ALOCS: An Allocation-Aware Key-Value Repository
Response to Reviewer Comments

We thank the reviewers for their feedback and suggestions. We have made the utmost effort to carefully incorporate most of the feedback in the short time given for reviewing the article. We think the article has improved substantially as a result of these changes. All substantial changes made to the article have been highlighted with red font color. In this letter, we discuss the changes made and respond to the specific revision items listed in the reviews. As an overall summary, we made three major changes to the article: (1) The focus of the article has been changed from scalability to the ability to control data placement; this is in fact the major contribution of the proposed system; (2) A reference to the SBBD conference paper and list of differences between the submissions have been included; (3) A discussion on the role ALOCS can play in a layered DBMS architecture has been added; besides providing a better contextualization for the system, it helped in pinpointing the differences between ALOCS and related works, such as Autoplacer.

1. REVIEWER A

Comment 1: The paper describes ALOCS, a key-value storage system that is allocation aware. This is an extended version of a full paper published in SBBD 2016. The original paper is not even cited in this JIDM submission, and thus there is no clear statement of what the differences between these two submissions are.

Response: We have included a reference to the SBBD 2016 paper, and listed the differences. The new version highlights the distinctive features proposed by ALOCS: (1) a data model that supports access to individual key-value pairs, while grouping them into *buckets* in order to reduce inter-server communication; (1) caching of buckets; and (3) a modular design. To this end, we have included the interface specification of all modules that compose the system, as well as examples to show the flexibility of the proposed storage model, and a new experiment to determine the effectiveness of our design goals. Moreover, we extended the article with: (1) a more detailed contextualization of the functionality ALOCS can provide in the physical layer of a DBMS; (2) extended the related work discussion; (3) presented a more detailed discussion on the importance of a standard interface between the modules in order to obtain "pluggable" components.

Comment 2: In my opinion, this is not enough to justify a new submission. What I would like to see in this paper is a clear comparison of ALOCS with existing storage systems. For instance, what would figure 9 look like for Autoplacer and Scalaris? This would give robustness to the paper, since you would not be only comparing different configurations of ALOCS. The current experiments of the paper shows benefits of the allocation schema, but also shows that there are overheads. There is no clue on what is the performance of related systems, and so no clue on the benefit that it brings to applications in practice. Also, related work should compare ALOCS with Autoplacer. What are the advantages of ALOCS when compared to Autoplacer?

Response: A more detailed comparison of ALOCS with Autoplacer has been added in the related work section. Although we did not have enough time to run new experiments, we included a discussion at the end of Experiment 3, highlighting the differences of accessing key-value pairs using ALOCS and a DHT such as Scalaris.

Comment 3: In the experiments, you mention that using an odd number of servers is better "to guarantee the majority of the quorum". However, the concept of quorum was not explained in the paper. Where is it used? Why?

Response: A paragraph in Experiment 4 has been added to present the concept of quorum and justify the choice of odd number of servers.

Comment 4: The last paragraph of the conclusion mentions the Metamodel module. What is this?

Response: We apologize for the mistake. We meant “metadata” module.

Comment 5: The paper also needs a careful English revision. There are several grammar problems along the text, and also words that are being used wrongly.

Response: The entire text has been revised to correct the mistakes.

2. REVIEWER E

Comment 6: Este artigo traz uma proposta de arquitetura de persistência de dados distribuída com foco no controle de localidade. De maneira bem argumentada pelos autores, este é um problema que ainda não conta com boas soluções, quando buscamos os trabalhos da literatura e ferramentas disponíveis. O artigo é bem escrito de maneira geral mas, mesmo para não "native english speakers", nota-se que há bastante margem para melhorias. Nada que comprometa a leitura do texto, com boa organização e clareza na maior parte do tempo.

Response: O artigo foi revisado completamente. Pedimos desculpas pelos erros.

Comment 7: Apesar da ênfase em localidade, um dos assuntos mais discutidos e que motiva a proposta do trabalho envolve escalabilidade. De fato, a ideia dos buckets de pares chave-valor para dados usualmente acessados em conjunto é interessante e, quase óbvia. E está na base da ideia da possível escalabilidade. Mas aqui me parece que o artigo peca bastante em termos de discussão da solução. Não somente acaba se limitando às situações onde a clusterização é possível com grande grau de independência como também a atribuição da responsabilidade da política de fragmentação mais alocação física ficar com o desenvolvedor da aplicação pode comprometer bastante os resultados.

Response: O foco do artigo foi mudado para dar ênfase à questão da localidade ao invés da escalabilidade. Além disso, o introdução coloca em evidência o objetivo do sistema proposto, mostrando que componentes de particionamento e políticas de alocação física são ortogonais à infra-estrutura de armazenamento.

Comment 8: Em particular, clusterizar é um problema por si só complicado e computacionalmente complexo. Os exemplos simples mencionados pelos autores nem sempre estão presentes na prática. No caso de bancos de dados em grafos, por exemplo, a distribuição dos nós em clusters nem sempre é trivial, inclusive quando arcos relacionam nós de clusters distintos. No caso de bancos relacionais, o livro de Ozsu e Valduriez discute bem o problema de otimização relacionado à modelos de fragmentação e, sobretudo, alocação física.

Avaliando o artigo pontualmente no que diz respeito a proposta de acesso via localidade, de fato, não há dúvida de que se trata de uma solução bem planejada e pensada pelos autores. O trabalho de implementação, mesmo descrito de maneira sucinta no artigo, convence o leitor de que é robusto o suficiente para permitir a bateria de testes inicialmente proposta. As escolhas de tecnologias parecem corretas, mas mesmo assim sente-se falta de discussão sobre alternativas e argumentação melhor para os produtos adotados. A ideia de flexibilidade na hierarquia é muito boa e valoriza a contribuição trazida no artigo.

Response: De fato os problemas de fragmentação e alocação de dados são bastante complexos e há um grande número de soluções propostas. O ALOCS foi proposto para *dar suporte* à implementação destas técnicas. A escolha das ferramentas utilizadas na implementação foi justificada no artigo.

Comment 9: Acredito também que faltou maior detalhamento e, principalmente, motivação para cada uma das funções e operadores propostos para os 3 módulos. Simplesmente listar obriga o leitor a ficar pensando nas situações, utilidade e completeza.

Response: O detalhamento e justificativa das funções propostas foram adicionadas no artigo.

Comment 10: No que diz respeito aos testes experimentais, me parecem que são bem simples e iniciais. De novo, com relação à localidade, tudo certo. Mas não há nada com respeito à escalabilidade, aspecto fundamental para a proposta de arquitetura distribuída.

Response: O foco do artigo foi alterado para a questão do controle de localidade. Devido a restrições de tempo para preparar a nova versão, não foi possível a realização de novos experimentos. No entanto, a análise dos experimentos foi estendida, justificando as escolhas e comparando com possíveis alternativas.

Comment 11: Me parece particularmente estranho ler um trecho do trabalho onde os autores mencionam que o módulo de persistência pode herdar da ferramenta Ceph algumas de suas propriedades, como escalabilidade, performance e alta disponibilidade. Idem depois quanto à Zookeeper. Não faz sentido isso pois são adjetivos que precisam ser demonstrados e qualificados. Insisto: o tema de escalabilidade, mesmo fundamental no artigo, acaba sendo pouco explorado. O leitor se permite "imaginar" que é possível mas sem mais discussões ou explicações por parte dos autores.

Response: Concordamos com o argumento do revisor. De fato algumas características do sistema desenvolvido não são decorrentes do modelo proposto, mas das ferramentas sobre as quais ele foi implementado. Como o objetivo do ALOCS foi incorporar conceitos aos repositórios chave-valor (buckets e diretórios) que permitam o controle de localidade, o artigo foi alterado de acordo. A idéia central é que buckets correspondam às páginas em disco nos SGBDs centralizados. Assim, como os SGBDs atuais já foram desenvolvidos baseados no modelo de páginas (compostas por um conjunto de registros) e são otimizados para minimizar a quantidade de transferências entre disco e memória, as mesmas técnicas podem ser diretamente aplicadas utilizando o ALOCS como backend de armazenamento. Neste caso, a unidade de acesso dos SGBDs, que são os registros, correpondem aos pares chave-valor do ALOCS e a unidade de transferência entre servidores é o bucket.

Comment 12: Alguns comentários pontuais sobre o texto: binary tree apenas não! É binary search tree! Só não ficou claro por que binary e não n-ary. Depois fala rapidamente no problema de balanceamento e dá a entender que deveria ser b-tree-like.

Response: O texto foi corrigido para “binary search tree” e tentamos remover os termos “abrasileirados”. A necessidade do balanceamento da árvore ficou clara com os resultados dos experimentos e será implementada futuramente.

Comment 13: Em suma: o artigo é sucinto e pouco detalhado em partes relevantes porém traz algumas claras contribuições que podem interessar ao público alvo do JIDM e comunidade de banco de dados no Brasil. Talvez acrescentando-se uma maior discussão sobre escalabilidade de arquitetura tivéssemos um texto com maiores contribuições. Por outro lado, é verdade que o propósito principal após motivações genéricas diz respeito a um repositório de pares chave-valor com localidade devidamente tratada. Restringindo-se o texto para esta motivação e objetivo específico, acredito que o artigo ficaria devidamente auto-contido.

Response: Agradecemos a sugestão. Alteramos o foco do artigo para ressaltar a importância do controle de localidade e acreditamos que a contribuição ficou mais clara na nova versão.

3. REVIEWER F

Comment 14: There is relatively little difference between this version of the paper and the one published at SBBD. If one requires the usual 30% new contribution wrt the conference version, I do not believe that is the case. Thus I think the paper should be revised. A fair question then is "revised how?", while I appreciate the question I don't think it's the job of the reviewer to suggest how the paper should be revised to address that (30% new contribution) issue. I do have a few questions about the submitted version, which are presented below.

Response: Please refer to the response of Comment 1.

Comment 15: The paper proposes ALOCS, a distributed key-value data repository that allows applications to make use of data locality to avoid metadata network traffic and increase its performance. The authors claim that ALOCS is scalable and is able to deal with large volumes of data. ALOCS is based on two existing systems (Ceph and Zookeeper). Although it is reasonable to believe that ALOCS provides scalability and the ability to deal with large volumes of data, two facts should be noticed: (1) these features depend on the underlying systems and (2) the experiments do not consider large volumes of data/transactions to support such claim.

Response: We agree with the reviewer that scalability derives from the tools on which ALOCS has been implemented. As pointed out in the response for Reviewer F, we modified the article to focus on the locality control provided by ALOCS. The concept of *buckets* as communication units brings key-value repositories closer to traditional disk page accesses, but in a distributed setting. Clustering commonly accessed data items on disk pages has long been an important technique to optimize DBMS performance. Moreover, query plans are constructed to minimize the number of page transfers between disk and memory. Results of our experimental study show that the same applies in a distributed setting. That is, minimizing the number of communication among servers is essential for optimizing the overall system performance. The similarity between page/items in a traditional DBMSs and buckets/pairs in ALOCS may allow traditional query optimization techniques to be more readily applied, and highligths the importance of using a repository that supports data allocation control as a DBMS backend.

Comment 16: One of ALOCS' main features is that it gives the user/application total control over data location. However, some parts of the process are not very clear in the text. It seems that the user/application has to create buckets (with their associated intervals of keys) before inserting any data items into them. However, in cases where the application does not know how to properly define the intervals, the performance of the system may be compromised (as shown in the experiments, where sequential bucket creations lead to a poor performance). This brings me to another question: if these situations lead to such a poor performance, why not dealing with it by rebalancing the metadata trees?

Response: Controlling data placement based on key intervals was a design choice, which resembles the idea adopted by DHTs. However, as opposed to DHTs, which have dynamic intervals, in ALOCS intervals are fixed and determine the buckets location. An alternative would be to control data locality based on key-value pairs, instead of buckets. In this case, the volume of metadata would be much larger and may not fit in main memory, and would lead to poor performance. We intend to balance metadata trees in a future version of ALOCS.

Comment 17: Still regarding the data locality, why is it better to give the user/application control over the data allocation while other systems, such as Autoplacer, take care of the data distribution automatically? Even though the systems (ALOCS and Autoplacer) do not deal with data in the same fashion, in a higher level they serve the same purpose. They should be compared in the experimental evaluation.

Response: Indeed they have the same goal of minimizing inter-server communication. However, Autoplacer proposes an algorithm for placing *replicas* of key-value pairs stored on a DHT, whereas ALOCS proposes a storage system on which different partitioning and allocation algorithms can be developed. We have extended the related work section to make this distinction clear, as well as a better contextualization of ALOCS in a DBMS architecture in the introduction. Due to the short time for revising the article, it was not possible to run additional experiments, especially ones that require third party software. However, the analysis of the existing experiments has been extended.

Comment 18: Regarding the experimental setup, the authors have not considered large scale environments. Not only the used infrastructure was relatively small, the volume of data and the amount

of transactions do not reflect real situations. If the use of a larger infrastructure was not viable, why haven't the authors considered at least using a benchmark to evaluate the performance of ALOCS?

Response: There are some benchmarks for evaluating key-value systems. However, these benchmarks do not address the allocation problem. Thus, we generated a synthetic database in order to control data co-allocation and run the experiments.

Comment 19: The authors say that data collocation is inherited from CRUSH, an algorithm in Ceph that implements the distribution of the data items according to a set of rules. Are these rules user-defined? If so, it would be interesting to have them discussed, as they are highly related to a major feature of the system (data locality).

Response: Rules are user-defined. They are compiled and kept in binary format in order to be used when pools are created. The system administrator is allowed to modify the rules after pools are created. We have added this information in Section 4.1.

Comment 20: Overall, ALOCS seems like a promising idea. The architecture proposed and the generic interfaces allow it to be extended in several manners. However, the experimental evaluation does not seem to explore all the flexibility and scalability that ALOCS supposedly has to offer. Issues like the unbalanced metadata tree or the choice of the experimental setup could have been discussed in more details.

Response: Indeed the experimental results show the importance of keeping the metadata search structure balanced. We intend to implement it in the future. Although we have not been able to run new experiments, we extended the analysis of the experiments, and discussed how they are meaningful in the development of a DBMS using ALOCS as its storage backend.

Comment 21: (a) The authors state that vertical scalability is expensive, however in cloud environments it is not exactly the case. (b)"... bring information closed to their user applications." - what does that mean exactly? (c)"Once in the buffer, it is unlikely that subsequent access to key-value pairs in the same bucket require additional transmissions" - why just unlikely? Unless it is a "write" operation, it should always use the cache to access the data, right? (d)"For grouping a set of key-value pairs in a bucket and store them in a Ceph object, we've adopted the following strategy." - Although it is important detail of the paper, such strategy is not clear to the reader.

Response: (a) The focus of the article has been shifted from scalability to allocation control; (b) The idea is to store the data on the same server as the one that uses them more often in order to minimize inter-server communication; (c) The cache has a limited size. We implemented an LRU policy (which has not been detailed in the article), but it is possible that the bucket has been flushed out before subsequent accesses of keys in the same bucket; (d) The mapping of concepts in ALOCS to Ceph has been rewritten to make the strategy clear.

Comment 22: "An important point to notice is that the total execution time for the configuration with replication on three metadata servers is only 34% higher than the time without replication." - Wouldn't 34% be a relatively large amount in a large scale environment?

Response: In fact, the difference is negligible for sequences with up to 40 (bucket) insertion operations, and 34% higher for 50 insertions. Moreover, Experiment 4 shows that with 3 metadata servers the throughput of the system for read operations almost doubles. We can conclude that the number of metadata servers must be determined by taking into consideration the expected load of read an write operations.

Comment 23: The placement of the Figures in the experimental section compromises the readability of the paper. The results of the experimental evaluation deserve a more detailed discussion.

Response: Figures placement has been corrected and the experimental analysis has been extended in order to highlight the main features supported by ALOCS.