Confluence: Speeding Up Iterative Distributed Operations by Key-dependency-aware Partitioning

**Response to Review Comments**

Our revision on the last version of the paper mainly focuses on the improvement of three aspects raised by the reviewers, namely, the skewness analysis, the executor tying issues, and the data compression influence.

1. **Skewness analysis part**: in the last version, when we said "keys of input entries follow a distribution Dist”, where the “Dist” we wanted to mean was actually the value distribution between keys. By considering the key distribution between nodes in joints, we obtained the workload skewness of RKP. We found such presentation was a bit confusing. Therefore, in Section 3.3 of the revised version, we redefine the distribution model, which divides the value distribution of partition nodes into the joint of three sub-distribution: Dv |><| Dk |><| D\_f(k), where Dv is the values distribution between keys, Dk is keys between mapped keys, and D\_k(k) is mapped keys between partitions. We found that this redefinition is preciser on one hand, it can describe both the skewness of RKP and CKP on the other hand. Moreover, an observation on the only different distribution “Dk” helps us easily conclude in which condition, CKP does not increase workload skewness as compared to RKP. After all, we highlight the last paragraph of this section to avoid misunderstanding that CKP would remove workload skewness. It only tries not to increase it in fact.

2. **Executor tying issue:** we found that when multiple applications are running, some executors do have more tasks than others due to inappropriate hashing. For a concrete example, suppose there were three executors in the Spark cluster, and there were three applications, whose number of tasks is 1, 2, 3, respectively, and the task IDs are numbered incrementally from 0, 1, until 2 if possible. If we use a mod-by-3 function as the hash function, all of the three applications will compete to tie their task 0 to the executor with hash code being 0, but only 1 applications with 3 tasks will be tied to the executor whose hash code is 2. We address this problem in Section 5.1 in the revised version, and propose to use a hash function that will randomly hash a task to an executor.

3. **Data compression:** in our existing evaluation, data are already compressed in “lz4” in shuffle network streams. We agree that it is important to specify this and compare the effect of CKP with shuffle data compression. We supplement this comparison in Section 6.6, and find that compression does reduce the shuffle size, but CKP can work with compression together to reduce more. We also notice the potential time overhead of data compression and decompression.

4. **Dependency Probability:** The calculation of key dependency may not work well without application example. We found that we have discussed in detailed about dependency probability in cases it is not 1, i.e., MultiAdjacentList and KMeans. Nonetheless, we add a hint in Section 2.3 where readers can discovery more about the dependency probability.

5. We have also polish the presentation by fixing some typos or grammar issue.