Fault Tolerant Matrix-Matrix Multiplication: Correcting Soft Errors On-Line

Panruo Wu, Chong Ding, Longxiang Chen, Teresa Davies, Christer Karlsson, and Zizhong Chen

Colorado School of Mines

November 13, 2011

Errors

- Soft errors
  - We focus on fail-continue soft errors
  - Soft errors have already been reported in tera-scale systems.
- How to tolerate errors
  - Triple Modular Redundancy(TMR)
  - Algorithm Based Fault Tolerance(ABFT)
- We designed a novel ABFT alogrithm for DGEMM
  - on-line: meaning that our algorithm checks the validity of the partial results during computation
  - low overhead: compared with high performance non-fault-tolerant DGEMM

Related Work

- Triple Modular Redundancy(TMR)
  - General technique;
  - Requires a factor of 2 or 3 additional hardware; big overhead
- traditional ABFT:
  - Specific algorithm for every operation
  - Very little overhead;
  - Offline, checks only at the end of computation thus cannot prevent errors from propagating.
- online ABFT
  - Extention to traditional ABFT;
  - very little overhead;
  - online, checks validity of computation in the middle of the computation; more flexible and reliable

Traditional ABFT mm(matrix multiplication)

In 1984 Huang & Abraham proposed the ABFT mm algorithm. The idea is: To compute $C = AB$,

- encode A, B into checksum matrices A^c, B^r
- do mm on encoded matrices $C^f = A^c B^r$
- verify the full checksum relationship of C^f

$$A^c = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \\ \sum_{i=1}^{n} a_{i1} & \dots & \sum_{i=1}^{n} a_{in} \end{bmatrix}$$

$$B^r = \begin{bmatrix} b_{11} & \dots & b_{1n} & \sum_{j=1}^{n} b_{1j} \\ \vdots & & \vdots & \vdots \\ b_{n1} & \dots & b_{nn} & \sum_{j=1}^{n} b_{nj} \end{bmatrix}$$

It turns out that the result of $C^f = A^c B^r$(whatever mm algorithm is used) is a full checksum matrix:

$$C^f = \begin{bmatrix} c_{11} & \dots & c_{1n} & \sum_{j=1}^{n} c_{1j} \\ \vdots & & \vdots & \vdots \\ c_{m1} & \dots & c_{mn} & \sum_{j=1}^{n} c_{nj} \\ \sum_{i=1}^{m} c_{i1} & \dots & \sum_{i=1}^{m} c_{in} & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \end{bmatrix}$$

If the full checksum relationship does not maintain then the mm procedure is faulty. If only one entry c_{ij} is corrupted, then it can be recovered:

$$c_{ij} = \sum_{j=1}^{n} c_{ij}(\text{is } C^f_{i,n+1}) - \sum_{k=1,k\neq j}^{n} c^f_{ik}$$

Online ABFT MM: Motivation

- The traditional ABFT only checks the result at the end of the whole computation.
- What if we can check some partial results in the middle?
  - Handle faults in time
  - More reliable and flexible
- the key problem: does the checksum relationship maintain in the middle of the mm computation?
  - for most MM algorithms, NO
  - for Outer Product MM, YES

Outer product mm algorithm(rank-1 update each iteration)

for $s = 1$ to k **do**
$C = C + A(:, s) * B(s, :)$
end for

$$AB = \left[\ A_1 A_2 \cdots A_n\ \right] \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = A_1 B_1 + \ldots + A_n B_n$$

Theorem

If outer product version of mm is used to compute $A^c \times B^f$, and we denote the partial result C at the end of iteration s by C_s^f, then $C_s^f (s = 1, \ldots, k)$ preserve the full checksum relationship.

Proof: Let $A_s := A(1:n, 1:s)$ and $B_s := B(1:s, 1:n)$ then

$$\begin{aligned} C_s &= A_s^c \times B_s^r \\ &= \begin{bmatrix} A_s \\ e^T A_s \end{bmatrix} \times \begin{bmatrix} B_s & B_s e \end{bmatrix} \\ &= (A_s B_s)^f \end{aligned}$$

We now have at most n oppurtunities to tolerate faults during one mm execution. In practice, we can do the FT routine every several iterations to achieve high performance.

Differentiate soft errors from roundoff errors

- Computers do floating point calculation in finite precision. Therefore the checksum relationship cannot hold *exactly* due to roundoff errors.
- We need a threshold to differentiate soft errors from roundoff errors.
- How to choose the threshold?
  - Too large a threshold may hide errors
  - Too small one may interrupt correct computation

A classic result from matrix analysis(From Matrix Computation)

Theorem

For outer product matrix multiplication $C = AB$, the floating point result $fl(AB)$ satisfies:

$$||fl(AB) - AB||_\infty \leq \gamma_k ||A||_\infty ||B||_\infty =: \lambda$$

where $\gamma_n = \frac{nu}{1-nu}$, and u is the unit roundoff error of the machine.

We begin by assuming the computations are correct, i.e. $C^f = A^c B^r$. As a convention the floating point version of a variable has a hat over the corresponding name. Then we have

$$\left| \sum_{j=1}^{n} \hat{c}_{ij} - \hat{c}_{i,n+1} \right| \leq ||fl(C^f) - C^f||_\infty \leq \gamma_n ||A^c||_\infty ||B^r||_\infty =: \lambda$$

Performance Analysis

- The time complexity of generating row or column checksum for input matrices can be expressed as follow.

$$T_{encode} = 4N^2\gamma \tag{1}$$

- Because of the matrix size increases the overhead of computation is

$$\begin{aligned} T_{comp} &= 2(N+1) \times N \times (N+1) - 2N^3 \\ &= (4N^2 + 2N)\gamma \\ &\approx 4N^2\gamma \end{aligned} \tag{2}$$

- If the program is to tolerate m errors, the overhead to detect a matrix is:

$$T_{detect} = 2mN^2\gamma \tag{3}$$

Experimental Results

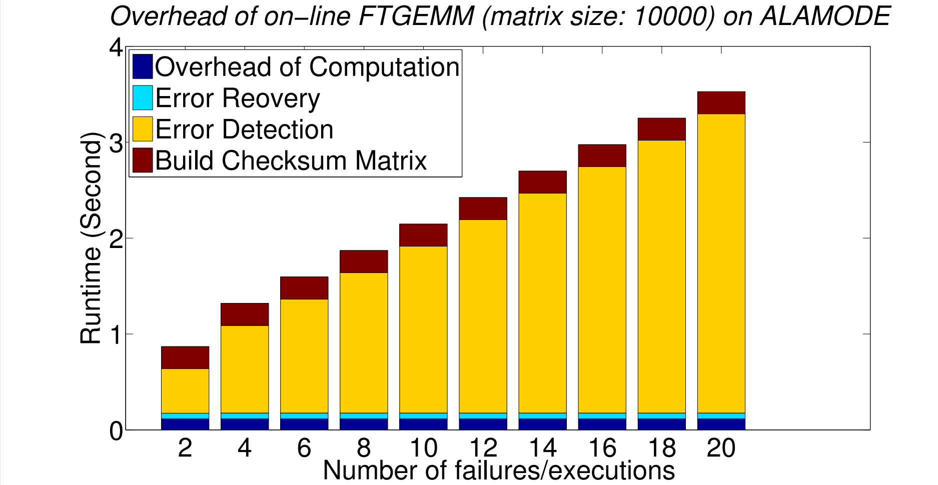
- Performance and overhead of on-line FTGEMM (Fault Tolerant General Matrix Multiply).
- Performance comparison between on-line FTGEMM, ABFT and TMR.
- Performance comparison between on-line FTGEMM and ATLAS.

We show tests done on Alamode machines provided by CCIT in Colorado School of Mines. The CPU one Alamode is Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz and theoretical peak FLOPS is 13.60 GFLOPS. For simplicity we test on square matrix of size 10,000 and 15,000

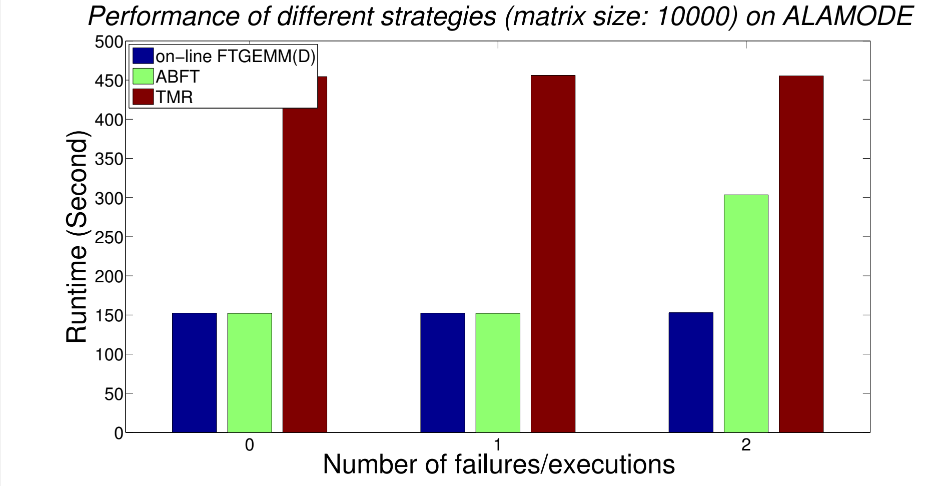
Special note

- The failure rate, or *Number of failures/execution* in the x-axis of all figures and texts in this section is a design parameter which refers to the maximum number of errors our implementation is able to tolerate during one matrix multiplication.
- Be aware that this number does not indicate our implementation is guaranteed to tolerate that many errors; this parameter should be set higher than expected actual failure rate of applications to ensure reliability.

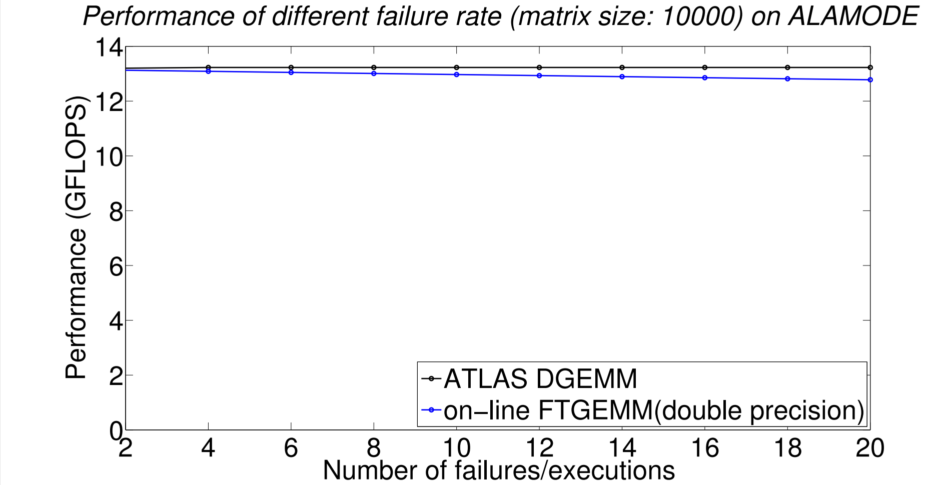
Approx DGEMM(by ATLAS) runtime: 147s



Approx DGEMM(by ATLAS) runtime: 147s



Approx DGEMM(by ATLAS) runtime: 147s



Conclusion

- In this paper we extended the traditional ABFT matrix multiplication technique from off-line to on-line.
- In our approach, soft errors are detected, located, and corrected in the middle of the computation in a timely manner, before they propagate .
- Experimental results demostrate that the proposed technique can correct one error per ten seconds with negligible performance penalty over ATLAS DGEMM()

Thank you

That's all, Thank you!