Decentralized Data Detection for Massive MU-MIMO on a Xeon Phi Cluster

Kaipeng Li[1], Yujun Chen[1], Rishi Sharan[2], Tom Goldstein[3], 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
[3]Department of Computer Science, University of Maryland, College Park, MD

***Abstract*—Conventional *centralized* data detection algorithms for massive multi-user multiple-input multiple-output (MU-MIMO) systems, such as minimum mean square error (MMSE) equalization, result in excessively high raw baseband data rates and computing complexity at the centralized processing unit. Hence, practical base-station (BS) designs for massive MU-MIMO that rely on state-of-the-art hardware processors and I/O interconnect standards must find new means to avoid these bottlenecks. In this paper, we propose a novel *decentralized* data detection method, which partitions the BS antenna array into separate clusters. Each cluster is associated with independent computing hardware to perform decentralized data detection, which requires only local channel state information and receive data, and a minimum amount of information exchange between clusters. To demonstrate the benefits of our approach, we map our algorithm to a Xeon Phi cluster, which shows that BS designs with hundreds or thousands of BS antennas can be supported.**

I. Introduction

Massive multi-user multiple-input multiple-output (MU-MIMO) is a key technology in the upcoming fifth-generation (5G) wireless standards [1]. By equipping the basestation (BS) with hundreds or thousands of antenna elements to serve tens of users simultaneously and in the same frequency band, massive MU-MIMO is able to achieve higher spectral efficiency and link reliability compared to traditional, small-scale MIMO systems [2]. In the massive MU-MIMO uplink, in which user terminals simultaneously transmit data to the BS, data detection at the BS is a central task that untangles the received data streams from multiple simultaneously-transmitting users with the aid of channel state information (CSI).

A. Challenges of Centralized Data Detection

While massive MU-MIMO promises a variety of advantages over traditional, small-scale MIMO, the large amount of BS antenna elements pose several critical challenges for the system architecture and design of practical BS implementations. Specifically, most existing data detection algorithms, such as zero-forcing (ZF) or minimum mean square error (MMSE) equalization, rely on *centralized* baseband processing in order to realize the full benefits of massive MIMO. Such schemes require aggregation of high-rate raw baseband data streams sampled from hundreds or thousands of radio-frequency (RF) chains at the BS, which must be transmitted and processed at a centralized processing node. Consider, for example, a 128 BS-antenna massive MU-MIMO system with 10-bit analog-to-digital (ADC) converters running at 40 MHz bandwidth. Such a system generates over 200 Gb/s raw baseband data. The bandwidth limitations of existing high-speed interconnects, such as the common public radio interface (CPRI) [3], as well as the input-output (I/O) bandwidth and on-chip storage limits of conventional baseband accelerators, such as field-programmable gate arrays (FPGAs) or graphics processing units (GPUs) [4], however, prevent the use of such naïve centralized solutions. In fact, existing massive MU-MIMO testbeds, such as the Argos testbed [5], the LuMaMi testbed [6], and the BigStation [7], have shown that centralized baseband processing is infeasible with current I/O bandwidths and hardware processing capabilities. Therefore, these testbeds either perform maximum ratio combining (MRC) [5], which naturally enables distributed data detection at the antenna elements, or distribute baseband processing across subcarriers [6], [7]. Unfortunately, MRC prevents the use of high-rate modulation and coding schemes that achieve throughputs in the Gb/s regime. Distributing baseband processing across subcarriers requires high-throughput data transfer from *all* RF components to *all* baseband processors (but only for a subset of subcarriers), which still causes extremely high I/O bandwidth for systems with thousands of BS antenna elements. As a consequence, alternative BS architectures are required that mitigate the I/O bandwidth and baseband processing bottlenecks.

B. Contributions

In this paper, we propose a novel decentralized data detection architecture and a suitable algorithm for massive MU-MIMO systems. We divide the BS antenna array into separate antenna clusters, each connected to local RF elements and providing storage for local CSI. Data detection is then carried out in a decentralized manner with minimum information exchange between the antenna clusters. Our data detection algorithm relies on a conjugate gradient (CG) method that performs inversion-free MMSE equalization in a decentralized manner. To demonstrate the practical relevance of our solution, we present reference implementation results on a Xeon Phi cluster,

This work was supported in part by Xilinx Inc., the US National Science Foundation under grants CNS-1265332, ECCS-1232274, ECCS-1408370, ECCS-1408006, CCF-1535902, and the Texas Advanced Computing Center.

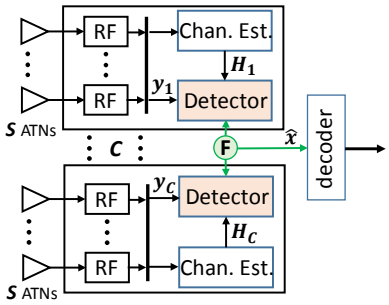


Fig. 1. Overview of the proposed decentralized data detection architecture.

in which each computing node is equipped with a state-of-the-art many-core Intel Xeon Phi *Knights Landing* processor.

II. Decentralized Data Detection

A. Uplink System Model

We consider a massive MU-MIMO uplink system that uses orthogonal frequency division multiplexing (OFDM). The system consists of U single-antenna users transmitting data contained in the vector $\mathbf{x}_w \in \mathbb{C}^U$ on each OFDM subcarrier $w = 1, 2, \ldots, W$ over the uplink channel that is modeled by the matrix $\mathbf{H}_w \in \mathbb{C}^{B \times U}$, to a BS that is equipped with $B \geq U$ antenna elements. The input-output relation of the uplink channel is modeled as $\mathbf{y}_w = \mathbf{H}_w \mathbf{x}_w + \mathbf{n}_w$ per subcarrier w. Here, $\mathbf{y}_w \in \mathbb{C}^B$ is the received signal at the BS antenna array and $\mathbf{n}_w \in \mathbb{C}^B$ is i.i.d Gaussian noise with variance N_0 per complex entry. For each subcarrier w, the BS performs data detection using the received signal $\mathbf{y}_w$ and the estimated channel matrix $\mathbf{H}_w$ to demultiplex the transmitted data from all users. In what follows, we omit the subcarrier index w.

B. Decentralized Data Detection Architecture

In order to arrive at computationally efficient data detection algorithms that can be decentralized, we focus on linear MMSE equalization. This approach consists of an equalization and a detection stage. For the MMSE equalization stage, we are interested in solving the following optimization problem:

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{C}^U}{\arg\min} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \tfrac{N_0}{E_s}\|\mathbf{x}\|_2^2, \qquad (1)$$

where $E_s = \mathbb{E}[|x_i|^2]$ is the average per-user transmit energy for user i.[1] The solution to (1) is the well-known linear MMSE estimate, which is used to compute approximate soft-output values, e.g., in the form of log-likelihood ratio (LLR) values [8].

To solve (1) in a decentralized manner, we propose a decentralized detection architecture as shown in Fig. 1. We evenly partition the BS antenna array into C antenna clusters, where each cluster is associated with S antenna elements so that $B = CS$. Each cluster performs decentralized data detection using only local[2] CSI and received data. Each cluster only observes part of the received vector that pertains to cluster c, i.e., $\mathbf{y}_c$, $c = 1, 2, \ldots, C$, where $\mathbf{y}^T = [\mathbf{y}_1^T, \ldots, \mathbf{y}_C^T]^T$. Analogously, the uplink channel matrix $\mathbf{H}$ is partitioned row-wise into blocks as $\mathbf{H}^T = [\mathbf{H}_1^T, \ldots, \mathbf{H}_C^T]$ with $\mathbf{H}_c \in \mathbb{C}^{S \times U}$, $c = 1, 2, \ldots, C$. The ultimate goal of decentralized data detection is to solve (1) where each cluster only accesses local CSI and received data with minimum information exchange among the clusters.

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

[2]Each cluster performs independent channel estimation in the uplink. The locally-acquired CSI will not be distributed to the other antenna clusters.

Algorithm 1 Decentralized CG-based MMSE Data Detection

Input: $\mathbf{H}_c$, $c = 1, \ldots, C$, and $\mathbf{y}_c$, and ρ
Preprocessing:
  $\mathbf{y}_c^{\text{MF}} = \mathbf{H}_c^H \mathbf{y}_c$ /* Decentralized */
  $\mathbf{y}^{\text{MF}} = \sum_{c=1}^{C} \mathbf{y}_c^{\text{MF}}$ /* Centralized */
CG iterations:
Init: $\mathbf{r}^{(0)} = \mathbf{y}^{\text{MF}}, \mathbf{p}^{(0)} = \mathbf{r}^{(0)}, \mathbf{s}^{(0)} = \mathbf{0}$
for $t = 1, \ldots, T$ **do**
  Decentralized (each cluster c performs the same operations):
    $\mathbf{w}_c^{(t)} = \mathbf{H}_c^H \mathbf{H}_c \mathbf{p}^{(t-1)}$
  Centralized (consensus on centralized processing unit):
    $\mathbf{w}^{(t)} = \sum_{c=1}^{c} \mathbf{w}_c^{(t)}$ /* Consensus */
  Decentralized (each cluster c performs the same operations):
    $\mathbf{e}^{(t)} = \frac{N_0}{E_s}\mathbf{p}^{(t-1)} + \mathbf{w}^{(t)}$
    $\alpha = \|\mathbf{r}^{(t-1)}\|^2 / ((\mathbf{p}^{(t-1)})^H \mathbf{e}^{(t)})$
    $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)} + \alpha \mathbf{p}^{(t-1)}$
    $\mathbf{r}^{(t)} = \mathbf{r}^{(t-1)} - \alpha \mathbf{e}^{(t-1)}$
    $\beta = \|\mathbf{r}^{(t)}\|^2 / \|\mathbf{r}^{(t-1)}\|^2$
    $\mathbf{p}^{(t)} = \mathbf{r}^{(t)} + \beta \mathbf{p}^{(t-1)}$
end for
Output: $\hat{\mathbf{x}} = \mathbf{x}^{(T)}$

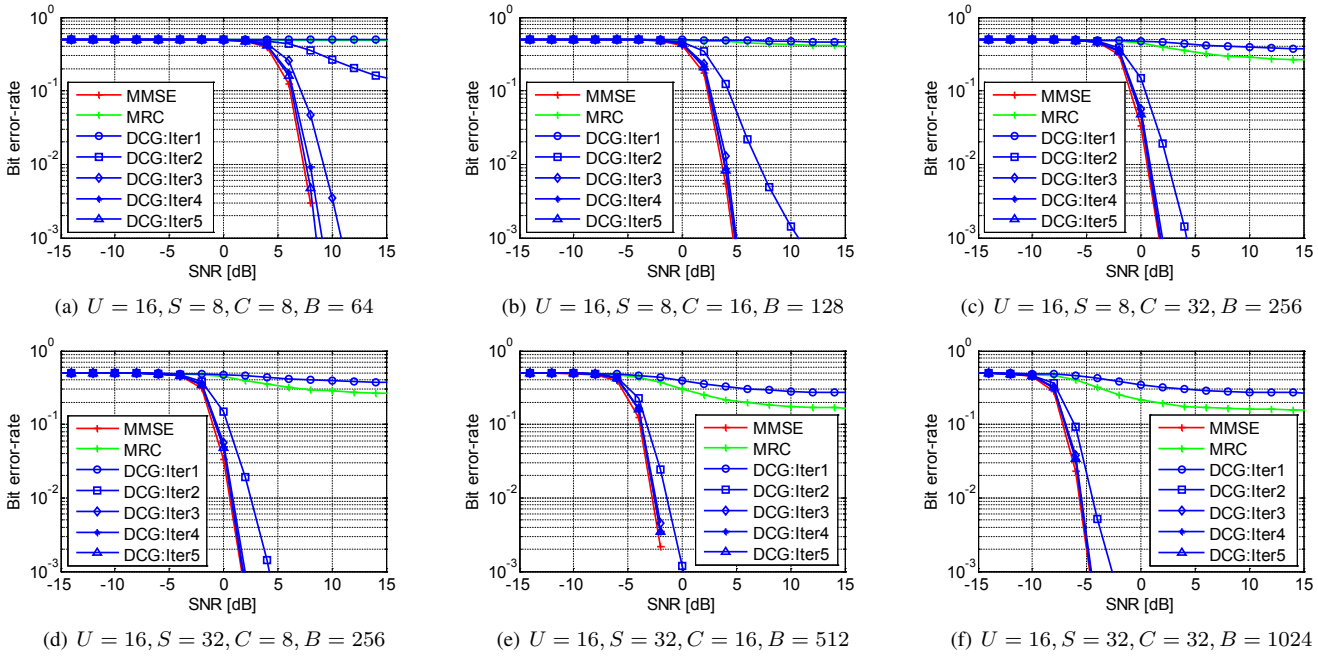
C. Decentralized Equalization using Conjugate Gradient

Since the MMSE equalization problem in (1) is quadratic in the vector $\mathbf{x}$, we can solve it using the conjugate gradient (CG) method in a decentralized manner. Reference [9] proposed a *centralized* equalization-based detection method using CG to approximate the equalized signal $\hat{\mathbf{x}}$ iteratively without the need of an explicit matrix inversion. Our decentralized CG-based detection builds on this centralized algorithm. The key of the proposed decentralization approach 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 (i.e., $\mathbf{H}_c$ and $\mathbf{y}_c$). The centralized CG-based detector in [9] involves two stages: a preprocessing stage for calculating the matched filter output $\mathbf{y}^{\text{MF}}$ and a CG iteration stage to estimate $\hat{\mathbf{x}}$ using $\mathbf{y}^{\text{MF}}$ as initialization.

For the preprocessing stage, it is important to realize that one can rewrite the matched filter (MF) $\mathbf{y}^{\text{MF}} = \mathbf{H}^H \mathbf{y}$ as $\mathbf{y}^{\text{MF}} = \sum_{c=1}^{C} \mathbf{H}_c^H \mathbf{y}_c$, 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 MF output $\mathbf{y}^{\text{MF}}$.

For the CG iteration stage, we need to update the estimation of transmit vector $\mathbf{x}$, as well as some intermediate vectors $\mathbf{e}$, $\mathbf{r}$, and $\mathbf{p}$ required by the CG algorithm in the order of $\mathbf{e}$, $\mathbf{x}$, $\mathbf{r}$, and $\mathbf{p}$ in each iteration (see [9] for the details). In contrast to the update procedure of $\mathbf{x}$, $\mathbf{r}$, and $\mathbf{p}$, which only involves vector operations that are not dependent on global CSI $\mathbf{H}$, the update procedure of $\mathbf{e}$ requires global CSI for each iteration t:

$$\mathbf{e}^{(t)} = \left(\tfrac{N_0}{E_s}\mathbf{I} + \mathbf{H}^H\mathbf{H}\right)\mathbf{p}^{(t-1)}, \qquad (2)$$



(a) $U = 16, S = 8, C = 8, B = 64$ (b) $U = 16, S = 8, C = 16, B = 128$ (c) $U = 16, S = 8, C = 32, B = 256$

(d) $U = 16, S = 32, C = 8, B = 256$ (e) $U = 16, S = 32, C = 16, B = 512$ (f) $U = 16, S = 32, C = 32, B = 1024$

Fig. 2. Bit error rate (BER) performance comparison between decentralized conjugate gradient (DCG) and centralized MMSE detection for different system configurations: U denotes the number of users, S the number of antennas per cluster, C the number of antenna clusters, and B the number of BS antennas.

i.e., requires access to the full channel matrix $\mathbf{H}$ and thus, must be decentralized. It is key to realize that the Gram matrix can be computed by $\mathbf{H}^H\mathbf{H} = \sum_{c=1}^{C} \mathbf{H}_c^H\mathbf{H}_c$, i.e., by summing results of local CSI $\mathbf{H}_c$. Hence, we can reformulate (2) as

$$\mathbf{e}^{(t)} = \tfrac{N_0}{E_s}\mathbf{p}^{(t-1)} + \textstyle\sum_{c=1}^{C} \mathbf{H}_c^H\mathbf{H}_c\mathbf{p}^{(t-1)}. \qquad (3)$$

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 (3) by performing the following centralized computations that do not require global CSI: $\mathbf{w}^{(t)} = \sum_{c=1}^{C} \mathbf{w}_c^{(t)}$ and $\mathbf{e}^{(t)} = \frac{N_0}{E_s}\mathbf{p}^{(t-1)} + \mathbf{w}^{(t)}$.

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 thus, can be carried out at the centralized processing unit. We must, however, broadcast the centralized 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 the local processing units, leaving the calculation of consensus $\mathbf{w}$ as the only centralized computation in a CG iteration; this enables the concatenation of data gathering and broadcasting, which for example, can be implemented using a single message-passing function call as discussed in Section III-B. Therefore, we use this decentralization scheme and the resulting decentralized CG-based MMSE detection algorithm is summarized in Algorithm 1. We emphasize that the centralized computations of $\mathbf{y}^{\text{MF}}$ at line 4 and consensus $\mathbf{w}$ at line 11 involve only little information exchange for data gathering from and broadcasting to every antenna cluster, since $\mathbf{y}^{\text{MF}}$ or $\mathbf{w}$ are U-dimensional vectors.

D. Simulations and Trade-offs

In this part, we show the error-rate performance of our decentralized data detection algorithm using system-level simulations, and evaluate its computational complexity. We also study the associated performance/complexity trade-off.

We simulate our decentralized CG-based (DCG) data detector in an LTE-based massive MU-MIMO-OFDM uplink system with 2048-subcarrier 20 MHz wideband signals, 16-QAM modulation, a 5/6 rate convolutional encoding, and max-log soft-output BCJR channel decoding. The channel matrices are generated using the Winner-II model to reflect real-world wireless environments. We also consider channel estimation errors using a standard pilot-based ML estimation scheme.

Fig. 2 shows the bit error rate (BER) performance against average SNR per receive antenna for our proposed DCG data detector in the massive MU-MIMO uplink system. We perform extensive simulations under various parameter configurations to verify the effectiveness of our proposed scheme. Specifically, we set the user number $U = 16$ and individual cluster antenna number $S = 8$ or $S = 32$, and scale the BS antenna number $B = CS$ from 64 to 1024 by varying the cluster number $C = 8$, $C = 16$ or $C = 32$. From our simulation results, we see that for small numbers of BS antenna elements, such as $B = 64$ or $B = 128$, three CG iterations are sufficient to approach the BER performance of a centralized MMSE detector. With more BS antennas, for example, $B \geq 256$, our approach achieves improved BER performance and only two

TABLE I
COMPUTATIONAL COMPLEXITY.

DCG	Preprocessing	$4CSU + 2U$
	1st iteration	$C(8SU + 4U) + 2U$
	$2 - T$ iters.	$(T-1)(C(8SU + 10U) + 2U)$
MMSE		$6CSU^2 + \frac{10}{3}U^3 + 4CSU - \frac{1}{3}U$

iterations are sufficient to approach near MMSE performance. We also emphasize that the BER performance of our DCG detector does not depend on the number of antenna clusters C for a given antenna configuration B and U. For example, in Fig. 2(c) and Fig. 2(d), we obtain the same BER performance for either $S = 8$, $C = 32$, or $S = 32$, $C = 8$ given the total BS antenna number $B = 256$. The reason is that equation (3) is an equivalent decentralized transformation of (2) that does not involve any approximation. Hence, the BER results in Fig. 2 also apply to fully-decentralized antenna systems, i.e., with $C = B$ and $S = 1$, which outperforms the fully-distributed MRC detector for more than one CG iteration.

In Table I, we show the number of real-valued multiplications that dominate the computational complexity for both our DCG detector and a centralized MMSE detector. For the decentralized detector, we count the total number of real-valued multiplications from *all* decentralized parts.

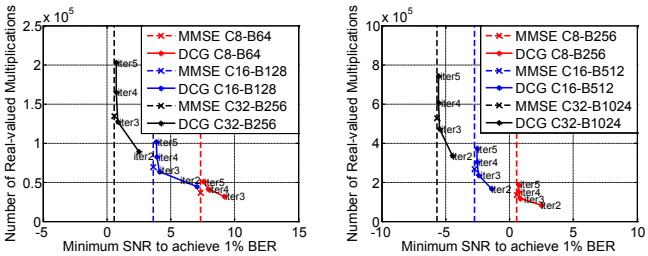
Fig. 3 illustrates the trade-off between error-rate performance and complexity, where we extract the minimum SNR to achieve 1% BER from Fig. 2 and calculate corresponding complexity numbers for each CG iteration of a given antenna configuration according to the results in Table I. With fewer than three iterations, which are typically enough for achieving near-MMSE performance, our DCG scheme also reduces the computational complexity. We conclude by noting that the low complexity, high scalability, and low error-rate of our decentralized method is advantageous in practical massive MU-MIMO systems compared to centralized solutions.

III. IMPLEMENTATION ON A XEON PHI CLUSTER

We now describe the implementation details of our DCG algorithm on a Xeon Phi cluster. To achieve high computing efficiency, we explore the parallelism of DCG for our implementation on the Phi cluster using parallel programming techniques and realize efficient data communication, such as data gathering and broadcasting for consensus calculation. We start by introducing the Intel Xeon Phi processor and then show our design mapping strategies.

A. Introduction to the Intel Xeon Phi Processor

The Intel Xeon Phi *Knights Landing* (KNL) processor [10], the second-generation Phi following previous *Knights Corner* (KNC) coprocessor, is a shared-memory computer containing tens of x86 cores organized with Intel many integrated core (MIC) architecture on a single chip. Each Phi core supports simultaneous multithreading with four hardware threads, and is equipped with a vector processing unit (VPU) and vector registers to enable 512-bit wide single instruction multiple data



(a) $U = 16, S = 8$. (b) $U = 16, S = 32$.

Fig. 3. Performance/complexity trade-off.

(SIMD) operations besides scalar processing. Every two cores with two VPUs and 1 MB L2 cache construct a *tile*, and all tiles are interconnected in a 2D mesh on the chip with a total of over 30 MB L2 cache. A high-speed MCDRAM is installed around the 2D mesh on Phi package for fast memory access besides the use of conventional off-chip DDR4 memory. The KNL Phi processor can run a self-boot operating system without the need of a host Xeon CPU, which leads to more efficient workload deployment within the chip. A Phi cluster can be constructed by connecting multiple such KNL Phi processor nodes via Intel high-bandwidth Omni-Path network interconnection, in which we will realize our proposed decentralized algorithm.

B. Design Mapping on a Xeon Phi Cluster

We start by analyzing the inherent parallelism of our proposed data detection algorithm. As discussed before, data detection in OFDM systems can be performed independently for each subcarrier. Therefore, some previous work [6], [7] has addressed decentralized processing for downlink beamforming or uplink detection across subcarriers, i.e., the processing workloads for different subset of subcarriers are deployed on different independent processors; this approach, however, still requires high-throughput data transfer from all antennas to all processors. Our proposed decentralized architecture introduces an additional level of parallelism across antenna clusters in the spatial dimension (beyond subcarriers in the frequency dimension): data detection for C antenna clusters can be performed in parallel, while maintaining the parallelism across subcarriers within each local antenna group. For local data detection at each subcarrier, the matrix and vector operations described in Algorithm 1, such as matrix-vector multiplication, vector addition/subtraction, multiplication and norm, involve further data-level parallelism across data elements.

In our implementation, we first generate C *message passing interface* (MPI) processes on the Phi cluster across a total of C Phi processors, each process controlling the data detection associated with an individual antenna group of S antennas on a single Phi processor. The local data detection workloads for a batch of N_{scr} subcarriers can be processed in parallel using multiple cores and VPUs on each Phi processor. Specifically, we implement the matrix-matrix or matrix-vector multiplications with the Intel *Math Kernel Library* (MKL), which automatically optimizes the computation efficiency on Intel processors using multithreading techniques and SIMD instructions for fast math

TABLE II
LATENCY (L) IN [MS] AND THROUGHPUT (T) IN [MB/S] AT $U = 16$.

	$S = 8$			$S = 32$		
Iter.	$C = 8, B = 64$ L / T	$C = 16, B = 128$ L / T	$C = 32, B = 256$ L / T	$C = 8, B = 256$ L / T	$C = 16, B = 512$ L / T	$C = 32, B = 1024$ L / T
1	7.801 / 206.7	8.512 / 189.5	9.182 / 175.6	8.925 / 180.7	9.613 / 167.8	10.40 / 155.1
2	11.18 / 144.2	12.29 / 131.2	13.05 / 123.5	12.35 / 130.6	13.38 / 120.5	14.26 / 113.1
3	14.52 / 111.1	16.08 / 100.3	17.16 / 93.97	15.65 / 103.0	17.18 / 93.88	18.26 / 88.32

processing. Here, we use the `cblas_cgemm_batch` MKL library call, which performs a batch of, for example, N_{scr} subcarriers, matrix-matrix or matrix-vector multiplications in a single function execution. By setting a large N_{scr} number, we can keep high occupancy of Xeon Phi computing resources. For vector addition/subtraction, multiplication and norm computations, we resort to our customized multithreading and SIMD implementations for higher design flexibility. In particular, on each Phi processor, we invoke N_t *OpenMP* threads to be deployed evenly across computing cores, where each thread independently handles local vector operations for N_{scr}/N_t subcarriers using VPUs. Here, at each thread, we perform vector computation by taking advantage of 512-bit *advanced vector extension* (AVX) SIMD intrinsics of Phi, which can process sixteen 32-bit floating-point elements in a single instruction on VPUs with 512-bit vector registers. While vector addition/subtraction can be executed element-wise on a pair of vectors, vector multiplication and norm, especially for complex-valued vectors, should be broken into multiple SIMD intrinsics including extra data shuffling and packing intrinsics for efficient vector operation on interleaved real and imaginary elements.

The collective centralized computations indicated at lines 4 and 11 of Algorithm 1 require data communication across all of the C processors. By performing the inter-process communication using `MPI_Allreduce` MPI function call, we can efficiently gather the local results generated at each Phi computing node for the sum reduction and then broadcast the calculated consensus back to every Phi node over the Omni-Path network.

IV. IMPLEMENTATION RESULTS

Our design is implemented on the Stampede KNL cluster of the Texas Advanced Computing Center (TACC) [11], which consists of 508 Intel Xeon Phi 7250 KNL 68-core compute nodes. Each KNL node runs a self-hosted CentOS 7 with a KNL-compatible software stack, including Intel compiler and Intel MKL, OpenMP and MPI libraries. The KNL nodes are connected with 100 Gb/s Omni-Path network with a fat-tree topology. We measure the timing characteristics using CPU wall-clock time by running our design compiled with the -O3 compiler optimization setting.

Table II summarizes latency and throughput performance for various antenna configurations indicated by U, S, C, and B for 64-QAM modulation. We perform measurements on processing a total of $N_{scr} = 1200 \times 14$ subcarriers as detection workloads, corresponding to a 20 MHz LTE subframe which contains 14 OFDM symbols with each symbol including 1200 subcarriers. For each antenna configuration, we show the data rate performance for up to three CG iterations, which are sufficient to achieve near-MMSE BER performance. We can see that for a given S, there is little performance degradation when we scale up the number B of BS antennas by increasing the number C of clusters. This observation demonstrates the high scalability and modularity of our proposed decentralized method for supporting hundreds or even thousands of BS antennas. For example, our reference Phi implementation can achieve over 110 Mb/s throughput for a very large 1024×16 massive MU-MIMO system at two CG iterations for a near-MMSE BER performance. We finally emphasize that, with possible lower message passing latency and higher computing capability, our approach would achieve throughputs in the Gb/s regime, e.g., by using FPGA or ASIC implementations.

REFERENCES

[1] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. Soong, and J. C. Zhang, "What will 5G be?," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

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

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

[4] K. Li, B. Yin, M. Wu, J. R. Cavallaro, and C. Studer, "Accelerating massive MIMO uplink detection on GPU for SDR systems," in *Proc. IEEE Dallas Circuits and Systems Conference (DCAS)*, Oct 2015, pp. 1–4.

[5] 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 International Conference on Mobile Computing and Networking*, Aug. 2012, pp. 53–64.

[6] 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 *Proc. IEEE Globecom Workshops*, Dec 2014, pp. 287–293.

[7] 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 *Proc. of the ACM SIGCOMM 2013*, New York, NY, USA, 2013, pp. 399–410, ACM.

[8] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 7, pp. 1754–1765, May 2011.

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

[10] http://www.hotchips.org/wp-content/uploads/hc_archives/hc27/HC27.25-Tuesday-Epub/HC27.25.70- Processors-Epub/HC27.25.710-Knights-Landing-Sodani-Intel.pdf, *Intel Xeon Phi Knights Landing processor*.

[11] https://portal.tacc.utexas.edu/user-guides/stampede, *Stampede Cluster at TACC*.