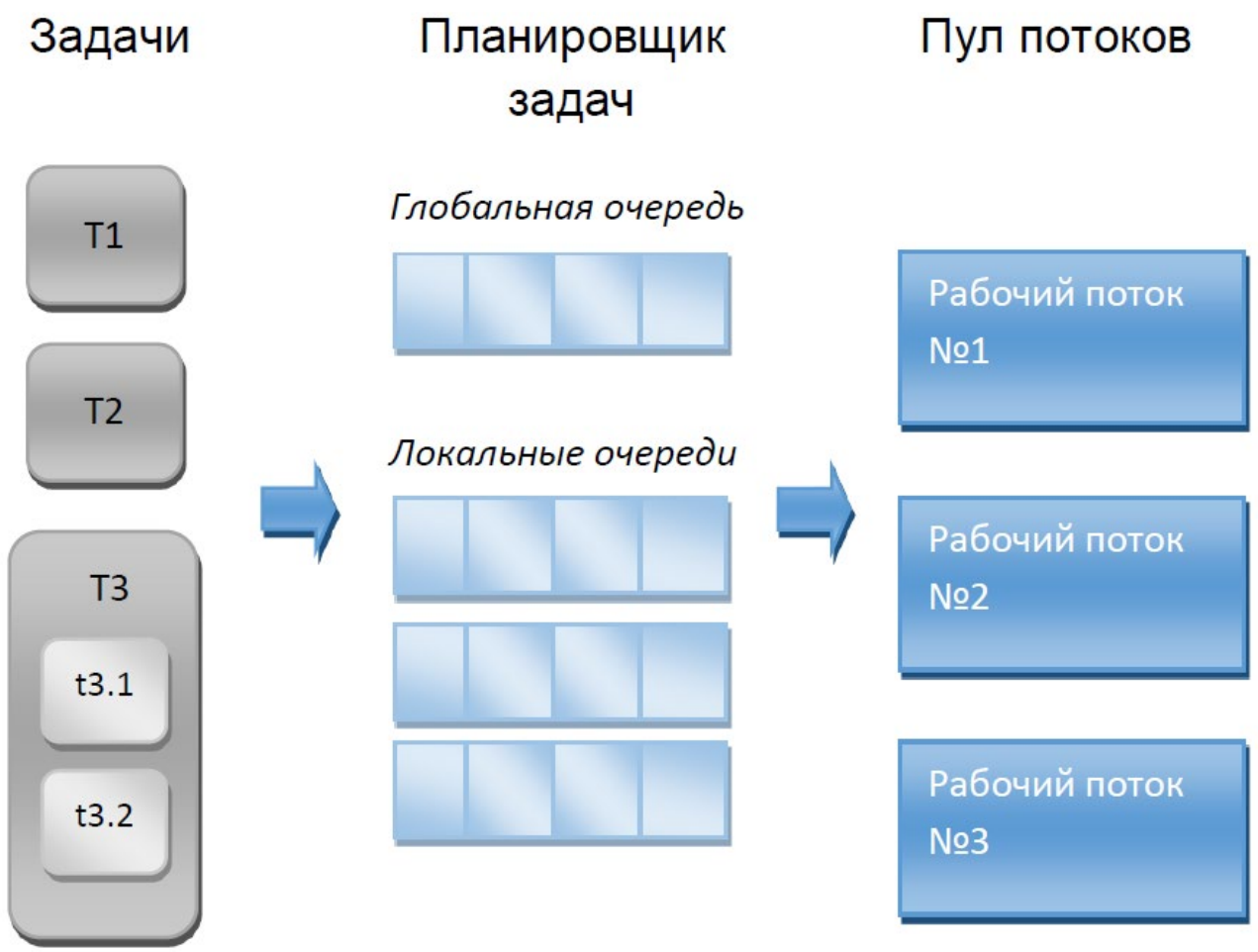
ПЛАНИРОВЩИК ЗАДАЧ

Основные принципы

Составляющие элементы процесса работы планировщика задач



Основные принципы

Запуск пользовательских и рабочих потоков

```
static void Main()
{
  Action IsPool = () =>
  {
    Console.WriteLine(Thread.CurrentThread.IsThreadPoolThread);
  };
  Console.WriteLine("Parallel.Invoke");
  Parallel.Invoke(IsPool, IsPool, IsPool, IsPool, IsPool);
  Console.WriteLine("Parallel.For");
  Parallel.For(0, 5, i => IsPool());
  Console.WriteLine("Task");
  Task.Factory.StartNew(IsPool).Wait();
  Console.WriteLine("Thread");
  new Thread(() => IsPool()).Start();
}
```

Основные принципы

Глобальная и локальные очереди задач

Основные принципы

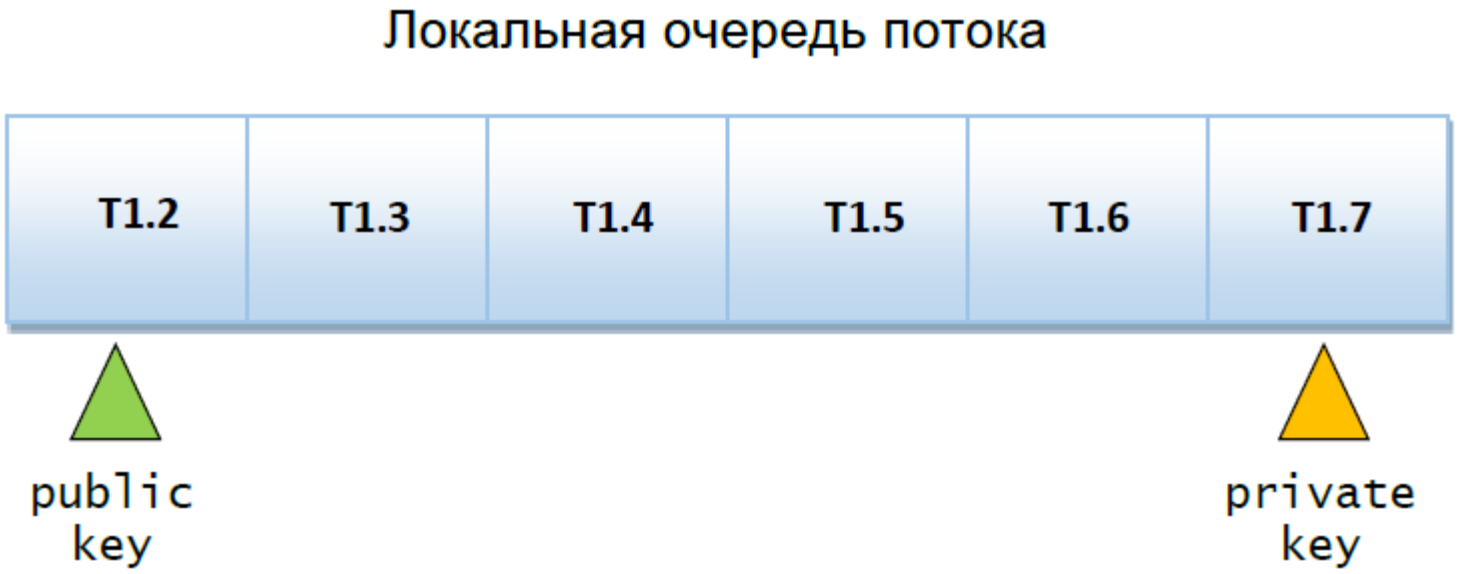
Родительская и вложенные задачи

```
Task t = Task.Factory.StartNew(() =>
{
  // готовим данные для подзадачи 1
  data1 = f1(data);
  // добавляем в очередь подзадачу 1
  Task t1 = Task.Factory.StartNew(() => { work(data1); });

  // готовим данные для подзадачи 2
  data2 = f2(data);
  // добавляем в очередь подзадачу 2
  Task t2 = Task.Factory.StartNew(() => { work(data2); });

  // готовим данные для подзадачи 3
  data3 = f3(data);
  // добавляем в очередь подзадачу 3
  Task t3 = Task.Factory.StartNew(() => { work(data3); });

  // ожидаем завершения подзадач
  Task.WaitAll(t1, t2, t3);
});
```

Опция PreferFairness

```
static void Main()
{
  Task tMain = Task.Factory.StartNew(() =>
  {
    var t1 = Task.Factory.StartNew(() => ...);
    var t2 = Task.Factory.StartNew(() => ...);
    var t3 = Task.Factory.StartNew(() => ...,
          TaskCreationOptions.PreferFairness);
    var t4 = Task.Factory.StartNew(() => ...,
          TaskCreationOptions.PreferFairness);
  });
}
```

Стратегии планировщика
Work Stealing



Стратегии планировщика
Inlined Execution

```
static void Main()
{
  Task tParent = Task.Factory.StartNew(() =>
  {
    var tChild1 = Task.Factory.StartNew(() => ...);
    var tChild2 = Task.Factory.StartNew(() => ...);
    tChild1.Wait();
  });
}
```

Планировщик фиксирует вызов *Wait* и передает управление ожидаемой задаче *tChild1*.

Стратегии планировщика
Thread Injection

Применяются два механизма динамического изменения числа рабочих потоков:

- планировщик добавляет еще один рабочий поток, если не наблюдает прогресса при выполнении задачи;
- добавляет или удаляет рабочие потоки в зависимости от результатов выполнения предыдущих задач.