Subject: Re: Map implementation
Posted by mirek on Wed, 26 Jun 2019 10:07:56 GMT
View Forum Message <> Reply to Message

Well, with benchmark constructed like this, beating set<int>::erase is accidentally hard.

Interesting things is that if you change the order of loops:

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

using namespace Upp;

CONSOLE_APP_MAIN
{
#ifdef _DEBUG
 const int v_num = 10000;
#else
 const int v_num = 100000;
#endif

 const int isize = 100;

 {
  Vector<Index<int> > v;
  v.SetCount(v_num);
  {
   RTIMING("FindAdd v_num outer");
   for (int j = 0; j < v_num; ++j)
    for (int i = 0; i < isize; ++i)
     v[j].FindAdd(i);
  }
  {
   RTIMING("UnlinkKey v_num outer");
   for (int j = 0; j < v_num; ++j)
    for (int i = 0; i < isize; ++i)
     v[j].UnlinkKey(i);
  }
  RTIMING("Sweep v_num outer");
  const int jsize = v_num;
  for (int j = 0; j < jsize; ++j)
   v[j].Sweep();
 }
 {
  Vector<Index<int> > v;
  v.SetCount(v_num);
  {
```

```
  RTIMING("FindAdd v_num inner");
  for (int i = 0; i < isize; ++i)
   for (int j = 0; j < v_num; ++j)
    v[j].FindAdd(i);
 }
 {
  RTIMING("UnlinkKey v_num inner");
  for (int i = 0; i < isize; ++i)
   for (int j = 0; j < v_num; ++j)
    v[j].UnlinkKey(i);
 }
 RTIMING("Sweep v_num inner");
 const int jsize = v_num;
 for (int j = 0; j < jsize; ++j)
  v[j].Sweep();
}

{
 std::set<int> *v = new std::set<int>[v_num];
 {
  RTIMING("insert v_num outer");
  for (int j = 0; j < v_num; ++j)
   for (int i = 0; i < isize; ++i)
    v[j].insert(i);
 }

 {
  RTIMING("erase v_num outer");
  for (int j = 0; j < v_num; ++j)
   for (int i = 0; i < isize; ++i)
    v[j].erase(i);
 }
 delete[] v;
}

{
 std::set<int> *v = new std::set<int>[v_num];
 {
  RTIMING("insert v_num inner");
  for (int i = 0; i < isize; ++i)
   for (int j = 0; j < v_num; ++j)
    v[j].insert(i);
 }

 {
  RTIMING("erase v_num inner");
  for (int i = 0; i < isize; ++i)
   for (int j = 0; j < v_num; ++j)
```

```
    v[j].erase(i);
  }
  delete[] v;
 }
}
```

results are very different:

```
* C:\upp\out\benchmarks\MINGWx64\Index.exe 26.06.2019 11:36:42, user: cxl

TIMING erase v_num inner: 480.00 ms - 480.00 ms (480.00 ms / 1 ), min: 480.00 ms, max: 480.00 ms, nesting: 0 - 1
TIMING insert v_num inner: 702.00 ms - 702.00 ms (702.00 ms / 1 ), min: 702.00 ms, max: 702.00 ms, nesting: 0 - 1
TIMING erase v_num outer: 427.00 ms - 427.00 ms (427.00 ms / 1 ), min: 427.00 ms, max: 427.00 ms, nesting: 0 - 1
TIMING insert v_num outer: 399.00 ms - 399.00 ms (399.00 ms / 1 ), min: 399.00 ms, max: 399.00 ms, nesting: 0 - 1
TIMING Sweep v_num inner: 22.00 ms - 22.00 ms (22.00 ms / 1 ), min: 22.00 ms, max: 22.00 ms, nesting: 0 - 1
TIMING UnlinkKey v_num inner: 995.00 ms - 995.00 ms (995.00 ms / 1 ), min: 995.00 ms, max: 995.00 ms, nesting: 0 - 1
TIMING FindAdd v_num inner: 683.00 ms - 683.00 ms (683.00 ms / 1 ), min: 683.00 ms, max: 683.00 ms, nesting: 0 - 1
TIMING Sweep v_num outer: 28.00 ms - 28.00 ms (28.00 ms / 1 ), min: 28.00 ms, max: 28.00 ms, nesting: 0 - 1
TIMING UnlinkKey v_num outer: 118.00 ms - 118.00 ms (118.00 ms / 1 ), min: 118.00 ms, max: 118.00 ms, nesting: 0 - 1
TIMING FindAdd v_num outer: 242.00 ms - 242.00 ms (242.00 ms / 1 ), min: 242.00 ms, max: 242.00 ms, nesting: 0 - 1
```

which probably means that set<int> is more cache friendly in the original benchmark....
