Leveraging GPUs and Self-Tuning Systems on the Road to Exascale: a view from ARM

Anton Lokhmotov, Media Processing Division

Joint GPGPU-5 / EXADAPT-2 Round Table

3 March 2012, London

ARM

- A company licensing IP to all major semiconductor companies (form of R&D outsourcing)
  - Established in 1990 (spin-out of Acorn Computers)
  - Headquartered in Cambridge, UK, with 27 offices in 13 countries and 2000+ employees
- ARM is the most widely used 32-bit CPU architecture
  - Dates back to the mid 1980s (Acorn RISC Machine)
  - Dominant in mobile devices (e.g. on average 3 processors per phone)
- Mali is the most widely licensed GPU architecture
  - Dates back to the early 2000s (developed by Falanx, Norway)
  - Media Processing Division established in 2006 (acquisition of Falanx)
  - Released products:
    - Mali-55 (OpenGL ES 1.1), Mali-200, Mali-400 (OpenGL ES 2.0)
    - Mali-T604 (OpenGL ES 2.0 + OpenCL 1.1)

Landscape of accelerator programming

Interface	CUDA	OpenCL	DirectCompute	RenderScript
Originator	NVIDIA	Khronos (Apple)	Microsoft	Google
Year	2007	2008	2009	2011
Area	HPC, desktop	Desktop, mobile, embedded, HPC	Desktop	Mobile
OS	Windows, Linux, Mac OS	Windows, Linux, Mac OS (10.6+)	Windows (Vista+)	Android (3.0+)
Devices	GPUs (NVIDIA)	CPUs, GPUs, custom	GPUs (NVIDIA, AMD)	CPUs, GPUs, DSPs
Work unit	Kernel	Kernel	Compute shader	Compute script
Language	CUDA C/C++	OpenCL C	HLSL	Script C
Distributed	Source, PTX	Source	Source, bytecode	LLVM bitcode

- *Portability* is likely to remain an issue despite standardisation efforts
- *Performance portability* is perhaps even more of an issue!

Performance tuning

- Why should we tune?
  - To improve selected performance metrics (e.g. energy efficiency)
- What should we tune for?
  - Architecture & implementation
  - System behaviour (resource contention, failures)
  - Data (probability distribution of characteristic inputs)
- When should we tune?
  - Compile-time, run-time, install-time, idle-time…
- How should we tune?
  - By automatically synthesising software based on extracted features

Challenges

- Understanding and exploring the search space (cf. EXADAPT)
  - Extracting and interpreting features
  - Using collective knowledge
- Enabling synthesis of efficient software (cf. GPGPU)
  - Compilers have always had a limited ability to invent new algorithms
  - Algorithm representation must be clear and flexible enough to enable powerful transformations
  - What you generate may change over time (e.g. system and driver improvements)
- Marrying the above (cf. GPGPU+EXADAPT)

Final note

- Building *exascale* supercomputers is probably less important than making *petascale* embedded and mobile devices work *together*!