



**Erasure Coded Computations**

# **New Models for Fault Tolerance**

Ph.D. Candidate

Advisor

**Xuejiao Kang, David Gielch, Ahmed  
Sameh, Ananth Grama**



Department of  
Computer Science

November 1, 2022

**Background  
and Motivation**

1

Adaptive Fault Tolerant  
Linear System Solver

3

2

Distributed Fault Tolerant  
Linear System Solver

4

Erasure Coded  
Eigensolver

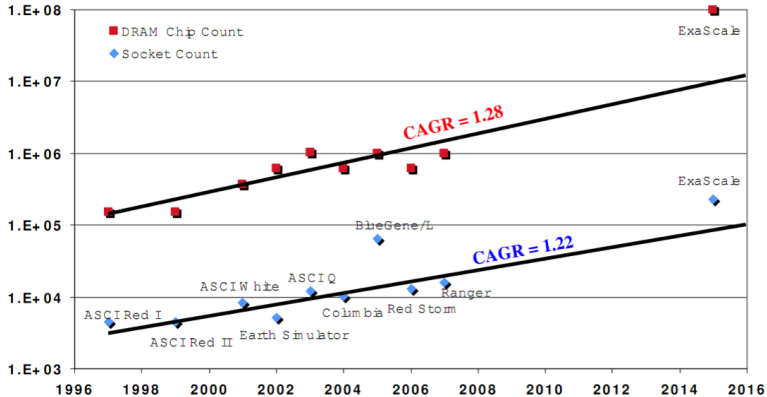
Plan of Research

5

# Faults in Parallel and Distributed Systems

- As parallel and distributed systems scale to millions of cores, faults become one of the most critical challenges.
- As data centers scale to hundreds of thousands of nodes, faults are a prime consideration for distributed computations.
- As networks scale from data center to wide area, network faults and partitions constitute a major consideration for wide area distributed computations.

# Estimated Chip Counts in Exascale Systems



Source: DARPA Exascale Technology Study [Kogge et al.]



# BlueGene Failure In Time (FIT) budget

<i>Component</i>	<i>FIT per component<sup>†</sup></i>	<i>Components per 64Ki compute node partition</i>	<i>FITs per system (K)</i>	<i>Failure rate per week</i>
Control-FPGA complex	160	3,024	484	0.08
DRAM	5	608,256	3,041	0.51
Compute + I/O ASIC	20	66,560	1,331	0.22
Link ASIC	25	3,072	77	0.012
Clock chip	6.5	~1,200	8	0.0013
Nonredundant power supply	500	384	384	0.064
Total (65,536 compute nodes)			5,315	0.89

<sup>†</sup> $T = 60^{\circ}\text{C}$ ,  $V = \text{Nominal}$ , 40K POH. *FIT* = Failures in ppm/KPOH. One FIT =  $0.168 \times 10^{-6}$  fails per week if the machine runs 24 hours a day.

Source: P. COTEUS ET AL., IBM J. RES. & DEV. VOL. 49 NO. 2/3

# Fault and Failure Models

## Types of Fault

(i) Permanent;                      (ii) Transient;                      (iii) Intermittent.

## Failure Model

Failure model is an abstraction of system behavior in the presence of a fault.

- Byzantine: a component can exhibit **arbitrary and malicious behavior**, perhaps involving collusion with other faulty components.
- Fail-stop: a component changes to a **state** that permits other components to **detect** the failure and then **stops**.

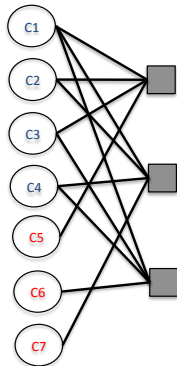
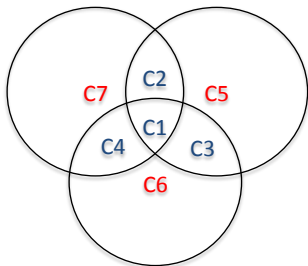
# Fault Tolerance

Algorithm-based methods and System-supported methods.

- Algorithmic methods alter the algorithm to make it robust to faults.
- System-supported methods include checkpoint-restart, active replicas and deterministic replay.
  - Checkpoint-restart schemes involve the overhead of consistent checkpointing and I/O.
  - Active replicas execute multiple replicas of each task.
  - Tasks in deterministic replay are scheduled at different execution units and monitored for successful completion. They are rescheduled at other execution units if failures are detected.

# Error Correcting Codes

Error Correcting Codes are commonly used in communication systems. Messages are coded (augmented) with error correction (redundant) bits, which can help detect and correct transmission errors.



Parity check equations:

$$C_1 \oplus C_2 \oplus C_3 \oplus C_5 = 0$$

$$C_1 \oplus C_3 \oplus C_4 \oplus C_6 = 0$$

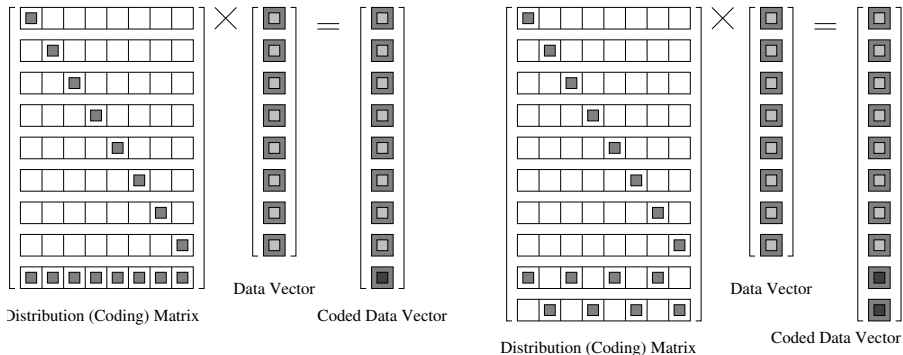
$$C_1 \oplus C_2 \oplus C_4 \oplus C_7 = 0$$

Parity check matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

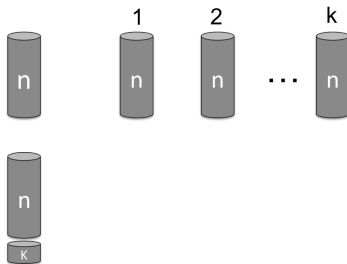
# Erasure Coded Storage

Erasure codes are derived from the Error Correcting Codes and are typically used in storage systems to deal with known data erasures or loss.



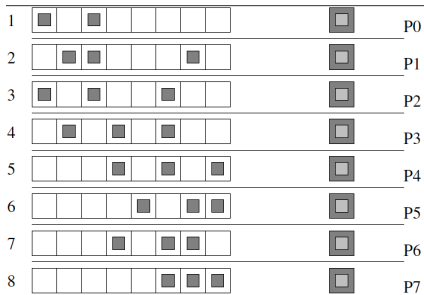
# Fault Tolerant Storage--Replication v.s. Erasure Coding

- Replication based schemes maintain as many copies of data as are needed to guarantee required tolerance.
- Erasure coding schemes transform the data so that the original data can be reconstructed from (a subset of) the available coded data.

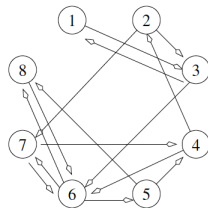


# Erasure Coded Computation Basic Kernels

We show the idea for erasure coded computations using a simple example of a sparse matrix-vector product.



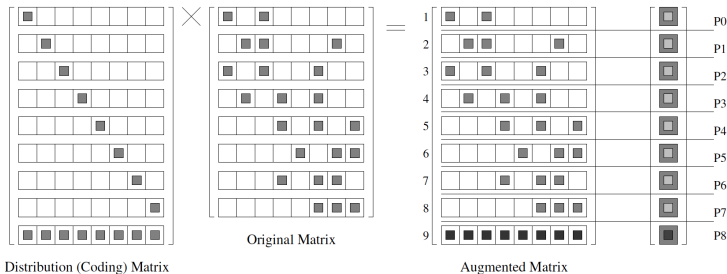
(a) Original Matrix



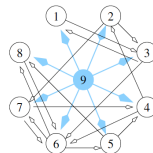
(b) Graph of Original Matrix

# Erasure Coded Computation Basic Kernels

We multiply the distribution matrix with the given sparse matrix. This results in an augmented matrix.



(c) Augmentation Process: Multiplication by a distribution/ coding matrix.  
Augmented matrix is tolerant to one node (row) failure.

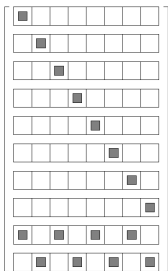


(d) Graph of augmented mat

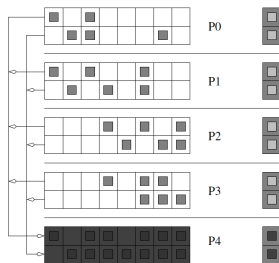


# Erasure Coded Computation Basic Kernels

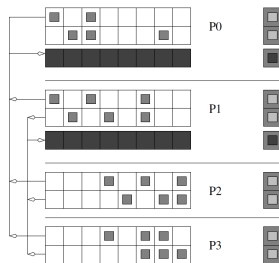
Example of an alternative distribution matrix, which allows us to control the fill in the augmentation rows. We also show the augmentation block distribution across processors.



Distribution (Coding) Matrix



Controlling density of augmented blocks by combining blocks (first row at each processor is combined into first row of the augmented block). Note the reduced fill. Matvec is still tolerant to one process failure.



Augmentation block is now distributed across processors. This addresses problem of load imbalance. Note higher communication cost of this scheme. This can be amortized through coarse-grained processor partitions.

# Goals and Challenges

## Goal

- Code the result of a computation in a fault tolerant manner (in storage or communication, the computation is an identity operator).
- Data is not a linear bit string or a block – but rather it is a sparse matrix.
- Recovery of result of computation must be easy.

## Challenges

- Design suitable coding matrices.
- Reformulate traditional linear algebraic methods in the erasure coding framework.
- Design efficient recovery algorithms.
- Analyze parallel performance.
- Validate tolerance to different models of fault arrivals and rates.

# Fault Oblivious Computation

The concept of fault oblivious parallel execution, based on [Erasure Coded Computations](#), is as follows:

- [Augment](#) the input to a parallel program.
- [Execute](#) on the augmented input in a faulty environment, oblivious to faults, and generate an augmented output.
- [Compute](#) the true output based on the augmented output from the faulty execution.

Background  
and Motivation

1

Adaptive Fault Tolerant  
Linear System Solver

3

Plan of Research

5

2

Distributed Fault Tolerant  
Linear System Solver

4

Erasure Coded  
Eigensolver

///

# Erasure Coded Linear System Solver

Given a linear system  $\mathbf{Ax} = \mathbf{b}$  ( $\mathbf{A}_{n \times n}$  is *SPD*), with true solution  $\mathbf{x}^*$ , and coding matrix  $\mathbf{E}_{n \times k}$ , we construct the augmented system  $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$ .

$$\underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{AE} \\ \mathbf{E}^T\mathbf{A} & \mathbf{E}^T\mathbf{AE} \end{bmatrix}}_{\tilde{\mathbf{A}}} \underbrace{\begin{bmatrix} \mathbf{x} \\ \mathbf{r} \end{bmatrix}}_{\tilde{\mathbf{x}}} = \underbrace{\begin{bmatrix} \mathbf{b} \\ \mathbf{E}^T\mathbf{b} \end{bmatrix}}_{\tilde{\mathbf{b}}}$$

$\tilde{\mathbf{x}} = [\mathbf{x}^*; 0]$  is a solution to the augmented system.

# Erasure Coded Linear System Solver

Properties of  $\tilde{\mathbf{A}}$ :

- If  $\mathbf{A}$  is *SPD*, then  $\tilde{\mathbf{A}}$  is *SPSD*;
- The null space basis of  $\tilde{\mathbf{A}}$  is  $\begin{bmatrix} \mathbf{E} \\ -\mathbf{I}_k \end{bmatrix}$ ;
- Any solution of the augmented system can be written as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^* \\ 0 \end{bmatrix} + a \begin{bmatrix} \mathbf{E} \\ -\mathbf{I}_k \end{bmatrix}$$

## Theorem

Based on the properties of  $\tilde{\mathbf{A}}$ , we can recover the true solution as follows:

$$\begin{bmatrix} \mathbf{x}^* \\ 0 \end{bmatrix} = \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{E} \\ -\mathbf{I}_k \end{bmatrix} \mathbf{r} \quad (1)$$

# Erasure Coded Linear System Solver

When faults happen, we can compress the augmented system to a purified system.

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{Z}_1 \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{Z}_2 \\ \mathbf{Z}_1^T & \mathbf{Z}_2^T & \mathbf{E}^T \mathbf{A} \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{f} \\ \mathbf{r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{E}^T \mathbf{b} \end{bmatrix}$$

↓

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{Z}_1 \\ \mathbf{Z}_1^T & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{E}^T \mathbf{b} \end{bmatrix} - \begin{bmatrix} \mathbf{A}_{12} \\ \mathbf{Z}_2^T \end{bmatrix} \mathbf{f}$$

## Theorem

If  $[c; r]$  is a solution for the purified system, then  $\tilde{\mathbf{x}} = [c; f; r]$  is a solution to the augmented system.

# Coding Matrix

## Conditions on matrix $E$ :

- There is always a solution to the augmented system for faults happening on any rows, as long as total number of faults  $\leq k$ .
- Given any solution computed with faulty components, we can extract and recover a solution for the original system.

## Desiderata of $E$ :

- Satisfy properties of Kruskal rank, which means every subset of  $k$  rows of matrix  $E$  is linearly independent.
- Be as sparse as possible to minimize the fill in the augmented matrix.



# Coding Matrix

## Definition

An  $n \times k$  matrix  $E$  satisfies the recovery-at-random property if a random subset of  $k$  rows (selected uniformly with replacement) is rank  $k$  with probability approaching 1.

## Proposed Coding Matrix $E$

$$\begin{bmatrix} \bullet & \bullet & \bullet & 0 & 0 & 0 \\ 0 & \bullet & \bullet & \bullet & 0 & 0 \\ 0 & 0 & \bullet & \bullet & \bullet & 0 \\ 0 & 0 & 0 & \bullet & \bullet & \bullet \\ \bullet & 0 & 0 & 0 & \bullet & \bullet \\ \bullet & \bullet & 0 & 0 & 0 & \bullet \end{bmatrix}$$

# Coding Matrix

## Proposition

Let  $p$  be the number of nonzeros per row in  $E$  and Let  $E'$  be a submatrix of  $E$  formed by selecting any  $p$  rows of matrix  $E$ . The matrix  $E'^T$  has rank  $p$ .

All rows have distinct non-zero structure.

$$\begin{bmatrix} \bullet & \bullet & \bullet & & & \\ & \bullet & \bullet & \bullet & & \\ & & \bullet & \bullet & \bullet & \end{bmatrix}$$

All rows have same non-zero structure.

$$\begin{bmatrix} \bullet & \bullet & \bullet & 0 & 0 & 0 \\ \bullet & \bullet & \bullet & 0 & 0 & 0 \\ \bullet & \bullet & \bullet & 0 & 0 & 0 \end{bmatrix}$$

Some rows from case1 and Some from case2.

$$\begin{bmatrix} \bullet & \bullet & \bullet & 0 & 0 & 0 \\ 0 & \bullet & \bullet & \bullet & 0 & 0 \\ 0 & \bullet & \bullet & \bullet & 0 & 0 \end{bmatrix}$$

# Coding Matrix

## Theorem

The probability that a randomly chosen set of  $k$  rows from the matrix  $E$  are linearly dependent is less than  $\left(\frac{e}{p+1}\right)^{p+1}$ .

*Proof:* A sufficient condition for  $k$  rows to be linearly dependent is that some selection of  $p+1$  rows from these  $k$  rows have the same non-zero structure. There are  $k$  distinct non-zeros structures for the matrix  $E$ .

$$\binom{k}{p+1} \left(\frac{1}{k}\right)^{p+1} \leq \left(\frac{e}{p+1}\right)^{p+1}.$$

As  $p$  increases, this probability rapidly approaches 0, which means that the matrix  $E$  is recovery-at-random.

# Coding Matrix

To keep the coding matrix and the associated augmented matrix sparse,  $p$  should be as small as possible.

## Theorem

The expected number of rows from among  $k$  randomly selected rows of matrix  $E$  that have same nonzero structure is  $O\left(\frac{\ln k}{\ln \ln k}\right)$ .

Define a random variable  $M$  to be the number of rows that have the same non-zero structure when we select  $k$  rows uniformly at random from the matrix  $E$ .

$$Pr(M = t) = \binom{k}{1} \binom{k}{t} \left(\frac{1}{k}\right)^t \left(1 - \frac{1}{k}\right)^{k-t} \leq k \left(\frac{e}{t}\right)^t$$

# Coding Matrix

The expected number of rows  $E(M)$  is given by:

$$\begin{aligned} E(M) &= \sum_{t=1}^k t \cdot \Pr(M = t) \\ &= \sum_{t=1}^{c \frac{\ln k}{\ln \ln k}} t \cdot \Pr(M = t) + \sum_{t=\frac{c \ln k}{\ln \ln k}}^k t \cdot \Pr(M = t) \\ &\leq \sum_{t=1}^{\frac{c \ln k}{\ln \ln k}} \frac{c \ln k}{\ln \ln k} \cdot \Pr(M = t) + \sum_{t=\frac{c \ln k}{\ln \ln k}}^k k \cdot \Pr(M = t) \\ &\leq \frac{c \ln k}{\ln \ln k} + k \cdot \frac{1}{k^{c/2-1}} \\ &= O\left(\frac{\ln k}{\ln \ln k}\right) \end{aligned} \tag{2}$$

## Parallel Implementation

Since  $A$  is SPD and  $\tilde{A}$  is SPSD, we can apply CG to  $\tilde{A}\tilde{x} = \tilde{b}$ .

---

### Algorithm Fault Oblivious CG with a Two-term Recurrence

---

- 1: Let  $x_0$  be the initial guess and  $r_0 = b - Ax_0$ ,  $\beta_0 = 0$ .
  - 2: **for**  $t = 0, 1, \dots$  until convergence **do**
  - 3:   **if** Fault detected **then**  $\beta_t = 0$  **else**  $\beta_t = \frac{\|r_t\|_2^2}{\|r_{t-1}\|_2^2}$
  - 4:    $p_t = r_t + \beta_t p_{t-1}$
  - 5:    $q_t = Ap_t$
  - 6:    $\alpha_t = \frac{\|r_t\|_2^2}{\langle q_t, p_t \rangle}$
  - 7:    $x_{t+1} = x_t + \alpha_t p_t$
  - 8:    $r_{t+1} = r_t - \alpha_t q_t$
-

## Parallel Implementation

Assume that each viable process can detect the breakdown of its neighbor processes.

- Inner products  $\langle \mathbf{r}_t, \mathbf{r}_t \rangle$  and  $\langle \mathbf{q}_t, \mathbf{p}_t \rangle$ .

$$\begin{aligned}\langle \mathbf{r}_t, \mathbf{r}_t \rangle &= \langle (\mathbf{r}_t)_{[n+k] \setminus F_t}, (\mathbf{r}_t)_{[n+k] \setminus F_t} \rangle \\ \langle \mathbf{q}_t, \mathbf{p}_t \rangle &= \langle (\mathbf{q}_t)_{[n+k] \setminus F_t}, (\mathbf{p}_t)_{[n+k] \setminus F_t} \rangle\end{aligned}\tag{3}$$

- Matrix-vector multiplication  $\mathbf{q}_t = \mathbf{A}\mathbf{p}_t$ .

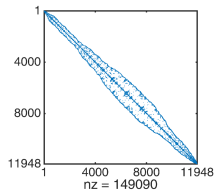
$$\mathbf{A}_{l_i, :} \mathbf{p}_t = \mathbf{A}_{l_i, [n+k] \setminus F_t} (\mathbf{p}_t)_{[n+k] \setminus F_t}\tag{4}$$

- When observing a fault, we truncate the update  $\mathbf{p}_t = \mathbf{r}_t + \beta_t \mathbf{p}_{t-1}$  to be

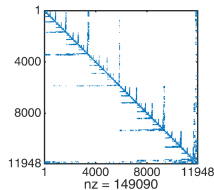
$$\mathbf{p}_t = \mathbf{r}_t.\tag{5}$$

This corresponds to a reset of the Krylov process.

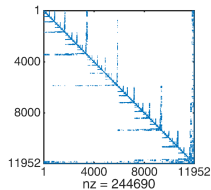
# Reordering and Partitioning



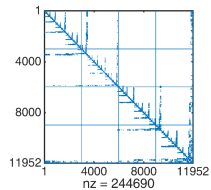
(a) original matrix



(b) original reordered matrix



(c) augmented matrix



(d) augment reordered matrix



## Experimental Data

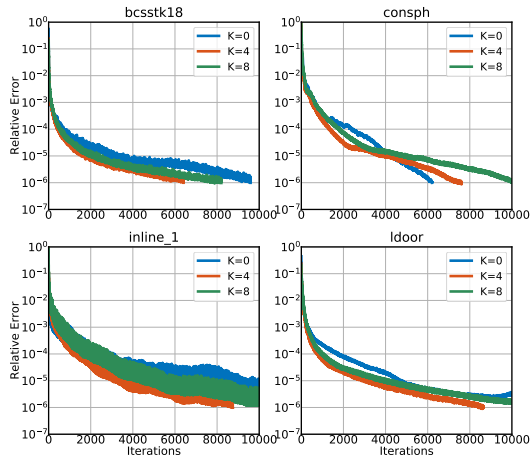
We select matrices from the University of Florida Matrix Collection for our tests.

bcsstk18	11,948	149,090
consph	83,334	6,010,480
inline_1	503,712	36,816,170
ldoor	952,203	42,493,817

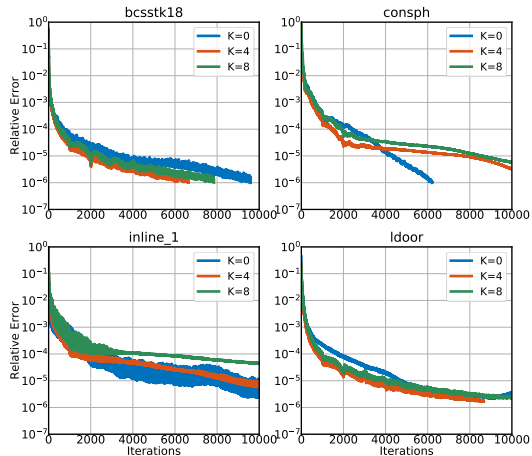
## Experiment Setup

- The right-hand-side vector  $\mathbf{b}$  is first normalized (which means  $\|\mathbf{b}\|_2 = 1$ ). The relative error  $rtol = \frac{\|A\mathbf{x} - \mathbf{b}\|_2}{\|\mathbf{b}\|_2}$  equals the residual norm  $\|r\|_2 = \|A\mathbf{x} - \mathbf{b}\|_2$ .
- The termination condition is set to  $\|r\|_2 < 10^{-6}$  for all matrices, and the maximum number of iterations for CG is set to 10000.
- For the instantaneous fault arrival model, faults arrive at the 1000-th iteration.
- For the exponential fault arrival model, the fault rate is set as  $10^{-3}$ , which implies the average number of steps between two consecutive faults is 1000.

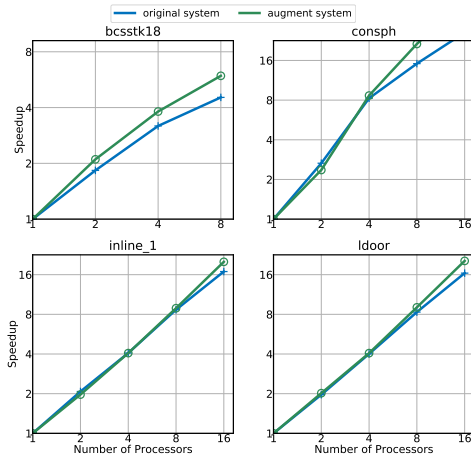
# Convergence--Fault-Free Mode



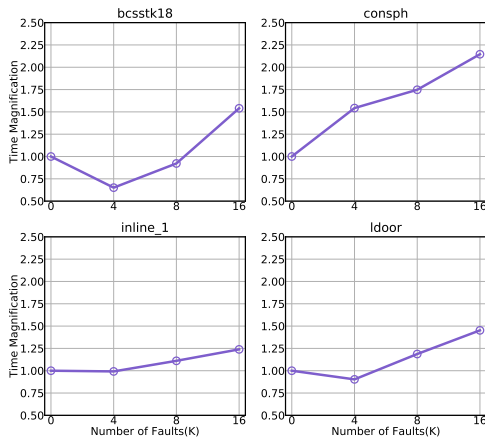
# Convergence--Faulty Mode



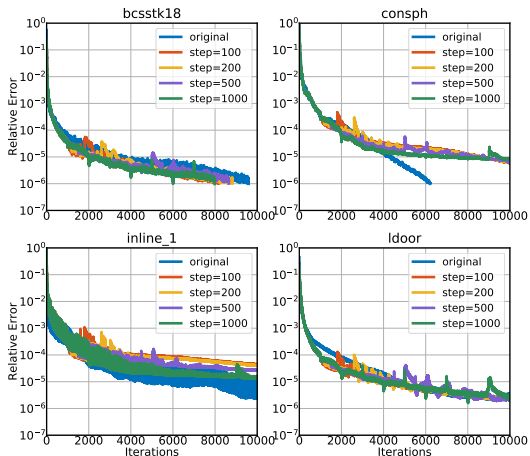
# Speedup



# Time Overhead



# Convergence under Different Fault Rates ( $K = 8$ )



## Different Fault Models

The exponential distribution is the most commonly used random fault arrival model. It assumes the time to failure to be exponentially distributed.

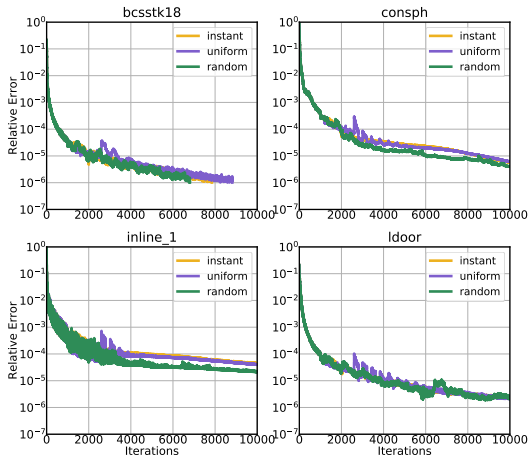
The probability distribution function (PDF) of the time ( $\tau$ ) to failure is given by:

$$P_e(t < \tau) = 1 - e^{-r_e \tau}$$

Here  $r_e$  is the failure rate.



# Convergence under Different Fault Arrival Models ( $K = 8$ )



# Conclusions

We show how to take recently proposed erasure coding schemes and apply them efficiently on parallel/ distributed platforms.

- Creating a new encoding matrix that satisfies recovery requirements for almost all sets of failing components;
- Partitioning these matrices and demonstrating their suitability w.r.t. parallel execution (low fill, low added communication);
- Demonstrate good parallel performance and low time overhead;
- Robustness to different fault arrival models and arrival rates.

Our proposed erasure coded computation scheme is general, and can be applied to a number of other problems.

Background  
and Motivation

Adaptive Fault Tolerant  
Linear System Solver

Plan of Research



Distributed Fault Tolerant  
Linear System Solver

Erasure Coded  
Eigensolver

# Adaptive Fault Tolerant Linear System Solver

The Distributed Fault Tolerant Linear System Solver runs the augmented system from the beginning and can tolerate as many faults as the size of augmentation block during the execution.

- System size is augmented ( $n \rightarrow n + k$ ).
- System property changes (SPD  $\rightarrow$  SPSD).
- Computational overhead paid at each iteration.

# Adaptive Fault Tolerant Linear System Solver

Adaptive Fault Tolerant Linear System Solver runs on the original system until a fault occurs. The erased blocks are compensated for by the addition of an identical number of rows (and columns) selected from the pre-computed coding blocks  $[E^T A, E^T A E]$ .

- System size is the same (always  $n$ ).
- System property is maintained (always SPD).
- Computational Overhead is negligible.

# Adaptive Fault Tolerant Linear System Solver

The initial solution of the original system can be written as:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} \begin{bmatrix} x_c \\ x_f \end{bmatrix} = \begin{bmatrix} b_c \\ b_f \end{bmatrix} \quad (6)$$

The augmented system can now be written as:

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{Z}_1 \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{Z}_2 \\ \mathbf{Z}_1^T & \mathbf{Z}_2^T & \mathbf{E}^T \mathbf{A} \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{x}_c \\ \mathbf{x}_f \\ \mathbf{x}_r \end{bmatrix} = \begin{bmatrix} \mathbf{b}_c \\ \mathbf{b}_f \\ \mathbf{E}^T b \end{bmatrix} \quad (7)$$

After erasures, we solve the new system:

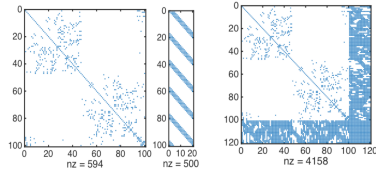
$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{Z}_1 \\ \mathbf{Z}_1^T & \mathbf{E}^T \mathbf{A} \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{x}_c \\ \mathbf{x}_r \end{bmatrix} = \begin{bmatrix} \mathbf{b}_c \\ \mathbf{E}^T b \end{bmatrix} - \begin{bmatrix} \mathbf{A}_{12} \\ \mathbf{Z}_2^T \end{bmatrix} \mathbf{x}_f \quad (8)$$

# Adaptive Fault Tolerant Linear System Solver

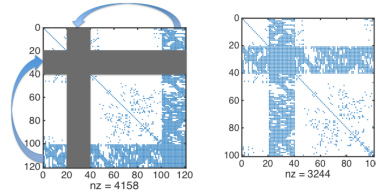
## Algorithm Adaptive Fault Oblivious CG

1: (Reliably) Compute and save the entries $Z_1, Z_2, E^T A E, E^T b$ for matrix $E$	13: $\alpha_t = \frac{\ r_{t-1}\ _2^2}{\langle q_t, p_t \rangle}$
2: $A^{(cur)} = A$	14: $x_t = x_{t-1} + \alpha_t p_t$
3: $b^{(cur)} = b$	15: $r_t = r_{t-1} - \alpha_t q_t$
4: $x_0 =$ the initial guess	16: if Faults detected then
5: $r_0 = b^{(cur)} - A^{(cur)} x_0$	17: $A^{(cur)} = \begin{bmatrix} A_{11} & \tilde{Z}_1 \\ \tilde{Z}_1^T & \tilde{E}^T A \tilde{E} \end{bmatrix}$
6: $\beta_0 = 0$	18: $b^{(cur)} = \begin{bmatrix} b_c - A_{12} x_f \\ \tilde{E}^T b - \tilde{Z}_2^T x_f \end{bmatrix}$
7: for $t = 1, \dots$ until convergence do	19: $x_t = \begin{bmatrix} x_c \\ 0 \end{bmatrix}$
8:   if Fault detected then	20: $r_t = b^{(cur)} - A^{(cur)} x_t$
9: $p_t = r_{t-1}$	
10:   else	
11: $p_t = r_{t-1} + \frac{\ r_{t-1}\ _2^2}{\ r_{t-2}\ _2^2} \cdot p_{t-1}$	
12: $q_t = A^{(cur)} p_t$	

# Reordering and Partition



(a) Input matrix  $A$  (left) and Coding matrix  $E$  (right)



(c) Compensation of erasures in input matrix (d) Compensated matrix (of same size) after erasures



## Experimental Data

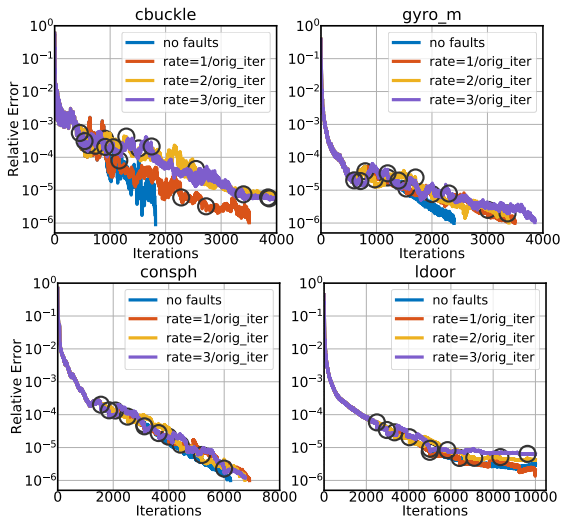
We select matrices from the University of Florida Matrix Collection for our tests – `cbuckle` and `gyro_m` are used to validate the convergence of adaptive fault tolerant linear solver; `consph` and `ldoor` are used to validate parallel scalability and robustness to different fault arrival models.

<code>cbuckle</code>	13,681	676,515
<code>gyro_m</code>	17,361	340,431
<code>consph</code>	83,334	6,010,480
<code>ldoor</code>	952,203	42,493,817

## Experiment Setup

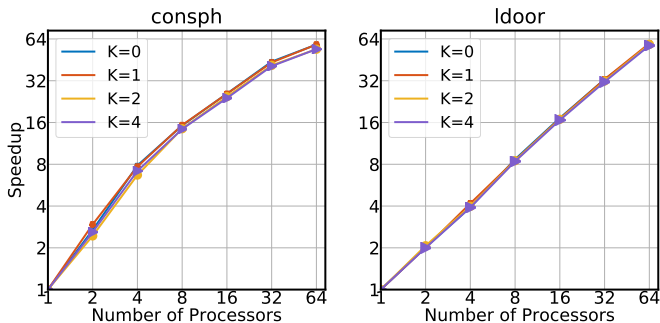
- The right hand side  $\mathbf{b}$  is normalized ( $\|\mathbf{b}\|_2 = 1$ ). The relative residual  $rtol = \frac{\|\mathbf{Ax} - \mathbf{b}\|_2}{\|\mathbf{b}\|_2}$  (equals to  $\|\mathbf{r}\|_2 = \|\mathbf{Ax} - \mathbf{b}\|_2$ ) is calculated.
- $\|\mathbf{r}\|_2$  is monitored at each iteration and the termination condition is set as  $\|\mathbf{r}\|_2 < 10^{-6}$  and the maximum number of iterations of CG is set to 10000 for all matrices.
- For parallel performance, the matrices are first reordered using Metis.
- For exponential fault arrival model, different fault rates( $r_e$ ) ranging from  $\frac{1}{orig\_iter}$  to  $\frac{3}{orig\_iter}$  are tested.
- In our tests, we set the first fault to happen at  $\frac{orig\_iter}{1+orig\_iter/r_e}$ .

# Convergence



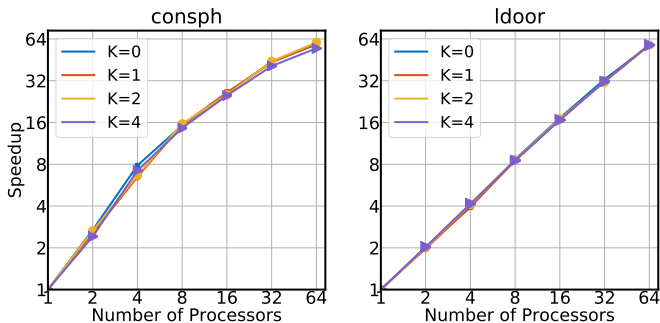
# Speedup

First, we show the parallel performance of adaptive linear solver under exponential fault arrival model.



# Speedup

We all show the parallel performance of adaptive linear solver under instantaneous fault arrival model.



# Overhead

The time overhead of adaptive linear solver is:



# Conclusion

We have developed an **Adaptive Fault Tolerant Linear System Solver** capable of scaling to large numbers of processors and associated faults.

- Coding blocks are only added when faults are detected;
- Convergence properties of adaptive system closely follow those of original system;
- The solver is robust to a wide range of fault characteristics.
- Speedup is near linear and time overhead is small;

Background  
and Motivation

1

Adaptive Fault Tolerant  
Linear System Solver

3

2

Distributed Fault Tolerant  
Linear System Solver

4

Erasure Coded  
Eigensolver

Plan of Research

5



# Reformalization

## Theorem

Given an eigenvalue problem

$$Ax^* = \lambda^* x^*, \quad (9)$$

where  $A \in \mathbb{R}^{n \times n}$ . We construct a generalized eigenvalue problem:

$$\underbrace{\begin{bmatrix} A & AE \\ E^T & E^T AE \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} x \\ r \end{bmatrix}}_{\tilde{x}} = \lambda \underbrace{\begin{bmatrix} I & E \\ E^T & E^T E \end{bmatrix}}_{\tilde{B}} \underbrace{\begin{bmatrix} x \\ r \end{bmatrix}}_{\tilde{x}}, \quad (10)$$

where  $E \in \mathbb{R}^{n \times k}$  is a coding matrix, then  $x^* = x + Er$  and  $\lambda^* = \lambda$ .

## Equivalence of Eigensystems

We can write the generalized eigenvalue system as:

$$\begin{bmatrix} A_{11} & A_{12} & Z_1 \\ A_{12}^T & A_{22} & Z_2 \\ Z_1^T & Z_2^T & R \end{bmatrix} \begin{bmatrix} c \\ f \\ r \end{bmatrix} = \lambda \begin{bmatrix} B_{11} & B_{12} & Q_1 \\ B_{12}^T & B_{22} & Q_2 \\ Q_1^T & Q_2^T & S \end{bmatrix} \begin{bmatrix} c \\ f \\ r \end{bmatrix} \quad (11)$$

where

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} E, \quad \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} \\ B_{12}^T & B_{22} \end{bmatrix} E, \quad \text{and} \\ R = E^T A E, \quad S = E^T B E.$$

## Equivalence of Eigensystems

The generalized eigenvalue system will become the following purified  $n \times n$  system when faults happen:

$$\begin{bmatrix} A_{11} & Z_1 \\ Z_1^T & R \end{bmatrix} \begin{bmatrix} c \\ r \end{bmatrix} = \lambda \begin{bmatrix} B_{11} & Y_1 \\ Y_1^T & S \end{bmatrix} \begin{bmatrix} c \\ r \end{bmatrix} - \begin{bmatrix} \lambda B_{12} - A_{12} \\ \lambda Y_2^T - Z_2^T \end{bmatrix} f$$

### theorem

If  $[c; r]$  is the solution of the purified system, then  $[c; f; r]$  is the solution of the generalized eigenvalue system.

## Equivalence of Eigensystems

The purified system gives us

$$A_{11}c + Z_1r - \lambda B_{11}c - \lambda Q_1r = \lambda B_{12}f - \lambda A_{12}f \quad (12)$$

$$Z_1^T c + Rr - \lambda Q_1^T - \lambda Sr = \lambda Q_2^T f - \lambda Z_2^T f \quad (13)$$

Equation (13)  $-E_1^T \times$  Equation (12) yields

$$E_2^T A_{12}c + E_2^T Z_2r - \lambda E_2^T B_{12}c - \lambda E_2^T Q_2r = E_2^T (\lambda B_{22} - A_{22})f \quad (14)$$

Premultiplying Equation (14) by  $E_2^{-T}$  gives

$$A_{12}c + A_{22}f + Z_2r = \lambda B_{12}c + \lambda B_{22}f + \lambda Q_2r$$

which is the second equation in the Equation (11).

## Perturbation

As  $\tilde{\mathbf{A}}$  and  $\tilde{\mathbf{B}}$  are SPSD, the potential eigenvectors may fall into their null space. We add perturbation to the augmented systems to avoid the problem.

$$\tilde{\mathbf{A}}_p = \begin{bmatrix} \mathbf{A} & \mathbf{A}\mathbf{E} \\ \mathbf{E}^T\mathbf{A} & \epsilon\mathbf{I}_k + \mathbf{E}^T\mathbf{A}\mathbf{E} \end{bmatrix}, \quad \tilde{\mathbf{B}}_p = \begin{bmatrix} \mathbf{I} & \mathbf{E} \\ \mathbf{E}^T & \epsilon\mathbf{I}_k + \mathbf{E}^T\mathbf{E} \end{bmatrix}$$

The perturbation is added to the lower-right  $k \times k$  block ( $\epsilon = 10^{-6}$  used here).

- $\tilde{\mathbf{A}}_p$  and  $\tilde{\mathbf{B}}_p$  are *SPD* and TraceMin can be used to solve the generalized eigenvalue problem.
- Purification (*once only*) will be done once the trace is small enough.

## Purification

For the system  $\tilde{\mathbf{A}}_p \tilde{\mathbf{x}} = \lambda \tilde{\mathbf{B}}_p \tilde{\mathbf{x}}$ , we obtain the approx  $(\mu, \mathbf{u})$  (approximate true eigenpairs  $\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$ ) after TraceMin iterations. Hence, we have:

$$\mathbf{A}(\mathbf{u} + \delta \mathbf{u}) = (\mu + \delta \mu)(\mathbf{u} + \delta \mathbf{u}) \quad (15)$$

Also,

$$\mathbf{u}^T \delta \mu = 0 \quad (16)$$

Combining Equation (15) and Equation (16), we will get the linear system:

$$\begin{bmatrix} \mathbf{A} - \mu \mathbf{I}_n & -\mathbf{u} \\ -\mathbf{u}^T & 0 \end{bmatrix} \begin{bmatrix} \delta \mathbf{u} \\ \delta \mu \end{bmatrix} = \begin{bmatrix} -(\mathbf{A}\mathbf{u} - \mu \mathbf{u}) \\ 0 \end{bmatrix} \quad (17)$$

Based on  $\delta \mathbf{u}$  and  $\delta \mu$ , we can update the approximation of the eigenpairs and continue the TraceMin procedure.

# Fault Oblivious TraceMin

---

## Algorithm Fault Oblivious Trace Minimization

---

- 1: Choose an  $n \times s$  random matrix  $V_1$  of full rank such that  $V_1^T \tilde{B}_p V_1 = I(s = 2p)$ .
- 2: **for**  $t = 0, 1, \dots$  until convergence **do**
- 3:     Compute  $W_t = \tilde{A}_p V_t$  and the interaction matrix  $H_t = V_t^T W_t$
- 4:     Compute the eigenpairs of  $(Y_t, \Theta_t)$  for  $H_t$ .
- 5:     Do the purification if the purification condition is satisfied.
- 6:     Sort the eigenvalue in ascending order and rearrange eigenvectors.
- 7:     Compute the corresponding Ritz Vectors  $X_t = V_t Y_t$ .
- 8:     Compute the residue  $R_t = \tilde{A}_p X_t - \tilde{B}_p X_t \Theta_t$ .
- 9:     Test for Convergence.
- 10:    Solve the following linear system approximately via the CG to get  $\Delta_t$ .

$$\begin{bmatrix} \tilde{A}_p & \tilde{B}_p X_t \\ X_t^T \tilde{B}_p & 0 \end{bmatrix} \begin{bmatrix} \Delta_t \\ L_t \end{bmatrix} = \begin{bmatrix} \tilde{A}_p X_t \\ 0 \end{bmatrix}$$

- 11:     $\tilde{B}_p$ -orthonormalize  $X_t - \Delta_t$  into  $V_{t+1}$ .
  - 12: **return**  $\Theta$ .
-

# Implementation

The operations affected by faults in a distributed environment are the aggregation operations – the matrix-matrix and matrix-vector multiplication.

- The Matrix-Matrix Operation:

$$(\tilde{\mathbf{A}}_p, \tilde{\mathbf{V}}_t) = \left( (\tilde{\mathbf{A}}_p)_{[n+k] \setminus F_t}, (\tilde{\mathbf{V}}_t)_{[n+k] \setminus F_t} \right)$$

$$(\tilde{\mathbf{A}}_p, \tilde{\mathbf{X}}_t) = \left( (\tilde{\mathbf{A}}_p)_{[n+k] \setminus F_t}, (\tilde{\mathbf{V}}_t)_{[n+k] \setminus F_t} \right)$$

$$(\tilde{\mathbf{B}}_p, \tilde{\mathbf{X}}_t) = \left( (\tilde{\mathbf{B}}_p)_{[n+k] \setminus F_t}, (\tilde{\mathbf{X}}_t)_{[n+k] \setminus F_t} \right)$$

- The Matrix-Vector Operation in Step 10:

$$(\tilde{\mathbf{A}}_p, \Delta_t) = \left( (\tilde{\mathbf{A}}_p)_{[n+k] \setminus F_t}, (\Delta_t)_{[n+k] \setminus F_t} \right)$$



## Experimental Data

We select two matrices from the University of Florida Matrix Collection.

minsurfo	40,806	203,622	Optimization Problem
s3dkq4m2	90,449	4,427,725	Structural Problem

## Experiment Setup

- For validation of convergence, we monitor  $\frac{\|r_1\|_2}{\lambda_1}$ . The stopping criteria is set as  $10^{-08}$  for all the matrices.
- Define  $t_i = \frac{aug_{\lambda_i} - orig_{\lambda_i}}{orig_{\lambda_i}}$  and construct  $t = [t_1, t_2, \dots, t_{10}]$ . Then final relative error  $rtol = \|t\|_2$ .
- Augmented blocks with different sizes are added to the original system.  $K = 0$  corresponds to the original system;  $K = d$  corresponds to an augmented block size of  $d$ , and  $d$  faults happen during the execution ( $d = 1, 8, 16$ ).

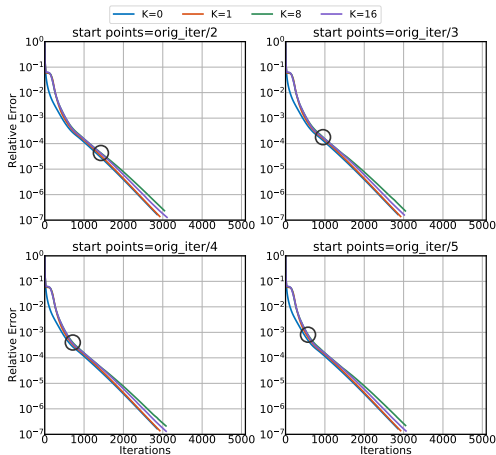
## Experiment Setup

- Leverage score is used to sample a large matrix and can measure the importance of each row of  $A$ . For  $A = U\Sigma V$  (SVD decomposition), the leverage score for each row is calculated as follows:

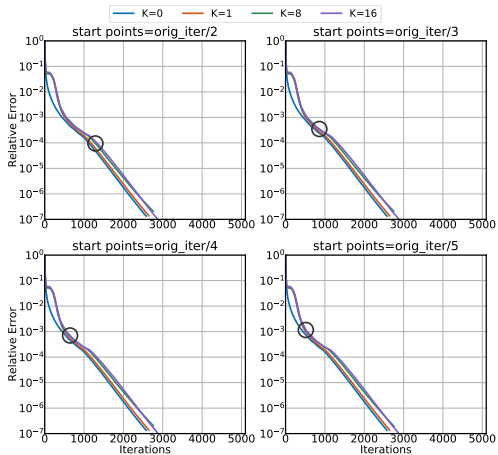
$$l(i) = \sum_{j=1}^n U(i,j)^2$$

- Two different fault arrival models, instantaneous and exponential, were tested.
- For exponential fault model, failure rates ranging from  $\frac{1}{\text{orig\_iter}}$  to  $\frac{4}{\text{orig\_iter}}$  were tested.

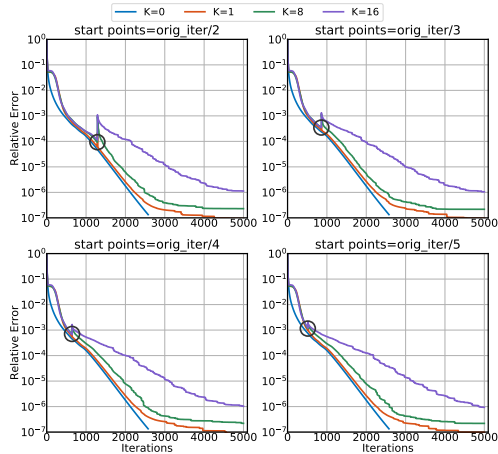
# Convergence for Random Case (minsurfo)



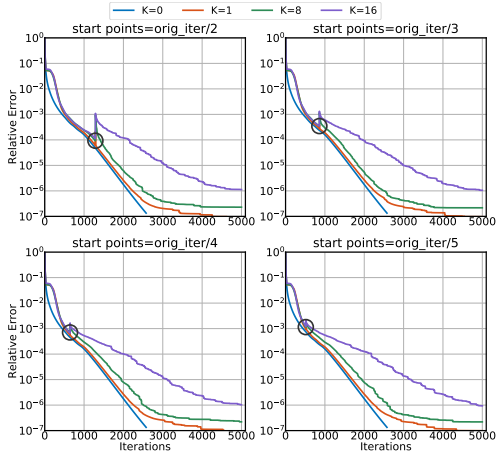
# Convergence for Random Case (s3dkq4m2)



# Convergence for Worst Case (minsurfo)



# Convergence for Worst Case (s3dkq4m2)



## Adaptive Coding Scheme

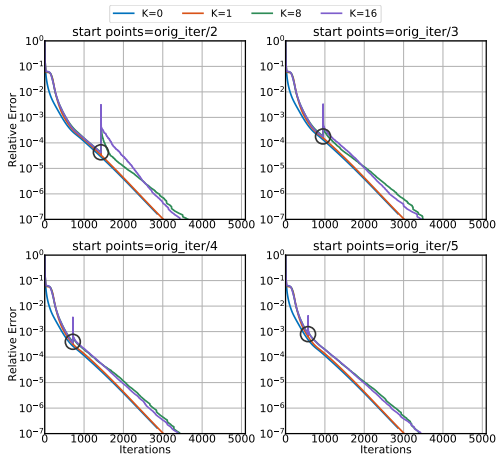
Coding blocks are periodically updated using estimates of leverage scores from prior iteration. The coding matrix  $\mathbf{E}$  is adaptively updated as follows:

$$E(i, :) = E(i, :) * \frac{l(i)}{\bar{l}}$$

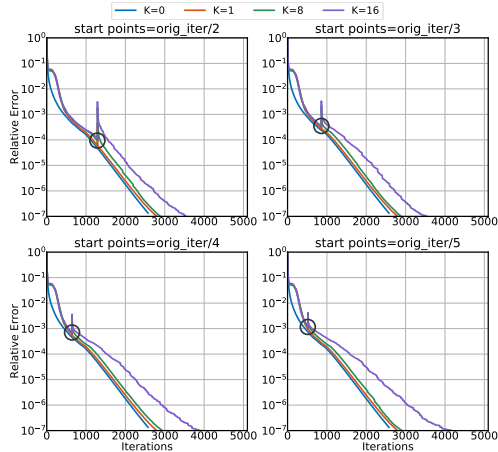
Here  $E(i, :)$  is the  $i^{th}$  row of coding matrix  $E$ ,  $l(i)$  is the leverage score of  $i^{th}$  row and  $\bar{l}$  is the average leverage score of all rows.



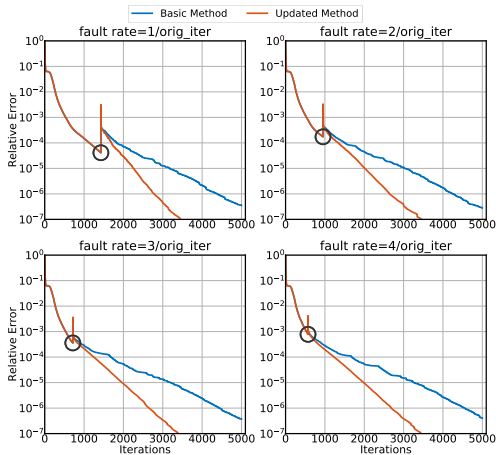
# Convergence of Updating Method (minsurfo)



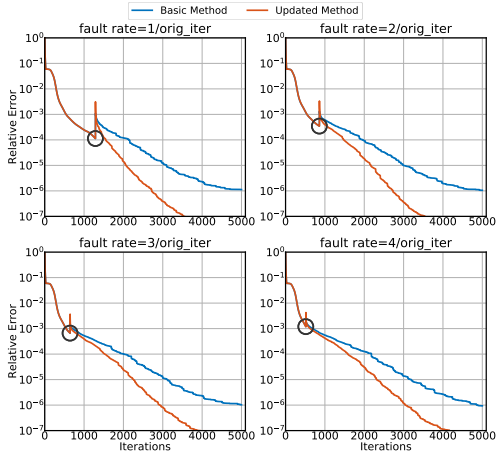
# Convergence of Updating Method (s3dkq4m2)



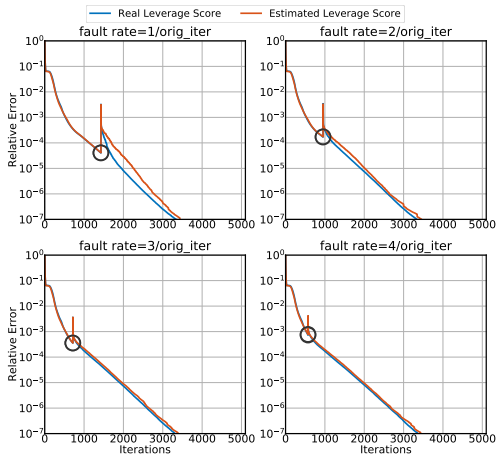
# Importance Results (minsurfo)



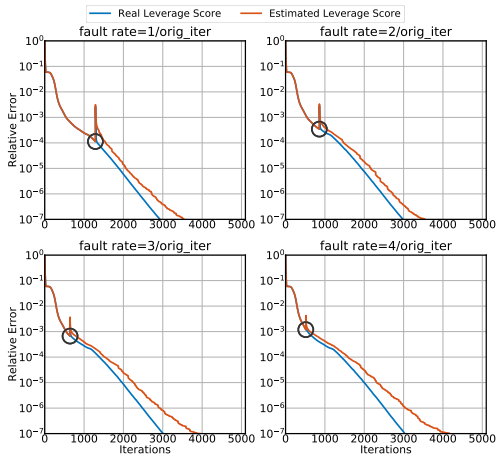
# Importance Results (s3dkq4m2)



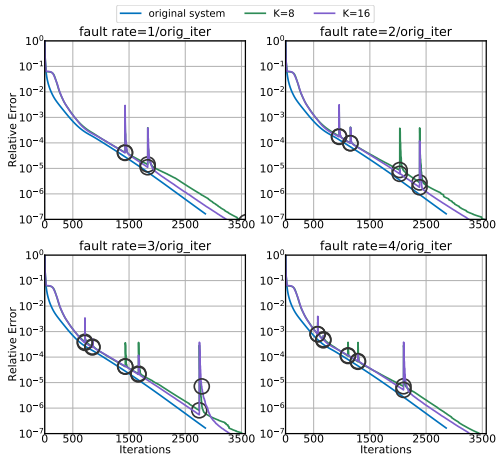
# Comparison of Results (minsurfo)



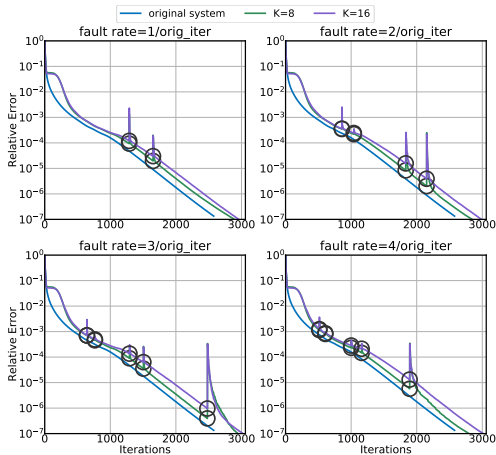
## Comparison of Results (s3dkq4m2)



# Different Fault Arrival Models (minsurfo)



# Different Fault Arrival Models (s3dkq4m2)





# Conclusion

We have developed an **Erasure Coded Fault Tolerant Eigensolver**, which can compute real eigenvalues in faulty environments. We show that our methods:

- Converge to real eigenvalues in both random and worst cases;
- Utilize estimated leverage score to further improve the convergence for worst case;
- Are robust to different fault arrival models including instantaneous and exponential fault model;

Background  
and Motivation

1

Adaptive Fault Tolerant  
Linear System Solver

3

2

Distributed Fault Tolerant  
Linear System Solver

4

Erasure Coded  
Eigensolver

Plan of Research

5

# Plan of Research

- Developing sparse coding schemes for matrix computation.
- Developing distributed fault tolerant linear solver.
- Developing adaptive fault tolerant linear solver.
- Developing erasure coded eigensolver.
- Developing a distributed fault tolerant eigensolver.
- Developing erasure coded matrix computations for other linear algebraic methods.

Thank You.