THE EFFECT OF NON-EXECUTION AND PART TASK RESULT INTEGRATION IN DISTRIBUTED DATA-INTENSIVE COMPUTING SYSTEMS

András Benczúr[1], and Antal Buza[2]

[1]Eötvös Loránd University, Faculty of Science, Department of Information Systems,
Pázmány sétány 1/c, Budapest, H-1117 Hungary
abenczur@ludens.elte.hu

[2]College of Dunaujvaros, Institute of Informatics,
Tancsics utca 1/a, Dunaújváros, H-2400 Hungary
buza@mail.poliod.hu

Abstract. The distributed computational systems and the distribution algorithms are thoroughly studied. In this paper we focus on data intensive distributed problems including of task distributions, network delay times, join of the part task results, and the effect of non-execution. We first make a study of dedicated multi computer systems in which the number of computers and the capacity of computers are almost constant. Then we analyze non-dedicated systems where the number and the free capacity of computers are frequently variable. The solution time of the full task depends on a large number of complex parameters. The effects of different parameters are demonstrated by numerical analysis based on behavior of distributed data intensive super-computing systems.

1. Introduction

The users of the computers appraise the suitability and goodness of the systems by means of the completion time, including the correct functioning of course. The aim of this paper is to examine the reduction of the completion time.
The elaborating time of the answer is influenced by the following components:

- network-times, network-delay times
- task-distribution and redistribution times
- integration times of part-task results
- algorithm-times (processor times, running times)
- data-access times (local or remote data access)

Units of experiments: The study of response times has a great importance only in cases of long runtime systems and/or in cases of use huge mass of data. Hence for it is assumed, that the programming algorithms, the technical algorithms (use of disks, use of network and so on) and the data store are the bests. It means that we focus attention on the limits of completion time of ideal systems. (There are no

more subjective things to be corrected.) The limits of completion times of ideal cases are searched for. These limits may be reached but never can be exceeded. We show a frame for managing this problem.
The basic scheme of response time, in case of a "theoretical" simple computer which has only one processor and one data store:

$$V = r^* - o^* + a^* \cdot t \qquad (1)$$

where r^* is the algorithm time, a^* denotes the quantity of data (expressed in some unit, for example in Gbytes), t denotes the access time of a unit of data. The algorithm-times and the data access times may be overlapped. In order to keep this form in validity we must calculate with the overlapping time. Let o^* denote the overlapping time between the algorithm-times and data access times. Thus, the runtime without overlapping is: $r^* - o^*$.

2. Local systems

We call a system as "local system" where all data accessed locally and the computer network has no role. The full task completion times of local systems are discussed in [4] as "one-box" in detail.

3. The networked systems

The complex systems are called "multi-box" systems, which consist of more (partially) independent computers, and they are connected together with computer networks. The data, the tasks, or both of them may be distributed in the multi-box systems.
About the non-execution. It may occur that one or more computers are not able to complete the distributed part-tasks. When it occur the task distribution system re-distributes these part-tasks using different re-distribution strategies.
Using the multi-box systems it is important and time-consuming to solve the problems of task-distribution, join of the part task results, managing of non-execution and the task redistribution.
About the network-times. The network-times constitute the biggest uncertainty in the measure of response time in case of multi-box systems. The network-times are unpredictable according to practical evidence. Essential differences can be seen in network-situations which are very similar to each other. Rising of traffic jams is an interesting question in computer networks.
About the questions of the task-distribution, integration of part-task results and redistribution. If an algorithm can be divided in part reasonably - in ideal situation - the most efficient distribution uses all computers in nearly equal time periods. In this way the start-stop time (for a full task) will be minimal.
Two main types of task-distributions are analyzed in the sequel. Section 3.1. deals with dedicated multi-box systems and section 3.2. the non-dedicated multi-box systems. Managing and effect of the redistribution studied in section 3.3.

3.1. The dedicated multi-box systems

The number of computers does not change essentially during the execution of the task. The idealized "good" task distribution when: $V_1 \cdot m_1 = V_2 \cdot m_2 = \ldots = V_N \cdot m_N$ where V_i denotes the execution time of one part-task using the i-th computer including the network times, m_i denotes the number of part-tasks for the i-th computer respectively. The demand of "=" in the above forms is too rigorous "≈" (approximations) are used in the practice. The number of part tasks: $\sum_{i=1}^{N} m_i = M$.

The start-stop time is:

$$V = D(N,M) + \max(V_i \cdot m_i) + I(M) \qquad (2)$$

where the redistribution times are not taken into consideration. In form (2) function D denotes the time of the distribution, N is the number of computers, M is the number of part-tasks, function I is the integration time of the final result from part-tasks results.

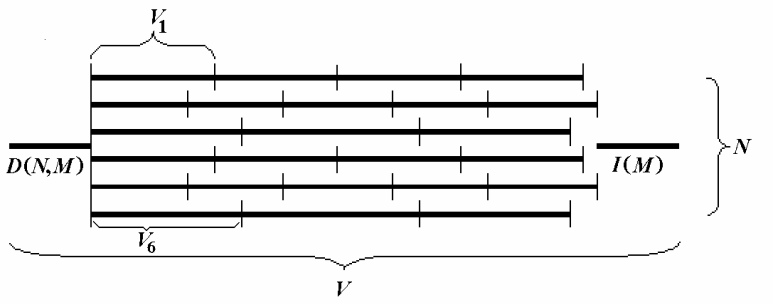


Fig. 1. The time-diagram of nearly-ideal part-task distribution without non-execution when the number of computers is constant.

On determining the number of elementary part-tasks, and the distributed part-tasks. Each of the full tasks has its own lower limit for the size of distributable elementary tasks. From practical point of view the part-task size used for distribution is larger than the elementary task size. (In this section we use the "size" as the part task execution time and suppose that the execution time of the selected part-task on i-th computer is V_i .) The numerical results for model 1 and 2 (showed in section 4.) shows the effect of use of different size of part-tasks. For practical use of model we can estimate the relative size of part-task for different computers by sending an experimental part-task for the computers and measuring the response time (by assuming the constant network times for the observed computer).

3.2. Non-dedicated multi-box systems with variable number of computers

The number of used computers may change during the execution of the task. The dynamic distribution of the task necessary in such environment where the number

of computers is variable. This type of task distribution is used for example on Internet. During this operational mode when a computer has free capacity it may offer its free capacity for the task distribution system. Computers may exit from this co-operating community at unpredictable moments of time. It may occur that the computers do not accept new part-tasks, or do not execute the part-task received earlier. The task distribution system does not know the number of the available computers. In this case the scheme of process is significantly different from the dedicated scheme. Figure 2. illustrates that all new applicant caused intermediate distribution period and all part-task completions caused intermediate integration and new distribution period.

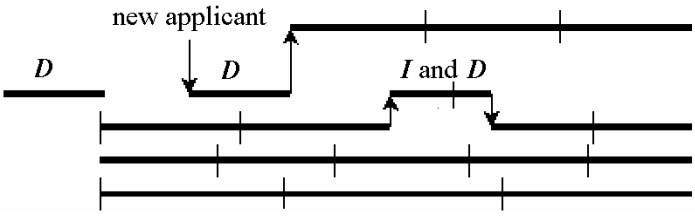


Fig. 2. One possible time-scheme with various number of computers. The distribution routine gives part-tasks for a new applicant.

3.3. Managing and effect of the redistribution

Recognition of the non-execution has a great importance especially in the last case mentioned above. When a computer does not execute its first part-task during x-times of the maximum of part-task execution times then the task distribution system releases it. The distribution manager considers that this computer has not enough free capacity or it is "far" on the network. During the co-operation when a computer exceeds by y-times its average executes time then its part-task is considered as non-executed and this computer shows non-execution and its considered as an exited computer. The values x and y are environment-dependable selected values. We can determine in all concrete cases what the limit of time-out is.

Dedicated cases. Figure 3. illustrates the effect of non-execution with constant number of computers. The method of result-communication is: after the completion of all distributed part-tasks for given computer. Using this method the caused traffic of computer network relative rare, the recognition of non-execution is late happened and all part-tasks distributed for selected computer must be re-distributed (because all their results are lost although some of them are completed). K denotes the number of dropped out computers.

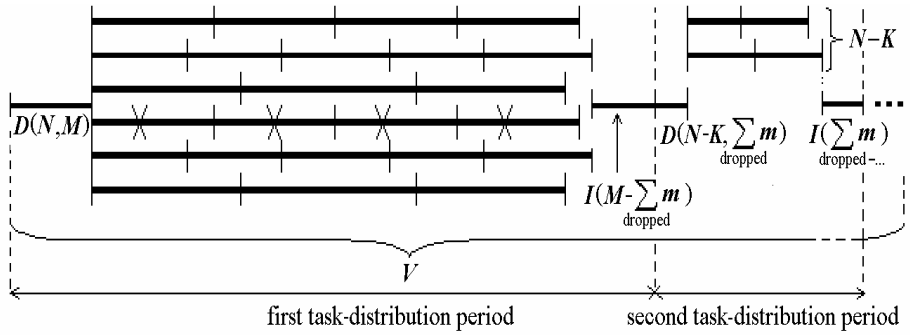


Fig. 3. Time-scheme of distribution-execution-integration schedule.

Figure 4. illustrates the effect of non-execution with constant number of computers. The method of result-communication is: "continuous" i.e. immediately when one part-task is performed. Thus may decrease the number of re-distributed part-tasks and render the earlier recognition of non-execution possible. Moreover decreases the full-task completion time by the (partial) overlap of execution and integration times. K denotes the number of dropped out computers, p denotes the probability of non-execution of a part-task .

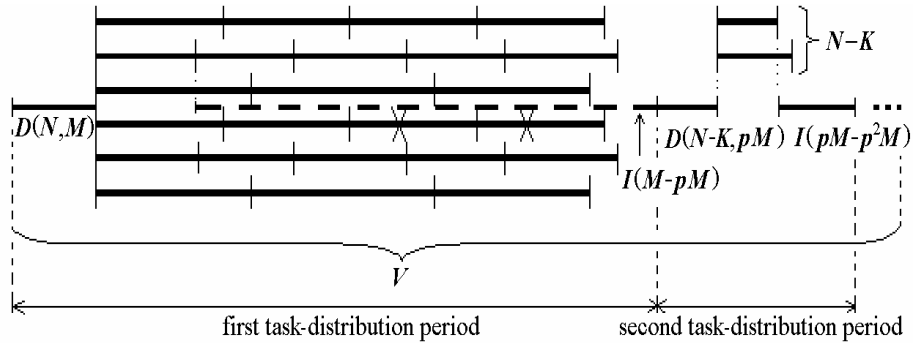


Fig. 4. Time-scheme of overlapped execution and distribution-integration schedule.

The expected completion-time of the full task

The expected completion-time of the full task showed on fig. 3. is: $V = \sum_{i=0}^{\infty} V(N(1-p)^i, p^i M)$ in case of essentially constant number of computers it is simplified to: $V = \sum_{i=0}^{\infty} V(N, p^i M)$ where p denotes the probability of non-execution of a part-task; $V(N,M)$ denotes the time of completion of M part-tasks on N computers without non-execution.

In case of $V(N,M)$ which is a linear operator and the changes the number of computers are small the completion time of the full task approximately:

$$V = V\left(N, \frac{M}{1-p}\right)$$

The expected completion-time of the full task showed on fig. 4. is upper bounded by the above results in case of showed on fig. 3.

Plenty of conditions are used. The algorithms, the network-situations, the errors appearing are various. They depend on many parameters. Number of these parameters contain random or probability elements. Exact description of the elements is a hopeless task. The more precise formulas hide its essential parts.

Non-dedicated case. Figure 5. illustrates the time-scheme with various number of computers considering of non-execution. At the beginning the distribution manager distributes only a part of all part-tasks and later when new applicant computer appear the distribution manager gives for him some part-tasks.

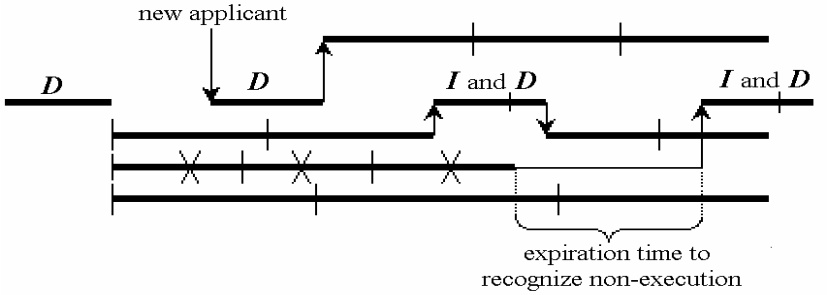


Fig. 5. One possible time-scheme with various number of computers considering of non-execution.

In scheme shoved on figure 5. the distribution and integration times are overlapped by the execution times (expect at the start and at the end of process). The number of non-complete part-tasks is increased in each time when the system recognized the non-execution situation. This increasing is at least:

$$\frac{M}{k}\sum_{i=1}^{k} p_i \approx \frac{M}{k} kp = pM$$

where M is the number of part-tasks, k is the number of simultaneously distributed part-tasks, p_i is the probability of the non-execution by i-th computer. The above form is reduced when we suppose, $p_i = p$ for all i. The completion time of full task is the completion time of $M+pM$ part tasks, i.e. approximately:

$$V \approx \frac{M(1+p)}{k} V\left(k, \frac{M(1+p)}{k}\right).$$

In the reality the number of computers and the probability of non-execution are variable. It is possible to give a more precise formula, but it will be more complicated.

We do not complicate the above formulas with the whole values, but in the numerical analysis and in the simulations we used only whole part-tasks and all consequences of course.

Decreasing the loss of time caused by non-execution. Nearing its end of the part task distribution it is possible, that the number of available computers is larger than the non-complete part-tasks till now. In this case we can distribute the same part-task for more computers. By doing so the complex probability of the non-execution of the multiple distributed task will significantly decreased. Consequently the re-distribution and part task completion times were decreased. The required time for part-tasks result integration is unchanged while the number of complete part-tasks are constant. (This strategy is usable when the use of more computers is not cost-sensitive.)

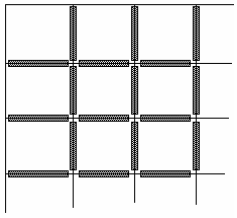
Importance of the numerical analysis and simulation. The study on task-distribution completion time accomplished by analysis/simulation is very useful in the practice. The parameters and the algorithms of the simulation are to be set and changed according to the measurement and theoretical results. The shortest task executing time can be determined with the help of analysis/simulation. The influence of the elements and the parameters can be observed. The reasonable number of used computers and the reasonable task-distribution strategies can be studied.

4. Numerical analysis and simulations

We used the MATLAB program package for the numerical analysis and simulations.

4.1. Model 1

The model showed in section 3.1. when the full completion time of a task is: $V = D(N,M) + \max(V_i \cdot m_i) + I(M)$. We suppose that the part-task times are the same for all computers, and the computers are totally dependable (not occurred non-execution). Functions D and I are similar to functions used in the real image-processing task. We use a simple distribution function: $D(N,M) = (N+M)/10000$. The Integration function is: $I(M) = (2M - 2\sqrt{M}) / 20$. The modeling environment permits of use variable distribution and integration functions. The motivation of $D(N,M)$: We measured and approximated the distribution times of an simple distribution algorithm.



The motivation of $I(M)$: when we can integrate the parts of distributed image, we can process the connecting areas. In the rows and columns are $\sqrt{M}$ elementary connecting areas, the number of rows is $\sqrt{M}$ -1, so this way the total number of elementary integration tasks is: $2\sqrt{M}$ ($\sqrt{M}$ -1)=$2M$-$2\sqrt{M}$.

The result of simulation showed figure 6 and 7.

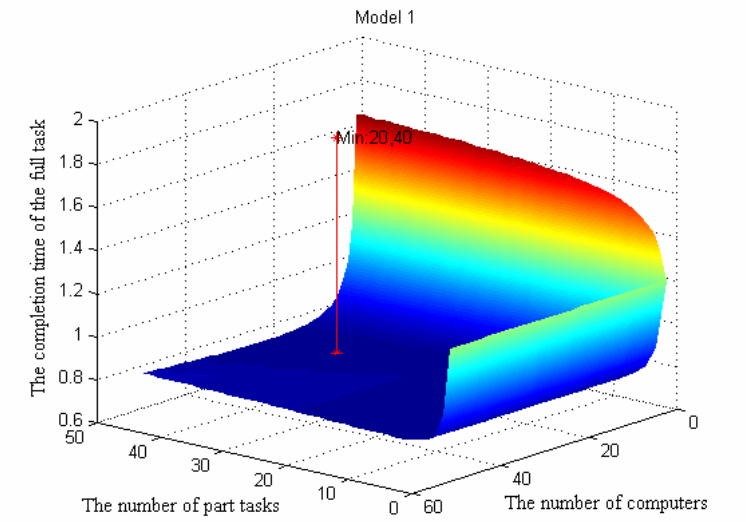


Fig. 6. The number of computers is changing from 1 to 50, the number of part tasks is changing from 1 to 50 (the whole task is constant, and the maximal number of useable part-task changed from 1 to 50). The minimum of full task completion time given by using 20 computers and 40 part-tasks.

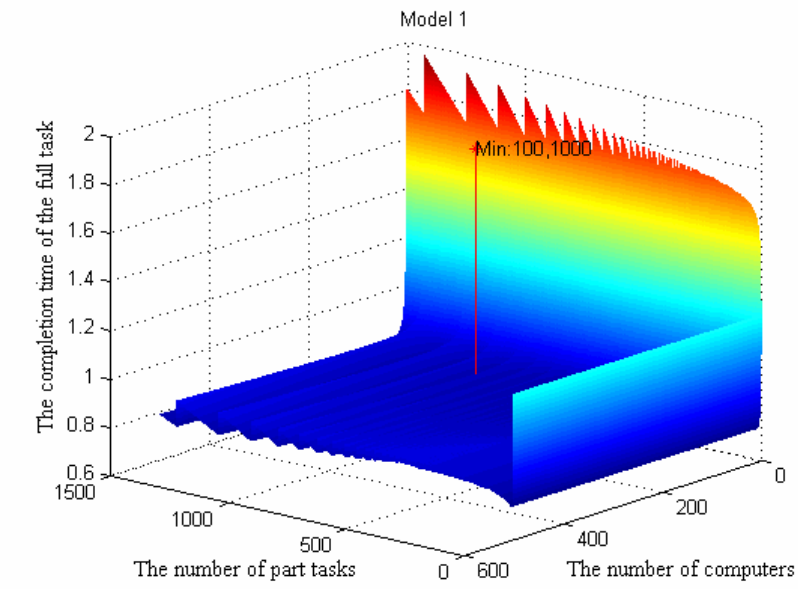


Fig. 7. The number of computers is changing from 1 to 500, the number of part tasks is changing from 1 to 1500, the minimum of full task completion time given by using 100 computers and 1000 part-tasks.

4.2. Model 2

Model 4.2 is an improvement of model 4.1. using the calculations illustrated by Fig.3. We consider the non-execution by probability of p.

The results: the available best full task completion times (and M and N belonging to the best cases) are summarized in the following table:

N at the start time	p=0.1			p=0.2			P=0.3			p=0.4		
	V	N	M	V	N	M	V	N	M	V	N	M
20	0.116	10	10	0.120	20	20	0.171	20	20	0.221	20	20
50	0.061	50	100	0.082	50	100	0.102	50	100	0.123	50	50
100	0.043	100	100	0.054	100	100	0.065	100	100	0.846	100	100
200	0.037	200	1000	0.044	197	589	0.051	200	200	0.058	186	186
500	0.030	500	1000	0.034	477	1429	0.044	295	295	0.041	500	1000
1000	0.029	1000	1000	0.032	715	1429	0.034	1000	1000	0.035	1000	1000

Where V denotes the full task completion time, N denotes the number of computers, M denotes the total number of part-tasks, p denotes the probability of non-execution of a selected part-task.

4.3. Model 3

The model based on figure 5. The number of available computers is changed in the time, and the computers are totally dependable (i.e. the probability of non-execution: $p = 0$). The results showed on figure 8. and 9.

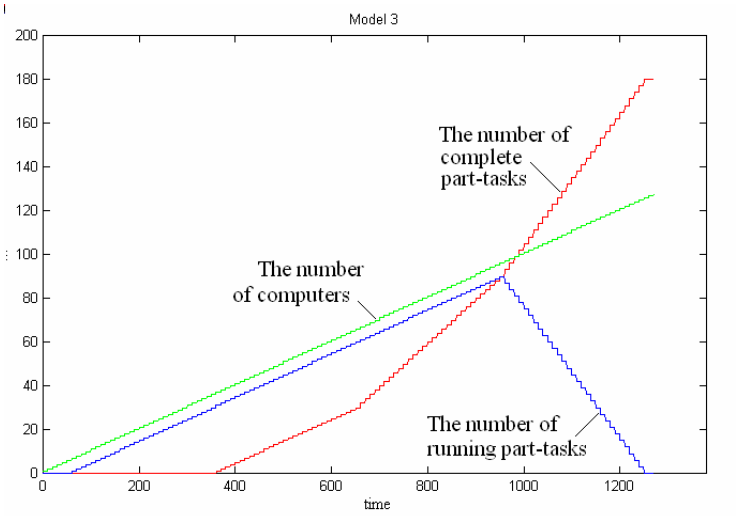


Fig. 8. At the beginning the number of computers is 1, increased linearly, the number of complete part tasks increasing by periodically increased degree because of the computers, which completed one part-task called on new part-tasks.

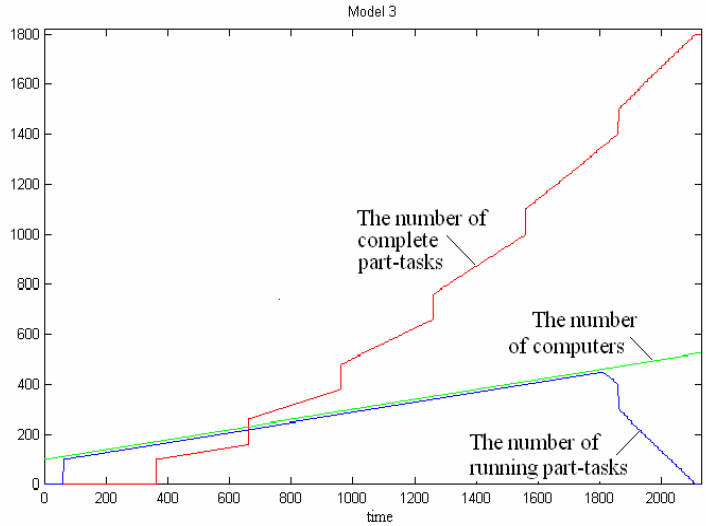


Fig. 9. At the beginning the number of computers is 100, increased linearly, the number of complete part tasks increasing by periodically increased degree by suddenly jumping between the periods.

4.4. Model 4

Model 4.4 is an improvement of model 4.3. The number of available computers is changed in the time, and the computers are not totally dependable, the non-completion is possible.

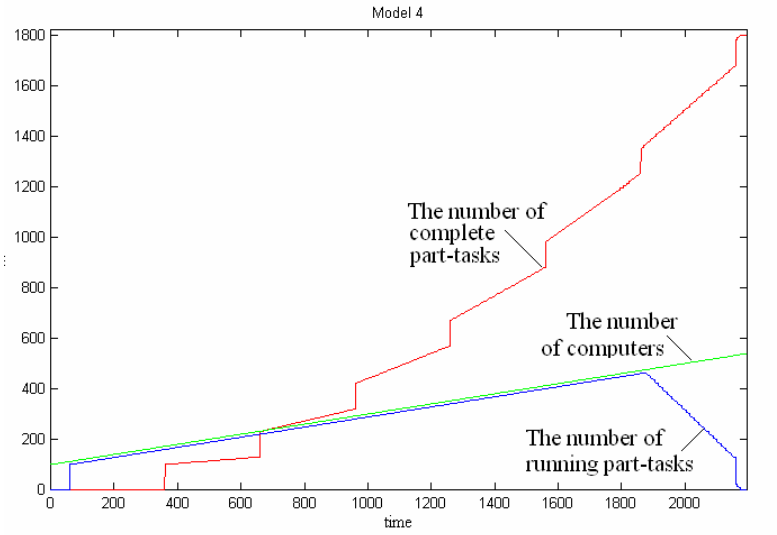


Fig. 10.

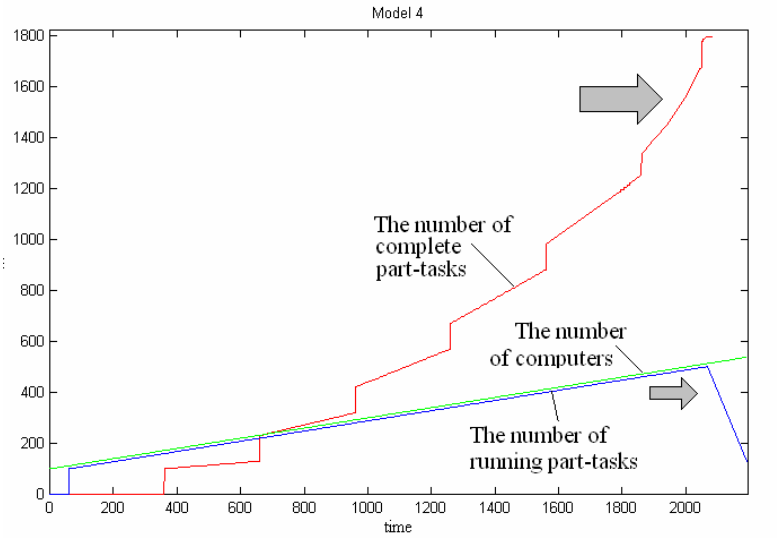


Fig. 11.

The non-completions delayed the completion time of full task of course.
The Figure 10. shows the effect of strategy when the part-task-distribution process distributed (and re-distributed) the part-tasks for only one-one computers.

Figure 11. illustrates the effect of the distribution strategy in which the task-distribution algorithm gives a part-task one of each computer. It means that in the nearing the end of project the still non-complete part task will be multiply distributed for more computers. Therefore the probability of non-execution of a specified part-task decreased significantly. By doing so the time-requirements of full project may decreased. This distribution strategy is good when the cost of computers is not important (for example when we use the free capacity of computers) or when the decreasing of project time is very important. The main differences between the situations showed on Figure 10. and Figure 11. are marked by arrows.

5. Conclusions and future perspectives

The numerical analysis and the simulation of data-intensive distributed systems gives a good help to find the quite good distribution strategy. At present we showed some relatively simple simulations. The simulation-algorithms give a possibility to use more variables and functions for modeling the changing of the number of computers, for the manipulations with free capacity of computers, modeling the data distributions and use of distributed data. The simulation of the effect of network traffic, traffic jams and a failure for distributed systems seems to be difficult. In the future we focus on these areas too and we wish improve the model illustrated on Figure 4. It is true that the capacity of networks keeps growing but the quantity of data and the size of data-intensive projects will bee growing too, we assume that finding of the good distributions will be important in the future too.

6. Additional Reading

1. Agha, G.A.: A model of concurrent computation in distributed systems. MIT Press, Cambridge, 1986.
2. Bar-Noy, A., Dolev, D., Koller, D., Peleg, D.: Fault-tolerant critical section management in asynchronous environments. Inform. Comput., 95(1):1-20, 1991
3. Ben-Ari, M.: Principles of Concurrent Programming. Prentice-Hall, Engelwood, 1982.
4. Benczúr, A., Buza, A.: Modelling of response time of distributed data intensive computing systems. ARGESIM Report no.24. Vienna, Austria, 2003, 1331-1338
5. Buza, A., Kis, P.: The Third Period of Information technology, The Sixth Fenno-Ugric Symposium on Software Technology, Tallin, Estonia, 1999, 269-275.
6. 9. Foster, I. (editor), The Grid: Blueprint of a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., San Francisco, USA, 1999.
7.Lamport, L., Lynch, N.:Distributed Computing: Models and methods. In Handbook of Theoretical Computer Science, 1157-1199, North-Holland, New York, 1990.
8.Lynch, N., A., Distributed Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, USA, 2000.
9.Milner, R.: Communication and Concurrency. Prentice-Hall, United Kingdom, 1989.
10.Tannenbaum, A. S.: Computer Networks, Prentice-Hall, 1997
11.Tel, G.: Introduction to distributed algorithms. Cambridge Univ. Press, Cambridge, 2001.