General Purpose GPU Programming

Advanced Operating Systems
Tutorial 7
Tutorial Outline

• Review of lectured material
• Key points
• Discussion
  • OpenCL
  • Future directions
Review of Lectured Material

• Heterogeneous instruction set systems
• Heterogeneous multi-kernel systems – Helios
• Main core with heterogeneous offload
  • Graphics offload hardware – GPGPU
  • Programming model
  • OpenCL
  • Integration with operating systems

• Heterogenous virtual machines – Hera JVM

• Hybrid models – Accelerator
  • Lazy encoding of SIMD-style operations and JIT compilation into type system
Key Points

- Increasing heterogeneity of hardware
- Programming models are complex
  - Too limited to run a full operating system
  - Too different to effectively run standard programming languages
- OpenCL-style offload model performs well, but is complex to program
- Attempts to hide complexity in VM have had mixed success
Discussion

• What is complexity versus performance trade-off in OpenCL – how does this limit usefulness?

• How can SIMD-style processing be more cleanly incorporated into modern languages?

• Is the embedded DSL approach of Accelerator a set in the right direction, or is the complexity of the VM excessive?

• How to use heterogenous processing resources?
Course Wrap-up

- Real-time scheduling of periodic tasks
- Real-time scheduling of aperiodic & sporadic tasks
- Resource management
- Real-time and embedded systems programming
- Garbage collection
- Message passing
- Transactions
- General purpose GPU programming
Key Points

- Real-time systems – predictability and reliability are critical; desire to raise level of abstraction to help to achieve these goals
- Garbage collection is effective, but at high memory overhead cost – real-time garbage collection exists
- Message passing effective for multi-core systems; potential of multi-kernel operating systems model
- Transactions seem to have limited applicability
- No effective GPGPU programming model; OpenCL is too low-level and not a long-term solution
Discussion

• Wide spectrum of research ideas and concepts

• Which are seeing widespread use?
  • Functional languages and message passing concurrency
  • Garbage collection – potential for integration with kernels
  • Increased use of static code analysis tools, to debug the limitations of C

• Opportunities for dependable kernels
  • New implementation frameworks and safe programming languages
  • Approaches similar to Singularity have large potential
Examination

• Final examination:
  • Worth 80% of marks for the course
  • 2 hours; answer 3-out-of-4 questions
  • Sample exam and past papers available on Moodle, and on the website

• All material covered in the lectures, tutorials, and papers is examinable
  • Aim is to test your understanding of the material, not simply to test your memory of all the details – in particular, read papers to understand the concepts, not details
  • Explain why, don’t just recite what – are looking for your reasoned and justified technical opinion about the material
The End

http://csperkins.org/teaching/adv-os/