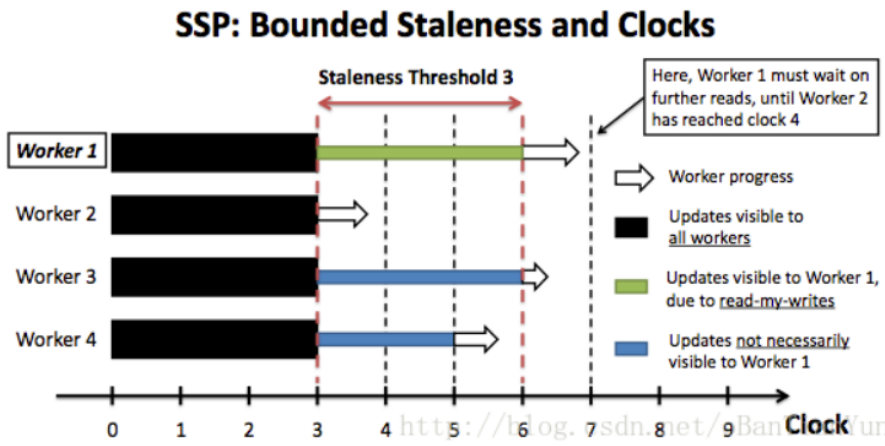
SSP模型



假设现在有p个workers(线程或进程)，需要在一个时钟(如一次迭代)去通过累加的方法去更新一个一致性参数如$x \leftarrow x + u$，每个worker有自己的时钟，并在每次时钟结束时向服务器提交自己的更新，但是每一个worker可能不会立马得到其他worker的更新数据，这就是ssp模型的思想所在，每一个worker查询参数服务器所得数据可能是"旧"的，具体的给定一个用户自定义的时钟阈值s，ssp模型需要保证以下条件成立:

- 快的worker和慢的worker时钟间隔是必须小于等于s，否则快的 worker 必须要停下来等待，这里快的 worker 并不是停下来不做任何事，而是去访问一次parameter server，使本地更新到最新值，这样做既满足时间窗口的条件，又可以更新最新值，减少噪音(图中worker 1在等worker 2);
- 当一个 worker在时钟c时读取parameter server中x时，读到可能是旧值，即本地 cache里面值，而旧值中每个worker更新部分一定包含 c-s-1之前的（图中黑色部分），但也可能包含某些worker的c-s-1之后的更新数据(图中蓝色部分);
- 每一个 worker每一次读 parameter server 时，读到的参数更新部分一定包含自己的上一次提交的那一部分，这就是所谓的"read-my-write(图中绿色部分);
- 每一个 worker本地时钟是 s 整数倍时，去读取一次 parameter server，而不是访问本地 cache;
- 每一个 worker提交更新时会更新本地时钟(编程时需要自己调用Clock操作)，同时更新本地 cache 的值。

以上就是ssp模型基本条件，简单的说，ssp模型要求快的worker一旦突破了用户给定的时间窗口就必须停下来去访问一次parameter server，得到最新的参数，这样慢的worker访问cache一定可以慢慢追上快的worker，另一方面，每一个worker得到parameter server的参数更新的部分的时间窗口一定包含[0,c-s-1]，可能包含的时间窗口[c-s,c+s-1]，这里可能有几种情况，有可能是有些worker很快通过了下一个s整数倍，去访问了一次parameter server，也可能是出现了快的worker去等待慢的worker去访问了parameter server，而时间s整数倍去访问parameter server是为了避免所有的worker一直都没有一个worker去突破时间窗口。