

Excercise Sheet 1

Solution

Lecture Parallel Processing

Winter Term 2025/26

Exercise 1: Run-Time Analysis of a Sequential Program

The output of `gprof` is as follows (shortened for brevity):

Flat profile:

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
62.52	24.18	24.18	260	0.09	0.09	funct7()
9.14	27.72	3.53	5551	0.00	0.00	funct1()
6.73	30.32	2.60	26	0.10	1.24	funct8()
6.50	32.83	2.51	3913	0.00	0.00	funct3()
5.15	34.82	1.99	39	0.05	0.21	funct2()
3.39	36.14	1.31	13	0.10	2.82	funct5()
1.01	36.53	0.39	13	0.03	0.24	funct4()
0.31	36.65	0.12	13	0.01	1.25	funct6()
0.26	36.75	0.10				main
...						

Call graph (explanation follows)

granularity: each sample hit covers 2 byte(s) for 0.03% of 36.75 seconds

index	% time	self	children	called	name
					<spontaneous>
[1]	100.0	0.10	36.65		main [1]
		1.31	35.34	13/13	funct5() [2]

		1.31	35.34	13/13	main [1]
[2]	99.7	1.31	35.34	13	funct5() [2]
		0.12	16.07	13/13	funct6() [5]
		1.30	14.77	13/26	funct8() [3]
		0.39	2.69	13/13	funct4() [9]

		1.30	14.77	13/26	funct6() [5]
		1.30	14.77	13/26	funct5() [2]
[3]	87.4	2.60	29.53	26	funct8() [3]
		24.18	0.00	260/260	funct7() [4]
		1.33	4.02	26/39	funct2() [6]

		24.18	0.00	260/260	funct8() [3]
[4]	65.8	24.18	0.00	260	funct7() [4]

[5]	44.1	0.12	16.07	13/13	funct5 () [2]
		0.12	16.07	13	funct6 () [5]
		1.30	14.77	13/26	funct8 () [3]
[6]	21.8	0.66	2.01	13/39	funct4 () [9]
		1.33	4.02	26/39	funct8 () [3]
		1.99	6.03	39	funct2 () [6]
		2.50	2.48	3900/3913	funct3 () [7]
		1.04	0.00	1638/5551	funct1 () [8]
[7]	13.6	0.01	0.01	13/3913	funct4 () [9]
		2.50	2.48	3900/3913	funct2 () [6]
		2.51	2.49	3913	funct3 () [7]
		2.49	0.00	3913/5551	funct1 () [8]
[8]	9.6	1.04	0.00	1638/5551	funct2 () [6]
		2.49	0.00	3913/5551	funct3 () [7]
		3.53	0.00	5551	funct1 () [8]
[9]	8.4	0.39	2.69	13/13	funct5 () [2]
		0.39	2.69	13	funct4 () [9]
		0.66	2.01	13/39	funct2 () [6]
		0.01	0.01	13/3913	funct3 () [7]

...

Copyright (C) 2012–2018 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification, are permitted in any medium without royalty provided the copyright notice and this notice are preserved.

Note that if you repeat the measurement on your computer, you may get different times (but the same number of calls).

- a) The function with the highest exclusive computing time is `funct7()`, which has a self time of 24.18 seconds (62.52% of the program's runtime, see flat profile).
- (i) According to Amdahl's law, the maximum speedup would be
- $$S(p) \leq \frac{1}{1 - 0.6252} = 2.668$$
- (ii) The function is called 260 times (see flat profile).
- (iii) From the call graph, we can see that `funct7()` calls no other functions (see the entry labelled [4]).
- b) The function that is called most often is `funct1()` (see flat profile). It is called 1638 times from `funct2()` and 3913 times from `funct3()` (see the entry labelled [8] in the call graph).
- c) `funct5()` itself just needs 1.31 seconds (for a total of 13 invocations), however, its children (`funct6()`, `funct8()` and `funct4()`) together need 35.24 seconds (see the entry labelled [2] in the call graph).

Exercise 2 (Compulsory): Data Dependences, Programming with C++ Threads

Submit until Tuesday, November 11th, 10:00 via [moodle](#)

Exercise 3 (Compulsory): Parallelization of Quicksort using C++ Threads

Submit until Tuesday, November 11th, 10:00 via [moodle](#)