摘要

对如下map实现作benchmark：

- standard map
- standard hash map
- google dense hash map
- google sparse hash map
- nark hash map

概念

google dense & sparse hash map

不同点

dense hash map

- 查找速度快 [1]
- 使用之前，**必须**调用`set_empty_key()`函数设置`empty bucket`的key值 [1]

sparse hash map

- 空间开销小，每个entry占用1-2bit [1]

相同点

- 如果需要删除操作，**必须**调用`set_deleted_key()`函数设置`deleted bucket`的key值 [1]
- 在初始化时，插入元素的数量可以作为可选参数，相比于SGI的`hash_map`可以把桶的数量作为初始化时的可选参数 [1]
- `erase()`操作 [1]
  - `erase()`操作不会立即释放内存，也不会导致哈希表的`resize`操作；相反，调用`erase()`会增加哈希表的内存占用 [2,3]
  - `erase()`操作不会使迭代器失效，也就是说，下面代码合法

    ```
    for (it = ht.begin(); it != ht.end(); ++it)
            if (...) ht.erase(it);
    ```

  - `erase()`操作不会减少哈希表的内存占用，其会在下次调用`insert()`函数时compact，但是可以通过调用`resize()`方法手动compact
  - `resize(0)`可以调整哈希表的size为最小的合法值 [2,3]

- `sparse_hash_map::iterator` 和 `sparse_hash_map::iterator` 不是一个可变迭代器，因为 `sparse_hash_map::value_type` 和 `dense_hash_map::value_type` 不能被赋值 [2,3]
- `sparse_hash_map` 和 `dense_hash_map` 必须是 `plain old data` [4,2,3]

适用

dense hash map

- 适用于需要快速访问存储在内存中的相对较小的“字典”的应用程序 [2]

sparse hash map

- 适用于需要在内存中存储大型“字典”的应用程序，以及需要这些字典持久化的应用程序 [3]

底层数据结构

- 基于二次探测的 hash table [1]

nark hash map

特点

- 使用两块连续内存 [5]
- 缓存hash value (hash key) [5]
- 删除元素更简单 [5]
  - 直接将末尾元素拷贝到删除元素处
  - 删除元素时插入空闲链表，优先从空闲链表分配（id 不变）

底层数据结构

- 基于开链的hash table [6]

测试

环境

```
$ uname -a
Linux c3-data-ml00.bj 3.10.0-514.16.1.el7.x86_64 #1 SMP Wed Apr 12 15:04:24
UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
...
Model name:            Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz
CPU MHz:               2303.515
...
$ gcc -v
...
Thread model: posix
gcc version 4.8.5 20150623 (Red Hat 4.8.5-11) (GCC)
```

说明

实验在google `item_hash_map.cc` [7] 基础上进行修改，添加支持`nark hash map(gold_hash_map)`，针对Hash Map的不同实现、Key的不同大小，指定迭代次数执行map的相关操作并进行耗时统计，对统计结果进行对比。

不同的**Hash Map实现**说明如下表所示。

名称	说明
sparse_hash_map	google sparse hash map
dense_hash_map	google dense hash map
unordered_map	standard hash_map
map	standard map
gold_hash_map	nark hash map

不同操作说明如下表所示。

操作	说明
map_grow	插入指定迭代次数元素
map_predict/grow	hash map初始化时指定元素数量为指定迭代次数，然后插入指定迭代次数元素
map_replace	替换指定迭代次数元素值
map_fetch_random	随机遍历hash map（元素数量为指定的迭代次数）
map_fetch_sequential	顺序遍历hash map（元素数量为指定的迭代次数）
map_fetch_empty	判断元素是否在map内（元素数量和循环次数为指定的迭代次数）
map_remove	删除所有元素（初始数量为指定的迭代次数）
map_toggle	添加一个元素，紧接着删除该元素（次数指定的迭代次数）
map_iterate	使用迭代器遍历map

迭代次数如下表所示。

Key大小(byte)	迭代次数
4	10000000
8	5000000
16	2500000
256	312500

实验共进行8组，结果参见链接 [8]，示例结果参考附录，统计对应变量去除最大、最小值后的平均值，统计程序 [9] 见附录。

编译

编译命令如下。

```
$ g++ -DHAVE_CONFIG_H -I. -I./src  -I./src -std=c++11 -D_GNU_SOURCE -
I../nark-bone/src -I../nark-hashmap/src -Wall -W -Wwrite-strings -
Woverloaded-virtual -Wshadow -g -O2 -MT time_hash_map-time_hash_map.o -MD -
MP -MF .deps/time_hash_map-time_hash_map.Tpo -c -o time_hash_map-
time_hash_map.o `test -f 'src/time_hash_map.cc' || echo
'./'`src/time_hash_map.cc
```

结果

测试结果说明

key为4byte, map_grow操作，不同hash map实现的耗时对比。

```
 ++++++++++++++++++++4 byte && map_grow++++++++++++++++++++
   'SPARSE_HASH_MAP' :  175.0
    'DENSE_HASH_MAP' :  37.6
'STANDARD HASH_MAP' :  78.9
     'STANDARD MAP' :  457.2
         'NARK MAP' :  43.8
```

测试结果

random fetch(find)

```
++++++++++++++++++++4 byte  && map_fetch_random++++++++++++++++++++
   'SPARSE_HASH_MAP': 145.0
    'DENSE_HASH_MAP': 20.8
 'STANDARD HASH_MAP': 72.5
      'STANDARD MAP': 1431.9
          'NARK MAP': 53.1

++++++++++++++++++++8 byte  && map_fetch_random++++++++++++++++++++
   'SPARSE_HASH_MAP': 141.4
    'DENSE_HASH_MAP': 29.7
 'STANDARD HASH_MAP': 74.0
      'STANDARD MAP': 1414.0
          'NARK MAP': 60.5

++++++++++++++++++++16 byte  && map_fetch_random++++++++++++++++++++
   'SPARSE_HASH_MAP': 147.5
    'DENSE_HASH_MAP': 47.4
 'STANDARD HASH_MAP': 100.9
      'STANDARD MAP': 1207.7
          'NARK MAP': 81.6

++++++++++++++++++++256 byte  && map_fetch_random++++++++++++++++++++
   'SPARSE_HASH_MAP': 183.7
    'DENSE_HASH_MAP': 118.0
 'STANDARD HASH_MAP': 169.5
      'STANDARD MAP': 650.8
          'NARK MAP': 149.7
```

map_remove(erase)

```
++++++++++++++++++++4 byte  && map_remove++++++++++++++++++++
   'SPARSE_HASH_MAP': 49.9
```

```
    'DENSE_HASH_MAP': 9.8
 'STANDARD HASH_MAP': 36.2
      'STANDARD MAP': 231.8
          'NARK MAP': 23.7

++++++++++++++++++++8 byte  && map_remove++++++++++++++++++++
   'SPARSE_HASH_MAP': 54.3
    'DENSE_HASH_MAP': 11.3
 'STANDARD HASH_MAP': 37.8
      'STANDARD MAP': 227.4
          'NARK MAP': 26.3

++++++++++++++++++++16 byte  && map_remove++++++++++++++++++++
   'SPARSE_HASH_MAP': 64.9
    'DENSE_HASH_MAP': 27.3
 'STANDARD HASH_MAP': 44.1
      'STANDARD MAP': 212.1
          'NARK MAP': 34.6

++++++++++++++++++++256 byte  && map_remove++++++++++++++++++++
   'SPARSE_HASH_MAP': 305.1
    'DENSE_HASH_MAP': 239.2
 'STANDARD HASH_MAP': 97.3
      'STANDARD MAP': 182.3
          'NARK MAP': 110.7
```

fetch empty

```
++++++++++++++++++++4 byte  && map_fetch_empty++++++++++++++++++++
   'SPARSE_HASH_MAP': 16.3
    'DENSE_HASH_MAP': 4.2
 'STANDARD HASH_MAP': 15.2
      'STANDARD MAP': 3.9
          'NARK MAP': 13.4

++++++++++++++++++++8 byte  && map_fetch_empty++++++++++++++++++++
   'SPARSE_HASH_MAP': 16.8
    'DENSE_HASH_MAP': 6.4
 'STANDARD HASH_MAP': 16.4
      'STANDARD MAP': 4.3
          'NARK MAP': 15.0

++++++++++++++++++++16 byte  && map_fetch_empty++++++++++++++++++++
   'SPARSE_HASH_MAP': 17.4
    'DENSE_HASH_MAP': 7.2
 'STANDARD HASH_MAP': 22.4
      'STANDARD MAP': 4.3
          'NARK MAP': 20.8
```

```
++++++++++++++++++++256 byte  && map_fetch_empty++++++++++++++++++++
   'SPARSE_HASH_MAP': 39.5
    'DENSE_HASH_MAP': 29.1
 'STANDARD HASH_MAP': 66.0
      'STANDARD MAP': 25.1
          'NARK MAP': 55.1
```

map_fetch_sequential

```
++++++++++++++++++++4 byte  && map_fetch_sequential++++++++++++++++++++
   'SPARSE_HASH_MAP': 41.2
    'DENSE_HASH_MAP': 5.0
 'STANDARD HASH_MAP': 15.6
      'STANDARD MAP': 269.1
          'NARK MAP': 15.2

++++++++++++++++++++8 byte  && map_fetch_sequential++++++++++++++++++++
   'SPARSE_HASH_MAP': 44.7
    'DENSE_HASH_MAP': 7.6
 'STANDARD HASH_MAP': 18.7
      'STANDARD MAP': 266.7
          'NARK MAP': 17.3

++++++++++++++++++++16 byte  && map_fetch_sequential++++++++++++++++++++
   'SPARSE_HASH_MAP': 55.0
    'DENSE_HASH_MAP': 19.2
 'STANDARD HASH_MAP': 24.8
      'STANDARD MAP': 245.5
          'NARK MAP': 23.0

++++++++++++++++++++256 byte  && map_fetch_sequential++++++++++++++++++++
   'SPARSE_HASH_MAP': 163.4
    'DENSE_HASH_MAP': 120.7
 'STANDARD HASH_MAP': 124.6
      'STANDARD MAP': 220.9
          'NARK MAP': 69.7
```

map_grow

```
++++++++++++++++++++4 byte  && map_grow++++++++++++++++++++
   'SPARSE_HASH_MAP': 175.0
    'DENSE_HASH_MAP': 37.6
 'STANDARD HASH_MAP': 78.9
      'STANDARD MAP': 457.2
          'NARK MAP': 43.8

++++++++++++++++++++8 byte  && map_grow++++++++++++++++++++
   'SPARSE_HASH_MAP': 225.0
```

```
    'DENSE_HASH_MAP': 57.2
 'STANDARD HASH_MAP': 81.7
      'STANDARD MAP': 470.8
          'NARK MAP': 62.1

++++++++++++++++++++16 byte  && map_grow++++++++++++++++++++
   'SPARSE_HASH_MAP': 320.1
    'DENSE_HASH_MAP': 103.3
 'STANDARD HASH_MAP': 105.9
      'STANDARD MAP': 400.5
          'NARK MAP': 78.6

++++++++++++++++++++256 byte  && map_grow++++++++++++++++++++
   'SPARSE_HASH_MAP': 1319.9
    'DENSE_HASH_MAP': 598.2
 'STANDARD HASH_MAP': 260.9
      'STANDARD MAP': 298.9
          'NARK MAP': 245.9
```

map_iterate

```
++++++++++++++++++++4 byte  && map_iterate++++++++++++++++++++
   'SPARSE_HASH_MAP': 6.9
    'DENSE_HASH_MAP': 5.2
 'STANDARD HASH_MAP': 3.8
      'STANDARD MAP': 23.0
          'NARK MAP': 1.7

++++++++++++++++++++8 byte  && map_iterate++++++++++++++++++++
   'SPARSE_HASH_MAP': 6.5
    'DENSE_HASH_MAP': 5.5
 'STANDARD HASH_MAP': 3.9
      'STANDARD MAP': 29.2
          'NARK MAP': 2.3

++++++++++++++++++++16 byte  && map_iterate++++++++++++++++++++
   'SPARSE_HASH_MAP': 6.6
    'DENSE_HASH_MAP': 6.8
 'STANDARD HASH_MAP': 6.5
      'STANDARD MAP': 18.3
          'NARK MAP': 2.4

++++++++++++++++++++256 byte  && map_iterate++++++++++++++++++++
   'SPARSE_HASH_MAP': 21.6
    'DENSE_HASH_MAP': 36.7
 'STANDARD HASH_MAP': 13.3
      'STANDARD MAP': 60.5
          'NARK MAP': 11.4
```

map_predict/grow

```
++++++++++++++++++++4 byte  && map_predict/grow++++++++++++++++++++
   'SPARSE_HASH_MAP': 71.2
    'DENSE_HASH_MAP': 16.2
 'STANDARD HASH_MAP': 50.3
      'STANDARD MAP': 450.0
          'NARK MAP': 37.8

++++++++++++++++++++8 byte  && map_predict/grow++++++++++++++++++++
   'SPARSE_HASH_MAP': 117.9
    'DENSE_HASH_MAP': 18.1
 'STANDARD HASH_MAP': 51.2
      'STANDARD MAP': 475.4
          'NARK MAP': 48.4

++++++++++++++++++++16 byte  && map_predict/grow++++++++++++++++++++
   'SPARSE_HASH_MAP': 181.9
    'DENSE_HASH_MAP': 31.3
 'STANDARD HASH_MAP': 63.1
      'STANDARD MAP': 364.3
          'NARK MAP': 65.1

++++++++++++++++++++256 byte  && map_predict/grow++++++++++++++++++++
   'SPARSE_HASH_MAP': 936.3
    'DENSE_HASH_MAP': 282.0
 'STANDARD HASH_MAP': 165.5
      'STANDARD MAP': 301.6
          'NARK MAP': 236.2
```

map_replace

```
++++++++++++++++++++4 byte  && map_replace++++++++++++++++++++
   'SPARSE_HASH_MAP': 28.4
    'DENSE_HASH_MAP': 8.5
 'STANDARD HASH_MAP': 16.2
      'STANDARD MAP': 254.3
          'NARK MAP': 16.8

++++++++++++++++++++8 byte  && map_replace++++++++++++++++++++
   'SPARSE_HASH_MAP': 31.7
    'DENSE_HASH_MAP': 8.3
 'STANDARD HASH_MAP': 15.9
      'STANDARD MAP': 276.5
          'NARK MAP': 20.1

++++++++++++++++++++16 byte  && map_replace++++++++++++++++++++
   'SPARSE_HASH_MAP': 40.9
```

```
    'DENSE_HASH_MAP': 21.3
 'STANDARD HASH_MAP': 22.3
      'STANDARD MAP': 211.3
          'NARK MAP': 30.9

++++++++++++++++++++256 byte  && map_replace++++++++++++++++++++
   'SPARSE_HASH_MAP': 169.7
    'DENSE_HASH_MAP': 137.5
 'STANDARD HASH_MAP': 90.3
      'STANDARD MAP': 221.8
          'NARK MAP': 90.1
```

map_toggle

```
++++++++++++++++++++4 byte  && map_toggle++++++++++++++++++++
   'SPARSE_HASH_MAP': 164.0
    'DENSE_HASH_MAP': 46.8
 'STANDARD HASH_MAP': 75.8
      'STANDARD MAP': 79.8
          'NARK MAP': 35.6

++++++++++++++++++++8 byte  && map_toggle++++++++++++++++++++
   'SPARSE_HASH_MAP': 178.8
    'DENSE_HASH_MAP': 51.4
 'STANDARD HASH_MAP': 79.1
      'STANDARD MAP': 78.0
          'NARK MAP': 43.8

++++++++++++++++++++16 byte  && map_toggle++++++++++++++++++++
   'SPARSE_HASH_MAP': 197.1
    'DENSE_HASH_MAP': 74.7
 'STANDARD HASH_MAP': 95.8
      'STANDARD MAP': 77.9
          'NARK MAP': 67.0

++++++++++++++++++++256 byte  && map_toggle++++++++++++++++++++
   'SPARSE_HASH_MAP': 529.6
    'DENSE_HASH_MAP': 284.9
 'STANDARD HASH_MAP': 260.3
      'STANDARD MAP': 171.5
          'NARK MAP': 145.0
```

参考

- Ref 1
- Ref 2
- Ref 3

- Ref 4
- Ref 5
- Ref 6
- Ref 7
- Ref 8
- Ref 9

附录

耗时统计程序

```
#!/bin/python3

"""
@file analysis.py
@author bovenson
@date 2018-11-29 18:55:15
@desc -
"""

import os
import pathlib
import re

size_list = [4, 8, 16, 256]
map_list = ['SPARSE_HASH_MAP', 'DENSE_HASH_MAP', 'STANDARD HASH_MAP',
'STANDARD MAP', 'NARK MAP']
item_list = ['map_grow', 'map_predict/grow', 'map_replace',
'map_fetch_random', 'map_fetch_sequential',
    'map_fetch_empty', 'map_remove', 'map_toggle', 'map_iterate']
res = [[[[] for _ in item_list] for _ in map_list] for _ in size_list]

def extract_data(data):
    _res = re.findall(r'(\d+\.\d+) ns', data)
    # print(len(_res))
    _cnt = 0
    for _i in range(len(size_list)):
        for _j in range(len(map_list)):
            for _k in range(len(item_list)):
                res[_i][_j][_k].append(float(_res[_cnt]))
                _cnt += 1


file_list = []
log_file_path = './log'
for item in os.listdir(log_file_path):
    item = os.path.join(log_file_path, item)
```

```
    if os.path.isfile(item):
        with open(item) as f:
            extract_data(f.read())

# print(res[0][0][0])
# print(res[-1][-1][-2])

# average
for _i in range(len(size_list)):
    for _j in range(len(map_list)):
        for _k in range(len(item_list)):
            _cur = res[_i][_j][_k]
            res[_i][_j][_k] = (sum(_cur) - min(_cur) - max(_cur)) /
(len(_cur) - 2)

# print(res[0][0][0])
# print(res[-1][-1][-2])

def print_all():
    with open('all.txt', 'w+') as f:
        for _i in range(len(size_list)):
            for _k in range(len(item_list)):
                f.write('\n' + '+' * 20 + str(size_list[_i]) + ' byte ' + '
&& ' + str(item_list[_k]) + '+' * 20)
                for _j in range(len(map_list)):
                    f.write(repr(map_list[_j]).rjust(20) + ': ' +
'{0:.1f}'.format(res[_i][_j][_k]))


# print oper
def print_oper(oper):
    _k = item_list.index(oper)
    _file_name = os.path.join('.', 'analysis', (oper + '.txt').replace('/',
''))
    with open(_file_name, 'w+') as f:
        for _i in range(len(size_list)):
            f.write('\n' + '+' * 20 + str(size_list[_i]) + ' byte ' + ' &&
' + str(item_list[_k]) + '+' * 20 + '\n')
            for _j in range(len(map_list)):
                f.write(repr(map_list[_j]).rjust(20) + ': ' +
'{0:.1f}'.format(res[_i][_j][_k]) + '\n')



for _oper in item_list:
    print_oper(_oper)
```

time_hash_map示例输出

```
======
Linux | **** | 3.10.0-514.16.1.el7.x86_64 | #1 SMP Wed Apr 12 15:04:24 UTC
2017 | x86_64
Average over 10000000 iterations
Current time (GMT): Thu Nov 29 10:19:21 2018

SPARSE_HASH_MAP (4 byte objects, 10000000 iterations):
map_grow              176.2 ns  (23421757 hashes, 43421814 copies)
map_predict/grow       71.6 ns  (10000000 hashes, 30000000 copies)
map_replace            28.1 ns  (10000000 hashes,        0 copies)
map_fetch_random      143.2 ns  (10000000 hashes,        0 copies)
map_fetch_sequential   41.0 ns  (10000000 hashes,        0 copies)
map_fetch_empty        16.0 ns  (       0 hashes,        0 copies)
map_remove             49.9 ns  (10000000 hashes, 10000000 copies)
map_toggle            164.0 ns  (20399999 hashes, 41599996 copies)
map_iterate             7.0 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 280.5ns/insertion
stresshashfunction map_size=256 stride=256: 204.2ns/insertion
stresshashfunction map_size=1024 stride=1: 498.6ns/insertion
stresshashfunction map_size=1024 stride=1024: 509.8ns/insertion

DENSE_HASH_MAP (4 byte objects, 10000000 iterations):
map_grow               36.2 ns  (26777220 hashes, 113886160 copies)
map_predict/grow       16.1 ns  (10000000 hashes, 30000000 copies)
map_replace             8.8 ns  (10000000 hashes,        0 copies)
map_fetch_random       21.1 ns  (10000000 hashes,        0 copies)
map_fetch_sequential    5.0 ns  (10000000 hashes,        0 copies)
map_fetch_empty         4.0 ns  (       0 hashes,        0 copies)
map_remove             10.1 ns  (10000000 hashes, 10000000 copies)
map_toggle             47.0 ns  (20624999 hashes, 64999960 copies)
map_iterate             5.3 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 45.2ns/insertion
stresshashfunction map_size=256 stride=256: 23.0ns/insertion
stresshashfunction map_size=1024 stride=1: 85.3ns/insertion
stresshashfunction map_size=1024 stride=1024: 54.7ns/insertion

STANDARD HASH_MAP (4 byte objects, 10000000 iterations):
map_grow               78.4 ns  (27427396 hashes, 30000000 copies)
map_predict/grow       50.4 ns  (10000000 hashes, 30000000 copies)
map_replace            16.5 ns  (10000000 hashes,        0 copies)
map_fetch_random       72.0 ns  (10000000 hashes,        0 copies)
map_fetch_sequential   15.7 ns  (10000000 hashes,        0 copies)
map_fetch_empty        15.2 ns  (10000000 hashes,        0 copies)
map_remove             36.0 ns  (10000000 hashes,        0 copies)
map_toggle             76.1 ns  (20000000 hashes, 30000000 copies)
map_iterate             3.8 ns  (       0 hashes,        0 copies)
```

```
stresshashfunction map_size=256 stride=1: 52.3ns/insertion
stresshashfunction map_size=256 stride=256: 51.4ns/insertion
stresshashfunction map_size=1024 stride=1: 54.1ns/insertion
stresshashfunction map_size=1024 stride=1024: 55.2ns/insertion

STANDARD MAP (4 byte objects, 10000000 iterations):
map_grow              449.7 ns  (       0 hashes, 10000000 copies)
map_predict/grow      436.5 ns  (       0 hashes, 10000000 copies)
map_replace           251.7 ns  (       0 hashes,        0 copies)
map_fetch_random     1412.9 ns  (       0 hashes,        0 copies)
map_fetch_sequential  267.5 ns  (       0 hashes,        0 copies)
map_fetch_empty         3.9 ns  (       0 hashes,        0 copies)
map_remove            233.2 ns  (       0 hashes,        0 copies)
map_toggle             79.9 ns  (       0 hashes, 10000000 copies)
map_iterate            22.9 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 62.2ns/insertion
stresshashfunction map_size=256 stride=256: 62.4ns/insertion
stresshashfunction map_size=1024 stride=1: 69.8ns/insertion
stresshashfunction map_size=1024 stride=1024: 69.2ns/insertion

NARK MAP (4 byte objects, 10000000 iterations):
map_grow               43.2 ns  (20066444 hashes, 36777215 copies)
map_predict/grow       37.8 ns  (20066444 hashes, 36777215 copies)
map_replace            16.4 ns  (10000000 hashes, 10000000 copies)
map_fetch_random       52.6 ns  (10000000 hashes,        0 copies)
map_fetch_sequential   14.8 ns  (10000000 hashes,        0 copies)
map_fetch_empty        13.7 ns  (10000000 hashes,        0 copies)
map_remove             23.9 ns  (15000000 hashes,  5000000 copies)
map_toggle             35.8 ns  (20000000 hashes, 20000000 copies)
map_iterate             1.8 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 47.6ns/insertion
stresshashfunction map_size=256 stride=256: 47.8ns/insertion
stresshashfunction map_size=1024 stride=1: 40.3ns/insertion
stresshashfunction map_size=1024 stride=1024: 41.4ns/insertion

SPARSE_HASH_MAP (8 byte objects, 5000000 iterations):
map_grow              224.4 ns  (11710870 hashes, 21710924 copies)
map_predict/grow      119.0 ns  ( 5000000 hashes, 15000000 copies)
map_replace            32.2 ns  ( 5000000 hashes,        0 copies)
map_fetch_random      140.8 ns  ( 5000000 hashes,        0 copies)
map_fetch_sequential   45.3 ns  ( 5000000 hashes,        0 copies)
map_fetch_empty        16.4 ns  (       0 hashes,        0 copies)
map_remove             53.6 ns  ( 5000000 hashes,  5000000 copies)
map_toggle            178.1 ns  (10199999 hashes, 20799996 copies)
map_iterate             6.3 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 273.6ns/insertion
```

```
stresshashfunction map_size=256 stride=256: 200.3ns/insertion
stresshashfunction map_size=1024 stride=1: 489.5ns/insertion
stresshashfunction map_size=1024 stride=1024: 505.4ns/insertion

DENSE_HASH_MAP (8 byte objects, 5000000 iterations):
map_grow               52.8 ns  (13388611 hashes, 56943112 copies)
map_predict/grow       18.0 ns  ( 5000000 hashes, 15000000 copies)
map_replace             7.9 ns  ( 5000000 hashes,        0 copies)
map_fetch_random       29.8 ns  ( 5000000 hashes,        0 copies)
map_fetch_sequential    7.6 ns  ( 5000000 hashes,        0 copies)
map_fetch_empty         6.7 ns  (       0 hashes,        0 copies)
map_remove             11.5 ns  ( 5000000 hashes,  5000000 copies)
map_toggle             52.5 ns  (10312499 hashes, 32499960 copies)
map_iterate             5.4 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 45.2ns/insertion
stresshashfunction map_size=256 stride=256: 16.8ns/insertion
stresshashfunction map_size=1024 stride=1: 82.1ns/insertion
stresshashfunction map_size=1024 stride=1024: 52.8ns/insertion

STANDARD HASH_MAP (8 byte objects, 5000000 iterations):
map_grow               86.4 ns  (13582537 hashes, 15000000 copies)
map_predict/grow       51.1 ns  ( 5000000 hashes, 15000000 copies)
map_replace            15.5 ns  ( 5000000 hashes,        0 copies)
map_fetch_random       73.4 ns  ( 5000000 hashes,        0 copies)
map_fetch_sequential   18.8 ns  ( 5000000 hashes,        0 copies)
map_fetch_empty        16.3 ns  ( 5000000 hashes,        0 copies)
map_remove             38.3 ns  ( 5000000 hashes,        0 copies)
map_toggle             81.0 ns  (10000000 hashes, 15000000 copies)
map_iterate             4.0 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 55.0ns/insertion
stresshashfunction map_size=256 stride=256: 59.0ns/insertion
stresshashfunction map_size=1024 stride=1: 60.7ns/insertion
stresshashfunction map_size=1024 stride=1024: 61.3ns/insertion

STANDARD MAP (8 byte objects, 5000000 iterations):
map_grow              474.8 ns  (       0 hashes,  5000000 copies)
map_predict/grow      466.7 ns  (       0 hashes,  5000000 copies)
map_replace           269.3 ns  (       0 hashes,        0 copies)
map_fetch_random     1419.0 ns  (       0 hashes,        0 copies)
map_fetch_sequential  277.7 ns  (       0 hashes,        0 copies)
map_fetch_empty         4.3 ns  (       0 hashes,        0 copies)
map_remove            232.6 ns  (       0 hashes,        0 copies)
map_toggle             78.2 ns  (       0 hashes,  5000000 copies)
map_iterate            28.9 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 63.3ns/insertion
stresshashfunction map_size=256 stride=256: 63.3ns/insertion
```

```
stresshashfunction map_size=1024 stride=1: 76.0ns/insertion
stresshashfunction map_size=1024 stride=1024: 71.0ns/insertion

NARK MAP (8 byte objects, 5000000 iterations):
map_grow               62.1 ns  ( 5000000 hashes, 18388607 copies)
map_predict/grow       48.9 ns  ( 5000000 hashes, 18388607 copies)
map_replace            20.3 ns  ( 5000000 hashes,  5000000 copies)
map_fetch_random       60.0 ns  ( 5000000 hashes,        0 copies)
map_fetch_sequential   16.9 ns  ( 5000000 hashes,        0 copies)
map_fetch_empty        14.7 ns  ( 5000000 hashes,        0 copies)
map_remove             26.5 ns  ( 5000000 hashes,  2500000 copies)
map_toggle             44.2 ns  (10000000 hashes, 10000000 copies)
map_iterate             2.3 ns  (       0 hashes,        0 copies)

stresshashfunction map_size=256 stride=1: 48.5ns/insertion
stresshashfunction map_size=256 stride=256: 51.7ns/insertion
stresshashfunction map_size=1024 stride=1: 41.4ns/insertion
stresshashfunction map_size=1024 stride=1024: 41.7ns/insertion

SPARSE_HASH_MAP (16 byte objects, 2500000 iterations):
map_grow              320.7 ns  ( 5855426 hashes, 10855477 copies)
map_predict/grow      181.2 ns  ( 2500000 hashes,  7500000 copies)
map_replace            41.1 ns  ( 2500000 hashes,        0 copies)
map_fetch_random      146.2 ns  ( 2500000 hashes,        0 copies)
map_fetch_sequential   55.0 ns  ( 2500000 hashes,        0 copies)
map_fetch_empty        17.4 ns  (       0 hashes,        0 copies)
map_remove             63.2 ns  ( 2500000 hashes,  2500000 copies)
map_toggle            197.4 ns  ( 5099999 hashes, 10399996 copies)
map_iterate             6.5 ns  (       0 hashes,        0 copies)

DENSE_HASH_MAP (16 byte objects, 2500000 iterations):
map_grow              102.4 ns  ( 6694306 hashes, 28471584 copies)
map_predict/grow       29.9 ns  ( 2500000 hashes,  7500000 copies)
map_replace            22.0 ns  ( 2500000 hashes,        0 copies)
map_fetch_random       48.6 ns  ( 2500000 hashes,        0 copies)
map_fetch_sequential   19.4 ns  ( 2500000 hashes,        0 copies)
map_fetch_empty         7.2 ns  (       0 hashes,        0 copies)
map_remove             27.7 ns  ( 2500000 hashes,  2500000 copies)
map_toggle             72.3 ns  ( 5156249 hashes, 16249960 copies)
map_iterate             6.7 ns  (       0 hashes,        0 copies)

STANDARD HASH_MAP (16 byte objects, 2500000 iterations):
map_grow              106.5 ns  ( 6726830 hashes,  7500000 copies)
map_predict/grow       61.6 ns  ( 2500000 hashes,  7500000 copies)
map_replace            20.8 ns  ( 2500000 hashes,        0 copies)
map_fetch_random      103.0 ns  ( 2500000 hashes,        0 copies)
map_fetch_sequential   25.1 ns  ( 2500000 hashes,        0 copies)
map_fetch_empty        22.2 ns  ( 2500000 hashes,        0 copies)
map_remove             42.0 ns  ( 2500000 hashes,        0 copies)
```

```
map_toggle             97.1 ns  ( 5000000 hashes,  7500000 copies)
map_iterate             6.5 ns  (       0 hashes,        0 copies)

STANDARD MAP (16 byte objects, 2500000 iterations):
map_grow              395.0 ns  (       0 hashes,  2500000 copies)
map_predict/grow      359.2 ns  (       0 hashes,  2500000 copies)
map_replace           209.0 ns  (       0 hashes,        0 copies)
map_fetch_random     1195.1 ns  (       0 hashes,        0 copies)
map_fetch_sequential  246.8 ns  (       0 hashes,        0 copies)
map_fetch_empty         4.4 ns  (       0 hashes,        0 copies)
map_remove            217.2 ns  (       0 hashes,        0 copies)
map_toggle             78.6 ns  (       0 hashes,  2500000 copies)
map_iterate            17.4 ns  (       0 hashes,        0 copies)

NARK MAP (16 byte objects, 2500000 iterations):
map_grow               80.8 ns  ( 2500000 hashes,  9194303 copies)
map_predict/grow       67.2 ns  ( 2500000 hashes,  9194303 copies)
map_replace            29.1 ns  ( 2500000 hashes,  2500000 copies)
map_fetch_random       78.3 ns  ( 2500000 hashes,        0 copies)
map_fetch_sequential   23.3 ns  ( 2500000 hashes,        0 copies)
map_fetch_empty        20.8 ns  ( 2500000 hashes,        0 copies)
map_remove             35.4 ns  ( 2500000 hashes,  1250000 copies)
map_toggle             67.8 ns  ( 5000000 hashes,  5000000 copies)
map_iterate             2.4 ns  (       0 hashes,        0 copies)

SPARSE_HASH_MAP (256 byte objects, 312500 iterations):
map_grow             1325.5 ns  (  731912 hashes,  1356954 copies)
map_predict/grow      937.7 ns  (  312500 hashes,   937500 copies)
map_replace           156.6 ns  (  312500 hashes,        0 copies)
map_fetch_random      186.5 ns  (  312500 hashes,        0 copies)
map_fetch_sequential  163.5 ns  (  312500 hashes,        0 copies)
map_fetch_empty        40.6 ns  (       0 hashes,        0 copies)
map_remove            318.2 ns  (  312500 hashes,   312500 copies)
map_toggle            516.6 ns  (  637499 hashes,  1299996 copies)
map_iterate            21.1 ns  (       0 hashes,        0 copies)

DENSE_HASH_MAP (256 byte objects, 312500 iterations):
map_grow              605.8 ns  (  836787 hashes,  3558980 copies)
map_predict/grow      262.4 ns  (  312500 hashes,   937500 copies)
map_replace           136.2 ns  (  312500 hashes,        0 copies)
map_fetch_random      117.5 ns  (  312500 hashes,        0 copies)
map_fetch_sequential  121.2 ns  (  312500 hashes,        0 copies)
map_fetch_empty        30.0 ns  (       0 hashes,        0 copies)
map_remove            228.0 ns  (  312500 hashes,   312500 copies)
map_toggle            289.7 ns  (  644531 hashes,  2031240 copies)
map_iterate            35.7 ns  (       0 hashes,        0 copies)

STANDARD HASH_MAP (256 byte objects, 312500 iterations):
map_grow              273.9 ns  (  817789 hashes,   937500 copies)
```

```
map_predict/grow      153.5 ns  (  312500 hashes,   937500 copies)
map_replace            84.3 ns  (  312500 hashes,        0 copies)
map_fetch_random      171.4 ns  (  312500 hashes,        0 copies)
map_fetch_sequential  122.5 ns  (  312500 hashes,        0 copies)
map_fetch_empty        61.9 ns  (  312500 hashes,        0 copies)
map_remove            100.0 ns  (  312500 hashes,        0 copies)
map_toggle            265.7 ns  (  625000 hashes,   937500 copies)
map_iterate            13.0 ns  (       0 hashes,        0 copies)

STANDARD MAP (256 byte objects, 312500 iterations):
map_grow              255.1 ns  (       0 hashes,   312500 copies)
map_predict/grow      308.2 ns  (       0 hashes,   312500 copies)
map_replace           219.9 ns  (       0 hashes,        0 copies)
map_fetch_random      676.3 ns  (       0 hashes,        0 copies)
map_fetch_sequential  222.5 ns  (       0 hashes,        0 copies)
map_fetch_empty        25.5 ns  (       0 hashes,        0 copies)
map_remove            194.3 ns  (       0 hashes,        0 copies)
map_toggle            170.6 ns  (       0 hashes,   312500 copies)
map_iterate            60.9 ns  (       0 hashes,        0 copies)

NARK MAP (256 byte objects, 312500 iterations):
map_grow              260.3 ns  (  312500 hashes,   836787 copies)
map_predict/grow      240.7 ns  (  312500 hashes,   836787 copies)
map_replace            92.2 ns  (  312500 hashes,        0 copies)
map_fetch_random      149.5 ns  (  312500 hashes,        0 copies)
map_fetch_sequential   70.2 ns  (  312500 hashes,        0 copies)
map_fetch_empty        54.1 ns  (  312500 hashes,        0 copies)
map_remove            111.5 ns  (  312500 hashes,   156250 copies)
map_toggle            144.5 ns  (  625000 hashes,   312500 copies)
map_iterate            11.4 ns  (       0 hashes,        0 copies)
```