Academic Journal of Architecture and Civil Engineering
Print ISSN: 2960-0235
Online ISSN: 2960-0243
DOI: https://doi.org/10.61784/ajace4020

RSRBENCH: REPRODUCIBILITY AND PERFORMANCE BENCHMARKING FOR REGIONAL SEISMIC RISK WORKFLOWS

Dai Teng
University of Illinois Urbana-Champaign, Urbana 61801, IL, USA.

Abstract: Regional seismic risk assessment involves computational workflows that integrate hazard mapping, structural response analysis, and damage-and-loss estimation. Although these workflows are increasingly executed in parallel, their reproducibility across different execution configurations and failure conditions remains insufficiently characterized. This paper presents RSRBench, a reproducible benchmarking framework for evaluating the consistency, performance, and reliability of regional seismic assessment workflows. Using publicly available San Francisco and Anchorage datasets and the NHERI SimCenter software, we examine the effects of execution parallelism, random-number generation, task scheduling, and failure recovery. An execution-level audit reveals that changing the number of parallel processes alters hazard assignments for 596–598 of 600 evaluated buildings, while applying a single global random seed increases portfolio-loss variability by 38% in the tested configuration. To address these issues, RSRBench introduces asset-specific random-number streams, order-independent aggregation, and task-level recovery mechanisms. Experiments demonstrate identical per-asset results across the tested scheduling and failure-recovery configurations. We further evaluate computational scalability using measured task execution times and a trace-driven simulator validated against process-pool emulation, with a mean absolute prediction error of 3.5%. At 1,024 simulated workers, cost-aware scheduling improves parallel efficiency from 0.88 to 0.99 compared with static cyclic scheduling. These findings establish a reproducible evaluation methodology for improving the consistency and computational efficiency of regional seismic risk workflows.
Keywords: Regional seismic risk; Reproducibility; Benchmarking; Scientific workflows; Parallel random streams; Fault tolerance; NHERI SimCenter

1 INTRODUCTION

Regional seismic risk simulation estimates the damage, repair cost and functional loss of every building in a city for one or many earthquake scenarios. A typical simulation maps a gridded ground-motion field to each asset, predicts structural demands either directly from intensity measures or through nonlinear time-history analysis, and propagates fragility and consequence uncertainty by Monte Carlo sampling [1-4]. Because each asset is evaluated independently once the hazard is mapped, the computation is embarrassingly parallel at the asset level, and inventories of 10^5–10^6 buildings are routinely processed on clusters [5-6].

The NHERI SimCenter has made this class of simulation broadly accessible. Its regional workflow driver rWHALE, the R2D desktop front end, the EE-UQ application and the pelicun damage-and-loss engine form an open, modular framework in which hazard, mapping, modeling, response, and loss applications are chained per asset [5, 7-8]. The framework is therefore not missing; what has received far less attention is how its results behave when the *execution* changes while the scientific inputs stay fixed. A study that is rerun on a different number of processes, with a different task order, after a node failure, or with an updated library should either reproduce its previous numbers or explain precisely why it does not. Without that property, regression testing of new model versions, comparison of scheduling strategies, and independent verification of published loss estimates all degrade to informal judgments about whether two noisy results "look similar".

This paper addresses that gap with RSRBench, a reproducible benchmarking framework that drives the *unmodified* public NHERI components on two public testbeds and specifies (i) standardized workloads with content-addressed inputs, (ii) execution profiles that vary process count, schedule, fault pattern and environment, (iii) a tiered result-reconciliation protocol, and (iv) performance and resilience metrics. Rather than proposing another platform, RSRBench turns controlled variations of the existing workflows into measurable experiments. Its technical contributions are:

1. An execution-level audit of the reference workflows that locates five reproducibility and performance bottlenecks at the source-code level and quantifies them with real runs. Under the MPI mode generated by rWHALE, the hazard-to-asset mapping assigns different ground motions to 596–598 of 600 buildings when only the process count changes; the damage-and-loss stage is unseeded, so two identical runs differ for every building; and fixing one global seed restores repeatability but forces all buildings to share their random draws, which inflates the standard deviation of the portfolio loss by 38%.
2. Keyed-stream execution, a non-intrusive technique that derives every random stream from (global seed, stage, asset key) and aggregates with a correctly rounded, order-independent sum. It needs an 11-line patch to one application plus

© By the Author(s) 2025, under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0).

per-asset seed injection through the existing configuration, and it makes results bit-identical across 1–16 MPI ranks, six scheduling policies, injected task failures and worker crashes.
3. A tiered reconciliation protocol (bitwise, numerical, statistical, completeness) whose sensitivity is measured with controlled perturbations. Because keyed streams align realizations across runs, a hazard perturbation of 0.5% is localized to the affected buildings exactly, whereas a conventional statistical comparison of independent runs with the reference 100 realizations per building detects none of the perturbations up to 10%.
4. A trace-driven performance and resilience study calibrated with 600 measured damage-and-loss tasks and 220 measured nonlinear time-history tasks, validated against process-pool emulation (mean absolute error 3.5%). At 1024 workers on the 84,436-building Anchorage inventory, cost-model-guided longest-processing-time scheduling raises parallel efficiency from 0.88 (static cyclic, the rWHALE default) to 0.99 (dynamic self-scheduling alone reaches 0.98), and task-level recovery keeps the time-to-solution within 0.09% of the failure-free value at a per-task failure probability of 10^{-3}, where whole-job restart does not complete.
All code, configurations, raw measurements and the scripts that generate every number in this paper are released with the benchmark.

2 BACKGROUND AND RELATED WORK

2.1 Regional Seismic Risk Workflows

Building-level loss estimation for regions is commonly based on the Hazus methodology, which couples capacity-spectrum or intensity-based fragility functions with consequence models for repair cost and time [1-2], and on the component-based FEMA P-58 methodology for higher-fidelity assessments [3]. The OpenQuake engine provides an open-source implementation of event-based and scenario risk calculations for portfolios [9]. City-scale nonlinear time-history analysis replaces intensity-based fragilities with multi-degree-of-freedom shear models calibrated from Hazus data [4, 10]; its open-source implementation is integrated in the SimCenter workflow as the MDOF-LU modeling application [4, 8]. The SimCenter framework organizes these models into applications that exchange JSON files per asset and are orchestrated by rWHALE on desktops or on the DesignSafe cyberinfrastructure [5-6]; pelicun implements the damage and loss calculations and ships the Hazus and FEMA P-58 model libraries [7]. Spatial correlation of ground-motion intensity is known to control the dispersion and tail of portfolio losses [11-12]; the present work shows that the correlation structure of the *sampling* itself, which is an execution detail, has a similar effect when it is not controlled.

2.2 Reproducibility of Computational Studies

Reproducibility has been identified as a requirement for credible computational science, together with the publication of code, data and the exact computational environment [13-14]. Provenance records describe what should be stored with a result, and containers and package managers fix the software environment. These measures keep the environment stable but do not, by themselves, make a parallel Monte Carlo computation invariant to how work is partitioned. Benchmarking methodology for parallel systems emphasizes reporting variability, specifying the environment, and comparing against bounds [15]; RSRBench follows these rules for a domain workflow.

2.3 Parallel Random Streams and Reproducible Reductions

Parallel stochastic simulation requires many statistically independent random streams. Recurrence-based generators obtain streams by jumping ahead, while counter-based generators compute each number from a key and a counter and are therefore naturally suited to deterministic, order-free generation [16-17]. NumPy's SeedSequence hashes arbitrary integer tuples into well-mixed generator states [18]. Floating-point summation is not associative, so parallel reductions can change results with the order of operands; correctly rounded summation [19] and reproducible summation algorithms [20] remove this dependence. RSRBench combines both ideas at the level of a workflow whose tasks are applications rather than loops.

2.4 Scheduling and Fault Tolerance of Task Farms

For independent tasks, list scheduling in longest-processing-time order has a worst-case bound of $4/3 - 1/(3P)$ of the optimum [21]; self-scheduling and guided self-scheduling trade dispatch overhead against load balance [22]. Checkpoint/restart theory gives optimal checkpoint intervals for long-running jobs [23-24]; task farms with idempotent tasks can instead recover at task granularity, as many scientific workflow systems do. RSRBench quantifies how these choices interact with the heavy-tailed task costs of time-history analysis and with the fail-stop semantics of the reference driver.

3 REFERENCE WORKFLOWS AND EXECUTION-LEVEL AUDIT

3.1 Execution Model of the Reference Workflows

We study the public SimCenter backend at commit 048d099 (19 Oct. 2026) and pelicun 3.10.0 with the Damage and Loss Model Library 3.2 , executed on the R2D example data [25]. The prior R2D and pelicun platform releases are documented in [8] and [7], respectively. In the intensity-measure (IM) path (R2D example E1), rWHALE creates one asset file per building, maps the hazard with the nearest-neighbor application (NNE.py: $K = 4$ nearest grid sites, inverse-squared-distance weights, $S = 100$ samples, seed 687), and then runs, for each asset, an event application, the IM-as-demand application and the pelicun wrapper. In the time-history path (example E2), the modeling application MDOF-LU generates a Hazus-calibrated shear model and OpenSees performs the nonlinear analysis [4, 26]. For cluster execution, rWHALE's parSETUP mode writes a script that launches asset creation, the mapping application and the per-asset loop with mpiexec -n P; inside the loop, rank r processes the assets with count % numP == r, i.e., a static cyclic partition. pelicun 3.10 additionally provides regional_sim, an in-process alternative that sorts the inventory by occupancy class, splits it into chunks of 1000 buildings and evaluates the chunks in parallel with joblib.

3.2 Findings

Table 1 summarizes five findings; each is traced to source code and measured in Section 6.
F1 – Rank-dependent hazard mapping. NNE.py initializes np.random.default_rng(seed) once per MPI rank and then draws the neighbor samples of its assets sequentially. The draws received by an asset therefore depend on the number of ranks and on the asset's position within its rank's subset, although the seed is fixed.
F2 – Unseeded damage and loss sampling. pelicun's default configuration sets Seed to null, the Hazus auto-population script sets no seed, and regional_sim constructs each chunk's assessment without a seed. Repeated runs of an identical configuration consequently sample different damage and loss realizations.
F3 – Common random numbers under a global seed. The configuration does accept a seed (Options/Seed), but a single value is copied to every asset. Since every asset consumes the same number of draws in the same order, all buildings then share their uniform variates, which couples their capacity and consequence realizations.
F4 – Fail-stop execution with silent omission. The workflow helper run_command terminates the driver with sys.exit(-20) when an application returns a non-zero code, and the result aggregation skips assets whose DL.json is missing after printing a warning. A failure therefore either aborts a whole rank or, if an application fails without an error code, yields a regional result with missing buildings.
F5 – Process-per-stage execution. The in-process execution path of run_command is disabled, so every Python application is a separate interpreter; the IM path starts three interpreters per asset and writes about 26 files per asset.

Table 1 Execution-Level Findings in the Reference Workflows

ID	Location	Observable effect (Sec. 6)
F1	NNE.py, find_neighbors	event assignment changes with P for 596–598 of 600 assets
F2	pelicun default Seed; regional_sim	identical runs differ for all assets
F3	DL/Options/Seed copied to all assets	portfolio-loss std. +38%
F4	run_command; aggregate_results	whole-job restart; missing assets only warned
F5	run_command (subprocess per app)	1.86 s vs. 0.52 s per asset

4 RSRBENCH DESIGN

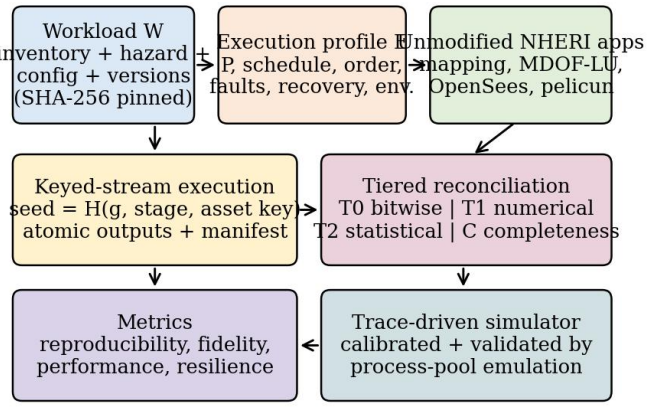


Figure 1 RSRBench. Content-Addressed Workloads Are Executed by Unmodified NHERI Applications under Controlled Execution Profiles; Outputs Are Recorded in a Manifest and Reconciled in Tiers; Traces Feed a Validated Simulator for Scales beyond the Measurement Node

Figure 1 shows the structure of RSRBench. A benchmark case is the tuple (W, E, R, M) of a workload, an execution profile, reconciliation rules and metrics; two runs are comparable only if they share W and differ in a documented subset of E.

4.1 Workloads

A workload fixes the inventory subset, hazard files, workflow configuration and software versions, each identified by a SHA-256 digest that is stored with the results. Table 2 lists the tiers. SF-600 is a systematic sample (every 26th building) of the 15,836-building San Francisco inventory of R2D example E1 and is executed end to end with real runs; SF-15.8k and SF-150k reuse the measured task costs for the full E1 inventory and for the 149,970-building city-wide inventory distributed with the example. ANC-trace consists of 220 measured time-history tasks stratified by story count from the 84,436-building Anchorage inventory of example E2, each excited by the nearest of the 12 recorded ground motions of the 2018 M_w 7.1 Anchorage earthquake that are included in the example; ANC-84k scales these costs to the whole inventory through a cost model.

Table 2 Workload Tiers (All Data from the Public R2D Examples)

Tier	Assets	Hazard	Stages	Use
SF-600	600	M7.2 scen., 192 sites	map+IM+DL	real runs
SF-15.8k	15,836	same	same	trace sim.
SF-150k	149,970	same	same	trace sim.
ANC-trace	220	12 records, 2018	MDOF-LU+NLTHA	real runs
ANC-84k	84,436	same	same	trace sim.

4.2 Execution Profiles

An execution profile fixes everything that should *not* influence the result: process count P, scheduling policy, task order, fault pattern, recovery policy and library versions. RSRBench provides four families: E0, the native rWHALE driver in sequential mode; E1, the native parSETUP script executed with mpiexec -n P; E2, pelicun's regional_sim; and E3, a benchmark executor that runs the same per-asset task in persistent worker processes under the policies block, cyclic, dynamic self-scheduling with chunk size c, guided self-scheduling (GSS), longest-processing-time first (LPT) and random order. The E3 task writes the identical asset file and demand file that rWHALE writes and calls the same pelicun entry point (run_pelicun); with an identical seed it reproduces the native per-asset outputs bit for bit (Section 6), so differences between E0 and E3 isolate the process model.

4.3 Keyed-Stream Execution

Let g be a global seed, s a stage identifier (mapping, modeling, damage-and-loss) and $k(a)$ a stable 63-bit key of asset a (its numeric identifier or a hash of it). Keyed-stream execution seeds every random draw of stage s for asset a with

$$\sigma_{s,a} = H\big(g, s, k(a)\big), \tag{1}$$

where H is NumPy's SeedSequence hash, which maps integer tuples to well-mixed generator states [18]. Let a run be described by a partition π of the assets over workers, an order ω of execution and an attempt history η (failures and retries). If the output of every task is a deterministic function $y_a = f_s\big(x_a, \sigma_{s,a}\big)$ of its inputs and its seed, then y_a does not depend on (π, ω, η). The regional result is a reduction $Y = \bigoplus_a y_a$; with floating-point addition the reduction depends on the order of operands, so RSRBench aggregates in canonical asset order with a correctly rounded sum (Python's math.fsum), whose result is the exactly rounded value of the real sum and is therefore order independent [19]. Together these two rules give

$$Y(\pi, \omega, \eta) = Y\big(\pi', \omega', \eta'\big) \quad \text{bitwise, for all executions.} \tag{2}$$

Distinct keys yield statistically independent streams, so keyed streams also avoid the cross-asset coupling of F3. Applying the technique to the reference workflow requires (i) an 11-line change in NNE.py that draws each asset's neighbor samples from default_rng([seed, k(a)]) instead of a per-rank stream, and (ii) injecting $\sigma_{\mathrm{DL},a}$ through the existing per-asset option DL/Options/Seed, which needs no change to pelicun. Figure 2 shows the patch and Algorithm 1 the resulting execution procedure. Task outputs are written atomically (temporary file followed by an atomic rename) and recorded in an append-only manifest with their SHA-256 digest, attempt number and duration; a restarted executor reuses every output whose digest is recorded and recomputes only the rest.

Algorithm 1: Keyed-stream execution of one benchmark run

1. Input: assets $\mathcal{A}$, global seed g, stages $\mathcal{S}$, policy, P, manifest $\mathcal{M}$
2. $\mathcal{R} \leftarrow \{a \in \mathcal{A}:\ a$has no verified entry in$\mathcal{M}\}$ (resume)
3. partition or queue $\mathcal{R}$ over P workers according to the policy
4. for all tasks a executed by a worker
5. for all stages $s \in \mathcal{S}$
6. $\sigma_{s,a} \leftarrow H\big(g, s, k(a)\big)$

7. run the unmodified application of stage s with seed $\sigma_{s,a}$
8. end for
9. write y_a to a temporary file and rename it atomically
10. append $(a,\ \mathrm{SHA256}(y_a),\ \text{attempt, duration})$ to $\mathcal{M}$
11. on error: retry a; on worker loss: re-queue unfinished tasks
12. end for
13. $Y \leftarrow$ correctly rounded sum of y_a in canonical key order
14. return Y and the dataset digest $\mathrm{SHA256}(\text{sorted per-asset digests})$

```
+def _asset_key(asset_data):     # stable 63-bit key
+      aid = <asset identifier from asset_data>
+      if aid.isdigit():
+            return int(aid)
+      return int.from_bytes(sha256(aid.encode())
+                                    .digest()[:8], 'big') >> 1
 ...
-   nbr_samples = np.where(rng.multinomial(
-                          1, weights, samples) == 1)[1]
+   rng_a = np.random.default_rng(
+                     [seed, _asset_key(asset_data)])
+   nbr_samples = np.where(rng_a.multinomial(
+                          1, weights, samples) == 1)[1]
```

Figure 2 Keyed Patch of the Mapping Application NNE.py (Abridged Unified Diff; the Full Patch Changes 11 Lines). The Per-Rank Generator rng Is No Longer Used for Sampling

4.4 Tiered Reconciliation

Two result sets A and B over the same workload are compared in four tiers. T0 (bitwise): per-asset digests of the canonical byte serialization of all realizations, combined into a dataset digest. T1 (numerical): element-wise $|A - B| \le \epsilon_a + \epsilon_r|B|$ for aligned realizations. T2 (statistical): for unaligned realizations, a Welch test of the per-asset mean loss with Benjamini–Hochberg control of the false discovery rate at $q = 0.05$ across assets, plus a Welch test of the mean portfolio loss [27]. C (completeness): the asset sets and manifest coverage must coincide. A comparison is classified as *identical*, *numerically equivalent*, *statistically indistinguishable* or *divergent* with the list of flagged assets. T0 and T1 are applicable only when realizations are aligned, which is exactly what keyed streams provide.

4.5 Reporting Specification

To make results comparable across groups, every RSRBench run is reported with the fields of Table 3. The fields are produced automatically from the manifest and the run logs; two reports with equal workload and environment fields and different execution fields form a valid benchmark comparison.

Table 3 RSRBench Reporting Specification (One Record per Run)

Group	Required fields
Workload	tier; asset list digest; inventory, hazard, record and model-library digests; workflow configuration digest
Software	backend commit; versions of pelicun, model library, NumPy, SciPy, pandas, OpenSees(Py), MPI; interpreter
Execution	P; policy and chunk size; order seed; fault model and seed; recovery policy; platform description
Sampling	global seed g; stage identifiers; asset key rule; realizations per asset
Results	per-asset digests; dataset digest; manifest coverage; aggregated totals (correctly rounded)
Reconciliation	tier outcome (T0/T1/T2/C) against the reference run; flagged assets
Performance	makespan; efficiency η; CPU time; per-asset time; files and bytes written
Resilience	failed attempts; restarts; executions; time-to-solution ratio

4.6 Metrics

Reproducibility is measured by the fraction of assets with identical digests and, for the mapping, the fraction of ground-motion samples drawn from a different site. Statistical fidelity is measured by the mean, standard deviation and upper quantiles of the portfolio repair cost and by the mean pairwise correlation of building loss ratios across realizations. Performance is measured by makespan, parallel efficiency $\eta = \max\left(\sum_i\ d_i/P, \max_i d_i\right)/T$, where d_i

are task durations and T the makespan, CPU time, per-asset overhead and files written. Resilience is measured by the time-to-solution relative to the failure-free makespan, the number of restarts and the consumed worker time.

4.7 Fault Model, Recovery and Trace-Driven Simulation

Faults are injected per task attempt with probability p as an application error, or as a crash of the worker process; the draw for an attempt is itself taken from a keyed stream so that fault patterns are repeatable. Three recovery policies are compared: *job* restart (a failure aborts the run, which is restarted from scratch, as with F4), *skip* (the run is aborted but restarted runs reuse completed outputs) and *task* (the failed task is retried immediately and other workers continue). Scales beyond the measurement node are evaluated by a discrete-event simulator driven by measured task durations. Its dispatch overhead is calibrated with empty tasks, and its predictions are validated against a real process pool that executes sleep tasks with the same duration distribution, which provides genuine concurrency independently of the number of cores of the measurement node.

5 EXPERIMENTAL SETUP

Data. The SF workloads use the inventory, configuration and hazard of R2D example E1: 192 grid sites with 10,000 spatially consistent PGA realizations of an $M7.2$ San Andreas scenario, $K = 4$ neighbors, $S = 100$ samples, Hazus earthquake damage and loss models for buildings (DLML Hazus v6.1, lifeline-facility option as in the example) and 100 loss realizations per building; repair costs are converted to dollars with each building's replacement cost from the inventory [25]. The Anchorage workloads use the 84,436-building inventory and the 12 two-component acceleration records (time step 0.005 s, 11,400 steps) of example E2, the MDOF-LU modeling application compiled from the backend sources, and the E2 simulation settings (Newmark $\gamma = 0.5$, $\beta = 0.25$; Newton iterations; norm-unbalance test 10^{-2} with 10 iterations; sub-stepping 2 levels $\times$ 10; Rayleigh damping); stiffness and damping factors are sampled from keyed streams with the E2 dispersions 0.2 and 0.1. The OpenSees model is built with OpenSeesPy following the translation rules of the SimCenter OpenSees preprocessor (hysteretic shear springs in zero-length elements) [28].
Table 4 lists the digests that identify the workload inputs; any change of a digest defines a new workload.
Software and platform. Python 3.12.3, NumPy 2.4.4, pandas 2.3.3, SciPy 1.15.3, scikit-learn 1.8.0, pelicun 3.10.0, DLML 3.2, OpenSeesPy 3.8.0.0, mpi4py 4.1.2 built against Open MPI 4.1.6, Ubuntu 24.04. All runs were executed on a dedicated x86-64 node with one vCPU and 3 GB of memory; native multi-process runs (E1, E3 with $P > 1$) are therefore time-sliced on that node, which is immaterial for the reproducibility experiments and is why parallel performance at scale is evaluated with the validated trace-driven simulator. Every reported number is produced by scripts in the released artifact from raw logs and manifests.
Experiments. (A) native E1 runs of asset creation and mapping with $P \in \{1,2,4,8,16\}$ ranks, with and without the keyed patch; (B1) five E3 runs of SF-600 with keyed seeds (two global seeds), one global seed and no seed (twice); (B2) eleven E3 runs of the first 120 SF-600 buildings that vary P, policy, order, faults and recovery; (B3) controlled perturbations of a spatial cluster of 30 buildings; (C) native rWHALE runs (E0, 60 buildings, twice), a per-stage profile and two regional_sim runs (E2); (D) 220 measured time-history tasks; (S) trace-driven scheduling and recovery studies with emulation-based validation.

6 RESULTS

6.1 Execution Changes the Answer (F1–F3)

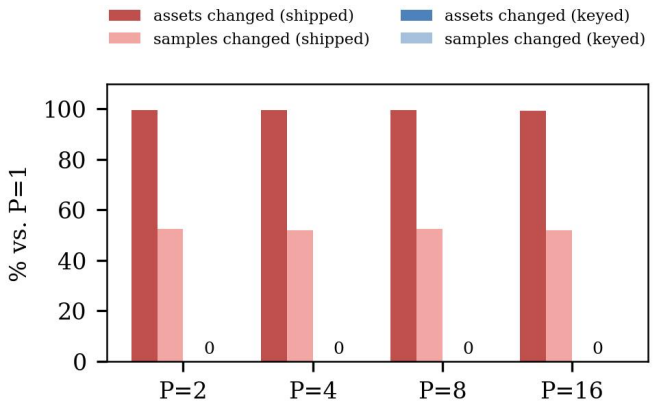


Figure 3 Hazard-to-Asset Mapping of SF-600 Executed with mpiexec -n P: Share of Buildings Whose Event List Differs from the $P = 1$ Run and Share of Ground-Motion Samples Taken from a Different Grid Site, for the Shipped Application and with the Keyed Patch

Table 4 Content-Addressed Workload Manifest (Excerpt)

Workload item	SHA-256 (first 12 hex)
SF inventory, E1 (15,836 bldgs)	e73f08533256
SF inventory, city-wide (149,970)	23099dedc749
SF hazard grid / 192 site files	70543e18b86b / 5e4e907c42e1
E1 workflow configuration	4b26ad26dc0c
ANC inventory (84,436 bldgs)	127e3f8bd0b0
ANC record grid / 12 records	7dbe043c907a / 25a106dd13ce
Hazus MDOF parameter file	9dff956871ea
DLML Hazus v6.1 fragility / repair	4746c12744c9 / 2ea57a633324
NNE.py shipped / keyed	3495189c6198 / 6a7ef4f30e17

Process count (F1). Figure 3 reports the native E1 runs. With the seed fixed at 687, every process count $P \in \{2,4,8,16\}$ changes the event list of 596–598 of the 600 buildings, and 51.8–52.5% of all ground-motion samples are drawn from a different grid site than with $P = 1$. Because the four candidate sites of a building carry different intensity fields, the change propagates to every downstream demand, damage and loss realization. With the keyed patch, all five process counts produce the same event lists (identical dataset digest), and the mapping time is unchanged within run-to-run variation (5.4 s vs. 5.7 s at $P = 1$; 29.0 s vs. 27.1 s at $P = 16$).

Table 5 confirms the mechanism. With $K = 1$ the mapping involves no random draw and is invariant; with $K = 2$, 4 and 8 the share of samples that move to another site grows with the number of candidate sites, because a per-rank stream assigns different uniform variates to the same building. To isolate the downstream effect, we repeated the damage and loss stage of SF-600 with identical keyed damage-and-loss seeds on the mapping produced with $P = 4$. The expected loss ratio changes for 553 of 600 buildings (median absolute change 0.0020, 95th percentile 0.0141, maximum 0.047), and the portfolio loss of an individual realization changes by 2.1% in the median and by up to 23.2%, while the portfolio mean moves from 536.8 to 533.8 million USD and the two distributions remain statistically indistinguishable (Kolmogorov–Smirnov $p = 1.00$). This is the typical signature of an execution effect: invisible to a distributional comparison, but large enough to change every realization-level and building-level result that is reported or reused downstream.

Repeated identical runs (F2). Two native rWHALE runs (E0) of the same 60 buildings with the same configuration produce identical event lists for 60 of 60 buildings, because the mapping is seeded, but identical loss realizations for 0 buildings; 52% of all realizations differ. In the E3 executor the same holds at scale: none of the 600 buildings is reproduced, the median absolute difference of a building's mean loss ratio between two runs is 0.0076 and its 95th percentile is 0.033. pelicun's regional_sim behaves alike: two runs on the same 600 buildings are different, with 51% of the repair-cost cells differing. Before using E3 for the remaining experiments we verified that, given the same seed, it reproduces the native per-asset outputs of rWHALE bit for bit (10 of 10 buildings, all realizations of repair cost, repair time and collapse).

Table 6 Portfolio Repair Cost of SF-600 under Different Sampling (100 Realizations, Million USD; $\bar{\rho}$: Mean Pairwise Correlation of Building Loss Ratios)

Sampling	Seed	Mean	Std. [95% CI]	P90	P99	$\bar{\rho}$
keyed	2024	537	397 [341, 448]	1155	1571	0.35
keyed	2025	534	395 [335, 448]	1123	1628	0.36
unseeded	run A	539	405 [345, 461]	1151	1575	0.35
unseeded	run B	542	403 [340, 463]	1128	1628	0.35
global seed	2024	547	552 [447, 659]	1316	1783	0.71

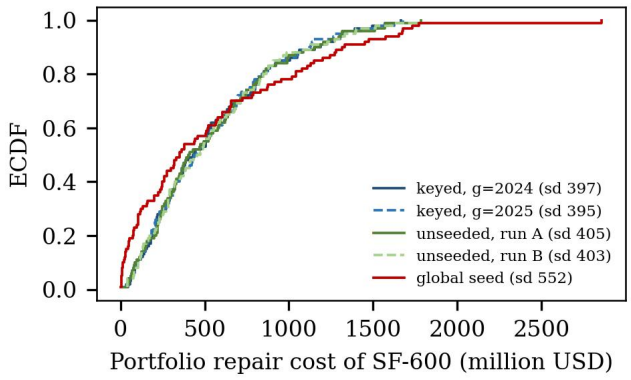


Figure 4 Empirical Distribution of the SF-600 Portfolio Repair Cost. Keyed and Unseeded Runs Are Statistically Indistinguishable; a Single Global Seed Widens the Distribution

Global seed (F3). Table 6 and Figure 4 compare five full runs of SF-600. The four runs with independent sampling (keyed with two global seeds and two unseeded runs) agree with each other (pairwise Kolmogorov–Smirnov $p \geq 0.97$) and have a portfolio standard deviation of 400 million USD on average. With one global seed, the standard deviation rises to 552 million USD (+38%), the upper quantiles shift upward, and the mean pairwise correlation of building loss ratios grows from 0.35 to 0.71; the Brown–Forsythe test rejects equal dispersion against the keyed run ($p = 0.036$). Repeatability obtained by a global seed is therefore not neutral: it adds a perfectly correlated sampling component on top of the physically meaningful ground-motion correlation, and the portfolio tail is overstated. Keyed streams provide the same repeatability without this bias.

6.2 Invariance across Execution Contexts and Recovery

Table 7 SF-120 Executed under Different Profiles with Keyed Streams. "Ident.": Buildings Whose Digest Equals the $P = 1$ Run

P	Policy	Faults / recovery	Wall (s)	Exec.	Ident.
1	dynamic	–	68.2	120	120/120
2	cyclic	–	76.8	120	120/120
4	block	–	83.0	120	120/120
8	dynamic, c=4	–	85.1	120	120/120
4	LPT	–	79.4	120	120/120
4	random	–	82.0	120	120/120
4	dynamic	p=.01, none drawn	79.9	120	120/120
4	dynamic	p=.01, none drawn, job	78.7	120	120/120
4	dynamic	p=.01, 2 fail, task	80.9	122	120/120
4	dynamic	p=.01, 2 fail, job	172.3	240	120/120
4	dynamic	p=.05 + crash .01, task	99.3	132	120/120

Table 7 lists the E3 runs of the first 120 buildings of SF-600. All 11 keyed runs, covering P from 1 to 8, the block, cyclic, dynamic, LPT and random policies, injected task failures and worker crashes, reproduce all 120 per-building digests of the reference run, and also agree with the corresponding buildings of the 600-building run. The keyed streams add no measurable cost: the median task time is 0.521 and 0.535 s for the two keyed runs of SF-600 versus 0.536 and 0.541 s for the unseeded runs; recording a task in the manifest takes 187 bytes, its SHA-256 digest 9.1 μs, and the correctly rounded aggregation of all 600×100 realizations 3.3 ms. The portfolio total aggregated with the correctly rounded sum is identical in every run (1 distinct value), whereas left-to-right summation in completion order yields 11 distinct totals with a maximum relative deviation of 7.5e-16; such results pass tier T1 but fail T0, which is precisely the distinction the reconciliation protocol is designed to make. Under a repeatable fault pattern that fails two tasks in the first pass, task-level retry completes after 122 task executions in 80.9 s, while whole-job restart, the semantics of F4, needs 240 executions and 172.3 s; with a higher fault rate ($p = 0.05$ task errors and $p = 0.01$ worker crashes) task-level recovery absorbs 12 failed attempts and 2 pool restarts at 99.3 s versus 68.2 s without faults, still with bit-identical outputs.

6.3 Sensitivity of Result Reconciliation

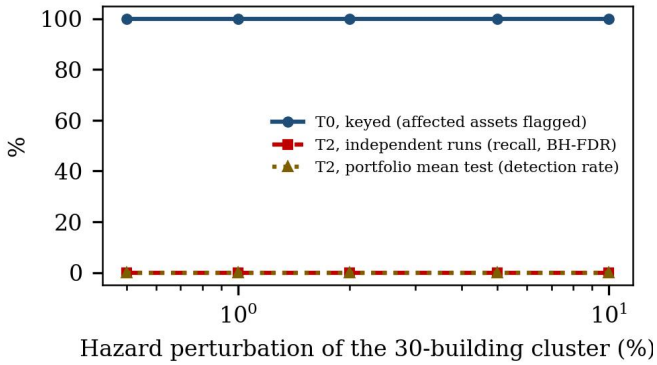


Figure 5 Detection of a Hazard Perturbation Applied to a Cluster of 30 Buildings (Mission District) out of 600. Keyed Streams Allow Exact Tier-T0 Localization; Independent Runs Require Statistical Tests (Four Trials per Level)

A regression test of a workflow must detect small, localized changes without raising false alarms on unchanged inputs. We scaled the demand of the 30 buildings closest to a point in the Mission District by $1+\delta$, $\delta \in \{0.5,1,2,5,10\}\%$, which is equivalent to lowering their fragility medians by the same factor, and reconciled the perturbed runs with the reference (Figure 5). With keyed streams the realizations of both runs are aligned, only the 30 affected buildings are recomputed, and tier T0 flags 30 of the 30 buildings already at $\delta = 0.5\%$ with no unaffected building flagged, because unaffected buildings are bitwise identical by construction; restricted to the loss realizations alone, 19 of the 30 buildings (0.7% of their realizations) change at $\delta = 0.5\%$ and 28 at $\delta = 2\%$. Without aligned realizations, tier T2 must separate the perturbation from Monte Carlo noise: in four independent trials per level, per-building Welch tests with false-discovery-rate control flag no building for any δ up to 10%, and the portfolio-level test never rejects (median $p \geq 0.96$); under the null hypothesis ($\delta = 0$) it likewise produces 0.00 false discoveries. With the 100 realizations per building of the reference configuration, a 10% demand change of a 30-building cluster is therefore invisible to statistical comparison but exactly localized under aligned realizations. Aligned realizations therefore change regression testing from a statistical decision into an exact one and localize the change to individual buildings.

6.4 Cost Structure of the Reference Execution (F5)

Table 8 Per-Building Cost of the IM Path under the Three Execution Families (SF Workloads, One vCPU)

Family	s/bldg	Interp./bldg	Files/bldg	Seeding
E0 rWHALE (native)	1.86	3	26	none / global
E2 regional_sim	0.031	0 (chunked)	0	none
E3 RSRBench executor	0.52	0 (persistent)	1	keyed

Table 8 decomposes the per-building cost. In E0 a building costs 1.86 s, of which the three application processes take 0.11, 0.16 and 1.63 s when re-executed in isolation; an empty interpreter starts in 0.013 s but importing the pelicun entry point alone takes 1.00 s, so most of the per-building time is interpreter and import overhead rather than damage and loss computation. E3 performs the identical computation in 0.52 s (3.6× less) with outputs that are bitwise equal to E0 under an equal seed. E2 amortizes setup over chunks of 1000 buildings (0.031 s per building) but provides neither seeding nor a per-building task boundary for recovery. E0 additionally writes 26 files per building, i.e., about 4.0 million files for the 149,970-building city-wide inventory.

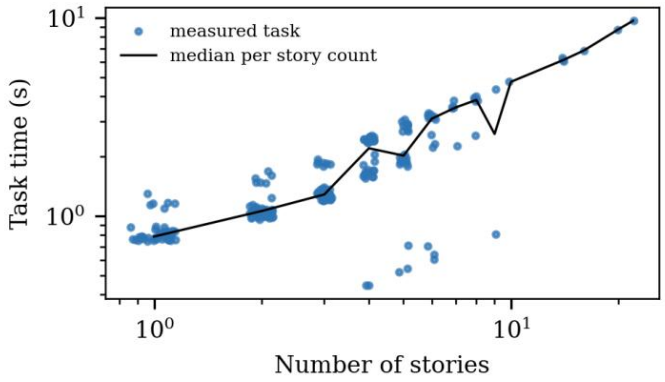


Figure 6 Measured Cost of MDOF-LU Model Generation plus Nonlinear Time-History Analysis for 220 Anchorage Buildings (One Recorded Ground Motion Each)

Table 9 Story-Count Distribution of the Anchorage Inventory and Measured Cost of One Time-History Task

Stories	Buildings	Measured	Median (s)	Max (s)
1	43,969	43	0.79	1.29
2	39,198	45	1.06	1.68
3	1,138	39	1.28	1.93
4	62	34	2.19	2.55
5–9	59	52	2.76	4.37
10–22	10	7	6.27	9.66

The time-history tier is markedly heterogeneous (Figure 6, Table 9): 98.5% of the buildings have one or two stories, but the rare tall buildings cost up to an order of magnitude more per task; the 220 measured tasks take between 0.45 and 9.7 s (median 1.28 s), growing with the story count. A log-linear cost model on story count, record intensity and length, and design era predicts task cost with cross-validated $R^2 = 0.99$ (mean absolute percentage error 4%). Applied to the 84,436-building inventory it yields 23.0 CPU-hours per ground-motion sample, with a longest task of 10 s.

6.5 Scheduling at Scale

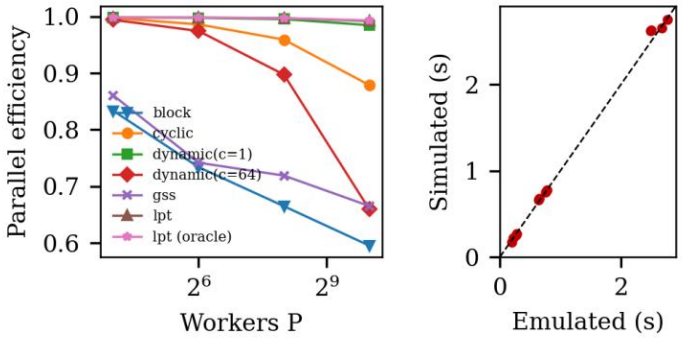


Figure 7 Left: Parallel Efficiency of the ANC-84k Time-History Tier versus Worker Count. Right: Simulator Validation against Process-Pool Emulation (12 Configurations)

Table 10 Validation of the Simulator against Process-Pool Emulation (400 Tasks Drawn from the Time-History Trace, Scaled to a Mean of 25 ms)

P	Policy	Emulated (s)	Simulated (s)	Error
4	block	2.674	2.657	-0.7%
4	cyclic	2.771	2.756	-0.6%
4	dynamic	2.501	2.632	+5.3%
4	lpt	2.495	2.624	+5.2%
16	block	0.759	0.755	-0.6%
16	cyclic	0.784	0.780	-0.6%
16	dynamic	0.648	0.676	+4.3%
16	lpt	0.638	0.660	+3.5%
64	block	0.263	0.253	-3.8%
64	cyclic	0.274	0.271	-1.1%
64	dynamic	0.224	0.218	-2.7%
64	lpt	0.199	0.171	-13.9%

The simulator was calibrated with a dispatch overhead of 1.73 ms per chunk and reproduces twelve emulated configurations ($P \in \{4,16,64\}$, four policies) with a mean absolute error of 3.5% (maximum 13.9%), Figure 7 (right) and Table 10; the largest deviation (-13.9%) occurs for LPT at $P = 64$, where only about six tasks fall on each worker and the makespan is set by a few long tasks, and at $P = 4$ the dynamic policies are overestimated by about 5% because the calibrated per-chunk overhead is conservative at low concurrency. For ANC-84k in inventory order (Figure 7, left), static partitions lose efficiency as P grows because tall buildings are few, expensive and spatially clustered: at $P = 1024$ the static block and cyclic partitions reach 0.60 and 0.88, dynamic self-scheduling 0.98, GSS 0.67, and LPT with the cost model 0.99, close to LPT with exact costs (0.99); chunked self-scheduling with 64 tasks per chunk, as used by batch-oriented executors, drops to 0.66. The makespan drops from 1.5 to 1.4 minutes. LPT degrades gracefully when the cost model is perturbed: with multiplicative log-normal errors of $\sigma = 0.5$, 1 and 2 the efficiency at $P = 1024$ is 0.98, 0.99 and 0.97. For the homogeneous IM tier of the city-wide SF-150k inventory, static cyclic partitioning already reaches 0.97 at $P = 1024$ (dynamic: 0.99); there the dominant lever is the per-building overhead of Table 8, which raises the total cost from 22.1 CPU-hours with persistent workers to 68.0 CPU-hours with the native process model.

6.6 Failures, Retries and Recovery at Scale

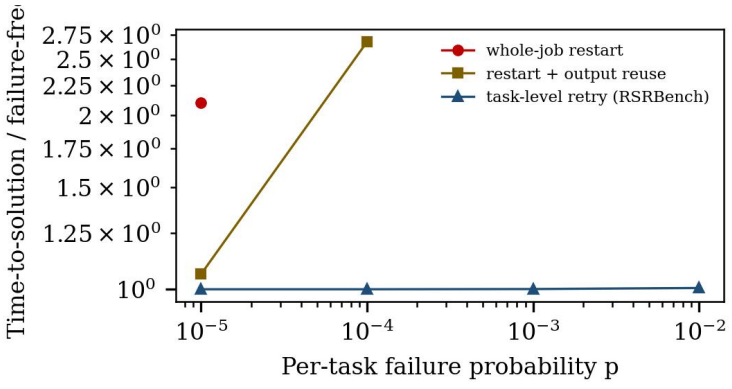


Figure 8 Time-to-Solution of ANC-84k on 256 Workers Relative to the Failure-Free Makespan under Per-Task Failure Probability p (60 s Restart Cost; Missing Points: No Completion within 60 Restarts)

Figure 8 extends the measured behavior to ANC-84k on 256 workers. Whole-job restart is practical only while the expected number of failures per run is well below one: at $p = 10^{-5}$ it already needs 2.10× the failure-free time (5.4 minutes), and from $p = 10^{-4}$ on it does not complete within 60 restarts (at $p = 10^{-3}$, about 84 failures are expected per pass). Reusing completed outputs after an abort (skip) still completes at $p = 10^{-4}$, at 2.68× the failure-free time after 9 restarts, but not at $p = 10^{-3}$. Task-level retry keeps the overhead at 0.09% for $p = 10^{-3}$ and 0.5% for $p = 10^{-2}$ because each failure costs only the elapsed part of one task; with keyed streams the recovered result is, in addition, bitwise identical to the failure-free result, so recovery can be verified rather than assumed.

6.7 Ablation

Table 11 Ablation of the RSRBench Execution Rules (Measured Unless Marked S = Simulated)

Variant	Repeat.	Unbiased	Recovery	Eff. (S)
Full rules	120/120	σ 400, $\bar{\rho}$ 0.35	80.9 s	0.99
−keyed (unseeded)	0/600	yes	unverifiable	–
−keyed (global)	yes	σ +38%, $\bar{\rho}$ 0.71	–	–
−exact sum	totals: 11 values	yes	–	–
−task recovery	yes	yes	172.3 s; n.c. at 10^{-3} (S)	–
−cost model	yes	yes	–	0.98

Table 11 removes one rule at a time. Without keyed streams, runs are either not repeatable (unseeded) or biased (global seed); without exact aggregation, per-building outputs remain identical but portfolio totals differ in the last bits between schedules; without task-level recovery, two task failures multiply the measured time-to-solution of SF-120 by 2.13and the job stops completing at scale; without the cost model, dynamic self-scheduling stays close to LPT (0.98 vs. 0.99) while the static cyclic default loses more than ten points (0.88). Each rule therefore addresses a distinct failure mode, and the combination is required for results that are simultaneously repeatable, unbiased, verifiable and efficient.

7 DISCUSSION

The audit and the benchmark lead to concrete recommendations for regional seismic risk workflows. First, every random draw should be keyed by asset and stage rather than consumed from a shared stream; in the reference workflow this needs one local change in the mapping application and a per-asset value in an existing configuration field, and it removes the process-count dependence of F1 without any runtime cost. Second, a single global seed should not be used as a reproducibility fix, because it couples the sampling of all assets; for SF-600 it inflates the portfolio standard deviation by 38%, an error of the same order as the effect of modeling ground-motion correlation itself [12]. Third, regional totals should be aggregated in canonical order with a correctly rounded sum so that bitwise comparison remains meaningful at the portfolio level. Fourth, workflow drivers should treat each asset as an idempotent task with a recorded digest; this turns failures into local retries and makes the completeness check of the reconciliation protocol (missing assets, F4) automatic. Finally, the cost of time-history tasks is predictable from inventory attributes, and a simple cost model suffices for near-optimal LPT scheduling at a thousand workers, while for intensity-based assessments the per-asset interpreter and import overhead is the dominant performance factor.

RSRBench is designed to grow with the workflows it measures. The same workload and reconciliation definitions apply to other asset types and hazards supported by the SimCenter applications, to multi-sample (“possible worlds”) studies, and to new releases of the damage and loss model library, for which the tiered protocol separates intended model updates (divergent at T2 with an explained cause) from unintended numerical drift (T1) and from execution effects (T0).

8 CONCLUSION

We presented RSRBench, a reproducible benchmarking framework for large-scale regional seismic risk simulation that drives the public NHERI SimCenter workflows on two public testbeds under controlled execution profiles. An execution-level audit showed that, with unchanged inputs and seeds, the hazard mapping changes for 598 of 600 buildings when only the MPI process count changes, that identical runs never reproduce damage and loss realizations, and that the obvious remedy of a global seed inflates the portfolio-loss dispersion by 38%. Keyed-stream execution with order-independent aggregation and idempotent tasks removes these effects with an 11-line patch and seed injection through the existing configuration, yielding bit-identical results across process counts, schedules, failures and retries, and enabling exact, building-level localization of regressions as small as 0.5%. Measured traces and a validated simulator further show that dynamic, cost-aware scheduling and task-level recovery are required for efficient and robust

execution of time-history workloads at a thousand workers. The benchmark, patches, raw measurements and analysis scripts are released to support verifiable comparisons of regional risk workflows and their execution strategies.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Federal Emergency Management Agency. Hazus Earthquake Model Technical Manual, Hazus 6.1. Washington, DC: FEMA, 2024.

[2] Kircher C A, Whitman R V, Holmes W T. HAZUS Earthquake Loss Estimation Methods. Natural Hazards Review, 2006, 7(2): 45-59. DOI: 10.1061/(ASCE)1527-6988(2006)7:2(45).

[3] Federal Emergency Management Agency. Seismic Performance Assessment of Buildings, Volume 1 – Methodology, Second Edition (FEMA P-58-1). Washington, DC: FEMA, 2018.

[4] Lu X, McKenna F, Cheng Q, et al. An Open-Source Framework for Regional Earthquake Loss Estimation Using the City-Scale Nonlinear Time History Analysis. Earthquake Spectra, 2020, 36(2): 806-831.

[5] Deierlein G G, Zsarnóczay A, Singh K, et al. A Cloud-Enabled Application Framework for Simulating Regional-Scale Impacts of Natural Hazards on the Built Environment. Frontiers in Built Environment, 2020, 6: 558706. DOI: 10.3389/fbuil.2020.558706.

[6] Rathje E M, Dawson C, Padgett J E, et al. Enhancing Research in Natural Hazards Engineering through the DesignSafe Cyberinfrastructure. Frontiers in Built Environment, 2020, 6: 547706. DOI: 10.3389/fbuil.2020.547706.

[7] Zsarnóczay A, NHERI SimCenter. pelicun: Probabilistic Estimation of Losses, Injuries, and Community Resilience under Natural Hazard Events, Version 3.8.1. Software release, 2025. NHERI SimCenter. Regional Resilience Determination Tool (R2D), Version 5.5.0 Release Notes. 2025.

[8] Silva V, Crowley H, Pagani M, et al. Development of the OpenQuake Engine, the Global Earthquake Model's Open-Source Software for Seismic Risk Assessment. Natural Hazards, 2014, 72(3): 1409-1427. DOI: 10.1007/s11069-013-0618-x.

[9] Xiong C, Lu X, Lin X, et al. Parameter Determination and Damage Assessment for THA-Based Regional Seismic Damage Prediction of Multi-Story Buildings. Journal of Earthquake Engineering, 2017.

[10] Jayaram N, Baker J W. Correlation Model for Spatially Distributed Ground-Motion Intensities. Earthquake Engineering & Structural Dynamics, 2009, 38(15): 1687-1708.

[11] Weatherill G, Silva V, Crowley H, et al. Exploring the Impact of Spatial Correlations and Uncertainties for Portfolio Analysis in Probabilistic Seismic Loss Estimation. Bulletin of Earthquake Engineering, 2015, 13(4): 957-981. DOI: 10.1007/s10518-015-9730-5.

[12] Peng R D. Reproducible Research in Computational Science. Science, 2011, 334(6060): 1226-1227. DOI: 10.1126/science.1213847.

[13] Stodden V, McNutt M, Bailey D H, et al. Enhancing Reproducibility for Computational Methods. Science, 2016, 354(6317): 1240-1241. DOI: 10.1126/science.aah6168.

[14] Hoefler T, Belli R. Scientific Benchmarking of Parallel Computing Systems: Twelve Ways to Tell the Masses When Reporting Performance Results. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC15), 2015: 73:1-73:12. DOI: 10.1145/2807591.2807644.

[15] Salmon J K, Moraes M A, Dror R O, et al. Parallel Random Numbers: As Easy as 1, 2, 3. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC11), 2011: 16:1-16:12.

[16] L'Ecuyer P. Multiple Streams with Recurrence-Based, Counter-Based, and Splittable Random Number Generators. Proceedings of the 2021 Winter Simulation Conference (WSC), 2021.

[17] Harris C R, Millman K J, van der Walt S J, et al. Array Programming with NumPy. Nature, 2020, 585(7825): 357-362. DOI: 10.1038/s41586-020-2649-2.

[18] Shewchuk J R. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. Discrete & Computational Geometry, 1997, 18(3): 305-363. DOI: 10.1007/PL00009321.

[19] Demmel J, Nguyen H D. Fast Reproducible Floating-Point Summation. Proceedings of the 21st IEEE Symposium on Computer Arithmetic (ARITH), 2013: 163-172. DOI: 10.1109/ARITH.2013.9.

[20] Graham R L. Bounds on Multiprocessing Timing Anomalies. SIAM Journal on Applied Mathematics, 1969, 17(2): 416-429. DOI: 10.1137/0117039.

[21] Polychronopoulos C D, Kuck D J. Guided Self-Scheduling: A Practical Scheduling Scheme for Parallel Supercomputers. IEEE Transactions on Computers, 1987, C-36(12): 1425-1439. DOI: 10.1109/TC.1987.5009495.

[22] Young J W. A First Order Approximation to the Optimum Checkpoint Interval. Communications of the ACM, 1974, 17(9): 530-531.

[23] Daly J T. A Higher Order Estimate of the Optimum Checkpoint Interval for Restart Dumps. Future Generation Computer Systems, 2006, 22(3): 303-312. DOI: 10.1016/j.future.2004.11.016.

[24] NHERI SimCenter. R2D User Manual: Regional Earthquake Assessment Examples (E1 Basic HAZUS and E2 MDOF Building Response). R2D v5.5 Documentation, 2025.
[25] McKenna F. OpenSees: A Framework for Earthquake Engineering Simulation. Computing in Science & Engineering, 2011, 13(4): 58-66. DOI: 10.1109/MCSE.2011.66.
[26] Benjamini Y, Hochberg Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society: Series B, 1995, 57(1): 289-300. DOI: 10.1111/j.2517-6161.1995.tb02031.x.
[27] Zhu M, McKenna F, Scott M H. OpenSeesPy: Python Library for the OpenSees Finite Element Framework. SoftwareX, 2018, 7: 6-11. DOI: 10.1016/j.softx.2017.10.009.