FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Bezeztrátová komprese pro vysokorychlostní časové řady dat pro monitorování a analýzu elektrické energie

Studijní program:	B0613A140005 Informační technologie
Studijní obor:	Aplikovaná informatika
Autor práce:	**Miroslav Čambor**
Vedoucí práce:	Ing. Jan Kraus, Ph.D. Ústav mechatroniky a technické informatiky

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Bezeztrátová komprese pro vysokorychlostní časové řady dat pro monitorování a analýzu elektrické energie

Jméno a příjmení:	**Miroslav Čambor**
Osobní číslo:	M20000204
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav mechatroniky a technické informatiky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Seznamte se s formáty pro archivaci posloupností hodnot vzorků elektrických veličin a s možnostmi optimalizace jejich velikosti při ukládání, přenosu a následném zpracování.
2. Vyberte vhodné bezestrátové kompresní algoritmy a na větším archivu reálných dat podrobně experimentálně ověřte jejich charakteristické vlastnosti.
3. Zaměřte se na rychlost a paměťovou náročnost jednotlivých řešení a jejich jednotlivých konfigurací, zkoumejte vliv pořadí a zvoleného formátu vzorků veličin ve vstupních datech na sledované parametry, pro každý algoritmus stanovte optimální velikost vstupních dat apod.
4. Své poznatky stručně a přehledně shrňte v průvodní zprávě bakalářské práce a v závěru diskutujte možnosti dalšího rozvoje tématu či praktického využití vašich poznatků.

Rozsah grafických prací: dle potřeby dokumentace
Rozsah pracovní zprávy: 30-40 stran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] JUMAR, Richard; MAAß, Heiko; HAGENMEYER, Veit. Comparison of lossless compression schemes for high rate electrical grid time series for smart grid monitoring and analysis. *Computers & Electrical Engineering*, 2018, 71: 465-476.
[2] LOUIS, Sancho Simmy, et al. Wide Area Synchronous Disturbance Recording System. In: *2021 9th IEEE International Conference on Power Systems (ICPS)*. IEEE, 2021. p. 1-6.
[3] IEC INTERNATIONAL ELECTROTECHNICAL COMMISSION, et al. Common format for transient data exchange (COMTRADE) for power systems. 2001.
[4] BLAIR, Steven; COSTELLO, Jason. Slipstream: High-Performance Lossless Compression for Streaming Synchronized Waveform Monitoring Data. In: *2022 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*. IEEE, 2022. p. 1-6.
[5] KING, J. A.; GUNTHER, E. W. Comtrade/pqdif conversion. In: *Transmission and Distribution Conference and Exhibition*. 2005. p. 359-364.

Vedoucí práce: Ing. Jan Kraus, Ph.D.
Ústav mechatroniky a technické informatiky

Datum zadání práce: 12. října 2022
Předpokládaný termín odevzdání: 15. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2022

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

22. května 2023 Miroslav Čambor

Využití komprese pro efektivní uchování archivů měření charakteristik elektrické energie

Abstrakt

Každý, kdo má nějaká data a chce je nějakým způsobem uložit, si jednou za čas položí jednu zásadní otázku. Jak by se daly uložit všechny důležitá data, aby o ně nepřišel, a hlavně aby zabírali málo místa. To všechno řeší kompresní algoritmy a cílem této práce je zkontrolovat jednotlivé implementace od různých firem, jestli jsou všechny zavedené předpoklady pravdivé.

Klíčová slova: komprese, dekomprese, EXE

Usage of compress for saving of electric energy characteristic into archive

Abstract

Everybody, who has some data and wants to have it save. He has one specific question. How can he save his critical information to have it in save but it cannot have a lot of space. Everything can solve compress algorithms. The main target of this labor is if known information are correct.

Keywords: compresion, decompresion, EXE

Poděkování

Děkuji především rodině a vedoucímu práce za pevné nervy během vypracovávání této bakalářské práce.

Obsah

Seznam tabulek

Seznam obrázků

Seznam zkratek

ALS	Audio LossLess
BWT	Burrows-Wheeler Transform
C#	C Sharp (programovací jazyk vyvinutý společností Microsoft)
DWT	Discrete Wavelet Transform
EXE	executable
FIR	Finite Impulse Response
FLAC	Free Lossless Audio Codec
IIR	Infinite Impulse Response
LMS	Least Mean Square error
LPC	Linear Predictive Coding
LZW	Lempel Ziv Welch
MB	Megabyte
RAM	Random Access Memory
RLE	Run Length Encoding
TTA	The True Audio
URL	Uniform Resource Locator

Úvod

Protože ukládaná data, na které se zaměřuje tato práce, jsou sinusového průběhu a tím se dají uložit jako soubor typu WAVE, tak je používán právě tento formát pro vtupní soubory. Právě díky této úvaze se může použít více typů algoritmů a nemusí být tyto algoritmy univerzální, ale mohou být i úzce zaměřené. Proto bylo možné použít audio kodeky, obsahující kompresní algoritmy, které jsou zaměřené na audio nahrávky a tím mají sinusový průběh.

Algoritmy, které jsou součástí práce, jsou proto klasické kompresní algoritmy, ale obsahem jsou i audio kodeky. Klasických kompresních algoritmů bylo vybráno celkem čtyři. Jsou to především 7ZIP [1], BZIP2 [2], GZIP [3] a LZW [4]. Audio kodeky byly použity dva FLAC [5] a TTA [6]. Klasické kompresní algoritmy jsou použitelné na komprese různých typů formátů. Některé dokáží komprimovat celý adresář najednou. Univerzálnost dokáže pomoct, ale pokud člověk má specifické potřeby a specifická data, může použít specifická řešení jako jsou už dříve zmíněné audio kodeky.

7ZIP je univerzální algoritmus, který umožňuje i komprese více souborů najednou. Tato možnost je umožněna, protože je stavěn pro archivaci celých archívů a složek. Má také velké množství možných parametrů, které se dají nastavit a tím ovlivnit výsledek algoritmu. Výsledek ovlivňují především parametry, které byly použity v této práci. Jako první se nastavuje parametr level, který umožňuje definovat míru komprese bez práce s dalšími parametry. Další z parametrů slouží k nastavení kompresních metod a filtrů, při správném výběru může tento parametr velmi ušetřit místo na disku, ale také nemusí ušetřit žádné místo. Jako poslední se dá nastavit počet vláken, které ovlivňují délku běhu algoritmu. Mezi další parametry, které neovlivňují míru komprese jsou například nastavení výstupní složky, nastavení hesla, rekurzivní procházení a další.

Algoritmy BZIP2, GZIP a FLAC jsou skromnějšími konkurenty už zmíněného algoritmu 7ZIP, který je mnohonásobně komplexnější. Tyto algoritmy podporují na vstupu jen jeden rozšiřující parametr level, který musí při kompresi zastoupit všechny parametry ovlivňující kompresi u předešlého algoritmu 7ZIP.

Zato algoritmy LZW a TTA mají nejméně možností. Neumožňují nastavit ani jeden rozšiřující parametr, proto nemají v mé práci takové zastoupení, ale i přes tuto vlastnost mají mezi zkoumanými algoritmy své místo. Díky své strohosti mají velkou výhodu pro uživatele. Jsou pro uživatele jednodušší a tím pro některé přívětivější.

1 Rešerše

Každý, kdo používá počítač, mobil, tablet nebo nějakou jinou technologii této doby, na kterou se dají uložit nějaká data pocítil ten pocit bezmoci, když dojde interní paměť. V tomto okamžiku člověku nezbývá než řešit situaci. Jsou zde čtyři možnosti koupit nové zařízení, vyměnit interní uložiště (pokud to jde), rozšířit paměť externím uložištěm (hmatatelným jako je SD karta nebo zaplacením cloudového uložiště) nebo zkusit tato data nějak stlačit na menší velikost. Na rozdíl od nastíněného problému, kdy člověk většinou radši šáhne po prvních tří možnostech, tak v průmyslu, na internetu nebo v jiných veřejných odvětvích tohoto typu se více myslí na úspory, kde se dá. Proto firmy radši začnou dřív nebo později s kompresí dat. Tato možnost je více výpočetně náročná, ale ve firmách se každá koruna počítá.

Každá komprese se nehodí na všechno. Jsou zde nějaké možnosti, které se dají použít prakticky na jakýkoliv zdroj (např. 7-ZIP), ale jsou zde i takové technologie, které jdou použít jen na nějaký typ souboru. Pro soubory typu obrázek nebo fotografie se dají použít formáty typu SVG, PNG, JPEG a další. První z nich je využívá jazyk typu podobný typu HTML, proto v základu nemá obsažen žádný kompresní algoritmus, ale PNG a JPEG využívají kompresní algoritmy, ale každá technologie používá jiné. JPEG používá ztrátovou kompresi, proto se moc nehodí pro prezentování nějakých výsledků nebo pro profesionály. Spíše se hodí pro domácí použití. PNG je více univerzální formát. Používá bezeztrátovou kompresi a tím zobrazí zdroj ve stejné kvalitě jako při pořízení, proto se více hodí pro profesionální tvorbu nebo práce jako je tato.

Při poslechu hudby se pro změnu používají tzv. audio kodeky, které jsou dobré řešit hlavně když uživatel chce si poslechnout hudbu přes bezdrátová sluchátka. Základní a nejméně kvalitním audio kodekem je SBC (viz. [7]), který má bitový tok až 328 kbps se vzorkovací frekvencí 44.1 kHz. Nenáročnému posluchači tato informace možná je jedno a spokojí se i s tímto kodekem, ale pokud chcete vyšší kvalitu musíte použít něco lepšího. Mezi lepší kodeky patří aptX (viz. [8], [9]), AAC (viz. [8], [10]) a další, které využívají kvalitnější bezeztrátové kodeky. Kodek aptX je hlavně používán u uživatelů operačního systému Android a má vyvinutých více verzí. Každá verze se dá použít v jiné situaci. První verze se nazývá aptX a je to základní verze kodeku, kterou vývojáři vydali jako první a je nejvíce rozšířena. V tomto případě je bitový tok už 352 kbps a vzorkovací frekvence 48 kHz. Mezi další verze patří aptX Low Latency, který je dobrý při hraní her, kde hráč potřebuje nízkou odezvu a nepotřebuje tu nejlepší kvalitu. Tato verze má bitový tok 352 kbps a vzorkovací frekvenci 44.1 kHz Další verzí je aptX HD, která se nejvíce zaměřuje na kvalitu hudby, protože má bitový tok 576 kbps se vzorkovací frekvencí 48 kHz.

Poslední nejnovější verze je aptX Adaptive, která je zpětně kompatibilní se základní verzí aptX a aptX HD. Tato verze má odezvu přibližně 80 ms. Bitový tok se pohybuje v rozmezích 279 a 420 kbps. Kodek AAC je nejvíce rozšířen u uživatelů Apple zařízení a zaostává s kvalitou za svým konkurentem aptX s bitovým tokem jen 264 kbps a vzorkovací frekvencí 44.1 kHz. Mezi nejkvalitnější patří FLAC (viz. [11]), LDAC (viz. [12], [8]) a ALAC (viz. [10]). Jediný LDAC je používaný společností Sony pro bezdrátový poslech hudby při bitovém toku 990 kbps a vzorkovací frekvenci 96 kHz. FLAC a ALAC jsou používaný hlavně pro drátový poslech nebo pro bezeztrátové uložení dat. FLAC má vzorkovací tok 16 bit a vzorkovací frekvenci 44,1 kHz (CD kvalita). ALAC je používán hlavně výrobky společnosti Apple a má vzorkovací tok od 16 do 24 bitů se vzorkovací frekvencí od 44,1 do 192 kHz.

1.1 Kompresní algoritmy a audio kodeky

Tyto algoritmy jsou základní algoritmy pro kompresi dat nebo k uchování dat, které mají sinusový průběh, který je nejvíce zastoupen u souborů obsahující hudební, hlasová nebo další hudební data, které lze určitými specializovanými programy přehrát.

Budou zde popsány, jak algoritmy použité v mé práci, tak i algoritmy, které zde zmiňuji pro vyšší úplnost mé práce. Algoritmy zmíněné navíc pro úplnost práce, ale mohou být použity v rámci jiného algoritmu jako otevřená („open source“) komponenta, za kterou se nemusí platit, a proto ji autor použil ve svém algoritmu.

Otevřené algoritmy nemají za výsledek jenom bezplatnost algoritmu, ale spojení více kompresních algoritmů do jednoho může vést k vyšší efektivitě ukládaní dat a tím i ušetření místa na disku.

1.1.1 Kompresní algoritmy

Formát 7-ZIP je otevřená a bezplatná technologie, vyvinutá Igorem Pavlovem, která je dostupná na všech hlavních platformách (Windows, Linux a Mac OS) – viz. [13]. Instalační soubory pro tyto platformy po nainstalování zpřístupní grafické rozhraní, které jsou graficky přívětivé, proto je mohou používat i uživatelé, kteří neumí programovat a chtějí jen ušetřit místo. Vývojáři vytvořili také verzi pro příkazovou řádku a knihovnu pro programovací jazyky C, C++, C# a Java. První hlavní vlastnost (viz. [14]) této technologie dříve zmíněná otevřenost, která umožnuje bezplatné stahování a možnost nahlédnout do kódu této technologie. Kód lze stáhnout ze stránek výrobce. Další vlastnost je velký kompresní poměr. Tato funkce umožňuje velké ušetření místa na zařízení. Výsledný soubor, který dostaneme jako výstup tohoto algoritmu je zašifrován pomocí AES-256. Podporuje soubory až o velikosti $16 \cdot 10^9$ GB a názvy ve formátu Unicode, proto nemá problémy se znaky, které se nepoužívají v anglofonních zemích (neboli i české znaky). Mezi podporované metody (viz. [15]) pro kompresi souborů patří LZMA, LZMA2, PPMD, BZIP2 a standartně používaný Deflate. V balíčku metod se také vyskytuje jedna metoda, která nepoužívá žádný kompresní algoritmus, nýbrž jen překopíruje data do formátu 7-ZIP pojmenovaná jako Copy. Tyto metody jsou také rozšířeny jednotlivými filtry, které se dají použít v kombinaci s jednotlivými kompresními metodami nebo i samostatně. Mezi tyto filtry patří Delta, BCJ, BCJ2, ARM, ARMT, IA64, PPC a SPARC.

Algoritmus LZMA je hlavním algoritmem (viz. [16]), který používá algoritmus 7ZIP a je také vyvinut Igorem Pavlovem. V základu je to jen rozšíření algoritmu LZ77. Blízký je Deflate, ale místo Huffmanovo kódování používá rozsahové kódování. Huffmanovo kódování je přirozenou číselnou variantou aritmetického kódování, protože LZMA používá jen číselné operace využívající jen přirozená čísla.

Algoritmus Deflate (viz. [17] a [16]) používá pro každý blok kombinaci algoritmu LZ77 a Huffmanova kódování. Ke komprimaci pro ušetření místa na disku používá zpětné reference až 32 tisíc bytů zpět na předchozí stejný textový řetězec. Tento řetězec je limitován délkou nepřekračující 258 bytů. Informace, kde byl nalezen předchozí výskyt shodný s hledaným řetězcem se uloží do Hashovací tabulky. Hash v hashovací tabulce se vždy vypočítá pro počet bytů, který je roven délce posledního vkládaného řetězce. Jako hlavní kompresní algoritmus byl zvolen dříve zmíněný LZ77 a Huffmanovo kódování je použito jen pro další zmenšení velikosti výstupního souboru.

Huffmanovo kódování (viz. [17] a [18]) je konstruováno s pomocí binárního stromu. Každé písmeno nebo číslice vstupní zprávy je uloženo na určitý list, který má svou váhu vyjádřenou v procentech vyjadřující frekvenci výskytu jednotlivého znaku. Nejblíže k základnímu uzlu jsou vždy skupiny znaků, které mají součet frekvencí nejvyšší a nejdále jsou ty, které mají nejmenší součet. K hranám jsou přiřazeny hodnoty 0 a 1, kde na levé straně od předchozího výše hodnoceného uzlu je přiřazena hodnota 0 a na pravé 1 nebo naopak. Cesta od hlavního nejvýše položeného uzlu k jednotlivým listům určuje, jaké kódové slovo z nul a jedniček je použito k zakódování jednotlivého znaku.

LZ77 (viz. [17]) je kompresní algoritmus, který je vyvinut dvěma vývojáři Jacobem Zivem a Abrahamem Lempelem v roce 1977. Tento algoritmus je také známý pod zkratkou LZW. Funkce algoritmu je jednoduchá. Při každém průchodu se vždy zjistí, jestli při předchozím průchodu náhodou nenašla aspoň nějaká část řetězce, pokud ano, tak se uloží, o kolik pozic zpátky se našel stejný řetězec, jak byl dlouhý, a nakonec další písmeno za naleznutým řetězcem. Když se nenajde žádný předchozí výskyt, tak první dvě hodnoty jsou nula a uloží se jen naleznutý znak.

PPMd je implementací Dimitrie Shakarina PPMdH varianty pro algoritmus 7ZIP (jak si můžete přečíst ve článku [16]). Predikce je prováděna pomocí částečné shody (přeloženo z angličtiny: Partial Matching – PPM), které používá aritmetické kódování. Také používá Markovův model různého a adaptivního kontextového modelování. Požadavky na paměť jsou stejné pro kompresi tak i pro dekompresi.

BZIP2 je metoda používající ke kompresi Burrow-Wheelerovo metodu (viz. [16]). Hlavní myšlenkou tohoto algoritmu je přeskládat symboly v originální zprávě, tak aby se vyvinula účinnější metoda pro další komprese. Každý blok dat vstupu je cyklicky posunut okolo jedno jediného symbolu a tyto rotace se dočasně setřídí do tabulky. Setřídění je lexikografické a s touto tabulkou se uloží index kde je uložena původní zpráva – poslední řádek této tabulky a index originální zprávy.

RLE je technologie vhodná ke kompresi obrazových informací (viz. [18]). Nejvíce se hodí pro opakující data, protože když se najde nějaká opakující se informace, tak ji algoritmus nahradí referencí na předchozí místo uložení. Tím se dá ušetřil velké množství místa.

Diskrétní vlnková transformace (DWT) je bezeztrátová metoda, která používá ke kompresi obrazových informací vlastnosti elektromagnetických vln. Základním principem komprese je využití závislosti a návaznosti jednotlivých vln. Tento princip je velmi žádoucí u komunikačních nástrojů a při velké rozdílnosti dat.

Aritmetické kódování (viz. [19] je velmi známá, univerzální, bezeztrátová metoda, která vykazuje výsledky velmi blízko optimálním výsledkům. Hlavním důvodem je, že se u každé komprimované zprávy spočítají frekvence výskytu u jednotlivých symbolů. Z těchto frekvencí se spočítají intervaly, které se dosazují do vzorce. Na konci dostaneme interval, který vymezuje čísla, ze kterých se dá vybrat výsledné číslo a uloží se délka kódovaného textového řetězce (pro dekódování textového řetězce). Výsledné získané číslo vybereme ze zmiňovaného intervalu. Pro ukázku stačí vybrat jakékoliv ze zmíněného intervalu, ale pro dosáhnutí malého počtu bitů při uložení výsledku se vybere to číslo s nejkratším počtem desetinných míst.

U Burrows-Wheelerovo transformace (viz. [20]) se prvně provede rotace zprávy, kterou chceme komprimovat. Rotace je v tomto případě provedena tolikrát, jak je dlouhá zpráva. To znamená, pokud napočítáme ve zprávě 10 symbolů (s opakováním), tak se provede 10 rotací (poslední rotace obsahuje originální zprávu). Následně uložíme všechny rotace a lexikograficky je setřídíme. Výsledkem jsou písmena z posledního sloupce.

1.1.2 Audio kodeky

FLAC je bezeztrátový audio kodek, jak můžete naleznout na [21]. Nevyužívá žádný dostupný patent, proto je považován také jako otevřený software. Je dostupný na všech větších operačních systémech jako jsou Windows, Linux, Mac OS, BSD, Solaris a další. Formát jako takový podporuje štítky, vložení formátu do jiného formátu a rychlé přesouvání kurzoru v rámci audio nahrávky. U vstupního signálu se dá nahrávka rozdělit na menší bloky (viz. [16]), které se dají lépe zpracovat i třeba na slabších strojích. Vícekanálové vstupy (8 a více) obsahují jeden sub-blok, který je ukládán bez prokládání. Ke kompresi se používá FIR (Finite Impulse Response) prediktor, který je také známý jako LPC (Linear Predictive Coding). Prediktor umožňuje nastavit až 32 proměnných využívaných pro chod algoritmu, které si mohou uživatelé nastavit. Koeficienty této metody jsou vypočítávány Levinson-Durbinovým algoritmem. K zakódování se používá takzvané „Rice coding“ (volně přeloženo: „Rýžové kódování“), které spolupracuje s Laplacovou distribucí. Poslední zmíněná metoda je používaná k optimálnímu odhadu parametrů pro „Rice coding“.

MPEG-4 ve verzi ALS (Audio LossLess) k výsledku používá kombinaci rozdělení na bloky dat (viz. [16]), predikci (krátkodobou tak i dlouhodobou), kódování spojnic a náhodné kódování zbytkových dat. Každý krok tohoto algoritmu je navrhnut, tak aby byl co nejvíce reakce schopný na jednotlivé změny. Lineární FIR predikce jsou spočítány pomocí Levinson-Durbinovo algoritmu jako u algoritmu FLAC. Rekurzivní vlastnosti algoritmu umožňují, že v rámci jednotlivých bloků se minimalizuje celkový bitový tok a koeficienty lineární predikce. Po kvantování je použito „Rice coding". Při kompresi jsou k dispozici dva módy základní a pokročilý. Základní používá během rozdělování skladeb na bloky „Rice coding". Zato druhý pokročilý mode analyzuje distribuci zbytků přebytků a prioritních bitů pomocí blokové kontroly Gilbert-Moorového kódování, a nakonec se použije „Rice coding".

TTA je bezeztrátový kodek (viz. [22] a [16], který umí komprimovat vstupní soubory na 30 až 70 procent jejich původní velikosti. Většina přehrávačů umí tento méně známý kodek přehrát už od základu nebo má rozšíření obsahující formát TTA. Základní přehrávače, které jsou dodávány s operačním systémem nejdou většinou použít v základním stavu. V tomto případě je třeba doinstalovat přehrávače třetích stran nebo dříve zmíněné rozšíření. Je to otevřená technologie a algoritmus je navržen podobně jako MPEG-4 ALS a FLAC, se kterými sdílí moderní audio techniky. Umí komprimovat vícekanálové vstupy s kvalitou zvuku 8, 16 nebo 24 bitů při vstupním souboru WAVE. Vzájemné vtahy v jednotlivých kanálech a rozdělení na bloky jsou prováděny pomocí IIR (Infinite Impulse Response) filtrů. K minimalizování přebytků je použito Windrow-Hoffovo LMS (Least Mean Square error) algoritmu.

Monkey's Audio Codec je otevřený bezeztrátový audio kodek (viz. [16]) vyvinutý Mattem Ashlandem. První krok, který provede, je transformace signálu do středních hodnot, následně se zjistí diference kanálů (odečteme pravý kanál od levého). Jako poslední se zjistí nejmenší signifikantní bit z prvního kroku neboli ze středních hodnot. Následně se použije „first-order" lineární prediktor s adaptivním offsetovým filtrem založeným na umělé inteligenci. Rozdíl mezi originálním signálem a predikovaným signálem je nakonec předložen výběrovému kodéru a následně setříděno do jednotlivých oken. Pro kontrolu se používá MD5 součet originálního signálu.

Ačkoliv je ALAC veden pod Apache licencí od 27 října 2011, tak je známo jen málo důležitých informací díky uzavřenosti ze strany firmy Apple. Z druhotných zdrojů je známo jen pár informací o lineární predikci, kompresi zbytkových informací ze souboru a o modifikované verzi Golomb-Rice kódování (jak lze přečíst ve článku [16]).

2 Návrh řešení

Tato kapitola se bude zabývat především popisem struktury programu mé práce. Jako první se bude psát o hlavní funkci programu. Ta se zaměřuje především na spouštění jednotlivých kompresních algoritmů a následné změření charakteristik. Další neodmyslitelnou částí této práce je prezentace výsledků neboli generování grafů. Tuto problematiku můžete nalézt v druhé podkapitole. V třetí a poslední podkapitole se budete moct dozvědět informace o dalších větvích výzkumu a vytvořených funkcí, které jsem použil pro získání povědomí o používaných datech. Případné slepé cesty ve výzkumu budou popisovány v poslední podkapitole.

2.1 Hlavní funkce programu

Tato podkapitola se bude věnovat jádru této práce, jak bylo zmiňováno už dříve. Nezákladnějším kamenem této práce jsou jednotlivé algoritmy. Algoritmy byli vybrány bezeztrátové, jak bylo požadováno v zadání práce. Z univerzálních algoritmů byly vybrány 7ZIP, BZIP2 a GZIP a z audio kodeků FLAC a TTA. Hlavní prioritou této práce je autenticita výsledků, proto byly vybrány oficiální verze algoritmů přímo od programátorů. Tyto všechny algoritmy jsou dostupné jako EXE soubor spustitelný pomocí příkazové řádky, právě tuto možnost jsem použil pro svojí práci.

Tyto algoritmy jsou spouštěné pomocí programovacího jazyku C# a tříd Process a ProcessStartInfo. Tato kombinace tříd byla použita pro usnadnění spouštění kompresních algoritmů, které prvně spustí program CMD.exe, který následně v příslušném adresáři, kde se nacházejí příslušné algoritmy, spustí tyto algoritmy s příslušnými parametry. Pro fungování testů bylo potřeba přidat ke každému příkazu “ && exit()”, pro ukončení běhu procesu v třídě Process.

K měření jednotlivých charakteristik byly vybrány třídy PerformanceCounter, DateTime, TimeSpan, FileStream a File. Třída PerformanceCounter je používaná především na měření procesorového času a využití paměti RAM. Kombinace DateTime a TimeSpan jsou použité pro získání informace o délce běhu algoritmů. Poslední kombinaci tříd FileStream a File následně používám pro získání informace o velikosti vstupního a výstupního souboru jak při kompresi, tak při dekompresi jednotlivých souborů.

Testy jsou spouštěny automaticky s jednotlivými parametry, které mohou změnit jednotlivé měřené charakteristiky. Tyto testy jsou spouštěny vždy pětkrát pro každou jednotlivou možnost a jsou spouštěny pro jednoduchost programu hromadně pro adresáře, ve kterých jsou samotné testovací soubory. Výsledky jsou uložené do souboru pomocí třídy StreamWriter. Tyto výsledky jsou nakonec zprůměrovány na konci testů věnovaných jednotlivým adresářům.

Tyto výsledky jsou zjišťovány pro jednotlivé soubory, ale v případě, kdy je možné zkomprimovat více souborů najednou, tak se provádí testy i pro tyto případy.

2.2 Interpretace výsledků

Interpretace výsledků je prováděna pomocí grafů. Každý graf je plněn daty, které byly naměřeny během běhů jednotlivých algoritmů a zprůměrovány. Každý z grafu, lze omezit ve podle všech dostupných informací s výjimkou naměřených hodnot, které mají závislost na zbytku informací. Každý graf je dále omezen počtem výsledků na jeden graf, aby se dalo pracovat s těmito grafy pohodlněji.

První z grafů zobrazují získané informace o naměřených charakteristikách do grafů typu boxplot. Informace, které jsou zobrazované, jsou spotřeba procesorového času, paměti RAM a čas běhu komprese. Tento typ grafu má dále dvě možnosti zaměření. Program pracuje se surovými daty, které jsou následně omezeny filtry a zobrazeny.

Druhý typ grafů zobrazuje kompresní poměry jednotlivých algoritmů je také zobrazen do grafu typu boxplot. Implementace omezení zobrazení počtu výsledků na jednom grafu byla také umožněna.

Jednotlivé grafy jsou generovány jak pro obecnou orientaci ve výsledcích, tak i pro jednotlivé parametry nakonec budou tyto grafy zobrazovány ve skupinách, aby bylo vidět jednotlivé změny v charakteristikách jednotlivých běhů kompresí. Pod grafem bude popsáno, k jakému algoritmu, popřípadě parametrům patří jednotlivá data.

Všechny tyto data budou generovány pomocí nugetů Plotly.NET [23] a Plotly.NET-.ImageExport [24], které jsou dostupné pro programovací jazyk C#. Pomocí Plotly.NET se budou generovat jednotlivé grafy typu boxplot. Třídy, které byly vybrány pro generování těchto grafů, jsou LinearAxis, Layout, Trace a GenericChart.

LinearAxis slouží k nastavení osy x a y. Layout nastavuje funkce a vzhled, které mají být obsahem výsledného grafu. Třída Trace slouží k výslednému spojení všech dat a připravuje data pro generování výsledného grafu. Jako poslední třída použitá pro samotné generování grafů byla použita třída GenericChart, která spojí všechny dostupné známé parametry a vygeneruje výsledný graf.

Výsledný graf má dvě hlavní možnosti pro vizualizaci těchto grafů. První možností je zobrazení v prohlížeči, ve kterém se dá přiblížit jednotlivé části grafu, výběr jednotlivých podoblastí, které mohou sloužit k rozdělení větších grafů na podgrafy a návrat k zobrazení, které bylo použito po vygenerování samotného grafu.

V mém případě byla použita druhá možnost pro vizualizaci dat pomocí druhého rozšíření nuget Plotly.NET.ImageExport, která umožňuje uložit výsledný graf jako PNG, JPG nebo SVG. Z těchto možností byla použita poslední možnost, a to uložení grafu ve formátu SVG, protože výsledný graf bude při jakékoliv velikosti ostrý. Jediná nevýhoda spojená s výběrem této technologie je větší náročnost při generování výsledného dokumentu, který bude trvat delší dobu, protože je složitější vypočítat všechny části takového obrázku.

2.3 Data pro testovací účely

Pro testovací účely byly poskytnuty archivy dat ukdale [25] a Archive of OSF Storage [26]. Každý z těchto archivu má odlišné formáty dat a také data (viz. [27]). Data archivu ukdale pochází z měření pěti domácností, ve kterých byli získávány jednotlivá data. Tyto data jsou uložená v jednotlivých adresářích pojmenována house_ a číslo jednotlivé domácnosti. Autor těchto dat vybral pro indexaci těchto domácností jako počáteční index hodnotu 1, a proto výsledný interval čísel pro výsledné pojmenování je v rozsahu 1 až 5. V každém z těchto adresářů s výjimkou třetího a čtvrtého, který zastupuje jednotlivé instance těchto domácností, jsou tři typy souborů.

První z nich nazvaný channel_. Tento textový řetězec následuje index sledovaného spotřebiče a je doplněn o příponu .dat. V tomto prvním typu souboru jsou obsaženy dva typy informací, které byly sledovány a byly rozděleny do dvou sloupců. První ze sloupců obsahuje informaci o výkonu tohoto zařízení ve wattech. Druhý sloupec vyjadřuje spotřebovanou energii ve volt-ampérech.

Druhý typ je pojmenovaný vždy stejně a obsahuje vysvětlení jaký index patří k jakému spotřebiči. První sloupec obsahuje číslo indexu a ve druhém je textově pojmenovaný vybraný spotřebič. Poslední ze souborů je main.dat a právě tento soubor není obsažen v domácnostech tři a čtyři. Tento soubor jako jediný obsahuje čtyři sloupce s daty. První vyjadřuje časovou značku, kdy měření bylo naměřeno. Druhý vyjadřuje výkon ve wattech pro celý dům a třetí vyjadřuje celkovou spotřebu spotřebovávané energie v celém domě ve volt-ampérech. Poslední ze sloupců obsahuje RMS hodnotu ve voltech neboli střední hodnotu ve voltech.

Druhý archiv čtyři podadresáře obsahující data o třech kanálech měřených dat. Každý z těchto kanálů byl pořízen při vzorkovací frekvenci 12,8 kHz. Jediný první kanál byl pořízen také při vzorkovací frekvenci 25 kHz. Všechna data byla získána z kanceláře v severním campusu Karlovo institutu technologií v Německu. Přičemž všechny soubory obsažené v tomto archivu byli uloženy při šestnácti bitovém kódování typu RIFF WAV (viz. [28]).

První z podadresářů začínající 1Channel při obou případech vzorkovacích frekvencí obsahují informace z voltmetrů a ampérmetrů tohoto kampusu. V rámci obou adresářů byly data rozděleny do dvou podadresářů pod jmény current a voltage. V podadresáři current jsou uloženy informace o jednotlivých proudech v ampérech a podadresář voltage obsahuje data z voltmetru vyjadřují napětí v rámci kampusu. Každá naměřená charakteristika byla rozdělena do 120 souborů, které obsahují jednotlivé podadresáře a byli uloženy jako soubory typu WAVE a byl použit jeden kanál pro uložení dat.

Adresáře začínající 2Channel a 7Channel neobsahují už žádné podadresáře a data jsou uložena rovnou v těchto adresářích. V každém je obsaženo 60 souborů také typu WAVE. V adresáři začínají hodnotou dva byly použity dva kanály pro uložení dat a adresář začínající hodnotou sedm obsahuje sedm kanálů v každém souboru.

Pro větší možnosti byl použit jen archiv Archive of OSF Storage, protože obsahuje soubory uložené ve formátu WAVE, který je potřeba při kompresi pomocí audio kodeků. To znamená, že první adresář nebyl vůbec použit a pro testovány v rámci práce byly použity jen data obsahující jeden kanál při vzorkovací frekvenci 12,8 kHz a obsahující informaci o proudu. Tyto data byly uloženy do 120 souborů, proto stačily pro testování jednotlivých kompresních algoritmů.

2.4 Doplňující informace

Pro doplňující funkce byl použit programovací jazyk python. Tyto funkce mají jedno hlavní specifikum, a to práci s WAVE soubory. Programovací jazyk python byl použit kvůli horší podpoře programovacího jazyku C#, který pro toto odvětví nemá tak dobré knihovny. Tyto knihovny jsou použity pro vizualizaci obsahu jednotlivých souborů. Program umožňuje vizualizaci dat samostatně po jednom souboru nebo po skupinách. Tyto možnosti jsou možné ovládat z programovacího jazyku C# pro celistvost aplikace.

2.5 Diagram

Obrázek 2.1: Diagram aplikace – Nejvyšší dvě vrstvy

Obrázek 2.2: Diagram aplikace – Funkce komunikující s algoritmy

3 Měření charakteristik algoritmů

Tato kapitola shrnuje naměřená data získaná prováděním testů na jednotlivých kompresních algoritmech. Testy jsou spouštěny, tak aby na konci práce byl výstupem soubor s daty, který obsahuje co nejvíce možných kombinací jednotlivých parametrů, které se dají u každého algoritmu nastavit.

Testy jsou prováděny jednoduchým způsobem. Pro každou kombinaci parametrů, které mohou měnit vlastnosti jednotlivých algoritmů. Je prováděno pět identických testů (testů se stejnými parametry). Tyto data se zapíší do souboru. Každý výsledný soubor dat je specifikován jedním adresářem s daty. Protože tyto testy jsou prováděny po sobě, a tím testy se stejnými parametry se vyskytují na stejném místě za sebou, tak se zprůměruje každých pět po sobě jdoucích řádků s daty. Toto zprůměrování počítá také s chybou, a tudíž neprůměruje přímo každých pět řádků, ale zprůměrují se vždy ty po sobě jdoucí řádky, které mají stejné parametry. Specifikované v příkazu při spuštění testu, a to pro kompresi i dekompresi.

Hlavní naměřená data, která čítají využití procesoru, paměti RAM a délka běhu jednotlivých algoritmů, jsou rozděleny do jednotlivých grafů z důvodu lepší orientace. Dále je vypočítán kompresní poměr jednotlivých algoritmů a zobrazen v grafech jako boxplot graf, který udává v procentech, jak velký výsledný soubor oproti vstupnímu souborů. Každá hodnota je tvořena minimálně stodvaceti hodnotami.

Hlavní naměřená data, která jsou spotřeba procesorového času, spotřeba paměti RAM a délka běhu jednotlivých kompresí, jsou zobrazeny také v grafech typu boxplot. Přičemž každý typ měřených dat je vždy věnován jeden graf, který se mění podle typu požadovaných parametrů. Pro využití procesorového času a délku běhu jednotlivých typů operací byly použity milisekundy jako hlavní jednotka. Spotřebě paměti RAM byly přiřazeny megabajty. Kompresní poměr je prezentován v procentech.

3.1 Testy na jednom souboru

Tato kapitola se zabývá kompresemi jednotlivých souborů. Na vstupu je specifikována cesta ke vstupního souboru, který je určen pro komprimaci. Následně se každý soubor, který se vyskytuje ve specifikovaném adresáři, zkomprimuje a jsou získány výsledky. Na začátku každé podkapitoly se budete moct dočíst, základní neomezené informace o jednotlivých algoritmech. Následně se budou přidávat naměřené hodnoty, které se získali při omezení jednotlivých parametrů, které poukazují na detailní informace o jednotlivých algoritmech. U každé sledované charakteristiky je jen první graf věnovaný všem algoritmům a druhý všem s výjimkou algoritmů TTA a LZW. TTA a LZW nemají žádnou možnost specifikace parametrů a kompresní metody, filtry a vlákna podporuje jen algoritmus 7ZIP.

3.1.1 Komprese

Spotřeba procesorového času

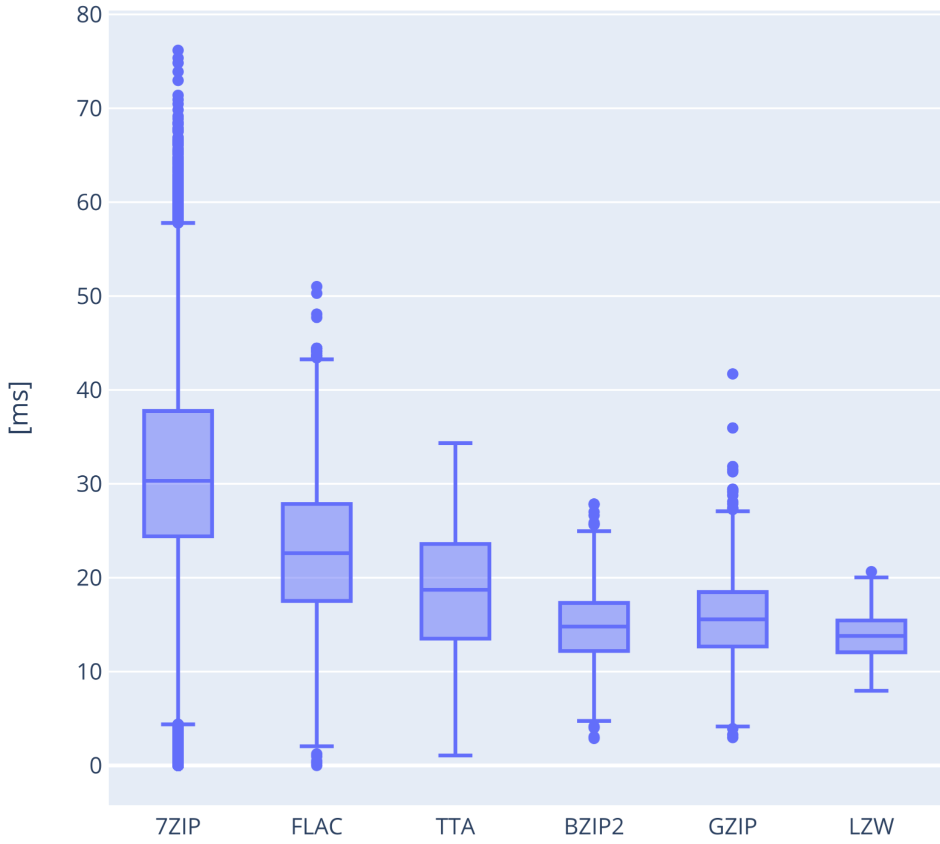
Na prvním grafu 3.1 jdou vidět obecné informace o spotřebě procesorového času při kompresi. Když se podíváme na nejvíce vyskytující spotřebu procesorového času, tak nejlepší kandidáti na kompresi jsou algoritmy BZIP2, GZIP a LZW. Tyto algoritmy ve většině případů (pokud se nechceme zaobírat podrobným nastavováním parametrů) si vezmou málo procesorového času. Nejméně procesorového času si, ale vezme při určitých parametrech algoritmus 7ZIP, ale lze vidět, že také při některých parametrů zaměstná nepoměrně více procesor než konkurenti, a i než sám o sobě v nejnižších možných hodnotách. V průměru se zde pohybují algoritmy, FLAC a TTA.

Druhý graf 3.2 věnovaný kompresi se zaměřuje na spotřebu procesorové času při použití levelů. V průměru zde vykonávají nejlepší práci algoritmy BZIP2 a GZIP. BZIP2 především při použití levelů 1 až 2 a 4 až 6 a GZIP při použití levelu 5 a 7. Lze zde ale vidět stejný efekt jako u minulého grafu, že při určitých parametrů je nejlepší algoritmus 7ZIP, ale nadále nelze zpozorovat původce nejlepších výsledků. Proto usuzuji, že parametr level nemá na malou spotřebu moc velký vliv, i když se zde vyskytují nejvíce odlehlé hodnoty u základní komprese a u kompresí využívající levely 5 až 9, které můžou poukazovat trochu na cestu ke zmíněným lepším výsledkům. Nejnižší hodnoty byly získány také od algoritmu FLAC při použití filtru jedna.

Na dalším grafu 3.3 nalezneme, jaká je návaznost spotřeby procesorového času na kompresní metody a filtry. Nejlepší ve skupině nejpočetněji vyskytujících se hodnot jsou kompresní metody LZMA, LZMA2 a komprese bez specifikace kompresní metody nebo filtru. Celkové nejlepší výsledky lze naleznout u všech možností až na kompresní algoritmy LZMA a LZMA2, proto v cestě při hledání algoritmu spotřebovávajícího nejméně procesorového času vypadávají tyto kompresní metody nejhůře.

Následující graf se pro změnu věnuje vlivu spotřeby procesorového času při použití vláken. Hodnoty jsou si zde velmi podobné, proto lze vidět hned na první pohled, že tento parametr nemá moc velký vliv na výslednou práci procesoru. Protože tento graf má stejné průběhy u všech možností, byl tento graf přesnut do přílohy.

Další graf poukazuje na spotřebu procesorového času při použití kombinace levelů a kompresních metod nebo filtrů. Nejpočetnější nejlepší výsledky jsou zde k vidění u levelů 2 až 5 při nespecifikování kompresních metod a filtrů, také při specifikování kompresních metod LZMA a LZMA2 a při levelu 8 a filtru ARMT, ale ani jeden znova nevykazuje jedny z globálně nejlepších výsledků. Jediný dvě výjimky jsou zde filtr ARMT dokonce několikrát a jednou kompresní metoda LZMA při nespecifikování žádného levelu. Z většiny se vyskytují nejnižší hodnoty u kompresních filtrů, u kterých je prakticky jedno, jaký vybereme lvel a u kopírovacích metod. Nejnižší

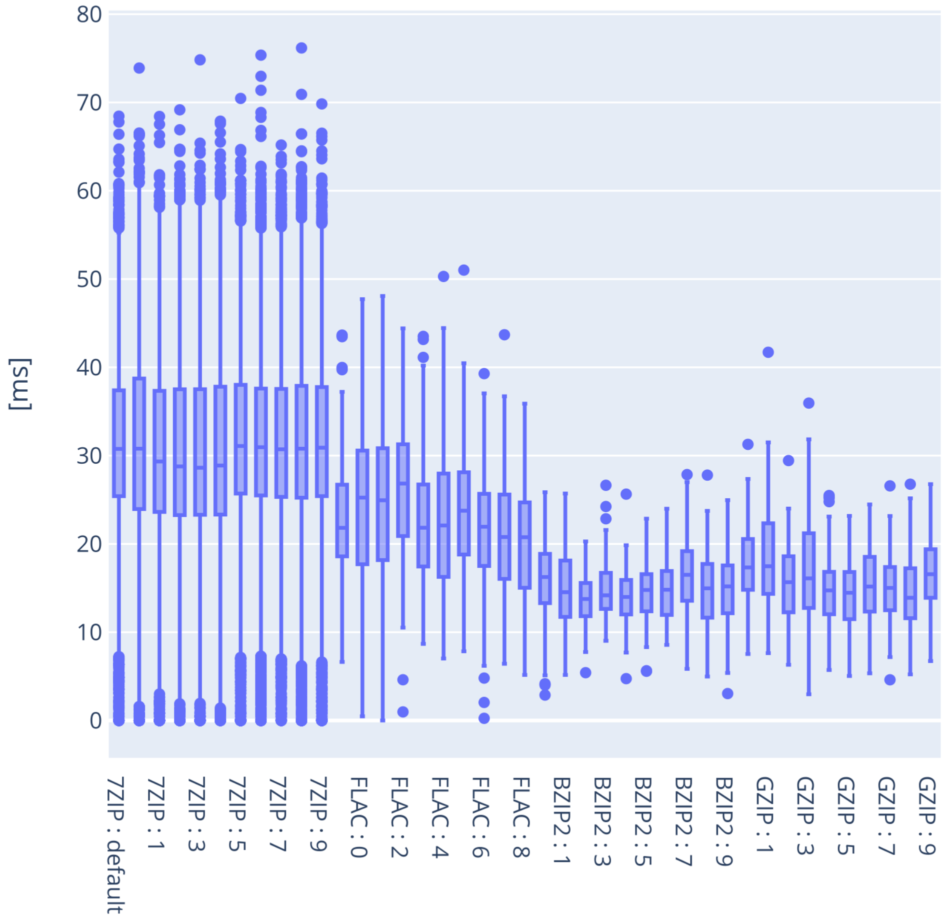


Obrázek 3.1: Graf zobrazující obecné vlastnosti kompresních algoritmů – CPU

hodnota je také k vidění u levelu jedna a bez specifikování druhého sledovaného parametru.

U skupiny parametrů věnujícím se levelu a vláknům jsou nejlepší výsledky nejpočetněji vyskytujících se hodnot přibližně na jedné úrovni, a i zobrazují alespoň jednou nejnižší hodnotu, proto jako u samostatných vláken nemají tyto nastavení moc velký vliv na výsledné komprese.

Předposlední sledovaná skupina sleduje vliv kompresních metod, filtrů a vláken na spotřebu procesorového času. Nejnižší nejpočetnější hodnoty nalezneme u kompresních metod LZMA a LZMA2 a při nespecifikování kompresních metod a filtrů v obou případech při použití vláken. Nejnižší hodnoty jsou, ale znova u kompresních filtrů a kopírovacích metod. Základní komprimace při použití jednoho vlákna se také



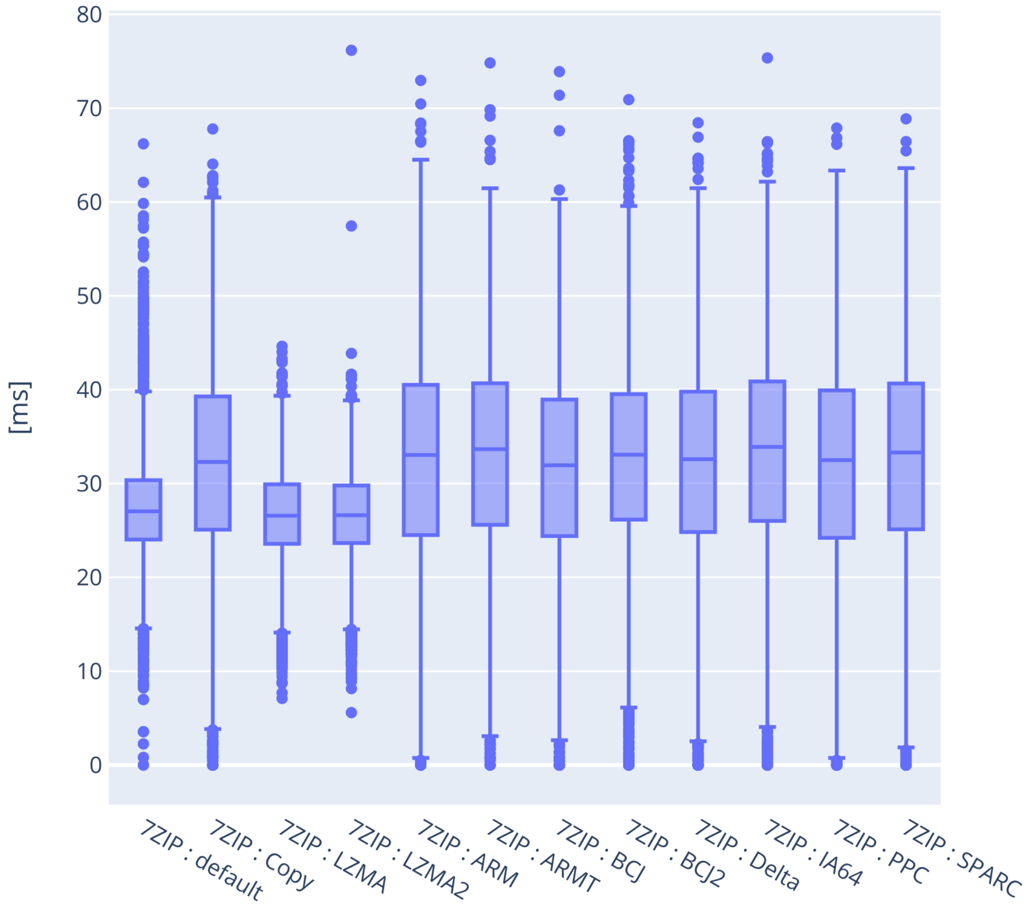
Obrázek 3.2: Graf zobrazující komprese podporující levely – CPU

velmi blíží nejnižší hodnotě.

Když shrneme všechny výsledky tak nejlepší volbou jsou algoritmy 7ZIP a FLAC. 7ZIP při nižších a průměrných kompresích a při použití kompresních filtrů. Audio kodek FLAC vychzel nejlépe při specifikování levelu rovnému jedničce.

Spotřeba paměti RAM

Na prvním grafu 3.4 věnujícím se spotřebě paměti RAM jsou při sledování nejpočetnějších výskytů mají jednotlivé algoritmy podobné hodnoty, proto na první pohled při prostém zprůměrování výsledků by nemuseli být vidět moc velké rozdíly, ale u algoritmu 7ZIP jdou vidět i odlehlé hodnoty, které vybočují ze standardu. Nižších hodnot je méně a vyšších hodnot je více. Protože více zajímavé jsou nižší hodnoty

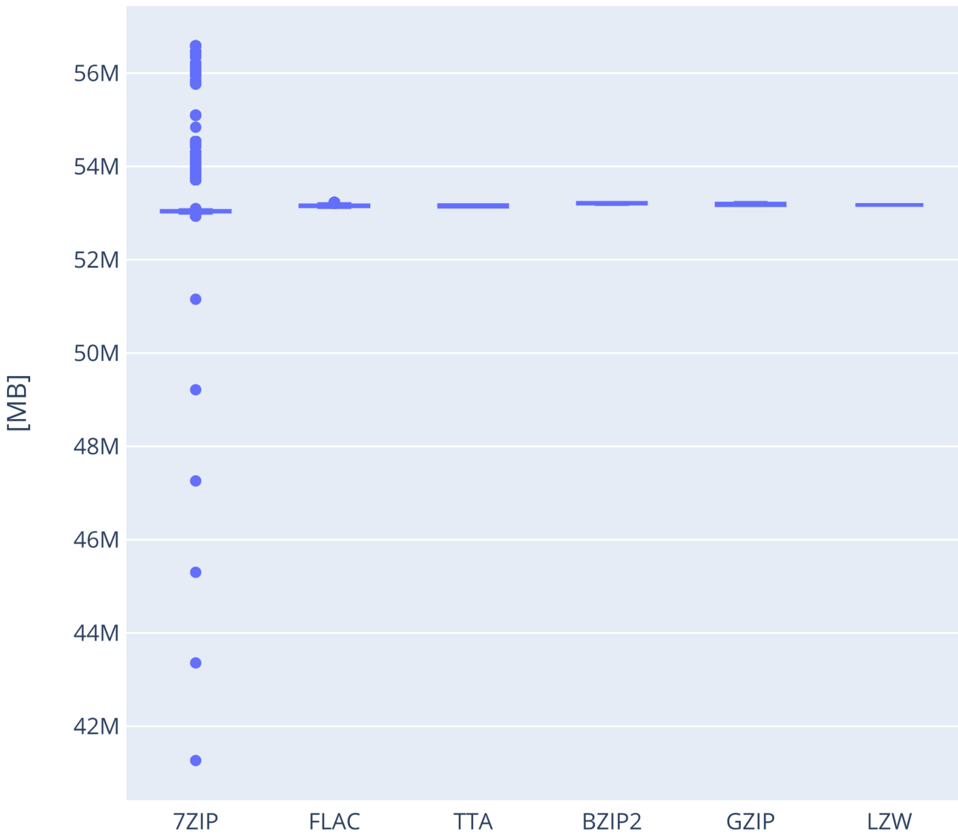


Obrázek 3.3: Graf zobrazující komprese podporující kompresní metody a filtry – CPU

kvůli ušetření paměti RAM, proto se hodí do budoucna spíše nižší odlehlé hodnoty.

Na grafu, který se věnuje spotřebě paměti RAM při použití levelů, lze vidět, že algoritmy BZIP2, GZIP a FLAC jsou nejvíce vyskytující se hodnoty spotřeby paměti RAM přibližně podobné. 7ZIP má tyto hodnoty nižší. Je zde už více vidět odkud se berou jednotlivé odlehlé hodnoty algoritmu 7ZIP. O tyto hodnoty se především starají komprese, které jsou maximálně průměrné. Přičemž nejnižší hodnotu lze zpozorovat u komprese, která neměla specifikovaný žádný level a následně levelu nula až po level pět se tyto hodnoty postupně zvyšují.

Na grafu 3.5 spotřeby paměti RAM při použití kompresních metod a filtrů, lze vidět přetrvávající efekt podobných nejvíce vyskytujících se hodnot této charakte-

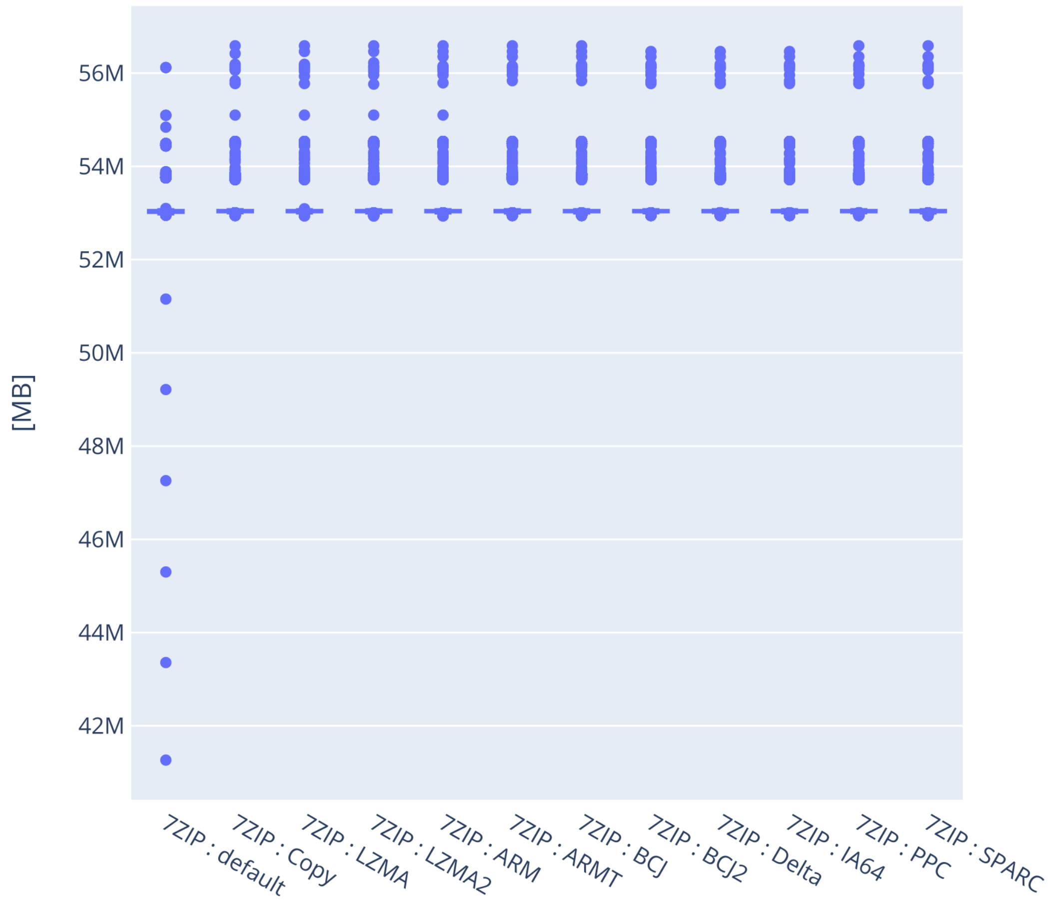


Obrázek 3.4: Graf zobrazující obecné vlastnosti kompresních algoritmů – RAM

ristiky. I tento graf zobrazuje odlehlé hodnoty, které u všech sledovaných parametrů mají při zaměření na vyšší odlehlé hodnoty podobný průběh, ale při nedefinování žádných kompresních metod nebo filtrů můžeme pozorovat i nižší odlehlé hodnoty, které mají až periodické rozmístění.

I na grafu zobrazující závislost paměti RAM na specifikaci vláken je vidět velká podobnost s minulými grafy, kdy nejvíce vyskytující hodnoty se nadále nahází kolem stejného prahu hodnot. Rozdělení vyšších odlehlých hodnot je spíše přiřazeno ke sloupci, který zobrazuje data při nespecifikování vláken. V tomto případě můžeme pozorovat i nižší odlehlé hodnoty, které udávají určitou periodicitu.

Na grafu zobrazující závislost spotřeby paměti RAM na parametrech level a kompresních metod nebo filtrů lze zpozorovat, že nejvíce se vyskytující hodnoty využití



Obrázek 3.5: Graf zobrazující komprese podporující kompresní metody a filtry – RAM

této komodity jsou na dále prakticky stejné. Rozdíly jsou nadále jen u odlehlých hodnot, které u vyšších odlehlých hodnot jsou z většiny podobné, ale jsou tam i nějaké rozdíly. Zajímavější jsou nadále nižší odlehlé hodnoty, které se vyskytují u nespecifikování těchto parametrů, u této možnosti je sledovaná hodnota nejnižší. Následně se tato hodnota postupně zvyšuje od hodnoty levelu nula až po hodnotu čtyři při nespecifikování kompresních metod nebo filtrů.

Na grafu, který se věnuje především závislosti spotřeby paměti RAM na parametrech level a vláknech lze vidět úplně stejný průběh jako na minulém grafu.

Graf věnující se spojitostí parametrů kompresních metod nebo filtrů a vláken na hodnoty spotřeby paměti RAM spojuje informace z minulých grafů. Nejvíce vyskytující se hodnoty spotřeby paměti jsou i tady stejné. Odlehlé hodnoty se nadále více vyskytují u nespecifikování vláken, přičemž nižší odlehlé hodnoty se vyskytují jen u nespecifikování kompresních metod ani vláken. Nejvíce vyšších odlehlých hodnot je pro změnu při nespecifikování vláken a při použití kompresních metod nebo filtrů, které jsou především Copy, LZMA, LZMA2, ARM, ARMT, BCJ, BCJ2, Delta, IA64 a PPC.

Když shrneme všechny známé informace, zjistíme, že nejnižší hodnoty se dají získat především bez specifikace žádného parametru a následně lze použít algoritmus 7ZIP při nespecifikování žádné kompresní metody nebo filtru a vláken při specifikace levelu od nuly po hodnotu 4. Tyto hodnoty jsou k vidění jen v jednom exempláři, proto bych se na tyto hodnoty moc nespoléhal.

Délka běhu algoritmu

Při zaměření na nevíce vyskytující se hodnoty délek běhu (viz. 3.6) jednotlivých kompresí jsou nejlepší 7ZIP, FLAC a TTA. Nejnižší hodnoty se dají zpozorovat také u těchto algoritmů. Když bychom tyto algoritmy chtěli seřadit od nejrychlejších po nejpomalejší, tak bychom je seřadili přesně ve stejném pořadí, jak byli vyjmenovány. Jediný algoritmus 7ZIP má hodně odlehlý hodnot, které z většiny jsou docela dost vzdálené od nejvíce se vyskytujících hodnot.

Na grafu 3.7 nejrychlejších kompresí při použití parametru level vykazují nejlepší hodnoty algoritmy 7ZIP a FLAC. Přičemž nejrychlejší je algoritmus 7ZIP. Hodnoty jsou nejnižší u všech možnostech parametru level, ale nejnižší jsou u levelu nula, který nekomprimuje, ale jen kopíruje data do archivu 7ZIP. Při nedefinování parametru level a při hodnotách pět až devět lze pozorovat nejvíce odlehlých hodnot.

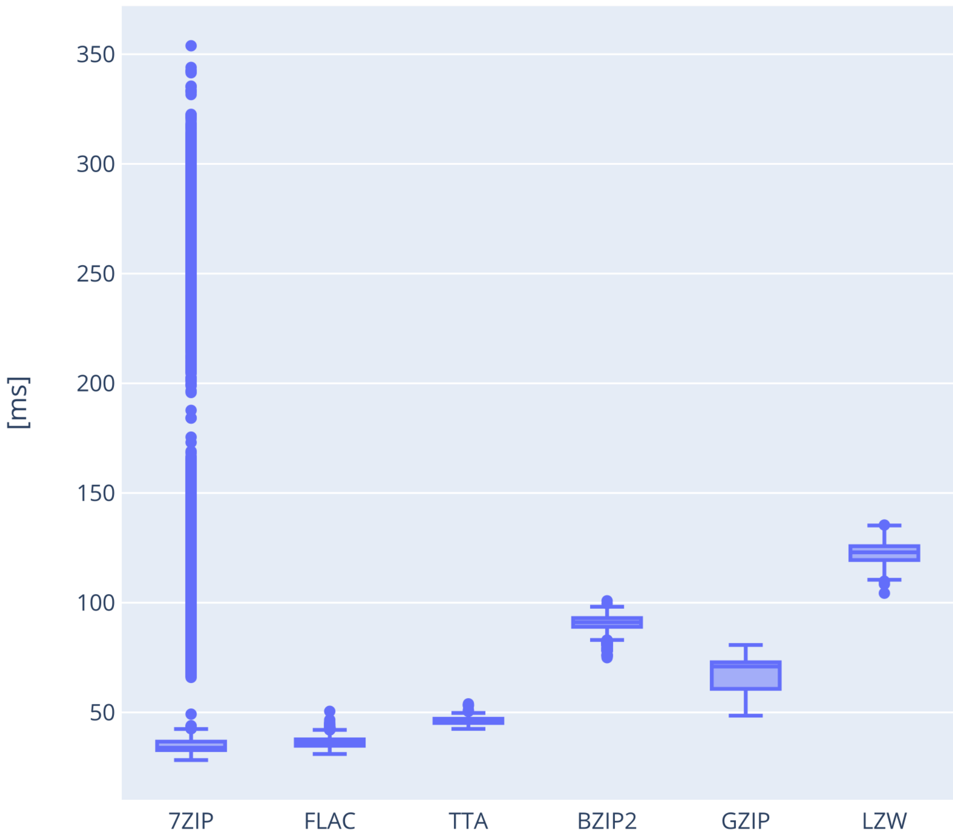
Na grafu 3.8 zobrazující délku běhu komprese je při použití kompresních motod nebo filtrů ve většině případů nejrychlejší kompresní metoda Copy a všechny použité filtry a nejpomalejší je komprese bez použití sledovaného parametru a kompresní metoda LZMA a LZMA2. Když vezmeme všechny výsledky, které nemusí být zastoupené u většiny výsledků, tak se k nejlepším výsledkům přidá i komprese, u které se tento parametr nenastavil.

Při použití vláken je algoritmus rychlý ve většině případů, ale jsou zde u každého testu s vlákny i odlehlé hodnoty, při kterých délky běhu nejsou moc rychlé. Nejpomalejší výsledky se vyskytují nejvíce u specifikace jednoho vlákna.

Při zaměření na skupinu dvou parametru level a kompresních metod nebo filtrů jsou nejrychlejší běhy algoritmu při použití kompresní metody Copy a většiny filtrů. Nejpomalejší jsou spíše běhy algoritmů při nespecifikování levelů a u levelů větších než 4 v kombinaci bez kompresních metod nebo filtrů a při specifikování kompresních metod LZMA a LZMA2. V průměru se pohybují stejné specifikace, ale při kompresích využívající levely jedna až čtyři.

Na grafu zaměřující se na kombinaci parametrů level a vláken lze zpozorovat, že většina kompresí je rychlá. Při specifikaci jednoho vlákna nemusí být komprese moc rychlá. Nejrychlejší jsou komprese při použití levelu rovném nule a je zde prakticky jedno, kolik vláken bylo nastaveno.

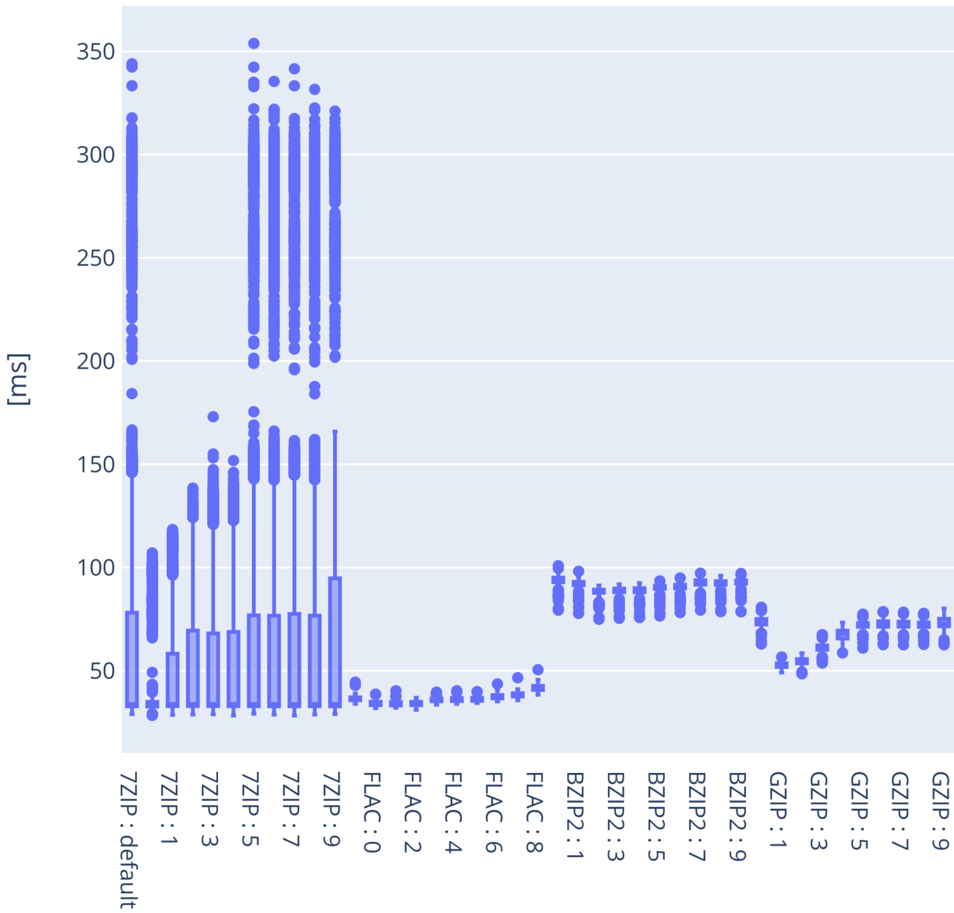
Na grafu 3.9 zaměřující se na parametry kompresních metod nebo filtrů a vlákna



Obrázek 3.6: Graf zobrazující obecné vlastnosti kompresních algoritmů – Časy běhu

lze zpozorovat stejnou charakteristiku jako na samostatných grafech věnujících se kompresním metodám nebo filtrů a vláknům. Lze zde zpozorovat nejrychlejších kompresí při nespecifikování kompresních metod nebo filtrů, při specifikaci filtrů a kompresní metody Copy. Kompresní metody LZMA a LZMA2 jsou pomalejší.

Nejrychlejší jsou především filtry (například PPC, SPARC, IA64 a další). Při specifikaci levelu nula až čtyři je větší šance na rychlejší kompresi v jakémkoliv případě a při použití kompresní metody LZMA a LZMA2 jsou komprese pomalejší.



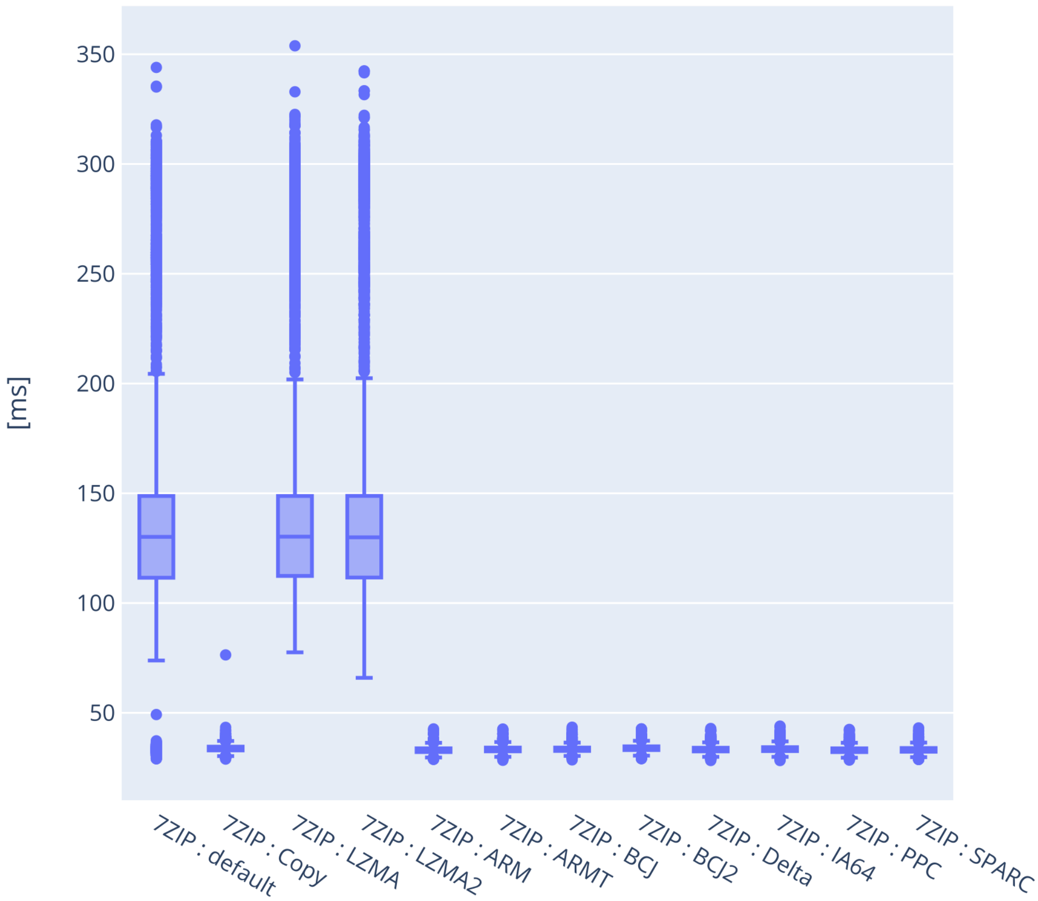
Obrázek 3.7: Graf zobrazující komprese podporující levely – Časy běhu

3.1.2 Dekomprese

Spotřeba procesorového času

Z celkového hlediska spotřeba procesorového času u dekomprese u nejčastějších výskytů v rámci algoritmu (viz. 3.10) vychází nejlépe pro algoritmus LZW. Jako druhý se umístilo BZIP2 a třetí nejlepší TTA. Celkově nejlepší výsledky umožnili algoritmy 7ZIP a FLAC. Všechny tyto algoritmy měli globálně nejlepší výsledky díky odlehlým hodnotám. Nevíce však algoritmus 7ZIP.

Když přidáme první sledovaný parametr level, tak podle nejvíce se vyskytujících hodnot vychází nejlépe algoritmus BZIP2 v celém svém rozpětí levelů, zbytek má podobné hodnoty, ale když přidáme do uvažování také odlehlé hodnoty, tak nejlépe



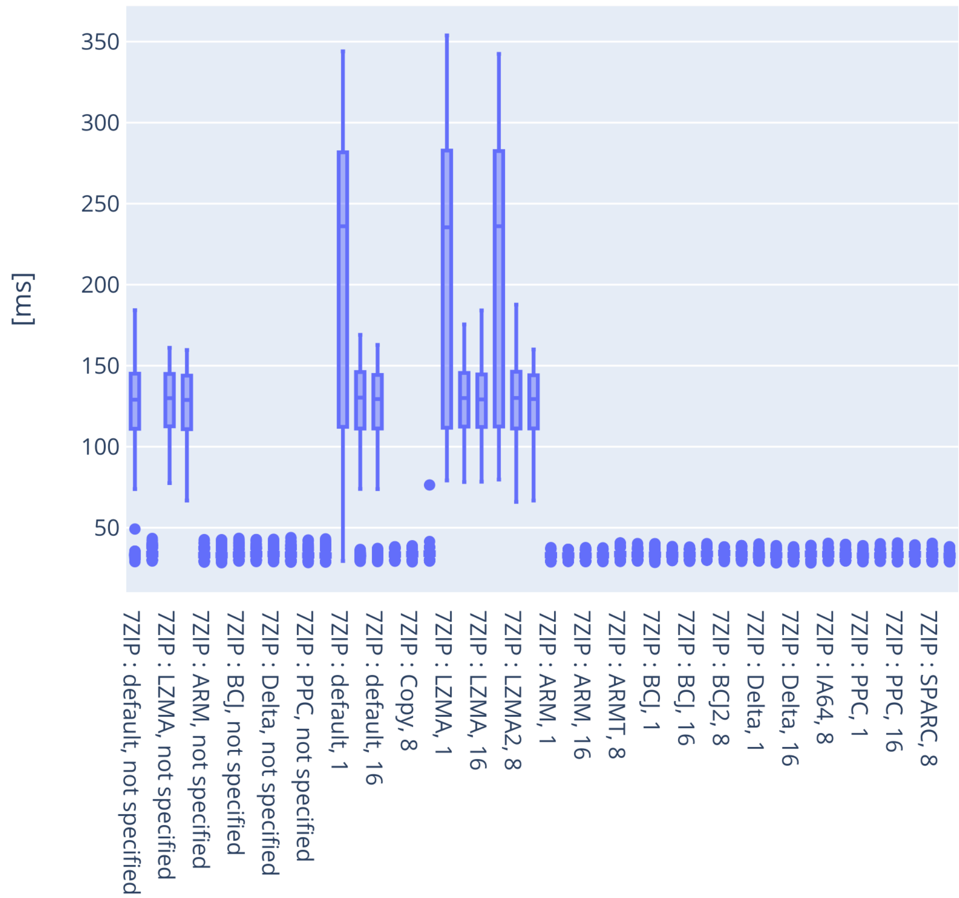
Obrázek 3.8: Graf zobrazující komprese podporující kompresní metody a filtry – Časy běhu

vychází algoritmus 7ZIP.

Při změně uvažovaného jediného parametru kompresních metod nebo filtrů (viz. 3.11) zjistíme, že podle mediánu a nejvíce vyskytujících se hodnot vychází nejlépe 7ZIP běhy bez specifikace kompresní metody nebo filtru a kompresní metody LZMA a LZMA2. Při uvažování i odlehlých hodnot do výsledku vychází nadále nejlépe algoritmus 7ZIP bez kompresních metod nebo filtru a s kompresními filtry s výjimkou filtru Delta.

Při zkoumání parametru pro vlákna jsou si průběhy velmi podobné, jak okolo středních hodnot, tak i u odlehlých hodnot.

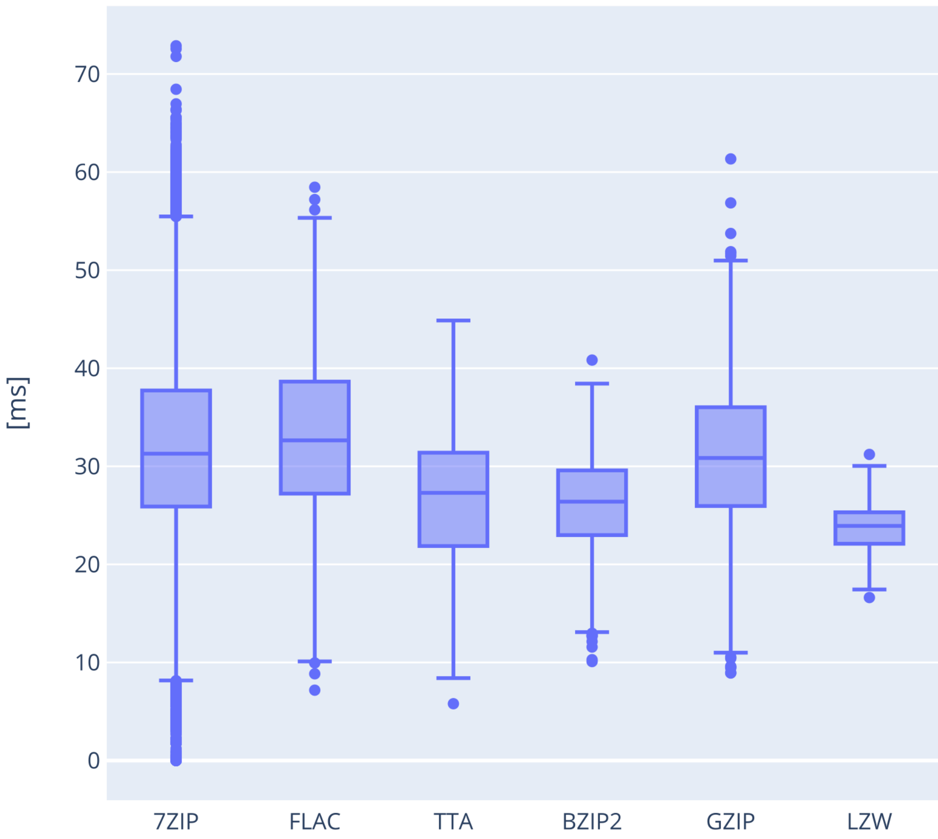
Na grafu zobrazující zkoumající skupinu parametrů level a kompresní metody



Obrázek 3.9: Graf zobrazující komprese s parametry kompresní metody, filtry a vlákny – Časy běhu

nebo filtry nejlepší výsledky kolem mediánu vychází nejlépe u kompresních metod LZMA a LZMA2 jako u komprese s levely ode 2 do 8 při nespecifikování kompresních metod nebo filtrů. Při uvažovaní nejlepších výsledků s odlehlými hodnotami vychází nejlépe level jedna bez specifikování druhého parametru a některé filtry jak při specifikování levelů, tak i bez specifikování levelů. Jsou to například filtry SPARC, IA64, PPC, ARMT a další. V některých případech si vedla ještě dobře kompresní metoda Copy.

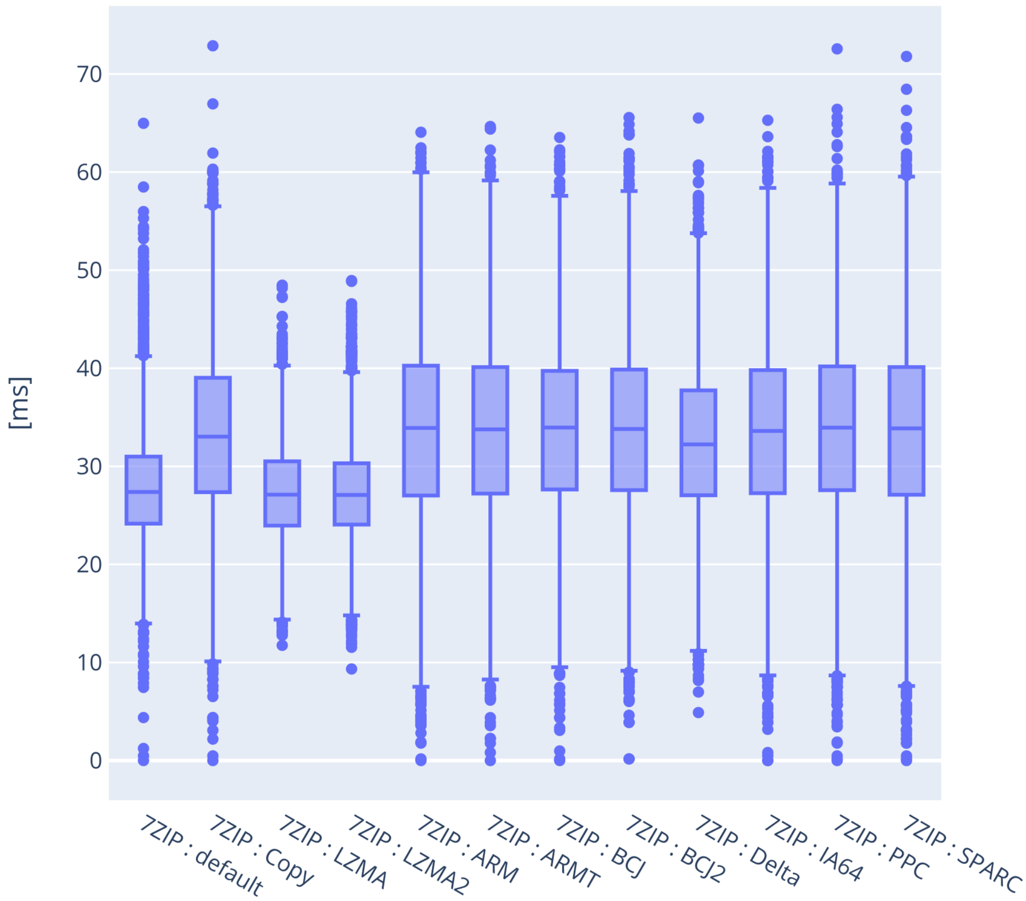
Při sledování parametrů level a vláken se vyskytují nejčastější hodnoty přibližně kolem stejné hodnoty jako u grafu věnující se jen vláknům. Oproti grafu s parametrem level se dají zjistit nové informace. Mezi nejlepšími výsledky je především level



Obrázek 3.10: Graf zobrazující obecné vlastnosti kompresních algoritmů při dekompresi – CPU

nula u všech zkoumaných hodnot vláken. Také mezi nejlepšími výsledky se vyskytuje komprese bez definice ani jednoho parametru. Když se nastavilo počet vláken na šestnáct, tak nízký hodnoty se objevovali u nenastaveného parametru level u hodnot čtyři, pět a osm. Při nastavení vláken na hodnotu osm by se dali ještě použít hodnoty levelu dva a pět.

Na grafu 3.12 sledující kombinaci parametrů kompresních metod nebo filtrů a vláken kolem mediánu vychází nejlépe kompresní metody LZMA a LZMA2 a hodnoty při nespecifikování kompresních metod ani filtrů, ale jako v předešlých případech nejnižší hodnoty jsou spíše u filtrů ARM, AMRT, IA64, BCJ, BCJ2, PPC a SPARC.

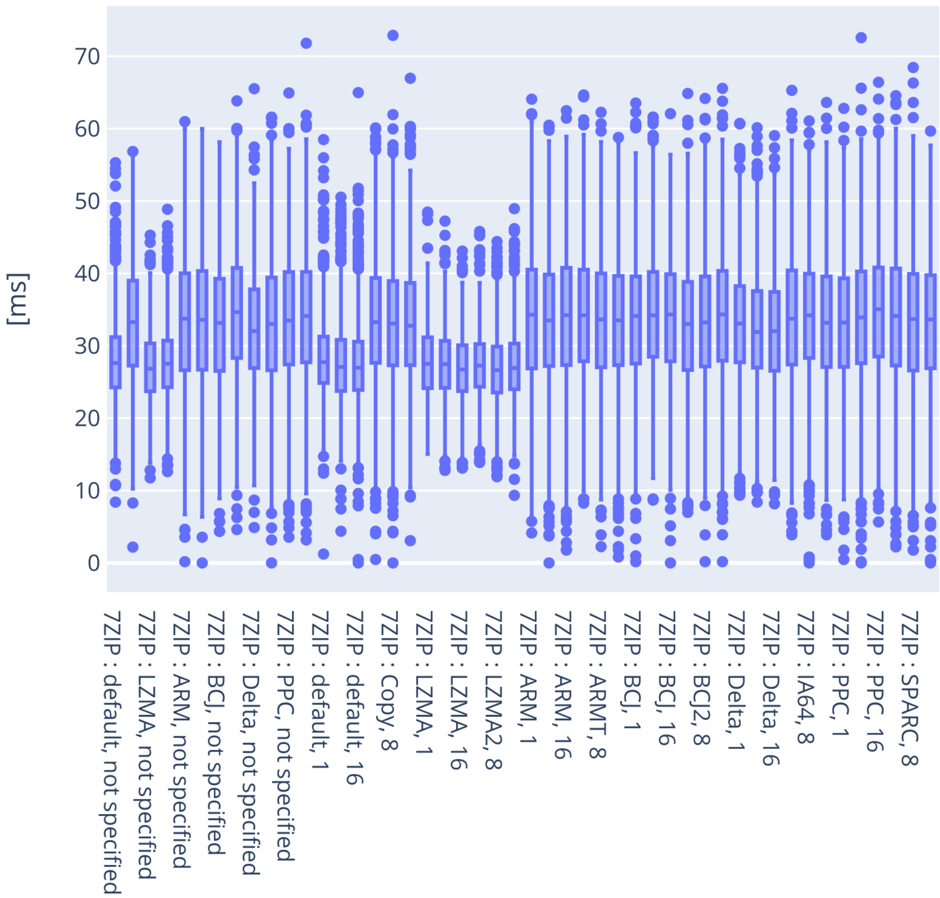


Obrázek 3.11: Graf zobrazující komprese podporující kompresní metody a filtry při dekompresi – CPU

Kompresní metody LZMA, LZMA2 a komprese bez specifikace kompresních metod při sledování hodnot kolem mediánu jsou nejnižší, ale při potřebě nejnižší spotřeby procesorového času, by se vybrali už dříve zmíněné filtry ARM, AMRT, IA64, BCJ, BCJ2, PPC a SPARC.

Spotřeba paměti RAM

Hodnoty spotřeby paměti RAM u dekomprese jsou nejnižší (viz. 3.13), jak u hodnot kolem mediánu, tak i globálně u algoritmu 7ZIP. Přičemž nejnižší hodnoty vykazuje algoritmus 7ZIP následují ho algoritmy FLAC a TTA. U algoritmu 7ZIP lze vidět také hodně nízkých odlehlých hodnot, které snižují jeho nejnižší hodnotu této

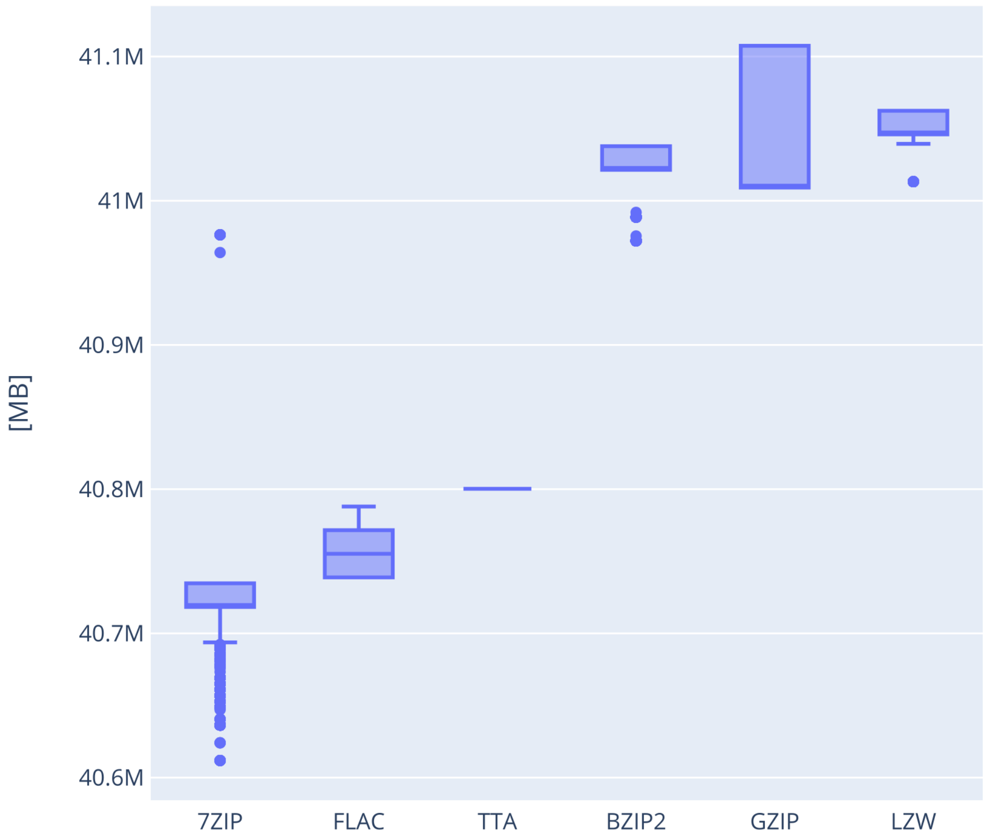


Obrázek 3.12: Graf zobrazující komprese s parametry kompresní metody, filtry a vlákny při dekompresi – CPU

charakteristiky.

I na grafu 3.14 zobrazující hodnoty získané při zkoumaní parametru level ukazuje stejný průběh jako minulém grafu, přičemž algoritmus 7ZIP vykazuje nejnižší hodnoty. Je prakticky jedno, jaký level uživatel zvolí, ale nejnižší spotřeby paměti vykazuje komprese bez parametru level, která je mírně nižší. Algoritmus FLAC je druhý a za algoritmem 7ZIP moc nestrádá a je jedno jaký level byl použit. Algoritmy BZIP2 a GZIP moc nedoporučuji při výběru algoritmu pro menší paměti RAM, protože rozestup od nejnižších hodnot je skokový.

Graf s parametrem kompresních metod a filtrů nevykazuje nějaké dramatické rozdíly při výběru kompresních metod i filtrů pro ušetření paměti při dekompresi.

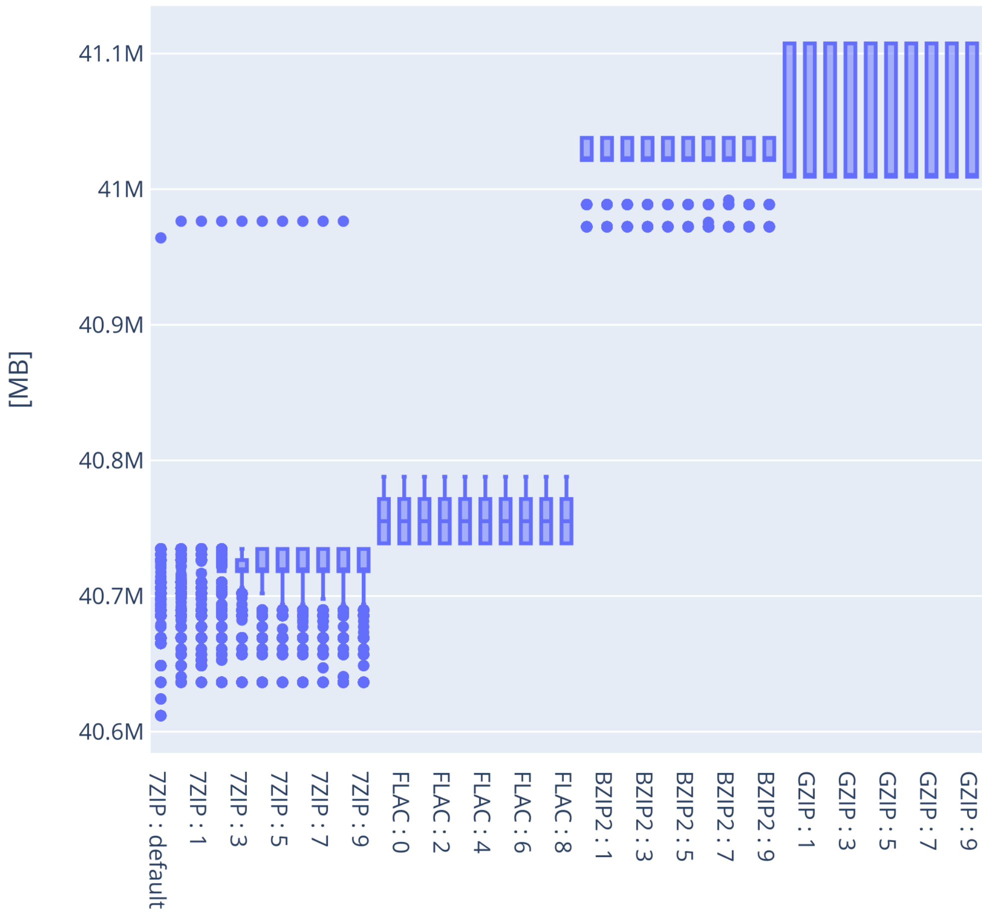


Obrázek 3.13: Graf zobrazující obecné vlastnosti kompresních algoritmů při dekompresi – RAM

Jediný rozdíl je ve třech hodnotách u nespecifikovaného parametru, ale tyto hodnotu je možné zanedbat.

U parametru nastavující počet vláken je situace velmi podobná, proto výsledek je stejný.

Na grafu, který zkoumá parametry level a kompresní metody nebo filtry, je rozdělen do tří pod grafů kvůli přehlednosti výsledků. Do dokumentu byl, ale použit jen první z nich, protože nejzajímavější informace se vyskytuje právě na něm a jsou stejná jako na předeslých grafech, jeno se změnila jediná informace, že nejnižší hodnoty byly naměřeny u běhů algoritmu 7ZIP bez specifikace parametru level. Zbytek informace je identický.



Obrázek 3.14: Graf zobrazující komprese podporující levely při dekompresi – RAM

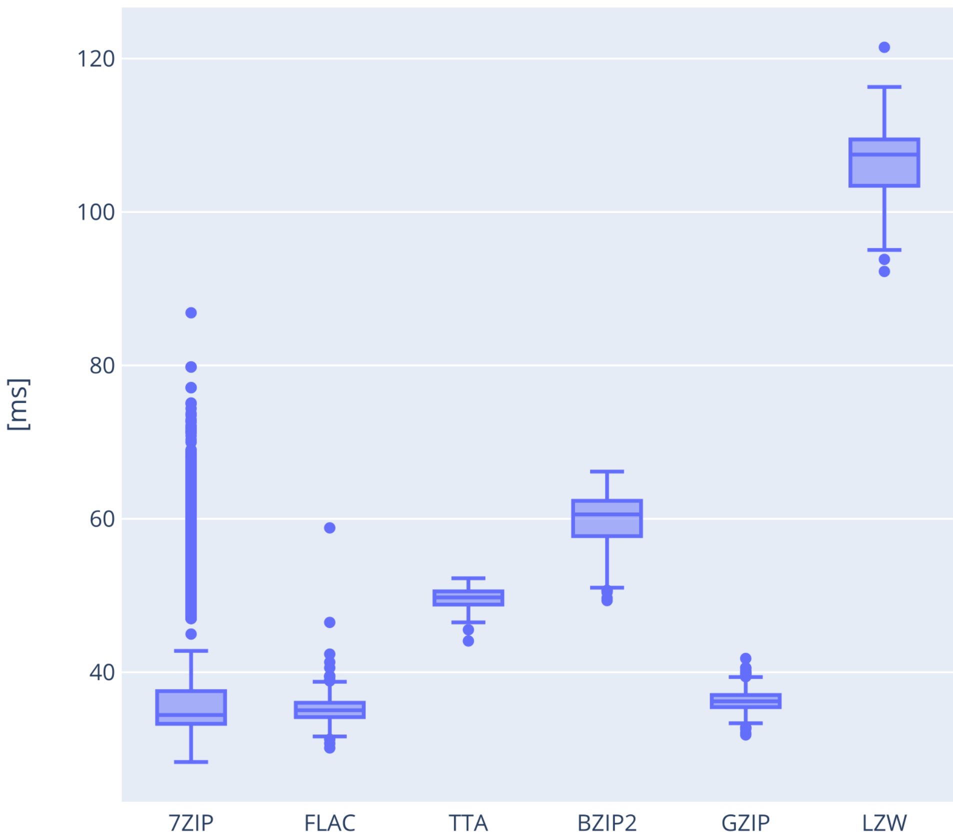
Tato informace je patrná i z grafu sledující kombinaci parametrů level a vláken, kde jsou především vidět nejnižší hodnoty u nespecifikovaného parametru level a jdou zde vidět vyšší odlehlé hodnoty u každého specifikovaného i nespecifikovaného parametru level bez specifikace vláken. Zbytek informace je podobný a tím nezajímavý pro tento graf.

Parametry kompresních metod nebo filtrů v kombinaci s vlákny vykazují podobné výsledky u všech možností, ale u specifikace jakékoliv kompresní metody nebo filtru se objevují nejnižší výsledky v grafu. Vysoké odlehlé hodnoty z předešlých grafů jsou způsobené nespecifikováním obou sledovaných parametrů.

Všechny vyšší odlehlé hodnoty se vyskytují u nespecifikovaných vláken a nejnižší hodnoty vykazují specifikované kompresní metody a filtry. Nejlepší volbou pro dekompresi pro ušetření paměti RAM zůstává algoritmus 7ZIP.

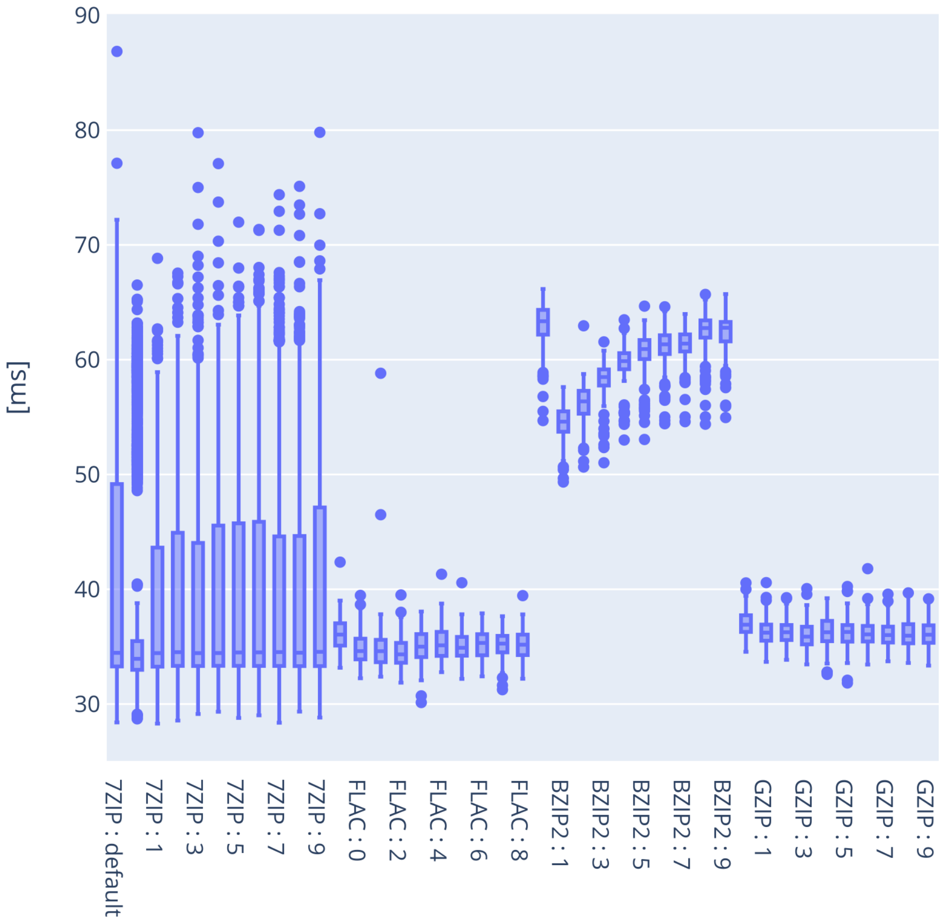
Délka běhu algoritmu

Nejrychlejší dekompresi vykazují algoritmy (viz. 3.15) 7ZIP, FLAC a GZIP i přes to, že algoritmus 7ZIP má i v tomto případě hodně odlehlých hodnot. Nejhorší a skokově vzdálené hodnoty od nejlepších výsledků má algoritmus LZW, který při délce běhu algoritmu je zdaleka nejhorší volbou. V průměru se pohybují algoritmy TTA a BZIP2.



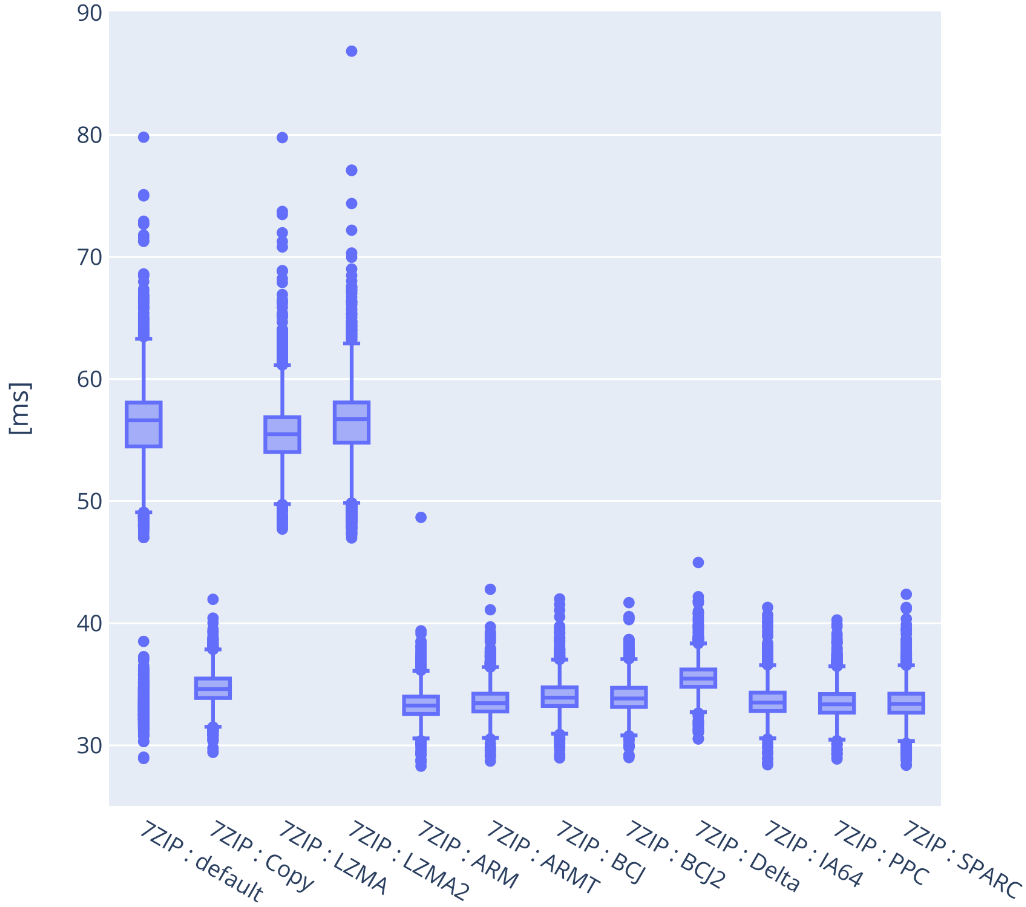
Obrázek 3.15: Graf zobrazující obecné vlastnosti kompresních algoritmů při dekompresi – Časy běhu

Na grafu 3.16 se zobrazenými výsledky při sledovaní parametru level, lze zpozorovat, že v průměru jsou nejlepší algoritmy 7ZIP, FLAC a GZIP. Tyto data jsou si velmi podobná, proto při výběru algoritmu, kde by byla důležitá délka dekomprese, jsou nejlepší volbou předem vyjmenované algoritmy. Pokud by nebylo možné zjišťovat u jakých dalších parametrů podává každý algoritmus nejlepší výsledky, tak nejlepší volbou zůstávají algoritmy FLAC a GZIP. 7ZIP jako nejlepší algoritmus má nejlepší výsledky při nespecifikování parametru level a při hodnotách nula, jedna a sedm.



Obrázek 3.16: Graf zobrazující komprese podporující levely při dekompresi – Časy běhu

Při zaměření na parametr kompresních metod a filtrů (viz. 3.17), pokud bychom se soustředili pouze na nejlepší výsledky hodnot kolem mediánu, tak by nejlépe vycházeli kompresní filtry i při trochu vyšších hodnot při použití filtru Delta a kompresní metoda Copy. Do celkových výsledků, ale musíme přidat běhy dekomprese bez specifikace kompresní metody. Nejhorší výsledky vykazují kompresní metody LZMA a LZMA2 a případě hodnot kolem mediánu i dekomprese bez specifikace parametru.



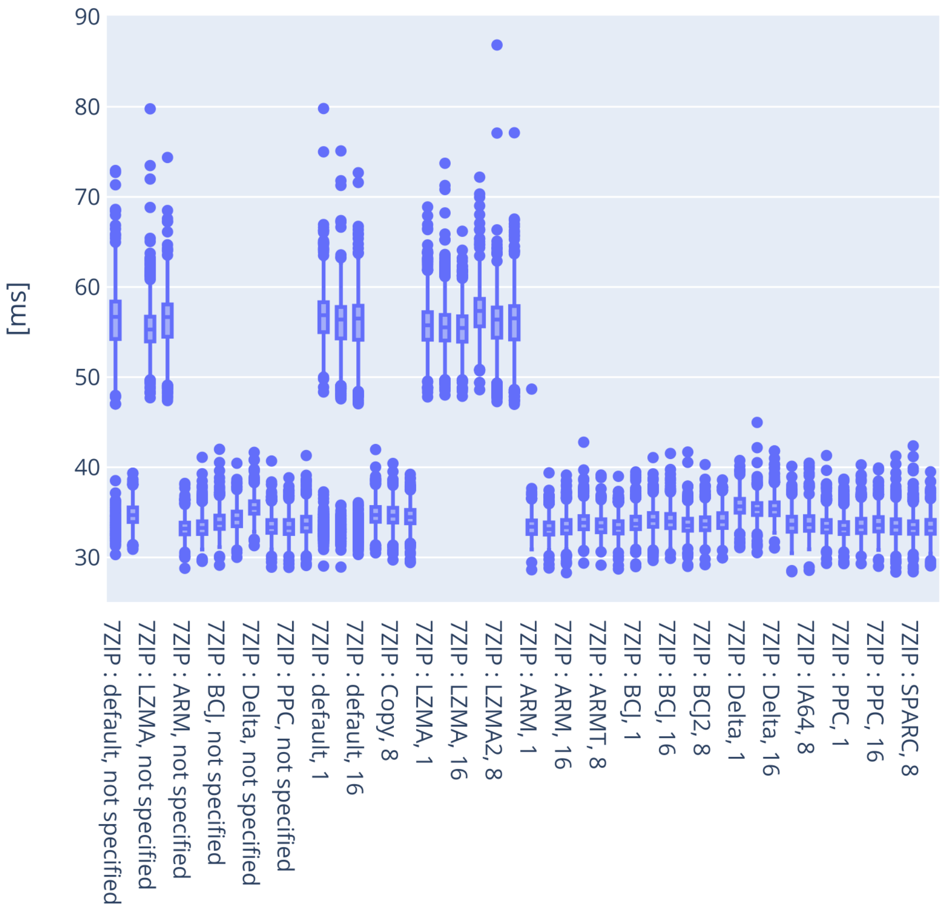
Obrázek 3.17: Graf zobrazující komprese podporující kompresní metody a filtry při dekompresi – Časy běhu

Při zaměření na vlákna lze zjistit, že i při dekompresi jsou všechny sloupce boxplot velmi podobný a neovlivňují délky běhu jednotlivých dekompresí.

Při sledovaní kombinace parametrů level a kompresních metod nebo filtrů vykazují nejlepší výsledky dekomprese se specifikovanými filtry a kopírovacími metodami. Při nespecifikování ani jednoho parametru, druhého parametru nebo při specifikaci kompresních metod LZMA a LZMA2 jsou hodnoty vyšší.

Při specifikacích levelu a vláken jsou data velmi podobná jen kopírovací algoritmy vykazují u hodnot kolem mediánu lehce nižší hodnoty.

U kombinace parametrů kompresních metod nebo filtrů a vláken jsou v globálu nízké skoro všechny hodnoty (viz. 3.18). Výjimkou jsou už dříve zmíněné kompresní metody LZMA a LZMA2. Při zaměření na hodnoty kolem mediánu se objevuje mezi nejlepšími výsledky i nespecifikovaní ani jednoho parametru.

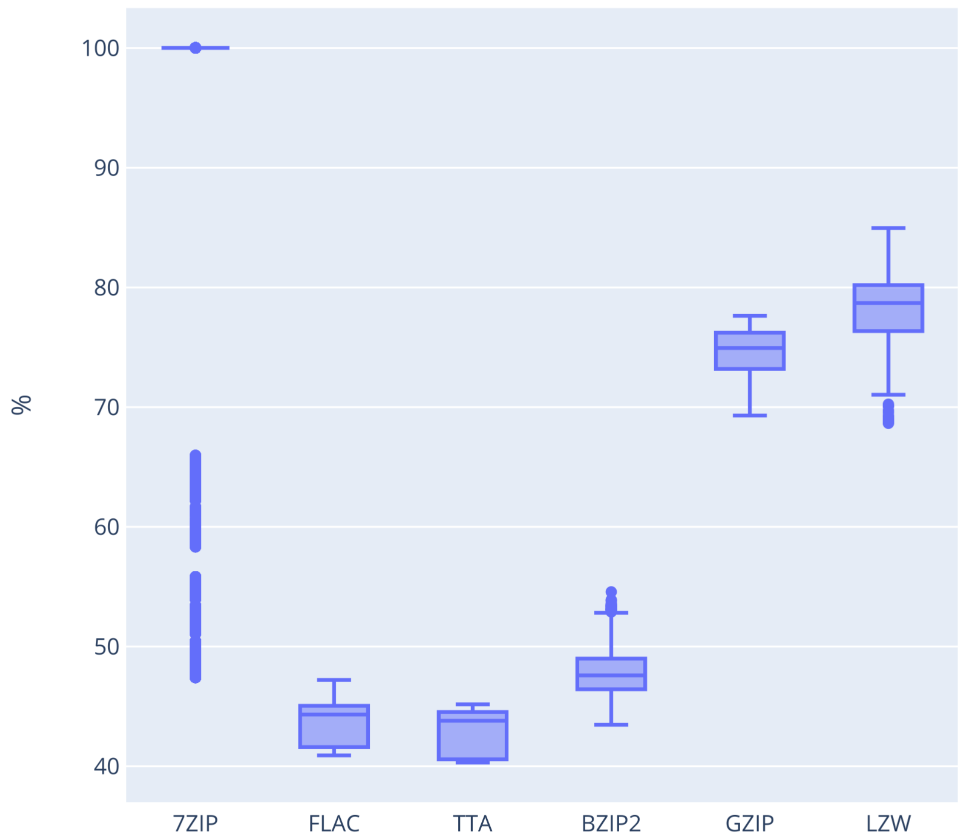


Obrázek 3.18: Graf zobrazující komprese s parametry kompresní metody, filtry a vlákny při dekompresi – Časy běhu

Při specifikaci kopírovacích metod a filtrů proběhne dekomprese rychle, ale při nespecifikování žádného parametru a kompresních metod jsou výsledky horší. Specifikace počtu vláken nemá žádný vliv na výslednou délku běhu dekomprese.

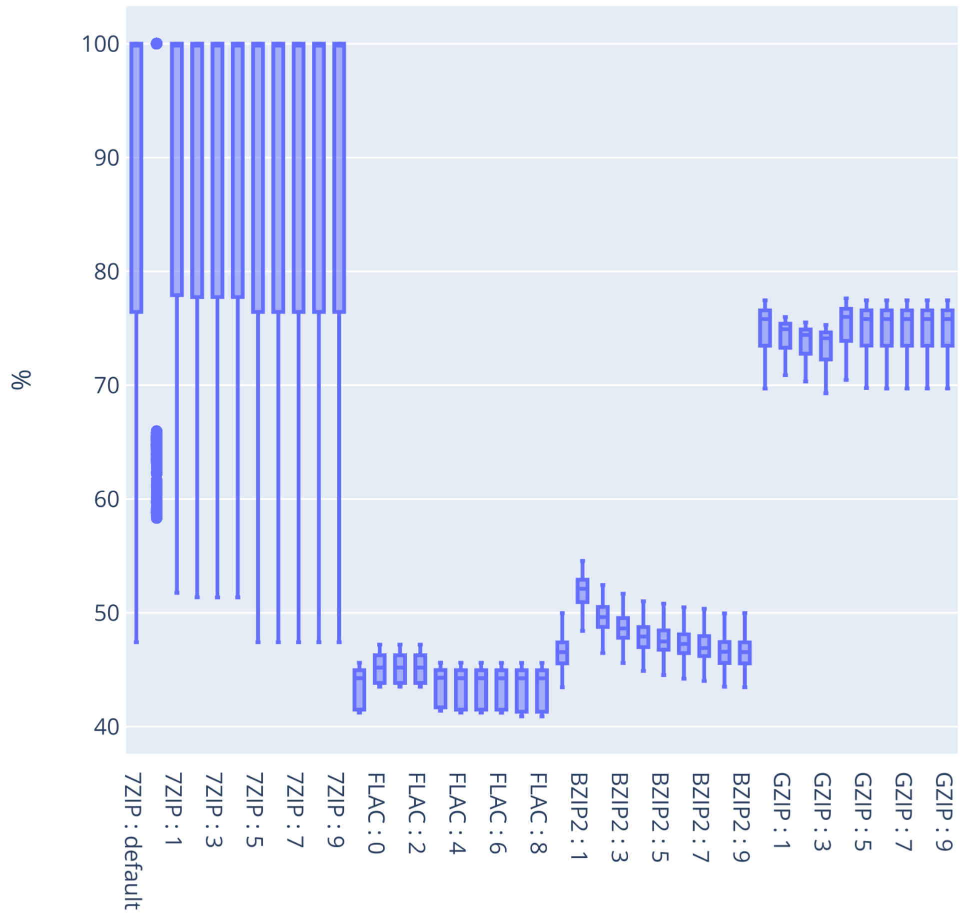
3.1.3 Kompresní poměry

V rámci kompresních poměrů nejlépe vychází nejlépe algoritmus TTA (viz. 3.19) a lehce za ním FLAC s kompresními poměry blížícími se hodnotě 40 %. Třetí nejlepší je algoritmus je BZIP2, který se nejlepšími výsledky blíží hodnotě 45 %. Algoritmus 7ZIP má výsledky nejlepších hodnot blížících se přibližně 48 %, ale většina běhu tohoto algoritmu neuspořila žádné místo. Algoritmy GZIP a LZW se řadí s nejlepšími hodnotami kolem 70 % výsledného souboru spíše k horším výsledkům.



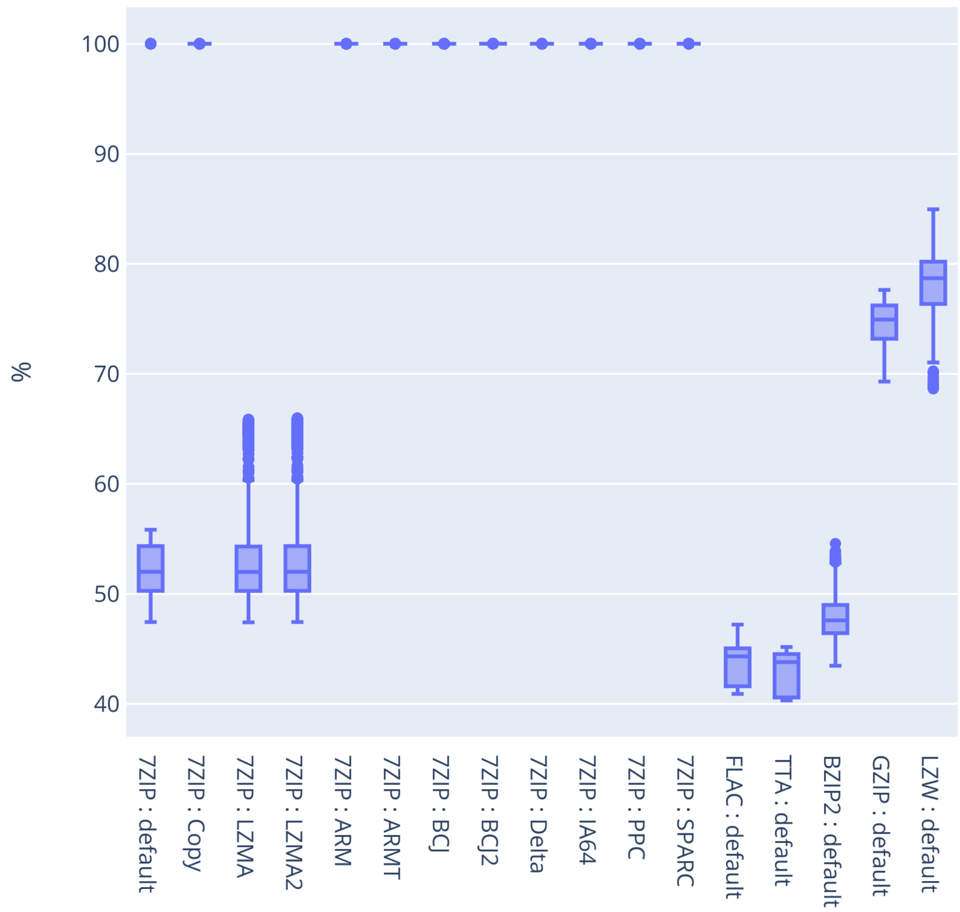
Obrázek 3.19: Graf zobrazující kompresní poměry

Pří specifikaci parametru level vychází nejlépe algoritmus FLAC a BZIP2 (viz. 3.20). Přičemž FLAC má nejvyšší komprese při nespecifikování tohoto parametru a při vyšších hodnotách, než je hodnota dva, při kterých se blíží komprimovaný soubor přibližně 40 % původního. U algoritmu BZIP2 se vyskytuje nejlepší výsledek pouze při nespecifikování parametru level blížící se přibližně 42 %. GZIP vykazuje průměrné výsledky kolem 78 %. Algoritmus je ve většině případů nejméně použitelný, ale v méně případech se blíží i k hodnotě 48 %, a to hlavně při nespecifikování levelu a levelech vyšších než čtyři.



Obrázek 3.20: Graf zobrazující kompresní poměry s parametrem level

Při specifikaci parametru kompresních metod nebo filtrů u algoritmu 7ZIP jsou nejlepší komprese při bez specifikace tohoto parametru a při specifikaci algoritmů LZMA a LZMA2 (viz. 3.21). Tyto komprese v nejlepším případě dosahují přibližně 48 % původního souboru. U nespecifikování tohoto parametru dochází v nejhorším případě komprese přibližně 57 % původního souboru a u LZMA a LZMA2 se v nejhorších případech pohybují hodnoty komprese přibližně okolo 68 % původního souboru. Zbytek možností tohoto parametru mají výsledný soubor stejně velký jako základní soubor, ale ani nejlepší výsledky při tomto parametru nedosahují kvalit algoritmů TTA, FLAC a BZIP2, které jsem přidal do grafu pro srovnání.



Obrázek 3.21: Graf zobrazující kompresní poměry s parametrem kompresní metody a filtry

Při specifikaci parametru nastavující počet vláken jsou všechny výsledky podobné, jak při nespecifikování vláken, tak při specifikování.

U parametrů level a kompresních metod nebo filtrů lze zpozorovat, že při specifikaci levelů i nespecifikování a nespecifikování kompresních metod nebo filtrů komprimuje algoritmus 7ZIP obstojně. Hodnoty se pohybují okolo 58 % pro nižší komprese, než je level pět, ale v ostatních případech bez specifikace druhého parametru jsou výsledky lepší a vyskytují se blíže 50 %. Při specifikaci kompresních metod LZMA a LZMA2 při jakémkoliv levelu komprimovala vstupní soubor pod 70 % a to i při specifikaci levelu rovnému nule. Nejlepší výsledky se dají zpozorovat u těchto algoritmů u level při hodnotách vyšších jak 4. Kompresní filtry nadále neušetří žádné místo na disku. V žádném případě, ale nestačilo nastavení parametrů tohoto algoritmu na algoritmus FLAC, ale nejlepší výsledky se pohybují blízko algoritmu BZIP2.

Při specifikaci parametrů level a vláken, lze zpozorovat, že nejhorší výsledky vykazuje nespecifikování ani jednoho parametru, při specifikaci levelu od pěti do 9 jak v případě nespecifikovaných vláken i specifikovaných a při nespecifikovaných levelů a specifikovaných vláken. Právě tyto komprese v některých případech jsou srovnatelné s kompresí BZIP2.

Na grafu, který se věnuje kompresním poměrům při parametrech kompresních metod nebo filtrů a vláken lze asi nejlépe zpozorovat v jakém případě je algoritmus 7ZIP použitelný. Jsou to hlavně nastavení, kdy se nespecifikuje žádný z těchto parametrů nebo když se specifikují pouze vlákna nebo při specifikaci kompresních metod LZMA a LZMA2 bez vláken i s vlákny.

Na posledním grafu jsou vidět výsledky při specifikaci všech parametrů. Můžeme vidět všechny dříve specifikované charakteristiky, které byli naleznuty už na minulých grafech. Především, že specifikace vláken nemá na výslednou kompresi žádný vliv. Při specifikaci parametru kompresních metod nebo filtrů jsou použitelné jen kompresní metody LZMA a LZMA2, a to bez specifikace levelů i se specifikací levelů, ale platí pravidlo, že nespecifikování levelů a levely vyšší než 4 podávají nejlepší výsledky.

3.2 Testy na více souborech

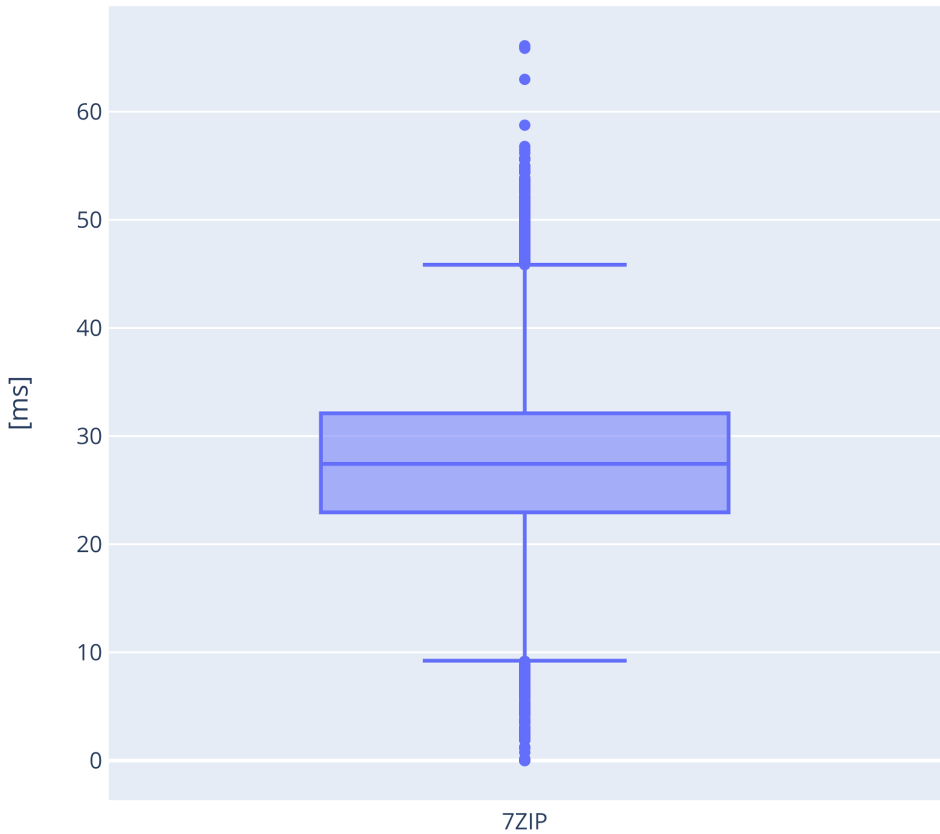
Algoritmus 7ZIP umožňuje komprimovat více souborů do jednoho výsledného souboru, proto bylo potřeba udělat všechny testy i pro komprimaci více souborů najednou. Pro účely práce jsem použil u každého jednoho testu tři soubory. Výběr souborů do jednotlivých skupin byl vybírán jednoduchých pravidlem. Při procházení souborů se přidal soubor do seznamu souborů při dosažení délky tři byl vždy tento seznam přidán do seznamu seznamu souborů. Každý seznam souborů obashuje jiné soubory.

3.2.1 Komprese

Spotřeba procesorového času

Na prvním grafu 3.22 celkových výsledku komprese více souborů lze zpozorovat interval naměřený pro algoritmus 7ZIP, který jako jediný umí komprimovat více souborů do jednoho výsledného souboru. Na tomto grafu se především dozvídáme, že procesor pracoval o poznání méně oproti jednotlivým souborům. Nejvíce je tato informace patrná z mediánu, který ukazuje o dvě milisekundy méně než u jednotlivých souborů. Dále je tato informace vidět na odlehlých hodnotách, které jsou v případě více souborů o poznání níže.

Na grafu, který se věnuje spotřebě procesorového času při definování parametru level lze vidět podobný trend jako u minulého grafu. Jednotlivé hodnoty se lehce



Obrázek 3.22: Graf zobrazující obecné vlastnosti kompresních algoritmů – více souborů, CPU

snížili. Lze také zpozorovat, že nejpoužitelnější kompresi dostáváme při nespecifikování tohoto parametru a při specifikaci levelu rovnému pěti a sedmi.

Při pozorovaní parametru kompresních metod nebo filtrů lze také vidět pokles hodnot. U některých kompresních metod a filtrů je pokles až 8 milisekund. Nejpoužitelnější filtry jsou především ARMT, Delta, IA64, PPC a SPARC.

Na grafu, který zkoumá přínosy parametru nastavující počet vláken, je vidět také pokles oproti kompresi jednotlivých souborů, ale nadále není moc velký rozdíl mezi počty použitých vláken.

Při použití kombinace parametrů levelu a kompresních metod nebo filtrů lze zpozorovat, že v rámci grafu jsou hodnoty velmi podobné. Jednou u nižších kompresí bez použití druhého parametru a při nižších kompresích a použití kompresní metody Copy jsou výsledky příznivější, ale není to dramatické. Pokles oproti kompresi jednotlivých souborů je také vidět, ale už není až tak markantní a ubylo některých nižších hodnot odlehlých hodnot než u jednotlivých kompresí, které z některých kombinací dělali při určitých podmínek použitelnější nastavení.

Na grafu s parametry level a vlákny jdou nadále vidět podobné hodnoty v rámci jednoho grafu, ale i na tomto grafu jde vidět rozdíl mezi jednotlivými soubory a komprimací více souborů najednou.

Na grafu, zobrazující kombinaci parametrů kompresních metod nebo filtrů a vláken, jsou velmi podobné hodnoty až na nějaké výjimky. Lze zpozorovat, že při nespecifikování prvního parametru nebo při specifikaci kompresních metod LZMA a LZMA2 v kombinaci s jedním vláknem mají hodnoty o poznání menší rozestupy mezi prvním a třetím kvartilem.

I u komprese více souborů lze vidět stejný trend použitelnosti jednotlivých kombinací parametrů. Nejpoužitelnější jsou nižší komprese bez použití kompresních metod nebo filtrů a při použití kompresních metod LZMA a LZMA2. Od vyšších kompresí, než je hodnota levelu 4 se výsledné hodnoty blíží ke stejné hodnotě.

Spotřeba paměti RAM

Celková spotřeba paměti RAM se také skokově snížila z přibližných průměrných 53,5 tisíce MB kolem mediánu se snížila spotřeba na 39 tisíc MB. Zmizely také nižší odlehlé hodnoty (viz. 3.23).

Při přidání do zkoumání parametru level se zdají všechny možnosti stejně použitelné (viz. 3.24). Když srovnáme výsledky s kompresí jednotlivých souborů, tak zmizely nízké odlehlé hodnoty a snížila se hodnoty kolem mediánu přesně jako na minulém grafu.

Na grafu 3.25, který sleduje parametr kompresních metod nebo filtrů, můžeme vidět, všechny změny, co se událi na minulých dvou grafech oproti kompresi jednotlivých souborů, můžeme sledovat i na tomto grafu. Především všechny kompresní metody, filtry i nespecifikování tohoto parametru má velmi podobný průběh.

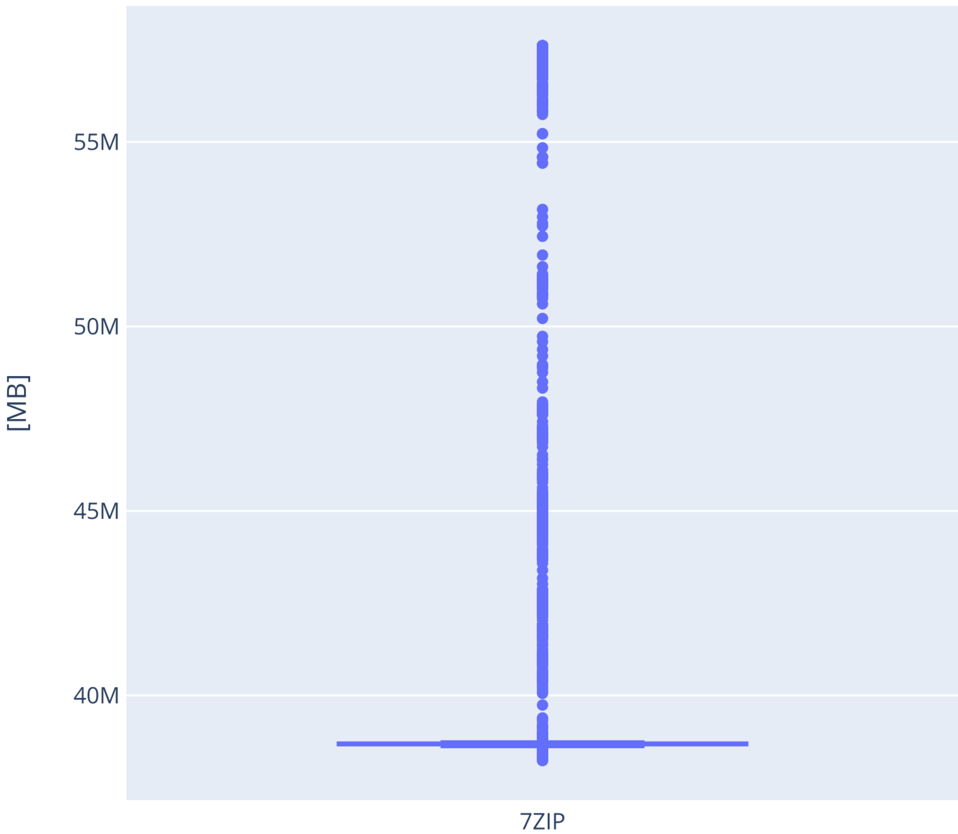
Při sledování parametru, který nastavuje počet vláken, vykazuje podobné průběhy při všech možnostech parametru. Oproti kompresi jednoho souboru se jako u minulých grafů snížila náročnost na paměť a nejsou žádné nízké odlehlé hodnoty.

Při sledování kombinace parametrů level a kompresních metod nebo filtrů jsou hodnoty velmi podobné až na snížení hodnoty mediánu a žádných odlehlých hodnot. Jediná změna je jiný průběh vyšších odlehlých hodnot.

Na grafu, který sleduje kombinaci parametrů level a vláken se především přidali vyšší odlehlé hodnoty a nejsou zde nižší odlehlé hodnoty. Hodnoty kolem mediánu se také snížili.

Na grafu sledující kombinaci parametrů kompresních metod nebo filtrů a vláken je největší změnou větší počet vyšších odlehlých hodnot a snížení hodnot kolem mediánu..

Při sledování všech parametrů jsou průběhy velmi podobné, jak už jsem psal už u grafů, největší změnou je nižší spotřeba paměti a větší odstupy mezi vyššími



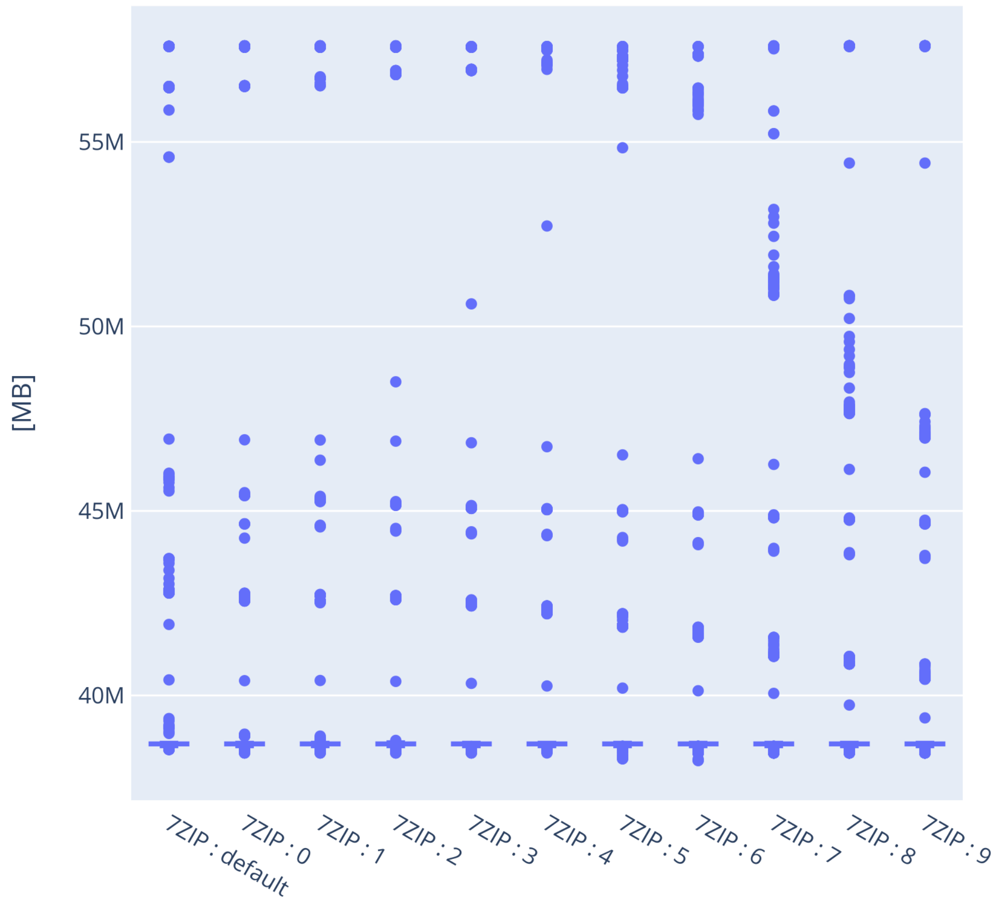
Obrázek 3.23: Graf zobrazující obecné vlastnosti kompresních algoritmů – více souborů, RAM

odlehlými hodnotami.

Délka běhu algoritmu

Graf 3.26 se všemi naměřenými hodnotami poukazuje, že čas komprese jako jediný se zvýšil oproti kompresi jednotlivých souborů i přes zachování podobné hodnoty mediánu.

Zvýšení času běhu je vidět i na grafu věnujícímu se podrobněji této charakteristice při nastavování parametru level. Potvrzují se však vlastnosti jednotlivých levelů, které jdou více vidět oproti grafu pro jednotlivé soubory. Nenastavení levelu je podobné vlastnostmi levelům pět, šest a osm. Platí především, že čím větší číslo

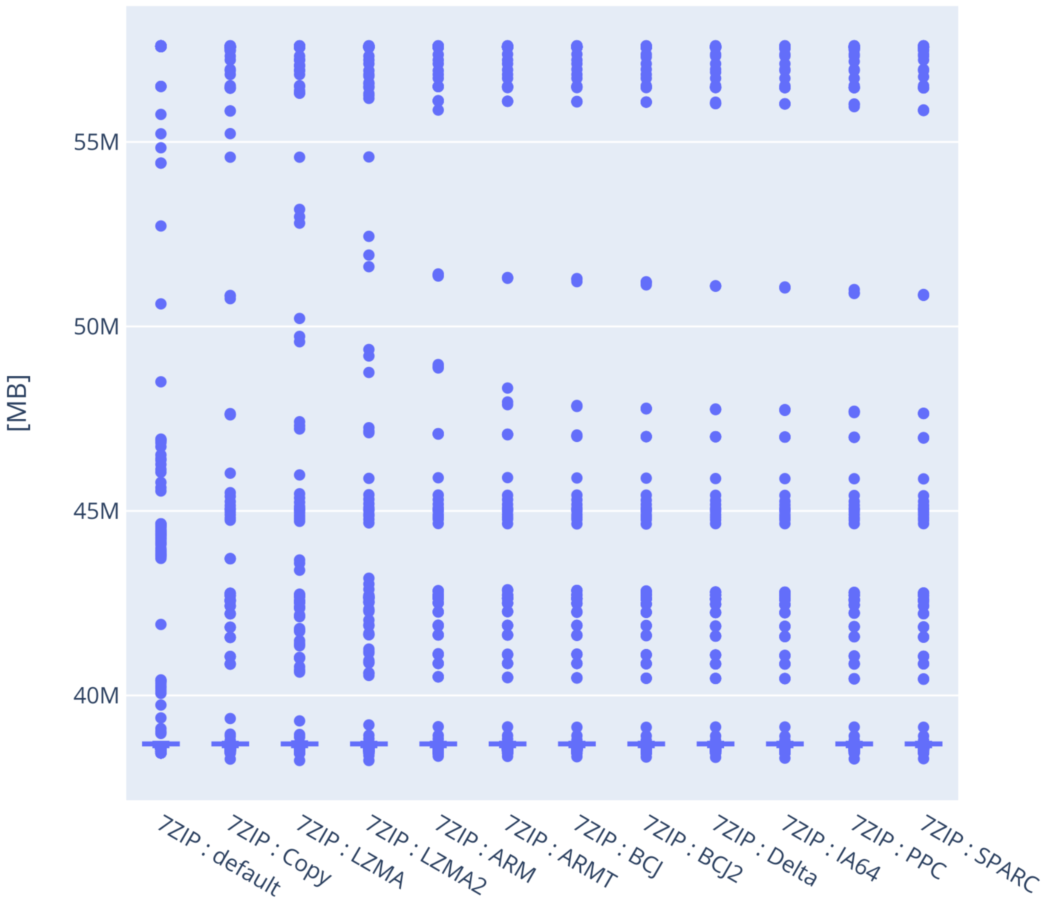


Obrázek 3.24: Graf zobrazující komprese podporující levely – více souborů, RAM

se zapíše do parametru level tím, déle komprese potrvá, protože vyšší komprese má větší nároky. Level nula je nejrychlejší, protože jen kopíruje data.

Na grafu 3.27 sledujícího délku komprese při nastavení parametru kompresních metod nebo filtrů, je vidět jako u kompresí jednotlivých souborů, kde se používá kompresní metoda a kde jen filtr. Základní komprese bez specifikace tohoto parametru a komprese pomocí kompresních metod LZMA a LZMA2 trvala déle. Jde zde i více vidět na první pohled, že kompresní metoda LZMA je méně optimalizovaná, díky starší architektuře, protože nejnižší hodnoty jsou při kompresi jednoho i více souborů vyšší. Kompresní filtry jsou rychlé a trvají oproti kompresním metodám zanedbatelný čas.

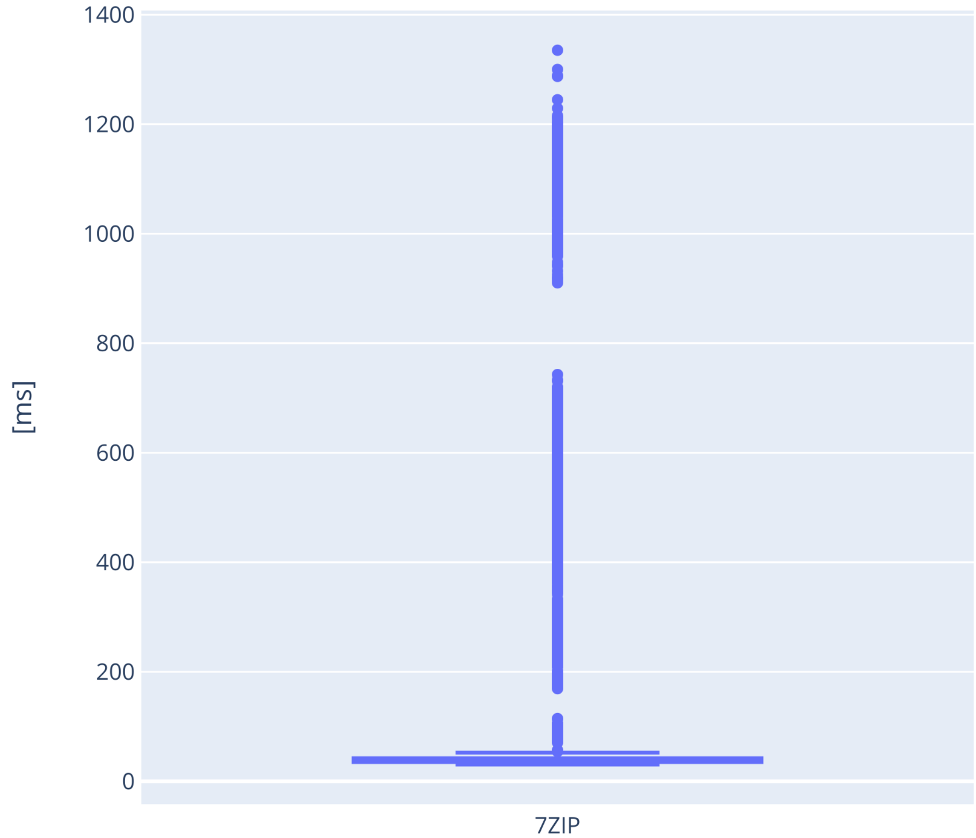
I při specifikacích parametru nastavující počet vláken se nezměnili markantně



Obrázek 3.25: Graf zobrazující komprese podporující kompresní metody a filtry – více souborů, RAM

délky běhu komprese pro více souborů. Všechny hodnoty jsou podobné mezi sebou a grafy kompresí jednoho i více souborů jsou si také velmi podobné, jen se zvýšila délka běhu algoritmu.

Při spojení parametrů level a kompresních metod nebo filtrů lze zpozorovat, jak jednotlivé kompresní metody, filtry nebo levely mění chování algoritmu. I na tomto grafu jako při kompresi jednotlivých souborů jsou průběhy velmi podobné. Kompresní filtry jako na předminulém grafu neběží moc dlouho. Nedefinování žádného parametru nebo jen druhého parametru a definice levelu vyššího, než čtyři způsobí velký nárůst délky běhu algoritmu. Přesně stejně se chovají kompresní metody LZMA a LZMA2, ale v tomto případě se dá lehce korigovat délka běhu pomocí

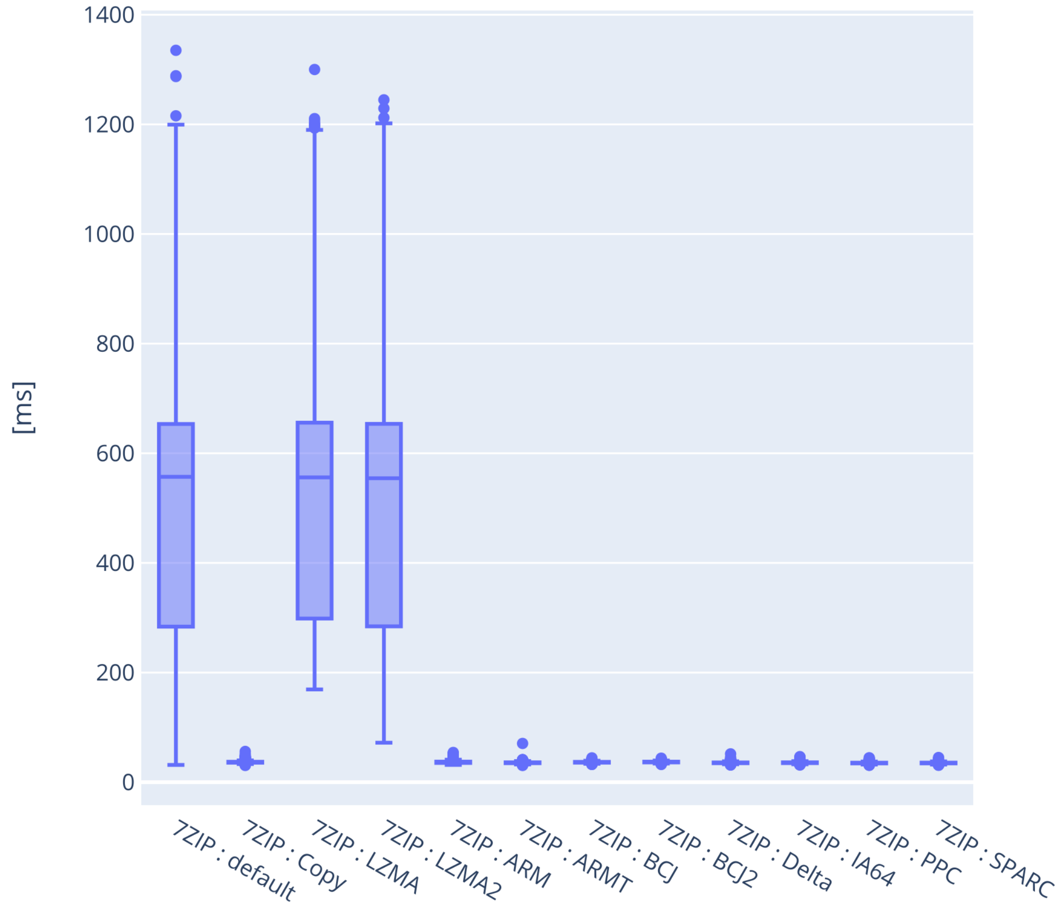


Obrázek 3.26: Graf zobrazující obecné vlastnosti kompresních algoritmů – více souborů, Časy běhu

nastavení parametru level. Především při kompresích do hodnoty čtyři lze lehce zpozorovat skokové narůstání délky běhu.

Délka běhu algoritmu i v případě kombinace parametrů level a vláken je velmi podobná kompresi jednotlivých souborů a zvýšení času běhu jse také vidět. Lze také zpozorovat, že délka běhu levelu nula je zanedbatelná a také při použití více vláken u levelu jedna. U komprese pomocí levelu jedna neměly vlákna takový vliv, ale při komprimaci více souboru jdou vidět vyšší hodnoty. Ve zbytku grafu je graf velmi podobný průběhem grafu jen s parametrem level.

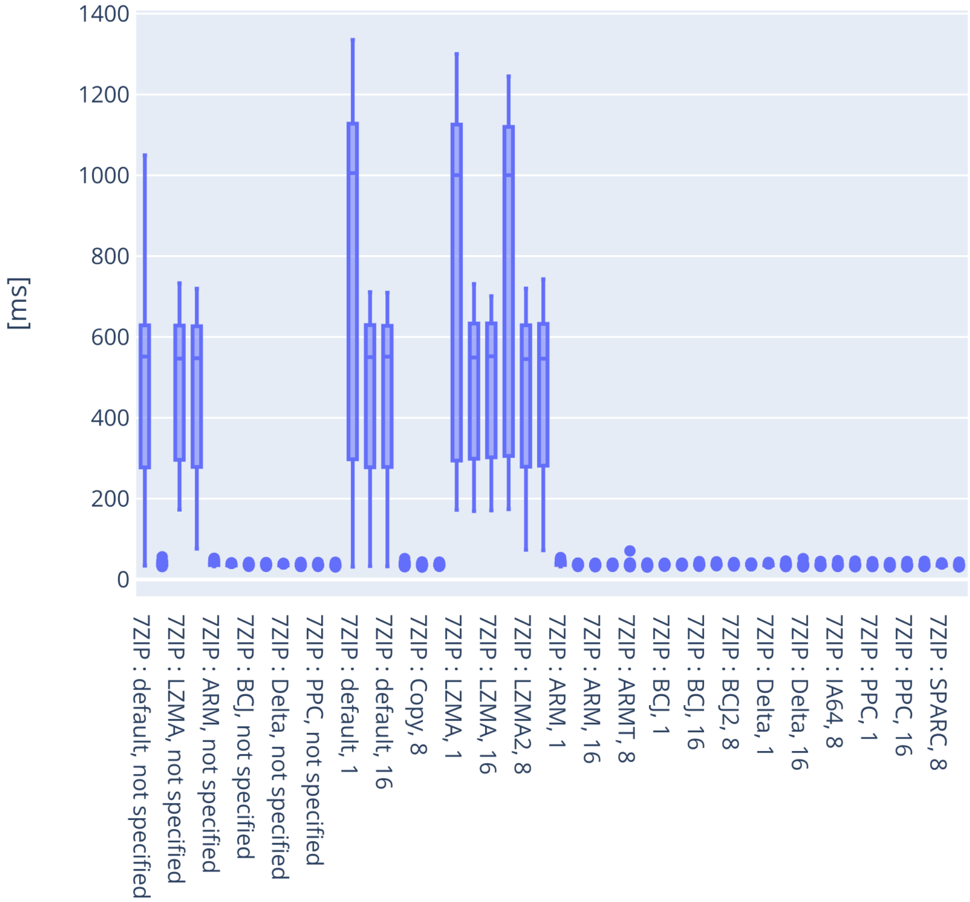
Graf 3.28 kombinující parametry kompresních metod nebo filtrů a vlákna. Výsledky jsou velmi podoné hlavně při nastavení jednoho vlákna a u nedefinování prvního pa-



Obrázek 3.27: Graf zobrazující komprese podporující kompresní metody a filtry – více souborů, Časy běhu

rametru nebo definovaní kompresních metod LZMA a LZMA2, protože délky běhu také vyrostou, ale při nastavení vláken rovných osmi nebo šestnácti délky běhu se ustálí a v některých případech u kompresí více souborů se i sníží. Také nedefinování prvního parametru nebo definování kompresních metod způsobí nárůst délky běhu algoritmu oproti filtrům a kompresní metodě Copy.

Na předešlých grafech jdou především vidět nízké délky běhu filtrů a větší délky běhu kompresních metod LZMA a LZMA2. Lze také zpozorovat, že nastavení vyšších počtů vláken sníží délku běhu komprese při specifikování kompresních metod LZMA a LZMA2 nebo při nespecifikování tohoto parametru. Lze také pozorovat skokový nárůst běhu při specifikacích levelů nižších než pět a při specifikaci kompresních



Obrázek 3.28: Graf zobrazující komprese s parametry kompresní metody, filtry a vlákny – více souborů, Časy běhu

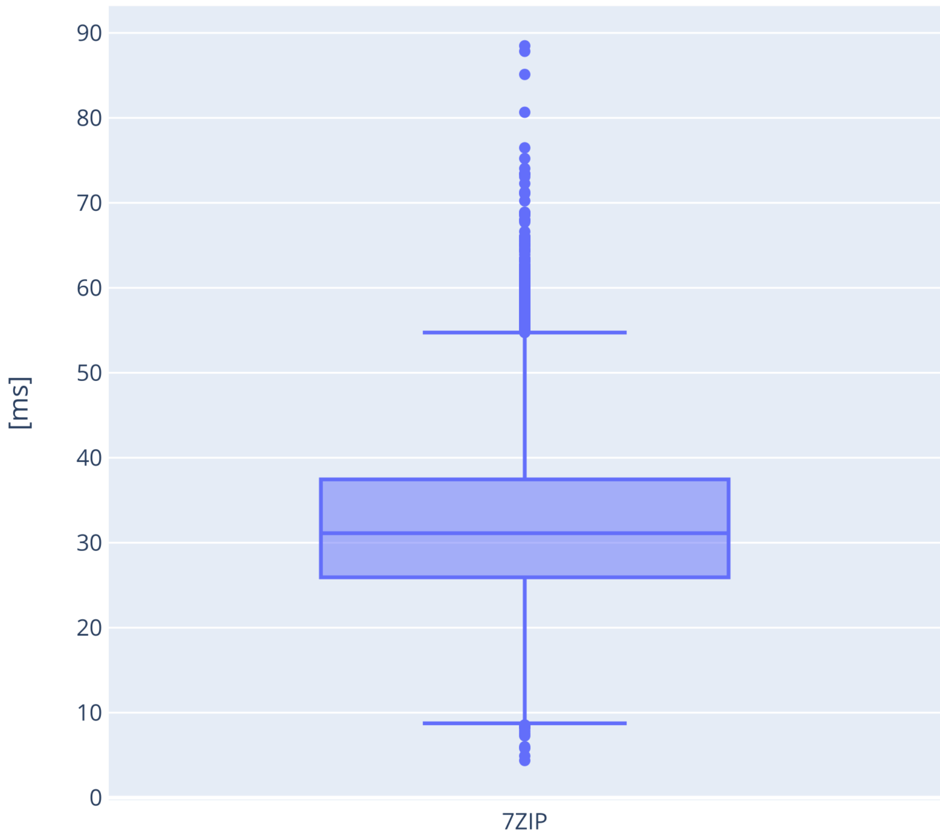
metod LZMA a LZMA2 a u levelů vyšších než čtyři, tento nárůst není až tak dramatický.

3.2.2 Dekomprese

Spotřeba procesorového času

Celkový průběh dekomprese vypadá velmi podobně a není vidět moc velký rozdíl oproti dekompresím jednotlivých souborů (viz. 3.29).

Na grafu se zkoumaným parametrem level nejlépe probíhají dekomprese u levelů 0, 1, 3 až 6 a devět. Oproti dekompresím jednotlivým souborům jsou zde k vidění

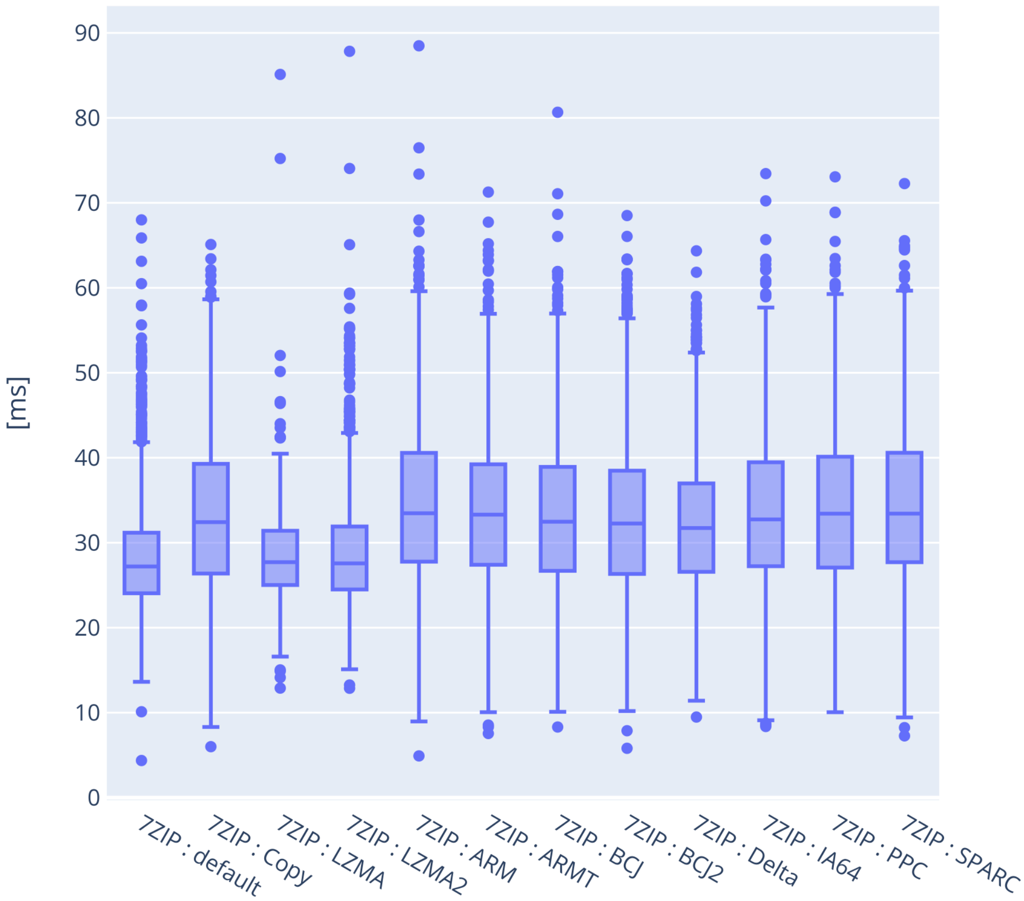


Obrázek 3.29: Graf zobrazující obecné vlastnosti kompresních algoritmů při dekompresi – více souborů, CPU

méně nižší odlehlé hodnoty, ale vyšší odlehlé hodnoty zůstali přibližně ve stejném rozsahu.

Při zaměření pozornosti na parametr kompresních metod a filtrů (viz. 3.30), lze zpozorovat, že ve většině případů dopadají nejlépe možnosti s nespecifikováním tohoto parametru a při specifikaci kompresních metod LZMA a LZMA2, ale v některých případech si vede lépe kompresní metoda Copy a všechny filtry, přičemž filtry Delta a PPC lehce zaostávají. Lze nadále zpozorovat úbytek nižších odlehlých hodnot jako na minulém grafu.

U parametru specifikujícího počet vláken je průběh jednotlivých vláken v rámci svých sloupců velmi podobný.

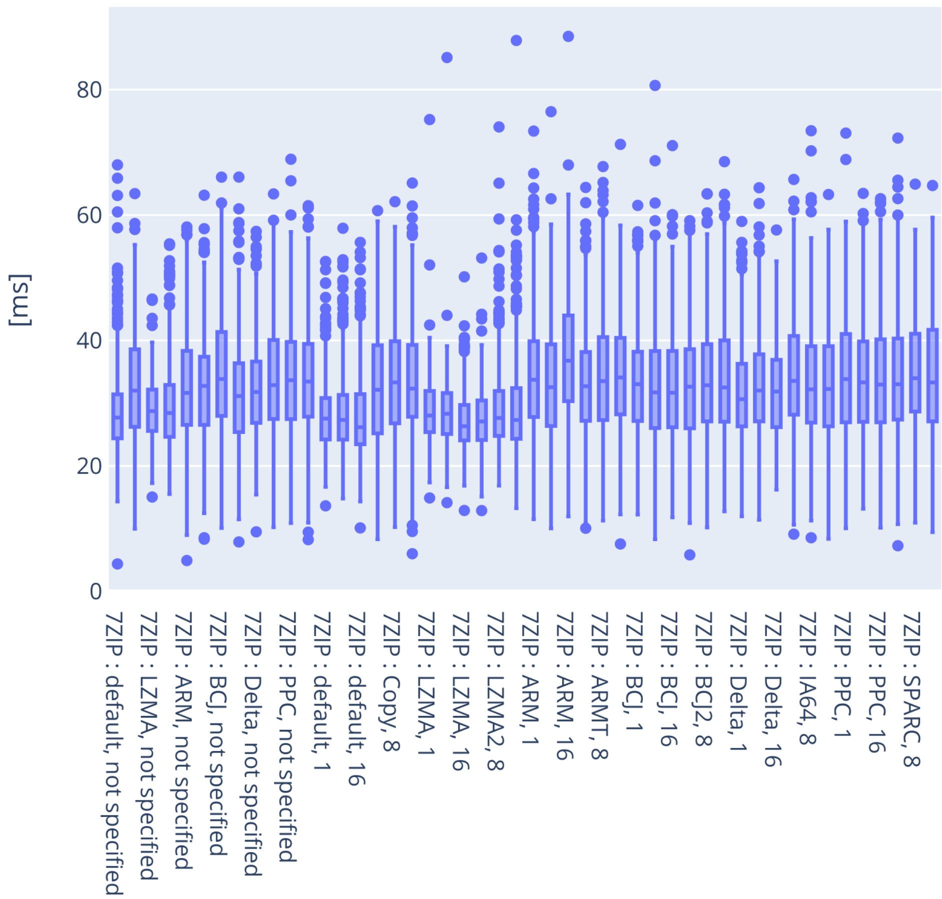


Obrázek 3.30: Graf zobrazující komprese podporující kompresní metody a filtry při dekompresi – více souborů, CPU

Na grafu, který zobrazuje průběhy dekompresí při specifikaci parametrů level a kompresních metod nebo filtrů, lze především zpozorovat nižší náročnost na procesor oproti konkunrečním nastvením u levelů 2 až 9 v kombinaci bez druhého parametru. Při specifikace kompresních metod LZMA a LZMA2 jsou výsledky lehce nižší, ale rozdíl není tak velký jako u samostatných souborů. Protože zmizeli nižší odlehlé hodnoty, které se velmi blížili nule, tak u většiny průběhů dekompresí se více vyhladil průběh na grafu, ale u některých dekompresí se přidali vyšší odlehlé hodnoty – především u levelů nula a jedna.

Při zobrazení spotřeby procesorového při závislosti na parametru nastavující počet vláken zjistíme, že průběhy spotřeby jsou u jednotlivých běhu komprese velmi podobné jako u dekomprese jednotlivých souborů. Jen není takové množství odlehlých hodnot oproti dekompresi jednotlivých souborů.

I při nastavení kombinace parametrů kompresních metod nebo filtrů a vláken nemají vlákna na dekompresi velký vliv a je prakticky jedno, jestli budou nastavené nebo nebudou. Nadále lepší výsledky při dekompresi vykazuje nenastavení prvního parametru nebo nastavení kompresních metod LZMA a LZMA2 (viz. 3.31).

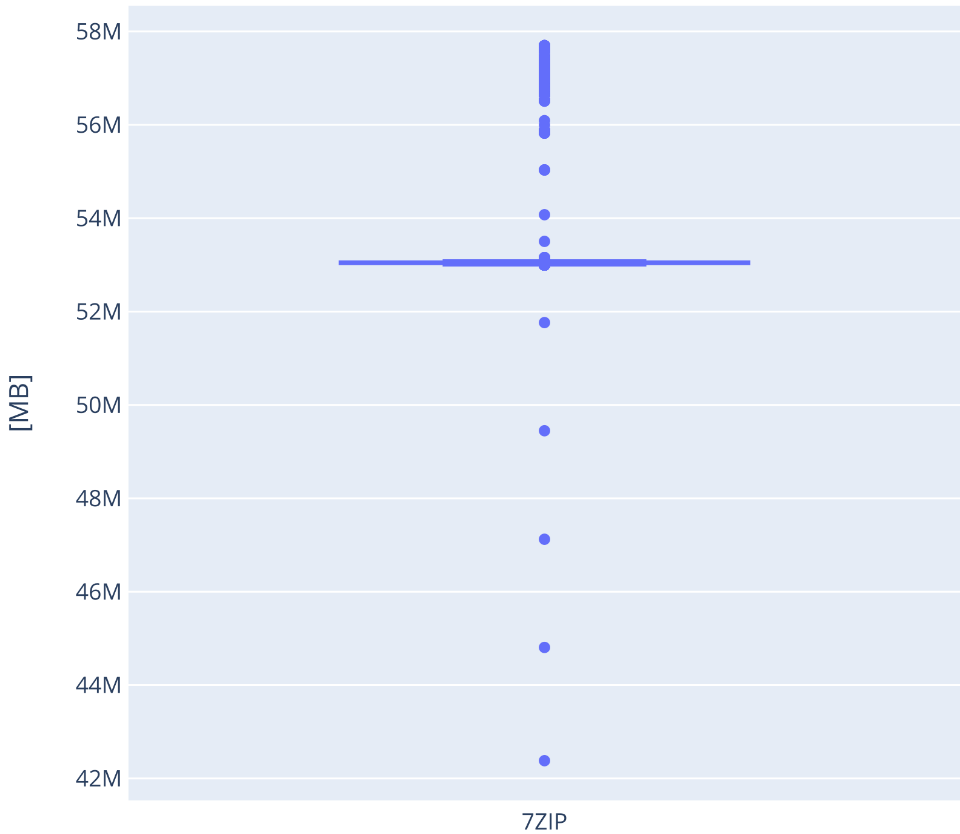


Obrázek 3.31: Graf zobrazující komprese s parametry kompresní metody, filtry a vlákny při dekompresi – více souborů, CPU

Nejlepší výsledky vykazují dekomprese bez specifikace parametru kompresních metod nebo filtrů a při specifikaci kompresních metod LZMA a LZMA2. Naměřené výsledky filtrů se vyskytují kolem podobné hodnoty a tím ve většině případů je jedno jaký uživatel vybere.

Spotřeba paměti RAM

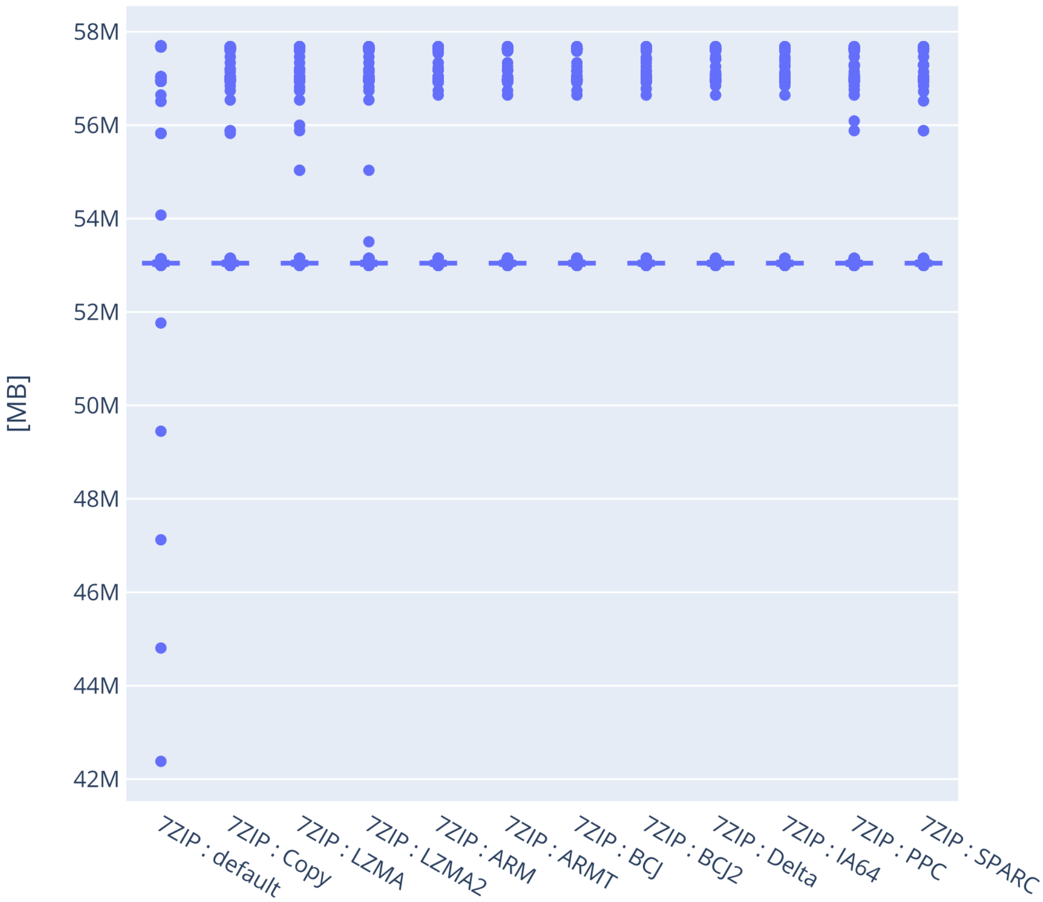
Na grafu 3.32 se všemi hodnotami, které byly získány během dekomprese, si lze všimnout, že se hodnoty oproti dekompresi jednotlivých souborů zvýšily a nižší odlehlé hodnoty začínají přibližně na hodnotách, které na některých končí vyšší odlehlé hodnoty dekomprese jednotlivých souborů.



Obrázek 3.32: Graf zobrazující obecné vlastnosti kompresních algoritmů při dekompresi – více souborů, RAM

Na grafu, zobrazující data získaná během specifikace parametru level, se objevuje více vyšších odlehlých hodnot oproti dekompresi jednotlivých souborů a u hodnot menších než 3 a při nespecifikování parametru se vyskytuje pro každý jeden sloupec jedna odlehlá hodnota, což je míň než u jednotlivých souborů. Přesto spotřeba paměti RAM je u všech možností stejná.

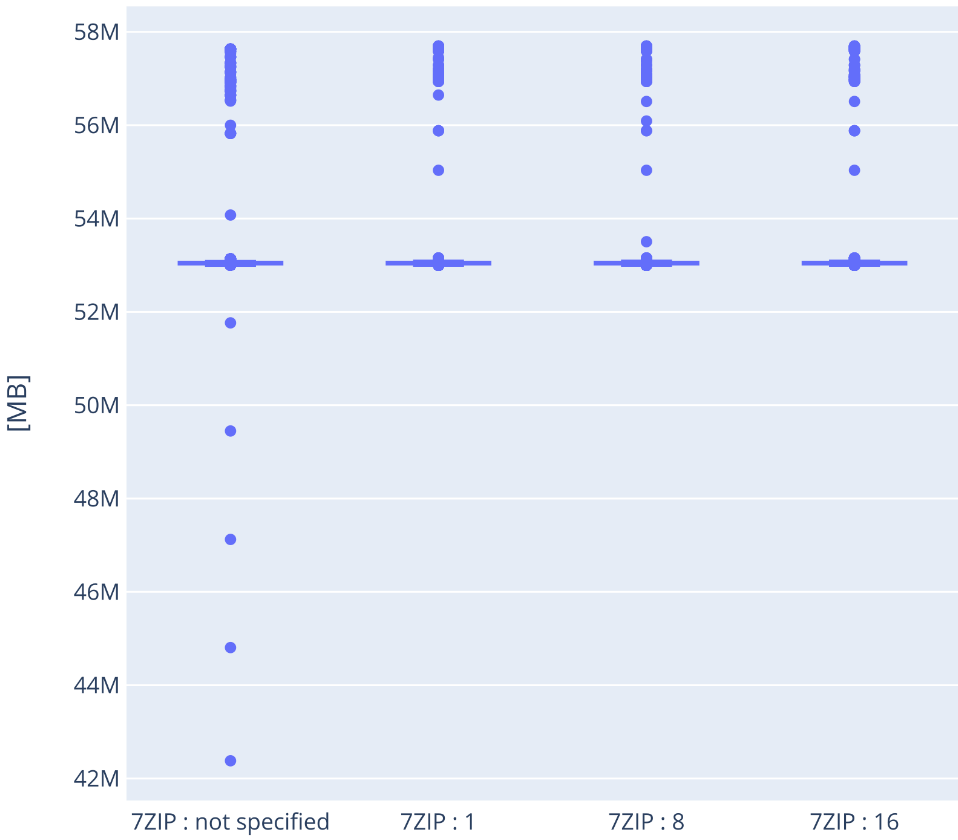
Na grafu 3.33 s parametrem kompresních metod a filtrů je také vidět vyrovnanost jednotlivých kompresních metod a filtrů. Přičemž je vidět, jaký parametr je odpovědný za nižší odlehlé hodnoty, které se objevili při nespecifikování nižších kompresních metod a to nespecifikace tohoto parametru.



Obrázek 3.33: Graf zobrazující komprese podporující kompresní metody a filtry při dekompresi – více souborů, RAM

I u dekomprese komprimovaných více souborů je vidět (viz. 3.34), že vlákna nemají žádný vliv při dekompresi na spotřebu paměti RAM, protože všechny hodnoty jsou přibližně na stejných místech. Jsou zde také vidět odlehlé hodnoty od nespecifikování parametru kompresních metod a filtrů.

Na grafu, který kombinuje parametry level a kompresní metody nebo filtry, má stejnou informaci, jakou ukazují i grafy samostatných parametrů. Především jsou



Obrázek 3.34: Graf zobrazující komprese podporující vlákna při dekompresi – více souborů, RAM

to všechny hodnoty kolem mediánu na stejné pozici a jsou zde vidět i odlehlé hodnoty při nespecifikování druhého parametru, které se ale následně promítají i do vyšších odlehlých u vyšších levelů. Menší nárůst vyšších odlehlých hodnot je i vidět v polovině grafu.

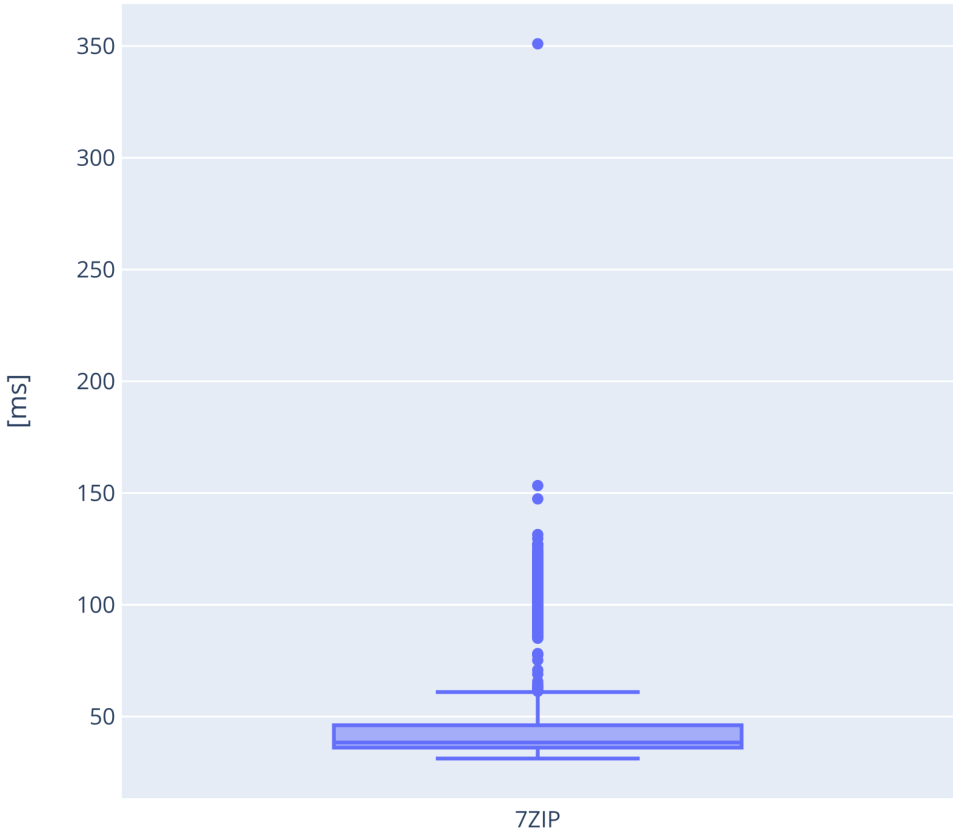
Na grafu kombinující parametry level a vlákna je vidět velmi podobný průběh z minulého grafu, ale žádnou novou informaci nepřidává.

Na grafu zobrazující kombinaci parametrů kompresních metod nebo filtrů a vláken jsou vidět předem známé informace, že nižší odlehlé hodnoty způsobuje nespecifikování prvního parametru, ale máme zde novou informaci. Tyto nižší odlehlé hodnoty způsobuje tento parametr v kombinaci s nespecifikovanými vlákny.

Na předešlých grafech jsou především vidět vyšší a nižší odlehlé hodnoty, které jsou způsobené nespecifikováním kompresních metod nebo filtrů v kombinaci s nespecifikováním vláken, také jsou vidět vyšší odlehlé hodnoty u některých kompresních metod i filtrů.

Délka běhu algoritmu

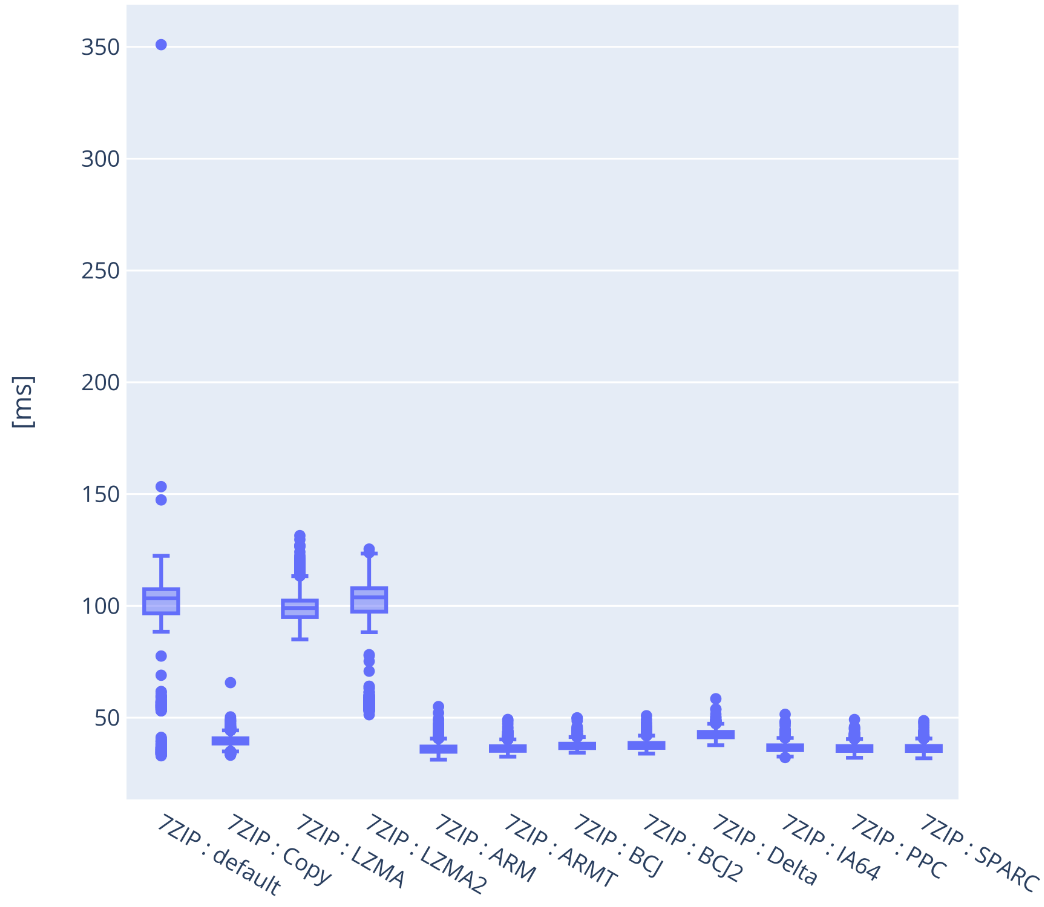
Na grafu 3.35 celkových výsledků délky běhů dekomprese je vidět zvýšení délky běhu při dekompresi více souborů. Odhadem se zvýšili délky běhu o 11 až 12 milisekund.



Obrázek 3.35: Graf zobrazující obecné vlastnosti kompresních algoritmů při dekompresi – více souborů, Časy běhu

Na grafu, zobrazující výsledky parametru level, lze zpozorovat, že všechny mediány jsou přibližně u stejné hodnoty. Přičemž hodnoty prvního a třetího kvartilu se při zvyšování hodnoty level postupně oddalují.

Zvýšení délky běhu je také vidět na grafu 3.36 zabývajícího se kompresními metodami a filtry. Také lze zpozorovat, že dekomprese komprimovaných souborů pomocí kompresních metod LZMA nebo LZMA2 a při nespecifikování tohoto parametru se navýšila více délka běhu dekomprese než u použitých filtrů.



Obrázek 3.36: Graf zobrazující komprese podporující kompresní metody a filtry při dekompresi – více souborů, Časy běhu

Na grafu, zobrazující hodnoty parametru vláken, jde zpozorovat velmi podobný průběh mezi specifikacemi tohoto parametru, ale potvrzuje odhad nárustu délky běhů dekomprese o 12 milisekund.

Při zkoumaní kombinace parametrů level a kompresních metod nebo filtrů trvá nejkratší dobu dekomprimovat komprese filtrů a kompresní metodu Copy, přičemž je jedno jaký level byl zvolen. Nejdéle trvala dekomprese komprimace pomocí levelu nula a algoritmu LZMA. Přičemž při nastavení kompresních metod LZMA a LZMA2 nebo při nespecifikování tohoto parametru trvala dekomprese vždy déle. V průměru se pohybuje dekomprese při levelu nula pro kompresní metodu LZMA2 a nespecifikovaný tento parametr a také pro level roven jedničce a kompresní metodu LZMA2.

Graf s kombinací parametrů level a vláken nabízí velmi vyrovnaný průběh přes celý graf, ale dekompresi u levelu nula nebo u levelu jedna v kombinaci s osmi a šestnácti vlákny se sníží hodnota třetího kvartilu a maximální hodnoty boxplotu.

U specifikace parametrů kompresních metod nebo filtrů a vláken lze také zpozorovat (viz. 3.37), že nejrychlejší komprese vykazují kompresní filtry a metoda Copy. Nejdéle trvá dekomprese u nespecifikování ani jednoho parametru a při nespecifikování prvního parametru u všech vláknech nebo při specifikaci kompresních metod LZMA a LZMA2. U poslední možnosti je jedno, jestli se specifikuje počet vláken nebo nespecifikuje.

Nejkratší dobu trvá dekomprese filtrů a kompresní metody Copy. V průměru se pohybují dekomprese při nespecifikování kompresních metod nebo při specifikaci kompresní metody LZMA2 a levelů nula a jedna. Nejdéle trvá dekomprimovat nespecifikované kompresní metody nebo filtry a kompresní metody LZMA a LZMA2 u levelu většího než jedna. Přičemž je jedno, jaká hodnota počtu vláken je specifikována.

3.2.3 Kompresní poměry

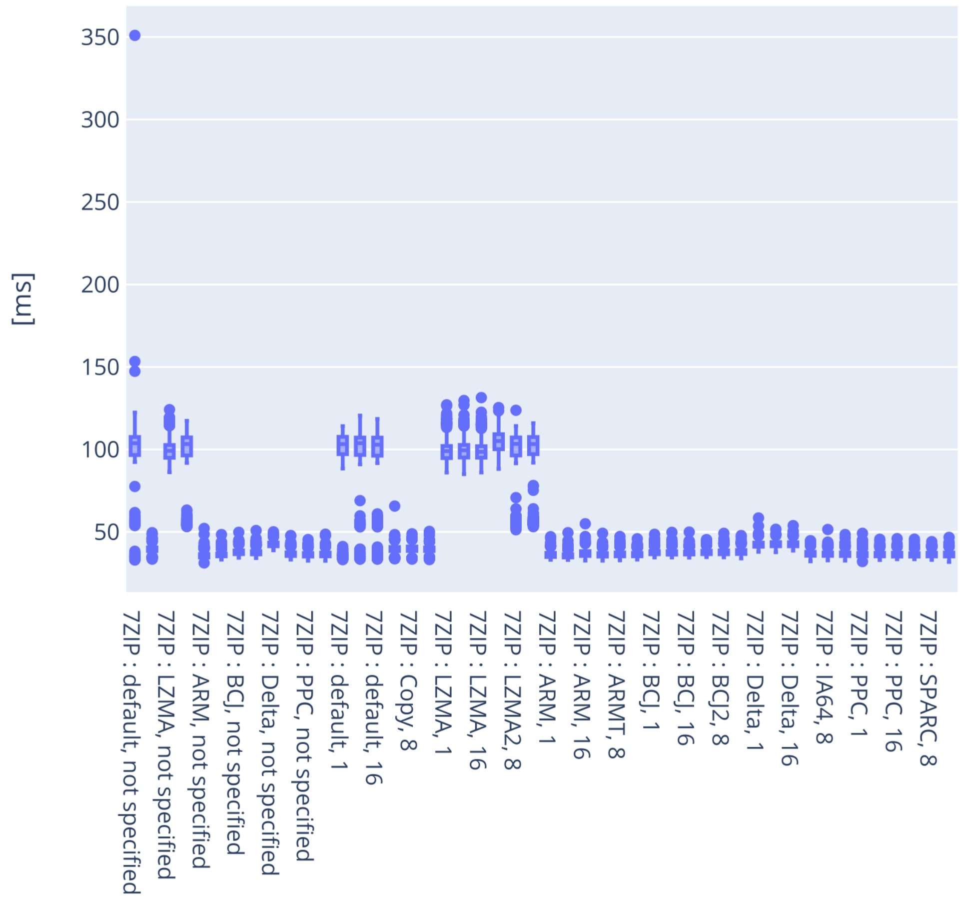
Celkové výsledky kompresních poměrů při kompresi více souborů najednou se nezměnili oproti kompresním poměrů jednotlivých souborů (viz. 3.38). Nadále platí, že nejnižší naměřená hodnota je přibližně 48 % původního souboru a nejvyšší 100 %. Sto procent je také nadále nejčastější hodnota v testu.

Kompresní poměry při zobrazení parametru level vychází nadále nejlépe při zaměření na nižší komprese bez definice tohoto parametru a pro levely vyšší, než je hodnota čtyři (viz. 3.39). Přičemž hodnota levelu rovna devíti vychází z testu nejlépe při pohledu na první kvartil boxplotu. Tento údaj se jediný změnil oproti kompresním jednotlivým souborům. Nejhůře vychází jako u kompresí jednotlivých souborů metoda Copy.

Při změně zkoumaného parametru na kompresní metody a filtry, vychází nejlépe komprese bez definice parametru nebo při specifikaci kompresních metod LZMA a LZMA2 (viz. 3.40). U nespecifikovaného parametru je odlehlá hodnota na 100 % (tato hodnota je určitě způsobena levelem rovným nule). Nejhorší výsledky v testu byly naměřeny u kompresní metody Copy a filtrů, které mají výsledné soubory velké jako vstupní.

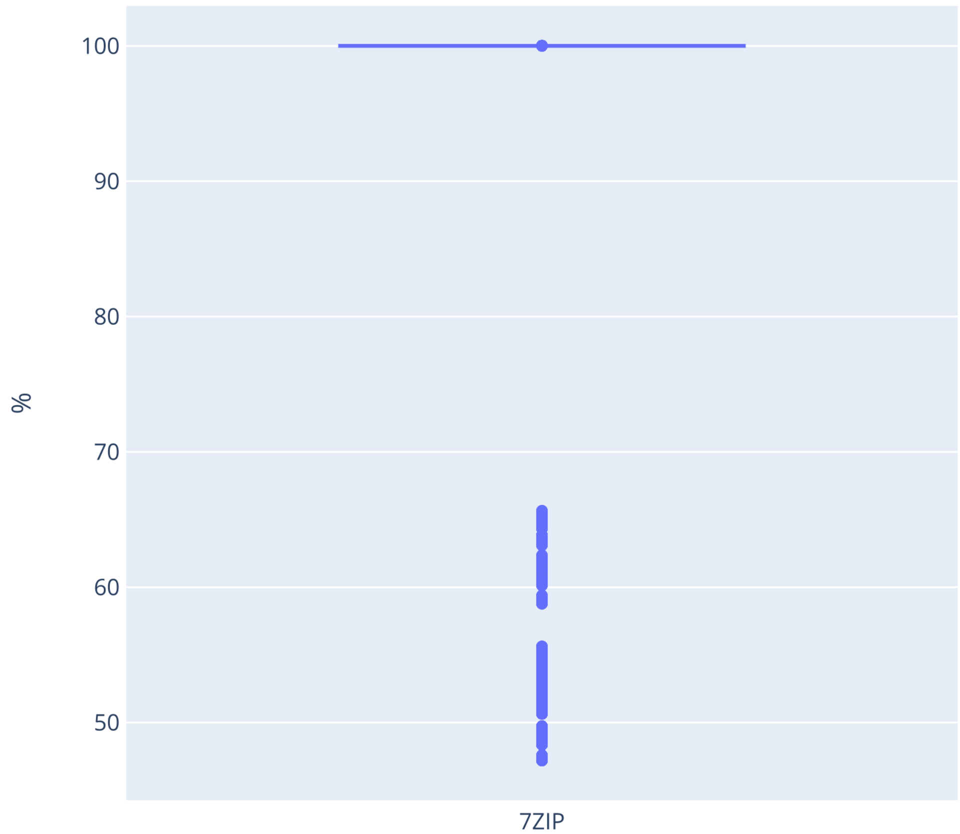
Nastavení parametru nastavující počet vláken nemá logicky na kompresi žádný vliv. Všechny sloupce typu boxplot jsou si velmi podobné a nebál bych se říct že i stejné.

Na grafu, zobrazující kombinaci parametrů level a kompresních metod nebo filtrů



Obrázek 3.37: Graf zobrazující komprese s parametry kompresní metody, filtry a vlákny při dekompresi – více souborů, Časy běhu

lze zpozorovat, že průběh grafu je velmi podobný kompresím jednotlivých souborů a kompresní poměry se také nezměnili. Nejlepší komprese vychází u nedefinování levelu v kombinaci s nedefinovaním druhého parametru a specifikace kompresních metod LZMA a LZMA2. Také bych do nejlepších výsledků zařadil levely větší, než je 4 v kombinaci bez specifikace druhého parametru a specifikace kompresních metod LZMA a LZMA2. Přičemž kompresní metody LZMA a LZMA2 jsou lepší možností. V průměru použitelnosti kompresí se pohybují kompresní poměry při levelech 0 až 4 při nespecifikování druhého parametru a levely 1 až 4 při použití kompresních metod LZMA a LZMA2. Nejhorší použitelné komprese používají kombinaci levelu 0 a kompresních metod LZMA a LZMA2. Nepoužitelné komprese jsou v kombinaci



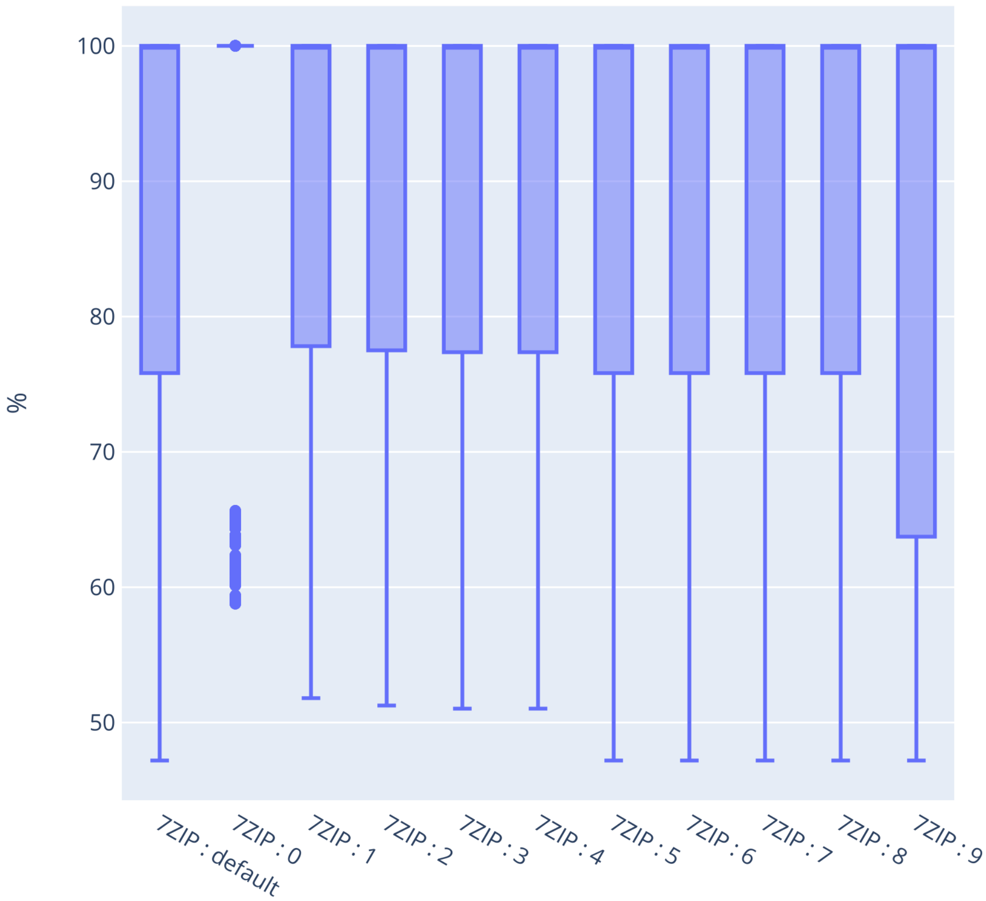
Obrázek 3.38: Graf zobrazující kompresní poměry - více souborů

s filtry a kompresní metodou Copy.

Graf s kombinací parametrů level a vláken má velmi podobný průběh jako graf s jediným parametrem levelem. Hlavním důvodem je, že změna počtu vláken, jak bylo zjištěno na grafu s vlákny, nemá žádný vliv na kompresní poměry. Proto výsledné boxploty působí jen jako by byly zkopírovány.

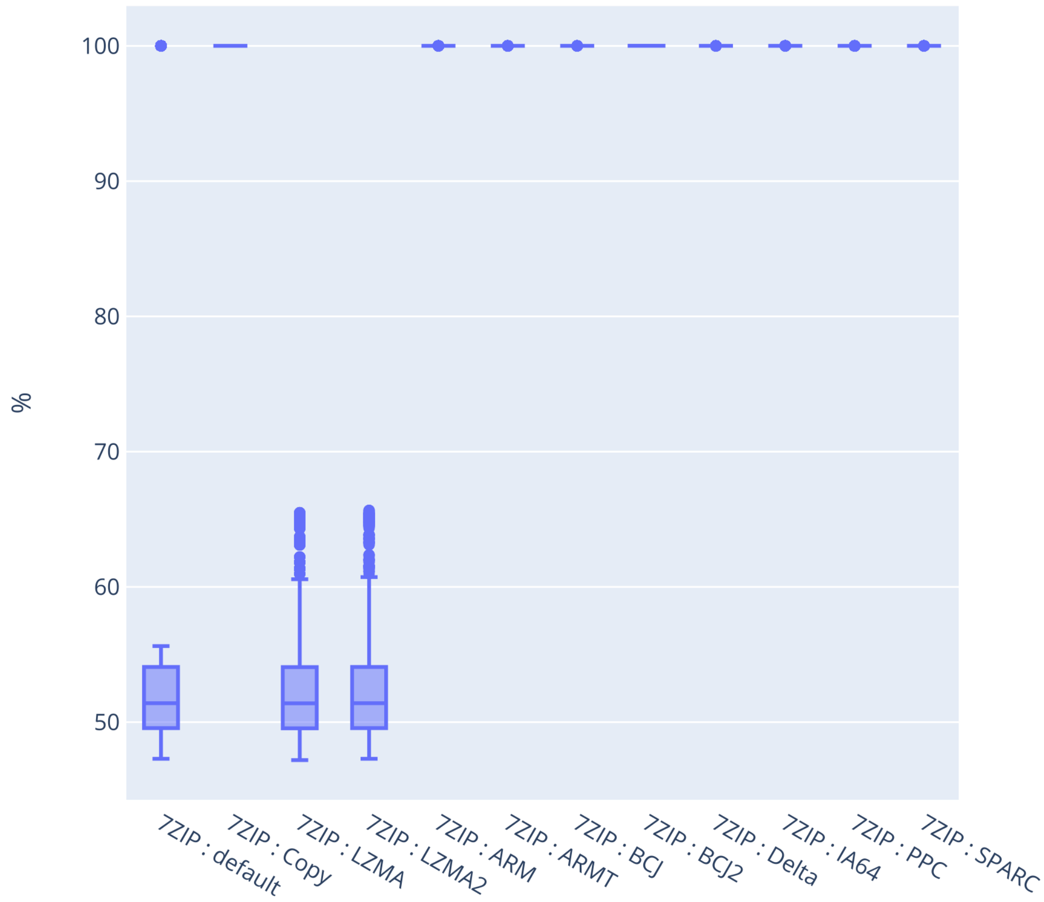
Graf s kombinací kompresních metod nebo filtrů a vláken má prakticky stejný průběh jako u kompresí jednotlivých souborů a prakticky stejnou informaci. Nejlepší jsou především kombinace nespecifikování prvního parametru a specifikace kompresní metod LZMA a LZMA2 v kombinaci se nespecifikováním druhého parametru i při specifikaci druhého parametru.

Na posledním grafu, který je věnován kompresním metodám je podrobný pohled



Obrázek 3.39: Graf zobrazující kompresní poměry s parametrem level - více souborů

na všechny parametry. Nejlepší výsledky jdou zpozorovat především u levelů větších, než je hodnota 5 nebo při nedefinování levelů a při kombinaci kompresních metod LZMA a LZMA2. O něco hůře, ale nadále kvalitně, komprimuje 7ZIP při nespecifikování kompresní metody ani filtru se stejnou kombinací u parametru level. V průměru použitelnosti kopresí jsou především kompresní metody LZMA a LZMA2 v kombinaci s levely 1 až 4 a nespecifikovanými kompresními metodami nebo filtry v kombinaci s levely 0 až 4. Nejhorší z použitelných kompresí jsou komprese levelem 0 v kombinaci s kompresními metodami LZMA a LZMA2 a bez kompresních metod ani filtrů. Nepoužitelné algoritmy jsou především ve spojení s kompresní metodou Copy a všemi filtry.



Obrázek 3.40: Graf zobrazující kompresní poměry s parametrem kompresní metody/kompresní filtry - více souborů

3.2.4 Shrnutí základních charakteristik

3.2.5 Optimální velikosti souborů

Algoritmus 7ZIP ve většině případů na všech typech dat je nepoužitelný (viz. 3.41). Tyto hodnoty, jak bylo zjištěno na minulých grafech, jsou způsobeny filtry a kompresními metodami Copy, ale při zaměření na nižší hodnoty, které jsou v tomto případě znázorněné jako odlehlé hodnoty. Zjistíme, že nejoptimálnější data z ze získaného archivu, jsou velké 1 536 066 bitů a na druhé nejlepší data, které byly použity jsou 1 377 346 bitů. Nejhorší data použitá v testu jsou o velikosti 1 321 026, 1 433 666 a 1 510 466 bitů.

Tabulka 3.1: Tabulka shrnující výsledky komprese

Algoritmus	**Počet vstupů**	**Charakteristika**	**MIN**	**MAX**
7ZIP	1	CPU [ms]	0	76,2
7ZIP	1	RAM [MB]	41,3M	56,6M
7ZIP	1	Čas běhu [ms]	28,3	353,9
7ZIP	1	Kompresní poměr [%]	47,4	100
7ZIP	3	CPU [ms]	0	66,1
7ZIP	3	RAM [MB]	38,2M	57,6M
7ZIP	3	Čas běhu [ms]	30,6	1335,1
7ZIP	3	Kompresní poměr [%]	47,2	100
BZIP2	1	CPU [ms]	2,9	27,9
BZIP2	1	RAM [MB]	53,2M	53,2M
BZIP2	1	Čas běhu [ms]	74,9	100,9
BZIP2	1	Kompresní poměr [%]	43,5	54,6
GZIP	1	CPU [ms]	2,9	41,7
GZIP	1	RAM [MB]	53,2M	53,2M
GZIP	1	Čas běhu [ms]	48,5	80,7
GZIP	1	Kompresní poměr [%]	69,3	77,6
LZW	1	CPU [ms]	7,9	20,6
LZW	1	RAM [MB]	53,2M	53,2M
LZW	1	Čas běhu [ms]	104,3	135,4
LZW	1	Kompresní poměr [%]	68,7	85
FLAC	1	CPU [ms]	0	51
FLAC	1	RAM [MB]	53,1M	53,2M
FLAC	1	Čas běhu [ms]	31	50,5
FLAC	1	Kompresní poměr [%]	40,9	47,2
TTA	1	CPU [ms]	1,1	34,3
TTA	1	RAM [MB]	53,1M	53,2M
TTA	1	Čas běhu [ms]	42,5	53,9
TTA	1	Kompresní poměr [%]	40,3	45,2

U algoritmu BZIP2 mají jednotlivé kompresní poměry lepší průběhy (viz. 3.42). Při zaměření na hodnoty mediánu vychází nejlépe data o velikosti 1 484 864 bitů následující velikostí 1 218 624 bitů a nejhorší výsledky vychází velikosti souborů 1 433 666 bitů následující soubory o velikosti 1 321 026 bitů, ale zaměření na nejnižší možné hodnoty vychází nejlépe velikosti souborů 1 536 066 bitů následující velikostí 1 377 346 bitů. Tyto poslední hodnoty vykazují při určitých hodnotách i nejhorší výsledky. Proto každý si musí vybrat jaké výsledky ho více zajímají, ale v tomto případě jsou více vypovídající spíše mediány.

Algoritmus GZIP vykazuje velmi podobné charakteristiky (viz. 3.43) jako algoritmus BZIP2 jen s jinými hodnotami. Přičemž nejlepší výsledky, když se zaměříme na mediány, vykazují soubory o velikosti 1 484 864 bitů následující velkým skokem velikostmi souborů s hodnotami 1 295 428 a 1 218 624 bitů. Nejvyšší mediány se

Tabulka 3.2: Tabulka shrnující výsledky dekomprese

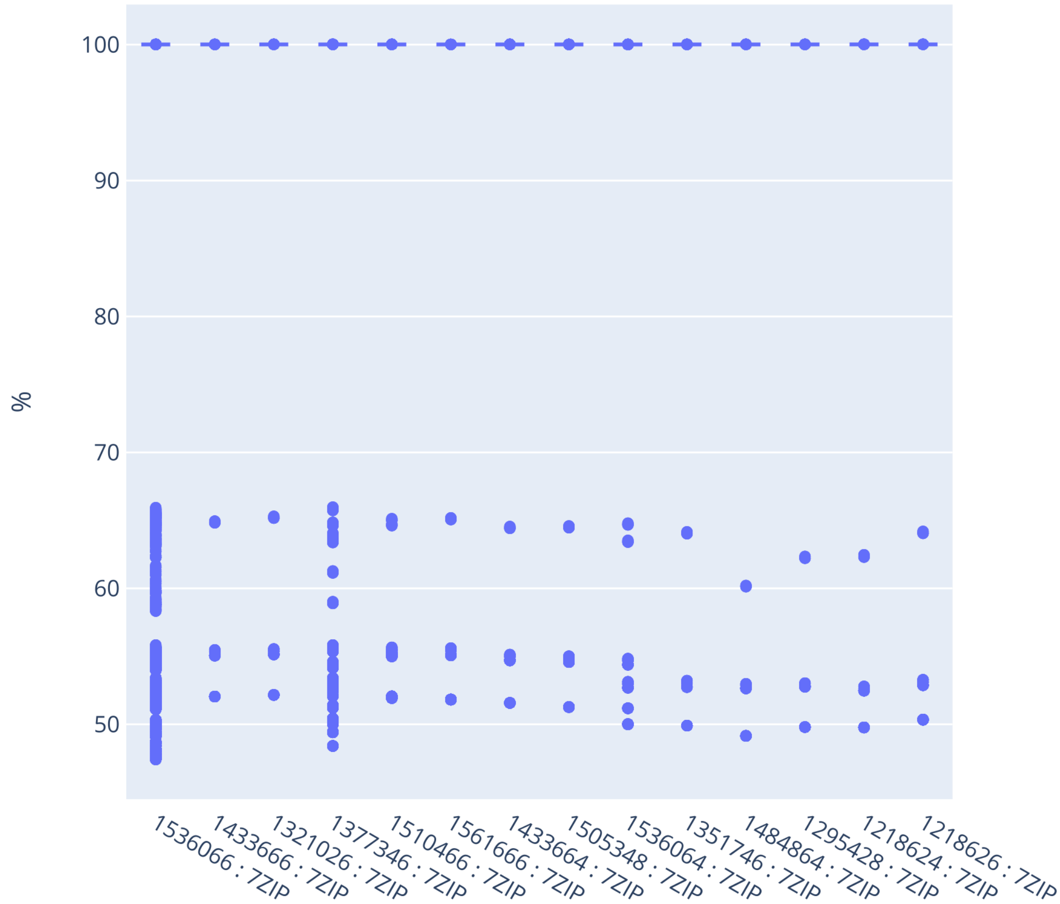
Algoritmus	Počet vstupů	Charakteristika	MIN	MAX
7ZIP	1	CPU [ms]	0	72,9
7ZIP	1	RAM [MB]	40,6M	41M
7ZIP	1	Čas běhu [ms]	28,3	86,9
7ZIP	3	CPU [ms]	4,4	88,5
7ZIP	3	RAM [MB]	42,4M	57,7M
7ZIP	3	Čas běhu [ms]	31,2	351
BZIP2	1	CPU [ms]	10,1	40,1
BZIP2	1	RAM [MB]	41M	41M
BZIP2	1	Čas běhu [ms]	49,3	66,2
GZIP	1	CPU [ms]	8,9	61,3
GZIP	1	RAM [MB]	41M	41,1M
GZIP	1	Čas běhu [ms]	31,8	41,8
LZW	1	CPU [ms]	16,6	31,2
LZW	1	RAM [MB]	41M	41,1M
LZW	1	Čas běhu [ms]	92,2	121,5
FLAC	1	CPU [ms]	7,2	58,5
FLAC	1	RAM [MB]	40,7M	40,8M
FLAC	1	Čas běhu [ms]	30,1	58,8
TTA	1	CPU [ms]	5,8	44,9
TTA	1	RAM [MB]	40,8M	40,8M
TTA	1	Čas běhu [ms]	44,1	53,3

vykytují u velikosti souboru 1 561 666 bitů následující 1 377 346 bitů. Zato nejlepší i nejhorší výsledky se vyskytují u velikostí souborů 1 536 066 bitů a 1 377 346 bitů.

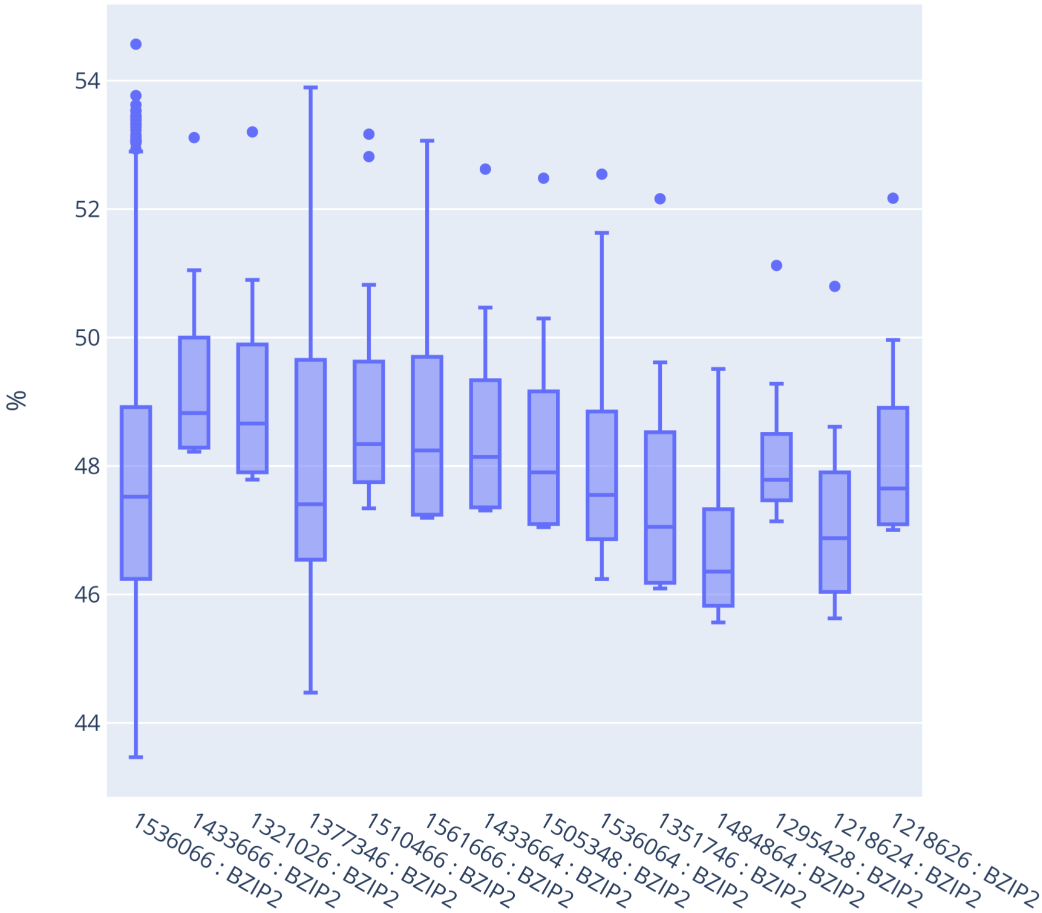
U algoritmu LZW podle mých měření je podle mediánu nejoptimálnější velikost 1 484 864 bitů (viz. 3.44) a nejhorší soubory s velikostí 1 321 026 bitů následující velikostmi 1 433 666 a 1 295 428 bitů. Ale při zaměření globální hodnoty vychází nejlépe velikost souboru 1 536 066 bitů.

Nejoptimálnější velikost u algoritmu FLAC při zaměření na mediány je především 1 484 864 bitů (viz. 3.45). Přičemž velmi blízko jsou i průběhy souborů o velikosti 1 351 746, 1 295 428, 1 218 624 a 1 218 626 bitů a nejhorší výsledky byly naměřeny u šesti velikostí souborů. Celkově nejhorší výsledky podle mediánů jsou u velikostí 1 321 026 a 1 510 466 bitů. Globálně nejlepší výsledky byly zjištěny u velikostí souborů 1 536 066 bitů a nejhorší 1 377 346 bitů.

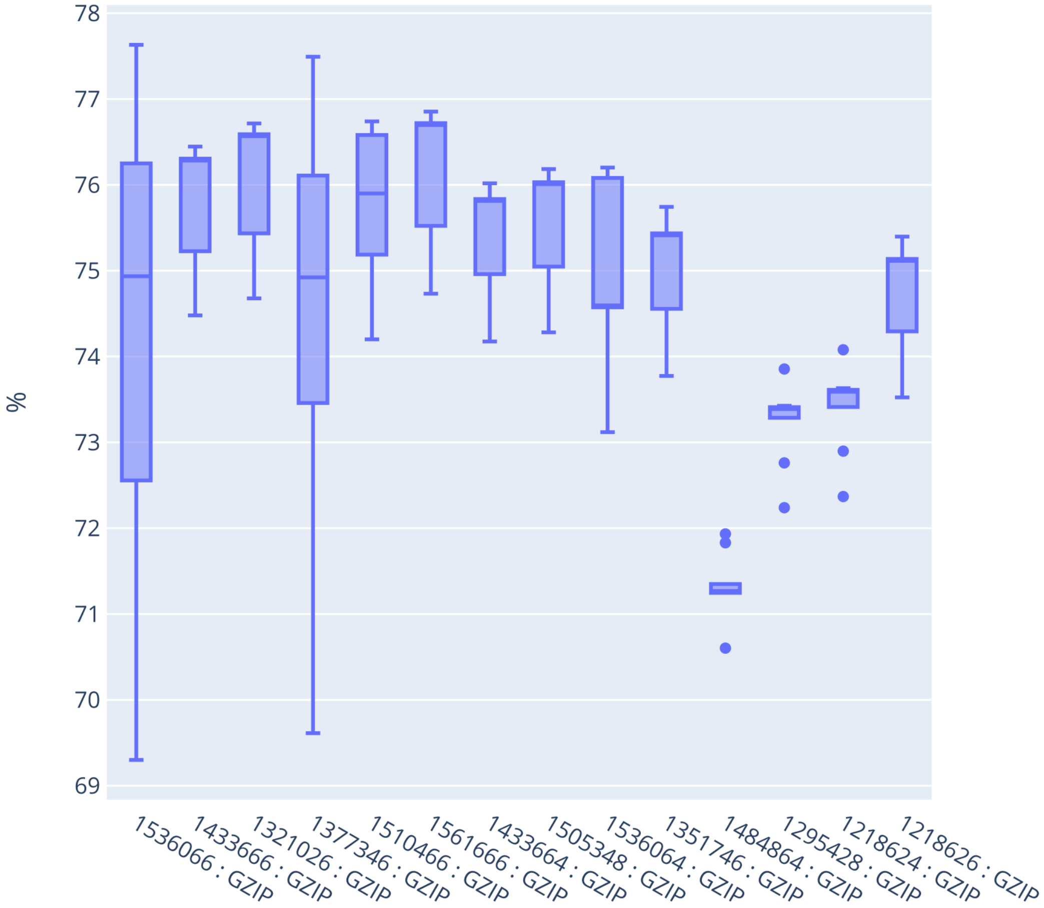
Algoritmus TTA vykazuje horší výsledky při zaměření na mediány spíše u větších souborů (viz. 3.46) v testu a nejlepší u menších souborů, ale při zaměření na globální výsledky vychází nejlépe i nejhůře velikosti souborů 1 536 066 a 1 321 026 bitů.



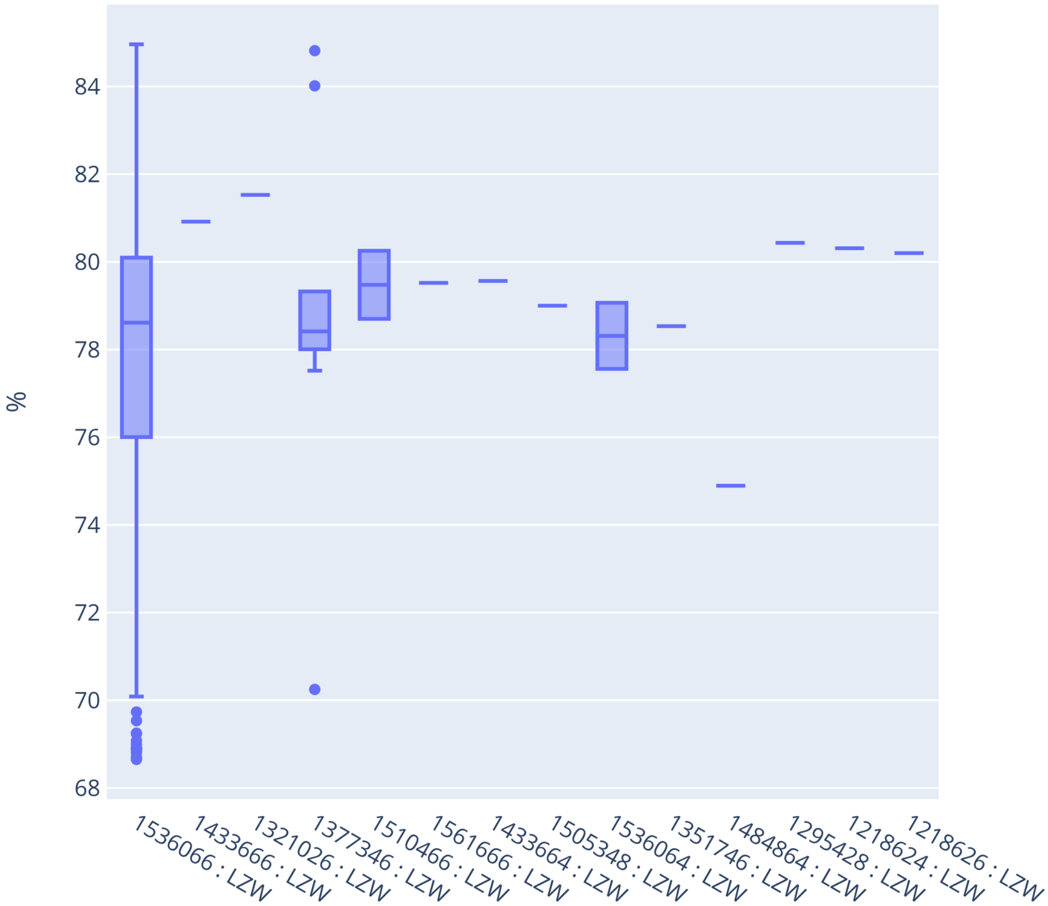
Obrázek 3.41: Graf zobrazující optimální velikosti vstupních souborů u algoritmu 7ZIP – kompresní poměry



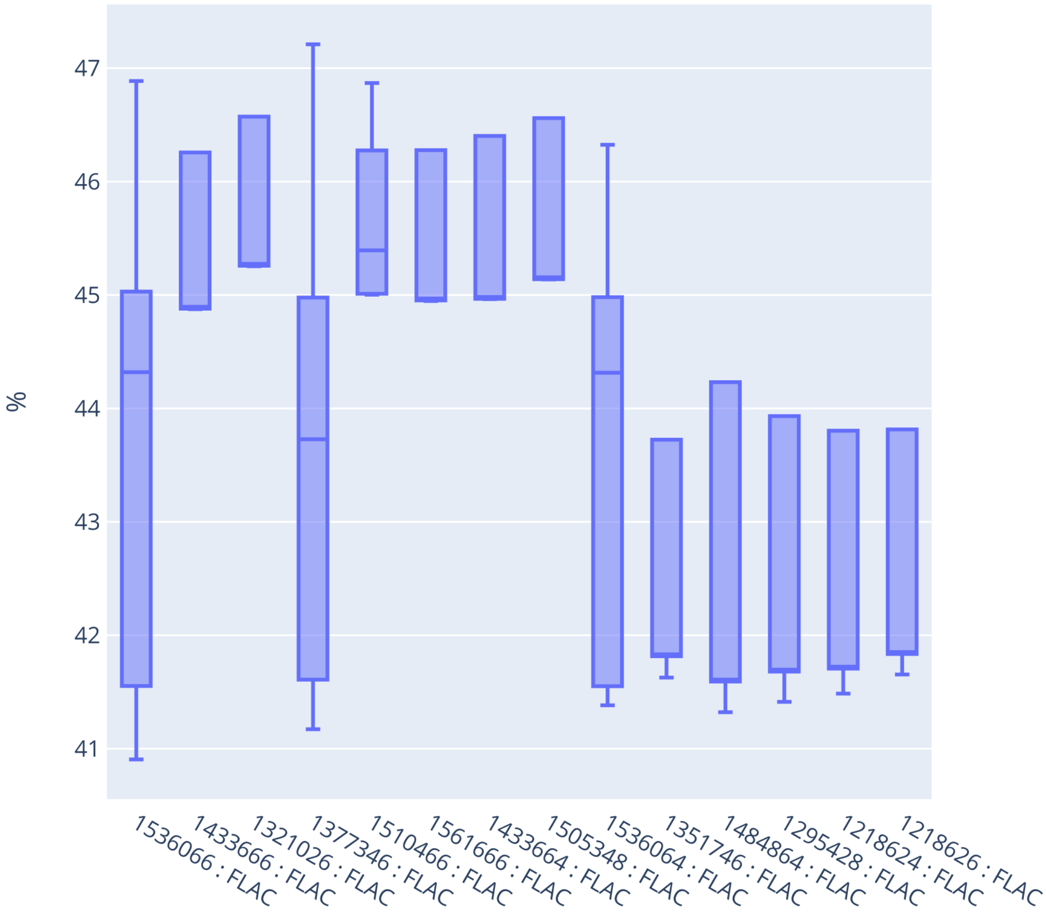
Obrázek 3.42: Graf zobrazující optimální velikosti vstupních souborů u algoritmu BZIP2 – kompresní poměry



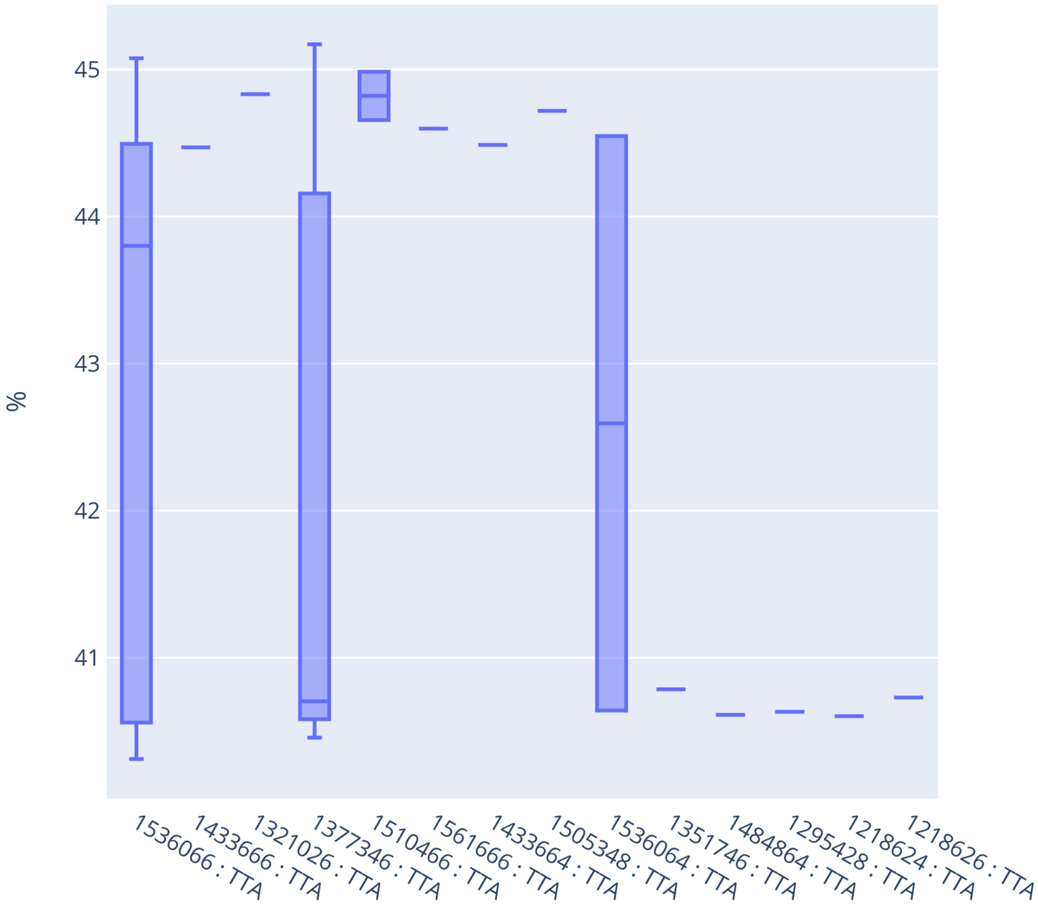
Obrázek 3.43: Graf zobrazující optimální velikosti vstupních souborů u algoritmu GZIP – kompresní poměry



Obrázek 3.44: Graf zobrazující optimální velikosti vstupních souborů u algoritmu LZW – kompresní poměry



Obrázek 3.45: Graf zobrazující optimální velikosti vstupních souborů u algoritmu FLAC – kompresní poměry



Obrázek 3.46: Graf zobrazující optimální velikosti vstupních souborů u algoritmu TTA – kompresní poměry

4 Použitá počítačová sestava

- základní deska: GIGABYTE B650 GAMING X AX
- procesor: AMD Ryzen 5 7600X
- pevný disk: WD Black SN770 NVMe 1TB
- paměť RAM: 1x Kingston 16GB DDR5 5200MHz CL36 FURY Beast Black EXPO
- zdroj: Corsair TX750M (2021)
- chladič procesoru: SilentiumPC Fera 5 Dual Fan
- chladič do PC skříně: 2x Fractal Design Aspect 12 Black, 1x ARCTIC P14
- PC skříň: Fractal Design Focus 2 Black TG Clear Tint

5 Závěr

Zkoumáním poskytnutých dat vyskytující se v archivu ukdale a Archive of OSF Storage, jsem zjistil, jaká data obsahují jednotlivé archivy (viz. kapitola 2.3). První obsahuje textové soubory a druhý soubory typu WAVE. Proto do výsledné rešerše dostupných kompresních algoritmů (viz. kapitola 1.1.1) jsem zařadil i audio kodeky (viz. kapitola 1.1.2), které obsahují bezeztrátové kompresní algoritmy.

Po vyzkoušení naleznutých nalezených kompresních algoritmů a audio kodeků jsem zjistil, že použité audio kodeky podporují jako vstup jiné audio vstupy, proto pro testování jednotlivých algoritmů byl vybrán archiv s jménem Archiv of OSF Storage, který obsahuje soubory formátu *.wav a textové soubory s příponou *.dat jsem vyřadil z výsledné práce.

Pro účely práce jsem se rozhodl, že použiji kompresní algoritmy a audio kodeky jako spustitelné soubory typu EXE, které jsou dostupné ve verzi příkazovou řádku. Tímto krokem jsem se především snažil omezit algoritmy na oficiální verze, které jsou dostupné přímo od vývojářů těchto algoritmů, také jsem se chtěl zaměřit na verze, které se dají použít i bez znalosti programovacích jazyků jako je použitý programovací jazyk pro spouštění a vyhodnocení jednotlivých testů pro tuto práci.

Při zaměření na algoritmus 7ZIP jsem zjistil, že má velké množství nastavitelných parametrů. Pár jich umožňuje upravit výsledné chování algoritmu, mezi které patří především parametry level, parametr nastavující použití metody nebo filtrů a počet vláken. Levely se v tomto případě dají nastavit od hodnoty 0 až po hodnotu 9, přičemž hodnota 0 jen kopíruje data do výsledného archivu, takže pokud se nenastaví žádná kompresní metoda, tak výsledný soubor je prakticky stejně velký jako vstupní soubory. Metoda Copy se také dá nastavit pomocí druhého parametru kompresních metod a filtrů. Další používané kompresní metody jsou Deflate, který je nastaven v základu a nastavitelné metody LZMA a LZMA2, které oproti kompresní metodě Copy komprimují. Při kompresi se také dají použít kompresní filtry ARM, ARMT, BCJ, BCJ2, Delta, IA64, PPC a SPARC. Parametr nastavující počet vláken byl v mém případě použit v kombinaci s hodnotami 1, 8 a 16. Mezi další parametry, které se dají použít při kompresi, jsou například nastavení hesla a rekurzivní procházení, které nebyli použity.

Algoritmy BZIP2, GZIP a FLAC používají k ovlivnění komprese jen jediný parametr level, který do určité míry dokáže změnit chování těchto algoritmů. V případě algoritmu FLAC se dá tento parametr nastavit na hodnoty od 0 po 8 a zbytek algoritmu na hodnoty v rozmezí 1 až 9. Hodnota rovná nule v tomto případě také znamená, že algoritmus nekomprimuje.

Zbylé algoritmy použité v práci nepoužívají žádný parametr pro výslednou kompresi, proto se každý musí spokojit se základní kompresí, která je nastavená od vývojáře a nelze ji měnit, třeba za účelem ušetření paměti RAM nebo procesorové času.

Nejhorší komprese (viz. kapitola 3), které se nedají použít jsou především komprese pomocí filtrů a kompresní metodou Copy. Tyto dvě možnosti z většiny měli nízkou kompresi a výsledné komprimované soubory byli prakticky stejné, ale kompresní metoda Copy při některých nastaveních parametrů podává ještě dobré výsledky.

Nejlepší při vybraných souborech jsou audio kodeky následující kompresní metodou BZIP2, které zmenší soubory nejvíce. Algoritmus 7ZIP vykazuje spíše průměrné výsledky a nejhorší výsledky podávají GZIP a LZW.

Pro nejnižší hodnoty spotřeby procesorového času, paměti RAM a délky běhu algoritmu je nejlepší vybrat po každé algoritmus 7ZIP, i přes to, že většina horších hodnot je většinou způsobena kompresními filtry. Druhým nejlepším kandidátem je audio kodek FLAC, který je u spotřeby procesorového času a délky běhu algoritmu velmi blízko algoritmu 7ZIP.

Z hlediska kompresního poměru vychází nejlépe v testech algoritmus TTA a FLAC, který má nejlepší hodnoty velmi blízko algoritmu TTA, ale velmi blízko je také algoritmus BZIP2, který zaostává za audio kodeky jen o 3 %. Nejhorší výsledky podává algoritmus 7ZIP v kombinaci s filtry nebo při použití kopírovacích metod ve většině možností.

Při použití kompresních univerzálních kompresních metod je nejlepší vybírat spíše větší soubory a při použití audio kodeků menší soubory.

Mezi možná další rozšíření bych zařadil přidání algoritmů z různých knihoven dostupných pro použitý programovací jazyk, které by rozšířili použité kompresní algoritmy a audio kodeky. Následně by se dali najít novější kompresní algoritmy, které byly vymyšleny nedávno a tím rozšířit povědomí i o jiných kompresních algoritmech. V mém případě jsem při použití spustitelných EXE souborů žádný našel. Většina byla ve fázi, u které by se teprve EXE soubor musel vytvořit nebo napojit přes knihovny nebo nugety.

Bibliografie

1. *7ZIP – commandline*. 2022-11. Dostupné také z: https://7-zip.org/download.html.
2. *BZIP2 – commandline*. 2023-03. Dostupné také z: https://sourceforge.net/projects/gnuwin32/files/bzip2/1.0.5/bzip2-1.0.5-bin.zip/download?use_mirror=deac-fra&download=.
3. *GZIP – commandline*. 2023-03. Dostupné také z: https://sourceforge.net/projects/gnuwin32/.
4. *LZW – commandline*. 2023-03. Dostupné také z: https://sourceforge.net/projects/lzw/.
5. *FLAC – commandline*. 2022-11. Dostupné také z: https://xiph.org/flac/download.html.
6. *TTA 2.3 – commandline*. 2022-11. Dostupné také z: https://sourceforge.net/projects/tta/files/tta/ttaenc-win/.
7. *SBC vs aptX*. 2022-12. Dostupné také z: https://www.rtings.com/headphones/learn/sbc-aptx-which-bluetooth-codec-is-the-best.
8. *Bluetooth Audio Codecs Explained*. 2022-12. Dostupné také z: https://www.headphonesty.com/2020/03/bluetooth-audio-codecs-explained/?__cf_chl_tk=EvnSua7MUl.UFuo5HzHDkxglVKwmdBRlKtXN2sDlZi0-1671388103-0-gaNycGzNFqU.
9. *aptX*. 2022-12. Dostupné také z: https://www.aptx.com/.
10. *About lossless audio in Apple Music*. 2022-12. Dostupné také z: https://support.apple.com/en-us/HT212183.
11. *FLAC format*. 2022-12. Dostupné také z: https://xiph.org/flac/format.html.
12. *LDAC*. 2022-12. Dostupné také z: https://www.sony.net/Products/LDAC/.
13. *7-ZIP Download*. 2022-12. Dostupné také z: https://7-zip.org/download.html.
14. *7-ZIP Format*. 2022-12. Dostupné také z: https://7-zip.org/7z.html.
15. *7-ZIP Compression methods*. 2022-12. Dostupné také z: https://sevenzip.osdn.jp/chm/cmdline/switches/method.htm.

16. JUMAR, Richard; MAASS, Heiko; HAGENMEYER, Veit. Comparison of lossless compression schemes for high rate electrical grid time series for smart grid monitoring and analysis. *Computers & Electrical Engineering*. 2018, roč. 71, s. 465–476. ISSN 0045-7906. Dostupné z DOI: https://doi.org/10.1016/j.compeleceng.2018.07.008.

17. OSWAL, Savan; SINGH, Anjali; KUMARI, Kirthi. Deflate compression algorithm. *International Journal of Engineering Research and General Science*. 2016, roč. 4, č. 1, s. 430–436.

18. V, Pandimurugan; L, Sathish Kumar; J, Amudhavel; M, Sambath. Hybrid compression technique for image hiding using Huffman, RLE and DWT. *Materials Today: Proceedings*. 2022, roč. 57, s. 2228–2233. ISSN 2214-7853. Dostupné z DOI: https://doi.org/10.1016/j.matpr.2021.12.346. International Conference on Innovation and Application in Science and Technology.

19. CHEN, Gonglong; DONG, Wei. AdapTracer:Adaptive path profiling using arithmetic coding. *Journal of Systems Architecture*. 2018, roč. 88, s. 74–86. ISSN 1383-7621. Dostupné z DOI: https://doi.org/10.1016/j.sysarc.2018.05.011.

20. GIANCARLO, Raffaele; MANZINI, Giovanni; RESTIVO, Antonio; ROSONE, Giovanna; SCIORTINO, Marinella. The Alternating BWT: An algorithmic perspective. *Theoretical Computer Science*. 2020, roč. 812, s. 230–243. ISSN 0304-3975. Dostupné z DOI: https://doi.org/10.1016/j.tcs.2019.11.002. In memoriam Danny Breslauer (1968-2017).

21. *FLAC Feautures*. 2023-01. Dostupné také z: https://xiph.org/flac/features.html.

22. *FLAC Feautures*. 2023-01. Dostupné také z: https://fileinfo.com/extension/tta.

23. *Plotly.NET nuget*. 2023-05. Dostupné také z: https://www.nuget.org/packages/Plotly.NET/.

24. *Plotly.NET.ImageExport nuget*. 2023-05. Dostupné také z: https://www.nuget.org/packages/Plotly.NET.ImageExport/.

25. *Archive Ukdale – data*. 2023-05. Dostupné také z: https://jack-kelly.com/data/.

26. *Archive of OSF Storage – data*. 2023-05. Dostupné také z: https://osf.io/zye6d/files/osfstorage.

27. *Archive Ukdale – metadata*. 2023-05. Dostupné také z: https://www.nature.com/articles/sdata20157#Sec10.

28. *Archive of OSF Storage – metadata*. 2023-05. Dostupné také z: https://osf.io/zye6d/metadata?mode=&revisionId=&view_only=.