1. Question 1

Consider the “two-tiered bin packin” algorithm described in the paper, and the “greedy extended” algorithm described in the paper. What is one advantage of the two-tiered bin packing algorithm over the greedy extended algorithm? What is one disadvantage?

Answer
The short answer is that the integer programming solution gives a better allocation with a higher cost of computing that allocation, while the greedy gives sub-optimal solutions with a lower computational cost.

The integer programming solution is optimal, i.e., it finds the allocation of tuples to partitions that minimizes the overall transmission cost, while keeping the load on each partition very close to the average load. Nevertheless, the number of all possible assignments of tuples and blocks to partitions is $P(H+C)$; where $P$ is the number of partitions, $H$ is the number of hot tuples, and $C$ is the number of cold blocks. While an efficient integer programming solver will have to examine much less assignments than $P(H+C)$, it will still have to run for an exponential number of steps; here we have a 0-1 integer linear program (ILP) and 0-1 ILP is NP-complete.

The greedy on the other hand will generate an allocation that may not be optimal (mostly in terms of the transmission cost), but the time to compute that allocation will be linear to $H+C$.

2. Question 2

Consider a blogging application. Each user can create one or more blogs, and add postings to a blog over time. Some blogs turn out to be significantly more popular than others, but all blogs have some readers. When a new posting is added to a blog, it may be read multiple times, but once it becomes older, it will almost never be read. Imagine we are using E-store with the “first fit” algorithm, and we are trying to decide between two partitioning schemes. In option 1, we partition the data by blog, so that all the postings of a given blog are stored in the same partition. In option 2, we partition the data by posting, so that the postings of a given blog may be stored in separate tuples. Which option is likely to be better? Why?

Answer
Here, there is no short answer and no clear winner.

There are three main factors that one must consider when choosing between the two options:
(1) **How well-balanced is the allocation:** Partitioning by posting under First-Fit is more likely to give a more well-balanced allocation compared to partitioning by blog. The intuition here is the following: Imagine you have already packed your stuff into boxes (blogs) and you want to fit those boxes into large containers (partitions). We care only about the weight and space is not an issue. You start placing the boxes into containers starting from the heaviest box (weight is based on the stuff inside the box). You place boxes into a container until you reach the weight capacity of a container (each container has the same capacity). (That is, if the next box to place will end up in exceeding the capacity of the container, you place it in the next container.) Your second option (partitioning by posting), is to place your stuff (postings) in the containers without placing them in a box first. Again, you place your stuff to containers starting from the heaviest item. This second option is more likely to end up in containers having weights closer to the average. Note also that both options are likely to have a very high transmission cost, since they use First-Fit.

(2) **Exploiting CPU caches:** In both options, the first and last partitions are likely to have a very similar sum of loads (ideally equal to the average). However, in partitioning by posting, the load in the first partition(s) comes from the hottest postings and those postings require far less memory to be stored compared to having the same load from thousands of cold postings. Therefore, we can expect a much higher likelihood of finding something in the cache when trying to serve a request in the first partition(s).

(3) **How stable is the partitioning generated:** Partitioning by blog seems more likely to generate a more “stable” partitioning. That is, a partitioning that will not cause E-Monitor to trigger a re-partitioning phase so often. In case of partitioning by posting, once the hottest postings become “old” (and cold), we will have very few postings allocated in the first partitions; and those postings will have a very low sum of loads. In case of partitioning by blog, assuming that a new posting (after the re-partitioning phase) is assigned to the partition of the posting’s blog, it is more likely that the partitions will keep having loads close to the average. (This may also be the case because hot postings usually come from popular blogs.) In addition, when partitioning by posting, a few hot tuples may be much more popular than what E-Monitor estimated and we may end up having a huge load on a few partitions. In summary, how stable a partitioning is depends on a number of factors and statistics. Overall, partitioning by blog seems to give a more stable partitioning in our scenario.
3. Question 3

Consider a social networking application. This application has two tables: Users, which lists individual users and is keyed by userid, and Friends, which lists pairs of users that are friends with each other. Thus, Users might contain a tuple with key “Mary” and another tuple with Key “Bob”, and if they are friends, the Friends table contains a tuple with key (“Mary”, “Bob”). Would E-store be a useful database system for this application? Why or why not?

Answer

The problem here is that the schema in our toy DB of Users and Friends is not a tree-schema. It is not straightforward how to adjust the mechanisms proposed in the paper to cover our scenario.

4. Grades

Look for the following codes in your homework evaluation:

• Q1: Not mentioning the tradeoff in the short answer for question 1 (-0.5).
• Q2a: Not mentioning factor 1 in question 2.
• Q2b: Not mentioning factor 2 in question 2.
• Q2c: Not mentioning factor 3 in question 2.
• Q3: Not mentioning the problem with the schema in question 3 (-0.5).

For question 2, mentioning one (or more) of the three factors gives you the full score for that question. Otherwise, you lose 0.5. Many of you have pointed out several very interesting factors related to the scenarios discussed in the three questions that we did not cover here. Still, the main factors are the ones we discussed above.