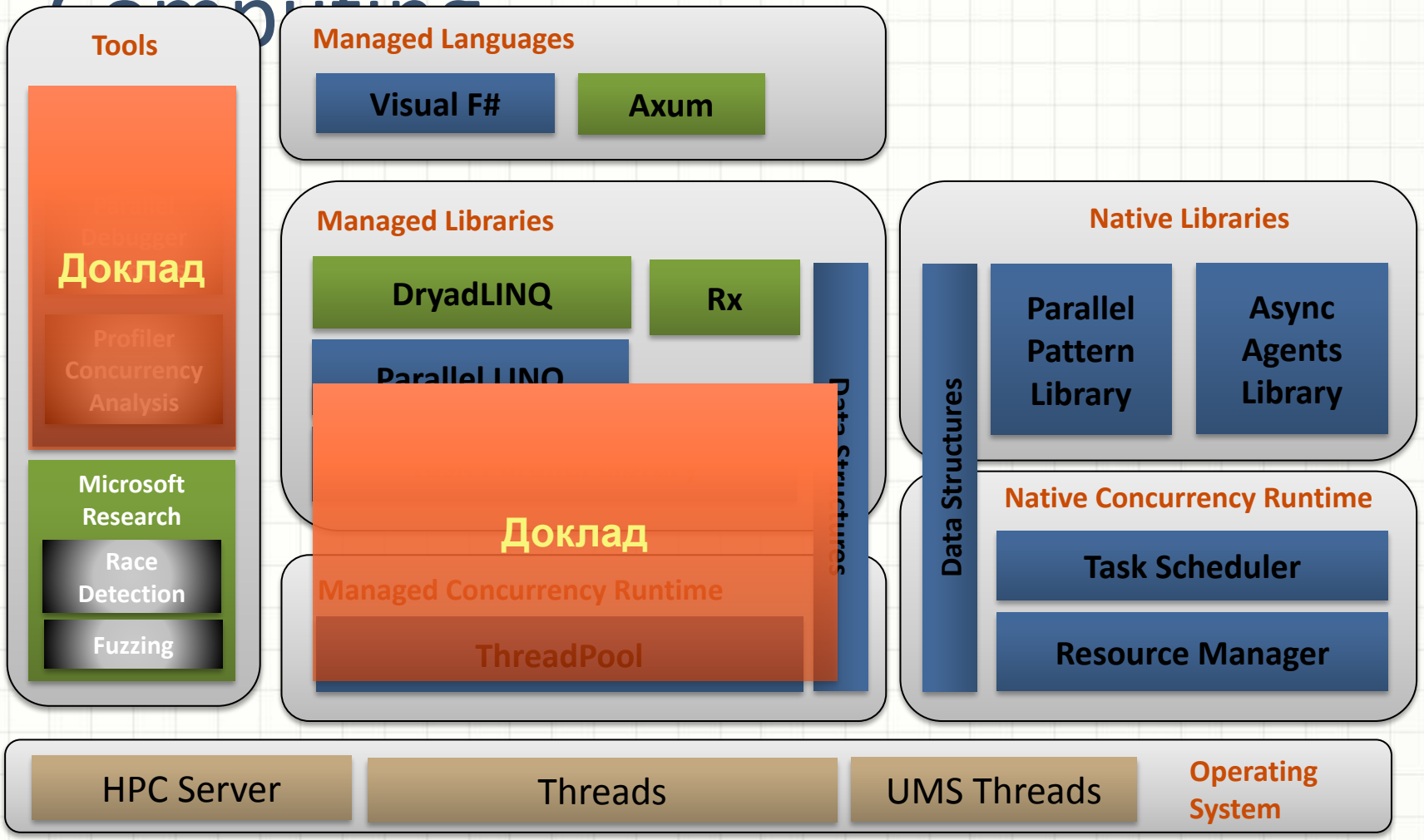
PARALLELISM В .NET 4.0 И VISUAL STUDIO 2010

Калита Роман

TaskManagementSoft

Нововведения в Parallel Computing

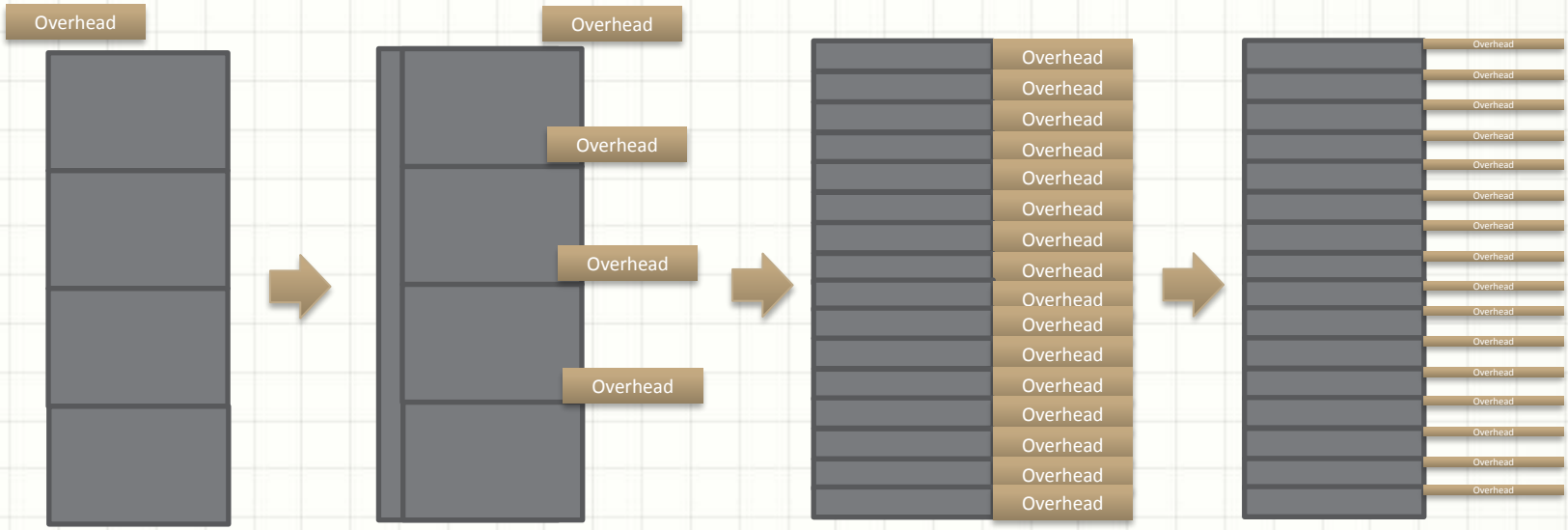


Легенда: Research / Incubation | Visual Studio 2010 / .NET 4 | Windows 7 / Server 2008 R2

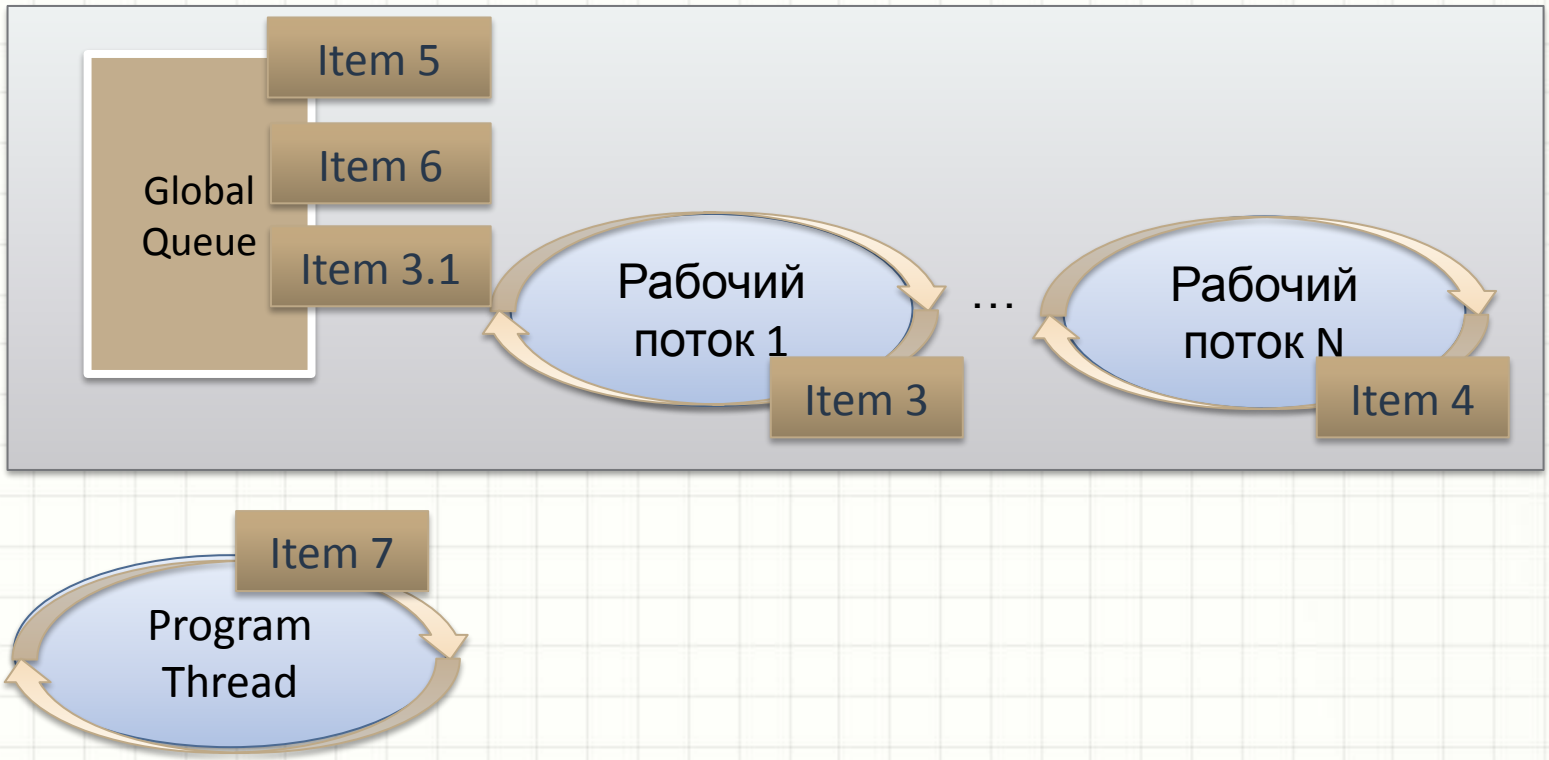
Демо - Parallel Sudoku

Накладые расходы при параллелизме

- Необходимо деление операций
- Выполнения потоков вызывает лишние накладные расходы
- Чем больше потоков тем больше лишних расходов
- Для быстрого запуска/выполнения потоков необходимо уменьшить накладные расходы связанные с этим

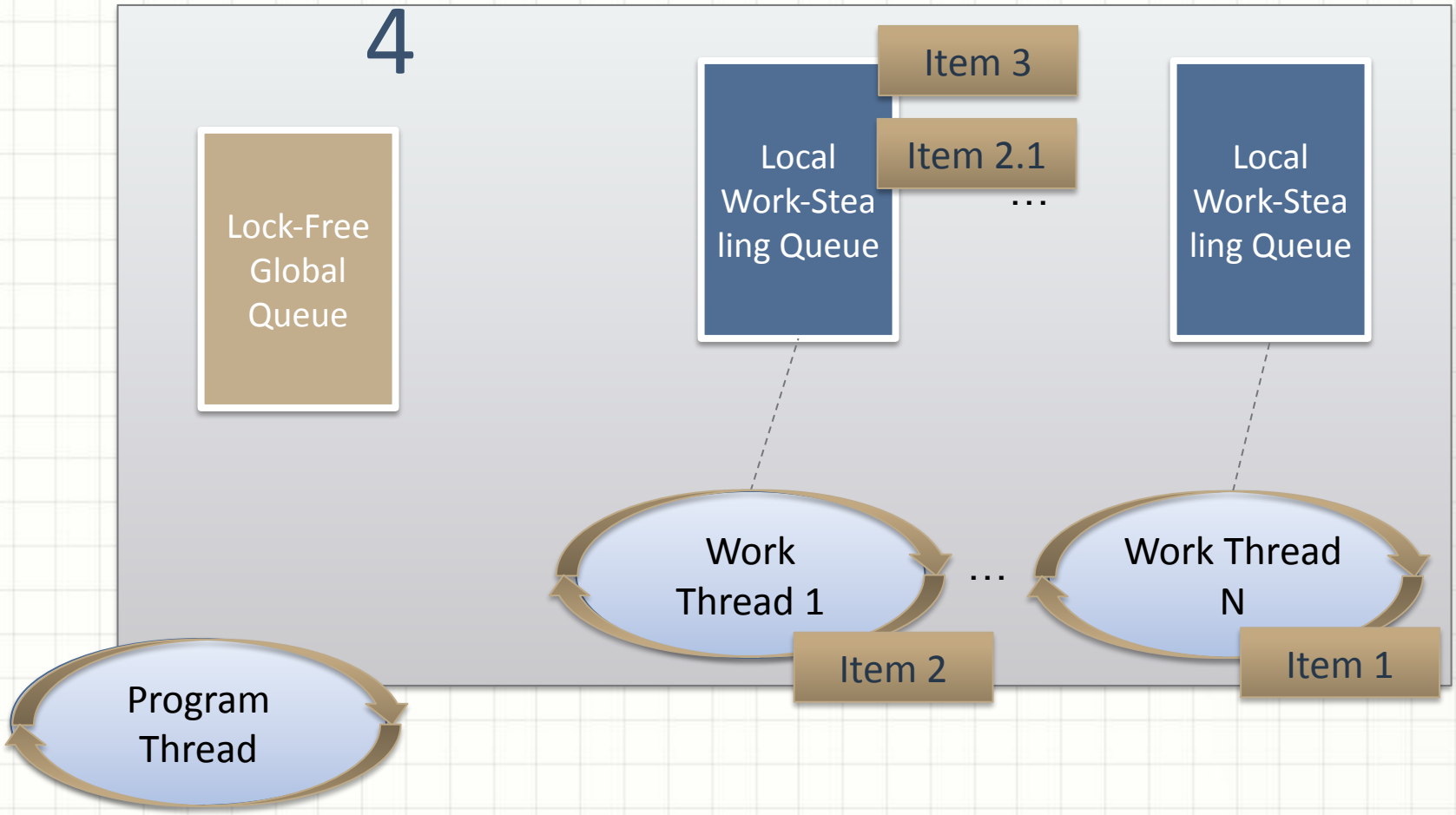

Overhead
Overhead
Overhead

ThreadPool в .NET 3.5



- При создании множества потоков все они обращаются в глобальную очередь за потоками – результат накладные расходы
- И возникает все больше и больше блокировок

ThreadPool в .NET 4



- Минимизации синхронизации и блокировок
- Hill-Climbing – определение оптимального числа потоков в пуле в зависимости от нагрузки на CPU

Task – новый тип для многопоточности

- ThreadPool.QueueUserWorkItem
  - Хорошо подходит для того чтобы стартовать и «забыть»
  - Но нехватает:
    - Waiting
    - Canceling
    - Continuing
    - Exceptions
    - Debugging
    - Dataflow between operations
    - …

Демо – Task, Concurrent collection

Новые примитивы синхронизации

- Использованы при разработке PLINQ и TPL
- Для того чтобы решать большинство задач в многопоточности

- Thread-safe, scalable collections
  - IProducerConsumerCollection<T>
    - ConcurrentQueue<T>
    - ConcurrentStack<T>
    - ConcurrentBag<T>
  - ConcurrentDictionary<TKey,TValue>
- Phases and work exchange
  - Barrier
  - BlockingCollection<T>
  - CountdownEvent
- Partitioning
  - {Orderable}Partitioner<T>
    - Partitioner.Create
- Exception handling
  - AggregateException
- Initialization
  - Lazy<T>
    - LazyInitializer.EnsureInitialized<T>

Распараллеливаем циклы

- Control flow is a primary source of work

```
for (int i = 0; i < n; i++)
{
    work(i);
}
```

```
foreach(var item in data)
{
    work(item);
}
```

```
StatementA();
StatementB;
StatementC();
```

- Распараллеливаем если итерации независимы

```
Parallel.For(0, n, i=>
{
    work(i);
});
```

```
Parallel.ForEach(data, item=>
{
    work(item);
});
```

```
Parallel.Invoke(
    () => StatementA(),
    () => StatementB,
    () => StatementC());
```

- «Синхронное» поведение, поток выполнения не пройдет пока цикл не выполнится
- Возможности
  - Cancelation, breaking, task-local state, scheduling, degree of parallelism
- Поддержка профайлера Visual Studio 2010 *(как у PLINQ)*

Легко с LINQ на PLINQ

- LINQ to Objects:

```
int[] output = arr
    .Select(x => Foo(x))
    .ToArray();
```

- PLINQ:

```
int[] output = arr.AsParallel()
    .Select(x => Foo(x))
    .ToArray();
```

Легко с LINQ на PLINQ

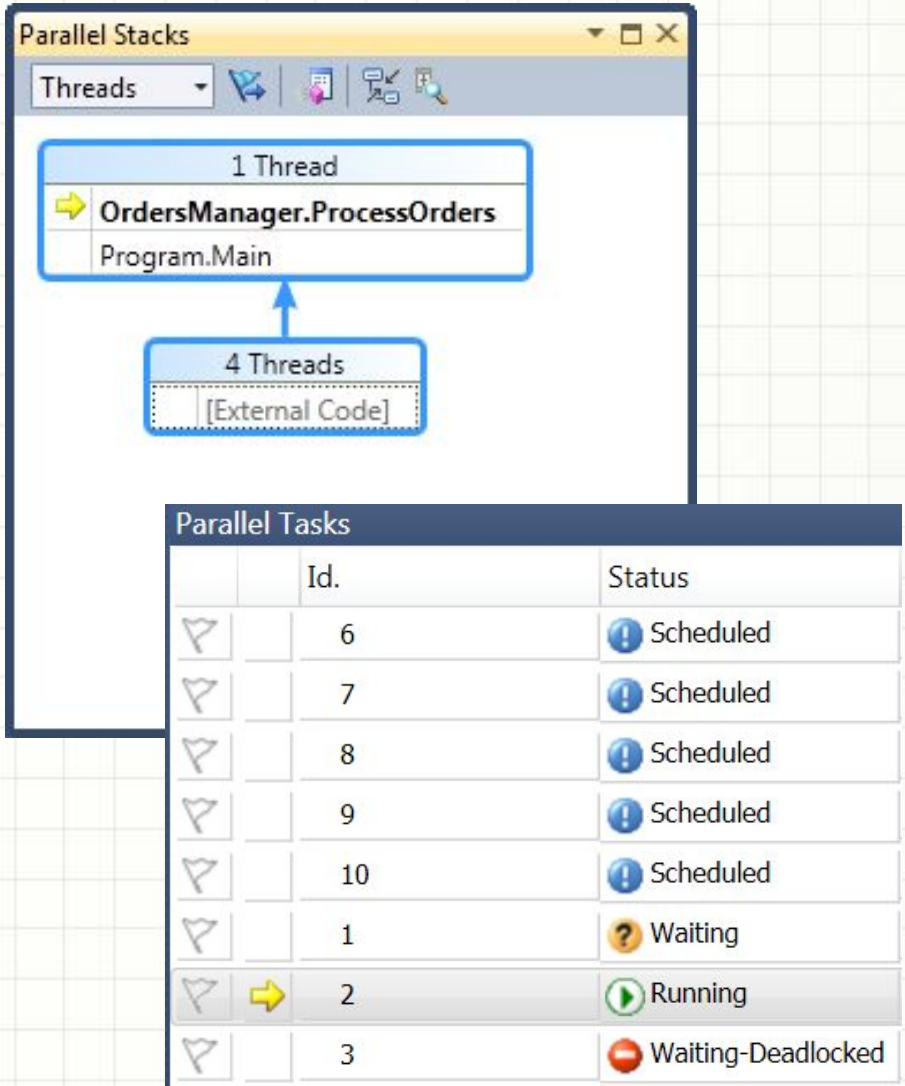
- PLINQ может выполнить все LINQ запросы
- Простые запросы – проще выполнить
- Разбивайте сложные запросы на более простые, так чтобы только та часть которую нужно распралелить была PLINQ:

```
src.Select(x => Foo(x))
       .TakeWhile(x => Filter(x))
       .AsParallel()
       .Select(x => Bar(x))
       .ToArray();
```

Демо – PLINQ

Отлаживаем параллелизм

- Concurrency Profiler
- Parallel Debugger
  - Parallel tasks
  - Parallel stack



Демо – Profiler views, debugger tools

Ссылки и источники

- DevCenter
  - http://msdn.com/concurrency
- Исходные коды примеров
  - http://code.msdn.microsoft.com/ParExtExamples
- Блоги
  - http://blogs.msdn.com/pfxteam
  -
  - Parallel stack
- Доклады, видео
  - http://channel9.msdn.com/learn
  - http://microsoftpdc.com/

Спасибо за внимание:)