PARALLEL DATA MINING TECHNIQUES ON GRAPHICS PROCESSING FOR MULTIVARIATE TIME SERIES

[1]S.Baskaran, [2]S.Gandhi,

[1]Head,Dept.of.Computer science, Tamil University (Established by the Govt.of.Tamilnadu), Thanjavur-613010.

[2]Research Scholar, Dept.of.Computer Science, Tamil University, Thanjavur-613010.

ABSTRACT

The traditional data processing algorithms add consecutive manner that will increase their time of execution. These algorithms ought to use the data processing capabilities of the trendy GPUs to execute parallel programs with efficiency. thus a parallel process} rule ought to be enforced that may utilize the processing power of GPUs to hurry up the execution. when put next with the straightforward set mining drawback and string mining drawback, the hierarchi- cal structure of consecutive pattern mining (due to the requirement to contemplate frequent subsets among every itemset, also as order among itemsets) and therefore the ensuing massive permutation area makes SPM extraordinarily costly on typical pro-cessor architectures. HAC estimation for long and high-dimensional statistic is computationally costly. This paper describes a unique pipeline-friendly HAC estimation rule derived from a mathematical specification, by applying transformations to eliminate conditionals, to parallelise arithmetic, and to push information apply in computation. we have a tendency to then develop a fully-pipelined hardware design supported the planned rule. This design is shown to be economical and ascendible from each theoretical and empirical views. Experimental results show that AN FPGA-based implementation of the planned design is up to 111 times quicker than AN optimised processor implementation with one core, and fourteen times quicker than a processor with eight cores.

Keywords: Parallel Computing , CUDA , Data mining , Classification , Clustering , GPU Databases.

I. INTRODUCTION

THE traditional data processing algorithms add successive manner that will increase their time of execution. With data processing in multicore systems, just one core will pro-cessing whereas alternative cores stay idle. A se-quential data processing formula handling massive knowledge sets would doubtless take an oversized quantity of your time. data {processing} is that the process of finding patterns in massive databases and is additionally called information discovery.

Now-a-days ,as computations and datasets ar growing, quick algorithms ar gaining impor-tance. In recent years, the data on the market is growly enormously. therefore the info mining algorithms ought to use the data processing capabilities for execution of parallel programs in associate degree economical manner. additional memory is re-quired thanks to use of multiple processors. To use the resources to their fullest ,parallel com-puting techniques ar essential. The motivation behind usage of correspondence is low power still as low consumption. It additionally provides quicker execution in comparison to serial techniques. victimisation parallel setting these data processing algorithms is used for pattern discovery.

to hurry up the execution , these paral-lel algorithms use the process power of GPUs(Graphics process Unit). the benefits of parallel data processing ar data processing of information, less execution time and higher resource utilization. This shows that on several domains parallel techniques ar robust.

whereas period systems will typically have the benefit of the speed and ease of hardware implementations, hardware acceleration of your time series process isn't a well-studied topic. To the most effective of our information, though there's recent analysis on fast pattern matching in statistic, our work is that the initial to use reconfigurable computing to statistic analysis. Our key contributions ar as follows.

We derive a pipeline-friendly HAC estimation formula by playing mathematical transformations. This formula exploits the process power of the hardware platform with conditional-free logic and parallelised arithmetic. Moreover, it avoids memory bottlenecks with a strong knowledge employ theme. we have a tendency to map the projected formula to a fully-pipelined hardware design by customising on-chip memory to be a first-in-first-out buffer.

This design takes full advantage of the pipeline-friendly options of our projected formula in a chic means. we have a tendency to implement our style during a industrial FPGA

acceleration platform. With measure and experimental results, we have a tendency to demonstrate the performance and measurability of our hardware style, which might be up to fourteen times quicker than associate degree 8-core electronic equipment implementation.

GENERALIZED successive PATTERN FRAMEWORK

The GSP methodology is predicated on the downward-closure prop-erty and represents the dataset during a horizontal format. The downward-closure property suggests that all the subsequences of a frequent sequence are frequent associate degreed so for an in-frequent sequence, all its supersequences should even be infre-quent. In GSP, candidates of (k+1)-sequences ar generated from renowned frequent k-sequences by adding an additional pos-sible frequent item. The mining begins at 1-sequence and therefore the size of candidate sequences will increase by one with every pass. In every pass, the GSP formula has 2 major oper- ations: 1) Candidate Generation: generating candidates of frequent (k+1)-sequences from renowned frequent k-sequences 2) Matching and Counting: Matching candidate sequences and enumeration support.

NVIDIA CUDA ARCHITECTURE[5]

CUDA is associate degree computer programme Interface(API) created by NVIDIA that provides a platform for parallel computing. It permits general purpose computing on Graphics process Unit(GPU). CUDA provides access to parallel process components and therefore the virtual instruction set. It additionally features a unified computer storage. CUDA will work with programming languages like C,C++ and algebraic language. CUDA is compatible with all commonplace operative systems.
GPU could be a specialised processor that works on high resolution tasks like 3D graphics. GPU permits manipulation of enormous block of information quicker than electronic equipment as GPU is evolution of parallel multicore systems. GPU design hides latency from computation.

A CUDA application can run serial code on host i.e {cpu|central process unit|CPU|C.P.U.|central processor|processor|mainframe|electronic equipment|hardware|computer hardware} whereas the parallel code on GPU threads through multiple processing components. GPU executes parallel portion as Kernel. Kernel could be a operate dead on GPU for the asking of host(CPU) as associate degree array of threads that executes in parallel through completely different methods. Kernel is dead during a fashion of grid of block of threads. Block is assortment of Threads and Grid is assortment of Blocks.

data processing TECHNIQUES

data {processing} is that the process of discovering pat-terns by analyzing info through differ-ent views. The discovered information is then used for generating revenue. it's accustomed notice correlations among numerous massive fields particularly relative databases.

data processing techniques includes association rule mining, cluster etc. several algorithms are developed like K-Nearest Neighbour classifier ,Nave Bayesian classifier, FP Growth, Apriori, K-means for data processing functions.

Parallel knowledge Mining[2][3]

once data processing tools or formula imple-mentations ar done victimisation parallel computers, it results into high performance computing which might analyze huge knowledge briefly time. at the side of quicker computations, advanced knowledge is analyzed which might be during a bigger amount and thereby would offer improved results.

numerous challenges ar to be taken into con-sideration for parallel data processing like Com-munication , Synchronization and knowledge De-composition. If ignored, these will degrade the standard of information mining results. Dynamic load reconciliation is that the necessary issue to be consid-ered as parallel info servers has transient hundreds and multiple users.

The EMAC framework

The numerical world atmosphere–chemistry model EMAC (ECHAM/MESSy atmospherical Chemistry) could be a standard world model that simulates the chemistry and dynamics of the layer and layer. The model includes completely different submodels for the calculation of concentrations within the atmosphere, their interaction with the ocean and land surfaces, and therefore the evolution influences. The EMAC model runs on many platforms, however it's presently unsuitable for massively parallel computers thanks to its measurability limitations and enormous memory needs per core.

The untidy submodel MECCA executes severally the gas-phase chemical dynamics as a result of there aren't any dependencies between physical neighbors and no limitations by vertical closeness relations. during a typical configuration of untidy with a hundred and fifty five species and 310 chemical reactions, MECCA takes seventieth of the simulation execution time (Christou et al., 2016). the proportion of execution time will go up to ninetieth in simulations with additional advanced chemistry.

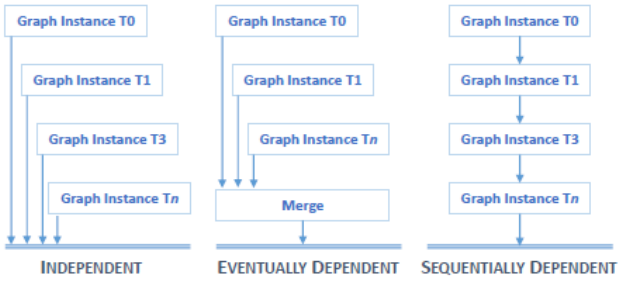
Currently, EMAC uses coarse-grained correspondence supported the Message Passing Interface (MPI). However, the present approach doesn't have the benefit of the accelerators that exist in fashionable hybrid superior computing (HPC) architectures. This puts severe limitations on current world climate time-length atmospherical chemistry and pollution transport simulations in terms of movableness, complexity, and determination.

EMAC uses the Kinetic Pre-processor (KPP) (Sandu and Sander, 2006; Damian et al., 2002) ASCII text file general analysis tool to formulate the mechanism. KPP integrates terribly economical numerical analysis routines and mechanically generates algebraic language and C code that computes the time evolution of chemical species from a specification of the mechanism during a domain-specific language.

Taking a group of chemical reactions and their rate coefficients as input, KPP generates code of the ensuing coupled standard differential equations (ODEs). finding the lyric poem system permits the temporal integration of the kinetic system. potency is obtained by exploiting the scantiness structures of the Jacobian matrix.

The largest challenge to deal with within the application is that the imbalance caused by the adjustive time-step measuring instrument finding the differential equations that describe the chemical equations computed. The varied strength at sunrise and sunset together with concentrations of precursors and gases (such as Nox and O3) ends up in chemical science reactions (mostly over midlatitudes within the stratosphere) that heavily alter the stiffness of the ODEs. as an example, Fig. 1a presents the accumulative range of execution steps needed for the combination method for every model column, presents the most distinction within the range of steps between cells in every column. The distinction within the range of steps within and between columns provides a sign of the robust imbalance created between execution times of various processes.

The ECHAM atmospherical self-propelled circulation part of EMAC solely scales up to or so a number of hundred cores (Christou et al., 2016) thanks to the significant all-to-all communication overhead of the spectral decomposition. At higher levels of correspondence, at or on the far side or so a thousand cores, the MECCA load imbalance thanks to the chemical science additionally becomes a limiting issue.



GPU ACCELARATED DATABASES

nowadays there area unit a embarrassment of widespread and useful databases to settle on from whereas beginning a contemporary direction project. of these databases offer exceptional func-tionality in numerous fields like RDBMS/NoSQL, ACID properties, Maintaining redundancy, protective dependencies, Providing security, Providing an expensive and powerful source language and then on.

However, in todays era of similarity, a number of these informations lack the necessity of providing similarity for execution of database opera-tions.

large similarity may be achieved if GPUs area unit employed in conjunction with databases to ac-celerate the operations and save precious time which can later be used for the other compu-tation. There area unit a number of GPU accelerated informations out there within the market that use the huge computation power of the GPU for general pur-pose database operations and speed the ex-ecution compared to ancient databases.

AUTOMATA PROCESSOR

design

The AP chip has 3 styles of useful parts - the state transition component (STE), counters, and Boolean ele-ments [5].The STE is that the central feature of the AP chip and is that the component with the very best population density. associate STE holds a set of 8-bit symbols via a DRAM column associated rep-resents an NFA state, activated or deactivated, via associate one-bit register. The AP uses a homogenous NFA representa-tion [5] for a additional natural match to the hardware operation.

In terms of Flynn's taxonomy, the AP is thus a awfully un-usual multiple-instruction, single-data (MISD) architecture:each state (column) holds distinctive responses (instructions) to potential inputs, and that they all respond in parallel to every in-put. Most different industrial architectures area unit John von Neumann architectures, e.g. single processor cores (SISD), multicore or multiprocessors (MIMD), and GPUs (SIMD).

The counter component counts the incidence of a pattern described by the NFA connected thereto and activates different parts or reports once a given threshold is reached. One counter will count up to 212 − one. 2 or additional counters may be daisy-chained to handle larger threshold. Counter parts area unit a scarce resource of the AP chip, and thus become a vital limiting issue for the capability of the SPM automaton planned during this work.

Micron's current generation AP-D480 boards use AP chips designed on 50nm DRAM technology, running at associate input sym-bol (8-bit) rate of 133 megacycle per second. A D480 chip has 192 blocks,with 256 STEs, four counters and twelve Boolean parts per block [5]. we have a tendency to assume associate

AP board with thirty two AP chips, in order that all AP chips method computer file stream in parallel.

Input and output

The AP takes input streams of 8-bit symbols. Any STE may be organized to simply accept the primary image within the stream (called start-of-data mode, little "1" within the left-upper corner of STE within the following automaton illustrations), to simply accept each image within the input stream (called all-input mode, little "∞" within the left-upper corner of STE within the following illustrations) or to simply accept a logo solely upon activation. Any sort of component on the AP chip may be organized as a news component; one news element generates a one-bit signal once it matches the input image.

If any re-porting component reports on a specific cycle, the chip can generate associate output vector that contains 1's in positions comparable to {the parts|the weather} that report and 0's for re-porting elements that don't report. Too frequent outputs can cause AP stalls, thus minimizing output vectors is a vital thought for performance improvement.

Programming and configuration

The Micron's AP SDK provides Automata NetworkMarkup Language (ANML), associate XML-like language for describing automata networks, also as C, Java and Python bind-ing interfaces to explain automata networks, produce input streams, analyse output and manage process tasks on the AP board. A "macro" could be a instrumentation of automata for encapsulating a given practicality, kind of like a perform or procedure in common programming languages.

Deploying automata onto the AP cloth involves 2 stages: placement-and-routing compilation (PRC) and loading (con-figuration) [1]. within the China stage, the AP compiler deduces the simplest component layout and generates a binary version of the automata network. within the cases of huge range of topologically identical automata, macros or templates may be precompiled in China stage and composed later [13]. This shortens China time, as a result of solely alittle automata network among a macro must be processed, then the board may be covered with as several of those macros as work.

A pre-compiled automata solely desires the loading stage. The loading stage, that desires concerning fifty milliseconds for an entire AP board [13], includes 2 steps: routing configu-ration / reconfiguration that programs the connections, and therefore the image set configuration/reconfiguration that writes the matching rules for the STEs. The dynamical of STE rules solely involves the second step of loading, that takes forty five mil-liseconds for an entire AP board.

The feature of quick partial reconfiguration play a key role in a very fortunate AP imple-mentation of SPM: the quick image replacement helps to affect the case that the overall set of candidate patterns exceeds the AP board capacity; the fast routing recon-figuration permits a quick switch from k to k + one level in a very multiple-pass rule like GSP for sequence mining.

PGStrom : PostGre SQL

PostGre SQL is one among the foremost widespread open supply direction systems out there within the market. like every ancient RDBMS system, it doesn't support GPU acceleration however it's potential to feature GPU acceleration to the present information. PG-Strom is a further extension for PostGre SQL information that is meant to utilize NVidia CUDA GPUs large similarity capability to perform intensive information operations. PG-Strom uses JIT compiler to choose whether or not a question may be with success paralleized and dead on GPU.

CONCLUSIONS

The paper focuses on the parallelization of information mining techniques. for various data processing applications , GPU with CUDA provides US advantages for parallelization. The opensource PostgreSQL info is employed for this purpose. It took around a hundred and twenty seconds for execution of 2 tables joined along mistreatment this info. The results demonstrate the power of those abstractions to scale and therefore the advantages of getting native support for time-series graphs in distributed frameworks. whereas we've extended our GoFFish framework to support TI-BSP, these abstractions will be extended to different partition- and vertex-centric programming framework too. The question execution time magnified roughly by forty seconds anytime a replacement table was joined. for 3 tables joined it took some a hundred and sixty sec , for four 220 sec then on. so applications that used distributed envi-ronment or supercomputers will be currently resolved employing a single desktop having NVIDIA graph-ics card for parallel computing usig CUDA. The AP-accelerated resolution alsooutperforms PrefixSpan and SPADE on multicore electronic equipment byup to 300X and 30X. By parallelizing candidate generation, these speedups area unit more improved to 452X and 49X. Even a lot of performance enhancements will be achieved by hard-ware support to reduce image replacement latency. The AP advantage will increase with larger datasets, showing smart scaling properties for larger datasets whereas the alternatives scale poorly.

REFERENCES

[1] Micron Automata Processor website, 2015. http://www.micronautomata.com/documentation.

[2] C. C. Aggarwal and J. Han, editors. Frequent Pattern Min-ing. Springer International Publishing, Cham, 2014.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In Proc. ICDE'95, pages 3–14. IEEE, 1995.

[4] S. Cong, J. Han, J. Hoeflinger, and D. Padua. A sampling-based framework for parallel data mining. In Proc. PPoPP '05. ACM, 2005.

[5] P. Dlugosch et al. An efficient and scalable semiconductor architecture for parallel automata processing. IEEE TPDS, 25(12):3088–3098, 2014.

[6] W. Fang et al. Frequent itemset mining on graphics processors. In Proc. DaMoN '09, 2009.

[7] P. Fournier-Viger et al. Spmf: A java open-source pattern mining library. Journal of Machine Learning Research, 15:3569–3573, 2014.

[8] V. Guralnik and G. Karypis. Parallel tree-projection-based sequence mining algorithms. Parallel Comput., 30(4):443–472, Apr. 2004.

[9] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proc. SIGMOD '00. ACM, 2000.

[10] K. Hryni´ow. Parallel pattern mining-application of gsp algorithm for graphics processing units. In ICCC '12, pages 233–236. IEEE, 2012.

[11] H. Noyes. Micron automata processor architecture: Reconfigurable and massively parallel automata processing. In Proc. of Fifth International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies, 2014. Keynote presentation.

[12] J. Pei et al. Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Trans. on Knowl. and Data Eng., 16(11):1424–1440, 2004.

[13] I. Roy and S. Aluru. Discovering motifs in biological sequences using the micron automata processor. IEEE/ACM T COMPUT BI, 13(1):99–111, 2016.

[14] T. Shintani and M. Kitsuregawa. Mining algorithms for sequential patterns in parallel: Hash based approach. In Proceedings of the Second Pacific−Asia Conference on Knowledge Discovery and Data mining, pages 283–294, 1998.

[15] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In Proc. EDBT '96, 1996.

[16] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009, pp. 497–506.

[17] D. Easley and J. Kleinberg, Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press, 2010.

[18] Y. Simmhan, C. Wickramaarachchi, A. Kumbhare, M. Frincu, S. Nagarkar, S. Ravi, C. Raghavendra, and V. Prasanna, "Scalable analytics over distributed time-series graphs using goffish," arXiv preprint arXiv:1406.5975, 2014.

[19] J. G. Siek, L.-Q. Lee, and A. Lumsdaine, Boost Graph Library: User Guide and Reference Manual, The. Pearson Education, 2001.

[20] S. J. Plimpton and K. D. Devine, "Mapreduce in mpi for large-scale graph algorithms," Parallel Computing, vol. 37, no. 9, pp. 610–632, 2011.

[21] P. Harish and P. Narayanan, "Accelerating large graph algorithms on the gpu using cuda," in High performance computing–HiPC 2007. Springer, 2007, pp. 197–208.

[22] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," CACM, vol. 51, no. 1, 2008.

[23] U. Kang, C. E. Tsourakakis, A. P. Appel, C. Faloutsos, and J. Leskovec, "Hadi: Mining radii of large graphs," TKDD, vol. 5, 2011.

[24] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "pregel: a system for large-scale graph processing," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010, pp. 135–146.

[25] L. G. Valiant, "A bridging model for parallel computation," CACM, vol. 33, no. 8, 1990.