PARALLEL PROCESSING OF STATISTICAL DATA. C++ PROGRAMS

Lorentz Jäntschi*, Mihaela Ungureşan*,
*Technical University of Cluj - Napoca

Abstract

In this work we presented a model for parallel processing of statistical data. The model presented implement and comparatively test four versions obtained from three algorithms, which process a set of statistical data. The versions differ through communication mode; two are of serial type and two are of parallel type. At last algorithm that order data, was implemented also by three versions, which differ by ordering mechanism of data (list and tree). The programs were tested at a large set of data (more than 15.000 input fields, more than 19.000 output fields). A version of these programs (those labeled "csf") can run in parallel at the same files (one write, other read) even at non-based parallel systems. A comparative study for execution of these programs was give.

Introduction

Parallel processing of information is one of most major concerning of researchers in field of information systems.

A significant part of this preoccupations is concerning to applications designed for multiprocessor systems[1]. The pipeline mechanism then is considered[2].

For IBM-PC architecture systems there exists a series of results related to transforming of linear algorithms used in classic programming into parallel algorithms, which it efficiently correspond to time and speed of calculus necessities. Here are some results:

In work [3] is applied parallel programming concepts at Sylow groups, which are fundamentally in group theory asymptotic efficiently algorithms. Is give an algorithm that find and conjugate Sylow subgroups of permuting groups, with application on problems that use this type of subgroups.

[4] it follow other direction, namely the planning for distributed execution of algorithms in neural networks. The authors establish theoretical links related to maximum efficiency that can be reach in browsing of neural networks and also present some algorithms that rich this efficiency.

The numerical calculus is frequently reconsidered through the perspective of parallel processing. [5] introduce a parallel algorithm for calculus of Lagrange interpolation in $N = n\cdot 2^n$ points at cubic connected cycle n dimensionally (CCC_n). The algorithm it have three stages: initialization, main and finalizing of calculus. While do not make calculus in initialization part, it contain one step, and main stage it contain $n\cdot 2^{n-1}$ steps. Every step consist in four multiplications, four additions and a transfer operations; an additional step include a division and a multiplications. The finalizing of calculus can be decomposed in two phases. First sub phase have n/2 steps, which include two additions and a transmission of data, followed by second phase, with n steps and every step with an addition and two transfer operations.

[6] describe constructing of parallel block of predictor - corrector methods based on correctors introduced by Runge-Kuta-Niström method. The followed objective was the applying of predictor-corrector method not only for the step with width of h (such as do Runge-Kuta-Niström method), but also for steps of width $a_i h$ ($i = 1,\ldots,r$). In this case, for every step, the entire block of approximations for exact solutions is calculated. In the next step, these approximations are used in order to obtain a predictor expressed by a formula of high rank using a Lagrange or Hermite

interpolation. Such as this block of approximations can be parallel processed, the evaluating time for this predictor-corrector methods are comparable with time obtained for conventional predictor-corrector methods. More, using of Runge-Kuta-Niström method does that the calculus of every approximation to be higher parallel. The numerical comparisons at partitioned memory slowing the presented methods efficiency for the problems where has complex functions to evaluate. In these article conclusions, the authors remark that a parallel block processing computer the method frequently necessitate two sequential evaluations of evaluating function for every step of selected precision order. The authors investigations shows that one of him methods (BPIRKN-B) has, in this case, favorable results. The comparisons of method with other computing method with optimized code (ODEX2) shows the advantages of the BPIRKN methods for some complex problems.

The initial application and statistical processing

It was developed a set of applications that has as input data measured properties (the file pr.pr) and 3D structures (from *.hin files). Are generated structure indices, which models different types of physic interactions through atoms from molecules (the files *.001, *.001, *.002).

Moreover, are correlated the structural indices that was generated (appreciatively 40.000 in actual version of programs) with measured property, in order to identify the optim model of property and finding of best regression property-structure equation. Are searching both mono-varied regressions (Y = Y(X)) and bi-multi-varied (Y = Y((X1,X2),(X3,X4),...). From realized applications, it was chose for implementing of parallel processing code type the application that make mono-varied regressions through structure indices calculated from files *.000, *.001, *.002 and measured property from pr.pr file. In order to parallel processing can make, is necessary to transfer data from files *.000, *.001, *.002 from column – type format to line – type format (making transposed of data matrix). For this operation was make the stat_gen program. The rest of calculus is parallel processing compatibles. Thus, first parallel processing program, let's call **p1**, will read index by index stored at one line in input file Ind_calc.txt and calculate the index covariances as follows: let property column read from pr.pr file at initialization stage noted with Y; let a index read at every step of processing X_i; at initialization step is calculated the values: $M(Y)$ and $M(Y^2)$ where M is mean operator; at every step of processing **p1** calculate the values: $M(X_i)$, $M(X_i^2)$, $M(YX_i)$, $M(\ln X_i)$, $M((\ln X_i)^2)$, $M(Y\ln X_i)$, $M(1/X_i)$, $M(Y/X_i)$; are transmitted values: at initialization stage, for initialization of second program, named **p2**, values $M(Y)$ and $M(Y^2)$; at every step of processing it transmit values: $M(X_i)$, $M(X_i^2)$, $M(YX_i)$, $M(\ln X_i)$, $M((\ln X_i)^2)$, $M(Y\ln X_i)$, $M(1/X_i)$, $M(Y/X_i)$;

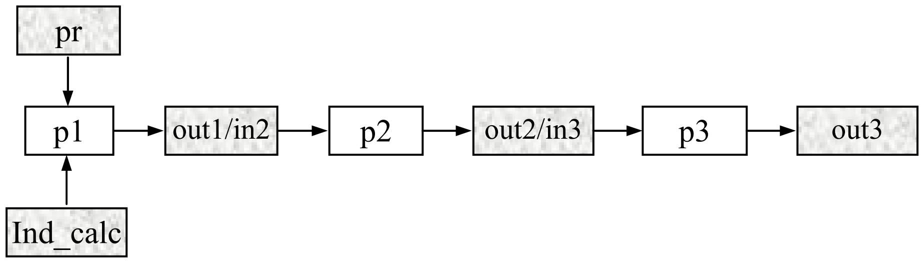

Figure. Processing scheme of statistical data

Second parallel processing program, **p2**, is initialized with values $M(Y)$ and $M(Y^2)$; at every step of processing, it read values $M(X_i)$, $M(X_i^2)$, $M(YX_i)$, $M(\ln X_i)$, $M((\ln X_i)^2)$, $M(Y\ln X_i)$, $M(1/X_i)$, $M(Y/X_i)$, calculate and write according to regression equations: $\hat{Y}_1 = a_1X_i + b_1$, $\hat{Y}_2 = a_2\ln X_i + b_2$, $\hat{Y}_3 = a_3/X_i + b_3$, $r_k = r(Y,\hat{Y}_k)$ the values: r_1, a_1, b_1, „ „+index_name; r_2, a_2, b_2, „ln„ + index_name; r_3, a_3, b_3, „1/„ + index_name; The third program, **p3**, it read the values r, a, b, linearizing_operator + index_name and place into an ordered tree (p3ver2, p3ver3) or a ordered list (p3ver1). At ending of him activity walk the list (p3ver1) or tree in recursively procedure (p3ver2) or iterative procedure (p3ver3) and write ordered values into a output file (divert after version of p3 program, for comparisons).

The implementation of code for these three applications p1, p2 and p3 it was make in C++; are used in implementation those libraries and function that are available in all three platforms: dos, windows and unix.

Versions of parallel processing programs

For case study, it was implemented more functional versions, that are presented in following table.

program / mechanism	p1	p2	p3		
			lista	arrec	arnrec
ssf	p1	p2s	p3sli	p3sar	p3san
csf	p1	p2c	p3cli	p3car	p3can
ssp	p1sp	p2psp	p3psli	p3psar	p3psan
csp	p1cp	p2pcp	p3pcli	p3pcar	p3pcan

The symbol "s" attached at program names represents that reading from input files are make through strings and data was writes through same string mechanism; the "c" symbol it represent that reading from input files are make character by character ("fgets(sir,2,din)" in code) and writing are make through the same mechanism; "p" symbol attached before "s" or "c" represent that output file are redirected to standard input file ("stdin").

In mechanism description (in same table) the letter "f" represent that programs will communicate through files and the letter "p" represent that the programs communicate through pipeline mechanism. Related to program p3, the sorting algorithm perspective was considered. The modality of direct placing of index read from input stream at correct position in order structure defined was choused.

It was followed two order structures, one "list" (represented by "li" in program name) and one tree ("ar" or "an" in program name). At tree mechanism, walking of ordered tree for writing of indices (sorted by correlation coefficient "r") was implemented both with recursive procedure "ar" and iterative procedure "an". The measure of time execution efficiency was observed related to algorithm type. All versions was passed at execution stage for an set of 640 indices at 48 molecules when was computed 15360 of indices and was maked 46080 regressions. The different values (after "r") it was write in output file; was resulted a number of 19346 of divert indices through "r" criterion.

One of important conclusions related to execution phase, are that versions "csf" it permit parallel processing of data even under dos shell windows; execution times proved this affirmation.

The versions of p3 program with list necessitate more execution time comparatively to tree mechanism, as he was expected. In case of both two tree mechanisms, with iterative and recursive disposing of memory, are not present observable differences in real time of execution. Following table present execution times for programs compiled under dos32 platform:

no	execution line	time (s)
1	p1.exe, p2s.exe, p3sar.exe	$55^{\underline{26}}$
2	p1.exe, p2c.exe, p3car.exe	$81^{\underline{84}}$
3	p1sp.exe | p2psp.exe | p3psar.exe	$55^{\underline{26}}$
4	p1sp.exe | p2pcp.exe | p3pcar.exe	$81^{\underline{84}}$
5	p3car.exe, p2c.exe, p1.exe (paralel)	$133^{\underline{00}}$

Observations at execution: the difference between the code executed by programs presented in table in line 2 and 4 compared to 1 and 3 it explained by modifying of input format, from string format to character format; the difference of $26^{\underline{58}}$ s are imputed to supplementary input operations; the differences between code executed by program from line 5 and programs from lines 2 and 4 (execution time difference is approximately double, $51^{\underline{16}}$ s side to $26^{\underline{58}}$ s); from here result that parallel execution do that intermediary outputs is makes character by character, through time quanta allocated by system for every active application.

References

[1] H.J. Park, K.M. George (1999) Efficient parallel hardware algorithms for string matching, Microprocessors and Microsystems, 23, 155-168.

[2] K. V. Asari, T. Srikanthan, S. Kumar, D. Radhakrishnan (1999) A pipelined architecture for image segmentation by adaptive progressive tresholding, Microprocessors and Microsystems, 23, 493-499.

[3] William M. Kantor, Eugene M. Luks, Peter D. Mark (1999) Sylow Groups in Parallel, Journal of Algorithms, 31, 132-195.

[4] Márton Nagy, Suresh Singh (1998) Multicast scheduling algorithms in mobile networks, Cluster Computing, 1, 177-185.

[5] H. Sarbazi Azad, M. Ould-Khaoua, L.M. Mackenzie (2000) A parallel algorithm for Lagrange interpolation on the cube-connected cycles, Microprocessors and Microsystems, 24, 135-140.

[6] Nguyen Huu Cong, Karl Stremhel, Rüdiger Weiner, Helmut Podhaisky, (1999). Runge-Kuta-Niström-type Paralel Block Predictor-Corector Methods, Avances in Computational Mathematics, 10, 115-133.