Algorithms for modern shared-memory parallel architectures

Providing efficient HPC for everybody

Pedro Gonnet
School of Engineering and Computing Sciences, Durham University
SNF Presentation, January 24, 2013

Overview

What my research is all about

Durham University

- In order to continue getting *faster*, programs need to become *more parallel*.
- Unfortunately, most current parallel approaches do not scale well.
- Better exploiting parallelism is possible, but requires completely re-thinking the algorithms we use.
- Doing things right can lead to performance gains of more than a factor of ten.
  $\longrightarrow$ Better use of existing and future infrastructure.
- Applied to interdisciplinary problems, which provide interesting computational questions, and require close collaborations to produce immediately useable software.
- In summary: Providing efficient High-Performance Computing for everybody.

Not faster, but more parallel

Increasingly parallel hardware

Durham University

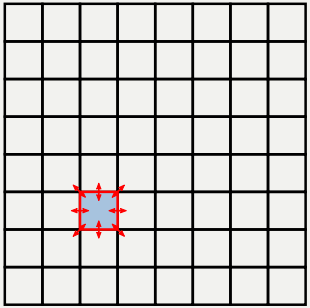
- Until a few years ago, we could rely on computer processors getting faster every year (Moore's Law).
- Now, due to the physical limitations on individual processor speeds, computers are not getting *faster*, but *more parallel*.
- Instead of relying on increasing processor speeds to make codes continuously faster, we now need to rely on exploiting increasing parallelism.
- As a consequence, many problems that do not scale well or have reached their maximum degree of parallelism, won't get any faster. Ever.

Not faster, but more parallel

Distributed-memory parallelism

Durham University

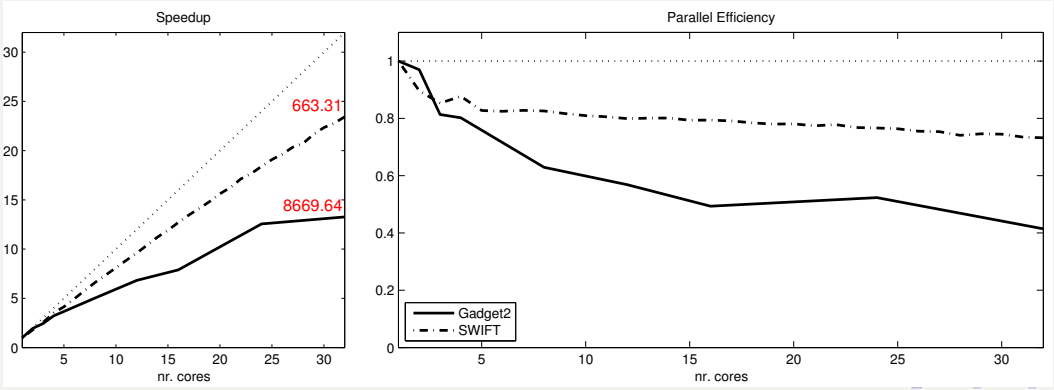
- Distributed-memory parallelism, e.g. using MPI, is based on data decomposition, i.e. each processor is assigned part of the problem to work on and communicates with its neighbours.
- Surface-to-volume ratio problem: As the number of cores increases, the amount of computation per core (volume) decreases while the relative amount of communication (surface) increases.
- The entire computation is then dominated by communication, thus seriously degrading parallel efficiency.



Case in point

Good vs. bad scaling

- Cosmological simulation (galaxy formation) with 1.8 M particles in a cubic box of 6.25 Mpc on a 4 $\times$ Intel Xeon X7550 with 32 cores, 2 GHz.
- GADGET-2, MPI-based, used for multi-billion particle simulations.
- SWIFT, my own code for the same problem. 13$\times$ faster on 32 cores.



Case in point
Performance benefits

Durham University

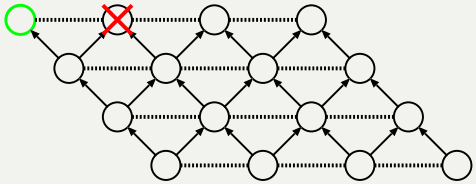
- Better value for money from existing computing infrastructure.
- Lowering the hardware costs makes HPC accessible to a much wider audience.
- Faster generation of research outputs in the application domain.
- Ability to study new problems beyond the range of what is currently possible.

Task-based parallelism

What it is

Durham University

- Shared-memory parallel programming paradigm in which the computation is formulated in an implicitly parallelizable way that automatically avoids most of the problems associated with concurrency and load-balancing.
- We first reduce the problem to a set of inter-dependent tasks.
- For each task, we need to know:
  - Which tasks it depends on,
  - Which tasks it conflicts with.
- Each thread then picks up a task which has no unresolved dependencies or conflicts and computes it.



Task-based parallelism

Why it is good

Durham University

- The order in which the tasks are processed is dynamic and adapts automatically to load imbalances.
- If the dependencies and conflicts are specified correctly, we do not have to worry about concurrency at the level of the individual tasks.
  $\longrightarrow$ No need for expensive explicit locking, synchronization, or atomic operations.
- The same approach can be applied to more unconventional many-core systems such as general-purpose Graphics Processing Units (GPUs), making code reuse between different platforms easier.
- However, this usually means that we have to re-think our entire computation, e.g. redesign it from scratch to make it task-based.

My work in this area

Four main topics

Durham University

- Task-based computational models, i.e. develop memory-efficient, task-based algorithms for specific computational problems.
- Algorithms for task-based parallelism, i.e. task sorting, task clustering, task selection, and efficient task queues.
- New architectures, i.e. algorithms for emerging high-performance/low-cost many-core architectures, e.g. GPUs, the Intel MIC, or Adapteva's Epiphany IV.
- Hybrid shared/distributed-memory schemes, i.e. use both MPI and task-based parallelism on clusters of multi-cores, interleaving communication and computation such as to hide communication latencies.

My work in this area

Interdisciplinary research and applications

Durham University

- Tight coupling between theory and practice: all algorithms are implemented as part of specific interdisciplinary applications.
- I am currently working on two such software projects:
  - `mdcore`: An flexible and efficient library for shared-memory parallel Molecular Dynamics simulations on multi-cores and GPUs.
  - SWIFT: A hybrid shared/distributed-memory SPH and gravity simulation software for large-scale cosmological simulations.
- Interdisciplinary projects require close collaboration with a counterpart in the domain of application such as to make the results immediately useful.

My work in this area

Research outlook

- Continue work on `mdcore` and SWIFT, adding functionality as it becomes necessary by the application areas.
- Extend both software projects to emerging architectures, e.g. newer GPUs, the Intel MIC, and Adapteva's Epiphany IV architecture.
- Actively look for new interdisciplinary collaborations, both within and outside of traditional high-performance computing areas.

  $\longrightarrow$ Developing new, better, and useful algorithms, both for task-based parallelism in general, and for specific applications.

Conclusions
In summary

Durham University

- In order to continue getting *faster*, programs need to become *more parallel*.
- Task-based parallelism is a simple and efficient, cross-platform paradigm for shared-memory parallel computing.
- Speedups of ten times faster and more are possible.
- Theoretical results applied to interdisciplinary applications which provide interesting computational challenges.
- Much work still needs to be done, both in developing better task-based algorithms…But also in teaching the next generation of Computer Scientists to use them.
- In summary: Providing efficient High-Performance Computing for everybody.

Durham University

Thank you for your attention!