Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 8 (4): 304-313, 2014
ISSN 1819-4311 / DOI: 10.3923/jse.2014.304.313

An OpenFlow Switch Model using Kepler GPU

Li Xiuhai
Shandong Yingcai University, Jinan Shandong, 250104, China

ABSTRACT

OpenFlow makes up for traditional network, but network session identification is inefficient and packet forwarding path selection is poor. Focusing on forwarding path and matching; we propose OGMD model which combin GPU with biological sequence algorithms and machine learning methods to accelerate matching speed and to improve network environment, we also proposed matching algorithm and path selection algorithm. Experiments show matching algorithm gives a speedup of 325 and path selection algorithm makes link loss rate less than 5%, with an average decline 69.35% and network delay less than 20 msec, average fell 63.28%.

Key words: OpenFlow, GPU computation, machine learning, Kepler GPU

INTRODUCTION

With the development of network technology, innovation of existing network becomes difficult. In order to meet needs of next generation network, Greenberg *et al*. (2005) and McKeown *et al*. (2008) proposed a novel network named OpenFlow which supports session management, could make up for deficiencies of traditional network and provides higher quality service and more service types. OpenFlow installed on switch which has a session table, could easily add or delete items and effectively determine session type. Session processed by rules. However, how to quickly and efficiently identify network sessions and select a high-quality forwarding path are current hot problems. For these, Moore and Papagiannaki (2005), proposed identification method from packet load, in order to solve port identification inefficiency, however, scanning packet load is not conducive to privacy protection. Now-a-days, there are two ways could quickly identify, one is using optimization algorithms, such as biological sequences; another is using dedicated hardware devices, such as FPGA, specialized hardware costs too much, is not conducive to popularity. So, using GPU to compute is an alternative method. Session forwarding based on IP address and network status, selecting appropriate path to forward. Previously forwarding based on matching session table, choosing right path, but not all paths optimized. OpenFlow provides more services, how rationally and efficiently use these functions and improve forwarding quality is a serious problem. Current researches take advantage of machine learning methods, such as multi-layer perceptron, support vector machines, neural networks, train historical information and find optimal path.

In this study, focusing on quickly and efficiently identify network session and high-quality forwarding packets in OpenFlow, we propose combining GPU parallel computing ability with biological sequence to enhance network session identification speed and accuracy, at the same time, using machine learning method to select appropriate packet forwarding path.

BIOLOGICAL SEQUENCE METHOD

Biological sequence method is mainly used for amino acids similarity and homology matching in proteins, its main approaches are Needleman and Wunsch (1970) proposed Needleman-Wunsch

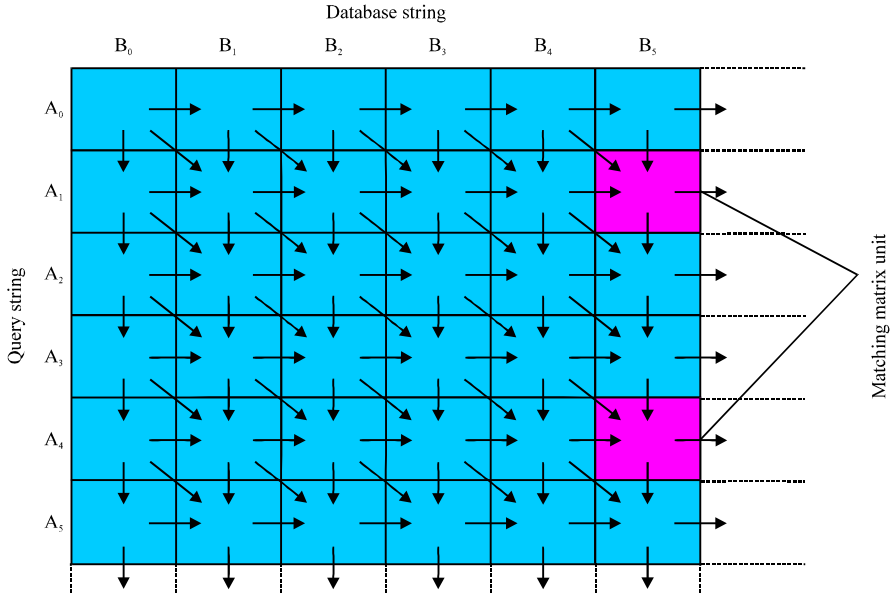

Fig. 1: Smith-Waterman algorithm data correlation

algorithm (Guillou *et al.*, 2013), Smith and Waterman (1981) proposed Smith-Waterman algorithm (Liu *et al.*, 2013), Aji AM proposed WaveFront algorithm (Aji *et al.*, 2008) in 2008 and Martins proposed TWF algorithm in 2001 (Martins *et al.*, 2001), these algrithms have a trmendous impact on addressing similarity comparision.

Needleman-Wunsch algorithm (Guillou *et al.*, 2013) was first applied to solve the similarity of protein amino acid sequence and determine whether were homologous proteins. In the begining, initializing a matrix as global alignment, in Eq. 2, vertical and horizontal axes represent needed sequences. Using recursive calculate H_{ij} in Eq. 1. Where, $S(s_i, b_j)$ is the similarity of sequences a_i and b_j, w_k is gap penalties. At last, getting the highest score by calculating two sequences and two aligned sequences matching results through back matrix but this algorithm is adapted to comparision sequences in which length and content are similar, is not suited sequence piece comparison:

$$H_{ij} = \max\left\{\begin{array}{c} H_{i-1,j-1} + S(a_i, b_j) \\ \max\limits_{1\le k\le i}\left(H_{i-k,j-1} - w_k + S(a_i, b_j)\right) \\ \max\limits_{1\le k\le j}\left(H_{i-k,j-k} - w_k + S(a_i, b_j)\right) \end{array}\right\} \quad (1)$$

$$H_{i0} = H_{0j} = 0 \quad (2)$$

Smith-Waterman algorithm (Liu *et al.*, 2013) was also applied to similarity comparision, compare with Needleman-Wunsch, this algorithm using local matching, Fig. 1 shows its data correlation. Firstly initializing matrix for local alignments, as Eq. 4. Vertical and horizontal axes represent aligned sequences. Using recursive calculate H_{ij} in Eq. 3. If one unit is negative, using

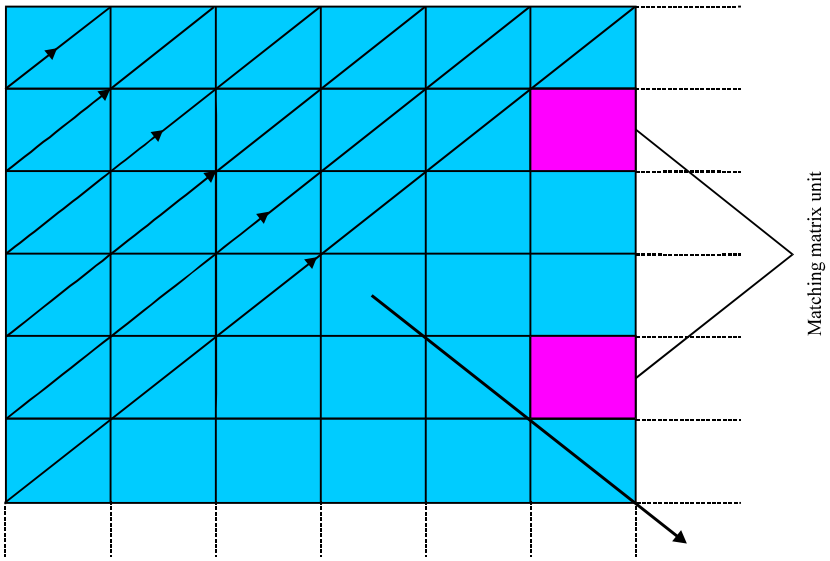

Fig. 2: Wavefront algorithm

0 alternatives, way back with the literature (Guillou *et al*., 2013) algorithm is the same. Literature (Guillou *et al*., 2013; Liu *et al*., 2013) algorithms' time and space complexity are O(mn), where, m and n are the length of sequence:

$$H_{ij} = \max\begin{Bmatrix} H_{i-1,j-1} + S(a_i, b_j) \\ \max\limits_{1\le k\le i}(H_{i-k,j} - w_k) \\ \max\limits_{1\le k\le j}(H_{i,j-k} - w_k) \\ 0 \end{Bmatrix} \quad (3)$$

$$H_{i0} = H_{0j} = 0 \quad (4)$$

Wavefonrt algorithm (Aji *et al*., 2008) changes computation method, its diagonally iteration begining from top left corner, computing all of the anti-diagonal matrix elements, as shown in Fig. 2. Because of independence of anti-diagonal matrix elements, so could taken parallel computing. However, if using literature (Liu *et al*., 2013) algorithm would cause a large number of communications, that is due to three times element value copy.

TWF algorithm can effectively solve literature (Aji *et al*., 2008) communication overhead problem. Martins *et al*. (2001) implemented multi-threaded parallel computing with TWF algorithms. TWF merges a plurality of matrices into a block, in order to reduce communication spending. Block merging that takes R×C matrix elements into a complete block, block computation needs copy left R elements, C elements above and one upper-left element. Block constituted by R×C elements, each iteration calculates anti-diagonal row blocks, Fig. 3 shows TWF algorithm.

In short, biological sequence method could improve matching acuracy which is particularly important for OpenFlow switch but how efficiently and accurately use biological sequences method to enhance network session matching is important.

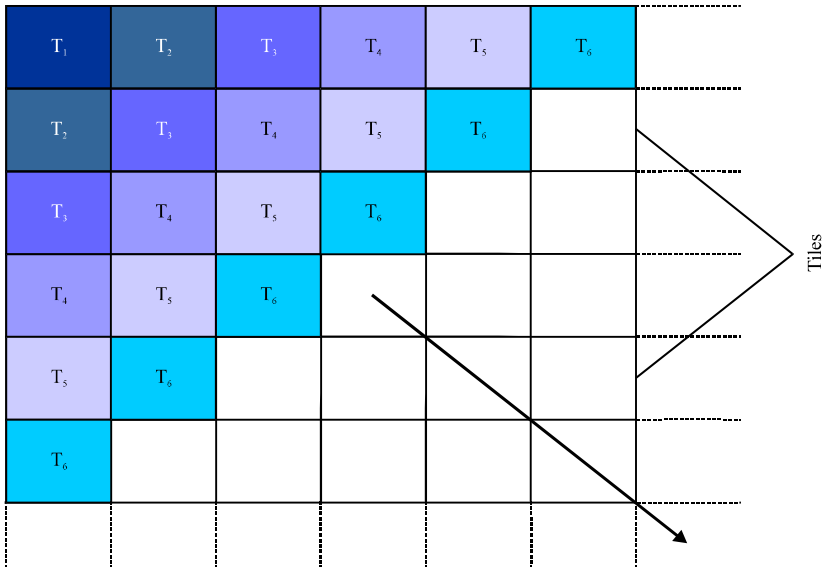

Fig. 3: TWF algorithm

Machine learing method: Machine learning through developing existing knowledge, provide basis in the future, Simon and Leijonhufvud (2012) thought, learning is an adaptive changing process, allows systems to perform same or similar tasks more effectively. Binford *et al.* (2013), Herskovits and Cooper (2013) thought learning is experience which could organize or correct things. Expert system developers think that learning is a process of acquiring knowledge. In short,

- Learning is an external act
- Learning is a intrinsic process
- Learning from practical engineering knowledge

The main machine learning methods are Support Vector Machine (SVM), Artificial Neural Networks (ANN), Fuzzy System (FS), combining these researches with this study is to train existing network sessions and rules, applying learning patterns on network sessions forwarding to provide better and more services. However, forwarding sessions are massive, high concurrency and high similarity. For these characteristics, current studies take Support Vector Machine (SVM) (Maji *et al.*, 2013) and so on.

Cortes and Vapnik (1995) used SVM analysis small samples, nonlinear and high dimensional pattern recognition (Maji *et al.*, 2013). Este *et al.* (2009) proposed using SVM to classify network data stream, although this method is not sensitive to small size sample, but depends on sample data acuracy. Li *et al.* (2007) proposed using SVM method to solve 7 types of protocls, based on feture extraction, finding best combination, its accuracy more than 99.4%. This method requires samples screening. Although these methods present innovative approachs but they are difficult to implement in massive data training, mainly because SVM need to solve quadratic programming for supportting vector and quadratic programming required to calculate n matrix, its time and space complexity is high. Platt (1998) proposed SMO algorithm (Crammer *et al.*, 2013), Joachims (1999) proposed SVM (Chang and Lin, 2011), Burges (1998) proposed PCGC (Hastie *et al.*, 2009) and Mangasarian and Musicant (1999) proposed SOR algorithm (Carrizosa and Romero Morales, 2013) solving secondary plan for improving n-order matrix.

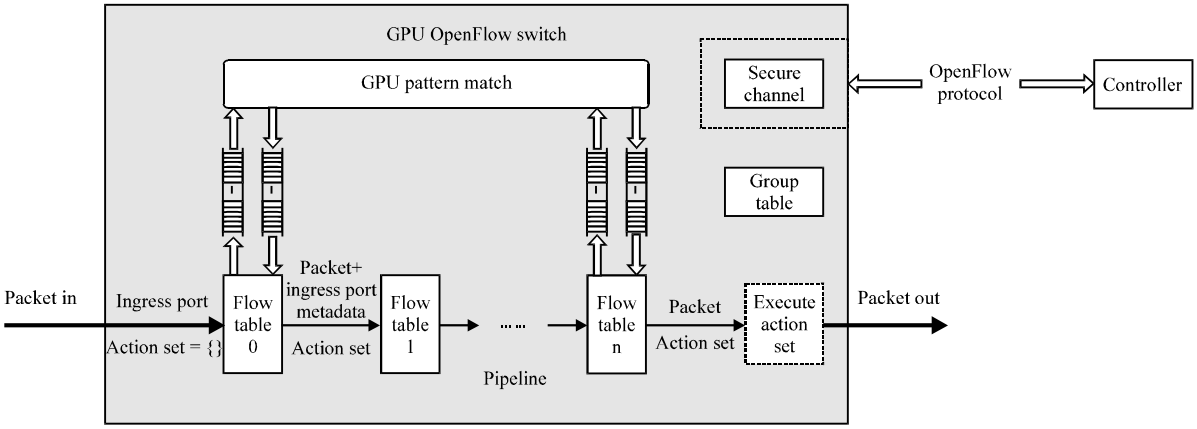

Fig. 4: Kepler OpenFlow switch architecture

In summary, although machine learning methods provide basis for practical work, regardless Kmeans or SVM exists more or less problems, mainly around sample pre-selection and data size, support vector machine requiring samples high accuracy, as well as restrictions on size. K means method suits for data set less than 10 K.

OGMD OPENFLOW SWITCH MASSIVE NETWORK DATA ANALYSIS MODEL

OGMD OpenFlow switch model is consisted by two parts, one is session matching, the other is session forwarding, matching using biological sequence method based on GPU, forwarding using session ranking algorithm.

OGMD OpenFlow switch massive network data analysis model architecture: OGMD OpenFlow switch effectively integrates GPU into OpenFlow Switch, in order to improve matching accuracy and efficiency, network packets coming from interface, Flow Table module will handle incoming data and then copy these data to GPU Pattern Match which complete rule matching and label network session. OGMD OpenFlow Switch processing framework shown in Fig. 4.

OGMD OpenFlow switch network session matching algorithm: OGMD OpenFlow switch network session matching algorithm based on GPU implement efficient network session matching, achieve wire-speed processing, this section describes concepts and networking session matching algorithm.

- **Related concepts:**
  - Kernel, referring to CUDA GPU running function
  - Thread blocks, refer to thread set, each thread independently run in thread block. Assuming one thread block N threads
  - SM, scheduling concurrent threads in hardware, scheduling overhead does not exist, threads running on SM, each SM could run multi-threaded block

- **OGMD OpenFlow switch network session matching algorithm:** In previous section, we mainly introduce biological sequence matching algorithm, although compare TWF algorithm with literature (Aji *et al*., 2008) algorithm the communication overhead has decreased, but still not take advantage of GPU's parallel processing capabilities. The main reason is that only

one thread involved in calculation, in order to enhance thread utilization of system resources, we propose KTWF algorithm which puts tiles into diamond-shaped pieces, all threads involved in calculation in thread block, using kepler architecture GPU. Tile number: Thread_N, each thread block is responsible for calculating a row tiles, the same color tiles represent different threads parallel execute in the same cycle; arithmetic sequence is consistent with TWF algorithm

Algorithm 1: OGMD OpenFlow switch network session match

Input: Session_Packet_Data

Output: Max_Match

- Idle = Kepler G Copy Text Share (Session Data)
  /*Copy data from Global to Texture, return copy results*/
- Matri x = Kepler Session (Idle, Session Data)
  /*thread block compute tile, save results at Matrix*/
- Session = Kepler Copy DG (Matrix, Session Data)
  /*thread block compute last row of tile and copy data to Global*/
- Max_Match = Kepler Tree C (Session)
  /*get max matching date*/

Because network session matching needs thread blocks parallel computing and atomic operations will affect efficiency. In order to reduce the number of atomic operations, this study selects tree algorithm to improve efficiency:

- Thread keep maximum matching value of matrix elements
- After completion calculation, using tree algorithm to obtain maximum tile
- Compare tile with global maximum atomic, complete the newest updating

OGMD OpenFlow switch network session forwarding algorithm:

- **Related concepts:** During OGMD OpenFlow switch network session forwarding algorithm performing. At first retrieval the most relevant network path, retrieval returns two values, one is authority value which is the center value of all session paths; another is center value refers to network path points authority value. Algorithm iterative steps: update authority value and central value, the center and authority value calculation as shown in Algorithm 2

Update authority value rule: ∀P, Update authority value auth(p) as shown in Eq. 5:

$$\text{auth}(p) = \int \frac{\sum_{i=1}^{n} \text{hub}(i)}{n} \tag{5}$$

Update center value rule: ∀p, Update center value hub(p) as shown in Eq. 6:

$$\text{hub}(p) = \int \frac{\sum_{i=1}^{n} \text{auth}(i)}{n} \tag{6}$$

where, n is network session number and i is network session p selected path.

Algorithm 2: Node auth and hub value calculate algorithm

Input: Node

Output: Auth(p), hub(p)

- Init ∀p, auth(p) = 1 and hub(p) = 1 //update auth and hub
- Auth_rule (p)//update auth rules
- Hub_rule (p)// update hub rules
- Repeat Step 2

OGMD OpenFlow switch networksession forwarding algorithm: At the beginning of execution, ∀p, auth(p) = 1 and hub(p) = 1, assuming two types updating rules: Update authority value rule and Update center value rule. In order to calculate each node two values, you need to repeat the iterative update rules. Steps show in Algorithm 3.

Algorithm 3: OGMD session caculate algorithm

Input: Sessions

Output: auth, hub values of Sessions

- G←Session set
- IntiSession (G)//Init G
- Hub_Auth_Update (G)//Update Algorithm
- For 1 to k //Start k steps of Algorithm
- norm←0
- For ∀Session P??G//Update all Auth Value
- p.auth←0
- Session_In (Session p)//p.in point to G
- norm+ = square (p.auth)
- End for
- norm = aqrt(norm)
- End for
- for ∀Session p∈G
- p.auth∀p.auth/norm
- End for
- norm←0
- For ∀Session p∈G
- p.hub←0
- For Session_Out (Session p)//p.out points to G
- norm+ = square (p.hub)
- End for
- norm = sqrt (norm)
- End for
- For ∀Session p∈G
- p.auth∀p.auth/norm
- End for
- End Hub_Auth_Update

OGMD OPENFLOW SWITCH EXPERIMENT

OGMD OpenFlow Switch experiment use server equipped with 8 i7 processor; 128 GB memory; 2 NVidia Tesla K20 graphics, network card chosen two pairs fiber ports 10000 Mbps card, peak

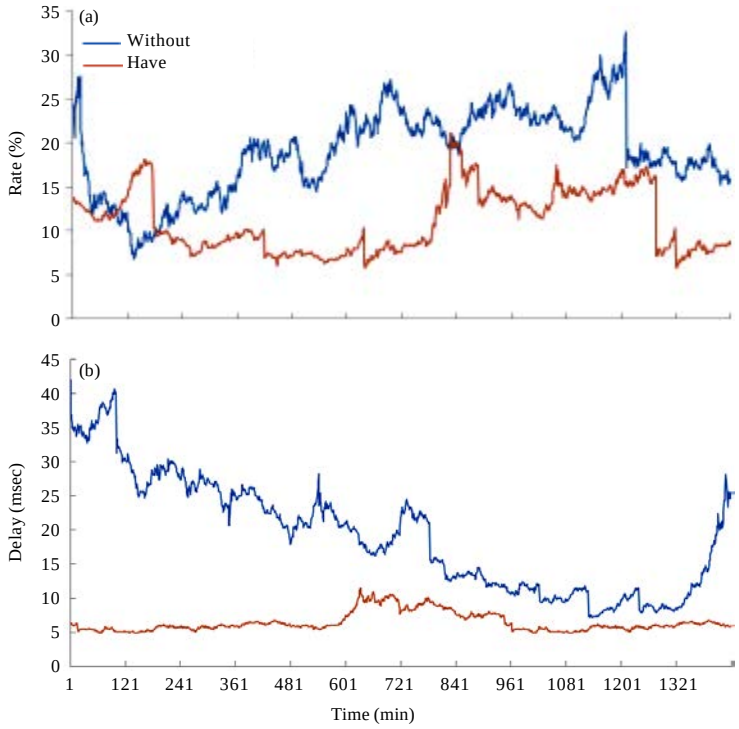

Fig. 5(a-b): OpenFlow network forwarding algorithm. (a) Network packet loss and (b) Network time delay

transfer rate up to 40000 Mbps; install Ubuntu OS 64 Bit operating system. Test data collects from network backbone real-time data; its size is 1357 GB. Flow table capacity is 370,254 items. Experiments mainly test network session matching efficiency, comparative testing KTWF algorithms are in CPU and GPU environment speedup. Speedup tests flow table capacity from 50000-250,000, data size from 10-100 GB; forwarding algorithm test mainly focus on link state, whether is an appropriate path or existing congestion, through packet delay and loss for analysis.

OGMD OpenFlow switch network session matching algorithm: The OGMD OpenFlow Switch network session matching algorithm testing Needleman-Wunsch, WaveFront, Smith-Waterman, TWF and KTWF algorithm at CPU, GPU runtime and speedup, GPU running time as shown in Table 1, laboratory tests indicate that acceleration KTWF algorithm closer than 325, much better than other algorithms and can meet massive data network needs.

OGMD OpenFlow switch network session forwarding algorithm: Comparison with whether using algorithm link status, test link bandwidth is 1 Gbps. Packet loss as shown in Fig. 5a, network delay as shown in Fig. 5b. Experiments show that using forwarding algorithm packet loss rate and delay appear overall downward trend, average, respectively dropped 69.35 and 63.28%, computation as shown in Eq. 7:

Table 1: Matching algirithm GPU runtime

OpenFlow (GB)	OpenFlow table: 50000 (Item) CPU	NW*	WF*	SW*	TWF	KT	150000 (Item) CPU	NW*	WF*	SW*	TWF	GP*	250000 (Item) CPU	NW*	WF*	SW*	TWF	GP*
40	17.54	4.496	4.290	3.273	3.149	1.065	75.85	12.76	9.407	7.821	8.574	3.209	226.9	20.69	18.58	15.35	10.72	4.305
70	34.03	8.291	7.027	6.003	5.300	2.024	121.7	18.87	12.72	9.734	10.10	4.686	167.2	25.53	22.79	20.16	15.12	6.792
100	56.36	12.66	11.08	10.67	7.491	3.196	255.4	22.44	14.21	12.29	16.43	5.649	307.4	30.00	26.63	26.43	25.77	7.265

NW*: Needleman-Wunsch, WF*: WaveFront, SW*: Smith-Waterman, KT: KTWF

$$AR = \frac{\int_i^n \frac{y_i^{Without} - y_i^{Have}}{y_i^{Without}}}{n} \quad (7)$$

where, i is total numbers, $y_i^{without}$ is data without adopting algorithmm, y_i^{have} is data adopting algorithm.

CONCLUSION

This study investigates biological sequence algorithm and GPU, takes both into OpenFlow network session matching, using machine learning method to study network session properties, forwarding sessions to appropriate network link. Focusing on forwarding path and matching, we propose OGMD model which combining GPU with biological sequence algorithms and machine learning methods to accelerate matching speed and improve network environment; we also proposed matching algorithm and path selection algorithm. Experiments show matching algorithm gives a speedup 325 and path selection algorithm makes link loss rate less than 5%, with an average decline 69.35% and network delay less than 20 msec, average fell 63.28%.

ACKNOWLEDGMENTS

Thanks to National Supercomputer Center (Tianjin) and National Internet Information CITIC providing data and technical support for this article.

REFERENCES

Aji, A.M., W.C. Feng, F. Blagojevic and D.S. Nikolopoulos, 2008. Cell-SWat: modeling and scheduling wavefront computations on the cell broadband engine. Proceedings of the 5th Conference on Computing Frontiers, May 5-7, 2008, Ischia, Italy, pp: 13-22.

Binford, T.O., T.S. Levitt and W.B. Mann, 2013. Bayesian inference in model-based machine vision. http://arxiv.org/abs/1304.2720.

Burges, C.J.C., 1998. A tutorial on support vector machines for pattern recognition. Data Mining Knowl. Discov., 2: 121-167.

Carrizosa, E. and D.R. Morales, 2013. Supervised classification and mathematical optimization. Comput. Oper. Res., 40: 150-165.

Chang, C.C. and C.J. Lin, 2011. LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol., 2: 1-39.

Cortes, C. and V. Vapnik, 1995. Support vector networks. Mach. Learn., 20: 273-297.

Crammer, K., A. Kulesza and M. Dredze, 2013. Adaptive regularization of weight vectors. Mach. Learn., 91: 155-187.

Este, A., F. Gringoli and L. Salgarelli, 2009. Support vector machines for TCP traffic classification. Comput. Networks, 53: 2476-2490.

Greenberg, A., G. Hjalmtysson, D.A. Maltz, A. Myers and J. Rexford *et al.*, 2005. A clean slate 4D approach to network control and management. ACM SIGCOMM Comput. Commun. Rev., 35: 41-54.

Guillou, L., D. Bachar, S. Audic, D. Bass and C. Berney *et al.*, 2013. The protist ribosomal reference database (PR2): A catalog of unicellular eukaryote small sub-unit rRNA sequences with curated taxonomy. Nucl. Acids Res., 41: D597-D604.

Hastie, T., R. Tibshirani, J. Friedman, T. Hastie, J. Friedman and R. Tibshirani, 2009. The Elements of Statistical Learning. Springer, New York, ISBN-13: 9780387848587.

Herskovits, E.H. and G.F. Cooper, 2013. Kutato: An entropy-driven system for construction of probabilistic expert systems from databases. http://arxiv.org/abs/1304.1088.

Joachims, T., 1999. Making Large-Scale SVM Learning Practical. In: Advances in Kernel Methods-Support Vector Learning, Scholkopf, B., C.J.C. Burges and A.J. Smola (Eds.)., MIT Press, Cambridge, MA., ISBN-10: 0-262-19416-3, pp: 169-184.

Li, Z., R. Yuan and X. Guan, 2007. Accurate classification of the internet traffic based on the svm method. Proceedings of the IEEE International Conference on Communications, June 24-28, 2007, Glasgow, pp: 1373-1378.

Liu, Y., A. Wirawan and B. Schmidt, 2013. CUDASW++ 3.0: Accelerating smith-waterman protein database search by coupling CPU and GPU SIMD instructions. BMC Bioinf., Vol. 14. 10.1186/1471-2105-14-117

Maji, S., A.C. Berg and J. Malik, 2013. Efficient classification for additive kernel SVMs. IEEE Trans. Pattern Anal. Mach. Intell., 35: 66-77.

Mangasarian, O.L. and D.R. Musicant, 1999. Successive overrelaxation for support vector machines. IEEE Trans. Neural Networks, 10: 1032-1037.

Martins, W.S., J.B. del Cuvillo, F.J. Useche, K.B. Theobald and G.R. Gao, 2001. A multithreaded parallel implementation of a dynamic programming algorithm for sequence comparison. Pac. Symp. Biocomputing, 6: 311-322.

McKeown, N., T. Anderson, H. Balakrishnan, G. Parulkar and L. Peterson *et al.*, 2008. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev., 38: 69-74.

Moore, A.W. and K. Papagiannaki, 2005. Toward the accurate identification of network applications. Proceedings of the 6th International Workshop on Passive and Active Network Measurement, March 31-1 April, 2005, Boston, MA., USA., pp: 41-54.

Needleman, S.B. and C.D. Wunsch, 1970. A general method applicable to the search for similarities in the amino acid sequences of two proteins. J. Mol. Biol., 48: 443-453.

Platt, J., 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. http://research.microsoft.com/en-us/um/people/jplatt/smotr.pdf.

Simon, H.A. and A. Leijonhufvud, 2012. Appendix 3: Herbert Simon's Letters Regarding Computable Economics. In: Computable Foundations for Economics, Velupillai, K.V. (Ed.). Routledge, Pittsburgh, PA., ISBN-13: 9781134253364, pp: 407-410.

Smith, T.F. and M.S. Waterman, 1981. Identification of commonmolecular subsequences. J. Mol. Biol., 147: 195-197.