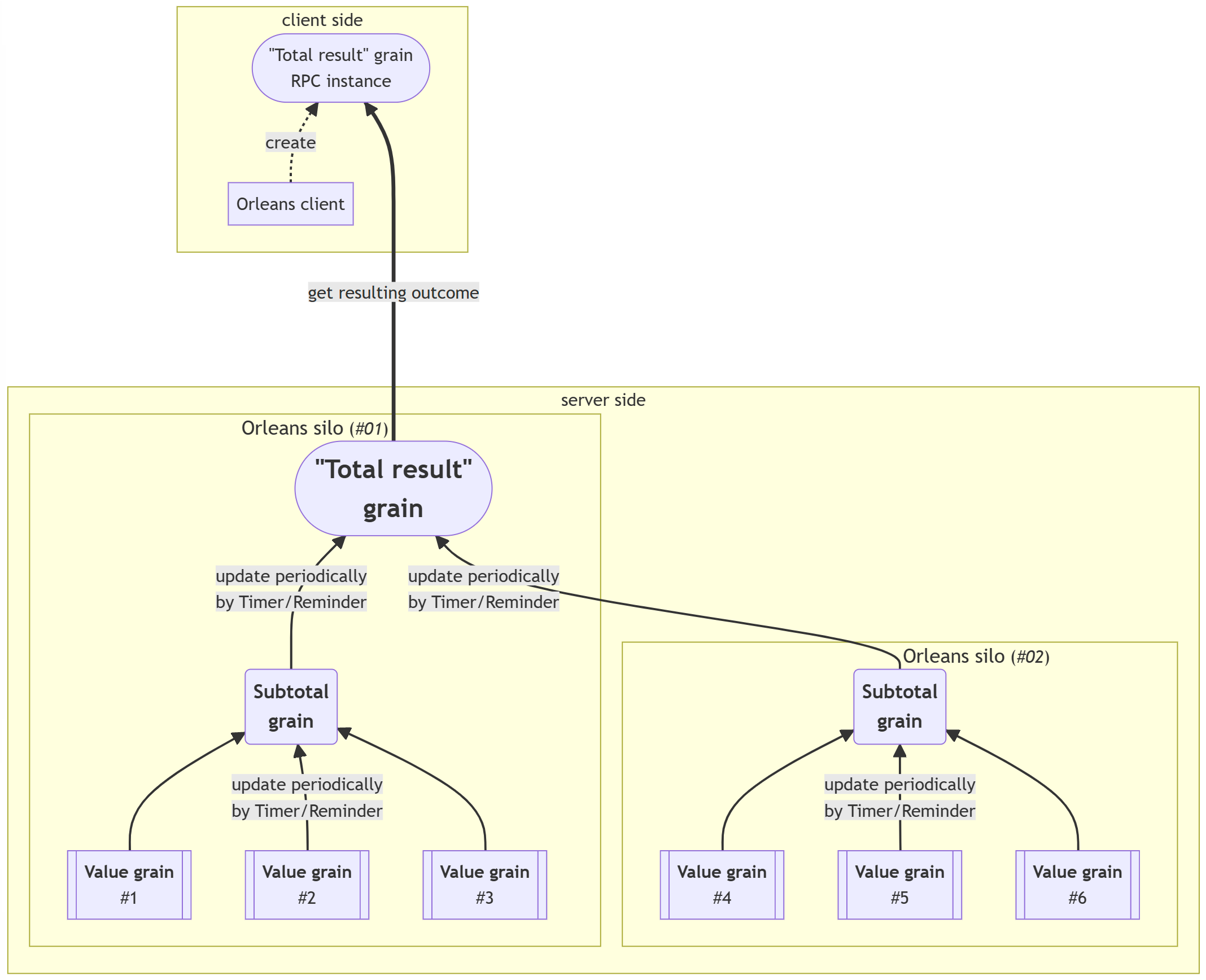
# Orleans常見系統架構模式：Reduce Pattern及其應用範例

## Reduce Pattern 介紹

[Reduce Pattern](https://github.com/OrleansContrib/DesignPatterns/blob/master/Reduce.md) 類似 Map-Reduce 演算法的概念，將每一個運算單元的運算結果回報給其上層的彙整單元，彙整單元可以是階層式的連結關係，以便再進一步有更上層的彙整單元將資料整個收集彙整累加，最後讓Client端在呼叫時，直接跟最高階層的彙整單元溝通，如此便可以一次獲得原本在個別運算單元的運算結果快取，而不需要等待：



如上圖，Value grain是一些分散的商業邏輯業務執行單元grain，如果需要得知這些grain的相關統計數據或是grain和grain彼此之間如排名比較的關係資料，假如一開始的實作方式沒有套用Reduce Pattern，則必須至少遍歷每個Value Grain進行一次RPC呼叫以便拉出資料，然後自己彙整分析等資料處理運算，運算時間和網路輸入輸出等資源成本很高；但套用Reduce Pattern這種分而治之的架構後，每個Value grain會定期將其資料上傳給『彙整單元』（Subtotal grain），然後Subtotal grain也會再度定時上傳資料到更上一階層的總計彙整單元（“Total result” grain）。所以當Client端需要這些資料時，就不必自己去一一蒐集資料來計算，直接跟最上層架構的總計彙整單元溝通就能獲得彙整後的結果。

Reduce Pattern和昨天介紹的Registry Pattern在架構上有點類似，但不同的是Registry Pattern還會跟終端的節點grain溝通，而Reduce Pattern則是跟終端節點grain溝通，或是跟每一個需求相關的grain互動太耗費資源了，所以先將當時的部分計算結果每隔一段時就上傳給上層的彙整節點grain將暫態資料先暫存起來，而更上層的就能拿這些暫態資料計算出當時的結果；最後Client端需要的時候，最上層的彙整grain就能馬上直接輸出之前累積的計算結果。

Reduce Pattern的優點是發揮Orleans提供的分散式架構功能，還可如前面的示意圖所繪製的，當太多底層grain需要包含時，移轉到另一個Silo叢集來做負載分散。

Reduce Pattern的缺點是，由於需要底層定期上傳當時暫態結果的架構，所以可能當Client端需要資料而進行查詢時，撈取到的資料並不會是當下實際上正確的版本，算是一種『最終一致性(eventually consistent)』的系統。

Reduce Pattern常用在一些類似統計數字的需求用途，例如游戲系統的玩家排行榜，以下就以一個排行榜服務的實作範例來示範：

## Reduce Pattern範例