Otter: Towards Data-Driven Taskification of Parallel Programs

A. Tuft [1], T. Weinzierl [1], A. Chalk [2], L. Morgenstern [1], R. Ford [2]

[1] Dept. of Computer Science, Durham University, [2] STFC Hartree Centre, Daresbury Laboratory

December 7, 2023

Research Questions & Project Goals

- Will tasks pay off in my code?
- Can I do better than trial-and-error when adding tasks?
- Do task runtimes make reasonable task scheduling decisions?
- Can user knowledge about the task graph be used to improve schedulers?

PROJECT GOALS

1. Determine task graph at runtime (without major performance costs).
2. Runtime-agnostic task-tracing API.
3. Scheduling simulator to explore possible task schedules.
4. Estimate potential benefits of tasking.
5. Feed scheduling insights back into runtime.

Applications

ExaHyPE
An Exascale Hyperbolic PDE Engine

www.exahype.org

SWIFT

https://swift.strw.leidenuniv.nl/

PSyclone

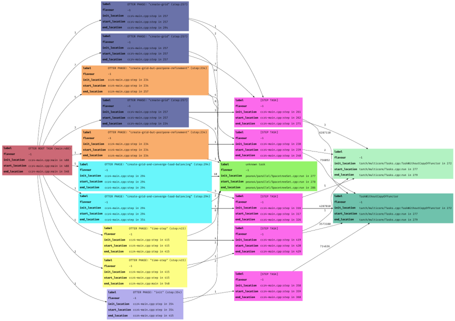
https://github.com/stfc/PSyclone

Current Progress

Scientific Computing Group

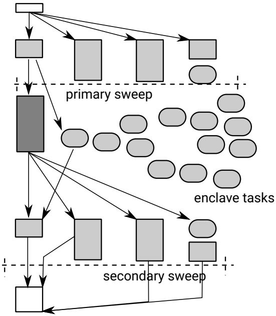
1. Tracing API:
   - → direct annotation
   - → OMPT plugin
   - → Fortran wrapper
2. Post-processing & visualisation
3. Prototype simulator
4. Tracing demonstrated with realistic science workload (ExaHyPE SBH benchmark, 31M events).

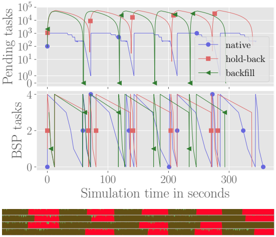
```
EnclaveTask::EnclaveTask(int taskTypeID)
 : Task(taskTypeID) {
  OTTER_DEFINE_TASK(enclave_task,
    OTTER_NULL_TASK,
    otter_add_to_pool,
    "[enclave■task■(type=%d)]",
    taskTypeID);
}

bool EnclaveTask::run() {
  OTTER_REMOVE_FROM_POOL(enclave_task,
    "[enclave■task■(type=%d)]",
    getTaskId());
  OTTER_TASK_START(enclave_task);
  computeTask();
  OTTER_TASK_END(enclave_task);
  return false;
}
```


label OTTER PHASE: "create-grid" (step:257) / flavour -1 / init_location cc24-main.cpp:step in 257 / start_location cc24-main.cpp:step in 257 / end_location cc24-main.cpp:step in 294
label OTTER PHASE: "create-grid" (step:257) / flavour -1 / init_location cc24-main.cpp:step in 257 / start_location cc24-main.cpp:step in 257 / end_location cc24-main.cpp:step in 257
label OTTER PHASE: "create-grid-but-postpone-refinement" (step:234) / flavour -1 / init_location cc24-main.cpp:step in 234 / start_location cc24-main.cpp:step in 234 / end_location cc24-main.cpp:step in 257
label OTTER PHASE: "create-grid" (step:257) / flavour -1 / init_location cc24-main.cpp:step in 257 / start_location cc24-main.cpp:step in 257 / end_location cc24-main.cpp:step in 234
label OTTER ROOT TASK (main:488) / flavour -1 / init_location cc24-main.cpp:main in 488 / start_location cc24-main.cpp:main in 488 / end_location cc24-main.cpp:main in 548
label OTTER PHASE: "create-grid-but-postpone-refinement" (step:234) / flavour -1 / init_location cc24-main.cpp:step in 234 / start_location cc24-main.cpp:step in 234 / end_location cc24-main.cpp:step in 234
label OTTER PHASE: "create-grid-and-converge-load-balancing" (step:294) / flavour -1 / init_location cc24-main.cpp:step in 294 / start_location cc24-main.cpp:step in 294 / end_location cc24-main.cpp:step in 294
label OTTER PHASE: "create-grid-and-converge-load-balancing" (step:294) / flavour -1 / init_location cc24-main.cpp:step in 294 / start_location cc24-main.cpp:step in 294 / end_location cc24-main.cpp:step in 354
label OTTER PHASE: "time-step" (step:415) / flavour -1 / init_location cc24-main.cpp:step in 415 / start_location cc24-main.cpp:step in 415 / end_location cc24-main.cpp:step in 415
label OTTER PHASE: "time-step" (step:415) / flavour -1 / init_location cc24-main.cpp:step in 415 / start_location cc24-main.cpp:step in 415 / end_location cc24-main.cpp:main in 548
label OTTER PHASE: "init" (step:354) / flavour -1 / init_location cc24-main.cpp:step in 354 / start_location cc24-main.cpp:step in 354 / end_location cc24-main.cpp:step in 415
label [STEP TASK] / flavour -1 / init_location cc24-main.cpp:step in 261 / start_location cc24-main.cpp:step in 262 / end_location cc24-main.cpp:step in 271
label [STEP TASK] / flavour -1 / init_location cc24-main.cpp:step in 238 / start_location cc24-main.cpp:step in 239 / end_location cc24-main.cpp:step in 248
label unknown task / flavour -1 / init_location peano4/parallel/SpacetreeSet.cpp:run in 277 / start_location peano4/parallel/SpacetreeSet.cpp:run in 278 / end_location peano4/parallel/SpacetreeSet.cpp:run in 286
label [STEP TASK] / flavour -1 / init_location cc24-main.cpp:step in 316 / start_location cc24-main.cpp:step in 317 / end_location cc24-main.cpp:step in 326
label [STEP TASK] / flavour -1 / init_location cc24-main.cpp:step in 419 / start_location cc24-main.cpp:step in 420 / end_location cc24-main.cpp:step in 429
label [STEP TASK] / flavour -1 / init_location cc24-main.cpp:step in 358 / start_location cc24-main.cpp:step in 359 / end_location cc24-main.cpp:step in 368
label / flavour -1 / init_location tarch/multicore/Tasks.cpp:TaskWithoutCopyOfFunctor in 272 / start_location tarch/multicore/Tasks.cpp:run in 277 / end_location tarch/multicore/Tasks.cpp:run in 279
label TaskWithoutCopyOfFunctor / flavour -1 / init_location tarch/multicore/Tasks.cpp:TaskWithoutCopyOfFunctor in 272 / start_location tarch/multicore/Tasks.cpp:run in 277 / end_location tarch/multicore/Tasks.cpp:run in 279
6287110
756052
4287810
3573180
714636

Observed Tasking Inefficiencies

- ExaHyPE: Exascale hyperbolic PDE engine (OpenMP+MPI).
- Once per timestep, spawn low-priority, ready ”enclave” tasks.
- Intended to allow overlap with MPI communication.
- Observation: task runtime simply consumes ready tasks, so threads later left spinning.





(a) OpenMP tasks consumed eagerly, so few left during communication phase and threads left to spin (in red).

Figure: From H. Schulz, G. Gadeschi, O. Rudyy, T. Weinzierl: Task Inefficiency Patterns for a Wave Equation Solver. IWOMP 2021

Next Steps

Next

- Tracing interface
  - → unify direct & OMPT event sources
  - → add Fortran wrapper
- Prototype scheduling simulator
  - → predict execution time & experiment with scheduling algorithms
  - → validate on simple benchmarks [→] realistic workloads

Later

- Simulator refinements
  - → NUMA effects, user-specified task scheduling algorithm
- Simulator-driven analysis of where tasks pay off
  - → Recommend where to add tasks in realistic application & predict speedup
- Improve runtime performance
  - → Explore whether task-graph information can be used to improve runtime performance.
- Accelerator support
  - → When do *offloaded* tasks pay off?

Project Details

- The authors acknowledge the support through the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (http://www.archer2.ac.uk) under grant no ARCHER2-eCSE04-2, Durham's oneAPI Academic Centre of Excellence made by Intel, ExCALIBUR's Phase Ia grant ExaClaw (EP/V00154X/1) and ExCALIBUR's cross-cutting project EX20-9 Exposing Parallelism: Task Parallelism (grant ESA 10 CDEL).
- https://excalibur.ac.uk/projects/exposing-parallelism-task-parallelism/
- Joint work by Durham's Department of Computer Science, Department of Physics, Durham's Advanced Research Computing and the Hartree Centre