Introduction to large-scale computing
大規模計算序論

Institute for NanoScience Design, Osaka University
大阪大学ナノサイエンスデザイン教育研究センター
Masaaki Geshi
下司　雅章

34th CMD workshop, supercomputer course
2019, February 18th (Mon.)

Contents

- Necessity of large-scale computing
- Methods of the large-scale computing(Mainly parallelization)
- Today's supercomputers
- Future direction of large-scale computing

Necessity of large-scale computing

- Systems we want to study have many atoms.
- The computational cost increases N^3, N^6, or exponentially.(N is the number of atoms, electrons,…)
- Some physical quantities needs a huge computational cost in spite of a small system(high accuracy).
- There is scientific and/or industrial significance obtained by large-scale computing.

Problems raised by large-scale computing

- Because of huge date must be handled, huge HDD and fast IO is needed. (~30PB in K computer, ~150PB in post-K)
- The problem of data analysis including visualization.
- In some cases, data analysis costs more time than computational time.
- …

When we perform large-scale computing,…

- It costs huge computational time.
- In some cases, huge physical memory of CPU is needed.

- In order to solve these problems, we have to improve software codes to reduce the computational time and memory.

- Parallel computing is definitely necessary.

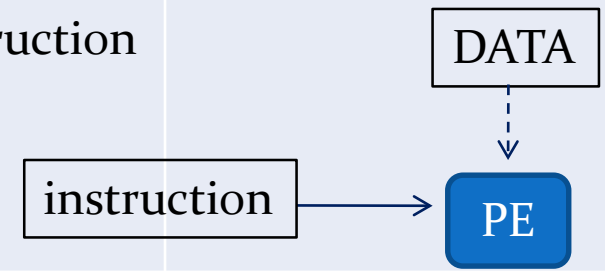
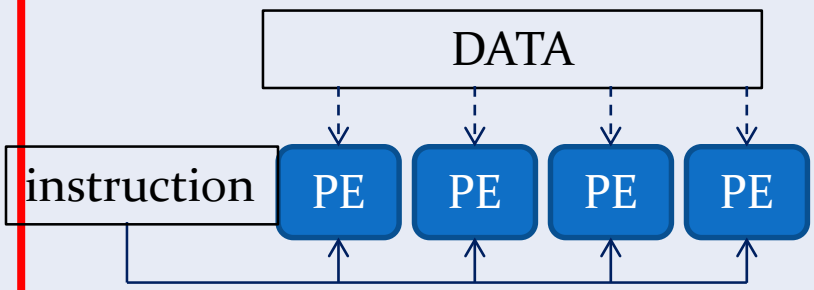
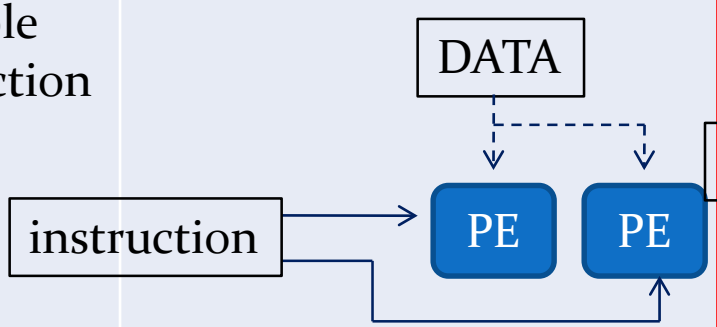
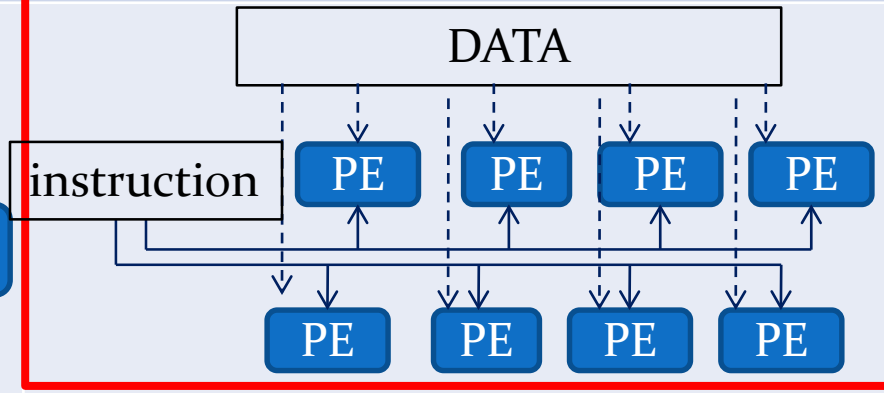
Methods for large-scale computing

- Speed-up techniques
  - Vectorization→SIMD(many cores)
  - Parallelization
  - Order-N method(reduce the cost $o(N^p)$→$o(N)$(p=1))
  - Use of specialized machine or accelerator(GPU).
- These are used in a code depending on one's necessity.

The easiest way to speed up is to be able to realize it by choosing proper compile options(vectorization, Automatic parallelization, optimization,…).

Since it is not possible to speed up the code automatically, we have to tune the code significantly.

Concept of Flynn's taxonomy

	Single data	Multiple data
	SISD	SIMD
Single instruction		
	MISD	MIMD
Multiple instruction		

Vectorization

- A vector processor, or array processor, is a CPU that implements an instruction set containing instructions that operate on one-dimensional arrays of data called *vectors*(row, column, or diagonal elements, etc…).
- Vectorization is the method that we program a code to construct regularly-arrayed data, and process the data at a time by the vector CPU.
- Vectorization realizes the reduction of a CPU time.
- Today, this concept is included in SIMD.

Vector processing

```
do i=1,imax
   a(i) = c(i) + d(i)
   b(i) = c(i) × d(i)
enddo
```

We tune to increase the part being processed at the deepest do-loop in multi do-loop.

Scalar processing

```
a(1) = c(1) + d(1)
b(1) = c(1) × d(1)
a(2) = c(2) + d(2)
b(2) = c(2) × d(2)
⋮
a(imax) = c(imax) + d(imax)
b(imax) = c(imax) × d(imax)
```

Vector processing

```
a(1) = c(1) + d(1)
a(2) = c(2) + d(2)
:
:
a(imax) = c(imax) + d(imax)
b(1) = c(1) × d(1)
b(2) = c(2) × d(2)
⋮
b(imax) = c(imax) × d(imax)
```

"a" array

"b" array

Order-*N*

- Realized by algorithm

→No approximation Ex.: Screened-KKR

- Realized by approximation of Hamiltonian

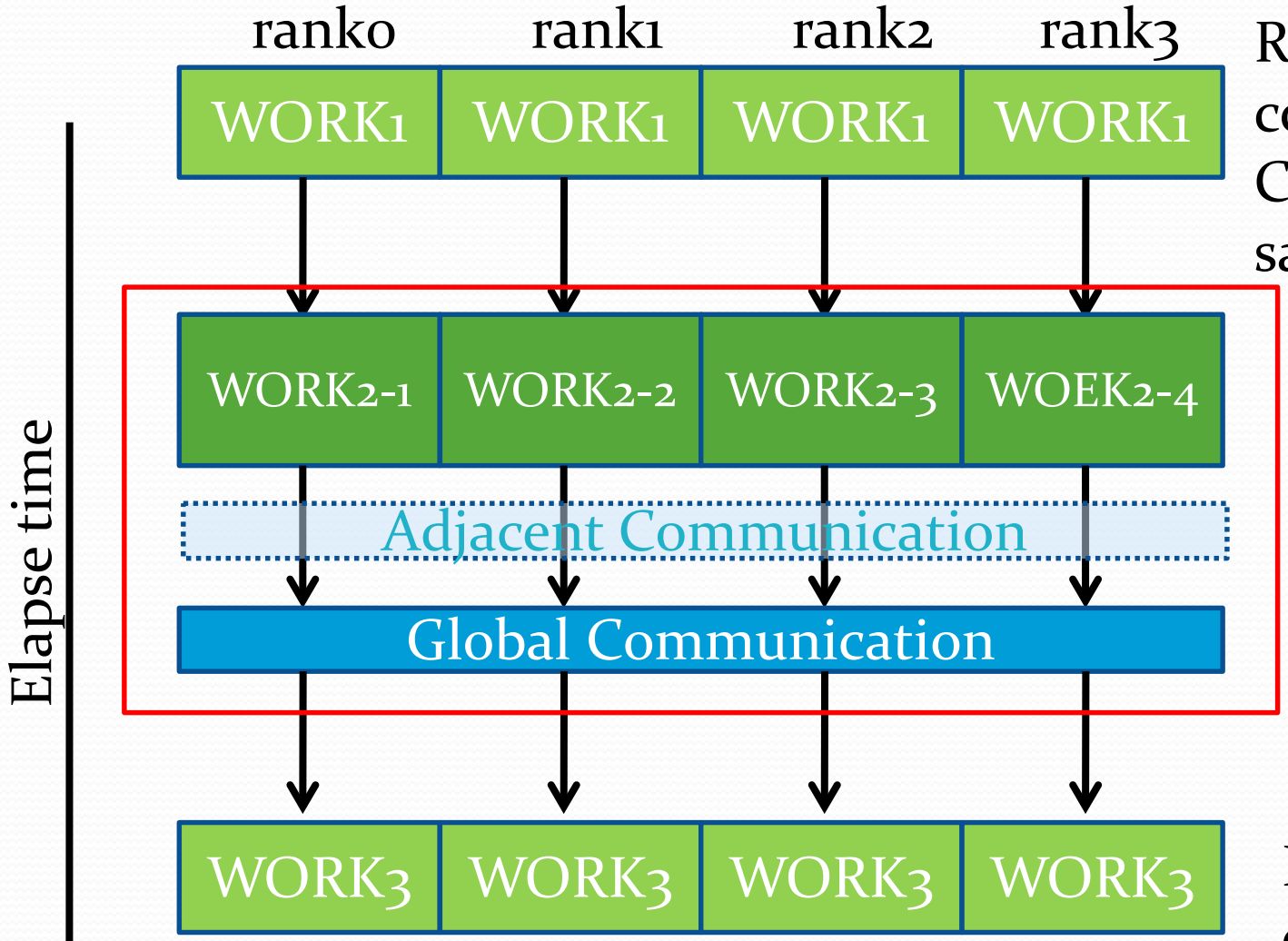
→Approximation by localization of interaction. This type of methods were actively studied in 90's.

  - Roughly speaking, matrices are block-diagonalized or including similar method.
  - $N \times N \rightarrow N \times N_{\mathrm{local}}$(fixed). If N increases, N_{local} not increase. (like a tight-binding approximation)
  - The accuracy of the approximation can be adjustable depending on a required accuracy.
  - One of the famous method is a Divide and conquer method（分割統治法）. This is used in "OpenMX(Ozaki)", "DC(Kobayashi and Nakai)",CONQUEST(Bowler, Miyazaki, Gillan)…

Parallelization

- This is the method to shorten the elapse time by sharing task with more than one CPU without changing the processing time of the task.
- This is used to share big memory(array).
- SIMD(Single instruction, multiple data), MIMD(Multiple instruction, multiple data)

Concept of parallelization



Redundant computation (all CPUs compute the same calculation)

Parallel computation (different CPU computes different calculation)

Redundant computation

Point: Increase parallelizable time and decrease communication time.

Problems in parallelization

- The ratio of parallelizable part in an elapse time on 1CPU sequential job is conclusive (Amdahl' law) .
- When we parallelize a software code, the data communication between nodes must occur. If the number of nodes increases, the communication time between nodes costs more than the computational time.(In the case of classical MD simulation, this is a crucial problem.)

Amdahl's Law

Speed up ratio $\alpha_P \equiv \dfrac{t_1}{t_{N_{\text{core}}}}$

Elapse time of one job on 1 core

Elapse time of one job on N core

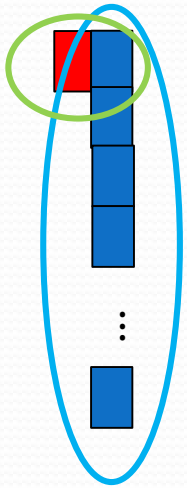
$$(\alpha_P)_{\max} = t_1 \times \left(\frac{1}{t_{N_{\text{core}}}} \right)_{\max} = \frac{1}{(1-P) + \dfrac{P}{N_{\text{core}}}}$$

P: parallelizable time in a sequential job

Upper bound of parallelizable ratio

For examples,

Not parallelizable (1 sec.) parallelizable (99 sec.)



100 processors

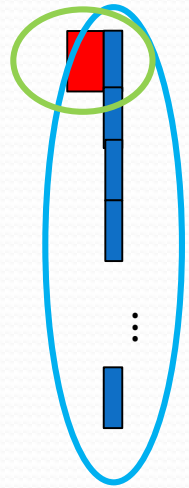
1+0.99=1.99 sec.(almost 50 times faster)

Even w

obtain

We have to increase the parallelizable part as possible.

We have to decrease the non parallelizable part as possible.



1000 processors

1+0.099=1.099 sec. (almost 91 times faster)

Although we uses 1000 processors, the efficiency does not become 100 times even!!!

We feel that the merit of parallelization does not increase even if the number of processors increases more and more.

Gustafson-Barsis' law

Speed-up ratio $\leq n + (1-n)s$

n: the number of cores

s: the ratio of sequential computation(this part cannot parallelize.)

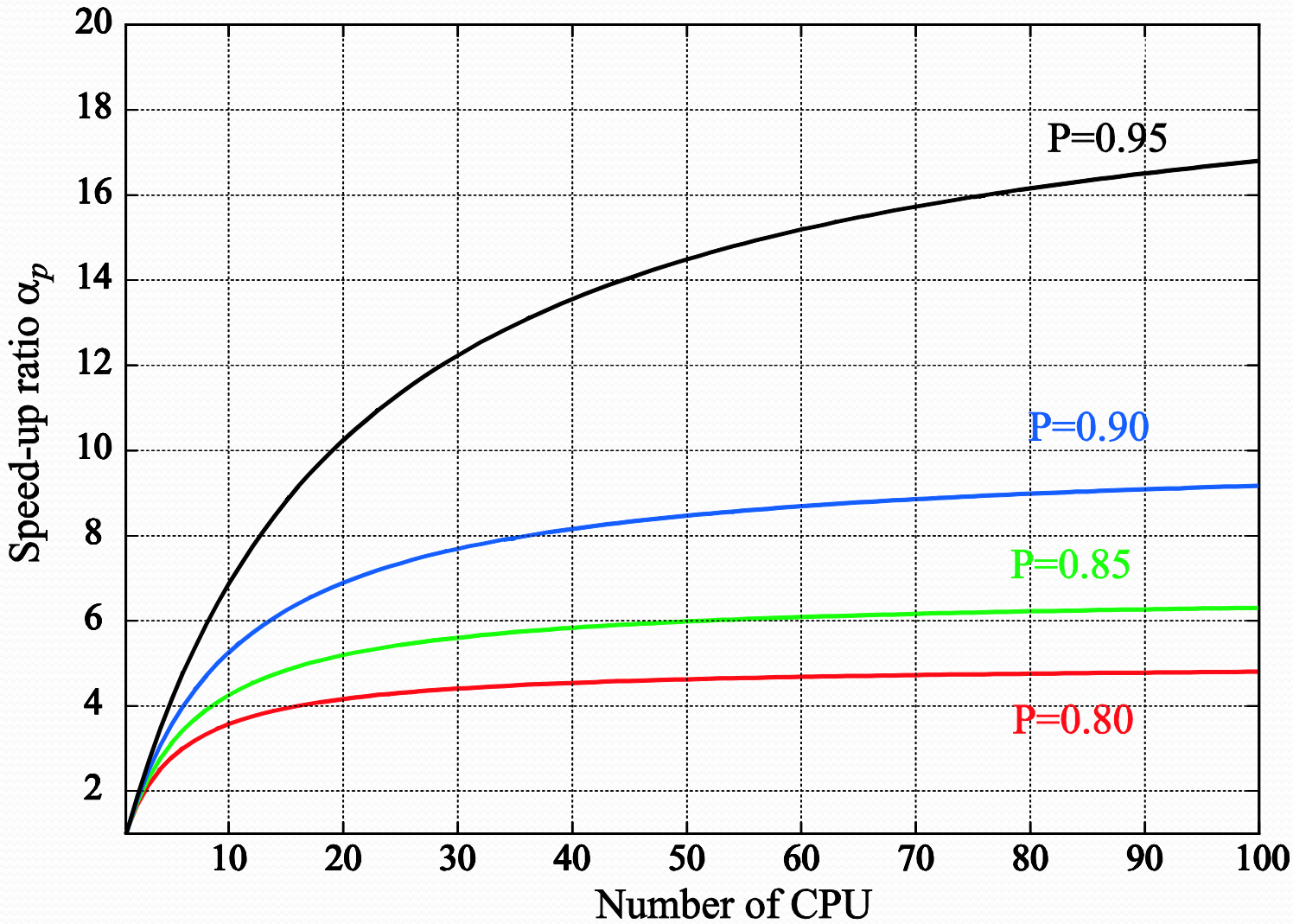
In this law, the communication time among nodes does not considered.

Indices of parallelization efficiency

- Strong scaling－The overall problem size (number of atoms or electrons etc…) is fixed and the number of processors is increased(based on Amdahl's law).
- Weak scaling－The problem size per processor is fixed and the overall problem size is increased with increasing processors(based on Gustafson's law). The processing of one processor is not changed. The ideal situation is the computation time should be constant even if the increase of the number of processors. If the computation time increases, we can see the non parallel part remains, and if the adjacent communication time increases, the method of the communication has a problem.

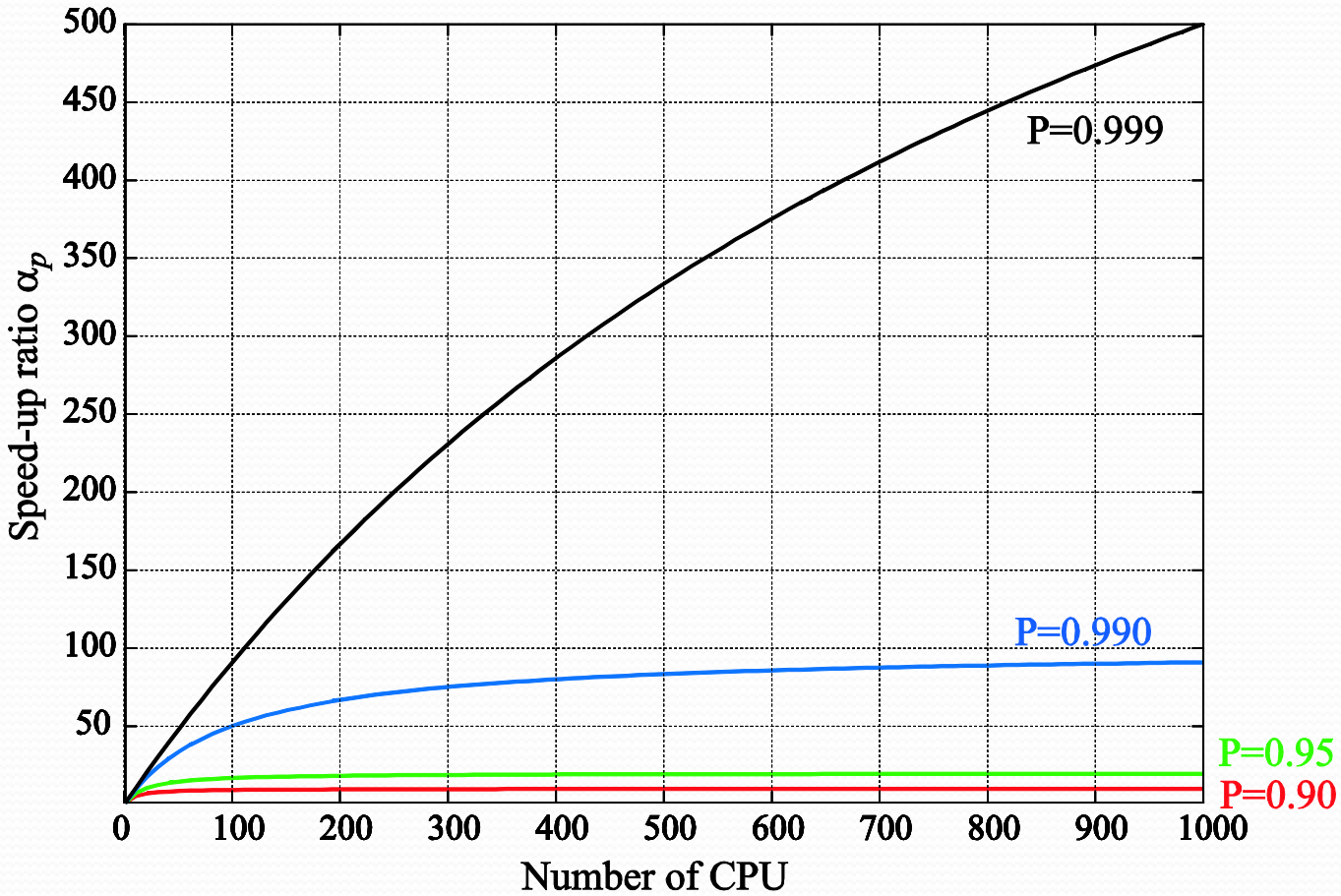
Reference: http://www.r-ccs.riken.jp/library/event/tokuronB_180406.html

Example of Amdahl's law(1)



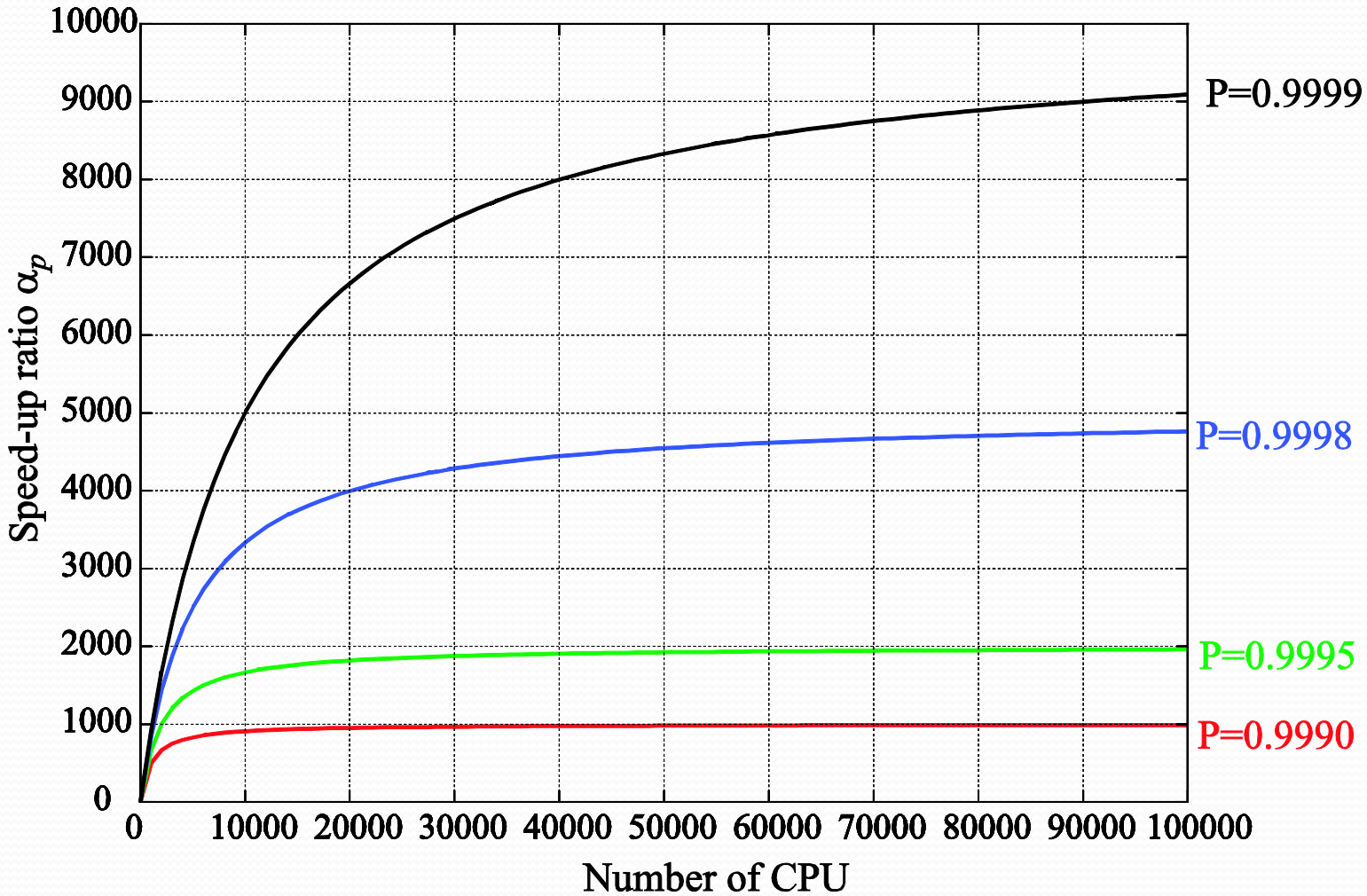
Ordinary calculations using PC cluster

Example of Amdahl's law(2)



Calculations using high-end PC cluster or supercomputers

Example of Amdahl's law(3)

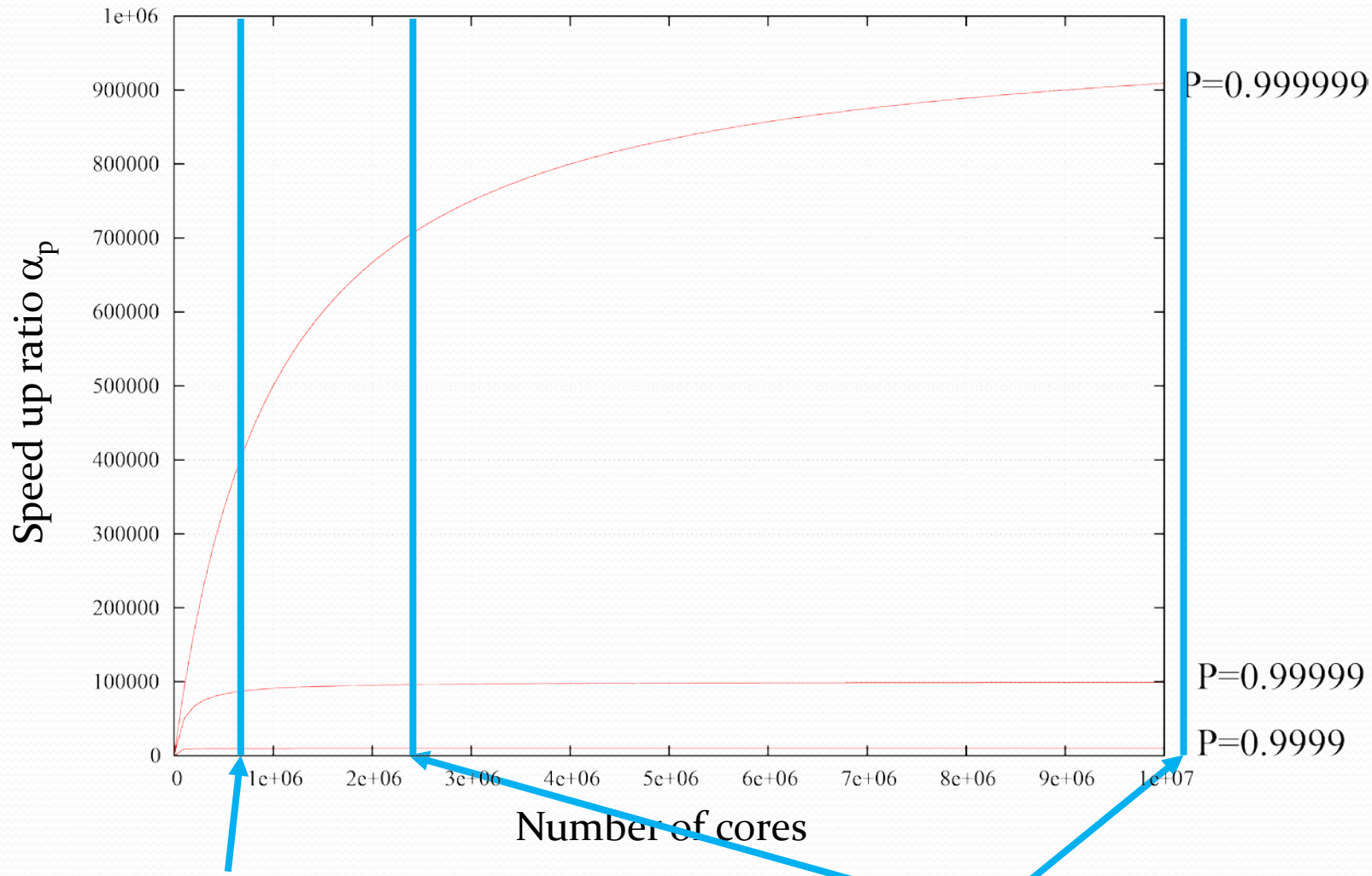


K computer: 705,024 cores

Calculations using high-end supercomputers

Example of Amdahl's law(4)

Calculations using top class supercomputers in the world



K computer: 705,024 cores

Summit: 2,397,824 cores

Sunway TaihuLight: 10,649,600 cores

Parallel programming languages

- MPI(message passing interface)
  - The most versatile method.
  - Both core-to-core and node-to-node communication can be handled.
  - The programming may not be easy.
- OpenMP
  - Only core-to-core communication.
  - The programming may be easier than that of MPI.
- PVM
- HPF(High Performance Fortran)

(within non-complex case)

Now, these are seldom used.

Example of parallelization with MPI

```
program main
include (mpif.h)
        ...
call MPI_INIT(IERR)
call MPI_COMM_SIZE(MPI_COMM_WORLD,NPROCS,IERR)
call MPI_COMM_RANK(MPI_COMM_WORLD,MYRANK,IERR)
        ...
call para_range(1, n, nprocs, myrank, ista, iend)
do i=ista, iend
        ...
   a(i)=a(i)+value
enddo
 call MPI_ALLREDUCE(a,a1,n_element,
     & MPI_DOUBLE_PRECISION,
     & MPI_SUM, MPI_COMM_WORLD, IERR)
    ...
call MPI_FINALIZE(IERR)
    ...
end
```

ista, iend are the first array numbers allocated to each processor.

Each processor has only allocated elements of the array a() .

Sum up all elements of the array a() for all processors and distributed it for all processors (all processors have the same value of a()).

It is necessary to rewrite so as to calculate only the part allocated to each node and to communicate the data.

Execution: mpirun (or mpiexec) –np 4(# of parallel) ./a.out

Example of parallelization with OpenMP

```
program main
implicit none
integer omp_get_thread_num,I
double precision z(100), a, x(100), y
 do i= 1, 100
    z(i) = 0.0
    x(i) = 2.0
  end do
 a = 4.0
 y = 1.0
print *, "Welcome to the parallel world"
!$omp parallel
print *, "Excuted on", omp_get_thread_num()
 call daxpy(z, a, x, y)
!$omp end parallel
end program main

subroutine daxpy(z, a, x, y)
integer I
double precision z(100), a, x(100), y
!$omp parallel do
 do i = 1, 100
    z(i) = a * x (i) + y
 enddo
 return
end
```

For simple parallelization, we only need to insert directives, so if we compile without parallelization options the directives are commented out, so basically we do not need to change the original code.

Directive of the start of parallel processing

Directive of the end of parallel processing

Do parallel of just below do loop

Execution: We do not use a special command. Set environmental value "OMP_NUM_THREADS=4" (# of parallel)

The difference between MPI and OpenMP

- OpenMP can be parallelizable only for cores which share the memory. We cannot make a large-scale parallelization code by using OpenMP only.
- The typical software codes which correspond to the large-scale parallelization are tuned by using MPI for its main part.
- The role of OpenMP is to assist the speed up of the code which is parallelized by using MPI.
- Even if you do not develop the software code, you have to know the details of the code in order to get maximum performance.

The other environments to develop parallel computing codes using GPU

easy

- CUFFT, CUBLAS,…
  - Only call libraries to accelerate specific parts. The other parts do not be accelerated.
- OpenACC
  - a programming standard for parallel computing developed by Cray, CAPS, NVIDIA and PGI, designed to simplify parallel programming of heterogenious CPU/GPU system.
  - Very similar with OpenMP. In near future, OpenMP and OpenACC may be merged.
  - Fortran and C are supported.
- CUDA
  - a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce. (only PGI Fortran can used.))
- OpenCL
  - a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), DSPs and other processors.

hard

OpenACC(Open Accelerator)

```
program picalc
    implicit none
    integer, parameter :: n=1000000
    integer :: i
    real(kind=8) :: t, pi
    pi = 0.0
  !$acc parallel loop
    do i=0, n-1
      t = (i+0.5)/n
      pi = pi + 4.0/(1.0 + t*t)
    end do
  !$acc end parallel loop
    print *, 'pi=', pi/n
  end program picalc
```

As the development environment, the best results can be obtained by improving according to the message issued by the compiler.

http://www.nvidia.co.jp/object/openacc-gpu-directives-jp.html
http://www.cms-initiative.jp/ja/events/2014-haishin
http://www.r-ccs.riken.jp/library/event/tokuronB_180406.html
(The 14th lecture given by Dr. A. Naruse is useful.)

The cutting-edge techniques

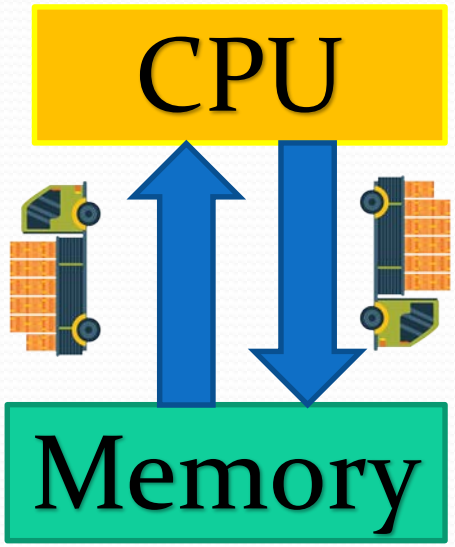
- Avoid memory wall problem(cache control)
  - Computing power>data transfer from memory to computing unit(CPU)→reuse the data on caches
- Pipeline processing
- Continuous access in do loop
- loop unrolling
- Divide data into blocks
- Use the highly-optimized libraries
- …

We have to know details of hardware to make highly optimized software codes.

Please see these sites; (only Japanese)
http://www.r-ccs.riken.jp/library/event/tokurona_170406.html
http://www.r-ccs.riken.jp/library/event/tokuronB_180406.html

Memory wall problem

Old computer

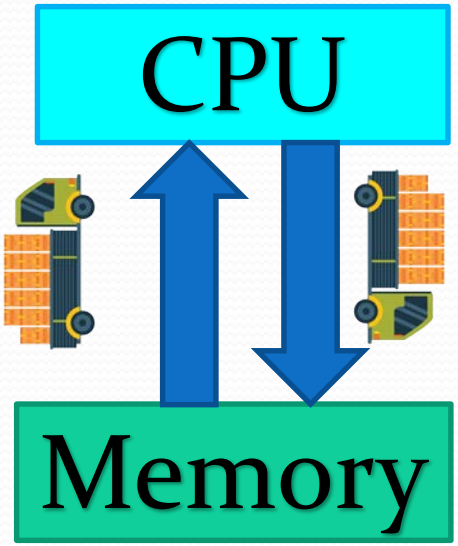


Computation performance (number of clocks) was slow (~10MHz)

The memory can not be accessed until a certain time passes

Sending data from memory to the CPU was slow, and the computation speed was also slow, so it was not the problem at "data transfer speed ≈ calculated speed".

Today's computer



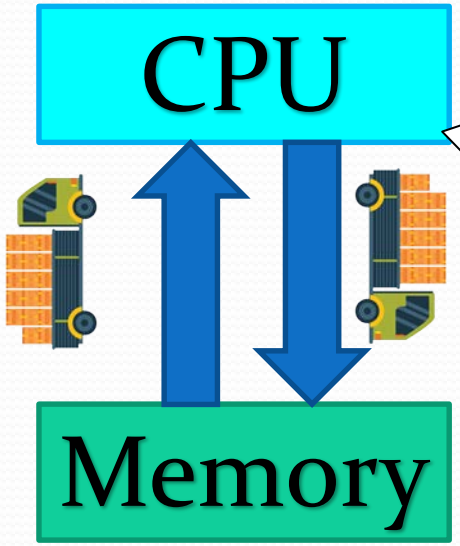
The computation performance becomes extremely fast (~a few GHz)

The fact that memory cannot be accessed only for a fixed time does not change much.

Sending data from the memory to the CPU is slow, but since the computing speed has become extremely fast, the CPU cannot wait "the transfer speed << compute speed" (supplement with cache etc…).

Memory wall pr

Today's computer

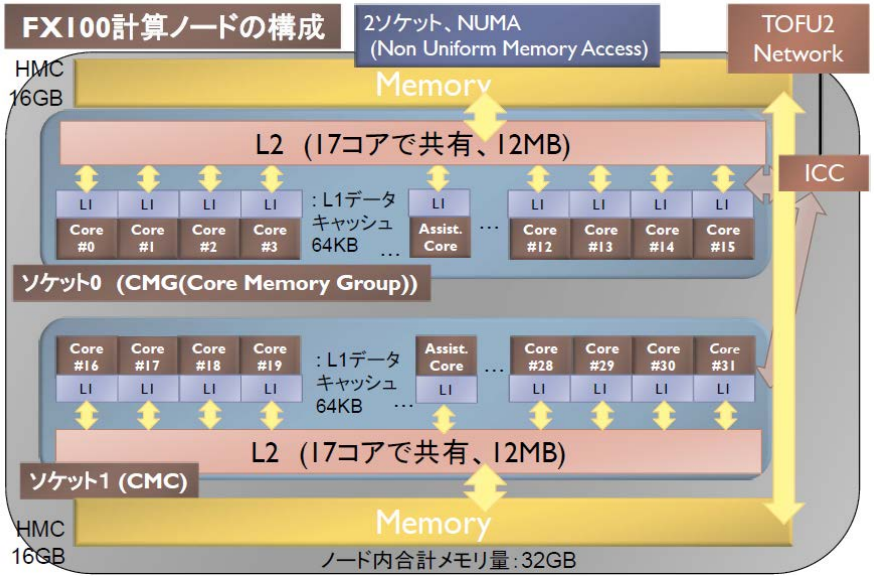


The comp
performa
becom

Th
mem
accesse
fixed ti
change

Sending data from the memory
CPU is slow, but since the comput
speed has become extremely fast, th
CPU cannot wait "the transfer speed
<< compute speed" (supplement with
cache etc…).

The configuration of CPU is not uniform; Non Uniform Memory Access(NUMA).



level cache

e

MByte

yte~TByte

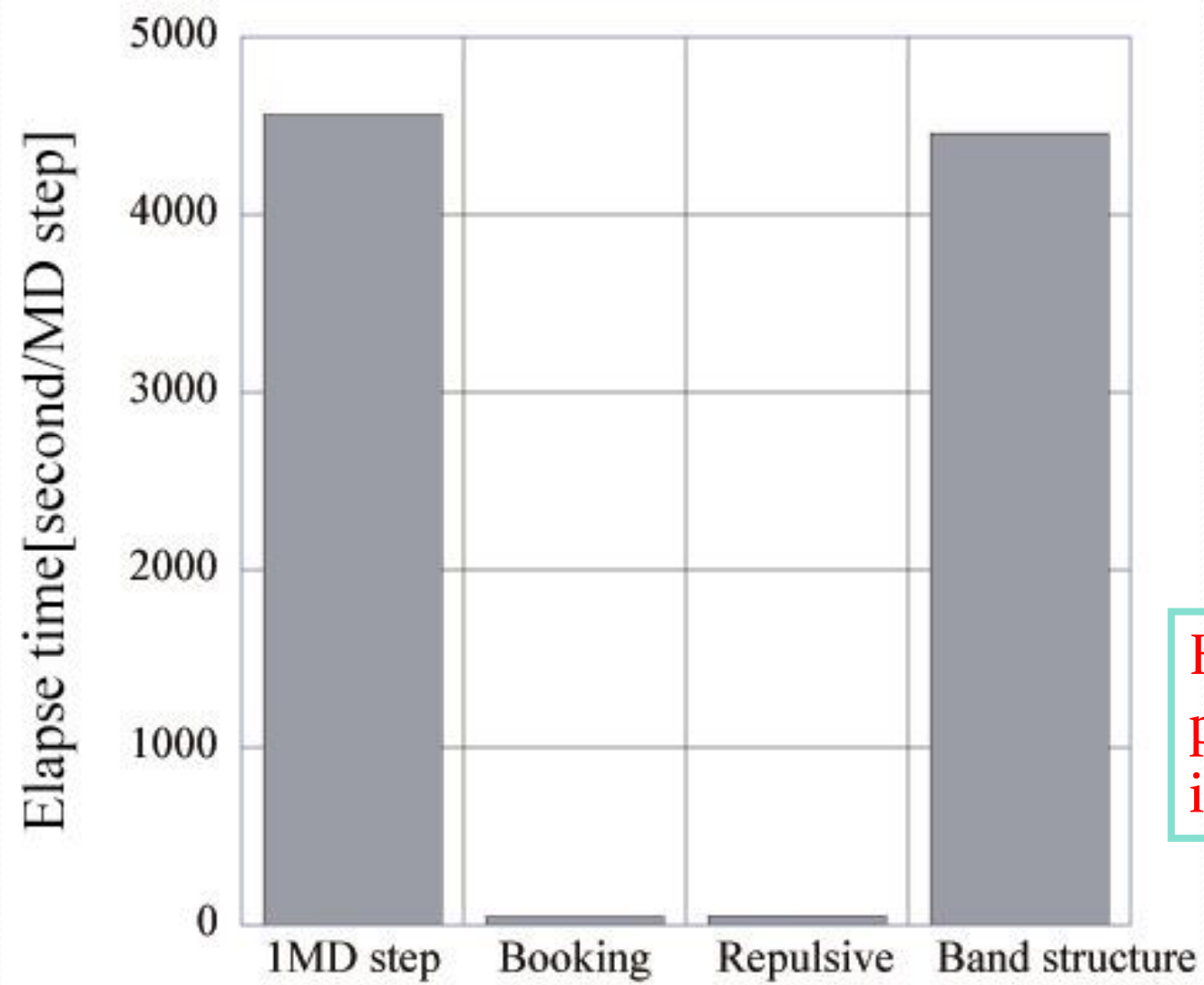
High performance Python

- In place of Fortran and C / C ++, which have been used in scientific computing in the past, HPC software development has been advanced in the USA by using Python.
- Python has overwhelming code readability compared to Fortran and C / C ++.

→It is more likely that the maintenance of the code will be continued without reading by experts.

- We do not need write everything in Python. If we know that another language is faster than Python for a certain part, we should use the part in the original. We combine Python and the other language and improve the ease of maintenance. (It is said that Python is a "glue language" because it is a language that "sticks" programs in multiple languages.)

Analysis of the computational time in order-*N* tight-binding method



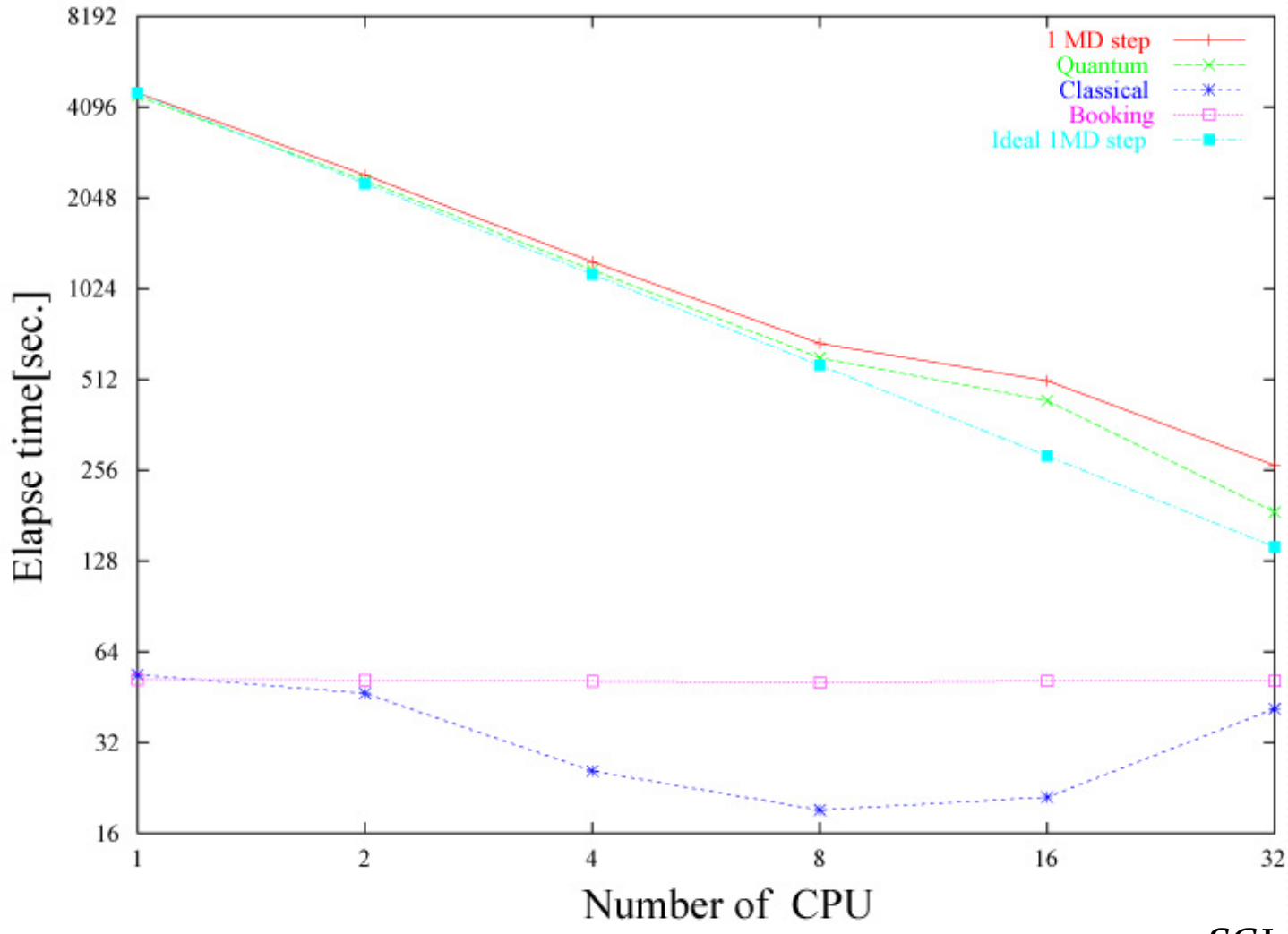
A electronic structure part occupies 97.6% of total elapse time in a sequential job.

High parallelizability is expected.

Parallelization by means of MPI

- We used MPI_ALLREDUCE and MPI_BCAST to communicate data. (Although this is quite simple and includes useless data communication, the speed-up ratio was good.)
- We divided do-loop as outer as possible.
- In this method, we can make all matrix elements as long as we determine the atomic positions. These data determined from 1 MD step are stored in CPU memory of all node. Although almost all data are useless, we do not need to communicate the data among nodes and there is no loss of data communication.

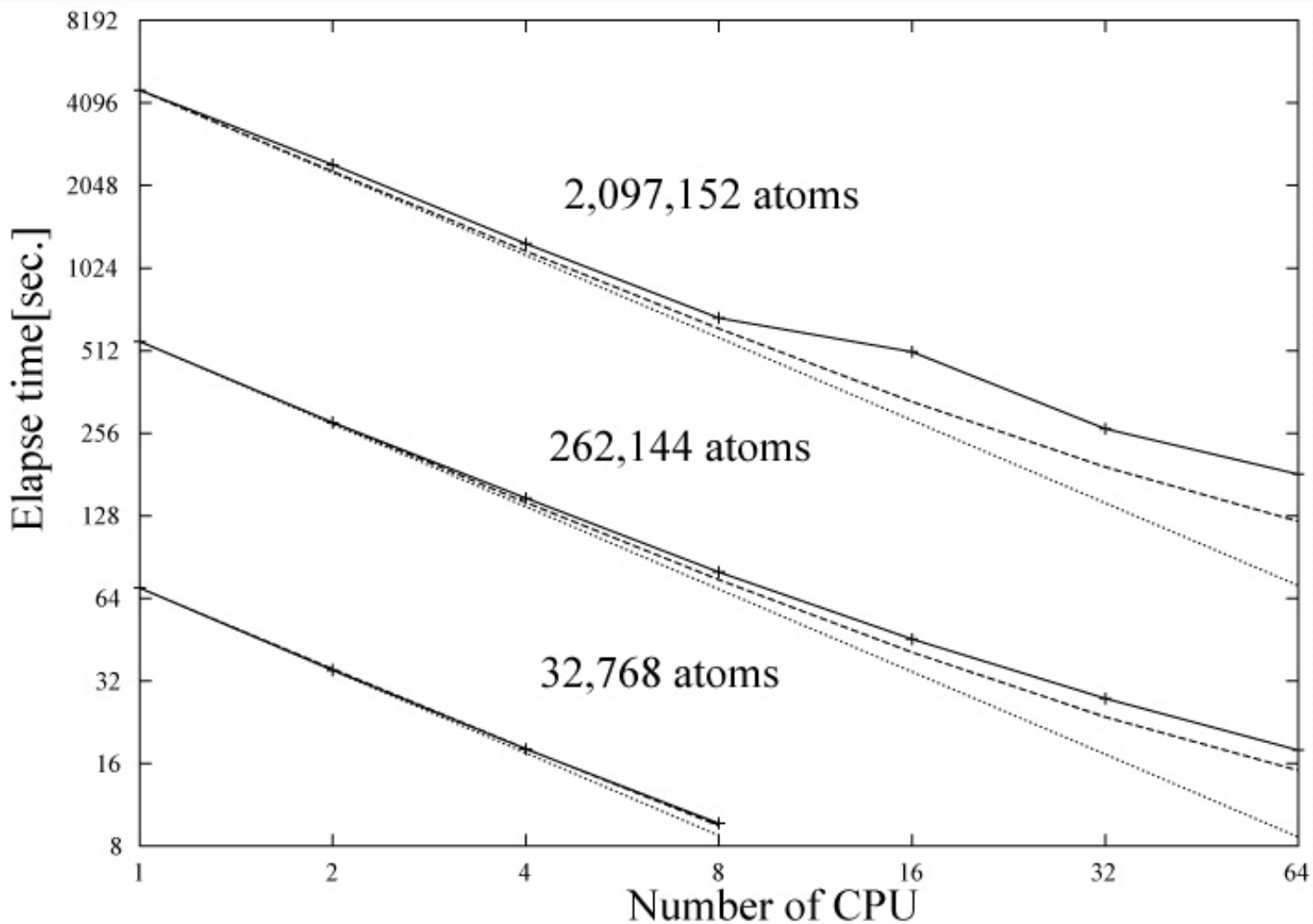
Elapse time of each part as a function of the number of CPU



SGI Origin 3800

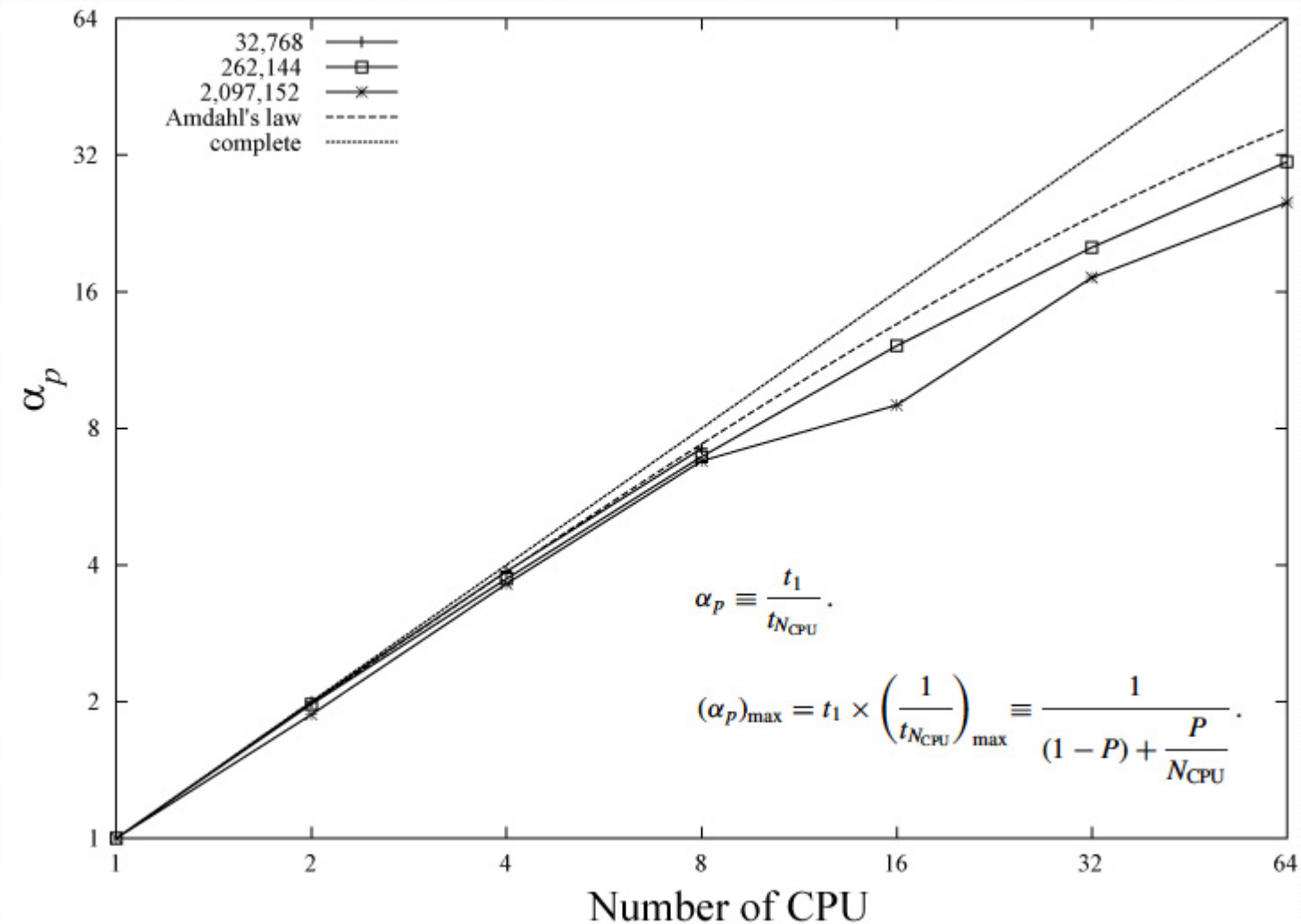
Elapse time as a function of the number of CPU

M. Geshi *et al*, J. Phys. Soc. Jpn. **72** ,2880 (2003).



Parallelization ratio

M. Geshi *et al*, J. Phys. Soc. Jpn. **72** ,2880 (2003).



P=0.988

Today’s supercomputers

Classes of parallel computers

- Multicore computing
  - a processor that includes multiple execution units ("cores") on the same chip.
- Symmetric multiprocessing
  - a computer system with multiple identical processors that **share memory** and connect via a bus.
- Distributed computing
  - a distributed memory computer system in which the processing elements are connected by a network.(**Cluster computing, Massive parallel processing, Grid computing**)
- Specialized parallel computers
  - Within parallel computing, there are specialized parallel devices that remain niche areas of interest. (**GPGPU,Application-specific integrated circuits, Vector processors**)

From http://en.wikipedia.org/wiki/Parallel_computing

K computer

- K computer is massive parallel computer which has 705,024 cores(88,128CPUs).
- In order to use this highly efficiently, a source code has to be tuned with extremely high parallel efficiency and very less communication.
- GPU is not used in K computer. Therefore, we can use standard Fortran and C++. (We do not need to learn other language like CUDA.) However, **we have to understand the specific properties of K computer.**
- K computer is open for the use of general uses from Sep. 28th, 2012.
- K computer will be stop at August 16th,2019, and replace post-K computer until 2021.

The latest top 500 supercomputer list (2018.11)(Latest)

http://www.top500.org/

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)	
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,82	143,500.0	200,794.9	9,783	USA
2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	712.0	7,438	USA
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,60	93,014.6	125,435.9	15,371	China
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482	China
5	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384	
6	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/NNSA/LANL/SNL United States	979,072	20,158.7	41,461.2	7,578	USA
7	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	649	Japan
8	**SuperMUC-NG** - ThinkSystem SD530, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path , Lenovo Leibniz Rechenzentrum Germany	305,856	19,476.6	26,873.9		
9	**Titan** - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x , Cray Inc. DOE/SC/Oak Ridge National Laboratory United States	560,640	17,590.0	27,112.5	8,209	USA
10	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom , IBM DOE/NNSA/LLNL United States	1,572,864	17,173.2	20,132.7	7,890	USA

IBM

Cray

GPU

GPU

Designed for data and AI

Intel Xeon(many-cores)

GPU

Intel Xeon(many-cores)

GPU

By the way, K-computer is 18, Oakforest PACS is 14.

GPU

Linpack

- A software library for performing numerical linear algebra on digital computers.
- It is used for the performance assessment of supercomputers. In practice, High-Performance Linpack(HPL) is used.
- It makes use of the BLAS libraries for performing basic vector and matrix operations.
- The benchmark used in the LINPACK Benchmark is to solve a dense system of linear equations.

This benchmark shows nothing more than one side of performance of supercomputers.

HPC Challenge

The HPC Challenge benchmark consists of basically 7 tests:

- HPL - the Linpack TPP benchmark which measures the floating point rate of execution for solving a linear system of equations.
- DGEMM - measures the floating point rate of execution of double precision real matrix-matrix multiplication.
- STREAM - a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.
- PTRANS (parallel matrix transpose) - exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.
- RandomAccess - measures the rate of integer random updates of memory (GUPS)
- FFT - measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).
- Communication bandwidth and latency - a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns; based on b_eff (effective bandwidth benchmark).

http://www.hpcchallenge.org/

HPCG Benchmark(From 2014)

The High Performance Conjugate Gradients (HPCG) Benchmark project is an effort to create a new metric for ranking HPC systems. HPCG is intended as a complement to the High Performance LINPACK (HPL) benchmark, currently used to rank the TOP500 computing systems. The computational and data access patterns of HPL are still representative of some important scalable applications, but not all. HPCG is designed to exercise computational and data access patterns that more closely match a different and broad set of important applications, and to give incentive to computer system designers to invest in capabilities that will have impact on the collective performance of these applications.

HPCG is a complete, stand-alone code that measures the performance of basic operations in a unified code:

•Sparse matrix-vector multiplication.
•Vector updates.
•Global dot products.
•Local symmetric Gauss-Seidel smoother.
•Sparse triangular solve (as part of the Gauss-Seidel smoother).
•Driven by multigrid preconditioned conjugate gradient algorithm that exercises the key kernels on a nested set of coarse grids.
•Reference implementation is written in C++ with MPI and OpenMP support.

http://www.hpcg-benchmark.org/index.html

HPCG ranking(2017.11)

November 2017 HPCG Results

November 2017 HPCG Results

Top500	Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Frac of P	
10	1	RIKEN Advanced Institute for Computational Science Japan	**K computer** – , SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10.510	10	0.603	5	Japan
2	2	NSCC / Guangzhou China	**Tianhe-2 (MilkyWay-2)** – TH-IVB-FEP Cluster, Intel Xeon 12C 2.2GHz, TH Express 2, Intel Xeon Phi 31S1P 57-core NUDT	3,120,000	33.863	2	0.580	1	China
7	3	DOE/NNSA/LANL/SNL USA	**Trinity** – Cray XC40, Intel Xeon E5-2698 v3 300160C 2.3GHz, Aries Cray	979,072	14.137	7	0.546	1	USA
3	4	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** – Cray XC50, Intel Xeon E5-2690v3 12C 2.6GHz, Cray Aries, NVIDIA Tesla P100 16GB Cray	361,760	19.590	3	0.486	1	
1	5	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** – Sunway MPP, SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93.015	1	0.481	0	China
6	6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** – PRIMERGY CX600 M1, Intel Xeon Phi Processor 7250 68C 1.4GHz, Intel Omni-Path Architecture Fujitsu	557,056	13.555	9	0.385	1	Japan
8	7	DOE/SC/LBNL/NERSC USA	**Cori** – XC40, Intel Xeon Phi 7250 68C 1.4GHz, Cray Aries Cray	632,400	13.832	8	0.355	1	USA
6	8	DOE/NNSA/LLNL USA	**Sequoia** – IBM BlueGene/Q, PowerPC A2 1.6 GHz 16-core, 5D Torus IBM	1,572,864	17.173	6	0.330	1	USA
5	9	DOE/SC/Oak Ridge Nat Lab USA	**Titan** – Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray	560,640	17.590	5	0.322	1	USA
13	10	GSIC Center, Tokyo Institute of Technology Japan	**TSUBAME3.0** – SGI ICE XA (HPE SGI 8600), IP139-SXM2, Intel Xeon E5-2680 v4 15120C 2.9GHz, Intel Omni-Path Architecture, NVIDIA TESLA P100 SXM2 with NVLink HPE	136,080	8.125	13	0.189	1	Japan

HPCG ranking(2018.6)

June 2018

June 2018 HPCG Results

Top500	Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Fraction of Peak	
1	1	DOE/SC/ORNL USA	**Summit** – AC922, IBM POWER9 22C 3.07GHz, dual-rail Mellanox EDR Infiniband, NVIDIA Volta V100 IBM	2,392,000	122.300	1	2.926	1.5%	USA
3	2	DOE/NNSA/LLNL USA	**Sierra** – S922LC, Power9 180224C 3.1GHz, Mellanox EDR, NVIDIA Tesla V100 IBM / NVIDIA / Mellanox	835,584	71.610	3	1.796	1.5%	USA
16	3	RIKEN Advanced Institute for Computational Science Japan	**K computer** – , SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10.510	16	0.603	5.3%	Japan
9	4	DOE/NNSA/LANL/SNL USA	**Trinity** – Cray XC40, Intel Xeon E5-2698 v3 300160C 2.3GHz, Aries Cray	979,072	14.137	9	0.546	1.8%	USA
6	5	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** – Cray XC50, Intel Xeon E5-2690v3 12C 2.6GHz, Cray Aries, NVIDIA Tesla P100 16GB Cray	361,760	19.590	6	0.486	1.9%	
2	6	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** – Sunway MPP, SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93.015	2	0.481	0.4%	China
12	7	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** – PRIMERGY CX600 M1, Intel Xeon Phi Processor 7250 68C 1.4GHz, Intel Omni-Path Architecture Fujitsu	557,056	13.555	12	0.385	1.5%	Japan
10	8	DOE/SC/LBNL/NERSC USA	**Cori** – XC40, Intel Xeon Phi 7250 68C 1.4GHz, Cray Aries Cray	632,400	13.832	10	0.355	1.3%	USA
14	9	Commissariat a l'Energie Atomique (CEA) France	**Tera-1000-2** – Bull Sequana X1000, Intel KNL 7250 522240C 1.4GHz, Bull BXI Bull, Atos Group	522,240	11.965	14	0.334	1.4%	
8	10	DOE/NNSA/LLNL USA	**Sequoia** – IBM BlueGene/Q, PowerPC A2 1.6 GHz 16-core, 5D Torus IBM	1,572,864	17.173	8	0.330	1.6%	USA

HPCG ranking(2018.11)(Latest)

Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	HPCG (TFlop/s)	
1	1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	2925.75	USA
2	2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	1795.67	USA
3	18	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect , Fujitsu RIKEN Advanced Institute for Computational Science (AICS) Japan	705,024	10,510.0	602.74	Japan
4	6	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/NNSA/LANL/SNL United States	979,072	20,158.7	546.12	USA
5	7	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	508.85	Japan
6	5	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	496.98	
7	3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	480.85	China
8	13	**Nurion** - Cray CS500, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path , Cray Inc. Korea Institute of Science and Technology Information Korea, South	570,020	13,929.3	391.45	USA
9	14	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path , Fujitsu Joint Center for Advanced High Performance Computing Japan	556,104	13,554.6	385.48	Japan
10	12	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/SC/LBNL/NERSC United States	622,336	14,014.7	355.44	USA

Green500(省エネスパコンリスト)

- The purpose of the Green500 is to provide a ranking of the most energy-efficient supercomputers in the world. For decades, the notion of "performance" has been synonymous with "speed" (as measured in FLOPS). This particular focus has led to the emergence of supercomputers that consume egregious amounts of electrical power and produce so much heat that extravagant cooling facilities must be constructed to ensure proper operation. In addition, the emphasis on speed as the ultimate metric has caused other metrics such as reliability, availability, and usability to be largely ignored. As a result, there has been an extraordinary increase in the total cost of ownership (TCO) of a supercomputer.

http://www.green500.org/

Green 500 2018.11(Latest)

Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	Power (kW)	Power Efficiency (GFlops/watts)		
1	375	**Shoubu system B** - ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 , PEZY Computing / Exascaler Inc. Advanced Center for Computing and Communication, RIKEN Japan	953,280	1,063.3	60	17.604	Japan	
2	374	**DGX SaturnV Volta** - NVIDIA DGX-1 Volta36, Xeon E5-2698v4 20C 2.2GHz, Infiniband EDR, NVIDIA Tesla V100 , Nvidia NVIDIA Corporation United States	22,440	1,070.0	97	15.113	USA	GPU
3	1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	9,783	14.668	USA	GPU
4	7	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	1,649	14.423	Japan	GPU
5	22	**TSUBAME3.0** - SGI ICE XA, IP139-SXM2, Xeon E5-2680v4 14C 2.4GHz, Intel Omni-Path, NVIDIA Tesla P100 SXM2 , HPE GSIC Center, Tokyo Institute of Technology Japan	135,828	8,125.0	792	13.704	Japan	GPU
6	2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	7,438	12.723	USA	GPU
7	446	**AIST AI Cloud** - NEC 4U-8GPU-Server, Xeon E5-2630Lv4 10C 1.8GHz, Infiniband EDR, NVIDIA Tesla P100 SXM2 , NEC National Institute of Advanced Industrial Science and Technology Japan	23,400	961.0	76	12.681	Japan	GPU
8	411	**MareNostrum P9 CTE** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Tesla V100 , IBM Barcelona Supercomputing Center Spain	19,440	1,018.0	86	11.865		GPU
9	38	**Advanced Computing System(PreE)** - Sugon TC8600, Hygon Dhyana 32C 2GHz, Deep Computing Processor, 200Gb 6D-Torus , Sugon Sugon China	163,840	4,325.0	380	11.382		
10	20	**Taiwania 2** - QCT QuantaGrid D52G-4U/LC, Xeon Gold 6154 18C 3GHz, Mellanox InfiniBand EDR, NVIDIA Tesla V100 SXM2 , Quanta Computer / Taiwan Fixed Network / ASUS Cloud National Center for High Performance Computing Taiwan	170,352	9,000.0	798	11.285		GPU

This list fluctuates violently every time. That is why emphasis is placed on the development of this technology.

These machines balance energy saving and computing performance.

Graph500 (graph analysis for big data)

Brief Introduction

Data intensive supercomputer applications are increasingly important for HPC workloads, but are ill-suited for platforms designed for 3D physics simulations. Current benchmarks and performance metrics do not provide useful information on the suitability of supercomputing systems for data intensive applications. A new set of benchmarks is needed in order to guide the design of hardware architectures and software systems intended to support such applications and to help procurements. Graph algorithms are a core part of many analytics workloads.

Backed by a steering committee of over 50 international HPC experts from academia, industry, and national laboratories, Graph 500 will establish a set of large-scale benchmarks for these applications. The Graph 500 steering committee is in the process of developing comprehensive benchmarks to address three application kernels: concurrent search, optimization (single source shortest path), and edge-oriented (maximal independent set). Further, we are in the process of addressing five graph-related business areas: Cybersecurity, Medical Informatics, Data Enrichment, Social Networks, and Symbolic Networks.

This is the first serious approach to complement the Top 500 with data intensive applications. Additionally, we are working with the SPEC committee to include our benchmark in their CPU benchmark suite. We also maintain a list of the most energy efficient systems for data intensive computing. The Green Graph 500 will be released together with the Graph500 list. We anticipate the list will rotate between ISC and SC in future years.

http://www.graph500.org/

Graph500 (2018 June)(Latest)

Performance evaluation of "graph analysis" used for analysis of big data

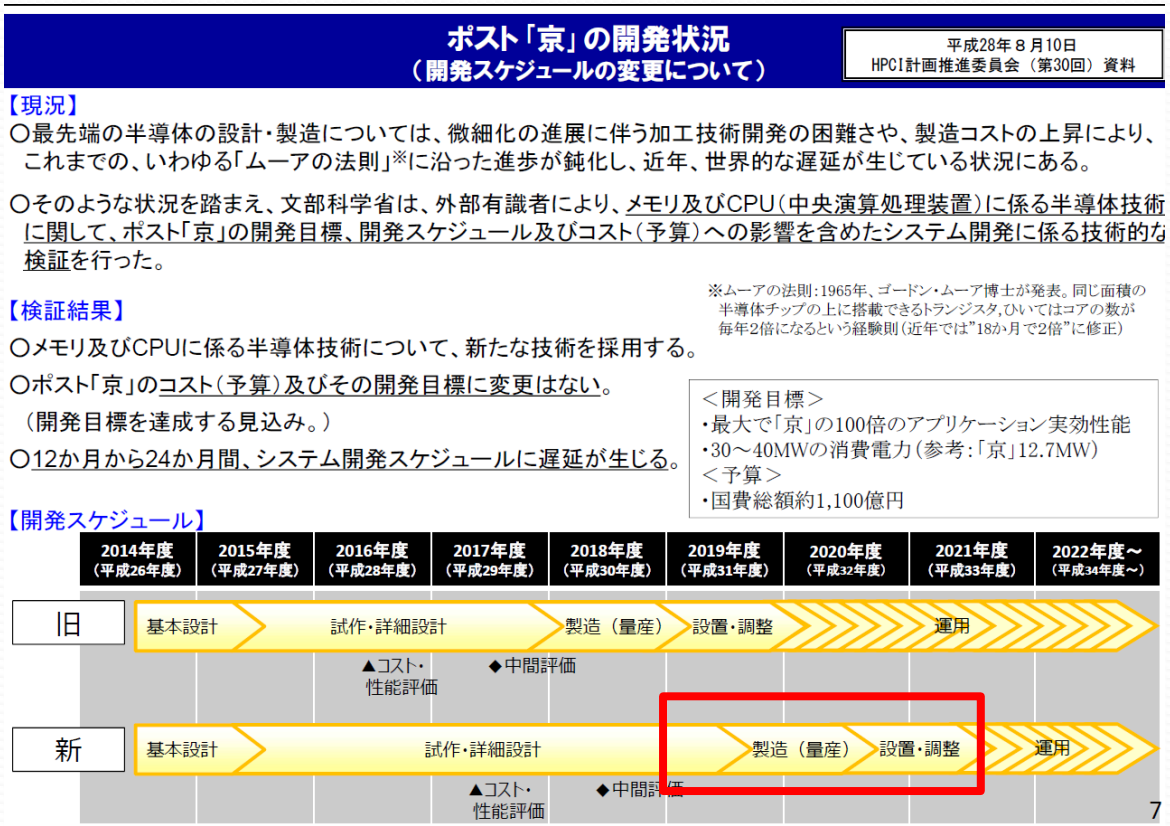
NOVEMBER 2018 BFS

Show 10 entries Search:

RANK	PREVIOUS RANK	MACHINE	VENDOR	TYPE	NETWORK	INSTALLATION SITE	LOCATION	COUNTRY	YEAR	APPLICATION	USAGE	NUMBER OF NODES	NUMBER OF CORES	MEMORY	IMPLEMENTATION	SCALE	GTEPS	C_TIME	POWER
1	1	K computer	Fujitsu	Custom	Tofu	RIKEN Advanced Institute for Computational Science (AICS)	Kobe Hyogo	Japan	2011	Various scientific and instudrial fields	Academic and industry	82944	663552	1327100 gigabytes	Custom	40	38621.4	609.395 seconds	0 Watts
2	2	Sunway TaihuLight	NRCPC	Sunway MPP	Sunway	National Supercomputing Center in Wuxi	Wuxi	China	2015	research	research	40768	10599680	1304580 gigabytes	Custom	40	23755.7	961.455 seconds	
3	3	DOE/NNSA/LLNL Sequoia	IBM	BlueGene/Q Power BQC 16C 1.60 GHz	Custom	Lawrence Livermore National Laboratory	Livermore CA	USA	2012	Scientific Research	Government	98304	1572864	1572860 gigabytes	Custom	41	23751	368.7 seconds	6389760 Watts
4	4	DOE/SC/Argonne National Laboratory Mira	IBM	BlueGene/Q Power BQC 16C 1.60 GHz	Custom	Argonne National Laboratory	Chicago IL	USA	2012	Scientific Research	Research	49152	786432	786432 gigabytes	Custom	40	14982	259.8 seconds	0 Watts
5	new	SuperMUC-NG	Lenovo	ThinkSystem SD530 Xeon Platinum 8174 24C 3.1GHz Intel Omni-Path		Leibniz Rechenzentrum	Garching	Germany	2018	Academic	Research	4096	196608	393216	custom-aml-heavy-dropt	39	6279.47	1547.10	
6	5	JUQUEEN	IBM	BlueGene/Q Power BQC 16C 1.60 GHz	Custom	Forschungszentrum Juelich (FZJ)	Juelich	Germany	2012	Scientific Research	Research	16384	262144	262144 gigabytes	Custom	38	5848	565 seconds	1081340 Watts
7	6	ALCF Mira - 8192 partition	IBM	IBM - BlueGene/Q Power BQC 16C 1.60 GHz		Argonne National Laboratory	Chicago IL	United States	2012	Scientific Research	Research	8192	131072	131072 gigabytes	Custom	36	4212	467 seconds	0 Watts
8	8	Fermi	IBM	BlueGene/Q Power BQC 16C 1.60 GHz	Custom	CINECA	Casalecchio Di Reno	Italy	2012	Scientific Research	Academic	8192	131072	131072 gigabytes	Custom	37	2567	400 seconds	0 Watts
9	new	NERSC Cori - 1024 haswell partition	Cray	XC40	Aries	NERSC/LBNL	DOE/SC/LBNL/NERSC	United States	2017	Government	basic science and simulation	1024	32768	133376	custom-aml-heavy-dropt	37	2562.16	1369.29	
10	9	ALCF Mira - 4096 partition	IBM	IBM - BlueGene/Q Power BQC 16C 1.60 GHz	5D Torus	Argonne National Laboratory	Chicago IL	United States	2012	Scientific Research	Research	4096	65536	65536 gigabytes	Custom	35	2348	462 seconds	0 Watts

http://www.graph500.org/

Development situation of post-K



ポスト「京」の開発状況
（開発スケジュールの変更について）

平成28年８月10日
HPCI計画推進委員会（第30回）資料

【現況】

○最先端の半導体の設計・製造については、微細化の進展に伴う加工技術開発の困難さや、製造コストの上昇により、これまでの、いわゆる「ムーアの法則」※に沿った進歩が鈍化し、近年、世界的な遅延が生じている状況にある。

○そのような状況を踏まえ、文部科学省は、外部有識者により、メモリ及びCPU（中央演算処理装置）に係る半導体技術に関して、ポスト「京」の開発目標、開発スケジュール及びコスト（予算）への影響を含めたシステム開発に係る技術的な検証を行った。

※ムーアの法則：1965年、ゴードン・ムーア博士が発表。同じ面積の半導体チップの上に搭載できるトランジスタ,ひいてはコアの数が毎年2倍になるという経験則（近年では"18か月で2倍"に修正）

【検証結果】

○メモリ及びCPUに係る半導体技術について、新たな技術を採用する。

○ポスト「京」のコスト（予算）及びその開発目標に変更はない。
（開発目標を達成する見込み。）

○12か月から24か月間、システム開発スケジュールに遅延が生じる。

＜開発目標＞
・最大で「京」の100倍のアプリケーション実効性能
・30～40MWの消費電力（参考：「京」12.7MW）
＜予算＞
・国費総額約1,100億円

【開発スケジュール】

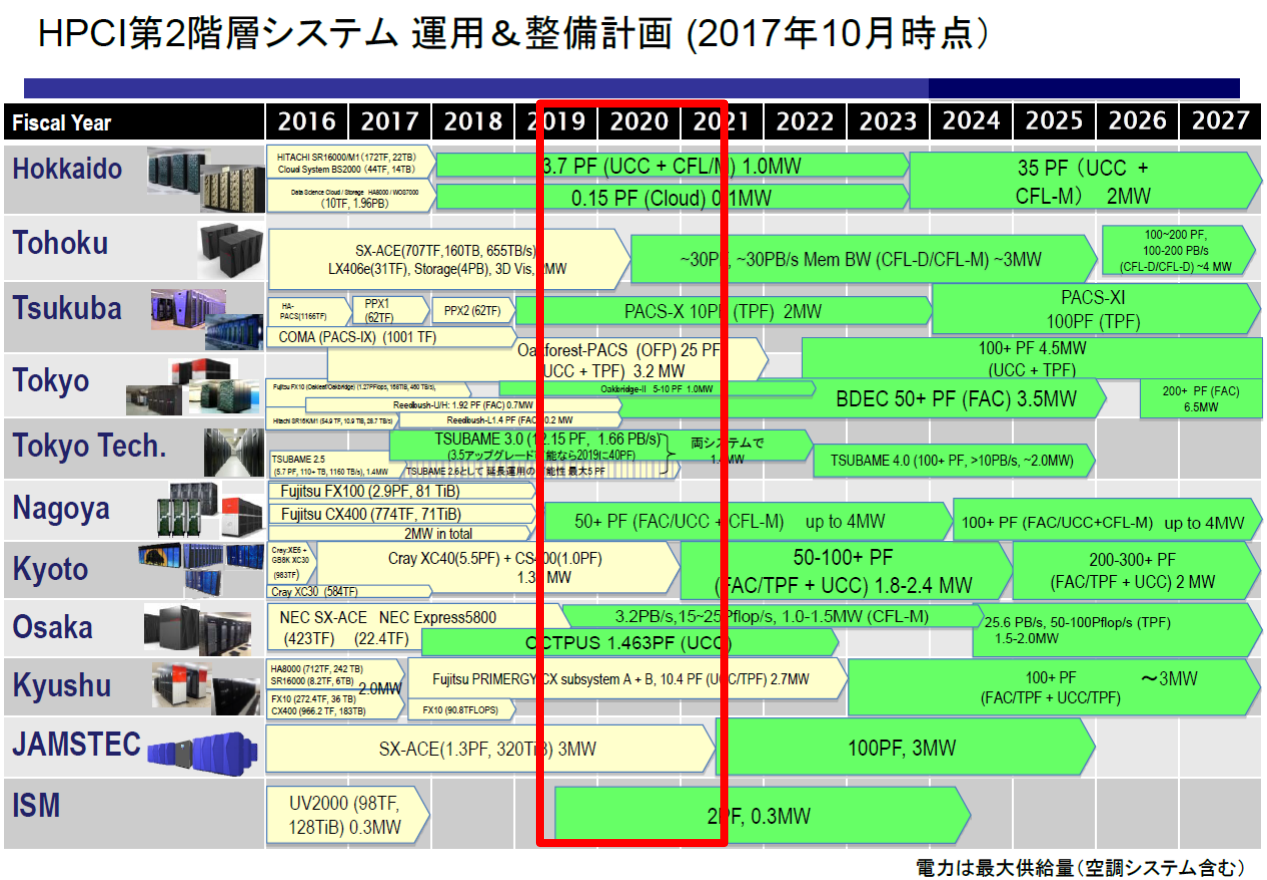
- ✓ K computer will stop at August 16th of 2019.
- ✓ The general use of the post-K will start at 2021 at the earliest.
- ✓ This period is supplemented by the use of supercomputers provided by each institution in HPCI. About 30% of computing resources are provided to each institution.
- ✓ We can not do calculations using nodes on a large scale during this period(at most 1,000 nodes).

Published specifications of post-K

	Functions & Architectures	Post-K	K computer
Processor	Base ISA + SIMD Extensions	ARMv8-A+SVE	SPARCv9+HPC-ACE
	SIMD width[bit]	512	128
	FP16(half precision) support	✔	-
	FMA: Floating-point multiply and add	✔	✔
	Math. Acceleration primitives	✔Enhanced	✔
	Inter-core barrier	✔	✔
	Sector cache	✔Enhanced	✔
	Hardware “prefetch” assist	✔Enhanced	✔
Interconnect	Tofu	✔Enhanced	✔

There is a high possibility that the installed memory is equal to or lower than K computer.

HPCI supercomputer situation



- ✓ The highest use hope is the name FX 100. The highest use hope is the name FX 100. At present, it is scheduled to suspend operation in FY 2018, but extension for one year is being studied.
- ✓ Since there are supercomputers besides these (supercomputer of ISSP etc…), there is still much computing resources.

K computer is working on 24h/365days

- K computer is active for public use on 24hours/365 days for general users. Almost all the top class machines of the top500 ranking are for closed users and are used for limited purposes.
- The several rankings show that K computer is almighty.
- Users may optimize their code for K computer easier than for the other ones(in the sense that it does not use a special architecture like GPU.).

Future direction of large-scale computing

From the above rankings

- From now on, both performance and energy-saving are important (Summit makes energy saving and computing performance highly compatible).
- The trend of many-core system with not so high frequency CPU is still going on. →post K project is proceeding in this direction.
- Or many supercomputers have GPU and realize energy saving.
- As GPU is prominent in the field of AI and machine learning, this architecture will continue to be adopted in the future.
- K computer is an almighty supercomputer for general purpose from several rankings. However, I do not know whether supercomputers made up of GPU are effective in all fields or not.
- Probably, many supercomputers designed for data analysis and AI will appear after now.

However,…

- Effective performances measured by LINPACK or HPCG benchmarks are not necessarily accordance with the performance of our code. Usually, we spend so much effort to get better performance on supercomputers than typical PC clusters.
  →The post-K machine will be constructed for getting better performance for scientific application than benchmark programs (Co-design). (However, it may be the national mission to get the first prize of top 500 for taxpayers.)
- The effort goes beyond knowledge and techniques of physicists, chemist, biologists,… We need those of computer scientists and numerical mathematicians and need to collaborate them.
- The post-K computer will be also a computer for general purpose. (cf. Anton (supercomputer for performing classical MD.))

Future direction1

- The tendency of architecture is multi cores or many cores (including SIMD,GPU, …).
- It is not easy to use all cores effectively.

(As personal opinion, I think that I do not have to worry about using all the cores and it is enough to get the best efficiency to obtain the calculation result.)

- Is it reasonable and proper to devote many researchers' effort to develop parallel efficiency? (very serious problem!!!)
- For DFT calculations, it may be difficult to speed-up the software by using GPU without breakthrough.
- We have to perform feasibility studies continuously to follow a leading-edge architecture.

Future direction2

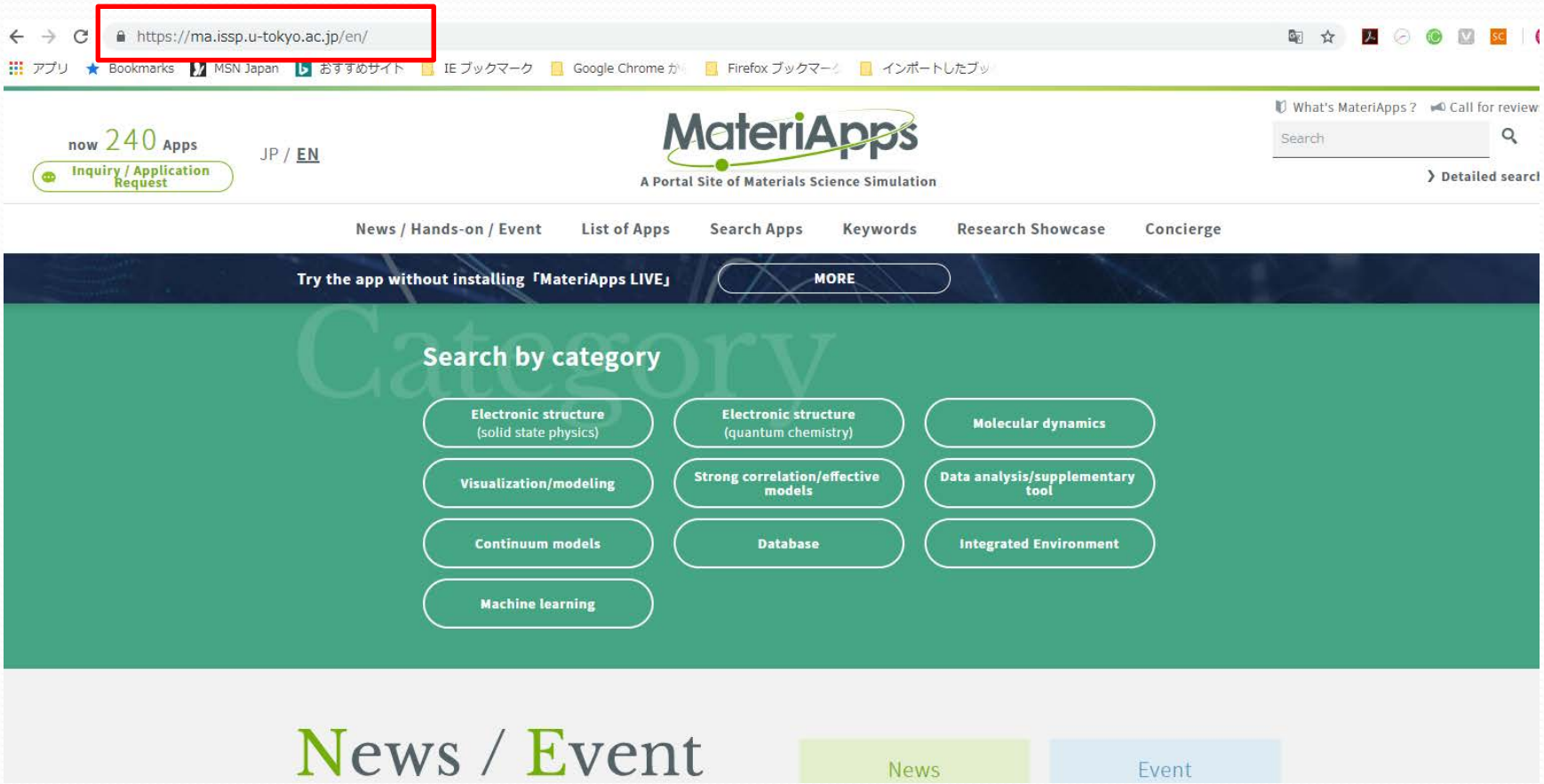
- USA is developing both architecture and the standard of programming language.
- It is too much hard-pressed for physicists and chemists to cover computer science or numerical analysis.

  →it is needed to establish a system to promote to work together with computational scientists and computer scientists or experts of numerical analysis.
- Still now, the guarantee of accuracy of calculation is inadequacy. There is no standard for summation of data from cores (threads) on MPI or OpenMP,… We must treat it carefully, otherwise our calculation may become meaningless.

Future direction3

- Today's amassed skills through K computer can be used during the next 10 years. (In Japan, the post-K project, and the next project)
- However, the development of supercomputer must be stop in near future.(Prof. Hiraki said it was 2029. The reason is the necessity of huge electric power, failure of Moors' law, etc…)
- We should consider a new idea to get high performance of our own calculations for both software and hardware as soon as possible. We may undergo a paradigm shift before 2029.

MateriApps



This portal site provides the information of applications that are used in materials science. You can find your desired application. “MateriApps LIVE” gives you an environment to perform open-source software codes without any installation process.

Finally,

- We should advance the development of human resources to be able to use supercomputers and/or HPC machines effectively.
- The relation between the ranking of top500 and the software code we use is not necessarily clear. We need to have the knowledge of the architecture of the supercomputers and/or the HPC machines if we'd like to maximize the performance of our software codes.
- We should cultivate people who can develop the software code.
- We should cultivate people who can effectively use scientific software codes to progress the researches. (CMD-WS, distance learning etc…)