MPhil in Advanced Computer Science
Multicore Programming

- Lecturers: Dr Peter Sewell, Dr Tim Harris, and Dr Jaroslav Ševčík
- 8 two-hour blocks (including two one-hour practical sessions)

Aims

In recent years multiprocessors have become ubiquitous, but building reliable concurrent systems with good performance remains very challenging. The aim of this module is to introduce some of the theory and the practice of concurrent programming, from hardware memory models and the design of high-level programming languages to the correctness and performance properties of concurrent algorithms.

Syllabus

Part 1: Introduction and relaxed-memory concurrency (Peter Sewell)

- (1 block) Introduction. Sequential consistency, atomicity, basic concurrent problems.
- (2 blocks) Concurrency on real multiprocessors: the relaxed memory model(s) for x86, and theoretical tools for reasoning about x86-TSO programs. Other hardware models for contrast: Power/ARM/Itanium.
- (1 block) High-level languages. An introduction to Java and C++0x shared-memory concurrency, including the basic Java concurrency primitives and some discussion of Java memory models and compiler optimisations.

Part 2: Concurrent algorithms (Tim Harris)

- (3 blocks, including two one-hour practical sessions) Concurrent programming. Simple algorithms (readers/writers, stacks, queues) and correctness criteria (linearisability and progress properties). Advanced synchronisation patterns (e.g. some of the following: optimistic and lazy list algorithms, hash tables, double-checked locking, RCU, hazard pointers), with discussion of performance and on the interaction between algorithm design and the underlying relaxed memory models.
- (1 block) research topics, likely to include one hour on transactional memory and one guest lecture.

Assessment

The assessment will be based on assessed coursework exercises (75%) and practical reports (25%).

Prerequisites

Some familiarity with discrete mathematics (sets, partial orders, etc.) and with sequential Java programming will be assumed. Experience with operational semantics and with some concurrent programming would be helpful.

References

- *The Art of Multiprocessor Programming*, Herlihy and Shavit.