Subject: Re: CoWork::Finish() can wait in a worker thread while there are jobs to do
Posted by mirek on Fri, 16 Dec 2016 00:33:03 GMT
View Forum Message <> Reply to Message

busiek wrote on Thu, 15 December 2016 21:30Hi Mirek,

It is said that CoWork instances can be nested. Looking into the code I see that it is also possible to use CoWork in a job (i.e. in a worker thread). Therefore CoWork::Finish() can be called inside a worker thread. The current implementation of Finish() waits if there is no more jobs to take from the local queue of CoWork while not all local jobs are finished. This is a waste of resources because a worker should never wait while there are any global jobs to do from the pool. Why not implement CoWork::Finish() like that:

```
void CoWork::Finish() {
 Pool& p = GetPool();
 p.lock.Enter();
 while(todo) {
  if(!jobs.IsEmpty(1)) {
   LLOG("Finish: todo: " << todo << " (CoWork " << FormatIntHex(this) << ")");
   p.DoJob(*jobs.GetNext(1));
  } else if(is_worker && !p.jobs.IsEmpty()) {
   LLOG("Do global job while WaitForFinish (CoWork " << FormatIntHex(this) << ")");
   p.DoJob(*p.jobs.GetNext());
  } else if(is_worker)
   p.waiting_threads++;
   LLOG("Waiting for job in WaitForFinish");
   p.waitforjob.Wait(p.lock);
   LLOG("Waiting ended in WaitForFinish");
   p.waiting_threads--;
  } else {
   LLOG("WaitForFinish (CoWork " << FormatIntHex(this) << ")");
   waitforfinish.Wait(p.lock);
  }
 }
 p.lock.Leave();
 LLOG("CoWork " << FormatIntHex(this) << " finished");
}?
```

The only problem is that a worker can wait on different conditional variables. If the worker waits on the global conditional variable (p.waitforjob) and some other worker finishes remaining jobs in local CoWork queue reducing todo to zero, the worker will not be waked up since the other worker signals waitforfinish. Also removing the entire "else if(is_worker)" block and waiting only on waitforfinish is a bad solution either since if a new job is scheduled by PushJob() the worker will not be waked up. Therefore more complicated code is needed. I attach a patch with a proposition. Simply one has to track all waiting workers in Finish() which I call "waiting masters" and singal one of them in PushJob().

Jakub

Hi,

it is deliberate. The issue is that

a) 'global' job can be part of some really unrelated work and can take order of magnitude more time, thus blocking this CoWork to finish.

b) stack issue. Diving into global job can end in another Finish, that would dive into yet another globa... and we unexpectedly run out of stack.

Really, Finish doing 'any global job' was previous implementation, the limit to 'my jobs' is new.

Mirek
