Cooperative Spectrum Sensing in TV White Spaces: When Cognitive Radio Meets Cloud

Chun-Hsien Ko, Din Hwa Huang and Sau-Hsuan Wu,
Institute of Communications Engineering, National Chiao Tung University, Hsinchu, Taiwan 300
E-mail:{dke.cm99g@nctu.edu.tw, dinhwa.cm94g@g2.nctu.edu.tw, sauhsuan@cm.nctu.edu.tw}

Abstract—**A Cognitive Radio Cloud Network (CRCN) in TV White Spaces (TVWS) is proposed in this paper. Under the infrastructure of CRCN, cooperative spectrum sensing (SS) and resource scheduling in TVWS can be efficiently implemented making use of the scalability and the vast storage and computing capacity of the Cloud. Based on the sensing reports collected on the Cognitive Radio Cloud (CRC) from distributed secondary users (SUs), we study and implement a sparse Bayesian learning (SBL) algorithm for cooperative SS in TVWS using Microsoft's Windows Azure Cloud platform. A database for the estimated locations and spectrum power profiles of the primary users are established on CRC with Microsoft's SQL Azure. Moreover to enhance the performance of the SBL-based SS on CRC, a hierarchical parallelization method is also implemented with Microsoft's dotNet 4.0 in a MapReduce-like programming model. Based on our simulation studies, a proper programming model and partitioning of the sensing data play crucial roles to the performance of the SBL-based SS on the Cloud.**

Index Terms—**Cognitive Radio, Cloud Computing, MapReduce, Windows Azure, Sparse Bayesian Learning.**

I. Introduction

The concept of cognitive radio (CR) is first introduced by Joseph Mitola III in [1]. Since then CR has attracted significant research attentions either in telecommunications technologies or their related regulations. In view of the inefficient usages of the licensed spectrum (less than 25% overall [2]) and the opportunity of the termination of analog TV broadcasting, the Federal Communications Commission (FCC) of the U.S has established the regulation for CR accesses in its TV White Spaces (TVWS) in 2007 [3] and has recently granted field tests of the CR network in part of the TVWS. Encouraged by the acts of FCC, many international organizations have also started to define CR standards on TVWS, *e.g.* IEEE 802.22, 1900, 802.16m and ECMA 392, etc.

To ensure the received signal quality of TV sets, the FCC requires CR operators in TVWS being able to detect the TV signal even if its strength is 0.8dBm below the noise level (-106.2dBm). In addition, the CR operators should also provide databases that maintain the geographical locations of TV base stations (BS) and their radiation powers, antenna heights and numbers of channels, *etc*. To help achieve these goals of spectrum sensing (SS) in TVWS, the secondary users (SUs) of the CR network in TVWS are suggested to provide their sensing data and geographical locations for CR operators to perform cooperative SS.

Compared to the BSs of regular cellular networks, the radiation power of a TV base station (BS) typically covers a much larger area than that covered by the transmit power of a mobile device. To reconstruct the radiation power profile of even a TV BS, it requires sensing reports from SUs located in different positions inside the coverage area of the TV signal. However, the distribution and population density (sparsity) of SUs are not uniform in different areas, and vary in different time of a day. Moreover, the received signal strength of a SU is likely to be attenuated by the shadowing effects of wireless channels. Considering these characteristics of sensing measurements and the strict requirements of SS in TVWS, we study a cooperative SS algorithm for TVWS in this paper based on the concept of sparse Bayesian learning (SBL) and the relevance vector machine (RVM) [4].

As the number of SUs vary with time, the computational demands to reconstruct the power propagation map (PPM) of a large area may change significantly over time if sensing reports inside the area are all used in cooperative SS. To control the algorithm complexity and in the meantime to maintain the quality of SS, not only should the number of sensing measurements be limited inside an area, but the area from which measurements are collected for the computation of a RVM should also be adjusted with time. Consequently, the overall computational quantity to reconstruct the PPM of a nation or region will scale up and down significantly over time. This makes the SS in TVWS an ideal application for Cloud computing.

A similar concept of cognitive wireless Cloud (CWC) has been introduced by H. Harada *et al* in [5] where they consider a heterogeneous network that consists of various types of wireless networks and propose a Cloud-based algorithm to optimize the spectrum resource scheduling among the heterogeneous networks in CWC. In contrast to their ideas in heterogeneous networks, we propose herein a more complete concept of Cognitive Radio Cloud Network (CRCN) that enables and integrates cooperative SS and resource scheduling in TVWS. Making use of the scalability and the vast storage and computing capacity of the Cloud, the database of PPM can be established, updated and accessed by a large amount of SUs in an efficient manner. Under this infrastructure of CRCN, we study and implement a SBL-based cooperative SS algorithm on Microsoft's Windows Azure Cloud platform, and propose a scalable mapping method under a MapReduce-like programming model to dynamically partition the geographical area according to the SU density. Utilizing the scalable mapping method and the dynamic computing resource allocation of the Cloud, the CRCN can provide a PPM in different precisions according to the density of SUs. Based on our simulation studies, a proper programming model and partitioning of the sensing data play crucial roles to the performance of the SBL-based cooperative SS on the Cloud.

This paper is organized as follows. Section II specifies the system model for CRCN and provides some background knowledges on Windows Azure. Section III reviews the basic concept of SBL and the MapReduce [6] programming model, and introduces a SBL-based cooperative SS and a scalable mapping method on Window's Cloud platform. Simulation results are presented in Section IV followed by some conclusions and discussions made in Section V.

II. The Infrastructure of CRCN

The purpose of CR is to utilize the precious radio resources more intelligently. Fig. 1 illustrates the infrastructure of the CRCN proposed in this paper. SUs in the CRCN are allowed to use a spectrum in time and space as long as not seriously deteriorating the signal qualities of primary users (PUs) in the same spectrum. To make the most out of the available spectra, a command and control center (also referred to the CR Cloud (CRC)) is used to coordinate and manage the entire radio resources in TVWS. In the CRCN, there are various CR BSs to collect sensing measurements from distributed SUs. The sensing results are fed back through CR BSs to the CRC to estimate the PPM with a SBL algorithm implemented on Microsoft's Windows Azure. The resultant PPM of SS contains the number of PUs and their locations and corresponding radio power profiles, and are stored in Microsoft's SQL Azure.

A. The Windows Azure CRC Platform

The CRC is in fact implemented on Microsoft's Windows Azure Cloud platform which can support program developments in JAVA or in C$^\sharp$ and Visual Basic on Visual Studio. The operating system for Windows Azure is Windows Azure Guest OS 1.8 which is a virtual machine (VM) version of Windows Server 2008. Windows Azure supports three types of data storages which are BLOB for general binary data, Table for systematic data and Queue for data passing between webs and programs.

For the programming model in Windows Azure, there are two different roles which are:

- Web Role: The task of web role is to communicate between users and background processes. It can be implemented by dynamic web language, for example, ASP.NET and PHP, etc.
- Worker Role: It is a background process in Windows Azure. Worker Roles grab and execute jobs, and then export the results periodically.

On the other hand, SQL Azure is the Cloud version of SQL server and is built on Windows Azure. Designed for Cloud, SQL Azure only supports part of the functions of SQL server.

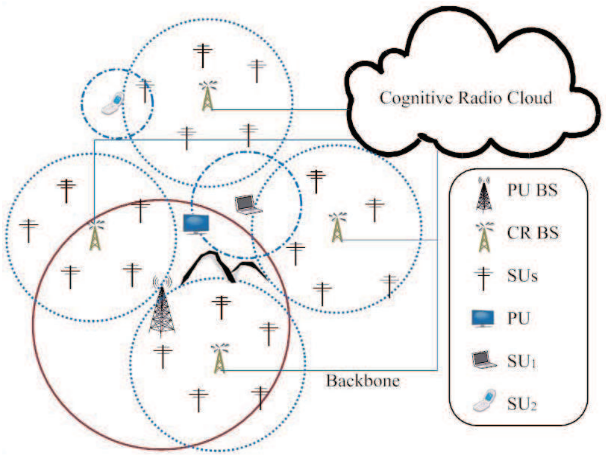

Fig. 1. A conceptual map of the *Cognitive Radio Cloud Network*, which illustrates the geographical relationship among the PUs, PU BSs, CR SUs and CR BSs of the network. As shown in the figure, the received signal of SU$_1$ is blocked by a mountain which makes the SU$_1$ difficult to detect the PU BS by itself. Thus, SU$_1$ will interfere with the PU when it uses the same frequency band to connect to the CR BS. With the CRCN, this shadowing effect can be easily resolved and the frequency band will be allocated to SU$_2$ to enhance the spectra efficiency.

To reconstruct the PPM on Windows Azure, each SU inside the CRCN is assumed equipped with a global positioning system (GPS) device and is able to feed back its location, time and value of the received signal strength indicator (RSSI) to the CRC through the Web Role of Windows Azure as shown in Fig. 2. Each Web Role takes the inputs sent from a CR BS and stores the sensing reports in the input database of SQL Azure. The CRC then partitions the sensing measurements in the input database into blocks according to their associated positions, and maps the data of each block to a Worker Role of Windows Azure to parallelizes the SBL algorithm. A Worker Roles performs the SBL-based SS algorithm with the sensing measurements of each block and stores the reconstructed PPM of PUs of each time slot in the output database of SQL Azure. If a SU wants to access the CRCN, it first sends a request together with its location to the CRC to ask for permission. The CRC will allocate the radio resource to the SU according to the PPM of PUs and the locations of all users both stored in the input and output databases of SQL Azure.

III. The Implementation of the SBL-Based Cooperative SS on CRC

More details about the SBL-based cooperative SS algorithm and how we implement and parallelize the algorithm on the CRC are provided in this Section. A scalable mapping function is also proposed to adjust the block scale of each Worker Role.

A. The SBL-Based Cooperative SS Algorithm

Assume that there are N PUs in an area of $N_p \times N_p$, and M_p CR BSs to collect these PUs' sensing results $\boldsymbol{t} = (t_1, t_2, \cdots, t_N)^T$ and locations $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$, with $\boldsymbol{x}_j \triangleq [x_j, y_j]^T$, and feed back them to the CRC. We select a basis function $\phi_j(\boldsymbol{x}_i) =$

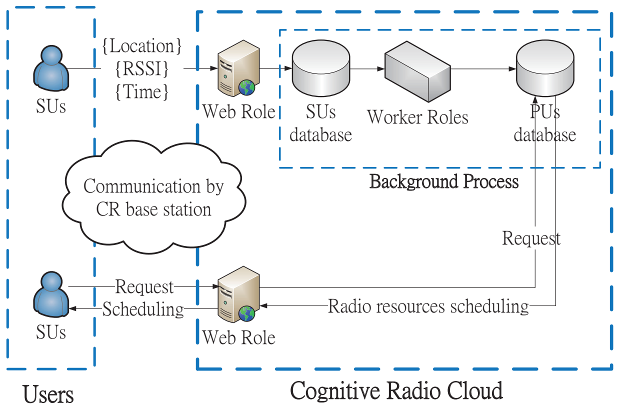

Fig. 2. The *The block diagram of the Cognitive Radio Cloud platform*. The platform includes two dynamic web pages and one database of SQL Azure for SUs and one for PUs. The SBL-based cooperative SS algorithm is operated in Worker Roles and triggered periodically.

$\frac{1}{2s_j}\exp\left\{-\sqrt{(x_{i,x}-\mu_{j,x})^2+(x_{i,y}-\mu_{j,y})^2}/s_j\right\}$ and use the SBL to solve the regression problem of [7]

$$\boldsymbol{t}=\boldsymbol{\Phi w}+\boldsymbol{n} \tag{1}$$

where $\boldsymbol{\Phi}_{N\times M}=[\boldsymbol{\psi}_1(\boldsymbol{X}),\boldsymbol{\psi}_2(\boldsymbol{X}),\cdots,\boldsymbol{\psi}_M(\boldsymbol{X})]$, with $\boldsymbol{\psi}_j(\boldsymbol{X})\triangleq(\phi_j(\boldsymbol{x}_1),\phi_j(\boldsymbol{x}_2),\cdots,\phi_j(\boldsymbol{x}_N))^T$, is the basis matrix determined by the number M, the locations $\boldsymbol{\mu}=(\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\cdots,\boldsymbol{\mu}_M)^T$, with $\boldsymbol{\mu}_j\triangleq[\mu_{j,x},\mu_{j,y}]$, and the power decaying rates $\boldsymbol{s}=(s_1,s_2,\cdots,s_M)^T$ of the bases, and also by the number N and the location vector $\boldsymbol{X}$ of the SUs. The vector $\boldsymbol{w}=(w_1,w_2,\cdots,w_M)^T$ denotes the weighting coefficients of the basis functions, and each of its entries w_j is endowed a *prior* probability $\mathcal{N}(0,\alpha_j^{-1})$. The $\boldsymbol{n}$ denotes the shadowing effect, with each entry being a zero-mean Gaussian random variable (RV) with variance β^{-1}. The SBL iteratively modifies the RVM and estimates the parameters M, α_j^{-1}, β^{-1}, $\boldsymbol{\mu}_j$, and s_j to maximize the marginal likelihood function $p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{\alpha},\beta,\boldsymbol{\mu},\boldsymbol{s},M)$ from sparse measurements.

The SBL can also be viewed as an alternative EM algorithm. In the E-step, the covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{m}$ of the posterior distribution of weighting coefficients $\boldsymbol{w}$ are evaluated by

$$\boldsymbol{\Sigma}=(\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\boldsymbol{A})^{-1}\text{, and }\boldsymbol{m}=\beta\boldsymbol{\Sigma\Phi}^T\boldsymbol{t} \tag{2}$$

where $\boldsymbol{A}\triangleq\text{diag}(\alpha_j)$ is a $M\times M$ diagonal matrix. Consequently, we can obtain the estimated weighting coefficients $\widetilde{\boldsymbol{w}}=\boldsymbol{m}$ and then delete those low-weighted bases according to $\widetilde{\boldsymbol{w}}$. Here we select the bases whose weighting coefficients are larger than a threshold η and count their number to renew the M. For the k-th iteration of SBL, we have

$$M_{(k)}=\sum_{j=1}^{M_{(k-1)}}\mathbf{I}(\widetilde{w}_j\geq\eta) \tag{3}$$

where $\mathbf{I}$ is the indicating function. Since there are two M-steps in SBL and $\boldsymbol{m}$ is mainly adjusted in the first M-step. The bases deleting criterion is only applied in the first EM in which the basis parameters $\boldsymbol{\mu}$, $\boldsymbol{s}$ and M are assumed known. Therefore, the variance parameters α_j^{-1} and β^{-1} are estimated as

$$\alpha_j^{-1}=\frac{m_j^2}{\gamma_j}\text{, and }\beta^{-1}=\frac{\|\boldsymbol{t}-\boldsymbol{\Phi m}\|^2}{N-\sum_{j=1}^M\gamma_j} \tag{4}$$

where $\gamma_j\equiv1-\alpha_j\Sigma_{jj}$, and Σ_{jj} are the diagonal terms of $\boldsymbol{\Sigma}$. The smaller is the α_j^{-1}, the more likely is ϕ_j a redundant basis. Besides, the deleting threshold η in (3) should be set based on the noise variance β^{-1}. In the second M-step, $\boldsymbol{w}$, β^{-1}, and M are assumed known. We infer the basis parameters $\boldsymbol{\mu}$ and $\boldsymbol{s}$ that maximize the likelihood function $p(\boldsymbol{t}|\boldsymbol{\mu},\boldsymbol{s};\boldsymbol{X},M,\beta,\boldsymbol{\alpha})$ by the *gradient descent method*

$$\begin{bmatrix}\mu_{j,x}(k)\\ \mu_{j,y}(k)\\ s_j(k)\end{bmatrix}=\begin{bmatrix}\mu_{j,x}(k-1)\\ \mu_{j,y}(k-1)\\ s_j(k-1)\end{bmatrix}-\delta\begin{bmatrix}\left.\frac{\partial Q}{\partial\mu_{j,x}}\right|_{\mu_{j,x}(k-1)}\\ \left.\frac{\partial Q}{\partial\mu_{j,y}}\right|_{\mu_{j,y}(k-1)}\\ \left.\frac{\partial Q}{\partial s_j}\right|_{s_j(k-1)}\end{bmatrix} \tag{5}$$

where k is the iteration index, $Q\triangleq-\ln p(\boldsymbol{t}|\boldsymbol{\mu},\boldsymbol{s};\boldsymbol{X},M,\beta,\boldsymbol{\alpha})$ and $\delta>0$ is the step size or referred to as the *learning rate*. The details of the iteration process is shown in Table I.

According to the compressive sensing (CS) theorem [8], a signal can be exactly reconstructed when the measurement rate (N/N_p^2) is larger than 0.16. Even though the SBL algorithm does not adopt orthogonal bases and as such does not abide by the CS theorem, the CS theorem still provides for the SBL algorithm a useful reference figure on the measurement rate. In the sequel, we only simulate the cases whose measurement rates are less than 0.15. The estimation errors do not improve significantly when the measurement rates are larger than 0.15, while the complexity will increase dramatically.

TABLE I

The Sparse Bayesian Learning Algrithm
1) Uniformly spread M bases ϕ_j in the area of interest.
2) Initiate the iterations with $\alpha_j=1$, $\beta=1$ and $k=0$, and evaluate the corresponding mean $\boldsymbol{m}$ and covariance $\boldsymbol{\Sigma}$.
3) Let $k=k+1$. Update α^{-1} and β^{-1} and then evaluate $\boldsymbol{m}$, $\boldsymbol{\Sigma}$ and $Q(k)$.
4) Delete the bases whose corresponding weights $\widetilde{w}_j<\eta$ and then renew the M equal to the number of the surviving bases. Renew the matrix $\boldsymbol{\Phi}$ and $\boldsymbol{A}$.
5) Let $k=k+1$. Update $\boldsymbol{\mu}_j=[\mu_{j,x},\mu_{j,y}]$ and s_j and and then evaluate $\boldsymbol{m}$, $\boldsymbol{\Sigma}$ and $Q(k)$. Go to step 6) if $(Q(k)-Q(k-1))/Q(k-1)<0.0001$. Otherwise, repeat this step for L times, then go back to step 3).
6) Output the $\boldsymbol{\mu}_j=[\mu_{j,x},\mu_{j,y}]$ and s_j. Let $\widetilde{M}_p=M$ and $\widetilde{\boldsymbol{w}}=\boldsymbol{m}$.

B. Area Parallelization with a MapReduce-like method

Because the algorithm complexity of the SBL-based SS scheme grows in the third order of the number of sensing measurements, we partition the sensing data of the distributed

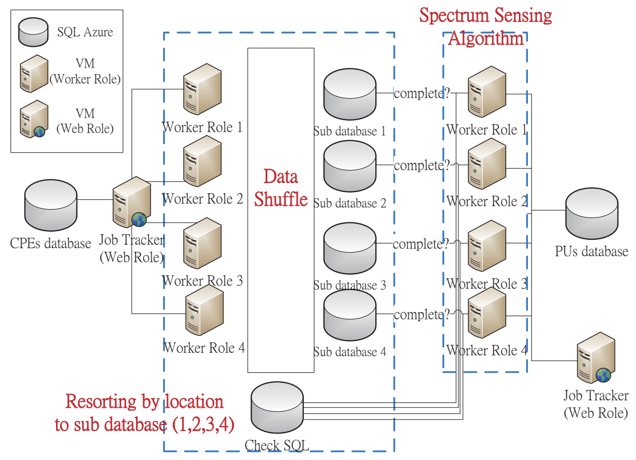

Fig. 3. The flowchart of the parallelized SBL-based cooperative SS algorithm in the background process of Windows Azure. In this example, there are one web role and four worker roles, and each role operates on an individual VM. The web role distributes the sensing data; in contrast, the worker roles execute programs. The detailed execution steps of the algorithm are listed in Table II.

SUs by their locations into blocks to reduce the processing time of the SS algorithm. The data of each block are processed independently by a Worker Role of a VM to execute the SBL-based SS scheme for the block. When the number, locations and the RSSI levels of PUs are obtained, each VM reports the results to a common PUs database.

Under the MapReduce programming model [6], VMs exchange data in a format of $(key, value)$ pair. Applying this concept to our SS problem, we define the time and the location measurements as the Mapper's *input data key* and *output data key*, respectively. For the Reducer, both the *input data key* and the *output data key* are location information. The flow chart is shown in Fig. 3. We note that VMs and SQLs are not guaranteed to be implemented in the same server, thus exchanging data between VMs might become the bottleneck of our implementation. It is a tradeoff between the degree of parallelism and data exchange. The detailed description of this MapReduce Programming Model is shown in Table II.

TABLE II

MapReduce Programming Model
1) Job tracker distributes SUs' data to different Worker Roles according to the chronical order.
2) Worker Role distributes the SUs' data to different sub-databases according to the SUs' locations. Each worker Role renews its $state = 1$ in Check SQL when the distribution is done.
3) Worker Roles check $state$ value in the Check SQL. If all $state$ values are equal to 1, then start to run the spectrum sensing algorithm. Otherwise, check $state$ value periodically.
4) Export the estimation results into PUs' database.

C. *Hierarchical Parallelization*

Although area parallelization can reduce the processing time significantly, the speed improvement is still restricted by the power coverage areas of the PUs, in particular, for a PU like a TV broadcasting station. This is because data processed by a VM should come from an area larger than that covered by the power of a PU to ensure the correctness of the reconstructed PPM of a PU. To lift this fundamental limit on the computational speed of the SS algorithm, one can consider a traditional parallelization method of multi-threading.

Specifically, we consider a hierarchical parallelization structure for the computation of the SBL-based SS algorithm. Measurement data are first partitioned by area into blocks for the algorithm complexity is of the third order of the number of measurements. Each block are handled by one VM with multiple CPU cores. Signal processing within each VM is further parallelized with multi-threading over multiple cores.

In Microsoft's dotNet 4.0, a simple multi-thread instruction of $Parallel.For$ can be used to parallelize computations This is an advantage of Windows Azure. Unlike Hadoop, Windows Azure allows users to define some system-level properties for the different VMs of Web Roles and Worker Roles. Therefore, using multi-threading in VMs with multiple cores on Windows Azure platform can also reduce the communication cost between VMs when only single-thread instructions are allowed in each VM as in typical MapReduce programming model.

IV. Simulation Results

Before we introduce the simulation results, we first give some figures about the Windows Azure Platform. Windows Azure offers different options of VMs whose system parameters are listed in Table III. These options allow us to do a fairer comparison between the speeds and accuracies of different measurement rates.

We consider herein an area of 60×60 with 3 PUs located at $(15, 45)$, $(45, 45)$ and $(15, 45)$, respectively. A baseline SBL-based SS algorithm is performed for this area on Windows Azure using the large instance in Table III. To study the performance of parallelization on the Cloud, we test three types of parallelization methods for the SBL-based SS algorithm. The Type I performs parallelization for the SS algorithm by simple multi-threading using four CPU cores of small instance in one VM of the Worker Role. In comparison, the Type II (in Host) partitions the entire area into four blocks. Each area includes at most one PU located at the same position relative to the baseline example. Data from each block are processed by one CPU core of a VM with 4 cores. In contrast, the Type II (on Cloud) processes data from each block on a VM of a single CPU, *i.e.*, each Worker Role processes the measurement data from an area of 30×30. Finally, the Type III processes data from each block on a VM with 4 CPU cores. As a results, the total number of CPU cores for the Worker Role becomes 16.

The simulation results for different measurement rates (sparsities) are listed in Table IV to VIII and are also shown in Fig. 4 to Fig. 8. Fig. 4 shows that parallelization by partitioning the area is most crucial to the computation of the SS algorithm.

TABLE III
THE COMPUTE INSTANCE SIZE OF WINDOWS AZURE

Computer Instance Size	CPU	RAM	Storage	I/O efficiency
Extra Small	1.0GHz	768 MB	20 GB	Low
Small	1.6GHz	1.75 GB	225 GB	Moderate
Medium	2×1.6GHz	3.5 GB	490 GB	High
Large	4×1.6GHz	7 GB	$1,000$ GB	High
Extra large	8×1.6GHz	14 GB	$2,040$ GB	High

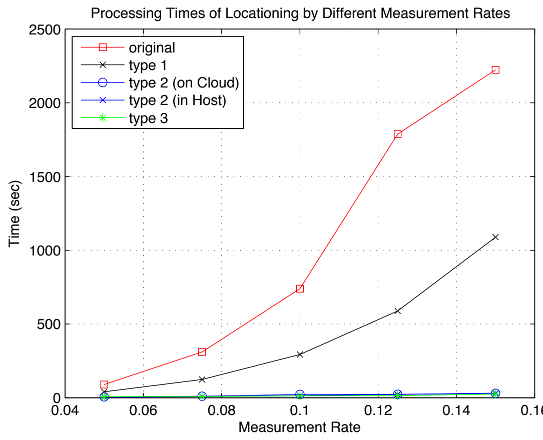

Fig. 4. The average processing times in the CRC platform versus the measurement rates. Only 10 runs are executed for each point.

At the measurement rate of 0.1, Type II (in Host) can improve the processing speed of Type I by 20 times, while Type II (on Cloud) improves the processing speed of Type I around 13 times due to the communication time between the VMs. The computational advantage of Type II results from the matrix inversion involved in the SBL-based SS algorithm since, for the Gaussian-Jordan method, the time complexity grows with the third order of the amount of sensing reports.

TABLE IV
THE SPENDING TIMES (SEC) UNDER DIFFERENT MEASUREMENT RATES

Measurement rates	0.05	0.075	0.1	0.125	0.15
original	89.4	309.9	739.4	1788.4	2223.5
Type I	40.0	124.1	293.2	588.1	1089.0
Type II (on Cloud)	6.3	10.2	22.1	22.9	29.9
Type II (in Host)	3.8	7.8	14.2	22.1	31.2
Type III	6.5	8.9	12.5	14.6	24.5

The hierarchial parallelization algorithm will not effect the complexity, it only reduces the processing time for each area. Nevertheless, this feature is particular useful for TVWS due to the large scale of the power coverage areas of PUs. For the SBL-based SS algorithm, a VM should process sensing data at least from a PU, which prevents from partitioning the area into very small processing blocks.

Table V and VI show the mean squared errors (MSE) of estimation. Table V shows the MSEs in the locationing of the PUs, and Table VI presents the MSEs of the reconstructed PPM. If the radiation power of a PU is 5KW, one scale in our simulations corresponds to 15km. As a result, the MSE in locationing is around 1.8 km when measurement rate is at 0.1. More results on MSEs are presented in Fig.5 and 6.

Table VII and VIII show the missing ratios and the false alarm ratios of the different types of the implementation

TABLE V
THE MSES OF LOCATION UNDER DIFFERENT MEASUREMENT RATES

Measurement rates	0.05	0.075	0.1	0.125	0.15
original	0.138	0.091	0.075	0.060	0.050
Type I	0.180	0.09	0.068	0.057	0.048
Type II	0.414	0.276	0.212	0.162	0.167
Type III	0.510	0.306	0.217	0.190	0.158

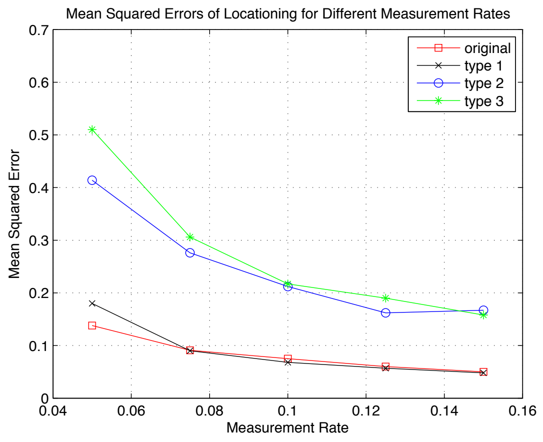

Fig. 5. The average mean squared errors of the SBL-based SS algorithm versus the measurement rates. The result doesn't include the false alarms and missing detection cases.

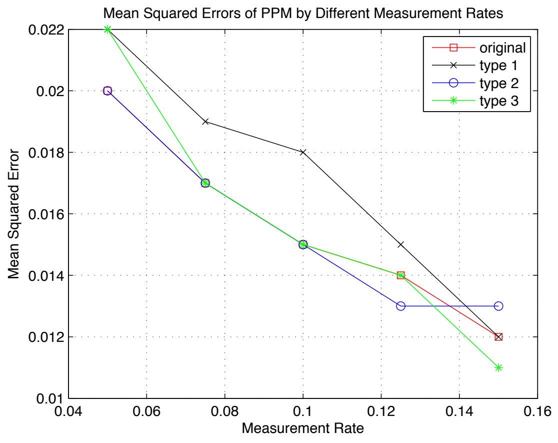

Fig. 6. The average mean squared errors of the SBL-based SS algorithm versus the measurement rates. The result doesn't include the false alarms and missing detection cases.

TABLE VI
THE MSES OF PPM UNDER DIFFERENT MEASUREMENT RATES

Measurement rates	0.05	0.075	0.1	0.125	0.15
original	0.020	0.017	0.015	0.014	0.012
Type I	0.022	0.019	0.018	0.015	0.012
Type II	0.020	0.017	0.015	0.013	0.013
Type III	0.022	0.017	0.015	0.014	0.011

TABLE VII
THE MISSING RATIOS UNDER DIFFERENT MEASUREMENT RATES

Measurement rates	0.05	0.075	0.1	0.125	0.15
original	0.09	0.04	0.04	0.04	0.04
Type I	0.108	0.02	0.011	0	0
Type II	0.11	0.04	0.02	0	0
Type III	0.13	0.03	0.01	0	0

methods for the SBL-based SS algorithm. For the Type II, it appears to have a higher false alarm ratio in an area without PU. To resolve this problem, we set a threshold for the estimated power. With this mechanism, we can find for all of the proposed algorithms that they exhibit consistent results either in the false alarm ratios or the missing ratios.

V. DISCUSSIONS AND FUTURE WORKS

A CRCN was proposed for cooperative SS in TVWS. Based on the SBL algorithm, a cooperative SS algorithm was tested on Microsoft's Windows Azure Cloud platform. Making use of the multi-threading features of the Windows Azure platform, a hierarchial parallelization method was proposed to improve the processing speed of the SBL-based SS algorithm on the Cloud. According to our simulation studies, the performance of the SS algorithm can be greatly improved with the parallel computing capacity and the MapReduce-like programming model of the Cloud. Under the framework of CRCN, more advanced algorithms or ideas on cooperative SS or spectrum resource scheduling can be tested for CR in TVWS.

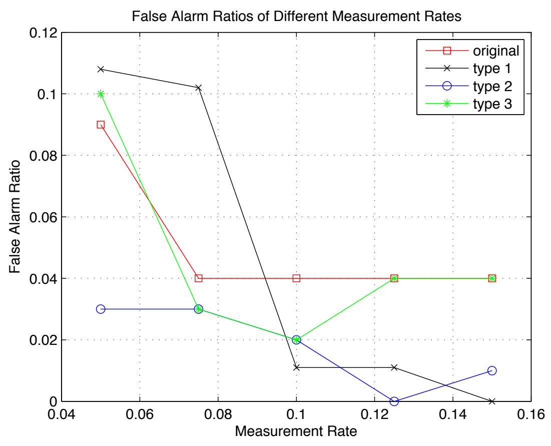

Fig. 7. The false alarm ratios of the SBL-based SS algorithm versus the measurement rates. The power of PU is 30 and the noise is zero mean with variance equal to 2 in this simulation case. Each point runs for 100 times.

TABLE VIII
THE FALSE ALARM RATIOS UNDER DIFFERENT MEASUREMENT RATES

Measurement rates	0.05	0.075	0.1	0.125	0.15
original	0.12	0.04	0.01	0	0
Type I	0.108	0.102	0.011	0.011	0
Type II	0.03	0.03	0.02	0	0.01
Type III	0.10	0.03	0.02	0.04	0.04

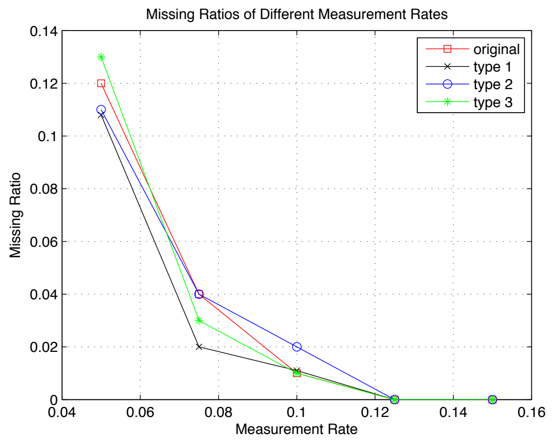

Fig. 8. The missing ratios of the SBL-based SS algorithm versus the measurement rates.The power of PU is 30 and the noise is zero mean with variance equal to 2 in this simulation case. Each point runs for 100 times.

REFERENCES

[1] J. Mitola and G. Q. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, Aug. 1999.
[2] M. A. McHenry, *NSF spectrum occupancy measurements project summary*, shared spectrum co. report, Aug. 2005.
[3] FCC, *Second report and order and memorandum opinion and order*, FCC 08-260, Nov. 2008.
[4] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1.
[5] H. Harada, H. Murakami, K. Ishizu, S. Filin, Y. Saito, H. N. Tran, G. Miyamoto, M. Hasegawa, Y. Murata, and S. Kato, "A Software Defined Cognitive Radio System Cognitive Wireless Clouds ," in *IEEE proceeding of IEEE Global Communications Conference (GLOBECOM)*. Washington DC, USA.
[6] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[7] D.-H. Tina Huang, S.-H. Wu, and P.-H. Wang, "Cooperative Spectrum Sensing and Locationing: A Sparse Bayesian Learning Approach," in *Proc. IEEE GLOBECOM*. Miami, USA, Dec. 2010.
[8] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles : Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.