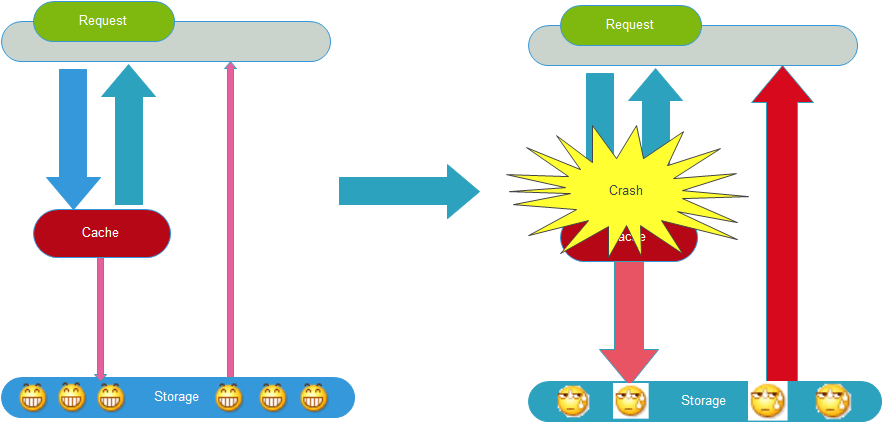
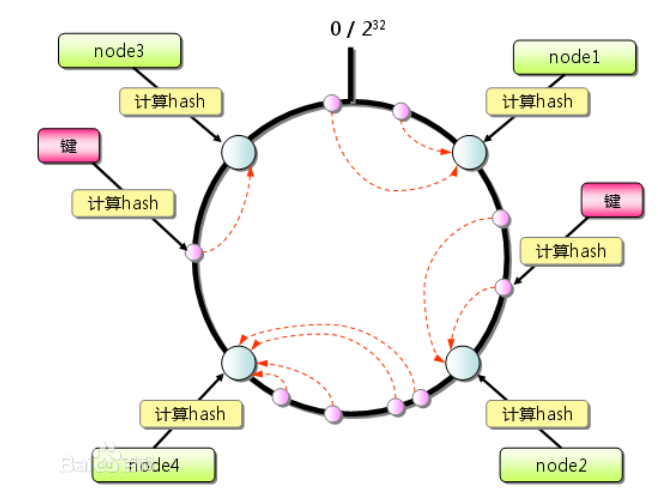
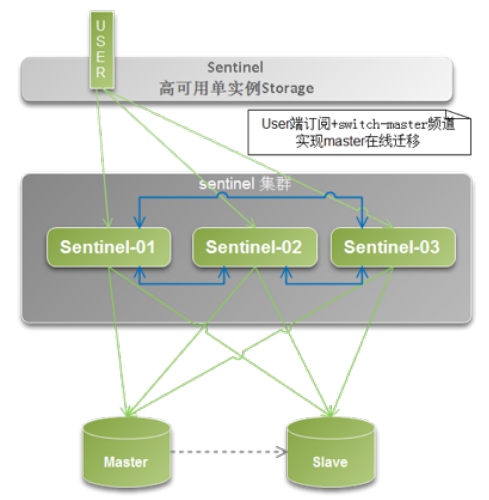
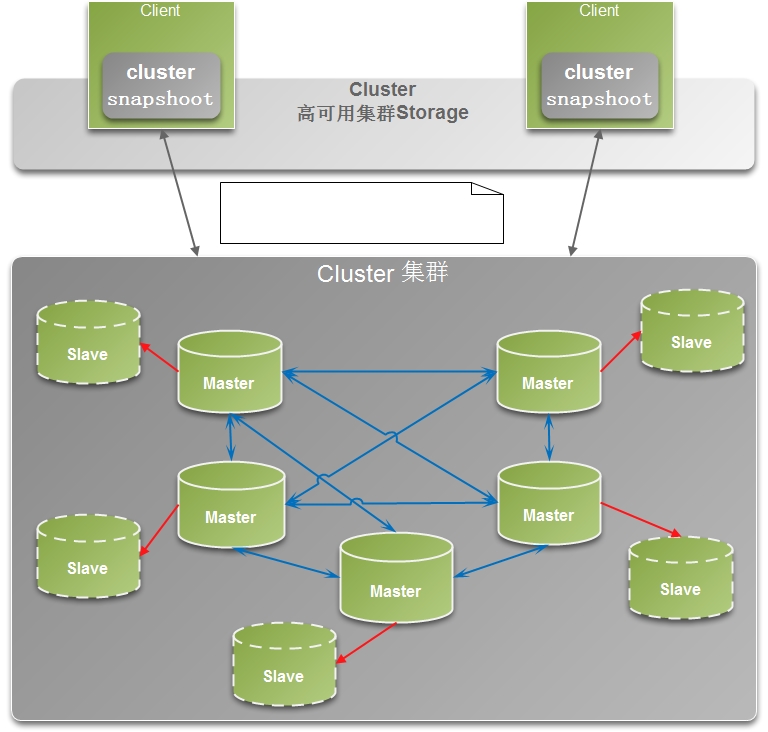
1. 雪崩问题：
   1. <http://carlosfu.iteye.com/blog/2249316>
      1. 
      2. 英文名： stampeding herd, 流量会像奔逃的野牛一样，打向后端.
      3. 现象描述：
         1. 由于Cache层承载着大量请求，有效的保护了Storage层(通常认为此层抗压能力稍弱)，所以Storage的调用量实际很低，所以它很爽。大笑
         2. 如果Cache层由于某些原因(宕机、cache服务挂了或者不响应了)整体crash掉了，也就意味着所有的请求都会达到Storage层，所有Storage的调用量会暴增，所以它有点扛不住了，甚至也会挂掉 哭
      4. 通常来讲可能很久以前storage已经扛不住大量请求了，于是加了cache层，所以雪崩会使得storage压力山大，甚至是挂掉
      5. 如何预防缓存雪崩

* + - 1. 保证缓存集群高可用
         1. memcache的一致性hash



* + - * 1. redis的sentinel和cluster机制：





* + - 1. 依赖隔离组件为后端限流：
         1. netfilx公司提供了一个很牛逼的工具：hystrix



* + - 1. 提前演练
         1. 

1. <https://oom.me/archives/cache-in-java.html>
   1. 加锁方式限制对DB的过量访问。
   2. 通过妥协准确性和实时性能换来更好的性能和用户体验
2. <https://en.wikipedia.org/wiki/Cache_stampede> （翻译）
   1. 又名Dogpile
   2. Cache stampede mitigation
      1. Locking
         1. To prevent multiple simultaneous recomputations of the same value, upon a cache miss a process will attempt to acquire the lock for that cache key and recompute it only if it acquires it.
         2. different implementation options for the case when the lock is not acquired
            1. Wait until the value is recomputed
            2. Return a "not-found" and have the client handle the absence of the value properly
            3. Keep a stale item in the cache to be used while the new value is recomputed
         3. the main drawback
            1. correct implementation of the locking mechanism which also takes care of edge cases including failure of the process acquiring the lock, tuning of a time-to-live for the lock, race-conditions, and so on
      2. External recomputation
         1. This solution moves the recomputation of the cache value from the processes needing it to an external process
         2. triggered in different ways
            1. When the cache value approaches its expiration
            2. Periodically
            3. When a process needing the value encounters a cache miss
         3. This approach requires one more moving part - the external process - that needs to be maintained and monitored. In addition, this solution requires unnatural code separation/duplication and is mostly suited for static cache keys (ie., not dynamically generated, as in the case of keys indexed by an id).
      3. Probabilistic early expiration
         1. With this approach, each process may recompute the cache value before its expiration by making an independent probabilistic decision, where the probability of performing the early recomputation increases as we get closer to the expiration of the value. Since the probabilistic decision is made independently by each process, the effect of the stampede is mitigated as less processes will expire at the same time.

* 1. <http://cseweb.ucsd.edu/~avattani/papers/cache_stampede.pdf> 打印
  2. https://www.percona.com/blog/2010/09/10/cache-miss-storm/

1. <https://en.wikipedia.org/wiki/Network_congestion> Network congestion （更广义的扩展）
2. <https://www.adayinthelifeof.nl/2010/07/29/minimizing-cache-stampedes/>
   1. 原因
      1. for instance the caching server is unavailable
      2. the time to live of an object expires
   2. 解决方式：
      1. 用其它线程来更新Cache，而不是当前的获取线程
      2. 加锁
         1. One might be tempted to change the code above: instead of having a lock with a TTL, you will delete the lock when setting the data in a set() method. This, however, can trigger a race-condition which means that multiple processes still can generate the data (not at the same time, but right after we removed the lock).
3. <https://newspaint.wordpress.com/2013/07/12/avoiding-thundering-herd-in-memcached/>
4. <http://www.ehcache.org/documentation/2.8/recipes/thunderingherd.html>
5. <https://www.quora.com/What-is-a-memcached-stampede>
6. <https://metacpan.org/pod/Cache::Memcached::Turnstile>
7. <http://stackoverflow.com/questions/9858557/avoiding-dog-piling-or-thundering-herd-in-a-memcached-expiration-scenario>
   1. <https://github.com/ericflo/django-newcache> （Python 实现）
8. <http://blog.plataformatec.com.br/2009/09/how-to-avoid-dog-pile-effect-rails-app/> （Rails 实现）
9. <http://caiknife.github.io/blog/2013/11/20/how-to-deal-with-dog-pile-effect/>
10. <http://highscalability.com/blog/2014/7/30/preventing-the-dogpile-effect-problem-and-solution.html>
11. http://highscalability.com/strategy-break-memcache-dog-pile