Virginia Tech ❖ Bradley Department of Electrical and Computer Engineering

ECE 5984 Advanced Transactional Programming
ECE 4984 Transactional Programming
Off campus online access through WebEx

Course Syllabus Fall 2017

Instructor
Dr. Sandeep Hans
Postdoctoral Research Associate
ECE Dept., Virginia Tech
Office: 454 Durham Hall, Blacksburg, VA 24061
Website: http://www.ssrg.ece.vt.edu/~hans/
Phone: 540-449-5663, Email: sandeep.hans@vt.edu

Meeting Times and Location
Mondays and Wednesdays 12:30pm to 1:45pm
Location: TBA

Course Website
http://www.ssrg.ece.vt.edu/tm-prog

Course Objectives
The computer industry is undergoing a paradigm shift as computer hardware has reached a limit on the single-threaded CPU performance. To circumvent this limit, chip vendors are now designing and releasing a new generation of multi-processor chips called multicores. In developing application software for such hardware, one of the central challenges is how to write concurrent code – increasingly necessary to exploit the parallelism of multicore hardware. The core programming difficulty here is reasoning about concurrency, in particular, understanding how concurrent computations on separate processors coordinate with one another to ensure consistency of the shared state, while achieving high performance with high programmability. Software engineers often face these challenges while coding complex applications.

A breakthrough recent result in concurrent programming is the Transactional Memory (TM) abstraction. TM enables a programmer to delineate a code block that accesses shared memory data as “atomic”. The abstraction ensures that the block executes atomically, with additional properties such as memory consistency and execution isolation. The infrastructure that implements the abstraction, either in hardware or software, ensures these properties using a concurrency control algorithm. TM has high programmability, yields high performance, and enables code composability.

The course teaches Transactional Memory programming and the transition from the classical lock-based programming model to the new programming abstraction where transactions are used. Transactional memory building blocks, including those for versioning, conflict detection, and contention management will be illustrated and developed. Software and hardware transactional memory algorithms including TL2, NORec, and Intel TSX cache-coherence extensions will be taught, along with their implementations. Techniques for transactional memory programming including standard APIs, code instrumentation, and correctness guarantees will be discussed in class. Case studies on transforming production applications coded using locks into transactional applications will be analyzed with students.

At the end of the course, students will be able to:
- Design and implement concurrent applications using fundamental lock-based synchronization patterns.
- Describe algorithms for software transactional memory and hardware transactional memory.
- Design transactional memory components and construct a transactional memory library or subsystem.
- Replace locks in legacy concurrent applications with transactional memory.
- Write concurrent applications using transactional memory, exploiting best programming practices.

**Prerequisites**
Graduate standing. The course requires a previous introduction to multithreaded programming and concurrency using mutual-exclusion algorithms and lock-based synchronization primitives.

**Required and Recommended Texts**

**Required:**

**Recommended:**

**Grading (tentative):**

<table>
<thead>
<tr>
<th>Type of graded work</th>
<th>4984</th>
<th></th>
<th>5984</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Final grade %</td>
<td>Number</td>
<td>Final grade %</td>
</tr>
<tr>
<td>Assignments (Questions and Programming)</td>
<td>5</td>
<td>50% (10% each)</td>
<td>5</td>
<td>50% (10% each)</td>
</tr>
<tr>
<td>Quiz</td>
<td>1</td>
<td>10%</td>
<td>1</td>
<td>10%</td>
</tr>
<tr>
<td>Programming Project</td>
<td>1</td>
<td>20%</td>
<td>1</td>
<td>30%</td>
</tr>
<tr>
<td>Final exam</td>
<td>1</td>
<td>20%</td>
<td>1</td>
<td>10%</td>
</tr>
</tbody>
</table>

**Details:**
The course is planned to have 5 assignments, where each consists of question(s) and programming work. Each of these assignments counts as 10% of the final grade. One take-home quiz is planned to be distributed to students, and it counts as 10% of the final grade. A programming project is required, and it weights 20% of the final grade for 4984, and 30% of the final grade for 5984. The final exam contributes to 20% of the final grade for 4984, and 10% of the final grade for 5984.
<table>
<thead>
<tr>
<th>Topic</th>
<th>Req. Book Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to concurrency and synchronization</td>
<td>Chapter 1</td>
</tr>
<tr>
<td>Concurrent programming abstractions: lock-based, lock-free, and optimistic synchronization</td>
<td>Chapter 1 - 2</td>
</tr>
<tr>
<td>Transactional memory building blocks (e.g., versioning, conflict detection, contention management)</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>Designing concurrent applications using transactions: APIs, instrumentation, and correctness issues</td>
<td>Literature Paper</td>
</tr>
<tr>
<td>Software transactional memory: algorithms and mechanisms (e.g., TL2, NORed, RingSTM)</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>Hardware transactional memory: algorithms and mechanisms (e.g., cache-coherence-based, software fallback)</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>Case study on converting a lock-based application into transactional</td>
<td>Literature Paper</td>
</tr>
</tbody>
</table>