fanout本质是用空间换时间的模型。

NewsFeed的结构比较简单，而且写的工作量非常大，采用fanout的方式进行NewsFeed的存储，如果有一百万粉丝，就会写一百万次，另外NewsFeed的查询方式比较简单，所以很适合HBase。

HBase非常适合添加各种各样的log日志，记录用户的行为信息。

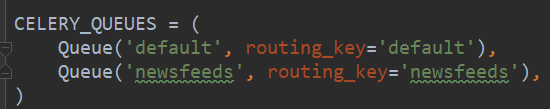
每个数据库都有其特定的应用场景，不能指望一个数据库覆盖所有的情况。

Batch操作的好处就是可以一次性把一批数据都发送到数据库服务上，因为通常情况下Web Server和数据库服务不在一台机器上，所以会有request的请求，Batch只有一次request，如果是多次写入，则会产生多个request。类似Memcached中，如果希望获取多个key，可以使用get\_many()，一次性拿多个key。

Batch操作就类似于一次去超市买10样东西，和10次去超市每次买一样东西，肯定是一次去超市买10样东西更快，一次10样东西结账的时间，和10次每次结账一样东西的时间差不多，但去超市的时间要很长的。

如果明星用户有100万粉丝，在进行fanout操作时，不要把所有的任务都发到一台worker上，而是要把任务进行拆分，每1000个粉丝分一组创建一个子任务，这样可以利用多个worker分别领取任务进行处理。这样会引起另外一个问题，就是异步任务不止fanout，还会有其他的异步任务，甚至优先级更高的异步任务，解决办法有两种：

1. worker进行分组，一部分fanout，一部分执行其他异步任务，利用celery中创建不同的队列，不同的任务投递到不同的队列中，然后worker监听不同的队列。缺点是没法100%利用所有机器的性能，因为fanout的工作量很大，执行其他异步任务的机器可能会经常处于空闲状态。



如果希望高效利用所有worker，可以所有的机器都参与fanout，并且都监听所有的队列：100万个fanout操作，拆分成1000个一组，共有1000组，每组完成假设要10秒，如果有100个worker，最开始这个1000组fanout操作都在MessageQueue中，而不是直接把每10组fanout放到1个worker上。worker监听MessageQueue，一次从MessageQueue拿1个或两个任务(可以配置)，如果worker监听MessageQueue中的多个队列，Celery执行平均分配原则，轮询每个监听队列，这次监听队列A，下一次就好从队列B中取任务。这样高优先级的任务最坏要等待10秒。

如果希望这个最坏等待时间变少，可以减少每组任务数量，例如100个fanout一组。这样带来的新问题，就是任务量会增大10倍。这样MessageQueue可能会很多，会容易崩溃或者性能下降。

如果不希望任务量过多，仍然是1000个fanout一组，这样可以采用翻页的方式来处理这个任务。可以先启动一个task，执行1-10个fanout，完成后把11-1000的fanout生成一个新的任务放回到MessageQueue中，这样MessageQueue中的任务量并没有增加(取了一个任务，然后又放回一个任务)，如果有worker拿到11-1000的任务，就执行11-20，然后把剩下的生成一个新的任务再放回去。执行1-10的fanout会非常快，这样就给了其它优先级更高的任务一个插队的机会，因为Celery会轮询每个监听队列。

总结一下：

方案一：fanout的任务发给一个worker执行，非常慢

方案二：多个worker执行fanout任务，所有其他任务也都在一个队列中。

方案三：多个worker，监听不同的队列

方案四：方案三的基础上，每个worker执行的fanout少一些，增加了任务数量

方案五：方案三的基础上，采用上述类似于翻页的方式。

Celery支持轮询监听不同的队列

Celery可以支持时间限制，@shared\_task中可以设置timelimit

Celery支持定时任务，重试机制(重新放入到消息队列中)。

Celery是多进程模式，一个进程挂了，不影响其它进程。

外部的API，服务是不可信的，所以需要重试机制。

红包自动退款，有些MessageQueue支持延迟任务。Redis不支持延迟任务，Rabbit支持延迟任务。如果支持延迟任务，可以创建一个一天之后回收红包的延迟任务，如果到时红包没有收取，就回收红包，如果收取了，就不执行任务操作。

Photo/Video这类文件存储在对象存储系统例，例如AWS S3。

TweetPhoto对应的文件是不会放在cache中。TweetPhoto中存储的是一个文件地址，Web Server返回给用户的也是一个文件地址。文件是不会在response中的。客户端拿到文件地址后，会根据这个文件地址去请求文件。互联网早期时，加载一张图片很慢，几乎是一行一行加载的。

如果希望加速文件访问速度，可以使用CDN，它类似一个文件的cache。因为文件在S3上，直接访问S3可能在某些地区很慢，用CDN可以预加载一些Photo/Video。请求文件时，首先会到达离你最近的CDN上去查找是否有你要的文件，如果没有就一级一级的去查找，直到去S3上访问。一般大型的互联网社交公司会在全球各个地区都设置CDN结点。

数据库中存储的URL是S3的域名，不能是CDN对应的域名。

如果已发布的URL，就不能轻易去淘汰掉，需要一直兼容这个URL。