IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 21, 2017, accepted December 2, 2017. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.2782763

A New Processing Approach for Reducing Computational Complexity in Cloud-RAN Mobile Networks

ALI M. MAHMOOD[1,2], ADIL AL-YASIRI[1], AND OMAR Y. K. ALANI[1]
[1]University of Salford, Manchester M5 4WT, U.K.
[2]University of Technology, Iraq, Baghdad 10066, Iraq

Corresponding author: Ali M. Mahmood (a.mahmood7@edu.salford.ac.uk)

This work was supported by the Ministry of Higher Education and the Scientific Research of Iraq–University of Technology.

ABSTRACT Cloud computing is considered as one of the key drivers for the next generation of mobile networks (e.g. 5G). This is combined with the dramatic expansion in mobile networks, involving millions (or even billions) of subscribers with a greater number of current and future mobile applications (e.g. IoT). Cloud Radio Access Network (C-RAN) architecture has been proposed as a novel concept to gain the benefits of cloud computing as an efficient computing resource, to meet the requirements of future cellular networks. However, the computational complexity of obtaining the channel state information in the full-centralized C-RAN increases as the size of the network is scaled up, as a result of enlargement in channel information matrices. To tackle this problem of complexity and latency, MapReduce framework and fast matrix algorithms are proposed. This paper presents two levels of complexity reduction in the process of estimating the channel information in cellular networks. The results illustrate that complexity can be minimized from $O(N^3)$ to $O((N/k)^3)$, where N is the total number of RRHs and k is the number of RRHs per group, by dividing the processing of RRHs into parallel groups and harnessing the MapReduce parallel algorithm in order to process them. The second approach reduces the computation complexity from $O((N/k)^3)$ to $O((N/k)^{2.807})$ using the algorithms of fast matrix inversion. The reduction in complexity and latency leads to a significant improvement in both the estimation time and in the scalability of C-RAN networks.

INDEX TERMS C-RAN, Channel State Information, Computation Complexity, MapReduce, Fast Matrix Algorithms, Strassen's Algorithm, Block LU Decomposition.

I. INTRODUCTION

Mobile networks have witnessed an unprecedented growth in terms of the number of users and the amount of data traffic. The 5G network is supposed to support 1 million user equipment (UE) per square kilometer with 1ms end-to-end latency [1]. Hence, the data rate of future 5G has been expected to be 10 times faster than the speed of 4G networks [2]. The expansion requires novel technologies to be developed to meet future increased demand for mobile users. Recently, C-RAN technology has been gaining enlarged recognition from researchers and mobile network operators and has been nominated as the architecture of 5G [3]. Unlike the current mobile networks, which have the baseband unit co-located within the cell site, baseband processing in C-RAN has been moved to cloud computing for central processing and management. C-RAN has three components (Fig. 1) viz (a) a remote radio head (RRH) [4] which acts as a remote antenna and is situated remotely, (b) low latency and high capacity optical communication networks known as fronthaul communicating links, which connect RHHs to the baseband unit (BBU) pool, and (c) a VBS (virtual base station) pool or BBU, which is situated in the cloud for centralized signal processing. Whilst C-RAN has many positive attributes, it also has some challenges. One of these challenges refers to an increase in computational complexity involved in acquiring the large size of channel information matrix H, with expansion of the network, due to centralized coordination and processing [5]. This matrix includes the channel state information (CSI) between the user equipment (UE) and the VBS. This means the delay in estimating this information will delay the process of linking between the UEs and the VBS, in terms of adaptation.

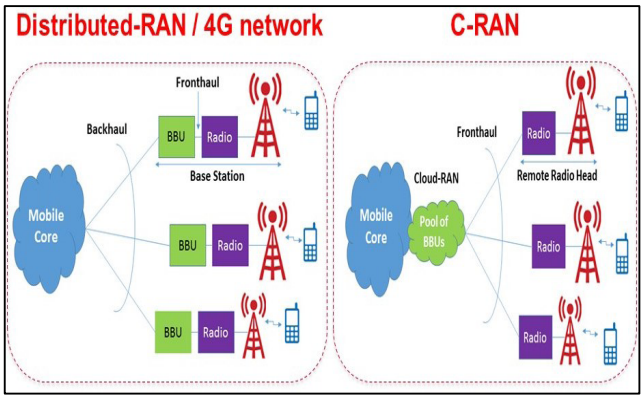

FIGURE 1. C-RAN architecture versus distributed 4G network architecture.

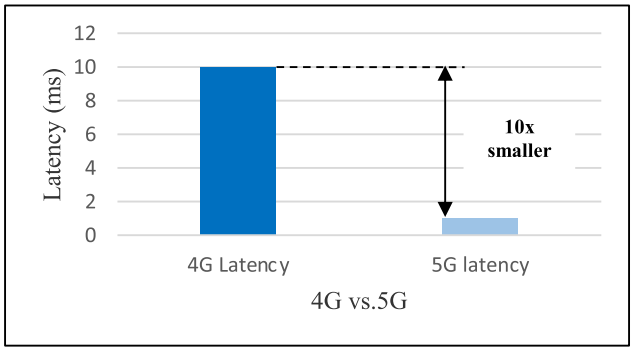

FIGURE 2. Theoretical latency requirement in 4G versus the expected 1ms in 5G.

Consequently, the acquisition of CSI will affect the entire performance of cellular networks, particularly the system throughput, which ultimately limits C-RAN scalability. In this paper, two novel approaches are proposed for the C-RAN architecture to decrease the overhead of acquiring the CSI: MapReduce framework [6] and fast matrix inversion with multiplication algorithms. Deploying these two approaches in C-RAN will support network scalability and maintain the next generation of ultra-low latency requirements.

The motivation and the contribution of this paper can be summarised as follows.

The most important challenges in C-RAN is the challenge of dimensionality [7]. This is due to centralized coordination of all network elements in cloud computing. In other words, the magnitude of channel matrix H in full centralized C-RAN increases dramatically at the increase in the number of RRHs and the UEs in the network. The burden of estimating this information leads to high processing time, which increases the network latency and reduces the throughput accordingly. However, as shown in Fig.2, in the next generation 5G networks, the target latency is 1ms, which is considered a challenge and the key driver to implement the future 5G technologies (e.g. autonomous cars and tactile internet). To overcome this challenge, two novel approaches are proposed.

- The first is to deploy MapReduce as a processing framework in C-RAN networks to maintain low computational complexity in the centralized pool of VBS, to meet the future low latency and coherence time requirements, and then to support network scalability (for large numbers of RRHs). To the best of our knowledge, there is no prior work that has used MapReduce in the channel estimation of communication systems.
- The second is to propose fast matrix inversion algorithms, to reduce the processing time of channel estimation in C-RAN architecture. This algorithm takes the advantage of both Strassen's and Block LU decomposition to reduce the execution time of the matrix inversion of the MMSE estimator.The list of notations used in the paper is specified in Table 1, which aids in understanding the concepts discussed in the paper. The rest of the paper is structured as follows: section II discusses some related work on the research problem outlined above. Section III demonstrates the background on the main components of the research. Section IV defines the research problem. Section V presents MapReduce as a proposed solution along with the complexity analysis and simulation results. Section VI includes a mathematical modeling for the MapReduce framework using queuing theory. Section VII deals with the proposed fast matrix operations algorithms (Strassen and Block LU decomposition). Section VIII illustrates simulation results and discussion. Section IX finally presents the conclusions and the possible future avenues of exploration.

TABLE 1. List of notations.

Notation	Explanation
$[\bullet]^T$	Transpose
$[\bullet]^H$	Conjugate transpose
$E\{\bullet\}$	The expectation
$\widehat{\mathrm{H}}$	The estimated value of H
I	Identity matrix
$\mathrm{O}(\bullet)$	Big O notation
$T(n)$	Total number of mathematical operations
P	The permutation matrix

II. RELATED WORK

The problem of increased computational complexity in obtaining the CSI has a negative effect on the scalability of the C-RAN networks. A large number of research studies have focused on sparsification technique studies [7]–[9], to make the matrix of channel information sparser by excluding some entries, and then to reduce computational complexity of acquiring the channel information. However, these approaches may limit the network performance or decrease the network capacity, since the number of users is reduced according to - e.g. their distances from RRHs without considering the inter-site distance for different types of base stations (macro, micro and pico).

Another research strategy focuses on the antenna selection approaches [10]–[12], either selecting subsets or coordinating the number of active antennas. These approaches may reduce the overhead of CSI acquisition. However, the consequences might minimize the overall network capacity, because of the reduction in the number of antennas. These studies might consider the best case having low UE density, which requires less antennas. However, these approaches may fall short behind the provision of any improvement in worst-case scenarios, such as when having high density UEs, in busy urban areas when all antennas are required.

The authors in [13] and [14] have tried to minimize the complexity in the channel estimation algorithm itself. This involves either suggesting new estimators or modifying the current estimators. However, it has been observed that there is a trade-off between the performance (accuracy) and the complexity of the estimator. Many studies have tried to minimize the overhead of the most common estimator, which is the minimum mean square estimator (MMSE) by approximating the cubic complexity of matrix inversion by L-degree matrix polynomial, such as those presented in [14]–[18].

One of the trends in the research studies is to use time division duplex (TDD) systems, which utilize the channel reciprocity to reduce the overhead of the CSI acquisition. However, a TDD system has the following problems: first i) the ''pilot contamination'' is the biggest problem in TDD systems, which happens when the channel estimation at the base station in one cell becomes contaminated by users from other cells [19], [20]; ii) The same uplink/downlink timeslot arrangement must be used at all cell sites in adjacent service areas; iii) If the TDD spectrum is divided amongst multiple operators in the same area then all operators must be strictly time synchronized and have the same uplink/downlink timeslot arrangements; iv) According to the study in [21], the authors state that in the TDD systems there is still an essential need for a downlink reference signal (RS) and the uplink CSI feedback, since the measurement at the transmitter may not capture the downlink interference of the neighboring cells. Therefore, the downlink RS is still essential to find the CQI for the TDD mode [21]; v) Currently the LTE licenses worldwide are less than or equal to 40 for TDD systems, while for the frequency division duplexing (FDD) systems, LTE has almost 300 more than TDD [19]; vi) a TDD system requires a large guard period for the base station to switch from downlink transmission to uplink transmission and vice versa, which leads to decline both the efficiency and the cell throughput in comparison with the FDD system, that has two separated frequencies for the uplink and downlink [22]. Hence, it is worth investigating the problem of CSI acquisition complexity in the C-RAN for the FDD systems.

In [5], [23], and [24], to decrease the overhead of CSI acquisition the authors have suggested the clustering methods in large networks, because in populated networks, the aim of obtaining CSI can be achieved by controlling the cluster size rather than the whole network. Clustering approaches can be considered as promising techniques to reduce computational complexity. However, choosing the size and the radius of the cluster is considered as one of the main challenges. Several studies demonstrate that it is possible to implement a clustering technique for a large group of RRHs in C-RAN architecture. In the literature, clustering has been deployed for different purposes, such as for power minimization [25]–[28], and for interference mitigation by using cooperative multi point (CoMP) among neighboring clusters [29]–[31]. Another purpose of using clustering in C-RAN is for cost reduction [32], [33] by shortening the overall fiber cable length required in the fronthaul connection via deploying ring topology. Clustering is also used for complexity reduction, as in [5], [34]–[36]. This is owing to the cooperative property of C-RAN architecture, which enables full sharing of channel information by exchanging the CSI among the VBSs in the cloud. However, the focus of these studies was more on the implementation of the clustering technique, overlooking the method of processing a large number of clusters (which is formulated from a great number of RRHs). Particularly with the deployment of next generation 5G networks, there is a need for a large number of access points. For instance, the distance between two access points is expected to be less than 150 meters. Hence, an efficient and powerful processing framework is required as a processing paradigm for providing scalable distribution of hundreds or thousands of RRH groups to cope with the requirements of the next generation of mobile networks.

In this research, two novel approaches are proposed to reduce processing complexity and latency. The first approach is in managing CSI acquisition in a well distributed manner using MapReduce framework. The algorithm lies in the clustering category, in which a group of RRHs is chosen to be assigned to a single VBS. All other VBSs cooperate and work in a parallel manner to minimize the latency whilst maintaining the network performance. It is worth stating that MapReduce has been employed as a scalable processing paradigm to accelerate the processing of big datasets in the cloud for a large number of applications, such as scalable streaming systems, real-time prediction for explosive traffic flow data, and also, MapReduce has been used in indexing web content with the database system in the Google search engine.

Secondly, the concept of fast matrix inversion using Strassen's algorithm and Block LU decomposition, is proposed to minimize the complexity in the computation of MMSE estimator.

III. RESEARCH ENABLERS

This section discusses the important components of this research: MapReduce, CSI, and common channel estimation algorithms.

A. BACKGROUND TO MapReduce

This is a programming framework, which enables the implementation of jobs in a distributed and parallel manner, as shown in Fig.3.

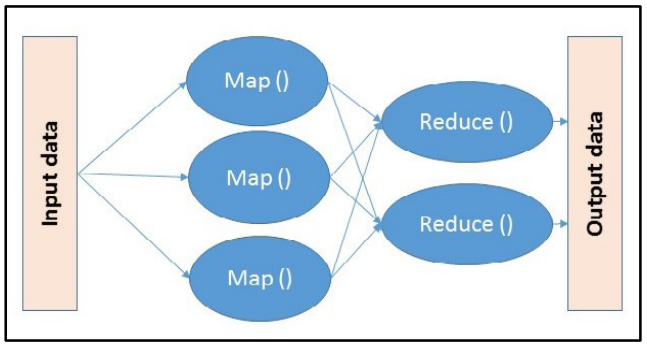

FIGURE 3. The general diagram of MapReduce framework.

MapReduce was introduced by Google [37] in order to process big data sets. Tasks are submitted initially to a division phase. Throughout this phase, the jobs include the number of tasks, mapped to a group of available mappers, for the purpose of processing. The mapper receives and produces an input and intermediate key/value pairs, respectively. The reducer takes an intermediate key and values set for that key, combining them together to form a smaller set of values. The main features of MapReduce are easy programming, automatic parallelism, and fault tolerance. In general, MapReduce exploits the "***Divide and Conquer***" principle. Hence, in this research instead of acquiring the channel information of all network elements (RRHs with their UEs) in one big channel matrix, the estimation algorithms are applied for a prespecified group of RRHs for each VBS using the MapReduce framework.

B. CHANNEL STATE INFORMATION IN CELLULAR NETWORKS

The mobile UE has to send its channel situation, which should be represented by the channel state information, to the base station for link adaptation between the VBS and the UE. The CSI mainly includes three reports; the precoding matrix indicator (PMI), rank indicator (RI), and channel quality indicator (CQI) [38]. Since adapting the modulation and coding scheme (MCS) depends on the CSI for instance, when the value channel quality indicator (CQI) (which is one of the CSI reports that are used to indicate the quality of communication link) is high, the value of MCS is also high. The RI is the number of useful transmitter antennas that can be used in the spatial multiplexing mode. The PMI is a measure that provides a preferred precoding matrix to be used in a VBS for a given radio link in spatial multiplexing mode. Hence, the former functions of the CSI reports reveal that the inaccuracy in this information leads to performance degradation in the entire mobile network.

C. SYSTEM MODEL AND CHANNEL ESTIMATION ALGORITHMS

The channel model of the received signal is presented in Equation (1). In full centralized C-RAN, the received signal in the equation for uplink transmission [5], [11], where N represents the number of RRHs with single-antenna while K represents the number of antenna in UEs as follows:

$$Y_N = \sum_{i=1}^{K} H_i X_i + Z_i \tag{1}$$

where, Y is the vector of the received signal; H is the channel state information matrix; X is the transmitted signal vector from K users; and Z_i is the received noise vector. In wireless communication systems, there are two common estimation algorithms, which are the MMSE and the least square (LS) estimator. The function of the estimator is to estimate the channel information (Matrix H), which includes the channel state information (CSI). The following two subsections present a brief description for both, the LS and MMSE estimators.

1) LEAST SQUARE ESTIMATOR

The objective of the channel LS estimator is to minimize the square value between the received signal Y and the pilot signal X. The least square estimate of the channel can be obtained by dividing the received signal by its expected value, as shown in Equation (2). The LS estimator has low computational complexity, since it is designed to work without any knowledge of the statistics of the channels. However, this estimator suffers from performance degradation due to the high mean square error (MSE) [14] in comparison with the MMSE, as shown in Fig.4.

$$\hat{H}_{LS} = \left[\frac{Y}{X}\right]^T = X^{-1}Y \tag{2}$$

Where, $\hat{H}_{LS}$ denotes the estimate of channel H

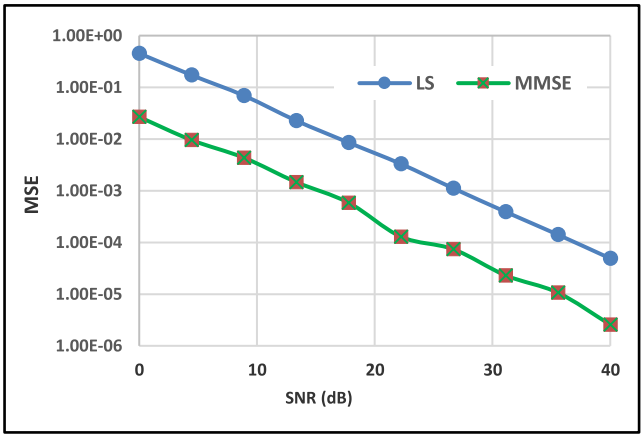

FIGURE 4. Comparison between MMSE and LS estimators in terms of MSE.

2) MMSE ESTIMATOR

The MMSE estimator performs second-order statistics to minimize the mean square error (MSE). The MMSE estimate of the channel responses as given in Equation (1) can be obtained as follows [14], [39].

The statistics of MMSE estimator is represented by three auto-covariance matrices, Rgg, R_{YY} and R_{HH} and the cross-covariance matrix R_{gY}. These matrices can be calculated as follows.

$$R_{YY} = E\left\{YY^H\right\} = xFR_{gg}F^H x^H + \delta_n^2 I_n \quad (3)$$

$$R_{HH} = E\left\{HH^H\right\} = FR_{gg}F^H \quad (4)$$

$$R_{gY} = E\left\{gY^H\right\} = R_{gg}F^H x^H \quad (5)$$

Then, the estimation of the channel matrix in the MMSE estimator can be determined as follows:

$$g_{mmse} = R_{gY}R_{YY}^{-1}Y \quad (6)$$

$$\hat{H}_{mmse} = Fg_{mmse} = F\left[\left(F^H X^H\right)^{-1} R_{gg}^{-1}\delta_n^2 + XF\right]^{-1} Y$$

$$= FR_{gg}\left[\left(F^H X^H XF\right)^{-1}\delta_n^2 + R_{gg}\right]^{-1} F^{-1}\hat{H}_{LS}$$

$$\hat{H}_{mmse} = R_{HH}\left[R_{HH} + \delta_n^2(XX^H)^{-1}\right]^{-1} X^{-1}Y \quad (7)$$

where,

- g: is the channel energy.
- Y: is the received signal.
- F: is the discrete-time Fourier transform (DFT) matrix.
- R_{gY}: is the cross-covariance matrix of g and y.
- R_{gg}: is the auto-covariance matrix of g.
- R_{YY}: is the auto-covariance matrix of Y.
- R_{HH}: is the auto-covariance matrix of H.

It is worth noticing that the MMSE estimator has the best performance in comparison with the LS and with other estimators, in terms of MSE [14], [39]. The simulation test is conducted as shown in Fig. 4 to quantify the performance of both estimators with the following key settings, 1UE and 1VBS, 4×4 antennas at UE and VBS and 1.4 BW. The result in Fig.4 shows superior performance for MMSE in comparison with LS. However, the main drawback of the MMSE is the high computational complexity as matrix inversion is required every time data changes [14]–[18]. Therefore, in C-RAN unlike the current mobile networks e.g. LTE-A, this computational complexity of acquiring channel information is extremely expensive and will be increased several times due to full centralized coordination on cloud computing for hundreds of RRHs which leads to huge channel information matrices. Hence, the focus of this study is to minimize the complexity of acquiring the CSI using MMSE estimator in future C-RAN architecture.

IV. PROBLEM DEFINITION

In C-RAN, the extremely large channel matrices can be considered one of the main causes of the imperfection in the CSI. The detailed explanation for this problem is as follows.

After the introduction of MIMO technology, the importance of accurate CSI acquisition increased, since this affects how efficiently the MIMO system works [40]. Practically, however, when there is an increase in the number of antennas, the size of the matrix H increases and this leads to increased overheads of acquiring the CSI [34]. In the mathematical equation, the size of matrix H can be expressed by the following equation [38], [41], [42]:

$$H_{size} = Sc \times N_s \times A_r \times A_t \quad (8)$$

Where,

- Sc: number of subcarriers;
- N_s: number of OFDM symbols;
- A_r: number of receive antennas;
- A_t: number of transmit antennas

The number of OFDM subcarriers in the 3gpp standard is represented in Table 2 and the number of OFDM symbols is either (14 or 12) per subframe based on whether the normal or extended cyclic prefix is used [38].

TABLE 2. OFDM subcarriers for uplink in each bandwidth of LTE systems [38].

Bandwidth	Resource Blocks	Subcarriers (uplink)
1.4 MHz	6	72
3 MHz	15	180
5 MHz	25	300
10 MHz	50	600
15 MHz	75	900
20 MHz	100	1200

To quantify the amount of increase in the estimation time, a model test was performed with one base station (BS) and five user equipment (UE) using Minimum Mean Square Error (MMSE) estimation algorithm. Table 3 and Fig. 5 show how the dimension of channel matrix H increases, along with the estimated time increasing in proportion. Contrary to the LTE-A network that operates in one-to-one mapping between the base station and RRH, in the centralized RAN architecture, the dimensions of the channel matrix rise equivalently with the number of RRHs and UEs; because the signal processing is aggregated in the cloud [6]. In addition, C-RAN operates (albeit ineffectively) in one to many mapping. This means that each one of the VBS in the cloud manages hundreds of RRHs to enlarge the network capacity [8]. As a result of this, the computational complexity increases at the centralized BBU pool, thus decreasing effectiveness. Table 3 and Fig. 6 also illustrate that there is a significant increase in the percentage of latency with an increase in the number of antennas at the VBS. For instance, with 128×4 antennas system, the latency is increased almost 700 times compared to the latency of 1x1 antenna system. It is worth stating that the first value of estimation time for 1x1 system is considered as a base for calculating the percentage of increase in the latency for larger numbers of antennas. As per the above Equation (1), the reason for the overhead in C-RAN is due to the increase in the size of the matrix H. This also corresponds to the growth in the number of RRHs and UEs. Hence, for the extremely large channel matrix H, the estimation and processing delays the CSI acquisition in C-RAN architecture.

TABLE 3. The dimension of the channel matrix H against the estimation time overhead increase.

No. of BS	No. of UE	No. of Antennas at BS (N)	No. of Antennas of UE (K)	Size of H BW =1.4 MHz	Estimation time (sec) for (MMSE estimator)	Percentage of latency increase % with respect to 1x1 antenna system
1	5	1	1	1008	0.00037962	0
1	5	2	1	2016	0.0007574	0.995
1	5	2	2	4032	0.0010	1.634
1	5	4	2	8064	0.0026	5.849
1	5	4	4	16128	0.0053	12.961
1	5	8	4	32256	0.0156	40.094
1	5	16	4	64512	0.0308	80.134
1	5	32	4	129024	0.0619	162.058
1	5	64	4	258048	0.1269	333.282
1	5	128	4	516096	0.2685	706.286

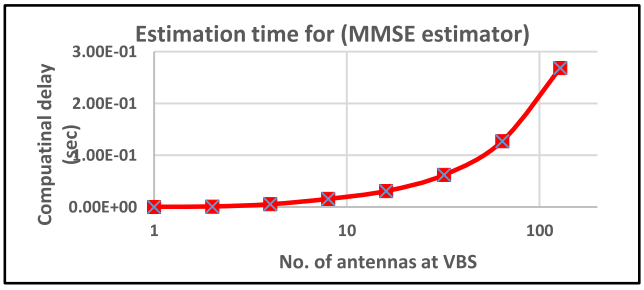

FIGURE 5. The rise of estimation time in relation to the no. of antennas.

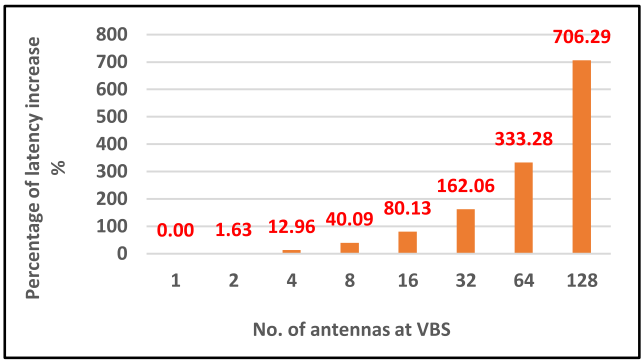

FIGURE 6. Percentage of latency increase versus number of antennas at VBS.

That is, with the expansion of the network size, the burden of computational complexity per user expands as well [5]. As a result, the delay of CSI leads to inaccurate decisions at the VBS. This is because the recently obtained CSI is not updated and consequently VBS does not represent the current (true) state at the mobile user. Figure 7 clarifies the problem of overheads by presenting how the increase in the dimension of H is proportional to the number of RRHs and the antennas of UEs with different channel bandwidths.

The Figures and Table above show that the magnitude of the channel matrix increases significantly with the increase in RRHs, which eventually causes the problem of

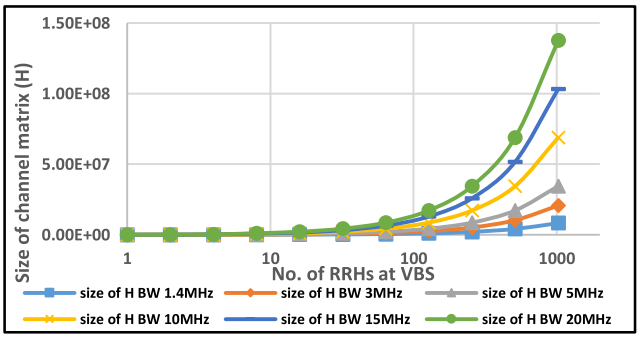

FIGURE 7. The increase in the dimension of H with the growth in RRH for different bandwidths.

increased computation overheads and increased time taken to acquire CSI. As a result, this limits the scalability of the network.

It is worth stating that the system bandwidth is one of the important factors that also causes the growth in the dimension of matrix H as shown in equation 1, although this aspect is not within the scope of this paper. To compound this problem, the bandwidth in future networks is anticipated to increase significantly by using the white space and millimeter wave bands. Therefore, when using central processing C-RAN, a considerable amount of RRHs with their own large bandwidth will create a very large matrix H and the entire RRHs will yield multiple bandwidth increases. However, using the proposed MapReduce, the bandwidth will not increase in the same manner. The aggregated bandwidth is a multiple increase of the RRHs group only, not the complete number of antenna. This will definitely off-load the computational complexity in C-RAN architecture.

V. MapReduce DESIGN AS A SOLUTION

In this work, the idea of distributed and parallel processing will be implemented in the C-RAN architecture. To attain this objective, MapReduce is used. As highlighted earlier in section IV, MapReduce is a powerful framework for performing varied jobs in a distributed manner [6]. The advantage of adopting this framework is to split the task of obtaining CSI into several parts. This is to gain the advantage of parallel processing to minimize the delay of CSI acquisition. Another benefit is its support of scalability, which is a key feature of next generation cellular networks, such as 5G. Furthermore, the objective of using MapReduce is to utilize the processing capabilities of cloud computing by formulating C-RAN in "group-to-one" mapping between the VBSs and the RRHs, as illustrated in Fig. 8. In this division, the relation between the scalability of network and the channel estimation overhead is broken by using MapReduce. This is because the size of matrix H is limited to fewer numbers of RRHs and users. The operations of MapReduce require two phases; at the beginning, it splits the input data and then the parallel processing is performed on the partitioned data. The Mappers or workers will contain the algorithm of channel

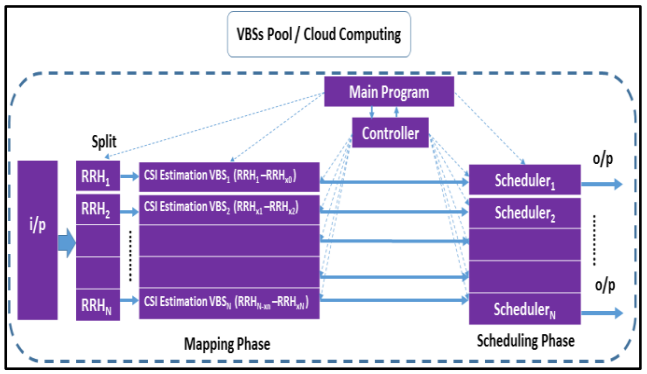

FIGURE 8. The proposed MapRedcue framework.

Algorithm 1 RRHs_Splitting

Input:
- Total number of RRHs (N)
- Size of the group of RRHs (n)

Output:
- Group of RRHs per VBS (rrh)
- No. of required VBSs (V) and no. of mappers (M)

```
function RRHs splitting (N, n, RRHs)
Set V = M = [N/n]
for i = 1 to N do
    for j = 1 to V do
rrh(i, j) ←RRH(i, j)
    end for
end for
return (V, M, rrh)
end function
```

estimation (e.g. MMSE or LS) to acquire the CSI of the pre-specified group of RRHs. MapReduce can be deployed without the reducing phase [43]. Hence, this research amends the MapReduce framework in order to enhance its capabilities to gain the advantage of parallel processing.

The most important amendment is the use of the framework with the omission of the reducing functions, as reduction is not a prerequisite in this design. Therefore, in this work the name 'reducing' phase can be changed into 'scheduling' phase, since its main function is to schedule the estimated CSI directly to the scheduler of the VBS. The detailed description for the proposed MapReduce design is explained in Algorithms 1 and 2.

VI. ANALYTICAL DESCRIPTION TO THE OPERATION OF MapReduce USING QUEUEING THEORY

This section presents an analytical model to analyze the performance of C-RAN architecture with MapReduce using queuing theory. Two main benefits can be achieved from grouping RRHs in C-RAN using MapReduce. Firstly, it can reduce the time of acquiring the CSI, since the size of channel matrix is minimized based on the size of the group. Secondly, the network with this formulation can be scaled up without

Algorithm 2 CSI Acquisition Using MapReduce in C-RAN

Input:
- Y : the received signal for groups of RRHs
- X : the transmitted signal vector
- Z : noise vector

Output:
- The estimated H (CSI) for VBSs

```
Initialization
RRHs; N: no. of RRHs; K; no. of UEs per RRHs;
n: group size of RRHs (n < N)
(V, M, rrh) = RRHs_splitting (N, n)
- Select Master node / Controller;
 - Perform copies of user program (e.g. MMSE
 estimator) for (M) Mappers;
for id = 1 to V // V: no. of VBSs
(data) = Read (Y, X, Z) // call for read function
   for i = 1 to n
       for j = 1 to K
Y(i, j) = H(i, j) * X(i, j) + Z(i, j)
Data (id) = Y(i, j)
       end for
   end for
//Call MapReduce function
result = mapreduce (read, @CSIAcquisitionMapper,
@ Scheduling_CSI);
readall (result)
end for
// Call function of Mapper (1) to Mapper (M) [from
line 17, parallel processing at a time]
// function of Mappers 1 //apply channel estimation for
 VBS(1)
function CSIAcquisitionMapper1 (data, VBS_id)
{ (Y, X, Z) = data
{ find H_ estimate // using MMSE or LS
Add (intermKVStore,' VBS_id ', H_ estimate)
}
{Repeat the previous function on M mappers
// Call scheduling_CSI function
Function Scheduling_CSI (intermKey, outKVStore)
// intermKey: intermediate KeyValueStore object
// outKVStore: final KeyValueStore object
    i =1 do
   Add (outKVStore, 'VBS_id', H_ estimate (i))
While (i ≠ M)
End function
```

increasing or sharing the burden of computational complexity of the CSI acquisition among all elements of the network. Therefore, MapReduce relies on increasing the number of mappers (workers or servers or processors) in a parallel manner to complete the processing of input data. Hence, queuing theory is the most appropriate tool to describe the internal operation of the MapReduce framework. Generally, the objective of queuing analysis is to predict the performance

of the system for the purpose of minimizing the total costs of waiting time and then providing superior services. Likewise, using MapReduce reduces the time of CSI acquisition and this increases the system utilization via increasing the data throughput. Hence, this section will try to answer two questions: i) why does the data throughput increase after using MapReduce? ii) why does the estimation time decrease after deploying the MapReduce framework?

From a queuing theory point of view, MapReduce can be represented as a multiple server queuing system (M/M/S).

For the purpose of accurate description for the proposed MapReduce framework in C-RAN, the following two assumptions have been considered in the calculations of the throughput and the waiting time. Firstly, both the arrival rate λ and the service rate μ are the same for all servers. This is for the purpose of fair comparison and to be compatible with the real simulation environment. Secondly, all servers are identical and have equal capabilities.

A. THROUGHPUT OF THE SYSTEM

In general, based on the principles of queueing theory, the throughput (TP) of the M/M/S can be calculated by summing the throughput or the service rate μ of all servers as shown in the following formula [44]:

$$TP_{total\_MMS} = \sum_{i=1}^{m} TP_{per-node(i)} = \mu_1 + \mu_2 + \ldots + \mu_m \tag{9}$$

The throughput of the pool of VBSs can be calculated by aggregating the throughput of all connected UEs in each VBS and then the total TP of all VBSs represents the total throughput of the pool of VBSs. The calculation of TP can be expressed in the following three equations:

$$TP_{UE(i)} = B * log\left(1 + SINR_{(i)}\right) \tag{10}$$

The aggregated throughput of all UEs represents the throughput per cell or VBS as follows:

$$TP_{VBS(j)} = \sum_{i=1}^{z} TP_{UE}(i) \tag{11}$$

From equation 11, the total throughput for the pool of VBSs can be calculated as follows:

$$TP_{pool\ of\ VBSs} = \sum_{j=1}^{m} TP_{VBS(j)} \tag{12}$$

Where,

TP_{UE}:	Throughput of the user equipment
TP_{VBS}:	Throughput of the virtual base station
$TP_{pool\ of\ VBSs}$:	Throughput of the pool of virtual base stations
B:	Bandwidth
SINR:	Signal-to-noise-plus-interference ratio
z:	Number of UEs

Before adopting MapReduce, there was a problem in the scalability of the VBSs because of the extremely large channel matrices, which make the acquisition of the CSI a formidable task in C-RAN. Then, after considering the grouping technique of the RRHs and parallel processing of MapReduce framework, the VBSs can be scaled-up. It is worth noting that calculating the throughput in C-RAN after adding MapReduce can follow similar characteristics of the multiple servers queuing system in increasing the system throughput when scaling-up the number of servers.

B. WAITING TIME

In queuing theory, the total waiting time (T_{total}) or commonly known as the response time [45] includes two parts as expressed in equation 13. The average waiting time in the queue (T_q) and the job service time (T_s) which is the time required for the job to be served in the server.

$$T_{total} = T_q + T_s \tag{13}$$

In the present design, MapReduce can be presented as a queuing system with multi-phase service. Therefore, the service time is divided into three phases; the classifier (T_c) or the splitter of the incoming data, the mapping phase (T_M) and the reducing phase (T_R). Hence, the total service time for MapReduce framework can be expressed as follows:

$$T_s = T_c + T_M + T_R \tag{14}$$

There is no need for the function of the reducing phase in the current design. The original reducing phase takes a set of an intermediate key-value pairs produced by the mapper as the input and runs a reducer function such as shuffling, sorting, filtering, aggregating, and combining. However, for link adaptation purposes, the VBS scheduler must directly receive the acquired CSI information at each mapper without the extra reducer stage. Therefore, the reducing phase is excluded from the current design by applying MapReduce with zero number of reducers. This can be considered as an advantageous point since the reducing phase takes additional processing time and can represent a bottleneck stage in MapReduce without careful design planning [46]. Hence, the total waiting time can be represented by:

$$T_{total} = T_q + T_c + T_M \tag{15}$$

It is worth stating that the total waiting time can be affected by the number of servers, the service rate per server and the length of the queue. The rest of this section involves the mathematical representation for the component of the response time.

1) SERVICE TIME IN CLASSIFIER

The classifier is used to split the input data and send it to the servers, with a service rate $\beta \geq m\mu$ to avoid the case of bottleneck at the classifier, where, m is number of servers. Therefore the service time of classifier can be expressed as $T_c = \frac{1}{\beta}$.

2) SERVICE TIME IN MAPPING PHASE

The service time (T_M) for each mapper is assumed to be independent and exponentially distributed [47]. Therefore,

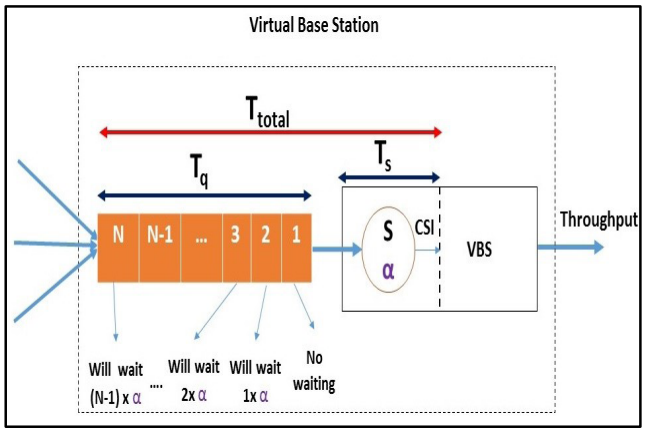

FIGURE 9. M/M/1 queuing system.

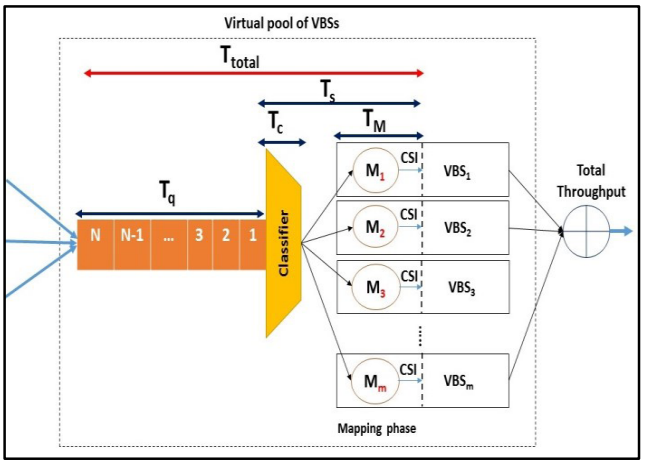

FIGURE 10. M/M/S queue with (N = m).

two possible assumptions for determining the service time at the mapping phase, which either consider all servers as identical and have the same capabilities (therefore the service time will be α where $\alpha = \frac{1}{\mu}$ or if the servers have different capabilities then the service time of mapping phase can be expressed as $\alpha = \max\left[\frac{1}{\mu_1}, \frac{1}{\mu_2}, \ldots, \frac{1}{\mu_m}\right]$ by taking the time of the slowest server.

3) WAITING TIME IN QUEUE

For the purpose of analyzing the average waiting time in the queue (T_q), a Poisson arrival λ is considered. Three different cases have been studied: (a) a queuing system with one server, (b) a queuing system with a number of servers equal to the number of chunks of input data, and (c) the queuing system with a number of servers less than the number of chunks of incoming data.

a: QUEUING SYSTEM WITH A SINGLE SERVER

In C-RAN the VBS that manage hundreds of RRHs can be considered a M/M/1 queue system. In the M/M/1 system the incoming data will suffer more waiting times in comparison to multi-server systems. In the case of M/M/1, the first chunk of data (N) will enter the server without waiting while the others will suffer from waiting in the queue as shown in Fig.9. Hence, the total waiting time (TWq) in the queue can be described as follows:

$$\mathrm{TWq} = (\mathrm{N}-1)*\alpha + (\mathrm{N}-2)*\alpha + \ldots\ldots + 1*\alpha \tag{16}$$

From equation 16, the sum of positive numbers can be represented by the following:

$$TW_q = \alpha * \sum_{i=1}^{N-1} i$$

$$TW_q = \alpha * \frac{(N-1)(N-1+1)}{2} = \alpha * \frac{N(N-1)}{2} \tag{17}$$

The average waiting time of Tq can be calculated by dividing the total waiting time TW_q by the total number of chunks of input data, which is the received service.

$$T_q = \frac{\alpha * \frac{N(N-1)}{2}}{N} = \alpha * \frac{(N-1)}{2} \tag{18}$$

Where, α is the service time $= \frac{1}{\mu}$, hence the response time can be determined as follows:

$$T_{total} = T_q + T_M = \alpha * \frac{(N-1)}{2} + \alpha \tag{19}$$

b: QUEUING SYSTEM WITH NUMBER OF CHUNKS OF ARRIVAL DATA IS EQUAL TO NUMBER OF SERVERS

In this case, the number of chunks of data (N) which represent the data from the groups of RRHs in the queuing system is equal to the number of servers (m) in the multiple server system, as shown in Fig.10. Due to the assumption above (N = m), then all the incoming N chunks of data will enter the m servers at the same time. Hence, there is no waiting time in the queue (Tq = 0). Therefore, the service time of the mapping phase requires the system to finish the processing of all data in the system, which will equal the total service time T_s. This is due to the fact that all the data will be processed and finished concurrently. Therefore, the total time can be expressed as follows:

$$T_{total} = T_s = T_c + T_M \tag{20}$$

c: QUEUING SYSTEM WITH NUMBER OF SERVERS LESS THAN THE NUMBER OF CHUNKS OF ARRIVAL DATA (N)

The system, as shown in Fig.11, is more realistic, since in the normal condition the incoming data is more than the available servers. Therefore, the average waiting time in the queue can be calculated as follows:

According to the Figure above, the number of chunks (the number of input data N) can be expressed as follows:

$$N = aK + b \tag{21}$$

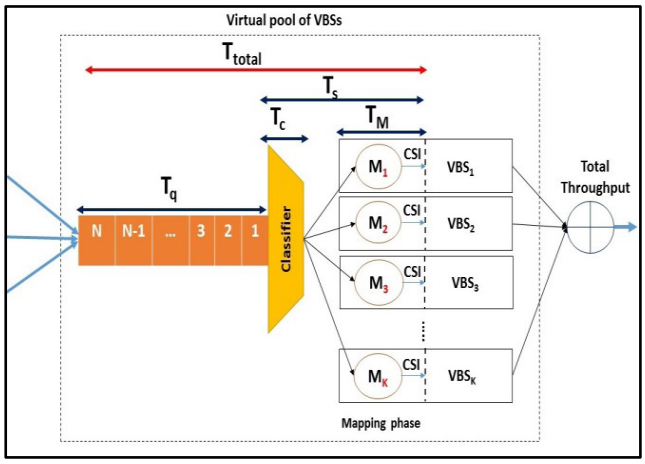

FIGURE 11. M/M/S queue with (N ≥ K).

where,

- N: number of input data in form of chunks/jobs (group of tasks) of the arrival data.
- K: number of servers.
- b: the reminder of N/K.
- a: number of times the input data has matched with the number of servers.

From the Figure above, it is noticeable that the first K chunks of data will enter the service directly without waiting in the queue: $(0)\delta$, while the second K chunks will wait $(1)\delta$ the third K chunks will wait $(2)\delta$ and this continues for (a-1) times, where δ is the summation of service time of classifier and mapping phase ($\delta = T_c + T_M$) Hence, the total waiting time in the queue can be expressed as follows:

$$\sum_{i=1}^{a-1} K\delta i + ab\delta$$

But: a = $\frac{N-b}{K}$ therefore, after replace (a) by $\frac{N-b}{K}$

$$TW_q = \left(\frac{N-b}{K}\right)\delta\left(\frac{K\left(\left(\frac{N-b}{K}\right)-1\right)+2b}{2}\right)$$

$$TW_q = \left(\frac{N-b}{K}\right)\delta\left(\frac{N-b-K+2b}{2}\right)$$

$$TW_q = \left(\frac{N-b}{K}\right)\left(\frac{N-K+b}{2}\right)\delta \quad (22)$$

For the average waiting time divide equation 24 by N

$$T_q = \frac{\left(\frac{N-b}{K}\right)\left(\frac{N-K+b}{2}\right)\delta}{N} \quad (23)$$

Therefore the overall response time can be expressed as follows:

$$T_{total} = T_q + T_c + T_M = \frac{\left(\frac{N-b}{K}\right)\left(\frac{N-K+b}{2}\right)\delta}{N} + \delta \quad (24)$$

The former part of this work investigated the potential of reducing the overall network computational complexity. This has been achieved by processing data of a group of RRHs instead of the entire number of RRHs in the network. In order to complement this, the next part of this paper analyzes the computational complexity - per VBS – using a modified MMSE estimator, minimizing the execution time of the matrix inversion using fast matrix inversion by Strassen's algorithm and Block LU decomposition.

VII. REDUCING PROCESSING TIME OF MATRIX INVERSION USING FAST MATRIX INVERSION ALGORITHMS

In the MMSE estimator, the matrix inversion was considered the main reason of its complexity [14], [18]. This section includes a novel idea of combining two algorithms, which are Block LU decomposition and Strassen's algorithm. Both of these algorithms share the principle of dividing the matrix into sub-blocks of small matrices to find the inversion or the multiplication of matrices. Strassen's algorithm breaks down the complexity of matrix inversion and multiplication from $O(n^3)$ to $O(n^{2.807})$ [48]. Block LU also improves computing efficiency by speeding up the execution time and utilizing memory hierarchies efficiently [49]. The details of both algorithms are illustrated in the next sections. It is worth noting that although the reduction in complexity of matrix inversion might be small with Strassen, the reduction will be per matrix inverse. In other words, the aggregated total saving in the estimation time (ET) of the MMSE estimator will be large, since the MMSE includes more than one matrix inversion operation as shown in Equation (7).

A. STRASSEN's ALGORITHM

In this algorithm, the matrix inversion or multiplication is calculated by partitioning the matrix into smaller square matrices. Hence, in Strassen's algorithm these operations are applied on small sub matrices instead of applying the matrix operations directly on one large matrix. In this algorithm, the matrix inversion will convert into matrix inversion and multiplication. Strassen's algorithms for the matrix multiplication and inversion are explained as follows:

1) MATRIX MULTIPLICATION IN STRASSEN's ALGORITHM

The matrix multiplication in Strassen's algorithm is faster than the traditional multiplication algorithms. In the equation expression, Z is the result of multiplying two square matrices X and Y, (Z = XY). According to the conventional matrix multiplication algorithms, the complexity of this multiplication is $O(N^3)$ [50] while using Strassen's method the complexity of Z will be of order $O(N^{2.807})$. In the method, since the matrices are partitioned into half sized blocks, the matrices of (Z = XY) can be written in the following form:

$$\begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}\begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \quad (25)$$

Algorithm 3 Strassen's Algorithm for Matrix Multiplication

Input:
Matrix X, Matrix Y, Block_size (B_S)
Output:
Matrix Z, multiplication of X and Y

```
function Z = Strassen_mul (X, Y, B_S)
Read the dimension (n) of X
if dimension of n is not equal to power of 2 then
Error ('Enter matrix with power of 2 dimension ')
End if
if n is less or equal to B_S
Z = X*Y
else
Divide the current dimension of n,(z = n / 2)
Define two indexes i = 1:z; j = z+1:n;
R1 = Strassen_mul(X(i, i)+X(j, j), Y(i, i)+Y(j, j), B_S)
R2 = Strassen_mul(X(j, i)+X(j, j), Y(i, i), B_S)
R3 = Strassen_mul(X(i, i), B(i, j)-Y(j, j), B_S)
R4 = Strassen_mul(X(j, j), B(j, i)-Y(i, i), B_S)
R5 = Strassen_mul(X(i, i)+X(i, j), Y(j, j), B_S)
R6 = Strassen_mul(X(j, i)−X(i, i), Y(i, i)+Y(i, j), B_S)
R7 = Strassen_mul(X(i, j)−X(j, j), YB(j, i)+Y(j, j), B_S)
Z = [R1+R4−R5+R7 R3+R5; R2+R4 R1+R3−R2+R6];
End if
Return (Z)
End function
```

Then the matrix multiplication can be calculated as follows:

$$\begin{aligned} R_1 &= (X_{11} + X_{22})(Y_{11} + Y_{22}) \\ R_2 &= (X_{21} + X_{22})\, Y_{11} \\ R_3 &= X_{11}(Y_{11} + Y_{22}) \\ R_4 &= X_{22}(Y_{21} + Y_{11}) \\ R_5 &= (X_{11} + X_{12})\, Y_{22} \\ R_6 &= (X_{21} + X_{11})(Y_{11} + Y_{12}) \\ R_7 &= (X_{12} + X_{22})(Y_{21} + Y_{22}) \end{aligned} \tag{26}$$

The overall results of matrix multiplication are:

$$\begin{aligned} Z_{11} &= R_1 + R_4 - R_5 + R_6 \\ Z_{12} &= R_3 + R_5 \\ Z_{21} &= R + R_4 \\ Z_{22} &= R_1 + R_3 - R_2 + R_6 \end{aligned} \tag{27}$$

It is worth noting that, Strassen's algorithm requires 7 multiplications, while in the traditional multiplication algorithms requires 8 multiplications. Several researchers have studied Strassen's algorithm such as [51], [52]. The details of Strassen's matrix multiplication algorithm is illustrated in Algorithm 3.

2) MATRIX INVERSION IN STRASSEN's ALGORITHM

As mentioned earlier, the calculation of matrix inversion should be achieved by breaking down the matrix inversion into multiplications of several matrices. To find the inverse of $Z = X^{-1}$ for a square matrix X, the matrix (X) should be divided into half sub matrices. The size of matrix X is $N = m2^k$, where m and k are positive integer numbers and 2^k is the size of the sub-matrices. Strassen's algorithm of matrix inversion has been studied broadly [53], [54]. The steps of Strassen's matrix inversion can be expressed as follows:

$$\begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \tag{28}$$

Then the matrix inversion can be calculated as follows:

$$\begin{aligned} R_1 &= X_{11}^{-1} \\ R_2 &= X_{21} x R_1 \\ R_3 &= R_1 x X_{12} \\ R_4 &= X_{21} x R_3 \\ R_5 &= R_4 - X_{22} \\ R_6 &= R_5^{-1} \end{aligned} \tag{29}$$

The final results of matrix inversion are:

$$\begin{aligned} Z_{12} &= R_3 x R_6 \\ Z_{21} &= R_6 x R_2 \\ Z_{11} &= R_1 - R_3 x Z_{21} \\ Z_{22} &= -R_6 \end{aligned} \tag{30}$$

The algorithm for the previous equations (28, 29, 30) can be illustrated as in Algorithm 4 below. Algorithm 4 is reformulated from [54].

3) COMPLEXITY ANALYSIS OF STRASSEN's ALGORITHM

The traditional matrix multiplication, such as two matrices of size (2×2) requires 8 multiplications. Hence, its complexity can be calculated as follows:

$$T(n) = 8T\left(\frac{n}{2}\right) + O\left(n^2\right) \tag{31}$$

where, $n \geq 2$, solving Equation (31) using master theorem,

$$T(n) = O\left(n^{\log_2 8}\right) = O\left(n^3\right) \tag{32}$$

Equation 34 illustrates that the complexity of traditional multiplication for the smaller 2×2 matrix is $O\left(n^3\right)$, while, in Strassen's algorithm the multiplication of two matrices of size 2×2 requires 7 multiplications, and the complexity is $O\left(n^{2.807}\right)$.

$$T(n) = 7T\left(\frac{n}{2}\right) + O\left(n^2\right) \tag{33}$$

$$T(n) = O\left(n^{\log_2 7}\right) = O\left(n^{2.807}\right) \tag{34}$$

Likewise, the matrix inversion requires about $6/5(n^{\log_2 7})$ multiplications. Hence, in Strassen's algorithm the complexity of

Algorithm 4 Strassen's Algorithm for Matrix Inversion

Input:
Matrix X, Block_size (B_S)
Output:
Matrix Z, Inverse of Matrix X

```
function Z = Strassen_inv (X, B_S)
Read the dimension (n) of X
if dimension of n is not equal to power of 2 then
Error ('Enter matrix with power of 2 dimension ')
End if
if n is less or equal to B_S
Z = inv(X)
else
Divide the current dimension of n, (z = n / 2)
Define two indexes i = 1:z; j = z+1:n
R1 = Strassen_inv(X(i, i), B_S);
R2 = Strassen_mul(X(j, i), R1, B_S)
R3 = Strassen_mul(R1,X(i, j), B_S)
R4 = Strassen_mul(X(j, i), R3, B_S)
R5 = R4-X(j, j)
R6 = Strassen_inv(R5, B_S);
Z(i, j) = Strassen_mul(R3, R6, B_S);
Z(j, i) = Strassen_mul(R6, R2, B_S)
Z(i, i) = R1-Strassen_mul(R3, Z(j, i), B_S)
Z(j, j) = -R6
End if
Return (Z)
End function
```

matrix inversion is considered the same as the matrix multiplication $O(n^{2.807})$. Details of the derivation of Strassen's inversion complexity is explained in [55]. In conclusion, Strassen's algorithm is faster than the traditional methods.

4) CHALLENGE OF STRASSEN's ALGORITHM

The main challenge in Strassen's algorithm is that the dimension of the matrix X must be of order of 2^k, where, k is an integer [17]. Therefore, this might be the main reason that makes this algorithm uncommon for the mathematical operations of matrices. On the other hand, the dimension of the channel information matrices, where we need to reduce complexity is not always of power 2. Hence, in this research, two methods have been studied to generalize Strassen's algorithm. These methods are illustrated as follows:

a: INCREASING THE DIMENSION OF MATRIX TO THE NEXT HIGHER POWER OF 2

As mentioned earlier, to use Strassen's algorithm, the dimension of matrix must be a power of 2. Hence, to solve this limitation, it can scale up the size of matrix to the next power of 2 by adding rows and columns of zeros with ones on the main diagonal at the end of the input matrix. This can be achieved using the Matlab function "*nextpow2(n)*" which returns the next power of 2. The following algorithm and numerical example clarify the operation of this technique.

Algorithm 5 Generalization of Strassen' Algorithm Using Next Power of 2

Input:
Matrix M, Block_size
Output:
Inverse of M, (M^{-1})

```
Read matrix M
Check the dimension (n) of M
if dimension (n) of M is not a power of 2
//Check if M is a square matrix
if size(M,1) ~ = size(M,2)
   error('The matrix must be square.')
end if
Scaling up the size of M to the next power of 2
Set 1 for the main diagonal
Set 0 for the additional rows and columns
M^-1 = Strassen_inv (B, Block_size);
Return the original size of M
Return (M^-1)
End if
```

b: NUMERICAL EXAMPLE

M is a matrix of dimension 5x5 as shown below. Then to find the inverse of M by using Strassen's algorithm, the steps in Algorithm 5 above are applied as follows.

$$M = \begin{bmatrix} 1 & 2 & 5 & 7 & 8 \\ 3 & 2 & 7 & 2 & 1 \\ 5 & 1 & 2 & 6 & 9 \\ 2 & 2 & 4 & 5 & 6 \\ 9 & 7 & 1 & 5 & 2 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 2 & 5 & 7 & 8 & 0 & 0 & 0 \\ 3 & 2 & 7 & 2 & 1 & 0 & 0 & 0 \\ 5 & 2 & 7 & 2 & 1 & 0 & 0 & 0 \\ 2 & 2 & 4 & 5 & 6 & 0 & 0 & 0 \\ 9 & 7 & 1 & 5 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Then the next power of 2 after 5 is 8, hence, the size will be M_1^{-1} and M^{-1}, as shown at the bottom of the next page.

It is worth stating that the aforementioned method can work with Strassen's algorithm. However, this contradicts the aim of this research, which is to reduce the complexity of acquiring the channel information of large channel matrices. In large channels matrices, the computation complexity will be increased with scaling up the size of matrix M to the next power of 2, until when the additional rows and columns are zeros. Hence, it is an inefficient approach in this research. To verify the system performance with this technique, a simulation test is conducted with the following settings, 5UEs, 1VBS, 64 antennas at the VBS, 4 antennas at UEs, 1.4 BW. The results in Fig. 12 below clarify the performance of the network before and after using

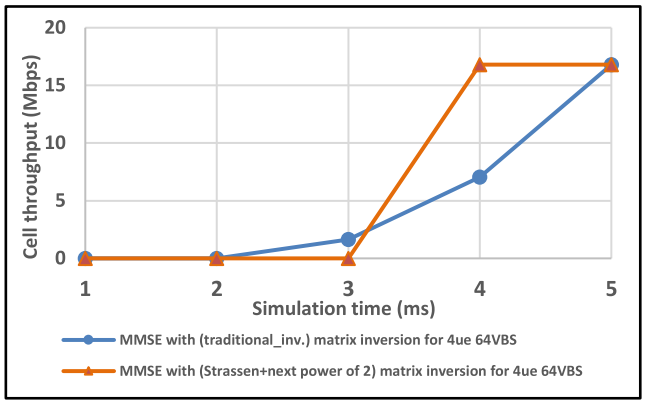

FIGURE 12. Network performance using Strassen with next power of 2.

Strassen's algorithm with the technique of next power of 2. The degradation in the performance comes from enlarging the original size of the matrix into a larger size of power of 2. In the next section, the block LU decomposition is proposed as a novel idea to tackle the problem of dimensions in Strassen's algorithm. It is worth stating that in 5G networks the target end-to-end latency is 1ms as mentioned earlier. Hence, any reduction in the estimation time will contribute to the overall latency minimization.

B. GENERALIZATION OF STRASSEN's ALGORITHM USING BLOCK LU DECOMPOSITION

Block LU decomposition is an approach used to speed up the operation of matrices [56]. The idea is to break down the high computation overhead of the big matrix into a set of smaller blocks of sub-matrices. The Block LU is used in this research for the purpose of generalization of Strassen's algorithm, which requires the dimension of power 2. It is possible to make the size of the block of sub-matrices equal to power of 2 (n = 2k) through Block LU, to fit with the requirement of Strassen's algorithm. The Block LU method is an improvement to the standard LU factorization approach. In Block LU decomposition, the operations of matrix inversion or multiplication start after dividing the upper and lower triangular into sub-blocks of small matrices. As a result of this division, the length of the operations vectors will reduce from the original size of matrix N into n, which is the size of the sub-blocks. This approach has been studied extensively, such as in [57] and [58]. In general, the procedure of this approach can be classified into three main parts: partitioning of the original matrix into L and U triangular; dividing the lower and upper triangular into sub-blocks with a size of n; and then finding the inverse of each block and augmenting the results of sub matrices to determine the inverse of the original matrix. The details of Block LU decomposition and the combination of both Block LU- Strassen's algorithms will be explained in the following sections.

1) PART 1 (MATRIX DECOMPOSITION INTO L AND U MATRICES)

Y is a square matrix with order N, the decomposition of Y into a product of two upper and lower matrices (Y = LU) is illustrated in equation (35) below:

$$\begin{bmatrix} y_{11} & y_{12} & \dots & y_{1N} \\ y_{21} & y_{22} & \dots & y_{2N} \\ y_{31} & y_{32} & \dots & y_{3N} \\ \dots & & & \\ y_{N1} & y_{N2} & \dots & y_{NN} \end{bmatrix} = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ l_{31} & l_{32} & l_{33} & \\ \dots & & & \\ l_{N1} & l_{N2} & \dots & l_{NN} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1N} \\ & u_{22} & \dots & u_{2N} \\ & & u_{33} & \dots \\ & & & \dots \\ & & & u_{NN} \end{bmatrix} \tag{35}$$

where: the values of the main diagonal of the lower triangular l_{ii} are set to one and also the unwritten values at both L and U set to zeros. The rest of values can be calculated using equations 38 and 39.

$$u_{ij} = y_{ij} - \sum_{z=1}^{j-1} l_{jz} u_{zj} \tag{36}$$

$$l_{ij} = \frac{1}{u_{jj}} (y_{ij} - \sum_{z=1}^{j-1} l_{iz} u_{zj}) \tag{37}$$

$$M_1^{-1} = \begin{bmatrix} 0.0805 & 0.1421 & 0.2151 & -0.4667 & 0.0390 & 0 & 0 & 0 \\ -0.9069 & -0.3044 & -0.4075 & 1.8667 & 0.0138 & 0 & 0 & 0 \\ -0.0742 & 0.1346 & -0.0264 & 1.8667 & -0.0516 & 0 & 0 & 0 \\ 1.5145 & 0.2365 & 0.2340 & 0.1333 & 0.1711 & 0 & 0 & 0 \\ -0.9371 & -0.2327 & -0.1132 & 1.667 & -0.1258 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} 0.0805 & 0.1421 & 0.2151 & -0.4667 & 0.0390 \\ -0.9069 & -0.3044 & -0.4075 & 1.8667 & 0.0138 \\ -0.0742 & 0.1346 & -0.0264 & 0.1333 & -0.0516 \\ 1.5145 & 0.2365 & 0.2340 & -2.4667 & 0.1711 \\ -0.9371 & -0.2327 & -0.1132 & 1.6667 & -0.1258 \end{bmatrix}$$

The pivoting matrix P is also calculated by decomposition PY instead of Y, (PY = LU) to maintain high numerical stability and accuracy; where P is a permutation matrix of the rows, which includes 1s and 0s provided that the rest of the row and the column of each 1s are zeros. P matrix has no effect on the final result of matrix inverse.

2) PART 2 (PARTITIONING L AND U INTO SUB-BLOCKS)

In this stage, the equation of the original LU factorization can be written in the form of sub matrices as follows:

$$\begin{bmatrix} p_1 & 0 \\ 0 & p_2 \end{bmatrix} \begin{bmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{bmatrix} = \begin{bmatrix} L_1 & 0 \\ L_2 & L_3 \end{bmatrix} \begin{bmatrix} U_1 & U_2 \\ 0 & U_3 \end{bmatrix} \quad (38)$$

Hence, from equation 40 we can get the following matrices:

$$P_1 Y_1 = L_1 U_1 \quad (39)$$
$$P_1 Y_2 = L_1 U_2 \quad (40)$$
$$P_2 Y_3 = L_2 U_1 \quad (41)$$
$$P_2 Y_4 = L_2 U_2 + L_3 U_3 \quad (42)$$

Then the following steps will be followed to find the sub matrices:

1. If the size of the sub matrix Y_1 reaches the desired dimension (n), then decompose Y_1 to find L_1, U_1, P_1 and find the L_1^{-1}, U_1^{-1} elsewhere, continue partitioning into smaller matrices.
2. Then, calculate $\hat{L}_2$, U2 from L_1, U_1, Y_2 and Y_3.
3. Then, determine $\hat{Y} = Y_4 - \hat{L}_2 U_2$, where, $\hat{L}_2 = Y_3 U_1^{-1}$ if the dimension of $\hat{Y}$ equal to n then decopmse $\hat{Y}$ to find L_3, U_3, elsewhere, continue partitioning into smaller matrices.
4. Obtain L_2 from P_2 and $\hat{L}_2$.

$$\left(\hat{L}_2\right)_{ij} = \frac{1}{(U_1)_{ii}}((Y_3)_{ij} - \sum_{z=1}^{i-1} \left(\hat{L}_2\right)_{iz} (U_1)_{zj}) \quad (43)$$
$$(U_2)_{ij} = \frac{1}{(L_1)_{ii}}((Y_2)_{ij} - \sum_{z=1}^{i-1} (L_1)_{iz} (U_2)_{zj}) \quad (44)$$

After obtaining $\hat{L}_2$ and U_2 now it can find $\hat{Y} = Y_4 - \hat{L}_2 U_2$, then if the dimension of $\hat{Y}$ is equal to the block size n it can be decomposed to find L_3, U_3. It is worth mentioning that at the stage of calculating Y_1 and $\hat{Y}$ if they are not small enough (size larger than n), the processes of partitioning and calculating the intermediate sub matrices continue recursively.

3) PART 3 (AUGMENTING THE RESULTS OF SUB-MATRICES AND FINDING THE FINAL INVERSE)

The inverse of the original lower triangular L, the upper triangular U matrices can be calculated in parallel, since both of them are independent, by augmenting all the sub-matrices of L (L_1, L_2, L_3) and U (U_1, U_2, U_3), and obtaining the permutation matrix P by augmenting P_1 and P_2 as follows. Algorithm 6 is recalled from the Block LU algorithm in [57].

$$L^{-1} = \begin{bmatrix} L_1^{-1} & 0 \\ -L_3^{-1} L_2 L_1^{-1} & L_3^{-1} \end{bmatrix} \quad (45)$$

Algorithm 6 Block LU decomposition [57]

Input:
N × N square matrix Y, block size n
Output:
inverse of Y^{-1}

```
function BlockLU (Y)
if the order of Y equal to the block size
(L, U, P) = lu (Y)
L^-1 = inverse (L)
U^-1 = inverse (U)
else
Divide Y into Y1, Y2, Y3, Y4
(L1^-1, U1^-1, P1) BlockLU (Y1)
Calculate U2 from L1^-1, P1 and Y2
Calculate L̂2 from U1^-1 and Y3
Calculate Ŷ = Y4 − L̂2 U2
(L3^-1, U3^-1, P2) BlockLU (Ŷ)
Calculate L2 from P2 and L̂2
Augmenting L1^-1, L2, L3^-1 to find L^-1
Augmenting U1^-1, L2, U3^-1 to U^-1
Augmenting P1 and P2 to find P
End if
Y^-1 = U^-1 L^-1 P
Return (Y^-1)
End function
```

$$U^{-1} = \begin{bmatrix} U_1^{-1} & -U_1^{-1} U_2 U_3^{-1} \\ 0 & U_3^{-1} \end{bmatrix} \quad (46)$$
$$P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix} \quad (47)$$

Finally, the inverse of matrix Y will be:

$$Y^{-1} = U^{-1} L^{-1} P \quad (48)$$

C. BLOCK LU – STRASSEN's INVERSION

The inversion of Strassen's algorithm is faster than the traditional inverse algorithms used in traditional MMSE, since the complexity has been reduced into $O(N^{2.807})$. In this work, and to generalize Strassen's algorithm, the Block LU decomposition is proposed with modification. The idea is to use Strassen's inversion as the core of the Block LU decomposition instead of traditional inversion, as shown in Algorithm 7. The proposed algorithm will ensure lower complexity and hence lower latency compared with Algorithm 6.

VIII. SIMULATION RESULTS AND DISCUSSION

This section is divided into two parts. The first part presents the results for deploying the MapReduce framework in the C-RAN network. The second part contains results for implementing fast matrix algorithms in the MMSE. This paper applies MATLAB R2016a to run the simulation tests. The simulation parameters setting are illustrated in Table 4.

Part 1 (Simulation Results of C-RAN With MapReduce): In this part, the results of deploying MapReduce in C-RAN

Algorithm 7 The proposed Block LU – Strassen

Input:
N × N square matrix Y, block size(B_S) = $2^{\wedge L}$, L: is integer no.
Output:
inverse of Y^{-1}

```
function BlockLU (Y)
if the order of Y equal to the block size
(L, U, P) = lu (Y)
L-1 = Strassen_inv (L, B_S)
U-1 = Strassen_inv (U, B_S)
else
Partition matrix (Y) into (Y1,Y2,Y3,Y4)
(L1^-1, U1^-1, P1) BlockLU (Y1)
Calculate U2 from L1^-1, P1 and Y2
Calculate L̂2 from U1^-1 and Y3
Calculate Ŷ = Y4 − L̂2 U2
(L3^-1, U3^-1, P2) BlockLU (Ŷ)
Calculate L2 from P2 and L̂2
Augmenting L1^-1, L2, L3^-1 to find L^-1
Augmenting U1^-1, L2, U3^-1 to U^-1
Augmenting P1 and P2 to find P
End if
Y^-1 = U^-1 L^-1 P
Return (Y^-1)
End function
```

TABLE 4. Simulation parameters.

Transmission	Uplink
Transmission mode	Closed Loop Spatial Multiplexing (CLSM)
Simulation type	Link level
Network size	2-16 VBSs with 5 UEs per VBS
Carrier frequency	1.9 GHz
Bandwidth	1.4 MHz
UE mobility speed	100 Km/hr
Estimation Algorithm	MMSE and LS
Scheduling	Round Robin
Simulation length	10 TTI

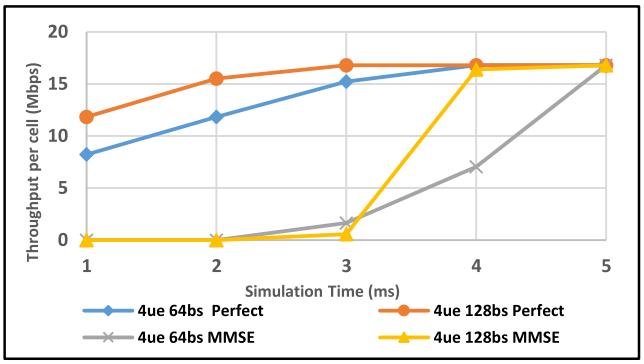

FIGURE 13. Test in above figure shows the effect of expansion on the cell throughput with (64 and 128 antennas) per VBS. using MMSE and perfect estimation algorithm.

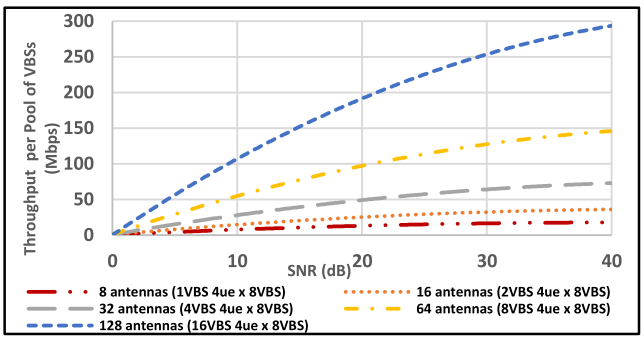

FIGURE 14. Throughput per pool of VBSs (group of VBSs with 8 RRHs) using MapRedcue.

are presented. At the start, to quantify the amount of high computational complexity for large channel matrices on the performance of the network, the first simulation test is conducted with 64 and 128 antennas at one VBS using MMSE estimator with 1.4 MHz bandwidth. To establish a logical line of reasoning, one should note that in this work, the RRH is deployed with a single antenna, therefore the words antenna and RRH are alternated to describe the same concept. For the purpose of comparison, the test has been repeated using perfect channel estimation. As shown in Fig.13, the results reveal that the MMSE and the perfect channel estimation differ significantly in terms of data throughput. The perfect estimation or the theoretical estimation process means that no previous acquisition processing is necessary because of the perfect knowledge of CSI at the VBS. Consequently, unlike the real estimator (MMSE), the perfect estimator eradicates the issue of high computation overhead in obtaining the CSI with a large number of antennas in the VBS. The subsequent test uses the distributed processing in the C-RAN architecture to reduce the computation complexity faced with a large number of antennas.

The aim of deploying MapReduce is to switch the technique of processing in C-RAN from centralized to a distributed one. This reduces the size of H and minimizes the acquisition time of the CSI.

Figure 14 shows the possibility of expanding the number of antennas without raising the computation overhead, while keeping the performance of the network the same for data throughput. While Fig.15 illustrates the increase in spectral efficiency, Fig.16 shows its percentage. In other words, the scalability of C-RAN with MapReduce increases the spectral efficiency of the network as the number of distributed RRHs rises. Simultaneously, the estimation time and the overall response time (RT) remain constant within the time of the applied group. This is due to processing small manageable channel matrices instead of a large matrix with high computational complexity.

The response time can increase dramatically in the central processing approach, as shown in Fig.17. Such increases can be controlled throughout the clustering with the MapReduce approach. Here, the response time has been maintained for the 8th, 16th and 32th group order. The response is a crucial factor

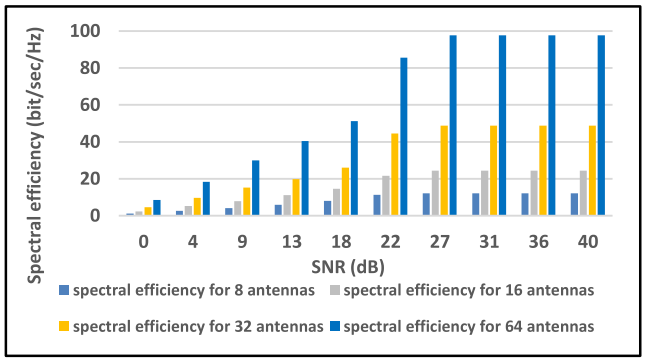

FIGURE 15. Spectral efficiency with the rise the number of antennas.

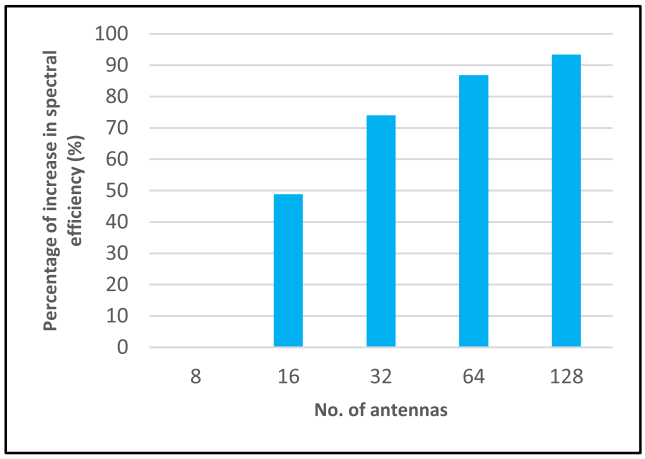

FIGURE 16. Percenatge of increase in the spectral efficiency compared to the lagacy 8 antennas system.

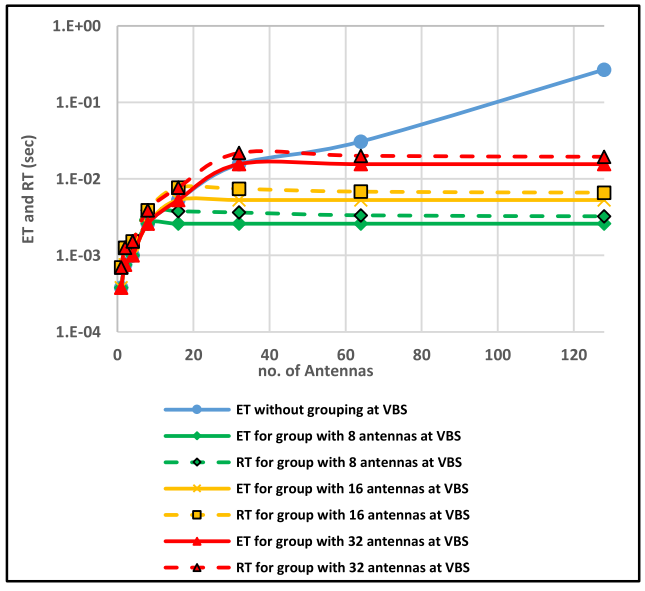

FIGURE 17. Total response time for three groups of 8, 16 and 32 antennas at each VBS.

when considering next generation networks that require low end-to-end latency.

Figure 18 also demonstrates the gain in the CSI acquisition time for different groups of RRHs. The figures demonstrate

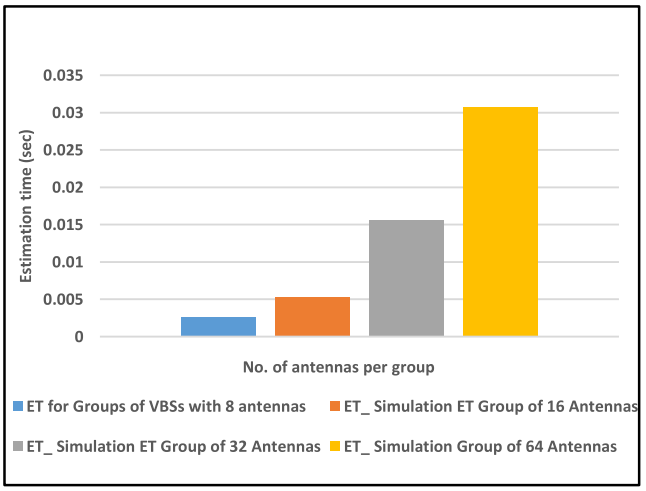

FIGURE 18. Average estimation time for different groups of antennas: 8,16,32 and 64 antennas.

that the total estimation time can be minimized when grouping is performed. In addition, the estimation time can be limited by allocating a group of RRHs to each VBS, and this reduces the computational overhead of big channel matrices. The advantage is that it meets the low coherence time requirements for high carrier frequencies and UE speeds, and then to improve the accuracy of the CSI, since it conveys the state of the communication link between the UEs and the VBSs. The proposed distribution approach is beneficial in the next generation of 5G networks since the latency will be minimized to a significant level, as shown in Fig.2, to meet the future critical time technologies. Simultaneously, a large number of antennas can be used in a scalable manner without raising the problem of the acquisition overhead in the whole network. The advantage is that in the cloud, the data, and the CSI can be completely distributed between VBSs [27]. Therefore, instead of employing a large number of antennas /RRHs per VBS (that causes a high overhead, with MapReduce) a set of VBS with the pre-specified group of RRHs have been used to obtain the CSI.

The results in Fig.19 demonstrate the possibility of reducing the estimation time for the CSI based on the chosen group size. Certainly, when the group size starts to increase, the size of the channel matrix also begins to increase as well. Hence, with a smaller group of antennas (8 antennas), the higher gain in the percentage of reduction RT is observed, and this percentage starts to decrease when increasing the size of the group. In Fig.20, both the simulation and the theoretical results are drawn for the purpose of comparison between the simulation and the analytical results in terms of data throughput. The results show almost the same trend between analytical and simulation results. The result in Fig. 21 presents a comparison between the overall response time of the simulation results and the theoretical calculations with the number of deployed servers. The result indicates that as the number of servers increases, the estimation time and

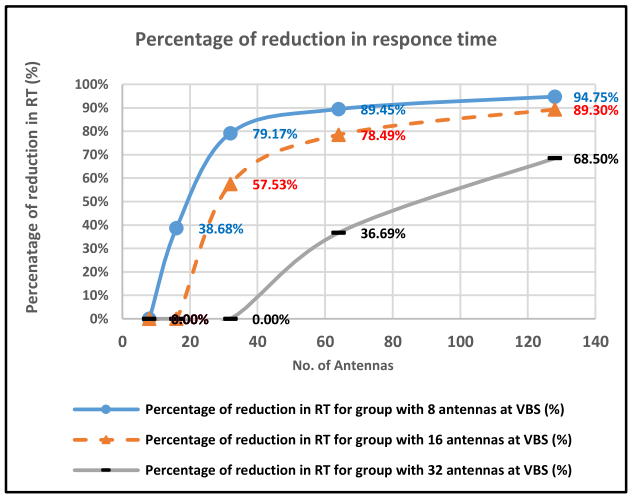

FIGURE 19. Reduction gain (%) in response time for different RRH groups.

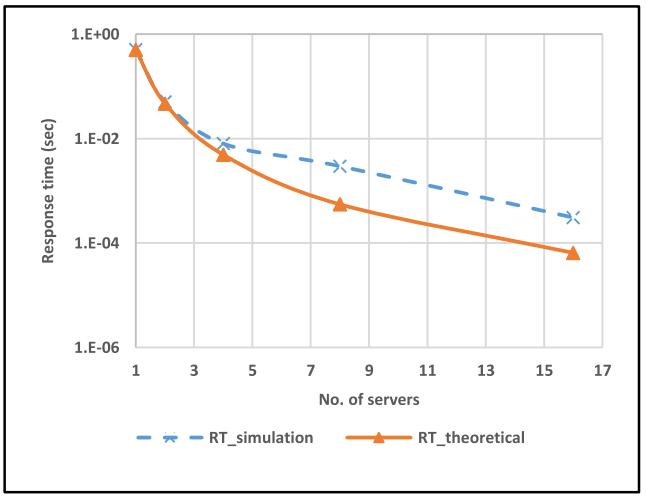

FIGURE 21. Reduction in response time versus no. of processors/ mappers.

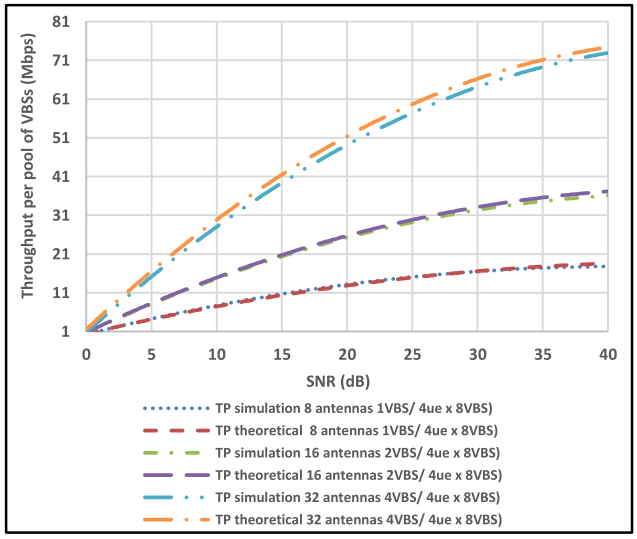

FIGURE 20. Throughput of pool of VBSs (simulation vs. therotical).

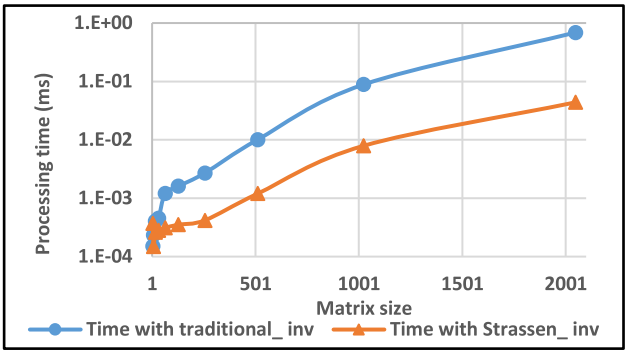

FIGURE 22. Strassen_inv versus. traditional inv. with power of 2 matrices.

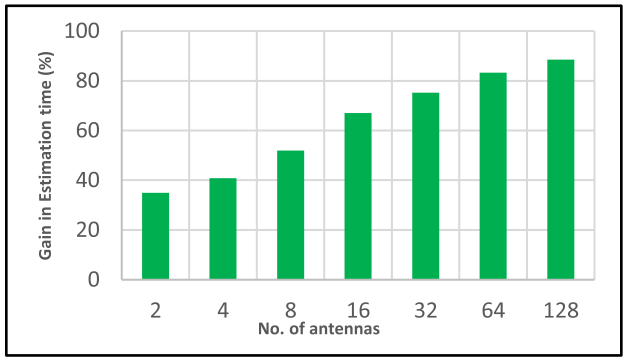

FIGURE 23. gain in estimation time (%) versus no. of antennas using Blocklu-Strassen alogorithm.

the overall response time decreases proportionally. Therefore more VBSs are recommended to be added in C-RAN-based MapReduce to reduce the response time. There is a noticeable difference between the analytical and the simulation results in Fig.21.In the simulation results, there are several parameters under consideration, and the simulation tool facilitates the calculations. While in the analytical method, for the sake of simplicity, fewer parameters have been considered in the calculations.

Part 2 (Simulation Results of C-RAN With Fast Matrix Algorithms): Several tests are conducted to examine the proposed fast matrix algorithms. At the start, to clarify the speed of Strassen's algorithm, a comparison in the processing time is made between Strassen's inverse algorithm and traditional inverse function. The two main points are illustrated in Fig.22, which are that: Strassen's algorithm requires less processing time and it gives more gain in time when scaling up the size of the matrix. However, this test is limited to a dimension of power 2 matrices.

Applying the combination of Strassen's and Block LU enables a considerable reduction in the estimation time of acquiring CSI, due to decreasing the processing time of matrix inversion. The result in Fig. 23 illustrates the percentage of gain in the estimation time of channel information when using block LU – Strassen's algorithms over the traditional inversion. As mentioned earlier, Strassen's algorithm

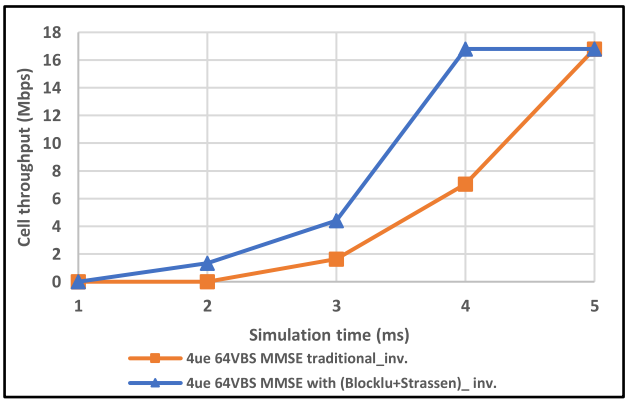

FIGURE 24. Throughput per cell for 64 antennas at the VBS with fast algorithms (Block LU + Strassen).

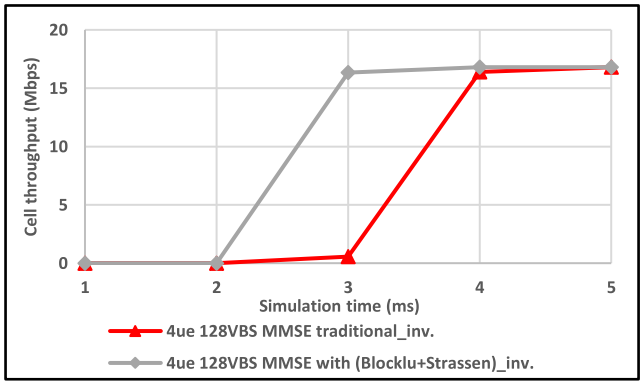

FIGURE 25. Throughput per cell for 128 antennas at the VBS with fast algorithms (Block LU + Strassen).

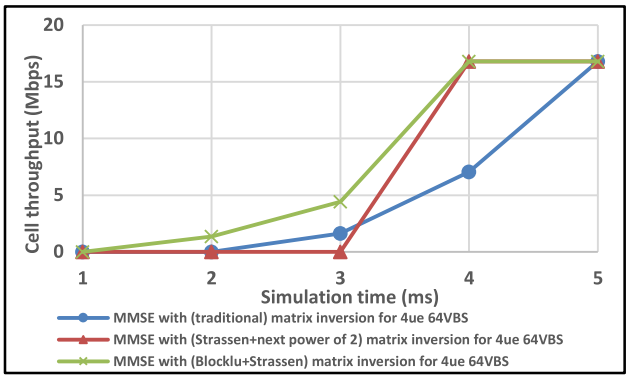

FIGURE 26. System performance comparison between the proposed methods for generalizing Strassen's algorithm.

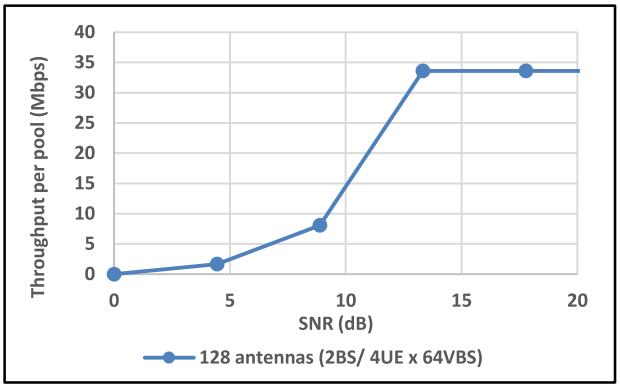

FIGURE 27. Throughput per pool of VBSs for scalable number of antennas (group of 2VBSs with 64 antennas) using MapReduce.

TABLE 5. Summary of complexity reduction.

MMSE estimator	Complexity
MMSE	$O(Y^3)$ where, $Y = N_R * N_T$
MMSE + MapReduce	$O(y^3)$, where, $y = n*N_T$, $n = N_R/k$, k is the group size
MMSE + MapReduce + Blocklu-Strassen	$O(y^{2.807})$

can support scalability. As the number of antennas (or equivalently the size of the channel matrix) increases, the gain of execution time decreases.

The results in Figures 24 and 25 show noticeable improvement in the data throughput of the network when considering block LU–Strassen's algorithms in calculating the matrix inverse in the MMSE estimator for VBSs with 64 and 128 RRHs. The reason is due to the reduction in the processing time of the matrix inversion, which leads to a reduction in the estimation time of the CSI acquisition. Therefore, the accuracy of the CSI is improved by adapting the communication channel more quickly between the UE and the VBS.

Figure 26 demonstrates a comparison of the network performance in terms of data throughput between the two methods that have been used in this research for generalizing Strassen's algorithm. The results show that Stassen-Block LU is more efficient, since it minimizes the initial system delay and speeds up the system response time.

The advantage of reducing the estimation time with Stassen-Block LU - particularly with the case of 64 antennas at the VBS - is that it can increase the size of the group of RRHs in MapReduce to 64 RRHs with acceptable system performance. Hence, the C-RAN network with 128, 256, 512, 1024 RRHs can be represented with groups of 2VBSs, 4VBSs, 8VBSs and 16VBSs respectively, with 64 RRHs per VBS. For instance, Fig. 27, illustrates that 128 antennas can be represented by two VBSs with a group size of 64 RRHs, this scenario is not possible to implement with traditional matrix inversion due to the high execution time of matrix inversion.

It is worth mentioning that both of the proposed techniques (MapReduce and fast matrix algorithms) provide a considerable improvement in the reduction of computational complexity of acquiring CSI. The summary of the overall reduction is illustrated in Table 5.

IX. CONCLUSION

The architecture of C-RAN provides central network management and reduces the expenses of network deployment.

However, it has challenging computational complexity, which leads to limited network scalability. Two novel approaches have been developed for C-RAN architecture to reduce the computational complexity in acquiring channel state information while maintaining network scalability to meet the demand of future 5G networks. The proposed approaches are supported by results that show improvement of the system performance (particularly, the data throughput) due to a significant reduction in the acquisition time. MapReduce, as a distributed processing framework, can minimize per network estimation time, based on the size of the group of antennas. Additionally, fast matrix algorithms have reduced per VBS estimation time, via decreasing the time of execution for the matrix inversion in the MMSE estimator. Further work will involve using dynamic grouping size, based on the optimal available capacity, and furthermore, planning and optimization for the optimal capacity of VBSs of C-RAN in cloud computing requires more investigation.

REFERENCES

[1] G. Liu and D. Jiang, "5G: Vision and requirements for mobile communication system towards year 2020," *Chin. J. Eng.*, vol. 2016, Mar. 2016, Art. no. 5974586.

[2] M. Shafi *et al.*, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[3] K. M. S. Huq, S. Mumtaz, J. Rodriguez, P. Marques, B. Okyere, and V. Frascolla, "Enhanced C-RAN using D2D network," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 100–107, Mar. 2017.

[4] "C-RAN: The road towards green RAN," Mobile China, Hong Kong, China, White Paper ver. 2, 2011.

[5] C. Fan, Y. J. Zhang, and X. Yuan, "Dynamic nested clustering for parallel PHY-layer processing in cloud-RANs," *IEEE Trans. Wireless Commun.*, vol. 15, no. 3, pp. 1881–1894, Mar. 2016.

[6] A. M. Mahmood and A. Al-Yasiri, "Scalable processing in 5G cloud-RAN networks using MapReduce framework," in *Proc. 8th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, 2016, pp. 1–6.

[7] Y. Shi, J. Zhang, and K. Letaief. (2013). "Optimal stochastic coordinated beamforming for wireless cooperative networks with CSI uncertainty." [Online]. Available: https://arxiv.org/abs/1312.0363

[8] Y. Shi, J. Zhang, K. B. Letaief, B. Bai, and W. Chen, "Large-scale convex optimization for ultra-dense Cloud-RAN," *IEEE Wireless Commun.*, vol. 22, no. 2, pp. 84–91, Jun. 2015.

[9] C. Fan, Y. J. Zhang, and X. Yuan, "Scalable uplink processing via sparse message passing in C-RAN," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.

[10] A. Liu and V. K. N. Lau, "Joint power and antenna selection optimization in large cloud radio access networks," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1319–1328, Mar. 2014.

[11] K. Ntougias, D. Ntaikos, and C. B. Papadias. (2015). "Reducing complexity in next-generation MU-MIMO systems." [Online]. Available: https://arxiv.org/abs/1507.04050

[12] J. Park and R. W. Heath, Jr., "Threshold-based antenna selection algorithm for dense cloud radio access networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.

[13] S. Galih, T. Adiono, and A. Kurniawan, "Low complexity MMSE channel estimation by weight matrix elements sampling for downlink OFDMA mobile WiMAX system," *Int. J. Comput. Sci. Netw. Secur.*, vol. 10, no. 2, pp. 280–285, Feb. 2010.

[14] A. Khlifi and R. Bouallegue. (2011). "Performance analysis of LS and LMMSE channel estimation techniques for LTE downlink systems." [Online]. Available: https://arxiv.org/abs/1111.1666

[15] Z. Mao, M. Peng, H. Wang, J. Zhou, and X. Xie, "Low-complexity segment training channel estimation in cloud radio access networks," in *Proc. IEEE 82nd Veh. Technol. Conf. (VTC Fall)*, Sep. 2015, pp. 1–5.

[16] O. Edfors, M. Sandell, J. J. van de Beek, S. K. Wilson, and P. O. Börjesson, "OFDM channel estimation by singular value decomposition," *IEEE Trans. Commun.*, vol. 46, no. 7, pp. 931–939, Jul. 1998.

[17] N. Shariati, E. Björnson, M. Bengtsson, and M. Debbah, "Low-complexity polynomial channel estimation in large-scale MIMO with arbitrary statistics," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 815–830, Oct. 2014.

[18] N. Shariati, E. Bjornson, M. Bengtsson, and M. Debbah, "Low-complexity channel estimation in large-scale MIMO using polynomial expansion," in *Proc. IEEE 24th Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2013, pp. 1157–1162.

[19] Y. Xu, G. Yue, and S. Mao, "User grouping for massive MIMO in FDD systems: New design methods and analysis," *IEEE Access*, vol. 2, pp. 947–959, 2014.

[20] Y. Wang, C. Li, Y. Huang, D. Wang, T. Ban, and L. Yang, "Energy-efficient optimization for downlink massive MIMO FDD systems with transmit-side channel correlation," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7228–7243, Sep. 2016.

[21] H. Ji *et al.* (2015). "Overview of full-dimension MIMO in LTE-advanced pro." [Online]. Available: https://arxiv.org/abs/1601.00019

[22] T. V. Krishnamurthy and R. Shetty, *4G: Deployment Strategies and Operational Implications Managing Critical Decisions in Deployment of 4G/LTE Networks and their Effects on Network Operations and Business*. New York, NY, USA: Apress, 2014.

[23] D. Liu, S. Han, C. Yang, and Q. Zhang, "Semi-dynamic user-specific clustering for downlink cloud radio access network," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2063–2077, Apr. 2016.

[24] K. Wu and D. Li, "Channel estimation for large antenna systems," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, May 2015, pp. 1–5.

[25] A. Alameer and A. Sezgin, "Joint beamforming and network topology optimization of green cloud radio access networks," in *Proc. 9th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, 2016, pp. 375–379.

[26] V. N. Ha, L. B. Le, and N.-D. Dao, "Energy-efficient coordinated transmission for cloud-RANs: Algorithm design and trade-off," in *Proc. 48th Annu. Conf. Inf. Sci. Syst. (CISS)*, 2014, pp. 1–6.

[27] X. Chen, N. Li, J. Wang, C. Xing, L. Sun, and M. Lei, "A dynamic clustering algorithm design for C-RAN based on multi-objective optimization theory," in *Proc. IEEE 79th Veh. Technol. Conf. (VTC Spring)*, May 2014, pp. 1–5.

[28] K. Boulos, M. El Helou, and S. Lahoud, "RRH clustering in cloud radio access networks," in *Proc. Int. Conf. Appl. Res. Comput. Sci. Eng. (ICAR)*, 2015, pp. 1–6.

[29] D. Pompili, A. Hajisami, and H. Viswanathan, "Dynamic provisioning and allocation in cloud radio access networks (C-RANs)," *Ad Hoc Netw.*, vol. 30, pp. 128–143, Jul. 2015.

[30] A. Davydov, G. Morozov, I. Bolotin, and A. Papathanassiou, "Evaluation of joint transmission CoMP in C-RAN based LTE-A HetNets with large coordination areas," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2013, pp. 801–806.

[31] Y. Du and G. De Veciana, "'Wireless networks without edges': Dynamic radio resource clustering and user scheduling," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 1321–1329.

[32] N. Agata, A. Agata, and K. Nishimura, "A design algorithm for ring topology centralized-radio-access-network," in *Proc. 17th Int. Conf. Opt. Netw. Design Modeling (ONDM)*, Apr. 2013, pp. 173–178.

[33] J. Tang, W. P. Tay, T. Q. S. Quek, and B. Liang, "Towards system cost minimization in cloud radio access network," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, 2015, pp. 1460–1464.

[34] M. Peng, C. Wang, V. Lau, and H. V. Poor, "Fronthaul-constrained cloud radio access networks: Insights and challenges," *IEEE Wireless Commun.*, vol. 22, no. 2, pp. 152–160, Apr. 2015.

[35] C. Fan, Y. J. Zhang, and X. Yuan, "Advances and challenges toward a scalable cloud radio access network," *IEEE Commun. Mag.*, vol. 54, no. 6, pp. 29–35, Jun. 2016.

[36] J. Liu, S. Xu, S. Zhou, and Z. Niu, "Redesigning fronthaul for next-generation networks: Beyond baseband samples and point-to-point links," *IEEE Wireless Commun.*, vol. 22, no. 5, pp. 90–97, Oct. 2015.

[37] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[38] C. Cox, *An Introduction to LTE: LTE, LTE-Advanced, SAE and 4G Mobile Communications*. Hoboken, NJ, USA: Wiley, 2012.

[39] M. N. I. Khan and M. J. Alam, "Noise reduction algorithm for LS channel estimation in OFDM system," in *Proc. 15th Int. Conf. Comput. Inf. Technol. (ICCIT)*, 2012, pp. 310–315.
[40] J.-C. Shen, J. Zhang, K.-C. Chen, and K. B. Letaief, "High-dimensional CSI acquisition in massive MIMO: Sparsity-inspired approaches," *IEEE Syst. J.*, vol. 11, no. 1, pp. 32–40, Mar. 2017.
[41] M. Rupp, S. Schwarz, and M. Taranetz, *The Vienna LTE-Advanced Simulators*. Singapore: Springer, 2016.
[42] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-Advanced for Mobile Broadband*. Orlando, FL, USA: Academic, 2013.
[43] A. Pal and S. Agrawal, "An experimental approach towards big data for analyzing memory utilization on a hadoop cluster using HDFS and MapReduce," in *Proc. 1st Int. Conf. Netw. Soft Comput. (ICNSC)*, 2014, pp. 442–447.
[44] S. Pratschner, E. Zöchmann, and M. Rupp, "Low complexity estimation of frequency selective channels for the LTE-A uplink," *IEEE Wireless Commun. Lett.*, vol. 4, no. 6, pp. 673–676, Dec. 2015.
[45] K. Salah and J. M. A. Calero, "Achieving elasticity for cloud MapReduce jobs," in *Proc. IEEE 2nd Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2013, pp. 195–199.
[46] D. Warneke and O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 985–997, Jun. 2011.
[47] K. Salah, "A queueing model to achieve proper elasticity for cloud cluster jobs," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jul. 2013, pp. 755–761.
[48] V. Strassen, "Gaussian elimination is not optimal," *Numer. Math.*, vol. 13, no. 4, pp. 354–356, 1969.
[49] J. Chen, K. Ji, Z. Shi, and W. Liu, "Implementation of block algorithm for LU factorization," in *Proc. WRI World Congr. Comput. Sci. Inf. Eng.*, 2009, pp. 569–573.
[50] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," *ACM Trans. Algorithms*, vol. 1, no. 1, pp. 2–13, 2005.
[51] S. Yoshizawa, Y. Yamauchi, and Y. Miyanaga, "A complete pipelined MMSE detection architecture in a 4×4 MIMO-OFDM receiver," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 2486–2489.
[52] D. H. Bailey and H. R. P. Gerguson, "A Strassen–Newton algorithm for high-speed parallelizable matrix inversion," in *Proc. ACM/IEEE Conf. Supercomput.*, Nov. 1988, pp. 419–424.
[53] R. Steffen, "Exascale ready work-optimal matrix inversion," Ph.D. dissertation, Dept. Inform., Karlsruhe Inst. Technol., Karlsruhe, Germany, 2012.
[54] M. D. Petković and P. S. Stanimirović, "Generalized matrix inversion is not harder than matrix multiplication," *J. Comput. Appl. Math.*, vol. 230, no. 1, pp. 270–282, 2009.
[55] L. Kronsjö and D. Shumsheruddin, *Advances in Parallel Algorithms*. Hoboken, NJ, USA: Wiley, 1992.
[56] M. K. Jaiswal and N. Chandrachoodan, "FPGA-based high-performance and scalable block LU decomposition architecture," *IEEE Trans. Comput.*, vol. 61, no. 1, pp. 60–72, Jan. 2012.
[57] J. Liu, Y. Liang, and N. Ansari, "Spark-based large-scale matrix inversion for big data processing," *IEEE Access*, vol. 4, pp. 2166–2176, 2016.
[58] J. Xiang, H. Meng, and A. Aboulnaga, "Scalable matrix inversion using MapReduce," in *Proc. 23rd Int. Symp. High-Perform. Parallel Distrib. Comput.*, 2014, pp. 177–190.

ALI M. MAHMOOD received the B.Sc. degree in computer engineering from the University of Technology-Baghdad in 2005, and the M.Sc. degree in computer engineering from Al-Nahrain University in 2010. He is currently pursuing the Ph.D. degree in wireless telecommunications with the School of Computing, Science and Engineering, University of Salford, U.K. His research interests include the scalability of mobile networks, queuing theory, and minimizing the computational complexity and end-to-end latency in the next generation 5G networks.

ADIL AL-YASIRI received the Ph.D. degree in software engineering from Liverpool JM University, U.K., in 1997. He was a Lecturer of computer systems with Liverpool JM University. He was an Associate Professor in computer science at UAE University until 2000, when he moved to Industry to work as an Independent Software Engineer and Consultant. During this time, he was involved in a number of projects around the world, advising clients (nationally and internationally) on various aspects of the software development process. In late 2003, he joined the School of Computing, Science and Engineering, University of Salford, U.K., as a Senior Lecturer in computer network systems, and led the development of a number of graduate and undergraduate programs, where he is currently the Post-Graduate Research Director. His research interests include Internet of Things, cloud computing, software engineering, and wireless sensor networks.

OMAR Y. K. ALANI received the Ph.D. degree in telecommunication engineering from De Montfort University, U.K., in 2005. He was a Lecturer of telecommunications systems with De Montfort University until 2006, after which, he was a Researcher at the Institute of Advanced Telecommunication, Swansea University. From 2007 to 2009, he was a Research Fellow with the School of Electrical and Electronic Engineering, University of Leeds. He is currently the Program Leader of computer networks at the School of Computing, Science and Engineering, University of Salford, U.K. He has published over 60 papers in high-quality journals and conferences proceedings in the field of telecommunications and networking. His research interests include 5G systems, wireless multimedia communications, radio resource management and location/mobility management in next generation mobile communication systems, diversity, and adaptive modulation techniques, and ad hoc and sensors networks.