Subject: Re: RE: Job package: A scope-bound worker thread for non-blocking operations.
Posted by Oblivion on Tue, 19 Sep 2017 06:12:31 GMT
View Forum Message <> Reply to Message

Hello Mirek,

(Thank you for your comments. I deeply appreciate it.)

I was curious about if the new design of Job class will pay off, whether it is also a resonable general paralellization tool, and did some benchmarking with the Divisors example.

I assumed Job and CoWork to be functionally and effectively identical in this example (In the sense that, regardless of their internals, they are both doing the same thing: Calculating divisiors for the number 1000, 10.000 times in available worker threads, then printing the results to the screen.)

I simply changed the jobs loop to take advantage of new return semantics (I don't know if CoWork can be put into a similar loop, so I am taking this with a grain of salt):
The loop for the job is simply a very crude slot manager for 8 Job workers. (Tested on AMD FX 6100, six core processor.)

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

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

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

  TIMING("CoWork -- With stdout output");
  for(int i = 0; i < 10000; i++)
   cowork & [=, &results] { String h = GetDivisors(); CoWork::FinLock(); results.At(i) = h; };
  cowork.Finish();
  // Stdout output section.
  for(auto& r : results)
   Cout() << r << '\n';

 }
 {
  TIMING("Job -- With stdout output");

  int i = 0;
  while(i < 10000) {
```

```
  for(auto& job : jobs) {
   if(!job.IsFinished()) {
    continue;
   }
   job & [=]{ Job<String>::Data() = GetDivisors(); };
   if(!(~job).IsEmpty()) {
    Cout() << ~job << '\n';
    if(++i == 10000) break;
   }
  }

 }
}
```

Results (consistent):

For 10000 computiation.

CPU_Cores() = 6
TIMING Job -- With stdout output: 370.00 ms - 370.00 ms (370.00 ms / 1), min: 370.00 ms, max: 370.00 ms, nesting: 1 - 1
TIMING CoWork -- With stdout output: 461.00 ms - 461.00 ms (461.00 ms / 1), min: 461.00 ms, max: 461.00 ms, nesting: 1 - 1

CPU_Cores() = 6
TIMING Job -- Without stdout output: 228.00 ms - 228.00 ms (228.00 ms / 1), min: 228.00 ms, max: 228.00 ms, nesting: 1 - 1
TIMING CoWork -- Without stdout output: 234.00 ms - 234.00 ms (234.00 ms / 1), min: 234.00 ms, max: 234.00 ms, nesting: 1 - 1

for 1000 compuitation.

CPU_Cores() = 6
TIMING Job -- With stdout output: 34.00 ms - 34.00 ms (34.00 ms / 1), min: 34.00 ms, max: 34.00 ms, nesting: 1 - 1
TIMING CoWork -- With stdout output: 53.00 ms - 53.00 ms (53.00 ms / 1), min: 53.00 ms, max: 53.00 ms, nesting: 1 - 1

CPU_Cores() = 6
TIMING Job -- Without stdout output: 24.00 ms - 24.00 ms (24.00 ms / 1), min: 24.00 ms, max: 24.00 ms, nesting: 1 - 1
TIMING CoWork -- Without stdout output: 31.00 ms - 31.00 ms (31.00 ms / 1), min: 31.00 ms, max: 31.00 ms, nesting: 1 - 1

What do you think?

Best regards,
Oblivion
