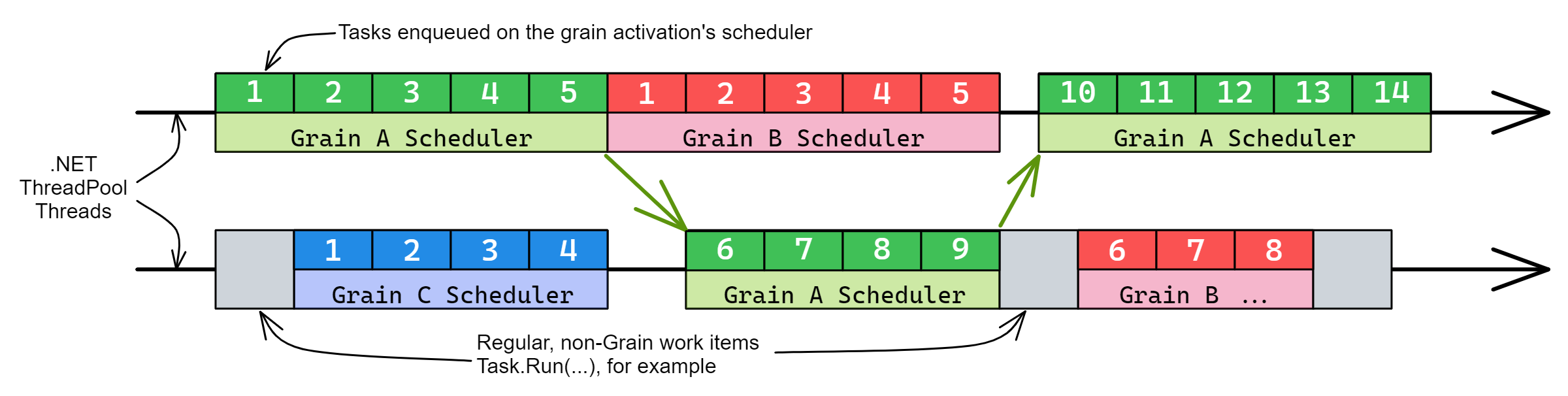
# Orleans Grain的 重新進入(Reentrant) 功能介紹與 死結(Deadlock)問題解決

## 重新進入(Reentrant)功能

在Orleans中，Grain的RPC方法預設是不允許同時被多重呼叫的，因為Grain的RPC方法是以Actor模型的概念設計成的機制，一個Grain的實例同時只能執行一個RPC方法呼叫，因此而避免有同時被多個來源呼叫而同時去改變/共用內部變數狀態等的情況發生。

實際上的實作，Orleans框架底層有自製 .NET ThreadPool的『[工作排程器(Task Scheduler)](https://github.com/dotnet/orleans/blob/3.x/src/Orleans.Runtime/Scheduler/ActivationTaskScheduler.cs)』，和本來CLR 的標準 [TaskScheduler](https://learn.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskscheduler) 排程器不同，負責調度要執行哪個Grain的RPC方法：

 *圖片來源：*[*官方說明網頁*](https://learn.microsoft.com/en-us/dotnet/orleans/implementation/scheduler)

這每個Grain都有的自製工作排程器，就是控制Grain的同時刻只有單個非同步RPC方法被執行，執行完才能跑下一個的Actor Model機制，此拘束機制可以藉啟用Grain的重新進入(Reentrant)功能來解除，這個功能的啟用設定方法有兩種：固定啟用和動態啟用。

* 固定啟用：又分為在單一方法層級或是Grain層級。
  + 單一方法層級：在Grain的RPC方法定義介面上加入 [[AlwaysInterleave]](https://learn.microsoft.com/dotnet/api/orleans.concurrency.alwaysinterleaveattribute) 屬性標註，就可以讓該方法被多重呼叫。
  + Grain層級：在Grain的介面定義上加入 [[Reentrant]](https://learn.microsoft.com/dotnet/api/orleans.reentrancy.reentrantattribute) 屬性標註，就可以讓該Grain的所有方法都可以被多重呼叫。
* 動態啟用：在RPC方法上標註 [[MayInterleave]](https://learn.microsoft.com/en-us/dotnet/api/orleans.concurrency.mayinterleaveattribute)，並實作相關的判斷函式，這種方式可以讓每次呼叫的RPC方法前Grain都可以自行判斷是否要啟用重新進入(Reentrant)功能。

重新進入(Reentrant)功能在提升一些不改變狀態(資料唯讀)的執行動作上啟用，可以有效提升效能，例如一些類似Ping或IsHealthy的RPC方法，但是在一些需要改變狀態的RPC方法上啟用，就會造成Grain的狀態改變行為由於多方法同時呼叫而結果不一致的問題，因此在使用時要注意。

另一個用途是解決Grain對Grain呼叫之間可能造成的死結問題， 以下討論Grain RPC方法的死結(Deadlock)問題與如何解決。

## Grain RPC方法死結問題與避免

當Grain的RPC方法中有呼叫其他Grain的RPC方法時，就會發生死結(Deadlock)的問題，因為呼叫其他Grain的RPC方法時，會等待其他Grain的RPC方法回應，而其他Grain的RPC方法又會等待呼叫它的Grain的RPC方法回應，這樣就會造成死結(Deadlock)的問題。

例如Grain A呼叫Grain B，Grain B又呼叫Grain A，這種情況下，Grain A和Grain B都會等待對方的回應，造成死結，這時候就可以在Grain A和Grain B上啟用重新進入(Reentrant)功能，讓Grain A和Grain B可以同時執行，避免死結。

有些複雜的狀況甚至是Grain A呼叫Grain B，Grain B又呼叫Grain C，Grain C又呼叫Grain A的迴圈(Loop)情形，這種情況下，Grain A、Grain B、Grain C都會等待對方的回應，造成死結：