Procedia Computer Science

Volume 80, 2016, Pages 2169–2180

ICCS 2016. The International Conference on Computational Science

Efficient Sphere Detector Algorithm for Massive MIMO using GPU Hardware Accelerator

Mohamed-Amine Arfaoui[1], Hatem Ltaief[2], Zouheir Rezki[2], Mohamed-Slim Alouini[2], and David Keyes[2]

[1] Texas A&M University, Doha, Qatar
mohamed.arfaoui@qatar.tamu.edu

[2] Computer, Electrical, and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
{Hatem.Ltaief, Zouheir.Rezki, Slim.Alouini, David.Keyes}@kaust.edu.sa

Abstract

To further enhance the capacity of next generation wireless communication systems, massive multiple-input multiple-output (MIMO) has recently appeared as a necessary enabling technology to achieve high performance signal processing for large-scale multiple antennas. However, massive MIMO systems inevitably generate signal processing overheads, which translate into ever-increasing rate of complexity, and therefore, such systems may not maintain the inherent real-time requirement of wireless systems. We redesign the nonlinear sphere decoder method to increase the performance of the system, cast most memory-bound computations into compute-bound operations to reduce the overall complexity, and maintain the real-time processing thanks to the graphics processing unit (GPU) computational power. We show a comprehensive complexity and performance analysis on an unprecedented MIMO system scale, which can ease the design phase toward simulating future massive MIMO wireless systems.

Keywords: Massive MIMO, Sphere Decoder Algorithm, GPU Acceleration, Real-Time Systems.

1 Introduction

Massive multiple-input multiple-output (MIMO) is a new emerging technology that scales up the standard MIMO compared to the current state-of-the-art. With massive MIMO, systems use antenna arrays possibly with a few hundred antennas, simultaneously serving in the same time-frequency resource tens of terminals. The basic premise behind massive MIMO is to amplify all the benefits of conventional MIMO, but on a larger scale. Overall, massive MIMO is the key to enable the development of future generation networks (fixed and mobile), which will be robust, secure and energy-efficient, and will use the spectrum efficiently.

Despite the great potential of massive MIMO, it is commonly agreed that achieving (almost) real-time signal processing for power-efficient and reliable communications with a large

doi:10.1016/j.procs.2016.05.377

number of antennas is quite cumbersome. This is in part due to the complexity of optimal decoding algorithms that grows exponentially with the number of antennas. Clearly, there is a trade-off between complexity and error probability performance for massive MIMO constitutes a daunting challenge and a unique opportunity. The opportunity arises from the necessary re-design of existing detection algorithms to follow the evolution of the high performance hardware architecture landscape.

In this paper, we study this trade-off by redesigning the nonlinear sphere decoder algorithm (SD) and proposing an efficient SD implementation on graphics processing units (GPU) in the context of massive MIMO systems. Based on the original Maximum Likelihood decoder (ML), the SD method provides high performance in terms of bit error rates (BER) but high complexity in terms of elapsed time, due to the large-scale antenna system. This high complexity can be reduced (a) by tuning the sphere radius, which contains the pool of possible candidates, i.e., the transmitted signal vectors, (b) by exposing more parallelism thanks to a breadth-first tree traversal, and, (c) by casting the memory-bound matrix-vector multiplication kernels into compute-bound matrix-matrix multiplication operations. These standard basic linear algebra subroutines (BLAS) operations [1] are often well-supported on today's processing units through high performance numerical libraries, provided by the vendors' hardware architectures, e.g., cuBLAS [12] for NVIDIA GPUs.

To the best of our knowledge, most theoretical and experimental results presented in the literature rely on linear detection algorithms to trade performance against complexity on a rather limited MIMO system, typically less than ten antennas. Here, we present performance and complexity analysis of antenna systems of an order of magnitude higher, thanks to our new GPU-based SD framework implementation. We consider only uncoded signals and we adopt the case of a perfect channel state information, i.e., the channel matrix is known by the decoder. For a configuration of 50×25 transmitter/receiver antennas, the obtained complexity (10ms) is close to real-time while the BER performance ($1e^{-2}$) is decent for ensuring high quality uncoded wireless systems. The corresponding signal-to-noise ratio (SNR) is 17db, which translates at least to a 22% saving compared to the state-of-the-art decoder methods, for the same BER performance.

The paper is organized as follows. Section 2 describes related work. Section 3 provides background information on the system model and existing detection algorithms. The nonlinear SD algorithm is illustrated in Section 4. Our proposed parallel SD algorithm based on the breadth-first-search tree algorithm is highlighted in Section 5 and its GPU implementation in Section 6. Section 7 shows a comprehensive performance and complexity analysis of our SD implementations against existing state-of-the-art implementations on various antennas configurations. Finally, Section 8 summarizes the results of the paper and presents some future works.

2 Related Work

Achieving high throughput with reduced complexity and high performance on both optimal detection and optimal precoding for massive MIMO systems has been studied extensively in the literature, with many attempts to adopt hardware accelerators and software design. A Fixed Complexity Sphere Decoder (FCSD) for large scale MIMO uplink systems is presented in [2], where the authors propose to use GPUs to reduce the complexity. In [18], the authors propose an application specific integrated circuit (ASIC) to reduce the computational complexity of linear soft-output detection. A flexible N-Way MIMO detector is presented in [17] with excellent error-rate performance and high throughput on GPUs. A new parallel algorithm is introduced in [8], inspired by the Sphere Detector (SD) algorithm, which can efficiently solve the ML detection of the MIMO systems with high throughput on parallel architectures. In [9], the authors present a high performance GPU-based software-defined base station to achieve both real-time high performance and high reconfigurability. A reconfigurable GPU-based uplink detector (linear MMSE) is presented in [11] for massive MIMO software-defined radio (SDR) systems.

All aforementioned related works performed experimental results on small MIMO configuration, typically less than 10 antennas. Our work presents a new algorithm for the sphere decoder based on the breadth-first-search tree algorithm on an unprecedented number of antennas (50×25). The GPU-based implementation uses the highly optimized numerical library NVIDIA cuBLAS [12]. Our proposed efficient algorithm not only achieves the same results in terms of bit error rate (BER) for different large MIMO configurations but also reaches real-time signal processing starting from a fixed signal-to-noise ratio (SNR), while sustaining close to real-time complexity.

3 System Model and Detection Algorithms for MIMO

This section gives some background information on the typical model for MIMO system and recalls the linear and nonlinear detection algorithms.

3.1 System Model

For an $M \times N$ MIMO configuration, the transmitter sends different signal streams on M antennas and the receiver receives N different signal streams, one per receiver antenna. An $M \times N$ MIMO system can be described by the input-output relation $y = \mathbf{H}s + w$, where $\boldsymbol{y}= [y_1, ..., y_N]^T$ is the received vector. $\boldsymbol{H}$ is the $N \times M$ channel matrix, where each element $\mathbf{H}_{i,j}$ is an independent zero mean circularly symmetric complex Gaussian random variable with variance σ_c^2. The noise at the receiver is $\boldsymbol{w}= [w_1, w_2, ..., w_N]^T$, where w_i is an independent zero mean circularly symmetric complex Gaussian random variable with variance σ^2. The transmitted vector is $\boldsymbol{s}= [s_1, s_2, ..., s_M]$, where s_i is drawn from a finite complex constellation alphabet set Ω of cardinal M_c, ($s \in S = \Omega^M$). Once the model system is defined, there are usually two types of detection algorithms to recover the transmitted signals, which are described in the subsequent sections. We also assume that there is an average power constraint on the input $E[\|s\|^2] <= P$, where P is the average transmit power.

3.2 Linear Decoders

Let $\mathbf{H}_{inv}$ be an $M \times N$ linear detector matrix which depends on the channel H. By using the linear detector, the received signal is separated into streams by multiplying the received signal $\boldsymbol{y}$ by $\mathbf{H}_{inv}^H$ to get the decoded signal $\hat{s} = Q(\mathbf{H}_{inv}^H \cdot y)$, where $Q(.)$ designates a mapping function to the original constellation. The three conventional linear detectors are the maximum ratio combining (MRC) $\mathbf{H}_{inv} = \mathbf{H}$, the zero forcing (ZF) $\mathbf{H}_{inv} = \mathbf{H} \cdot (\mathbf{H}^H.\mathbf{H})^{-1}$, and the minimum mean square error (MMSE) $\mathbf{H}_{inv} = \mathbf{H} \cdot (\mathbf{H}^H.\mathbf{H} + \frac{1}{SNR} \cdot I_n)^{-1}$, where SNR is defined as a measure of the signal-to-noise-ratio of the link, i.e., $SNR = P/\sigma^2$.

3.3 Nonlinear Decoders

3.3.1 The Maximum Likelihood Decoder

The Maximum Likelihood (ML) [3,4,7] decoder calculates the *a posteriori* probability in terms of log likelihood ratio for each possible transmitted vector s by browsing all the set S.
The ML estimate of the transmitted vector s, $\hat{s}_{ML}$ is given by:

$$nonlinear\hat{s}_{ML} = \underset{s \in S}{\operatorname{argmin}} \quad ||y - Hs||^2. \tag{1}$$

Thus, the ML detector chooses the messages, which yields the smallest distance between the received vector y and the hypothesized message Hs. Since the optimization problem is performed over the set $S = \Omega^M$, the algorithmic complexity of the ML decoder is $O({M_c}^M)$, which becomes very high for a large number of transmitted antennas and for hard constellations.

3.3.2 The Sphere Decoder

The sphere decoder [5,6,10,14] is a variant decoding scheme of the ML decoder with a lower complexity than the ML decoder. The idea behind the sphere decoder is to solve equation 1 by enumerating all points, which belong to a hypersphere of radius r around the received point y, i.e., all points which satisfy a criterion on the form given by:

$$||y - Hs||^2 \leq r^2. \tag{2}$$

The search process of the ML solution through the sphere decoding algorithm is a combinatorial optimization problem which can be solved using the ascendant-tree-search algorithms. For a fixed radius r, the algorithmic complexity of the sphere decoder is equal to $O(Mc^{\gamma M})$, where $\gamma = \gamma_{H,w,s}(r)$ is a real random variable between 0 and 1 and whom statistic is induced by those of the channel matrix H, the signal noise at the receiver w and the transmitted signal s.

The next section provides further information on the sphere decoder algorithm and introduces a more efficient search process, associated with an adequate sphere radius definition.

4 Redesigning The Sphere Decoder Algorithm

4.1 Problem Statement

As we have defined the sphere decoder in the previous section, we are going to transform the problem in Equation 2 to another equivalent problem. Let $H = QR$ the QR decomposition of the matrix H, where $Q \in C^{N \times N}$ an orthogonal matrix and $R \in C^{N \times M}$ an upper triangular matrix. We note by $\bar{y} = [\bar{y}_1, \bar{y}_2, ..., \bar{y}_N]^T$ the vector $Q^H y$ and since Q is orthogonal, we can simplify Equation 2 with $||y - Hs||^2 = ||\bar{y} - R.s||^2$. The sphere decoding algorithm can then be written as:

$$\begin{aligned} \hat{s}_{SD} &\triangleq \underset{s \in L_M}{\operatorname{argmin}} \quad ||\bar{y} - Rs||^2 \\ L_M &\triangleq \{s \in S, \ \ ||\bar{y} - Rs||^2 \leq r^2\}. \end{aligned} \tag{3}$$

4.2 The Search Process

To solve Equation 3, we consider the set L_k for $1 \leq k \leq M$ defined by $L_k = \{\mathbf{s}_k \in \Omega^k, ||\bar{\mathbf{y}}_k - R_k\mathbf{s}_k||^2 \leq r^2\}$, where R_k, is the lower right bloc of R and $\bar{\mathbf{y}}_k$ the lower bloc of $\bar{y}$. The enumeration procedure can be described by a tree pruning algorithm, which enumerates all points satisfying Equation 3 recursively, starting with $k = 1$. Note that $\bar{\mathbf{y}}_k$ is the vector given by $\bar{\mathbf{y}}_k = [s_{M-k+1}, ..., s_M]^T$. Figure 1 illustrates the search process for the sphere decoder for each level of the tree for $M = 3$ and BPSK modulation $\left(\Omega = \{-1, 1\}^3\right)$. The sphere decoder algorithm traverses all nodes of the tree from top to bottom and searches for possible candidates. At each depth, k, it verifies if some candidates $s_{M-k+1} \in \Omega$ will satisfy $[s_{M-k+1}, \mathbf{s}_{k-1}] \in L_k$, with $\mathbf{s}_{k-1}$ being the previous choice. The two known tree-search algorithms are the depth-first-search tree (DFS) and the breadth-first-search tree (BFS). The algorithmic complexity of the BFS and DFS algorithms are the same, which is equal to $O(M)$ in our case. However, the BFS algorithm exposes more parallelism compared to the DFS algorithm. Indeed, because of the nature of the DFS algorithm, only a single possible candidate is processed at a time and this limits the degree of parallelism. Therefore, the BFS algorithm opens new opportunities for efficient implementations on highly parallel architecture systems in order to reduce time to solution.

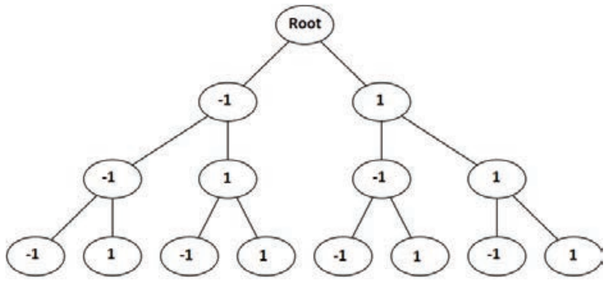


Figure 1: The sphere decoder based on the breadth-first-search (BFS) tree algorithm for $M = 3$ and BPSK modulation $\left(\Omega = \{-1, 1\}^3\right)$.

4.3 The Sphere Radius

A critical parameter of the sphere decoder algorithm is the radius since the performance (in terms of BER) and the complexity (in terms of elapsed time) of this detector directly depend on its magnitude. Indeed, if the sphere radius is large, the sphere detector reaches the ML detector with high performance (low BER) and high complexity (slow computation time). Otherwise, if the sphere radius is small, the sphere decoder algorithm moves away from the ML decoder with low performance (high BER) and low complexity (fast computation time). An ideal choice of the criterion for selecting r is a radius such that the sphere contains the ML solution as well as the minimum possible number of other candidates from S, so that both, high performance and low complexity are achieved. The idea is to evaluate the expectation of $||\bar{y} - Rs||^2$, which is a random variable such that its probability distribution is induced by those of R, s and w, for the case where $s = s_{ML}$. Consequently, $||\bar{y} - Rs||^2$ follows a gamma distribution with parameters N and σ^2, saying $\gamma(n, \sigma^2)$. So, $\mathbb{E}(||\bar{y} - Rs||^2) = N\sigma^2$ and $\mathrm{var}(||\bar{y} - Rs||^2) = N\sigma^4$. The goal here is to chose the radius r such that $r^2 = N\sigma^2$ to ensure most of the ML predominant solution candidates will be encapsulated within the sphere. For the case where the sphere is empty, we update the radius with $r^2 = r^2 + var(||\bar{y} - Rs||^2) = r^2 + N\sigma^4$.

The next section describes the parallel sphere decoder algorithm, which combines the efficient BFS technique with the aforementioned choice of radius r.

5 Parallel Bread-First-Search Sphere Decoder Algorithm

The concurrency delivered by the BFS enables the processing of all nodes located in the same tree level in an embarrassingly parallel fashion. We reconsider the sphere decoder problem in Equation 2. The algorithm sweeps all the levels l_k of the tree one by one $\forall 1 \leq k \leq M$. At each stage l_k, we determine the set L_k defined in Equation 3. At the bottom of the tree i.e., level l_M, we search the minimum of the set L_M, which corresponds to the ML solution of the sphere decoder problem. For each level $l_k, k = 2, .., M$, we note by M_{k-1} the matrix which contains all the vectors of the set L_{k-1}, G_{k-1} the cardinal of L_{k-1}. We design a matrix V_k with size $(k,\ M_c G_{k-1})$. This matrix contains all the vectors $\underline{s}_{k-1} \in L_{k-1}$ duplicated M_c times and then for each one we add a possible constellation symbol from Ω. Now, by browsing all the selected signals in the matrix V_k, we evaluate for each signal $\underline{s}_k$ the quantity $||\bar{y}_k - R_k \underline{s}_k||^2$ and we verify if it is smaller than r^2 or not. This criterion can be expressed for all the signals simultaneously using a matrix formulation based on the matrix-matrix multiplication computational kernel, a highly compute-intensive operation, which runs close to the theoretical peak performance of the system. Let $\mathbf{P}_k$ be the matrix defined by $\mathbf{P}_k = \bar{\mathbf{y}}_k - R_k V_k$, where $\bar{\mathbf{y}}_k$ is the matrix which contains the vector $\bar{y}_k$ duplicated $M_c G_{k-1}$ time. If the DFS algorithm was used, this criterion will be based on the matrix-vector multiplication kernel, which is memory-bound, i.e., its parallel performance is solely limited by the bus bandwidth of the system. We note also by P_k the $1 \times M_c G_{k-1}$ matrix which contains the square Euclidean norm of each vector $\mathbf{P}_{K,i}$ in $\mathbf{P}_k$. The previous criterion is reduced now to just verifying if the weights in the matrix P_k are lower then r^2 or not (under the sphere). For each evaluation, we generate the matrix M_k and we derive the set L_k. If the L_k is empty, we increase the radius r and we restart the algorithm. For the case of k=1, the process will be the same by considering L_0, which corresponds to the root node, an empty set. By reaching the last level, we search the minimum weight in P_M and we derive then from the matrix M_M the s_{ML} solution. Algorithm 1 summarizes the sphere decoder algorithm based on the BFS tree algorithm for $M \leq N$ MIMO systems, starting from the parameters estimations of the decoded received signal. This algorithm can be also generalized for $M > N$. This standard search process can be further generalized by accumulating and processing nodes not only from a single level of the tree, but also from multiple subsequent levels of the tree simultaneously. The number of levels of the tree to consider is a runtime tunable parameter, which should be carefully chosen, as it represents a trade-off between parallel performance and concurrency.

6 Implementation Details

Due to the real-time processing requirements of wireless systems, GPUs appear to be a suitable architecture to consider because these devices are capable of sustaining high memory throughput in terms of bandwidth (byte/s) and high computation rate in terms of floating point operations per second (flops/s), as opposed to standard x86 CPU architectures. Furthermore, given the low power envelope available at the base station, GPUs are also one of the most energy efficient architectures and may deliver up to 10 times more computational power for the same energy consumption [15]. However, there are two main challenges to assess before working with theses devices. The GPU memory is located off-chip and can be accessed through a slow PCIe link,

Algorithm 1 Parallel Standard BFS Tree Algorithm

```
Inputs:
Received signal: $y$
Constellation order: $\Omega$;
Channel estimation: $H$
Noise variance estimation: $\sigma^2$
QR Decomposition:
  $H = QR$
Preliminary:
  $\bar{y} = Q^H y$
Initialization:
$r^2 = M\sigma^2$
$M_0 = [\,]$
$G_0 = 1$
$Empty = 1$
 1: while $Empty = 1$ do
 2:    for $k = 1 : 1 : M$ do
 3:       $Calculate\, V_k(M_{K-1})$
 4:       $\bar{\mathbf{y}}_k = \bar{y}_k \circ \mathbb{I}(1, M_c G_{k-1})$
 5:       $\mathbf{P}_k = \bar{\mathbf{y}}_k - R_k V_k$
 6:       $Calculate\, P_k$
 7:       $size = 0$
 8:       for $i = 1 : 1 : M_c G_{k-1}$ do
 9:          if $P_k(i) \leq r^2$ then
10:             $M_k(:, i) = V_k(:, i)$
11:             $size++$
12:          end if
13:       end for
14:       if $size \neq 0$ then
15:          $Empty = 0$
16:       else
17:          $Empty = 1$
18:          $r^2 = r^2 + M\sigma^4$
19:          $break$
20:       end if
21:    end for
22: end while
23: $\hat{s}_{ML} = M_M(:, min(P_M))$
```

requiring two orders of magnitude less bandwith than the internal GPU bandwidth. In other words, for parallel performance, it is important to reduce the data off-loading between the CPU/device memory, by reusing the freshly moved data on the GPU board as much as possible. This persistent data technique is made possible by the BFS tree algorithm, which allows to cast operations involving multiple subsequent levels of the tree into a single matrix-matrix multiplication kernel. The other challenge resides in the programmability and/or productivity. Scientific codes can be accelerated on GPUs through compiler directives [16], CUDA programming model [13], or accelerated libraries (e.g., NVIDIA cuBLAS [12]). In the BFS tree implementation, we rely on the efficient NVIDIA implementation of the matrix-matrix multiplication kernel (CGEMM) from cuBLAS to deliver high performance computing. This kernel is part of the legacy BLAS [1] implementation and are supported by many other vendors on different x86 and accelerators, which make our code portable from one give architecture to another.

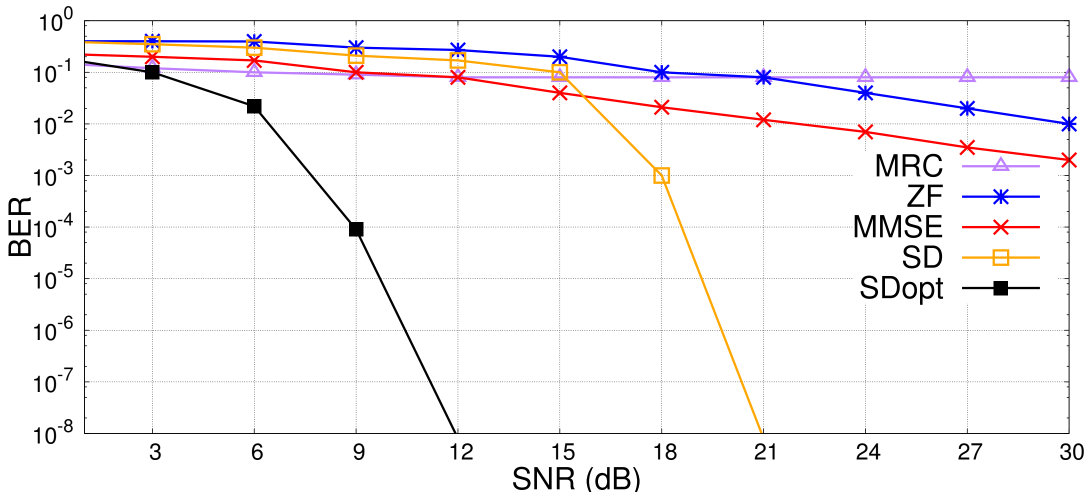
7 Performance and Complexity Analysis

7.1 Environments Settings

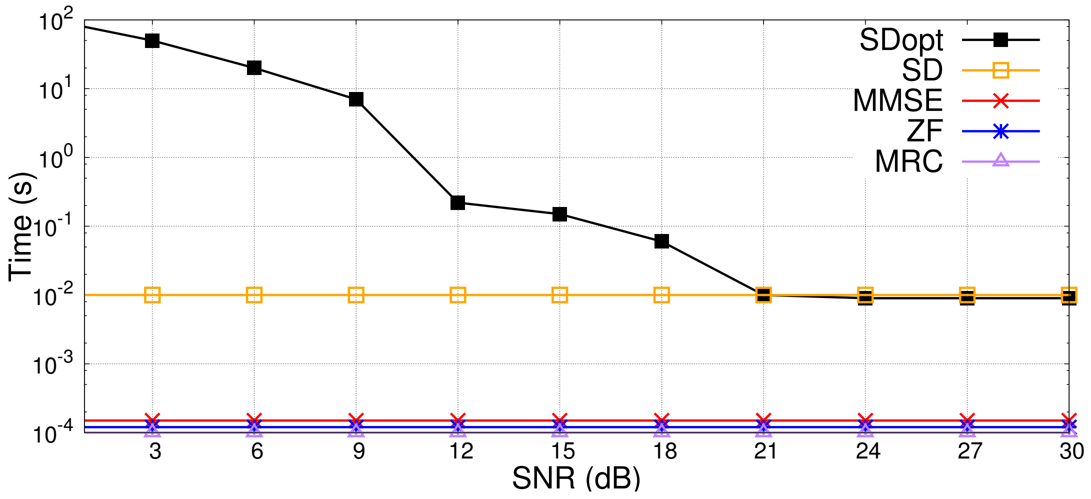
The host system is equipped with two-sockets of ten-core Intel Ivy Bridge (20 cores total) running at 2.8GHz with 256GB of memory. There are three Tesla Kepler K40 GPU accelerators attached to the CPU board through PCIe Gen3 x16, each with 12GB of GDDR5 memory. All experiments reported in this section consider only uncoded signals and we adopt the case of a perfect channel state information, i.e., the channel matrix is known by the decoder. All computations are performed in single complex precision arithmetic.

7.2 Experimental Results

Figure 2 presents the performances of the sphere decoder, with optimal and fixed radius against the linear decoders for a 50×25 MIMO system and BPSK modulation. In particular, Figure 2(a) shows that the sphere decoder performs better in terms of error probability performance than the linear decoders. For the bit error rate (BER) equal to 10^{-2}, which is a common error



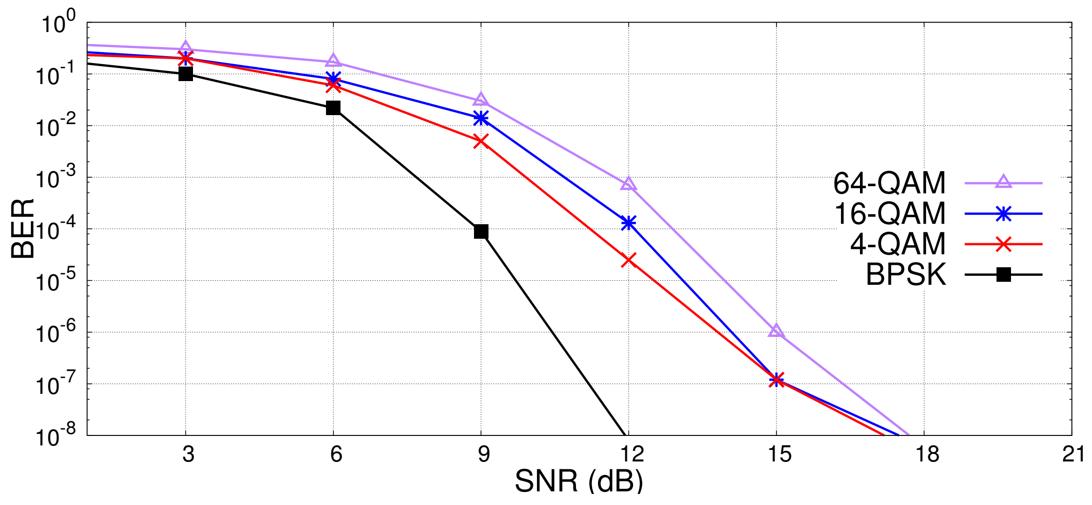
(a) Bit error rate of the sphere decoder and the linear receivers for 50×25 MIMO system and BPSK modulation.



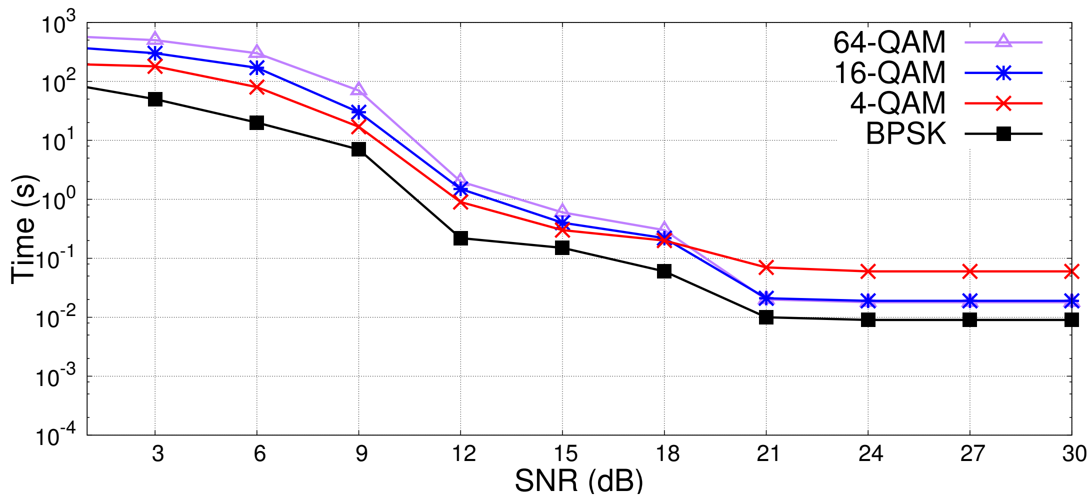
(b) Execution time of the sphere decoder and the linear receivers for 50×25 MIMO system and BPSK modulation.

Figure 2: Performances of the sphere decoder and the linear receivers for 50×25 MIMO system and BPSK modulation.

probability performance for wireless systems, the sphere decoder reaches this rate for $SNR =$ 7dB with an optimal radius and for $SNR =$ 17dB with a fixed radius, while the best linear decoder (which is MMSE) reaches it for $SNR =$ 21dB, giving a difference in power of the order of 14dB for the optimal radius and 4dB for a fixed radius which is very humongous for both cases. We can also notice that the BER of the sphere decoder degrades in case we adopt a low fixed radius. Concerning the execution time, Figure 2(b) shows that the linear receivers and the sphere decoder with fixed radius are almost real-time for such system's configuration. However, for the sphere decoder with optimal radius, we note that the decoding process can be performed in real-time, starting from $SNR =$ 21dB. Figure 3 presents the performances of the sphere decoder with optimal radius for a 50×25 MIMO system and different modulations. This figure shows that, for fixed number of transmitter and receiver antennas, the performances decrease



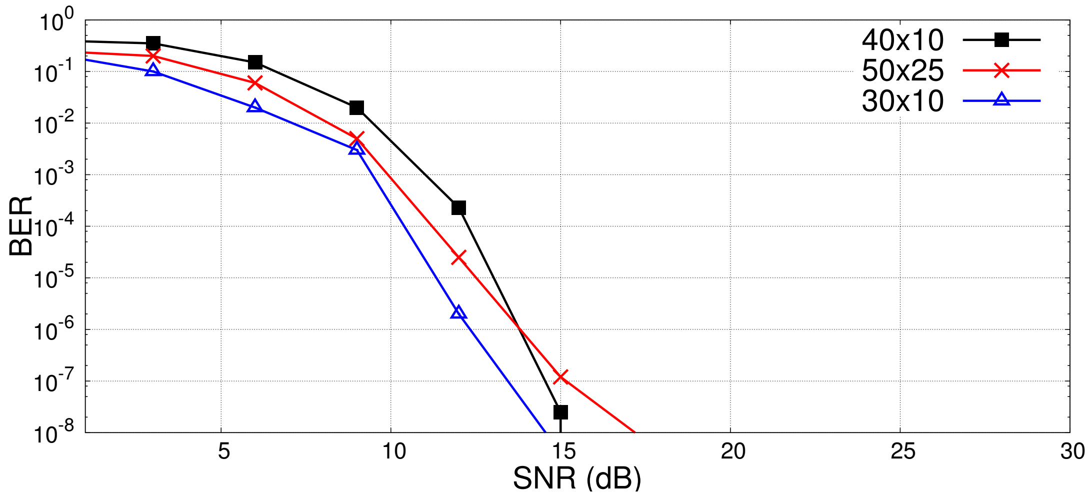
(a) Bit error rate of the proposed sphere decoder for 50×25 MIMO system and different constellation.



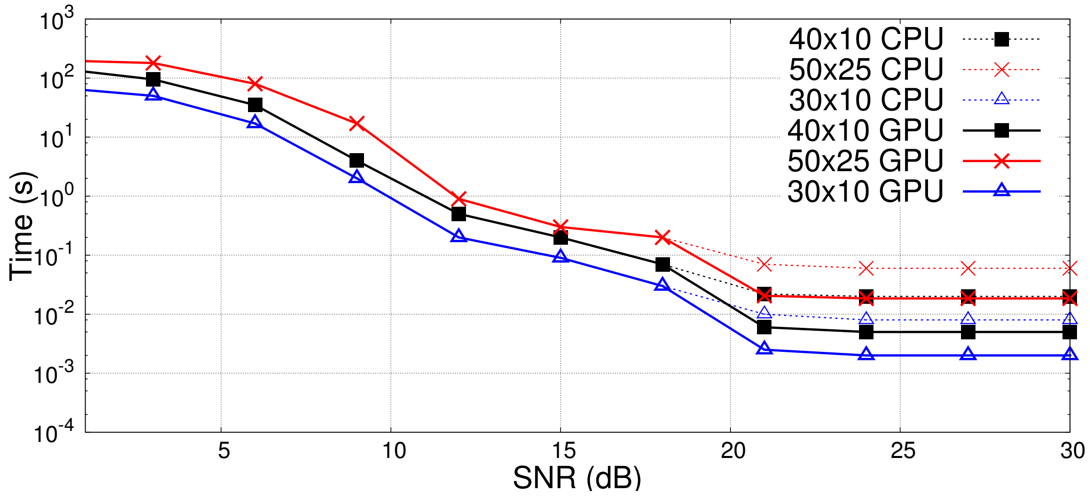
(b) Execution time of the proposed sphere decoder for 50×25 MIMO system and different constellation.

Figure 3: Performances of the proposed sphere decoder for 50×25 MIMO system and different constellation.

when we increase the order of the modulation (BPSK $= 2^1$, 4-QAM $= 2^2$, 16-QAM $= 2^4$ and 64-QAM $= 2^8$). Figure 4 illustrates the performances of the proposed sphere decoder algorithm for 4-QAM across various MIMO configuration systems on CPU and GPU platforms, i.e., 50×25, 30×10 and 40×10. From Figure 4(a), the bit error rate for a given MIMO configuration (MIMO system + modulation) does not change between GPU and CPU, however, it is as good as the diversity order is large. We mean by the diversity order of a $M \times N$ MIMO system the number $d = N$. Concerning the execution time, Figure 4(b) shows the gain of time resulted when accelerating the computation with GPUs. Finally, we can notice that the execution time and the complexity increase as the number of the transmitter antennas increases. For even larger MIMO systems, one may consider revisiting the algorithm by further tuning the number of subsequent levels to process simultaneously and/or adding more hardware resources to keep the complexity close to real-time.



(a) Bit error rate of the sphere decoder for 4-QAM modulation and for 50×25, 30×10 and 40×10 MIMO system.



(b) Execution time of the sphere decoder for 4-QAM modulation and for 50×25, 30×10 and 40×10 MIMO system on GPU and CPU platforms.

Figure 4: Performances of the sphere decoder for 4-QAM modulation and for 50×25, 30×10 and 40×10 MIMO system on GPU and CPU platforms.

8 Conclusions

A new implementation of the breadth-first-search tree in the context of the non linear sphere decoder algorithm has been presented on a GPU plateform. The experimental results show that we can sustain real-time signal decoding starting from a given SNR for each configuration of the massive MIMO system while keeping a low bit error rate (BER). In particular, for a configuration of 50×25 transmitter/receiver antennas, the largest MIMO system studied up to now, the obtained complexity (10ms) is close to real-time while the BER performance ($1e2$) is decent for ensuring high quality of service (QoS) wireless systems. The corresponding signal-to-noise ratio (SNR) is 16db, which translates at least to a 22% saving compared to the existing state-of-the-art decoder methods, for the same BER performance. The overall framework permits also to understand the current limitations of existing signal detection algorithms in presence of large MIMO systems by identifying some possible cross-over points. Furthermore, we would like to investigate the performance and complexity behavior for even larger MIMO configurations. This may require running on multiGPU systems to cope with the real-time challenge.

References

[1] BLAS. Basic Linear Algebra Subprograms. http://www.netlib.org/blas.

[2] Tianpei Chen and Harry Leib. GPU Acceleration for Fixed Complexity Sphere Decoder in Large MIMO Uplink Systems. In *Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on*, pages 771–777. IEEE, 2015.

[3] Chi-Chao Chao and R.J. McEliece and L. Swanson and E.R. Rodemich. Performance of Binary Block Codes at Low Signal-to-Noise Ratios. *IEEE Transactions On Information Theory*, 38(6):1677–1687, 1992.

[4] Chi-Chao Chao and R.J. McEliece and L. Swanson and E.R. Rodemich. Performance of Binary Block Codes at Low Signal-to-Noise Ratios. *IEEE Transactions On Information Theory*, 38(6):1677–1687, 1992.

[5] E. Agrell and T. Eriksson and A. Vardy and K. Zeger. Closest Point Search in Lattices. *IEEE Transactions On Information Theory*, 48(8):2201–2214, 2002.

[6] H. Vikalo and B. Hassibi and T. Kailath. Iterative Decoding for MIMO Channels via Modified Sphere Decoding. *IEEE Transactions Wireless Communication*, 3(6):2299–2311, 2004.

[7] Simon Haykin. *Communication Systems*. J. Wiley, fourth edition, 2001.

[8] Csaba M Józsa, Géza Kolumbán, Antonio M Vidal, Francisco-José Martínez-Zaldívar, and Alberto González. New Parallel Sphere Detector Algorithm Providing High-Throughput for Optimal MIMO Detection. *Procedia Computer Science*, 18:2432–2435, 2013.

[9] Kaipeng Li and Michael Wu and Guohui Wang and Joseph R. Cavallaro. A High Performance GPU-Based Software-Defined Basestation. *48th IEEE Asilomar Conference on Signals, Systems, and Computers (ASILOMAR)*, pages 2060–2064, 2014.

[10] L. Brunel. Multiuser Detection Techniques Using Maximum Likelihood Sphere Secoding in Multicarrier CDMA Systems. *IEEE Transactions Wireless Communication*, 3(3):949–957, 2004.

[11] Kaipeng Li, Bei Yin, Michael Wu, Joseph R Cavallaro, and Christoph Studer. Accelerating massive mimo uplink detection on gpu for sdr systems. In *Circuits and Systems Conference (DCAS), 2015 IEEE Dallas*, pages 1–4. IEEE, 2015.

[12] NVIDIA. The NVIDIA CUDA Basic Linear Algebra Subroutines. https://developer.nvidia.com/cublas.

[13] NVIDIA. The NVIDIA CUDA Programming Model. https://developer.nvidia.com/cuda-downloads.

[14] O. Damen and A. Chkeif and J. C. Belfiore. Lattice Code Decoder for Space-Time Codes. *IEEE Communications Letters*, 4(5):161–163, 2000.

[15] Tom Scogland and Balaji Subramaniam and Wu-chun Feng. The Green500 List: Escapades to Exascale. *Computer Science - R&D*, 28(2-3):109–117, 2013.

[16] Michael Wolfe. Introduction to GPU Computing with OpenACC. In *SC'12 Tutorials CD-ROM: Conference on High Performance Computing Networking, Storage and Analysis*, Salt Lake City, UT, USA, November 2012. ACM SIGARCH/IEEE Computer Society.

[17] Michael Wu, Bei Yin, Guohui Wang, Christoph Studer, and Joseph R Cavallaro. GPU Acceleration of a Configurable N-Way MIMO Detector for Wireless Systems. *Journal of Signal Processing Systems*, 76(2):95–108, 2014.

[18] Bei Yin, Min Wu, Guohui Wang, Chris Dick, Joseph R Cavallaro, and Christoph Studer. A 3.8 Gb/s Large-Scale MIMO Detector for 3GPP LTE-Advanced. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3879–3883. IEEE, 2014.