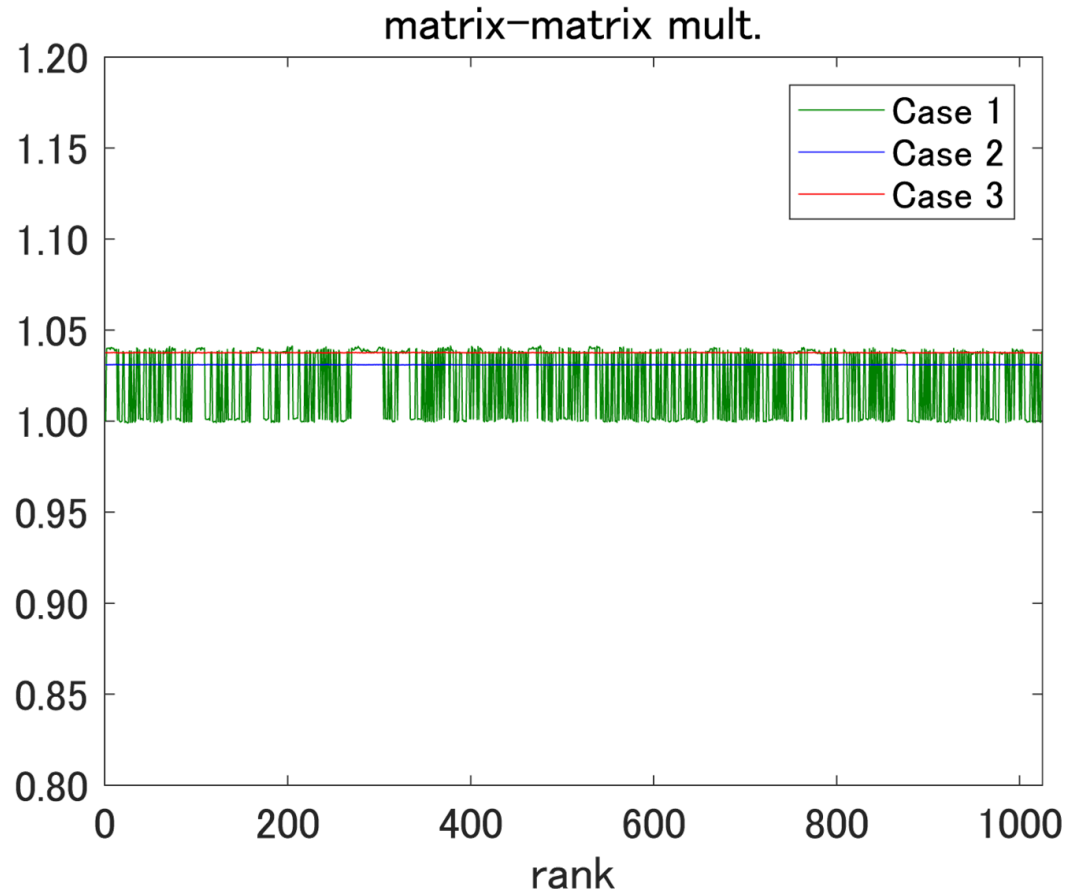


Variation of measured time on each rank

- Case 1: no barrier
- Case 2: barrier at (*)
- Case 3: barrier at (*) & (**)

N=50,000 P=1,024
(normalized by the value of 1st rank in Case 1)

```
subroutine function1(...)
  MPI_Barrier(...) --- (*)
  t1 = MPI_Wtime()
  [computation]
  MPI_Barrier(...) --- (**)
  ct1 = MPI_Wtime()
  MPI_Allreduce(...)
  MPI_Barrier(...) --- (**)
  ct2 = MPI_Wtime()
  g_ctime1 = g_ctime1 + (ct2 - ct1)
  [computation]
  MPI_Barrier(...) --- (*)
  t2 = MPI_Wtime()
  g_time1 = g_time1 + (t2 - t1)
end subroutine
```

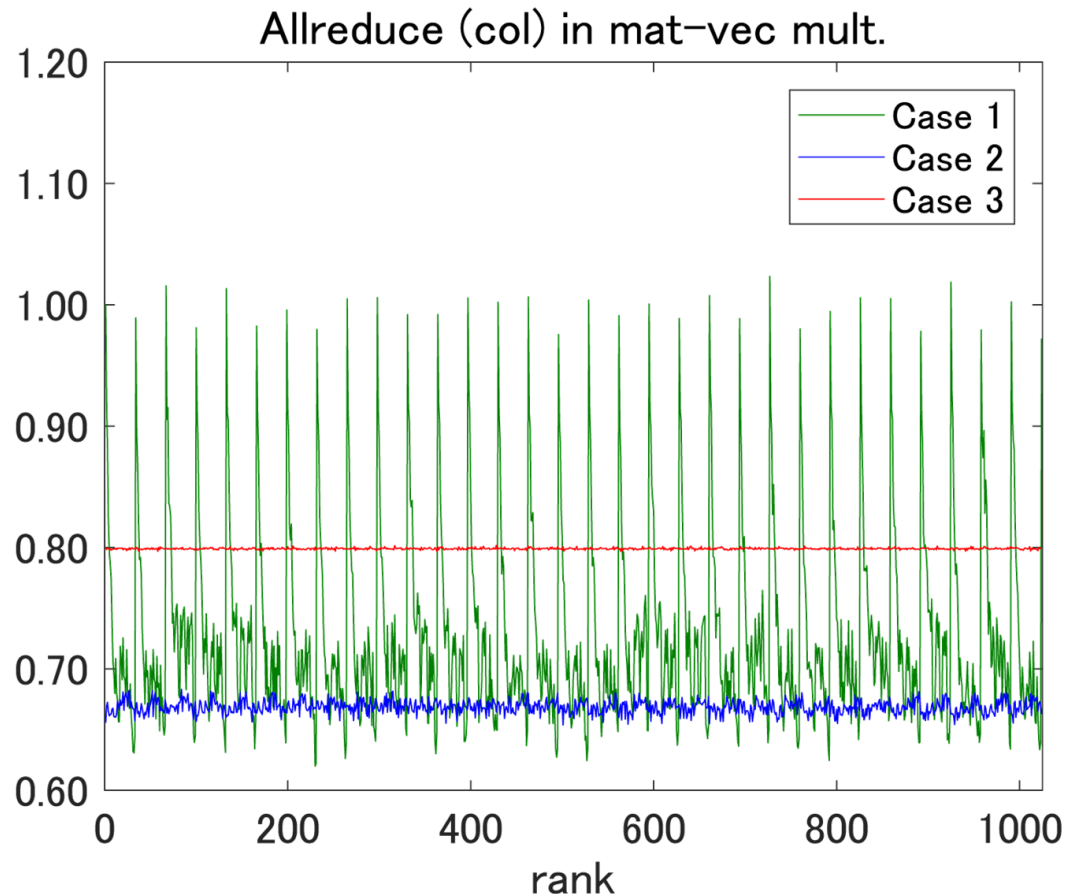


Variation of measured time on each rank

- Case 1: no barrier
- Case 2: barrier at (*)
- Case 3: barrier at (*) & (**)

N=50,000 P=1,024
(normalized by the value of 1st rank in Case 1)

```
subroutine function1(...)
  MPI_Barrier(...) --- (*)
  t1 = MPI_Wtime()
  [computation]
  MPI_Barrier(...) --- (**)
  ct1 = MPI_Wtime()
  MPI_Allreduce(...)
  MPI_Barrier(...) --- (**)
  ct2 = MPI_Wtime()
  g_ctime1 = g_ctime1 + (ct2 - ct1)
  [computation]
  MPI_Barrier(...) --- (*)
  t2 = MPI_Wtime()
  g_time1 = g_time1 + (t2 - t1)
end subroutine
```



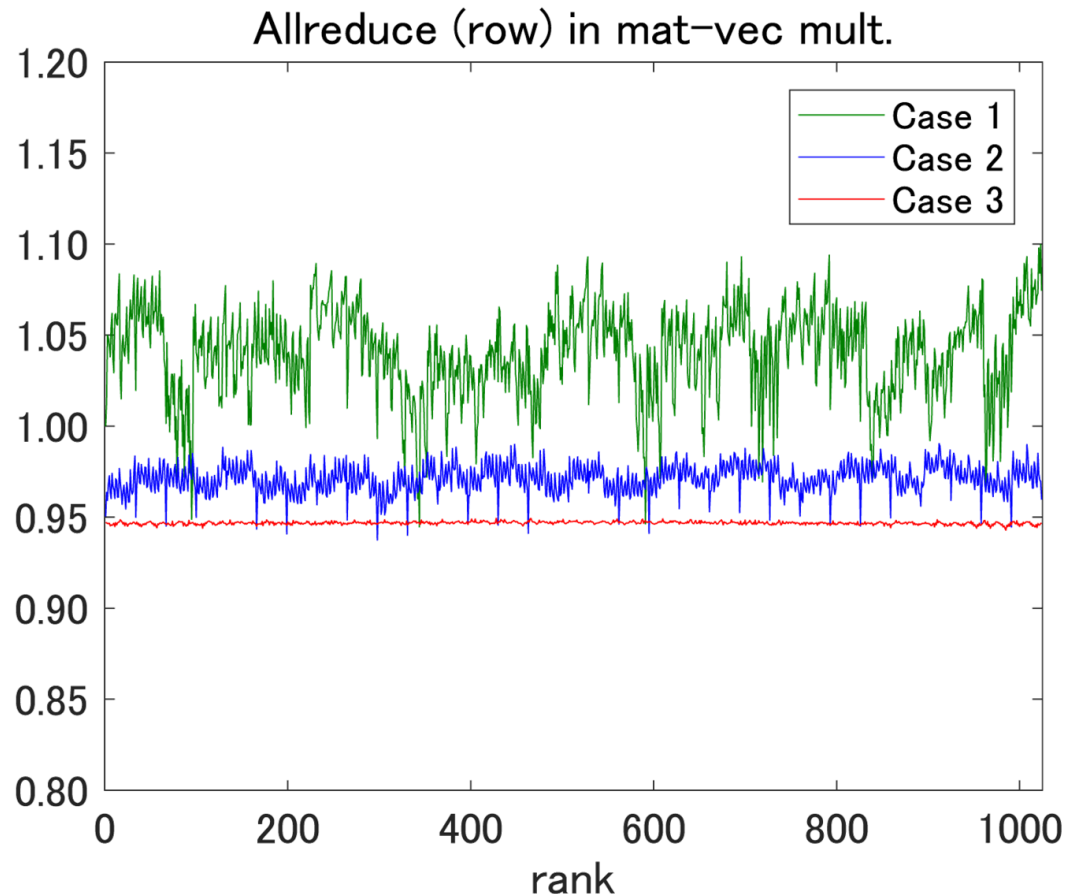
Variation of measured time on each rank

- Case 1: no barrier
- Case 2: barrier at (*)
- Case 3: barrier at (*) & (**)

N=50,000 P=1,024

(normalized by the value of 1st rank in Case 1)

```
subroutine function1(...)
  MPI_Barrier(...) --- (*)
  t1 = MPI_Wtime()
  [computation]
  MPI_Barrier(...) --- (**)
  ct1 = MPI_Wtime()
  MPI_Allreduce(...)
  MPI_Barrier(...) --- (**)
  ct2 = MPI_Wtime()
  g_ctime1 = g_ctime1 + (ct2 - ct1)
  [computation]
  MPI_Barrier(...) --- (*)
  t2 = MPI_Wtime()
  g_time1 = g_time1 + (t2 - t1)
end subroutine
```



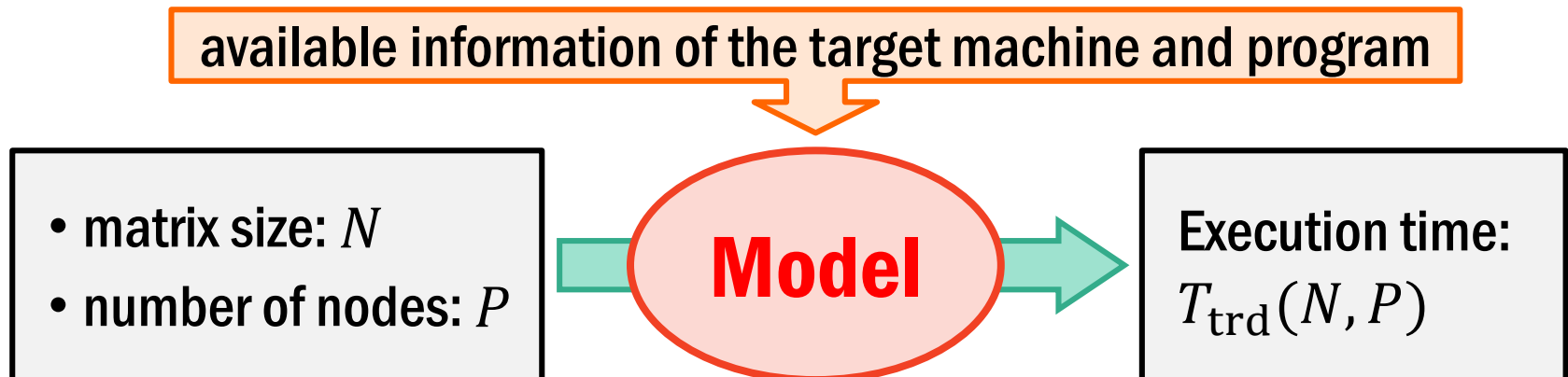
Approaches to Performance modeling

Assumption and goal of modeling

◆ Assumption

- Target machine is given: K computer, information of the machine is limited (in next slides).
- Target program is given: eigen_trd in EigenExa, **information of the program (computational and communication pattern) is already known.**

◆ Goal



Situations

We attempt modeling in four situations:

1. The target machine does not exist:
only the **specifications** of the machine are available.
2. Simple benchmark programs can be run using a part of the target machine:
results of **BLAS routines** and **MPI ping-pong** are available.
3. Specified benchmark programs can be run using limited computational resources:
results of **modified version of the target program** (without MPI routines) and **MPI collective routines** are available.
4. The target program can be run using sufficient resources:
measured results of the target program are available.

In each situation, available information for modeling is different.

Situation 1

Modeling in Situation 1

◆ Specifications of the K computer

Symbol	Item	Specification
F	FLOPS/node	1.28×10^{11} (flop/sec)
B_M	Memory bandwidth/node	6.40×10^{10} (byte/sec)
B_N	Network bandwidth	5.00×10^9 (byte/sec)
L	Network latency	9.20×10^{-7} (sec)

◆ Main idea

$$T_{\text{trd}}(N, P) = T_{\text{flop}}(N, P) + T_{\text{comm}}(N, P)$$

time for arithmetic

time for communication

We separately model the time for arithmetic and communication.

Modeling the time for arithmetic

$$T_{\text{flop}} = (\text{time for one operation}) \cdot (\# \text{ of operations})$$



We distinguish between mat-mat and mat-vec. mult.
(compute bound and memory bound)

$$T_{\text{flop}}(N, P) = \gamma_{\text{mm}} \cdot \frac{2N^3}{P} + \gamma_{\text{mv}} \cdot \frac{2N^3}{P}$$

mat-mat mult.

mat-vec mult.



B/F ratio in symmetric mat-vec mult. is 4flops/8bytes.

$$\gamma_{\text{mm}} = \frac{1}{F}, \quad \gamma_{\text{mv}} = \frac{2}{B_M}$$

Modeling the time for communication

$$T_{\text{p2p}}(w) = \alpha + \beta \cdot w$$



w : data size

$$T_{\text{comm}}(N, P) = \alpha \cdot M_{\text{p2p}}(N, P) + \beta \cdot W_{\text{p2p}}(N, P)$$

total number of issues

total amount of data



We assume MPI_Allreduce and MPI_Bcast are operated along with a binary tree.

$$M_{\text{p2p}}(N, P) = 8N \log_2 \sqrt{P}, \quad W_{\text{p2p}}(N, P) = \frac{5N^2}{2\sqrt{P}} \log_2 \sqrt{P}$$



$$\alpha = L, \quad \beta = \frac{8}{B_N}$$

Obtained model in Situation 1

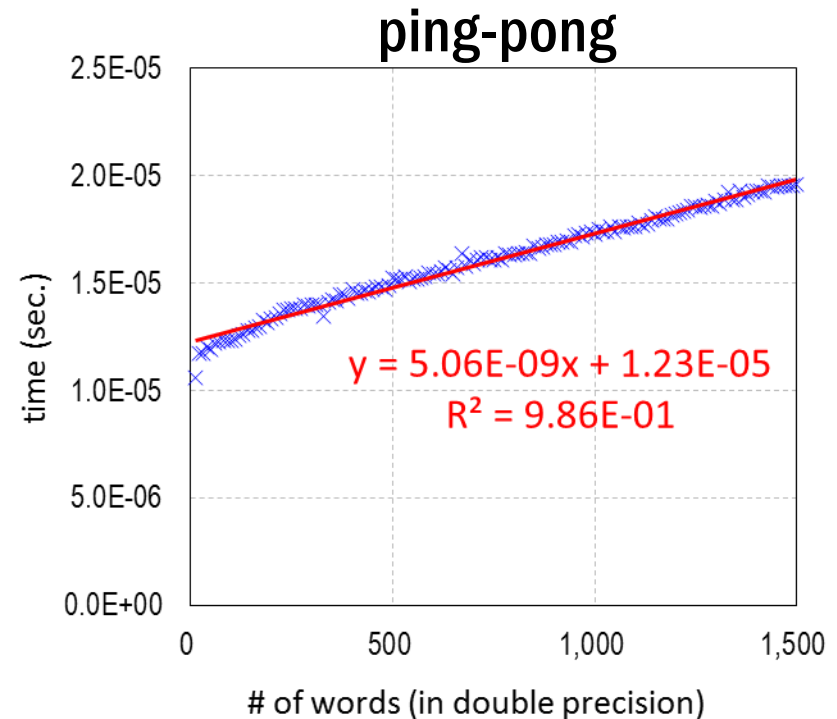
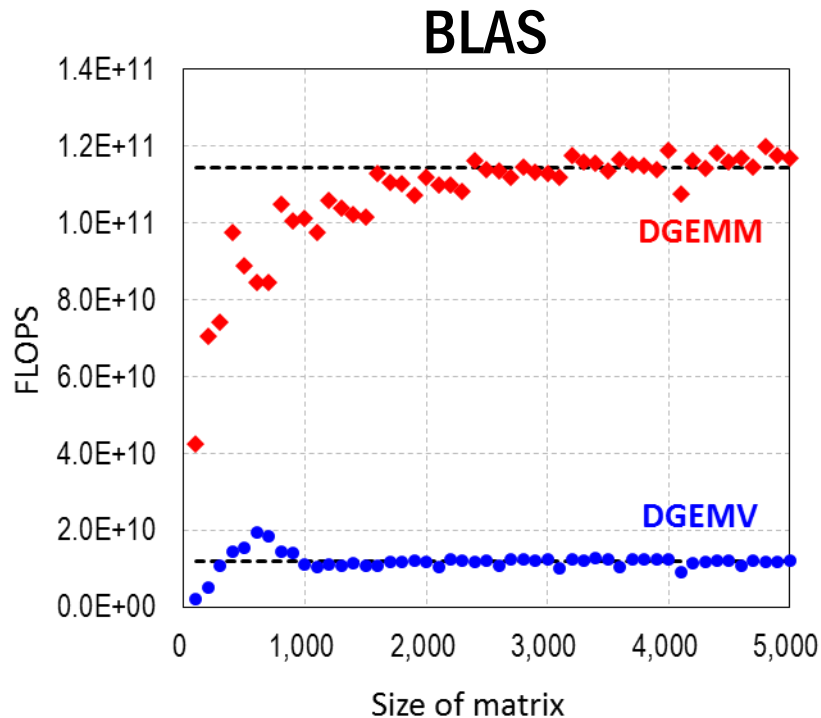
$$T_{\text{trd}}(N, P) = \left(\frac{1}{F} + \frac{2}{B_M} \right) \cdot \frac{2N^3}{P} + L \cdot 8N \log_2 \sqrt{P} + \frac{8}{B_N} \cdot \frac{5N^2}{2\sqrt{P}} \log_2 \sqrt{P}$$

Symbol	Item	Specification
F	FLOPS/node	1.28×10^{11} (flop/sec)
B_M	Memory bandwidth/node	6.40×10^{10} (byte/sec)
B_N	Network bandwidth	5.00×10^9 (byte/sec)
L	Network latency	9.20×10^{-7} (sec)

Situation 2

Modeling in Situation 2

◆ Benchmark results of BLAS & MPI ping-pong

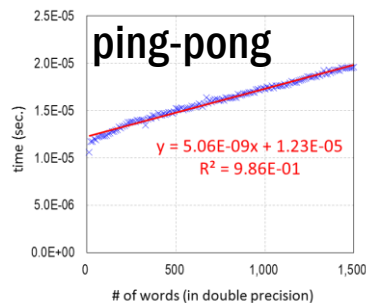
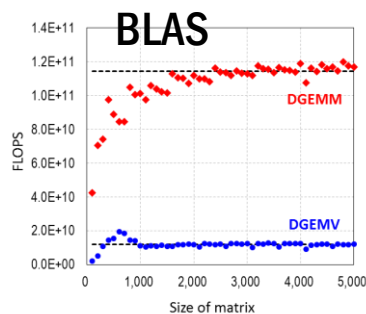


◆ Main idea

We employ the same approach in Situation 1,
but **model parameters are determined from benchmark results.**

Obtained model in Situation 2

$$T_{\text{trd}}(N, P) = (\gamma_{\text{mm}} + \gamma_{\text{mv}}) \cdot \frac{2N^3}{P} + \alpha \cdot 8N \log_2 \sqrt{P} + \beta \cdot \frac{5N^2}{2\sqrt{P}} \log_2 \sqrt{P}$$



Parameter	Value
γ_{mm}	8.77×10^{-12}
γ_{mv}	4.23×10^{-11}
α	1.23×10^{-5}
β	5.06×10^{-9}

B/F ratio in dsymv is 1/2 that in dgemv.

By LS fitting.

Situation 3

Modeling in Situation 3

◆ Execution of specified benchmark program

- A modified version of eigen_trd **without MPI routines** (Program can finish, but result is nonsense.)
- MPI collective routines using \sqrt{P} **processes**.

◆ Main idea

$$T_{\text{trd}}(N, P) = T_{\text{flop}}(N, P) + T_{\text{al}}(N, P) + T_{\text{bc}}(N, P)$$

time for MPI_Allreduce

time for MPI_Bcast

- $T_{\text{flop}}(N, P)$ is **directly measured** by the modified program.
- $T_{\text{al}}(N, P)$ and $T_{\text{bc}}(N, P)$ are **modeled using the benchmark results** of MPI collective routines.

Time of MPI collective routines

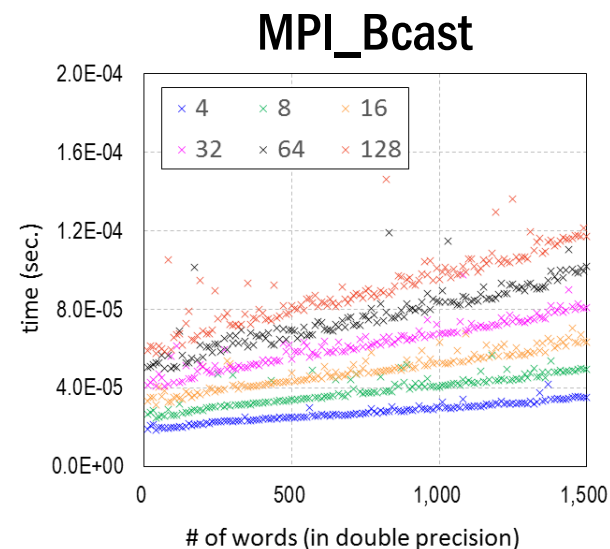
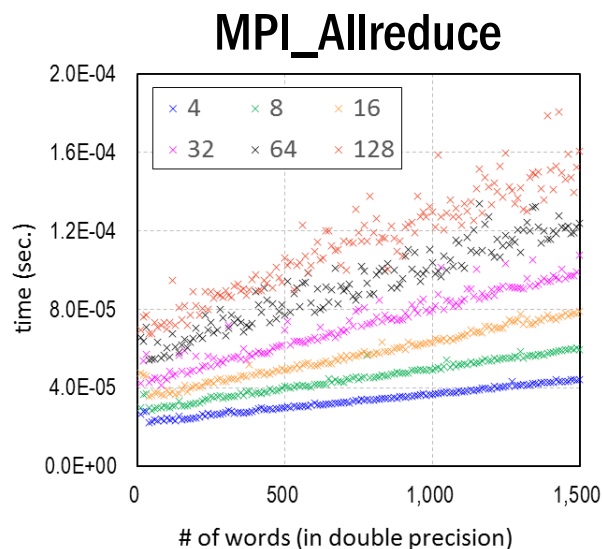
- $T_{al}(N, P) = \alpha_{al}(\sqrt{P}) \cdot 5N + \beta_{al}(\sqrt{P}) \cdot \frac{N^2}{\sqrt{P}}$
- $T_{bc}(N, P) = \alpha_{bc}(\sqrt{P}) \cdot 3N + \beta_{bc}(\sqrt{P}) \cdot \frac{3N^2}{2\sqrt{P}}$



total number of issues

total amount of data

We determine the model parameters from benchmark results.

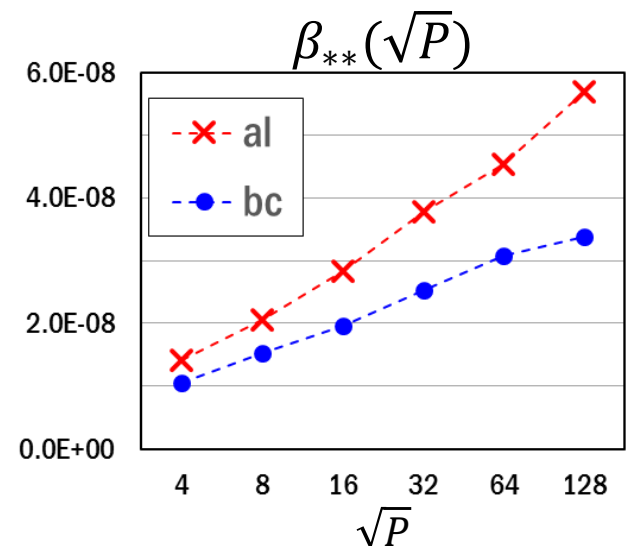
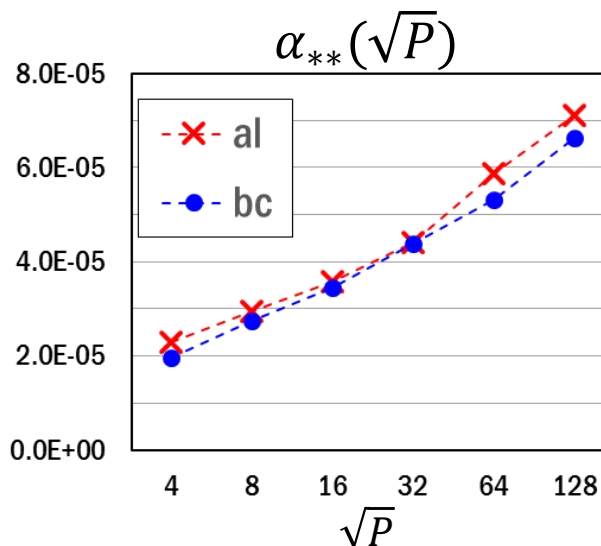
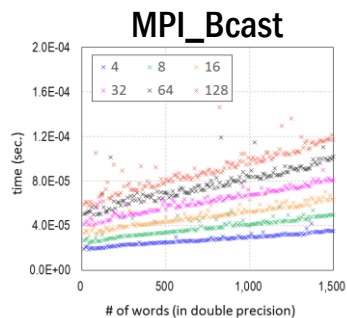
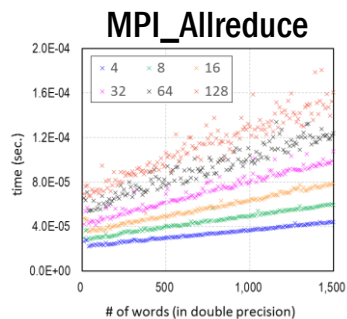


Obtained model in Situation 3

$$T_{\text{trd}}(N, P) = T_{\text{flop}}(N, P) \quad \text{Measured}$$

$$+ \alpha_{\text{al}}(\sqrt{P}) \cdot 5N + \beta_{\text{al}}(\sqrt{P}) \cdot \frac{N^2}{\sqrt{P}}$$

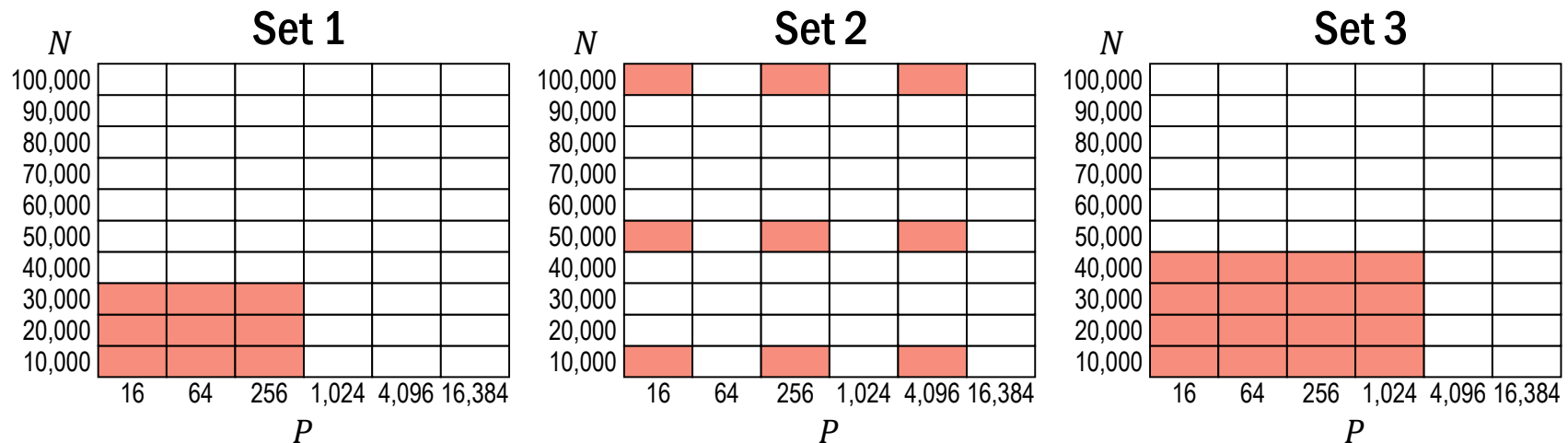
$$+ \alpha_{\text{bc}}(\sqrt{P}) \cdot 3N + \beta_{\text{bc}}(\sqrt{P}) \cdot \frac{3N^2}{2\sqrt{P}}$$



Situation 4

Modeling in Situation 4

◆ A number of measured timing results of eigen_trd



◆ Main idea

- We construct a model as a **linear combination of basis functions of N and P** :


$$T_{\text{trd}}(N, P) = \sum c_i \cdot F_i(N, P).$$

- We determine the model parameters by **LS fitting using the sample data**.

Obtained model in Situation 4

$$T_{\text{trd}}(N, P) = c_1 \cdot \frac{N^3}{P} + c_2 \cdot N \log_2 \sqrt{P} + c_3 \cdot \frac{N^2}{\sqrt{P}} \log_2 \sqrt{P}$$

Sample data: (T_k, N_k, P_k) $k = 1, \dots, K$


$$\min_{c_1, c_2, c_3} \sum_{k=1}^K \left(\frac{T_k - T_{\text{trd}}(N_k, P_k)}{T_k} \right)^2$$

Set	c_1	c_2	c_3
1	4.06×10^{-11}	5.10×10^{-5}	4.18×10^{-8}
2	4.07×10^{-11}	5.58×10^{-5}	3.41×10^{-8}
3	4.01×10^{-11}	5.21×10^{-5}	3.92×10^{-8}

Summary of modeling in each situation

Summary of modeling in each situation

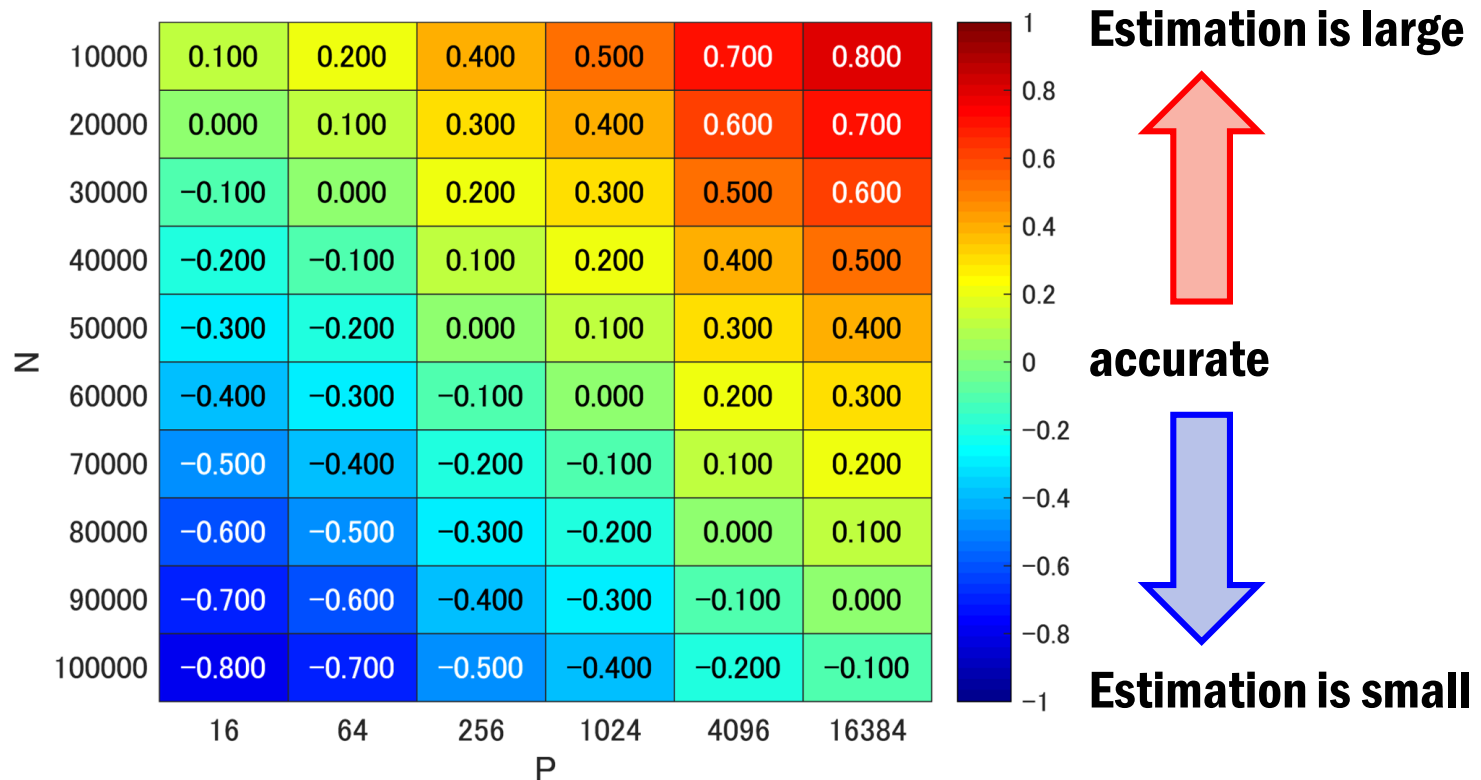
	Model form	Features	Required resources
1	$T_{\text{flop}}(N, P) + T_{\text{comm}}(N, P)$	<ul style="list-style-type: none"> Parameters: determined from specifications 	Nothing
2	$T_{\text{flop}}(N, P) + T_{\text{comm}}(N, P)$	<ul style="list-style-type: none"> Parameters: determined from benchmark results of BLAS & MPI ping-pong 	2 nodes
3	$T_{\text{flop}}(N, P) + T_{\text{al}}(N, P) + T_{\text{bc}}(N, P)$	<ul style="list-style-type: none"> T_{flop}: measured by modified eigen_trd Parameters: determined from benchmark results of MPI collective routines 	Up to \sqrt{P} nodes
4	$\sum_{i=1}^3 c_i \cdot F_i(N, P)$	<ul style="list-style-type: none"> Parameters: determined by LS fitting using sample data 	Depending on sample data

Evaluation of the performance models

Evaluation metric

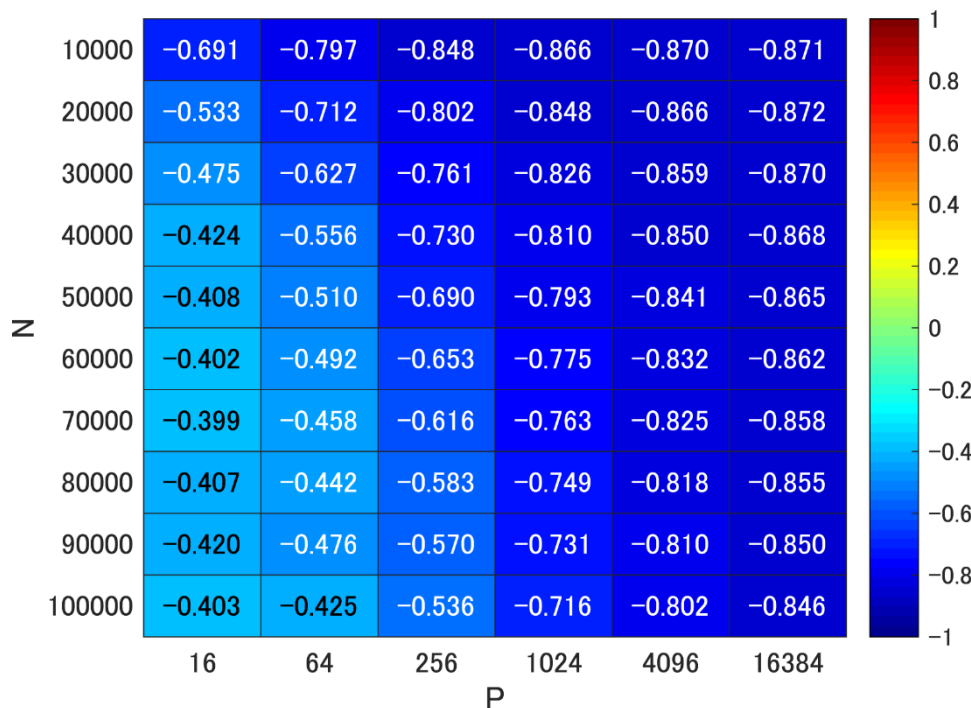
◆ Compare the estimation with measured timing data

$$(\text{relative error}) = \frac{(\text{estimated time}) - (\text{measured time})}{(\text{measured time})}$$



Evaluation of the model in Situation 1

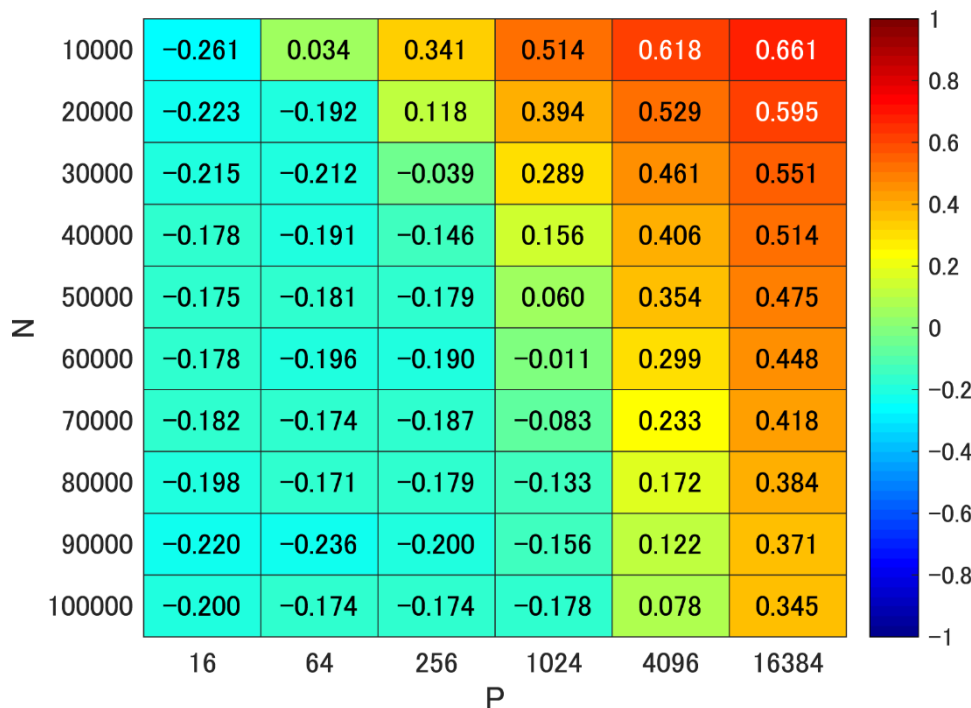
	Model form	Features	Required resources
1	$T_{\text{flop}}(N, P)$ + $T_{\text{comm}}(N, P)$	• Parameters: determined from specifications	Nothing



- Accurate modeling using only the specifications is quite difficult.
- Estimation by the model is much smaller than the actual time.
(Because theoretical performance was used.)

Evaluation of the model in Situation 2

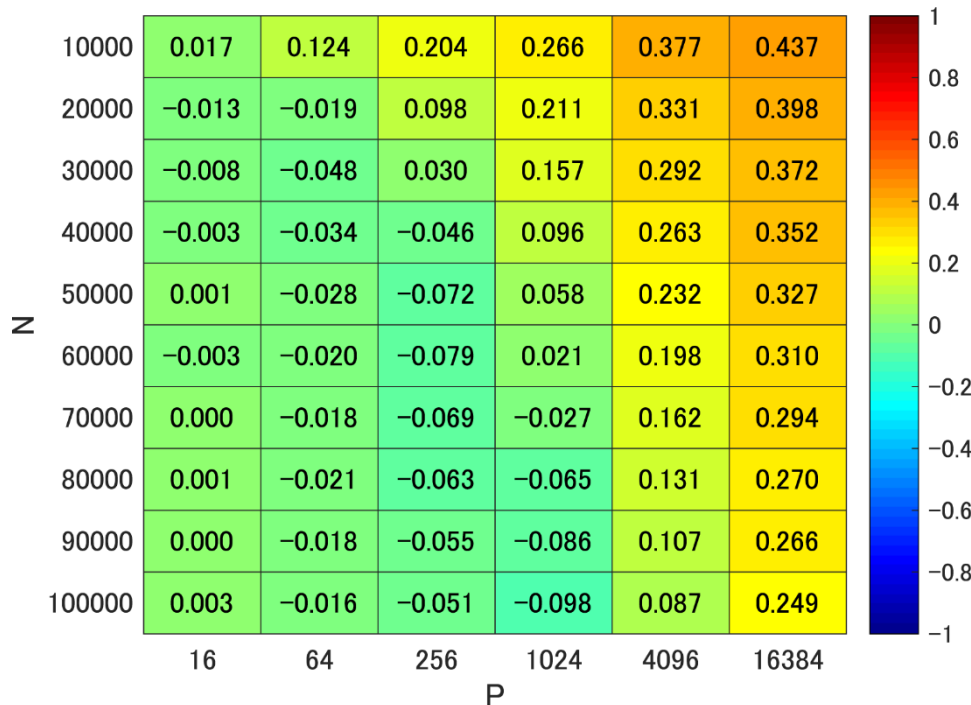
	Model form	Features	Required resources
2	$T_{\text{flop}}(N, P)$ + $T_{\text{comm}}(N, P)$	<ul style="list-style-type: none"> Parameters: determined from benchmark results of BLAS & MPI ping-pong 	2 nodes



- Estimation error is about 20% for some cases.
- Error tends to be larger when P is large and N is small.
(Difficulty of estimation of communication cost?)

Evaluation of the model in Situation 3

	Model form	Features	Required resources
3	$T_{\text{flop}}(N, P)$ $+ T_{\text{al}}(N, P)$ $+ T_{\text{bc}}(N, P)$	<ul style="list-style-type: none"> T_{flop}: measured by modified eigen_trd Parameters: determined from benchmark results of MPI collective routines 	Up to \sqrt{P} nodes



- For all cases excepting P is large and N is small, estimation is accurate. (error is in 5~10%.)
- One of main reasons for accurate estimation would be the measurement of the time for T_{flop} .

Evaluation of the model in Situation 4

	Model form	Features	Required resources
4	$\sum_{i=1}^3 c_i \cdot F_i(N, P)$	<ul style="list-style-type: none"> Parameters: determined by LS fitting using sample data 	Depending on sample data

Sample data used for fitting



- For every case, estimation is very accurate.
(error is within 10%)

Effect of sample data set in Situation 4



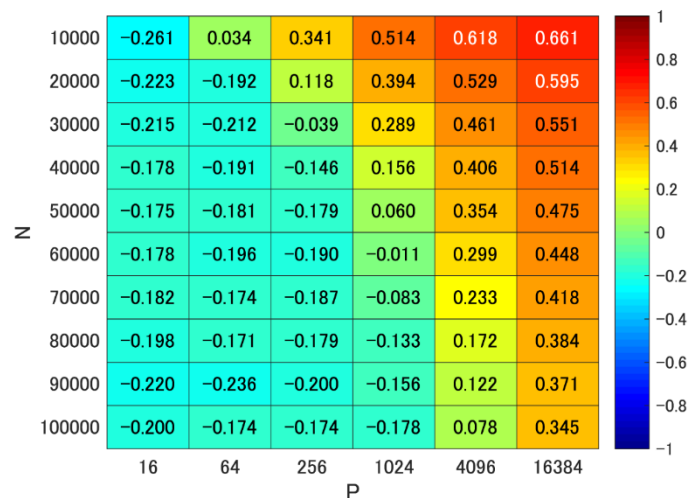
- No significant differences are found.
(Differences of sample data set rarely impacted the accuracy of estimation.)

Overall comparison

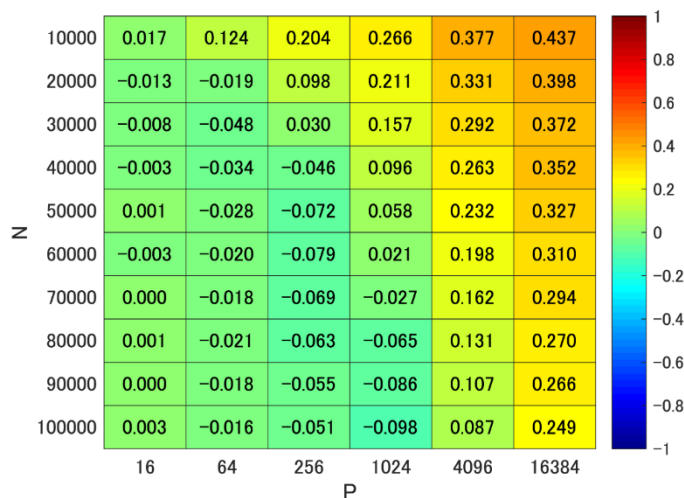
Model 1



Model 2



Model 3



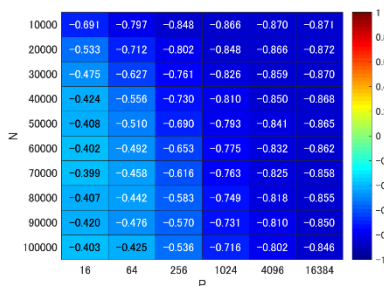
Model 4 (data set 1)



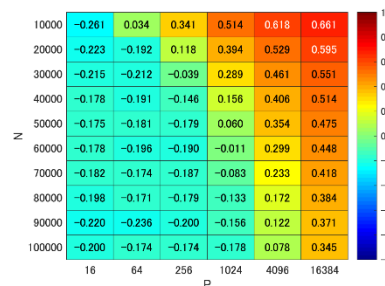
Conclusion

Summary and future work

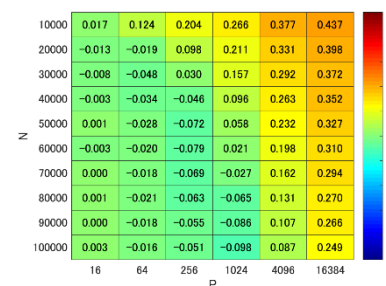
- We attempted to model the performance of eigen_trd (tridiagonalization routine) in EigenExa on the K computer.
- We considered 4 situations, where different limited information is available for performance modeling.
- We evaluated the estimation by each model comparing with the measured timing data.
- Evaluation with other dense matrix computation and on other computer systems remains to be investigated.



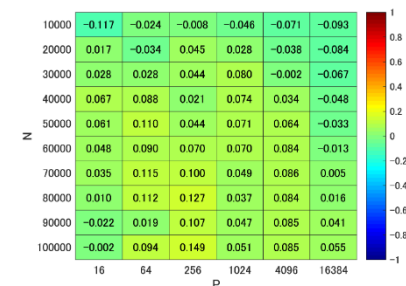
Specifications



Simple benchmark
(BLAS & ping-pong)



Specified benchmark
(modified program &
MPI collective routine)



Fitting using
measured sample data