Comments:

Reviewer #2: The submission initiates the algorithmic study of data locality aware virtual
cluster embeddings on datacenter topologies. Namely, it formalizes the virtual cluster embedding as an optimization problem, and extends it with data locality, an aspect that was ignored by existing algorithms, even though it is crucial to the performance of cloud applications such as MapReduce.

Formally, the authors model the problem as follows:

* The physical network (also called substrate network) forms a tree,
whose leaves are servers and internal nodes are routers.
Each server (or leaf) s may host a number of virtual
machines (or "nodes") no larger than cap(s). The edges may have [bandwidth] capacities, denoted cap(e), and all have unit lengths. While the assumption that the physical network forms a tree may seem a big restriction from a theoretical standpoint, this conforms with real applications.

* The data to be processed is distributed over the physical network; the data
locations are given and are not subject to optimization. The data consists of <tau> chunk types, denoted c_1,..,c_<tau>, where each chunk type c_i has r_i = 1
replicas, stored at different servers. However, a single server may store an arbitrary number of replicas of different chunk types. Lastly, it is important to note that for each chunk type, only one replica has to be chosen for processing.

* The virtual network is characterized by the number of nodes (virtual servers)
to be hosted on the physical servers, subject to the servers capacities.

* The algorithm is to (i) select the servers to host the nodes, (ii) select
one replica per chunk type to be processed, and (iii) assign each selected replica to a node; in the last part there is an important assumption that the nodes load is to be balanced, i.e., that all of them are assigned an equal number of replicas for processing. (So, in particular, the number of chunk types is a multiple of the number of nodes.)

* The goal is to minimize the "resource footprint", which is a weighted sum of
two types of connection costs: the first one, multiplied by b_1, is total distance between each selected replica and the node processing it (i.e., the physical servers hosting the replica and hosting the node), and the other, multiplied by b_2, is the total distance between every pair of nodes (again, in actuality: servers that host them).

This describes the problem at its most general, but the authors consider 5 possible independent restrictions, which means that overall they consider 2^5=32 (potentially) different settings. The *lack* of such restrictions are:

- replica selection (RS), i.e., redundancy of replicas of some chunk type, which
necessitates selection of a single replica per each such type

- multiple chunk assignment (MA), i.e., the necessity to assign multiple
replicas to each node when there are more chunk types than nodes

- flexible node placement (FP), i.e., the ability to embed/host the nodes at
arbitrary servers (as opposed to a scenario where the nodes are already placed)

- node interconnect (NI): b_2 > 0, i.e., the cost of pairwise connections of the
nodes does contribute to the objective

- bandwidth constraints (BW): cap(e) is bounded (as opposed to infinite)
at least for some edges of the substrate network.

The article gives a complete classification of all 32 variants: for each variant it either provides a polynomial time algorithm to solve it or a proof of NP-hardness via reduction from the 3D-Matching problem. As one might suspect, the issue with such kind of an article is that none of these multiple results seems particularly interesting. Indeed, all the algorithms are standard applications of either flow techniques (matchings in particular) or dynamic programming, and the reductions from 3D-Matching are straightforward. The only possible exception is Section 5, which gives two reductions that are more involved, as they aim at proving NP-hardness of fairly restricted variants of the problem, namely FP+RS(2)+MA(4) and RS(2)+FP+NI+BW, where the two letter acronyms stand for aforementioned lack of restrictions and RS(n) means that even though replica selection is required, there are at most n replicas per chunk type. Unfortunately, this part of the article is rather poorly written: Section 5.1 (which gives the first of the two reductions) in particular looks as if no one read it in its current form.

Still, even Section 5 is not too technically involved, so it seems that the largest contribution of the article is introducing a sensible model. I am unable to foresee how much follow-up work it will inspire, given that in principle the authors have completely resolved all of its variants.

Thus, I would rather refrain from suggesting accepting or rejecting the submission, and merely point out that a major revision would be required before the draft could be accepted; specific comments follow at the end.

QUESTIONS / FUTURE WORK that the authors or others might consider interesting:

1) All edge lengths in the physical network are unit. What would happen if they were not? Presumably the flow-based algorithms would still work, but perhaps something changes in the dynamic programming approach? I.e., perhaps some of them become pseudopolynomial?

%%%%%%%%%% RESPONSE %%%%%%%%%%
Flow-, dynamic programming- and matching-based algorithms remain correct for arbitrary non-negative edge weight function.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

2) Perhaps some relaxation of the requirement that replicas are distributed uniformly between nodes makes sense?

%%%%%%%%%% RESPONSE %%%%%%%%%%
%maciek: I do not understand the usage of "uniform distribution" with respect to input. I assume that this question is indeed about the co-location of more than one chunk replica in a single node. Hence, we can respond something like: "Presented algorithms remain correct for any distribution of chunk replicas. In addition, we do not restrict the number of replicas that are hosted in a single node."
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

3) As some of the problems are NP-hard, and moreover of practical interest, considering approximation algorithms seems a worthwhile direction.

%%%%%%%%%% RESPONSE %%%%%%%%%%
Quality approximation algorithms for stated problems are broad topic on its own. Such solutions are under ongoing research.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

3) I suppose that one might argue that in the applications the underlying physical network is (more or less) fixed. In such case, when the instance specifies only the data distribution, number of nodes to be embedded, and the b_1 and b_2 coefficients, would some of the previously NP-hard problems become tractable?

%%%%%%%%%% RESPONSE %%%%%%%%%%

Assume that the physical network is fixed and name it T_fixed. We presented hardness results for two variants:
1) RS+MA+NI using the construction that we name (for sake of this response) the C1
2) RS+MA+FP using the construction that we name (for sake of this response) the C2
3) RS(2)+MA+FP using the construction that we name (for sake of this response) the C3
4) RS(2)+NI+FP+BW using the construction that we name (for sake of this response) the C4

The remaining hardness results (RS+MA+FP+NI, RS+MA+FP+BW, RS+FP+NI+BW, RS+MA+FP+NI+BW) are the consequences of hardness results (1) and (2).

Consider the hardness result of RS+MA+NI. If the construction C1 can be embedded in T_fixed (which requires certain height of the tree as well as the number of nodes and the existance of desired links), then the hardness result holds. The carefully-crafted threshold value can be satisfied only by solutions that co-locates nodes with replicas. Hence, the reduction is correct with possible additional leaves in Triple Gadgets.

Consider the hardness result of RS+MA+FP. If the construction C2 can be embedded in T_fixed, then the hardness result holds. The threshold value can be satisfied only by solutions that place exactly one node in every Triple Gadget. Hence, the reduction is correct with possible additional leaves in any Triple Gadget, as well as additional leaves outside Triple Gadgets.

Consider the hardness result of RS(2)+MA+FP. If the construction C2 can be embedded in T_fixed, then the hardness result holds. Due to the value of threshold, leaves without chunk replicas cannot be considered candidates for hosting the node.

Consider the hardness result of RS(2)+NI+FP+BW. If the construction C2 can be embedded in T_fixed and number of nodes to be placed is at least two, then the hardness result holds. In presence of bandwidth capacities (the BW variant), each edge has the specified bandwidth capacity. For each edge of T_fixed that is outside of C2, we set its capacity to zero, disallowing the placement of any node in such leaf.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SPECIFIC COMMENTS:

SECTION 5 in general: You keep changing the names for instances of the two problems. Be consistent!

%%%%%%%%%% RESPONSE %%%%%%%%%%
We changed every occurence of FP+RS(2)+MA(4) to RS(2)+MS(4)+FP. The naming convention is to list all the variants in the following order: RS, MA, FP, NI, BW.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SECTION 5.1:

Let me reiterate that I firmly believe that this section requires a major revision. Even though the argument should be easy to follow, this is impossible in the current version. When you do, please provide the calculations. And check if the threshold value is what it should be, because it appears to me that it is slightly off.

- Observation 2, points 1 and 2: some chunk types are unique to either of the two subtrees, so clearly, they do not have exactly one replica in the other; please refomulate.

%%%%%%%%%% RESPONSE %%%%%%%%%%
We changed the Observation in such way that it excludes U_e

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

- "Reduction" (page 26):
   * In point 2: (1) the n nodes should be placed each in a distinct leaf,
     not all in the same one, as your writing suggests; (2) the 3D-Matching
     solution selects a triple, not a Triple Gadget --- the latter
     corresponds to the former

%%%%%%%%%% RESPONSE %%%%%%%%%%
We clarified the construction.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   * In point 3, you refer to "Element Subtree" but, strictly speaking,
     you have never defined it.

%%%%%%%%%% RESPONSE %%%%%%%%%%
We changed the Element Subtree to Element Gadget, as it was the object that we meant to refer to.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

- "Proof of correctness" (right after the "reduction"):
   * Correctness of what?

%%%%%%%%%% RESPONSE %%%%%%%%%%
We changed the "Proof of correctness" to "Proof of correctness of the reduction"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   * (1) The two costs you introduce presumably correspond to something,
     but you never explain that. (2) Moreover, t is undefined, and should
     probably not be used at all, since on page 23 you state that t will be
     used for a triple from T. (3) Similarly, it is only in Lemma 6 that
     you specify the domain for u.

- Lemma 7: Just like the "Element Subtree", "Unique Subtree" was not defined.
   And neither was sigma.

- Lemma 8: \ell was not defined. I can only assume it is *not* the \ell from
   the previous Lemma, since that one was to prove that \ell=0.

SECTION 5.2:

- I believe that you should provide a figure with the tree, similar to the one
   in Section 5.1

- Element Gadget: you write that it has 4*(deg(e)-1)+1 leaves, but it seems that
   this was copied verbatim from Section 5.1 and that the right number is deg(e)

- The threshold value formula (bottom of page 29):
   * Could you please increase the font size?
   * It appears that every single occurrence of |V| in the formula should be
     replaced by n
   * Shouldn't the summand for e in the sum on the right end of the first line
     be \binom{deg(e)-1}{2} rather than deg(e)-1?
   * Are you sure of the coefficient (i.e., their lack) next to the terms in
     the last two lines?

- Lemma 10:
   * last line of statement: n should be replaced by |V|
   * could you explain in detail how you arrive at the final inequalities?

- Lemma 12 and its proof: It appears that you mean the "Element Gadget", even
   though you are referring to the "Triple Gadget"

MINOR COMMENTS:

throughout article: I think you never define/explain the term "uplink"

p3l54-55: "constituting parts" --> "constituent parts"

%%%%%%%%%% RESPONSE %%%%%%%%%%
We changed the "constituting parts" to "constituent parts"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p4l14-17: The sentence "The standard datacenter topologies today are
(multi-rooted) fat-tree resp. Clos topologies [2, 21], hierarchical networks
recursively made of sub-trees at each level; servers are located at the tree
leaves." sounds gibberish --- please reformulate.

%%%%%%%%%% RESPONSE %%%%%%%%%%
%maciek: I suggest to change the sentence to:
"The standard datacenter topologies today are
(multi-rooted) fat-tree and Clos topologies [2, 21], the hierarchical networks
recursively made of sub-trees at each level. Servers are located at the tree
leaves."
%maciek: however, I haven't committed the change yet
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p8l48: remove the first 'a' from "can now be computed a from a solution to the"

%%%%%%%%%% RESPONSE %%%%%%%%%%
Excessive 'a' was removed
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p8l40: insert space in "withthe"

%%%%%%%%%% RESPONSE %%%%%%%%%%
We inserted the space in the desired location.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p11l54: change "we can computed" to either "can be computed" or "we can compute"

%%%%%%%%%% RESPONSE %%%%%%%%%%
We changed "we can computed" to "can be computed"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p1626-27: was "communicated" supposed to be "connected"?

%%%%%%%%%% RESPONSE %%%%%%%%%%
We changed "communicated" to "connected" for clarity purposes. However, please note that in our model the connection implies the communication and the reservation of bandwith for such communication.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p16l44: remove the repetition of "formula"

%%%%%%%%%% RESPONSE %%%%%%%%%%
We removed the excessive repetition of "formula"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p19l23 (end of the paragraph): all elements of the *union* of X,Y,Z have to be
covered (once), not their Cartesian product.

%%%%%%%%%% RESPONSE %%%%%%%%%%

We changed the Cartesian product to the union
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NP-hardness proofs: I think you should use a single symbol, perhaps \tau, for the threshold; Th rather confusingly looks like two disconnected symbols.

p22-23: Lemma 4 and Theorem 5 --- since you do the easy calculation of the cost a specified solution in proof of Theorem 5, I believe you should also properly explain the argument in the proof of Lemma 4: it is straightforward but definitely no easier than the mere calculations in Theorem 5, as the claim is that that specific solution minimizes the cost.

Reviewer #3: The paper studies a well-motivated problem related to virtualized data centers. It formulates and solves several versions pertaining to the problem of data locality aware virtual cluster embeddings. The model (although a bit complicated) does seem to capture most of the problems related to data aware embeddings. The authors have two layers, the substrate network and the virtual network, and the former needs to get mapped to the latter. They consider several aspects (5, to be exact) namely replica selection, multiple assignment, flexible placement, node interconnect, and bandwidth capacities.

The solutions/algorithms are fairly simple, and I would not claim that the technical content of the paper is substantial. However, the novelty of the problem and the arguably sound formulation merit a publication.

Given the number of problems (32), I would have liked the authors to perhaps create a table; it is hard for a reviewer to keep track of which problems can be solved efficiently and which are hard. Also, the optimization function - the footprint, is just stated and not discussed at all. Perhaps this is a well-agreed upon function in this field (which is new to me). Why is this the idea function to optimize?

%%%%%%%%%% RESPONSE %%%%%%%%%%
Summary of results is presented in Table 1 at page 34.

%maciek: I cannot answer the question about the optimization function being the footprint.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Lastly, for the problems that are hard, I would also like the authors to mention how hard they are to approximate, or whether one can get a nice approximate solution.

%%%%%%%%%% RESPONSE %%%%%%%%%%
Quality approximation algorithms for stated problems are broad topic on its own. Such solutions are under ongoing research.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%