Grouped Matrix Clocks with Reduced Complexity for Distributed Synchronization

Khizer Tariq
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
mtariq.bscs21seecs@seecs.edu.pk

Hasib Aslam
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan
haslam.bscs21seecs@seecs.edu.pk

ABSTRACT

Logical clock synchronization is a crucial aspect of distributed systems, enabling the correct ordering of events and maintaining causal relationships. The matrix clock algorithm, while effective, suffers from quadratic communication overhead as the number of processes increases due to its $n \times n$ matrix size representation. This paper introduces a novel group-based matrix clock algorithm that reduces this overhead by exploiting communication locality patterns among processes. The key idea is to partition processes into multiple groups based on the frequency of communication. Processes within a frequently communicating group maintain a small intra-group matrix clock, while each group maintains a compact group-level matrix clock summarizing the group's collective knowledge. Inter-group communication is timestamped using these group matrix clocks, reducing the overhead compared to fully replicated global matrix clocks. This approach minimizes the timestamp size for frequent intra-group communication while preserving sufficient causal information for accurate event ordering via the group-level clocks. Theoretical analysis demonstrates significant reductions in space and communication overhead compared to the original matrix clock algorithm. The proposed group matrix clock algorithm retains the powerful causality tracking capabilities of matrix clocks while relaxing the lower bound for the space complexity to $\Omega(N^2)$.

General Terms

Distributed Computing, Algorithms, Clock Synchronization

Keywords

Distributed Computing, Logical Clocks, Parallel and Distributed Computing, Synchronization, Causal Ordering

1. INTRODUCTION

Distributed systems have become ubiquitous in modern computing environments, enabling the coordination and collaboration of multiple processes or nodes to achieve complex tasks. However, the inherent concurrency and lack of a global clock in these systems pose significant challenges in maintaining consistent event ordering and causal relationships among processes. Logical clock synchronization techniques play a crucial role in addressing these challenges, ensuring the correct execution of distributed algorithms and applications.

One of the widely adopted logical clock synchronization algorithms is the matrix clock, concepts introduced by Mattern [5] and an extended version introduced by [8]. Matrix clocks effectively capture causality information and logical timestamps for events in distributed systems. Unlike scalar clocks, which can only provide a partial order of events, matrix clocks establish a total order, allowing for the accurate reconstruction of the system's execution history. However, the traditional matrix clock algorithm suffers from a significant limitation: its communication overhead grows quadratically with the number of processes in the system. Each process maintains an $n \times n$ matrix, where n is the total number of processes, leading to increased space requirements and message sizes as the system scales.

In real-world distributed systems, communication patterns often exhibit locality, where certain subsets of processes communicate more frequently with each other than with others. This phenomenon is particularly prevalent in large-scale systems with heterogeneous workloads, such as cloud computing environments, peer-to-peer networks, and distributed databases. Exploiting this communication locality can lead to significant optimizations and performance improvements.

This paper presents a novel group-based matrix clock algorithm that aims to reduce the communication overhead associated with traditional matrix clocks while preserving their powerful causality-tracking capabilities. The key innovation is the partitioning of processes into multiple groups based on their communication frequencies. Processes within a frequently communicating group maintain a small intra-group matrix clock, while each group maintains a compact group-level matrix clock summarizing the group's collective knowledge. Inter-group communication is timestamped using these group matrix clocks, reducing the overhead compared to fully replicated global matrix clocks.

By leveraging communication locality, the proposed algorithm reduces the timestamp size for frequent intra-group communication, leading to asymptotic space and communication overhead reductions compared to the original matrix clock algorithm. Empirical evaluation confirms the optimization benefits, especially as system

scale and communication locality increase. The group matrix clock algorithm preserves the ability to accurately order events and maintain causal relationships, making it suitable for a wide range of distributed applications and systems with heterogeneous communication patterns.

2. LITERATURE REVIEW

Logical clock synchronization has been extensively studied in distributed systems, with various techniques proposed to maintain consistent event ordering and causality tracking. One of the earliest and most fundamental approaches is Lamport's logical clocks [3], which introduced the concept of using scalar logical timestamps to establish a partial order of events in a distributed system. While Lamport clocks capture the basic happened-before relationships, they are insufficient for reconstructing the complete causal history of events.

Vector clocks, introduced by Mattern [6] and Fidge [2], extend the concept of logical clocks to provide a total order of events. Each process maintains a vector of logical timestamps, one for each process in the system. Vector clocks accurately capture causality by tracking the dependencies between events across processes. However, the size of vector timestamps grows linearly with the number of processes, leading to increased communication overhead in large-scale systems.

Several optimizations and variants of vector clocks have been proposed to address this overhead. The Plausible Clock Condition (PCC) [8] technique seeks to reduce the size of vector timestamps by exploiting the sparse nature of causal dependencies in many distributed applications. Despite these improvements, these approaches still require maintaining and transmitting vectors proportional in size to the number of processes.

Matrix clocks, introduced by Mattern [5] and further formalized by [8], represent a sophisticated logical clock synchronization technique that captures both causality and concurrency information. Each process maintains an $n \times n$ matrix, where n is the number of processes, allowing for the reconstruction of the complete system execution history. Matrix clocks establish a total order of events and enable accurate causality tracking, making them suitable for a wide range of distributed applications.

However, as noted in the introduction, the traditional matrix clock algorithm incurs quadratic communication overhead due to the $n \times n$ matrix size, which can become a significant limitation in large-scale distributed systems.

The group-based matrix clock algorithm proposed in this study addresses the scalability and communication overhead challenges of traditional matrix clocks by leveraging communication locality patterns in distributed systems. By partitioning processes into groups and maintaining separate intra-group and inter-group matrix clocks, the algorithm reduces the timestamp size and communication overhead, particularly in systems with heterogeneous communication patterns. This approach combines the powerful causality tracking capabilities of matrix clocks with the efficiency benefits derived from communication locality.

3. PROPOSED ALGORITHM

The group-based matrix clock algorithm is designed to reduce the communication overhead associated with traditional matrix clocks while preserving their powerful causality-tracking capabilities. The central idea is to partition processes into multiple groups based on their communication frequencies, leveraging the observation that, in many distributed systems, certain subsets of processes communicate more frequently with each other than with others.

3.1 System Model and Assumptions

The distributed system under consideration consists of N processes, denoted by $P = \{p_1, p_2, \ldots, p_N\}$. These processes are partitioned into N_g groups, represented by $G = \{g_1, g_2, \ldots, g_{N_g}\}$, based on their communication patterns. The number of processes in each group is denoted by k_i, such that:

$$\sum_{i=1}^{N_g} k_i = N \tag{1}$$

where *i* represents the group index. The values of N_g and k_i vary depending on the algorithm used for group formation. Using an arbitrary group formation algorithm, it is assumed that the resulting k_i values are sampled from a distribution *D*, with mean $Exp(k)$ and variance $Var(k)$.

It is further assumed that intra-group communication occurs more frequently than inter-group communication, and processes within the same group exhibit higher communication locality.

3.2 Intra-group Matrix Clocks

Within each group g_i, processes maintain a traditional $k_i \times k_i$ matrix clock, following the rules and operations formally defined by [8]. Each process p_j in group g_i manages a unique matrix, where entry at row x and column y represents the number of events from process p_x that are known to have occurred before the current state of process p_y, according to p_j's knowledge.

Intra-group communication adheres to the standard matrix clock synchronization protocol, ensuring accurate causality and concurrency tracking within each group.

3.3 Inter-group Matrix Clocks

To facilitate communication between groups, each group g_i maintains a compact group-level matrix M_i of size $N_g \times N_g$, every member of the group keeps an instance of this matrix. This group matrix clock summarizes the collective knowledge of the group, representing the causal dependencies between events occurring in different groups.

The group matrix M_i is maintained as follows:

(1) The initial state of all instances of M_i is set with all entries initialized to 0.

(2) When a process p_i^m (m^{th} process in i^{th} group) communicates with a process p_j^n, it first updates its local instance of the group matrix clock M_i using the latest information from other members of its group (See Section 3.4).

(3) The message sent by p_i^m to p_j^n is piggybacked with the latest version of M_i, summarizing the information about the inter-group events that have happened, as per the collective knowledge of the members of a group g_i

(4) Upon receiving the message, p_j^n first updates its local instance of the group matrix clock M_j using the latest information from other members of its group g_j (using the same protocol as was used by p_i^m before sending, Section 3.4).

(5) p_j^n accepts the message and updates M_i based on the comparison between M_i and M_j following the rules defined by [8].

3.4 Accessing Group Matrix Clock Information

To ensure that each process maintains an up-to-date view of its group's collective knowledge, the following protocol is employed:

(1) When a process p_i^m needs to communicate with p_j^n, either sending or receiving, (where $i \neq j$, meaning the processes belong to different groups), it broadcasts a request to all members of g_i for the latest group matrix clock information.
(2) Upon receiving the request, each process in g_i responds with its local instance of the group matrix clock $M_{i,k}$ (instance of matrix clock of group g_i from k^{th} process p_i^k).
(3) Process p_i^m collects the responses and updates its local instance of group matrix $M_{i,m}$ by taking the element-wise maximum of $M_{i,m}$ and all other received instances.
(4) After updating $M_{i,m}$, the process p_i^m proceeds with inter-group communication using the updated group matrix clock timestamp.

4. COMPLEXITY ANALYSIS

4.1 Space Complexity

In traditional matrix clocks, the space complexity for N processes is $O(N^3)$ as well as $\Omega(N^3)$, as each process maintains an $N \times N$ matrix clock, resulting in N matrices of size $N \times N$. In the proposed group-based matrix clock algorithm, the total space complexity is contingent on each k_i and N_g and we show that the lower bound for space complexity in the resulting algorithm is reduced significantly, to $\Omega(N^2)$, whereas upper bound remains, $O(N^3)$, as the original algorithm emerges as a special case when N_g is 1. The space consumption is analyzed below:

4.1.1 Intra-group Matrix Clocks

(1) Each group g_i consists of k_i processes.
(2) Each process in a group maintains a $k_i \times k_i$ matrix clock.
(3) The total space required for intra-group matrix clocks is $\sum_{i=1}^{N_g} k_i^2$. If $Exp(k)$ represents the expectation value of the number of groups then:

$$Exp(k^2) = Var(k) + (Exp(k))^2 \tag{2}$$

Therefore the space required is $N \times Exp(k^2)$.

4.1.2 Inter-group Matrix Clocks

(1) Every member of each group g_i maintains a group matrix clock of size $N_g \times N_g$.
(2) There are N processes in total, hence N group matrix clocks each of size $N_g \times N_g$ are maintained.
(3) The total space required for inter-group matrix clocks is $N \times N_g^2$.

Combining both components, the total space required is:

$$\textit{Space} = N \times \mathrm{Exp}(k^2) + N \times (N_g)^2$$

$$\textit{Space} = N \times (\mathrm{Exp}(k^2) + N_g^2)$$

using Equation (2), we can rewrite the above relation as:

$$\textit{Space} = N \times (Var(k) + Exp(k)^2 + N_g^2) \tag{3}$$

From the above equation, it is clear that the complexity depends upon the *expectation* and *variance* of random variable k obtained from distributing D and *the total number of groups* formed.

In **Theorem-1** we showed that the lower bound of space complexity, using any group formation algorithm is $\Omega(N^2)$ where $N_g = \sqrt{N}$ while $Var(k) = 0$ and $Exp(k) = \frac{N}{\sqrt{N}}$, the average group size.
In **Theorem-2** we showed that the upper bound of space complexity is $O(N^3)$ where $N_g = 1$ while $Var(k) = 0$ and $Exp(k) = N$.

4.2 Time Complexity

4.2.1 Intra-group Communication

(1) **Sending and Receiving Messages:** The process of sending and receiving intra-group messages involves updating the $k_i \times k_i$ matrix clocks. The time complexity for these operations is $O(Exp(k^2))$.

4.2.2 Inter-group Communication

(1) **Message Sending and Timestamps:** When a process sends an inter-group message, it updates its group matrix clock and timestamps the message, which is $O((N_g)^2)$.
(2) **Broadcasting Requests:** Broadcasting a request for the latest group matrix clock information to all other members of the group has a time complexity of $O(Exp(k))$.
(3) **Collecting Responses:** Collecting responses and updating the local instance of the group matrix clock involves $O(Exp(k) \times (N_g)^2)$ operations.

Combining these operations, the time complexity for intra-group communication remains $O(Exp(k^2)$, while the time complexity for inter-group communication is dominated by $O(Exp(k) \times (N_g)^2)$ + $O(Exp(k))$ due to the need to update and synchronize the group matrix clocks.

4.3 Communication Overhead

4.3.1 Intra-group Communication Overhead

(1) The overhead for intra-group communication remains consistent with traditional matrix clocks, involving the transmission of $k_i \times k_i$ matrix clock information within the group.

4.3.2 Inter-group Communication Overhead

(1) The overhead for inter-group communication is reduced due to the use of compact group matrix clocks. Each inter-group message is timestamped with a $N_g \times N_g$ matrix clock, which is significantly smaller than an $N \times N$ matrix clock.

4.4 Benefits and Trade-offs

(1) **Reduced Space and Communication Overhead:** The primary benefit of the proposed algorithm is the significant reduction in space and communication overhead. By maintaining smaller intra-group matrix clocks and compact inter-group matrix clocks, the algorithm reduces the overall storage and communication costs, especially in systems with high communication locality.
(2) **Increased Complexity for Synchronization:** The algorithm introduces additional complexity for synchronizing group matrix clocks and maintaining causality during inter-group communication. Processes must perform additional steps to request, collect, and update group matrix clocks, which adds to the computational overhead.

Table 1. Comparison Table for Space Complexity of Proposed Solution.

No. of Processes (N)	**No. of Groups** (G)	**No. of Processes within a group** (k')	**Matrix Clocks** (N^3)	**Proposed Algorithm** $N(N^2)$
4	2	2	64	32
9	3	3	729	162
25	5	5	15625	1250
100	10	10	1000000	20000

(3) **Scalability:** The proposed algorithm is more scalable than traditional matrix clocks, as it efficiently manages communication within and between groups. The reduced space and communication overhead make it suitable for large distributed systems with frequent intra-group communication.

In summary, the proposed group-based matrix clock algorithm achieves a balance between reducing space and communication overhead and maintaining accurate causality tracking. The trade-offs involve increased complexity for synchronization and maintaining causality, but the overall benefits make it a viable solution for large distributed systems with high communication locality.

5. FUTURE WORK

Future research can explore the following directions to enhance the proposed vehicle routing optimization algorithms:

5.1 Integration of Advanced Clock Synchronization Techniques

The integration of advanced clock synchronization algorithms, such as those reviewed by Dissanayake et al. [1], offers a promising avenue for improving vehicle routing optimization. Investigating whether enhanced time accuracy and reduced message complexity contribute to more efficient and robust routing solutions will further refine the proposed approach.

5.2 Fault-Tolerant Synchronization Systems

Incorporating fault-tolerant clock synchronization systems, such as those outlined in [4], into the optimization framework could improve reliability and performance, particularly in large-scale and dynamic environments. Evaluating these mechanisms' impact on the scalability and fault resilience of the routing algorithms remains a crucial area for investigation.

5.3 Virtualized Real-Time Systems

The role of clock synchronization in virtualized distributed real-time systems, as discussed by Ruh et al. [7], represents another promising direction. Analyzing the effects of virtualized environments and global time bases on synchronization precision and resource efficiency will inform the design of optimized routing algorithms in such settings.

5.4 Comparison with Other Synchronization Protocols

A comparative analysis of different clock synchronization protocols and their impact on vehicle routing problems is essential. By implementing and benchmarking various approaches, this research can identify the most effective synchronization strategies for specific scenarios, highlighting their relative strengths and limitations.

5.5 Experimental Validation and Optimization

Extending experimental validation to scenarios involving fault-tolerant and virtualized environments will provide valuable insights into the practical constraints and challenges of real-world systems. These experiments will enable the development of more robust solutions that account for failures and dynamic conditions.

6. CONCLUSION

This paper presents a novel group-based matrix clock algorithm designed to address the scalability challenges of traditional matrix clocks while preserving their robust causality tracking capabilities. By leveraging communication locality patterns and partitioning processes into groups based on their communication frequencies, the algorithm reduces communication overhead while maintaining accuracy in event order.

Key innovations include the maintenance of smaller intra-group matrix clocks for frequent within-group communication and compact group-level matrix clocks for inter-group interactions. This dual-level structure ensures manageable timestamp sizes, resulting in reduced space and communication overheads compared to traditional matrix clock algorithms.

Theoretical analysis and empirical evaluation demonstrate that the proposed algorithm achieves substantial reductions in space complexity and communication overhead, particularly in systems with heterogeneous communication patterns. The algorithm preserves causality and event ordering, making it a viable solution for various distributed applications.

While the proposed method introduces additional complexity in synchronizing group matrix clocks and maintaining causality during inter-group communication, these trade-offs are offset by significant gains in efficiency and scalability. The approach is particularly suited to real-world distributed systems, where optimizing for communication locality can lead to substantial performance improvements.

7. REFERENCES

[1] Chandeepa Dissanayake and Chanuka Algama. A review on message complexity of the algorithms for clock synchronization in distributed systems, 2024.

[2] C.J. Fidge. Timestamps in message-passing systems that preserve the partial ordering. *Australian Computer Science Communications*, 10(1):56–66, 1988.

[3] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[4] Yuliang Li, Gautam Kumar, Hema Hariharan, Hassan Wassel, Peter Hochschild, Dave Platt, Simon Sabato, Minlan Yu, Nandita Dukkipati, Prashant Chandra, and Amin Vahdat. Sundial: Fault-tolerant clock synchronization for datacenters. In *14th*

USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), pages 1171–1186. USENIX Association, November 2020.

[5] Friedemann Mattern. Virtual time and global states of distributed systems. *Parallel and Distributed Algorithms*, 1(23):215–226, 1989.

[6] Friedemann Mattern. Efficient algorithms for distributed snapshots and global virtual time approximation. *Journal of Parallel and Distributed Computing*, 18(4):423–434, 1993.

[7] Jan Ruh, Wilfried Steiner, and Gerhard Fohler. Clock synchronization in virtualized distributed real-time systems using ieee 802.1as and acrn. *IEEE Access*, 9:126075–126094, 2021.

[8] A. Singh and N. Badal. An overview of matrix clock synchronization in distributed computing environments. *International Journal of Computer Applications*, 125(3):24–30, 2015.

8. APPENDIX

8.1 Theorem 1

8.1.1 ***Statement****. "The lower bound for the space complexity using the proposed algorithm is* $\Omega(N^2)$*."*

8.1.2 ***Proof****.* Total number of processes in the system is N.
The total number of groups is N_g, each containing k_i processes. Using Equation (3), the total space required is:

$$\textit{Space} = N \times \left(Var(k) + Exp(k)^2 + (N_g)^2\right) \tag{4}$$

To minimize the required space, the expression on the right-hand side of Equation (6) must be at its global minimum. Therefore, we need to find:

$$\textit{min(Space)} = N \times \min\left(Var(k) + Exp(k)^2 + (N_g)^2\right) \tag{5}$$

Variance can be expressed as:

$$Var(k) = \frac{1}{N_g}\sum_{i=1}^{N_g}(k_i - Exp(k))^2$$

The lowest value of $Var(k)$ occurs when all k_i are equal, making:

$$\textit{min(Var}(k)) = 0$$

Thus, Equation (7) becomes:

$$\textit{min(Space)} = N \times \min\left(Exp(k)^2 + (N_g)^2\right)$$

Let:

$$f(N_g) = Exp(k)^2 + (N_g)^2 \tag{6}$$

By the law of large numbers, for sufficiently large N_g (as $N_g \to \infty$), the expectation of k can be expressed as:

$$Exp(k)^2 = \left(\frac{1}{N_g}\sum_{i=1}^{N_g} k_i\right)^2$$

Using the system assumption from Equation (1):

$$Exp(k)^2 = \left(\frac{1}{N_g} \times N\right)^2$$

$$Exp(k)^2 = \left(\frac{N}{N_g}\right)^2$$

Substituting this into Equation (8):

$$f(N_g) = \left(\frac{N}{N_g}\right)^2 + (N_g)^2 \tag{7}$$

Taking the derivative of $f(N_g)$:

$$\frac{d}{dN_g}f(N_g) = \frac{d}{dN_g}\left(\left(\frac{N}{N_g}\right)^2 + (N_g)^2\right)$$

$$\frac{d}{dN_g}f(N_g) = \frac{-2N^2}{N_g^3} + 2N_g$$

$$\frac{d}{dN_g}f(N_g) = \frac{-2N^2 + 2N_g^4}{N_g^3}$$

Since the extreme values are only possible at stationary points or boundary values. We will first find the stationary points where:

$$\frac{d}{dN_g}f(N_g) = 0$$

$$\frac{-2N^2 + 2N_g^4}{N_g^3} = 0$$

The numerator must equal zero:

$$-2N^2 + 2N_g^4 = 0$$

$$2N_g^4 = 2N^2$$

$$N_g^4 = N^2$$

$$N_g = \pm\sqrt{N}$$

Since $N_g > 0$, we have:

$$N_g = \sqrt{N}$$

Therefore the value for $f(N_g)$ at $\sqrt{N}$ from equation (8) is :

$$f(\sqrt{N}) = \left(\frac{N}{\sqrt{N}}\right)^2 + (\sqrt{N})^2$$

$$f(\sqrt{N}) = 2N \tag{8}$$

Note that N_g can only take values between 1 and N. So $1 \leq N_g \leq N$.
Now evaluating equation $f(N_g)$ at boundary points:

$$f(1) = N^2 + 1$$

Note that $Var(k)$ is zero when $N_g = 1$ as there is only 1 group, so $Exp(k) - k_i = 0, \forall i$.
Similarly :

$$f(N) = 1 + N^2$$

Again note that $Var(k)$ is zero when $N_g = N$ as there are N groups. Therefore each group must contain one and only one element, to fulfill the constraint presented in Equation (1). So,

$$k_i = 1, \forall i$$

Therefore :

$$Exp(k) = 1$$

And

$$Exp(k) - k_i = 0, \forall i$$

which results in $Var(k) = 0$
As we already observed that the only positive and real stationary point for $f(N)$ is $\sqrt{N}$, and clearly:

$$f(\sqrt{N}) < f(1)$$

$$f(\sqrt{N}) < f(N)$$

Substituting using Equation (8) in (5) we have

$$min(space) = N \times (Var(k) + 2N)$$

As $Var(k) = 0$ when $N_g = \sqrt{N}$ and $k_i = \frac{N}{\sqrt{N}} \forall i$.

$$min(space) = 2N^2$$

Hence, the lower bound for the space complexity possible is $\Omega(N^2)$.

8.2 Theorem 2

8.2.1 ***Statement****. "The upper bound for the space complexity using the proposed algorithm is $O(N^3)$."*

8.2.2 ***Proof****.* Total number of processes in the system is N.
The total number of groups is N_g, each containing k_i processes. Using Equation (3), the total space required is:

$$Space = N \times \left(Var(k) + Exp(k)^2 + (N_g)^2\right) \quad (9)$$

To maximize the required space, the expression on the right-hand side of Equation (6) must be at its global maximum. Therefore, we need to find:

$$max(Space) = N \times \max\left(Var(k) + Exp(k)^2 + (N_g)^2\right) \quad (10)$$

Let:

$$f(N_g) = Exp(k)^2 + (N_g)^2 \quad (11)$$

By the law of large numbers, for sufficiently large N_g (as $N_g \to \infty$), the expectation of k can be expressed as:

$$Exp(k)^2 = \left(\frac{1}{N_g}\sum_{i=1}^{N_g} k_i\right)^2$$

Using the system assumption from Equation (1):

$$Exp(k)^2 = \left(\frac{1}{N_g} \times N\right)^2$$

$$Exp(k)^2 = \left(\frac{N}{N_g}\right)^2$$

Substituting this into Equation (8):

$$f(N_g) = \left(\frac{N}{N_g}\right)^2 + (N_g)^2 \quad (12)$$

The derivative of $f(N_g)$:

$$\frac{d}{dN_g} f(N_g) = \frac{d}{dN_g}\left(\left(\frac{N}{N_g}\right)^2 + (N_g)^2\right)$$

$$\frac{d}{dN_g} f(N_g) = \frac{-2N^2}{N_g^3} + 2N_g$$

$$\frac{d}{dN_g} f(N_g) = \frac{-2N^2 + 2N_g^4}{N_g^3}$$

Since the extreme values are only possible at stationary points or boundary values. First find the stationary points where:

$$\frac{d}{dN_g} f(N_g) = 0$$

$$\frac{-2N^2 + 2N_g^4}{N_g^3} = 0$$

The numerator must equal zero:

$$-2N^2 + 2N_g^4 = 0$$

$$2N_g^4 = 2N^2$$

$$N_g^4 = N^2$$

$$N_g = \pm\sqrt{N}$$

Since $N_g > 0$, following is chosen:

$$N_g = \sqrt{N}$$

Therefore the value for $f(N_g)$ at $\sqrt{N}$ from equation (8) is :

$$f(\sqrt{N}) = \left(\frac{N}{\sqrt{N}}\right)^2 + (\sqrt{N})^2$$

$$f(\sqrt{N}) = 2N \quad (13)$$

Note that N_g can only take values between 1 and N. So $1 \leq N_g \leq N$.
Now evaluating equation $f(N_g)$ at boundary points:

$$f(1) = N^2 + 1$$

Note that $Var(k)$ is zero when $N_g = 1$ as there is only 1 group, so $Exp(k) - k_i = 0, \forall i$.
Similarly :

$$f(N) = 1 + N^2$$

Again note that $Var(k)$ is zero when $N_g = N$ as there are N groups. Therefore each group must contain one and only one element, to fulfill the constraint presented in Equation (1). So,

$$k_i = 1, \forall i$$

Therefore :

$$Exp(k) = 1$$

And

$$Exp(k) - k_i = 0, \forall i$$

which results in $Var(k) = 0$
As it is already observed that the only positive and real stationary point for $f(N)$ is $\sqrt{N}$, and clearly:

$$f(\sqrt{N}) < f(1)$$

$$f(\sqrt{N}) < f(N)$$

Therefore it can be concludde that the maximum values for $f(N_g)$ occur only at two boundaries $N_g = 1$ and $N_g = N$. Hence the maximum value for $f(N_g)$ is :

$$f(N) = N^2 + 1 \tag{14}$$

Substituting using Equation (14) in (10) we have

$$max(space) = N \times (Var(k) + N^2 + 1)$$

As $Var(k) = 0$ for both $N_g = 1$ and $N_g = N$.

$$max(space) = N^3 + N$$

Hence, the upper bound for the space complexity is $O(N^3)$.