Overview

Relational databases for lattice data analysis
Daniel C. Hackett

Automated analysis becomes straightforward
Can keep metadata (e.g. code version, run date) associated without it getting in the way.
Reproducibility & provenance
- working with an incomplete dataset, using outdated results, etc.
and results live in one place. [Prevents errors converting formats!]
Enforces standardization for collaborations
- No black boxes: structure DB so that analysis can be traced from raw data to final output

Abstract Structure

MCMC
- Map: from gauge config to next gauge config
- Ensemble: ~ config generated with same physical & MCMC parameters

Gradient Flow
- Map: from gauge config to measurements of flowed observables like $t^2E(t)$
- Scale Estimation: Reduce from ensemble of measurements to estimates of observables

Make equilibration cut, average observables, solve e.g. $(t^2E(t)) = M$

Smart Updating with Hashes

To start: assign a unique output_hash to each gauge config [e.g. checksum, gauge_config]
Every result has an input_hash and an output_hash.
input_hash: Hash together output_hashes of all inputs & map_ids/reduce_ids
output_hash: Hash together all outputs of a measurement/reduction with the input_hash
Compute hashes outwards from gauge_config to determine which analyses need update
input_hash changes → something upstream has changed – analysis needs update
output_hash changes but input_hash does not → analysis has been corrupted – easy integrity checking

Relational Databases

In practice, this means: SQL databases
We use PostgreSQL [open source!]

Data is stored in tables
Key columns provide unique specifiers for each entry
Tables can refer to each other's key columns
- Use this to encode structure in data

Must specify table structure and relations at initialization.
After, any data put in must conform to these types and relations [Feature, not a bug: enforces conventions!]

Retrieve data from DB using SQL query language

Example Lattice Database

Gauge configs with output_hashes:

Flow measurements with input_hashes and output_hashes

New flow measurement with its own output_hash

Output hash from new measurement added to input pool. Input hash changes, results need update

Automation

Run "bulk analysis" scripts N times daily:
- Check for newly-generated raw data and sync to DB
- Determine which analyses need to be updated/new analyses to perform
- Perform necessary analyses (e.g. computing scales, fitting correlators, picking best fits)
No human intervention required to keep analysis up-to-date (wake up to fresh results every morning)

Closing the automation loop:
- Workflow manager runs simulations [e.g. taxi github.com/dchackett/taxi]
- Data stored and automatically analyzed in database
- Automated run-specifier looks at analysis in DB, tells workflow manager to launch new simulations
- Applications:
  - Tuning lattice spacing, $m_a$, etc
  - Phase diagram exploration (ask about video!)