Subject: Re: RE: Job package: A scope-bound worker thread for non-blocking operations.
Posted by mirek on Tue, 19 Sep 2017 08:12:17 GMT
View Forum Message <> Reply to Message

Small issues with the code:

sData template does not compile with MSC (I believe it is not legit C++ code, probably GCC eats it but it is not OK). Replaced by

```
template<class T>
One<T>*& sData()
{
 static thread_local One<T>* sData = NULL;
 return sData;
}
```

GetNewWorkerId has int64 internal counter (which IMO is reasonable), but the rest of the code is using int.

Anyway, as you are using "TIMING", it appears you are benchmarking in debug mode. I have done some changes:

```
#include <Core/Core.h>
#include <Job/Job.h>

using namespace Upp;

String GetDivisors()
{
 String s;
 int number = (int) 1000;
 Vector<int> divisors;
 for(auto i = 1, j = 0; i < number + 1; i++) {
  auto d = number % i;
  if(d == 0){
   divisors.Add(i);
   j++;
  }
  if(i == number)
   s = Format("Worker Id: %d, Number: %d, Divisors (count: %d): %s",
      GetWorkerId(),
      number,
      j,
```

```
      divisors.ToString());
 }
 return pick(s);
}

// #define OUTPUT
#define N 100000

CONSOLE_APP_MAIN
{
 Array<Job<String>> jobs;
 jobs.SetCount(CPU_Cores() + 2);

 CoWork cowork;
// cowork.SetPoolSize(CPU_Cores() + 2);

 Vector<String> results;
 RDUMP(CPU_Cores());
 {

  RTIMING("CoWork -- With stdout output");
  for(int i = 0; i < N; i++)
   cowork & [=, &results] { String h = GetDivisors(); CoWork::FinLock(); results.At(i) = h; };
  cowork.Finish();
  // Stdout output section.
 #ifdef OUTPUT
  for(auto& r : results)
   Cout() << r << '\n';
 #endif
 }
 {
  RTIMING("Job -- With stdout output");

  int i = 0;
  while(i < N) {
   for(auto& job : jobs) {
    if(!job.IsFinished()) {
     continue;
    }
    job & [=]{ Job<String>::Data() = GetDivisors(); };
    if(!(~job).IsEmpty()) {
    #ifdef OUTPUT
     Cout() << ~job << '\n';
    #endif
     if(++i == N) break;
    }
   }
```

```
  }
 }
}
```

with results in _RELEASE_ mode:

* d:\upp.out\MyApps\MSC15\JobBenchmark.exe 19.09.2017 10:01:27, user: cxl

CPU_Cores() = 8
TIMING Job -- With stdout output: 123.00 ms - 123.00 ms (123.00 ms / 1 ), min: 123.00 ms, max: 123.00 ms, nesting: 1 - 1
TIMING CoWork -- With stdout output: 103.00 ms - 103.00 ms (103.00 ms / 1 ), min: 103.00 ms, max: 103.00 ms, nesting: 1 - 1

Now I fully understand that performance is not the reason for Job, but as I have seen no technical reasons why CoWork should be that much slower, I had to check....