A Fault Tolerance Transaction System Based on Persistent Memory

Projekt do predmetu Systémy odolné proti poruchám

Ladislav Šulák
Fakulta informačních technológii VUT v Brně
xsulak04@stud.fit.vutbr.cz

Úvod

Vysoko výkonné počítačové systémy, tiež známe ako *HPC (high performance computing)*:

- patria sem masívne paralelné a distribuované systémy, výpočetné clustre, superpočítače...
- čím viac rastie škálovateľnosť, tým väčší problém s poruchami.
- techniky patriace pod fault tolerance, teda odolnosti voči zlyhaniu, sú doležité čím ďalej tým viac. Sú založené na princípe ukladania checkpointov a vykonaní obnovy, tzv. *rollback-recovery* pomocou reštartu systému.
  - čoraz vyššia réžia spojená s kopírovaním pamäti, kvôli ukladaniu checkpointov.
  - navrhnuté optimalizácie nevyriešili problém spojený so zápisom do pamäte, viď [6].

Navrhnutá nová technika v oblasti odolnosti voči zlyhaniu v *HPC*. Založená na transakciách a *NVRAM*:

- metódy pracujúce s transakciami sa používajú najmä pre riešenie konfliktov so zdielanou pamäťou a z toho dôvodu bola nutnosť vytvoriť nový transakčný systém. Bol pomenovaný ako *NV-TS* [7].
- *NV-TS* garantuje, že aktualizovanie stavu aplikácie je atomické a trvalé. Pokiaľ systém zrazu spadne, napríklad počas vykonávania aplikácie, atomicita transakcie zaistí konzistenciu stavu aplikácie a po reštarte aplikácia môže pokračovať vo svojom behu.
- *NVRAM*, teda *Non-Volatile RAM* je energeticky nezávislá pamäť s priamym prístupom, napríklad EPROM, EEPROM, FLASH, SSD. Umožňuje, aby bolo trvalé úložisko pripojené k pamäťovej zbernici a dostupné pomocou inštrukcií *LOAD*/*STORE*.

NVRAM

Technológie poskytujúce granularitu prístupu k dátam a schopnosť uschovať dáta perzistentne aj počas systémového rebootu založené na *NVRAM*:

- *PCM*, fázová zmena pamäte [4]. Na zápis informácie nepoužívajú elektróny, ale zmenu stavu materiálu medzi kryštalickou a amorfnou fázou. Vzniká tak zmena elektrického odporu, ktorej vieme priradiť informáciu. V porovnaní s *DRAM* je operácia čítania u *PCM* asi 2x pomalšia a operácia zápisu niekoľko desiatok až 100 krát pomalšia.
- *Memristory* [1], elektrické prvky, chovajú sa ako premenlivé odpory, ktorých veľkosť závisí na množstve elektrického náboja, ktorý pretiekol. Odpor *memristora* je teda možné zvyšovať a znižovať elektrickým prúdom, ktorý do neho tečie po určitú dobu a určitým smerom. Ak prúd prerušíme, súčiastka si zapamatá poslednú hodnotu odporu.

NV-TS systém

- architektúra systému sa skladá z kompilátoru, jednotky riadenia pamäti a z komponenty, ktorá vytvára logovacie správy. Tento systém využíva aj operačný systém, ktorý poskytuje systémovú podporu a požaduje, aby boli aplikácie spustené transakčným spôsobom.
- v hardwarovej časti je znázornená pamäť typu *BPRAM* (označovaná niekedy tiež ako *PCM*).

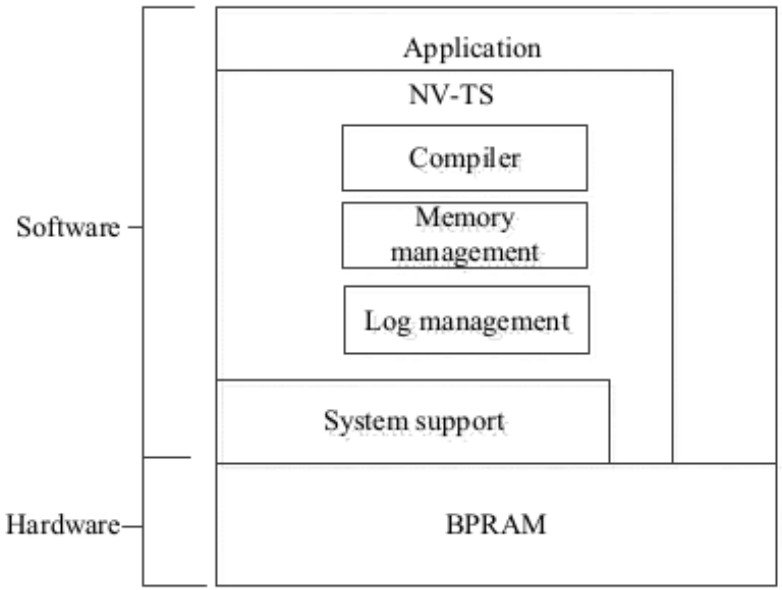

Schéma navrhnutého systému *NV-TS*.

Transakčný program a kompilátor

- kompilátor založený na ROSE[1], podporuje C, C++, Fortran, Java, Python, PHP a ešte zopár ďalších. Používa sa vo výskumnej oblasti (a tiež v U.S. Department of Energy) pre vykonávanie rôznych transformácií, analýz a optimalizácií zdrojového kódu. Je to *source-to-source* kompilátor, tzv. *transcompiler*, ktorý má na vstupe zdrojový kód v jednom jazyku a na výstupe zdrojový kód v inom jazyku. Oba jazyky ale musia operovať na rovnakej úrovni abstrakcie, na rozdiel od klasického kompilátoru.
- *NV-TS* systém obsahuje tento typ kompilátoru a prekladá kód v jazyku C do transakcií. Programátor musí obaliť segment kódu kľúčovým slovom *atomic*{...} a kompilátor vloží všetky ukazatele do pamäte do transakčného systému.

[1] ROSE projekt: `http://rosecompiler.org/`

Správca pamäte

- komponenta zodpovedná za prácu s pamäťovými operáciami z transakcií. Poskytuje rozhranie pre *nv_malloc*() a *nv_free*().
- pracuje atomicky, kvôli nutnosti manažovat perzistentnú pamäť. V opačnom prípade by vznikali perzistentné úniky pamäte.
- alokovanie pamäte bolo implementované na základe mechanizmov *dlmalloc* a *Hoard*.
  - *dlmalloc* je veľmi efektívny a používa sa v rôznych vstavaných systémoch a rôznych iných prostrediach ako napríklad v knižnici GNU jazyka C. Využíva obojsmerne zreťazený spojovaný zoznam pre správu voľnych pamäťových blokov, ktoré spojuje po každej pamäťovej operacií *free()*. Tento spôsob správy voľných pamäťových blokov nebol implementovaný.
  - *Hoardov* spôsob správy voľných pamäťových blokov rozdeľuje heap na niekoľko superblokov, pričom je každý rozdelený na rovnako veľké pamäťové bloky. V každom superbloku využíva jednoduchý zoznam pre správu voľných blokov a využíva deskriptor pamäťových operácií pre zaručenie atomicity operácií.

Správca pamäte

Navrhnutý systém poskytuje deskriptor pamäťových operácií pre každé alokovanie alebo uvoľnenie pamäte, čo má za následok prevenciu voči únikom pamäte. Deskriptor pamäťových operácií nahráva zdrojový a cieľový ukazateľ na pamäťový blok, typ operácie a tzv. *valid bit*. Ak operačný systém spadne počas alokovaní alebo uvolňovaní pamäte, môže zrušiť nedokončenú pamäťovú operáciu a po reštarte pamäť nebude stratená. Týmto spôsobom bola docielená prevencia straty pamäťových blokov.

Správca logovacích správ

- pre zaistenie atomicity transakcií vytvára táto komponenta 3 rozdielne typy správ:
  - správy o operáciách zápisu pre uchovávanie zmien v transakciách a premenných. Bol implementovaný tzv. *redo log*, ktorý zapisuje nové hodnoty a ich adresy počas vykonávania danej transakcie. Dôvod, prečo bol implementovaný tento druh správ je ten, že je jednoduché zaistiť zápisy z cache do pamäte. Je akurát potreba pripraviť dáta do vyrovnávacej pamäte pred transakčným commitom.
  - správy o alokovaní pamäte pre uchovanie informácií o pamäťových blokoch, ktoré transakcia alokuje. Ak transakcia zlyhá, pri následnom zotavovaní sa po zrušení transakcie tieto pamäťové bloky uvolnia.
  - správy o uvoľňovaní pamäte pre uchovávanie informácií o pamäťových blokoch, ktoré transakcia z pamäti uvoľňuje.
- pre zachovanie atomicity logovacieho zápisu sa pre každý logovací element využíva tzv. *valid bit*, ktorý je aktualizovaný vždy ako posledný.

Systémová a HW podpora

- Slúžia pre korektné obnovenie procesu a zaisťujú to, že aktualizácie do pamäte budú trvalé a v takom poradí, v akom boli zapísané. Systém *NV-TS* využíva podporu zo strany operačného systému a hardwaru:
  - Od operačného systému požaduje, aby pre aplikácie bol k dispozícií perzistentný pamäťový región.
  - Od hardwaru vyžaduje atomickosť operácie zápisu a musí byť zaistené, že v prípade zlyhania pamäte nie je v žiadnom medzistave žiadna aktualizácia. Okrem toho ešte systém vyžaduje, aby aktualizácie dosiahli pamäť *NVRAM* v správnom poradí. Momentálne existujú dva mechanizmy pre zaistenie toho, aby boli operácie zapísané v správnom poradí: Prvý, *Epoch barrier*, predstavuje nový druh súborového systému. Bol navrhnutý v príbuznej práci [3], ktorý ale vyžaduje špecializovaný hardware. V prácach *Mnemosyne* [2] a *CDDS* [5] bol navrhnutý princíp založený na pomocnej vyrovnávacej pamäti a špecializovaných inštrukciách procesoru, čo sa využíva aj v systéme *NV-TS*.

Vyhodnotenie výsledkov

NV-TS a normálny systém

Pri vyhodnocovaní výkonnosti tohto novo navrhnutého mechanizmu autori najprv porovnali odolnosť voči zlyhaniu systému *NV-TS* na aplikačnej úrovni. Potom vyhodnotili pamäťovú réžiu aplikácie na systéme *NV-TS* a normálnej aplikácie.
Je pravda, že systém *NV-TS* udržiava logovacie správy pre každú transakciu, čo zvyšuje réžiu spojenú s pamäťou. Autori pri experimentovaní porovnávali extrémne prípady spotreby pamäte medzi *NV-TS* a normálnou aplikáciou a porovnali celkovo alokovanú pamäť medzi nimi. Je pravda, že pamäťové stopy každej aplikácie sú rozlišné, no u väčšiny testovacích aplikácií bola réžia zvýšená len minimálne.

Záver

Výsledky ukázali, že systém *NV-TS* môže u *HPC* výrazne zlepšiť odolnosť voči zlyhaniu a spôsobuje malú pamäťovú réžiu.

Použitá literatúra

D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams.

The missing memristor found.

Nature, vol. 453, pages 80 – 83, 2008.

H. Volos, A. J. Tack, and M. M. Swift.

Mnemosyne: Lightweight persistent memory.

16th Architectural support for programming languages and operating systems (ASPLOS'11) Newport Beath, California, USA: ACM, pages 91 – 104, 2011.

J. C. E. B. Nightingale, C. Frost, E. I. B. Lee, and D. B. D. Coetzee.

Better I/O through byte-addressable, persistent memory.

22nd symposium on Operating systems principles (SOSP'09) Big Sky, Montana, USA, pages 133 – 146, 2009.

S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y. C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S. H. Chen, and H. L. Lung.

Phase-change random access memory: A scalable technology.

IBM Journal of Research and Development, vol. 52, pages 465 – 479, 2008.

S. Venkataraman, N. Tolia, P. Ranganathan, and R. H. Campbell.

Consistent and durable data structures for non-volatile byte-addressable memory.

9th USENIX Conference on File and Storage Technologies (FAST'11) San Jose, CA, USA: USENIX Association, pages 61 – 75, 2011.

X. Dong, N. Muralimanohar, N. Jouppi, R. Kaufmann, and Y. Xie.

Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exascale systems.

High Performance Computing Networking, Storage and Analysis (SC'09) Portland, Oregon, USA: ACM, 2009.

Xu Li, Kai Lu, Xu Zhou.

NV-TS: A Fault Tolerance Transaction System Based on Persistent Memory.

National University of Defense Technology, China, pages 221 – 224, 2012.