

High Performance Computing

Yusuke Nagasaka

Tokyo Institute of Technology

Dept. of mathematical and computing sciences

Matsuoka Lab.

Review Paper

- Algorithmic Approaches to Low Overhead Fault Detection for Sparse Linear Algebra [DSN2012]
 - Joseph Sloan
 - University of Illinois
 - Rakesh Kumar
 - University of Illinois
 - Greg Bronevetsky
 - Lawrence Livermore National Laboratory

Sloan, Joseph, Rakesh Kumar, and Greg Bronevetsky. "Algorithmic approaches to low overhead fault detection for sparse linear algebra." Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on. IEEE, 2012.

Fault detection in large scale system

- HPC systems grow large and complex
 - Increase of the number of components
 - Smaller chip size
- Soft error rate will grow
 - Corrupt the computations, produce incorrect output
 - Hardware-based fault detection isn't enough
 - => Algorithm-Based Fault Tolerance

Algorithm-based fault tolerance

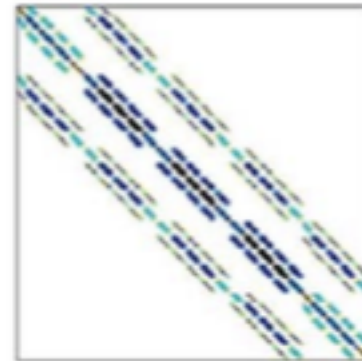
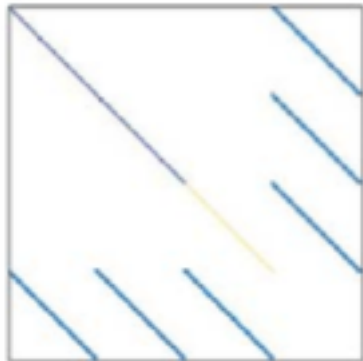
- Software or algorithmic approaches
 - Detect the soft errors with low overhead
- Check sum for matrix vector multiplication (MVM)
 - $c (A x) = (c A) x$: c is check vector
 - $c = (1 \cdots 1)$
 - Low overhead for dense problems
 - Check : $O(N^2) + O(k * N) \Leftrightarrow$ Dense MVM : $O(k * N^2)$
 - High overhead for sparse problems
 - Check : $O(N) + O(k * N) \Leftrightarrow$ Sparse MVM : $O(k * N)$

Contribution

- Reduce the check overhead by using sampling technique
 - Exploiting features of sparse matrix and algorithm
 - Approximate Random, Approximate Clustering
 - Identity Conditioning, Null Conditioning
- Compared to traditional dense check, overhead is reduced up to
 - 50% in sparse matrix vector multiplication
 - 20% in iterative linear solvers

Properties of sparse algorithm

- Sparse applications have
 - Inherent structure
 - Diagonal, banded diagonal, block diagonal
 - Significant reuse
 - Iterative methods : CG, IR



Algorithmic fault detection

Approximate Technique

- Exploiting inherent structure
- Approximate Random (AR)

$$1^T (Ax) \approx ((c^T A)x)s$$

- $c_i = \{0,1\}$, s : scaling factor related to x
- Useful for low variance of column sum

Algorithmic fault detection

Approximate Technique

- Approximate Clustering (AC)
 - For matrices with more variable column sums
 - Clustering on the set of column sum
 - => Randomly sampling the clusters
 - Trade off
 - Additional setup overhead
 - Improving accuracy

Algorithmic fault detection

Conditioning

- Sparse algorithms include reuse of MVM
 - Need to set the low overhead check sum
- Identify Conditioning (IC)

$$(c^T A)x \approx 1^T x = \sum x$$

- Solving $\min |A^t x - 1|$
 - Computing cost is equivalent to multiple MVMs
- Low cost
- Joint IC to AR and AC : ICAR and ICAC
 - Using IC for precondition and AR during the runtime

Algorithmic fault detection

Conditioning

- Null Conditioning (NC)

$$(c^T A)x \approx 0 \quad x = 0$$

- Finding a vector c by computing its smallest singular value using SVD

- $A c = \sigma u$ and $A^* u = \sigma c$ (σ : singular value)

- Smaller singular value provides high accuracy

- Joint NC to AR and AC : NCAR and NCAC

- Using NC for precondition and AR during the runtime

Parameter space

- Fault injection
 - Into the arithmetic operation and check operation
 - Various fault models
 - Fault rates : 0, $1e-6$, $1e-5$, $1e-4$, $1e-3$, $1e-2$, $1e-1$
- Sample rate (AR, AC)
 - 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, ... ,1.0

Metrics

- F-Score
 - Used to summarize an algorithm's effectiveness
 - $F\text{-Score} = 2 * TP / (2 * TP + FP + FN)$
 - TP : True positives, detect the fault
 - FP : False positives, detector signals when no fault
 - FN : False negatives, not detect the fault
- Choose the best technique
 - Oracle : so that F-Score is best
 - Decision Tree
 - More practical



Results and Analysis

Experiment condition

- Datasets
 - University of Florida Sparse Matrix Collection
- Benchmarks
 - Matrix vector multiplication (MVM)
 - Iterative linear solver
 - Conjugate Gradient (CG)
 - Iterative Refinement (IR)

Evaluation

MVM

- Compare each detection technique
 - AR, AC and IC show same accuracy as traditional dense check
 - NC achieved F-Score above 0.9 for **less than 10%**
 - Smallest singular value is large
 - Eigenvectors have many zeros
 - Faults is masked

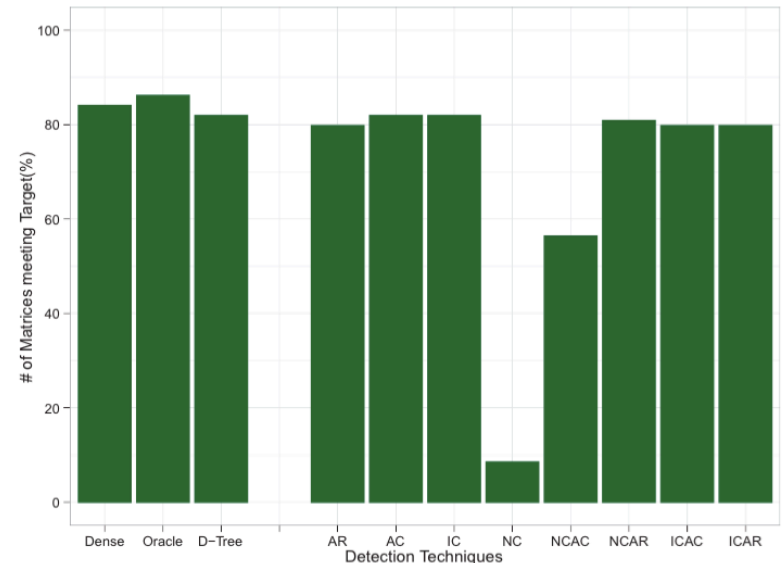


Fig. 7. Number of problems meeting F-Score target. F-Score target=0.9, Fault Rate= $1e - 3$, FaultModel=1

Evaluation

MVM

- Compare each detection technique
 - Overhead of AR was **50% lower**
 - AC was useful for lower variance pattern matrix
 - NC was useful for what contain small singular values

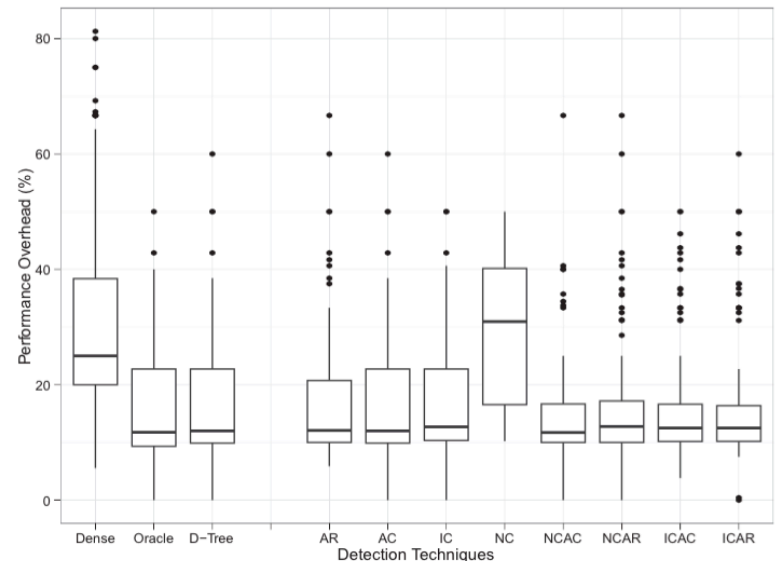


Fig. 6. Runtime overhead of each technique. F-Score target=0.9, Fault Rate= $1e-3$, FaultModel=1

Evaluation

MVM (Less frequent fault rate)

- Dense check becomes worse
 - Faults are likely to occur in the check operations
- Approximate checks work well

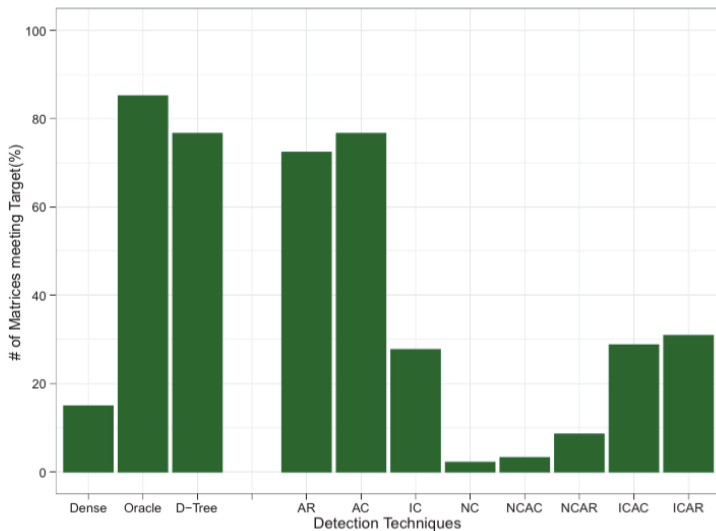


Fig. 9. Number of problems meeting F-Score target. F-Score target=0.9, Fault Rate=1e-6, FaultModel=1

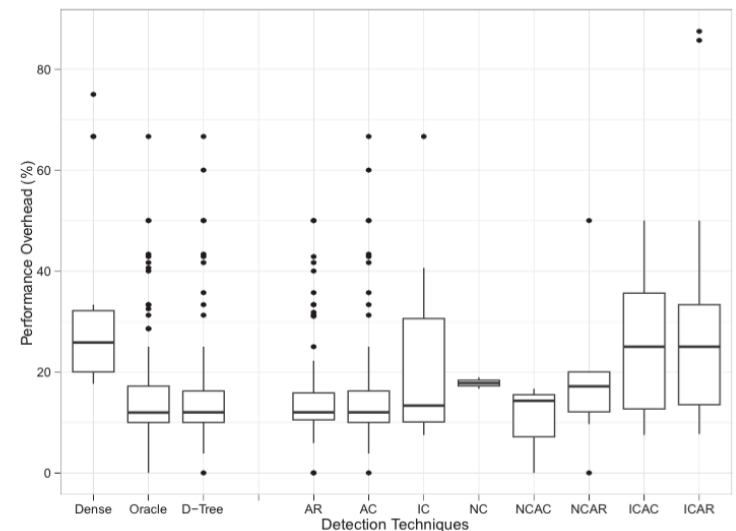
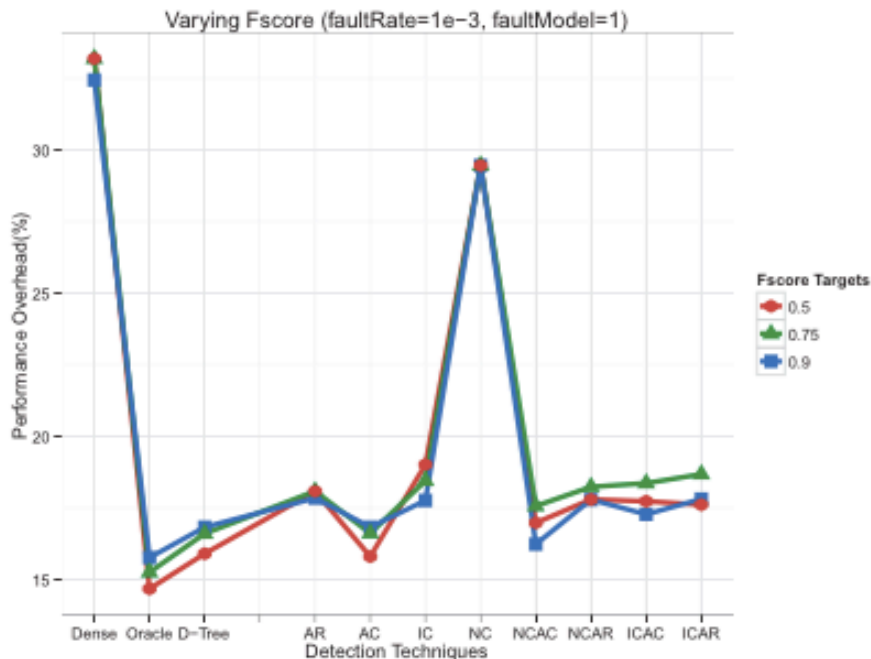


Fig. 8. Runtime overhead of each technique. F-Score target=0.9, Fault Rate=1e-6, FaultModel=1

Evaluation

MVM

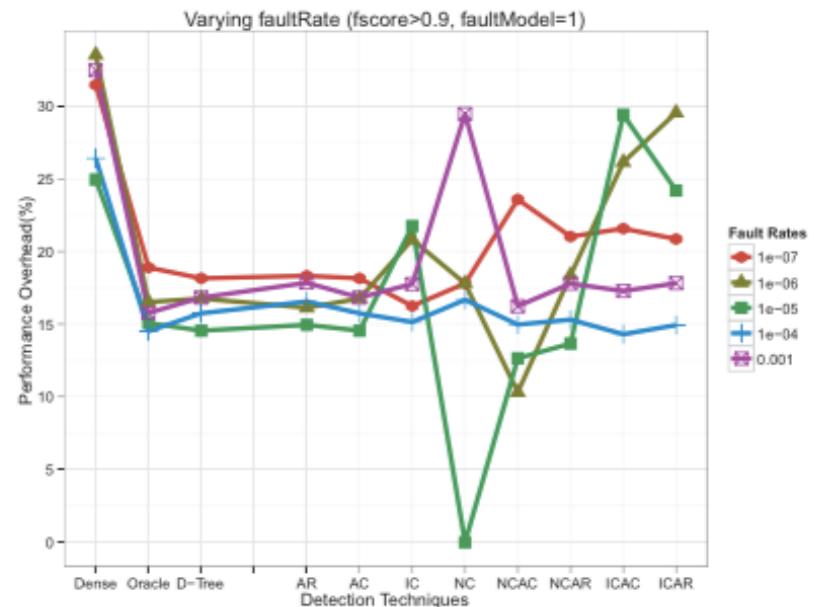
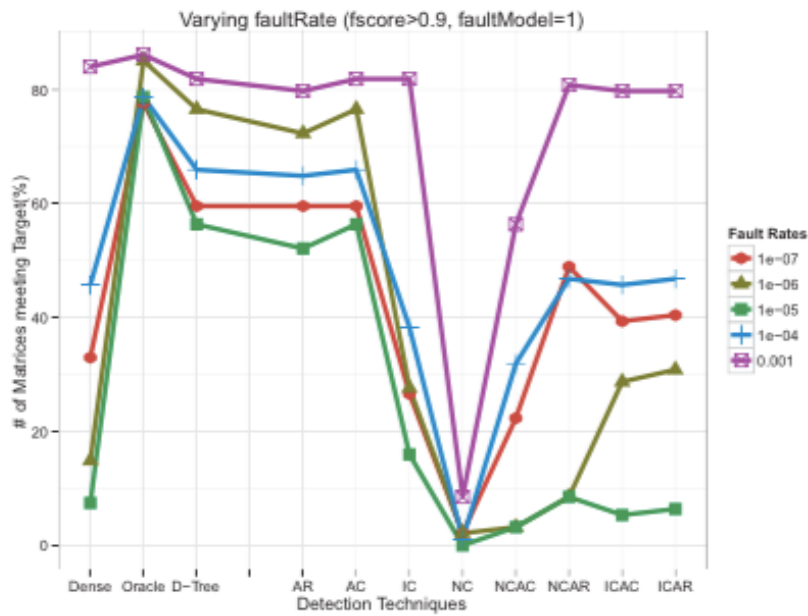
- Across different F-Score targets, fault models
 - Overhead is not sensitive to these parameters



Evaluation

MVM

- Across different fault rates
 - Detection is most difficult in the middle fault rate
 - Tree algorithm is resilient



Evaluation

Linear solvers

- Evaluation of iterative method
 - CG and IR

- Overhead includes set up
 - Conditioning

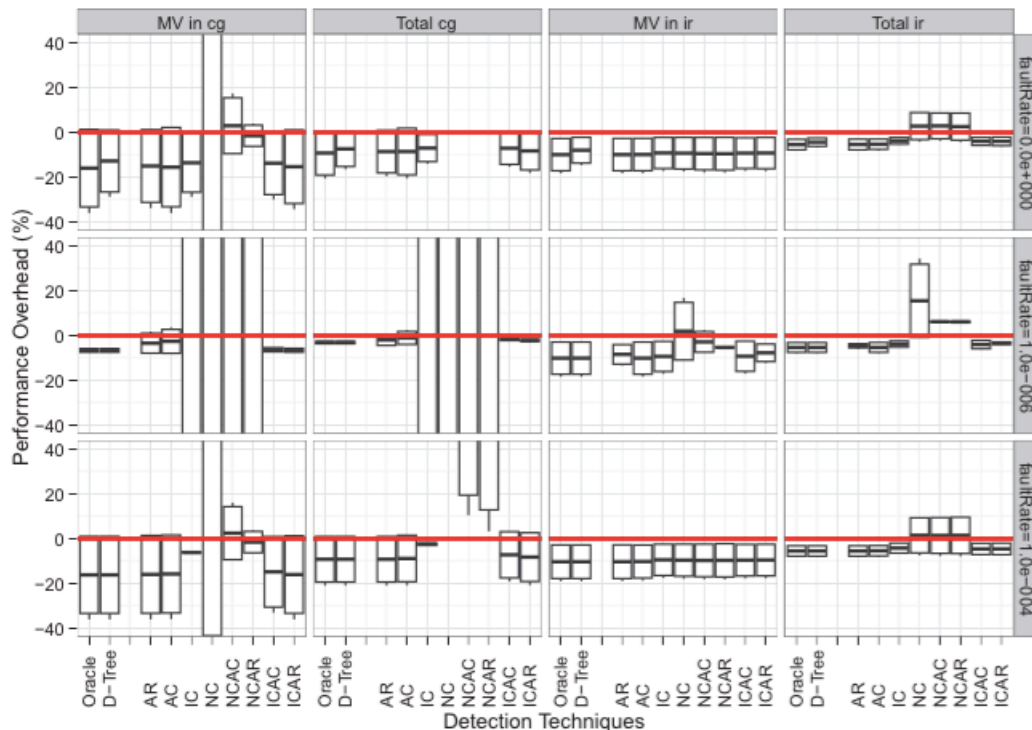
$$Overhead = \frac{Time_sparse - Time_dense}{Time_dense}$$

- Problems used with preconditioning solvers achieved significant benefits (> 2x)
 - 5 problems for CG, 1 problem for IR

Evaluation

Linear solvers

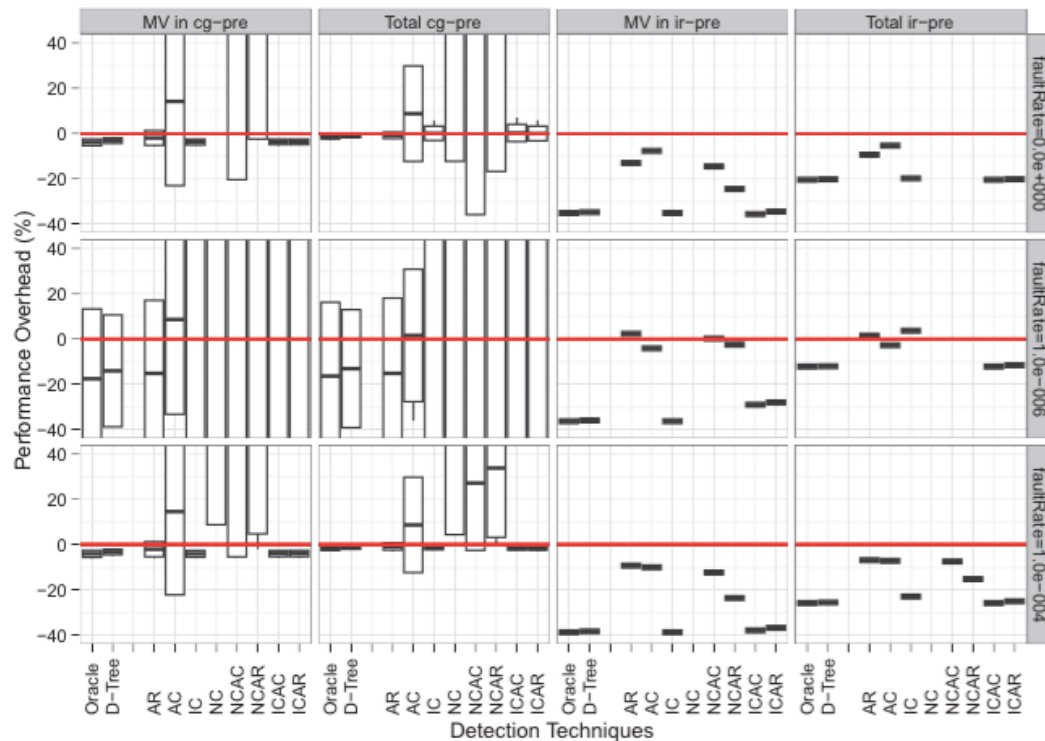
- Sparse check reduce the overhead
 - 17% less time in MVM for CG on average
 - 9% less time in total for CG



Evaluation

Linear solvers with preconditioning

- Sparse checks with small overhead on average
 - CG-pre : - 5% ~ - 10%
 - IR-pre : - 30% ~ - 40%



Conclusion

- Sparse check technique reduce the overhead of fault detection from traditional dense check exploiting the properties of sparse algorithms
 - Approximate check : AR, AC
 - Conditioning : IC, NC
 - Up to 2x over in MVM
 - Effective for the iterative solver

Discussion

- Apply sparse check technique to
 - Unstructured sparse matrix
 - Dense matrix
- Result of CG-per and IR-pre are not enough
 - Few dataset
- Not enough information about
 - Datasets
 - Experiment environment