Large-scale and user-friendly exact diagonalization in Chapel

Tom Westerhout (Radboud University)
Mikhail I. Katsnelson (Radboud University)
Bradford L. Chamberlain (HPE)

10 June
CHIUW 2022
Radboud University

- Located in Nijmegen — the oldest city in the Netherlands;
- In 2010 Andre Geim and Konstantin Novoselov got a Nobel Prize in Physics for the discovery of graphene;
- Strong Programming Languages department: Clean, SAC (Single Assignment C);
Our research aims to predict and explain the properties of condensed matter. We utilize and develop advanced mathematical frameworks and state-of-the-art numerical approaches.
Exact diagonalization

A numerical technique to simulate small quantum systems.

Goal:
Competitive performance while preserving user friendliness.

Constraint:
Limited developer time.

Approach:
A mix of languages among which Chapel.
Exact diagonalization

- Physical system is described by a matrix $H$, the Hamiltonian.
- Eigenvalue problem (i.e. find a scalar $E$ and a vector $v$ such that $Hv = Ev$).

$$H = \sigma_a \otimes \sigma_b + \sigma_b \otimes \sigma_c + \sigma_a \otimes \sigma_c$$

$$= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \mathbb{I}$$

$$+ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \mathbb{I} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$+ \ldots$$
Challenges

- Exponential scaling of resources (both memory and compute);
- Trading memory for computations;

**Target:** 50 spins

- $10^{15} \times 10^{15}$ matrix (8PB for one vector);
- with symmetries $-10^{11} \times 10^{11}$ (15.5TB for the simulation);

Fix Hamming weight

Translation invariance

- Exponential scaling of resources (both memory and compute);
- Trading memory for computations;
lattice_symmetries
(old architecture)

- **Python wrapper module**
- **Haskell application**
- **C++ library** SIMD for kernels

- High-level functions; algorithm prototyping
- Simple interface; portable static executable

Kernels for a few processors; runtime dispatch to the right one; x86-64 only

---

Tom Westerhout. Journal of Open Source Software 6, 64 (2021), 3537.

Institute for Molecules and Materials
Radboud University
"Halide is a programming language designed to make it easier to write high-performance image and array processing code on modern machines."

```cpp
Func blur_3x3(Func input) {
    Func blur_x, blur_y;
    Var x, y, xi, yi;

    // The algorithm - no storage or order
    blur_x(x, y) = (input(x-1, y) + input(x, y) + input(x+1, y))/3;
    blur_y(x, y) = (blur_x(x, y-1) + blur_x(x, y) + blur_x(x, y+1))/3;

    // The schedule - defines order, locality; implies storage
    blur_y.tile(x, y, xi, yi, 256, 32)
        .vectorize(xi, 8).parallel(y);
    blur_x.compute_at(blur_y, x).vectorize(x, 8);
    return blur_y;
}
```
Potential for multi-node parallelism;
In single-locale setting, we can still generate a standalone shared library;
Simpler and shorter code;
New architecture

Python wrapper module

high-level Haskell code

low-level C code

Halide kernels

Chapel application

Chapel for parallel algorithms

Compiled twice!
Singularity

1. Debian with custom OpenMPI;
2. Chapel with CHPL_LAUNCHER=none:
   - CHPL_COMM=none
   - CHPL_COMM=gasnet & CHPL_COMM_SUBSTRATE=mpi
   - CHPL_COMM=gasnet & CHPL_COMM_SUBSTRATE=ibv
3. Our project;
4. Minimal release container based on Busybox;

```bash
# On your laptop
singularity exec hello-world.sif /project/bin/none/hello6-taskpar-dist
# On our local cluster (ethernet)
mpirun -np 3 singularity exec hello-world.sif /project/bin/mpi/hello6-taskpar-dist -nl 3
# On the Dutch National supercomputer
salloc -N 20 --ntasks-per-node=1 --exclusive \n   srun -n 20 singularity exec hello-world.sif /project/bin/ibv/hello6-taskpar-dist -nl 20
```
Current limitations

- Performance of distributed implementation should be improved; Thanks to Engin Kayraklioglu and Ben Harshbarger we now have a plan 😊
- Non x86-64 architectures should work, but ...

Chapel feature requests

- Compile times;
- Tools for profiling;
- ref record & class attributes (currently using c_ptr as a hack);
Conclusion

- Exact diagonalization — a method to simulate small quantum systems;
- Exponential scaling of compute resources — need distributed parallelism;
- We mix Halide, C, Haskell, Python, and Chapel to reduce the amount of code (800 lines of C, 3500 lines of Haskell, 2000 lines of Chapel) vs. other projects with >25000 lines;
- Singularity as the packaging mechanism;

**Next step:** performance tuning