Tobias Weinzierl, Durham University, United Kingdom

Stop talking to me - some communication-avoiding programming patterns

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.