MergeSort

朴素的MergeSort

单线程，$O(n)$额外空间分配的归并算法在时间上已经战胜了go内置的`sort`，显然归并排序可以通过多核的优势并行排序原始数组的各个部分，使用多协程来优化`MergeSort`的时间性能。

```
(base) qinggniq wangcongdeMacBook-Pro:~/Git/talent-plan/tidb/mergesort [2643]% make bench
go test -bench Benchmark -run xx -count 5 -benchmem
goos: darwin
goarch: amd64
pkg: pingcap/talentplan/tidb/mergesort
BenchmarkMergeSort-8                   1        1573866869 ns/op        134219264 B/op         8 allocs/op
BenchmarkMergeSort-8                   1        1445721649 ns/op        134218528 B/op         5 allocs/op
BenchmarkMergeSort-8                   1        1455964959 ns/op        134218240 B/op         5 allocs/op
BenchmarkMergeSort-8                   1        1498893772 ns/op        134218624 B/op         6 allocs/op
BenchmarkMergeSort-8                   1        1433952825 ns/op        134217760 B/op         3 allocs/op
BenchmarkNormalSort-8                  1        3708450493 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3693819807 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3690332334 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3722189869 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3703189538 ns/op              64 B/op          2 allocs/op
PASS
ok      pingcap/talentplan/tidb/mergesort       31.527s
```

多线程优化时间

由于排序过程是一个**CPU密集型**的任务，所以我们每个核分配数组的一个部分就行了。假如是2核CPU，数组有100个元素，那么可以给前50个数分配一个goroutine，后50个分配一个goroutine。找到分配的边界的任务可以递归完成。

```
(base) qinggniq wangcongdeMacBook-Pro:~/Git/talent-plan/tidb/mergesort [2646]% make bench
go test -bench Benchmark -run xx -count 5 -benchmem
goos: darwin
goarch: amd64
pkg: pingcap/talentplan/tidb/mergesort
BenchmarkMergeSort-8                   2         783953665 ns/op        134222608 B/op        24 allocs/op
BenchmarkMergeSort-8                   2         724002215 ns/op        134222064 B/op        21 allocs/op
BenchmarkMergeSort-8                   2         713229146 ns/op        134220784 B/op        17 allocs/op
BenchmarkMergeSort-8                   2         817278776 ns/op        134220016 B/op        15 allocs/op
BenchmarkMergeSort-8                   2         676654346 ns/op        134221904 B/op        20 allocs/op
BenchmarkNormalSort-8                  1        3547575260 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3592426610 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3536206106 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3552197273 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3566834815 ns/op              64 B/op          2 allocs/op
PASS
ok      pingcap/talentplan/tidb/mergesort       34.741s
```

物理核为4核，逻辑核为8核的情况下，性能比单线程快了大约4倍，符合预期。

优化完时间复杂度之后，优化空间复杂度。

原地归并优化空间

C++中有原地归并算法，网上也有人移植到各种语言，Go语言内置的归并排序也是使用的这种合并算法。算法来源于这篇论文Jyrki Katajainen, Tomi Pasanen, Jukka Teuhola. ``Practical in-place mergesort''. Nordic Journal of Computing, 1996，大致思想就是通过交换来实现原地合并。

把网上http://thomas.baudel.name/Visualisation/VisuTri/inplacestablesort.html的实现移植到了golang上面。

```
(base) qinggniq wangcongdeMacBook-Pro:~/Git/talent-plan/tidb/mergesort [2696:130]% make bench
go test -bench Benchmark -run xx -count 5 -benchmem
goos: darwin
goarch: amd64
pkg: pingcap/talentplan/tidb/mergesort
BenchmarkMergeSort-8                   1        3202679702 ns/op           10272 B/op         38 allocs/op
BenchmarkMergeSort-8                   1        3259973542 ns/op            4832 B/op         21 allocs/op
BenchmarkMergeSort-8                   1        3065777449 ns/op            5216 B/op         25 allocs/op
BenchmarkMergeSort-8                   1        3034175814 ns/op            6720 B/op         28 allocs/op
BenchmarkMergeSort-8                   1        3425476643 ns/op            5600 B/op         26 allocs/op
BenchmarkNormalSort-8                  1        3725894424 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3750742940 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3702547978 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3720333651 ns/op              64 B/op          2 allocs/op
BenchmarkNormalSort-8                  1        3802455128 ns/op              64 B/op          2 allocs/op
PASS
ok      pingcap/talentplan/tidb/mergesort       48.176s
```

可以看见时间上来说和单线程的sort相差不大（因为`merge`过程的时间复杂度从$O(n)$变成了$O(nlog(n))$，所以总的时间复杂度也变成了$O(nlognlogn)$），之前通过多线程优化的尝试就被抵消了，但是空间分配相比于原来的归并算法少了很多。