Advanced Computer Architecture
Bonus Assignment 3 — Scalability and Programming

Instructions

- The assignment should be solved *individually*.
- Solutions should be properly *motivated*, a few short sentences is usually enough.
- Answer the questions in *your* own words. Copying answers from other sources, such as the text book, is not acceptable. You may quote other sources, in that case, make sure to include the source and say how you interpret the quoted text.
- Solutions should be given in *English* or *Swedish*. English is preferred.
- Solutions should be handed in *before* the deadline by:
  - Posting them in *Jonas Flodin's* mailbox (house 1, floor 4).
  - Sending an email to `jonas.flodin@it.uu.se`. The *answers should be in the message body*, solutions attached as Word documents, or similar, will be ignored. If you need to include graphics use one of the following formats: PNG, JPEG, SVG, PDF, EPS, PS
- Fill out and include this cover page when you hand in your solution. If you are submitting by email, make sure to include the same information in the email.
- Unreadable solutions will be treated as incorrect.
- You need to score at least 9 points to get a bonus point on the exam.

Deadline

Solutions handed in after the deadline specified on the course homepage will be marked on a best effort basis and will not result in any bonus on the exam.

Student

Name

Civic registration number (personnummer)

Email address

Date

1 Scalable Multiprocessors

1. What is the difference between NUMA and UMA architectures? (1)

2. What is the main difference between a bus based cache coherency protocol and a directory based one? Why does the latter provide better scalability? (2)

3. In a directory based coherence protocol, a special data structure, the directory, is used to keep track of where a particular cache line is stored. In a naïve implementation, the directory has to be large enough to store book-keeping for every cache line in every cache in the system.

   Instead of using a bit vector with a bit representing each processor in the system, the *limited pointers* approach stores a limited number of pointers to processors that store a specific cache line. This allows the directory to scale to a larger number of processors than the bit vector.

   The number of pointers that can be stored per cache line might not be enough to represent all copies of that line if the line is shared between a large number of processors. In this case, we have an *overflow* condition.

   Describe *two* different methods to handle *overflows*.

   (2)

4. The directory normally holds a pointer to all copies of a cache line. The Scalable Coherence Interface (SCI) has a clever way of keeping track of multiple copies of a cache line that makes the directory size independent of the number of nodes in a system. What kind of data structure is used to keep track of cache lines in SCI? Where is this structure stored?

   (2)

2 Programming Multiprocessors

5. What's the main difference when programming a message passing system compared to when programming a shared memory system? (1)

6. A process in most operating system contains at least the following resources:

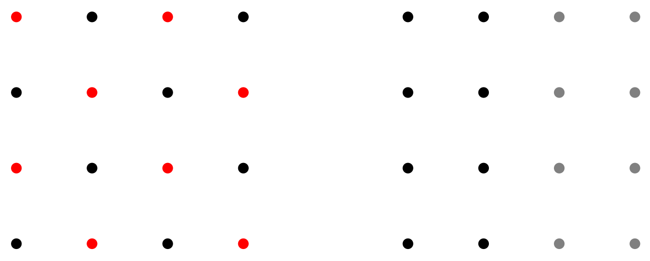
   - Program counter
   - Stack pointer
   - File descriptors
   - Virtual to physical memory mappings

   Which of the above resources are *not* shared between threads? (1)

7. In which of the following frameworks does *false sharing* normally *not* occur:

   - MPI
   - Pthreads
   - OpenMP

   (1)



(a) Red-Black (b) Vertical blocks

Figure 1: Two different approaches to parallelizing the Gauss-Seidel algorithm. The red-black implementation provides parallelism by breaking up the dependencies between elements, i.e. all the elements of one color can be processed in parallel. The block wise parallelization uses large chunks of elements which are computed in parallel.

8. There are multiple ways to implement a Gauss-Seidel sweep. Two of them were discussed in Sverker's lecture.

   The *red-black* breaks the dependencies between elements by coloring even elements and odd elements with different colors (Figure 1a). The algorithm only updates one color per iteration and alternates between the colors. This allows all the element in one iteration (i.e. all the elements of one color) to be updated in parallel.

   The striped implementation (Figure 1b) divides the matrix into vertical blocks, where each block can be updated in parallel.

   There are a several problems with the red-black version, one of them being a lower convergence rate. Mention one problem that is specific to the red-black version that is due to architectural issues when implemented on a shared memory system.

   (1)