Compilation for Many-Core Parallel Architectures

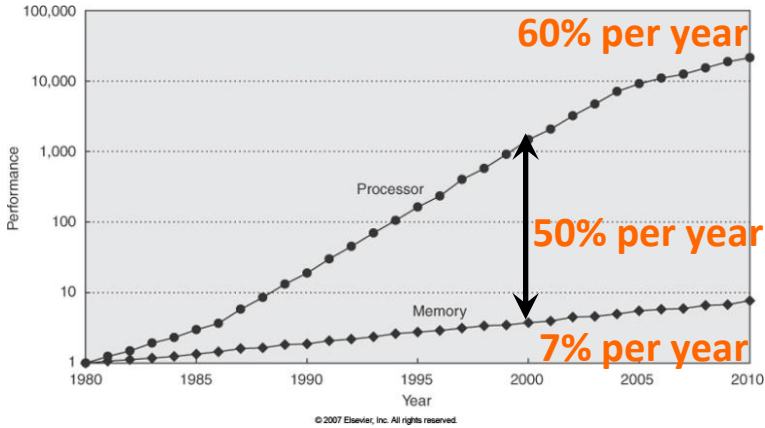
Radu Prodan, Thomas Fahringer, Philipp Gschwandtner, Klaus Kofler, Hans Moritsch, Simone Pellegrini, Heiko Studt, Peter Thoman, John Thomson

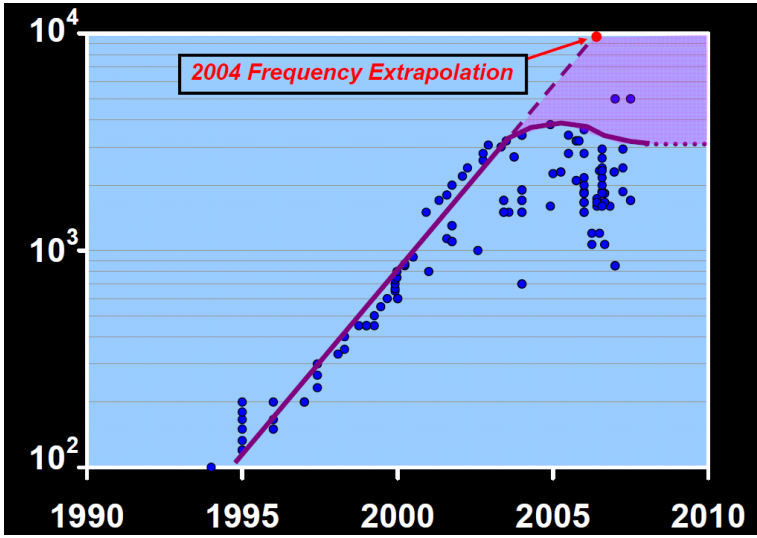
Distributed and Parallel Systems

Institute of Computer Science, University of Innsbruck

Multicore Parallel Processing

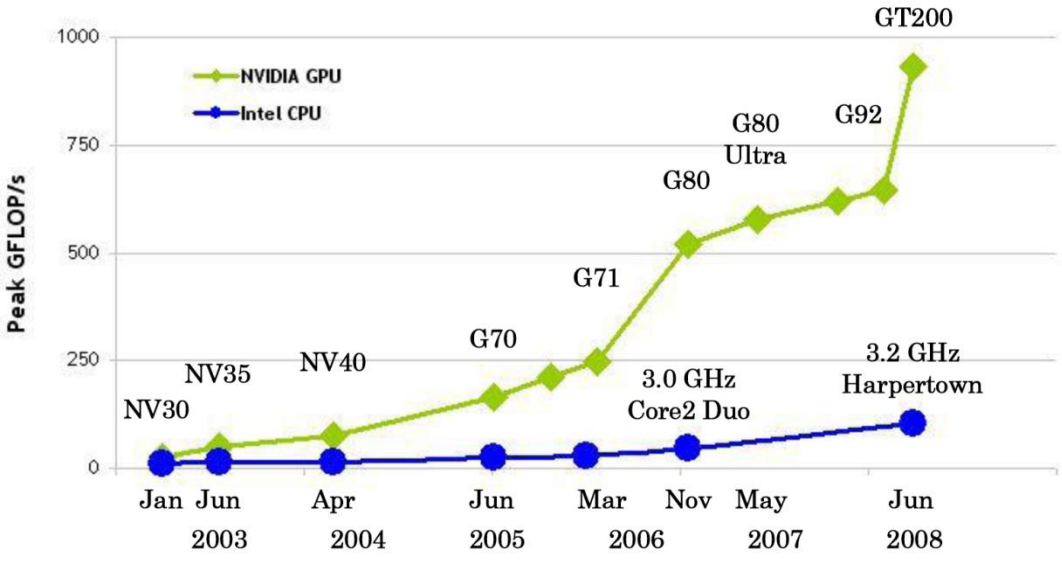
- Moore's law is alive and will stay alive
  - The number of transistors on a chip will double every 18 months
- 1980 – 2004: processors performance increased through frequency scaling
  - Applications benefit from new processors architectures transparently
- Frequency scaling drawbacks
  - Memory wall
  - Instruction level parallelism wall
  - Power wall (*Power* $\propto$ *Frequency*^{3})
- Multi-core processor architectures
  - Package multiple cores in the same integrated circuit
  - Operate the individual cores at lower frequency
  - Combine "commodity processors" with "stream accelerators"

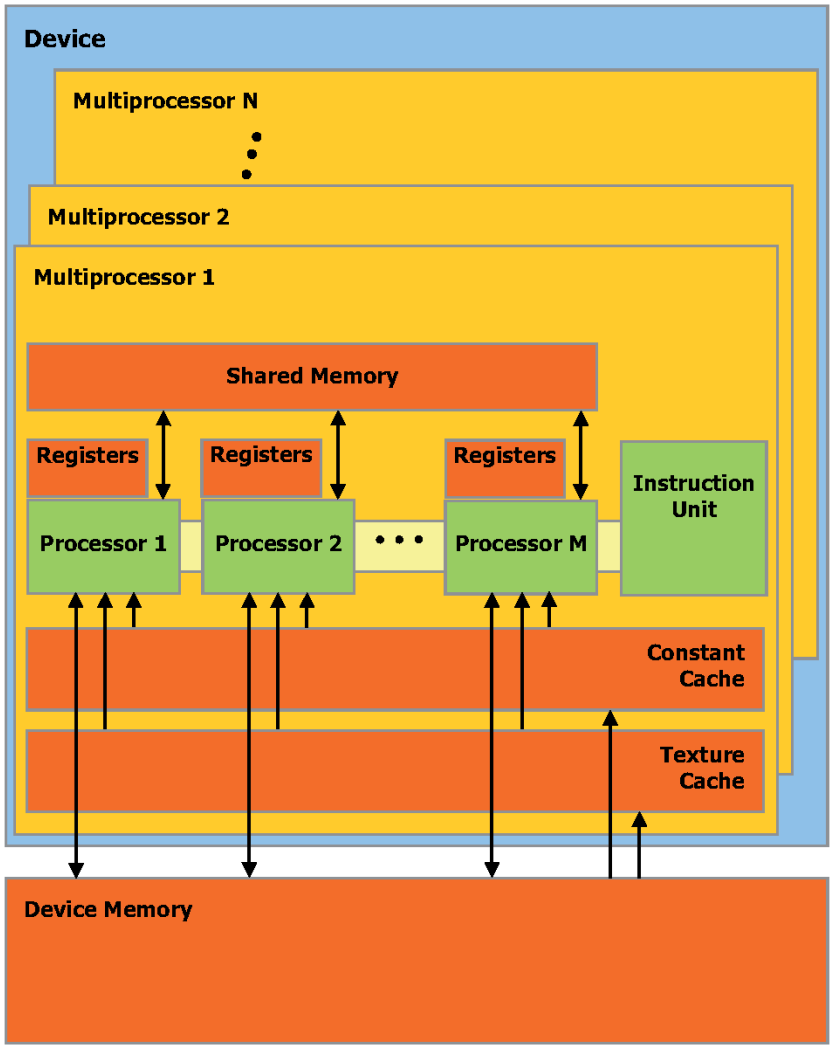




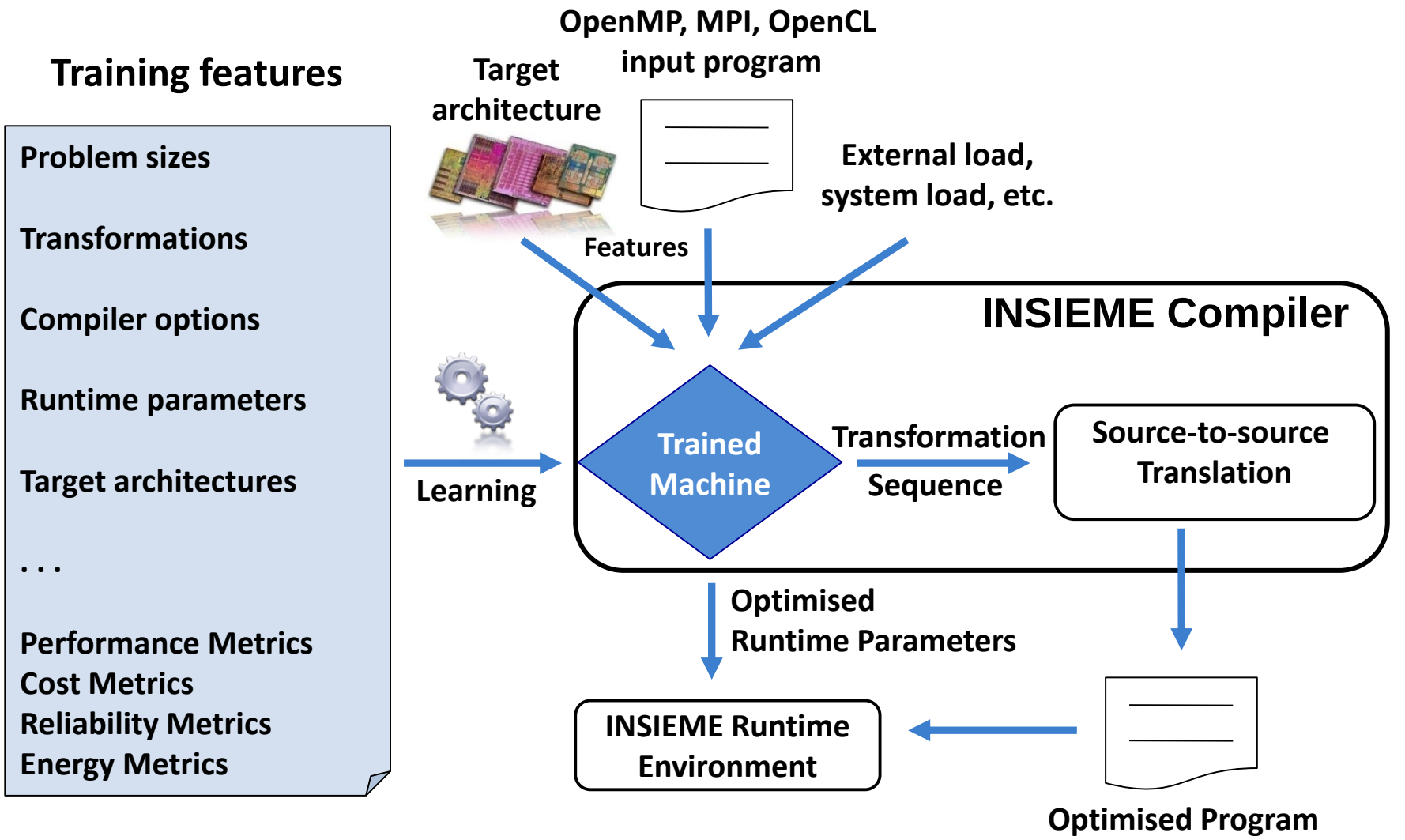
Graphics Accelerators

- Graphics Processing Units (GPU)
  - Nvidia, AMD/ATI, …





INSIEME Compilation Environment

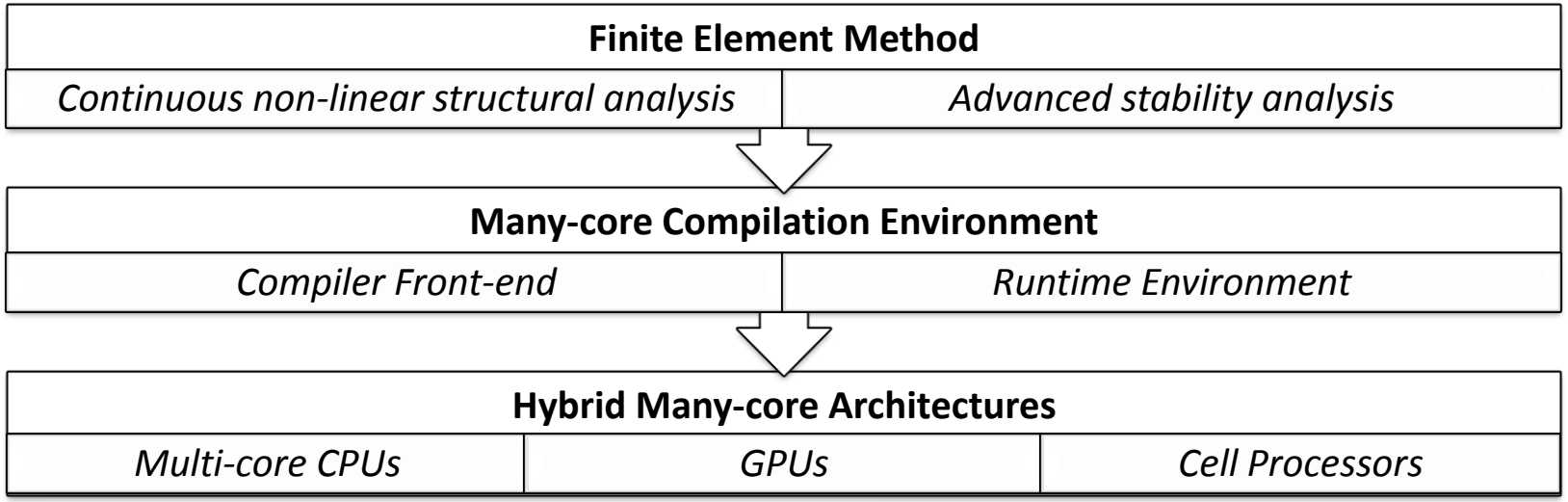


OpenCore

- A Many-core Compiler for Industrial Engineering Stability Analysis
- Funded by FFG
  - BRIDGE 1 basic program
- Total budget: € 220.000
- Partners
  - University of Innsbruck
  - INTALES GmbH
- Duration: 2 years
- Official start: 1.07.2010

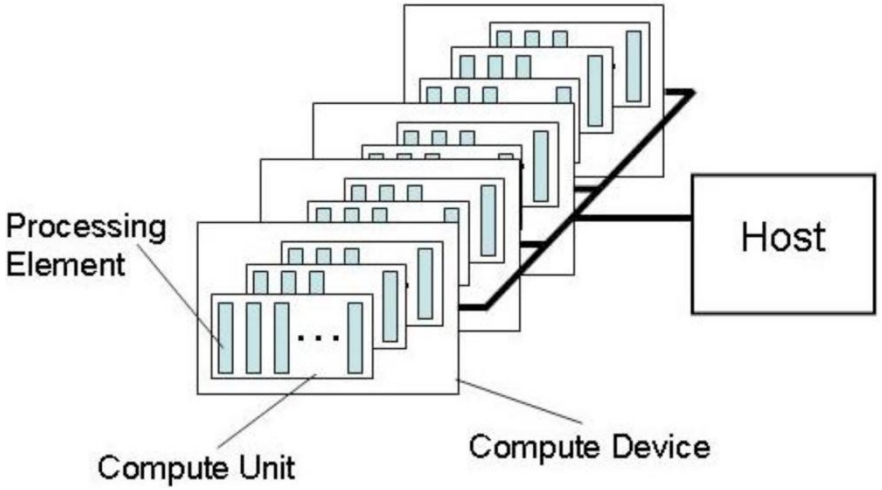
Project Goals

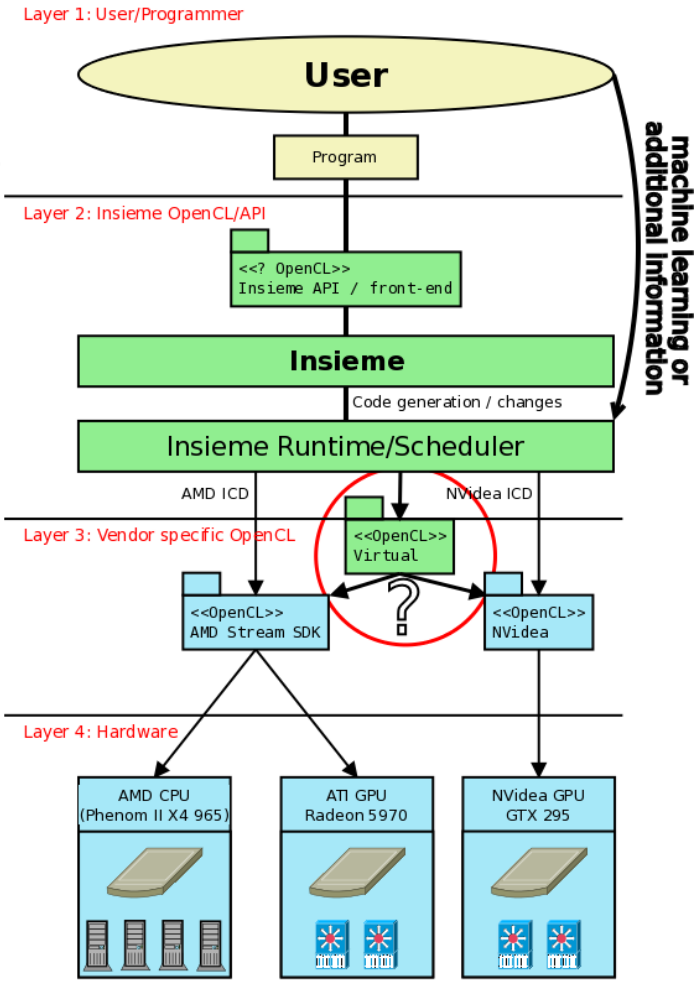
- INTALES
  - Parallel FEM-based stability and branching analysis algorithms
- University of Innsbruck
  - A single **compiler** for hybrid many-core processor architectures
  - **Portability** versus **performance**



Open Computing Language (OpenCL)

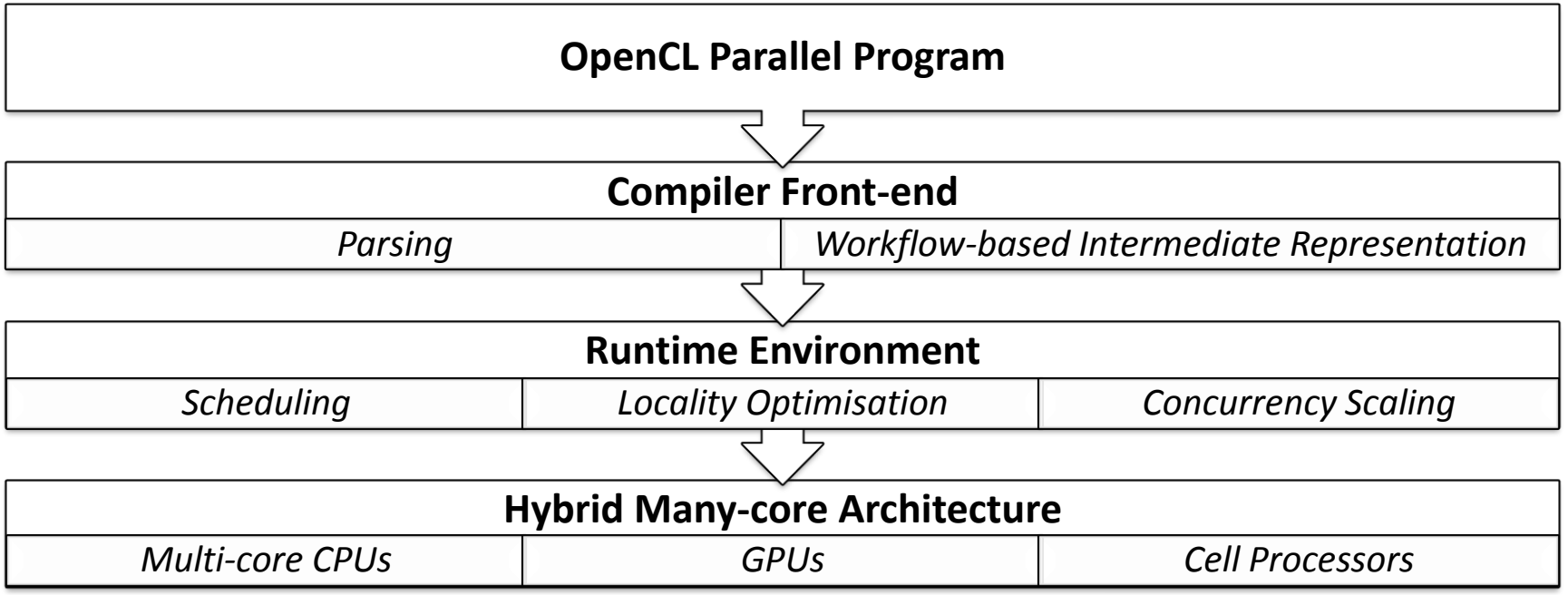
- Framework for programming CPUs, GPUs and DSPs
  - Khronos Group: AMD, Intel, Nvidia, Apple, …
  - Version 1 released on December 8, 2008
  - Industrial compiler support: AMD, NVidia, IBM, Apple
- **Task parallelism** specified in a C "host program"
  - Coordination of data parallel kernels (workflow)
- **Data parallelism** specified in OpenCL kernels
  - Submitted by the main program for execution on the SIMD compute device





OpenCore Compiler Research Goals

- Concurrent support for heterogeneous hardware from multiple vendors
- Automatic scheduling of kernels to cores: optimisation heuristics
- Locality optimisation: prefetching, dynamic multi-buffering
- Dynamic concurrency scaling
  - Adaptation of the amount of used cores



Questions?