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

\[ Ax = \lambda B x \]

2. In practical simulation, Sparse and dense solver must be tightly cooperating


Prof. Sakurai Team ‘H4ES’

Application:

Hoshi(Tottori)
Kuramashi(Tsukuba)

EigenExa:: dense solver (RIKEN)
Z-Pares:: sparse solver (U.Tsukuba)

http://zpares.cs.tsukuba.ac.jp/
http://www.aics.riken.jp/labs/lpnctrts/EigenExa_e.html

Dense:
Imamura(RIKEN)
Yamamoto(UEC)

Sparse & LS:
Sakurai(Tsukuba)
Zhang(Nagoya)
The EigenExa Project is described as having a long history, with the Earth Simulator version published in SC06 and updates continuing approximately every 10 years. The project is partly funded by another CREST organized by Prof. Yagawa. Currently, the project is focused on development for the K computer funded by JST CREST (2011/4 - 2016/3).

Two big trends in HPC Numerical Linear Algebra are highlighted:
1. Block algorithm
   - Reduce memory transfer to overcome the wall of memory
2. Communication Avoiding
   - Reduce times of data communication to overcome the wall of network latency

Consequently, the Block algorithm results in the CA T2K PC cluster, the ES K computer, and Exa-scale System?

The diagram illustrates different approaches and algorithms, including:
- ScaLAPACK
- DPLASMA
- ELPA
- Novel 1-step Scheme
- 2-step Scheme (Low Byte/Flop rate)
- Band and dense matrices

This is a heavy part. It is difficult to implement in order to perform fast in parallel. As many reports are there, to compute all the eigen-pairs has a big disadvantage.
\begin{itemize}
  \item \( k_1 = i - i_{\text{base}} \)
  \item \( k_2 = m_0 \)
  \item \( L = i - 1 \)
  \item \( n = \text{eigen_translate}_g2l(L, x_{\text{nnod}}, x_{\text{inod}}) \)
  \item \text{ISOMP MASTER}
  \begin{itemize}
    \item call \text{eigen_vector_zeropad}_x(v_x(1), L)
  \end{itemize}
  \item \text{ISOMP END MASTER}
  \item \text{ISOMP BARRIER}
  \begin{itemize}
    \item prod_uv = \( u_t(1) \)
    \item if \((k_2 < k_1)\) then
    \item \( \alpha = \text{prod}_uv(2*\alpha) \)
    \item \text{ISOMP DO}
    \begin{itemize}
      \item do_{j_1=1}^{n}
      \item \( v_x(j_1) = (v_x(j_1) - \alpha * u_x(j_1))/\beta \)
      \end{itemize}
    \item \text{ISOMP ENDDO}
    \item \text{end if}
  \end{itemize}
\end{itemize}
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
  \[ \frac{1}{2\pi i} \oint_{\Gamma} \sum_{i=1}^{n} \frac{\nu_i}{z - \lambda_i} \, dz = \sum_{\lambda_i \in G} \nu_i \]

- Spectral decomposition of \((zB - A)^{-1}B\):
  \[ (zB - A)^{-1}B = \sum_{i=1}^{n} \frac{P_i}{z - \lambda_i} \]

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

Localization of spectral decomposition using contour integral
\[ P_{\Gamma} = \frac{1}{2\pi i} \oint_{\Gamma} (zB - A)^{-1}B \, dz = \sum_{\lambda_i \in G} P_i \]
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)

\[ Ax = \lambda Bx \]

* 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)

\[ \max \| A v_i - \lambda_i v_i \|_2 / \| A \|_F = 3.1 \times 10^{-13} \]
\[ \| V^T V - I \|_F = 2.1 \times 10^{-10} \]

✓ $n$ 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.
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.

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

\[
\begin{align*}
\max \|Av_i - \lambda_iv_i\|_2 / \|A\|_F &= 3.1 \times 10^{-13} \\
\|V^TV - I\|_F &= 2.1 \times 10^{-10}
\end{align*}
\]

\( n \) 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
What we got from the ultra-scale experiments

World Largest Dense Eigenvalue Computation

Time Breakdown

Highlighted pie = communication

Related performance report is
Allreduce is an expensive operation.

BENCHMARK OF MUTI-MPI_ALLREDUCE ON K COMPUTER

Significant range for the parallel Householder tridiagonalization

startup cost is 25~60 microseconds!
~equivalent to pure time of allreduce with 1500 words
**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.

  \[ s = \text{sign}(\|u\|, -(u, e)) \]

  \[ u := u - se \]

  \[ v = Au \]

  \[ [C_U; C_V] = [U^T; V^T]u \]

  \[ v := v - (UC_V + VC_U) \]

  \[ f = (u, v) \]

  \[ v := v - afu \]

\[ TI, \text{ etc, “CAHTR: Communication-Avoiding Householder Tridiagonalization”, ParCo15} \]

\[ TI, \text{ “Parallel dense eigenvalue solver and SVD solver for post-petascale computing systems”, PMAA16} \]
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

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

optimal

\[
\begin{align*}
s &= \text{sign}(\sqrt{s}, -t) \\
[u, v] &= [u, v] - se \\
[C_U; C_V] &= [U^T; V^T]u \\
v &= v - (UC_V + VC_U) \\
f &= g - 2C_U^T C_V \\
v &= v - afu
\end{align*}
\]
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

- Post-K (10P>) supercomputer
- Porting to other peta-systems
- 1 million x 1 million
- Introduction of CA

Implementation of K

Performance Evaluation

We are here

Development dedicated on K

→ Almost hundred thousand proc.
→ Feasibility of algorithm and parallel implementation
→ Performance and scalability
future of the EigenExa Project

- Porting the EigenExa library from K to other systems.

[Diagram showing a flow from ES to T2K PC cluster to K computer to Exa-scale System? to Oakforest-PACS.]

- SX-ACE @ U. Osaka
- BlueGene/Q @ Juelich
SX-ACE, SX-6, K

Hardware specification

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

• **SX-6/ES → SX-ACE**
  – FLOPS/node increases 4 folds.
  – Relative bandwidth of memory and network degrade

• 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.

<table>
<thead>
<tr>
<th>N = 4,000</th>
<th>N = 12,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>SX-ACE:</td>
<td>SX-6:</td>
</tr>
</tbody>
</table>

- 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**

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

**MIC**

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.lbl.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.

\[ a_{dd} := a_{hi} + a_{err}, \quad (|a_{hi}| > |a_{err}|) \]

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: 1 million 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!

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

THANKS!
ご清聴ありがとうございました！