Feedforward Architectures for Decentralized Precoding in Massive MU-MIMO Systems

Author(s):
Li, Kaipeng; Jeon, Charles; Cavallaro, Joseph R.; Studer, Christoph (iD)

Feedforward Architectures for Decentralized Precoding in Massive MU-MIMO Systems

Kaipeng Li[1], Charles Jeon[2], Joseph R. Cavallaro[1], and Christoph Studer[2]

[1]Department of Electrical and Computer Engineering, Rice University, Houston, TX
[2]School of Electrical and Computer Engineering, Cornell University, Ithaca, NY

Abstract—**Massive multi-user multiple-input multiple-output (MU-MIMO) enables significant gains in spectral efficiency and link reliability compared to conventional, small-scale MIMO technology. In addition, linear precoding using zero forcing or Wiener filter (WF) precoding is sufficient to achieve excellent error-rate performance in the massive MU-MIMO downlink. However, these methods typically require centralized processing at the base-station (BS), which causes (i) excessively high interconnect and chip input/output data rates, and (ii) high implementation complexity. We propose two feedforward architectures and corresponding decentralized WF precoders that parallelize precoding across multiple computing fabrics, effectively mitigating the limitations of centralized approaches. To demonstrate the efficacy of our decentralized precoders, we provide implementation results on a multi-GPU system, which show that our solutions achieve throughputs in the Gbit/s regime while achieving (near-)optimal error-rate performance in the massive MU-MIMO downlink.**

I. Introduction

Massive multi-user (MU) multiple-input multiple-output (MIMO) will be among the core technologies of fifth-generation (5G) cellular wireless systems [1]. The key idea of this technology is to equip the infrastructure base-stations (BSs) with hundreds to thousands of antenna elements while serving tens of user equipments (UEs) at the same time and in the same frequency band. The fine-grained nature of beamforming enabled by massive MU-MIMO antenna arrays and coherent transmission yields significantly improved spectral efficiency, coverage, and range compared to that of traditional, small-scale multi-antenna wireless systems [2], [3]. Unfortunately, the advantages of massive MU-MIMO come at the cost of significant practical implementation challenges, which must be solved to realize the gains of this technology in practice.

A. Interconnect Bandwidth and Complexity Bottlenecks

As discussed in [4]–[7], the excessively high amount of data that must be transferred between the baseband processing unit and the antenna array is among the most critical challenges. For example, the raw baseband data rates, from and to the
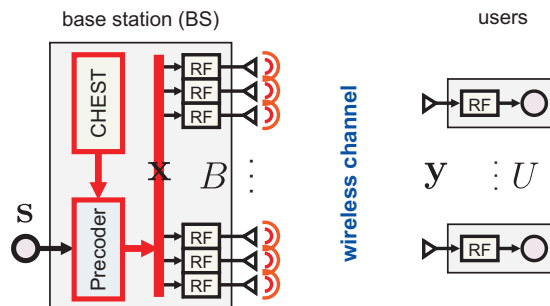
Fig. 1. Overview of a centralized baseband processing architecture for the massive MU-MIMO downlink. One of the most critical implementation challenges is the excessively high amount of raw baseband data that must be transferred between the computing fabric and the RF chains (shown in red).

radio-frequency (RF) chains, of a 128 antenna massive MU-MIMO system with 10 bit digital-to-analog converters (DACs) for a bandwidth of 40 MHz exceed 200 Gbit/s, which not only poses significant challenges for existing interconnect technology, such as the Common Public Radio Interface (CPRI) [8], but also for the input/output (I/O) interfaces of existing computing fabrics, such as application-specific integrated circuits (ASICs), field-programable gate arrays (FPGAs), or graphics processing units (GPUs). Figure 1 illustrates a conventional centralized baseband processing architecture for the massive MU-MIMO downlink (BS transmits to UEs), which suffers from interconnect bandwidth and complexity bottlenecks (highlighted with red color). While maximum ratio transmission (MRT) enables fully distributed channel estimation (CHEST) and beamforming in the downlink, which alleviates the bandwidth and I/O bottlenecks, MRT is unable to fully exploit the spectral-efficiency advantages of massive MU-MIMO [2]. In contrast, more sophisticated precoding strategies, such as zero-forcing (ZF) or Wiener filter (WF) precoding [9], enable near-optimal spectral efficiency. Such methods, however, require centralized baseband processing, which results in high interconnect bandwidth, I/O data rates, and complexity [2].

B. Decentralized Baseband Processing

To mitigate the bandwidth and complexity bottlenecks of centralized baseband processing algorithms, a variety of solutions have been proposed recently. For example, existing massive MU-MIMO testbeds parallelize the computations

across orthogonal subcarriers [10], [11]. Parallelization across subcarriers, however, exhibits dependence between the subcarriers and all antenna elements, which prevents its straightforward use for arrays with thousands of antennas. More recently, references [12], [13] proposed *decentralized baseband processing* (DBP), an approach that decentralizes the key computations required for baseband processing in massive MU-MIMO systems in order to alleviate the bandwidth and complexity bottlenecks. The idea of DBP is to partition the antenna array into clusters, each associated with separate RF circuitry and processing fabrics. Each antenna cluster then only communicates with the associated processing fabrics that carry out (de-)modulation, channel estimation, data detection, and precoding, while only exchanging a small amount of consensus information among the clusters. However, as it has been demonstrated with real-world implementations on GPU clusters in [5], the iterative exchange of consensus information among the clusters negatively affects the processing latency and throughput. As an effective remedy, reference [14] proposed the use of *feedforward architectures* for equalization the uplink (UEs transmit to BS). Such architectures have, up to this point, not been studied for the downlink.

C. Contributions

We propose two new feedforward architectures and corresponding algorithms for decentralized precoding in the massive MU-MIMO downlink. Both architectures are compatible with the ones proposed for the uplink in [14] and prevent the repeated exchange of consensus information to effectively reduce latency and throughput. For both architectures, we propose linear precoding algorithms that build upon the WF precoder that minimizes the mean-square error (MSE) at the UE side. We show that the WF precoder for the *partially decentralized* (PD) architecture achieves the same performance as the centralized WF precoder; the WF precoder for the *fully decentralized* (FD) architecture further reduces the interconnect bandwidth at a small loss in error-rate performance. We demonstrate the efficacy of our feedforward architectures and precoding algorithms using real-world implementations on a multi graphics processing unit (GPU) system. Our implementation results reveal that decentralized precoding with feedforward architectures reaches throughputs in the Gb/s regime while achieving (near-)optimal error-rate performance.

D. Notation

Lowercase and uppercase boldface letters denote column vectors and matrices, respectively. The transpose and Hermitian transpose of the matrix $\mathbf{A}$ are dentoed by $\mathbf{A}^T$ and $\mathbf{A}^H$, respectively. The $M \times M$ identity matrix is denoted by $\mathbf{I}_M$. We use $\mathrm{tr}(\mathbf{A})$ to denote the trace of the matrix $\mathbf{A}$ and $\mathbb{E}_\mathbf{a}[\cdot]$ to denote expectation with respect to the random vector $\mathbf{a}$.

II. System Model and Centralized Precoding

We now introduce the system model and discuss the basics of centralized precoding for massive MU-MIMO systems.

A. Downlink System Model

We focus on the massive MU-MIMO downlink. The system consists of a base-station (BS) with B antennas serving U single-antenna user-equipments (UEs) in the same time-frequency resource. We consider a block-fading and narrowband[1] scenario modeled as follows:

$$\mathbf{y}[k] = \mathbf{H}\mathbf{x}[k] + \mathbf{n}[k], \quad k = 1, \dots, K. \quad (1)$$

Here, the U-dimensional vector $\mathbf{y}[k] = \begin{bmatrix} y_1[k], \dots, y_U[k] \end{bmatrix}^T$ contains the signals received at all U UEs with $y_u[k] \in \mathbb{C}$ corresponding to the signal received at UE u in time slot k. The matrix $\mathbf{H} \in \mathbb{C}^{U \times B}$ represents the downlink MIMO channel and is assumed to remain constant for K time slots. The vector $\mathbf{n}[k] \in \mathbb{C}^U$ models additive noise and is assumed to be i.i.d. circularly-symmetric complex Gaussian with variance N_0 per complex entry. We assume the channel matrix $\mathbf{H}$ and noise variance N_0 to be known perfectly at the BS. The precoded vector $\mathbf{x}[k] \in \mathbb{C}^B$ at time slot k is given by the function

$$\mathbf{x}[k] = \mathcal{P}(\mathbf{s}[k], \mathbf{H}, N_0, \rho^2),$$

which depends on transmit signal vector $\mathbf{s}[k] \in \mathcal{O}^U$, where $\mathcal{O}$ is the constellation set (e.g., 64-QAM), the channel matrix $\mathbf{H}$, the noise variance N_0, and the power constraint ρ^2. The precoded vector is assumed to satisfy the average power constraint

$$\mathbb{E}_\mathbf{s} \big[\|\mathbf{x}[k]\|^2 \big] \leq \rho^2, \quad k = 1, \dots, K, \quad (2)$$

and the vector $\mathbf{s}[k] = \begin{bmatrix} s_1[k], \dots, s_U[k] \end{bmatrix}^T$ contains the information symbols $s_u[k] \in \mathcal{O}$ to be transmitted to UE u in time slot k. In what follows, we omit the time-slot index k.

B. Linear Wiener Filter (WF) Precoding

Since the UEs are unable to perform joint signal processing, the BS has to precode the information symbols with the goals of mitigating multi-user interference (MUI) and focusing power towards the UEs. The literature describes numerous optimization criteria for precoding, such as sum-rate throughput or error probability [15]. In what follows, we focus on linear precoders of the form $\mathbf{x} = \mathbf{P}\mathbf{s}$ that minimize the mean-square error (MSE) between the estimated symbol vector $\hat{\mathbf{s}}$ and the transmit signal vector $\mathbf{s}$. Since coherent transmission with a multi-antenna BS leads to an array gain, we assume that the UEs are able to scale the received signals y_u, $u = 1, \dots, U$, by a precoding factor $\beta_u \in \mathbb{C}$, i.e., the UEs compute estimates of the transmit symbols as follows:

$$\hat{s}_u = \beta_u y_u.$$

While each UE u would able to estimate their own precoding factor β_u, we design precoders that minimize the MSE for a joint[2] precoding factor $\beta \in \mathbb{C}$ defined as [9]

$$MSE = \mathbb{E}_{\mathbf{s},\mathbf{n}} \big[\|\mathbf{s} - \hat{\mathbf{s}}\|^2 \big] = \mathbb{E}_{\mathbf{s},\mathbf{n}} \big[\|\mathbf{s} - \beta \mathbf{y}\|^2 \big]$$

---

[1]An extension to wideband systems that use orthogonal frequency division multiplexing (OFDM) is straightforward and considered in Section IV.

[2]Designing precoders for the case of having individual precoding factors β_u, $u = 1, \dots, U$, is left for future work.
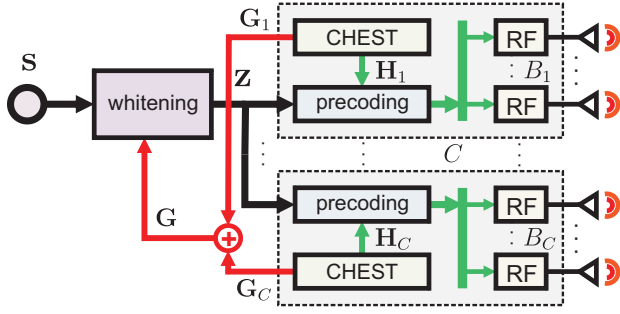
(a) Partially decentralized (PD) precoding architecture.
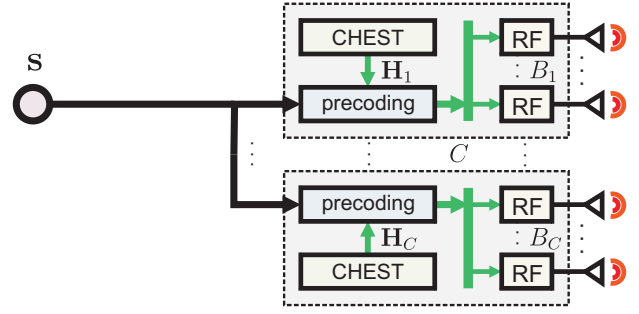
(b) Fully decentralized (FD) precoding architecture.

Fig. 2. Partially decentralized (PD) and fully decentralized (FD) precoding architectures for the massive MU-MIMO downlink with C clusters. (a) PD performs decentralized channel estimation (CHEST) in the uplink and averages the partial Gram matrices $\mathbf{G}_c$ while feeding them to the centralized whitening unit; the $\oplus$ operator denotes matrix addition. In the downlink, precoding is performed in two steps: centralized whitening followed by decentralized precoding in each cluster. (b) FD performs decentralized CHEST in the uplink. In the downlink, precoding is performed locally at each cluster in a fully decentralized manner.

$$= \mathbb{E}_{\mathbf{s}}\big[\|\mathbf{s} - \beta\mathbf{H}\mathbf{x}\|^2\big] + |\beta|^2 U N_0.$$

The resulting MSE-optimal linear precoding matrix $\mathbf{P} \in \mathbb{C}^{B \times U}$ can be designed by solving the following optimization problem

$$\{\mathbf{P}^{\mathrm{WF}}, \beta^{\mathrm{WF}}\} = \begin{cases} \underset{\mathbf{P},\beta}{\text{minimize}} & \mathbb{E}_{\mathbf{s}}\big[\|\mathbf{s} - \beta\mathbf{H}\mathbf{P}\mathbf{s}\|^2\big] + \beta^2 U N_0 \\ \text{subject to} & \mathbb{E}_{\mathbf{s}}\big[\|\mathbf{P}\mathbf{s}\|^2\big] \le \rho^2. \end{cases} \quad (3)$$

The solution to this optimization problem is known as the Wiener filter (WF) precoder [9] and is summarized by the following result; a short proof is given in Appendix A.

Theorem 1. *The Wiener filter (WF) precoding matrix $\mathbf{P}^{\mathrm{WF}}$ is given by $\mathbf{P}^{\mathrm{WF}} = \frac{1}{\beta^{\mathrm{WF}}}\mathbf{Q}^{\mathrm{WF}}$, where we define the matrix*

$$\mathbf{Q}^{\mathrm{WF}} = \big(\mathbf{H}^H\mathbf{H} + \kappa^{\mathrm{WF}}\mathbf{I}_B\big)^{-1}\mathbf{H}^H. \quad (4)$$

The associated regularization parameter κ^{WF} and precoding factor β^{WF} are defined as

$$\kappa^{\mathrm{WF}} = \frac{U N_0}{\rho^2} \quad and \quad \beta^{\mathrm{WF}} = \sqrt{\frac{\mathrm{tr}\big((\mathbf{Q}^{\mathrm{WF}})^H\mathbf{Q}^{\mathrm{WF}})E_s\big)}{\rho^2}}. \quad (5)$$

A straightforward computation of the precoding factor β^{WF} in (5) involves the inversion of a $B \times B$ matrix followed by a number of matrix-matrix multiplications. We propose a computationally-efficient alternative that can be implemented efficiently given the $U \times U$ whitening matrix $\mathbf{A}^{-1}$ has been precomputed; a short proof is given in Appendix B.

Lemma 2. *The precoding factor β^{WF} of the WF precoder in (5) can be computed efficiently as follows:*

$$\beta^{\mathrm{WF}} = \sqrt{\frac{E_s}{\rho^2}\big(\mathrm{tr}\,(\mathbf{A}^{-1}) - \kappa^{\mathrm{WF}}\|\mathbf{A}^{-1}\|_F^2\big)}. \quad (6)$$

III. DECENTRALIZED PRECODING

We next propose two decentralized precoding schemes that rely on feedforward architectures and linear WF precoding.

A. System Model for Decentralized Precoding

We extend the feedforward architecture put forward in [14] for the uplink by the capability to perform downlink precoding. In contrast to the centralized processing architecture shown in Figure 1, we partition the BS antenna array into $C \ge 1$ clusters. This decentralized architecture alleviates the chip-I/O bottlenecks, as shown in Figure 2. Each cluster is associated with $B_c = w_c B \in \mathbb{N}^+$ BS antennas so that $w_c \in (0, 1]$ and $\sum_{c=1}^{C} w_c = 1$, and contains local RF circuitry and requires access to only local channel state information (CSI) acquired in the uplink via reciprocity. By omitting the time-slot index k, we can rewrite the downlink system model in (1) as

$$\mathbf{y} = \sum_{c=1}^{C} \mathbf{H}_c\mathbf{x}_c + \mathbf{n}, \quad (7)$$

where $\mathbf{H} = [\mathbf{H}_1, \ldots, \mathbf{H}_C]$ with $\mathbf{H}_c = \mathbb{C}^{U \times B_c}$ and $\mathbf{x}^T = [\mathbf{x}_1^T, \ldots, \mathbf{x}_C^T]$ with $\mathbf{x}_c \in \mathbb{C}^{B_c}$, we see that each cluster $c = 1, \ldots, C$ has to generate a precoding vector $\mathbf{x}_c$ with information of the per-cluster channel matrix $\mathbf{H}_c$, the noise variance N_0, the power constraint ρ^2, and the transmit symbol vector $\mathbf{s}$, i.e., the precoding functions are as follows:

$$\mathbf{x}_c = \mathcal{P}_c(\mathbf{s}, \mathbf{H}_c, N_0, \rho^2), \quad c = 1, \ldots, C. \quad (8)$$

Since each of these functions only depends on local CSI contained in $\mathbf{H}_c$, the exchange of CSI is reduced significantly—the vector $\mathbf{s}$ is the only signal that must be broadcast to all clusters. We now present two distinct feedforward architectures that perform decentralized precoding, differing in the amount of CSI that must be exchanged during the training phase.

B. Partially-Decentralized WF Precoding

The first feedforward architecture is illustrated in Fig. 2(a) and called *partially decentralized WF (PD-WF) architecture*. The operating principle can be derived directly from (16), which results in the precoding rule

$$\mathbf{x} = \frac{1}{\beta^{\mathrm{WF}}}\mathbf{H}^H\mathbf{A}^{-1}\mathbf{s}.$$

The idea of PD-WF precoding is to first whiten the transmit vector $\mathbf{s}$ at a centralized *whitening node* as follows:

$$\mathbf{z} = \frac{1}{\beta^{\mathrm{WF}}} \mathbf{A}^{-1} \mathbf{s}.$$

The whitened transmit vector $\mathbf{z}$ is then transmitted to each cluster, which independently compute $\mathbf{x}_c = \mathbf{H}_c^H \mathbf{z}$.

Clearly, this PD-WF architecture requires the whitening matrix $\mathbf{A}^{-1}$ as well as the precoding factor β^{WF} to be calculated at the centralized whitening node—both of these quantities require the computation of the Gram matrix $\mathbf{G}$. To compute this matrix in an decentralized architecture, we follow the approach for PD equalization put forward in [9], where each cluster $c = 1, \ldots, C$, first computes the local Gram matrix $\mathbf{G}_c = \mathbf{H}_c \mathbf{H}_c^H$ after estimating the channel in the uplink phase, followed by computing the (centralized) Gram matrix $\mathbf{G} = \sum_{c=1}^{C} \mathbf{G}_c$ in a feedforward adder tree; see the blue feedback path in Fig. 2(a). The centralized whitening node then computes the whitening matrix $\mathbf{A}^{-1}$ and the precoding factor β^{WF} as detailed in Section II-B. Since we have that

$$\sum_{c=1}^{C} \mathbf{H}_c \mathbf{x}_c = \sum_{c=1}^{C} \mathbf{H}_c \mathbf{H}_c^H \mathbf{A}^{-1} \mathbf{s} = \mathbf{G} \mathbf{A}^{-1} \mathbf{s} = \mathbf{H} \mathbf{P}^{\mathrm{WF}} \mathbf{s},$$

the PD-WF architecture implements exactly the centralized WF precoder from Theorem 1 but in a decentralized fashion.

C. Fully-Decentralized WF Precoding

The second feedforward architecture, called *fully decentralized WF (FD-WF) architecture*, is illustrated in Fig. 2(b) and avoids transmitting partial Gram matrices to the centralized whitening node. The key idea of this architecture is to first broadcast the transmit vector $\mathbf{s}$ to each cluster, and then compute the local precoding vector as follows $\mathbf{x}_c = \mathbf{P}_c \mathbf{s}$. In order to adhere to the (total) power constraint in (2), we have to define a per-cluster power constraint $\mathbb{E}\left[\|\mathbf{x}_c\|^2\right] \leq \rho_c^2$ for which $\sum_{c=1}^{C} \rho_c^2 = \rho^2$. In what follows, we allocate the same amount of power[3] to each cluster, i.e., $\rho_c^2 = \rho^2/C$, which results in the following precoder carried out at each cluster

$$\mathbf{x}_c = \sqrt{\frac{\rho_c^2}{\mathrm{tr}(\mathbf{Q}_c^H \mathbf{Q}_c) E_s}} \mathbf{Q}_c \mathbf{s}.$$

The remaining piece is to identify a suitable choice of the regularization parameters κ_c that are used to calculate the matrices $\mathbf{Q}_c$. A straightforward way would be to assume that each cluster operates independently and to set the regularization parameter to $U N_0 / \rho_c^2$. In practice, however, it turns out that this choice of the regularization parameter is overly pessimistic. Since computing an optimal set of regularization parameters is difficult in the decentralized scenario, we simply set

$$\kappa_c = \tau_c \frac{U N_0}{\rho_c^2}, \quad c = 1, \ldots, C, \tag{9}$$

---

[3]We investigated a number of strategies that allocate different power levels to each cluster. Such methods did not provide significant performance advantages in massive MU-MIMO, but may, for example, be critical for cell-free massive MU-MIMO systems in which the clusters are spread over a large area [16].

and tune the parameters $\tau_c \in [0, \infty)$ for best error-rate performance via numerical simulations. Specifically, we use

$$\mathbf{Q}_c = \begin{cases} \left(\mathbf{H}_c^H \mathbf{H}_c + \kappa_c \mathbf{I}_{B_c}\right)^{-1} \mathbf{H}_c^H & \text{if } B_c < U \\ \mathbf{H}_c^H \left(\mathbf{H}_c \mathbf{H}_c^H + \kappa_c \mathbf{I}_U\right)^{-1} & \text{if } B_c \geq U, \end{cases}$$

which further reduces the computational complexity depending on the number of antennas per cluster.

D. Simulation Results

We now show uncoded bit error-rate (BER) simulation results for a Rayleigh fading massive MU-MIMO system with 64-QAM. Figs. 3 (a), (b), (c) show the BER for $B = 256$ BS antennas, with varying cluster sizes $B_c = 128, 64, 32$, and number of clusters $C = 2, 4, 8$. Figs. 3 (d), (e), (f) show the BER for a fixed cluster size $B_c = 32$, with a varying number of BS antennas $B = 64, 128, 256$, and number of clusters $C = 2, 4, 8$. For each antenna configuration, we compare the performance of the proposed decentralized solutions PD-WF and FD-WF, as well as existing methods including centralized WF precoding, fully-distributed MRT, and the fully-decentralized ADMM-based WF precoder from [5].

Evidently, PD-WF achieves the same BER as the centralized WF precoder for all antenna configurations. In contrast, FD-WF suffers a moderate BER loss if B_c is small. To minimize the performance loss of FD-WF precoding, we have tuned the parameter τ_c in (9). Concretely, we found that $\tau_c = 0.125$ performs well for a broad range of antenna and cluster configurations; a corresponding theoretical analysis is left for future work. In addition, we see that the fully decentralized ADMM-based WF precoder proposed in [5] is able to outperform FD-WF precoding but requires multiple iterations of consensus exchange (we use two ADMM iterations) that dramatically reduces the throughput due to the typically high interconnect latency between computing fabrics; see [5] for the details.

IV. MULTI-GPU IMPLEMENTATION

As a proof-of-concept, we now present a multi-GPU implementation of PD-WF and FD-WF precoding, and show corresponding throughput and latency results.

A. System Architecture

We implemented PD-WF and FD-WF precoding on an Nvidia DGX-1 multi-GPU system [17], as illustrated in Fig. 4. The architecture consists of two 20-core Intel Xeon E5-2698 v4 CPUs and eight Tesla V100 Volta GPUs with $300\,\mathrm{GB/s}$ bi-directional NvLink GPU-to-GPU communication links. Each Tesla V100 GPU contains 5120 CUDA cores with $16\,\mathrm{GB}$ high bandwidth memory (HBM). For PD-WF and FD-WF precoding, we use the message passing interface (MPI) library OpenMPI to generate a total of C processes in the multi-GPU system, where each process controls a GPU for accelerating the decentralized local workload using CUDA [18] with CUDA v9.1. While FD-WF only requires broadcasting of transmit signals $\mathbf{s}[k]$ across GPUs prior to the precoding computations, PD-WF necessitates gathering of the local Gram matrices from all GPUs via sum reduction at the centralized whitening unit (in the master GPU

(a) $B = 256$, $B_c = 128$, $C = 2$

PD-WF: 0.879 ms, 0.979 Gb/s
FD-WF: 0.789 ms, 1.091 Gb/s

(b) $B = 256$, $B_c = 64$, $C = 4$

PD-WF: 0.678 ms / 1.270 Gb/s
FD-WF: 0.571 ms / 1.507 Gb/s

(c) $B = 256$, $B_c = 32$, $C = 8$

PD-WF: 0.607 ms / 1.418 Gb/s
FD-WF: 0.472 ms / 1.824 Gb/s

(d) $B = 64$, $B_c = 32$, $C = 2$

PD-WF: 0.532 ms / 1.618 Gb/s
FD-WF: 0.441 ms / 1.952 Gb/s

(e) $B = 128$, $B_c = 32$, $C = 4$

PD-WF: 0.559 ms / 1.540 Gb/s
FD-WF: 0.451 ms / 1.909 Gb/s

(f) $B = 256$, $B_c = 32$, $C = 8$

PD-WF: 0.607 ms / 1.418 Gb/s
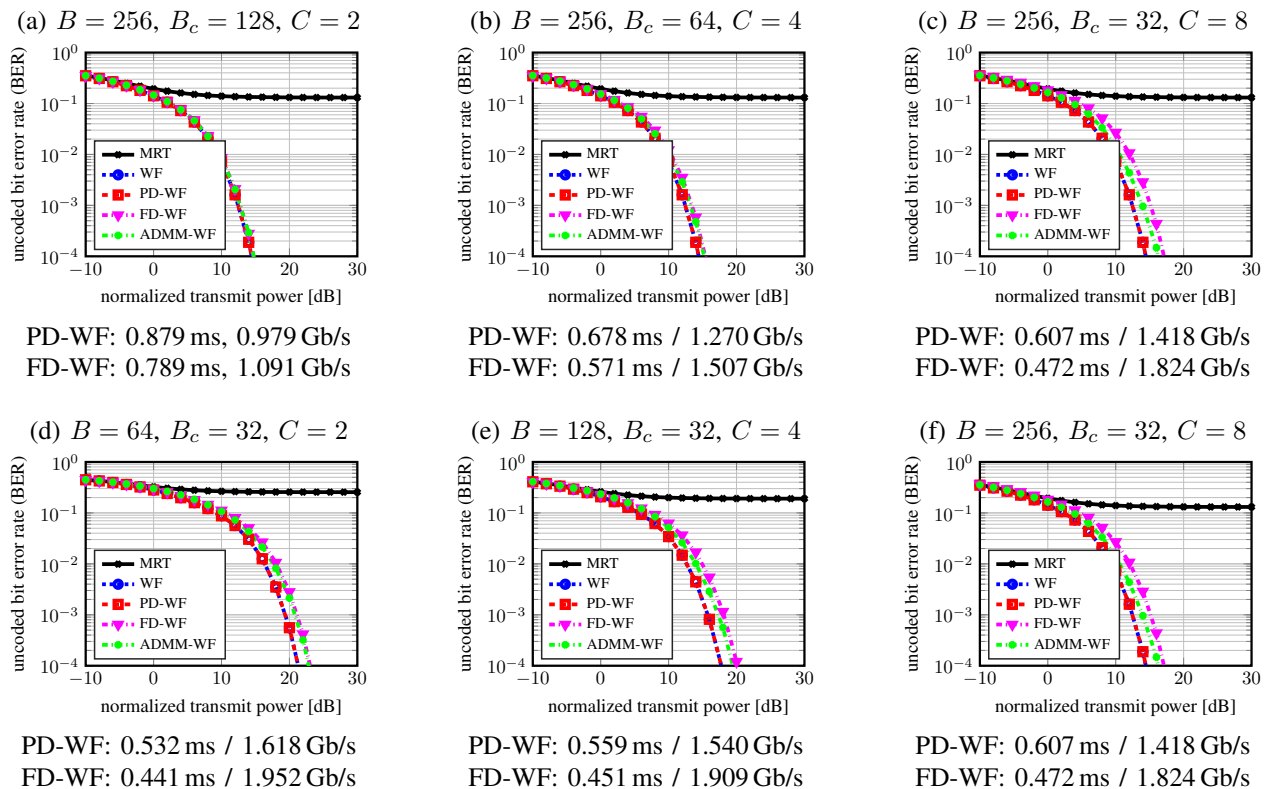FD-WF: 0.472 ms / 1.824 Gb/s

Fig. 3. Uncoded bit error-rate, latency, and throughput results for decentralized baseband processing with $U = 16$ users. Top row: fixed number of BS antennas $B = 256$, varying cluster size B_c and number of clusters C. Bottom row: fixed cluster size $B_c = 32$, varying number of BS antennas B and number of clusters C. PD-WF achieves the same error-rate performance as centralized precoding; FD-WF achieves near-WF performance for cluster sizes $B_c \geq 32$; the ADMM-based WF method outperforms FD-WF but requires iterative exchange of consensus information resulting in high latency.

as shown in Fig. 4), and broadcasting of the whitened vectors $\mathbf{z}[k]$. These message passing operations are implemented using the NVIDIA Collective Communications Library (NCCL) [19] v2.1 that builds on MPI for high efficiency over NvLink.

B. Implementation Details

To increase the throughput on the multi-GPU system, we need to feed the GPU a sufficient amount of workloads to fully exploit the available resources. In what follows, we assume the downlink transmission with orthogonal frequency division multiplexing (OFDM) with N_{sc} subcarriers. Each OFDM subcarrier corresponds to an independent narrowband block-fading downlink system as in (1), where we assume that the channel remains constant across K OFDM symbols. The vector $\mathbf{s}^w[k]$ indicates the transmit vector $\mathbf{s}[k]$ on subcarrier w in time slot k. In our implementations, we aggregate the precoding workloads for K OFDM symbols, each including N_{sc} subcarriers, and process them together in parallel to improve the GPU occupancy and throughput. In what follows, we omit the superscript ^{w} as well as the OFDM symbol index k.

1) PD-WF Implementation: For PD-WF, we invoke C MPI processes that control C GPUs, and each process initializes computation of the local Gram matrix $\mathbf{G}_c$ using the local channel $\mathbf{H}_c$ on a certain GPU. Within each GPU, we calculate N_{sc} such $\mathbf{G}_c$ matrices to achieve high throughput. These matrix multiplications can be efficiently
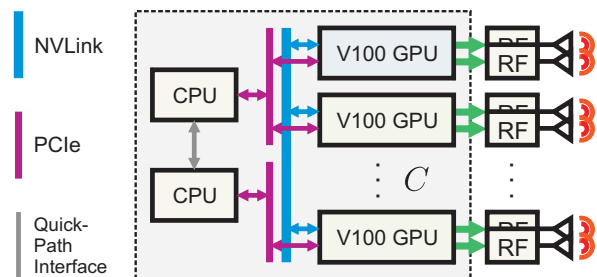

Fig. 4. Overview of the experimental platform with up to eight Tesla Volta GPUs and high speed NvLink GPU-to-GPU interconnect [17]. The uppermost GPU is the master GPU that collects results from other GPUs, performs centralized computations for PD-WF, and broadcasts the transmit vectors $\mathbf{s}[k]$ for FD-WF or the whitened vectors $\mathbf{z}[k]$ for PD-WF to other GPUs.

implemented using the *cuBLAS* [20] library; specifically, we use the `cublasCgemmBatched` function for complex-valued floating-point arithmetic. Once all local $\mathbf{G}_c$ matrices have been computed, we gather $\mathbf{G}_c$ from all C GPUs to a reduced sum (resulting in the global Gram matrix $\mathbf{G}$) at the master GPU using the NCCL `ncclReduce` function. The NCCL library leverages *CUDA-aware MPI* [21] for direct GPU-to-GPU memory copy over high-speed NvLink interconnect.

We compute $\mathbf{A} = \mathbf{G} + \kappa^{WF}\mathbf{I}_U$ for N_{sc} subcarriers (corresponding to a given OFDM symbol k) at the mas-

ter GPU in parallel. We then invert this matrix using the *cuBLAS* `cublasCgetrfBatched` Cholesky decomposition, followed by `cublasCgetriBatched` that implements forward and backward substitution operations to obtain $\mathbf{A}^{-1}$. Since the local channel matrix $\mathbf{H}_c$ is assumed to remain constant for K OFDM symbols, we store $\mathbf{A}^{-1}$ for a given OFDM symbol k, and reuse this matrix for all K OFDM symbols. To compute the whitened vector $\mathbf{z} = \frac{1}{\beta^{\text{WF}}}\mathbf{A}^{-1}\mathbf{s}$, we first multiply the transmit vector $\mathbf{s}$ with the matrix $\mathbf{A}^{-1}$ using the `cublasCgemmBatched` function for a total of $N_{\text{sc}} \times K$ subcarriers. We then calculate the precoding factor β^{WF}. As shown in (6), β^{WF} depends on $\text{tr}(\mathbf{A}^{-1})$ and $\|\mathbf{A}^{-1}\|_F^2$, which involve sum reduction operations across the diagonal entries or all entries of matrix $\mathbf{A}^{-1}$. To resolve such data dependencies efficiently, we design a customized kernel function to calculate β^{WF}, where we take advantage of fast local registers and shared memories for inter-thread communications. Specifically, we invoke this kernel with N_{sc} thread-blocks to calculate N_{sc} different β^{WF} values in parallel. In each thread-block, we generate $U \times U$ threads to access each entry of the $U \times U$ matrix $\mathbf{A}^{-1}$, and perform inter-thread shuffle of local register values within a *warp* using *warp shuffle* [22], and inter-thread communication across different *warps* within this thread-block using shared memory, to realize the sum reductions required to compute $\text{tr}(\mathbf{A}^{-1})$ and $\|\mathbf{A}^{-1}\|_F^2$. Analogously to the computations for $\mathbf{A}^{-1}$, we can reuse the parameter β^{WF} across K OFDM symbols, and compute the whitened vector $\mathbf{z}$. For PD-WF, whitening happens at the master GPU in a centralized manner, and therefore we need to broadcast the whitened vector $\mathbf{z}$ to all GPUs using NCCL `ncclBcast`. We finally compute the local precoding vector $\mathbf{x}_c = \mathbf{H}_c^H \mathbf{z}$ by `cublasCgemmBatched` function for all $N_{\text{sc}} \times K$ subcarriers at each GPU in a decentralized fashion.

2) FD-WF Implementation: For FD-WF, we use *cuBLAS* and customized kernels as for PD-WF in order to implement the local WF precoder corresponding to $\mathbf{B}_c$ BS antennas with the power constraint $\rho_c^2 = \frac{\rho^2}{C}$. To invoke the FD-WF precoder, we broadcast the transmit vectors $\mathbf{s}$ to the C MPI processes, each running a local WF precoder on a separate GPU to compute the local precoding vectors $\mathbf{x}_c$ in parallel.

C. Implementation Results

Fig. 3 shows the latency and throughput results of PD-WF and FD-WF measured on the multi-GPU system for various BS antenna configurations and $U = 16$ UEs. For all configurations, we set $N_{\text{sc}} = 1200$, $K = 7$ corresponding to a slot-frame of 20 MHz LTE signal with OFDM and 64-QAM transmission.

In the top row of Fig. 3 , we fix the number of BS antennas to $B = 256$, and increase the number of clusters $C = 2, 4, 8$ (and, hence, the number of used GPUs) while decreasing the cluster size $B_c = 128, 64, 32$. By decreasing B_c, the throughput of PD-WF and FD-WF precoding increases as less local workload is processed in parallel. FD-WF achieves higher data rate than PD-WF, since FD-WF only requires to broadcast the transmit vector $\mathbf{s}$ which scales with $N_{\text{sc}} \times K \times U$, while PD-WF requires a higher amount of message passing, which includes (i) gathering

of local Gram matrices $\mathbf{G}_c$ (scaling with $N_{\text{sc}} \times U \times U$) and (ii) broadcasting of whitened vector $\mathbf{z}$ (scaling with $N_{\text{sc}} \times K \times U$).

In the bottom row of Fig. 3, we fix the number of antennas per cluster to $B_c = 32$, and increase $B = 64, 128, 256$ by scaling the number of clusters $C = 2, 4, 8$. We observe that the throughput only degrades slightly with increasing B and C for both PD-WF and FD-WF, indicating that the message-passing latency remains nearly constant; this is a result of the direct GPU-to-GPU gathering/broadcasting communications realized by NCCL. This observation also implies that we can increase the number of BS antennas with only a small loss in throughput using the proposed decentralized feedforward architecture. For all configurations show in Fig. 3, our designs achieve throughputs in the Gb/s regime with latencies below 1 ms. We also see that FD-WF outperforms PD-WF in terms of throughput due to the reduced amount of message passing, while PD-WF achieves superior error-rate performance.

V. Conclusions

We have proposed two novel feedforward architectures and corresponding decentralized precoding algorithms based on the linear Wiener filter (WF) precoder. We have demonstrated that the partially-decentralized (PD) WF precoder achieves the error-rate performance of the centralized WF precoder, while significantly reducing the interconnect and chip input/output bandwidths. To further reduce the interconnect bandwidth, we have proposed a fully-decentralized (FD) WF precoder that incurs only a small error-rate performance loss compared to the PD-WF precoder. To showcase the efficiency and scalability of PD-WF and FD-WF to large antenna arrays, we have presented implementations on a multi-GPU system. Our results demonstrate that throughputs in the Gb/s regime at latencies below 1 ms are achievable. These results indicate that the proposed decentralized precoding are a solution to combat the interconnect and complexity bottlenecks while being able to fully exploit the spectral efficiency and link reliability advantages provided by massive MU-MIMO systems.

There are many avenues for future work. A theoretical analysis of the optimal regularization parameter τ_c for FD-WF precoding in (9) is an open research question. Combining decentralized feedforward precoding with data detection as in [14] may further reduce the processing latency and increase the throughput as a large number of quantities can be re-used between the uplink and downlink. The development and analysis of feedforward architectures for cell-free massive MU-MIMO as put forward in [16] is part of ongoing work.

APPENDIX A
PROOF OF THEOREM 1

The precoder resulting from Section 3 is known as the Wiener filter (WF) precoder [9] and can be derived as follows. Let us first form the Lagrangian

$$\mathcal{L}(\mathbf{P}, \beta, \lambda) = \mathbb{E}_{\mathbf{s}}\big[\|\mathbf{s} - \beta\mathbf{H}\mathbf{P}\mathbf{s}\|^2\big] + \beta^2 U N_0$$
$$+ \lambda(\mathbb{E}_{\mathbf{s}}\big[\|\mathbf{x}\|^2\big] - \rho^2).$$

We can now formulate the optimality conditions for $\mathbf{P}$ and β by using the Wirtinger derivative as follows. For the precoding matrix $\mathbf{P}$, we have the following optimality condition:

$$\frac{\delta}{\delta \mathbf{P}^H} \mathcal{L}(\mathbf{P}, \beta, \lambda) = \mathbf{0} \implies \beta^2 \mathbf{H}^H \mathbf{H} \mathbf{P} + \lambda \mathbf{P} = \beta \mathbf{H}^H. \quad (10)$$

For the precoding factor β, we compute $\frac{\delta}{\delta \beta^*} \mathcal{L}(\mathcal{P}, \beta, \lambda) = 0$ and obtain the following optimality condition:

$$\beta \operatorname{tr}(\mathbf{P}^H \mathbf{H}^H \mathbf{H} \mathbf{P}) + \beta \frac{U N_0}{E_s} = \operatorname{tr}(\mathbf{H}^H \mathbf{P}^H). \quad (11)$$

From the power constraint, it follows that

$$\mathbb{E}_{\mathbf{s}}\left[\|\mathbf{x}\|^2\right] = \rho^2 \implies \operatorname{tr}(\mathbf{P}^H \mathbf{P}) = \frac{\rho^2}{E_s}. \quad (12)$$

To derive the optimal value for the Lagrange multiplier λ, we apply the following steps the optimality condition in (10):

$$\beta^2 \mathbf{H}^H \mathbf{H} \mathbf{P} + \lambda \mathbf{P} = \beta \mathbf{H}^H$$
$$\beta^2 \mathbf{H}^H \mathbf{H} \mathbf{P} \mathbf{P}^H + \lambda \mathbf{P} \mathbf{P}^H = \beta \mathbf{H}^H \mathbf{P}^H$$
$$\beta^2 \operatorname{tr}(\mathbf{H}^H \mathbf{H} \mathbf{P} \mathbf{P}^H) + \lambda \operatorname{tr}(\mathbf{P} \mathbf{P}^H) = \beta \operatorname{tr}(\mathbf{H}^H \mathbf{P}^H)$$
$$\beta^2 \operatorname{tr}(\mathbf{P}^H \mathbf{H}^H \mathbf{H} \mathbf{P}) + \lambda \frac{\rho^2}{E_s} = \beta \operatorname{tr}(\mathbf{H}^H \mathbf{P}^H), \quad (13)$$

where the last step results from (12). We now multiply both sides of the optimality condition in (11) with β to obtain

$$\beta \operatorname{tr}(\mathbf{P}^H \mathbf{H}^H \mathbf{H} \mathbf{P}) E_s + \beta U N_0 = \operatorname{tr}(\mathbf{H}^H \mathbf{P}^H) E_s$$
$$\beta^2 \operatorname{tr}(\mathbf{P}^H \mathbf{H}^H \mathbf{H} \mathbf{P}) + \beta^2 \frac{U N_0}{E_s} = \beta \operatorname{tr}(\mathbf{H}^H \mathbf{P}^H). \quad (14)$$

Subtracting (14) from (13) yields the Lagrange multiplier

$$\lambda = \frac{U N_0}{\rho^2}. \quad (15)$$

From (10) and (15), it follows that the WF precoding matrix is given by $\mathbf{P}^{\mathrm{WF}} = \frac{1}{\beta^{\mathrm{WF}}} \mathbf{Q}$ with the matrix

$$\mathbf{Q} = \left(\mathbf{H}^H \mathbf{H} + \frac{U N_0}{\rho^2} \mathbf{I}\right)^{-1} \mathbf{H}^H.$$

The remaining piece is to identify the WF precoding factor β^{WF}. To this end, we plug $\mathbf{P}^{\mathrm{WF}}$ into (12), which leads to

$$\frac{1}{\beta^2} \operatorname{tr}(\mathbf{Q}^H \mathbf{Q}) = \frac{\rho^2}{E_s} \implies \frac{1}{\beta^{\mathrm{WF}}} = \frac{\rho}{\sqrt{\operatorname{tr}(\mathbf{Q}^H \mathbf{Q}) E_s}}.$$

APPENDIX B
PROOF OF LEMMA 2

To reduce the complexity of computing β^{WF} in (5), we first use the matrix inversion lemma [23] to arrive at an equivalent expression of (4) given by

$$\mathbf{Q}^{\mathrm{WF}} = \mathbf{H}^H \left(\mathbf{H} \mathbf{H}^H + \kappa^{\mathrm{WF}} \mathbf{I}_U\right)^{-1},$$

which requires the inversion of an $U \times U$ matrix. By precomputing the $U \times U$ Gram matrix $\mathbf{G} = \mathbf{H} \mathbf{H}^H$ and inverting the regularized Gram matrix defined as $\mathbf{A} = \mathbf{G} + \kappa^{\mathrm{WF}} \mathbf{I}_U$, we have

$$\mathbf{Q}^{\mathrm{WF}} = \mathbf{H}^H \mathbf{A}^{-1} \quad (16)$$

and consequently, $\operatorname{tr}(\mathbf{Q}^H \mathbf{Q}) = \operatorname{tr}\left(\mathbf{A}^{-1} \mathbf{G} \mathbf{A}^{-1}\right)$. A direct evaluation of this expression still requires two matrix-matrix multiplications of dimension $U \times U$. We can further reduce complexity by noting that the following equivalence holds

$$\operatorname{tr}\left(\mathbf{A}^{-1} \mathbf{G} \mathbf{A}^{-1}\right) = \operatorname{tr}\left(\mathbf{A}^{-1}\right) - \kappa^{\mathrm{WF}} \|\mathbf{A}^{-1}\|_F^2,$$

where we used a Searle-type identity [24] and a matrix version of partial fraction expansion to finally arrive at (6).

REFERENCES

[1] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What Will 5G Be?," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[2] 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 Commun.*, vol. 31, no. 2, pp. 160–171, Feb. 2013.

[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] A. Puglielli, N. Narevsky, P. Lu, T. Courtade, G. Wright, B. Nikolic, and E. Alon, "A scalable massive MIMO array architecture based on common modules," in *IEEE Intl. Conf. Commun. Workshop (ICCW)*. IEEE, June 2015, pp. 1310–1315.

[5] K. Li, R. R. Sharan, Y. Chen, T. Goldstein, J. R. Cavallaro, and C. Studer, "Decentralized Baseband Processing for Massive MU-MIMO Systems," *IEEE J. Emerg. and Sel. Topics in Circ. and Sys.*, vol. 7, no. 4, pp. 491–507, Dec 2017.

[6] S. Jacobsson, G. Durisi, M. Coldrey, T. Goldstein, and C. Studer, "Quantized Precoding for Massive MU-MIMO," *IEEE Trans. on Comm.*, vol. 65, no. 11, pp. 4670–4684, Nov 2017.

[7] E. G. Larsson E. Bertilsson, O. Gustafsson, "A distributed processing architecture for modular and scalable massive MIMO base stations," *arXiv preprint: 1801.07967*, Jan. 2018.

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

[9] M. Joham, W. Utschick, and J. A. Nossek, "Linear transmit processing in MIMO communications systems," *IEEE Trans on Sig. Proc.*, vol. 53, no. 8, pp. 2700–2712, Aug 2005.

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

[11] 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 *IEEE Globecom*, Dec. 2014, pp. 287–293.

[12] K. Li, Y. Chen, R. Sharan, T. Goldstein, J. R. Cavallaro, and C. Studer, "Decentralized data detection for massive MU-MIMO on a Xeon Phi cluster," in *Asilomar Conf. Sig. Sys. Comp.*, Nov. 2016, pp. 468–472.

[13] K. Li, R. Skaran, Y. Chen, J. R. Cavallaro, and C. Studer, "Decentralized beamforming for massive MU-MIMO on a GPU cluster," in *IEEE GlobalSIP*, Dec. 2016, pp. 590–594.

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

[15] N. Fatema, G. Hua, Y. Xiang, D. Peng, and I. Natgunanathan, "Massive MIMO Linear Precoding: A Survey," *IEEE Sys. Journal*, vol. 12, no. 4, pp. 3920–3931, 2018.

[16] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta, "Cell-free massive MIMO versus small cells," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1834–1850, Jan. 2017.

[17] https://www.nvidia.com/en-us/data-center/dgx-1, *Nvidia DGX-1 system*.

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

[19] https://developer.nvidia.com/nccl, *NVIDIA Collective Communications Library (NCCL)*.

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

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

[22] https://devblogs.nvidia.com/faster-parallel-reductions-kepler/, *Parallel reduction using warp shuffle*.

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

[24] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*, Technical University of Denmark, Nov. 2012.