EPTI: Efficient Defence against Meltdown Attack for Unpatched VMs

Zhichao Hua, Dong Du, Yubin Xia, Haibo Chen, Binyu Zang

Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University

Meltdown - krótkie przypomnienie

- Obecnie produkowane procesory Intela wykonują ok. 200 instrukcji do przodu względem instrukcji wskazywanej przez RIP (tzw. speculative execution).
- Jeżeli któraś z tych 200 instrukcji naruszy ochronę pamięci to wyjątek jest zgłaszany dopiero wtedy, gdy RIP dojdzie do tej instrukcji. Do tego czasu wykonanie programu nie zostaje przerwane.

Meltdown - krótkie przypomnienie

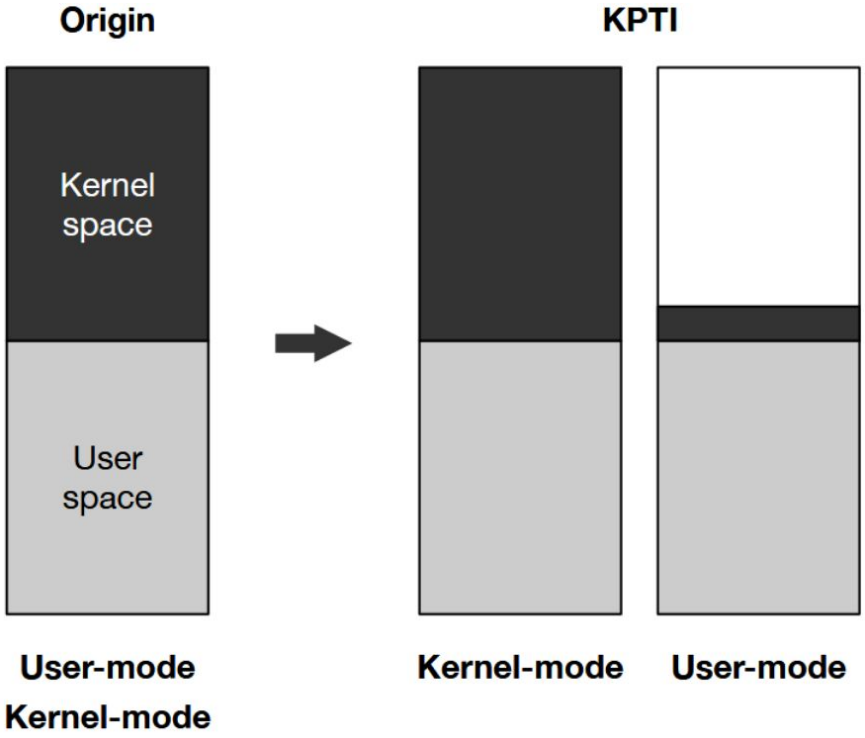
- Po zgłoszeniu przerwania wynik wykonania kolejnych instrukcji zostaje anulowany.
- Ale czy na pewno?
- Nie! Zostaje ślad w cache procesora, który można odczytać poprzez zmierzenie czasu dostępu do określonej strony.
- Anulowane instrukcje mogą korzystać z wartości odczytanej przez instrukcję, która wywołała wyjątek.

Meltdown - skutki

- Atakujący przy pomocy programu działającego w trybie użytkownika może odczytać dowolny bajt pamięci kernela.
- W kernelu Linux jest zmapowana cała pamięć fizyczna, więc atakujący może w ten sposób odczytać pamięć innych programów.

KPTI - ochrona przed Meltdown

- Każdy odczyt pamięci przechodzi przez tablicę stron.
- Stosujemy dwie tablice - jedną dla kernela, drugą dla procesu użytkownika (bez stron należących do kernela).
- Zmieniamy tablice przy przechodzeniu między trybami.



KPTI - wady

- Zmiana tablicy stron przy pomocy rejestru CR3 kosztuje 300 cykli procesora.
- Każda zmiana CR3 wymusza wyczyszczenie TLB, co jeszcze bardziej pogarsza wydajność.
- Wg. autorów pracy spadek może wynieść nawet 30%.
- W chwili publikacji pracy KPTI było dostępne od wersji 4.15 kernela Linux. Starsze wersje np. 4.4 jeszcze nie wspierają poprawki.

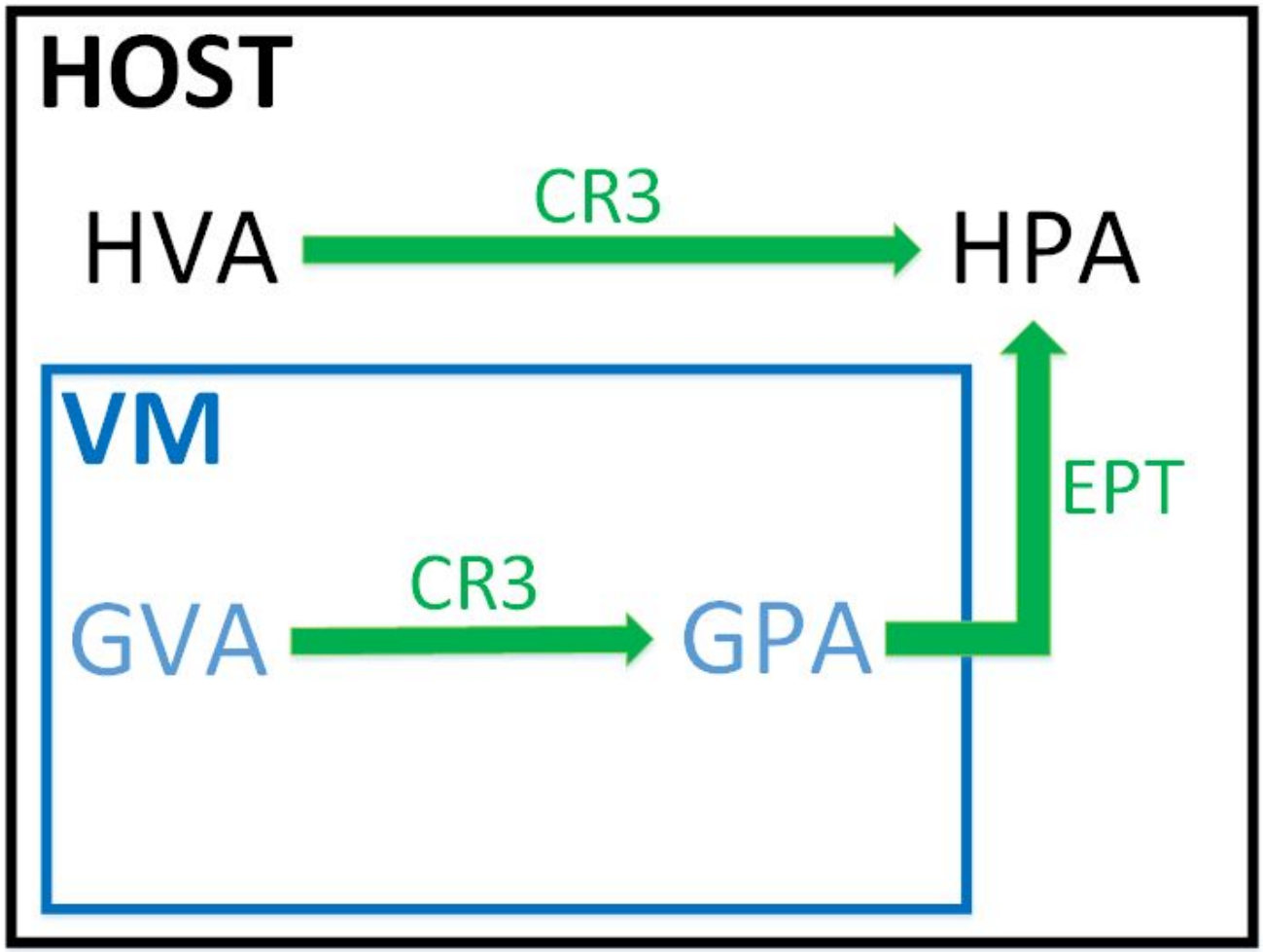
Problem

- Trzymamy serwery w wirtualizowanym środowisku, np. w chmurze Amazona.
- Nie możemy sobie pozwolić na aktualizację kernela do nowej wersji, np. dlatego, że oprogramowanie z którego korzystamy działa tylko na Ubuntu 12 albo nie możemy sobie pozwolić na długie przestoje.
- Chcemy być chronieni przed Meltdown.
- Nasza chmura wspiera live migration (na którą możemy sobie pozwolić) oraz korzysta ze stosunkowo nowych procesorów (Haswell, 2013).

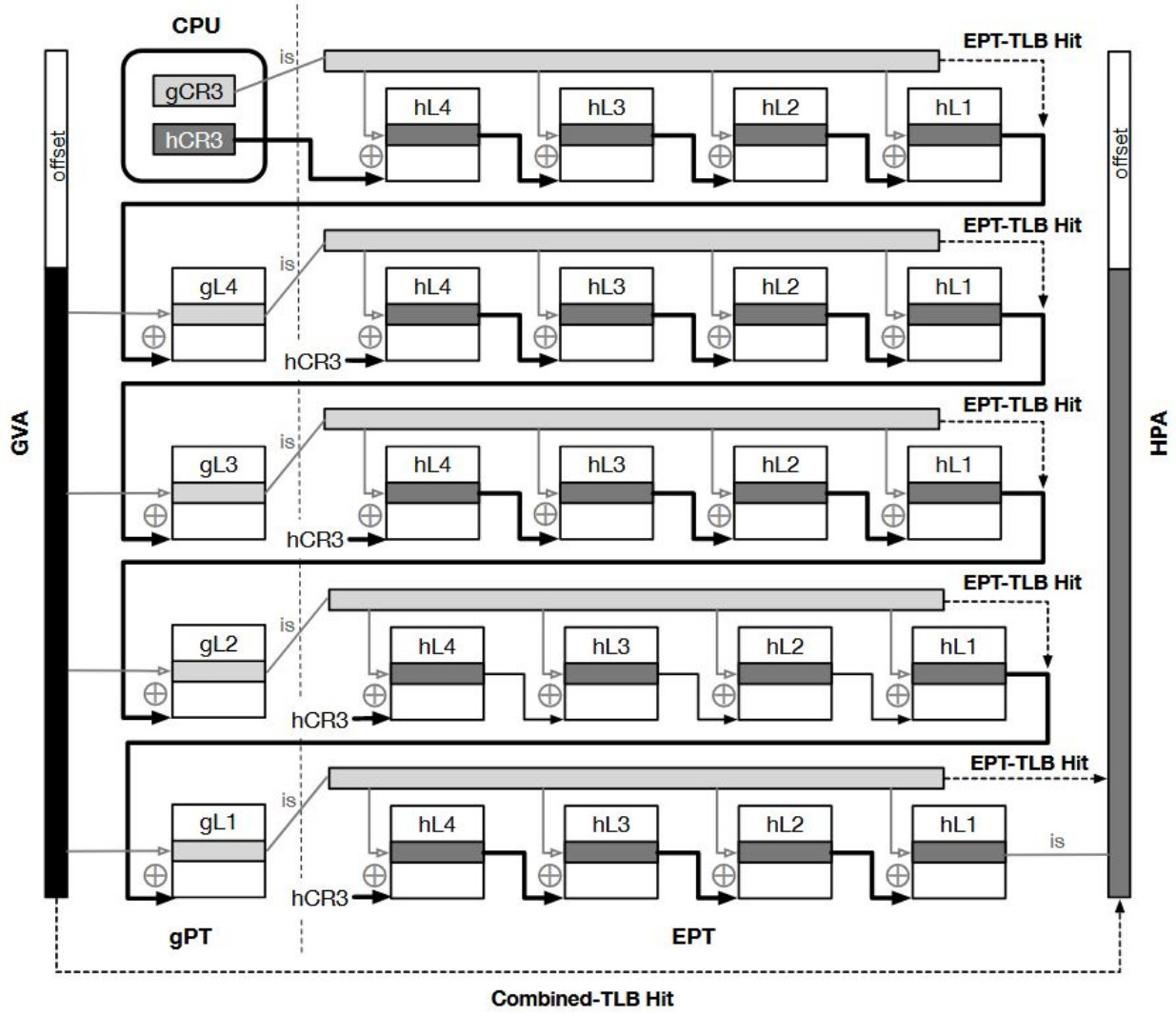
Pomysł

- Nie możemy nic zrobić z maszyną wirtualną, ale może dałoby się poprawić nadzorcę (np. KVM), by chronił nasze serwery przed Meltdown?
- Wówczas moglibyśmy przemigrować wirtualki na serwer zapasowy, wgrać poprawiony KVM i ponownie uruchomić obrazy maszyn na oryginalnym serwerze.
- Wgrywaniem patcha zajmowałby się dostawca chmurowy, więc nie stanowiłoby to problemu dla użytkownika chmury.

EPT - Extended Page Table



EPT - Extended Page Table



EPT - Extended Page Table

- Dzięki EPT (Nested Page Tables u AMD) host nie musi monitorować i ręcznie podmieniać wpisów w tablicy stron maszyny wirtualnej.

- Host może udostępnić maszynie wirtualnej kilka EPT do wyboru.
- Guest może w dowolnym momencie wywołać instrukcję VMFUNC zmieniającą aktywne EPT, zarówno w trybie użytkownika jak i w trybie kernela.
- Zmiana EPT trwa 160 cykli i zachowuje EPT-TLB, zmiana CR3 kosztuje 300 cykli i czyści TLB wszystkich EPT.

EPTI - Extended Page Table Isolation

- KPTI polega na stworzeniu dwóch osobnych tablic stron, po jednej dla kernela i procesu w trybie użytkownika.
- W EPTI tworzymy EPT-k i EPT-u i poprawiamy w locie wejścia i wyjścia kernela w taki sposób, by przełączały pomiędzy nimi.
- Wejścia kernela można znaleźć przez tablicę przerwań i rejestry MSR.
- Wyjścia kernela muszą zawierać konkretne instrukcje, np. sysretq.

Konstrukcja EPT-u

- Kopiujemy oryginalne EPT.
- Bierzemy tablicę stron L4 gościa, której adres jest zapisany w CR3.
- Odczytujemy górną połowę tak otrzymanej tablicy. Są w niej zapisane adresy tablic L3, które zawierają mapowania należące do kernela.
- Modyfikujemy EPT-u w taki sposób, by otrzymane wyżej adresy tablic L3 prowadziły do strony wypełnionej zerami (czyli oznaczały brak mapowania). W ten sposób blokujemy dostęp do fragmentu tablicy stron tłumaczącego adresy wirtualne kernela.
- Robimy wyjątek dla strony zawierającej trampolinę przełączającą pomiędzy trybami kernela i użytkownika.

Dzięki temu atakujący nie odczyta zawartości żadnego adresu wirtualnego należącego do kernela, bo nie będą podmapowane w tablicy stron.

Konstrukcja EPT-k

- Kopiujemy oryginalne EPT.
- Proces użytkownika może w każdej chwili przełączać pomiędzy EPT-k i EPT-u. Zatem wystarczy, żeby przed wykonaniem ataku Meltdown przełączył się na EPT-k i wtedy strony kernela znowu będą dla niego widoczne.
- Trik jest taki, by w EPT-k zamarkować strony użytkownika jako niewykonywalne. Dzięki temu po przełączeniu na EPT-k proces się zawiesi.

Meltdown może czytać strony do których nie jest uprawniony, ale nie może pobierać kodu do wykonania, który jest oznaczony jako niewykonywalny.

Aktualizacja EPT

- Tablice stron na które wskazuje CR3 mogą się zmieniać w czasie, np. w trakcie alokowania nowej strony. Nadzorca musi reagować na takie zmiany.
- Nadzorca monitoruje zmiany rejestru CR3 i w razie potrzeby poprawia EPT.
- Nadzorca przechwytuje zmiany na najwyższym poziomie tablicy stron gościa w celu aktualizacji listy wyzerowanych stron w EPT-u.

Możliwe optymalizacje:

- Nie przechwytywanie zmian wprowadzanych przez procesor przy oznaczaniu strony flagami Access i Dirty.
- Nie przechwytywanie wszystkich zmian CR3 (tj. ładowania już załadowanego adresu).
- Monitorowanie tablicy stron kernela na poziomie L3 zamiast L4.

Ewaluacja

- Maszyna: Intel Core i7-7700, 16GB RAM, Samsung 512GB SSD
- System gospodarza: Ubuntu 14.04, Linux 4.9.75 z zaimplementowanym EPTI w KVM
- Maszyna wirtualna: 4 vCPU, 8 GB RAM, Ubuntu 16.04, Linux 4.9.75

Ewaluacja

- Maszyna: Intel Core i7-7700, 16GB RAM, Samsung 512GB SSD
- System gospodarza: Ubuntu 14.04, Linux 4.9.75 z zaimplementowanym EPTI w KVM
- Maszyna wirtualna: 4 vCPU, 8 GB RAM, Ubuntu 16.04, Linux 4.9.75

- **Spoiler!** W większości testów wydajnościowych KPTI wprowadza narzut 12%-20%, a EPTI 6%-12% względem czystego kernela.

Ewaluacja - ochrona przed Meltdown

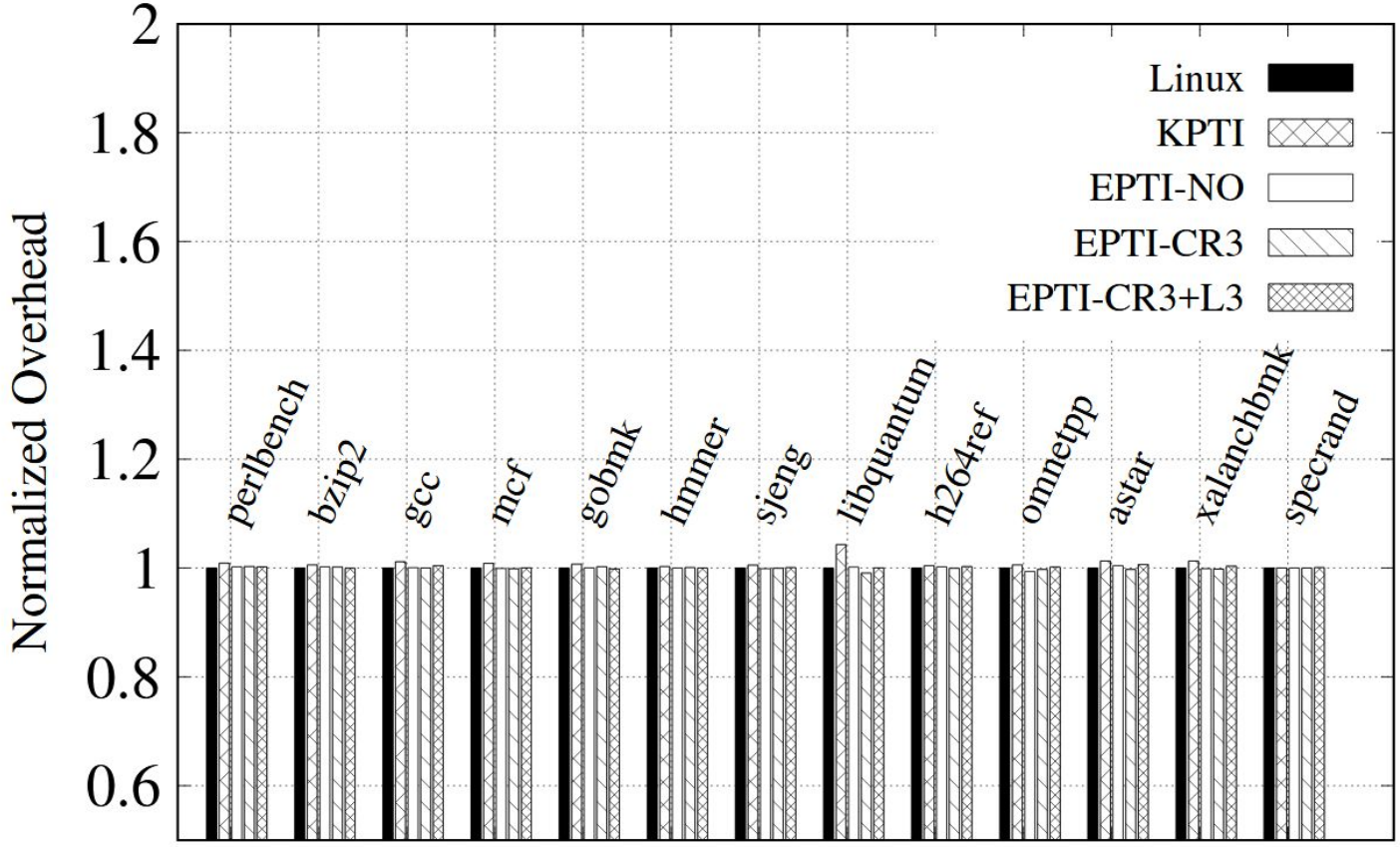
- Sposób ataku - ordynarne czytanie pamięci kernela z zarejestrowanym handlerem przechwytującym SEGFAULTy
- Wykradana wartość to linux_proc_banner składowany w przestrzeni kernela
- Atak przeprowadzony z powodzeniem na systemie bez KPTI i EPTI
- Zarówno KPTI jak i EPTI chronią przed atakiem

LMBench

Operation	Linux	KPTI	EPTI-No	EPTI-CR3	EPTI-CR3+L3
Null syscall	0.04	0.16	0.12	0.12	0.12
Null I/O	0.07	0.2	0.17	0.17	0.16
Open/Close	0.70	0.93	0.84	0.84	0.83
Signal Handle	0.68	0.81	0.76	0.76	0.76
Fork syscall	72.9	79	80	80	75
Exec syscall	212	243	242	234	221
ctsw 16P/64K	6.07	7.37	7.66	7.66	6.39

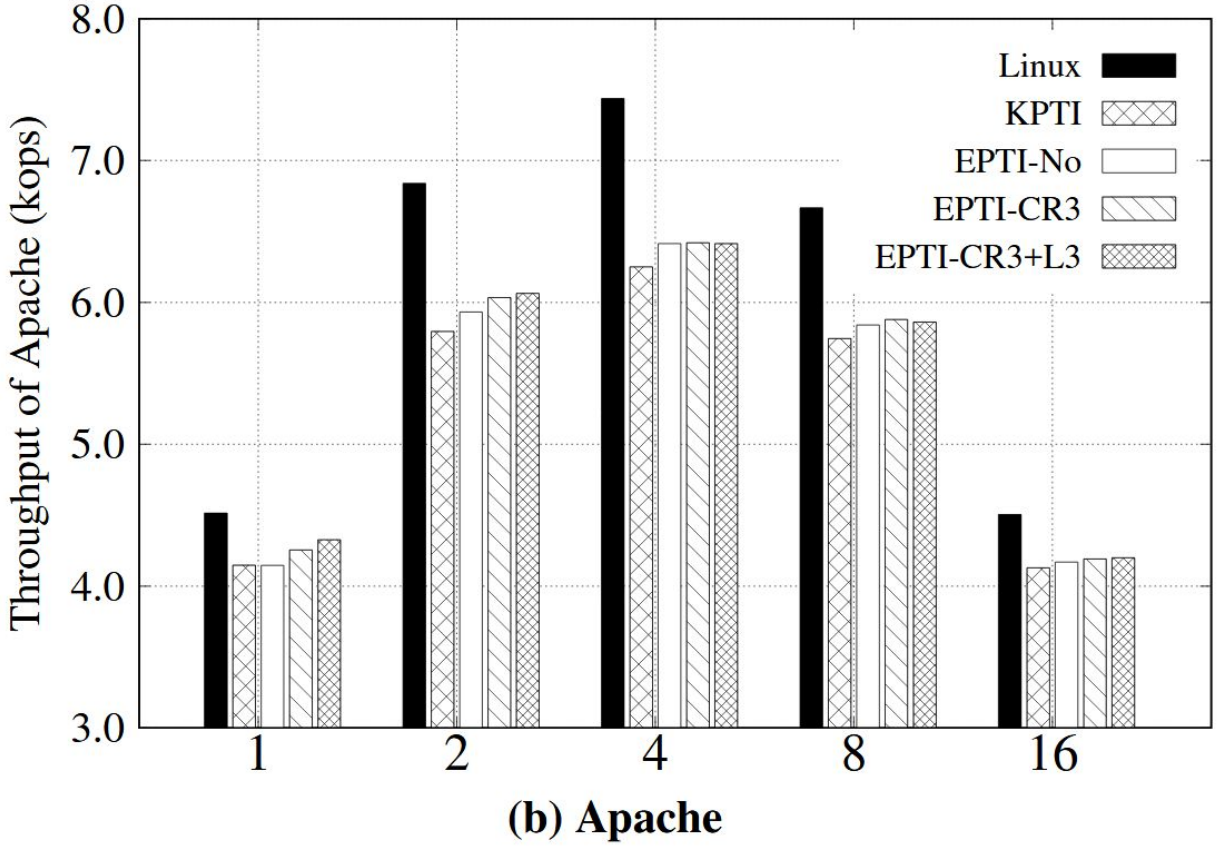
Wniosek: EPTI ma mniejszy narzut niż KPTI

SPEC_CPU 2006 - aplikacje z grupy INT



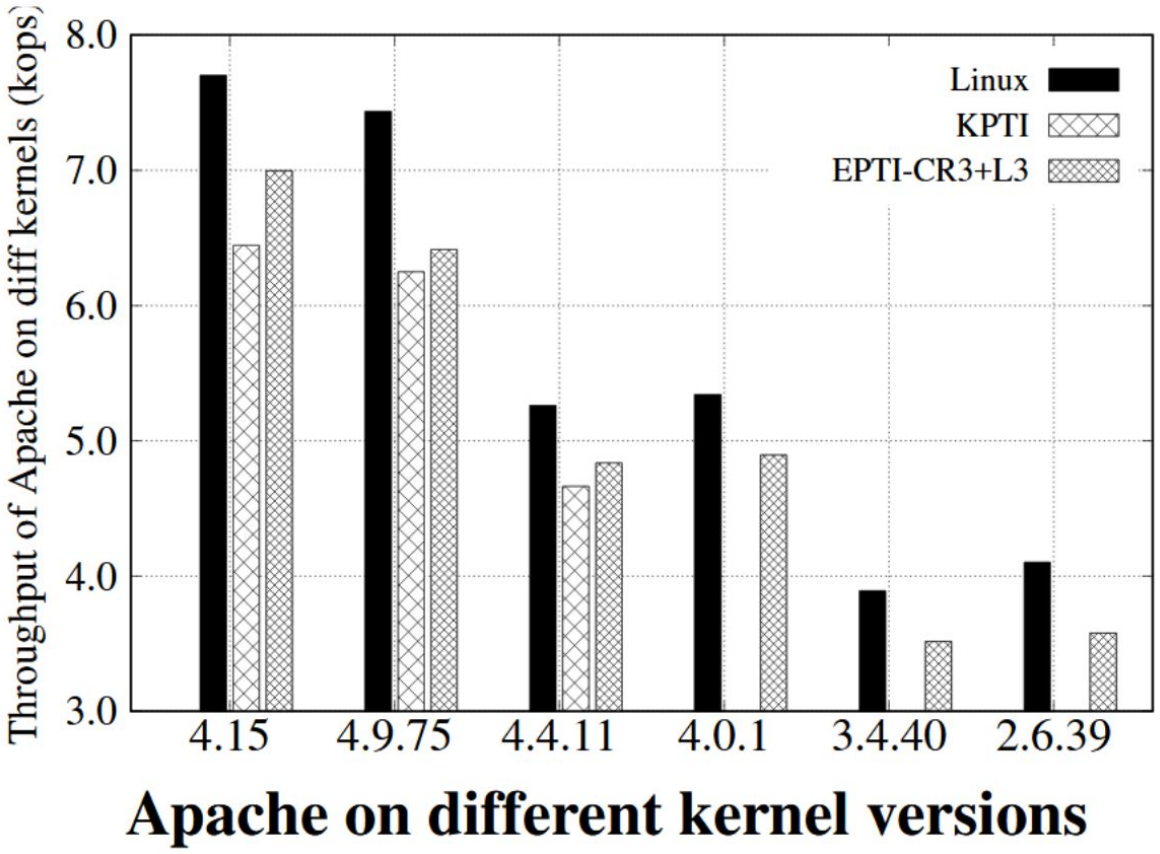
Wniosek: ani EPTI ani KPTI nie wprowadzają dużego narzutu w aplikacjach stricte obliczeniowych

Apache (ab - apache benchmark)



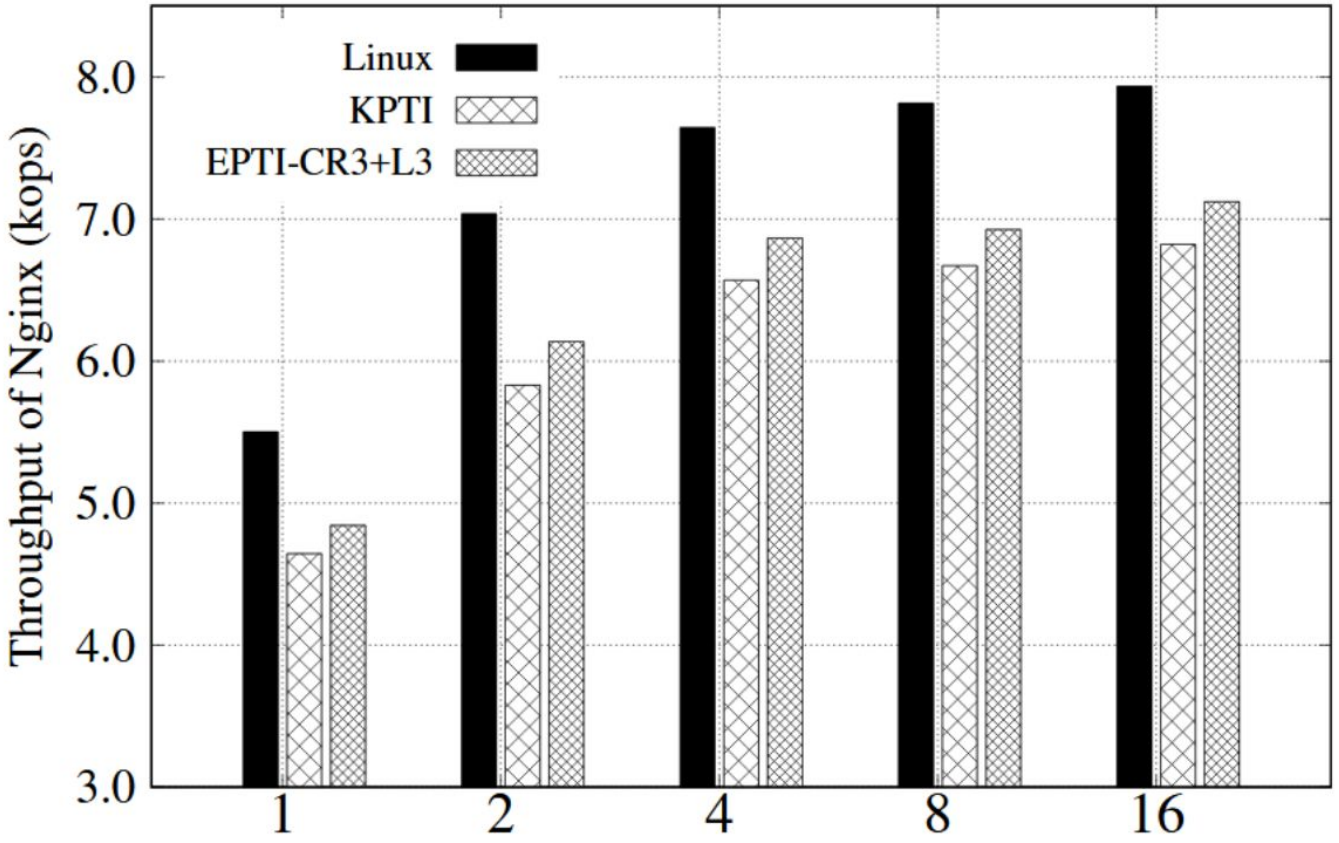
Wniosek: EPTI wprowadza spory narzut, ale i tak sprawuje się lepiej albo porównywalnie do KPTI (w zależności od wariantu)

Apache (ab - apache benchmark)



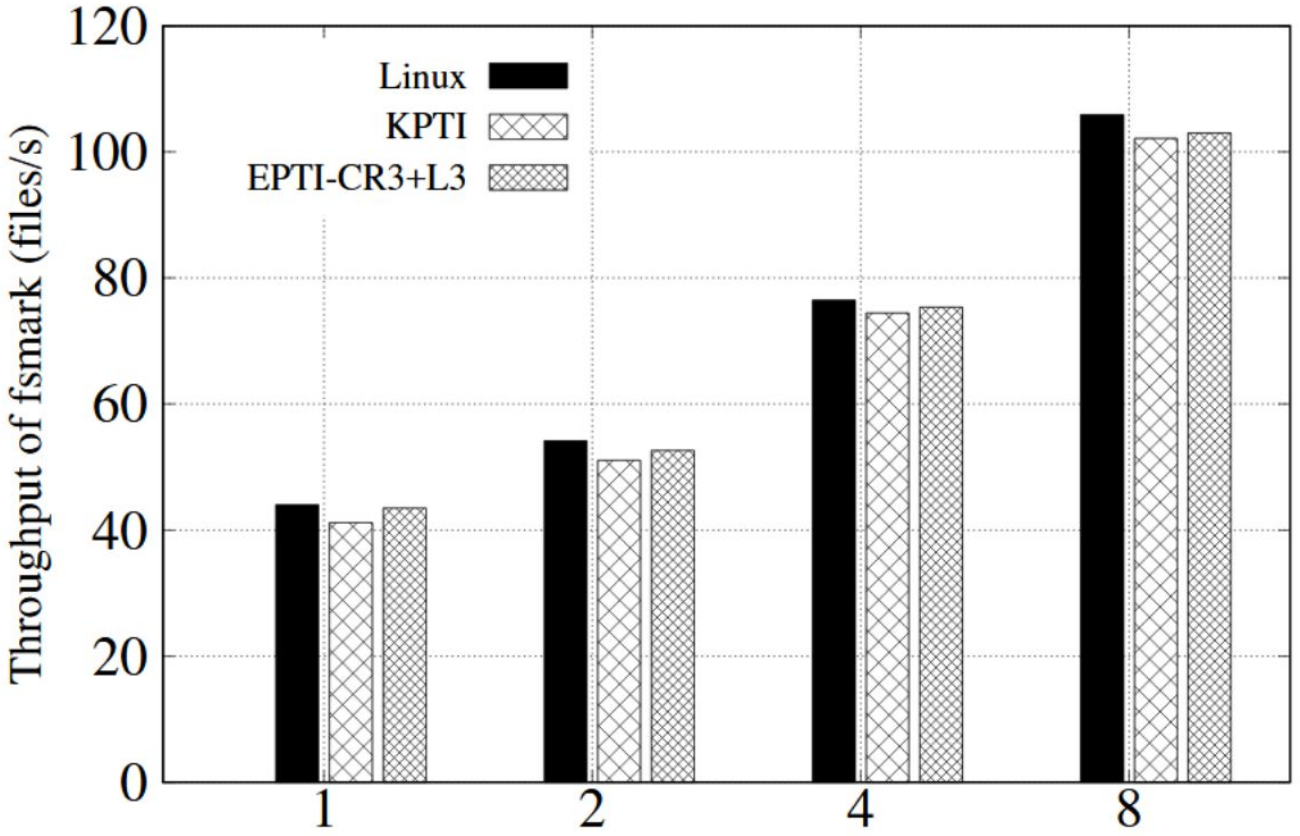
Wnioski: EPTI jest lepsze od KPTI. Im nowszy kernel tym lepszy wynik.

Nginx (ab - apache benchmark)



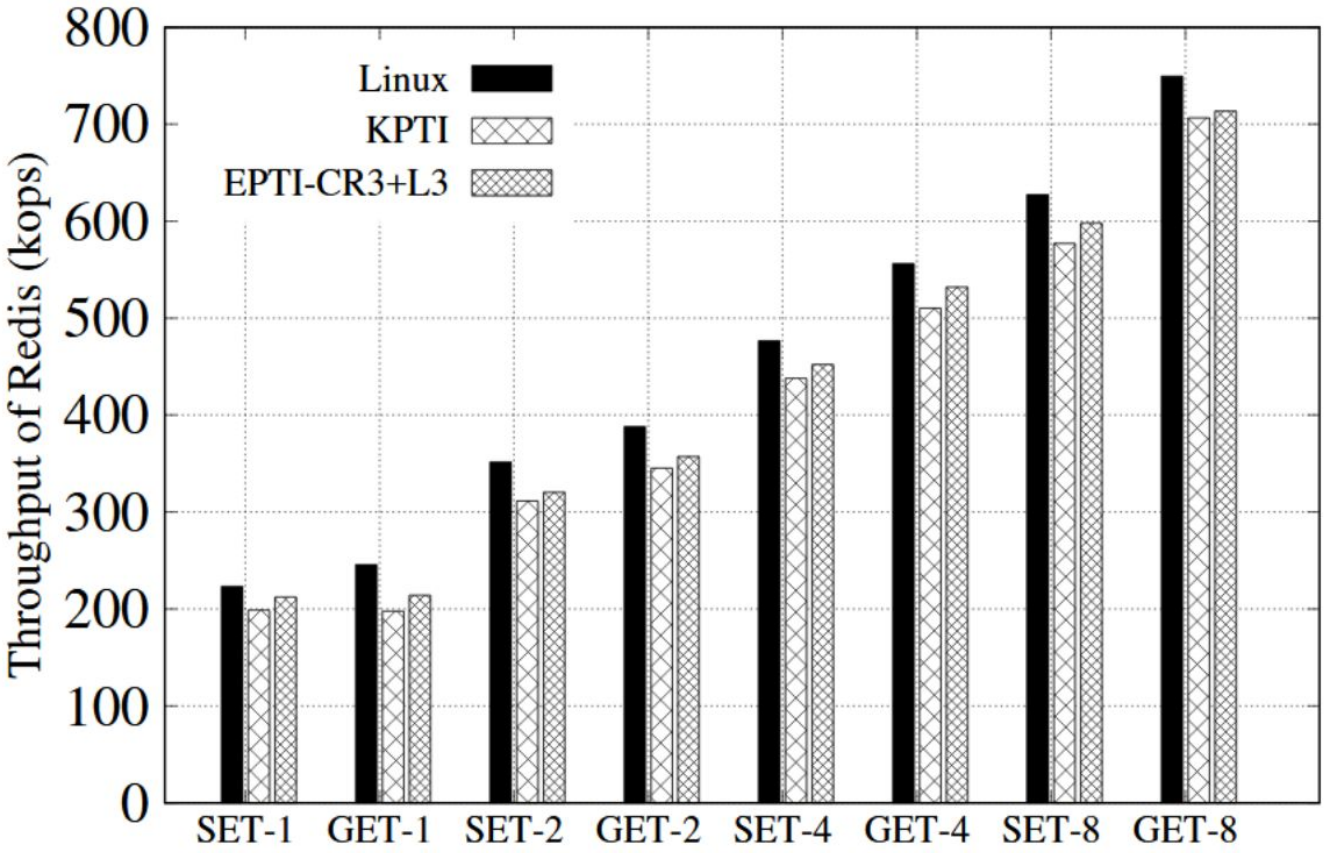
Wniosek: EPTI jest szybsze od KPTI

System plików (fs_mark)



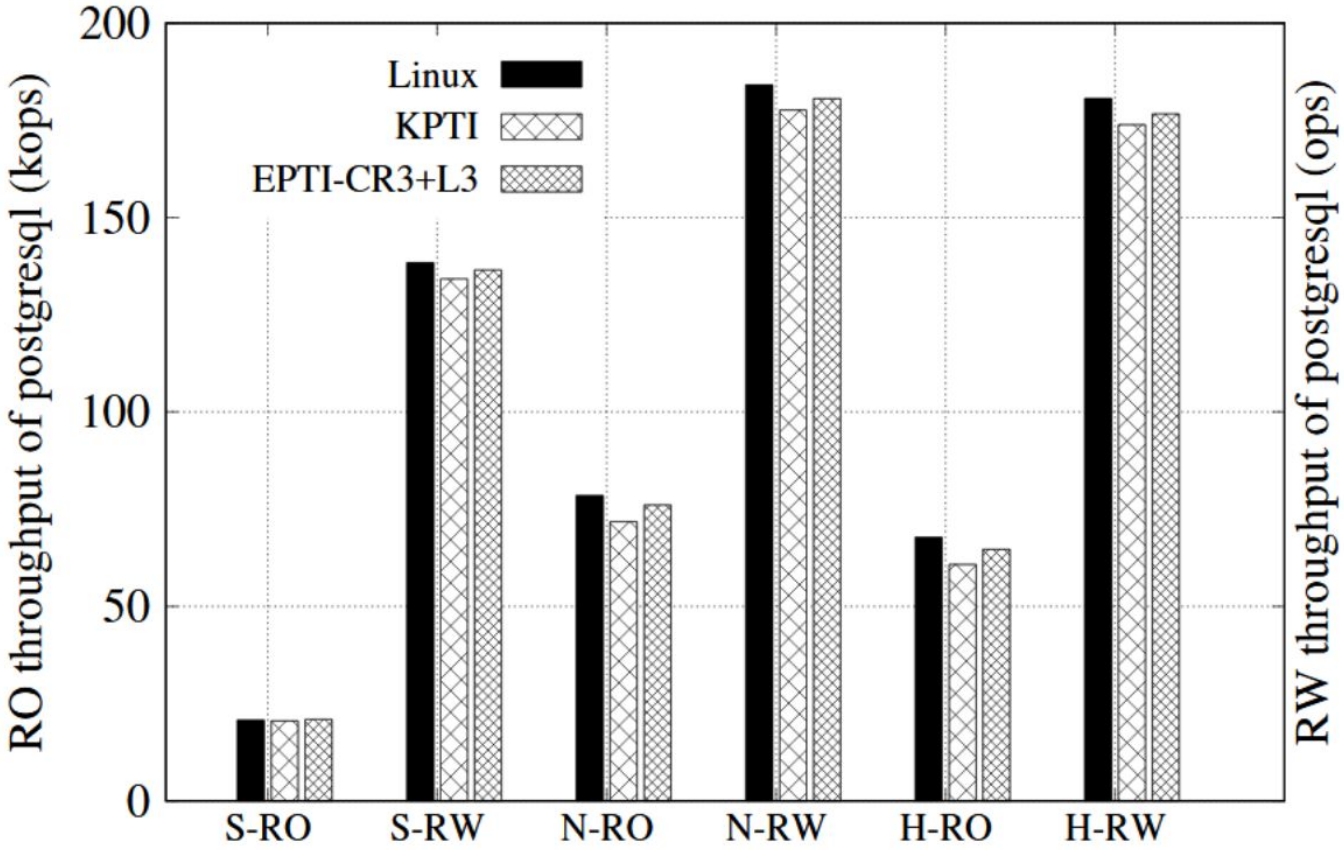
Wnioski: EPTI (narzut 1,1%) jest szybsze od KPTI (narzut 6,5%). Różnica jest niezauważalna, bo czynnikiem dominującym jest czas IO dysku.

Redis (redis-benchmark)



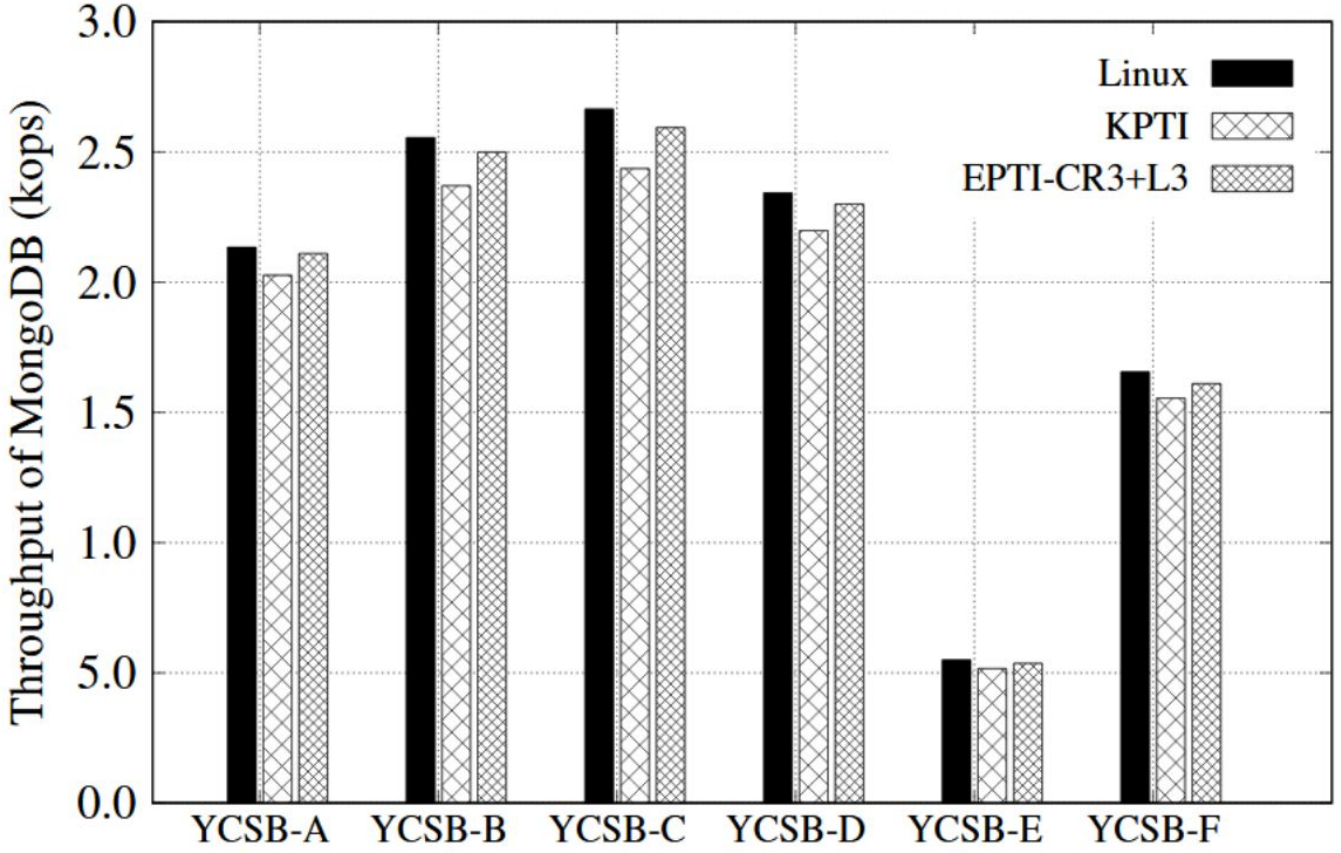
Wnioski: EPTI (średni narzut 6%, najgorszy 12%) jest szybsze od KPTI (średni narzut 12%, najgorszy ponad 20%).

PostgreSQL (pgbench)



Wnioski: EPTI (narzut 4% w H-RO) jest szybsze od KPTI (narzut 12% w H-RO).

MongoDB (YCSB)



Wnioski: EPTI (średni narzut 2%) jest szybsze od KPTI (średni narzut 7%).

Migracja VM

Table 4: VM live migration to host with EPTI, in *ms*.

	KVM w/o EPTI	KVM w/ EPTI
Total time	15779.5 ±1112.03	15782.6 ±1111.86
Downtime	6.1 ±0.82	9.2 ±1.03

Wnioski: EPTI potrzebuje 3 ms więcej, gdyż oprócz samej migracji należy przeprowadzić skanowanie pamięci, podmienić trampoliny i przygotować tablice EPT.

Podsumowanie

Przewagi EPTI nad KPTI:

- Kompatybilność
- Wydajność - średni spadek wydajności w EPTI to 6%, w KPTI 11%, co daje zysk 45%
- Bezproblemowe wdrożenie

Obecne braki w EPTI - brak wsparcia dla:

- architektury x86 systemu gościa
- pięciopoziomowych tablic stron
- Windows

Modyfikacje, jakie EPTI czyni w obrazie maszyny wirtualnej nie są transparentne dla gościa, aczkolwiek nie wpływają na funkcje takie jak VMI czy sprawdzanie integralności kernela.