Disaggregating Persistent Memory and Controlling Them Remotely: An Exploration of Passive Disaggregated Key-Value Stores

Shin-Yeh Tsai, Yizhou Shan, Yiying Zhang
Purdue University and University of California, San Diego

Abstract

Many datacenters and clouds manage storage systems separately from computing services for better manageability and resource utilization. These existing disaggregated storage systems use hard disks or SSDs as storage media. Recently, the technology of persistent memory (PM) has matured and seen initial adoption in several datacenters. Disaggregating PM could enjoy the same benefits of traditional disaggregated storage systems, but it requires new designs because of its memory-like performance and byte addressability.

In this paper, we explore the design of disaggregating PM and managing them remotely from compute servers, a model we call *passive disaggregated persistent memory*, or *pDPM*. Compared to the alternative of managing PM at storage servers, pDPM significantly lowers monetary and energy costs and avoids scalability bottlenecks at storage servers.

We built three key-value store systems using the pDPM model. The first one lets all compute nodes directly access and manage storage nodes. The second uses a central coordinator to orchestrate the communication between compute and storage nodes. These two systems have various performance and scalability limitations. To solve these problems, we built Clover, a pDPM system that separates the location, communication mechanism, and management strategy of the data plane and the metadata/control plane.

1 Introduction

Disaggregating storage and compute has become a common practice in many datacenters and clouds. Disaggregation makes it easy to manage and scale both the storage and the compute pools. By allowing the storage pool to be shared across applications and users, disaggregation consolidates storage resources and reduces their cost.

Existing disaggregated storage systems are all SSD- or HDD-based. Today, a new storage media, non-volatile memory (or persistent memory, *PM*) has arrived and has already seen adoption in several datacenters. Existing distributed PM systems [2, 4] have mainly taken a non-disaggregated approach, where each server in a cluster hosts PM for applications running both on the local server and remote servers (Figure 1(a)).

Disaggregating PM could enjoy the same management and resource-utilization benefits as traditional disaggregated storage systems. However, building a PM-based disaggregated system is very different from traditional disaggregated storage systems as PM is byte addressable and orders of magnitude faster than SSDs and HDDs. It is also different from disaggregated memory systems [3], since when treated as storage systems, disaggregated PM systems need to sustain power failure and be crash consistent.

There are two possible design directions in building disaggregated PM systems, and they differ in where management software runs. The first type, and the type that has been adopted in traditional disaggregated storage systems, runs management software at the storage nodes, *i.e.*, actively managing data at where the data is. When applying this model to PM, we call the resulting system *active disaggregated PM*, or *aDPM* (Figure 1(b)). By co-locating data and their management, aDPM could offer low-latency performance to applications. However, aDPM requires significant processing power at storage nodes to sustain high-bandwidth networks and to fully deliver PM's superior performance.

In this paper, we explore an alternative approach of building disaggregated PM by treating storage nodes as *passive* parties that do not perform any data processing or data management tasks, a model we call *pDPM*. pDPM offers several practical benefits and research value. First, pDPM lowers owning and energy cost. Without any processing need, a PM node (we call it a *data node* or *DN*) can either be a regular server that dedicates its entire CPU to other applications or a hardware device that directly attaches a NIC to PM. Second, pDPM avoids DN's processing power being the performance scalability bottleneck. Finally, pDPM is an approach in the design space of disaggregated storage systems that has largely been overlooked in the past. Exploring pDPM systems would reveal various performance, scalability, and cost tradeoffs that could help future researchers and systems builders make better design decisions.

pDPM presents several new challenges, the biggest of which is the need to avoid processing all together from where data is hosted. Existing in-memory data stores heavily rely on local processing power for both the data path and the control path. Without any processing power, accesses to DNs have to come all from the network, which makes data operations like concurrent writes especially hard. Moreover, DNs cannot perform any management tasks or metadata operations locally, and each DN can fail independently.

A key question in designing pDPM systems is where to perform data and metadata operations when we cannot perform them at DNs. Our first approach is to let client/compute nodes (*CN*s) perform all the tasks by directly accessing DNs with *one-sided* network communication, a model we

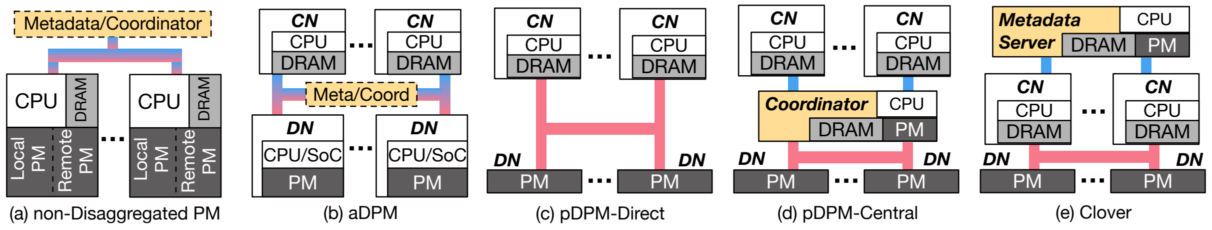

Figure 1: **PM Organization Comparison.** *Blue bars indicate two-way communication and pink ones indicate one-way communication. Bars with both blue and pink mean support for both. Dashed boxes mean some but not all existing solutions adopt centralized metadata server (or a coordinator).*

call *pDPM-Direct* (Figure 1(c)). After building and evaluating a real pDPM-Direct key-value store system, we found that since CNs cannot be efficiently coordinated, pDPM-Direct performs and scales poorly when there are concurrent reads/writes to the same data. Our second approach is *pDPM-Central* (Figure 1(d)), where we use a central server (the *coordinator*) to manage DNs and to orchestrate all accesses from CNs to DNs. Although pDPM-Central provides a way to coordinate CNs, it adds more hops between CNs and DNs, and the coordinator is a new scalability bottleneck.

To solve the issues of the above two pDPM systems, we build Clover, a key-value store system with a new architecture of pDPM (Figure 1(e)). Clover's main ideas are to separate the location of data and metadata, to use different communication mechanisms to access them, and to adopt different management strategies for them. Data is stored at DNs. Metadata is stored at one or few global metadata servers (*MS*s). CNs directly access DNs for all data operations using *one-sided* network communication. They use *two-sided* communication to talk to MS(s). MS(s) perform all metadata and control operations.

Clover achieves low-latency, high-throughput performance while delivering the consistency and reliability guarantees that are commonly used in traditional distributed storage systems. We designed a set of novel techniques at the data and the metadata plane to achieve these goals. Our data plane design is inspired by log-structured writes and skip-lists. This design achieves 1-/2-RTT read/write performance when there is no high write contention, while ensuring proper synchronization and crash consistency of concurrent writes with satisfactory performance. We move all metadata and control operations off performance critical path. We *completely* eliminate the need for the MS to communicate with DNs; it performs space management and other control tasks without accessing DNs. In addition, Clover supports replicated writes for high availability and reliability.

We evaluate Clover, pDPM-Direct, and pDPM-Central using a cluster of servers connected with RDMA network (some acting as CNs and MSs, some acting as emulated DNs). We compare these pDPM systems with two non-disaggregated PM systems [2, 4] and an aDPM key-value store system [1] running on CPU-based servers and on ARM-SoC-based Bluefield SmartNIC. We perform an extensive set of experiments to study the latency, throughput, scalability, CPU utilization, and owning cost of these systems using microbenchmarks and YCSB workloads. Our evaluation results demonstrate that Clover is the best-performing pDPM system, and it significantly outperforms traditional distributed PM systems. Clover achieves similar or better performance as aDPM systems under common datacenter workloads, while reducing CapEx and OpEx by 1.4× and 3.9×. However, we also discovered a fundamental limitation of pDPM-based storage systems: no processing at where data sits could hurt write performance, especially under high contention of concurrent accesses to the same data entry. Fortunately, most datacenter workloads are read-most. Thus, we believe pDPM and Clover to be good choices future systems builders can consider, given their overall benefits in cost, performance, and scalability.

Overall, this paper makes the following contributions:

- Thorough exploration of the passive disaggregated persistent-memory architecture, revealing its benefits, tradeoffs, and pitfalls.
- Implementation of Clover and two alternative pDPM key-value stores, all guaranteeing proper synchronization, crash consistency, and high availability.
- A detailed design of how to separate the data plane and the metadata plane under the pDPM model.
- Comprehensive evaluation results that can guide future DPM research.

The original paper was published at USENIX ATC 2020 [5]. All our pDPM systems are publicly available at https://github.com/WukLab/pDPM.

References

[1] A. Kalia, M. Kaminsky, and D. G. Andersen. Using RDMA Efficiently for Key-value Services. In *SIGCOMM 2014*.
[2] Y. Lu, J. Shu, Y. Chen, and T. Li. Octopus: an rdma-enabled distributed persistent memory file system. In *ATC 2017*.
[3] Y. Shan, Y. Huang, Y. Chen, and Y. Zhang. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In *OSDI 2018*.
[4] Y. Shan, S.-Y. Tsai, and Y. Zhang. Distributed shared persistent memory. In *SoCC 2017*.
[5] S.-Y. Tsai, Y. Shan, and Y. Zhang. Disaggregating persistent memory and controlling them remotely: An exploration of passive disaggregated key-value stores. In *ATC 2020*.