北海道大学

Performance Evaluation of the Mixed Precision GMRES(m) Method using FP64 and FP32

The international Conference on High Performance Computing in Asia-Pacific Region
(HPC Asia 2022)
January 12, 2022 | online

Presenter: Yingqi Zhao (Hokkaido University)

Collaborator： Takeshi Fukaya (Hokkaido University, JST PRESTO)

Takeshi Iwashita (Hokkaido University)

Overview

Background

- In scientific computations, exploiting the high ability of low precision computing is now getting crucial.
- In order to accelerate linear solvers, it is necessary to developing a new method that can exploit low precision computing and provide computed results with the same accuracy as by traditional methods using only FP64.

Main study

- Target problem : $Ax = b, A \in \mathbb{R}^{n\times n}, x, b \in \mathbb{R}^{n}$, where A is large, sparse and non symmetric.
- Mixed Precision GMRES(m) (called MP-GMRES(m)) using FP64 and FP32 was investigated through the numerical experiments and compared with GMRES(m) using only FP64(called FP64-GMRES(m)).
- Numerical behaviors (e.g. Number of the total iterations, achievable accuracy) were examined and evaluated.

Outline

- Introduction
  - Background
  - Main study
- Method
  - GMRES(m) method
  - Mixed precision GMRES(m) method
- Numerical experiments
  - Experimental settings
  - Number of iterations for attaining the convergence criterion
  - Achievable accuracy within the limit of iterations
  - Execution time for attaining convergence criterion
- Conclusion
- References

Introduction

Background

Traditional scientific computing

- FLOPS(Floating-point Operations Per Second) for Double-precision floating-point number (FP64) operation is metric for performance of CPU or computer system.
- FP64 has been used in scientific computations as standard.

Current situation

- Due to several reasons(e.g. power budget), it is getting difficult to improve FP64 FLOPS.
- Recent hardware trend: Some new applications accept low precision, for example, AI applications.

Advantages of using low precision computing

- Reduce the cost of data transfer.
- Increase the arithmetic performance, e.g. wider SIMD operations.

Exploit the advantages of low precision computing

➔ Key idea is **mixed precision computing**, in which both FP64 and low precision computing are combined.

Introduction

Target Problem

- $Ax = b, A \in \mathbb{R}^{n \times n}, x, b \in \mathbb{R}^{n}$

A : coefficient matrix
large, sparse, and non symmetric

Method

- Original algorithm: GMRES(m) method (Saad & Schultz, 1986; Saad, 2003).
- Target algorithm: mixed precision GMRES(m) method, a mixed precision variant of the GMRES(m) method using FP64 and FP32 (single-precision floating-point number).

Objective

- Investigate the numerical behavior of mixed precision GMRES(m) method using FP64 and FP32.
- Make a comparison between traditional GMRES(m) method using only FP64 and mixed precision GMRES(m) method.

Method : Iterative refinement

Iterative refinement

The solution x can be iteratively improved and reach the required accuracy by iterative refinement.

Let $\tilde{x}$ be the current approximate solution for $Ax = b$,

Step 1 : Compute the residual $r = b - A\tilde{x}$.

Step 2 : Solve the error equation $Ae = r$ and obtain an approximate solution $\tilde{e}$

Step 3 : Update the solution $\tilde{x} = \tilde{x} + \tilde{e}$

A mixed precision variant of the iterative refinement scheme

- Solution $\tilde{x}$ is stored in *high precision.*
- Step 1 and 3 use *high precision.*
- Step 2 uses *low precision.*

Method : GMRES(m) method

Algorithm 1 GMRES(m)

Input : x_0: initial guess, ϵ: convergence criterion, A: coefficient matrix, b: right-hand side vector, maximum number of iterations

1:	**repeat**
2:	$r_0 = b - Ax_0$
3:	$\beta = \|r_0\|_2$
4:	**If** $\beta/\|b\|_2 \leq \epsilon$ **then return** x_0
5:	$v_0 = r_0/\beta$
6:	Compute V_m and $\widetilde{H}_m$ by the m-step Arnoldi process with A and v_0.
7:	Compute y_m from β and $\widetilde{H}_m$.
8:	$x_m = x_0 + V_m y_m$
9:	$x_0 = x_m$
10:	**until** attain the maximum number of iterations

Output: x_0

Step 1: Solve $Ax = b$ by m-iteration GMRES with initial guess x_0 and obtain approximate solution x_m.

Step 2: Update the initial guess :$x_0 = x_m$.

Method : Mixed precision GMRES(m) method

In GMRES(m) method

Step 1 : Solve $Ax = b$ by m-iteration GMRES with initial guess x_0 and obtain approximate solution x_m.

Step 2 : Update the initial guess :$x_0 = x_m$.

Mathematically equivalent
(Imakura, A. et al., 2012)

Step 1 : Solve $Ae = r$ by m-iteration GMRES with initial guess $e_0 = 0$ and obtain approximate solution e_m, where $r = b - Ax_0$.

Step 2 : Update the initial guess : $x_0 = x_0 + e_m$.

GMRES(m) has the iterative refinement structure.

➔ Obtain a mixed precision variant of GMRES(m) method based on iterative refinement (mixed precision GMRES(m) method).

Method : Mixed precision GMRES(m) method

Algorithm 2 Mixed Precision GMRES(m)

Input : x_0: initial guess, ϵ: convergence criterion, A: coefficient matrix, b: right-hand side vector, maximum number of iterations

$Low()$: precision converts from *standard* to *low*

1:	$A^{(L)} = Low(A)$
2:	**repeat**
3:	$r_0 = b - Ax_0$,$\beta = \|r_0\|_2$
4:	**If** $\beta/\|b\|_2 \leq \epsilon$ **then return** x_0
5:	$v_0 = r_0/\beta$
6:	$\beta^{(L)} = Low\ (\beta)$, $v_0^{(L)} = Low\ (v_0)$
7:	Compute $V_m^{(L)}$ and $\widetilde{H}_m^{(L)}$ by the m-step Arnoldi process with $A^{(L)}$ and $v_0^{(L)}$.
8:	Compute $y_m^{(L)}$ from $\beta^{(L)}$ and $\widetilde{H}_m^{(L)}$.
9:	$z_m^{(L)} = V_m^{(L)} y_m^{(L)}$
10:	$z_m = Std(z_m^{(L)})$
11:	$x_m = x_0 + z_m$, $x_0 = x_m$
12:	**until** attain the maximum number of iterations

Using *low precision* arithmetic

$Std()$: precision converts from *low* to *standard*

Output: x_0

Numerical experiments: Experimental settings

Environment

- CPU : Xeon Gold 6148 (20 cores, 2.4GHz)×2
- Program : C language, OpenMP, CRS format for sparse matrix A
- Compiler : “Icc” (ver. 19.1.3.304), “-xCORE-AVX512 –qopenmp”(40 threads)

Test matrices

- Database : SuiteSparse Matrix Collection
- Matrix kind : Real matrices appear in scientific simulation
- Condition number $\kappa_2(A)(= \|A\|_2/\|A^{-1}\|_2)$: Range from $O(10^1)$ to $O(10^8)$

Settings

- $b = [1,1,\dots,1]^T$, and initial guess $x_0 = [0,0,\dots,0]^T$
- Convergence criterion : $\|b - Ax\|_2/\|b\|_2 \leq 10^{-10}$
- Iteration limit : (total # of inner iterations) $\leq$ n
- Candidates of m : 50, 100, 200, 300, 400, 500

Numerical experiments

—— Number of iterations for attaining the convergence criterion

FP64-GMRES(m) attained within one restart

$\kappa_2(A)$	Matrix	$m = 50$		$m = 100$		$m = 200$		$m = 300$		$m = 400$		$m = 500$	
		FP64	MP	FP64	MP	FP64	MP	FP64	MP	FP64	MP	FP64	MP
$O(10^1)$	cage10	26	59	26	110	26	211	26	311	26	411	26	511
$O(10^2)$	Zhao1	43	73	43	123	43	223	43	323	43	423	43	523
$O(10^3)$	epb1	1994	1881	1781	1786	1516	1517	1290	1302	1153	1153	1095	1272
$O(10^4)$	af23560	—	—	—	—	—	—	4799	4777	4305	4283	4442	4351
$O(10^5)$	memplus	5900	4521	3030	2943	1942	2044	1532	2404	1459	3265	1352	3108
$O(10^6)$	inlet	—	—	—	—	—	—	—	—	—	—	—	—
$O(10^7)$	garon2	—	—	—	—	—	—	—	—	—	—	—	—
$O(10^8)$	rajat15	—	—	—	—	—	—	—	—	—	—	—	—

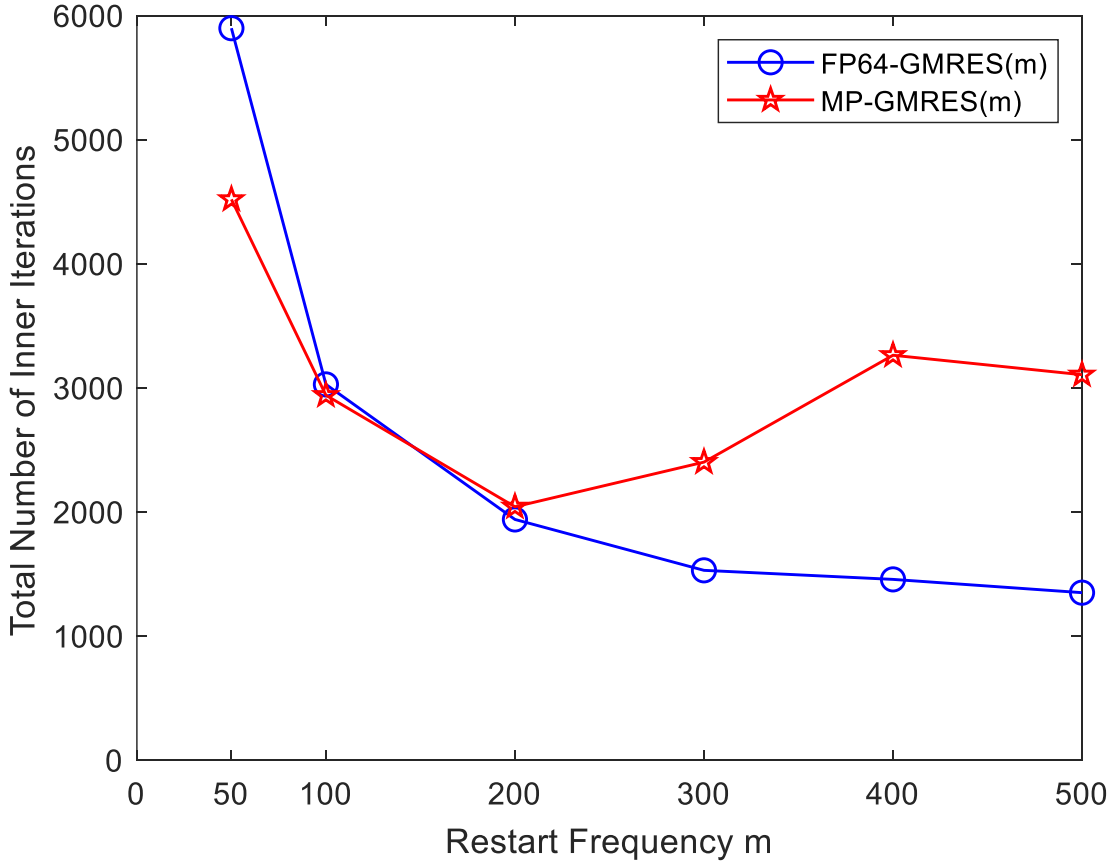
Results marked in red are the minimum # of iterations, "—" means the method didn't attain the convergence within limit of iterations

Analysis:

- When m is small, the number of iterations of the two methods are almost equal.
- When m becomes larger, the number of iterations of FP64-GMRES(m) basically decreases, but that of MP-GMRES(m) tends to increase.

Numerical experiments

—— Case study : matrix $memplus$



- The number of iterations:
  - FP64-GMRES(m): monotonically decrease
  - MP-GMRES(m): first decrease and then increase
- MP-GMRES(m) attained the convergence criterion with smaller iterations than FP64-GMRES(m) when $m = 50, 100.$

Fig.1 The change of the number of the total inner iterations in FP64-GMRES(m) and MP-GMRES(m) for $memplus$.

Numerical experiments

—— Achievable accuracy within the limit of iterations

$\kappa_2(A)$	Matrix	$m = 50$		$m = 100$		$m = 200$		$m = 300$		$m = 400$		$m = 500$	
		FP64	MP	FP64	MP	FP64	MP	FP64	MP	FP64	MP	FP64	MP
$O(10^1)$	cage10	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11
$O(10^2)$	Zhao1	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11
$O(10^3)$	epb1	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11
$O(10^4)$	af23560	-1	-1	-1	-1	-1	-1	-11	-11	-11	-11	-11	-11
$O(10^5)$	memplus	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11	-11
$O(10^6)$	inlet	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
$O(10^7)$	garon2	-1	-1	-1	-1	-2	-2	-3	-3	-3	-3	-3	-3
$O(10^8)$	rajat15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

$\log_{10}(\|b - Ax\|_2/\|b\|_2)$ when attained the iteration limit or the convergence criterion("-11" corresponds to this case)

Analysis:

- In most cases, the final accuracy of both FP64-GMRES(m) and MP-GMRES(m) is "-11", which means that both methods attained the convergence criterion $\frac{\|b-Ax\|_2}{\|b\|_2} \leq 10^{-10}$.
- Generally, if both two methods did not converge, there is almost no difference in the final accuracy.

Numerical experiments

—— Execution time for attaining convergence criterion

$\kappa_2(A)$	Matrix	Method	$m = 50$	$m = 100$	$m = 200$	$m = 300$	$m = 400$	$m = 500$
$O(10^1)$	cage10	FP64	1.03×10^{-2}	1.22×10^{-2}	1.10×10^{-2}	1.25×10^{-2}	9.49×10^{-3}	1.14×10^{-2}
		MP	1.43×10^{-2}	3.40×10^{-2}	9.85×10^{-2}	2.01×10^{-1}	4.09×10^{-1}	5.43×10^{-1}
$O(10^2)$	Zhao1	FP64	3.38×10^{-2}	3.77×10^{-2}	3.72×10^{-2}	3.79×10^{-2}	3.27×10^{-2}	3.49×10^{-2}
		MP	2.90×10^{-2}	7.75×10^{-2}	3.03×10^{-1}	7.62×10^{-1}	1.39×10^{0}	2.18×10^{0}
$O(10^3)$	epb1	FP64	6.09×10^{-1}	9.65×10^{-1}	1.59×10^{0}	2.19×10^{0}	2.81×10^{0}	3.50×10^{0}
		MP	3.24×10^{-1}	5.19×10^{-1}	8.09×10^{-1}	9.96×10^{-1}	1.24×10^{0}	1.50×10^{0}
$O(10^4)$	af23560	FP64	—	—	—	1.47×10^{1}	1.78×10^{1}	2.46×10^{1}
		MP	—	—	—	5.89×10^{0}	7.35×10^{0}	9.91×10^{0}
$O(10^5)$	memplus	FP64	2.32×10^{0}	2.12×10^{0}	2.69×10^{0}	3.49×10^{0}	4.36×10^{0}	5.16×10^{0}
		MP	1.30×10^{0}	1.23×10^{0}	1.30×10^{0}	2.18×10^{0}	3.37×10^{0}	4.14×10^{0}

Results marked in red are the shortest execution time, "—" means that the method did not attain the convergence criterion.

Analysis:

- When m is small, the execution time of both methods tends to be short.
- For many test matrices, MP-GMRES(m) has shorter execution time than MP-GMRES(m).

Numerical experiments
—— Case study : matrix $wang4$

Even in the case that larger m reduce the number of iterations in FP64-GMRES(m), MP-GMRES(m) is still faster.

$\kappa_2(A)$	Matrix	Item	Method	$m=50$	$m=100$	$m=200$	$m=300$	$m=400$	$m=500$
$O(10^5)$	wang4	Execution time	FP64	—	—	3.53×10^0	3.62×10^0	3.34×10^0	3.65×10^0
			MP	—	—	1.69×10^0	2.00×10^0	2.85×10^0	4.55×10^0
		# of iterations	FP64	—	—	1741	1101	738	688
			MP	—	—	1896	1423	1477	1792

For matrix $wang4$, about three times the iterations are required, MP-GMRES(m) is still faster.

Numerical experiments
—— Performance modeling

- **Our approach to performance modeling.**
  - In order to confirm the acceleration effect of MP-GMRES(m), speedup of MP-GMRES(m) over FP64-GMRES(m) is considered. Firstly, a model is established to analyze the speedup theoretically.
  - GMRES(m) is memory bounded computation and Arnoldi process is used. The m-step Arnoldi process mainly consists of m SpMVs and MGS for m vectors.
  - The estimation of the memory access cost in the m-step Arnoldi process in FP64 and FP32 is summarized.

Numerical experiments
—— Expected speedup

- **Model construction and expected speedup**

	FP64	MP
SpMVs	$(12N_{nz}+20n)m$	$(8N_{nz}+12n)m$
MGS	$8\cdot\frac{1}{2}m^2n$	$4\cdot\frac{1}{2}m^2n$
Total	$4m^2n+12mN_{nz}+20mn$	$2m^2n+8mN_{nz}+12mn$

Let γ be the ratio of the number of the iterations in MP-GMRES(m) over that in FP64-GMRES(m)

The speedup of MP-GMRES(m) over FP64-GMRES(m) is estimated as

$$Speedup=\frac{4m^2n+12mN_{nz}+20mn}{\gamma(2m^2n+8mN_{nz}+12mn\ )}=\frac{2m+6N_{nz}/n+10}{\gamma(m+4N_{nz}/n+6)}\simeq\frac{2c+6}{\gamma(c+4)}$$

where $c=m/(N_{nz}/n)$

Numerical experiments
—— Comparison between actual speedup and estimated speedup

For the cases in which the setting of m that provides the shortest execution time is not different between FP64-GMRES(m) and MP-GMRES(m), the actual speedup and the estimated speedup is compared.

$\kappa_2(A)$	Matrix	m	N_{nz}/n	c	γ	Actual speedup	Estimated speedup
$O(10^2)$	ns3Da	50	82	0.61	1.00	1.26	1.56
$O(10^3)$	epb1	50	6	8.33	0.94	1.88	1.94
$O(10^4)$	af23560	300	21	14.29	1.00	2.50	1.90
$O(10^5)$	memplus	100	7	14.29	1.02	1.73	1.86

- Since the estimated speedup is calculated based on the rough modeling, the estimated speedup seems to be close to the actual speedup.
- However, as SpMV is more complicated than MGS, it might bring difficulties to estimation which may cause a gap between actual and estimated speedup.

Conclusion

What we did

- MP-GMRES(m) using FP64 and FP32 was investigated through the numerical experiments and compared with FP64-GMRES(m) using only FP64.
- The numerical behaviors(e.g. number of the total iterations and achievable accuracy) were examined.
- The execution time of the methods using 40 threads was also evaluated on a standard CPU platform.

Main observations

- If a problem is solved by FP64-GMRES(m), basically, MP-GMRES(m) can solve it.
- When m is small, the number of iterations of the two methods are almost equal.
- As m grows, the number of the total iterations often decreases in FP64-GMRES(m) but tends to increase in MP-GMRES(m).
- On the standard CPU platform, it can be expected that MP-GMRES(m) outperforms FP64-GMRES(m) in terms of the execution time.

References

[1] Abdelfattah, A., Anzt, H., Boman, E. G., Carson, E., Cojean, T., Dongarra, J., Fox, A., Gates, M., Higham, N. J., Li, X. S., Loe, J., Luszczek, P., Pranesh, S., Rajamanickam, S., Ribizel, T., Smith, B. F., Swirydowicz, K., Thomas, S., Tomov, S., Tsai, Y. M. and Yang, U. M.: A survey of numerical linear algebra methods utilizing mixed-precision arithmetic, *The International Journal of High Performance Computing Applications*, Vol. 35, No. 4, pp. 344–369 (2021).

[2] Saad, Y. and Schultz, M. H.: GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM Journal on Scientic and Statistical Computing*, Vol. 7, No. 3, pp. 856-869 (1986).

[3] Saad, Y.: *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics (2003).

[4] Imakura, A., Sogabe, T. and Zhang, S.-L.: An Effecient Variant of the GMRES(m) Method Based on the Error Equations, *East Asian Journal on Applied Mathematics*, Vol. 2, No. 1, pp. 19-32 (2012).

[5] Lindquist, N., Luszczek, P. and Dongarra, J.: Improving the Performance of the GMRES Method Using Mixed Precision Techniques, *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, pp. 51–66 (2020).

[6] Lindquist, N., Luszczek, P. and Dongarra, J.: Accelerating Restarted GMRES with Mixed Precision Arithmetic, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 33, pp. 1027–1037 (2022).

[7] Demmel, J. W.: *Applied Numerical Linear Algebra*, Society for Industrial and Applied Mathematics (1997).

[8] Higham, N. J.: *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, second edition (2002).

THANKS