# Real World Concurrency: Programming Real-World Concurrency

Real World Concurrency: Next-Level Computer Multi-tasking.

Author: Raphael M. Tolentino (rm731)

## **Introduction**

In Computer Science, Concurrency refers to multiple computations occurring in concurrency. This is widely present in modern computing as CPU processors keep evolving into more cores, with developers finding innovative methods to optimize their programs. [1]

In this report, I am displaying my research into concurrency and its impact on programming. I will describe its real-life origins and its origins within Computer Science. I will explain the 2 benefits and 2 drawbacks of this paradigm. I will explain other options used by programmers to reach similar goals. I will give my opinion on mobile concurrency. I will also evaluate the future of concurrency, its place in the current/future technological landscape.

**Choice of Concurrency in Discussion**

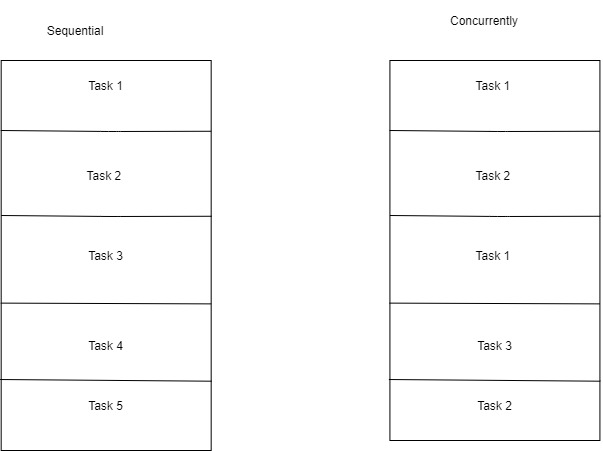
The idea of concurrency mainly derives from 19th railroads engineering as it was needed to use a new system that multiple trains could use the same railroad to reach its destinated station without causing casualties and collateral damage. [2]

 In 1965, Edsger W. Dijkstra published his scientific paper called “Cooperating Sequential Processes”, he provided fundamental concepts used in both modern Operating Systems and Concurrent Programming. Dijkstra discussed concepts like semaphores, deadlocks, and mutual exclusions; fundamental concepts used in modern programming. [3] [4]

I chose to discuss Concurrency and its multiple uses in programming in this report. Researching this paradigm in detail, taught me its fundamentals ideas, its current place in modern programming, and made me have educated guesses about its future.

Diagram 1 is a photograph of Edser W. Dijkstra [5]

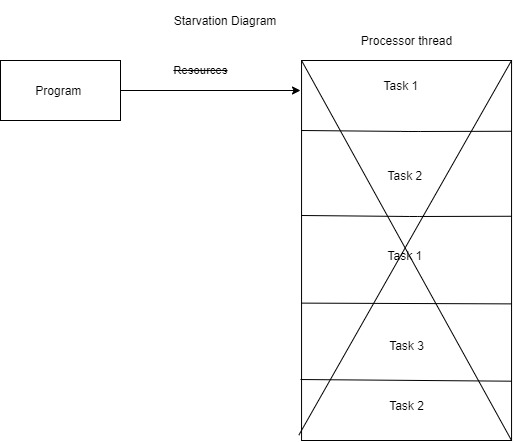
## **Advantages**

A benefit is improved program output. Simple programs execute tasks in sequential order (task-1, task-2, task-3, …), this limits the number of tasks that could be executed in a specific time period. In concurrency, the program executes tasks in arbitrary order with the same outcome (task-1, task-2, task-1, task-3, task-2, …), this increases the number of tasks completed in a specific time period. [6]

Another benefit is processor unwasted time. Programs that either receives user input or examine certain variables usually halt until its done; this slows the program for a certain task. In concurrency; the processor can continue executing program other tasks in the background instead of halting. [6]

Diagram 2 shows sequential and concurrent task order.

**Disadvantages**

A drawback is user overflow. Having multiple users launching/using the same concurrent program (e.g., cloud apps) at the same time, it takes longer to execute the software application. This is because it is dividing all executions into a thread sending output to multiple users. [6]

Another drawback is starvation. In concurrency, program tasks are divided in the CPU processor thread to be executed concurrently. If a program keeps denying resources to a processor thread, it will never start causing the program to execute slower than usually. [7]

Diagram Shows Starvation diagram

**Alternate Computing Paradigms**

The streaming algorithm model is a great alternative to concurrency dealing with big data sets. This model assumes its presented data to the algorithm as one (or more) input streams that processes sequentially, but only once. This model is agnostic about the stream source as it could come from data centers (Amazon web-services) or from data gathering centers (Facebook data centers). Streaming works well with time-related data (status updates, news feeds, etc.) as this is a “batch-oriented” model.

Modern devices work well with a streaming algorithm. Users want the convenience of data access e.g., stream videos, write status, reading the news; without needing to download heavy programs or learning IT jargons. Programmers can include various features e.g., language model, translation model, stream update model; this can reach users devices faster compared to if they were using concurrency program which is present mostly in desktops. [8]

**Mobile Concurrency**

The scenario chosen for this report was Garden concurrency. I described in early sections of this report, that programming in concurrency involves dividing tasks into arbitrary order to improve program output. This also applies to taking care of a garden with multiple produce. You must water the farm goods in arbitrary order to ensure its ready for harvesting.

Diagram Shows a gardener planting a farm good. [9]

In my program, the user checks, plant, water, and harvest 4 produce items (fruit and veggies), this is because in a normal garden you would have multiple plants that need overseeing. It prints the task order and starts executing its tasks (watering and harvesting). After it completed its current tasks it prints a checklist describing the completion of this task. I coded this program with Python and concurrent. futures library, this divided the tasks into threads that completed all tasks under 1 second.

**Evaluation**

It’s has been over 50 years since E. Dijkstra paper in concurrent programming, in which he describes fundamentals for this paradigm. As multicore CPU became widely used in personal computers, new application software using concurrency started being developed and used e.g., videogames. Certain issues of this paradigm became harder to troubleshoot e.g., user overflow forces programs to slow down output if multiple users are trying to execute it simultaneously.

Otherwise, the streaming algorithm is a popular model used to develop and host current widely used application program. In this model, programs are hosted from a Datacentres (Amazon web services) and stream directly to end-users. This benefits developers as they can develop software without certain limitation, and they can troubleshoot/update files and it will be sent to all its users. If the application software is present in multiple devices, users don’t need a specific device to use it, as the stored data can be accessed using multiple devices.

Honestly, currently, concurrent programming will not become obsolete, as developers still use its architecture to develop essential software used in computing e.g., Operating System, video games, etc. However, the streaming model it will become more popular when developing application software as its benefits overshadow its drawbacks.

**Other Relevant Information**

Parallel computing is another paradigm used in Computer Science. This model program tasks are divided into smaller and independent sections that are executed at the same time in multiple CPU threads via sharing memory. The aim of this paradigm is improving computation power for faster program processing and problem solving.

Multi-core is a key feature provided by this paradigm. This feature allows programs to use multiple CPU cores to execute a single program, this allows hardware depended on software to run and solve problems faster.

In Diagram 4, its shows parallel programming. It shows in this paradigm program tasks are divided in smaller sections that are executed in different CPU threads; this allows for faster processing that just executing sequentially in one thread.

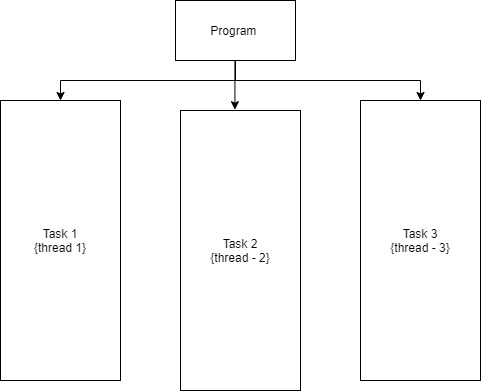


Diagram Show Parallel Programming [10]

**Conclusion**

In this report, I displayed my research into concurrency and its impact on programming. I dwelt its real-life origins with rail engineering and its Computer Science instruction with Edsger W. Dijkstra research paper in 1965. I provided compelling benefits and drawbacks of this paradigm. I explained the current streaming algorithm and its futuristic advantages over concurrency.

I described my case study Garden Concurrency, providing an overall view of its features and its development. I evaluated the current state of concurrency and gave a guess of its future in application and system applications. I discussed parallel programming which is another paradigm used to speed up program execution and problem solving.

It was my intention with this report, display my knowledge about this very interesting topic. During my research, I read multiple articles, book sections, websites and tutorial videos that explained with simple words this paradigm. With the program, I had issues with troubleshooting it as I did not dwelt this far in python, but I am happy with the end result.

**References**

*[1]: web.mit.edu. (n.d.). Reading 17: Concurrency. [online] Available . [Accessed 20 Mar. 2021]*

*[2]: subscription.packtpub.com. (n.d.). {{metadataController.pageTitle}}. [online] Available at:*[*https://subscription.packtpub.com/book/application\_development/9781787285378/1/ch01lvl1sec7/history-of-concurrency*](https://subscription.packtpub.com/book/application_development/9781787285378/1/ch01lvl1sec7/history-of-concurrency)*. [Accessed 20 Mar. 2021]*

*[3]: Lamport, L. (n.d.). Turing Lecture: The Computer Science of Concurrency: The Early Years. [online] cacm.acm.org. Available at:* [*https://cacm.acm.org/magazines/2015/6/187316-turing-lecture-the-computer-science-of-concurrency/fulltext*](https://cacm.acm.org/magazines/2015/6/187316-turing-lecture-the-computer-science-of-concurrency/fulltext) *[Accessed 20 Mar. 2021].*

*[4] : Lamport, L. (2015). Turing lectureThe computer science of concurrency. Communications of the ACM, 58(6), pp.71–76. [Accessed 20 Mar. 2021]*

*[5]: Misa, P.L.F., Thomas J (n.d.). An Interview With Edsger W. Dijkstra | August 2010 | Communications of the ACM. [online] cacm.acm.org. Available at:* [*https://cacm.acm.org/magazines/2010/8/96632-an-interview-with-edsger-w-dijkstra/fulltext?mobile=false*](https://cacm.acm.org/magazines/2010/8/96632-an-interview-with-edsger-w-dijkstra/fulltext?mobile=false) *[Accessed 21 Mar. 2021].*

*[6] www.youtube.com. (n.d.). OCR A’Level Benefits and trade offs of concurrent processing. [online] Available at:* [*https://www.youtube.com/watch?v=Dz2KH-qfI\_c*](https://www.youtube.com/watch?v=Dz2KH-qfI_c) *[Accessed 25 Mar. 2021].*

*[7]: processing?, T. and Newham, J., 2021. Thoughts on definition for disadvantage of concurrent processing?. [online] Computer Science Stack Exchange. Available at:* [*https://cs.stackexchange.com/questions/101917/thoughts-on-definition-for-disadvantage-of-concurrent-processing*](https://cs.stackexchange.com/questions/101917/thoughts-on-definition-for-disadvantage-of-concurrent-processing) *[Accessed 25 March 2021].*

*[8]: Dcs.bbk.ac.uk. n.d. [online] Available at:* [*https://www.dcs.bbk.ac.uk/~Dell/teaching/cc/book/ditp/ditp\_ch8\_2.pdf*](https://www.dcs.bbk.ac.uk/~Dell/teaching/cc/book/ditp/ditp_ch8_2.pdf) *[Accessed 25 March 2021].*

*[9]:* infosquarefootindia (2015). *Taking care of your garden in Monsoon*. [online] squarefootindiablog. Available at: <https://squarefootindiablog.wordpress.com/2015/07/07/taking-care-of-your-garden-in-monsoon/> [Accessed 25 Mar. 2021].

‌*[10] : Kumar, S. (2019). Parallel Programming vs. Concurrent Programming. [online] Medium. Available at:* [*https://medium.com/@sanju.skm/parallel-programming-vs-concurrent-programming-f993d3f9ceea*](https://medium.com/@sanju.skm/parallel-programming-vs-concurrent-programming-f993d3f9ceea)*.* [Accessed 25 Mar. 2021].

*‌***Source Code**

import concurrent.futures # This will allow threading in the program

import time # This allows time features in the program

start = time.perf\_counter() # this is an global variable that starts the counter

def garden\_welcome(seconds): #this is the welcoming message

    print('----------------Welcome to your Garden------------------- \n') #Welcomes the User to the Garden

    time.sleep(seconds) #This counts the total time in the treads

    return '\nCheckbox:  Welcome Message Completed.'

def garden\_plants(seconds): #This describes the activities

`

    print(' -Today we are going to check current farm goods'),

    print(' ----------------------------------------------- '),

    print(' -Today we are going to plant new farm goods      ')

    print(' ------------------------------------------------')

    print(' -Today we are going to Water the new Plants ')

    print(' ------------------------------------------------')

    print(' -Today we are going to Harvest the grown plants')

    print(' ------------------------------------------------')

    print('We Currently Have:')

    print(f'Fruits    : Lemons, Grapes:')

    print(f'Veggies : Onions, Potatoes:')

    print(' ------------------------------------------------')

    print('Time to water the Plants\n')

    print('\nWatering\n')

    print('\nWatering Plants\n')

    print(' ------------------------------------------------')

    print('\nTime to Harvest the Plants\n')

    print('\nHarvesting\n')

    print('\nPlants Harversted\n')

    print('Checkbox : Harvesting step Completed')

    print('Checkbox : Water step Completed')

    time.sleep(seconds) #This counts the total time in the treads

    return 'Checkbox : Show step Completed'

with concurrent.futures.ThreadPoolExecutor() as executor: # This loads the threads, and links to a variable called executor

    f1 = executor.submit(garden\_welcome, 1) # This is the first thread

    f2 = executor.submit(garden\_plants, 1)    # This is the second thread

    print(f1.result()) # This loads the first thread and its content

    print(f2.result()) # This loads the second thread and its content

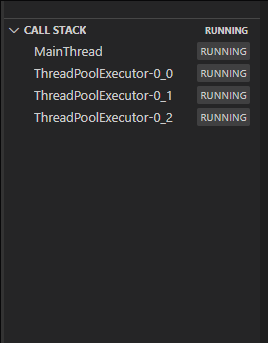
finish = time.perf\_counter() # this is an global variable that finishes the counter

print(f'Your Visited the Garden for{round(finish-start, 2)} seconds(s)') # This prints the total time needed to perform this program

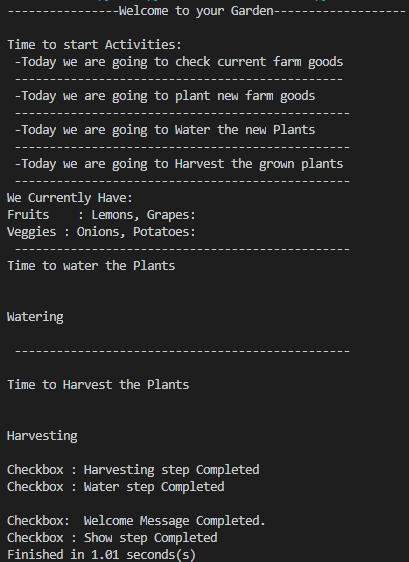
###www.youtube.com. (n.d.). Python Multiprocessing Tutorial: Run Code in Parallel Using the Multiprocessing Module.

##[online] Available at: https://www.youtube.com/watch?v=fKl2JW\_qrso&t=11s [Accessed 29 Mar. 2021].

**Screenshots**

****

Screenshot 1 shows the program running with multiple cores (Visual Studio Code IDE)



Screenshot 2 Shows the Outcome of my Concurrent Program in Python (Taken from Visual Studio code IDE)