Subject: Parallel sort
Posted by mirek on Tue, 05 Jan 2016 18:03:21 GMT
View Forum Message <> Reply to Message

Inspired by hugely misleading article here:

http://www.codeproject.com/Articles/1062000/Multithreaded-Pa rallel-Selection-Sort-Algorithm-Us

I have decided it is time to provide parallel sort :) I guess it will a nice theme for the next CodeProject article and also a beginning of eventual 'parallel algorithms' part of Core (although I am not a huge believer in low-level parallel routines).

Here is the code, including simple benchmark:

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

using namespace Upp;

struct CoSorter {
 CoWork cw;

 enum { PARALLEL_THRESHOLD = 50 };

 template <class I, class Less>
 void CoSort(I l, I h, const Less& less)
 {
  for(;;) {
   int count = int(h - l);
   if(count < 2)
    return;
   if(count < 8) {                        // Final optimized SelectSort
    ForwardSort(l, h, less);
    return;
   }
   int pass = 4;
   for(;;) {
    I middle = l + (count >> 1);        // get the middle element
    OrderIter2__(l, middle, less);      // sort l, middle, h-1 to find median of 3
    OrderIter2__(middle, h - 1, less);
    OrderIter2__(l, middle, less);      // median is now in middle
    IterSwap(l + 1, middle);            // move median pivot to l + 1
    I ii = l + 1;
    for(I i = l + 2; i != h - 1; ++i)   // do partitioning; already l <= pivot <= h - 1
     if(less(*i, *(l + 1)))
      IterSwap(++ii, i);
    IterSwap(ii, l + 1);                // put pivot back in between partitions
    I iih = ii;
```

```
    while(iih + 1 != h && !less(*ii, *(iih + 1))) // Find middle range of elements equal to pivot
     ++iih;
    if(pass > 5 || min(ii - l, h - iih) > (max(ii - l, h - iih) >> pass)) { // partition sizes ok or we have
done max attempts
     if(ii - l < h - iih - 1) {       // recurse on smaller partition, tail on larger
      if(ii - l < PARALLEL_THRESHOLD)
       CoSort(l, ii, less);
      else
       cw & [=] { CoSort(l, ii, less); };
      l = iih + 1;
     }
     else {
      if(h - iih - 1 < PARALLEL_THRESHOLD)
       CoSort(iih + 1, h, less);
      else
       cw & [=] { CoSort(iih + 1, h, less); };
      h = ii;
     }
     break;
    }
    IterSwap(l, l + (int)Random(count));     // try some other random elements for median pivot
    IterSwap(middle, l + (int)Random(count));
    IterSwap(h - 1, l + (int)Random(count));
    pass++;
   }
  }
 }
};

template <class I, class Less>
void CoSort(I l, I h, const Less& less)
{
 CoSorter().CoSort(l, h, less);
}

template <class T, class Less>
void CoSort(T& c, const Less& less)
{
 CoSort(c.Begin(), c.End(), less);
}

template <class T>
void CoSort(T& c)
{
 typedef typename T::ValueType VT;
 CoSort(c.Begin(), c.End(), StdLess<VT>());
}
```

```
#ifdef _DEBUG
#define N 1000000
#else
#define N 10000000
#endif

CONSOLE_APP_MAIN
{
 Vector<String> a, b;
 for(int i = 0; i < N; i++) {
  String s = AsString(Random());
  a.Add(s);
  b.Add(s);
 }

 {
  RTIMING("Sort");
  Sort(a);
 }
 {
  RTIMING("CoSort");
  CoSort(b);
 }

 ASSERT(a == b);
}
```

Quite simple to do this with CoWork, is not it? :)

Now for benchmark numbers (Core i7):

```
TIMING CoSort        : 499.00 ms - 499.00 ms (499.00 ms / 1 ), min: 499.00 ms, max: 499.00 ms, nesting: 1 - 1
TIMING Sort          :  2.15 s  -  2.15 s  ( 2.15 s  / 1 ), min:  2.15 s , max:  2.15 s , nesting: 1 - 1
```

Ratio is 4.3, which is better than number of physical cores. Hard to say if one should expect more from hyperthreading (virtual cores), but I would say it is OK result.

My only concern is actually about naming.

Is using "Co" as prefix for all things multithreaded a good idea?

Mirek

Subject: Re: Parallel sort
Posted by Klugier on Tue, 05 Jan 2016 20:27:38 GMT
View Forum Message <> Reply to Message

Hello,

Quote:
Is using "Co" as prefix for all things multithreaded a good idea?

I think the other name in current situation (We have CoWork) doesn't make sense. Maybe just ParrallelSort and ParalleWork is good alternative, if we take into account my first argument it is not. Is Co from Cooperation?

Moreover I think that, the good idea is to make parallel image scaling in Draw.

Sincerely,
Klugier

Subject: Re: Parallel sort
Posted by mirek on Tue, 05 Jan 2016 20:42:56 GMT
View Forum Message <> Reply to Message

Klugier wrote on Tue, 05 January 2016 21:27Hello,

Quote:
Is using "Co" as prefix for all things multithreaded a good idea?

I think the other name in current situation (We have CoWork) doesn't make sense. Maybe just ParrallelSort and ParalleWork is good alternative, if we take into account my first argument it is not. Is Co from Cooperation?

Moreover I think that, the good idea is to make parallel image scaling in Draw.

Sincerely,
Klugier

Well, coworker is somebody who works with you, so the idea was that CoWork is something done with multiple agents :)

Anyway, the reason why I ask is that "coroutines" have established meaning and are sort of opposite of multithreading.