上海科技大学
ShanghaiTech University

Victory: Porting and Optimizing

GPU Hackathon@SJTU

GeekPie_HPC
SIST@ShanghaiTech University

Victory

Github repository: gangli-SHT/victory

Gang Li, Nils Wentzell, Petra Pudleiner,
Patrik Thunström, and Karsten Held

Phys. Rev. B 95, 165103 (2016)

Victory is an efficient implementation of the Parquet equation for the single-band Hubbard model. It fully respects the periodic and anti-periodic boundary condition of the parquet equation in momentum and frequency spaces. Currently the following feasures are implemented:

(1) Both 1D and 2D calculations of the Hubbard model with on-site Coulomb interactions.

(2) Single-band Hubbard model on ladder, square, and triangluar lattices.

(3) Efficient use of memory, currently a 4 x 4 square lattice study is feasible.

(4) Both single- and two-particle quantities can be equally obtained.

(5) Channel instability is interpreted as the divergence of the corresponding eigenvalue, which is useful for analyzing phase transitions.

(6) Density of states can be obtained after supplementing with analytical continuation.

Code Structure

Gang Li update README		Latest commit d72ddc0 on May 25
lib	upload victory source code	a month ago
out	upload victory source code	a month ago
src	upload victory source code	a month ago
Makefile	upload victory source code	a month ago
README.md	update README	a month ago
logo.png	upload logo picture	a month ago
make.inc	upload victory source code	a month ago

→ "lib" directory contains the FFT subroutines, here we adopt the "dfftpack" package from Netlib

→ "src" directory contains the Victory source code

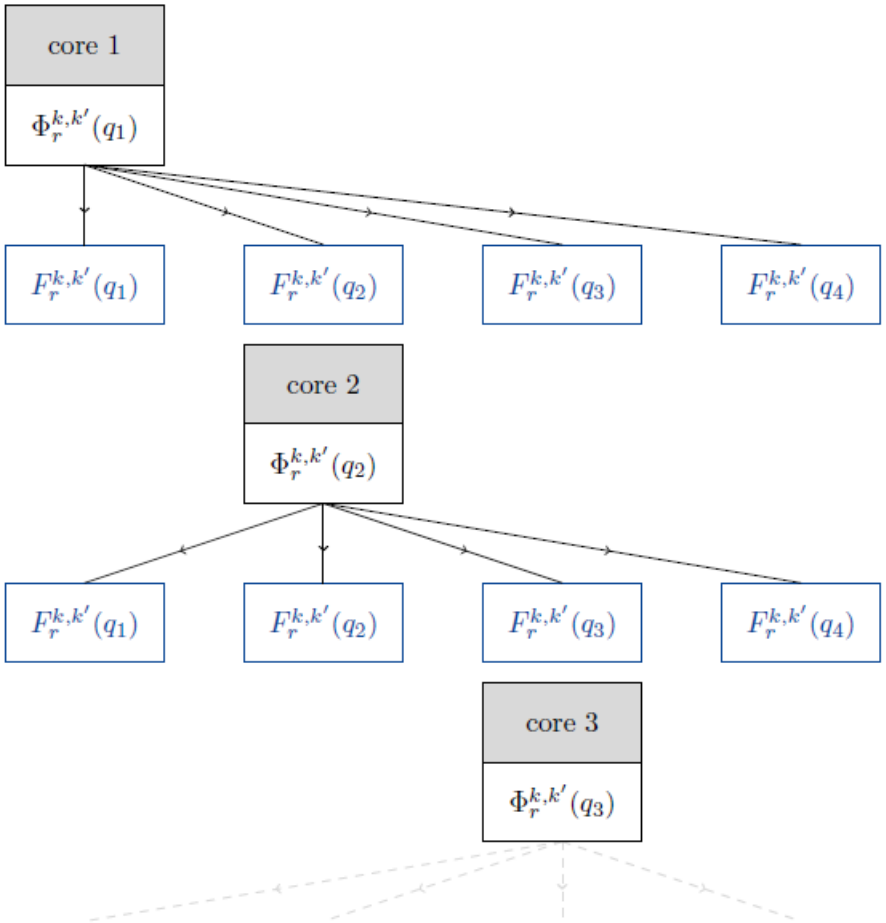
SRC_MOD	→	all supporting library routines are stored here
SRC_PA	→	the core part of Victory code
SRC_TEMPLATE	→	here provides a template main routine for users to organize own calculations.

Procedure

main.f90

```
do while (.NOT. Converged .and. ite < 50)

    ! calculate single-particle green's function
    call pa_Gkw_Chi0(ite, Grt)

    ! determine reducible vertex function and its kernel approximation
    call reducible_vertex(ite)
    call get_kernel_function(ite)

    ! solve the parquet equation
    call solve_parquet_equation(ite)

    ! calculate the self-energy
    call self_energy(ite, Grt, converged)

    ! --- update irreducible vertex in each channel ---
    G_d = F_d - G_d
    G_m = F_m - G_m
    G_s = F_s - G_s
    G_t = F_t - G_t

    ite = ite + 1
end do

! ---- calculate eigen-values in each channel -------
call solve_eigen_equation
```

Parallelism



Data are uniformed distributed to cores

Heavy matrix computation are done inside every core

Information are exchanged in certain phases between cores

Profiling Result

Advanced Hotspots Hotspots viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform | parquet_

Grouping: Function / Call Stack

Function / Call Stack	CPU Time ▼	Instructions Retired	CPI Rate	CPU Frequency Ratio	Module
solve_parquet_equation	114.250s	755,140,000,000	0.448	1.186	victory
kernel	72.585s	675,517,500,000	0.327	1.222	victory
reducible_vertex	66.839s	413,005,000,000	0.481	1.191	victory
self_energy	64.490s	334,135,000,000	0.585	1.215	victory
list_index	26.823s	214,265,000,000	0.380	1.216	victory
[vmlinux]	25.815s	79,387,500,000	1.005	1.239	vmlinux
index_operation	21.552s	172,860,000,000	0.370	1.191	victory
poll_all_fboxes	16.257s	128,675,000,000	0.384	1.218	libmpi.so.12.0
MPID_nem_mpich_blocking_recv	15.634s	140,985,000,000	0.337	1.219	libmpi.so.12.0
[MKL BLAS]@avx2_zgemm_kernel_0_b0	15.531s	148,582,500,000	0.298	1.142	libmkl_avx2.so
[MKL BLAS]@avx2_zgemm_kernel_0	15.527s	149,062,500,000	0.297	1.144	libmkl_avx2.so
poll_active_fboxes	15.480s	144,912,500,000	0.320	1.202	libmpi.so.12.0
cvtas_t_to_a	6.665s	43,240,000,000	0.490	1.274	victory
main	5.402s	51,825,000,000	0.320	1.231	victory
[Outside any known module]	3.636s	1,702,500,000	6.367	1.195	
cvt_ieee_t_to_text_ex	3.145s	17,577,500,000	0.539	1.207	victory
for_write_seq_fmt_xmit	2.793s	15,567,500,000	0.590	1.317	victory
I_MPI_memcpy_movsb	2.614s	27,500,000	279.727	1.180	libmpi.so.12.0
for__format_value	2.051s	14,962,500,000	0.442	1.293	victory

Profiled by Intel VTune Amplifier

Profiling Result

Grouping: Function / Call Stack

Function / Call Stack	CPU Time ▼		
	Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Spin Time	
		MPI Busy Wait Time	Other
kernel	72.585s	0s	0s
self_energy	58.892s	0s	0s
list_index	26.823s	0s	0s
[vmlinux]	25.815s	0s	0s
index_operation	21.552s	0s	0s
poll_all_fboxes	0s	16.257s	0s
[Loop at line 85 in reducible_vertex]	16.043s	0s	0s
MPID_nem_mpich_blocking_recv	0s	15.634s	0s
poll_active_fboxes	0s	15.480s	0s
[Loop at line 143 in reducible_vertex]	15.296s	0s	0s
[Loop at line 284 in solve_parquet_equation]	12.554s	0s	0s
[Loop at line 228 in solve_parquet_equation]	12.324s	0s	0s
[Loop at line 91 in solve_parquet_equation]	11.508s	0s	0s
[Loop at line 159 in solve_parquet_equation]	11.391s	0s	0s
[Loop at line 124 in solve_parquet_equation]	10.754s	0s	0s
[Loop at line 112 in reducible_vertex]	10.625s	0s	0s
[Loop at line 193 in solve_parquet_equation]	10.341s	0s	0s
[Loop at line 54 in reducible_vertex]	10.296s	0s	0s
[Loop@0x92e710 in [MKL BLAS]@avx2_zgemm	8.734s	0s	0s
[Loop@0x92fb10 in [MKL BLAS]@avx2_zgemm	8.670s	0s	0s

Profiled by Intel VTune Amplifier

Porting Plan

- Implement the computation kernels for GPU platform
  - Accelerate the massive linear matrix computation
  - cuBLAS, cuFFT, etc
- Design the communication pattern
  - GPU-aware MPI
- Optimize the memory accessing issues
  - Unified memory, Streaming