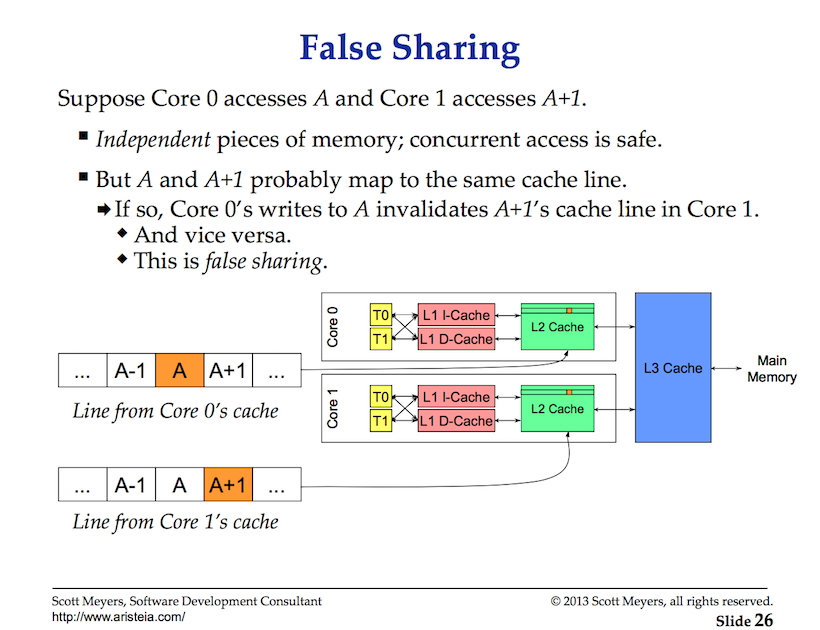
## Data Races

A data race is when two or more goroutines attempt to read and write to the same resource at the same time. Race conditions can create bugs that appear totally random or can never surface as they corrupt data. Atomic functions and mutexes are a way to synchronize the access of shared resources between goroutines.

## Notes

* Goroutines need to be coordinated and synchronized.
* When two or more goroutines attempt to access the same resource, we have a data race.
* Atomic functions and mutexes can provide the support we need.

[](https://github.com/ardanlabs/gotraining/blob/master/topics/go/concurrency/data_race/figure1.png)

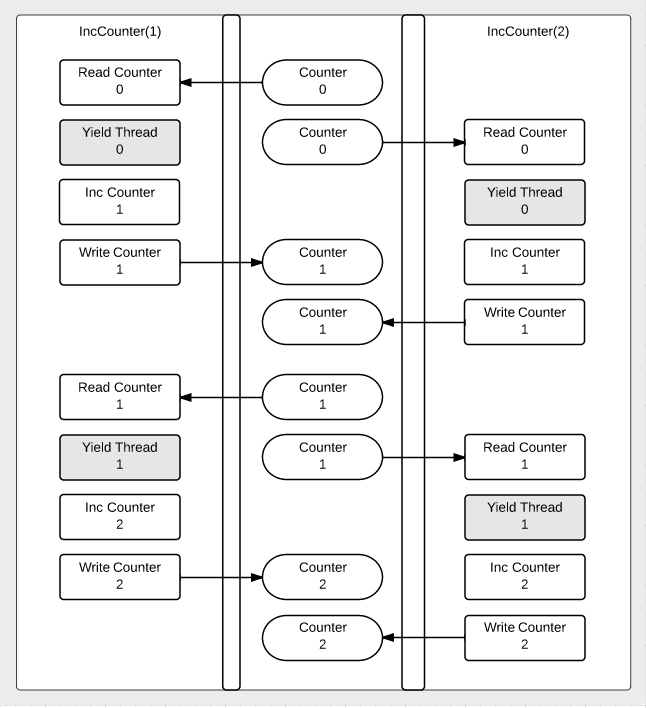
## Cache Coherency and False Sharing Notes

* Thread memory access matters.
* If your algorithm is not scaling look for false sharing problems.

[Detecting Race Conditions With Go](https://www.ardanlabs.com/blog/2013/09/detecting-race-conditions-with-go.html) 

## Diagram

### View of Data Race in Example1.

[](https://github.com/ardanlabs/gotraining/blob/master/topics/go/concurrency/data_race/data_race.png)