CISC 322/326 - Software Architecture
Assignment #3: Report
Wednesday, April 12, 2023

Enhancement Proposal for Bitcoin Core

Team Bingus II
Connor Decan - 19ctd3@queensu.ca
Allen Geng - 20lg5@queensu.ca
Kevin Jiang - 19shj1@queensu.ca
Yuanqi Liang - 18yl204@queensu.ca
Jayden Ting - 19tkt1@queensu.ca
Kevin Zhang - 19jyz5@queensu.ca

Abstract
In this report we will cover our proposed enhancement for Bitcoin Core. Our enhancement is a proposed feature to limit the amount of storage space that Bitcoin Core is able to take up on the user's computer. The report discusses two variations of the proposal, a checker and a limiter. Using a SAAM analysis, we analyzed each option's availability, compatibility, and customizability among other attributes. This helped us decide which implementation was preferable. There was also a stakeholder analysis to show how each version would affect the developers and users. After this, we discuss the ways each implementation will affect the previous architectures, and the ways we can test them. We then discuss two use cases and provide diagrams to demonstrate them. Finally, we show the potential risks each implementation poses.

Introduction
Previously, we derived the conceptual and concrete architectures for Bitcoin Core. This report will focus on how those architectures can be improved by introducing a new feature to the system. It also shows how that new feature interacts with the overall existing architecture.

In this report we will first explain the newly proposed feature; what it does and how it works. Alternatives for realizing the proposed enhancement are presented by demonstrating two possible implementations on how the enhancement may be achieved. After the proposal, a SAAM analysis will be conducted to compare the implementations on various attributes of quality, and the analysis will determine which implementation is better.

After the analysis is completed, we will demonstrate in more detail how the new feature will interact with the overall architecture of Bitcoin Core. Then, we will use two sequence diagrams to demonstrate how the implementation will be used based on two user scenarios. Then, we will show how the new enhancement will be tested based on certain requirements that need to be fulfilled. Finally, we will conclude the report by discussing potential risks that the new feature might incur, and the lessons learned from coming up with this new proposed feature.

Enhancement
When thinking about bitcoin, one of the main problems that people think of is how resource intensive it can be. Oftentimes people think of the issue of processing power, such as how people purchase high-end hardware for the purposes of mining bitcoin, but another overlooked issue is that of storage space. As the amount of bitcoin belonging to a user increases, so does the amount of storage that it takes up, and this can be an especially big problem for people with lower amounts of storage space on their devices. From this, Bitcoin Core should have a contingency to make sure that a user is able to make use of the software without taking up too much storage.

Our proposed enhancement is a module for Bitcoin Core that acts as a storage limiter. Since the limiter should be modifiable, it serves as a solution that would help out the users that need it to compensate for their low amounts of storage, while also able to be ignored by the users who don't need it, as they can just set it to be unlimited. Here are two possible implementations for the proposed module:

The first implementation is a disk space limiter. The user will allocate a specific, predetermined amount of storage that they are willing to let Bitcoin Core access, and the app will make checks to ensure that the amount of blocks stored on the disk does not overtake that amount. If the limit is reached, Bitcoin Core will halt the mining component.

The second implementation is a disk space checker. Rather than the user implementing a specific amount of space for Bitcoin Core to use, the software will instead assume that the entire disk is usable and keep mining until it is full. However, while mining, Bitcoin Core will make checks to the disk space to make sure that it does not go over the maximum capacity of the user's disk.

Both implementations have the same changes to the architecture that will have to be made. Specifically, the new component will have to depend on the user's disk space, since it will have to frequently check the available storage, and it will also have to have dependencies with the Storage Engine, to see how much storage Bitcoin Core is using. The Miner component will depend on the Storage Limiter or Checker to stop mining any further if the maximum allowed storage is reached. In regards to the Storage Engine, the dependency would depend on the chosen implementation. For the storage limiter, the dependency is required since the program needs to know how much space it is taking up in order to stay within the limit. For the disk space checker, the program may not need this dependency since the program only needs to know how much space is remaining on the device, which can be learned from the operating system.

SAAM analysis

Shareholders that are involved

Both implementations share the same shareholders, which are the developers and the users.

- Developers
  Bitcoin core developers who will be implementing this enhancement into their code. The most important NFRs for developers would be, maintainability, testability, and integration.
- Users
  The end-user of bitcoin, those who store bitcoin in their local storage and may find themselves running out of space. The most important NFRs for the users would be, availability, performance, scalability, compatibility, reliability, and customizability.

Some NFRs that are also important but are irrelevant to both implementations are: security, privacy, usability, support.

Effects of the enhancement on NFRs

NFR	Disk Space Checker	Disk Space Limiter
Availability	Will be available as long as there is enough storage to meet the minimum requirement.	May not be available even if there is enough storage for the minimum requirement depending on the specific allocation.
Compatibility	Should be compatible with almost all situations, unless the specific environment does not allow a software to check for disk space.	Depending on the amount of storage and RAM allocation, it may have issues with other RAM hoarders like chrome, adobe, teams, or video games.
Customizability	Does not apply to the checker since there is no need for customization. If there really is a need, you can do a check of "have at least _ of space left over after this process"	The user may change the allocated resources as they wish, this may be a little challenging for those users without any knowledge on how hardware resources work.
Integration	Can be easily integrated.	Can be easily integrated, but the developers will need to think about making adjustments based on the specific hardware.
Maintainability	Easy to maintain.	Easy to maintain.
Performance	Low to no impact on performance, this would only run during installation or when it is needed to use any hardware storage.	Possible impact on performance. Depending on the size of the limit, bitcoin core performance could go down or it may be hoarding too much resource.
Reliability	If there is a very small amount of storage left, there may be issues if bitcoin core and another application both are trying to write to the same space, known as a resource contention.	High level of reliability. If the program can run it means the storage cannot be accessed by anything else.
Scalability	Good scalability, with increasing size of data needed to write to storage, it simply only needs to	If there is more data than the amount of storage allocation, then the program will stop.

	check for a larger amount.	
Testability	Easy to test.	Easy to test.

Stakeholder analysis

	Developers	
	Pros	Cons
Disk Space Checker	Faster implementation than the limiter.	There will have to be an estimation on how much space a certain action requires, if such things do not exist already.
Disk Space Limiter	Implementing this may help the developers identify any unexpected temporary usages with the current storage system.	This will require more time to implement than the checker. Will have to create a UI for the user to allocate. Will have to find a default amount and a recommendation for the user.
	Users	
	Pros	Cons
Disk Space Checker	The user does not have to worry about configuration.	The disk may be completely filled by bitcoin without the user noticing.
Disk Space Limiter	Disk storage will never be filled more than the allocation.	The user may have to adjust the limit frequently. If the limit is too low, it may impact the functionality of the application. Most likely a new user won't know what is an optimal storage.

Comparison Conclusion

After comparing advantages and disadvantages of both implementations for our enhancement, relative to the NFRs and the stakeholders, we think that the disk space checker will be the better option. It does as well as the limiter or better on almost all categories, there is only one major drawback, which is the disk may be filled if left for too long without attention, this can be resolved by giving a simple prompt to the user periodically.

Testing
If a storage limiter enhancement is implemented, its interactions between different features and components in the architecture can be tested by logging interactions to a console window to see if the correct action is taken for each.

Both the Disk Space Limiter and the Disk Space Checker would first interact with the Miner component to calculate the size of the next block that needs to be mined; it then logs the size of the block.

The Disk Space Limiter would then interact with the Storage to check if there is still enough space in the disk to mine the next block, the Disk Space Checker would also interact with the storage but checks for space in the assigned storage instead of the entire disk. This interaction can be logged by printing the remaining memory space available for Bitcoin Core and the size of the next block; it then prints out if it should continue mining. If there is enough available space, the storage limiter would interact again with the Miner component to continue mining.

If there is not enough available space, then the storage limiter may interact with the App & GUI component to output a warning or error message so that the user is aware that mining has stopped due to lack of space. This would be tested by logging the content of the output message and seeing if a matching message is outputted in the GUI.

Effects on Architecture
Storage Limiter -> Storage Engine
The Storage Limiter implementation relies on the Storage Engine to ensure that the entirety of the storage space used up by Bitcoin Core is within the limits set by the user.

Miner -> Storage Limiter/Storage Checker
The Miner would be required to conform to the memory limits set by the limiter or checker whenever it mines a new block.

App & GUI -> Storage Limiter
The App & GUI component should allow the user to change the storage limits placed on the node.

RPC & API -> Storage Limiter
The RPC & API component should allow the storage limits placed on the node to be changed.

Storage Limiter/Storage Checker -> Apps & GUI

The Storage Limiter or Checker component should be able to provide warnings about the status of storage usage relating to Bitcoin Core to the GUI.

Storage Limiter/Storage Checker -> RPC & API
The Storage Limiter or Checker component should be able to provide warnings about the status of storage usage relating to Bitcoin Core to the RPC & API component.

Use Cases

Miner requesting disk space with no pre-allocation

For the implementation of storage checker, where no limit of disk space was previously allocated, this is the use case where a miner is requesting space.

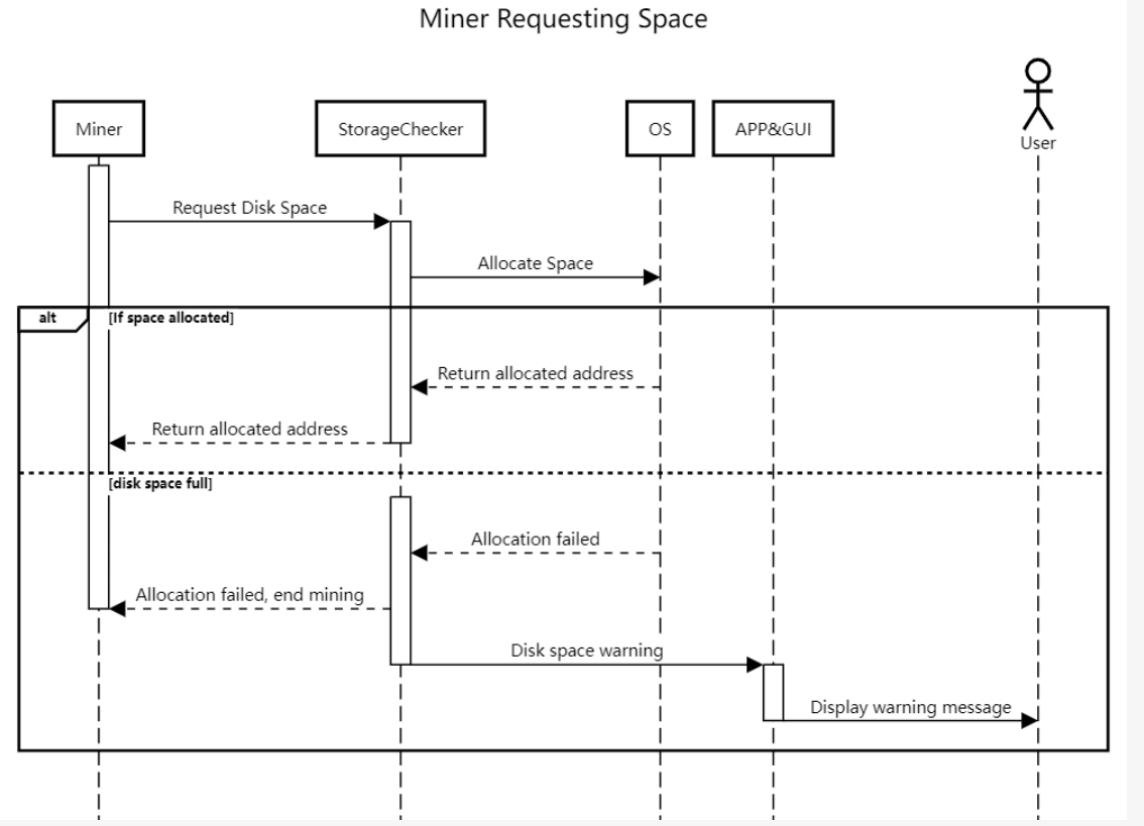

Figure 1. Sequence Diagram for Use Case 1

When a Miner is generating hashes, it will be necessary to allocate it with space. The Miner will first send a request to the storage checker to request such space on the disk. The request will be forwarded by the storage checker to the OS. If the OS is able to allocate the amount of space requested, the storage checker will send the allocated address to the miner, and allow it to continue mining. Else, the storage checker will send a message to the miner, suspending it from mining. In addition, it will send a warning message to the APP GUI component, the message will be displayed by the App as a notification to the user. If a storage limiter is implemented, then the miner only needs to

confirm with the limiter. This would save the time from communicating with OS on every request

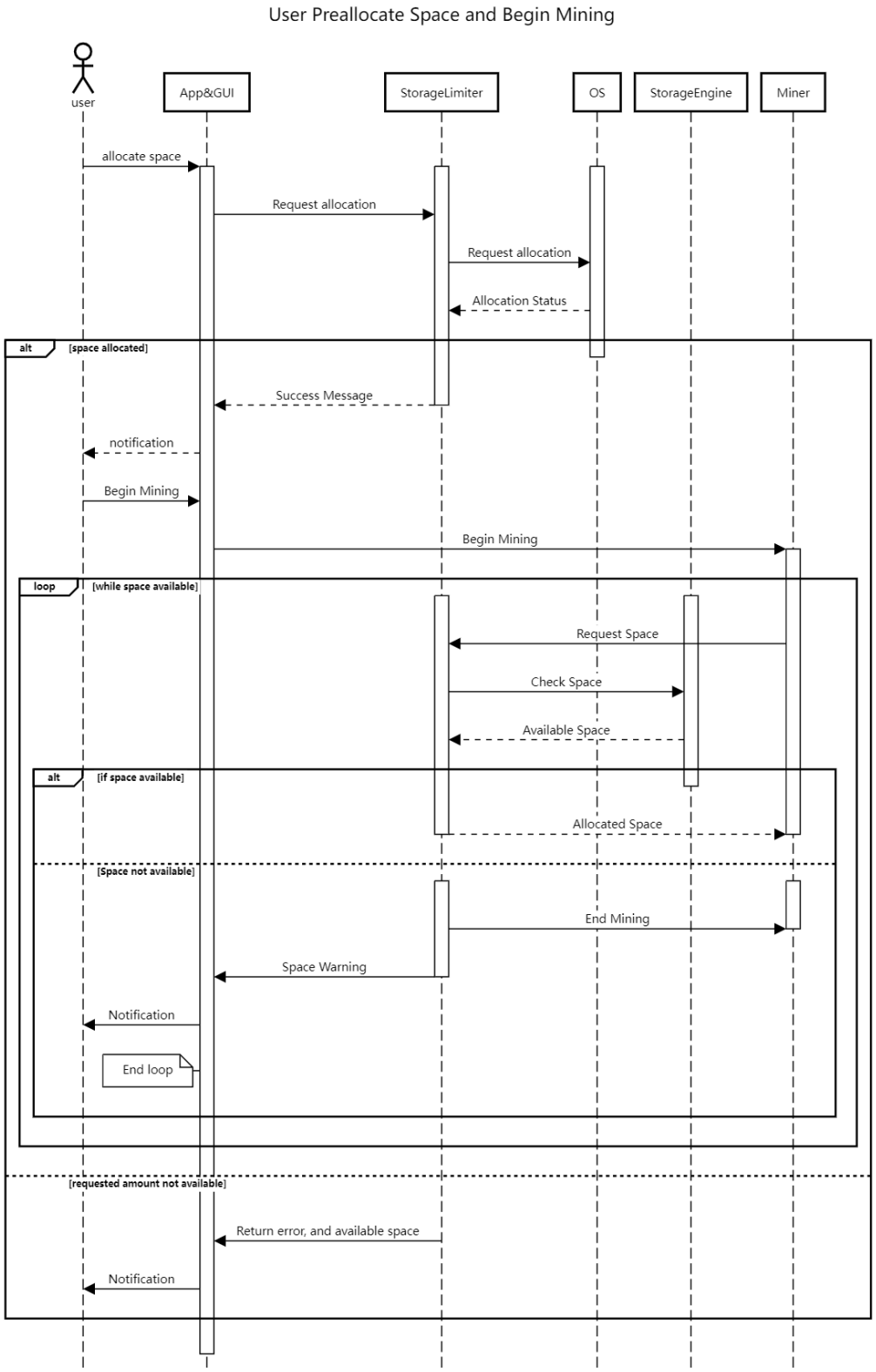

Figure 2 :Sequence Diagram for Use Case 2

2. User allocate space using storage limiter and begin mining

In this use case, the user will be allocating space on the drive and begin the mining process. The user will be interacting with the App, and sending the request with the amount of space desired to the Storage Limiter. The Limiter will forward this request to the OS. If space can not be allocated, then the Limiter will send the message to App, with the current available space on disk so the user can select another value. If the space can be allocated then the user can begin mining. Each time the Miner needs storage space, it will communicate with the Storage Limiter. The Limiter will check how much of the preallocated space has been used by communicating with the Storage Engine. If there is still space available, it will return the address to the miner and allow it to continue mining. Else, it will halt the mining process and send a warning to the App to be displayed.

Potential Risks & Limitation

Approach 1: Disk space limiter

The Disk Space Limiter requires the user to assign an appropriate amount of storage space to Bitcoin Core. Inadequate or excessive allocation of storage space may result in errors. If under-allocated, there may not be enough space to store the necessary blockchain data, affecting the user's ability to effectively mine Bitcoin as it may prevent the program from mining or running correctly. Over-allocation may cause Bitcoin Core to utilize unnecessary resources, decrease the performance of the computer or other applications running on the computer. It may further cause Bitcoin Core to fill up quickly, using up the allocated space sooner than expected. Disk space limiters also have security and performance problems. Insecure implementations of disk space limiters can be exploited by attackers to induce program failures or blockchain corruption, resulting in financial loss to users. Another issue is security, insecure implementations of disk space limiters can be exploited by attackers to induce program failures or blockchain corruption, resulting in financial loss to users. Moreover, in terms of performance, disk space limiters may affect performance by stopping the mining component when the limit is reached.

Approach 2: Disk space checker

The addition of a disk space checker presents only one potential risk. The disk space checker may not be sufficient in avoiding capacity issues if the user fails to assign more storage space or perform regular cleanup in a timely manner. With minimal remaining disk space, if multiple applications attempt to write to the same limited space, it will cause resource contention, leading

to performance degradation or system crashes. Furthermore, if the disk space checker detects that the disk is full and continues to write data, it may also engage in competition with other processes. However, Disk space checkers do have some limitations. Disk space checkers have security issues. Disk space checkers may be vulnerable to attacks that manipulate or exploit disk space monitoring programs. For example, a malicious user may attempt to manipulate the monitoring system to create the illusion of available space, which could lead to a system crash or instability. Meanwhile, for performance, regular and frequent disk space checks can lead to increased system overhead during system usage.

Conclusion

Through the SAAM analysis, we concluded that the better implementation would be the Space Checker as it has a lower impact on performance that could be used for mining and a reduction in the cost for implementing the feature. The main drawbacks to the implementation is that it allows the drive to be filled, but is better than the downsides to the disk space limiter as discussed. We were also able to come up with two use cases to show the interactions with the rest of the system.

Lessons Learned

To comprehend the project management side of a project requires the analysis of new systems added to the system throughout the lifespan of its deployment. Adding a Storage Limiter or a Storage Checker requires checking the interactions of the component to other components to ensure that any component that needs access to storage space does not bypass the restrictions placed on it by the Limiter or Checker. The component also requires the analysis of the needs and wants of stakeholders, as modifications to the project should maintain compatibility with existing use cases. New use cases were also discussed in order to comprehend the interactions between the user, the rest of the system, and the new components added. Analyzing the risks and limitations of the component was also important to deciding the better implementation.