CS4021/4521 Advanced Computer Architecture II

Dr Jeremy Jones

Office: O’Reilly Institute F.11

Email: jones@scss.tcd.ie
# BACS / MCS Year 4 Timetable

<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.00 – 12.00</td>
<td>MT: CS4400: Lect LB04</td>
<td>MT: CS4032: Lect LB04 / ICT Lab1/2</td>
<td>MT: EE4C16: Lab AP2.28</td>
<td>MT: CS4001: Lect LB08</td>
<td>MT: CS4051: Lect LB08</td>
</tr>
<tr>
<td>13.00 – 14.00</td>
<td>MT: CS4053: Lect. LB04/ICTLab 1</td>
<td>MT: CS4053: Lect LB01/ICTLab 1</td>
<td>MT: CS4001: Lect Synge</td>
<td>MT: CS4001: Lect LB04</td>
<td>MT: CS4021: Lect LB08</td>
</tr>
<tr>
<td>17.00 – 18.00</td>
<td>MT: CS4021: Lect Salmon</td>
<td>MT: CS4004: Salmon</td>
<td>MT: CS4081: Lect LB01</td>
<td>MT: CS4004:Lect LB01</td>
<td>MT: CS4021: Lect LB08</td>
</tr>
</tbody>
</table>
## MAI Year 5 Timetable

<table>
<thead>
<tr>
<th>DAY</th>
<th>SEMESTER</th>
<th>0900 - 1000</th>
<th>1000 - 1100</th>
<th>1100 - 1200</th>
<th>1200 - 1300</th>
<th>1300 - 1400</th>
<th>1400 - 1500</th>
<th>1500 - 1600</th>
<th>1600 - 1700</th>
<th>1700 - 1800</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MONDAY</strong></td>
<td>First semester</td>
<td>CS7DS3 [LB1.1C/LCLAB1]</td>
<td>CS7NS1 [LB04]</td>
<td>CS7CS3 [LTEE2]</td>
<td>CS7CS3 [LB1.07]</td>
<td>CS7GSV3 [LB1.07]</td>
<td>CS7NS2 [LB1.07]</td>
<td>5C3 [MD0]</td>
<td>5C3 [MD2]</td>
<td>CS4521 [HAM1]</td>
</tr>
<tr>
<td><strong>TUESDAY</strong></td>
<td>First semester</td>
<td>CS4501 [LB01]</td>
<td>CS7DS1 [LB1.07/LG12]</td>
<td>5B9 [CLT]</td>
<td>CS7DS1 laboratory [LB01/LG12]</td>
<td>5C3 [MD2]</td>
<td>CS77S1 [LB1.07]</td>
<td>CS4504 [HAM1]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
</tr>
<tr>
<td></td>
<td>Second semester</td>
<td>CS77S6 [LB1.07]</td>
<td>CS77S5 [LB01]</td>
<td>5C1 laboratory session [CADLAB]</td>
<td>5C7 [MD2]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
</tr>
<tr>
<td><strong>WEDNESDAY</strong></td>
<td>First semester</td>
<td>5B9 [CLT]</td>
<td>5C4 [AP2.03]</td>
<td>CS77S2 [HAM1]</td>
<td>5E2 [PHY311]</td>
<td>5E2 [HAM3]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
</tr>
<tr>
<td></td>
<td>Second semester</td>
<td>5C7 [AP2.04]</td>
<td>5C1 [CADLAB]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
<td>CS77S1 [LB01]</td>
</tr>
<tr>
<td><strong>THURSDAY</strong></td>
<td>First semester</td>
<td>5C4 [AP2.03/CAD/LAB]</td>
<td>CS4501 [LB01]</td>
<td>5C3 [MD0]</td>
<td>CS4501 [HAM2]</td>
<td>CS77S4 [LB08]</td>
<td>CS77S4 [LB08]</td>
<td>CS77S3 [LB08]</td>
<td>CS77S3 [LB08]</td>
<td>CS77S3 [LB08]</td>
</tr>
</tbody>
</table>
CONCURRENT PROGRAMMING WITH AND WITHOUT LOCKS

• Algorithms are the poetry of the 21st Century!
• Peterson and Bakery locks [locks without atomic instructions]
• Spin model checker [revision?]
• atomic instructions
• cache coherency [revision]
• memory ordering
• serialising instructions
• cost of sharing data
CONCURRENT PROGRAMMING WITH AND WITHOUT LOCKS

- lock implementations and performance [TAS, TATAS, MCS, ticket, ...]
- lockless data structures and algorithms
  - CAS based
  - LIFOs, FIFOs, linked, lists, trees, hash tables, ...
  - memory management [eg. hazard pointers]
- hardware transactional memory [HTM]
  - Herlihy and Moss [1993]
  - first implementation Intel Haswell CPU [2012]
  - hardware lock elision (HLE)
  - restricted transactional memory (RTM)
USEFUL BOOKS

- *The Art of Multiprocessor Programming*
  Maurice Herlihy and Nir Shavit

- *The Spin Model Checker: Primer and Reference Manual*
  Gerald J. Holzmann

- *Principles of the Spin Model Checker*
  Mordechai Ben-Ari

  - lecture notes
  - tutorials and coursework
  - miscellaneous materials (eg. papers, documentation, sample code, ...)

CS4021/4521 © 2017 jones@scss.tcd.ie School of Computer Science and Statistics, Trinity College Dublin 5-Oct-17
ASSESSMENT [5 ECTS]

Coursework: 20%

• 3 or 4 coursework projects

Examination: 80%

• January 2018
• answer 3 out of 4 questions in 2 hours
MALBEC [malbec.scss.tcd.ie]

Supermicro 1U SuperServer 5018D-FNFT
Intel Xeon D-1540 2.0 GHz Broadwell CPU 45W
8 cores / 16 threads
128GB ECC RDIMM

Transactional Synchronization Extensions (TSX)
Haswell TSX implementation had a bug, Broadwell OK

Linux (Debian)

use for coursework

remote access via macneill or VPN

HOPEFULLY CAN NOW USE NEW MSc PCs IN LABS
Why lockless algorithms?

- Clock rate of a single CPU core currently limited to ≈ 4GHz
- Single CPU core processing power no longer doubling every 18 months
- Intel, AMD, Sun, IBM, ... producing multicore CPUs instead
- Typical desktop has 4 cores with each core capable of executing 2 threads [hyper-threading] giving a total of 8 concurrent threads
- Top-of-range desktop 2014 16 threads, 2016 32 threads, ... [Moore's Law and Joy's Law]
- Need to be able to exploit cheap threads on multicore CPUs
- Locked based solutions are simply not scalable as locks inhibit parallelism
- Need to explore lockless data structures and algorithms
Consider a Binary Search Tree (BST) as an example

- contains(key)
  - returns 1 if key in tree

- add(key)
  - always adds to a leaf node

- remove(key)
  - 3 cases depending if node has zero, one or two children

- operations on tree normally protected by a per tree lock which inhibits parallelism

- why can't operations be performed in parallel?

- how much parallelism is possible?
BST Operations

- add (50) [single pointer updated]
- add(45) [single pointer updated]
- remove(45) – NO children [one pointer updated]
- remove(25) – ONE child [single pointer updated]
- remove(20) – TWO children

  find node (20)
  find smallest key in its right sub tree (22)
  overwrite key 20 with 22
  remove old node 22 (will have zero or one child)
  [key and a pointer updated]

- variations

  find largest key in left sub-tree instead of smallest key in right sub tree
  move node instead of value
Concurrent add operations

- concurrently add(27) and add(50)
  OK if adding to different nodes

- concurrently add(23) and add(24)
  problem as adding to same leaf node
  result depends on how steps of operations are interleaved [pointer updates]
  could work correctly, BUT...
  if there is a conflict ONLY one node may be added [23 or 24, BUT still a valid tree]
Concurrent remove operations

- concurrently remove(21) and remove(27)
  
  OK as both are leaf nodes [have NO children]

- concurrently remove(20) and remove(22)
  
  smallest key in 20's right sub tree is 22
  result depends on how steps of operations are interleaved [key and pointer updates]
  could work correctly, BUT ...
  one possible interleave is as follows
  
  both operations find 22
  20 is overwritten with 22
  old node 22 removed [by both operations], BUT
  22 still in tree!

other interleaves possible
Concurrent add and remove operations

- concurrently add(50) and remove(25)
- OK as modifying links in different nodes
Concurrent add and remove operations...

- concurrently add(50) and remove(40)
- result depends on how steps of operations are interleaved [key and pointer updates]
- could work correctly, BUT ...
- one possible interleave as follows
- 40 deleted, BUT ...
- 50 also deleted as attached to 40
Concurrent Operations on a BST

• concurrent operations ARE possible

• probability of a conflict inversely proportional to size of tree

• conflicts proportional to number of concurrent operations

• with a large tree, conflicts between operations will be rare

• with a large tree, should be able to achieve a linear speedup proportional to number of threads provided that conflicts can be detected and resolved

• protecting tree with a single lock is pessimistic as it assumes conflicts will occur resulting in NO parallelism

• a lockless algorithm is optimistic as it assumes conflicts unlikely to occur and, when they are detected, they are resolved – allows parallelism while there are no conflicts [which hopefully is most of the time]