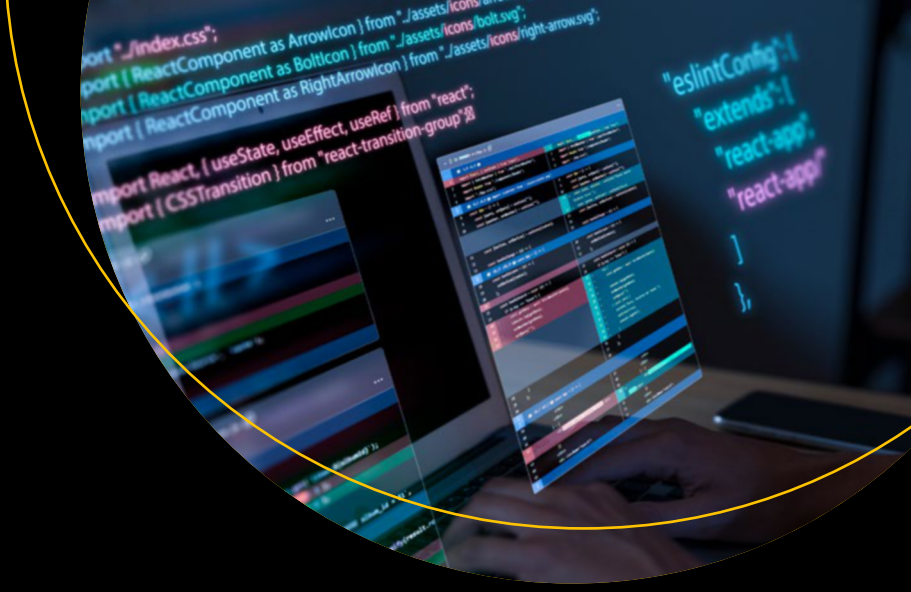


Parallel Programming with C# and .NET

Fundamentals of Concurrency and Asynchrony Behind Fast-Paced Applications

Vaskaran Sarcar

Foreword by Naga Santhosh Reddy Vootukuri

Apress®

Parallel Programming with C# and .NET

Fundamentals of Concurrency and Asynchrony Behind Fast-Paced Applications

Vaskaran Sarcar

Foreword by Naga Santhosh Reddy Vootukuri

Apress®

Parallel Programming with C# and .NET: Fundamentals of Concurrency and Asynchrony Behind Fast-Paced Applications

Vaskaran Sarcar
Kolkata, West Bengal, India

ISBN-13 (pbk): 979-8-8688-0487-8 ISBN-13 (electronic): 979-8-8688-0488-5
https://doi.org/10.1007/979-8-8688-0488-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Laura Berendson
Coordinating Editor: Kripa Joseph

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (https://github.com/Apress/Parallel-Programming-with-CSharp-and-.NET). For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper

To those who are often labeled as “workaholics,” “introverted,” or “antisocial,” only because they forget the world while coding.

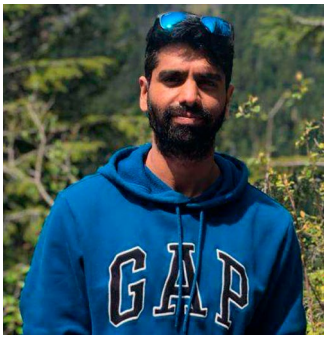
Table of Contents

About the Author

Vaskaran Sarcar obtained his master's of engineering from Jadavpur University, Kolkata (India), and his master's of computer applications from Vidyasagar University, Midnapore (India). He was a National Gate Scholar (2007–2009) and has more than 12 years of experience in education and the IT industry. He devoted his early years (2005–2007) to the teaching profession at various engineering colleges, and later he joined HP India PPS R&D Hub in Bangalore. He worked there for more than 10 years and became a senior software engineer and team lead. After that, he decided to follow his passion and is now an independent full-time author.

You can find his books at https://amazon.com/author/vaskaran_sarcar and find him on LinkedIn at www.linkedin.com/in/vaskaransarcar.

About the Technical Reviewer



Naga Santhosh Reddy Vootukuri is a senior software engineering manager at Microsoft, specializing in cloud computing and artificial intelligence. With a distinguished career spanning 16 years across India, China, and the United States, Naga has amassed a wealth of experience in distributed systems, microservices, and large-scale infrastructure management building cloud and intelligent systems.

At Microsoft, Naga leads the Azure SQL Database team, driving initiatives to optimize SQL deployment processes for millions of databases worldwide. His leadership was pivotal in the development of Master Data Services (MDS), a critical component of Microsoft's SQL enterprise solutions, which saw a significant increase in customer adoption under his stewardship.

Naga has authored and published numerous research articles in peer-reviewed and indexed journals. He is a senior member of IEEE, teaches workshops on AI, and writes technical articles as a Core MVB member at DZone, engaging with millions of active readers. You can read his articles at `https://dzone.com/authors/sunnynagavo`. He also serves as an editorial board member for a highly reputed science journal (SCI), where he reviews research articles on cloud computing and AI, further solidifying his influence in the academic and professional spheres.

Beyond his corporate roles, Naga actively contributes to the tech community by speaking at events, reviewing books for Apress, and actively helping people in forums like the Microsoft Tech Community. His commitment to advancing technology is underscored by his recent role as a judge for the Globee Awards, where he evaluated and recognized innovations in the industry. He also served as a judge for the Microsoft AI hackathon, with more than 10,000 developers participating.

Throughout his career, Naga has demonstrated exceptional leadership and technical prowess, evident in his mentorship on ADP List and his key role in conducting more than 100+ interviews to recruit top talent for Microsoft. You can contact him at https://adplist.org/mentors/sunny.

In summary, Naga embodies a blend of technical expertise, leadership excellence, and community engagement, making profound contributions to both Microsoft and the broader tech industry. His dedication to innovation and mentorship continues to inspire professionals and shape the future of technology globally.

Acknowledgments

First, I thank the Almighty. I sincerely believe that with His blessings only I could complete this book. I also extend my deepest gratitude and thanks to the following:

- **Naga Santhosh Reddy Vootukuri:** He is the technical reviewer of this book, located in a different country, and works in a different time zone. Despite these factors, whenever I was in need, he provided support. He answered all my queries through phone calls, WhatsApp, and emails. Thank you one more time.
- **Smriti, Laura, Celestin, and the Apress team:** I sincerely thank each of you for giving me another opportunity to work with you and with Apress.
- **Nirmal, Kim Wimpsett, Selvakumar and Pushparaj:** Thanks to each of you for your exceptional support in beautifying my work. Your efforts are extraordinary.

Finally, I thank those people from the online C# developer community, .NET developer community, and Stack Overflow community who have shared their knowledge in various forms. In fact, I thank everyone who directly or indirectly contributed to this work.

Foreword

Mastering a programming language is much like an artist perfecting their craft; it demands dedication, practice, and the right tools. In the realm of modern computing, where efficiency and speed are paramount, parallel programming stands as a cornerstone. In fact, parallel programming is like conducting an orchestra—each thread must play its part perfectly, or you end up with chaos instead of harmony. This book, *Parallel Programming with C# and .NET*, is an indispensable guide for anyone looking to harness the full power of parallel computing, ensuring their applications are both fast and scalable.

Vaskaran Sarcar, an esteemed author, is known for his ability to explain complex ideas clearly. He is passionate, well-informed, skilled, and very knowledgeable in this area. His systematic approach and curiosity have earned him respect in the programming community. I've seen Vaskaran tackle tough problems with focused determination, consistently finding elegant solutions that showcase his analytical skills and dedication.

Parallel Programming with C# and .NET focuses sharply on the complex world of parallel computing. Each chapter breaks down different techniques step-by-step, starting with basic concepts explained in a simple manner. It includes practical sample programs and quizzes to reinforce your understanding. This structured approach simplifies learning and ensures you can apply these concepts effectively. This pedagogical style ensures that readers not only grasp fundamental concepts but also retain them effectively.

In today's tech landscape, performance and efficiency are crucial. C# has grown beyond its Windows origins, now used in diverse environments like Mac and Linux, thanks to Microsoft's open-source culture. This book provides advanced knowledge to help developers fully utilize C#'s capabilities, crafting efficient solutions optimized for various platforms.

I am confident that *Parallel Programming with C# and .NET* will accelerate many developers' learning so they become proficient in parallel computing in record time. Vaskaran's exemplary work establishes a solid foundation for understanding and utilizing advanced language features, paving the way for creating robust software.

Developers will gain clarity on complex concepts and rely on this book as their essential resource for mastering parallel computing challenges. Happy coding!

Naga Santhosh Reddy Vootukuri
Senior Software Engineering Manager
Microsoft, Azure SQL Server
(Cloud + AI division)

Introduction

Modern-day software is highly responsive and scalable. As a result, support for parallel computation is an essential part of it. Undoubtedly this is an advanced concept, and the solutions are not straightforward. Many developers have been burned (and are still burning) by them.

In addition, parallel programming is a vast topic that requires many different considerations. You may also note that many patterns used in the past to deal with asynchronous and parallel programming are not recommended now. This book tries to simplify the concept using modern C# features and libraries that Microsoft recommends. Welcome to your journey through *Parallel Programming with C# and .NET: Fundamentals of the Concurrency and Asynchrony Behind Fast-Paced Applications*.

C# is a powerful programming language, well-accepted in the programming world, that can help you make a wide range of applications. Throughout C# development, supportive features and libraries have been developed to support parallel programming. This is one of the primary reasons that it is continuously growing and always in high demand. So, it is not a surprise that existing and upcoming developers (for example, college students and programming lovers) want to use C# for parallel programming.

Many developers try to learn parallel programming in the shortest possible time frame. Trying to learn something as quickly as possible is laudable, but do you know the problem? We are living in a world that offers you lots of materials, advertisements, and quick fixes to capture your attention. We human beings love to daydream. So, they take advantage of this behavior and start claiming that you can learn everything in a day, a week, or a month. Is this true? Ask yourself and you'll get the answer.

Malcolm Gladwell in his book *Outliers* (Little, Brown, and Company) talked about the 10,000-hour rule. This rule says that the key to achieving world-class expertise in any skill is, to a large extent, a matter of practicing the correct way for a total of about 10,000 hours. So, you can probably see that even though we may claim that we know something very well, we actually probably know very little. Learning is a continuous process, with no end.

Then should we stop learning? Definitely, the answer is no. What should we do then? We can follow an *effective learning* process that teaches you how to learn quickly to serve the need. This is where I like to remind you about the Pareto principle, or 80-20 rule. This rule simply states that 80% of outcomes come from 20% of all causes. This is useful in programming too. When you truly learn the fundamental and most important aspects of parallel programming, that is when you can use it effectively to improve your code. Most importantly, your confidence level will rise, and you won't be afraid to experiment more. This book is for those who acknowledge these facts.

How Is the Book Organized?

This book helps you to understand the core principles of parallel programming using six chapters with plenty of supportive materials. To give you an idea how each chapter is organized, the following list talks about the contents of the book:

- Chapter 1 starts with an overview of the Task Parallel Library (TPL) and discusses tasks. These topics are the foundation for the upcoming chapters. Chapter 2 discusses special scenarios such as handling exceptions and cancellations. Chapter 3 discusses synchronization techniques and concurrent collections. Chapter 4 discusses the `Parallel` class, which helps you experiment with parallel loops to speed up

computations. Chapter 5 discusses Parallel LINQ (PLINQ). Chapter 6 discusses simplifying asynchronous programming using the `async` and `await` keywords. Finally, the appendix provides you with some extra materials that are not discussed in the previous chapters.

- I have always enjoyed learning when analyzing case studies, asking questions, and performing exercises. So, throughout this book, you will see interesting program code, "Q&A Sessions," and exercises. By analyzing these Q&As and doing the exercises, you can verify your progress. As mentioned, these are presented to make your learning easier and more enjoyable, but most importantly, they will make you confident as a developer.
- Each question in these "Q&A Sessions" is marked with **Q<chapter#>.<question#>**. For example, Q5.3 means question 3 from Chapter 5. At the end of the chapter, you'll see some exercises. You can use them to evaluate your progress. Each question in these exercises is marked with **E<chapter#>.<question#>**. For example, E6.2 means exercise 2 from Chapter 6.
- You can download all the source code for the book from the publisher's website.

Prerequisite Knowledge

I expect you to be very familiar with C#. In fact, knowing about some of the advanced concepts like delegates and lambda expressions can accelerate your learning. So, I assume that you know how to compile or run a C# application in Visual Studio. This book does not invest time in basic

topics, such as how to install Visual Studio on your system, how to write a "Hello World" program in C#, and so forth. In short, the target readers of this book are those who want to make the most of C# by harnessing the power of both object-oriented programming (OOP) and functional programming (FP).

Who This Book Is For

You will get the most from this book if you can answer "yes" to the following questions:

- Are you familiar with .NET, C#, and basic object-oriented concepts such as polymorphism, inheritance, abstraction, and encapsulation?
- Are you familiar with some of the advanced concepts in C# such as delegates, lambda expressions, and generics?
- Do you know how to set up your coding environment?
- Are you interested in knowing how the modern-day constructs of C# can help you in parallel programming?

You will probably struggle with this book if you can answer "yes" to any of the following questions:

- Are you looking for a C# tutorial or reference book?
- Are you not ready to experiment with parallel programming using a programming language other than C#?
- Do you dislike Windows, Visual Studio, and/or .NET or want to learn parallel programming without them?

Useful Software

These are the important tools that I used for this book:

- All the programs were tested with C# 12 and .NET 8. In this context, it is useful to know that nowadays the C# language version is automatically selected based on your project's target framework so that you can always get the highest compatible version by default. In the latest versions, Visual Studio doesn't support the UI to change the value, but you can change it by editing the `.csproj` file.
- As per the new rule, C# 12 is supported only on .NET 8 and newer versions. C# 11 is supported only on .NET 7 and newer versions. C# 10 is supported only on .NET 6 and newer versions. If you are interested in learning more about the C# language versioning, you can refer to https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/configure-language-version.
- During the development of this book, software updates kept coming and I also kept updating. When I finished my initial draft, I had the latest edition of Visual Studio Community 2022 (64-bit, version 17.8.5).
- The good news for you is that this community edition is free. If you do not use the Windows operating system, you can use Visual Studio Code, which is also a source-code editor developed by Microsoft that runs on Windows, macOS, and Linux operating systems.

> This multiplatform integrated development environment (IDE) is also free. However, I recommend that you check the license and privacy statement as well because this statement may change in the future.

Author's Note: I have tested my code only on Visual Studio. At the time of this writing, Visual Studio 2022 for Mac is scheduled for retirement on August 31, 2024. To learn more about this, you can refer to https://learn.microsoft.com/en-us/visualstudio/mac/what-happened-to-vs-for-mac?view=vsmac-2022.

Guidelines for Using This Book

Here are some suggestions so that you can get the most out of this book:

- This book suits you best if you are familiar with some advanced features in C# such as delegates and lambda expressions. If not, please read about those topics before you start reading this book.
- I organized the chapters in a manner that can help you understand parallel programming. For example, if you directly start using the `async` and `await` keywords in your programs, you may miss many other concepts that actually caused them to appear in C# in a later version. This is why I believe you should read the chapters sequentially. Another reason for this suggestion is that some useful topics may have already been discussed in a previous chapter, and I have not repeated those discussions in the later chapters.
- The programs in this book should give you the expected output in the upcoming versions of C#/Visual Studio. Though I believe that these results should not vary in

other environments, you know the nature of software: it is naughty. So, I recommend that if you want to see the same output, you should mimic the same environment that I am using.

- You can download and install the Visual Studio IDE from `https://visualstudio.microsoft.com/downloads/`. You are expected to get the screen shown in Figure FM-1.

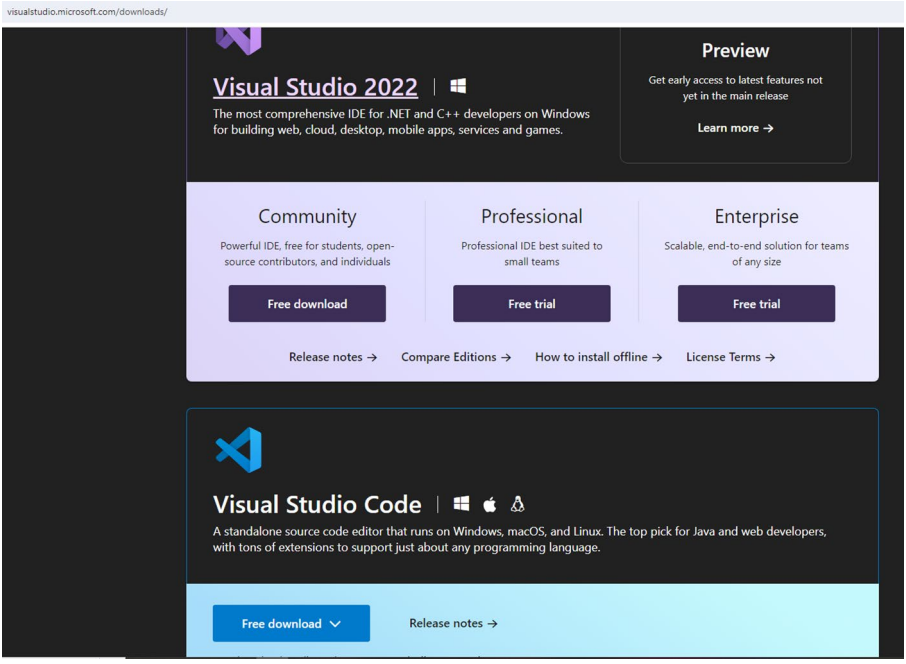


Figure FM-1. *Download screen for Visual Studio 2022 and Visual Studio Code*

Note At the time of this writing, this link works fine, and the information is correct. However, the link and policies may change in the future. The same comment applies to all the mentioned links in this book.

Conventions Used in This Book

In many places, I give you links to Microsoft's documentation. Why? For me, the creators of the products are the authenticated source of information to describe a feature.

In addition, to draw your attention to some places in the code, I have made them bold. For example, consider the following code fragment (taken from Chapter 1 where I discuss continuation tasks and want to draw your attention to a particular line):

```
// Using C#12's "Collection expression" feature
var task3 = Task.Factory.ContinueWhenAll(
  [task1, task2],
  tasks =>
  {
    WriteLine("Arranging dinner.");
  }
 );
```

Final Words

You are an intelligent person. You have chosen a topic that can assist you throughout your career. As you learn and review these concepts, I suggest you write your own code; only then will you master this area. There is no shortcut for this.

Do you know the ancient story of Euclid and Ptolemy, ruler of Egypt? Euclid's approach to mathematics was based on logical reasoning and rigorous proofs, and Ptolemy asked Euclid if there was an easier way to learn mathematics. Euclid's reply to the ruler? "There is no royal road to geometry."

Though you are not studying geometry, the essence of this reply applies here. You must study these concepts and code. Do not give up when you face challenges. They are the indicators that you are growing better.

Errata: I have tried my best to ensure the accuracy of the content. However, mistakes can happen. So, I plan to maintain "errata," and if required, I will make some updates/announcements there. You can visit the online link: https://github.com/Apress/Parallel-Programming-with-CSharp-and-.NET to receive any important corrections or updates.

An appeal: You probably understand that good-quality work takes many days and months (even years!) to complete. Many authors like me invest most of their time in writing and heavily depend on it. You can encourage and help these authors by preventing piracy. If you come across any illegal copies of our works in any form on the Internet, I would be grateful if you would provide me or the Apress team with the location using the following link: https://www.apress.com/gp/services/rights-permission/piracy.

Share your feedback: I believe that this book is designed for you in such a way that you will develop a keen knowledge of parallel programming using C# and .NET. If you value the effort, I request you provide your valuable feedback on the Amazon review page or any other platform you like.

CHAPTER 1

Understanding Tasks

Modern-day life is full of work. Consider those working mothers who need to complete a lot of housework before they go to the office. You may see them performing all these activities asynchronously. For example, they start preparing breakfast, and they come back from the kitchen to prepare the kids for school. Then they go back to the kitchen to check the status of the food only to leave the kitchen to start preparing themselves for the office. You can see that with this approach, when they start a task, they do not wait for that task to complete; instead, they start the next task and then go back to check the status of the previous task. This process continues until all these tasks are finished. Psychologists who vote for doing one task at a time may not like this approach, but at the same time, we cannot ignore that this is how people do things. The programming world tries to mimic real-world scenarios. Throughout this book, we will explore those possibilities using C# and .NET.

Stepping into Parallel Development

You are probably familiar with synchronous method calls where you see that the caller method is blocked until the callee method finishes its execution. To illustrate, consider the following code segment:

```
// Some code before
int temp = RetrieveSomeValue(); // Line 1
int result = Process(temp);// Line 2
// Some code after
```


V. Sarcar, *Parallel Programming with C# and .NET*,
https://doi.org/10.1007/979-8-8688-0488-5_1

If you exercise this code in a single-threaded environment, you cannot execute line 2 before you get the `temp` value in line 1. Why? Calling the `RetrieveSomeValue` method blocks the thread until the method finishes its execution. We are probably all familiar with this type of coding. In fact, we often exercise typical .NET calls that are blocking.

However, it is not a good idea to always rely on the blocking calls. For example, consider the situation when you trigger a method that attempts to download a large file. While the download is in progress, if you cannot do any other work, that is a problem for sure. What is the solution? You guessed it: you will opt for a multithreading environment and opt for asynchronous operations.

You may also notice that computing machines are becoming faster by the day. The role of multicore CPUs is inevitable. So, regardless of the programming languages, developers will want to take advantage of them. This is why parallel programming is becoming an essential technique for developing powerful software that is highly responsive.

Parallel programming is interesting as well as challenging. Undoubtedly it is hard, but in the past it was harder. In those days, developers had the following options:

- Creating and using threads directly
- Using the `ThreadPool` class
- Using event-based asynchrony
- Using the `AsyncResult` pattern
- Using the callback methods

These are not the recommended approaches now. I'd like to start the discussion with tasks that make up the foundation of the Task Parallel Library (TPL). Why? Once .NET Framework 4 was released, Microsoft stated the following (see https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming):

TPL is the preferred API for writing multithreaded, asynchronous, and parallel code in .NET

Note It is also interesting to know that behind the scenes, tasks are queued to the `ThreadPool` class, which determines and adjusts the number of threads. The `ThreadPool` class follows some algorithms to control the load balancing to gain maximum throughput.

Introduction to the Task Parallel Library

Why is the TPL important? As mentioned, handling concurrent programs and adding parallelism to an application are challenging tasks. Undoubtedly, these are advanced concepts of programming. The TPL aims to make them easy for you by providing a set of public types and APIs. You can find them in the following namespaces:

- `System.Threading`
- `System.Threading.Tasks`

Author's Note: Developers often refer to the `Parallel` class with the task parallelism constructs as the TPL. You'll learn about the `Parallel` class in Chapter 4. However, at this stage, you do not need to worry about them. You'll learn about these constructs one at a time.

How Does the TPL Help?

Microsoft's documentation states the following (see https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl):

> *The purpose of the TPL is to make developers more productive by simplifying the process of adding parallelism and concurrency to applications.*

These are some of the useful scenarios for the TPL:

- Managing a multithreaded environment efficiently
- Scaling the concurrency level
- Providing support for task cancellations
- Managing state
- Partitioning your work

You can easily guess that we are going to cover all these scenarios. For now, it will be sufficient for you to understand that the TPL simplifies the multithreading scenarios and helps you write high-performance code without worrying about the nitty-gritty of threading or other low-level details.

The Concept of Tasks

What are tasks? Microsoft states the following (see https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming):

> *The Task Parallel Library (TPL) is based on the concept of a task, which represents an asynchronous operation. In some ways, a task resembles a thread or ThreadPool work item but at a higher level of abstraction. The term task parallelism refers to one or more independent tasks running concurrently.*

A *task* is a code block that represents a unit of work. You use tasks to inform the scheduler that this code block can execute on a separate thread while the main thread of execution continues. For example, consider the following code segment:

```
static void PrintNumbers()
{
    WriteLine("Starts executing the task1.");
    // Doing Some work
    Thread.Sleep(5);
    WriteLine("The task1 is finished.");
}
```

This can be a unit of work, and we can execute it on a separate thread. Some common examples of tasks include the following:

- Perform a calculation and display the result.
- Compute a value with/without a supplied input.
- Ask for a network resource.
- Check the health of a website, for example, pinging a website.

Creating and Executing a Task

You can create and execute tasks in different ways. Let me show you a sample code segment that creates and starts executing the task:

```
Task task1=new Task(PrintNumbers);
task1.Start();
```