UNIVERSIDAD NACIONAL DEL ALTIPLANO
"UNA PUNO"

ESCUELA PROFESIONAL DE INGENIERIA
ESTADISTICA E INFORMATICA

使用线程优化生命事件数据分析

Coasaca Callacondo Rubi melania
rcoasacac@est.unap.edu.pe
Chata Iscarra Grylia Yaneth
gchatai@est.unap.edu.pe
Fred Torres-Cruz
ftorres@unap.edu.pe

2022年11月28日

arXiv:2301.01710v1 [cs.DC] 4 Jan 2023

摘要

目标：使用线程优化生命事件数据（出生、死亡和婚姻）的分析时间。

方法：创建了一份不使用线程的 Python 代码和另一份使用线程的代码，随后他进行了 5 次测试，分别使用单个属性以及 1、2、3、4、5、6、7、8 和 9 个属性。两份代码均使用相同数量的数据进行测试，以确定使用线程的 Python 代码在解析生命事件数据时是否最优。

结果： 采用线程的 Python 代码效果最佳，因为它优化了 5 项测试的编译时间。通过将 1 个属性增加到 16 个属性，结果表明，分组的属性越多，线程的使用效率就越高。

结论： 采用线程的 Python 代码比不采用线程的代码更优。因此，建议在类似研究中使用线程分析数据。

引言：

秘鲁每天都会生成数百万条数据记录，其中一类记录是生命事件（出生、死亡和婚姻）。随着每天不断记录新的事件，这类记录也在逐日增加。由于数据量庞大，分析这些数据需要时间，而且

为解决这一问题，本研究旨在优化对生命事件登记处 500,000 条数据的分析，通过使用 Python 编写代码，借助并行计算实现数据分析优化，并采用线程。此项优化将通过读取时间以及对 9 个属性进行计数来展示。

方法：

并行计算：并行计算将一个大型问题划分为若干较小、彼此独立、有时相似的部分，由多个处理器同时执行。处理器通过共享内存进行通信，最终得到的结果如同所有工作都由一个通用算法依次执行一样。

线程： 线程是在单个程序中同时并行运行、执行不同任务的多个线程，[7] 将创建共享内存线程，以加快读取和计数 500000 条包含 9 个属性的数据的进程。

数据： 数据对应于国家开放数据平台“治理”下的“生命事件”选项，可通过以下链接查看：bit.ly/3udD9Wh 这些信息由国家统计与信息研究所（INEI）提供，数据来源为 RENIEC。

数据说明： 我们将使用 500,000 条 CSV 格式的生命事件数据，其中包含 9 个属性。这些数据最近一次更新于 2022 年 9 月。数据包含全国各地民事登记处登记的出生、死亡和婚姻记录，按登记年份和月份、性别、生命事件、登记类型，以及登记所在的部门、省和区细分。

数据字典：

Attribute	Description	Datatype
REGISTRATION_YEAR	Enrollment Record Year	Text
REGISTRATION_MONTH	Month of registration of the event	Text
CODE_ SEX	Sex of the record holder	Text
CODE_DONE	Vital event (Births, marriages or deaths)	Text
CODE_REGISTRATION_1	Type of registration: Ordinary, extemporaneous, judicial, foreign, etc.)	Text
CODE_REGISTRATION_2	Record of registration or generating record	Text
DEPA_CONT_1	Department where registration was made	Text
PROV_COUNTRY_1	Province where registration was made	Text
DIST_CITY_1	District where registration was made	Text

实现逻辑： 将使用 Python 语言编写代码，通过实现线程进行并行计算 [8]

- 加载 CSV 格式的生命事件数据，代码使用 Python 语言编写，并在 Google Colab 中运行。

- 代码开始读取并计数数据，直到完成对所请求属性的全部遍历。

- 对 500,000 条数据进行读取和计数，代码中指定了 1、2 或最多 9 个属性。

- 通过线程并行读取和计数 500000 条数据，缩短执行时间。

实现： 使用以下库、函数和一致性设置，编写了有线程和无线程的 Python 代码。- 使用 pandas 库导出和处理 500000 条数据。

- time 库负责控制数据的执行时间。

- 线程库允许代码同时执行多个操作。

- NumPy 库用于更快地创建矩阵并存储大量信息。 [2]

-数据导入后，我们继续提取数据属性，并选择要处理的属性。

-将属性分配给计时器后，计时器便开始运行。

-计数数据函数负责统计属于所请求属性的所有数据。

-它从已分配的第一个属性及其对应的读取值和计数开始。

-随后，它会根据所请求的属性，依次转到各个属性，并获取相应的计数。

-显示数据量，以及计数和读取数据所花费的时间。

伪代码：伪代码是设计算法最常用的方法之一，并且独立于所使用的编程语言。[11]

伪代码： [3]

```
beginning
export database
read database
database loop
        extract attributes
remove attributes not entered
function count(attribute)
        accountant
        loop data in database
                if data == attribute
                        counter+1
        print counter
attribute in attribute loop
    start thread (count, attribute)
```

库和无线程代码：

```
import pandas as pd
import numpy as np
import time
```

```
for i in data.index:
        if data[v][i] not in clase:
            clase.append(data[v][i])
            cont.insert(clase.index(data[v][i]), 1)
        else:
            cont[clase.index(data[v][i])] += 1
    print(v)
    for i in clase:
        print(i + ":")
        print(cont[clase.index(i)])
end = time.time()
print("Done, time taken", end - start)
```

库和多线程代码：

```
import pandas as pd
import time
from threading import Lock, Thread
```

```
start = time.time()
for v in atributos:
    print("hilo "+str(atributos.index(v)))
    t = Thread(target=contar,args = (v,data))
    t.start()
t.join()
```

这里展示了 Python 代码：它会根据所请求的属性读取并统计数据，随后仅展示实现线程的代码行部分。

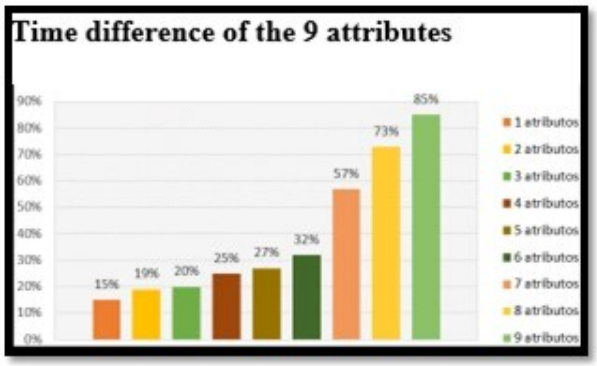
结果：

在两段代码中，50万条生命事件数据首先按1个属性编译，然后按2个属性编译……依此类推，直到完成9个属性的编译。同样，还进行了5次测试，分别使用5个独立属性（年份、登记月份、性别、事件和登记1）。

具有 9 个属性的时间差：

attributes	Compile time		Time difference (%)
	Without threads	With threads	
1	232.3663998	213.1211798	20.7%
2	225.3104243	205.8117213	19.2%
3	221.7641857	201.9206567	19.4%
4	226.0203583	205.9935672	19.8%
5	228.8250995	209.8369839	20%
6	255.7286234	236.1488495	18.9%
7	211.1879025	187.7333343	19.5%
8	241.0740626	222.0782423	21.4%
9	242.9323359	223.7528777	18.9%
10	232.3663998	213.1211798	19.1%

图表：



解读：从条形图中可以看出，属性1与属性9之间的差异很大；还可以观察到，条形呈升序排列。这说明，编译的属性越多，线程的使用效率越高。

均值检验：

Groups	N(attributes)	Half	Standard deviation	Mean Standard Deviation
Data with threads	9	16,56789342	,4957622161	,0953356132
without threads	9	42,55872547	3,1235762216	,8957626178

解释：

根据表格可以看出，未经优化的数据分析平均耗时为 42.55872547 秒，而经过优化后平均耗时为 16.56789342 秒，这表明通过使用

线程进行优化效果更好。因此，我们可以根据所分析的属性数量，得出响应时间和改进百分比；每次测试的改进百分比在 60% 到 65% 之间，如下表所示。

- **H_0 = There is no significant difference in response times**
- **H_1 = There is a significant difference in response times**

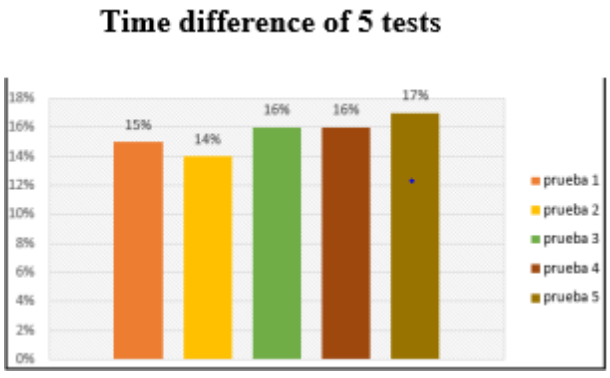
	Prueba de medias						
	t	df	Sig. (2 tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
						Lower	Upper
Datos	-24,043	17	,000	-16,89452	,69712	-18,34423	-15,33432
	-24,043	9,214	,000	-16,89452	,69712	-18,34423	-15,21421

解释： 对数据进行均值检验，设显著性水平为 0.05。我们拒绝零假设，这意味着在对比年份和登记情况时，响应存在显著差异；数据总量为 500000 条。

5 次测试的时间差：

Tests	Compile time		Time difference (%)
	Without threads	With threads	
1	5.256921529798975	5.102679014205933	15%
2	5.335506439208984	5.197928428649902	14%
3	5.15278930664062	4.9940960407257	16%
4	5.184887409210205	5.026702404022217	16%
5	5.10582280158996	4.93348098754882	17%

图表：



解读：条形图显示，测试 1 和测试 5 之间的差异

几乎相同，这使我们可以得出结论：单独编译每个属性时，使用线程可提升 16%。

平均值： 为此，我们将使用以下公式：

$$\overline{x} = \frac{x_1 + x_2 \ldots + x_n}{n}$$

$$\overline{x} = \frac{15 + 14 + 16 + 16 + 17}{5}$$

$$\overline{x} = 16$$

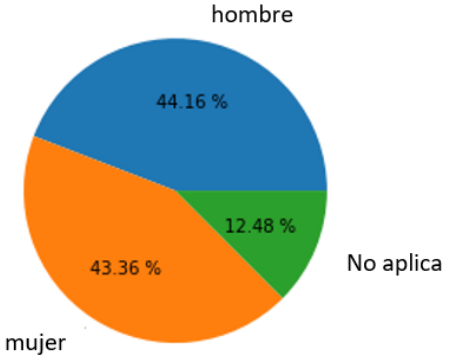
解读： 在 5 次针对独立属性的测试中，多线程 Python 代码与非多线程 Python 代码之间的差异平均为 16%。

结论：

在 9 个属性之间的差异，以及针对单个属性使用线程进行的 5 次测试中，都能观察到明显差异。因此可以得出结论：对于需要优化数据分析的类似情况，使用线程是有效的。

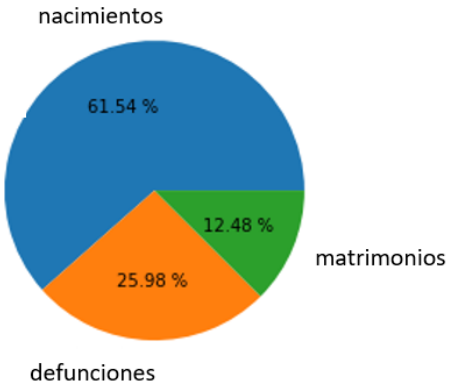
关于数据的观察：

性别图表属性：



解读： 饼图显示，在包含 500,000 条数据的生命事件登记册中，文件持有者中男性占 44.16%，女性占 43.36%，不适用者占 12.48%。由此可见，男性和女性是持有记录最多的人群，而不适用者所占比例很低。

属性图表已完成：



解读：饼图显示，在包含 500,000 条数据的生命事件登记册中，出生占 61.54%，死亡占 25.98%，婚姻占 12.48%。由此可推断，出生数量多于死亡和婚姻数量；图表还显示，婚姻所占比例是死亡的一半。

讨论：

本文提出，在生物学领域中，针对三种语义相似度度量，采用并行处理来缩短大量数据的处理时间。结果表明，处理时间有所缩短，且随着数据量增加，缩短幅度也逐渐增大。 [1]

在家庭环境中进行视觉信息数据处理时，比较哪种语言更适合处理条件十分重要。结果表明，代码在以下方面更有效：

Python 使用线程并行处理图像，因为它接收同一幅图像，但在不同线程上进行处理。 [5]

一般而言，数据处理通常还包括查询、用于处理上下文无关路径的算法，这可能涉及无数符号，并反映执行时间。我们通过 Go 和 Python 测试进行了评估，使用了一个生物学数据库。结果表明，使用线程而非仅使用单个进程进行处理，能够提升性能。 [12]

将数十亿乃至数百万条经纬度位置数据投影到数十万个高度不规则的人口普查区块多边形上，在计算方面颇具挑战性；这可以使用任何脚本语言实现，例如 Python。测试分为简单方法和快速方法。结果表明，快速方法利用线程实现优化，从而快速整合位置数据和人口统计数据。 [9]

如果计划实现并评估两种误差传播度量指标——第一种是结构化数据中的误差传播，第二种是位置损坏比例——则这两种度量指标均可通过软件和硬件进行评估。通过运行单线程和多线程进行了评估，结果表明，使用多线程能够更有效地检测这两种度量指标。 [14]

据观察，通过在更广泛的分析领域中开展开源项目和行业项目，采用并行处理可以优化行程并加快大量数据的处理时间，从而实现潜在的时间缩减。随着数据挖掘的增加，这种效率得到了有益提升。 [6]

为了处理马尔可夫链蒙特卡罗算法的大量数据，开展了相关工作，并观察到

通过实现线程进行并行化处理，可以优化并显著提升处理速度，并在编译时更好地处理读取操作。 [10]

为便于处理这些数据文件，人们创建了大量软件工具，并基于原始 SAMtools 实现开发了一个库，重点关注性能和稳健性。其性能得到了显著提升：BAM 的读取和写入循环速度提高了 5 倍，BAM 到 SAM 的转换速度提高了 13 倍。在处理大量数据时，通过实现线程进行并行化处理，可以观察到令人满意的处理速度；在编译时，这种优化效果最佳。 [4]

尤其是在数据开发的解析算法方面，在处理数据并编译多线程代码时，它们能够提供强大的工具来加快开发进程；另一方面，还可以针对给定脚本专门化解释器。随后，可以将专用代码中的循环进行并行化，以进一步提升性能。 [13]

我们实现了一段代码，旨在利用线程并行处理大量数据，以优化处理时间。凭借强大的计算能力，海量数据推动了计算机科学取得重大进展，但也使得使用现有技术分析和理解结果变得不可能。Neurolytics 的实现主要聚焦于神经科学模拟，旨在帮助科学家快速、高效地整理和理解实验结果。 [15]

参考文献

[1] A.M.Almasoud、H.S. Al-Khalifa 和 A.S. Al-Salman。处理大数据

生物领域中采用并行和分布式处理实现的可扩展性：三种生物语义相似度度量方法的案例。《生物医学研究国际》，2019 年，2019。被引用 2 次。

[2] Alexander David Andrango Ayo. 基于计算机视觉的硬币检测系统实现：简单硬币模式检测系统。学士学位论文，基多：EPN，2022，2022。

[3] Mat´ıas S ´Avalos. 1. 问题求解。

[4]James K Bonfield、John Marshall、Petr Danecek、Heng Li、Valeriu Ohan、Andrew Whitwham、Thomas Keane 和 Robert M Davies。Htslib：用于读取/写入高通量测序数据的 C语言库。*Giga-science*，10(2)：giab007，2021。

[5]A. Bustamante、L.M. Belmonte、R. Morales、A. Pereira 和 A. Fern´andez-Caballero。虚拟无人机视频处理：比较在 Unity 中使用 OpenCV 的两种方法。*AppliedSciences(Switzerland)*，12(12)，2022。引用次数：0。

[6] Fabian Heseding、Willy Scheibel 和 J¨urgen D¨ollner。用于高效利用时间和空间进行 Git 仓库挖掘的工具。*arXiv* 预印本 *arXiv:2205.01351*，2022。

[7]Mario Ib´a˜nez Bolado 等。水产养殖企业决策技术的并行化。2022。

[8]Oriel Keba Manianga 等。基于线程的并行计算应用于谐波分析中使用的指标计算。2021年。

[9]J. Kepner、A. Kipf、D. Engwirda、N. Vembar、M. Jones、L. Milechin、V. Gadepally、C. Hill、T. Kraska、W. Arcand、D. Bestor、W. Berg- eron、C. Byun、M. Hubbell、M. Houle、A. Kirby、A. Klein、J. Mullen、A. Prout、A. Reuther、A. Rosa、S. Samsi、C. Yee 和 P. Michaleas。快速映射至人口普查区块。2020年。被引用2次。

[10]Botao Li、Synge Todo、Anthony C Maggs 和 Werner Krauth。基于局部时间的多线程事件链蒙特卡洛。计算物理通讯，261:107702，2021。

[11]Antonio L´opez Garc´ıa 和 Jaime Urquiza-Fuentes。伪代码与 Java 在职业教育课程和高中教育编程教学中的应用经验。2022。

[12] C.M. Medeiros、M.A. Musicante 和 U.S. Costa。使用上下文无关文法查询图数据库。《计算机语言学报》，68，2022。引用次数：0。

[13]Brandon Neth 和 Michelle Mills Strout。不规则 x86-64 循环的自动并行化。发表于*2019* 年 *IEEE/ACM*代码生成与优化国际研讨会（*CGO*），第 266–266 页。IEEE，2019。

[14] Z. Ozturk、H.R. Topcuoglu 和 M.T. Kandemir。研究误差在应用程序数据结构和硬件上的传播。《超级计算学报》，78(17):18691–18724，2022。引用次数：0。

[15] Judit Planas、Fabien Delalondre 和 Felix Sch¨urmann。利用大数据技术加速神经科学模拟中的数据分析。发表于国际计算科学会议，第 363–377 页。施普林格，2018。