Abstract

While we speculate what exascale hardware might look like, state-of-the-art numerics and our machines already diverge. Many new hardware generations or ingredients such as Skylake, manycores or Intel's Optane reduce caches or cache backuping per core, make more and more cores share one interconnect, or introduce additional memory levels with high latency. At the same time, many modern algorithmic paradigms such as multigrid, particle-in-cell or predictor-corrector schemes require irregular, non-continuous, repeated memory accesses. As a result, data assembly, movement and exchange, i.e. communication, become constraining factors when we upscale or tune scientific software. We have to avoid them.

In this talk, we generalise the term communication-avoiding. We make it comprise (i) the reduction of data volume, (ii) the elimination of (meta) data generation, (iii) the reduction of data exchange frequency, (iv) the homogenisation of data access, (v) data access hiding and (vi) the localisation of data transfers. These criteria apply to both classic data exchange between compute nodes as well as data movements on the chip.

Communication-avoiding then tackles the problematic divergence sketched above. While every code might require tailored solutions of its own to become communication-avoiding, we present some algorithmic techniques - for multigrid, particle-in-cell and predictor-corrector schemes - which seem to be generic patterns. They can inspire us how to write communication-avoiding software for various applications. That is the good news. The bad news is: you still have to program you algorithms in the right way.

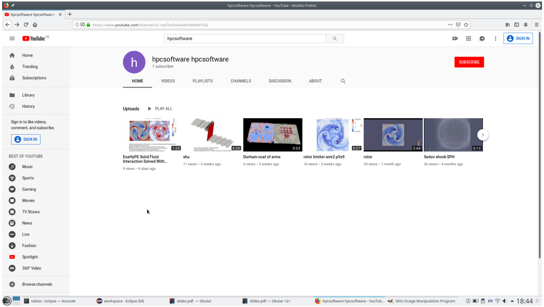
Durham University
Department of Computer Science

Stop talking to me—some communication-avoiding programming patterns

PPAM 2019

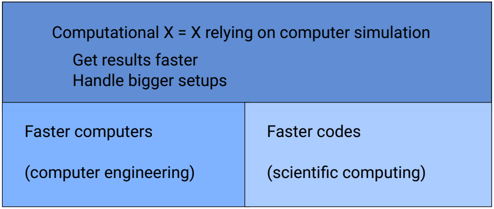
D.E. Charrier, B. Hazelwood, B. Reps, B. Verleye, M. Weinzierl, me and many others

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671698 (ExaHyPE).



https://www.youtube.com/channel/UC1aAT5nSVeAwdYbifdxW1EQ

Why I love to think about algorithms and their implementation . . .



Computational X = X relying on computer simulation

Get results faster
Handle bigger setups

Faster computers

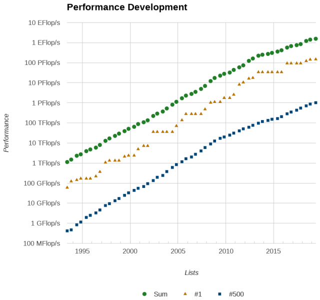
(computer engineering)

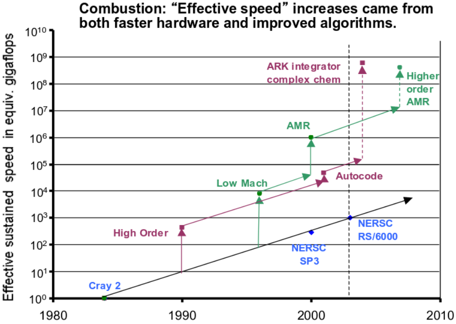
Faster codes

(scientific computing)

C. Johnson (SCI): Before the great discovery was the creation of a new tool!
(from the movie “The Golden Age of Computing”)

The two facets of “better tools”



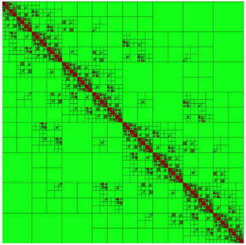


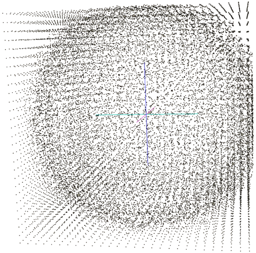
www.top500.org, 2019-09-07 (left)

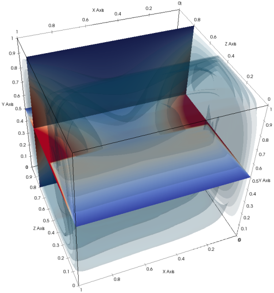
D. Keyes: SCaLeS Report, Vol. 2, 2004 (right)

- Example: transition from Gauss elimination into multigrid is worth 35 years of computer evolution [SCaLeS]
- Both sides improve our tools, yet are not orthogonal

Increasing hardware concurrency impacts software



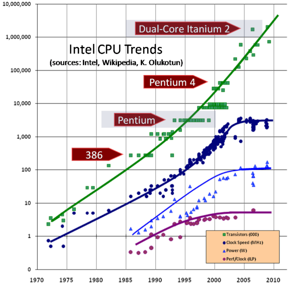




- **Hierarchical (block-structured) sparse matrices**
  dense blocks depending on content (ranks for $\mathcal{H}$; non-zeroes) *and* matrix size
- **Short range forces**
  cut off radius (cell size) depending on interaction potential *and* number of particles
- **Algebraic-geometric multigrid**
  geometric multigrid in smooth, high resolution areas and algebraic multigrid otherwise (*)

(*) See Charles Murray's talk on Monday about lazy operator assembly.

The bitter taste of “co”-design (the Sutter effect)

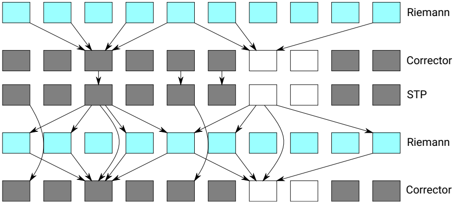


H. Sutter: The Free Lunch Is Over

- Hardware’s evolution has severe impact on software/algorithms
- Don’t be naïve: hardware poses challenges to algorithms(*)
- Sometimes, for some problem sizes, “inferior” algorithms are superior
  (cmp. previous slide)
- And we haven’t even started to discuss ML/NNs
  (even though they drive the HW evolution)

(*) https://www.hpcwire.com/2016/07/12/isc-workshop-tackles-co-developments-thorny-challenges

Two flavours of change management



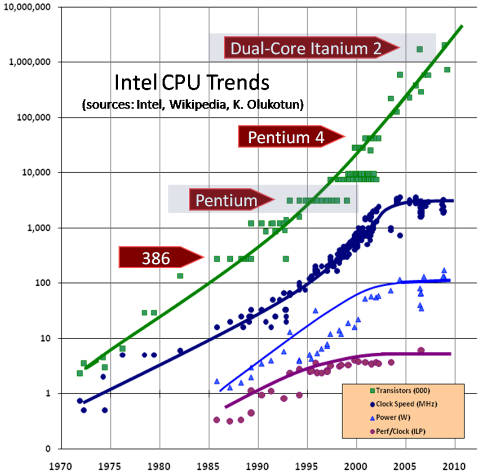
Observation: The paradigm shift on the hardware side has recalibrated and changed the correlation of FLOP numbers to performance.

State of the art:

- Divide et impera: decompose problems into tasks
- Balance/recalibrate against new cost model: rethink “less efficient”
- All important, but . . .

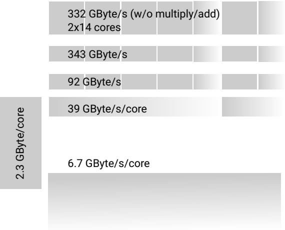
Where are the “exascale” codes?

Some detail is missing here



Switching from massive roasts to tappas makes no free lunch!

Data movements become showstopper



Left: Node of SuperMUC Phase 2; right: SuperMUC ©IBM

Vertical New memory layers (Optane), new cache modi (Skylake)

Horizontal More cores per node, cache and network competition

The next big jumps on the algorithm side

eliminate communication between computer components

= introduce communication-avoiding algorithms

but don't give up on the cool mathematics/algorithms

Outline

Motivation

Demonstrators

Volume reduction

Single-touch implementations

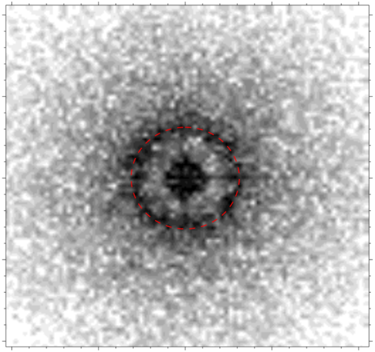
Reduce synchronisation frequency

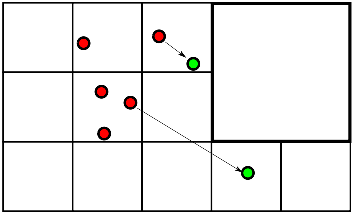
Homogenisation

Localisation

Wrap-up

Particle-In-Cell (PIC)

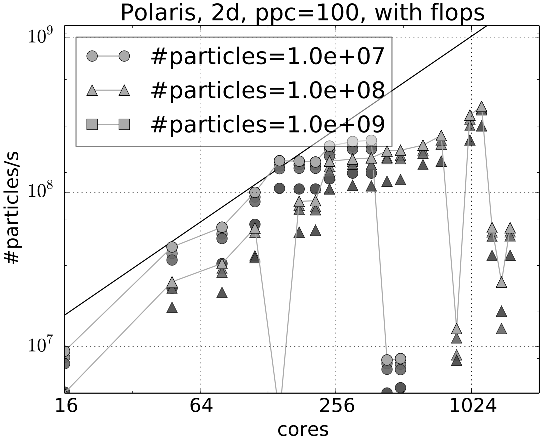




1. Solve PDE on (dynamically adaptive) mesh
2. Interpolate PDE solution onto particles
3. Move particles (suprathermal)
4. Restrict to PDE's rhs

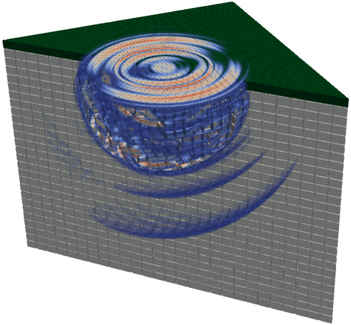
Weinzierl, T. *The Peano software—parallel, automaton-based, dynamically adaptive grid traversals*. ACM Transactions on Mathematical Software (TOMS), 45(2), 14:1–14:41, 2019, arXiv:1506.04496.
Weinzierl, T., Verleye, B., Henri, P., Roose, D. *Two Particle-in-Grid realizations on Spacetrees*. Parallel Computing 52, 42–64, 2016. arXiv:1508.02435
Eckhardt, W., Glas, R., Korzh, D., Wallner, S., Weinzierl, T. (2016), *On-the-fly memory compression for multibody algorithms*, in Joubert, G.R., Leather, H., Parsons, M., Peters, F., Sawyer, M. eds, Advances in Parallel Computing 27: International Conference on Parallel Computing (ParCo) 2015. Edinburgh, Scotland, IOS Press, Amsterdam, 421–430

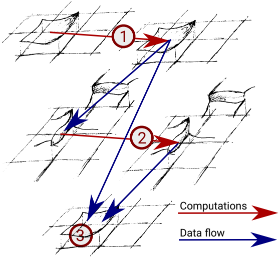
PIC (without any computations here)



Disclaimer: If we added computations, drops would not be that severe.

ADER-DG

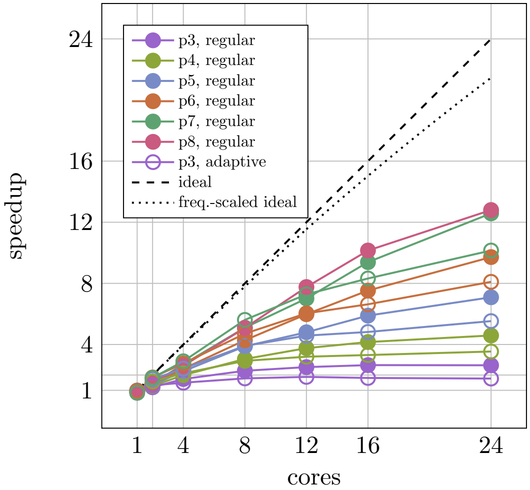




1. Predictor (non-linear iterate, implicit space-time)
2. Riemann
3. Corrector

Charrier E. D., B. Hazelwood, Weinzierl T. *Enclave Tasking for DG Methods on Dynamically Adaptive Meshes*. arXiv:1801.08682
Dumbser M., Fambri F., Tavelli M., Bader M., Weinzierl T. *Efficient implementation of ADER discontinuous Galerkin schemes for a scalable hyperbolic PDE engine*. arXiv:1808.03788
D.E. Charrier, B. Hazelwood, E. Tutlyaeva, M. Bader, M. Dumbser, A. Kudryavtsev, A. Moskovsky, T. Weinzierl: *Studies on the energy and deep memory behaviour of a cache-oblivious, task-based hyperbolic PDE solver*. International Journal of High Performance Computing Applications, 33(5), 973–986, 2019, arXiv:1810.03940.

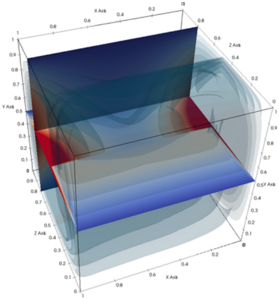
Straightforward ADER-DG

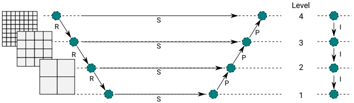


Disclaimer: If we hadn't used a fine-tuned, optimised code (minimal local iteration counts, aggressive optimisation, elimination of temporary storage, . . .), scalability would be way better.

(Additive) Multigrid, BPX, . . .

with complex shift

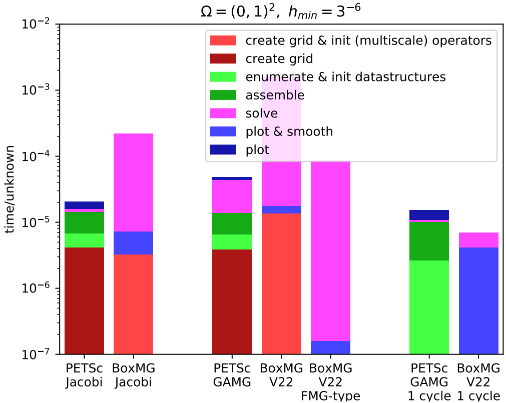




1. Solve on fine level
2. Solve correction on coarser level
3. Prolongate correction and sum up on fine level

C.D. Murray and T. Weinzierl: *Lazy stencil integration in multigrid algorithms*. 13th International Conference on Parallel Processing and Applied Mathematics (PPAM 2019)
Weinzierl, M., Weinzierl, T. Quasi-matrix-free hybrid multigrid on dynamically adaptive Cartesian grids, ACM Transactions on Mathematical Software (TOMS), 44(3), 32:1–32:44, 2018. arXiv:1607.00648
Reps, B., Weinzierl, T. *Complex additive geometric multilevel solvers for Helmholtz equations on spacetrees*. ACM Transactions on Mathematical Software (TOMS), 44(1), 2:1–2:36, 2017. arXiv:1508.03954

(Additive) Multigrid, BPX, . . .

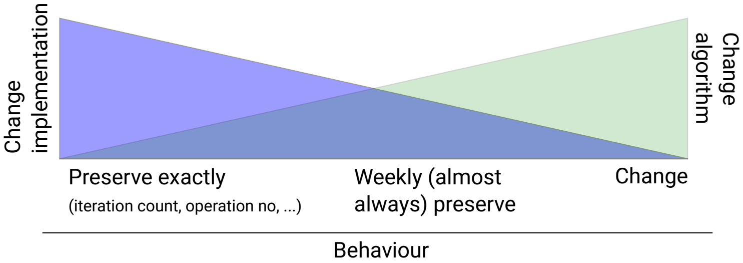


Disclaimer: If we hadn't used totally dynamic AMR and started with a proper fine grid, we would have been faster.

Flavours of communication-avoiding

The root of all evil: We communicate.

Nuances of communication-avoiding programming techniques



“preserve exactly” = refactoring in software engineering [Fowler99]

This talk

This talk: A tour de force through some communication avoiding techniques tailored to my demonstrators(*).

(*) With the potential to pay off in your projects, too.

Outline

Motivation

Demonstrators

Volume reduction

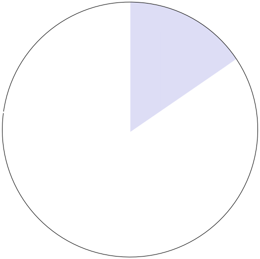
Single-touch implementations

Reduce synchronisation frequency

Homogenisation

Localisation

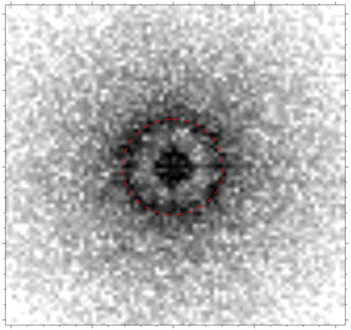
Wrap-up

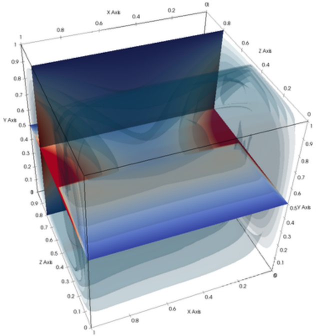


Technique #1: Forget IEEE

(but preserve 64-bit semantics)

Massive memory





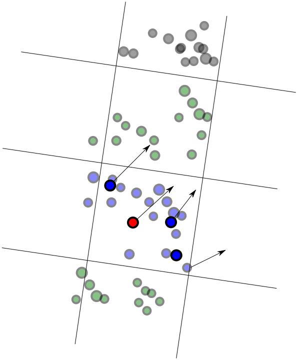
More particles

- ⇒ more accurate physics
- ⇒ higher memory footprint
- ⇒ memory-bound

Algebraic (Ritz-Galerkin+BoxMG) MG

- ⇒ pure geometric operators unstable
- ⇒ matrix-free + rediscretisation fails
- ⇒ massive assembly cost

Information density



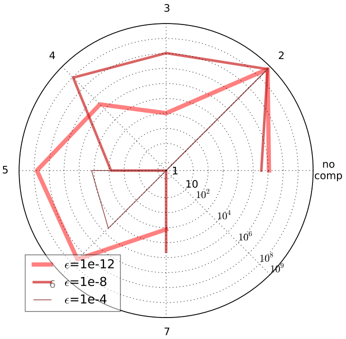
- Particles per cell carry similar values (often)
- Compute average attribute $\widetilde{A}(c)$ per cell
- Encode data hierarchical $\hat{A}(p) = A(p) - \widetilde{A}(c)$ per cell
- Compute on-the-fly how many bytes per attribute are required such that

$$\forall p \in c : |f_{bpa}(\hat{A}(p)) - \hat{A}(p)| \leq \epsilon$$

Reproducible science

```
void peano::heap::decompose( double value, char exponent[8], long int mantissa[8], double error[8] ) {
  int shiftExponent = 6;
  const long int sign = value < 0.0 ? -1 : 1;
  if (sign<0) {
    value = -value;
  }
  int           integerExponent;
  const double  significand = std::frexp(value , &integerExponent);
  for (int i=0; i<8; i++) {
    const double shiftMantissa = std::pow( 2.0,shiftExponent );
    exponent[i]  = static_cast<char>( integerExponent-shiftExponent );
    mantissa[i]  = static_cast<long int>
      ( std::round(significand*shiftMantissa) );
    error[i] = std::abs( std::ldexp(mantissa[i],exponent[i]) - value );
    std::bitset<64>*  mantissaAsBitset =
      reinterpret_cast<std::bitset<64>*>( &(mantissa[i]) );
    if (sign<0) {
      mantissaAsBitset->flip( (i+1)*8-1 );
    }
    shiftExponent+=8;
  }
}
```

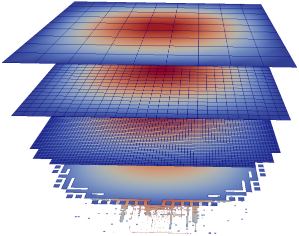
Non-IEEE for particle systems



- `double` values do often not carry enough significant bits compared to their neighbours/averages/recomputed data
- Reduction of memory footprint by factor of four

Eckhardt, W., Glas, R., Korzh, D., Wallner, S., Weinzierl, T. (2016), *On-the-fly memory compression for multibody algorithms*, in Joubert, G.R., Leather, H., Parsons, M., Peters, F., Sawyer, M. eds, Advances in Parallel Computing 27: International Conference on Parallel Computing (ParCo) 2015. Edinburgh, Scotland, IOS Press, Amsterdam, 421–430

(Almost) Matrix-free, geometric-algebraic multigrid



- Operator stencil:
  - Hold $\epsilon(x)/v(x)$ within vertex at $x \in \Omega$
  - Rediscretisation stencil $S_{geom}(\epsilon)$ cheap
  - Determine $\hat{S} = S - S_{geom}(\epsilon)$
  - Store $\hat{S}$ in-between iterations
  - Convert back to S prior to usage
  - $\Rightarrow$ no change of computational kernels
  - $\Rightarrow$ $\hat{S}$ holds few significant bits
- Inter-grid transfer operator stencils:
  - Let $P_{d-linear}$ be d-linear operator
  - Determine $\hat{P} = P - P_{d-linear}$
  - Determine $\hat{R} = R - P^T_{d-linear}$
  - Store $\hat{P}$ and $\hat{R}$ in-between iterations
  - Convert back before we use operators
  - $\Rightarrow$ no change of computational kernels
  - $\Rightarrow$ $\hat{P}$ and $\hat{R}$ hold few significant bits
- Store $f_{bpa}(.)$ (for given accuracy ϵ_{mf}) if

$$|\hat{.} - f_{bpa}(\hat{.})| \leq \epsilon_{mf}$$

$_{bpa}$: bytes per attribute

Memory footprint

h	rediscr.	$\epsilon_{mf} = 1e-2$	$\epsilon_{mf} = 1e-4$	$\epsilon_{mf} = 1e-8$
3^{-3}	2.24e+03	2.24e+03/1.33e-02	2.24e+03/1.99e-02	2.24e+03/3.31e-02
3^{-4}	1.36e+05	1.27e+05/5.68e-03	2.06e+05/8.35e-03	1.36e+05/1.38e-02
3^{-5}	1.37e+05	2.97e+05/4.28e-03	4.57e+05/5.12e-03	1.37e+05/1.39e-02
3^{-6}	2.09e+06	2.92e+06/2.83e-03	2.09e+06/4.39e-03	2.09e+06/7.29e-03
3^{-7}	2.09e+06	4.80e+06/2.82e-03	2.09e+06/4.39e-03	2.09e+06/7.29e-03
3^{-3}	1.08e+04	5.10e+03/1.68e-02	7.75e+03/2.53e-02	2.33e+04/4.10e-02
3^{-4}	2.43e+04	2.41e+04/9.57e-03	2.43e+04/1.42e-02	2.43e+04/2.25e-02
3^{-5}	3.43e+05	2.05e+05/1.20e-02	3.43e+05/1.64e-02	3.43e+05/2.67e-02
3^{-6}	1.84e+07	1.78e+07/1.14e-02	1.60e+06/1.62e-02	1.84e+07/2.51e-02
3^{-7}	1.45e+07	1.86e+07/1.14e-02	1.60e+06/1.62e-02	1.45e+07/2.56e-02

$d = 2$ Poisson equation (top) vs. “circular flow” material parameter (convection), i.e. worst-case setup (bottom); memory footprint/relative footprint

- Effective memory footprint close to two doubles/dof (unknown plus rhs)
- Idea applies to hierarchical vs. nodal values
  (other paper; further compression)

⇒ Algebraic multigrid with geometric multigrid’s memory footprint

Weinzierl, M., Weinzierl, T. Quasi-matrix-free hybrid multigrid on dynamically adaptive Cartesian grids, ACM Transactions on Mathematical Software (TOMS), 44(3), 32:1–32:44, 2018. arXiv:1607.00648

Outline

Motivation

Demonstrators

Volume reduction

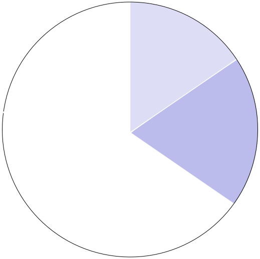
Single-touch implementations

Reduce synchronisation frequency

Homogenisation

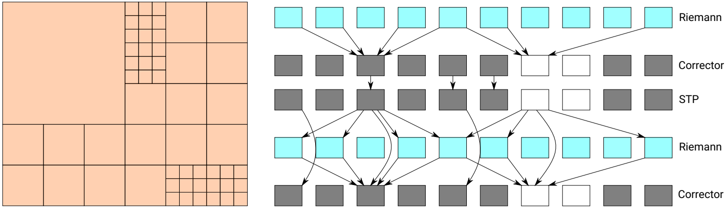
Localisation

Wrap-up



Technique #2: Bring operations forward

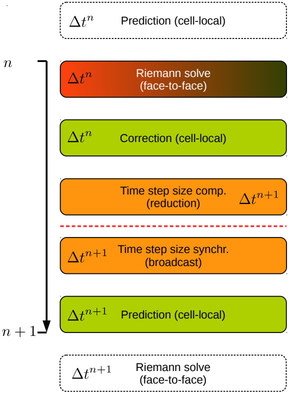
HPC’s lingua franca: ADER-DG as task graph



Observations:

- ADER-DG describes a task pattern
- Mesh instantiates task graph from pattern

Shifted tasking


Δt^n Prediction (cell-local)
n
Δt^n Riemann solve (face-to-face)
Δt^n Correction (cell-local)
Time step size comp. (reduction) Δt^{n+1}
Δt^{n+1} Time step size synchr. (broadcast)
Δt^{n+1} Prediction (cell-local)
$n+1$
Δt^{n+1} Riemann solve (face-to-face)

For the time

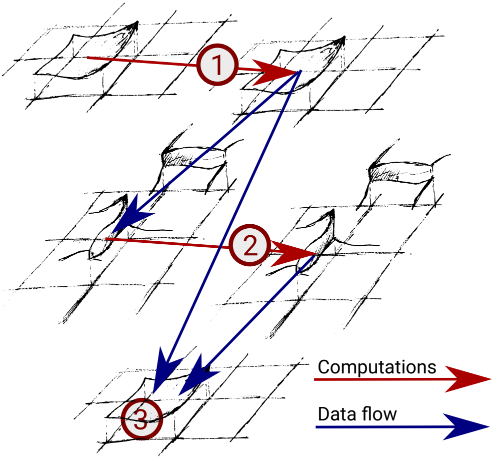
Shifted algorithm:

- First read of face: trigger Riemann
- Enter cell: correct solution
  (all $2d$ faces are read already)
- Trigger subsequent STP immediately

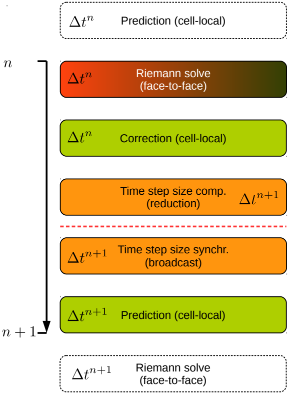
Observations:

- Shifted execution model
- ⇒ Bring tasks forward
- Only two (logical) kernels (cell+face)
- ⇒ Task-fusion
  (incl. elimination of memory and memory accesses)
- One time step per sweep
- ⇒ Amortised single-touch semantics

Illustration



Shifted tasking


Δt^n Prediction (cell-local)
n
Δt^n Riemann solve (face-to-face)
Δt^n Correction (cell-local)
Time step size comp. (reduction) Δt^{n+1}
Δt^{n+1} Time step size synchr. (broadcast)
Δt^{n+1} Prediction (cell-local)
$n+1$
Δt^{n+1} Riemann solve (face-to-face)

For the time

Shifted algorithm:

- First read of face: trigger Riemann
- Enter cell: correct solution

  (all $2d$ faces are read already)
- Trigger subsequent STP immediately

Observations:

- Shifted execution model

⇒ Bring tasks forward

- Only two (logical) kernels (cell+face)

⇒ Task-fusion

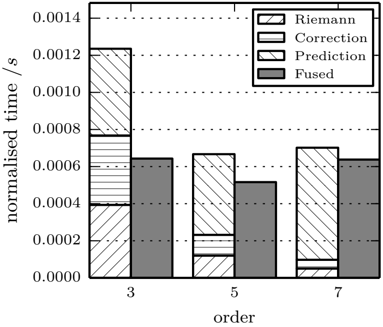
  (incl. elimination of memory and memory accesses)

- One time step per sweep

⇒ Amortised single-touch semantics

Impact of fused timestepping

Durham University
Department of Computer Science



- High impact for low orders (Finite Volumes)
- Take care: there's an easter egg here (potential bug) if you use this for non-linear settings

Charrier E. D., Weinzierl T. *Stop talking to me—a communication-avoiding ADER-DG realisation*. arXiv:1801.08682
D.E. Charrier, B. Hazelwood, E. Tutlyaeva, M. Bader, M. Dumbser, A. Kudryavtsev, A. Moskovsky, T. Weinzierl: *Studies on the energy and deep memory behaviour of a cache-oblivious, task-based hyperbolic PDE solver*. International Journal of High Performance Computing Applications, 33(5), 973–986, 2019, arXiv:1810.03940.

Outline

Motivation

Demonstrators

Volume reduction

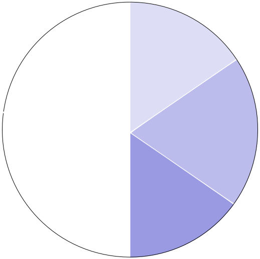
Single-touch implementations

Reduce synchronisation frequency

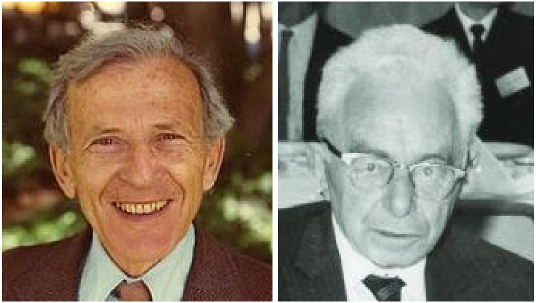
Homogenisation

Localisation

Wrap-up

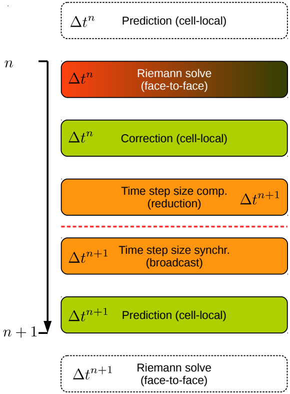


Technique #3: Be optimistic



Lewy and Courant (C) Wikipedia

Revision: Shifted tasking


Δt^n Prediction (cell-local)
n
Δt^n Riemann solve (face-to-face)
Δt^n Correction (cell-local)
Time step size comp. (reduction) Δt^{n+1}
Δt^{n+1} Time step size synchr. (broadcast)
Δt^{n+1} Prediction (cell-local)
$n+1$
Δt^{n+1} Riemann solve (face-to-face)

Shifted algorithm:

- First read of face: trigger Riemann
- Enter cell: correct solution
  (all $2d$ faces are read already)
- Trigger subsequent STP immediately

Observations:

- CFL/λ_{max} condition has to be known
- *h* may not drop suddenly
- Both are global quantities (global time stepping)

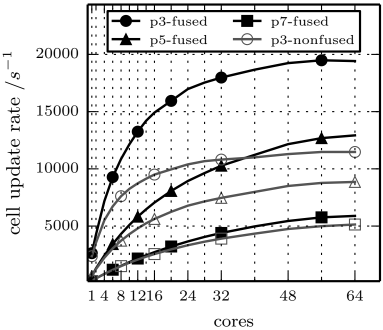
$\Rightarrow$ Maybe not known when we fire STP

Be optimistic and anarchic

- Intermix compute phases over different grid entities
- Bring STP calculations forward
- Fuse correction with STP tasks
- Be optimistic (and eliminate synchronisation):
  - Work with estimate $\Delta t_{est}^{(n+1)}$
  - Determine $\Delta t_{adm}^{(n+1)}$ on-the-fly
  - If admissible $\Delta t_{adm}^{(n+1)} > \Delta t_{est}^{(n+1)}$ use $\Delta t_{est}^{(n+1)} \leftarrow 0.5(\Delta t_{adm}^{(n+1)} + \Delta t_{est}^{(n+1)})$
  - If $\Delta t_{adm}^{(n+1)} < \Delta t_{est}^{(n+1)}$ reset $\Delta t_{est}^{(n+1)} \leftarrow 0.9\ \Delta t_{adm}^{(n+1)}$ and rerun

⇒ Limiter, AMR, time step decreases might make approach fall back to two-sweep paradigm

Some results (not really)



- Failed guesses very infrequently
- Numerical dispersion negligible so far (weekly preserving semantics)

Outline

Motivation

Demonstrators

Volume reduction

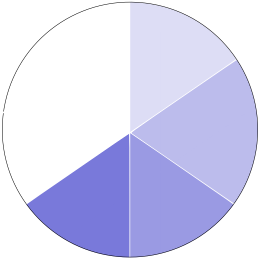
Single-touch implementations

Reduce synchronisation frequency

Homogenisation

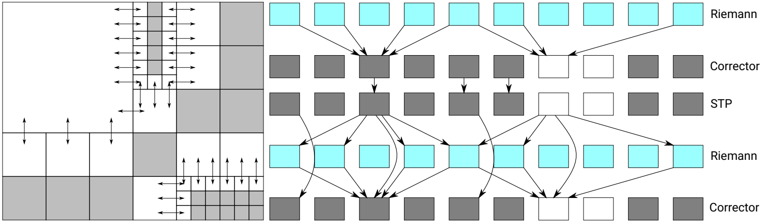
Localisation

Wrap-up



Technique #4: Intermix data-access intesive and compute-intensive tasks

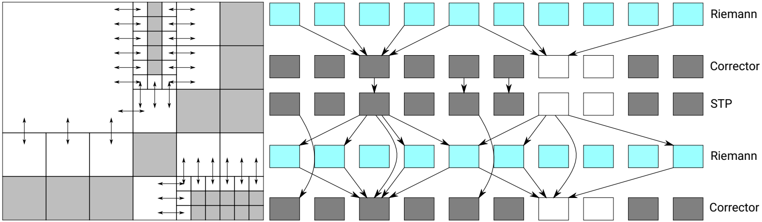
Challenge



Shifting + optimism with classic domain decomposition help . . .

- + Memory- and compute-intense tasks take turns
- - AMR (memory intense) tends to happen towards end of sweep
- - Riemann solves still might run concurrently

Enclave tasking—inspired by Ghattas et al.



- Mark all cells along MPI boundary and resolution transitions ⇒ skeleton grid
  (those are involved in MPI and might refine/coarsen; this is an optimistic assumption)

⇒ reordering challenging

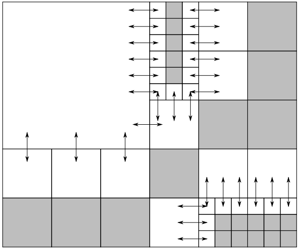
- Give up on idea to run Riemann solves parallel (weakly)

⇒ bandwidth-bound
  (cheap and skeleton operations done immediately and close-to sequentially)

- Make remaining cells (enclave) give us the scaling
  (job stealing makes brings in idling cores)

Ingredient: Prioritise

and re-introduce a second grid traversal

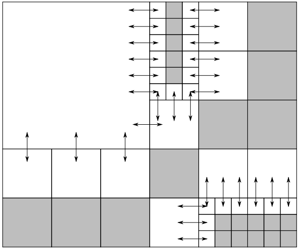


Primary sweep:

- Per face
  - wait for adjacent STPs to finish
  - Riemann solves
  - restrict image along resolution transitions
- Per cell
  - Corrector
  - Dynamic adaptivity, limiter reruns, . . .
  - determine skeletons on-the-fly
- Spawn STP
  - on skeleton: high priority
  - on enclaves: low priority (background)

Ingredient: Prioritise

and re-introduce a second grid traversal



Primary sweep:

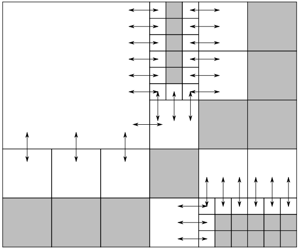
- Per face
  - wait for adjacent STPs to finish
  - Riemann solves
  - restrict image along resolution transitions
- Per cell
  - Corrector
  - Dynamic adaptivity, limiter reruns, . . .
  - determine skeletons on-the-fly
- Spawn STP
  - on skeleton: high priority
  - on enclaves: low priority (background)

Secondary sweep:

- Ignore enclaves (partial sweep)
- Per skeleton cell: wait for STP
- Trigger MPI exchange
- Interpolate along AMR transitions for next Riemann solve

Ingredient: Prioritise

and re-introduce a second grid traversal



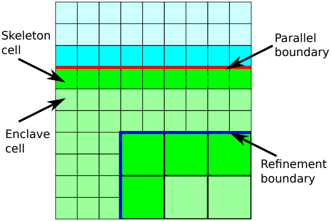
Primary sweep:

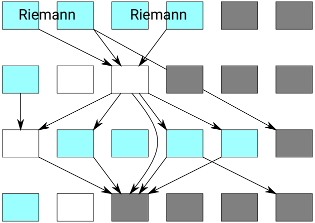
- Per face
  - wait for adjacent STPs to finish
  - Riemann solves
  - restrict image along resolution transitions
- Per cell
  - Corrector
  - Dynamic adaptivity, limiter reruns, . . .
  - determine skeletons on-the-fly
- Spawn STP
  - on skeleton: high priority
  - on enclaves: low priority (background)

Secondary sweep:

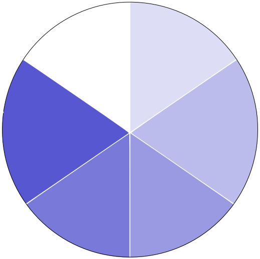
- Ignore enclaves (partial sweep)
- Per skeleton cell: wait for STP
- Trigger MPI exchange
- Interpolate along AMR transitions for next Riemann solve

Enclave implications



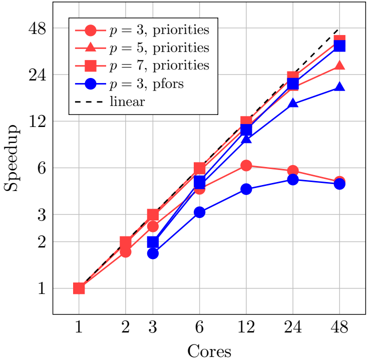


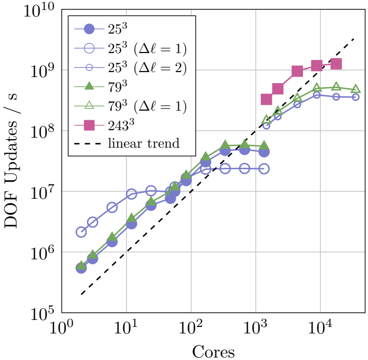
- Low intensity tasks trickle through system
  (homogenise arithmetic load/avoid memory access bursts)
- Coarsening & refinement accompanied by background/enclave STPs
  (memory allocation and initialisation to not throttle everybody else)
- STPs yielding MPI messages ran before majority of (enclave) tasks
  (more time to overlap messaging and processing)



Technique #5: Overlap data movements and computations

Low intensity jobs trickle through system and communication is hidden





Charrier E. D., B. Hazelwood, Weinzierl T. *Enclave Tasking for DG Methods on Dynamically Adaptive Meshes*. arXiv:1801.08682 (submitted)

Outline

Durham University
Department of Computer Science

Motivation

Demonstrators

Volume reduction

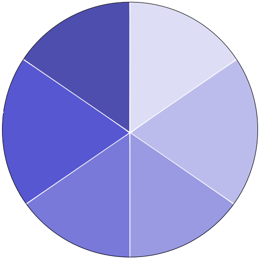
Single-touch implementations

Reduce synchronisation frequency

Homogenisation

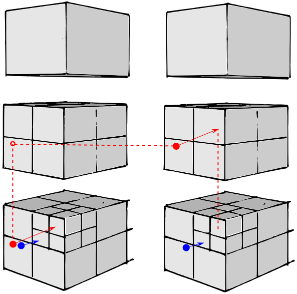
Localisation

Wrap-up



Technique #6: Exploit sparsity of communication graph

The particle code



Single touch through shifts (postpone update)

- Bookmark position update
  (do not apply)
- Apply position update as preamble to next step
  (first touch)
- Update grid-particle association in preamble, too

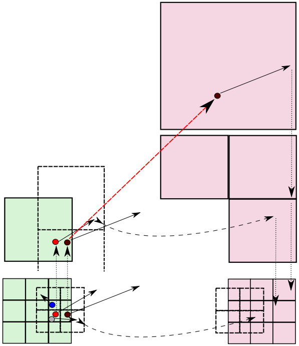
⇒ simple iff particles travel at most one cell per time step

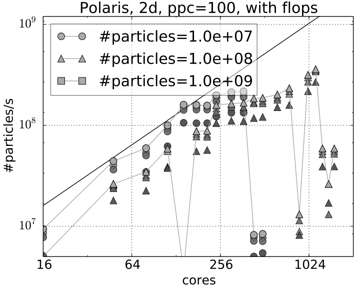
Lift-n-drop mechanism

- Use cascade of grids (octree)
- Particles on finest level for computation
- Otherwise on level such that they drift at most one level

⇒ works but . . .

The particle code



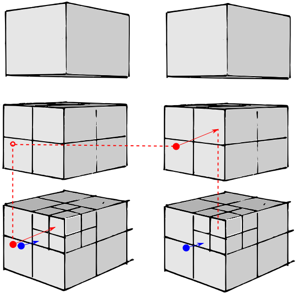


Horizontal data exchange

- Along multiscale domain boundaries (non-blocking MPI hiding behind computation)
- Very fast particles are lifted aggressively
- Non-neighbour MPI ranks involved (global sort)

⇒ Latency-sensitive

Predict data flow



Vertical data flow:

- Max velocity $v_{max}(c)$ per cell $c \in \mathcal{T}$
- Extrapolate velocity updates (fine grid)
- Analysed tree grammar

$$v_{max}(c) = \begin{cases} max_{p \in c}|v(p)| & \text{for leaves} \\ max_{c'}\ v_{max}(c') & \text{otherwise} \\ & \forall\ c' \sqsubseteq_{child\ of}\ c \end{cases}$$

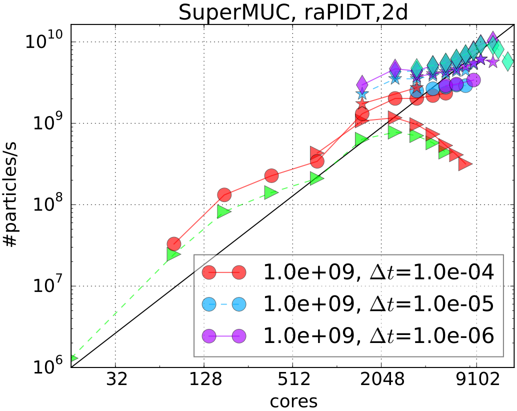
- Next traversal:
  - Particle p drops into domain

    $$v_{max} \leftarrow max(v_{max}, |v(p)|)$$

  - $v_{max} \leq h/\Delta t$: no tunnelling possible and likely not to happen either

  $\Rightarrow$ skip reduction for these rank pairs
  - Security factor for extrapolation

reduction-avoiding Particle-In-Tree



- Triangle: $2.0 \cdot 10^8$ particles; Diamond: $4.0 \cdot 10^9$ particles; Star: $1.0 \cdot 10^{10}$ particles
- Colours encode time step sizes

Weinzierl, T., Verleye, B., Henri, P., Roose, D. *Two Particle-in-Grid realizations on Spacetrees.* Parallel Computing 52, 42–64, 2016. arXiv:1508.02435

Outline

Durham University
Department of Computer Science

Motivation

Demonstrators

Volume reduction

Single-touch implementations

Reduce synchronisation frequency

Homogenisation

Localisation

Wrap-up

My notion of “communication-avoiding”

Techniques presented:

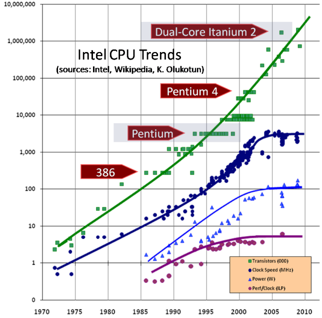
1. reduction of data volume (below-IEEE)
2. reduction of data access frequency (single-touch)
3. reduction of synchronisation (optimism)
4. homogenisation of data access (enclave tasking)
5. data access hiding (enclave tasking)
6. localisation of data transfer (reduction grammar)

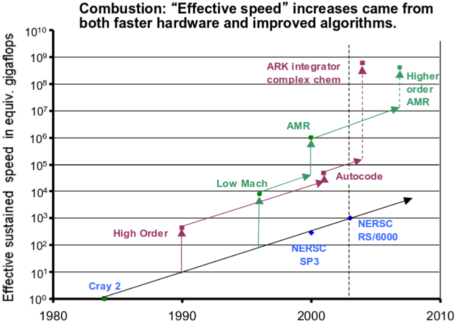
Communication avoiding $\mapsto$ Communication-flaw-avoiding

Properties:

- Generalised interpretation of “communication-avoiding”
- Pay-off through combination of techniques
- One technique often tickboxes multiple criteria
- Applies to both message exchange and on-chip data access
- Might still be incomplete

Context




Combustion: "Effective speed" increases came from both faster hardware and improved algorithms.

H. Sutter: The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software, 2005 (left)

D. Keyes: SCaLeS Report, Vol. 2, 2004 (right)

- I don't know the next big algorithmic thing
- Some ideas had multifaceted impact on various disciplines
  - Think multiresolution
  - Increase smoothness locally, give up on global smoothness
- Time might be right for similar impact of communication avoiding techniques

Famous last words

It is all open source (www.peano-framework.org).
Follow us on Twitter (@hpcsoftware_).

Support & Grants

- Peano (www.peano-framework.org) has been developed for more than 10 years under various grants made by Technische Universität München (TUM), KAUST, German Research Foundation (DFG), Durham University, and many more.
- ExaHyPE (www.exahype.eu) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671698.

  ExaHyPE An Exascale Hyperbolic PDE Engine

  European Commission | Horizon 2020 European Union funding for Research & Innovation
- The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC at Leibniz Supercomputing Centre (LRZ, www.lrz.de).

  lrz Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities
- Experiments were made possible through Durham's supercomputer Hamilton.

  Durham University
- We appreciate early-adopters access to Intel Xeon Phi 7200 (codenamed KNL) family processors through the RSC Group.

  RSC