Lecture 9

More on synchronization

Application Study:
Protein sequence querying
Announcements
More on Synchronization
Synchronization in Java

• What’s the simplest synchronization mechanism in Java?
• **Semaphores** come in handy for a producer consumer application
• Revisit CAS (Compare and Swap)
Semaphores

• Locks handle mutual exclusion
• But they aren’t an appropriate solution for long critical sections
• Semaphores
  ▶ Counting
  ▶ Binary (mutex)
What is a semaphore?

• An abstract data type that provides mutual exclusion to critical sections [Dijkstra, 1968]

• Semaphore $S$: a special integer variable that supports two operations
  - acquire(): wait until $S > 0$, then decrement [Dijkstra’s $P()$ or down()]
  - release(): increment, allow another thread to enter [Dijkstra’s $V()$, or up()]
Two types of semaphores

• **Binary** semaphore (or **mutex** semaphore)
• **Counting** semaphore
  ▶ Initialized with general integer values
  ▶ Keeps track of multiple resources, also manages certain kinds of unsynchronized concurrent access (e.g., reading)
• Semaphores have no notion of ownership
Producer Consumer Example

• Located in $pub/examples/MT/PC
• Used in A1
import java.util.concurrent.Semaphore;

public class Producer extends Thread{
    public void run() {
        int N = PC.N;
        for (int r = 0; r<N; r++){
            int offs = r*N + 1;
            for (int i = 0; i < N; ++i)
                PC.x[i] = i + offs;
            PC.produced.release();
            try {
                PC.produced.acquire();
            } catch (InterruptedException e) {} 
        }
    }
}

public class PC{
    static int N;  static int[] x;
    static Semaphore produced;
    public static void main (String[] args) throws Exception {
        produced = new Semaphore(1,true);
        produced.acquire();
        Producer producer = new Producer();
        producer.start();
        int numMatched = Consumer();
    }
}
Incorrect results

import java.util.concurrent.Semaphore;
public class Producer extends Thread{
    public void run() {
        int N = PC.N;
        for (int r = 0; r<N; r++){
            int offs = r*N + 1;
            for (int i = 0; i < N; ++i)
                PC.x[i] = i + offs;
            PC.produced.release();
            try {
                PC.produced.acquire();
            } catch (InterruptedException e) {}
        }
    }
}

private static int Consumer(){
    int numMatched = 0;
    for (int r = 0; r<N; r++){
        try {
            produced.acquire();
        } catch (InterruptedException e) {}
        for (int i = 0, sum=0; i < N; ++i)
            sum += x[i];
        produced.release();
        int sumE = (N*(N+1)/2) + r*N*N;
        if (sumE == sum)
            numMatched++;
        else
            SUM DIDN'T MATCH
    }
    return(numMatched);
}

©2010 Scott B. Baden /CSE 160/ Winter 2010
Drawbacks

• Doesn’t provide...
  ▶ A connection between the semaphore and the data it controls
  ▶ control or guarantee of proper usage

• Alternatives
  ▶ Monitors
  ▶ PL Support

• java.util.concurrent provides higher level abstractions, like BlockingQueue

• How would a queue affect our producer consumer example?
Compare and set

• Atomically set value to \textit{updated} value if current value = expected value
• Returns \textit{false} if actual value \(\neq\) expected value, else true
• Let \textit{value} = 0
  ▶ \textit{value.compareAndSet(0, 100)}: value \(\leftarrow\) 100
  ▶ \textit{value.compareAndSet(10, 100)}: no change

\textbf{boolean compareAndSet(int expect, int update)}
public class AtomicCounter{
    private int value;
    public int getValue() { return value; }
    public int increment() {
        int oldVal = value.getValue();
        while (!value.CASw(oldVal, oldVal + 1)) {
            oldVal = value.getValue();
        }
        return oldVal + 1;
    }
}

value ← updated if current = expected
Sequence Querying
Initial sequencing and comparative analysis of the mouse genome
*Nature* 420, 520-562 (5 December 2002)

**Mouse Genome Sequencing Consortium**

The sequence of the mouse genome is a key informational tool for understanding the contents of the human genome and a key experimental tool for biomedical research. Here, we report the results of an international collaboration to produce a high-quality draft sequence of the mouse genome. We also present an initial comparative analysis of the mouse and human genomes, describing some of the insights that can be gleaned from the two sequences....
Why is this interesting?

• Comparative genomics allows one to read evolution's laboratory notebook. In the roughly 75 million years since the divergence of the human and mouse lineages, the process of evolution has altered their genome sequences and caused them to diverge …

• At the nucleotide level, approximately 40% of the human genome can be aligned to the mouse genome. These sequences seem to represent most of the orthologous sequences that remain in both lineages from the common ancestor, with the rest likely to have been deleted in one or both genomes.

• Roughly 3 billion nucleotides
Background

• We’ll look at proteins, sequences of (20) amino acids

• Alphabet of amino acids: single letter symbols

• Goal: form an alignment; compare two sequences to gauge their similarity

• Score: $\alpha = +2$ for match $\beta = -1$ for mismatch

<table>
<thead>
<tr>
<th>Query</th>
<th>X A R K M I R K C W D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>F F A R K Q M I K B W L X</td>
</tr>
</tbody>
</table>

©2010 Scott B. Baden /CSE 160/ Winter 2010
Local alignments and gaps

- We may be able to increase the score by creating *gaps*: $\gamma = -1$
- Score $= +9$
Best alignment

• What is the score?
• Another example:
  www.med.nyu.edu/rcr/rcr/course/sim-sw.html
Smith Waterman algorithm

• Computes the best possible alignment
• For symbols $a_i$ and $b_i$, $S[i,j] = \text{best possible alignment score when } a_i \text{ is aligned with } b_i$
• Suppose we have computed the alignment up to an including $a_i$ and $b_i$
• We have $S[0,0]$ through $S[i-1,j-1], S[i,j-1], S[i-1,j]$
• We compute $S[i,j]$ using a simple formula
The next alignment score

- $a_i$ aligns with $b_i$
  
  \[ S[i,j] = S[i-1,j-1] + \alpha \text{ if match, else } \beta = - \]

- Introduce a gap in $B$

  \[ S[i,j] = S[i-1,j] + \gamma \]

- Introduce a gap in $A$

  \[ S[i,j] = S[i,j-1] + \gamma \]

- Neither $a$ nor $b$ in the alignment: $S[i,j] = 0$

- We take the maximum value

\[ S_{i,j} = \text{Max}(S_{i,j} + f(a_i,b_i), S_{i-1,j} - \gamma, S_{i,j-1} - \gamma) \]

\[ a_i = b_i \rightarrow f(a_i,b_i) = \alpha \]

\[ a_i \neq b_i \rightarrow f(a_i,b_i) = \beta \]
Dynamic programming solution and the Traceback

<table>
<thead>
<tr>
<th>Subject Sequence</th>
<th>F</th>
<th>F</th>
<th>A</th>
<th>R</th>
<th>K</th>
<th>Q</th>
<th>M</th>
<th>I</th>
<th>K</th>
<th>B</th>
<th>W</th>
<th>L</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Query Sequence:

<table>
<thead>
<tr>
<th>F</th>
<th>F</th>
<th>A</th>
<th>R</th>
<th>K</th>
<th>Q</th>
<th>M</th>
<th>I</th>
<th>K</th>
<th>B</th>
<th>W</th>
<th>L</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Query and Subject:

©2010 Scott B. Baden /CSE 160/ Winter 2010
Dependence Structure
Protein Sequence Example

- http://www.uniprot.org/downloads
Parallel Control Flow

• 2 choices
  ▶ Distribute the query sequence across threads, one subject sequence at a time
  ▶ Compute separate queries in different threads, each thread has a different subject sequence

• What are the issues?
Dependence Structure
Inter-block Dependencies

Thread 0

Thread 1

Thread 2

Thread 3

1,1 1,2 1,3

1,4 1,5 1,6

1,7 1,8 1,9

1,10 1,11 1,12

2,1 2,2 2,3

2,4 2,5 2,6

2,7 2,8 2,9

2,10 2,11 2,12

3,1 3,2 3,3

3,4 3,5 3,6

3,7 3,8 3,9

3,10 3,11 3,12

4,1 4,2 4,3

4,4 4,5 4,6

4,7 4,8 4,9

4,10 4,11 4,12

5,1 5,2 5,3

5,4 5,5 5,6

5,7 5,8 5,9

5,10 5,11 5,12

6,1 6,2 6,3

6,4 6,5 6,6

6,7 6,8 6,9

6,10 6,11 6,12

©2010 Scott B. Baden /CSE 160/ Winter 2010
Computational Rounds
How do we implement this?

• Synchronization?
• Locality?
• False sharing?
• Load imbalance?