Multi-Thread Programming
RCL.ParallelExtensions (in RCL.Core)

2011.03

Realweb

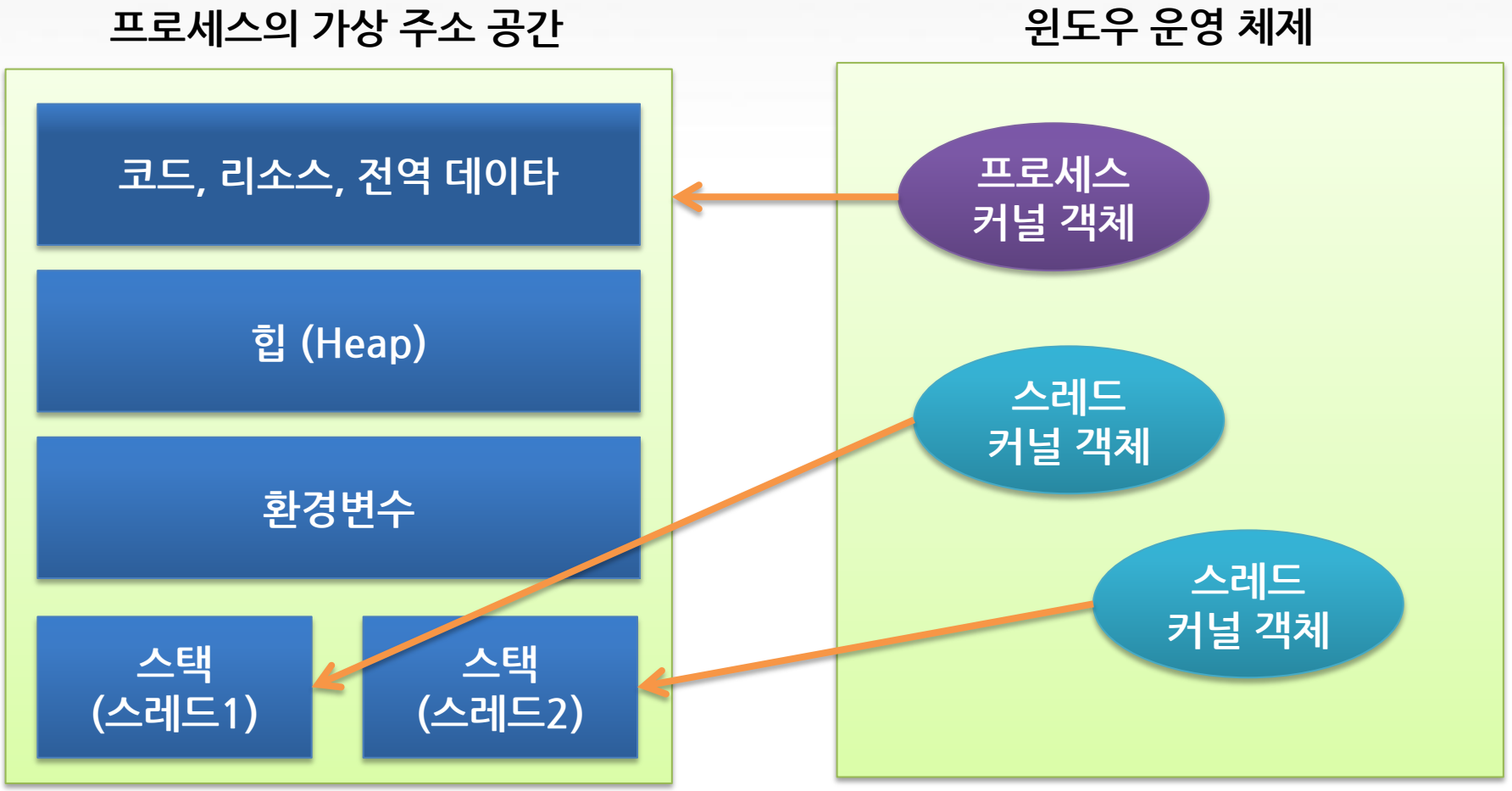
목차

Multi-Thread Overview

- Multi-Tasking vs. Multi-Threading
  - Multi-Tasking : 하나의 CPU가 여러 개의 프로세스를 교대로 수행
  - Multi-Threading : 하나의 CPU가 여러 개의 Thread를 교대로 수행

- Multi-Threading 의 중요성
  - 프로세스 간의 통신은 속도 문제가 심함.➔Thread 간의 통신은 빠름
  - 한 프로세스로 작업하는 것은 사용자 응답성에 문제가 많음.
  - 프로세스보다 Thread 가 비용이 적게 듦
  - 프로세스 내에 Multi-Thread를 사용하므로서 성능을 향상 시킬 수 있지만, 일반적인 안정성은 떯어짐.

■ Process 와 Thread 의 구성요소



CPU Scheduling

- CPU Scheduling 이란
  - 한정된 CPU의 작업 처리 시간을 여러 프로세스 혹은 스레드가 공동으로 이용할 수 있도록 분배하는 정책

- Microsoft Windows의 CPU Scheduling
  - 선점형 스케쥴링
  - 우선순위 (Priority)에 기반한 CPU 스케쥴링 기법을 사용
  - OS에서 Priority를 기반으로 Thread Context Switching 수행

- Windows 에서 우선순위란
  - Process 우선 순위
  - Thread 우선 순위

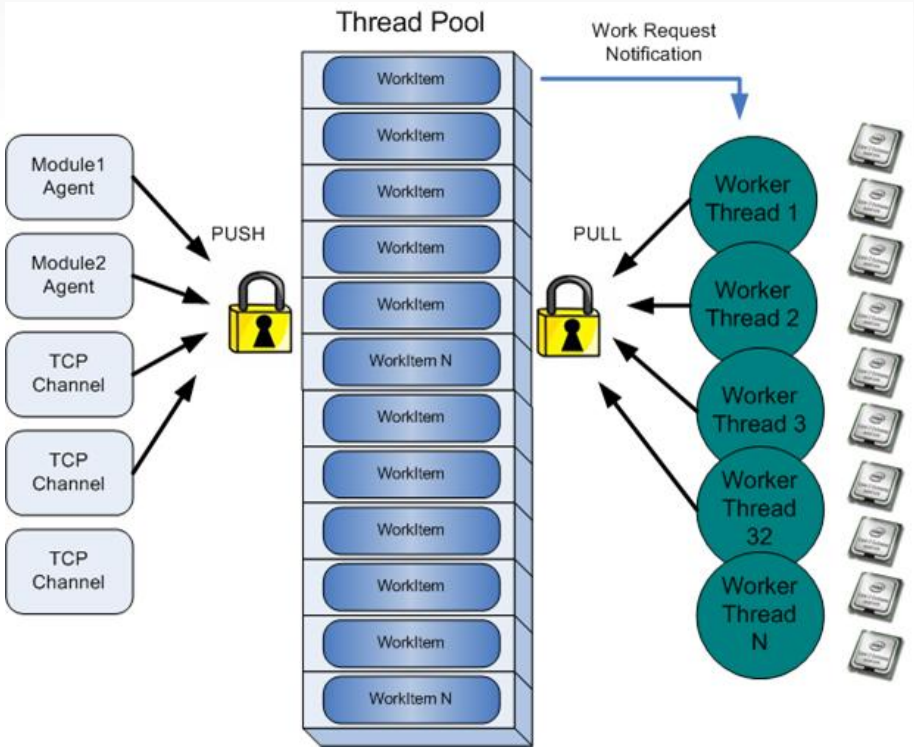
Thread Synchronization

- 스레드 동기화란?
  - 복수개의 스레드가 같은 리소스를 사용하고자 할 때, 순서대로 사용할 수 있도록 하여 리소스에 대한 처리가 제대로 될 수 있도록 함.

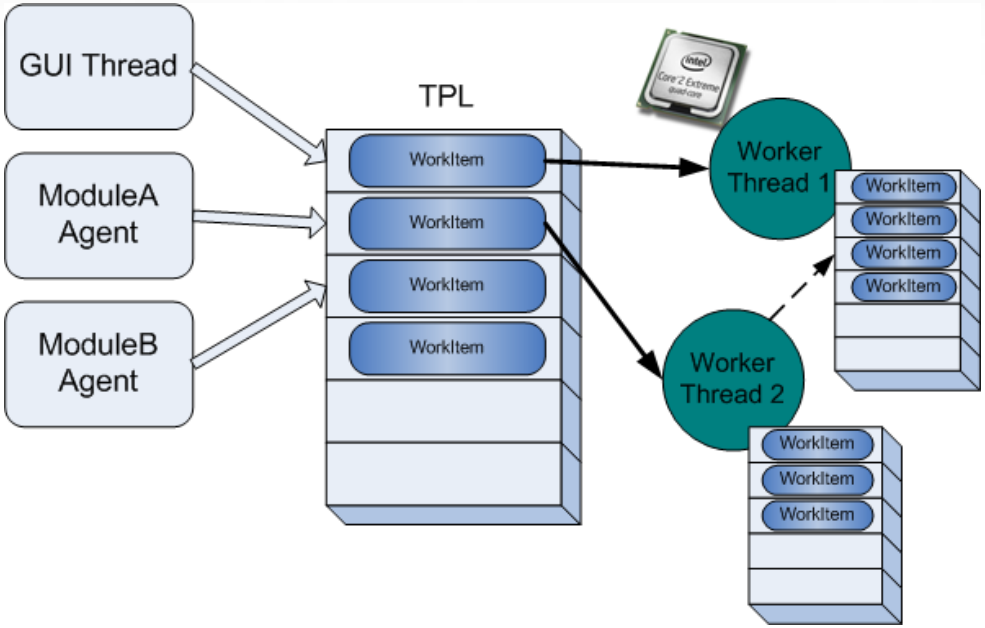
- Critical Section
  - 가장 가벼운 Thread Kernel object, 작은 범위, 적은 비용으로 내에서 동기화 수행
  - 같은 프로세스 내에서만 사용
- Mutex
  - 프로세스 간의 동기화도 수행 가능
  - 비용 증가, 처리 속도 느림
- Semaphore
  - 스레드를 제한된 개수만큼만 리소스를 사용할 수 있도록 동기화에 허용하는 객체
- Event
  - Signal을 이용하여 동기화를 수행하는 객체
  - Multi-Thread 에서 가장 유용하게 사용됨.

ThreadPool 비교

CLR 2 ThreadPool



CLR 4 ThreadPool

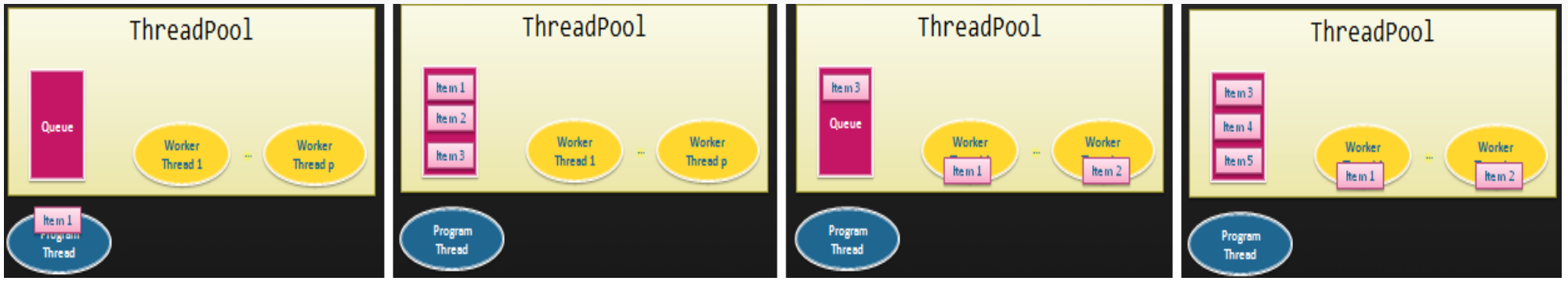


참고 : http://aviadezra.blogspot.com/2009/04/task-parallel-library-parallel.html

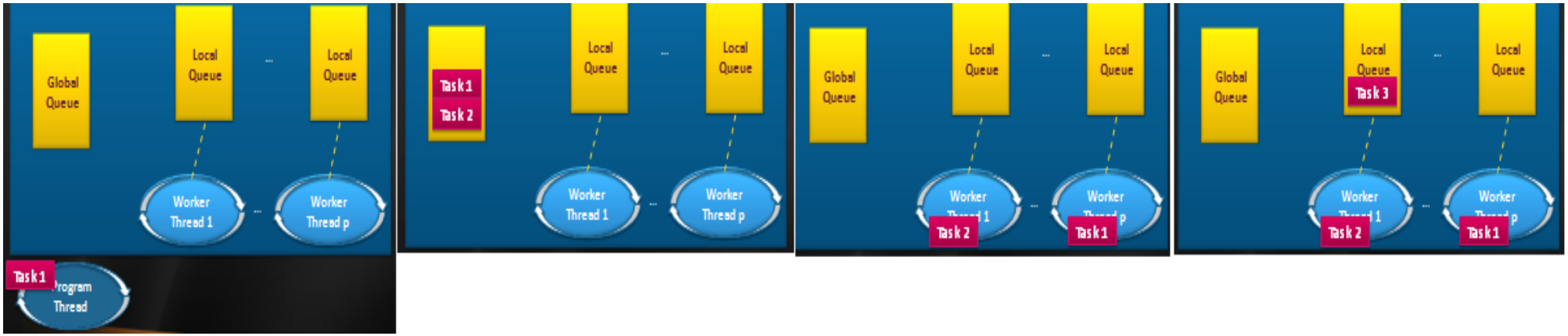
참고 : http://debop.egloos.com/3795151

ThreadPool 비교

CLR 2 ThreadPool

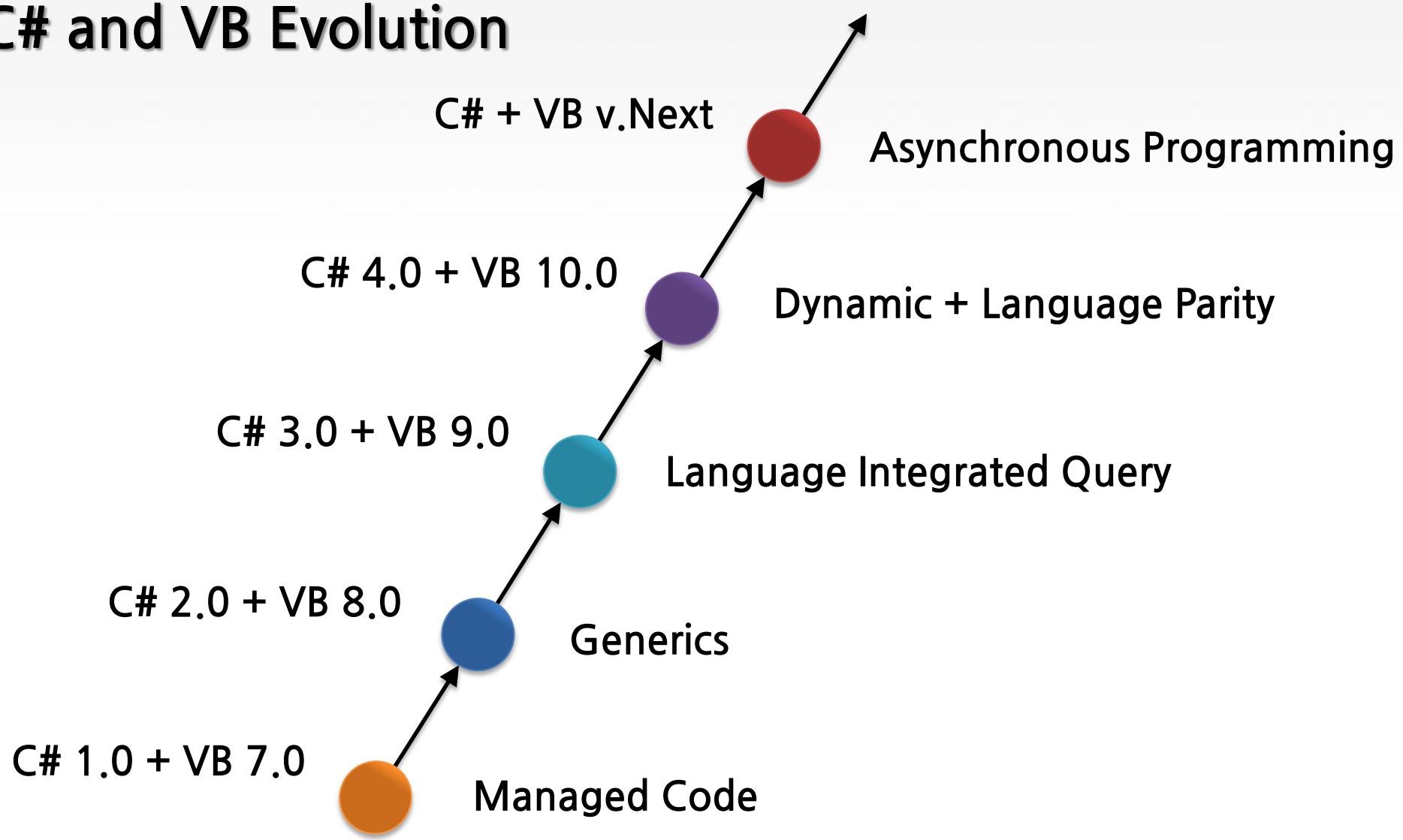


CLR 4 ThreadPool



Machine	.NET 3.5	.NET 4	Improvement
A dual-core box	5.03 seconds	2.45 seconds	2.05x
A quad-core box	19.39 seconds	3.42 seconds	5.67x

C# and VB Evolution

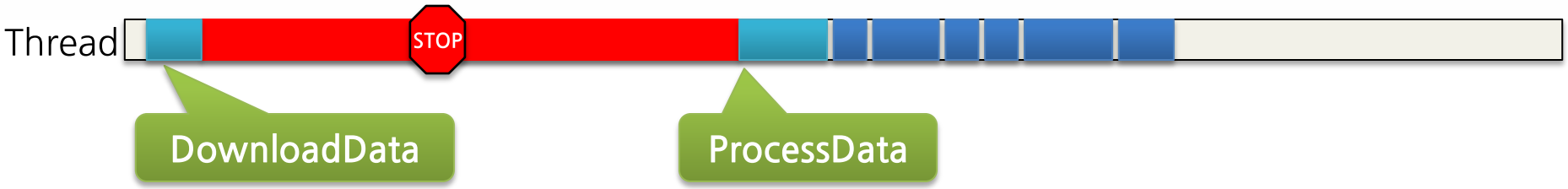


Asynchrony in a Nutshell

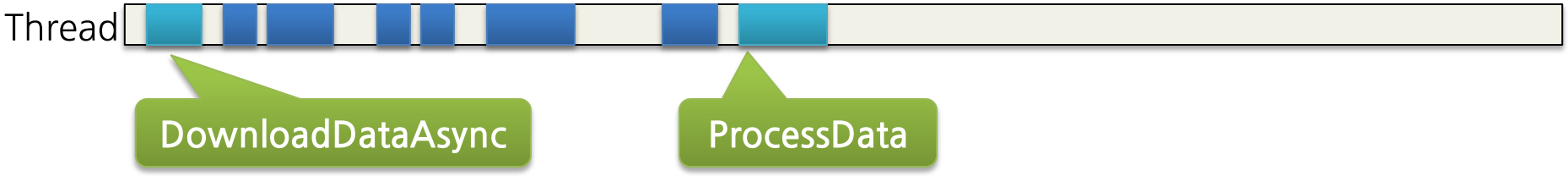
- Synchronous → Wait for result before returning
  - string DownloadString(...);
- Asynchronous → Return now, call back with result
  - void DownloadStringAsync(..., Action<string> callback);
- Asynchrony benefits
  - UI responsiveness: Frees UI thread for interaction
  - Server scalability: Thread can be reused for other requests

2.2 Synchronous vs. Asynchronous

```
var data = DownloadData(...);
ProcessData(data);
```

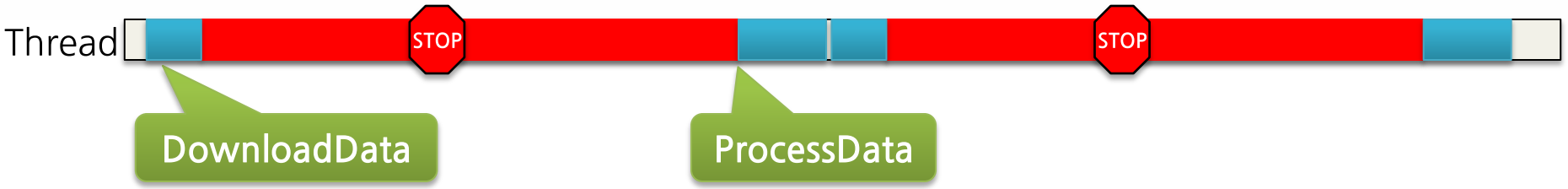



```
DownloadDataAsync(... , data => {
    ProcessData(data);
});
```

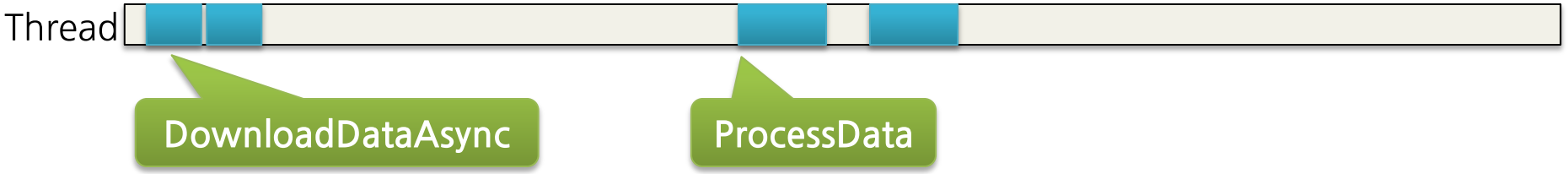


2.2 Synchronous vs. Asynchronous

```
var data = DownloadData(...);
ProcessData(data);
```




```
DownloadDataAsync(... , data => {
    ProcessData(data);
});
```



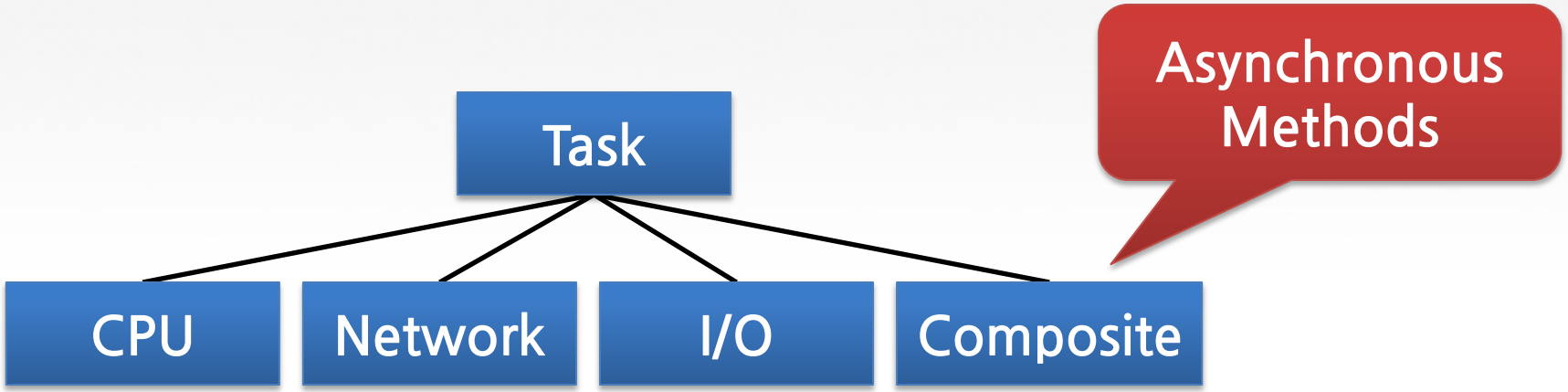
2.2 비동기 프로그래밍 디자인 패턴

- Asynchronous Programming Method (APM)
  - BeginXXXX, EndXXXX 메소드를 제공 (SqlCommand)
  - AsyncWaitHandle을 사용하여 실행 블로킹
  - AsyncCallback 대리자를 사용하여, 비동기 작업 종료

- Event-based Asynchronous Method (EAM)
  - XXXXAsync() 메소드 호출로 비동기 작업 시작
  - 비동기 작업 완료 시에 XXXXCompleted 이벤트 호출됨
  - 작업 취소 시에는 CancelAsync() 또는 MethodNameAsyncCancel() 호출
  - BackgroundWorker, WebClient 등이 이에 해당됨

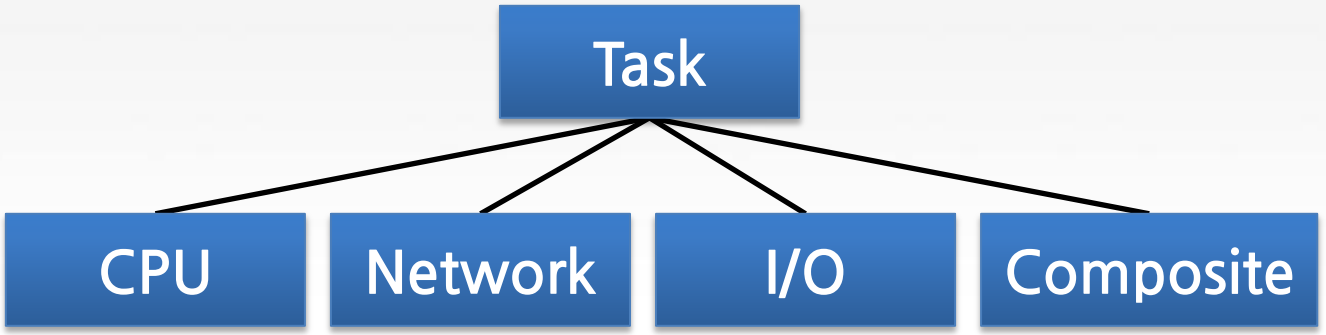
- Delegate를 이용한 비동기 작업
  - BeginInvoke(), EndInvoke() 사용
  - Task.Factory.FromAsync(), RCL.ParallelExtensions.DelegateAsync 클래스 참고

2.3 Unifying Asynchrony



- An asynchronous scenario
  - Scrape YouTube for video links
  - Download two or more videos concurrently
  - Create a mashup from downloaded videos
  - Save the resulting video

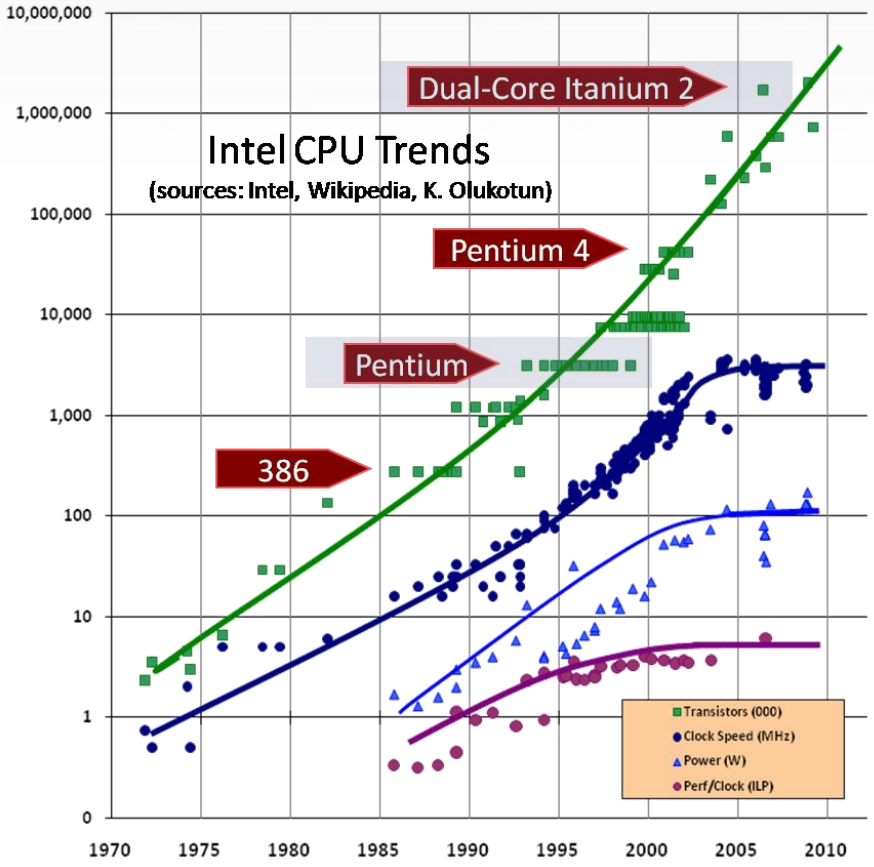
2.3 Unifying Asynchrony

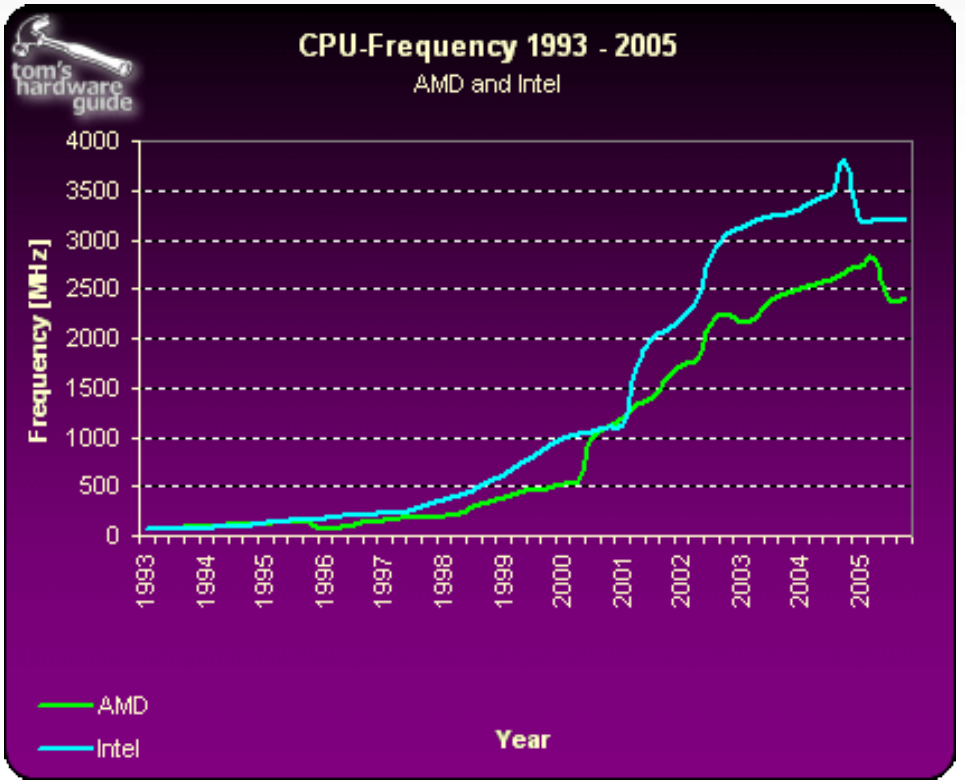



```
try {
    string[] videoUrls = await ScrapeYoutubeAsync(url);    // Network-bound
    Task<Video> t1 = DownloadVideoAsync(videoUrls[0]);     // Start two downloads
    Task<Video> t2 = DownloadVideoAsync(videoUrls[1]);
    Video[] vids = await Task.WhenAll(t1, t2);             // Wait for both
    Video v = await MashupVideosAsync(vids[0], vids[1]);   // CPU-bound
    await v.SaveAsync(textbox.Text);                       // IO-bound
}
catch (WebException ex) {
    ReportError(ex);
}
```

3.1 병렬 프로그래밍 필요성

CPU 개발 Trend - Clock 속도에서 Core 수 증가로!!! ➔ Multi-Core CPU 가 대세





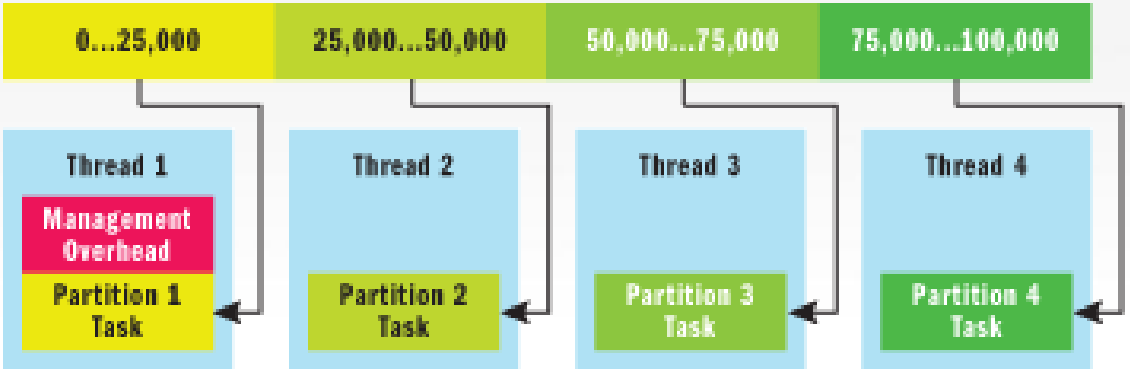
http://msdn.microsoft.com/en-us/library/ff963553.aspx

4.2 TPL (Task Parallel Library)

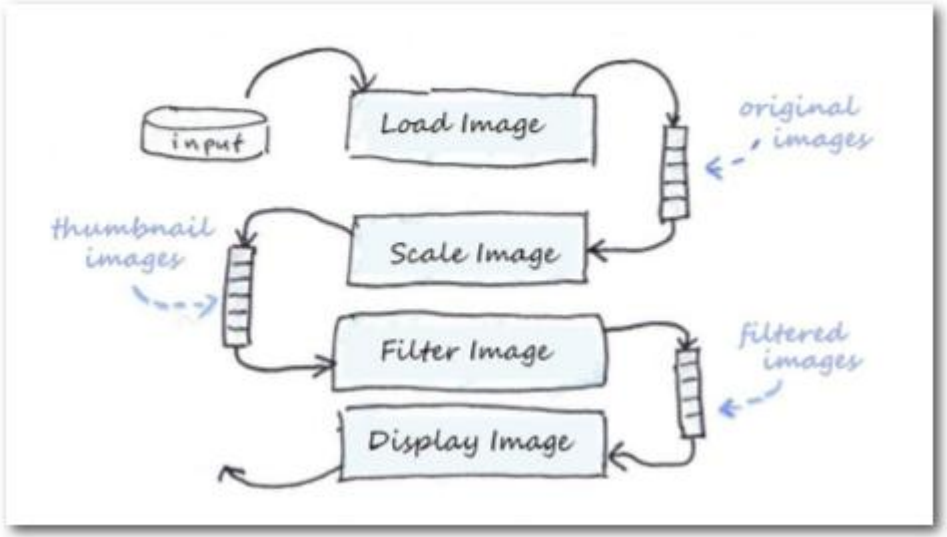
- Task
  - TaskFactory
  - Task

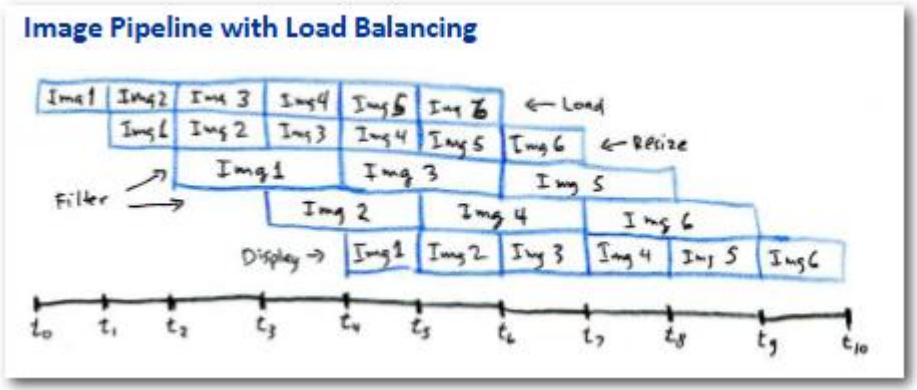
- Parallel.For, Parallel.ForEach, Parallel.Invoke
- PLINQ (.AsParallel())

4.2 TPL (Task Parallel Library)



Data Patitioning & Aggregate (PLINQ, Parallel.For())

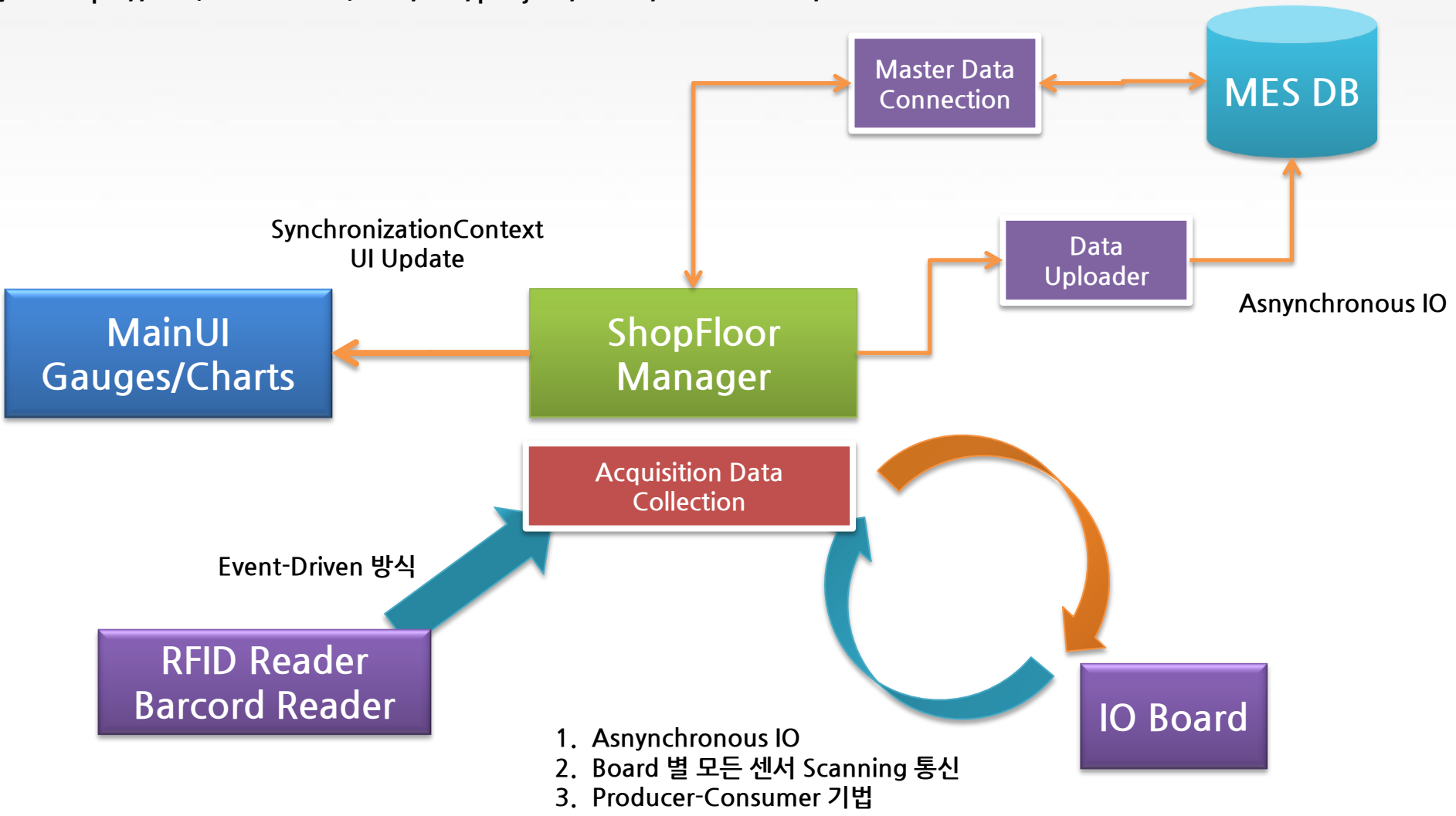




Pipeline algorithm (Task.ContinueWith 등)

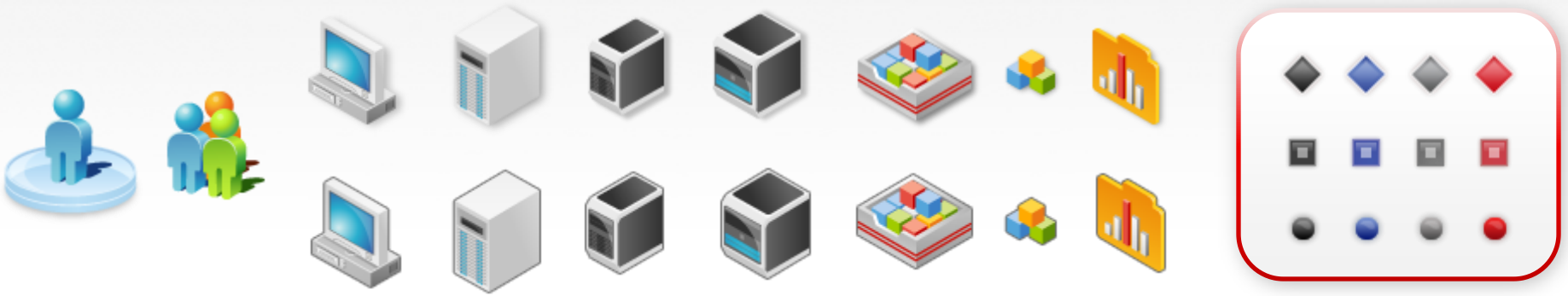
4.2 TPL Samples - ShopFloor Manager v2.0 구조

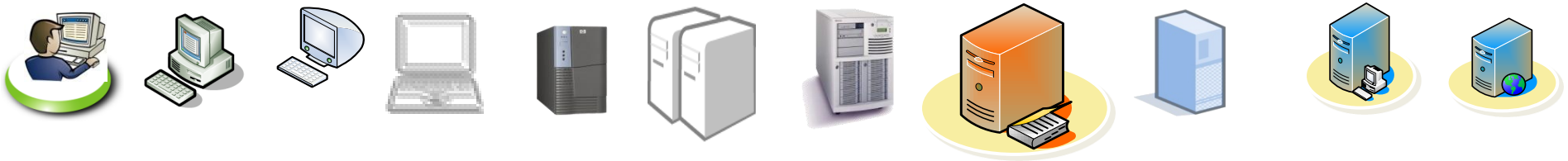
참고: https://svn.realweb21.com/svn/project/2009/KIMM-MES/trunk



2. ICON SET

Task Groups Computer Server Sub Process Bullet





감사합니다

Realweb