A=SIT Technology
E-Government Innovationszentrum

19. April 2021 Studie · Blockchain · 19 Min. Lesezeit

Inkrementelle Bitcoin Snapshots

Archiviert. Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

Stand 2021-04-19
Quelle https://technology2.a-sit.at/veroeffentlichungen/inkrementelle-bitcoin-snapshots/

In diesem Projekt wird eine Architektur für eine komprimierbare Blockchain vorgestellt, welche die primäre Blockchain durch eine zweite verkettete Blockchain ergänzt. Diese zweite Blockchain besteht aus regelmäßig erzeugten Snapshot-Blöcken, welche das aktuelle UTXO-Set zusammenfassen. Durch die Aufteilung des UTXO-Sets in drei Teile können sowohl neue als auch gelegentlich gestartete Nodes schnell und effektiv alle relevanten Informationen bekommen. Ein neuer Node muss nur den letzten Block in der zweiten Kette, sowie alle danach erzeugten Blöcke in der ersten Kette laden. Bereits teilweise synchronisierte Nodes können je nach Rückstand die inkrementellen Updates laden oder ebenfalls den gesamten letzten Block. Mittels der vorstellten komprimierbaren Blockchain, lässt sich der Speicherplatz- und Bandbreitenverbrauch um bis zu 98% reduzieren.

1. Einleitung

In diesem Projekt wird eine neue Architektur für Blockchains vorgestellt, welche neuen Nodes sowie gelegentlich gestarteten Nodes eine schnelle Synchronisation erlaubt. Bei Bitcoin, kann es derzeit je nach Computer und Internetanbindung mehrere Tage bis Wochen dauern, bis ein Full Node vollständig synchronisiert ist. Ein Full Node lädt üblicherweise die gesamte Blockchain herunter und überprüft diese. Ein Lightweight Node hingegen muss nicht die gesamte Blockchain herunterladen, braucht aber einen vertrauenswürdigen Full Node um zu funktionieren. Der in diesem Projekt vorgestellte Ansatz soll die Vorteile beider Lösungen, die Sicherheit und Privatsphäre eines Full Nodes, und die verringerten Speicher- und Bandbreitenanforderungen eines Lightweight Nodes, vereinen. Ein ähnlicher Ansatz wurde bereits in [1] vorgestellt, jedoch ohne Berücksichtigung von gelegentlich gestarteten Nodes. Dieser Nachteil wird in diesem Ansatz beseitigt. In den nächsten Abschnitten wird zuerst das nötige Hintergrundwissen vermittelt und anschließend der neue Ansatz vorgestellt.

2. Hintergrund Wissen

In diesem Bericht wird davon ausgegangen, dass ein Grundverständnis von Bitcoin vorhanden ist. Daher werden nur die für den vorgestellten Ansatz relevanten Aspekte beschrieben. Für die Grundlagen wird auf das Bitcoin Whitepaper [2] verwiesen. Die folgenden Abschnitte wurden aus

[1] übernommen und wo nötig aktualisiert bzw. angepasst.

2.1. UTXO Set

In der Bitcoin-Blockchain besteht jede Transaktion aus einem oder mehreren Transaktions-Inputs und Transaktions-Outputs. Diese werden in Folge als Inputs und Outputs bezeichnet. Outputs können als Inputs in darauffolgenden Transaktionen verwendet werden, wodurch Währungseinheiten verschoben oder aufgeteilt werden können. Eine Ausnahme sind sogenannte Coinbase-Transaktionen. Diese Transaktionen haben keinen Input, dürfen aber eine definierte Menge an Währungseinheiten erschaffen. Diese Outputs stellen die Belohnung für Miner dar, die einen gültigen Block gefunden und die Blockchain erweitert haben. Die Menge aller noch nicht ausgegebenen Outputs wird als UTXO-Set bezeichnet. UTXO steht für „Unspent Transaction Output“. Mittels dieses Sets können neu empfangene Blöcke und Transaktionen effizient auf ihre Gültigkeit überprüft werden, ohne die gesamte Blockchain durchsuchen zu müssen. Ein Vorteil dieses Sets gegenüber der gesamten Blockchain ist zudem, dass das UTXO Set nicht stetig größer wird, sondern auch schrumpfen kann. Im Vergleich dazu werden an die Blockchain laufend Blöcke angehängt, wodurch diese mit der Zeit immer größer wird. Mitte März 2021 benötigte die Bitcoin-Blockchain bereits ca. 330GB Speicherplatz [3]. Abbildung 1 zeigt das Wachstum der Bitcoin-Blockchain seit ihrem Start.

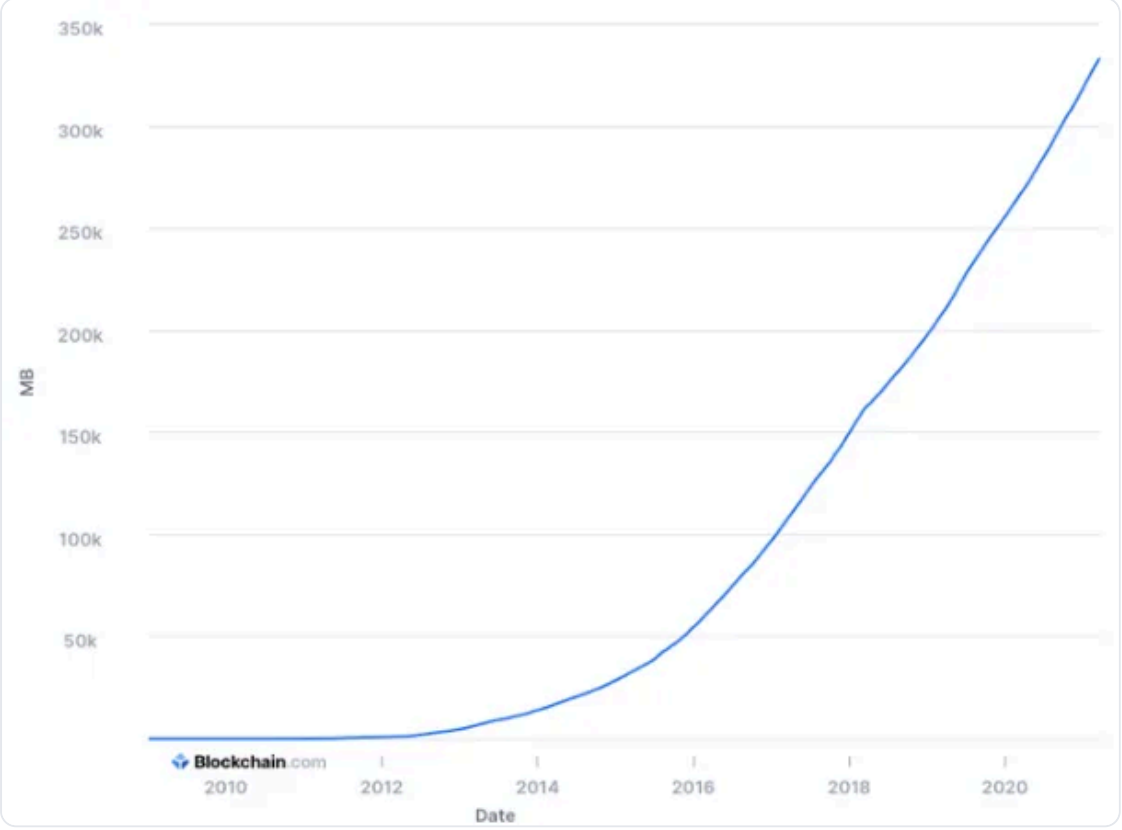


Abbildung 1: Größer der Bitcoin-Blockchain (Quelle: [3])

Es ist zu sehen, dass der benötige Speicherplatz anfangs nur sehr langsam stieg. In den letzten Jahren wurde Bitcoin bekannter und beliebter, was sich auch in volleren Blöcken und dadurch größerem Wachstum zeigt. Im Vergleich zu den ca. 330GB, benötigt das derzeitige UTXO-Set ca. 4,5GB [4]. Um das UTXO-Set aktuell zu halten, muss es nach jedem akzeptierten Block aktualisiert werden. Dabei werden alle Outputs,

die in dem neuen Block als Inputs verwenden wurden, entfernt und alle neu erstellten Outputs hinzugefügt. Dabei werden die Transaktionen genau in der Reihenfolge abgearbeitet, wie sie im Block hinterlegt sind. Das UTXO-Set ist speziell für Full Nodes relevant. Diese werden im nächsten Abschnitt vorgestellt.

2.2. Full Node

Bei einem Full Node handelt es sich um eine Software, welche sich zum Bitcoin-Netzwerk verbindet, die gesamte Blockchain herunterlädt, validiert und üblicherweise speichert. Die Software verifiziert jeden Block und jede Transaktion auf Einhaltung der Regeln. Blöcke müssen beispielsweise den Vorgängerblock referenzieren, dürfen nur einen definierten Maximalbetrag im Rahmen der Coinbase-Transaktion auszahlen und alle enthaltenen Transaktionen müssen gültig sein. Zudem dürfen Transaktionsoutputs nicht mehrmals ausgegeben werden. Auch darf nicht mehr ausbezahlt werden, als einbezahlt wurde. Es dürfen keine negativen Beträge vorkommen. Weiteres muss die im Transaktionsoutput definierte Bedingung erfüllt sein, um diesen Output verwenden zu können. In der Regel ist eine gültige Signatur erforderlich. Nachdem sich ein Full Node mit dem Netzwerk synchronisiert hat, d.h. die gesamte Blockchain geladen und verifiziert hat, beginnt die Software das Netzwerk zu unterstützen. Beispielsweise durch die Verifikation von empfangenen Transaktionen, die, sofern sie gültig sind, weiter im Netzwerk verteilt werden. Als Basis für diese Überprüfung zieht jeder Full Node jeweils die aktuelle lokale Kopie der Blockchain heran. Im Idealzustand haben alle Nodes im Netzwerk dieselbe lokale Kopie. Durch Netzwerkverzögerungen und andere Ereignisse kann es jedoch vorkommen, dass Nodes temporär unterschiedliche lokale Kopien haben. Da definiert ist, dass die regelkonforme Kette mit dem größten summierten Proof-of-Work als gültig zu werten ist, einigen sich alle Nodes üblicherweise relativ schnell auf eine gemeinsame Kette. Der Hauptvorteil eines Full Nodes liegt in der Sicherheit, da keinem fremden Anbieter vertraut werden muss. Zudem wird dem Netzwerk durch die Verifizierung und Weiterleitung von Transaktionen und Blöcken geholfen. Der größte Nachteil liegt im benötigen Speicherbedarf, welcher derzeit bei einigen hundert Gigabytes liegt [1] . Zusätzlich muss ein Full Node noch das derzeit aktuelle UTXO-Set errechnen und aktuell halten. Das UTXO-Set enthält alle noch nicht ausgegebenen Transaktionsoutputs und benötigt einige Gigabyte an Speicher. Dieses Set wird benötigt, um neue Transaktionen schnell und effizient auf Gültigkeit zu überprüfen. Da der Speicherbedarf eines Full Nodes mit der Zeit immer wächst, aber nicht alle Geräte über ausreichend Speicherplatz verfügen bzw. der Speicher anderweitig benötigt wird, wurden Ansätze entwickelt, wie ein Bitcoin Node mit geringeren Speicher- oder Bandbreitenanforderungen betrieben werden kann. Zwei dieser Ansätze, die von Bitcoin Core, der Referenzimplementierung von Bitcoin unterstützt werden, heißen „Pruning“ and „blocksonly“. Diese werden in den nächsten beiden Abschnitten kurz vorgestellt.

2.2.1. Pruning

Seit Bitcoin Core Version 0.11.0 (2015) wird Pruning unterstützt [5]. Pruning erlaubt das Betreiben eines Full Nodes ohne die komplette Blockchain lokal zu speichern. Stattdessen wird ein Limit für den maximal zu verwendeten Speicherplatz angegeben. Der Node speichert nur die aktuellsten Blöcke und löscht ältere, sobald das Limit überschritten wird. Bei diesem Ansatz muss trotzdem die gesamte Blockchain heruntergeladen und verifiziert werden. Sollte es zu Problemen kommen oder ein „rescan“ notwendig sein, muss die gesamte Blockchain neu geladen werden.

1 Stand März 2021 ca. 330GB für die Blockchain.

2.2.2. Blocks Only

In Bitcoin Core Version 0.12 wurde die Option „-blocksonly“ hinzugefügt. Diese ermöglicht, den Netzwerkverkehr zu reduzieren. Wird diese Option aktiviert, fordert der Node keine unbestätigten Transaktionen mehr an und leitet diese auch nicht weiter. Somit reduziert der Node den Netzwerkverkehr auf ein notwendiges Minimum [6]. 2016 wurde bei einem Experiment ein um 88% reduzierter Netzwerkverkehr beobachtet, wenn „-blocksonly“ aktiviert wurde [7]. Eine andere Möglichkeit, Speicherplatz und Bandbreite zu sparen, bieten Lightweight Nodes. Diese werden im nächsten Abschnitt vorgestellt.

2.3. Lightweight Node

Lightweight Nodes laden nicht die komplette Blockchain herunter und benötigen daher weniger Speicherplatz und Bandbreite. Stattdessen laden Lightweight Nodes nur die Blockheader herunter. Dadurch können sie überprüfen, ob die Kette prinzipiell gültig ist, d.h. ob die Verlinkung und der Schwierigkeitsgrad passen. Zur Transaktionsvalidierung wird ein Ansatz namens „Simplified Payment Verification“ oder kurz SPV verwendet. Mittels SPV kann ein Lightweight Node sicherstellen, dass eine Transaktion in einem Block aufgenommen wurde. Dafür muss der Lightweight Node einen Full Node kontaktieren, welcher eine Bestätigung schickt. Weiteres benachrichtigt der Full Node den Lightweight Node über Transaktionen, die ihn betreffen. Lightweight Nodes müssen dem Full Node vertrauen, dass dieser Transaktionen und Blöcke entsprechend den Regeln verifiziert und auch, dass sie über alle relevanten Transaktionen informiert werden. Zusätzlich hat die Verwendung eines Lightweight Nodes Auswirkungen auf die Privatsphäre, da zu beobachtende Adressen bekannt gegeben werden müssen. Daher sollten Lightweight Nodes nur mit eigenen oder vertrauenswürdigen Full Nodes verbunden werden. Im nächsten Abschnitt wird ein neuer Ansatz vorgestellt, welcher höhere Sicherheit und Privatsphäre bietet und gleichzeitig den Speicherplatz und Bandbreitenbedarf senkt.

3. Komprimierbare Blockchain

Ziel dieses Ansatzes ist es, die Synchronisation für neue Nodes sowie für gelegentlich genutzte Nodes schneller und mit weniger Speicher- und Bandbreitenbedarf zu ermöglichen. Weiteres soll nach der erfolgreichen Synchronisation keine Abhängigkeit zu Full Nodes bestehen. Stattdessen soll der Node selbstständig Blöcke und Transaktionen verifizieren können, sowie nach vollständiger Synchronisation auch das Netzwerk unterstützen können. Die Grundidee ist, neben der Hauptkette eine zweite Kette zu bauen, deren Blöcke alle relevanten Daten für eine schnelle Synchronisation enthalten. Diese Blöcke werden Snapshot-Blöcke genannt. Im Projekt „Komprimierbare Blockchain“ [1] wurde das aktuelle UTXO-Set in den Snapshot-Blöcken gespeichert, wodurch sich neue Nodes schnell synchronisieren konnten. Der in [1] vorgestellte Ansatz ist jedoch nicht optimal für gelegentlich genutzte Nodes, da diese entweder die bereits geladenen Daten verwerfen und neu synchronisieren, oder auf die Snapshot-Blöcke verzichten und alle Blöcke seit der letzten Synchronisation laden müssen. Im ersten Fall hat der Node keinen Vorteil gegenüber einem frischen Node, da alle bereits geladenen Daten verworfen werden. Im zweiten Fall nutzt der Node die herkömmliche Synchronisation und profitiert nicht von den Snapshot-Blöcken. Um diese Nachteile zu vermeiden, teilen wir die in Snapshot-Blöcken gespeicherte Information in drei Teile auf:

- Der erste Teil enthält alle UTXOs, bei denen davon ausgegangen wird, dass sie nicht mehr ausgegeben werden und die noch nicht im ersten Teil eines vorherigen Snapshot-Blockes aufgenommen wurden.
- Der zweite Teil enthält alle UTXOs, die in keinem bisherigen Snapshot-Block im ersten Teil vorkommen.
- Der dritte und letzte Teil enthält alle aktuellen UTXOs, welche in diesem Snapshot-Block noch nicht vorkommen.

Dadurch bekommt ein Node verschiedene Möglichkeiten, um vom lokalen Blockchain-Stand zum aktuellen zu gelangen:

1. Der Node kann alle Blöcke ab dem letzten lokal vorhanden Block laden. Die Variante ist voraussichtlich nur für beinahe synchronisierte Nodes optimal.
2. Der Node kann ausgehend vom letzten lokal vorhandenem Snapshot-Block alle darauffolgenden ersten Teile der Snapshot-Blöcke laden, sowie den zweiten Teil des letzten Snapshot-Blockes sowie alle nach dem letzten Snapshot-Block hinzugefügten Blöcke der ersten Kette.
3. Der Node kann alle lokal vorhanden Daten verwerfen und stattdessen den letzten Snapshot-Block sowie alle nach dem letzten Snapshot-Block hinzugefügten Blöcke der ersten Kette laden.

Damit die Nodes die optimale Entscheidung treffen können, werden in den Snapshot-Blöcken neben den drei Teilen auch deren Größe gespeichert. Dadurch kann effizient festgestellt werden, welche Synchronisationsmethode optimal ist. Durch die Verlinkung der Snapshot-Blöcke in der ersten Kette kann deren Validität und Integrität leicht überprüft werden.

Der Ansatz ist in Abbildung 2 visualisiert. Blöcke in der Hauptkette (in schwarz dargestellt) zeigen neben ihrem Vorgängerblock auch noch auf den derzeit aktuellsten Snapshot-Block. Gibt es noch keinen Snapshot-Block, wird stattdessen der gemeinsame Genesis-Block verlinkt. Der erste Block wird Genesis-Block genannt und dient als vertrauenswürdiger Startpunkt. Dies ist in Abbildung 2a dargestellt. Die Snapshot-Blöcke werden regelmäßig erstellt, beispielsweise alle 10.000 Blöcke. Abbildung 2b zeigt eine komprimierbare Blockchain mit einem Snapshot-Block und Abbildung 2c zeigt die komprimierbare Blockchain nachdem ein zweiter Snapshot-Block hinzugefügt wurde. Es ist nun erkennbar, dass die Snapshot-Blöcke eine zweite Blockchain bilden (in grün dargestellt).

(a) Eine Standard Blockchain mit einer zusätzlichen Referenz auf den letzten Block in der Snapshot-Kette bzw. solange kein Block vorhanden ist auf den gemeinsamen Genesis-Block.

Genesis	**BP BP BP 2 δ-1 δ Snapshot 1**	**Col3**
Genesis	BPδ BPδ-1 Snapshot 1 BP2	Snapshot 1

(b) Komprimierbare Blockchain mit einem Snapshot-Block.

(c) Komprimierbare Blockchain mit zwei Snapshot-Blöcken.

Abbildung 2: Komprimierbare Blockchain, bei der regelmäßig ein Snapshot-Block erstellt wird. Die Snapshot- Blöcke formen eine zweite Kette. Wenn sich ein neuer Node mit dem Netzwerk verbindet, erhält er im Idealfall von allen Nachbarn dieselbe Information über die derzeit gültige Kette. Falls nicht, muss der Node alle Ketten betrachten und die beste auswählen. Zuerst lädt der Node alle Blockheader und überprüft, ob die Verlinkung und die Schwierigkeitsgrade korrekt sind. Schlägt diese Überprüfung fehl, werden alle ungültigen Blöcke verworfen. Nachdem alle Ketten überprüft wurden, wird die (gültige) Kette mit dem größten kombinierten Proof-of-Work ausgewählt. Im nächsten Schritt errechnet der Node die optimale Synchronisierungsmethode und lädt die entsprechenden Blöcke. Nach der erfolgreichen Synchronisierung kann der Node neue Blöcke und Transaktionen selbstständig überprüfen und das Netzwerk unterstützen.

Abbildung 3 zeigt die Größe der Bitcoin-Blockchain, der einzelnen Blöcke, sowie des UTXO-Sets in Abhängigkeit von der Blocknummer in linearer Darstellung (Abbildung 3a) sowie mit logarithmischer Y-Achse (Abbildung 3b). Abbildung 3a verdeutlicht die Größenunterschiede der Blockchain im Vergleich zum UTXO-Set. Durch die logarithmische Y-Achse in Abbildung 3b sind die unterschiedlichen Blockgrößen gut erkennbar. Es ist zu sehen, dass zu Beginn die Blöcke relativ klein waren, dann aber immer voller und größer wurden, bis schließlich die maximale Blockgröße von 1MB erreicht wurde. Mit Block 477120 wurde BIP 91, bekannt als „Segregated Witness“ aktiviert. Bei „Segregated Witness“ wird die Größe der Zeugendaten nur zu 25% berücksichtigt, wodurch die berechnete Blockgröße unter dem 1MB Limit liegt, gleichzeitig Blöcke aber mehr als 1MB an Speicher benötigen können.

Col1	Col2	Col3	Col4	Col5	Col6
0 100	000 200	000 300	000 400	000 500	000 600

Col1	250000 200000 [Megabyte] 150000 100000 Größe 50000 0 0 100000 200000 300000 400000 500000 600000 -50000 Blocknummer Blockgröße Blockchain Größe UTXO Größe	Col3
(a)	(a)	
	0,0001 0,001 0,01 0,1 1 10 100 1000 10000 100000 1000000 0 100000 200000 300000 400000 500000 600000 Größe [Megabyte] Blocknummer Blockgröße Blockchain Größe UTXO Größe	
(b)	(b)	

Abbildung 3: Visualisierung der Größe der Blöcke, der UTXO Snapshots und der Größe der gesamten Blockchain. Im nächsten Abschnitt wird der vorgeschlagene Ansatz evaluiert. Aufgrund der in Abbildung 3 erkennbaren Größenunterschiede der Blockchain und des UTXO-Sets erwarten wir ein großes Einsparungspotential.

4. Evaluierung

Als erstes wurde die Lebensdauer der Transaktionsoutputs analysiert. Als Lebensdauer bezeichnen wir den Zeitraum zwischen der Erstellung eines Transaktionsoutputs und der Verwendung als Transaktionsinput. Die Lebensdauer wird in Blöcken gemessen. Abbildung 4 und Abbildung 5 visualisieren das Ergebnis. Die x-

Achse zeigt die Lebensdauer in Blöcken, die y-Achse die Anzahl der Transaktionsoutputs. Beide Grafiken zeigen dieselben Daten, jedoch verwendet Abbildung 4 eine lineare x-Achse wohingegen Abbildung 5 eine logarithmische x-Achse verwendet. Beide Abbildungen verwenden eine logarithmische y-Achse. Es ist erkennbar, dass sehr viele Transaktionsoutputs relativ kurz nach der Erstellung wieder als Input verwendet werden. Weiteres ist erkennbar, das viele der frühen Transaktionsoutputs bis heute nicht ausgegeben wurden.

Abbildung 4: Histogramm über die Lebensdauer von UTXOs. Die x-Achse zeigt die Lebensdauer in Blöcken, die y-Achse die Anzahl der Transaktionen. Aufgrund des großen Wertebereichs wurde eine logarithmische y-Achse gewählt. Die grünen Datenpunkte stellen bereits ausgegebene Transaktionsoutputs dar, die blauen stellen die bis zum Block 620.000 noch nicht ausgegebenen UTXOs dar.

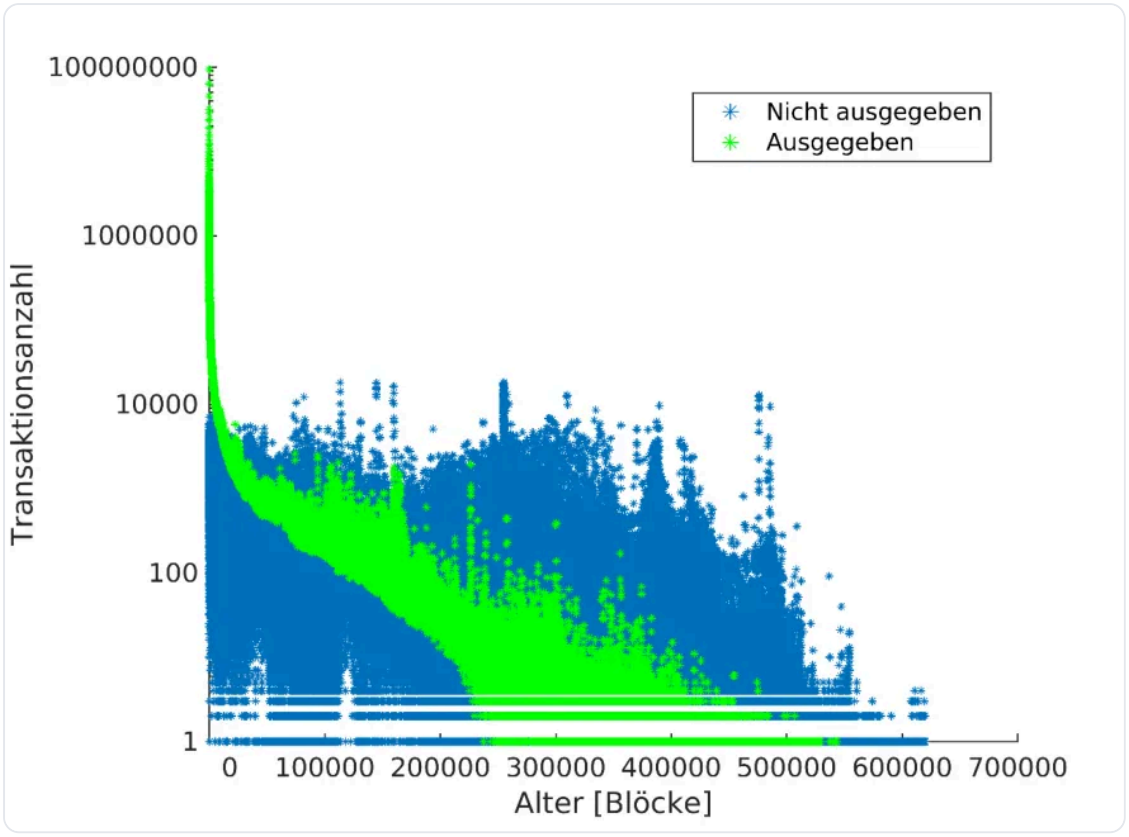


Abbildung 5: Diese Abbildung zeigt wie Abbildung 4 ein Histogramm über die Lebensdauer von UTXOs, allerdings mit logarithmischer x- und y-Achse. Die grünen Datenpunkte stellen bereits ausgegebene Transaktionsoutputs dar, die blauen stellen die bis zum Block 620.000 noch nicht ausgegebenen UTXOs dar. Abbildung 6 zeigt statt der Anzahl der Transaktionsoutputs deren aufsummierte Größe in Abhängigkeit der Lebensdauer. Abbildung 6 und Abbildung 7 zeigen dieselben Daten, jedoch verwenden Abbildung 6 eine lineare und Abbildung 7 eine logarithmische x-Achse.

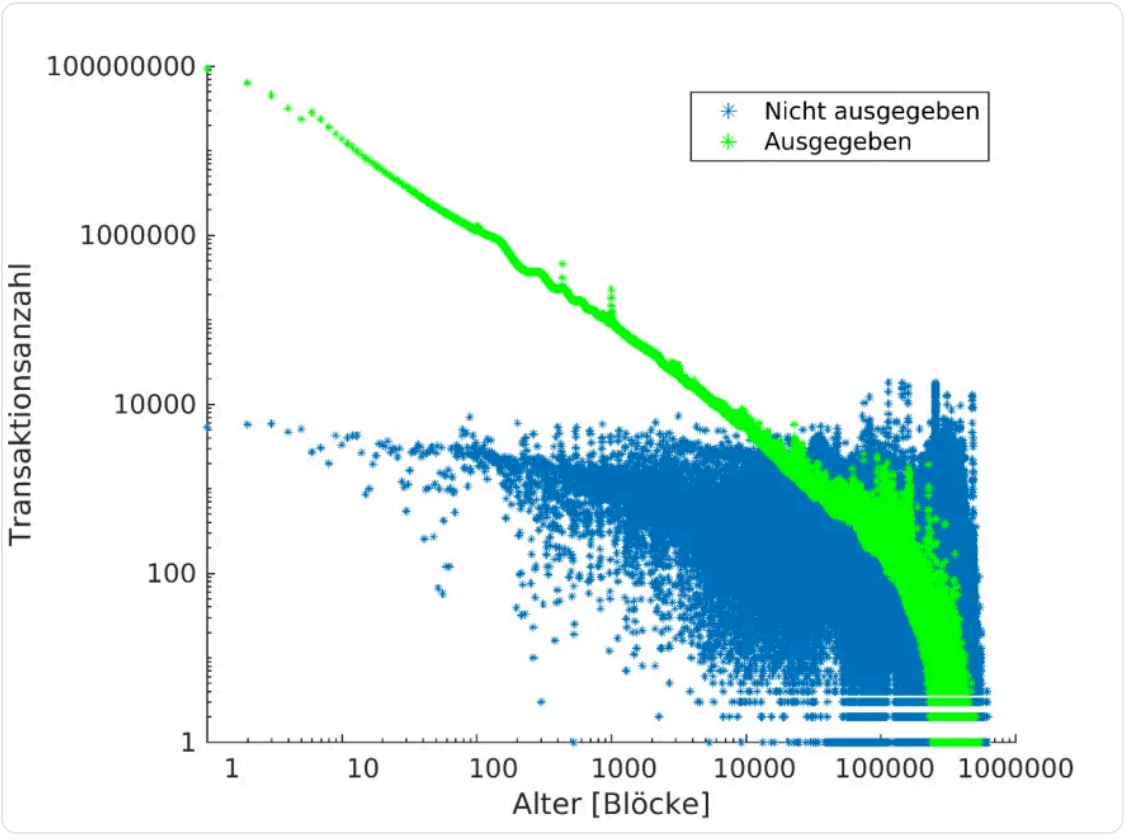
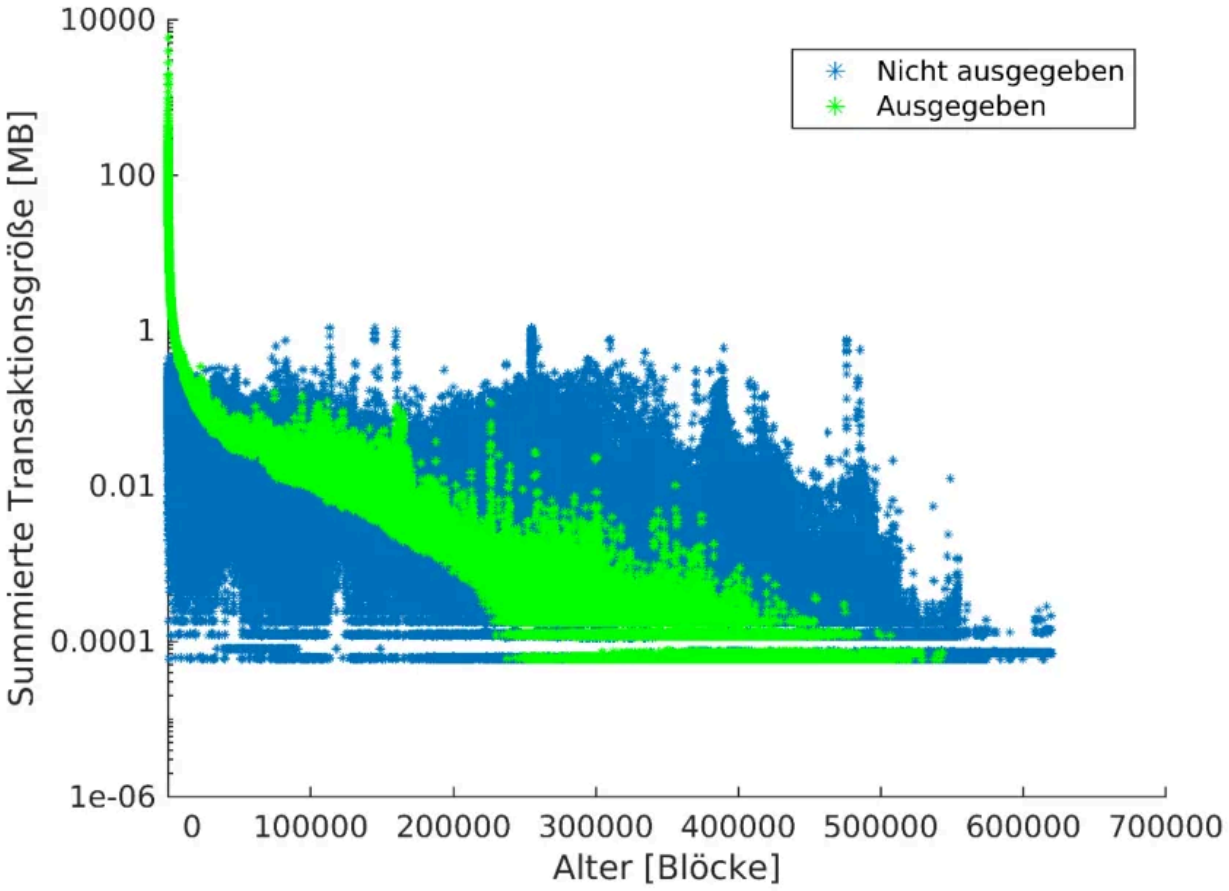


Abbildung 6: Diese Abbildung zeigt die summierte Größe alle Transaktionsoutputs je nach Lebensdauer. Die x-Achse zeigt die Lebensdauer in Blöcken, die y-Achse die Größe der Transaktionen. Aufgrund des großen Wertebereichs wurde eine logarithmische y-Achse gewählt. Die grünen Datenpunkte stellen bereits ausgegebene Transaktionsoutputs dar, die blauen stellen bis zum Block 620.000 noch nicht ausgegebene UTXOs dar. Abbildung 7: Diese Abbildung zeigt wie Abbildung 6 die summierte Größe alle Transaktionsoutputs je nach Lebensdauer. Die x-Achse zeigt wieder die Lebensdauer in Blöcken, die y-Achse die Größe der


Nicht ausgegeben
Ausgegeben
Summierte Transaktionsgröße [MB]
10000
100
1
0.01
0.0001
1e-06
0
100000
200000
300000
400000
500000
600000
700000
Alter [Blöcke]

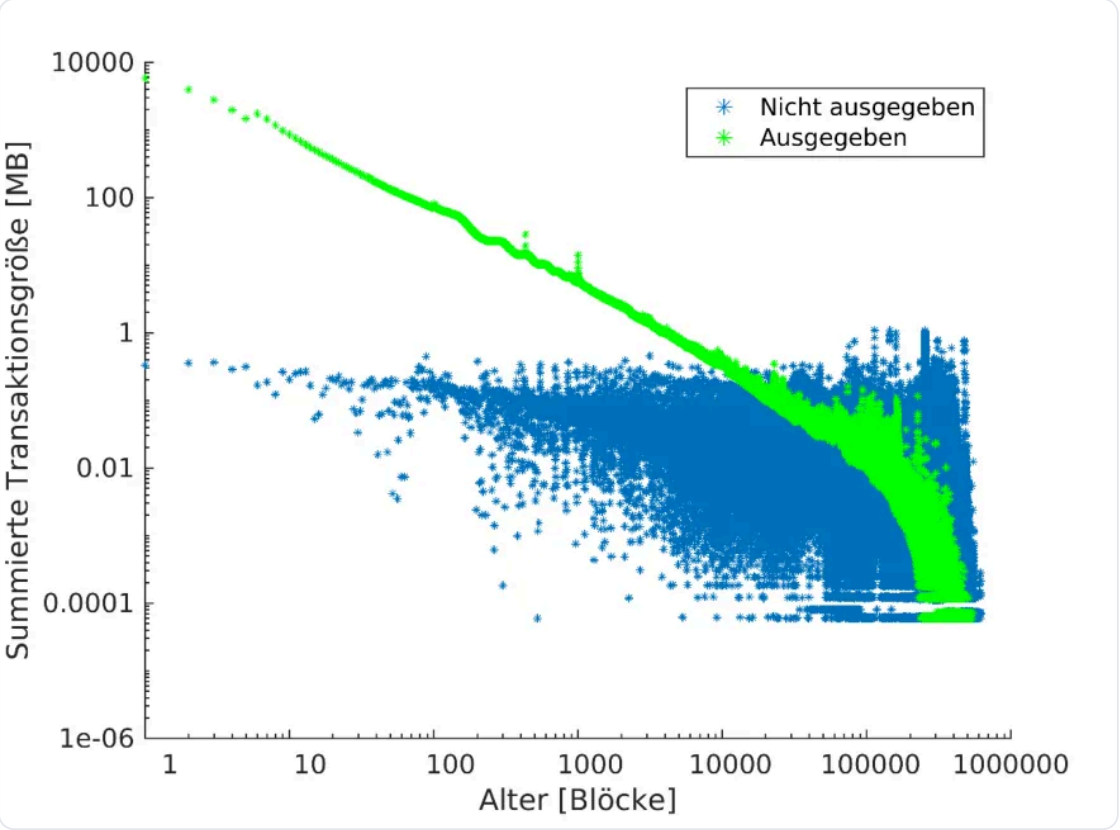


Abbildung 8: Zu ladende Datenmenge in Abhängigkeit der Länge, der bereits lokal vorhandenen Blockchain. Als nächstes werden die anderen beiden in Abschnitt 3 vorgestellten Synchronisierungsmethoden evaluiert. Hierfür sind zweit Faktoren wesentlich, einerseits das Snapshot-Block-Intervall und andererseits, die Heuristik, die für jede Transaktion abzuschätzen versucht, ob die Transaktion in Zukunft ausgegeben wird. Als Intervall wurde 1.000 und 10.000 Blöcke gewählt, als Heuristik wurde ein einfacher Schwellwert gewählt. Bei allen Transaktionen, die seit mindestens 100 bzw. mindestens 1.000 Blöcken nicht ausgegeben wurden, wird davon ausgegangen, dass sie auch in Zukunft nicht mehr ausgegeben werden. Sollte diese Annahme falsch sein, funktioniert der Ansatz trotzdem, es kommt nur zu minimal größeren Snapshot-Blöcken, da zusätzlich eine Korrektur gespeichert werden muss (40 Byte pro Output). Abbildung 9 zeigt die zu ladende Datenmenge, wenn die zweite Synchronisierungsmethode gewählt wird, bei einem Snapshot-Block-Intervall von 1.000 Blöcken und einem Schwellwert von 100 Blöcken. Abbildung 10 zeigt die zu ladende Datenmenge, wenn die dritte Synchronisierungsmethode gewählt wird und Abbildung 11 zeigt das Minimum aller drei Varianten. Abbildung 12 zeigt welche Synchronisierungsmethode in Abhängigkeit des vorhandenen und gewünschten Blockes die kleinste Datenmenge benötigt.

Transaktionen. Beide Achsen verwenden eine logarithmische Skalierung. Die grünen Datenpunkte stellen bereits ausgegebene Transaktionsoutputs dar, die blauen stellen bis zum Block 620.000 noch nicht ausgegebene UTXOs dar. Abbildung 8 zeigt die zu ladende Datenmenge in Abhängigkeit des letzten bereits lokal vorhandenen Blocks, ohne Berücksichtigung der Snapshot-Blöcke. Dies entspricht der ersten der drei in Abschnitt 3 vorgestellten Synchronisierungsmethoden. Die x-Achse definiert den gewünschten Block, die y-Achse gibt den letzten bereits vorhandenen Block an und die z-Achse zeigt die zu ladende Datenmenge in Gigabyte. Wie zu erwarten war, sinkt die zu ladende Datenmenge je mehr Blöcke bereits lokal vorhanden sind. Zu beachten ist, dass es sich bei den dreidimensionalen Abbildungen nicht um Oberflächen, sondern um diskrete Datenpunkte handelt. Um die Datenmenge zu begrenzen wurde nur jeder fünfhundertste Datenpunkt dargestellt.

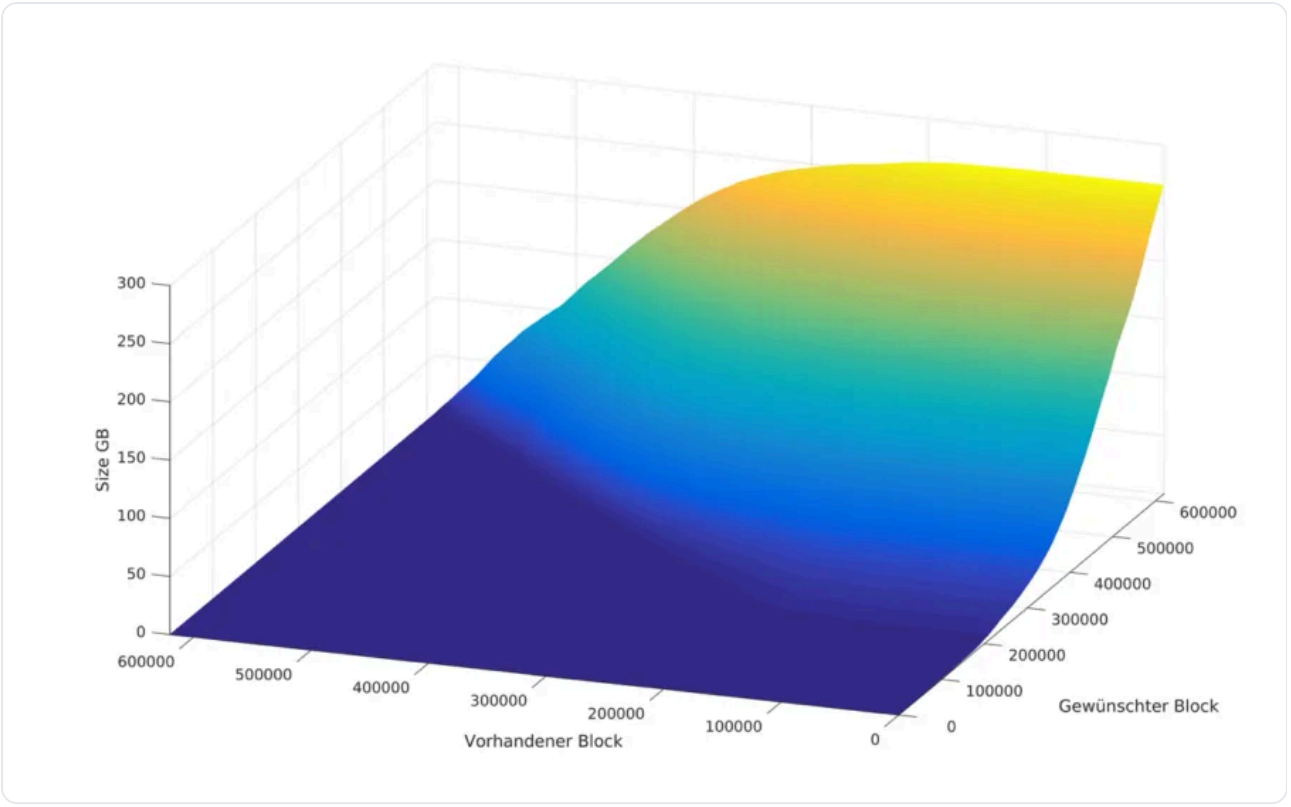
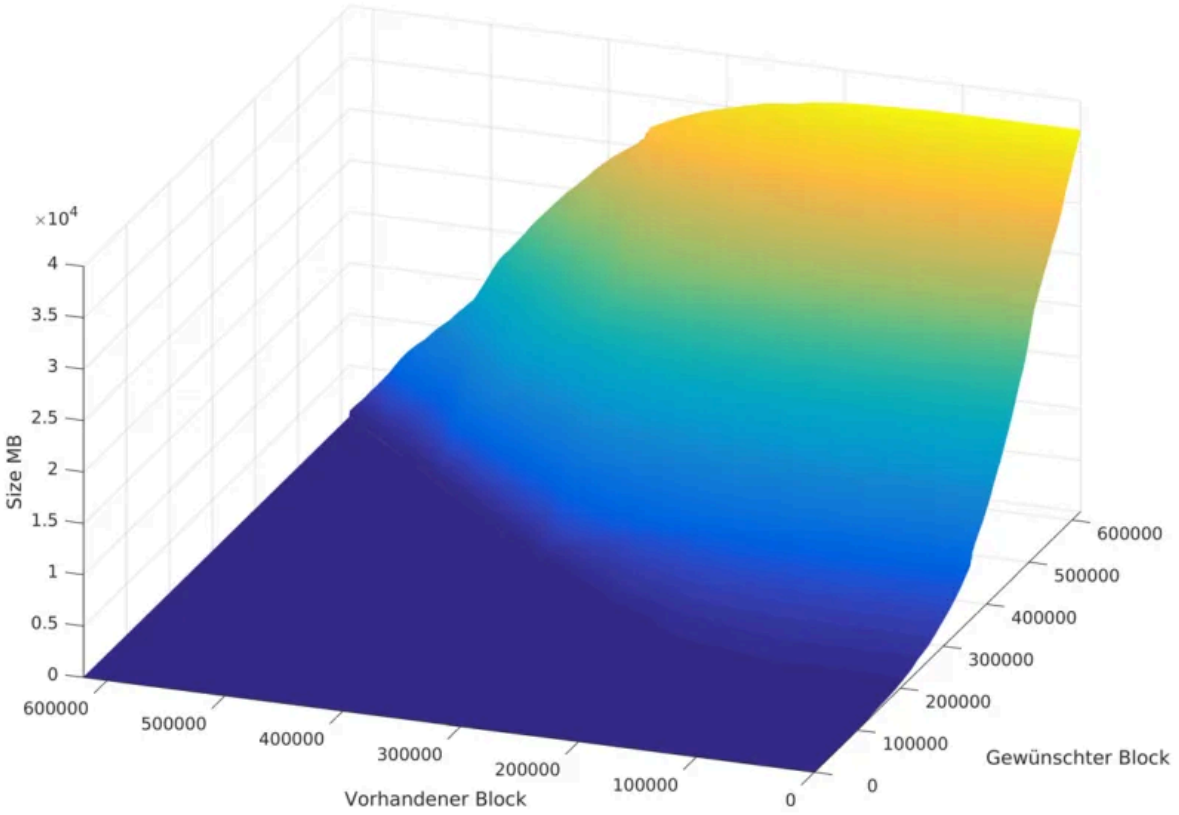


Abbildung 9: Zu ladende Datenmenge, wenn die zweite Synchronisierungsmethode gewählt wird, bei einer Blockchain mit einem Snapshot-Block-Intervall von 1.000 Blöcken und einem Schwellwert von 100 Blöcken. Abbildung 10: Zu ladende Datenmenge, wenn die dritte Synchronisierungsmethode gewählt wird, bei einer Blockchain mit einem Snapshot-Block-Intervall von 1.000 Blöcken und einem Schwellwert von 100 Blöcken.


×10^4
4
3.5
3
2.5
2
1.5
1
0.5
0
Size MB
600000
500000
400000
300000
200000
100000
0
Vorhandener Block
600000
500000
400000
300000
200000
100000
0
Gewünschter Block

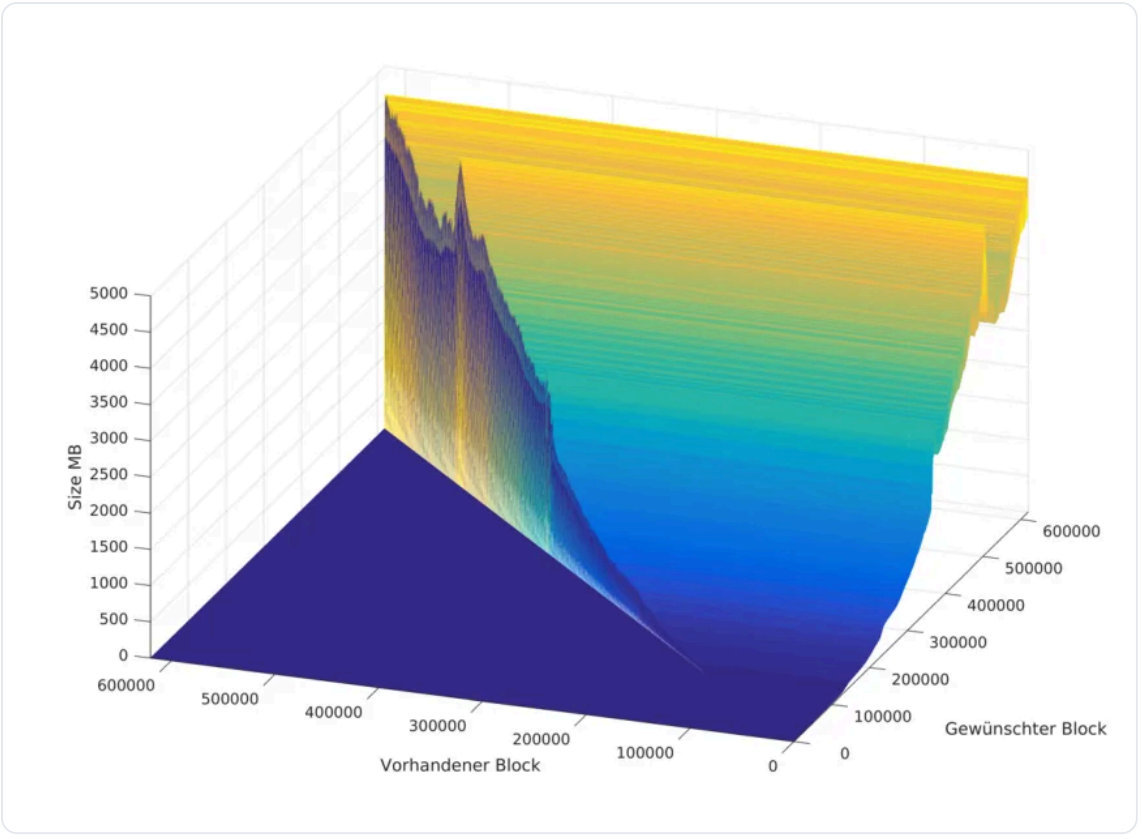
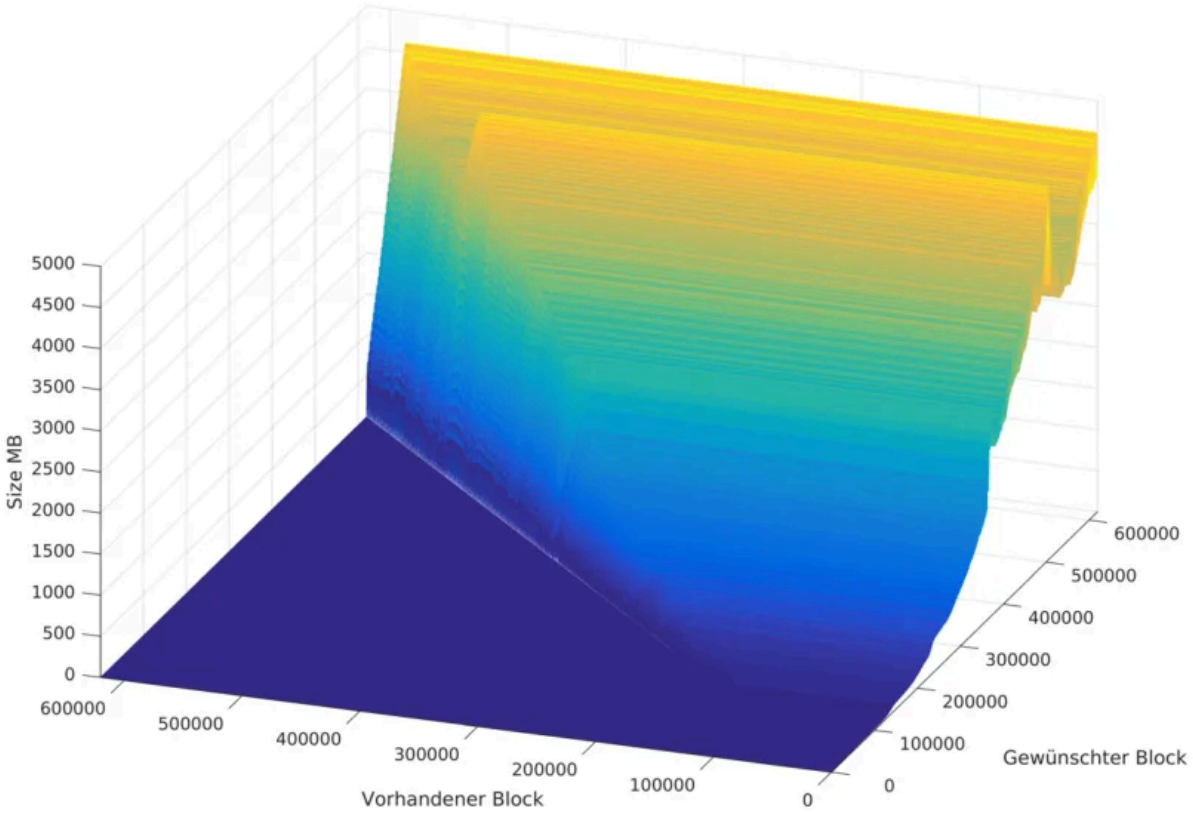


Abbildung 11: Zu ladende Datenmenge, wenn die beste der drei Synchronisierungsmethoden gewählt wird, bei einer Blockchain mit einem Snapshot-Block-Intervall von 1.000 Blöcken und einem Schwellwert von 100 Abbildung 12: Diese Abbildung zeigt, welche Synchronisierungsmethode optimal ist, in Abhängigkeit des lokal vorhandenen und des gewünschten Blockes. Es gibt vier verschiedene Bereiche. Beim blauen Bereich ist die lokal vorhandene Blockchain bereits mindestens so lange, wie die gewünschte. Beim kaum wahrnehmbaren hellblauen Bereich (wenige Pixel) ist die erste Synchronisierungsmethode, beim grünlichen


Size MB
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
600000
500000
400000
300000
200000
100000
0
Vorhandener Block
600000
500000
400000
300000
200000
100000
0
Gewünschter Block

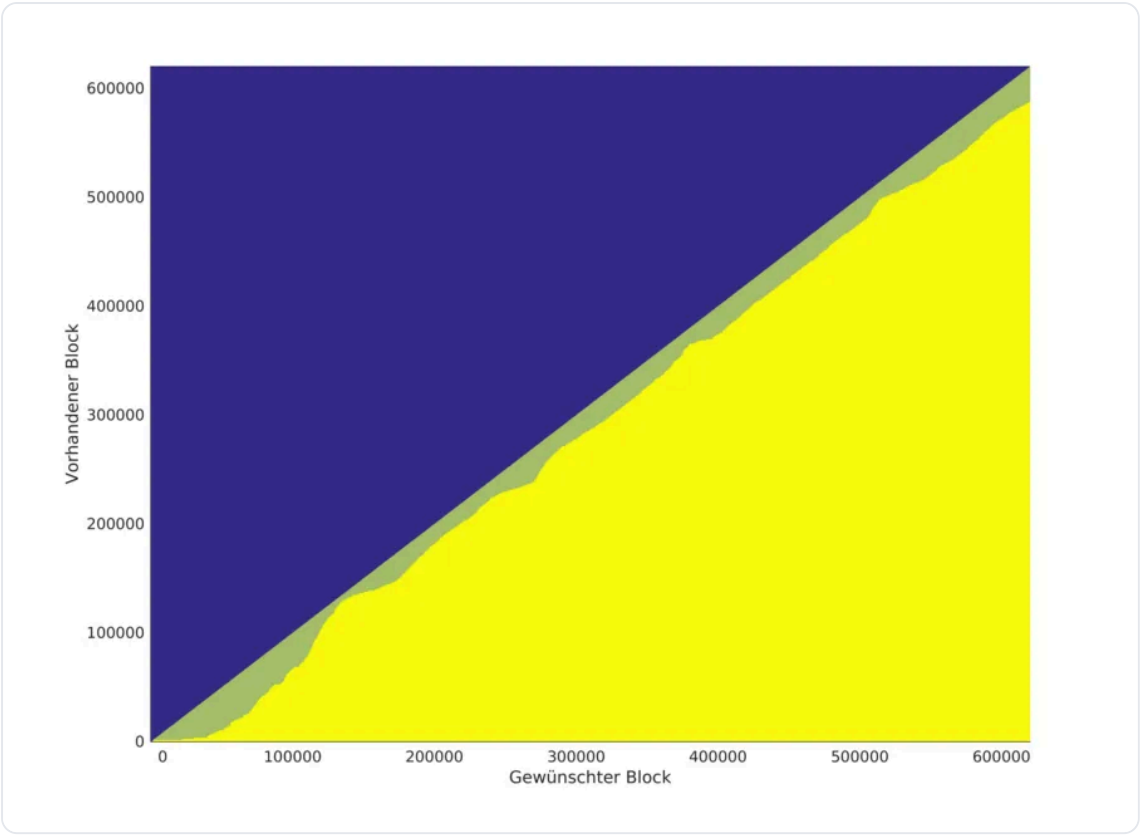


Abbildung 13: Zu ladende Datenmenge, wenn die beste der drei Synchronisierungsmethoden gewählt wird, bei einer Blockchain mit einem Snapshot-Block-Intervall von 1.000 Blöcken und einem Schwellwert von 1.000

Bereich die Zweite und beim gelben Bereich die Dritte optimal, bei einer Blockchain mit einem Snapshot-Block-Intervall von 1.000 Blöcken und einem Schwellwert von 100 Blöcken. Abbildung 13 und Abbildung 14 zeigen dieselben Informationen wie die Abbildung 11 und Abbildung 12, jedoch mit einem Schwellwert von 1.000 Blöcken. Der größere Schwellwert verkleinert die zu ladende Datenmenge nur minimal, der Einfluss ist kaum wahrnehmbar.

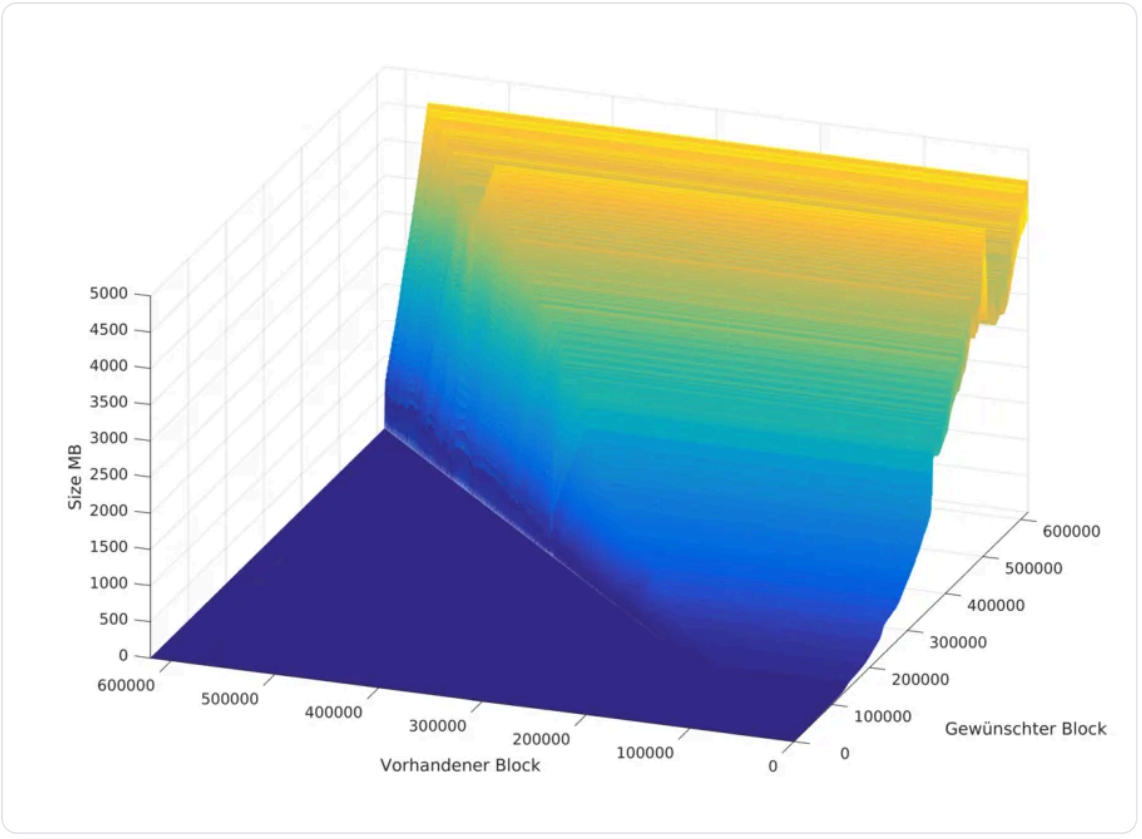


Abbildung 14: Diese Abbildung zeigt, welche Synchronisierungsmethode optimal ist, in Abhängigkeit des lokal vorhandenen und des gewünschten Blockes. Es gibt vier verschiedene Bereiche. Beim blauen Bereich ist die lokal vorhandene Blockchain bereits mindestens so lange, wie die gewünschte. Beim kaum wahrnehmbaren hellblauen Bereich (wenige Pixel) ist die erste Synchronisierungsmethode, beim grünlichen Bereich die Zweite und beim gelben Bereich die Dritte optimal, bei einer Blockchain mit einem Snapshot-Block-Intervall von 1.000 Blöcken und einem Schwellwert von 1.000 Blöcken. Die Abbildung 15 bis Abbildung 18 zeigen dieselben Informationen wie die Abbildung 9 bis Abbildung 12, jedoch für ein Snapshot-Block-Intervall von 10.000 Blöcken. Der Einfluss des Snapshot-Block-Intervalls ist deutlich wahrnehmbar. Wie erwartet steigt die zu ladende Datenmenge bei größeren Snapshot-Block-Intervallen. Im Gegenzug haben die Miner weniger Aufwand, da seltener Snapshot-Blöcke erzeugt werden müssen. Zudem benötigen Miner und Full Nodes, welche alle Blöcke aufheben wollen, weniger Speicherplatz. Da alte Snapshot-Blöcke jedoch jederzeit wieder neu erstellt werden können und zudem auch kaum von neuen Nodes benötigen werden, können diese allerdings auch gelöscht werden.

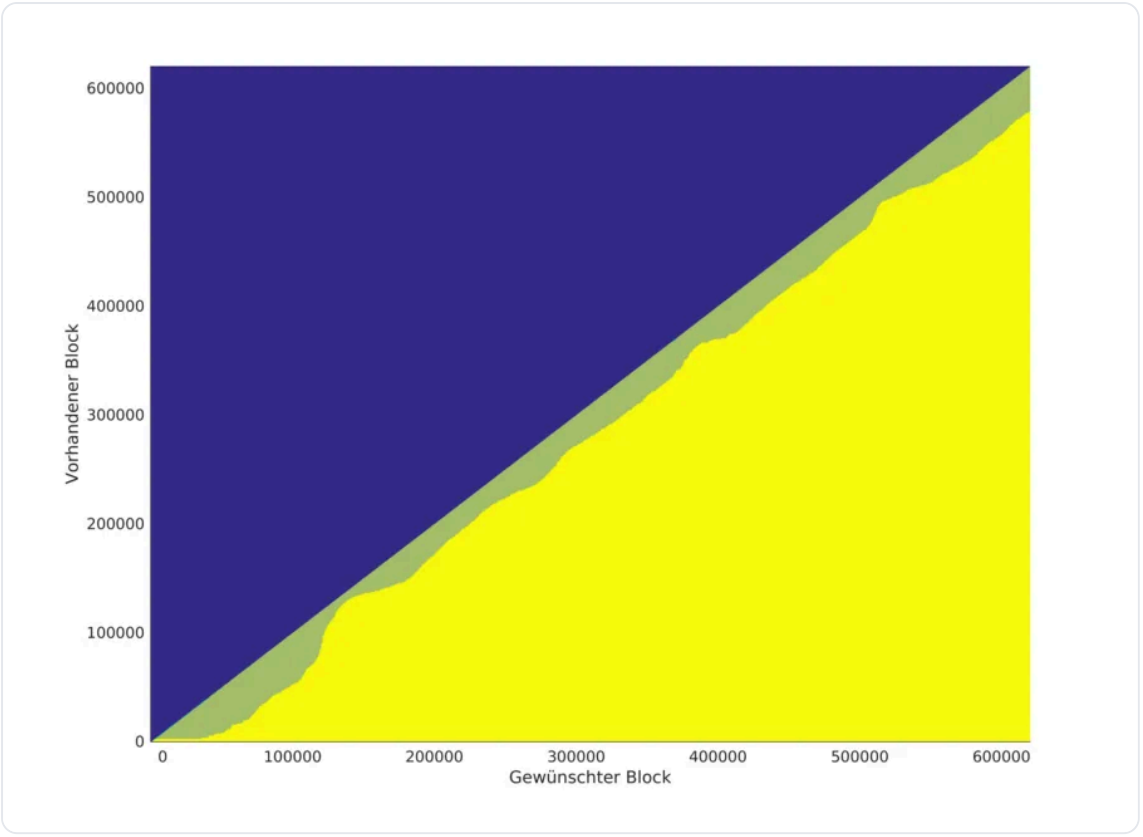
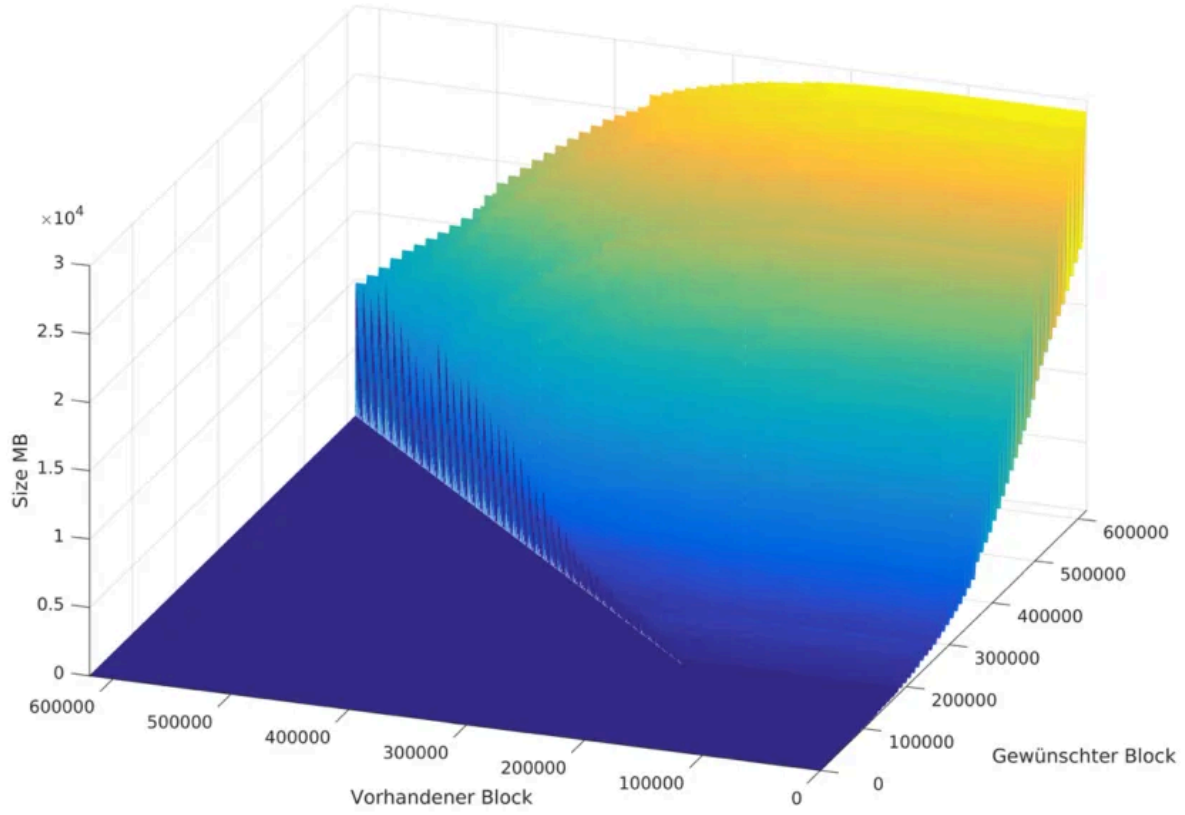


Abbildung 15: Zu ladende Datenmenge, wenn die zweite Synchronisierungsmethode gewählt wird, bei einer Blockchain mit einem Snapshot-Block-Intervall von 10.000 Blöcken und einem Schwellwert von 100 Blöcken. Abbildung 16: Zu ladende Datenmenge, wenn die dritte Synchronisierungsmethode gewählt wird, bei einer Blockchain mit einem Snapshot-Block-Intervall von 10.000 Blöcken und einem Schwellwert von 100 Blöcken.


×10^4
3
2.5
2
1.5
1
0.5
0
Size MB
600000
500000
400000
300000
200000
100000
0
Vorhandener Block
600000
500000
400000
300000
200000
100000
0
Gewünschter Block

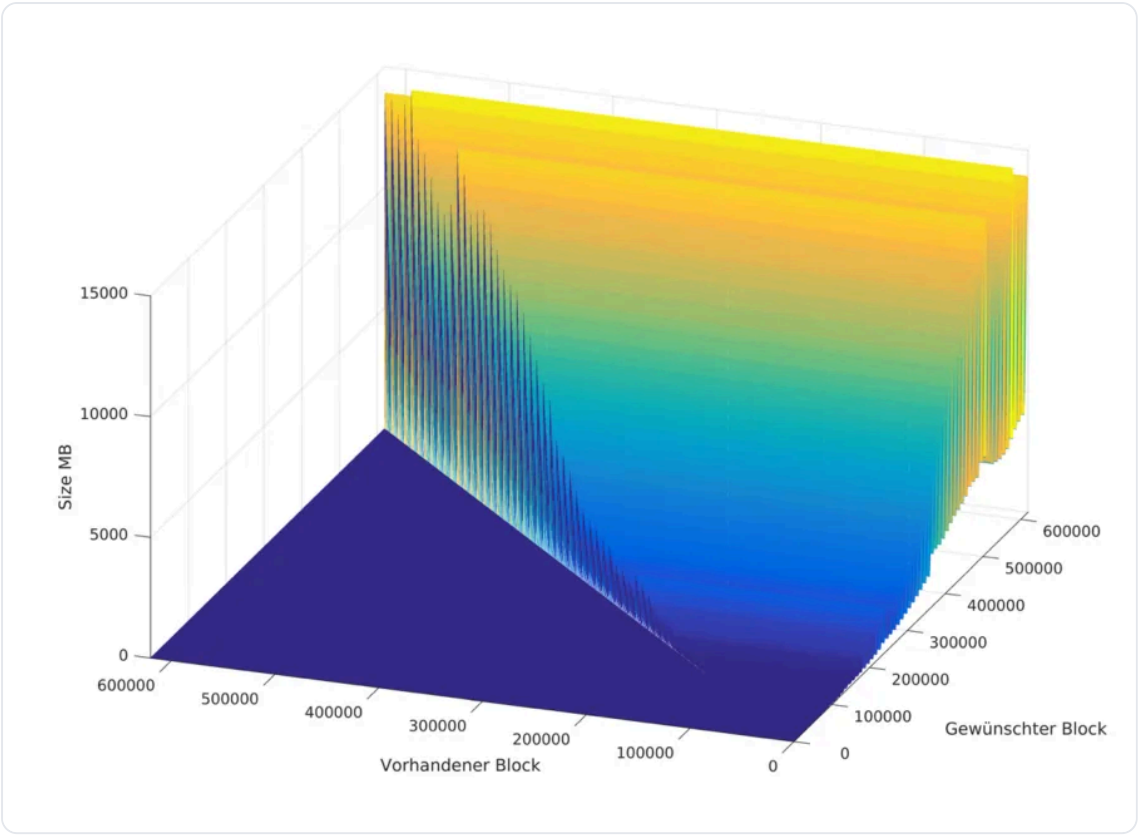
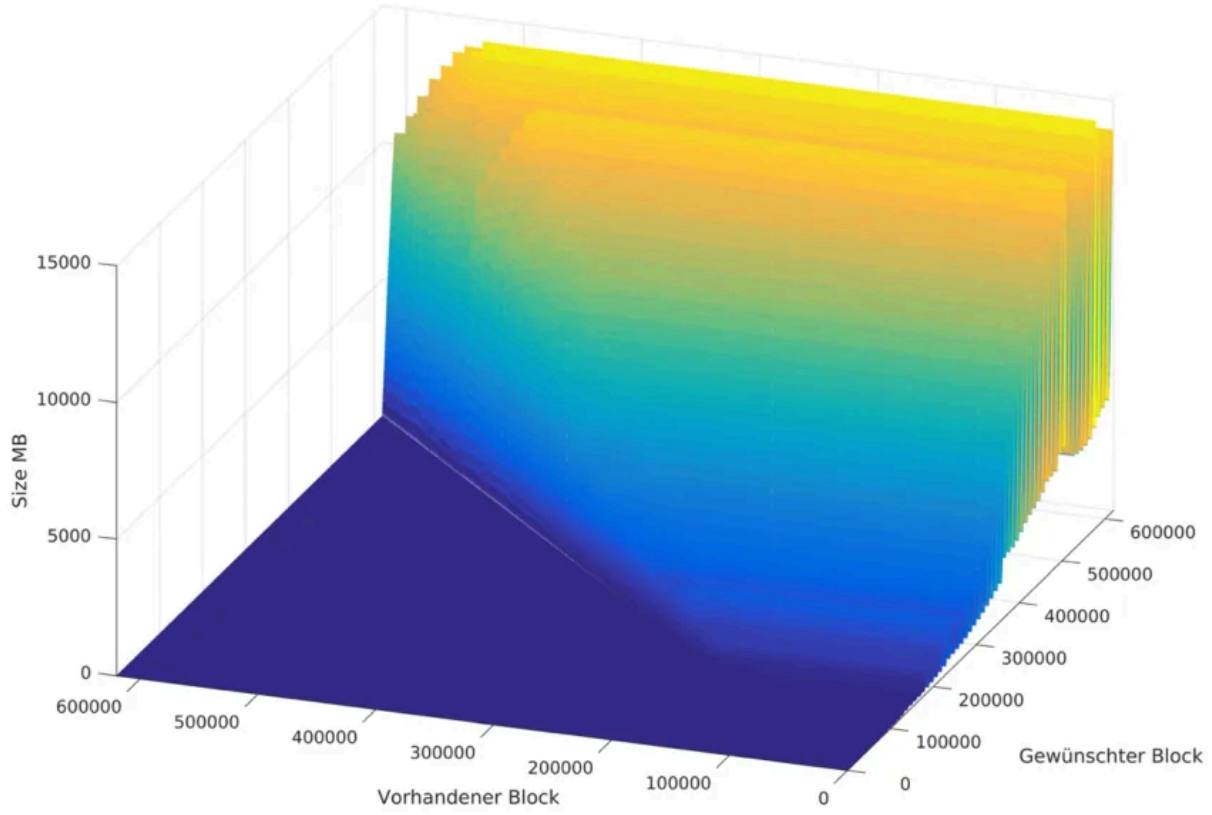
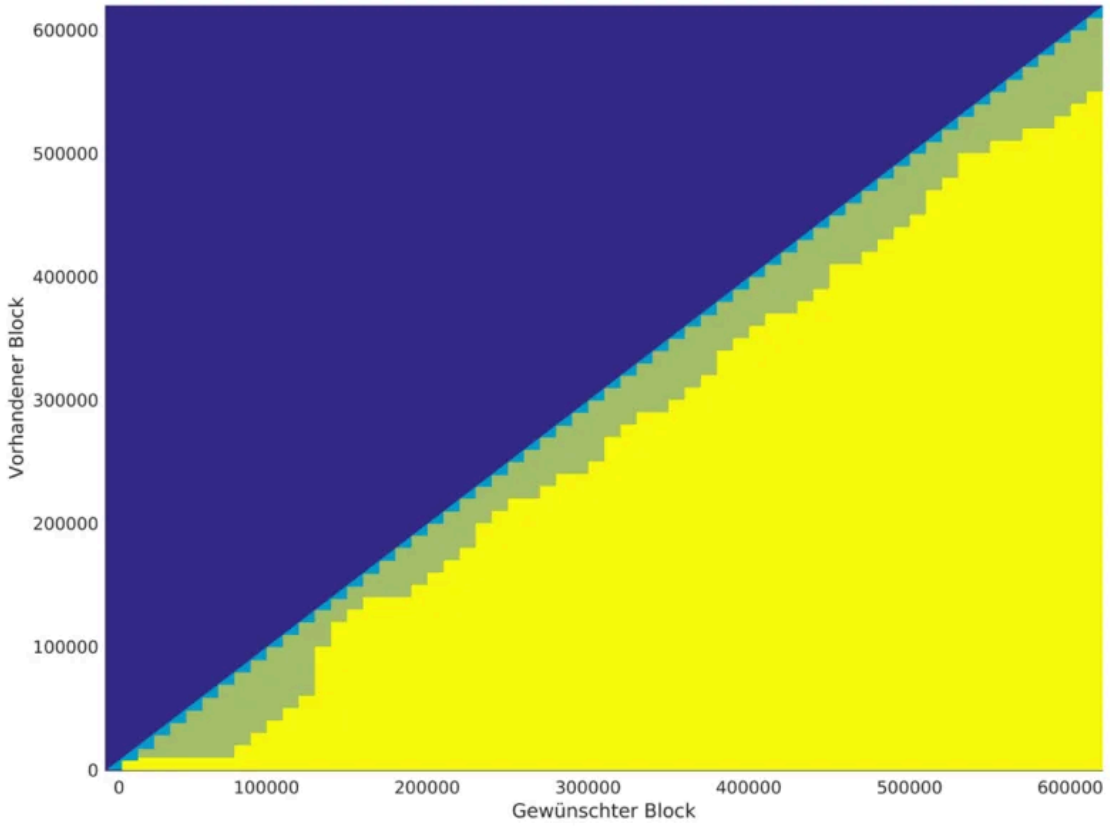


Abbildung 17: Zu ladende Datenmenge, wenn die beste der drei Synchronisierungsmethoden gewählt wird, bei einer Blockchain mit einem Snapshot-Block-Intervall von 10.000 Blöcken und einem Schwellwert von 100 Abbildung 18: Diese Abbildung zeigt, welche Synchronisierungsmethode optimal ist, in Abhängigkeit des lokal vorhandenen und des gewünschten Blockes. Es gibt vier verschiedene Bereiche. Beim dunkelblauen Bereich ist die lokal vorhandene Blockchain bereits mindestens so lange, wie die gewünschte. Beim hellblauen Bereich


15000
10000
5000
0
Size MB
600000
500000
400000
300000
200000
100000
0
Vorhandener Block
600000
500000
400000
300000
200000
100000
0
Gewünschter Block



ist die erste Synchronisierungsmethode, beim grünlichen Bereich die Zweite und beim gelben Bereich die Dritte optimal, bei einer Blockchain mit einem Snapshot-Block-Intervall von 10.000 Blöcken und einem Schwellwert von 100 Blöcken. Abbildung 19 und Abbildung 20 zeigen dieselben Daten wie die Abbildung 17 und Abbildung 18, jedoch mit einem Schwellwert von 1.000 Blöcken. Wieder zeigt sich, dass der größere Schwellwert nur zu einer geringen Verminderung der Datenmenge führt. Aus diesem Grund wurden die

Abbildung 19: Zu ladende Datenmenge, wenn die beste der drei Synchronisierungsmethoden gewählt wird, bei einer Blockchain mit einem Snapshot-Block-Intervall von 10.000 Blöcken und einem Schwellwert von 1.000

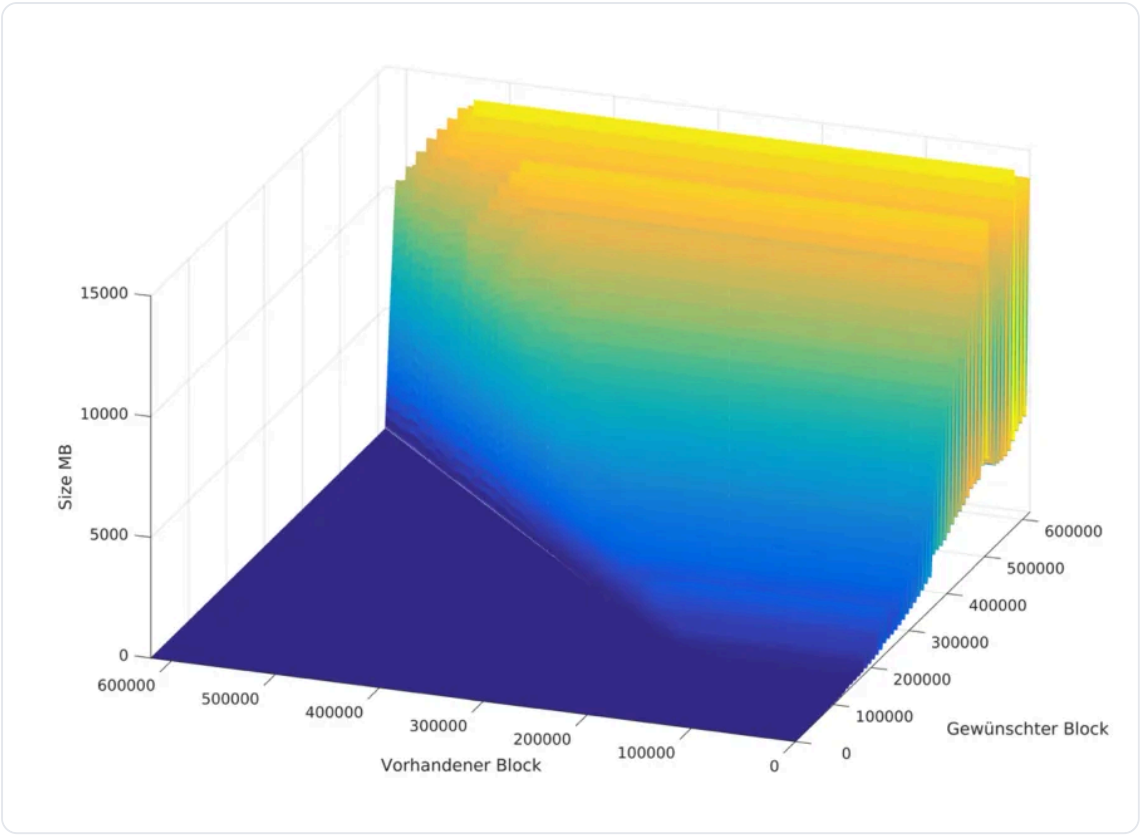


Abbildung 20: Diese Abbildung zeigt, welche Synchronisierungsmethode optimal ist, in Abhängigkeit des lokal vorhandenen und des gewünschten Blockes. Es gibt vier verschiedene Bereiche. Beim blauen Bereich ist die lokal vorhandene Blockchain bereits mindestens so lange, wie die gewünschte. Beim hellblauen Bereich ist die erste Synchronisierungsmethode, beim grünlichen Bereich die Zweite und beim gelben Bereich die Dritte optimal, bei einer Blockchain mit einem Snapshot-Block-Intervall von 10.000 Blöcken und einem Schwellwert von 1.000 Blöcken. Insgesamt zeigt sich, dass alle evaluierten Parameterkombinationen zu einer deutlichen Einsparung des Datenvolumens führen. Die Abbildung 21 und Abbildung 22 zeigen die prozentuelle Einsparung bei einem Snapshot-Block-Intervall von 1.000 und 10.000 Blöcken und einem Schwellwert von 100. Die Abbildungen für einen Schwellwert von 1.000 Blöcken sehen sehr ähnlich aus und wurden daher nicht abgedruckt. Bei Abbildung 22 ist der Einfluss des größeren Snapshot-Block-Intervalls gut erkennbar. Um dies noch zu verdeutlichen, zeigt Abbildung 23 die prozentuelle Einsparung bei einem Snapshot-Block-Intervall von 20.000 Blöcken und einem Schwellwert von 100 Blöcken. Wie erwartet hängt das Einsparungspotential bei größeren Snapshot-Block-Intervallen stärker davon ab, ob der gewünschte Block kurz nach einem Snapshot-Block liegt oder ob nach dem letzten Snapshot-Block auch noch viele Einzelblöcke geladen werden müssen.

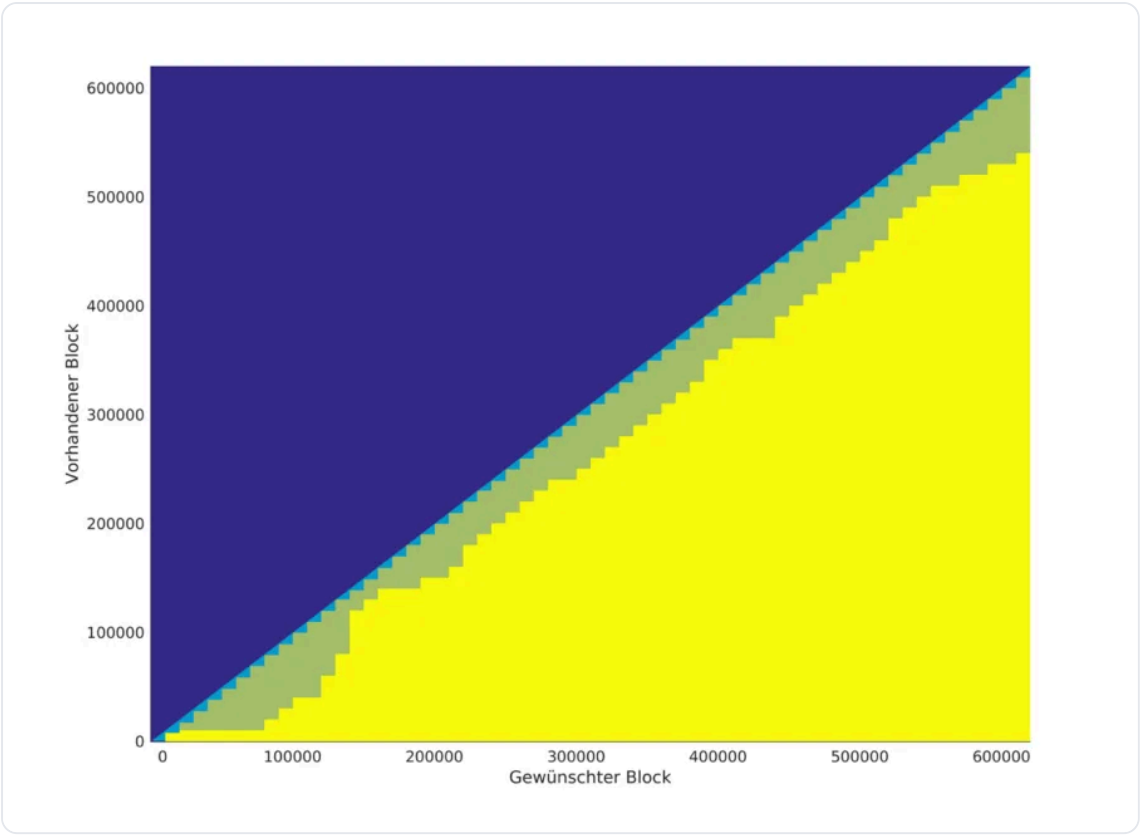


Abbildung 21: Eingesparte Datenmenge in Prozent, bei einer komprimierbaren Blockchain mit einem Snapshot-Block-Intervall von 1.000 Blöcken und einem Schwellwert von 100 Blöcken.

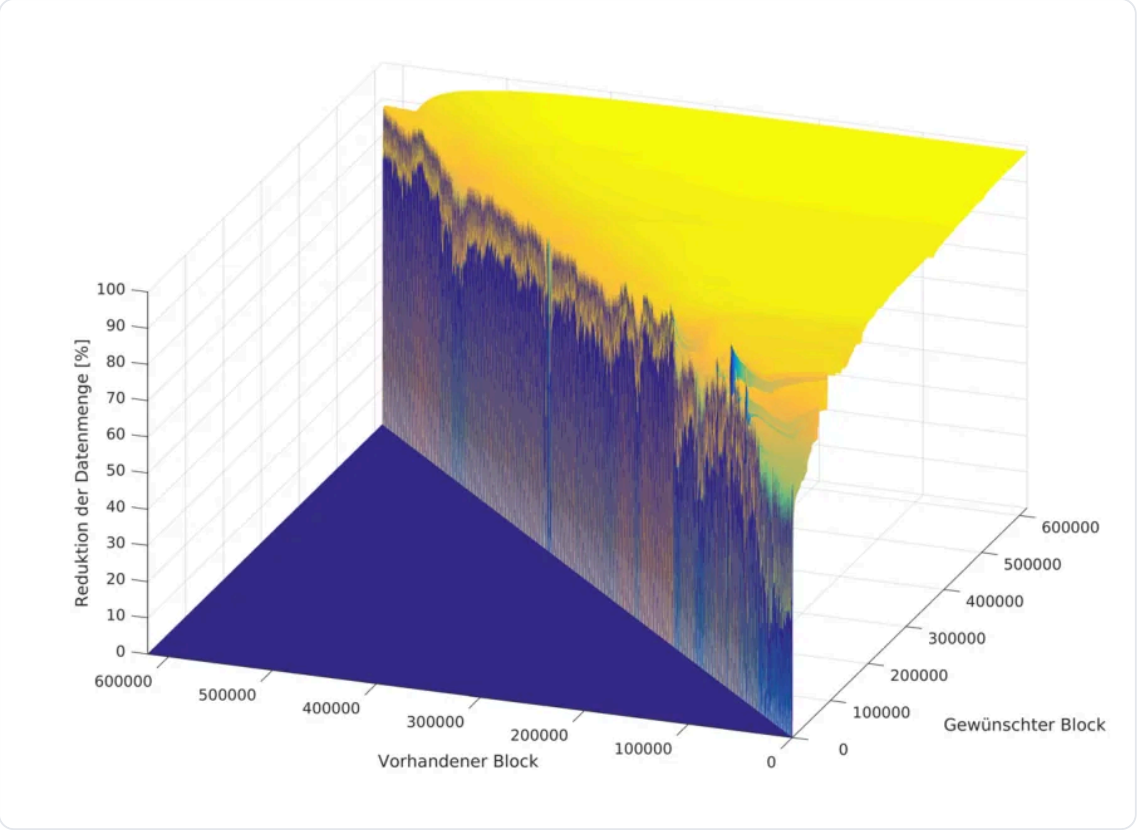
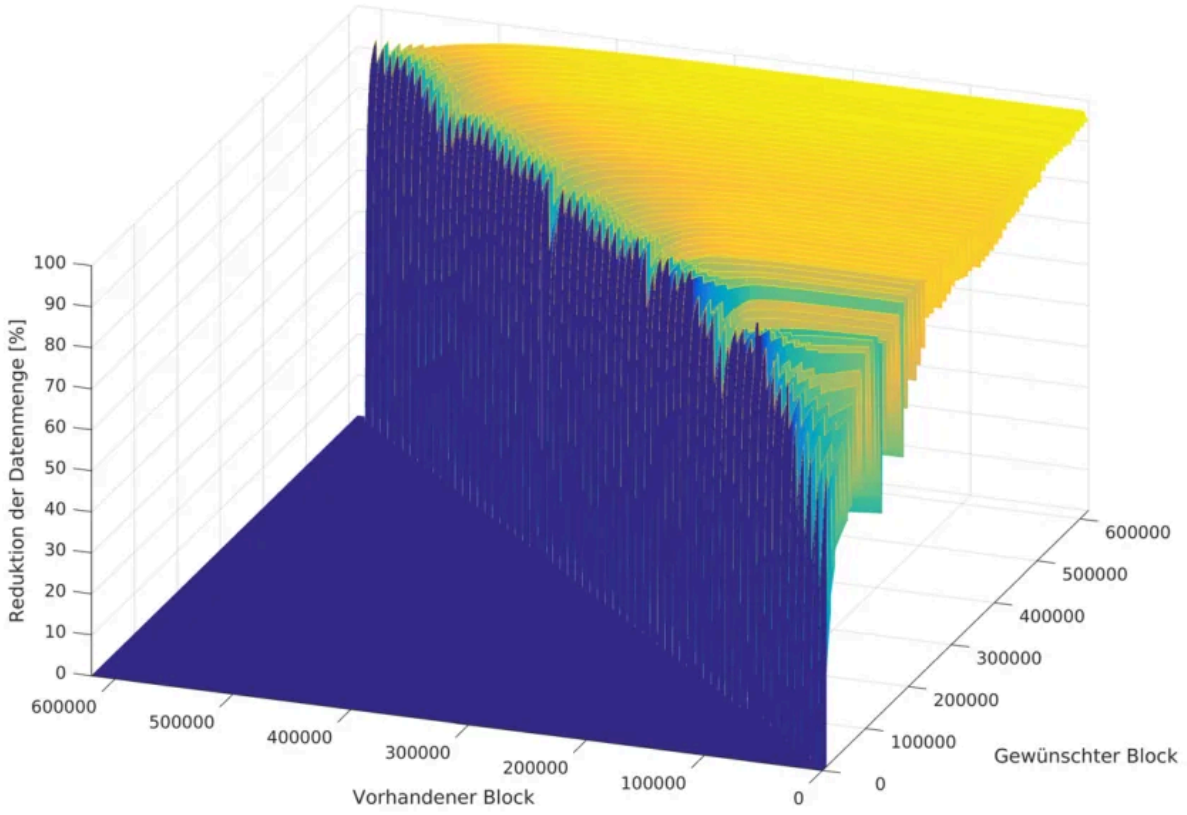
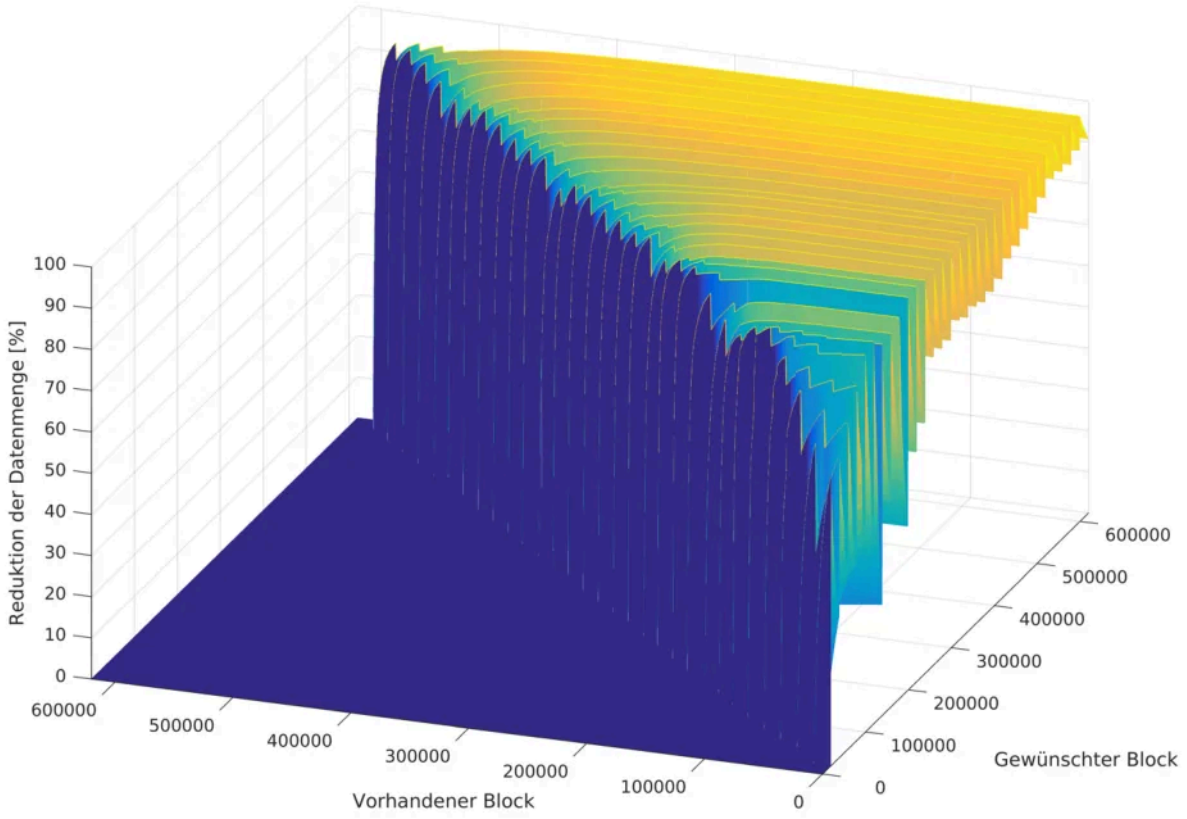


Abbildung 22: Eingesparte Datenmenge in Prozent, bei einer komprimierbaren Blockchain mit einem Snapshot-Block-Intervall von 10.000 Blöcken und einem Schwellwert von 100 Blöcken. Abbildung 23: Eingesparte Datenmenge in Prozent, bei einer komprimierbaren Blockchain mit einem Snapshot-Block-Intervall von 20.000 Blöcken und einem Schwellwert von 100 Blöcken.


Reduktion der Datenmenge [%]
100
90
80
70
60
50
40
30
20
10
0
600000
500000
400000
300000
200000
100000
0
Vorhandener Block
600000
500000
400000
300000
200000
100000
0
Gewünschter Block



5. Conclusio

Die in diesem Projekt vorgestellte Architektur einer komprimierbaren Blockchain ermöglicht Nodes eine wesentlich effizientere Synchronisierung mit dem Netzwerk. Das Einsparpotential hängt stark vom gewählten Snapshot-Intervall ab. Kleine Intervalle ermöglichen eine effizientere Synchronisation, bedeuten aber auch mehr Aufwand für Full Nodes und Miner. Full Nodes und Miner müssen zusätzlich die Snapshot-Blöcke speichern, wodurch sich ein erhöhter Speicherplatz Bedarf ergibt. Gleichzeitig müssen Miner öfter Snapshot-Blöcke erstellen wodurch sich ein Mehraufwand ergibt. Die Evaluierung zeigt, dass sich selbst mit großen Snapshot-Intervallen enorme Vorteile für neue und gelegentlich genutzte Nodes ergeben.

Referenzen

[1] A. Marsalek, „Komprimierbare Blockchain," 12 2020. [Online]. Available: https://technology.a sit.at/downloads/4118. [Zugriff am 12 2020].

[2] S. Nakamoto, „Bitcoin: A Peer-to-Peer Electronic Cash System," [Online]. Available:

https://bitcoin.org/bitcoin.pdf. [Zugriff am 2020].

[3] Blockchain.com, „Blockchain Size," [Online]. Available:

https://www.blockchain.com/charts/blocks-size. [Zugriff am 18 09 2020].

[4] statoshi.info, „Size of Serialized UTXO Set,“ [Online]. Available:

https://statoshi.info/dashboard/db/unspent-transaction-output-set. [Zugriff am 18 09 2020].

[5] Bitcoin, „Bitcoin Core version 0.11.0,“ 2015. [Online]. Available:

https://github.com/bitcoin/bitcoin/blob/v0.11.0/doc/release-notes.md. [Zugriff am 16 09 2020].

[6] Bitcoin Project, „Running A Full Node,“ [Online]. Available: https://btcinformation.org/en/full node#blocks-only-mode. [Zugriff am 16 09 2020].

[7] gmaxwell, „Blocksonly mode BW savings, the limits of efficient block xfer, and better relay,“ 26 02 2016. [Online]. Available: https://bitcointalk.org/index.php?topic=1377345.0. [Zugriff am 16 09 2020].

[8] Bitcoin Core, „Bitcoin Core,“ [Online]. Available: https://github.com/bitcoin/bitcoin. [Zugriff am 18 03 2020].

[9] A. Marsalek, T. Zefferer, E. Fasllija und D. Ziegler, „Tackling data inefficiency: Compressing the bitcoin blockchain,“ in *18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, Rotorua, 2019.