Die Verwendung von Phase-change Memory in modernen DBMS-Architekturen

Jan Stärz

University of Kaiserslautern

j_staerz11@cs.uni-kl.de

12.07.2013

Überblick

Motivation

- Parallelisierung von immer größer werdenden Aufträgen durch Multi-Threading benötigt zunehmend größeren Arbeitsspeicher
- DRAM schlecht skalierbar
- Skalierbare Alternative zu DRAM gesucht bei gleichem Laufzeitverhalten
- Vielversprechendster Kandidat ist Phase-change Memory (PCM)

Phase-change Memory (PCM)

Was ist PCM?

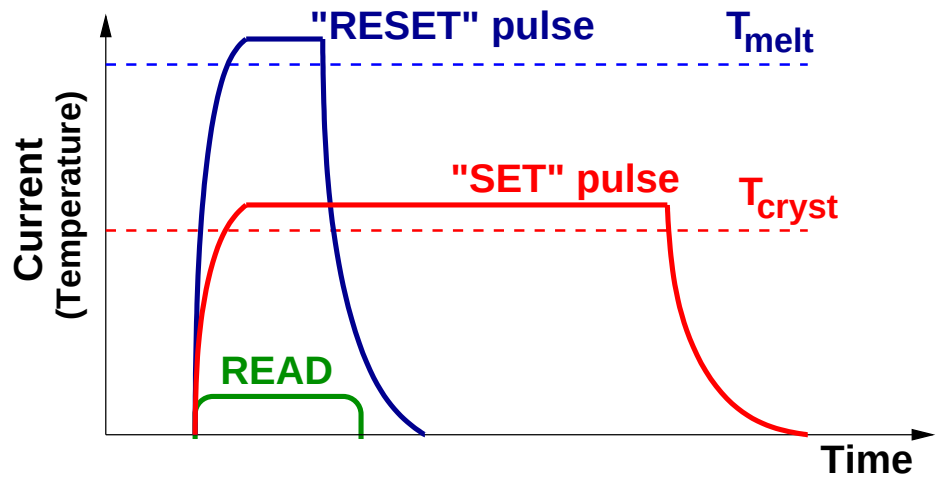
- Neuartiger Speicher
- Datenspeicherung geschieht über ein Phasenwechselmaterial, häufig $Ge_2Sb_2Te_5$, kurz GST

Phase	elektr. Widerstand	Zustand	logisch
kristallin	gering	SET	1
amorph	hoch	RESET	0

- ohne äußere Einflüsse bleibt die Phase erhalten
  $\rightarrow$ **persistent**

PCM-Operationen SET, RESET, READ

- **SET/RESET:** Zufuhr von elektrischem Strom unterschiedlicher Stärke und Dauer
- **READ:** Lesen der gespeicherten Daten über Widerstandsmessung des Materials

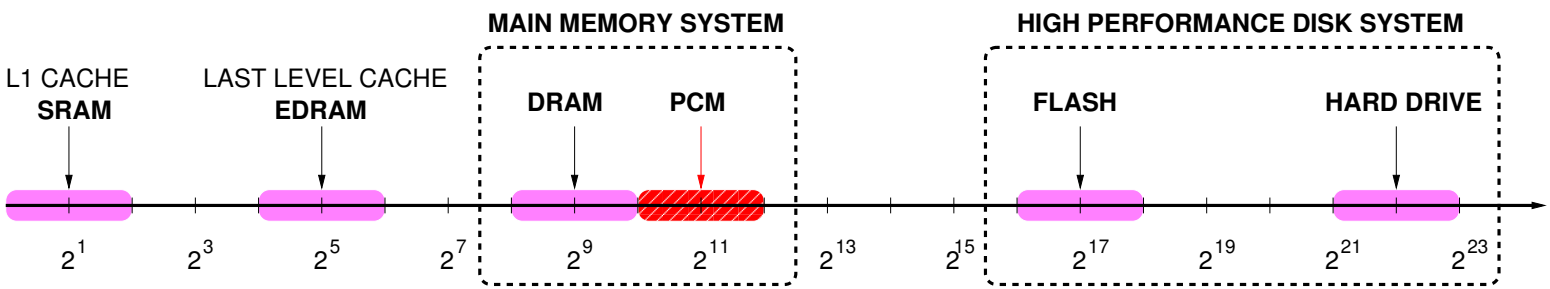


Einordnung in bekannte Speicher

	DRAM	**NAND Flash**	**HDD**	**PCM**
Dichte	1X	4X	N/A	2-4X
Leselatenz (Seitengröße)	20-50ns (64B)	~25μs (4KB)	~5ms (512B)	~50ns (64B)
Schreiblatenz (Seitengröße)	20-50ns (64B)	~500μs (4KB)	~5ms (512B)	~1μs (64B)
Idle-Verbrauch	~100 mW/GB	1-10 mW/GB	~10 W/TB	~1 mW/GB
Read-Verbrauch	0.8 J/GB	1.5 J/GB	65 J/GB	1 J/GB
Write-Verbrauch	1.2 J/GB	17.5 J/GB	65 J/GB	6 J/GB
Beständigkeit (in Schreibzyklen)	N/A	10^4-10^5	∞	10^6-10^8

Einordnung in bekannte Speicher

	DRAM	NAND Flash	HDD	PCM
Dichte	1X	4X	N/A	2-4X
Leselatenz (Seitengröße)	20-50ns (64B)	~25μs (4KB)	~5ms (512B)	~50ns (64B)
Schreiblatenz (Seitengröße)	20-50ns (64B)	~500μs (4KB)	~5ms (512B)	~1μs (64B)
Idle-Verbrauch Read-Verbrauch Write-Verbrauch	~100 mW/GB 0.8 J/GB 1.2 J/GB	1-10 mW/GB 1.5 J/GB 17.5 J/GB	~10 W/TB 65 J/GB 65 J/GB	~1 mW/GB 1 J/GB 6 J/GB
Beständigkeit (in Schreibzyklen)	N/A	10^4-10^5	∞	10^6-10^8



Typical Access Latency (in terms of processor cycles for a 4 GHz processor)

Vor- und Nachteile zusammengefasst

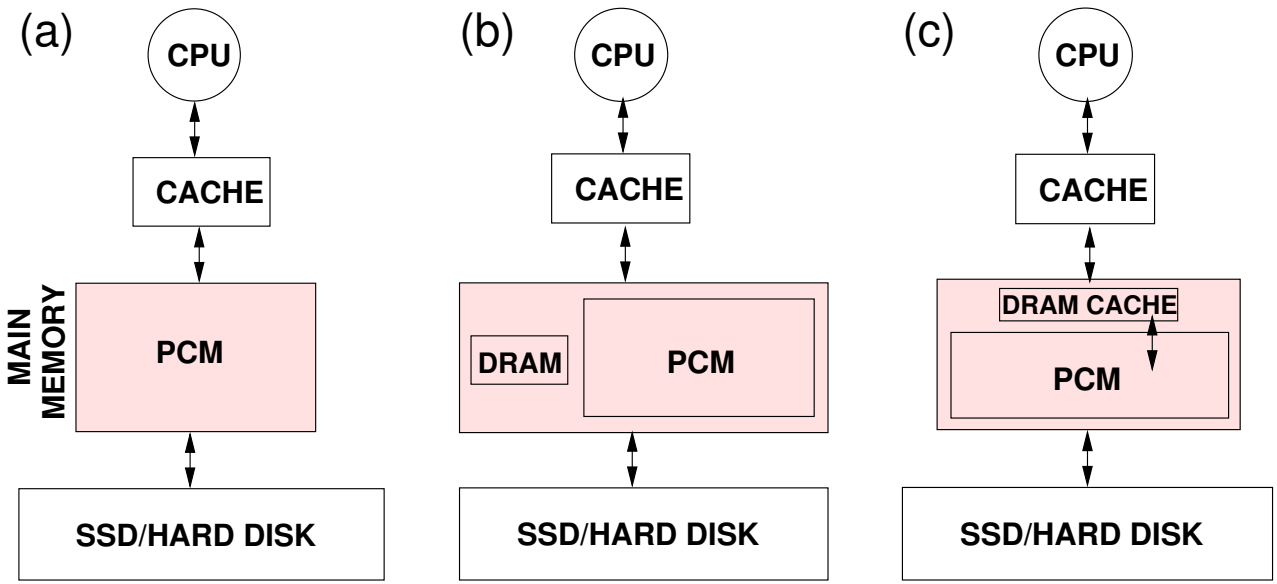
Vorteile:

- Persistente Speicherung
- Geringer Stromverbrauch
- Skalierbarkeit
- Byte-Adressierung
- Bit-Modifikation

Nachteile:

- Hoher Verschleiß (derzeitiger Stand)

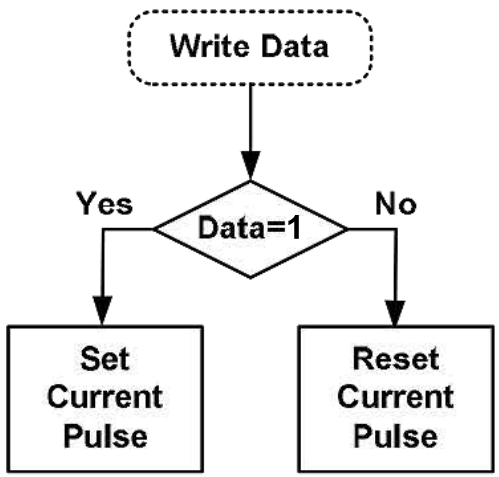
Vorstellbare Architekturen



Algorithmen zur Verschleißreduzierung

Data-Comparison Write Scheme

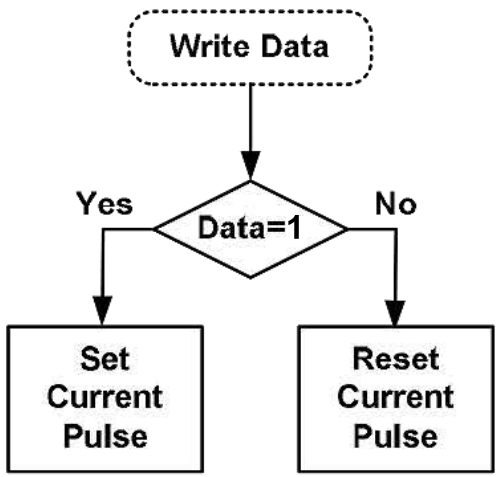
- **Ziel:** Reduzierung der Schreibzugriffe auf PCM



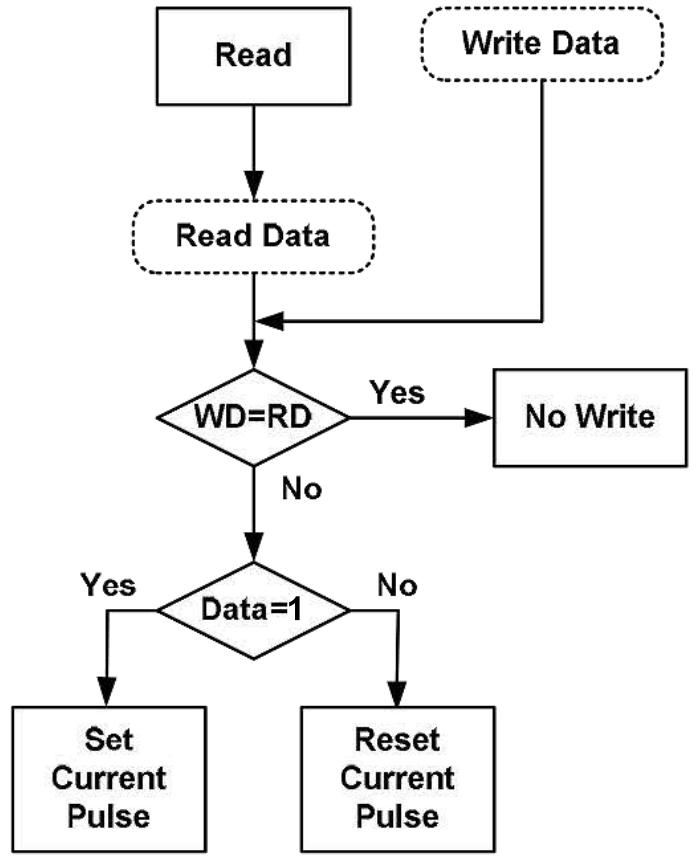
(a)

Data-Comparison Write Scheme

- **Ziel:** Reduzierung der Schreibzugriffe auf PCM



(a)



(b)

Seitenersetzungsstrategie LWD

Least wear-unit difference (LWD)

- **Ziel:** Seite ersetzen, die der neuen physikalisch am nächsten kommt
- Entscheidungskriterium ist die Wear-Unit-Differenz (WD)
  - WD zweier Seiten wird ermittelt, indem jedes korrespondierende Paar von PCM-Einheiten verglichen wird
  - voneinander verschiedene Paare werden gezählt
- Verwendung des Data-Comparison Write Scheme
- Zeitkomplexität von $O(B * P)$ für P die Seitengröße und B die Speicherkapazität
- **Nachteil:** Keine Beachtung der Anzahl bisher getätigter Überschreibungen pro Seite

Seitenersetzungsstrategie LPW

Least page wear (LPW)

- **Ziel:** Seite ersetzen, die den geringsten Verschleißzähler hat
- Bei der Ersetzung einer Seite im PCM wird der Verschleißzähler dieser Seite um die Anzahl der zu ändernden PCM-Zellen erhöht
- Zeitkomplexität von $O(B)$
- **Nachteil:** Verschleißverteilung innerhalb einer Seite unbeachtet

Seitenersetzungsstrategie LFM

Least frequently modified (LFM)

- **Ziel:** Seite ersetzen, die den niedrigsten Modifikationszähler besitzt
- Ähnlich dem Algorithmus least frequently used (LFU)
- Verdrängung einer Seite führt zur Inkrementierung des Zählers
- Verschleißzähler bleibt nach der Verdrängung bestehen
- Zeitkomplexität von $O(B)$
- Analog least recently modified (LRM) vorstellbar

Record-Swapping-Algorithmus

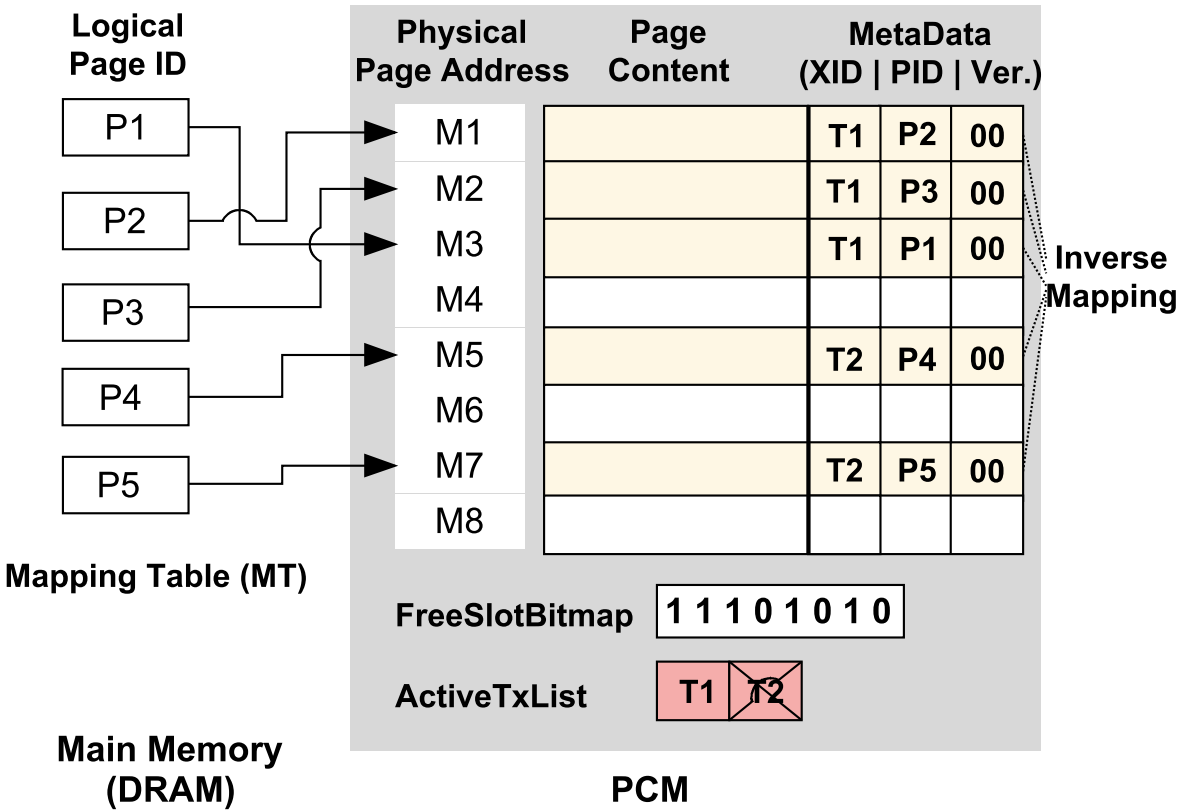
- **Ziel:** Schreiblastverteilung über das gesamte PCM-Modul, Vermeidung von hochfrequentiertem Schreiben in immer dieselbe PCM-Stelle
- Ausgewählter Speicherinhalt wird ersetzt, wenn er kälter als der zu schreibende Datensatz ist
  - in diesem Fall wird zuvor der zu ersetzende Datensatz an eine freie Speicheradresse verschoben
- Swap, wenn $XID_c - XID_n \geq \delta$
- Fest gewähltes δ nicht optimal
- Lebensdauerverlängerung des PCM um das 5-fache bei 34% mehr Schreibzugriffen

Anwendung PCMLogging

Datenstrukturen

- **Mapping Table (MT):** Abbildung logischer Seiten-IDs zu physischen PCM-Adressen
- **Inverse Mapping:** Teil der Seitenmetadaten, Rekonstruktion der initialen Mapping Table beim Systemstart
- **FreeSlotBitmap:** Speichert freie PCM-Speicherplätze
- **ActiveTxList:** Speichert alle Transaktionen, die noch laufen und dirty pages im PCM besitzen
- **Transaction Table (TT):** Beinhaltet alle Transaktionen und deren dirty pages im DRAM oder PCM
- **Dirty Page Table (DPT):** Hält Referenzen auf vorher gültige Versionen von dirty pages laufender Transaktionen

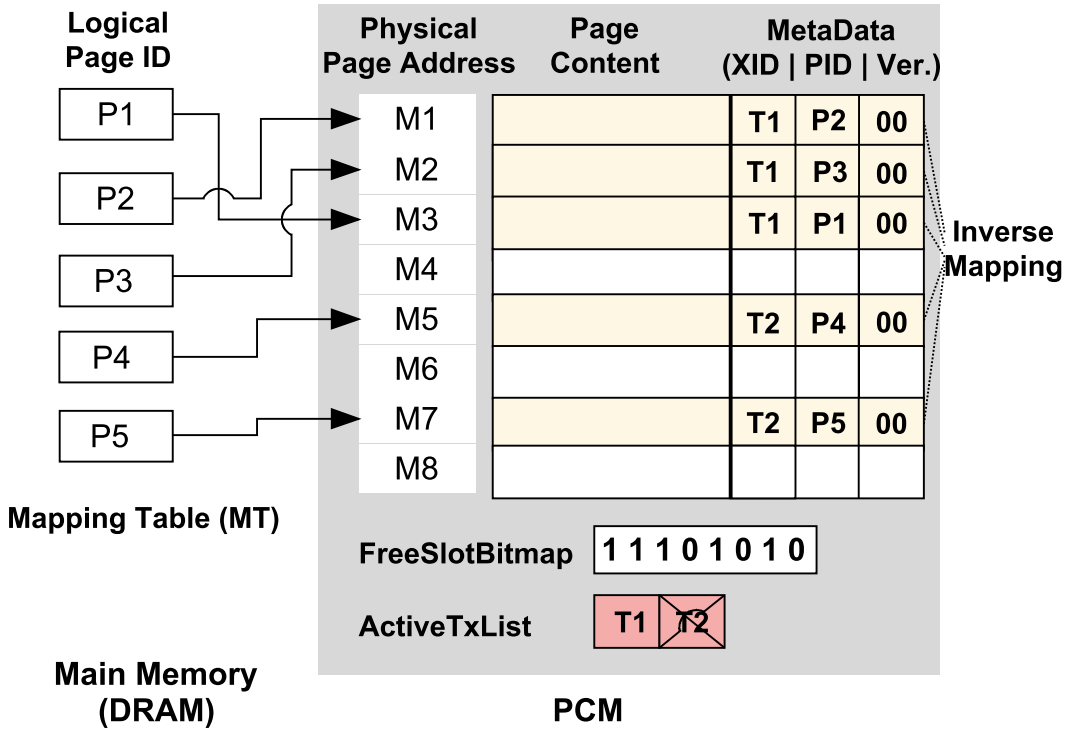
Datenstrukturen



Grundlagen

- Nur aus dem DRAM verdrängte dirty pages oder Seiten einer Transaktion, die mit Commit abschließt, werden im PCM gespeichert
- Shadow-Paging (siehe Zweck Dirty Page Table)
- Seite einer Transaktion wird erst in den PCM geschrieben, wenn XID in die ActiveTxList aufgenommen wurde
- XID wird erst wieder entfernt, wenn ein Commit oder Abort getätigt wurde
- Auf den Externspeicher werden nur Seiten erfolgreich abgeschlossener Transaktionen zurückgeschrieben

Grundlagen - Beispiel

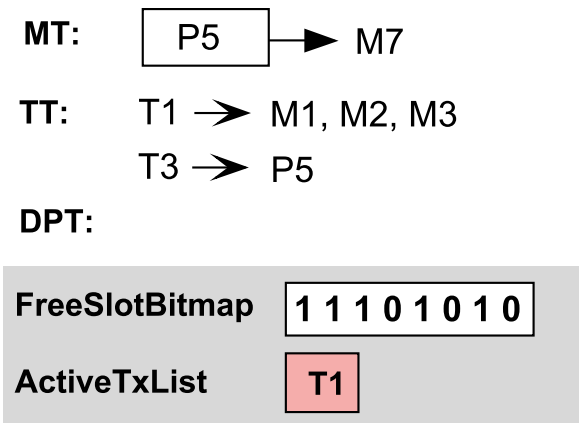


- T1 war noch nicht abgeschlossen, T2 schon
- Anpassung der FreeSlotBitmap zu '00001010'

Seitenebene - Verdängung zum PCM

- **Verdrängung modifizierter Seiten zum PCM:**
  - XID wird der ActiveTxList hinzugefügt
  - Kopie M' von Seite M wird out-of-place erstellt
  - Dirty Page Table und Transaction Table erhalten spezifische Einträge
  - Mapping Table wird aktualisiert

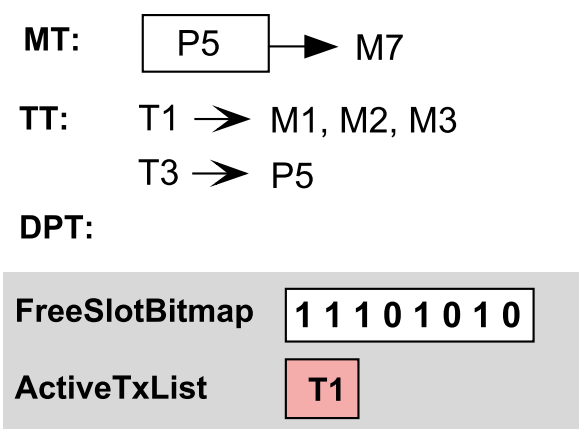
Verdängung zum PCM - Beispiel



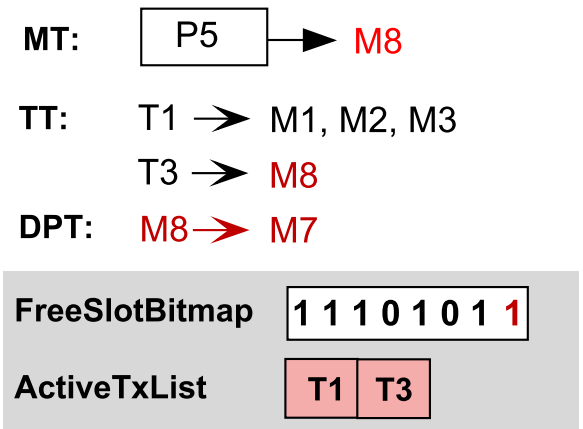
a) Before flushing dirty page

- (a) T1 in Arbeit, T2 bereits abgeschlossen

Verdängung zum PCM - Beispiel



a) Before flushing dirty page



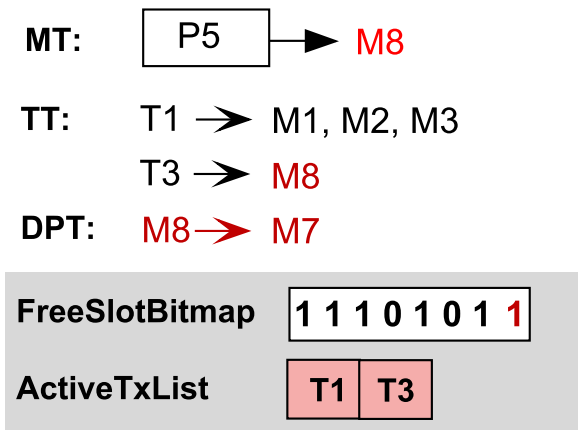
b) After flushing dirty page to M8

- (a) T1 in Arbeit, T2 bereits abgeschlossen
- (b) Verdrängung von P5, Kopieerstellung in M8

Seitenebene - Commit

- **Commit:**
  - Jede noch im DRAM befindliche Seite wird zum PCM zurückgeschrieben
  - XID wird aus der ActiveTxList entfernt
  - Anpassung der FreeSlotBitmap mithilfe der Dirty Page Table
  - Aktualisierung der Dirty Page Table und der Transaction Table

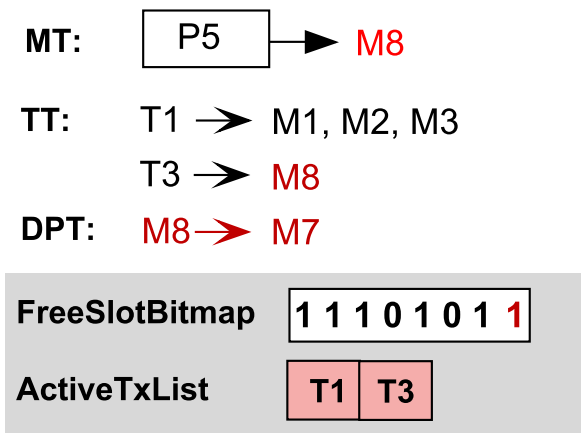
Commit - Beispiel



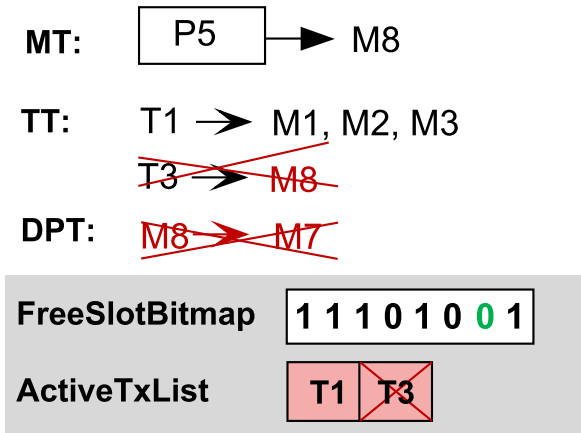
b) After flushing dirty page to M8

- T3 möchte mit einem Commit abschließen

Commit - Beispiel



b) After flushing dirty page to M8

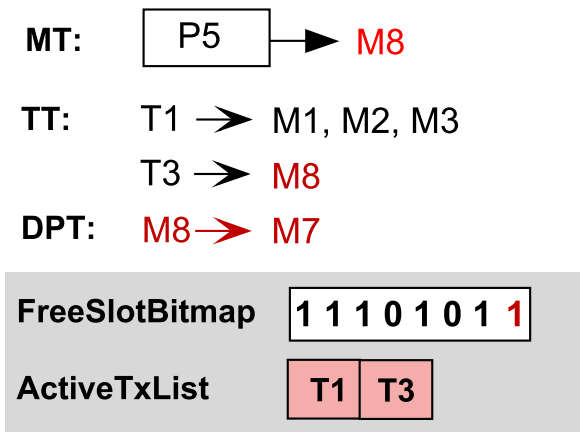


c) Case 1: T3 is committed

- T3 möchte mit einem Commit abschließen
- Alle zugehörigen Seiten bereits im PCM
- Vorher gültige Version von P5 in M7 wird verworfen

- **Abort:**
  - Alle dirty pages der Transaktion werden verworfen
  - Vorherige Versionen der dirty pages werden mithilfe der Mapping Table wiederhergestellt
  - XID wird aus der ActiveTxList entfernt
  - Transaction und Dirty Page Table werden aktualisiert

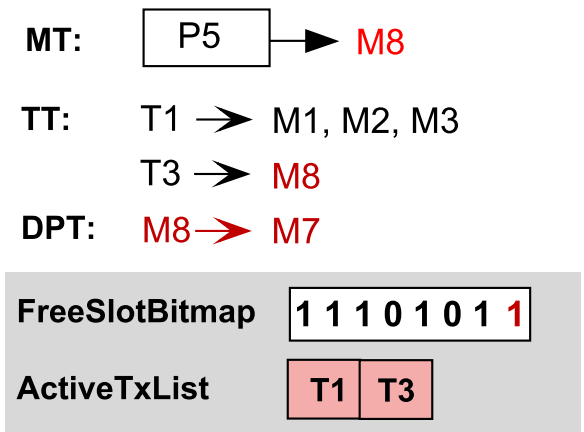
Abort - Beispiel



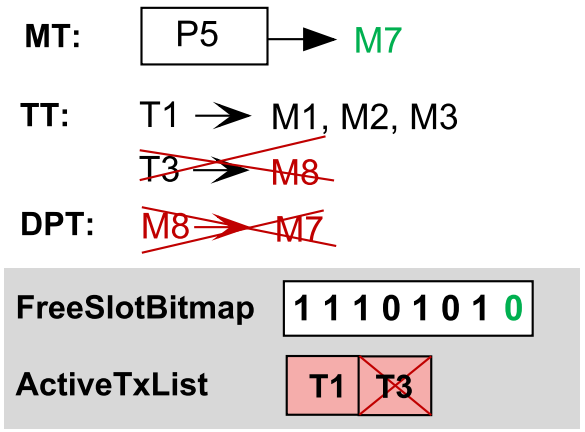
b) After flushing dirty page to M8

- Abort von T3

Abort - Beispiel



b) After flushing dirty page to M8



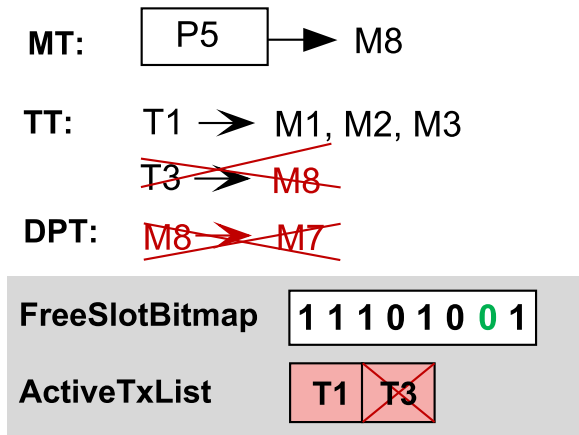
d) Case 2: T3 is aborted

- Abort von T3
- M8 wird verworfen, Verweis auf M7 wird wiederhergestellt
- Zugehörige Einträge werden gelöschen

Seitenebene - Recovery

- **Recovery:**
  - Überprüfung aller im PCM befindlichen Seiten mithilfe der FreeSlotBitmap
  - Jede dirty page einer zum Systemabsturz laufenden Transaktion wird verworfen
  - Aus den übrig gebliebenen Seiten wird die Mapping Table rekonstruiert

Recovery - Beispiel



c) Case 1: T3 is committed

- T3 aus der ActiveTxList entfernt, M7 noch **nicht** verworfen
  ↪ M8 hat höhere Versionsnummer

Satzebene - Datenstrukturen, Grundlagen (1)

- Speicherplätze des PCM werden auf die Satzgröße angepasst
- Sätze erhalten zusätzliches Metadatenfeld SNO (Slot-Nummer), die mit der Seiten-ID die Satz-ID (RID) bildet
- Mapping Table beinhaltet weiterhin modifizierte Seiten, jedoch speichert jede Seite Referenzen auf dirty records
- Dirty Page Table beinhaltet weiterhin Seiten, jedoch mit Verweisen zu Vorgängerversionen der jeweiligen Sätze

Satzebene - Datenstrukturen, Grundlagen (2)

- Zusätzliche Liste für jede Seite, die zugehörige, im DRAM befindliche dirty records speichert
- Transaction Table unterhält eine Liste aller laufenden Transaktionen und deren dirty records
- Wird eine ganze Seite angefordert, wird diese vom Externspeicher geladen und mit den zuletzt gültigen records im PCM zusammengeführt
- Ähnliches Prinzip bei der Verdrängung von Sätzen zum Externspeicher während des Betriebs und in Systemleerlaufzeiten

Satzebene - Verdängung zum PCM

- **Verdrängung modifizierter Sätze zum PCM:**
  - XID wird der ActiveTxList hinzugefügt
  - Kopie t' von Satz t wird out-of-place erstellt
  - Falls bereits eine Kopie vorliegt, wird diese verworfen (Kopie stammt von der gleichen Transaktion)
  - Dirty Page Table und Transaction Table erhalten spezifische Einträge
  - Mapping Table wird aktualisiert

Satzebene - Commit

- **Commit von Transaktion T:**
  - Jeder dirty record von T wird aus dem DRAM verdrängt
  - T wird aus der ActiveTxList entfernt
  - Vorherige Versionen der dirty records von T werden verworfen
  - Einträge von T werden aus der Transaction Table und der Dirty Page Table entfernt

Satzebene - Abort

- **Abort von Transaktion T:**
  - Alle dirty records von T im DRAM oder PCM werden verworfen
  - Wenn T noch in der ActiveTxList zu finden ist, werden die zuletzt gültigen records der dirty records von T wiederhergestellt
  - Mapping Table wird auf die vorherigen Versionen aktualisiert
  - Einträge von T werden aus der ActiveTxList, der Transaction Table und der Dirty Page Table gelöscht

Satzebene - Recovery

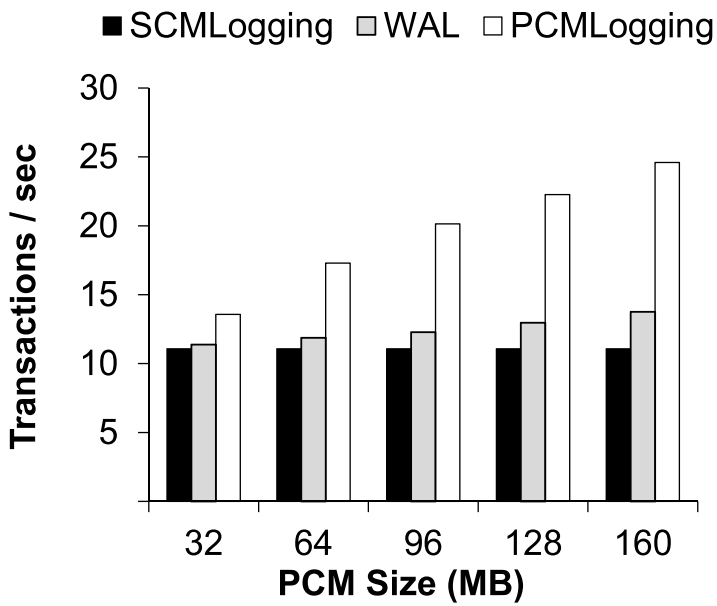
- **Recovery:**
  - Alle validen Sätze werden geprüft, ob die ändernde Transaktion zum Zeitpunkt des Systemabsturzes noch aktiv war oder eine höhere Version vorliegt
  - Erfüllt ein Satz diese Bedingung, wird er verworfen
  - Ansonsten wird für den Satz ein neuer Eintrag in der Mapping Table angelegt

Performance-Aspekte (1)

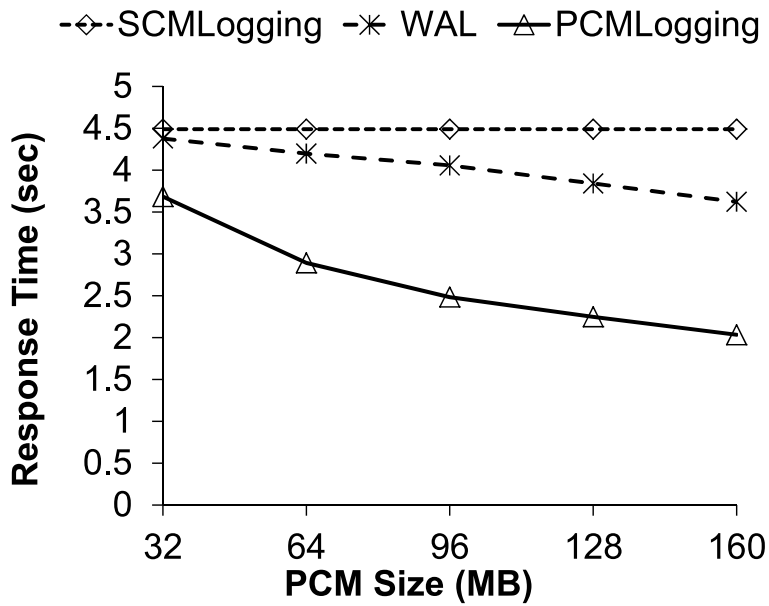
Verglichen wurden drei Logging-Schemata

- SCMLogging: PCM-Modul speichert ausschließlich das Transaktionslog
- WAL:
  - 1. Zone: Zwischenspeicher für dirty pages
  - 2. Zone: Zwischenspeicher für Logeinträge
  - Update-in-Place
- PCMLogging auf Satzebene

Performance-Aspekte (2)



(a) Transaction throughput



(b) Transaction response time

Performance-Aspekte (3)

- Hit-Wahrscheinlichkeit nimmt mit steigender Größe zu
- PCMLogging liest vor dem Zurückschreiben lediglich Sätze, keine ganzen Seiten
- Nur angeforderte Sätze werden ins PCM geschrieben
- PCMLogging vermeidet zeitaufwändige Operationen, wodurch Transaktionen ihre Sperren kürzer halten
- Commit-Prozess benötigt durch wenige Schreibzugriffe auf den PCM wenig Zeit

Performance-Aspekte (3)

- Hit-Wahrscheinlichkeit nimmt mit steigender Größe zu
- PCMLogging liest vor dem Zurückschreiben lediglich Sätze, keine ganzen Seiten
- Nur angeforderte Sätze werden ins PCM geschrieben
- PCMLogging vermeidet zeitaufwändige Operationen, wodurch Transaktionen ihre Sperren kürzer halten
- Commit-Prozess benötigt durch wenige Schreibzugriffe auf den PCM wenig Zeit

Außerdem:

- PCMLogging reduziert die Datenredundanz zwischen Logeinträgen und zwischengespeicherten Daten
- Recovery kommt ohne Checkpoints aus, da kein explizites Log
- Recovery-Prozess sehr simpel und effizient

Fazit

Fazit

- PCM vielversprechendste Speichertechnologie, um DRAM zu unterstützen, bzw. zu erweitern
- Starke Nähe zum DRAM und Persistenz eröffnen neue Möglichkeiten
- Schwerwiegender Nachteil: Hoher Verschleiß, gilt es zu reduzieren
- Erstes PCM-Modul von Numonyx (2010) und Ankündigung der 8GB-PCM-Modulproduktion von Samsung (vor kurzem)
- PCM als Teil der Speicherhierarchie eines Computersystems in naher Zukunft vorstellbar

Literatur

Literatur

- S. Gao, J. Xu, T. Härder, B. He, B. Choi, and H. Hu: PCMLogging: Optimizing Transaction Logging and Recovery Performance with PCM, submitted, 2013
- Y. Ou, S. Gao, J. Xu, and T. Härder: Wear-Aware Algorithms for PCM-Based Database Buffer Pools, submitted, 2013
- B.-D. Yang, J.-E. Lee, J.-S. Kim, J. Cho, S.-Y. Lee, and B.-G. Yu: A Low Power Phase-Change Random Access Memory using a Data-Comparison Write Scheme, in Proc. ISCAS, pages 3014-3017, 2007
- M. K. Qureshi, V. Srinivasan, and J. A. Rivers: Scalable High Performance Main Memory System Using Phase-Change Memory Technology, in Proc. ISCA, 2009
- S. Chen, P. B. Gibbons, and S. Nath: Rethinking Database Algorithms for Phase Change Memory, in Proc. CIDR, pages 21–31, 2011