HETEROGENEOUS **CPU+GPU** COMPUTING

Ana Lucia Varbanescu – University of Amsterdam

a.l.varbanescu@uva.nl

Significant contributions by: **Stijn Heldens** (U Twente), **Jie Shen** (NUDT, China), **Basilio Fraguela** (A Coruna University, ESP),

What have we covered?

- Preliminaries

- Part I: Introduction to CPU+GPU heterogeneous computing
  - Performance promise vs. challenges
- Part II: Programing models
- Part III: Workload partitioning models
  - Static vs. Dynamic partitioning
- Part IV: Static partitioning and Glinda

Slides available:

- Part V: Tools for (programming) heterogeneous systems
  - Low-level to high-level

- Take home message

Take ~~home~~ to the office message [1]

- Heterogeneous computing works!
  - *More resources.*
  - *Specialized resources.*
- Most efforts in static partitioning and run-time systems
  - Glinda = static partitioning for single-kernel, data parallel applications
    - Now works for multi-kernel applications, too
  - StarPU, OmpSS = run-time based dynamic partitioning for multi-kernel, complex DAG applications
- Domain-specific efforts, too
  - HyGraph - graph processing
  - Cashmere – divide-and-conquer, distributed

Take ~~home~~ to the office message [2]

- Choose a system based on your application scenario:
  - Single-kernel vs. multi-kernel
  - Massive parallel vs. Data-dependent
  - Single run vs. Multiple run
  - Programming model of choice
- There are models to cover combinations of these choices!
  - No framework to combine them all – food for thought?

Future research directions

- More heterogeneous platforms
- Extension to more application classes
  - Multi-kernel with complex DAGs
  - (streaming applications, graph-processing applications)
- Integration with distributed systems
  - Intra-node workload partitioning + inter-node workload scheduling
- Extension of the partitioning model
  - Energy consumption

Open questions ?

- Analytical modeling instead of profiling
  - Statistical modeling – ask me more!
- Extending to other type of workloads
  - Graph processing – ask me more!
- Performance portability
  - HPL/specialized OpenCL – ask me more!