The Parallel Programming on the LISP-base

N.Archvadze, M.Pkhovelishvili

I. Javakhishvili Tbilisi State University. Tbilisi, 0128, Chavchavadze str.1.
N. Muskhelishvili Institute of Computational Mathematics. Tbilisi,0193, Akuri str.7

Summary

It is shown, how it is possible on LISP, which is the functional programming language, will be made the parallel calculations organization. The possibility of the parallel calculations is stipulated by the existence of the high row functions-functionals. It is necessary in the limit of the existed syntax and semantics will be changed the map-functionals interpretation so that would be the possibility the calculations each part realization on the separate processor (is implied, that the computer is multiprocessor). The broadening by the modern procedure and on object oriented languages map-functionals type functionals gives to consumer the possibility to write the parallel algorithms.

Key words: *Style programming, parallel programming, functional programming, map-functional.*

The natural reserve, that will be the computer productivity increased, is the parallel processes management. Its organization demands the temporal connections foresee and the action management no imperative style.

The programming language LISP [1,2,3] during its existing semi centennial history till to date remains the symbolic programming system with the practical unlimited possibilities LISP and its dialects got the bases of a number the applied researches, which have the big role in the informational technologies spreading.

LISP is the programmed programming functional language, therefore we studied the universal programming questions on the LISP base [4]. Now we want to show, how it may be organized the parallel calculations for the multiprocessor personal computer and thee these possibilities realization in modern languages (for example, in C/C++).

The parallel calculations possibility was put in LISP as the functionals. the functionals realization could the LISP first versions too. So the transition from the programming functional style to the parallel programming we believe is natural. Moreover we want to mark, that now, when the computing technics development made in direction the multiprocessor personal computers creation, very actual is, that in modern programming languages will be put the possibility of the parallel calculations.

In the functional languages are determined the high level functions the name of which is functionals [2,5]. The functionals that are the functions, which use as argument other functions or determine function results. At functionals determination the role of the variables execute the functions names, their definitions by the outward formulas. Which use functionals are given.

It necessary mark, that data and programs in LISP are occurred equally, therefore the difference between concepts „the date'' and „the function'' is defined not by their structure but by their using. If the argument in the function use as the object, which participates only in the calculation, than it is usual the argument-date, but if it is used as the mean, which determines the calculations, for example, plays the role of the **lambda** image, then it is function.

For example:

```
(car '(lambda (x)(list x))) → lambda          ; date
((lambda (x)(list x)) car)→(car)              ;  function
```

In the LISP are determined the reflected functionals or **map**-functionals as the functions, which the lisp (the succession) reflect by any mode (**map**) in a new list, or by the second action create this succession. The **map**-functionals names begin by the word **map**, their call has the form:

(MAPx f_n l_1 l_2 ... l_N) there l_1 ... l_N are lists, but in- the function with N-argument. As a rule **map**-functions are used are used on the one argument-list, or the Fn-function is with the one argument (MAPx fn list).

The reflected functionals divide on the two group. The first, which provides the given list's each element's treatment **(mapcar, mapcan**) and the second, which provides the initial list's and its each tail treatment (**maplist, mapcon).** The list's tail is the list, from which the first element is removed.

The functional **mapcar** provides the functional argument realization on the list all element and the result unites in the list. For example, (mapcar 'list '(a b c)) gives the result '((a) (b) (c)). The functional **mapcan** is analogy to **mapcar** with the difference, that for the carrying out of the result is used the function **ncons**. For example, (mapcan 'list '(a b c)) gives the result '(a b c).

The functional **maplist** provides the functional argument realization on the list and its all tails. For example, (maplist 'list '(a b c)) gives the result '(((a b c))((b c))(©)). The functional **mapcan** is analogy to **maplist** with the difference, that for the result carrying out is used the function **ncons.** For example, (mapcon 'list '(a b c)) gives the result '((a b c)(b c)©).

The example:

(mapc 'list '(a b c)) gives the result '(a b c).

(mapl 'list '(a b c)) gives the result '(a b c).

In the Clisp language [6] to functionals belong the following functions: Mapc, Mapcan, Mapcar, Mapcon, Mapl, Maplist. In the LISP one modern version Objective CAML [7] is introduced the stream concept and it is the possibility of the parallel algorithm writing. The Objective CAML has the the library for the ,,light'' processes stream, which is organized by self process but no by the operative system. The such streams use the area of the creating them process and therefore demand the smaller resources. The principle difference between the stream and process is this, that simultancously use or not the memory for date the same program for the daughted processes. The streams using is the mean for the parallel algorithms realization in the language limits.

The functionals using in the programming language gives a number advantages:

- by their means it is possible to make programs from the more big actions (than with functions help);
- they provide the reflection flexibility;
- the function definition may be independent from the function name (the function definition by **lambda** image);
- by the function means may be the result forms management;
- the functions parameter may be any function, which transforms the function's elements;
- the functionals give the possibility to form the function series from the common date;
- the intercommunicational functions any system may be transforms in one function by the nameless functions calls.

As we see, the **map**-functionals so make calculations, that by first argument given function will be calculated to by the second argument given list each member (at the first group functionals), or to by the second argument given list and its tails (at the second group functionals), therefore may be to say that **map**-functionals by its nature are ,,parallel''. It is necessary for the multiprocessor computer the language compiler will be made so, that will be conducted on the different processor. The each calculation will be made independently on processor, which will be returned the result to **map**-functional with the indication, with which was made its call. More exact, at the first group functionals to processor is transferred the function name and the list, on which executes the action, but at the second group functionals- the function names and the calculated tails. The processor the calculation result with the indieation of this calculation number returns to **map**-functional and just

oneself releases executes the next calculations. Cleary this process not will be depended on the processors quantity in computer.

It may seem to us expedient, that the modern procedure and on the object oriented languages must be broadened by the **map**-functionals and namely these functionals will be means for the ,,parallelism'' of these languages. Moreover it will be made the transfer of the each type **map**-functional so, that as the second argument will be used the list, massif (of the whole and real numbers), line, file and by consumer determined type. By **map**-functionals using it is possible any parallel algorithm to record.

The parallel calculation realization on the multiorocessor computers gives a number advantage in the information search tasks. The by the nets presented information light may be given by the LISP lists [8]. Instead on them standard search algorithm as by the straight way passage, reverse way passage and passage by last row [9], the search may be realized by the wave principle. By the wave search method it is possible the search will be made from the top on the first level. If it is found a result, the process will be finished if not, the search will be continued on the all branches second level. For the each branch it is necessary to choose the separate processors. Therefore we have the parallel structures, the search touches place simultaneously, parallel.

For example, the such type task is the words search in the natural languages dictionary, when the language's semantic net may be presented as the separate trees. The search may be made by the separate processors in separate trees; with this may be sharply to shorten the need time for the search process.

Literature

1. Хьювенен Э., Сеппанен Й. Мир Лиспа., т.1,2, Москва: "Мир", 1990.
2. Хендерсон П.. Функциональное программирование. Применение и реализация. Москва: "Мир", 1983.
3. Лавров С., Силагадзе Г. Автоматическая обработка данных – Язык Лисп и его реализация. Наука, Москва,1978.
4. Archvadze N., Pkhovelishvili M. The issue of universal programming. "Science and Technologies". #7-9, 2003, 49-52.
5. Интернет университет информационных технологий. http://www.intuit.ru. Курсы: Введение в программирование на Лиспе, Основы функционального программирования.
6. The sait, on which is disposed for the free spreading the system of the **Clisp**-s realization (**Clisp**-2.29-win32.zip) ftp://ftp.gnu.org/gnu/clisp/release/2.29/
7. Description of programming language Objective Caml: http://caml.inria.fr/pub/docs/oreilly-book/html/index.html
8. Archvadze N., Pkhovelishvili M.,.Shetsiruli L. Data presentation by LISP structures. "Science and Technologies", 2008, #3.
9. Кнут. Искусство программирования. Том 1.

Article received: 2008-09-27