Operating Systems, Spring 2020, Exercise 1 (Lectures 1-2)

Please bring your solutions (on paper, PC, or tablet) to the practice session, so that it is easy to discuss your solutions with the group. It is not sufficient just to say that you have done some problem with nothing to show.

1. OS overview
    a. What is the difference between OS and OS kernel?
    b. What is the relationship between multithreading, multicore, and SMP?
    c. Explain the difference between a monolithic kernel and a microkernel.
    d. An I/O-bound program is one that, if run alone, would spend more time waiting for I/O than using a processor. A processor-bound program is opposite. Suppose a short term scheduling algorithm favors those programs that have used little processor time in the recent past. Explain why this algorithm favors I/O-bound programs and yet does not permanently deny processor time from processor-bound programs. (Probl 2.2 [Sta12])

2. Cache
    a. Why is large cache block size better than small cache block size?
    b. Why is many cache lines better than just a few?
    c. What is the cache mapping function? What problem does it solve?
    d. How do you find the referred word (e.g., memory address 0x1234ABCD) from a cache? (Assume cache hit.)
    e. What is a cache miss and what is the cost of cache miss?
    f. Why is it more difficult to design caches for multicore systems than for 1-core systems?
    g. In a CPU, cache access time is 500 ns and main memory access time is 2 µs. If a block is found, on average, 90% of the time in cache, what is the average access time?

3. (Review Questions 4.6, 4.7, 4.8, and 4.X [Sta12])
    a. 4.6: List three advantages of ULTs over KLTs.
    b. 4.7: List three disadvantages of ULTs over KLTs.
    c. 4.8: Define jacketing.
    d. 4.X: Assume that thread T in process P makes a blocking system call. What difference is there whether T is ULT or KLT? Why does this difference exist? Which one is better in this case? Why?

4. (Problems 4.7 and 4.8 [Sta18])
    4.7 Many current language specifications, such as for C and C++, are inadequate for multithreaded programs. This can have an impact on compilers and the correctness of code, as this problem illustrates. Consider the following declarations and function definition:

```
int global_positives = 0;
typedef struct list {
     struct list *next;
     double val;
     } * list;

void count_positives(list l)
{
     list p;
     for (p = l; p; p = p -> next)
         if (p -> val > 0.0)
             ++global_positives;
}
```

    Now consider the case in which thread A performs

```
count_positives(list containing only negative values);
```

    while thread B performs

```
++global_positives;
```

    a. What does the function do?
    b. The C language only addresses single-threaded execution. Does the use of two parallel threads create any problems or potential problems?

    4.8 But some existing optimizing compilers (including gcc, which tends to be relatively conservative) will “optimize” count_positives to something similar to

```
void count_positives(list l)
{
   list p;
   register int r;

   r = global_positives;
   for (p = l; p; p = p -> next)
       if (p -> val > 0.0) ++r;
   global_positives = r;
}
```

    What problem or potential problem occurs with this compiled version of the program if threads A and B are executed concurrently?

5. (Problem 4.10 modified. [Sta12])
   Suppose a single application is running on a multicore system with 16 processors. If 20% of the code is inherently serial, what is the performance gain over a single processor system?
   What would it be if only 2% of the code would be inherently serial?

6. (Problem 4.11[Sta18])
   In Solaris 9 and Solaris 10, there is one-to-one mapping between ULTs and LWPs. In Solaris 8, a single LWP supports one or more ULTs.
   a. What is the possible benefit of allowing a many-to-one mapping of ULTs to LWPs?
   b. In Solaris 8, the thread execution state of a ULT is distinct from that of itw LWP. Explain why.
   c. Figure 4.18 shows the state transition diagrams for a ULT and its associated LWP in Solaris 8 and Solaris 9. Explain the operation of the two diagrams and their relationships.

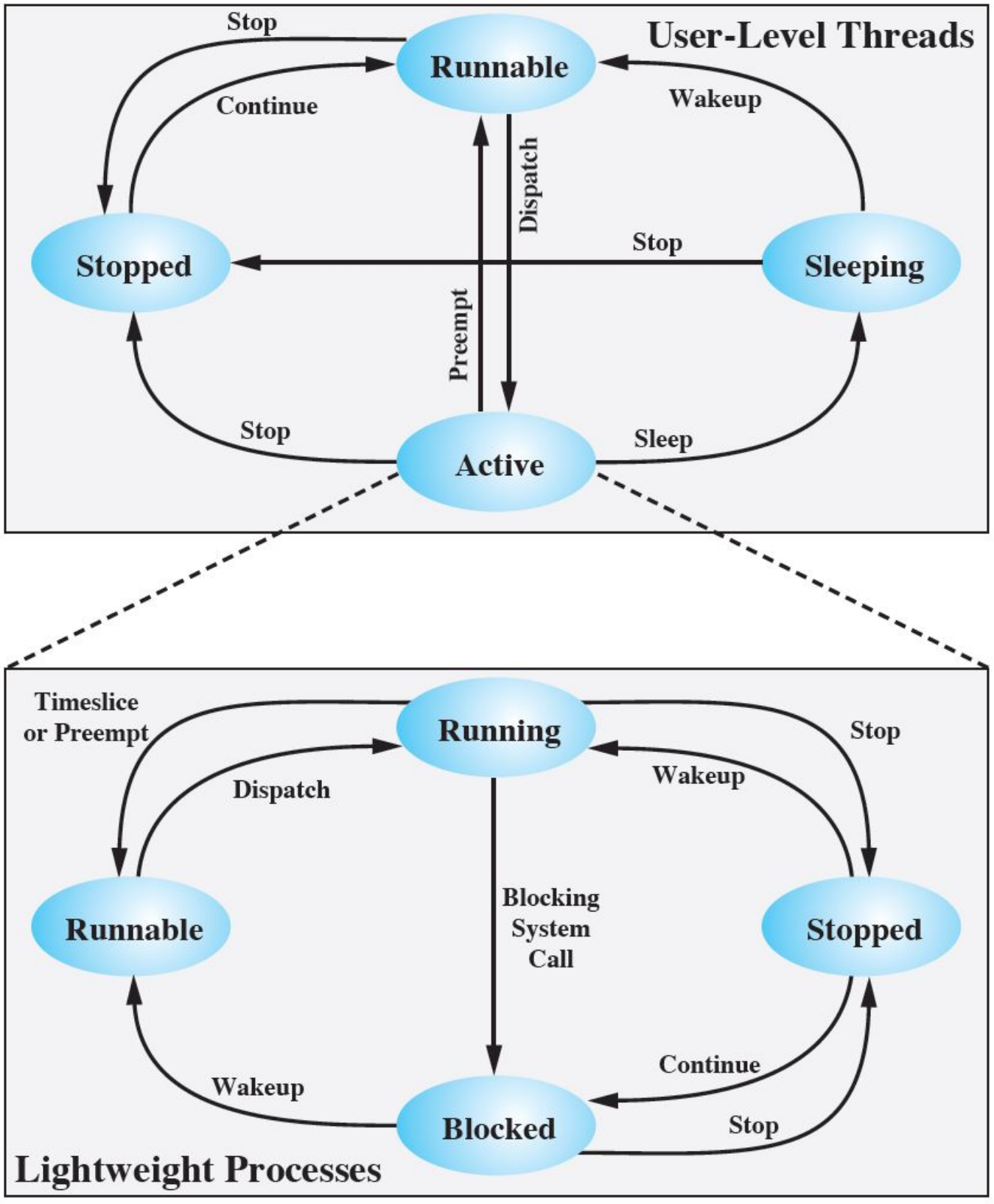

Figure 4.18 Solaris User-Level Thread and LWP States