OAK RIDGE National Laboratory

Optimizing Dynamical Cluster Approximation on the Summit supercomputer

Arghya Chatterjee
Performance Engineer, Scientific Computing Group
National Center for Computational Sciences (NCCS)
Aug 8, 2019

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

SciDAC: Computational Framework for Unbiased Studies of Correlated Electron Systems

- I would like to thank:

  Giovanni Balduzzi (PhD. Student, ETH Zurich)
  Urs Hähner (PhD. Student, ETH Zurich)
  Ying Wai Li (NCCS, ORNL)
  Thomas Maier (CNMS, ORNL)
  Thomas Schulthess (Director, CSCS, ETH Zurich)

- *The **profiling and optimization of DCA++** was supported by the **Sci**entific **D**iscovery through **A**dvanced **C**omputing (sciDAC) program funded by U.S. DOE, Office of Science, Advanced Computing Scientific Computing Research (ASCR) and Basic Energy Sciences (BES), Division of Materials Science and Engineering.*

- *This research used resources of the **O**ak Ridge **L**eadership **C**omputing **F**acility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.*

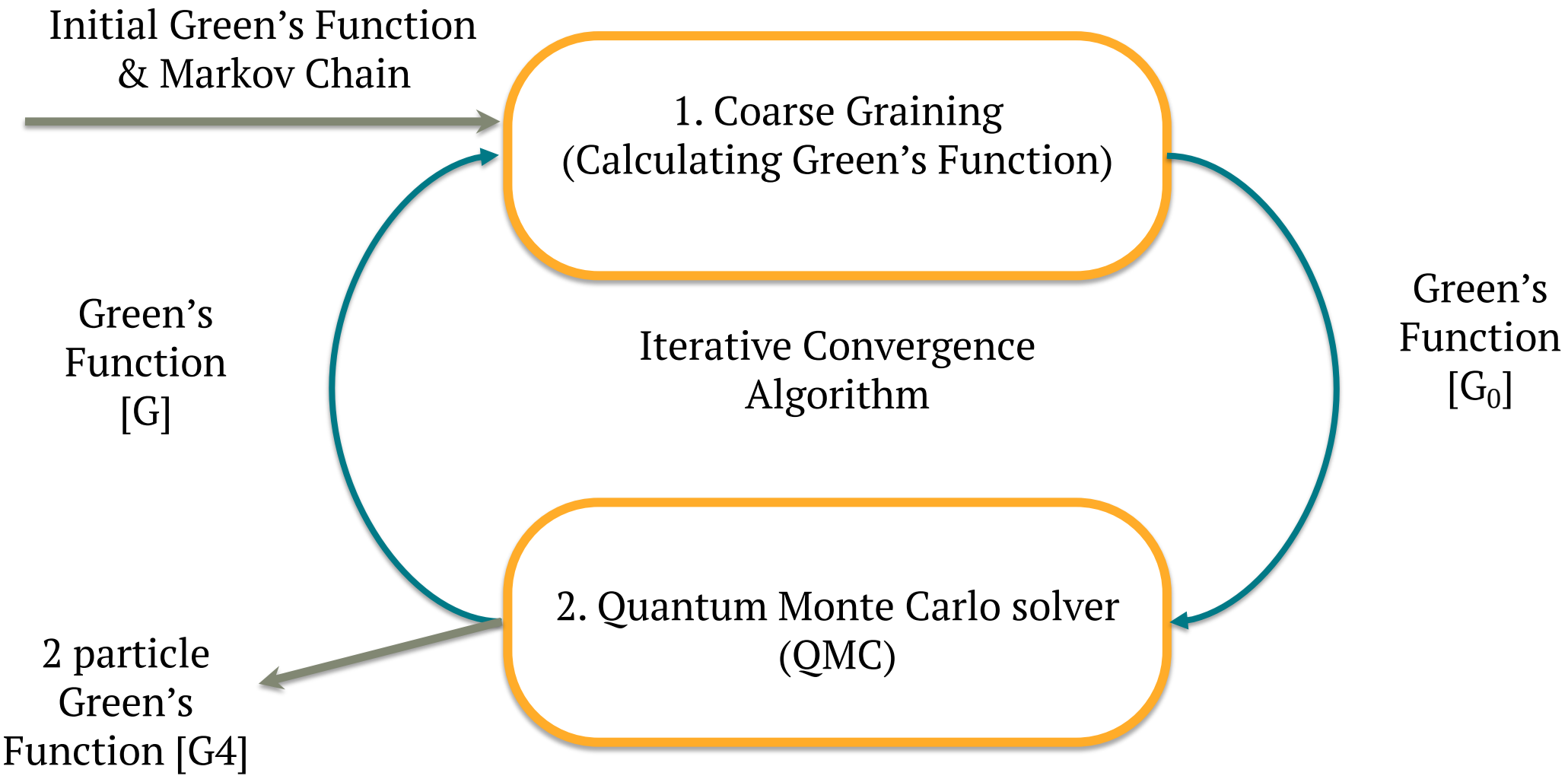
Overview : DCA++

- DCA++ → Dynamical Cluster Approximation
- Numerical simulation tool — predict behaviors of co-related quantum materials (such as **superconductivity**, **magnetism**)
- DCA++ — computes **many-body Green's function** — material's properties can be calculated from this function.
- Iterative self consistent algorithm — two primary kernels
  - **Coarse graining** of single particle *Green's function* (reduces complexity of infinite size lattice problem)
  - **Quantum Monte Carlo** solver

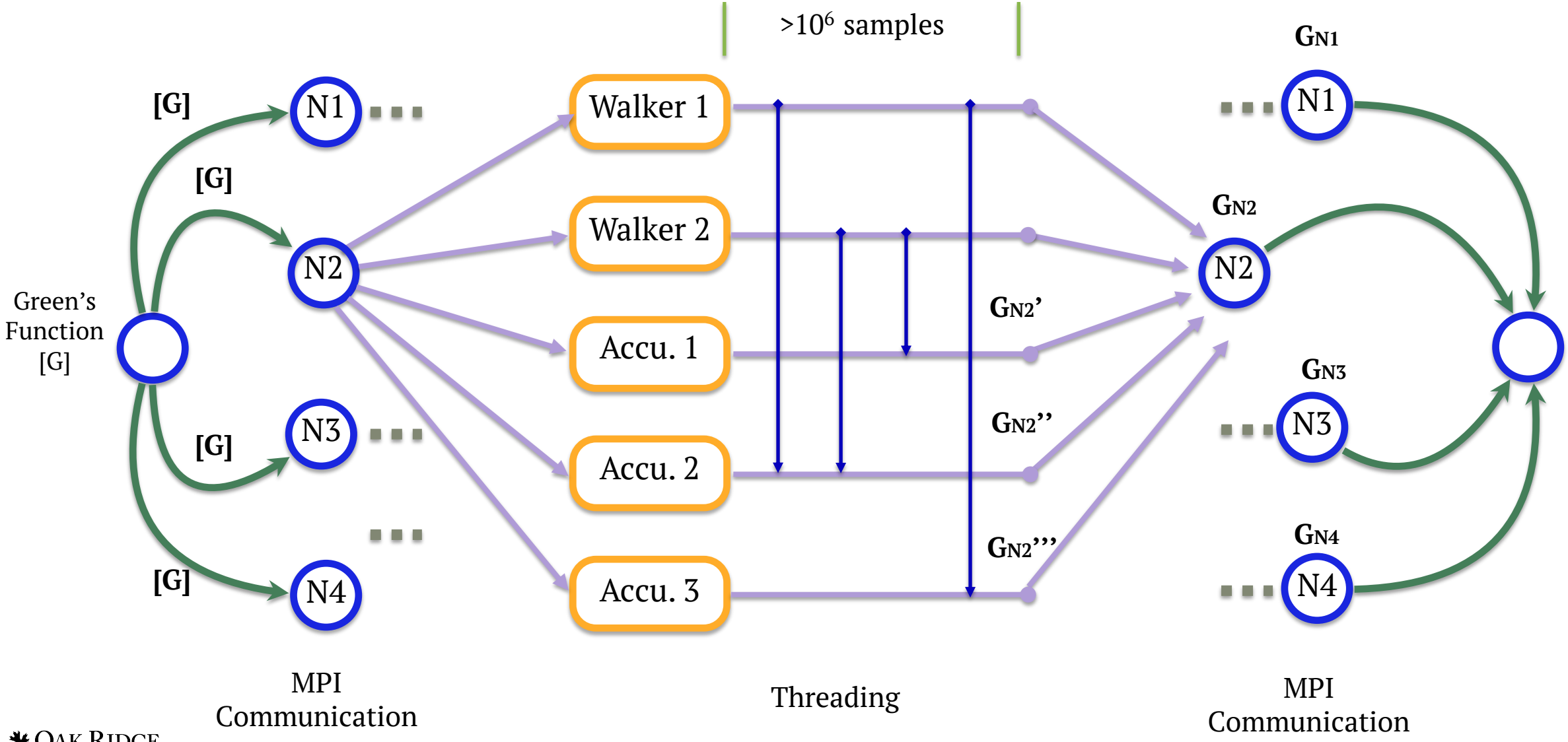
DCA++ ran on Titan – 18600 nodes at16 Petaflop rate (peak), sustained 1.3 Petaflop rate [Gordon Bell 2009]

DCA++: A software framework to solve correlated electron problems with modern quantum cluster methods. *U. Hähner, et al. .CPC 2019.*

DCA++ : Primary kernels workflow

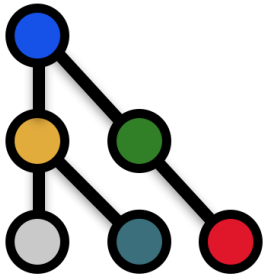


DCA++ : Quantum Monte Carlo Solver



Walkers running on GPU : Accumulators run on the CPU

Profiling DCA++ on Titan

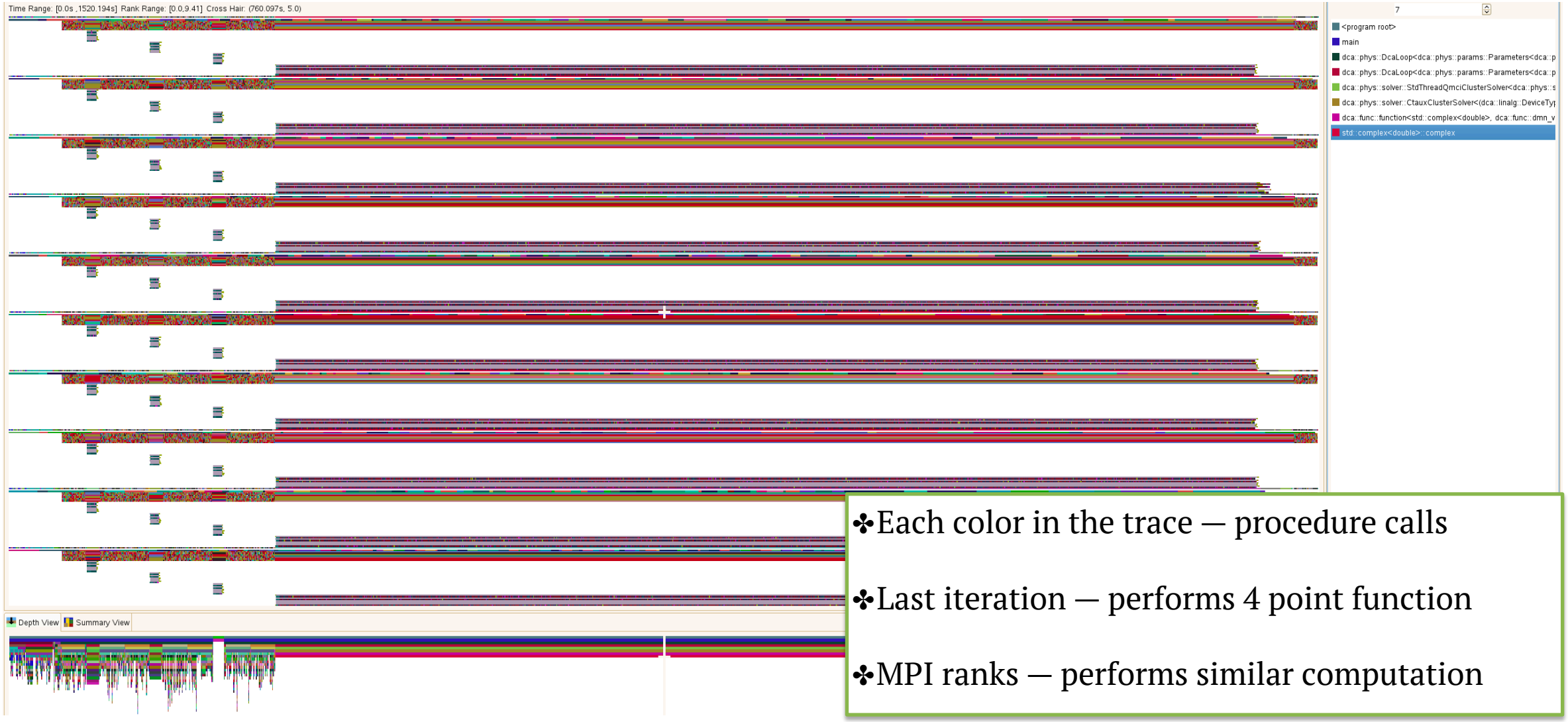


- **Profiler used**: HPCToolkit [Visualizer: HPCTraceviewer]

- **DCA++ code**:
  - Synthetic dataset (realistic to the science)
  - Beta: 20
  - Number of Titan nodes used: 10
  - Number of MPI ranks used: 10 (1 rank per node)
  - Iterations: 4 (till convergence)
  - Last iteration step — performs the 4 point function

- **Acknowledgment**: Dr. John Mellor Crummey & Dr. Laksono Adihanto (Rice U.)

Profiling DCA++ on Titan (using HPCToolkit)



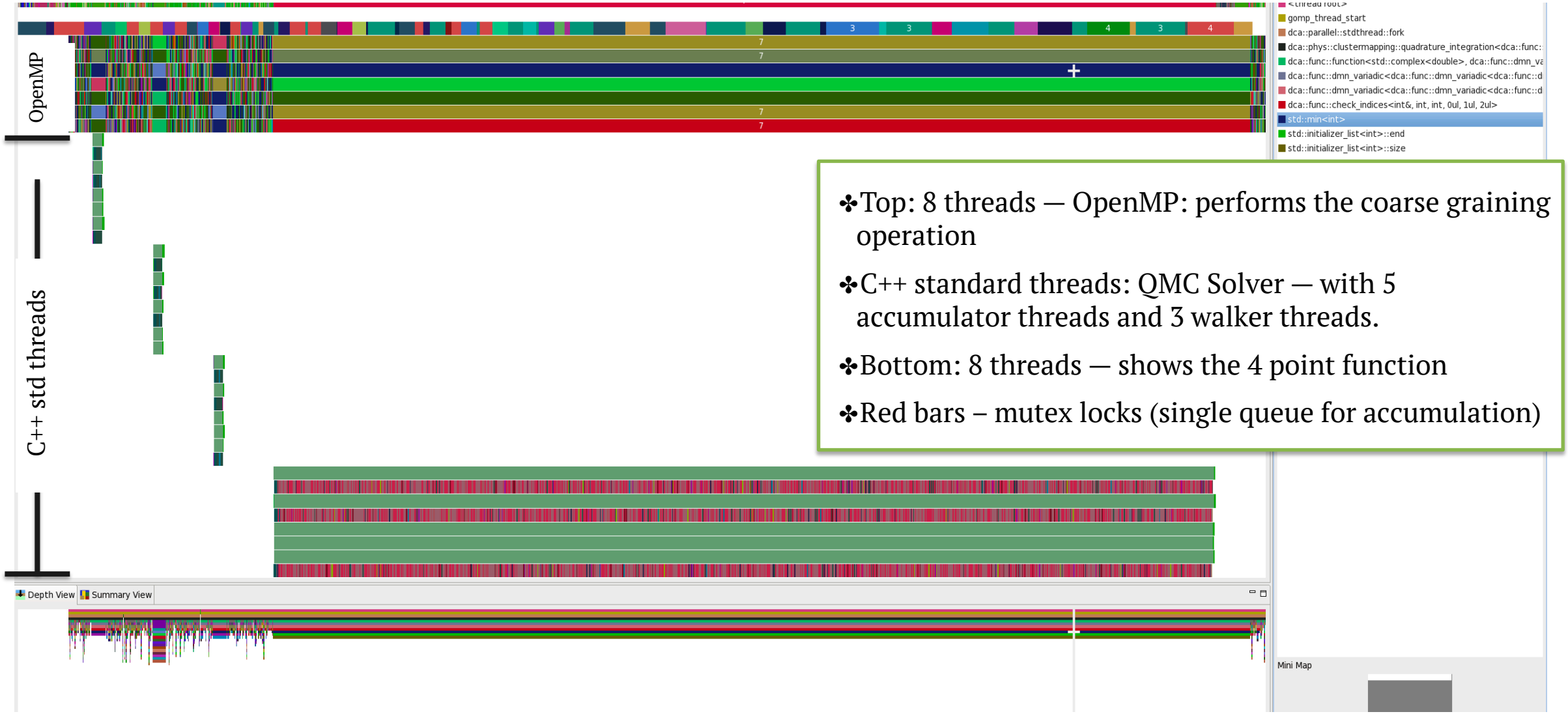
MPI Ranks + Threads

Iterations [execution time]

- Each color in the trace — procedure calls
- Last iteration — performs 4 point function
- MPI ranks — performs similar computation

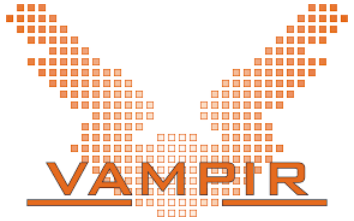
OAK RIDGE National Laboratory

Profiling DCA++ on Titan (using HPCToolkit)



- Top: 8 threads — OpenMP: performs the coarse graining operation
- C++ standard threads: QMC Solver — with 5 accumulator threads and 3 walker threads.
- Bottom: 8 threads — shows the 4 point function
- Red bars – mutex locks (single queue for accumulation)

Profiling DCA++ on Titan

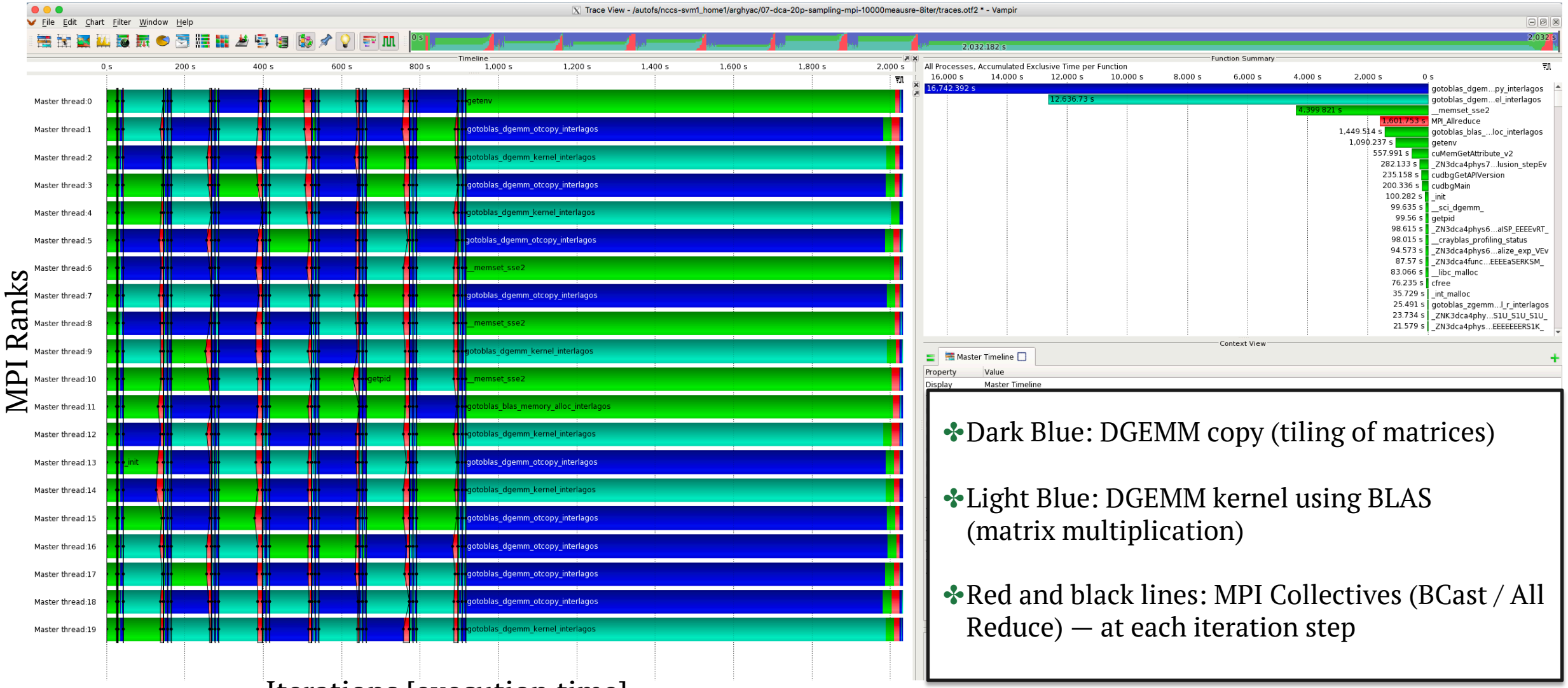


- **Profiler used**: ScoreP [Visualizer: VAMPIR]

- **DCA++ code**:
  - Synthetic dataset (realistic to the science)
  - Beta: 20
  - Number of Titan nodes used: 20
  - Number of MPI ranks used: 20 (1 rank per node)
  - Iterations: 8 (till convergence)
  - Last iteration step — performs the 4 point function

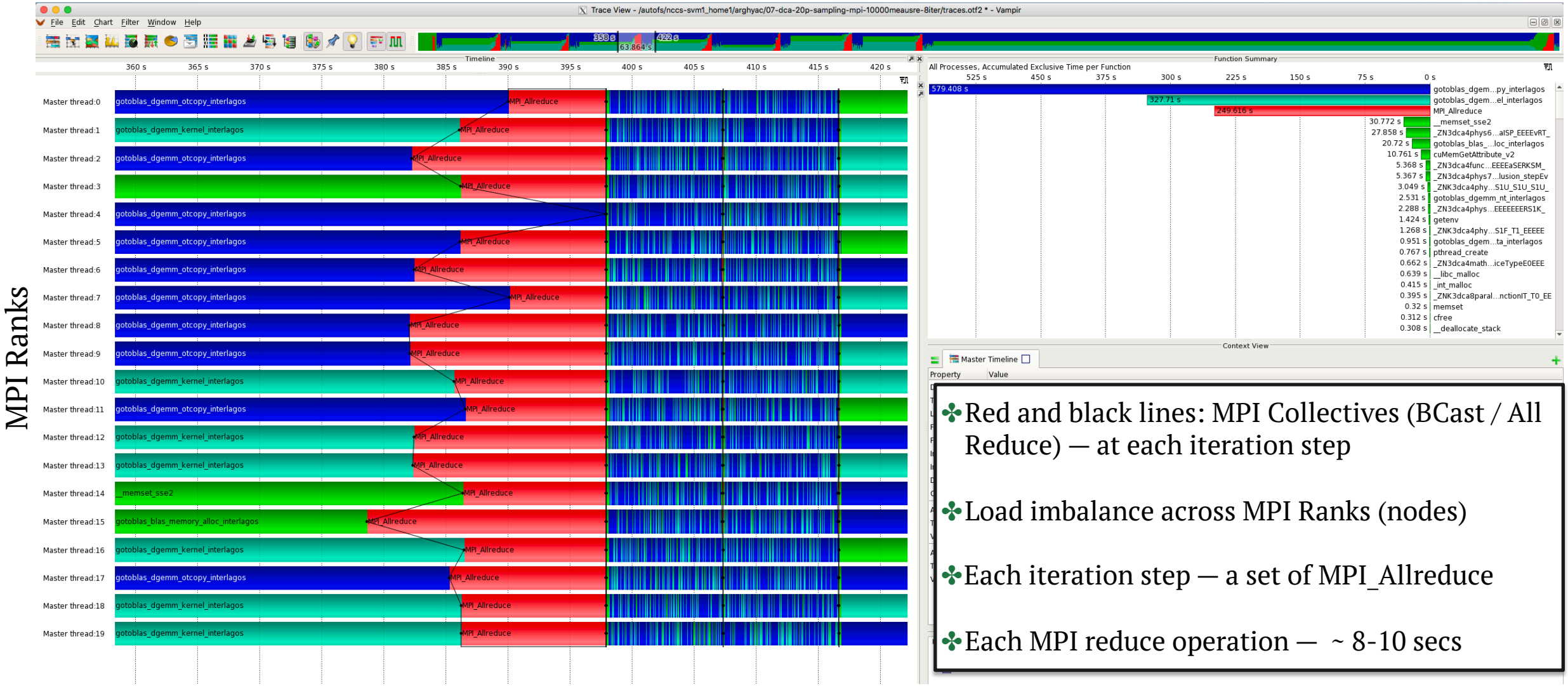
- **Acknowledgment**: Ronny Brendel (T.U. Dresden / ScoreP ; Currently: NVIDIA)

Profiling DCA++ on Titan (using ScoreP)



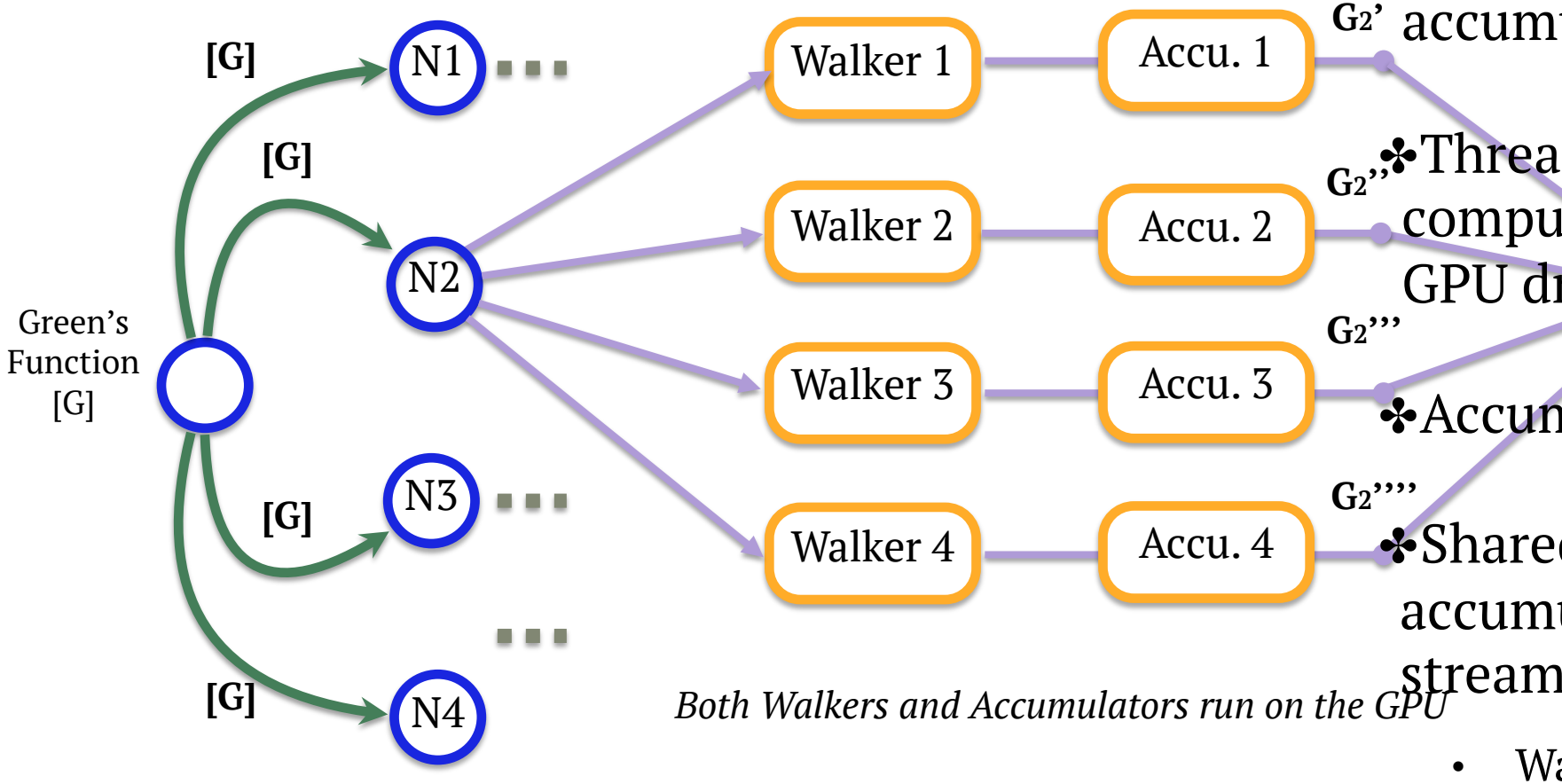
- Dark Blue: DGEMM copy (tiling of matrices)
- Light Blue: DGEMM kernel using BLAS (matrix multiplication)
- Red and black lines: MPI Collectives (BCast / All Reduce) — at each iteration step

Profiling DCA++ on Titan (using ScoreP)



- Red and black lines: MPI Collectives (BCast / All Reduce) — at each iteration step
- Load imbalance across MPI Ranks (nodes)
- Each iteration step — a set of MPI_Allreduce
- Each MPI reduce operation — ~ 8-10 secs

Quantum Monte Carlo Solver (new code)



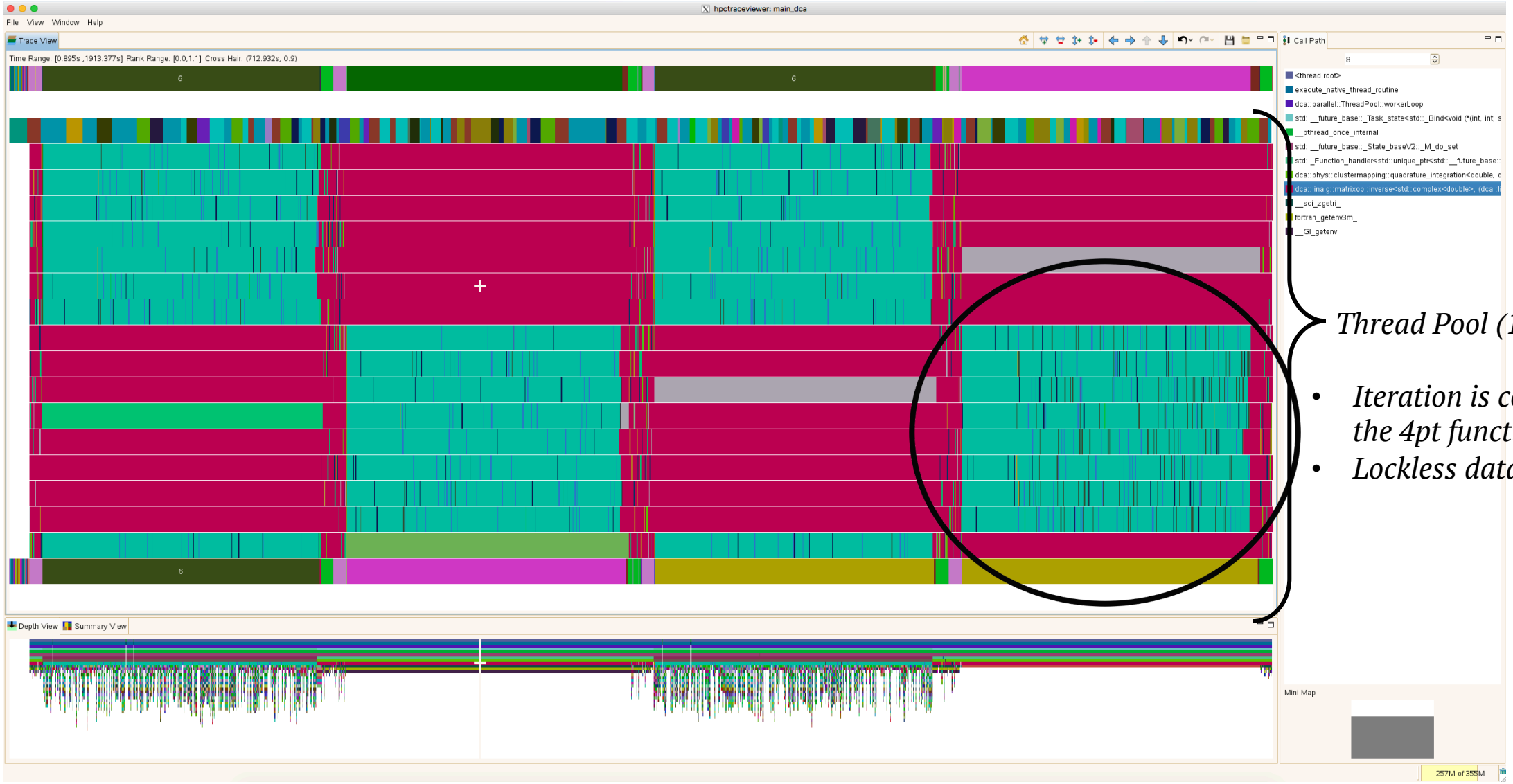
Both Walkers and Accumulators run on the GPU

"Walker – Accumulator" mapping: 1:1

- ✤Lockless data structure for the accumulator queue
- ✤Thread pool for the CPU computation coarse graining / GPU driver threads
- ✤Accumulation now on the GPUs
- ✤Shared walker / accumulators: accumulation on a separate GPU stream
  - Walkers continue random walk

Profiling DCA++ on Titan (using HPCToolkit)

MPI Ranks + Threads



Thread Pool (16 threads)

- *Iteration is computing the 4pt function*
- *Lockless data structure*

Iterations [execution time]

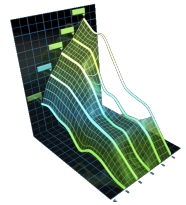
Profiling old / improved DCA++ on Summit

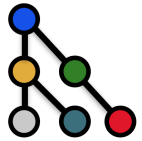
- **Profiler used**:

  NVProf (CUDA Toolkit)

  TAU (SciDAC Institute) – Ongoing Collaboration

  HPC Toolkit (Rice University)– Ongoing Collaboration





- **Versions of DCA++ code (compare and contrast)**:
  [Both codes are now publicly available: *https://github.com/CompFUSE/DCA*]

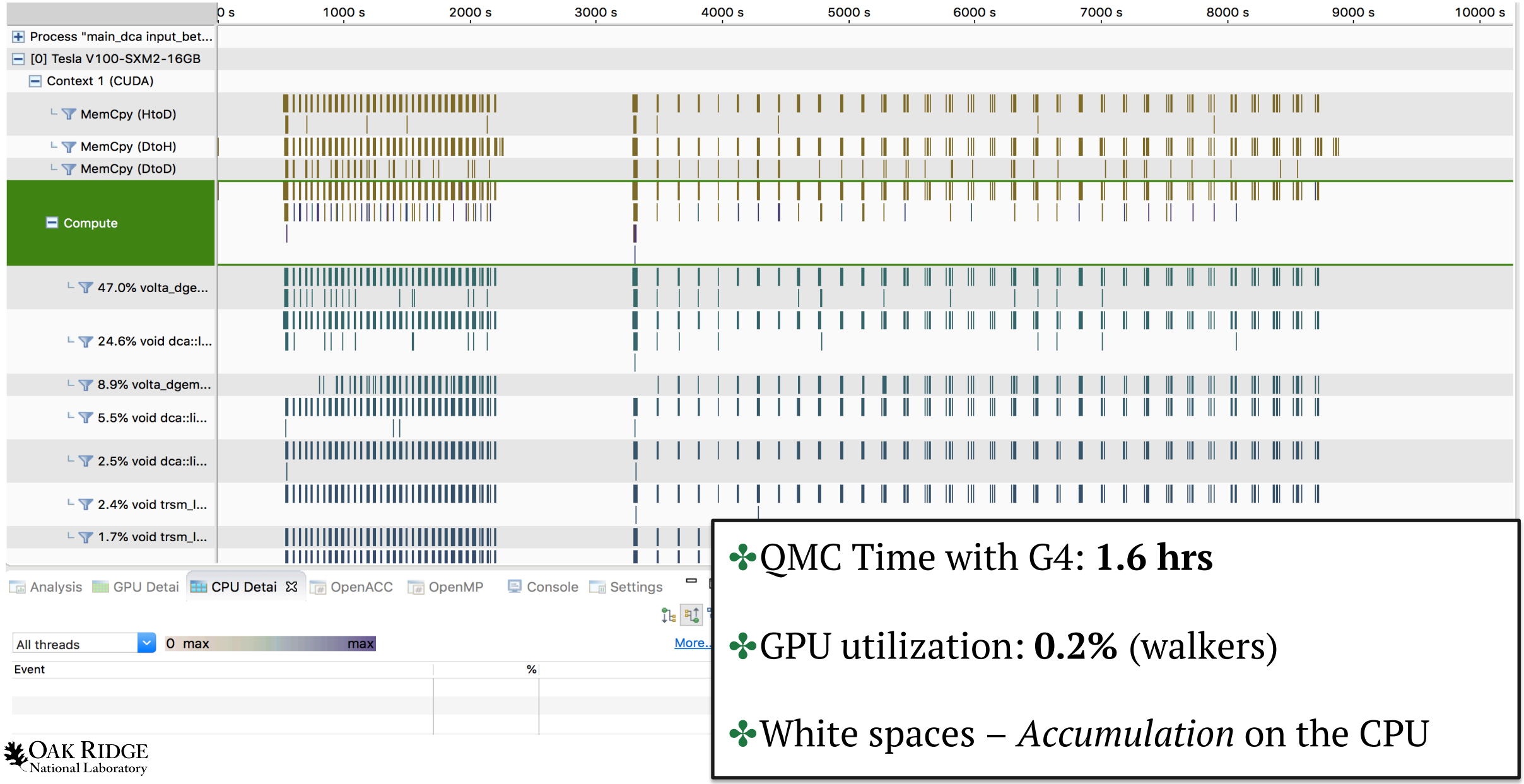
  - Old Code [pre SciDAC] – version 1.0.0
  - New Code [March 10th , 2019] – Ongoing

Input Parameters [large case]

- **Beta**: 50
- **Initial self-energy**= “T= 0.02/dca_sp.hdf5”
- **Iterations**: 2
- **Do-finite-qmc**: true (coarse graining is ON) – Turned off for production run
- **Measurements:** 1020
- **Cluster size:** 24
- **Initial configuration size**: 3300
- **Walkers**: 7 (=h/w threads mapped)
- **Accumulator**: 5
- **Shared Walker / Accumulator**: false
- **G4**: ON (last iteration)

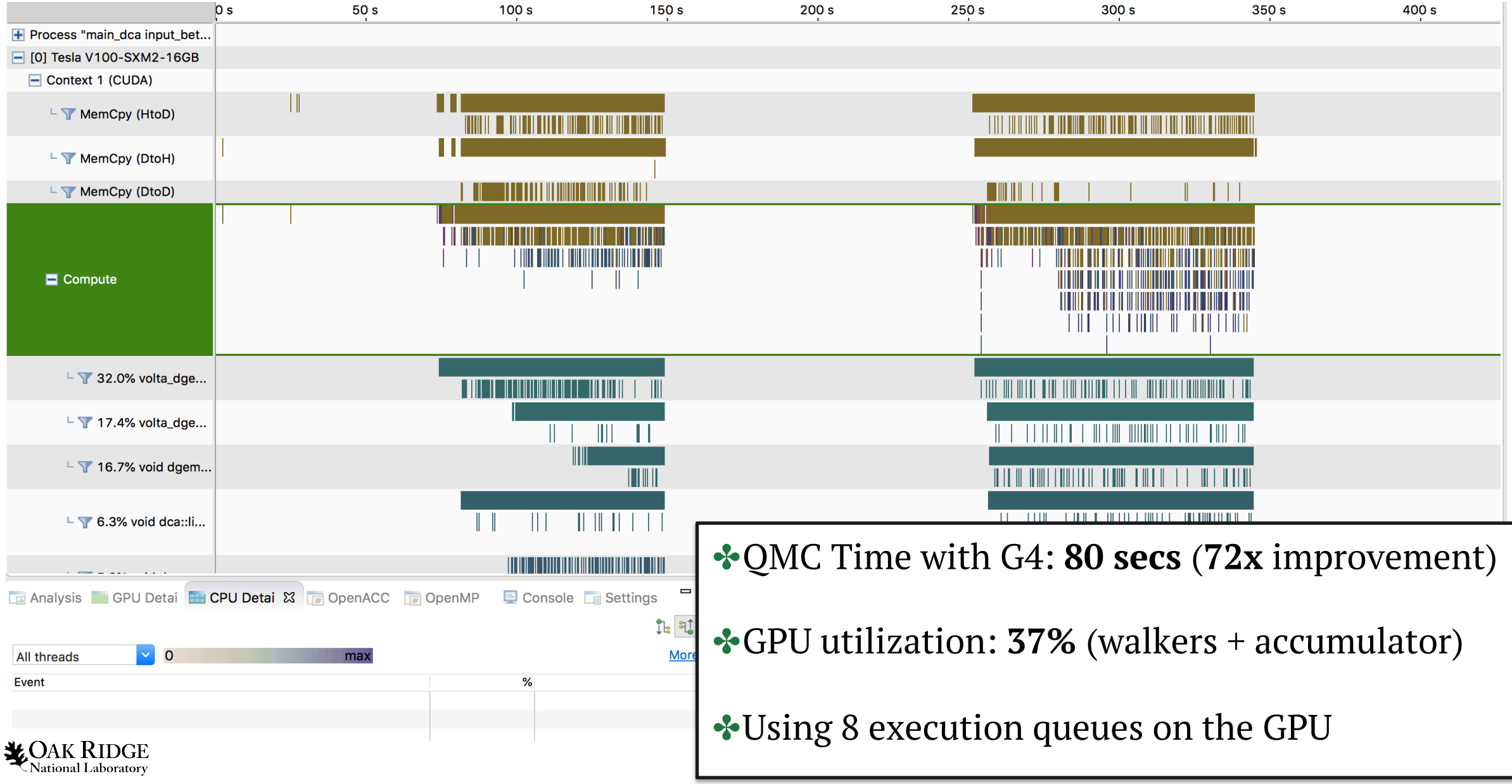
- Summit: 1 node :: 6 MPI ranks :: 1 rank/rs :: 1 rank/gpu :: smt1 :: 7 h/w threads /rs

DCA++ Profile [Old Code]



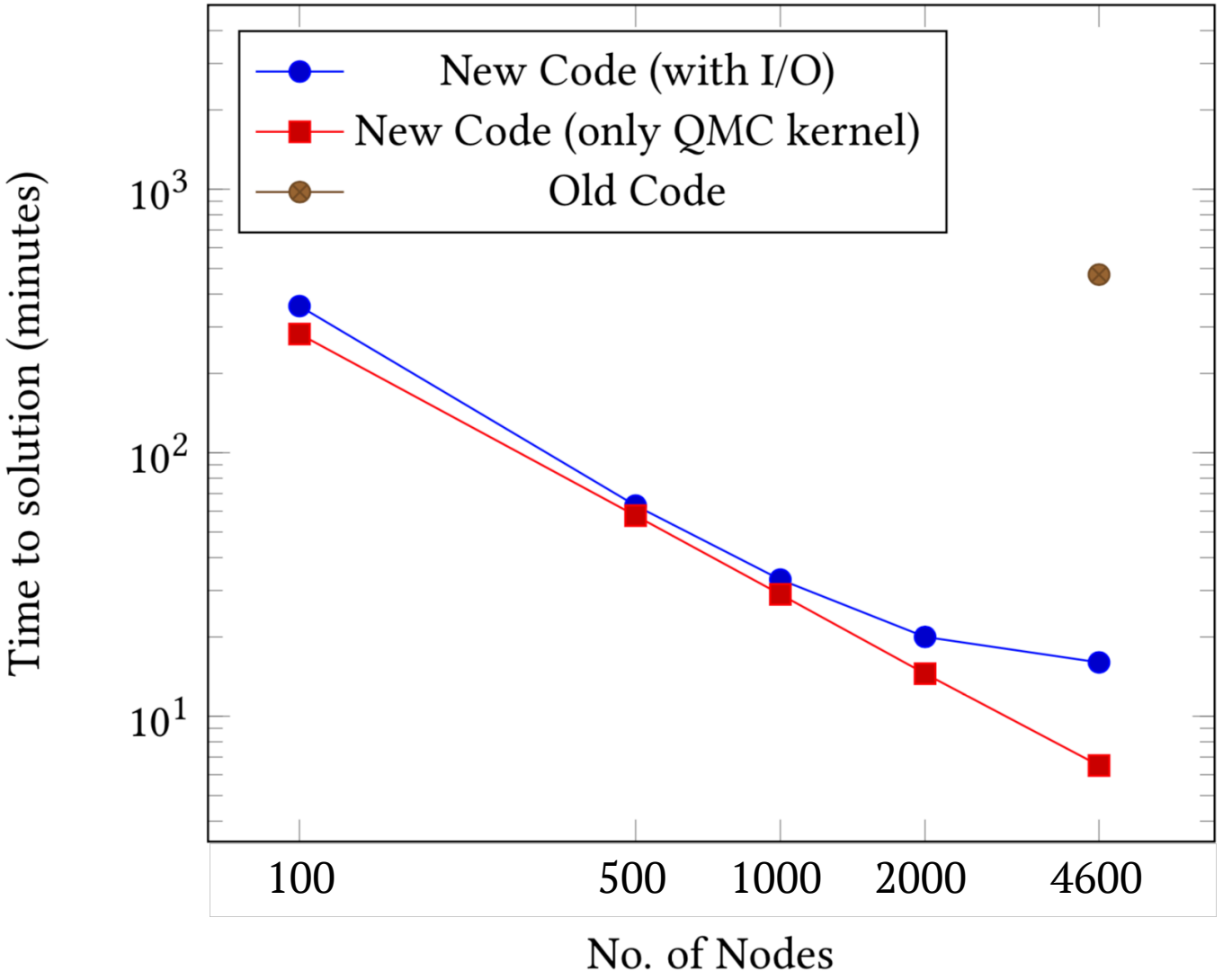
- QMC Time with G4: **1.6 hrs**
- GPU utilization: **0.2%** (walkers)
- White spaces – *Accumulation* on the CPU

DCA++ Profile [New Code]



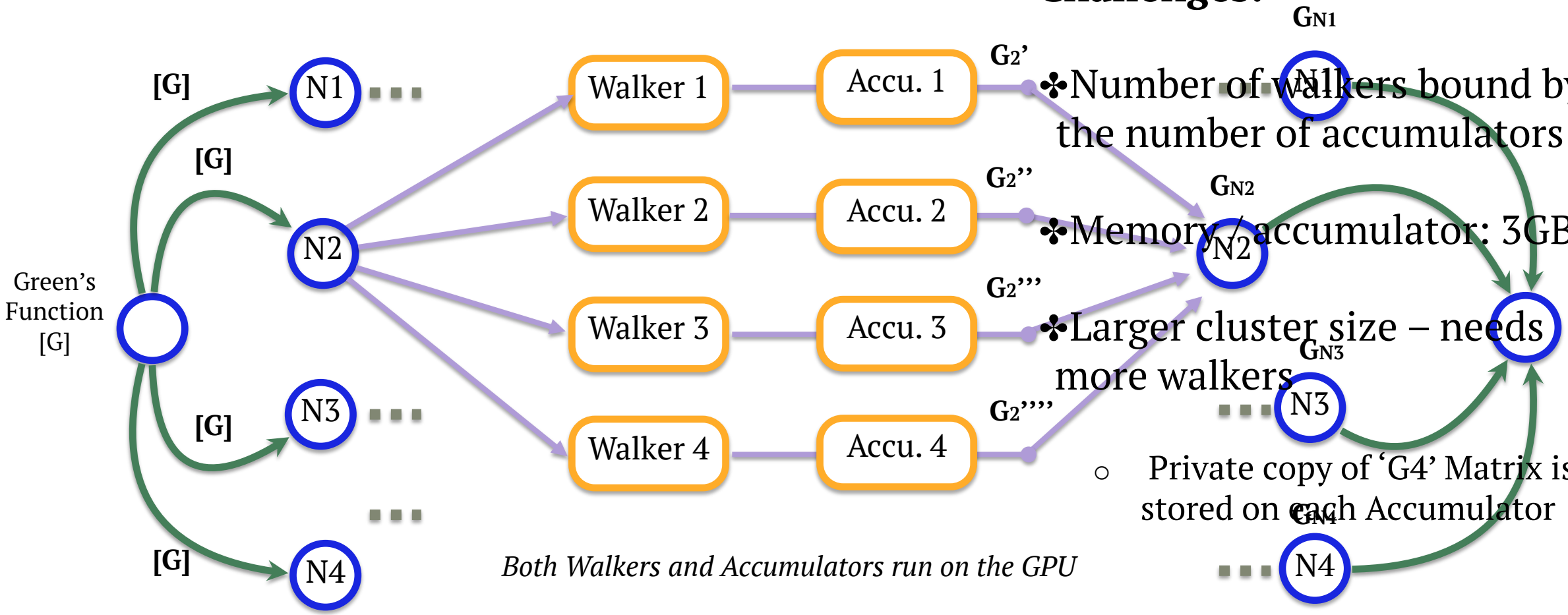
- QMC Time with G4: **80 secs** (**72x** improvement)
- GPU utilization: **37%** (walkers + accumulator)
- Using 8 execution queues on the GPU

DCA++ at scale on Summit (*INCITE*)



- Strong scaling : production run
- Cluster size: 6 ([6 0] [0 6])
  - 80M Measurements
- 1 MPI Process / GPU
- ~ 3000 Measurements / rank
  - **46% GPU** utilization
- Entire Summit (4600 nodes)
  - Old Code – **9.65 hrs.**
  - New 'Improved' Code – **6.5 mins.**
- Sustained performance: **43 PFLOPS**
- Peak performance: 73.5 PFLOPS
- Peak power: **9MW** [for 4600 nodes]

Quantum Monte Carlo Solver (new code)



Both Walkers and Accumulators run on the GPU

"Walker – Accumulator" mapping: 1:1

Challenges:

- Number of walkers bound by the number of accumulators
- Memory / accumulator: 3GB
- Larger cluster size – needs more walkers
  - Private copy of 'G4' Matrix is stored on each Accumulator

Ongoing efforts : SciDAC

- Single-band Hubbard model –> *2 / 3 – band Hubbard Model*
- Memory challenges for accumulators
- Use of mixed precision (*half*-, *single*- and *double* precision) [Tensor cores]
- Collaboration with TAU:
  - TAU/CUDA initialization and CPU/GPU timestamp sync. on Power9 + NVIDIA (Summit)
  - Updated CUDA support – multi-GPU nodes
  - TAU data organization – for GPU streams
  - Continuous Integration – performance database (past / future developments)
- *Autocorrelation* time based on system size and number of measurements
  - Time it takes for two measurements to be decorrelated and hence not biased configurations

Optimizing Dynamic Cluster Approximation on the Summit supercomputer

Arghya "Ronnie" Chatterjee

Performance Engineer, Scientific Computing Group,
National Center for Computational Sciences (NCCS), ORNL

chatterjeea@ornl.gov

Aug 8th, 2019