Algorithms for coping with silent errors

Anne Benoit[1], Aurélien Cavelan[1], Yves Robert[1,2] and Hongyang Sun[1]

1. ENS Lyon
2. University of Tennessee Knoxville

yves.robert@inria.fr

ICL Lunch – October 24, 2014

Definitions

- Instantaneous error detection ⇒ fail-stop failures, e.g. resource crash
- Silent errors (data corruption) ⇒ detection latency

Silent error detected only when corrupt data is activated and modifies application behavior

- Includes some software faults, some hardware errors (soft errors in L1 cache, ALU), double bit flip
- Cannot always be corrected by ECC memory

Probability distributions for silent errors

?

Theorem: $\mu_p = \frac{\mu_{\text{ind}}}{p}$ for arbitrary distributions

(a.k.a, scale is the enemy)

Probability distributions for silent errors

?

Theorem: $\mu_p = \frac{\mu_{\text{ind}}}{p}$ for arbitrary distributions

(a.k.a, scale is the enemy)

Lesson learnt for fail-stop failures

(Not so) Secret data

- Tsubame 2: 962 failures during last 18 months so $\mu = 13$ hrs
- Blue Waters: 2-3 node failures per day
- Titan: a few failures per day
- Tianhe 2: wouldn't say

$$T_{\text{opt}} = \sqrt{2\mu C} \quad \Rightarrow \quad \textsc{Waste}_{\text{opt}} \approx \sqrt{\frac{2C}{\mu}}$$

Petascale:	$C = 20$ min	$\mu = 24$ hrs	$\Rightarrow \textsc{Waste}_{\text{opt}} = 17\%$
Scale by 10:	$C = 20$ min	$\mu = 2.4$ hrs	$\Rightarrow \textsc{Waste}_{\text{opt}} = 53\%$
Scale by 100:	$C = 20$ min	$\mu = 0.24$ hrs	$\Rightarrow \textsc{Waste}_{\text{opt}} = 100\%$

Lesson learnt for fail-stop failures

(... so) Secret data

- Tsuba... 962 failures during last 18 months so ... 13 hrs
- Blue Waters: 2-... de failures per day
- Titan: a few failures pe...
- Tianhe...

Exascale $\neq$ Petascale $\times 1000$

Need more reliable components

Need to checkpoint faster

Petascale:	$C = 20$ min	$\mu = 24$ hrs	$\Rightarrow$ WASTE$_{opt}$ = 17%
Scale by 10:	$C = 20$ min	$\mu = 2.4$ hrs	$\Rightarrow$ WASTE$_{opt}$ = 53%
Scale by 100:	$C = 20$ min	$\mu = 0.24$ hrs	$\Rightarrow$ WASTE$_{opt}$ = 100%

Lesson learnt for fail-stop failures

(Not so) Secret data

- Tsubame 2: 962 failures during last 18 months so $\mu = 13$ hrs
- Blue Waters: 2-3 node failures per day
- Titan: a few failures per day
- Tianhe 2: wouldn't say

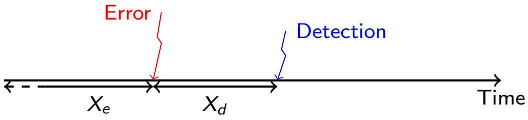
Silent errors:
detection latency $\Rightarrow$ additional problems

Petascale:	$C = 20$ min	$\mu = 24$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 17\%$
Scale by 10:	$C = 20$ min	$\mu = 2.4$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 53\%$
Scale by 100:	$C = 20$ min	$\mu = 0.24$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 100\%$

Outline

Outline

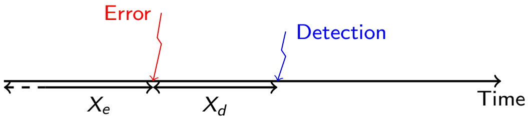
General-purpose approach



Error and detection latency

- Last checkpoint may have saved an already corrupted state
- Saving k checkpoints (Lu, Zheng and Chien):
  1. Critical failure when all live checkpoints are invalid
  2. Which checkpoint to roll back to?

General-purpose approach



Error and detection latency

- Last checkpoint may have saved an already corrupted state
- Saving k checkpoints (Lu, Zheng and Chien):
  1. Critical failure when all live checkpoints are invalid
     Assume unlimited storage resources
  2. Which checkpoint to roll back to?
     Need a verification mechanism ☹ ☹ ☹

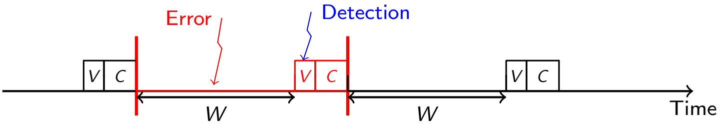
Outline

1. General-purpose approach
2. Checkpointing and verification
   - Arbitrary probability distribution
   - Exponential probability distributions
3. Simulations
   - SINGLESPEED scenario for makespan
   - SINGLESPEED scenario for energy
   - REEXECSPEED and MULTISPEED scenarios

Coupling checkpointing and verification

- Verification mechanism of cost V
- Silent errors detected only when verification is executed
- Approach agnostic of the nature of verification mechanism (checksum, error correcting code, coherence tests, triple modular redundancy, etc)
- Fully general-purpose
  (application-specific information, if available, can always be used to decrease V)

Base pattern (and revisiting Young/Daly)



	Fail-stop (classical)	Silent errors
Pattern	$T = W + C$	$S = W + V + C$
WASTE_{FF}	$\frac{C}{T}$	$\frac{V+C}{S}$
$\text{WASTE}_{\text{fail}}$	$\frac{1}{\mu}(D + R + \frac{W}{2})$	$\frac{1}{\mu}(R + W + V)$
Optimal	$T_{\text{opt}} = \sqrt{2C\mu}$	$S_{\text{opt}} = \sqrt{(C+V)\mu}$
$\text{WASTE}_{\text{opt}}$	$\sqrt{\frac{2C}{\mu}}$	$2\sqrt{\frac{C+V}{\mu}}$

On-line ABFT scheme for PCG

Compute $r^{(0)} = b - Ax^{(0)}, z^{(0)} = M^{-1}r^{(0)}, p^{(0)} = z^{(0)}$,
  and $\rho_0 = {r^{(0)}}^T z^{(0)}$ for some initial guess $x^{(0)}$
checkpoint: A, M, and b
for $i = 0, 1, \ldots$
  if ((i>0) and ($i\%d = 0$))
    if ($\frac{{p^{(i+1)}}^T q^{(i)}}{||p^{(i+1)}||.||q^{(i)}||} > 10^{-10}$
     or $\frac{||r^{(i+1)} + Ax^{(i+1)} - b||}{||b||.||A||} > 10^{-10}$)
      recover: A, M, b, i, ρ_i,
        $p^{(i)}$, $x^{(i)}$, and $r^{(i)}$.
    else if ($i\%(cd) = 0$)
      checkpoint: i, ρ_i, $p^{(i)}$, and $x^{(i)}$
    endif
  endif
  $q^{(i)} = Ap^{(i)}$
  $\alpha_i = \rho_i / {p^{(i)}}^T q^{(i)}$
  $x^{(i+1)} = x^{(i)} + \alpha_i p^{(i)}$
  $r^{(i+1)} = r^{(i)} - \alpha_i q^{(i)}$
  solve $Mz^{(i+1)} = r^{(i+1)}$, where $M = M^T$
  $\rho_{i+1} = {r^{(i+1)}}^T z^{(i+1)}$
  $\beta_i = \rho_{i+1} / \rho_i$
  $p^{(i+1)} = z^{(i+1)} + \beta_i p^{(i)}$
  check convergence; continue if necessary
end

Zizhong Chen, PPoPP'13

- Iterate PCG
  Cost: SpMV, preconditioner solve, 5 linear kernels
- Detect soft errors by checking orthogonality and residual
- Verification every d iterations
  Cost: scalar product+SpMV
- Checkpoint every c iterations
  Cost: three vectors, or two vectors + SpMV at recovery
- Experimental method to choose c and d

Outline

1. General-purpose approach
2. Checkpointing and verification
    - Arbitrary probability distribution
    - Exponential probability distributions
3. Simulations
    - SingleSpeed scenario for makespan
    - SingleSpeed scenario for energy
    - ReExecSpeed and MultiSpeed scenarios

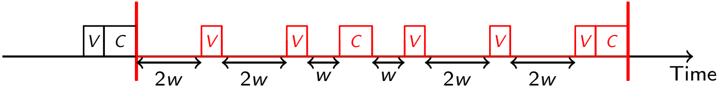
Problems

- Given a pattern with p checkpoints and q verifications:
  where to position them?
  what is the optimal period (pattern length)?

  PCG example:
  1 checkpoint, c verifications, length $= cd$ iterations

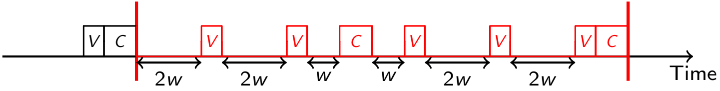
- What is the best pattern?

BalancedAlgorithm



- p checkpoints and q verifications, $p \leq q$
- $p = 2$, $q = 5$, $S = 2C + 5V + W$
- $W = 10w$, six chunks of size w or $2w$
- May store invalid checkpoint (error during third chunk)
- After successful verification in fourth chunk, preceding checkpoint is valid
- Keep only two checkpoints in memory and avoid any fatal failure

BalancedAlgorithm



① (proba $2w/W$) $T_{\text{lost}} = R + 2w + V$

② (proba $2w/W$) $T_{\text{lost}} = R + 4w + 2V$

③ (proba w/W) $T_{\text{lost}} = 2R + 6w + C + 4V$

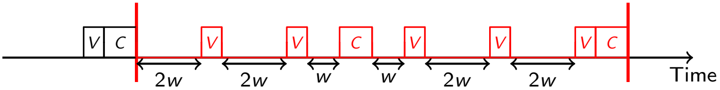
④ (proba w/W) $T_{\text{lost}} = R + w + 2V$

⑤ (proba $2w/W$) $T_{\text{lost}} = R + 3w + 2V$

⑥ (proba $2w/W$) $T_{\text{lost}} = R + 5w + 3V$

$$\textsc{Waste}_{\text{opt}} \approx 2\sqrt{\frac{7(2C+5V)}{20\mu}}$$

BalancedAlgorithm



Theorem

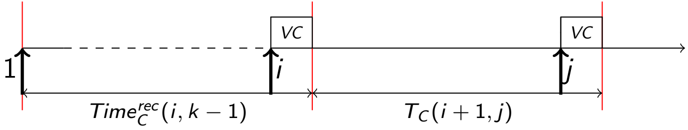
1. *Given pattern with p checkpoints and q verifications,* BalancedAlgorithm *is optimal when MTBF is large in front of resilience parameters C, R, V*
2. *Given pattern, can compute optimal period length*
3. *Given C/V ratio, can compute optimal pattern (best values of p and q)*

Outline

Linear chain

- $\{T_1, T_2, \ldots, T_n\}$: linear chain of n tasks
- Each task T_i fully parametrized:
  - w_i computational weight
  - C_i, R_i, V_i : checkpoint, recovery, verification
- Error rates:
  - λ^F rate of fail-stop errors
  - λ^S rate of silent errors

VC-ONLY

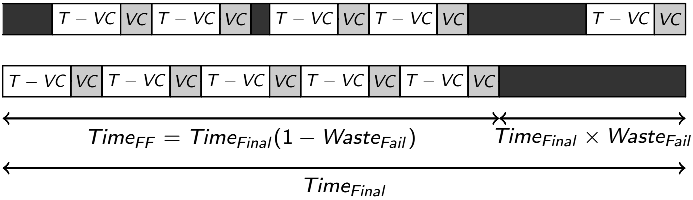


$$\min_{0 \leq k < n} Time_C^{rec}(n, k)$$

$$Time_C^{rec}(j, k) = \min_{k \leq i < j} \{ Time_C^{rec}(i, k-1) + T_C^{SF}(i+1, j) \}$$

$$T_C^{SF}(i,j) = p_{i,j}^F \left(T_{lost_{i,j}} + R_{i-1} + T_C^{SF}(i,j) \right)$$
$$+ \left(1 - p_{i,j}^F\right) \left(\sum_{\ell=i}^{j} w_\ell + V_j + p_{i,j}^S \left(R_{i-1} + T_C^{SF}(i,j) \right) + \left(1 - p_{i,j}^S\right) C_j \right)$$

Young/Daly



$$\text{Waste} = \text{Waste}_{ef} + \text{Waste}_{fail}$$

$$\text{Waste} = \frac{V + C}{T} + \lambda^F(s)\left(R + \frac{T}{2}\right) + \lambda^S(s)(R + T)$$

$$T_{\text{opt}} = \sqrt{\frac{2(V + C)}{\lambda^F(s) + 2\lambda^S(s)}}$$

Extensions

- VC-ONLY and VC+V
- Different speeds with DVFS, different error rates
- Different execution modes
- Optimize for time or for energy consumption

☺ ☺ ☺

- Use verification to correct some errors (ABFT)
- Same analysis (smaller error rate but higher verification cost)

Extensions

- VC-ONLY and VC+V
- Different speeds with DVFS, different error rates
- Different execution modes
- Optimize for time or for energy consumption

☺ ☺ ☺

- Use verification to correct some errors (ABFT)
- Same analysis (smaller error rate but higher verification cost)

Outline

1. General-purpose approach

2. Checkpointing and verification
   - Arbitrary probability distribution
   - Exponential probability distributions

3. Simulations
   - SingleSpeed scenario for makespan
   - SingleSpeed scenario for energy
   - ReExecSpeed and MultiSpeed scenarios

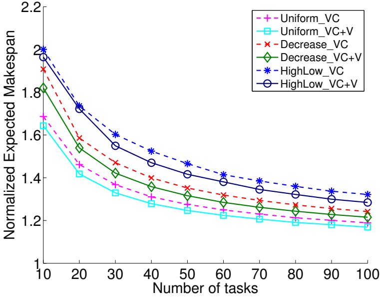
Settings

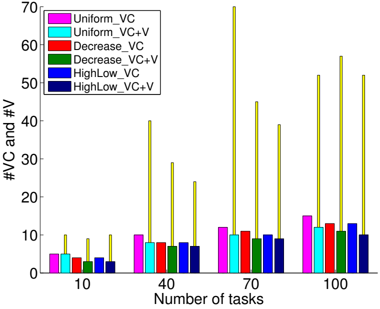
- Linear chains with n tasks
  - Total work $W \approx 14$ hours
  - Patterns: (1) Uniform; (2) Decrease; (3) HighLow
- Set of speeds from Intel Xscale processor
  - Normalized speeds $\{0.15, 0.4, 0.6, 0.8, 1\}$
  - Fitted power function $P(s) = 1550s^3 + 60$
  - $\lambda^F(s) = \lambda^F_{\text{ref}} \cdot 10^{\frac{d \cdot |s_{\text{ref}} - s|}{s_{\max} - s_{\min}}}$
  - Reference speed $s_{\text{ref}} = 0.6$ and $\lambda^F_{\text{ref}} = 10^{-5}$ for fail-stop errors
  - Sensitivity parameter $d = 3$
  - Corresponds to $0.83 \sim 129$ errors over entire chain
  - Silent errors: $\lambda^S(s) = \eta \cdot \lambda^F(s)$
- Checkpoint and verification costs for a task
  - cr ratio of checkpointing cost over computational cost
  - vr ratio of verification cost over computational cost
  - Default: checkpoint cost $\gg$ verification cost

Outline

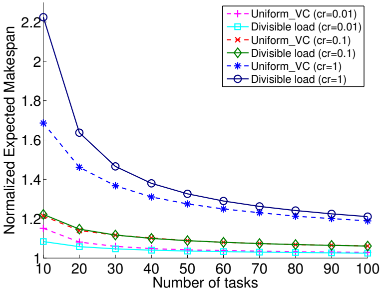
1. General-purpose approach
2. Checkpointing and verification
   - Arbitrary probability distribution
   - Exponential probability distributions
3. Simulations
   - SingleSpeed scenario for makespan
   - SingleSpeed scenario for energy
   - ReExecSpeed and MultiSpeed scenarios

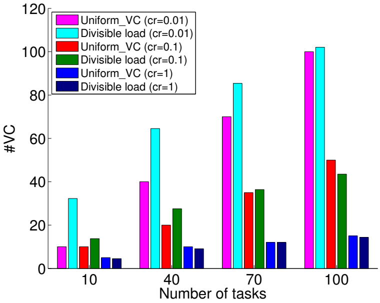
Impact of n and cost distribution



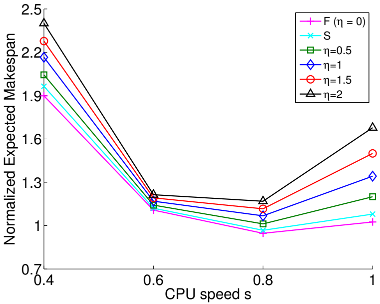


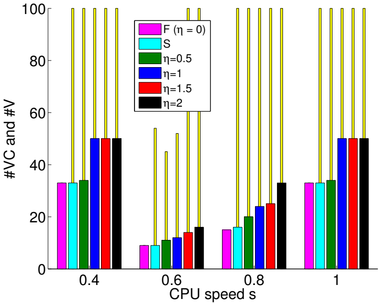
Comparison with a divisible load application



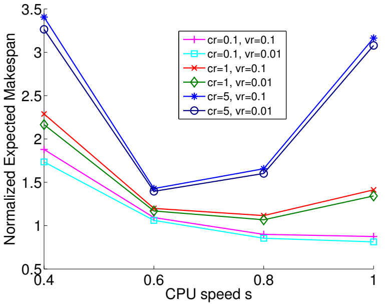


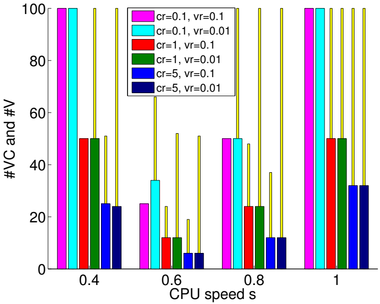
Impact of η (TIME-VC+V, $n = 100$, Uniform)





Impact of *cr* and *vr* (TIME-VC+V, $n = 100$, Uniform)

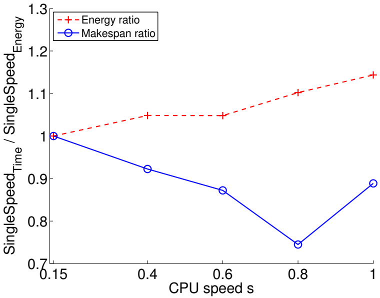


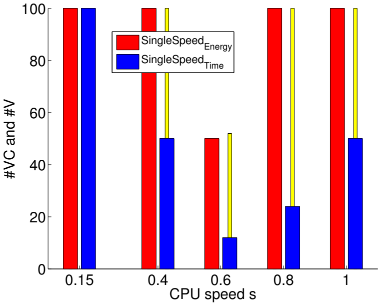


Outline

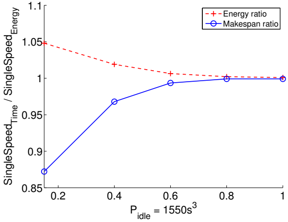
1. General-purpose approach
2. Checkpointing and verification
   - Arbitrary probability distribution
   - Exponential probability distributions
3. Simulations
   - SINGLESPEED scenario for makespan
   - SINGLESPEED scenario for energy
   - REEXECSPEED and MULTISPEED scenarios

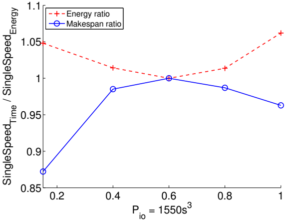
Impact of CPU speed s

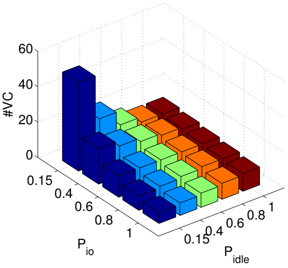




Impact of P_{idle} and P_{io}



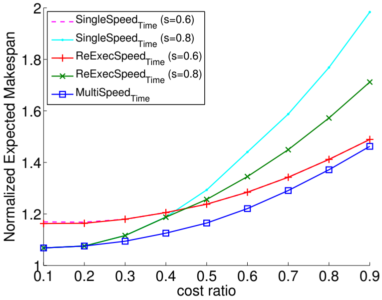


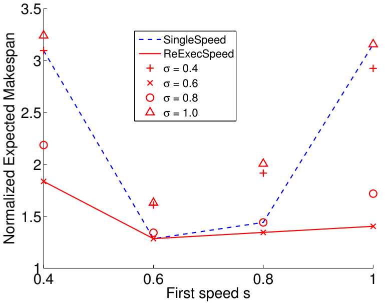


Outline

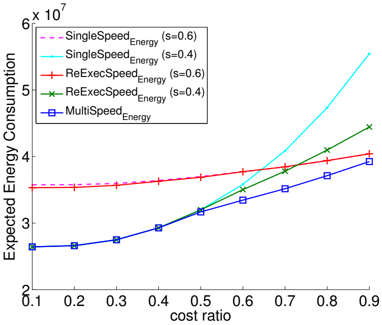
1. General-purpose approach
2. Checkpointing and verification
   - Arbitrary probability distribution
   - Exponential probability distributions
3. Simulations
   - SINGLESPEED scenario for makespan
   - SINGLESPEED scenario for energy
   - REEXECSPEED and MULTISPEED scenarios

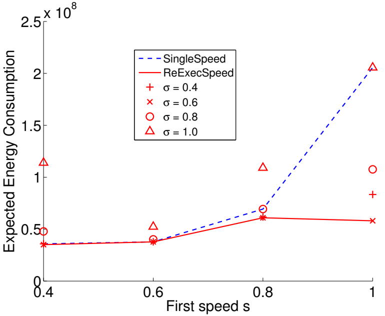
TIME-VC+V (HighLow)





ENERGY-VC+V (HighLow)





Conclusion

- Soft errors difficult to cope with, even for divisible workloads or linear chains
- Investigate general task graphs
- Combine checkpointing, replication and application-specific techniques
- Multi-criteria optimization problem
  execution time/energy/reliability
  best resource usage (performance trade-offs)

Several challenging algorithmic/scheduling problems ☺

Back to task graphs?

Framework

- You're given a (very big) task graph
- Each task produces files that you can save (checkpoint) or not
- Each task can choose from different execution speeds, with different error probabilities
- You can replicate some tasks, either for verification or for faster execution of successor tasks
- You may also be able to verify results by some application-specific mechanism

Problem

- Given energy budget or power cap, minimize execution time
- For each task, many things to be decided by schedule ☹

Back to task graphs?

Framework

- You're given
- Each task pro ... checkpoint) or not
- Each task car ... on speeds, with different error
- You can repli ... ication or for faster executi
- You may also ... ne application-sp

Problem

- Given energy ... execution time
- For each task ... schedule ☹

Back to task graphs?

Fr

ot

Pr

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

Model + Theorem ➜ Paper

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

Model + Theorem $\not\rightarrow$ Paper ☹

Model + Theorem + Matlab code ➜ Paper

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

Model + Theorem $\not\rightarrow$ Paper ☹

Model + Theorem + Matlab code $\not\not\rightarrow$ Paper ☹

Model + Theorem + Matlab code + Simulations $\rightarrow$ Paper

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

Model + Theorem ↛ Paper ☹

Model + Theorem + Matlab code ↛ Paper ☹

Model + Theorem + Matlab code + Simulations ↛ Paper ☹

Model + Theorem + Matlab code + Simulations +
Experiments on large platforms ➜ Paper

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

Model + Theorem $\not\rightarrow$ Paper ☹

Model + Theorem + Matlab code $\not\rightarrow$ Paper ☹

Model + Theorem + Matlab code + Simulations $\not\rightarrow$ Paper ☹

~~Model + Theorem + Matlab code + Simulations~~ +
Experiments on large platforms $\rightarrow$ Paper

A few questions

Silent errors

- Error rate? MTBE?
- Selective reliability?
- New algorithms beyond iterative? matrix-product, FFT, ...
- Resilient research?

Models needed to assess techniques at scale
without bias ☺