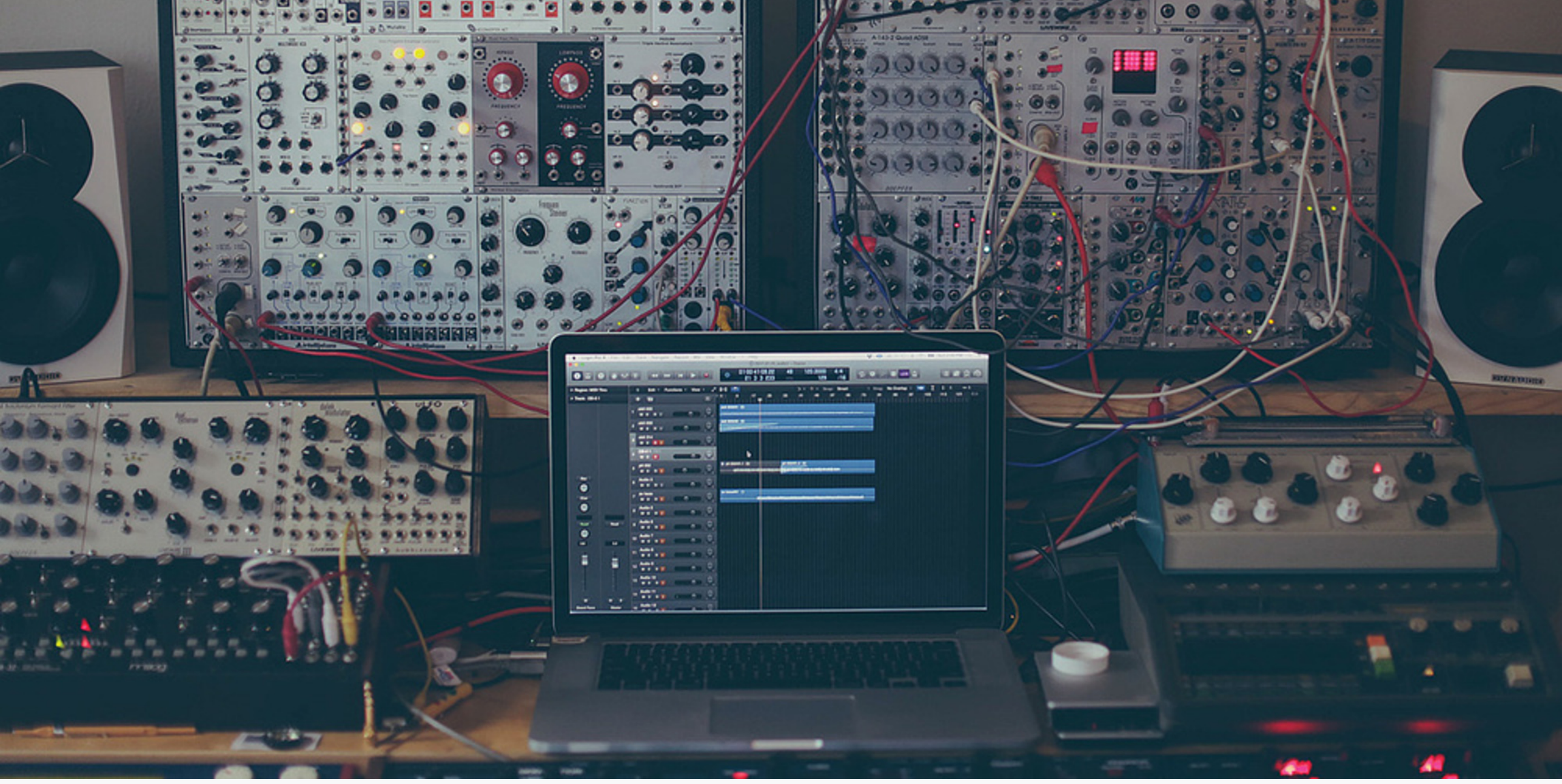


Hadrons, a Grid-based workflow management system for lattice field theory simulations

Antonin Portelli

4th of November 2020

R-CCS seminar/tutorial

THE UNIVERSITY *of* EDINBURGH

Grid: a data parallel C++ mathematical object library

https://github.com/paboyle/Grid

https://arxiv.org/abs/1512.03487

The Grid library

- Free (GPLv2) data parallel C++11 library.
  https://github.com/paboyle/Grid

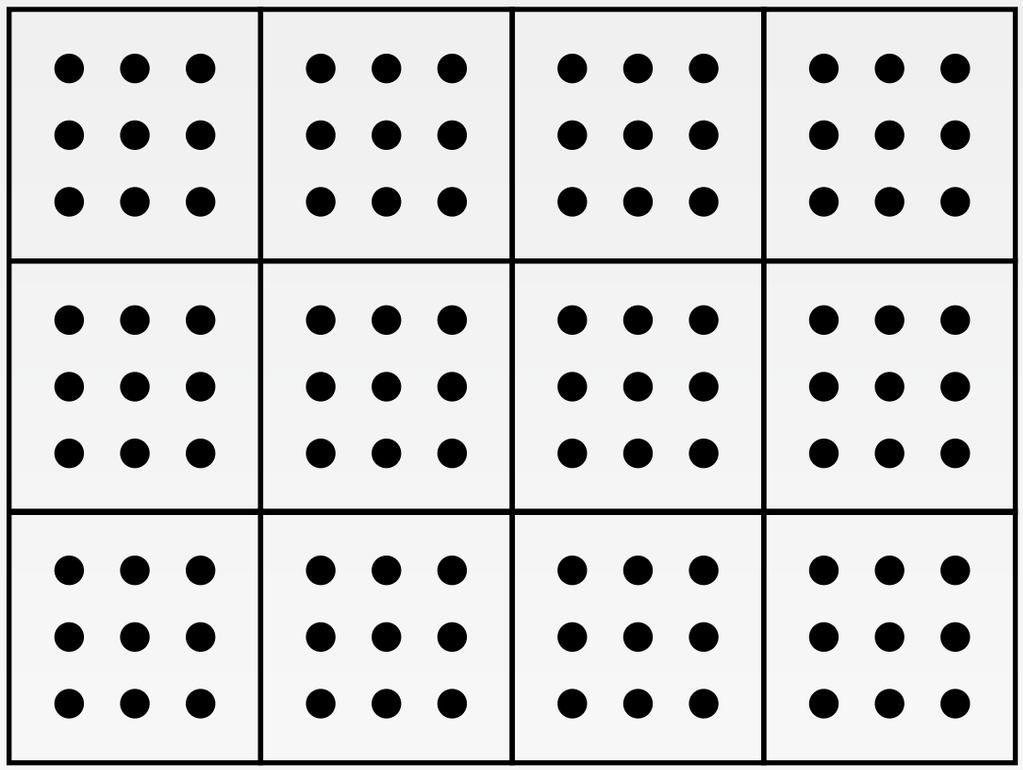
- Multi-platform, most code platform-agnostic.
  SSE, AVX, AVX2, AVX512, QPX, NEONv8, NVIDIA,
  AMD GPUs (experimental)

- Implements popular lattice fermion actions
  (Wilson, DWF, Staggered, …)

- Implements many solvers
  (CG (many flavours), multi-grid CG, Lanczos, …)

- Implements full HMC/RHMC interface

Grid lattice layout



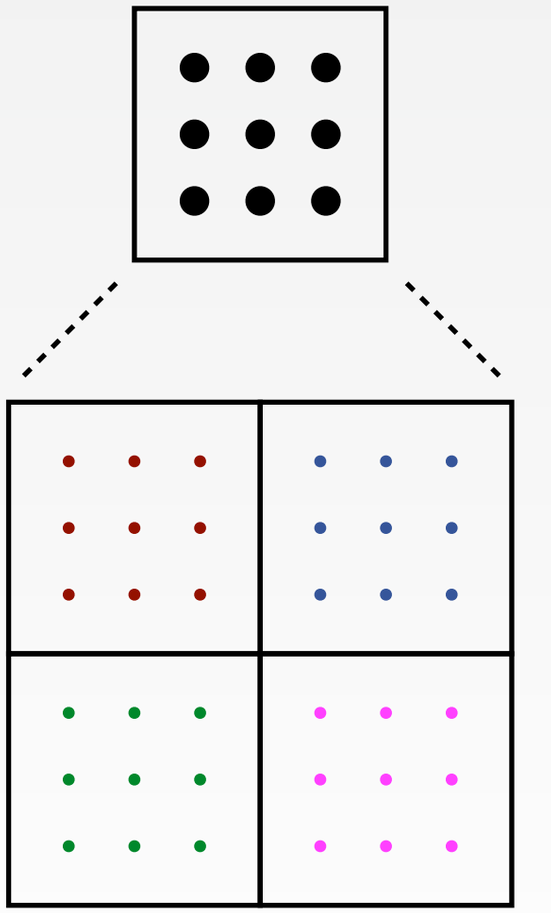
MPI Cartesian layout

High-efficiency halo exchange

Shared buffer and multi-endpoint comms

● = [· · · ·]

SIMD/SIMT vector

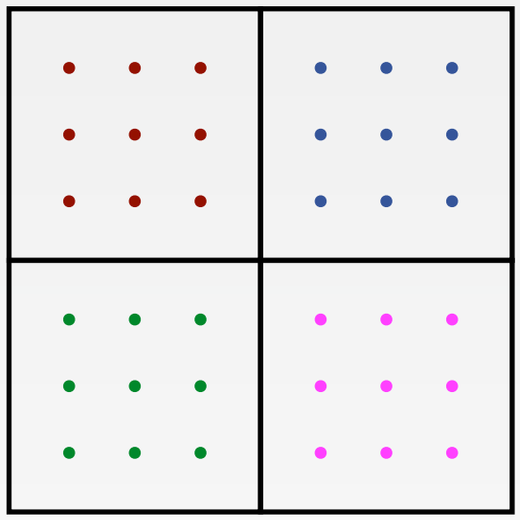


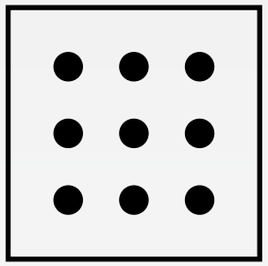
Vectorised layout

Explicit examples

6x6 lattice - AVX 256bit SIMD

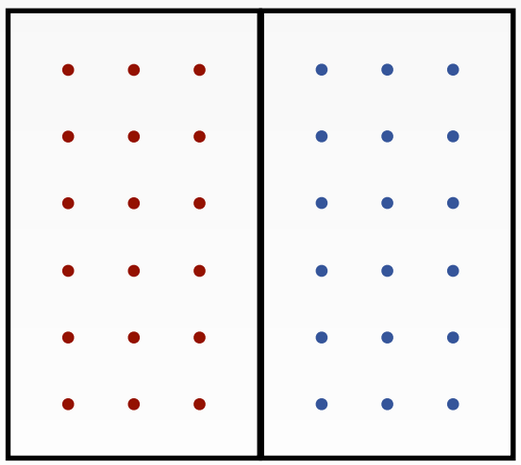
Lattice of `double`

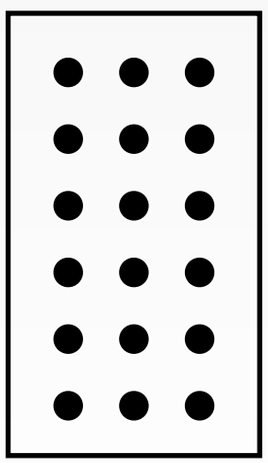




• = [· · · ·] ~ `__mm256d`

Lattice of `std::complex<double>`





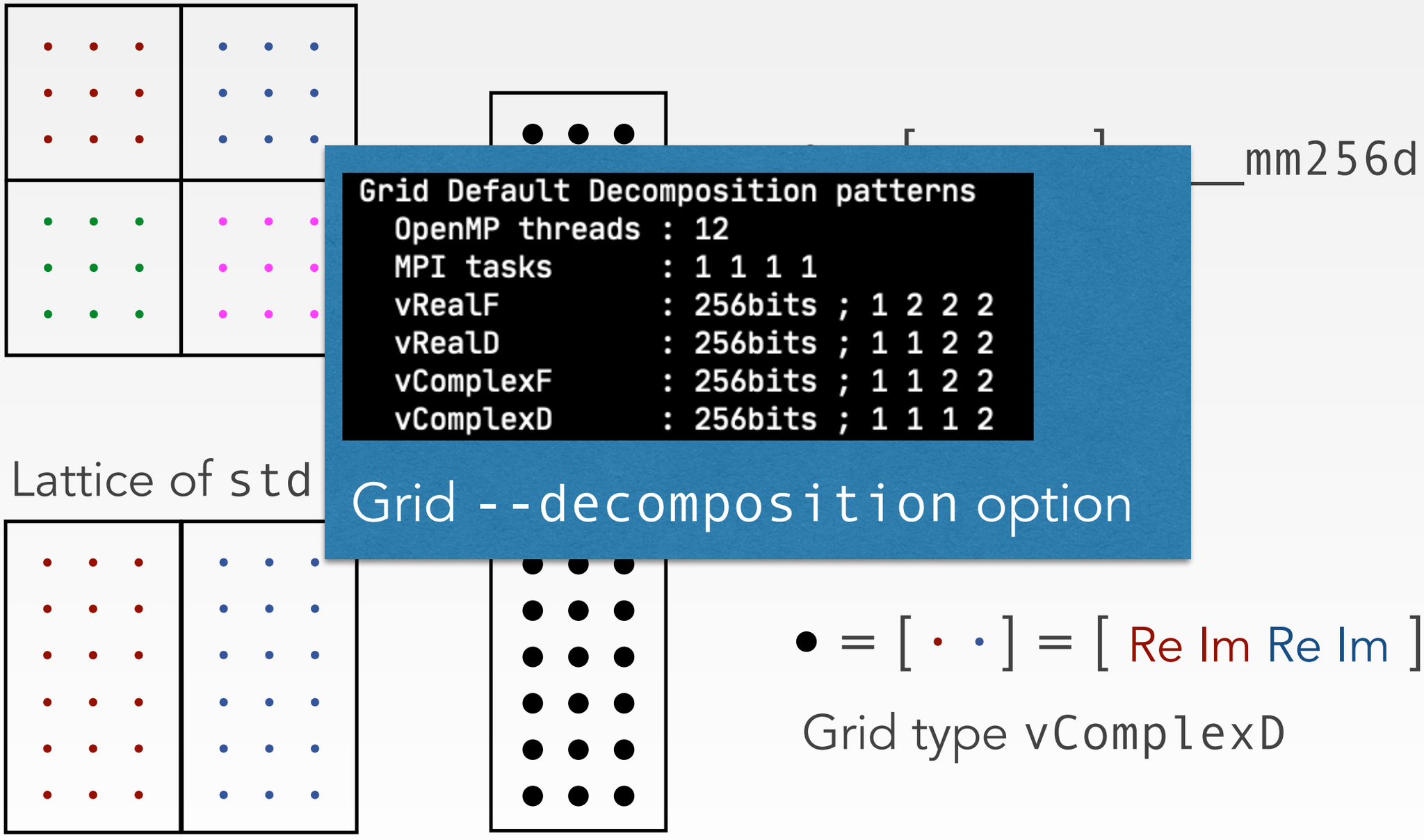
• = [· ·] = [Re Im Re Im]

Grid type `vComplexD`

Explicit examples

6x6 lattice - AVX 256bit SIMD

Lattice of `double`

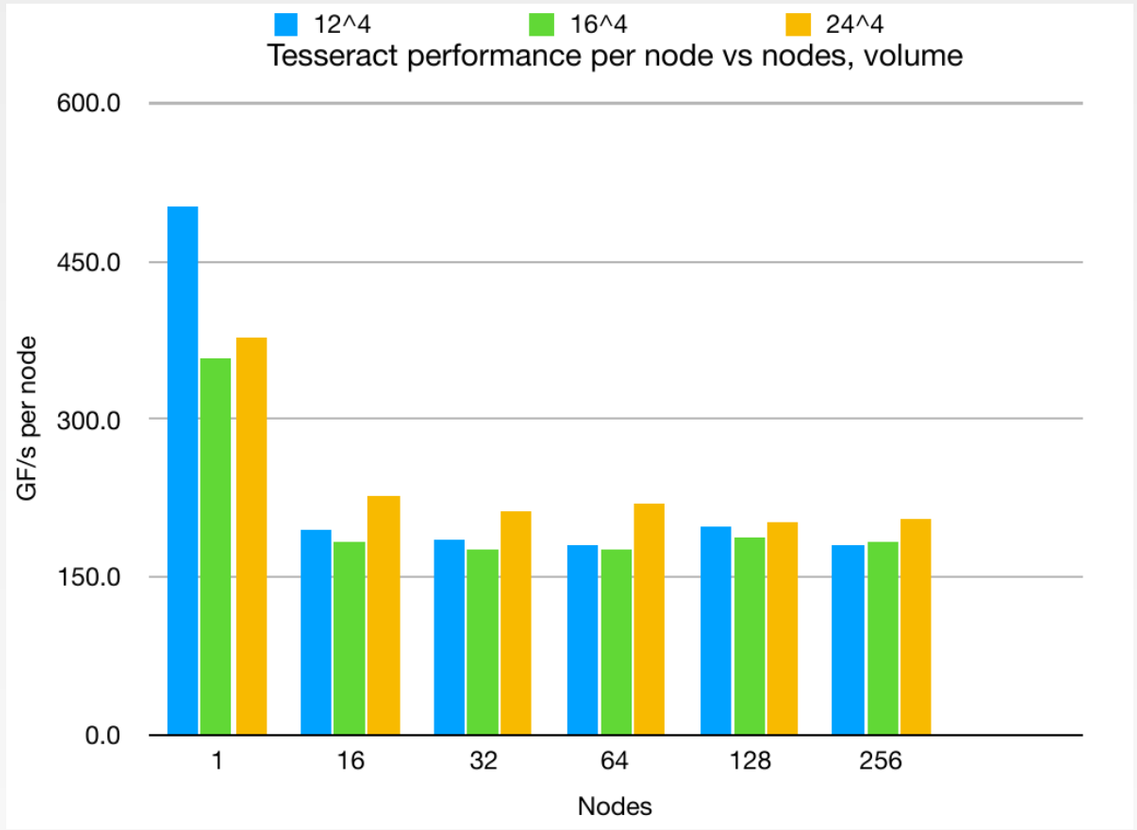


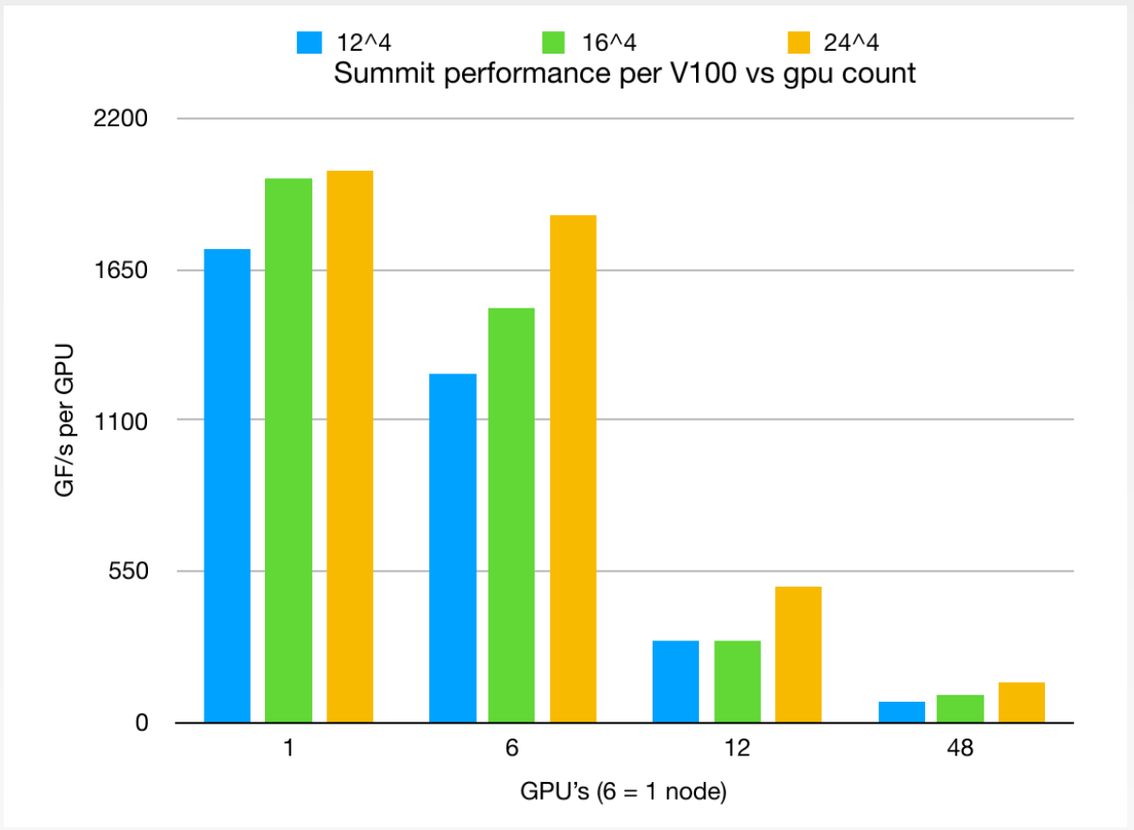
Grid lattice expressions

```
C = tr(g5*gSnk*q1*adj(gSrc)*g5*adj(q2));
```

- C++ **expression template** engine
- Site-wise operation **automatically parallelised**
- **100% vectorised** thanks to vector layout
- Loops over sites multi-threaded
- **Symbolic gamma matrix** algebra
- High-level **circular shift** operator & **stencil** interfaces

Performances





Grid single precision Dslash, [P. Boyle, USQCD All-Hands Collaboration Meeting 2019]

- DiRAC Extreme Scaling (Tesseract):
  hypercubic network topology (HPE SGI-8600 blades)

Hadrons: a Grid-based workflow management system

https://github.com/aportelli/Hadrons

https://doi.org/10.5281/zenodo.4063666

Lattice measurements

- In QCD basically:
  Solver - Propagators - Contractions

- More and more involved:
  Deflation, LMA, distillation, n-pt functions…

Things I did not want to repeat

(no hard feelings, just trying to improve 🙂)

- **Very complicated inputs.**
  (100k lines XML files, machine generated inputs)

- **Very rigid programs.**
  (lots of global variables scattered in the program)

- **No safety net.**
  (dependency between steps, memory consumption)

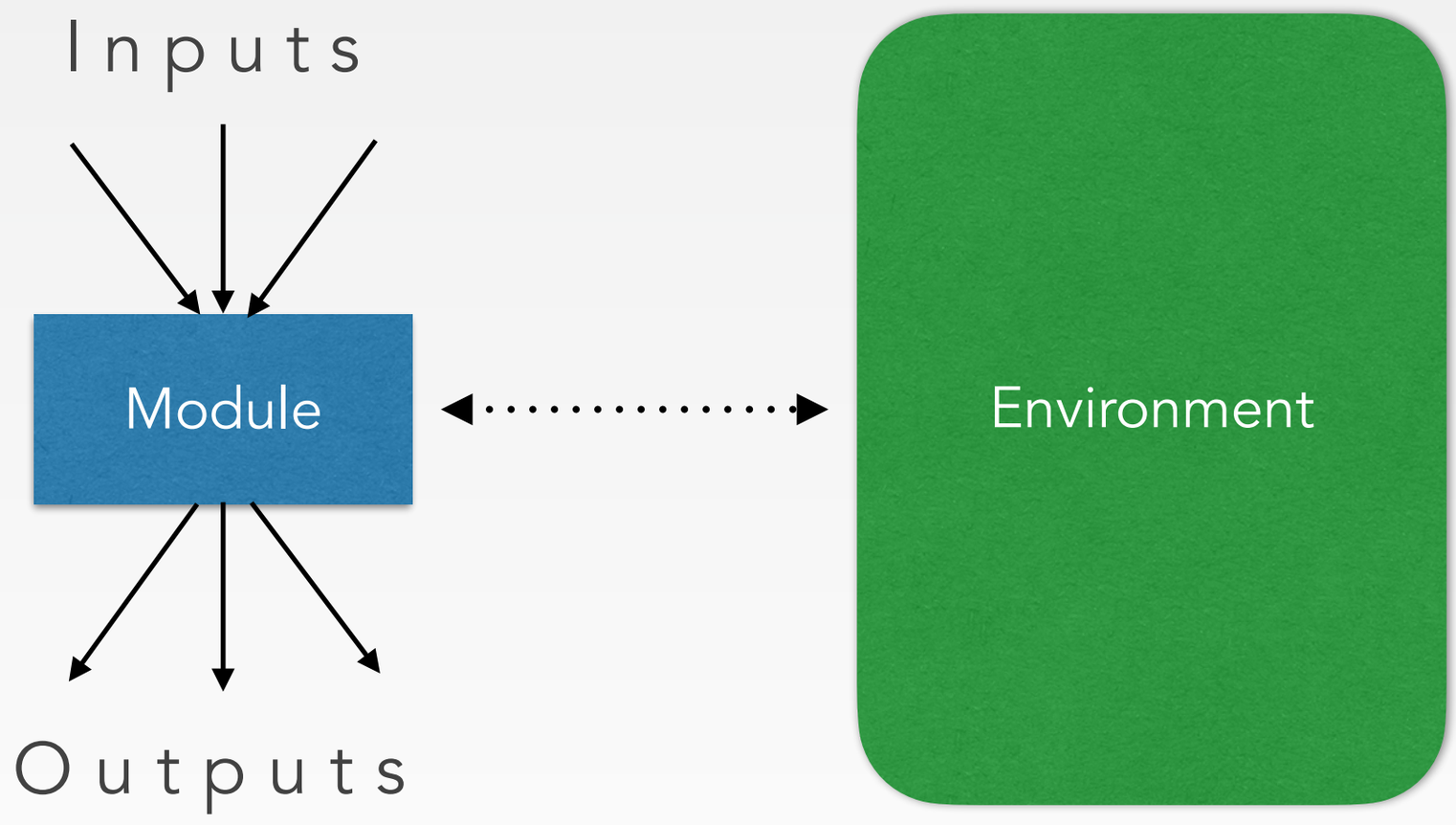
Directions for solutions

- **High modularity** — building a new project is easy.

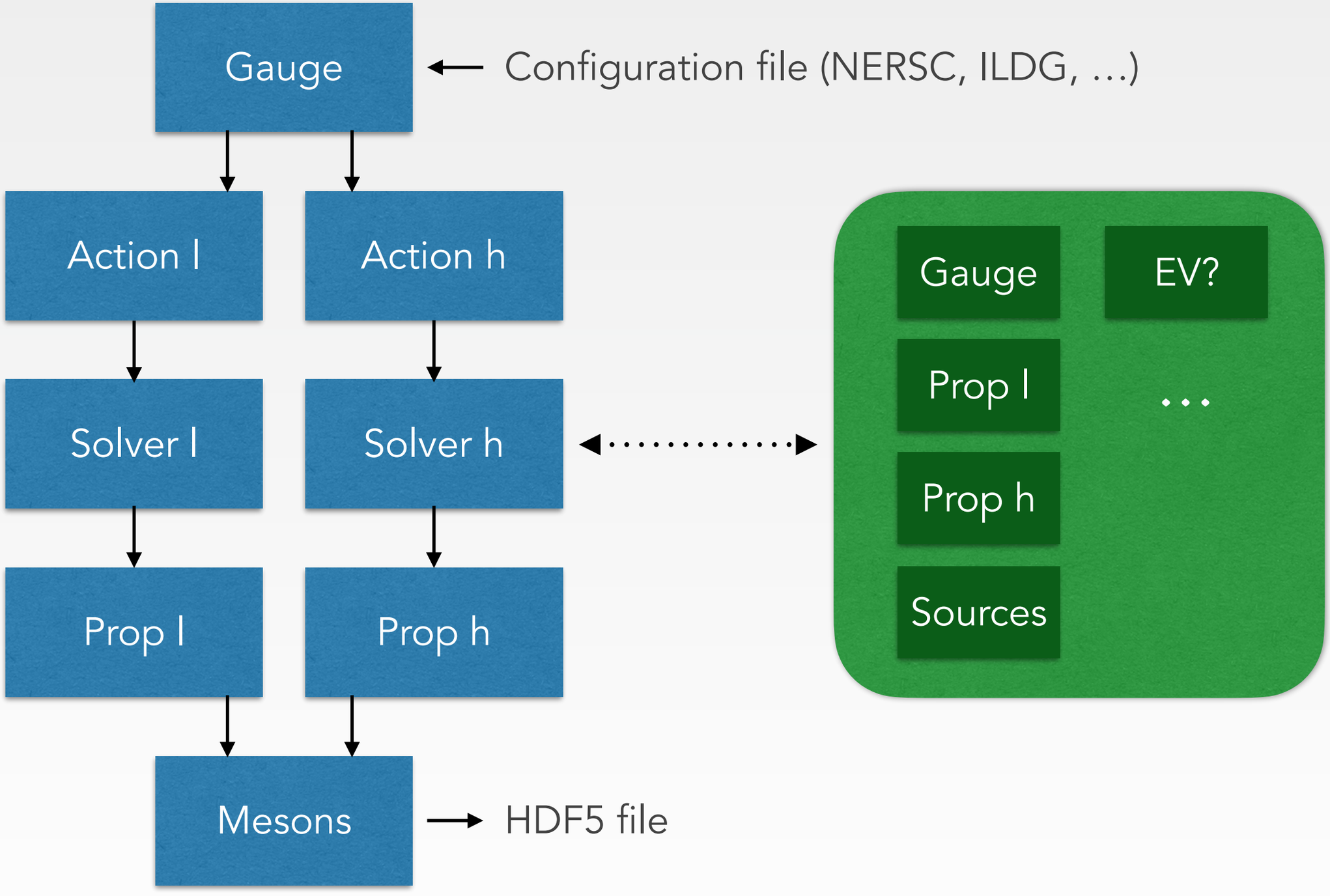
- **Flexible I/O & control** — highly customisable input.

- **Automatic scheduling** — more self-consistency checks.

Measurement data flow



Measurement data flow



Scheduling

- Dataflow diagram: **Directed Acyclic Graph**.
- Dependency solving: DAG **topological sort**.
- Memory optimisation 1: garbage collection.
- Memory optimisation 2: constrained topological sort.
- Very likely NP-hard problem: need a **heuristic solution**.
- So far: **genetic algorithm** minimising high-water function on the space of topological sorts.
- Find a schedule in O(10 min) for big graphs.

Flexible control

Hardcoded C++ ASCII input (*e.g.* XML)

- Hardcoded: risk of code (and bug) duplication.
- ASCII input: too general, complicated input.
- Matter of taste: user should be able to choose.
- Achieved with modules + Grid generic serialisation.

Data considerations

- How to store a whole application (modules, object catalog, schedule, …) in an efficient and queryable way? (avoiding ASCII things like XML, JSON, …)

- How to build a global, real-time instrumentation of physics runs? (again in a simply queryable way)

- How to catalog automatically measurements produced by a run with specialised metadata related to physics of the run? (again in a simply queryable way)

SQLite DB support

- SQLite embedded in Hadrons, no dependencies.
- High-level Database class.
- DB class can execute arbitrary SQL statements and return table of string as answer.
- Generic serialisable SQL entry types.
- DB class can serialise and de-serialise any entry from/to any Grid serialisable type.

Hadrons standard databases

- Application DB: store modules and parameters, object list with types and footprint, schedule.
  Application can be entirely reconstructed from DB.

- Result DB: catalog of produced result with custom metadata.

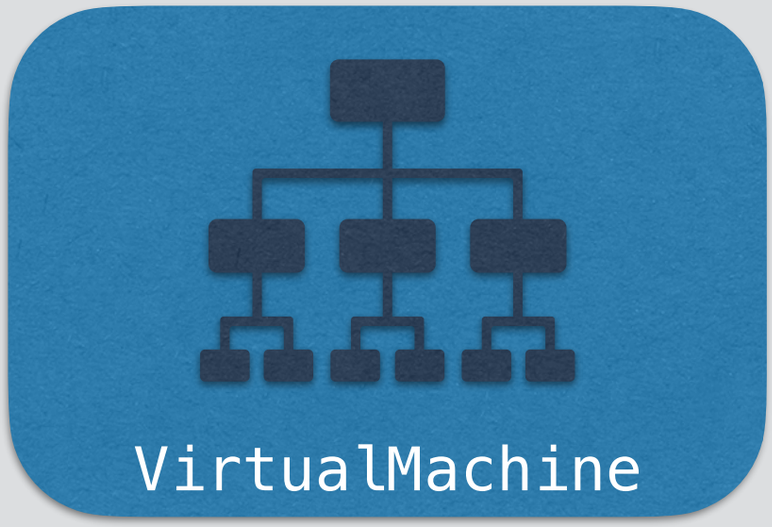
- Stat DB: real-time statistics on run (2 Hz sampler).

Stat DB example

UKQCD QCD+QED production run — made using DB Browser (https://sqlitebrowser.org)

Full structure

- Module DAG
- Scheduling & garbage collection
- DB for modules & objects



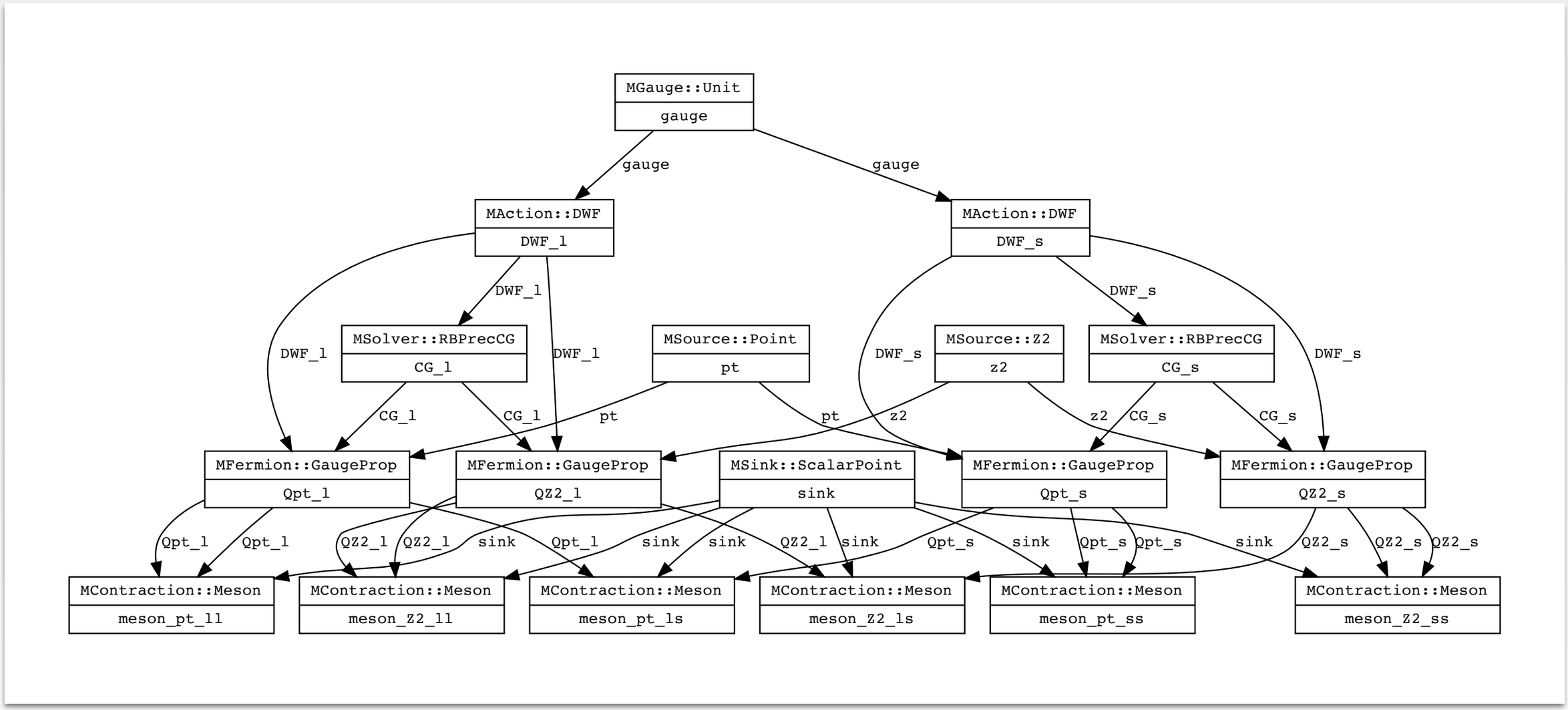
- Named object store
- Memory footprint aware

Application

- High-level control interface
- DB for profiling and result catalog

Hadrons::

Workflow example



Strange & light meson spectrum
(trimmed down version of `Test_hadrons_spectrum`)

UKQCD production workflow examples

- Rare kaon decays
  O(10000) modules
- Isospin breaking corrections to light leptonic decays
  O(1000) modules
- Scattering with distillation
  O(1000) modules
- Holographic cosmology
  O(10) modules

Available modules

- Actions: Wilson, clover, various flavours of DWF, …
- Solvers: RB prec CG, mixed-precision CG, exact deflation (Lanczos)
- Contraction: gamma matrices 2 & 3-pt functions, 4-quark weak operators, meson & baryons, …
- Distillation, A2A, LMA, …
- Various sources, EM potential generation, sequential solves, scalar field theory, other exotic things…

Outlook

- Grid + Hadrons: cross-platform, high-performance lattice software.

- Grid: high-performance data parallel library.

- Hadrons: high-level interface focused on physics measurements, using Grid for performance routines.

- Modular structure, with automatic scheduling.
  Aimed at fast & future-proof project development.

- Used in production for a wide variety of calculations.

Thank you!

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreements No 757646 & 813942.