Assignment 1 2IN05 2008

Ruud Koolen - 0595143
Stef Louwers - 0590864

September 24, 2008

1 Overview

We faced three major synchonization-related issues while writing this program:

1. Exclusive access to the input stream;
2. Writing output buckets in the same order input buckets were read;
3. Sending the sorted buckets to the output thread using a bounded buffer.

2 Input stream synchronization

The first problem we faced was the problem of reading the input buffer in such a way that no more than one thread would use the input buffer at any time. We solved this problem using a mutex to control access to this input buffer: we introduced a mutex variable "inputstream_mutex" with the following interpretation:
"One cannot touch the input_buffer, input_buckets, current_input_item, and current_bucket variables when one does not hold the inputstream_mutex mutex."
By enforcing this condition in our program, we solved the problem of exclusive access to the input stream.

3 Output stream ordering

We solved this problem by introducing a queue of threads waiting to transport their data to the output thread; a thread adds itself to the end of the queue while reading a bucket from the input stream, and it can only execute the transport process when it's the first item in the queue. Access to the queue is protected by a mutex (outputstream_mutex), and a condition variable (outputstream_cond) is used by sorting threads to wait for the moment that they are first in the queue. Sorting threads execute a sigall operation on this condition variable whenever they finish transporting their bucket to the output thread, so that the next thread in line can start its work.

4 Bounded buffer

The last problem was to transport a potentially large amount of data from an arbitrary thread (in this case, any sorting thread) to the output thread using a bounded buffer. We used the obvious standard implementation for this problem: two semaphores, one that denotes the number of data elements in the buffer, and one that denotes the amount of free space in the buffer; these semaphores are incremented and decremented or decremented and incremented when writing to the buffer and reading from the buffer, respectively. The output stream ordering system ensures that no more than one thread can write its bucket of data to the bounded buffer, as to avoid interleaving of different buckets during the transport phase.