Durham University
Department of Computer Science

Dynamic task fusion with SYCL for an explicit hyperbolic equation system solver with dynamic AMR and local time stepping

ISC 2022

Andrew Mallinson, Adam Tuft, Tobias Weinzierl

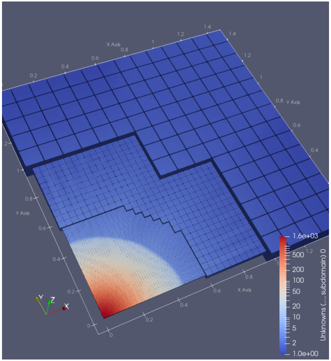
May 9, 2022

Outline

Durham University
Department of Computer Science

Spherical accretion of collisional gas

Durham University
Department of Computer Science



Setup:

- Hubble expansion: expand coordinate system
- Gas: simple Euler equation
- Gravity (with some initial overdensity in the centre): some additional forces

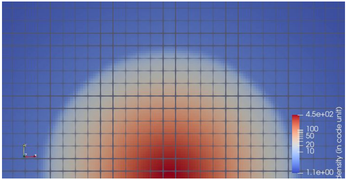
⇒ some turn-around effect

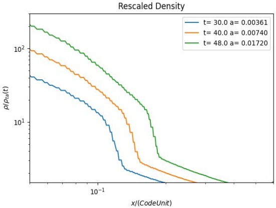
Research question:

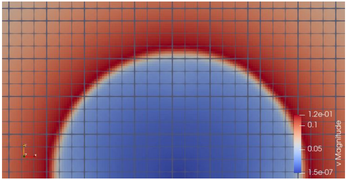
- Maybe *not plain* potential of Poisson equation
- Solution's self-similarity

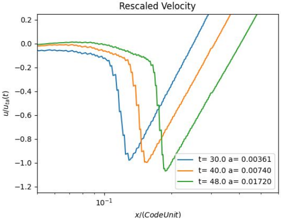
Code requirements

Durham University
Department of Computer Science









- Hydrodynamics: Finite Volumes
- Hubble expansion vs. contraction: AMR around “shock”
- Mass accreditation: Dynamic AMR
- Long-term, very accurate simulation: HPC

AMR dilemma

Durham University
Department of Computer Science

p	Baseline	Patch-wise AoS	Patch-wise SoA	Patch-wise in-situ
3	$1.12 \cdot 10^{-6}$	$8.28 \cdot 10^{-7}$	$8.81 \cdot 10^{-7}$	$\mathbf{4.27 \cdot 10^{-7}}$
4	$9.11 \cdot 10^{-7}$	$8.07 \cdot 10^{-7}$	$8.10 \cdot 10^{-7}$	$\mathbf{3.93 \cdot 10^{-7}}$
7	$7.91 \cdot 10^{-7}$	$7.43 \cdot 10^{-7}$	$7.85 \cdot 10^{-7}$	$\mathbf{3.54 \cdot 10^{-7}}$
8	$7.84 \cdot 10^{-7}$	$7.67 \cdot 10^{-7}$	$7.70 \cdot 10^{-7}$	$\mathbf{3.52 \cdot 10^{-7}}$
15	$7.99 \cdot 10^{-7}$	$7.48 \cdot 10^{-7}$	$7.72 \cdot 10^{-7}$	$\mathbf{3.44 \cdot 10^{-7}}$
16	$7.95 \cdot 10^{-7}$	$7.41 \cdot 10^{-7}$	$7.62 \cdot 10^{-7}$	$\mathbf{3.45 \cdot 10^{-7}}$
3	$1.84 \cdot 10^{-5}$	$1.73 \cdot 10^{-5}$	$1.70 \cdot 10^{-5}$	$\mathbf{1.17 \cdot 10^{-5}}$
4	$1.68 \cdot 10^{-5}$	$1.65 \cdot 10^{-5}$	$1.65 \cdot 10^{-5}$	$\mathbf{1.12 \cdot 10^{-5}}$
7	$1.56 \cdot 10^{-5}$	$\mathbf{1.57 \cdot 10^{-5}}$	$1.56 \cdot 10^{-5}$	$\mathbf{1.02 \cdot 10^{-5}}$
8	$1.55 \cdot 10^{-5}$	$\mathbf{1.70 \cdot 10^{-5}}$	$\mathbf{1.68 \cdot 10^{-5}}$	$\mathbf{1.03 \cdot 10^{-5}}$

Cost per FV update; [t]=s; lower is better; AMD EPYC 7702; 2d (top) vs. 3d (bottom)

Small patches:

- High inter-patch concurrency
- Accurate adaptivity

Large patches:

- High intra-patch concurrency (SIMD)
- Low administration overhead

> Punchline: Algorithms and AMR would want us to use small patches (aka tasks later on). Vector registers (and GPUs later on) would like us to use large (Cartesian) patches.

Outline

An Exascale Hyperbolic PDE solver Engine

Durham University
Department of Computer Science

Vision: Allow groups with decent computational background to write an exascale solver for

$$\boldsymbol{M}\frac{\partial}{\partial t}Q + \boldsymbol{\nabla}\cdot\mathbf{F}(Q) + \sum_i \boldsymbol{\mathcal{B}}_i \frac{\partial Q}{\partial x_i} = \mathbf{S} + \sum \delta$$

within a year.

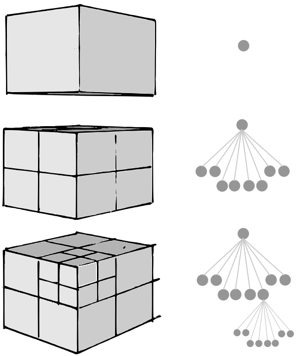
- Engine terminology: You buy into our compute-n-feel and tailor it towards your needs.
- User view: Focus *what* to compute, leave the other stuff to engine (clean software design)
- Software view: Engine decides *how*, *when* and *where* to compute (efficiency)

⇒ Radical (academic) interpretation of separation-of-concerns

Development paradigm: Trade software quality (ease of use, separation of concerns, abstraction, performance portability) for methodological freedom (and that last bit of efficiency).

ExaHyPE 2's engine ingredients

Durham University
Department of Computer Science

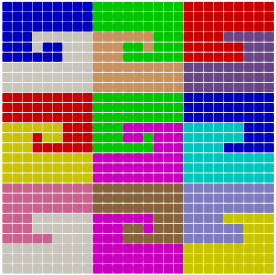


- Spatial discretisation
  - Octree/spacetree formalism for dynamically adaptive Cartesian meshes
  - Block-structured dynamic AMR for low order methods
  - Cell-based AMR for higher order methods
  - ⇒ Peano AMR framework
- Numerical schemes
  - Block-structured Finite Volumes
  - Runge-Kutta DG (experimental)
  - ADER-DG (experimental)
  - Tracer (Particle-in-Cell)
  - SPH (experimental)

    (all explicit)
  - ⇒ ExaHyPE2 layer above Peano
- Target architecture
  - MPI+X
  - OpenMP tasking + OpenMP offloading
  - C++ tasking
  - Intel TBB tasking + SYCL offloading

    (no genuine GPU support; strict offloading/accelerator paradigm)
  - ⇒ Peano's MPI/tasking layer plus ExaHyPE2 compute kernels

Classic domain decomposition: MPI+X



SFC-based non-overlapping domain decomposition:

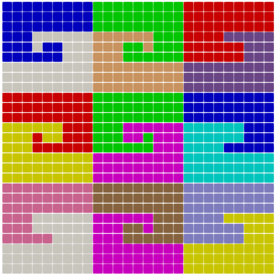
- Peano runs through subpartitions (SPMD+BSP with thread overbooking)
- Logically no difference between MPI and shared memory parallelisation
- Data copying after each traversal
- Load (re-)balancing realised through plug-ins

Separation of concerns:

- You do not know when calculations are triggered (in-between SPMD/BSP sync points)
- You do not know where calculations are triggered (core/rank)
- Consistency code hidden from user
- You do not know how data are distributed (in default mode)

Intra-kernel parallelism (FV/block-structured only)

Durham University
Department of Computer Science



Nested loops over “micro-kernels”:

- Evaluate flux, ncp, source, ... or add outcome of flux, ncp, source, ... to solution
- Exploit knowledge about underlying temporary data structures (AoS vs. SoA vs. AoSoA)
- Available with normal (virtual) and stateless (static) callback to user code

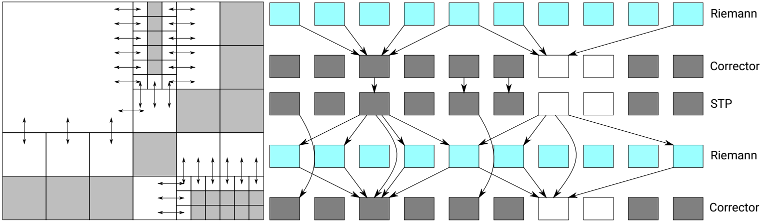
Flavours of overall kernel:

- Loop orderings
- Evaluate all terms first or update in-situ
- Data layout for temporary data (such as flux outcomes)
- Use static or virtual callbacks
- Use C++ Cartesian loops, nested loops with OpenMP annotations, SYCL’s ranges

Durham University
Department of Computer Science

Enclave tasking

Idea: Tasks=intermediate parallelisation layer between SPMD+BSP and kernels

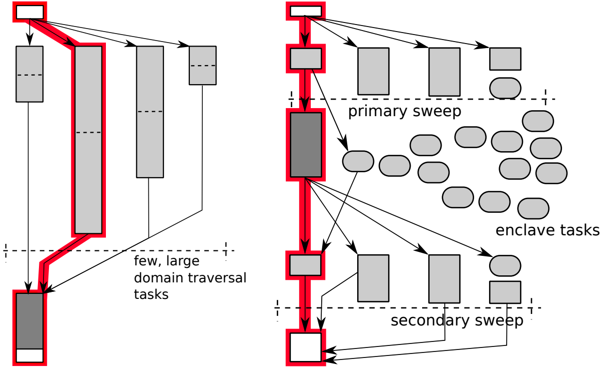


- Mark all cells along MPI boundary and resolution transitions ⇒ skeleton grid
  (those are involved in MPI and might refine/coarsen)
  - reordering of these cells challenging
  - these cells are along critical path in task graph (latency sensitive)
- Remaining cells define real tasks ⇒ skeleton cells
  - overlap with MPI and AMR
  - compensate for BSP imbalances

Outline

Task creation pattern

Durham University
Department of Computer Science



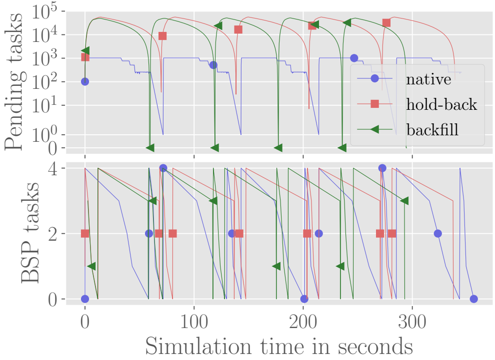
- Primary domain sweep: create task and run the critical ones
- Secondary domain sweep: work in enclave task outcomes

Properties:

- Producer-consumer pattern
- Burst of large number of spawned ready tasks

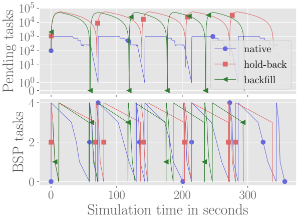
A native task realisation in OpenMP?

Durham University
Department of Computer Science



A native task realisation in OpenMP?

Durham University
Department of Computer Science

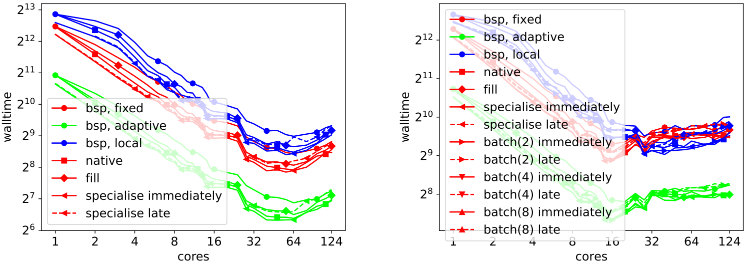


From H. Schulz, G. Brito Gadeschi, O. Rudyy, T. Weinzierl: *Task inefficiency patterns for a wave equation solver*, IWOMP 2021

- ▶ Performance flaws for large meshes and GNU
- ⇒ Process tasks immediately (this is allowed according to standard)
- ▶ Performance flaws for imbalanced BSP, heavy tasks and LLVM
- ⇒ Switch to other heavy task at BSP end and thus make thread unavailable for upcoming urgent tasks
- ▶ Introduce one manual queue and hold back tasks
- ⇒ Performance flaws on NUMA machines (AMD)
- ▶ Introduce one manual queue per core and hold back tasks
- ⇒ Software design (two replicated layers of task queues) and overhead

Task architecture in oneAPI

Durham University
Department of Computer Science



Left: OpenMP, right: oneTBB; AMD EPYC 7702

- SYCL queues are not an option as our tasks have states (virtual function calls)
- oneTBB offers `::tbb::task_group` (direct fit to paradigm)
- Better than OpenMP for small core counts, OpenMP faster for large core counts

Open questions:

- SYCL queues which support virtual functions
- Swap in tasks from oneAPI queues at end of (BSP) task group
- Process only some tasks from group (`backgroundTaskGroup.waitForSomeTasks();`)

Outline

Aggregate multiple tasks

Durham University
Department of Computer Science

- Idea:
  - Label stateless tasks within OneTBB task group with identifier
  - Assemble *k* tasks into one large meta task
- Flavours:
  - Assemble tasks immediately when we span
  - Assemble tasks late when BSP section has nothing else to do
- Opportunities:
  - Reduce pressure on task queues
  - Inline into templated compute kernel
  - Permute loops once more

  ⇒ Vectorise over multiple kernel calls

  ⇒ Offload

Batched vs. patch-wise kernels

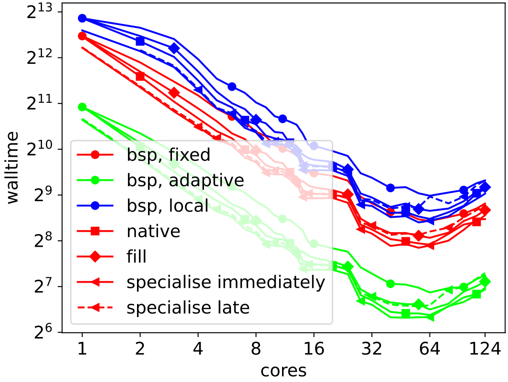
Durham University
Department of Computer Science

Missing; want to finish assessment first

Missing; want to finish assessment first

Performance on host

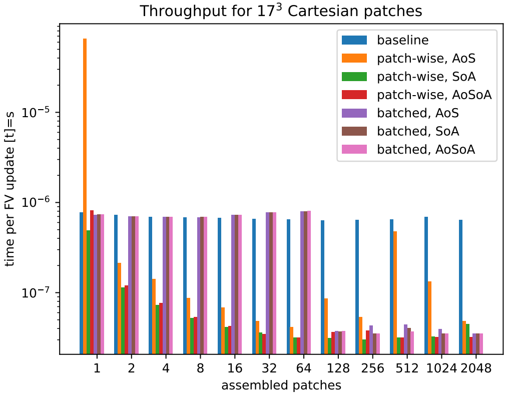
Durham University
Department of Computer Science



- BSP alone is not a good idea
- Specialisation is expensive, i.e. run immediately
- Fusion into task assemblies does not pay off (not shown)
- Specialisation effect significant for low AI, insignificant for high AI (not shown)

Performance on the GPU (OpenMP)

Durham University
Department of Computer Science



- Task assembly is a must
- Once the task assemblies are large enough, switching batched (multi-kernel) compute routines is an option
- AoS is unfortunate choice for internal (temporary) data structures

Three more things

Durham University
Department of Computer Science

Open issues:

- Issuing SYCL GPU calls from multiple tasks does not work at the moment
- NUMA impact of whole concept not clear
- Balancing between multiple SYCL queues not possible

Outline

Durham University
Department of Computer Science

Summary

Durham University
Department of Computer Science

> Take away: If people tell you that you need reasonably large patches in block-structured AMR to get high performance, you should challenge this statement!

- Task group concept direct match to our software architecture
  - Mirror priorities via hierarchy of task groups
  - Hold back some tasks in dedicated groups
- Open questions
  - Task migration between groups (flag ready tasks and steal tasks)
  - NUMA affinity preserved
  - Process only some tasks rather than all in one rush
- Flaws
  - Having both SYCL queues and task groups is not nice (support virtual calls in SYCL queues)
  - Race condition on GPUs requires manual synchronisation

Acknowledgements

Durham University
Department of Computer Science

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). They furthermore received support through the European Research Council via grant ERC-StG-716532-PUNCA, the STFC Consolidated Grants ST/T000244/1 and ST/P000541/1, and the China Scholarship Council (CSC) studentship at Durham University. This work has made use of the Hamilton HPC Service of Durham University.