記憶體內處理系統中的高效去重分塊方法

Compute-Efficient Chunking for Deduplication on Processing-in-Memory Systems

組別：D-2
組員：陳星宇
指導老師：何建忠

Backgrounds and Motivations

近年來資料量爆炸式的增長，對儲存系統造成了極大的負擔，其中資料去重被作為主要的應對方式，而資料去重中最耗時的階段即為「資料分塊 (Data Chunking) 」。 **佔比53%！**

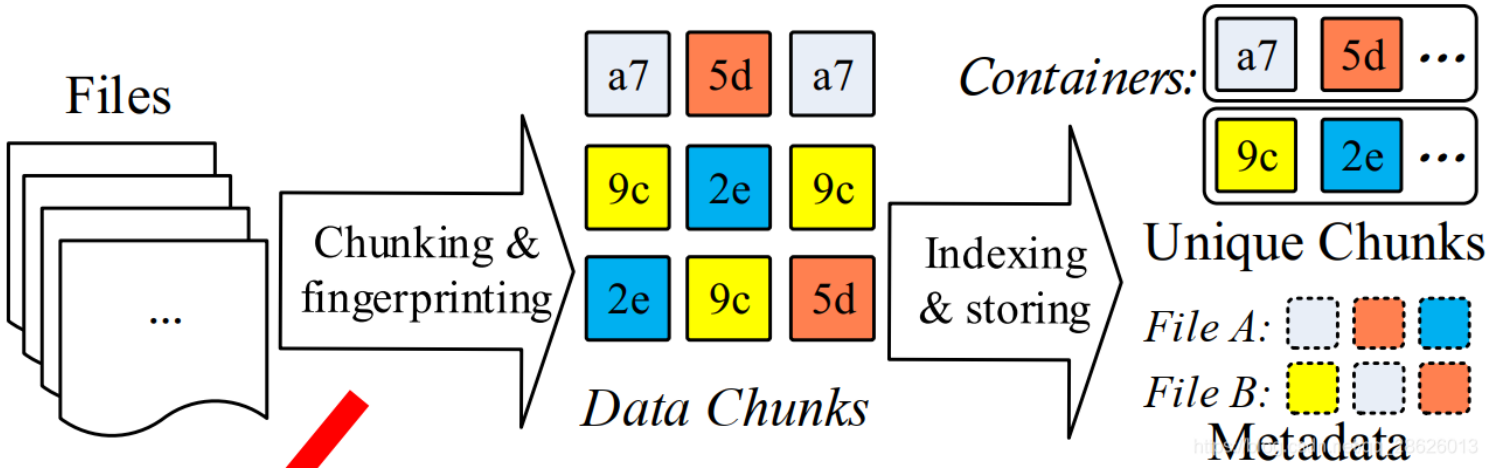

Fig. 1. The process of data deduplication.

Backgrounds and Motivations

而現今資料分塊的演算法即便經過優化，
在傳統架構(Von Neumann architecture)下仍
受限於資料必須從記憶體中提取到處理器
才能處理的限制。
記憶體內處理（PIM）技術減輕了傳統處理
器中心架構的資料移動開銷 [1]

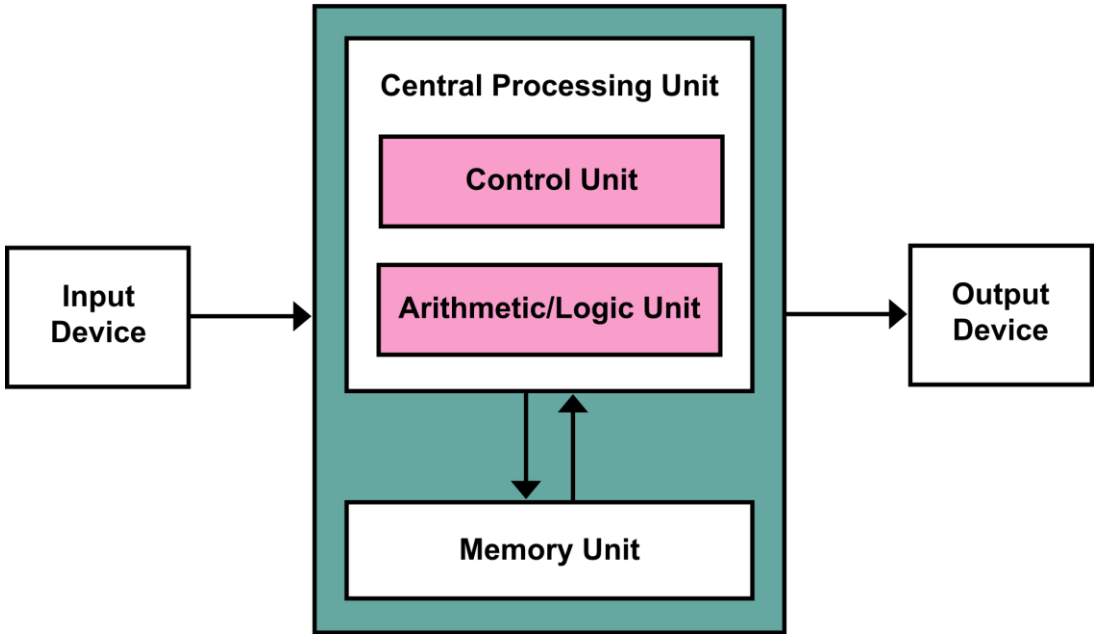

Fig. 2. The design concept of the von Neumann architecture.

[1] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," 2022.

Backgrounds and Motivations

UPMEM's DPU 是一種記憶體內運算（processing-in-memory, PIM）技術，透過直接在記憶體內執行運算來減少資料搬移。但是為 DPU 設計資料去重系統仍面臨多項特殊挑戰，包括 (1) 缺乏原生乘法支援、(2) DPU 之間資料共享受限，以及 (3) 顯著的 DPU-CPU 通訊開銷。

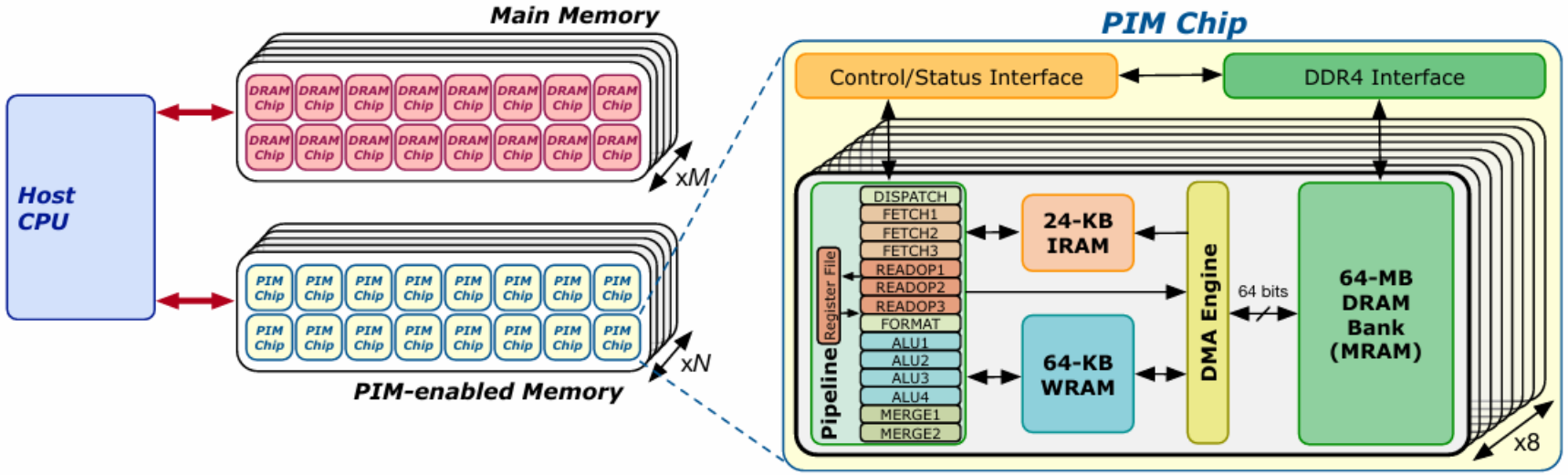

Fig. 3. UPMEM's DPU System Organization.

Methodology (1)

問題一：DPU缺乏原生乘法支援

解決方式：Skip-Ahead Local-Maximum Chunking, SALM

用較適合 DPU 的位元比較，尋找區間中的最大值作為分塊邊界，並透過 skip-ahead 機制減少重複的窗口檢查，使其可以更有效率地平行化

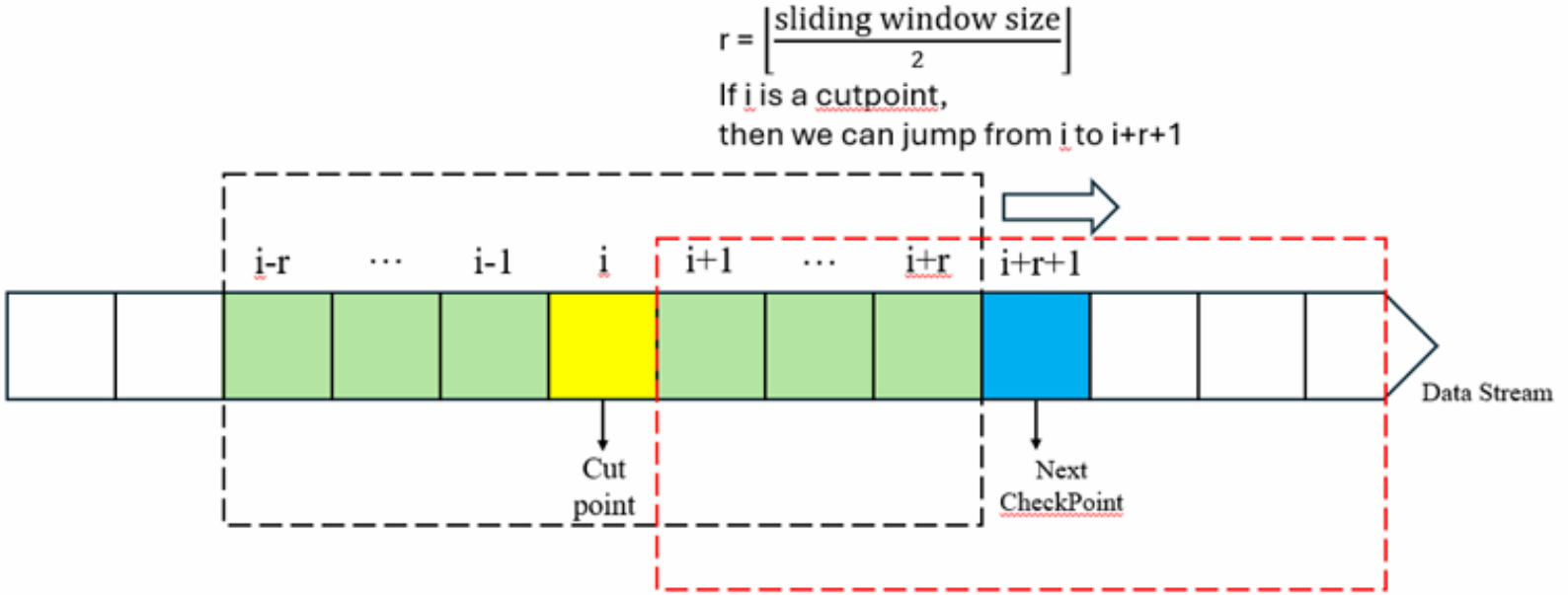

Fig. 4. Skip-Ahead Local-Maximum Chunking algorithm.

Methodology (2)

問題二：DPU之間資料共享受限

解決方式：Anti-Splitting File Segmentation, ASFS

因為檔案會被分配給多個 DPU 平行處理，如果只做單純切割，就可能在切割點附近失去必要的上下文，ASFS 會在不同資料段落之間加入重疊區域，讓每個 DPU 即使不能互相通訊，也能保留足夠資訊，維持和 CPU 版本一致的分塊結果。

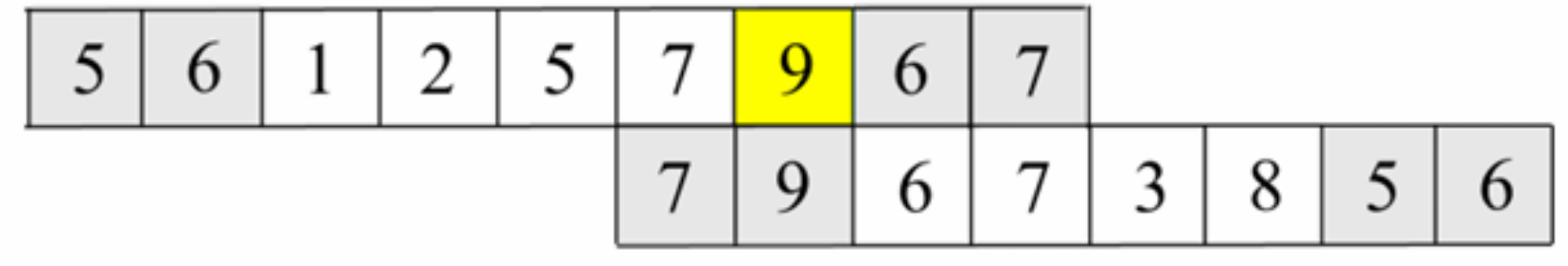

Fig. 5. Anti-Splitting File Segmentation, ASFS

Methodology (3)

問題三：減輕顯著的DPU-CPU通訊開銷

解決方式：Encoded Boundary Vector

DPU 找到分塊邊界後，不回傳大量原始資料，而是把邊界位置記成二進位向量，並將每 8 個bit壓縮成 1 個byte。透過這個方式，DPU 回傳到 Host 的資料量可以降到原本的八分之一，降低通訊開銷。

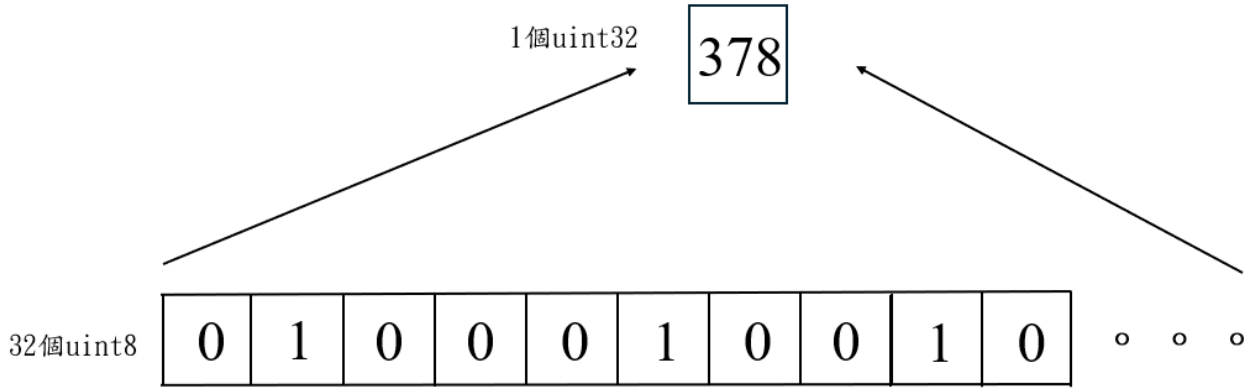

Fig. 6. Encoded boundary vector

Evaluations

Dataset：1/2/4/8/16/32/64 GB

Dataset：Linux/VM Image/新聞網快照/TAR

window size：129 bytes

CPU-DPU數量：32 ranks

CPU：MPI 16核

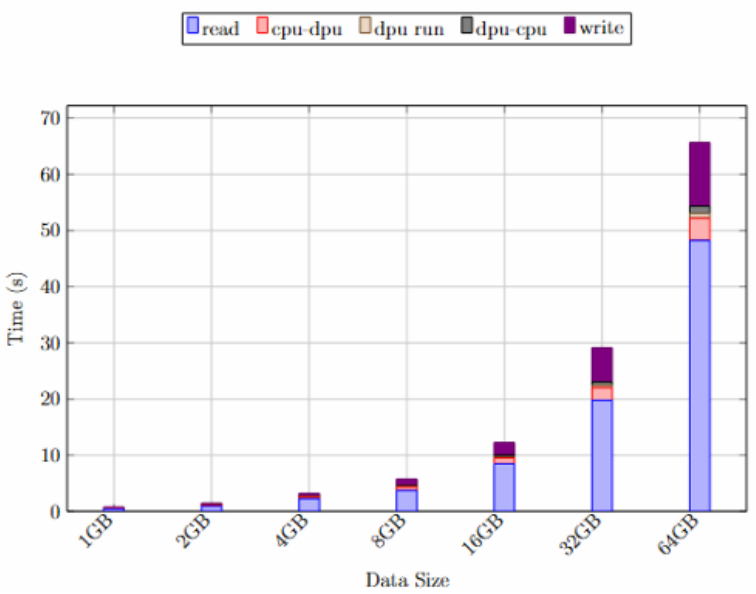

Fig. 1. Performance impact of different datasets under a fixed number of DPUs.

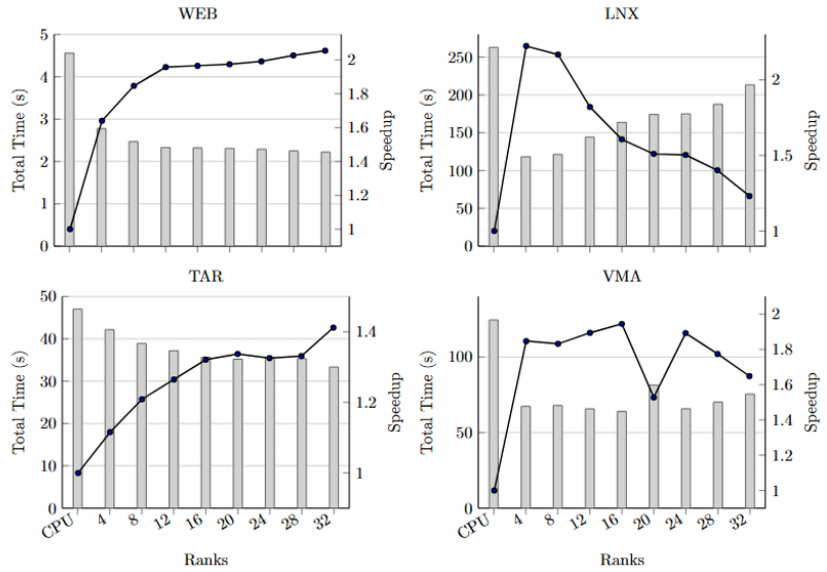

Fig. 2. Execution time and speedup across multiple real-world datasets.

結論：

(1) DPU在資料集成等比成長的情形，其時間、比例也能維持在等比成長

(2) 參數設定良好的情況下，DPU運算在各個資料集下相對於CPU運算都有顯著加速

(3) 在重複性高的資料集，去重率高，在重複性低的資料集，也能維持在一個穩定的去重率

Dataset	129	257	385	513	641	769	897	1025
WEB	94.33%	90.02%	88.26%	85.00%	77.15%	75.45%	73.77%	72.32%
LNX	91.34%	80.24%	75.43%	64.84%	60.56%	56.70%	52.95%	49.44%
TAR	14.34%	13.97%	13.69%	13.43%	13.14%	12.79%	12.35%	11.86%
VMA	69.67%	69.52%	69.36%	69.19%	68.98%	68.72%	68.40%	68.01%

Table 1. Deduplication ratio across multiple real-world datasets.

Conclusion

研究提出專為記憶體內運算（processing-in-memory, PIM）技術優化的資料去重方法，透過三大機制 SALM、ASFS 及資料壓縮解決了 PIM 架構在跨處理器中資料共享、硬體運算限制與通訊頻寬上的瓶頸，在確保 100% 分塊結果一致性的前提下，相較於傳統基於 CPU 的系統有顯著提升。

Thank you for listening