A framework for parallel unstructured grid applications on GPUs

Mike Giles

mike.giles@maths.ox.ac.uk

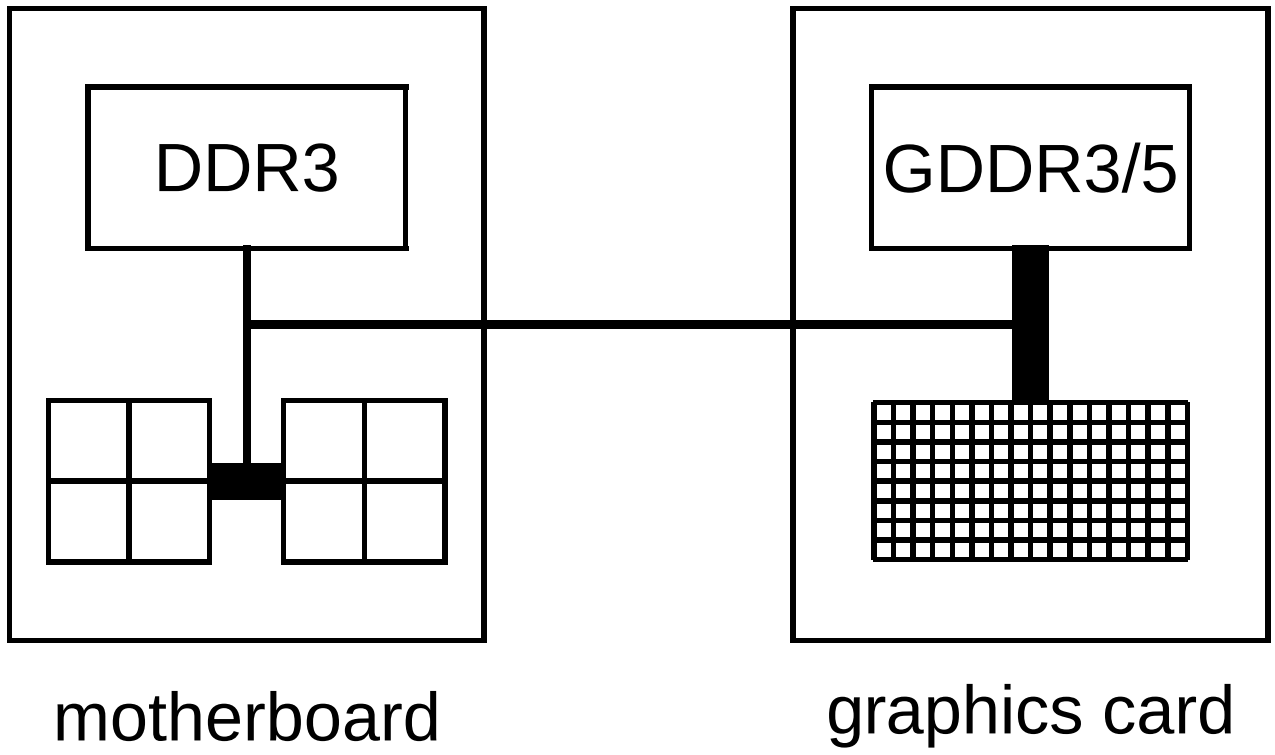
Oxford University Mathematical Institute

Oxford eResearch Centre

CFMS Workshop, March 10, 2010

CPUs and GPUs

Typically, a PCIe graphics card with a many-core GPU sits inside a PC/server with one or two multicore CPUs:



CPUs and GPUs

- CPUs have up to 6 cores (each with a SSE vector unit) and 10-30 GB/s bandwidth to main system memory
- NVIDIA GPUs have up to 30×8 cores on a single chip and 100+ GB/s bandwidth to graphics memory
- offer 50–100$\times$ speedup relative to a single CPU core
- roughly 10$\times$ speedup relative to two quad-core Xeons
- also 10$\times$ improvement in price/performance and energy efficiency

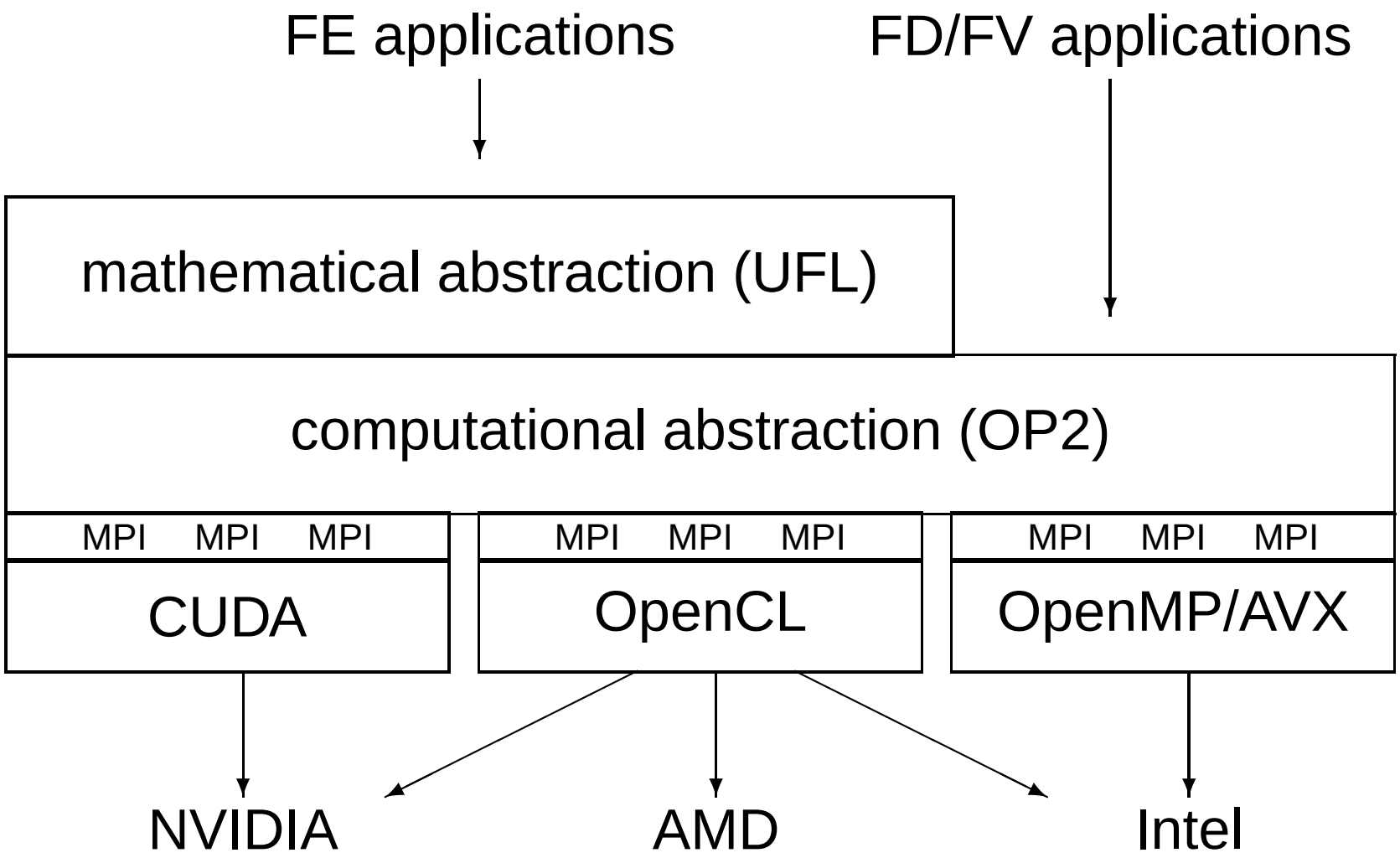
How is this possible? Much simpler cores (SIMD units, no out-of-order execution or branch prediction) designed for vector computing, not general purpose

Opportunity and Challenge

- PDE applications are of major importance in both academia and industry
- new HPC hardware (GPUs, AVX, etc.) offers 10$\times$ improvement in performance of affordable HPC but greatly increased programming complexity
- want a suitable level of abstraction to separate the user's specification of the application from the details of the parallel implementation
- aim to achieve code longevity and near-optimal performance through re-targetting the back-end to different hardware

Context

Part of a larger project led by Paul Kelly at Imperial College



History

OPlus (Oxford Parallel Library for Unstructured Solvers)

- developed for Rolls-Royce 10 years ago
- MPI-based library for HYDRA CFD code on clusters with up to 200 nodes

OP2

- open source project
- keeps OPlus abstraction, but slightly modifies API
- an “active library” approach with code transformation generates CUDA, OpenCL and OpenMP/AVX code for GPUs and CPUs

OP2 Abstraction

- sets (e.g. nodes, edges, faces)
- datasets (e.g. flow variables)
- pointers (e.g. from edges to nodes)
- parallel loops
  - operate over all members of one set
  - datasets have at most one level of indirection
  - user specifies how data is used
    (e.g. read-only, write-only, increment)
  - set elements can be processed in any order,
    doesn't affect result to machine precision
    - explicit time-marching, or multigrid with an explicit smoother is OK
    - Gauss-Seidel or ILU preconditioning is not

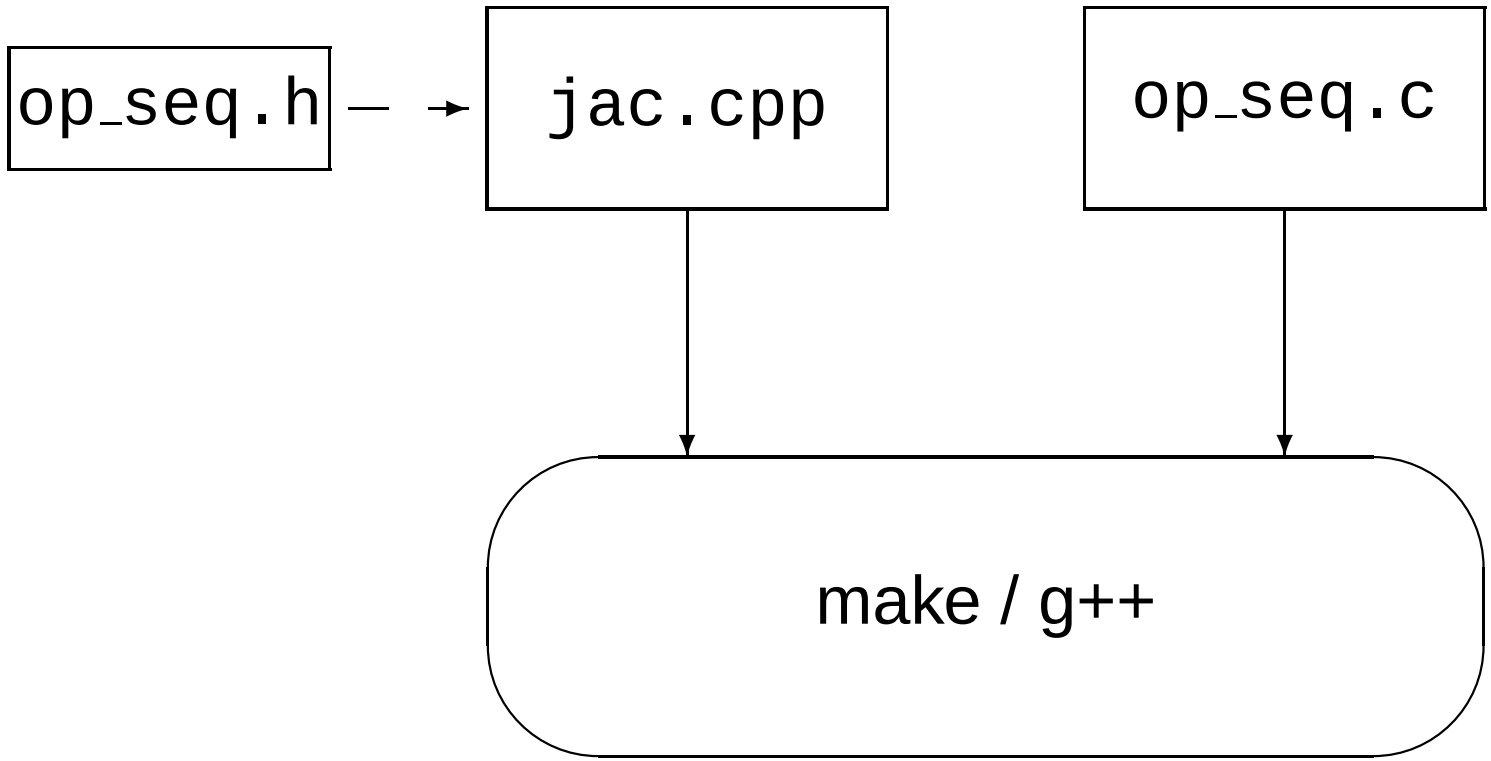
User build processes

Using the same source code, the user can build different executables for different target platforms:

- sequential single-thread CPU execution
  - purely for program development and debugging
  - very poor performance
- CUDA / OpenCL for single GPU
- OpenMP/AVX for multicore CPU systems
- MPI plus any of the above for clusters

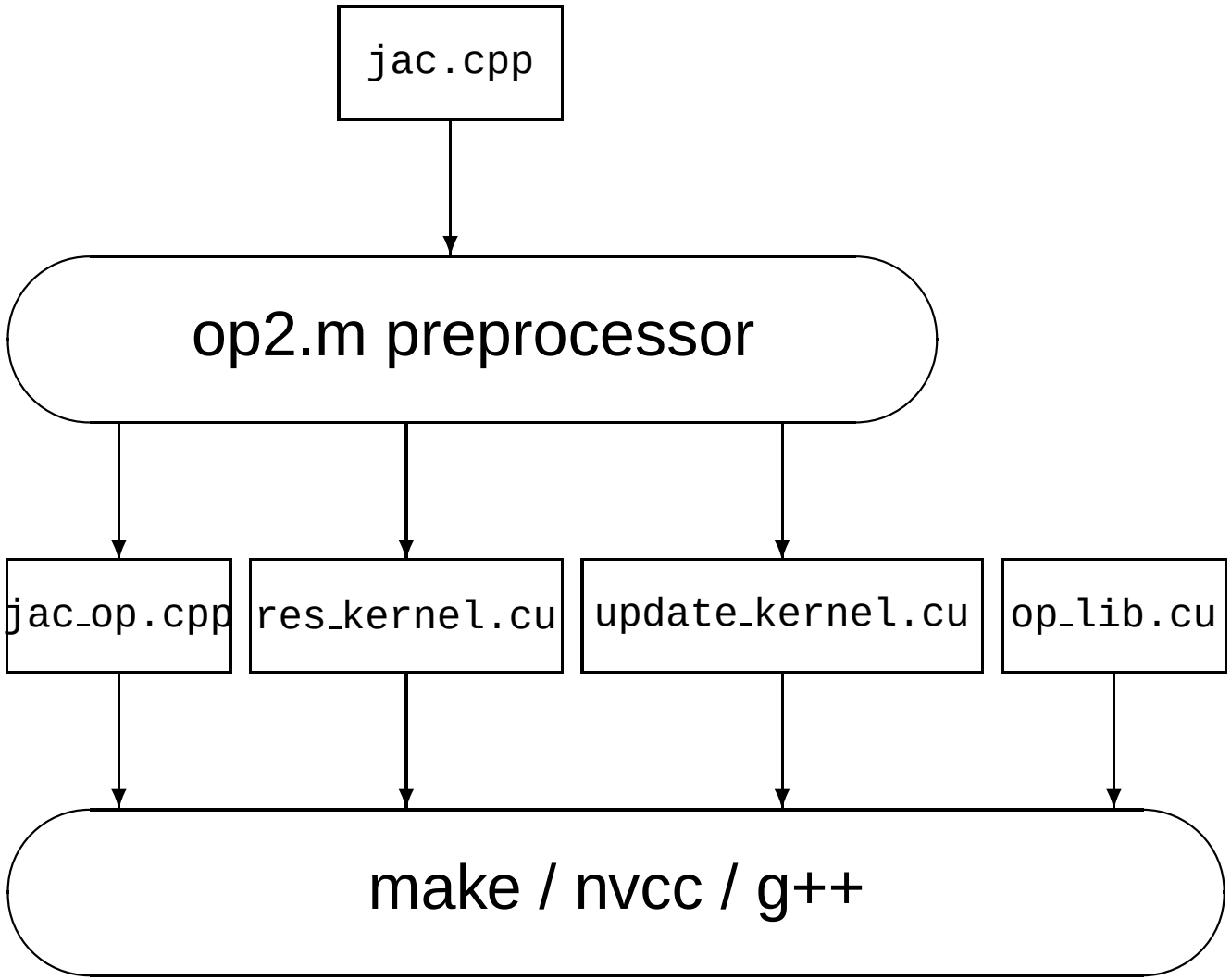
Sequential build process

Traditional build process, linking to a conventional library in which many of the routines do little but error-checking:



CUDA build process

Preprocessor parses user code and generates new code:



Current status

- SILOET sub-project funded by Rolls-Royce and TSB started in January
- EPSRC funded project starts in October
- already have CUDA prototype for single GPU, with preprocessor written in MATLAB
- plan to look at OpenCL and PGI FORTRAN CUDA
- waiting for new NVIDIA Fermi hardware to assess performance – expanded shared memory and L1/L2 caches will help a lot
- long-term goal: simulation of a complete aero-engine or an aircraft in landing configuration using 100 GPUs

Collaborators

- Paul Kelly, Graham Markall (Imperial College)
- Jamil Appa, Pierre Moinier (BAESystems)
- Leigh Lapworth, Yoon Ho (Rolls-Royce)
- Nick Hills (Surrey)
- Tobias Brandvik, Graham Pullan (Cambridge)
- Stephen Jarvis (Warwick)
- plus technical support from NVIDIA
  and talking to Microsoft about auto-tuning

Project webpage:
`www.maths.ox.ac.uk/~gilesm/op2/`