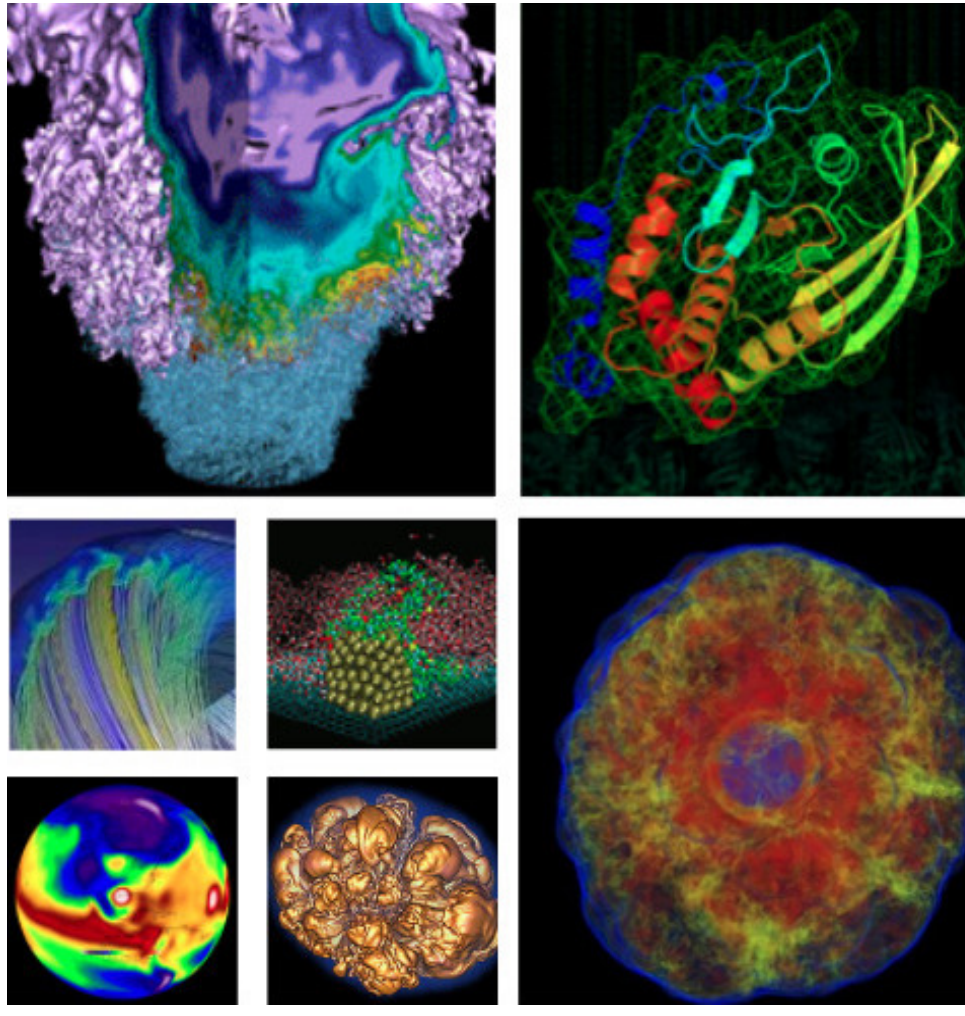
G4Tasking

Status of Task-Level Parallelism



National Energy Research Scientific Computing Center

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Jonathan R. Madsen

✉ jrmadsen@lbl.gov

National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory

September 25, 2019

What is a task?

Ambiguous term for unit of execution

In this context $\Rightarrow$ unit of execution not explicitly assigned to a particular thread

In this context $\Rightarrow$ Function call that is not necessarily immediately executed

Typically, execution is implemented through a "scheduler"

What are the benefits of tasks?

Load-balancing

High-level parallelism $\Rightarrow$ abstracts explicit use of threads

Nested parallelism $\Rightarrow$ task executed on one thread can create more tasks that execute on other threads

- Pool of threads in loop without a predefined call-stack
  - *e.g.*, `while(alive_flag == true)`
- Threads in pool are idle while task-queue is empty
  - *e.g.*, `if(my_queue.empty()) { wait_here(); }`
- Scheduler wakes up (potentially) idle threads
  - *e.g.*, `if(ntasks > nalive) { wake(ntasks - nalive); }`
- Threads pop tasks off of queue and execute the task
  - *e.g.*, `auto t = get_task(); execute_task(t);`
- Repeat (until `alive_flag == false`)

- Intel's Threading Building Blocks (TBB) is a very popular task-based programming library
  - Many constructs: `parallel_for`, `parallel_reduce`, `parallel_do`, `parallel_invoke`, task-groups, pipelines, flow-graph, scalable memory allocators
  - Excellent library but it has some drawbacks
    - Relatively large overhead
    - *Another* external dependency for Geant4 $\Rightarrow$ except more CLHEP-like instead of Qt, Xerces, etc.
- OpenMP has supported "tasks" since version 3.0
  - No . . . See last year's slides

- C++11 has a number of features that make generic tasking relatively easy
  - `std::packaged_task`
  - `std::future`
  - `std::promise`
  - `std::forward`
  - `std::function` and lambdas
  - variadic templates
- C++20 supports coroutines ⇒ stackless function calls that can be suspended and resumed later
  - Significantly simplifies recursive task scheduling/processing

- Elimination of `G4RunManager` vs. `G4MTRunManager`
  - *e.g.*, `using G4MTRunManager = G4RunManager;`
  - No threads in pool? Immediately execute instead of scheduling
- MT Visualization can use nested parallelism capabilities
- Sub-event parallelism can use nested parallelism capabilities
- Multiple thread pools for hardware off-loading

- Generic tasking with native C++ with TBB backend, if desired
- Ideal setup for performance comparison
- `G4TBBTaskGroup` calls `tbb::task_group::run(...)` and `tbb::task_group::wait()` on internal instance of `tbb::task_group`
- `G4ThreadPool` instance does not create any threads
- `G4TaskGroup` instance has internal `tbb::task_group`
- Essentially, forwards task to TBB scheduler instead of internal scheduler

- Separated into stand-alone library
  - github.com/jrmadsen/PTL
  - Include directly in Geant4, *e.g.*, similar to CLHEP
- Implementation:
  - gitlab.cern.ch/jmadsen/geant4-tasking
  - Implements a separate `G4TaskRunManager`
  - Target: create "global" (collaboration-only) repo for Geant4 R&D Task-Force and add it there
    - Similar to gitlab.cern.ch/geant4-dev/geant4
    - *e.g.*, gitlab.cern.ch/geant4-dev/geant4-research-dev

- Begin transition to singular `G4RunManager`
- Provide callbacks within PTL to support the Geant4 virtual overloading
  - Callback replacement will be invisible to user-applications
- Create/integrate support for Geant4 sub-event parallelism
- Investigate using the "backend" capabilities that provides TBB support and apply to Kokkos tasking framework
  - Preliminary analysis of Kokkos tasking code structure suggests this is possible