An Introduction to Parallel Programming: Errata

Peter Pacheco

Last update December 21, 2022

General

- Kindle edition only. The plural of a C type is printed as the type followed by a space and an "s." For example, "`doubles`" is printed as "`double` s." (May 21, 2011)
- Kindle edition only. Formatting of displayed code that is not enclosed in a box has no indentation. (May 26, 2011)

Table of Contents

- Kindle edition only. The link to Chapter 6 takes you to the first paragraph of Chapter 6. However, this paragraph is placed between the end of the Chapter 6 Exercises and the beginning of the Chapter 6 Programming Assignments. (November 26, 2011)

Chapter 2

- Section 2.3.3, p. 37, next to last sentence in paragraph 3: The number of links in a ring is p and the number of links in a toroidal mesh is $2p$. (Sept 8, 2011; corrected in second printing.)

  Tadeus Prastowo pointed out that if the links joining the processor and the switch are included, the formula given in the first printing is correct: $2p$ links for a ring, and $3p$ for a toroidal mesh. The "correction" only counts links joining two switches.

  The formula that is used should depend on the hardware under consideration. For example, if the processor and switch are on the same silicon wafer, but two processor-switch pairs are on different wafers, then the processor-switch link will probably be much faster, and the formulas p for a ring and $2p$ for a toroidal mesh will be closer to measured performance. If, however, the processors and switches are all on the same wafer, then the formulas $2p$ for a ring and $3p$ for a toroidal mesh may be better. (May 23, 2017)

- Section 2.3.3, p. 38, formulas in the second paragraph. Tadeus Prastowo also pointed out that some printings of IPP are missing parts of the three formulas. In order, the formulas in the paragraph should be

  $$p = q^2, \ldots,\ 2q = 2\sqrt{p}, \ldots,\ 2\sqrt{p}.$$

  (May 25, 2017)

- Section 2.3.3, p. 38, last sentence. The plus sign (`+`) should be a minus sign (`-`). That is, the formula for the number of links should be $p^2/2 - p/2$. Thanks to Curtis Gehman for spotting this. (November 6, 2011, corrected in second printing.)

- Section 2.3.4, p. 46, second pair of displayed, nested for loops: the outer loop for core 1 should start with `iter_count`, not `iter_count+1`:

  ```
  for (i = iter_count; i < 2*iter_count; i++)
  ```

  (February 24, 2012)

- Section 2.6.1. Chinese edition only. The formula for speedup is incorrect. The Chinese edition has

  $$S = \frac{T_{\text{parallel}}}{T_{\text{serial}}},$$

  and this *should be*

  $$S = \frac{T_{\text{serial}}}{T_{\text{parallel}}}.$$

  Thanks to Huang Xin for spotting this. (December 21, 2022)

- Section 2.7.1, p. 69, Figure 2.2.2: It looks like there is an arrow from the oval in the third row labelled `loc_bin_cts[b-1]++` to the oval labelled `bin_counts[b]+=`. This is actually a continuation of the arrow from the oval labelled `loc_bin_cts[b]++` in the second row. (July 5, 2011; corrected in second printing.)

- Exercise 2.16b, p. 79. Feng Chen pointed out that this is false. As an example, he gives $T_{serial} = 8 + n^2$ and $T_{overhead} = n$. Then clearly, T_{serial} grows faster than $T_{overhead}$, since

  $$\frac{d}{dn}T_{serial} = 2n > 1 = \frac{d}{dn}T_{overhead}$$

  for all $n \geq 1$. However, if, for example, $p = 2$, $n_1 = 1$ and $n_2 = 2$, we see that

  $$E(n_1) = \frac{1}{1 + 2 \cdot 1/(8+1)} = 9/11 \approx 0.82,$$

and

$$E(n_2) = \frac{1}{1 + 2 \cdot 2/(8+4)} = 3/4 = 0.75.$$

However,

$$T_{overhead}(n_2) - T_{overhead}(n_1) = 2 - 1 = 1,$$

and

$$T_{serial}(n_2) - T_{overhead}(n_1) = 12 - 9 = 3.$$

So $T_{overhead}$ is growing more slowly than T_{serial}, and the efficiency is decreasing. (May 11, 2017)

Chapter 3

- Section 3.1.1, p. 85, displayed command for compiling: Mohammed Sourouri points out that many C compilers will object to the `for` statement in Program 3.1:

  ```
  for (int q = 1; q < comm_sz; q++)
  ```

  This is because it includes a declaration of the variable `q` in the `for` statement. This can be suppressed if you're using `gcc` by including the `-std=c99` option on the command line:

  ```
  $ mpicc -g -Wall -std=c99 -o mpi_hello mpi_hello.c
  ```

  Thanks to Mohammed for spotting this. (February 19, 2012)

- Section 3.3.2, p. 100, code at bottom of page: call to `Get_data` should be a call to `Get_input`. Thanks to Wes Brewer for spotting this. (Oct 12, 2011; corrected in second printing.)

- Section 3.4.6, p. 110, Table 3.4: column 10 "+6'" should be "2". Thanks to Andrew Neiderer for spotting this. (January 14, 2012; corrected in second printing.)

- Section 3.5, p. 117, end of second paragraph: needs a closing period after `MPI_Pack/Unpack`. (Corrected in second printing.)

- Section 3.6.2, p. 125: Line 3 should be

  $$T_{\text{parallel}}(16,384,\ 4) = 3.9 \times T_{\text{parallel}}(8192, 4)$$

  (Corrected in second printing.)

- Exercise 3.4, p. 140: The possessives in the last line should have apostrophes: "process 0's output first, then process 1's ..." (Corrected in second printing.)
- Exercise 3.11b, p. 142: "`x_is`" (in fixed-width font) should be "x_is" (in math font). (July 14, 2011; corrected in second printing.)

Chapter 4

- Section 4.2, p. 154, Program 4.1. line 33. The assignment

  ```
      long my_rank = (long) rank
  ```

  should be terminated by a semi-colon. Thanks to Wes Brewer for spotting this. (January 14, 2012; corrected in second printing.)
- Section 4.9.2, p. 186, Program 4.12. The published version fails to lock the mutex associated with the first node of the list. The code should be

  ```
  int Member(int value) {
     struct list_node_s *temp_p, *old_temp_p;

     pthread_mutex_lock(&head_p_mutex);
     temp_p = head_p;

     /* If list is not empty, acquire the mutex
      * associated with first node */
     if (temp_p != NULL)
        pthread_mutex_lock(temp_p->mutex);

     /* Don't need head_p mutex anymore */
     pthread_mutex_unlock(&head_p_mutex);

     while (temp_p != NULL && temp_p->data < value) {
        if (temp_p->next != NULL)
           pthread_mutex_lock(&(temp_p->next->mutex));

        /* Advance to next element */
        old_temp_p = temp_p;
        temp_p = temp_p->next;
  ```

```
      /* Now unlock previous element's mutex */
      pthread_mutex_unlock(&(old_temp_p->mutex));
   }

   if (temp_p == NULL || temp_p->data > value) {
      if (temp_p != NULL)
         pthread_mutex_unlock(&temp_p->mutex);
      return 0;
   } else { /* temp_p != NULL && temp_p->data == value */
      pthread_mutex_unlock(&temp_p->mutex);
      return 1;
   }
}  /* Member */
```

I'm grateful to Steffen Christgau and Bettina Schnor for both finding and correcting the errors. (February 21, 2017)

- Section 4.11, p. 197, fourth and fifth sentences of last paragraph: "In some cases, the C standard specifies an alternate, thread-safe version of a function. In fact, there is a thread-safe version of `strtok`." Ivar Ursin Nikolaisen pointed out that this seems to suggest that `strtok_r` is part of the C standard. He goes on to observe that "`strtok_r` is not in the current C standard. It is however in Posix.1-2008. In the upcoming C1X standard there is a similar (optional) function `strtok_s` introduced in annex k." Thanks to Ivar for pointing this out. (November 3, 2011)

- Section 4.11, p.197, last sentence: The text in fixed width font, `saveptr Append ''_p'' to ''saveptr''`, should just be `saveptr_p`. Thanks to Lucas Levrel for spotting this. (February 2, 2012)

- Section 4.12, p. 200, second paragraph, next to last sentence. "Serval" should be "several." Thanks to Wes Brewer for spotting this. (January 14, 2012; corrected in second printing.)

- Kindle edition only. Exercise 4.7: next to last sentence should end with "thread $(q-1+t) \bmod t$?" The "mod t?" is missing.

- Kindle edition only. Exercise 4.15: the sizes of the matrices should be $k \times (k \cdot 10^6)$, $(k \cdot 10^3) \times (k \cdot 10^3)$, and $(k \cdot 10^6) \times k$. (Corrected in second printing.)

- Exercise 4.17e, p. 206: "falses sharing" should be "false sharing." (May 21, 2011; corrected in second printing.)

- Programming Assignment 4.2, p. 206: "it's area" should be "its area." (May 21, 2011; corrected in second printing.)

Chapter 5

- Section 5.1.2, p. 212, last paragraph: in the prototype for `strtol`, "`number p`" should be "`number_p`" and "`end p`" should be "`end_p`." Thanks to Lucas Levrel for spotting these. (February 2, 2012)
- Section 5.10, p. 257, Caption for Program 5.6: "multi threaded" should be "multi-threaded". (Corrected in second printing.)
- Section 5.10, p. 258, fourth and fifth sentences of second paragraph. See note for Section 4.11, p.197, above. (Nov 3, 2011)
- Exercise 5.5, p. 263. The array `a` should be declared as

  ```
  float a[] = {2.0, 2.0, 4.0, 1000.0};
  ```

  (Corrected in second printing.)
- Exercise 5.12, p. 265. The parenthesis before "With `gcc`" should be deleted. (Corrected in second printing.)
- Programming Assignment 5.4, p. 269: the last line before the bulleted list should be terminated with a colon, not a period. (Corrected in second printing.)

Chapter 6

- Kindle edition only. The introductory paragraph to Chapter 6 has been placed between the end of the Chapter 6 Exercises and the beginning of the Chapter 6 programming assignments. (Nov 26, 2011)
- Section 6.1.2, p. 274, Program 6.1: the computations of `force_qk[X]` and `force_qk[Y]` are missing a minus sign. The computations should be

  ```
  force_qk[X] = -G*masses[k]* ... ;
  force_qk[Y] = -G*masses[k]* ... ;
  ```

  This mistake is repeated in several other locations: Section 6.1.3, p. 280, code at the top of the page; Section 6.1.6, p. 285, code at the top of the page; and Section 6.1.6, p. 287, code in the second paragraph. Note that the online code does not have these errors. Thanks to Mingzhe Fang for finding these errors. (Jan 11, 2018)

- Section 6.1.2, p. 277, last block of displayed code: the line

  ```
  forces = memset(forces, 0, n*sizeof(vect_t);
  ```

  is missing a closing parentheses. It should be

  ```
  forces = memset(forces, 0, n*sizeof(vect_t));
  ```

  Thanks to Lucas Levrel for spotting this. (February 2, 2012)
- Section 6.1.6, p. 284, fourth line from bottom of page: the comment `/* Can be faster than memset */` should be deleted. (Corrected in second printing.)
- Section 6.2.1, p. 302, Program 6.4: The argument `city` to the function `Remove_last_city` isn't needed (and isn't used in the program). (Corrected in second printing.)
- Exercise 6.15, p. 345, last line: The list of basic arithmetic operators should include the modulus operator `%`. (Oct 15, 2011)
- Kindle edition only. Exercise 6.23, second sentence. The word "information" is enclosed in `\hbox{` ...`}`. The `\hbox{` and the `}` shouldn't be in the text. (November 26, 2011; corrected in second printing.)

Source Code

- Chapter 3: Deleted redundant code from `Merge_low` function in `mpi_odd_even.c` (July 26, 2011)
- Chapter 4: Added `pth_mat_vect_rand_split.c` to archive.
- Chapter 4: Modified `pth_mat_vect_rand_split.c` (May 10, 2011)
- Chapter 5: Added `omp_mat_vect_rand_split.c` to archive. (May 26, 2011)
- Chapter 5: Modified documentation for `omp_mat_vect_rand_split.c`. Added `timer.h`. (July 5, 2011)
- Chapter 6: Modified documentation for `Compute_force` function in `omp_nbody_basic.c`. (October 23, 2011)
- Chapter 6: Modified documentation for `mpi_tsp_stat.c`: eliminated reference to nonexistent file `ipp_mpi.c`. (January 15, 2012)

PowerPoint Slides

- Chapter 3, Table 3.5, slide 93. "Seconds" should be "milliseconds." Thanks to Wes Brewer for spotting this. (January 14, 2012)