System design

1. Design facebook mem cache
2. 设计一个图片分享app， 讨论了怎么实现image feed还有实现拍照获得图片
3. google photos app
4. Design Instagram
5. 设计一个景区签到系统，估计类似Yelp，答主答了Geohash 之类的东西
6. 加面的题目是设计单机版memcached
    a. 这个不是系统设计，我主要说了底层的设计，hashtable, lru，memory management，怎么处理高并发。可能这个题要看面试官怎么问，这个可以问很深我觉得
7. Design a tree of sensor
8. design load balancer
9. Facebook API News Feed
10. 给一个坐标，返回和它距离k的所有建筑
    a. https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=536531
11. 用户过去七天里听的最多的十首歌
    a. 用户每听一首歌，就调用一次API: hit(userID，songID)，设计一个系统，返回用户最近7天听过最多的歌
12. 经常被考的那道搜索自动补充
    a. 设计在线text autocomplete，要求reliability，low latency
13. location based design. 类似yelp
    a. asked to use quad-tree not geo hash)
    b. He said geohash is not accurate, as close points might not have close geo hash - i think guad tree also have the same issue
14. Design a translation/internationalization service for an application/web service like FB.
    a. 设计一个类似网页translator。网页的内容需要根据不同的地区翻译成该地区的语言。
    b. follow up - 对于dynamic的内容，如何实现
15. design leetcode, bottle neck
    a. discuss where is the bottle neck (compiler and runner), how to optimize (stateless, can be scaled out), queue ..., don't forget to mention security.
    b. 设计类似利口的做题网站，每天可以提供若干次竞赛，用户需要注册之后登陆然后提交代码，关键是要快速列出ranking page，这也是我工作的一部分，感觉这两轮系统有点位我定制的感觉。
16. design instagram，面试官说这个是API design 不是system design
    a. 要设计的功能包括Post，view, 小头像怎么存，如何like, 如何显示like (liked by xxx and 50 others), 如何评论，如何显示评论，如果网络不好会有什么影响 (还有一些不记得了）
17. 怎么利用几千台机器做个crawler
    a. 大概根据我对crawler的理解说了，包括DHT，DB设计，用task queue发布新的url，别的server来pull，还有一些估值什么的。
    b. 要求不重复下载，机器间通迅最少

c. 重点是要注意处理URL的hash，不要重复crawl，考察点是Sharding
d. 每台机器bfs去crawl那些倒不是重点
e. 这题我觉得可以用consistent hashing 的思路解决。每台机器都有一个指定的URI，设计一个hash function，能将url和机器的url的hash值平均分布在0~2^64中。每当一个机器获取一个List<url>, 算每个url的hashcode 然后发给指定的机器。所以每个机器中有两个表，一个是hashcode对应每个机器的表，另一个表是已经visited的url的map。至于减少机器之间的通讯，不知道怎么弄。。。
f. 如果在意速度不太在意精确可以试试bloomfilter + 参考mapreduce的partitioner 将类似的url发到指定的计算机上？ lz是怎么答得呢？
g. 一點想法獻醜:
h. 思路從 BFS 延伸，但最關鍵的點就在於只有一個起點為開始
i. 總不能讓所有機器都從這個起點算起，做去重，這樣會有大量的 conflict
j. 所以我的思路是邊 BFS，某些 neighbor link 就自己算，某些就分配到 nearby computers
k. 然後整個 workload 就會用傳染的方式，讓多台機器並行去算
l. 但這個方法，可能會有些機器根本就沒用到的狀況
m. 簡單說就是併發度不夠，明明可以更快卻快不起來
n.
o. 為了改善這個算法
p. 我想如果一開始就能知道由起點展開的完整圖，那就可以最佳化
q. 將 graph 平均拆分成幾塊 sub-graph
r. 每個機器去算若干個 sub-graphs，這樣就能最大化利用到每台機器
s.
t. 所以我的想法是:
u. 一開始先用方法一 (某台機器為起點，傳染方式給周遭) 這步來建圖,
v. 1 billion 的 URL，假設長度 100 字元，每個 graph node 跟 edge，假設要 50 Bytes，那麼一個 node 需要 1 billion * (100 + 50)(bytes) = 150 GB
w. 10k 機器，總 memory 為 2500 GB，要拿其中約 150 GB 存 graph，綽綽有餘
x.
y. 接著用方法二來分配 workload 給所有機器並行下載 HTML
z.
aa. 但有個問題是分配 workload 這步要怎麼切圖，這部分再請大老們分享一下思路，謝謝
bb.

18. 设计怎么通过hashtag找posts，instgram为例。讲了通常分别式系统的设计，怎么build index，sharding之类的
19. design ads 投放系统，每个ads有bid，也有target，还有budget，最高的符合target的bid的ad将被推给user，如果还没有超过budget的话
    a. 比如budget是不是必须要到了0了马上不deliver这个ad了，还是可以用async的方式减少latency但是可能over deliver，另外target这个地方，一种方式是全扫描，另外就是考虑加index，还有就是利用类似decision tree的方式呢对于这个user所在的分类/group直接返回相应的一组ads（这个方法assume read》》write new ads）不过最基本的就是先分析要求，其次high level end to end的design，然后试图deep dive
20. design a dropbox/file-sharing system

21. 日志收集系统
    a. 分布式收集，统一处理， 可以参考一下脸熟的Scribe
22. Type lookahead
    a. 设计search infra to return typeahead (mainly focused on the components after tokenization and query understanding, machine learning for ranking is not considered here)
23. design a photo synchronization app 感觉跟dropbox有点像
24. sys design, given list of places, find all places within circular distances of requested point at x, y.
    a. 这题我回答的是geohash，但对方一直追究geohash角落问题，个人对location理解不深，大家good luck
25. 一个上传/下载系统，支持图片视频之类
26. 设计Facebook News Feed
27. Design infra and storage for posts with same keywords
28. Design search bar
29. Facebook Messanger
30. Design a distributed block storage
31. Design Facebook search。比较常规的系统设计题，无非就是如何scale up，如何shard数据，如何handle突然变得popular的search
32. knight in a infitite chess board，输出从起点到另一点的最短距离，有obstacle。BFS。
33. 做题，不是原题但类似。给个字符串”123456789“，要求插入”+-*“，使得最后计算数字等于某个给定数字，输出所有满足条件的结果。dfs。写到最后他看起来不太高兴：+12*3和12*3是等价的，要求只输出一个。仔细问了时空复杂度及优化。就做了这一题
34. design一个http download lib。题目就这么长，具体需求都是自己去问他。我问了文件大小（MB~GB）、主要面向的使用场景（desktop下电影下图片），多job，允许用底层库比如libcurl。我的思路是用一个thread pool去实现，API提供了两种：callback式的和promise/future式的。最后一直扯到thread的scheduling。。（android 题）
35. 设计一个fb食堂的订餐系统 员工和visitor可以从一个食堂order meals 订单的信息需要persistent。按照某章思路答，问下用户量 设计service和schema 比如meal可以存在no sql里，订单信息和user信息可以存在sql里，怎么scale。我觉得可以参考怎么设计黑车吃（lbs部分除外）
    a. 设计一个非死不可员工自己用的订菜外卖系统， 支持两种用户，客人和参观，然后一些预定分发功能等等
36. 设计 Calenda。我忘了Event object除了time, user, 还有location也很重要。
    a. use cases多了
    b. recurring event
    c. handle event conflit
    d. send user notification ahead of event
37. 实现一个搜索的功能，给定一个query，返回相关的documents。
38. 给一个地点和一个距离，和一大堆places，设计一个service返回这个距离内所有的places，重点focus在如何存储这些places和如何query，以及如何把这么多数据分别存储
39. 设计一个在分布式集群环境下提交job的service

    a. service本身有哪些可能失败的地方，如何解决
    b. service如果提交job失败，然后报错是内存不足，怎么处理
40. nearest point of interest， geohashing
41. system design：design a Facebook scale chess game which you can play 1V1 with your friends, you can play multiple games on multiple devices with multiple friends together at the same time, you can undo your last move before the other player takes the move
    a. 在資料庫裡不需要存棋盤，只需要跟ＣＨＡＴ一樣存對話就好，比方說：
    b. Game 12345
    c. ==========
    d. Msg id 1 | Player A | (0,0) -> (0,1)
    e. Msg id 2 | Player B | (7,0) -> (6,0)
    f. Msg id 3 | Player A | (0,1) -> (0,2)
    g. Msg id 4 | Player A | revert
    h. Msg id 5 | Player A | revert
    i. Msg id 6 | Player B | (6,0) -> (5,0)
    j.
    k. 因為棋盤跟規則是固定的，ＬＯＡＤ ＧＡＭＥ的時後只需要重放操作（無效的操作會被skip比方說msg id 5)，在local/device上cache棋盤根已經播放到哪一步就好。
    l. 剩下的就跟chat system 一樣
    m. 棋盘就是用8*8的2D array存就行了，应该还可以再压缩点，然后感觉就和Design chat app的套路差不多了。
    n. 感觉难点是undo your last move 如何处理race conditions
    o. fb不考OOD的吧，我在想题目说 play multiple games on multiple devices with multiple friends together at the same time, 和device有什么关系，另外是不是要考多线程？
    p. 我也觉得跟design chat 差不多 差别在一个group 只有两个人 但是undo last move 感觉不用care race condition 因为轮流turn 一定自你的turn 才能ondo, 既然棋盘都能存了感觉问题不大?
42. design the newsfeed product, focus on the API design and front-end integration, need to support web browser, smartphone and feature phone, etc. how to optimise for different devices to have a good user experience?
43. 设计100K的web服务器到10K(1000qps)搜索服务器的最佳通讯方式，利用load balancer或者message queue来做，我是第一次onsite system design，直接懵了，恰好自己在做类似的工作，所以直接按照工作经验在找最好的解决方案，快结束的时候才想起来这是面试要多交流，多问问题，但是已经完了，估计国人小哥当时也是懵了，给的数据基本上都没怎么用上
44. design friend list online and offline feature
    a. 简单的思路：
    b. assumption: 针对的use-case是用户的朋友数量在1~1000个的情况，通常fB也限制朋友的最大数。
    c. 1在用户初次登陆的时候：从server读取在线friend的列表，并更新自己的online friend列表。同时，用户的状态由offline变成online也作为statuschange的消息push到server。

d. 2当用户在线使用fB的时候：朋友都有自己的状态，如果一旦发生status change；把status change作为notification push到server，然后转给自己。例如，我正在使用fB，我的页面上显示了目前在线的friend，如果其中某个朋友下线了，他的状态改变会通过notification push到server再转给我，或者直接notify我的client端。
e.
f. 没fB和系统架构的经验，求拍！
g. 你的思路对头。细节问的很细，你的列表肯定存在缓存里，比如redis，application server 可以直接fetch, 比如 schema 的设计，web server and data access server , DB and redis 具体如何同步。可以zoom in 的很细。还有
h. DAU 给我的数据是 1billion 的用户。我这没设计过tps 这么高的系统。
i. 感谢你把系统面试drill into这么细的程度。给我进步思考的信息。
j.
k. 背景：我没有设计和开发分布式系统的经验，目前正在看design data-intesnive application那本书。同时在研究了很多论坛的总结帖子和油管的技术会议视频。我没有太多经验，只能根据看这些资料来讲我的理解了。
l.
m. 1. 缓存： 缓存的使用应该是比较直接，用户登录系统之后，web server通过缓存获取自己全部信息（包括自己所有朋友列表），如果缓存没有，就去route to dB获取用户信息。在得到用户自己的朋友列表之后，再去访问另一个用户状态的缓存服务器（假设存在这样一个用户状态缓存服务器）。假设用户自己和朋友的社交网络graph的地域分布特征为本区域和本国家内的用户的connect较密集，国家之间、洲际之间用户的connect较稀疏；所以这个用户在线状态缓存，主要保存本地区用户状态。同时少量的其他国家、地区的用户在线状态,方便那些具有多国朋友的人查询自己的好友是否在线。
n.
o. 2. Schema的设计： 这里的schema，我理解的是用户是否在线状态的设计，这个应该是in-Mem的schema-less的JSON对象。如果只是为了online/offline这个应用，在设计schema的时候只需要用少数几个attribute，例如： userID，userName，country，Region etc 这样也可以减少内存需求空间。至于脸书这个社交网络平台的其他schema的设计和考虑就比较宽了，设计的内容太多。
p.
q. 3. web server与data access server的同步：不知道需要讨论那些问题。
r.
s. 4. 数据库与缓存（例如Redis）的同步： 如我上面理解的，用户在线状态信息应该全部通过in-mem获取，如果仅仅是确定在线好友状态，不用去访问关系型数据库（RMDB)。如果是其他信息的话，遵从先缓存更新，后RDMS更新的基本原则。当然需要考虑读/写数据的request load来采取具体策略。
t.
u. 5. 1B的用户数量： 还是需要考虑分片（sharding）。我前面假设脸书的用户具有很明显地域分布特征。在分片的时候，主要考虑把本国本区域的人的在线状态信息存储在附近的datacenter。
v.
w. 这五个方面我的理解肯定都比较肤浅。望楼主多多指教。也方便活跃版面热烈讨论技术的氛围。

45. design privacy content 是share 你的post 给specific group you want

46. design聊天软件 不用考虑group chat和offline notification
47. design netflix
    a. 感谢你把系统面试drill into这么细的程度。给我进步思考的信息。
    b.
    c. 背景：我没有设计和开发分布式系统的经验，目前正在看design data-intesnive application那本书。同时在研究了很多论坛的总结帖子和油管的技术会议视频。我没有太多经验，只能根据看这些资料来讲我的理解了。
    d.
    e. 1. 缓存： 缓存的使用应该是比较直接，用户登录系统之后，web server通过缓存获取自己全部信息（包括自己所有朋友列表），如果缓存没有，就去route to dB获取用户信息。在得到用户自己的朋友列表之后，再去访问另一个用户状态的缓存服务器（假设存在这样一个用户状态缓存服务器）。假设用户自己和朋友的社交网络graph的地域分布特征为本区域和本国家内的用户的connect较密集，国家之间、洲际之间用户的connect较稀疏； 所以这个用户在线状态缓存，主要保存本地区用户状态。同时少量的其他国家、地区的用户在线状态,方便那些具有多国朋友的人查询自己的好友是否在线。
    f.
    g. 2. Schema的设计： 这里的schema，我理解的是用户是否在线状态的设计，这个应该是in-Mem的schema-less的JSON对象。如果只是为了online/offline这个应用，在设计schema的时候只需要用少数几个attribute，例如： userID，userName，country，Region etc 这样也可以减少内存需求空间。至于脸书这个社交网络平台的其他schema的设计和考虑就比较宽了，设计的内容太多。
    h.
    i. 3. web server与data access server的同步：不知道需要讨论那些问题。
    j.
    k. 4. 数据库与缓存（例如Redis）的同步： 如我上面理解的，用户在线状态信息应该全部通过in-mem获取，如果仅仅是确定在线好友状态，不用去访问关系型数据库（RMDB)。如果是其他信息的话，遵从先缓存更新，后RDMS更新的基本原则。当然需要考虑读/写数据的request load来采取具体策略。
    l.
    m. 5. 1B的用户数量： 还是需要考虑分片（sharding）。我前面假设脸书的用户具有很明显地域分布特征。在分片的时候，主要考虑把本国本区域的人的在线状态信息存储在附近的datacenter。
    n.
    o. 这五个方面我的理解肯定都比较肤浅。望楼主多多指教。也方便活跃版面热烈讨论技术的氛围。

Design 总结：
https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=537948
Design messaging system ---- 我觉得这个偏backend, 因为数据量比较大，db设计也比较复杂，感觉后端很有考点。
Design instagram --- 这个我觉得像API design， 因为我觉得和news feed很像。。。
Design news feed API --- 这个肯定是API咯，

Design POI, Design typeahead --- 这两个都用比较特别的data structure来存储数据 （quadtree/geohash, trie), 所以应该偏后端
Design web-crawler --- 这个应该也是后端， 因为本身就不是面对用户的产品。
Design memcache --- 这个应该也是后端
Design search system --- 这个像API， 因为前端蛮多use case的， 觉得API有些讲头
Design internatlionalization service --- 这个也是API， db没啥可讲。。。。
Design leetcode ---？？？其实不知道这个题的requirement....
设计一个系统, 返回用户最近7天听过最多的歌 --- 这个也像是backend, 因为use case比较单一， 难点在后端
设计一个fb食堂的订餐系统 ---？？？？不知道。。。。

Behavior:

最成功的项目；有跟别人有conflict吗；最不喜欢跟什么样的人work；最喜欢组里的什么人，为什么；有什么失败吗，学到了啥；

个人喜欢套 “举个例子”➕“所以我学到了blah blah”这个模板，建议楼主去看看Dan Croitor的油管，讲得很详细

1) dive deep into most recent project 2) most intestering debug/bug 3) describe most favorite day in your current company（呵呵） 4) how to work across team 5) give me an example you push back 6) example you regret your design

Culture fit 这轮真的非常重要。 千万要重视起来。 如果是面E5。 一定要尽力往怎么lead project 上靠。 core values 也一定要记牢。

和同事conflict啊，最proud的project啊之类的

e5 one system design
E6，两轮设计

谈经验 谈leadership 谈项目 谈各种soft skills

HR mentioned:

1. 1. Show ***leadership*** in both ppl management and Project scope, in both experience and system design round
2. 2. Clean code with test cases
3. 3. System Design 里一定要***主动drive***这个chat 面试官会很希望你多问多想 他会帮助你narrow down这个题目的components
4. 4. 在system面试里有机会可以dive deep into your domain knowledge. 如果你可以show一些面试官感兴趣但是maybe也没有很懂的东西 很有利于定级

NYC Onsite:
Timeline: coding1 + product desing + lunch + background + coding2 + coding3

常规behavior问题，聊proud的项目（问得很细，比如metrics），如何处理和cowoker的冲突，等等。这轮并不像recruiter说的会有一题coding。

依然不知道怎么答比较好，最后面试官说：你说了太多detail，我没看出你是怎么从这些经验中学习的，感觉凉了。全称我采用了一个策略：我承认自己虽然目前情商不行，但愿意学习，最后问了他好多这个方面的问题，问他怎么学习这方面的能力的。他啪啪啪地记下来，感觉我的策略奏效了？

她问你有没有过，完成了一件事情后有如释重负的感觉。楼主答的是most challenging，但这里她可能更想要我为什么觉得它很难，以及我的态度。

和同事老板conflict怎么样，同事给你feedback有啥受益的么，老板好么怎么变得更好么，好员工有哪些品质？这个品质你有吗，举个例子啊。这是亚麻学的面试么，脸书也走这一套了。而且要求和亚麻一样，例子要详细，技术细节上要面试官听的懂，而且故事也要make sense，要面试官也同意你的做法是对的。连说45分钟，很累。

why facebook

one project that you are proud of/Team project/Disagreements/Constructive suggestions from manager 基本是地里常见的

bq 问了most proud project，project business value， internship 喜欢哪点 学到什么， team conflict

常规bq，比较难的大概就是收到最有建设性的feedback是什么

第一轮：bq。能记住的问题有：

1） 介绍骄傲的project。遇到了什么问题。
2） 怎么处理和manager的矛盾
3） 你有role model么？是谁？你觉得你和她的差距是什么？

问简历并且说的很具体。你收到的manager 的最差的feedback是什么。你最喜欢的教授和最讨厌的教授。bq答的不是很好。

bq记得的有先说一个team work的project，然后讲有没有和队友产生分歧以及解决方法，然后最proud的project，然后在team中有没有其他队员之间有冲突或者不engage你是怎么解决的，还有最difficult的队友

希望改进fb哪一点

最proud的项目；收到的positive的feedback是什么；和其他team合作的经历；直接和客户对需求的经历；和别人有分歧的经历；和别人有分歧但其实错在自己的经历；不太agree with的经理的管理方式

Tell me about one of your successful projects.
What would you have done differently in this project?
Tell me the type of person you don't want to work with.
利口刷tag，高频从上到下就够了，我看了之前其他同学的面经总结，高频题都一样，低频的各不相同，低频碰上原题的概率极低，尤其是uday，几十个人至少也得几十个面试官。所以高频准备好好的就行了。

BQ: 最骄傲的project，为啥骄傲，如何convince others，如果能再做一遍这个project，你会改进什么地方

fav project / biggest challenge as in technology, communication, scale, people, etc / disagreement with manager, peers, ppl from other teams, how to solve at the end / what to expect at next job / least fav in work

bq各种least问题，least internship， least person you work with,

https://www.1point3acres.com/bbs/thread-195416-1-1.html

经验：
结合我最近看design的资料，给楼主一点建议吧。

1 着重在thinking process，不要想45分钟给一个完美的解决方案

2 条理要清晰，有big picture，有data modeling，有data flow

3 多从customer角度来考虑，用户要什么知道了，你才好决定你的设计中的取舍

4 不要一开始就挖大坑，从MVP开始，最后时间足够才加新的feature

5 从最开始简单的架构开始迭代，分析负载上去了之后的bottle neck在哪儿，你要怎么优化。这时候再带入一点customer最关心的是哪里，可以做哪些tradeoff。你做这个技术选型的原因是什么，因为技术mature还是没有spof

6 最后一定要review一下系统的scalability，负载x10的话，你的架构是不是线性的加资源进去就好了？

7 架构确定好了之后，最好来个预估要达到这个性能指标，我们需要多少机器以及怎么部署

8 别忘了提一嘴logging monitoring

9 扯一扯service oriented architecture也是极好的

10 把qconf 2012里scaling Pinterest from 0 to billions看几遍，我看了那么多资料里面，最有启发的一篇

总的来说，我不太建议一上去就摆一个高大上的架构，从小到大循序渐进的来，你就可以展现你更多的能力，不光有breadth还有depth。系统设计都是相通的，碰到没见过没准备的不要蒙蔽，毕竟要考察的是思维过程，不是要求你设计一个完美解决方案。

我自己的系统设计套路：
1. 讨论用户是谁
2. 根据用户讨论feature
3. 问一下系统需要handle 的traffic, 问问需不需要进行计算。 面了8次系统设计，只有roblox 要求计算。其他都不要。。。
4. 根据feature讨论系统需要存储和serve哪些data, 这些data用什么存， 讨论sql/nosql/cache/object storage/hdfs 取舍， 巴拉巴拉。。。
5. 根据数据， 设计service。 画图。
6. work through一个use case, 把所有service连起来， 同时修改刚才画好的图。 比如 做uber eats, 讨论用户要order 一个食物，到餐馆接到订单， 到司机接到订单。。。。
7. 讨论use case细节， 比如 uber eats司机进入某个区域怎么识别啊， cache里怎么存啊。面试官全程都会drive你的design的， 不会丢你在那里自言自语。
8. 面试官会问， 某些环节挂掉了，怎么处理。 无非就是1. 要么replica， master slave, active-passive 或者 2.周期存snapshot 在磁盘上，然后存action log... 挂了可以重新恢复。。。
9. 一些环节怎么scale... multi instance, partition 这些呗。。 偶尔说说service mesh...

onsite面试经验：
- 我面的8次系统设计体验都很好， 面试官会drive design全程， 他会不停的问你小问题带你走。当然他也会根据你的设计不停的提出小问题。 交流交流交流啦～
- 有的时候面试官会质疑你的设计。 此时有两种解法， 1. 解释自己为什么这么做， 让面试官认同你的做法 2. 想一想是不是哪里做的不对， 面试官是不是再给你hint，要换一种设计。 具体情况具体分析啦。。。
- 我至少3次面试，解释了consistent hash 和 virtual node 怎么回事。。 不懂得朋友， 学学看哈。。
- 每个公司的系统设计面试题都蛮固定的。 提前好好刷以下面经，准备准备。

背景：我没有设计和开发分布式系统的经验，目前正在看design data-intesnive application那本书。同时在研究了很多论坛的总结帖子和油管的技术会议视频。我没有太多经验，只能根据看这些资料来讲我的理解了。

1. 缓存： 缓存的使用应该是比较直接，用户登录系统之后，web server通过缓存获取自己全部信息（包括自己所有朋友列表）， 如果缓存没有，就去route to dB获取用户信息。在得到用户自己的朋友列表之后，再去访问另一个用户状态的缓存服务器（假设存在这样一个用户状态缓存服务器）。假设用户自己和朋友的社交网络graph的地域分布特征为本区域和本国家内的用户的connect较密集，国家之间、洲际之间用户的connect较稀疏； 所以这个用户在线状态缓存，主要保存本地区用户状态。同时少量的其他国家、地区的用户在线状态,方便那些具有多国朋友的人查询自己的好友是否在线。

2. Schema的设计： 这里的schema，我理解的是用户是否在线状态的设计，这个应该是in-Mem的schema-less的JSON对象。如果只是为了online/offline这个应用，在设计schema的时候只需要用少数几个attribute，例如： userID，userName，country，Region etc 这样也可以减少内存需求空间。至于脸书这个社交网络平台的其他schema的设计和考虑就比较宽了， 设计的内容太多。

3. web server与data access server的同步：不知道需要讨论那些问题。

4. 数据库与缓存（例如Redis）的同步： 如我上面理解的，用户在线状态信息应该全部通过in-mem获取， 如果仅仅是确定在线好友状态，不用去访问关系型数据库（RMDB)。如果是其他信息的话，遵从先缓存更新，后RDMS更新的基本原则。当然需要考虑读/写数据的request load来采取具体策略。

5. 1B的用户数量： 还是需要考虑分片（sharding）。我前面假设脸书的用户具有很明显地域分布特征。在分片的时候，主要考虑把本国本区域的人的在线状态信息存储在附近的datacenter。

这五个方面我的理解肯定都比较肤浅。望楼主多多指教。也方便活跃版面热烈讨论技术的氛围。