Decentralized Baseband Processing for Massive MU-MIMO Systems

Journal Article

Author(s):
Li, Kaipeng; Sharan, Rishi R.; Chen, Yujun; Goldstein, Tom; Cavallaro, Joseph R.; Studer, Christoph

Decentralized Baseband Processing
for Massive MU-MIMO Systems

Kaipeng Li, Rishi Sharan, Yujun Chen, Tom Goldstein, Joseph R. Cavallaro, and Christoph Studer

Abstract—Achieving high spectral efficiency in realistic massive multi-user (MU) multiple-input multiple-output (MIMO) wireless systems requires computationally-complex algorithms for data detection in the uplink (users transmit to base-station) and beam-forming in the downlink (base-station transmits to users). Most existing algorithms are designed to be executed on centralized computing hardware at the base-station (BS), which results in prohibitive complexity for systems with hundreds or thousands of antennas and generates raw baseband data rates that exceed the limits of current interconnect technology and chip I/O interfaces. This paper proposes a novel decentralized baseband processing architecture that alleviates these bottlenecks by partitioning the BS antenna array into clusters, each associated with independent radio-frequency chains, analog and digital modulation circuitry, and computing hardware. For this architecture, we develop novel decentralized data detection and beamforming algorithms that only access local channel-state information and require low communication bandwidth among the clusters. We study the associated trade-offs between error-rate performance, computational complexity, and interconnect bandwidth, and we demonstrate the scalability of our solutions for massive MU-MIMO systems with thousands of BS antennas using reference implementations on a graphic processing unit (GPU) cluster.

Index Terms—Alternating direction method of multipliers (ADMM), conjugate gradient, beamforming, data detection, equalization, general-purpose computing on graphics processing unit (GPGPU), massive MU-MIMO.

I. INTRODUCTION

MASSIVE multi-user (MU) multiple-input multiple-output (MIMO) is among the most promising technologies for realizing high spectral efficiency and improved link reliability in fifth-generation (5G) wireless systems [3], [4]. The main idea behind massive MU-MIMO is to equip the base-station (BS) with hundreds or thousands of antenna elements, which increases the spatial resolution and provides an energy-efficient

K. Li, Y. Chen, and J. R. Cavallaro are with the Department of Electrical and Computer Engineering, Rice University, Houston 77251, TX (e-mail: kl33@rice.edu; yujun.chen@rice.edu; cavallar@rice.edu).

R. Sharan was with the School of Electrical and Computer Engineering, Cornell University, Ithaca 14853, NY, and is now at The MITRE Corporation, McLean 22102, VA (e-mail: rrs72@cornell.edu).

T. Goldstein is with the Department of Computer Science, University of Maryland, College Park 20740, MD (e-mail: tomg@cs.umd.edu).

C. Studer is with the School of Electrical and Computer Engineering, Cornell University, Ithaca 14853, NY (e-mail: studer@cornell.edu; web: http://vip.ece.cornell.edu; corresponding author).

way to serve a large number of users in the same time-frequency resource. Despite all the advantages of this emerging technology, the presence of a large number of BS antenna elements results in a variety of implementation challenges. One of the most critical challenges is the excessively high amount of raw baseband data that must be transferred from the baseband processing unit to the radio-frequency (RF) antenna units at the BS (or in the opposite direction). Consider, for example, a 128 BS-antenna massive MU-MIMO system with 40 MHz bandwidth and 10-bit analog-to-digital converters (ADCs). For such a system, the raw baseband data rates from and to the RF units easily exceed 200 Gbit/s. Such high data rates not only pose severe implementation challenges for the computing hardware to carry out the necessary baseband processing tasks, but the resulting raw baseband data stream may also exceed the bandwidth of existing high-speed interconnects, such as the common public radio interface (CPRI) [5].

A. Challenges of Centralized Baseband Processing

Recent testbed implementations for massive MU-MIMO, such as the Argos testbed [6], [7], the LuMaMi testbed [8], and the BigStation [9], reveal that the centralized baseband processing required for data detection in the uplink (users communicate to BS) and downlink (BS communicates to users using beamforming) is extremely challenging with current interconnect technology. In fact, all of the proposed data detection or beamforming algorithms that realize the full benefits of massive MU-MIMO systems with realistic (finite) antenna configurations, such as zero-forcing (ZF) or minimum mean-square error (MMSE) equalization or beamforming [10], rely on *centralized baseband processing*. This approach requires that full channel state information (CSI) and all receive/transmit data streams and all subcarriers (for wideband systems) are available at a centralized node, which processes and generates the baseband signals that are received from and transmitted to the radio-frequency (RF) chains. To avoid such a traditional, centralized baseband processing approach, existing testbeds, such as the Argos testbed [6], rely on maximum-ratio combining (MRC), which enables fully decentralized channel estimation, data detection, and beamforming directly at the antenna elements. Unfortunately, MRC significantly reduces the spectral efficiency for realistic antenna configurations compared to that of ZF or MMSE-based methods [10], which prevents the use of high-rate modulation and coding schemes that fully exploit the advantages of massive MU-MIMO.
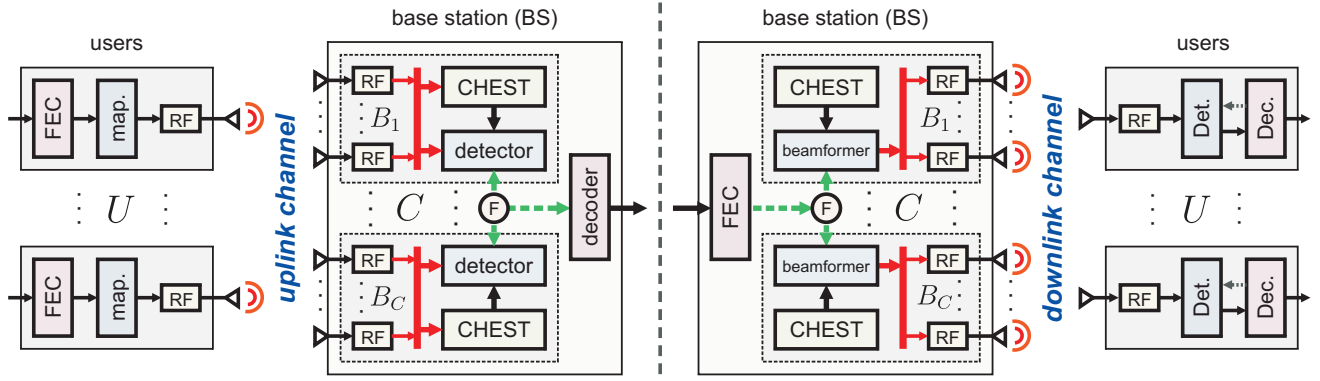
Fig. 1. Overview of the proposed decentralized baseband processing (DBP) architecture. Left: Massive MU-MIMO uplink: U single-antenna users communicate to the base-station (BS). The B BS antenna elements are divided into C clusters, which independently perform channel estimation (CHEST) and decentralized data detection. Right: Massive MU-MIMO downlink: The BS performs decentralized beamforming; each of the C clusters only uses local channel state information. In both scenarios, only a minimum amount of consensus information is exchanged among the clusters (indicated by the dashed green lines).

B. Decentralized Baseband Processing (DBP)

In this paper, we propose a *decentralized baseband processing (DBP)* architecture as illustrated in Figure 1, which alleviates the bottlenecks of massive MU-MIMO caused by extremely high raw baseband data rates and implementation complexity of centralized processing.[1] We partition the B BS antennas into C independent clusters, each having B_c antennas for the cth cluster so that $B = \sum_{c=1}^{C} B_c$. For simplicity, we will assume clusters of equal size and set $S = B_c$ which implies $SC = B$. Each cluster is associated with local computing hardware, a so-called *processing element (PE)*, that carries out the necessary baseband processing tasks in a decentralized and parallel fashion. A central fusion node ("F" in Figure 1) processes a small amount of consensus information that is exchanged among the clusters and required by our decentralized baseband algorithms (the dashed green lines in Figure 1).

Throughput the paper, we focus on time-division duplexing (TDD), i.e., we alternate between uplink and downlink communication within the same frequency band. In the uplink phase, U users communicate with the BS. First, CSI is acquired via pilots at the BS and stored locally at each cluster. Then, data is transmitted by the users and decoded at the BS. In the downlink phase, the BS transmits data to the U users. By exploiting channel reciprocity, the BS performs decentralized beamforming (or precoding) to mitigate MU interference (MUI) and to focus transmit power towards the users. As for the uplink, the C decentralized beamforming units only access local CSI.

The key features of the proposed DBP architecture can be summarized as follows: (i) DBP reduces the raw baseband data rates between each cluster and the associated RF chains. In addition, the I/O bandwidth of each PE can be reduced significantly as only raw baseband data from a (potentially small) subset of antennas must be transferred on and off chip. (ii) DBP lowers the computational complexity per PE by distributing and parallelizing the key signal-processing tasks. In addition to decentralizing channel estimation (CHEST), data detection, and beamforming (or precoding), DBP enables frequency-domain processing (e.g., fast Fourier transforms

for orthogonal frequency-division multiplexing) as well as impairment compensation (e.g., for carrier frequency and sampling rate offsets, phase noise, or I/Q imbalance) locally at each cluster. (iii) DPB enables modular and scalable BS designs; adding or removing antenna elements simply amounts to adding or removing computing clusters and the associated RF elements, respectively. (iv) DPB allows one to distribute the antenna array and the associated computing hardware over multiple buildings—an idea that was put forward recently in the massive MU-MIMO context [11].

C. Relevant Prior Art

The literature describes mainly three methods that are related to DBP: coordinated multipoint (CoMP), cloud radio access networks (C-RAN), and testbeds that perform distributed baseband processing across frequencies (or subcarriers). The following paragraphs discuss these results.

1) Coordinated multipoint (CoMP): Coordinated multipoint (CoMP) is a distributed communication technology to eliminate inter-cell interference, improve the data rate, and increase the spectrum efficiency for cell-edge users [12]. CoMP distributes multiple BSs across cells, which cooperate via backhaul interconnect to perform distributed uplink reception and downlink transmission. CoMP has been studied for cooperative transmission and reception [13], [14] in 3GPP LTE-A, and is widely believed to play an important role in 5G networks [15] along with other technologies, such as massive MU-MIMO [16]. Several algorithms for distributed beamforming with CoMP have been proposed in [17]–[19]. The paper [17] proposes a distributed precoding algorithm for multi-cell MIMO downlink systems using a dirty-paper coding. The papers [18], [19] propose distributed beamforming algorithms based on Gauss-Seidel and alternating direction method of multipliers (ADMM). These methods assume that the BSs in different cells have access to local CSI and coordinate with each other with limited backhaul information exchange. While these results are, in spirit, similar to the proposed DBP approach, our architecture (i) considers a decentralized architecture in which the computing hardware is *collocated* to support low-latency consensus information exchange, (ii) takes explicit advantage of massive

---

[1]For the sake of simplicity, the BS illustration in Figure 1 only includes the components that are relevant to the results in this paper.

MU-MIMO (the other results in [17]–[19] are for traditional, small-scale MIMO systems), and (iii) proposes a practical way to partition baseband processing that is complementary to CoMP. In fact, one could integrate DBP together with CoMP to deal with both intra-cell multi-user interference and inter-cell transmission interference more effectively, and to realize decentralized PHY layer processing using our DBP and higher layer (MAC layer, network layer, etc.) resource allocation and coordination with CoMP schemes. In addition, we propose more sophisticated algorithms that enable superior error-rate performance compared to the methods in [18], [19].

2) Cloud radio access networks (C-RAN): The idea behind C-RAN is to separate the BS into two modules, a remote radio head (RRH) and a baseband unit (BBU), which are connected via high-bandwidth interconnect. The RRHs are placed near the mobile users within the cell, while the BBUs are grouped into a BBU pool for centralized processing and located remotely from RRH [20]–[22]. C-RAN and CoMP both coordinate data transmission among multiple cells but with different physical realizations. CoMP integrates each pair of radio heads with associated BBU together and allows low-latency data transfer between each radio head and its corresponding BBU. Different BBUs are separately placed across multiple cells, entailing long latency on coordination among BBUs. C-RAN, in contrast, shifts the BBU coordination latency in CoMP to the data transfer latency between RRHs and BBUs, since BBUs are now grouped in a pool and can coordinate efficiently. Therefore, whether CoMP or C-RAN is more appropriate depends on whether BBU coordination or RRH-BBU data transfer is more efficient in a real-world deployment. Analogously to CoMP, we could integrate DBP together with C-RAN to exploit the benefits of both technologies. For example, each RRH now can be a large-scale antenna array (requiring higher RRH-BBU interconnection bandwidth). The associated BBU itself may rely on DBP and perform our algorithms to resolve intra-cell multi-user interference, while coordinating with other BBUs for inter-cell interference mitigation.

3) Distributed processing across frequencies: Existing testbeds, such as the LuMaMi testbed [8], [23] and the BigStation [9], distribute the necessary baseband processing tasks across frequencies. The idea is to divide the total frequency band into clusters of subcarriers in orthogonal frequency-division multiplexing (OFDM) systems where each frequency cluster is processed concurrently, enabling high degrees of parallelism [8], [9], [23]. Unfortunately, each frequency cluster still needs access to all BS antennas, which may result in high interconnect bandwidth. Furthermore, the frequency band must somehow be divided either using analog or digital circuitry, and frequency decentralization prevents a straightforward use of other waveform candidates, such as single-carrier frequency-division multiple access (SC-FDMA), filter bank multi-carrier (FBMC), and generalized frequency division multiplexing (GFDM) [24]. In contrast, our DBP architecture performs decentralization across antennas, which is compatible to most waveforms and requires data transmission only between a subset of antennas and the clusters. We emphasize, however, that DBP can be used *together* with frequency decentralization—in fact, our reference GPU implementation results shown in Section VI

exploit both spatial decentralization and frequency parallelism.

D. Contributions

We propose DBP to reduce the raw baseband and chip I/O bandwidths, as well as the signal-processing bottlenecks of massive MU-MIMO systems that perform centralized baseband processing. Our main contributions are as follows:

- We propose DBP, an novel architecture for scalable, TDD-based massive MU-MIMO BS designs, which distributes computation across clusters of antennas.
- We develop two decentralized algorithms for near-optimal data detection in the massive MU-MIMO uplink; both algorithms trade off error-rate performance vs. complexity.
- We develop a decentralized beamforming algorithm for the massive MU-MIMO downlink.
- We perform a simulation-based tradeoff analysis between error-rate performance, consensus data rate, and computational complexity for the proposed decentralized data detection and beamforming algorithms.
- We present implementation results for data detection and beamforming on a GPU cluster that showcase the efficacy and scalability of the proposed DBP approach.

Our results demonstrate that DBP enables modular and scalable BS designs for massive MU-MIMO with thousands of antenna elements while avoiding excessively high baseband and I/O data rates and significantly reducing the high computational complexity of conventional centralized algorithms.

E. Notation

Lowercase and uppercase boldface letters designate column vectors and matrices, respectively. For a matrix $\mathbf{A}$, we indicate its transpose and conjugate transpose by $\mathbf{A}^T$ and $\mathbf{A}^H$ respectively. The $M \times M$ identity matrix is denoted by $\mathbf{I}_M$ and the $M \times N$ all-zeros matrix by $\mathbf{0}_{M \times N}$. Sets are denoted by uppercase calligraphic letters; the cardinality of the $\mathcal{A}$ is denoted by $|\mathcal{A}|$. The real and imaginary parts of a complex scalar a are $\Re\{a\}$ and $\Im\{a\}$, respectively. The Kronecker product is $\otimes$ and $\mathbb{E}[\cdot]$ denotes expectation.

F. Paper Outline

The rest of the paper is organized as follows. Section II details the DBP architecture and introduces the associated uplink and downlink system models. Section III proposes two decentralized data detection algorithms. Section IV proposes a decentralized beamforming algorithm. Section V provides performance and complexity results. Section VI discusses our GPU cluster implementations. We conclude in Section VII. All proofs are relegated to Appendix A.

II. DBP: DECENTRALIZED BASEBAND PROCESSING

We now detail the DBP architecture illustrated in Figure 1 and the system models for the uplink and downlink. We consider a TDD massive MU-MIMO system and we assume a sufficiently long coherence time, i.e., the channel remains constant during both the uplink and downlink phases. In what follows, we focus on narrowband communication; a generalization to wideband systems is straightforward.

A. Uplink System Model and Architecture

1) Uplink system model: In the uplink phase, U single-antenna[2] user terminals communicate with a BS having $B \geq U$ antenna elements. Each user encodes its own information bit stream using a forward error correction (FEC) code and maps the resulting coded bit stream to constellation points in the set $\mathcal{O}$ (e.g., 16-QAM) using a predefined mapping rule (e.g., Gray mappings). At each user, the resulting constellation symbols are then modulated and transmitted over the wireless channel (subsumed in the "RF" block in Figure 1). The transmit symbols s_u, $u = 1, \ldots, U$, of all U users are subsumed in the uplink transmit vector $\mathbf{s}^u \in \mathcal{O}^U$. The baseband-equivalent input-output relation of the (narrowband) wireless uplink channel is modeled as $\mathbf{y}^u = \mathbf{H}^u \mathbf{s}^u + \mathbf{n}^u$, where $\mathbf{y}^u \in \mathbb{C}^B$ is the received uplink vector, $\mathbf{H}^u \in \mathbb{C}^{B \times U}$ is the (tall and skinny) uplink channel matrix, and $\mathbf{n}^u \in \mathbb{C}^B$ is i.i.d. circularly-symmetric complex Gaussian noise with variance N_0 per complex entry. The goal of the BS is to estimate the transmitted code bits given (approximate) knowledge of $\mathbf{H}^u$ and the received uplink vector $\mathbf{y}^u$. This information is then passed to the decoder, which computes estimates for the data bits of each user.

2) Decentralized architecture: Consider the left-hand side (LHS) of Figure 1. The proposed DBP architecture partitions[3] the receive vector $\mathbf{y}$ into C clusters so that $(\mathbf{y}^u)^T = [(\mathbf{y}_1^u)^T, \ldots, (\mathbf{y}_C^u)^T]$ with $\mathbf{y}_c^u \in \mathbb{C}^{B_c}$ and $B = \sum_{c=1}^C B_c$. As mentioned in Section I-B, we assume clusters of equal size and set $S = B_c$. By partitioning the uplink channel matrix $(\mathbf{H}^u)^T = [(\mathbf{H}_1^u)^T, \ldots, (\mathbf{H}_c^u)^T]$ *row-wise* into blocks of dimension $\mathbf{H}_c^u \in \mathbb{C}^{B_c \times U}$, $c = 1, \ldots, C$, and, analogously, the noise vector as $(\mathbf{n}^u)^T = [(\mathbf{n}_1^u)^T, \ldots, (\mathbf{n}_C^u)^T]$, we can rewrite the uplink input-output relation at each cluster as follows:

$$\mathbf{y}_c^u = \mathbf{H}_c^u \mathbf{s}^u + \mathbf{n}_c^u, \quad c = 1, \ldots, C. \tag{1}$$

The goal of DBP in the uplink is to compute an estimate for $\mathbf{s}^u$ in a decentralized manner: each cluster c only has access to $\mathbf{y}_c^u$, $\mathbf{H}_c^u$, and consensus information (see Section III).

As shown in LHS of Figure 1, each antenna element is associated to local RF processing circuitry; this includes analog and digital filtering, amplification, mixing, modulation, etc. As a consequence, all required digital processing tasks (e.g., used for OFDM processing) are also carried out in a decentralized manner. Even though we consider perfect synchronization and impairment-free transmission (such as carrier frequency and sampling rate offsets, phase noise, or I/Q imbalance), we note that each cluster and the associated RF processing circuitry would be able to separately compensate for such hardware non-idealities with well-established methods [25]. This key property significantly alleviates the challenges of perfectly synchronizing the clocks and oscillators among the clusters.

3) Channel estimation: During the training phase, each cluster c must acquire local CSI, i.e., compute an estimate of $\mathbf{H}_c^u$. To this end, U orthogonal pilots are transmitted from the users prior to the data transmission phase. Since each cluster c has access to $\mathbf{y}_c^u$, it follows from (1) that the associated local

channel matrix $\mathbf{H}_c^u$ can be estimated per cluster. The estimate for the channel matrix (as well as $\mathbf{y}_c^u$) is then stored locally at each cluster and *not* made accessible to the other clusters; this prevents a bandwidth-intensive broadcast of CSI (and receive vector data) to all clusters during the training phase.

4) Data detection: During the data transmission phase, decentralized data detection uses the receive vector $\mathbf{y}_c^u$, the associated CSI $\mathbf{H}_c^u$, and consensus information to generate an estimate of the transmitted data vector $\mathbf{s}^u$. This estimate is then passed to the decoder which computes estimates for the information bits of each user in a centralized manner; suitable data detection algorithms are proposed in Section III.

B. Downlink System Model and Architecture

1) Downlink system model: In the downlink phase, the B BS antennas communicate with the $U \leq B$ single-antenna user terminals. The information bits for each user are encoded separately using a FEC. The BS then maps the resulting (independent) coded bit streams to constellation points in the alphabet $\mathcal{O}$ to form the vector $\mathbf{s}^d \in \mathcal{O}^U$. To mitigate MUI, the BS performs beamforming (BF) or precoding, i.e., computes a BF vector $\mathbf{x}^d \in \mathbb{C}^B$ that is transmitted over the downlink channel. Beamforming requires knowledge of the (short and wide) downlink channel matrix $\mathbf{H}^d \in \mathbb{C}^{U \times B}$ and the transmit vector $\mathbf{s}^d \in \mathcal{O}^U$ to compute a BF vector that satisfies $\mathbf{s}^d = \mathbf{H}^d \mathbf{x}^d$ (see Section IV for the details). By assuming channel reciprocity, we have the property $\mathbf{H}^d = (\mathbf{H}^u)^T$ [3], [4], which implies that the channel matrix estimated in the uplink can be used in the downlink. The baseband-equivalent input-output relation of the (narrowband) wireless downlink channel is modeled as $\mathbf{y}^d = \mathbf{H}^d \mathbf{x}^d + \mathbf{n}^d$, where $\mathbf{y}^d \in \mathbb{C}^U$ is the receive vector at all users and $\mathbf{n}^d \in \mathbb{C}^U$ is i.i.d. circularly-symmetric complex Gaussian noise with variance N_0 per complex entry. By transmitting $\mathbf{x}^d$ over the wireless channel, the equivalent input-output relation is given by $\mathbf{y}^d = \mathbf{s}^d + \mathbf{n}^d$ and contains no MUI. Each of the users then estimates the transmitted code bits from y_u^d, $u = 1, \ldots, U$. This information is passed to the decoder, which computes estimates for the user's data bits.

2) Decentralized architecture: Consider the right-hand side (RHS) of Figure 1. Since the partitioning of the BS antennas was fixed for the uplink (cf. Section II-A), the BF vector $\mathbf{x}^d$ must be partitioned into C clusters so that $(\mathbf{x}^d)^T = [(\mathbf{x}_1^d)^T, \ldots, (\mathbf{x}_C^d)^T]$ with $\mathbf{x}_c^d \in \mathbb{C}^{B_c}$. By using reciprocity and the given antenna partitioning, each cluster c has access to only $\mathbf{H}_c^d = (\mathbf{H}_c^u)^T$. With this partitioning, we can rewrite the downlink input-output relation as follows:

$$\mathbf{y}^d = \sum_{c=1}^C \mathbf{H}_c^d \mathbf{x}_c^d + \mathbf{n}^d. \tag{2}$$

The goal of DBP in the downlink is to compute all local BF vectors $\mathbf{x}_c^d$, $c = 1, \ldots, C$, in a decentralized manner: each cluster c has access to only $\mathbf{s}^d$, $\mathbf{H}_c^d$, and consensus information (see Section IV for more details).

As shown in the RHS of Figure 1, each antenna element is associated to local RF processing circuitry. Analogously to the uplink, the required analog and digital signal processing tasks (e.g., used for OFDM modulation or impairment compensation) can be carried out in a decentralized manner, which alleviates the challenges of perfectly synchronizing the clusters.

---

[2]A generalization to multi-antenna user terminals is straightforward but omitted for the sake of simplicity of exposition.

[3]Other partitioning schemes may be possible but are not considered here.

3) Beamforming: In the downlink phase, decentralized BF uses the transmit vector $\mathbf{s}^d$, decentralized CSI $\mathbf{H}_c^d$, and consensus information in order to generate BF vectors $\mathbf{x}_c^d$ that satisfy $\mathbf{s}^d = \sum_{c=1}^{C} \mathbf{H}_c^d \mathbf{x}_c^d$. This ensures that transmission of the vectors $\mathbf{x}_c^d$ removes MUI; a suitable algorithm is detailed in Section IV.

III. DECENTRALIZED UPLINK: DATA DETECTION

We now propose two decentralized data detection algorithms for the massive MU-MIMO uplink. We start by discussing the general equalization problem and then, detail our novel ADMM and CG-based data detection algorithms. To simplify notation, we omit the uplink superscript ^{u} in this section.

A. Equalization-Based Data Detection

In order to arrive at computationally efficient algorithms for decentralized data detection, we focus on equalization-based methods. Such methods contain an equalization stage and a detection stage. For the equalization stage, we are interested in solving the following equalization problem

$$\text{(E0)} \quad \hat{\mathbf{x}} = \arg\min_{\mathbf{s} \in \mathbb{C}^U} g(\mathbf{s}) + \tfrac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{s}\|_2^2$$

in a decentralized manner. Here, the function $g(\mathbf{s})$ is a convex (but not necessarily smooth or bounded) regularizer, which will be discussed in detail below. For the detection stage, the result $\hat{\mathbf{x}}$ of the equalization problem (E0) can either be sliced entry-wise to the nearest constellation point in $\mathcal{O}$ to perform hard-output data detection or used to compute approximate soft-output values e.g., log-likelihood ratio (LLR) values [26].

For zero-forcing (ZF) and minimum mean-squared error (MMSE) data detection, we set the regularizer to $g(\mathbf{s}) = 0$ and $g(\mathbf{s}) = N_0/(2E_s)\|\mathbf{s}\|_2^2$, respectively, where $E_s = \mathbb{E}[|s_u|^2]$ for $u = 1, \ldots, U$, is the expected per-user transmit energy.[4] The generality of the equalization problem (E0) also encompasses more powerful data detection algorithms. In particular, we can set $g(\mathbf{s}) = \chi(\mathbf{s} \in \mathcal{C})$, where $\chi(\mathbf{s} \in \mathcal{C})$ is the characteristic function that is zero if $\mathbf{s}$ is in some convex set $\mathcal{C}$ and infinity otherwise. Specifically, to design data-detection algorithms that outperform ZF or MMSE data detection, we can use the convex polytope around the constellation set $\mathcal{O}$, which is given by

$$\mathcal{C} = \left\{ \sum_{i=1}^{|\mathcal{O}|} \alpha_i s_i \mid (\alpha_i \geq 0, \forall i) \wedge \sum_{i=1}^{|\mathcal{O}|} \alpha_i = 1 \right\}.$$

For QPSK with $\mathcal{O} = \{\pm 1 \pm i\}$, the convex set $\mathcal{C}$ is simply a box with radius 1 (i.e., side length of 2) centered at the origin. In this case, (E0) corresponds to the so-called box-constrained equalizer [27] which was shown recently to (often significantly) outperform linear ZF or MMSE data detection [28], [29]. In addition, box-constrained equalization does not require knowledge of the noise variance N_0, which is in stark contrast to the traditional MMSE equalizer. The decentralized equalization algorithm proposed next enables the use of such powerful regularizers.

---

[4]For the sake of simplicity, we assume an equal transmit power at each user. An extension to the general case is straightforward.

B. Decentralized Equalization via ADMM

To solve the equalization problem (E0) in a decentralized fashion, we make use of the ADMM framework [30]. We first introduce C auxiliary variables $\mathbf{z}_c = \mathbf{s}$, $c = 1, \ldots, C$, which allow us to rewrite (E0) in the equivalent form

$$\text{(E0}')\quad \hat{\mathbf{x}} = \arg\min_{\mathbf{s} \in \mathbb{C}^U,\, \mathbf{z}_c = \mathbf{s},\, c=1,\ldots,C} g(\mathbf{s}) + \sum_{c=1}^{C} \tfrac{1}{2}\|\mathbf{y}_c - \mathbf{H}_c \mathbf{z}_c\|_2^2.$$

Note that the added constraints in (E0$'$) enforce that each local vector $\mathbf{z}_c$ agrees with the global value of $\mathbf{s}$. As detailed in [30], these constraints can be enforced by introducing Lagrange multipliers $\{\boldsymbol{\lambda}_c\}_{c=1}^C$ for each cluster, and then computing a saddle point (where the augmented Lagrangian is minimal for $\mathbf{s}$ and $\mathbf{z}$, and maximal for $\boldsymbol{\lambda}$) of the so-called scaled augmented Lagrangian function, which is defined as

$$\mathcal{L}(\mathbf{s}, \mathbf{z}, \boldsymbol{\lambda}) = g(\mathbf{s})$$
$$+ \sum_{c=1}^{C} \left\{ \tfrac{1}{2}\|\mathbf{y}_c - \mathbf{H}_c \mathbf{z}_c\|_2^2 + \tfrac{\rho}{2}\|\mathbf{s} - \mathbf{z}_c - \boldsymbol{\lambda}_c\|_2^2 \right\}$$

for some fixed penalty parameter $\rho > 0$. Here, we stack all C auxiliary variables into the vector $\mathbf{z}^T = [\mathbf{z}_1^T \cdots \mathbf{z}_C^T]$ and stack all C scaled Lagrange multiplier vectors into the vector $\boldsymbol{\lambda}^T = [\boldsymbol{\lambda}_1^T \cdots \boldsymbol{\lambda}_C^T]$, where $\mathbf{z}_c, \boldsymbol{\lambda}_c \in \mathbb{C}^U$.

The saddle-point formulation of (E0$'$) is an example of a global variable consensus problem [30, Sec. 7.1] and can be solved using ADMM. We initialize $\mathbf{s}^{(1)} = \mathbf{0}_{U \times 1}$ and $\boldsymbol{\lambda}_c^{(1)} = \mathbf{0}_{U \times 1}$ for $c = 1, \ldots, C$, and carry out the following iterative steps:

$$\text{(E1)} \quad \mathbf{z}_c^{(t+1)} = \arg\min_{\mathbf{z}_c \in \mathbb{C}^U} \tfrac{1}{2}\|\mathbf{y}_c - \mathbf{H}_c \mathbf{z}_c\|_2^2 + \tfrac{\rho}{2}\big\|\mathbf{s}^{(t)} - \mathbf{z}_c - \boldsymbol{\lambda}_c^{(t)}\big\|_2^2$$

$$\text{(E2)} \quad \mathbf{s}^{(t+1)} = \arg\min_{\mathbf{s} \in \mathbb{C}^U} g(\mathbf{s}) + \sum_{c=1}^{C} \tfrac{\rho}{2}\big\|\mathbf{s} - \mathbf{z}_c^{(t+1)} - \boldsymbol{\lambda}_c^{(t)}\big\|_2^2$$

$$\text{(E3)} \quad \boldsymbol{\lambda}_c^{(t+1)} = \boldsymbol{\lambda}_c^{(t)} - \gamma\big(\mathbf{s}^{(t+1)} - \mathbf{z}_c^{(t+1)}\big)$$

for the iterations $t = 1, 2, \ldots$ until convergence or a maximum number $T_{\max}$ of iterations has been reached. The parameter $\rho > 0$ controls the step size and $\gamma = 1$ is a typical choice that guarantees convergence. See [31] for a more detailed discussion on the convergence of ADMM.

Steps (E1) and (E2) can be carried out in a decentralized and parallel manner, i.e., each cluster $c = 1, \ldots, C$ only requires access to local variables and local channel state information, as well as the consensus vectors $\mathbf{s}^{(t)}$ and $\mathbf{s}^{(t+1)}$. Step (E2) updates the consensus vector. While the vectors $\{\mathbf{z}_c^{(t+1)}\}$ and $\{\boldsymbol{\lambda}_c^{(t)}\}$ for every cluster appear in (E2), it is known that this can be computed using only the global average of these vectors, which is easily stored on a fusion node [30]. The architecture proposed in Section II can compute these averages and perform this update in an efficient manner. We next discuss the key details of the proposed decentralized data detection algorithm.

C. ADMM Algorithm Details and Decentralization

1) Step (E1): This step corresponds to a least-squares (LS) problem that can be solved in closed form and independently on each cluster. For a given cluster c, we can rewrite the

---

Algorithm 1 Decentralized ADMM-based Data Detection

1: **Input**: $\mathbf{y}_c$, $\mathbf{H}_c$, $c = 1, 2, \ldots, C$, ρ, γ, N_0, and E_s
2: *Preprocessing*:
3: **if** $S \leq U$ **then**
4: $\quad \mathbf{A}_c^{-1} = (\mathbf{H}_c \mathbf{H}_c^H + \rho \mathbf{I}_S)^{-1}$
5: $\quad \mathbf{y}_c^{\text{reg}} = \mathbf{H}_c^H \mathbf{A}_c^{-1} \mathbf{y}_c$
6: **else**
7: $\quad \mathbf{B}_c^{-1} = (\mathbf{H}_c^H \mathbf{H}_c + \rho \mathbf{I}_U)^{-1}$
8: $\quad \mathbf{y}_c^{\text{reg}} = \mathbf{B}_c^{-1} \mathbf{H}_c^H \mathbf{y}_c$
9: **end if**
10: *ADMM iterations*:
11: **Init**: $\boldsymbol{\lambda}_c^{(1)} = \mathbf{0}$, $\mathbf{z}_c^{(1)} = \mathbf{y}_c^{\text{reg}}$, $\mathbf{s}^{(1)} = \left(\frac{N_0}{\rho E_s} + C\right)^{-1}\left(\sum_{c=1}^{C} \mathbf{z}_c^{(1)}\right)$
12: **for** $t = 2, 3, \ldots, T_{\max}$ **do**
13: $\quad \boldsymbol{\lambda}_c^{(t)} = \boldsymbol{\lambda}_c^{(t-1)} + \gamma(\mathbf{z}_c^{(t-1)} - \mathbf{s}^{(t-1)})$
14: $\quad$ **if** $S \leq U$ **then**
15: $\quad\quad \mathbf{z}_c^{(t)} = \mathbf{y}_c^{\text{reg}} + (\mathbf{s}^{(t-1)} - \boldsymbol{\lambda}_c^{(t)}) - \mathbf{H}_c^H \mathbf{A}_c^{-1} \mathbf{H}_c(\mathbf{s}^{(t-1)} - \boldsymbol{\lambda}_c^{(t)})$
16: $\quad$ **else**
17: $\quad\quad \mathbf{z}_c^{(t)} = \mathbf{y}_c^{\text{reg}} + \rho \mathbf{B}_c^{-1}(\mathbf{s}^{(t-1)} - \boldsymbol{\lambda}_c^{(t)})$
18: $\quad$ **end if**
19: $\quad \mathbf{w}_c^{(t)} = \mathbf{z}_c^{(t)} + \boldsymbol{\lambda}_c^{(t)}$
20: $\quad \mathbf{w}^{(t)} = \sum_{c=1}^{C} \mathbf{w}_c^{(t)}$ $\qquad$ // Consensus
21: $\quad \mathbf{s}^{(t)} = \left(\frac{N_0}{\rho E_s} + C\right)^{-1} \mathbf{w}^{(t)}$
22: **end for**
23: **Output**: $\hat{\mathbf{x}} = \mathbf{s}^{(T_{\max})}$

---

minimization in Step (E1) in more compact form as

$$\mathbf{z}_c^{(t+1)} = \arg\min_{\mathbf{z}_c \in \mathbb{C}^U} \left\| \begin{bmatrix} \mathbf{y}_c \\ \sqrt{\rho}(\mathbf{s}^{(t)} - \boldsymbol{\lambda}_c^{(t)}) \end{bmatrix} - \begin{bmatrix} \mathbf{H}_c \\ \sqrt{\rho}\,\mathbf{I}_U \end{bmatrix} \mathbf{z}_c \right\|_2^2,$$

which has the following closed-form solution:

$$\mathbf{z}_c^{(t+1)} = \mathbf{y}_c^{\text{reg}} + \rho \mathbf{B}_c^{-1}(\mathbf{s}^{(t)} - \boldsymbol{\lambda}_c^{(t)}). \tag{3}$$

Here, $\mathbf{y}_c^{\text{reg}} = \mathbf{B}_c^{-1} \mathbf{H}_c^H \mathbf{y}_c$ is the regularized estimate with $\mathbf{B}_c^{-1} = (\mathbf{H}_c^H \mathbf{H}_c + \rho \mathbf{I}_U)^{-1}$. To reduce the amount of recurrent computations, we can precompute $\mathbf{B}_c^{-1}$ and reuse the result in each iteration. For situations where the cluster size S is smaller than the number of users U, we can use the Woodbury matrix identity [32] to derive the following equivalent update:

$$\mathbf{z}_c^{(t+1)} = \mathbf{y}_c^{\text{reg}} + (\mathbf{I}_U - \mathbf{H}_c^H \mathbf{A}_c^{-1} \mathbf{H}_c)(\mathbf{s}^{(t)} - \boldsymbol{\lambda}_c^{(t)}). \tag{4}$$

Here, $\mathbf{y}_c^{\text{reg}} = \mathbf{H}_c^H \mathbf{A}_c^{-1} \mathbf{y}_c$ is a regularized estimate of the transmit vector with $\mathbf{A}_c^{-1} = (\mathbf{H}_c \mathbf{H}_c^H + \rho \mathbf{I}_S)^{-1}$. This requires the inversion of an $S \times S$ matrix, which is more easily computed than the $U \times U$ inverse required by (3). We note that whether (3) or (4) leads to lower computational complexity depends on U, S, and the number of ADMM iterations (see Section V-A).

2) Step (E2): This step requires gathering of local computation results, averaging the sum in a centralized manner, and distributing the averaged consensus information. To reduce the amount of data that must be exchanged, each cluster only communicates the intermediate variable $\mathbf{w}_c^{(t)} = \mathbf{z}_c^{(t+1)} + \boldsymbol{\lambda}_c^{(t)}$, and only the average of these vectors is used on the fusion node. This simplification is accomplished using the following lemma; a proof is given in Appendix A-A.

Lemma 1. *The problem in Step (E2) simplifies to*

$$\mathbf{s}^{(t+1)} = \arg\min_{\mathbf{s} \in \mathbb{C}^U} g(\mathbf{s}) + \frac{C\rho}{2} \left\| \mathbf{s} - \mathbf{v}^{(t)} \right\|_2^2 \tag{5}$$

with $\mathbf{v}^{(t)} = \frac{1}{C} \mathbf{w}^{(t)} = \frac{1}{C} \sum_{c=1}^{C} \mathbf{w}_c^{(t)}$ *and* $\mathbf{w}_c^{(t)} = \mathbf{z}_c^{(t+1)} + \boldsymbol{\lambda}_c^{(t)}$.

Computation of (5) requires two parts. The first part

corresponds to a simple averaging procedure to obtain $\mathbf{v}^{(t)}$, which can be carried out via sum reduction in a tree-like fashion followed by centralized averaging. The second part is the minimization in (5) that is known as the proximal operator for the function $g(\mathbf{s})$ [33]. For ZF, MMSE, and box-constrained equalization with QAM alphabets, the proximal operator has the following simple closed-form expressions:

$$(\text{E2-ZF}) \quad \mathbf{s}^{(t+1)} = \mathbf{v}^{(t)}$$
$$(\text{E2-MMSE}) \quad \mathbf{s}^{(t+1)} = \frac{C\rho E_s}{N_0 + C\rho E_s} \mathbf{v}^{(t)}$$
$$(\text{E2-BOX}) \quad s_u^{(t+1)} = \text{sgn}(\Re\{v_u^{(t)}\})\min\{|\Re\{v_u^{(t)}\}|, r\}$$
$$\qquad\qquad + i\,\text{sgn}(\Im\{v_u^{(t)}\})\min\{|\Im\{v_u^{(t)}\}|, r\}$$

for $u = 1, \ldots, U$. Here, (E2-BOX) is the orthogonal projection of the vector $\mathbf{v}^{(t)}$ onto the hypercube with radius r that covers the QAM constellation. For BPSK, the proximal operator corresponds to the orthogonal projection onto the real line between $[-r, +r]$ and is given by $s_u^{(t+1)} = \text{sgn}(\Re\{v_u^{(t)}\})\min\{|\Re\{v_u^{(t)}\}|, r\}$, $u = 1, \ldots, U$.

After computation of (5), the consensus vector $\mathbf{s}^{(t+1)}$ needs to be distributed to all C clusters. In practice, we distribute $\mathbf{w}^{(t)}$ as soon as it is available, and the scaling steps to get $\mathbf{s}^{(t+1)}$ from $\mathbf{w}^{(t)}$ are computed locally on each cluster after it receives $\mathbf{w}^{(t)}$. With this approach, no cluster waits for the computation of $\mathbf{s}^{(t+1)}$ on a central/master worker (fusion node) before ADMM iterations proceed.

3) Step (E3): This step can be carried out independently in each cluster after $\mathbf{s}^{(t+1)}$ has been calculated.

We summarize the resulting decentralized ADMM procedure for MMSE equalization in Algorithm 1. The equalization output is the consensus vector $\hat{\mathbf{x}} = \mathbf{s}^{(T_{\max})}$. Note that Algorithm 1 slightly deviates from the procedure outlined in Section III-B. Specifically, Algorithm 1 performs the steps in the following order: Step (E3), Step (E1), and Step (E2); this is due to the fact that the global equalizer output $\hat{\mathbf{x}}$ results from Step (E2). We note that re-ordering the steps as in Algorithm 1 has no effect on the convergence and the equalization result.

We will analyze the algorithm's complexity[5] in Section V-A; a GPU cluster implementation will be discussed in Section VI.

D. Decentralized Equalization via Conjugate Gradients

If the regularization function $g(\mathbf{s})$ of (E0) is quadratic, as in the case for MMSE equalization where $g(\mathbf{s}) = N_0/(2E_s)\|\mathbf{s}\|_2^2$, then we can solve (E0) with an efficient decentralized conjugate gradient (CG) method [34]–[36]. Our method builds on the CG algorithm used in [34] for *centralized* equalization-based data detection in massive MU-MIMO systems. Our idea is to break all centralized computations that rely on global CSI and receive data (i.e., $\mathbf{H}$ and $\mathbf{y}$) into smaller, independent problems that only require local CSI and receive data ($\mathbf{H}_c$ and $\mathbf{y}_c$). The centralized CG-based detector in [34] involves two stages: a

---

[5]The operataion $\mathbf{H}_c^H \mathbf{A}_c^{-1} \mathbf{H}_c$ on line 15 of Algorithm 1 could be computed once in a preprocessing stage to avoid recurrent computations during the iterations. Instead, in Algorithm 1 we directly compute $\mathbf{H}_c^H \mathbf{A}_c^{-1} \mathbf{H}_c(\mathbf{s}^{(t-1)} - \boldsymbol{\lambda}_c^{(t)})$ in each iteration because this approach requires only three matrix-vector multiplications per iteration; precomputing $\mathbf{H}_c^H \mathbf{A}_c^{-1} \mathbf{H}_c$ requires two costly matrix-matrix multiplications. Hence, our complexity analysis in Section V-A refers to the procedure detailed in Algorithm 1.

Algorithm 2 Decentralized CG-based Data Detection

1: **Input:** $\mathbf{H}_c$, $c = 1, \ldots, C$, and $\mathbf{y}_c$, and ρ
2: *Preprocessing:*
3:     $\mathbf{y}_c^{\mathrm{MRC}} = \mathbf{H}_c^H \mathbf{y}_c$ // Decentralized
4:     $\mathbf{y}^{\mathrm{MRC}} = \sum_{c=1}^C \mathbf{y}_c^{\mathrm{MRC}}$ // Centralized
5: *CG iterations:*
6: **Init:** $\mathbf{r}^{(0)} = \mathbf{y}^{\mathrm{MRC}}$, $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$, $\mathbf{x}^{(0)} = \mathbf{0}$
7: **for** $t = 1, \ldots, T_{\max}$ **do**
8:     Decentralized (each cluster c performs the same operation):
9:        $\mathbf{w}_c^{(t)} = \mathbf{H}_c^H \mathbf{H}_c \mathbf{p}^{(t-1)}$
10:    Centralized (consensus on a centralized processing unit):
11:       $\mathbf{w}^{(t)} = \sum_{c=1}^c \mathbf{w}_c^{(t)}$ // Consensus
12:    Decentralized (each cluster c performs the same operations):
13:       $\mathbf{e}^{(t)} = \rho \mathbf{p}^{(t-1)} + \mathbf{w}^{(t)}$
14:       $\alpha = \|\mathbf{r}^{(t-1)}\|^2 / ((\mathbf{p}^{(t-1)})^H \mathbf{e}^{(t)})$
15:       $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)} + \alpha \mathbf{p}^{(t-1)}$
16:       $\mathbf{r}^{(t)} = \mathbf{r}^{(t-1)} - \alpha \mathbf{e}^{(t-1)}$
17:       $\beta = \|\mathbf{r}^{(t)}\|^2 / \|\mathbf{r}^{(t-1)}\|^2$
18:       $\mathbf{p}^{(t)} = \mathbf{r}^{(t)} + \beta \mathbf{p}^{(t-1)}$
19: **end for**
20: **Output:** $\hat{\mathbf{x}} = \mathbf{x}^{(T_{\max})}$

preprocessing stage for calculating the MRC output $\mathbf{y}^{\mathrm{MRC}}$ and a CG iteration stage to estimate $\hat{\mathbf{x}}$.

In the preprocessing stage, we rewrite the MRC vector $\mathbf{y}^{\mathrm{MRC}} = \mathbf{H}^H \mathbf{y}$ as $\mathbf{y}^{\mathrm{MRC}} = \sum_{c=1}^C \mathbf{H}_c^H \mathbf{y}_c$, which decentralizes the preprocessing stage. Specifically, each cluster computes $\mathbf{H}_c^H \mathbf{y}_c$; the results of each cluster are then summed up in a centralized manner to obtain the MRC output $\mathbf{y}^{\mathrm{MRC}}$.

For the CG iteration stage, we need to update the estimated transmit vector and a number of intermediate vectors required by the CG algorithm (see [34] for the details). While most operations are not directly dependent on global CSI $\mathbf{H}$ but on intermediate results, the update of the following vector

$$\mathbf{e}^{(t)} = \big(\rho \mathbf{I} + \mathbf{H}^H \mathbf{H}\big) \mathbf{p}^{(t-1)}, \qquad (6)$$

requires direct access to the global channel matrix $\mathbf{H}$ and thus, must be decentralized. Here, $\rho = N_0 / E_s$ for MMSE equalization and $\rho = 0$ for zero-forcing equalization. It is key to realize that the Gram matrix can be written as $\mathbf{H}^H \mathbf{H} = \sum_{c=1}^C \mathbf{H}_c^H \mathbf{H}_c$. Hence, we can reformulate (6) as

$$\mathbf{e}^{(t)} = \rho \mathbf{p}^{(t-1)} + \sum_{c=1}^C \mathbf{H}_c^H \mathbf{H}_c \mathbf{p}^{(t-1)}. \qquad (7)$$

Put simply, by locally computing $\mathbf{w}_c^{(t)} = \mathbf{H}_c^H \mathbf{H}_c \mathbf{p}^{(t-1)}$ at each antenna cluster, we can obtain the result in (7) by performing the following centralized computations that do not require global CSI: $\mathbf{w}^{(t)} = \sum_{c=1}^C \mathbf{w}_c^{(t)}$ and $\mathbf{e}^{(t)} = \rho \mathbf{p}^{(t-1)} + \mathbf{w}^{(t)}$.

The decentralized CG-based data detection algorithm is summarized in Algorithm 2. The computations of $\mathbf{e}^{(t)}$, $\mathbf{x}^{(t)}$, $\mathbf{r}^{(t)}$, and $\mathbf{p}^{(t)}$ do not require access to the (global) channel matrix $\mathbf{H}$ and can be carried out in a centralized processing unit. We must, however, broadcast the vector $\mathbf{p}^{(t)}$ to each antenna cluster before the decentralized update of $\mathbf{w}_c^{(t+1)}$ in the next iteration can take place. Alternatively, we can directly broadcast the consensus vector $\mathbf{w}^{(t)}$, so that each antenna cluster can simultaneously compute their *own copy* of $\mathbf{e}^{(t)}$, $\mathbf{x}^{(t)}$, $\mathbf{r}^{(t)}$, and $\mathbf{p}^{(t)}$ in a decentralized manner to ensure the local existence of $\mathbf{p}^{(t)}$ for updating $\mathbf{w}_c^{(t+1)}$. With this alternative approach, we can completely shift the complexity from the centralized processing unit to each cluster, leaving the calculation of $\mathbf{w}^{(t)}$

as the only centralized computation in a CG iteration. This approach also enables the concatenation of data gathering and broadcasting, which can be implemented using a single message-passing function (see Section VI for the details).[6]

IV. DECENTRALIZED DOWNLINK: BEAMFORMING

We now develop a decentralized beamforming algorithm for the massive MU-MIMO downlink. We start by discussing the general beamforming (or precoding) problem, and then detail our ADMM-based beamforming algorithm. To simplify notation, we omit the downlink superscript ^{d}.

A. Beamforming Problem

We solve the following beamforming problem

$$(\text{P0}) \quad \hat{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathbb{C}^B} \|\mathbf{x}\|_2 \quad \text{subject to } \|\mathbf{s} - \mathbf{H}\mathbf{x}\|_2 \leq \varepsilon,$$

which aims at minimizing the instantaneous transmit energy while satisfying the precoding constraint $\|\mathbf{s} - \mathbf{H}\mathbf{x}\|_2 \leq \varepsilon$. By defining the residual interference as $\mathbf{e} = \mathbf{s} - \mathbf{H}\hat{\mathbf{x}}$, we see that transmission of the solution vector $\hat{\mathbf{x}}$ of (P0) leads to the input-output relation $\mathbf{y} = \mathbf{s} + \mathbf{e} + \mathbf{n}$ with $\|\mathbf{e}\|_2 \leq \varepsilon$. Hence, each user only sees their dedicated signal contaminated with Gaussian noise $\mathbf{n}$ and residual interference $\mathbf{e}$, whose energy can be controlled by the parameter $\varepsilon \geq 0$. By setting $\varepsilon = 0$, this problem has a well-known closed-form solution and corresponds to the so-called zero-forcing (ZF) beamformer, which is given by $\hat{\mathbf{x}} = \mathbf{H}^H (\mathbf{H}\mathbf{H}^H)^{-1} \mathbf{s}$ assuming that $U \leq B$ and $\mathbf{H}$ is full rank. Our goal is to develop an algorithm that computes the solution of (P0) in a decentralized fashion.

B. Decentralized Beamforming via ADMM

By introducing C auxiliary variables $\mathbf{z}_c = \mathbf{H}_c \mathbf{x}_c$, $c = 1, \ldots, C$, we can rewrite (P0) in the following equivalent form:

$$(\text{P0}') \quad \hat{\mathbf{x}} = \arg\min_{\mathbf{x} \in \mathbb{C}^B} \|\mathbf{x}\|_2 \quad \text{subject to } \big\|\mathbf{s} - \sum_{c=1}^C \mathbf{z}_c\big\|_2 \leq \varepsilon$$
$$\text{and } \mathbf{z}_c = \mathbf{H}_c \mathbf{x}_c, \ c = 1, \ldots, C.$$

Here, $\mathbf{H}_c$ is the downlink channel matrix at cluster c. The solution to the beamforming problem (P0$'$) corresponds to a saddle point of the scaled augmented Lagrangian function:

$$\mathcal{L}(\mathbf{s}, \mathbf{z}, \boldsymbol{\lambda}) = \tfrac{1}{2}\|\mathbf{x}\|_2^2 + \sum_{c=1}^C \tfrac{\rho}{2}\|\mathbf{H}_c \mathbf{x}_c - \mathbf{z}_c - \boldsymbol{\lambda}_c\|_2^2 + \mathcal{X}(\mathbf{z}),$$

where $\mathcal{X}(\mathbf{z})$ is the characteristic function for the convex constraint of the beamforming problem (P0), i.e., $\mathcal{X}(\mathbf{z}) = 0$ if $\|\mathbf{s} - \sum_{c=1}^C \mathbf{z}_c\|_2 \leq \varepsilon$ and $\mathcal{X}(\mathbf{z}) = \infty$ otherwise. The problem (P0) corresponds to a sharing consensus problem with regularization [30, Sec. 7.3].

In order to arrive at a decentralized beamforming algorithm, we now use the ADMM framework to find a solution to (P0$'$). We initialize[7] $\mathbf{z}_c^{(1)} = \max\{U/B, 1/C\}\mathbf{s}$. We then perform the

---

[6]The Gram matrix $\mathbf{G}_c = \mathbf{H}_c^H \mathbf{H}_c$ can be precomputed to avoid recurrent computations (line 9 in Algorithm 2). However, practical systems only need a small number of CG iterations, and $\mathbf{H}_c^H \mathbf{H}_c \mathbf{p}^{(t-1)}$ at line 9 is computed using two matrix-vector multiplications, which avoids the expensive matrix-matrix multiplication needed to form $\mathbf{G}_c$.

[7]This initializer is a properly-scaled version of the MRC beamforming vector and exhibits good performance for the considered channel model.

following three-step procedure until convergence or a maximum number of iterations has been reached:

(P1) $\quad \mathbf{x}_c^{(t+1)} = \underset{\mathbf{x}_c \in \mathbb{C}^S}{\arg\min} \frac{1}{2}\|\mathbf{x}_c\|_2^2 + \frac{\rho}{2}\|\mathbf{H}_c\mathbf{x}_c - \mathbf{z}_c^{(t)} - \boldsymbol{\lambda}_c^{(t)}\|_2^2$

(P2) $\quad \mathbf{z}^{(t+1)} = \underset{\mathbf{z}_c \in \mathbb{C}^U}{\arg\min} \sum_{c=1}^{C} \frac{\rho}{2}\|\mathbf{H}_c\mathbf{x}_c^{(t+1)} - \mathbf{z}_c - \boldsymbol{\lambda}_c^{(t)}\|_2^2 + \mathcal{X}(\mathbf{z})$

(P3) $\quad \boldsymbol{\lambda}_c^{(t+1)} = \boldsymbol{\lambda}_c^{(t)} - \gamma(\mathbf{H}_c\mathbf{x}_c^{(t+1)} - \mathbf{z}_c^{(t+1)})$.

Here, $\mathbf{z}_c$ is the local beamforming output and $\mathbf{z}$ is the consensus solution of (P2). The parameter $\rho > 0$ affects the step size and $\gamma = 1$ ensures convergence of the algorithm. While both the Steps (P1) and (P3) can efficiently be computed in a decentralized manner, it is not obvious how Step (P2) can be decentralized. We next show the details to transform Step (P2) into a form that requires simple global averaging.

C. ADMM Algorithm Details and Decentralization

1) Step (P1): Analogous to Step (E1), this step corresponds to a LS problem that can be solved in closed form and independently in every cluster. For a given cluster $c = 1, \ldots, C$, we can rewrite the minimization in (P1) as

$$\mathbf{x}_c^{(t+1)} = \underset{\mathbf{x}_c \in \mathbb{C}^S}{\arg\min} \left\| \begin{bmatrix} \sqrt{\rho}(\mathbf{z}_c^{(t)} + \boldsymbol{\lambda}_c^{(t)}) \\ \mathbf{0}_{S \times 1} \end{bmatrix} - \begin{bmatrix} \sqrt{\rho}\mathbf{H}_c \\ \mathbf{I}_S \end{bmatrix} \mathbf{x}_c \right\|_2^2,$$

which has the following closed-form solution:

$$\mathbf{x}_c^{(t+1)} = \mathbf{A}_c^{-1}\mathbf{H}_c^H(\mathbf{z}_c^{(t)} + \boldsymbol{\lambda}_c^{(t)}).$$

Here, $\mathbf{A}_c^{-1} = (\mathbf{H}_c^H\mathbf{H}_c + \rho^{-1}\mathbf{I}_S)^{-1}$ requires the computation of an $S \times S$ matrix inverse. If the cluster size S is larger than the number of users U, then we can use the Woodbury matrix identity [32] to derive the following equivalent update:

$$\mathbf{x}_c^{(t+1)} = \mathbf{H}_c^H\mathbf{B}_c^{-1}(\mathbf{z}_c^{(t)} + \boldsymbol{\lambda}_c^{(t)}).$$

Here, $\mathbf{B}_c^{-1} = (\mathbf{H}_c\mathbf{H}_c^H + \rho^{-1}\mathbf{I}_U)^{-1}$ requires the computation of an $U \times U$ matrix inverse. We note that U, S, and the number of iterations can determine which of the two $\mathbf{x}_c^{(t+1)}$ variations leads to lower overall computational complexity.

2) Step (P2): The presence of the indicator function $\mathcal{X}(\mathbf{z})$ makes it non-obvious whether this step indeed can be carried out in a decentralized fashion. The next results show that a simple averaging procedure—analogously to that used in Step (E1) for decentralized data detection—can be carried out to perform Step (E2); the proof is given in Appendix A-B.

Lemma 2. *The minimization in Step (P2) simplifies to*

$$\mathbf{z}_c^{(t+1)} = \mathbf{w}_c^{(t)} + \max\left\{0, 1 - \frac{\varepsilon}{\|\mathbf{s} - \mathbf{v}^{(t)}\|_2}\right\}\left(\frac{1}{C}\mathbf{s} - \mathbf{v}^{(t)}\right) \quad (8)$$

with $\mathbf{v}^{(t)} = \frac{1}{C}\mathbf{w}^{(t)} = \frac{1}{C}\sum_{c=1}^{C}\mathbf{w}_c^{(t)}$; $\mathbf{w}_c^{(t)} = \mathbf{H}_c\mathbf{x}_c^{(t+1)} - \boldsymbol{\lambda}_c^{(t)}$.

For $\varepsilon = 0$, we get an even more compact expression

$$\mathbf{z}_c^{(t+1)} = \mathbf{w}_c^{(t)} + \frac{1}{C}\mathbf{s} - \mathbf{v}^{(t)}, \ c = 1, \ldots, C$$

Evidently (8) only requires a simple averaging procedure, which can be carried out by gathering local computation results from and broadcasting the averaged consensus back to each cluster.

Algorithm 3 Decentralized ADMM-based Beamforming

1: **Input:** $\mathbf{s}$, $\mathbf{H}_c$, $c = 1, 2, \ldots, C$, ρ, and γ
2: *Preprocessing:*
3: **if** $S \leq U$ **then**
4: $\quad \mathbf{A}_c^{-1} = (\mathbf{H}_c^H\mathbf{H}_c + \rho^{-1}\mathbf{I}_S)^{-1}$
5: **else**
6: $\quad \mathbf{B}_c^{-1} = (\mathbf{H}_c\mathbf{H}_c^H + \rho^{-1}\mathbf{I}_U)^{-1}$
7: **end if**
8: *ADMM iterations:*
9: Init: $\mathbf{z}_c^{(1)} = \max\{U/B, 1/C\}\mathbf{s}$, $\boldsymbol{\lambda}_c^{(1)} = \mathbf{0}$
10: $\quad \mathbf{x}_c^{(1)} = \mathbf{A}_c^{-1}\mathbf{H}_c^H\mathbf{z}_c^{(1)}(S \leq U)$ or $\mathbf{H}_c^H\mathbf{B}_c^{-1}\mathbf{z}_c^{(1)}(S > U)$
11: **for** $t = 2, 3, \ldots, T_{\max}$ **do**
12: $\quad \mathbf{m}_c^{(t-1)} = \mathbf{H}_c\mathbf{x}_c^{(t-1)}$
13: $\quad \mathbf{w}_c^{(t-1)} = \mathbf{m}_c^{(t-1)} - \boldsymbol{\lambda}_c^{(t-1)}$
14: $\quad \mathbf{w}^{(t-1)} = \sum_{c=1}^{C}\mathbf{w}_c^{(t-1)}$ $\quad$ // Consensus
15: $\quad \mathbf{z}_c^{(t)} = \mathbf{w}_c^{(t-1)} + C^{-1}(\mathbf{s} - \mathbf{w}^{(t-1)})$
16: $\quad \boldsymbol{\lambda}_c^{(t)} = \boldsymbol{\lambda}_c^{(t-1)} - \gamma(\mathbf{m}_c^{(t-1)} - \mathbf{z}_c^{(t)})$
17: $\quad$ **if** $S \leq U$ **then**
18: $\quad\quad \mathbf{x}_c^{(t)} = \mathbf{A}_c^{-1}\mathbf{H}_c^H(\mathbf{z}_c^{(t)} + \boldsymbol{\lambda}_c^{(t)})$
19: $\quad$ **else**
20: $\quad\quad \mathbf{x}_c^{(t)} = \mathbf{H}_c^H\mathbf{B}_c^{-1}(\mathbf{z}_c^{(t)} + \boldsymbol{\lambda}_c^{(t)})$
21: $\quad$ **end if**
22: **end for**
23: **Output:** $\hat{\mathbf{x}} = [\mathbf{x}_1^{(T_{\max})}; \mathbf{x}_2^{(T_{\max})}; \cdots; \mathbf{x}_C^{(T_{\max})}]$

---

3) Step (P3): This step can be performed independently in each cluster after distributing $\mathbf{w}^{(t)}$ and getting local $\mathbf{z}_c^{(t+1)}$.

The resulting ADMM-based decentralized beamforming procedure is summarized in Algorithm 3, where we assume $\varepsilon = 0$. To facilitate implementation of the decentralized beamforming algorithm, we initialize $\mathbf{z}_c^{(1)}, \boldsymbol{\lambda}_c^{(1)}, \mathbf{x}_c^{(1)}$ and then update the variables in the order of $\mathbf{z}_c^{(t)}, \boldsymbol{\lambda}_c^{(t)}, \mathbf{x}_c^{(t)}$ realizing that the final output of the local beamformer is simply $\mathbf{x}_c^{(T_{\max})}$. Note that Algorithm 3 slightly deviates from the step-by-step beamforming procedure in Section IV-B. Specifically, we carry out Steps (P2), (P3), and (P1), as the beamforming output results from Step (P1). This re-ordering has no effect on the algorithm's convergence or beamforming result. We will analyze the algorithm complexity[8] in Section V-A and show the reference implementation of Algorithm 3 in Section VI.

Remark 1. *Although we propose a decentralized scheme using CG for uplink data detection in Section III-D, a similar decentralization method of CG is not applicable in the downlink. Since we partition the uplink channel matrix* $\mathbf{H}$ *row-wise into C blocks, we should similarly partition the downlink channel matrix column-wise into blocks due to the channel reciprocity; this prevents an expansion analogously to* (7). *Consequently, we focus exclusively on ADMM-based beamforming.*

V. RESULTS

We now analyze the computational complexity and consensus bandwidth of our proposed algorithms. We also show error-rate simulation results in LTE-like massive MU-MIMO uplink and downlink systems. We investigate the performance/complexity

---

[8]The matrices $\mathbf{P}_c = \mathbf{A}_c^{-1}\mathbf{H}_c^H$ or $\mathbf{P}_c = \mathbf{H}_c^H\mathbf{B}_c^{-1}$ could be precomputed to avoid recurrent computations within the ADMM iterations (at line 18 or 20 in Algorithm 3). Instead, we directly compute $\mathbf{A}_c^{-1}\mathbf{H}_c^H(\mathbf{z}_c^{(t)} + \boldsymbol{\lambda}_c^{(t)})$ or $\mathbf{H}_c^H\mathbf{B}_c^{-1}(\mathbf{z}_c^{(t)} + \boldsymbol{\lambda}_c^{(t)})$, which only requires two matrix-vector multiplications; precomputing $\mathbf{P}_c$ requires costly matrix-matrix multiplications. Hence, our complexity analysis in Section VI refers to Algorithm 3.

TABLE I
COMPUTATIONAL COMPLEXITY OF CENTRALIZED AND DECENTRALIZED DATA DETECTION AND BEAMFORMING.

Algorithm	Mode		Preprocessing	1st iteration	Subsequent iterations (each)
ADMM-DL	$S \times S$	TM	$2US^2 + \frac{10}{3}S^3 - \frac{1}{3}S$	$4SU + 4S^2$	$8SU + 4S^2 + 6U + 1$
		AR	$C(2US^2 + \frac{10}{3}S^3 - \frac{1}{3}S)$	$C(4SU + 4S^2)$	$C(8SU + 4S^2 + 2U) + 4U + 1$
	$U \times U$	TM	$2SU^2 + \frac{10}{3}U^3 - \frac{1}{3}U$	$4SU + 4U^2$	$8SU + 4U^2 + 6U + 1$
		AR	$C(2SU^2 + \frac{10}{3}U^3 - \frac{1}{3}U)$	$C(4SU + 4U^2)$	$C(8SU + 4U^2 + 2U) + 4U + 1$
ADMM-UL	$S \times S$	TM	$2US^2 + \frac{10}{3}S^3 + 4US + 4S^2 - \frac{1}{3}S$	$2U$	$8SU + 4S^2 + 4U$
		AR	$C(2US^2 + \frac{10}{3}S^3 + 4US + 4S^2 - \frac{1}{3}S)$	$2U$	$C(8SU + 4S^2 + 2U) + 2U$
	$U \times U$	TM	$2SU^2 + \frac{10}{3}U^3 + 4SU + 4U^2 - \frac{1}{3}U$	$2U$	$4U^2 + 6U$
		AR	$C(2SU^2 + \frac{10}{3}U^3 + 4SU + 4U^2 - \frac{1}{3}U)$	$2U$	$C(4U^2 + 4U) + 2U$
CG-UL		TM	$4SU + 2U$	$8SU + 6U$	$8SU + 12U$
		AR	$4CSU + 2U$	$C(8SU + 4U) + 2U$	$C(8SU + 10U) + 2U$
Centralized	ZF-DL		$6CSU^2 + \frac{10}{3}U^3 + 4CSU - \frac{4}{3}U$		
	MMSE-UL		$6CSU^2 + \frac{10}{3}U^3 + 4CSU - \frac{1}{3}U$		

trade-offs and show practical operating points of our decentralized methods under various antenna configurations, providing design guidelines for decentralized massive MU-MIMO BSs.

A. Computational Complexity

In Table I, we list the number of real-valued multiplications[9] of our decentralized ADMM-based downlink beamforming (ADMM-DL), ADMM-based uplink detection (ADMM-UL) and CG-based uplink detection (CG-UL) algorithms. We also compare the complexity to that of conventional, centralized ZF downlink beamforming (ZF-DL) and MMSE uplink detection (MMSE-UL). For all decentralized algorithms and modes, for example, the "$S \times S$ mode" when $S \leq U$ and the "$U \times U$ mode" when $S > U$, we show the *timing (TM) complexity* and *arithmetic (AR) complexity*. We assume that the centralized computations take place on a centralized PE while decentralized computations are carried out on multiple decentralized PEs. For the centralized computations, both the TM and AR complexities count the number of real-valued multiplications on the centralized PE. For the decentralized operations, the TM complexity only counts operations that take place on *a single* local processing unit where all decentralized local processors perform their own computations *in parallel* at the same time, thus reflecting the latency of algorithm execution; in contrast, the AR complexity counts the *total* complexity accumulated from *all* local processing units, thus reflecting the total hardware costs. The complexity of our methods depends on the number of clusters C, the number of users U, the number of BS antennas S per antenna cluster, and the number of iterations $T_{\max}$ to achieve satisfactory error-rate performance. We also divide the complexity counts into three parts: preprocessing before ADMM or CG iterations, first iteration, and subsequent iterations. The complexity in the first iteration is typically lower as many vectors are zero.

Table I reveals that preprocessing for ADMM exhibits relatively high complexity, whereas CG-based detection is computationally efficient. The per-iteration complexity of

ADMM is, however, extremely efficient (depending on the operation mode). Overall, CG-based data detection is more efficient than the ADMM-based counterpart, whereas the latter enables more powerful regularizers. Centralized ZF or MMSE beamforming or detection, respectively, require high complexity, i.e., scaling with U^3, but generally achieve excellent error-rate performance [4]. We will analyze the trade-offs between complexity and error-rate performance in Section V-D.

B. Consensus Bandwidth

The amount of data passed between the centralized processing unit and the decentralized local units during ADMM or CG iterations scales with the dimension of the consensus vector $\mathbf{w}^{(t)}$. For a single subcarrier, $\mathbf{w}^{(t)}$ is a vector with U complex-valued entries. If we perform detection or precoding for a total of N_{CR} subcarriers, then in each ADMM or CG iteration, we need to gather $U \times N_{\mathrm{CR}}$ complex-valued entries from each local processing unit for consensus vectors corresponding to N_{CR} subcarriers, and broadcast all N_{CR} consensus vectors to each local processor afterwards. Such a small amount of data exchange relaxes the requirement on interconnection bandwidth among decentralized PEs, and avoids the large data transfer between the entire BS antenna array and BS processor in a conventional centralized BS. However, as we will show with our GPU cluster implementation in Section VI, the interconnect *latency* of the network critically effects the throughput of DBP.

C. Error-Rate Performance

We simulate our decentralized data detection and beamforming algorithms in an LTE-based large-scale MIMO system.[10] For both the uplink and downlink simulations, we consider OFDM transmission with 2048 subcarriers in a 20 MHz channel, and incorporate our algorithms with other necessary baseband processing blocks, including 16-QAM modulation with Gray mapping, FFT/IFFT for subcarrier mapping, rate-5/6 convolutional encoding with random interleaving and soft-input Viterbi-based channel decoding [37]. We generate the

---

[9]We ignore data-dependencies or other operations, such as additions, divisions, etc. While this complexity measure is rather crude, it enables insights into the pros and cons of decentralized baseband processing.

[10]A simplified MATLAB simulator for DBP in the uplink and downlink is available on GitHub at https://github.com/VIP-Group/DBP

(a) Error-rate performance of decentralized ADMM uplink data detection.

(b) Error-rate performance of decentralized CG uplink data detection.

(c) Error-rate performance of decentralized ADMM downlink beamforming.
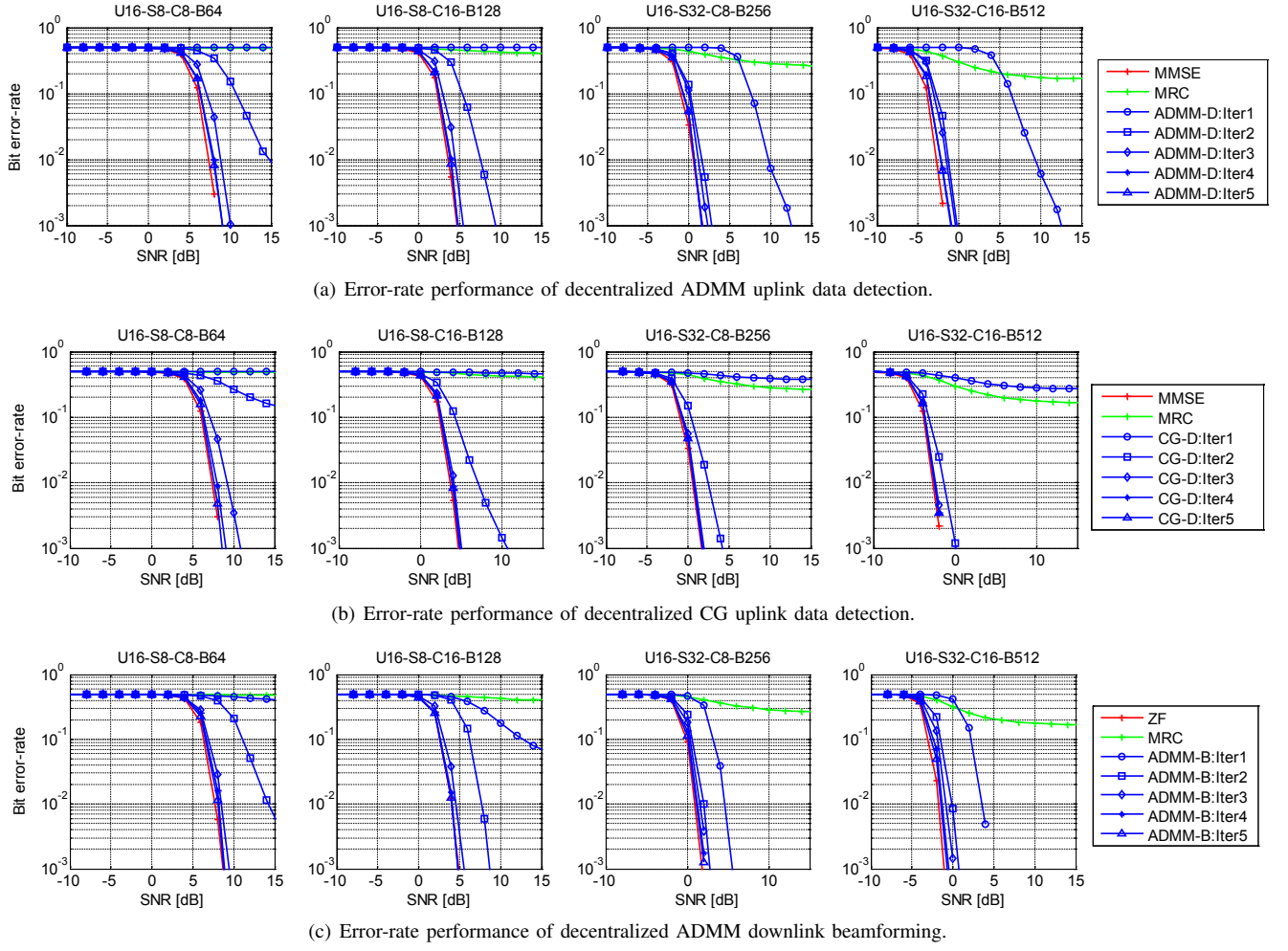
Fig. 2. Bit error-rate (BER) performance of decentralized data detection and beamforming; we use the notation $U - S - C - B$ (representing the number of users U, antennas per clusters S, clusters C, and BS antennas B) as subtitle of each figure to indicate the corresponding system configuration.

channel matrices using the Winner-II channel model [38] and consider channel estimation errors, i.e., we assume a single orthogonal training sequence per user and active subcarrier. For the sake of simplicity, we avoid rate adaptation, the use of cyclic redundancy checks, and (hybrid) ARQ transmission protocols.

In Figure 2, we show the coded bit error-rate (BER) performance against average SNR per receive antenna for decentralized ADMM detection (Figure 2(a)), for decentralized CG detection (Figure 2(b)) in the uplink, and for decentralized ADMM beamforming (Figure 2(c)) in the downlink. We consider various antenna configurations. We fix the number of users $U = 16$, and set $S = 8$ (for $S \leq U$ case) or $S = 32$ (for $S > U$ case), and scale the total BS antenna number $B = S \times C$ from 64 to 512 by choosing $C = 8$ and $C = 16$.

We see that for all the considered antenna and cluster configurations, only 2-to-3 ADMM or CG iterations are sufficient to approach the performance of the linear MMSE equalizer. For the $S > U$ case, even a single ADMM iteration enables excellent BER performance for detection and beamforming without exhibiting an error floor, which outperforms CG with one iteration. We note that the amount of consensus information that must be exchanged during each ADMM or CG iteration

is rather small. Hence, our decentralized data detection and beamforming algorithms are able to achieve the error-rate performance of centralized solutions (such as MMSE and MRC data detection or ZF beamforming) without resulting in prohibitive interconnect or I/O bandwidth—this approach enables highly scalable and modular BS designs with hundreds or thousands of antenna elements.

D. Performance/Complexity Trade-off Analysis

Figure 3 illustrates the trade-off between error-rate performance and computational complexity of our proposed methods. As a performance metric, we consider the minimum required SNR to achieve 1% BER; the complexity is characterized by the TM complexity and depends on the number of ADMM or CG iterations (the numbers next to the curves). As a reference, we also include the BER performance of centralized MMSE data detection and ZF beamforming (dashed vertical lines).

For the uplink, Figures 3(a) and 3(b) show the trade-offs for ADMM-based and CG-based data detection, respectively. We see that only a few CG iterations are sufficient to achieve near-MMSE performance whereas ADMM requires a higher number of iterations to achieve the same performance. CG-based data
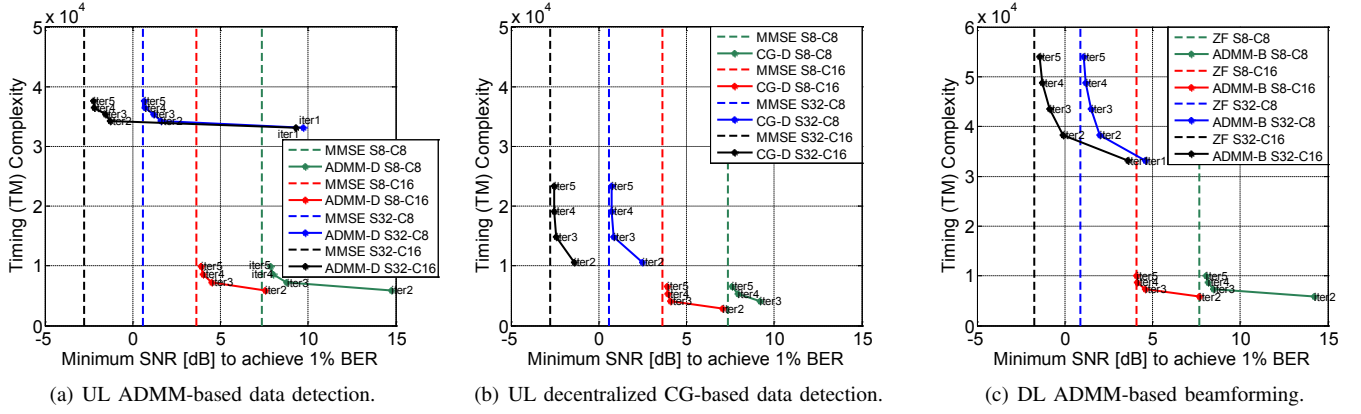
Fig. 3. Performance/complexity trade-off of decentralized data detection and beamforming in an LTE-like massive MU-MIMO system with $U = 16$ users.

detection exhibits the better trade-off here, and is the preferred method. However, for scenarios such as $U < S$, ADMM-based detection exhibits no error floor, even for a single iteration, while CG-based data detection performs rather poorly at one iteration. In addition, our ADMM-based method supports more sophisticated regularizers (such as the BOX regularizer).

For the downlink, Figure 3(c) shows our proposed ADMM-based beamformer. We see that only a few iterations (e.g., 2 to 3 iterations) are necessary to achieve near-optimal performance. In addition, for small antenna cluster sizes (e.g., $S = 8$), the complexity is comparable to CG-based detection; for large antenna cluster sizes, the complexity is only $2\times$ higher.

VI. GPU CLUSTER IMPLEMENTATION

General purpose computing on GPU (GPGPU) is widely used for fast prototyping of baseband algorithms in the context of reconfigurable software-defined radio (SDR) systems [35], [39], [40]. We now present reference implementation results of the proposed decentralized data detection and beamforming algorithms on a GPU cluster to demonstrate the practical scalability of DBP in terms of throughput. We consider a wideband scenario, which enables us to exploit decentralization across subcarriers and in the BS antenna domain. Fig. 4 illustrates the mapping of our algorithms onto the GPU cluster, the main data flow, and the key computing modules. For all our implementations, we use the message passing interface (MPI) library [41] to generate C independent processes on C computing nodes in the GPU cluster, where each process controls a GPU node for accelerating local data detection or beamforming using CUDA [42]. Data collection and broadcasting among GPUs nodes can be realized by MPI function calls over a high-bandwidth Cray Aries [43] or Infiniband [44] interconnect network. We benchmark our implementations for a variety of antenna and cluster configurations to showcase the efficacy and scalability of DBP to very large BS antenna arrays with decentralized computing platforms.

A. Design Mapping and Optimization Strategies

We next discuss the implementation details and optimizations that achieve high throughput with our decentralized algorithms.
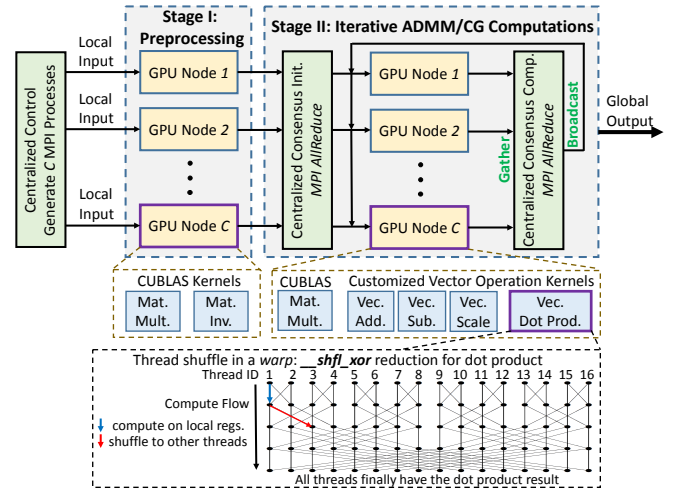

Fig. 4. Mapping of the algorithms on the GPU cluster. Each GPU node performs local data detection (uplink) or beamforming (downlink) on C CUDA GPUs using CUBLAS or customized kernel blocks as detailed in Algorithms 1, 2, and 3. The bottom part illustrates the dot product kernel, in which we use a *warp shuffle* [42] approach to minimize the processing latency.

1) Optimizing kernel computation performance: The local data detection and beamforming computations in each cluster are mapped as GPU kernel functions, which can be invoked with thousands of threads on each GPU node to realize inherent algorithm parallelism and to exploit the massive amount of computing cores and memory resources. All of our decentralized algorithms mainly require matrix-matrix and matrix-vector multiplications. The ADMM methods also involve an explicit matrix inversion step. Such computations are performed efficiently using the *cuBLAS* library [45], a CUDA-accelerated basic linear algebra subprograms (BLAS) library for GPUs. We use the `cublasCgemmBatched` function to perform matrix-matrix multiplications and matrix-vector multiplications, and use `cublasCgetrfBatched` and `cublasCgetriBatched` to perform fast matrix inversions via the Cholesky factorization followed by forward-backward substitution [46]. For these functions, "C" implies that we use complex-valued floating point numbers and "`Batched`" indicates that the function can complete a batch of computations in parallel, which are scaled by the `batchsize` parameter of

the function with a single function call. Since the local data detection or beamforming problems are solved independently for each subcarrier, we can group a batch of subcarriers and process them together to achieve high GPU utilization and throughput. For each data detection or beamforming computation cycle, we define N_{sym} OFDM symbols, each including N_{sc} subcarriers, as the total workload to be processed by such `Batched` kernel function calls. We assume that the channel remains static for N_{sym} symbols.

For the preprocessing stage, the matrix-matrix multiplications and matrix inversions, which only depend on $\mathbf{H}_c$, can be calculated with `batchsize` $= N_{sc}$ for N_{sc} subcarriers in an OFDM symbol, and then broadcast to all N_{sym} symbols inside GPU device memory to reduce complexity. For the matrix-vector multiplications, we invoke *cuBLAS* functions with `batchsize` $= N_{sc} \times N_{sym}$, because these computations depend on transmit or receive symbols as well.

For all other types of computations, such as vector addition/subtraction and inner product calculations, we use customized kernel functions. In this way, we can combine several vector computation steps into a single kernel, and take advantage of local registers or shared memories to store and share intermediate results instead of using slower GPU device memories and multiple *cuBLAS* functions. Vector addition/subtraction for $N_{sc} \times N_{sym}$ number of U-element vectors exposes explicit data parallelism and can proceed with $N_{sc} \times N_{sym} \times U$ GPU threads in parallel. However, the dot product for each pair of U-dimensional vectors requires internal communication among a group of U threads, each thread controlling an element-wise multiplication, to be reduced to a sum. A typical way for such a reduction is to resort to the shared memory, an on-chip manageable cache with L1 cache speed, where a certain group of U threads associated with a vector sums up their element-wise multiplication results to a shared variable atomically, for example, using the `atomicAdd` CUDA function call. However, shared memory typically suffers from higher latency (compared to that of local registers) and also from possible resource competition among multiple threads. In the CG-based data detector, we utilize the *warp shuffle* technique, which is supported by Kepler-generation GPUs for efficient register-to-register data shuffling among threads within a thread *warp* [42], for faster parallel reduction. As shown in Fig. 4, we use the `__shfl_xor(var,laneMask)` intrinsic, which retrieves the register value of variable `var` from a certain thread with lane ID *source_id* for the calling thread with lane ID *dest_id* within the same *warp*, where *source_id* satisfies: *source_id* XOR *dest_id* = `laneMask`, XOR indicating bitwise exclusive or. In this way, the parallel sum reduction for the inner product of two U-element vectors can be realized by `var=__shfl_xor(val,laneMask)+var` in $\log_2(U)$ iterations on a reduction tree with initial `laneMask` $= U$, and `laneMask` reduced to half in each iteration. Finally, as shown in Fig. 4, each of the U threads will have a copy of the inner-product result stored in its own `var`, i.e., the above process is actually an operation of *allreduce* rather than *reduce*, which facilitates downstream computations in which each thread requires the value of the inner product. Here, we assume that the number of user antennas satisfies

TABLE II
PARALLELISM ANALYSIS AND MAPPING STRATEGIES.

Module	Strategy	Parallelism	Memory[1]
Mat. mult.	batch cuBLAS	$N_{sc} \times N_{sym}$	g,c,s,r
Mat. inv.	batch cuBLAS	$N_{sc} \times N_{sym}$	g,c,s,r
Vec. +/-/scale	multi-threading	$N_{sc} \times N_{sym} \times U$	g,c,r
Vec. dot prod.	warp shuffle	$N_{sc} \times N_{sym} \times U$	g,c,r
GPU comm.	MPI, RDMA	Among C GPUs	g

[1]g: global (device) memory; c: cache; s: shared memory, r: register

$U \leq$ *warpsize* $= 32$ and is a power of two, for example, $U = 8$ or $U = 16$. Otherwise, we can resort to the alternative vector inner product solution by `atomicAdd` using shared memory. The optimizations described above enable the computation of each iteration using $N_{sc} \times N_{sym} \times U$ threads using fast on-chip memory resources and efficient inter-thread communication schemes that avoid blocking.

2) Improving message passing efficiency: Message passing latency is critical to the efficiency of our design. For our decentralized algorithms, the consensus operations require data collection and sharing among C GPU nodes. This can be realized by MPI *collective* function calls for inter-process communication among C controlling processes with messages of size $N_{sc} \times N_{sym} \times U$ complex samples. More specifically, we choose the `MPI_Allreduce` function to sum (and then average) vectors across nodes. We then broadcast the resulting consensus vector $\mathbf{w}$ to all local nodes within this single collective MPI function call. Typically, `MPI_Allreduce` operates on the CPU's memory and requires GPU arrays to be copied into a CPU memory buffer before calling `MPI_Allreduce`. To eliminate redundant memory copy operations, we take advantage of *CUDA-aware MPI* [47] and *GPUDirect* remote device memory access (RDMA) techniques [48], which enable the MPI function to explicitly operate on GPU memories without using a CPU intermediary buffer. This results in reduced latency and higher bandwidth. Table II summarizes the key mapping strategies, degrees of parallelism, and associated memory usage for both intra-GPU computing modules and inter-GPU communication mechanisms of our GPU implementations.

B. Implementation Results

We implemented our algorithms on a Navy DSRC Cray XC30 cluster [49] equipped with 32 GPU nodes connected by a Cray Aries network interface. Each node has a 10-core Intel Xeon E5-2670v2 CPU and an Nvidia Tesla K40 GPU with 2880 CUDA cores and 12 GB GDDR5 memory. The Cray Aries network uses a novel Dragonfly topology that enables fast and scalable network communication with a peak all-to-all global bandwidth of 11.7 GB/s per node for the full network [43]. The hybrid MPI and CUDA designs are compiled by Nvidia's nvcc compiler and the Cray compiler, and linked with CUDA's runtime library, the *cuBLAS* library, and the Cray MPICH2 library. Our software implementations can be easily reconfigured with new design parameters, such as number of BS antennas, modulation schemes, etc, and recompiled in a few seconds to enable high design flexibility. In what follows,

TABLE III
LATENCY (L) IN [MS] AND THROUGHPUT (T) IN [MB/S] FOR DECENTRALIZED DATA DETECTION AND BEAMFORMING ($U = 16$).

B	64	128	256	128	256	512	256	512	1024
C	8	16	32	8	16	32	8	16	32
S	8	8	8	16	16	16	32	32	32
Iter.	L / T	L / T	L / T	L / T	L / T	L / T	L / T	L / T	L / T
ADMM-based decentralized uplink data detection									
1	2.060 / 417.5	2.166 / 397.1	2.520 / 341.4	3.810 / 225.7	4.079 / 210.9	4.466 / 192.6	4.329 / 198.7	4.516 / 190.5	4.693 / 183.3
2	4.989 / 172.4	5.411 / 159.0	6.451 / 133.3	6.756 / 127.3	7.597 / 113.2	8.495 / 101.3	7.173 / 119.9	7.855 / 109.5	8.615 / 99.84
3	7.728 / 111.3	8.561 / 100.5	9.910 / 86.80	9.712 / 88.56	10.94 / 78.62	12.68 / 67.85	10.22 / 84.15	11.27 / 76.30	12.98 / 66.28
4	10.80 / 79.67	11.92 / 72.16	13.79 / 62.39	12.44 / 69.12	14.08 / 61.10	16.64 / 51.68	13.49 / 63.78	14.83 / 57.99	17.27 / 49.81
5	13.43 / 64.01	15.69 / 54.83	17.59 / 48.90	15.18 / 56.66	17.88 / 48.10	21.11 / 40.75	16.50 / 52.13	18.65 / 46.12	21.53 / 39.95
CG-based decentralized uplink data detection									
1	3.516 / 244.7	4.077 / 211.0	4.594 / 187.2	3.811 / 225.7	4.325 / 198.9	4.960 / 173.4	4.232 / 203.3	4.729 / 181.9	4.984 / 161.8
2	5.078 / 169.4	6.192 / 138.9	6.574 / 130.8	5.597 / 153.7	6.160 / 139.6	7.190 / 119.6	6.207 / 138.6	7.067 / 121.7	7.150 / 112.8
3	6.567 / 131.0	8.250 / 104.3	8.490 / 101.3	7.323 / 117.5	8.436 / 102.0	9.310 / 92.39	7.841 / 109.7	9.100 / 94.52	9.314 / 86.58
4	8.080 / 106.5	10.02 / 85.87	10.79 / 79.70	9.243 / 93.06	10.22 / 84.12	11.91 / 72.20	9.775 / 88.00	10.90 / 78.94	11.45 / 70.43
5	9.787 / 87.88	11.97 / 71.85	12.74 / 67.52	11.07 / 77.69	11.95 / 71.99	14.03 / 61.30	11.93 / 72.08	13.27 / 64.82	13.61 / 59.25
ADMM-based decentralized downlink beamforming									
1	0.744 / 1156	0.745 / 1155	0.746 / 1154	2.261 / 380.5	2.266 / 379.6	2.271 / 378.8	2.785 / 308.8	2.792 / 308.1	2.797 / 307.6
2	2.685 / 320.3	2.849 / 301.9	2.974 / 289.2	4.048 / 212.5	4.318 / 199.2	4.557 / 188.8	4.618 / 186.3	4.762 / 180.6	5.084 / 169.2
3	4.567 / 188.4	4.954 / 173.6	4.803 / 179.1	5.741 / 149.8	5.966 / 144.2	6.240 / 137.9	6.323 / 136.0	6.651 / 129.3	7.213 / 119.2
4	6.269 / 137.2	6.797 / 126.6	6.769 / 127.1	7.352 / 117.0	7.741 / 111.1	8.720 / 98.65	7.849 / 109.6	8.371 / 102.8	9.380 / 91.70
5	8.055 / 106.8	9.012 / 95.45	8.753 / 98.27	8.766 / 98.12	9.558 / 89.99	10.42 / 82.54	9.570 / 89.88	10.15 / 84.76	11.11 / 77.40
Centralized MMSE uplink data detection and ZF downlink beamforming (baseline implementations)									
MMSE	3.755 / 229.1	5.371 / 160.2	8.607 / 99.94	5.371 / 160.2	8.607 / 99.94	15.04 / 57.20	8.607 / 99.94	15.04 / 57.20	27.97 / 30.75
ZF	4.063 / 211.7	5.936 / 144.9	9.698 / 88.70	5.936 / 144.9	9.698 / 88.70	17.18 / 50.06	9.698 / 88.70	17.18 / 50.06	32.19 / 26.72

we benchmark the latency (in milliseconds) and throughput (in Mb/s) of our implementations based on CPU wall-clock time.

Table III summarizes the latency and throughput performance of ADMM-based decentralized data detection, CG-based decentralized data detection, and ADMM-based decentralized beamforming, depending on the number of iterations $T_{\max}$. We also include the performance of *centralized* MMSE data detection and ZF beamforming designs based on our previous results reported in [35] as a baseline.[11] We consider a scenario with 64-QAM, a coherence interval of $N_{sym} = 7$ symbols, and $N_{sc} = 1200$ active subcarriers, which reflects a typical slot of a 20 MHz LTE frame. We fix the number of users to $U = 16$ and show results for three scenarios: (i) $U > S$ with $S = 8$, (ii) $U = S = 16$, and (iii) $U < S$ with $S = 32$. For each scenario, we vary the number of BS antennas as $B = CS$ for different cluster sizes $C \in \{8, 16, 32\}$. The measured latency includes both kernel-computation and inter-GPU message-passing latencies. The computation latency scales up with local computation workload, while the average message-passing latency is approximately $1 \sim 2$ ms in each ADMM or CG iteration and remains nearly constant for $C \leq 32$ thanks to the scalable Dragonfly topology of Cray Aries.

We see that by increasing the number of clusters C, and hence the total number of BS antennas B, the achieved throughput degrades only slightly; this demonstrates the excellent

scalability of DBP to large antenna arrays. In stark contrast, centralized MMSE and ZF methods suffer from an orders-of-magnitude throughput degradation when increasing the number of BS antennas; this clearly shows the limits of centralized data detection and beamforming methods. We also note that for MIMO systems with a relatively small number of BS antennas, such as, when $B = 64$ or $B = 128$, we see that centralized data detection and precoding is able to achieve a higher throughput than decentralized schemes. We emphasize, however, that centralized processing assumes that one is able to get the raw baseband data into the single, centralized computing fabric at sufficiently high data rates. We furthermore we see that for a given number of clusters C, the throughput for the $S = 32$ case is smaller than that of the $S = 8$ case. The reason for this behavior is the fact that having a large number of antennas per cluster S leads to a higher complexity associated with larger Gram matrix multiplications in each local processing unit while supporting more total BS antennas. For example, for $C = 32$ and $S = 32$, we have $B = 1024$ BS antennas and achieve relatively high throughput. We also see that the CG detector achieves comparable or higher throughput than the ADMM detector for most cases due to its lower computational complexity. Quite surprisingly, the ADMM beamformer can enable even higher performance than both ADMM and CG detectors. In the $S = 8$ case, for example, over 1 Gb/s of beamforming throughput can be achieved using a single ADMM iteration. This behavior is due to the fact that a single ADMM beamforming iteration (Algorithm 3) only requires

---

[11]Centralized data detectors and beamformers for massive MU-MIMO have been implemented on FPGAs and ASICs in, e.g., [50]–[52]. A direct and fair comparison with our GPU implementations is, however, difficult.

local computations but no message passing, while ADMM and CG detectors require message passing. This indicates that, despite the optimizations described above, message passing latency still has a crucial effect on performance and further improvements in messaging may yield even higher data rates.

Remark 2. *We emphasize these GPU cluster implementations serve as a proof-of-concept to showcase the efficacy and design scalability of DBP to large BS antenna arrays. The achieved throughputs are by no means high enough for 5G wireless systems, which is mainly a result of the relatively high interconnect latency. Nevertheless, we expect that DBP achieves throughputs in the Gb/s regime if implemented on FPGA or ASIC clusters, which offer higher computing efficiency and lower interconnect latency (e.g., using Xilinx's GTY [53] or Aurora protocols [54]) than that of GPU clusters.*

Remark 3. *Power efficiency is another key aspect of practical BS designs. The thermal design power (TDP) of the Tesla K40 GPU used in our implementation is $235\,W$, leading to a maximum power dissipation of $C \times 235\,W$ with C fully-utilized GPUs. While this is a pessimistic power estimate, we expect that dedicated implementations on FPGA or ASIC will yield orders-of-magnitude better performance per watt.*

VII. CONCLUSIONS

We have proposed a novel decentralized baseband processing (DBP) architecture for massive MU-MIMO BS designs that mitigates interconnect and chip I/O bandwidth as well as complexity and signal processing bottlenecks. DBP partitions the BS antenna array into independent clusters which perform channel estimation, data detection, and beamforming in a decentralized and parallel manner by exchanging only a small amount of consensus information among the computing fabrics. The proposed data detection and beamforming algorithms achieve near-optimal error-rate performance at low complexity. Furthermore, our simple consensus algorithms have low bandwidth requirements. Our GPU cluster implementation shows that the proposed method scales well to BS designs with thousands of antenna elements, and demonstrates that DBP enables the deployment of modular and scalable BS architectures for realistic massive MU-MIMO systems.

We see numerous avenues for future work. A rigorous error-rate performance analysis of the proposed algorithms is an open research topic. The integration of our intra-cell decentralization schemes with inter-cell CoMP and C-RAN frameworks is a direction worth to pursue in the future. The development of decentralized algorithms for other 5G waveform candidates, such as SC-FDMA, FMBC, or GFDM, is left for future work. To alleviate the latency bottleneck, decentralized *feedforward* architectures as in [55] should be investigated for the downlink. Finally, an implementation of DBP on clusters with computing fabrics that have low interconnect latency and power consumption, such as FPGA or ASIC clusters, or heterogeneous or hybrid processors and accelerators for optimized workload deployment is part of ongoing work.

APPENDIX A
PROOFS

A. Proof of Lemma 1

We start by reformulating Step E2 as follows:

$$\mathbf{s}^{(t+1)} = \arg\min_{\mathbf{s} \in \mathbb{C}^U} g(\mathbf{s}) + \sum_{c=1}^{C} \frac{\rho}{2} \left\| \mathbf{s} - \mathbf{w}_c^{(t)} \right\|_2^2, \quad (9)$$

where we use the shorthand $\mathbf{w}_c^{(t)} = \mathbf{z}_c^{(t+1)} + \boldsymbol{\lambda}_c^{(t)}$. Let $\mathbf{v}^{(t)} = \frac{1}{C}\mathbf{w}^{(t)} = \frac{1}{C}\sum_{c=1}^{C}\mathbf{w}_c^{(t)}$. Then, we can complete the square in the sum of the objective function of (9), which yields

$$\sum_{c=1}^{C} \left\| \mathbf{s} - \mathbf{w}_c^{(t)} \right\|_2^2 = C\|\mathbf{s}\|_2^2 - \mathbf{s}^H C \mathbf{v}^{(t)} - (\mathbf{v}^{(t)})^H C \mathbf{s}$$
$$+ \sum_{c=1}^{C} \left\| \mathbf{w}_c^{(t)} \right\|_2^2 = C \left\| \mathbf{s} - \mathbf{v}^{(t)} \right\|_2^2 + K,$$

where we define the constant $K = \sum_{c=1}^{C} \left\| \mathbf{w}_c^{(t)} \right\|_2^2 - C \left\| \mathbf{v}^{(t)} \right\|_2^2$. Since K is independent of the minimization variable $\mathbf{s}$ in (9), we obtain the equivalent minimization problem in (5).

B. Proof of Lemma 2

We start by reformulating Step (P2) as follows:

$$\mathbf{z}^{(t+1)} = \arg\min_{\mathbf{z} \in \mathbb{C}^{UC}, \|\mathbf{s}-\mathbf{Dz}\|_2 \leq \varepsilon} \frac{1}{2} \left\| \mathbf{w}_{\text{all}}^{(t)} - \mathbf{z} \right\|_2^2, \quad (10)$$

where we define $\mathbf{D} = \mathbf{1}_{1 \times C} \otimes \mathbf{I}_U$, $\mathbf{z}^T = [\mathbf{z}_1^T \cdots \mathbf{z}_C^T]$, and $(\mathbf{w}_{\text{all}}^{(t)})^T = [(\mathbf{w}_1^{(t)})^T \cdots (\mathbf{w}_C^{(t)})^T]$ with $\mathbf{w}_c^{(t)} = \mathbf{H}_c \mathbf{x}_c^{(t+1)} - \boldsymbol{\lambda}_c^{(t)}$. Now, observe that the minimization problem (10) is the orthogonal projection of $\mathbf{w}_{\text{all}}^{(t)}$ onto the constraint $\|\mathbf{s} - \mathbf{Dz}\|_2 \leq \varepsilon$. We have the following closed-form expression for $\mathbf{z}^{(t+1)}$ [56]:

$$\mathbf{w}_{\text{all}}^{(t)} + \max\left\{0, 1 - \frac{\varepsilon}{\|\mathbf{s}-\mathbf{Dw}_{\text{all}}^{(t)}\|_2}\right\} \mathbf{D}^H (\mathbf{DD}^H)^{-1} (\mathbf{s} - \mathbf{Dw}_{\text{all}}^{(t)}).$$

We can simplify this expression using the identity

$$(\mathbf{DD}^H)^{-1} = ((\mathbf{1}_{1 \times C} \otimes \mathbf{I}_U)(\mathbf{1}_{1 \times C} \otimes \mathbf{I}_U)^H)^{-1}$$
$$= (C \otimes \mathbf{I}_U)^{-1} = C^{-1} \mathbf{I}_U$$

and $\mathbf{D}^H \mathbf{D} = \mathbf{1}_{C \times C} \otimes \mathbf{I}_U$. With these results, we obtain the following equivalent expression for $\mathbf{z}^{(t+1)}$

$$\mathbf{w}_{\text{all}}^{(t)} + \max\left\{0, 1 - \frac{\varepsilon}{\|\mathbf{s}-\mathbf{Dw}_{\text{all}}^{(t)}\|_2}\right\} \times$$
$$\frac{1}{C}\left(\mathbf{D}^H \mathbf{s} - (\mathbf{1}_{C \times C} \otimes \mathbf{I}_U)\mathbf{w}_{\text{all}}^{(t)}\right),$$

which can be written using the per-cluster variables as

$$\mathbf{z}_c^{(t+1)} = \mathbf{w}_c^{(t)} + \max\left\{0, 1 - \frac{\varepsilon}{\|\mathbf{s}-\mathbf{v}^{(t)}\|_2}\right\}\left(\frac{1}{C}\mathbf{s} - \mathbf{v}^{(t)}\right)$$

with $\mathbf{v}^{(t)} = \frac{1}{C}\mathbf{w}^{(t)} = \frac{1}{C}\sum_{c=1}^{C}\mathbf{w}_c^{(t)}$; $\mathbf{w}_c^{(t)} = \mathbf{H}_c \mathbf{x}_c^{(t+1)} - \boldsymbol{\lambda}_c^{(t)}$.

REFERENCES

[1] K. Li, R. Sharan, Y. Chen, T. Goldstein, J. R. Cavallaro, and C. Studer, "Decentralized Beamforming for Massive MU-MIMO on a GPU Cluster," in *Proc. of IEEE Global Conf. on Signal and Information Processing*, Dec. 2016.

[2] K. Li, R. Sharan, Y. Chen, T. Goldstein, J. R. Cavallaro, and C. Studer, "Decentralized data detection for massive MU-MIMO on a Xeon Phi cluster," in *Proc. of Asilomar Conf. on Signals, Systems, and Computers*, Oct. 2016.

[3] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An Overview of Massive MIMO: Benefits and Challenges," *IEEE J. Sel. Topics in Sig. Proc.*, vol. 8, no. 5, pp. 742–758, Oct. 2014.

[4] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 186–195, Feb. 2014.

[5] http://www.cpri.info, *Common public radio interface*.

[6] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical Many-antenna Base Stations," in *Proc. of the 18th Annual Intl. Conf. on Mobile Computing and Networking*, Aug. 2012, pp. 53–64.

[7] C. Shepard, H. Yu, and L. Zhong, "ArgosV2: A Flexible Many-antenna Research Platform," in *Proc. of the 19th Annual Intl. Conf. on Mobile Computing and Networking*, Aug. 2013, pp. 163–166.

[8] J. Vieira, S. Malkowsky, K. Nieman, Z. Miers, N. Kundargi, L. Liu, I. Wong, V. Öwall, O. Edfors, and F. Tufvesson, "A flexible 100-antenna testbed for Massive MIMO," in *2014 IEEE Globecom Workshops*, Dec. 2014, pp. 287–293.

[9] Q. Yang, X. Li, H. Yao, Ji. Fang, K. Tan, W. Hu, J. Zhang, and Y. Zhang, "BigStation: Enabling Scalable Real-time Signal Processing in Large MU-MIMO Systems," in *Proc. of the ACM Conference on SIGCOMM*, Aug. 2013, pp. 399–410.

[10] J. Hoydis, S. ten Brink, and M. Debbah, "Massive MIMO in the UL/DL of Cellular Networks: How Many Antennas Do We Need?," *IEEE J. Sel. Areas in Commun.*, vol. 31, no. 2, pp. 160–171, Feb. 2013.

[11] http://www.ni.com/white paper/52382/en/, *5G Massive MIMO Testbed: From Theory to Reality*.

[12] R. Irmer, H. Droste, P. Marsch, M. Grieger, G. Fettweis, S. Brueck, H. P. Mayer, L. Thiele, and V. Jungnickel, "Coordinated multipoint: Concepts, performance, and field trial results," *IEEE Commun. Mag.*, vol. 49, no. 2, pp. 102–111, Feb. 2011.

[13] M. Sawahashi, Y. Kishiyama, A. Morimoto, D. Nishikawa, and M. Tanno, "Coordinated multipoint transmission/reception techniques for LTE-advanced," *IEEE Wireless Commun.*, vol. 17, no. 3, pp. 26–34, June 2010.

[14] D. Lee, H. Seo, B. Clerckx, E. Hardouin, D. Mazzarese, S. Nagata, and K. Sayana, "Coordinated multipoint transmission and reception in LTE-advanced: deployment scenarios and operational challenges," *IEEE Commun. Mag.*, vol. 50, no. 2, pp. 148–155, Feb. 2012.

[15] V. Jungnickel, K. Manolakis, W. Zirwas, B. Panzner, V. Braun, M. Lossow, M. Sternad, R. Apelfrojd, and T. Svensson, "The role of small cells, coordinated multipoint, and massive MIMO in 5G," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 44–51, May 2014.

[16] C. Choi, L. Scalia, T. Biermann, and S. Mizuta, "Coordinated multipoint multiuser-MIMO transmissions over backhaul-constrained mobile access networks," in *IEEE 22nd Int. Symp. on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2011, pp. 1336–1340.

[17] W. W. L. Ho, T. Q. S. Quek, S. Sun, and R. W. Heath, "Decentralized Precoding for Multicell MIMO Downlink," *IEEE Trans. on Wireless Commun.*, vol. 10, no. 6, pp. 1798–1809, June 2011.

[18] T. M. Kim, F. Sun, and A. J. Paulraj, "Low-Complexity MMSE Precoding for Coordinated Multipoint With Per-Antenna Power Constraint," *IEEE Signal Processing Letters*, vol. 20, no. 4, pp. 395–398, April 2013.

[19] C. Shen, T. H. Chang, K. Y. Wang, Z. Qiu, and C. Y. Chi, "Distributed Robust Multicell Coordinated Beamforming With Imperfect CSI: An ADMM Approach," *IEEE Trans. on Sig. Proc.*, vol. 60, no. 6, pp. 2988–3003, June 2012.

[20] M. Peng, Y. Li, Z. Zhao, and C. Wang, "System architecture and key technologies for 5G heterogeneous cloud radio access networks," *IEEE Network*, vol. 29, no. 2, pp. 6–14, Mar. 2015.

[21] C. Liu, K. Sundaresan, M. Jiang, S. Rangarajan, and G. K. Chang, "The case for re-configurable backhaul in cloud-RAN based small cell networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1124–1132.

[22] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks - A Technology Overview," *IEEE Commun. Surveys Tutorials*, vol. 17, no. 1, pp. 405–426, 2015.

[23] S. Malkowsky, J. Vieira, K. Nieman, N. Kundargi, I. Wong, V. Owall, O. Edfors, F. Tufvesson, and L. Liu, "Implementation of Low-latency Signal Processing and Data Shuffling for TDD Massive MIMO Systems," in *IEEE Workshop on Sig. Proc. Systems*, Oct. 2016.

[24] E. N. Tunali, M. Wu, C. Dick, and C. Studer, "Linear large-scale MIMO data detection for 5G multi-carrier waveform candidates," in *49th Asilomar Conf. on Signals, Systems, and Computers*, Nov. 2015, pp. 1149–1153.

[25] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Trans. on Commun.*, vol. 45, no. 12, pp. 1613–1621, Dec. 1997.

[26] C. Studer, S. Fateh, and D. Seethaler, "ASIC Implementation of Soft-Input Soft-Output MIMO Detection Using MMSE Parallel Interference Cancellation," *IEEE J. of Solid-State Circuits*, vol. 46, no. 7, pp. 1754–1765, July 2011.

[27] P. H. Tan, L. K. Rasmussen, and T. J. Lim, "Box-constrained maximum-likelihood detection in CDMA," in *Proc. of Intl. Zurich Seminar on Broadband Communications. Accessing, Transmission, Networking*, 2000, pp. 55–62.

[28] C. Thrampoulidis, E. Abbasi, W. Xu, and B. Hassibi, "BER analysis of the box relaxation for BPSK signal recovery," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 3776–3780.

[29] C. Jeon, A. Maleki, and C. Studer, "On the performance of mismatched data detection in large MIMO systems," in *IEEE Intl. Symposium on Info. Theory (ISIT)*, July 2016, pp. 180–184.

[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[31] M. Hong and Z.-Q. Luo, "On the linear convergence of the alternating direction method of multipliers," *Math. Programming*, pp. 1–35, 2016.

[32] N. J. Higham, *Accuracy and stability of numerical algorithms*, Siam, 2002.

[33] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge Univ. Press, 2004.

[34] B. Yin, M. Wu, J. R. Cavallaro, and C. Studer, "Conjugate gradient-based soft-output detection and precoding in massive MIMO systems," in *2014 IEEE Global Communications Conference*, Dec 2014, pp. 3696–3701.

[35] K. Li, B. Yin, M. Wu, J. R. Cavallaro, and C. Studer, "Accelerating massive MIMO uplink detection on GPU for SDR systems," in *IEEE Dallas Circuits and Systems Conf. (DCAS)*, Oct. 2015.

[36] T. Goldstein and S. Setzer, "High-order methods for basis pursuit," *UCLA CAM Report*, pp. 10–41, 2010.

[37] C. Studer, S. Fateh, C. Benkeser, and Q. Huang, "Implementation trade-offs of soft-input soft-output MAP decoders for convolutional codes," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2774–2783, Nov. 2012.

[38] *WINNER Phase II Model*.

[39] K. Li, M. Wu, G. Wang, and J. R. Cavallaro, "A high performance GPU-based software-defined basestation," in *Proc. 48th Asilomar Conf. on Signals, Systems and Computers*, Nov. 2014, pp. 2060–2064.

[40] M. Wu, S. Gupta, Y. Sun, and J. R. Cavallaro, "A GPU implementation of a real-time MIMO detector," in *IEEE Workshop on Sig. Proc. Systems*, Oct. 2009, pp. 303–308.

[41] https://computing.llnl.gov/tutorials/mpi, *Message passing interface*.

[42] http://docs.nvidia.com/cuda, *Nvidia CUDA programming guide*.

[43] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray XC Series Network," *Cray Inc., White Paper WP-Aries01-1112 (2012)*.

[44] https://en.wikipedia.org/wiki/InfiniBand, *infiniband*.

[45] http://docs.nvidia.com/cuda/cublas, *Nvidia cuBLAS library*.

[46] G. H. Golub and C. F. van Loan, *Matrix Computations, 3rd ed*, The Johns Hopkins Univ. Press, 1996.

[47] https://devblogs.nvidia.com/parallelforall/introduction-cuda-aware-mpi/, *CUDA-aware MPI*.

[48] http://docs.nvidia.com/cuda/gpudirect rdma, *GPU Direct RDMA*.

[49] https://navydsrc.hpc.mil, *Navy DSRC, Cray XC30 User Guide*.

[50] H. Prabhu, J. Rodrigues, L. Liu, and O. Edfors, "3.6 a 60pj/b 300Mb/s 128×8 massive MIMO precoder-detector in 28nm FD-SOI," in *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*, Feb. 2017, pp. 60–61.

[51] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, "Large-scale MIMO detection for 3GPP LTE: Algorithms and FPGA implementations," *IEEE J. of Sel. Topics in Sig. Proc.*, vol. 8, no. 5, pp. 916–929, Oct. 2014.

[52] B. Yin, M. Wu, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, "A 3.8 Gb/s large-scale MIMO detector for 3GPP LTE-Advanced," in *IEEE Intl. Conf. Acoustics, Speech and Sig. Process. (ICASSP)*, May 2014, pp. 3879–3883.

[53] *Xilinx High Speed Serial*.

[54] *Xilinx Aurora Protocol*.

[55] C. Jeon, K. Li, J. R. Cavallaro, and C. Studer, "On the achievable rates of decentralized equalization in massive MU-MIMO systems," in *IEEE Intl. Symposium on Info. Theory (ISIT)*, June 2017, pp. 1102–1106.

[56] C. Studer, T. Goldstein, W. Yin, and R. G. Baraniuk, "Democratic Representations," *arXiv preprint: 1401.3420*, Apr. 2015.