RUTGERS

THE STATE UNIVERSITY
OF NEW JERSEY

RDI²

Exploring Failure Recovery for Stencil-based Applications at Extreme Scales

Marc Gamell[1], Keita Teranishi[2], Michael Heroux[2], Jackson Mayo[2], Hemanth Kolla[2], Jacqueline Chen[2], Manish Parashar[1]

[1] Rutgers Discovery Informatics Institute (RDI²), Rutgers University

[2] Sandia National Laboratories

SAND2015-4650 C

How do we recover after a Failure?

- Current FT approach → Coordinated PFS-based *Checkpointing*
  → On failure, stop application and *Restart*

Unfeasible at exascale!

- Online recovery can dramatically reduce failure overhead
- **Global recovery** involves all the cores in the recovery process
  - This can be done in a semi-transparent way, but...
  - **Scalability issues**!
- **Local recovery** can further benefit certain classes of applications

How do we recover after a Failure?

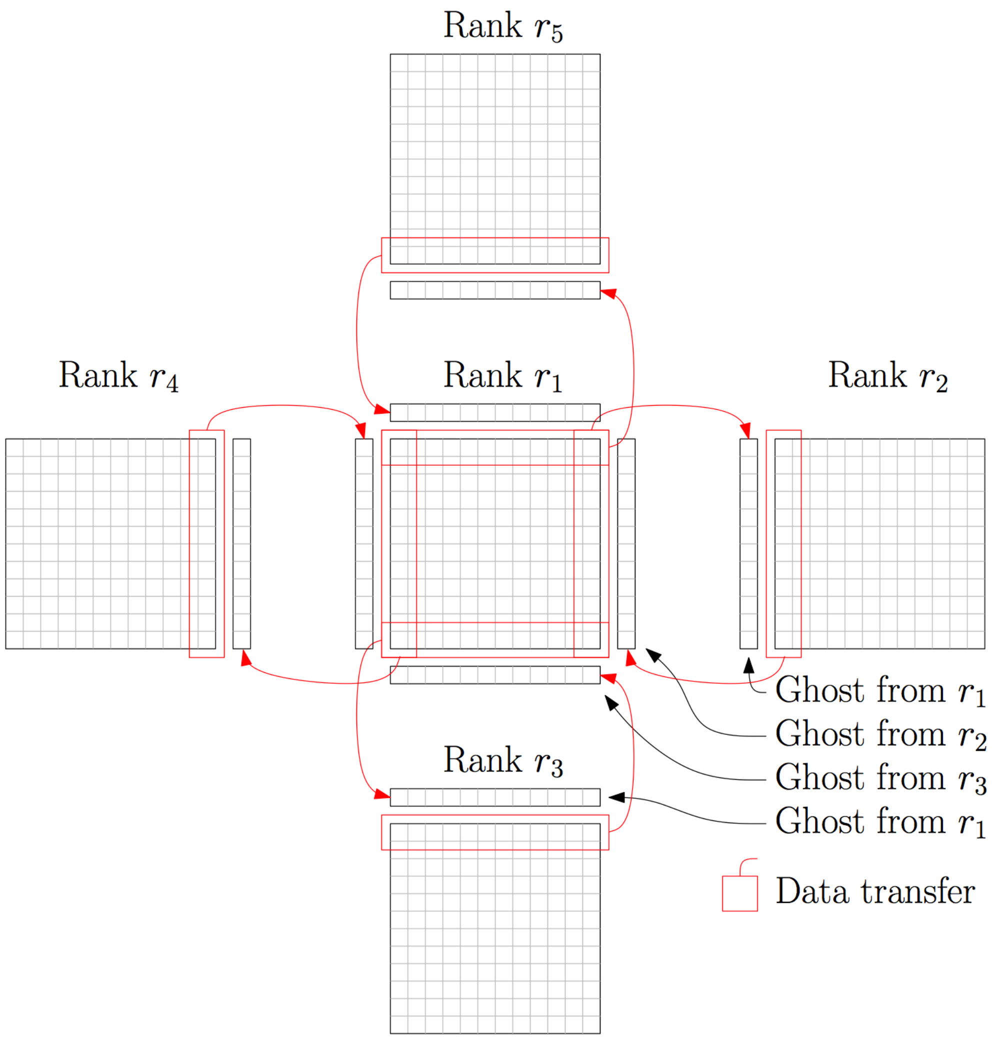
- Current FT approach → Coordinated PFS-based *Checkpointing*
  → On failure, stop application and *Restart*

Unfeasible at exascale!

- Online recovery can dramatically reduce failure overhead
- **Global recovery** involves all the cores in the recovery process
  - This can be done in a semi-transparent way, but...
  - **Scalability issues**!
- **Local recovery** can further benefit certain classes of applications

Goal:
Study the feasibility of local recovery for stencil-based parallel applications

Target: Stencil-based Scientific Applications



- Application domain is partitioned using a block decomposition across processes
- Typically, divided in iterations (*timesteps*), which include:
  - Computation to advance the local simulated data
  - Communication with immediate neighbors
- Example: PDEs using finite-difference methods

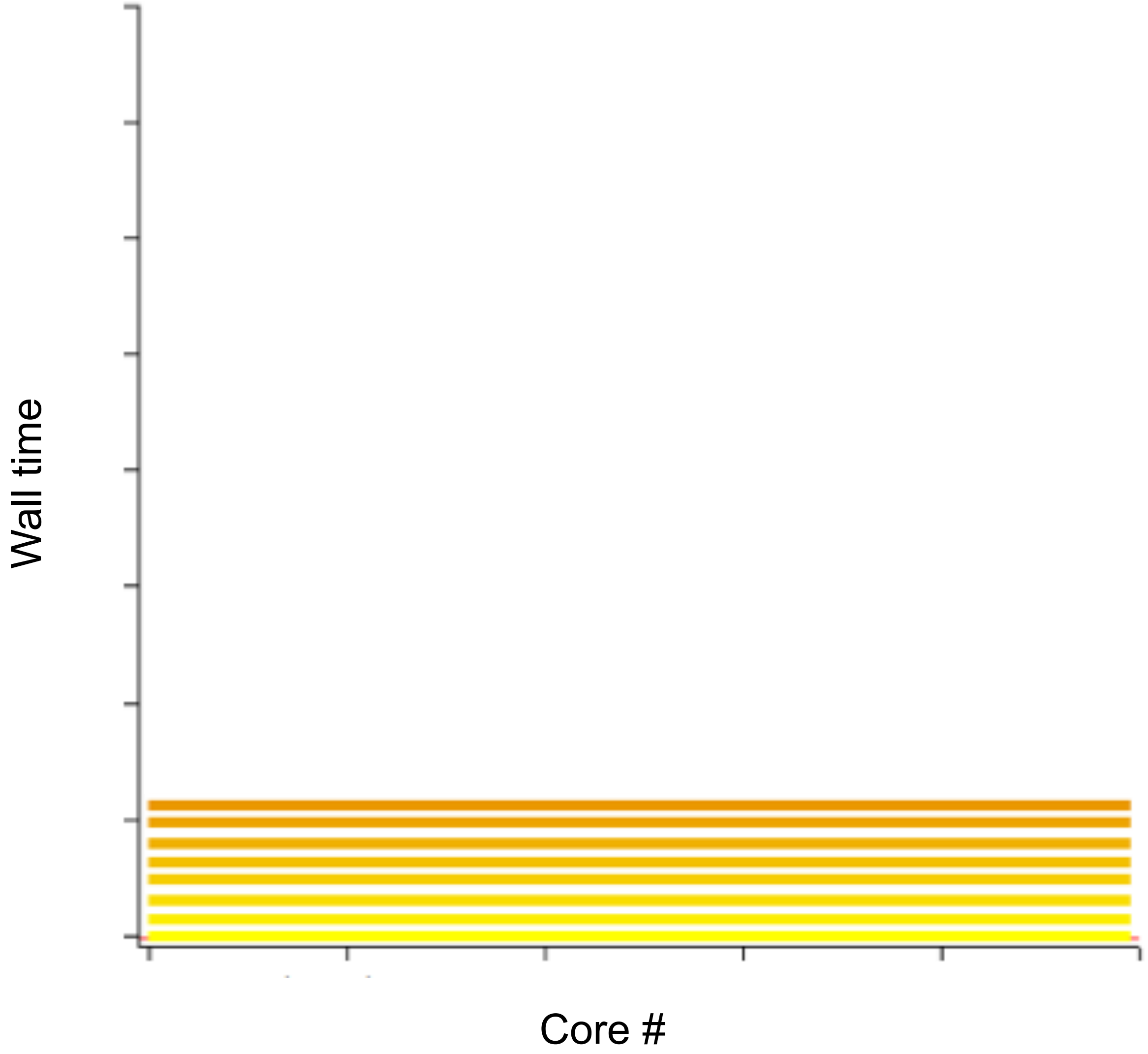
Local Recovery Technique

- How to recover?
  - replace failed processes
  - (recovered processes) rollback to the last checkpoint
- Distant parts of the domain continue the simulation

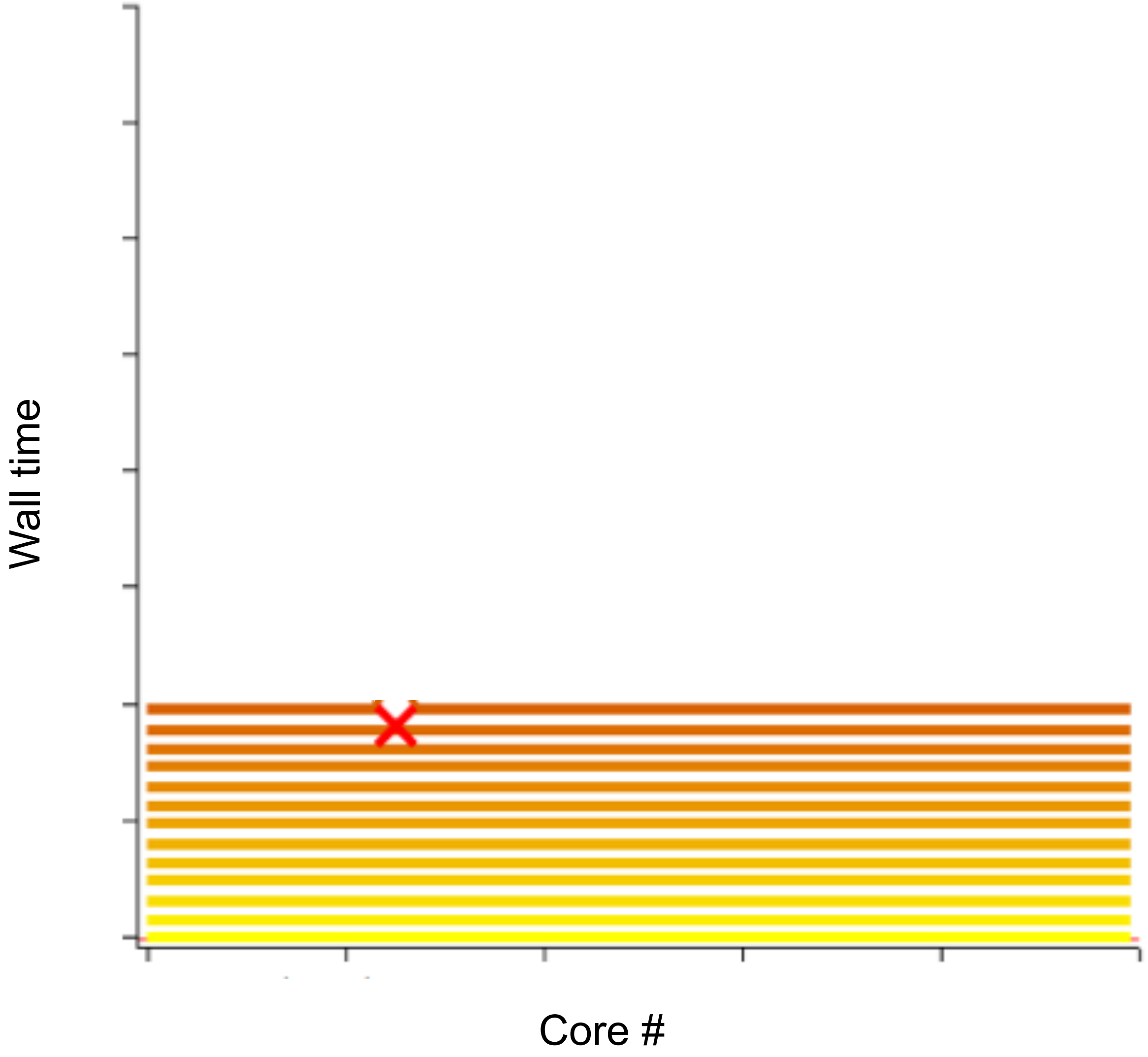
- Failure effect will slowly propagate through the machine
  - Only immediate neighbors will be immediately affected by that failure

- **Perfect scalability**
- **Mask multiple failures**
  - time to solution appear as if only a single failure occurred

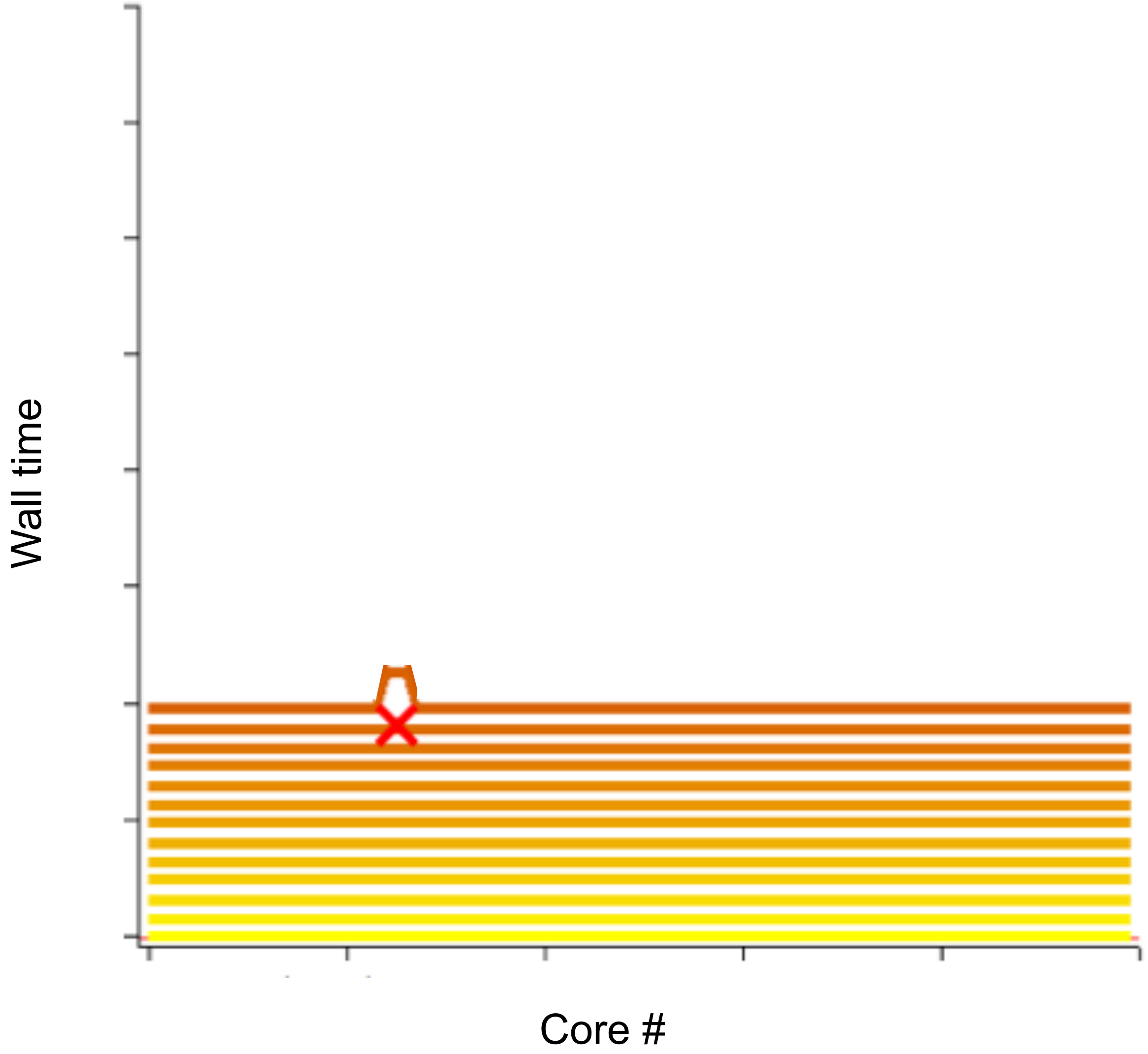
Why delay is slowly propagated?



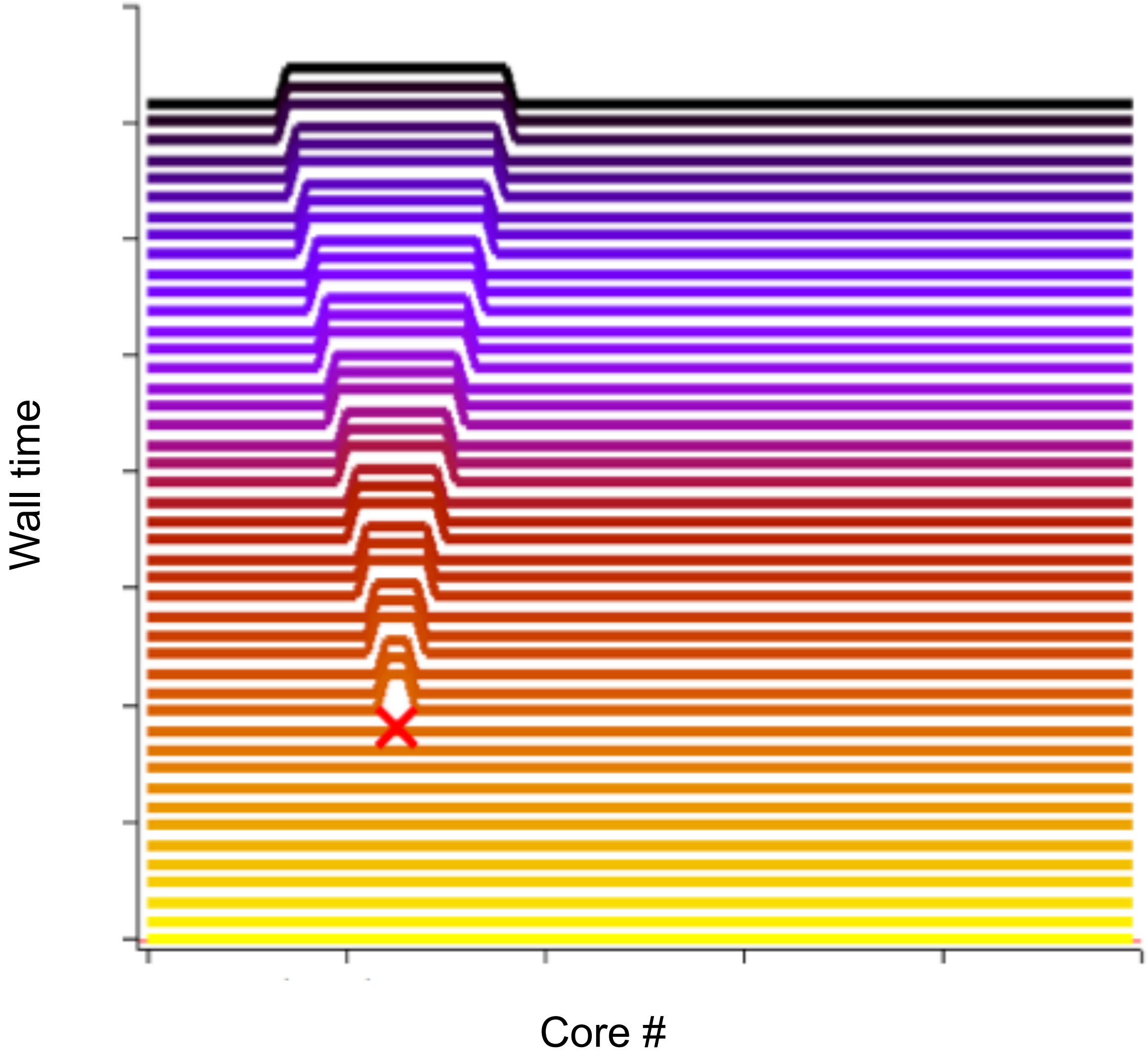
Why delay is slowly propagated?



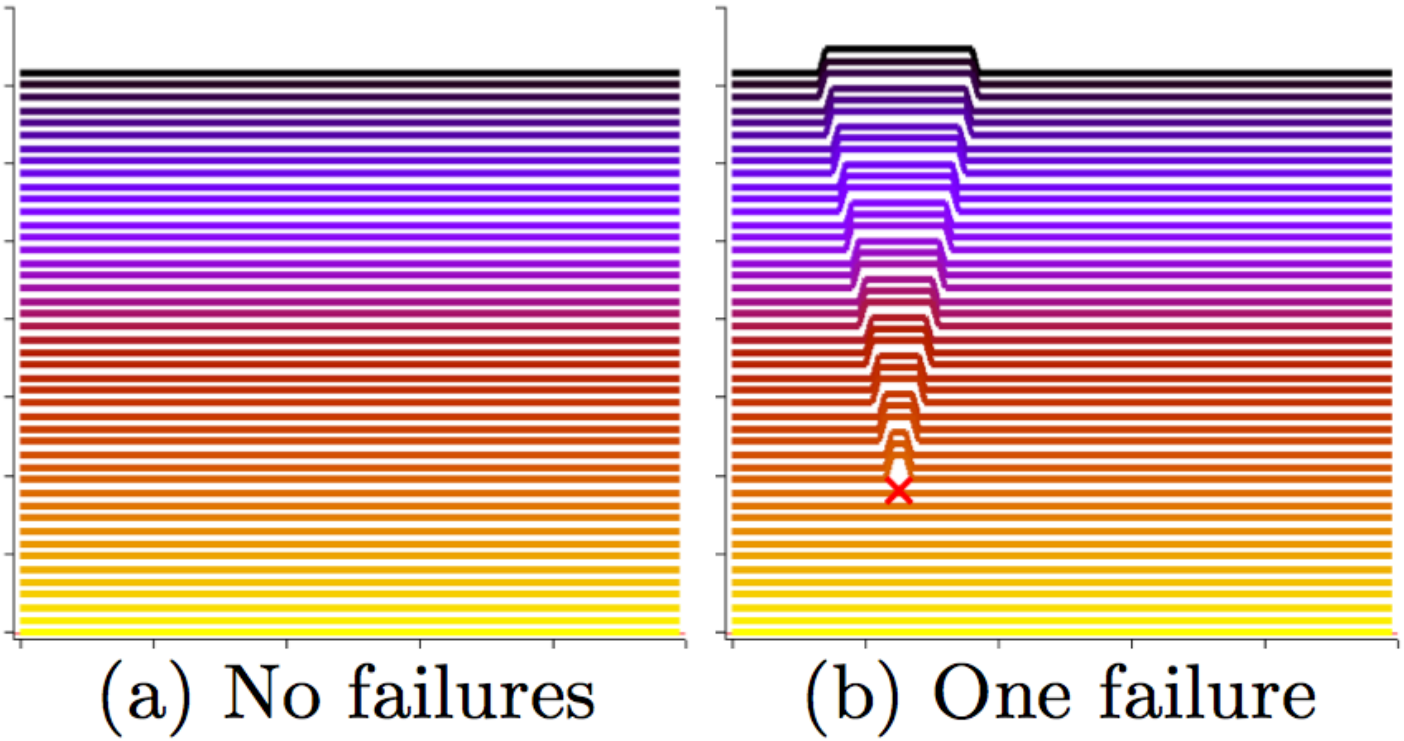
Why delay is slowly propagated?



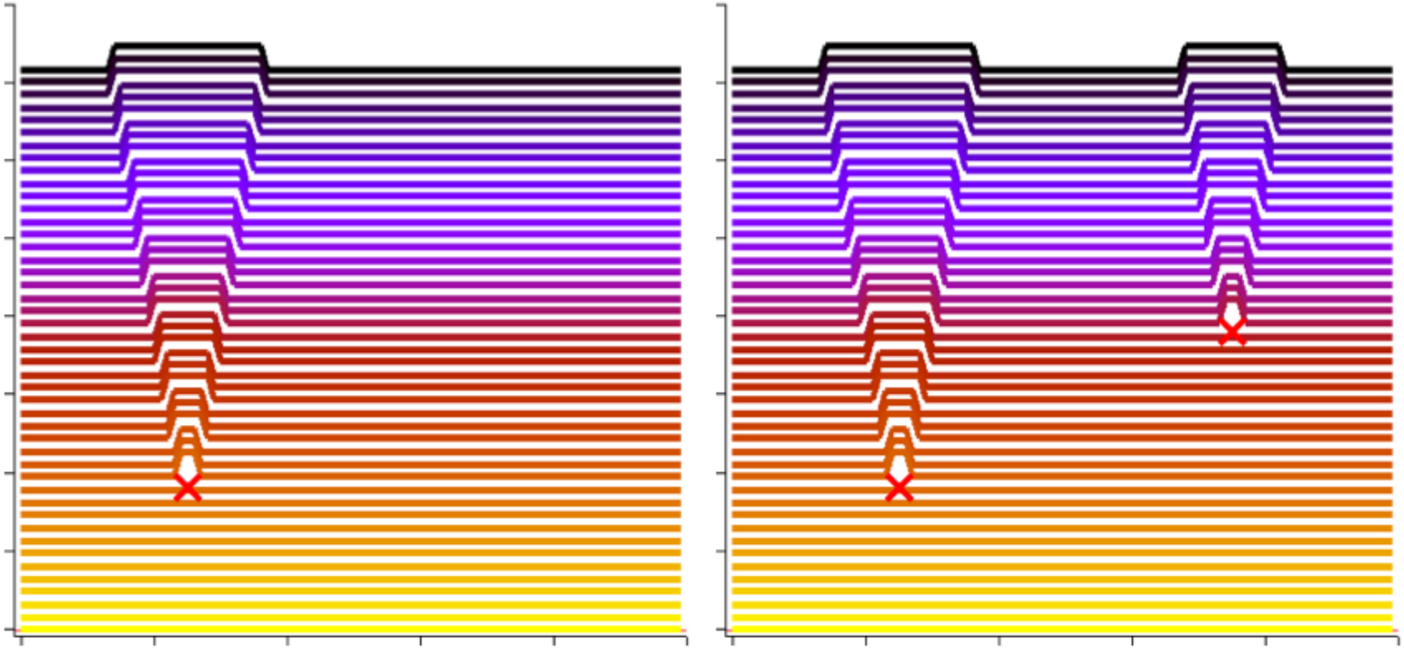
Why delay is slowly propagated?



Masking the effect of multiple failures


(a) No failures
(b) One failure

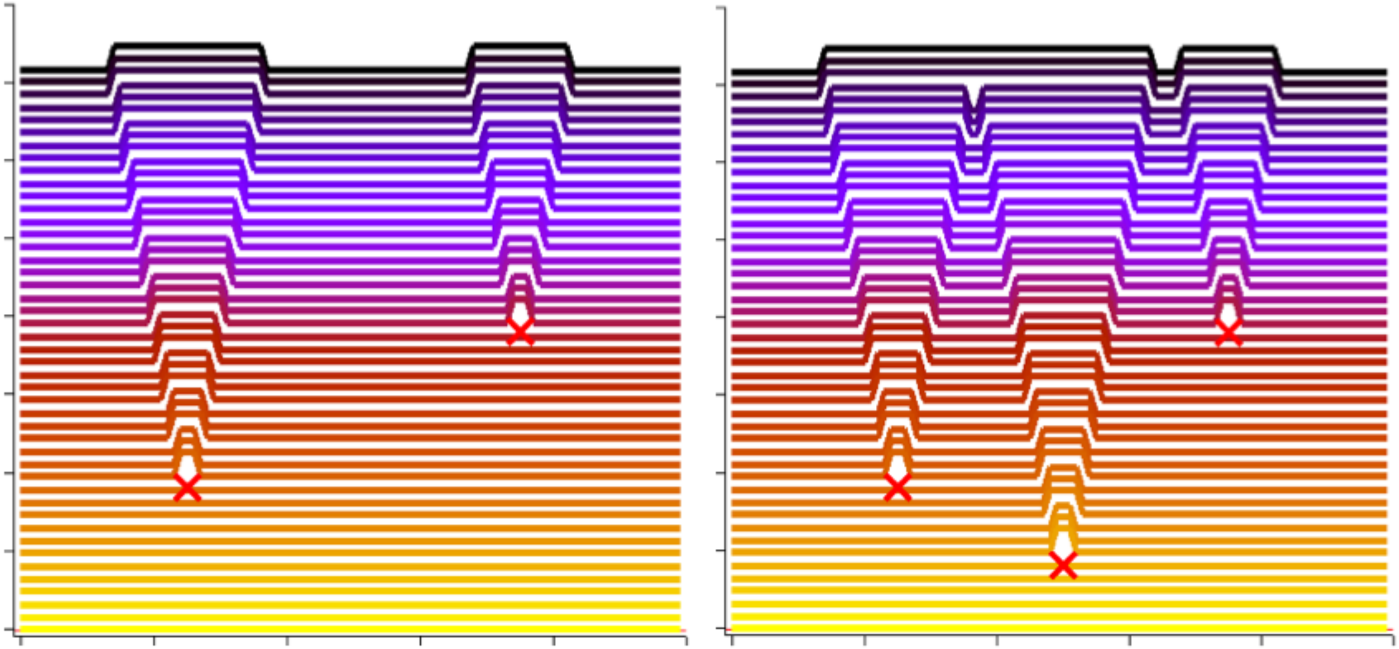
Masking the effect of multiple failures



(b) One failure

(c) Two failures

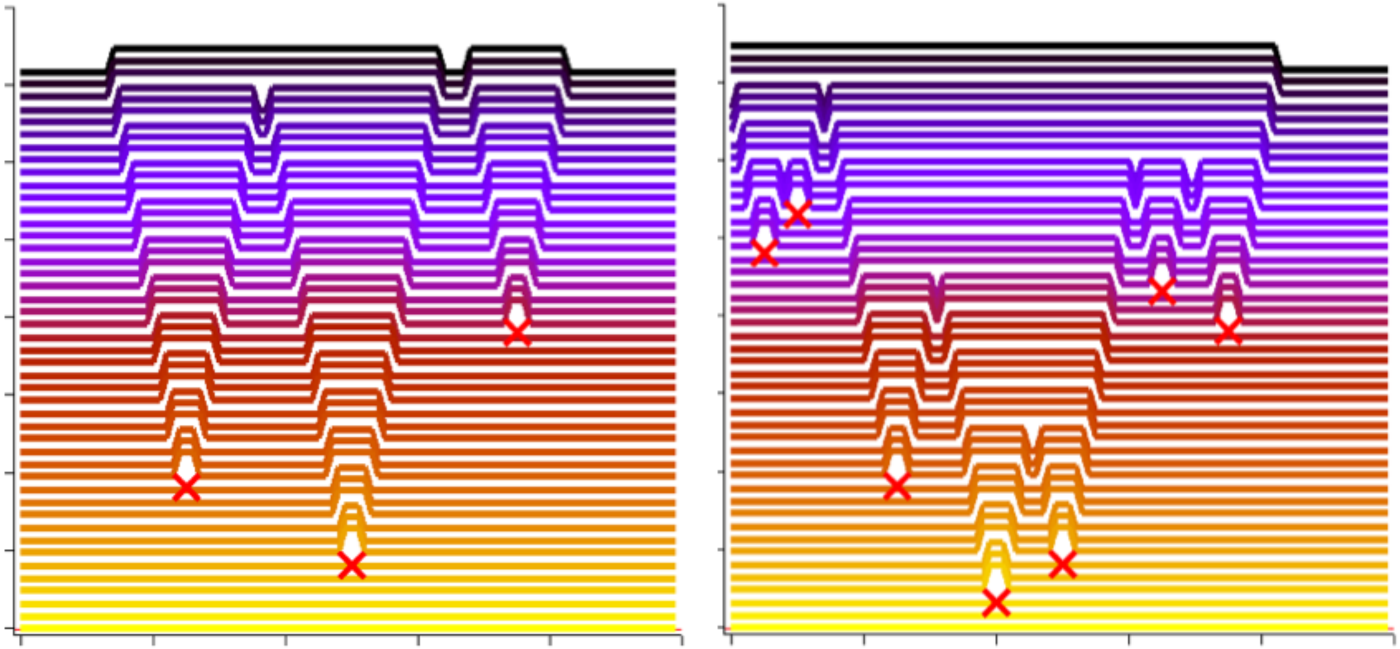
Masking the effect of multiple failures



(c) Two failures

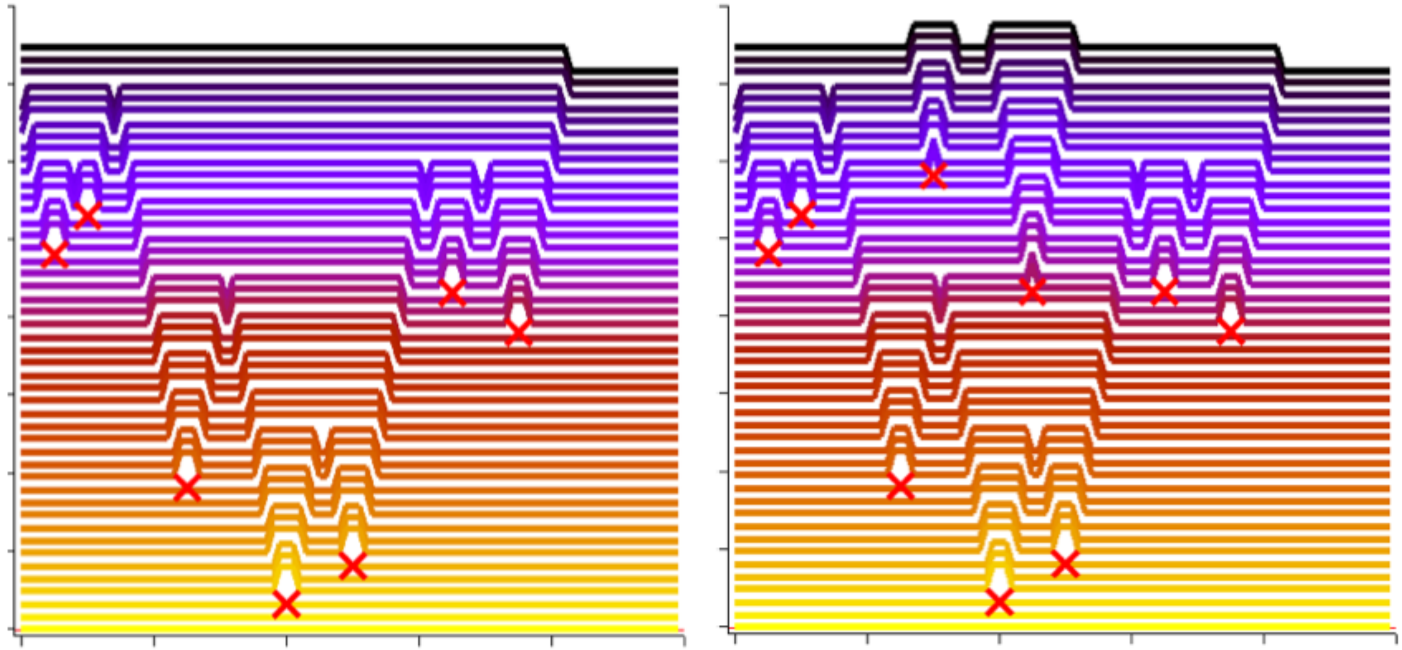
(d) Three failures

Masking the effect of multiple failures



(d) Three failures (e) Seven failures

Masking the effect of multiple failures



(e) Seven failures (f) Nine failures

Conclusion

- Local recovery is beneficial both for the application and the runtime

- Runtime
  - Scalable implementation of recovery constructs
  - No need to coordinate the whole domain in order to recovery
- Application
  - No Global Work Recomputation
  - Lower Energy Footprint
  - Failure Masking
    - it has been shown that failures don’t come alone, but they come in bursts

- We studied certain type of applications only
- How the conclusions apply to other types?

“Exploring Failure Recovery for Stencil-based Applications at Extreme Scales”
Marc Gamell, Keita Teranishi, Michael Heroux, Jackson Mayo, Hemanth Kolla, Jacqueline Chen, Manish Parashar

Thank you