FACTA UNIVERSITATIS (NIŠ)
Series: Electronics and Energetics vol. 10, No.2 (1997), 231-236

LOAD BALANCING IN DISTRIBUTED SYSTEMS

Wolfgang Weber and Michael Witt

Abstract. Distributed systems can be either connected computer or an array of single processor systems. The described method of speed performance, realized by load balancing procedures, can be used to both of these basic structures. By the application of genetic algorithms, using mutation as well as crossover, an optimal distribution mode is guaranteed.

1. Introduction

To get a high speed performance in parallel systems, an optimal process distribution is the basic condition [1]. Generally it is not possible to parallelize an algorithm totally, which means, that always a serial part remains limiting the maximum calculation speed compared to a totally parallel distributed process.

They are several conceptions to express the influence of this serial part to the theoretical computing capacity [2]. We find here the

- the Amdahl's law and
- the scaled law of Gustavson

Both formulas are based on the direct comparison between a one– and a multi–processorsystem. The gain in speed–up defined by

$$S_N = \frac{t_{s,ges}}{t_{p,ges}} \tag{1}$$

is regarded.

Here is t_s the execution time of the serial part of the algorithm and t_p the execution time of the equivalent parallel algorithm.

Amdahl's law results from the idea, that the influence of the serial part grows with the number of processes. In comparison to that the hypothesis of

Manuscript received March 11, 1997.
Prof. Dr. Dr. h.c. Wolfgang Weber is director, Dipl.-Ing. Michael Witt senior research fellow of the Institute of Computer Science, Ruhr–University Bochum, Germany.

Gustavson's law argues that even using massivily parallel computer systems does not influence the serial part and regards this part as a constant one. Practical experiences could proof that [2]. Figure 1 illustrates the two laws.

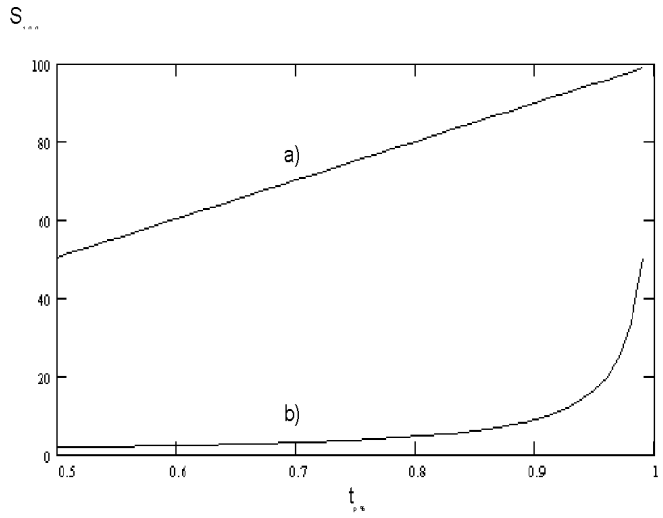

Figure 1. Theoretical speed–up a) Gustavson and b) Amdahl.

Not regarding the granularity, i.e. the degree of parallelism of the algorithm which can be influenced by the user their communication and administration needs are a further value of influence which cannot be neglected. This can be shown by modelling, f.e. realized with the work–/exchange model [1], [3].

Distributing the processes means additional efforts and costs. So the communication and administration demands cannot only be reduced by the system software. Instead of that the used hardware should already offer some facilities.

Possible points of influence are hierarchical system topology and – on the execution level – packet switching (see figure 2).

1.2 Dynamical load balancing

The aim of the load balancing is to optimize the process distribution for the theoretical computing capacity. Here the set of the possible transformations of the process graph $\mathcal{P}$ onto the system graph $\mathcal{N}$ (see figure 3) must be regarded.

$$\mathcal{V} = \{\nu \mid \nu : \mathcal{P} \rightarrow \mathcal{N}\} \tag{2}$$

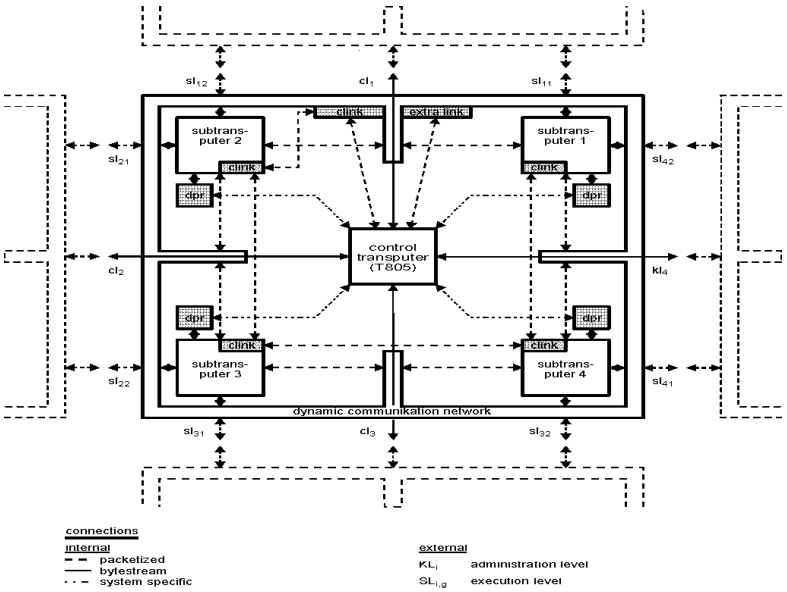

Figure 2. Hierarchical system topology.

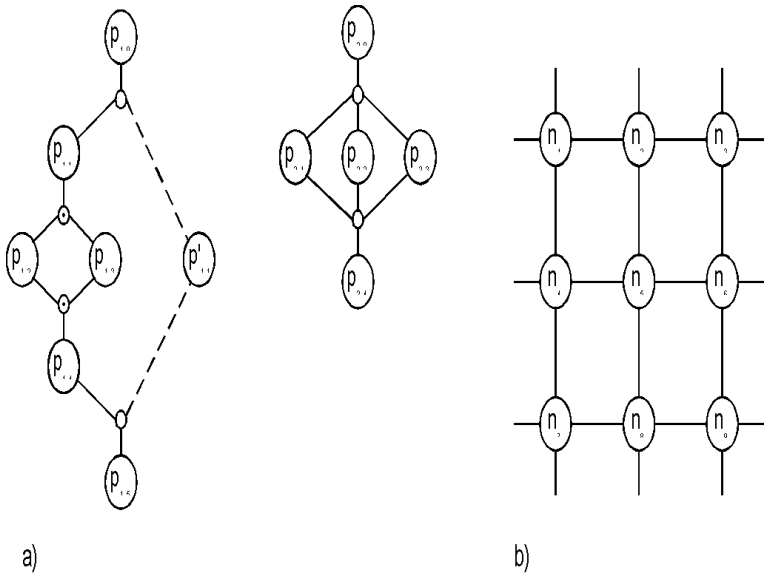

Figure 3. a) Process graph and b) System graph.

For the load quotient L_q i.e. the unbalance of load between the processors is demanded:

$$L_q \frac{w_{max}(\nu)}{w_{opt}} = 1 \text{ with } w_{max} = \max_{i=1}^{N} w_i, \tag{3}$$

where w_{opt} results by definition from an equal distribution (ν_{opt}) of the load.

The number of possible process distributions increases by $N!$ (N is the number of processors). So the load balancing problem is not at all trivial and cannot be solved easily.

In the following section a dynamical load balancing is described. This means the distribution of processes during their runtime. In comparison to static methods – using the number of processes and/or the memory requirements – the stochastical runtime behaviour of all processes is not neglected [4].

2. Realization

Most of the conventional dynamic load balancing methods use only the number and/or the RAM demand of the processes. They so don't regard the time related values. Here the number of processes is the only parameter in the optimization process. The execution time of the processes is additionally assumed to be constant. This asumption is valid only in exceptions.

Therefore the load balancing method described in the following sections begins with the time related definition of the load quotient

$$L_q = \frac{t_{max}(\nu)}{t_{opt}} \tag{4}$$

2.1 Estimation of load

Before distribution change of the load can be realized the load data must be derived by special methods. Here we must separate

- computing load
- communication load.

The estimation uses a simple counting method. Here within a given measuring interval the difference of in and out going communication demands and the number of calls of the measuring process are counted. The measuring process is added to the APL (Active Process List) of the processes of low priority [5].

The resulting load data are used to calculate process related values regarding the work and data storage of the processes and then to leave it to the control instance of the processor node.

2.2 Evaluation procedure

The evaluation procedure bases on a "genetic" algorithm. The simplicity of the operators used for the modelling of the "natural optimization process" [6], as crossover and mutation is used for the solution of the load balancing problem. Further advantages of genetic algorithms are the convergence and easy parallization.

Figure 4 shows the basic structure of the method. From a given number of possible solution vectors, i.e. process distributions (basic population) first the two best solutions are calculated and changed by stochastical mutations within the vectors (mutation) and between the vectors (crossover) [6].

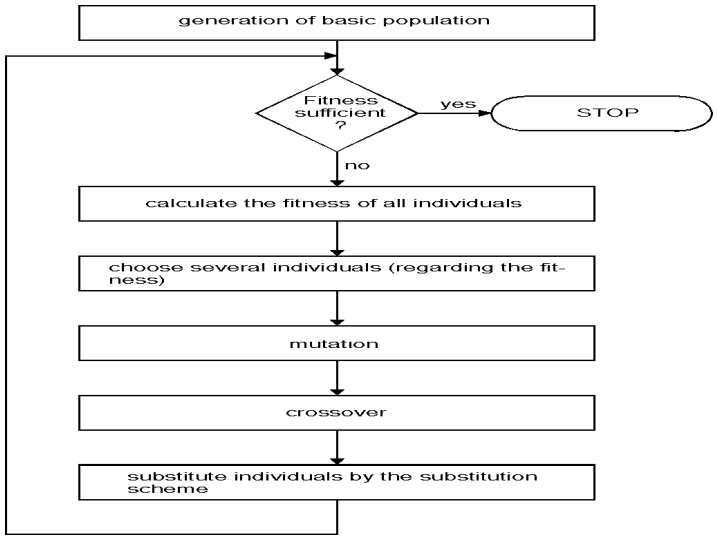

Figure 4. Structure of the genetic algorithm

Next the new generated solution vectors get fitness values and are eventually added to the population. The optimum is so represented by the solution vector with the greatest fitness. The calculated load data influence the calculation of the fitness. The aim of the optimization is to minimize the difference of load data over all the processors. The type of the method results in a favorization of solutions with high "quality", i.e. brings process distribution in the environment of the optimum.

Hereby also the good convergence of the method can be explained as optimization operators the position based crossover and the position based mutation with the probability of $0.4 - 0.7$ (crossover) resp. $0.6 - 0.8$ (muta-

tion) is used. More complicated operators bring only a worse time behaviour because quite different to the travelling salesmen problem the uniqueness of the solution vectors is not demanded in advance.

Regarding the special structure of the parallel system [3] the load balancing method is parallelized following the network model. The process distribution is in the beginning only locally done, i.e. within the processor node. For the exchange of load tables between the single processor nodes static connections for the communication are used.

Results of simulation show that very short process times have a negative influence to the efficiency of the method. This can be easily understood as the load estimation and not least the execution time of the genetical algorithm cause a certain latency time. This grows with the number of processes.

For average and big process execution times and high values of granularity the method anyhow works very efficiently. This exactly is knownly given for the most scientific applications.

REFERENCES

1. Wikstrom M.C.: *The Work–/Exchange Model: A generalized approach to dynamic load balancing.* Dissertation, Iowa States University, 1991.
2. Harms U.: *Amdahls Gesetz und die Parallelverarbeitung.* Design & Elektronik 26 (1990), 21. Dezember 1990, S. 112 ff, Markt & Technik, Haar 1990.
3. Witt M., Weber W.: *Dynamischer Lastausgleich in Parallelrechnersystemen unter Berücksichtigung des Work-/Exchange Modells.* Facta Universitatis, Series: Electronics and Energetics 1(1995), S. 119ff., University of Nis, 1995.
4. Schabernach J.: *Lastausgleichsverfahren in verteilten Systemen. Überblick und Klassifikation.* Informationstechnik, Nr.5, München 1992.
5. Behlert D.: *Implementation und Test von Algorithmen zur Lastabschätzung bei Transputern am Beispiel einer FFT.* Diplomarbeit D261, Lehrstuhl für Datenverarbeitung, Bochum 1995.
6. Schöneburg E.: *Genetische Algorithmen und Evolutionsstrategien: Eine Einführung in Theorie und Praxis der simulierten Evolution.* Bonn, Paris 1994.