Experiences on K computer from a topic focused on the large-scale eigenvalue solver project

Toshiyuki IMAMURA, RIKEN AICS,
Joint work with
Tetsuya Sakurai, Yasunori Futamura, Akira Imakura
University of Tsukuba
Takeshi Fukaya, Yusuke Hirota
RIKEN Advanced Institute for Computational Science
and Susumu Yamada, Masahiko Machida
Japan Atomic Energy Agency

**** This work is supported by CREST JST, collaboration with ESSEX under the joint initiative of DFG-JST-ANR (2016-2018).**

24th Workshop for Sustained Simulation Performance,
Stuttgart HLRS Aquarium, 5-6Dec, 2016

K computer

Agenda

1. Quick Overview of Project

- Past and Present of EigenExa
- Diagonalization of a 1million x 1million matrix on K computer

2. The latest updates

3. Future direction

4. Summary

Key topic is
How to remove three walls;
i) Memory bandwidth, ii) Network Latency, iii) Parallelism.

H4ES (2011-2016)

1. Eigenvalue problem is of significance in many scientific and engineering fields

$$Ax = \lambda Bx$$

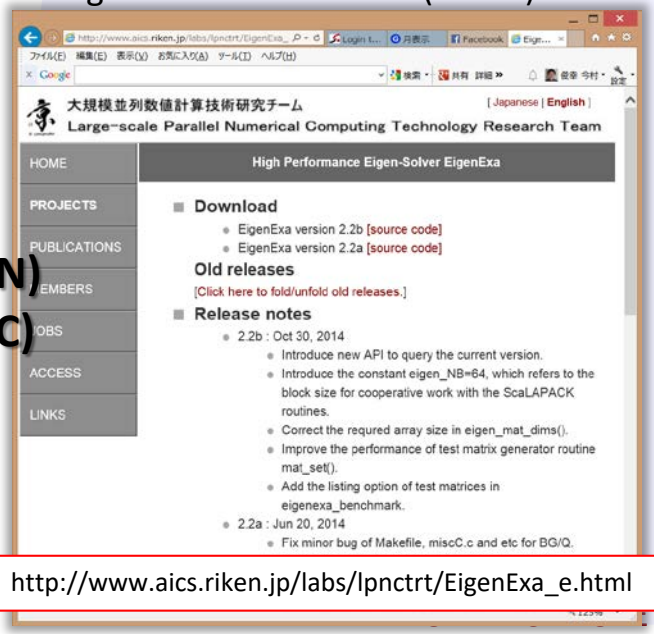
2. In practical simulation, Sparse and dense solver must be tightly cooperating
3. Currently, collaboration with ESSEX under the joint initiative of DFG-JST-ANR (2016-2018).

Prof. Sakurai Team 'H4ES'

Application:
Hoshi(Tottori)
Kuramashi(Tsukuba)

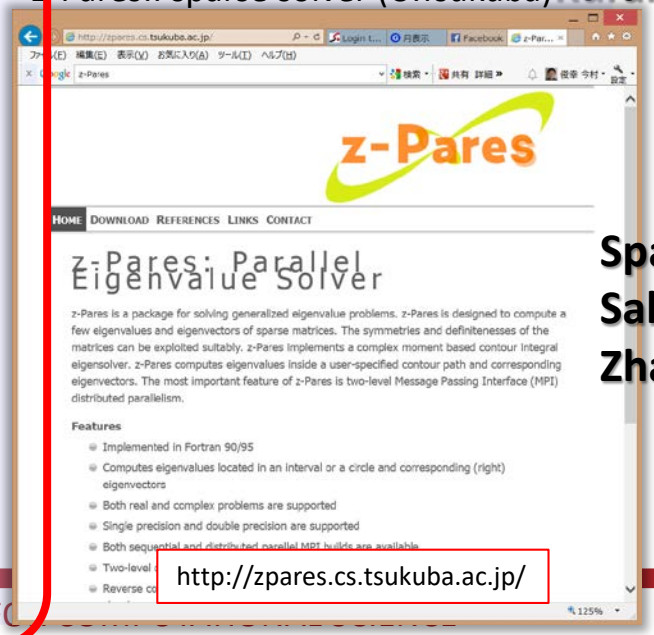
Dense:
Imamura(RIKEN)
Yamamoto(UEC)

EigenExa:: dense solver (RIKEN)



http://www.aics.riken.jp/labs/lpnctrt/EigenExa_e.html

z-Pares:: sparse solver (U.Tsukuba)

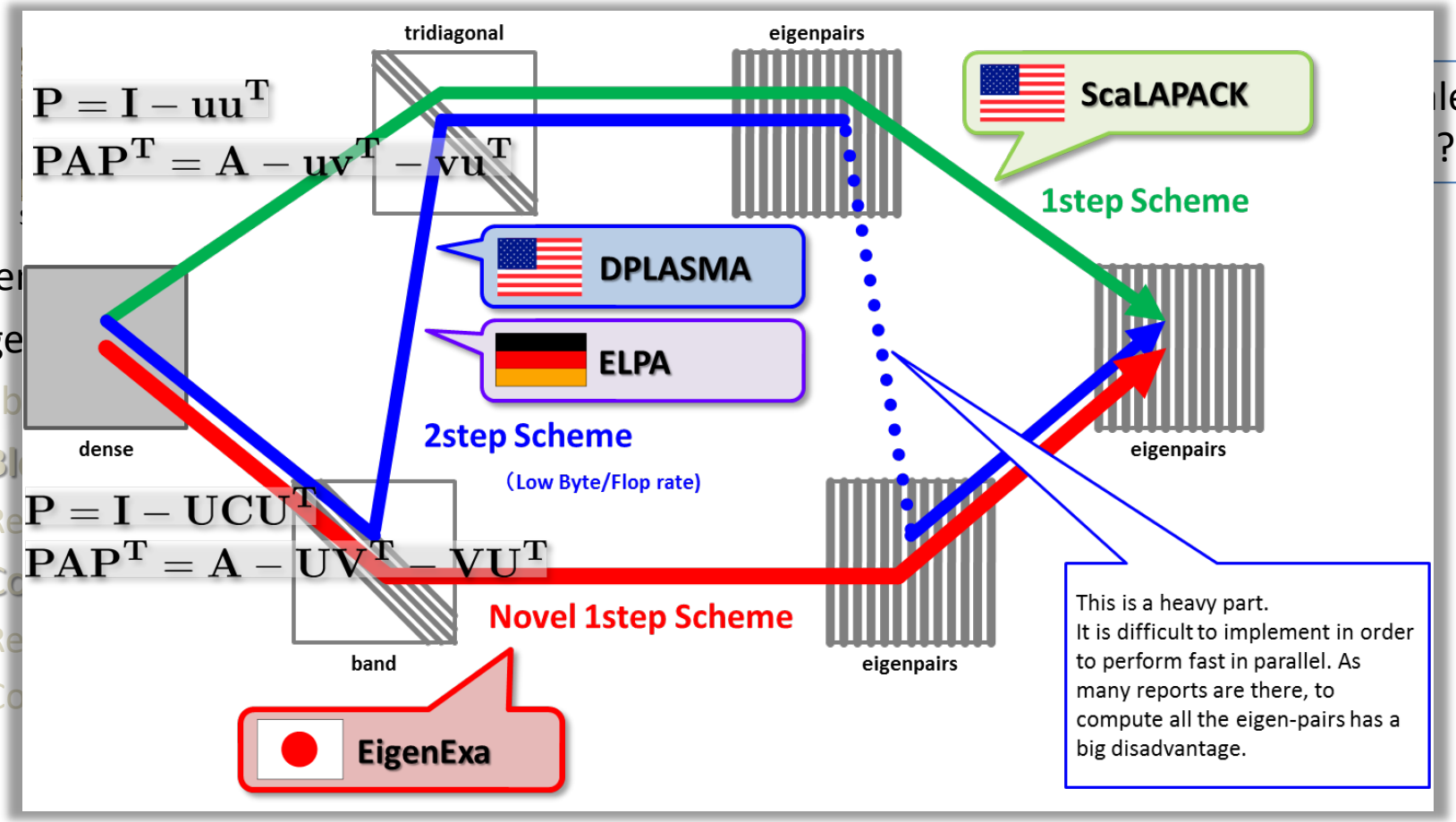


http://zpares.cs.tsukuba.ac.jp/

Sparse & LS:
Sakurai(Tsukuba)
Zhang(Nagoya)

EigenExa Project

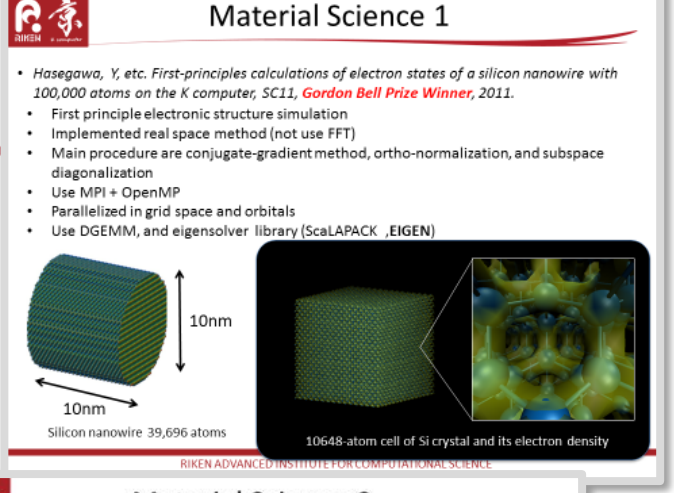
- Project itself is old…
  - Earth Simulator version was published in SC06. and the speakers continue to update it approximately 10 years. Partly funded by another CREST organized by Prof. Yagawa.

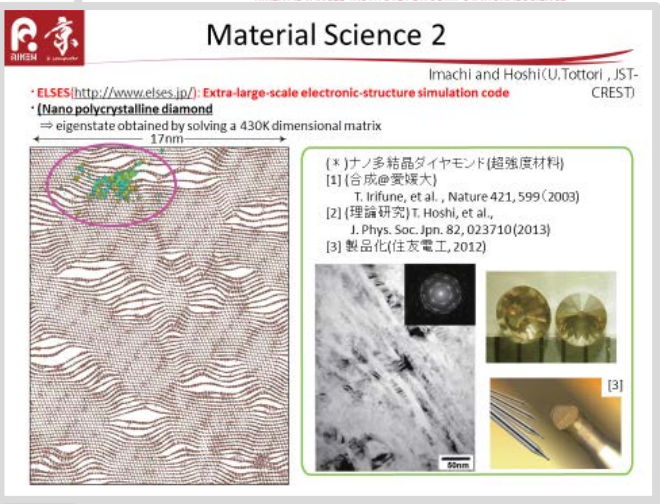


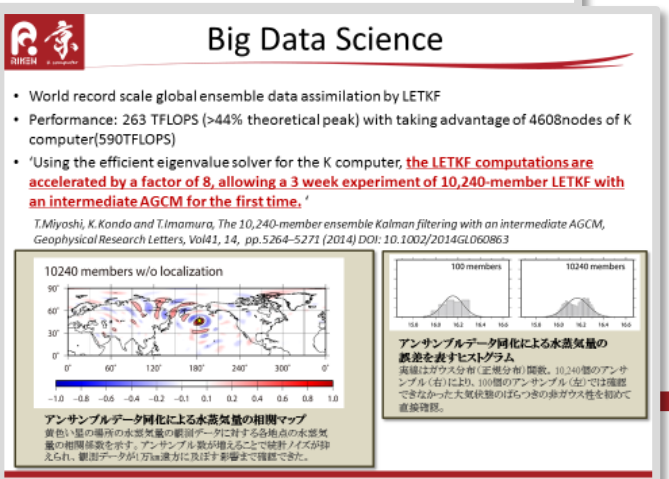
```
      k_1 = i - i_base
      k_2 = m0

      L = i-1
      n  = eigen_translate_g2l(L, x_nnod,x_inod)

!$OMP MASTER
      call eigen_vector_zeropad_x( v_x(1), L )
!$OMP END MASTER

!$OMP BARRIER
      prod_uv = u_t(1)
      if ( k_2 <= k_1 ) then
        if ( beta /= ZERO ) then
          alpha = prod_uv/(2*beta)
!$OMP DO
          do j_1=1,n
            v_x(j_1) = (v_x(j_1)-alpha*u_x(j_1))/beta
          end do          ! j_1
!$OMP ENDDO
        end if
      else
!
!   v=v-(UV+VU)u
!
        l_4 = MOD(k_2-k_1, 3)+k_1+1
        LX  = 64            ! L1_LSIZE*L1_WAY/16

        LL = (n-1)/y_nnod+1
        LL = ((LL-1)/2+1)*2

        alpha = ddot( k_2-k_1, u_t(2), 2, u_t(3), 2 )
        prod_uv = prod_uv - 2*alpha

        if ( n > VTOL ) then
          jj_2 = 1+LL*(y_inod-1)
          jj_3 = MIN(n, LL*y_inod)
        else
          jj_2 = 1
          jj_3 = n
        endif

!$OMP DO
        do jj_1=jj_2,jj_3,LX
          j_2 = jj_1; j_3 = MIN(jj_1+LX-1, jj_3)
          if(l_4-1==k_1+1)then
            l_1 = k_1+1     ! 0
            j = l_1-k_1

            u0 = u_t(2*(j+0)-1+1)
            v0 = u_t(2*(j+0)-0+1)
            do j_1=j_2,j_3
              w0 = v_x(j_1)
              ux0 = ux(j_1,l_1+0)
              vx0 = vx(j_1,l_1+0)
              w0 = w0-ux0*u0-vx0*v0
              v_x(j_1) = w0
            end do          ! j_1
          end if
          if(l_4-2==k_1+1)then
            l_1 = k_1+1     ! 1

            j = l_1-k_1

            u0 = u_t(2*(j+0)-1+1)
            v0 = u_t(2*(j+0)-0+1)
            u1 = u_t(2*(j+1)-1+1)
            v1 = u_t(2*(j+1)-0+1)
            do j_1=j_2,j_3
              w0 = v_x(j_1)
              ux0 = ux(j_1,l_1+0)
              vx0 = vx(j_1,l_1+0)
              w0 = w0 -ux0*u0 -vx0*v0
              ux1 = ux(j_1,l_1+1)
              vx1 = vx(j_1,l_1+1)
              w0 = w0 -ux1*u1 -vx1*v1
              v_x(j_1) = w0
            end do          ! j_1
          end if
          do l_1=l_4,k_2,3    ! 2

            j = l_1-k_1

            u0 = u_t(2*(j+0)-1+1)
            v0 = u_t(2*(j+0)-0+1)
            u1 = u_t(2*(j+1)-1+1)
            v1 = u_t(2*(j+1)-0+1)
            u2 = u_t(2*(j+2)-1+1)
            v2 = u_t(2*(j+2)-0+1)
            do j_1=j_2,j_3
              w0 = v_x(j_1)
              ux0 = ux(j_1,l_1+0)
              vx0 = vx(j_1,l_1+0)
              w0 = w0 -ux0*u0 -vx0*v0
              ux1 = ux(j_1,l_1+1)
              vx1 = vx(j_1,l_1+1)
              w0 = w0  -ux1*u1 -vx1*v1
              ux2 = ux(j_1,l_1+2)
              vx2 = vx(j_1,l_1+2)
              w0 = w0  -ux2*u2 -vx2*v2
              v_x(j_1) = w0
            end do          ! j_1
          end do
        end do          ! jj_1
!$OMP ENDDO

        if ( beta /= ZERO ) then
          alpha = prod_uv/(2*beta)
!$OMP DO
          do j_1=jj_2,jj_3
            v_x(j_1) = (v_x(j_1)-alpha*u_x(j_1))/beta
          end do          ! j_1
!$OMP ENDDO
        end if
!$OMP MASTER
        if ( n > VTOL ) then
          call allgather_dbl(v_x(jj_2), v_t, LL, 1, y_COMM_WORLD)
          j_3 = eigen_loop_end(L, x_nnod, x_inod)
          v_x(1:j_3) = v_t(1:j_3)
        end if
!$OMP END MASTER
      end if
!$OMP BARRIER

!$OMP MASTER
      v_n    = (v_n-alpha*u_n)/beta
      x_owner_nod = eigen_owner_node   (L, x_nnod,x_inod)
      x_pos       = eigen_translate_g2l(L, x_nnod,x_inod)
      if ( x_inod == x_owner_nod ) then
        v_x(x_pos) = v_n
      end if
      if ( kk == 0 ) then
        call datacast_dbl( v_y(1), v_x(1), u_t(1), v_t(1), x_pos, 2 )
      end if
      call eigen_vector_zeropad_x( v_x(1), L )
      if ( kk == 0 ) then
        call eigen_vector_zeropad_y( v_y(1), L )
      end if
!$OMP END MASTER
```

Simulation codes

- 8 Applications
  - **Platypus QM/MM:** gives the precise analysis for a biological polymer such as a kinase reaction mechanism by introducing the electron state effect to the molecular mechanics (MM) approach
  - **RSDFT:** is an ab-initio program with the real-space difference method and a pseudo-potential method
  - **PHASE:** is a Computer Software for Band Calculations based on First-principles Pseudo-potential Method
  - **ELSES:** is large-scale atomistic simulation with quantum mechanical freedom of electrons manipulating a large Hamiltonian matrix.
  - **NTChem:** is a high-performance software package for the molecular electronic structure calculation for general purpose on the K computer
  - **Rokko:** Integrated Interface for libraries of eigenvalue decomposition
  - **LETKF:** data assimilation for atmospheric and oceanic systems
  - **POD:** proper orthogonal decomposition (POD) to compress data for example video data



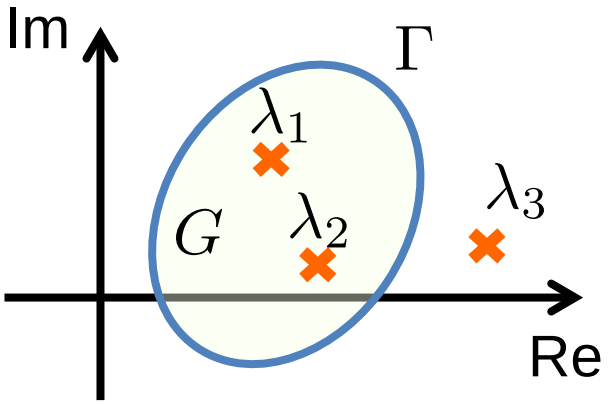




Sakurai-Sugiura eigenvalue solver

- Contour integral for a rational function

$$\frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} \sum_{i=1}^{n} \frac{\nu_i}{z-\lambda_i} \mathrm{d}z = \sum_{\lambda_i \in G} \nu_i$$



- Spectral decomposition of $(zB-A)^{-1}B$:

$$(zB-A)^{-1}B = \sum_{i=1}^{n} \frac{P_i}{z-\lambda_i}$$

λ_i : eigenvalue, P_i : spectral projection with respect to λ_i
(for simplicity, we consider the case that λ_i is simple)

Localization of spectral decomposition using contour integral

$$P_{\Gamma} = \frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} (zB-A)^{-1}B \mathrm{d}z = \sum_{\lambda_i \in G} P_i$$

z-Pares

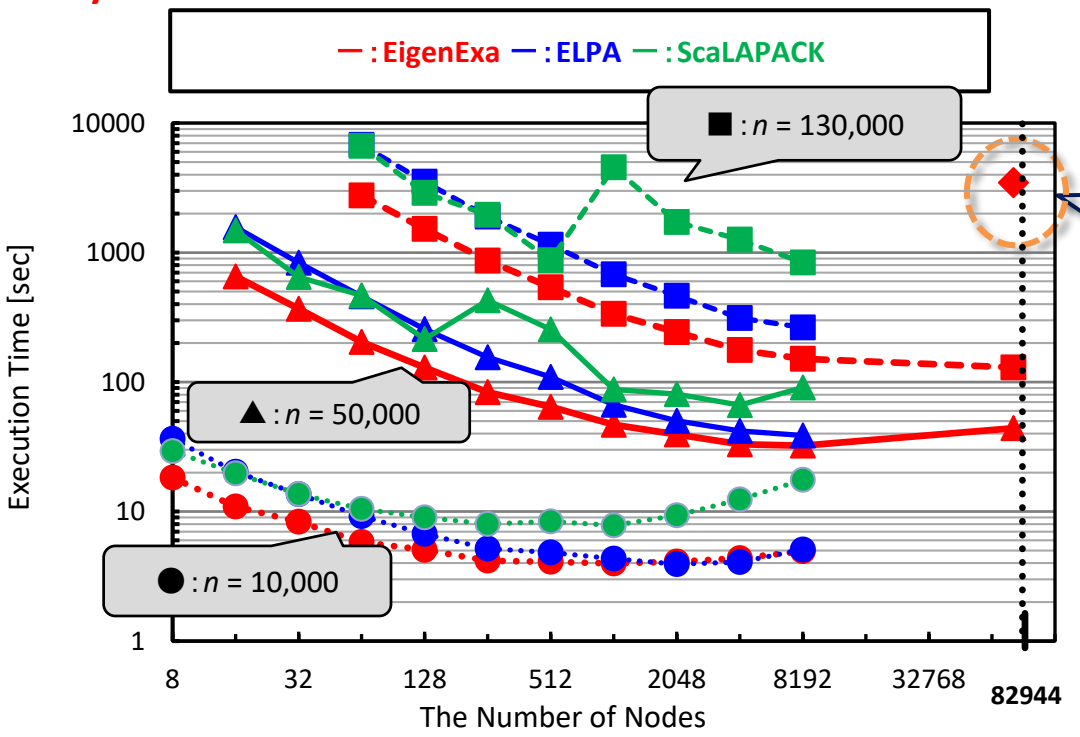
- Implemented in Fortran 95 and MPI
  - C interface will be provided
- Provides subroutines for
  - A,B real symm, B positive definite
  - A,B Hermitian, B positive definite
  - A,B real unsymmetric
  - A,B complex non-Hermitian
- Provides efficient implementation for standard problem
- Dependencies
  - BLAS/LAPACK
  - MUMPS* （Optional）

$$A\boldsymbol{x} = \lambda B\boldsymbol{x}$$

* MPI distributed parallel sparse direct linear solver

World Largest Dense Eigenvalue Computation

- We have successfully done a world largest-scale dense eigenvalue benchmark (one million dimension) by EigenExa taking advantage of the overall nodes (82,944 processors) of K computer in 3,464 seconds. Our EigenExa achieves 1.7 PFLOPS (16% of the K computer's peak performance).
- **Feasibility and Reliability for algorithm and library are confirmed, especially assumed on a post-K system.**



◆ : n = 1,000,000

EigenExa solves a world largest-scale problem.
(1.7 PFLOPS, 16% of K computer's theoretical peak performance)

$$\max_i \|Av_i - \lambda_i v_i\|_2 / \|A\|_F = 3.1 \times 10^{-13}$$
$$\|V^\top V - I\|_F = 2.1 \times 10^{-10}$$

- ✓ n is the dimension of problems.
- ✓ 1 MPI process * 8 threads per node.
- ✓ Test matrices are randomly generated.

Specification of K computer
- **Peak performance: 10.6 PFLOPS**
- **Num. of Nodes: 82,944**
- **Performance/node: 128 GFLOPS (One octa-core SPARC 64 VIIIfx)**
- **Network: Tofu interconnect (6D mesh-torus)**

Related performance report is
T.Fukaya, TI. "Performance evaluation of the EigenExa eigensolver on Oakleaf-FX: tridiagonalization versus pentadiagonalization", PDSEC2015

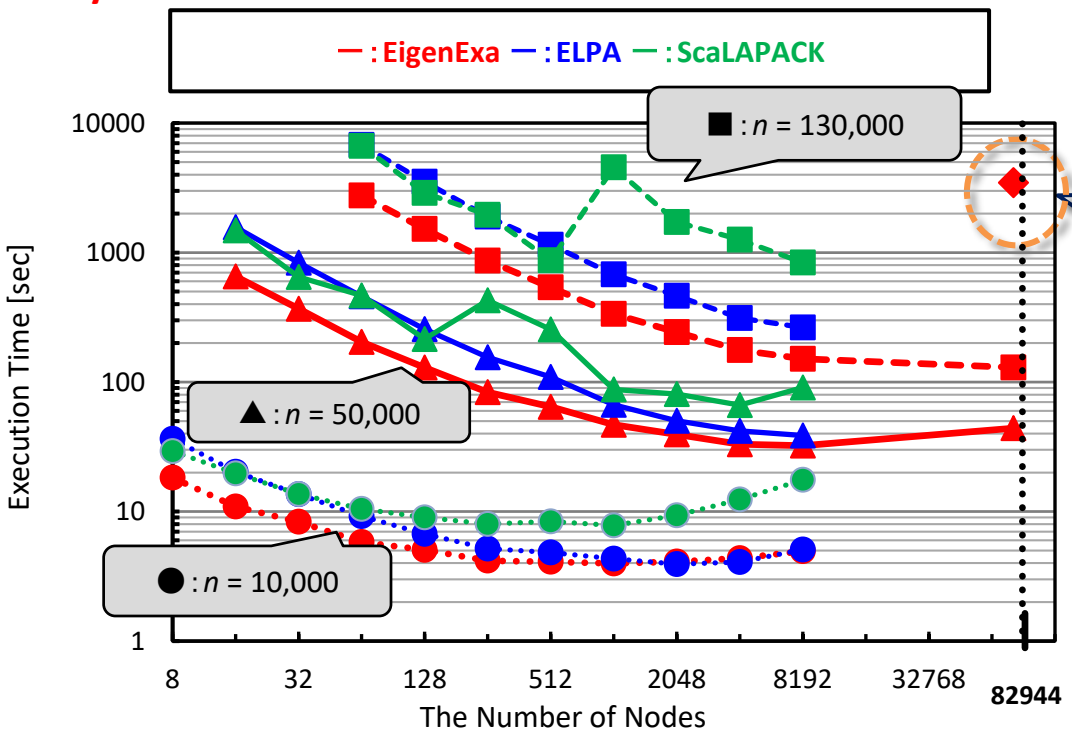
Agenda

1. Quick Overview of Project
   - Past and Present of EigenExa
   - Diagonalization of a 1million x 1million matrix on K computer
2. **The latest updates**
3. Future direction
4. Summary

Key topic is
How to remove three walls;
i) Memory bandwidth, ii) Network Latency, iii) Parallelism.

World Largest Dense Eigenvalue Computation

- We have successfully done a world largest-scale dense eigenvalue benchmark (one million dimension) by EigenExa taking advantage of the overall nodes (82,944 processors) of K computer in 3,464 seconds. Our EigenExa achieves 1.7 PFLOPS (16% of the K computer's peak performance).
- **Feasibility and Reliability for algorithm and library are confirmed, especially assumed on a post-K system.**



◆ : n = 1,000,000

EigenExa solves a world largest-scale problem.
(1.7 PFLOPS, 16% of K computer's theoretical peak performance)

$$\max_i \|A\boldsymbol{v}_i - \lambda_i \boldsymbol{v}_i\|_2 / \|A\|_F = 3.1 \times 10^{-13}$$
$$\|V^\top V - I\|_F = 2.1 \times 10^{-10}$$

- ✓ n is the dimension of problems.
- ✓ 1 MPI process * 8 threads per node.
- ✓ Test matrices are randomly generated.

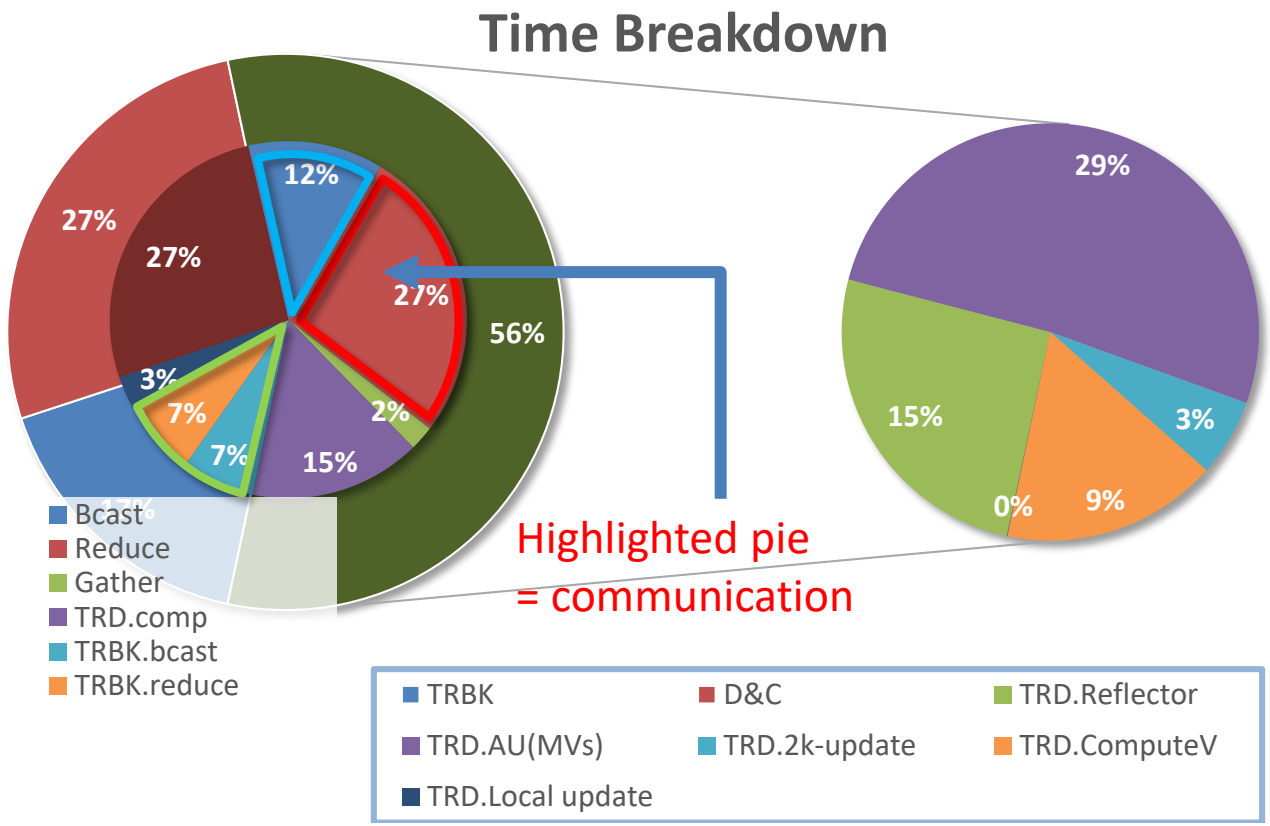
Specification of K computer
- **Peak performance: 10.6 PFLOPS**
- **Num. of Nodes: 82,944**
- **Performance/node: 128 GFLOPS (One octa-core SPARC 64 VIIIfx)**
- **Network: Tofu interconnect (6D mesh-torus)**

Related performance report is
T.Fukaya, TI. "Performance evaluation of the EigenExa eigensolver on Oakleaf-FX: tridiagonalization versus pentadiagonalization", PDSEC2015

What we got from the ultra-scale experiments

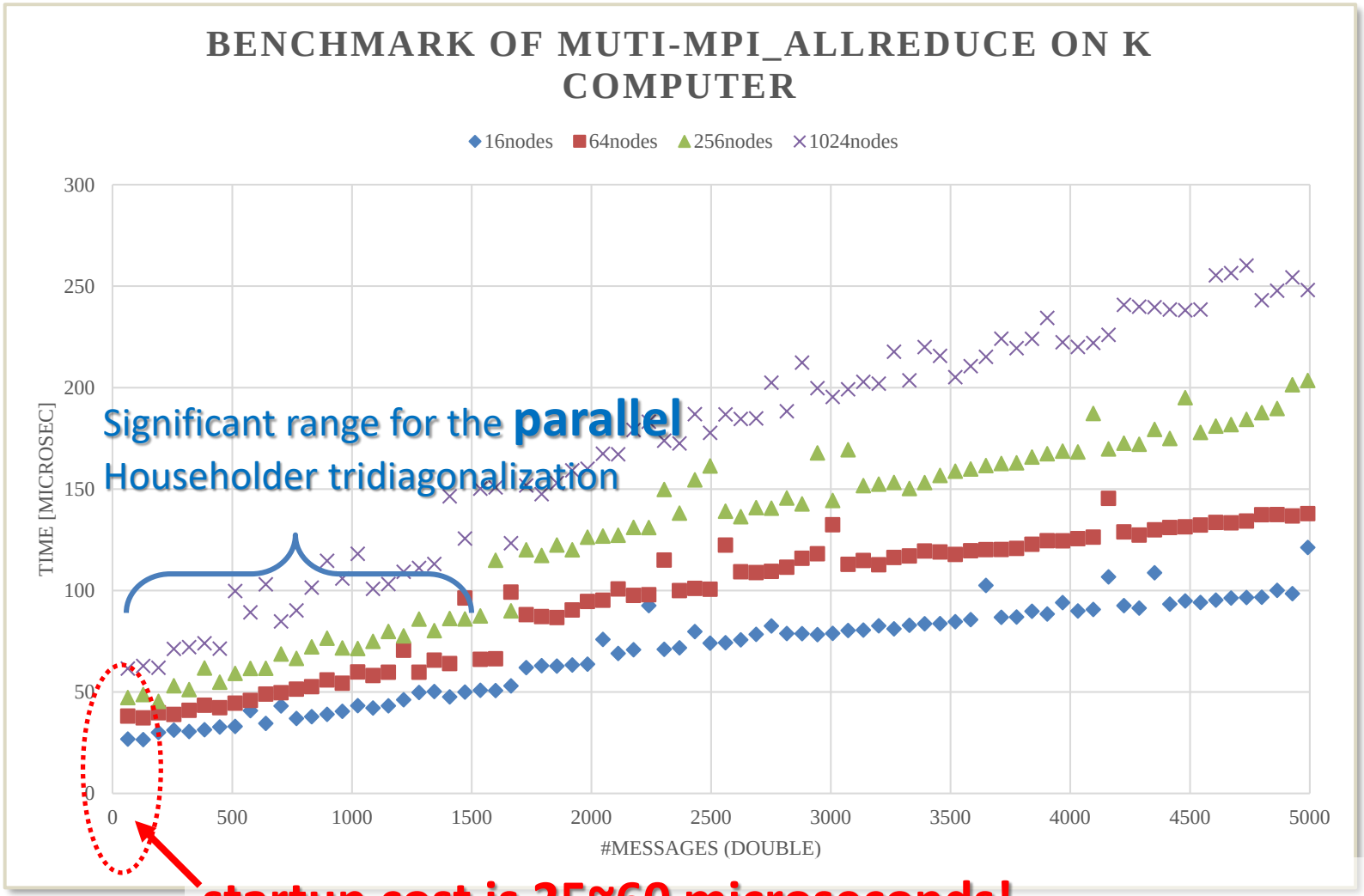
World Largest Dense Eigenvalue Computation

```
NUM.OF.PROCESS= 82944 ( 288 288 )
NUM.OF.THREADS= 8
calc (u,beta)     503.0970594882965
mat-vec (Au)      1007.285000801086 661845.1244051798
2update (A-uv-vu) 117.4089198112488
5678160.294281102
calc v            0.000000000000000
v=v-(UV+VU)u      328.3385872840881
UV post reduction 0.6406571865081787
COMM_STAT
   BCAST  ::  424.3022489547729
   REDUCE ::  928.1299135684967
   REDIST ::  0.000000000000000
   GATHER ::  78.28400993347168
TRD-BLK 1000000 1968.435860157013
677356.7583893638 GFLOPS
TRD-BLK-INFO 1000000   48
before PDSTEDC 0.1448299884796143
PDSTEDC 905.2210271358490
MY-REDIST1 1.544256925582886
MY-REDIST2 14.75343394279480
RERE1 4.861211776733398E-02
COMM_STAT
   BCAST  ::  4.860305786132812E-02
   REDUCE ::  2.155399322509766E-02
   REDIST ::  0.000000000000000
   GATHER ::  0.000000000000000
PDGEMM 532.6731402873993 5417097.565200453
GFLOPS
D&C 921.8044028282166 3130319.580211733 GFLOPS
TRBAK= 573.9026420116425
COMM= 533.7601048946381
   573.9026420116425 3484911.644577213 GFLOPS
   182.3303561210632 5484550.248648792 GFLOPS
   152.0370917320251 6577342.335399065 GFLOPS
   0.1022961139678955 7.379654884338379
COMM_STAT
   BCAST  ::  229.3666801452637
   REDUCE ::  234.4477448463440
   REDIST ::  0.000000000000000
   GATHER ::  0.000000000000000
TRBAKWY 573.9029450416565
TRDBAK 1000000 573.9216639995575 3484796.141101135
GFLOPS
Total 3464.162075996399 1795203.448396145 GFLOPS
Matrix dimension =  1000000
Internally required memory =  480502032  [Byte]
Elapsed time =  3464.187163788010  [sec]
```



Related performance report is
H.Imachi, T. Hoshi "Hybrid Numerical Solvers for Massively Parallel Eigenvalue Computations and Their Benchmark with Electronic Structure Calculations", Journal of Information Processing Vol. 24 (2016) No. 1 pp. 164-172

Allreduce is an expensive operation



CA for EigenExa

- **Communication Avoiding algorithm**
  - Blocking technique, increasing locality by data replication, and exchange the operation order.
  - Introducing an extended form of vector 'A'.
  - Computing Au and u^Tu, simultaneously.

TI, etc, "CAHTR: Communication-Avoiding Householder Tridiagonalization", ParCo15
TI, "Parallel dense eigenvalue solver and SVD solver for post-petascale computing systems", PMAA16

Naïve version of the 2-sided HH trans.

$$s = \mathrm{sign}(\|u\|, -(u, e))$$
$$u := u - se$$
$$v = Au$$
$$[C_U; C_V] = [U^T; V^T]u$$
$$v := v - (UC_V + VC_U)$$
$$f = (u, v)$$
$$v := v - afu$$

Single or several word allreduce

CA for EigenExa

- **Communication Avoiding algorithm**
  - Blocking technique, increasing locality by data replication, and exchange the operation order.
  - Introducing an extended form of vector 'A'.
  - Computing Au and u^Tu, simultaneously.

TI, etc, "CAHTR: Communication-Avoiding Householder Tridiagonalization", ParCo15

TI, "Parallel dense eigenvalue solver and SVD solver for post-petascale computing systems", PMAA16

Principles :

Distributive Law && Exchange order && Introducing correction terms && Combine couples of collective operations into one

naive

$$
\begin{aligned}
s &= \mathrm{sign}(\|u\|, -(u,e)) \\
u &:= u - se \\
v &= Au \\
[C_U; C_V] &= [U^T; V^T]u \\
v &:= v - (UC_V + VC_U) \\
f &= (u,v) \\
v &:= v - afu
\end{aligned}
$$

$$
\begin{bmatrix} v & d \\ s & t \end{bmatrix} = \begin{bmatrix} A & u \\ u^t & 0 \end{bmatrix} \begin{bmatrix} u & e \\ 0 & 0 \end{bmatrix}
$$

$$
\begin{aligned}
s &= \mathrm{sign}(\sqrt{s}, -t) \\
[u,v] &:= [u,v] - s[e,d] \\
[C_U; C_V; g] &= [U^T; V^T; v^T]u \\
v &:= v - (UC_V + VC_U) \\
f &= g - 2C_U^T C_V \\
v &:= v - afu
\end{aligned}
$$

optimal

$$
\left[\begin{array}{c|c} v & d \\ s & t \\ C_U & \gamma_U \\ C_V & \gamma_V \\ \hline g & v_1 \\ v_1 & a_{11} \end{array}\right] = \left[\begin{array}{c|ccc} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ \hline u^t & 0 & 0 & 0 \\ e^t & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c|ccc} A & u & U & V \\ \hline u^t & 0 & 0 & 0 \\ U^t & 0 & 0 & 0 \\ V^t & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c|c} u & e \\ \hline 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{array}\right]
$$

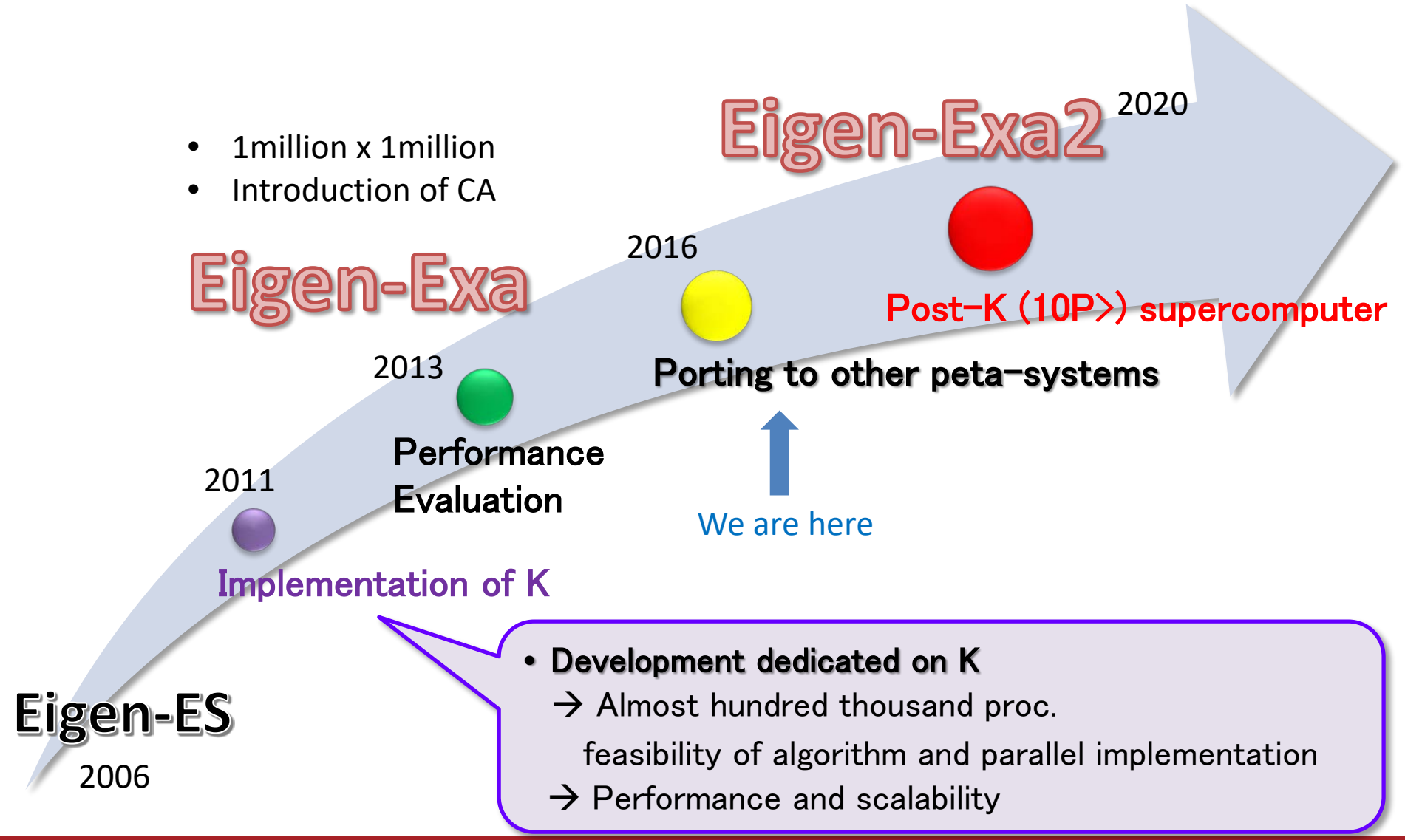
$$
\begin{aligned}
s &= \mathrm{sign}(\sqrt{s}, -t) \\
[u,v] &:= [u,v] - s[e,d] \\
[C_U; C_V] &= [C_U; C_V] - s[\gamma_U; \gamma_V] \\
v &:= v - (UC_V + VC_U) \\
f &= g - 2C_U^T C_V - s(2v_1 - sa_{11}) \\
v &:= v - afu
\end{aligned}
$$

Agenda

1. Quick Overview of Project
   - Past and Present of EigenExa
   - Diagonalization of a 1million x 1million matrix on K computer
2. The latest updates
3. Future direction
4. Summary

Key topic is
How to remove three walls;
i) Memory bandwidth, ii) Network Latency, iii) Parallelism.

Current status

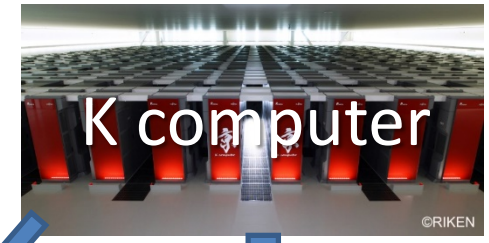


future of the EigenExa Project

- Porting the EigenExa library from K to other systems.

Sorry photo is ES2

T2K
PC cluster



Exa-scale
System??

Oakforest-PACS?

SX-ACE @ U. Osaka

BlueGene/Q @ Juelich

SX-ACE, SX-6, K

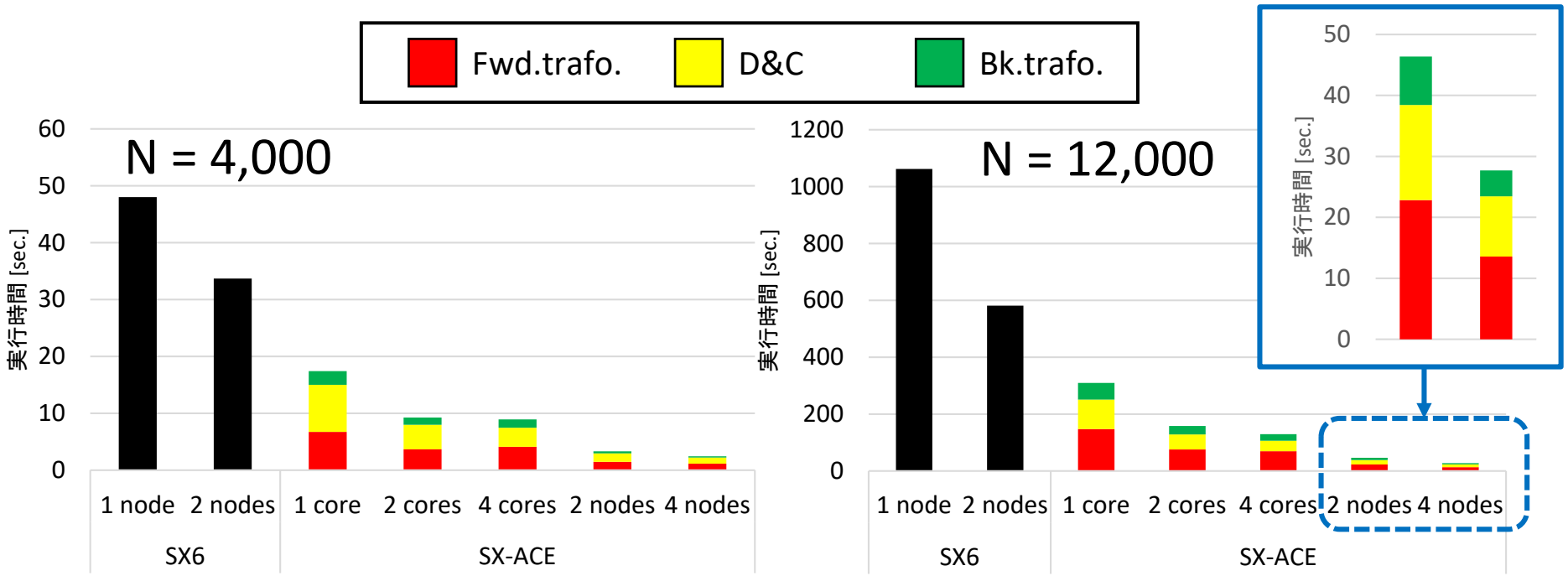
Hardware specification

	SX-ACE	SX-6 / ES1	京
FLOPS/node(PE or core)	64 GFLOPS/core	8 GFLOPS/PE	16 GFLOPS/core
Numbers of PEs	4 cores/node	8 PEs/node	8 cores/node
Memory bandwidth	256 GB/s	256 GB/s	64 GB/s
B/F	1	4	0.5
Network bandwidth（bi-direction）	8 GB/s	12.3 GB/s	10 GB/s

- SX-6/ES → SX-ACE
  - FLOPS/node increases 4 folds.
  - Relative bandwidth of memory and network degradate
- Next slide: Performance evaluation of eigensolver for a small problem
  - Randomly generated matrix
  - SX-ACE：EigenExa（eigen_sx），not optimized for SX-ACE
  - SX-6：the algorithm equivalent to eigen_s, 4PE/node was utilized.

Performance evaluation of eigensolver for a small problem

- SX-ACE：EigenExa(eigen_sx)，not optimized for SX-ACE
- SX-6：the algorithm equivalent to eigen_s, 4PE/node was utilized.



- SX-6/SX-ACE = x12.5 when 2nodes and N=12,000
  - Hardware improvement + newly developed algorithm
- Peak performance when 4 nodes and N=12,000 reaches 32% of theoretical peak. (Fwd.trafo. 17%, D&C 33%, Bk.trafo. 79%)
- Even vector system, the forward transform is dominant.

Next Steps

- **Hardware**
  - Near-future architecture, such as GPUs, MICs, FPGAs, accelerator boards, …
  - We always change and adapt the target architectures…

    for example, distributed parallel of multi-vector processors, on ES1

    the second generation was cluster of commodity processor and interconnect.

    present version is the third generation.

- **Target problems** (Complex, Tensor, Higher precision)
  - Standard type eigenvalue problems is currently supposed.
  - Generalized version is optional.
  - Not only building IEEE754 double but wider format QP (quadruple precision) is being developed by taking advantage of double-double or multiple-double data format.

- **Algorithm** (revival of old but solid idea to post-Moore era's processing elements)
  - Non-block algorithm but Titling when we focus on local computing
  - Hierarchical block strategy for a case of distributed computing

Target Architecture in near future

- We also have two branched projects from EigenExa on the K computer architecture
  - GPU:
    - Eigen-G = Experimental code on a single node + a single GPU environment
    - ASPEN.K2 = Automatic-tuning GPU BLAS kernels, especially, SYMV kernel
  - Intel Xeon Phi
    - Divide and conquer algorithm for GEVP focused on a pair of banded matrices
  - FPGAs ?

GPU

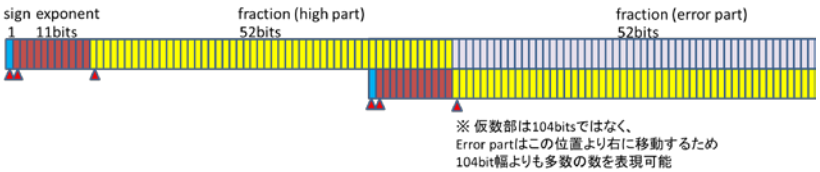
TI, etc, "Eigen-G: GPU-based eigenvalue solver for real-symmetric dense matrices", PPAM2013, LNCS8384
TI, etc. "High Performance SYMV Kernel on a Fermi-core GPU", VECPAR 2012, LNCS 7851,
TI, etc. "Automatic-tuning for CUDA-BLAS kernels by Multi-stage d-Spline Pruning Strategy", @^2HPSC 2014

MIC

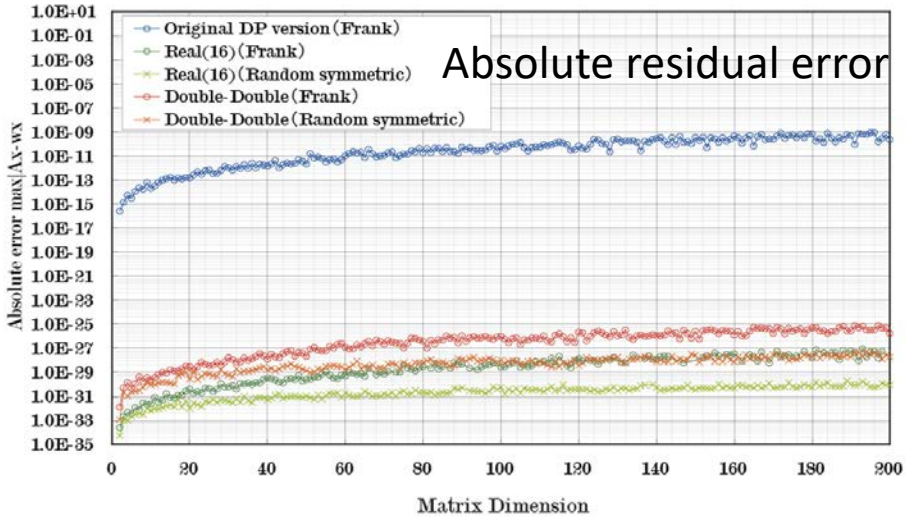
Y.Hirota, etc, "Divide-and-Conquer Method for Symmetric-Definite Generalized Eigenvalue Problems of Banded Matrices on Manycore Systems",SIAM LA15
Y.Hirota, etc. "Acceleration of Divide and Conquer Method for Generalized Eigenvalue Problems of Banded Matrices on Manycore Architectures, PMAA14.

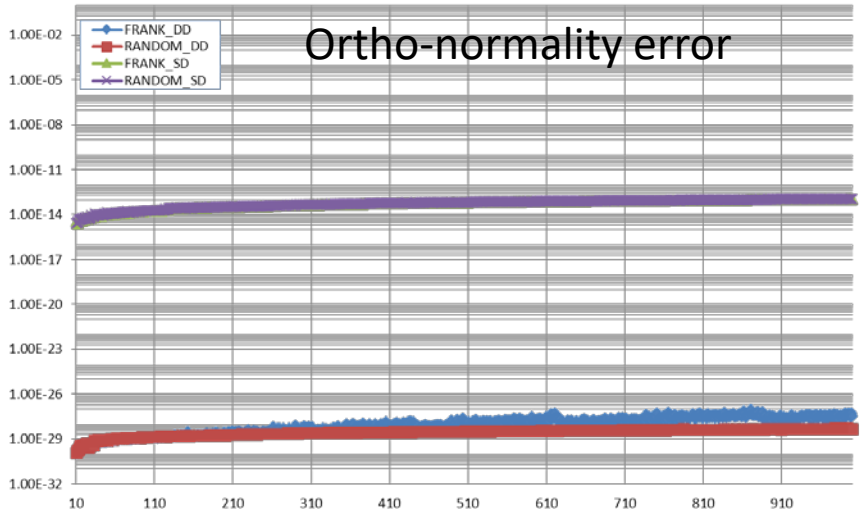
QP(Quadruple Precision)

- Emerging long-time and large-scale computation, **rounding error** on the IEEE754 'double' floating point format with O(10^15) operations will be a considerable issue. The **DD, double-double, format** (D.H.Bailey, DDFUN90, http://crd.ldl.gov/~dhbailey/mpdist) is one of promising technologies to ensure higher precision without the help of special hardware. The DD format consists of the 'high' and the 'error' parts, and their summation represents higher precision data.
- $$\mathbf{a_{dd}} := \mathbf{a_{hi}} + \mathbf{a_{err}}, \quad (|\mathbf{a_{hi}}| > |\mathbf{a_{err}}|)$$



- Addition and multiplication of two DD-format data are defined simply with approximately 20 double-precision floating operations. It is expected to help several issues on multicore platforms like accuracy and utilization problems. In this study, we are developing a double-double precision (**quadruple precision**) eigenvalue solver, '**QPEigenK**'. It performs on distributed memory parallel computers. In addition, **OpenMP and MPI parallel models** are supported.





Y.Hirota, etc. HPC in Asia Award Winning Poster: Performance of Quadruple Precision Eigenvalue Solver Libraries QPEigenK & QPEigenG on the K Computer

Summary of talk

- **EigenExa project (2011-2016)**
  - The first milestone : 1million order eigenvalue computation with full nodes of K computer.
  - Second milestone : optimization of communication
- **We struggled against 3 walls of bottleneck**
  - **Memory bandwidth → Block algorithm**
  - **Network Latency → Communication avoiding（CA) and communication hiding（CH)**
  - **Parallelism → on-going work towards new hardware**
- **Near-Future work**
  - Establish the CA technology for total performance of EigenExa
  - Quadruple Precision version
  - Vector computers, other platforms
  - GPU cluster, MIC cluster, etc.
- **Topics for Collaboration is broad,**
  - **New target architectures, FPGA ? or ?**
  - **New topics must be also concerned like Reproducibility and FT**
  - **New Collaboration with CS and Applications!**

THANKS!
ご清聴
ありがとう
ございました

The results of the present study were obtained in part using the K computer at RIKEN Advanced Institute for Computational Science.