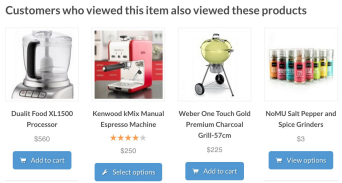
Systems Design Example:

Product CoOccur

Counting popular product-pairs

(from primer doc)



1. Amazon/Walmart/Alibaba (AWA) need to compute 'related products' for all products so their users can explore and buy new products. AWA would like to use collaborative filtering (aka 'wisdom of crowds') from their website logs of user views of products. That is, each time a user views a set of products, those products are said to co-occur. By computing product pairs and their co-occurrence frequency (or **co-occur count)** across all users, AWA can compute related products for their catalog.

2. Data input: AWA's product catalog is 1 billion items. Each product record is ~1MB with a product description and image. AWA has 10 billion product views each week, from 1 billion users. Each log record stores <userID, productID, viewID, viewtime>.

3. Your mission is to design an efficient system to compute co-occur counts on *Sundays* from weekly logs and produce a CoOccurCount table <productID, productID, count>
    - AWA's data quality magicians recommend (a) retaining only the top billion popular pairs, and (b) dropping product pairs with co-occur counts less than million. Also, assume users view ten products on average (**UserSession** assumption).
    - For simplicity, LogOfViews is stored sorted by <userID, productID>. You can sequentially scan the log and produce co-occurring product pairs for each user. In other words, (p_i, p_j) if a user viewed p_i and p_j. This "stream" of tuples (TempCoOccur) may then be (a) stored on disk or (b) discarded after updating any data structures.

Systems Design Example:

Product CoOccur

Pre-design

	Size	Why?
ProductId	4 bytes	1 Billion products ⇒ Need at least 30 bits (2^30 ~= 1 Billion) to represent each product uniquely. So use 4 bytes.
UserID	4 bytes	"
LogOfViewsID	8 bytes	10 Billion product views.
Product	1 PB	1 Billion products of 1 MB each
Users	Unknown	
LogOfViews	240 GB	Each record is <userID, productID, viewID, viewtime>. Assume: we use 8 bytes for viewTime. So that's 24 bytes per record. 10 Billion*24 bytes = 240 GBs.
CoOccur	12 GB	The output should be <productID, productID, count> for the co-occur counts. That is, 12 bytes per record (4 + 4 + 4 for the two productIDs and 4 bytes for count). To keep top billion product pairs (as recommended by AWA data quality), you need 1 billion * 12 bytes = 12 GBs.
TempCoOccur	10^9 * 12 GB	To count all product pairs as we scan input, we may need 1 billion*1 billion (10^18) counters (10^9 * 12 GB of storage).
TempCoOccur (with UserSession assumption, of ~10 views/user)	800 GB	# product pairs produced: 1 billion users * 10^2 = 100 billion Size @8 bytes/record = 800 GBs.

Systems Design Example:

Product CoOccur

Managing RAM/Disk

	Keep table in RAM? Size?	Sequentially scan from disk? If so, how many disk pages?
Products	Not needed for problem	-
Users	Not needed for problem	-
LogOfViews	No, 240 GB You need to only scan the per-user records once. No need for random access.	Yes ~4 million disk blocks (recall: 64KB per page, disk block)
CoOccur	Yes, 12 GB. Prefer random access.	No, Keep in RAM. (Flush later to disk, if necessary.)
TempCoOccur	No. Worst-case: 1 billion * 1 billion counters. Size = 10^18*12 bytes	Yes, must leave on disk. Worst case: 12 * 10^18/64KB pages (= 18.75 * 10^13 pages)
TempCoOccur (with UserSession assumption)	No. 800 GBs	Yes, must leave on disk. # pages: 800 GBs/64 KB ~= 12.5 million

Systems Design Example:

Product CoOccur

Design #2

Design #2: With 1 machine, Analyze with UserSession assumption.

Design 2

1. Scan LogOfviews. For each user, append <p_i, p_j> to a log TempCoOccurLog if the user has viewed product p_i and p_j. (i.e., produce per-user co-occur product pair)
2. Externally sort TempCoOccurLog on disk, so identical product pairs are adjacent to each other in the sorted file
3. Scan sorted TempCoOccurLog. This With a single pass, you can count co-occur pairs. Drop co-occur pairs with < 1 million.

Systems Design Example:

Product CoOccur

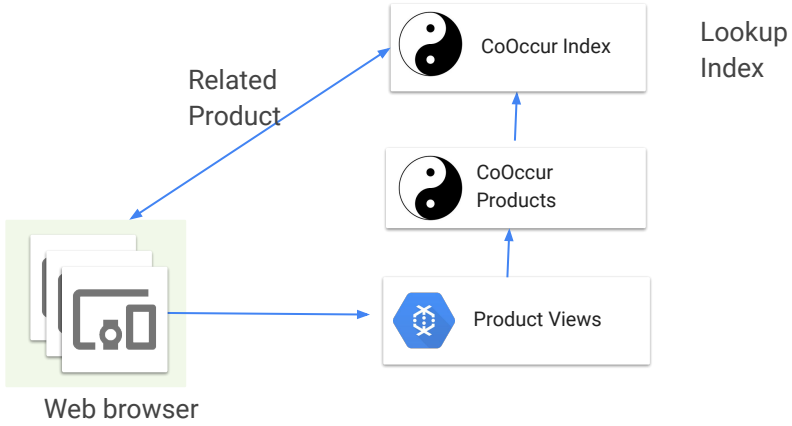
Design #2

Design #2: With 1 machine, Analyze with UserSession assumption.

Steps	Cost (time)	Why?
Scan LogOfViews Append <p_i, p_j> to TempCoOccurLog	~2400 secs ~8000 secs	240GB @100 MB/sec 800 GB @100 MB/sec
Externally sort TempCoOccurLog on disk (Assume sort cost is ~2N, where N is number of pages for table and B is number of buffers, and B ~~ N)	~16,000 secs	IO cost is (appx) 2 * (1 seek + scan cost for 12.5 million pages* 64 KB/per page) = 2* scan cost of 800 GBs. That is, 16000 secs (2*800 GB @100 MB/sec). Assume TempCoOccurLog (and runs) are stored sequentially.
Scan TempCoOccurLog (sorted) and keep counts in CoOccur	~8000 secs	800 GB @100 MB/sec

Data Systems Design Example:

Product CoOccur



Systems Design Example:

Product CoOccur

B+ tree index

Evaluate the cost of lookups in a clustered B+ tree, clustered on productId we look up. How many IO lookups can we expect if we had 1 GB of RAM for the index?

Recall: Let ‘d’ be the degree of the B+ tree nodes, and ‘F’ be fill-factor. Leaf nodes have between d and 2d keys.

Systems Design Example:

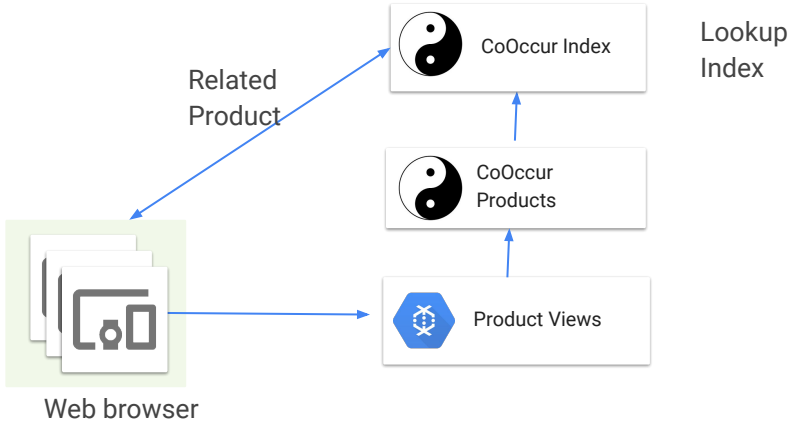
Product CoOccur

B+ tree index

CoOccur # of data records	1 Billion	Given
Index page size	64KB	Given
Number of pages you can fit with 1 GB RAM	~16k pages	1 GB/64KB
How large is d?	~2730	4 bytes for productId + 8 bytes for pointers @ 64KB/page 2d * 4 + (2d+1)*8 <= 64k ⇒ d ~= 2730
Avg number of index records per page, with F=3/4	4000	¾ of 2d ~= 4000
Number of index pages to index (N)	250,000	1 billion records/4000
Root node (level 0)	1 page	Has ~4000 pointers
# Pages at level 1, 2	4000, 4000^2	At level n, has 4000^n pages
# of levels in B+ tree to index 1 billion data records	Root + 2	4000 <= 250,000 4000 + 4000^2 >= 250,000
Number of IOs to get data record for query=urlID	2	Assume root & level1 can be in RAM (Level2 needs 4000^2 pages >> 16k pages in RAM) # IOs: 1 for Level 2, 1 for data record

Data Systems Design Example:

Product CoOccur



Data Systems Design Example:

Bigger Product CoOccur

Problem so far

- AWA's product catalog is 1 billion items. AWA has 10 billion product views each week, from 1 billion users. Each log record stores <userID, productID, viewID, viewtime>

Consider 1000x Bigger problem!

- Product catalog is 1 trillion items. AWA has 10 billion product views. Rest stays same

⇒ What changes?

Systems Design Example:

Product CoOccur

Pre-design

	Size	Why?
ProductId	8 bytes	1 trillion products ⇒ Need at least 40 bits (2^40 ~= 1 Billion) to represent each product uniquely. So use 8 bytes.
UserID	4 bytes	"
LogOfViewsID	8 bytes	10 Billion product views.
Product	1000 PB	1 Trillion products of 1 MB each
Users	Unknown	
LogOfViews	280 GB	Each record is <userID, productID, viewID, viewtime>. Assume: we use 8 bytes for viewTime. So that's 28 bytes per record. 10 Billion*28 bytes = 280 GBs.
CoOccur	20 GBs	The output should be <productID, productID, count> for the co-occur counts. That is, 20 bytes per record (8 + 8 + 4 for the two productIDs and 4 bytes for count). To keep top billion product pairs (as recommended by AWA data quality), you need 1 billion * 20 bytes = 20 GBs.
TempCoOccur	10^24 counters	To count all product pairs as we scan input, we may need 1 trillion*1 trillion (10^24) counters .
TempCoOccur (with UserSession assumption, of ~10 views/user)	1600 GB	# product pairs produced: 1 billion users * 10^2 = 100 billion Size @16 bytes/record = 1600 GBs.

Data Systems Design

Popular Systems design pattern

1. Efficiently compute ‘batch’ of data (sort, hash, count)
2. Build Lookup index on result (b+ tree, hash table)
3. For ‘streaming’ data, update with ‘micro batches’

Popular problems

1. Related videos (youtube), people (Facebook), pages (web)
2. Security threats, malware (security), correlation analysis