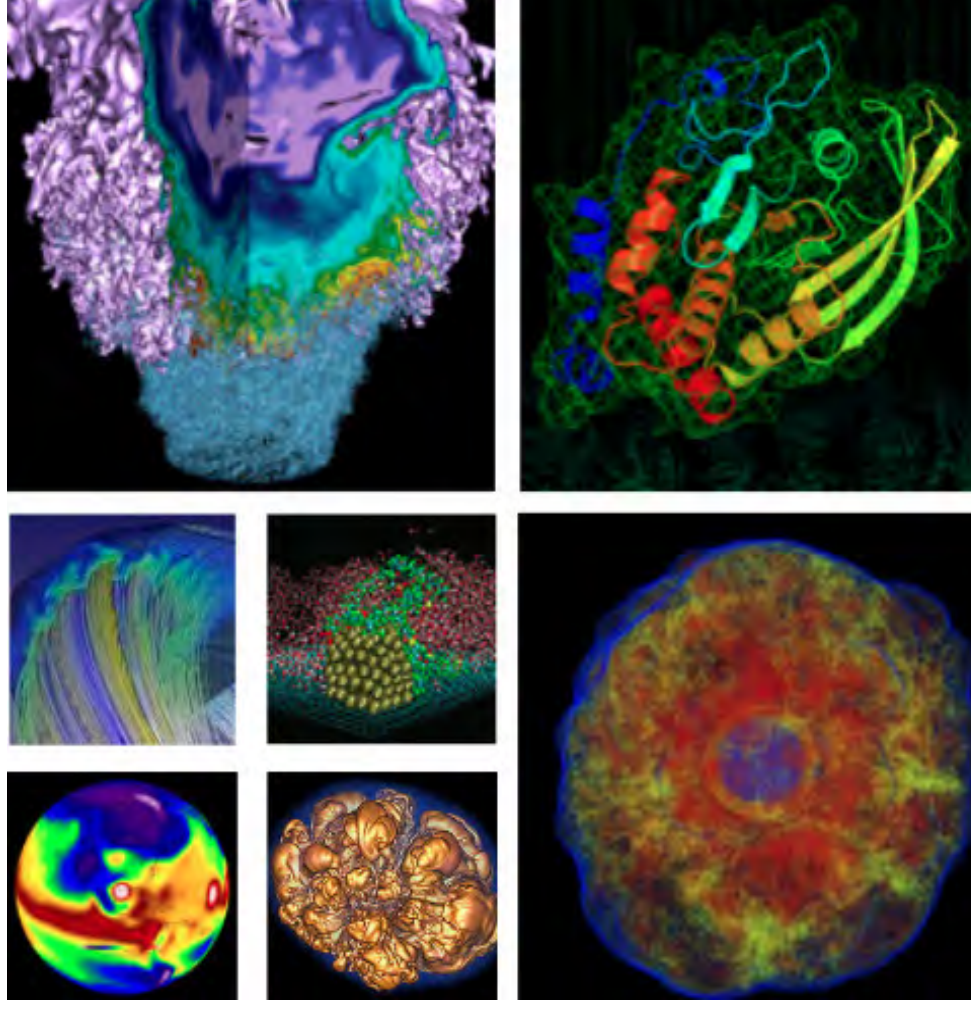
TaskWorks: A task engine for HPC applications

ECP ExaIO-HDF5 Project



Kaiyuan Hou, Quincey Koziol, and Suren Byna

✉ [khl7265,koziol,sbyna]@lbl.gov

National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory

July 22, 2020

TaskWorks: Asychronous tasks for HPC applications

- Task Engine
  - User-defined task dependencies
  - Managed thread pool, runs tasks in the background
  - Optional threadless (polled) task execution
- Event Monitor
  - File operations
  - Socket (internet) events
  - Timers
  - MPI communication
  - Task-related events
- Portable
  - Plugable interface to work with various backends - Argobots, pthreads, etc.

Motivation and Background

- Context and Motivation
  - Support asynchronuous operations in libraries, such as HDF5
  - Handle dependency between operations, e.g.:
    - An HDF5 file must be opened before any groups can be added
    - HDF5 datasets must be opened before they can be written
  - A standalone library to support other libraries and applications.
- Gaps in the state of the art
  - Programming low-level thread APIs is tedious and error-prone
  - Many existing task libraries only work in POSIX environment, but HDF5 (and other middleware) require cross-platform support
  - Desire for enhanced task dependency logic
  - Existing high-level task libraries have a complex API and programing model that is difficult to learn

Why TaskWorks?

- Hide complexity of low-level thread API
  - POSIX/Windows thread API is complex and takes effort to learn
  - Special care required for synchronization
- Portability
  - Thread API differs across operating system
  - Package availability varies across supercomputers
- Convenient - A unified task and event API
  - Easily add asynchronuous operations to existing application / middleware
  - Easy to use - "post-and-forget" interface

Design Goals

- User-friendly interface
  - Simple API: "post and forget"
- Cross-platform
  - Runs on POSIX and Win32
  - Can be built on various backends
    - Argobots, OpenMP, pthreads, Win32 threads ...
- Flexible
  - Custom task dependencies
  - Interaction of task and events
- Designed for HPC environments
  - Coordinate with other packages, such as MPI, for shared system resource
  - Cross-node task dependency

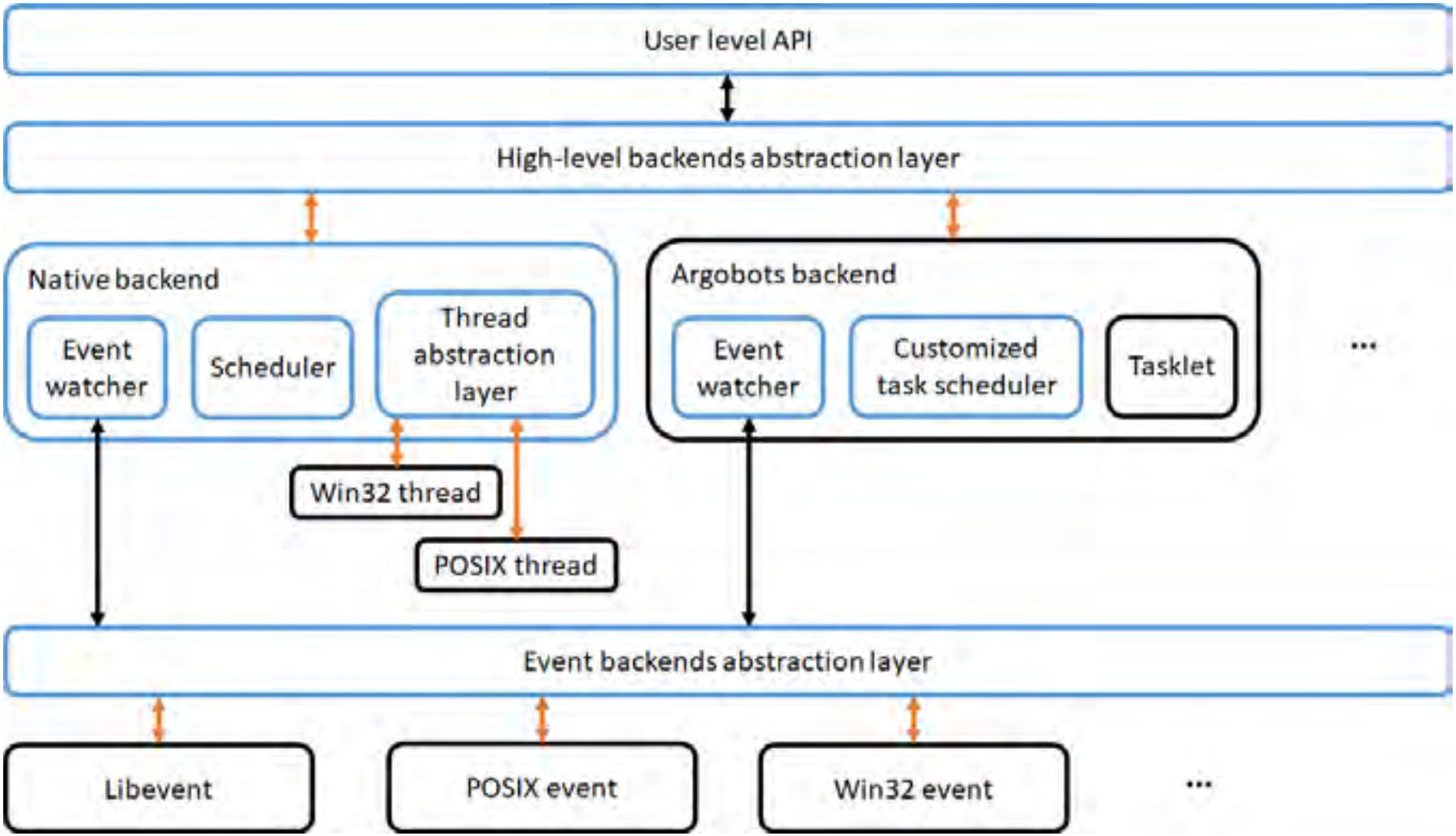
Comparison with similar packages

Feature	TaskWorks	Argobots	OpenMP	Pthreads
Task (high-level) API	Y	Y	N	N
Thread (low-level) API	N	Y	Y	Y
Cross-Platform	Y	N	Y	N
Task Dependencies	Y	Y	N	N
Custom Dependency Logic	**Y**	N	N	N
Threadless (polled) Execution	**Y**	N	N	N
Event Loop Integration	**Y**	N	N	N
MPI Compatibility	Y	Y	Y	Y
PMIx Integration	Y	N	Y	N

TaskWorks Components

- Engine
  - Coordinate tasks and events
  - Pluggable underlying interfaces
    - Adapt to different backends
    - Both high- and low- level
- Tasks
  - User-defined functions run asynchonuously
  - Can depend on other tasks, or events
- Events
  - Run user-defined task when certain events occur
  - Plugable interface for different event sources

Overview


User level API
High-level backends abstraction layer
Native backend
Event watcher
Scheduler
Thread abstraction layer
Win32 thread
POSIX thread
Argobots backend
Event watcher
Customized task scheduler
Tasklet
...
Event backends abstraction layer
Libevent
POSIX event
Win32 event
...

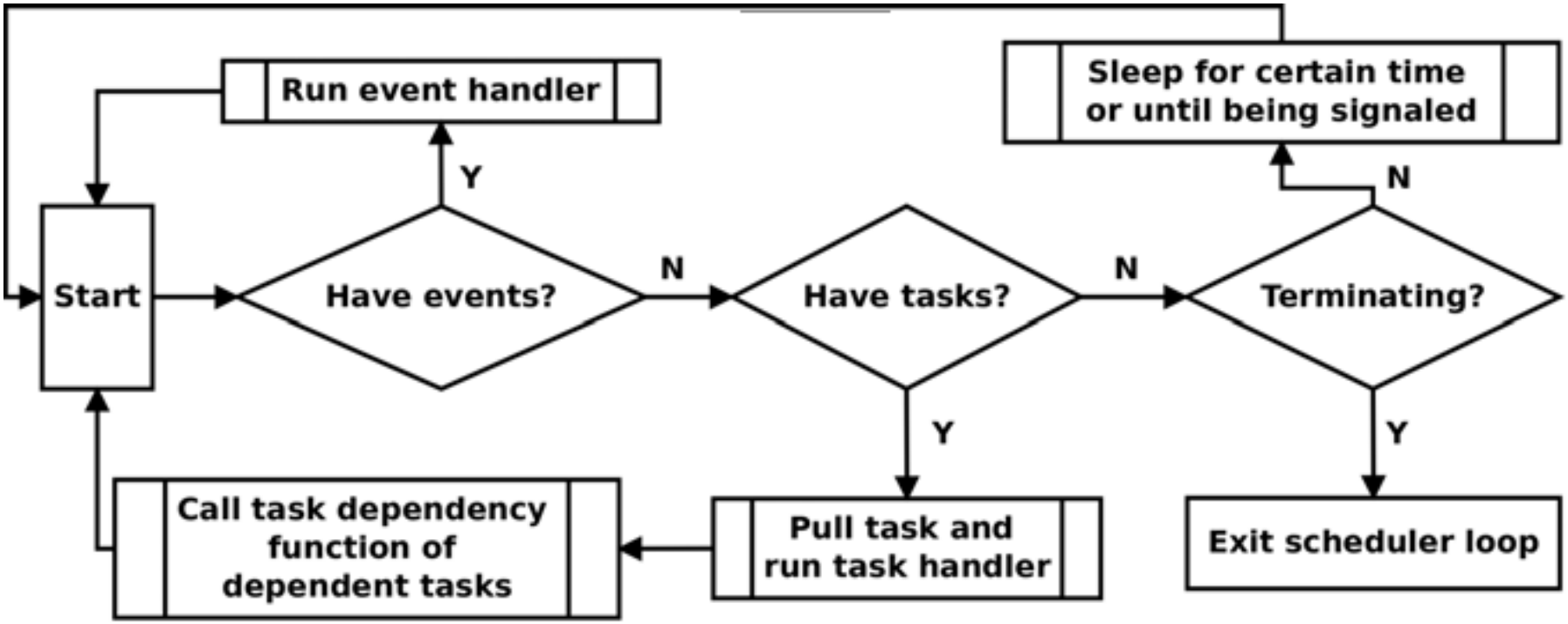
TaskWorks Engine

- Maintains list of watched events
- Controls task dependencies
  - Workflow graph: graph formed by dependency relations of tasks
    - Implicitly represented by dependency list of committed tasks
  - Job queue: A set of ready-to-run tasks that workers pull from
- Contains worker threads to execute tasks in the background
  - Optional, application can run tasks with main thread
  - Number of workers can be dynamically adjusted
- Allows multiple engines running in parallel
  - Dedicate different number of threads for a specific type of task

Tasks

- A function to execute along with an argument list
- List of dependent tasks
  - Whether the task is ready to run depends on the status of dependent tasks.
  - Not limited to tasks handled by the same engine
  - Can depend on tasks on another MPI rank
- Customized task dependency logic
  - A user-provided dependency callback function can decide whether a task is ready to run
  - Does not need to wait for all dependencies to complete
  - Allows cycles in the workflow graph

Task scheduler



Events

- Execute a task when a certain event occurs
- Handled by the worker threads of the engine
  - Priorized over tasks
  - Subject to the availibility of worker threads - can be delayed
- Type of events
  - File system events
  - Socket, MPI events: incoming message
  - Task events: Change of task status
  - Time-based events

OpenMP

- OpenMP will run in parallel alongside TaskWorks
  - Shared CPU resource
  - Need to coordinate resource usage
- Allow OpenMP in task/event functions
  - Avoid the need to breakdown existing parallel functions into tasks

MPI

- Asynchronous communication
  - Utilize background threads
- Collective communication
  - Tasks on all ranks need to start together to prevent blocking of task
  - Cross-rank (process) dependency

PMIx

- Register our resource usage
  - Number of worker threads
  - Status of the workers
- Query OpenMP resource usage
  - Presence of OpenMP
  - Number of thread used by OpenMP
- Query MPI resource usage
  - How many asynchronous communications?

Thank you

- Questions?
- Contact
  - Quincey Koziol - koziol@lbl.gov
  - Kaiyuan Hou - khl7265@lbl.gov