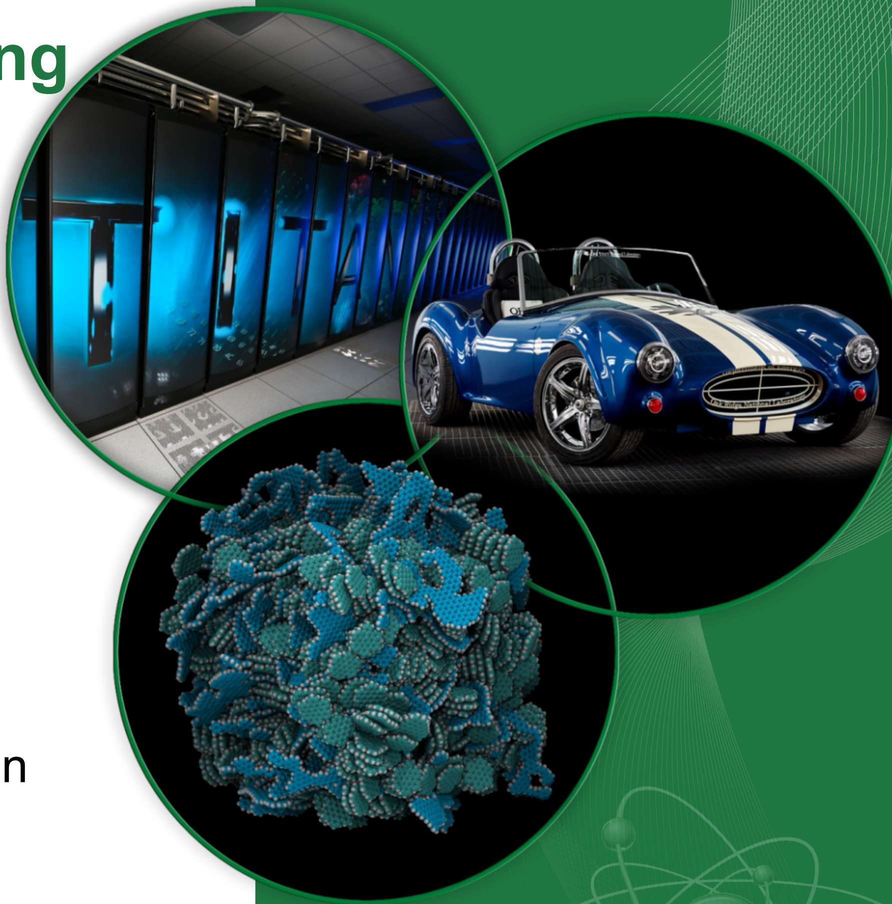
Towards New Metrics for High-Performance Computing Resilience



Saurabh Hukerikar
Rizwan A. Ashraf
Christian Engelmann

Computer Science & Mathematics Division
Oak Ridge National Laboratory

Design Patterns for HPC Resilience

- **The Paradox of Choice:** Several resilience solutions [hardware, system software, algorithm-based, programming model-based, etc.]

× Incomplete understanding of protection coverage against high-probability & high-impact vs. less likely & less harmful faults

× No evaluation methods & metrics that consider
  - Fault impact scope, handling coverage and handling efficiency
  - Performance, resilience and power trade-offs

× No mechanisms and interfaces for coordination for avoidance of costly overprotection

× No portability across architectures and software environments

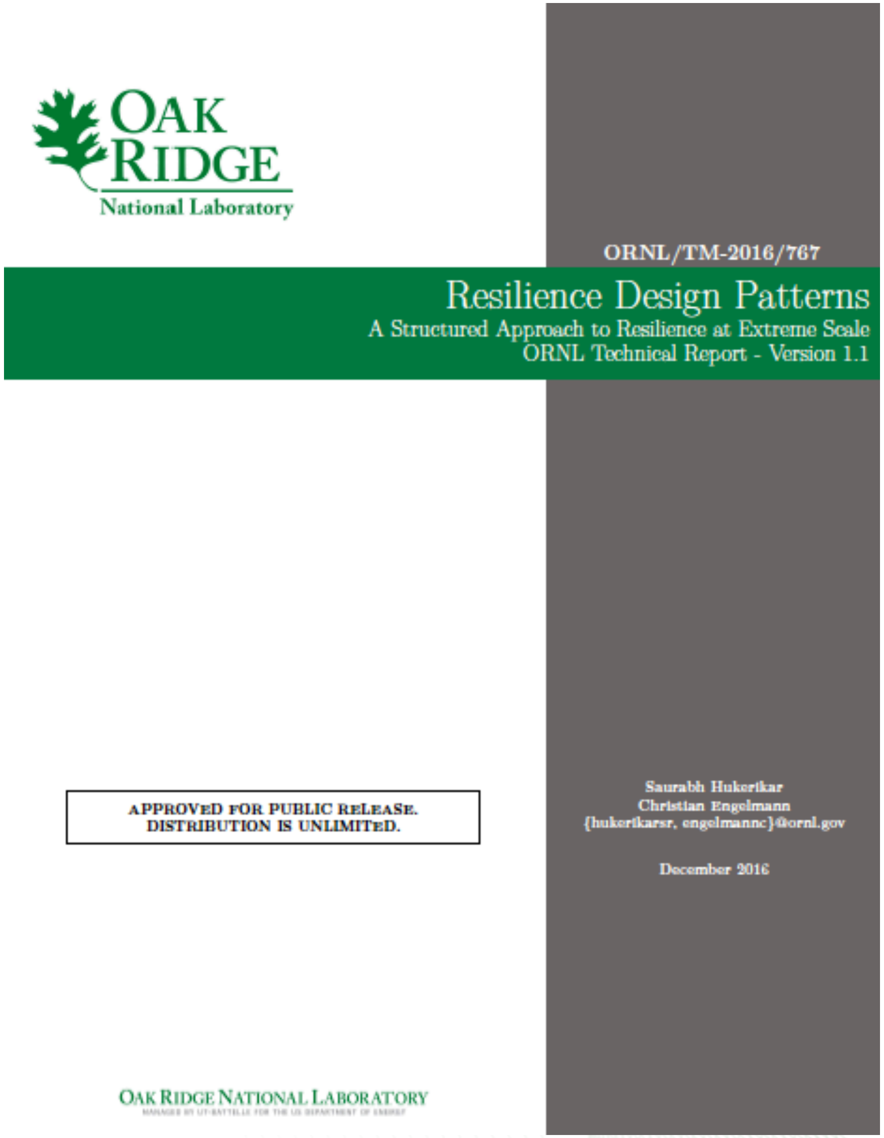
Resilience Design Patterns
A Structured Approach to Resilience at Extreme Scale

- Design Patterns
  - Structural elements that capture an idea in architectural design
  - Patterns describe the essence of a solution to a problem that occurs often in practice
  - Every pattern is an unfinished design
- Each pattern described a problem, which occurs repeatedly in our environment, and then described the core of the solution to that problem, in such a way that this **solution may be used a million times over, without ever doing it the same way twice.**

Resilience Design Patterns Specification

Specification Document v1.1:

- Complete catalog of resilience design patterns
- Detailed descriptions of the components that make up detection, containment, mitigation solutions
- Pattern solutions may be adapted to any system architecture, software environment


OAK RIDGE National Laboratory

ORNL/TM-2016/767

Resilience Design Patterns
A Structured Approach to Resilience at Extreme Scale
ORNL Technical Report - Version 1.1

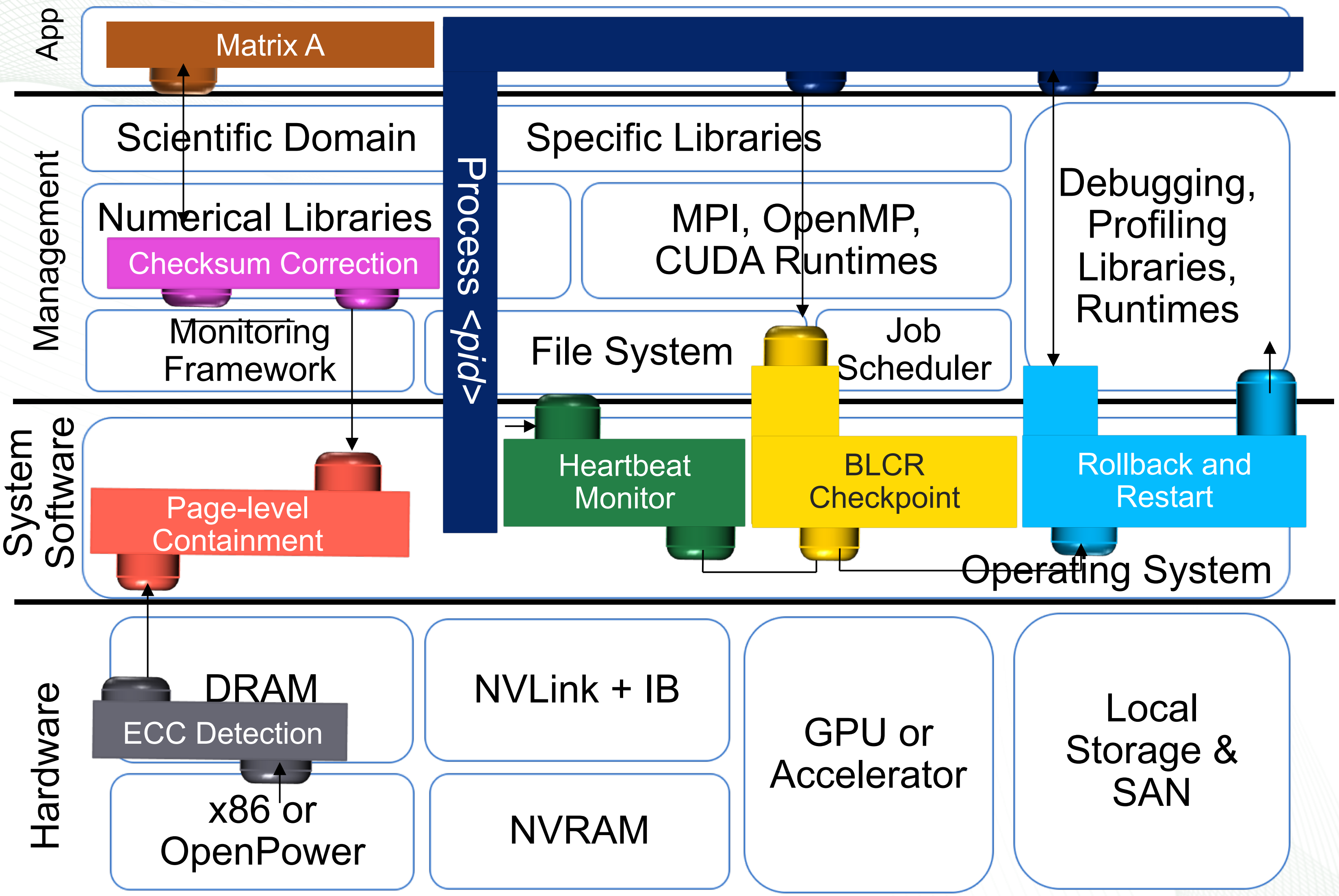
APPROVED FOR PUBLIC RELEASE.
DISTRIBUTION IS UNLIMITED.

Saurabh Hukerikar
Christian Engelmann
{hukerikarsr, engelmannc}@ornl.gov

December 2016

OAK RIDGE NATIONAL LABORATORY

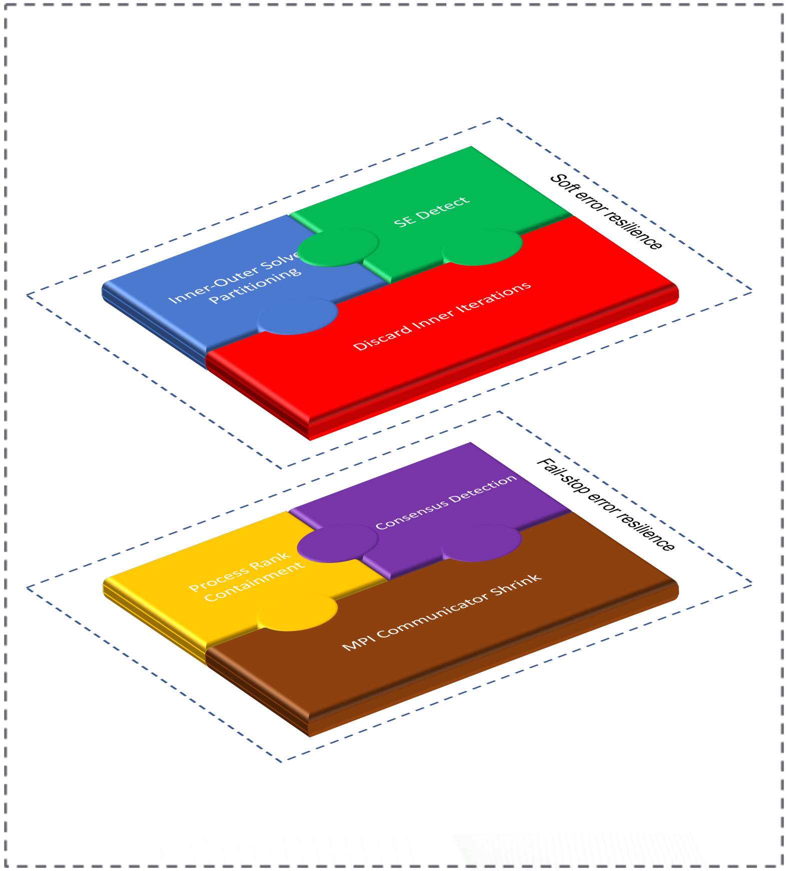
Structured Approach to Building Resilience Solutions



OAK RIDGE
National Laboratory

Pattern Use Case: Composing Resilience Solutions

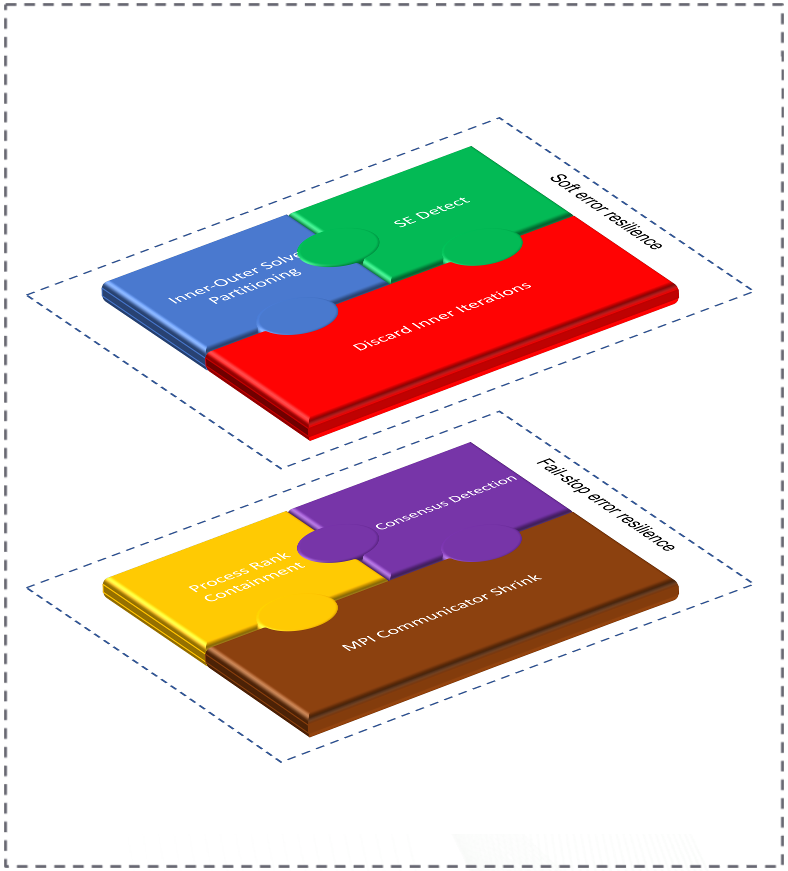
- Two classes of errors we care about:
  - Hard Errors -> Process Failures
  - Soft Errors -> Silent Data Corruption
- Building a unified resilience solution for multi error types using discrete solutions
- Resilience design patterns enable:
  - Identifying detection, containment, mitigation patterns
  - Composition of patterns refinement and optimization of patterns into full solutions



How do we measure improvement in application resilience?

Measuring Resilience

- How do the inclusion of specific hardware or software-based solutions improve an application's ability to deliver a correct outcome and its impact on the application's performance?
- How does the combination of multiple resilience solutions implemented across multiple layers of the system stack impact application reliability and performance?



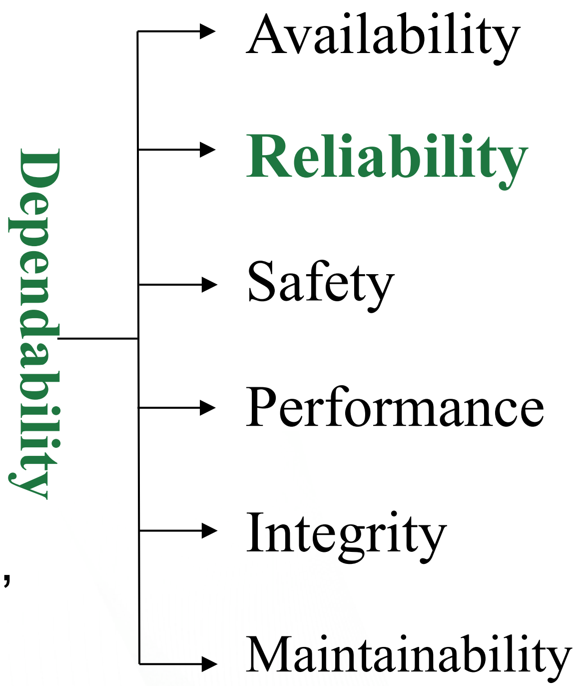
How do we measure improvement in application resilience?

Resilience 101

Resilience is concerned with the correctness of an HPC application in lieu of, or even at the expense of, the reliability of the system. Resilience solutions are designed to enable ***effective*** *and* ***cost efficient*** *management of faults, errors and failures in HPC systems.*

What's wrong with MTTF?

- Dependability is a property of a system that indicates whether is operating properly.
  - Formally, it is the quality of **delivered service** by a computing system such that reliance can *justifiably* be placed on the service
- MTTF is a metric for the **system reliability** attribute:
  - Service-oriented view of system
  - Measured in terms of continuous service accomplishment, or the time to failure from a reference point in time
- Justifiably useful for measuring reliability of system, i.e., how long can my application run before failure causes it not to run.
  - BUT, does it measure the application's ability to produce a correct outcome and performance cost of dealing with faults, errors, failures?
  - Same argument can be made about its variants MTTI, AMTTF and others

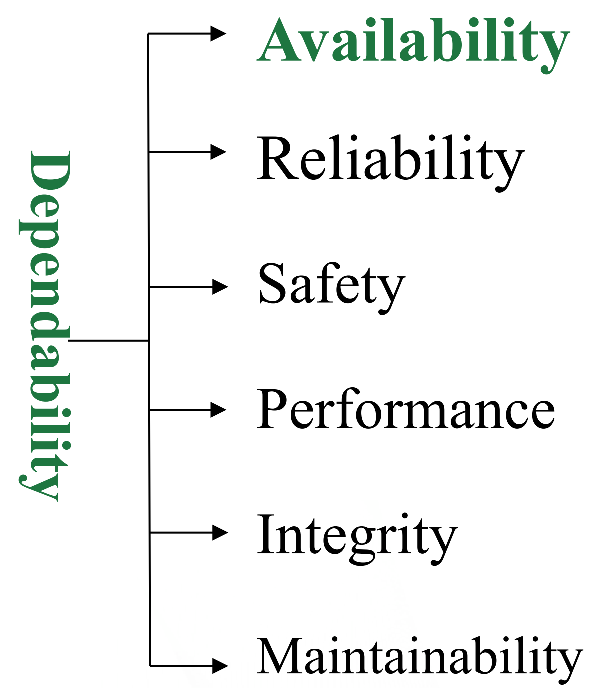


Attributes of dependability

OAK RIDGE National Laboratory

What's wrong with Availability metrics?

- A = $\frac{MTTF}{MTTF + MTTR}$
- Also based on service-based view of system
- Captures the time to repair & restore service
- Justifiably useful for measuring availability of platform, i.e., what fraction of time is the system can provide continuous service for my application to run.
  - AGAIN, does it measure the application's ability to produce a correct outcome and performance cost of dealing with faults, errors, failures?



OAK RIDGE National Laboratory

Vulnerability Factors?

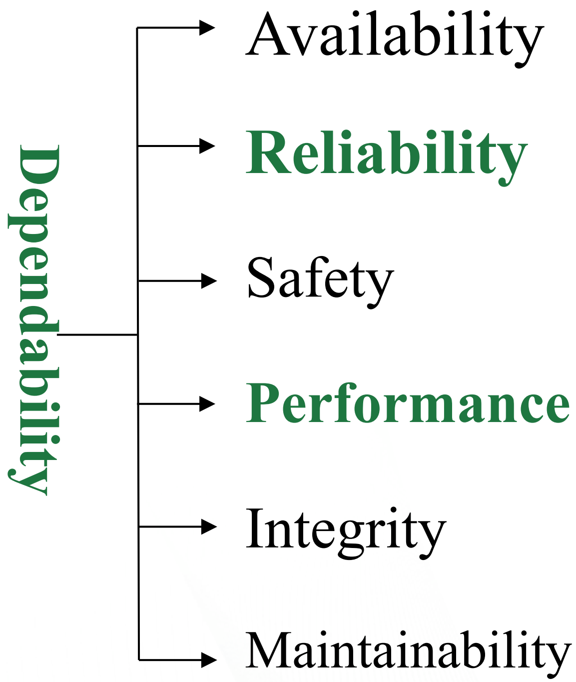
- Architectural Vulnerability Factor (AVF)
  - Measures vulnerability of μ-arch structures to silent errors in terms of impact on program outcome.
- Several variations:
  - TVF, DVF, PVF, etc.
- Vulnerability and resilience are different attributes
  - Negatively correlated
  - Non-perfect correlation

Quantifying Application Resilience

- **Scenario 1:** Application affected by multiple types of fault, error and failure events that impact applications in different ways (incorrect outcomes, performance degradations, fatal failures)
- **Scenario 2:** Resilience solutions that improve platform's dependability may not proportionally increase application resilience
- **Scenario 3:** Cost-benefit analysis of new resilience solutions, whether hardware- or software-based that claim improvement in resilience under specific fault injection scenarios
- **Scenario 4:** Quantify impact on application performance and reliability due to approximation, self-correcting, healing algorithms.
- **Scenario 5:** Applications run on degraded platforms or software environments
- **Scenario 6:** Cross-layer resilience solutions, which use capabilities from multiple layers of the system stack

Outcome Metrics: Measuring What Matters

- Due to the complexity of modern HPC environments, understanding the chain of events from the activation of a fault, the propagation of the resulting error, and the ultimate impact on an application's execution is hard
- Outcome metrics: focus on measuring quantifiable indicators that gauge impact on **results** or **outcomes**
  - Often used in process improvement, engineering of complex systems
  - Holistically evaluate attainment of objectives
- Resilience outcome metrics must focus on reliability **and** performance attributes of the application
  - Scientific outcome and time to solution



Resilience Factor (Value Efficiency)

- Relative value efficiency of a application value

$$RF_{VE} = \frac{ProgramValue_{event-free}}{ProgramValue_{events}} = \frac{V_x}{V_x + |\sigma|}$$

- This **Δ** term represents the variance in a program's value due to the occurrence of fault events during its execution
- $\text{ProgramValue}_{\text{event-free}}$ can be obtained from runs that provably fault free, theoretical values, average of several runs, or uncertainty quantification methods
- Value efficiency metric is designed to measure the impact of faults on **scientific outcome** of an application
  - Obviously not applicable to any control flow variables, pointer and address values
- The key to using value efficiency outcome metrics is identifying the right application outcome values

Resilience Factor (Performance Efficiency)

- Performance efficiency of achieving the outcome in the presence of fault events

$$RF_{PE} = \frac{time - to - solution_{event-free}}{time - to - solution_{events}}$$

- Relative efficiency measure: quantifies the extent to which the performance of an application is impacted by the occurrence of fault events
  - time-to-solution$_{\text{event-free}}$ can be obtained from runs that provably fault free, theoretical peaks, average, or uncertainty quantification methods

- Performance efficiency of resilience solution

$$RF_{PE} = \frac{time - to - solution_{Original}}{time - to - solution_{SolutionX}}$$

  - Measures relative efficiency of a resilience solution for similar fault rates

Resilience Factor Yield (RY)

- **Composite measure of resilience by aggregating multiple RFs**
  - RF is a ratio that calculates performance and value efficiency rather than an absolute execution time or absolute data value

- Based on Geometric Mean of RF values
  - Provides a measure of central tendency
  - Geometric mean has the property that the geometric mean of the ratios is the same as the ratio of the geometric means

- Composite measure of value efficiency of several application variables provides a more complete measure of application reliability

$$RY_{VE} = \sqrt[n]{RF_{VE_1} RF_{VE_2} \ldots RF_{VE_n}}$$

- Composite measure of performance efficiency of several tasks, processes:

$$RY = \sqrt[n]{RF_{T1} RF_{T2} \ldots RF_{Tn}}$$

Applications of the Resilience Factor

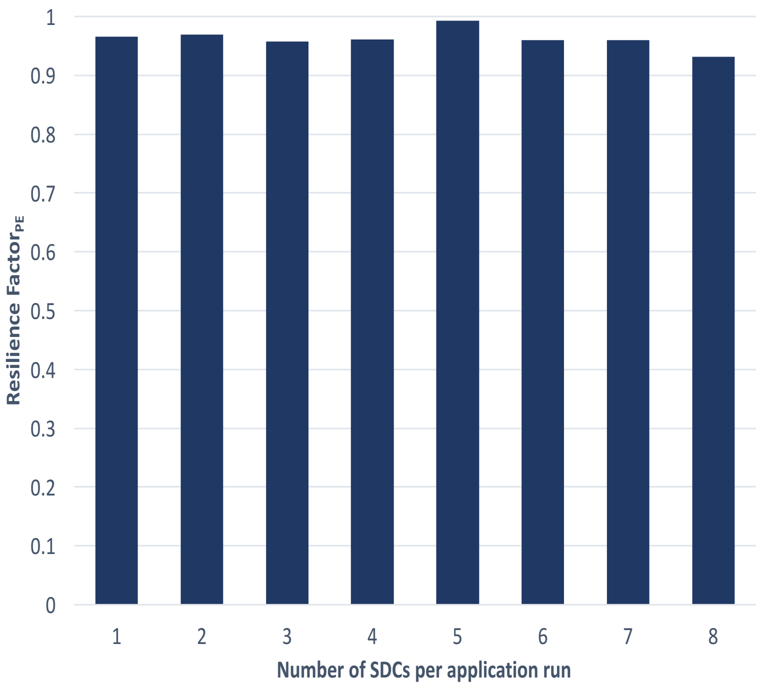
- Understanding the application resilience in terms of performance efficiency and reliability of its outcome for a range of scenarios:
  - **Multicomponent hardware/software environment:** applicable for various granularities, e.g. evaluation of RF of functions, threads, libraries, etc.
  - **Portability of resilience solutions:** evaluating application resilience properties on new architectures, with different software environments, programming models, tools
  - **Fault rate scalability of applications:** standardized measure for evaluating the performance and reliability
  - **Protection coverage versus Performance Overhead:** when stacking several discrete solutions efficiency measures provide impact on application reliability and performance

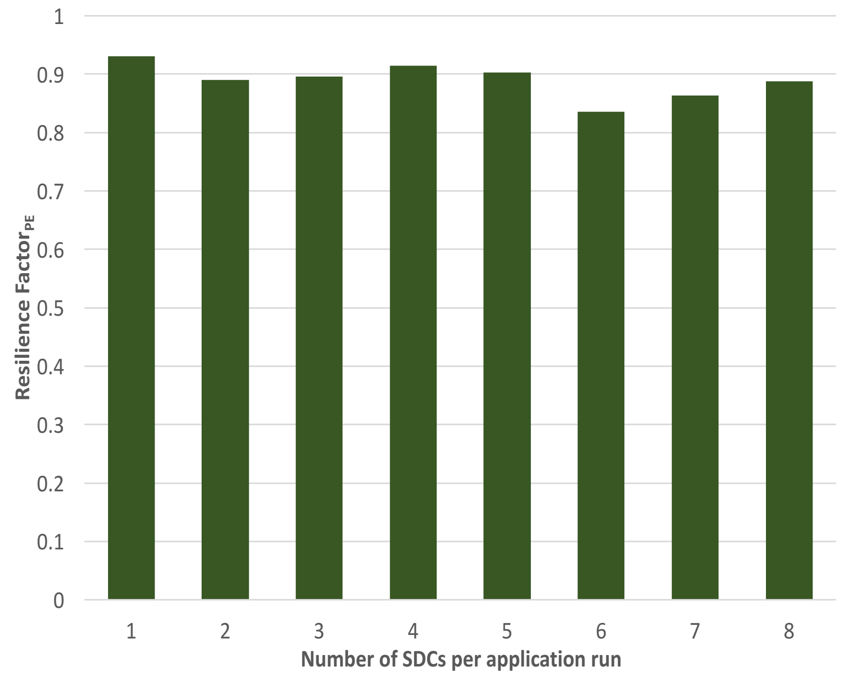
Applying the RF to measure Hard and Soft Error Resilience of a Linear Solver

- Soft Error Resilience for Fault Tolerant GMRES
  - Algorithm-based (ABFT) Resilience for Silent Data Corruptions
  - Based on concept of selective reliability [*Hoemmen et al.*]
  - **Outer solve:** highly reliable; **Inner solve:** “bulk” reliability
  - **Detection:** track residual norm of solver
  - **Mitigation:** discard limited solver iterations
- Hard Error Resilience
  - User Level Fault Mitigation (ULFM) extensions to MPI
  - **Failure detection:** based on ULFM return codes
  - **Failure recovery:** revoke communicator and shrink

Measuring performance efficiency of selective reliability model for FT-GMRES

- Silent data corruptions (SDC) often do not raise interrupts
- May not even affect correctness of solver outcome but performance penalty may incurred due to additional solver iterations to converge





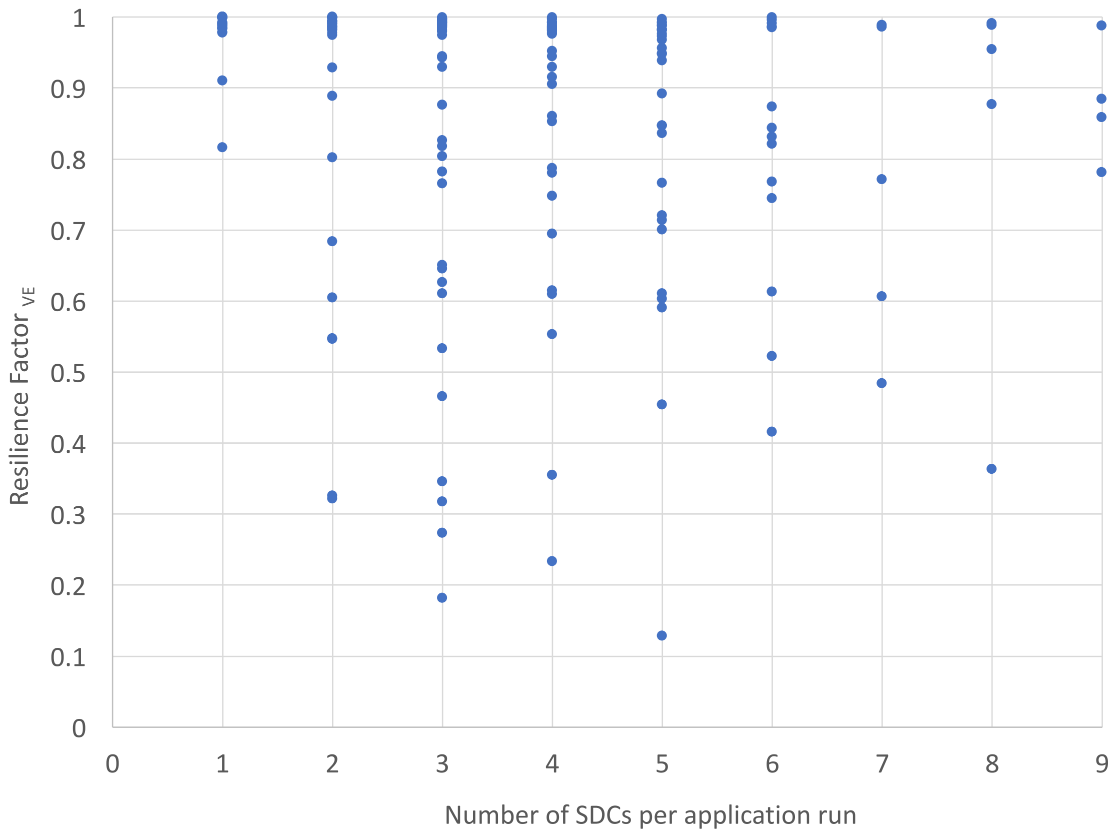
- RF_{PE} captures the impact of loss of performance on account of SDCs

- The same RF_{PE} metric captures the cost of applying the selective reliability (sandbox) model

Measuring FT-GMRES resilience to SDCs

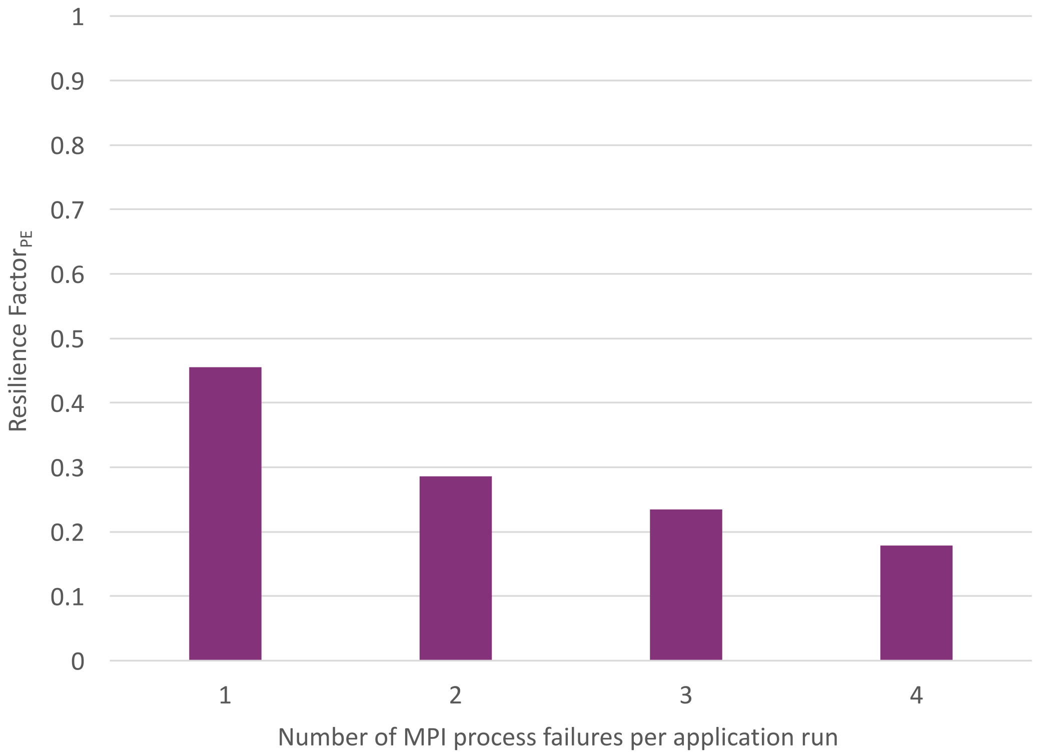
- Quantifying application resilience of the solver in terms of the impact of silent data corruptions:
  - Measurement of RF_{VE} of the solver's residual norm value

- Using the selective reliability model for mitigation of SDCs works well most of the time



Measuring performance efficiency of process failure recovery using ULFM

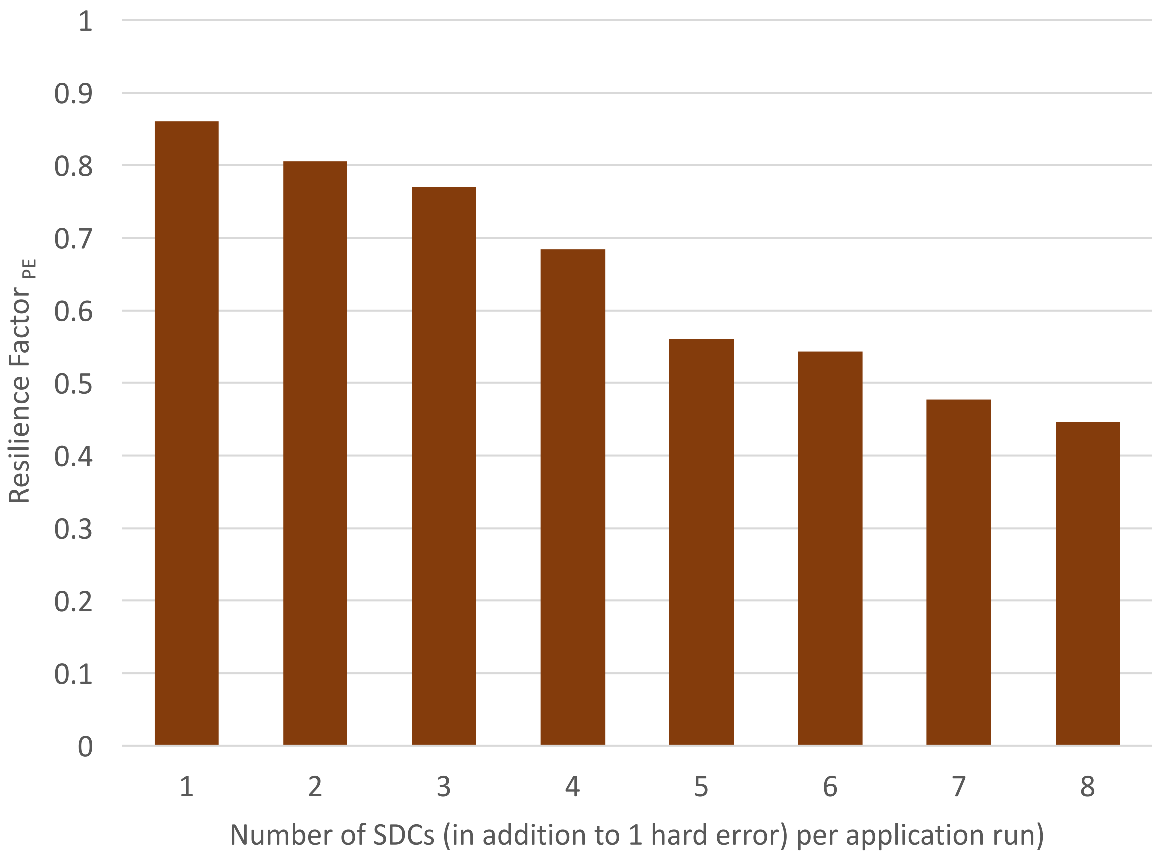
- Recovery from MPI process failures performed using MPI communicator revoke and shrink
  - 32 MPI ranks, up to 4 failures per solver run
- RF_{PE} metric used to captures the performance efficiency of handling process failures
- Same metric, different fault model (process failures) and very different type of solution (MPI library-based)



OAK RIDGE National Laboratory

Measuring performance efficiency of stacking solutions

- FT-GMRES code protected against two fault models: SDCs and process failures.
- Application runs subjected to both types of faults on a random basis
- RF_{PE} metric used to capture the combined impact of an ABFT solution combined with a MPI-layer resilience solution
- Single measure of resilience (in terms of performance efficiency) of solutions that protect against two different fault models

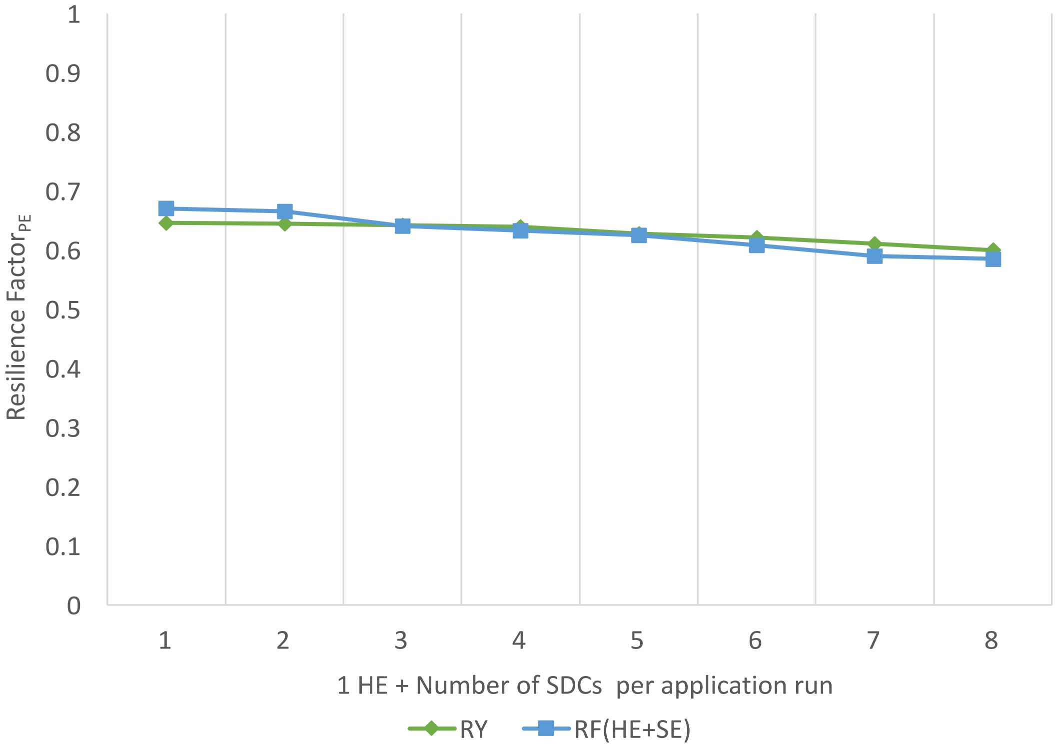


Calculating Resilience Factor Yield (RY)

- RY is computed using RF_{PE} of the ULFM solution and RF_{PE} ABFT solution

$$RY = \sqrt{RF_{PE\text{-}HE} \cdot RF_{PE\text{-}SE}}$$

- The RF_{HE+SE} is measured using experiments runs that include hard and soft error injection; FT-GMRES protected using ULFM and ABFT solution



OAK RIDGE National Laboratory

Conclusion

- We have accepted that faults, errors, failures will be the norm given the complexity of modern HPC environments
- Resilience solutions are all about applications learning to live in these environment
  - Resilience is concerned with reliability of scientific application outcomes and performance efficiency
  - To quantify these **attributes**, the traditional dependability metrics are incompatible (they provide measures for platform reliability and availability attributes instead)
- Outcome metrics
  - RF_{PE} captures performance efficiency, i.e., the impact on performance on account of dealing with fault, error, failure events.
  - RF_{VE} captures the impact of events on application's output values (scientific outcomes)
  - Focus on combined impact on reliability and performance
- Provide measure of resilience from an application's perspective
  - Independent of nature of fault, type of solution(s), programming model, system architecture
  - Enables measurement of combined impact of multiple event types, solutions

Thank-you!

Oak Ridge National Laboratory