Batching Stata Monte Carlos: Memory-Safe, Resume-Friendly, and Parallel

Yunhan (Max) Liu Matthew D. Webb

Carleton University

2025 Canadian Stata Conference — Poster Presentation

TL;DR — What This Does

- **Problem.** Large Stata Monte Carlo runs crash or crawl as memory accumulates across replications.
- Single core (or n-core) license for Stata, throttles CPU usage.
- **Idea.** Partition total replications R into B batches of r each ($R = B \times r$). Launch each batch in a *fresh Stata session* via a shell script; run sequentially or concurrently.
- **Why it works.** Fresh process $\Rightarrow$ clean heap and no RAM accumulation; multiple processes $\Rightarrow$ OS-level parallelism (even with Stata/SE).
- **Bonus.** *Crash-resume:* re-run the launcher; only unfinished batches execute.

Motivation — Real Case

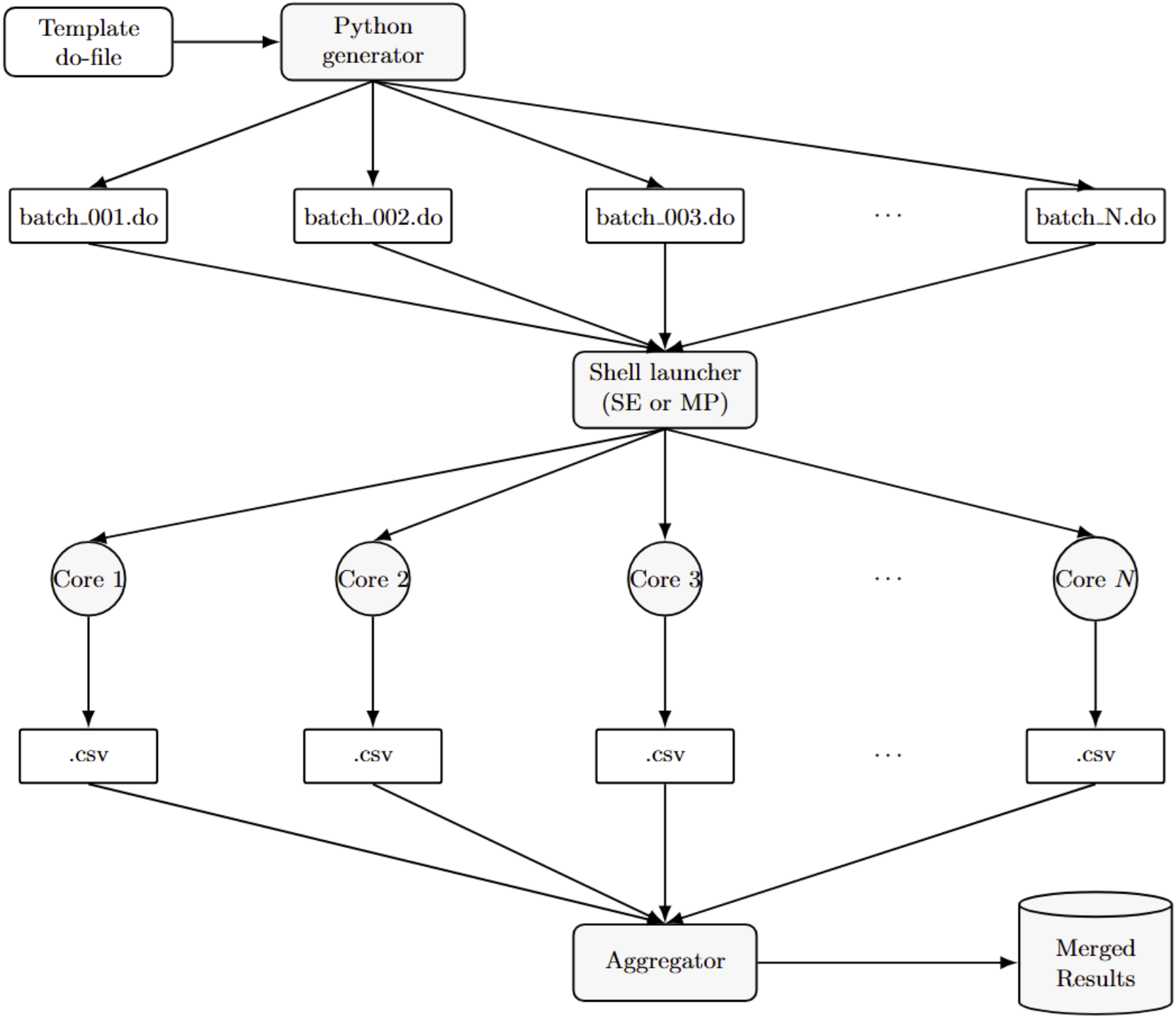
- For simulations in MacKinnon et al. [2025], to test Liu [2025] we needed ≈11 parameter settings × ≈2,000 reps ⇒ ≈22,000 trials.
  - Each replication took up to 30 minutes.
  - The simulations involved a brute force jackknife of Callaway and Sant'Anna [2021], unlike the fast jackknife in MacKinnon et al. [2023].
  - Each DGP could take up to 2 months of CPU time on a single core for only 2500 replications.
- Manual multi-core attempts: later batches slowed; some machines failed.
- **Need:** memory-safe, parallel, *and* resume-friendly workflow that runs on modest hardware.

What People Try (and Why It Fails)

- **Manual batching across do-files:** hand edits, seed/file-name typos, painful restarts.
- **"Intermediate" approach:** loop over batches *inside one Stata session*; Stata never exits between batches $\Rightarrow$ memory creeps.
- **Proposed:** close Stata after each batch; next batch opens in a *new* Stata instance.

Workflow Overview (Simple Division of R)

1. **Template do-file** (author once): mark fields varying by batch (IDs, rep range, parameters).
2. **Python generator** (edit minimally): reads the template; emits B batch do-files with correct indices/paths and deterministic seeds.
3. **Shell launcher** (no edits): determines number of cores available and chips away at the uncompleted set of batch do-files.
4. **Declare SE or MP:** pass one flag; multiple `SE` processes can still saturate multiple cores.
5. **Run & watch:** each batch in a fresh process; memory resets automatically.
6. **Append/Merge:** per-batch .DTAs $\rightarrow$ one results file.


Template do-file
Python generator
batch_001.do
batch_002.do
batch_003.do
· · ·
batch_N.do
Shell launcher (SE or MP)
Core 1
Core 2
Core 3
· · ·
Core N
.csv
.csv
.csv
· · ·
.csv
Aggregator
Merged Results

Crash–Resume (Key Benefit)

- If power fails or a run is interrupted, simply re-run the launcher.
- Completed batch do-files are automatically moved to `done_dofiles`.
- Files remaining in `generated_dofiles/` are the only ones executed.
- *No manual bookkeeping, or tedious forvalues updates.*

For Stata/SE Users — Why This Still Parallels

- Stata/SE is single-core *per process*. We launch multiple processes concurrently using stata's batch mode; the OS schedules them on different cores.
- Net effect: near-linear speedup up to core count (I/O permitting).
- Works via Git Bash/WSL/PowerShell.
- Increase CPU utilization from 8.5% to over 90%

Seed Discipline (Reproducibility)

- Can specify unique seeds for all batch files in advance to ensure reproducibility.
- Requires storing seeds in a separate file before hand
- This can be simplified using Stata's `set rngstream` command

Memory Reset (Mechanism)

- Fresh process = fresh heap; no cumulative objects across batches.
- Eliminates "last batch is slowest" pathology in long single sessions.
- Robust to occasional leaks in user code, because each batch starts clean.

Tuning Knobs (Pick B, r Quickly)

- **Batch size r:** choose so peak RAM $<$ 50–60% of available memory.
- **Number of batches B:** for a given number of reps, more batches allows for frequent output.
- **Cores vs threads:** Further experimentation is needed
- **Concurrency cap:** set a number of cores free so the machine remains usable.
- Prefer local SSD for logs/CSVs; avoid network drive contention.

Limitations / Trade-offs

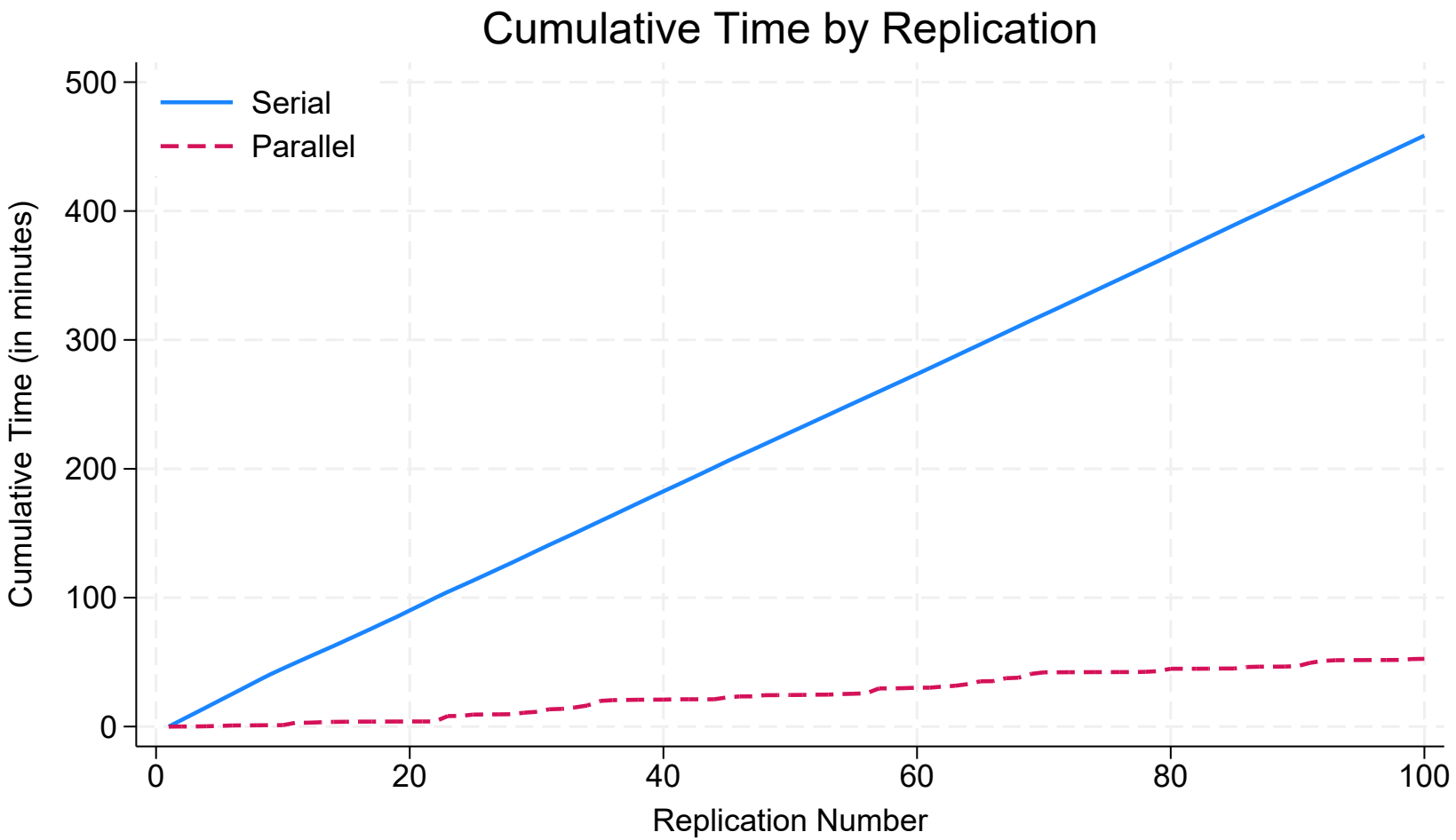
- Requires **Python** (batch generator).
- Requires a **shell** (Bash).
- **Multi-file touchpoints:** edit `template.do`, generator config, and append/merge script.
- **System saturation risk:** too many concurrent batches can make the computer feel unusable.

Mitigations: prebuilt generators; fixed, no-edit launcher; concurrency cap; compact per-rep output.

Quick Simulation Results

- **Setup** To illustrate the problem, we ran 100 replications of a modern difference-in-difference estimator
- **Parallel Setup:** Used a Machine with 16 cores, 24 logical process (requested 22 of them).
- **Serial:** 100 reps took 458 minutes in total, or 4.63 minutes on average
- **Parallel:** 100 reps took 52.5 minutes in total, or 0.53 minutes on average

Timing Comparison



Practical Checklist

- Set R, B, r, S_0 and output folder.
- Write **one** `template.do` with clear placeholders.
- Run generator $\rightarrow$ create batch DO files.
- Launch with SE/MP flag and concurrency cap.
- Append CSVs/DTAs $\rightarrow$ analyze results.

FAQ

- **Do I need Stata-MP?** No—SE works; parallelism is via multiple processes.
- **Different OS support?** Yes — through Git Bash.
- **Huge logs?** One log per batch; rotate/compress if needed.
- **Partial reruns?** Yes—only unfinished batches execute on relaunch.

Takeaways

- **Memory-safe:** fresh Stata per batch prevents RAM accumulation.
- **Resume-friendly:** crash? Re-launch runs only what's left.
- **Portable parallelism:** multi-core speedups without add-ons.

Future Work

- Additional massive Monte Carlos for Karim et al. [2025] which uses the estimators in Karim et al. [2024] and Karim and Webb [2024]
- Generalizing code for easier adoption
- Experimenting with optimal number of cores requested

Example Code

- Code for a very simple example is available at `https://github.com/liu-yunhan/Batched-Monte-Carlos`
- Note there are 6 files:
  - template.do and create_replications.py - *modify these*
  - 1.run_do_files, MP, and SE - *shell scripts, no need to edit*
  - readme
- This code just calculates 100 sample means across 100 instances of Stata

References I

Brantly Callaway and Pedro HC Sant'Anna. Difference-in-differences with multiple time periods. *Journal of Econometrics*, 225(2):200–230, 2021.

Sunny Karim and Matthew D Webb. Good controls gone bad: Difference-in-differences with covariates. *arXiv preprint arXiv:2412.14447*, 2024.

Sunny Karim, Matthew D Webb, Nichole Austin, and Erin Strumpf. Difference-in-differences with unpoolable data. *arXiv preprint arXiv:2403.15910*, 2024.

Sunny Karim, Matthew D. Webb, Nichole Austin, and Erin Strumpf. Which policy works and where? estimation and inference of state level treatment effects using difference-in-differences. Mimeo, 2025.

Yunhan Liu. csdidjack: Cluster jackknife inference for callaway and sant'anna difference-in-differences (stata package). https://github.com/liu-yunhan/csdidjack, 2025. GitHub repository; Stata package.

References II

James G. MacKinnon, Morten Ø. Nielsen, and Matthew D. Webb. Fast and reliable jackknife and bootstrap methods for cluster-robust inference. *Journal of Applied Econometrics*, 38:671–694, 2023.

James G. MacKinnon, Morten Ørregaard Nielsen, Matthew D. Webb, and Sunny Karim. Improving inferences for callaway and sant'anna DiD using the cluster jackknife. Mimeo, 2025.