



Факультет программной инженерии и компьютерной техники

Параллельные вычисления

Лабораторная работа №1

Автоматическое распараллеливание программ

Преподаватель: Жданов Андрей Дмитриевич

Выполнил: студент: Кульбако Артемий Юрьевич, Р4115

Санкт-Петербург
2023

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ	2
ЗАДАНИЕ.....	3
ВЫПОЛНЕНИЕ.....	3
ВЫВОД.....	6
ИСХОДНЫЙ КОД	6

ЗАДАНИЕ

На компьютере с многоядерным процессором установить ОС Linux и компилятор GCC версии не ниже 4.7.2. При невозможности установить Linux или отсутствии компьютера с многоядерным процессором можно выполнять лабораторную работу на виртуальной машине. Минимальное количество ядер при использовании виртуальной машины - 2.

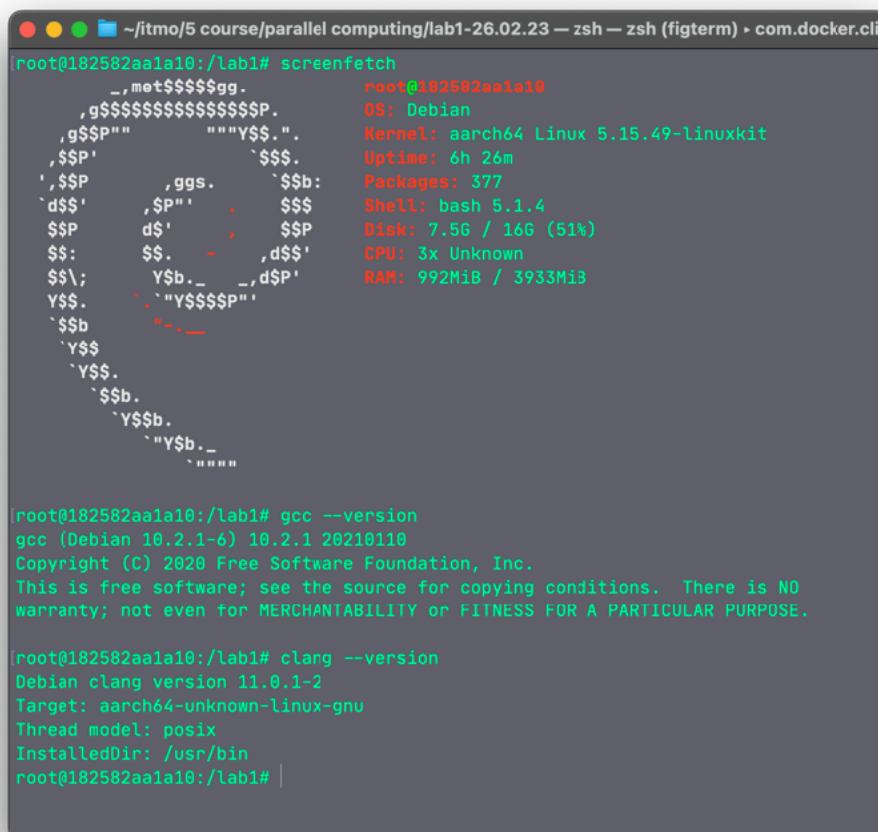
На языке C написать консольную программу по алгоритму по варианту и провести замеры влияния автоматического распараллеливания на время выполнения.

$$A = 392; \text{map} = [3,1]; \text{merge} = 5; \text{sort} = 3$$

* данные функции были модифицированы по индивидуальной договорённости с преподавателем.

ВЫПОЛНЕНИЕ

Характеристики виртуальной машины (Docker):



```
root@182582aa1a10:/lab1# screenfetch
_,met$$$$$gg.
,g$$$$$$$$$$$$$P.
,g$P'          "Y$.
,$$P'          $$$
',,$P'         ,ggs.  `$$b:
`d$$'          ,P"    .  $$$
$$P'           d$'     ,,$P
$$:           $$      ,d$$'
$$\;          Y$b._   _dP'
Y$$          '."Y$$$$P'
`$$b         "_,_
`Y$b
`Y$$
`Y$.
`$$b.
`Y$$b.
`"Y$b._
`""""

root@182582aa1a10
OS: Debian
Kernel: aarch64 Linux 5.15.49-linuxkit
Uptime: 6h 26m
Packages: 377
Shell: bash 5.1.4
Disk: 7.5G / 16G (51%)
CPU: 3x Unknown
RAM: 992MiB / 3933MiB

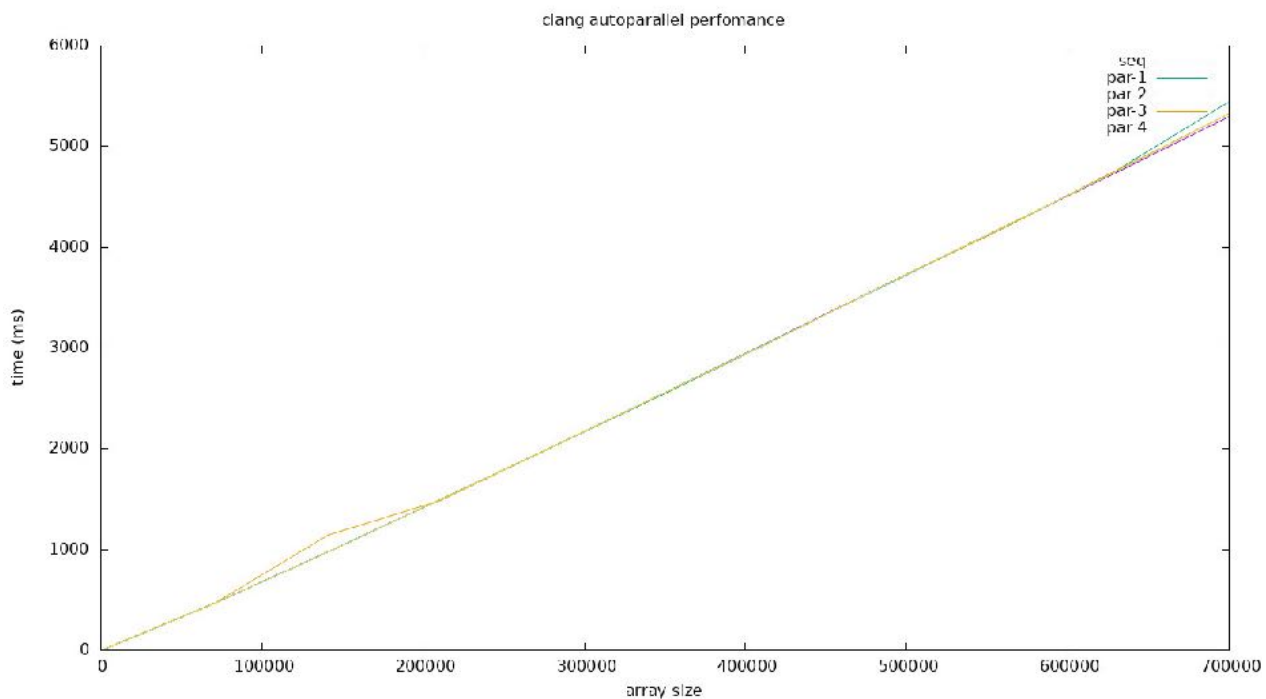
root@182582aa1a10:/lab1# gcc --version
gcc (Debian 10.2.1-6) 10.2.1 20210110
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

root@182582aa1a10:/lab1# clang --version
Debian clang version 11.0.1-2
Target: aarch64-unknown-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
root@182582aa1a10:/lab1#
```

Результаты тестирования:

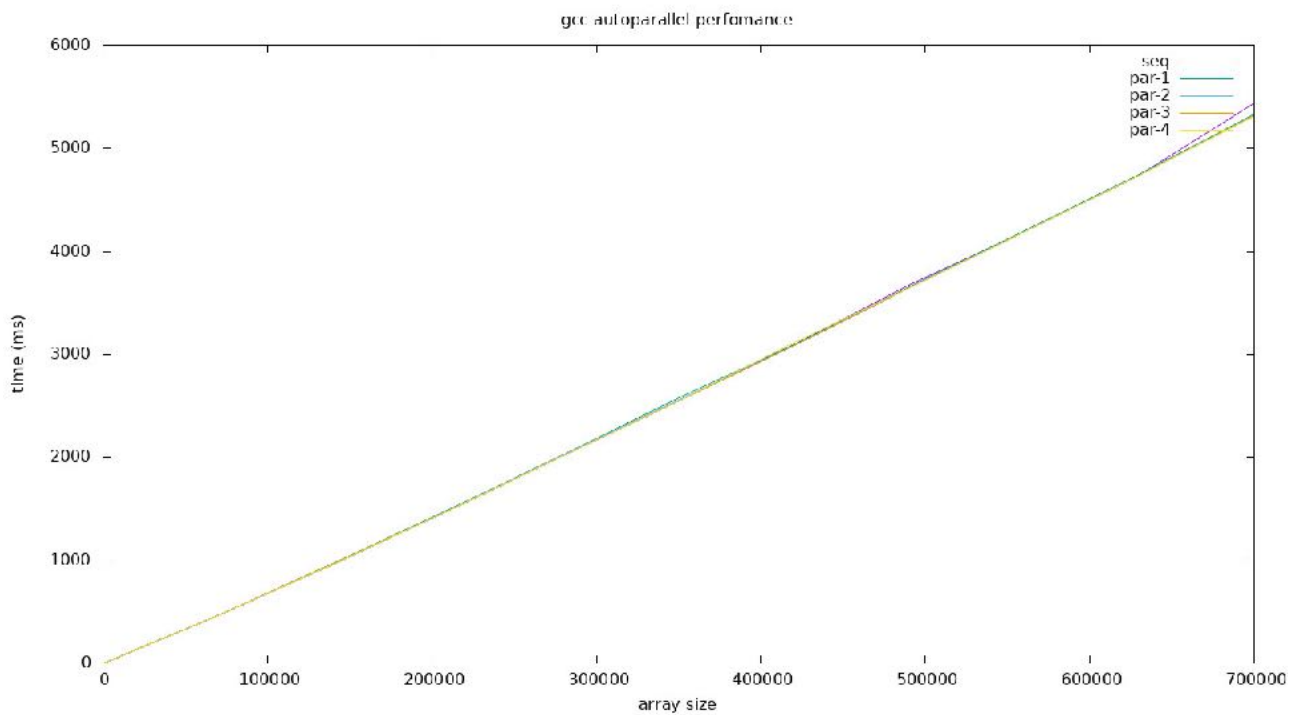
clang

N	seq	par-1	par-2	par-3	par-4
900	2	2	2	2	2
70809	472	467	466	466	467
140718	980	981	976	1141	974
210627	1491	1492	1493	1489	1492
280536	2027	2025	2021	2026	2019
350445	2561	2554	2562	2559	2564
420354	3107	3097	3104	3094	3095
490263	3654	3652	3645	3653	3655
560172	4203	4193	4200	4197	4201
630081	4738	4757	4751	4758	4753
699990	5294	5439	5328	5330	5330



gcc

N	seq	par-1	par-2	par-3	par-4
900	2	2	2	2	2
70809	466	471	471	468	466
140718	971	974	971	975	970
210627	1488	1491	1488	1488	1490
280536	2022	2028	2035	2021	2018
350445	2548	2580	2556	2551	2553
420354	3095	3100	3092	3092	3119
490263	3679	3656	3644	3645	3648
560172	4192	4203	4192	4195	4192
630081	4741	4751	4735	4738	4736
699990	5442	5332	5321	5321	5316



ВЫВОД

Параметры автопараллелизма компилятора не дали никакого прироста в производительности. На графиках видно, что время выполнения почти не отличается, как и отсутствует разница между компиляторами. Первое неудивительно: из-за присутствия в алгоритме операций обработки данных i элемента массива в паре с $i-1$ элементом, при этом, данная операция является модифицирующей, компилятор не может полноценно распараллелить программу. Второе (разница между компиляторами) скорее всего следует из того, что оба компилятора open-source и уже позаимствовали лучшие практики друг друга.

Большого всего меня поразило то, как много операций с числами может выполнить компьютер даже сильно ограниченный в ресурсах: рискну предположить (но не буду утверждать наверняка), что это следствие ARM архитектуры, которая хорошо подходит для выполнения простых команд, коими и являются математические операции из данных алгоритмов.

ИСХОДНЫЙ КОД

<https://github.com/testpassword/Parallel-computing/tree/master/lab1-26.02.23>

