

Systemy plików

główne cechy, rozwiązania, wydajność

Tomasz Potęga

Wirtualna Polska S.A.

- “FS” (ok. 1970)
 - bardzo prosta konstrukcja
 - przetrwał do czasów Systemu V
 - 2% fizycznej wydajności dysków
- Berkeley Fast File System (FFS, 4.2 BSD, 1984)
 - grupy cylindrów, nowe metody alokacji, większa funkcjonalność
 - osiągał 47% wydajności medium
 - baza dla wielu innych rozwiązań

- szybki dostęp do danych
 - dyski są coraz bardziej pojemne (w sprzedaży urządzenia o pojemności 1 TB)
 - danych jest coraz więcej
 - ... ta tendencja raczej nie ulegnie zmianie
- błyskawiczne operacje
- wysoka niezawodność
- łatwa ewentualna naprawa

- co zostaje zapisane?
 - dane – informacje zapisywane przez użytkowników
 - metadane – “informacje o informacjach”: struktury katalogów, atrybuty, bez nich nie mamy dostępu do właściwych danych

- logging/journaling
 - system plików zapisuje “plan” operacji do wykonania, później uzupełniany
 - w przypadku awarii struktura nie zostaje uszkodzona
 - wiemy, co było do zrobienia
 - alternatywne podejście – soft updates (FFS, 1999)
- nowe algorytmy alokacji bloków
 - ekstenty – alokacja przestrzeni “offset, długość” zamiast klasycznej, poszatkowanej listy bloków
 - zamiana kolejności, grupowanie zapisów

- przechowywanie metadanych
 - coraz bardziej skomplikowane struktury danych
 - do tej pory dominowały proste listy
 - teraz: drzewa (głównie warianty B drzew)
 - szybsze, ale wymagające większej uwagi
- konstrukcja zakładająca dużą pojemność
 - 64–, czasem nawet 128 bitowe adresowanie
- kontrola poprawności danych
- integracja volume managerów

- czy po awarii zasilania system plików nadaje się do montowania?
 - czy może mamy 48h wyjęte z życiorysu?
 - my jak my, ale co na to użytkownicy...
- co znajdziemy na dysku
 - dane – jak wiele zostanie uszkodzone
 - metadane
- odporność na awarie urządzenia
 - błędy odczytu, “nietrafione” zapisy

Dostępne rozwiązania - Linux

- początkowo system plików Minix
 - ograniczenie do 64 MB
 - krótkie nazwy plików
- później – extended file system (extfs)
 - niezadowalająca wydajność
 - niepełna funkcjonalność
- co dalej?
 - zapomniany xiafs
 - ext2fs: rozwinięcie extfs, bazujące na ideach FFS

- oparty na ext2fs, i *często* z nim zgodny – prosta migracja
- główna zmiana – journaling (JBD)
 - jedno słowo, a **zupełnie** inny system plików
- ciągle rozwijany – “dzisiejszy” ext3fs to trochę inny system od tego sprzed kilku lat:
 - dir_index
 - resize_inode
- dostępny od końca 2001 roku (kernel 2.4.15)

- data=ordered (domyślny)
 - logowanie metadanych, ale to *dane* przypisane do transakcji jako pierwsze trafiają na dysk
- data=writeback
 - logowanie metadanych, bez gwarancji wcześniejszego zapisu danych
- data=journal
 - tak dane, jak i metadane “przechodzą” przez dziennik
 - w pierwszych wydaniach jedyny tryb pracy

Wielki następca: ext4fs

- rozszerzenie struktur do 64 bitów
 - 2^{32} bloków po 4 KB to tylko 16 TB
 - wcześniej powstawały łąaty na ext3 umożliwiające wykorzystanie większych przestrzeni dyskowych
- ekstenty
 - konieczne jest podanie dodatkowej opcji przy montowaniu
 - traci się wsteczną zgodność z ext3
- docelowo: ma zastąpić kod ext3 w kernelu

- pierwszy w linuksowym świecie naprawdę popularny system z kroniką
 - włączony do kernela już w początkach serii 2.4
- korzysta z wariantu B+ drzew
- specjalność: *tail packing*, dane mogą pozostać w głównym drzewie systemu plików
- swego czasu uznawany za bardzo szybki
- ...ale też dosyć często krytykowany
 - fragmentacja, problemy po awarii

- wiele nowych pomysłów, które może nie za bardzo pasują do klasycznej wizji systemu plików
- definiowane przez użytkownika transakcje
- “wtyczki”
- opóźniona alokacja bloków (*allocate on flush*)

- współpraca z Hansem Reiserem nie należy do najłatwiejszych
- firma Reisera – Namesys – uznała wersję 3 za zakończoną, angażując zasoby w pracę nad systemem Reiser4
- kontrowersje wokół sytuacji prawnej Reisera
- powolny odwrót od reiserfs (Novell)

<http://www.namesys.com/>

- pierwszy znaczący system plików z dziennikiem
- opracowany przez SGI na początku lat 90.
(premiera: IRIX 5.3, 1994)
- źródła udostępnione na licencji GPL w 2000 roku
- szybko trafił do jądra Linuksa (linie 2.4 oraz 2.6)

<http://oss.sgi.com/projects/xfs/>

- stworzony przez IBM na potrzeby systemu AIX, dostępny też w OS/2 Warp Server
- kod udostępniony pod koniec 1999 roku
- nie zyskał tak dużego poparcia dostawców oprogramowania, jak XFS

<http://jfs.sourceforge.net/>

- powstał jako implementacja FFS dla systemu SunOS
- dalszy rozwój:
 - *I/O clustering* (SunOS 4.1.1)
 - dziennik (Solaris 2.4 + SDS 3.0, Solaris 7, domyślnie w S10)
 - *Direct I/O* (Solaris 2.6)
 - *snapshots* (Solaris 9)

- filesystem i volume manager w jednym
 - uproszczone zarządzanie zasobami
 - podział na partycje, slices, ...
 - RAID-Z/RAID-Z2
- wszystko jest transakcją
 - całość się uda – lub nie, nie ma “połowicznie udanych” operacji
 - zawsze spójna struktura

- model Copy on Write (COW)
 - dane nie są nigdy nadpisywane
 - błyskawiczne snapshoty i kopie
- sumy kontrolne danych
 - ZFS stara się sam naprawiać uszkodzone obszary
- sumy kontrolne metadanych
- zapas na kilka lat
 - adresowanie 128 bitowe

- ZFS on FUSE – Filesystem in Userspace (beta)
http://www.wizy.org/wiki/ZFS_on_FUSE
- port do FreeBSD (Paweł Jakub Dawidek)
- zainteresowanie ze strony Apple
 - ZFS w buildach Mac OS X 10.5 (Leopard)

- co pokazują mikrobenchmarki?
- “ja mam dłuższy... słupek”
- ale uwaga – nawet jeśli powtarzalne operacje dają wysokie wyniki, nie musi to wcale oznaczać wysokiej wydajności “prawdziwej” aplikacji
- odpowiednio dobrany zestaw operacji może dawać zakłamaną obraz (ale piękny marketingowo)

- bonnie/bonnie++
 - sekwencyjne odczyty/zapisy, etc.
- iozone
 - nie tylko read/write, ale też mmap, async I/O
 - wiele wątków/strumieni
 - działa na wielu platformach

<http://www.coker.com.au/bonnie++/>

<http://www.iozone.org/>

- PostMark
 - symulacja pracy serwera poczty
 - w pewnym stopniu konfigurowalny
 - obecnie raczej nie jest stosowany
 - Network Appliance “nie przyznaje się”

- inne podejście do modelowania obciążenia
 - odwzorowanie pracy aplikacji
- modele zapisane w postaci skryptów *f*
 - zestaw testów nie jest ograniczony
 - gotowe narzędzia do generowania struktur katalogów
 - wiele procesów, wątków, także synchronizowanych
 - dodatkowe mechanizmy dla symulacji baz danych

<http://www.solarisinternals.com/wiki/index.php/FileBench>

- spory zestaw dołączonych testów:
 - varmail (a la PostMark)
 - webserver, webproxy
 - oltp
 - mongo
 - ale też microbenchmarki
- główna platforma: Solaris (x86/SPARC)
 - jest też port linuxowy

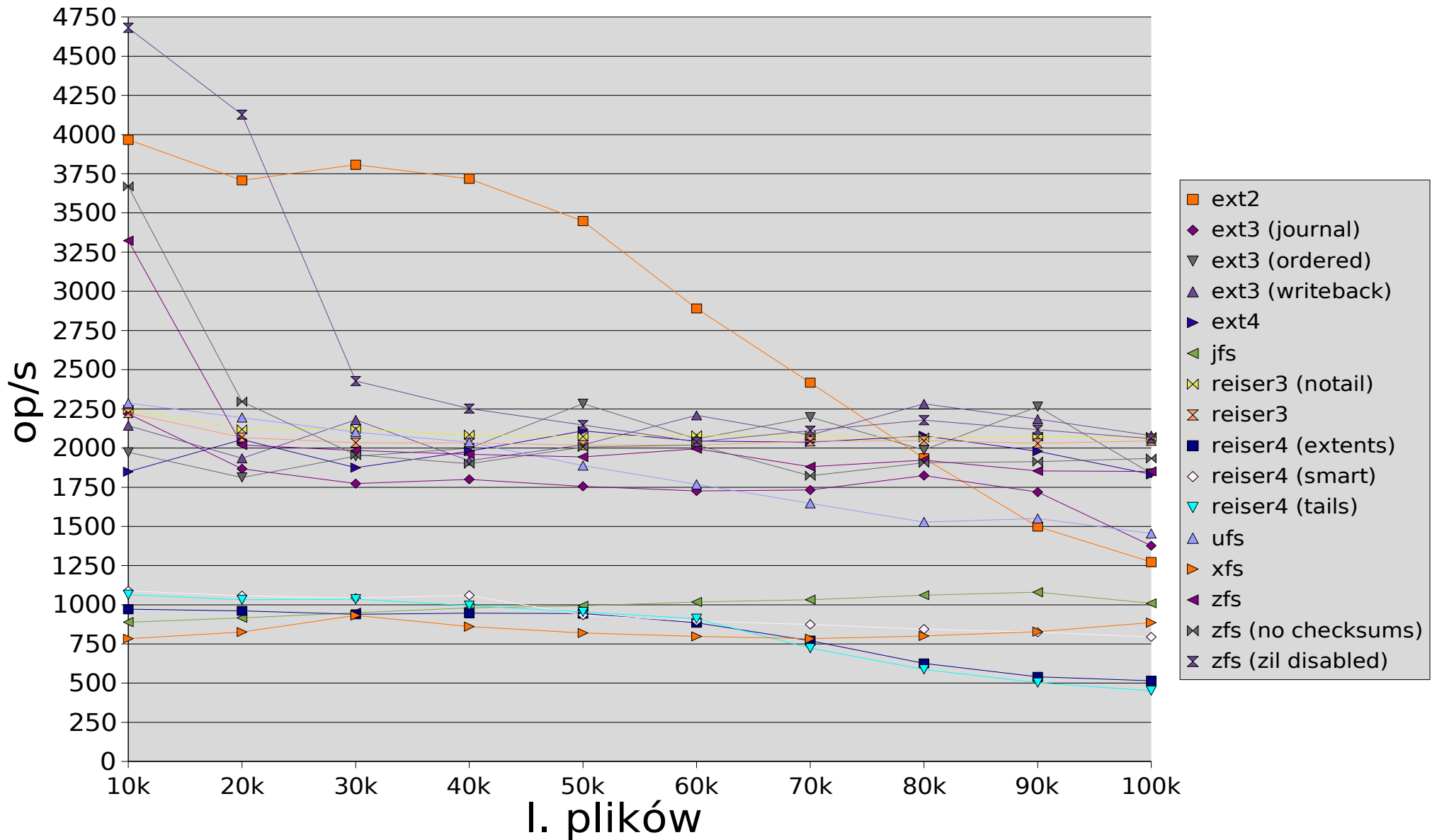
filebench - przykład

```
set $nfiles=1000
set $meandirwidth=1000000
set $filesize=16k
set $nthreads=16
set $meaniosize=16k

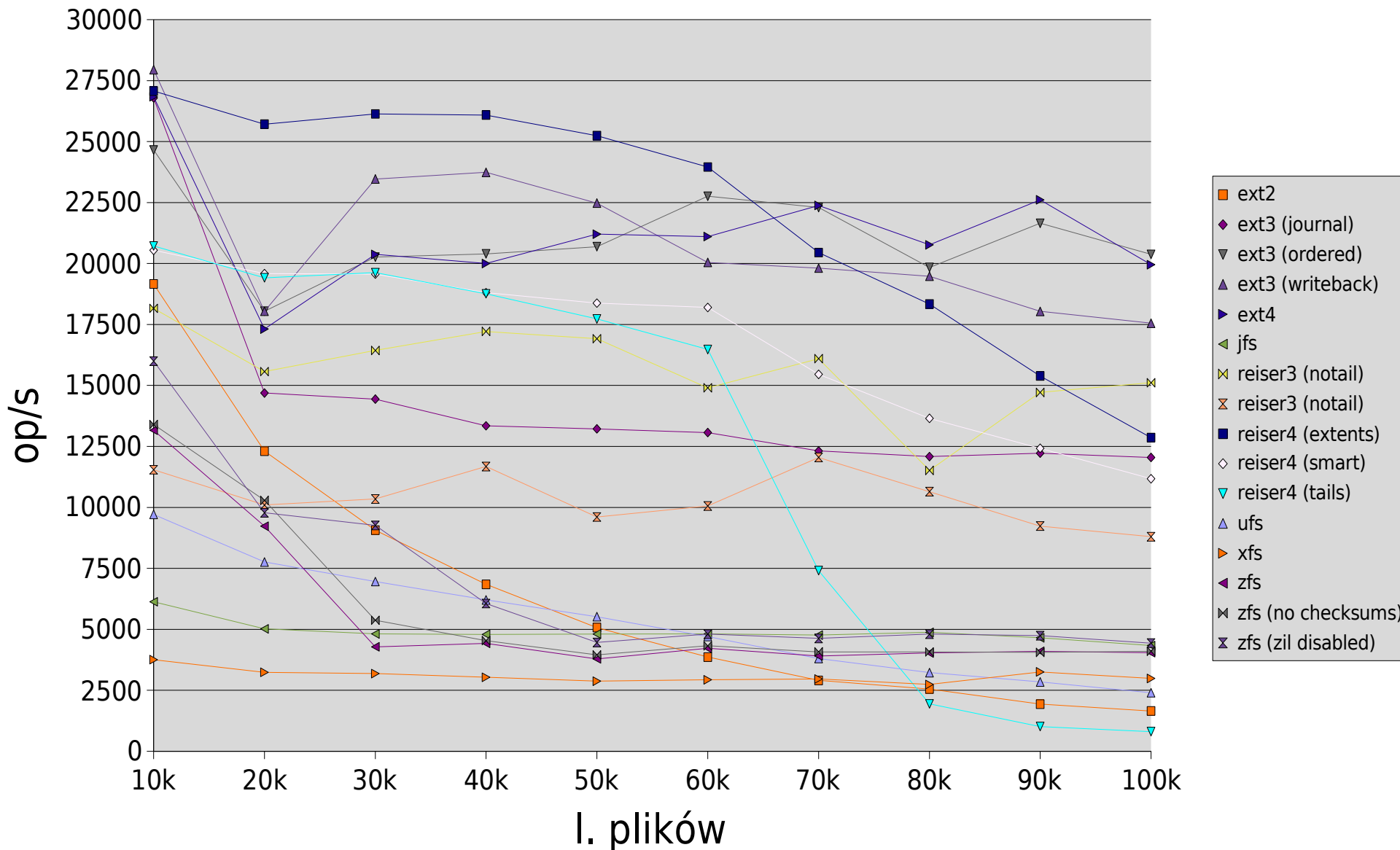
define fileset name=bigfileset,path=$dir,size=$filesize,
    entries=$nfiles,dirwidth=$meandirwidth,prealloc=80

define process name=filereader,instances=1
{
    thread name=filereaderthread,memsize=10m,instances=$nthreads
    {
        flowop deletefile name=deletefile1,filesetname=bigfileset
        flowop createfile name=createfile2,filesetname=bigfileset,fd=1
        flowop appendfilerand name=appendfilerand2,iosize=$meaniosize,fd=1
        flowop fsync name=fsyncfile2,fd=1
        flowop closefile name=closefile2,fd=1
        flowop openfile name=openfile3,filesetname=bigfileset,fd=1
        flowop readwholefile name=readfile3,fd=1
        flowop appendfilerand name=appendfilerand3,iosize=$meaniosize,fd=1
        flowop fsync name=fsyncfile3,fd=1
        flowop closefile name=closefile3,fd=1
        flowop openfile name=openfile4,filesetname=bigfileset,fd=1
        flowop readwholefile name=readfile4,fd=1
        flowop closefile name=closefile4,fd=1
    }
}
```

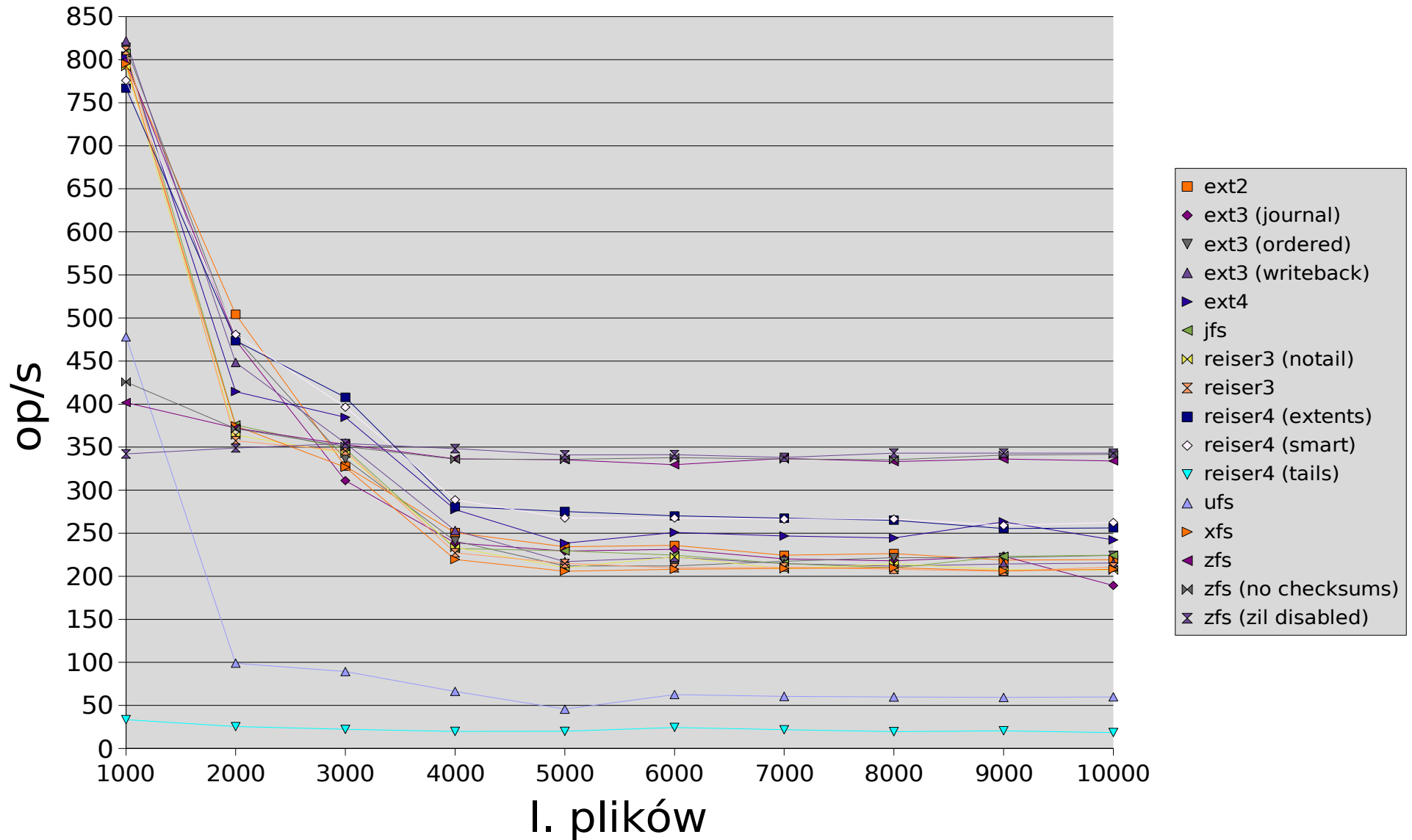
filebench: varmail



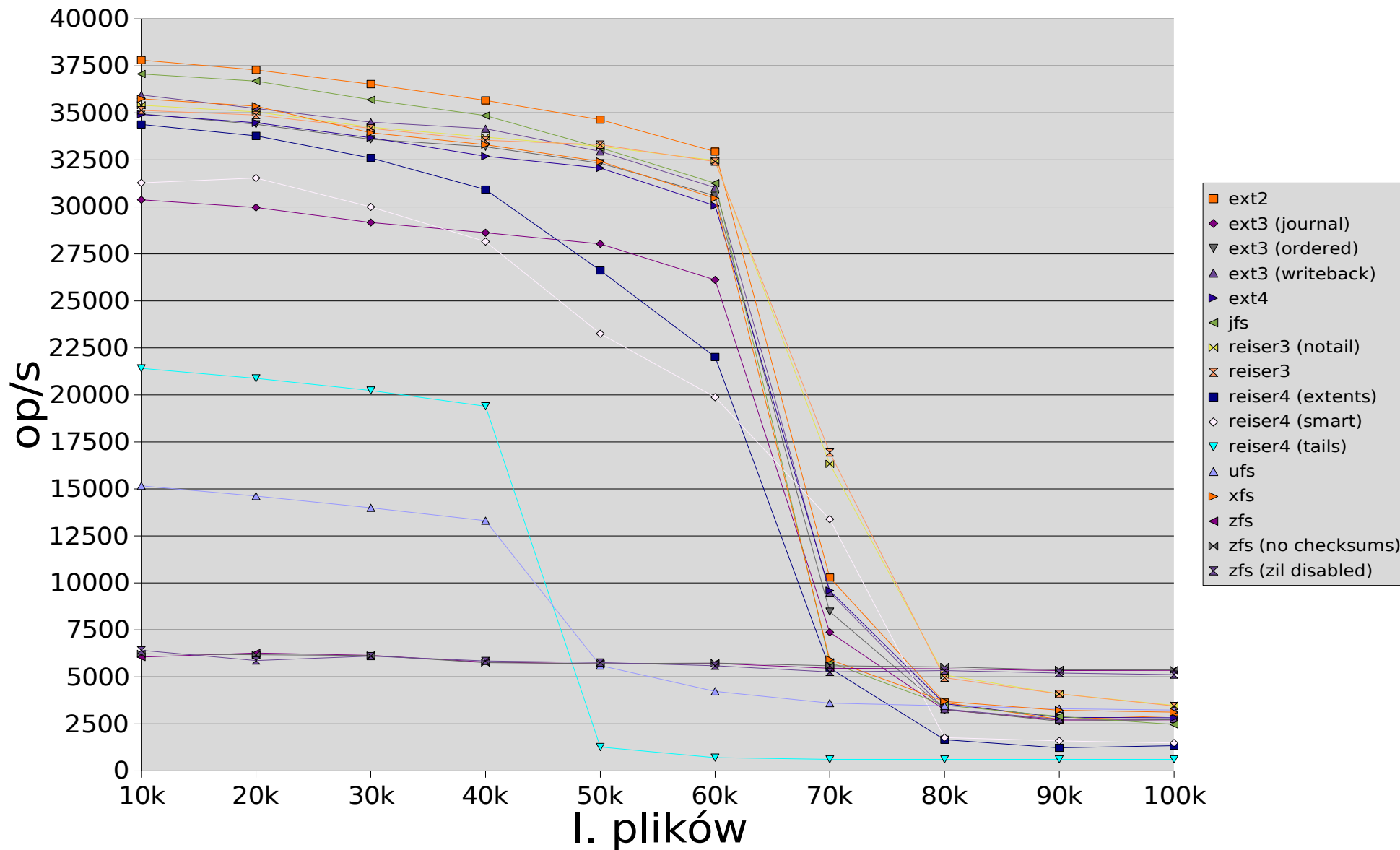
filebench: webproxy



filebench: fileserver



filebench: webserver



- sprzęt:
 - 4 x Pentium III Xeon 700 MHz, 1 MB L2
 - 2 GB RAM
 - 3 x Seagate Cheetah 36XL
- software:
 - Debian Etch (kernel 2.6.21.1, CFQ) / LVM2
 - Solaris 10u3 / SVM / ZFS
- ustawienia filebench:
 - webserver: *dirwidth 200, nthreads 50*
 - webproxy: *nthreads 50*
 - fileserver: *nthreads 24, filesize 1m*

Podziękowania dla Pawła Sasina za pomoc w realizacji testów.

- jakie są wymagania aplikacji?
 - czy będziemy operować na dużych zbiorach małych plików?
 - może plik będzie jeden, za to ogromny?
 - jaki będzie rozkład operacji?
- czy możemy sobie pozwolić na utratę danych?
 - czy przerwana aplikacja uda się ponownie uruchomić?
 - serwer odpowiedział “250 OK”, a maila nie widać...

Co wybrać?

- bierzmy pod uwagę jakość dostarczanych narzędzi
 - właściwie każdy zaliczył wpadki
- dokładnie sprawdzajmy możliwości systemu
 - ograniczona ilość inode'ów?
 - quota?
 - extended attributes/ACLs?

- każdy system ma swoje mocne punkty, przykładowo:
 - ZFS – efektywne losowe zapisy
 - ext3 – ogromna baza narzędzi
 - XFS – wysoka wydajność przy pracy z dużymi plikami
- aplikacja czasem “wie lepiej”
 - DirectIO, “surowe” urządzenia

Pytania, komentarze?